Learning A Task-Specific Deep Architecture For Clustering

Zhangyang Wang[*]   Shiyu Chang[*]   Jiayu Zhou[†]   Meng Wang[‡]   Thomas S. Huang[*]

Abstract

While sparse coding-based clustering methods have shown to be successful, their bottlenecks in both efficiency and scalability limit the practical usage. In recent years, deep learning has been proved to be a highly effective, efficient and scalable feature learning tool. In this paper, we propose to emulate the sparse coding-based clustering pipeline in the context of deep learning, leading to a carefully crafted deep model benefiting from both. A feed-forward network structure, named TAGnet, is constructed based on a graph-regularized sparse coding algorithm. It is then trained with task-specific loss functions from end to end. We discover that connecting deep learning to sparse coding benefits not only the model performance, but also its initialization and interpretation. Moreover, by introducing auxiliary clustering tasks to the intermediate feature hierarchy, we formulate DTAGnet and obtain a further performance boost. Extensive experiments demonstrate that the proposed model gains remarkable margins over several state-of-the-art methods.

1 Introduction

Clustering aims to learn the hidden data patterns and group similar structures in an unsupervised way. While many classical clustering algorithms have been proposed, such as K-means, Gaussian mixture model (GMM) clustering [5], maximum-margin clustering [28] and information theoretic clustering [16], most only work well when the data dimensionality is low. Since high-dimensional data exhibits dense grouping in low-dimensional embeddings [20], researchers have been motivated to first project the original data into a low-dimensional subspace [21] and then cluster on the feature embeddings. Among many feature embedding learning methods, sparse codes [27] are proven to be robust and efficient features for clustering, as verified by many [8, 31].

Effectiveness and scalability are two major concerns in designing a clustering algorithm under Big Data scenarios [6]. Conventional sparse coding models rely on iterative approximation algorithms, whose inherently sequential structure as well as the data-dependent complexity and latency often constitute a major bottleneck in the computational efficiency [11]. That also results in the difficulty when one tries to jointly optimize the unsupervised feature learning and the supervised task-driven steps [17]. Such a joint optimization usually has to rely on solving complex bi-level optimization [4], such as [26], which constitutes another efficiency bottleneck. What is more, to effectively model and represent datasets of growing sizes, sparse coding needs to refer to larger dictionaries [15]. Since the inference complexity of sparse coding increases more than linearly with respect to the dictionary size [26], the scalability of sparse coding-based clustering work turns out to be quite limited.

To conquer those limitations, we are motivated to introduce the tool of deep learning in clustering, to which there has been a lack of attention paid. The advantages of deep learning are achieved by its large learning capacity, the linear scalability with the aid of stochastic gradient descent (SGD), and the low inference complexity [3]. The feed-forward networks could be naturally tuned jointly with task-driven loss functions. On the other hand, generic deep architectures [13] largely ignore the problem-specific formulations and prior knowledge. As a result, one may encounter difficulties in choosing optimal architectures, interpreting their working mechanisms, and initializing the parameters.

In this paper, we demonstrate how to **combine the sparse coding-based pipeline into deep learning models for clustering**. The proposed framework takes advantage of both sparse coding and deep learning. Specifically, the feature learning layers are inspired by the graph-regularized sparse coding inference process, via reformulating iterative algorithms [11] into a feed-forward network, named **TAGnet**. Those layers are then jointly optimized with the task-specific loss functions from end to end. Our technical novelty and merits are summarized in three-folds:

- As a deep feed-forward model, the proposed framework provides extremely efficient inference process and high scalability to large scale data. It allows to learn more descriptive features than conventional sparse codes.

- We discover that incorporating the expertise of sparse code-based clustering pipelines [8, 31] improves our performances significantly. Moreover, it greatly facilitates the model initialization and interpretation.

- We further enforce auxiliary clustering tasks on the hierarchy of features, we develop **DTAGnet** and observe further performance boosts on the CMU MultiPIE dataset [12].

---

[*]Beckman Institute, University of Illinois at Urbana-Champaign, IL 61821. {zwang119, chang87, t-huang1}@illinois.edu.

[†]Department of Computer Science and Engineering, Michigan State University, MI 48824. jiayuz@msu.edu

[‡]Hefei University of Technology, Hefei, Anhui, 230009, China. wang-meng@hfut.edu.cn.
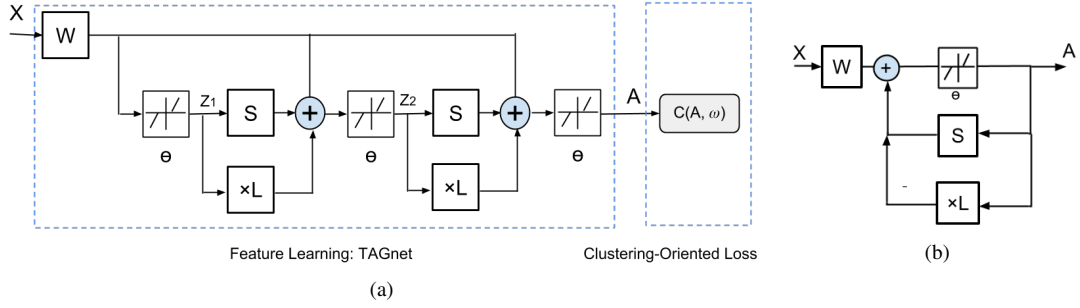
Figure 1: (a) The proposed pipeline, consisting of the TAGnet network for feature learning, followed by the clustering-oriented loss functions. The parameters $\mathbf{W}, \mathbf{S}, \boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are all learnt end-to-end from training data. (b) The block diagram of solving (3.3).

2 Related Work

2.1 Sparse coding for clustering Assuming data samples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, where $\mathbf{x}_i \in \mathbf{R}^{m \times 1}$ and $i = 1, 2, \cdots, n$. They are encoded into sparse codes $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n]$, where $\mathbf{a}_i \in \mathbf{R}^{p \times 1}$ and $i = 1, 2, \cdots, n$, using a learned dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_p]$, where $\mathbf{d}_i \in \mathbf{R}^{m \times 1}, i = 1, 2, \cdots, p$ are the learned atoms. The sparse codes are obtained by solving the following convex optimization (λ is a constant):

$$(2.1) \quad \mathbf{A} = \arg\min_{\mathbf{A}} \tfrac{1}{2} ||\mathbf{X} - \mathbf{DA}||_F^2 + \lambda \sum_i ||\mathbf{a}_i||_1,$$

In [8], the authors suggested that the sparse codes can be used to construct the similarity graph for spectral clustering [19]. Furthermore, to capture the geometric structure of local data manifolds, the graph regularized sparse codes are further suggested in [31, 29] by solving:

$$(2.2) \quad \begin{aligned} \mathbf{A} = \arg\min_{\mathbf{A}} \tfrac{1}{2} ||\mathbf{X} - \mathbf{DA}||_F^2 + \lambda \sum_i ||\mathbf{a}_i||_1 \\ + \tfrac{\alpha}{2} Tr(\mathbf{ALA^T}), \end{aligned}$$

where $\mathbf{L}$ is the graph Laplacian matrix and can be constructed from a pre-chosen pairwise similarity (affinity) matrix $\mathbf{P}$. More recently in [26], the authors suggested to simultaneously learn feature extraction and discriminative clustering, by formulating a task-driven sparse coding model [17]. They proved that such joint methods consistently outperformed non-joint counterparts.

2.2 Deep learning for clustering In [23], the authors explored the possibility of employing deep learning in graph clustering. They first learned a nonlinear embedding of the original graph by an auto encoder (AE), followed by a K-means algorithm on the embedding to obtain the final clustering result. However, it neither exploits more adapted deep architectures nor performs any task-specific joint optimization. In [7], a deep belief network with nonparametric clustering was presented. As a generative graphical model, DBN provides a faster feature learning, but is less effective than

AEs in terms of learning discriminative features for clustering. In [24], the authors extended the semi non-negative matrix factorization (Semi-NMF) model to a Deep Semi-NMF model, whose architecture resembles stacked AEs. Our proposed model is substantially different from all these previous approaches, due to its unique task-specific architecture derived from sparse coding domain expertise, as well as the joint optimization with clustering-oriented loss functions.

3 Model Formulation

The proposed pipeline consists of two blocks. As depicted in Fig. 1 (a), it is trained end-to-end in an unsupervised way. It includes a feed-forward architecture, termed *Task-specific And Graph-regularized Network* (**TAGnet**), to learn discriminative features, and the clustering-oriented loss function.

3.1 TAGnet: Task-specific And Graph-regularized Network Different from generic deep architectures, TAGnet is designed in a way to take advantage of the successful sparse code-based clustering pipelines [31, 26]. It aims to learn features that are optimized under clustering criteria, while encoding graph constraints (2.2) to regularize the target solution. TAGnet is derived from the following theorem:

THEOREM 3.1. *The optimal sparse code* $\mathbf{A}$ *from (2.2) is the fixed point of*

$$(3.3) \quad \mathbf{A} = h_{\frac{\lambda}{N}} [(\mathbf{I} - \tfrac{1}{N}\mathbf{D}^T\mathbf{D})\mathbf{A} - \mathbf{A}(\tfrac{\alpha}{N}\mathbf{L}) + \tfrac{1}{N}\mathbf{D}^T\mathbf{X}],$$

where $h_{\boldsymbol{\theta}}$ *is an element-wise shrinkage function parameterized by* $\boldsymbol{\theta}$:

$$(3.4) \quad [h_{\boldsymbol{\theta}}(\mathbf{u})]_i = sign(\mathbf{u}_i)(|\mathbf{u}_i| - \boldsymbol{\theta}_i)_+.$$

N *is an upper bound on the largest eigenvalue of* $\mathbf{D}^T\mathbf{D}$.

The complete proof of Theorem 3.1 can be found in the supplementary. Theorem 3.1 outlines an iterative algorithm to solve (2.2). Under quite mild conditions [2], after $\mathbf{A}$ is ini-

tialized, one may repeat the shrinkage and thresholding process in (3.3) until convergence. Moreover, the iterative algorithm could be alternatively expressed as the block diagram in Fig. 1 (b), where

$$(3.5) \qquad \mathbf{W} = \frac{1}{N}\mathbf{D}^T, \mathbf{S} = \mathbf{I} - \frac{1}{N}\mathbf{D}^T\mathbf{D}, \boldsymbol{\theta} = \frac{\lambda}{N}.$$

In particular, we define the new operator "$\times \mathbf{L}$": $\mathbf{A} \rightarrow -\frac{\alpha}{N}\mathbf{A}\mathbf{L}$, where the input $\mathbf{A}$ is multiplied by the pre-fixed $\mathbf{L}$ from the right side and scaled by the constant $-\frac{\alpha}{N}$.

By time-unfolding and truncating Fig. 1 (b) to a fixed number of K iterations ($K = 2$ by default)[1], we obtain the TAGnet form in Fig. 1 (a). $\mathbf{W}$, $\mathbf{S}$ and $\boldsymbol{\theta}$ are all to be learnt jointly from data. $\mathbf{S}$ and $\boldsymbol{\theta}$ are tied weights for both stages[2]. It is important to note that the output $\mathbf{A}$ of TAGnet is not necessarily identical to the predicted sparse codes by solving (2.2). Instead, the goal of TAGnet is to learn discriminative embedding that is optimal for clustering.

To facilitate training, we further rewrite (3.4) as:
$$(3.6)$$
$$[h_{\boldsymbol{\theta}}(\mathbf{u})]_i = \boldsymbol{\theta}_i \cdot \text{sign}(\mathbf{u}_i)(|\mathbf{u}_i|/\boldsymbol{\theta}_i - 1)_+ = \boldsymbol{\theta}_i h_1(\mathbf{u}_i/\boldsymbol{\theta}_i)$$

Eqn. (3.6) indicates that the original neuron with trainable thresholds can be decomposed into two linear scaling layers plus a unit-threshold neuron. The weights of the two scaling layers are diagonal matrices defined by $\boldsymbol{\theta}$ and its element-wise reciprocal, respectively.

A notable component in TAGnet is the $\times \mathbf{L}$ **branch** of each stage. The graph laplacian $\mathbf{L}$ could be computed in advance. In the feed-forward process, a $\times \mathbf{L}$ branch takes the intermediate $\mathbf{Z}_k$ ($k = 1, 2$) as the input, and applies the "$\times \mathbf{L}$" operator defined above. The output is aggregated with the output from the learnable $\mathbf{S}$ layer. In the back propagation, $\mathbf{L}$ will not be altered. In such a way, the graph regularization is effectively encoded in the TAGnet structure as a prior.

An appealing highlight of (D)TAGnet lies in its very effective and straightforward initialization strategy. With sufficient data, many latest deep networks train well with random initializations without pre-training. However, it has been discovered that poor initializations hamper the effectiveness of first-order methods (e.g., SGD) in certain cases [22]. For (D)TAGnet, it is however much easier to initialize the model in the right regime. That benefits from the analytical relationships between sparse coding and network hyperparameters defined in (3.5): we could initialize deep models from corresponding sparse coding components, the latter of which is easier to obtain. Such an advantage becomes much more important when the training data is limited

---

[1] We test larger K values (3 or 4), but they do not bring noticeable performance improvements in our clustering cases.

[2] Out of curiosity, we have also tried the architecture that treat $\mathbf{W}$, $\mathbf{S}$ and $\boldsymbol{\theta}$ in both stages as independent variables. We find that sharing parameters improves the performance.

3.2 Clustering-oriented loss functions Assuming K clusters, and $\boldsymbol{\omega} = [\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_K]$ as the set of parameters of the loss function, where $\boldsymbol{\omega}_i$ corresponds to the i-th cluster, $i = 1, 2, ..., K$. In this paper, we adopt the following two forms of clustering-oriented loss functions.

One natural choice of the loss function is extended from the popular softmax loss, and take the entropy-like form as:

$$(3.7) \qquad C(\mathbf{A}, \boldsymbol{\omega}) = -\sum_{i=1}^n \sum_{j=1}^K p_{ij} \log p_{ij}.$$

where p_{ij} denotes the the probability that sample $\mathbf{x_i}$ belongs to cluster j, $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, K$:

$$(3.8) \qquad p_{ij} = p(j|\boldsymbol{\omega}, \boldsymbol{a}_i) = \frac{e^{-\boldsymbol{\omega}_j^T \mathbf{a}_i}}{\sum_{l=1}^K e^{-\boldsymbol{\omega}_l^T \mathbf{a}_i}},$$

In testing, the predicted cluster label of input $\mathbf{a}_i$ is determined using the maximum likelihood criteria based on the predicted p_{ij}.

The maximum margin clustering (MMC) approach was proposed in [28]. MMC finds a way to label the samples by running an SVM implicitly, and the SVM margin obtained would be maximized over all possible labels [30]. By referring to the MMC definition, the authors of [26] designed the max-margin loss:

$$(3.9) \qquad C(\mathbf{A}, \boldsymbol{\omega}) = \frac{\lambda}{2}||\boldsymbol{\omega}||^2 + \sum_{i=1}^n C(\mathbf{a}_i, \boldsymbol{\omega}).$$

In the above equation, the loss for an individual sample $\mathbf{a}_i$ is defined as:

$$(3.10) \quad \begin{aligned} C(\mathbf{a}_i, \boldsymbol{\omega}) &= \max(0, 1 + f^{r_i}(\mathbf{a}_i) - f^{y_i}(\mathbf{a}_i)) \\ \text{where} \quad y_i &= \underset{j=1,...,K}{\arg\max} \ f^j(\mathbf{a}_i) \\ r_i &= \underset{j=1,...,K,j\neq y_i}{\arg\max} \ f^j(\mathbf{a}_i). \end{aligned}$$

where f^j is the prototype for the j-th cluster. In testing, the predicted cluster label of input $\mathbf{a}_i$ is determined by weight vector that achieves the maximum $\boldsymbol{\omega}_j^T \mathbf{a}_i$.

Model Complexity The proposed framework can handle large-scale and high-dimensional data effectively via the stochastic gradient descent (SGD) algorithm. In each step, the back propagation procedure requires only operations of order $O(p)$ [11]. The training algorithm takes $O(Cnp)$ time (C is a constant in terms of the total numbers of epochs, stage numbers, etc.). In addition, SGD is easy to be parallelized and thus could be efficiently trained using GPUs.

3.3 Connections to Existing Models There is a close connection between sparse coding and neural network. In [11], a feed-forward neural network, named LISTA, is proposed to efficiently approximate the sparse code $\mathbf{a}$ of input signal $\mathbf{x}$, which is obtained by solving (2.1) in advance. The LISTA network learns the hyperparameters as a general regression model from training data to their pre-solved sparse codes using back-propagation.

LISTA overlooks the useful geometric information among data points [31], and therefore could be viewed as a **special case** of TAGnet in Fig. 1 when $\alpha = 0$ (i.e., removing the $\times \mathbf{L}$ branches). Moreover, LISTA aims to approximate the "optimal" sparse codes pre-obtained from (2.1), and therefore requires the estimation of $\mathbf{D}$ and the tedious pre-computation of $\mathbf{A}$. The authors did not exploit its potential in supervised and task-specific feature learning.

4 A Deeper Look: Hierarchical Clustering by DTAGnet

Deep networks are well known for their capabilities to learn semantically rich representations by hidden layers [9]. In this section, we investigate how the intermediate features $\mathbf{Z}_k$ ($k = 1, 2$) in TAGnet (Fig. 1 (a)) can be interpreted, and further utilized to improve the model, for specific clustering tasks. Compared to related non-deep models [26], such a hierarchical clustering property is another unique advantage of being deep.

Our strategy is mainly inspired by the algorithmic framework of deeply supervised nets [14]. As in Fig. 2, our proposed Deeply-Task-specific And Graph-regularized Network (**DTAGnet**) brings in additional deep feedbacks, by associating a clustering-oriented local auxiliary loss $C_k(\mathbf{Z}_k, \boldsymbol{\omega}_k)$ ($k = 1, 2$) with each stage. Such an auxiliary loss takes the same form as the overall $C(\mathbf{A}, \boldsymbol{\omega})$, except that the expected cluster number may be different, depending on the auxiliary clustering task to be performed. The DTAGnet backpropagates errors not only from the overall loss layer, but also simultaneously from the auxiliary losses.

While seeking the optimal performance of the target clustering, DTAGnet is also driven by two auxiliary tasks that are explicitly targeted at clustering specific attributes. It enforces constraint at each hidden representation for directly making a good cluster prediction. In addition to the overall loss, the introduction of auxiliary losses gives another strong push to obtain discriminative and sensible features at each individual stage. As discovered in the classification experiments in [14], the auxiliary loss both acts as feature regularization to reduce generalization errors and results in faster convergence. We also find in Section V that every $\mathbf{Z}_k$ ($k = 1, 2$) is indeed most suited for its targeted task.

In [24], a Deep Semi-NMF model was proposed to learn hidden representations, that grant themselves an interpretation of clustering according to different attributes. The authors considered the problem of mapping facial images to their identities. A face image also contains attributes like pose and expression that help identify the person depicted. In their experiments, the authors found that by further factorizing this mapping in a way that each factor adds an extra layer of abstraction, the deep model could automatically learn latent intermediate representations that are implied for clustering identity-related attributes. Although there is a clustering interpretation, those hidden representations are not
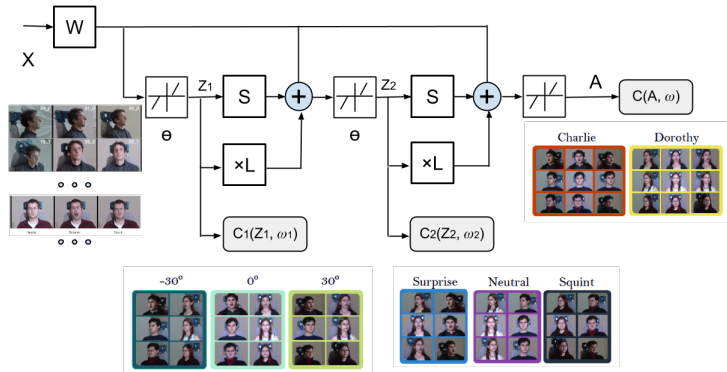

Figure 2: The DTAGnet architecture, taking the CMU MultiPIE dataset as an example. The model is able to simultaneously learn features for pose clustering ($\mathbf{Z}_1$), for expression clustering ($\mathbf{Z}_2$), and for identity clustering ($\mathbf{A}$). The first two attributes are related to and helpful for the last (overall) task. Part of image sources are referred from [12] and [24].

specifically optimized in clustering sense. Instead, the entire model is trained with only the overall reconstruction loss, after which clustering is performed using K-means on learnt features. Consequently, their clustering performance is not satisfactory. Our study shares the similar observation and motivation with [24], but in a more task-specific manner by performing the optimizations of auxiliary clustering tasks jointly with the overall task.

5 Experiment Results

5.1 Datasets and measurements We evaluate the proposed model on three publicly available datasets:

- **MNIST** [31] consists of a total number of 70, 000 quasi-binary, handwritten digit images, with digits 0 to 9. The digits are normalized and centered in fixed-size images of 28×28.

- **CMU MultiPIE** [12] contains around 750, 000 images of 337 subjects, that are captured under varied laboratory conditions. A unique property of CMU MultiPIE lies in that each image comes with labels for the identity, illumination, pose and expression attributes. That is why CMU MultiPIE is chosen in [24] to learn multi-attribute features (Fig. 2) for hierarchical clustering. In our experiments, we follow [24] and adopt a subset of 13, 230 images of 147 subjects in 5 different poses and 6 different emotions. Notably, we do not pre-process the images by using piece-wise affine warping as utilized by [24] to align these images.

- **COIL20** [18] contains 1, 440 32×32 gray scale images of 20 objects (72 images per object). The
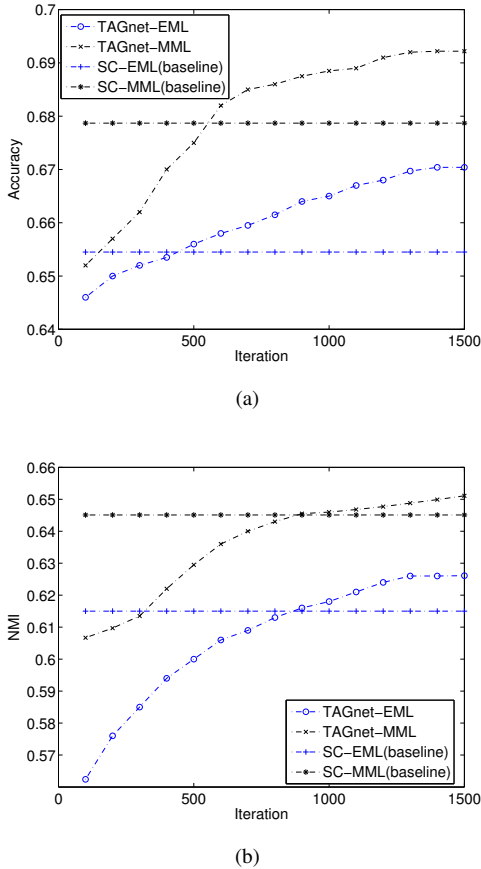
(a)

(b)

Figure 3: The accuracy and NMI plots of TAGnet-EML/TAGnet-MML on MNIST, starting from the initialization, and tested every 100 iterations. The accuracy and NMI of SC-EML/SC-MML are also plotted as baselines.

images of each object were taken 5 degree apart.

Although the paper only evaluates the proposed method using image datasets, the methodology itself is not limited to only image subjects. We apply two widely-used measures to evaluate the clustering performances: the accuracy and the Normalized Mutual Information(NMI) [31], [8]. We follow the convention of many clustering work [31, 29, 26], and do not distinguish training from testing. We train our models on all available samples of each dataset, reporting the clustering performances as our testing results. Results are averaged from 5 independent runs.

5.2 Experiment settings The proposed networks are implemented using the cuda-convnet package [13]. The network takes $K = 2$ stages by default. We apply a constant learning rate of 0.01 with no momentum to all trainable layers. The batch size of 128. In particular, to encode graph regularization as a prior, we fix $\mathbf{L}$ during model train-

ing by setting its learning rate to be 0. Experiments run on a workstation with 12 Intel Xeon 2.67GHz CPUs and 1 GTX680 GPU. The training takes approximately 1 hour on the MNIST dataset. It is also observed that the training efficiency of our model scales approximately linearly with data.

In our experiments, we set the default value of α to be 5, p to be 128, and λ to be chosen from [0.1, 1] by cross-validation[3]. A dictionary $\mathbf{D}$ is first learned from $\mathbf{X}$ by K-SVD [1]. $\mathbf{W}$, $\mathbf{S}$ and $\boldsymbol{\theta}$ are then initialized based on (3.5). $\mathbf{L}$ is also pre-calculated from $\mathbf{P}$, which is formulated by the Gaussian Kernel: $\mathbf{P}_{ij} = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{\delta^2})$ (δ is also selected by cross-validation). After obtaining the output $\mathbf{A}$ from the initial (D)TAGnet models, $\boldsymbol{\omega}$ (or $\boldsymbol{\omega}_k$) could be initialized based on minimizing (3.7) or (3.9) over $\mathbf{A}$ (or $\mathbf{Z}_k$).

5.3 Comparison experiments and analysis

5.3.1 Benefits of the task-specific deep architecture
We denote the proposed model of TAGnet plus entropy-minimization loss (EML) (3.7) as TAGnet-EML, and the one plus maximum-margin loss (MML) (3.9) as TAGnet-MML, respectively. We include the following comparison methods:

- We refer to the initializations of the proposed joint models as their **"Non-Joint"** counterparts, denoted as NJ-TAGnet-EML and NJ-TAGnet-MML (NJ short for non-joint), respectively.

- We design a **Baseline Encoder** (BE), which is a fully-connected feedforward network, consisting of three hidden layers of dimension p with ReLU neuron. It is obvious that the BE has the *same parameter complexity* as TAGnet[4]. The BEs are also tuned by EML or MML in the same way, denoted as BE-EML or BE-MML, respectively. We intend to verify our important argument, that *the proposed model benefits from the task-specific TAGnet architecture, rather than just the large learning capacity of generic deep models*.

- We compare the proposed models with their closest **"shallow"** competitors, i.e., the joint optimization methods of graph-regularized sparse coding and discriminative clustering in [26]. We re-implement their work using both (3.7) or (3.9) losses, denoted as SC-EML and SC-MML (SC short for sparse coding). Since in [26] the authors already revealed SC-MML outperforms the classical methods such as MMC and ℓ_1 graph methods, we do not compare with them again.

- We also include **Deep Semi-NMF** [24] as a state-of-the-art deep learning-based clustering work. We mainly

---

[3]The default values of α and p are inferred from the related sparse coding literature[31], and validated in experiments.

[4]except for the "θ" layers, each of which contains only p free parameters and thus ignored
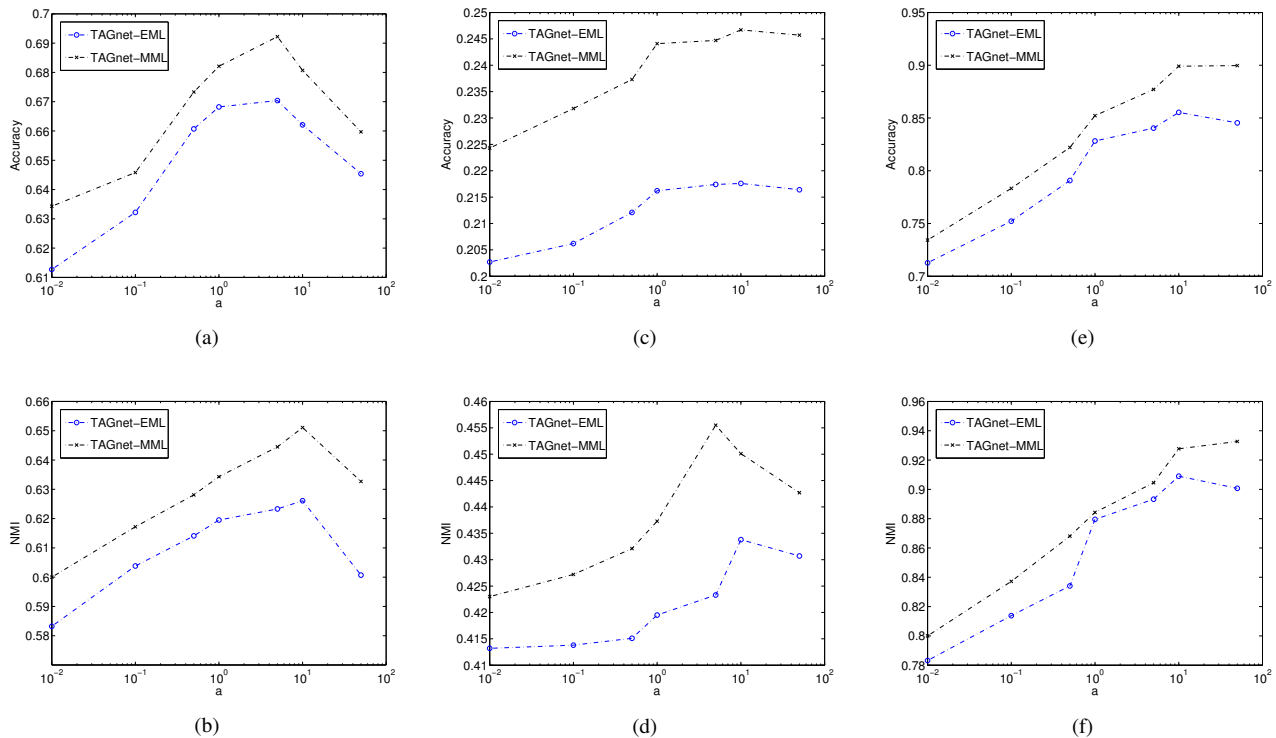
Figure 4: The clustering accuracy and NMI plots (x-axis logarithm scale) of TAGnet-EML/TAGnet-MML versus the parameter choices of α, on: (a) (b) MNIST; (c) (d) CMU MultiPIE; (e) (f) COIL20.

compare our results with their reported performances on CMU MultiPIE. [5].

As revealed by the full comparison results in Table 1, the proposed task-specific deep architectures outperform other with a noticeable margin. The underlying domain expertise guides the data-driven training in a more principled way. In contrast, the "general-architecture" baseline encoders (BE-EML and BE-MML) appear to produce much worse (even worst) results. Furthermore, it is evident that the proposed end-to-end optimized models outperform their "non-joint" counterparts. For example, on the MNIST dataset,TAGnet-MML surpasses NJ-TAGnet-MML by around 4% in accuracy and 5% in NMI.

By comparing the TAGnet-EML/TAGnet-MML with SC-EML/SC-MML, we draw a promising conclusion: adopting a more parameterized deep architecture allows a larger feature learning capacity compared to conventional sparse coding. Although similar points are well made in many other fields [13], we are interested in a closer look between the two. Fig. 3 plots the clustering accuracy and NMI curves of TAGnet-EML/TAGnet-MML on the MNIST dataset, along with iteration numbers. Each model is well

initialized at the very beginning, and the clustering accuracy and NMI are computed every 100 iterations. At first, the clustering performances of deep models are even slightly worse than sparse-coding methods, mainly since the initialization of TAGnet hinges on a truncated approximation of graph-regularized sparse coding. After a small number of iterations, the performance of the deep models surpass sparse coding ones, and continue rising monotonically until reaching a higher plateau.

5.3.2 Effects of graph regularization In (2.2), the graph regularization term imposes stronger smoothness constraints on the sparse codes with a larger α. It also happens to the TAGnet. We investigate how the clustering performances of TAGnet-EML/TAGnet-MML are influenced by various α values. From Fig. 4, we observe the identical general tendency on all three datasets. While α increases, the accuracy/NMI result will first rise then decrease, with the peak appearing between $\alpha \in [5, 10]$. As an interpretation, the local manifold information is not sufficiently encoded when α is too small ($\alpha = 0$ will completely disable the $\times \mathbf{L}$ branch of TAGnet, and reduces its to the LISTA network [11] fine-tuned by the losses). On the other hand, when α is large, the sparse codes are "over-smoothened" with a reduced discriminative ability. Note that similar phenomenons are

---

[5]With various component numbers tested in [24], we choose their best cases (60 components).

Table 1: Accuracy and NMI performance comparisons on all three datasets

		TAGnet -EML	TAGnet -MML	NJ-TAGnet -EML	NJ-TAGnet -MML	BE -EML	BE -MML	SC -EML	SC -MML	Deep Semi-NMF
MNIST	Acc	0.6704	0.6922	0.6472	0.5052	0.5401	0.6521	0.6550	0.6784	/
	NMI	0.6261	0.6511	0.5624	0.6067	0.5002	0.5011	0.6150	0.6451	/
CMU MultiPIE	Acc	0.2176	0.2347	0.1727	0.1861	0.1204	0.1451	0.2002	0.2090	0.17
	NMI	0.4338	0.4555	0.3167	0.3284	0.2672	0.2821	0.3337	0.3521	0.36
COIL20	Acc	0.8553	0.8991	0.7432	0.7882	0.7441	0.7645	0.8225	0.8658	/
	NMI	0.9090	0.9277	0.8707	0.8814	0.8028	0.8321	0.8850	0.9127	/

also reported in other relevant literature, e. g. , [31, 26].
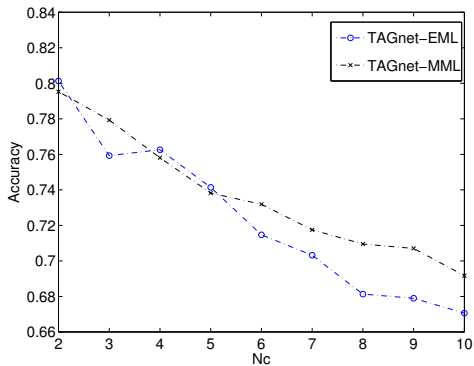
(a)
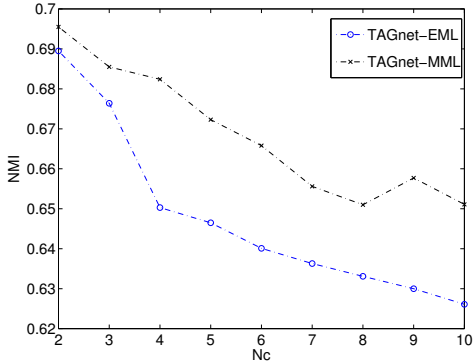
(b)

Figure 5: The clustering accuracy and NMI plots of TAGnet-EML/TAGnet-EML versus the cluster number N_c ranging from 2 to 10, on MNIST.

Furthermore, comparing among Fig. 4 (a) - (f), it is noteworthy to observe how graph regularization behaves differently on three of them. We notice that the COIL20 dataset is the one that is the most sensitive to the choice of α. Increasing α from 0.01 to 50 leads to a improvement of more than 10%, in terms of both accuracy and NMI. It verifies the significance of graph regularization when training samples are limited [29]. On the MNIST dataset, both models obtain a gain of up to 6% in accuracy and 5% in NMI, by

tuning α from 0.01 to 10. However, unlike COIL20 that almost always favors larger α, the model performance on the MNIST dataset tends to be not only saturated, but even significantly hampered when α continues rising to 50. The CMU MultiPIE dataset witnesses moderate improvements of around 2% in both measurements. It is not as sensitive to α as the other two. Potentially, it might be due to the complex variability in original images that makes the graph $\mathbf{W}$ unreliable for estimating the underlying manifold geometry. We suspect that more sophisticated graphs may help alleviate the problem, and will explore it in future.

5.3.3 Scalability and robustness On the MNIST dataset, We re-conduct the clustering experiments with the cluster number N_c ranging from 2 to 10, using TAGnet-EML/TAGnet-MML. Fig. 5 shows that the clustering accuracy and NMI change by varying the number of clusters. The clustering performance transits smoothly and robustly when the task scale changes.

To examine the proposed models' robustness to noise, we add various Gaussian noise, whose standard deviation s ranges from 0 (noiseless) to 0.3, to re-train our MNIST model. Fig. 6 indicates that both TAGnet-EML and TAGnet-MML own certain robustness to noise. When s is less than 0.1, there is even little visible performance degradation. While TAGnet-MML constantly outperforms TAGnet-EML in all experiments (as MMC is well-known to be highly discriminative [28]), it is interesting to observe in Fig. 6 that the latter one is slightly more robust to noise than the former. It is perhaps owing to the probability-driven loss form (3.7) of EML that allows for more flexibility.

5.4 Hierarchical clustering on CMU MultiPIE As observed, CMU MultiPIE is very challenging for the basic identity clustering task. However, it comes with several other attributes: pose, expression, and illumination, which could be of assistance in our proposed DTAGnet framework. In this section, we apply the similar setting of [24] on the same CMU MultiPIE subset, by setting pose clustering as the Stage I auxiliary task, and expression clustering as the Stage II auxiliary task[6]. In that way, we target $C_1(\mathbf{Z}_1, \boldsymbol{\omega}_1)$ at

---

[6]In fact, although claimed to be applicable to multiple attributes, [24] only examined the first level features for pose clustering without considering

5 clusters, $C_2(\mathbf{Z}_2, \boldsymbol{\omega}_2)$ at 6 clusters, and finally $C(\mathbf{A}, \boldsymbol{\omega})$ as 147 clusters.
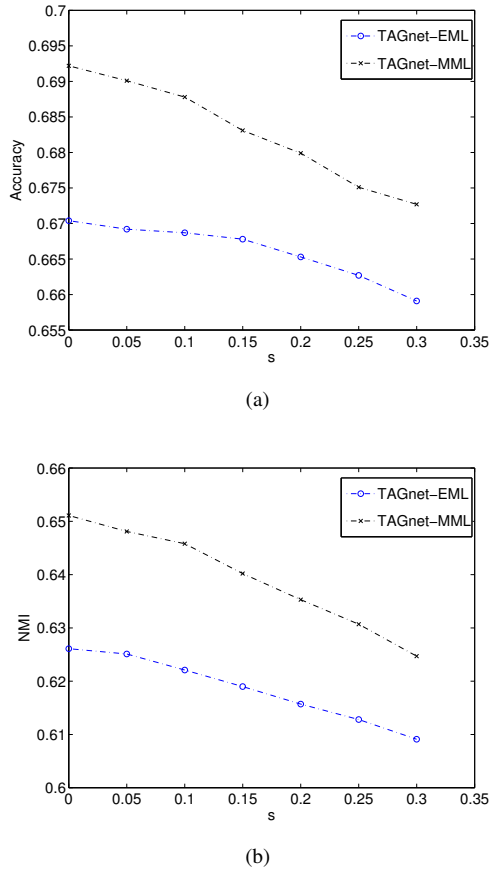
(a)

(b)

Figure 6: The clustering accuracy and NMI plots of TAGnet-EML/TAGnet-MML versus the noise level s, on MNIST.

The training of DTAGnet-EML/DTAGnet-MML follows the same aforementioned process except for considering extra back-propagated gradients from task $C_k(\mathbf{Z}_k, \boldsymbol{\omega}_k)$ in Stage k ($k = 1, 2$). After then, we test each $C_k(\mathbf{Z}_k, \boldsymbol{\omega}_k)$ separately on their targeted task. In DTAGnet, each auxiliary task is also jointly optimized with its intermediate feature $\mathbf{Z}_k$, which differentiate our methodology substantially from [24]. It is thus no surprise to see in Table 2 that each auxiliary task obtains much improved performances than [24][7] Most notably, the performances of the overall identity clustering task witness **a very impressive boost of around 7% in accuracy**. We also test DTAGnet-EML/DTAGnet-MML with only $C_1(\mathbf{Z}_1, \boldsymbol{\omega}_1)$ or $C_2(\mathbf{Z}_2, \boldsymbol{\omega}_2)$ kept. Experiments verify that by adding auxiliary tasks gradually, the overall task

---

expressions, since it relied on a warping technique to pre-process images, that gets rid of most expression variability.

[7]In [24] Table. 2, it reports that the best accuracy of pose clustering task falls around 28%, using the most suited layer features.

keeps being benefited. Those auxiliary tasks, when enforced together, can also reinforce each other mutually.

Table 2: Effects of incorporating auxiliary clustering tasks in DTAGnet-EML/DTAGnet-MML (P: Pose; E: Expression; I: Identity)

Method	Stage I		Stage II		Overall	
	Task	Acc	Task	Acc	Task	Acc
DTAGnet -EML	/	/	/	/	I	0.2176
	P	0.5067	/	/	I	0.2303
	/	/	E	0.3676	I	0.2507
	P	0.5407	E	0.4027	I	0.2833
DTAGnet -MML	/	/	/	/	I	0.2347
	P	0.5251	/	/	I	0.2635
	/	/	E	0.3988	I	0.2858
	P	0.5538	E	0.4231	I	0.3021

One might be curious that, **which one matters more in the performance boost**: the deeply task-specific architecture that brings extra discriminative feature learning, or the proper design of auxiliary tasks that capture the intrinsic data structure characterized by attributes?

Table 3: Effects of varying target cluster numbers of auxiliary tasks in DTAGnet-EML/DTAGnet-MML

Method	#clusters in Stage I	#clusters in Stage II	Overall Accuracy
DTAGnet -EML	4	4	0.2827
	8	8	0.2813
	12	12	0.2802
	20	20	0.2757
DTAGnet -MML	4	4	0.3030
	8	8	0.3006
	12	12	0.2927
	20	20	0.2805

To answer this important question, we vary the target cluster number in either $C_1(\mathbf{Z}_1, \boldsymbol{\omega}_1)$ or $C_2(\mathbf{Z}_2, \boldsymbol{\omega}_2)$, and re-conduct the experiments. Table 3 reveals that more auxiliary tasks, even those without any striaghtforward task-specific interpretation (e.g., partitioning the Multi-PIE subset into 4, 8, 12 or 20 clusters hardly makes semantic sense), may still help gain better performances. It is comprehensible that they simply promote more discriminative feature learning in a low-to-high, coarse-to-fine scheme. In fact, it is a complementary observation to the conclusion found in classification [14]. On the other hand, at least in this specific case, while the target cluster numbers of auxiliary tasks get closer to the ground-truth (5 and 6 here), the models seem to achieve

the best performances. We conjecture that when properly "matched", every hidden representation in each layer is in fact most suited for clustering the attributes corresponding to the layer of interest. The whole model can be resembled to the problem of sharing low-level feature filters among several relevant high-level tasks in convolutional networks [10], but in a distinct context.

We hence conclude that, the deeply-supervised fashion shows to be helpful for the deep clustering models, even when there are no explicit attributes for constructing a practically meaningful hierarchical clustering problem. However, it is preferable to exploit those attributes when available, as they lead to not only superior performances but more clearly interpretable models. The learned intermediate features can be potentially utilized for multi-task learning [25].

6 Conclusion

In this paper, we present a deep learning-based clustering framework. Trained from end to end, it features a task-specific deep architecture inspired by the sparse coding domain expertise, which is then optimized under clustering-oriented losses. Such a well-designed architecture leads to more effective initialization and training, and significantly outperforms generic architectures of the same parameter complexity. The model could be further interpreted and enhanced, by introducing auxiliary clustering losses to the intermediate features. Extensive experiments verify the effectiveness and robustness of the proposed models.

7 Acknowledgement

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE TSP*, 2006.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[3] Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[4] D. P. Bertsekas. *Nonlinear programming*. 1999.

[5] C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE TPAMI*, 22(7):719–725, 2000.

[6] S. Chang, W. Han, J. Tang, G. Qi, C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *ACM SIGKDD*, 2015.

[7] G. Chen. Deep learning with nonparametric clustering. *arXiv preprint arXiv:1501.03084*, 2015.

[8] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. S. Huang. Learning with l1 graph for image analysis. *IEEE TIP*, 19(4), 2010.

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[10] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520, 2011.

[11] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, pages 399–406, 2010.

[12] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 28(5), 2010.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[14] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*, 2014.

[15] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808, 2006.

[16] X. Li, K. Zhang, and T. Jiang. Minimum entropy clustering and applications to gene expression analysis. In *CSB*, pages 142–151. IEEE, 2004.

[17] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE TPAMI*, 34(4):791–804, 2012.

[18] S. A. Nene, S. K. Nayar, H. Murase, et al. Columbia object image library (coil-20). Technical report.

[19] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *NIPS*, 2:849–856, 2002.

[20] F. Nie, D. Xu, I. W. Tsang, and C. Zhang. Spectral embedded clustering. In *IJCAI*, pages 1181–1186, 2009.

[21] V. Roth and T. Lange. Feature selection in clustering problems. In *NIPS*, 2003.

[22] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, pages 1139–1147, 2013.

[23] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In *AAAI*, 2014.

[24] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pages 1692–1700, 2014.

[25] Y. Wang, D. Wipf, Q. Ling, W. Chen, and I. Wassail. Multi-task learning for subspace segmentation. In *ICML*, 2015.

[26] Z. Wang, Y. Yang, S. Chang, J. Li, S. Fong, and T. S. Huang. A joint optimization framework of sparse coding and discriminative clustering. In *IJCAI*, 2015.

[27] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE TPAMI*, 31(2):210–227, 2009.

[28] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *NIPS*, pages 1537–1544, 2004.

[29] Y. Yang, Z. Wang, J. Yang, J. Wang, S. Chang, and T. S. Huang. Data clustering by laplacian regularized l1-graph. In *AAAI*, 2014.

[30] B. Zhao, F. Wang, and C. Zhang. Efficient maximum margin clustering via cutting plane algorithm. In *SDM*, 2008.

[31] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai. Graph regularized sparse coding for image representation. *IEEE TIP*, 20(5):1327–1336, 2011.