

DeepMVC: A Unified Framework for Deep Multi-view Clustering

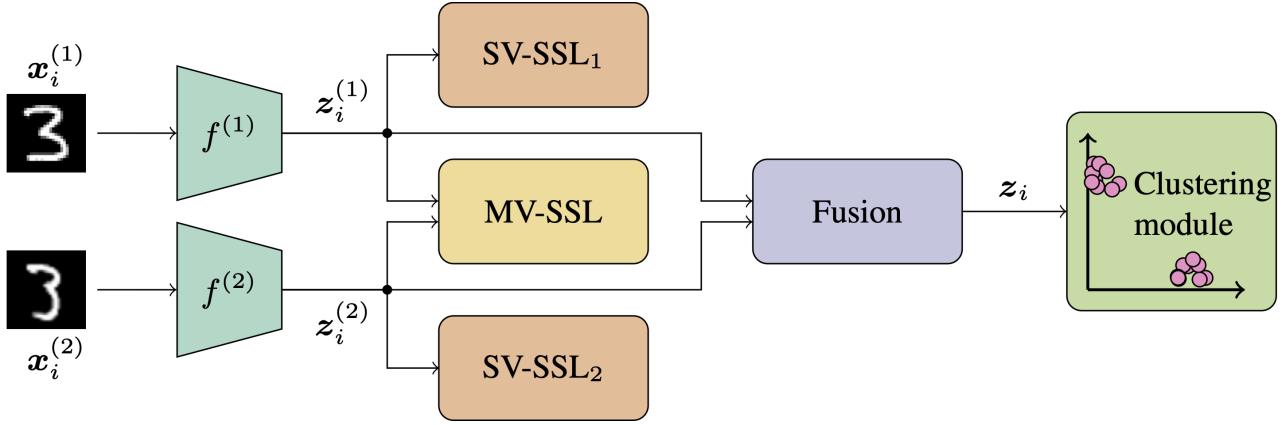


Figure 1: Overview of the DeepMVC framework and its components.

Abstract (from paper):

Recent works in deep multi-view clustering (MVC) vary significantly in the motivation, implementation and evaluation of new methods. This makes it difficult to reproduce and compare the performance of models, and to identify directions for future work. Here we illustrate that many recent methods can be regarded as instances of a new unified framework, referred to as DeepMVC. This provides novel insight into the field, allowing us to develop new instances of DeepMVC in a principled manner. We conduct an extensive experimental evaluation of recent methods and our new instances, showing that these outperform previous methods on several datasets. Finally, we present key findings from our experiments, and suggest promising directions for future research. To enhance the openness of the field, we provide an open-source implementation of DeepMVC, including the baselines and the new instances. Our implementation also includes a consistent evaluation protocol, facilitating fair and accurate evaluation of methods.

Evaluation results

Table 1: Main results from our experimental evaluation.

	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)
AECokDDC	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)
AECokDDC	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.

Preparing datasets

This repository contains the Caltech7 and Caltech20 datasets. NoisyMNIST, NoisyFashion, EdgeMNIST, EdgeFashion, and PatchedMNIST can be generated without additional downloads by running

```
python -m data.make_dataset noisymnist noisyfashionmnist edgmnist edgefashion
```

from the [src](#) directory.

Preparing the COIL-20 dataset

The COIL-20 dataset is generated by first downloading the original images from [here](#), and placing them in `data/raw/coil20`. The following command will then generate the training-ready dataset

```
python -m data.make_dataset coil20
```

Paired datasets for pre-training MvSCN

Pre-training the Siamese network for MvSCN requires a version of the dataset where samples are grouped together in positive and negative pairs. The paired datasets are generated by running

```
python -m data.pair_dataset -d <dataset_name>
```

where `<dataset_name>` is one of `noisymnist`, `noisyfashionmnist`, `edgemnist`, `edgefashionmnist`, `patchedmnist`, `caltech7`, `caltech20`, `coil20`.

Running experiments

Experiments are run with the command

```
python train.py -c <experiment_config>
```

where `<experiment_config>` is the name of a config defined in a file in [src/config/experiments](#). See [Creating new experiments](#) for further details on the Config objects. Directories containing experiments configs are

- Main evaluation: [src/config/experiments/benchmark](#)
- Ablation studies: [src/config/experiments/ablation](#)
- Increasing number of views (to reproduce Figure 2 in the paper):
[src/config/experiments/increasing_n_views](#)

Note that some new instances have names in this implementation, compared to in the paper:

Name in paper	Name in implementation	Name in experiment config
AE-KM	SAEKM	saekm
AE-DDC	SAE	sae
AECokM	CAEKM	caekm
AECokDDC	CAE	cae
InfoDDC	MIMVC	mimvc

Logging results with Weights and Biases.

`train.py` logs the experiment results to a [Weights and Biases](#) project. The project name is specified by setting `WANDB_PROJECT=<project_name>` in `src/lib/wandb_utils.py` (set to `"mvc-framework"` by default.)

Running experiments with Docker

A docker image with all dependencies can be built by running

```
docker build \
  --tag <image_tag> \
  --rm \
  --no-cache \
  --pull \
  -f docker/Dockerfile \
  .
```

from the root directory of the repository.

Overriding config parameters at the command line

Most config parameters can be overridden by specifying them at the command line. For instance, to run CoMVC on NoisyMNIST with batch size 256 instead of 100, run

```
python train.py -c noisymnist_comvc --batch_size 256
```

Note that the config hierarchy (described below) is "traversed" on the command line using `/`, so running CoMVC on NoisyMNIST without the adaptive weight on the contrastive loss is done with

```
python train.py -c noisymnist --model_config/loss_config/contrast_adaptive_wei
```

and with learning rate `1e-4`

```
python train.py -c noisymnist --model_config/optimizer_config/learning_rate 0.
```

Creating new experiments

The [Config](#) class

All experiments run with `src/train.py` are instances of [Experiment](#), a subclass of the generic [Config](#) class. Configs are implemented as hierarchical dataclass-like objects (based on [pydantic's](#) `BaseModel`), where [Experiment](#) is the top-level class.

For instance, instances of [Experiment](#) are required to specify the `dataset_config` attribute as an instance of [Dataset](#), containing the name and parameters of the dataset to use for the experiment.

All [Config](#) subclasses used in the experiments are defined in [src/config/templates](#). This directory includes [Config](#) subclasses for datasets, models, encoders, fusion, clustering modules, etc.

Defining a custom experiment

Defining a custom CoMVC experiment on NoisyMNIST from scratch is done with

```
from config.templates.experiment import Experiment
from config.templates.dataset import Dataset
from config.templates.models.simvc_comvc import CoMVC, CoMVCLoss
from config.templates.encoder import Encoder
from config.templates.optimizer import Optimizer

noisymnist_comvc_custom = Experiment(
    dataset_config=Dataset(
        name="noisymnist",
        n_train_samples=20000, # Reduce the number of samples to train quicker
    ),
    model_config=CoMVC(
        # Ene encoder per view
        encoder_configs=[
            Encoder(layers="cnn_small", input_size=[1, 28, 28]),
            Encoder(layers="cnn_small", input_size=[1, 28, 28]),
        ],
        loss_config=CoMVCLoss(
            tau=0.07 # Change the softmax temperature in the contrastive lo
        ),
        optimizer_config=Optimizer(
            learning_rate=1e-4 # Change the learning rate
        ),
    ),
    n_clusters=10, # The number of clusters is specified in the Experiment co
    batch_size=256, # so is the batch size
    n_epochs=200, # and the number of training epochs
)
```

This file should be imported in [src/config/experiments/__init__.py](#). The custom experiment can then be run with

```
python train.py -c noisymnist_comvc_custom
```