OXFORD

Briefings in Bioinformatics, 2025, **26(1)**, bbae643
https://doi.org/10.1093/bib/bbae643
Problem Solving Protocol

Few-shot classification of Cryo-ET subvolumes with deep Brownian distance covariance

Xueshi Yu[1,‡], Renmin Han[2,‡], Haitao Jiao[3], Wenjia Meng[1,*]

[1]School of Software, Shandong University, 1500 Shunhua Road, 250101 Jinan, China
[2]Research Center for Mathematics and Interdisciplinary Sciences, Shandong University, 72 Binhai Road, 266000 Qingdao, China
[3]Jinan Center For Disease Control And Prevention, Shandong University, 2 Weiliu Road, 250021 Jinan, China

*Corresponding author: Wenjia Meng, School of Software, Shandong University, 1500 Shunhua Road, 250101 Jinan, China. E-mail: wjmeng@sdu.edu.cn
‡These authors should be regarded as Joint First Author: Xueshi Yu and Renmin Han.

Abstract

Few-shot learning is a crucial approach for macromolecule classification of the cryo-electron tomography (Cryo-ET) subvolumes, enabling rapid adaptation to novel tasks with a small support set of labeled data. However, existing few-shot classification methods for macromolecules in Cryo-ET consider only marginal distributions and overlook joint distributions, failing to capture feature dependencies fully. To address this issue, we propose a method for macromolecular few-shot classification using deep Brownian Distance Covariance (BDC). Our method models the joint distribution within a transfer learning framework, enhancing the modeling capabilities. We insert the BDC module after the feature extractor and only train the feature extractor during the training phase. Then, we enhance the model's generalization capability with self-distillation techniques. In the adaptation phase, we fine-tune the classifier with minimal labeled data. We conduct experiments on publicly available SHREC datasets and a small-scale synthetic dataset to evaluate our method. Results show that our method improves the classification capabilities by introducing the joint distribution.

Keywords: Cryo-ET; macromolecules classification; few-shot learning; joint distribution

Introduction

Cryo-electron tomography (Cryo-ET is an important visualization technique that can observe macromolecules in a native cellular environment [1, 2]. Through subsequent analysis of tomograms using subtomogram averaging (STA) technology, the structures of *in situ* macromolecules can be accurately determined, enabling the prediction of the 3D structure and function of cells [3]. Among the STA processing steps (detection, classification, alignment, and averaging), macromolecule classification is particularly important as it directly affects the performance of downstream tasks. Current macromolecule classification methods can be broadly categorized into two types: template matching methods [4, 5] and deep learning-based methods. Specifically, template-matching methods classify macromolecules by computing a constrained cross-correlation between the data and known structures. However, template-matching methods suffer from reference-based biases and are sensitive to noise due to the dependency on templates. Deep learning-based methods [6, 7] improve classification accuracy by utilizing annotated data during the training process of deep neural networks. However, when encountering unseen macromolecule categories, deep learning-based methods need data re-annotation and model retraining, resulting in a time-consuming issue. In order to address this issue, few-shot learning recognizes new macromolecule categories by pre-training on a large dataset and quickly adapting to new tasks using a small support set of labeled examples.

Most macromolecule few-shot classification methods are based on metric learning, which learns image representations via similarity measures. These methods struggle to model high-dimensional feature distributions, as they rely on statistical moments. To address this, ProtoNet-CE [8] represents class prototypes in embedding space by mean vectors and predicts classes by calculating the Euclidean distance between query samples and prototypes. Few-Shot domain adaptation [9] focuses on reducing distribution differences by comparing covariance matrices of source and target domains. The few-shot learning method for the classification of novel macromolecules (named STA) [10] uses mean vectors for feature distribution representation during pre-training and approximate Gaussian distributions for distribution calibration during adaptation. However, these macromolecule few-shot classification methods only exploit marginal distributions while neglecting joint distributions, thereby limiting the performance of learned models.

To address the above issue, we propose a macromolecular few-shot classification method with deep Brownian distance covariance. This method utilizes Brownian distance covariance for joint distribution modeling at a lower cost and performs classification within a transfer learning framework. First, we pre-train the model using abundant macromolecular structures, leveraging the Brownian distance covariance matrix to model the joint distribution of the embedding space. Self-distillation techniques enhance the model's generalization capability. Finally,

Received: August 5, 2024. **Revised:** November 4, 2024. **Accepted:** November 27, 2024

we fine-tune the classifier with one or a few labeled data. Experiments on SHREC19, SHREC21, and a small-scale synthetic dataset demonstrate our method's superior modeling and generalization capabilities for macromolecular subvolumes and novel macromolecules. Our contributions are summarized as follows:

- We propose a macromolecule few-shot classification method based on deep Brownian distance covariance, incorporating joint distribution to address incomplete modeling of macromolecular structure embedding using only marginal distributions.
- Experiments on SHREC datasets and small-scale synthetic datasets show improved classification accuracy, with embedding space visualizations indicating better distinction of macromolecule data.

Related works

Macromolecule classification

Deep learning has achieved great success in macromolecule classification in recent years [11], which benefits from the powerful representational capabilities of deep neural networks. Current deep learning-based macromolecule classification methods can be categorized into two types: supervised learning-based methods and unsupervised learning-based methods. In the supervised learning-based category, macromolecule classification methods make use of supervised learning to achieve the classification task. Specifically, Chen *et al.* [12] simplify the complexity of neural networks by operating on tomograms slice by slice. Li *et al.* [13] further reduce the computational cost by utilizing Faster R-CNN. Subsequent supervised learning-based methods [6, 7] change the input to 3D subtomograms to allow for efficient processing without losing information. In the unsupervised learning-based category, macromolecule classification methods adopt unsupervised learning in the classification task. Several works [14, 15] are based on either clustering or geometric matching. Another unsupervised learning-based work [16] uses an orthogonal kernel initializer and zero bias initializer to improve training stability and greatly enhance processing in the macromolecule classification task. Different from these methods, this paper focuses on few-shot macromolecule classification, which utilizes few-shot learning to achieve rapid adaptation in macromolecule classification.

Few-shot classification

The emergence of few-shot classification is inspired by the rapid learning ability of humans. After learning from a large amount of data in certain categories, models require only a few samples to quickly learn new categories. In the field of computer vision, few-shot learning has flourished and can be roughly categorized into three classes. In the first class, model-based methods aim to directly establish a mapping function between input and prediction values through the design of model structure [17, 18]. In the second class, optimization-based methods [19] adjust the methods of optimization to accomplish the task of few-shot classification instead of the ordinary gradient descent methods, such as the Model-Agnostic-Meta-Learning(MAML) [20, 21] and the multidomain few-shot network [22]. In the third class, metric-based methods [23, 24] classify by measuring the distance between samples in the batch and samples in the support set with the nearest neighbor idea such as Siamese Netowrk [25], Bai *et al.*[26], and Liu *et al.* [27]. Our method follows the third class of few-shot learning but focuses on the macromolecule classification task instead of the traditional computer vision task.

Macromolecule few-shot classification

Macromolecule few-shot classification methods require only one or a few labeled large-molecule data to achieve rapid and accurate classification for new categories, making it highly suitable for the needs of large-molecule classification. Specifically, Li *et al.* [8] first apply few-shot learning to subtomogram classification, proposing a novel ProtoNet-CE model that integrates task-agnostic and task-specific embedding spaces to achieve more accurate classification. Then, Yu *et al.* [9] adopt a few-shot domain adaptation method, fully utilizing the distribution of abundant unlabeled target domain data and the correlation between the entire source domain dataset and a small number of labeled target domain data, enabling the model to adapt to the feature distribution of target domain. Recently, Gao *et al.* [10] proposed FSCC, which was based on the transfer learning method and utilizes distribution calibration in the macromolecule few-shot classification task. Different from the above methods, our approach introduces joint distribution for the first time, fundamentally improving the construction of the embedding space.

Distillation techniques

Knowledge distillation is a popular technique for model compression and knowledge transfer in deep learning, which has been widely applied in various fields, e.g. multi-task learning [28], generative adversarial problems [29], and medical image segmentation [30]. Current knowledge distillation methods can be divided into traditional distillation and self-distillation according to whether their student model and teacher model share the same network architecture. Traditional distillation methods transfer knowledge of the teacher model into a different student model to distill knowledge [31–35]. Self-distillation methods use the same network architecture for both the teacher model and student model and distill knowledge between them [36–38]. Different from existing distillation methods, we introduced self-distillation into the macromolecule few-shot classification tasks for the first time.

Preliminaries and notations

In this section, we introduce the joint distribution function, marginal distribution function, and the joint distribution based on the Brownian Distance Covariance (BDC) metric. To ensure clarity of notation, we provide a correspondence of notations in Appendix Table A.1.

Let $\mathbf{X} \in \mathbb{R}^p$ and $\mathbf{Y} \in \mathbb{R}^q$ be random vectors of dimensions p and q, respectively, with joint probability density function ($f_{\mathbf{XY}}(x, y)$). The joint characteristic function [39] of $\mathbf{X}$ and $\mathbf{Y}$ is defined as

$$\phi_{\mathbf{XY}}(\mathbf{t}, \mathbf{s}) = \int_{\mathbb{R}^p} \int_{\mathbb{R}^q} \exp\left(i(\mathbf{t}^T x + \mathbf{s}^T y)\right) f_{\mathbf{XY}}(x, y)\, dx\, dy, \tag{1}$$

where i is the imaginary unit, $\mathbf{t} \in \mathbb{R}^p$ and $\mathbf{s} \in \mathbb{R}^q$ are variables of the Fourier transform, the marginal distributions of $\mathbf{X}$ and $\mathbf{Y}$ are $\phi_{\mathbf{X}}(\mathbf{t}) = \phi_{\mathbf{XY}}(\mathbf{t}, \mathbf{0})$ and $\phi_{\mathbf{Y}}(\mathbf{s}) = \phi_{\mathbf{XY}}(\mathbf{0}, \mathbf{s})$, where $\mathbf{0}$ is a vector whose elements are all zero. According to probability theory, $\mathbf{X}$ and $\mathbf{Y}$ are independent if and only if $\phi_{\mathbf{XY}}(\mathbf{t}, \mathbf{s}) = \phi_{\mathbf{X}}(\mathbf{t})\phi_{\mathbf{Y}}(\mathbf{s})$.

The BDC metric [40] can represent the Euclidean distance between joint characteristic functions of random variables and the product of their marginal distributions. Provided $\mathbf{X}$ and $\mathbf{Y}$ have finite first moments, the joint characteristic function in Equation (1) can be defined as BDC metric:

$$\rho(\mathbf{X}, \mathbf{Y}) = \int_{\mathbb{R}^p} \int_{\mathbb{R}^q} \frac{|\phi_{\mathbf{XY}}(\mathbf{t}, \mathbf{s}) - \phi_{\mathbf{X}}(\mathbf{t})\phi_{\mathbf{Y}}(\mathbf{s})|^2}{c_p c_q \|\mathbf{t}\|_2^{1+p} \|\mathbf{s}\|_2^{1+q}} d\mathbf{t} d\mathbf{s}, \tag{2}$$

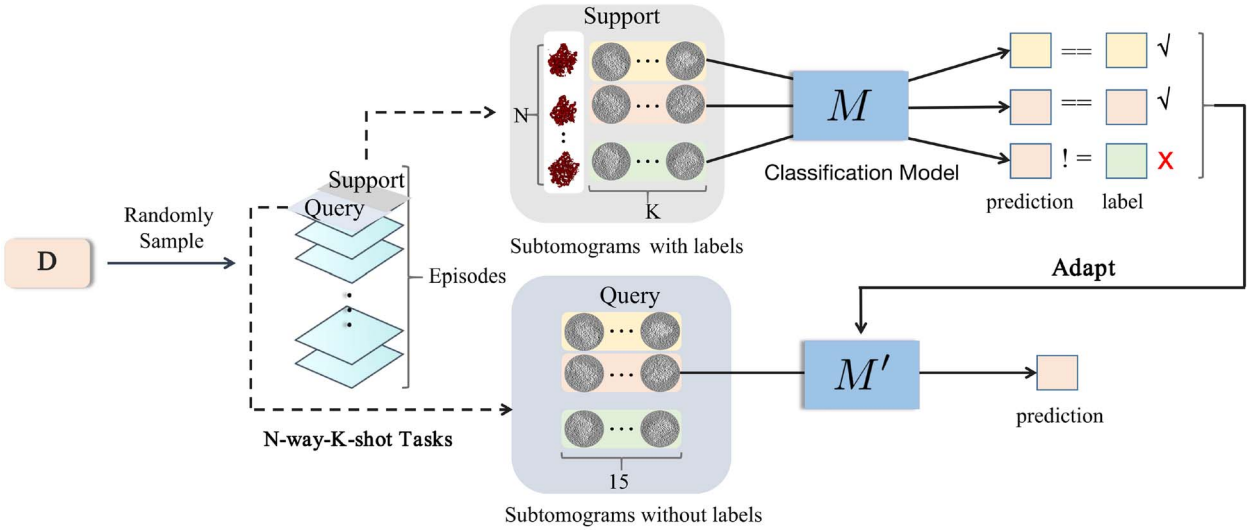

Figure 1. Flowchart of the macromolecular few-shot classification test: multiple N-way-K-shot tasks are randomly sampled from dataset *D*. Each task includes a labeled support set (*Support*) and an unlabeled query set (*Query*). The pre-trained classification model is adapted for predicting the query data using few support data. Both support data and query data belong to the novel class.

where $\|\cdot\|_2$ denotes the Euclidean norm, $c_p = \frac{\pi^{(1+p)/2}}{\Gamma((1+p)/2)}$, and Γ is the complete gamma function. For discrete observed values, the joint distribution between $\mathbf{X}$ and $\mathbf{Y}$ in Equation (2) can be expressed as a closed-form:

$$\rho(\mathbf{X}, \mathbf{Y}) = \mathrm{tr}(\mathbf{A}^T\mathbf{B}), \tag{3}$$

where $\mathrm{tr}(\cdot)$ denotes the matrix trace, $\mathbf{A}$ and $\mathbf{B}$ are the BDC matrices [39] of $\mathbf{X}$ and $\mathbf{Y}$, respectively. The joint distribution function in Equation (3) can be reformulated by vectorizing the BDC matrices ($\mathbf{A}$ and $\mathbf{B}$) into vectors ($\mathbf{a}$ and $\mathbf{b}$):

$$\rho(\mathbf{X}, \mathbf{Y}) = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T\mathbf{b}. \tag{4}$$

The joint distribution function $\rho(\mathbf{X}, \mathbf{Y})$ is invariant to translation and orthogonal transformations of inputs $\mathbf{X}$ and $\mathbf{Y}$, and equivariant to scaling operations.

Proposed method

Problem formulation of macromolecule few-shot classification

The macromolecule few-shot classification aims to accurately classify new categories with minimal new macromolecule observations after learning from a certain number of annotated samples. The problem of macromolecule few-shot classification can be formulated as two stages: a pre-training stage and an adaptation stage.

In the pre-training stage, the macromolecule classification model is pre-trained according to the tomographic images in the training set, which contains base class macromolecules (C_{base}). In the adaptation stage, as shown in Fig. 1, the classification model is adjusted according to the annotated macromolecules from a support set *Support*. The macromolecule few-shot classification task is called 'N-way K-shot task' when *Support* contains *N* classes, each with *K* macromolecule samples. In the macromolecule few-shot classification task, the query set *Query* consists of a certain number (usually set to 15) of unlabeled large molecules. Both *Query* and *Support* share the same label space of novel classes (C_{novel}), which satisfies

$$C_{\text{base}} \cap C_{\text{novel}} = \emptyset. \tag{5}$$

Equation (5) means that the categories of macromolecules in the test set have never appeared in the training set.

Current macromolecule few-shot classification methods only focus on utilizing the marginal distributions to represent features in embedding spaces without considering joint distributions. In the macromolecule few-shot classification methods, establishing a well-defined embedding space is crucial. The input features $\mathbf{X}$ are the original data mapped into this space, where the prototype[1] $\mathbf{Y}$ represents the mean of all features within a class, thus relating to class information. Modeling the joint distribution between $\mathbf{X}$ and $\mathbf{Y}$ is appealing, as it helps create an embedding space to better distinguish data with different classes, thereby improving classification accuracy.

Our framework

In order to introduce such joint distribution into the macromolecule few-shot classification, we propose a few-shot classification method of Cryo-ET subvolumes with deep Brownian distance covariance. In the following, we first introduce the network architecture of our method in detail (see Fig. 2). We then introduce three critical stages of our method: the pre-training stage, the self-distillation stage, and the adapting and evaluation stage (see Fig. 3).

Network architecture

The network architecture (shown in Fig. 2) in our method consists of the feature extractor f_ξ, the BDC module (BDC_φ), similarity calculation, and the classifier. In the following, we separately detail these components.

Feature Extractor. Given the dataset $\mathcal{D} = \{(z_i, y_i)\}_{i=1}^{N_\mathcal{D}}$, where $z_i \in \mathbb{R}^{H\times W\times L}$ is the input with height *H*, width *W* and length *L*,

[1] The prototype in one class is obtained by mapping samples in this class into a nonlinear embedding feature space and then calculating their mean value.

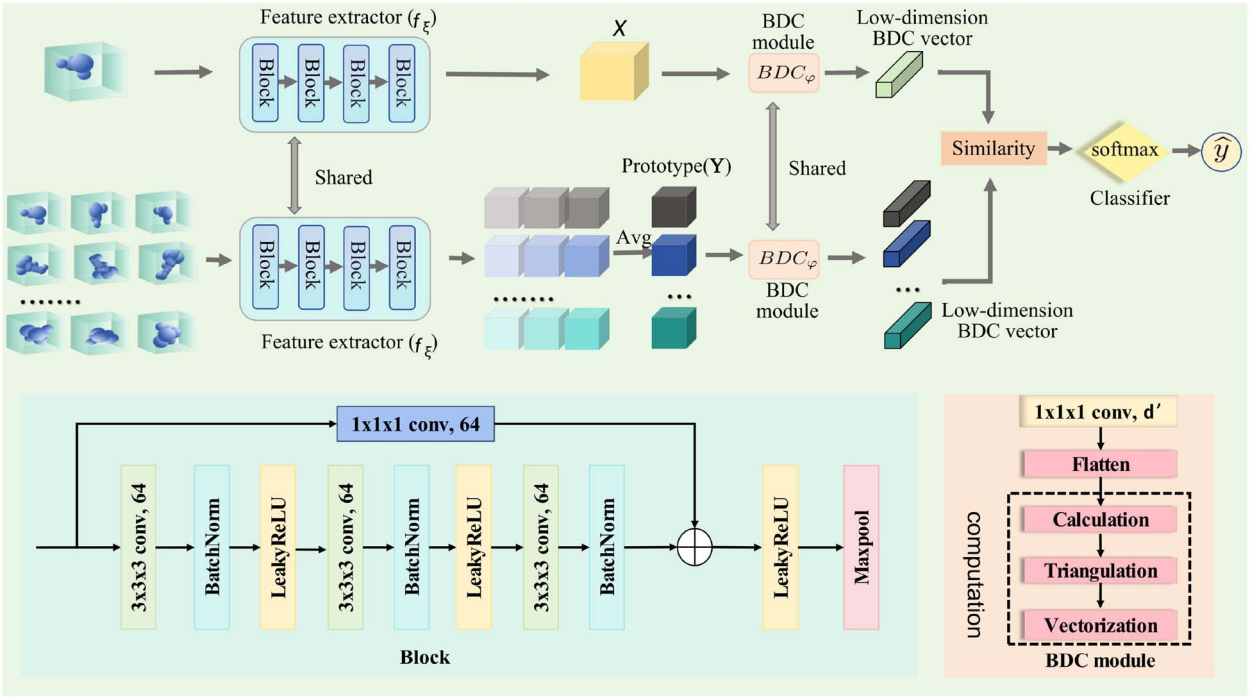

Figure 2. The network architecture of our method, which consists of feature extractor and BDC module, similarity calculating process, and classifier.

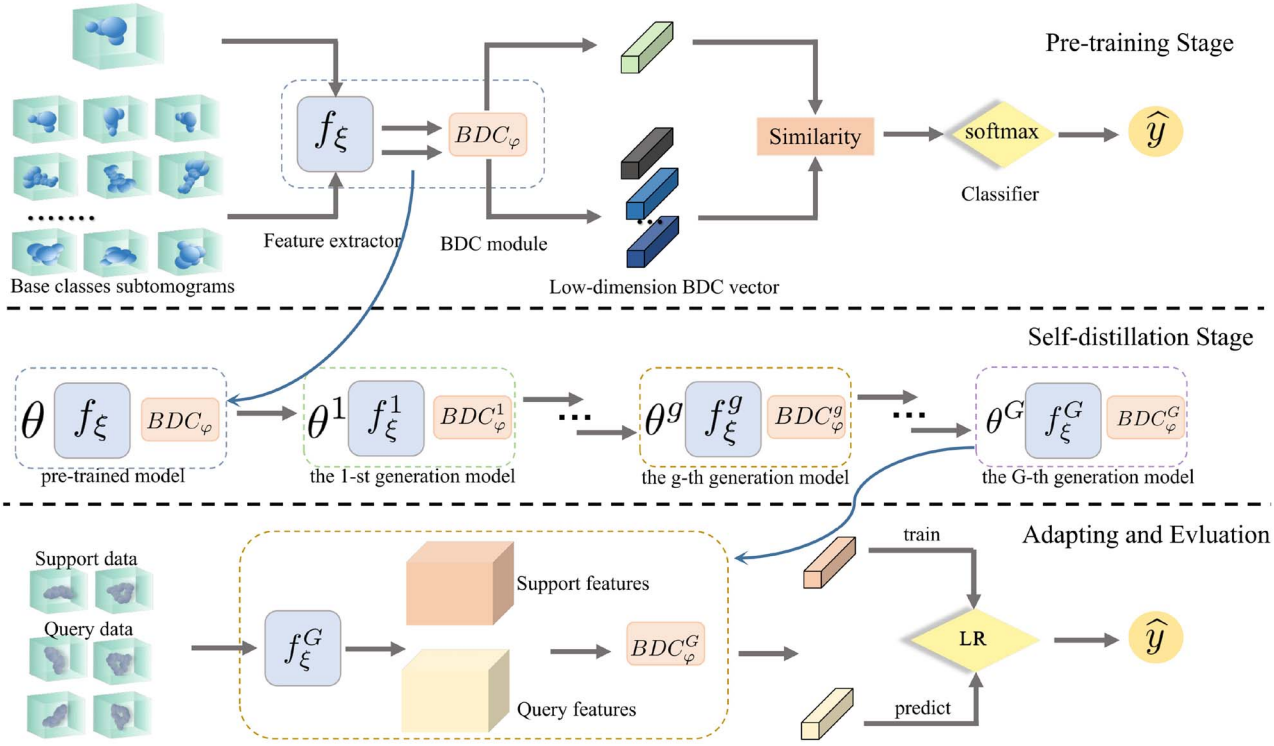

Figure 3. Our method's framework has three stages: pre-training, distillation, and adaptation. In pre-training, the model learns from labeled Base class data. The distillation stage uses self-distillation to enhance generalization, where the teacher model (earlier epochs) and student model share the same architecture. In the adaptation stage, a meta-learning algorithm is used. The model distilled for G generations computes BDC vectors of support and query data. An LR classifier is trained using support data and their labels, then used to classify the query data.

and $y_i \in \mathbb{R}$ is the class label, we adopt 3D-ResNet12 as the feature extractor (see Fig. 2). This model consists of four residual blocks, each containing three convolutional layers. The convolutional layers use 3×3×3 kernels, followed by LeakyReLU activation and Batch Normalization. The number of channels in each block is 64, 128, 256, and 640, respectively, and the number of channels for the convolutional kernels matches that of their corresponding blocks. Our model employs skip connections to capture residual information, where the input of a layer is directly added to the output of a later layer, bypassing a convolutional layer with a

1×1×1 kernel. The feature extractor f_ξ takes $z \in \mathcal{D}$ as input and outputs the feature $\mathbf{X} \in \mathbb{R}^{H\times W\times L\times d}$:

$$\mathbf{X} = f_\xi(z), \tag{6}$$

where H, W, L, and d represent the height, width, length, and number of channels, respectively.

BDC module. The BDC module (see Fig. 2) consists of three layers: a 3D convolutional layer with a $1\times1\times1$ kernel, a flattened layer, and a computation layer. Different from the 2D DeepBDC, we extend the BDC module to 3D subtomogram data with high dimensions and low signal-to-noise ratios and change the 2D convolution to 3D convolution in the convolutional layers. In our BDC module, taking input feature $\mathbf{X} \in \mathbb{R}^{H\times W\times L\times d}$ as input, a 3D convolutional layer reduces its fourth dimension d to d' and the flatten layer subsequently flattens the feature tensor channel-wise to output feature matrix $\mathbf{X}' \in \mathbb{R}^{(H*W*L)\times d'}$:

$$\mathbf{X} \in \mathbb{R}^{H\times W\times L\times d} \rightarrow \mathbf{X}' \in \mathbb{R}^{(H*W*L)\times d'}. \tag{7}$$

Following these above layers, the computation layer comprises three processes: calculation, triangulation, and vectorization. Based on feature matrix $\mathbf{X}'$ in Equation (7), the computation layer in the calculation process firstly acquires an Euclidean distance matrix $\widehat{\mathbf{A}} \in \mathbb{R}^{d'\times d'}$:

$$\widehat{\mathbf{A}}_{k,l} = \|\mathbf{X}'_{:,k} - \mathbf{X}'_{:,l}\|_2, \tag{8}$$

where $\widehat{\mathbf{A}}_{k,l}$ is the value at the kth row and the lth column of $\widehat{\mathbf{A}}$, $\mathbf{X}'_{:,k}$ and $\mathbf{X}'_{:,l}$ are separately the kth column vector and lth column vector of $\mathbf{X}'$, $\|\cdot\|_2$ is Euclidean norm. With above $\widehat{\mathbf{A}}$, the BDC matrix $\mathbf{A}$ can be obtained by the computation layer:

$$\mathbf{A}_{k,l} = \widehat{\mathbf{A}}_{k,l} - \frac{1}{d'}\sum_{k=1}^{d'}\widehat{\mathbf{A}}_{k,l} - \frac{1}{d'}\sum_{l=1}^{d'}\widehat{\mathbf{A}}_{k,l} - \frac{1}{d'^2}\sum_{k=1}^{d'}\sum_{l=1}^{d'}\widehat{\mathbf{A}}_{k,l}, \tag{9}$$

where $\mathbf{A}_{k,l}$ is the value at the kth row and the lth column of $\mathbf{A}$, the last three terms indicate means of the lth column, kth row and all entries of $\widehat{\mathbf{A}}$. Finally, after triangularization and vectorization processes, the BDC module outputs the BDC vector $\mathbf{a} \in \mathbb{R}^{\frac{d'*(d'+1)}{2}}$:

$$\mathbf{A} \in \mathbb{R}^{d'\times d'} \xrightarrow{\text{triang, vector}} \mathbf{a} \in \mathbb{R}^{\frac{d'*(d'+1)}{2}}, \tag{10}$$

where 'triang' and 'vector' denote the triangularization and vectorization processes, respectively.

Similarity calculation. With the above feature extractor and BDC module, we introduce how to build the joint distribution between input feature $\mathbf{X}$ and prototype $\mathbf{Y}$ and use their joint distribution function to measure their similarity.

Given a dataset $\mathcal{D} = \{(z_i, y_i)\}_{i=1}^{N_\mathcal{D}}$, we can sample input and class label pairs (z, y) and extract the input feature $\mathbf{X}$ according to Equation (6). To distinguish $\mathbf{X}$ with different input z, we rewrite the feature $\mathbf{X}$ as $\mathbf{X}_\xi(z)$ where ξ is the parameter of feature extractor. Based on $\mathbf{X}_\xi(z)$, the prototype $\mathbf{Y}$ [24] of the class c can be calculated by

$$\mathbf{Y}_c = \frac{1}{N_c}\sum_{(z_i,y_i)\in\mathcal{D}_c}\mathbf{X}_\xi(z_i), \tag{11}$$

where $\mathbf{Y}_c \in \mathcal{R}^{H\times W\times L\times d}$, $\mathcal{D}_c$ denotes the dataset belonging to class c in $\mathcal{D}$, N_c denotes the size of $\mathcal{D}_c$, and $\mathbf{Y}_c$ contains the label information of class c.

To capture the joint distribution between input feature $\mathbf{X}_\xi(z)$ and prototype $\mathbf{Y}_c$, we use BDC module to produce their BDC vectors according to Equation (10):

$$\mathbf{a}_z = BDC_\varphi(\mathbf{X}_\xi(z)), \tag{12}$$

$$\mathbf{b}_c = BDC_\varphi(\mathbf{Y}_c). \tag{13}$$

We can then formulate BDC metric-based joint distribution between $\mathbf{X}_\xi(z)$ and $\mathbf{Y}_c$ according to Equation (4):

$$\rho(\mathbf{X}_\xi(z), \mathbf{Y}_c) = \langle \mathbf{a}_z, \mathbf{b}_c\rangle = (\mathbf{a}_z)^T\mathbf{b}_c. \tag{14}$$

Because joint distribution in Equation (14) is a variant formulation of BDC metric in Equation (2), and the BDC metric as a distance metric [39] can measure the similarity, we use this joint distribution to measure the similarity between $\mathbf{X}_\xi(z)$ and $\mathbf{Y}_c$.

Classifier. We use the softmax as the classifier and the predicted probability for the jth class given a sample z is

$$P(y = j|z) = \frac{\exp((\mathbf{a}_z)^T\mathbf{b}_j)}{\sum_{k=1}^{C}\exp((\mathbf{a}_z)^T\mathbf{b}_c)}, \tag{15}$$

where C denotes the number of classes in dataset $\mathcal{D}$.

Training and evaluation stages

To train and evaluate our framework, we present three important stages: the pre-training stage, the self-distillation stage, and the adapting and evaluation stage.

Pre-training stage. In this stage, we use the whole base-class data in the training dataset ($\mathcal{D}_{train} = \{(z_i, y_i)\}_{i=1}^{N_{train}}$) to pre-train our model. Given input sample z_i, we aim to make the predicted label to be closer to the ground truth y_i and maximize the probability $P(y = y_i|z_i)$ in Equation (15), i.e.

$$\max \frac{\exp((\mathbf{a}_{z_i})^T\mathbf{b}_{y_i})}{\sum_{k=1}^{C}\exp((\mathbf{a}_{z_i})^T\mathbf{b}_c)}, \tag{16}$$

where C denotes the number of classes in dataset $\mathcal{D}_{train}$. Then, we formulate the following loss function to optimize our network:

$$\mathcal{L}(\mathcal{D}_{train}, \theta) = \arg\min_{\theta=\{\xi,\phi\}}\left[-\sum_{(z_i,y_i)\in\mathcal{D}_{train}}\log\frac{\exp(\tau(\mathbf{a}_{z_i})^T\mathbf{b}_{y_i})}{\sum_c\exp(\tau(\mathbf{a}_{z_i})^T\mathbf{b}_c)}\right], \tag{17}$$

where $\theta = \{\xi, \phi\}$ is the parameter set of our network, τ is a learnable scaling parameter. The parameters θ are then updated using gradient descent.

Self-distillation stage. Based on the above pre-training stage, we train our model and use this trained model to perform sequential self-distillation process [38] in the following. Before detailing the self-distillation process, we reformulate our whole model mainly comprising feature extractor and BDC module (see Fig. 2) as F_θ, for simplicity:

$$F_\theta(z) = BDC_\varphi(f_\xi(z)), \tag{18}$$

where z is input data, $\theta = \{\xi, \phi\}$ denote the parameter set of our model and has been optimized by loss in Equation (17).

In the self-distillation process, we initialize a student model $F_{\theta'}$ with the same architecture as the pre-trained model F_θ by minimizing both the loss in Equation (17) and Kullback–Leibler (KL) divergence.

$$\theta^{'} = \arg\min_{\theta'}(\alpha\mathcal{L}((z_i, y_i); \theta^{'}) + \beta \mathrm{KL}(F_{\theta'}(z_i), F_\theta(z_i))), \tag{19}$$

where (z_i, y_i) is the sample in $\mathcal{D}_{train}$, $\theta = \{\xi, \phi\}$ is the parameter set of our network, $\theta^{'}$ represents the parameters of student model, $\mathcal{L}((z_i, y_i); \theta^{'})$ represents the loss function Equation (17) using data (z_i, y_i), and α and β are balancing coefficients, typically set to $\alpha = \beta = 0.5$.

We applied distillation on the base-class data because we use the base-class data to pre-train our model, and as noted in [31], knowledge distillation using the same data as the pre-training stage positively impacts performance. Using the Born-again strategy [38], knowledge distillation is sequentially applied to generate multiple generations, where the maximum generation number is G. In each distillation step, the model in gth generation is trained using the knowledge of the nested model in $(g-1)$th generation:

$$\theta^g = \arg\min_{\theta}(\alpha\mathcal{L}((z_i, y_i); \theta^g) + \beta \mathrm{KL}(F_{\theta^g}(z_i), F_{\theta^{g-1}}(z_i))), \tag{20}$$

where $g = 2, \ldots, G$, θ^1 is distilled from the pre-trained model according to Equation (19), θ^g and θ^{g-1} separately represent the model parameters at g and $g-1$ iterations. The necessity of distillation in Cryo-ET lies in, compared to directly utilizing the pre-trained model, the distillation can enhance the model's representation and generalization capabilities to better classify the tomograph (with low signal-to-noise and complex noises in Cryo-ET). Distillation improves the capability of the network by leveraging the KL divergence as a regularization term to better distinguish different classes.

Adapting and evaluation stage. In the adaptation phase, we adapt the model in an N-way-K-shot manner and adopt the logistic regression classifier as our classifier.

Given test dataset ($\mathcal{D}_{test}$), we draw samples from $\mathcal{D}_{test}$ and construct the support set (*Support*) and query set (*Query*). In the adapting stage, we feed the data in the support set (*Support*) into our model (feature extractor and BDC module) to produce the BDC vectors. We take these BDC vectors as input and use the logistic regression (LR) classifier to predict the probability for the jth class, given the BDC vector ($\mathbf{a}$):

$$P(y = j|\mathbf{a}, \mathbf{w}) = \frac{\exp(h_j)}{\sum_{k=1}^{N}\exp(h_k)}, \text{with} \mathbf{h} = \mathbf{w}^T\mathbf{a} \in \mathbb{R}^N, \tag{21}$$

where N denotes the number of classes in support set, $\mathbf{w}$ denotes the parameters of logistic regression classifier, and h_j and h_k denote the jth and kth elements of $\mathbf{h}$. The predicted label is determined as the index of the maximum value of the probability in Equation (21):

$$\hat{y} = \arg\max_{j\in\{1,2,\ldots,N\}} P(y = j|\mathbf{a}, \mathbf{w}). \tag{22}$$

With the BDC vector ($\mathbf{a}$), the ground truth label (y), and the probability in Equation (21), we optimize the logistic regression classifier using the following loss function:

$$\mathcal{L}_{LR}(\mathbf{a}, y) = -\sum_{k=1}^{N} I\{y = k\} \log P(y = k|\mathbf{a}, \mathbf{w}) + \lambda||\mathbf{w}||_2, \tag{23}$$

where $I\{\cdot\}$ is the indicator function which returns 1 when the condition is true, and 0 when the condition is false, λ is a parameter of the L2 regularization term ($||\mathbf{w}||_2$). In the evaluation stage, we take the data in the query set (*Query*) as input and use our whole model (feature extractor, BDC module, and logistic regression classifier) to predict their labels. With the predicted label and ground truth, we can evaluate the classification accuracy of our model in the query set.

Algorithm 1 The algorithm of Our Method

Require: Training set $\mathcal{D}_{train} = \{(z_i, y_i)\}_{i=1}^{N_{train}}$, learning rate η

Pre-training Stage:
Initial feature extractor f_ξ and BDC module BDC_φ
for n = 1, 2, 3 ..., N_{epochs} **do**
  for i = 1, 2, 3, ..., N_{train} **do**
    Clear gradients of the SGD optimizer.
    $\mathbf{a}_{z_i} = BDC_\varphi(f_\xi(z_i))$ according to Eq. (12)
    Calculate $\mathcal{L}((z_i, y_i), \theta)$ according to Eq. (17)
    $\theta = \theta - \eta \cdot \nabla_\theta \mathcal{L}((z_i, y_i), \theta)$
  end for
end for

Self-Distillation Stage:
Initialize student model F_θ^1 by Equation (19)
Initialize the SGD optimizer.
for $g = 2, 3, ..., G$ **do**
  for n = 1, 2, ..., N_{epochs} **do**
    for i = 1, 2, 3, ..., N_{train} **do**
      Clear gradients of the SGD optimizer
      Update θ^g according to Eq. (20)
    end for
  end for
end for

Adapting and Evaluation Stage:
Splitting test dataset $\mathcal{D}_{test}$ into support set (*Support*) and query set (*Query*)
for $n = 1, 2, \ldots, N_{adapt\_epoch}$ **do**
  for i = 1, 2, 3, ..., $N_{Support}$ **do**
    Obtain data (z_i, y_i) in support set (*Support*)
    Optimize the classifier by the loss in Equation (30)
  end for
end for
Predict data label in *Query* by Eq (22)
Evaluate the classification accuracy using data in *Query*

Pesudo-code of our method

To clearly understand the algorithmic process of our method, we provide a pseudo-code in Algorithm 1. This pseudo-code outlines the three stages of our method: the pre-training stage, the self-distillation stage, and the adapting and evaluation stage. In the pre-training stage, the feature extractor (f_ξ) and BDC module (BDC_φ) are initialized. For each epoch, we extract BDC vector $\mathbf{a}$ from the input data by Equation (12). We then calculate the loss by Equation (17), and update the model parameters θ. In the self-distillation stage, we distill our model F_θ for G generations according to Equation (19) and (20). In the adapting and evaluation

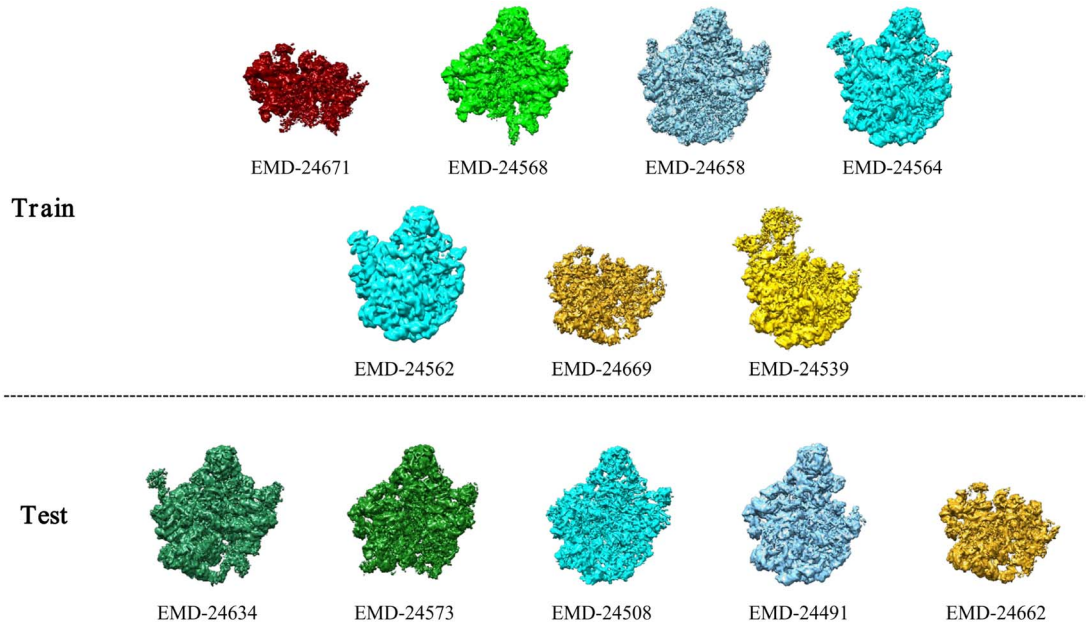

Figure 4. The density map of *Escherichia coli* 50S ribosome intermediates [41] from EMDB [42].

stage, a new classifier is trained on the support set (*Support*) using the loss function from Equation (23). The data labels in the query set (*Query*) are predicted using Equation (22), and the classification accuracy is subsequently evaluated based on these predictions.

Experimental results

In this section, we perform experiments to validate the effectiveness of our method. In the following, we first describe the datasets used in experiments, which include public datasets (SHREC19, SHREC21) and the synthetic dataset. We then introduce the experimental setup in detail. We next compare our method with the state-of-the-art (SOTA) methods to evaluate our method. Then we perform ablation experiments from the following aspects: dimension reduction parameter *d'*, the number of training samples, and the effect of BDC module, self-distillation, and BatchNorm layers. Finally, we analyze the embedding space visualizations to validate the effectiveness of our method in introducing joint distribution.

Datasets

In this section, we introduce the SHREC19 and SHREC21 datasets, as well as a small-scale synthetic dataset simulating ribosomal intermediates.

Public datasets. We validated our method using SHREC19 [6] and SHREC21 [7] datasets. These tomograms were segmented into $32 \times 32 \times 32$ voxels based on ground-truth information. SHREC19 contains 12 macromolecule classes totaling 20 785, with 7 base and 5 novel classes. SHREC21 includes 13 classes totaling 16 291, with 8 base and 5 novel classes.

Small-scale synthetic dataset. We created small-scale synthetic datasets with high similarity under various imaging conditions, as shown in Fig. 5. Using *Escherichia coli* 50S ribosome intermediates [41] from EMDB [42] (as shown in Fig. 4), we reconstructed 3D structures under different SNRs (0.1, 0.05, 0.02) and tilt angles (−60° to 60°, -50° to 50° with 1° intervals). We extracted 12 classes from assembly intermediates, each with 200 macromolecules, designating 7 as base and 5 as novel classes.

Experimental setup

We augmented the training set and sampled batches randomly. SHREC datasets input size was 32×32×32 with batch size 64, using ResNet12 for feature extraction. Training used the Cross-Entropy loss (Equation (17)), the Softmax classifier, and a SGD optimizer with a 0.05 learning rate. For the synthetic dataset, the input size was 64x64x64, the batch size was reduced to 32, and the learning rate was adjusted to 0.03. The self-distillation process involved three 100-epoch distillations, consistent with the pre-trained model. During meta-testing, the distilled model parameters were fixed, and the classifier was PyTorch's logistic regression with L2 regularization. We performed five rounds of 5-way-1-shot and 5-way-5-shot tests, each with 2000 episodes, and calculated the mean and variance of the classification accuracy to evaluate the performance of our method.

Comparison with SOTA methods

To validate our method's effectiveness, we compared it against five baselines (Baseline [43], Baseline++ [43], Siamese [25], MAML [20], FSCC [10]) on SHREC19, SHREC21, and our synthetic datasets. 'Baseline' is a transfer learning method proposed in [43], and 'Baseline++' is the modified version of 'Baseline'.

As shown in Table 1, our method achieved 72.05% and 80.87% accuracy for 1-shot and 5-shot tasks on SHREC19, and 75.48%/83.62% on SHREC21. Compared to FSCC, our approach improved by 1.49%/3.84% on SHREC19 and 7.24%/7.55% on SHREC21. On the synthetic dataset, our method achieved the best performance in most cases as shown in Tables 2, 3, and Fig. 6. MAML fails to perform on datasets with low SNR, likely due to its weak modeling ability [20]. Siamese Network achieved optimal results on the datasets '50°, 0.05' and '50°, 0.02'. But this method is limited to 1-shot tasks due to its design of a pair-wise recognition approach. The classification accuracy decreased with lower SNR and narrower tilt angles, with tilt angles having a more pronounced effect. These improvements may be attributed to enhanced modeling capability through joint distribution and better generalization via self-distillation. In most cases, our

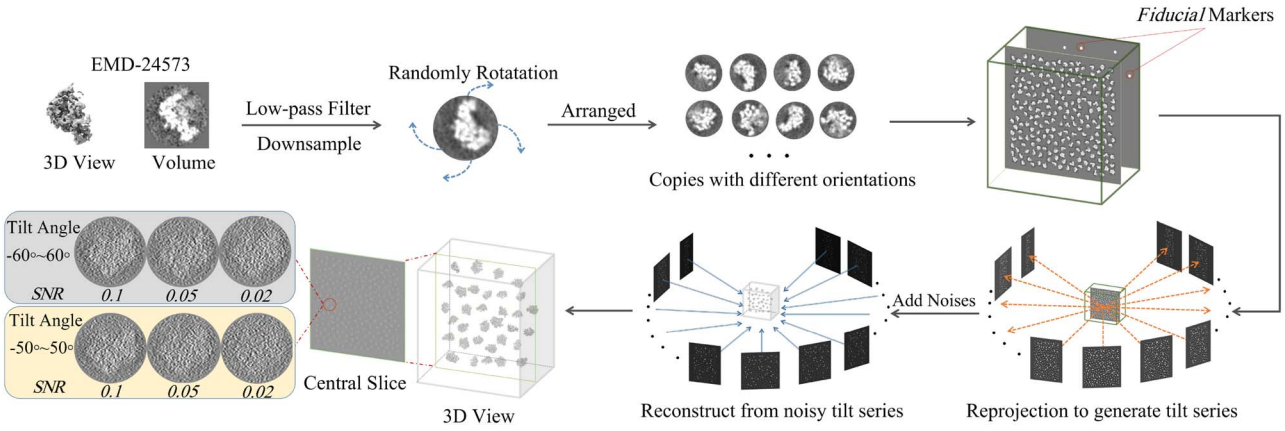

Figure 5. The process of creating a synthetic dataset using EMD-24573 as an example involves several steps. First, the volume from EMDB is preprocessed by downsampling and low-pass filtering. Random 3D rotations generate copies with different orientations. Then we place these copies and the fiducial makers within a tomogram. Reprojection sequences are computed at different tilt angles, and noise is added to each projection. The SART algorithm iteratively computes the 3D reconstruction. Finally, we present the 3D view and central slice of the reconstruction, as well as segmented macromolecules under different imaging conditions.

Table 1. The few-shot classification accuracy of our methods and SOTA methods on SHREC19 and SHREC21. Siamese is limited to 1-shot tasks due to its design of a pair-wise recognition approach. The optimal values in the tables of this paper are all in **bold**

	SHREC19		SHREC21	
Methods	**5-way-1-shot**	**5-way-5-shot**	**5-way-1-shot**	**5-way-5-shot**
Baseline	65.12% ± 0.74%	73.14% ± 0.47%	59.15% ± 1.70%	71.36% ± 1.19%
Baseline++	66.65% ± 0.67%	75.32% ± 0.37%	65.19% ± 1.02%	73.64% ± 1.04%
Siamese	52.76% ± 1.15%	None	50.97% ± 1.32%	None
MAML	68.90% ± 1.42%	76.32% ± 0.66%	64.74% ± 1.63%	73.19% ± 0.81%
FSCC	70.56% ± 1.61%	77.03% ± 1.21%	68.24% ± 1.03%	76.07% ± 1.03%
Ours	**72.05% ± 0.34%**	**80.87% ± 0.23%**	**75.48% ± 0.32%**	**83.62% ± 0.20%**

method surpasses SOTA macromolecule few-shot classification methods by leveraging joint distribution.

Ablation experiment

We conducted ablation experiments on the dimension reduction parameter d' (Table 4), the effect of the BDC module, the self-distillation (Table 5), and the number of the training samples (Fig. 7) using the SHREC19 and SHREC21 datasets.

Dimension Reduction Parameter d'. The BDC matrix size scales cubically with d', so decreasing d' reduces both model parameters and computational cost. With $d' = 128$, we achieved the highest accuracy, with time costs of 238ms/313ms (5-way-1-shot/5-way-5-shot, SHREC19) and 216ms/294ms (5-way-1-shot/5-way-5-shot, SHREC21). Higher d' values did not improve accuracy, likely due to overfitting. These results show that joint distributions help reduce time and computational costs, enhancing practical applicability.

Effect of BDC module. We conduct experiments comparing our method with the BDC module against the version without the BDC module. The comparison results (Table 5) show that using BDC improved accuracy on the SHREC19 dataset by 11.2%/7.31% (5-way-1-shot/5-way-5-shot) and on the SHREC21 dataset by 4.04%/2.95%. This demonstrates the effectiveness of the BDC module in enhancing feature representation.

Effect of self-distillation. To validate the effectiveness of self-distillation, we compared our method with self-distillation against the version without self-distillation (Table 5). Self-distillation technique achieved an improvement of 3.59%/2.24% (5-way-1-shot/5-way-5-shot) on the SHREC19 dataset. On the SHREC21 dataset, self-distillation on the SHREC21 dataset improved the 5-way 5-shot accuracy by 1.07%, while it slightly decreased in the 5-way 1-shot case. Overall, self-distillation generally enhances classification accuracy, proving to be effective in most scenarios.

Effect of BatchNorm layers. To validate the effect of layers and BatchNorm layers, we compared our method using BatchNorm layers against the version using layers. The results with BatchNorm layers showed improvements of 4.32%/5.03% (5-way-5-shot/5-way-1-shot) on the SHREC19 dataset and 5.30%/7.33% on the SHREC21 dataset. Batch normalization has a significant impact on the performance of few-shot classification models.

Number of training samples. To explore the effect of the training sample number, we conduct an ablation experiment using various training sample numbers (100, 200, 500, and 1000) per class from the training sets of SHREC19 and SHREC21. The results are shown in Fig. 7. It was observed that as the number of training samples increased, the classification accuracy exhibited an upward trend in most cases. The sampling number used in our method is the size of the dataset ('maximum' in Fig. 7 denotes this size), which helps achieve the best performance.

Ablation study for varying ways. To investigate the impact of the number of training classes and ways, we conducted ablation experiments on the SHREC19 and SHREC21 datasets using various numbers of training classes and ways. The results are shown in Fig. 8. Due to the limited number of classes, we set the way to match the number of test classes. The findings indicate that accuracy is generally higher when the way is smaller. Building on the above experimental results and considering that training classes are typically slightly more than test classes (ways), we

Table 2. The 5-way-1-shot classification accuracy on our small-scale synthetic dataset

	Tilt angles ± 60°			Tilt angles ± 50°		
SNR	**0.1**	**0.05**	**0.02**	**0.1**	**0.05**	**0.02**
Baseline	46.28%±0.51%	44.93%±0.52%	41.85%±0.48%	41.13%±0.53%	33.54%±0.48%	32.70%±0.52%
Baseline++	47.65%±0.40%	46.47%±0.41%	45.63%±0.42%	43.28%±0.41%	44.13%s±0.43%	38.23%±0.42%
Siamese	64.17%±0.67%	59.69%±0.72%	54.16%±0.66%	61.93%±0.73%	**61.19%±0.66%**	**56.64%±0.69%**
MAML	47.41%±1.75%	22.34%±1.72%	21.87%±1.76%	22.62%±1.79%	22.55%±1.75%	21.61%±1.74%
FSCC	63.44%±1.82%	61.57%±1.77%	55.84%±1.83%	53.77%±1.81%	48.21%±1.78%	46.37%±1.87%
Ours	**72.03%±0.20%**	**70.12%±0.21%**	**65.39%±0.21%**	**62.25%±0.20%**	57.68%±0.21%	54.05%±0.22%

Table 3. The 5-way-5-shot classification accuracy on our small-scale synthetic dataset

	Tilt angle ± 60°			Tilt angle ± 50°		
SNR	**0.1**	**0.05**	**0.02**	**0.1**	**0.05**	**0.02**
Baseline	50.96%±0.85%	49.88%±0.88%	47.38%±0.86%	45.89%±0.85%	38.49%±0.85%	35.93%±0.86%
Baseline++	54.79%±0.76%	53.65%±0.75%	52.69%±0.74%	52.50%±0.76%	51.06%±0.74%	41.94%±0.74%
MAML	56.98%±0.67%	54.10%±0.72%	22.37%±0.66%	22.52%±0.73%	22.13%±0.66%	21.90%±0.69%
FSCC	75.75%±1.13%	72.84%±1.21%	66.31%±1.16%	64.11%±1.18%	57.17%±1.10%	54.04%±1.13%
Ours	**87.18%±0.32%**	**84.86%±0.31%**	**82.23%±0.32%**	**78.86%±0.32%**	**74.57%±0.32%**	**68.76%±0.33%**

Table 4. Ablation analysis of our method with the backbone of ResNet-12. We conducted ablation experiments on the dimensionality reduction parameter d' using an NVIDIA A100 on the SHREC19 dataset, comparing the latency (ms), accuracy (%), and parameter count (Megabyte) for different values of d'

		5-way-1-shot		5-way-5-shot		
Dataset	d'	**Acc**	**Latency**	**Acc**	**Latency**	**Params**
SHREC19	512	67.83%±0.33%	1848	77.19%±0.22%	2734	37.04
	256	67.47%±0.34%	495	76.82%±0.22%	690	36.88
	196	68.99%±0.33%	364	77.94%±0.23%	485	36.84
	128	**72.05%±0.34%**	238	**80.87%±0.23%**	313	36.80
	64	67.17%±0.33%	**141**	75.93%±0.22%	**179**	**36.76**
SHREC21	512	74.61%±0.32%	1667	82.41%±0.21%	3058	37.04
	256	73.44%±0.33%	532	81.78%±0.21%	795	36.88
	196	73.66%±0.32%	404	82.06%±0.22%	576	36.84
	128	**75.48%±0.32%**	216	**83.62%±0.20%**	294	36.80
	64	72.63%±0.31%	**140**	81.42%±0.21%	**184**	**36.76**

Table 5. The results of the ablation experiments of our method on the SHREC19 and SHREC21 datasets

Method			**SHREC19**		**SHREC21**	
BDC	**BN**	**Distillation**	**5-way-1-shot**	**5-way-5-shot**	**5-way-1-shot**	**5-way-5-shot**
✓	✓		68.46% ± 0.32%	78.63% ± 0.22%	**75.78% ± 0.32%**	82.55% ± 0.22%
	✓	✓	60.85% ± 0.33%	73.56% ± 0.22%	71.44% ± 0.32%	80.67% ± 0.22%
✓	✓		67.02% ± 0.32%	76.55% ± 0.22%	68.15% ± 0.32%	78.32% ± 0.22%
✓	✓	✓	**72.05% ± 0.34%**	**80.87% ± 0.23%**	75.48% ± 0.32%	**83.62% ± 0.20%**

adopt the standard configuration of 7 training classes and 5-way testing in this paper.

Embedding space visualization analysis

Introducing joint distribution, our method exhibits enhanced capability in constructing embedding spaces. Leveraging t-SNE, we visualize the feature distributions in the embedding space. We load the optimal models of each transfer learning method, truncate them before the classifier, and input the features along with corresponding labels into the t-SNE model. Subsequently, we visualize the embedding spaces for five novel classes in the representative dataset (SHREC21), each containing approximately 2000 data points, as shown in Fig. 9. The models are trained using the Base class dataset, and truncating them before the classifier implies no adaptation to the novel classes, thus enabling a fair demonstration of the generalization ability of each method and

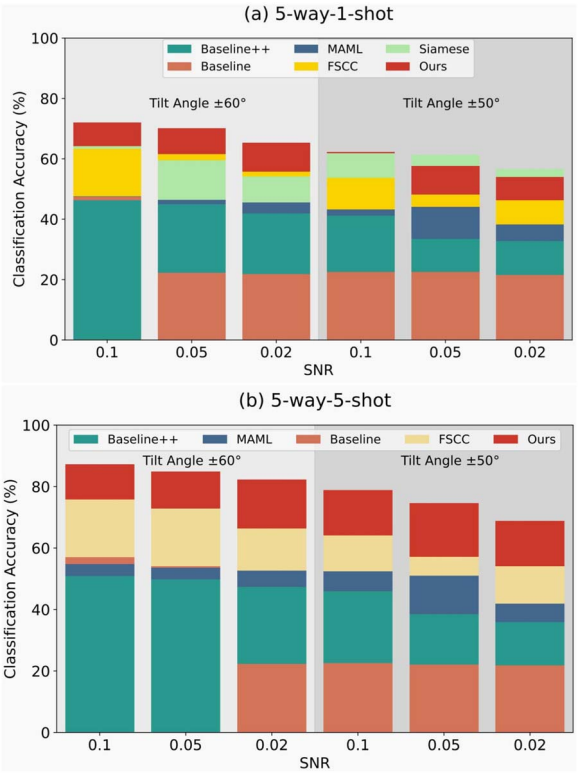

Figure 6. (a) and (b), respectively, show the 5-way-1-shot and 5-way-5-shot classification accuracy of our method and three baseline methods on the synthetic dataset. The x-axis represents data with varying SNRs at tilt angles of ± 60° and ± 50°. The height of each colored segment, starting from zero, represents the absolute accuracy of each method. Siamese is limited to 1-shot tasks due to its design of a pair-wise recognition approach.

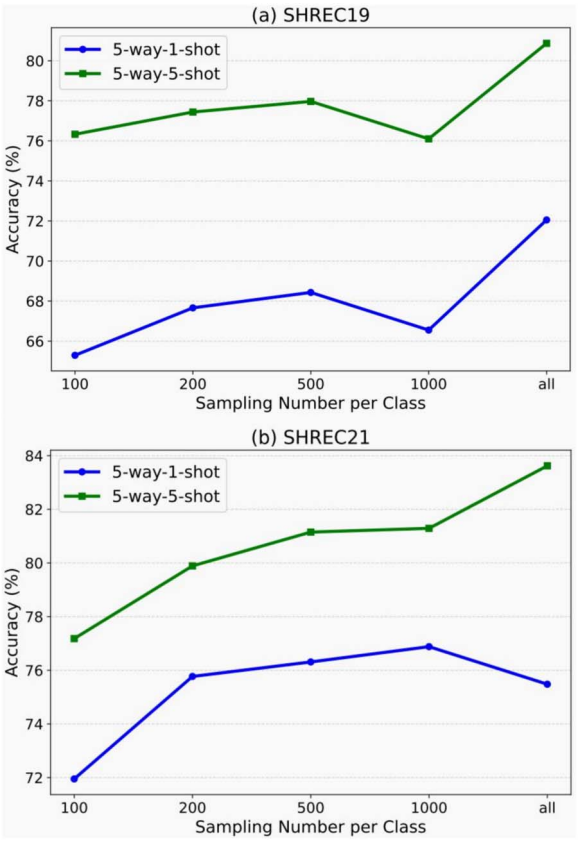

Figure 7. The ablation results of the training samples number in the pre-training stage on the SHREC19 and SHREC21 datasets. 'maximum' denotes the size of the dataset.

its suitability for few-shot classification tasks. Specifically, two classes *1U6G* and *3CF3* exhibit marginal overlap in other methods, and class *4V94* also overlaps with other classes. Conversely, our method demonstrates non-overlapping margins for classes *1U6G* and *3CF3*, with greater separation, while class *4V94* not only lacks overlap with other classes but also exhibits a smaller area. This indicates that our method achieves larger inter-class distances and more compact intra-class distributions, showcasing superior modeling capabilities.

Conclusion

Macromolecule classification is essential in Cryo-ET analysis, and few-shot learning is well-suited due to its ability to quickly adapt to novel macromolecules. Previous few-shot classification methods, however, relied on marginal distributions for embedding, which limits performance. In this work, we propose a few-shot classification method based on joint distributions. It employs a transfer learning framework and enhances model generalization through self-distillation. Experiments with synthetic datasets show that our method significantly enhances classification accuracy over SOTA methods. Visual comparisons of feature spaces reveal improved feature modeling, highlighting the effectiveness of joint distribution integration. Although we use a reasonable configuration of training class numbers (7) and N-way (5) in this paper, further exploration of different configurations in future work would be valuable.

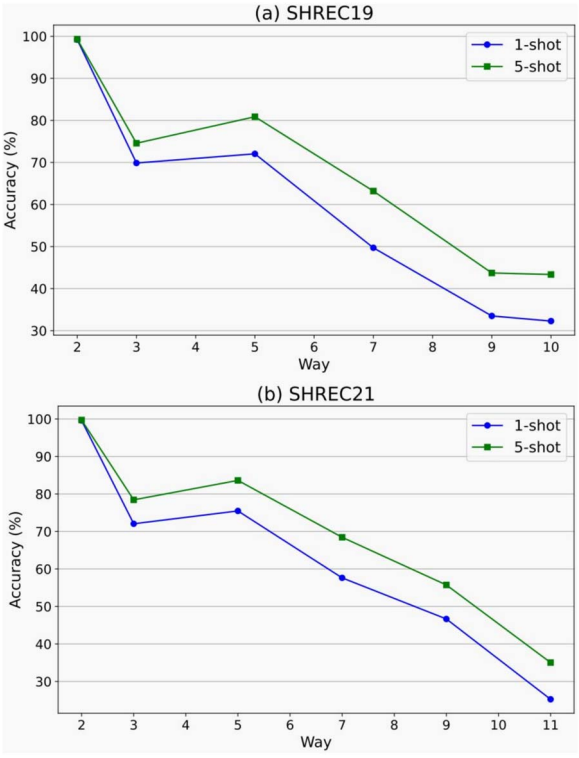

Figure 8. The ablation results of varying ways on the SHREC19 and SHREC21 datasets.

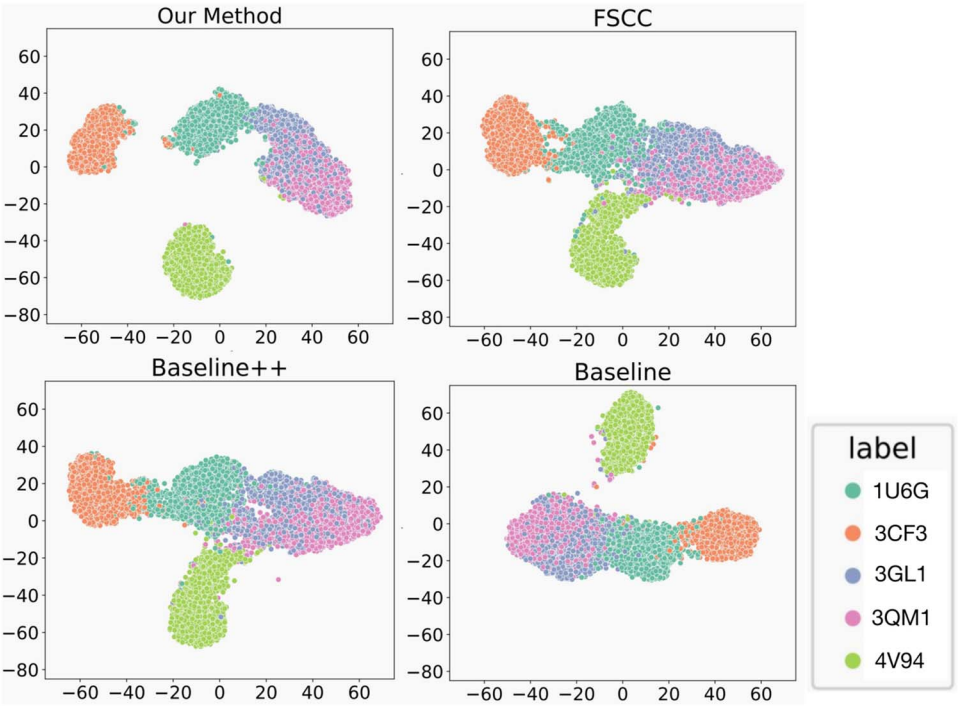

Figure 9. SHREC21 t-SNE embedding on Novel Class. We performed t-SNE visualization of the embedding space using four transfer learning methods, comprehensively covering five novel classes in SHREC21, with approximately 2000 data points per class.

Key Points

- We proposed a few-shot classification method of Cryo-ET subvolumes with deep Brownian Distance Covariance, introducing joint distribution for the first time to improve model capability and classification accuracy.
- We conducted experimental validation on two SHREC public datasets and a synthetic dataset. The visualization results demonstrated improved modeling capabilities.
- Our method meets the needs for rapid and accurate classification of novel macromolecules, avoiding the cost associated with extensive labeled data and model retraining.

Funding

This research was supported by the National Key Research and Development Program of China [2021YFF0704300], the National Natural Science Foundation of China [62206158, 62072280], the Natural Science Foundation of Shandong Province [ZR2022QF097, ZR2023YQ057], Shandong University Young Scholar Future Plan, and Instrument Improvement Funds of Shandong University Public Technology Platform [ts20230204], with the help of SDU's Biomedical Research Center for Structural Analysis.

References

1. Bharat TA, Scheres SH. Resolving macromolecular structures from electron cryo-tomography data using subtomogram averaging in relion. *Nat Protoc* 2016; **11**:2054–65. https://doi.org/10.1038/nprot.2016.124.
2. Wang Z, Grange M, Pospich S. *et al*. Structures from intact myofibrils reveal mechanism of thin filament regulation through nebulin. *Science* 2022; **375**:abn1934. https://doi.org/10.1126/science.abn1934.
3. Zhu X, Huang GX, Zeng C. *et al*. Structure of the cytoplasmic ring of the xenopus laevis nuclear pore complex. *Science* 2022; **376**:abl8280. https://doi.org/10.1126/science.abl8280.
4. Yu Z, Frangakis AS. Classification of electron sub-tomograms with neural networks and its application to template-matching. *J Struct Biol* 2011; **174**:494–504. https://doi.org/10.1016/j.jsb.2011.02.009.
5. Böhm J, Frangakis AS, Hegerl R. *et al*. Toward detecting and identifying macromolecules in a cellular context: template matching applied to electron tomograms. *Proc Natl Acad Sci* 2000; **97**:14245–50. https://doi.org/10.1073/pnas.230282097.
6. Gubins I, van der Schot G, RC. *et al*. Classification in Cryo-electron tomograms. In: Biasotti S, Lavoué G, Veltkamp R (eds.), *Eurographics Workshop on 3D Object Retrieval*. Eurographics, Pisa, Italy, 2019.
7. Gubins I, Chaillet ML, van der Schot G. *et al*. SHREC 2021: Classification in Cryo-electron Tomograms. In: *Eurographics Workshop on 3D Object Retrieval 2021*, pp. 5–17. The Eurographics Association, 2021. https://doi.org/10.2312/3DOR.20211307.
8. Li R, Yu L, Zhou B. *et al*. Few-shot learning for classification of novel macromolecular structures in cryo-electron tomograms. *PLoS Comput Biol* 2020; **16**:e1008227. https://doi.org/10.1371/journal.pcbi.1008227.
9. Yu L, Li R, Zeng XR. *et al*. Few shot domain adaptation for in situ macromolecule structural classification in cryoelectron tomograms. *Bioinformatics* 2021; **37**:185–91. https://doi.org/10.1093/bioinformatics/btaa671.
10. Gao S, Zeng X, Xu M. *et al*. FSCC: few-shot learning for macromolecule classification based on contrastive learning and distribution calibration in cryo-electron tomography. *Front Mol Biosci* 2022; **9**:931949. https://doi.org/10.3389/fmolb.2022.931949.

11. Förster F, Pruggnaller S, Seybert A. *et al*. Classification of cryo-electron sub-tomograms using constrained correlation. *J Struct Biol* 2008; **161**:276–86. https://doi.org/10.1016/j.jsb.2007.07.006.
12. Chen M, Dai W, Sun SY. *et al*. Convolutional neural networks for automated annotation of cellular cryo-electron tomograms. *Nat Methods* 2017; **14**:983–5. https://doi.org/10.1038/nmeth.4405.
13. Li R, Zeng X, Sigmund SE. *et al*. Automatic localization and identification of mitochondria in cellular electron cryo-tomography using faster-RCNN. *BMC Bioinform* 2019; **20**:75–85. https://doi.org/10.1186/s12859-019-2650-7.
14. Xu M, Chai X, Muthakana H. *et al*. Deep learning-based subdivision approach for large scale macromolecules structure recovery from electron cryo tomograms. *Bioinformatics* 2017; **33**:i13–22. https://doi.org/10.1093/bioinformatics/btx230.
15. Zeng X, Xu M. Gum-net: Unsupervised geometric matching for fast and accurate 3D subtomogram image alignment and averaging. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4073–84. IEEE, USA, 2020.
16. Zeng X, Kahng A, Xue L. *et al*. High-throughput cryo-et structural pattern mining by unsupervised deep iterative subtomogram clustering. *Proc Natl Acad Sci* 2023; **120**:e2213149120. https://doi.org/10.1073/pnas.2213149120.
17. Santoro A, Bartunov S, Botvinick M. *et al*. Meta-learning with memory-augmented neural networks. In: *International Conference on Machine Learning*, pp. 1842–50. PMLR, New York, USA, 2016.
18. Munkhdalai T, Yu H. Meta networks. In: *International Conference on Machine Learning*, pp. 2554–63. PMLR, Sydney, Australia, 2017.
19. Ravi S, Larochelle H. Optimization as a model for few-shot learning. In: *International Conference on Learning Representations*, 2016. San Juan, Puerto Rico, May 2–4, 2016. https://openreview.net/forum?id=rJY0-Kcll.
20. Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: *International Conference on Machine Learning*, pp. 1126–35. PMLR, Sydney, Australia, 2017.
21. Alsaleh AM, Albalawi E, Algosaibi A. *et al*. Few-shot learning for medical image segmentation using 3D U-Net and model-agnostic meta-learning (MAML). *Diagnostics* 2024; **14**:1213. https://doi.org/10.3390/diagnostics14121213.
22. Jha A, Banerjee B. Mdfs-net: Multi-domain few shot classification for hyperspectral images with support set reconstruction. *IEEE Trans Geosci Remote Sens* 2023; **61**:1–12. https://doi.org/10.1109/TGRS.2023.3324947.
23. Sung F, Yang Y, Zhang L. *et al*. Learning to compare: relation network for few-shot learning. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1199–208. IEEE, USA, 2018.
24. Snell J, Swersky K, Zemel R. Prototypical networks for few-shot learning. *Adv Neural Inf Process Syst* 2017; **30**.
25. Koch G, Zemel R, Salakhutdinov R. *et al*. Siamese neural networks for one-shot image recognition. In: *ICML Deep Learning Workshop*, Vol. **2**, pp. 1–30. JMLR, USA, 2015.
26. Bai J, Huang S, Zhu X. *et al*. Few-shot hyperspectral image classification based on adaptive subspaces and feature transformation. *IEEE Trans Geosci Remote Sens* 2022; **60**:1–17. https://doi.org/10.1109/TGRS.2022.3149947.
27. Liu B, Yu X, Yu A. *et al*. Deep few-shot learning for hyperspectral image classification. *IEEE Trans Geosci Remote Sens* 2018; **57**:2290–304. https://doi.org/10.1109/TGRS.2018.2872830.
28. Jha A, Kumar A, Banerjee B. *et al*. SD-MTCNN: self-distilled multi-task CNN. In: *BMVC*. BMVA, UK, 2020.
29. Liu P, Liu W, Ma H. *et al*. Ktan: Knowledge transfer adversarial network. In: *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, USA, 2020.
30. You C, Zhou Y, Zhao R. *et al*. SimCVD: simple contrastive voxel-wise representation distillation for semi-supervised medical image segmentation. *IEEE Trans Med Imaging* 2022; **41**:2228–37. https://doi.org/10.1109/TMI.2022.3161829.
31. Hinton G. Distilling the knowledge in a neural network. *arXiv preprint* arXiv:1503.02531. 2015. https://doi.org/10.48550/arXiv.1503.02531.
32. Romero A, Ballas N, Kahou SE. *et al*. Fitnets: Hints for thin deep nets. In: *International Conference on Learning Representations*, 2015. San Diego, California, May 7–9, 2015. https://doi.org/10.48550/arXiv.1412.655.
33. Passalis N, Tefas A. Learning deep representations with probabilistic knowledge transfer. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 268–84. Springer, Munich, Germany, 2018.
34. Yim J, Joo D, Bae J. *et al*. A gift from knowledge distillation: fast optimization, network minimization and transfer learning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4133–41. IEEE, USA, 2017.
35. Lee SH, Kim DH, Song BC. Self-supervised knowledge distillation using singular value decomposition. In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 335–50. Springer, Munich, Germany, 2018. https://doi.org/10.1007/978-3-030-01231-1_21.
36. Zhang L, Song J, Gao A. *et al*. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3713–22. IEEE, Seoul, South Korea, 2019.
37. Mary M. Phuong and Lampert CH. Distillation-based training for multi-exit architectures. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1355–64. IEEE, Seoul, South Korea, 2019.
38. Furlanello T, Lipton Z, Tschannen M. *et al*. Born again neural networks. In: *International Conference on Machine Learning*, pp. 1607–16. PMLR, Stockholm, Sweden, 2018.
39. Xie J, Long F, Lv J. *et al*. Joint distribution matters: deep Brownian distance covariance for few-shot classification. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7972–81. IEEE, USA, 2022.
40. Székely GJ, Rizzo ML. Brownian distance covariance. *Ann Appl Stat* 2009; **3**:1236–65. https://doi.org/10.1214/09-AOAS312.
41. Rabuck-Gibbons JN, Lyumkis D, Williamson JR. Quantitative mining of compositional heterogeneity in cryo-em datasets of ribosome assembly intermediates. *Structure* 2022; **30**:498–509.e4. https://doi.org/10.1016/j.str.2021.12.005.
42. The wwPDB Consortium. EMDB—the Electron Microscopy Data Bank. *Nucleic Acids Research*, Volume **52**, Issue D1, 5 January 2024, Pages D456–D465. Oxford University Press, United Kingdom. https://doi.org/10.1093/nar/gkad1019.
43. Chen WY, YC Liu, Kira Z. *et al*. A closer look at few-shot classification. In: *Proceedings of the 7th International Conference on Learning Representations*, 2019. New Orleans, Louisiana, May 6–9, 2019. https://openreview.net/forum?id=HkxLXnAcFQ.

A Appendix

Table A1. Notation table.

Notation	Meaning
$\mathbf{X}$, $\mathbf{Y}$	random vectors
$\phi_{\mathbf{XY}}$	the joint characteristic function of $\mathbf{X}$ and $\mathbf{Y}$
$\phi_{\mathbf{X}}$, $\phi_{\mathbf{Y}}$	the marginal distributions functions of $\mathbf{X}$ and $\mathbf{Y}$
ρ	the BDC metric
$\widehat{\mathbf{A}}$	the Euclidean distance matrix
$\mathbf{a}$, $\mathbf{b}$	the BDC vectors
T	matrix transpose process
$\mathrm{tr}(\cdot)$	matrix trace process
$\|\cdot\|_2$	Euclidean norm process
f_ξ	the feature extractor
BDC_φ	the BDC module
F_θ	our whole model comprising feature extractor f_ξ and BDC module BDC_φ
H, W, L, d	the height, width, length, and the channel of the feature output by the feature extractor
d'	the channel to which the features are reduced by the BDC module

Briefings in Bioinformatics, 2025, **26(1)**, bbae643
https://doi.org/10.1093/bib/bbae643
Problem Solving Protocol