Journal of Machine Learning Research 23 (2022) 1-28 Submitted 6/19; Revised 9/21; Published 2/22

XAI Beyond Classification: Interpretable Neural Clustering

Xi Peng[1] PENGX.GM@GMAIL.COM
Yunfan Li[1] YUNFANLI.GM@GMAIL.COM
Ivor W. Tsang[2,3] IVOR.TSANG@GMAIL.COM
Hongyuan Zhu[4] ZHUH@I2R.A-STAR.EDU.SG
Jiancheng Lv[1*] LVJIANCHENG@SCU.EDU.CN
Joey Tianyi Zhou[3,5] JOEY.TIANYI.ZHOU@GMAIL.COM
[1]*College of Computer Science, Sichuan University, Chengdu, China.*
[2]*Centre for Frontier Artificial Intelligence Research, A*STAR, Singapore.*
[3]*Australian Artificial Intelligence Institute, University of Technology, Sydney Australia.*
[4]*Institute for Infocomm Research, A*STAR, Singapore.*
[5]*Institute of High Performance Computing, A*STAR, Singapore.*

Editor: David Blei

Abstract

In this paper, we study two challenging problems in explainable AI (XAI) and data clustering. The first is how to directly design a neural network with inherent interpretability, rather than giving post-hoc explanations of a black-box model. The second is implementing discrete k-means with a differentiable neural network that embraces the advantages of parallel computing, online clustering, and clustering-favorable representation learning. To address these two challenges, we design a novel neural network, which is a differentiable reformulation of the vanilla k-means, called inTerpretable nEuraL cLustering (TELL). Our contributions are threefold. First, to the best of our knowledge, most existing XAI works focus on supervised learning paradigms. This work is one of the few XAI studies on unsupervised learning, in particular, data clustering. Second, TELL is an interpretable, or the so-called intrinsically explainable and transparent model. In contrast, most existing XAI studies resort to various means for understanding a black-box model with post-hoc explanations. Third, from the view of data clustering, TELL possesses many properties highly desired by k-means, including but not limited to online clustering, plug-and-play module, parallel computing, and provable convergence. Extensive experiments show that our method achieves superior performance comparing with 14 clustering approaches on three challenging data sets. The source code could be accessed at `www.pengxi.me`.

Keywords: transparent neural networks, stochastic k-means clustering, differentiable programming.

1. Introduction

As a fundamental topic in machine learning, clustering aims to group similar samples into the same cluster and separate dissimilar ones into different clusters. During the past decade, a variety of

*. Corresponding author.

clustering methods (Jain et al., 1999) have been proposed and achieved encouraging success in various applications. In recent, the main focus of the community shifts to how to handle high-dimensional data that is usually linear inseparable.

To effectively cluster high-dimensional data, many kinds of methods have been proposed, *e.g.*, spectral clustering (Ng et al., 2001), kernel clustering (Wang et al., 2019), convex clustering (Hocking et al., 2011; Yi et al., 2013), subspace clustering (Elhamifar and Vidal, 2013; Liu et al., 2013; Peng et al., 2017), and the recent popular deep clustering (Yang et al., 2016; Peng et al., 2016; Ji et al., 2017; Peng et al., 2020; Li et al., 2021). The aforementioned methods share a common clustering paradigm of first learning a shallow or deep representation and then applying a traditional clustering method (k-means in most cases) to make cluster assignments.

Though promising results have been achieved on many applications, these methods still suffer from the following limitations. Namely, though grounded in theory, traditional approaches such as subspace clustering might be incapable of handling more complex data due to its limited representability. On the contrary, although deep clustering methods could capture the hidden nonlinear structure of data, as a "black box" model, their lack of explainability makes its working mechanism hard to understand. Consequently, unguided and laborious hyper-parameter tuning is usually required to achieve satisfying results.

In this paper, we propose a novel neural network (illustrated in Fig. 1) from the perspective of differentiable programming (DP) and learning-based optimization (Gregor and LeCun, 2010; Sprechmann et al., 2015; Zheng et al., 2015; Liu et al., 2016; Chen et al., 2018; Liu et al., 2019; Long et al., 2018). The proposed inTerpretable nEuraL cLustering (TELL) is a differentiable alternative of the vanilla k-means, which reformulates the k-means objective as a neural layer. As a differentiable reformulation, TELL equips the vanilla k-means with advantages of neural networks, including end-to-end optimization, pluggability, provable convergence, and interpretable working mechanism. It could achieve clustering for large-scale and online data, which is impractical for the vanilla k-means.

The contribution and novelty of this work are summarized as follows:

- From the view of XAI, our contribution is twofold. On the one hand, we directly build an interpretable neural network rather than design some post-hoc analyses to explain a neural network like most existing XAI works did. As pointed out by (Rudin, 2019), a large number of works have been conducted on the explainability of black-box models, but few efforts have been made on directly building an interpretable model. This work could be a valuable attempt towards this direction. On the other hand, most existing interpretable neural networks like the well-known perceptron (Rosenblatt, 1961; Freund and Schapire, 1999) are designed for supervised tasks. To the best of our knowledge, this could be the first attempt on interpretable neural networks for unsupervised tasks, or more specifically, clustering in this work.

- From the view of clustering, TELL implements the vanilla k-means with a neural network by reformulating its discrete objective as a neural layer, which enjoys the following advantages. First, the proposed TELL could be easily optimized by SGD in parallel, and we theoretically prove that the loss could be monotonously reduced. Second, the vanilla k-means requires the entire data set to update cluster centers in each iteration, which is computationally inefficient for large-scale data and even incapable of handling online data, *i.e.*, the data presented in streams. In contrast, our TELL optimizes the cluster centers through batch-wise SGD and directly predicts the cluster assignment for each point, which is promising in clustering large-scale and online data. Third, different from the vanilla k-means, TELL can be plugged into

any neural network to help it learning a clustering-favorable representation in an end-to-end fashion.

- From the view of differentiable programming, as far as we know, this could be one of the first attempts to benefit clustering with DP. On the one hand, this work aims at differentiable data clustering, whereas most existing DP works only focus on solving an optimization problem using a neural network (Gregor and LeCun, 2010; Liu et al., 2016; Zuo et al., 2015; Chen et al., 2015). On the other hand, our TELL recasts the vanilla k-means as a one-layer feedforward neural network (FNN), whereas most existing DP methods (Wang et al., 2015; Sprechmann et al., 2015; Liu et al., 2018) are build on recurrent neural networks (RNN). Therefore, this work might provide some novel insights to the community.

Mathematical Notations: Throughout the paper, **lower-case bold letters** represent column vectors and **upper-case bold letters** denote matrices. $\mathbf{A}^\top$ denotes the transpose of the matrix $\mathbf{A}$ and $\mathbf{I}$ denotes the identity matrix.

2. Interpretable Neural Clustering

In this section, we first show how to recast the vanilla k-means objective to a differentiable one on which a neural layer is built. Then, we discuss the interpretability of our model from the perspective of XAI, followed by the convergence proofs.

2.1 Deficiency of the Vanilla k-means

For a given data set $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n\}$, k-means aims to group each point $\mathbf{X}_i$ into one of $k \leq n$ sets $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k\}$ by minimizing the distance of the within-cluster data points, *i.e.*,

$$\underset{\mathcal{S}}{\operatorname{argmin}} \sum_j \sum_{\mathbf{X}_i \in \mathcal{S}_j} \|\mathbf{X}_i - \mathbf{\Omega}_j\|_2^2, \tag{1}$$

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

$$\mathbf{\Omega}_j = \frac{1}{|\mathcal{S}_j|} \sum_{\mathbf{X}_i \in \mathcal{S}_j} \mathbf{X}_i, \tag{2}$$

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 $\mathcal{S}$ and $\mathbf{\Omega}$ iteratively, *i.e.*, fixing one while optimizing the other. Such an iterative optimization has several drawbacks.

First, it is NP-hard to find the optimal solution for k-means in the Euclidean space, even for the bi-cluster problem. To ease the NP-hard problem, some variants of k-means are proposed, *e.g.*, parametric methods like Fuzzy c-means (Dunn, 1973; Bezdek, 1981). However, these methods are sensitive to the value of hyper-parameters that are daunting to tune.

Second, the vanilla k-means requires the entire data set to compute the cluster centers in each iteration (Yang et al., 2018; Bera et al., 2019; Marom and Feldman, 2019). As a result, it is impractical in large-scale or online clustering scenario, where data is presented in streams. More precisely, although one could assign the new-coming data to its nearest cluster center, the centers cannot be further updated unless one replicates the algorithm on all data, including the old and new.

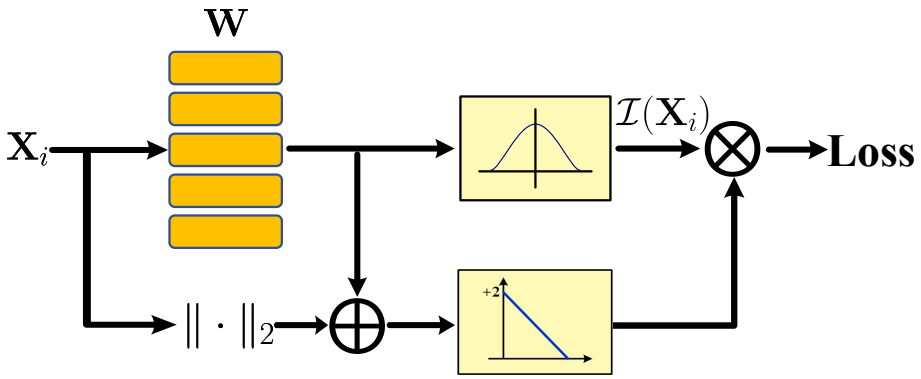


Figure 1: An illustration of the proposed TELL. This structure is exactly derived from the vanilla k-means objective and exhibits explicit interpretability. In brief, the hyperplane $\mathbf{W}$ is spanned by the clustering centers $\mathbf{\Omega}$ and the activation function normalizes the cluster assignment $\mathcal{I}(\cdot)$, which enjoys the model decomposability and algorithmic transparency as discussed in Section 2.3.

Third, the vanilla k-means is conducted on the fixed inputs and cannot assist the representation learning. As the success of deep learning largely depends on end-to-end learning, a plug-and-play neural clustering module is highly expected. In the proposed TELL, the cluster layer could not only perform clustering but also help the network to learn clustering-favorable representations in an end-to-end manner.

2.2 The Proposed Method

In this section, we first elaborate on how to reformulate the vanilla k-means into a differentiable neural layer in Section 2.2.1. Then, in Section 2.2.2, we theoretically prove the necessity of decoupling the weight $\mathbf{W}$ and bias $\mathbf{b}$ of the neural layer even though they are inherently correlated. In Section 2.2.3, we further reveal that such a decoupling strategy could cause divergent and unstable training. As a solution, we propose normalizing both the weight of the cluster layer and its gradient to stabilize the training. Finally, we present an end-to-end framework that could simultaneously learn a clustering-favorable representation and achieve clustering in Section 2.2.4, which proves the plug-and-play characteristic and effectiveness of the proposed TELL.

2.2.1 NEURAL NETWORK IMPLEMENTATION OF k-MEANS

To overcome drawbacks of the vanilla k-means mentioned in Section 2.1, we recast its objective function into a neural layer by rewriting Eq. (1) into

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

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.

The right part of Eq. (3) could be expanded into

$$\|\mathbf{X}_i - \mathbf{\Omega}_j\|_2^2 = \|\mathbf{X}_i\|_2^2 - 2\mathbf{\Omega}_j^\top \mathbf{X}_i + \|\mathbf{\Omega}_j\|_2^2. \tag{4}$$

We then define

$$\mathbf{W}_j = 2\mathbf{\Omega}_j, \quad \mathbf{b}_j = -\|\mathbf{\Omega}_j\|_2^2, \quad \|\mathbf{X}_i\|_2^2 = \beta_i \geq 0, \tag{5}$$

where $\mathbf{W}_j$ is the j-th column of $\mathbf{W}$, $\mathbf{b}_j$ is a scalar which denotes the j-th entry of $\mathbf{b}$, and β_i is a nonnegative constant corresponding to the length of data point $\mathbf{X}_i$.

With the above formulations, we could equivalently recast the scatter between data point $\mathbf{X}_i$ and cluster center $\mathbf{\Omega}_j$ as

$$\|\mathbf{X}_i - \mathbf{\Omega}_j\|_2^2 = \beta_i - \mathbf{W}_j^\top \mathbf{X}_i - \mathbf{b}_j. \tag{6}$$

For a given temperature factor $\tau > 0$, we relax the categorical variable $\mathcal{I}_j(\mathbf{X}_i)$ into

$$\mathcal{I}_j(\mathbf{X}_i) = \frac{\exp(-\|\mathbf{X}_i - \mathbf{\Omega}_j\|_2^2 \,/\tau)}{\sum_k \exp(-\|\mathbf{X}_i - \mathbf{\Omega}_k\|_2^2 \,/\tau)}. \tag{7}$$

In fact, the above definition of $\mathcal{I}_j(\mathbf{X}_i)$ can be regarded as the attention of $\mathbf{X}_i$ on the j-th cluster, which will be elaborated later in Section 2.3.

Combining Eq. (6) and Eq. (7), $\mathcal{I}_j(\mathbf{X}_i)$ could be computed with the proposed neural layer through

$$\mathcal{I}_j(\mathbf{X}_i) = \frac{\exp((\mathbf{W}_j^\top \mathbf{X}_i + \mathbf{b}_j - \beta_i)/\tau)}{\sum_k \exp((\mathbf{W}_k^\top \mathbf{X}_i + \mathbf{b}_k - \beta_k)/\tau)}. \tag{8}$$

Notably, the continuous categorical variable $\mathcal{I}_j(\mathbf{X}_i)$ could be computed using any normalization function including but not limited to *softmax* here. To avoid exhaustively tuning on the temperature parameter, in our implementation, we adopt an alternative by simply keeping the maximal entry of $\mathcal{I}_j(\mathbf{X}_i)$, which is the case when τ approaches 0 and is consistent with the vanilla k-means.

2.2.2 DECOUPLING THE NETWORK WEIGHT AND BIAS

To avoid confusions brought by complex mathematical notations, in the following analysis, we simply consider the case of one sample $\mathbf{x}$ without loss of generality. In this case, the objective function of TELL could be formulated as

$$\mathcal{L} = \sum_j \mathcal{L}_j = \sum_j \mathcal{I}_j(-\mathbf{W}_j^\top \mathbf{x} - \mathbf{b}_j + \beta), \tag{9}$$

where $\mathcal{I}_j$ shorts for $\mathcal{I}_j(\mathbf{x})$.

Though $\mathbf{W}$ and $\mathbf{b}$ are inherently coupled (*i.e.*, $\mathbf{b}_j = -\frac{\|\mathbf{W}_j\|_2^2}{4}$) according to the definition in Eq. (5), we theoretically prove that $\mathbf{W}$ and $\mathbf{b}$ should be decoupled during the training. In other words, $\mathbf{W}$ and $\mathbf{b}$ are optimized independently and the final cluster centers $\mathbf{\Omega}^*$ are recovered via $\mathbf{\Omega}^* = \frac{1}{2}\mathbf{W}^*$.

To demonstrate the necessity of decoupling $\mathbf{W}$ and $\mathbf{b}$, similar to the above reformulation, we rewrite Eq. (9) into

$$\begin{aligned}\mathcal{L} &= -\sum_j \frac{\exp((\mathbf{b}_j + \mathbf{W}_j^\top \mathbf{x} - \beta)/\tau)(\mathbf{b}_j + \mathbf{W}_j^\top \mathbf{x} - \beta)}{\sum_k \exp((\mathbf{b}_k + \mathbf{W}_k^\top \mathbf{x} - \beta)/\tau)} \\ &= -\sum_j \frac{\exp(\mathbf{z}_j/\tau)}{\sum_k \exp(\mathbf{z}_k/\tau)}\mathbf{z}_j \\ &= -\sum_j f(\mathbf{z}_j), \end{aligned} \tag{10}$$

where $\mathbf{z}_j = (-\frac{\|\mathbf{W}_j\|_2^2}{4} + \mathbf{W}_j^\top \mathbf{x} - \beta)$.

Correspondingly, the objective function becomes

$$\begin{aligned} &\max \ \sum_j f(\mathbf{z}_j) \\ &\text{s.t. } \mathbf{z}_j = -\frac{\|\mathbf{W}_j\|_2^2}{4} + \mathbf{W}_j^\top \mathbf{x} - \beta. \end{aligned} \tag{11}$$

As can be seen, Eq. (11) is equivalent to Eq. (3) when $\mathbf{W}$ and $\mathbf{b}$ are coupled, *i.e.*, $\mathbf{b}_j = -\frac{\|\mathbf{W}_j\|_2^2}{4}$. Since Eq. (11) obtains the optimum at the boundary with $\mathbf{z}_1 = \mathbf{z}_2 = \cdots$ and $f(\mathbf{z}) = \infty$ when $\mathbf{z} = \infty$, there exists $\mathbf{z}^*$ such that $\mathbf{z}_j = \mathbf{z}^*$ and $f(\mathbf{z}^*)$ reaches the optimum. We can always find a $\mathbf{W}_j$ and $\mathbf{b}_j$ such that $\mathbf{b}_j + \mathbf{W}_j^\top \mathbf{x} - \beta = \mathbf{z}^*$, while it is not guaranteed to have a $\mathbf{W}_j$ and $\mathbf{b}_j$ such that $-\frac{\|\mathbf{W}_j\|_2^2}{4} + \mathbf{W}_j^\top \mathbf{x} - \beta = \mathbf{z}^*$. Notably, though the above analyses are based on the case of a single sample, the conclusion still holds for multiple samples since they are independent from each other. In this sense, we have to decouple $\mathbf{W}_j$ and $\mathbf{b}_j$ during training to avoid the trivial solution.

2.2.3 NORMALIZE THE CLUSTER LAYER WEIGHT AND GRADIENT

In Section 2.2.2, we have shown that it is necessary to decouple $\mathbf{W}$ and $\mathbf{b}$ for preventing the network from descending into a trivial solution. However, we further notice that when $\mathbf{W}$ and $\mathbf{b}$ are decoupled, directly optimizing them would lead to divergent and unstable training. To address this issue, we propose normalizing both the cluster layer weight and its gradient to achieve a stable training, as illustrated in Fig. 2.

To be specific, when $\mathbf{W}$ and $\mathbf{b}$ are decoupled, minimizing the loss function $\sum_j \mathcal{I}_j(-\mathbf{W}_j^\top \mathbf{x} - \mathbf{b}_j + \beta)$ in Eq. (9) would lead both $\mathbf{W}_j^\top$ and $\mathbf{b}_j$ to infinity, as shown in Fig. 2(a). In this case, the optimization of the cluster layer never converges. To solve this problem, we propose simultaneously normalizing the weight and bias of the cluster layer. In practice, we adopt a more direct way by normalizing the cluster centers $\mathbf{\Omega}_j, j \in [1, k]$ to have a length of 1 (*i.e.*, $\mathbf{\Omega}_j = \mathbf{\Omega}_j/\|\mathbf{\Omega}_j\|$). Accordingly, to preserve the validity of Euclidean distance, data points are normalized to have a unit length as well (*i.e.*, $\beta = 1$). In this sense, $\mathbf{W}_j$ would have a length of 2 and $\mathbf{b}_j$ becomes a constant. As a result, the loss function in Eq. (9) could be rewritten into

$$\mathcal{L} = \sum_j \mathcal{L}_j = \sum_j \mathcal{I}_j(2 - \mathbf{W}_j^\top \mathbf{x}). \tag{12}$$

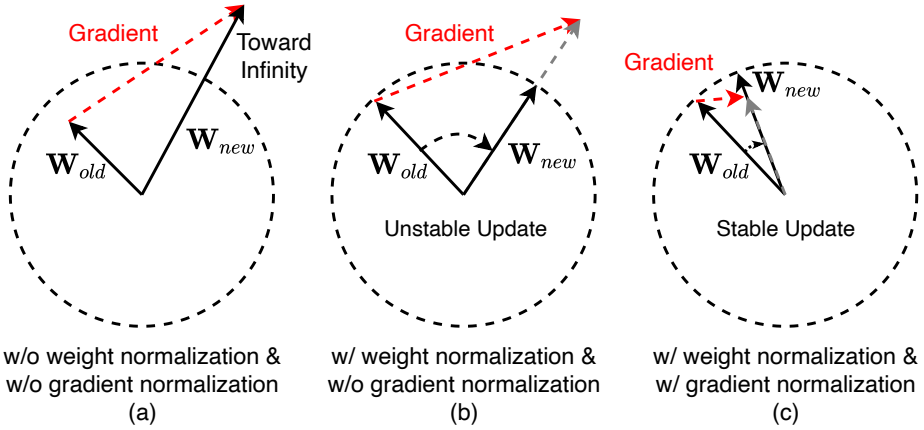


Figure 2: Three different cases of optimizing the cluster layer $\mathbf{W}$, *i.e.*, updating the cluster centers $\mathbf{\Omega}$. (a) Directly minimizing $-\mathbf{W}^\top\mathbf{X}$ with SGD, which leads $\mathbf{W}$ to infinity and non-convergence. (b) Conducting weight normalization after each SGD update. This will prevent $\mathbf{W}$ from going to infinity. However, when the gradient is much larger than the weight, $\mathbf{W}$ will be greatly changed after each update, which ends up in unstable training because the physical meaning of $\mathbf{W}$ may differ in each iteration. (c) Conducting gradient normalization and weight normalization, which promises stable training and is adopted in our implementation.

Note that since $\mathbf{W}_j$ is optimized through SGD, in practice, we have to renormalize it after each update. However, as illustrated in Fig. 2(b), when the gradient is much larger than the length of $\mathbf{W}_j$, $\mathbf{W}_j$ will be greatly changed after each update. Taking the MNIST data set as an example, $\mathbf{W}_{old}$ may correspond to the cluster center of digit "3" at first. However, when the gradient is considerably large, $\mathbf{W}_{new}$ would shift to the center of digit "5" after optimization. In other words, the intrinsic semantic meaning of $\mathbf{W}_j$ may differ in each iteration, which would cause unstable optimization, and thus the network is hard to converge.

Considering the aforementioned drawbacks, we propose simultaneously normalizing the weight and gradient as illustrated in Fig. 2(c). When the gradient is small enough, the cluster centers are mildly optimized and their semantic meanings keep the same across the training process, which promises a stable convergence. The ablation studies in Section 4.4 proves the effectiveness of such a gradient normalization strategy. In practice, we experimentally normalize the gradient to 10% of the length of $\mathbf{W}_j$.

2.2.4 END-TO-END TRAINING FOR CLUSTERING AND REPRESENTATION LEARNING

Based on the above discussions, we have recast the vanilla k-means to a neural layer with the following differentiable loss, namely,

$$\mathcal{L}_{clu} = \sum_{i,j} \mathcal{I}_j(\mathbf{X}_i)(2 - \mathbf{W}_j^\top \mathbf{X}_i). \tag{13}$$

Comparing with the vanilla k-means, one major advantage of our TELL is its plug-and-play characteristic, namely, it could be plugged into any neural network so that the deep representation

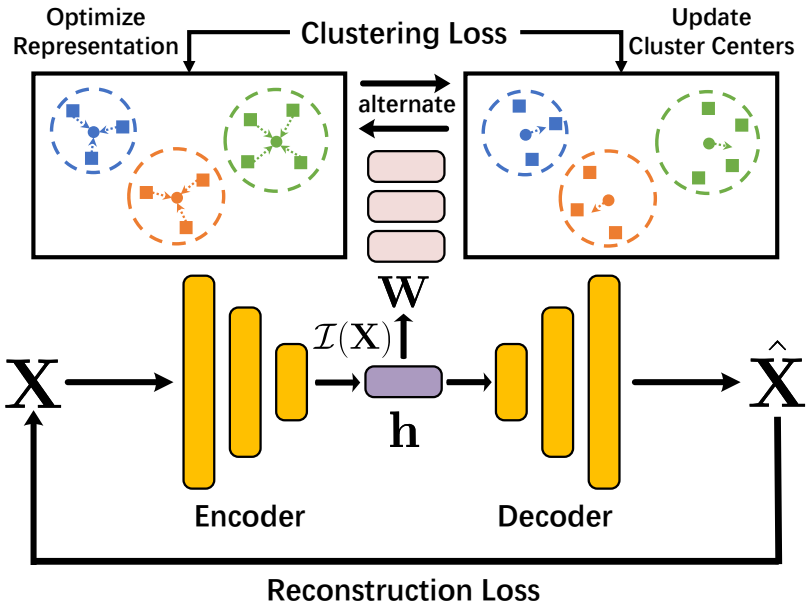


Figure 3: The end-to-end training framework. The reconstruction loss is used to optimize encoder and decoder simultaneously, while the clustering loss is used to optimize the cluster layer and encoder alternatively.

could be utilized to boost the clustering performance. To this end, instead of directly conducting clustering in the raw feature space, we use an autoencoder (AE) to extracted more discriminative features $\mathbf{h} = \{\mathbf{h}_1, \mathbf{h}_2, \dots\}$ by minimizing the following reconstruction loss, namely,

$$\begin{aligned} \mathbf{h}_i &= f(\mathbf{X}_i), \\ \hat{\mathbf{X}}_i &= g(\mathbf{h}_i), \\ \mathcal{L}_{rec} &= \sum_i \|\mathbf{X}_i - \hat{\mathbf{X}}_i\|_2^2, \end{aligned} \tag{14}$$

where $f(\cdot)$ and $g(\cdot)$ denote the encoder and decoder respectively, and $\mathbf{h}_i$ is normalized to have a unit length as aforementioned. By replacing $\mathbf{X}_i$ with $\mathbf{h}_i$ in Eq. (13), the overall loss of TELL is a combination of the reconstruction loss and the clustering loss, *i.e.*,

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{rec} + \lambda \mathcal{L}_{clu} \\ &= \sum_i \|\mathbf{X}_i - g(f(\mathbf{X}_i))\|_2^2 + \lambda \sum_{i,j} \mathcal{I}_j(\mathbf{X}_i)(2 - \mathbf{W}_j^\top f(\mathbf{X}_i)), \end{aligned} \tag{15}$$

where $\lambda = 0.01$ weights the two losses.

As can be seen, the reconstruction loss is used to simultaneously optimize the encoder $f(\cdot)$ and decoder $g(\cdot)$. For the clustering loss, we have shown that it can optimize the cluster layer weight $\mathbf{W}_j$. Here, to further improve the representability of features, we also optimize the encoder $f(\cdot)$ with the clustering loss by pulling features to their corresponding cluster centers. In practice, to

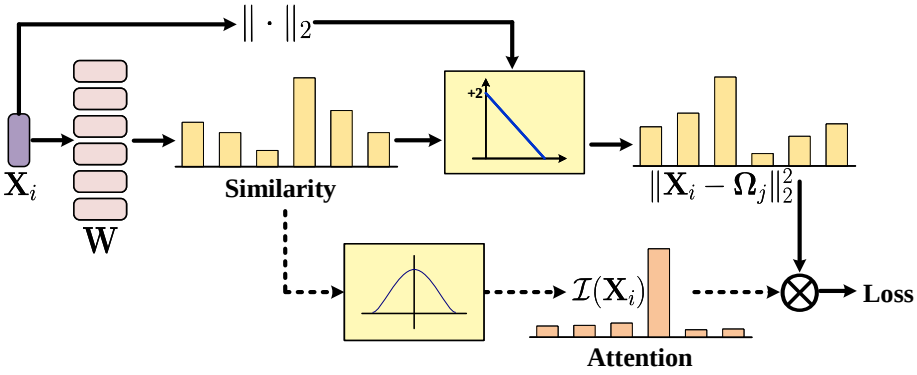


Figure 4: Besides the inhere interpretability, our TELL could also be understood from the perspective of the attention mechanism. The pathway below (denoted by the dot line) implements attention on the loss of the hyperplane $\mathbf{W}$ *w.r.t.* the input $\mathbf{X}_i$. In brief, our TELL learns the attention $\mathbf{X}_i$ paid on the cluster centers $\mathbf{\Omega}$.

stabilize the training, the right term of Eq. (15) is used to optimize $\mathbf{W}$ and $f(\cdot)$ alternatively. The overall end-to-end training framework is summarized in Fig. 3.

2.3 Interpretability of TELL

Although explainable artificial intelligence (XAI) has achieved remarkable progress recently (Arrieta et al., 2020), one barrier to the consensus on common grounds is the interchangeable misuse of "explainability" and "interpretability" in the literature. In brief, explainability often refers to post-hoc explanations by various approaches to enhance the understandability of the model, such as text explanations, visual explanations, explanations by simplification, and feature relevance explanations techniques. Different from explainability, interpretability is rooted in the design of the model itself, which is highly expected but also quite challenging. The interpretability is also expressed as transparency, which includes the model decomposability and algorithmic transparency (Arrieta et al., 2020). In the following, we will show that the proposed method (the cluster layer to be specific) enjoys these two interpretable characteristics.

Our TELL embraces model decomposability which stands for the feasibility to explain each part of the cluster layer. In other words, the input, weight parameters, activation, and loss function of our cluster layer are all interpretable. To be specific, the input to the cluster layer corresponds to the given data points, the weight $\mathbf{W}$ is exactly the cluster centers $\mathbf{\Omega}$, the *argmax* activation is used to achieve clustering by assigning each data point to its nearest cluster, and the loss function is recast from the vanilla k-means as shown in Eq. (4-5). To strengthen our claim on the interpretability, we also made some post-hoc explanations by visualizing the learned cluster centers reconstructed from $\mathbf{\Omega}$. As shown in Fig. 5, the reconstruction of cluster centers exactly corresponds to the MNIST digits, which demonstrates that TELL indeed captures the intrinsic semantic meanings.

Our TELL also possesses algorithmic transparency since its error surface or dynamic behavior can be reasoned about mathematically, allowing the user to understand how the model acts. To be specific, we not only theoretically provide the convergence analysis of our method later in Sec-

tion 2.4, but also show the necessity of decoupling $\mathbf{W}$ and $\mathbf{b}$, as well as normalizing both the weight $\mathbf{W}$ and its gradient, to achieve proper and stable optimization.

In addition, we could also understand the working manner of TELL from the standpoint of attention mechanism that is popular in natural language processing (Bahdanau et al., 2014; Vaswani et al., 2017). As shown in Fig. 4, TELL aims to learn a linear hyperplane $\mathbf{W}$ (denoted by the upper pathway) spanned by a set of cluster centers. The hyperplane is able to partition similar data points into the same cluster and dissimilar ones into different clusters based on attention. More specifically, to learn the hyperplane through Eq. (6), TELL computes the dissimilarity between input $\mathbf{X}_i$ and cluster centers $\mathbf{\Omega}$ via $(2 - \mathbf{W}^\top \mathbf{X}_i)$. After that, the loss of TELL is the summation of the weighted dissimilarity based on $\mathcal{I}(\mathbf{X}_i)$. Intuitively, this implements the attention mechanism as shown in the pathway below in Fig. 4, which decides the cluster centers $\mathbf{X}_i$ pays attention to. Actually, the attention here serves as the clustering assignment.

2.4 Convergence Proofs

In this section, we theoretically prove that the proposed loss $\mathcal{L}$ sufficiently converges with the SGD optimization. Due to the space limitation, we provide full details of the proof in Appendix A, including some supporting experimental studies.

For ease of presentation, let $\mathcal{L}^*$ denote the optimal loss, $\mathcal{L}_t^*$ be the smallest loss so far at step t, and $\mathbf{W}^*$ be the desirable weight which corresponds to the optimal cluster centers $\mathbf{\Omega}^*$. We consider the case that the standard SGD is used to optimize our network, *i.e.*,

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla \mathcal{L}(\mathbf{W}_t), \tag{16}$$

where $\nabla \mathcal{L}(\mathbf{W}_t)$ denotes the gradient of $\mathcal{L}$ *w.r.t.* $\mathbf{W}_t$. In the following, we abbreviate $\nabla \mathcal{L}(\mathbf{W}_t)$ to $\nabla \mathcal{L}_t$ for simplicity.

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

$$\|f(x_1) - f(x_2)\| \leq \epsilon \|x_1 - x_2\|, \tag{17}$$

where ϵ is termed as the Lipschitz constant.

Namely, the objective function $\mathcal{L}$ of TELL is Lipschitz continuous *i.i.f.* $\|\nabla \mathcal{L}_t\| \leq \epsilon$. In other words, to meet the Lipschitz continuity, we need to prove that the upper boundary of $\nabla \mathcal{L}_t / \tau$ exists.

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

Theorem 1 shows 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. As a matter of fact, there exists an upper boundary of $\|\mathbf{z}_i\|$ for any real-world data set. Furthermore, without loss of generality, one could normalize $\mathbf{x}$ and $\mathbf{\Omega}_i$ to meet $\|\mathbf{x}\| = \|\mathbf{\Omega}_i\| = 1$, and thus $\|\mathbf{W}_i\| = 2$ is bounded. Based on Theorem 1, we have the following theorem.

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

$$\mathcal{L}_T^* - \mathcal{L}^* \leq \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + \epsilon^2 \sum_t^T \eta_t^2}{2 \sum_{t=1}^T \eta_t}. \tag{18}$$

Based on Theorem 2, we could derive the following two lemmas.

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

$$\mathcal{L}_T^* - \mathcal{L}^* \to \frac{\eta\epsilon^2}{2}. \tag{19}$$

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

$$\mathcal{L}_T^* - \mathcal{L}^* \to \frac{\eta\epsilon}{2}. \tag{20}$$

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

3. Related Works

This work is closely related to XAI, clustering, and differentiable programming, which will be briefly introduced in this section.

3.1 Model Explainability vs. Interpretable Model

Generic deep architectures, as often referred to as "black-box" methods, rely on stacking somewhat ad-hoc modules, which makes it prohibitive to understand 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 (Zeiler and Fergus, 2014; Koh and Liang, 2017; Bau et al., 2017; Dosovitskiy and Brox, 2016; Kim et al., 2016) have devoted towards the explainability of neural networks. In general, these works mainly focus on establishing some post-hoc explanations by designing some visualization techniques or agent models to enhance the understandability of neural networks.

Different from these studies on the explainability of neural networks, we directly develop a novel interpretable neural network as advocated in (Rudin, 2019). To be specific, the proposed TELL enjoys not only the post-hoc explainability but also the interpretability by design (see Section 2.3). In short, one could explicitly understand why the structure of the TELL is presented as itself, the physical meaning of each part of the cluster layer, and why it can perform data clustering.

3.2 Stochastic k-means Clustering

To enhance the scalability of the vanilla k-means, the stochastic approximation was first presented in (Bottou and Bengio, 1995), which is also called online k-means. Another pathway is generalizing the idea to mini-batch k-means (Newling and Fleuret, 2016; Tang and Monteleoni, 2017). The major difference between stochastic and mini-batch k-means is that the former updates all centers asynchronously whereas the latter updates cluster centers after each iteration. Another difference is that mini-batch k-means is provable to converge to a local optimum, whereas it is not easy to promise that stochastic k-means could monotonically approximate the k-means objective, or even its expectation.

Compared with the stochastic methods, our TELL enjoys both online and mini-batch characteristics brought by its neural network implementation. In addition, instead of computing cluster

centers as the mean of samples, TELL optimizes the network weight $\mathbf{W}$ to obtain the centers via $\mathbf{\Omega} = \frac{1}{2}\mathbf{W}$. As far as we know, there has not been any attempt like TELL before to establish a differential neural network for the vanilla k-means or its stochastic variants. Another advantage of TELL is that it could be plugged into any neural network to perform clustering and help the network to learn a clustering-favorable representation in an end-to-end manner (see Section 2.2.4).

3.3 Differentiable Programming

Differentiable programming (DP, also called model-based optimization) is an emerging and impactful topic. It bridges classical machine learning models and deep neural networks by emphasizing problem-specific prior and interpretability. DP advocates building complicated end-to-end machine learning pipelines by assembling parameterized functional blocks, that are later jointly trained from examples, using some form of differential calculus—mostly stochastic gradient descent (SGD). It bears resemblances to building software, except that it is parameterized, automatically differentiated, and trainable/optimizable.

To the best of our knowledge, Learned ISTA (LISTA) (Gregor and LeCun, 2010) could be the first well-known DP work in the area of deep learning, which unfolds the ISTA algorithm (Blumensath and Davies, 2008), 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 have been proposed to address a variety of problems, *e.g.* image restoration (Chen et al., 2015), audio processing (Sprechmann et al., 2015), segmentation (Zheng et al., 2015), hashing (Liu et al., 2018), and clustering (Wang et al., 2015).

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

4. Experimental Results

In this section, we carry out experiments to verify the effectiveness of the proposed TELL comparing with 14 state-of-the-art clustering approaches. Due to the space limitation, we present additional theoretical and experimental analyses in the attached supplementary material.

4.1 Experimental Settings

All experiments are conducted on a Nvidia 2080Ti GPU with PyTorch 1.7.0 and CUDA 11.0. For all the compared baselines, we use the source code released by authors.

Baselines: We compare TELL with **i)** three popular subspace clustering approaches including Spectral Clustering (SC) (Ng et al., 2001), LRR (Liu et al., 2013), and LSR (Lu et al., 2012); **ii)** two large-scale clustering methods including Scalable LRR (SLRR) (Peng et al., 2015) and Large-scale Spectral Clustering (LSC) (Cai and Chen, 2015); **iii)** two matrix decomposition based methods and agglomerative clustering methods, *i.e.* NMF (Cai et al., 2011) and Zeta function based Agglomerative Clustering (ZAC) (Zhao and Tang, 2009); and **iv)** two deep learning based clustering methods, *i.e.*, Deep Embedding Clustering (DEC) (Xie et al., 2016) and Variational Deep Embedding (VaDE) (Jiang et al., 2016). Moreover, we also use the vanilla k-means, GMM, and FCM (Bezdek, 1981) as baselines. Notably, either of LSR and LSC has two variants, which are denoted by LSR1/LSR2 and LSC-R/LSC-K, respectively.

Implementation Details: We adopt a convolutional autoencoder to extract 10-dimensional features for all the data sets and then pass the feature into our cluster layer (see Fig. 3). More specifically, the encoder consists of four convolutional layers $conv(16, 3, 1, 1)$-$conv(32, 3, 2, 1)$-$conv(32, 3, 1, 1)$-$conv(16, 3, 2, 1)$ followed by a two-layer MLP $fc(256)$-$fc(10)$, where $conv(16, 3, 1, 1)$ denotes a convolutional layer with a channel number of 16, a kernel size of 3, a stride length of 1, and a padding size of 1, and $fc(256)$ denotes a fully connected layer with 256 neurons. Batch normalizations are applied after each convolutional layer, and the *ReLU* activation is used at the end of each layer except the last. The decoder is mirrored from the encoder, with the *sigmoid* activation at the output layer. Both the autoencoder and the cluster layer are randomly initialized with Kaiming uniform (He et al., 2015), which are then simultaneously trained for 3000 epochs with the default Adadelta (Zeiler, 2012) optimizer. Motivated by the vanilla k-means, in practice, we run TELL five times with different random initializations and obtain the final result by the run with the minimal clustering loss $\mathcal{L}_{clu}$ in Eq. (13)(see Appendix B for more details). For fair comparisons, we have tuned hyper-parameters for compared methods following the parameter tuning strategies suggested in the original papers and report them with corresponding results. Note that there is no hyper-parameter in the proposed method and no laborious tuning is needed.

Data sets: Our method is evaluated on the following three data sets, namely, the full MNIST handwritten digital database (Lecun et al., 1998), the full CIFAR-10 image database (Krizhevsky and Hinton, 2009), and the full CIFAR-100 image database (Krizhevsky and Hinton, 2009). For CIFAR-100, we adopt its 20 super-classes as partitions. In other words, we conduct experiments on the 20 super-classes of CIFAR-100 and report the mean, median, and maximum of the performance over these subsets, respectively. We normalize the data to be in the range of $[0, 1]$ before feeding them into the network, and no more preprocessing is applied.

Evaluation Metrics: Three widely used metrics are used to evaluate the clustering performance, including Clustering Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI). For these three metrics, a higher value indicates a better clustering performance.

4.2 Experimental Comparisons

In this section, we evaluate the performance of TELL on three image benchmarks, including MNIST, CIFAR-10, and CIFAR-100. The training and test split are merged in all our experiments. Specifically, the MNIST data set consists of 70,000 handwritten digits over 10 classes, and each grayscale image is of size 28×28. The CIFAR-10 data set consists of 60,000 RGB images of size $32 \times 32 \times 3$ from 10 classes. The CIFAR-100 data set contains 60,000 RGB images of size $32 \times 32 \times 3$ from 100 fine-grained classes which belong to 20 coarse-grained superclasses. The CIFAR-10/100 data sets are quite challenging and have been less touched in prior clustering works.

The clustering results on MNIST and CIFAR-10 are shown in Table 1. As can be seen, the proposed TELL gives superior clustering results in all three metrics. For example, the proposed TELL outperforms VaDE, a Gaussian Mixture Model (GMM) based deep clustering method, by 4.57% in the term of ARI on MNIST, which proves its effectiveness. Note that LRR and SLRR show inferior performance on these two data sets, which may attribute to that the data does not meet the low-rank assumption well. We would like to point out that the performance of TELL could be further improved when a more powerful representation learning method is adopted, as verified in Table 5.

Methods	MNIST				CIFAR-10			
	ACC	NMI	ARI	Parameter	ACC	NMI	ARI	Parameter
k-means	78.32	77.75	70.53	—	19.81	5.94	3.01	—
GMM	80.83	84.40	76.84	—	19.31	7.06	3.33	—
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
VaDE	92.36	86.58	85.09	—	20.87	7.20	3.95	—
TELL	**95.16**	**88.83**	**89.66**	—	**25.65**	**10.41**	**5.96**	—

Table 1: Clustering performance on the MNIST and CIFAR-10 data set.

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
GMM	28.37	38.07	28.80	27.53	32.10	**34.10**	32.30	33.57	29.67	28.43	38.07	31.29	30.89
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
VaDE	28.47	35.83	23.83	25.67	35.23	29.57	33.10	36.53	28.20	26.27	36.53	30.27	29.02
TELL	**34.07**	**46.20**	30.03	**31.47**	**37.30**	31.03	**36.90**	38.73	29.83	**35.13**	**46.20**	**35.07**	**34.60**

Table 2: Clustering accuracy on the first 10 super-classes of the CIFAR-100 data set.

For CIFAR-100, we conduct clustering on its 20 super-classes of which each contains 3000 images from 5 fine-grained classes. As shown in Tables 4.2 and 3, the proposed TELL shows encouraging performance, which is 1.72% and 3.30% higher than its best competitor in terms of mean ACC on the first and last 10 super-classes, respectively. Comparing with the recently proposed DEC and VaDE, our method earns a performance gain of 6.43% and 5.87% on the last 10 super-classes, respectively. The results demonstrate the effectiveness of TELL and the benefits brought by the end-to-end training paradigm. Note that only the ACC metric is reported here due to the space limitation.

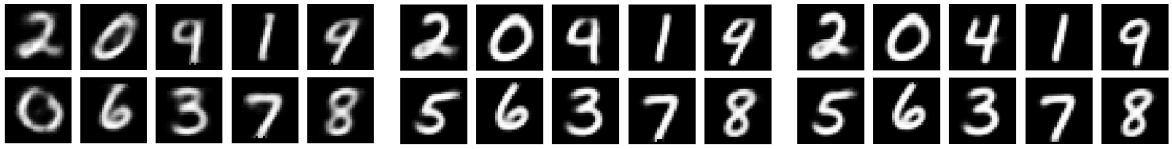
Methods	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	Max	Mean	Median
k-means	39.53	28.37	25.23	26.87	24.10	31.83	27.50	32.83	**31.00**	39.80	39.80	30.71	29.69
GMM	30.10	30.03	27.07	**29.50**	24.83	32.80	27.70	31.83	29.27	38.83	38.83	30.20	29.77
FCM	41.80	29.33	23.77	26.77	23.30	29.40	27.43	23.27	25.70	32.97	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
VaDE	31.07	32.00	27.17	28.47	**25.73**	24.63	28.23	21.87	24.57	38.90	38.90	28.26	27.70
TELL	**46.70**	**35.93**	**29.73**	29.10	24.63	**35.93**	30.53	**37.37**	27.57	**43.83**	**46.70**	**34.13**	**33.23**

Table 3: Clustering accuracy on the last 10 super-classes of the CIFAR-100 data set.

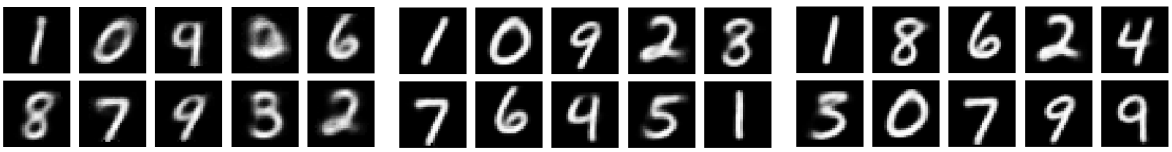
4.3 Visualization Analyses

To give a more intuitive understanding of the proposed TELL, in this section, we conduct two visualization analyses, including cluster center reconstruction and t-SNE visualization.

Cluster Center Reconstruction: We feed the cluster representations into the decoder and reconstruct the clustering centers learned by TELL and AE+k-means across the training process. According to reconstruction outcomes in Fig. 5, both TELL and k-means learn more representative cluster centers as the training goes. However, k-means could fail to capture the intrinsic 10 digit numbers and mix confusing digits like '3'-'8' or '3'-'5' up. Notably, TELL also confuses the digit '4' and '9' at the early training stage (see results at epoch 1000), but it successfully distinguishes them after more iterations. In other words, TELL is more likely to achieve the global optimum than the vanilla k-means.

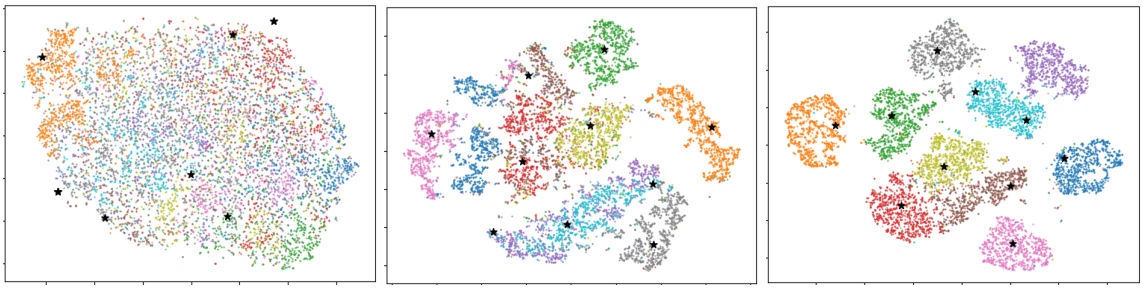


(a) TELL Epoch=100 (b) TELL Epoch=1000 (c) TELL Epoch=3000

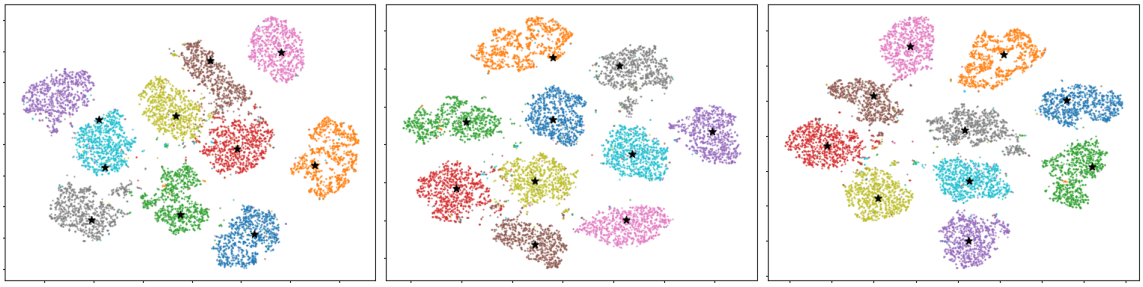


(d) AE+k-means Epoch=100 (e) AE+k-means Epoch=1000 (f) AE+k-means Epoch=3000

Figure 5: Cluster centers reconstruction on the MNIST data set.



(a) epoch=0 (b) epoch=50 (c) epoch=500



(d) epoch=1000 (e) epoch=1500 (f) epoch=3000

Figure 6: t-SNE visualization on the learned MNIST representations across the training process. Different digits are denoted with different colors, and the black pentagrams denote the cluster centers estimated by TELL.

t-SNE Visualization: To visualize the clustering result, we employ t-SNE (Maaten and Hinton, 2008) to reduce the dimensionality of the learned representation to two. As shown in Fig. 6(a), as the training goes, TELL learns a more compact and discriminative representation, which improves the separability of the estimated cluster centers.

4.4 Ablation Studies

In this section, we conduct three ablation studies to investigate the robustness and effectiveness of the proposed TELL. Specifically, we test the performance of TELL with different training paradigms, features, and optimization strategies.

Effectiveness of End-to-end Training: In our method, the clustering loss is used to iteratively update cluster centers and optimize instance features. To prove the effectiveness of such an end-to-end learning paradigm, we test TELL on fixed features learned by autoencoders, *i.e.*, the clustering loss is used to update cluster centers only.

In practice, we adopt a convolutional autoencoder to extract features from images with different channels (*e.g.*, 1 for MNIST and 3 for CIFAR-10/100). Here, to see how our TELL relies on the feature extraction ability of the network, we also test TELL with the fully connected autoencoder on the MNIST data set. The structure of encoder is $fc(500)$-$fc(500)$-$fc(2000)$-$fc(10)$, and the

Method	FCN			CNN		
	ACC	NMI	ARI	ACC	NMI	ARI
k-means	78.32	77.75	70.53	80.40	79.65	74.34
TELL-TwoStage	80.68	77.12	72.27	84.07	79.04	74.52
TELL	**93.25**	**85.52**	**85.76**	**95.16**	**88.83**	**89.66**

Table 4: Influence of different network structures on the MNIST data set. "TwoStage" means TELL is conducted on the fixed representations learned by the autoencoder instead of the standard end-to-end training.

decoder is symmetric. The *ReLU* activation is applied at the end of each layer except the last layer of encoder where no activation is used and the last layer of decoder where *sigmoid* is applied instead.

Table 4 shows that the performance of TELL with end-to-end training is much better than the "Two-Stage" paradigm, which suggests that pulling features to its corresponding cluster center helps the network to learn clustering-favorable representations. It is also interesting to note that under the "Two-Stage" learning paradigm, TELL finds slightly better cluster centers with higher ACC and ARI but a bit lower NMI than the vanilla k-means. Besides, TELL achieves slightly inferior performance with FCN than CNN due to the weaker feature extraction ability. But we would like to point out that the clustering accuracy of TELL with FCN still outperforms all the 14 compared baselines.

Influence of Feature Representability: We notice that the deep clustering method IMSAT (Hu et al., 2017) could achieve a promising result by employing some data augmentation techniques. It achieves 98.4% ACC on MNIST compared with 95.16% by TELL. However, we would like to point out that TELL enjoys the following advantages. On the one hand, TELL is a transparent neural model which could be interpreted from perspectives of model decomposability, algorithmic transparency, and post-hoc explainability. On the other hand, TELL is a plug-and-play online clustering module complementary to any neural network. In other words, the clustering performance of TELL would benefit from a better representation learning module. To see how the proposed TELL relies on the quality of representations, we test TELL on the raw data and features learned by IIC (Ji et al., 2019), which uses data augmentations to enhance the feature extraction ability. As shown in Table 5, TELL achieves a higher ACC than IMSAT. Besides, consistent with the results in Table 4, TELL could find slightly better cluster centers than the vanilla k-means with batch-wise optimization, while the latter needs the entire data set and is impractical for large-scale or online data sets.

Effectiveness on handling online data: To see how well TELL handles new coming data, we also test it under a different online scenario wherein the model is first trained on a relatively small data set and then evaluated on a large online data set. Specifically, 1,000 digits are randomly sampled from the MNIST test set as the training data on which we apply the vanilla k-means and train our TELL to obtain the clustering centers. After that, 60,000 digits from the MNIST training set are used to evaluate the performance. During the evaluation, the data is presented in an online manner (*i.e.*, each time only a batch of data comes, and it will not be accessible afterward). As the vanilla k-means could not update cluster centers without accessing the entire data set, clustering is achieved by assigning each digit to the closest cluster center computed on the early 1,000 digits.

Method	Raw			IIC		
	ACC	NMI	ARI	ACC	NMI	ARI
k-means	55.34	53.20	40.28	98.20	96.08	96.06
TELL-TwoStage	56.20	52.82	40.33	98.64	96.54	96.98
k-means†	48.52	46.75	31.25	98.10	95.90	95.85
TELL-TwoStage†	54.59	50.80	38.33	98.44	96.22	96.56

Table 5: Influence of features with different representability on the MNIST data set. '†' indicates that the network is trained on 1,000 samples randomly selected from the test set, and evaluated on 60,000 training samples in an online manner.

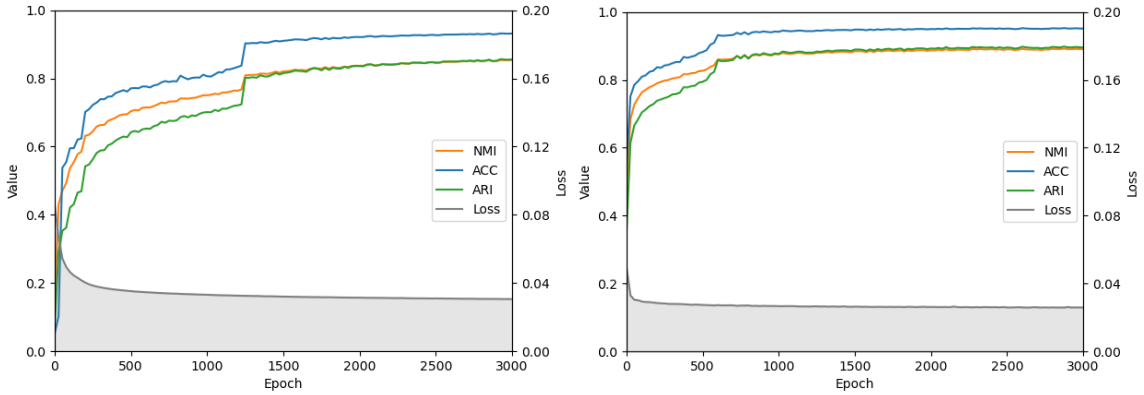
Optimizers	ACC	NMI	ARI	AMI	Homo.	Comp.	V_Mea.
AdaDelta	95.16	88.83	89.66	88.83	88.80	88.86	88.83
Adam	93.75	87.36	87.03	87.36	87.34	87.38	87.36
AdaDelta w/o grad. norm.	81.08	82.85	76.09	82.84	81.69	84.04	82.85
Adam w/o grad. norm.	80.49	80.48	73.36	80.47	79.32	81.67	80.48

Table 6: Influence of different optimizers on the MNIST data set.

On the contrary, our TELL could timely update parameters of the cluster layer to fit new coming data through Eq. (13). The last two rows in Table 5 indicate that such a batch-wise optimization of cluster centers improves the clustering performance on online data, especially when there is a fair degree of biases between the training and test set. Note that here the superior performance of TELL than the vanilla k-means solely attributes to its timely update of cluster centers, since the feature is fixed and will not be optimized through end-to-end representation learning introduced in Sec 2.2.4.

Influences of Optimizers: Besides the above investigations on different initializations and features, we further consider the role of the used optimization strategies. To this end, we carry out experiments on the MNIST data set by training TELL with two popular SGD variants, namely, Adadelta (Zeiler, 2012) and Adam (Kingma and Ba, 2015). In the implementation, we adopt the default setting for these optimizers. For a more comprehensive study, we adopt four more metrics to evaluate the clustering quality, *i.e.* Adjusted Mutual Index (AMI), Homogeneity (Homo.), Completeness (Comp.), and V_Measure (V_Mea.). Note that ACC, AMI, ARI, and AMI are external metrics that are computed based on the ground-truth, while Homogeneity, Completeness, and V_Measure are internal metrics that measure the compactness/divergence of within-/between-cluster samples. Table 6 shows that TELL with Adadelta performs slightly better than with Adam, and it is not necessary to tune the optimizer parameter.

Recall that to stabilize the update of cluster centers, we normalize their gradient to have an L2-norm of 0.1, which is 10% of the L2-norm of cluster centers. Here, to show the necessity of such a gradient normalization strategy, we conduct the following ablation study by simply applying the standard gradient descent on the cluster layer. One could see that both two optimizers give inferior performance when no normalization is performed. Because when the length of the gradient is much larger than that of cluster centers, the cluster centers change exceedingly at every iteration, thus preventing them from convergence.



(a) TELL with FCN.

(b) TELL with CNN.

Figure 7: Performance w.r.t training epoch on the MNIST data set. The left and right y-axis denote the clustering metrics and the loss, respectively.

4.5 Convergence Analysis

In Section 3.2, we have theoretically proved that our method will sufficiently approximate the global optimum under some mild conditions. In this section, we conduct experiments on the MNIST data set to verify our theoretical analysis. In Figure 7, we report the clustering performance and the loss value of our method with the fully connected neural network and the convolutional neural network. From the result, one could observe that TELL convergences after $\sim$1400/800 epochs with FCN/CNN in terms of NMI, ACC, ARI, and the loss value. It should be pointed out that the results reported in Table 1—3 are with 3000 training epochs, and if we continue training TELL to 5000 epochs, the performance could be further improved. Namely, TELL achieves better results of 93.62%, 86.25%, 86.47% and 95.54%, 89.72%, 90.44% in terms of ACC, NMI, and ARI with FCN and CNN, respectively.

5. Conclusion

In this paper, we directly build an interpretable neural layer which is a differentiable alternative of the vanilla k-means. The proposed clustering neural network overcomes some shortcomings of the vanilla k-means and owns the properties of parallel computing, provable convergence, online clustering, and clustering-favorable representation learning. In addition, our method enjoys interpretability in terms of model decomposability, algorithmic transparency, and post-hoc explainability. Such interpretability is inhered in the model itself rather than by an agent model, which is highly expected in XAI. It should be pointed out that this paper only focuses on the interpretability of the cluster layer. In the future, we plan to investigate how to create an interpretable neural module for representation learning as well.

Acknowledgments

The authors would thank to the handing editor Prof. Blei and anonymous reviewers for the constructive comments and valuable suggestions that greatly improve this work, and also thank Dr. Huan Li and Prof. Zhouchen Lin to provide helps and initial theoretical analysis on the decoupling of the network weight and bias. This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1406702, in part by NFSC under Grant U19A2081, 61625204, and 61836006; in part by the 111 Project under grant B21044; in part by ARC under Grant DP180100106 and DP200101328; in part by A*STAR AME Programmatic Funding Scheme A18A1b0045 and A18A2b0046.

Appendix A.

In this appendix, we provide full details of the convergence analysis of our TELL. In brief, we theoretically prove that the loss $\mathcal{L}$ of our network will sufficiently converge with the SGD optimization.

For ease of presentation, let $\mathcal{L}^*$ denote the optimal loss, and $\mathcal{L}_t^*$ be the smallest loss found so far at the t-th step. Similarly, $\mathbf{W}^*$ denotes the optimal weight corresponds to cluster centers $\mathbf{\Omega}^*$. We consider the case when the standard SGD is used to optimize our network, *i.e.*

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

where $\nabla \mathcal{L}(\mathbf{W}_t)$ denotes the gradient of $\mathcal{L}$ *w.r.t.* $\mathbf{W}_t$. In the following, we abbreviate $\nabla \mathcal{L}(\mathbf{W}_t)$ to $\nabla \mathcal{L}_t$ for simplicity.

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

$$\|f(x_1) - f(x_2)\| \leq \epsilon \|x_1 - x_2\|, \tag{2}$$

where ϵ is termed as the Lipschitz constant.

Namely, the objective function $\mathcal{L}$ of TELL is Lipschitz continuous *i.i.f.* $\|\nabla \mathcal{L}_t\| \leq \epsilon$. In other words, to meet the Lipschitz continuity, we need to prove that the upper boundary of $\nabla \mathcal{L}_t / \tau$ exists. To this end, we propose the following theorems.

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

Proof Without loss of generality, we consider our loss in the form of

$$\mathcal{L}(\mathbf{W}_i) = -\frac{\exp((\mathbf{W}_i^\top \mathbf{x} - 2)/\tau)(\mathbf{W}_i^\top \mathbf{x} - 2)}{\sum_k \exp((\mathbf{W}_k^\top \mathbf{x} - 2)/\tau)}. \tag{3}$$

Let $\mathbf{z}_i = (\mathbf{W}_i^\top \mathbf{x} - 2)/\tau$, we have

$$f(\mathbf{z}_i) = -\tau \frac{\exp(\mathbf{z}_i)(\mathbf{z}_i)}{\sum_j \exp(\mathbf{z}_j)} = -\tau \mathbf{p}_i \mathbf{z}_i, \tag{4}$$

and then

$$\begin{aligned} \nabla_i f(\mathbf{z}_i) =& -\tau \left(\frac{(\exp(\mathbf{z}_i) + \exp(\mathbf{z}_i)\mathbf{z}_i) \sum_j \exp(\mathbf{z}_j)}{(\sum_j \exp(\mathbf{z}_j))^2} - \frac{\exp(\mathbf{z}_i) \exp(\mathbf{z}_i)\mathbf{z}_i}{(\sum_j \exp(\mathbf{z}_j))^2} \right) \\ =& \tau(-\mathbf{p}_i - \mathbf{p}_i \mathbf{z}_i + \mathbf{p}_i \mathbf{p}_i \mathbf{z}_i). \end{aligned} \tag{5}$$

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

$$\|\nabla_i f(\mathbf{z}_i)\| \leq \tau \|\mathbf{p}_i\| (1 + \|\mathbf{z}_i\| + \mathbf{p}_i \|\mathbf{z}_i\|) \leq \tau(1 + \|\mathbf{z}_i\| + \|\mathbf{z}_i\|). \tag{6}$$

It shows that our objective function $\mathcal{L}(\mathbf{W}_i)$ will be upper bounded by a positive real number ϵ when $\|\mathbf{z}_i\|$ is bounded (see Figure 1 for an illustrative example). In fact, there exists the upper boundary of $\|\mathbf{z}_i\|$ for any real-world data set. Moreover, without loss of generality, one could enforce $\|\mathbf{X}_i\| = 1$ and $\|\mathbf{\Omega}_i\| = 1$ and thus $\|\mathbf{W}_i\| = 2$ is bounded. ■

Based on Theorem 1, we have the following convergence result based on (Boyd et al., 2003).

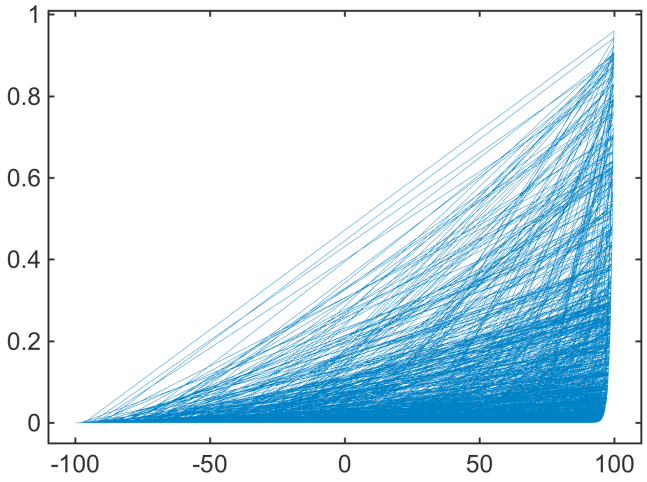


Figure 1: A toy example to show the bound of our loss function in the 1-dimensional case. The x-axis denotes the data points ($\mathbf{z}$) randomly sampled from -100 to 100, and the y-axis denotes the corresponding loss value. One could see that our loss function will be bounded if $\mathbf{z}$ is bounded.

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.,*

$$\mathcal{L}_T^* - \mathcal{L}^* \leq \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + \epsilon^2 \sum_t^T \eta_t^2}{2\sum_{t=1}^T \eta_t}. \tag{7}$$

Proof Let $\mathbf{W}^* = 2\mathbf{\Omega}^*$ be the minimizer to our objective function, *i.e.* Eq. (3), then

$$\|\mathbf{W}_{T+1} - \mathbf{W}^*\|_F^2 = \|\mathbf{W}_T - \mathbf{W}^*\|_F^2 - 2tr(\eta_T \nabla \mathcal{L}_T^\top (\mathbf{W}_T - \mathbf{W}^*)) + \eta_t^2 \|\nabla \mathcal{L}_T\|_F^2, \tag{8}$$

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

By recursively applying the above equation, we have

$$\|\mathbf{W}_{T+1} - \mathbf{W}^*\|_F^2 = \|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 - 2\sum_{t=1}^T \eta_t tr(\mathbf{W}_t - \mathbf{W}^*) + \sum_{t=1}^T \eta_t^2 \|\nabla \mathcal{L}_t\|_F^2. \tag{9}$$

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

$$f(x^*) \geq f(x_t) + \nabla \mathcal{L}_t^\top (x^* - x_t) \tag{10}$$

then,

$$\|\mathbf{W}_{T+1} - \mathbf{W}^*\|_F^2 \leq \|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 - 2\sum_{t=1}^T \eta_t (\mathcal{L}_t - \mathcal{L}^*) + \epsilon^2 \sum_{t=1}^T \eta_t^2, \tag{11}$$

$$2\sum_{t=1}^T \eta_t (\mathcal{L}_t - \mathcal{L}^*) \leq \|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + \epsilon^2 \sum_{t=1}^T \eta_t^2, \tag{12}$$

$$\mathcal{L}_t - \mathcal{L}^* \geq \min_{t=1,2,\cdots,T}(\mathcal{L}_t - \mathcal{L}^*) = \mathcal{L}_T^* - \mathcal{L}^*, \tag{13}$$

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

Combining Eq. (12) and Eq. (13), we finally have

$$\mathcal{L}_T^* - \mathcal{L}^* \leq \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + \epsilon^2 \sum_{t=1}^T \eta_t^2}{2\sum_{t=1}^T \eta_t}. \tag{14}$$

■

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

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

$$\mathcal{L}_T^* - \mathcal{L}^* \to \frac{\eta\epsilon^2}{2}. \tag{15}$$

Proof After T steps, we have

$$\mathcal{L}_T^* - \mathcal{L}^* \leq \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2 + T\epsilon^2\eta^2}{2T\eta} = \frac{\|\mathbf{W}_1 - \mathbf{W}^*\|_F^2/(T\eta) + \eta\epsilon^2}{2}. \tag{16}$$

■

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

$$\mathcal{L}_T^* - \mathcal{L}^* \to \frac{\eta\epsilon}{2} \tag{17}$$

Proof Similar to the proof for Lemma 1. ■

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

Run	ACC	NMI	ARI	$\mathcal{L}_{clu}$
1	**95.16**	88.83	**89.66**	**0.018138861**
2	95.10	**89.03**	89.53	0.018156468
3	82.88	87.00	81.01	0.018166682
4	82.04	83.90	77.05	0.018166506
5	81.69	83.87	77.33	0.018164165

Table S1: The clustering performance on MNIST in five different runs with random initialization. The best and second-best results are shown in bold and underline, respectively.

Run	ACC	NMI	ARI	$\mathcal{L}_{clu}$
1	24.44	**10.59**	5.37	0.018088905
2	24.97	10.19	5.73	0.018088135
3	22.19	10.40	5.50	0.018093264
4	23.25	10.12	5.36	0.018089087
5	**25.65**	10.41	**5.96**	**0.018088113**

Table S2: The clustering performance on CIFAR-10 in five different runs with random initialization. The best and second-best results are shown in bold and underline, respectively.

Appendix B.

As a neural surrogate of the vanilla k-means, the performance of TELL naturally relates to the network initialization. Just like all machine learning methods, TELL faces the problem of model selection as well. To solve model selection in unsupervised setting, motivated by the vanilla k-means, in practice, we run TELL several times with different random initializations and obtain the final result by the run with the minimal clustering loss $\mathcal{L}_{clu}$. To show the correctness of such a criterion, we provide the clustering performance of TELL in five different runs on MNIST and CIFAR-10 in Table S1 and S2.

The results show that the clustering accuracy of TELL is correlated to the clustering loss $\mathcal{L}_{clu}$. On both data sets, the best and second-best results correspond to the smallest two clustering losses. As the clustering loss measures the within-cluster distance, a smaller clustering loss indicates a more compact clustering result. Thus, in practice, one could always run TELL several times and choose the run with the smallest $\mathcal{L}_{clu}$ as the final output.

References

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, cs.CL, 2014.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network Dissection - Quantifying Interpretability of Deep Visual Representations. In *Proc of IEEE Conf Comput Vis and Pattern Recognit*, pages 3319–3327. IEEE, 2017.

Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In *Advances in Neural Information Processing Systems*, pages 4954–4965, Dec. 2019.

James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. ISBN 0306406713.

Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5):629–654, 2008.

Léon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 585–592. MIT Press, 1995.

Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.

D. Cai and X. Chen. Large scale spectral clustering via landmark-based sparse representation. *IEEE Trans. on Cybern.*, 45(8):1669–1680, Aug 2015. ISSN 2168-2267. doi: 10.1109/TCYB.2014.2358564.

Deng Cai, Xiaofei He, and Jiawei Han. Locally consistent concept factorization for document clustering. *IEEE Trans. Knowl. Data Eng.*, 23(6):902–913, Jun. 2011. ISSN 1041-4347.

Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Theoretical Linear Convergence of Unfolded ISTA and Its Practical Weights and Thresholds. In *Proc. of Adv. in Neural Inf. Process. Syst.*, pages 9079–9089, Vancouver, CA, December 2018.

Y. Chen, Wei 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*, pages 5261–5269, Boston, MA, Jun. 2015. doi: 10.1109/CVPR.2015.7299163.

Alexey Dosovitskiy and Thomas Brox. Inverting Visual Representations with Convolutional Networks. In *Proc of IEEE Conf Comput Vis and Pattern Recognit*, pages 4829–4837, 2016.

J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2765–2781, 2013.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, Dec 1999.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proc of 27th Int Conf Mach Learn*, pages 399–406, USA, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Toby Hocking, Jean-Philippe Vert, Francis R Bach, and Armand Joulin. Clusterpath - An Algorithm for Clustering using Convex Fusion Penalties. In *Proc Int Conf Mach Learn*, pages 745–752, Washington, USA, Jul 2011.

Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proc Int Conf Machine Learning*, volume 70, pages 1558–1567, Aug 2017.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput Surv*, 31(3): 264–323, Sep. 1999. ISSN 0360-0300.

Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. In *Proc of 29th Adv in Neural Inf Process Syst*, Montréal, Canada, Dec. 2017.

Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874, 2019.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

Been Kim, Oluwasanmi Koyejo, and Rajiv Khanna. Examples are not enough, learn to criticize! Criticism for Interpretability. In *Proc of Adv. in Neural Inf Process Syst*, pages 2288–2296, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of 3th Int. Conf. Learn Rep.*, pages 1–15, 2015.

Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *Proc of Int Conf Machine Learn*, 2017.

A. Krizhevsky and G.E. Hinton. Learning multiple layers of features from tiny images, 2009.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of IEEE*, 86(11):2278–2324, Nov. 1998.

Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. 35, Feb. 2021.

Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1): 171–184, 2013.

Jialin Liu, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. ALISTA: Analytic Weights Are As Good As Learned Weights in LISTA. In *Proc Int Conf Learn Rep*, New Orleans, May 2019.

Q. Liu, G. Liu, L. Li, X. Yuan, M. Wang, and W. Liu. Reversed spectral hashing. *IEEE Trans Neur Netw Learn Syst*, 29(6):2441–2449, June 2018. ISSN 2162-237X. doi: 10.1109/TNNLS.2017.2696053.

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*, 38(12):2457–2471, Dec 2016. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2522415.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from data. In *Proc 35th Int Conf Machine Learn*, pages 5067–5078, Jan. 2018.

Canyi Lu, Hai Min, ZhongQiu Zhao, Lin Zhu, DeShuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In *Proc. of 12th Eur. Conf. Comput. Vis.*, pages 347–360, Florence, Italy, Oct. 2012.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *J Mach Learn Res*, 9 (Nov):2579–2605, 2008.

Yair Marom and Dan Feldman. k-means clustering of lines for big data. In *Advances in Neural Information Processing Systems*, pages 12817–12826, Dec. 2019.

James Newling and François Fleuret. Nested mini-batch k-means. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1352–1360. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6481-nested-mini-batch-k-means.pdf`.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. of 14th Adv. in Neural Inf. Process. Syst.*, pages 849–856, Vancouver, Canada, Dec. 2001.

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*, PP(99):1–14, 2015. ISSN 2162-237X.

Xi Peng, Shijie Xiao, Jiashi Feng, Wei-Yun Yau, and Zhang Yi. Deep subspace clustering with sparsity prior. In *Proc of 25th Int Joint Conf Artif Intell*, pages 1925–1931, New York, NY, USA, Jul. 2016.

Xi Peng, Zhiding Yu, Zhang Yi, and Huajin Tang. Constructing the l2-graph for robust subspace learning and subspace clustering. *IEEE Trans. Cybern.*, 47(4):1053–1066, Apr. 2017. ISSN 2168-2267.

Xi Peng, Hongyuan Zhu, Jiashi Feng, Chunhua Shen, Haixian Zhang, and Joey Tianyi Zhou. Deep clustering with sample-assignment invariance prior. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4857–4868, Nov. 2020. ISSN 2162-237X. doi: 10.1109/TNNLS.2019.2958324.

Frank Rosenblatt. Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019.

P. Sprechmann, A. M. Bronstein, and G. Sapiro. Learning efficient sparse and low rank models. *IEEE Trans Pattern Anal and Machine Intelli*, 37(9):1821–1833, Sep. 2015.

Cheng Tang and Claire Monteleoni. Convergence rate of stochastic k-means. In *Proc of Inf Conf Artificial Intelligence and Statistics*, volume 54, pages 1495–1503, Fort Lauderdale, FL, Apr 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Proc of Adv. in Neural Inf Process Syst*, 2017.

Shusen Wang, Alex Gittens, and Michael W Mahoney. Scalable Kernel K-Means Clustering with Nyström Approximation: Relative-Error Bounds. *J Mach Learn Res*, 20(12):1–49, 2019.

Zhangyang Wang, Shiyu Chang, Jiayu Zhou, Meng Wang, and Thomas S. Huang. Learning a task-specific deep architecture for clustering. In *Proc of SIAM Int Conf Data Mining*, pages 369–377, Miami, Florida, May 2015.

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.

Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint Unsupervised Learning of Deep Representations and Image Clusters. In *Proc of 29th IEEE Conf Comput. Vis and Pattern Recognit*, pages 5147–5156, Las Vegas, NV, Jun 2016. IEEE.

Yingzhen Yang, Jiashi Feng, Nebojsa Jojic, Jianchao Yang, and Thomas S. Huang. Subspace learning by l0-induced sparsity. *Int J of Computer Vis*, Jul 2018. ISSN 1573-1405. doi: 10.1007/s11263-018-1092-4. URL https://doi.org/10.1007/s11263-018-1092-4.

Jinfeng Yi, Lijun Zhang, Rong Jin, Qi Qian, and Anil K. Jain. Semi-supervised clustering by input pattern assisted pairwise similarity matrix completion. In *Proc of the Int Conf Machine Learn*, pages 1400–1408, 2013.

Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Proc Euro Conf Computer Vis*, pages 818–833, Cham, 2014. Springer International Publishing.

Deli Zhao and Xiaoou Tang. Cyclizing clusters via zeta function of a graph. In *Proc. of 21th Adv. in Neural Inf. Process. Syst.*, pages 1953–1960, Vancouver, Canada, Dec. 2009.

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*, pages 1529–1537, Santiago, Chile, Dec 2015. doi: 10.1109/ICCV.2015.179.

Wangmeng Zuo, Dongwei Ren, Shuhang Gu, Liang Lin, and Lei Zhang. Discriminative learning of iteration-wise priors for blind deconvolution. In *Proc. of 28th IEEE Conf. Comput. Vis. and Pattern Recognit.*, pages 3232–3240, Boston, MA, Jun. 2015.