Kernel Net for Clustering

Chieh Wu, Zulqarnain Khan, Yale Chang, Stratis Ioannidis, Jennifer Dy
Electrical and Computer Engineering Dept., Northeastern University, Boston, MA

Abstract

We propose Kernel Net (KNet), which combines deep neural networks with kernels, to automatically learn the kernel in spectral clustering using only a subset of the data. The resulting KNet formulation can also be interpreted through Hilbert Schmidt Independence Criterion as learning a non-linear feature mapping that approximates the spectral clustering embedding. This allows us to apply this non-linear transformation on out-of-sample data, where data in the new space can be partitioned by k-means clustering. Exploiting this property, KNet is capable of clustering while training on a significantly smaller subset of data. To determine the sufficient amount of samples required to generalize KNet, we further propose a pseudo-metric based on the L_∞ norm of the difference of normalized eigenvalues between two unequal sized kernels. Experiments on synthetic and real data show that KNet outperforms competing methods in terms of clustering performance even when trained on only a subset of samples.

1 Introduction

Clustering is the process of grouping similar samples together based on some notion of similarity. Spectral clustering is a flexible algorithm capable of discovering arbitrarily shaped clusters [1]. In spectral clustering, pairwise similarity between samples is usually defined through kernels. However, because the choice for an appropriate kernel is data-dependent, the kernel design process is often an art that requires intimate knowledge of the data. A common alternative is to simply use a general-purpose kernel that performs well under various conditions (e.g., polynomial or Gaussian kernels).

In this paper, we seek to automatically learn the kernel in spectral clustering; instead of pre-defining it as in standard spectral clustering. Inspired by the success of deep neural networks (DNNs) in learning complex representations for many applications (e.g., object and speech recognition), we combine the advantages of DNNs with kernels. This provides us with a deep kernel representation, capable of representing an infinite mapping (through a Gaussian kernel) of highly adaptive basis functions (through DNN). By incorporating a DNN parameterization to a Gaussian kernel, we are able to learn a flexible deep kernel for clustering. We call our algorithm *Kernel Net (KNet).*

Interestingly, optimizing for a DNN parameterization of a Gaussian kernel with a spectral clustering objective can be alternatively interpreted from the perspective of the Hilbert Schmidt Independence Criterion [2] as learning a non-linear transformation of the input that approximates the spectral clustering embedding. This allows us to apply this non-linear transformation to out-of-sample data, leading to compact clusters in the new space that can be partitioned by k-means clustering.

Constructing a Kernel often involves $O(N^2)$ memory and computational complexity, where N is the number of samples. We propose a strategy for sub-sampling n samples $\ll N$ using a pseudo-metric based on the L_∞ norm of the difference of normalized eigenvalues between two unequal sized kernels (of the subset and the complete set) to find the subset sufficient to approximate the full kernel. Because KNet learns a non-linear transformation that can be applied to out-of-sample data, it can simply apply this learned transformation on the rest of the data. As an example of this ability, Figure 1 demonstrates the clustering process with KNet. In this example, KNet was trained on

Preprint. Work in progress.

one percent of the data shown in **A**. The resulting DNN transforms the data into linearly separable clusters shown in **B**. The complete set in **E** is then passed through the trained network to create a new linearly separable embedding in **F**. The similarity matrices are shown below to demonstrate how KNet generates perfectly block diagonal structures.

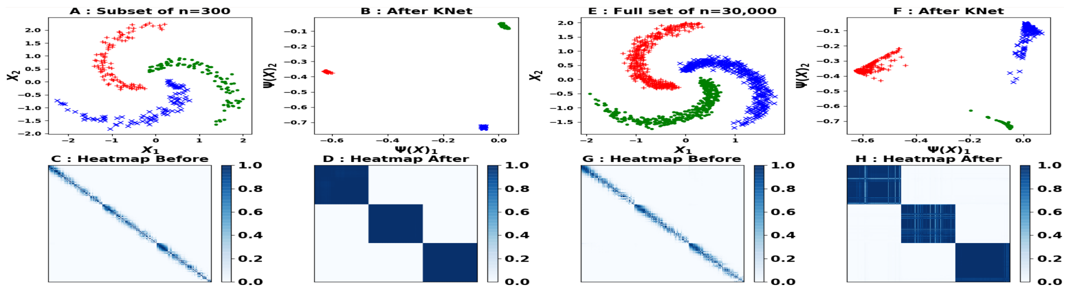

Figure 1: Using only one percent of the data for KNet, 100% of the data as $\Psi(X)$ can achieve a linearly separable representation.

In summary, the contributions of this paper are:

- We propose Kernel Net, a DNN for discovering kernels suitable for spectral clustering using only a subset of the data.
- We propose a pseudo-metric to quantify the distance between two unequal sized kernels to enable finding a subset of samples sufficient to approximate the full kernel.
- We extensively evaluate the performance of KNet with synthetic and real data. Empirical results show that KNet outperforms competing methods in terms of clustering performance.

Related Work. An earlier approach to unsupervised kernel discovery was demonstrated by Niu et al. [3, 4]. They learn a kernel by learning a linear projection of the data as input to a kernel. However, a linear projection lacks the flexibility of DNNs. Applying DNNs to discover kernels has been applied in supervised settings. Wilson et al. [5, 6, 7] combined the flexibility of deep neural nets with kernels in Gaussian processes.

In unsupervised settings, instead of discovering a kernel, the goal of the clustering methods involving DNNs concentrates on leveraging the DNN to discover a nonlinear feature mapping that leads to good clustering. Several of these methods utilize a DNN in the form of an autoencoder (AE), where the output of the encoder is then used with a clustering objective. In one of the earlier approaches, Song et al. [8] embed a k-means objective with the reconstruction objective of an AE, simultaneously learning a non-linear feature mapping and the clustering labels based on this objective. Ji et al. [9] make use of a similar approach by combining the objective of Subspace Clustering as introduced in [10] with an AE objective. Motivated by Spectral Clustering, Tian et al. [11] use the output of the deepest layer of a Sparse AutoEncoder trained on a pre-computed similarity matrix of the input data as the embedding to run k-means on , but unlike Kernel Net their objective itself is not directly related to spectral clustering and their similarity matrix (kernel) is fixed.

On the other hand, some techniques pursue a different approach. Xie et al. [12] presented the Deep Embedding for Clustering (DEC), a self-training method where it iteratively transforms the data via the most confident samples to improve the assignments, they achieve this by minimizing the KL Divergence between a Student's t-distribution of soft cluster assignments and an auxiliary distribution which is designed to reinforce confident assignments from previous iterations. The DEC objective was later extended to handle image data by Guo et al. [13]. In 2017, Hu et al. [14], presented yet another approach to DNN clustering. Borrowing the adversarial approach, their main idea is to encourage that the predictions from a Neural Network remain robust to data augmentation, by penalizing representation dissimilarity between the original and augmented data, while at the same time maximizing the mutual information between data and its representation.

2 Kernel Net

Given a data set $X = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^T$ where $\boldsymbol{x}_i \in \mathbb{R}^d$, n and d are the number of samples and features respectively. Our goal is to cluster the data samples into c groups and to automatically learn the kernel defining pairwise similarity between samples simultaneously. Our solution is to cluster the data using spectral clustering. However, instead of a pre-defined kernel (as in standard spectral clustering), we automatically learn the kernel. In particular, learning the kernel with our suggested parameterization through a spectral clustering objective enables us to also simultaneously learn the nonlinear embedding $\psi(\boldsymbol{x})$ for mapping out-of-sample $\boldsymbol{x}$ to a new space. This leads to compact clusters that can be partitioned by k-means clustering. In this section, we start by providing a review of spectral clustering. Then we introduce our proposed Kernel Net formulation. We explain how our Kernel Net formulation also allows us to learn a nonlinear embedding. We then describe how we initialize Kernel Net and finally describe our solution to the Kernel Net optimization problem.

Review of Spectral Clustering. Given a data set X and a similarity matrix K with elements $K_{i,j}$ generally obtained from a Gaussian kernel where $K_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2))$. The goal of spectral clustering [1] is to partition the data into c disjoint groups or partitions, such that the similarity of the samples *between groups* is low, and the similarity of the samples *within groups* is high. There are several objective functions that capture this property; one version used in spectral clustering [15] is the *normalized association (Nassoc)* objective. Let $U \in \mathbb{R}^{n \times c}$ be a cluster indicator matrix, where element $u_{ij} = 1$ means that sample $\boldsymbol{x}_i$ belongs to cluster j and $u_{ij} = 0$ means that sample $\boldsymbol{x}_i$ does not belong to cluster j. In spectral clustering, a combinatorial optimization problem is converted into a continuous optimization problem by relaxing the indicator matrix to allow its entries to take on any real value, leading to the following relaxed *Nassoc* optimization problem:

$$U^* = \arg\max_U \operatorname{Tr}(U^T D^{-1/2} K D^{-1/2} U), \quad \text{s.t. } U^T U = I, \tag{1a}$$

where the degree matrix $D \in \mathbb{R}^{n \times n}$ is defined as $D = Diag(\mathbf{1}_n^T K)$ with $\mathbf{1}$ as a column vector of length n filled with all 1s. The solution to this optimization problem is to set the c most dominant eigenvectors of the matrix $D^{-1/2} K D^{-1/2}$ to U where U is the learned spectral embedding. The discrete partitioning of the data is obtained from a "rounding" step; this follows [15], which renormalizes each row of U to a unit length and then applies k-means to the rows of the normalized matrix; $\boldsymbol{x}_i$ is then assigned to the cluster that row u_i is assigned to.

Kernel Net (KNet) Formulation. In spectral clustering, the similarity matrix (kernel) is assumed known and the goal is to learn the cluster assignments. In this paper, we would like to automatically learn the kernel as well. We propose to learn the kernel by parameterizing the kernel $K_{X;\theta}$ as a Gaussian kernel of a flexible nonlinear function $\psi(\theta)$; i.e., $K_{X;\theta} \in \mathbb{R}^{n \times n}$ is a matrix with elements:

$$K_{i,j} = e^{-\frac{1}{2\sigma^2}||\psi(\boldsymbol{x}_i;\theta) - \psi(\boldsymbol{x}_j;\theta)||^2}. \tag{2}$$

Eq. (2) is motivated by the observation that when the raw data is not informative enough, a Gaussian kernel applied directly to $\boldsymbol{x}$ may not be sufficient to capture the clustering similarity structure. It would be beneficial to automatically learn the data representation that maximizes the clustering quality. Since DNN has been widely used to automatically learn data representations for various supervised learning tasks, we propose $\psi(\theta)$ to be a DNN to transform the raw data into a suitable feature representation. We then form our kernel as a composition of a Gaussian kernel with $\psi(\theta)$ to ensure $K_{X;\theta}$ as a valid kernel with values of each element between 0 and 1. We thus propose Kernel Net (KNet) to simultaneously learn the kernel (by learning θ) and the cluster embedding U by optimizing:

$$[U^*, \theta^*] = \arg\max_{U,\theta} \operatorname{Tr}(U^T H D^{-1/2} K_{X;\theta} D^{-1/2} H U), \quad \text{s.t. } U^T U = I. \tag{3a}$$

This objective follows the spectral clustering objective of Ng et al. [15], where the kernel is assumed to be normalized by the degree matrix, D. In addition, we assume that the kernel after normalization is centered to 0 by a centering matrix H. $H \in \mathbb{R}^{n \times n}$ is a symmetric centering matrix defined as $H = I - \frac{1}{n} 1_n 1_n^T$ where $I \in \mathbb{R}^{n \times n}$, is an identity matrix.

Alternative Interpretation. While Eq. (3a) can be interpreted as discovering a $K_{X;\theta}$ optimal for Spectral Clustering, there is another interpretation, i.e., the feature map of KNet is approximating the embedding U of Spectral Clustering. Namely, let $\Psi(X;\theta) = [\psi(x_1), .., \psi(x_n)]^T$ and U as defined previously, Eq. (3a) is simultaneously approximating U with $\Psi(X;\theta)$. This is because Eq. (3a) can be rewritten as the maximization of the Hilbert Schmidt Independence Criterion (HSIC) [2] between $\Psi(X;\theta)$ and U such that

$$[U^*, \theta^*] = \arg\max_{U,\theta} \operatorname{HSIC}(\Psi(X;\theta), U). \tag{4}$$

The original proof is in [4] but it is also provided in the Appendix A for completeness. HSIC measures the dependency between two random variables through kernels without having the need to explicitly estimate joint probability distributions [2]. By maximizing the relationship between $\Psi(X;\theta)$ and U, KNet also learns a mapping from X directly to cluster embedding U using $\Psi(X;\theta)$. Therefore, a trained KNet can be utilized to cluster samples outside the training subset (out-of-sample data) by applying a nonlinear transformation $\Psi(X;\theta)$ followed by k-means clustering to obtain the clustering results.

2.1 Initialization and Kernel Net Architectures.

Our Kernel Net objective Eq. (3a) is nonconvex with multiple local optima. Thus it is imperative to provide a good initialization. In this subsection, we describe initializing U and θ respectively.

Initializing U. We apply a Gaussian kernel directly on input X and set U equal to the c most dominant eigenvectors of $HD^{-1/2}KD^{-1/2}H$.

Initializing θ. We utilize a deep neural network to represent $\psi(x_i;\theta)$. As such, this function representation is very flexible. We would like $\psi(x_i;\theta)$ functions that are good representations of X. To obtain this representation, the key is to leverage the appropriate DNN architecture that can recover the original data during initialization. From this, we investigated three architectures for KNet: a multi-layer perceptron (MLP) reconstructing the input, Autoencoders (AE), and Convolutional Autoencoder (CNN-AE) [13], as shown in Figure 2. The MLP serves as our most basic DNN which is suitable for low dimensional input data. We explored autoencoders to enable the handling of high dimensional data and convolutional autoencoders to deal with image data. We initialize θ by pretraining all three DNNs with a reconstruction loss (i.e., $\theta^* = \arg\min_\theta \frac{1}{N}\sum_i^N (x_i - f(x_i;\theta))^2$). Then, we let the $\psi(x_i;\theta)$ in Eq. (3a) equal to the $\psi(x_i;\theta)$ shown in the figure corresponding to the encoder portion of autoencoders and $\psi(x_i;\theta) = f(x_i;\theta)$ for the MLP network. Note that this initialization constrains ψ by linking it to local minima near the reconstructor (or autoencoder) of the data.

2.2 Optimization.

We initialize U and θ as described above and solve objective Eq. (3a) by alternating minimization between U and θ, as described in detail in the next two sections. We iterate until the c largest eigenvalues of $HD^{-1/2}KD^{-1/2}H$ converge. The KNet algorithm is summarized in Algorithm 1.

Algorithm 1: Kernel Net Algorithm

```
Input  : dataset $X \in \mathbb{R}^{N\times d}$, or $X \in \mathbb{R}^{N\times w\times h}$
         A pretrained KNet with weights $\theta$
Output : Feature mapping $\Psi(X)$
         spectral clustering embedding $U$
Initialize $U_0$, $D_0$ using Spectral Clustering
while ($U$ not converged) do
    Update $\theta$ by solving Eq. (5)
    Update $D$
    Update $U$ by solving Eq. (6a)
Normalize each row of $U$ to 1
Run $k$-means on $U$
```

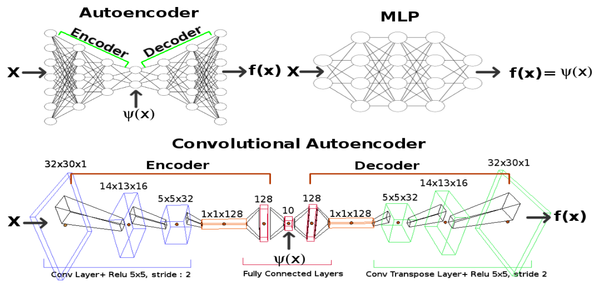

Figure 2: KNet Architectures with X as the input and $\Psi(X)$ as the feature mapping.

Optimization of $\boldsymbol{\theta}$. Holding U constant, consider the matrix $Y = Y^T \equiv D^{-1/2}HUU^THD^{-1/2}$ with elements $Y_{i,j}$, $i,j \in \{1,\ldots,n\}$. Then, the objective in Eq. (3a) becomes

$$\theta = \arg\max_{\theta} Tr(YK_{X;\theta}) = \arg\max_{i,j} \textstyle\sum_{i,j} Y_{i,j} e^{-\gamma||\psi(x_i;\theta)-\psi(x_j;\theta)||^2}. \tag{5}$$

We solve this optimization sub-problem by stochastic gradient descent using back-propagation on mini-batches of size q (default $q = 5$ in our experiments). As the back-propagation iterates through each epoch to improve Eq. (5), it is simultaneously pulling similar samples into a clump while pushing dissimilar samples apart. This process is demonstrated in Figure 3 as it plots out the effect of back-propagation on the spiral dataset at various epochs. To provide an intuitive understanding of

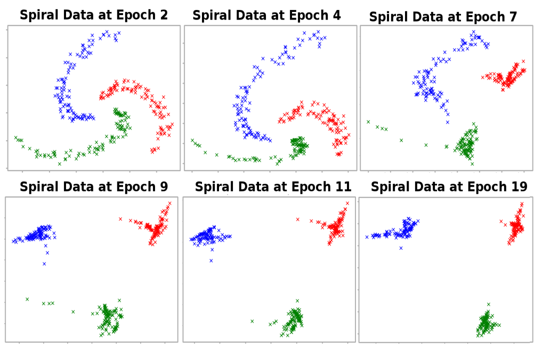

Figure 3: The process of learning the Kernel at various Epoch of SGD. It can be seen that data of the same clusters are being pulled together.

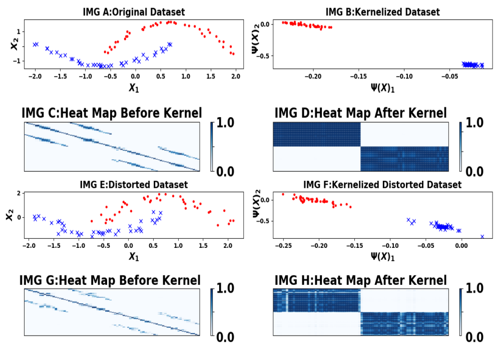

Figure 4: Gaussian noise is added to the original data to see if the kernel maps the data into linearly separable representations.

how each sample moves mechanically, Appendix H provides a simple example to demonstrate the relationship between Eq. (5) and the effect of back-propagation on each sample.

Optimization of U. Holding θ constant implies $L_X = HD^{-1/2}K_{X;\theta}D^{-1/2}H$ is also a constant, the formulation for optimizing U becomes

$$U = \underset{U,\theta}{\arg\max}\ \mathrm{Tr}(U^T L_X U), \quad \mathrm{s.t}: U^T U = I. \tag{6a}$$

All matrices that are constant are combined into a single matrix, L_X. The formulation reduces down to a Rayleigh quotient [16]. The standard approach to optimize this objective is to apply eigendecomposition to L_X and use the c most dominant eigenvectors as U. Note that since U is bounded on a Stiefel manifold, and the elements of $K_{i,j}$ is bounded between [0,1] by the Gaussian kernel, the objective resides in a bounded space. Since the iterations of optimizing U and θ monotonically increases the objective, the Monotone convergence theorem [17] states that such a sequence converges.

Clustering. We discretize U by normalizing each row to the unit norm, then applying k-means on each row of the normalized U provides us the clustering solution for the samples in our training set. To cluster out-of-sample (test) data, we project each test data, x_{test}, to a non-linear embedding space, $\psi(x_{test};\theta)$, then apply k-means on this transformed space.

3 Sample Subset Selection

As demonstrated by [15], a major advantage of Spectral Clustering over other algorithms is its ability to cluster datasets with manifold assumptions, e.g., datasets shown in Figures 1 and 4. However, because the algorithm includes a construction and an eigendecomposition of a kernel, it has a computational and memory complexity of $O(N^2)$. Using Eq. (4), KNet solves this problem by approximating the embedding, and as a result, also bypassing these expensive operations on out-of-sample data. The advantage comes from the fact that only a subset of the samples is required to train and generalize the kernel. Once a KNet is trained, the complete set of samples can be passed into the feature map of KNet to retrieve the labels by applying k-means to the approximated embedding. The advantage of this approach is demonstrated in Figure 1, where the computation for a kernel of size $30,000^2$ is required. The construction of a kernel of this magnitude poses a serious challenge in terms of both time and memory. For this reason, the utilization of a representative subset is crucial in overcoming the challenge. While there are clear advantages to using a subset, the challenge is to find a representative subset to generalize KNet. From this motivation, this section describes the subset selection process for KNet.

Let's assume there exists a score $\mathbb{D}(.,.)$ such that it measures the distance between two positive semi-definite matrices of unequal sizes. Given the full dataset $\mathbb{S}$ of size N and a subset $\mathbb{A} \subset \mathbb{S}$ of size $n < N$ we can measure how close $\mathbb{S}$ is to $\mathbb{A}$ by calculating $\mathbb{D}$ between the kernel matrices $K_{\mathbb{S}} \in \mathbb{R}^{N\times N}$ and $K_{\mathbb{A}} \in \mathbb{R}^{n\times n}$ of the two sets. If $\mathbb{D}$ is below some pre-defined threshold δ, we propose to use $\mathbb{A}$ to train KNet. The pseudo-code of the algorithm finding the subset $\mathbb{A}$ is outlined in Algorithm 2, and we include implementation details in Appendix I. Having obtained a subset $\mathbb{A}$ from

Algorithm 2, the initialization and the entire KNet algorithm is performed on the smaller set $\mathbb{A}$. Once Ψ has been trained, the mapping can be applied to the complete dataset.

Algorithm 2: Subset Selection

```
Input  : dataset $\mathbb{S} \in \mathbb{R}^{N \times d}$, threshold = $\delta$
         sample repetition $J$, sample increment $\Delta$
Output : $\mathbb{A} \in \mathbb{R}^{n \times d}$
Init   : $n = n_0$
while ($\mathbb{D} \geq \delta$ and $n < N$) do
    $itr = 0$
    while ($itr < J$) do
        Sample $\mathbb{A}^{itr}$ from $\mathbb{S}$ uniformly at random
        Compute $K_\mathbb{S}$ and $K_\mathbb{A}^{itr}$
        Compute $\mathbb{D}^{itr} = D(K_\mathbb{S}, K_\mathbb{A}^{itr})$
    $\mathbb{D} = \text{mean}(\mathbb{D}^{0:J})$
    $i = \text{argmin } \mathbb{D}^{0:J}$
    $\mathbb{A} = \mathbb{A}^i$
    $n = n + \Delta$
```

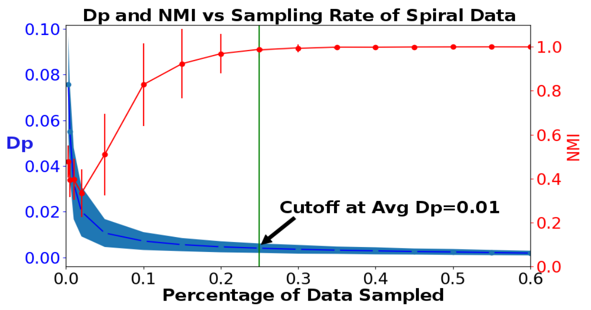

Figure 5: $\mathbb{D}_p$ and NMI versus percentage of data sampled.

We now present a possible choice for $\mathbb{D}$ based on the eigenvalues of $K_\mathbb{A}$ and $K_\mathbb{S}$. These Kernel matrices are created using the Gaussian Kernel to match the initialization of KNet. A standard way to approximate a matrix is to use its low-rank approximation by applying Eckart–Young–Mirsky theorem [18]. This theorem suggests that the difference between the eigenvalues of two matrices can quantify the distance between matrices. However, when the size of the matrices is different, the eigenvalues are no longer scaled proportionally. From this perspective, theories from Principal Component Analysis (PCA) [19] and Kernel PCA (KPCA) [20] provide insight into how the eigenvalues can be scaled. Based on KPCA, the eigenvalues of the centered kernel are related to the variances along each Principal Component (PC) scaled by the number of samples. Therefore, If the kernel eigenvalues are normalized to the range of $[0, 1]$ the scaling difference of the eigenvalues is eliminated. Hence, the difference between normalized eigenvalues is capable of comparing kernel matrices of different sizes. A visualization of how the variances of the PCs capture the shape of the data can be found in Appendix D. We use these observations to formally propose our choice for $\mathbb{D}$.

Let $K_\mathbb{A}$ and $K_\mathbb{S}$ be the centered Gaussian kernels of $\mathbb{A}$ and $\mathbb{S}$ and let the m largest eigenvalues of $K_\mathbb{A}$ be $\boldsymbol{\lambda}_\mathbb{A} = [\lambda_\mathbb{A}^{(1)}, ..., \lambda_\mathbb{A}^{(m)}]$, where $\lambda_\mathbb{A}^{(1)} \geq ... \geq \lambda_\mathbb{A}^{(m)}$, and the m largest eigenvalues of $K_\mathbb{S}$ be $\boldsymbol{\lambda}_\mathbb{S} = [\lambda_\mathbb{S}^{(1)}, ..., \lambda_\mathbb{S}^{(m)}]$, where $\lambda_\mathbb{S}^{(1)} \geq ... \geq \lambda_\mathbb{S}^{(m)}$, such that m is determined by p percentage of the total variance ($\sum_{i=1}^{N} \lambda_\mathbb{S}^{(i)}$) we wish to preserve. We define the distance between $K_\mathbb{A}$ and $K_\mathbb{S}$ as

$$\mathbb{D}_p(K_\mathbb{A}, K_\mathbb{S}) = \left\| \frac{1}{\|\boldsymbol{\lambda}_\mathbb{A}\|_1} \boldsymbol{\lambda}_\mathbb{A} - \frac{1}{\|\boldsymbol{\lambda}_\mathbb{S}\|_1} \boldsymbol{\lambda}_\mathbb{S} \right\|_\infty, \tag{7}$$

where $\|\cdot\|_1$ and $\|\cdot\|_\infty$ are L_1 and L_∞ norms respectively. Having provided a general justification for using $\mathbb{D}_p$ as a measure of distance between $\mathbb{A}$ and $\mathbb{S}$, we provide a more formal justification as implied by the following two properties with their proofs in Appendix B, and C.

Property 1. *$\mathbb{D}_p$ is a pseudo-metric*

Property 2. *At $p = 100\%$, $\mathbb{D}_p(K_\mathbb{A}, K_\mathbb{S}) = 0$ if and only if the variances along their PCs are equal.*

Following the sampling procedure from Algorithm 2, Figure 5 plots out the subset sampling process for the Spiral data when we set $|\mathbb{S}| = 1200$. In the figure, a dataset is sampled at various sizes between 0 to 60% of set $\mathbb{S}$. For each size, it is repeatedly sampled 100 more times to compute the average $\mathbb{D}_p$, as well as its standard deviation (std). For demonstration purposes, each sampled subset is used by KNet to generalize clustering to $\mathbb{S}$. The results are measured with normalized mutual information (NMI) against the ground truth. This is also repeated 100 times for each size to compute the mean and standard deviation of NMI. It can be seen in the figure that as $\mathbb{D}_p$ decreases, the NMI increases logarithmically. The improvement saturates quickly at a sampling rate of 25%, with a $\mathbb{D}_p$ of 0.01. This figure demonstrates that the saturation of $\mathbb{D}_p$ corresponds inversely to the saturation of NMI.

4 Experiments

We first examine KNet's ability to generate representations suitable for clustering. Second, we validate $\mathbb{D}_p$ as an appropriate pseudo-metric to determine a subset's ability to approximate the complete set.

These two contributions can be tested by training the kernels from a subset determined by $\mathbb{D}_p$ and then applying the kernel to a complete set. By comparing the KNet clustering quality against competing techniques, we can objectively infer the relative quality of the kernel for clustering. The second contribution can then be tested by comparing the clustering quality of the subset against the results of the complete set. If a generalized mapping is legitimately learned, the kernel should extend this mapping to the complete set without having seen the data. Therefore, if $\mathbb{D}_p$ predicted an appropriate subset, the clustering quality of the subset should mirror the quality of the complete set. Furthermore, we also include experiments subjecting the other DNN clustering techniques to only train on a subset of the data, and attempt to generalize the DNN to the complete set. With this, we demonstrate that KNet is superior in retaining the clustering quality for generalization.

Evaluation Metric. As suggested by [21], the experiments use the Normalized Mutual Information (NMI) to compare clustering allocations against the ground truth. The NMI is a similarity measure confined within the range of [0,1] with 0 denoting no relationship and 1 as a perfect match. If we let Z_i and Z_j be two clustering assignments, NMI can be calculated with $NMI(Z_i, Z_j) = I(Z_i, Z_j)(H(Z_i)H(Z_j))^{-1/2}$, where $I(Z_i, Z_j)$ is the mutual information between Z_i and Z_j, and $H(Z)$ computes the entropy of Z. Although NMI is sufficient to assess the correctness of the labels, we also include the accuracy measure (ACC) [12] in Appendix for completeness.

Hyperparamter Settings. The architectures of the network can be MLP, Autoencoding, or Convolutional. All hidden layers use rectified linear units as activation functions except for the output of the KNet where no activation function is used. All KNets are optimized by Adam [22] with a learning rate of 0.001, and a mini-batch of 5. The code is written in Pytorch. Since these hyperparameters are fixed, 3 hyperparameters remain for tuning, i.e., the width of the hidden layers, the depth of the network and the σ value inside the Gaussian kernel. These values are determined through a grid search that optimize Eq. (3a) using only the subset. In Appendix E in the supplement, we demonstrate that objective Eq. (3a) on a subset is highly correlated to the NMI of the complete set.

Datasets. The experiment includes 6 datasets, 2 synthetics and 4 real. They have been chosen carefully to exhibit various aspects of KNet and cover a variety of clustering problems. Each dataset is split into subset and complete sets. Except for the Moon dataset, the size of the subset is determined by the point where $\mathbb{D}_p \approx 0.01$. The kernel is learned from the subset and applied to the complete set. The two synthetic datasets are the Spiral and the Moon shown in Figure 1 and 4. Because both can only be clustered through manifold patterns, they are excellent for validating KNet's ability to map data into representations suitable for k-means. The Spiral dataset shown in Figure 3 visually demonstrates the effect of back-propagation during kernel creation. In our experiments, it is also used to demonstrate its ability to generalize patterns when only 1% of the samples were used. The purpose of the Moon dataset is to demonstrate how KNet performs when noise is added to the subset.

For the real datasets, the goal is to include a wide variety of internal structure and sources. Both the Breast Cancer [23] used in [24] and the Wine datasets [25] have standard structures in the form of $X \in \mathbb{R}^{N \times d}$ with corresponding sizes of 683 and 178 respectively. With a different internal structure, the RCV dataset [26] is text data where the features are in the form of tf-idf. Because of its high dimension, PCA was used to keep 80% of the variance during the preprocessing stage. The dataset had 10,000 samples and only 6% was required to achieve a low $\mathbb{D}_p$. The Face dataset [27] allows us to complete the convolutional architecture. It includes 640 images of size 32x30 pixels from 20 people and is directly passed into the Convolutional structure without any preprocessing. However, since the benchmark techniques are incapable of handling image data, the images are vectorized and compressed by PCA down to 80% of the total variance.

Benchmark Method. The clustering results of KNet will be compared against 5 benchmark methods. k-means and Spectral Clustering are included as a baseline. The technique by Song et al. [8], which we will refer to as AEC, is included as an early work of clustering using DNN with an autoencoder. The DEC algorithm by Xie et al. [12] is included because it is a recent work that didn't follow the dominant approach. The IMSAT by Hu et al. [14] is included because it is the most recent and advanced DNN clustering technique. The hyperparameters of the competing methods are set to the default values provided in the respective papers. Since the input of KNet is a pretrained autoencoder, the output of the autoencoder clustered by k-means (AE+Kmeans) is also recorded to demonstrate how KNet improves upon this initial point.

Run times. Table 2 reports the run time of the DNN benchmark methods and KNet. For KNet, the stages are broken into subset sampling (sampling), autoencoder pre-training (Autoencoder), and the actual training time. For the benchmark DNN methods, their times are broken into pre-processing and the run time where pre-processing includes calculating nearest neighbor distances for IMSAT, pre-training for DEC, and pre-training and fine-tuning for AEC. The execution times indicate that run times for the complete KNet pipeline are comparable to the competing methods. Further details about the settings for the run time experiments can be found in Appendix G.

Results. The clustering results for both the complete (top) and subset (bottom) in NMI are shown in Table 1. The complete table shows that KNet has outperformed the benchmark techniques in terms of NMI while trained on only a subset of the data. Since KNet handles manifold patterns, the improvement is especially significant compared to other DNN clustering techniques. Since AE+Kmeans is the starting point of KNet, it can be seen that KNet further improves the result. The performance difference between MLP, Autoencoder, and CNN provides a justification for having different architectures. By noticing the similarity in clustering quality between the subset and the complete set of KNet, we conclude that the subset was appropriately chosen by $\mathbb{D}_p$, and therefore, a generalizable kernel is learned. By comparing the subset and the complete set (right side of /), we see that other DNN techniques are also capable of generalizing results similar to KNet, however, KNet's training set is a superior predictor of the complete set with results even better than competing models that are trained on the complete set of samples (left side of /).

Table 1: Benchmark comparisons with $100\times$NMI. There are two results for AEC, DEC, and IMSAT methods. The left side of the slash is the result trained on the set $\mathbb{S}$. The right side is trained on $\mathbb{A}$ and generalized to $\mathbb{S}$ similar to KNet. During training, all 3 architectures are tested and the best architecture is used for KNet. The last column of the training row details the percentage of data suggested by $\mathbb{D}_p$.

Complete	AEC	DEC	IMSAT	KMeans	Spectral	AE+Kmeans	KNet
Moon	56.2 ± 0.0 / 51.05 ± 0.0	42.2 ± 0.0 / 39.2 ± 0	51.3 ± 20.3 / 45.5 ± 17.3	42.3 ± 0.0	72 ± 0.0	39 ± 0.0	**100 ± 0.0**
Spiral	28.3 ± 0.0 / 22.34 ± 0.0	32.02 ± 0.01 / 16.0 ± 0.0	59.6 ± 7.5 / 48.8 ± 7.3	41.9 ± 0.0	**100 ± 0.0**	42.0 ± 0.0	**100 ± 0.0**
Cancer	79.9 ± 0.2 / 79.9 ± 0.0	79.2 ± 0.0 / 45.7 ± 0.0	74.6 ± 2.2 / 74.9 ± 2.7	73.2 ± 0.4	69.8 ± 0.0	73 ± 0.0	**84.2 ± 0.4**
Wine	54.6 ± 0.01 / 52.8 ± 0.1	80.6 ± 0.0 / 85.0 ± 0.0	72.3 ± 11.4 / 70.8 ± 15.3	87 ± 0.5	88 ± 0.0	87.6 ± 0.0	**97.0 ± 0.1**
RCV	39.3 ± 0.0 / 27.95 ± 0.0	51.3 ± 0.0 / 41.88 ± 0.0	39.0 ± 5.5 / 35.16 ± 3.9	55.8 ± 2.0	41.0 ± 0.2	52.0 ± 0.0	**56.7 ± 0.1**
Face	76.8 ± 0.0 / 54.8 ± 2.5	75.8 ± 1.6 / 66.6 ± 2.2	83.8 ± 3.5 / 77.8 ± 1.5	83.7 ± 1.4	66.0 ± 0.4	78.0 ± 0.0	**84 ± 3**
Subset	AEC	DEC	IMSAT	MLP	AutoEncoder	CNN	Data %
Moon	51.05 ± 0.0	45.6 ± 0.0	45.5 ± 17.3	**100 ± 0.0**	51 ± 2.0	N/A	100%
Spiral	32.2 ± 0.0	49.5 ± 0.0	48.7 ± 7	**100 ± 0.0**	62 ± 10	N/A	1%
Cancer	76.4 ± 0.000	76.9 ± 0.0	74.9 ± 2.6	.83 ± 0.0	**85 ± 0.1**	N/A	30%
Wine	49.1 ± 0.10	81.5 ± 0.0	69.4 ± 9.47	82.5 ± 1	**94 ± 0.3**	N/A	75%
RCV	26.78 ± 0.0	43.23 ± 0.0	35.2 ± 3.9	**56 ± 2**	53.9 ± 1	N/A	6%
Face	52.5 ± 1.2	67.1 ± 2.0	77.8 ± 1.5	87 ± 0.0	86.8 ± 1	**85 ± 2**	35%

Table 2: Time for various stages of Kernel Net against DNN benchmark methods. (s = seconds, m = minutes, h = hours)

	Kernel Net Time				AEC Time		DEC Time		IMSAT Time	
	$\mathbb{D}_p$	Sampling *	Autoencoder	Training	Preprocess	Run time	Preprocess	Run time	Preprocess	Run time
Moon	N/A	N/A	28.5 s	41.2 s	18.23 s	28.04 s	332 s	3.52 s	1.2 s	34.34 s
Spiral	0.003	42.4 s	55 s	4 m	42.9 m	2.2 h	5.73 m	2.02 m	53.06 s	5.15 m
Cancer	0.008	8 s	19 m	63 s	1.65 m	4.67 m	343.71 s	38.73 s	1.1 s	141.5 s
Wine	0.01	1.5 s	11.3 m	74.6 s	33.81 s	41.69 s	5.75 m	38.02 s	1.1 s	33.32 s
RCV	0.009	3.2 h	28.7 m	10.9 m	2.36 m	42.28 m	6.35 m	13.07 m	10.39 s	1.22 m
Face	0.003	51 m	2.1 m	21 m	27.5 s	3.15 m	5.74 m	4.235 m	1.2 s	2.03 m

$*$During the Sampling stage, a grid search is conducted on multiple nodes simultaneously at various sampling percentages. Since they are running in parallel, the longest time would be the time which n is large enough such that the average $\mathbb{D}_p \approx 0.01$. Therefore, the time recorded during the sampling stage uses the slowest time among the nodes. Similarly, during Autoencoder pretraining stage, multiple autoencoders are also running simultaneously. The autoencoder that yields the lowest error is selected.

5 Conclusions

KNet performs unsupervised kernel discovery using only a subset of the data. By discovering a kernel that optimizes the Spectral Clustering objective, it simultaneously discover an approximation of its embedding through a DNN. Furthermore, experimental results has confirmed that KNet can be trained using only a subset of the data determined by $\mathbb{D}_p$, the output of KNet yields suprerior clustering quality, and that the kernel is generalizable to the full set.

References

[1] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[2] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[3] Donglin Niu, Jennifer Dy, and Michael Jordan. Dimensionality reduction for spectral clustering. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 552–560, 2011.

[4] Donglin Niu, Jennifer G Dy, and Michael I Jordan. Iterative discovery of multiple alternativeclustering views. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1340–1353, 2014.

[5] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

[6] Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016.

[7] Andrew Gordon Wilson, David A Knowles, and Zoubin Ghahramani. Gaussian process regression networks. *arXiv preprint arXiv:1110.4411*, 2011.

[8] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Iberoamerican Congress on Pattern Recognition*, pages 117–124. Springer, 2013.

[9] Hongdong Li Mathieu Salzmann Ian Reid Pan Ji, Tong Zhang. Deep subspace clustering network. In *Advances in Neural Information Processing Systems*, 2017.

[10] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

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

[12] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[13] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International Conference on Neural Information Processing*, pages 373–382. Springer, 2017.

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

[15] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[16] Roger A Horn, Roger A Horn, and Charles R Johnson. *Matrix analysis*. Cambridge university press, 1990.

[17] James Yeh. *Real Analysis: Theory of Measure and Integration Second Edition*. World Scientific Publishing Company, 2006.

[18] Moody T Chu, Robert E Funderlic, and Robert J Plemmons. Structured low rank approximation. *Linear algebra and its applications*, 366:157–172, 2003.

[19] Ian T Jolliffe. Principal component analysis and factor analysis. In *Principal component analysis*, pages 115–128. Springer, 1986.

[20] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[21] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] William H Wolberg. Wisconsin breast cancer dataset. *University of Wisconsin Hospitals*, 1992.

[24] Olvi L Mangasarian. Cancer diagnosis via linear programming. *SIAM news*, 23(5):18, 1990.

[25] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[26] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[27] Stephen D Bay, Dennis Kibler, Michael J Pazzani, and Padhraic Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *ACM SIGKDD Explorations Newsletter*, 2(2):81–85, 2000.

Appendix A Derivation for Equation 4

Proof. Using the definition of empirical HSIC by Gretton et al. [2], Eq(4) can be rewritten as

$$[U^*, \theta^*] = \arg\max_{U,\theta} Tr(D^{-1/2} K_{X;\theta} D^{-1/2} H K_U H), \tag{8}$$

where $D^{-1/2} K_{X;\theta} D^{-1/2}$ are K_U are kernels for $\Psi(X)$ and U. As shown by Niu et al. [3], if we let K_U be a linear kernel such that $K_U = UU^T$, add the constraint such that $U^T U = I$ and rotate the trace terms we get

$$[U^*, \theta^*] = \arg\max_{U,\theta} Tr(U^T H D^{-1/2} K_{X;\theta} D^{-1/2} H U) \tag{9a}$$

$$\text{s.t} : U^T U = I. \tag{9b}$$

Therefore, Eq(3a) is equivalent to Eq(4). □

Appendix B Proof for Property 1

Property : Subset Divergence is a pseudo-metric

Proof. By definition $\mathbb{D}_p : \mathbb{K}_i \times \mathbb{K}_j \to \mathbb{R}_{\geq 0}$, and we further show the following 3 properties :

1. $\mathbb{D}_p(\mathbb{K}_i, \mathbb{K}_i) = 0$

$$\begin{aligned} \mathbb{D}_p(\mathbb{K}_i, \mathbb{K}_i) &= |\lambda_{\mathbb{K}_i} - \lambda_{\mathbb{K}_i}|_\infty & (10)\\ &= |0|_\infty & (11)\\ &= 0 & (12)\end{aligned}$$

2. $\mathbb{D}_p(\mathbb{K}_i, \mathbb{K}_j) = \mathbb{D}_p(\mathbb{K}_j, \mathbb{K}_i)$

$$\begin{aligned} \mathbb{D}_p(\mathbb{K}_i, \mathbb{K}_j) &= |\lambda_{\mathbb{K}_i} - \lambda_{\mathbb{K}_j}|_\infty & (13)\\ &= |\lambda_{\mathbb{K}_j} - \lambda_{\mathbb{K}_i}|_\infty & (14)\\ &= \mathbb{D}_p(\mathbb{K}_j, \mathbb{K}_i) & (15)\end{aligned}$$

3. $\mathbb{D}_p(\mathbb{K}_i, \mathbb{K}_j) \leq \mathbb{D}_p(\mathbb{K}_i, \mathbb{K}_k) + \mathbb{D}_p(\mathbb{K}_k, \mathbb{K}_j)$

$$\begin{aligned} \mathbb{D}_p(\mathbb{K}_i, \mathbb{K}_j) &= |\lambda_{\mathbb{K}_i} - \lambda_{\mathbb{K}_j}|_\infty & (16)\\ &= |\lambda_{\mathbb{K}_i} - \lambda_{\mathbb{K}_j} + \lambda_{\mathbb{K}_j} - \lambda_{\mathbb{K}_j}|_\infty & (17)\\ &\leq |\lambda_{\mathbb{K}_i} - \lambda_{\mathbb{K}_j}| + |\lambda_{\mathbb{K}_j} - \lambda_{\mathbb{K}_j}|_\infty & (18)\\ &\leq \mathbb{D}_p(\mathbb{K}_i, \mathbb{K}_k) + \mathbb{D}_p(\mathbb{K}_k, \mathbb{K}_j) & (19)\\ & & (20)\end{aligned}$$

It can be seen that $\mathbb{D}_p$ satisfies the definition of a pseudo-metric. □

Appendix C Proof for Property 2

Property : At $p = 100\%$, $\mathbb{D}_p(K_{\mathbb{A}}, K_{\mathbb{S}}) = 0$ *if and only if the variances along their PCs are equal.*

The proof for this property is broken down into 3 lemmas.

Lemma 1. *If the covariance matrix* $\Sigma \in \mathbb{R}^{d \times d}$ *and a kernel matrix* $K \in \mathbb{R}^{N \times N}$ *are computed from a dataset* $\Phi(X)$ *in feature space, where* $\Sigma = \frac{1}{N}\Phi(X)^T \Phi(X)$ *and* $K = \Phi(X)\Phi(X)^T$. *The nonzero eigenvalues of* K *are equal to the nonzero variances of* $\Phi(X)$ *scaled by the number of samples along its Principal Components.*

Proof. First note that $\mathbb{S} = \{\phi(x_1), ..., \phi(x_N)\}$ and $\mathbb{A} \subseteq \mathbb{S}$. $\Phi(X) = [\phi(x_1), ..., \phi(x_N)]^T$. Let $V = [\boldsymbol{v_1}, ..., \boldsymbol{v_N}]$ and $\Lambda = diag(\lambda_\Sigma^{(1)}, ..., \lambda_\Sigma^{(N)})$ be the eigenvectors and the eigenvalues of the Σ, then the eigendecomposition of Σ implies

$$\Sigma V = V\Lambda. \tag{21}$$

Since the eigenvector matrix V is within the span of $\Phi(X)$, given a matrix of coefficients $A \in \mathbb{R}^{N\times d}$, the eigenvectors can be written as

$$V = \Phi(X)^T A. \tag{22}$$

If we note that $\Sigma = \frac{1}{N}\Phi(X)^T\Phi(X)$ and $K = \Phi(X)\Phi(X)^T$, we can multiply both sides of Eq(21) by $\Phi(X)$ to get

$$\frac{1}{N}\Phi(X)\Phi(X)^T\Phi(X)V = \Phi(X)V\Lambda. \tag{23}$$

We next replace V from Eq(23) by Eq(22) and apply the definition of K to get

$$KA = A(N\Lambda). \tag{24}$$

From Eq(24), it is clear that A is the eigenvector of K, and if we let Λ_K be the eigenvalues of K, Λ_K is related to Λ by

$$\frac{1}{N}\Lambda_K = \Lambda \tag{25}$$

Eq(25) demonstrates that the eigenvalues of K is proportional to the eigenvalues of Σ by the number of samples. Since the eigenvalues of Σ are the variances along the Principal Components of $\Phi(X)$ based on PCA, the eigenvalues of K is equal to the variance of $\Phi(X)$ scaled by the number of samples. $\square$

Lemma 2. *If $\mathbb{D}_{p=100\%}(K_\mathbb{A}, K_\mathbb{S}) = 0$ then the variances along their PCs are equal.*

Proof. Let the kernels of $\mathbb{A}$ and $\mathbb{S}$ be $K_\mathbb{A}$ and $K_\mathbb{S}$, and m as all the non-zero eigenvalues. Following Eq(25), the eigenvalues of the kernels can be related to the variances of each dataset along the Principal Components. Therefore we let the eigenvalues of $K_\mathbb{A}$ and $K_\mathbb{S}$ be $\boldsymbol{\lambda}_\mathbb{A} = [\frac{1}{N}\lambda_\mathbb{A}^{(1)}, ..., \frac{1}{N}\lambda_\mathbb{A}^{(m)}]^T$ and $\boldsymbol{\lambda}_\mathbb{S} = [\frac{1}{N}\lambda_\mathbb{S}^{(1)}, ..., \frac{1}{N}\lambda_\mathbb{S}^{(m)}]^T$ where $\frac{1}{N}\lambda_\mathbb{S}^{(i)}$ and $\frac{1}{N}\lambda_\mathbb{A}^{(i)}$ are variances along the Principal Components of $\mathbb{S}$ and $\mathbb{A}$. At $p = 100\%$, $\mathbb{D}_p(K_\mathbb{A}, K_\mathbb{S})$ becomes

$$\mathbb{D}_p(K_\mathbb{A}, K_\mathbb{S}) = \left\| \frac{\frac{1}{N}[\lambda_\mathbb{A}^{(1)}, ..., \lambda_\mathbb{A}^{(m)}]^T}{\frac{1}{N}\left\|[\lambda_\mathbb{A}^{(1)}, ..., \lambda_\mathbb{A}^{(m)}]\right\|_1} - \frac{\frac{1}{N}[\lambda_\mathbb{S}^{(1)}, ..., \lambda_\mathbb{S}^{(m)}]^T}{\frac{1}{N}\left\|[\lambda_\mathbb{S}^{(1)}, ..., \lambda_\mathbb{S}^{(m)}]\right\|_1} \right\|_\infty \tag{26}$$

Due to the L_1 normalization, the effect from the number of sample with $\frac{1}{N}$ is cancelled when computing $\mathbb{D}_p$, and therefore, the normalized eigenvalues of the kernel is a direct comparison of the variances along the Principal Components between two datasets. Hence, if $\mathbb{D}_p = 0$, the variances along the Principal Components must also be 0. $\square$

Lemma 3. *If the variances along the PCs of $\mathbb{A}$ and $\mathbb{S}$ are equal then $\mathbb{D}_p(K_\mathbb{A}, K_\mathbb{S}) = 0$.*

Proof. Following lemma 1 and 2, if the variances along the Principal Components are equal, then $\boldsymbol{\lambda}_\mathbb{A} = \boldsymbol{\lambda}_\mathbb{S}$ and $\mathbb{D}_p(K_\mathbb{A}, K_\mathbb{S})$ becomes

$$\mathbb{D}_p(K_\mathbb{A}, K_\mathbb{S}) = \left\| \frac{1}{\|\boldsymbol{\lambda}_\mathbb{A}\|_1}\boldsymbol{\lambda}_\mathbb{A} - \frac{1}{\|\boldsymbol{\lambda}_\mathbb{A}\|_1}\boldsymbol{\lambda}_\mathbb{A} \right\|_\infty = 0 \tag{27}$$

$\square$

Appendix D Using Variances of Principal Components to Compare Data

The variances or the eigenvalues from the Principal Components of the data is capable of summarizing the shape of the data even when the number of samples is different. In Figure D a Gaussian distribution of 3,000 samples is shown on the left. Two other datasets are generated with 1000 samples each. The middle figure is generated with the same distribution while the right figure is generated with a different distribution. It can be seen in the figures that the variances and $\mathbb{D}_p$ from the same distribution are nearly identical and significantly different when compared to a dissimilar distribution.

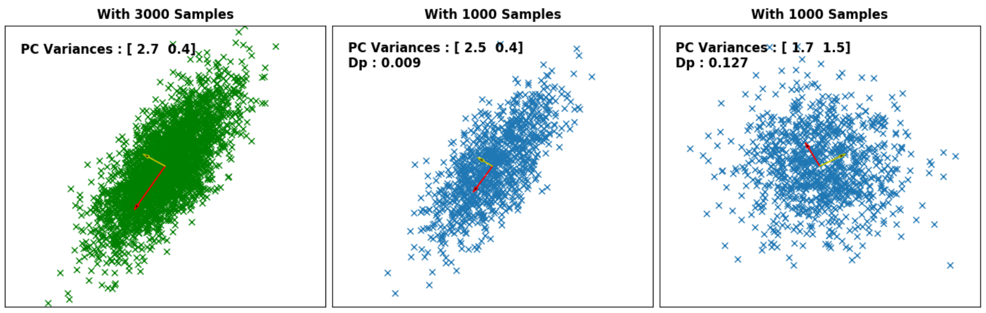

Figure 6

Appendix E Using HSIC From Subset To Set Hyperparameters

There are 3 hyperparameters that can be varied while computing the objective in Eq. (3a), e.i., the width of the hidden layers, the depth of the network and the σ value within the Gaussian kernel. By varying the 3 hyperparameters, Eq. (3a) for the subset is computed and plotted against the NMI from the complete set for four datasets. From the four plots, three observations should be emphasized. First, Eq. (3a) from the subset set is highly predictive of the clustering quality for the complete set. Second, the subset chosen by $\mathbb{D}_p$ is capable of generalizing KNet. Third, since applying Eq. (3a) on the subset provide a predictive measure of the complete set clustering quality, Eq. (3a) can also be used to tune the hyperparameters.

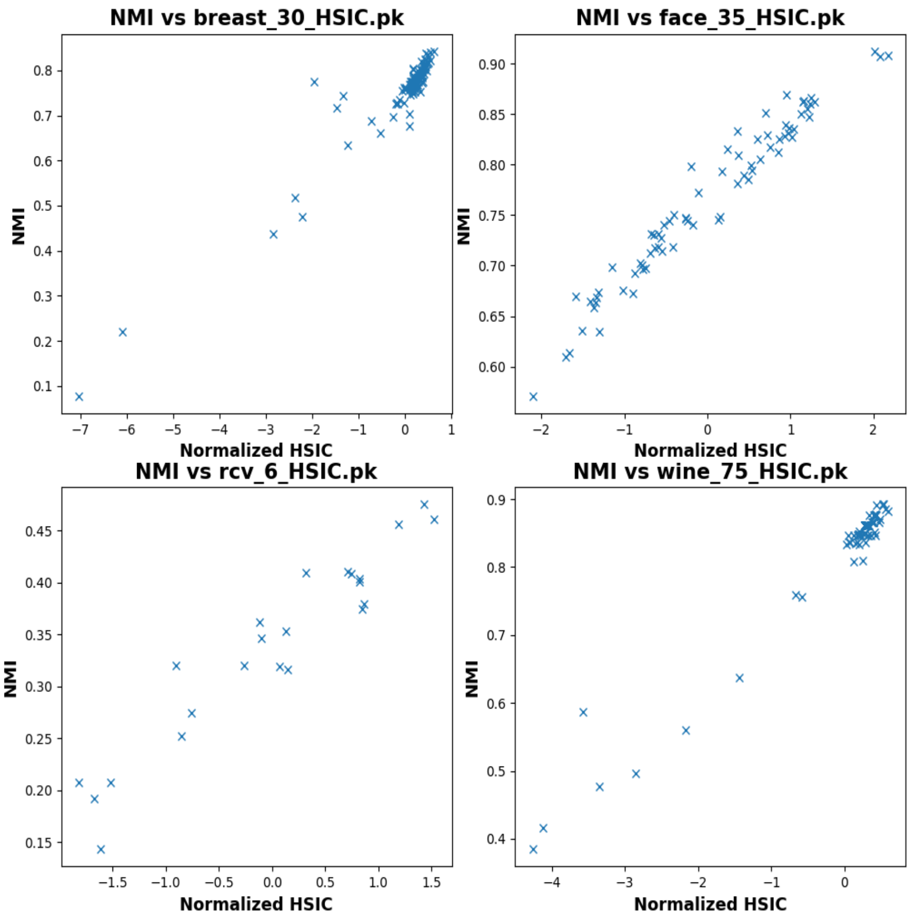

Figure 7

Appendix F Experimental Results In ACC

Beside NMI, accuracy measure (ACC) [12] is another common measure often used. It was proposed in [12] and is calculated with $ACC = \max_q \frac{\sum_{n=1}^{N} 1\{l_n = q(c_n)\}}{N}$. l_n and c_n are respectively the true allocation and the computed allocation. Through q, ACC searches through all permutations of the allocation to find the pairing that yields the maximum accuracy. The following table compares the various benchmark results in terms ACC.

Table 3: Results of comparing KNet with NMI and ACC against various other clustering techniques.

ACC	KMeans	Spectral	AEC	DEC	IMSAT	AE+Kmeans(PreTrain)	KNet
Moon	0.863 ± 0.000	0.930 ± 0.000	0.875 ± 0.000	0.850 ± 0.000	0.849 ± 0.009	0.850 ± 0.000	**1.000 ± 0.000**
Spiral	0.658 ± 0.001	**1.00 ± 0.000**	0.557 ± 0.000	0.738 ± 0.000	0.671 ± 0.054	0.65 ± 0.000	**1.000 ± 0.000**
Cancer	0.957 ± 0.001	0.933 ± 0.000	0.969 ± 0.001	0.970 ± 0.000	0.959 ± 0.0004	0.960 ± 0.000	**0.98 ± 0.001**
Wine	0.96 ± 0.000	0.97 ± 0.000	0.64 ± 0.001	0.388 ± 0.000	0.629 ± 0.020	0.966 ± 0.000	**0.983 ± 0.002**
RCV	0.77 ± 0.000	0.594 ± 0.000	0.650 ± 0.000	0.698 ± 0.002	0.630 ± 0.047	0.724 ± 0.000	**0.810 ± 0.004**
Face	0.749 ± 0.020	0.52 ± 0.009	0.625 ± 0.000	0.635 ± 0.023	0.730 ± 0.062	0.64 ± 0.000	**0.920 ± 0.04**

Appendix G Run Time Comparisons

Table 2 compares the run time of the DNN benchmark methods against KNet. For KNet, the stages are broken into subset sampling (sampling), autoencoder pretraining (Autoencoder), and the actual training time. For the benchmark DNN methods, their times are broken into preprocessing and the run time. The recording is based on experiments that ran on Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz. with s, m, h denoting seconds, minutes, and hours respectively.

Except the autoencoder portion of the Face dataset, where GPU is utilized, the stages of KNet are recorded based on a CPU implementation. As for the benchmark methods, the source code provided by the respective authors have been used for the recording. AEC is written in Matlab with a CPU implementation while both DEC and IMSAT are written in Python and had a GPU implementation.

During the Sampling stage, a grid search is conducted on multiple nodes simultaneously at various sampling percentages. Since they are running in parallel, the longest time would be the time which n is large enough such that the average $\mathbb{D}_p \approx 0.01$. Therefore, the time recorded during the sampling stage uses the slowest time among the nodes. Similarly, during Autoencoder pretraining stage, multiple autoencoders are also running simultaneously. The autoencoder that yields the lowest error is selected. The time here is also the longest time among the nodes. During training, a grid search is necessary to discover the appropriate hyperparameters, i.e., sigma, KNet width, KNet depth, KNets size of the output. However, the time recorded in the table assumes that the hyperparameters are already known. Therefore, it is the time of a single run.

Appendix H Intuition of how the KNet Objective Move Each Sample

To provide a deeper intuition of how Eq. (5) contributes to clustering, we provide a simple example to help visualize the effect of the objective on individual samples. Given an example of 4 samples with 2 clusters where X = [1,1,-1,-1]. Using a Gaussian kernel with $\sigma = 1$, if we use Eq(3a) to solve for K, UU^T, HUU^TH and Y, it yields

$$K = \begin{bmatrix} 1 & 1 & 0.135 & 0.135 \\ 1 & 1 & 0.135 & 0.135 \\ 0.135 & 0.135 & 1 & 1 \\ 0.135 & 0.135 & 1 & 1 \end{bmatrix}, \qquad UU^T = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix} \tag{28}$$

$$HUU^TH = \begin{bmatrix} 0.25 & 0.25 & -0.25 & -0.25 \\ 0.25 & 0.25 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.25 & 0.25 \\ -0.25 & -0.25 & 0.25 & 0.25 \end{bmatrix}, \qquad Y = \begin{bmatrix} 0.2 & 0.2 & -0.2 & -0.2 \\ 0.2 & 0.2 & -0.2 & -0.2 \\ -0.2 & -0.2 & 0.2 & 0.2 \\ -0.2 & -0.2 & 0.2 & 0.2 \end{bmatrix}. \tag{29}$$

Looking at the matrices, a couple of observations are in order. Since the kernel K measures the similarity between the samples, each element $K_{i,j}$ is high if samples i and j are similar and low if they are far apart. Since UU^T represents the spectral embedding of K, this similar relationship is carried over. When the centering matrices are applied, each row is shifted by the average of itself. Therefore, the higher relationship samples remain positive, while lower relationship sample becomes negative. Lastly, since the degree matrix only scales the matrix, it does not affect the signs of the elements.

To solve for θ, Eq(5) is a summation of $Y_{i,j}K_{i,j}$. Since $K_{i,j}$ is a Gaussian kernel, and its range is restricted between [0,1], the optimal solution for Eq(5) is for $K_{i,j}$ to equal to 1 when $Y_{i,j}$ is positive and 0 when $Y_{i,j}$ is negative. As a result, the process of back-propagation is continuously improving a mapping that is pulling similar samples into a clump while pushing dissimilar samples apart. As we update Y, the samples drawn closer will experience an even stronger pull. Conversely, the samples that drifted apart will have less effect on each other. As the process of back-propagation converges, KNet learns a function Ψ that maps the original data to the final state of the back-propagation. For this reason, the output of Ψ have tendencies of transforming samples into tightly compacted clusters that are far apart from each other.

Appendix I Implementation Details for the Subset Selection Process

Note that there are a few undefined parameters in Algorithm 2, i.e., δ, Δ, and K. While further work is needed to establish a more formal basis, we found that empirically setting δ to 0.01, n_0 to 1 percent of N, and K to 100 seems to work well. It is pertinent to mention that the selection procedure will still work even if these values are set differently, however it might result in a subset size larger than these default values. The reason for sampling J times at each n_i and comparing the mean of the distance measure against the threshold is to reduce the effects of random sampling. Furthermore, in the event that the size of $\mathbb{S}$ is extremely large, n is normally significantly smaller than N. It is not necessary to use the complete set as S to compute the $\mathbb{D}_p$. Instead, we suggest to uniformly sample the largest subset that can supported by the PC, and use that set in place of $\mathbb{S}$. Although this practice is heuristic, it works well in practice.

Appendix J Figures In Higher Resolution

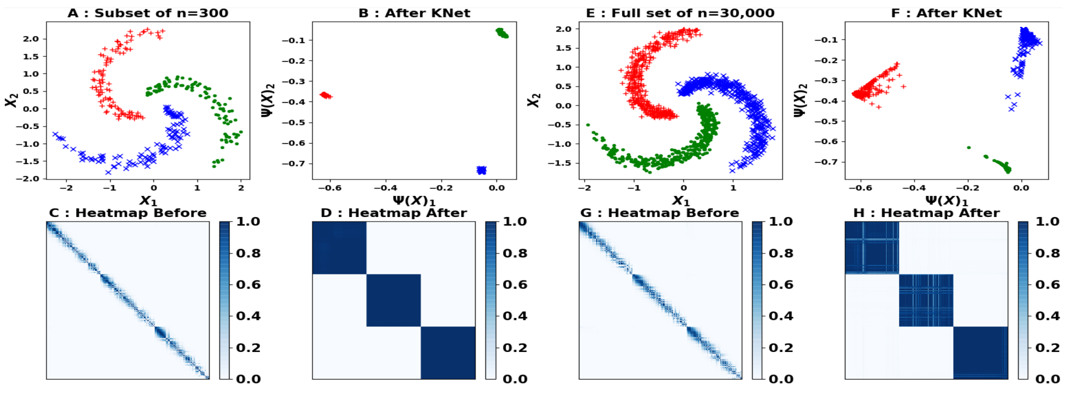

Figure 8: Using only one percent of the data for KNet, 100% of the data as $\Psi(X)$ can achieve a linearly separable representation.

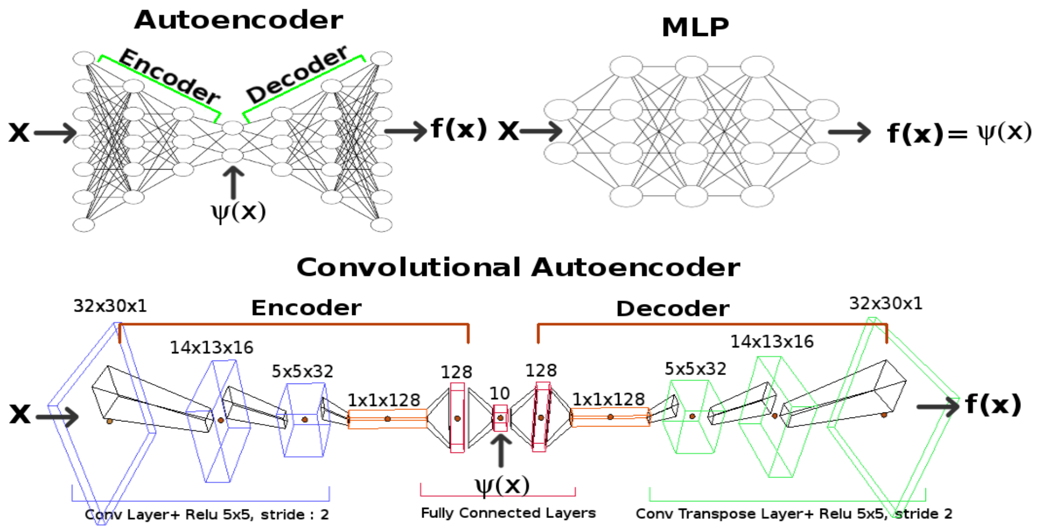

Figure 9

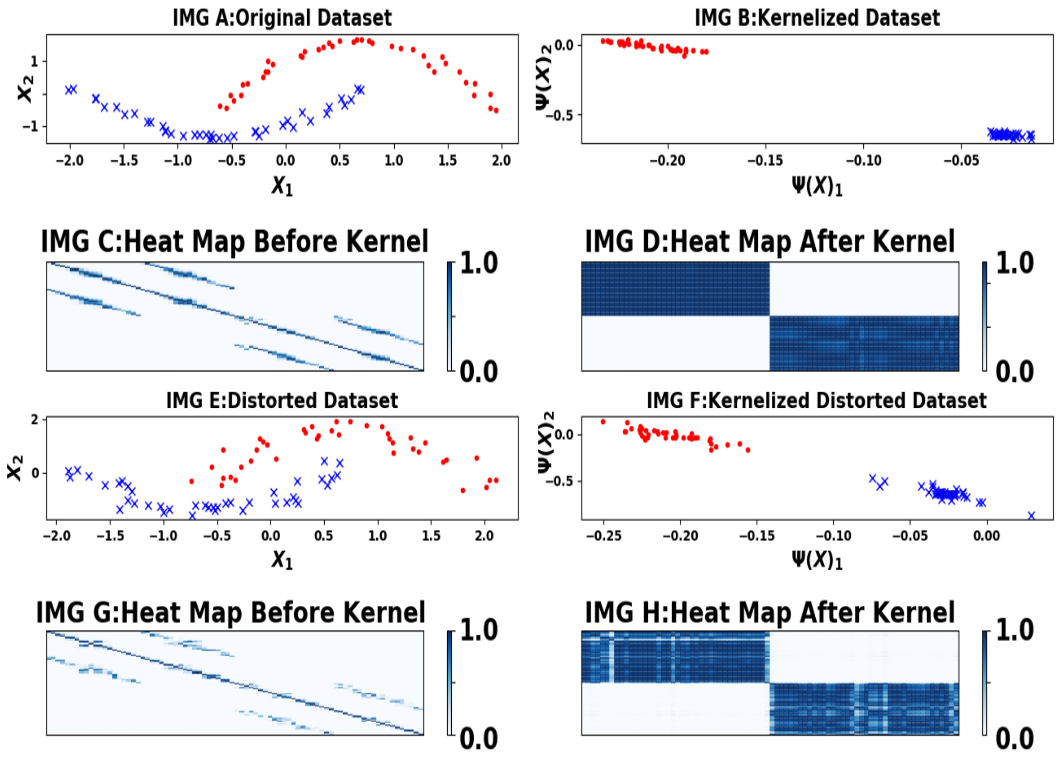

Figure 10