Supplementary material:
Reconsidering Representation Alignment for Multi-view Clustering

Daniel J. Trosten Sigurd Løkse Robert Jenssen Michael Kampffmeyer
Department of Physics and Technology, UiT The Arctic University of Norway*

1. Pitfalls of distribution alignment in multi-view clustering

1.1. Proof sketch for Proposition 1

Proof sketch. Suppose that, for view v, the k_v clusters in the input space, are mapped to c_v unique points in the representation space. This is possible under assumptions 1 and 2. The number of unique clusters after fusion is then upper bounded by the number of unique linear combinations on the form:

$$\sum_{v=1}^{V} w_v c_\star^{(v)} \tag{1}$$

where $c_\star^{(v)}$ is one of the c_v points obtained by mapping the k_v unique points (clusters) for view v, to the representation space. Note that the encoders might not be injective, meaning that we can have $c_v < k_v$. Under assumption 3, the maximum number of unique such linear combinations is equal to $c_1 \cdot c_2 \cdots c_V = \prod_{v=1}^{V} c_v$. Since we only have k clusters in the entire dataset, the number of unique clusters after fusion will also be upper bounded by k. This gives:

$$\kappa_\cdot^{\text{fused}} = \min\left\{k, \prod_{v=1}^{V} c_v\right\}. \tag{2}$$

Perfectly aligned representations. Clusters that are separated in the input space can be mapped to the same centroid in the representation space, but not vice versa. I.e. it is not possible for the encoding network to separate two clusters that lie at the same point in the input space. The perfect alignment constraint therefore forces the number of unique points to be equal to the smallest k_v for each cluster. That is:

$$c_v = \min_{w=1,\dots,V}\{k_w\}, \quad v = 1, \dots, V. \tag{3}$$

We then get

$$\kappa_{\text{aligned}}^{\text{fused}} = \min\left\{k, \prod_{v=1}^{V} \min_{w=1,\dots,V}\{k_w\}\right\} \tag{4}$$

$$= \min\left\{k, \left(\min_{v=1,\dots,V}\{k_v\}\right)^V\right\} \tag{5}$$

Unaligned representations. Here the encoder for view v has the ability to map the k_v separable clusters to k_v unique representations, which do not coincide with the representations from any other views. We therefore get $c_v = k_v$, and

$$\kappa_{\text{not aligned}}^{\text{fused}} = \min\left\{k, \prod_{v=1}^{V} c_v\right\} \tag{6}$$

$$= \min\left\{k, \prod_{v=1}^{V} k_v\right\}. \tag{7}$$

□

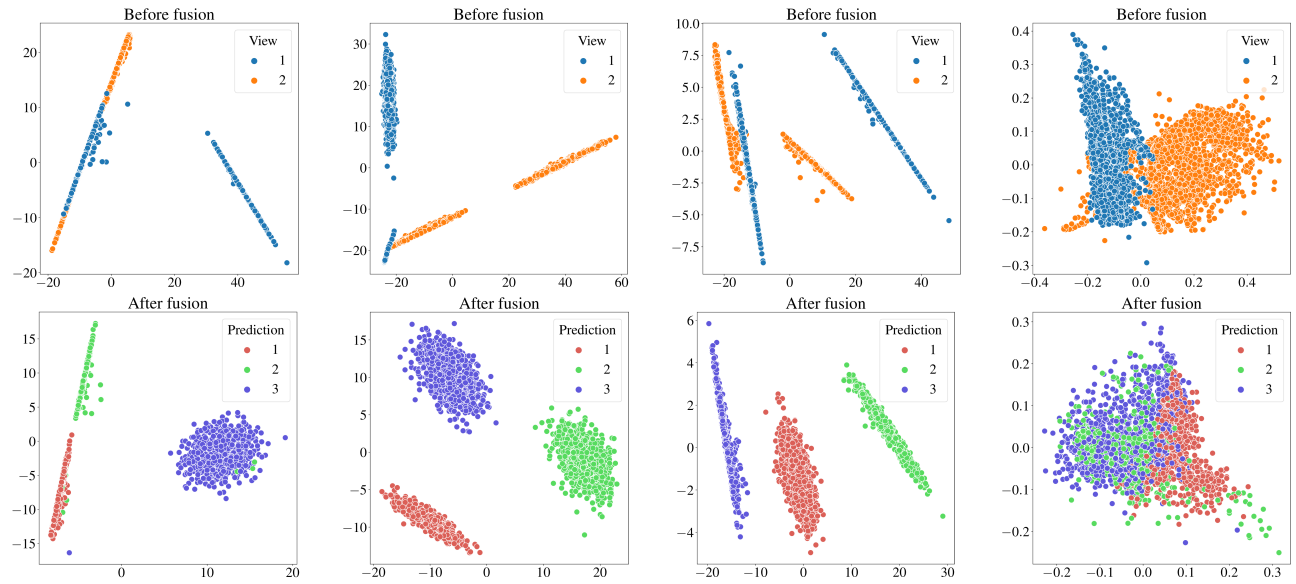
1.2. Experiments with toy data

Figure 1 shows the results of our toy experiment with 3 clusters instead of 5. The dataset is shown in Figure 2. Similarly to the experiment with 5 clusters, we observe that SiMVC + Adv. partially aligns the distributions. Due to the reduced number of clusters however, it is still possible to separate the clusters after fusion. This outcome is consistent with Proposition 1, from which we get $\kappa_{\text{aligned}}^{\text{fused}} = \min\{3, 2^2\} = 3$.

For CoMVC, we see that the angles between representations have been aligned, which also results in separable clusters.

EAMC attempts to align the distributions, which results in all clusters being mixed together after fusion – similar to what we observed for the experiment with 5 clusters. For this experiment, we also observe that EAMC produces approximately equal fusion weights. This violates assumption 3 in Proposition 1, and can further reduce the cluster separability in the space of fused representations.

*UiT Machine Learning Group, machine-learning.uit.no



(a) SiMVC +Adv. ACC = 0.99 (b) SiMVC. ACC = 1.0. (c) CoMVC. ACC = 1.0. (d) EAMC. ACC = 0.44 .

Figure 1: Representations for SiMVC with and without adversarial alignment, CoMVC, and EAMC on a version of our toy dataset with 3 clusters.

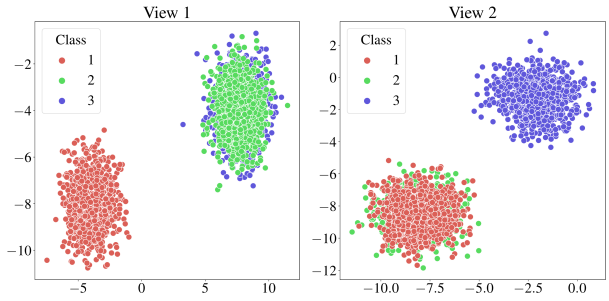


Figure 2: Toy dataset with 3 clusters. View 1: Class 1 is isolated, and classes (2,3) overlap. View 2: Class (1,2) overlap, and class 3 is isolated.

2. Methods

2.1. CoMVC with projection head

Table 1 shows the results of an extension of the ablation study for CoMVC, where we also include a projection head between the view representations and the cosine similarity. Following [1], we let the projection head be two fully connected layers, separated by a ReLU-nonlinearity. Batch normalization is applied after both layers. The results show that some configurations benefit marginally from the addition of a projection head. However, adding the projection head does not improve the overall performance of CoMVC. We therefore chose to not include it in the final model.

Dataset	Projection head	Negative sampling	Adaptive weight	ACC [%]	NMI [%]
E-MNIST	–	–	–	87.4	86.8
	–	✓	–	87.5	86.6
	–	–	✓	94.7	89.5
	–	✓	✓	**95.5**	**90.7**
	✓	–	–	87.5	86.9
	✓	✓	–	88.2	86.3
	✓	–	✓	87.4	87.2
	✓	✓	✓	77.1	77.5
VOC	–	–	–	54.7	61.3
	–	✓	–	58.5	67.4
	–	–	✓	55.3	60.7
	–	✓	✓	61.9	**67.5**
	✓	–	–	53.4	58.2
	✓	✓	–	57.0	63.2
	✓	–	✓	**62.4**	65.3
	✓	✓	✓	55.2	59.6

Table 1: CoMVC ablation study with and without a projection head.

2.2. Ablation study: clustering loss

Here, we perform an ablation study on the E-MNIST dataset, in order to show the effects of the individual terms in the DDC [2] clustering loss. Note that, since not all config-

Model	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	ACC [%]	NMI [%]
SiMVC	✓	–	–	19.2	19.6
	–	✓	–	38.1	31.4
	–	–	✓	75.2	73.9
	✓	✓	–	78.2	78.6
	✓	–	✓	76.6	77.5
	–	✓	✓	77.4	76.9
	✓	✓	✓	**86.2**	**82.6**
CoMVC	✓	–	–	19.3	20.6
	–	✓	–	36.5	25.2
	–	–	✓	72.8	71.7
	✓	✓	–	71.3	73.2
	✓	–	✓	78.0	78.2
	–	✓	✓	74.8	73.5
	✓	✓	✓	**95.5**	**90.7**

Table 2: Results of an ablation study where we systematically drop terms from the clustering loss. The checkmarks indicate which terms that are included in each configuration.

urations include the $\mathcal{L}_1$ term, we select models based on the sum of the included terms instead. The resulting accuracies for SiMVC and CoMVC when we systematically drop terms from the clustering loss, are listed in Table 2. These results are in line with previous ablation studies conducted on the DDC clustering loss [2, 5]: The models perform best when all terms are included – dropping terms from the clustering loss reduces the performance of both SiMVC and CoMVC.

3. Experiments

3.1. Source code

The source code for our experiments is publicly available at https://github.com/DanielTrosten/mvc.

3.2. Details of pre-trained models

For RGB-D, we use the following pre-trained models to extract features for the respective views:

- View 1: ResNet-50 pre-trained on the ImageNet dataset. We use the version available in PyTorch[1], and remove the last (classification) layer.
- View 2: Doc2Vec pre-trained on the Wikipedia dataset. We use the pre-trained model available at https://github.com/jhlau/doc2vec.

Note that the same types of architectures are used in [5] to extract features for RGB-D. However, the authors do not supply the model and training details required to exactly reproduce their features.

[1]Documentation for the model can be found at https://pytorch.org/docs/stable/torchvision/models.html

Layer type	Neurons	Activation	Batch-norm
FC	512	ReLU	×
FC	512	ReLU	×
FC	256	ReLU	×

(a) Fully connected encoder. FC: fully connected layer.

Layer type	Filter size	Filters	Activation	Batch-norm
Conv	5×5	32	ReLU	×
Conv	5×5	32	ReLU	✓
MaxPool	2×2	–	–	×
Conv	3×3	32	ReLU	×
Conv	3×3	32	ReLU	✓
MaxPool	2×2	–	–	×

(b) Convolutional neural network encoder. Conv: convolutional layer. MaxPool: max-pooling layer. Note that Batch normalization is applied before the activation function.

Layer type	Neurons	Activation	Batch-norm
FC	100	ReLU	✓
FC	k	softmax	×

(c) Clustering module. k denotes the number of clusters.

Table 3: Network architectures.

3.3. Model architectures and hyperparameters

SiMVC and CoMVC trained on VOC, CCV, and RGB-D use fully connected encoders (Table 3a) for all views. On E-MNIST, E-FMNIST and COIL our models use convolutional neural network encoders for all views (Table 3b). The clustering module (Table 3c) is the same for all experiments with SiMVC and CoMVC.

Table 4 lists the other hyperparameters that are not part of the model architectures. We use gradient clipping, and clip gradients with norms greater than "Max gradient norm". For some datasets, we found that decaying the learning rate helped the models converge. On these datasets, we reduce the learning rate once, at epoch "Decay step", with a factor of "Decay factor".

3.4. Evaluation protocol

For VOC, CCV, and E-MNIST, we use the baseline results obtained by [5]. For all baseline models, except EAMC, they run the model 10 times and report the average ACC and NMI. For EAMC, they train the model 20 times, and report the results from the run which resulted in the lowest value of the loss function. We follow the same evaluation procedure for EAMC, when we evaluate it on E-FMNIST, COIL-20,

Dataset	Model	Batch size	Epochs	τ	δ	Negative samples	Max gradient norm	Initial learning rate	Decay step	Decay factor
VOC	SiMVC	100	100	0.1	0.1	–	5	0.001	50	0.1
	CoMVC	100	100	0.1	0.1	25	5	0.001	–	–
CCV	SiMVC	100	100	0.1	20	–	5	0.001	–	–
	CoMVC	100	100	0.1	20	25	5	0.001	50	0.1
E-MNIST	SiMVC	100	100	0.1	0.1	–	5	0.001	–	–
	CoMVC	100	100	0.1	0.1	25	5	0.001	–	–
E-FMNIST	SiMVC	100	100	0.1	0.1	–	5	0.001	–	–
	CoMVC	100	100	0.1	0.1	25	5	0.001	–	–
COIL-20	SiMVC	100	100	0.1	20	–	5	0.001	–	–
	CoMVC	100	100	0.1	20	25	5	0.001	–	–
RGB-D	SiMVC	100	100	0.1	0.1	–	5	0.001	–	–
	CoMVC	100	100	0.1	0.1	25	5	0.001	50	0.5

Table 4: Hyperparameters used to train SiMVC and CoMVC.

	SwMPC [4]	RSwMPC [3]	SiMVC	CoMVC
ACC	0.1679	0.2778	0.2717	**0.2892**
NMI	0.0899	0.1810	0.1767	**0.2052**

Table 5: Comparison with [4, 3] on NUS-WIDE-Animal [3].

	EAMC	SiMVC	CoMVC
sec/epoch	33.48	**12.18**	14.28

Table 6: Time spent per training epoch for EAMC, SiMVC and CoMVC on E-FMNIST.

and RGB-D.

3.5. Experiments on NUS-WIDE-Animal

Table 5 shows the performance of our models on NUS-WIDE-Animal [3] (a subset of NUS-WIDE containing the animal classes only) compared to two additional models [4, 3] (as reported in [3]). SiMVC performs comparable, while CoMVC outperforms the competitors.

3.6. Training times

Table 6 shows the average time spent per training epoch for EAMC, SiMVC, and CoMVC. Both SiMVC and CoMVC are more than twice as fast to train per epoch, when compared to EAMC. We believe that this is due to the extra components (attention network, discriminator) included in EAMC. SiMVC is a bit faster to train than CoMVC, due to the extra computations introduced by the contrastive loss.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*, 2020.

[2] Michael Kampffmeyer, Sigurd Løkse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Børre Salberg, and Robert Jenssen. Deep divergence-based approach to clustering. *Neural Networks*, 113, 2019.

[3] Beilei Wang, Yun Xiao, Zhihui Li, Xuanhong Wang, Xiaojiang Chen, and Dingyi Fang. Robust Self-Weighted Multi-View Projection Clustering. In *AAAI Conference on Artificial Intelligence*, 2020.

[4] R. Wang, F. Nie, Z. Wang, H. Hu, and X. Li. Parameter-Free Weighted Multi-View Projected Clustering with Structured Graph Learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(10), 2020.

[5] Runwu Zhou and Yi-Dong Shen. End-to-End Adversarial-Attention Network for Multi-Modal Clustering. In *Computer Vision and Pattern Recognition*, 2020.