BOOSTING UNSUPERVISED CONTRASTIVE LEARNING USING DIFFUSION-BASED DATA AUGMENTATION FROM SCRATCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised Contrastive learning has gained prominence in fields such as vision, natural language processing, and biology, leveraging predefined positive and negative samples for representation learning. Data augmentation, categorized into hand-designed and model-based methods, has been identified as a crucial component for enhancing contrastive learning. However, hand-designed methods require human expertise in domain-specific data while sometimes distorting the meaning of the data. In contrast, model-based methods, such as generative models, usually require supervision or large-scale external data. To address the problems presented above, this paper proposes DiffAug, a novel unsupervised contrastive learning technique with diffusion mode-based positive data generation. DiffAug consists of a semantic encoder and a conditional diffusion model; the conditional diffusion model generates new positive samples conditioned on the semantic encoding to serve the training of unsupervised contrast learning. With the help of iterative training of the semantic encoder and diffusion model, DiffAug improves the representation ability in an uninterrupted and unsupervised manner. Experimental evaluations show that DiffAug outperforms hand-designed augmentation and classical representation learning methods in classification and clustering tasks on visual and biological datasets, highlighting its potential for generalizing unsupervised learning techniques. The code for review is released at https://anonymous.4open.science/r/diffaug_review-804E.

1 INTRODUCTION

Contrastive learning, as shown by many studies (He et al., 2020; Chen et al., 2020; Cui et al., 2021; Wang & Qi, 2022; Assran et al., 2022; Zang et al., 2023), has become important in areas like vision (He et al., 2021; Zang et al., 2022b), natural language processing (Rethmeier & Augenstein, 2023), and biology (Yu et al., 2023; Krishnan et al., 2022). It learns representations using predefined positive and negative samples. Many studies (Tian et al., 2020; Zhang & Ma, 2022; Peng et al., 2022; Zhang et al., 2023b) have found that data augmentation helps contrastive learning by making it more robust and preventing model problems.

Data augmentation falls into two main types: *hand-designed methods* and *model-based methods* (Xu et al., 2023). Hand-designed methods require humans to understand the meaning of the data and then change the input features while maintaining or extending that meaning. In visual tasks, methods such as color change (Yan et al., 2022), random cropping (Cubuk et al., 2020), and rotation (Maharana et al., 2022) are used to aid in contrastive learning. However, the problem is that the above techniques must be more data-specific. For some data (genes or proteins or others), it isn't easy to visualize the data due to the complexity of its meaning. Consequently, it isn't easy to design a good augmentation strategy. Semantics-independent augmentation methods such as adding noise (Huang et al., 2022) and random hiding (Theodoris et al., 2023) are used, but only sometimes with significant results. Another problem with hand-designed methods is that they do not smoothly change the semantics of the data, e.g., a slight change in the magnitude of the magnification in a random cropping of an image may imply the risk of a semantic mutation (in Fig. 1). As a result, many positive and negative samples are needed to distribute these risks to obtain a stable representation. At the same time, it is

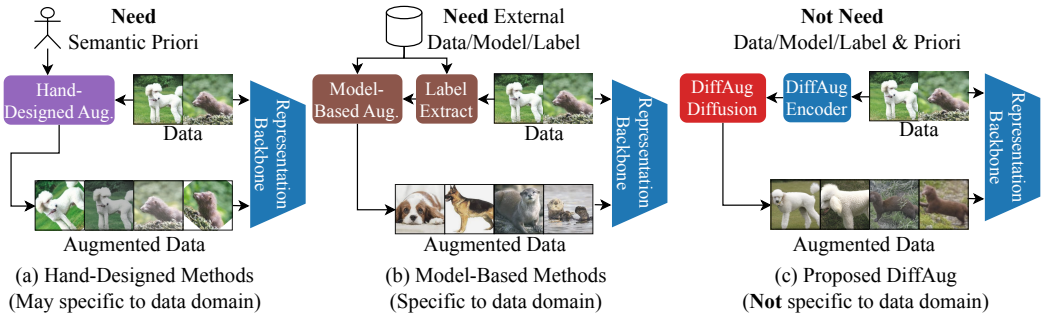


Figure 1: **Comparison of DiffAug with other methods.** (a) Hand-designed augmentation methods are based on human a priori design strategies that produce new data that differ in input features and are semantically similar. (b) Model-based augmentation methods are mainly based on generative models that produce new data with the same labels with the help of a large amount of data, labels, or pre-trained models. Such methods tend to be specific to the data domain. (c) DiffAug attempts to reduce the dependence on external data and prior knowledge through iterative training with encoders and diffusion. Expanding the application areas of unsupervised contrastive learning.

also challenging to train contrastive learning models with fewer samples for certain domains where data acquisition is costly, such as biology.

Given the challenges mentioned earlier, model-based methods (generative models) based on deep learning are used to create better data. In the vision domain, techniques using VAE (Kingma & Welling, 2014), GAN (Goodfellow et al., 2014), and diffusion models (Ho et al., 2020; Nichol & Dhariwal, 2021; Saharia et al., 2022; Nichol et al., 2022; Ramesh et al., 2022) have been developed to improve model training. For supervised learning, several studies have received attention. Du et al. (2023) proposed the DREAM-OOD framework, which uses diffusion models to generate photo-realistic outliers from in-distribution data for improved OOD detection. Zhang et al. (2023a) developed the Guided Imagination Framework (GIF) using generative models like DALL-E2 and Stable Diffusion for dataset expansion, enhancing accuracy in both natural and medical image datasets. However, there are concerns about these methods, especially about their diversity and how well they generalize. The detailed related works are in Appendix.A. Moreover, most of these generative models are trained with supervision or need much external data. This makes them less suitable for areas like gene and protein data (in Fig. 1). This leads to an important question: **Is it possible to design a data augmentation framework to enhance unsupervised contrastive learning in different domains without requiring expert knowledge or additional data?**

We introduce DiffAug, a novel unsupervised contrastive learning technique with diffusion mode-based positive data generation to address the posed question. Explicitly designed for unsupervised contrastive learning, DiffAug eliminates the need for training labels. Instead, we employ a semantic estimator to gauge the semantics of the input data, subsequently guiding the augmentation process. At its core, DiffAug operates through two synergistic modules: a semantic encoder and a diffusion generator. Utilizing a soft contrastive loss, the semantic encoder crafts latent representations that act as guiding vectors for the diffusion generator. This generator then methodically produces augmented data in the input space, ensuring varying levels of semantic consistency based on the guiding vectors and specific adjustable hyperparameters.

In our experiments, we thoroughly evaluated our method on both visual and biological datasets. Our findings indicate that the proposed method can produce sensible data augmentations, subsequently enhancing the performance of unsupervised contrastive learning that utilizes these augmentations. Notably, DiffAug performs superior classification and clustering tasks compared to all benchmark methods. The primary contributions of this paper are: (a) We introduce DiffAug, a novel unsupervised contrastive learning technique with diffusion mode-based positive data generation. DiffAug's data augmentation replaces traditional domain-specific hand-designed data augmentation strategy. (b) DiffAug operates independently of external data or manually designed rules. Its versatility allows for integration with various models, encompassing domains like vision or biology studies. (b) The experimental results show the efficacy of DiffAug in enhancing the performance of contrastive

learning in different tasks. This suggests that DiffAug can generate positive sample data unsupervised, which in turn promotes the development of unsupervised learning techniques.

2 How to Build Unsupervised Conditional Generative Models

In the context of unsupervised data augmentation, the training dataset providing potential semantic categories is denoted as $\mathcal{D}_t = \{\mathbf{x}_i\}_{i=1}^N$, where N is the size of the training set. To boost the training efficiency of unsupervised contrastive learning with positive samples generated by the diffusion model, a novel framework called DiffAug is proposed.

2.1 Preliminaries of Contrastive Learning and Soft Contrastive Learning

Contrastive Learning. Contrastive learning learns visual representation via enforcing the similarity of the positive pairs and enlarging distance of negative pairs. Formally, loss is defined as,

$$\mathcal{L}_{\text{cl}} = -\log \mathcal{Q}\left(\mathbf{z}_i, \mathbf{z}_i^+\right) + \log \left[\mathcal{Q}\left(\mathbf{z}_i, \mathbf{z}_i^+\right) + \sum_{\mathbf{z}_i^- \in V^-} \mathcal{Q}\left(\mathbf{z}_i, \mathbf{z}_i^-\right) \right] \tag{1}$$

where $\mathbf{z}_i$ is the low dimensional embedding $\mathbf{z}_i = \text{Enc}^{\text{cl}}(\mathbf{x}_i)$, $Q\left(\mathbf{z}_i, \mathbf{z}_i^+\right)$ indicates the similarity between positive pairs while $Q\left(\mathbf{z}_i, \mathbf{z}_i^-\right)$ is the similarity between negative pairs. For the traditional scheme, in the computer vision domain, data augmentation methods such as random cropping (Cubuk et al., 2020) or data mixup (Zhang et al., 2017) are used to generate new positive data. The negative samples v_i^- are sampled from negative distribution V^-.

Soft Contrastive Learning. To address the performance degradation due to view noise in contrastive learning and to accomplish unsupervised learning on smaller scale datasets, Zang et al. (2023) designed soft contrastive learning, which smoothes sharp positive and negative sample pair labels by evaluating the credibility of the sample pairs. Consider the loss form for multiple positive samples and multiple negative samples as,

$$\begin{aligned}
&\mathcal{L}_{\text{scl}}(\mathbf{y}_c, \mathbf{y}_j, \mathbf{z}_c, \mathbf{z}_j) = -\sum_{j=1}^{\mathcal{B}} \{ \underbrace{\mathcal{P}(\mathbf{y}_c, \mathbf{y}_j)}_{\text{soft positive aim}} \log\left(\mathcal{Q}(\mathbf{z}_c, \mathbf{z}_j)\right) + \underbrace{(1 - \mathcal{P}(\mathbf{y}_c, \mathbf{y}_j))}_{\text{soft negative aim}} \log\left(1 - \mathcal{Q}(\mathbf{z}_c, \mathbf{z}_j)\right) \}, \\
&\mathcal{P}(\mathbf{a}, \mathbf{b}) = \left(1 + \mathcal{H}_{ij}\left(e^{\beta} - 1\right)\right) \mathcal{Q}(\mathbf{a}, \mathbf{b}),
\end{aligned} \tag{2}$$

where the $\mathbf{y}_i, \mathbf{z}_i$ are the high dimensional embedding and low dimensional embedding $\mathbf{y_i}, \mathbf{z_i} = \text{Enc}(\mathbf{x_i})$. The $\mathcal{P}(\mathbf{a}, \mathbf{b})$ is soft learning weight and calculated by the positive/negative pair indicator $\mathcal{H}_{cj}$. The hyper-parameter $\beta \in [0, 1]$ introduces prior knowledge of data augmentation relationship $\mathcal{H}_{cj}$ into the model training. Details of contrastive and soft contrastive learning are in Appendix. B.

2.2 DiffAug Design Details and Training Strategies

DiffAug Framework. DiffAug accomplishes the tasks of *positive sample generation* and *data representation* by iterating the two modules over each other (in Fig. 2). DiffAug consists of two main modules, a semantic encoder $\text{Enc}(\cdot|\theta)$ and a diffusion generator $\text{Gen}(\cdot|\phi)$, where θ and ϕ are model parameters. The $\text{Enc}(\cdot|\theta)$ maps the input data $\mathbf{x}_i$ to the discriminative latent space $\mathbf{v}_i$, and the generator $\text{Gen}(\cdot|\phi)$ generates new data with a semantic vector $\mathbf{v}_i$. Similar to the Expectation maximization algorithm (Gupta et al., 2011), the semantic encoder $\text{Enc}(\cdot|\theta)$ and the diffusion generator $\text{Gen}(\cdot|\phi)$ are trained in turn by two different loss functions (see Fig. 2(a) and Fig. 2(b)).

Semanticity Modeling (E-Step). In the semanticity modeling step, given a central data $\mathbf{x}_c$, we generate a background set $\mathcal{B}_c$,

$$\mathcal{B}_c = \{\mathbf{x}_1, \cdots, \mathbf{x}_j, \cdots, \mathbf{x}_{\mathcal{N}_b}\}, \begin{cases} \mathbf{x}_j \sim \mathcal{D}_t & \text{if} \quad \mathcal{H}_{cj} = 0 \\ \mathbf{x}_j \sim A_{\text{ug}}(\mathbf{x}_c) & \text{if} \quad \mathcal{H}_{cj} = 1 \end{cases} \tag{3}$$

where N_b is the number of background data points. The $\mathcal{H}_{cj} = 0$ indicates $\mathbf{x}_j$ is sampled from the dataset $\mathcal{D}_t$, and $\mathbf{x}_c$ and $\mathbf{x}_j$ are negative pair. Meanwhile, $\mathcal{H}_{cj} = 1$ indicates $\mathbf{x}_c$ and $\mathbf{x}_j$ are positive

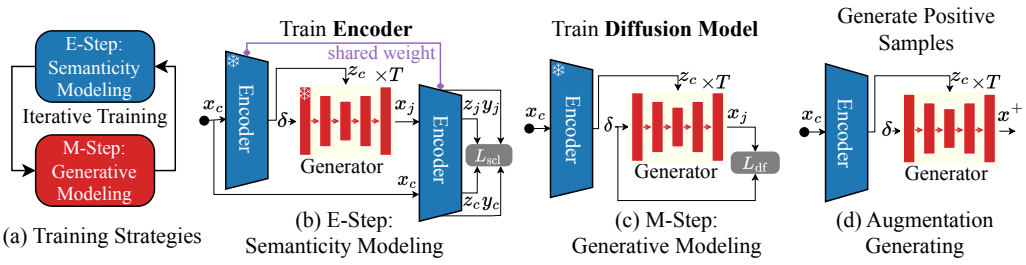


Figure 2: **The DiffAug framework and training strategy.** DiffAug includes a semantic encoder $\text{Enc}(\cdot|\theta)$ and a diffusion generator $\text{Gen}(\cdot|\phi)$. (a) shows how $\text{Enc}(\cdot|\theta)$ and $\text{Gen}(\cdot|\phi)$ are interative trained. (b) and (c) show how to calculate the loss functions. (d) shows how to generate new augmentation data with the trained model. The snowflake (❄) indicates that the module is frozen.

pair and $\mathbf{x}_j$ is sampled from data augmentation. For details, new positive data are generated by the diffusion model according to DDMP (Ho et al., 2020),

$$\mathbf{x}_j = \text{Gen}(\delta, \mathbf{z}_c|\phi^*), \mathbf{y}_c, \mathbf{z}_c = \text{Enc}(\mathbf{x_c}|\theta^*), \tag{4}$$

where $\text{Gen}(\delta, \mathbf{z}_c|\phi^*)$ is the generation process of the diffusion model, and the generating details are in Eq. (7). The $\delta \sim \mathcal{N}(0, \mathbf{1})$ is the random initialized data, and $\mathbf{z}_c$ is a conditional vector. The $*$ in ϕ^* and θ^* means the parameter is frozen. To avoid unstable positive samples from untrained generative models, training starts exclusively with traditional data augmentation tools, and then, the data generated by DiffAug is replaced with data generated by DiffAug, with a replacement probability of the hyperparameter λ, an oversized λ introduce toxicity, which we will discuss in Sec. 3.5. We update the parameter of the semantic encoder with the soft contrastive learning loss,

$$\theta = \theta - \eta \sum_{\mathbf{x}_j \in \mathcal{B}_c} \{\mathcal{L}_{\text{scl}}(\mathbf{y}_c, \mathbf{z}_c, \mathbf{y}_j, \mathbf{z}_j)\}, \text{ where } \mathbf{y}_j, \mathbf{z}_j = \text{Enc}(\mathbf{x}_j|\theta), \tag{5}$$

where the η is the learning rate, and the $\mathcal{L}_{\text{scl}}(\mathbf{y}_c, \mathbf{z}_c, \mathbf{y}_j, \mathbf{z}_j)$ is in Eq. (2).

Generative Modeling (M-Step). In the generative modeling step, the conditional diffusion generator $\text{Gen}(\cdot|\phi)$ is trained by the vanilla diffusion loss $\mathcal{L}_{\text{df}}(\mathbf{x}_c, \mathbf{z}_c|\phi)$ (Ho et al., 2020),

$$\phi = \phi - \eta \sum_{t=1}^{T} \left\{ \left\| \delta - g_\phi \left(\sqrt{\bar{\alpha}_t} \widetilde{\mathbf{x}}_c^t + \sqrt{1-\bar{\alpha}_t}, t, \mathbf{z}_c \right) \right\|_2^2 \right\}, \tag{6}$$

where the conditional vector $\mathbf{z}_c$ is generated from the semantic encoder in Eq.(4). The $g_\phi(\cdot)$ is the conditional diffusion neural network. The α_t is the noise parameter in the diffusion process, and $\bar{\alpha}_t = 1 - \alpha_t$. The $\widetilde{\mathbf{x}}_c^t$ is the intermediate data in the diffusion process, and the $\widetilde{\mathbf{x}}_c^0 = \mathbf{x}_c$. T is the time step of the generation process. When $g_\phi(\cdot)$ is trained, the detailed generating process is,

$$\text{Gen}(\delta, \mathbf{z}_c|\phi^*) = \left\{ \widetilde{\mathbf{x}}^0 \mid \widetilde{\mathbf{x}}^{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\widetilde{\mathbf{x}}^t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} g_\phi(\widetilde{\mathbf{x}}^t, t, \mathbf{z}_c^*) \right) + \sigma_t \mathcal{N}(0,1), t \in \{T, \cdots, 1\} \right\}, \tag{7}$$

The $g_\phi(\cdot)$ is a neural network approximator intended to predict δ with $\widetilde{\mathbf{x}}$ and the condition vector $\mathbf{z}_c^*$.

Augmentation Generation Given the trained semantic encoder $\text{Enc}(\cdot|\theta)$ and diffusion generator $D(\cdot)$, and DiffAug generate new augmented data $\mathbf{x}_i^+$ from any input data $\mathbf{x}_i$.

$$\mathbf{x}_i^+ = \text{Gen}(\delta|\mathbf{z}_i), \quad \mathbf{y}_i, \mathbf{z}_i = \text{Enc}(\mathbf{x}_i). \tag{8}$$

Meanwhile, DiffAug's semantic encoder can be seen as a feature extractor. It is considered to have good discriminative performance because it is trained simultaneously as the diffusion generator.

3 Results

We completed experiments on both the vision dataset and the biological dataset. We want to prove that Diffaug can work effectively and bring enhancement in different domains.

Algorithm 1 The DiffAug Training Algorithm:

Input: Data: $\mathcal{D}_t = \{\mathbf{x}_c\}_{i=1}^N$, Learning rate: η, E or M State: S, Batch size: B, Network parameters: θ, ϕ,
Output: Updateed Parameters: θ, ϕ.

while $b = 0; b < [|\mathcal{X}|/B]; b$++ **do**
  $\mathbf{x}_c \sim \mathcal{D}_t$; # Sample the centering data
  $\mathbf{y}_c, \mathbf{z}_c \leftarrow \text{Enc}(\mathbf{x}_c|\theta)$; # Generate frozen condition vector
  if S==M-step **then**
    $L_1 \leftarrow L_{\text{df}}(\mathbf{x}_c, \text{SG}(\mathbf{z}_c)|\phi)$ by Eq. (6); $\phi \leftarrow \phi - \eta \frac{\partial \mathcal{L}_1}{\partial \phi}$, # Calculate diffusion loss
  else
    $\mathcal{B}_c = \{\mathbf{x}_1, \cdots, \mathbf{x}_{\mathcal{B}} | \mathbf{x}_j \sim \mathcal{D}_t$ **if** $\mathcal{H}_{ij} = 0; \mathbf{x}_j \sim \text{Aug}(\mathbf{x}_c)$ **else** $\}$; # Generate/sample data
    $\mathcal{Y} = \{\mathbf{y}_1, \cdots, \mathbf{z}_j, \cdots, \mathbf{y}_{\mathcal{B}}\}, \mathcal{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_j, \cdots, \mathbf{z}_{\mathcal{B}}\}, \mathbf{y}_j, \mathbf{z}_j = \text{Enc}(\mathbf{x}_j|\theta)$
    $L_2 \leftarrow L_{\text{scl}}(\mathcal{Y}, \mathcal{Z})$ by Eq. (2); $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}_2}{\partial \theta}$ # Calculate scl loss
  end if
end while

Table 1: **Comparison of Linear-test and KMeans clustering performance on computer vision dataset.** The contrastive learning methods, including SimCLR (Chen et al., 2020), MOCO v2 (He et al., 2020), BYOL (Grill et al., 2020), SimSiam (Chen & He, 2021), and DLME (Zang et al., 2022b) are chosen for comparison. The SimC.+Mix. and MoCo.+Mix. are SimCLR and MoCoV2 with Mixup data augmentation which processed by Zhang et al. (2022). The SimC.+Dif. and MoCo.+Mix. are SimCLR and MoCoV2 with DiffAug data augmentation. Improvements over the best baseline are shown in parentheses. The 'AVE' represents the average of the performance on all datasets. The best results are marked in **bold**. Performance gains of more than 1.0 are underlined.

	Linear-test Performance				Clustering Performance				
	CF10	CF100	STL10	TINet	CF10	CF100	STL10	TINet	AVE
SimCLR	89.8	57.4	86.9	38.4	78.2	38.6	77.2	12.9	58.6
MoCoV2	90.1	64.2	85.6	39.4	79.9	36.4	75.4	14.4	59.3
BYOL	91.0	62.7	88.7	43.8	82.6	45.6	78.5	18.8	62.6
SimSiam	90.6	63.5	84.8	44.9	79.5	42.1	82.8	18.9	62.5
DLME	91.3	66.1	90.1	44.9	83.1	44.1	88.3	18.2	64.9
SimC.+Mix.	90.9	62.9	89.6	—	83.4	43.5	87.3	—	—
MoCo.+Mix.	91.5	62.7	90.1	—	83.2	43.2	88.4	—	—
SimC.+Dif.	91.3	62.8	90.6	46.3	84.2	46.3	80.1	19.7	63.8
MoCo.+Dif.	91.4	64.1	90.5	46.2	84.1	47.9	83.6	19.9	65.3
DiffAug	**93.4**(+1.9)	**69.9**(+3.8)	**92.5**(+1.9)	**49.7**(+3.4)	**86.2**(+2.0)	**48.6**(+0.7)	**89.4**(+1.0)	**19.9**(+0.0)	**67.8**(+2.5)

3.1 DiffAug Outperforms Hand Designed Augmentations on Vision Datasets

We showcase DiffAug's versatility across multiple test protocols, emphasizing its potential to enhance vision data. We focus on data augmentation techniques that can be used for unsupervised contrastive learning in the unsupervised case. Therefore, the comparison does not include some labeling-based methods (Nichol et al., 2021; He et al., 2023; Trabucco et al., 2023; Zhang et al., 2023a). Comparative outcomes based on linear-test performance and clustering are detailed in Table 1, and the data of CF10 and CF100 is from Huang et al. (2023). Different baseline methods use different hand-designed data augmentations (in Appendix C.3).

Test Protocols. Experiments are performed on CIFAR-10 [CF10] and CIFAR-100 [CF100] (Krizhevsky et al., 2009), STL10 (Coates et al., 2011), TinyImageNet [TINet] (Le & Yang, 2015) dataset. We followed a procedure similar to SimCLR (Chen et al., 2020) for the Linear-test performance assessment. We evaluated the model's representations linearly on top of the frozen features. This ensures that the quality of the representations is attributed only to the pre-training task, without any influence from potential fine-tuning. We used the ResNet (He et al., 2015) backbone from the baseline. In contrast, for DiffAug, its semantic encoder served as the contrastive learning backbone, trained using DiffAug-augmented images. We extracted feature vectors from the models for the K-means clustering evaluation, leaving out the top classification

Table 2: **Comparison of the classification performance on biological dataset.** Unsupervised representation learning methods that have been widely used on biological analyze are compared, including kPCA (Halko et al., 2010), Ivis (Szubert et al., 2019), PHATE (Moon & van Dijk, 2019), PUMAP (Sainburg et al., 2021), PaCMAP (Wang et al., 2022), EVNet (Zang et al., 2022a) and hNNE (Sarfraz et al., 2022). The improvement over the best baseline is shown in parentheses.

	kPCA	PUMAP	Ivis	PHATE	Topo-AE	PaCMAP	EVNet	hNNE	DiffAug
GA1457	40.2	66.0	36.9	72.2	74.6	85.3	85.7	77.4	**92.7**(+7.0)
SAM561	34.6	59.9	45.6	71.5	72.4	83.7	83.6	83.8	**89.3**(+4.5)
MC1374	45.6	62.2	45.8	61.3	61.3	61.3	71.4	62.3	**71.8**(+0.4)
HCL500	26.5	36.3	24.4	33.8	56.0	36.2	62.3	62.2	**64.7**(+2.4)
AVE	36.7	56.1	38.2	59.1	66.1	74.8	75.8	71.4	**79.6**(+3.8)

layer. We then applied K-means clustering to these features. The primary metric for evaluation was clustering accuracy. Details of the experimental setup are in Appendix C. The training strategy of DiffAug is E-step: 200 epochs → M-step: 400 epoch → E-step: 800 epoch. The data of training time consumption is in the Table A.3.

Analysis. From Table 1, it's evident that DiffAug consistently outperforms state-of-the-art (SOTA) methods across all datasets. It surpasses other techniques by at least 1.0% in five out of the eight projects, raising the average metrics by a minimum of 2.5% compared to other evaluated methods. This showcases the effectiveness of DiffAug's data augmentation. (a) *Beyond hand-designed augmentation methods.* DiffAug's versatility indicates that its approach is on par with, or even better than, traditional hand-crafted methods. The encoder in DiffAug produces robust features. (b) *Beyond Mixup improved contrastive learning methods.* DiffAug outperforms the Mixup improved contrastive learning method of typical contrast learning methods, and additionally, models trained using DiffAug-generated data and contrast learning methods bring some improvement. (c) For datasets with many classes, like CF100 and TINet, DiffAug's encoder might only sometimes capture every detail. Still, augmented data is crucial in guiding contrastive learning to produce better results.

3.2 DiffAug Delivers a Boost in Unsupervised Representation Learning for High-Cost Biological Data

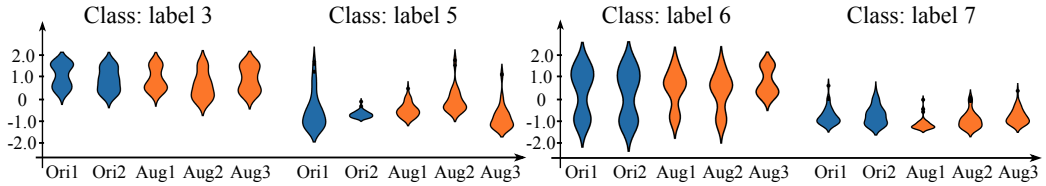
Next, we benchmark DiffAug against SOTA unsupervised representation learning methods in biological datasets. We present the comparative outcomes of SVC classification performance (Platt, 1999) in Table 2. The data of training time consumption is in the Table A.5.

Test protocols. Experiments are performed on biological datasets, including GA1457 (Rouillard et al., 2016), SAM (Weber & Robinson, 2016), MC1374 (Han et al., 2018), and HCL500 (Han et al., 2020) datasets. To assess the efficacy of the proposed methods, following Wang et al. (2022); Sarfraz et al. (2022), we utilized linear SVM performance to evaluate the performance of different methods. In the linear SVM evaluation, embeddings are partitioned with 90% designated for training and 10% for testing. Detailed specifics of this configuration are elaborated in the Appendix D. The training strategy of DiffAug is E-step: 330 epochs → M-step: 330 epoch → E-step: 340 epoch. The training details and the correctness change curve are in Fig. A.2.

Analysis. DiffAug consistently surpasses all other methods across eight evaluations spanning four datasets, registering a performance enhancement between 0.4% and 7.0% over its counterparts. Several key advantages of DiffAug emerge, especially when considering classification metrics: (a) Notably, the strengths of DiffAug aren't confined to vision data. It also excels in areas such as biology, where data is difficult to visualize and understand by humans and where it is challenging to manually design appropriate data augmentation methods. (b) Data processed through DiffAug exhibits reduced overlap among distinct groups, facilitating enhanced classification and clustering. This suggests that DiffAug delineates more explicit boundaries between data categories, culminating in more precise outcomes. (c) The approach underpinning DiffAug is versatile, making it a valuable addition to other unsupervised learning techniques. Historically, the quest for potent data augmentation strategies in biology has been arduous. We posit that DiffAug paves new avenues in this domain.

(a) CF10 (b) STL10

Figure 3: **The display of original and generated images illustrates that DiffAug generates semantically similar augmented images.** Ori means original image and Aug1, Aug2 and Aug3 are augmentated images. More detailed results are in the appendix.



(a) MC1374 (b) HCL500

Figure 4: **The violin plot of original and generated samples illustrates that the DiffAug generated samples have a similar distribution to the original samples.** Ori1 and Ori2 are two original sample and Aug1, Aug2 and Aug3 are augmentated data. Due to the large number of zero values in the biological data, we chose the genes with the highest 100 expression to plot violin plots to show the data distribution. Class: label 3 and Class: label 5 indicate two different classes in the dataset.

3.3 DiffAug Effectiveness Analysis

Next, we verify the tasks and roles of each module by showing the details of how DiffAug works.

Effectiveness analysis of diffusion generator. The diffusion module generates new positive data by inputting the provided condition vector. To demonstrate that the diffusion module works appropriately, we show the generation results for both the image and biological datasets (in Fig. 3 and Fig. 4). A more detailed implementation and more results are in the Appendix C and Appendix D. We can observe that the generated data retains semantic similarity to the original data. For example, the objects described in the image data are consistent, while the distribution of the gene is also consistent. At the same time, the generated data is not simply copied but varied without changing the semantic information.

In addition, to further explore the semantic differences between the newly generated and original data, we computed the cos-similarity of the original augmented sample in latent space. As depicted in Fig. 5, DiffAug's similarity distribution is smoother and broader. In comparison, Mixup tends to produce augmentations that are very similar semantically, while methods like cropping might introduce data with semantically distinct noise samples.

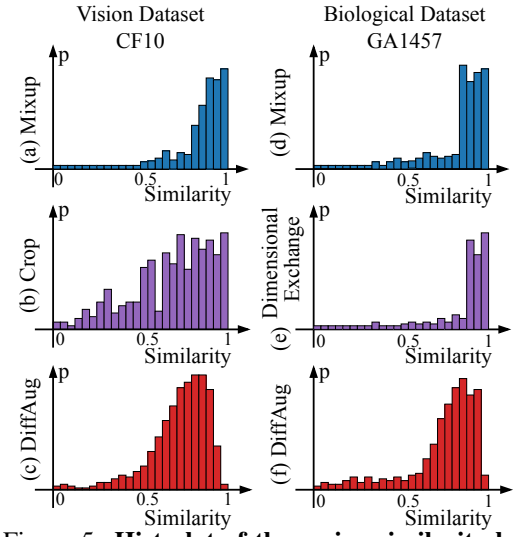


Figure 5: **Hist plot of the cosine similarity between original data and the augmentation data in latent space indicate that DiffAug generates semantically smooth augmentations.** For the image data, we compared similar mixups with random cropping. For biological datasets, we compared same-label Mixup and random dimension swapping.

Effectiveness analysis of semantic encoder. Next, we confirm that the semantic encoder of DiffAug works well by visualizing the representation of DiffAug and baseline methods (in Fig. 6). The

Table 3: **Ablation study of the semantic encoder includes DiffAug's encoder is necessary and can efficiently generate conditional vectors.** Linear-tests performance of different ablation setups on on vision dataset and biological dataset.

Datasets	Vision Datasets				Biological Datasets			
	CF10	CF100	STL10	TINet	GA1457	SAM561	MC1374	HCL500
A1. Gen$(\cdot)$ + Sup. Condition	**93.4**	**70.9**	**92.9**	45.9	92.5	**89.6**	71.1	63.9
A2. Gen$(\cdot)$ + Rand. Condition	34.2	10.4	30.1	7.3	10.5	16.9	13.9	10.0
A3. Gen$(\cdot)$ + Enc$(\cdot\|\theta)$ (DiffAug)	**93.4**	69.9	92.5	**49.7**	**92.7**	88.3	**71.8**	**64.7**

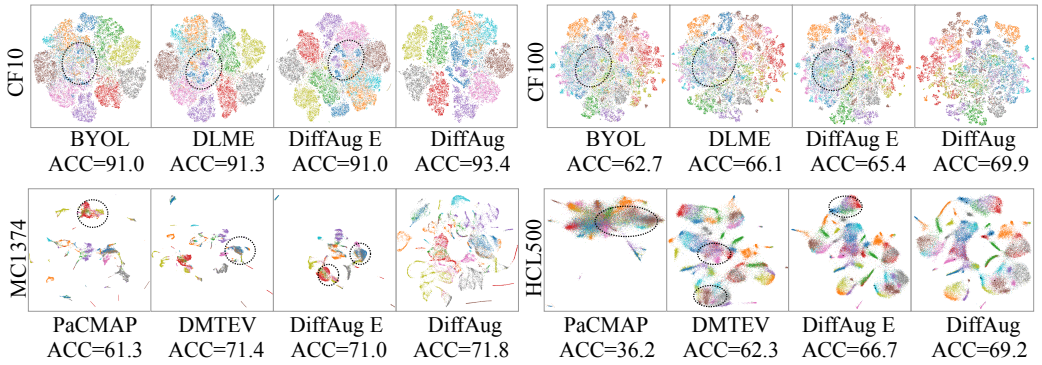


Figure 6: **The scatter visualization of representation indicates DiffAug's encoder learns cleaner embedding.** The colors represent different categories; there are 100 categories in CF100; we used the superclasses label provided by Deng et al. (2021). DiffAug E means only using the soft contrastive loss to train the encoder.

t-SNE (Van der Maaten & Hinton, 2008) is used to analyze the BYOL, DLME, and DiffAug embedding on CF10, CF100, MC1374, and HCL500 datasets. The results show that DiffAug's encoder learns cleaner embedding than baseline methods. In Fig. 6, DiffAug E means the first E-step's results of the DiffAug. By comparing DiffAug E and DiffAug, we observe that the augmented data further improves the embedding quality, significantly enhancing the depiction of local structures. The same conclusion is shown in Fig. A.2.

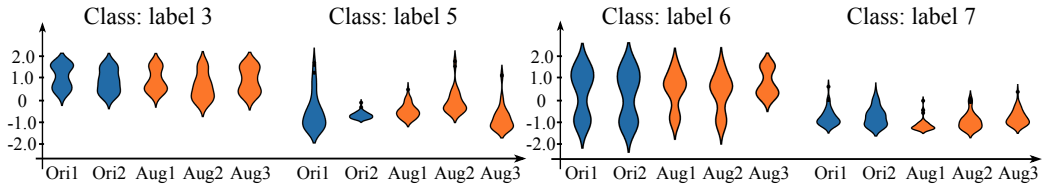
3.4 Ablation Study and Effectiveness of Each Component

Ablation study of the semantic encoder. In the ablation study presented in Table. 3, we consider three configurations: A1 and A2 confirm the significance of DiffAug's semantic encoder by ablating it in two ways. A1 directly uses supervised one hot label as the conditional, bypassing the condition vectors generated by the unsupervised neural network. A2 employs random conditional vectors instead of those the encoder produces. A3 means the proposed DiffAug method. The results from these experiments can be found in Table 3. We observe that the average performance of A1 is highest due to the access to the label. And not accessing the label at all brings a huge performance drop. The results in A3 illustrate that DiffAug's performance is comparable to the fully supervised condition, demonstrating its ability to model supervised annotation within an unsupervised framework.

Ablation study of training strategy and scl loss function. For Ablation in Table. 4, B1 means that the model is trained by SimCLR (Chen et al., 2020). B2 omits the diffusion loss and trains the encoder with only the soft contrastive learning loss. B3 omits the soft contrastive learning loss and trains the encoder with InfoNCE loss. B4 and B5 talk about the training strategy of DiffAug. B4 denotes training the model by integrating two loss functions, i.e., mixing E-Step and M-Step to update the parameters of both networks simultaneously through a single forward propagation.B5 denotes the default training strategy, which trains the model by alternating the two loss functions. The results from these experiments can be found in Table 4. First, we observe that either replacing the scl loss or replacing the diff model (B2 or B3) brings about performance degradation, which implies that the two modules of DiffAug work in conjunction with each other. Second, we observe

Table 4: **Ablation study of scl loss function and training strategy.** The classifier accuracy of each setting is displayed in this table. Soft contrastive learning is improved with typical contrast learning, and EM training is more stable.

Datasets	Vision Datasets				Biological Datasets			
	CF10	CF100	STL10	TINet	GA1457	SAM561	MC1374	HCL500
B1. SimCLR	89.8	57.4	86.9	38.4	9.4	16.8	14.3	16.8
B2. DiffAug w/o $\mathcal{L}_{\text{df}}$	91.3	66.1	90.1	44.9	89.1	82.1	59.3	62.3
B3. DiffAug w/o $\mathcal{L}_{\text{scl}}$	92.7	68.4	90.9	45.1	89.2	82.4	69.2	61.3
B4. DiffAug Syn. Training	92.9	69.7	92.7	45.3	90.1	**89.6**	68.1	62.3
B5. DiffAug EM Training	**93.4**	**69.9**	92.5	**49.7**	**92.7**	88.3	**71.8**	**64.7**



(a) MC1374 (b) HCL500

Figure 7: **Hypter-parametric λ analysis.** The Box plots of augmentation strength hyperparameters λ and model Linear-tests performance. For each parameter, statistical results of experiments based on grid search were performed.

that on some datasets, the performance of the two training strategies (B4 and B5) is comparable, but on others, the EM method demonstrates higher stability. We attribute this to the fact that the difficulty of diffusion model training varies from data to data, and simultaneous training may result in the two modules being unable to match at all times, bringing about instability in training. However, the E-M training approach avoids this problem.

3.5 Hyperparametric Analysis and the Toxicity of Generated Data

Finally, we investigate the performance improvement and potential toxicity of the DiffAug method through hyperparametric analysis. The hyperparameter λ determines how often the model generated by DiffAug affects the training of the semantic encoder. Introducing the least amount of augmentation data ($\lambda = 0$) brings the method back to traditional contrastive learning methods, while too much ($\lambda = 1$) will lead to encoder to crash. To demonstrate this, we tested the model performance of different λ counterparts on two visual datasets (CF10, CF100) and two biological datasets (SAM561 and MC1374). As shown in Fig. 7, the change in performance brought about by λ is consistent across datasets. Specifically, setting $\lambda = 0.1$ or $\lambda = 0.15$ provides the most significant gain. We believe that $\lambda = 0.1$ may be a suitable default setting for most datasets.

4 Conclusion

In summary, we presented DiffAug, an innovative contrastive learning framework that leverages diffusion-based augmentation to enhance the robustness and generalization of unsupervised learning. Unlike many existing methods, DiffAug operates independently of prior knowledge or external labels, positioning it as a versatile augmentation tool with notable performance in vision and life sciences. Our tests reveal that DiffAug consistently boosts classification and clustering accuracy across multiple datasets, such as CF10, CF100, STL10, TINet, HCL500, GA1457, SAM561, and MC1374. Given its capabilities, we see DiffAug evolving into a benchmark data augmentation method in unsupervised contrastive learning. We suggest fellow researchers delve into DiffAug and harness its potential for diverse applications.

REFERENCES

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks, 2017. URL https://arxiv.org/abs/1711.04340.

Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *ECCV*, pp. 456–473. Springer, 2022.

Victor Besnier, Himalaya Jain, Andrei Bursuc, Matthieu Cord, and Patrick Pérez. This dataset does not exist: Training models from generated images. In *ICASSP 2020*, pp. 1–5. IEEE, 2020. doi: 10.1109/ICASSP40776.2020.9053146. URL https://doi.org/10.1109/ICASSP40776.2020.9053146.

Alexandre Blanco-Gonzalez, Alfonso Cabezon, Alejandro Seco-Gonzalez, Daniel Conde-Torres, Paula Antelo-Riveiro, Angel Pineiro, and Rebeca Garcia-Fandino. The role of ai in drug discovery: challenges, opportunities, and strategies. *Pharmaceuticals*, 16(6):891, 2023.

Andrea Botton, Gianmarco Barberi, and Pierantonio Facco. Data augmentation to support biopharmaceutical process development through digital models—a proof of concept. *Processes*, 10(9): 1796, 2022.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR 2019*, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2002.05709. arXiv: 2002.05709.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, pp. 15750–15758, 2021.

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

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR*, pp. 702–703, 2020.

Jiequan Cui, Zhisheng Zhong, Shu Liu, Bei Yu, and Jiaya Jia. Parametric contrastive learning. In *ICCV*, pp. 715–724, 2021.

Pham Thanh Dat, Anuvabh Dutt, Denis Pellerin, and Georges Quénot. Classifier training from a generative model. In *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*, pp. 1–6, 2019. doi: 10.1109/CBMI.2019.8877479.

Danruo Deng, Guangyong Chen, Jianye Hao, Qiong Wang, and Pheng-Ann Heng. Flattening sharpness for dynamic gradient projection memory benefits continual learning. *Advances in Neural Information Processing Systems*, 34:18710–18721, 2021.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *NeurIPS 2021*, pp. 8780–8794, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Xuefeng Du, Yiyou Sun, Xiaojin Zhu, and Yixuan Li. Dream the impossible: Outlier imagination with diffusion models, 2023.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *NeurIPS*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. *arXiv:2006.07733 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2006.07733. arXiv: 2006.07733.

Maya R Gupta, Yihua Chen, et al. Theory and use of the em algorithm. *Foundations and Trends® in Signal Processing*, 4(3):223–296, 2011.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2010.

Xi Han, Liheng Zhang, Kang Zhou, and Xiaonan Wang. Progan: Protein solubility generative adversarial nets for data augmentation in dnn framework. *Computers & Chemical Engineering*, 131:106533, 2019.

Xiaoping Han, Renying Wang, Yincong Zhou, Lijiang Fei, Huiyu Sun, Shujing Lai, Assieh Saadatpour, Ziming Zhou, Haide Chen, Fang Ye, et al. Mapping the mouse cell atlas by microwell-seq. *Cell*, 172(5):1091–1107, 2018.

Xiaoping Han, Ziming Zhou, Lijiang Fei, Huiyu Sun, Renying Wang, Yao Chen, Haide Chen, Jingjing Wang, Huanna Tang, Wenhao Ge, et al. Construction of a human cell landscape at single-cell level. *Nature*, 581(7808):303–309, 2020.

Ayaan Haque. EC-GAN: low-sample classification using semi-supervised algorithms and gans (student abstract). In *AAAI*, pp. 15797–15798, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17895.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. URL http://arxiv.org/abs/1512.03385. arXiv: 1512.03385.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. *arXiv:1911.05722 [cs]*, March 2020. URL http://arxiv.org/abs/1911.05722. arXiv: 1911.05722.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. 2021.

Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition?, 2022. URL https://arxiv.org/abs/2210.07574.

Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? *ICLR*, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Hai-Hui Huang, Hao Rao, Rui Miao, and Yong Liang. A novel meta-analysis based on data augmentation and elastic data shared lasso regularization for gene expression. *BMC bioinformatics*, 23(Suppl 10):353, 2022.

Weiran Huang, Mingyang Yi, and Xuyang Zhao. Towards the generalization of contrastive self-supervised learning. *ICLR*, 2023.

Ali Jahanian, Xavier Puig, Yonglong Tian, and Phillip Isola. Generative models as a data source for multiview representation learning. In *ICLR*, 2022. URL https://openreview.net/forum?id=qhAeZjs7dCL.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *ICLR,*, 2014. URL http://arxiv.org/abs/1312.6114.

Rayan Krishnan, Pranav Rajpurkar, and Eric J Topol. Self-supervised learning in medicine and healthcare. *Nature Biomedical Engineering*, pp. 1–7, 2022.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

Bolian Li, Zongbo Han, Haining Li, Huazhu Fu, and Changqing Zhang. Trustworthy long-tailed classification. *arXiv: Learning*, 2021.

Menglu Li and Wen Zhang. Phiaf: prediction of phage-host interactions with gan-based data augmentation and sequence-based feature fusion. *Briefings in Bioinformatics*, 23(1):bbab348, 2022.

Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, 3(1):91–99, 2022.

Miles McGibbon, Sam Money-Kyrle, Vincent Blay, and Douglas R Houston. Scorch: improving structure-based virtual screening with machine learning classifiers, data augmentation, and uncertainty estimation. *Journal of Advanced Research*, 46:135–147, 2023.

Kevin R Moon and van Dijk. Visualizing structure and transitions in high dimensional biological data. *Nature biotechnology*, 37(12):1482–1492, 2019.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang (eds.), *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8162–8171. PMLR, 2021. URL http://proceedings.mlr.press/v139/nichol21a.html.

Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16784–16804. PMLR, 2022. URL https://proceedings.mlr.press/v162/nichol22a.html.

Xiangyu Peng, Kai Wang, Zheng Zhu, Mang Wang, and Yang You. Crafting better contrastive views for siamese representation learning. In *CVPR 2022*, pp. 16031–16040, 2022.

J Platt. Probabilistic outputs for svms and comparisons to regularized likehood methods. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *NeurIPS 2019*, pp. 14837–14847, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Abstract.html.

Nils Rethmeier and Isabelle Augenstein. A primer on contrastive pretraining in language processing: Methods, lessons learned, and perspectives. *ACM Computing Surveys*, 55(10):1–17, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR 2022*, pp. 10674–10685. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01042. URL https://doi.org/10.1109/CVPR52688.2022.01042.

Andrew D Rouillard, Gregory W Gundersen, Nicolas F Fernandez, Zichen Wang, Caroline D Monteiro, Michael G McDermott, and Avi Ma'ayan. The harmonizome: a collection of processed datasets gathered to serve and mine knowledge about genes and proteins. *Database*, 2016, 2016.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. URL https://arxiv.org/abs/2205.11487.

Tim Sainburg, Leland McInnes, and Timothy Q. Gentner. Parametric UMAP embeddings for representation and semi-supervised learning. *arXiv:2009.12981 [cs, q-bio, stat]*, April 2021. URL http://arxiv.org/abs/2009.12981. arXiv: 2009.12981.

Saquib Sarfraz, Marios Koulakis, Constantin Seibold, and Rainer Stiefelhagen. Hierarchical nearest neighbor graph embedding for efficient dimensionality reduction. In *CVPR*, pp. 336–345, 2022.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022. URL https://arxiv.org/abs/2210.08402.

Benjamin Szubert, Jennifer E. Cole, Claudia Monaco, and Ignat Drozdov. Structure-preserving visualisation of high dimensional single-cell datasets. *Scientific Reports*, 9(1):8914, June 2019. ISSN 2045-2322.

Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. Data augmentation using gans, 2019. URL https://arxiv.org/abs/1904.09135.

Christina V Theodoris, Ling Xiao, Anant Chopra, Mark D Chaffin, Zeina R Al Sayed, Matthew C Hill, Helene Mantineo, Elizabeth M Brydon, Zexian Zeng, X Shirley Liu, et al. Transfer learning enables predictions in network biology. *Nature*, pp. 1–9, 2023.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *NeurIPS*, 33:6827–6839, 2020.

Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. Effective data augmentation with diffusion models. *arXiv preprint arXiv:2302.07944*, 2023.

Toan Tran, Trung Pham, Gustavo Carneiro, Lyle J. Palmer, and Ian D. Reid. A bayesian data augmentation approach for learning deep models. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *NeurIPS 2017*, pp. 2797–2806, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/076023edc9187cf1ac1f1163470e479a-Abstract.html.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008.

Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *TPAMI*, 45(5): 5549–5560, 2022.

Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *JMLR*, 22(201):1–73, 2022. URL http://jmlr.org/papers/v22/20-1061.html.

Lukas M Weber and Mark D Robinson. Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. *Cytometry Part A*, 89(12):1084–1096, 2016.

Sajila Wickramaratne and Md Shaad Mahmud. Conditional-gan based data augmentation for deep learning task classifier improvement using fnirs data. *Frontiers in Big Data*, 4:659146, 07 2021. doi: 10.3389/fdata.2021.659146.

Wei Xiong, Yutong He, Yixuan Zhang, Wenhan Luo, Lin Ma, and Jiebo Luo. Fine-grained image-to-image transformation towards visual recognition. In *CVPR 2020*, June 2020.

Mingle Xu, Sook Yoon, Alvaro Fuentes, and Dong Sun Park. A comprehensive survey of image augmentation techniques for deep learning. *Pattern Recognition*, pp. 109347, 2023.

Shin'ya Yamaguchi, Sekitoshi Kanai, and Takeharu Eda. Effective data augmentation with multi-domain learning gans. In *AAAI 2020*, pp. 6566–6574. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6131.

Zipei Yan, Linchuan Xu, Atsushi Suzuki, Jing Wang, Jiannong Cao, and Jun Huang. Rgb color model aware computational color naming and its application to data augmentation. In *2022 IEEE International Conference on Big Data (Big Data)*, pp. 1172–1181. IEEE, 2022.

Tianhao Yu, Haiyang Cui, Jianan Canal Li, Yunan Luo, Guangde Jiang, and Huimin Zhao. Enzyme function prediction using contrastive learning. *Science*, 379(6639):1358–1363, 2023.

Zelin Zang, Shenghui Cheng, Linyan Lu, Hanchen Xia, Liangyu Li, Yaoting Sun, Yongjie Xu, Lei Shang, Baigui Sun, and Stan Z Li. Evnet: An explainable deep network for dimension reduction. *TVCG*, 2022a.

Zelin Zang, Siyuan Li, Di Wu, Ge Wang, Kai Wang, Lei Shang, Baigui Sun, Hao Li, and Stan Z Li. Dlme: Deep local-flatness manifold embedding. *ECCV*, pp. 576–592, 2022b.

Zelin Zang, Lei Shang, Senqiao Yang, Fei Wang, Baigui Sun, Xuansong Xie, and Stan Z. Li. Boosting novel category discovery over domains with soft contrastive learning and all-in-one classifier. *ICCV*, 2023.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. URL http://arxiv.org/abs/1710.09412.

Junbo Zhang and Kaisheng Ma. Rethinking the augmentation module in contrastive learning: Learning hierarchical augmentation invariance with expanded views. In *CVPR 2022*, pp. 16650–16659, 2022.

Shaofeng Zhang, Meng Liu, Junchi Yan, Hengrui Zhang, Lingxiao Huang, Xiaokang Yang, and Pinyan Lu. M-mix: Generating hard negatives via multi-sample mixing for contrastive learning. In *KDD*, pp. 2461–2470, 2022.

Yifan Zhang, Daquan Zhou, Bryan Hooi, Kai Wang, and Jiashi Feng. Expanding small-scale datasets with guided imagination, 2023a.

Yuhan Zhang, He Zhu, and Shan Yu. Adaptive data augmentation for contrastive learning. *ICASSP 2023*, pp. 1–5, 2023b.

Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. In *ICLR*, 2021a.

Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *CVPR*, pp. 10145–10155, 2021b.

Zhedong Zheng, Liang Zheng, and Yi Yang. Unlabeled samples generated by GAN improve the person re-identification baseline in vitro. In *ICCV 2017*, pp. 3774–3782. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.405. URL https://doi.org/10.1109/ICCV.2017.405.

Zhiwei Zheng, Nguyen Quoc Khanh Le, and Matthew Chin Heng Chua. Maskdna-pgd: An innovative deep learning model for detecting dna methylation by integrating mask sequences and adversarial pgd training as a data augmentation method. *Chemometrics and Intelligent Laboratory Systems*, 232:104715, 2023.

Appendix:
Boosting Unsupervised Contrastive Learning Using Diffusion-Based Data Augmentation From Scratch

Contents

A Appendix: Related Works

Generative Models Generative models have been the subject of growing interest and rapid advancement. Earlier methods, including VAEs (Kingma & Welling, 2014) and GANs (Goodfellow et al., 2014), showed initial promise generating realistic images, and were scaled up in terms of resolution and sample quality (Brock et al., 2019; Razavi et al., 2019). Despite the power of these methods, many recent successes in photorealistic image generation were the result of diffusion models (Ho et al., 2020; Nichol & Dhariwal, 2021; Saharia et al., 2022; Nichol et al., 2022; Ramesh et al., 2022). Diffusion models have been shown to generate higher-quality samples compared to their GAN counterparts (Dhariwal & Nichol, 2021), and developments like classifier free guidance (Ho & Salimans, 2022) have made text-to-image generation possible. Recent emphasis has been on training these models with internet-scale datasets like LAION-5B (Schuhmann et al., 2022). Generative models trained at internet-scale (Rombach et al., 2022; Saharia et al., 2022; Nichol et al., 2022; Ramesh et al., 2022) have unlocked several application areas where photorealistic generation is crucial.

Synthetic Image Data Generation Training neural networks on synthetic data from generative models was popularized using GANs (Antoniou et al., 2017; Tran et al., 2017; Zheng et al., 2017).

Various applications for synthetic data generated from GANs have been studied, including representation learning (Jahanian et al., 2022), inverse graphics (Zhang et al., 2021a), semantic segmentation (Zhang et al., 2021b), and training classifiers (Tanaka & Aranha, 2019; Dat et al., 2019; Yamaguchi et al., 2020; Besnier et al., 2020; Xiong et al., 2020; Wickramaratne & Mahmud, 2021; Haque, 2021). More recently, synthetic data from diffusion models has also been studied in a few-shot setting (He et al., 2022). These works use generative models that have likely seen images of target classes and, to the best of our knowledge, we present the first analysis for synthetic data on previously unseen concepts. Du et al. (2023) proposed the DREAM-OOD framework, which uses diffusion models to generate photo-realistic outliers from in-distribution data for improved OOD detection. By learning a text-conditioned latent space, it visualizes imagined outliers directly in pixel space, showing promising results in empirical studies. Zhang et al. (2023a) developed the Guided Imagination Framework (GIF) using generative models like DALL-E2 and Stable Diffusion for dataset expansion, enhancing accuracy in both natural and medical image datasets.

Synthetic Biology Data Generation The realm of synthetic biology has witnessed a surge in the utilization of data-driven approaches, particularly with the advent of advanced computational models. The generation of synthetic biological data has been instrumental in predicting protein structures (McGibbon et al., 2023). The use of Generative Adversarial Networks (GANs) has also found its way into this domain, aiding in the c reation of synthetic DNA sequences (Zheng et al., 2023; Li & Zhang, 2022; Han et al., 2019) and simulating cell behaviors (Botton et al., 2022). Furthermore, the integration of machine learning with synthetic biology has paved the way for innovative solutions in drug discovery (Blanco-Gonzalez et al., 2023; McGibbon et al., 2023). Unlike the synthetic image data generation, where models have often seen images of target classes, synthetic biology data generation often grapples with the challenge of generating data for entirely novel biological entities. This presents a unique set of challenges and opportunities, pushing the boundaries of what synthetic data can achieve in the realm of biology.

B Appendix: Details of Contrastive Learning and Soft Contrastive Learning

B.1 The t-kernel similarity in soft contrastive learning

To map the high-dimensional embedding vector to a probability value, a kernel function $\mathcal{S}(\cdot)$ is used. In this paper, we use the t-distribution kernel function $\mathcal{S}^{\nu}(\cdot)$ because it exposes the degrees of freedom and allows us to adjust the closeness of the distribution in the dimensionality reduction mapping (Li et al., 2021). The t-distribution kernel function is defined as follows,

$$\mathcal{S}(\mathbf{z}_i, \mathbf{z}_j) = \Gamma\left((\nu+1)/2\right)\left(1+\|z_i - z_j\|_2^2/\nu\right)^{-\frac{\nu+1}{2}} / \sqrt{\nu\pi}\Gamma\left(\nu/2\right), \tag{9}$$

where $\Gamma(\cdot)$ is the Gamma function. The degrees of freedom ν control the shape of the kernel function. The different degrees of freedom (ν^y, ν^z) is used in $\mathcal{R}^y$ and $\mathcal{R}^z$ for the dimensional reduction mapping.

B.2 Why Soft Contrastive Learning is a softened version of Contrastive Learning

Lemma 1. *Let* $\mathcal{L}_{\text{cl}} = -\log \mathcal{Q}\left(\mathbf{z}_i, \mathbf{z}_i^+\right) + \log\left[\mathcal{Q}\left(\mathbf{z}_i, \mathbf{z}_i^+\right) + \sum_{\mathbf{z}_i^- \in V^-} \mathcal{Q}\left(\mathbf{z}_i, \mathbf{z}_i^-\right)\right]$ *and* $\mathcal{L}_{\text{cl}}^p = -\sum_{j=1}^{N_K+1}\left\{\mathcal{H}_{ij}\log Q_{ij} + (1-\mathcal{H}_{ij})\log \dot{Q}_{ij}\right\}$ *Then* $\lim_{x\to\infty}\mathcal{L}_{\text{cl}} - \mathcal{L}_{\text{cl}}^p = 0$.

Proof. We start with $L_{\text{CL}} = -\log\frac{\exp(S(z_i,z_j))}{\sum_{k=1}^{N_K}\exp(S(z_i,z_k))}$ (Eq. (3)), then

$$L_{\text{CL}} = \log N_K - \log\frac{\exp(S(z_i,z_j))}{\frac{1}{N_K}\sum_{k=1}^{N_K}\exp(S(z_i,z_k))}.$$

We are only concerned with the second term that has the gradient. Let (i, j) are positive pair and $(i, k_1), \cdots, (i, k_N)$ are negative pairs. The overall loss associated with point i is:

$$
\begin{aligned}
& -\log \frac{\exp(S(z_i, z_j))}{\frac{1}{N_K}\sum_{k=1}^{N_K}\exp(S(z_i, z_k))} \\
=& -\left[\log\exp(S(z_i, z_j)) - \log\frac{1}{N_K}\sum_{k=1}^{N_K}\exp(S(z_i, z_k))\right] \\
=& -\left[\log\exp(S(z_i, z_j)) - \sum_{k=1}^{N_K}\log\exp(S(z_i, z_k)) + \sum_{k=1}^{N_K}\log\exp(S(z_i, z_k)) - \log\frac{1}{N_K}\sum_{k=1}^{N_K}\exp(S(z_i, z_k))\right] \\
=& -\left[\log\exp(S(z_i, z_j)) - \sum_{k=1}^{N_K}\log\exp(S(z_i, z_k)) + \log\Pi_{k=1}^{N_K}\exp(S(z_i, z_k)) - \log\frac{1}{N_K}\sum_{k=1}^{N_K}\exp(S(z_i, z_k))\right] \\
=& -\left[\log\exp(S(z_i, z_j)) - \sum_{k=1}^{N_K}\log\exp(S(z_i, z_k)) + \log\frac{\Pi_{k=1}^{N_K}\exp(S(z_i, z_k))}{\frac{1}{N_K}\sum_{k=1}^{N_K}\exp(S(z_i, z_k))}\right]
\end{aligned}
$$

We focus on the case where the similarity is normalized, $S(z_i, z_k) \in [0, 1]$. The data i and data k is the negative samples, then $S(z_i, z_k)$ is near to 0, $\exp(S(z_i, z_k))$ is near to 1, thus the $\frac{\Pi_{k=1}^{N_K}\exp(S(z_i, z_k))}{\frac{1}{N}\sum_{k=1}^{N_K}\exp(S(z_i, z_k))}$ is near to 1, and $\log\frac{\Pi_{k=1}^{N_K}\exp(S(z_i, z_k))}{\frac{1}{N}\sum_{k=1}^{N_K}\exp(S(z_i, z_k))}$ near to 0. We have

$$
L_{\text{CL}} \approx -\left[\log\exp(S(z_i, z_j)) - \sum_{k=1}^{N_K}\log\exp(S(z_i, z_k))\right]
$$

We denote ij and ik by a uniform index and use $\mathcal{H}_{ij}$ to denote the homology relation of ij.

$$
\begin{aligned}
L_{\text{CL}} &\approx -\left[\log\exp(S(z_i, z_j)) - \sum_{k=1}^{N_K}\log\exp(S(z_i, z_k))\right] \\
&\approx -\left[\mathcal{H}_{ij}\log\exp(S(z_i, z_j)) - \sum_{j=1}^{N_K}(1-\mathcal{H}_{ij})\log\exp(S(z_i, z_j))\right] \\
&\approx -\left[\sum_{j=1}^{N_K+1}\{\mathcal{H}_{ij}\log\exp(S(z_i, z_j)) + (1-\mathcal{H}_{ij})\log\{\exp(-S(z_i, z_j))\}\}\right]
\end{aligned}
$$

we define the similarity of data i and data j as $Q_{ij} = \exp(S(z_i, z_j))$ and the dissimilarity of data i and data j as $\dot{Q}_{ij} = \exp(-S(z_i, z_j))$.

$$
L_{\text{CL}} \approx -\left[\sum_{j=1}^{N_K+1}\left\{\mathcal{H}_{ij}\log Q_{ij} + (1-\mathcal{H}_{ij})\log\dot{Q}_{ij}\right\}\right]
$$

□

The proposed SCL loss is a smoother CL loss:

This proof tries to indicate that the proposed SCL loss is a smoother CL loss. We discuss the differences by comparing the two losses to prove this point. the forward propagation of the network is, $z_i = H(\hat{z}_i), \hat{z}_i = F(x_i)$, $z_j = H(\hat{z}_j), \hat{z}_j = F(x_j)$. We found that we mix y and $\hat{z}$ in the main text, and we will correct this in the new version. So, in this section $z_i = H(y_i), y_i = F(x_i)$, $z_j = H(y_j), y_j = F(x_j)$ is also correct.

Let $H(\cdot)$ satisfy K-Lipschitz continuity, then $d_{ij}^z = k^* d_{ij}^y, k^* \in [1/K, K]$, where k^* is a Lipschitz constant. The difference between L_{SCL} loss and L_{CL} loss is,

$$L_{\text{CL}} - L_{\text{SCL}} \approx \sum_j \left[\left(\mathcal{H}_{ij} - [1 + (e^\alpha - 1)\mathcal{H}_{ij}] \kappa \left(d_{ij}^y \right) \right) \log \left(\frac{1}{\kappa \left(d_{ij}^z \right)} - 1 \right) \right]. \tag{10}$$

Because the $\alpha > 0$, the proposed SCL loss is the soft version of the CL loss. if $\mathcal{H}_{ij} = 1$, we have:

$$(L_{\text{CL}} - L_{\text{SCL}})|_{\mathcal{H}_{ij}=1} = \sum \left[\left((1 - e^\alpha) \kappa \left(k^* d_{ij}^z \right) \right) \log \left(\frac{1}{\kappa \left(d_{ij}^z \right)} - 1 \right) \right] \tag{11}$$

then:

$$\lim_{\alpha \to 0} (L_{\text{CL}} - L_{\text{SCL}})|_{\mathcal{H}_{ij}=1} = \lim_{\alpha \to 0} \sum \left[\left((1 - e^\alpha) \kappa \left(k^* d_{ij}^z \right) \right) \log \left(\frac{1}{\kappa \left(d_{ij}^z \right)} - 1 \right) \right] = 0 \tag{12}$$

Based on Eq.(12), we find that if i, j is neighbor ($\mathcal{H}_{ij} = 1$) and $\alpha \to 0$, there is no difference between the CL loss L_{CL} and SCL loss L_{SCL}. When if $\mathcal{H}_{ij} = 0$, the difference between the loss functions will be the function of d_{ij}^z. The CL loss L_{CL} only minimizes the distance between adjacent nodes and does not maintain any structural information. The proposed SCL loss considers the knowledge both comes from the output of the current bottleneck and data augmentation, thus less affected by view noise.

Details of Eq. (10). Due to the very similar gradient direction, we assume $\dot{Q}_{ij} = 1 - Q_{ij}$. The contrastive learning loss is written as,

$$L_{\text{CL}} \approx - \sum \{ \mathcal{H}_{ij} \log Q_{ij} + (1 - \mathcal{H}_{ij}) \log (1 - Q_{ij}) \} \tag{13}$$

where $\mathcal{H}_{ij}$ indicates whether i and j are augmented from the same original data.

The SCL loss is written as:

$$L_{\text{SCL}} = - \sum \{ P_{ij} \log Q_{ij} + (1 - P_{ij}) \log (1 - Q_{ij}) \} \tag{14}$$

According to Eq. (4) and Eq. (5), we have

$$\begin{aligned} P_{ij} &= R_{ij} \kappa(d_{ij}^y) = R_{ij} \kappa(y_i, y_j), R_{ij} = \begin{cases} e^\alpha & \text{if } \mathcal{H}(x_i, x_j) = 1 \\ 1 & \text{otherwise} \end{cases}, \\ Q_{ij} &= \kappa(d_{ij}^z) = \kappa(z_i, z_j), \end{aligned} \tag{15}$$

For ease of writing, we use distance as the independent variable, $d_{ij}^y = \|y_i - y_j\|_2, d_{ij}^z = \|z_i - z_j\|_2$.

The difference between the two loss functions is:

$$
\begin{aligned}
& L_{\mathrm{CL}}-L_{\mathrm{SCL}} \\
=&-\sum\left[\mathcal{H}_{ij} \log \kappa\left(d_{ij}^z\right)+\left(1-\mathcal{H}_{ij}\right) \log \left(1-\kappa\left(d_{ij}^z\right)\right)-R_{ij} \kappa\left(d_{ij}^y\right) \log \kappa\left(d_{ij}^z\right)-\left(1-R_{ij} \kappa\left(d_{ij}^y\right)\right) \log \left(1-\kappa\left(d_{ij}^z\right)\right)\right] \\
=&-\sum\left[\left(\mathcal{H}_{ij}-R_{ij} \kappa\left(d_{ij}^y\right)\right) \log \kappa\left(d_{ij}^z\right)+\left(1-\mathcal{H}_{ij}-1+R_{ij} \kappa\left(d_{ij}^y\right)\right) \log \left(1-\kappa\left(d_{ij}^z\right)\right)\right] \\
=&-\sum\left[\left(\mathcal{H}_{ij}-R_{ij} \kappa\left(d_{ij}^y\right)\right) \log \kappa\left(d_{ij}^z\right)+\left(R_{ij} \kappa\left(d_{ij}^y\right)-\mathcal{H}_{ij}\right) \log \left(1-\kappa\left(d_{ij}^z\right)\right)\right] \\
=&-\sum\left[\left(\mathcal{H}_{ij}-R_{ij} \kappa\left(d_{ij}^y\right)\right)\left(\log \kappa\left(d_{ij}^z\right)-\log \left(1-\kappa\left(d_{ij}^z\right)\right)\right)\right] \\
=&\sum\left[\left(\mathcal{H}_{ij}-R_{ij} \kappa\left(d_{ij}^y\right)\right) \log \left(\frac{1}{\kappa\left(d_{ij}^z\right)}-1\right)\right]
\end{aligned}
\tag{16}
$$

Substituting the relationship between $\mathcal{H}_{ij}$ and R_{ij}, $R_{ij}=1+(e^{\alpha}-1)\mathcal{H}_{ij}$, we have

$$
L_{\mathrm{CL}}-L_{\mathrm{SCL}}=\sum\left[\left(\mathcal{H}_{ij}-[1+(e^{\alpha}-1)\mathcal{H}_{ij}] \kappa\left(d_{ij}^y\right)\right) \log \left(\frac{1}{\kappa\left(d_{ij}^z\right)}-1\right)\right] \tag{17}
$$

We assume that network $H(\cdot)$ to be a Lipschitz continuity function, then

$$
\frac{1}{K} H(d_{ij}^z) \leq d_{ij}^y \leq K H(d_{ij}^z) \quad \forall i, j \in\{1,2, \cdots, N\} \tag{18}
$$

We construct the inverse mapping of $H(\cdot)$ to $H^{-1}(\cdot)$,

$$
\frac{1}{K} d_{ij}^z \leq d_{ij}^y \leq K d_{ij}^z \quad \forall i, j \in\{1,2, \cdots, N\} \tag{19}
$$

and then there exists k^*:

$$
d_{ij}^y=k^* d_{ij}^z \quad k^* \in[1 / K, K] \quad \forall i, j \in\{1,2, \cdots, N\} \tag{20}
$$

Substituting the Eq.(20) into Eq.(17).

$$
L_{\mathrm{CL}}-L_{\mathrm{SCL}}=\sum\left[\left(\mathcal{H}_{ij}-[1+(e^{\alpha}-1)\mathcal{H}_{ij}] \kappa\left(k^* d_{ij}^z\right)\right) \log \left(\frac{1}{\kappa\left(d_{ij}^z\right)}-1\right)\right] \tag{21}
$$

C Appendix: Details of Vision Experiments

C.1 Dataset Setups

Experiments are performed on CIFAR-10 [CF10][1] and CIFAR-100[2] [CF100] (Krizhevsky et al., 2009), STL10[3] (Coates et al., 2011), TinyImageNet[4] [TINet] (Le & Yang, 2015) dataset.

To compare with the two different baseline methods, the setting of the dataset is shown in Table. A.1.

[1] https://www.cs.toronto.edu/ kriz/cifar.html
[2] https://www.cs.toronto.edu/ kriz/cifar.html
[3] https://cs.stanford.edu/ acoates/stl10/
[4] https://www.kaggle.com/c/tiny-imagenet

Table A.1: Dataset setting of linear-test Performance.

Dataset	Train Split	Test Split	Train Samples	Test Samples	Classes
CF10	Train	Test	50,000	10,000	10
CF100	Train	Test	50,000	10,000	100
STL10	Train + Unlabeled	Test	5,000+100,000	8,000	10
TINet	Train	Test	100,000	100,000	200

Table A.2: Dataset setting of clustering test.

Dataset	Train & Test Split	Train & Test Samples	Classes
CF10	Train+Test	60,000	10
CF100	Train+Test	60,000	20
STL10	Train+Test	13,000	10
TIN	Train	100,000	200

C.2 Baseline Methods and Implementation Details

The contrastive learning methods, including SimCLR (Chen et al., 2020), MOCO v2 (He et al., 2020), BYOL (Grill et al., 2020), SimSiam (Chen & He, 2021), and DLME (Zang et al., 2022b) are chosen for comparison. The SimC.+Mix. and MoCo.+Mix. are SimCLR and MoCoV2 with Mixup data augmentation which processed by Zhang et al. (2022). The SimC.+Dif. and MoCo.+Mix. are SimCLR and MoCoV2 with DiffAug data augmentation. Improvements over the best baseline are shown in parentheses.

For the Linear-test performance assessment, we followed a procedure similar to SimCLR (Chen et al., 2020). We evaluated the model's representations linearly on top of the frozen features. This ensures that the quality of the representations is attributed only to the pre-training task, without any influence from potential fine-tuning. We used the ResNet-50 (He et al., 2015) backbone as the encoder and a standard diffusion backbone as diffusion model (in code below). In contrast, for DiffAug, its semantic encoder served as the contrastive learning backbone, trained using DiffAug-augmented images. For the kMeans clustering evaluation, we extracted feature vectors from the models, leaving out the top classification layer. We then applied kMeans clustering to these features. The primary metric for evaluation was clustering accuracy.

Listing 1: DiffusionModel for Vision Task

```
class DiffusionModelVision(nn.Module):
    def __init__(self, c_in=3, c_out=3, time_dim=256):
        super().__init__()
        self.time_dim = time_dim
        self.remove_deep_conv = remove_deep_conv
        self.inc = DoubleConv(c_in, 16)
        self.down1 = Down(16, 32)
        self.sa1 = SelfAttention(32)
        self.down2 = Down(32, 64)
        self.sa2 = SelfAttention(64)
        self.down3 = Down(64, 64)
        self.sa3 = SelfAttention(64)
        self.up1 = Up(128, 32)
        self.sa4 = SelfAttention(32)
        self.up2 = Up(64, 16)
        self.sa5 = SelfAttention(16)
        self.up3 = Up(32, 16)
        self.sa6 = SelfAttention(16)
        self.outc = nn.Conv2d(16, c_out, kernel_size=1)
        def pos_encoding(self, t, channels):
        inv_freq = 1.0 / (10000 ** (torch.arange(0, channels, 2,
            device=one_param(self).device).float() / channels))
        pos_enc_a = torch.sin(t.repeat(1, channels // 2) * inv_freq)
        pos_enc_b = torch.cos(t.repeat(1, channels // 2) * inv_freq)
        pos_enc = torch.cat([pos_enc_a, pos_enc_b], dim=-1)
```

STL10 (96*96*3) CF10 (32*32*3)

Figure A.1: **The display of original and generated images illustrates that DiffAug generates semantically similar augmented images.** Ori means original image and Aug1, Aug2 and Aug3 are augmentated images. More detailed results are in the appendix.

```
            return pos_enc

        def forward(self, x, t):
            t = t.unsqueeze(-1)
            t = self.pos_encoding(t, self.time_dim)
            return self.unet_forwad(x, t)
```

Our training strategy is as follows: E-step: 200 epochs → M-step: 400 epoch → E-step: 800 epoch. Continued training will further improve performance, but we did not increase the amount of computation due to computational resource constraints. The time loss of the method does improve due to the use of the diffusion model. However, on small datasets, this boost is acceptable. In this way at the same time DiffAug gives the possibility to accomplish unsupervised comparison learning training on small datasets.

Table A.3: Details of the training process in vision dataset.

CF10	ν	Learning Rate	Weight Decay	Batch Size	GPU	Training Time
CF10	1	0.001	1e-6	256	1*V100	7.1 hours
CF100	2	0.001	1e-6	256	1*V100	7.2 hours
STL10	5	0.001	1e-6	256	1*V100	15.1 hours
TINet	3	0.001	1e-6	256	1*V100	20.6 hours

C.3 Data Augmentation of the Compared Methods

BYOL augmentation. The BYOL augmentation method is a hand-designed method. It is composed of four parts: random cropping, left-right flip, color ji

- Random cropping: A random patch of the image is selected, with an area uniformly sampled between 8% and 100% of that of the original image, and an aspect ratio logarithmically sampled between 3/4 and 4/3. This patch is then resized to the target size of 224 × 224 using bicubic interpolation.
- Optional left-right flip.

- Color jittering: The brightness, contrast, saturation, and hue of the image are shifted by a uniformly random offset applied to all the pixels of the same image. The order in which these shifts are performed is randomly selected for each patch.
- Color dropping: An optional conversion to grayscale. When applied, the output intensity for a pixel (r, g, b) corresponds to its luma component, computed as 0.2989r + 0.5870g + 0.1140b1.

SimCLR augmentation.

- Random Cropping: This involves taking a random crop of the image and then resizing it back to the original size. This can be seen as a combination of zooming and spatial location changes.
- Random Flipping: Randomly flip the image horizontally.
- Color Distortion: Apply a random color distortion. In the SimCLR paper, they use a combination of random brightness, random contrast, random saturation, and random hue changes. The strength of these distortions is controlled by a factor.
- Gaussian Blur: Apply a random Gaussian blur to the image. The extent of blurring is controlled by a factor.

MoCo v2 augmentation. For MoCo v2, the data augmentations are similar to those used in SimCLR, but there might be slight differences in implementation details. Here are the main augmentations used in MoCo v2:

- Random Cropping: This involves taking a random crop of the image and then resizing it back to the original size. Random Flipping: Randomly flip the image horizontally.
- Color Jitter: Randomly change the brightness, contrast, saturation, and hue of the image.
- Gaussian Blur: Apply Gaussian blur to the image with a certain probability.
- Solarization: This is an augmentation introduced in MoCo v2. It inverts pixel values above a threshold, which can create a unique visual effect.

MAE augmentation. The core idea behind MAE is to mask out parts of an image and then train an autoencoder to reconstruct the original image from the masked version. This is somewhat analogous to the masked language modeling task used in models like BERT for NLP, where parts of the text are masked out and the model is trained to predict the masked words.

D Appendix: Details of Biology Experiments

D.1 Dataset Setups

Experiments are performed on biological datasets, including MC1374[5] (Han et al., 2018), GA1457[6] (Rouillard et al., 2016), SAM[7] (Weber & Robinson, 2016), , and HCL500[8] (Han et al., 2020) datasets.

To ensure a fair comparison, we first embed the data into a 2D space using the method under evaluation. We then assess the method's performance through 10-fold cross-validation. Classification accuracy is determined by applying a linear SVM classifier in the latent space, while clustering accuracy is gauged using k-means clustering in the same space. Further details about the datasets, baseline methods, and evaluation metrics can be found in Table A.4.

[5] https://bis.zju.edu.cn/MCA/
[6] https://maayanlab.cloud/Harmonizome/gene/GAST
[7] https://github.com/abbioinfo/CyAnno
[8] https://db.cngb.org/HCL/

Table A.4: Datasets information of simple manifold embedding task

Dataset	Train Samples	Test Samples	Input Dimension	Class Number in label
MC1374	24,000	6,000	1,374	98
GA1457	8,510	2,127	1,457	49
SAM561	69,491	17,373	561	52
HCL500	48,000	12,000	500	45

D.2 Baseline Methods and Implementation Details

Dimension reduction methods that have been widely used on biological analyze are compared, including kPCA (Halko et al., 2010), Ivis (Szubert et al., 2019), PHATE (Moon & van Dijk, 2019), PUMAP (Sainburg et al., 2021), PaCMAP (Wang et al., 2022), DMTEV (Zang et al., 2022a) and hNNE (Sarfraz et al., 2022).

For DiffAug, both the semantic encoder Enc$(\cdot)$, and the diffusion generator Gen$(\cdot)$, are implemented using a Multi-Layer Perceptron (MLP). Their respective architectures are defined as: Enc$(\cdot)$: [-1, 500, 300, 80]. The Gen$(\cdot)$: is defined below,

Listing 2: DiffusionModel for Biology Task

```
    class AE(nn.Module):
    def __init__( self,in_dim, mid_dim=2000, time_step=1000,):
        super().__init__()
        self.enc1 = self.diff_block(in_dim, mid_dim)
        self.enc2 = self.diff_block(in_dim, mid_dim)
        self.enc3 = self.diff_block(in_dim, mid_dim)
        self.enc4 = self.diff_block(in_dim, mid_dim)

        self.dec1 = self.diff_block(in_dim, mid_dim)
        self.dec2 = self.diff_block(in_dim, mid_dim)
        self.dec3 = self.diff_block(in_dim, mid_dim)
        self.dec4 = self.diff_block(in_dim, mid_dim)
        self.time_encode = nn.Embedding(time_step, in_dim)

    def diff_block(in_dim, mid_dim):
        return nn.Sequential(
        nn.LeakyReLU(), nn.InstanceNorm1d(in_dim),
        nn.Linear(in_dim, mid_dim), nn.LeakyReLU(),
        nn.InstanceNorm1d(mid_dim), nn.Linear(mid_dim, in_dim),)

    def forward(self, input, time, cond=None):
        input_shape = input.shape
        if len(input.size()) > 2:
            input = input.view(input.size(0), -1)
        ti = self.time_encode(time)
        cd = self.cond_model(cond).reshape(input.shape[0], -1)
        ee1 = self.enc1(input + ti + cd)
        ee2 = self.enc2(ee1 + ti+ cd) + ee1
        ee3 = self.enc3(ee2 + ti+ cd) + ee1 + ee2
        ee4 = self.enc4(ee3 + ti+ cd) + ee1 + ee2 + ee3

        ed1 = self.dec1(ee4 + ti+ cd)
        ed2 = self.dec2(ed1 + ti+ cd) + ee3 + ed1
        ed3 = self.dec3(ed2 + ti+ cd) + ee2 + ed1 + ed2
        ed4 = self.dec4(ed3 + ti+ cd) + ee1 + ed1 + ed2 + ed3
        return ed4.reshape(input_shape)
```

To assess the efficacy of the proposed methods, following Wang et al. (2022); Sarfraz et al. (2022), we utilized linear SVM performance to evaluate the performance of differences methods. For the linear SVM evaluation, embeddings were partitioned with 90% designated for training and 10% for

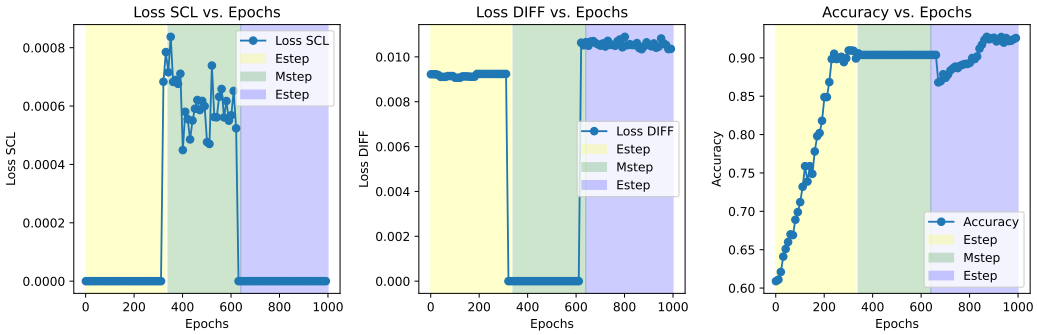


Figure A.2: Training curves on the GA1457 dataset, including two Esteps and one Mstep. We can observe that the new generated data improves the correctness of E step.

testing; the training set facilitated the linear SVM training, while the test set yielded the performance metrics. Detailed specifics of this configuration are elaborated in the Table A.5.

Table A.5: Details of the training process in biological dataset.

CF10	ν	Learning Rate	Weight Decay	Batch Size	GPU	Training Time
MC1374	1	0.0001	1e-6	300	1*V100	4.2 hours
GA1457	1	0.0001	1e-6	300	1*V100	4.6 hours
SAM561	1	0.0001	1e-6	300	1*V100	12.1 hours
HCL500	0.1	0.0001	1e-6	300	1*V100	20.1 hours