大数据技术报告（二）

姓名 金家耀

学号 1193210320

班级 人工智能 2103

学院 人工智能与计算机学院

指导老师 朱书伟

2023 年 10 月 21 日

目录

Multi-level Feature Learning for Contrastive Multi-view Clustering 阅读与理解

1 实验要求

为了让大家通过课后作业的实践加深对课堂知识的理解，也为平时成绩提供更多的依据，安排本周作业如下，其中二选一即可，选题没有高低之分。

1. 任意挑选数据（如：UCI 数据集、sklearn 包自带的数据集详见课本第二章介绍、或者下面这个 github 网站里挑选数据：https://github.com/milaan9/Clustering-Datasets）对比 K-means 和 DBSCAN 算法的性能（如何设置参数，如何比较聚类效果等内容自由发挥），做一个分析报告。并且，报告主要包含：数据介绍与提供数据来源、对算法的基本理解、主要代码的介绍、实验结果的分析（可配合可视化分析）等。

2. 结合 github 提供的源代码学习当前先进的聚类方法，结合源代码自由挑选数据运行算法，并给出自己对算法的理解和对结果的分析报告，报告主要包含内容同上。

2 多视图聚类介绍

多视图聚类（MVC）在近几年引起了越来越多的注意，MVC 主要分为传统 MVC 方法和深度学习 MVC 方法。传统的 MVC 方法分为三类，分别是子空间聚类、矩阵分解方法、图方法；深度 MVC 方法大致可以分为两步法和一步法。

而传统方法和深度学习方法都有各自的缺点：传统方法由于其表现能力差、计算复杂等原因导致其性能不佳；而先前的深度学习方法总是把多种视图的特征进行融合，这可能导致了一些噪声本存在与某一个视图的特征在后续的模型训练中起主导作用，最终导致其性能达不到高水平要求。

3 MFLVC 介绍

Multi-level Feature Learning for Contrastive Multi-view Clustering 发表在 CVPR-2022 上，MFLVC 为解决（1）无意义的某些视图中的私有信息在特征融合后产生影响。（2）重构隐空间与对比学习产生矛盾。MFLVC 主要做出了以下贡献：

- 设计了一个不需要融合各个视图特征的网络，用来解决上述的问题（1）；

- 提出了多视图之间的一致性网络，用来训练高级特征与语义标签之间的一致性，利用高级特征来提高语义标签的质量；
- 由于设计良好的框架，方法对超参数的设置具有鲁棒性。进行了详细的消融研究，包括损失成分和对比学习结构，以理解所提出的模型。大量的实验表明，达到了最先进的聚类效果。

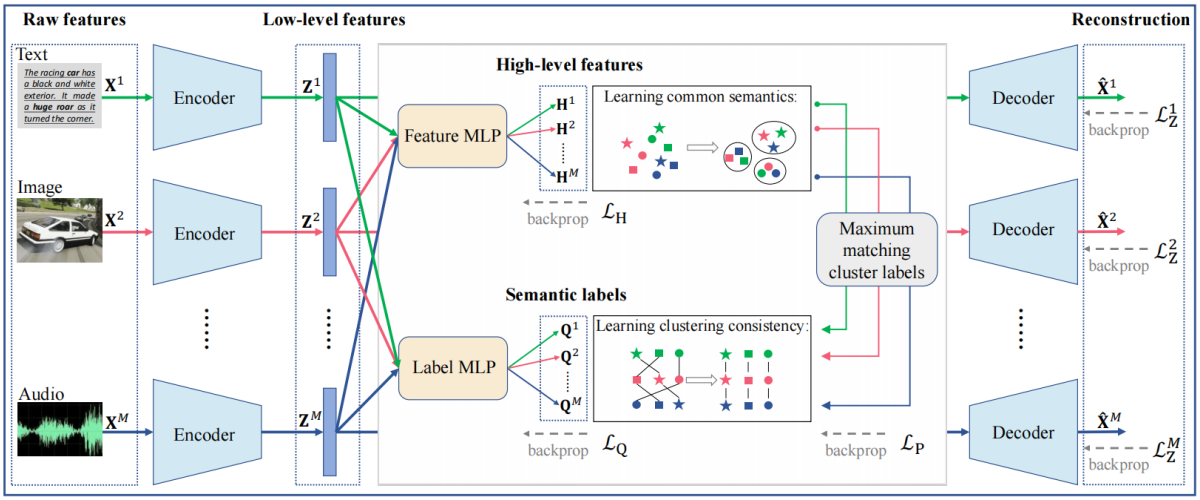


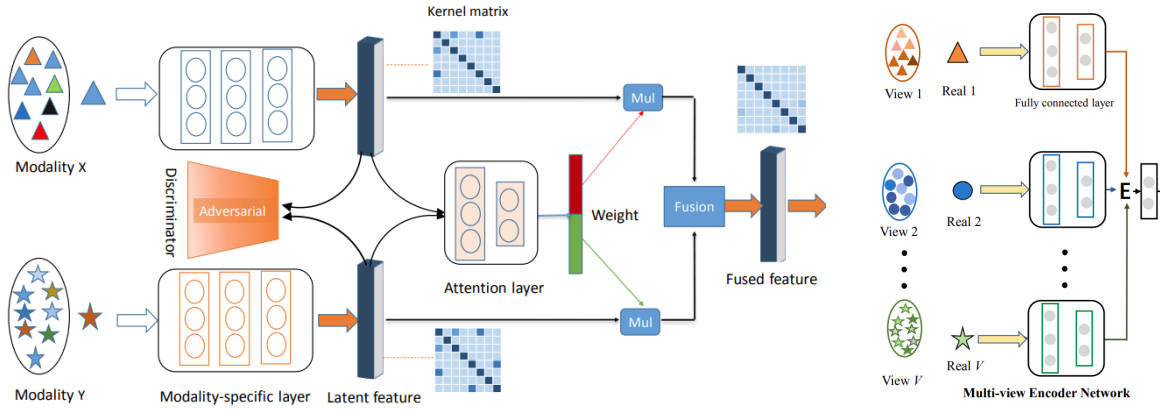
图 1: MFLVC 框架图

如图 1所示，为 MFLVC 模型主要框架图。大致过程为，利用 Encoder 将各个多视图的信息 $\{\mathbf{X}^m\}_1^M$ 编码成隐空间向量 $\{\mathbf{Z}^m\}_1^M$，再经过 Decoder 将隐空间向量解码成与 $\{\widehat{\mathbf{X}}^m\}_1^M$，这里引出 $\mathcal{L}_Z$。同时，利用隐空间向量 $\{\mathbf{Z}^m\}_1^M$，经过 Feature MLP 以及 Label MLP 构造出高级特征向量 $\{\mathbf{H}^m\}_1^M$ 和语义标签 $\{\mathbf{Q}^m\}_1^M$，再利用对比学习一致性学习来约束 $\{\mathbf{Z}^m\}_1^M$ 和 $\{\mathbf{H}^m\}_1^M$，这里引出 $\mathcal{L}_H$ 和 $\mathcal{L}_Q$，最后利用 K_means 算法对 $\{\mathbf{H}^m\}_1^M$ 进行聚类并利用 $\{\mathbf{H}^m\}_1^M$ 计算出的语义标签对其结果进行调整。

3.1 自动编码器（AutoEncoder）

原文中介绍使用 AE 的动机是因为一些原始样本中特征往往带有噪声，为了在同一视图中尽量减少这种影响，使用 AE 来提取普遍的特征模式。据我个人理解，使用 AE 不仅能够提取普遍的特征，还可以进行特征下采样降维，使得特征维度减少，运算复杂度变低。

具体而言，文章中训练了一个编码器 $E^m(\mathbf{X}^m;\theta^m)$ 和解码器 $D^m(\mathbf{Z}^m;\phi^m)$，这里的 θ^m 和 ϕ^m 是神经网络中的参数（原文中描述网络时，往往都会跟上一个训练参数），而 $\mathbf{z}_i^m = E^m(x_i^m) \in \mathbb{R}^L$ 是在第 m 视图第 i 个样本下 L 维的隐空间向量，$\widehat{\mathbf{x}}_i^m = D^m(\mathbf{z}_i^m)$ 为通过解码器解码之后的原始样本空间的重构样本特征。为了更好地提升隐空间向量的质量，所有视图中重构损失为：



(a) EAMC 主要框架图　　(b) DAMC 主要框架图

图 2: 特征融合的两个示例图

$$\mathcal{L}_{\mathbf{Z}} = \sum_{m=1}^{M} \mathcal{L}_{\mathbf{Z}}^{m} = \sum_{m=1}^{M} \sum_{i=1}^{N} \|\mathbf{x}_i^m - D^m(E^m(\mathbf{x}_i^m))\|_2^2. \tag{1}$$

原文中提到，经过了 AE 后，如图 2所示，先前的很多论文都会把不同视图下的隐空间向量 $\{\mathbf{Z}^m\}_1^M$ 进行融合，目的是想要让模型提取出不同视图中共同的特征。但效果往往不如人意，是因为前面提到的这样融合会导致某些视图的私有特征可能会误导神经网路，进而使网络性能较差。另外，这样的融合是为了提取所有视图中的共同特征，而 AE 的目的又是保留其不同视图中的不同的私有特征，这样会使两个网络训练相互矛盾，导致性能较差。

Code Listing 1: 自动编码器代码

```
    class Encoder(nn.Module):
    def __init__(self, input_dim, feature_dim):
        super(Encoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, 500),
            nn.ReLU(),
            nn.Linear(500, 500),
            nn.ReLU(),
            nn.Linear(500, 2000),
            nn.ReLU(),
            nn.Linear(2000, feature_dim),
        )

    def forward(self, x):
        return self.encoder(x)

```

```
17
18 class Decoder(nn.Module):
19     def __init__(self, input_dim, feature_dim):
20         super(Decoder, self).__init__()
21         self.decoder = nn.Sequential(
22             nn.Linear(feature_dim, 2000),
23             nn.ReLU(),
24             nn.Linear(2000, 500),
25             nn.ReLU(),
26             nn.Linear(500, 500),
27             nn.ReLU(),
28             nn.Linear(500, input_dim)
29         )
30
31     def forward(self, x):
32         return self.decoder(x)
```

上述代码为自动编码器的模型代码，训练过程大致就是读入样本先进行自动编码，再进行自动解码，最后进行 Loss 计算，迭代网络反向传播。

3.2 多视图对比学习（Multi-view Contrastive Learning）

1. 对比学习

对比学习（Contrastive Learning）是一种机器学习方法，旨在通过将数据样本与其他样本进行比较，从而学习有关数据的有用表示。对比学习通常用于无监督或自监督学习任务，其中没有明确的标签或类别信息可供模型学习。它的核心思想是通过使相似的样本更加接近，不相似的样本更加分散，从而促使模型学习到数据的有用特征。

对比学习的一般工作流程包括以下步骤:

- 选择一对或一组数据样本，通常称为”正样本”（positive samples）和”负样本”（negative samples）。
- 通过某种方式编码或嵌入这些样本，将它们映射到一个特征空间中。
- 通过一定的损失函数，鼓励正样本之间的特征表示更加接近，而负样本之间的特征表示更加分散。
- 模型通过训练逐渐调整特征表示，以更好地区分相似和不相似的样本。

对比学习的一个典型应用是图像或文本表示学习，其中模型被训练以将相似的图像或文本映射到相邻的特征空间位置，而不相似的样本则映射到较远的位置。这些学到的表示可以用于各种任务，例如图像检索、文本分类、目标检测和推荐系统等。

最近，对比学习在深度学习领域中引起了广泛的兴趣，因为它已经在许多领域取得了出色的结果，特别是在自监督学习任务中。通过学习有用的表示，对比学习有助于克服数据稀缺和标签不足的问题，从而在各种应用中提高了性能。

2. 多视图对比学习

由于低维度的隐空间向量 $\{\mathbf{Z}^m\}_1^M$ 融合了公共的语义信息和私有的空间信息，所以需要高级的特征 $\{\mathbf{H}^m\}_1^M$，其中 $\mathbf{h}_i^m \in \mathbb{R}^H$ 和语义信息特征 $\{\mathbf{Q}^m\}_1^M$，其中 $\mathbf{q}_i^m \in \mathbb{R}^K$，K 表示样本总类别数。

对于通过多层感知机学习出高级的特征 $\{\mathbf{H}^m\}_1^M$，我们希望不同的视图中同一个样本之间的特征应该越接近越好，即正样本共有 $M-1$，所有试图中不同的样本之间的特征应该越远越好，即负样本共有 $M(N-1)$。原文中利用余弦相似度来计算两个样本（分视图）之间的特征距离：

$$d(\mathbf{h}_i^m, \mathbf{h}_j^n) = \frac{\langle \mathbf{h}_i^m, \mathbf{h}_j^n \rangle}{\|\mathbf{h}_i^m\| \|\mathbf{h}_j^n\|}, \tag{2}$$

那么 m 和 n 视图下的高级特征 $\mathbf{H}^m$，$\mathbf{H}^m$ 之间的特征损失为：

$$\ell_{fc}^{(mn)} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{d(\mathbf{h}_i^m, \mathbf{h}_i^n)/\tau_F}}{\sum_{j=1}^{N} \sum_{v=m,n} e^{d(\mathbf{h}_i^m, \mathbf{h}_j^v)/\tau_F} - e^{1/\tau_F}}, \tag{3}$$

其中 τ_F 表示温度系数。所以高级特征一致性训练损失定义为：

$$\mathcal{L}_{\mathbf{H}} = \frac{1}{2} \sum_{m=1}^{M} \sum_{n \neq m} \ell_{fc}^{(mn)}. \tag{4}$$

Code Listing 2: 高级特征对比学习代码

```
def forward_feature(self, h_i, h_j):
    N = 2 * self.batch_size
    h = torch.cat((h_i, h_j), dim=0)

    sim = torch.matmul(h, h.T) / self.temperature_f
    sim_i_j = torch.diag(sim, self.batch_size)
    sim_j_i = torch.diag(sim, -self.batch_size)

    positive_samples = torch.cat((sim_i_j, sim_j_i), dim=0).reshape(N
        , 1)
    mask = self.mask_correlated_samples(N)
    negative_samples = sim[mask].reshape(N, -1)

    labels = torch.zeros(N).to(positive_samples.device).long()
    logits = torch.cat((positive_samples, negative_samples), dim=1)
```

```
    loss = self.criterion(logits, labels)
    loss /= N
    return loss
```

通过多层感知机学习出的 $\{\mathbf{Q}^m\}_1^M$，在现实场景中，某些视图下的标签信息可能会由于私有信息而被误导，导致其标签信息的错误，为了提高聚类的鲁棒性，在语义标签 $\{\mathbf{Q}^m\}_1^M$ 中需要进行对比学习。

语义标签的对比学习与高级特征的对比学习一致，故公式为：

$$\ell_{lc}^{(mn)} = -\frac{1}{K}\sum_{j=1}^{K}\log\frac{e^{d(\mathbf{Q}_{\cdot j}^m,\mathbf{Q}_{\cdot j}^n)/\tau_L}}{\sum_{k=1}^{K}\sum_{v=m,n}e^{d(\mathbf{Q}_{\cdot j}^m,\mathbf{Q}_{\cdot k}^v)/\tau_L}-e^{1/\tau_L}}, \tag{5}$$

其中 τ_L 表示温度系数。所以语义一致性训练损失定义为：

$$\mathcal{L}_{\mathbf{Q}} = \frac{1}{2}\sum_{m=1}^{M}\sum_{n\neq m}\ell_{lc}^{(mn)} + \sum_{m=1}^{M}\sum_{j=1}^{K}s_j^m\log s_j^m, \tag{6}$$

其中，$s_j^m = \frac{1}{N}\sum_{i=1}^{N}q_{ij}^m$。上述公式的第二部分是一个正则项，为了防止在迭代过程中所有的样本都聚为一类。

Code Listing 3: 语义信息对比学习代码

```
def forward_label(self, q_i, q_j):
    p_i = q_i.sum(0).view(-1)
    p_i /= p_i.sum()
    ne_i = math.log(p_i.size(0)) + (p_i * torch.log(p_i)).sum()
    p_j = q_j.sum(0).view(-1)
    p_j /= p_j.sum()
    ne_j = math.log(p_j.size(0)) + (p_j * torch.log(p_j)).sum()
    entropy = ne_i + ne_j

    q_i = q_i.t()
    q_j = q_j.t()
    N = 2 * self.class_num
    q = torch.cat((q_i, q_j), dim=0)

    sim = self.similarity(q.unsqueeze(1), q.unsqueeze(0)) / self.
        temperature_l
    sim_i_j = torch.diag(sim, self.class_num)
    sim_j_i = torch.diag(sim, -self.class_num)

    positive_clusters = torch.cat((sim_i_j, sim_j_i), dim=0).reshape(
        N, 1)
```

```
    mask = self.mask_correlated_samples(N)
    negative_clusters = sim[mask].reshape(N, -1)

    labels = torch.zeros(N).to(positive_clusters.device).long()
    logits = torch.cat((positive_clusters, negative_clusters), dim=1)
    loss = self.criterion(logits, labels)
    loss /= N
    return loss + entropy
```

总的来说，训练的总损失函数如下：

$$\mathcal{L} = \mathcal{L}_{\mathbf{Z}} + \mathcal{L}_{\mathbf{H}} + \mathcal{L}_{\mathbf{Q}}. \tag{7}$$

3.3 高级特征语义聚类（Semantic Clustering with High-level Features）

为了利用高级特征中包含的聚类信息来提高语义标签的聚类有效性，原文中首先利用 K_means 算法在高级特征 $\{\mathbf{H}^m\}_1^M$ 中进行聚类，定义 $\{\mathbf{c}_k^m\}_{k=1}^K \in \mathbb{R}^H$ 为 K 个类别的聚类中心，并定义 $\mathbf{p}^m \in \mathbb{R}^N$ 为每个样本依据聚类之后的标签。

同时，令 $l_i^m = \text{argmax}_j\, q_{ij}^m$，即前面提到的语义信息的处理结果。然后把 l_i^m 作为锚点通过以下的最大匹配公式来优化 p_i^m：

$$\begin{aligned} &\min_{\mathbf{A}^m} \mathbf{M}^m \mathbf{A}^m, \\ &s.t. \sum_{i=1} a_{ij}^m = 1, \sum_{j=1} a_{ij}^m = 1, \\ &a_{ij}^m \in \{0,1\}, i,j = 1,2,...,K, \end{aligned} \tag{8}$$

其中，$\mathbf{A}^m$ 是一个 01 矩阵，$\mathbf{M}^m$ 定义为价值矩阵，其中 $\mathbf{M}^m = \max_{i,j} \tilde{m}_{ij}^m - \tilde{\mathbf{M}}^m$，$\tilde{m}_{ij}^m = \sum_{n=1}^N \mathbb{1}[l_n^m = i]\mathbb{1}[p_n^m = j]$，$\mathbb{1}$ 表示指示函数。

更新后的标签 $\hat{\mathbf{p}_i^m} \in \{0,1\}^K$ 为独热向量，该向量第 k 个元素为 1 当且仅当 $k = k\mathbb{1}[a_{ks}^m = 1]\mathbb{1}[p_i^m = s], k, s \in \{1,2,\ldots,K\}$，最后利用交叉熵损失函数对模型进行微调：

$$\mathcal{L}_{\mathbf{P}} = -\sum_{m=1}^{M} \hat{\mathbf{P}}^m \log \mathbf{Q}^m, \tag{9}$$

Code Listing 4: 微调模型

```
def fine_tuning(epoch, new_pseudo_label):
    loader = torch.utils.data.DataLoader(
        dataset,
        batch_size=data_size,
```

```
        shuffle=False,
    )
    tot_loss = 0.
    cross_entropy = torch.nn.CrossEntropyLoss()
    for batch_idx, (xs, _, idx) in enumerate(loader):
        for v in range(view):
            xs[v] = xs[v].to(device)
        optimizer.zero_grad()
        _, qs, _, _ = model(xs)
        loss_list = []
        for v in range(view):
            p = new_pseudo_label[v].numpy().T[0]
            with torch.no_grad():
                q = qs[v].detach().cpu()
                q = torch.argmax(q, dim=1).numpy()
                p_hat = match(p, q)
            loss_list.append(cross_entropy(qs[v], p_hat))
        loss = sum(loss_list)
        loss.backward()
        optimizer.step()
        tot_loss += loss.item()
    print('Epoch {}'.format(epoch), 'Loss:{:.6f}'.format(tot_loss /
       len(data_loader)))
```

最后，第 i 个样本的语义标签计算公式为：

$$y_i = \underset{j}{\operatorname{argmax}} \left(\frac{1}{M} \sum_{m=1}^{M} q_{ij}^m \right). \tag{10}$$

4 实验复现

4.1 数据集

文章中使用 5 个公开数据集 BDGP、Caltech-5V、Fashion、MNIST-USPS、CCV 数据集来进行验证实验。MNIST-USPS 是一个流行的手写数字数据集，它包含 5000 个样本和两种不同风格的数字图像。BDGP 包含 2500 个果蝇胚胎样本，每一个样本都由视觉和文本特征所代表。哥伦比亚消费者视频（CCV）是一个视频数据集，包含 6,773 个样本，属于 20 个类，并提供了手工制作的三个视图的字袋表示，如 STIP、SIFT 和 MFCC。Fashion 是一个关于产品的图像数据集，文章根据文献将不同的三种风格视为一种产品的三种视图。

4.2 训练

模型并不是端对端的学习，而是通过三个模块相继连接，先后训练得到。

Code Listing 5: AE 训练

```
epoch = 1
while epoch <= args.mse_epochs:
    pretrain(epoch)
    epoch += 1
```

上述代码代表的是前几个 epoch 会训练一个预训练模型（AE）来重构隐空间向量，为了给后续的模型训练带来便利。

Code Listing 6: 对比学习训练

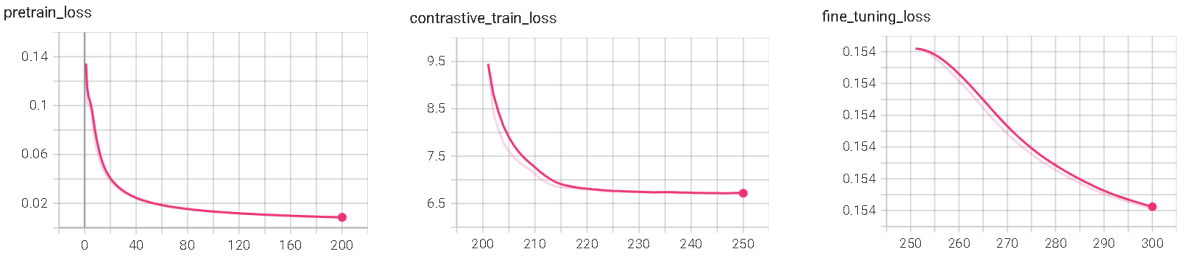
```
while epoch <= args.mse_epochs + args.con_epochs:
    contrastive_train(epoch)
    if epoch == args.mse_epochs + args.con_epochs:
        acc, nmi, pur = valid(model, device, dataset, view, data_size
            , class_num, eval_h=False)
    epoch += 1
```

上述代码代表的是后续几个 epoch 经过高级特征和标签信息的对比学习，为了使得相同的类之间的距离更近，不同的类之间的距离更远，学习一个模式来使样本聚类。

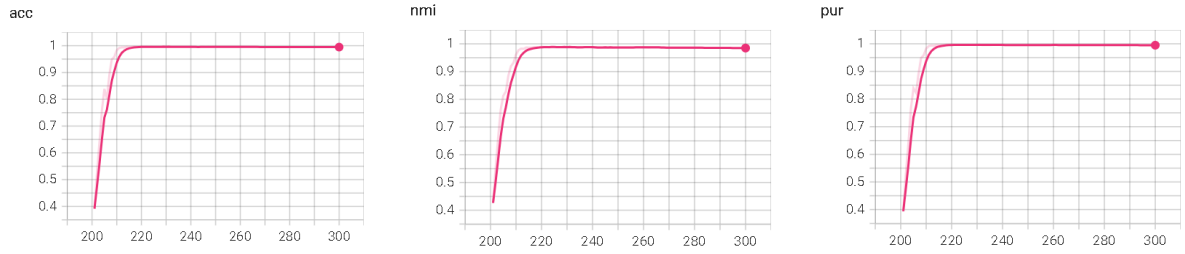
Code Listing 7: 微调模型

```
new_pseudo_label = make_pseudo_label(model, device)
while epoch <= args.mse_epochs + args.con_epochs + args.tune_epochs:
    fine_tuning(epoch, new_pseudo_label)
    if epoch == args.mse_epochs + args.con_epochs + args.tune_epochs:
        acc, nmi, pur = valid(model, device, dataset, view, data_size
            , class_num, eval_h=False)
        state = model.state_dict()
        torch.save(state, './models/' + args.dataset + '.pth')
        print('Saving..')
        accs.append(acc)
        nmis.append(nmi)
        purs.append(pur)
    epoch += 1
```

上述代码代表的是最后几个 epoch 用来微调模型，利用高级特征来引导标签信息，实现更好的聚类。



(a) 预训练损失　(b) 对比学习损失　(c) 微调损失

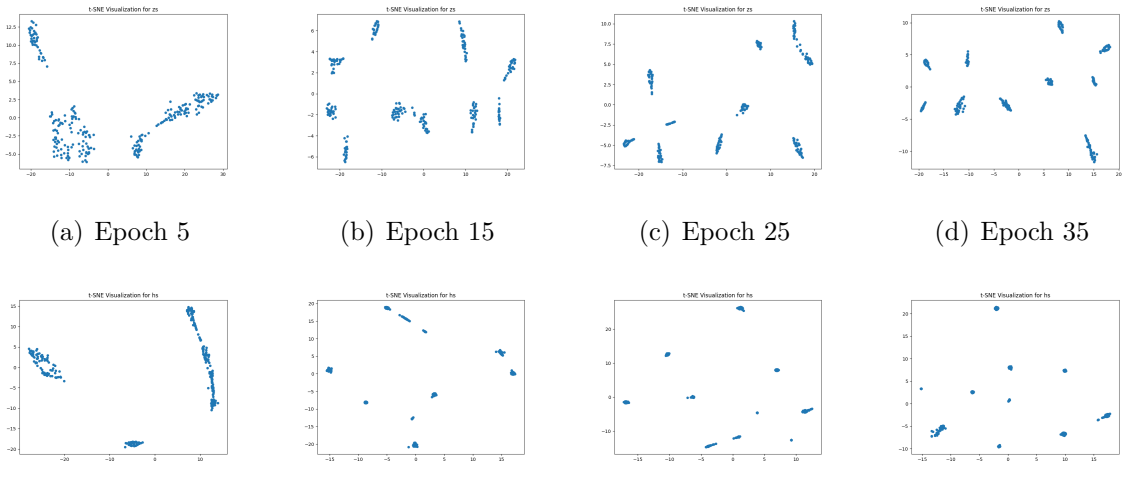


(d) 准确率　(e) NMI　(f) PUR

图 3: MNIST-USPS 训练过程

由于自身电脑的局限性，只能复现小样本的数据集训练过程。这里选择数据集 MNIST-USPS、BDGP、CCV 进行复现，如图 3所示，并通过 tensorboard 工具箱进行可视化 MNIST-USPS 数据集上训练过程展示。

从图中的趋势可以明显看出，每个模块的损失都呈下降趋势，这表明所提出的模块在小规模数据集上也表现出了良好的有效性。值得注意的是，尽管在小样本数据集上进行训练，但实验结果非常令人满意，几乎达到了百分之百的聚类效果。



(a) Epoch 5　(b) Epoch 15　(c) Epoch 25　(d) Epoch 35

(e) Epoch 5　(f) Epoch 15　(g) Epoch 25　(h) Epoch 35

图 4: (a)-(d) 表示高级特征可视化；(e)-(h) 表示低级特征可视化

如图 4所示，为高级特征和低级特征在 t-SNE 方法下的二维嵌入可视化结果，可以从图上得出，高级特征的一致性学习是非常有效的。

4.3 测试

源代码的介绍中提供了下载模型的路径，我将所有的模型下载下来，进行测试，测试结果如下：

表 1: 在不同的数据集上聚类效果表

Dataset	ACC (%)	NMI (%)	ARI (%)	PUR (%)
MNIST-USPS	99.48	98.48	98.85	99.48
BDGP	98.32	95.33	95.88	98.32
CCV	30.52	29.99	14.85	33.68
Fashion	99.38	98.37	98.64	99.38
Caltech-2V	62.00	54.30	43.50	62.00
Caltech-3V	66.79	58.42	49.64	68.36
Caltech-4V	78.43	71.06	64.42	78.43
Caltech-5V	86.21	77.01	72.65	86.21

从表中数据可以看出，模型在小数据集上表现较为良好，虽然在大数据集上有时表现不佳，但随着视图的增加，其性能也相应地提升。

5 总结

这篇研究引入了一种创新性的模型，旨在应对多视图学习领域的一系列挑战。这一模型的独特之处在于，它消除了通常需要将不同视图的特征融合在一起的需求，从而避免了潜在的私有信息干扰，提高了整体模型性能。通过消除特征融合的复杂性，该模型有望在多视图学习中提供更加清晰和稳定的解决方案。

另一个关键的创新点是该模型整合了对比学习的理念，为聚类任务引入了一种全新的方法。对比学习已经在无监督学习中表现出很大潜力，因为它可以帮助模型从数据中学到更有用的表示。这种整合为聚类任务带来了更高的稳定性和准确性，这在实际应用中可能具有重大意义。

最重要的是，这一方法的通用性，它不仅适用于多视图学习，还可以在各种领域中应用。无论是在图像处理、文本分析还是推荐系统等领域。