Deep Kernel Learning for Clustering [*]

Chieh Wu   Zulqarnain Khan   Stratis Ioannidis   Jennifer G. Dy [†]

Abstract

We propose a deep learning approach for discovering kernels tailored to identifying clusters over sample data. Our neural network produces sample embeddings that are motivated by and are at least as expressive as spectral clustering. Our training objective, based on the Hilbert Schmidt Independence Criterion, can be optimized via gradient adaptations on the Stiefel manifold, leading to significant acceleration over spectral methods relying on eigen-decompositions. Finally, our trained embedding can be directly applied to out-of-sample data. We show experimentally that our approach outperforms several state-of-the-art deep clustering methods, as well as traditional approaches such as k-means and spectral clustering over a broad array of real and synthetic datasets.

1 Introduction

Clustering algorithms group similar samples together based on some predefined notion of similarity. One way of representing this similarity is through kernels. However, the choice for an appropriate kernel is data-dependent; as a result, the kernel design process is frequently 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 propose KernelNet (KNet), a methodology for learning a kernel and an induced clustering directly from the observed data. In particular, we train a *deep* kernel by combining a neural network representation with a Gaussian kernel. More specifically, given a dataset $\{x_i\}_{i=1}^N$ of N samples in $\mathbb{R}^d$, we learn a kernel $\tilde{k}(\cdot,\cdot)$ of the form:

$$\tilde{k}(x_i, x_j) = e^{-\frac{\|\psi_\theta(x_i)-\psi_\theta(x_j)\|_2^2}{2\sigma^2}} / \sqrt{d_i d_j}, \tag{1.1}$$

where $\psi_\theta(\cdot)$ is an embedding function modeled as a neural network parametrized by θ, and σ, d_i, d_j are normalizing constants. Intuitively, incorporating a neural network (NN) parameterization to a Gaussian kernel, we are able to learn a flexible deep kernel for clustering, tailored specifically to a given dataset.

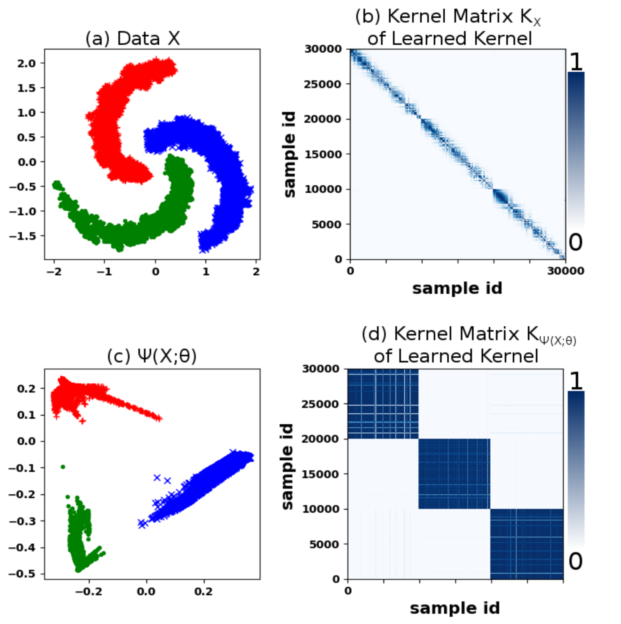

Figure 1: Illustration of our learned embedding on the Spiral dataset. The full dataset and its kernel matrix are shown in (a) and (b), respectively. Applying a Gaussian kernel with $\sigma = 0.3$ directly to this dataset leads to a highly uninformative kernel matrix, as shown in (b). Our embedding of the entire dataset and the resulting kernel matrix are shown in (c) and (d), respectively. Our embedding is trained only on 1% of the samples. Yet, it generalizes to the remaining dataset, produces convex clusters, and yields an informative kernel with a clear block diagonal structure.

We train our deep kernel with a spectral clustering objective based on the Hilbert Schmidt Independence Criterion [1]. This training can be interpreted as learning a non-linear transformation ψ *as well as* its spectral embedding U *simultaneously.* Via an appropriate and intuitive initialization of our training process, we ensure that our clustering method is at least as powerful as spectral clustering. In particular, just as spectral clustering, our learned kernel and the induced clustering work exceptionally well on non-convex clusters. In practice, by training the kernel directly from the data, our proposed method significantly outperforms spectral

[*]The supplementary material can be found at [15]. Source code is available at https://github.com/neu-spiral/KernelNet

[†]All authors are associated with Department of Electrical and Computer Engineering, Northeastern University, Boston, MA

clustering.

The non-linear transformation ψ learned directly from the dataset allows us to readily handle *out-of-sample* data. Given a new unobserved sample $x_u \in \mathbb{R}^d$, we can easily identify its cluster label by first computing its image $\psi_\theta(x_u)$, thus embedding it in the same space as the (already clustered) existing dataset. This is in contrast to spectral clustering, that would require a re-execution of the algorithm from scratch on the combined dataset of $N+1$ samples. The aforementioned properties of our algorithm are illustrated in Fig. 1. A dataset of $N = 30,000$ samples in $\mathbb{R}^2$ with non-convex spiral clusters is shown in Fig. 1(a). Applying a Gaussian kernel with $\sigma = 0.3$ directly to these samples leads to a highly uninformative kernel matrix, as shown in Fig 1(b). We train our embedding $\psi_\theta(\cdot)$ on only 1% of the samples, and apply it to the *entire dataset*; the embedded data, shown in Fig. 1(c), consists of nearly-convex, linearly-separable clusters. More importantly, the corresponding learned kernel $\tilde{k}(\cdot,\cdot)$, yields a highly informative kernel matrix that clearly exhibits a block diagonal structure as shown in Fig. 1(d). In summary, **our major contributions are**:

- We propose a novel methodology of discovering a deep kernel tailored for clustering directly from data, using an objective based on the Hilbert-Schmidt Independence Criterion.
- We propose an algorithm for training the kernel by maximizing this objective, as well as for selecting a good parameter initialization. Our algorithm, KNet, can be perceived as alternating between training the kernel and discovering its spectral embedding.
- We evaluate the performance of KNet with synthetic and real data compared to multiple state-of-the-art methods for deep clustering. In 5 out 6 datasets, KNet outperforms state-of-the-art by as much as 57.8%; this discrepancy is more pronounced in datasets with non-convex clusters, which KNet handles very well.
- Finally, we demonstrate that the algorithm does well in clustering out-of-sample data. This generalization capability means we can significantly accelerate KNet through subsampling: learning the embedding ψ on only 1%-35% of the data can be used to cluster an entire dataset, leading only to a 0%-3% degradation of clustering performance.

2 Related Work

Several recent works propose autoencoders specifically designed for clustering. Song et al. [2] combine an autoencoder with k-means, including an ℓ_2-penalty w.r.t. distance to cluster centers. They optimize this objective by alternating between stochastic gradient descent (SGD) and cluster center assignment. Ji et al. [3] incorporate a subspace clustering penalty to an autoencoder, and alternate between SGD and dictionary learning. Tian et al. [4] learn a stacked autoencoder initialized via a similarity matrix. Xie et al. [5] incorporate a KL-divergence penalty between the encoding and a soft cluster assignment, both of which are again alternately optimized; a similar approach is followed by Guo et al. [6] and Hu et al. [7]. In KNet, we significantly depart from these methods by using an HSIC-based objective, motivated by spectral clustering. In practice, this makes KNet better tailored to learning non-convex clusters, on which the aforementioned techniques perform poorly. We demonstrate this experimentally in Section 6.

Our work is closest to *SpectralNet* [8] by Shaham et al. in that the authors propose a neural network approach for spectral clustering. However, they first learn a similarity matrix using a Siamese network, and then keeping this similarity *fixed* they optimize a spectral clustering objective to learn the spectral embedding. In contrast, KNet learns *both* the kernel similarity matrix and the spectral embedding *jointly*, iteratively improving both. By not having a mechanism to improve upon the previously learnt similarity matrix once a spectral embedding is learnt, as KNet does, SpectralNet can only do as well as the initially learnt similarity matrix: this is evidenced by the overall improved performance of KNet over SpectralNet (see also Sec. 6).

KNet also has some relationship to methods for kernel learning. A series of papers [11, 9, 10] regress deep kernels to model Gaussian processes. Zhou et al. [12] learn (shallow) linear combinations of given kernels. Closest to us, Niu et al. [13] use HSIC to jointly discover a subspace in which data lives as well as its spectral embedding; the latter is used to cluster the data. This corresponds to learning a kernel over a (shallow) projection of the data to a linear subspace. KNet, therefore, generalizes the work by Niu et al. [13] to learning a deep, non-linear kernel representation (c.f. Eq. (4.10)), which improves upon spectral embeddings and is used directly to cluster the data.

3 Hilbert-Schmidt Independence Criterion

Proposed by Gretton et al. [1], the Hilbert Schmidt Independence Criterion (HSIC) is a statistical dependence measure between two random variables. Like Mutual Information (MI), it measures dependence by comparing the joint distribution of the two random variables with the product of their marginal distributions. However, compared to MI, HSIC is easier to compute empirically, since it does not require a direct estimation of the joint distribution. It is used in many applications due to this advantage, including dimensionality reduction [13], feature selection [17], and alternative clustering [14], to name a few.

Formally, consider a set of N i.i.d. samples $\{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}^c$ are drawn from a joint distribution. Let $X \in \mathcal{R}^{N\times d}$ and $Y \in \mathcal{R}^{N\times c}$ be the corresponding matrices comprising a sample in each row. Let also $k_X : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be any characteristic kernel, in this paper we consider Gaussian kernel $k_X(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$, and $k_Y : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ be another characteristic kernel, assumed to be the linear kernel $k_Y(y_i, y_j) = y_i^\top y_j$ here. Define $K_X, K_Y \in \mathbb{R}^{N\times N}$ to be the kernel matrices with entries $K_{X_{i,j}} = k_X(x_i, x_j)$ and $K_{Y_{i,j}} = k_Y(y_i, y_j)$, respectively, and let $\tilde{K}_X \in \mathbb{R}^{N\times N}$ be the normalized Gaussian kernel matrix given by

$$\tilde{K}_X = D^{-1/2} K_X D^{-1/2}, \tag{3.2}$$

where the degree matrix

$$D = \operatorname{diag}(K_X \mathbf{1}_N) \in \mathbb{R}^{N\times N} \tag{3.3}$$

is a normalizing diagonal matrix. Then, the HSIC between X and Y is estimated empirically via:

$$\mathbb{H}(X, Y) = \tfrac{1}{(N-1)^2} \operatorname{tr}(\tilde{K}_X H K_Y H), \tag{3.4}$$

where intuitively, HSIC empirically measures the dependence between samples of the two random variables. Though HSIC can be more generally defined for any arbitrary characteristic kernels, this particular choice has a direct relationship with (and motivation from) spectral clustering. In particular, given X, consider the optimization:

$$\max_U \quad \mathbb{H}(X, U) \tag{3.5a}$$
$$\text{subject to} \quad U^\top U = I,\ U \in \mathbb{R}^{N\times c}, \tag{3.5b}$$

where $\mathbb{H}$ is given by (3.4). Then, the optimal solution $U_0 \in \mathbb{R}^{N\times c}$ to (3.5) is precisely the spectral embedding of X [13]. Indeed, U_0 comprises the top c eigenvectors of the normalized similarity matrix, given by:

$$\mathcal{L} = H D^{-1/2} K_X D^{-1/2} H. \tag{3.6}$$

For completeness, we prove this in Appendix A.

4 Problem Formulation

We are given a dataset of samples grouped in (potentially) non-convex clusters. Our objective is to cluster samples by first embedding them into a space in which the clusters become convex. Given such an embedding, clusters can subsequently be identified via, e.g., k-means. We would like the embedding, modeled as a neural network, to be at least as expressive as spectral clustering: clusters separable by spectral clustering should also become separable via our embedding. In addition, the embedding should generalize to out-of-sample data, thereby enabling us to cluster new samples outside the original dataset.

Learning the Embedding and a Deep Kernel. Formally, we wish to identify c clusters over a dataset $X \in \mathbb{R}^{N\times d}$ of N samples and d features. Let $\psi : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^{d'}$ be an embedding of a sample to $\mathbb{R}^{d'}$, modeled as a DNN parametrized by $\theta \in \mathbb{R}^m$; we denote by $\psi_\theta(x)$ the image of $x \in \mathbb{R}^d$ under parameters θ. We also denote by $\Psi : \mathbb{R}^{N\times d} \times \mathbb{R}^m \to \mathbb{R}^{N\times d'}$ the embedding of the entire dataset induced by ψ, and use $\Psi_\theta(X)$ for the image of X.

Let $U_0 \in \mathbb{R}^{N\times c}$ be the spectral embedding of X, obtained via spectral clustering. We can train ψ to induce similar behavior as U_0 via the following optimization:

$$\max_{\theta\in\mathbb{R}^m} \mathbb{H}(\Psi_\theta(X), U_0), \tag{4.7}$$

where $\mathbb{H}$ is given by Eq. (3.4). Since HSIC is a dependence measure, by training θ so that $\Psi_\theta(X)$ is maximally dependent on U_0, Ψ becomes a surrogate to the spectral embedding, sharing similar properties.

However, the surrogate Ψ learned via (4.7) is restricted by U_0, hence it can only be as discriminative as U_0. To address this issue, we depart from (4.7) by jointly discovering both Ψ as well as a *coupled* spectral embedding U. In particular, we solve the following optimization problem w.r.t. *both* the embedding *and* U:

$$\max_{\theta,U} \quad \mathbb{H}(\Psi_\theta(X), U) - \lambda \|X - f_{\theta,\theta'}(X)\|_2^2 \tag{4.8a}$$
$$\text{subject to:} \quad U^\top U = I, \tag{4.8b}$$
$$\theta, \theta' \in \mathbb{R}^m,\ U \in \mathbb{R}^{N\times c}, \tag{4.8c}$$

where,

$$f_{\theta,\theta'}(X) = \Psi'_{\theta'}(\Psi_\theta(X)) \tag{4.9}$$

is an autoencoder, comprising $\Psi : \mathbb{R}^{n\times d} \times \mathbb{R}^m \to \mathbb{R}^{N\times d'}$ and $\Psi' : \mathbb{R}^{N\times d'} \times \mathbb{R}^m \to \mathbb{R}^{N\times d}$ as an encoder and decoder respectively. This autoencoder objective is theoretically necessary to ensure that the embedding Ψ is injective, as stated in Theorem 1 by Li et al. in [30]. However, as observed in [30], our empirical experiments also suggest that this autoencoder objective is not necessary in practice: the dependence of the local minimum found by gradient descent to the starting point ensures that the trained embedding Ψ is representative of the input X even in the absense of this penalty (see Sec. 6.1).

To gain some intuition into how problem (4.8) generalizes (4.7), observe that if the embedding Ψ is fixed to be the identity map (i.e., for $\Psi_\theta(X) \equiv X$) then, by Eq. (3.5), optimizing only for U produces the spectral embedding U_0 . The joint optimization of both Ψ and U allows us to further improve upon U_0, as well as on the coupled Ψ; we demonstrate experimentally in Section 6 that this significantly improves clustering quality.

Kernel Learning. The optimization (4.8) can also be interpreted as an instance of *kernel learning*. Indeed, as discussed in the introduction, by learning ψ, we discover in effect a normalized kernel $\tilde{k}$ of the form

$$\tilde{k}(x_i, x_j) = e^{-\frac{\|\psi_\theta(x_i)-\psi_\theta(x_j)\|_2^2}{2\sigma^2}} / \sqrt{d_i d_j}, \tag{4.10}$$

where d_i, d_j are the corresponding diagonal elements of degree matrix D.

Out-of-Sample Data. The embedding Ψ can readily be applied to clustering out-of-sample data. In particular, having trained Ψ over dataset X, given a new dataset $Y \in \mathbb{R}^{N' \times d}$, we can cluster this new dataset efficiently as follows. First, we use the pre-trained Ψ to map every sample y_i in Y to its image, producing $\Psi_\theta(Y)$: this effectively embeds Y to the same space as $\Psi_\theta(X)$. From this point on, clusters can be recomputed efficiently via, e.g., k-means, or by mapping the images $\psi_\theta(y_i)$ to the closest existing cluster head. In contrast to, e.g., spectral clustering, this avoids recomputing the joint embedding of the entire dataset $(X; Y)$ from scratch.

The ability to handle out-of-sample data can be leveraged to also accelerate training. In particular, given the original dataset X, computation can be sped up by training the embedding Ψ by solving (4.8) *on a small subset of* X. The resulting trained Ψ can be used to embed, and subsequently cluster, the entire dataset. We show in Section 6 that this approach works very well, leading to a significant acceleration in computations without degrading clustering quality.

Convex Cluster Images. The first term in objective (4.8a) naturally encourages Ψ to form convex clusters. To see this, as derived in Appendix C, ignoring the reconstruction error, the objective (4.8a) becomes:

$$\textstyle\sum_{i,j} \Gamma_{i,j} e^{-\frac{1}{2\sigma^2}||\psi_\theta(\mathbf{x}_i)-\psi_\theta(\mathbf{x}_j)||^2}, \tag{4.11}$$

where $\Gamma_{i,j}$ are the elements of matrix $\Gamma = D^{-1/2}HUU^THD^{-1/2} \in \mathbb{R}^{N \times N}$. The exponential terms in Eq. (4.11) compel samples under which $\Gamma_{i,j} > 0$ to become attracted to each other, while samples for which $\Gamma_{i,j} < 0$ drift farther apart. This is illustrated in Figure 2. Linearly separable, increasingly convex cluster images arise over several iterations of solving our algorithm at Eq. (4.8). The algorithm, KNet, is described in the next section.

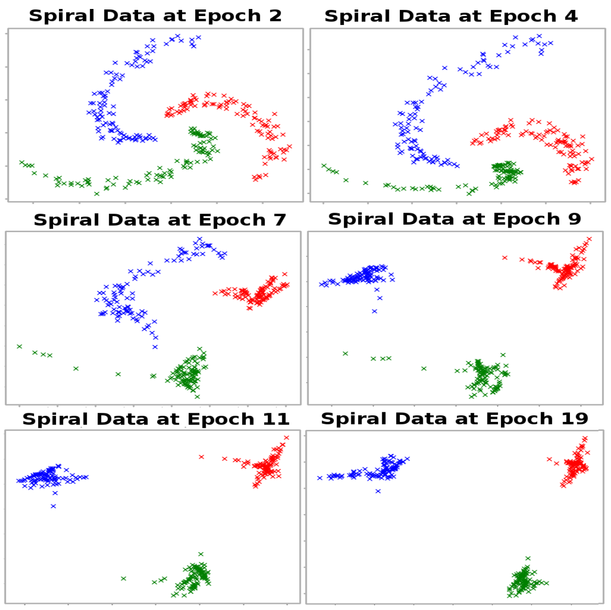

Figure 2: Cluster images after several epochs of stochastic gradient descent over (4.8a). The first term of objective (4.8a) pulls cluster images further apart, while making them increasingly convex. Note that this happens in a fully unsupervised fashion.

5 KNet Algorithm

We solve optimization problem (4.8) by iteratively adapting $\tilde{\theta} = (\theta, \theta')$ and U. In particular, we initialize Ψ to be the identity map, and U to be the spectral embedding U_0, and then alternate between adapting U and $\tilde{\theta}$. We optimize $\tilde{\theta}$ via stochastic gradient ascent (SGA). To optimize U, we adopt two approaches: one based on eigendecomposition, and one based on optimization over the Stiefel manifold. We describe each of these steps in detail below; a summary can be found in Algorithm 1.

Initialization. The non-convexity of (4.8) necessitates a principled approach for selecting good initialization points for U and $\tilde{\theta} = (\theta, \theta')$. We initialize U to U_0, computed via the top-c eigenvectors of the normalized similarity matrix $\mathcal{L}$ of X, given by (3.6). We initialize θ so that Ψ is the identity map; This is accomplished by pre-training θ, θ' via SGD as solutions to:

$$\min_{\theta,\theta'} \|X - \Psi_\theta(X)\|_2^2 + \|X - f_{\theta,\theta'}(X)\|_2^2. \tag{5.12}$$

Note that, in this construction, we use $d' = d$.

Updating $\tilde{\theta}$. A simple way to update $\tilde{\theta}$ is via gradient

Algorithm 1 KNet Algorithm

```
1: Input: data $X \in \mathbb{R}^{N\times d}$
2: Output: $\Psi$ and clustering labels
3: Initialization: Initialize $\tilde{\theta} = (\theta, \theta')$ via (5.12)
   Initialize $U$ with $U_0$
4: repeat
5:     Update $\tilde{\theta}$ via one epoch of SGA over (4.8a), holding $U$ and $D$ fixed
6:     Update $D$ via Eq. (3.3)
7:     KNet$_{\texttt{EIG}}$: Update $U$ via eigendecomposition of Laplacian (5.14),
       or
       KNet$_{\texttt{SMA}}$: Update $U$ via Stiefler Manifold Ascent (5.16).
8: until $K_U$ has converged
9: Run $k$-means on $\Psi(X;\theta)$
```

ascent, i.e.:

$$\tilde{\theta}_k = \tilde{\theta}_{k-1} + \gamma_k \nabla F(\tilde{\theta}_{k-1}, U_{k-1}), \tag{5.13}$$

for $k \geq 1$, where F is the objective (4.8a). In practice, we wish to apply stochastic gradient ascent over mini-batches; for U fixed, the first term in the objective (4.8a) reduces to (4.11); however, the terms in the sum are coupled via the normalizing degree matrix D, which depends on θ via (3.3). This significantly increases the cost of computing mini-batch gradients. To simplify this computation, instead we hold both U *and* D fixed, and update $\tilde{\theta}$ via one epoch of SGA over (4.8a). At the conclusion of one epoch, we update the Gaussian kernel K_X and the corresponding degree matrix D via Eq. (3.3). We implemented both this heuristic and regular SGA, and found that it led to a significant speedup without any observable degradation in clustering performance (see also Section 6).

Updating U via Eigendecomposition. Our first approach to adapting U relies on the fact that, holding $\tilde{\theta}$ constant, problem (4.8) reduces to the form (3.5). That is, at each iteration, for $\tilde{\theta}$ fixed, the optimal solution

$$U^*(\tilde{\theta}) = \arg\max_{U:U^\top U = I} F(\tilde{\theta}, U)$$

is given by the top c eigenvectors of matrix

$$\mathcal{L}_\theta = HD^{-1/2} K_{\Psi_\theta(X)} D^{-1/2} H. \tag{5.14}$$

Hence, given $\tilde{\theta}$ at an iteration, we update U by returning $U^*(\tilde{\theta})$. Note that when $c \ll N$, there are several algorithms for computing the top eigenvectors efficiently (see, e.g., [16, 18]).

Updating U via Stiefel Manifold Ascent. The manifold in $\mathbb{R}^{N\times c}$ defined by constraint (4.8b), a.k.a. the *Stiefel Manifold*, is not convex; nevertheless, techniques such as those outlined in [19, 20] for optimization over this set are available in the literature. These techniques exploit the fact that descent directions that maintain feasibility can be computed efficiently. In particular, following [20], treating $\tilde{\theta}$ as a constant, and given a feasible $U \in \mathbb{R}^{N\times c}$ and the gradient of the objective $\nabla_U F(U) \in \mathbb{R}^{N\times c}$ w.r.t U, define

$$A(U) = -(\nabla_U F(U) U^T + U \nabla_U F(U)^T) \in \mathbb{R}^{N\times N}. \tag{5.15}$$

Using A and a predefined step length τ, the maximization proceeds iteratively via:

$$U_{k+1} = Q(U_k) U_k, \tag{5.16}$$

where Q is the so-called Cayley transform, defined as

$$Q(U) = (I + \frac{\tau}{2} A(U))^{-1} (I - \frac{\tau}{2} A(U)). \tag{5.17}$$

The Cayley transform satisfies several important properties [20]. First, starting from a feasible point, it maintains feasibility over the Stiefel manifold (4.8b) for all $k \geq 1$. Second, for small enough τ, it is guaranteed to follow an ascent direction; combined with line-search, convergence is guaranteed to a stationary point. Finally, $Q(U_{k+1})$ given by (5.17) can be computed efficiently from $Q(U_k)$, thereby avoiding a full matrix inversion, by using the Sherman-Morrison-Woodbury identity [21]: this results in a $O(N^2c + c^3)$ complexity for (5.16), which is significantly faster than eigendecomposion when $c \ll N$. In our second approach to updating U, we apply (5.16) rather than eigendecomposition of $\mathcal{L}_\theta$ when adapting U iteratively. Both approaches are summarized in line 7 of Alg. 1; we refer to them as KNet$_{\texttt{EIG}}$ and KNet$_{\texttt{SMA}}$, respectively, in our experiments in Sec. 6.

6 Experimental Evaluation

Datasets. The datasets we use are summarized in Table 6. The first three datasets (**Moon**, **Spiral1**, **Spiral2**) are synthetic and comprise non-convex clusters; they are shown in Figure 3. Among the remaining four real-life datasets, the features of **Breast Cancer** [22] are discrete integer values between 0 to 10. The features of the **Wine** dataset [23] consist of a mix of real and integer values. The Reuters dataset (**RCV**) is a collection of news articles labeled by topic. We represent each article via a *tf-idf* vector using the 500 most frequent words and apply PCA to further reduce the dimension to $d = 5$. The **Face** dataset [24] consists of grey scale, 32×30-pixel images of 20 faces in different orientations. We reduce the dimension to $d = 20$ via PCA. As a final preprocessing step, we center and scale all datasets so that their mean is 0 and the standard deviation of each feature is 1.

Clustering Algorithms. We evaluate 8 algorithms, including our two versions of KNet described in Alg. 1. For existing algorithms, we use architecture designs (e.g., depth, width) as recommended by the respective authors during training. We provide details for each algorithm below.

Data	N	c	d	Type	σ
Moon	1000	2	2	Geometric Shape	0.1701
Spiral1	3000	3	2	Geometric Shape	0.1708
Spiral2	30000	3	2	Geometric Shape	0.1811
Cancer	683	2	9	Medical	3.3194
Wine	178	3	13	Classification	4.939
RCV	10000	4	5	Text	2.364
Face	624	20	27	Image	6.883

Table 1: Dataset Summary.

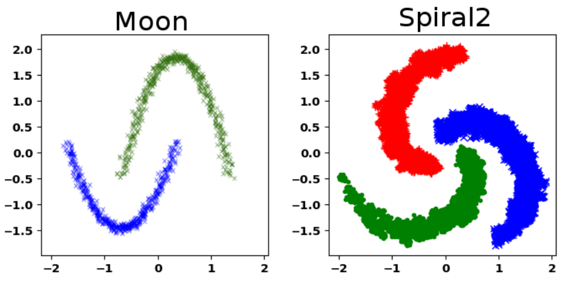

Figure 3: Synthetic Datasets. Both dataset contain non-convex clusters. Dataset Spiral2 (depicted) contains $N = 30,000$ samples, while Spiral1 contains a sub-sampled version of $N = 3,000$.

***k*-means:** We use the CUDA implementation[1] by [25].
SC: We use the python scikit implementation of the classic spectral clustering algorithm by [26].
AEC: Proposed by [2], this algorithm incorporates a k-means objective in an autoencoder. As suggested by the authors, we use 3 hidden layers of width 1000, 250, and 50, respectively, with an output layer dimension of 10.[2]
DEC: Proposed by [5], this algorithm couples an autoencoder with a soft cluster assignment via a KL-divergence penalty. As recommended, we use 3 hidden layers of width 500, 500, and 2000 with an output layer dimension of 10.[3]
IMSAT: Proposed by [7], this algorithm trains a network adversarially by generating augmented datasets. It uses 2 hidden layers of width 1200 each, the output layer size equals the number of clusters c.[4]
SN: Proposed by [8], SN uses an objective motivated by spectral clustering to map to a target similarity matrix.[5]
KNet$_{\texttt{EIG}}$ and **KNet$_{\texttt{SMA}}$:** These are the two versions of KNet, as described in Alg. 1, in which U is updated via eigendecomposition and Stiefel Manifold Ascent, respectively. For both versions, the encoder and decoder have 3 layers. For Cancer, Wine, RCV, and Face dataset, we set the width of all hidden layers to d. For Moon and Spiral1, we set the width of all hidden layers to 20. We set the Gaussian kernel σ to be median of the pairwise Euclidean distance between samples in each dataset (see Table 6).[6]

[1]https://github.com/src-d/kmcuda
[2]https://github.com/developfeng/DeepClustering
[3]https://github.com/XifengGuo/DEC-keras
[4]https://github.com/weihua916/imsat
[5]https://github.com/KlugerLab/SpectralNet

Evaluation Metrics. We evaluate the clustering quality of each algorithm by comparing the clustering assignment generated to the ground truth assignment via the Normalized Mutual Information (NMI). NMI is a similarity metric lying in $[0, 1]$, with 0 denoting no similarity and 1 as an identical match between the assignments. Originally recommended by [27], this statistic has been widely used for clustering quality validation [13, 28, 14, 29]. We provide a formal definition in Appendix D in the supplement.

For each algorithm, we also measure the execution time, separating it into preprocessing time (Prep) and runtime (RT); in doing so, we separately evaluate the cost of, e.g., parameter initialization from training.
Experimental Setup. We execute all algorithms over a machine with 16 dual-core CPUs (Intel Xeon® E5-2630 v3 @ 2.40GHz) with 32 GB of RAM with a NVIDIA 1b80 GPU. For methods we can parallelize either over the GPU or over the 16 CPUs (IMSAT,SN,k-means,KNet), we ran both executions and recorded the fastest time. The code provided for DEC could only be parallelized over the GPU, while methods AEC and SC could only be executed over the CPUs. For each dataset in Table 2, we run all algorithms on the full dataset 10 times and report the mean and standard deviation of the NMI of their resulting clustering performance against the ground truth. As SGA is randomized, we repeat experiments 10 times and report NMI averages and standard deviations.

For algorithms that can be executed out-of-sample (AEC, DEC, IMSAT, SN, KNET), we repeat the above experiment by training the embedding only on a random subset of the dataset. Subsequently, we apply the trained embedding to the entire dataset and cluster it via k-means. For comparison, we also find cluster centers via k-means on the subset and new samples to the nearest cluster center. For each dataset, we set the size of the subset (reported in Table 4) so that the spectrum of the resulting subset is close, in the ℓ_∞ sense, to the spectrum of X.

6.1 Results Selecting λ.

As clustering is unsupervised, we cannot rely on ground truth labels to identify the best hyperparameter λ. We, therefore, need an unsu-

[6]https://github.com/neu-spiral/KernelNet

Dataset	AEC	DEC	IMSAT	SN	SC	k-means	KNet$_{\texttt{EIG}}$	KNet$_{\texttt{SMA}}$
Moon	56.2 ± 0.0	42.2 ± 0.0	51.3 ± 20.3	**100** ± 0.0	72.0 ± 0.0	66.1 ± 0.0	**100.0 ± 0.0**	**100.0 ± 0.0**
Spiral1	28.3 ± 0.0	32.0 ± 0.01	59.6 ± 7.5	**100.0 ± 0.0**	**100.0 ± 0.0**	42.0 ± 0.0	**100.0 ± 0.0**	**100.0 ± 0.0**
Cancer	79.9 ± 0.2	79.2 ± 0.0	74.6 ± 2.2	82.9 ± 0.0	69.8 ± 0.0	73.0 ± 0.0	**84.2 ± 0.4**	82.5 ± 0.1
Wine	54.6 ± 0.0	80.6 ± 0.0	72.3 ± 11.4	79.7 ± 0.2	88.0 ± 0.0	42.8 ± 0.0	**91.0 ± 0.8**	90.0 ± 0.7
RCV	39.3 ± 0.0	51.3 ± 0.0	39.0 ± 5.5	43.5 ± 0.2	46 ± 0.0	**56.0 ± 0**	46.3 ± 0.4	46.1 ± 0.2
Face	76.8 ± 0.0	75.8 ± 1.6	83.8 ± 3.5	75.6 ± 0.1	66.0 ± 0.4	91.8 ± 0.0	**93.0 ± 0.3**	92.6 ± 0.5

Table 2: The **clustering results** measured by NMI as percentages are shown above where the best mean results are highlighted in bold text. Besides the RCV dataset, KNet generally outperforms competing methods by a significant margin. The improvement is especially large with the Moon and Spiral1 dataset due to KNet's ability to identify non-convex clusters.

	AEC		DEC		IMSAT		SN	SC	k-means	KN		
Dataset	Prep	RT	Prep	RT	Prep	RT	RT	RT	RT	Prep	RT$_{\texttt{EIG}}$	RT$_{\texttt{SMA}}$
Moon	18.23	28.04	331.80	3.52	1.20	34.34	324.00	0.34	0.04	129.00	28.90	18.40
Spiral1	2574.00	7920.00	343.80	121.20	53.06	309.00	444.00	4.20	0.12	300.00	42.00	19.00
Cancer	99.00	280.20	342.00	38.73	1.10	144.00	234.00	0.18	0.03	150.00	19.30	10.30
Wine	33.81	41.69	345.00	38.02	1.10	33.32	330.00	0.03	0.06	462.00	7.20	3.40
RCV	141.60	2536.80	381.00	784.20	10.39	73.20	438.00	83.40	0.35	1080.00	1830.00	1116.00
Face	27.50	189.00	344.40	254.10	1.20	121.80	170.90	0.26	0.15	1320.00	20.90	3.30

Table 3: The **preprocessing (Prep) and runtime (RT)** for all benchmark algorithms are displayed in seconds. The table demonstrates that KNet's speed is comparable to competing methods.

Dataset	Data %	AEC	DEC	IMSAT	SN	k-means	KNet$_{\texttt{EIG}}$	KNet$_{\texttt{SMA}}$
Moon	25%	51.1	45.6	45.5	**100.0**	21.5	**100.0**	**100.0**
Spiral1	10.0%	32.2	49.5	48.7	**100.0**	56.7	**100.0**	**100.0**
Cancer	30.0%	76.4	76.9	74.9	82.2	Fails	**84.0**	83.6
Wine	75.0%	49.1	81.5	69.4	77.2	25.0	**91.1**	89.3
RCV	6.0%	26.8	43.2	35.2	41.3	**52.5**	45.2	43.1
Face	35.0%	52.5	67.1	77.8	75.5	87.2	**92.7**	91.1

Table 4: The **out-of-sample clustering** result measured by NMI as percentages are shown above where the best mean results are highlighted in bold text. All algorithms are trained on a subset of data. We report the results of the total dataset clustered out-of-sample via each algorithm.

pervised method for selecting this value. We find that, in practice, just like [30], selecting $\lambda = 0$ works quite well. Because the problem is not convex, local optima reached by KNet depend highly on the initialization. Initializing (a) Ψ to approximate the identity map via (5.12), and (b) U to be the spectral embedding U_0 indeed leads to a local maximum that is highly dependent on the input X, eschewing the need for the reconstruction error in the objective (4.8).

Our choie of $\lambda = 0$ is further grounded in experiments shown in Table 6 which shows results for the Wine dataset. We found that at smaller values of λ result in improved performance both in terms of HSIC and NMI; tables for additional datasets can be found in Appendix B in the supplement. Alongwith the HSIC we also provide AE reconstruction error at convergence. Beyond the good performance of $\lambda = 0$, the table suggests that an alternative unsupervised method is to select λ so that the ratio of the two terms at convergence is close to one. Of course, this comes at the cost of parameter exploration; in the remainder of this section, we report the performance of KNet with $\lambda = 0$.

Comparison Against State-of-the-Art. Table 2 shows the NMI performance of different algorithms over different datasets. With the exception of RCV dataset, we see that KNet outperforms every algorithm in Table 2. AEC, DEC, and IMSAT perform especially poorly when encountering non-convex clusters as shown in the first two rows of the table. Spectral clustering (SC), and SN that is also based on a spectral-clustering motivated objective, perform equally well as KNet on discovering non-convex clusters. Nevertheless, KNet outperforms them for real datasets: e.g., for the Face dataset, KNet$_{\texttt{EIG}}$ surpasses SN by 28%. Note that, for the RCV dataset, k-means outperformed all methods, though overall performance is quite poor; a reason for this may be the poor quality of features extracted via TFIDFs and PCA.

KNet's ability to handle non-convex clusters is evident in the improvement over k-means to KNet for the first two datasets. The kernel matrix K_X shown in Fig. 1(b) illustrates why k-means performs

	AEC		DEC		IMSAT		SN	k-means	KN		
Data	Prep	RT	Prep	RT	Prep	RT	RT	RT	Prep	$\text{RT}_{\texttt{EIG}}$	$\text{RT}_{\texttt{SMA}}$
Moon	15.01	22.30	201.00	2.95	1.10	27.63	140.10	0.03s	48.00	2.30	1.10
Spiral1	192.00	498.60	250.20	99.00	47.05	229.80	223.30	0.07	82.00	3.10	1.40
Cancer	55.30	75.00	306.00	30.01	1.65	100.20	38.90	0.03	16.00	4.00	1.80
Wine	25.30	35.31	279.00	32.31	2.40	25.60	20.40	0.03	74.00	1.30	0.09
RCV	63.00	316.20	339.00	672.60	3.21	56.00	40.67	0.03	72.00	5.50	2.10
Face	15.10	171.60	315.00	233.40	1.10	93.60	35.09	0.10	76.80	2.20	0.96

Table 5: The **out-of-sample preprocessing (Prep) and runtime (RT)** for all benchmark algorithms are displayed in seconds. The table demonstrates that KNet's speed is comparable to competing methods.

	$\text{KNet}_{\texttt{EIG}}$			$\text{KNet}_{\texttt{SMA}}$		
λ	**HSIC**	**AE error**	**NMI**	**HSIC**	**AE error**	**NMI**
10^0	98.11	21.58	0.90	92.01	8.98	0.89
10^{-1}	98.80	72.95	0.88	94.21	72.62	0.87
10^{-2}	112.32	101.45	0.87	101.11	88.39	0.89
0.005	109.01	105.37	0.91	108.22	107.22	0.90
10^{-4}	113.18	124.56	0.91	110.18	126.63	0.91
0	110.89	127.23	0.92	109.36	127.11	0.91

Table 6: HSIC, AE reconstruction error, and NMI at convergence of KNet on the Wine dataset, as a function of λ.

poorly on this dataset. In contrast, the increasingly convex cluster images learned by KNet, as shown in Fig. 1(c), lead to much better separability. This is consistently observed for both the Moon and Spiral1 dataset, for which KNet achieves NMI; we elaborate on this further in Appendix E. demonstrate KNet's ability to generate convex representations even when the initial representation is non-convex.

We also note that KNet consistently outperforms spectral clustering. This is worth noting because, as discussed in Sec. 4, KNet's initialization of both Ψ and U are tied to the spectral embedding. Table 2 indicates that alternatively learning both the kernel and the corresponding spectral embedding indeed leads to improved clustering performance.

Table 3 shows the time performance of each algorithm. In terms of total time, KNet is faster than AEC and DEC. We also observe that SN is faster than most algorithms in terms of run time. However, SN does require extensive hyperparameter tuning to reach the reported NMI performance (see App. F). We note that a significant percentage of the total time for KNet is spent in the preprocessing step, with $\text{KNet}_{\texttt{SMA}}$ being faster than $\text{KNet}_{\texttt{EIG}}$. This is due to the initialization of Ψ, i.e., training the corresponding autoencoder. Improving this initialization process could dramatically speed up the total runtime. Alternatively, as we discuss in the next section, using only a small subset to train the embedding and clustering out-of-sample can also significantly accelerate the total runtime, without a considerable NMI degradation.

Out-of-Sample Clustering. We report out-of-sample clustering NMI performance in Table 4; note that SC cannot be executed out-of-sample. Each algorithm is trained using only a subset of samples, whose size is indicated on the table's first column. Once trained, we report in the table the clustering quality of applying each algorithm to the full set without retraining. We observe that, with the exception of RCV, KNet clearly outperforms all benchmark algorithms in terms of clustering quality. This implies that KNet is capable of generalizing the results by using as litle as 6% of the data.

By comparing Table 2 against 4, we see that AEC, DEC, and IMSAT suffer a significant drop in performance, while KNet suffers only a maximum degradation of 3%. Therefore, training on a small subset of the data not only yields high-quality results, the results are almost as good as training on the full set itself. Table 5, reporting corresponding times, indicates that this can also lead to a significant acceleration, especially of the preprocessing step. Together, these two observations indicate that KNet can indeed be applied to clustering of large non-convex datasets by training the embedding on only a small subset of the provided samples.

7 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 discovers an approximation of its embedding through a DNN. Furthermore, experimental results have confirmed that KNet can be trained using only a subset.

Acknowledgments We would like to acknowledge support for this project from NSF grant IIS-1546428. We would also like to thank Yale Chang and Aria Masoomi for their insightful discussions.

References

[1] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, *Measuring statistical dependence with Hilbert-Schmidt norms*, International Conference on Algorithmic Learning Theory (2005), pp. 63–77.

[2] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, *Auto-encoder based data clustering*, Iberoamerican Congress on Pattern Recognition (2013), pp. 117–124.

[3] P. Ji, T. Zhang, H. Li, M. Salzmann and I. Reid, *Deep Subspace clustering Network*, Advances in Neural Information Processing Systems (2017).

[4] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Liu, *Learning deep representations for graph clustering.*, AAAI (2014), pp. 1293–1299.

[5] J. Xie, R. Girshick, and A. Farhadi, *Unsupervised deep embedding for clustering analysis*, International Conference on Machine Learning (2016), pp. 478–487.

[6] X. Guo, X. Liu, E. Zhu, and J. Yin, *Deep Clustering with Convolutional Autoencoders*, International Conference on Neural Information Processing (2017), pp. 373–382.

[7] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, *Learning Discrete Representations via Information Maximizing Self Augmented Training*, arXiv preprint arXiv:1702.08720 (2017).

[8] U. Shaham, K. Stanton, H. Li, R. Basri, B. Nadler and Y. Kluger, *SpectralNet: Spectral Clustering using Deep Neural Networks*, International Conference on Learning Representations (2018).

[9] A. G. Wilson, Z. Hu, R. Salakhutdinov and E. P. Xing, *Deep kernel learning*, Artificial Intelligence and Statistics (2016), pp. 370–378.

[10] A. G. Wilson, Z. Hu, R. Salakhutdinov and E. P. Xing, *Stochastic variational deep kernel learning*, Advances in Neural Information Processing Systems (2016), pp. 2586–2594.

[11] A. G. Wilson, D. A. Knowles, and Z. Ghahramani *Gaussian process regression networks*, arXiv preprint arXiv:1110.4411 (2011).

[12] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, B. and Schölkopf, *Learning with local and global consistency*, Advances in neural information processing systems (2004), pp. 321–328.

[13] D. Niu, J. Dy, and M. Jordan, *Dimensionality reduction for spectral clustering*, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (2011), pp.552–560.

[14] C. Wu, S. Ioannidis, M. Sznaier, X. Li, Xiangyu,D. Kaeli, David and J. Dy, *Iterative Spectral Method for Alternative Clustering*, International Conference on Artificial Intelligence and Statistics(2018), pp. 115–123.

[15] C. Wu, Z. Khan, Y. Chang, S.Ioannidis, and J. Dy, *Deep Kernel Learning for Clustering* arXiv:1908.03515 (2019).

[16] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, *Spectral grouping using the Nystrom method*, IEEE transactions on pattern analysis and machine intelligence (2004), pp. 214–225.

[17] L. Song, A. Smola, A. Gretton K. M. Borgwardt, and J. Bedo, *Supervised feature selection via dependence estimation*, Proceedings of the 24th international conference on Machine learning (2007), pp. 823–830.

[18] M. Vladymyrov, and M. Carreira-Perpiñán, *The Variational Nystrom method for large-scale spectral problems*, International Conference on Machine Learning (2016), pp. 211–220.

[19] P. A. Absil, R. Mahony and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press,Princeton, NJ ,(2008)

[20] Z. Wen and W. Yin, *A feasible method for optimization with orthogonality constraints*, Mathematical Programming (2013), pp. 397–434.

[21] R. A. Horn, and R. A. Horn, and C. R. Johnson *Matrix analysis*, Cambridge university press (1990).

[22] W. H. Wolberg, *Wisconsin breast cancer dataset*, University of Wisconsin Hospitals (1992).

[23] D. Dheeru, and E. Karra Taniskidou, year = "2017", *UCI Machine Learning Repository*, http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences.

[24] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, *The UCI KDD archive of large data sets for data mining research and experimentation*, ACM SIGKDD Explorations Newsletter (2000), pp. 81–85.

[25] Y. Ding, Y. Zhao, X. Shen, M. Musuvathi, and T. Mytkowicz, *Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup*, International Conference on Machine Learning (2015), pp. 579–587.

[26] A. Y. Ng, M. I. Jordan, and Y. Weiss, *On spectral clustering: Analysis and an algorithm*, Advances in Neural Information Processing Systems (2002), pp. 849–856.

[27] A. Strehl and J. Ghosh, *Cluster ensembles—a knowledge reuse framework for combining multiple partitions*, Journal of machine learning research (2002), pp. 583–617.

[28] X. H. Dang and J. Bailey, *Generation of alternative clusterings using the cami approach*, Proceedings of the 2010 SIAM International Conference on Data Mining (2010), pp. 118–129.

[29] J. Ross and J. Dy, *Nonparametric mixture of Gaussian processes with constraints*, International Conference on Machine Learning (2013), pp. 1346–1354.

[30] C. L. Li, W. C. Chang, Y. Cheng, Y. Yang, and B. Póczos, *MMD GAN: Towards deeper understanding of moment matching network*, Advances in Neural Information Processing Systems (2017), pp. 2203–2213.

[31] A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, Advances in Neural Information Processing Systems (2008), pp. 1177–1184.

[32] C. Musco and C. Musco, *Recursive Sampling for the Nystrom Method*, Advances in Neural Information Processing Systems (2017), pp. 3836–3848.

Appendix A Relating HSIC to Spectral Clustering

Proof. Using Eq. (3.4) to compute HSIC emprically, Eq (3.5) can be rewritten as

$$Y^* = \arg\max_Y Tr(D^{-1/2}K_X D^{-1/2} H K_Y H), \tag{1.18}$$

where $D^{-1/2}K_X D^{-1/2}$ and K_Y are the kernel matrices computed from X and Y. As shown by [13], if we let K_Y be a linear kernel such that $K_Y = YY^T$, add the constraint such that $Y^T Y = I$ and rotate the trace terms we get

$$[Y^*] = \arg\max_Y Tr(Y^T H D^{-1/2} K_X D^{-1/2} H Y) \tag{1.19a}$$

$$\text{s.t}: \; Y^T Y = I. \tag{1.19b}$$

By setting the Laplacian as $\mathcal{L} = HD^{-1/2}K_X D^{-1/2}H$, the formulation becomes identical to Spectral Clustering as

$$[Y^*] = \arg\max_Y Tr(Y^T \mathcal{L} Y) \tag{1.20a}$$

$$\text{s.t}: \; Y^T Y = I. \tag{1.20b}$$

□

Appendix B Effect of λ on HSIC, AE reconstruction error, and NMI

	KNet$_{\texttt{EIG}}$			KNet$_{\texttt{SMA}}$		
λ	**HSIC**	**AE error**	**NMI**	**HSIC**	**AE error**	**NMI**
10^0	2.989	0.001	0.446	2.989	0	0.421
10^{-1}	2.989	0.0001	0.412	2.989	0	0.414
10^{-2}	2.989	0.0001	0.421	2.989	0	0.418
10^{-3}	2.989	0.001	0.434	2.989	0	0.419
10^{-4}	2.988	0.001	0.421	2.988	0	0.418
10^{-5}	2.965	0.099	0.557	2.979	0.033	0.558
10^{-6}	1.982	0.652	0.644	2.33	0.583	0.56
10^{-7}	2.124	0.669	0.969	2.037	0.66	0.824
10^{-8}	2.249	0.69	1	2.368	0.67	1
0	2.278	0.715	1	2.121	0.718	1

Table 7: HSIC, AE reconstruction error, and NMI at convergence of KNet on the SPIRAL1 dataset, as a function of λ.

Appendix C Derivation for Eq. (4.11)

By ignoring the reconstruction error, the loss from Eq. (4.8) becomes

$$\mathbb{H}(\Psi_\theta(X), U). \tag{3.21}$$

From [1], this expression can be expanded into

$$Tr(K_{\Psi_\theta(X)} D^{-1/2} H K_U H D^{-1/2}). \tag{3.22}$$

Since $D^{-1/2}HK_U HD^{-1/2}$ is a constant matrix, we let it equal to Γ, and the objective becomes

$$Tr(\Gamma K_{\Psi_\theta(X)}). \tag{3.23}$$

The trace term can be converted into a matrix sum as

$$\sum_{i,j} \Gamma_{i,j} K_{\Psi_\theta(X),i,j}. \tag{3.24}$$

By replacing the kernel with the Gaussian kernel, Eq. (4.11) emerges as

$$\textstyle\sum_{i,j} \Gamma_{i,j} e^{-\frac{1}{2\sigma^2}||\psi(\mathbf{x}_i;\theta)-\psi(\mathbf{x}_j\theta)||^2}, \tag{3.25}$$

Appendix D Normalized Mutual Information

Consider two clustering assignments assigning labels in $\{1,\ldots,c\}$ to samples in dataset $\Omega = \{1,\ldots,N\}$. We represent these two assignments through two partitions of Ω, namely $\mathcal{C} = \{C_\ell\}_{\ell=1}^c$, $\mathcal{C}' = \{C'_\ell\}_{\ell=1}^c$, where

$$C_\ell, C'_\ell \subseteq \Omega, \ell = 1,\ldots,c$$

are the sets of samples receiving label ℓ under each assignment. Define the empirical distribution of labels to be:

$$\mathbf{P}(\ell,\ell') = \frac{|C_\ell \cap C'_\ell|}{N} \quad \text{for all } (\ell,\ell') \in \{1,\ldots,c\}^2. \tag{4.26}$$

The NMI is then given by the ratio

$$\frac{I(\mathcal{C},\mathcal{C}')}{\sqrt{H(\mathcal{C})H(\mathcal{C}')}}$$

where $I(\mathcal{C},\mathcal{C}')$ is the mutual information and $H(\mathcal{C}), H(\mathcal{C}')$ are the entropies of the marginals of the joint distribution (4.26).

Appendix E Moon and Spiral dataset

In this appendix, we illustrate that for the Moon and the Spiral datasets, we are able to learn (a) convex images for the clusters and (b) kernels that produces block diagonal structures. The kernel matrix is constructed with a Gaussian kernel, therefore the values are between 0 to 1. The kernel matrices shown in the figures below use white as 0 and dark blue as 1; all values in between are shown as a gradient between the two colors.

In Figure 4, the Moon dataset X is plotted in Fig. 4(a) and its kernel block structure in Fig. 4(b). After training Ψ, the image of Ψ is shown in Fig. 4(c) along with its block diagonal structure in Fig. 4(d). Using the same Ψ trained on X, we distorted X with Gaussian noise and plot it in Fig. 5(a) along with its kernel matrix in Fig. 5(b). We then pass the distorted X into Ψ and plot the resulting image in Fig. 5(c) along with its kernel matrix in Fig. 5(d). From this example, we demonstrate KNet's ability to embed data into convex clusters even under Gaussian noise.

In Figure 6, a subset of 300 samples of the Spiral dataset is plotted in Fig. 6(a) and its kernel block structure in Fig. 6(b). After training Ψ, the image of Ψ is shown in Fig. 6(c) along with its block diagonal structure in Fig. 6(d). The full dataset is shown in (a) of Figure 7 along with its kernel matrix in Fig. 7(b). Using the same Ψ trained from Fig. 6(a), we pass the full dataset into Ψ and plot the resulting image in Fig. 6(b) along with its kernel matrix in Fig. 6(d). From this example, we demonstrate KNet's ability to generalize convex embedding using only 1% of the data.

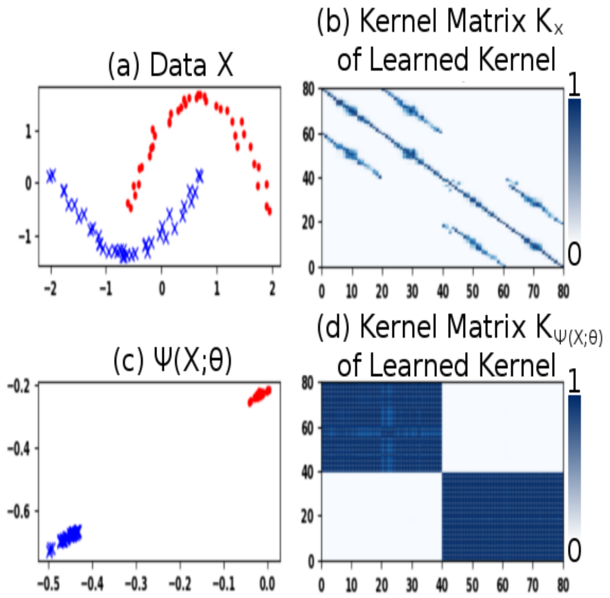

Figure 4: This figure plots out the effect of training Ψ on the Moon dataset. The original data and its kernel matrix is shown in (a) and (b) respectively. The embedding of the data with Ψ and its kernel matrix is shown in (c) and (d) respectively. This figure demonstrates KNet's ability to maps non-convex clustering into convex representation.

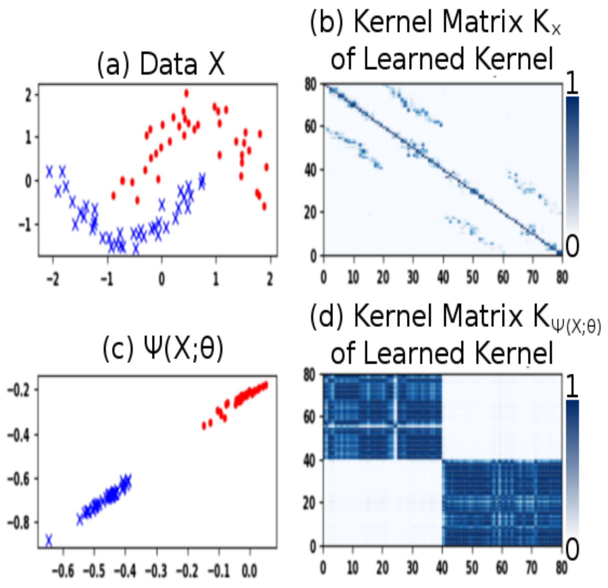

Figure 5: This figure plots out the effect of applying a trained Ψ on a distorted Moon dataset. The distorted data and its kernel matrix is shown in (a) and (b) respectively. The embedding of the distorted data with Ψ and its kernel matrix is shown in (c) and (d) respectively. This figure demonstrates KNet's robustness in mapping non-convex clustering into convex representation under Gaussian distortion.

Appendix F Algorithm Hyperparameter Details

Algorithm	Learning Rate	Batch size	Optimizer
AEC	-	100	SGD
DEC	0.01	256	SGDSolver
IMSAT	0.002	250	Adam
SN	0.001	128	RMSProp
KNet	0.001	5	Adam

Table 8: Here we include the typically used learning rates and batch sizes, and the optimizer type for each algorithm. These were set as recommended by the respective papers, except in the case of AEC which is silent on what learning rate needs to be set, the available implementation sets the learning rate with a line search. We use the above mentioned settings generally and only change them if the batch size is too big for a dataset or we notice the preset learning rate not leading to convergence.

The hyperparameters for each network are set as outlined in the respective papers. In the case of SN, the

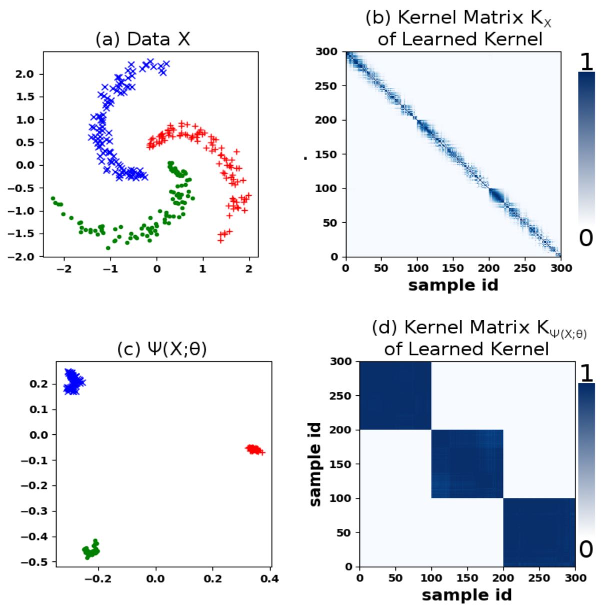

Figure 6: This figure plots out the effect of training and applying Ψ on a small subset of the Spiral dataset. The subset and its kernel matrix is shown in (a) and (b) respectively. The embedding of the subset and its kernel matrix is shown in (c) and (d) respectively. This figure demonstrates KNet's ability to map non-convex cluster into convex representation with only a small subset of the data.

hyper-parameters included the number of neighbors for calculation, the number of neighbors to use for graph Laplacian affinity matrix, the number of neighbors to use to calculate the scale of the Gaussian graph Laplacian, and the threshold for calculating the closest neighbors in the Siamese network. They were set by doing a grid search over values ranging from 2 to 10 and by using the loss over 10% of the training data.

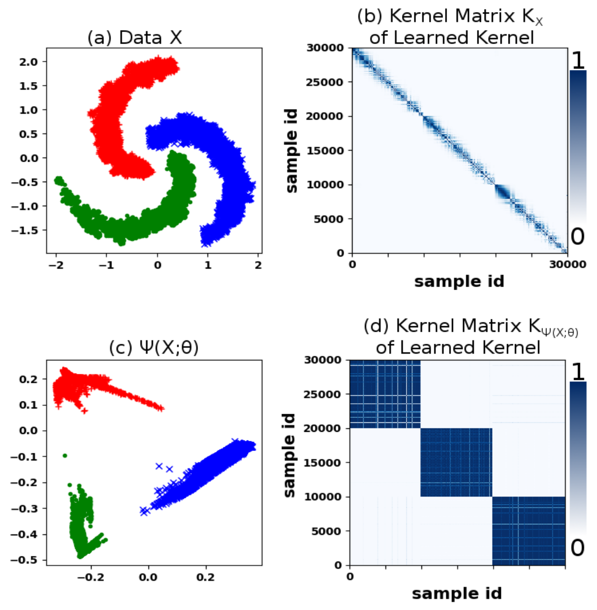

Figure 7: This figure plots out the effect of applying Ψ trained on a small subset to the full Spiral dataset. The full dataset and its kernel matrix is shown in (a) and (b) respectively. The embedding of the full dataset and its kernel matrix is shown in (c) and (d) respectively. This figure demonstrates KNet's ability to generate convex representation on a large scale while training only on a small subset of the data. In this particular case, 1%.