CoreSPECT: Enhancing Clustering Algorithms via an Interplay of Density and Geometry

Chandra Sekhar Mukherjee* **Joonyoung Bae*** **Jiapeng Zhang**
Thomas Lord Department of Computer Science, University of Southern California
*Equal contribution

Abstract

In this paper, we provide a novel perspective on the underlying structure of real-world data with ground-truth clustering via characterization of an abundantly observed yet often overlooked *density–geometry* correlation.

We leverage this correlation to design CoreSPECT (Core Space Projection based Enhancement of Clustering Techniques), a general framework that improves the performance of generic clustering algorithms. Our framework boosts the performance of clustering algorithms by applying them to strategically selected regions, then extending the partial partition to a complete partition for the dataset using a novel neighborhood graph based multi-layer propagation procedure.

We provide initial theoretical support of the functionality of our framework under the assumption of our model, and then provide large-scale real-world experiments on 20 datasets that include standard image datasets as well as genomics datasets.

We observe two notable improvements. First, CoreSPECT improves the NMI of K-Means by 20% on average, making it competitive (and in some cases surpassing) the state-of-the art manifold-based clustering algorithms, while being orders of magnitude faster.

Secondly, our framework boosts the NMI of HDBSCAN by more than 100% on average, making it competitive to the state-of-the-art in several cases *without requiring the true number of clusters and hyper-parameter tuning*. The overall ARI improvements are higher.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

1 INTRODUCTION

Density and geometry have long served as two of the fundamental guiding principles in clustering algorithm design, with algorithms usually focusing either on the density structure of the data (e.g., HDBSCAN (McInnes et al., 2017) and Density Peak Clustering (Rodriguez and Laio, 2014; Wang and Xu, 2017; Yu et al., 2023)) or the complexity of underlying geometry (e.g., manifold clustering algorithms). These paradigms each have some benefits and disadvantages.

While simple-geometry-based algorithm (such as K-Means (Lloyd, 1982)) can be extremely fast, they can have suboptimal performance if the clusters have non-spherical shape. On the other hand, manifold clustering algorithms enjoy stronger theoretical guarantee (Little et al., 2020; Trillos et al., 2023) and empirical performance (Von Luxburg, 2007; Shinde et al., 2024) in the presence of complex geometry, but can be prohibitively slow for large datasets.

On the other hand, while density based clustering algorithms can handle more non-linearity than K-Means while being faster than manifold clustering algorithms and can work without needing the true number of clusters in some case, they often suffer from sensitivity to noise, as well as choice of hyperparameters.

Contrasting with the typical focus on either the density or geometry of data, we observe that in many datasets, the geometric notion of cluster separability is closely correlated with the underlying density. This insight motivates the following contributions.

1.1 Contributions:

1. We observe a novel density-geometry correlation where the central, dense parts of the ground truth clusters are easily separable, surrounded by lower-

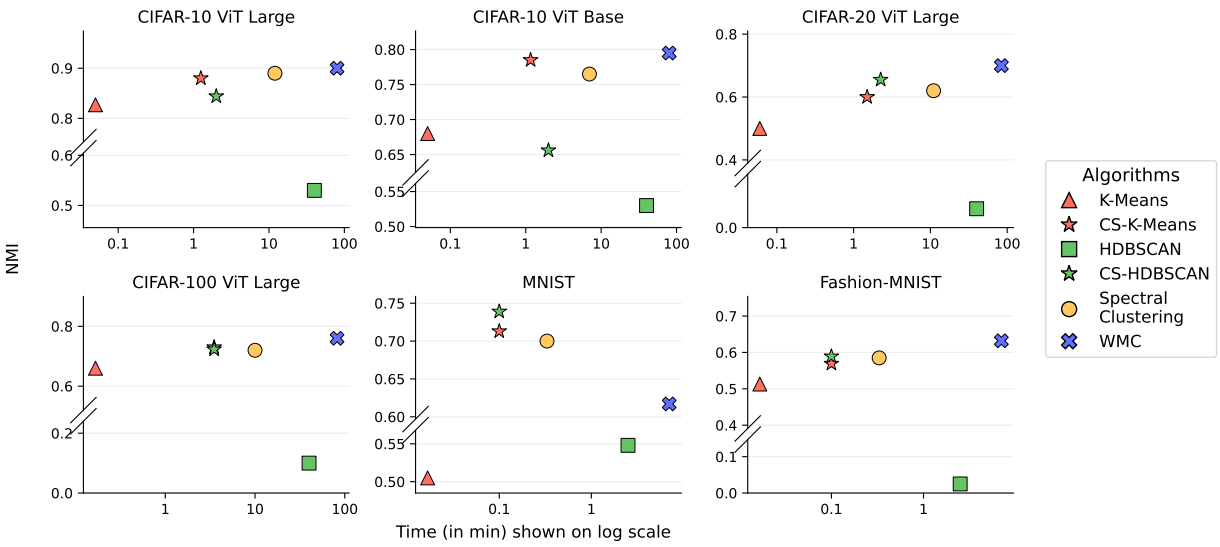

Figure 1: Performance vs. time plots of manifold clustering algorithms and CoreSPECT on image datasets. We observe that both CS-K-Means and CS-HDBSCAN obtain similar performance to the highly efficient implementation of K-NN-based spectral clustering as well as the SOTA manifold clustering algorithm for CIFAR (WMC (Shinde et al., 2024)), while being significantly faster. Impressively, CoreSPECT boosts HDBSCAN to be competitive with the manifold clustering algorithms in several cases, without requiring knowledge of true number of clusters.

density regions that are both harder to separate and have more nonlinear manifold-like geometry.

2. Using this observation we design a 4-step framework that is able to significantly boost the performance of clustering algorithms such as K-Means and HDBSCAN.

3. We test the performance of our algorithm on 19 large datasets, and observe that we make K-Means competitive with the state-of-the-art clustering algorithm, while being 50x faster, and also achieve very strong improvement in HDBSCAN in a **parameter-free** manner, both in performance as well as efficiency.

To the best of our knowledge, an efficient clustering framework using aforementioned density-geometry correlations has not been studied in the literature before. Next, we proceed with a detailed description of our contributions.

1. **Observing and Modeling a novel density-geometry correlation:** We observe that in many datasets with ground truth clusters that are known to have non-linear, manifold like geometry, the clusters contain dense regions that are still well-separable in an Euclidean sense. A primary argument supporting this observation is as follows.

It is well-known that the standard image datasets like MNIST, Fashion-MNIST, and CIFAR-10/100 have a manifold geometry with non-constant low-intrinsic dimensions (Brown et al., 2022). This has motivated design of several manifold-based-clustering algorithms (Von Luxburg, 2007; Trillos et al., 2023; Little et al., 2020; Shinde et al., 2024).

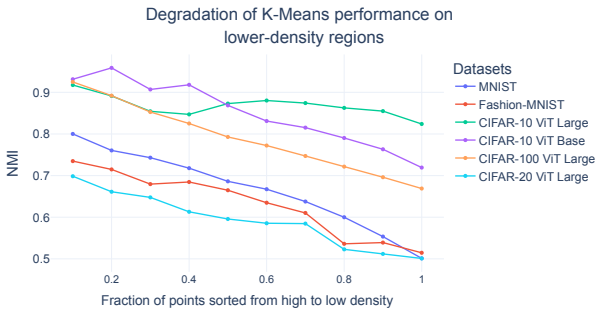

Figure 2: Correlation of K-Means performance and density in image datasets with manifold structures

We observe that for *all* of these datasets, the performance of K-Means on the relatively denser parts of the data (which we identify with our FlowRank algorithm) is significantly higher. This indicates that the denser parts of the clusters are better separable by K-Means, indicating a simpler, more spherical geometry. On the other hand, as we include the sparser parts of the data, the performance of K-Means consistently degrades. This is shown in Figure 2. We obtain for the same observation for the genomics dataset in the Supplementary Material. We term it as Layered Core Periphery based Density model, LCPDM, and describe

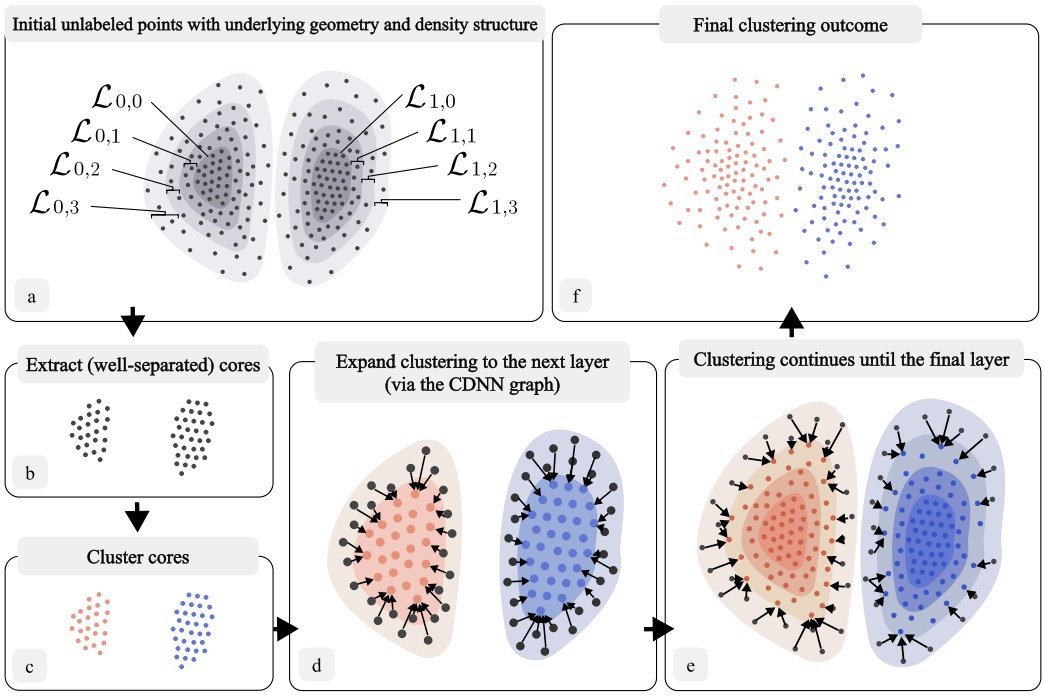

Figure 3: The underlying geometry-density structure in the data (a), and the step-by-step execution of the CoreSPECT framework. In step (b) we extract the cores. Then, in step (c) we cluster the cores with a simple algorithm. Steps (d) → (f) exhibit the layer-wise expansion of clustering using the CDNN graph.

it in detail in Section 2, with more statistical evidence present in Appendix A.

2. **Generic clustering enhancement framework:** Using these insights, we design a generic 4-*step clustering enhancement framework (Figure 4)* targeted at datasets with underlying LCPDM model. In short,

(a) We approximate the layers across the clusters with a notion of *relative centrality* (Mukherjee and Zhang, 2025), which we refine on to design a new algorithm FlowRank (Algorithm 2).

(b) We select the most central layers, which corresponds to the *relatively highest density* regions of the data (which are easily separable as per our model).

(c) Cluster those parts of the data with a generic algorithm (such as K-Means or HDBSCAN).

(d) Expand the clustering layer-by-layer using a novel layer-wise nearest-neighbor graph construction.

We provide a schematic run-through of the framework in Figure 3.

3. **Experiments on large scale genomics and image datasets:**

Importantly, we observe that our observation applies to a large variety of important datasets. As evidence, we test our framework on *19 datasets from three domains* (single-cell RNA-seq, bulk-RNA seq, and popular image datasets), of *size ranging from* $2,000$ *to* $50,000$ *and dimension* 50 *to* 768, on both *geometry based algorithms such as K-Means* (that requires the true number of clusters) as well as *density based algorithms such as HDBSCAN* (that does not require the true number of clusters). We refer the algorithms as CoreSPECTED-K-Means (CS-K-Means) and CS-HDBSCAN.

- *Improving K-means:* Across the datasets, we improve the NMI of K-Means by 18%, with the median improvement being 25%.
- *Improving HDBSCAN:* We improve the NMI of HDBSCAN with default parameters by **over 100%**, with the median improvement of **200%**.
- *Improving run-time of HDBSCAN:* For large datasets ($\geq$ 50,000 points), our framework speeds up HDBSCAN by a factor of 5-10.
- *Matching SOTA manifold clustering algorithms efficiently:* We make K-Means competitive to the performance of the SOTA manifold clustering algorithm (Shinde et al., 2024) in NMI on the popular CIFAR datasets, while using default parameters. Our framework achieves this with 50x faster runtime.

We present the results on 6 image datasets in Figure 1 and 8 genomics dataset in Figure 6. The rest of the results are present in the Appendix. In the rest of the paper, we expand on the aforementioned contributions.

1. **Step 1: Core (and subsequent density layers) extraction (Figure 3 (a), (b)).** We have described in our model that each cluster has some ℓ layers $\mathcal{L}_{i,j}, 0 \leq j \leq \ell - 1$. We first want to obtain $\hat{\mathcal{L}}_0 := \cup_{i=1}^{k} \hat{\mathcal{L}}_{i,0}$, that is, approximately the union of cores of all the clusters. Crucially, we do this *without* any clustering, but rather exploiting the density structure in the hierarchies. We describe this step in Section 2.1.1. We extend this to get an estimate of union of the outer density layers $\hat{\mathcal{L}}_j, 1 \leq j < \ell$. **We use S_j to denote our estimate of $\hat{\mathcal{L}}_j$.**
2. **Step 2: Clustering the core (Figure 3 (c)).** In this step, the cores (S_0) are clustered with a simple algorithm of user's choice. In our explanations and dry run, we use K-Means. We describe the rationale behind this in Section 2.1.2.
3. **Step 3: Core directed nearest neighbor (CDNN) graph construction (Figure 3 (d)).** Next, we design a novel layer-by-layer nearest neighbor graph structure using the estimated density layers S_j and show that it captures the underlying cluster structure of the data better than Euclidean distance. This is described in Section 2.1.3.
4. **Step 4: Layer-wise expansion of clustering (Figure 3 (d)→(f)).** Finally, we use the clustering of the core obtained in Step 2 and the graph in Step 3 to cluster each of the layers in the hierarchy in time *Linear in the number of points, clusters, and nearest neighbors to check.* This is described in Section 2.1.4.

Figure 4: The CoreSPECT Framework (See Figure 3 for a schematic representation and Algorithm 4 for its application to K-Means)

2 MODEL FORMULATION AND THE CORESPECT FRAMEWORK

In this Section we define our model assumptions on a high level. We require certain technical assumptions to prove our initial theoretical guarantees, which are presented in detail in Section C, along with all the proofs.

We assume points of each ground-truth cluster i is being generated from some $\mathcal{X}_i \subseteq \mathbb{R}^d$, where all the points lie on some m-dimensional smooth manifold $\mathcal{M}$. We make the following assumptions on the geometry-density interaction, motivated by our observations in Figure 2.

Concentric subspaces The fundamental assumption we make is that $\mathcal{X}_i$ can be expressed as a hierarchy of concentric subspaces $\mathcal{X}_{i,0} \subset \mathcal{X}_{i,1} \subset \cdots \mathcal{X}_{i,\ell-2} \subset \mathcal{X}_{i,\ell-1} = \mathcal{X}_i$ We call the difference between the j and $j+1$-th subspace as the j-th layer $\mathcal{L}_{i,j}$, defined as

$$\mathcal{L}_{i,0} := \mathcal{X}_{i,0}, \qquad \mathcal{L}_{i,j} := \mathcal{X}_{i,j} \setminus \mathcal{X}_{i,j-1},\ 1 \leq j < \ell$$

Here we assume that each $\mathcal{L}_{i,j}$ are smooth and have a minimum and maximum depth in any direction. We call the $\mathcal{L}_{i,0}$ as *cores* and the outer layers as more peripheral. We assume each layer is connected and has finite volume in a well-defined measure. We make the following assumptions, that we collectively term as the Layered-core-periphery-density-model (LCPDM).

The Layered-core-periphery-density-model LCPDM(k, ℓ):

1. *Cores are dense and peripheries are sparse:* The data is generated by sampling $n_{i,j}$ points from each layer $\mathcal{L}_{i,j}$ uniformly at random, such that there exists a constant $C > 1$ satisfying

$$\frac{n_{i,j}}{Vol(\mathcal{L}_{i,j})} > C \cdot \frac{n_{i,j+1}}{Vol(\mathcal{L}_{i,j+1})}$$

That is, for each cluster, the inner-most layer (the core) is the densest, and the density of the outer layer monotonically go down. We present this schematically in Figure 3 (a). We denote the points generated from $\mathcal{L}_{i,j}$ as $\hat{\mathcal{L}}_{i,j}$.

2. *Cores are well-separated.* The first assumption on the geometry dictates that the core-layers of the clusters are well separated in an Euclidean sense. For any clusters i, i'

$$\exists \mu_{i,i'} < 0.5 \text{ s.t. } \max_{\substack{\mathbf{x} \in \hat{\mathcal{L}}_{i,0} \\ \mathbf{x}' \in \hat{\mathcal{L}}_{i,0}}} \|\mathbf{x} - \mathbf{x}'\| \leq \mu_{i,i'} \cdot \min_{\substack{\mathbf{x} \in \hat{\mathcal{L}}_{i,0} \\ \mathbf{x}' \in \hat{\mathcal{L}}_{i',0}}} \|\mathbf{x} - \mathbf{x}'\|$$

When there are only two clusters in the data, we define $\mu := \mu_{0,1}$.

3. *Layer-wise clustering membership alignment.* There exists $\delta < 1$ such that for any $\mathbf{x} \in \mathcal{L}_{i,j}$ and any $i' \neq i$, $\min_{\mathbf{x}' \in \mathcal{L}_{i,j-1}} \|\mathbf{x} - \mathbf{x}'\| \leq \delta \cdot \min_{\mathbf{x}'' \in \mathcal{L}_{i',j-1}} \|\mathbf{x} - \mathbf{x}''\|$.

That is, any point in a cluster is closer to the boundary of the inner-layer of the same cluster than points in inner-layer of another cluster. Additionally, we do not make any assumptions on the geometry and proximity of points in the outermost layers.

2.1 The CoreSPECT framework

We first give a brief outline of our framework in Figure 4.

2.1.1 Step 1: Extracting the cores (and the subsequent layers)

We first recover the density layers, estimating the core $\hat{\mathcal{L}}_0$ with S_0 that is well-separated in an Euclidean sense, and also the subsequent layers.

Definition 1 (Layer-preserving ranking)**.** *Given a dataset X generated from the* $\mathsf{LCPDM}(k,\ell)$ *model, we say a ranking of the points in X is layer-preserving if for any cluster $\mathcal{X}_i$, if $\mathbf{x} \in \hat{\mathcal{L}}_{i,j}$ and $\mathbf{x}' \in \hat{\mathcal{L}}_{i,j'}$ where $j < j'$, then $\mathbf{x}$ is ranked above x'.*

Our ranking algorithm is motivated by the *relative centrality framework* of Mukherjee and Zhang (2025), designed to obtain the central parts of *each community*. In contrast, we aim to find the *densest regions of each cluster*. In this direction, we obtain a q-NN graph $G_{q,X}$ and obtain the distribution of $\log(n)$ step random walk, denoted as $\Pi(G_{q,X})$, mimicking *initial centrality* of Mukherjee and Zhang (2025). However, we note (and explore in the appendix) that their notion of relative centrality does not capture the density hierarchies in the underlying data. To circumvent this, we define the concept of *relative density*.

Relative density estimation. Given a dataset X with density estimation $\Pi : V \to [0,1]$ for each data point, we look at its r-nearest neighbors, and randomly move to one of the neighbors that have a higher Π value, continuing the process until we reach a maxima, as described in Algorithm 1.

Then, our core-ranking algorithm, *FlowRank* (Algorithm 2) calculates the average between the density of a point, and that of the average maxima reached by randomly ascending random walks starting from that point. We provide the following guarantee for the ranking, with the proof (and the proofs of the following theorems) present in the Appendix.

Theorem 1 (Core-detection by FlowRank)**.** *Let data be generated from the* $\mathsf{LCPDM}(2,\ell)$ *model. Let Π be the density of the space. Then, for some $r = \mathcal{O}(1)$, on expectation, all the core points ($\hat{\mathcal{L}}_0$) get a score of 1. Additionally, all non-core points get a score < 1.*

The proof can be found in Appendix C.3. Here we note that we make certain extra assumptions on the curvature of the manifold for theoretical completeness, which we note down in Appendix C.1.

In practice, we observe selecting top 10% ranked points as the core works well, as it consists of points from each underlying cluster and we will observe the top-points are also very separable. Additionally, we define the next 10% blocks as our proxy for the layers $\hat{\mathcal{L}}_j$. Here we note that in different datasets, the number of layers may vary, however, if the ranking is layer preserving, our partition either breaks a single layer into multiple ones, or only merges multiple *consecutive* layers into one. Obtaining a better approximation of $\hat{\mathcal{L}}_j$ is an interesting direction towards further improving our framework.

Algorithm 1 $\mathsf{RARW}(X, \Pi, r, i)$
(Randomly Ascending Random Walk)

Input: X, $\Pi : X \to [0,1]$ and index i.
Let $N_r(\mathbf{x})$ be the r closest points to $\mathbf{x}$.
Start: $\mathbf{x}^+ \leftarrow \mathbf{x_i}$
while $\Pi(\mathbf{x}^+) < \max_{\mathbf{x}' \in N_r(\mathbf{x}^+)} \Pi(\mathbf{x}')$ **do**
  $N^+ \leftarrow \mathbf{x}' \in N_r(\mathbf{x}^+) : \Pi(\mathbf{x}') > \Pi(\mathbf{x}^+)$
  Randomly select a point $\mathbf{x}''$ from N^+
  $\mathbf{x}^+ \leftarrow \mathbf{x}''$
end while
return $\Pi(\mathbf{x}^+)$

Algorithm 2 FlowRank: $\mathsf{FR}(X, \Pi, q, r)$

Input: X, a $n \times d$ dataset, neighborhood parameter r, density vector Π.

(We obtain the density estimation through a $\log n$-step random walk simulation on $G_{q,X}$.)

for i in 1:n **do**
  $z_i \leftarrow \mathbb{E}[\mathsf{RARW}(X, \Pi, r, i)]$ {Algorithm 1}
  $score[i] \leftarrow \frac{\Pi_i}{z_i}$
end for

2.1.2 Step 2: Applying the clustering algorithm to the core

Here, we formally (as well as experimentally) capture the separability of the cores by K-Means.

Proposition 1. *Let n_0 and n_1 points ($n_0 > n_1$ WLOG) be sampled from $\mathcal{X}_{0,0}$ and $\mathcal{X}_{1,0}$ respectively such that $\mu \cdot n_0 \leq n_1$. These points are denoted $\hat{\mathcal{L}}_{0,0}$ and $\hat{\mathcal{L}}_{1,0}$. Consider the slightly modified K-Means algorithm. Obtain two centers using the K-Means++ method and run one-step K-Means. Repeat this process some $2\log n$ times and accept the result with minimum K-Means objective value. This algorithm separates the two cores correctly with probability $1 - o(1)$.*

The proof can be found at the restated proposition 1 in Appendix C.2

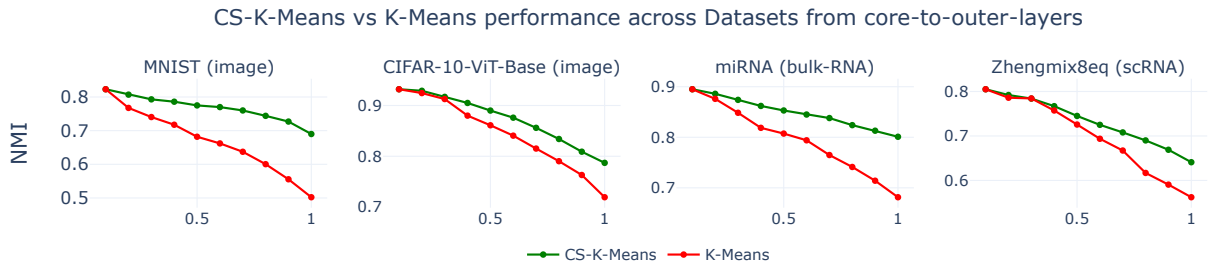

Figure 5: Comparing the accuracy of applying K-Means on the top x-fraction of the points (according to FlowRank) vs. applying K-Means to top 10% and then applying Layer-wise expansion (Algorithm 3) up to top x-fraction, for $x \in [0.1, 0.2, \ldots, 1]$ for some pairs of clusters. For $x = 0.1$ (the cores), K-Means has very good performance, but as we apply K-Means on the outer layers, its performance deteriorates. In contrast, the layer-wise propagation leads to significantly lower decay, leading to improvement in the overall performance.

Algorithm 3 Layer-wise-expansion of clustering: $\mathsf{Expansion}(S, G^+, W, \mathcal{C})$

Inputs: The CDNN graph $G^+_{t,S}$ (with weight matrix W), layers S, cluster membership vector $\mathcal{C} : S_0 \to \mathbb{R}^k$.
Initiate clustering: $V_1, \ldots, V_k : \forall \mathbf{x} \in S_0, \mathbf{x} \in V_t$ where $t := \arg\min_i \mathcal{C}[\mathbf{x}]_i$.
Define data structure $\hat{C}$: $\forall \mathbf{x} \in S_0, \hat{C}(\mathbf{x}) \leftarrow \mathcal{C}(\mathbf{x})$
for $j \in 1 : \ell$ **do**
  for $\mathbf{u} \in S_j$ **do**
    $\hat{C}(\mathbf{u}) \leftarrow \sum_{v \in N_{G^+}} W(\mathbf{u}, \mathbf{v}) \cdot \hat{C}(\mathbf{v})$;
    $k_u \leftarrow \mathsf{arg\,min}\, \hat{C}(\mathbf{u})$; $V_{k_u} \leftarrow V_{k_u} \cup \mathbf{u}$
  end for
end for
return Clustering $V_1, \ldots, V_k$

Non-linearity in the outer layers: While the cores are separable by K-Means, we observe that as we include the outer layers, the performance of K-Means degrades. This can be attributed to the fact that the shape of the data becomes more non-linear as we move away from the center. We observe that as we move away from the center, shortest-path-distance on a nearest-neighbor embedding becomes an increasingly better estimate of cluster membership compared to the Euclidean distance. This is shown in Appendix A.1. However, shortest-path-based approaches can be computationally expensive even with approximations (Little et al., 2020; Trillos et al., 2023).

Instead, we exploit the density structure and build a graph that allows us to *efficiently* cluster the rest of the points building on our third model assumption that cluster membership of points in outer layers are better captured by nearby points in inner layers.

Algorithm 4 CoreSPECTed-K-means

Inputs: Dataset X and number of clusters k, hyperparameters: r, ℓ, t.

Step 1: Obtain $\Pi : X \to [0, 1]$ a random walk based estimate of the density of each point.
$F \leftarrow \mathsf{FlowRank}(X, \Pi, q, r)$ {Algorithm 2}
$\forall j \in \{0, \ldots, \ell - 1\}$, define S_j as the top $(j + 1)/\ell$ fraction of points as per F.

Step 2: Cluster S_0 with K-Means, obtaining centroids $\mathbf{c_i}$.
Define cluster membership vector $\mathcal{C} : S_0 \to \mathbb{R}^k$: $\mathcal{C}(\mathbf{u}) = [\|\mathbf{u} - \mathbf{c_i}\|]_{1 \le i \le k}$

Step 3: Generate the CDNN graph $G^+_{t,S,X}$ and normalized weight matrix W {As per Definition 2}

Step 4: $V \leftarrow \mathsf{Expansion}(S, G^+, W, \mathcal{C})$ {Algorithm 3}

return Clustering $V_1, \ldots, V_k$

2.1.3 Step 3: Creating the Core Directed Nearest Neighbor (CDNN) graph

Our model assumption dictates any point in $\hat{\mathcal{L}}_{i,j}$ will be close to the points of the neighboring inner layer of the same ground truth cluster, irrespective of other distances. This motivates the following graph embedding.

Definition 2 (Centrally directed nearest neighbor graph (CDNN) $G^+_{(t,S)}$)**.** *For every datapoint in S_j (the layers obtained in the previous step), we connect it to some t nearest neighbors in $\cup_{j'=0}^{j-1} S_{j'}$.*[1]

[1]Ideally, we want a graph that connects vertices in S_j to vertices in S_{j-1}, but we choose the mentioned formulation to limit propagation of error in the FlowRank outcome.

We observe that distance on the CDNN graph is also more reliable than Euclidean distances. However, due to the layer-wise structure of the CDNN graph, we are able to devise a very fast algorithm to cluster the rest of the points.

2.1.4 Step 4: Expanding the clustering in a layer-wise manner

We instead expand the clustering a layer at a time. That is, given a clustering of the points in S_0, we label the points in S_1 using the edges going from S_1 to S_0, continuing this process layer-wise.

For this, we need two ingredients. First, for each point in S_0, we define a k-dimensional vector $\mathcal{C}$(each entry corresponding to a cluster) that gives us the membership value of the point to the cluster according to the clustering algorithm we apply to the core. For example, it can be the distance to the k centroids for K-Means.

Next, we define a weight function for each edge in $G^+_{t,S}$ that is inversely proportional to the distance. For this paper we use UMAP's weight function (McInnes et al., 2018), and observe the behavior w.r.t. different choices in the supplementary material. We use these two vectors to cluster the subsequent layers one at a time, which we describe in algorithm 3. The performance of this propagation step is captured in Figure 5. Finally, we record the computational efficiency of this step.

Theorem 2. *Given the CDNN graph $G^+_{t,S}$ and a clustering of S_0, the rest of the points can be clustered in $\mathcal{O}(n \cdot k \cdot t)$ time, which is linear in the number of the edges and the number of clusters in $G^+_{t,S}$.*

The proof can be found at the restated theorem 2 in Appendix C.4

2.2 Combining the framework explicitly for K-Means

Here, we write down the different steps of our framework when applied to K-Means as Algorithm 4.

Theoretical guarantee of CoreSPECTed-K-Means. Finally, we note that as long as each of our steps is approximately correct, CoreSPECTed-K-Means almost correctly recovers the underlying clusters in data that follows our model, irrespective of the proximity of points in the outer layers.

Theorem 3 (Clustering in the LCPDM model)**.** *Let X be n datapoints generated from the* LCPDM$(2, \ell)$ *model. Let us have an estimate of the density layers, given as $S_0, \ldots, S_{\ell-1}$ such that $|S_j \cap \hat{\mathcal{L}}_j| = (1-f)|\hat{\mathcal{L}}_j|, j > 0$. for a sufficiently small function $f = o(1)$ that is layer preserving. Then, applying a variant of K-Means to S_0 and expanding the clustering using a CDNN graph (with correctly chosen parameters) results in clustering with o(1) misclassification error rate on expectation.*

The proof can be found at the restated theorem 3 in Appendix C.4

Computational efficiency We note that our framework is efficient. The main computation involves generating nearest neighbor graphs (both for initial density estimation as well as the CDNN graph generation), for which we used the ultra-fast HNSW (Malkov and Yashunin, 2018) library for approximating the nearest neighbors. Our framework is also linearly dependent on k, the target number of clusters. For example, for CoreSPECTed-K-Means, our framework terminates in a few seconds for most of the datasets, taking around a minute for the TM dataset, which is a ≈ 50-dimensional dataset consisting of $\approx 50,000$ points and 54 underlying clusters. We provide a detailed study of the run time (along with asymptotic runtime) of individual steps and scope for improvement in the Appendix C.5.

Ablation studies Here we have provided a run-through as well as initial theoretical support of our framework w.r.t. K-Means. In Appendix B, we provide several ablation studies to further test and support several of our algorithmic design choices. Specifically,

i) We test the relative usefulness of FlowRank compared to that of relative centrality coined in Mukherjee and Zhang (2025). We observe that FlowRank has noticeably better performance.

ii) We also test the utility of our CDNN based clustering expansion compared to that of propagation methods popular in semi-supervised-learning, as it is a natural candidate. Our experiments showed that in many datasets the CDNN-based propagation method outperforms popular propagation methods.

iii) Finally, we also test the usefulness of the weighting function in our propagation mechanism.

3 LARGE SCALE EXPERIMENTS

In this Section we compile our real-world experimental results in detail. We focus on 19 datasets. This includes the 6 image datasets (MNIST, FMNIST, CIFAR10-ViT-Large, CIFAR10-ViT-Base, CIFAR20-ViT-Large and CIFAR100-ViT-Large), 11 single-cell RNA-sequencing datasets, and 2 bulk-RNA sequencing datasets. We also test a large dataset, extended-MNIST consisting of almost 700k points, to demonstrate the scalability of our framework.

Appendix D contains a detailed description of the datasets.

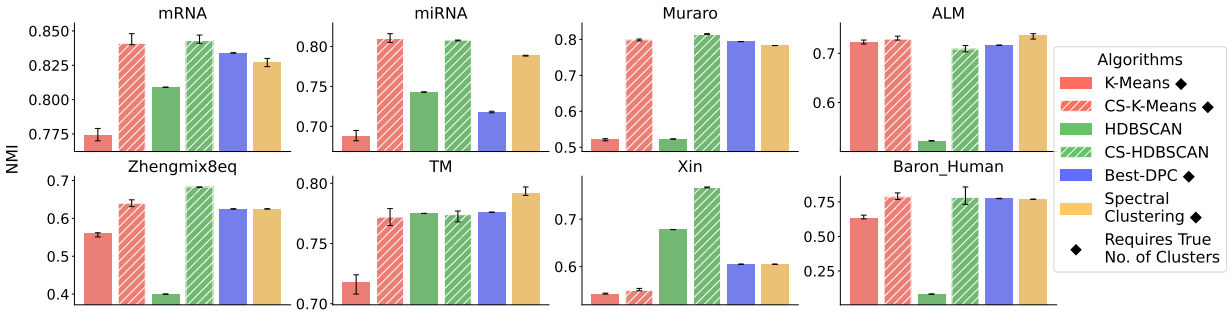

Figure 6: Improvement of NMI on K-Means and HDBSCAN due to CoreSPECT, compared to best density-peak-based clustering as well as spectral clustering. Impressively, CS-HDBSCAN performs on par (sometimes even being the best) compared to popular algorithms that need the true number of clusters.

3.1 Improvement in K-Means and HDBSCAN due to CoreSPECT

Observation (Improvement due to CoreSPECT)**.** *Overall, CoreSPECT improves the ARI of K-Means on 18/19 datasets, and NMI for 19/19 datasets with all hyperparameters fixed to default. On average,* **ARI improves by 40.82%** *and* **NMI improves by 18.51%**.

For CS-HDBSCAN the performance improvement is even more substantial. HDBSCAN is known to be sensitive to noise, and as such has poor performance for most of the datasets considered here. However, the performance of HDBSCAN is significantly higher in the core, which again justifies our modeling. We observe an **average ARI improvement of 468%** *and an* **average NMI improvement of 132%**. *Surprisingly, in the genomics dataset CS-HDBSCAN achieves the best NMI ranking, and is also close to the SOTA manifold clustering algorithm on image datasets, both being significantly faster, and not requiring true number of clusters.*

3.2 Comparison with other algorithms

We have observed that our framework consistently improves the performance of K-Means and HDBSCAN. Next, we want to compare the relevance of the improved performance w.r.t. more complex density-based and geometry-based clustering algorithms.

For the image datasets, we focused on the k-nn-based spectral clustering, and the recent manifold-based-clustering (Shinde et al., 2024) that shown to be near-optimal to even deep-clustering algorithms in image datasets.

For the genomics datasets, we select three recent or/and popular Density Peak Clustering algorithms the original DPC (Rodriguez and Laio, 2014), ADPclust (Wang and Xu, 2017), and a fast implementation of another popular DPC algorithm through a fast framework named PECANN (Yu et al., 2023). Additionally we also use HDBSCAN (Campello et al., 2013). We also attempted to use DBSCAN (Ester et al., 1996) and OPTICS (Ankerst et al., 1999), but they categorized most points as outliers for most datasets so we omit them. We show the results in Figure 6. We also use the EM-based GMM-fit algorithm (Dempster et al., 1977). We also obtain a ranking of the NMI and ARI across the genomics datasets. The results are shown in Table 2. Here we again emphasize that our results are obtained by running CoreSPECT with the *same hyperparameters across all datasets.*

3.3 Scalability of CoreSPECT

So far, we have presented the performance of CoreSPECT on K-Means and HDBSCAN on datasets with around 50,000 points. A crucial question is to ascertain the scalability of our framework on larger datasets. In this direction, we study the behavior of CoreSPECT when applied to the extended MNIST dataset (Cohen et al., 2017), consisting of roughly 700k points and 62 latent classes. This is a recent dataset compared to MNIST and Fashion-MNIST, and is known to be harder to cluster than the latter, as demonstrated by the baseline performance of K-Means (0.5 in Fashion-MNIST vs 0.47 in E-MNIST).

To demonstrate our scalability, we run two experiments, one for randomly sampled 350k points , and one for the whole dataset. We share the results in Table 1.

Our results show that the performance of CoreSPECT scales *almost linearly* for both K-Means and HDBSCAN. This can be attributed to the fact that the primary complexity of our framework (aside from the clustering algorithm itself) is in obtaining various approximate nearest neighbor graphs. For this task, we use the HNSW (Malkov and Yashunin, 2018) framework, which has excellent scalability in practice. Furthermore, our framework improves the run-time of HDBSCAN signifi-

	350k Points		698k Points	
Method	**NMI**	**Time**	**NMI**	**Time**
K-Means	0.451	1.4 min	0.474	3 min
CS-K-Means	0.613	23 min	0.624	50 min
HDBSCAN	—	>12 hr	—	>12 hr
CS-HDBSCAN	0.607	67 min	0.621	140 min
WMC	—	>12 hr	—	>12 hr

Table 1: Testing the performance of CoreSPECT on large-scale datasets.

cantly, as the algorithm is only applied to a smaller set of points (10% in our case). These experiments further support the usefulness of our framework for real world datasets.

3.4 Parameter selection process for CoreSPECT

Our framework uses some hyperparameters. Specifically, we have 3 neighborhood selection parameters for different stages q, r, t and thresholds for defining the layers. To ensure a fair comparison, we tested some common neighborhood values on two datasets MNIST and miRNA, and found that $q = 40$, $r = t = 20$ works well here. Then, we applied the *same hyperparameters* on all the other 18 datasets. The hyperparameters for each of these datasets could be further tuned to obtain better results, but we use these default hyperparameters throughout to design an unsupervised learning algorithm that can be generally applicable.

Table 2: NMI and ARI rank of CoreSPECTed-K-Means and HDBSCAN on 13 genomics datasets. The best value is in bold, and the second best is underlined.

	K-Means	CS-K-Means	HDBSCAN	CS-HDBSCAN	GMM
ARI	5.80	**2.73**	6.87	<u>3.00</u>	3.93
NMI	6.00	<u>2.80</u>	7.27	**2.60**	4.40

	PECANN	SC	Bi-K-Means	DP	ADPClust
ARI	3.47	4.20	7.07	8.67	9.00
NMI	3.80	**2.60**	7.73	8.53	8.93

4 CONCLUSION

We conclude this paper with a small discussion of limitations and future directions. The most fundamental question lies in Figure 5. Our framework improves the performance of K-means (and HDBSCAN) by propagating the labels in a better way compared to simple K-Means. The limit to which we can further improve this propagation, both in a theoretical and in an applied setting is an outstanding problem in our opinion. Beyond this, we aim to make our framework more efficient with parallelization based implementation. We discuss more limitations in Appendix E.

References

Abdelaal, T., Michielsen, L., Cats, D., Hoogduin, D., Mei, H., Reinders, M. J., and Mahfouz, A. (2019). A comparison of automatic cell identification methods for single-cell rna sequencing data. *Genome biology*, 20:1–19.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60.

Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Technical report, Stanford.

Boissonnat, J.-D., Lieutier, A., and Wintraecken, M. (2019). The reach, metric distortion, geodesic convexity and the variation of tangent spaces. *Journal of applied and computational topology*, 3:29–58.

Brown, B. C., Caterini, A. L., Ross, B. L., Cresswell, J. C., and Loaiza-Ganem, G. (2022). Verifying the union of manifolds hypothesis for image data. *arXiv preprint arXiv:2207.02862*.

Campello, R. J., Moulavi, D., and Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142.

Duò, A., Robinson, M. D., and Soneson, C. (2018). A systematic performance evaluation of clustering methods for single-cell rna-seq data. *F1000Research*, 7.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Federer, H. (1959). Curvature measures. *Transactions of the American Mathematical Society*, 93(3):418–491.

Little, A., Maggioni, M., and Murphy, J. M. (2020). Path-based spectral clustering: Guarantees, robustness to outliers, and fast algorithms. *Journal of machine learning research*, 21(6):1–66.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

McInnes, L., Healy, J., and Astels, S. (2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Mukherjee, C. S. and Zhang, J. (2025). Balanced ranking with relative centrality: A multi-core periphery perspective. In *ICLR*.

Niyogi, P., Smale, S., and Weinberger, S. (2008). Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39:419–441.

Rodriguez, A. and Laio, A. (2014). Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496.

Savas, P., Virassamy, B., Ye, C., Salim, A., Mintoff, C. P., Caramia, F., Salgado, R., Byrne, D. J., Teo, Z. L., Dushyanthen, S., et al. (2018). Single-cell profiling of breast cancer t cells reveals a tissue-resident memory subset associated with improved prognosis. *Nature medicine*, 24(7):986–993.

Shinde, N., Ding, T., Robinson, D., and Vidal, R. (2024). Geometric analysis of nonlinear manifold clustering. *Advances in Neural Information Processing Systems*, 37:128769–128797.

Smith, S. J., Sümbül, U., Graybuck, L. T., Collman, F., Seshamani, S., Gala, R., Gliko, O., Elabbady, L., Miller, J. A., Bakken, T. E., et al. (2019). Single-cell transcriptomic evidence for dense intracortical neuropeptide networks. *elife*, 8:e47889.

Tomczak, K., Czerwińska, P., and Wiznerowicz, M. (2015). Review the cancer genome atlas (tcga): an immeasurable source of knowledge. *Contemporary Oncology/Współczesna Onkologia*, 2015(1):68–77.

Trillos, N. G., He, P., and Li, C. (2023). Large sample spectral analysis of graph-based multi-manifold clustering. *Journal of Machine Learning Research*, 24(143):1–71.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17:395–416.

Wang, X.-F. and Xu, Y. (2017). Fast clustering using adaptive density peak detection. *Statistical methods in medical research*, 26(6):2800–2811.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xiaojin, Z. and Zoubin, G. (2002). Learning from labeled and unlabeled data with label propagation. In *Tech. Rep., Technical Report CMU-CALD-02–107*. Carnegie Mellon University.

Yu, S., Engels, J., Huang, Y., and Shun, J. (2023). Pecann: Parallel efficient clustering with graph-based approximate nearest neighbor search. *arXiv preprint arXiv:2312.03940*.

Zhou, D., Bousquet, O., Lal, T., Weston, J., and Schölkopf, B. (2003). Learning with local and global consistency. *Advances in neural information processing systems*, 16.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

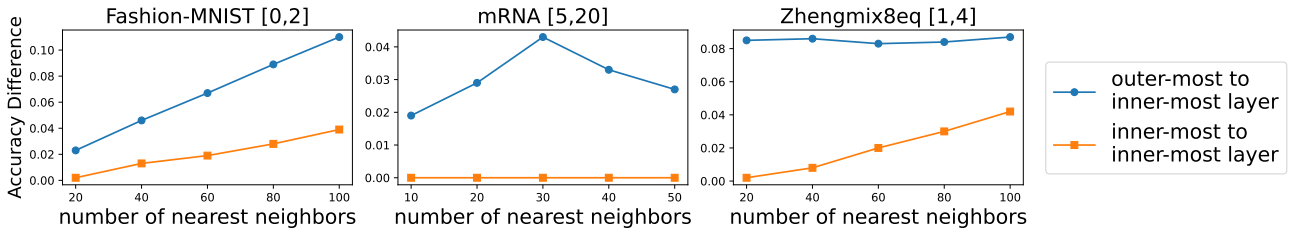

Figure 7: We first sort the nodes by the FlowRank values and select the nodes that are in the top and bottom 20% which we call the nodes from the inner-most and outer-most layers.Then we find the nearest neighbors using Euclidean distance and the shortest path distance in the CDNN graph generated. We show an empirical evidence that the periphery nodes exhibit more non-linear and the core nodes exihibit more euclidean structure by showing the differences of the nearest neighbor label accuracies.

A Statistical Evidence of the LCPDM model in real-world datasets

So far, we have provided initial theoretical support as well as various ablation study to better demonstrate the structural backgrounds and performance of our framework. Here, we provide some more statistical evidence that supports our model formulation and algorithm design. First we describe the datasets we use for evaluation.

Datasets We use a total of 15 datasets. We select the 11 single-cell datasets used by Mukherjee and Zhang (2025) (obtained from Duò et al. (2018); Abdelaal et al. (2019); Smith et al. (2019)), two popular bulk-RNA dataset from the The Cancer Genome Atlas Program (TCGA) (Tomczak et al., 2015), and the popular image datasets MNIST (Deng, 2012) and Fashion-MNIST (Xiao et al., 2017). The details of the datasets are provided in the supplementary material.

First, we show that the inner-most layers are indeed more Euclidean than the outer-most layer in terms of cluster-membership identities.

A.1 Nonlinearity of outer layers captured with relative accuracy of cluster-membership based on different distance measures

In figure 7, We first show the statistical evidence that further strengthens our claim that "the dense regions that are well-separated, while the surrounding outer layers exhibit more non-linear structure".

Specifically, we look at the cluster membership of some q nearest neighbors of points in the inner and outer layers via different notions of distances. To get a high resolution understanding, we focus on the same pairs of clusters that we used to observe the decaying performance of K-Means as points from outer layers were considered.

First, we look at the q nearest neighbors of core points among other core points. We consider the Euclidean distance and the shortest path on the K-NN graph distance. We record the difference in the fraction of intra-cluster points among nearest points according to different metrics. We observe that the accuracy for these two metrics remain relatively unchanged.

In comparison, we observe that if we look at the nearest neighbors of the outer-most layer among the cores, the intra-cluster accuracy according to Euclidean distance is relatively lower compared to shortest path on the CDNN graph that we generate. This implies, that the cluster-membership of the core-core points are well captured by Euclidean distance. In contrast, we need a more local-definition of distance (via shortest path on the CDNN graph) to get cluster-membership-preserving notions of distance.

A.2 Inversion in the behavior of K-Means objective value

We conclude this part with an interesting Phenomena on the K-Means objective value of the clustering obtained by K-Means and CoreSPECTed-K-Means. Essentially, we have the following observation.

1. **On the cores:** The K-Means objective value of the clustering obtained by K-Means on the core is lower than the K-Means objective value of the core points when K-Means was applied to the whole dataset.

Figure 8: We observe that the K-Means objective of the clustering due to CS-K-Means is higher than that of K-Means when looking at the whole data. This contrasts with CS-K-Means having significantly better accuracy. This implies, for the whole dataset, linearly geometric objectives like that of K-Means is not suitable. However, if we focus only on the cores, a different picture emerges. The K-Means objective of the clusters among the core obtained by directly applying K-Means on core is smaller than that of the core-points when K-Means is applied on the whole dataset. This further strengthens the assumptions that the cores obtained by FlowRank are geometrically well-separated, contrasting the more complex geometry of the outer layers.

2. **On the entire dataset:** Here, the K-Means objective value of CoreSPECTed-K-Means is higher than that of K-Means, even though the final clustering by CoreSPECTed-K-Means is of much higher quality, as we have observed.

We capture this in Figure 8. This further strengthens the assumptions that the cores obtained by FlowRank are geometrically well-separated, contrasting the more complex geometry of the outer layers.

B Ablation Studies

In the previous section we have demonstrated that our framework elevates the performance of both K-Means an CS-HDBSCAN across 19 datasets, often outperforming recent density-peak-based and manifold-based clustering algorithms. In this section we provide several ablation studies, directed at understanding the importance of each of the steps in our framework. We start by recollecting the different steps in our framework on a high level in Figure 9

B.1 Approximating the density layers: Comparing FlowRank with relative-centrality methods.

We have discussed in Section D.3 that for some the small datasets, FlowRank fails to add points from the smallest clusters to the cores even when they are well-separated, and identified it as avenue for improvement. In this

1. **Step 1: Core (and subsequent density layers) extraction.**

   In this step we want to obtain the density layers in the data, such that the top-most layer contains the most separable cores and the subsequent layers contain more complex and less separable peripheral points.

   As we have described in the main paper, we design an algorithm that we call FlowRank and then define the deciles as the layers. We aim to find the relatively dense regions of the data to obtain the cores of each cluster. Our method is inspired by the concept of relative centrality in Mukherjee and Zhang (2025).

2. **Step 2: Clustering the core.**

   In this step, the cores (S_0) are clustered with a simple algorithm of user's choice. We use K-Means for this part.

3. **Step 3 and 4: Expanding the clustering to the rest of the points**

   We do this in two steps.

   Step 3: Core directed nearest neighbor (CDNN) graph construction. Here, we design a layer-by-layer nearest neighbor graph structure using the estimated density layers.

   Step 4: Layer-wise expansion of clustering. Finally, we use the clustering of the cores and the CDNN graph to cluster the rest of the points.

   The key idea here is that our model dictates that the cluster membership of a point in a density layer is best determined to its proximity to points in an inner layer (as opposed to overall proximity to points).

Figure 9: The CoreSPECT Framework (on a high level)

direction, we compare the merits of FlowRank w.r.t. the concept *relative-centrality* (Mukherjee and Zhang, 2025) from where we drew inspiration. In Mukherjee and Zhang (2025), the authors focus on K-NN graph embeddings of biological datasets, and then define ranking algorithms such that

i) Top ranked points are better separable into their ground truth clusters compared to the whole data. They verify separability based on what fraction of edges in the induced subgraph of the top-ranked points are *intra-community*.

ii) The top ranked points contain points from all clusters

They verify this property for 11 single-cell datasets (that we also use in our paper). The two properties sought by Mukherjee and Zhang (2025) seems to fit our needs exactly. That is, if the top-ranked points are more separable and we have points from all underlying ground truth clusters, we will be able to obtain a high quality clustering of these points. However, crucially, we have a more specific requirement, which is **the core points should be geometrically well separated**. Note that this is not automatically ensured by the relative centrality methods. Indeed, a look at the algorithms in Mukherjee and Zhang (2025) suggests that they aim to obtain locally-high-centrality points. While these points themselves may be better separable, they may be from very different parts of the space, and therefore, the performance of simple Geometric clustering algorithms like K-Means may not be good. To further investigate this hypothesis, we run our experiments by replacing FlowRank with algorithms of Mukherjee and Zhang (2025). We capture the results in Figure 10.

We observe that FlowRank usually leads to the best result, with significant advantage over the relative centrality methods in 10 out of 15 datasets, while being very similar to the best relative centrality method per-dataset for each of the other 5 datasets.

Essentially, the reason behind this is that the relative centrality methods treat the K-NN graphs to have a single-layer core-periphery structure. However, we find that there are multiple density layers in the real world data, which necessitates our algorithm. However, we believe it is possible further improve our ranking and overall density-layer estimation algorithm, which we consider it as an important future direction.

Figure 10: Contrasting the performance (ARI) of FlowRank with that of Relative centrality methods in the density-layer-extraction step of CoreSPECT. In 11 out of 15 datasets, FlowRank produces the best result, and is within 2% of the best in the other four datasets.

B.2 Layer-wise propagation: A connection (and comparisons) of layer-wise-expansion to semi-supervised learning.

Next, we discuss a very obvious yet interesting connection between our framework and *semi supervised learning*.

In our framework, after we extract the cores and cluster them (which we expect and observe to have high correctness), we use this clustering to extend this to the rest of the data. This is very similar to the concept of *label propagation* (Xiaojin and Zoubin, 2002; Zhou et al., 2003), which is one of the fundamental approaches to semi-supervised learning (SSL). In SSL, given the true labels of some of the points, one wants to label the rest of the points. Therefore, our algorithm can be considered as some sort of a pseudo-semi-supervised-learning framework. This raises the following question.

Question 1. *Can SSL based label-propagation methods extend the clustering of the core as well as our CDNN graph based approach?*

In this direction, we look at popular two scikit-learn method, called Label-propagation (Xiaojin and Zoubin, 2002) and Label-spreading (Zhou et al., 2003). The first is a "hard-clamping" method. That is, the initial labels provided by to the algorithm are never altered. In contrast, Label-spreading is a "soft-clamping" method, where it may readjust some of the initial labels. In short, both the methods aim to create some affinity matrix from the labeled points to the rest of the data. In this direction, the original papers defined an rbf-based kernel for this purpose, which is also the default setting in the scikit-learn implementation. We first tried these two methods to enhance the clustering of the cores by K-Means. Here, we observe that both for label-propagation and label-spreading methods fail, resulting in significantly NMI and ARI compared to K-Means on the whole dataset.

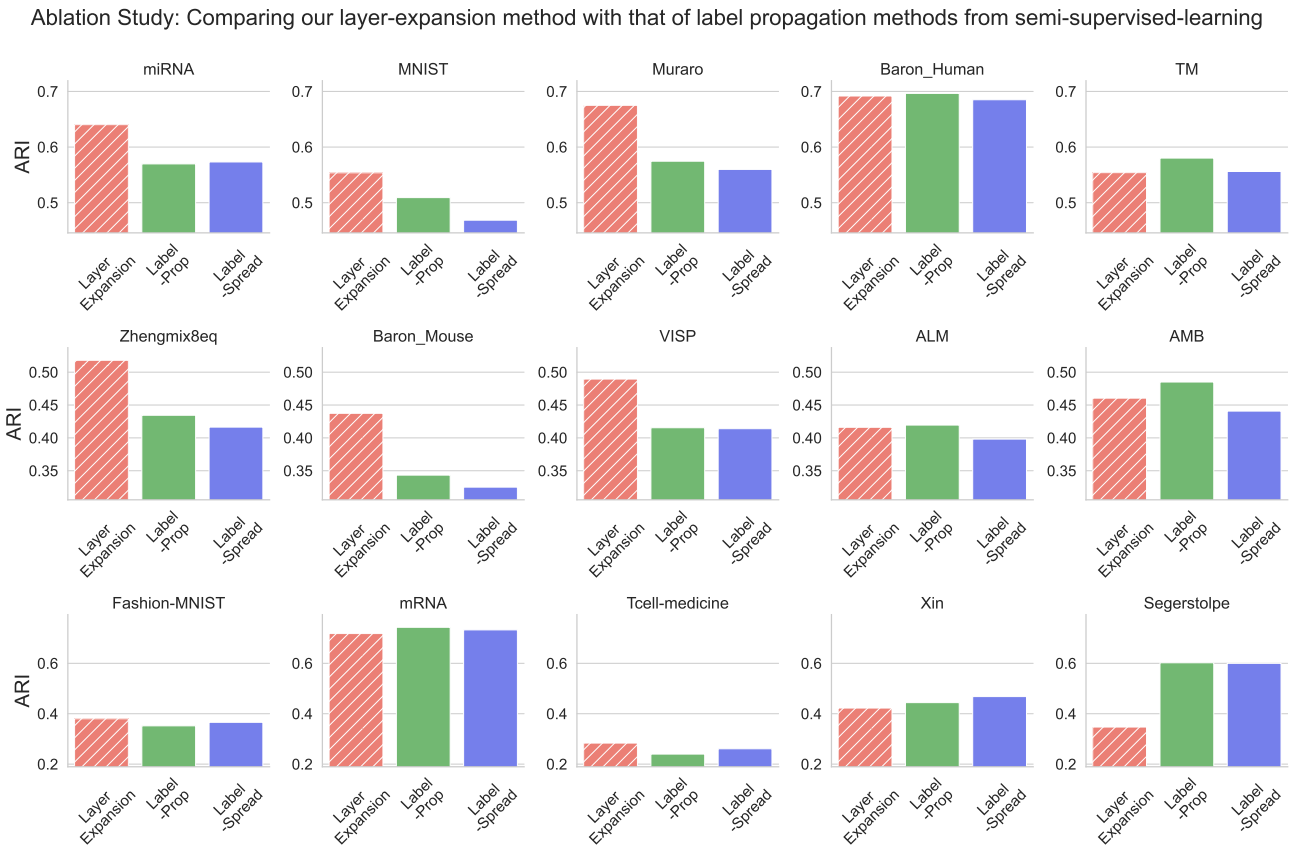

Figure 11: Contrasting the performance of CDNN graph generation+layer expansion with that of popular label propagation algorithms used in Semi supervised learning. Overall, our CDNN-graph based approach has the best rank, with noticeably better performance in 6 out of 15 datasets. We have almost identical performance in 8 more datasets, with doing noticeably worse in only one dataset (Segerstolpe).

K-NN-based SSL label-propagation algorithms To further investigate the possibility of using SSL based prorogation, we use the alternative kernel that uses only nearest neighbors to propagate the initial labels, performing this recursively until all the points are labeled. We place the ARI values of using Label-propagation and label-spread instead of our layer-expansion idea in Figure 11.

We observe that for this setting, both SSL methods perform well, resulting in ARI that is higher than K-Means on the whole dataset in most cases. Overall, our CDNN-based approach still performs the best (except in the Segerstolpe dataset, where both the SSl methods have much better performance). This is a reasonable outcome, as for datasets that are good fit for our density-layer-structure, the cores obtained by FlowRank are the points in the inner-most layers. In contrast, in SSL one often wants to labels from *all regions* of the data, which SSL based labale propagation ideas aim to implicitly exploit.

Ignoring the Segerstolpe dataset (which as we have discussed, does not contain a strong density-layer structure), our approach leads to roughly 7% higher ARI than label-prop and 8% higher than label-spread. On one hand, this further justifies our model formulation and algorithmic framework. On the other hand, this opens up the possibility of further improving our framework by incorporating ideas from more advanced SSL ideas.

We believe better understanding the applicability of such label propagation ideas and appropriating for our framework may lead to even better performance, which is an exciting future direction.

B.3 Choice of weight matrix in the layer-expansion approach.

Next, we briefly recall the structure of our layer-expansion procedure. After we have partitioned the points into deciles (using the FlowRank ranking), for any point in the j-th layer, we find nearest neighbors among the points

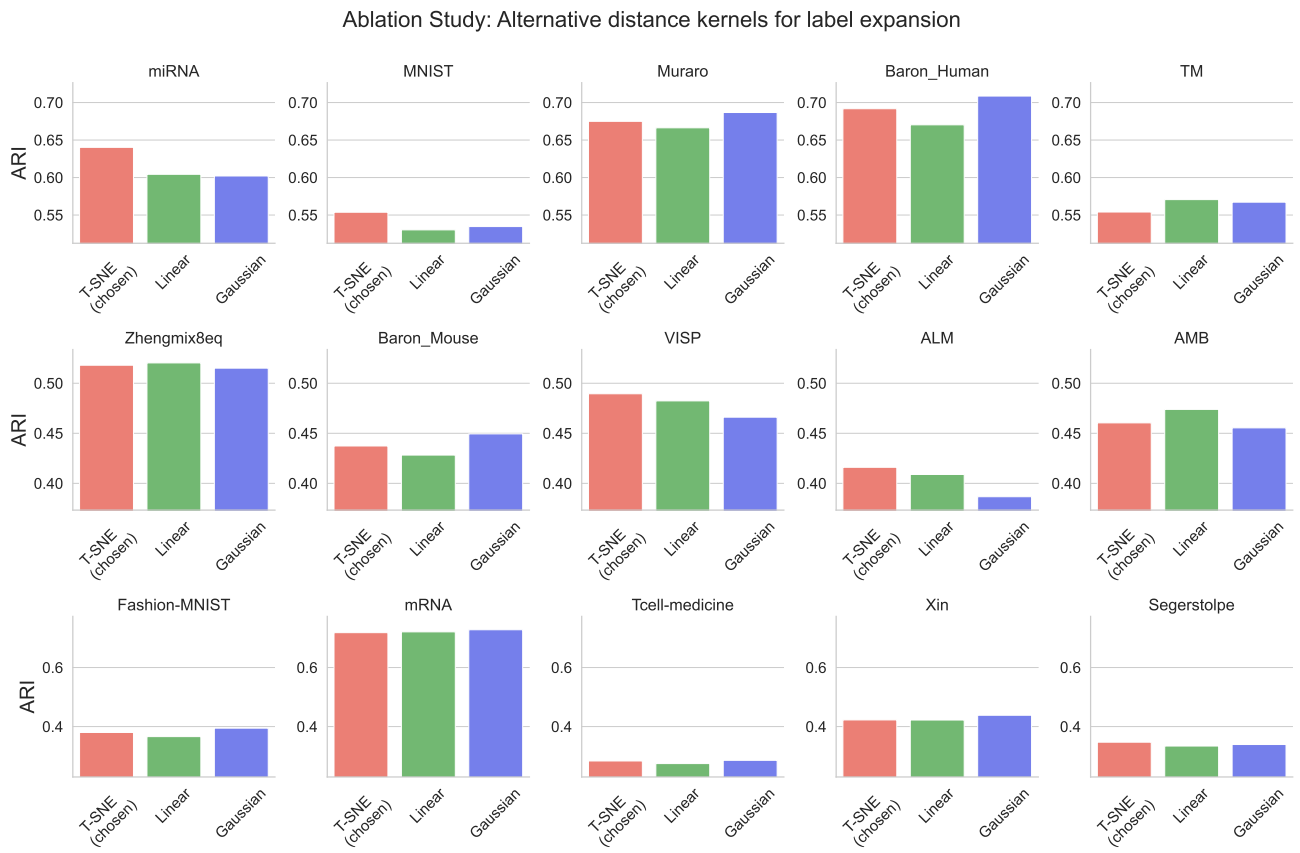

Figure 12: Performance of CoreSPECTed-K-Means for different choice of weight matrix for the layer-expansion

upto $j-1$-th layer (which have been already labeled) and then decide a (normalized) weight function on these edges such that the cluster membership vector of the new point is a weighted sum of the cluster membership vectors of its neighbors. In the results in our main paper, we have used the weight function of UMAP (McInnes et al., 2018) (which itself is inspired from t-SNE (Van der Maaten and Hinton, 2008)) to decide these weights.

Here we perform an ablation study to understand the variability in the performance of CoreSPECT based on the weight function itself (using K-Means as the core-clustering algorithm). We describe the methods we use below.

1. **Chosen setting:** Given a points $\mathbf{x}$ and some t neighbors Z_t, the weight of an edge $\mathbf{x} \to \mathbf{x}'$ is decided as $W(\mathbf{x}, \mathbf{x}') = exp\left(-\left(\|\mathbf{x}-\mathbf{x}'\| - \min_{\mathbf{u}\in Z_t} \|\mathbf{x}-\mathbf{u}\|\right) \Big/ \sigma_x\right)$ such as the sum of all weights is a fixed constant (this is achieved by choosing a different σ_x for each $\mathbf{x}$.
2. **Linear kernel:** In this setting, we simply choose $W(\mathbf{x}, \mathbf{x}') = \frac{1}{1+\|\mathbf{x}-\mathbf{x}'\|}$
3. **Global Gaussian kernel:** This is similar to our chosen setting, without choosing a different σ_x for each point. $W(\mathbf{x}, \mathbf{x}') = exp\left(-\left(\|\mathbf{x}-\mathbf{x}'\| - \min_{\mathbf{u}\in Z_t} \|\mathbf{x}-\mathbf{u}\|\right)^2\right)$

In Figure 12, we present the performance of the different distance kernels. We observe that overall, the t-SNE kernel has very slight advantage over the two methods, being better than linear kernel in 10/15 datasets, and Gaussian kernel in 8/15 datasets. However, we note that the overall differences in the three methods are negligible for most datasets.

C Detailed Model and Complete Proofs

In this Section, we will describe our Layered-core-periphery-density-model $\mathsf{LCPDM}(k, \ell)$ model in further detail and provide initial theoretical support to some of the steps described in the main paper. In doing so, we will also

obtain some insights into the different parameters of our algorithm. First we describe the configuration of our model in detail.

C.1 The LCPDM(k,ℓ) model

As we have described, we have some assumptions on the density and some on the geometry. Here, we describe them in greater detail. We assume points of each ground-truth cluster i is being generated from some $\mathcal{X}_i \in \mathbb{R}^d$. We make the following assumptions on the geometry-density interaction. For the rest of the discussion we focus on the case of $k = 2$. First, we formally define the reach of a manifold, which we use to derive different results.

Definition 3 (Federer (1959))**.** *The reach of a set $\mathcal{S}$ is defined as the infimum of distances between points in S and points in its medial axis, the points in ambient space for which there does not exist a unique closest point in $\mathcal{S}$.*

Detailed model formulation

1. *Globally flat manifold:* We assume the sets $\mathcal{X}_0$ and $\mathcal{X}_1$ lie on a flat m-dimensional smooth manifold $\mathcal{M}$, i.e., $\mathcal{X}_0 \cup \mathcal{X}_1 \subset \mathcal{M}$ with a reach of τ for a fixed m.

2. *Concentric sub-manifolds (with well-defined width):* The fundamental assumption we make is that $\mathcal{X}_i$ can be expressed as a hierarchy of concentric sub-manifolds $\mathcal{X}_{i,0} \subset \mathcal{X}_{i,1} \subset \cdots \mathcal{X}_{i,\ell-2} \subset \mathcal{X}_{i,\ell-1} = \mathcal{X}_i$. Recall that we define $\mathcal{L}_{i,j} = \mathcal{X}_{i,j} \setminus \mathcal{X}_{i,j-1}$.

These concentric sub-manifolds are then defined in the following way. We start with a $\mathcal{X}_{i,0}$ with reach $\tau_{i,0}$. Then, the super-sets are defined as

$$\mathcal{X}_{i,j+1} = \{x : x \in \mathcal{M}, d_{\mathcal{M}}(x, \mathcal{X}_{i,j}) < f_{i,j}(x)\}$$

where $f_{i,j} : \mathcal{M} \to \mathbb{R}$ is a smooth function with $\Delta^{\min}_{i,j} \leq f_{i,j}() \leq \Delta^{\max}_{i,j}$, defining the minimum and maximum width of the layer $\mathcal{L}_{i,j}$. Furthermore, Each layer $\mathcal{L}_{i,j}$ has a reach of $\tau_{i,j}$.

Density assumptions: Each community has a density hierarchy, with the most densest layer being the central one, and the layers being progressively less dense. This is realized as follows. The data is generated by sampling $n_{i,j}$ points from each layer $\mathcal{L}_{i,j}$ uniformly at random, such that there exists a constant $1 < C < 2$ satisfying

$$\frac{n_{i,j}}{Vol(\mathcal{L}_{i,j})} = C \cdot \frac{n_{i,j+1}}{Vol(\mathcal{L}_{i,j+1})}$$

We denote the points generated from $\mathcal{L}_{i,j}$ as $\hat{\mathcal{L}}_{i,j}$ and together let the points be called $\hat{X}$.

Geometric assumptions w.r.t. ground truth cluster

Assumption 1. *1. Cores are well-separated. The first assumption on the geometry dictates that the core-layers of the clusters are well separated in an Euclidean sense. For any clusters i, i'*

$$\exists \mu < 0.5 \quad s.t. \max_{\mathbf{x} \in \mathcal{L}_{i,0}, \mathbf{x}' \in \mathcal{L}_{i,0}} \|\mathbf{x} - \mathbf{x}'\| \leq \mu \cdot \min_{\mathbf{x} \in \mathcal{L}_{i,0}, \mathbf{x}' \in \mathcal{L}_{i',0}} \|\mathbf{x} - \mathbf{x}'\|$$

Furthermore, we assume $\min_{\mathbf{x} \in \mathcal{L}_{i,0}, \mathbf{x}' \in \mathcal{L}_{i',0}} \|\mathbf{x} - \mathbf{x}'\| \leq 0.5\tau$. This is to enforce the flatness of the underlying manifold. However, note that the individual layers themselves may have smaller reach, and thus be harder to cluster using simple algorithms like K-Means.

2. Layer-wise clustering membership alignment. There exists $\delta < 1$ such that for any $\mathbf{x} \in \mathcal{L}_{i,j}$ and any $i' \neq i$, $\min_{\mathbf{x}' \in \mathcal{L}_{i,j-1}} \|\mathbf{x} - \mathbf{x}'\| \leq \delta \cdot \min_{\mathbf{x}'' \in \mathcal{L}_{i',j-1}} \|\mathbf{x} - \mathbf{x}''\|$.

In this direction, we prove three results to provide initial theoretical support to our algorithm. The consistent performance of our framework warrants a more in-depth analysis and understanding and refinement of the model to further understand and exploit these underlying structures, and and we end the Section with a discussion on several such directions.

Against this backdrop, we prove three results. First, we prove that the cores can be clustered easily.

C.2 Clustering the cores

Here we formally define and prove Proposition 1 of Main paper. In fact, we show that for our definition, we do not need a multi-step K-Means++. Rather, we run *one-step* K-Means++ $\Theta(\log n)$ times, and select the outcome with the least K-Means objective score.

First we define the K-Means++ method of Arthur and Vassilvitskii (2006) in brief. In the case of $k = 2$, the centers are chosen as follows. The initial centroid is chosen randomly. Then, the second center is chosen as follows. Any datapoint u is chosen to be the second center with probability $\frac{\|u-c_1\|^2}{\sum_{v\in Y}\|v-c_1\|^2}$. That is, points are chosen with probability proportional to the squared distance to the center. Once the centers are chosen, the usual iterative K-Means algorithm is applied up to some step (or convergence).

Proposition (Restated Proposition 1)**.** *Let n_0 and n_1 points ($n_0 > n_1$ WLOG) be sampled from $\mathcal{X}_{0,0}$ and $\mathcal{X}_{1,0}$ respectively such that $\mu n_0 \leq n_1$. These points are denoted $\hat{\mathcal{L}}_{0,0}$ and $\hat{\mathcal{L}}_{1,0}$. Consider the slightly modified K-Means algorithm. Obtain two centers using the K-Means++ method and run one-step K-Means. Repeat this process some $2\log n$ times and accept the result with minimum K-Means objective value. This algorithm separates the two cores correctly with probability $1 - o(1)$.*

Proof. We condition on the event that the first center selected is from $\hat{\mathcal{L}}_{0,0}$. This happens with probability ≥ 0.5 (as it is the larger set).

Let, the minimum inter-community distance be α. Then, the maximum intra-community distance is $\mu \cdot \alpha$.

Then, the probability that the second center is a point from $\mathcal{X}_{1,0}$ is

$$\begin{aligned}
\Pr(c_2 \in \mathcal{X}_{1,0}) &= \frac{\sum_{\mathbf{x}\in\hat{\mathcal{L}}_{1,0}} \|c_1 - \mathbf{x}\|^2}{\sum_{\mathbf{x}\in\hat{X}} \|c_1 - \mathbf{x}\|^2} \\
&= \frac{\sum_{\mathbf{x}\in\hat{\mathcal{L}}_{1,0}} \|c_1 - \mathbf{x}\|^2}{\sum_{\mathbf{x}\in\hat{\mathcal{L}}_{0,0}} \|c_1 - \mathbf{x}\|^2 + \sum_{\mathbf{x}\in\hat{\mathcal{L}}_{1,0}} \|c_1 - \mathbf{x}\|^2} \\
&\geq \frac{\alpha^2 \cdot n_1}{\mu^2\alpha^2 n_0 + \alpha^2 n_1} \\
&\geq \frac{\alpha^2 n_1}{\mu\alpha^2(\mu n_0) + \alpha^2 n_1} \\
&\geq \frac{\alpha^2 n_1}{\mu\alpha^2 n_1 + \alpha^2 n_1} \geq \frac{1}{\mu+1}
\end{aligned}$$

That is, with probability $\frac{1}{\mu+1}$, the two centers chosen belong to the different ground truth clusters. Let 1_i be the indicator variable for the i-th initialization that is 1 if the two centers are from two clusters. Then, we have

$$\Pr\left(\sum_{i=1}^{2\log n} 1_i > 0\right) = 1 - (\Pr(1_i = 0))^{2\log n} \geq 1 - \left(\frac{2\mu+1}{2\mu+2}\right)^{2\log n} = 1 - o(1)$$

This shows that with high probability, one of the initialization will lead to two centers being chosen from two different ground truth clusters. Then, based on the distance definition, clustering the points by assigning each point to its closest center will lead to a solution with zero misclassification error.

Next, note that the K-Means objective of any solution with two centers being selected from the different ground truth cluster is smaller than the K-Means objective of any solution where both centers are selected from the same cluster. This is because of the following.

Case 1: The K-Means objective value if the two centers are selected from different clusters is upper bounded by $(\mu\alpha)^2(n_0 + n_1)$. This is assuming each point is farthest possible away from the centers.

Case 2: The K-Means objective value if both centers are selected from the same cluster is lower bounded by $\alpha^2 n_1$. This is assuming both centers are in the larger cluster $X_{0,0}$ and the K-Means objective value w.r.t. the points in $\mathcal{X}_{0,0}$ is zero.

Then, we have

$$(\mu\alpha)^2(n_0+n_1) \leq \mu\alpha^2 \cdot \mu n_0 + \mu^2\alpha^2 n_1 \leq \mu\alpha^2 n_1 + \mu^2\alpha^2 n_1 \leq (\mu+\mu^2)\alpha^2 n_1 \leq 0.75\alpha^2 n_1$$

This implies that the maximum K-Means objective value when the centers are chosen from different ground truth clusters is smaller than the minimum K-Means objective when the two centers are chosen from the same ground truth cluster. This, combined with the fact that with probability 1-o(1) one of the initializations selects centers from different ground truth clusters completes the proof.

□

C.3 Extracting the cores with FlowRank

Next, we prove that if the FlowRank algorithm were given the exact density value for each point (which we have assumed to be same for any Layer $\mathcal{L}_{i,j}$), FlowRank gives a score of 1 to all core points (points sampled from $\mathcal{L}_{i,0}$) and a score of < 1 to all non-core points. This gives us an initial insight into FlowRank's functionality. If all the core points are valued 1 and all non-core points are valued at less than 1, then selecting all the 1-valued points will give us the cores, which can then be clustered as shown in Section C.2.

C.3.1 Conditions for core-score being 1

Recall that in FlowRank, after we have a density estimation of the points, we have an ascending random-walk step such that for each point u, we look at its some $r = \mathcal{O}(1)$ neighbors, and move to a neighbor with higher density value at random. We continue this process until we reach a maxima, and the FlowRank value of u is the ratio of the density of u and the average densities of the reachable density peaks. Then, we have the following argument.

Lemma 1. *If for any core point $\mathbf{x} \in \mathcal{L}_{0,0}$, all of its r neighbors are in $\mathcal{X}_0$, the FlowRank score of $\mathbf{x}$ is 1. Furthermore, in the general case, this is a necessary condition.*

Proof. This is because, as long as all the neighbors of $\mathbf{x}$ are in $\mathcal{L}_{0,0}$, they are either another point in $\mathcal{L}_{0,0}$ with same density as $\mathbf{x}$ or a point in $\mathcal{X}_{0,j}, j > 0$ with lower density. following the formulation of the model. This makes $\mathbf{x}$ a local maxima of the walk itself. That is, any density-ascending random walk cannot take any step starting from $\mathbf{x}$, resulting in FlowRank score of $\mathbf{x}$ being 1.

The necessity of this condition follows from the fact that we do not make any assumptions on the relation of densities of the layers from different clusters. Even if $\mathcal{L}_{0,0}$ is the highest density region in $\mathcal{X}_0$, its density may be lower than the lowest density region in $\mathcal{X}_1$. In such a case, if there is any of the r neighbors of a point in $\mathcal{L}_{0,0}$ is from $\mathcal{X}_1$, that point will get a FlowRank score of less than 1.

□

Then, we want to prove that this condition indeed holds in our model. We first define some notations.

Definition 4. *We define* $\sup \mathrm{d}(S)$ *as the maximum Euclidean distance between two points in a set S. Similarly, we define* $\sup \mathrm{d}(S, X, r)$ *as the maximum distance to the r-th nearest neighbor of points in S in the Set X.*

In this direction, we first show that inter-core edges are impossible.

Proposition 2 (Inter-core edges are impossible)**.** *Let $\mathbf{x}$ be a point in $\mathcal{L}_{0,0}$. Then for any $r = \mathcal{O}(1)$, none of $\mathbf{x}$'s r-closest neighbors are from $\mathcal{L}_{1,0}$.*

Proof. From the first condition in Assumption 1, we know that the maximum intra-core distance is μ times smaller than the minimum inter-core distance. As there are $\omega(r)$ many points in $\mathcal{L}_{0,0}$, none of its r closest neighbor can be from $\mathcal{L}_{1,0}$.

□

Next we want to prove that none of the r neighbors of $\mathbf{x} \in \mathcal{L}_{0,0}$ is in $\mathcal{L}_{1,j}, j > 0$. We show that for the case of $j = 1$, the distance property of our model suffices.

Proposition 3. *Let $\mathbf{x}$ be a point in $\mathcal{L}_{0,0}$. Then for any $r = \mathcal{O}(1)$, none of $\mathbf{x}$'s r-closest neighbors are from $\mathcal{L}_{1,1}$*

Proof. Let $\mathbf{x}$ be any point in $\mathcal{L}_{0,0}$. Let $\mathbf{x}'$ be $\mathbf{x}$'s nearest neighbor (in Euclidean distance) from $\mathcal{L}_{1,1}$.

Let this distance be $d_{0,1}$. This implies $\min_{\mathbf{x}'' \in \mathcal{L}_{0,0}} \|\mathbf{x}' - \mathbf{x}''\| < d_{0,1}$. This implies $\min_{\mathbf{x}'' \in \mathcal{L}_{1,0}} \|\mathbf{x}' - \mathbf{x}''\| < \delta \cdot d_{0,1}$ From the second condition of Assumption 1. Then, by triangle inequality we have

$$\begin{aligned}
&\min_{\mathbf{x}'' \in \mathcal{L}_{1,0}} \|\mathbf{x} - \mathbf{x}''\| \leq (1+\delta) d_{0,1} \\
\implies &(1+\delta) d_{0,1} \geq \frac{1}{\mu} \cdot \sup \mathrm{d}(\mathcal{L}_{0,0}) \quad \text{[From Condition 1 of Assumption 1]} \\
\implies &d_{0,1} \geq \frac{1}{\mu(1+\delta)} \sup \mathrm{d}(\mathcal{L}_{0,0}) \\
\implies &d_{0,1} \geq \sup \mathrm{d}(\mathcal{L}_{0,0})
\end{aligned}$$

This completes the proof.

□

Next, we want to extend the proof to the further outer layers $\mathcal{L}_{1,2}, \ldots, \mathcal{L}_{1,\ell}$. This is more complicated, as the data lies on some manifold, with individual layers having more complex structure. Here we use the fact that the underlying Manifold $\mathcal{M}$ has a large reach, and get a bound on the distortion of Euclidean and Geodesic distance.

Theorem 4 (Lemma 3 of Boissonnat et al. (2019))**.** *Let $S \subset \mathbb{R}^d$ be a closed set with reach τ, as defined in Definition 3. Then for any $a, b \in S$ such that $\|a - b\| < 2\tau$, one has $\frac{d_S(a,b)}{\|a-b\|} \leq \frac{2\tau}{\|a-b\|} \cdot \arcsin \frac{\|a-b\|}{2\tau}$.*

Here $\|a - b\|$ is the Euclidean distance between a, b and $d_S(\cdot, \cdot)$ is the Geodesic distance on S (for our application $\mathcal{M}$). We shall use this to prove that none of the r nearest neighbors of $\mathcal{L}_{0,0}$ are in $\mathcal{X}_1$. Furthermore, note that $\alpha \arcsin \frac{1}{\alpha} < \pi/2$ for $\alpha > 1$.

Lemma 2. *Under the constraints of our model and $r = \mathcal{O}(1)$, if $\mu \leq (\frac{1}{c \cdot (1+\delta)})^{\ell-1}$ then none of the r-nearest Euclidean neighbor of a point in $\mathcal{L}_{0,0}$ lies in $\mathcal{X}_1$. This implies that the FlowRank score of any core point is 1.*

Proof. We prove this with an induction on the minimum distance of $\mathbf{x} \in \mathcal{L}_{0,0}$ to $\mathcal{L}_{1,j}$. For any such $\mathbf{x}$, let $d_{0,j}$ be the minimum distance from $\mathbf{x}$ to $\mathcal{L}_{1,j}$. In proposition 3 we have shown that $d_{0,1} \geq \frac{1}{\mu(1+\delta)} \sup \mathrm{d}(\mathcal{L}_{0,0})$. Then, our base case is as follows.

Let $d_{0,2}$ and $\tilde{d}_{0,2}$ be the smallest distance of $\mathbf{x}$ to $\mathcal{L}_{1,2}$ in the Euclidean and geodesic distance on $\mathcal{M}$, respectively. First, note that $1/c \cdot \tilde{d}_{0,2} \leq d_{0,2} \leq \tilde{d}_{0,2}$ for some $1 < c < \pi/2$. The lower bound follows from the distortion bound of Theorem 4 and the upper bound follows from the fact that geodesic distance between two points is always larger than the Euclidean distance.

Let the end points be $\mathbf{y}$ and $\tilde{\mathbf{y}}$ respectively. Then, there exists a point $\tilde{\mathbf{x}} \in \mathcal{L}_{0,1}$ such that $\mathrm{d}_{\mathcal{M}}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \leq \tilde{d}_{0,2}$. This is because, if we consider any geodesic path starting from $\mathcal{L}_{0,0}$ that goes out of $\mathcal{X}_1$, it has to go through $\mathcal{L}_{0,1}, \ldots, \mathcal{L}_{0,\ell}$. This is because the concentric subspaces are defined by expanding the smaller subspaces in all directions within the Manifold $\mathcal{M}$. Then $\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\| \leq \tilde{d}_{0,2}$. Then, from the second condition of Assumption 1, we have

$$\min_{\mathbf{y}' \in \mathcal{L}_{1,1}} \|\mathbf{y}' - \tilde{\mathbf{y}}\| \leq \delta \cdot \tilde{d}_{0,2}$$

Let $\mathbf{y}''$ be the point for which the minimum is achieved. Now, we use the distances between $\mathbf{x} \in \mathcal{L}_{0,1}$, $\tilde{\mathbf{y}} \in \mathcal{L}_{1,2}$ and $\mathbf{y}'' \in \mathcal{L}_{1,1}$, we have a triangle inequality that upper bounds the minimum distance from $\mathcal{L}_{0,0}$ to $\mathcal{L}_{0,1}$, which

we have defined as $d_{0,1}$ in Proposition 3. Combining, we get

$$\begin{aligned}
& d_{0,1} \leq \|\mathbf{x} - \tilde{\mathbf{y}}\| + \|\tilde{\mathbf{y}} - \mathbf{y}''\| \\
\implies & d_{0,1} \leq \tilde{d}_{0,2} + \delta \cdot \tilde{d}_{0,2} \\
\implies & d_{0,1} \leq (1+\delta)\tilde{d}_{0,2} \\
\implies & \tilde{d}_{0,2} \geq \frac{d_{0,1}}{(1+\delta)} \\
\implies & d_{0,2} \geq \frac{d_{0,1}}{c(1+\delta)} \qquad \text{[From Theorem 4]} \\
\implies & d_{0,2} \geq \frac{1}{c\mu(1+\delta)^2} \sup \mathrm{d}(\mathcal{L}_{0,0})
\end{aligned}$$

We can continue with this inductively. Let $d_{0,j}$ is the minimum distance from $\mathbf{x} \in \mathcal{L}_{0,0}$ to $\mathcal{L}_{1,j}$. Assume $d_{0,j} \geq \frac{1}{\mu c^{j-1}(1+\delta)^j} \sup \mathrm{d}(\mathcal{L}_{0,0})$. Note that this is proven for $d_{0,2}$.

Then, we can lower bound $d_{0,j+1}$ as follows. Consider any $\mathbf{x} \in \mathcal{L}_{0,0}$. Let $\tilde{\mathbf{y}}$ be the closest point of $\mathbf{x}$ in $\mathcal{L}_{1,j+1}$ in geodesic distance, and the distance be $\tilde{d}_{0,j+1}$ Then, there is a point $\tilde{x} \in \mathcal{L}_{0,j}$ such that $\mathrm{d}_{\mathcal{M}}(\tilde{x}, \tilde{\mathbf{y}}) \leq \tilde{d}_{0,j+1}$. Then $\|\tilde{x} - \tilde{\mathbf{y}}\| \leq \tilde{d}_{0,j+1}$. Then, there exists $\mathbf{y}'' \in \mathcal{L}_{1,j}$ such that $\|\mathbf{y}'' - \tilde{\mathbf{y}}\| \leq \delta \tilde{d}_{0,j+1}$. Note that this is an upper bound on $d_{0,j}$ via triangle inequality.

Then, we have

$$\begin{aligned}
& d_{0,j} \leq (1+\delta)\tilde{d}_{0,j+1} \\
\implies & \tilde{d}_{0,j+1} \geq \frac{d_{0,j}}{(1+\delta)} \\
\implies & d_{0,j+1} \geq \frac{d_{0,j}}{c(1+\delta)} \\
\implies & d_{0,j+1} \geq \frac{1}{\mu \cdot (c^j(1+\delta)^{j+1})} \sup \mathrm{d}(\mathcal{X}_{0,0})
\end{aligned}$$

This completes the induction. Then, as long as $d_{0,\ell-1} \geq \sup \mathrm{d}(\mathcal{X}_{0,0}, \mathcal{X}_{0,0}, r)$, there are no edges from $\mathcal{L}_{0,0}$ to $\mathcal{X}_1$. A trivial upper bound on $\sup \mathrm{d}(\mathcal{L}_{0,0}, \mathcal{X}, r)$ is $\sup \mathrm{d}(\mathcal{L}_{0,0})$ as $r = \mathcal{O}(1)$. Using this we get that if $\frac{1}{\mu c^{\ell-2}(1+\delta)^{\ell-1}} > 1$, then there are no edges going from $\mathcal{L}_{0,0}$ to $\mathcal{X}_1$. This completes the proof.

□

C.3.2 Non-core points have lower score.

Next we show there exists $r = \mathcal{O}(1)$ such that for each non-core point in $\mathcal{L}_{i,j}$, one of its r nearest neighbor lies in $\mathcal{L}_{i,j-1}$. We first place some notations and results on volumes of balls on Manifolds.

Theorem 5 (Niyogi et al. (2008))**.** *Let $p \in \mathcal{M}$ which is a compact smooth m-dimensional manifold with reach τ. Consider $A = \mathcal{M} \cap B_\epsilon(p)$ where $\epsilon \ll \tau$. Then $vol(A) \geq \cos(\theta)^m \cdot vol(B^m_\epsilon(p))$ where $B^m_\epsilon(p)$ is the m-dimensional ball in T_p (the tangent space) centered at p, $\theta = \arcsin(\epsilon/2\tau)$*

Furthermore, we know that $V^m(\epsilon) = B^m_\epsilon(p)$ where we define $V^m(\epsilon)$ as the volume of the m-dimensional ball of radius ϵ. As we need a more intricate relationship between the width of the layer and the points in local neighborhood, we make some assumptions about the non-linear structure of the each of the layers.

Assumption 2.

1. *We assume that all layers have the same width Δ and the reach of each sub-manifold $\mathcal{X}_{i,j}$ is $\bar{\tau}$ such that $\Delta \ll \bar{\tau} \ll \tau$. Furthermore, assume $\ell \leq m$. Furthermore, note that we do not any specific reach assumptions for the outermost layers.*

2. *Let there be $n_{0,0}$ points sampled in $\mathcal{L}_{0,0}$ such that $n_{0,0} = c_2 \frac{Vol(\mathcal{X}_{0,0})}{V^m(\epsilon)}$ for some $\epsilon \leq c_3\Delta$ where $c_2 > 1$ and $c_3 < 0.001$ are constants. This, combined with Theorem 5 essentially implies that any ϵ-radius ball in $\mathcal{L}_{0,0}$ has some c_2 points on expectation.*

Based on these assumptions, we have the following notion.

Proposition 4. *For any Layer $\mathcal{L}_{0,j}$. Then, the number of points $n_{0,j}$ sampled by the model satisfies $n_{0,j} \geq \frac{Vol(\mathcal{L}_{0,j})}{V^m(2\epsilon)}$*

Proof. In the definition of our model we have $\frac{n_{0,0}}{Vol(\mathcal{L}_{0,0})} = C \cdot \frac{n_{0,1}}{Vol(\mathcal{L}_{0,1})}$. Then, replacing with first condition of Assumption 2 we get

$$C \cdot \frac{n_{0,1}}{Vol(\mathcal{L}_{0,1})} \geq \frac{c_2}{V^m(\epsilon)} \implies n_{0,1} \geq \frac{c_2 Vol(\mathcal{L}_{0,1})}{C \cdot V^m(\epsilon)}$$

Continuing this for some j-steps we get $n_{0,j} \geq \frac{c_2 Vol(\mathcal{L}_{0,j})}{C^j \cdot V^m(\epsilon)}$.

However, as $C < 2$ we get $C^j V^m(\epsilon) \leq V^m(2\epsilon)$. This implies $n_{0,j} \geq \frac{c_2 Vol(\mathcal{L}_{0,j})}{V^m(2\epsilon)}$.

□

Then, we argue that for any point $\mathbf{x} \in \hat{\mathcal{L}}_{0,j}$, there exists a nearby point in $\hat{\mathcal{L}}_{0,j-1}$.

Proposition 5. *Under the constraint of Assumption 2, for any point $\mathbf{x} \in \hat{\mathcal{L}}_{0,j}$, on expectation there exists $\mathbf{x}' \in \hat{\mathcal{L}}_{0,j-1}$ such that $\|\mathbf{x} - \mathbf{x}'\| \leq 1.1\Delta$.*

Proof. This follows from our Proposition 4. Consider the closest point of $\mathbf{x}$ in $\mathcal{L}_{0,j-1}$. Let this point be $\mathbf{y}'$. Consider a point $\mathbf{y}'' \in \mathcal{L}_{0,j-1}$ such that $\|\mathbf{y}' - \mathbf{y}''\| = 6\epsilon$ and the minimum distance between $\mathbf{y}''$ and $\mathcal{L}_{0,j'}$ for $j' \neq j-1$ is at least 4ϵ. Such a point can always be found following the definitions of the layers (which have a width of $\Delta \geq 100\epsilon$. Consider the ball $B_{3\epsilon}(\mathbf{y}'')$. Find $\mathbf{y}''$ such that $B_{3\epsilon}(\mathbf{y}'') \cap (\mathcal{M} \setminus \mathcal{L}_{0,j-1}) = \emptyset$. Furthermore, from Theorem 5 we have $Vol(B_{3\epsilon}(\mathbf{y}'') \cap \mathcal{M}) \geq 0.99 V^m(3\epsilon)$. Then, the expected number of points in $B_{3\epsilon}(\mathbf{y}'') \cap \hat{\mathcal{L}}_{0,j-1}$ can be written as

$$\begin{aligned}
&\mathbb{E}\left[|\hat{\mathcal{L}}_{0,j-1} \cap Vol(B_{3\epsilon}(\mathbf{y}'')|\right] \geq n_{0,j-1} \cdot \frac{0.99 V^m(3\epsilon)}{Vol(\mathcal{L}_{0,j-1})} \\
&\geq \frac{c_2 Vol(\mathcal{L}_{0,j-1})}{V^m(2\epsilon)} \cdot \frac{V^m(3\epsilon)}{Vol(\mathcal{L}_{0,j-1})} \qquad \text{From Proposition 4} \\
&\geq c_2 \frac{V^m(3\epsilon)}{V^m(2\epsilon)} \geq 1
\end{aligned}$$

Then, the distance of this point to $\mathbf{x}$ is upper bounded by $\Delta + 9\epsilon \leq 1.1\Delta$.

Here the lower bound of $0.99 V^m(3\epsilon)$ follows from the fact that $\cos\Theta$ in Theorem 5 gets very close to 1 when radius is much smaller than reach, which is the case here as $\epsilon \leq 0.01\Delta \ll \tau_{i,j}$.

□

That is, we have proven that in our model, if you consider a radius of 1.1Δ, you can always find a neighbor in the inner layer. Then, finally, we are left to calculate what is the maximum possible points in this radius in our configuration.

Lemma 3. *Given our model and the conditions in Assumption 2, there exists $r = \mathcal{O}(1)$ such that if we consider the r-nearest neighbors of any point in $\hat{\mathcal{L}}_{0,j}$, at-least one neighbor lies in $\hat{\mathcal{L}}_{0,j-1}$. If our FlowRank algorithm is initialized with such an r, then all non-core points get a score of less than 1.*

Proof. In Proposition 5 we have obtained an upper bound on the radius that ensures this behavior. We finally calculate the maximum expected number of points in this radius. Consider the densest region of $\mathcal{X}_0$, that is $\mathcal{X}_{0,0}$. Then, the number of points in a radius of 1.1Δ can be upper bounded as

$$\mathbb{E}\left[|\hat{X} \cap B_{1.1\Delta}(\mathbf{x})| \leq n_{0,0} \cdot \frac{Vol(B_{1.1\Delta}(\mathbf{x}))}{Vol(\mathcal{X}_{0,0})} \leq c_3 \frac{V^m(1.1\Delta)}{V^m(2\epsilon)}\right]$$

Note that this value is exponential only in m, but as m is a constant, we get a constant upper bound on r.

□

Finally, we recall the Theorem from the main paper and combine the two results we obtained formally.

Theorem (Restated Theorem 1)**.** *Let data be generated from the* $\mathsf{LCPDM}(2, \ell)$ *model. Assume we get the exact density of the points and expected value of randomly ascending random walk in the FlowRank algorithm. Let this be called the ideal FlowRank outcome. Then all the core points* ($\hat{\mathcal{L}}_0$) *get a score of* 1*. Additionally, all non-core points get a score* < 1*.*

Proof. First we recall that $\mathcal{L}_{i,0}$ and $\mathcal{X}_{i,0}$ are synonymous. In Lemma 2 we have shown that for any point in $\mathcal{L}_{i,0}$ if the cores are sufficiently separated then all points in it (the core points) get a score of 1.

Next in Lemma 3 we show that under the additional density conditions of Assumption 2, there exists an r (that depends exponentially on the dimension of the underlying manifold $\mathcal{M}$, which is a constant) such that for each non-core point in $\mathcal{L}_{i,j}$, one of its r-nearest neighbor lies in $\mathcal{L}_{i,j-1}$ in expectation, and therefore the FlowRank score is less than 1. This completes our proof.

□

C.4 Correctness of CoreSPECT framework given the correct layers.

Theorem (Clustering in the LCPDM model (Restated Theorem 3)**.** *Let* X *be* n *datapoints generated from the* $\mathsf{LCPDM}(2, \ell)$ *model. Let us have an estimate of the density layers, given as* $S_0, \ldots, S_{\ell-1}$ *such that* $|S_j \cap \hat{\mathcal{L}}_j| = (1-f)|\hat{\mathcal{L}}_j|, j > 0$*. for a sufficiently small function* $f = o(1)$ *that is layer preserving. Then, applying a variant of K-Means to* S_0 *and expanding the clustering using a CDNN graph (with correctly chosen parameters) results in clustering with o(1) misclassification error rate on expectation.*

We prove this with a minimal adjustment. We have shown in Lemma 2 that all core points get a score of 1 and all non-core points get less than 1 by FlowRank. Therefore we assume $S_0 = \hat{\mathcal{L}}_0$. Furthermore, we simply set $t = 1$ for the proof. The proof is then as follows.

Proof. First from Proposition 1 we know that the $\log n$-attempt single-round K-Means separates the core with probability $1 - o(1)$. We continue conditioned on this scenario.

Now, consider S_1. We know $|S_1 \cap \hat{\mathcal{L}}_1| = (1-f)|\hat{\mathcal{L}}_1|$. Furthermore, as S is layer-preserving, we assume $|S_1 \cap \hat{\mathcal{L}}_3| = 0$. Otherwise, S_1 contains all points in $\hat{\mathcal{L}}_1$ and $\hat{\mathcal{L}}_2$ and we can treat them as a single-layer in our initial model definition (albeit with a more complex analysis).

Then, for all points in $\hat{\mathcal{L}}_1$, we know the following. Let $\mathbf{x} \in \hat{\mathcal{L}}_1 : \mathbf{x} \in \mathcal{X}_0$. Then $\min_{\mathbf{x}' \in \mathcal{L}_{0,0}} \|\mathbf{x} - \mathbf{x}'\| \leq \delta \cdot \min_{\mathbf{x}' \in \mathcal{L}_{1,0}} \|\mathbf{x} - \mathbf{x}'\|$. This implies if we apply our layer-expansion with one neighbor, all such points will be correctly classified.

Assume that in the worst case scenario we misclassify each $\mathbf{x} \in S_1 \cap \mathcal{M} \setminus \hat{\mathcal{L}}_1$, which is $\leq f(n)$ many points, with n being the total number of points. Lets call this set E_1.

Then, when considering the points in S_2, let's consider what is the criteria for this point to be correctly clustered. Consider $\mathbf{x} \in S_2 \cap \hat{\mathcal{L}}_2$. We know that on expectation there exists a point $\mathbf{x}' \in \hat{\mathcal{L}}_1$ such that $\|\mathbf{x} - \mathbf{x}'\| \leq 1.1\Delta$ (from Proposition 5). If $\mathbf{x}'$ is either correctly classified or is present in S_1, then $\mathbf{x}$ is correctly clustered.

Therefore, the set of points incorrectly clustered from S_2 is upper bounded by the set of points that is within 1.1Δ distance of a misclassified point or a point in $\hat{\mathcal{L}}_1$ that is not in S_1. Additionally, any point in S_2 that is not from $\hat{\mathcal{L}}_2$ may get incorrectly clustered.

Here, we note that the maximum number of points in the 1.1Δ-radius of any point is upper bounded by $\frac{n_{0,0} V^m(1.1\Delta)}{Vol(\mathcal{L}_{0,0})}$ assuming they are all from maximum density points. This can be simplified as

$$\frac{n_{0,0} V^m(1.1\Delta)}{Vol(\mathcal{L}_{0,0})} \leq \frac{c_2 V^m(1.1\Delta)}{V^m(1.1\epsilon)} \qquad \text{[From Condition 2 of Assumption 2]}$$

which we upper bound as the constant C_m (as m is fixed). Then, the total number of misclassified points in S_2 is upper bounded by

$$|E_1| \cdot C_m + |S_2 \setminus \hat{\mathcal{L}}_2| \cdot C_m + |\hat{\mathcal{L}}_1 \setminus S_1| \cdot C_m = \mathcal{O}(f(n))$$

We continue this for ℓ layers. Assume E_j points have been misclassified up to layer j. Then the number of misclassified points in S_{j+1} can be upper bounded as $|E_j| \cdot C_m + |S_{j+1} \setminus \hat{\mathcal{L}}_{j+1}| \cdot C_m + |\hat{\mathcal{L}}_j \setminus S_j| \cdot C_m$. Now, if we induct on $|E_j| = \mathcal{O}(f(n))$ then we get $|E_{j+1}| = \mathcal{O}(f(n))$. This implies that $|E_\ell| = \mathcal{O}(f(n)) = o(n)$.

This implies we finish clustering all the points with $o(n)$ many misclassifications, which leads to an $o(1)$ misclassification error rate, completing our proof.

□

Discussion of runtime. Finally, we note that given the CDNN graph, the run time of our layer-expansion method is $n \cdot k \cdot t$.

Theorem (Restated Theorem 2)**.** *Given the CDNN graph $G^+_{t,S}$ and a clustering of S_0, the rest of the points can be clustered in $\mathcal{O}(n \cdot k \cdot t)$ time, which is linear in the number of the edges and the number of clusters in $G^+_{t,S}$.*

Proof. All points in S_0 are already clustered. Starting with S_1, loop through every point $\mathbf{u}$ in S_1 and update the cluster membership vector $\hat{C}(\mathbf{u})$ and the cluster label k_u by the following: $\hat{C}(\mathbf{u}) \leftarrow \sum_{v \in N_{G^+}} W(\mathbf{u}, \mathbf{v}) \cdot \hat{C}(\mathbf{v})$; $k_u \leftarrow \arg\min \hat{C}(\mathbf{u})$. Repeat the same procedure with S_{j+1} once all points in S_j are clustered. Here, for each point in S_j, we look at the k-length cluster membership vector $\hat{C}(\mathbf{u})$ for some t many neighbors, which takes $\mathcal{O}(t \cdot k)$ time. This makes the total runtime $\mathcal{O}(n \cdot k \cdot t)$.

□

C.5 Runtime Analysis of the Complete Algorithm

First, we go through q-NN generation and FlowRank. Here we generate q-NN graph on n many d-dimensional points using HNSW. Let's call this runtime HNSW(n,d,q). Then, FlowRank is obtained by running $log\ n$ step random walks and then ascending random walks (that are truncated after a $log\ n$ step). This takes $O(nlog^2n)$ including $O(log\ n)$ iteration of ascending random walk from each node to estimate random walk behavior. Then we run any clustering algorithm on the top c fraction of nodes (the core nodes), which takes CLUST($c \cdot n$). Once the cores are extracted and clustered, we apply propagation. Each propagation step requires HNSW(n, d, t) + ntk run-time, where k is the number of clusters output when clustering the core. Then, the total run-time can be written as $O(nlog^2n)$ + CLUST$(c \cdot n)$ + $n \cdot t \cdot k \cdot l + l \cdot$ HNSW$(n, d, max(q, t))$ where l is the number of layers. It is well known that HNSW$(n, d, max(q, t))$ operates approximately at $O(max(q, t) \cdot nlogn)$, so the total run-time is $O(nlog^2n$ + CLUST$(c \cdot n)$ + $n \cdot t \cdot k \cdot l + l \cdot (q + t) \cdot nlogn)$. This is quite fast in practice, with our framework needing around a minute to cluster CIFAR-20 (50,000 points, 768 dimension, and 20 clusters).

D Experiments

D.1 Description of datasets and experiments

As we have described, we use a total of 15 datasets in this paper. We describe the details of the datasets here. We use 11 single-cell datasets, 2 bulk-RNA datasets, and two image datasets.

Single-cell datasets: We use all of the 11 datasets from Mukherjee and Zhang (2025). The details of the datasets are as follows.

Name	# of points	# of communities	Source
Baron_Human	8569	14	(Abdelaal et al., 2019)
Baron_Mouse	1886	13	(Abdelaal et al., 2019)
Muraro	2122	9	(Abdelaal et al., 2019)
Segerstolpe	2133	13	(Abdelaal et al., 2019)
Xin	1449	4	(Abdelaal et al., 2019)
Zhengmix8eq	3994	8	(Duò et al., 2018)
T-cell dataset	5759	10	(Savas et al., 2018)
ALM	10068	136	(Smith et al., 2019)
AMB	12382	110	(Abdelaal et al., 2019)
TM	54865	55	(Abdelaal et al., 2019)
VISP	15413	135	(Smith et al., 2019)

Table 3: Details of the single-cell RNA-seq datasets

Bulk-RNA datasets: For the bulk-RNA datasets, we take patient samples from the The Cancer Genome Atlas (Tomczak et al., 2015). This was part of a comprehensive project launched by the ational Human Genome Research Institute (NHGRI) in 2006 to catalog genetic mutations responsible for cancer using genome sequencing and bioinformatics. The project analyzed samples of around 11,000 tumor samples across 33 cancer types (which we use as the ground-truth cluster identity). We use two modalities of datasets, mRNA and micro-RNA, which form the two bulk-RNA datasets that we use.

For the single-cell and bulk-RNA datasets, we first log-normalize it and then apply PCA dimensionality reduction of dimension $\min\{50, \#\text{ of ground truth clusters}\}$, which is a standard pipeline in the single-cell (and genomics) analysis literature (Duò et al., 2018).

Image datasets: For image datasets, we use the test-set of the popular MNIST (Deng, 2012) and Fashion-MNIST (Xiao et al., 2017) dataset, as well as the popular CIFAR datasets. Following Shinde et al. (2024), we use the ViT embeddings for CIFAR 10, 20, and 100. Additionally, for CIFAR 10 we use both ViT-Large and ViT-Base embeddings.

D.2 Comparison algorithms

Density-based clustering As we have discussed, we focus on density based and manifold based clustering. For density based clustering, we implemented HDBSCAN (Campello et al., 2013). We also implemented DBSCAN (Ester et al., 1996) and OPTICS (Ankerst et al., 1999) but were unable to use them as benchmarks, as they marked most points as outliers for most datasets. We also uesd three density peak clustering algorithms (Rodriguez and Laio, 2014; Wang and Xu, 2017; Yu et al., 2023).

Manifold-based clustering For Manifold clustering, we first looked at the scikit-learn spectral clustering (Von Luxburg, 2007) implementation. We found the default setting of rbf kernel took prohibitively long time and also did not produce good quality cluster, and therefore we did not include it in our benchmarks. In comparison, the K-nearest-neighbor-based kernel had much better performance, proving to be a competent benchmark. We also implemented two recent manifold clustering algorithms (Little et al., 2020) (Trillos et al., 2023) with theoretical guarantees but the performances were suboptimal to spectral clustering by large margins, and thus were not reported in detail in this paper. It remains as future steps to analyze the underlying cause of their ineffectiveness on real world data.

Further comparisons Finally, we note that design of clustering algorithm is a very extensive field with new clustering algorithms being designed. Especially, both the areas of density-peak clustering and manifold clustering are very active. We aim to test our framework both on and against more such algorithms in future.

D.3 Performance of CoreSPECT on small noiseless datasets

In the main paper, we noted that in the four smallest datasets (Baron Mouse, Xin, Segerstolpe, and Muraro) our performance lacked behind that of Density based clustering (especially PECANN-DPC). Here we explore this in

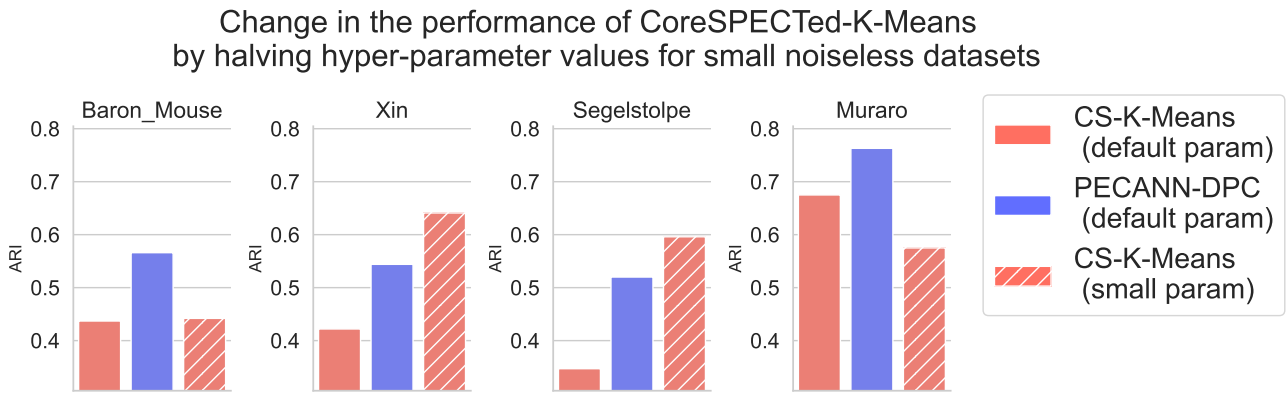

Figure 13: Enhancement in the performance of CoreSPECTed-K-Means by selecting smaller hyper-parameters for small noiseless datasets

more detail. First, we note the ARI of CoreSPECTed-K-Means and PECANN-DPC for the four datasets.

In fact, PECANN-DPC has noticeably higher ARI than our method for all four datasets. First we note that all of these clusters are well-separated. This can be observed as follows. If we look at any pair of clusters, all the nearest neighbors of any point are from the same cluster. This implies, the main hurdle of "hard-to-separate-peripheral points" is not present in these datasets. However, we still want to better understand if the performance of our framework can be improved in such cases. In this direction, we observe two interrelated reasons.

1. For all of these datasets, the cores (top 10% of the points) selected by FlowRank (with default parameter) sometimes does not include any points from the smallest (such as less than 20 points) clusters.
2. In our default setting, we choose $q = 40, r = 20, t = 20$. The parameter r decides how many neighbors of a point is looked at for the ascending random walk step. If r is larger than the size of a ground truth cluster, then there is a high chance *all the points* from that cluster will have a low FlowRank value. Similarly, in our layer-expansion step, we created a t-regular CDNN graph. Even if the core has points from a small ground truth cluster that is well separated in the core-clustering step, a large t can incorrectly assign non-core points from these small clusters.

Improving on the first point falls within the overarching direction of coming up with a better core-extraction algorithm, and we believe this is a very important question. With regards to the second point, we observe that our hyper-parameters can be changed slightly to get better results on these small datasets. As an example, we run CoreSPECTed-K-means with the hyper-parameters ($q = 20, r = 10, t = 10$), that is dividing every value with 2. The comparative performance of CS-K-Means w.r.t. the changed parameters are shown in Figure 13.

As we can observe, for 2 out of 4 datasets, we have considerable improvements (more than 50%), only 2% improvement for Baron-Mouse, and the ARI value for Muraro goes down by approximately 20%. This demonstrates the potential of choosing parameters in a more principled way.

However, this comparison may seem unfair as we are only using the default parameters for PECANN. In this direction, to understand the potential of the different clustering algorithms as well as our CoreSPECT framework, we report the best clustering performance for each dataset with the best hyper-parameters chosen from a grid search. Note that we do this in a supervised manner for all the clustering algorithms. The goal of this Section is to simply present the highest accuracy that can be achieved with the method, which is more of an indicator of how well the underlying mechanisms of the algorithms relate to/fit the underlying structures of the dataset.

E Limitations and Future Directions

We conclude with a discussion on our limitations.

1. Improving the core-extraction step is an important problem, as previously noted. We use the order provided

by FlowRank and partition it into ten parts. Ideally, different data will have different numbers of layers for different clusters, and this phenomena needs to be explored further. We believe FlowRank is only a placeholder, and an improved algorithm can be achieved.

2. Next, we note that we have achieved the gains mentioned so far by fixing the other hyperparameters of our framework. However, we believe these hyperparameters can be more informatively selected that can further boost our performance.

3. We have applied our framework to two algorithms. Applying and understanding the effect of our framework on different algorithms and different kind of embeddings is necessary to better understand the scope of application.

4. Data-based selection of the optimal hyper-parameter is an important direction. As we have observed, some of the optimal parameters depend on the internal geometry of the data. For example, the correct radius for choosing neighbors for ascending random walk depends on the width of the layers. A better inference of this underlying geometry may further raise confidence in our framework.