Improving Image Clustering With Multiple Pretrained CNN Feature Extractors

Joris Guérin
jorisguerin.research@gmail.com

Laboratoire d'Ingénierie des Systèmes
Physiques Et Numériques
Arts et Métiers ParisTech
Lille, France

Byron Boots
https://www.cc.gatech.edu/~bboots3

School of Interactive Computing
Georgia Institute of Technology
Atlanta, USA

Abstract

For many image clustering problems, replacing raw image data with *features* extracted by a pretrained convolutional neural network (CNN), leads to better clustering performance. However, the specific features extracted, and, by extension, the selected CNN architecture, can have a major impact on the clustering results. In practice, this crucial design choice is often decided arbitrarily due to the impossibility of using cross-validation with unsupervised learning problems. However, information contained in the different pretrained CNN architectures may be complementary, even when pretrained on the same data. To improve clustering performance, we rephrase the image clustering problem as a multi-view clustering (MVC) problem that considers multiple different pretrained feature extractors as different "views" of the same data. We then propose a multi-input neural network architecture that is trained end-to-end to solve the MVC problem effectively. Our experimental results, conducted on three different natural image datasets, show that: 1. using multiple pretrained CNNs jointly as feature extractors improves image clustering; 2. using an end-to-end approach improves MVC; and 3. combining both produces state-of-the-art results for the problem of image clustering.

1 Introduction

Image Clustering (IC) is a major research topic in machine learning which attempts to partition unlabeled images based on their content. It has been used to solve numerous problems including web-scale image clustering [12], story-line reconstruction from streams of images [17], and medical image annotation [29]. In this paper, we focus on the IC setting where the number of clusters is known in advance.

The first successful methods for IC focused on feature selection and used sophisticated algorithms to handle complex features [9, 11]. Recently, research in IC has shifted towards using features extracted from Convolutional Neural Networks (CNN) pretrained on ImageNet [23]. In [12, 16, 19, 29], pretrained CNN architectures are used to extract features from images before clustering. As shown in [24], there exist a variety of publicly available pretrained CNNs that are able to generate linearly separable latent spaces for many datasets.

© 2018. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

For unsupervised tasks, the choice of a good pretrained architecture cannot be cross-validated and thus is often arbitrary ([12, 16, 19, 29]). This is potentially problematic as [13] shows that the choice of architecture has a major impact on the clustering results.

In this paper, we aim to remove the need for this design choice. Following the intuition that different pretrained deep networks may contain complementary information (see Section 2.2), we propose to use multiple pretrained networks to generate multiple feature representations. Such representations are treated as different "views" of the data, thus casting the initial IC problem into Multi-View Clustering (MVC). The success of ensemble methods for clustering [28] suggests that such an approach can improve overall clustering results. Finally, building on the recent success of end-to-end clustering methods [1], we also propose a parallel feed-forward neural network architecture which allows us to solve the MVC problem within existing deep clustering frameworks.

1.1 Contributions

We propose to transform IC into MVC by extracting features from several different pretrained CNNs. This removes the crucial design choice of feature extractor selection. We also propose a deep learning approach to address MVC. Our experimental results suggest:

- *Image clustering can be improved by using features extracted from several pretrained CNN architectures, eliminating the need to select just one.*
- *Multi-view clustering can be improved by adopting end-to-end training.*
- *These two ideas can be combined to obtain state-of-the-art results at image clustering.*

We emphasize that the two steps of the proposed methods can be used independently. Generating multiple feature representations using several pretrained CNNs can be combined with any multi-view clustering algorithm. Similarly, the proposed architecture can be leveraged to solve any MVC problems end-to-end.

2 From Image Clustering to Multi-View Clustering

2.1 Related work

Ensemble clustering (EC) combines different clustering results to obtain a final partition of the original data with improved quality [28]. EC is composed of two steps: generation, which creates a set of partitions, and consensus, which integrates partitions into a better set of clusters. In contrast to EC, Multi-View Clustering (MVC), is concerned with finding a unified partition from multi-view data [5], which can be obtained by various sensors or represented with different descriptors. Recently, MVC has received a lot of attention. In [18], the authors propose different loss functions applied on the concatenated views, in [34] and [30] lower-dimensional subspaces are learned before clustering with standard methods.

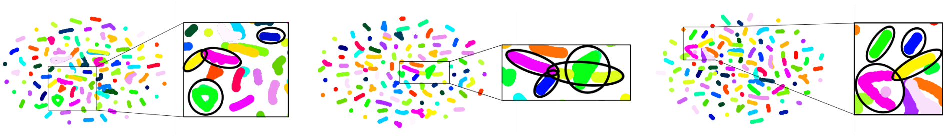
MVC and EC are closely related and have been combined in prior work. In [27], good MVC results are attained by embedding MVC within the EC framework. The authors leveraged the different views to generate independent partitions and then used a co-association-based method to obtain consensus. In both [10] and [4], generation mechanisms borrowed from EC are used to generate multiple artificial views of the data. In this paper, we propose to use multiple pretrained CNNs to generate different feature representations of an image dataset. Hence, we generate a MVC problem from an ensemble of pretrained CNN feature extractors.

2.2 Intuition

In [13], the authors show that different CNN feature extractors, pretrained on the same task (ImageNet classification), perform differently on a new target IC task. The best feature extractor does not always correspond to the best performing CNN on ImageNet, and there is no network which is consistently the best across different IC tasks. The discrepancy between the results of different methods usually motivates the use of ensemble methods.

In this case, as the different CNNs are pretrained on the same source task, the fact that they might contain complementary information for a target IC task might seem counter-intuitive. Indeed, one can expect that all networks have learned the same information. However, the task of ImageNet classification is so complex that it seems likely that there exist different latent spaces that can be leveraged to solve it. Consider the following contrived example: to recognize a car, one network might learn a wheel detector while another one might detect wing mirrors. Both sets of discriminative features would enable the solution to the ImageNet classification task, but would also carry very different information that might be useful in solving a new IC task.

This intuition is visualized on real data in Figure 1, which shows the 2D t-SNE representations [20] of the COIL100 dataset [22] for different feature extractors. Although our experimental results show that features extracted with ResNet50 present better clustering results on COIL100 (see Section 4), we can see that both VGG19 and Xception can better separate the circled classes. Hence, it may be expected that using these three feature representations as different views of the COIL100 dataset would help improve the clustering results on the final partition.



(a) Xception (b) ResNet50 (c) VGG19

Figure 1: *Best viewed in color.* 2D t-SNE visualization of features extracted by the last layer of three pretrained CNNs for the COIL100 dataset. These features form different *complementary* views of the data.

2.3 IC problem reformulation

Let $\mathcal{I} = \{I_1, ...I_n\}$ be a set of n unlabeled natural images, and let $\mathcal{FE} = \{FE_1, ...FE_m\}$ be a set of m feature extractors. In theory, $\mathcal{FE}$ can be composed of any function mapping raw pixel representations to lower-dimensional vectors, but in practice, we use pretrained deep CNNs. The first step in our approach is to generate a set of feature vectors from each element of $\mathcal{FE}$. $\forall i \in [1, ...m]$, we denote the matrix of features representing $\mathcal{I}$ as V_i, such that it's row $V_{i,k}$ is the feature vector representing I_k and extracted by FE_i:

$$V_{i,k} = FE_i(I_k). \tag{1}$$

$\mathcal{V} = \{V_1, ...V_m\}$ can be interpreted as a set of views representing the dataset. Thus, $\mathcal{V}$ is a multi-view dataset representing $\mathcal{I}$. The problem of clustering $\mathcal{V}$ is a MVC problem, which can be solved using any MVC algorithm [5]. A visual representation of the multiview generation mechanism can be seen in the blue frame of Figure 2.

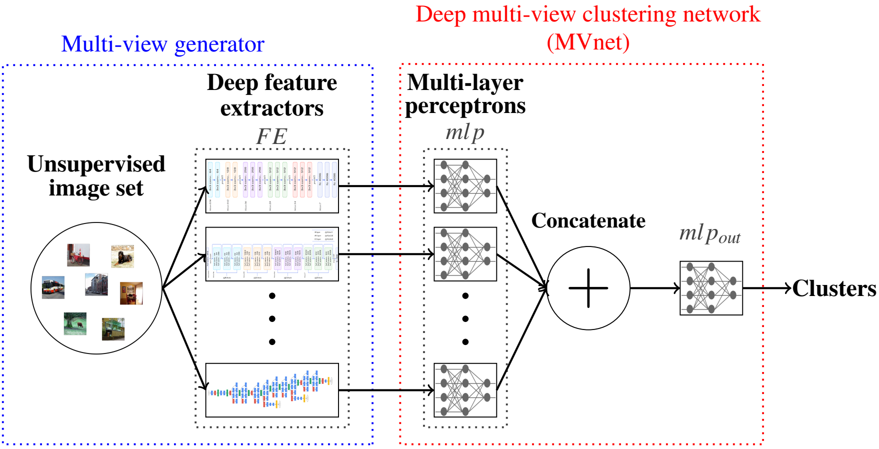


Figure 2: Two steps of the proposed approach to solve Image Clustering. Generating multiple “artificial views" of the original data from several CNNs improves the clustering results by complementarity of the feature representations. Solving Multi-View Clustering end-to-end (DMVC) improves MVC results while generating a new representation that is low-dimensional and compact.

3 Deep Multi-View Clustering

3.1 Preliminaries: Deep end-to-end clustering

End-to-end clustering methods based on neural networks have produced excellent results in the past two years. A complete literature review of the topic is outside the scope of this paper, for that we refer the reader to the following recent survey [1]. However, we describe two classes of methods relevant to our approach in more detail.

In [33], the authors introduce a connectivity-based clustering method called Joint Unsupervised Learning of deep representations and image clusters (JULE). This approach learns a new representation of the initial features with a neural network. The training of the neural network is embedded within the generic agglomerative clustering framework. The algorithm iteratively merges clusters and trains the neural network to minimize a clustering loss based on the affinity computed from the K-nearest neighbors of each point. This approach is currently producing state-of-the-art results on many image datasets.

In [32], the authors propose an end-to-end centroid-based clustering method, unsupervised Deep Embedding for Clustering analysis (DEC). This approach begins by pretraining a multi-layer perceptron (MLP) using an auto-encoder input reconstruction loss function. Then, the MLP is fine-tuned to output a set of cluster centers which define cluster assignments. Both the network optimization and the centroid optimization are based on the minimization of the Kullback-Leibler (KL) divergence between the current distribution of the features and an auxiliary target distribution, derived from high-confidence predictions. Improved Deep Embedding Clustering (IDEC), introduced in [14], modifies DEC by replacing the loss function with a combined loss, which takes into account the auto-encoder reconstruction loss during fine tuning. This approach preserves the local structure of the data and appears to improve the DEC clustering results.

3.2 Deep multi-view clustering (DMVC)

In this section, we define our approach for solving MVC. Let $\mathcal{C}_{\text{ee}}$ be any deep clustering framework. $\mathcal{C}_{\text{ee}}$ is defined by a loss function $\mathcal{L}$ and a procedure $\mathcal{P}$ to optimize the loss function. Multiple approaches have already been adopted to define the clustering-oriented loss $\mathcal{L}$ and the optimization procedure $\mathcal{P}$ (examples are discussed in Section 3.1). Applying $\mathcal{C}_{\text{ee}}$ to a new unsupervised dataset $\mathcal{D}$ requires one to specify a function f_θ, parameterized by θ, which transforms $\mathcal{D}$ into a new feature space $\mathcal{D}_\theta = f_\theta(\mathcal{D})$. From there, $\mathcal{C}_{\text{ee}}$ applies $\mathcal{P}$ to minimize $\mathcal{L}(\theta,\mathcal{D})$, producing both a good representation $\mathcal{D}_{\theta_{\text{final}}}$ and a set of cluster assignments y_{final}.

Architecture In general, f_θ is selected to be a neural network and the choice of the architecture depends on the kind of dataset. For example, when dealing with large images, f_θ can be a CNN and when $\mathcal{D}$ is composed of smaller vectors, f_θ can be a multi-layer perceptron (MLP). In the case of MVC, each element of $\mathcal{D}$ is a collection of vectors. For example, the k^{th} element of $\mathcal{D}$ is written as $\{V_{i,k},\ \forall i \in [1,...m]\}$. For this reason, to embed MVC into a deep clustering framework, we need to define a different neural network architecture for f_θ, which we call MVnet. MVnet consists of a set of m independent MLPs, denoted $mlp = \{mlp_1, ...mlp_m\}$, such that, $\forall i \in [1,...m]$, the dimension of the input layer of mlp_i is equal to the dimension of the output layer of the associated FE_i. We also define mlp_{out}, another MLP with input layer dimension equal to the sum of the dimensions of the output layers over the elements of mlp. Thus, an MVnet is composed of three layers: a parallel layer containing all the elements of mlp, followed by a concatenating layer which feeds into mlp_{out}. A visual representation of the MVnet architecture can be seen in the red box in Figure 2. We note that all the elements of mlp are independent and do not share any weights.

Training DMVC is a generic framework and MVnet can be optimized using most deep end-to-end clustering approaches. Given a deep clustering framework, MVnet should be trained on the dataset $\mathcal{V}$, from which sample K is $\mathcal{V}[K] = \{V_{i,K}, i \in [1,...m]\}$. In our experience, training MVnet from scratch does not provide good results. Instead, it is better to pretrain each mlp_i on V_i using a deep clustering framework. Then, pretrain mlp_{out} on the concatenation of the new feature representations extracted by the pretrained members of mlp. Finally, after initializing its weights with the appropriate pretrained MLPs, MVnet can be refined end-to-end on $\mathcal{V}$ to improve both the feature representation and the clustering results.

Clustering The method for obtaining the final cluster assignments depends on the deep clustering framework selected. For example, using DEC, the optimization procedure will output a set of centroids in the new feature space, which can straightforwardly be used to assign a cluster to each image. Using JULE, cluster assignments are generated during training. Data points are gradually grouped together to form clusters while f_θ is being trained.

4 Experimental setup

4.1 Datasets

We use three different image datasets to validate our work. *COIL100* [22] is composed of multiple images of the same objects from different angles. Images are centered around the object and background is neutral. *UMist* [31] is a facial recognition database. For each person, multiple pictures are taken under different light conditions and orientations. Each image is centered around the face with a neutral background. *VOC2007* [8] is an image classification dataset presenting visual objects from various classes in a realistic scene. This is a very challenging dataset for clustering: objects are not pre-segmented, backgrounds are complex, and the images are quite large. Data statistics can be found in Table 1.

Table 1: Statistics of the datasets used for our experiments.

Dataset	COIL100	UMist	VOC2007[1]
# Images	7200	575	2841
# Classes	100	20	20
Image Size[2]	128x128	112x92	Variable

[1] *We use a modified version of the VOC2007 test set. All the images presenting two or more labels have been removed in order to have ground truth to evaluate clustering quality.*

[2] *The images must be preprocessed to match the input sizes of the CNN used (*224×224 *for ResNet50 and VGG;* 299×299 *for Inception and Xception). This can be done with anti-aliasing interpolation.*

4.2 Other methods for comparison

4.2.1 DMVC evaluation

We implemented several different versions of the proposed DMVC approach. First, we consider a framework in which the MVnet weights are fixed after initialization (no end-to-end fine-tuning). We denote this DMVC-fix. DMVC-fix is implemented with two different end-to-end clustering frameworks, JULE [33] and IDEC [14]. Finally, the full DMVC pipeline is implemented within the JULE framework, which seems to perform the best on every dataset. This enables usto evaluate the influence of end-to-end training on multi-view datasets.

To evaluate the DMVC framework, we compare it against two other MVC methods. Let $\mathcal{C}$ stand for any clustering algorithm. The first naive approach consists in concatenating the different views and applying $\mathcal{C}$ on the merged representation. We denote this approach CC (concatenate + cluster). The second approach is based on ensemble clustering and is derived from [27]. Here we generate a set of partitions $\mathcal{P} = \{P_1, ..., P_m\}$ by applying $\mathcal{C}$ on each element of $\mathcal{V}$. Then, the co-association matrix (CAM) of $\mathcal{P}$ is built, measuring how many times any pair of elements have been clustered together. The CAM is then used as a graph-distance matrix by any connectivity-based clustering algorithm (spectral clustering, agglomerative clustering, etc.). In our result tables, this approach is referred to as Multi-View Ensemble Clustering (MVEC). MVEC has been shown to produced state-of-the art results on MV clustering recently [27].

4.2.2 Multiple CNN feature extraction evaluation

For any pretrained CNN, we also report the result of applying $\mathcal{C}$ to the single set of features it extracts. This is informative in evaluating the impact of multi-view generation. For each approach, both JULE and IDEC are used for $\mathcal{C}$. KMeans (KM) [2] and Agglomerative Clustering (AC) [21], two standard clustering methods, are also applied to every fixed CNN, CC and MVEC. Using simple clustering methods allows us to evaluate how multiple CNNs can benefit image clustering independently of other factors.

4.3 Practical implementations

In our experiments, we used the Keras implementations and pretrained weights [6] of five CNN architectures: two VGG architectures (VGG16 and VGG19 [25]), one ResNet architecture (ResNet50 [15]), and two Inception-like architectures: (InceptionV3 [26] and Xception [7]). When we discuss features extracted from a given CNN, we always refer to the activation just before the final linear layer for ImageNet classification. This layer is an average pooling layer for InceptionV3, Xception and ResNet50, and a fully-connected layer for both VGG architectures.

DMVC is a framework for *unsupervised* classification, hence, we should not do any hyperparameter tuning. In all of our experiments, we use default parameters for every sub-algorithm used. For both KM and AC we use the default configuration of the scikit-learn implementations [3]. For JULE and IDEC, we use the hyperparameters recommended by the authors (see original papers for more details). We only increase the learning rate to 5×10^2 for the MVnet fine tuning, as this appears to be consistently better for all the three datasets. Finally, for ensemble clustering, the co-association matrix is clustered with agglomerative clustering with average linkage. For IDEC, we generate centroids with a multilayer perceptron (MLP) with dimensions $d - 500 - 500 - 2000 - N$, where d is the extracted feature vector dimension and N is the number of classes that we are searching for. For JULE, the representation learning network is a MLP with dimensions $d - 160 - 160$. For both methods, hidden layer activations are rectified linear units. The building blocks of MVnet are of the shapes defined above.

The clustering results are evaluated using normalized mutual information (NMI), which is commonly used in unsupervised classification. NMI ranges between 0 and 1, with 1 representing perfect accuracy.

5 Experimental results

5.1 Clustering results

The clustering results on the three datasets are reported in Table 2. Table 3 reports the average results over the three datasets for each method. Computing averages makes sense in an unsupervised setting because, as cross-validation is impossible, a clustering pipeline needs to perform well on every dataset. BFN is the best fixed network over all datasets.

The method proposed in this paper is composed of two steps: multi-CNN feature extraction and deep multi-view clustering. Each of these contributions can be leveraged separately or jointly, thus the results are discussed as follows:

- In Section 5.1.1, we discuss the benefits of using several feature extractors instead of one for IC,
- In section 5.1.2 and 5.2, we discuss the advantages of using an end-to-end approach to address MVC,
- In Section 5.1.3 and 5.2, we explain the upsides of combining the two approaches to solve IC.

Table 2: Clustering performances (NMI) for different IC methods. The proposed multi-view generation methods is evaluated by comparing the five first rows (independent CNN feature extractors) and the four last ones (multi-CNN features). The proposed end-to-end method to solve MVC is evaluated by comparing DMVC to CC and MVEC.

	VOC2007				**COIL100**				**UMist**			
	JULE	IDEC	KM	AC	JULE	IDEC	KM	AC	JULE	IDEC	KM	AC
VGG16	0.687	0.666	0.660	0.665	0.989	0.963	0.939	0.956	0.920	0.771	0.707	0.755
VGG19	0.684	0.677	0.663	0.649	0.994	0.963	0.941	0.948	0.933	0.742	0.729	0.729
InceptionV3	0.768	0.760	0.620	0.675	0.984	0.957	0.942	0.953	0.823	0.705	0.646	0.692
Xception	0.759	0.779	0.668	0.720	0.986	0.955	0.933	0.955	0.829	0.707	0.591	0.678
ResNet50	0.679	0.691	0.681	0.649	**0.997**	0.973	0.962	0.967	0.919	0.784	0.686	0.723
CC	0.718	0.587	0.698	0.698	0.995	0.886	0.944	0.952	0.855	0.699	0.681	0.700
MVEC	0.785	0.782	0.728	0.741	**0.996**	0.977	0.958	0.967	**0.963**	0.797	0.748	0.761
DMVC-fix	**0.792**	0.730	N/A	N/A	**0.996**	0.973	N/A	N/A	**0.963**	0.737	N/A	N/A
DMVC	**0.810**	-	N/A	N/A	0.995	-	N/A	N/A	**0.971**	-	N/A	N/A

The two best methods for each column are in blue, the two best overall methods are in bold. N/A is for incompatible methods.

5.1.1 IC can be improved by using features extracted from several pretrained CNNs

Methods representing the data with different views extracted from multiple CNNs consistently outperform every method with a fixed feature extractor. This is also true with KM and AC, which supports the proposed way to solve image clustering as multi-view clustering, even before refinement. The only exception is for the COIL100/ResNet50 combination, but the results are similar. Moreover, NMI is very close to 1 for this configuration, which possibly indicates that the difference in performance derives from a few outliers. The importance of multi-view generation can also be observed in Table 3. The scores reported in Table 3 represent how well each method performs "in general," and show that there is no fixed feature extractor pipeline which is consistently better than a MV method across multiple datasets.

Table 3: Average NMI scores over the three datasets using JULE. The Best Fixed Network (BFN) represents the single network which performs best on average. As the feature extractor cannot be specifically selected for a given dataset, comparing multi-view results to BFN is relevant.

Clustering routine	BFN	CC	MVEC	DMVC-fix	DMVC
Average NMI score	0.870	0.856	0.915	0.917	**0.925**

Such results validate the hypothesis that different pretrained architectures contain complementary information about a new unsupervised dataset and *justify the use of multi-view generation from different CNN architectures* rather than a fixed CNN for feature extraction. In particular, for VOC2007, gathering results from multiple CNNs using MVEC or DMVC is highly beneficial, which suggests that the proposed MV framework can improve clustering for complex realistic datasets. Finally, it is interesting to note that, most of the time, CC performs worse than fixed networks. This means that features extracted from each CNN should first be processed independently before being used for clustering. This gives an experimental justification for why subnetworks of MVnet should first be pretrained separately.

5.1.2 MVC can be improved by adopting an end-to-end approach

For both VOC2007 and UMist, fine tuning MVnet using the JULE framework improves the clustering results over the DMVC with no end-to-end retraining (DMVC-fix), which *validates the use of an end-to-end approach for MVC*. Also, apart from COIL100, where results are very similar, DMVC outperforms MVEC. For COIL100, DMVC does not improve clustering over ResNet50. A possible explanation is that ResNet50 already separates the clusters well (NMI = 0.997) and there is not much left to be learned. Another reason for using deep methods for solving MVC is discussed in Section 5.2, where it is shown that solving MVC end-to-end generates a more compact, unified representation.

5.1.3 Combining multiple CNNs multi-view generation and DMVC produces state-of-the-art results at IC

To the best of our knowledge, the results reported in Table 2 represent *the new state-of-the-art for clustering these datasets*. The use of multiple pretrained feature extractors, combined with the proposed DMVC framework, enables us to outperform other approaches on the tested IC problems. The results indicate that, when facing a new unsupervised image classification dataset without any specific information about the data, we recommend that one adopt the approach proposed in this paper: transform the problem into MVC from all available pretrained CNNs and use DMVC (trained within the JULE framework) to solve it.

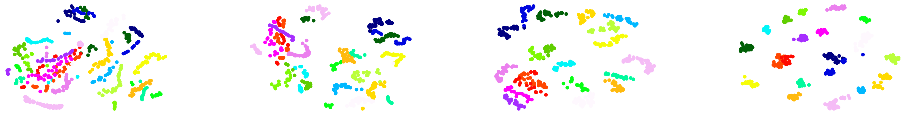
5.2 Learned representations

The quality of a clustering algorithm can also be evaluated by the quality of the new feature representation it generates. To analyze the quality of the feature representation extracted with DMVC, we re-cluster several feature representations, from different stages of the DMVC pipeline, using KM. KM is a simple clustering algorithm which performs better on representations presenting compact clusters, which are distant from each others. We choose InceptionV3 to represent the fixed CNN feature representation methods as it performs best on VOC2007. Results are reported in Table 4 and clearly demonstrate that we generate better features as we progress through the DMVC pipeline.

Table 4: Comparison of clustering performance (NMI) of KMeans applied to different representations of the dataset. If a feature representation gets a high score, it means that it presents compact clusters which are distant from each others.

	VOC2007	COIL100	UMist
InceptionV3	0.624	0.932	0.680
InceptionV3 + JULE	0.754	0.938	0.775
DMVC-fix	0.759	0.961	0.895
DMVC	**0.786**	**0.964**	**0.973**

Figure 3 provides a visual way to evaluate DMVC features. It is a 2d t-SNE representation of different features at different stages of DMVC for the UMist dataset. Comparing subfigures (b) and (c), we see that using multiple CNNs enables to obtain a better separated feature representation of the dataset. Comparing subfigures (c) and (d), we see that fine tuning DMVC end-to-end produces representations that generate more compact clusters.



(a) InceptionV3 features (b) InceptionV3 + JULE (c) DMVC-fix (d) DMVC

Figure 3: *Best viewed in color*. 2D t-SNE visualization of the features extracted from the UMist dataset at different stages of the DMVC framework. The feature representation becomes more compact and better separated as we progress in the DMVC pipeline.

6 Conclusion

We propose a two-step approach to solving the image clustering problem. First, we generate multiple representations of each image using pretrained CNN feature extractors, and reformulate the problem as a multi-view clustering problem. Second, we define a multi-input neural network architecture, MVnet, which is used to solve MVC in an end-to-end manner using any deep clustering framework. Implementing this pipeline with JULE is state-of-the-art and sets a new benchmark for image clustering on the datasets presented. This approach also has the advantage of removing the design choice of selecting a single feature extractor.

Our experimental results illustrate that different CNNs, pretrained on the same task, may contain different and complementary information about a dataset. Differences may arise from a number of sources including the architecture (number of layers, layer shape, presence of skip connections, etc.), regularization methods, or loss functions used for training. Investigating which parameters influence knowledge transfer to unsupervised tasks is an interesting axis of research for future work. Finally, we note that pretrained CNNs are used as feature extractors for many applications, not just clustering. Using multiple pretrained CNNs to define a multi-view learning problem may be appealing for other tasks where complementary information present in pretrained feature extractors may improve performance.

Acknowledgments

This work was carried out under a Fulbright Haut-de-France Scholarship as well as a scholarship from the Fondation Arts et Métiers (convention No. 8130-01). This work is also supported by the European Union's 2020 research and innovation program under grant agreement No.688807, project ColRobot (collaborative robotics for assembly and kitting in smart manufacturing).

References

[1] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[4] Michelangelo Ceci, Gianvito Pio, Vladimir Kuzmanovski, and Sašo Džeroski. Semi-supervised multi-view learning for gene network reconstruction. *PloS one*, 10(12): e0144031, 2015.

[5] Guoqing Chao, Shiliang Sun, and Jinbo Bi. A survey on multi-view clustering. *arXiv preprint arXiv:1712.06246*, 2017.

[6] François Chollet. Keras, 2015.

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[8] Mark Everingham, Andrew Zisserman, Christopher KI Williams, Luc Van Gool, Moray Allan, Christopher M Bishop, Olivier Chapelle, Navneet Dalal, Thomas Deselaers, Gyuri Dorkó, et al. The pascal visual object classes challenge 2007 (voc2007) results. 2007.

[9] Takayuki Fukui and Toshikazu Wada. Commonality preserving image-set clustering based on diverse density. In *International Symposium on Visual Computing*, pages 258–269. Springer, 2014.

[10] Hongchang Gao, Feiping Nie, Xuelong Li, and Heng Huang. Multi-view subspace clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4238–4246, 2015.

[11] Jacob Goldberger, Shiri Gordon, and Hayit Greenspan. Unsupervised image-set clustering using an information theoretic framework. *IEEE transactions on image processing*, 15(2):449–458, 2006.

[12] Yunchao Gong, Marcin Pawlowski, Fei Yang, Louis Brandy, Lubomir Bourdev, and Rob Fergus. Web scale photo hash clustering on a single machine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 19–27, 2015.

[13] Joris Guérin, Olivier Gibaru, Stéphane Thiery, and Eric Nyiri. Cnn features are also great at unsupervised classification. *arXiv preprint arXiv:1707.01700*, 2017.

[14] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 1753–1759, 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[16] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self augmented training. *arXiv preprint arXiv:1702.08720*, 2017.

[17] Gunhee Kim, Leonid Sigal, and Eric P Xing. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4225–4232, 2014.

[18] Abhishek Kumar and Hal Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 393–400, 2011.

[19] Hongfu Liu, Ming Shao, Sheng Li, and Yun Fu. Infinite ensemble for image clustering. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[20] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[21] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[22] S Nayar, S Nene, and Hiroshi Murase. Columbia object image library (coil 100). *Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96*, 1996.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015.

[24] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[27] Zhiqiang Tao, Hongfu Liu, Sheng Li, Zhengming Ding, and Yun Fu. From ensemble clustering to multi-view clustering. In *Proc. of the Twenty-Sixth Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2843–2849, 2017.

[28] Sandro Vega-Pons and José Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25 (03):337–372, 2011.

[29] Xiaosong Wang, Le Lu, Hoo-chang Shin, Lauren Kim, Mohammadhadi Bagheri, Isabella Nogues, Jianhua Yao, and Ronald M Summers. Unsupervised joint mining of deep features and image labels for large-scale radiology image categorization and scene recognition. *arXiv preprint arXiv:1701.06599*, 2017.

[30] Yang Wang, Wenjie Zhang, Lin Wu, Xuemin Lin, Meng Fang, and Shirui Pan. Iterative views agreement: An iterative low-rank based structured optimization method to multi-view spectral clustering. *arXiv preprint arXiv:1608.05560*, 2016.

[31] Harry Wechsler, Jonathon P Phillips, Vicki Bruce, Francoise Fogelman Soulie, and Thomas S Huang. *Face recognition: From theory to applications*, volume 163. Springer Science & Business Media, 2012.

[32] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487, 2016.

[33] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[34] Handong Zhao, Zhengming Ding, and Yun Fu. Multi-view clustering via deep matrix factorization. In *AAAI*, pages 2921–2927, 2017.