Learning the Essentials: Clusterable Representations via Locally Gaussian Deep Embedding

Anonymous authors
Paper under double-blind review

Abstract

Clustering — the grouping of similar objects in an unsupervised fashion — is a foundational task in machine learning. Deep learning methods have shown promise in this regard, as they can learn information-rich representations from complex data. However, all current deep learning-based clustering methods suffer from weaknesses in at least one key aspect: they require input parameters inaccessible to the user, their internal representations lack interpretability, or the resulting clustering cannot be guaranteed to truly correlate with the input data. To overcome these limitations, we introduce a novel method called Locally Gaussian Deep Embedding (LGDE). LGDE is an autoencoder featuring an embedding space that can be meaningfully clustered without requiring the number of clusters as an input. We derive a new loss function, mSIGReg — a variant of the SIGReg (Sketched Isotropic Gaussian Regularization) method — and demonstrate that its optimization yields well separated and easily identifiable clusters. Additionally, we show how compactness constraints can be incorporated, resulting in clustering that is highly descriptive of the input data. We demonstrate that LGDE retains its effectiveness in generating informative and clusterable representations across multiple data types ranging from established image datasets such as MNIST and EMNIST to structural biology data on signaling proteins relevant to drug design.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 Introduction

Many data distributions of interest contain inherent structure in the form of clusters. Identifying these clusters is often of great value to understand the underlying structure of the data.

Neural networks have demonstrated a remarkable aptitude for learning information-rich representations from complex data. These frameworks have therefore been widely applied in an effort to develop increasingly efficient clustering methods. Still, clustering using neural networks remains an unsolved problem, as all existing methods suffer to some extent from at least one of the following issues: hyperparameters have to be chosen manually (often with inadequate criteria), uninterpretable internal representations, or resulting clustering that cannot be guaranteed to correlate with the input data. The former has plagued almost all previously published methods, as they expect a parameter k, the number of desired clusters, which is often unknown in real applications. This problem has been somewhat alleviated recently, as a few individual methods have overcome this limitation. However, these (nonparametric) methods, although very efficient, do not aim for an interpretable latent representation of the data, which makes it difficult to understand how they determine these clusters. Most importantly, none of the existing nonparametric (and very few parametric) methods explicitly require clustering be directly inferred from the input data distributions.

To overcome these limitations, we introduce Locally Gaussian Deep Embedding (LGDE). LGDE can be used as a nonparametric clustering technique (k is not a required input). It forms a general feature learning framework, where the internal representation of the model is expected to contain an interpretable structure. LGDE is structured as an autoencoder, whose latent space is constrained to contain *local Gaussianity*. By "local", we mean that the latent distribution can be approximated using k non-overlapping Gaussian components, where $k > 1$. LGDE achieves this by optimizing a novel loss function — mSIGReg — which is a significantly modified version of the previously published SIGReg regularization method. The advantage of this objective is the significant increase in the clusterability of the latent space, as shown in ablation studies.

Our key contributions are: 1) we show how each input data point can be represented as a perturbed version of an "ideal state" of the system; 2) we demonstrate how local Gaussianity can be enforced in the latent representation using mSIGReg, leading to a significant enhancement of the clusterability of the latent space; 3) we show how information from the input data $\mathbf{X}$ can be explicitly incorporated into the clustering; and 4) we show that LGDE can learn highly informative representations over a wide range of data domains, even in very low-dimensional latent spaces, revealing insights that would likely be missed by current techniques.

2 Related Work

We partition the relevant existing techniques into two classes: parametric and nonparametric methods. Here, "parametric" means the necessity to provide k, the (maximum) number of clusters present in the latent space.

The recent surge in deep clustering research was in large part catalyzed by DEC (Xie et al., 2016), followed by autoencoder-based methods such as GMVAE (Dilokthanakul et al., 2016), DCN (Yang et al., 2017), VaDE (Jiang et al., 2016), and IDEC (Guo et al., 2017). These methods learn an encoding $\mathbf{z}_i = f_\theta(\mathbf{x}_i)$ and impose the clustering in the resulting latent space. This clustering can be based on distances in the latent space (as in DEC, IDEC, and DCN), or on responsibilities derived from a mixture model defined in the latent space (as in the GMVAE and VaDE methods). In addition, subsequent work has extended related ideas to graph-structured data (Wang et al., 2019; Bo et al., 2020).

All these methods are parametric, meaning they require an input parameter k that determines the number of clusters identified by the model. Although some of these methods can produce fewer than the specified number of clusters, most training runs result in exactly k clusters. Furthermore, k imposes a strict upper limit on the number of clusters,

meaning that no additional clusters can be identified, even if doing so would lead to a more meaningful latent space.

Furthermore — and, in our view, more importantly — these methods do not make significant use of information from the original $\mathbf{X}$ dataset during the clustering process. While most of these methods incorporate this information into the model via a reconstruction constraint, we feel that this constraint is typically not sufficiently strict. Consider, for instance, the traditional variational autoencoder (VAE) (Kingma and Welling, 2013): it can satisfy the reconstruction constraint almost perfectly — on datasets such as MNIST — even though its latent space is structurally unimodal. In other words, the autoencoder's decoder is often capable of faithful reconstruction even if the encoder has not actually captured the essential cluster structure of the input data.

Only a few deep clustering methods do not require the parameter k to be specified in advance. Some of these (Ronen et al., 2022; Chen, 2015) are based on Dirichlet Process Mixtures (DPMs) (Ferguson, 1973; Antoniak, 1974), which assume that the number of mixture components can be unbounded. In other words, new clusters can be created as needed, and their expected number is controlled by a concentration parameter. Other ways to avoid specifying the parameter k include, for example, hierarchical clustering (as in the JULE method (Yang et al., 2016)) and pairwise binary classification (as in the DAC method (Chang et al., 2017)). These methods, too, require the manual setting of other parameters in place of k. In a sense, this is merely a matter of replacing one choice with another. However, we consider this an advancement, as the latter approaches do not suffer from a strict upper limit on the number of identifiable clusters. Like parametric methods, these approaches also utilize the input data $\mathbf{X}$ for clustering only implicitly; the clustering loss objectives operate exclusively on the latent representation $\mathbf{Z}$.

This work aims to advance several aspects related to these alternative methods. First, we wish to avoid having to specify the parameter k while constructing the clusterable latent space $\mathbf{Z}$. Furthermore, we want the resulting clustering to correspond to the original data $\mathbf{X}$; the clusters should thus be *compact* in terms of $\mathbf{X}$. Third, the goal is for the resulting latent representations to be *interpretable*: a change applied to a point $\mathbf{x}_i$ should generally lead to a predictable change in the point $\mathbf{z}_i$. Finally, we aim to achieve all this by making only minor modifications to the familiar autoencoder architecture.

3 Method

3.1 Data as observations of perturbed states

Let $\mathbf{X} \in \mathbb{R}^{N \times D}$ be a clusterable dataset of N observations in D dimensions. In this work, we regard the generation process of such a dataset as follows: the observations $\mathbf{x}_i \in \mathbb{R}^D$ are generated via a nonlinear transform of a latent (hidden) variable $\mathbf{z}_i \in \mathbb{R}^d$, where $d < D$. Since $\mathbf{X}$ is postulated to be clusterable, we expect $\mathbf{Z} \in \mathbb{R}^{N \times d}$ to manifest the corresponding clustering. Moreover, we postulate $\mathbf{z}_i$ to have the following structure:

$$\mathbf{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\epsilon}_i, \tag{1}$$

where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{\mu_i}, \sigma \boldsymbol{I})$ for some σ. That is, the latent variable that generates the ith observation is itself the sum of a state $\boldsymbol{\mu}_i$ and a perturbation vector $\boldsymbol{\epsilon}_i$ added to that state (with $\boldsymbol{\epsilon}_i$ presumed to be normally distributed). The observed data are then

$$\mathbf{x}_i = g(\boldsymbol{\mu}_i, \boldsymbol{\epsilon}_i), \tag{2}$$

where $g(\cdot)$ is some nonlinear transformation. This formalism is purely deterministic; each input $\mathbf{x}_i$ corresponds to some $\boldsymbol{\mu}_i$ and $\boldsymbol{\epsilon}_i$. However, since $\boldsymbol{\mu}_i$ represents the *state* of the data point i, and the data are presumed to be clusterable, we have

$$|\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N\}| < N. \tag{3}$$

That is, the number of distinct states in the data is smaller than the number of data points. In practice, since $\boldsymbol{\mu}_i$ is to be generated by a neural network with continuous output, Eq. (3) implies that $\boldsymbol{\mu} \in \mathbb{R}^{N \times d}$ contains points distributed across multiple distinct high-density regions of $\mathbb{R}^d$.

This approach has two highly desirable properties. First, it allows us to implement the mSIGReg loss objective in a natural manner (see below), enabling the emergence of local Gaussianity in the latent space. Second, the partitioned embedding is highly decipherable; we are learning the ideal state of each observation, along with the perturbation added to that state. This endows the latent space with directional meaning.

To incorporate this partitioned nature of the latent embedding into our autoencoder architecture, we use a branched encoder network (Figure 1). Specifically, the encoder contains a shared trunk that learns an intermediate embedding, which is then passed to two distinct sub-networks. One of these networks is the $\boldsymbol{\mu}$ prediction head ($f_{\boldsymbol{\mu}}$), while the other ($f_{\boldsymbol{\epsilon}}$) produces the perturbation vectors $\boldsymbol{\epsilon}$. The $\boldsymbol{\mu}$ predictor is straightforward; it is simply a feedforward network with the number of output neurons matching the desired latent dimensionality d.

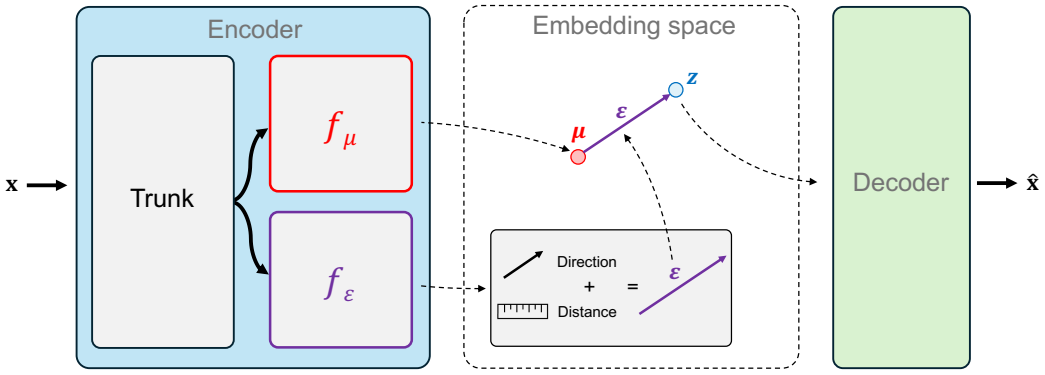


Figure 1: The proposed architecture of LGDE.

The perturbation predictor $f_{\boldsymbol{\epsilon}}$ needs to be designed more carefully. The output of this network must be constrained; each latent $\mathbf{z}_i$ should be in the vicinity of the corresponding state embedding $\boldsymbol{\mu}_i$. This is because each observation is by definition a realization of its own state. To that end, $f_{\boldsymbol{\epsilon}}$ has $d+1$ output neurons. The first d of these encode a direction vector $\mathbf{w}_i \in \mathbb{R}^d$ in the embedding space. The final output of this network encodes the perturbation distance r_i from the state embedding $\boldsymbol{\mu}_i$. To constrain this distance to the vicinity of $\boldsymbol{\mu}_i$, this output is passed through the logistic function

$$r_i = \sigma(c_i) \cdot R_{\max}, \tag{4}$$

where c_i is the raw value of the final output and $R_{\max}$ is the maximum permissible perturbation distance. The full latent embedding then becomes

$$\mathbf{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\epsilon}_i \tag{5}$$

$$= \boldsymbol{\mu}_i + \frac{\mathbf{w}_i}{|\mathbf{w}_i|} \cdot r_i, \tag{6}$$

where the direction vector $\mathbf{w}$ is normalized to ensure that the distance constraint holds true. See Section E of the Appendix for a discussion on how $R_{\max}$ is chosen.

In this manner, for every input $\mathbf{x}_i$, we produce an embedding of its state $\boldsymbol{\mu}_i$, along with the perturbation $\boldsymbol{\epsilon}_i$ that needs to be added to that state embedding, to produce the latent embedding $\mathbf{z}_i$ corresponding to the input. The resulting $\mathbf{z}_i$ is passed to the decoder network, which is trained to reconstruct the input. All parts of the network are trained simultaneously.

3.2 Enforcing local Gaussianity (masked SIGReg)

To achieve a clusterable latent representation, we encourage the encoder's outputs to exhibit *local Gaussianity*. In other words, the latent space should contain regions where embeddings follow a normal distribution, with these regions separated by areas of low density. Our method, in which each observation is encoded as the sum of a state embedding and a noise

embedding, offers a natural way to define the locations of these Gaussian distributions; ideally, each state embedding would correspond exactly to the mean of a specific Gaussian distribution. Put differently, the distribution of the embeddings $\mathbf{Z}$ should be $\mathcal{N}(\boldsymbol{\mu_i}, \sigma \boldsymbol{I})$ in the vicinity of each embedded state $\boldsymbol{\mu_i}$.

To design a loss function that promotes such local Gaussianity, we draw upon previous research. We employ a method called SIGReg (Sketched Isotropic Gaussian Regularization) (Balestriero and LeCun, 2025; Bayat, 2026) and apply it locally within the latent space, rather than globally. SIGReg is typically used to regularize latent embeddings so that they conform to an isotropic normal distribution. This is essentially the opposite of what we aim to achieve; by definition, a single isotropic normal distribution comprises only one cluster. However, if multiple distinct regions of the space follow a normal distribution, the space is inevitably partitionable into clusters (provided the regions are sufficiently far apart).

Consider a set of embeddings $\mathbf{Z} \in \mathbb{R}^{(N \times d)}$, along with the following transformation

$$\mathbf{p} = \mathbf{v}\mathbf{Z}^{\mathbf{T}}, \qquad \mathbf{v} \sim \text{Unif}(\mathbb{S}^{d-1}), \tag{7}$$

where we have projected the embeddings onto a random direction drawn from a unit sphere. SIGReg works by computing the empirical characteristic function (CF) of this projection vector $\mathbf{p} \in \mathbb{R}^N$ and comparing it against a target characteristic function, chosen to be that of the standard normal. If the CFs are equal for all $\mathbf{v}$, the full d-dimensional embedding space is guaranteed to be Gaussian (Cramér and Wold, 1936).

The CF of a continuous distribution $p_X(x)$ is its Fourier transform

$$\varphi_X(t) = \int_{-\infty}^{\infty} e^{itx} p_X(x) dx. \tag{8}$$

For the Gaussian distribution $\mathcal{N}(\mu, \sigma)$, we have

$$\varphi_{\mathcal{N}(\mu,\sigma)}(t) = e^{it\mu - \frac{1}{2}\sigma^2 t^2}. \tag{9}$$

The empirical CF of $\mathbf{p}$ is computed as

$$\varphi_{\mathbf{p}}(t) = \frac{1}{N} \sum_{n=1}^{N} e^{itp_n}. \tag{10}$$

To turn these expressions into a loss function, SIGReg considers the quantity

$$T_{\mathbf{p}} = N \int_{-\infty}^{\infty} w(t) |\varphi_{\mathbf{p}}(t) - \varphi_{\mathcal{N}(\mu,\sigma)}(t)|^2 dt, \tag{11}$$

where the weighting factor $w(t) = e^{-t^2/(2\lambda_T^2)}$ is introduced to ensure the convergence of the integral (for details, see Section E in the Appendix). This expression has a unique minimum value of 0 when $\varphi_{\mathbf{p}}(t) = \varphi_{\mathcal{N}(0,1)}(t)$ for all values of t; thus, minimizing the expression is a suitable objective when striving for Gaussianity. In practice, the integral is approximated as a weighted sum of K terms (using the trapezoidal rule):

$$T_{\mathbf{p}} \approx N \sum_{k=1}^{K} \alpha_k w(t_k) |\varphi_{\mathbf{p}}(t_k) - \varphi_{\mathcal{N}(\mu,\sigma)}(t_k)|^2, \tag{12}$$

where α_k is the trapezoidal weight and $\{t_1 \dots t_K\}$ are evenly spaced points on some interval $[a, b]$ (see Section E in the Appendix).

In Eq. (12), each element of the vector $\mathbf{p}$ is weighted equally. Precisely for this reason, minimizing the function $T_{\mathbf{p}}$ leads to a single Gaussian distribution in the embedding space. To encourage the formation of multiple regions following a Gaussian distribution, we propose a "softened" removal of the influence of certain elements p_n from this expression; in practice, this measures the extent to which a subset of projections conforms to a Gaussian distribution. To achieve this, we consider the weighted characteristic function of the vector $\mathbf{p}$, defined as follows:

$$\tilde{\varphi}_{\mathbf{p}}(t, \mathbf{m}) = \frac{\sum_{n=1}^{N} m_n e^{itp_n}}{\sum_{n=1}^{N} m_n}, \tag{13}$$

where $\mathbf{m} \in [0,1]^N$. Clearly, elements p_n for which the corresponding m_n are negligible are effectively discarded in the computation. Thus, this quantity can gauge the local Gaussianity of a vector $\mathbf{p}$, given the suitable masking vector $\mathbf{m}$. The quantity of interest then becomes

$$\tilde{T}_{\mathbf{p}}(\mathbf{m}) = \sum_{k=1}^{K} \alpha_k w(t_k) |\tilde{\varphi}_{\mathbf{p}}(t_k, \mathbf{m}) - \varphi_{\mathcal{N}(\mu,\sigma)}(t_k)|^2, \tag{14}$$

where we have omitted the factor N from Eq. (12) since it only scales the sum.

For a given set of latent embeddings $\mathbf{Z}$ and masking vector $\mathbf{m}$, along with a projection direction $\mathbf{v}$, the corresponding loss term becomes

$$l(\mathbf{Z}, \mathbf{m}) = \tilde{T}_{\mathbf{p}}(\mathbf{m}), \quad \mathbf{p} = \mathbf{v}\mathbf{Z}^{\mathbf{T}}. \tag{15}$$

This formulation allows us to promote local Gaussianity in $\mathbf{Z}$ by employing a suitable masking vector. For this purpose, in this work we propose generating appropriate masking vectors for each $\boldsymbol{\mu}_i$ — based on distances in the latent space — for a subset of the training set (consisting of embeddings of B data points), denoted by $\mathbf{Z}_B$ and $\boldsymbol{\mu}_B$. Given a latent embedding $\boldsymbol{\mu}_i$, we calculate pairwise distances $\mathbf{d}_i \in \mathbb{R}^B$, where $\mathbf{d}_{ij} = ||\boldsymbol{\mu}_i - \mathbf{z}_j||_2$. We then compute the mask vector for the state embedding $\boldsymbol{\mu}_i$,

$$\mathbf{m}_i = \sigma\left(\frac{R_{\max} - \mathbf{d}_i}{T}\right), \tag{16}$$

where $\sigma(\cdot)$ is the logistic function, and T is a temperature parameter. This vector contains large values (≈ 1) corresponding to latent variables $\mathbf{z}_j$ located close to the point $\boldsymbol{\mu}_i$ (relative to the maximum displacement distance $R_{\max}$). When the distance approaches $R_{\max}$, the masking weight drops to zero. The steepness of this drop is controlled by the parameter T. The region surrounding the point $\boldsymbol{\mu}_i$ where the corresponding $\mathbf{m}_i$ is not negligibly small is henceforth referred to as the *neighborhood* of the point $\boldsymbol{\mu}_i$. Due to this formulation, the value of the expression $l(\mathbf{Z}_B, \mathbf{m}_i)$ is high unless the points $\mathbf{Z}_B$ in the neighborhood of the embedding $\boldsymbol{\mu}_i$ follow the target distribution $\mathcal{N}(\boldsymbol{\mu_i}, \sigma\boldsymbol{I})$. In practice, we set the target distribution to $\mathcal{N}(0, \sigma\boldsymbol{I})$ and calculate the loss term for each $\boldsymbol{\mu}_i$ using centered embeddings $\mathbf{Z}_{B,\text{centered,i}} = \mathbf{Z}_B - \boldsymbol{\mu}_i$.

During training, $l(\mathbf{Z}_B, \mathbf{m}_i)$ is computed for each $i \in 1 \dots B$. The projection direction is independently sampled for each of these terms, $\mathbf{v}_i \sim \text{Unif}(\mathbb{S}^{d-1})$. The full Gaussian regularization loss of the batch, the masked SIGReg (mSIGReg) loss then becomes

$$\mathcal{L}_{\text{mSIGReg}}(\mathbf{Z}_B, \mathbf{M}_B) = \frac{1}{B} \sum_{i=1}^{B} l(\mathbf{Z}_{B,\text{centered,i}}, \mathbf{m}_i), \tag{17}$$

where $\mathbf{M}_B = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_B \end{bmatrix} \in \mathbb{R}^{B \times B}$ contains, as its rows, the masking vectors associated with each $\boldsymbol{\mu}_i$. Minimizing this quantity is possible only if the embeddings $\mathbf{Z}$ follow the distribution $\mathcal{N}(\boldsymbol{\mu}_i, \sigma\boldsymbol{I})$ in the vicinity of each state embeddings $\boldsymbol{\mu}_i$.

See Section E in the Appendix, which discusses the parameters $R_{\max}$ and T in greater detail, as well as how they should be selected in relation to the target standard deviation σ. Section F.1 presents an illustrative example of the mSIGReg method in operation.

3.3 Making the clusters meaningful

Since LGDE is based on an autoencoder architecture, its latent space can be expected to contain information related to the original data $\mathbf{X}$. This is also the primary rationale behind this design choice: the reconstruction constraint acts as a regularizer. However, as previously noted, this constraint is often insufficient, as vastly different data points $\mathbf{x}_i$ can be mapped to the same cluster. Therefore, LGDE incorporates the input data directly

into the clustering process via a third loss function (in addition to the reconstruction and mSIGReg objectives).

The goal of this loss function is to encourage each neighborhood in the latent space to achieve low variance with respect to $\mathbf{X}$. To this end, we first calculate the mean $\hat{\mathbf{x}}_i \in \mathbb{R}^D$ within the neighborhood of each $\boldsymbol{\mu}_i$:

$$\hat{\mathbf{x}}_i = \frac{\mathbf{m}_i \mathbf{X}_B}{\sum_n^B m_{in}}, \tag{18}$$

where $\mathbf{m}_i \in \mathbb{R}^{1\times B}$ is the mask vector associated with $\boldsymbol{\mu}_i$ and $\mathbf{X}_B \in \mathbb{R}^{B\times D}$ are the inputs of the training batch. We then compute distances from each $\mathbf{x}_i$ to each of these means, yielding $\mathbf{D}_{\hat{\mathbf{x}}} \in \mathbb{R}^{B\times B}$, where

$$\mathbf{D}_{\hat{\mathbf{x}},ij} = \text{dist}(\hat{\mathbf{x}}_i, \mathbf{x}_j), \tag{19}$$

with dist$(\cdot)$ being some distance function (*e.g.* the L2 norm if suitable). The local compactness loss is then defined as

$$\mathcal{L}_{\text{compact}} = \frac{1}{B^2} \sum_{i=1}^{B} \sum_{j=1}^{B} (\mathbf{M} \odot \mathbf{D}_{\hat{\mathbf{x}}})_{ij}, \tag{20}$$

where $\odot$ is the Hadamard product. This represents the average variance of $\mathbf{X}_B$ across all neighborhoods in the latent space. In practice, this loss function penalizes cases where a given $\mathbf{x}_i$ is located in a neighborhood where the average $\mathbf{x}$ differs significantly from it.

The previously defined compactness loss is minimized when each $\mathbf{x}_i$ is the only data point in its neighborhood (*i.e.*, $\mathbf{m}_i$ contains only one non-zero element, corresponding to the $\mathbf{z}_i$ value of that data point). This can potentially lead to a latent space where no cluster structures are formed. To prevent this, we add a fourth and final loss term to the overall objective function. The idea behind this loss term is to prevent the $\mathcal{L}_{\text{compact}}$ term from degrading the latent representation by requiring that each $\mathbf{x}_i$ resides in the same neighborhood as its nearest neighbor $\mathbf{x}_{i,\text{neigh}}$. For this loss term, we first calculate the distance matrix between the data points in the batch within the $\mathbf{X}$-space:

$$\mathbf{D}' \in \mathbb{R}^{B\times B} \quad \mathbf{D}'_{ij} = \text{dist}(\mathbf{x}_i, \mathbf{x}_j). \tag{21}$$

Next, we compute the weight matrix $\mathbf{W} \in \mathbb{R}^{B\times B}$ as follows: first, we set the diagonal values of the matrix $\mathbf{D}'$ to infinity. Then, we apply the softmax function to the (modified) matrix along the last dimension, yielding

$$\mathbf{W} = \text{softmax}(\frac{-\mathbf{D}'}{T_{\text{neigh}}}, \text{dim} = 1), \tag{22}$$

where T_{neigh} is a temperature parameter. If T_{neigh} is sufficiently small (*e.g.*, $1e^{-2}$), each row of $\mathbf{W}$ contains essentially a single non-zero entry located at the index of the nearest neighbor of $\mathbf{x}_i$ (denoted $\mathbf{x}_{i,\text{neigh}}$). To ensure that each $\mathbf{x}_i$ lies in the same neighborhood in the latent space as its nearest neighbor in the $\mathbf{X}$-space, we then calculate

$$\mathcal{L}_{\text{neigh}} = \frac{1}{B^2} \sum_{i=1}^{B} \sum_{j=1}^{B} 1 - (\mathbf{M} \odot \mathbf{W})_{ij}. \tag{23}$$

This loss reaches its minimum (*i.e.*, zero) when $\mathbf{M}_{ij} = \mathbf{W}_{ij} = 1$ for each pair of neighbors $\mathbf{x}_i, \mathbf{x}_j$, where $\mathbf{x}_j = \mathbf{x}_{i,\text{neigh}}$.

Incorporating additional loss functions into the general objective of LGDE offers a further advantage: it prevents representation collapse. Many deep learning-based clustering methods suffer from this type of collapse, where the network gets stuck in a (trivial) suboptimal solution. The reason is simple: the minima of the loss functions employed can be reached even when there is only a single cluster in the embedding space (all $\boldsymbol{\mu}_i$ correspond to the same point in the latent space, and $\mathbf{Z}$ is normally distributed around that point). Our method is no exception; the previously defined mSIGReg loss can be minimized by making the latent embeddings globally—rather than locally—Gaussian. $\mathcal{L}_{\text{compact}}$ effectively prevents this.

The full loss objective of LGDE is then

$$\begin{aligned}\mathcal{L} = &\mathcal{L}_{\text{recon}} \\ &+ \lambda_{\text{mSIGReg}}\mathcal{L}_{\text{mSIGReg}} \\ &+ \lambda_{\text{compact}}\mathcal{L}_{\text{compact}} \\ &+ \lambda_{\text{neigh}}\mathcal{L}_{\text{neigh}},\end{aligned}$$

where λ_{mSIGReg}, λ_{compact}, and λ_{neigh} are the weight coefficients for the mSIGReg loss, compactness loss, and neighbor loss, respectively. It is important to note that although LGDE is non-parametric in the sense that it does not require a k-value as input, the value of λ_{compact} influences the number of identified clusters; higher values lead to finer-grained clustering.

3.4 Cluster analysis using LGDE

LGDE does not produce explicit cluster labels for individual data points. The idea is to give the model "free rein" in forming latent embeddings, provided the resulting space exhibits local Gaussianity. When optimizing the mSIGReg loss function (assuming representation collapse does not occur), the latent space forms clearly distinct, dense regions with respect to both the $\mathbf{Z}$ values and, in particular, the state embeddings $\boldsymbol{\mu}$. Subsequently, assigning meaningful cluster labels is straightforward: we fit an additional (traditional) clustering model to the embeddings $\boldsymbol{\mu}$. Density-based clustering methods work well in practice. In this work, we have chosen HDBSCAN (McInnes et al., 2017) to determine the cluster label for each $\boldsymbol{\mu}_i$ and, by extension, for each $\mathbf{z}_i$. We set the minimum cluster size to one percent of the total number of input data points.

4 Experiments

4.1 Evaluation metrics

The number of clusters identified by LGDE cannot be precisely controlled. Furthermore, LGDE is explicitly encouraged to form compact clusters (with respect to the variable $\mathbf{X}$). Consequently, traditional clustering metrics — such as Normalized Mutual Information (NMI) and the Adjusted Rand Index (ARI) — are not the most informative in this context. Instead, we focus on more descriptive metrics, such as intra-cluster compactness and inter-cluster distinctness. Detailed descriptions of the relevant calculations can be found in Section B of the Appendix, where, for completeness, we also present ARI and NMI values, as well as comparisons with other methods (Section D).

4.2 Datasets

Due to space constraints, we present here the results for only one dataset: the MNIST dataset (LeCun et al., 2010) of handwritten digits. Results for the other two datasets — EMNIST (Cohen et al., 2017) and a dataset related to protein structure — can be found in the Appendix (Section D). A detailed description of each dataset and its preprocessing is provided in Section A. The training process and model architecture are described in detail in Section C.

4.3 Results

LGDE learns a highly informative and clusterable representation from the MNIST dataset. In the results presented here, the dimensionality of the LGDE latent space was set to just two. Despite this extremely tight bottleneck, LGDE succeeds in positioning the digits within the latent space in a sensible and clusterable manner (Figure 2a). Visualization of samples from selected clusters confirms that similar data points end up in the same cluster (Figure 2b). Indeed, the intra-cluster variance with respect to $\mathbf{X}$ is very low, whereas the data points in each cluster differ significantly from the means of other clusters (Figure 2c). Furthermore, mSIGReg has achieved its objective, as the random projections of the embeddings for each cluster closely match the target distribution $\mathcal{N}(0, 1)$ (Figure 2d).

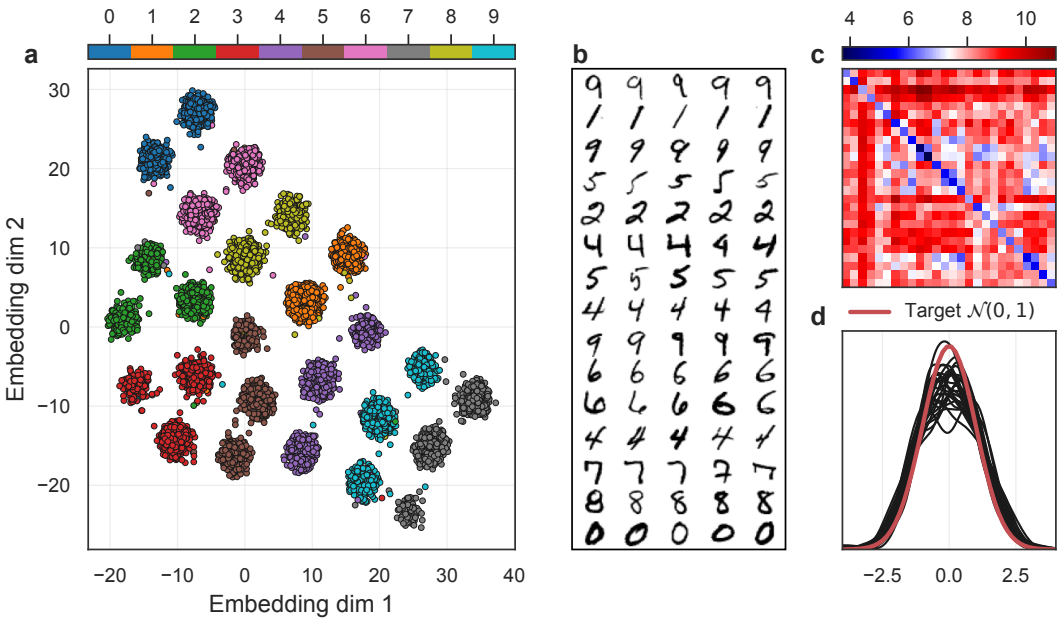


Figure 2: LGDE learns a highly informative representation of the MNIST dataset in a two-dimensional space. **a**, Learned embeddings colored according to ground truth (GT) classes. Similar digits cluster together, and all GT classes are split into at least two clusters. **b**, Random samples drawn from each cluster demonstrate that the clustering is meaningful. **c**, A matrix illustrating intra-cluster similarities and inter-cluster dissimilarities. The clusters are compact (diagonal elements are small) and distinct (off-diagonal elements are larger than the diagonal ones). See the detailed description of the matrix in the Appendix (Section B). **d**, mSIGReg imposes Gaussian characteristics on each cluster. Here, random projections of the embeddings from each cluster are compared to the target distribution $\mathcal{N}(0, 1)$, showing a close match.

We can examine the embedding learned by LGDE in greater detail. It is particularly worth asking what the state embeddings actually encode. To this end, we first decode all state embeddings $\boldsymbol{\mu}_i$ for each cluster individually (using LGDE's trained decoder). We then calculate the average of these decoded representations for each cluster to reveal that cluster's "state." Figure 3a displays these decoded representations; each row contains all decoded representations from clusters sharing a specific GT majority class. These decoded representations are clearly meaningful; they represent common ways in which each digit can be written. Consider, for example, the two states of the digit 1: one is slanted to the right, while the other is upright. Similar features can be observed for the other digits as well. To verify that these decoded states are meaningful, we demonstrate that they lie within regions of the original data space $\mathbf{X}$ where the data is dense. We do this by calculating the L2 distances from each of these representative decoded representations to the original data points and recording the average distance to the k nearest neighbors for various values of k. We also calculate the same metric for the data points themselves. The extracted states are clearly located in regions of space $\mathbf{X}$ where the data are dense, as they possess significantly more close neighbors than the data generally (Figure 3b).

Finally, we can assess the extent to which LGDE has learned directional significance within each cluster. To do this, we require an easily interpretable metric that can be calculated for each input image. We select the image's total intensity (the sum of pixel intensities) as this metric. The underlying idea is that this metric reflects the size of the drawn digit (either its height/width or its stroke thickness). We can then calculate the correlation between this metric and the various directions within the cluster. Figure 3c displays selected clusters, with the color of the circle surrounding each cluster determined by this correlation coefficient. A general direction of increasing image intensity is clearly discernible within each cluster. Furthermore, visualizing the samples along the directions of highest correlation illustrates

how the digit grows—whether in terms of its overall size (as with the digit 8) or its width (as with the digits 1 and 9). We conclude that the partitioned nature of the embedding has enabled the LGDE method to learn directional trends within each cluster.

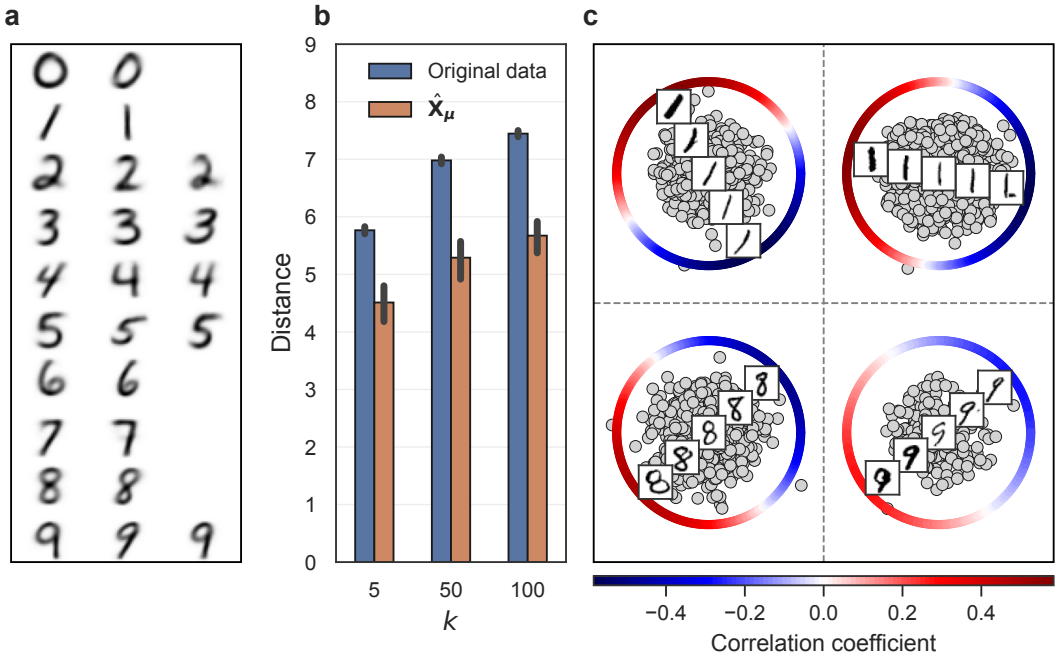


Figure 3: Analysis of state embeddings and perturbation vectors. **a**, Representative states are revealed by decoding the state embeddings (see the text). **b**, Representative digits are located in dense regions of the input space. Shown here is the average L2 distance to the k-nearest neighbors for both the representative digits and the entire dataset. Representative digits have more close neighbors in the dataset than data points generally do. **c**, LGDE has learned directional trends within each cluster (see the text). A general direction corresponding to a "larger digit" is observable in each cluster. Along these directions, the digits 1 and 9 widen, while the overall size of the digit 8, for example, increases.

5 Conclusions

In this article, we present the LGDE method as a novel approach to generating clusterable representations using deep neural networks. We derive a new loss function, masked SIGReg (mSIGReg), the optimization of which yields the desired clustering. LGDE does not require prior knowledge regarding the number of clusters the latent representation should contain. Furthermore, we demonstrate that each input data element can be represented as the sum of a state embedding and a perturbation vector. This framework aligns naturally with the mSIGReg objective but also offers additional advantages, as the approach renders the latent representation highly interpretable: state embeddings correspond to representative states of the system, while perturbation vectors capture directional information. In other words, LGDE learns the *essential* characteristics of the datasets being processed.

LGDE performs well on image datasets (MNIST and EMNIST) in both two- and ten-dimensional latent spaces. Minimizing the compactness loss results in clustering where the average cluster compactness is very high: LGDE has subdivided the original classes in a meaningful way. Similarly, for protein structure data, as an example of scenarios with limited data, LGDE learns a highly useful and clusterable representation. This highlights LGDE's ability to generalize results across diverse datasets and to function effectively even in data-scarce scenarios. In summary, LGDE represents a significant advancement alongside established clustering methods. It is an efficient and versatile framework that enables the generation of interpretable, clusterable representations from complex data.

6 Reproducibility

To enable reproducibility of our results, we supply a standalone directory containing all code and dependencies required to reach the clustering results shown in Figure 2. These files are found in the accompanying file lgde_reprod.zip.

7 AI Use Statement

We used an LLM (ChatGPT 5.5) to format the text in certain sections of this article. Additionally, we used the same model for coding assistance, particularly for checking and correcting PyTorch syntax. All decisions regarding the methods and experiments were made by the authors.

8 Acknowledgments

We thank the Research Council of Finland, the Sigrid Juselius Foundation, Helsinki Institute of Life Science (HiLIFE) Distinguished Fellow Program, and the Lundbeck Foundation for support. We acknowledge the computing resources provided by the CSC – IT Center for Science Ltd. (Espoo, Finland) and LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC and the LUMI Consortium.

References

Charles E Antoniak. Mixtures of Dirichlet processes with applications to bayesian nonparametric problems. *Ann. Stat.*, pages 1152–1174, 1974.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *Preprint arXiv:2511.08544*, 2025.

Reza Bayat. SIGReg from first principles: A step-by-step construction of an anti-collapse regularizer for JEPAs, 2026. URL `https://rezabyt.github.io/blogposts/sigreg-tutorial.html`. Blog post.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Adv. Neural Inf. Process. Syst.*, 19, 2006.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic Acids Res.*, 28(1):235–242, 2000.

Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In *Proceedings of the Web Conference 2020*, pages 1400–1410, 2020.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5880–5888. IEEE, 2017.

Gang Chen. Deep learning with nonparametric clustering. *Preprint arXiv:1501.03084*, 2015.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017. doi: 10.1109/ijcnn.2017.7966217.

Harald Cramér and Herman Wold. Some theorems on distribution functions. *J. London Math. Soc.*, 1(4):290–294, 1936.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with Gaussian mixture variational autoencoders. *Preprint arXiv:1611.02648*, 2016.

Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *Ann. Stat.*, pages 209–230, 1973.

Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI'17: Proceedings of the 26th International Joint Conference on Artificial Intelligence*, volume 17, pages 1753–1759, 2017.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Preprint arXiv:1312.6114*, 2013.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Leland McInnes, John Healy, Steve Astels, et al. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.

Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

Gáspár Pándy-Szekeres, Christian Munk, Tsonko M Tsonkov, Stefan Mordalski, Kasper Harpsøe, Alexander S Hauser, Andrzej J Bojarski, and David E Gloriam. GPCRdb in 2018: Adding GPCR structure models and ligands. *Nucleic Acids Res.*, 46(D1):D440–D446, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *Preprint arXiv:1511.06434*, 2015.

Meitar Ronen, Shahaf E Finder, and Oren Freifeld. Deepdpm: Deep clustering with an unknown number of clusters. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9851–9860. IEEE, 2022.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, 2007.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11(12), 2010.

Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering: A deep attentional embedding approach. *Preprint arXiv:1906.06532*, 2019.

Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian k-means. *IEEE Trans. Knowl. Data Eng.*, 27(1):180–192, 2014.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487. PMLR, 2016.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *International Conference on Machine Learning*, pages 3861–3870. PMLR, 2017.

Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

Appendix

A Datasets used in experiments

MNIST consists of 28 × 28 pixel images of handwritten digits (0–9). The training set comprises 60,000 samples, and the test set contains an additional 10,000 samples. EMNIST is a more challenging extension of the MNIST dataset that includes handwritten letters of the English alphabet in addition to digits. For the EMNIST dataset, we use a so-called balanced split; it contains 47 classes, as well as approximately 110,000 training samples and 20,000 test samples. The data for each image set were normalized to the range [0, 1] prior to the training phase.

The protein structure dataset covers experimentally determined three-dimensional (3D) structures of Class A GPCRs. To compile this dataset, we gathered all published structures of these receptors from the Protein Data Bank (PDB) (Berman et al., 2000), identified the common amino acids, and extracted the corresponding alpha-carbon atoms from each structure. We then aligned each set of atoms to a reference structure by minimizing the root-mean-square deviation (RMSD). Ultimately, we obtained 1035 aligned structures, each defined by the 3D coordinates of 166 atoms, resulting in a dataset with a shape of (1035, 498). Each feature of this dataset was normalized to zero mean and unit variance prior to training. Additionally, we retrieved classification data for each structure from the GPCRdb database (Pándy-Szekeres et al., 2018). This information includes whether a signaling protein (G protein) is bound to the structure, which ligands (if any) are bound, and the receptor's subfamily. We stress that the number of training samples in this dataset is clearly not optimal. Therefore, testing with this dataset serves two purposes: it demonstrates the applicability of LGDE to various data domains and illustrates the model's performance even in situations where the available amount of data is limited.

B Evaluation metrics

LGDE is explicitly encouraged to form clusters that are compact with respect to $\mathbf{X}$. For this reason, the most informative metric for LGDE's performance relates to the variance of $\mathbf{X}$ within each cluster; the desired outcome is for data points within a cluster to be close to that cluster's mean, while the data points of a given cluster should be far from the means of other clusters. The mean of the j-th cluster is defined as follows:

$$\mathbf{c}_j = \frac{1}{n_j} \sum_{i:y_i=j} \mathbf{x}_i, \tag{24}$$

where n_j is the number of data points assigned to the cluster and $\mathbf{y}$ contains the cluster assignments.

The mean intra-cluster distances are then computed as

$$d_{\text{intra},j} = \frac{1}{n_j} \sum_{i:y_i=j} \text{dist}(\mathbf{x}_i, \mathbf{c}_j), \tag{25}$$

with $\text{dist}(\cdot)$ being some distance metric (we use the L2 norm). This metric is also termed the *cluster compactness*. The inter-cluster distances from cluster m to the mean of cluster j are similarly computed as

$$d_{\text{inter},mj} = \frac{1}{n_m} \sum_{i:y_i=m} \text{dist}(\mathbf{x}_i, \mathbf{c}_j). \tag{26}$$

This is the *cluster distinctness*.

We present the clustering results based on these metrics by visualizing a $k \times k$ matrix (assuming there are k identified clusters); the diagonal elements of the matrix correspond to the average intra-cluster distances, and the other elements correspond to the average inter-cluster distances (*e.g.*, Figure 2c).

C Details on the training procedure and model architecture in experiments

We have observed that, during the initial phase of training, the mSIGReg term can exert an excessively strong influence, leading to representation collapse (*i.e.*, the formation of a single cluster in the latent space). Consequently, for the results presented in this study, we employ a delayed weighting schedule for the mSIGReg loss term. We set the term's weight to zero for the first 30 % of the training process. Thereafter, we linearly increase the weight to $\lambda_{\text{mSIGReg, final}} = 10.0$. This allows the compactness and neighborhood loss terms to take effect before distribution alignment begins, thereby preventing representation collapse. The weights for the other loss terms are set to 1: for image datasets, $\lambda_{\text{compact}} = \lambda_{\text{neigh}} = \lambda_{\text{recon}} = 1$. For the protein dataset, we observe that the compactness loss term has a significantly stronger effect, so we set $\lambda_{\text{compact}} = 0.001$. The relatively high value of $\lambda_{\text{mSIGReg, final}} = 10.0$ used in the final stage of training ensures that the clusters truly conform to a Gaussian distribution.

For the MNIST results, the model is trained for 200 epochs with a batch size of 2000 samples (see Section E.4 for the rationale behind such a large batch size). EMNIST is a more challenging dataset, so training lasts for 400 epochs. The protein dataset contains only about 1000 data points. For this dataset, we set the batch size to 980, meaning each epoch consists of only a single batch; consequently, we train the model on this dataset for 2000 epochs. In all training runs, we use a learning rate of 0.001 and the Adam optimization algorithm (Kingma and Ba, 2014).

For image datasets, we use a convolutional trunk in the encoder. This sub-network transforms the input from size 28×28 into an intermediate embedding of dimension 500:

$$(28 \times 28 \times 1) \to (14 \times 14 \times 32) \to (7 \times 7 \times 64) \to (7 \times 7 \times 128) \to 500.$$

Each convolution is followed by batch normalization and the SiLU activation function. For the protein structure dataset, the backbone is a fully connected feed-forward network that transforms input data of dimension 498 into an intermediate embedding of dimension 500:

$$498 \to 500 \to 500.$$

The network employs ReLU activation and layer normalization after the hidden layer. For each dataset, the $\boldsymbol{\mu}$ and $\boldsymbol{\epsilon}$ predictors are identical, and the dimensions of their hidden layers are

$$f_{\boldsymbol{\mu}} : 500 \to 2000 \to \text{latent dim}$$
$$f_{\boldsymbol{\epsilon}} : 500 \to 2000 \to \text{latent dim} + 1,$$

both with ReLU activation and layer normalization.

The decoder is the same for all datasets (except for the output dimension). The architecture of this sub-network is

$$\text{latent dimension} \to 2000 \to 500 \to 500 \to \text{output dimension}.$$

ReLU activation and layer normalization are applied after each hidden layer.

Furthermore, to make the representation learned by the LGDE more robust, we add noise to all inputs before the encoder; specifically, the inputs are

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i + 0.1\epsilon,$$

where $\epsilon \sim \mathcal{N}(0, 1)$. For image data, we use the binary cross-entropy as the reconstruction loss, while for protein data we use mean squared error.

D Additional Results

D.1 EMNIST

As with the MNIST dataset, LGDE learns a highly clusterable representation from the more challenging EMNIST dataset. Figure 4a shows the embedding space, where colors are determined by the GT classes. Samples drawn from the clusters demonstrate that similar data points are assigned to the same cluster (Figure 4b). The number of identified clusters is 37. Intra-cluster distances are generally very small compared to inter-cluster distances (Figure 4c) (the matrix was calculated as described in Section B). The resulting clusters closely follow a Gaussian distribution; random projections of the embeddings for each cluster closely match the target distribution $\mathcal{N}(0, 1)$ (Figure 4d).

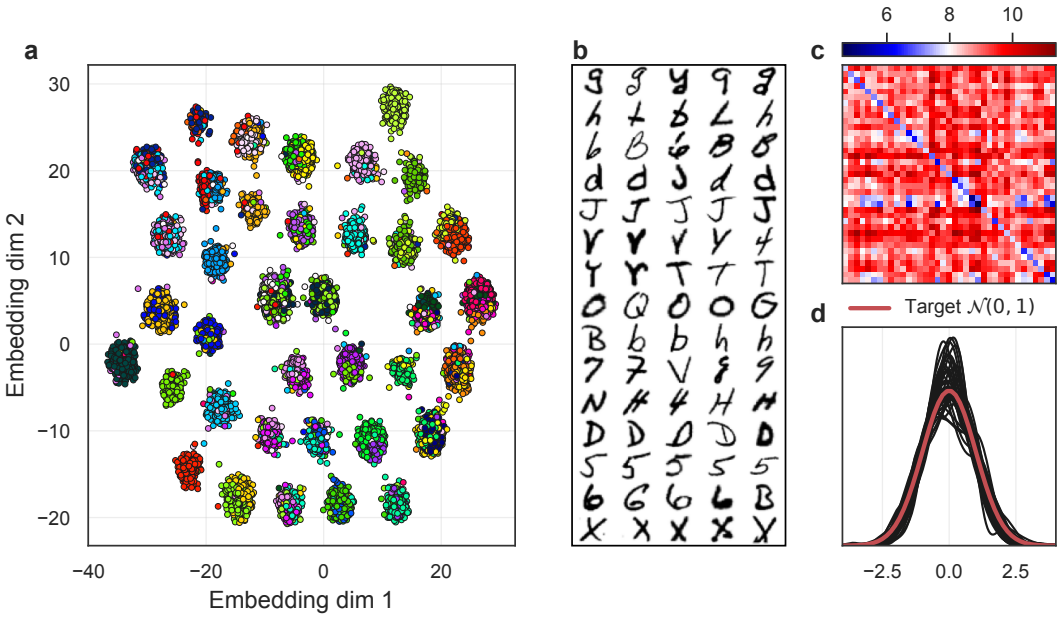


Figure 4: Results for the LGDE model trained on the EMNIST dataset using a two-dimensional latent representation. **a**, Learned embeddings colored according to GT classes. The space is clearly clusterable. The number of identified clusters is 37. **b**, Samples drawn from the clusters demonstrate that similar images have been successfully grouped together (each row shows five samples from the respective cluster). **c**, The identified clusters are compact and well-separated: within each cluster, distances to the cluster mean are small, whereas distances to the means of other clusters are large (Section B). **d**, The clusters are distributed according to the target distribution $\mathcal{N}(0, 1)$. Here, the target probability density function is overlaid on random projections of the embeddings for each cluster.

Mapping the entire EMNIST dataset into a two-dimensional latent space in such a meaningful way is undeniably extremely challenging. This highlights the capability of LGDE.

D.2 Protein data

LGDE is capable of generating a useful representation of protein structural data. Figures 5a–d display these embeddings, colored according to various identifiers. Although the sample size is small (1,035), the test set comprising 10 % of the data is encoded into a grouping pattern similar to that of the training data (panel a). The grouping reveals the activation state of the proteins: the embedding clearly depends on whether or not a G-protein is bound to the receptor (G-protein binding indicates the protein's active conformation) (panel b). Similarly, the grouping highlights the significance of other molecules—ligands—that bind to GPCRs; structures bound to activating agents (agonists) are distinguished from those bound

to inactivating agents (antagonists or inverse agonists) and from the apo state (no bound ligand) (panel c). Furthermore, the grouping takes the receptor subfamily into account (panel d).

This example demonstrates that LGDE is applicable to various data domains, not just image data analysis. It also shows that the method works even when the training dataset contains relatively few data points. Furthermore, direct conclusions of biological and medical significance can be drawn from the results: these proteins possess multiple active conformations (panels b and c), and the conformation depends on the receptor subfamily (panel d).

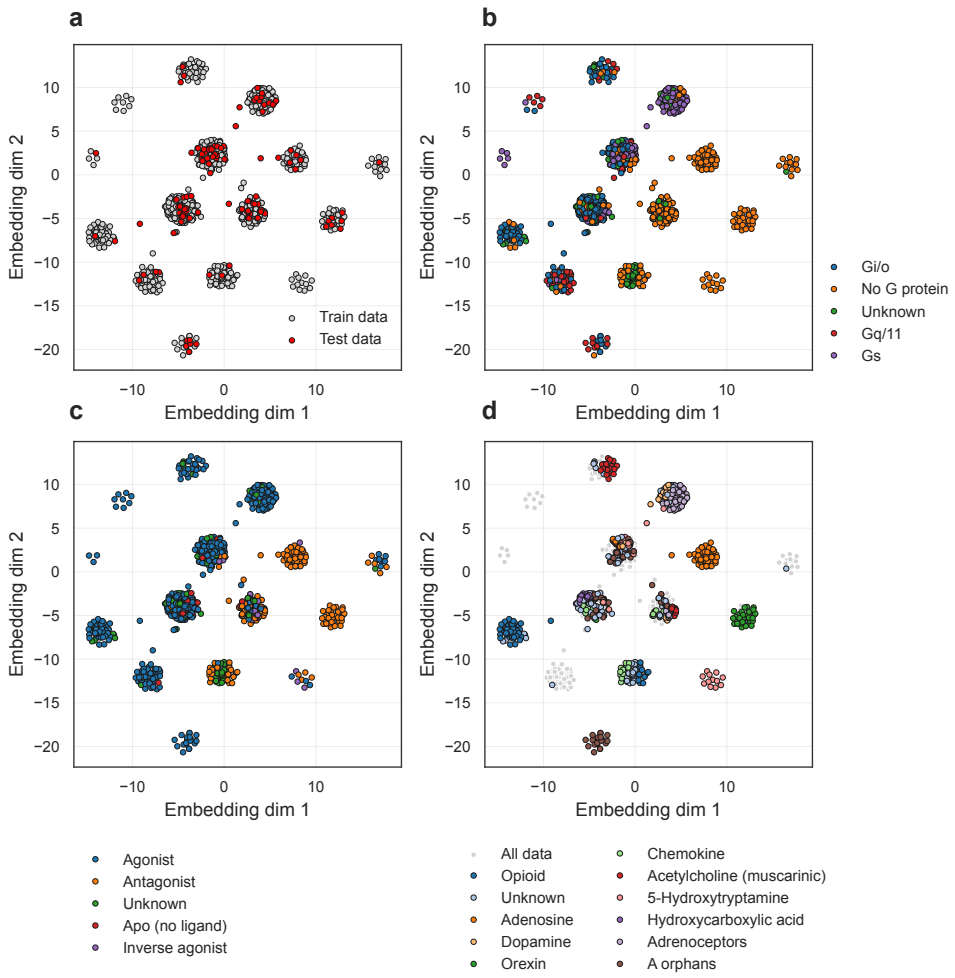


Figure 5: Results from LGDE trained on protein structural data. **a**, Although the training set contains a limited number of data points, LGDE learns a robust embedding, as the leave-out test data fall within identified clusters. **b**, Learned embeddings colored according to the signaling protein associated with the protein structure. Structures containing a G-protein are generally assumed to be in a biologically active state; thus, LGDE has learned to group structures in a physiologically meaningful way. **c**, Similarly, the embeddings capture information about ligands bound to the structure. Agonist-bound (active) structures typically cluster together, whereas antagonist-bound (inactive) structures form distinct clusters. The embeddings therefore encode information about bound ligands in addition to the signaling protein. **d**, The clustering also reflects receptor subfamilies: LGDE learns to account for subtle structural differences among class A GPCRs as well as their physiological states.

D.3 Clustering Performance (ARI and NMI)

We report here the classic clustering performance metrics (NMI and ARI) for the MNIST dataset. Only the best value we obtained is reported, along with the LGDE parameters we used and the number of clusters the model identified.

Table 1: Clustering performance on MNIST in terms of ARI and NMI.

Metric	K-means[1]	AE[2]	SAE[3]	DAE[4]	GAN[5]	JULE[6]	DEC[7]	DAC[8]	LGDE_{2d}	LGDE_{10d}
NMI	0.4997	0.7257	0.7565	0.7563	0.7637	0.9130	0.7716	**0.9246**	0.8256	0.8693
ARI	0.3652	0.6139	0.6393	0.6467	0.7360	0.9270	0.7414	**0.9406**	0.6988	0.8265

[1]Wang et al. (2014), [2]Bengio et al. (2006), [3] Ng et al. (2011), [4]Vincent et al. (2010), [5] Radford et al. (2015), [6] Yang et al. (2016), [7]Xie et al. (2016) [8] Chang et al. (2017)

The results achieved with LGDE are competitive with the other listed methods. We present two values for LGDE corresponding to the best results in both two-dimensional and ten-dimensional latent spaces (indicated by a subscript in the model name). In the two-dimensional latent space model, the weight factor for the compactness loss, λ_{compact}, is 0.1, and the number of identified clusters is 13. For the ten-dimensional model, λ_{compact} is 10.0, and the number of clusters is 11. The other loss function weight factors are the same as those described in Section C. The performance of the two-dimensional model is particularly noteworthy, given that most competing models utilize a latent space of at least ten dimensions. This serves as further evidence of the LGDE framework's ability to learn effectively.

D.4 How LGDE learns

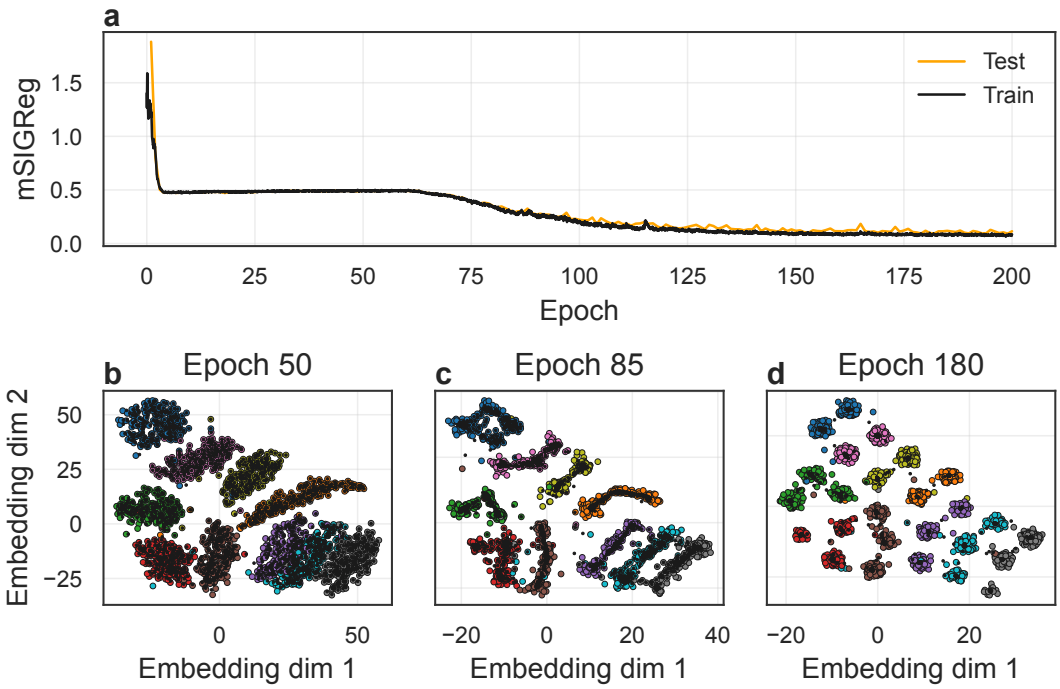


Figure 6: Examination of the LGDE learning process using the MNIST dataset. See the text for details.

We examined the progression of the LGDE learning process. Figure 6a shows the mSIGReg loss values on the MNIST dataset as a function of training epochs (with a latent space dimensionality of two). Figures 6b–d display the latent space after selected epochs (colored according to the true classes). Black dots represent the latent embeddings $\boldsymbol{\mu}$. In the early stages of training, before the mSIGReg loss is active (see Section C), the space exhibits general local similarity regarding $\mathbf{X}$ due to the effect of the compactness loss, but it is not yet locally Gaussian (Figure 6b). The weight of the mSIGReg term is gradually increased starting from epoch 60. At epoch 85, the onset of tighter clustering is observed as the latent embeddings begin to pack together (Figure 6c). By epoch 180, the mSIGReg value has decreased further and clustering is distinct; the latent embeddings form very tight clusters (Figure 6d).

D.5 Ablation studies

We investigated the roles of the four different loss terms in the LGDE objective function. To this end, we trained four models on the MNIST dataset, disabling each loss term at a time. In all other respects, the trained models were identical to those described in Section C.

When the neighborhood loss function $\mathcal{L}_{\text{neigh}}$ is disabled, the compactness loss function dominates the others, resulting in a highly dispersed latent space (Figure 7a). When $\mathcal{L}_{\text{mSIGReg}}$ is disabled, the latent embeddings retain a significant amount of information from the dataset $\mathbf{X}$ due to the compactness and reconstruction loss functions (Figure 7b); this configuration yields a better representation than a traditional autoencoder. Intuitively, when $\mathcal{L}_{\text{compact}}$ is disabled, the $\mathcal{L}_{\text{mSIGReg}}$ loss function takes over, leading to a single Gaussian distribution in the latent space (Figure 7c). Interestingly, LGDE's performance does not degrade significantly when the reconstruction loss function is disabled (Figure 7d); the other loss functions are clearly sufficient for forming the latent space and performing clustering. However, the reconstruction requirement offers advantages for LGDE, specifically the ability to visualize reconstructed embeddings and, above all, the state embeddings $\boldsymbol{\mu}$.

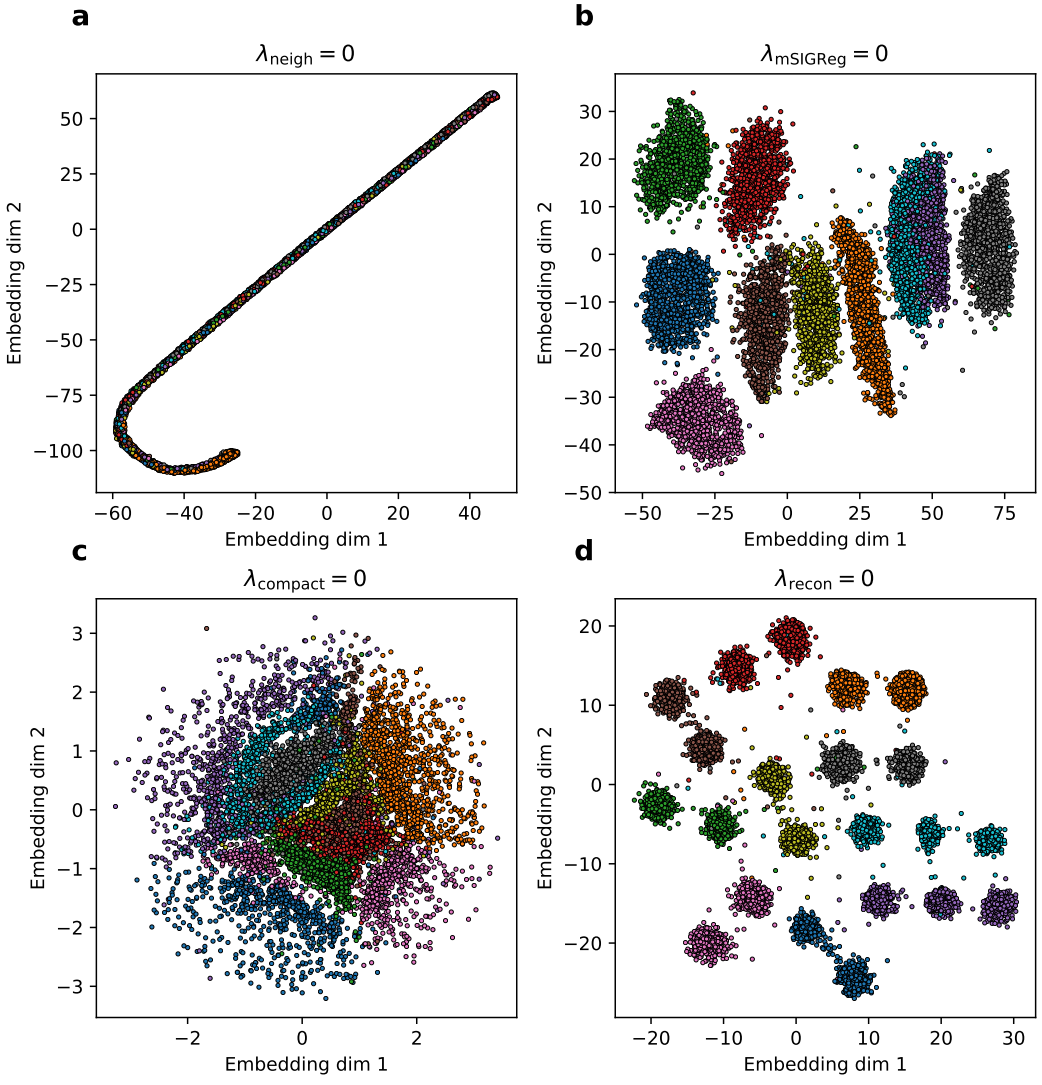


Figure 7: Ablation studies of LGDE's losses. See the text for analysis of each result.

E Choosing the Parameters of LGDE

E.1 The compactness loss weight

We investigate the effect of the λ_{compact} parameter on the final embedding space. Figure 8 illustrates how the metrics under consideration change as the parameter value varies, using either a two-dimensional or a ten-dimensional latent space. The results are aggregated from training runs performed on the MNIST dataset (three runs per λ_{compact} value with random initializations). As expected, the number of identified clusters increases as the λ_{compact} value rises. However, there are certain points to consider. First, with the MNIST dataset, if the parameter is too small, representation collapse occurs, resulting in a very low number of clusters—or even just a single cluster. Second, if the parameter is too large, the compactness loss dominates the outcome, and clustering quality deteriorates again (many points remain isolated within their clusters, manifesting as a low cluster count due to the minimum cluster size constraint in HDBSCAN) (Figure 8a). This is reflected in Figure 8d, where HDBSCAN classifies a larger proportion of data points as noise. Similar trends are observed in other metrics (cluster homogeneity (Rosenberg and Hirschberg, 2007) and compactness) (Figures 8b–c): performance degrades if the value is either too low or too large.

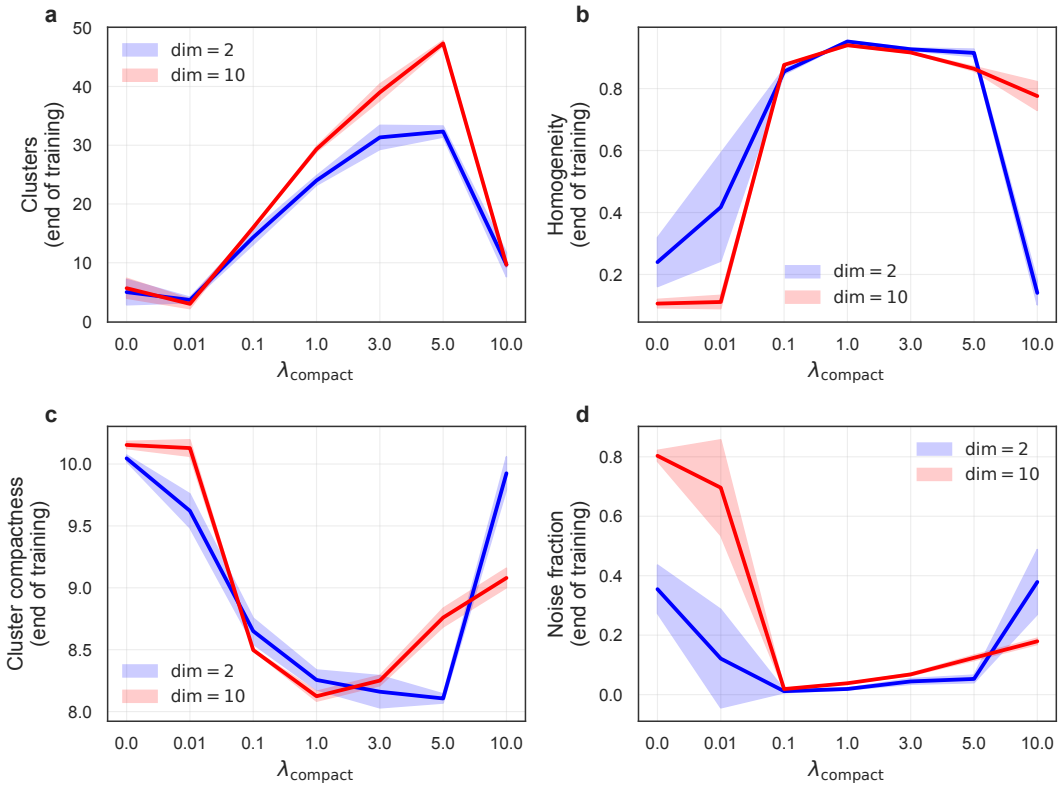


Figure 8: Studies of the impact of the compactness loss weight λ_{compact} on metrics of interest. See the text for analysis of these data.

E.2 The trapezoidal quadrature

The trapezoidal quadrature rule defines how to perform the integration over t-values to compare two characteristic functions (CFs) (Equation 12). The definition of this quadrature is analyzed in Bayat (2026). The integration interval $[a, b]$ depends on the parameter λ_T of the weighting function $w(t) = e^{-t^2/(a\lambda_T^2)}$. This parameter determines the frequency ranges of the characteristic functions on which the test primarily focuses: small values weight

low frequencies, whereas large values target higher frequencies. In Balestriero and LeCun (2025), the parameter is set to $\lambda_T = 1.0$. With this choice, the integration interval is set to $[0.2, 4.0]$. There are reasons for this choice: at values below 0.2, both characteristic functions (empirical and target) are structurally very small, resulting in a negligible impact on the loss term. Conversely, at $t = 4.0$, the weight term is three orders of magnitude smaller than its peak value at $t = 0$, so its contribution to the loss is again insignificant.

We observe that LGDE is quite robust regarding the choice of λ_T, provided the integration interval is reasonable. However, we obtained better results using $\lambda_T = 5.0$ instead of $\lambda_T = 1.0$; thus, the results presented in this article were obtained using that value. With this value of λ_T, we set the integration interval to $[0.2, 6]$.

The number of integration points K also matters, though only up to a certain point. If K is too small (*e.g.*, 4), the resolution of the test is too low and optimization fails. In practice, however, any value greater than 16 suffices. In this work, we set $K = 32$.

E.3 The masking vector parameters

Cluster formation in mSIGReg optimization depends on the parameters used to calculate the masking vector (Equation 16). The goal is thus to determine the maximum allowed displacement $R_{\max}$ and the temperature parameter T. These values determine how far each $\mathbf{z}_i$ can be placed from the state embedding $\boldsymbol{\mu}_i$ and the extent of the neighborhood around each state embedding. Since we want the embeddings to correspond to the target Gaussian distribution, the value of $R_{\max}$ should be reasonable relative to the σ value of the target distribution (*e.g.*, 4σ). Otherwise, an embedding might be placed in a region where the probability under the corresponding Gaussian distribution is extremely low. Figure 9 shows the masking weight as a function of the distance from the mean for various values of T; in this case, $R_{\max} = 5\sigma$. The lower the value of T, the steeper the decline in weight. The best results are obtained when T is not too small; this is likely because it allows for smoother gradients during the mSIGReg computation. In this work, we use the values $T = 1$ and $R_{\max} = 5$ for all examples. The σ value of the target distribution is set to 1 in all cases. This setting is illustrated by the solid line in Figure 9.

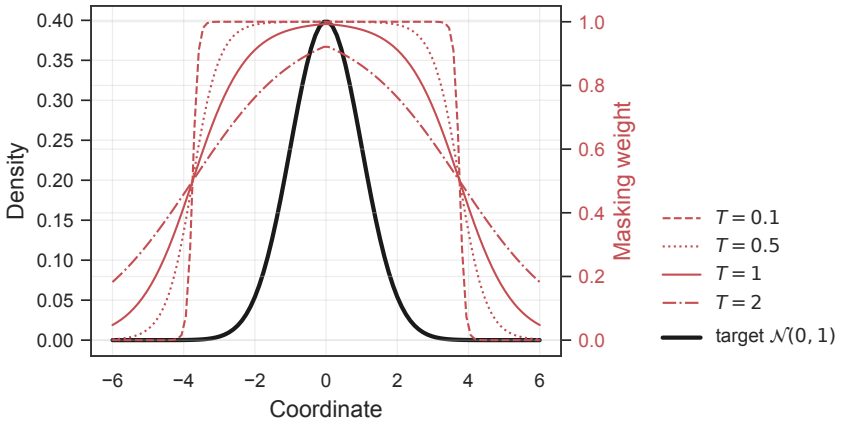


Figure 9: Illustration of the effect of the T parameter on the masking weight as a function of the distance from the mean. In this case, the target $\sigma = 1$ and the distance parameter $R_{\max} = 5$.

E.4 Batch Size

LGDE relies heavily on a novel mSIGReg objective function. One feature of the mSIGReg function that should not be overlooked is its dependence on the batch size. The method aims to partition latent embeddings into clusters by locally fitting an empirical characteristic function (CF) to a target CF. Due to the finite amount of data, the empirical CF is noisy. In other words, the mSIGReg loss value for a given neighborhood is influenced by the number

of data points within it; even samples drawn from the target distribution result in high mSIGReg loss if the sample size is too small. This imposes a practical upper limit on the number of clusters into which the set of embeddings can be partitioned. Fortunately, the problem is not severe: the neighborhood mSIGReg loss stabilizes with a relatively small number of samples. Figure 10 shows mSIGReg loss values calculated using varying numbers of samples *drawn from the target distribution*. With approximately 30 samples, the loss is already quite low, and mSIGReg optimization would yield the desired result. Based on this heuristic, the maximum number of clusters identified by LGDE would be 40 when using a batch size of 1200.

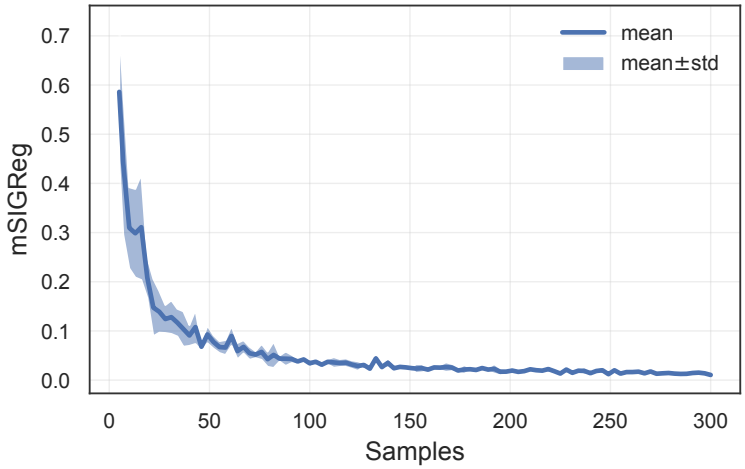


Figure 10: Evaluation of mSIGReg as a function of samples drawn from the target distribution in the two-dimensional case. Even with a moderate number of samples (a few tens), the loss is quite small, and optimization would likely proceed well.

F mSIGReg — Supplementals

F.1 Intuition Behind mSIGReg

Let us consider an example illustrating the operation of the mSIGReg method. Examine the two-dimensional embedding space shown in the top-left corner of Figure 11. The space clearly contains two dense regions. Projecting the data onto most directions in this space would reveal the bimodal nature of the data. Consequently, the SIGReg value (using the standard normal distribution as the target distribution) would be high, as the projections do not follow a Gaussian distribution (Figure 11, top-right). In other words, Equation (12) would yield a relatively large value. Next, consider the quantity $l(\mathbf{Z}, \mathbf{m})$ (Equation (15)), assuming a suitable masking vector is used. Let $m_n = 1$ for each point $\mathbf{z}_n$ in one dense region and $m_n = 0$ for points in the other region (Figure 11, bottom-left). When calculating the masked projections (weighting the contribution of each projection to the resulting density based on the masking vector), the result is approximately a Gaussian distribution. However, due to the nature of the projection, this distribution is not centered at the origin. When we correct this (by subtracting the mean from the masked projections), the resulting distribution matches the target distribution almost perfectly (Figure 11, bottom right). Thus, $l(\mathbf{Z}, \mathbf{m})$ would take a very small value for that set of embeddings and masking vector.

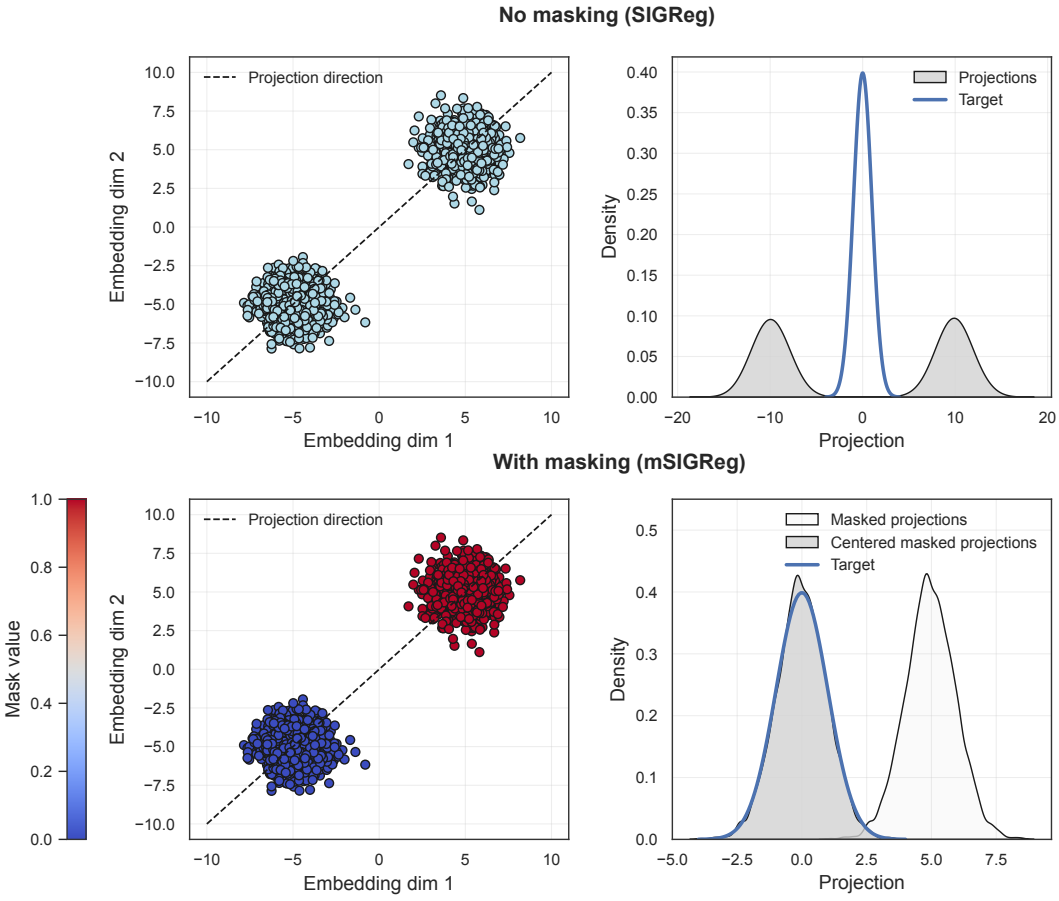


Figure 11: An illustrative example of the motivation behind the mSIGReg method. Without masking, the bimodal latent representation results in projections that do not align with the target (top row). With appropriate masking (mSIGReg), the masked projections align well (bottom row).

F.2 Computational complexity of mSIGReg

The memory and computational requirements of the mSIGReg method depend on the number of data points in the batch and the number of integration points used in the trapezoidal rule (see Section E.2). With N data points and K integration points, calculating the intermediate results of the mSIGReg method requires $O(N^2K)$ arithmetic operations and $O(N^2K)$ memory.

F.3 mSIGReg in Code

The following is a PyTorch implementation of the mSIGReg loss.

```
def mSIGReg(Z,
            mu,
            M,
            t_knots,
            weights,
            target_sigma=1.0):
    """
    Computes the mSIGReg loss for given latent representation.

    Z:              current latent embeddings (B, d)
    mu:             current latent state embeddings (B, d)
    M:              current masking vectors (B, B)
    t_knots:        frequency knots (T,)
    weights:        trapezoidal weights multiplied by Gaussian
                    weight term (w(t_k)) (T,)
    target_sigma    the standard deviation of the target Gaussian (scalar)
    """
    U = torch.randn_like(mu)
    U = U / U.norm(dim=1, keepdim=True)
    H = U @ Z.T - (mu * U).sum(dim=1, keepdim=True)
    angles = t_knots[None, :, None] * H[:, None, :]
    C = (torch.cos(angles) * M[:,None,:]).sum(dim=-1) / M.sum(dim=1,
    ↪  keepdim=True)
    S = (torch.sin(angles) * M[:,None,:]).sum(dim=-1) / M.sum(dim=1,
    ↪  keepdim=True)
    G = torch.exp(-t_knots ** 2 * target_sigma**2 / 2)
    diff_sq = (C - G[None, :]) ** 2 + S ** 2
    return (weights * diff_sq).sum(axis=1).mean()
```