ON THE ROLE OF SELF-SUPERVISION IN DEEP MULTI-VIEW CLUSTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-supervised learning is a central component in many recent approaches to deep multi-view clustering (MVC). However, we find large variations in the motivation and design of self-supervision-based methods for deep MVC. To address this, we present DeepMVC, a new, unified framework for deep MVC. Crucially, we show that many recent methods can be regarded as instances of our framework – allowing us to implement recent methods in a unified and consistent manner. We make key observations about the effect of self-supervision, and in particular, drawbacks of representation alignment. Motivated by these insights, we develop several new DeepMVC instances, with new forms of self-supervision. We conduct extensive experiments, and find that (i) the popular contrastive alignment degrades performance when the number of views becomes large; (ii) all methods benefit from some form of self-supervision; and (iii) our new instances outperform previous methods on several datasets. Based on our findings, we suggest several promising directions for future research. To enhance the openness of the field, we provide an open-source implementation of DeepMVC, including recent models and our new instances. Our implementation includes a consistent evaluation protocol, facilitating fair and accurate evaluation of methods and components.

1 INTRODUCTION

Multi-view clustering (MVC) generalizes the standard clustering task to data where the instances to be clustered are observed through multiple views, or by multiple modalities. In recent years, deep learning architectures have seen widespread adoption in MVC, resulting in the *deep MVC* subfield. Methods developed within this subfield have shown state-of-the-art clustering performance on several multi-view datasets (Zhou and Shen, 2020; Trosten et al., 2021; Xu et al., 2021a; Mao et al., 2021; Wang et al., 2022a;b), largely outperforming traditional, non-deep-learning-based methods (Zhou and Shen, 2020).

Despite these promising developments, we identify significant drawbacks with the current state of the field. Self-supervised learning (SSL) is a crucial component in many recent methods for deep MVC (Zhou and Shen, 2020; Trosten et al., 2021; Xu et al., 2021a; Mao et al., 2021; Wang et al., 2022a;b). However, the large number of methods, all with unique components and arguments about how they work, makes it challenging to identify clear directions and trends in the development of new components and methods. Methodological research in deep MVC thus lacks foundation and consistent directions for future advancements. This effect is amplified by the large variations in the implementation and evaluation of new methods. Network architectures, data preprocessing and data splits, hyperparameter search strategies, evaluation metrics, and model selection strategies all vary greatly across publications, making it difficult to properly compare methods from different papers.

To address these challenges, we present a unified framework for deep MVC, coupled with a rigorous and consistent evaluation protocol, and an open-source implementation. Our 4 main contributions are summarized as follows:

(1) DeepMVC framework. Despite the variations in the development of new methods, we recognize that the majority of recent methods for deep MVC can be decomposed into the following fixed set of components: (i) view-specific encoders; (ii) single-view SSL; (iii) multi-view SSL; (iv) fusion; and (v) clustering module. The DeepMVC framework (Figure 1) is obtained by organizing these components into a unified deep MVC model. Methods from previous work can thus be regarded as *instances* of DeepMVC.

(2) New instances of DeepMVC. Inspired by initial findings from the DeepMVC framework we develop 6 new instances of DeepMVC, which outperform current state-of-the-art methods on several multi-view datasets. The new instances include both novel and well-known types of self-supervision, fusion and clustering modules.

(3) Open-source implementation of DeepMVC and evaluation protocol. We provide an open-source[1] implementation of DeepMVC, including several recent methods, and our new instances. The implementation includes a shared evaluation protocol for all methods, and all datasets used in the experimental evaluation. By making the datasets and all parts of our implementation openly available, we aim to facilitate simpler development of new methods, as well as rigorous and accurate comparisons between methods and components.

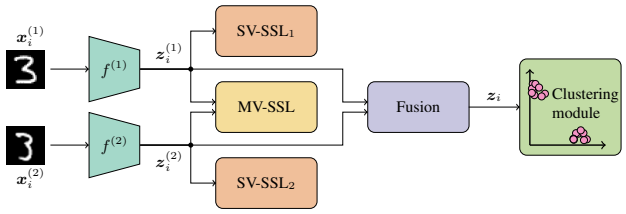

Figure 1: Overview of the DeepMVC framework for a two-view dataset. Different colors denote different components. The framework is generalizable to an arbitrary number of views by adding more view specific encoders (f) and SV-SSL blocks.

(4) Evaluation of methods and components. We use the implementation of DeepMVC to evaluate and compare several recent state-of-the-art methods and SSL components – both against each other, and against our new instances. In our experiments, we both provide a consistent evaluation of methods in deep MVC, and systematically analyze several SSL-based components – revealing how they behave under different experimental settings.

The main findings from our work are:

- We discover a previously unknown drawback of contrastive alignment. Contrastive alignment of view-specific representations works well for datasets with few views, but *significantly degrades performance when the number of views increases*. Conversely, we find that maximization of mutual information performs well on datasets with many views, while not being as strong on datasets with fewer views.
- All methods included in our experiments benefit from at least one form of SSL. In addition to contrastive alignment for few views and mutual information maximization for many views, we find that autoencoder-style reconstruction improves overall performance of methods.
- Properties of the datasets, such as class (im)balance and the number of views, heavily impact the performance of current MVC approaches. There is thus not a single "state-of-the-art" – it instead depends on the datasets considered.
- Results reported by the original authors differ significantly from the performance of our re-implementation for some baseline methods, illustrating the necessity of a unified framework with a consistent evaluation protocol.

2 DEEPMVC FRAMEWORK

In this section we present the DeepMVC framework, its components and their purpose, and how they fit together. This allows us to, in the next section, summarize recent work on deep MVC, and illustrate that the majority of recent methods can be regarded as instances of our unified DeepMVC framework.

Suppose we have a multi-view dataset consisting of n instances and V views, and let $\boldsymbol{x}_i^{(v)}$ be the observation of instance i through view v. The task of the DeepMVC framework is then to cluster the instances into k clusters, and produce cluster membership indicators $\alpha_{ic} \in [0, 1]$, $c = 1, \ldots, k$. The framework is illustrated in Figure 1. It consists of the following components.

View-specific encoders. The framework is equipped with V deep neural network encoders $f^{(1)}, \ldots, f^{(V)}$, one for each view. Their task is to produce the view-specific representations $\boldsymbol{z}_i^{(v)} = f^{(v)}(\boldsymbol{x}_i^{(v)})$ from the input data.

Single-view self-supervised learning (SV-SSL). The SV-SSL component consists of a set of pretext tasks (auxiliary objectives) that are designed to aid the optimization of the view-specific encoders. Specifically, the tasks should be designed to help the encoders learn representations that simplify

[1]The implementation is available in the supplementary, and will be made publicly available upon publication.

the clustering task. Each pretext task is specific to its designated view, and is isolated from all other views.

Multi-view self-supervised learning (MV-SSL). MV-SSL is similar to SV-SSL – they are both self-supervised modules whose goals are to help the encoders learn representations that are suitable for clustering. However, MV-SSL leverages all views simultaneously in the pretext tasks, allowing the model to exploit information from all views simultaneously to learn better features.

Fusion. The fusion component combines view-specific representations into a shared representation for all views. Fusion is typically done using a (weighted) average (Li et al., 2019; Trosten et al., 2021), or by concatenation (Huang et al., 2019a; Xin et al., 2021; Xu et al., 2021a). However, more complex fusion modules using *e.g.* attention mechanisms (Zhou and Shen, 2020), are also possible.

Clustering module (CM). The CM is responsible for determining the cluster memberships based on either the view-specific or fused representations. The CM can consist of a traditional clustering method, such as k-means (MacQueen, 1967) or Spectral Clustering (Shi and Malik, 2000). These CMs are applied to the fused representations after the other components have been trained, resulting in a two-stage method that first learns fused representations, and then applies a clustering algorithm to these representations.

Alternatively, the CM can be integrated into the model (Zhou and Shen, 2020; Trosten et al., 2021), allowing it to be trained alongside the other components, possibly resulting in fused representations that are better suited for clustering.

Loss functions and training. The loss functions for the models are specified by the SV-SSL, MV-SSL, and CM components. To train the model, the terms arising from the different components can be minimized simultaneously or they can be minimized in an alternating fashion. It is also possible with pre-training/fine-tuning setups where the model is pre-trained with one subset of the losses and fine-tuned with another subset of the losses.

We note that DeepMVC is a conceptual framework, and that a model is not necessarily completely described by a list of its DeepMVC components. Consequently, it is possible for two models with similar DeepMVC components to have slightly different implementations. This illustrates the importance of our open-source implementation of DeepMVC, which allows the implementation of a model to be completely transparent.

3 Previous methods as instances of DeepMVC

Table 1 shows selected recent methods for deep MVC (the full table can be found in the supplementary), categorized by its DeepMVC components, allowing for systematic comparisons between models.

View-specific encoders. As can be seen in Table 1, all models use view-specific encoders to encode views into view-specific embeddings. Multi-layer perceptrons (MLPs) are usually used for vector data, while convolutional neural networks (CNNs) are used for image data.

SV-SSL and MV-SSL. Alongside the encoder network, many methods use decoders to reconstruct the original views from either the view-specific representations or the fused representation. The reconstruction task is the most common self-supervised pretext task, both for SV-SSL and for MV-SSL. In SV-SSL, the views are reconstructed from their respective view-specific representations, without any influence from the other views (Wang et al., 2015; Abavisani and Patel, 2018; Tang et al., 2018; Sun et al., 2019; Zhang et al., 2020a;b; Zong et al., 2020; Xu et al., 2021b). In MV-SSL, it is common to either do (i) cross view reconstruction, where all views are reconstructed from all view-specific representations (Zhu et al., 2019); or (ii) fused view reconstruction, where all views are reconstructed from the fused representation (Zhu et al., 2019; Li et al., 2019; Yin et al., 2020; Wang et al., 2022a).

Aligning distributions of view-specific representations is another MV-SSL pretext task that has been shown to produce representations suitable for clustering (Zhou and Shen, 2020). However, Trosten et al. (2021) demonstrate that the alignment of representation distributions can be detrimental to the clustering performance – especially in the presence of noisy or non-informative views. To avoid these drawbacks, they propose Simple MVC (SiMVC) and Contrastive MVC (CoMVC). In the former, the alignment is dropped altogether, whereas the latter includes a contrastive learning module that aligns the view-specific representations at the instance level, rather than at the distribution level.

Clustering modules. Many deep MVC methods use subspace-based clustering modules (Abavisani and Patel, 2018; Zhu et al., 2019; Sun et al., 2019; Wang et al., 2022b). These methods assume

Table 1: Overview of selected methods from previous work (top) and proposed new instances (bottom), and their DeepMVC components. The complete table of previous methods is included in Supplementary Section B.

Model	Ref	Pub.	Enc.	SV-SSL	MV-SSL	Fusion	CM
DCCAE	Wang et al. (2015)	ICML	MLP	Reconstruction	CCA	1st view	SC
DMSC	Abavisani and Patel (2018)	J. STSP	CNN	Reconstruction	–	Affinity fusion	SR, SC
MvSCN	Huang et al. (2019a)	IJCAI	MLP	Sp. Emb.	MSE Al.	Concat.	k-means
DAMC	Li et al. (2019)	IJCAI	MLP	–	Reconstruction	Average	DEC
SGLR-MVC	Yin et al. (2020)	AAAI	MLP	Variational Reconstruction	Variational Reconstruction	Weighted sum	GMM
EAMC	Zhou and Shen (2020)	CVPR	MLP	–	Distribution Al., Kernel Al.	Attention	DDC
SiMVC	Trosten et al. (2021)	CVPR	MLP/ CNN	–	–	Weighted sum	DDC
CoMVC	Trosten et al. (2021)	CVPR	MLP/ CNN	–	Contrastive Al.	Weighted sum	DDC
Multi-VAE	Xu et al. (2021a)	ICCV	CNN	–	Variational Reconstruction	Concat.	Gumbel, k-means
DMIM	Mao et al. (2021)	IJCAI	MLP	Min. superflous information	Max. shared information	?	Encoder output
Model	**Category**	**Pub.**	**Enc.**	**SV-SSL**	**MV-SSL**	**Fusion**	**CM**
AE–KM	Simple	Ours	MLP/ CNN	Reconstruction	–	Concat.	k-means
AE–DDC	Simple	Ours	MLP/ CNN	Reconstruction	–	Weighted sum	DDC
AECoKM	Contrastive Al.	Ours	MLP/ CNN	Reconstruction	Contrastive Al.	Concat.	k-means
AECoDDC	Contrastive Al.	Ours	MLP/ CNN	Reconstruction	Contrastive Al.	Weighted sum	DDC
InfoDDC	Mutual information	Ours	MLP/ CNN	–	Max. mutual information	Weighted sum	DDC
MV-IIC	Mutual information	Ours	MLP/ CNN	–	IIC Over-clustering	–	IIC, k-means

Abbreviations: "–" = Not included, "?" = Not specified, Al. = Alignment, Concat. = Concatenate, CCA = Canonical correlation analysis, DDC = Deep divergence-based clustering, DEC = Deep embedded clustering, SC = Spectral clustering, Sp. Emb. = Spectral Embedding, SR = Self-representation,

that representations, either view-specific or fused, can be decomposed into linear combinations of each other. Once determined, the self-representation matrix containing the coefficients for these linear combinations is used to compute an affinity matrix, which in turn is used as input to spectral clustering. This requires the full $n \times n$ self-representation matrix available in memory, which is computationally prohibitive for datasets with a large number of instances.

Other clustering modules have also been adapted to deep MVC. The clustering module from Deep Embedded Clustering (DEC) (Xie et al., 2016), for instance, is used in several models (Li et al., 2019; Du et al., 2021; Xin et al., 2021; Xu et al., 2021b; Wang et al., 2022a). Recently, the Deep Divergence-Based Clustering (DDC) (Kampffmeyer et al., 2019) clustering module has been used in several state-of-the-art deep MVC models (Zhou and Shen, 2020; Trosten et al., 2021). In addition, some methods treat either the encoder output or the fused representation as cluster membership vectors (Zhang et al., 2020b; Mao et al., 2021).

Lastly, some methods adopt a two-stage approach, where they first use the SSL components to learn representations, and then apply a traditional clustering method, such as k-means (Huang et al., 2019b;a; Zhang et al., 2020a; Zong et al., 2020; Xu et al., 2021a), a Gaussian mixture model (Yin et al., 2020), or spectral clustering (Wang et al., 2015), on the trained representations.

4 SSL AND CONTRASTIVE ALIGNMENT – DRAWBACKS AND ADVANTAGES

The DeepMVC framework allows us to accurately examine the effect of different components across models and datasets. In this section we briefly present initial experimental results with previous methods – obtained with our open-source implementation of DeepMVC – related to self-supervision and contrastive alignment. This will motivate the new instances presented in Section 5, and guide the full experimental evaluation presented in Section 6.

Table 2: Clustering accuracies from ablation study with SSL components.

Model	NoisyMNIST w/o SSL	NoisyMNIST w/ SSL	Caltech7 w/o SSL	Caltech7 w/ SSL
DMSC	0.54	0.66 (+0.12)	0.35	0.50 (+0.15)
EAMC	1.00	0.83 (−0.17)	0.36	0.44 (+0.08)
Multi-VAE	0.52	0.98 (+0.46)	0.31	0.47 (+0.15)
CoMVC	1.00	1.00 (0.00)	0.41	0.38 (−0.02)

Table 3: Clustering accuracies on datasets with varying number of views. Standard deviations are shown in parentheses.

	EdgeMNIST (2 views)	Caltech7 (6 views)	PatchedMNIST (12 views)
SiMVC	0.89 (0.06)	**0.41** (0.02)	**0.84** (0.04)
CoMVC	**0.97** (0.08)	0.38 (0.01)	0.73 (0.12)

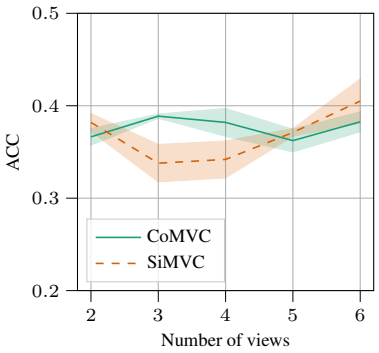

Figure 2: Clustering accuracy when the number of views in Caltech7 is increased from 2 to 6.

Advantage of self-supervision. Table 2 shows the results of an ablation study with the SV-SSL and MV-SSL components, conducted with 4 models from previous work. We observe that self-supervision is beneficial for the performance of the model in most cases. The additional supervisory signal from the SSL components prevents representation collapse, and helps the model learn clustered representations. Interestingly, EAMC on NoisyMNIST performs best without any form of self-supervision. This is likely because the distribution alignment in EAMC attempts to align representations from the noisy view to those from the noise-free view, resulting in representations that are less separable. We discuss this phenomenon further in the following paragraph.

Performance of contrastive alignment-based models. We experiment with SiMVC and CoMVC (Trosten et al., 2021) – where the former is a simple baseline model, and the latter is the same model with an additional contrastive learning module. The simplicity of these models allows us to accurately assess the effect of the contrastive learning component. The results in Table 3 show that contrastive learning (CoMVC) outperforms the baseline (SiMVC) when the number of views is low, but when the number of views increases, CoMVC is outperformed by the non-contrastive-learning-based SiMVC. The same trend can be seen in Figure 2, where the models are trained on Caltech7 with an increasing number of views. CoMVC performs best with only 3 views, whereas SiMVC performs best when all views are included. We argue that this is related to an issue with contrastive alignment, where a dataset with many views is more likely to have less informative views, where some clusters are inseparable from each other. Aligning representations will then cause view-specific representations to be aligned to the least informative view(s), resulting in clusters that are harder to separate in the representation space. For completeness, we provide a more thorough explanation of this phenomenon in Supplementary Section C.

5 New instances of DeepMVC

The above findings, as well as the overview in Table 1, indicate that SV-SSL and MV-SSL are central components in recent methods for deep MVC. With our new instances of DeepMVC, we aim to further investigate the effect of different SSL components, as well as to address the many-views-issue with contrastive alignment highlighted above. In addition to alignment of view-specific representations (Zhou and Shen, 2020; Du et al., 2021; Trosten et al., 2021), we identify reconstruction (Wang et al., 2015; Abavisani and Patel, 2018; Li et al., 2019) and mutual information maximization (Ji et al., 2019; Wang et al., 2022b) to be promising directions for the new instances. Furthermore, we recognize that simple baselines with few or no SSL components – exemplified by SiMVC (Trosten et al., 2021) – might perform similarly to more complicated methods, while being significantly easier to implement and faster to train. It is therefore crucial to include such methods in an experimental evaluation, in order to properly determine whether additional SSL-based components are beneficial for the models' performance. Finally, our overview of recent work shows that both traditional clustering modules (*e.g.* k-means) and deep learning-based clustering modules (*e.g.* DDC) are commonly used in deep MVC.

In total, we develop 6 new DeepMVC instances in 3 categories. Evaluating these instances and several methods from recent work, allows us to accurately evaluate methods and components, and investigate how they behave for datasets with varying characteristics.

Simple baselines: **AE–KM** has view-specific autoencoders (AEs) with a mean-squared-error (MSE) loss as its SV-SSL task. The views are fused by concatenation and the concatenated representations are clustered using k-means after the view-specific autoencoders have been trained. **AE–DDC** uses view-specific autoencoders with an MSE loss as its SV-SSL task. The views are fused using a weighted sum and the fused representations are clustered using the DDC clustering module (Kampffmeyer et al., 2019).

Contrastive alignment-based: **AECoKM** extends AE–KM with a contrastive loss on the view-specific representations. We use the multi-view generalization of the NT-Xent (contrastive) loss by Trosten et al. (2021), without the "other clusters" negative sampling. **AECoDDC** extends AE–DDC using the same generalized NT-Xent contrastive loss on the view-specific representations.

Mutual information-based: **InfoDDC** maximizes the mutual information (MI) between the view-specific representations, using the MI loss from Invariant Information Clustering (IIC) (Ji et al., 2019). The MI maximization is regularized by also maximizing the entropy of view-specific representations. The view-specific representations are fused using a weighted sum, and the fused representations are clustered using DDC (Kampffmeyer et al., 2019). **MV-IIC** is a multi-view generalization of IIC (Ji et al., 2019), where cluster assignments are computed for each of the view-specific representations. The MI between pairs of these view-specific cluster assignments is then maximized using the information maximization loss from IIC. In order to get a final shared cluster assignment for all views, the view-specific cluster assignments are concatenated and clustered using k-means. As in IIC, this model includes 5 over-clustering heads as its MV-SSL task.

The new instances are also summarized in Table 1. Further details, including the exact loss functions, are provided in Supplementary Section D.

6 Experiments

In this section we provide a rigorous evaluation of methods and their DeepMVC components. Inspired by the initial findings in Section 4, we focus mainly on the SSL and CM components in our evaluation. We found these components to be most influential on the methods' performance. For completeness, we also include experiments with different fusion types in Supplementary Section E.

6.1 Setup

Baselines. In addition to the new instances presented in Section 5, we include 6 baseline models from previous work in our experiments. The following baseline models were selected to include a diverse set of framework components in the evaluation: (i) Deep Multimodal Subspace Clustering (DMSC) (Abavisani and Patel, 2018); (ii) Multi-view Spectral Clustering Network (MvSCN) (Huang et al., 2019a); (iii) End-to-end Adversarial-attention Multimodal Clustering (EAMC) (Zhou and Shen, 2020); (iv) Simple Multi-View Clustering (SiMVC) (Trosten et al., 2021); (v) Contrastive Multi-View Clustering (CoMVC) (Trosten et al., 2021); (vi) Multi-view Variational Autoencoder (Multi-VAE) (Xu et al., 2021a).

As can be seen in Table 1, this collection of models includes both reconstruction-based and alignment-based SSL, as well as traditional (k-means and spectral) and deep learning-based clustering modules. They also include several fusion strategies and encoder networks. Section 6.3 includes an ablation study that examines the influence of SSL components in these models.

Datasets. We evaluate the baselines and new instances on 8 different multi-view datasets, prioritizing datasets that were also used in the original publications for the selected baselines. Not only does this result in a diverse collection of datasets common in deep MVC – it also allows us to compare the performance of our implementations to what was reported by the original authors. The results of this comparison are given in Supplementary Section E.

The following datasets are used for evaluation: (i) **NoisyMNIST / NoisyFashion**: A version of MNIST (Lecun et al., 1998) / FashionMNIST (Xiao et al., 2017) where the first view contains the original image, and the second view contains an image sampled from the same class as the first image, with added Gaussian noise ($\sigma = 0.2$). (ii) **EdgeMNIST / EdgeFashion**: Another version of MNIST / FashionMNIST where the first view contains the original image, and the second view contains an edge-detected version of the same image. (iii) **COIL-20**: The original COIL-20 (Nene et al., 1996) dataset, where we randomly group the images of each object into groups of size 3,

Table 4: Aggregated evaluation results for the dataset groups. Models are sorted from lowest to highest by average Z-score for each group. Higher Z-scores indicate better clusterings. Our new instances are underlined.

(a) **All datasets**			(b) **Random pairings**			(c) **Many views**				(d) **Balanced vs. imbalanced**			
Model	BL	$\bar{Z}$	Model	CA	$\bar{Z}$	Model	MI	CA	$\bar{Z}$	Model	DDC	$\bar{Z}_{\text{bal}}$	$\bar{Z}_{\text{imb}}$
MvSCN	✘	−2.23	MvSCN	✘	−2.49	MvSCN	✘	✘	−1.78	MvSCN	✘	−2.41	−1.78
AECoKM	✘	−0.32	DMSC	✘	−0.54	AECoKM	✘	✓	−0.83	DMSC	✘	−0.39	0.45
EAMC	✘	−0.22	InfoDDC	✘	−0.41	EAMC	✘	✘	−0.75	InfoDDC	✓	−0.13	1.18
DMSC	✘	−0.11	EAMC	✘	−0.17	AE–DDC	✘	✘	−0.36	EAMC	✓	0.00	−0.75
AE–KM	✓	0.16	MV-IIC	✘	0.05	CoMVC	✘	✓	−0.33	MV-IIC	✘	0.01	1.06
InfoDDC	✘	0.20	AE–KM	✘	0.11	SiMVC	✘	✘	−0.12	AE–KM	✘	0.03	0.56
AE–DDC	✓	0.26	Multi-VAE	✘	0.32	AE–KM	✘	✘	0.23	AECoKM	✘	0.08	−1.54
SiMVC	✓	0.27	SiMVC	✘	0.35	AECoDDC	✘	✓	0.28	CoMVC	✓	0.30	0.25
MV-IIC	✘	0.27	AE–DDC	✘	0.56	Multi-VAE	✘	✘	0.38	SiMVC	✓	0.31	0.16
CoMVC	✘	0.29	AECoKM	✓	0.59	DMSC	✘	✘	0.45	AE–DDC	✓	0.33	0.06
Multi-VAE	✘	0.43	CoMVC	✓	0.63	MV-IIC	✓	✘	0.98	Multi-VAE	✘	0.42	0.47
AECoDDC	✘	0.65	AECoDDC	✓	0.82	InfoDDC	✓	✘	1.15	AECoDDC	✓	0.92	−0.13

Abbreviations: BL = Simple baseline, CA = Contrastive alignment, DDC = Deep divergence-based clustering MI = Mutual information, $\bar{Z}$ = Average Z-score for group.

resulting in a 3-view dataset. (iv) **Caltech7 / Caltech20**: A subset of the Caltech101 (Fei-Fei et al., 2007) dataset including 7 / 20 classes. We use the 6 different features extracted by Li *et al*. (Li et al., 2015), resulting in a 6-view dataset[2]. (v) **PatchedMNIST**: A subset of MNIST containing the first three digits, where views are extracted as 7×7 non-overlapping patches of the original image. The corner patches are dropped as they often contain little information about the digit, resulting in a dataset with 12 views. Each patch is resized to 28×28.

All views are individually normalized so that the values lie in $[0, 1]$. Following recent work on deep MVC, we train and evaluate on the full datasets (Zhou and Shen, 2020; Trosten et al., 2021; Xu et al., 2021a; Mao et al., 2021). More dataset details are provided in Supplementary Section E.

Hyperparameters. The baselines use the hyperparameters reported by the original authors, because (i) it is not feasible for us to tune hyperparameters individually for each model on each dataset; and (ii) it is difficult to tune hyperparameters in a realistic clustering setting due to the lack of labeled validation data.

New instances use the same hyperparameters as for the baselines wherever possible[3]. Otherwise, we set hyperparameters such that loss terms have the same order of magnitude, and such that the training converges. We refrain from any hyperparameter tuning that includes the dataset labels to keep the evaluation fair and unsupervised. We include a hyperparameter sweep in the supplementary, in order to assess the new instances' sensitivity to changes in their hyperparameter. However, we emphasize that the results of this sweep were *not* used to select hyperparameters for the new instances. All models use the same encoder architectures and are trained for 100 epochs with the Adam optimizer (Kingma and Ba, 2015).

Evaluation protocol. We train each model from 5 different initializations. Then we select the run that resulted in the lowest value of the loss and report the performance metrics from that run, following (Kampffmeyer et al., 2019; Trosten et al., 2021). This evaluation protocol is both fully unsupervised, and is not as impacted by poorly performing runs, as for instance the mean performance of all runs. The uncertainty of the performance metric under this model selection protocol is estimated using bootstrapping[4]. We measure clustering performance with the accuracy (ACC) and normalized mutual information (NMI). Both metrics are bounded in $[0, 1]$, and higher values correspond to better performing models, with respect to the ground truth labels.

6.2 Evaluation results

To emphasize the findings from our experiments, we compute the average Z-score for each model, for 4 groups of datasets[5]. Z-scores are calculated by subtracting the mean and dividing by the standard deviation of results, per dataset and per metric. Table 4 shows Z-scores for the groups: (i) **All datasets**.

[2]The list of classes and feature types is included in Supplementary Section E.

[3]Hyperparameters for all models are listed in Supplementary Section E.

[4]Details on the computations of metrics and uncertainty are included in Supplementary Section E.

[5]A complete table with metrics and uncertainties is included in Supplementary Section E.

(ii) **Random pairings:** Datasets generated by randomly pairing within-class instances to synthesize multiple views (NoisyMNIST, NoisyFashion, COIL-20). (iii) **Many views:** Datasets with many views (Caltech7, Caltech20, PatchedMNIST). (iv) **Balanced vs. imbalanced:** Datasets with balanced classes (NoisyMNIST, NoisyFashion, EdgeMNIST, EdgeFashion, COIL-20, PatchedMNIST) vs. datasets with imbalanced classes (Caltech7, Caltech20). Our main experimental findings are:

Dataset properties significantly impact the performance of methods. We observe that the ranking of methods varies significantly based on dataset properties, such as the number of views (Table 4c) and class (im)balance (Table 4d). Hence, there is not a single "state-of-the-art" for all datasets.

Our new instances outperform previous methods. In Table 4a we see that the simple baselines perform remarkably well, when compared to the other, more complex methods. This highlights the importance of including simple baselines like these in the evaluation. Table 4a shows that AECoDDC overall outperforms the other methods, and on datasets with many views (Table 4c) we find that InfoDDC and MV-IIC outperform the others by a large margin.

Maximization of mutual information outperforms contrastive alignment on datasets with many views. Contrastive alignment-based methods show good overall performance, but they struggle when the number of views becomes large (Table 4c). This holds for both baseline methods (as observed in Section 4), and the new instances. As in Section 4, we hypothesize that this is due to issues with representation alignment, where the presence of less informative views is more likely when the number of views becomes large. Contrastive alignment attempts to align view-specific representations to this less informative view, resulting in clusters that are harder to separate in the representation space[6]. This is further verified in Figures 2 and 3, illustrating a decrease in performance on Caltech7 for contrastive alignment-based models with 5 or 6 views. Models based on maximization of mutual information do not have the same problem. We hypothesize that this is because maximizing mutual information still allows the view-specific representations to be different, avoiding the above issues with alignment. The MI-based models also include regularization terms that maximize the entropy of view-specific representations, preventing the representations from collapsing to a single value.

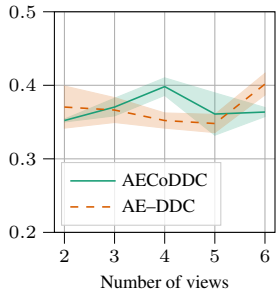

Figure 3: Accuracies on Caltech7 when the number of views is increased from 2 to 6. contrastive alignment-based models perform worse on many views, in contrast to simple baselines.

Contrastive alignment works particularly well on datasets consisting of random pairings (Table 4b). In these datasets, the class label is the only thing the views have in common. Contrastive alignment, *i.e.* learning a shared representation for all pairs within a class, thus asymptotically amounts to learning a unique representation for each class, making it easier for the clustering module to separate between classes.

The DDC clustering module performs better than the other clustering modules on balanced datasets. With the DDC clustering module, the models are end-to-end trainable – jointly optimizing all components in the model. The view-specific representations can thus be adapted to suit the clustering module, potentially improving the clustering result. DDC also has an inherent bias towards balanced clusters (Kampffmeyer et al., 2019), which helps produce better clusterings when the ground truth classes are balanced.

Reproducibility of original results. During our experiments we encountered issues with reproducibility with several of the methods from previous work. In Supplementary Section E, we include a comparison between our results and those reported by the original authors of the methods from previous work. We find that most methods use different network architectures and evaluation protocols in the original publications, making it difficult to accurately compare performance between methods and their implementations. This illustrates the difficulty of reproducing and comparing results in deep MVC, highlighting the need for a unified framework with a consistent evaluation protocol and an open-source implementation.

[6]A thorough explanation of the issues of alignment and many views is given in Supplementary Section C.

Table 5: Accuracies from ablation studies with SSL components.

(a) SV-SSL

Model	NoisyMNIST w/o SV-SSL	NoisyMNIST w/ SV-SSL	Caltech7 w/o SV-SSL	Caltech7 w/ SV-SSL
DMSC	0.54	0.66 (+0.12)	0.35	0.50 (+0.15)
AE–DDC	1.00	1.00 (0.00)	0.41	0.40 (0.00)
AE–KM	0.67	0.74 (+0.07)	0.39	0.44 (+0.05)
AECoDDC	1.00	1.00 (0.00)	0.38	0.36 (−0.02)
AECoKM	0.56	1.00 (+0.44)	0.22	0.20 (−0.02)

(b) MV-SSL

Model	NoisyMNIST w/o MV-SSL	NoisyMNIST w/ MV-SSL	Caltech7 w/o MV-SSL	Caltech7 w/ MV-SSL
EAMC	1.00	0.83 (−0.17)	0.36	0.44 (+0.08)
Multi-VAE	0.52	0.98 (+0.46)	0.31	0.47 (+0.15)
CoMVC	1.00	1.00 (0.00)	0.41	0.38 (−0.02)
AECoDDC	1.00	1.00 (0.00)	0.40	0.36 (−0.04)
AECoKM	0.74	1.00 (+0.26)	0.44	0.20 (−0.24)
InfoDDC	1.00	0.90 (−0.10)	0.41	0.51 (+0.10)
MV-IIC	0.52	0.52 (0.00)	0.53	0.53 (0.00)

6.3 Effect of SSL components

Table 5 shows the results of ablation studies with the SV-SSL and MV-SSL components[7]. As we observed in Section 4, these results show that having at least one form of SSL is beneficial for the performance of all models, with the exception being AE–DDC/AECoDDC, which on Caltech7 performs best without any self-supervision. We suspect that this particular result is due to the issues with many views and class imbalance discussed in Section 6.2. Further, we observe that having both forms of SSL is not always necessary. For instance is there no difference with and without SV-SSL for AECoDDC and AECoKM, both of which include contrastive alignment-based MV-SSL. Lastly, we note that contrastive alignment-based MV-SSL decreases performance on Caltech7 for most models. This is consistent with the results in Section 6.2 and in Figures 2 and 3, illustrating that contrastive alignment is not suitable for datasets with a large number of views.

7 Conclusion

We investigate the role of self-supervised learning (SSL) in deep MVC. To properly evaluate models and components, we develop DeepMVC – a new unified framework for deep MVC, including the majority of recent methods as instances. By leveraging the new insight from our framework, we develop 6 new DeepMVC instances with several promising forms of SSL, which perform remarkably well compared to previous methods. We conduct a thorough experimental evaluation of our new instances, previous methods, and their DeepMVC components – and find that SSL is a crucial component in state-of-the-art methods for deep MVC. However, we observe that the popular contrastive alignment worsens performance when the number of views becomes large. Further, we find that performance of methods depends heavily on dataset characteristics, such as number of views, and class imbalance. Developing methods that are robust towards changes in these properties can thus result in methods that perform well over a wide range of multi-view clustering problems. To this end, we make the following recommendations for future work in deep MVC:

Improving contrastive alignment or maximization of mutual information to handle both few and many views. Addressing the pitfalls of alignment to improve contrastive alignment-based methods on many views, is certainly a promising direction for future research. Similarly, we believe that improving the methods based on maximization of mutual information on few views, will result in better performing models.

Developing end-to-end trainable clustering modules that are not biased towards balanced clusters. The performance of the DDC clustering module illustrates the potential of end-to-end trainable clustering modules, which are capable of adapting the representations to produce better clusterings. Mitigating the bias towards balanced clusters thus has the potential to produce models that perform well, both on balanced and imbalanced datasets.

Proper evaluation and open-source implementations. Finally, we emphasize the importance of evaluating new methods on a representative collection of datasets, *e.g.* many views and few views, paired, imbalanced, *etc*. Also, in the reproducibility study (Supplementary Section E), we find that original results can be difficult to reproduce. We therefore encourage others to use the open-source implementation of DeepMVC, as open code and datasets, and consistent evaluation protocols, are crucial to properly evaluate models and facilitate further development of new methods and components.

[7]We found the SSL components to be most influential on performance, and therefore include the ablation results here. Ablation results for the fusion and CM components are given in the Supplementary Section E.

REPRODUCIBILITY STATEMENT

Reproducibility is one of the focus areas of our work. We found that some previous methods lacked crucial details in their publications, making it difficult to reproduce their respective results. This is a significant drawback of the deep MVC field. To address this issue, we re-implemented several recent methods, along with our new instances, in order to provide reproducible results and fair comparisons. To further facilitate reproducibility in future work, the supplementary includes our implementation of the baseline methods, the new instances, and all other code required to reproduce our experiments. The code, along with training-ready datasets will be made publicly available for download upon publication.

REFERENCES

R. Zhou and Y.-D. Shen. End-to-End Adversarial-Attention Network for Multi-Modal Clustering. In *CVPR*, 2020.

Daniel J. Trosten, Sigurd Løkse, Robert Jenssen, and Michael Kampffmeyer. Reconsidering Representation Alignment for Multi-view Clustering. In *CVPR*, 2021.

Jie Xu, Yazhou Ren, Huayi Tang, Xiaorong Pu, Xiaofeng Zhu, Ming Zeng, and Lifang He. Multi-VAE: Learning Disentangled View-Common and View-Peculiar Visual Representations for Multi-View Clustering. In *ICCV*, 2021a.

Yiqiao Mao, Xiaoqiang Yan, Qiang Guo, and Yangdong Ye. Deep Mutual Information Maximin for Cross-Modal Clustering. In *IJCAI*, 2021.

Qianqian Wang, Zhiqiang Tao, Wei Xia, Quanxue Gao, Xiaochun Cao, and Licheng Jiao. Adversarial Multiview Clustering Networks With Adaptive Fusion. *IEEE transactions on neural networks and learning systems*, 2022a.

Shiye Wang, Changsheng Li, Yanming Li, Ye Yuan, and Guoren Wang. Self-Supervised Information Bottleneck for Deep Multi-View Subspace Clustering. *arXiv:2204.12496 [cs]*, 2022b.

Z. Li, Q. Wang, Z. Tao, Q. Gao, and Z. Yang. Deep Adversarial Multi-view Clustering Network. In *IJCAI*, 2019.

Zhenyu Huang, Joey Tianyi Zhou, Xi Peng, Changqing Zhang, Hongyuan Zhu, and Jiancheng Lv. Multi-view Spectral Clustering Network. In *IJCAI*, 2019a.

Bowen Xin, Shan Zeng, and Xiuying Wang. Self-Supervised Deep Correlational Multi-View Clustering. In *IJCNN*, 2021.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967.

Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On Deep Multi-View Representation Learning. In *ICML*, 2015.

Mahdi Abavisani and Vishal M. Patel. Deep Multimodal Subspace Clustering Networks. *IEEE Journal of Selected Topics in Signal Processing*, 2018.

Ming Yin, Weitian Huang, and Junbin Gao. Shared Generative Latent Representation Learning for Multi-View Clustering. In *AAAI*, 2020.

Xiaoliang Tang, Xuan Tang, Wanli Wang, Li Fang, and Xian Wei. Deep Multi-view Sparse Subspace Clustering. In *ICNCC*, 2018.

Xiukun Sun, Miaomiao Cheng, Chen Min, and Liping Jing. Self-Supervised Deep Multi-View Subspace Clustering. In *ACML*, 2019.

Xianchao Zhang, Xiaorui Tang, Linlin Zong, Xinyue Liu, and Jie Mu. Deep Multimodal Clustering with Cross Reconstruction. In *PAKDD*, 2020a.

Xianchao Zhang, Jie Mu, Linlin Zong, and Xiaochun Yang. End-To-End Deep Multimodal Clustering. In *ICME*, 2020b.

Linlin Zong, Faqiang Miao, Xianchao Zhang, and Bo Xu. Multimodal Clustering via Deep Commonness and Uniqueness Mining. In *ICIKM*, 2020.

Jie Xu, Yazhou Ren, Guofeng Li, Lili Pan, Ce Zhu, and Zenglin Xu. Deep embedded multi-view clustering with collaborative training. *Information Sciences*, 2021b.

Pengfei Zhu, Binyuan Hui, Changqing Zhang, Dawei Du, Longyin Wen, and Qinghua Hu. Multi-view Deep Subspace Clustering Networks. *arXiv:1908.01978 [cs]*, 2019.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In *ICML*, 2016.

Guowang Du, Lihua Zhou, Yudi Yang, Kevin Lü, and Lizhen Wang. Deep Multiple Auto-Encoder-Based Multi-view Clustering. *Data Science and Engineering*, 2021.

Michael Kampffmeyer, Sigurd Løkse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Børre Salberg, and Robert Jenssen. Deep Divergence-based Approach to Clustering. *Neural Networks*, 2019.

Shuning Huang, Kaoru Ota, Mianxiong Dong, and Fanzhang Li. MultiSpectralNet: Spectral Clustering Using Deep Neural Network for Multi-View Data. *IEEE Transactions on Computational Social Systems*, 2019b.

Xu Ji, Andrea Vedaldi, and Joao Henriques. Invariant Information Clustering for Unsupervised Image Classification and Segmentation. In *ICCV*, 2019.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, 2017.

S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). Technical report, 1996.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 2007.

Yeqing Li, Feiping Nie, Heng Huang, and Junzhou Huang. Large-Scale Multi-View Spectral Clustering via Bipartite Graph. In *AAAI*, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Robert Jenssen, Jose C. Principe, Deniz Erdogmus, and Torbjørn Eltoft. The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels. *Journal of the Franklin Institute*, 2006.

SUPPLEMENTARY MATERIAL

A INTRODUCTION

Here, we provide additional details on the proposed new instances of DeepMVC; the datasets used for evaluation; the hyperparameters used by baselines and new instances; and the computation of metrics and uncertainties used in our evaluation protocol. We also include the full list of recent methods and their DeepMVC components, the complete table of results from the experimental evaluation. Finally, we include additional experiments and analyses of reproducibility, hyperparameters, and the Fusion and CM components.

In the supplementary archive includes a directory named `code/`, which includes our implementation of the DeepMVC framework, as well as the datasets and the evaluation protocol used in our experiments. See `code/README.md` for more details about the implementation, and how to reproduce our results. The implementation will be made publicly available upon publication of our paper.

B PREVIOUS METHODS AS INSTANCES OF DEEPMVC

The full list of recent methods and their DeepMVC components is given in Table 6. We observe that all but one model includes at least one form of SSL, but the type of SSL, and also fusion and CM, vary significantly for the different models. This illustrates the importance of the SSL components in deep MVC, as well as the need for a unified framework with a consistent evaluation protocol, in order to properly compare and evaluate methods.

C CONTRASTIVE ALIGNMENT AND MANY VIEWS

Trosten et al. (2021) highlight three issues with aligning distributions of view-specific representations in deep MVC:

1. **View prioritization**: When view-specific representations from different views are identical, it is not possible for the CM to prioritize between them to find the best possible clustering. Put differently, it is not possible for the CM to put un-equal emphasis on different views.
2. **Overlapping clusters in less informative views**: Some views might not contain the required information to separate between subsets of clusters in the datasets. When other, more informative views are aligned to these less informative views, the non-separability of clusters will be propagated from the less informative views to the view-specific representations *of all views*.
3. **Mis-aligned label distributions**: When the alignment focuses only on distributions (and not on aligning representations on the instance level), it can result in mis-aligned label distributions in the representation space. This means that a cluster from one view can be aligned to a different cluster from another view, which results in clusters that are harder to separate for the CM.

Trosten et al. (2021) only consider drawbacks of aligning *distributions* of representations. However, we emphasize that issues 1 and 2 above also apply to contrastive alignment as they are consequences of making the view-specific representations identical for a given instance.

In the following, we will focus on contrastive alignment and issue 2, since it is the one with the largest potential impact on the models' performance. An essential cause for this issue is the mapping performed by the view-specific encoders. Suppose we have a two-view dataset with two clusters. In view 1, the clusters are nicely separated, but in view 2 they are inseparable. In this case, the view specific encoder for view 2 will not be able to produce representations where the clusters are separated, because they are inseparable in the input space. However, despite the clusters being separable in view 1, the view-specific encoder for view 1 is able to produce representations where the clusters are merged. Hence, if we now force the representations from view-specific encoders 1 and 2 to be aligned, the only solution is to merge the clusters in the representation space, despite them being separable in view 1.

We hypothesize that this issue becomes more prominent when the number of clusters increases, and more importantly *when the number of views becomes large*. Specifically, when the number of views increases, it becomes increasingly likely that a subset of the views will be less informative. This is

Table 6: Full overview of methods from previous work and their DeepMVC components.

Model	Ref.	Pub.	Enc.	SV-SSL	MV-SSL	Fusion	CM
DCCAE	Wang et al. (2015)	ICML	MLP	Reconstruction	CCA	1st view	SC
DMSC	Abavisani and Patel (2018)	J. STSP	CNN	Reconstruction	–	Affinity fusion	SR, SC
DMVSSC	Tang et al. (2018)	ICNCC	CNN	Reconstruction	–	–	Sparse SR, SC
MvSN	Huang et al. (2019b)	T. CSS	MLP	Sp. Emb.	–	Weighted sum	k-means
MvSCN	Huang et al. (2019a)	IJCAI	MLP	Sp. Emb.	MSE Al.	Concat.	k-means
MvDSCN	Zhu et al. (2019)	arXiv	CNN	–	Reconstruction	Shared network	SR, SC
DAMC	Li et al. (2019)	IJCAI	MLP	–	Reconstruction	Average	DEC
S2DMVSC	Sun et al. (2019)	ACML	MLP	Reconstruction	–	MLP	SR, SC
DCMR	Zhang et al. (2020a)	PAKDD	MLP	Variational Reconstruction	Variational Reconstruction	MLP	k-means
DMMC	Zhang et al. (2020b)	ICME	MLP	Reconstruction	–	MLP	Fusion output
DCUMC	Zong et al. (2020)	ICIKM	MLP	Reconstruction	Commonness uniqueness	MLP	k-means
SGLR-MVC	Yin et al. (2020)	AAAI	MLP	–	Variational Reconstruction	Weighted sum	GMM
EAMC	Zhou and Shen (2020)	CVPR	MLP	–	Distribution Al., Kernel Al.	Attention	DDC
MVC-MAE	Du et al. (2021)	DSE	MLP	Reconstruction, Ngh. preserv.	Contrastive Al.	–	DEC
SDC-MVC	Xin et al. (2021)	IJCNN	MLP	–	CCA	Concat.	DEC
DEMVC	Xu et al. (2021b)	Inf. Sci.	CNN	Reconstruction	–	–	DEC
SiMVC	Trosten et al. (2021)	CVPR	MLP/ CNN	–	–	Weighted sum	DDC
CoMVC	Trosten et al. (2021)	CVPR	MLP/ CNN	–	Contrastive Al.	Weighted sum	DDC
Multi-VAE	Xu et al. (2021a)	ICCV	CNN	–	Variational Reconstruction	Concat.	Gumbel, k-means
DMIM	Mao et al. (2021)	IJCAI	MLP	Min. superflous information	Max. shared information	?	Encoder output
AMvC	Wang et al. (2022a)	TNNLS	MLP	–	Reconstruction	Weighted sum	DEC
SIB-MSC	Wang et al. (2022b)	arXiv	CNN	–	Reconstruction, Inf. Bottleneck	Affinity fusion	SR, SC

Abbreviations: "–" = Not included, "?" = Not specified, Al. = Alignment, Concat. = Concatenate, CCA = Canonical correlation analysis, DDC = Deep divergence-based clustering, DEC = Deep embedded clustering, Inf. Bottleneck = Information bottleneck, Ngh. preserv. = Neighborhood preservation, SC = Spectral clustering, Sp. Emb. = Spectral Embedding, SR = Self-representation, Sparse SR = Sparse self-representation,

supported by the plots in Figures 2 and 3, where adding views beyond a certain number *decreases* the performance of contrastive alignment-based models.

D NEW INSTANCES OF DEEPMVC

In this section we provide details on loss functions and training for our new instances. The loss functions used to train the new instances are on the form

$$\mathcal{L}^{\text{Total}} = w^{\text{SV}}\mathcal{L}^{\text{SV}} + w^{\text{MV}}\mathcal{L}^{\text{MV}} + w^{\text{CM}}\mathcal{L}^{\text{CM}} \tag{1}$$

where $\mathcal{L}^{\text{SV}}$, $\mathcal{L}^{\text{MV}}$, and $\mathcal{L}^{\text{CM}}$ denote the losses from the SV-SSL, MV-SSL, and CM components, respectively. $(w^{\text{SV}}, w^{\text{MV}}, w^{\text{CM}})$ are optional weights for the respective losses. The weights are all set to 1, unless specified otherwise.

Reconstruction loss. We use the mean squared error (MSE) loss, given by

$$\mathcal{L}^{\text{SV}}_{\text{Reconstruction}} = \frac{1}{nV}\sum_{i=1}^{n}\sum_{v=1}^{V}||\boldsymbol{x}_i^{(v)} - \hat{\boldsymbol{x}}_i^{(v)}||^2 \tag{2}$$

where $\hat{\boldsymbol{x}}_i^{(v)}$ is the reconstruction of $\boldsymbol{x}_i^{(v)}$, *i.e.* the decoder output for the view-specific representation $\boldsymbol{z}_i^{(v)}$.

Contrastive loss. For models with contrastive alignment-based MV-SSL, we use the multi-view generalization of the NT-Xent loss (Trosten et al., 2021), given by

$$\mathcal{L}^{\text{MV}}_{\text{Contrastive}} = \frac{1}{nV(V-1)}\sum_{i=1}^{n}\sum_{v=1}^{V}\sum_{u=1}^{V}\mathbb{1}_{\{u\neq v\}}\,\ell_i^{(uv)} \tag{3}$$

where

$$\ell_i^{(uv)} = -\log\frac{\exp(s_{ii}^{(uv)})}{\sum_{s'\in\text{Neg}(\boldsymbol{z}_i^{(u)},\boldsymbol{z}_i^{(v)})}\exp(s')} \tag{4}$$

and

$$s_{ij}^{(uv)} = \frac{1}{\tau}\frac{\boldsymbol{z}_i^u\cdot\boldsymbol{z}_j^{(v)}}{||\boldsymbol{z}_i^u||\cdot||\boldsymbol{z}_j^{(v)}||} \tag{5}$$

denotes the cosine similarity between $\boldsymbol{z}_i^u$ and $\boldsymbol{z}_j^v$. The set $\text{Neg}(\boldsymbol{z}_i^{(u)}, \boldsymbol{z}_i^{(v)})$ is the set of similarities of negative pairs for the positive pair $(\boldsymbol{z}_i^{(u)}, \boldsymbol{z}_i^{(v)})$, which consists of $s_{ij}^{(uv)}$, $s_{ij}^{(uu)}$, and $s_{ij}^{(vv)}$, for all $j \neq i$. τ is a hyperparameter, which we set to 0.1 for all experiments.

Mutual information loss. To maximize the mutual information between views, we minimize the following multi-view generalization of the IIC loss (Ji et al., 2019)

$$\mathcal{L}^{\text{MV}}_{\text{MI}} = \frac{2}{V(V-1)}\sum_{u=1}^{V-1}\sum_{v=u+1}^{V} - \left(\underbrace{I(\boldsymbol{Z}^{(u)},\boldsymbol{Z}^{(v)})}_{\text{mutual information}} + (\lambda-1)\underbrace{(H(\boldsymbol{Z}^{(u)}) + H(\boldsymbol{Z}^{(v)}))}_{\text{entropy regularization}}\right) \tag{6}$$

where the summands are computed as

$$I(\boldsymbol{Z}^{(u)},\boldsymbol{Z}^{(v)}) + (\lambda-1)(H(\boldsymbol{Z}^{(u)}) + H(\boldsymbol{Z}^{(v)})) = -\sum_{a=1}^{D}\sum_{b=1}^{D}\boldsymbol{P}_{ab}^{uv}\log\frac{\boldsymbol{P}_{ab}^{(uv)}}{(\boldsymbol{P}_a^{(u)}\boldsymbol{P}_b^{(v)})^{\lambda}}, \tag{7}$$

where D denotes the dimensionality of the view-specific representations. λ is a hyperparameter that controls the strength of the entropy regularization. We set $\lambda = 10$ for InfoDDC, and $\lambda = 1.5$ for MV-IIC. The joint distribution $\boldsymbol{P}^{(uv)}$ is estimated by first computing

$$\tilde{\boldsymbol{P}}^{uv} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{z}_i^{(u)}(\boldsymbol{z}_i^{(v)})^{\top} \tag{8}$$

and then symmetrizing it

$$\boldsymbol{P}^{(uv)} = \frac{1}{2}(\tilde{\boldsymbol{P}}^{(uv)} + (\tilde{\boldsymbol{P}}^{(uv)})^\top). \tag{9}$$

We assume that each view-specific representation is normalized such that its elements sum to one, and are all non-negative. The marginals $\boldsymbol{P}^{(u)}$ and $\boldsymbol{P}^{(v)}$ are obtained by summing over the rows and columns of $\boldsymbol{P}^{(uv)}$, respectively.

Weighted sum fusion As Trosten et al. (2021), we implement the weighted sum fusion as

$$\boldsymbol{z}_i = \sum_{v=1}^{V} w^{(v)} \boldsymbol{z}_i^{(v)}, \tag{10}$$

where the weights $w^{(1)}, \ldots, w^{(V)}$ are non-negative and sum to 1. These constraints are implemented by keeping a vector of trainable, un-normalized weights, from which $w^{(1)}, \ldots, w^{(V)}$ can be computed by applying the softmax function.

DDC clustering module. The DDC (Kampffmeyer et al., 2019) clustering module consists of two fully-connected layers. The first layer calculates the hidden representation $\boldsymbol{h}_i \in \mathbb{R}^{D_{DDC}}$ from the fused representation $\boldsymbol{z}_i$. The dimensionality of the hidden representation, D_{DDC} is a hyperparameter set to 100 for all models. The second layer computes the cluster membership vector $\boldsymbol{\alpha}_i \in \mathbb{R}^k$ from the hidden representation.

DDC's loss function consists of three terms

$$\mathcal{L}_{\text{DDC}}^{\text{CM}} = \mathcal{L}_{\text{DDC, 1}} + \mathcal{L}_{\text{DDC, 2}} + \mathcal{L}_{\text{DDC, 3}}. \tag{11}$$

The three terms encourage (i) separable and compact clusters in the hidden space; (ii) orthogonal cluster membership vectors; and (iii) cluster membership vectors close to simplex corners, respectively.

The first term maximizes the pairwise Cauchy-Schwarz divergence (Jenssen et al., 2006) between clusters (represented as probability densities) in the space of hidden representations

$$\mathcal{L}_{\text{DDC, 1}} = \binom{k}{2}^{-1} \sum_{a=1}^{k-1} \sum_{b=a}^{k} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ia}\kappa_{ij}\alpha_{jb}}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ia}\kappa_{ij}\alpha_{ja} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ib}\kappa_{ij}\alpha_{jb}}} \tag{12}$$

where $\kappa_{ij} = \exp\left(-\frac{||\boldsymbol{h}_i - \boldsymbol{h}_j||^2}{2\sigma^2}\right)$ and σ is a hyperparameter. Following Kampffmeyer et al. (2019), we set σ to 15% of the median pairwise difference between the hidden representations.

The second term minimizes the pairwise inner product between cluster membership vectors

$$\mathcal{L}_{\text{DDC, 2}} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j^\top. \tag{13}$$

The third term encourages cluster membership vectors to be close to the corners of the probability simplex in $\mathbb{R}^k$

$$\mathcal{L}_{\text{DDC, 3}} = \binom{k}{2}^{-1} \sum_{a=1}^{k-1} \sum_{b=a}^{k} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ia}\kappa_{ij}m_{jb}}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ia}\kappa_{ij}m_{ja} \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ib}\kappa_{ij}m_{jb}}} \tag{14}$$

where $m_{ia} = \exp(-||\boldsymbol{\alpha}_i - \boldsymbol{e}_a||^2)$, and $\boldsymbol{e}_a$ is the a-th simplex corner.

E Experiments

E.1 Datasets

Dataset details are listed in Table 7. The supplementary archive includes pre-processed Caltech7 and Caltech20 datasets in `code/data/processed/caltech{7|20}_train.npz`. The other

Table 7: Dataset details.

Dataset	n	v	k	n_{small}	n_{big}	Dim.	Licence
NoisyMNIST (Lecun et al., 1998)	70000	2	10	6313	7877	$(28 \times 28)^2$	CC BY-SA 3.0
NoisyFashion (Xiao et al., 2017)	70000	2	10	7000	7000	$(28 \times 28)^2$	MIT
EdgeMNIST (Lecun et al., 1998)	70000	2	10	6313	7877	$(28 \times 28)^2$	CC BY-SA 3.0
EdgeFashion (Xiao et al., 2017)	70000	2	10	7000	7000	$(28 \times 28)^2$	MIT
COIL-20 (Nene et al., 1996)	480	3	20	24	24	$(64 \times 64)^3$	None
Caltech7 (Fei-Fei et al., 2007)	1474	6	7	34	798	$48, 40, 254, 1984, 512, 928$	CC BY 4.0
Caltech20 (Fei-Fei et al., 2007)	2386	6	20	33	798	$48, 40, 254, 1984, 512, 928$	CC BY 4.0
PatchedMNIST (Lecun et al., 1998)	21770	12	3	6903	7877	$(28 \times 28)^{12}$	CC BY-SA 3.0

n = number of instances, v = number of views, k = number of classes/clusters, n_{small} = number of instances in smallest class, n_{big} = number of instances in largest class, Dim. = view dimensions.

Table 8: Network architectures

CNN encoder	CNN decoder	MLP encoder	MLP decoder
`Conv`$(64 \times 3 \times 3)$	`UpSample`(2×2)	`Dense`(1024)	`Dense`(256)
`ReLU`	`TransposeConv`$(64 \times 3 \times 3)$	`BatchNorm`	`BatchNorm`
`Conv`$(64 \times 3 \times 3)$	`ReLU`	`ReLU`	`ReLU`
`BatchNorm`	`TransposeConv`$(64 \times 3 \times 3)$	`Dense`(1024)	`Dense`(1024)
`ReLU`	`BatchNorm`	`BatchNorm`	`BatchNorm`
`MaxPool`(2×2)	`ReLU`	`ReLU`	`ReLU`
`Conv`$(64 \times 3 \times 3)$	`UpSample`(2×2)	`Dense`(1024)	`Dense`(1024)
`ReLU`	`TransposeConv`$(64 \times 3 \times 3)$	`BatchNorm`	`BatchNorm`
`Conv`$(64 \times 3 \times 3)$	`ReLU`	`ReLU`	`ReLU`
`BatchNorm`	`TransposeConv`$(1 \times 3 \times 3)$	`Dense`(1024)	`Dense`(1024)
`ReLU`	`Sigmoid`	`BatchNorm`	`BatchNorm`
`MaxPool`(2×2)		`ReLU`	`ReLU`
		`Dense`(256)	`Dense(input dim)`
			`Sigmoid`

datasets can be generated by following the instructions in `code/README.md` (these could not be included in the archive due to limitations on space).

Caltech details. We use the same features and subsets of the Caltech101 (Fei-Fei et al., 2007) dataset as Huang et al. (2019a).

- **Features:** Gabor, Wavelet Moments, CENsus TRansform hISTogram (CENTRIST), Histogram of Oriented Gradients (HOG), GIST, and Local Binary Patterns (LBP).
- **Caltech7 classes:** Face, Motorbikes, Dolla-Bill, Garfield, Snoopy, Stop-Sign, Windsor-Chair.
- **Caltech20 classes:** Face, Leopards, Motorbikes, Binocular, Brain, Camera,Car-Side, Dolla-Bill, Ferry, Garfield, Hedgehog, Pagoda, Rhino, Snoopy, Stapler, Stop-Sign, Water-Lilly, WindsorChair, Wrench, Yin-yang.

E.2 Hyperparameters

Network architectures. The encoder and decoder architectures are listed in Table 8. MLP encoders/decoders are used for Caltech7 and Caltech20 as these contain vector data. The other datasets contain images, so CNN encoders and decoders are used for them.

Other hyperparameters. Table 9 lists other hyperparameters used for the baselines and new instances.

E.3 Computational resources

We run our experiments on a Kubernetes cluster, where jobs are allocated to nodes with Intel(R) Xeon(R) E5-2623 v4 or Intel(R) Xeon(R) Silver 4210 CPUs (2 cores allocated per job); and Nvidia GeForce GTX 1080 Ti or Nvidia GeForce RTX 2080 Ti GPUs. Each job has 16 GB RAM available.

With this setup, 5 training runs on NoisyMNIST, NoisyFashion, EdgeMNIST, and EdgeFashion take approximately 24 hours. Training times for the other datasets are approximately between 1 and 3 hours.

The Dockerfile used to build our docker image can be found in `code/docker/`.

Table 9: Hyperparameters used to train the models.

Model	Batch size	Learning rate	w^{SV}	w^{MV}	w^{CM}	Pre-train	Gradient clip
DMSC	100	10^{-3}	1.0	–	–	✓	10
MvSCN	512	10^{-4}	0.999	0.001	–	✘	10
EAMC	100	†	–	1.0	1.0	✘	10
SiMVC	100	10^{-3}	–	–	1.0	✘	10
CoMVC	100	10^{-3}	–	0.1	1.0	✘	10
Multi-VAE	64	$5 \cdot 10^{-4}$	–	1.0	–	✓	10
AE–DDC	100	10^{-3}	1.0	–	1.0	✓	10
AECoDDC	100	10^{-3}	1.0	0.1	1.0	✓	10
AE–KM	100	10^{-3}	1.0	–	–	✘	10
AECoKM	100	10^{-3}	1.0	0.1	–	✘	10
InfoDDC	256	10^{-3}	–	0.1	1.0	✘	10
MV-IIC	256	10^{-3}	–	0.01	1.0	✘	10

† = EAMC (Zhou and Shen, 2020) has different learning rates for the different components, namely 10^{-5} for the encoders and clustering module, and 10^{-4} for the attention module and discriminator.

E.4 Evaluation protocol

Metrics. We measure performance using the accuracy

$$\text{ACC} = \max_{m \in \mathcal{M}} \frac{\sum_{i=1}^{n} \delta(m(\hat{y}_i) - y_i)}{n} \tag{15}$$

where $\delta(\cdot)$ is the Kronecker-delta, $\hat{y}_i$ is the predicted cluster of instance i, and y_i is the ground truth label of instance i. The maximum runs over $\mathcal{M}$, which is the set of all bijective mappings from $\{1, \ldots, k\}$ to itself.

We also compute the normalized mutual information

$$\text{NMI} = \frac{MI(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\frac{1}{2}(H(\hat{\boldsymbol{y}}) + H(\boldsymbol{y}))} \tag{16}$$

where $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_n]$, $\boldsymbol{y} = [y_1, \ldots, y_n]$, $MI(\cdot, \cdot)$ and $H(\cdot)$ denotes the mutual information and entropy, respectively.

Uncertainty estimation. The uncertainty of our performance statistic can be estimated using bootstrapping. Suppose the R training runs result in the R tuples

$$(L_1, M_1), \ldots, (L_R, M_R) \tag{17}$$

where L_i is the final loss of run i, and M_i is resulting performance metric for run i. We then sample B bootstrap samples uniformly from the original results

$$(L_j^b, M_j^b) \sim \text{Uniform}\{(L_1, M_1), \ldots, (L_R, M_R)\}, \quad j = 1, \ldots, R, \quad b = 1, \ldots B. \tag{18}$$

The performance statistic for bootstrap sample b is then given by

$$M_\star^b = M_{j_\star^b}^b, \quad j_\star^b = \arg\min_{j=1,\ldots,R} \{L_j^b\}. \tag{19}$$

We then estimate the uncertainty of the performance statistic by computing the standard deviation of the bootstrap statistics $M_\star^1, \ldots M_\star^B$

$$\hat{\sigma}_{M_\star} = \sqrt{\frac{\sum_{b=1}^{B} (M_\star^b - \bar{M}_\star)^2}{B-1}}, \text{ where } \quad \bar{M}_\star = \frac{\sum_{b=1}^{B} M_b^\star}{B}. \tag{20}$$

E.5 Results

Evaluation results. The complete evaluation results are given in Table 10.

Ablation study – Fusion and Clustering module. Table 11 shows the results of ablation studies with the fusion and clustering module (CM) components. Since these components can not be completely removed, we instead replace more complicated components, with the simplest possible component. Thus, we replace weighted sum with concatenate for the fusion component, and DDC with k-means for the CM component.

Table 10: Clustering results. Standard deviations (obtained by bootstrapping) are shown in parentheses.

	NoisyMNIST		NoisyFashion		EdgeMNIST		EdgeFashion	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DMSC	0.66 (0.02)	0.67 (0.01)	0.49 (0.05)	0.48 (0.03)	0.51 (0.02)	0.47 (0.02)	0.52 (0.01)	0.47 (0.00)
MvSCN	0.15 (0.00)	0.02 (0.00)	0.14 (0.00)	0.01 (0.00)	0.14 (0.00)	0.01 (0.01)	0.12 (0.00)	0.03 (0.00)
EAMC	0.83 (0.04)	0.90 (0.02)	0.61 (0.02)	0.71 (0.02)	0.76 (0.05)	0.79 (0.03)	0.51 (0.03)	0.47 (0.01)
SiMVC	**1.00** (0.02)	**1.00** (0.02)	0.52 (0.02)	0.51 (0.02)	0.89 (0.06)	0.90 (0.04)	0.61 (0.01)	0.56 (0.02)
CoMVC	**1.00** (0.00)	**1.00** (0.00)	0.67 (0.03)	0.68 (0.03)	**0.97** (0.08)	**0.94** (0.07)	0.56 (0.03)	0.52 (0.01)
Multi-VAE	0.98 (0.05)	0.96 (0.02)	0.62 (0.02)	0.60 (0.01)	0.85 (0.01)	0.76 (0.01)	0.58 (0.01)	**0.64** (0.00)
AE–KM	0.74 (0.03)	0.71 (0.00)	0.58 (0.02)	0.59 (0.01)	0.60 (0.00)	0.57 (0.00)	0.54 (0.00)	0.58 (0.00)
AE–DDC	**1.00** (0.04)	**1.00** (0.03)	0.69 (0.06)	0.65 (0.05)	0.88 (0.11)	0.88 (0.09)	0.60 (0.01)	0.58 (0.01)
AECoKM	**1.00** (0.00)	0.99 (0.00)	0.63 (0.07)	0.73 (0.03)	0.38 (0.03)	0.31 (0.02)	0.39 (0.04)	0.34 (0.02)
AECoDDC	**1.00** (0.00)	0.99 (0.00)	**0.80** (0.02)	**0.77** (0.01)	0.89 (0.10)	0.90 (0.09)	**0.67** (0.09)	0.62 (0.06)
InfoDDC	0.90 (0.05)	0.92 (0.04)	0.54 (0.03)	0.52 (0.04)	0.62 (0.04)	0.52 (0.06)	0.43 (0.01)	0.43 (0.03)
MV-IIC	0.52 (0.04)	0.79 (0.02)	0.52 (0.07)	0.74 (0.02)	0.31 (0.04)	0.21 (0.05)	0.52 (0.04)	0.59 (0.04)

	COIL-20		Caltech7		Caltech20		PatchedMNIST	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DMSC	–† (–)	–† (–)	0.50 (0.03)	0.50 (0.02)	0.35 (0.01)	0.55 (0.00)	–† (–)	–† (–)
MvSCN	0.21 (0.00)	0.23 (0.01)	0.29 (0.02)	0.02 (0.00)	0.13 (0.01)	0.09 (0.01)	–† (–)	–† (–)
EAMC	0.39 (0.15)	0.52 (0.22)	0.44 (0.02)	0.23 (0.03)	0.22 (0.04)	0.23 (0.02)	–‡ (–)	–‡ (–)
SiMVC	**0.90** (0.04)	**0.96** (0.02)	0.41 (0.02)	0.51 (0.09)	0.34 (0.02)	0.52 (0.01)	0.84 (0.04)	0.64 (0.11)
CoMVC	0.87 (0.03)	**0.96** (0.02)	0.38 (0.01)	0.55 (0.02)	0.34 (0.01)	0.59 (0.02)	0.73 (0.12)	0.57 (0.19)
Multi-VAE	0.74 (0.02)	0.84 (0.01)	0.47 (0.02)	0.47 (0.01)	0.40 (0.01)	0.57 (0.01)	0.94 (0.00)	0.77 (0.00)
AE–KM	0.88 (0.04)	0.92 (0.01)	0.44 (0.03)	0.52 (0.01)	0.45 (0.02)	0.57 (0.01)	0.87 (0.00)	0.68 (0.01)
AE–DDC	0.80 (0.04)	0.93 (0.02)	0.40 (0.01)	0.54 (0.07)	0.34 (0.01)	0.44 (0.03)	0.77 (0.10)	0.59 (0.17)
AECoKM	0.84 (0.04)	0.94 (0.02)	0.20 (0.01)	0.05 (0.00)	0.22 (0.02)	0.27 (0.02)	0.96 (0.00)	0.85 (0.00)
AECoDDC	0.87 (0.01)	**0.96** (0.00)	0.36 (0.01)	0.43 (0.03)	0.31 (0.02)	0.51 (0.02)	**0.99** (0.00)	**0.97** (0.00)
InfoDDC	0.25 (0.04)	0.54 (0.03)	0.51 (0.01)	0.60 (0.04)	**0.58** (0.07)	**0.63** (0.03)	**0.99** (0.00)	0.96 (0.00)
MV-IIC	0.83 (0.05)	0.94 (0.02)	**0.53** (0.00)	**0.63** (0.04)	0.49 (0.01)	0.61 (0.01)	0.97 (0.00)	0.90 (0.01)

† = training ran out of memory, ‡ = training resulted in NaN loss.

Table 11: Accuracies from ablation studies with the Fusion and CM components.

(a) Fusion

Model	NoisyMNIST		Caltech7	
	Concat.	Weighted	Concat.	Weighted
SiMVC	1.00	1.00 (0.00)	0.36	0.41 (+0.04)
CoMVC	1.00	1.00 (0.00)	0.42	0.38 (−0.04)
AE–DDC	1.00	1.00 (0.00)	0.36	0.40 (+0.04)
AECoDDC	1.00	1.00 (0.00)	0.39	0.36 (−0.03)
InfoDDC	0.93	0.90 (−0.03)	0.36	0.51 (+0.15)

(b) CM

Model	NoisyMNIST		Caltech7	
	k-means	DDC	k-means	DDC
SiMVC	0.67	1.00 (+0.33)	0.39	0.41 (+0.01)
CoMVC	0.56	1.00 (+0.44)	0.22	0.38 (+0.16)
AE–DDC	0.74	1.00 (+0.26)	0.44	0.40 (−0.04)
AECoDDC	1.00	1.00 (0.00)	0.20	0.36 (+0.16)
InfoDDC	0.14	0.90 (+0.76)	0.59	0.51 (−0.08)

Table 12: Accuracies from our experiment vs. accuracies reported by the original authors. † = method is originally evaluated on a slightly different dataset.

Model	Dataset	Orig.	Ours
MvSCN	N-MNIST	0.99	0.15 (−0.84)
	Caltech20	0.59	0.13 (−0.46)
EAMC	E-MNIST	0.67	0.76 (+0.09)
SiMVC	E-MNIST	0.86	0.89 (+0.03)
	E-Fashion	0.57	0.61 (+0.04)
	COIL-20	0.78	0.90 (+0.12)
CoMVC	E-MNIST	0.96	0.97 (+0.01)
	E-Fashion	0.60	0.56 (−0.04)
	COIL-20	0.89	0.87 (−0.02)
Multi-VAE	N-MNIST†	1.00	0.98 (−0.02)
	N-Fashion†	0.91	0.62 (−0.29)
	COIL-20	0.98	0.74 (−0.24)

For the fusion component, we see that the weighted sum tends to improve over the concatenation. For the CM, we observe that the performance is better with DDC than with k-means on NoisyMNIST, but the improvement more varied on Caltech7. This is consistent with what we observed in the evaluation results in the main paper.

Reproducibility of original results. Table 12 compares the results of our re-implementation of the baselines, to the results reported by the original authors. The comparison shows large differences in performance for several methods, and the differences are particularly large for MvSCN and Multi-VAE. For MvSCN, we do not use the same autoencoder preprocessing of the data. We also had difficulties getting the Cholesky decomposition to converge during training. For MultiVAE, we note that NoisyMNIST and NoisyFashion are generated without noise in the original paper, possibly resulting in datasets that are simpler to cluster. We were however not able to determine the reason for the difference in performance on COIL-20.

Additionally, all methods use different network architectures and evaluation protocols in the original publications, making it difficult to accurately compare performance between methods and their implementations. This illustrates the difficulty of reproducing and comparing results in deep MVC, highlighting the need for a unified framework with a consistent evaluation protocol and an open-source implementation.

Sensitivity to hyperparameters Table 13 shows the results of hyperparameter sweeps for the following hyperparameters:

- Weight of reconstruction loss (w^{SV}).
- Weight of contrastive loss (w^{MV}).
- Temperature in contrastive loss (τ).
- Weight of entropy regularization (λ).

We emphasize that these results were *not* used to tune hyperparameters for the new instances. Rather, they are included to investigate how robust these methods are towards changes in the hyperparameter configuration. The results show that the new instances are mostly insensitive to changes in their hyperparameters. We however observe two cases where the hyperparameter configurations can have significant impact on the model performance. First, AECoDDC shows a drop in performance when the weight of the contrastive loss is set to high on Caltech7 (Table 13b). This is consistent with our observations regarding contrastive alignment on datasets with many views. Second, InfoDDC and MV-IIC performs worse when the entropy regularization weight is set too low, indicating that sufficient regularization is required for these models to perform well.

Table 13: Results (NMI) of hyperparameter sweeps for the new instances.

(a) Weight of reconstruction loss (w^{SV}).

	NoisyMNIST				Caltech7			
Rec. weight	0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0
AE–DDC	1.00 (0.03)	0.94 (0.03)	1.00 (0.03)	0.94 (0.01)	0.41 (0.01)	0.41 (0.03)	0.44 (0.02)	0.45 (0.02)
AECoDDC	0.99 (0.00)	0.99 (0.00)	0.99 (0.00)	0.99 (0.00)	0.40 (0.05)	0.33 (0.02)	0.34 (0.03)	0.49 (0.04)
AECoKM	0.74 (0.02)	0.70 (0.04)	0.74 (0.03)	0.93 (0.02)	0.07 (0.01)	0.05 (0.00)	0.04 (0.01)	0.04 (0.02)

(b) Weight of contrastive loss (w^{MV}).

	NoisyMNIST				Caltech7			
Con. weight	0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0
AECoDDC	1.00 (0.00)	0.99 (0.00)	0.99 (0.00)	0.99 (0.00)	0.46 (0.02)	0.40 (0.05)	0.19 (0.03)	0.09 (0.01)
AECoKM	0.89 (0.01)	0.73 (0.03)	0.77 (0.01)	0.67 (0.02)	0.04 (0.02)	0.04 (0.01)	0.06 (0.01)	0.05 (0.00)

(c) Temperature in the contrastive loss (τ).

	NoisyMNIST				Caltech7			
τ	0.01	0.07	0.1	1.0	0.01	0.07	0.1	1.0
AECoDDC	0.99 (0.00)	1.00 (0.00)	0.99 (0.00)	1.00 (0.00)	0.31 (0.03)	0.39 (0.01)	0.35 (0.02)	0.48 (0.01)
AECoKM	0.99 (0.00)	0.91 (0.02)	0.74 (0.02)	0.78 (0.05)	0.34 (0.01)	0.05 (0.01)	0.06 (0.01)	0.46 (0.01)

(d) Weight of the entropy regularization (λ).

	NoisyMNIST				Caltech7			
λ	0.5	1.5	5.0	10.0	0.5	1.5	5.0	10.0
MV-IIC	0.03 (0.01)	0.81 (0.01)	0.82 (0.00)	0.82 (0.00)	0.04 (0.01)	0.64 (0.04)	0.60 (0.01)	0.52 (0.01)
InfoDDC	0.21 (0.02)	0.37 (0.02)	0.84 (0.04)	0.94 (0.07)	0.60 (0.06)	0.60 (0.02)	0.57 (0.01)	0.51 (0.01)