ICASSP 2024 Tutorial T-10: Building White-Box Deep Neural Networks

Lecture III: White-Box Autoencoding via Structured Denoising-Diffusion

Sam Buchanan, TTIC

Druv Pai, UC Berkeley

April 14, 2024

Lecture III Summary

- Today so far:
  1. Unrolled optimization and foundations of white-box models
  2. The model problem of compressive autoencoding, and an objective which promotes compressive representations
  3. A white-box transformer encoder which performs compressive encoding at scale

- Now:
  4. A white-box transformer *autoencoder* for compressive autoencoding, built using connections between classical signal processing primitives and tasks
  5. Beyond autoencoding for representation learning; principled self-consistent closed-loop learning systems

Outline

Recall: Compressive Autoencoding

Abstractly: seek maps $f : \mathbb{R}^D \to \mathbb{R}^d$, $g : \mathbb{R}^d \to \mathbb{R}^D$, such that

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x})} \boldsymbol{z} \in \mathbb{R}^d \xrightarrow{g(\boldsymbol{z})} \widehat{\boldsymbol{x}} \in \mathbb{R}^D.$$

satisfies $\boldsymbol{x} \approx \widehat{\boldsymbol{x}} = g \circ f(\boldsymbol{x})$ in a suitable sense; and $\boldsymbol{z}$ is

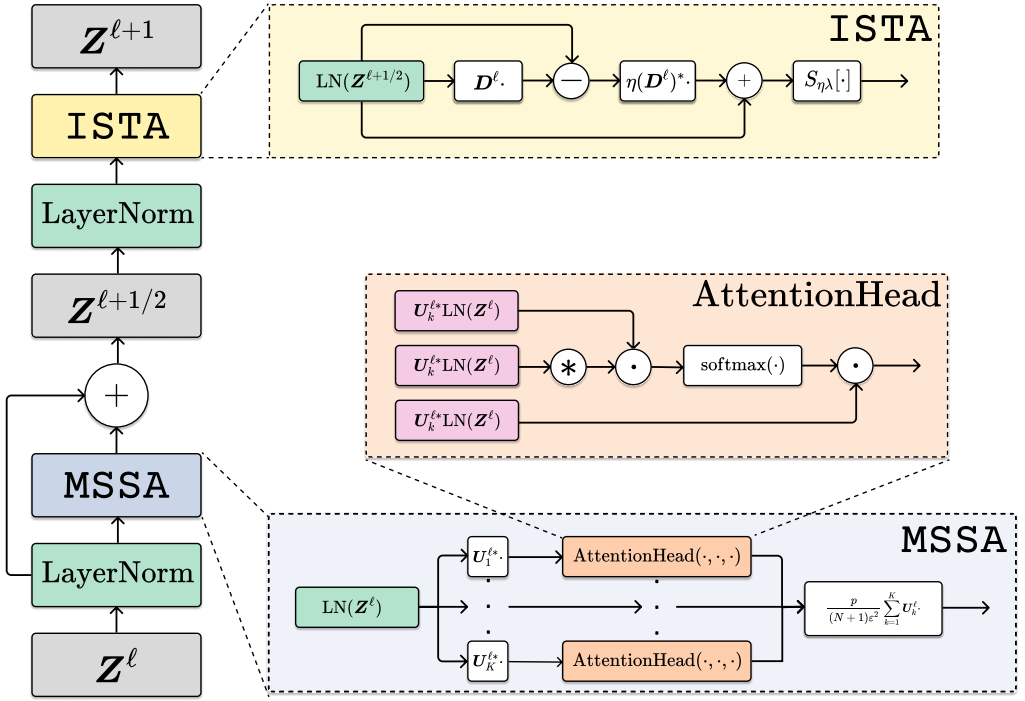
1. ***Compressed***, preserving only intrinsic low-dimensionality of $\boldsymbol{x}$;
2. ***Linearized***, having (piecewise) linear geometry, to aid interpolation and extrapolation in the representation space;
3. ***Sparse***: distinct modes of variability in $\boldsymbol{x}$ are geometrically *incoherent* and moreover *axis-aligned*.

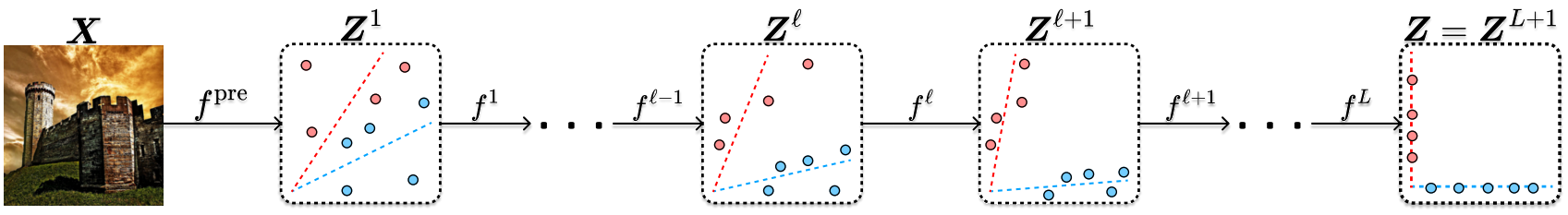
Goal: Build a decoder where we have the same level of layer-wise understanding as the encoder.

White-Box Autoencoder

We understand *encoder* via unrolled optimization and compression.

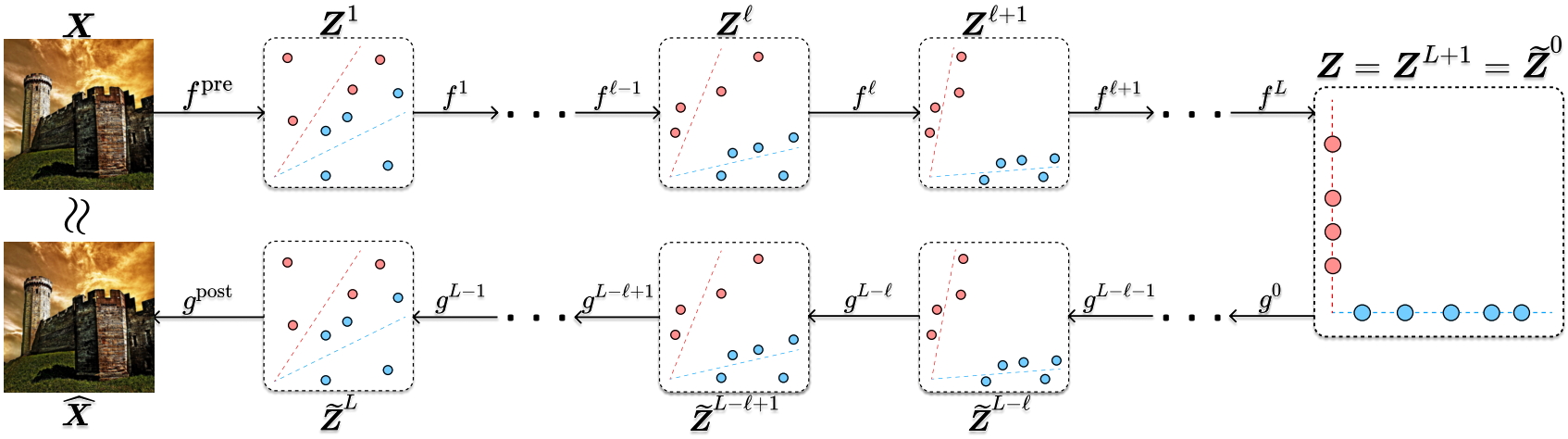
What about the *decoder*?





Unrolled Optimization for Building White-Box Decoder?

Decoder **cannot** optimize representation learning objective.

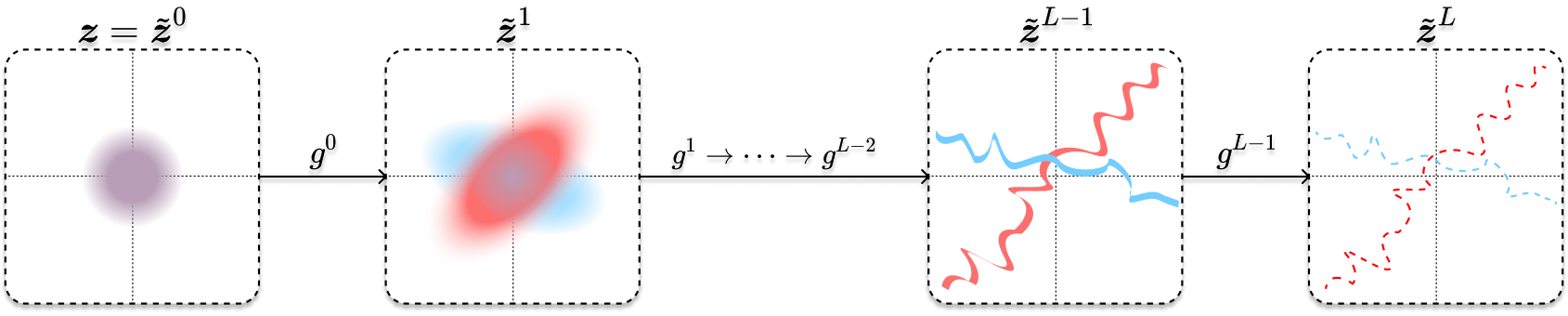


Want *reversible process* which converts *noisy, nonlinear* data to *compact, structured* representations (and back).

Diffusion for Building White-Box Decoder?

Diffusion process: scalable, (distribution-)invertible way to convert:

random noise $\longleftrightarrow$ low-dim. data distribution

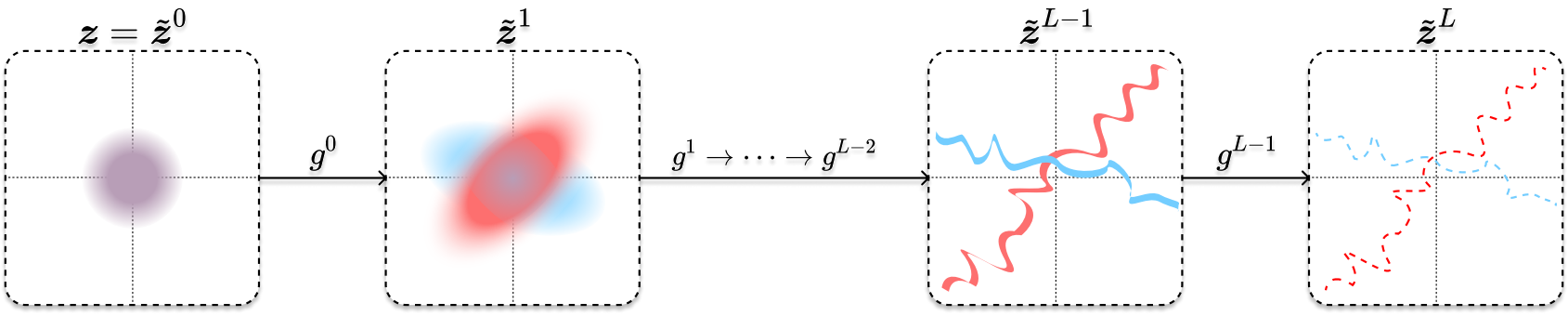


$$\tilde{\boldsymbol{z}}^{\ell+1} = g^\ell(\tilde{\boldsymbol{z}}^\ell) = \tilde{\boldsymbol{z}}^\ell + \tau^\ell \nabla \log q^\ell(\tilde{\boldsymbol{z}}^\ell)$$

Tweedie's formula [Robbins 1956]: If $\boldsymbol{z}^\ell = \boldsymbol{z}_\natural^\ell + (\sigma^\ell)^2 \boldsymbol{g}^\ell$, then

$$\mathbb{E}[\boldsymbol{z}_\natural^\ell \mid \boldsymbol{z}^\ell] = \boldsymbol{z}^\ell + (\sigma^\ell)^2 \nabla \log q^\ell(\boldsymbol{z}^\ell),$$

where $q^\ell \doteq \mathrm{PDF}(\boldsymbol{z}^\ell)$ and $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. C.f. diffusion models:



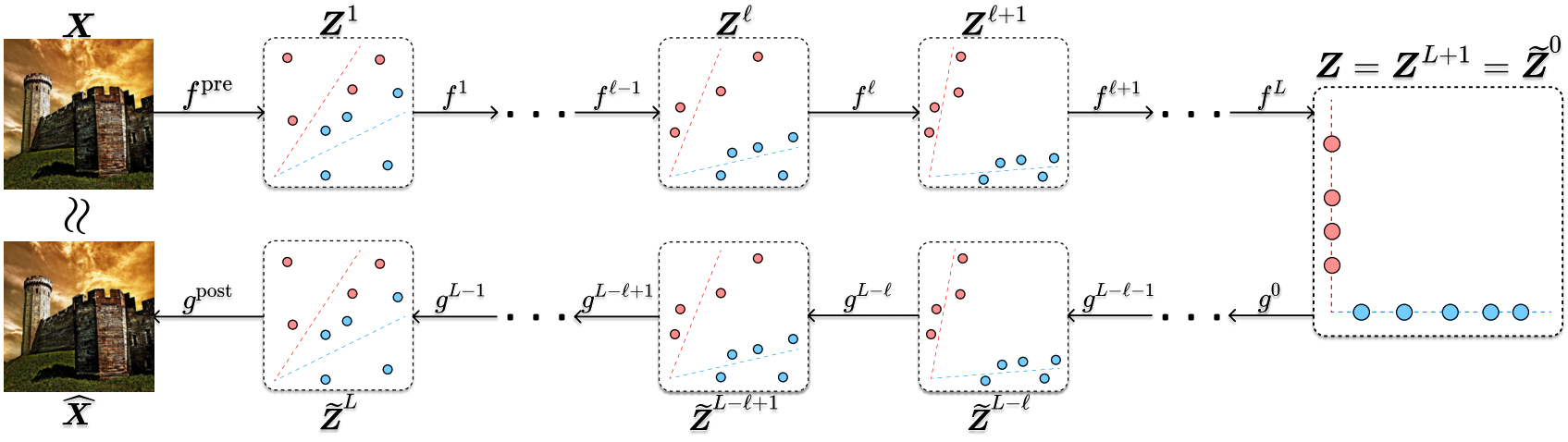
$$\tilde{\boldsymbol{z}}^{\ell+1} = g^\ell(\tilde{\boldsymbol{z}}^\ell) = \tilde{\boldsymbol{z}}^\ell + \tau^\ell \nabla \log q^\ell(\tilde{\boldsymbol{z}}^\ell)$$

Diffusion models are **iterative denoisers**.

Structured Denoising-Diffusion

Our alternative: **structured denoising-diffusion**, i.e.,

low-dim. data distribution $\longleftrightarrow$ structured representations.



Recall: Low-Dimensional Gaussian Mixture Codebook

We introduce a parametric statistical model for the representations:

Given orthonormal bases $\boldsymbol{U}_{[K]} = (\boldsymbol{U}_k)_{k\in[K]} \in (\mathbb{R}^{d\times p})^K$, we desire that **each token z_i of the representation Z has marginal distribution given by**

$$\boldsymbol{z}_i \overset{d}{=} \boldsymbol{U}_{s_i}\boldsymbol{\alpha}_i, \quad \forall i \in [n]$$

where $(s_i)_{i\in[n]} \in [K]^n$ are random variables corresponding to the subspace indices, and $(\boldsymbol{\alpha}_i)_{i\in[n]} \in (\mathbb{R}^p)^n$ are zero-mean standard Gaussian variables.

$\Longrightarrow$ **both the final representation distribution and incremental transformation operators are structured!**

Connection: Compression and Denoising

Recall: In CRATE, the `MSSA` operator approximates a *gradient descent step* on $R^c \implies$ compression.

Theorem (Informal)

Under certain verifiable conditions, if

- $\boldsymbol{Z} \sim$ Gaussian tokens model w/ signal model $\boldsymbol{U}_{[K]}$,
- and $\widetilde{\boldsymbol{Z}}_\sigma = \boldsymbol{Z} + \sigma \boldsymbol{G}$ where $\sigma > 0$ is small, $\boldsymbol{G} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and q^σ is density of $\widetilde{\boldsymbol{Z}}_\sigma$,

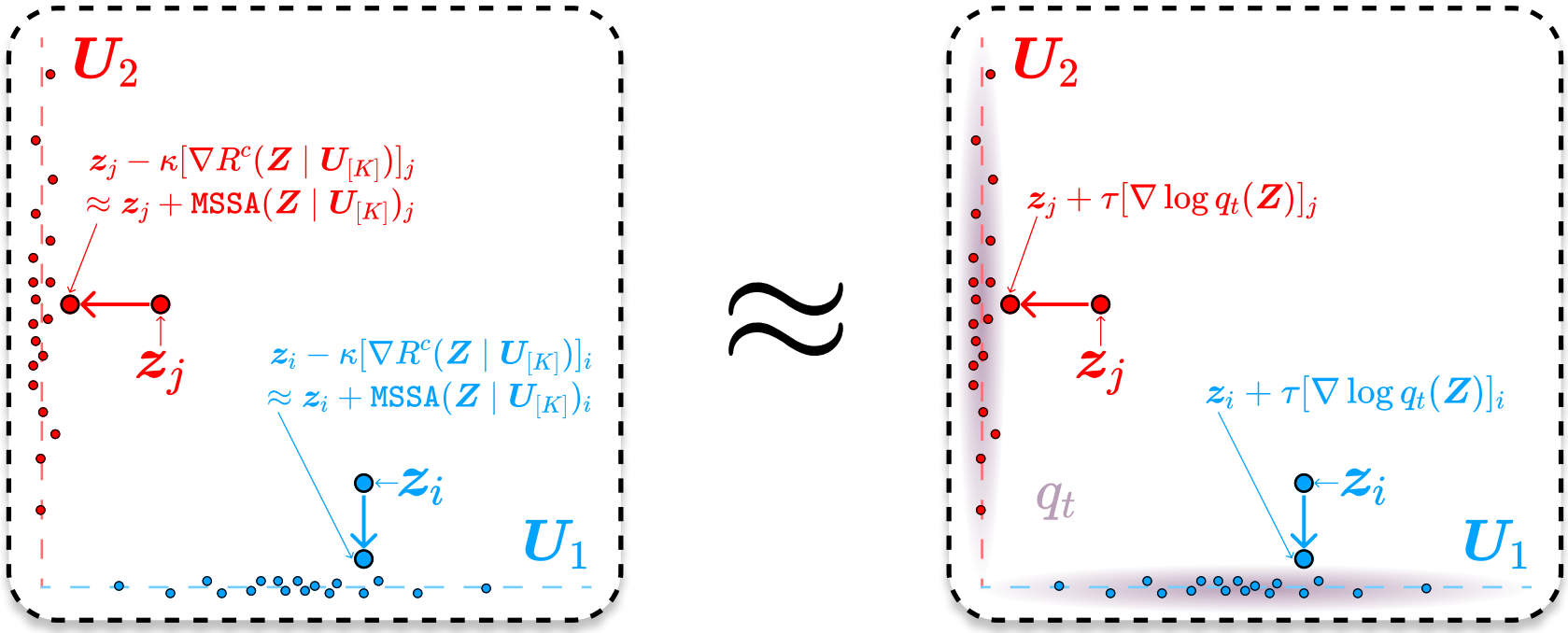
then w.h.p. $-\nabla R^c(\widetilde{\boldsymbol{Z}}_\sigma \mid \boldsymbol{U}_{[K]}) \propto \nabla \log q^\sigma(\widetilde{\boldsymbol{Z}}_\sigma)$ (approx.).

Upshot:

gradient descent on $R^c \approx$ gradient ascent on score function

$=$ denoising step

Compression and Structured Denoising-Diffusion



Structured denoising-diffusion models, where we replace the score with $-\nabla R^c$, are **iterative compressors**.

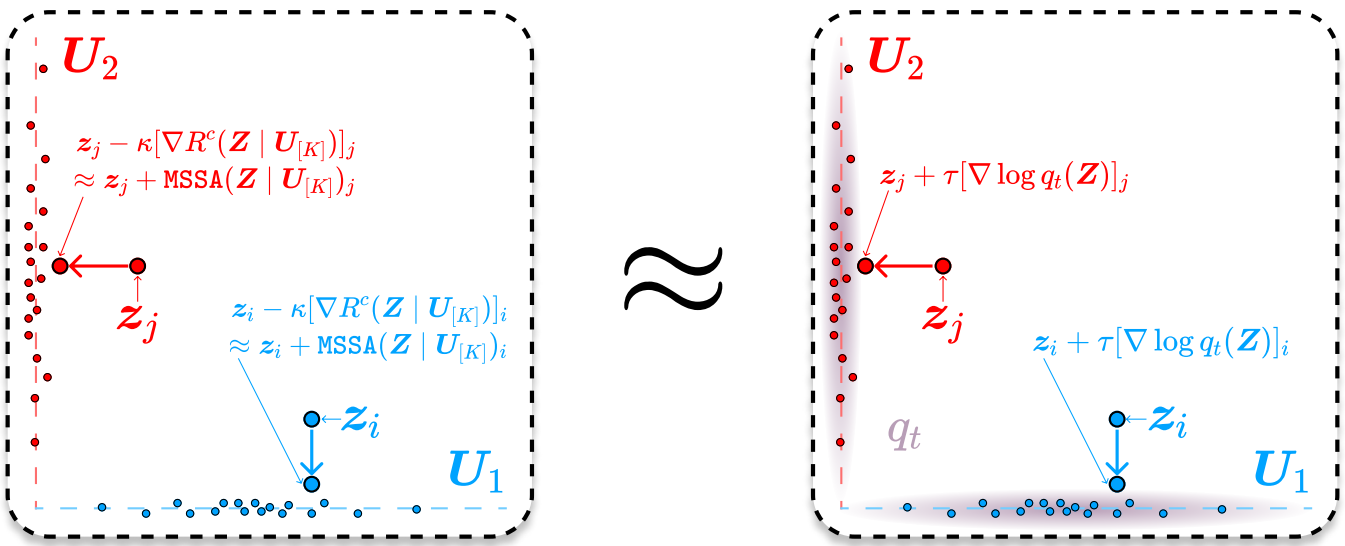
Structured Denoising-Diffusion

Structured denoising process:

$$\frac{d}{dt}\boldsymbol{Z}(t) = -\frac{1}{2(T-t)}\nabla R^c(\boldsymbol{Z}(t) \mid \boldsymbol{U}_{[K]}(t)), \qquad \forall t \in [0, T]$$

Time reversal = **structured diffusion** process:

$$\frac{d}{dt}\widetilde{\boldsymbol{Z}}(t) = \frac{1}{2t}\nabla R^c(\widetilde{\boldsymbol{Z}}(t) \mid \boldsymbol{U}_{[K]}(T-t)), \qquad \forall t \in [0, T]$$



Structured Denoising-Diffusion and Network Architectures

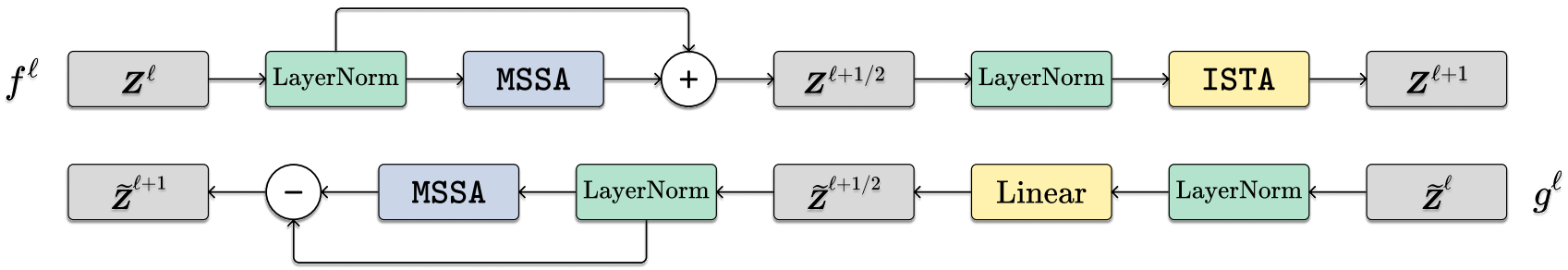
Discretizations:

$$\boldsymbol{Z}^{\ell+} = \boldsymbol{Z}^{\ell} - \kappa \nabla R^{c}(\boldsymbol{Z}^{\ell} \mid \boldsymbol{U}^{\ell}_{[K]}),$$
$$\widetilde{\boldsymbol{Z}}^{\ell+} = \boldsymbol{Z}^{\ell} + \kappa \nabla R^{c}(\widetilde{\boldsymbol{Z}}^{\ell} \mid \boldsymbol{U}^{L-\ell}_{[K]})$$

Approximations:

$$\boldsymbol{Z}^{\ell+} \approx (1 - \beta\kappa)\boldsymbol{Z}^{\ell} + \beta\kappa \texttt{MSSA}(\boldsymbol{Z}^{\ell} \mid \boldsymbol{U}^{\ell}_{[K]}),$$
$$\widetilde{\boldsymbol{Z}}^{\ell+} \approx (1 + \beta\kappa)\widetilde{\boldsymbol{Z}}^{\ell} - \beta\kappa \texttt{MSSA}(\widetilde{\boldsymbol{Z}}^{\ell} \mid \boldsymbol{U}^{L-\ell}_{[K]})$$

White-Box Autoencoder



(Encoder)

$$Z^{\ell+1/2} = Z^\ell + \texttt{MSSA}(Z^\ell \mid U^\ell_{[K]})$$

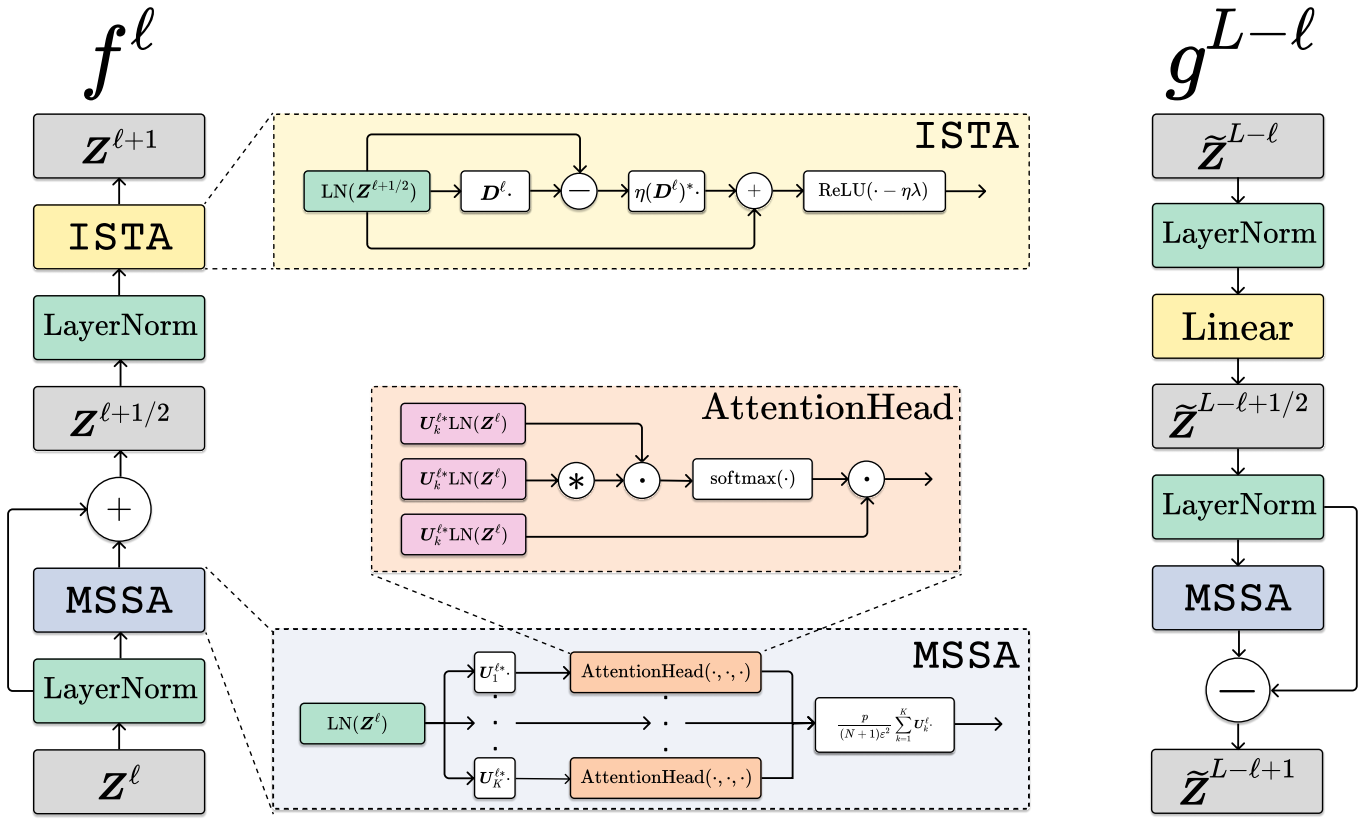
$$Z^{\ell+1} = \texttt{ISTA}(Z^\ell \mid D^\ell)$$

(Decoder)

$$\widetilde{Z}^{\ell+1/2} = E^\ell \widetilde{Z}^\ell$$

$$\widetilde{Z}^{\ell+1} = \widetilde{Z}^{\ell+1/2} - \texttt{MSSA}(\widetilde{Z}^{\ell+1/2} \mid V^\ell_{[K]})$$

CRATE-AE Operational Summary

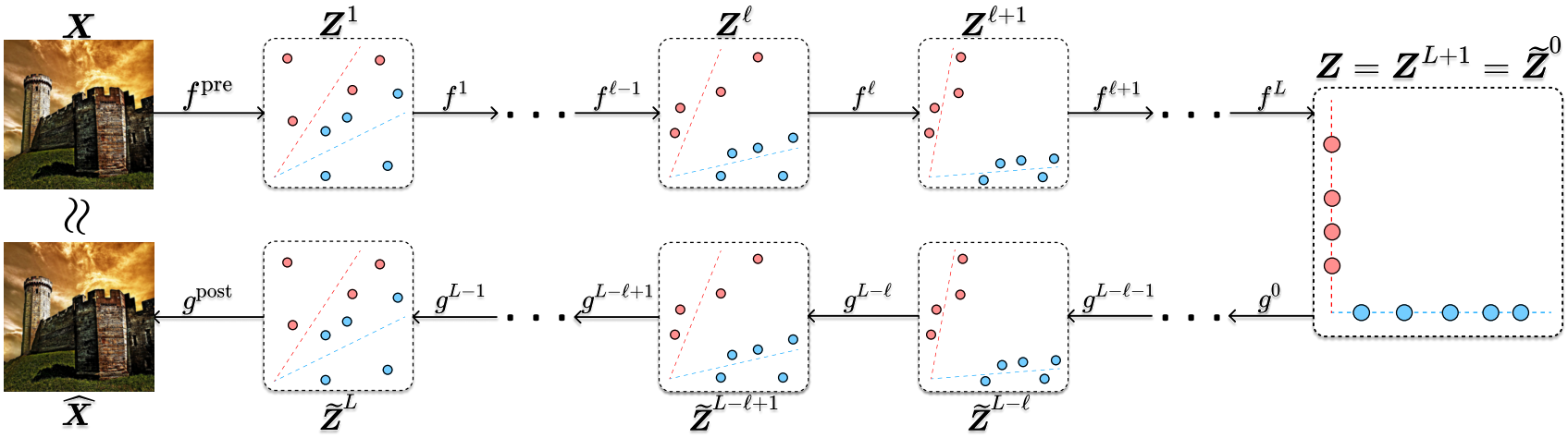


Encoder: Same as CRATE.

Decoder: `MSSA` is off by $\times -1$, "MLP" is a linear layer.

Learning Signal Models From Data

How to learn the true signal model(s)?



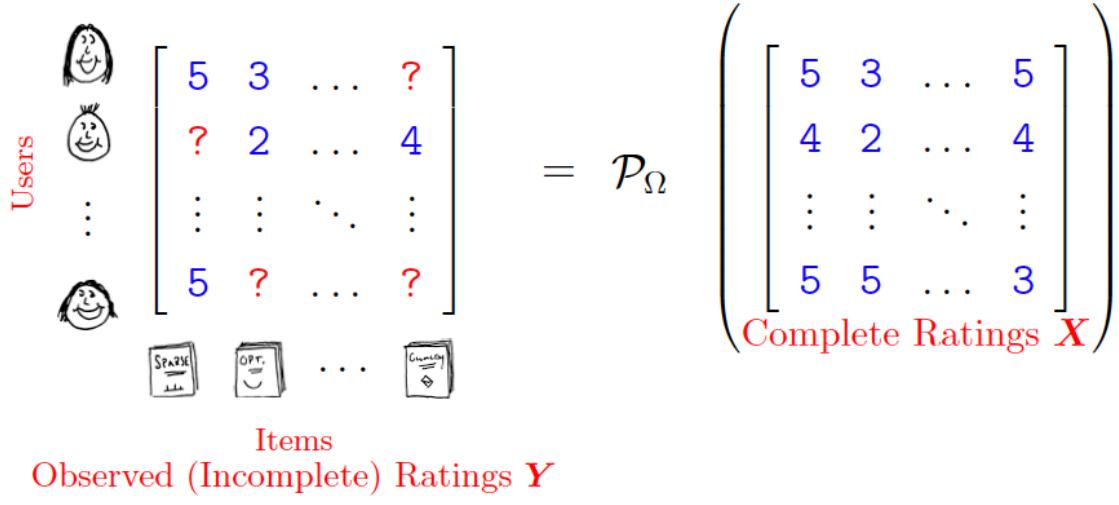
- Many ways to train this model — supervised/unsupervised
- Is there a training loss that enforces that the network models the structure of the data distribution?
- Revisit classical tools for learning + verifying low-dimensional structure: denoising, compression, **matrix completion**

Low-Rank Matrix Completion

Task: Given $\boldsymbol{X} = \texttt{Mask}_\Omega(\boldsymbol{X}_\natural)$, reconstruct $\boldsymbol{X}_\natural$.

Method: If $\boldsymbol{X}_\natural$ has low-rank, then we can efficiently recover it by *nuclear norm minimization*:

$$\min_{\boldsymbol{Y}} \left[\|\boldsymbol{Y}\|_* \doteq \sum_i \sigma_i(\boldsymbol{Y}) \right] \quad \text{subject to} \quad \boldsymbol{X} = \texttt{Mask}_\Omega(\boldsymbol{Y}).$$

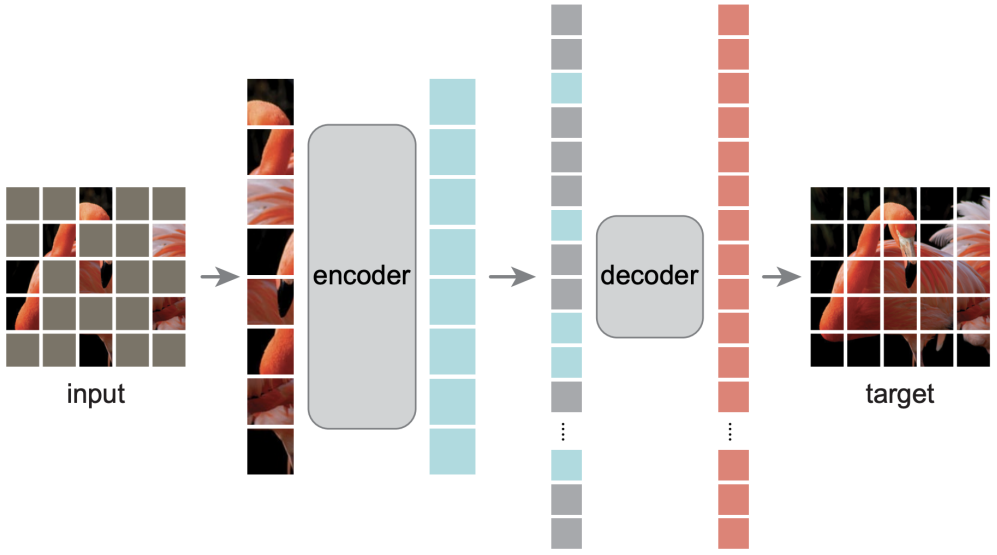


Wright and Ma 2022; Recht et al. 2010

Learning Low-Dimensional Structure via MAE

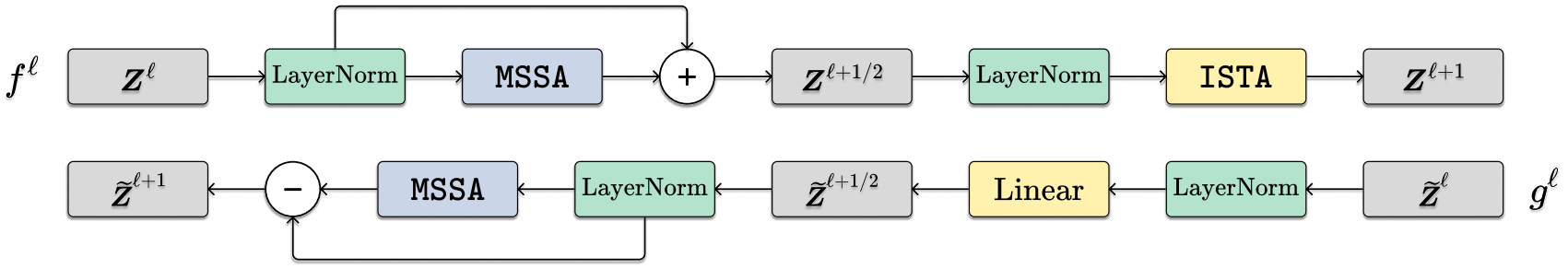
Question: What if the hidden $\boldsymbol{X}_\natural$ has *nonlinear, learnable* low-dimensional structure?

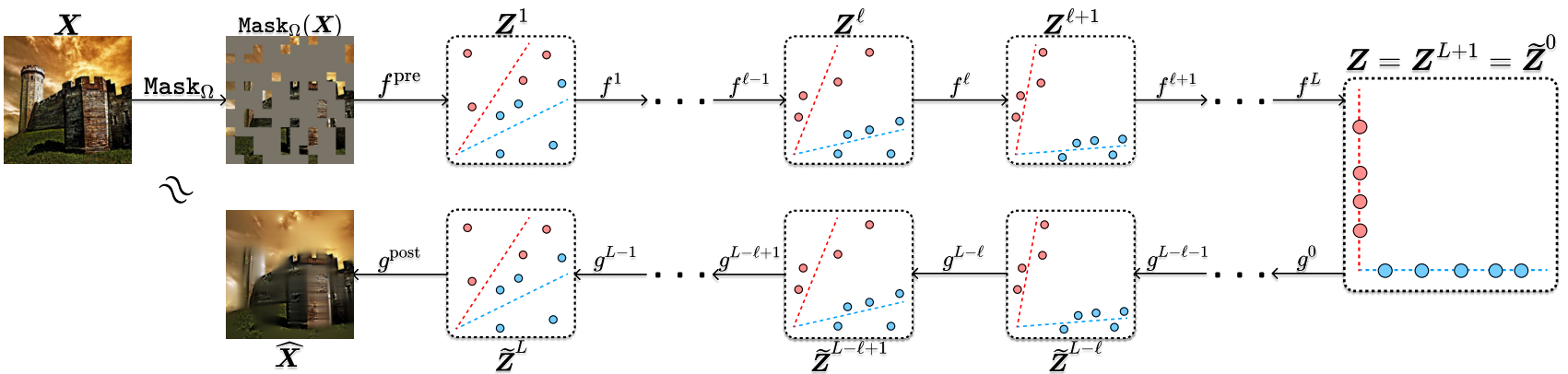
Modern, nonlinear, deep analogue: **masked autoencoding**.



He et al. 2022

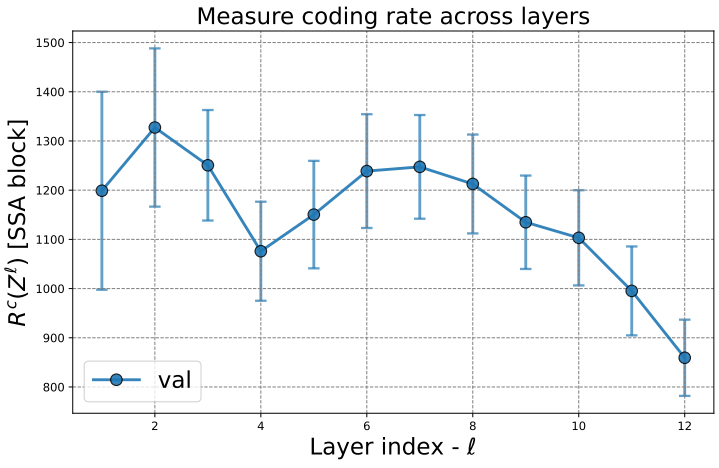
CRATE-MAE

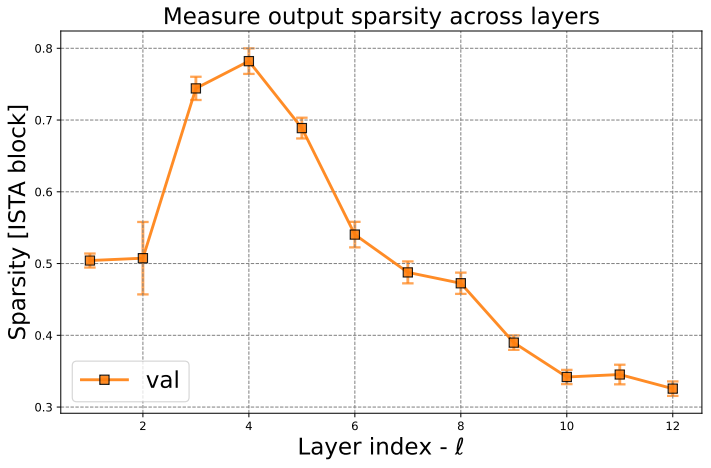




$$\mathcal{L}(f, g) \doteq \mathbb{E}\|\boldsymbol{X} - g(f(\mathtt{Mask}_\Omega(\boldsymbol{X})))\|_2^2$$

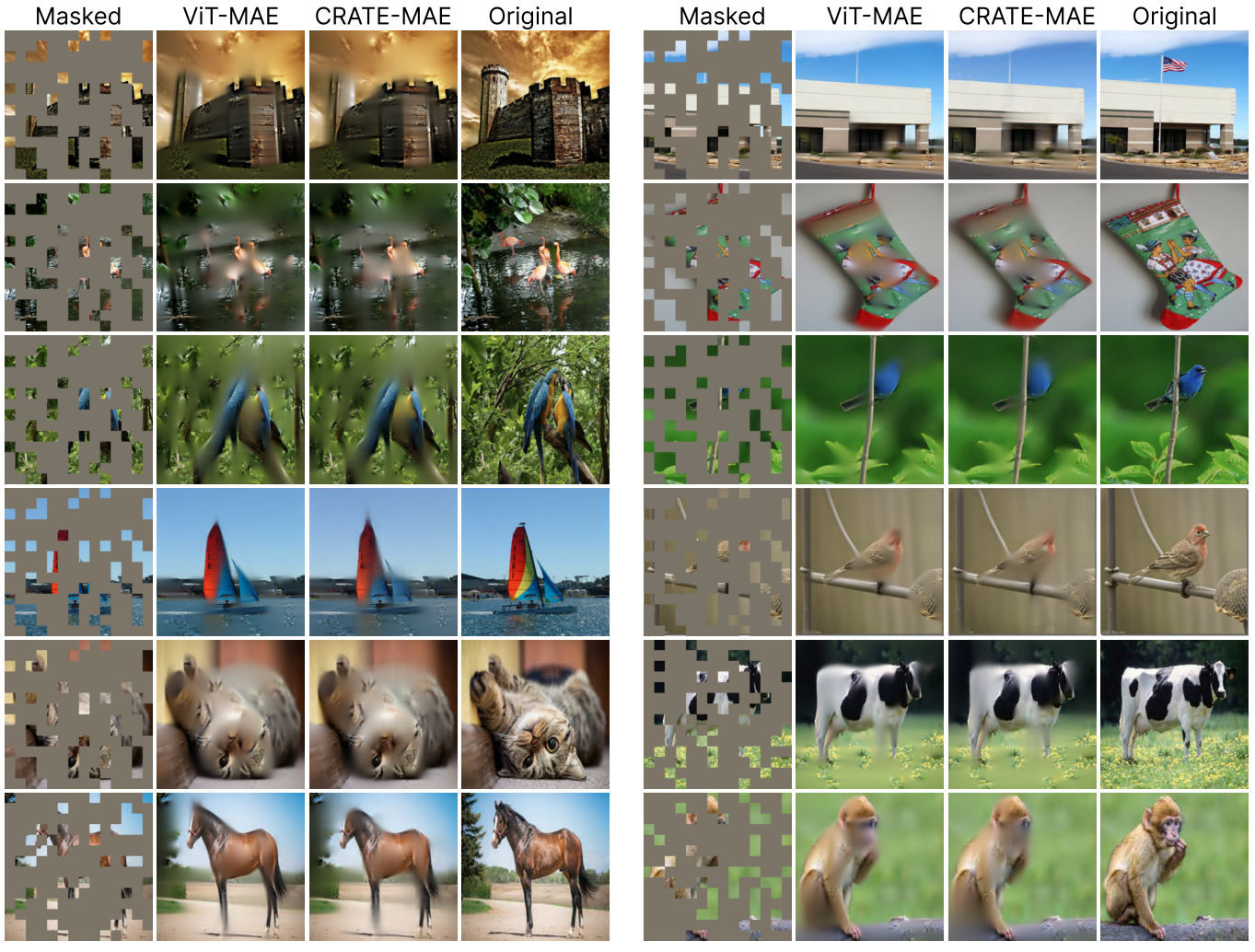
CRATE-MAE: Experimental Validation





Encoder learns compressed and sparse representation.

CRATE-MAE: Experimental Validation



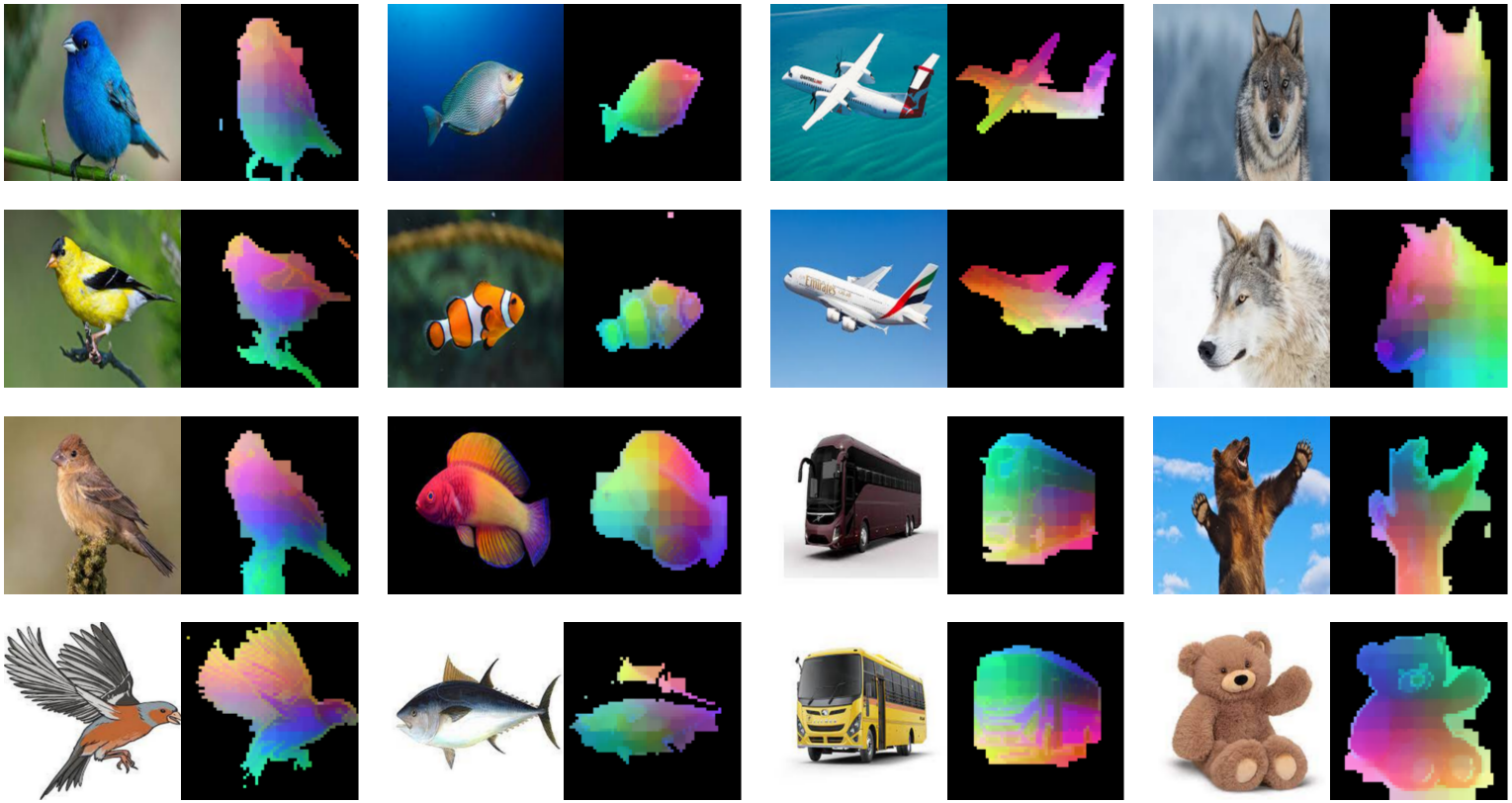
Visually similar reconstructions compared to ViT-MAE.

CRATE-MAE: Experimental Validation

Model	CRATE-MAE-S	CRATE-MAE-B	ViT-MAE-S	ViT-MAE-B
# parameters	25.4M	44.6M	47.6M	143.8M
Fine-Tuning				
CIFAR10	96.2	96.8	97.6	98.5
CIFAR100	79.0	80.3	83.0	87.0
Oxford Flowers-102	71.7	78.5	84.2	92.5
Oxford-IIIT-Pets	73.7	76.7	81.7	90.3
Linear Probing				
CIFAR10	79.4	80.9	79.9	87.9
CIFAR100	56.6	60.1	62.3	68.0
Oxford Flowers-102	57.7	61.8	66.8	66.4
Oxford-IIIT-Pets	40.6	46.2	51.8	80.1

Similar feature learning performance as ViT-MAE.

CRATE-MAE: Experimental Validation

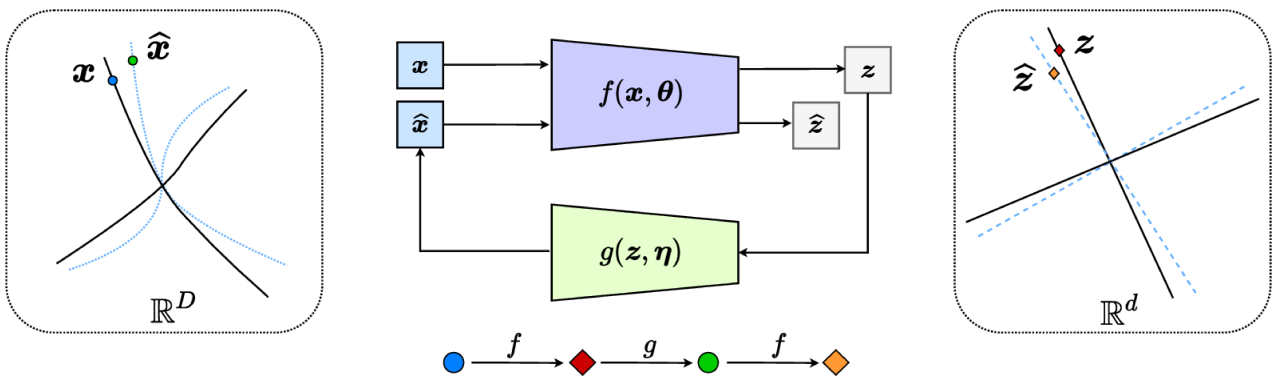


Features from the same class have linear subspace structure.

Outline

From Open-Loop to Closed-Loop?

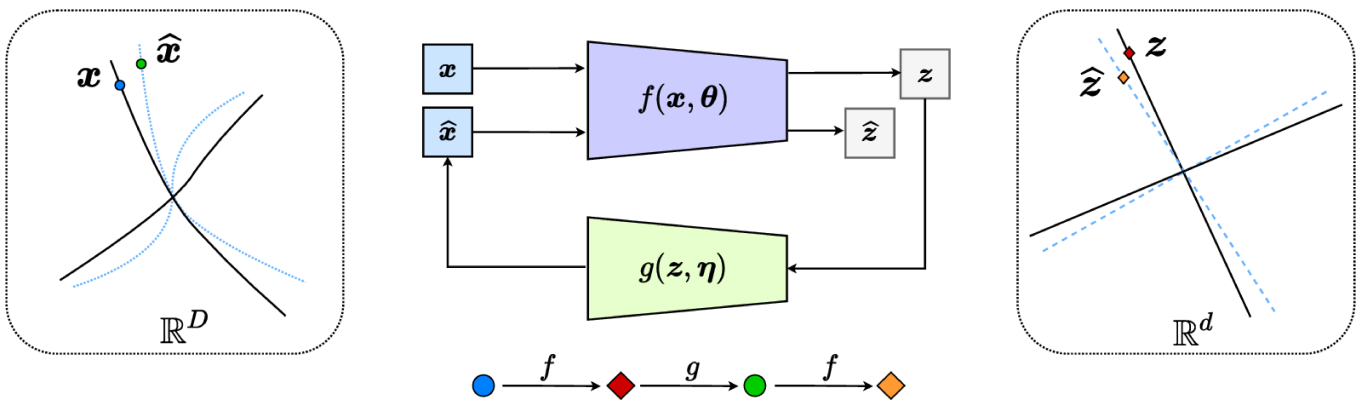
- Start: White-box deep architectures via unrolled optimization
- Recently: Auto-encoding for invertible representation
- Now: Closed-loop learning for self-consistency



How to ensure that the representations/signal models are learned *consistently* and *autonomously*?

From Open-Loop to Closed-Loop

- Open-loop white-box networks: need human-provided learning signal, i.e., class label (for classification task)
- Bi-directional white-box autoencoders: learning signal given by distance in $\boldsymbol{X}$ space, i.e., data space
- Closed-loop: learning signal given by distance in $\boldsymbol{Z}$ space between representations $\Longrightarrow$ self-consistency



From Auto-Encoding to Closed-Loop

Autoencoding loss:

$$\mathcal{L}(f, g) \doteq \mathbb{E}[d_{\ell^2}(\boldsymbol{X}, \widehat{\boldsymbol{X}})^2], \text{ where } \widehat{\boldsymbol{X}} = g(f(\boldsymbol{X})).$$

Masked autoencoding loss:

$$\mathcal{L}(f, g) \doteq \mathbb{E}[d_{\ell^2}(\boldsymbol{X}, \widehat{\boldsymbol{X}})^2], \text{ where } \widehat{\boldsymbol{X}} = g(f(\texttt{Mask}_\Omega(\boldsymbol{X}))).$$

Wasserstein GAN loss:

$$\mathcal{L}(f, g) \doteq d_{W_1}(\mathrm{Law}(\boldsymbol{X}), \mathrm{Law}(\widehat{\boldsymbol{X}}) \mid f), \text{ where } \widehat{\boldsymbol{X}} = g(\boldsymbol{W}).$$

What is the right distance $d(\boldsymbol{X}, \widehat{\boldsymbol{X}})$ for images?

Arjovsky et al. 2017

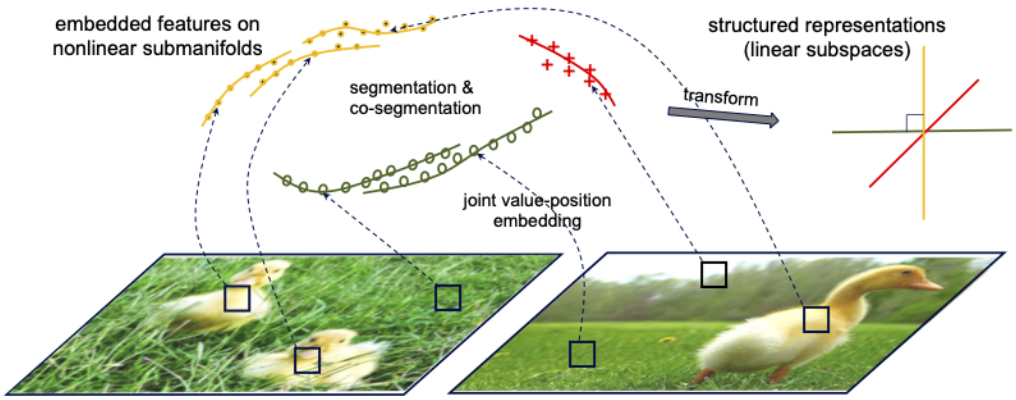
From Auto-Encoding to Closed-Loop

Rate reduction provides a natural measure of difference between representations $\boldsymbol{Z}_1$, $\boldsymbol{Z}_2$:

$$\Delta R(\boldsymbol{Z}_1, \boldsymbol{Z}_2) = R([\boldsymbol{Z}_1, \boldsymbol{Z}_2]) - \frac{1}{2}(R(\boldsymbol{Z}_1) + R(\boldsymbol{Z}_2)).$$

A good measure for representations $\boldsymbol{Z}$, but what about images $\boldsymbol{X}$?

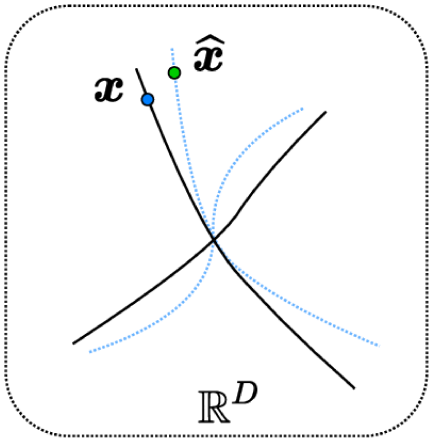
$$\boldsymbol{X} \xrightarrow{\ f\ } \boldsymbol{Z} \xrightarrow{\ g\ } \widehat{\boldsymbol{X}}$$

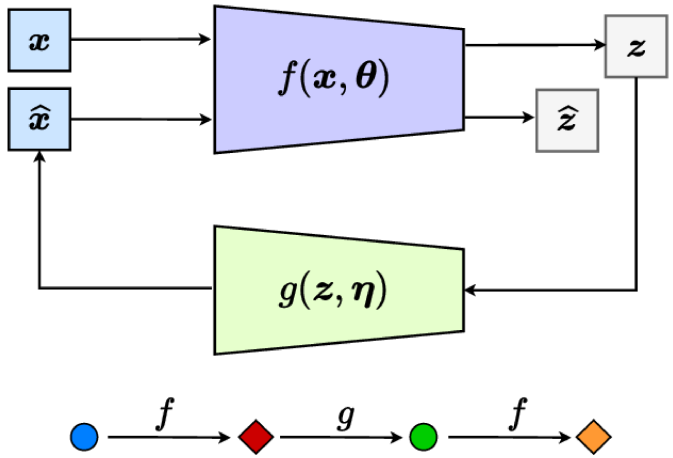


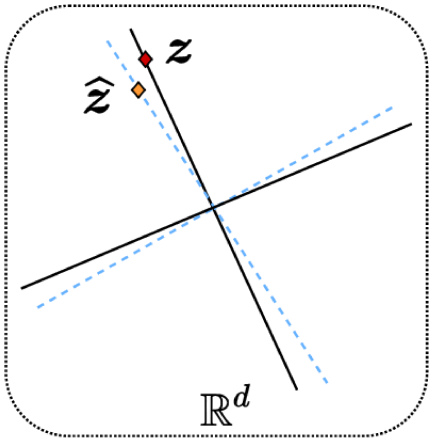
Do we *ever* need to measure distance in $\boldsymbol{X}$ space?

From Auto-Encoding to Closed-Loop

Can we measure distances **only** in Z space? **Yes!**







$$X \xrightarrow{f} Z \xrightarrow{g} \widehat{X} \xrightarrow{f} \widehat{Z}$$

Measure difference between data X and autoencoding $\widehat{X}$ through their representations Z and $\widehat{Z}$:

$$\Delta R(Z, \widehat{Z}) = R([Z, \widehat{Z}]) - \frac{1}{2}(R(Z) + R(\widehat{Z})).$$

Closed-Loop Transcription

This is the **closed-loop transcription (CTRL) framework**.

What are the roles of f and g, and how to train a model?

- f acts as an *encoder*, *discriminator*, and *sensor*.
- g acts as a *decoder*, *generator*, and *controller*.

A *closed-loop* notion of **self consistency** between $\boldsymbol{X}$ and $\widehat{\boldsymbol{X}}$ is defined by $f(\boldsymbol{X}) \approx f(\widehat{\boldsymbol{X}})$ and achieved by a *minimax game*:

$$\max_{f} \min_{g} \Delta R(\boldsymbol{Z}, \widehat{\boldsymbol{Z}}) \quad \text{where } \boldsymbol{Z} = f(\boldsymbol{X}),\ \widehat{\boldsymbol{Z}} = f(g(\boldsymbol{Z}))$$

CTRL: Theory for Special Case

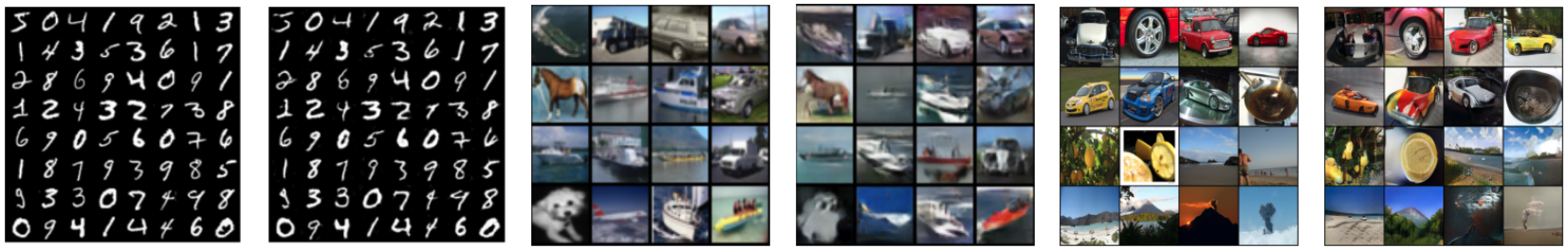
CTRL, Linear Case (Informal)

Suppose columns of $\boldsymbol{X}$ are drawn from (possibly correlated) linear subspaces. Then, under mild conditions (e.g., on dimensions), if f and g are linear functions, at the CTRL minimax game's equilibrium we have $\mathrm{span}(\boldsymbol{Z}) = \mathrm{span}(\widehat{\boldsymbol{Z}})$.

Upshot: CTRL learns self-consistent representations if data is piecewise linear (where linear functions are all you need).

Expect that this generalizes to more realistic settings (data supported on nonlinear manifolds, statistical assumptions, etc.), motivating its use in practice with NNs.

CTRL: Qualitative Generative Results



(a) MNIST $\boldsymbol{X}$ (b) MNIST $\hat{\boldsymbol{X}}$ (c) CIFAR-10 $\boldsymbol{X}$ (d) CIFAR-10 $\hat{\boldsymbol{X}}$ (e) ImageNet $\boldsymbol{X}$ (f) ImageNet $\hat{\boldsymbol{X}}$

CTRL maps semantically similar images to similar features.

$\Longrightarrow$ **self-consistent representation learning!**

CTRL: Quantitative Generative Results

Method	CIFAR-10		STL-10		ImageNet	
	IS↑	FID↓	IS↑	FID↓	IS↑	FID↓
GAN based methods						
DCGAN (Radford et al., 2015)	6.6	-	7.8	-	-	-
SNGAN (Miyato et al., 2018)	7.4	29.3	**9.1**	40.1	-	48.73
CSGAN (Wu et al., 2019b)	8.1	19.6	-	-	-	-
LOGAN (Wu et al., 2019a)	**8.7**	**17.7**	-	-	-	-
VAE/GAN based methods						
VAE (Kingma & Welling, 2013)	3.8	115.8	-	-	-	-
VAE/GAN (Larsen et al., 2016)	7.4	39.8	-	-	-	-
NVAE (Vahdat & Kautz, 2020)	-	50.8	-	-	-	-
DC-VAE (Parmar et al., 2021)	**8.2**	**17.9**	8.1	41.9	-	-
LDR-Binary (ours)	**8.1**	**19.6**	8.4	**38.6**	7.74	**46.95**
LDR-Multi (ours)	7.1	23.9	7.7	45.7	6.44	55.51

CTRL is competitive as a generative model vs. setups with similar compute/parameter count.

CTRL: Disentangled Representations

(a) Disentangled attributes as principal components

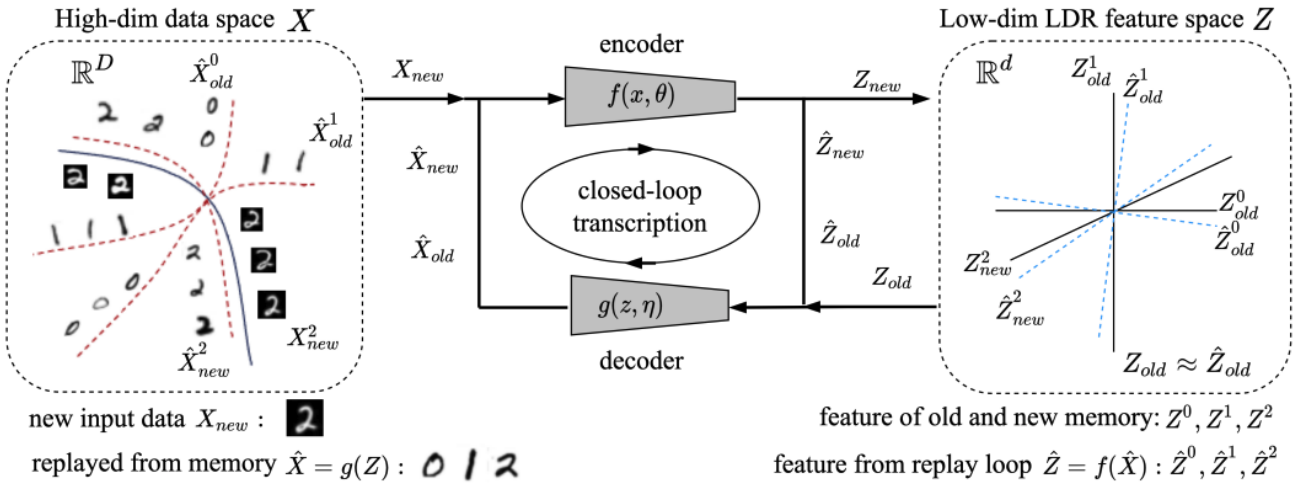
(b) Interpolation between distinct samples

$\Longrightarrow$ CTRL learns **disentangled latent representations**.

Eliminating Catastrophic Forgetting via CTRL

Closed-loop transcription alleviates catastrophic forgetting.

Test case: train a model one class at a time.



$$\max_f \min_g \quad \Delta R(\boldsymbol{Z}) + \Delta R(\widehat{\boldsymbol{Z}}) + \Delta R(\boldsymbol{Z}_{\text{new}}, \widehat{\boldsymbol{Z}}_{\text{new}})$$

$$\text{s.t.} \quad \Delta R(\boldsymbol{Z}_{\text{old}}, \widehat{\boldsymbol{Z}}_{\text{old}}) = 0.$$

CTRL: Quantitative Incremental Learning Results

Category	Method	MNIST				CIFAR-10			
		10-splits		5-splits		10-splits		5-splits	
		Last	Avg	Last	Avg	Last	Avg	Last	Avg
Regularization	LwF (Li & Hoiem, 2017)	-	-	0.196	0.455	-	-	0.196	0.440
	SI (Zenke et al., 2017)	-	-	0.193	0.461	-	-	0.196	0.441
Architecture	ReduNet (Wu et al., 2021)	-	-	0.961	0.982	-	-	0.539	0.645
Exemplar	iCaRL (Rebuffi et al., 2017)	0.322	0.588	0.725	0.803	0.212	0.431	0.487	0.632
	A-GEM (Chaudhry et al., 2018)	0.382	0.574	0.597	0.764	0.115	0.293	0.204	0.473
	CLR-ER (Arani et al., 2022)	-	-	-	0.895	-	-	-	0.662
Generative	DGMw (Ostapenko et al., 2019)	-	0.965	-	-	-	0.562	-	-
	EEC (Ayub & Wagner, 2020)	-	0.978	-	-	-	0.669	-	-
	EECS (Ayub & Wagner, 2020)	-	0.963	-	-	-	0.619	-	-
	i-CTRL (ours)	**0.975**	**0.989**	**0.978**	**0.990**	**0.599**	**0.727**	**0.627**	**0.723**

CTRL framework enables powerful incremental learning.

$\Longrightarrow$ **autonomous, continual learning?**

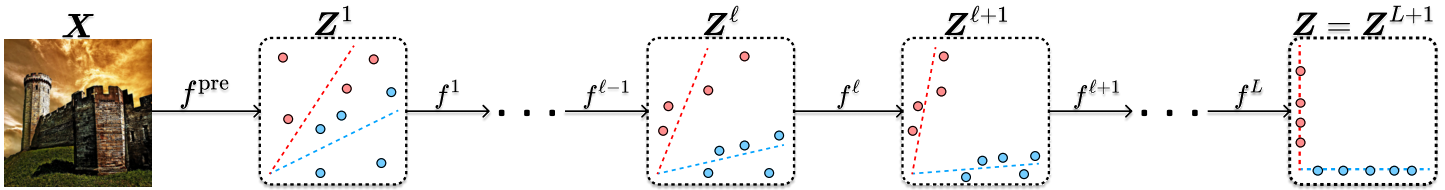
Key References

- Masked Completion via Structured Diffusion with White-Box Transformers
  https://arxiv.org/abs/2404.02446
- Closed-Loop Data Transcription to an LDR via Minimaxing Rate Reduction
  https://arxiv.org/abs/2111.06636
- Website: https://ma-lab-berkeley.github.io/CRATE/
- GitHub: https://github.com/Ma-Lab-Berkeley/CRATE

Outline

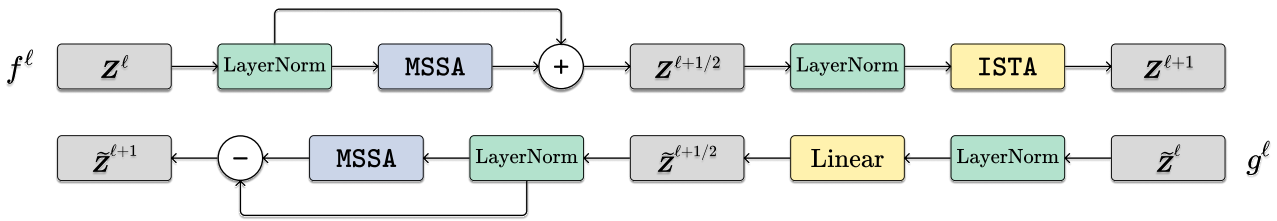
Conclusions of This Tutorial, Part 1

- **White-box deep networks via rate reduction.**
  - Introduce a principled objective which promotes *compressed, linearized* and *sparse* representations.
  - Then, deep networks, such as ReduNet ($\approx$ ResNet), are just unrolled iterative schemes for optimizing such an objective.
- **White-box transformer via sparse rate reduction.**
  - A principled objective which promotes *compressed*, *linearized*, and *sparse* representations.
  - A white-box transformer architecture, CRATE, constructed from unrolled optimization on the sparse rate reduction, which performs compressive encoding at scale.

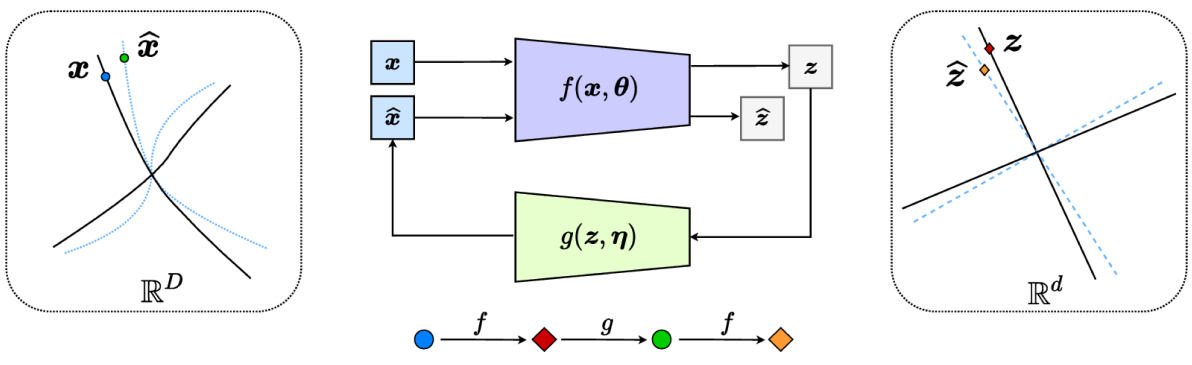


Conclusions of This Tutorial, Part 2

- **Bi-directional autoencoding for consistency.**
  - Structured denoising-diffusion (unifying compression, denoising, and transformers in the case where the desired representation is a mixture of subspaces).

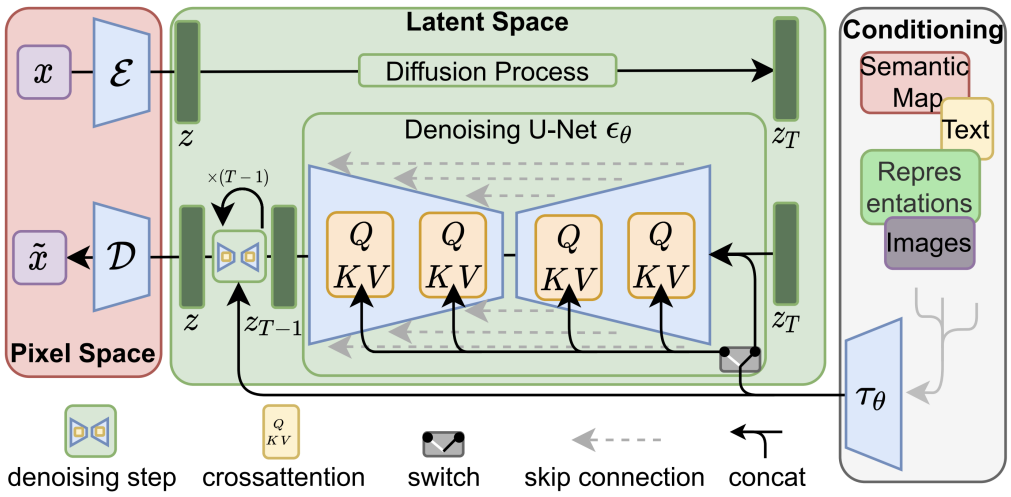


- **Closed-loop transcription for self-consistency.**
  - Autonomous self-consistent representation learning (provably correct for a mixture of subspaces).



What's Next?

- More efficient architectures using more advanced optimization and sampling techniques.
- Better learning strategies (layer-wise? backward and forward?)
- Truly controllable generation for generative models which also exploit learned compact and structured representations.



Tutorial Conclusion

Thank You! Questions?