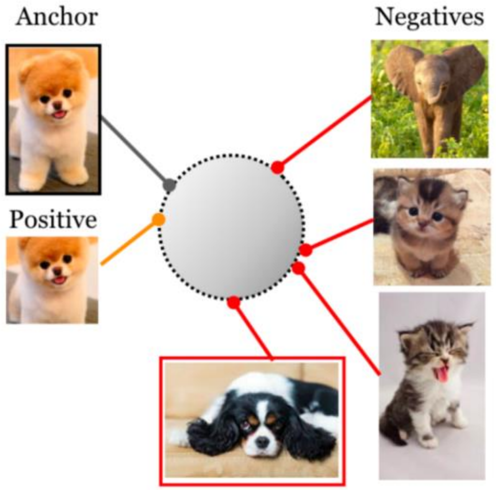
Rethinking Self-Supervised Learning From the Perspective of Distance Preserving

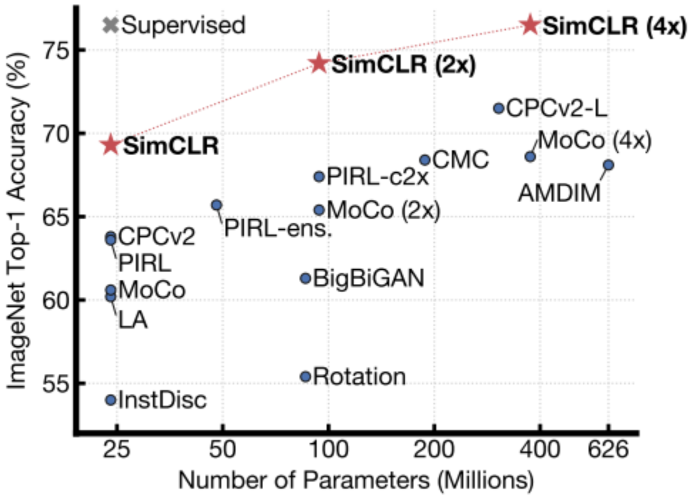
Tianyang Hu
Noah's Ark Lab

Pretrained Models Are Powerful

Image classification



Self Supervised Contrastive (from Google's [blog](#))

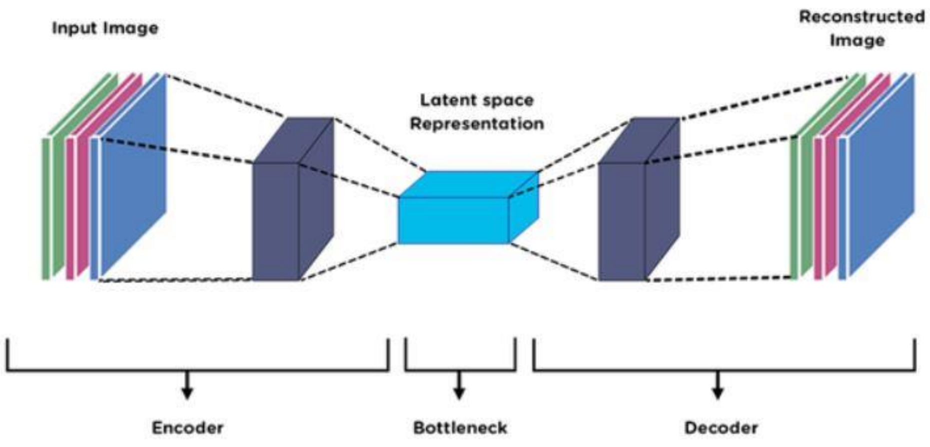


(from the SimCLR paper by Chen et al.)

Method	PACS	VLCS	OfficeHome
ERM†	85.5	77.5	66.5
IRM†	83.5	78.6	64.3
GroupDRO†	84.4	76.7	66.0
I-Mixup†	84.6	77.4	68.1
MMD†	84.7	77.5	66.4
SagNet†	86.3	77.8	68.1
ARM†	85.1	77.6	64.8
VREx†	84.9	78.3	66.4
RSC†	85.2	77.1	65.5
SWAD	88.1	79.1	70.6
			ZooD
Single*	96.0	79.5	84.6
Ensemble*	95.5	80.1	85.0
F. Selection*	96.3	80.6	85.1

(from our model zoo [paper](#))

Latent space generative modeling



Autoregressive

- **VQGAN** (2021)
- [Parti](#) by Google (2022)
- [CM3leon](#) by Meta (2023)

Diffusion

- [DALL·E 2](#) by OpenAI (2022)
- [Stable Diffusion](#) (2021)
- DiT (2022)

Characterizing Features From Self-Supervised Learning

Theoretical understanding of SSL is still lacking.

- What are the learned features?
- How does it depend on the (augmented) data?
- Why is the feature useful for downstream tasks?

Core: preserving **distributions** in **different dimensions**

How to measure the **closeness** between p_z and p_x?

Consider sample size 100 and $x \in R^{10}$ and $z \in R^2$.

- Gromov-Wasserstein distance [1]: to pairwise

$$GW_p(c_{\mathcal{X}}, c_{\mathcal{Y}}, \mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}^2 \times \mathcal{Y}^2} |c_{\mathcal{X}}(x,x') - c_{\mathcal{Y}}(y,y')|^p d\pi(x,y) d\pi(x',y') \right)^{\frac{1}{p}},$$

- Projection/embedding to same dimension [2]

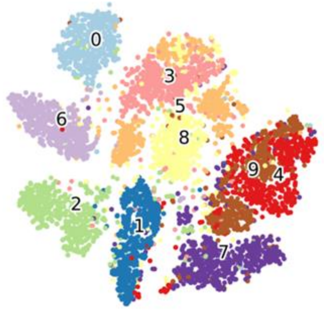
$$D^+(P_x, P_z) := \inf_{P_{\widehat{x}} \in \Phi^+(P_z, d)} D(P_x, P_{\widehat{x}}),$$

"Distance" btw distributions in different dimensions

By 2-pair joint density $p(x, x')$ vs. $p(z, z')$

Contrastive learning (CL)

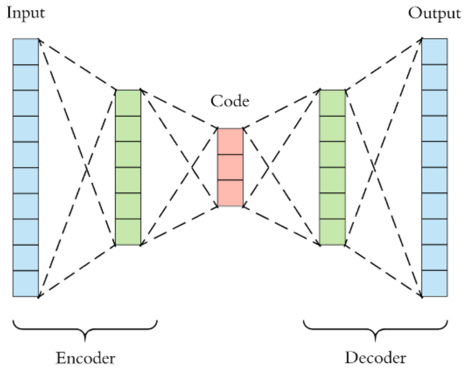
Stochastic Neighbor Embedding (SNE)



By projection/embedding $p(x)$ vs. $p(g(z))$

Encoder-Decoder

GAN



[1] Memoli. *Gromov–Wasserstein distances and the metric approach to object matching*. Foundations of Computational Mathematics, 2011
[2] Cai and Lim. *Distances between probability distributions of different dimensions*. IEEE Transactions on Information Theory, 2020
[3] Hu, T., Liu, Z., Zhou, F., Wang, W., & Huang, W., *Your Contrastive Learning Is Secretly Doing Stochastic Neighbor Embedding*. ICLR 2023
[4] Hu, T., Chen, F., Wang, H., Li, J., Wang, W., Sun, J., & Li, Z., *Complexity Matters: Rethinking the Latent Space for Generative Modeling*. arXiv preprint 2307.08283

Characterizing Features From Self-Supervised Learning

Core: preserving **distributions** in **different dimensions**

For Contrastive Learning [3]:

- The learning process is matching the pairwise joint distribution
- Data augmentation in SSCL specifies the pairwise similarity
- Insights from SNE can be used to improve SSCL, both **dimensional efficiency** and **out-of-distribution generalization**

For generative modeling in latent space [4]:

- We characterize the **optimal latent distribution** from the perspective of **distribution matching**
- **Complexity matters**: optimality in terms of minimizing the required complexity.
- Decoupled AutoEncoder (DAE) training improves the latent distribution, resulting in **better performance with lower complexity**

"Distance" btw distributions in different dimensions

By 2-pair joint density $p(x, x')$ **vs.** $p(z, z')$

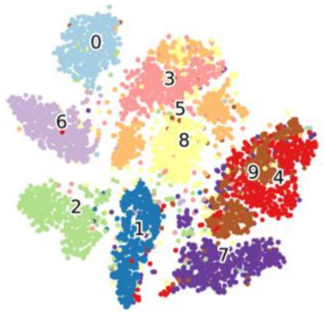
By projection/embedding $p(x)$ **vs.** $p(g(z))$

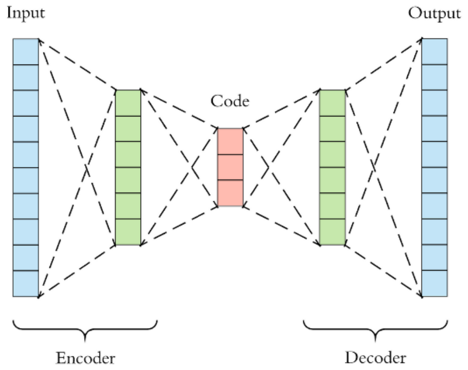
Contrastive learning (CL)

Encoder-Decoder

Stochastic Neighbor Embedding (SNE)

GAN





[3] Hu, T., Liu, Z., Zhou, F., Wang, W., & Huang, W., *Your Contrastive Learning Is Secretly Doing Stochastic Neighbor Embedding*. ICLR 2023

[4] Hu, T., Chen, F., Wang, H., Li, J., Wang, W., Sun, J., & Li, Z., *Complexity Matters: Rethinking the Latent Space for Generative Modeling*. arXiv preprint 2307.08283

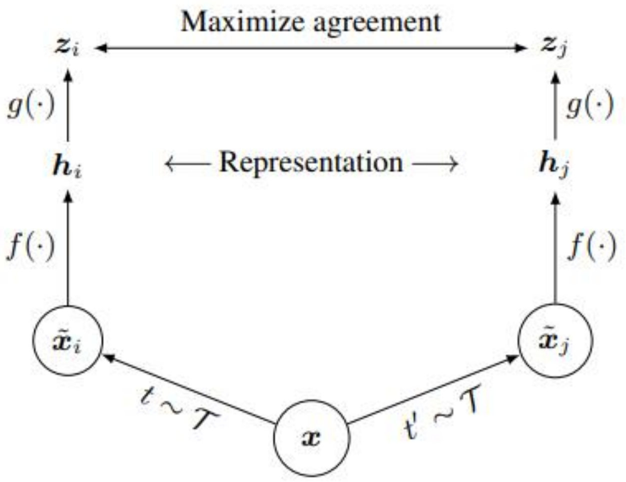
Background — Contrastive learning

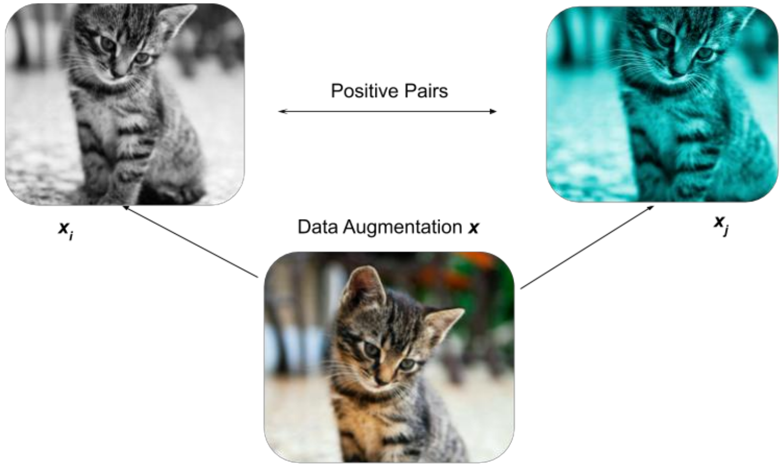
Self-supervised contrastive learning (SSCL) has drawn massive attention recently with many SoTA models following this paradigm in both CV and NLP.

Key steps:

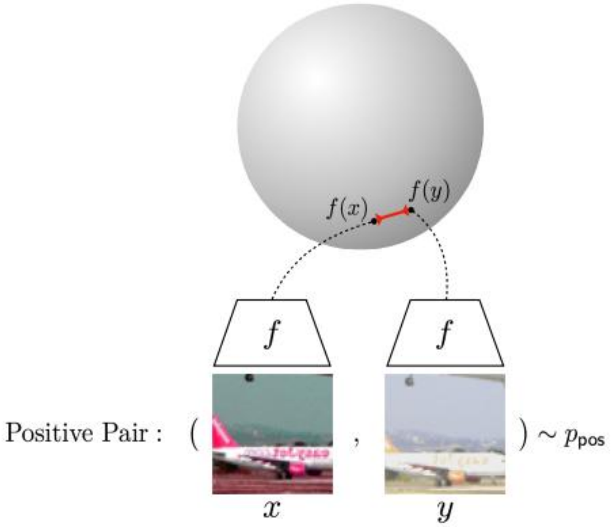
1. Data augmentation
2. Learning feature mapping z = f(x) by the training objective. The most Typical loss is the InfoNCE (in SimCLR, MoCo, CLIP, etc.):

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

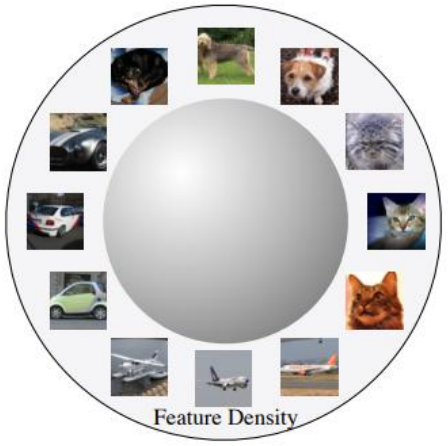




Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *ICML* 2020.



Alignment: Similar samples have similar features. (Figure inspired by Tian et al. (2019).)



Uniformity: Preserve maximal information.

Wang, T., & Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *ICML 2020*.

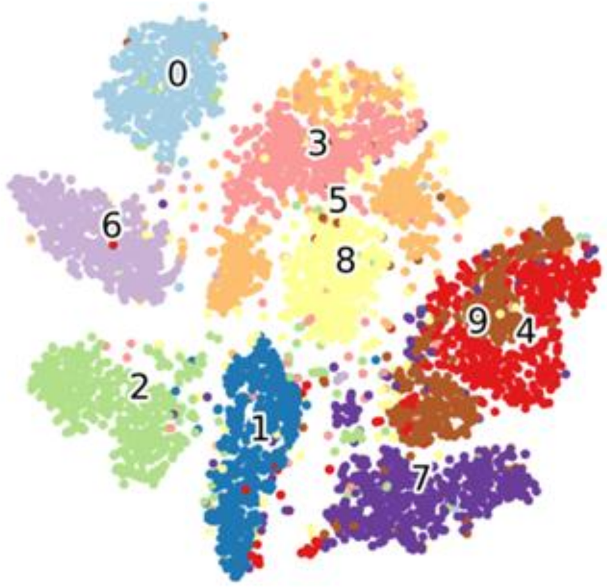
Background — Stochastic neighbor embedding

SNE is a popular method for visualizing high-dimensional data in 2D.
Given $x_1, \ldots, x_n$, the goal of SNE is to find $z_1, \ldots, z_n$ that **preserves** as much as **neighboring information** as possible.

Q1: How is neighboring info **modeled**?

A1: By using (conditional) **Gaussian** likelihood. We have P and Q.

$$P_{j|i} = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|_2^2 / 2\sigma_i^2)},$$



Q2: How is the neighboring info **preserved**?

A2: By minimizing the **KL-divergence.** Matching Q to P.

$$\inf_{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_n} \sum_{i=1}^{n} \sum_{j=1}^{n} P_{j|i} \log \frac{P_{j|i}}{Q_{j|i}}.$$

Van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing data using t-SNE." *Journal of machine learning research* 9.11 (2008).

Contrastive learning vs Stochastic Neighbor Embedding

The goal of SSCL, learning feature representations from unlabeled data, coincides with that of the classic method --- **Stochastic Neighbor Embedding (SNE).**

	SSCL	SNE
Empirical	Superb performance for CV and NLP tasks. Widely adopted for pre-training, with many OOD downstream tasks.	Does not work well for over-complicated data, e.g., CIFAR-10.
Theoretical understanding	Under-explored as to how the learned features depend on data and different components of the SSCL methods.	Far better understanding with theoretical guarantees

We would like to ask:

- Both trying to learn feature representations, are there any deep connections between SSCL and SNE?
- Can SSCL take the advantage of the theoretical soundness of SNE?
- Can SNE be revived in the modern era by incorporating SSCL?

Contrastive learning vs Stochastic Neighbor Embedding

The key observation is that SSCL can be viewed as **a special case of SNE**

	SNE	SSCL -- SimCLR
P: Pairwise similarity in the input space	By Gaussian distribution $P_{j\mid i}=\frac{\exp(-d(\boldsymbol{x}_i,\boldsymbol{x}_j))}{\sum_{k\neq i}\exp(-d(\boldsymbol{x}_i,\boldsymbol{x}_k))}$ $d(\cdot,\cdot)$ is usually ℓ_2 distance	$P_{j\mid i}=\begin{cases}\frac{1}{2n}, & \text{if } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are positive pairs}\\ 0, & \text{otherwise,}\end{cases}$ Only similarity between constructed positive pairs are nonzero
Q: Pairwise similarity in the feature space	By Gaussian distribution $Q_{j\mid i}=\frac{\exp(-d(f(\boldsymbol{x}_i),f(\boldsymbol{x}_j)))}{\sum_{k\neq i}\exp(-d(f(\boldsymbol{x}_i),f(\boldsymbol{x}_k)))}$ $d(\cdot,\cdot)$ is usually ℓ_2 distance	By Gaussian distribution $Q_{j\mid i}=\frac{\exp(\mathrm{sim}(f(\boldsymbol{x}_i),f(\boldsymbol{x}_j))}{\sum_{k\neq i}\exp(\mathrm{sim}(f(\boldsymbol{x}_i),f(\boldsymbol{x}_k))}$ $\mathrm{sim}(\cdot,\cdot)$ is usually cosine similarity
Divergence when matching **P** to **Q**	KL-divergence $\inf_{z_1,\cdots,z_n}\sum_{i=1}^{n}\sum_{j=1}^{n}P_{j\mid i}\log\frac{P_{j\mid i}}{Q_{j\mid i}}$	the same KL-divergence $-\log\frac{\exp(\mathrm{sim}(f(\boldsymbol{x}_i),f(\boldsymbol{x}_i'))/\tau)}{\sum_{\boldsymbol{x}\in\mathcal{D}_n\cup\mathcal{D}_n'\setminus\{\boldsymbol{x}_i\}}\exp(\mathrm{sim}(f(\boldsymbol{x}_i),f(\boldsymbol{x}))/\tau)}$

The objective of SimCLR mainly differs from the standard SNE in how **P** is specified.

SNE perspective of SSCL

The objective of SimCLR mainly differs from the standard SNE in how P is specified.

Thus, the feature learning process of SSCL can also be summarized as

(S1) The positive pair construction specifies the similarity matrix $\boldsymbol{P}$.

(S2) The training process then matches $\boldsymbol{Q}$ to $\boldsymbol{P}$ by minimizing some divergence between the two specified by the training objective, e.g., KL divergence in SimCLR.

- The main difference between SNE and SSCL is the first part, where the P in SNE is usually densely filled by l_p-distance, ignoring the semantic information within rich data like images and texts.
- SSCL omits all traditional distances for vectors and only specifies semantic similarity through data augmentation, and the resulting P is sparsely filled only by positive pairs.

What are the specified distance by data augmentation?

We answer the question in part by considering domain-agnostic data augmentation, by random noise injection.

Proposition 3.2 (Gaussian noise injection)**.** If the noise distribution is isotropic Gaussian, the induced distance is *equivalent* to the l_2 distance in $\mathbb{R}^d$, up to a monotone transformation.

SNE perspective of SSCL --- Practical guidance

t-SNE Style Matching:

t-SNE has significant improvement over SNE, with the main differences:

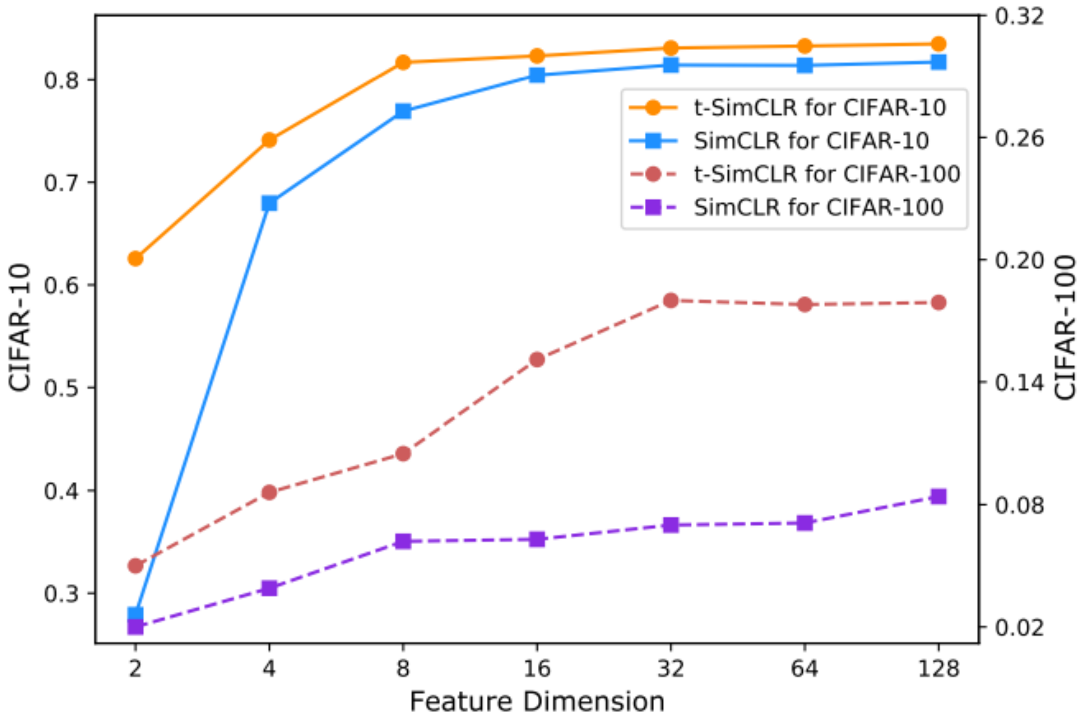
- Conditional to joint distribution
- Gaussian distribution to t-distribution, to avoid the "**crowding problem**"

The same advantage from SNE to t-SNE, can realized in SimCLR → t-SimCLR

$$\frac{1}{n}\sum_{i=1}^{n}-\log\frac{\left(1+\|f(\boldsymbol{x}_i)-f(\boldsymbol{x}_i')\|_2^2/(\tau\, t_{df})\right)^{-(t_{df}+1)/2}}{\sum_{1\le j\ne k\le 2n}\left(1+\|f(\widetilde{\boldsymbol{x}}_j)-f(\widetilde{\boldsymbol{x}}_k)\|_2^2/(\tau\, t_{df})\right)^{-(t_{df}+1)/2}},$$

Advantages:

- **Better dimensional efficiency**
- **Better OOD generalization**



CIFAR-10 training, 200 epoch, nearest neighbor accuracy

SNE perspective of SSCL --- Practical guidance

t-SNE Style Matching:

Advantages:
- **Better dimensional efficiency**
- **Better OOD generalization**

$$\frac{1}{n}\sum_{i=1}^{n}-\log\frac{\left(1+\|f(\boldsymbol{x}_i)-f(\boldsymbol{x}_i')\|_2^2/(\tau\, t_{df})\right)^{-(t_{df}+1)/2}}{\sum_{1\leq j\neq k\leq 2n}\left(1+\|f(\widetilde{\boldsymbol{x}}_j)-f(\widetilde{\boldsymbol{x}}_k)\|_2^2/(\tau\, t_{df})\right)^{-(t_{df}+1)/2}},$$

Larger scale experiments: ImageNet to OOD tasks

Table 1: Domain transfer results of vanilla MoCo-v2 and t-MoCo-v2.

Method	Aircraft	Birdsnap	Caltech101	Cars	CIFAR10	CIFAR100	DTD	Pets	SUN397	Avg.
MoCo-v2	82.75	44.53	83.31	85.24	95.81	72.75	**71.22**	86.70	56.05	75.37
t-MoCo-v2	**82.78**	**53.46**	**86.81**	**86.17**	**96.04**	**78.32**	69.20	**87.95**	**59.30**	**77.78**

Table 2: OOD accuracies of vanilla MoCo-v2 and t-MoCo-v2 on domain generalization benchmarks.

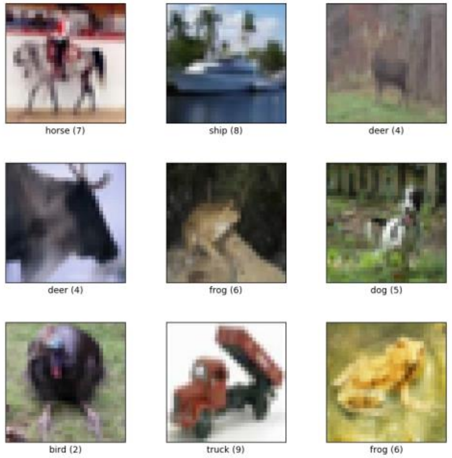
Method	PACS	VLCS	Office-Home	Avg.
MoCo-v2	58.5	70.4	36.6	55.2
t-MoCo-v2	**61.3**	**75.1**	**42.1**	**59.5**

SNE perspective of SSCL --- Practical guidance

SSCL revive t-SNE:

(S1) The positive pair construction specifies the similarity matrix $\boldsymbol{P}$.

(S2) The training process then matches $\boldsymbol{Q}$ to $\boldsymbol{P}$ by minimizing some divergence between the two specified by the training objective, e.g., KL divergence in SimCLR.



Utilizing data augmentation to specify better distance

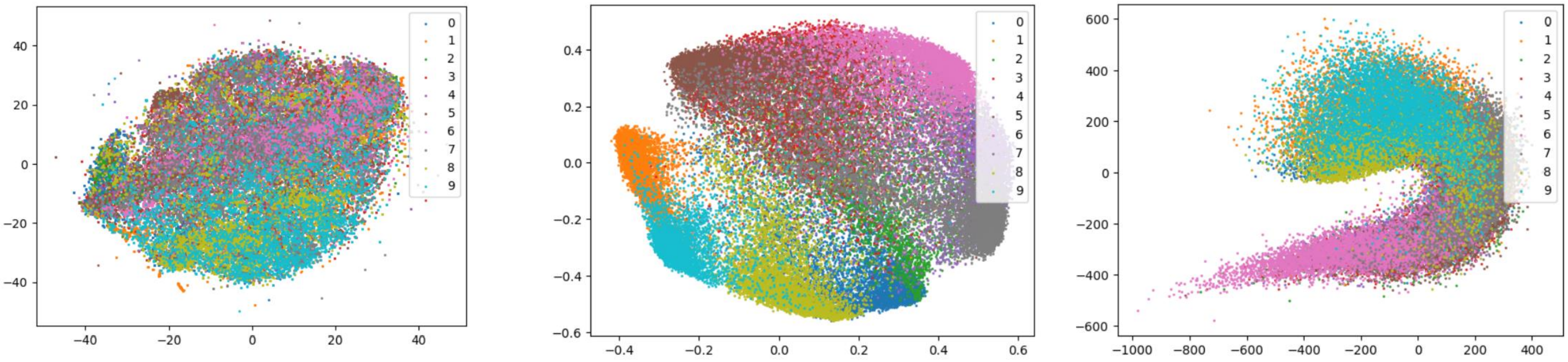


Figure B.10: 50K CIFAR-10 training images visualization in 2D with t-**SNE**.

Characterizing Features From Self-Supervised Learning

Core: preserving **distributions** in **different dimensions**

For Contrastive Learning [3]:

- The learning process is matching the pairwise joint distribution
- Data augmentation in SSCL specifies the pairwise similarity
- Insights from SNE can be used to improve SSCL, both **dimensional efficiency** and **out-of-distribution generalization**

For generative modeling in latent space [4]:

- We characterize the **optimal latent distribution** from the perspective of **distribution matching**
- **Complexity matters**: optimality in terms of minimizing the required complexity.
- Decoupled AutoEncoder (DAE) training improves the latent distribution, resulting in **better performance with lower complexity**

"Distance" btw distributions in different dimensions

By 2-pair joint density
$p(x, x')$ **vs.** $p(z, z')$

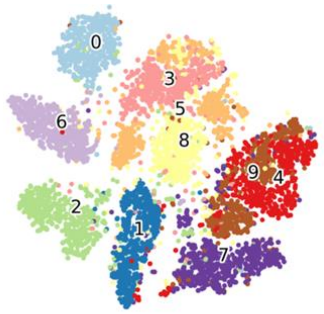
By projection/embedding
$p(x)$ **vs.** $p(g(z))$

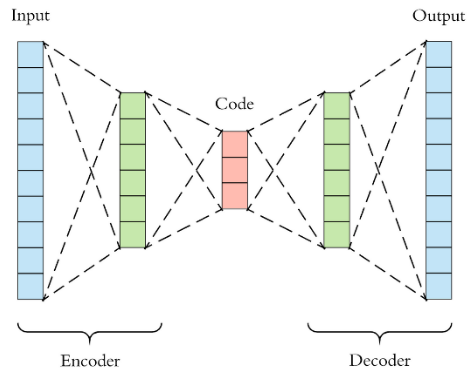
Contrastive learning (CL)

Encoder-Decoder

Stochastic Neighbor Embedding (SNE)

GAN





[3] Hu et al. *Your contrastive learning is secretly doing stochastic neighbor embedding*. ICLR 2023
[4] Hu et al. *Complexity Matters: Rethinking the Latent Space for Generative Modeling*.

An Ideal Latent Distribution for GAN

GAN generator aims to learn $\inf_{g \in \mathcal{G}} D_h(P_x, P_{g(z)}),\ \boldsymbol{z} \sim P_z,$

- Latent distribution P_z is usually **predefined and data-agnostic.**
- Different choice of P_z has great impact of the performance
- Many drawbacks of GAN can be traced back to the **mismatch** between P_z and P_x

How to define an ideal P_z ? Complexity is the key:

- If G has unlimited capacity, $P_{g(z)}$ and P_x can be arbitrarily close
- With limited capacity of G, the training loss of GAN can serve as a **relative measurement** of how good a latent is.

$$D^{\mathcal{G}}(P_z, P_x) := \inf_{g \in \mathcal{G}} D(P_{g(z)}, P_x).$$

- The above serves as a "distance" between P_z and P_x, which is a **generalized case** of [Cai & Lim 2020]
- The optimal latent P_z^* can be defined as the minimizer.

How to find the optimal P_z^* ?

Parametrization

- Using an encoder $P_z = P_{f(x)}$
- The above can be seen as a new type of **self-supervised learning problem** $f^* = \operatorname{argmin}_{f \in \mathcal{F}} D^{\mathcal{G}}(P_x, P_{f(x)}).$
- Will existing contrastive learning work?

Method	IS (↑)	FID (↓)
DCGAN (reproduced)	5.68	51.76
DCGAN-SimCLR	3.93	168.23
VAEGAN	5.82	48.11

Optimization

- GAN training with f suffice!

$$\inf_{f \in \mathcal{F}, g \in \mathcal{G}} D(P_x, P_{g \circ f(\boldsymbol{x})}) = \inf_{f \in \mathcal{F}} \left(\inf_{g \in \mathcal{G}} D(P_x, P_{g \circ f(\boldsymbol{x})}) \right)$$
$$= \inf_{f \in \mathcal{F}} D^{\mathcal{G}}(P_x, P_{f(\boldsymbol{x})}).$$

- VQGAN is already doing it!
- The balance between F and G is critical

Informativeness of Latent

Quality of Reconstruction

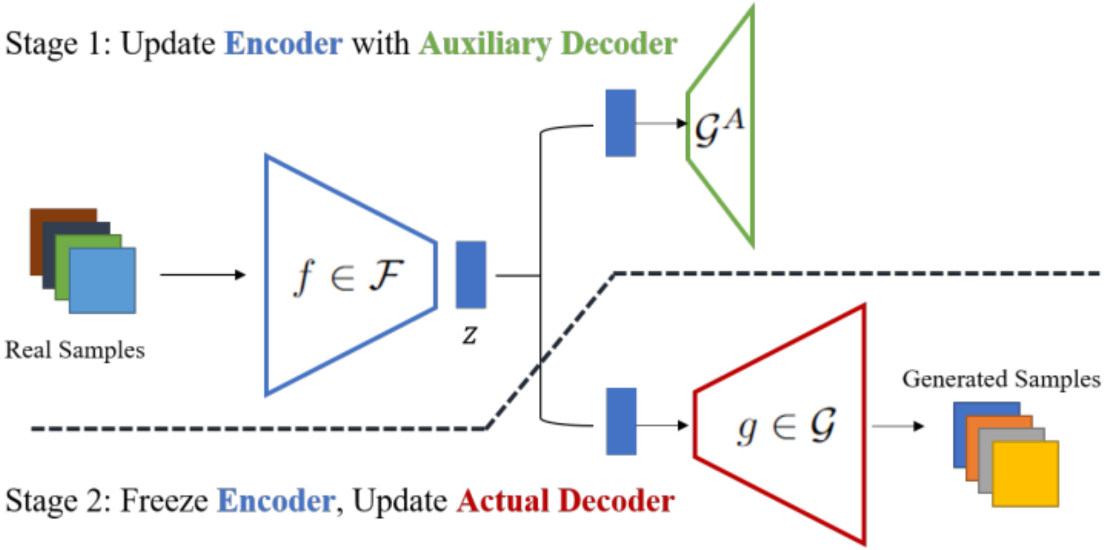
DAE: Balancing Encoder vs Decoder

Let $C(\cdot)$ be some general complexity measurement.

For Reconstruction Quality:
- Intuitively, $C(f) = C(g)$ seems the best
- Both encoder and decoder should be as powerful as possible

For Latent Informativeness:
- The decoder should be relatively weaker.

To address the tradeoff:



DAE enjoys the best of both worlds!

Better Reconstruction & Latent Generative Modeling
- DCGAN on CIFAR-10
- VQGAN on FFHQ and CelabaHQ 256*256
- Diffusion Transformer (DiT) on ImageNet 256*256

Table 2: The performance of DCGAN with different latents.

Method	IS (↑)	FID (↓)
DCGAN (reproduced)	5.68	51.76
DCGAN-SimCLR	3.93	168.23
VAEGAN	5.82	48.11
DAE-VAEGAN	**6.16**	**46.12**

Table 3: Reconstruction FID over FacesHQ training and validation sets, and transformer generation FID on CelebAHQ and FFHQ training sets. †: Evaluated on the publicly available pre-trained model on FacesHQ. *: Our reproduction is based on the official VQGAN implementation.

Method	Reconstruction		Generation	
	Train	Val	CelebaHQ	FFHQ
VQGAN	4.81†	6.27†	10.2	9.6
VQGAN*	4.23	5.83	9.97	10.44
DAE-VQGAN	**2.01**	**3.82**	**8.58**	**8.36**

DAE: Balancing Encoder vs Decoder

Let $C(\cdot)$ be some general complexity measurement.

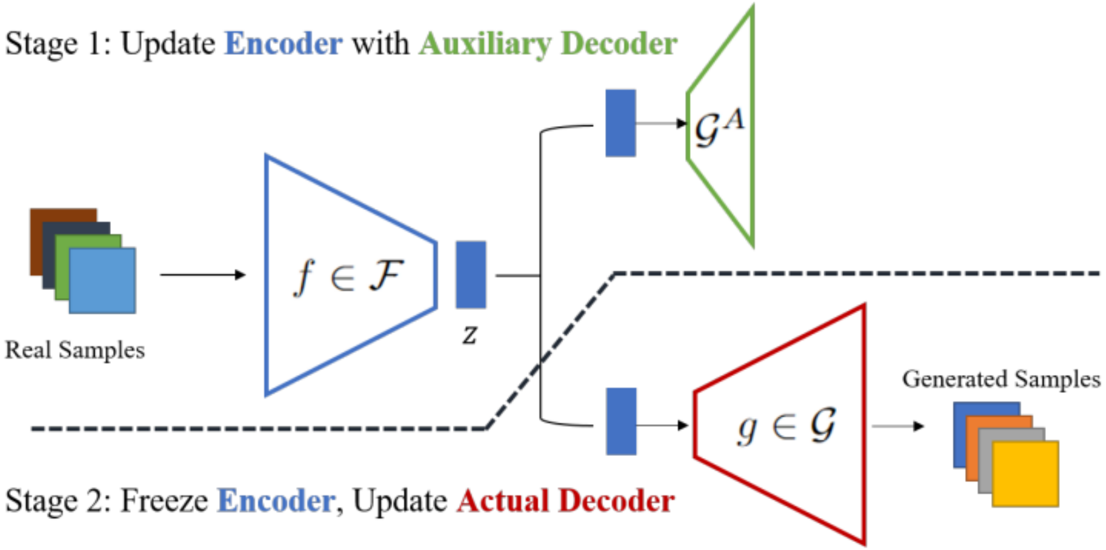
For Reconstruction Quality:

- Intuitively, $C(f) = C(g)$ seems the best
- Both encoder and decoder should be as powerful as possible

For Latent Informativeness:

- The decoder should be relatively weaker.

To address the tradeoff:



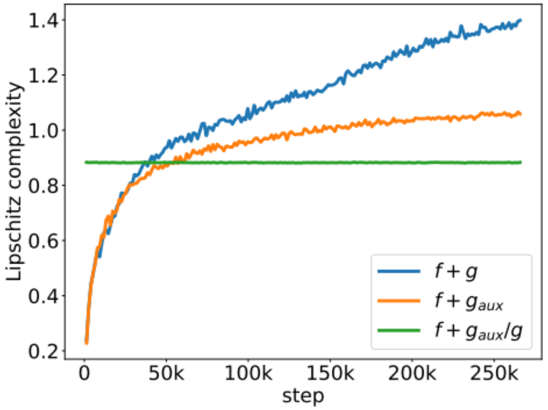
DAE enjoys the best of both worlds!

Better Reconstruction & Latent Generative Modeling

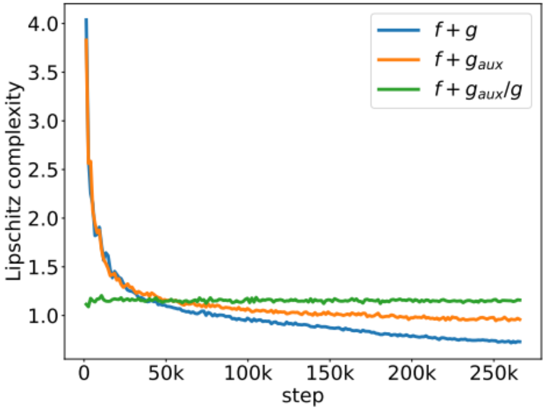
- DCGAN on CIFAR-10
- VQGAN on FFHQ and CelabaHQ 256*256
- Diffusion Transformer (DiT) on ImageNet 256*256

Decreased Complexity

- The Lipchitz constant is closer to 1.



(a) Encoder



(b) Decoder

DAE: Balancing Encoder vs Decoder

Let $C(\cdot)$ be some general complexity measurement.

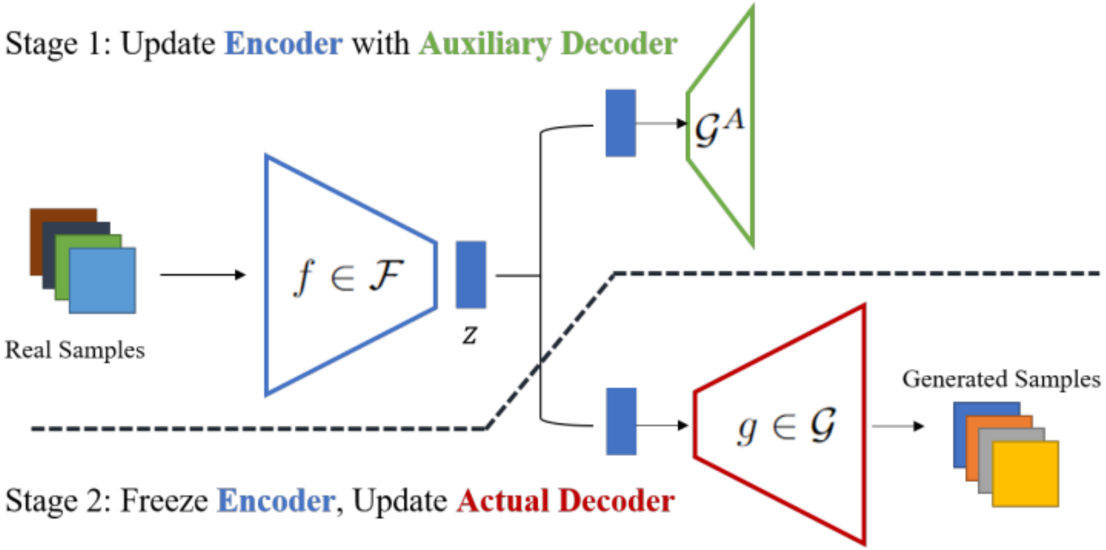
For Reconstruction Quality:

- Intuitively, $C(f) = C(g)$ seems the best
- Both encoder and decoder should be as powerful as possible

For Latent Informativeness:

- The decoder should be relatively weaker.

To address the tradeoff:



DAE enjoys the best of both worlds!

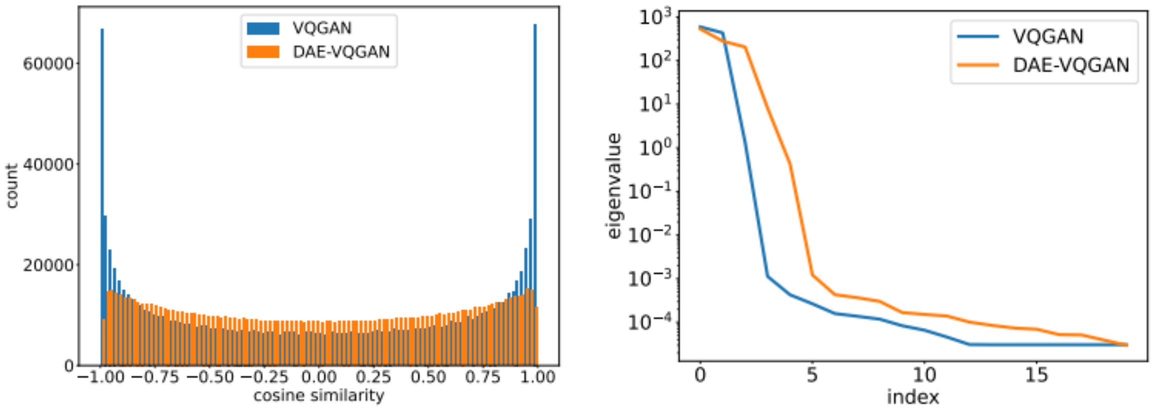
Better Reconstruction & Latent Generative Modeling

- DCGAN on CIFAR-10
- VQGAN on FFHQ and CelabaHQ 256*256
- Diffusion Transformer (DiT) on ImageNet 256*256

Decreased Complexity

- The Lipchitz constant is closer to 1.

Less collapsed latent:



(a) (b)

Summary

Core: preserving **distributions** in **different dimensions**

For Contrastive Learning [3]:

- The learning process is matching the pairwise joint distribution
- Data augmentation in SSCL specifies the pairwise similarity
- Insights from SNE can be used to improve SSCL, both **dimensional efficiency** and **out-of-distribution generalization**

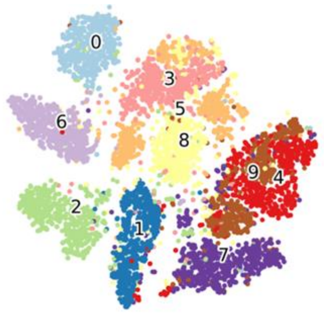
For generative modeling in latent space [4]:

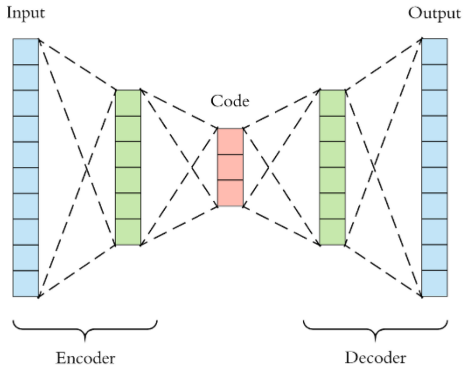
- We characterize the **optimal latent distribution** from the perspective of **distribution matching**
- **Complexity matters**: optimality in terms of minimizing the required complexity.
- Decoupled AutoEncoder (DAE) training improves the latent distribution, resulting in **better performance with lower complexity**

"Distance" btw distributions in different dimensions

By 2-pair joint density $p(x,x')$ **vs.** $p(z,z')$ ↔ **Contrastive learning (CL)** ↔ **Stochastic Neighbor Embedding (SNE)**

By projection/embedding $p(x)$ **vs.** $p(g(z))$ ↔ **Encoder-Decoder** ↔ **GAN**





[3] Hu, T., Liu, Z., Zhou, F., Wang, W., & Huang, W., *Your Contrastive Learning Is Secretly Doing Stochastic Neighbor Embedding*. ICLR 2023
[4] Hu, T., Chen, F., Wang, H., Li, J., Wang, W., Sun, J., & Li, Z., *Complexity Matters: Rethinking the Latent Space for Generative Modeling*. arXiv preprint 2307.08283