Optimal k-Discretization Learning

Tong Wang, Zhangyang Wang
University of Texas at Austin
`tong.wang@utexas.edu, atlaswang@utexas.edu`

The k-discretization problem is known to be NP-hard in general. Existing algorithms exploit various heuristics and obtain at most local minima that are sensitive to initializations. This paper starts by discussing how to leverage polynomial-time optimal solvers for 1-D k-discretization which can serve as a powerful and parsimonious regularizer for complex learning tasks. The algorithm can be accelerated by sampling, with bounded approximation errors proven. The paper then presents an embedding learning approach to handle multi-dimensional k-discretization, based on the 1-D solution. Equipped with many novel task-specific modifications, the proposed approach achieves highly promising performance on a vast variety of application tasks, including signal quantization, image clustering, and image smoothening. Our codes are available at `hhttps://github.com/VITA-Group/SnC`.

1. Motivation and Background

This paper explores the *k-discretization* problem, a general term defined for a class of optimization problems whose target solutions contain no more than k *distinct* elements. As a fundamental topic of interests, the k-discretization problem is found to be extremely useful, and bears many different names in various applications, such as *partition* in the set theory, *quantization* in digital signal processing, *clustering* in cluster analysis, *hashing* in information retrieval, *segmentation* or *smoothening* in image processing and computer vision, and so on. An exhaustive literature survey is beyond the scope of this paper. We hereby mention a few most related general algorithms below, and leave the application-specific prior work to be reviewed later.

The best-known existing algorithm for k-discretization is arguably *k-means* [1], or called the Lloyd's algorithm. It partitions data into k clusters (Voronoi cells), in which each observation belongs to the cluster with the nearest mean, serving as a *centroid* or *prototype* of the cluster. k-means is NP-hard, and is commonly solved by efficient heuristics which at most converge to local optima. As a result, the k-means solution is usually unstable and sensitive to the initialization [2]. Moreover, the local solution found by k-means can be arbitrarily bad with respect to the objective function value compared to the global optimum, although it could also converge to the global optimum with a good probability when clusters are well separated [3].

Many variants have been developed to improve the robustness of k-means. The *k-medoids* algorithm [4] chooses actual data instances as medoids. It minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances, and appears to be more robust to noise and outliers. The *k-median* algorithm [5] calculates the median for each cluster as its centroid, instead of the mean. The *k-means*++ algorithm [6] specifies a procedure to initialize the centroids before proceeding with the standard k-means, and can avoid some poor results. Broadly speaking, k-discretization is also loosely related to Gaussian mixture model (GMM), bilateral filtering and mean shift clustering [7].

While the general k-discretization problem is undoubtedly NP-hard for the same reason as k-means, this paper presents that the 1-D k-discretization problem guarantees to find the global optimum within polynomial time, using dynamical programming (DP) [8]. The algorithm can be accelerated by sampling, with the approximation error provably bounded. Simulations on 1-D signal quantization certify our theoretical results. Based on the 1-D algorithm, we then present an embedding-based approach to handle multi-dimensional k-discretization, and apply it to the task of clustering. We also evaluate our proposed algorithms to the tasks of image smoothening, combined with task-specific customizations such as dictionary learning and total variation regularization. While the main goal is to demonstrate the wide applicability of k-discretization, we are pleased to find that the proposed models have achieved appealing performance on all those tasks.

Third Conference on Parsimony and Learning (CPAL 2026).

2. 1-D k-Discretization: Algorithm and Theory

2.1. Dynamic programming with global optimality

Define $\|\mathbf{z}\|_{d0}$ as a function that counts the number of distinct elements in a 1-D vector $\mathbf{z}$. For example, $\|\mathbf{z}\|_{d0} = 3$ if $\mathbf{z} = [0, 4, 3, 0, 4, 4]$. Note that despite the notation form used for convenience, $\| \cdot \|_{d0}$ is neither a norm nor a semi-norm, since the absolute homogeneity is obviously not satisfied. We start by considering the following projection operator for 1-D k-discretization:

$$\min_{\mathbf{z}} \quad \|\mathbf{z} - \mathbf{x}\|^2 \quad \text{s.t.} \quad \|\mathbf{z}\|_{d0} \leq k. \tag{1}$$

This problem can be reduced to the following. Without the loss of generality, we assume that $\mathbf{x} = (x_1, \ldots, x_n)^\top$ is sorted, where $x_1 \geq x_2 \geq \ldots \geq x_n$. Let $D \in \mathbb{R}^{(n-1)\times n}$ be the first-order finite difference matrix [9]. Our goal is to find a piecewise constant approximation $\mathbf{z} \in \mathbb{R}^n$ for $\mathbf{x}$ to minimize the following objective:

$$\min_{\mathbf{z}} \quad \|\mathbf{z} - \mathbf{x}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \|D\mathbf{z}\|_0 \leq k - 1. \tag{2}$$

Apparently, the optimal solution $\mathbf{z}$ will be sorted too. Geometrically, the l_0 norm constraint requires less than $k-1$ discontinuous points in $\mathbf{z}$. Therefore, there are no more than k different values in $\mathbf{z}$. For an unsorted $\mathbf{x}$, we can first sort it, then follow the solution for a sorted $\mathbf{x}$, and finally sort back the solved $\mathbf{z}$ using the original sorting index.

The problem 2 can be solved using a dynamic programming (DP) approach with a recurrence relation to ensure global optimality [8]. Let $T(m, r)$ denote the minimum loss to approximate $\mathbf{x}_m$, the first m terms of $\mathbf{x}$, using at most r different values, $1 \leq m \leq n$, $1 \leq r \leq k$, that is,s

$$\begin{aligned} T(m, r) = &\min_{\mathbf{z}_m} \|\mathbf{z}_m - \mathbf{x}_m\|_{\mathrm{F}}^2 \\ &\text{s.t.} \quad \|D\mathbf{z}_m\|_0 \leq r - 1, \mathbf{x}_m = (x_1, \ldots, x_m)^\top. \end{aligned} \tag{3}$$

For the base cases with $r = 1$, we can simply calculate the mean of the first m terms, *i.e.*,

$$T(m, 1) = \sum_{j=1}^{m} \left(x_j - \frac{1}{m} \sum_{j=1}^{m} x_j \right)^2 .$$

For subproblems with $1 < r \leq k$, we can find an index $i < m$, such that we use $r - 1$ different values to fit the first i terms in $\mathbf{x}_m$, and use only one value (i.e., the average) to fit the remaining $m - i$ terms:

$$\begin{aligned} T(m, r) = &\min_{r-1 \leq i \leq m-1} T(i, r-1) + \\ &\sum_{j=i+1}^{m} \left(x_j - \frac{1}{m-i} \sum_{j=i+1}^{m} x_j \right)^2 \end{aligned} \tag{4}$$

The global optimality of the above DP solution is intuitive by definition and can be formally proven by deduction (in Section B).

It seems infeasible to directly extend the 1-D solution to multi-dimensional cases. For one reason, there are clearly defined sorting criteria in higher dimensions. Later, we will introduce an embedding-based approach to convert multi-dimensional k-discretization into the well-solved 1-D problem.

2.2. Sampling-then-Clustering

By re-using subproblems, pre-computing re-usable terms[1] and building a DP table, the above algorithm takes $O(n^2k)$ in time complexity and $O(nk)$ in space complexity. As time complexity increases quadratically with n, efficiency remains a major bottleneck for high dimension m. A fairly straightforward idea is to sample a

[1]A key acceleration trick is to pre-compute the $k = 1$ minimum losses, for all subsequences that start at x_i and end at x_j, $\forall 1 \leq i < j \leq n$. Although the trick does not change the $O(n^2k)$ time complexity, it speeds up the implementation dramatically.

Algorithm 1 Sampling-then-Clustering (SnC) Acceleration

```
Input: Large data vector $x \in \mathbb{R}^N$, number of levels $k$, sample rate $s \in (0, 1)$.
Output: Approximate discretized vector $z \in \mathbb{R}^N$.
 1: Step 1: Sampling
 2: $n \leftarrow \lceil s \cdot N \rceil$
 3: Sample a subset of indices $S \subset \{1, \ldots, N\}$ of size $n$, with replacement.
 4: Let $x_S$ be the vector of sampled values $\{x_i \mid i \in S\}$.
 5: Step 2: Optimal Clustering on Sample
 6:                                         ▷ Solve the exact 1-D problem on the smaller sample set
 7: $z_S \leftarrow$ Algorithm 2$(x_S, k)$                 ▷ Algorithm 2 is presented in Appendix
 8: Extract the set of unique values (centroids) found in $z_S$:
 9: $\mathcal{C} \leftarrow$ unique$(z_S) = \{c_1, c_2, \ldots, c_k\}$
10: Step 3: Full Quantization
11:                                ▷ Assign every element in the full dataset to the nearest centroid
12: for $i = 1$ to $N$ do
13:     $z_i \leftarrow \arg\min_{c \in \mathcal{C}} |x_i - c|$
14: end for
15: return $z$
```

subset of elements from $\mathbf{x}$ to form a "down-sampled" version $\mathbf{x}_s$, and solve 1-D k-discretization (1) on $\mathbf{x}_s$ to obtain $\mathbf{z}_s$. Denoting by $\mathcal{Z}_s$ the set of k distinct values in $\mathbf{z}_s$, we then scan through $\mathbf{x}$ and quantize each $\mathbf{x}_i$ to its nearest neighbor in $\mathcal{Z}_s$. We call the strategy *sampling-then-clustering* (**SnC**). The overview of SnC is presented in Algorithm 1. With the sample rate $s \in (0, 1)$, the time complexity is reduced to $O(s^2n^2k)$ after applying SnC. As straight-forward as SnC might look like, we will next establish the approximation error bound between the SnC solution and the global optimum.

2.2.1. Error bound for Sampling-then-Clustering

Given a sample set Ω of 1-D points $\{x_1, x_2, \cdots, x_N\}$, denote by $C_\Omega := \{C_\Omega^k\}_{k=1}^K$ the optimal K-partition and $\mu_\Omega := \{\mu_\Omega^k\}_{k=1}^K$ the corresponding centers. The convex hulls Conv(Ω^k) and Conv$(\Omega^{k'})$ are disjointed for any k and k'. Applying the optimal partition with respect to Ω to another set of 1-D points, we can denote by

$$f_{\Omega'}(\mu_\Omega, C_\Omega) = \frac{1}{|\Omega'|}\sum_{k=1}^{K}\sum_{x \in \Omega' \cap C_\Omega^k}(x - \mu_\Omega^k)^2.$$

Let S denote the set of n samples from Ω with replacement[2]. Let $\{C_S^k\}_{k=1}^K$ be the optimal partition based on the sample set S using the above SnC strategy, and $\{\mu_S^k\}_{k=1}^K$ the corresponding centers. Similarly, Conv(Ω^k) and Conv$(\Omega^{k'})$ are disjointed for any k and k'.

Theorem 1. *Without the loss of generality, we assume that the diameter of set Ω is bounded by 1. We have*

$$f_\Omega(\mu_S, C_S) - f_\Omega(\mu_\Omega, C_\Omega) \leq \mathcal{O}\left(\sqrt{\frac{K\log N + \log\frac{1}{\delta}}{n}}\right) \tag{5}$$

holds with a probability greater than $1 - \delta$.

Proof. From the definition of (μ_S, C_S), we have

$$f_S(\mu_S, C_S) \leq f_S(\mu_\Omega, C_\Omega). \tag{6}$$

First we have from the Hoeffding inequality

$$\mathbb{P}\left(|f_S(\mu_\Omega, C_\Omega) - f_\Omega(\mu_\Omega, C_\Omega) \geq \epsilon\right) \leq 2\exp\{-n\epsilon^2\}, \tag{7}$$

[2]As shown by A.1 in [10], the two sampling models with and without replacement become equivalent, as $N \to \infty$ asymptotically.

because $f_\Omega(\mu_\Omega, C_\Omega)$ can be viewed as the mean of $|\Omega|$ i.i.d. samples, and $f_S(\mu_\Omega, C_\Omega)$ is the sample mean:

$$f_\Omega(\mu_\Omega, C_\Omega) = \frac{1}{N}\sum_{i=1}^{N} y_i \quad f_S(\mu_\Omega, C_\Omega) = \frac{1}{n}\sum_{x_i \in S} y_i,$$

where $y_i = (x_i - \mu_\Omega^{k(i,C_\Omega)})^2$, and $k(i, C_\Omega)$ indicates the cluster index for sample $i \in \{1, 2, \cdots, N\}$ with respect to partition C_Ω. Next, we clarify the combinatorial nature of the hypothesis space. In the 1-D setting, a valid k-discretization corresponds strictly to partitioning the sorted data indices into K contiguous intervals. Consequently, the set of all possible optimal partitions is discrete and finite. The cardinality of this hypothesis space is bounded by $\binom{N-1}{K-1}$, which is upper bounded by $\binom{N}{K}$. This allows us to apply the union bound over all valid partitions:

$$\begin{aligned}
&\mathbb{P}\left(|f_S(\mu_S, C_S) - f_\Omega(\mu_S, C_S)| \geq \epsilon\right) \\
\leq &\mathbb{P}(|f_S(\mu, C) - f_\Omega(\mu, C)| \geq \epsilon, \exists(\mu, C) \text{ with } K \text{ partitions over } \Omega) \\
\leq &\binom{N}{K}\mathbb{P}\left(|f_S(\mu, C) - f_\Omega(\mu, C)| \geq \epsilon, \text{ fixed } (\mu, C)\right) \\
\leq &\binom{N}{K} 2\exp(-n\epsilon^2) \\
\leq &2\exp(-n\epsilon^2 + 2K\log N),
\end{aligned} \tag{8}$$

where the second inequality uses the union bound. Combining (6), (7), and (8), we obtain the claim. □

Recall $n = sN$, the right side of (5) is re-expressed as $\mathcal{O}\left(\sqrt{\frac{K}{s}\cdot\frac{\log N}{N} + \frac{\log\frac{1}{\delta}}{sN}}\right)$. Theorem 1 states the fact that the approximation error between the SnC solution and the global optimum is reduced as s increases. A less intuitive, but even more important fact is that *the regardless of s, the error vanishes as $N \to \infty$ asymptotically*. That backups SnC as a solid strategy, in particular for large-scale data.

3. Empirical Validation: Simulation on 1-D Signal Quantization

We first conduct a preliminary experiment on one-dimensional data. The goal of this simulation is two-fold: to empirically verify the global optimality and stability of the algorithm from Section 2.1, in contrast to heuristic baselines such as k-means; and to validate the effectiveness of the SnC acceleration strategy.

We generate a set of 10 random signals $\in \mathbb{R}^{256}$, with elements sampled i.i.d. from $N(0, 1)$. We quantize each signal into k discrete levels, k ranging from 2 to 200, using k-means, the more robust k-medoids, and the proposed 1-D k-discretization approach, respectively. Since both k-means and k-medoids are initialization-sensitive, we repeat running them 20 times with different random initializations. We measure the *quantization error* as the Euclidean distance between the original signal and the solution. The quantization errors of k-means and k-medoids are averaged from all 20 runs of 10 signals. We also calculate the variances of k-means and k-medoids quantization errors, by averaging the per-signal variances of 20 runs over all signals.

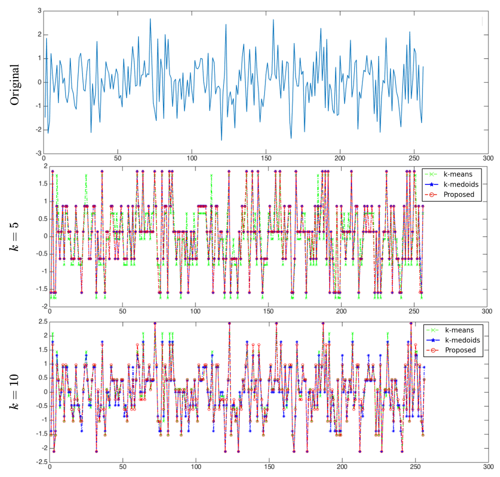


Figure 1: (a) A sample of real-valued signals; (b) the quantized signals from (a) at $k = 5$, using k-means, k-medoids and the proposed method, respectively; (c) the quantized signals from (a) at $k = 10$, using k-means, k-medoids and the proposed method, respectively.

The upper row of Figure 1 presents an example of the 100 signals, while the following two compare its quantization signals at $k = 5$ and 10. It is evident that both k-means and k-medoids reach local minima that are different from the global optimum by the proposed method. One can visually observe that the result of

k-means at $k = 5$ is problematic: several of its peaks appear to be distorted, as the original signal has only moderate magnitudes at those locations.

3.1. Comparison with k-means and k-medoids

Figure 2 (a) plots the *relative quantization errors* with respect to k, defined as the ratio between the k-means (or k-medoids) quantization errors and the error of the proposed method. Such relative errors are always larger than 1 and grow monotonically with k. Figure 2 (b) shows that both k-means and k-medoids suffer from large variances due to different initializations, especially with relatively small k. In contrast, the result variance of the proposed approach is a sharp zero even running repeatedly. Both plots back up the global optimality conclusion in Section 2, and confirm that our solver is indeed optimal and stable.

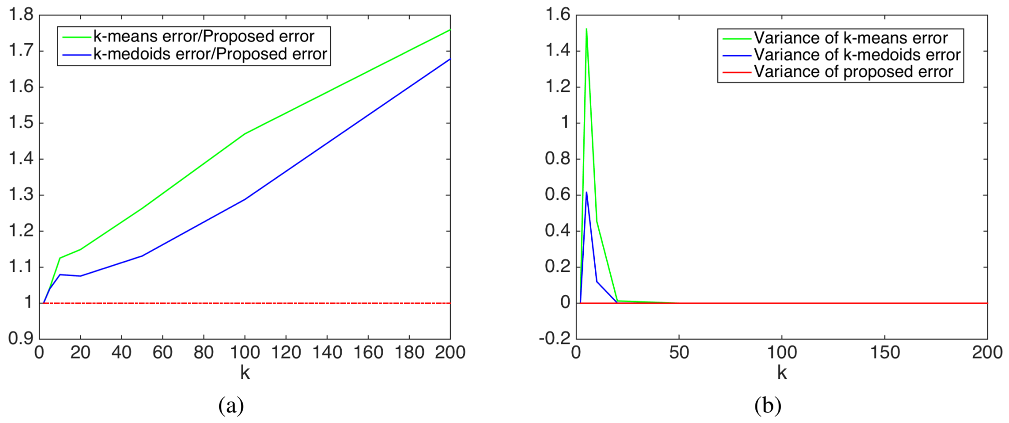


Figure 2: (a) the relative quantization error w.r.t. k, averaged from 100 signals and 20 runs; (b) the error variances of 20 runs w.r.t. k, averaged from 100 signals.

3.2. Experiments on Sampling-then-Clustering

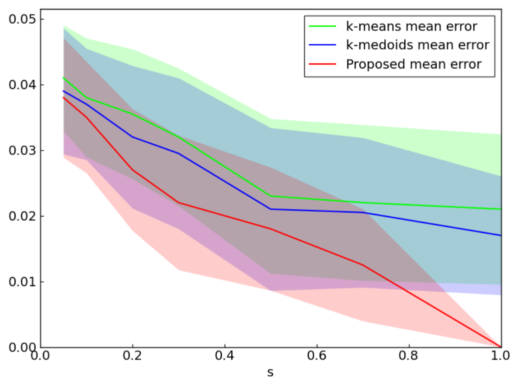
We re-generate a set of 100 random signals $\in \mathbb{R}^{1000}$ and fix $k = 10$. The sample rate s is varied from 0.1 to 1, with an interval of 0.1. We again run each algorithm (with SnC) for 20 times per sample. In each run, we not only vary the initialization of k-means and k-medoids throughout 20 runs, but also re-sample $\mathbf{x}_s$.

For each signal, let E_{SnC} denote the quantization error of the SnC quantized signal, and E_m denote the minimum quantization error at s = 1. We define the *relative SnC error* as $(\frac{E_{SnC}-E_m}{E_m})^2$. For three comparison methods, We measure their mean relative SnC errors (averaged from 100 signals and 20 runs), and the variance of relative SnC errors throughout 20 runs (averaged from 100 signals), at different s values. Figure 3 (a) demonstrates that under the SnC setting, the relative error of the proposed approach decreases much faster and more steadily to zero. Figure 3 (b) plots the actual average running time t (seconds) of our Matlab implementation at different s values, which grows quadratically with s and fits the theoretical complexity well. These results validate SnC as an effective and reliable acceleration method. Having established the correctness and efficiency of our core building block, we now proceed to apply it as a parsimonious regularizer in high-dimensional settings.

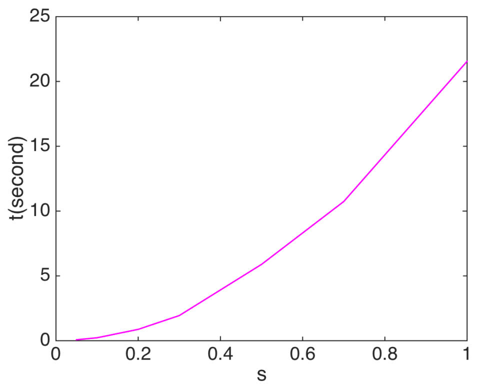
4. Multi-Dimensional Extension by Embedding Learning: Evaluation on Clustering

Clustering learns the hidden data patterns and group similar structures in an unsupervised way, with many classical algorithms proposed. Since many high-dimensional data exhibit dense grouping in lower-dimensional embeddings [11], researchers have been motivated to first project the original data into a low-dimensional subspace, and then cluster on the feature embeddings [12]. Lately, Tian et al. [13] explored the possibility of employing deep network embeddings in graph clustering. They first learned a nonlinear embedding of the original graph by an autoencoder (AE), followed by a k-means algorithm on the embedding to obtain the final clustering result. Trigeorgis et al. [14], Wang et al. [15] developed various more sophisticated architectures and performed task-driven optimization.

The clustering problem is a typical instance for *multi-dimensional k-discretization*, *i.e.,* quantizing a set of vectors into k different vector-valued prototypes. We project the high-dimensional samples space to 1-D as



(a)



(b)

Figure 3: (a) The mean relative SnC errors and the standard deviation w.r.t. s, averaged from 100 signals and 20 runs; (b) the running time t (seconds) w.r.t. s, averaged from 100 signals and 20 runs.

a form of ultimate parsimonious regularization. This forces the network to find the single most dominant discriminative axis for the data. The rationale for projecting to 1-D goes beyond simple dimensionality reduction: it enforces a strict constraint that encourages the neural network to learn a nonlinear mapping that "unfolds" the high-dimensional data manifold into a linearly separable order. We name this as *k-discrete embedding* (or **kDE**). Assume $\mathbf{X} \in \mathbb{R}^{p \times n}$ denoting the set of p-dimensional samples X_i ($p > 1$), $i = 1, 2, ..., n$, to be partitioned into k clusters, and $z \in \mathbb{R}^n$. kDE aims to solve the following problem:

$$\min_{\mathcal{W},\mathbf{z}} \|\mathbf{\Psi}_{\mathcal{W}}(\mathbf{X}) - \mathbf{z}\|_F^2 \quad s.t., \|\mathbf{z}\|_{d0} <= k, \tag{9}$$

where $\mathbf{\Psi}_{\mathcal{W}}$ denotes the *embedding function* parametrized by $\mathcal{W}$, to project each $X_i \in \mathbb{R}^p$ into its k-discrete embedding $z_i \in \mathbb{R}$. From a clustering perspective, (9) can be interpreted as learning an end-to-end mapping from a feature vector to its cluster assignment. Related ideas were found in the literature such as label-consistent dictionary learning [16], but the numerical cluster "label" z in (9) is unknown (with the number of different classes k as the only prior) and has to be learned jointly with $\mathbf{\Psi}_{\mathcal{W}}$, which adds to the difficulty.

The choice of $\mathbf{\Psi}_{\mathcal{W}}$ can vary depending on the application needs. Here we design a hierarchical alternating combination of linear and non-linear transformations, which resembles a $2d$-layer auto-encoder, in the form below (γ is a scalar):

$$\begin{aligned} \min_{\{\mathbf{W}_1,...,\mathbf{W}_{2d}\},\mathbf{z}} & \|\sigma(\mathbf{W}_d \cdot ...\sigma(\mathbf{W}_2 \cdot \sigma(\mathbf{W}_1\mathbf{X}))..) - z\|_F^2 \\ & +\gamma\|\sigma(\mathbf{W}_{2d}^T \cdot ...\sigma(\mathbf{W}_{d+2}^T \cdot \sigma(\mathbf{W}_{d+1}^T z))..) - \mathbf{X}\|_F^2 \\ & s.t., \quad \|\mathbf{z}\|_{d0} <= k. \end{aligned} \tag{10}$$

σ is the sigmoid function to introduce nonlinearity. $\{\mathbf{W}_1, ..., \mathbf{W}_d\}$ successively project $\mathbb{R}^p$ to $\mathbb{R}$, while $\{\mathbf{W}_{d+1}, ..., \mathbf{W}_{2d}\}$ transform $\mathbb{R}$ back to $\mathbb{R}^p$. We set $\mathbf{W}_t$ and $\mathbf{W}_{2d+1-t}$ to be of the identical dimension, $t = 1, ..., d$, so that (10) becomes a "symmetric" autoencoder-type model.

The objective in (10) simultaneously minimizes the reconstruction error from z to $\mathbf{X}$, as well as the quantization error of z. The former ensures that the embedding z largely preserves useful information of $\mathbf{X}$ after the dimensionality reduction, avoiding trivial solutions such as $z = 0$. The latter may recall the utility of a "bottleneck layer" in classical stacked autoencoders [17] for nontrivial feature extraction. The difference lies in the new k-discrete regularization on z, which is based on the global distribution of $\mathbf{X}$.

We solve (10) using a coordinate descent algorithm on the $2d + 1$ variables: in each iteration, updating each variable while fixing the other $2d$ ones. The $2d$ $\mathbf{W}_t$ subproblems all have differentiable objectives and can be minimized by gradient descent. As for the z subproblem, we first split the variable by adding an equality constraint $z - z' = 0$. It is then disentangled to a 1-D k-discretization problem, plus a minimization over a differentiable objective. Based on the empirical fact that the coordinate descent algorithm usually converges well without exactly solving each subproblem, we accelerate the solution of (10) via updating each $\mathbf{W}_t$ subproblem for a small fixed number (*e.g.*, 3) of steps per iteration. We also apply the SnC strategy to the 1-D k-discretization problem. An efficient practice is to start at $s = 0.01$, gradually increasing s by 1.25 times each iteration until it reaches 1. γ is chosen by default to be 1.

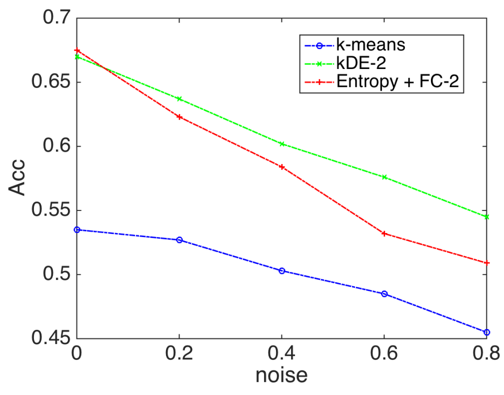
We evaluate the effectiveness of kDE by comparing the following methods, on the tasks of clustering two classical benchmarks, *MNIST* [18] and *YaleB* [19]: (1) **k-means** applied directly over the original data; (2) **kDE-1**: solving (10) with $d = 1$, *i.e.*, $\mathcal{W} = \{\mathbf{W}_1, \mathbf{W}_2\}$, $\mathbf{W}_2 \in \mathbb{R}^{1\times p}$; (3) **kDE-2**: solve (10) with $d = 2$, *i.e.*, $\mathcal{W} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4\}$, with $\mathbf{W}_1, \mathbf{W}_4 \in \mathbb{R}^{p_1\times p}$, $\mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{1\times p_1}$. p_1 is fixed as 128 for both datasets; (4) **k-means + AE-1**: train a two-layer AE [20] only with the mean square error (MSE) loss, then apply k-means on the first hidden layer features. We tried to train AE with two layers of $p \times 1$ and $1 \times p$ respectively, only to see terrible clustering performance, which highlights that the 1-D bottleneck is only effective when trained jointly with the k-discretization objective, as our kDE model does. We instead re-configure the two layers to be $p \times 10$ and $10 \times p$ respectively, and apply dropout with the rate 0.5 on the hidden layer. Such a model has much more reasonable performance and is adopted by us; (5) **k-means + AE-2**: train a four-layer AE under MSE, whose layers have dimensions $p \times p_1$, $p_1 \times 10$, $10 \times p_1$ and $p_1 \times p$, respectively, and then apply k-means on the features of the second hidden layer. A dropout rate of 0.5 is enforced; (6) **Entropy + FC-1**: train a two-layer fully-connected (FC) network *from end to end* with clustering entropy loss [15], which can be viewed as the unsupervised counterpart of the softmax loss. The layers have identical dimensionality with k-means + AE-1; and (7) **Entropy + FC-2**: training a four-layer FC network *from end to end* with entropy loss, whose layers have identical dimension with k-means + AE-2.

We measure the accuracy (Acc) and normalized mutual information (NMI) of the clustering results and present them in Table 1. We can observe that increasing the "depth" benefits all AE-type methods. kDE-2 performs best on the YaleB dataset and is marginally inferior to Entropy + FC-2 on the MNIST dataset. Both kDE-2 and Entropy + FC-2 outperform k-means + AE-2 notably, which demonstrates the superiority of joint optimization of feature embedding and cluster assignment over the separate pipeline.

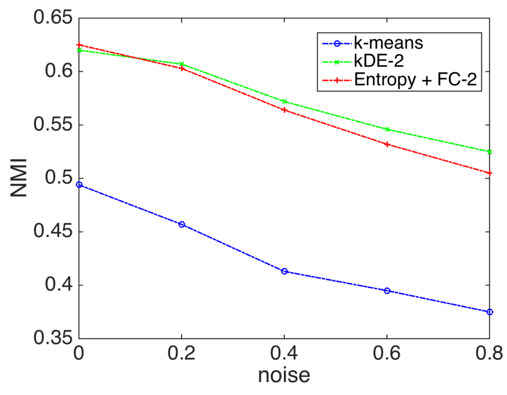
Table 1: Accuracy and NMI performance comparisons on MNIST and YaleB datasets.

	Metric	k-means	kDE-1	kDE-2	k-means + AE-1	k-means + AE-2	Entropy + FC-1	Entropy + FC-2
MNIST	Acc	0.535	0.532	0.670	0.443	0.481	0.622	**0.675**
	NMI	0.494	0.511	0.620	0.505	0.545	0.581	**0.629**
YaleB	Acc	0.721	0.727	**0.882**	0.731	0.834	0.781	0.864
	NMI	0.707	0.755	**0.893**	0.743	0.852	0.801	0.877

We hypothesize that this strong parsimonious embedding, by forcing the model to discard high-frequency noise, will lead to significant robustness to data corruption, which we validate in our experiments. Figure 4 manifests how adding Gaussian noise to the input data will affect the accuracy and NMI performance of k-means, kDE-2 and Entropy + FC-2, on the MNIST dataset. While kDE-2 lags slightly behind Entropy + FC-2 on the "clean" MNIST dataset, it has been proven to be a more stable embedding and quickly surpasses the method of Entropy + FC-2 as the data get increasingly corrupted by noise.



(a)



(b)

Figure 4: (a) The accuracy comparison with respect to noise levels; (b) the NMI comparison with respect to noise levels, on the MNIST dataset. Note that images are normalized between $[0, 1]$. The x-axis denotes the variance of the added (zero-mean) gaussian noise.

5. Evaluation on Image Smoothening by Combining Total Variation Regularization

We now demonstrate the modularity and power of the one-dimensional k-discrete prior by showing how it can be combined with other well-known regularizers.

Color quantization [21] is a useful technique that reduces the number of distinct colors in an image. This is helpful for displaying color images on devices limited by memory such as IoT devices, or for efficient image compression. It is straightforward to correlate the proposed k-discretization model with the color quantization task. However, color quantization can neither suppress noise nor accurately remove details, usually yielding incorrect boundaries in the quantized image. Xu et al. [22] proposed a relevant but more challenging *image smoothening* task that aims to enhance the fundamental components of images, *i.e.,* the salient edges, while diminishing insignificant details without hampering overall acuity. Image smoothening is found to benefit compression-artifact suppression, edge extraction, detail enhancement, HDR tone mapping. In this section, we examine how the k-discretization method can be adapted for the image smoothening task.

For an RGB image $\in \mathbb{R}^{m_1 \times m_2 \times 3}$, image smoothening could be treated as a clustering task on the $m_1 m_2$ samples $\in \mathbb{R}^3$, with additional edge and saliency preserving requirements. It is possible to apply the kDE model (9) to learn a mapping from $\mathbb{R}^3$ to $\mathbb{R}$ with the k-discrete constraint. Here we adopt a much simpler approach: first we convert the RGB image into the YUV format, then perform the 1-D k-discretization algorithm only on the Y channel (vectorized into $\mathbb{R}^{m_1 m_2}$). The U and V channels are quantized on the basis of the same partition of the Y channel.

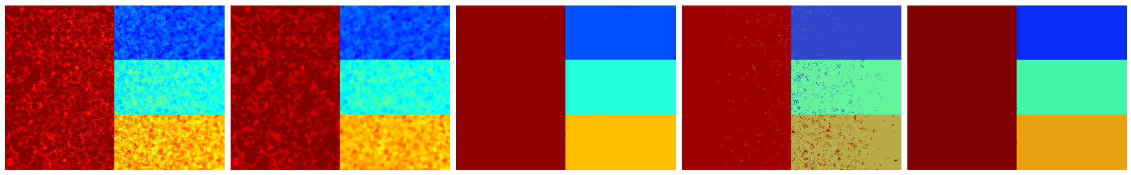

(a) (b) (c) (d) (e)

Figure 5: (a) Original *stripes jets* image; (b) TV-ℓ_1 model; (c) ℓ_0 gradient model [22]; (d) naive k-discrete model (k = 4); (e) TV-k-discrete smoothening (k = 4, λ = 0.7)

We start by looking at the synthetic *stripes jets* image, a piece-wise constant image contaminated with intensive noise in Figure 5. (b) and (c) show results of two representative methods: the total variation (TV)-ℓ_1 decomposition model [23], and the ℓ_0 gradient model [22][3]. The TV-ℓ_1 model suppresses the noise and enhances the local piece-wise continuity, but fails to give a holistically smoothened image. The ℓ_0 gradient model achieves the visually smooth result by globally selecting and maintaining the most prominent set of edges and increasing the steepness of the transition. Figure 5 (d) displays the result by applying our basic model (1) directly to the vectorized image with $k = 4$. Correctly restoring most pixels thanks to the strong k-discrete prior, the basic model also leaves many noise outliers, due to the absence of consideration for spatial pixel coherences. That motivates us to develop the following **TV-k-discrete smoothening** model that unites the local spatial-preserving properties of total variation with the global parsimony of our k-discrete constraint:

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_{TV} + \lambda \|\mathbf{Z} - \mathbf{X}\|_F^2 \quad s.t., \|\text{vec}(\mathbf{Z})\|_{d0} \leq k \tag{11}$$

where $\mathbf{X}, \mathbf{Z} \in \mathbb{R}^{m_1 \times m_2}$ are the input and output (Y channel) images, and vec$(\mathbf{Z})$ converts $\mathbf{Z}$ into an $\mathbb{R}^{m_1 m_2}$ vector. $\|\mathbf{Z}\|_{TV}$ is the classic local prior that encourages piecewise constant regions and preserves sharp edges, and $\lambda \|Z - X\|_F^2$ is the standard data fidelity term that pulls the solution back towards the original image. Most importantly, $\|\text{vec}(\mathbf{Z})\|_{d0} <= k$ is the global prior that enforces a parsimony budget on the entire image, forcing it to use no more than k unique intensity levels. To solve Equation (11), we can again follow a coordinate descent algorithm, since the projection operators for both the TV norm and the 1-D k-discrete constraint are well developed. We apply the same SnC strategy to solve the kDE problem in Equation (10) to the $\mathbf{Z}$ subproblem to speed up training. Figure 5. (e) shows the clean result of applying (11) with λ = 0.7.

We then compare the TV-k-discrete smoothening model (11) with the competitive baseline [22] on more natural images, where k *cannot be exactly pre-specified.* Figure 6 presents the comparison results on the

[3]The comparison models operate on the original RGB channels. All hyperparameters are set by following the descriptions of original papers.

Beach image: while small k values (50, 100) cause obvious color blockiness in the sky and beach regions, increasing k to 500 and 1,000 greatly alleviates visual artifacts and gives rise to comparable visual results to Xu et al. [22]. We also notice the wrong color on the tree trunk part at $k = 50$ and 100 (quantized to the same color as the sky), which is later corrected in the $k = 500$ and 1000 results. Similar phenomena are observed on the *Lady* image in Figure 7: although the face is incorrectly assigned by the same color with the grass at $k = 200$ and 500, it gets gradually refined when k increases. This demonstrates the direct, predictable effect of the global k-discrete prior. At a low k (e.g., 200), the model is forced to 'save' a color by merging the small, distinct region (the face) with a larger, dominant region (the grass). As the budget k increases to 800, the model gains the freedom to assign a dedicated, correct value to the face. It is also noteworthy to observe that while the result of Xu et al. [22] preserves sharper details on the dress than ours, it is accompanied by undesirable blockiness in the sky and also leaves more insignificant details unattended in the grass.

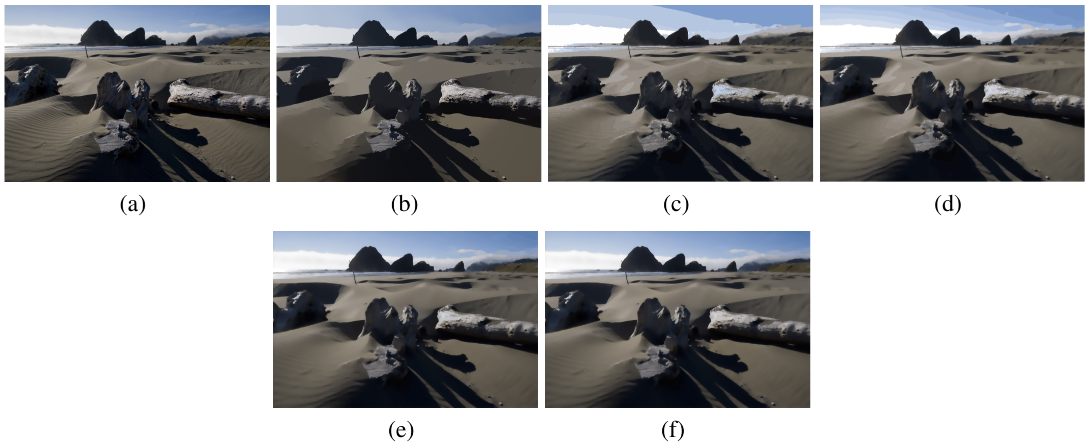


Figure 6: (a) Original *beach* image; (b) ℓ_0 gradient model [22]; (c) TV-k-discrete smoothening (k = 50, λ = 0.5); (d) TV-k-discrete smoothening (k = 100, λ = 0.5); (e) TV-k-discrete smoothening (k = 500, λ = 0.5); (f) TV-k-discrete smoothening (k = 1000, λ = 0.5).

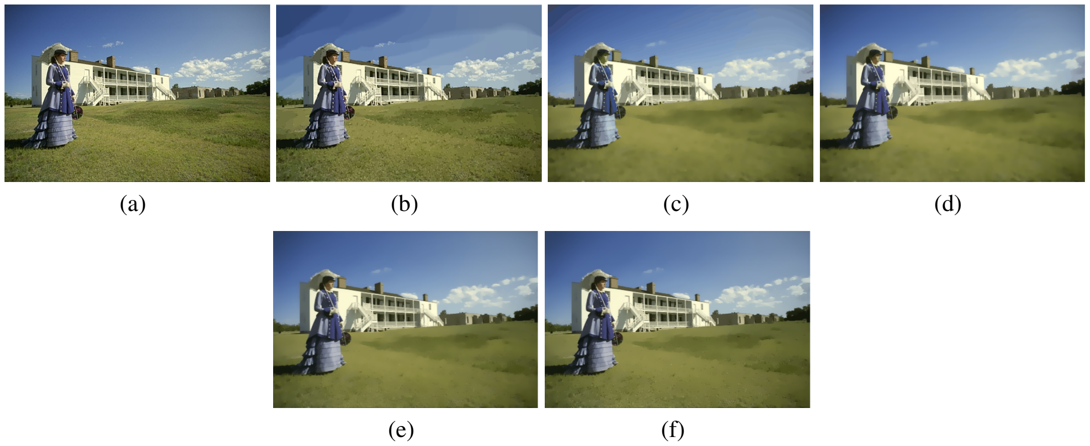


Figure 7: (a) Original *lady* image; (b) ℓ_0 gradient model [22]; (c) TV-k-discrete smoothening (k = 200, λ = 0.5); (d) TV-k-discrete smoothening (k = 500, λ = 0.5); (e) TV-k-discrete smoothening (k =800, λ = 0.5); (f) TV-k-discrete smoothening (k =1000, λ = 0.5).

Unlike other smoothening methods that produce an unconstrained, continuous-color output, our TV-k-discrete model provides an explicit and interpretable palette size (k) that directly controls the parsimony of the output. The results from Xu et al. [22] have used drastically more unique colors. In contrast, our method allows for the trade-off between k and the quality, implying its great potential for color quantization and palette coding [21]. The TV-k-discrete smoothening model uniquely combines the advantages of image smoothening and color quantization. On the one hand, it extracts faithful principal structures and preserves salient edges, proving to be a powerful option with results comparable to the seminal work [22]. However, such visually competitive results are obtained with an explicit control of the number of unique colors in the results, which is beyond the scope of Xu et al. [22].

5.1. Limitations and Failure Analysis

While the strict 1-D constraint is effective for many structures, we explicitly analyze failure scenarios where the data distribution is topologically complex. For instance, distributions with nested structures or complex boundaries often cannot be easily mapped onto a single axis without breaking continuity or losing local coherence.

We illustrate this limitation using Figure 8. As shown in the figure, our model fails to discriminate the salient leaf in the foreground from background variations of similar colors. This is likely because the boundaries between the leaf and background are not strictly defined by global intensity separation, but rather by local textural or chromatic contrast that the 1-D projection collapses. Despite this, we note that the result at $k = 100$ still achieves a PSNR of 26.37 dB, remaining visually competitive while using significantly fewer unique colors than baselines.

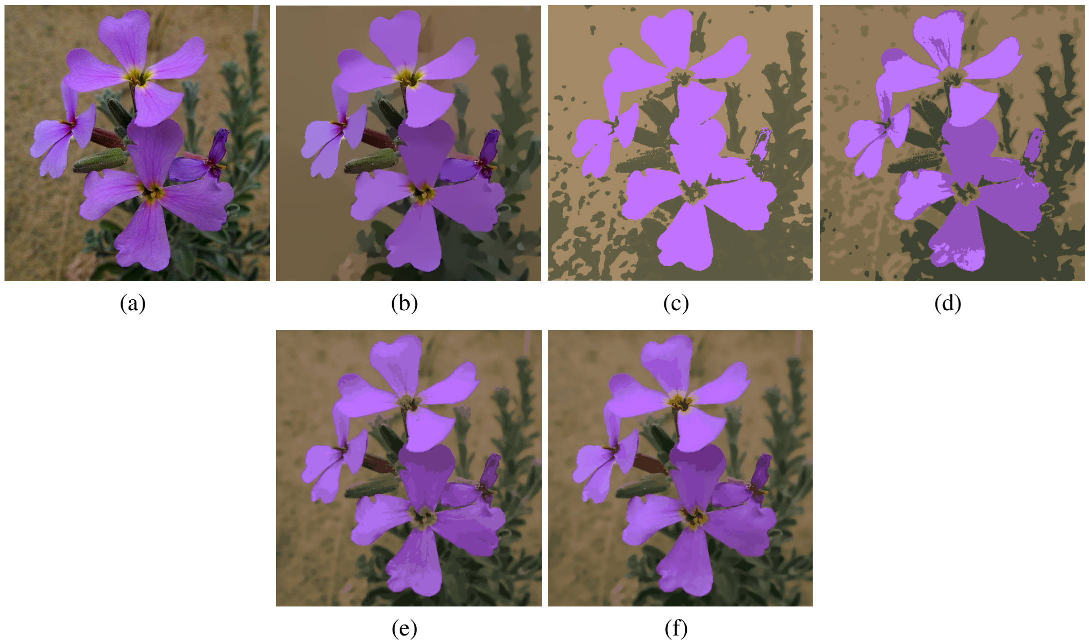


Figure 8: (a) Original *purple flower* image; (b) ℓ_0 gradient model [22]; (c) TV-k-discrete smoothening (k = 5, λ = 0.5); (d) TV-k-discrete smoothening (k = 10, λ = 0.5); (e) TV-k-discrete smoothening (k = 50, λ = 0.5); (f) TV-k-discrete smoothening (k = 100, λ = 0.5).

The restriction to the Y-channel ensures polynomial-time global optimality but assumes that the luminance aligns with structural boundaries. We identify two failure modes: (1) Iso-luminant Regions, where distinct hues with similar intensity are merged (e.g., the 'Flower' in Figure 8); and (2) High-Frequency Textures, which are smoothed out during 1-D projection. Future work may address this via joint YUV discretization, trading computational efficiency for chromatic sensitivity.

6. Conclusion

This paper investigates how the general NP-hard problem of k-discretization could be solved to the global optimum in the 1-D case, how the algorithm could be accelerated by sampling with bounded errors, and how multi-dimensional cases could be handled by embedding learning. With many task-specific modifications, we demonstrate that our proposed k-discretization approach serves as a powerful, modular, and parsimonious regularizer and achieves highly promising results in a variety of applications, including 1-D signal quantization, image clustering, and image smoothening.

References

[1] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[2] Ulrike Von Luxburg et al. Clustering stability: an overview. *Foundations and Trends® in Machine Learning*, 2(3):235–274, 2010.

[3] Marina Meil**u**a. The uniqueness of a good optimum for k-means. In *Proceedings of the 23rd international conference on Machine learning*, pages 625–632. ACM, 2006.

[4] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[5] Paul S Bradley, Olvi L Mangasarian, and W Nick Street. Clustering via concave minimization. *NIPS*, pages 368–374, 1997.

[6] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

[7] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[8] Haizhou Wang and Mingzhou Song. Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. *The R journal*, 3(2):29, 2011.

[9] Yilun Wang, Junfeng Yang, Wotao Yin, and Yin Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[10] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[11] Feiping Nie, Dong Xu, Ivor W Tsang, and Changshui Zhang. Spectral embedded clustering. In *IJCAI*, pages 1181–1186, 2009.

[12] Bin Cheng, Jianchao Yang, Shuicheng Yan, Yun Fu, and Thomas S Huang. Learning with l1 graph for image analysis. *IEEE TIP*, 19(4), 2010.

[13] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, pages 1293–1299, 2014.

[14] George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Bjoern Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pages 1692–1700, 2014.

[15] Zhangyang Wang, Shiyu Chang, Jiayu Zhou, Meng Wang, and Thomas S Huang. Learning a task-specific deep architecture for clustering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 369–377. SIAM, 2016.

[16] Zhuolin Jiang, Zhe Lin, and Larry S Davis. Label consistent k-svd: Learning a discriminative dictionary for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2651–2664, 2013.

[17] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel. Extracting deep bottleneck features using stacked auto-encoders. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3377–3381. IEEE, 2013.

[18] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist dataset of handwritten digits. *URL http://yann. lecun. com/exdb/mnist*, 1998.

[19] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001.

[20] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[21] Michael T Orchard and Charles A Bouman. Color quantization of images. *IEEE transactions on signal processing*, 39(12):2677–2690, 1991.

[22] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. Image smoothing via l 0 gradient minimization. In *ACM Transactions on Graphics (TOG)*, volume 30, page 174. ACM, 2011.

[23] Wotao Yin, Donald Goldfarb, and Stanley Osher. The total variation regularized l^1 model for multiscale decomposition. *Multiscale Modeling & Simulation*, 6(1):190–211, 2007.

A. Algorithms

We present the algorithm for solving the 1-D k-Discretization problem in Algorithm 2.

Algorithm 2 Optimal 1-D k-Discretization via Dynamic Programming

```
Input: Data vector $x \in \mathbb{R}^n$, number of levels $k$.
Output: Discretized vector $z \in \mathbb{R}^n$ minimizing $||z - x||^2$ s.t. $||z||_{d0} \leq k$.
 1: Sort $x$ such that $x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(n)}$.
 2: Initialize DP table $T \in \mathbb{R}^{n \times k}$.
 3: Pre-compute costs: Calculate quantization error for all segments $(i, j)$.
 4: for $m = 1$ to $n$ do
 5:     $T(m, 1) \leftarrow \sum_{j=1}^{m} (x_{(j)} - \text{mean}(x_{(1):x_{(m)}}))^2$
 6: end for
 7: DP Recursion:
 8: for $r = 2$ to $k$ do
 9:     for $m = r$ to $n$ do
10:         $T(m, r) \leftarrow \min_{r-1 \leq i < m} \left\{ T(i, r-1) + \sum_{j=i+1}^{m} (x_{(j)} - \text{mean}(x_{(i+1):x_{(m)}}))^2 \right\}$
11:         Store index $i^*$ that yields the minimum for backtracking.
12:     end for
13: end for
14: Backtracking: Reconstruct optimal partition boundaries from stored indices.
15: Assign mean values of optimal partitions to sorted $z$.
16: Restore original order of $z$ using sorting indices.
17: return $z$
```

B. Proof of Global Optimality for the Dynamic Programming Algorithm

The global optimality of DP can be proven by deduction. It is evident to see that (3) provides the global optimum for $r = 1, \forall m$. Now assume that the global optima of $T(i, r - 1), i = 1, ..., m - 1$ are known. To show that (4) guarantees the global optimality of $T(m, r)$, we observe that when x_m is appended to the sorted $\mathbf{x}_{m-1}$, since x_m is larger than all elements in $\mathbf{x}_{m-1}$, it can only be partitioned with a subsequence of the largest elements $(x_{i+1}, \ldots, x_{m-1}), i \leq m - 1$. The update rule of $T(m, r)$ must take the form

$$T(m, r) = \min_{r-1 \leq i \leq m-1} E_{\mathcal{P}_{r-1}}(i, r-1) + \sum_{j=i+1}^{m} \left(x_j - \frac{1}{m-i} \sum_{j=i+1}^{m} x_j \right)^2$$

$E_{\mathcal{P}_{r-1}}(i, r - 1)$ denotes the discretization loss of $\mathbf{x}_i$ on some $(r - 1)$-partition $\mathcal{P}_{r-1}$. As the last step, it suffices to show that $E_{\mathcal{P}_{r-1}}(i, r - 1)$ = $T(i, r - 1)$ is the necessary condition for the above minimum to be taken. That is evident, since according to the definition, $T(i, r - 1) \leq E_{\mathcal{P}_{r-1}}(i, r - 1), \forall \mathcal{P}_{r-1}$ with $r - 1$ partitions over $\mathbf{x}_i$.

As the projection operator (1) can be exactly solved to the global optimum, we can derive more complicated optimization models, that rely on (1) as a subproblem.

C. More Experiments

C.1. Architecture Ablation and Design Guidelines

In this section, we provide a detailed analysis of the architectural choices for the kDE embedding network, specifically focusing on the activation function and bottleneck width, to offer clearer guidance for future practitioners.

Activation Function Analysis The choice of the activation function in Equation 10 is critical for the numerical stability of the coordinate descent algorithm. We conducted ablations comparing the standard sigmoid function against ReLU and tanh:

1. **sigmoid**: By bounding the output embedding z to the range $(0, 1)$, the sigmoid function prevents the values from exploding during the iterative 1-D k-discretization updates. This bounded range also naturally aligns with image data normalized to $[0, 1]$.
2. **ReLU**: We observed that using ReLU activations in the bottleneck layer frequently led to optimization instability and lower performance.
3. **tanh**: While Tanh is bounded $(-1, 1)$, it introduces negative values that complicate the interpretation of the 1-D embedding as a strictly additive intensity metric, which is often desirable in image reconstruction tasks.

Bottleneck Width and Feature Separability The dimensionality of the layer immediately preceding the 1-D projection (*i.e.*, the "bottleneck width") plays a pivotal role in unfolding the data manifold. As noted in the main text, a naive linear projection or a shallow bottleneck (*i.e.*, width equal to 1) failed to capture the topological structure of the data. Internal experiments with bottleneck widths of 2 and 4 similarly yielded inferior clustering performance compared to our adopted width of 10.