k-meansNet: When *k*-means Meets Differentiable Programming

Xi Peng, Ivor W. Tsang, Joey Tianyi Zhou, and Hongyuan Zhu

Abstract—In this paper, we study two challenging problems. The first one is how to implement *k*-means in neural network, which enjoys efficient training based on stochastic algorithm. The second one is how to enhance the interpretability of network design for clustering. To solve the problems, we propose a neural network which is a novel formulation of the vanilla *k*-means objective. Our contribution are in twofold. From the view of neural networks, the proposed *k*-meansNet is with explicit interpretability in neural processing. We could understand not only why the network structure is presented like itself but also why it could perform data clustering. Such an interpretable neural network remarkably differs from the existing works that usually employ visualization technique to explain the result of neural network. From the view of *k*-means, three highly desired properties are achieved, *i.e.* robustness to initialization, the capability of handling new coming data, and provable convergence. Extensive experimental studies show that our method achieves promising performance comparing with 12 clustering methods on some challenging datasets.

Index Terms—Clustering, Model-based Optimization, Learning-based Iteration, Interpretable Neural Network.

◆

1 INTRODUCTION

CLUSTERING is a fundamental topic to machine learning, which aims to group similar patterns into the same cluster and dissimilar patterns into different clusters. During past decades, a variety of clustering methods [1] have been proposed and achieved huge success in various applications. In recent, the major focus of the community is paid on handling high-dimensional data whose key is addressing the linear inseparable issue.

To effectively clustering high-dimensional data, a variety of methods have been proposed, *e.g.* spectral clustering [2], [3], kernel clustering [4], convex clustering [5], subspace clustering [6]–[12], and recent popular deep learning based clustering [13]–[16]. The commonality of most of them is employing a shallow or deep model to learn a representation and then applying a traditional clustering approach (generally *k*-means) to obtain data partition.

Despite the success of these clustering methods, they have suffered from some limitations as below. They usually focus on representation learning while ignoring the clustering. In the era of deep learning, it is highly desirable to pay more attention to developing new clustering algorithm since deep neural networks have shown promising performance in learning representation. Furthermore, although shallow methods are with high interpretability since almost all of them are statistical models, they may be incapable to handle complex data due to the limited representative capacity. In

contrast, although deep learning based method could be better to capture the highly nonlinear structure hidden into data, they are well known as a "black box". In other words, the state-of-the-art clustering neural networks always lack interpretability, which makes difficulty in performance tuning and understanding its working manner. Moreover, an interpretable clustering neural network will facilitate representation learning in an end-to-end manner.

Motivated by the above observations, this paper proposes a novel neural network (see Fig. 1) from the prospective of differentiable programming (DP) [17]–[23]. The proposed *k*-meansNet is with a novel reformulated objective function of the vanilla *k*-means clustering, which enjoys following advantages. First, the proposed *k*-meansNet is a neural network but with explicit interpretability. In addition, it could also be regarded as a convex version of *k*-means with easier optimization by SGD. Second, our method is provable to monotonically decreasing in the loss under some mild conditions. Third, *k*-means iteratively optimizes the cluster centers and the label assignment with two separate steps, which leads to the incapability of handling new coming data and inferior performance. In contrast, the proposed *k*-meansNet simultaneously learns a set of cluster centers and the label assignments, which could enjoy the capability of handling new data and joint optimization of clustering centers and assignment. Fourth, we experimentally show that *k*-meansNet is robust to different initialization approaches that are highly desired by *k*-means.

The contribution of this work is twofold. One the one hand, from the view of clustering, we reformulate *k*-means as a neural network with a novel objective function. The proposed *k*-meansNet could overcome the drawbacks of *k*-means and neural networks, while enjoying their advantages, *i.e.* robustness to initialization, the capability of handling new coming data, provable convergence, and interpretable neural processing mechanism. On the other hand, to the best of our knowledge, this could be one of the first

- *X. Peng is with College of Computer Science, Sichuan University, Chengdu, 610065, China.*
  E-mail: pengx.gm@gmail.com
- *I. Tsang is with Centre for Artificial Intelligence, University of Technology Sydney, Australia.*
  E-mail: Ivor.Tsang@uts.edu.au
- *J. Zhou is with Institute of High Performance Computing, A*STAR, Singapore.*
  E-mail: joey.tianyi.zhou@gmail.com
- *H. Zhu is with Institute for Infocomm Research, A*STAR, Singapore.*
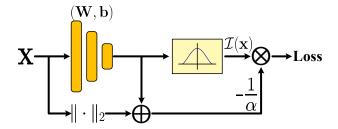 *Email: hongyuanzhu.cn@gmail.com*

Fig. 1. An illustration on the proposed *k*-meansNet which corresponds to the reformulated objective (Eqn.(6–7)). The network is a three layer neural network, namely, the input layer $\mathbf{x}$, the hidden layer $(\mathbf{W}, \mathbf{b})$, and the output layer $\mathcal{I}(\cdot)$. The network is with explicit interpretability in structure derived from the vanilla *k*-means. In brief, the hyperplane $\{\mathbf{W}, \mathbf{b}\}$ is derived from the clustering centers and the activation function normalizes the assignment, which results in interpretable structure. Moreover, we could also understand this model from the attention mechanism as elaborated in Section 3.2.

attempts to make clustering beneficial from differentiable programming. More specifically, *k*-meansNet could perform data clustering, whereas most existing DP works only focus on solving an optimization problem using a neural network. Furthermore, our method is a feedforward neural network (FNN), whereas the models obtained by most existing DP works are recurrent neural networks. Therefore, we believe that this work could be helpful to the community.

2 RELATED WORKS

This work closely relates to clustering, differentiable programming, and interpretability in neural network which are briefly introduced in this section.

2.1 Clustering

As one of most effective data clustering methods, either of shallow and deep subspace clustering approach computes a so-called self-expression $\mathbf{C} \in \mathcal{R}^{n \times n}$ for a given dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ and then applies *k*-means to the representation derived from $\mathbf{C}$.

The main difference among these existing methods lies on the constraint on $\mathbf{C}$. In practice, there are several popular forms, namely, ℓ_0-norm [24], ℓ_1-norm [6], [10], ℓ_2-norm [9], [25], nuclear-norm [7], and their variants [11], [26], [27]. By deeply utilizing $\mathbf{C}$, [15], [16] show that the representation learned by neural network could boost *k*-means performance.

Different from these existing works, we aim to develop a novel neural network to achieve clustering rather than representation learning. More specifically, we recast *k*-means as a neural network, which enjoys the merits of *k*-means and deep learning.

2.2 Differentiable Programming

Differentiable programming (DP) is an emerging and impactful topic, that bridges classical machine learning model and deep neural networks, by emphasizing problem-specific prior and interpretability. DP advocates to build complicated end-to-end machine learning pipeline, by assembling parameterized functional blocks, that are later jointly trained from examples, using some form of differential

calculus–most often stochastic gradient descent (SGD). It bears resemblances to building a software, except that it is parameterized, automatically differentiated, and trainable/optimizable.

To the best of our knowledge, Learned ISTA (LISTA) [17] could be the first well-known DP work in the era of deep learning, which unfolds the ISTA algorithm [28] – a popular ℓ_1-optimizer, as a simple RNN. In the unrolled RNN, the number of layers and the weight correspond to the iteration number and the dictionary, respectively. Inspired by the success of LISTA, numerous methods are proposed to address a variety of problems, *e.g.* image restoration [29], audio processing [18], segmentation [19], hashing [30], and clustering [31].

As discussed in Introduction, this work is remarkably different from most existing differentiable programming approaches in either of network structure (FNN vs. RNN) and applications (clustering vs. optimization), which may serve as a novel angle to facilitate future DP works.

2.3 Interpretability in Neural Network

Generic deep architectures, as often referred to as "black-box" methods, rely on stacking somewhat ad-hoc modules, which makes it prohibitive to interpret their working mechanisms. Despite a few hypotheses and intuitions, it appears difficult to understand why deep models work, how to analyze them, and how they are related to classical machine learning models.

To solve the aforementioned problem, a variety of works [32]–[37] have devoted towards the interpretability in neural network, which usually employ visualization techniques to disentangle the outputs of neural network.

Different from these works, the proposed *k*-meansNet is interpretable in structure which is largely ignored by previous studies. In other words, one could explicitly understand why the structure of the *k*-meansNet is presented as itself, the physical meaning of each unit of the model, as well as why it is able to perform data clustering. Moreover, we do not attempt to interpret an existing neural network like these works did. Instead, we directly develop a novel interpretable neural network.

3 INTERPRETABLE NEURAL NETWORK FOR *k*-MEANS CLUSTERING

In this section, we first show how to reformulate *k*-means as a neural network and then conduct convergence analysis on the proposed *k*-meansNet followed by the discussions on interpretability.

3.1 Formulation

For a given dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, *k*-means aims to partition it into $k \leq n$ different sets $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k\}$ by minimizing the distance of the within-cluster data points. In mathematical,

$$\mathrm{argmin}_{\mathcal{S}} \sum_j \sum_{\mathbf{x} \in \mathcal{S}_j} \|\mathbf{x} - \mathbf{\Omega}_j\|_2^2, \tag{1}$$

where $\mathbf{\Omega}_j$ denotes the j-th cluster center which is computed as the mean of points in $\mathcal{S}_j$, *i.e.*

$$\mathbf{\Omega}_j = \frac{1}{|\mathcal{S}_j|} \sum_{\mathbf{x}_i \in \mathcal{S}_j} \mathbf{x}_i, \qquad (2)$$

where $|\mathcal{S}_j|$ denotes the number of data points in the j-th cluster.

To solve Eq.(1), an EM-like optimization is adopted by updating $\mathcal{I}(\mathbf{x})$ or $\mathbf{\Omega}$ and simultaneously fixing the other one. Such an iterative optimization has several drawbacks. First, the method is sensitive to the initialization, which may achieve an inferior result for a given bad initialized $\mathbf{\Omega}$. In fact, to obtain a stable solution, over-thousands of works have been conducted, including the popular k-means++ [38]. Second, it is an NP-hard problem to finding the optimal solution to k-means in either of general Euclidean space even for bi-cluster and the plane for a general number of k. To solve the NP-hard problem, some variants of k-means are proposed, such as various parametric k-means including Fuzzy c-means [39], [40]. Third, k-means cannot handle the new coming data, which requires the whole dataset is observed.

To overcome these disadvantages, we recast k-means as a novel neural network with a convex objective from the prospective of DP. To the end, we first rewrite Eq.(1) by:

$$\min \sum_{i=1}^n \sum_{j=1}^k \mathcal{I}_j(\mathbf{x}_i) \|\mathbf{x}_i - \mathbf{\Omega}_j\|_2^2, \qquad (3)$$

where $\mathcal{I}_j(\mathbf{x}_i)$ indicates the cluster membership of $\mathbf{x}_i$ *w.r.t.* $\mathbf{\Omega}_j$ and only one entry of $\mathcal{I}_j(\mathbf{x}_i)$ is nonzero. In the following, we will alternatively use $\mathcal{I}_{ij}$ to denote $\mathcal{I}_j(\mathbf{x}_i)$ for simplicity.

The binary constraint on $\mathcal{I}_{ij}$ will lead to a NP-hard problem as the aforementioned. Hence, we relax such a constraint and define $\mathcal{I}_{ij}$ as a probability map based on the distance between $\mathbf{x}_i$ and $\mathbf{\Omega}_j$, *i.e.*

$$\mathcal{I}_j(\mathbf{x}_i) = \frac{\exp(-\alpha \|\mathbf{x}_i - \mathbf{\Omega}_j\|_2^2)}{\sum_j \exp(-\alpha \|\mathbf{x}_i - \mathbf{\Omega}_k\|_2^2)}, \qquad (4)$$

where $\alpha > 0$ is the normalization factor. Note that, here we adopt the *softmax* function. However, other activation functions could also be used as long as $\mathcal{I}_{ij}$ is normalized into the range of $[0.0, 1.0]$. It is worthy to point out that replacing the hard indicator of k-means with the *softmax* function is not the key contribution of our idea. Instead, the major novelty of this work is recasting k-means as a neural network via the following reformulation,

$$-\alpha \|\mathbf{x}_i - \mathbf{\Omega}_j\|_2^2 = -\alpha \|\mathbf{x}_i\|_2^2 + 2\alpha \mathbf{\Omega}_j^\top \mathbf{x}_i - \alpha \|\mathbf{\Omega}_j\|_2^2. \qquad (5)$$

Let $\mathbf{W} = 2\alpha \mathbf{\Omega}$, $\mathbf{b}_j = -\alpha \|\mathbf{\Omega}_j\|_2^2$, and $\alpha \|\mathbf{x}_i\|_2^2 = \beta_i$, the objective function of k-meansNet is formulated as below:

$$\mathcal{L} = \sum_{ij} \mathcal{L}_{ij} = \sum_{ij} \mathcal{I}_{ij}(-\frac{1}{\alpha}\mathbf{W}_j^\top \mathbf{x}_i - \frac{1}{\alpha}\mathbf{b}_j + \frac{1}{\alpha}\beta_i), \quad (6)$$

$$\mathcal{I}_{ij} = \frac{\exp(\mathbf{W}_j^\top \mathbf{x}_i + \mathbf{b}_j)}{\sum_k \exp(\mathbf{W}_k^\top \mathbf{x}_i + \mathbf{b}_k)}, \qquad (7)$$

where $\mathbf{b}_j$ is a scalar which denotes the j-th entry of $\mathbf{b}$ and $\mathbf{\Omega}_j$ denotes the j-th cluster center. Note that, β_i could be dropped out from Eqn.(7), which will not affect the network

training. $\exp(-\beta_j)$ is cancelled out in numerator and denominator of Eqn.(7). Furthermore, $\mathcal{I}_{ij}$ could be regarded as a parametric attention learned from data as elaborated later.

It should be pointed out that, $\mathbf{W}$ and $\mathbf{b}$ will be decoupled during training to favor an additional free degree and avoiding trivial solutions. In other words, after initialization with $\mathbf{\Omega}$, $\mathbf{W}$ and $\mathbf{b}$ are updated independently and the final cluster centers $\mathbf{\Omega}^*$ is recovered via $\mathbf{\Omega}^* = \frac{1}{2\alpha}\mathbf{W}^*$. To show the necessity of decoupling $\mathbf{W}$ and $\mathbf{b}$, similar to the above reformulation process, we rewrite Eqn.(6) as

$$\mathcal{L} = -\frac{1}{\alpha} \sum_i \sum_j \frac{\exp(\mathbf{b}_j + \mathbf{W}_j^\top \mathbf{x}_i - \beta_i)(\mathbf{b}_j + \mathbf{W}_j^\top \mathbf{x}_i - \beta_i)}{\sum_k \exp(\mathbf{b}_k + \mathbf{W}_k^\top \mathbf{x}_i - \beta_i)}$$
$$= -\frac{1}{\alpha} \sum_i \frac{\sum_j \exp(\mathbf{z}_j)\mathbf{z}_j}{\sum_k \exp(\mathbf{z}_k)}$$
$$= -\frac{1}{\alpha} \sum_i f(\mathbf{z}_j), \qquad (8)$$

where $\mathbf{z}_j = (-\frac{\|\mathbf{W}_j\|_2^2}{4\alpha} + \mathbf{W}_j^\top \mathbf{x}_i - \beta_i)$. Thus, we have

$$\max \sum_i \frac{1}{\alpha} f(\mathbf{z}_i)$$
$$\text{s.t.}\, \mathbf{z}_i = -\frac{\|\mathbf{W}_j\|_2^2}{4\alpha} + \mathbf{W}_j^\top \mathbf{x}_i - \beta_i, \|\mathbf{W}_j\|_2 \le 2\alpha\beta, \quad (9)$$

where $\beta = \frac{1}{\alpha} \max(\beta_1, \beta_2, \cdots)$. The inequality constraint is obtained since $\mathbf{\Omega}_j$ must be upper bounded as proved in Section 3.3.

Eqn.(9) is exact Eqn.(4), namely, without decoupling $\mathbf{W}$ and $\mathbf{b}$. Since $f(\mathbf{z})$ obtains its optimum at the boundary with $\mathbf{z}_1 = \mathbf{z}_2 = \cdots$ and $f(\mathbf{z}) = \infty$ when $\mathbf{z} = \infty$, there exists z such that $\mathbf{z}_j = z$ and $f(\mathbf{z})$ obtains its optimal value. We can always find a $\mathbf{W}_j$ and $\mathbf{b}_j$ such that $\mathbf{b}_j + \mathbf{W}_j^\top \mathbf{x} - \beta_j = z$, however, we cannot always find a $\mathbf{W}_j$ and $\mathbf{b}_j$ such that $-\frac{\|\mathbf{W}_j\|_2^2}{4\alpha} + \mathbf{W}_j^\top \mathbf{x}_i - \beta_i = z$. In other words, we have to decouple $\mathbf{W}_j$ and $\mathbf{b}_j$ during training to avoid the trivial solution and speed up the convergence.

3.2 Interpreting Clustering In Neural Processing

The proposed objective function (Eqn.(6–7)) corresponds to a feedforward neural network as demonstrated in Fig. 1. One major attractive of the obtained neural network is that clustering could be explained in neural processing. In other words, we could understand not only why the network structure is built as itself but also why such a network could perform data clustering.

In addition, we could also understand the working manner of k-meansNet from the standpoint of attention mechanism that is popular in natural language processing [41], [42]. As shown in Fig. 2, k-meansNet aims to learn a linear hyperplane (upper pathway) which is spanned by a set of cluster centers $\mathbf{W}$. The hyperplane is able to partition similar data points into the same cluster and dissimilar data points into different clusters via attention. More specifically, to learn the hyperplane, k-meansNet first computes the similarity between the input $\mathbf{x}_i$ and the cluster centers $\mathbf{W}$ using inner product and passes the similarity through an adaptive hinge loss function to obtain the corresponding
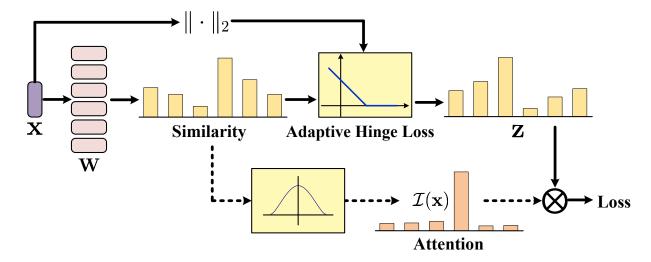
Fig. 2. Understanding k-meansNet from the attention mechanism. The lower pathway (dot line) implements attention on the loss caused by the hyperplane $\{\mathbf{W}, \mathbf{b}\}$ w.r.t. the input $\mathbf{x}$. Note that, let $\mathbf{y}_j = \mathbf{W}_j^\top \mathbf{x}_i + \mathbf{b}_j$, Eqn.(6) could be regarded as passing $\mathbf{y}_j$ through an adaptive hinge loss function that is with the margin β_i because $\beta_i - \mathbf{y}_j$ must be nonnegative as proved in Section 3.3.

dissimilarity $\mathbf{z}_i$. After that, the loss of k-meansNet is computed by summing the weighted $\mathbf{z}_i$ using $\mathcal{I}_i$. Intuitively, this implements a mechanism of attention in the lower pathway which decides the cluster centers to pay attention to. Meanwhile, the attention actually serves as the clustering label.

3.3 Convergence Proofs

In this section, we theoretically show that the loss $\mathcal{L}$ given by our k-meansNet will sufficiently converge to the optimizers.

Without loss of generality, the bias $\mathbf{b}$ could be enveloped into the weight $\mathbf{W}$ via $\mathbf{W}^\top = [\mathbf{W}^\top \ \mathbf{b}]$ and $\mathbf{x}_i = [\mathbf{x}_i^\top \ 1]^\top$. For ease of presentation, let $\mathcal{L}^*$ denote the smallest loss, and $\mathcal{L}_t^*$ be the smallest loss found at the t-step so far. Similarly, $\mathbf{W}^*$ denotes the desirable weight of which the first k columns are the optimal cluster centers $\boldsymbol{\Omega}^*$. We consider the standard SGD to optimize our network, *i.e.*

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla \mathcal{L}(\mathbf{W}_t), \tag{10}$$

where $\nabla \mathcal{L}(\mathbf{W}_t)$ denotes the gradient of $\mathcal{L}$ w.r.t. $\mathbf{W}_t$. In the following, we will alternatively use $\nabla \mathcal{L}(\mathbf{W}_t)$ and $\nabla \mathcal{L}_t$ without causing confusion.

Definition 1 (Lipschitz Continuity). *A function $f(x)$ is a Lipschitz continuous function on the set Ω, if there exists a constant $\epsilon > 0$, $\forall x_1, x_2 \in \Omega$ such that*

$$\|f(x_1) - f(x_2)\| \leq \epsilon \|x_1 - x_2\|, \tag{11}$$

where ϵ is termed as the Lipschitz constant.

Clearly, our objective function $\mathcal{L}$ is a Lipschitz continuous function *i.i.f.* $\|\nabla \mathcal{L}_t\| \leq \epsilon$. In other words, to utilize the Lipschitz Continuity, we need to prove the existence of upper boundary of $\nabla \mathcal{L}_t$.

Theorem 1. *There exists $\epsilon > 0$ such that $\|\nabla \mathcal{L}_t\| \leq \epsilon$, where $\epsilon = 1 + 2 \max(\|\mathbf{z}_i\|)$ and $\mathbf{z}_i = \mathbf{W}_i^\top \mathbf{x}_j$.*

Proof. Without loss of generality, Eqn.(6) could be rewritten in the form of

$$\mathcal{L}(\mathbf{W}) = \sum_j \frac{\sum_i \exp(\mathbf{W}_i^\top \mathbf{x}_j) \mathbf{W}_i^\top \mathbf{x}_j}{\sum_k \exp(\mathbf{W}_k^\top \mathbf{x}_j)}. \tag{12}$$

Let $\mathbf{z}_i = \mathbf{W}_i^\top \mathbf{x}_j$, we have

$$f(\mathbf{z}_i) = \frac{\sum_i \exp(\mathbf{z}_i) \mathbf{z}_i}{\sum_j \exp(\mathbf{z}_j)} = \mathbf{p}_i \mathbf{z}_i, \tag{13}$$

then

$$\begin{aligned}
\nabla_i f(\mathbf{z}_i) &= \frac{(\exp(\mathbf{z}_i) + \exp(\mathbf{z}_i) \mathbf{z}_i) \sum_j \exp(\mathbf{z}_j)}{(\sum_j \exp(\mathbf{z}_j)))^2} \\
&\quad - \frac{\exp(\mathbf{z}_i) \sum_j \exp(\mathbf{z}_j) \mathbf{z}_j}{(\sum_j \exp(\mathbf{z}_j)))^2} \\
&= \mathbf{p}_i + \mathbf{p}_i \mathbf{z}_i - \mathbf{p}_i \mathbf{p}_j \mathbf{z}_j. \tag{14}
\end{aligned}$$

As $0 \leq \|\mathbf{p}_i\| \leq 1$, we could further have

$$\begin{aligned}
\|\nabla_i f(\mathbf{z}_i)\| &\leq \|\mathbf{p}_i\|(1 + \|\mathbf{z}_i\| + \mathbf{p}_j \|\mathbf{z}_j\|) \\
&\leq 1 + \|\mathbf{z}_i\| + \|\mathbf{z}_j\|. \tag{15}
\end{aligned}$$

Clearly our objective function $\mathcal{L}(\mathbf{W})$ will be upper bounded by a positive real number ϵ when $\|\mathbf{z}_i\|$ is bounded (see Fig. 3 for an illustrative example). In fact, there exists the upper boundary of $\|\mathbf{z}_i\|$ for informative dataset. More specifically, without loss of generality, one could enforce $\|\mathbf{x}_i\| = 1$ and thus $\|\mathbf{W}\| \leq 2\alpha$ induced by $\mathbf{W}_i = 2\alpha \boldsymbol{\Omega}_i = \frac{2\alpha}{|\mathcal{S}_i|} \sum_{\mathbf{x}_j \in \mathcal{S}_i} \mathbf{x}_i$, where $|\mathcal{S}_i|$ denotes the size of the i-th cluster. $\qquad \square$

Theorem 1 tells us that the proposed objective function $\mathcal{L}(\mathbf{W})$ will be upper bounded by a positive real number ϵ when $\|\mathbf{z}_i\|$ is bounded. In fact, there exists the upper boundary of $\|\mathbf{z}_i\|$ for any real-world dataset that carries information. Furthermore, without loss of generality, one could normalize $\mathbf{x}_i$ by $\|\mathbf{x}_i\| = 1$ and thus $\|\mathbf{W}\| \leq 2\alpha$ induced by $\mathbf{W}_i = 2\alpha \boldsymbol{\Omega}_i = \frac{2\alpha}{|\mathcal{S}_i|} \sum_{\mathbf{x}_j \in \mathcal{S}_i} \mathbf{x}_i$, where $|\mathcal{S}_i|$ denotes the size of the i-th cluster.
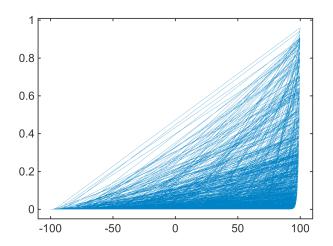
Fig. 3. A toy example to show the boundness of our loss function in 1-dimensional case. x-axis denotes the data points ($\mathbf{z}$) randomly sampled from -100 to 100, and y-axis denotes the corresponding loss $\frac{\exp(\mathbf{z})\mathbf{z}}{\sum_i \exp(\mathbf{z})}$. One could see that our loss function will be bounded if $\mathbf{z}$ is bounded.

Based on Theorem 1, we could have following convergence result by following [43].

Theorem 2. *One could always find an optimal model $\mathcal{L}_T^*$ which is sufficiently close to the desired $\mathcal{L}^*$ after T steps, i.e.*

$$\mathcal{L}_T^* - \mathcal{L}^* \leq \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + \epsilon^2 \sum_t^T \eta_t^2}{2 \sum_{t=1}^T \eta_t} \quad (16)$$

Proof. Let $\mathbf{W}^* = 2\alpha\mathbf{\Omega}^*$ be the minimizer to our objective function (*i.e.* Eqn.(6)), then

$$\|\mathbf{W}_{T+1} - \mathbf{W}^*\|_F^2 = \|\mathbf{W}_T - \mathbf{W}^*\|_F^2 \\ - 2tr(\eta_T \nabla\mathcal{L}_T^\top(\mathbf{W}_T - \mathbf{W}^*)) + \eta_t^2\|\nabla\mathcal{L}_T\|_F^2, \quad (17)$$

where $tr(\cdot)$ denotes the trace of a matrix.

Applying the above Equation recursively, it gives that

$$\|\mathbf{W}_{T+1} - \mathbf{W}^*\|_F^2 = \|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 - 2\sum_{t=1}^T \eta_t tr(\mathbf{W}_t - \mathbf{W}^*) \\ + \sum_{t=1}^T \eta_t^2\|\nabla\mathcal{L}_t\|_F^2. \quad (18)$$

As $\mathcal{L}(\mathbf{W})$ satisfies the Lipschitz Continuity and according to the definition of gradient, *i.e.*

$$f(x^*) \geq f(x_t) + \nabla\mathcal{L}_t^\top(x^* - x_t) \quad (19)$$

then,

$$\|\mathbf{W}_{T+1} - \mathbf{W}^*\|_F^2 \leq \|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 \\ - 2\sum_{t=1}^T \eta_t(\mathcal{L}_t - \mathcal{L}^*) + \epsilon^2\sum_{t=1}^T \eta_t^2. \quad (20)$$

Clearly,

$$2\sum_{t=1}^T \eta_t(\mathcal{L}_t - \mathcal{L}^*) \leq \|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + \epsilon^2\sum_{t=1}^T \eta_t^2. \quad (21)$$

Since

$$\mathcal{L}_t - \mathcal{L}^* \geq \min_{t=1,2,\cdots,T}(\mathcal{L}_t - \mathcal{L}^*) = \mathcal{L}_T^* - \mathcal{L}^*, \quad (22)$$

where $\mathcal{L}_T^*$ is the best $\mathcal{L}$ found within T steps so far.

Combining Eqn.(21) and Eqn.(22), it gives that

$$\mathcal{L}_T^* - \mathcal{L}^* \leq \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + \epsilon^2 \sum_{t=1}^T \eta_t^2}{2\sum_{t=1}^T \eta_t} \quad (23)$$

as desired. $\square$

Based on Theorem 2, the following two lemmas could be derived.

Lemma 1. *For the fixed step size (i.e. $\eta_t = \eta$) and $T \to \infty$,*

$$\mathcal{L}_T^* - \mathcal{L}^* \to \frac{\eta\epsilon^2}{2} \quad (24)$$

Proof. After T steps, we have

$$\mathcal{L}_T^* - \mathcal{L}^* \leq \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + T\epsilon^2\eta^2}{2T\eta} \\ = \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2/(T\eta) + \eta\epsilon^2}{2} \quad (25)$$

as desired. $\square$

Lemma 2. *For the fixed step length (i.e. $\eta_t = \eta/\nabla\mathcal{L}_t$) and $T \to \infty$,*

$$\mathcal{L}_T^* - \mathcal{L}^* \to \frac{\eta\epsilon}{2} \quad (26)$$

Proof. Similar to the proof for Lemma 1. $\square$

Lemmas 1–2 show that the loss will sufficiently converge to $\mathcal{L}^*$ with a radius of $\frac{\eta\epsilon^2}{2}$ and $\frac{\eta\epsilon}{2}$ within T steps.

4 EXPERIMENTAL RESULTS

In this section, we carry out experiments to verify the effectiveness of the proposed k-meansNet comparing with 12 state-of-the-art clustering approaches.

4.1 Experimental Settings

We conduct experiments on a Nvidia K40 GPU, an Intel Xeon CPU 2.40GHz, and a 64GB memory. For all the tested baselines, we use the source code released by respective authors. Regarding our method, we implement it in TensorFlow.

Baselines: We compare our method with 1) popular subspace clustering approaches including spectral clustering (SC) [2], LRR [7], and LSR [9]; 2) large-scale clustering methods including Scalable LRR (SLRR) [44] and large-scale spectral clustering (LSC) [45]; 3) matrix decomposition based method and agglomerative clustering, *i.e.* NMF [46] and Zeta function based agglomerative clustering (ZAC) [47]; and 4) state-of-the-art deep embedding clustering (DEC) [14]. Moreover, we also use the vanilla k-means and FCM [40] as baselines. Note that, either of LSR and LSC has two variants, which are denoted by LSR1, LSR2, LSC-R, and LSC-K.

In experiments, we employ a denoising auto-encoder [48] to extract low-dimensional features for all the investigated clustering approaches. To be specific, as did in DEC [14], the auto-encoder consists of an encoder with the fully connected layers (FCL) of m-(500)-(500)-(2000)-(10) and a decoder with the fully connected layers of (2000)-(500)-(500)-(m), where Z in (Z) represents fully connected layer

with Z number of neurons and m is the input dimension. We adopt the *ReLu* as the activation function for all layers except the last one of encoder and decoder which is with the *sigmoid* function. To alleviate the overfitting, a dropout layer with the rate of 0.2 is added after each layer of the encoder. The adadelta [49] optimizer is used to train the neural networks including the auto-encoder and k-meansNet. Either of the max training epoch and convergence error is satisfied, the neural network is regarded as convergent. In all experiments, these two parameters are fixed to 3000 and 10^{-3}. Furthermore, for fair comparisons, we tune parameters for all tested methods and report their best performance. For the baselines, we follow the parameter tuning method suggested in the original work. For our method, we only tune α ranged into $\{1, 5\} \times 10^p$, where p increases from -5 to 0. Note that, some other recent deep features extractors [13], [50] could also be used to improve the performance of k-meansNet. However, our experimental result will show that with the simple auto-encoder our method could perform competitive with these well-established networks.

Datasets: Our method is evaluated using following datasets, including the mnist handwritten digital full database [51], the CIFAR10 image full database [52], and the CIFAR100 full database [52]. For CIFAR100, we adopt superclass partitions. In other words, we conduct experiments on 20 supersets of CIFAR100 and report the mean, the median, and the maximum of result over these subsets. No preprocessing steps are conducted expected normalizing each data point to have a unit 1 of ℓ_2-norm.

Evaluation Metrics: Three metrics are used to evaluate the clustering performance, *i.e.* Accuracy or called Purity, normalized mutual information (NMI), and adjusted rand index (ARI). A higher value of these metrics indicates better clustering performance.

4.2 Experimental Comparisons

In this section, we examine the performance of k-meansNet on three challenging datasets. As all tested methods excepted ZAC directly/indirectly involve initialization of cluster centers, we adopt k-means++ as the initializer for fair comparisons. For our method, k-means++ first initialize Ω which is further used to compute $\{\mathbf{W}, \mathbf{b}\}$.

The mnist consists of 70,000 images distributed over 10 handwritten digits and each image is with the size of 28×28. The CIFAR10 dataset consists of 60,000 images sampled from 10 classes and each image is with the size of $32 \times 32 \times 3$. Table 1 demonstrates the results of the evaluated approaches on these two datasets. One could observe that k-meansNet is superior to all baselines in terms of three metrics and DEC achieves the second best result. For example, our method outperforms the second best approach 4.11%, 5.10%, and 6.61% on mnist *w.r.t.* Accuracy, NMI, and ARI thanks to the new reformulation and neural network based implementation. Note that, LRR and SLRR show inferior performance in experiments, and the possible reason is that these datasets do not satisfy the low-rank assumption well. In the following section, we will show that k-meansNet will further improve the state of the art if other feature extractors such as CNN are used.

Like CIFAR-10, the CIFAR100 dataset also includes 60,000 images. The difference between them is that CI-

FAR100 includes 100 subjects which could be further grouped into 20 superclasses each of which consists of 3000 samples. Noted that, CIFAR10 and CIFAR100 are more challenging than the mnist dataset, which are less investigated in prior clustering works.

Tables 2–3 show the clustering result of our method on 20 subsets of CIFAR100. Again, the proposed k-meansNet shows promising results on the datasets, which is 1.88% and 1.85% at least higher than the other methods in terms of mean Accuracy on the first and last 10 subsets. Comparing with the recently proposed DEC, our method earns a performance gain of 3.79% and 4.98% on the first and last 10 subsets, respectively. Moreover, k-meansNet remarkably outperforms k-means and FCM by a considerable margin in mean Accuracy. More specifically, the gains over k-means are 2.05% and 1.87%, and that over FCM are 6.07% and 4.31%.

4.3 Influence of Parameters

In this section, we investigate the influence of the parameter α on the mnist dataset. In experiment, we increase the value of α from 10^{-5} to 0.5 as shown in Fig. 5. From the result, one could find that the proposed k-meansNet achieves stable clustering result in general. The Accuracy, NMI, and ARI usually change around 85%, 77%, and 75%.

4.4 Influence of Initializations, Features, and Optimizers

As discussed in Introduction, the center-based clustering method is sensitive to the initial clustering centers. In this section, we examine the influence of three different initialization methods, namely, k-means++, k-means, and random method. Besides the performance with fully connected auto-encoder as aforementioned, we also investigate the performance of our method by collaborating with a convolutional auto-encoder. More specifically, the used convolutional encoder is a six-layer network which is with conv(64,5)-pool(2)-conv(32,5)-pool(2)-FCL(1024)-FCL(10), where "conv (64,5)" denotes a convolutional layer with the filter size of 64 and the kernel size of 5, "pool(5)" denotes max-pooling operation with the kernel size of 2, and "FCL(1024)" is a fully connected layer with 1024 neurons. The decoder is symmetric to the encoder. Similar to fully connected auto-encoder, *ReLu* is used as the activation function for all layers except the last one of encoder and decoder which adopts the *sigmoid* function. The experiments are conducted on mnist and k-means is used as the baseline. Furthermore, α of k-meansNet is fixed to 10^{-2} for the convolutional case and 10^{-3} for fully connected case.

From Table 5, we have following observations. First, the proposed k-meansNet is robust to the choice of initialization method. In the case of FCN, it almost keeps unchanged. For CNN, the difference between the maximal and minimal Accuracy is about 2%, and this gaps in NMI and ARI are 1.7% and 2.98%. Second, our method benefits much more from CNN than k-means. For example, our method improves the Accuracy by 3.46% versus 1.66% given by k-means.

Besides the above investigation on different initializations and features, we also consider the role of the used

TABLE 1
Clustering results on the **mnist** and the **CIFAR10** full dataset. The number in bold indicates the best result.

Methods	mnist				CIFAR10			
	Accuracy	NMI	ARI	Parameter	Accuracy	NMI	ARI	Parameter
k-means	78.32	77.75	70.53	-	19.81	5.94	3.01	-
FCM	21.56	12.39	5.10	-	17.02	3.92	2.56	-
SC	71.28	73.18	62.18	1	19.81	4.72	3.22	10
LRR	21.07	10.43	10.03	10.01	13.07	0.43	0.03	0.01
LSR1	40.42	31.51	21.35	0.4	19.79	6.05	3.64	0.6
LSR2	41.43	30.03	20.00	0.1	19.08	6.37	3.16	0.5
SLRR	21.75	7.57	5.55	2.1	13.09	1.31	0.94	0.1
LSC-R	59.64	56.68	45.98	6	18.39	5.67	2.58	3
LSC-K	72.07	69.88	60.81	6	19.29	6.34	3.89	3
NMF	46.35	43.58	31.20	10	19.68	6.20	3.21	3
ZAC	60.00	65.47	54.07	20	5.24	0.36	0.00	10
DEC	83.65	73.60	70.10	10	18.09	4.56	2.47	80
k-meansNet	**87.76**	**78.70**	**76.71**	1.00E-03	**20.23**	**6.87**	**3.95**	0.5

TABLE 2
Clustering **Accuracy** (%) on the first 10 supersets of the **CIFAR100** dataset.

Methods	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	max	mean	median
k-means	29.63	43.30	31.53	30.03	34.83	30.43	33.60	38.80	28.93	30.70	43.30	33.18	31.12
FCM	26.77	37.80	25.30	25.97	29.77	26.37	32.60	36.73	25.00	25.33	37.80	29.16	26.57
SC	31.90	39.30	33.67	27.53	34.27	27.77	33.10	36.17	26.90	32.30	39.30	32.29	32.70
LRR	21.77	21.73	21.37	20.13	21.60	21.80	21.53	21.27	21.90	21.50	21.90	21.46	21.57
LSR1	21.93	21.40	22.27	21.87	21.47	21.30	22.33	21.97	21.07	21.90	22.33	21.75	21.89
LSR2	22.93	22.67	22.87	23.80	24.10	21.83	22.07	25.30	21.77	22.10	25.30	22.94	22.77
SLRR	22.40	22.27	21.77	21.73	22.50	22.63	22.53	22.57	22.40	22.50	22.63	22.33	22.45
LSC-R	31.97	40.50	30.77	28.87	34.30	28.67	32.90	35.27	27.13	32.03	40.50	32.24	32.00
LSC-K	32.36	39.97	34.30	30.93	34.37	30.07	32.80	37.87	28.23	32.60	39.97	33.35	32.70
NMF-LP	31.30	43.93	33.40	30.57	34.87	30.93	31.03	34.33	29.47	32.23	43.93	33.21	31.77
ZAC	20.13	20.33	20.20	20.27	20.40	20.23	20.30	20.33	20.43	20.20	20.43	20.28	20.29
DEC	31.17	43.97	29.97	30.60	34.87	28.50	33.40	20.07	29.87	31.97	43.97	31.44	30.89
k-meansNet	**33.00**	**45.00**	**35.53**	**31.27**	**35.33**	**31.80**	**36.43**	**40.63**	**29.53**	**33.77**	**45.00**	**35.23**	**34.55**

TABLE 3
Clustering **Accuracy** (%) on the last 10 supersets of the **CIFAR100** dataset.

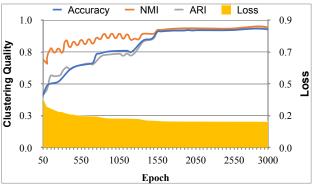| Methods	s11	s12	s13	s14	s5	s16	s17	s18	s19	s20	max	mean	median
k-means	39.53	28.37	25.23	26.87	24.10	31.83	27.50	32.83	31.00	39.80	39.80	30.71	29.69
FCM	41.80	29.33	23.77	26.77	23.30	29.40	27.43	23.27	25.70	32.97	41.80	28.37	27.10
SC	40.77	31.80	24.83	26.33	23.97	31.03	30.57	30.97	28.50	39.30	40.77	30.81	30.77
LRR	21.87	21.67	21.97	21.67	20.37	21.27	21.77	22.00	21.20	21.47	22.00	21.53	21.67
LSR1	21.80	21.93	21.57	21.30	22.10	22.27	22.00	21.70	21.60	21.90	22.27	21.82	21.85
LSR2	26.43	22.13	20.30	24.10	22.00	21.97	21.47	21.07	21.17	24.60	26.43	22.52	21.99
SLRR	22.63	22.50	22.03	22.90	22.27	21.57	21.47	22.77	23.30	22.37	23.30	22.38	22.44
LSC-R	41.53	30.87	24.47	26.43	23.70	20.97	29.10	31.97	28.63	32.93	41.53	29.06	28.87
LSC-K	43.90	30.67	24.67	26.57	24.10	29.10	30.77	30.37	29.57	38.60	43.90	30.83	29.97
NMF-LP	42.00	30.27	25.00	25.33	22.83	30.33	29.13	32.13	29.13	40.97	42.00	30.71	29.70
ZAC	20.20	20.23	20.30	20.27	20.23	20.30	20.30	20.27	20.23	20.23	20.30	20.26	20.25
DEC	21.80	20.17	25.03	26.90	23.80	31.83	27.07	28.57	30.63	41.17	41.17	27.70	26.99
k-meansNet	**44.20**	**32.23**	**25.87**	**27.50**	**24.80**	**33.60**	**31.33**	**34.10**	**30.00**	**43.17**	**44.20**	**32.68**	**31.78**

optimization approaches. In our experiments, we carry out experiments on the mnist dataset with the aforementioned CNN network and emply four popular SGD variants, namely, adadelta [49], adagrad [53], adam [54], and RMSprop [55], to train k-meansNet. In the evaluation, we adopt the default setting for these optimizers and experimentally set α to 10^{-2} for adadelta and adam, and to 10^{-3} for adagrad and RMSprop. For a more comprehensive study,

we further adopt four metrics for evaluating the clustering quality, *i.e.* adjusted mutual index (AMI), Homogeneity, Completeness, and v_measure. Noticed that, Accuracy, AMI, ARI, and AMI are external metrics which are computed based on the ground-truth, whereas Homogeneity, Completeness, and v_measure are internal metrics which measure the compactness/scatterness of within-/between-clusters. From TABLE 4, it is easy to observe that AdaDelta

TABLE 4
Influence of different optimizers on the **mnist** full dataset.

Optimizers	Accuracy	NMI	ARI	AMI	Homogeneity	Completeness	v_measure
AdaDelta	93.11	85.52	85.54	85.47	85.48	85.55	85.52
Adagrad	61.09	71.55	56.11	65.03	65.04	78.71	71.22
Adam	92.13	84.23	83.72	84.18	84.18	84.27	84.23
RMSprop	80.08	79.69	73.38	79.03	79.04	80.35	79.69

(a) k-meansNet with FCN.

(b) k-meansNet with CNN.

Fig. 4. Performance v.s. Training Epoch on the mnist dataset. The left and right y-axis denote the clustering result and the loss, respectively.

TABLE 5
Influence of different initialization methods and different features on the **mnist** full dataset.

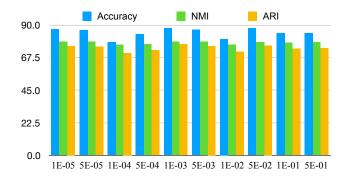| Methods	Initializations	Raw Data			FCN			CNN		
		ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
k-means	random	54.23	48.49	36.50	77.31	76.71	69.48	78.98	77.45	72.16
	k-means++	53.24	49.98	36.52	78.32	77.75	70.53	79.98	79.46	73.16
k-meansNet	random	57.31	51.14	38.89	87.72	78.26	75.91	**93.20**	**85.51**	**85.66**
	k-means++	57.55	50.31	38.90	**87.76**	**78.70**	**76.21**	91.26	83.81	82.68
	k-means	**58.08**	**51.21**	**39.66**	87.56	78.70	76.06	92.42	84.70	84.27

Fig. 5. Influence of the parameter α.

and Adam achieve the best performance in terms the used metrics, and Adagrad performs the worst. Such a result is consistent with the experimental results of prior works.

4.5 Convergence Analysis in Experiments

In Section 3.2, we have theoretically shown that our method will sufficiently approximate to the global optimum under some mild conditions. In this Section, we conduct experiments on the mnist dataset to verify our theoretical result.

In Fig. 4, we report the clustering results and the loss value of our method with the fully connected neural network and the convolutional neural network which are presented in Section 4.4. From the result, ones could observe that k-meansNet achieves convergence after ∼1700 epochs in terms of Accuracy, NMI, ARI, and the loss value. Considering the difference in k-meansNet with FCN and with CNN, the former gives slightly changed results after 1650 epochs, whereas the latter keeps stability after 1550. For example, the Accuracy of k-meansNet with FCN ranges from 84.37% to 88.09% after 1650 epochs, and the range of k-meansNet with CNN is $[91.04\%, 93.96\%]$ after 1550 epochs.

5 CONCLUSION

In this paper, we proposed an interpretable neural network which is derived from the vanilla k-means clustering algorithm from the prospective of differentiable programming. Besides proposing a clustering neural network to overcome some disadvantages of k-means, we contribute to differentiable programming in following two aspects. On the one hand, existing works in differentiable programming employ neural networks as an alternative optimization method, whereas we use neural networks to perform data clustering.

On the other hand, the pioneer works in differentiable programming obtained a recurrent neural network, whereas our obtain a feedforward neural network. Moreover, we bridge the attention mechanism with our method to understand its working manner.

ACKNOWLEDGMENT

REFERENCES

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput Surv*, vol. 31, no. 3, pp. 264–323, Sep. 1999. 1

[2] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. of 14th Adv. in Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2001, pp. 849–856. 1, 5

[3] F. Nie, Z. Zeng, T. I. W., D. Xu, and C. Zhang, "Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering," *IEEE Trans. Neural. Netw.*, vol. 22, no. 11, pp. 1796–1808, 2011. 1

[4] V. Patel and R. Vidal, "Kernel sparse subspace clustering," in *Proc. of IEEE Int. Conf. on Image Process.*, Paris, Oct. 2014, pp. 2849–2853. 1

[5] T. Hocking, J.-P. Vert, F. R. Bach, and A. Joulin, "Clusterpath - An Algorithm for Clustering using Convex Fusion Penalties." in *Proc Int Conf Mach Learn*, Washington, USA, Jul 2011, pp. 745–752. 1

[6] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, 2013. 1, 2

[7] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, 2013. 1, 2, 5

[8] R. Liu, Z. Lin, F. D. la Torre, and Z. Su, "Fixed-rank representation for unsupervised visual learning," in *Proc. of 25th IEEE Conf. Comput. Vis. and Pattern Recognit.*, Providence, RI, Jun. 2012, pp. 598–605. 1

[9] C. Lu, H. Min, Z. Zhao, L. Zhu, D. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *Proc. of 12th Eur. Conf. Comput. Vis.*, Florence, Italy, Oct. 2012, pp. 347–360. 1, 2, 5

[10] Y.-X. Wang and H. Xu, "Noisy sparse subspace clustering," *J Mach Learn Res*, vol. 17, no. 12, pp. 1–41, 2016. [Online]. Available: http://jmlr.org/papers/v17/13-354.html 1, 2

[11] C. You, C. G. Li, D. P. Robinson, and R. Vidal, "Oracle based active set algorithm for scalable elastic net subspace clustering," in *Proc of 29th IEEE Conf Comput Vis and Pattern Recognit*, Las Vegas, NV, Jun. 2016, pp. 3928–3937. 1, 2

[12] Q. W. M. Chen and X. Li, "Robust adaptive sparse learning method for graph clustering," in *Proc of IEEE Int Conf Image Process*. 1

[13] J. Yang, D. Parikh, and D. Batra, "Joint Unsupervised Learning of Deep Representations and Image Clusters," in *Proc of 29th IEEE Conf Comput. Vis and Pattern Recognit*. Las Vegas, NV: IEEE, Jun 2016, pp. 5147–5156. 1, 6

[14] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc of 33th Int Conf Mach Learn*, New York, Jun. 2016. 1, 5

[15] X. Peng, S. Xiao, J. Feng, W. Yau, and Z. Yi, "Deep subspace clustering with sparsity prior," in *Proc of 25th Int Joint Conf Artif Intell*, New York, NY, USA, Jul. 2016, pp. 1925–1931. 1, 2

[16] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," in *Proc of 29th Adv in Neural Inf Process Syst*, Montral, Canada, Dec. 2017. 1, 2

[17] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc of 27th Int Conf Mach Learn*, USA, 2010, pp. 399–406. 1, 2

[18] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," *IEEE Trans Pattern Anal and Machine Intelli*, vol. 37, no. 9, pp. 1821–1833, Sep. 2015. 1, 2

[19] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," in *Proc. of 21th Int Conf Comput Vis*, Santiago, Chile, Dec 2015, pp. 1529–1537. 1, 2

[20] R. Liu, G. Zhong, J. Cao, Z. Lin, S. Shan, and Z. Luo, "Learning to diffuse: A new perspective to design pdes for visual analysis," *IEEE Trans Pattern Anal Mach Intell*, vol. 38, no. 12, pp. 2457–2471, Dec 2016. 1

[21] X. Chen, J. Liu, Z. Wang, and W. Yin, "Theoretical Linear Convergence of Unfolded ISTA and Its Practical Weights and Thresholds," in *Proc. of Adv. in Neural Inf. Process. Syst.*, Vancouver, CA, Dec. 2018, pp. 9079–9089. 1

[22] J. Liu, X. Chen, Z. Wang, and W. Yin, "ALISTA: Analytic Weights Are As Good As Learned Weights in LISTA," in *Proc Int Conf Learn Rep*, New Orleans, May 2019. 1

[23] Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-Net: Learning PDEs from data," in *Proc 35th Int Conf Machine Learn*, Jan. 2018, pp. 5067–5078. 1

[24] Y. Yang, J. Feng, N. Jojic, J. Yang, and T. S. Huang, "Subspace learning by l0-induced sparsity," *Int J of Computer Vis*, Jul 2018. [Online]. Available: https://doi.org/10.1007/s11263-018-1092-4 2

[25] X. Peng, Z. Yu, Z. Yi, and H. Tang, "Constructing the l2-graph for robust subspace learning and subspace clustering," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1053–1066, Apr. 2017. 2

[26] P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *Proc. of 24th IEEE Conf. Comput. Vis. and Pattern Recognit.*, Colorado Springs, CO, Jun. 2011, pp. 1801–1807. 2

[27] H. Hu, Z. Lin, J. Feng, and J. Zhou, "Smooth representation clustering," in *Proc. of 27th IEEE Conf. Comput. Vis. and Pattern Recognit.*, Columbus, OH, Jun. 2014, pp. 3834–3841. 2

[28] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, 2008. 2

[29] Y. Chen, W. Yu, and T. Pock, "On learning optimized reaction diffusion processes for effective image restoration," in *Proc. of 28th IEEE Conf Comput Vis and Pattern Recognit*, Boston, MA, Jun. 2015, pp. 5261–5269. 2

[30] Q. Liu, G. Liu, L. Li, X. Yuan, M. Wang, and W. Liu, "Reversed spectral hashing," *IEEE Trans Neur Netw Learn Syst*, vol. 29, no. 6, pp. 2441–2449, June 2018. 2

[31] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang, "Learning a task-specific deep architecture for clustering," in *Proc of SIAM Int Conf Data Mining*, Miami, Florida, May 2015, pp. 369–377. 2

[32] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks." in *Proc Euro Conf Computer Vis*. Cham: Springer International Publishing, 2014, pp. 818–833. 2

[33] P. W. Koh and P. Liang, "Understanding Black-box Predictions via Influence Functions." in *Proc of Int Conf Machine Learn*, 2017. 2

[34] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network Dissection - Quantifying Interpretability of Deep Visual Representations." in *Proc of IEEE Conf Comput Vis and Pattern Recognit*. IEEE, 2017, pp. 3319–3327. 2

[35] A. Dosovitskiy and T. Brox, "Inverting Visual Representations with Convolutional Networks." in *Proc of IEEE Conf Comput Vis and Pattern Recognit*. IEEE, 2016, pp. 4829–4837. 2

[36] Q. Zhang, Y. N. Wu, and S.-C. Zhu, "Interpretable Convolutional Neural Networks." in *Proc of IEEE Conf Comput Vis and Pattern Recognit*, 2018. 2

[37] B. Kim, O. Koyejo, and R. Khanna, "Examples are not enough, learn to criticize! Criticism for Interpretability." in *Proc of Adv. in Neural Inf Process Syst*, 2016, pp. 2288–2296. 2

[38] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. of 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, PA, USA, 2007, pp. 1027–1035. 3

[39] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973. 3

[40] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981. 3, 5

[41] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate." *CoRR*, vol. cs.CL, 2014. 3

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need." in *Proc of Adv. in Neural Inf Process Syst*, 2017. 3

[43] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, pp. 2004–2005, 2003. 5

[44] X. Peng, H. Tang, L. Zhang, Z. Yi, and S. Xiao, "A unified framework for representation-based subspace clustering of out-of-sample and large-scale data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–14, 2015. 5

[45] D. Cai and X. Chen, "Large scale spectral clustering via landmark-based sparse representation," *IEEE Trans. on Cybern.*, vol. 45, no. 8, pp. 1669–1680, Aug 2015. 5

[46] D. Cai, X. He, and J. Han, "Locally consistent concept factorization for document clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 902–913, Jun. 2011. 5

[47] D. Zhao and X. Tang, "Cyclizing clusters via zeta function of a graph," in *Proc. of 21th Adv. in Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2009, pp. 1953–1960. 5

[48] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders - Learning Useful Representations in a Deep Network with a Local Denoising Criterion." *J Mach Learn Res*, vol. 11, no. Dec, pp. 3371–3408, 2010. 5

[49] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701 6, 7

[50] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised generative approach to Clustering," in *Proc of 26th Int Joint Conf Artif Intell.* Beijing Institute of Technology, Beijing, China, Jan 2017, pp. 1965–1972. 6

[51] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. 6

[52] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009. 6

[53] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011. 7

[54] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of 3th Int. Conf. Learn Rep.*, 2015, pp. 1–15. 7

[55] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012. 7