Connectivity-Optimized Representation Learning via Persistent Homology

Christoph D. Hofer, Roland Kwitt
University of Salzburg

Mandar Dixit
Microsoft

Marc Niethammer
UNC Chapel Hill
Q: What makes a **good** representation?

- Ability to **reconstruct** (→ prevalence of autoencoders)
- **Robust** to perturbations of the input
- Useful for **downstream tasks** (e.g., clustering, or classification)
- etc.
**Q: What makes a **good** representation?**

- Ability to **reconstruct** (→ prevalence of autoencoders)
- **Robust** to perturbations of the input
- Useful for **downstream tasks** (e.g., clustering, or classification)
- etc.

**Common idea:** Control (/or enforce) properties of (/on) the latent representations in \( Z \).

Unsupervised representation learning
**Q: What makes a **good** representation?**

- **Ability to reconstruct** (→ prevalence of autoencoders)
- **Robust** to perturbations of the input
- **Useful for downstream tasks** (e.g., clustering, or classification)
- etc.

**Common idea:** Control (or enforce) properties of (on) the latent representations in $\mathcal{Z}$.

![Diagram of unsupervised representation learning](image)

$\mathcal{X} \rightarrow \mathcal{Z} \rightarrow \mathcal{X}$

$\theta$: $\mathcal{X} \rightarrow \mathcal{Z}$

$\phi$: $\mathcal{Z} \rightarrow \mathcal{X}$

$\text{Rec}[x, \hat{x}] + \text{Reg}$

**Contractive AE's** [Rifai et al., ICML '11]
Q: What makes a good representation?

- Ability to reconstruct (→ prevalence of autoencoders)
- Robust to perturbations of the input
- Useful for downstream tasks (e.g., clustering, or classification)
- etc.

Common idea: Control (or enforce) properties of (on) the latent representations in $\mathcal{Z}$.

\[ f_\theta : \mathcal{X} \rightarrow \mathcal{Z} \]
Encoder

\[ g_\phi : \mathcal{Z} \rightarrow \mathcal{X} \]
Decoder

\[ x \xrightarrow{\text{Perturb, or zero-out}} \]

\[ \hat{x} \xrightarrow{\text{Rec}[x, \hat{x}]} \]

Denoising AE's [Vincent et al., JMLR '10]
Q: What makes a good representation?

- Ability to reconstruct (→ prevalence of autoencoders)
- Robust to perturbations of the input
- Useful for downstream tasks (e.g., clustering, or classification)
- etc.

Common idea: Control (or enforce) properties of (on) the latent representations in $Z$.

Unsupervised representation learning

$\theta$: $X \rightarrow Z$

$\theta$: $X \rightarrow Z$

Robust to perturbations of the input

Useful for downstream tasks (e.g., clustering, or classification)

etc.

Common idea: Control (or enforce) properties of (on) the latent representations in $Z$.

Encoder

Decoder

$g_\phi: Z \rightarrow \hat{x}$

$Rec[x, \hat{x}] + Reg$

Sparse AE's [Makhzani & Frey, ICLR ’14]
Q: What makes a good representation?

- Ability to reconstruct (→ prevalence of autoencoders)
- Robust to perturbations of the input
- Useful for downstream tasks (e.g., clustering, or classification)
- etc.

Common idea: Control (or enforce) properties of (or on) the latent representations in $\mathcal{Z}$.

Adversarial AE’s [Makhzani et al., ICLR ’16] (by far not exhaustive)

Encoder $f_\theta: \mathcal{X} \rightarrow \mathcal{Z}$

Latent space $\mathcal{Z}$

Decoder $g_\phi: \mathcal{Z} \rightarrow \mathcal{X}$

$x \rightarrow \hat{x} \rightarrow \text{Rec}[x, \hat{x}]$

Enforce distributional properties through **adversarial** training
Motivating (toy) example

We aim to control properties of the latent space, but from a **topological point of view**!
Motivating (toy) example

We aim to control properties of the latent space, but from a **topological point of view**!

Assume, we want to do **Kernel Density Estimation (KDE)** in the latent space $\mathcal{Z}$.

Data ($z_i$)  

Gaussian KDE

**Bandwidth selection**: Scott’s rule [Scott, 1992]
We aim to control properties of the latent space, but from a **topological point of view**!

Assume, we want to do **Kernel Density Estimation (KDE)** in the latent space $\mathcal{Z}$.

Bandwidth selection: Scott’s rule [Scott, 1992]
Controlling connectivity

Q: How do we capture topological properties and what do we want to control?
**Q:** How do we **capture** topological properties and what do we want to control?

**Vietoris Rips** Persistent Homology (PH)

Radius $r = r_1$

Latent space $\mathcal{Z}$
Q: How do we capture topological properties and what do we want to control?

**Vietoris Rips** Persistent Homology (PH)

![Diagram](image)

Radius \( r = r_2 \)

Latent space \( \mathcal{Z} \)
Q: How do we capture topological properties and what do we want to control?

Vietoris Rips Persistent Homology (PH)

- PH tracks topological changes as the ball radius $r$ increases.
- **Connectivity information** is captured by 0-dim. persistent homology.
Controlling connectivity

Q: How do we capture topological properties and what do we want to control?

Vietoris Rips Persistent Homology (PH)

- PH tracks topological changes as the ball radius $r$ increases
- Connectivity information is captured by 0-dim. persistent homology

Homogeneous arrangement!

What if $z \mapsto f_\theta(z)$

Beneficial for KDE
Q: How can we control topological properties (connectivity properties in particular)?
Q: How can we control topological properties (connectivity properties in particular)?

Consider batches $(x_1, \ldots, x_B)$

$$f_\theta : \mathcal{X} \to \mathbb{R}^n$$

$$g_\phi : \mathbb{R}^n \to \mathcal{X}$$

$\hat{x} \xrightarrow{\text{Rec}[\cdot, \cdot]} \text{Connectivity loss}$
Q: How can we control topological properties (connectivity properties in particular)?

Consider batches $(x_1, \ldots, x_B)$

$f_\theta : \mathcal{X} \rightarrow \mathbb{R}^n$

$g_\phi : \mathbb{R}^n \rightarrow \mathcal{X}$

$\hat{x}$

Rec[$\cdot$, $\cdot$] + Connectivity loss

penalize deviation from homogeneous arrangement (with scale $\eta$)
Q: How can we **control** topological properties (connectivity properties in particular)?

Consider **batches** \((x_1, \ldots, x_B)\)

\[
f_\theta : \mathcal{X} \to \mathbb{R}^n
\]

\[
g_\phi : \mathbb{R}^n \to \mathcal{X}
\]

\[
\hat{x} \xrightarrow{\text{Rec}[\cdot, \cdot]} + \text{Connectivity loss}
\]

penalize deviation from **homogeneous arrangement** (with scale \(\eta\))

\[
\mathcal{L}_\eta
\]
Q: How can we control topological properties (connectivity properties in particular)?

Consider batches $(x_1, \ldots, x_B)$

$f_\theta : \mathcal{X} \rightarrow \mathbb{R}^n$

$g_\phi : \mathbb{R}^n \rightarrow \mathcal{X}$

$\hat{x} \rightarrow \text{Rec}[, ,] + \text{Connectivity loss}$

Until now, we could not backpropagate through PH

penalize deviation from homogeneous arrangement (with scale $\eta$)

$L_\eta$
From a **theoretical perspective**, we show . . .

(1) . . . that under mild conditions, the **connectivity loss is differentiable**
From a **theoretical perspective**, we show . . .

(1) . . . that under mild conditions, the **connectivity loss is differentiable**

(2) . . . metric-entropy based guidelines for choosing the training batch size $B$
From a **theoretical perspective**, we show . . .

(1) . . . that under mild conditions, the **connectivity loss** is **differentiable**

(2) . . . metric-entropy based guidelines for choosing the training batch size $B$

(3) . . . “densification” effects occur for samples, $N$, larger than the training batch size $B$
From a **theoretical perspective**, we show ... 

\[ x_1, \ldots, x_N \rightarrow \text{Enc} \rightarrow \text{Dec} \rightarrow \cdots + \text{Connectivity loss} \]

(1) ... that under mild conditions, the **connectivity loss is differentiable**

(2) ... metric-entropy based guidelines for choosing the training batch size \( B \)

(3) ... “densification” effects occur for samples, \( N \), larger than the training batch size \( B \)

**Intuitively**, during training ... 

... the reconstruction loss controls **what** is worth capturing

... the **connectivity loss** controls **how** to topologically organize the latent space
Experiments – Task: One-class learning

Trained only once (e.g., on CIFAR-10 without labels)
**Experiments – Task:** One-class learning

**Diagram:**
- Auxiliary unlabeled data flows through $f_\theta$ and $g_\phi$.
- $g_\phi$ outputs a connectivity loss (with fixed scale $\eta$) plus reconstruction loss $\text{Rec}[\cdot, \cdot]$.
- The system is trained only once (e.g., on CIFAR-10 without labels).

KDE-inspired **one-class** "learning":
- One-class samples are processed by $f_\theta$.
- The output radius is $r = \eta/2$. 

**Equation:**
$$r = \eta/2$$
Experiments – Task: One-class learning

Trained only once (e.g., on CIFAR-10 without labels)

KDE-inspired one-class "learning"

Count #samples falling into balls of radius $\eta$, anchored at the one-class instances

One-class samples

Computations of a one-class score

In-class

Out-of-class
**Results – Task**: One-class learning

**CIFAR-10** (AE trained on CIFAR-100)

![Graph showing AUROC for various methods]

- **ADT** [Goland & El-Yaniv, NIPS ’18]
- **DAGMM** [Zong et al., ICLR ’18]
- **DSEBM** [Zhai et al., ICML ’16]
- **Deep-SVDD** [Ruff et al., ICML ’18]

Training batch size: $B = 100$
Results – Task: One-class learning

### CIFAR-10 (AE trained on CIFAR-100)

<table>
<thead>
<tr>
<th>Method</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSEBM</td>
<td>0.5</td>
</tr>
<tr>
<td>OC-SVM (CAE)</td>
<td>0.6</td>
</tr>
<tr>
<td>Deep-SVDD</td>
<td>0.7</td>
</tr>
<tr>
<td>ADT</td>
<td>0.8</td>
</tr>
<tr>
<td>Ours-120</td>
<td>0.8+7</td>
</tr>
<tr>
<td>ADT-100</td>
<td>0.7</td>
</tr>
<tr>
<td>ADT-500</td>
<td>0.6</td>
</tr>
<tr>
<td>ADT-120</td>
<td>0.5</td>
</tr>
<tr>
<td>Ours-120</td>
<td>0.7</td>
</tr>
</tbody>
</table>

**Low-sample size**

ADT [Goland & El-Yaniv, NIPS ‘18]
DAGMM [Zong et al., ICLR ‘18]
DSEBM [Zhai et al., ICML ‘16]
Deep-SVDD [Ruff et al., ICML ‘18]

Training batch size: $B = 100$
**Results – Task:** One-class learning

**CIFAR-20** (AE trained on CIFAR-10)

- **ADT** [Goland & El-Yaniv, NIPS ’18]
- **DAGMM** [Zong et al., ICLR ’18]
- **DSEBM** [Zhai et al., ICML ’16]
- **Deep-SVDD** [Ruff et al., ICML ’18]

Training batch size: $B = 100$
Results – Task: One-class learning

CIFAR-20 (AE trained on CIFAR-10)

ADT [Goland & El-Yaniv, NIPS ’18]
DAGMM [Zong et al., ICLR ’18]
DSEBM [Zhai et al., ICML ’16]
Deep-SVDD [Ruff et al., ICML ’18]

Training batch size: $B = 100$
Results – Task: One-class learning

CIFAR-100 (AE trained on CIFAR-10)

- ADT [Goland & El-Yaniv, NIPS ’18]
- DAGMM [Zong et al., ICLR ’18]
- DSEBM [Zhai et al., ICML ’16]
- Deep-SVDD [Ruff et al., ICML ’18]

Training batch size: \( B = 100 \)
**Results – Task:** One-class learning

**ImageNet** (i.e., evaluation of 1,000 one-class models)

![Graph showing AUROC for different models](image)

- **ADT** [Goland & El-Yaniv, NIPS ’18]
- **DAGMM** [Zong et al., ICLR ’18]
- **DSEBM** [Zhai et al., ICML ’16]
- **Deep-SVDD** [Ruff et al., ICML ’18]

Using **one** AE trained on CIFAR-10

Using **one** AE trained on CIFAR-100

Training batch size: \( B = 100 \)
Come see our poster
#83
at 6.30pm (Pacific Ballroom)

```
import torch
import chofer_torchex.pershom as pershom

batch = torch.randn(10,5, requires_grad=True)
batch = batch.to('cuda')

non_ess, ess = pershom.vr_persistence_l1(batch,0,0)

example_loss = non_ess[:,1].sum()
example_loss.backward()
```

https://github.com/c-hofer/COREL_icml2019