Supplementary Materials for "Graph Cut-guided Maximal Coding Rate Reduction for Learning Image Embedding and Clustering"

A Experimental Details

Dataset description. In Table A.1, we provide an overview of of all selected datasets. The images in MNIST and F-MNIST are grayscale, and images of all datasets are resized to 224×224 dimensions to serve as the inputs of CLIP image encoder. For Oxford Flowers-102 and Stanford Dogs-120, we train and test our CgMCR^2 on the entire dataset. For all other datasets, we use the train set and test set for training and testing, respectively.

Table A.1: Specification of all selected datasets.

Dataset	# Classes	# Training	# Testing
MNIST	10	60,000	10,000
F-MNIST	10	60,000	10,000
CIFAR-10	10	50,000	10,000
CIFAR-20	20	50,000	10,000
CIFAR-100	100	50,000	10,000
Flowers-102	102	8,192	N/A
Dogs-120	120	20,580	N/A
TinyImageNet	200	100,000	10,000
ImageNet-1k	1000	1,281,167	50,000

Table A.2: Model parameters of the pre-feature layer, feature head, and cluster head (from left to right).

Linear: $\mathbb{R}^{768} \to \mathbb{R}^{4096}$
BatchNorm1d(4096)
ReLU

Linear: $\mathbb{R}^{4096} \to \mathbb{R}^{4096}$
ReLU
Linear: $\mathbb{R}^{4096} \to \mathbb{R}^{d}$

Linear: $\mathbb{R}^{4096} \to \mathbb{R}^{4096}$
ReLU
Linear: $\mathbb{R}^{4096} \to \mathbb{R}^{k}$
Gumbel-Softmax

Parameters for CgMCR^2. In Table A.2, we detail the model parameters of our framework. In Table A.3, we detail the optimal hyper-parameters for CgMCR^2. The proposed CgMCR^2 demonstrates robustness to variations in batch size, γ

Table A.3: Optimal hyper-parameters."lr" and "wd" are the learning rate and weight decay of Adam optimizer, d is the output dimension of feature head, T_1 denotes warm-up epochs, T_2 denotes fine-tuning epochs, γ and ϵ are the hyper-parameters of CgMCR^2 objective, and s is the number of nonzero affinity entries kept in each row.

Dataset	lr	wd	d	T_1	T_2	bs	γ	ϵ	s
MNIST	0.001	0.001	128	20	30	2048	50	0.5	20
F-MNIST	0.001	0.001	128	20	30	2048	50	0.2	20
CIFAR-10	0.0001	0.0005	128	10	10	512	70	0.5	10
CIFAR-20	0.0001	0.0005	128	10	40	1500	80	0.2	50
CIFAR-100	0.0005	0.0001	128	20	30	2048	1400	0.5	20
Flowers-102	0.0005	0.0005	128	20	30	2048	1200	0.5	10
Dogs-120	0.001	0.001	128	20	30	2048	1100	0.2	40
TinyImageNet	0.0003	0.0005	256	20	30	2048	3000	0.5	20
ImageNet	0.001	0.0001	256	10	10	3000	50000	0.2	3

Table A.4: Parameter search with the following parameters for Spectral Clustering, EnSC and SCAN.

Method	Search scope for parameters
Spectral Clustering	$\sigma \in \{3, 2, 1, 0.5, 0.4, 0.3, 0.2, 0.1, 0.07, 0.05\}$, $s \in \{3, 10, 100, 1000\}$
EnSC	$\tau \in \{0.9, 0.95, 1\}$, $\beta \in \{1, 2, 5, 10, 50, 100, 200\}$
SCAN	$\mu \in \{1, 2, 4, 10, 20, 50, 100, 200, 500, 1000, 2000\}$

and ϵ. Typically, employing a larger batch size along with a higher learning rate tends to yield more stable performance. Meanwhile, CgMCR^2 with larger batch size requires more training iteration to converge.

Searching parameters for clustering methods. When comparing with classical clustering methods and reproduced deep clustering methods, we report their best performance through a greedy search for optimal parameters, as shown in Table A.4. In Spectral Clustering, σ serves as the bandwidth parameter of the Gaussian kernel, and we reserve the s largest entries of each row in the affinity matrix. In EnSC, $\tau \in [0, 1]$ is the parameter regulating the sparsity of self-expressive coefficients and β is the trade-off parameter balancing the self-expressive error against the sparsity regularizer. In SCAN, μ is the weight of the between-cluster entropy-maximizing regularization.

The MoCo pre-trained model. To train our CgMCR^2 from scratch, we leverage MoCo-v2, a self-supervised learning method, to learn pre-features. The MoCo-v2 image encoder takes two augmentations of each image as inputs, and we utilize the averaged output embedding of the two augmentations as the pre-feature. The augmentation strategy follows that in NMCE, and is detailed in Table A.5.

The CLIP pre-trained model. CLIP is a large-scale language-supervised learning method that learns general semantic meaning from over 400 million text-image pairs. In our approach, we utilize only the image encoder of the

Table A.5: Augmentation strategy of MoCo-v2.

```
from torchvision.transforms import *
```

```
Compose([
  RandomResizedCrop(32,scale=(0.08, 1.0)),
  RandomHorizontalFlip(p=0.5),
  RandomApply([ColorJitter(0.4, 0.4, 0.4, 0.1)], p=0.8),
  RandomGrayscale(p=0.2),
  ToTensor(),
  Normalize([0.4914, 0.4822, 0.4465], [0.2023, 0.1994, 0.2010])
  )])
```

Table B.6: Effect of output activation function.

Output activation	CIFAR-10		CIFAR-20		CIFAR-100		TinyImageNet	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
Softmax	97.3	92.6	66.8	69.3	75.4	80.8	71.4	81.0
Gumbel-Softmax (Ours)	97.7	94.3	68.8	74.0	78.3	82.5	72.9	81.4

pre-trained CLIP model. Images are resized to 224 along the smaller edge and center-cropped to 224×224 before being inputted to the CLIP image encoder. Subsequently, the features extracted by the CLIP image encoder are used as pre-features for our CgMCR^2.

B More Experiment Results

B.1 Ablation Study

Ablation on the output activation. In our method, we employ the Gumbel-Softmax as the output activation function of the cluster head. In Table B.6, we compare the use of Softmax as the output activation function with the use of Gumbel-Softmax and report their respective best performances on CIFAR-10, -20, -100 and TinyImageNet. As can be seen, the use of Gumbel-Softmax leads to slightly higher clustering accuracy on four standard datasets.

Ablation on the affinity. We examine the effect of $\mathbf{A}$ with various definitions. As described earlier, the affinity matrix in the proposed CgMCR^2 is defined by $\mathbf{A} := \mathcal{P}_s(\mathbf{Z}_\Theta^\top \mathbf{Z}_\Theta)$. Following traditional spectral clustering approaches, we additionally use Gaussian kernel (a.k.a. the Radial Basis Function kernel) to define the affinities, i.e.,

$$a_{i,j} = \exp\left(-\frac{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2}{2\sigma^2}\right), \tag{1}$$

where σ is the bandwidth parameter. In Table B.7, we report the best clustering performance on CIFAR-10, -100, training time per iteration and memory cost of

Table B.7: Varying definitions of A on CIFAR-10 an CIFAR-100.

Definition of **A**	Time (ms/it)	Memory (MB)	CIFAR-10 ACC	CIFAR-10 NMI	CIFAR-100 ACC	CIFAR-100 NMI
Gaussian kernel	23.2	2,309	97.5	93.8	75.8	81.7
Cosine similarity (Ours)	22.9	2,030	97.7	94.3	78.3	82.5

Table B.8: Varying post-processing operators of A on CIFAR-10.

Post-processing of **A**	Time (ms/it)	Memory (MB)	CIFAR-10 ACC	CIFAR-10 NMI	CIFAR-100 ACC	CIFAR-100 NMI
N/A	22.4	1,905	96.7	91.6	70.1	78.6
Doubly stochastic	74.1	5,465	97.2	92.4	75.1	80.8
Reserving top-s entries (ours)	22.9	2,030	97.7	94.3	78.3	81.9

using different affinity matrices. All the experiments are conducted on a single NVIDIA GeForce 3080Ti GPU, and the batch size is set to 512 when recording the training time and memory cost. As can be seen, computing the Gaussian kernel requires a bit higher computational and memory cost, and achieves slightly inferior performance compared to computing cosine similarity.

We proceed by evaluating the effect of different post-processing operator. We notice that the doubly stochastic projection enjoys solid theoretical guarantees [1] and *state-of-the-art* performance as a post-processing method in subspace clustering [2]. Specifically, the doubly stochastic projection projects the affinity matrix onto a doubly stochastic space

$$\mathcal{A} \coloneqq \left\{ \tilde{\mathbf{A}} \in \mathbb{R}^{N \times N} \mid \tilde{\mathbf{A}}\mathbf{1} = \mathbf{1}, \tilde{\mathbf{A}}^\top \mathbf{1} = \mathbf{1} \right\} \tag{2}$$

under the distance of a scaled $\mathbf{A}$:

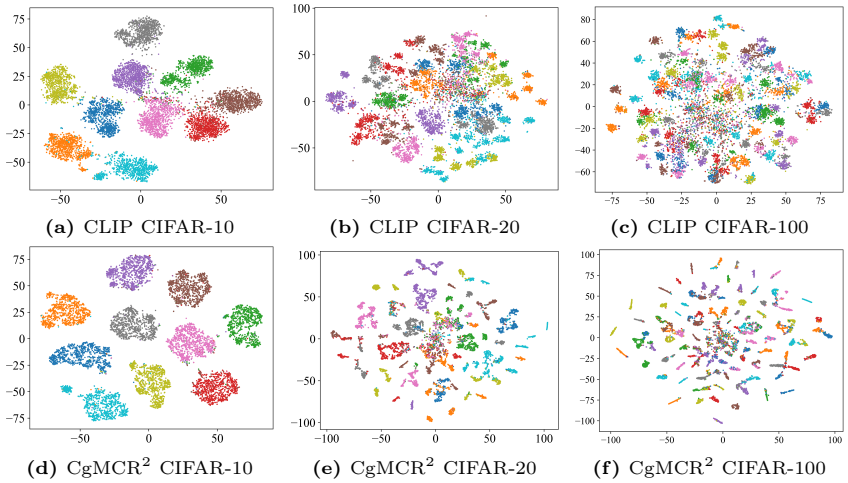
$$\arg\min_{\tilde{\mathbf{A}} \in \mathcal{A}} \left\| \tilde{\mathbf{A}} - \mu \mathbf{A} \right\|_F^2 . \tag{3}$$

This post-processing method also has been adopted in MLC and CPP. In Table B.8, we use cosine similarity to define the affinity matrix and compare our method with doubly stochastic projection and the baseline with no post-processing. In our framework, simply reserving s largest entries of each row in $\mathbf{A}$ achieves the highest accuracy with almost no computational and memory cost, while applying doubly stochastic projection produces less satisfactory clustering results and demands much more training time and GPU memory.

Ablation on parameter s. We previously conducted an ablation study to evaluated the effect of hyper-parameters γ and ϵ. Another important hyper-parameter is s, representing the number of entries reserved in each row of matrix $\mathbf{A}$. In this study, we proceed by evaluate the effect of varying s on CIFAR-10 and CIFAR-100. For CIFAR-10, we fix the batch size to 512 and report the clustering

Table B.9: Clustering accuracy (%) of the CgMCR^2 with varying s on CIFAR-10 and CIFAR-100.

Data \ s	3	5	10	20	50	100	200	300	400	500	1000	1500	2000
CIFAR-10	96.9	97.6	97.7	97.5	97.4	97.4	97.7	97.6	97.4	97.2	-	-	-
CIFAR-100	75.2	76.6	77.9	78.3	76.6	77.3	77.1	77.7	77.2	77.3	77.4	74.2	73.0



(a) CLIP CIFAR-10 **(b)** CLIP CIFAR-20 **(c)** CLIP CIFAR-100

(d) CgMCR^2 CIFAR-10 **(e)** CgMCR^2 CIFAR-20 **(f)** CgMCR^2 CIFAR-100

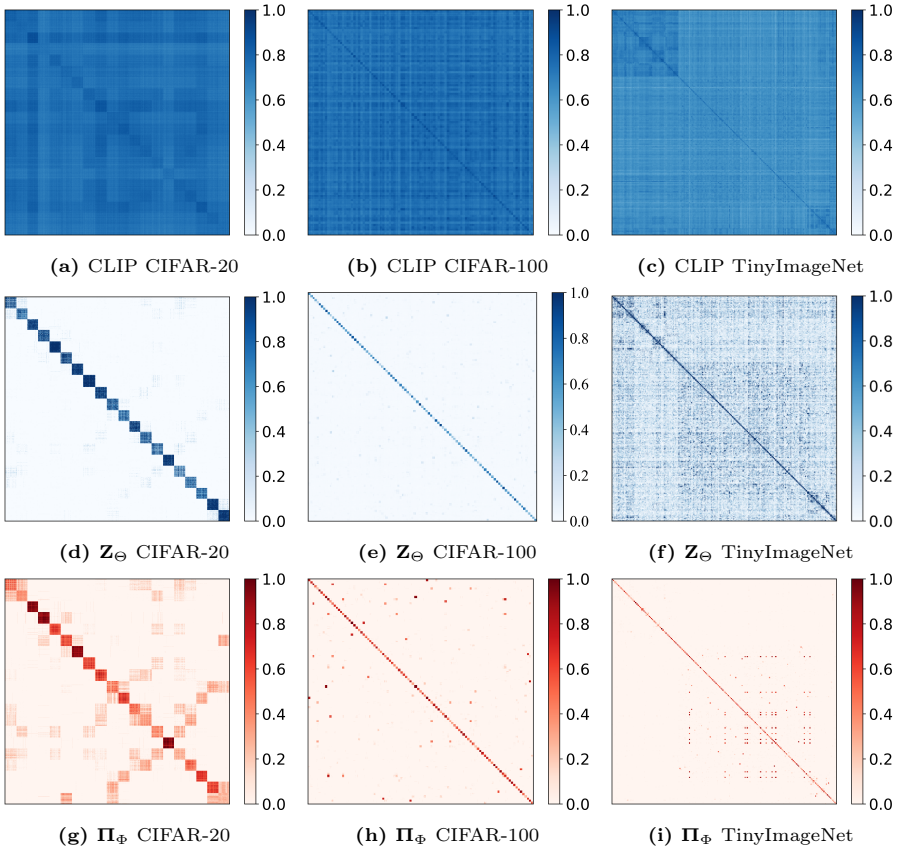
Fig. B.1: Utilizing t-SNE for 2-D visualization. We plot **(a)–(c):** the CLIP pre-features on CIFAR-10, -20, -100 and **(d)–(f):** the CgMCR^2 features on CIFAR-10, -20, -100.

performance of CgMCR^2 with $s \in \{3, 5, 10, 20, 50, 100, 200, 300, 400, 500\}$. For CIFAR-100, we fix the batch size to 2048 and report the clustering performance of CgMCR^2 with $s \in \{3, 5, 10, 20, 50, 100, 200, 300, 400, 500, 1000, 1500, 2000\}$. As can be seen from Table B.9, our method demonstrates robustness to the parameter s. Specifically, values of s within a wide range of $[5, 500]$ yield satisfactory performance on CIFAR-10, while on CIFAR-100, values of s within the range of $[10, 1000]$ yield satisfactory performance.

B.2 Visualization

Visualization via t-SNE. To demonstrate the properties of representations learned by the feature head of CgMCR^2, we also utilize t-SNE [3] to obtain 2-D visualization of the representations on CIFAR-10, CIFAR-20 and CIFAR-100. In Fig. B.1, it is evident that the proposed CgMCR^2 learns a more compact and discriminative representations from the CLIP features.

Ground-truth similarity matrix. The ground-truth similarity matrix is derived by computing by the cosine similarity between data pairs belonging to the



(a) CLIP CIFAR-20 **(b)** CLIP CIFAR-100 **(c)** CLIP TinyImageNet

(d) $\mathbf{Z}_\Theta$ CIFAR-20 **(e)** $\mathbf{Z}_\Theta$ CIFAR-100 **(f)** $\mathbf{Z}_\Theta$ TinyImageNet

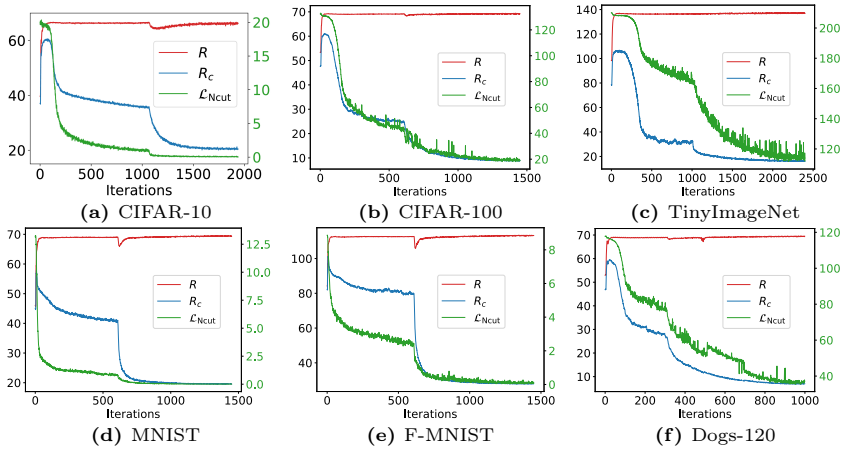
(g) $\mathbf{\Pi}_\Phi$ CIFAR-20 **(h)** $\mathbf{\Pi}_\Phi$ CIFAR-100 **(i)** $\mathbf{\Pi}_\Phi$ TinyImageNet

Fig. B.2: Ground-truth similarity matrices. We plot the similarity matrices of **(a)-(c):** CLIP pre-features, **(d)-(f):** features produced by the CgMCR^2's feature head, and **(g)-(i):** cluster memberships produced by the CgMCR^2's cluster head on CIFAR-20, -100, and TinyImageNet, respectively.

same ground-truth cluster. An optimal ground-truth similarity matrix of representations or memberships should exhibit a block-diagonal structure aligned with the sorted ground-truth labels. In Fig. B.2, we visualize the ground-truth similarity matrices of CLIP pre-features, as well as the features and cluster memberships generated by CgMCR^2 on CIFAR-20, -100 and TinyImageNet. The block-diagonal structures of the ground-truth similarity matrices in our method are clearer than that of the CLIP pre-features.

B.3 Learning Curve

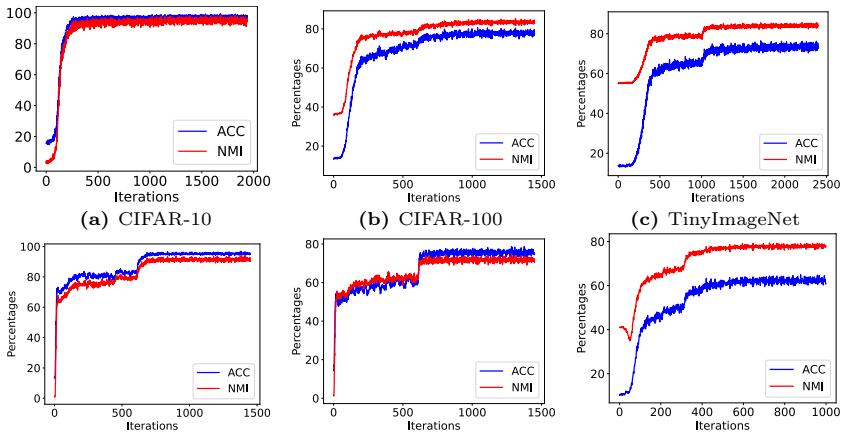
Loss curves. In Fig. B.3, we plot the loss curves of the CgMCR^2 objective during the training on CIFAR-10, -100, TinyImageNet, MNIST, F-MNIST and Dogs-120. As can be seen, the variation of these loss terms are consistent across



(a) CIFAR-10 (b) CIFAR-100 (c) TinyImageNet

(d) MNIST (e) F-MNIST (f) Dogs-120

Fig. B.3: Learning curves of each term in the CgMCR^2 objective on CIFAR-10, -100, TinyImageNet, MNIST, F-MNIST and Dogs-120.



(a) CIFAR-10 (b) CIFAR-100 (c) TinyImageNet

(d) MNIST (e) F-MNIST (f) Dogs-120

Fig. B.4: ACC and NMI curves of $\mathbf{\Pi}_\Phi$ on CIFAR-10, -100, TinyImageNet, MNIST, F-MNIST and Dogs-120.

all datasets. During the one-shot initialization, the term $R(\mathbf{Z}_\Theta;\epsilon)$ initially increases to its maximum to learn discriminative representations, and subsequently the term $\mathcal{L}_{\text{Ncut}}(\mathbf{\Pi}_\Phi;\mathbf{A},\gamma)$ decrease to their *local* minimum as it learns partition information from the discriminative representations. During the fine-tuning, both $\mathcal{L}_{\text{Ncut}}(\mathbf{\Pi}_\Phi;\mathbf{A},\gamma)$ and $R_c(\mathbf{Z}_\Theta,\mathbf{\Pi}_\Phi;\epsilon)$ decrease to their global minimum, while the value of $R(\mathbf{Z}_\Theta;\epsilon)$ remains relatively constant.

ACC and NMI curves. We take the outputs of the cluster head $\mathbf{\Pi}_\Theta$ as the cluster membership and plot its ACC and NMI during each training iteration on CIFAR-20, -100, TinyImageNet, MNIST, F-MNIST and Dogs-120 datasets.

In Fig. B.4, we can observe that our CgMCR^2 converges and achieves the stable clustering results on all tested datasets within 2,500 training iteration, or even fewer.

References

1. Ding, T., Lim, D., Vidal, R., Haeffele, B.D.: Understanding doubly stochastic clustering. In: International Conference on Machine Learning. pp. 5153–5165. PMLR (2022)
2. Lim, D., Vidal, R., Haeffele, B.D.: Doubly stochastic subspace clustering. arXiv preprint arXiv:2011.14859 (2020)
3. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)