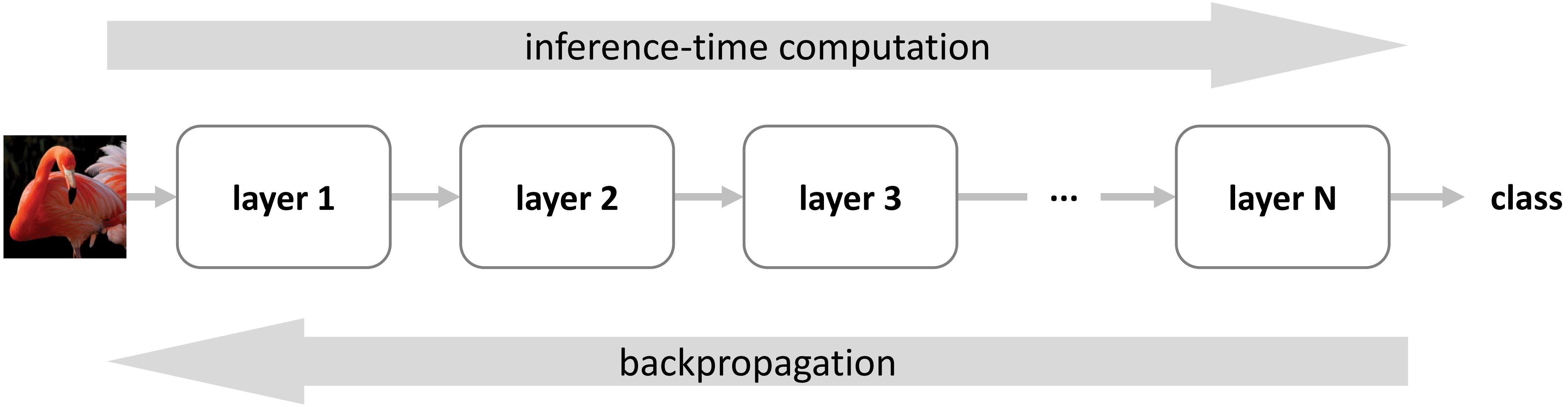
Towards End-to-End Generative Modeling

Kaiming He
Associate Professor, EECS, MIT

Tutorial/Workshop at CVPR 2025

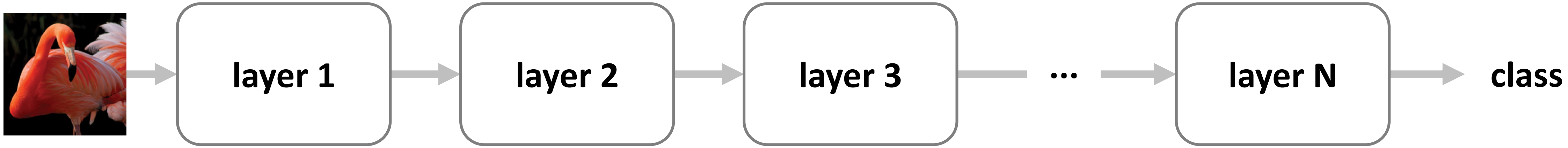
A Bit of History ...

- Since AlexNet, **recognition** models have been generally **end-to-end** ...



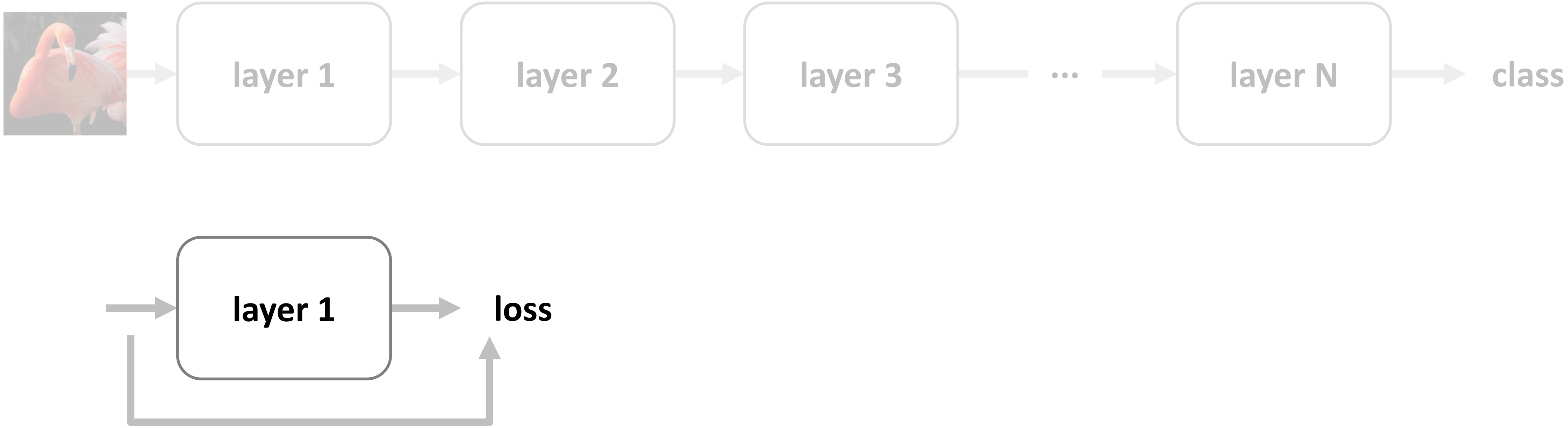
A Bit of History ...

- But before AlexNet, **layer-wise training** was a more popular solution
  - Deep Belief Nets (**DBN**) [Hinton et al, 2006]
  - Denoising Autoencoders (**DAE**) [Vincent et al, 2010, 2011]



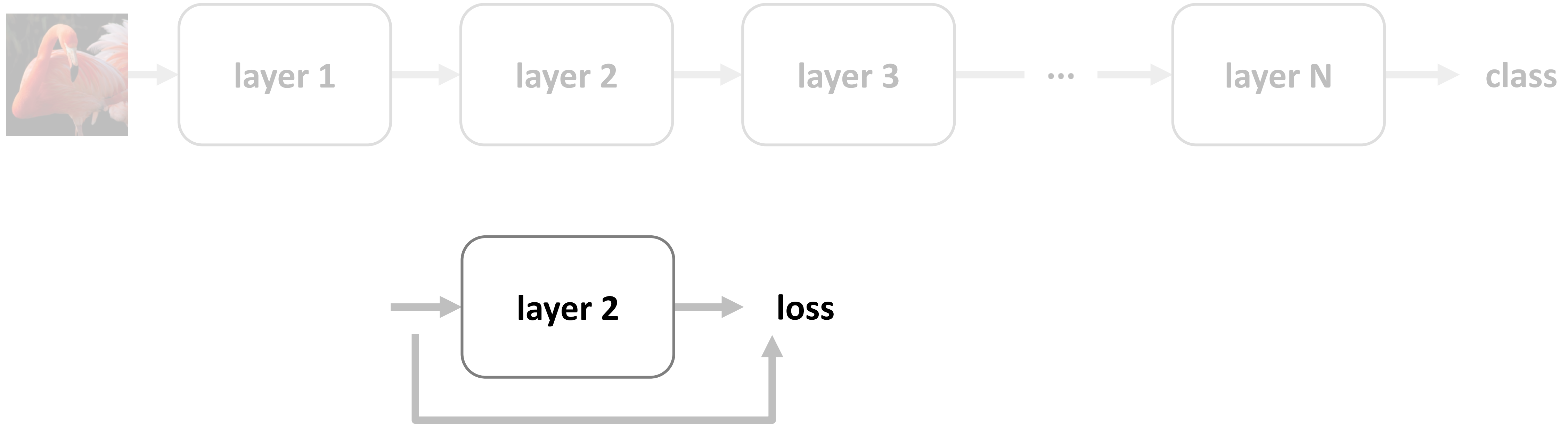
A Bit of History ...

- But before AlexNet, **layer-wise training** was a more popular solution
  - Deep Belief Nets (**DBN**) [Hinton et al, 2006]
  - Denoising Autoencoders (**DAE**) [Vincent et al, 2010, 2011]



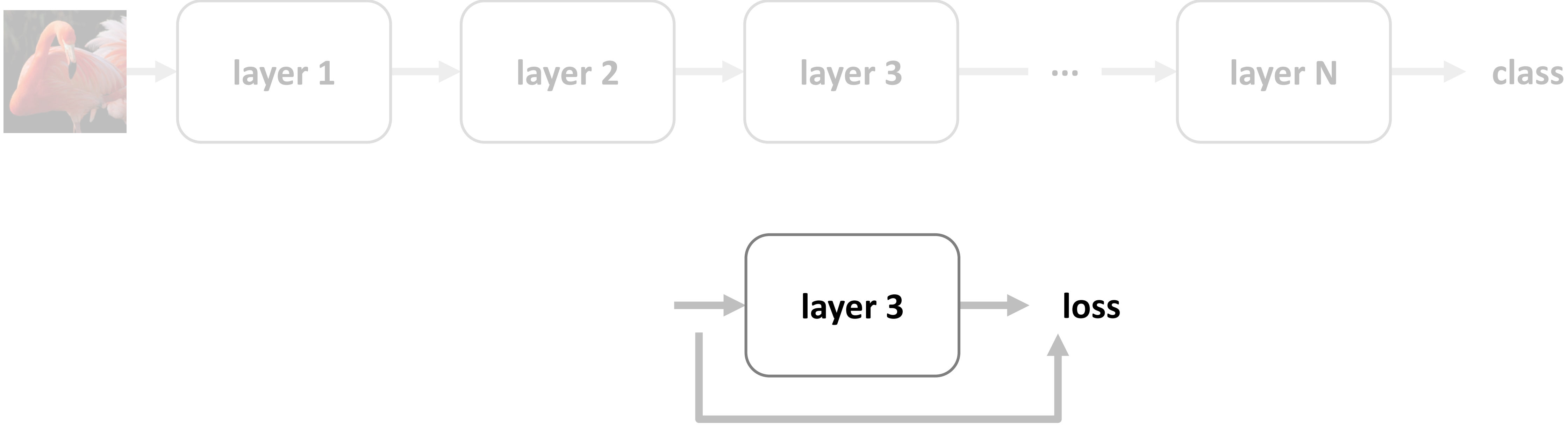
A Bit of History ...

- But before AlexNet, **layer-wise training** was a more popular solution
  - Deep Belief Nets (**DBN**) [Hinton et al, 2006]
  - Denoising Autoencoders (**DAE**) [Vincent et al, 2010, 2011]



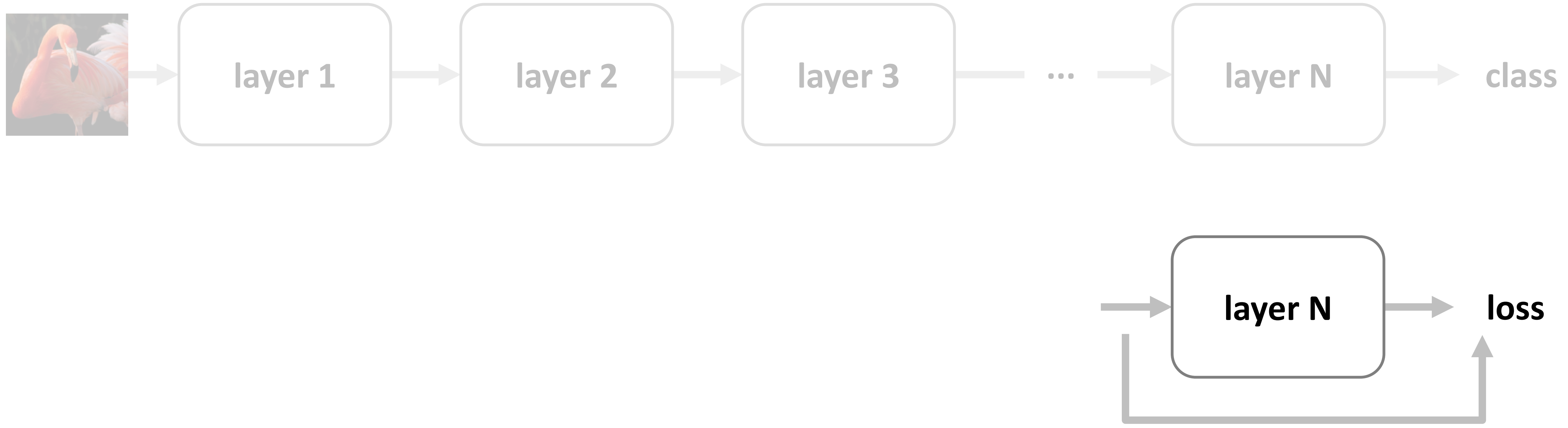
A Bit of History ...

- But before AlexNet, **layer-wise training** was a more popular solution
  - Deep Belief Nets (**DBN**) [Hinton et al, 2006]
  - Denoising Autoencoders (**DAE**) [Vincent et al, 2010, 2011]



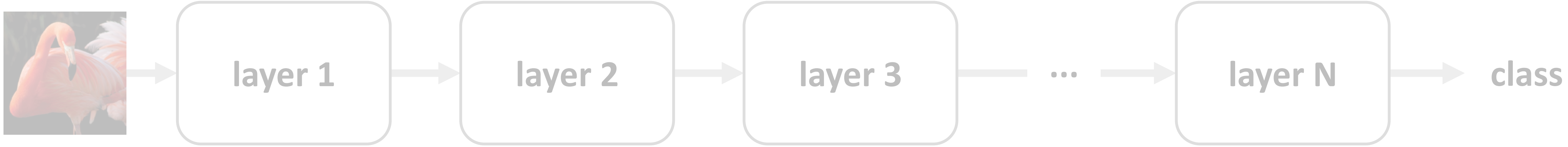
A Bit of History ...

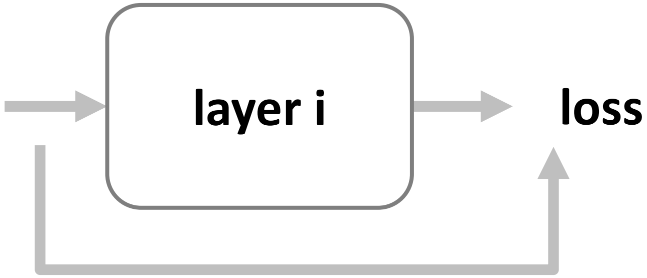
- But before AlexNet, **layer-wise training** was a more popular solution
  - Deep Belief Nets (**DBN**) [Hinton et al, 2006]
  - Denoising Autoencoders (**DAE**) [Vincent et al, 2010, 2011]



A Bit of History ...

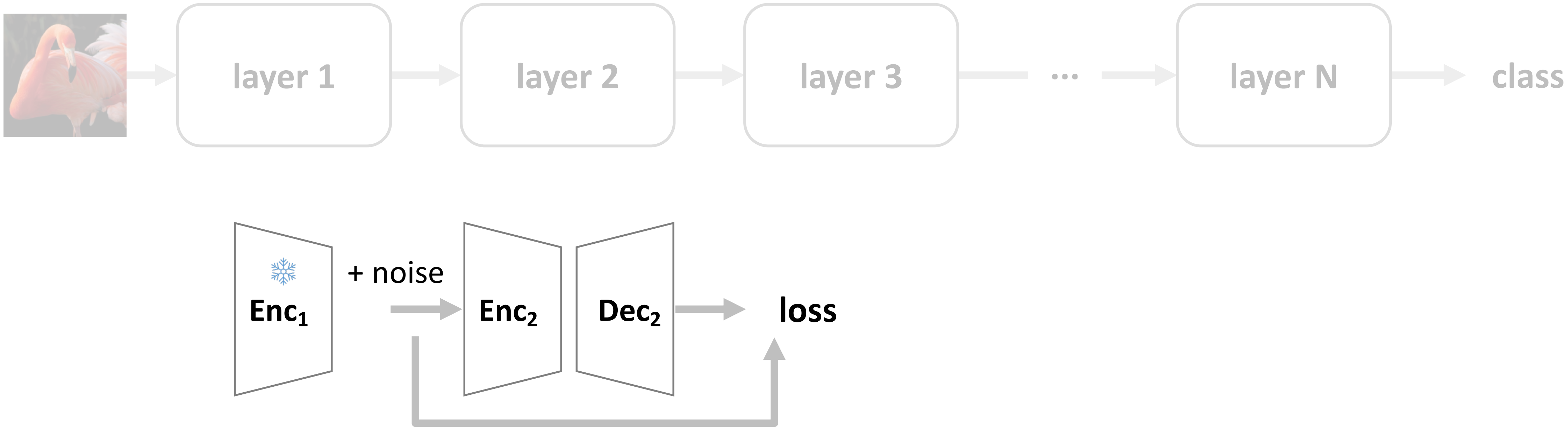
- But before AlexNet, **layer-wise training** was a more popular solution
  - Deep Belief Nets (**DBN**) [Hinton et al, 2006]
  - Denoising Autoencoders (**DAE**) [Vincent et al, 2010, 2011]





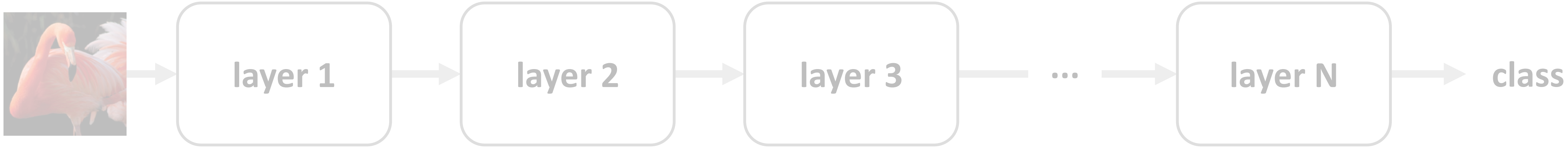
A Bit of History ...

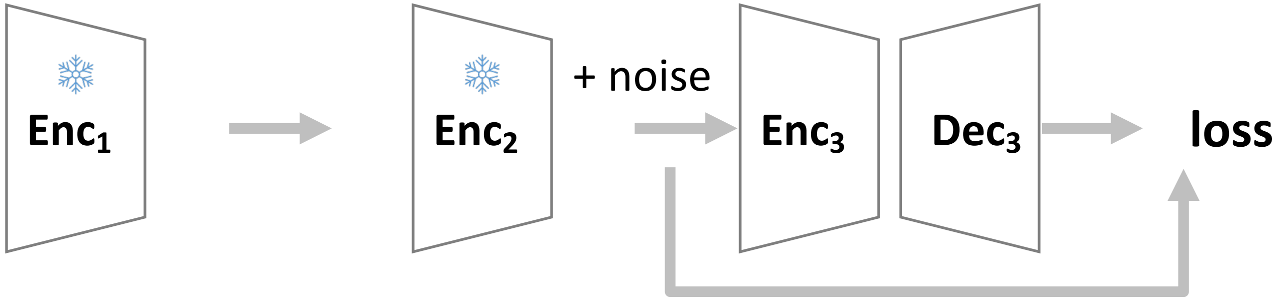
- But before AlexNet, **layer-wise training** was a more popular solution
  - Deep Belief Nets (**DBN**) [Hinton et al, 2006]
  - Denoising Autoencoders (**DAE**) [Vincent et al, 2010, 2011]



A Bit of History ...

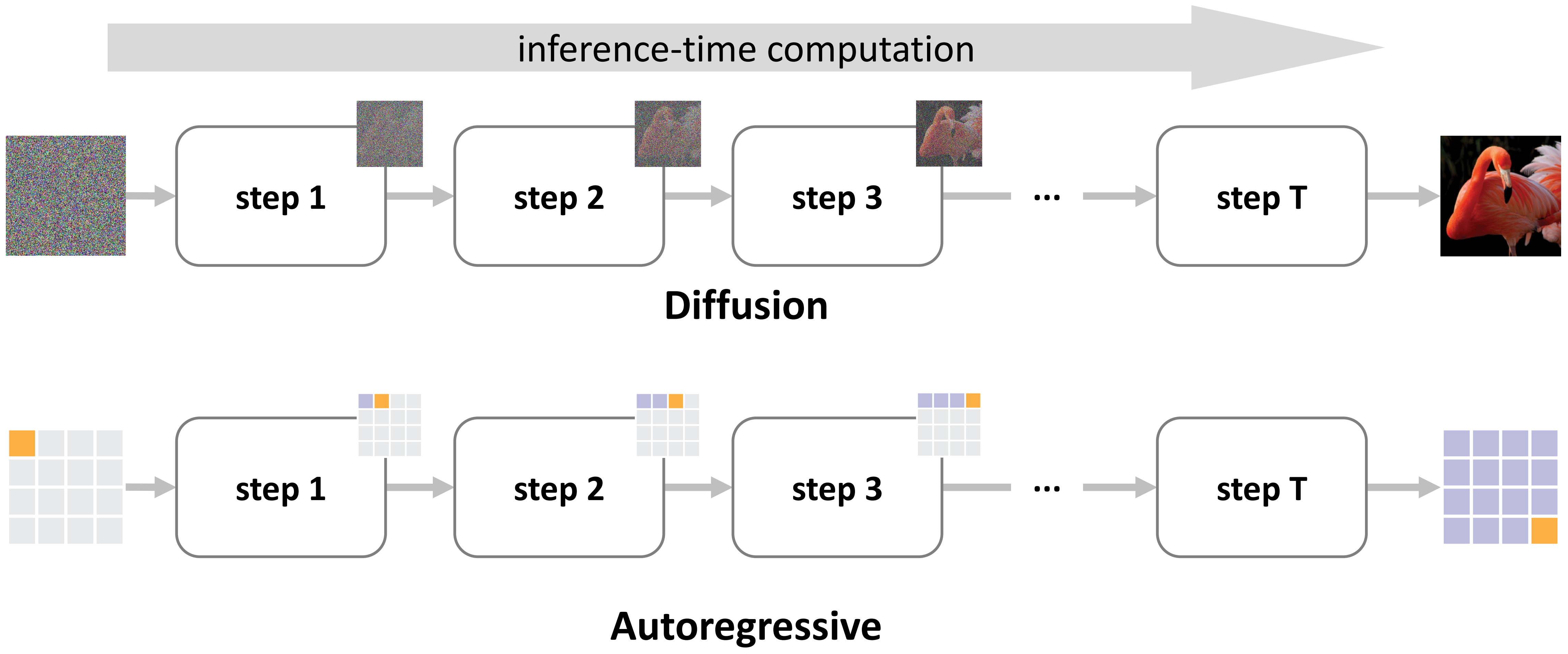
- But before AlexNet, **layer-wise training** was a more popular solution
  - Deep Belief Nets (**DBN**) [Hinton et al, 2006]
  - Denoising Autoencoders (**DAE**) [Vincent et al, 2010, 2011]





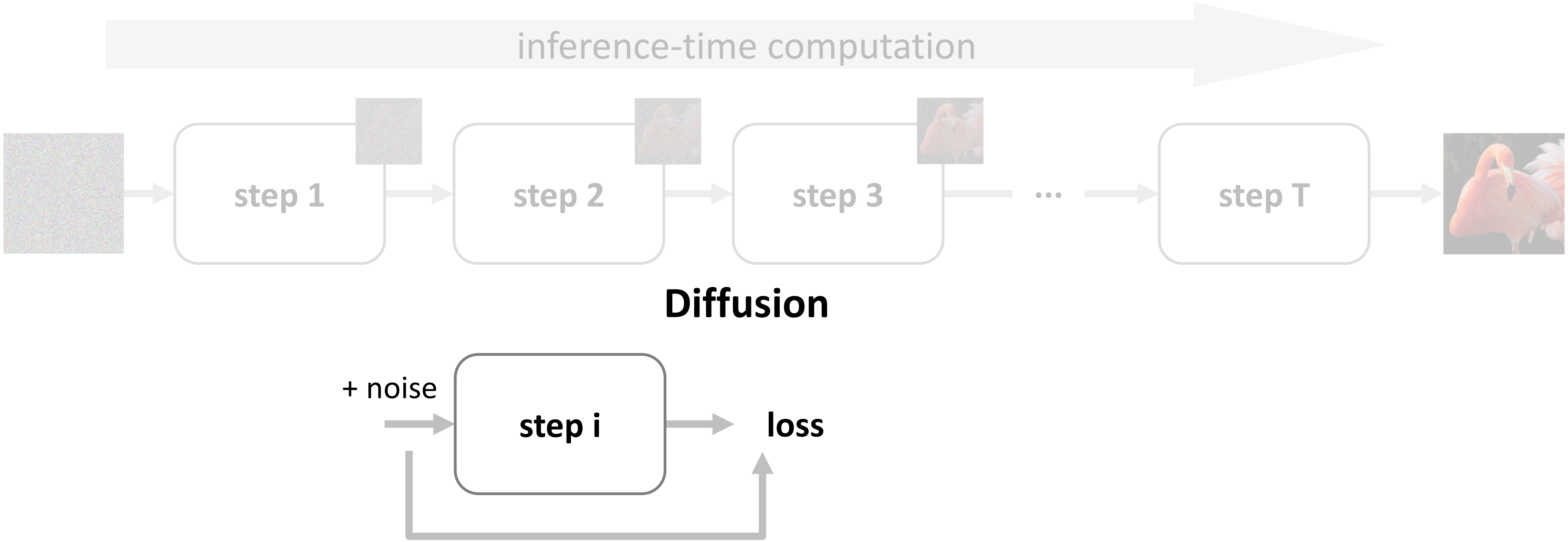
History Repeating in Generative Models?

- Today's generative models are conceptually like "layer-wise training"

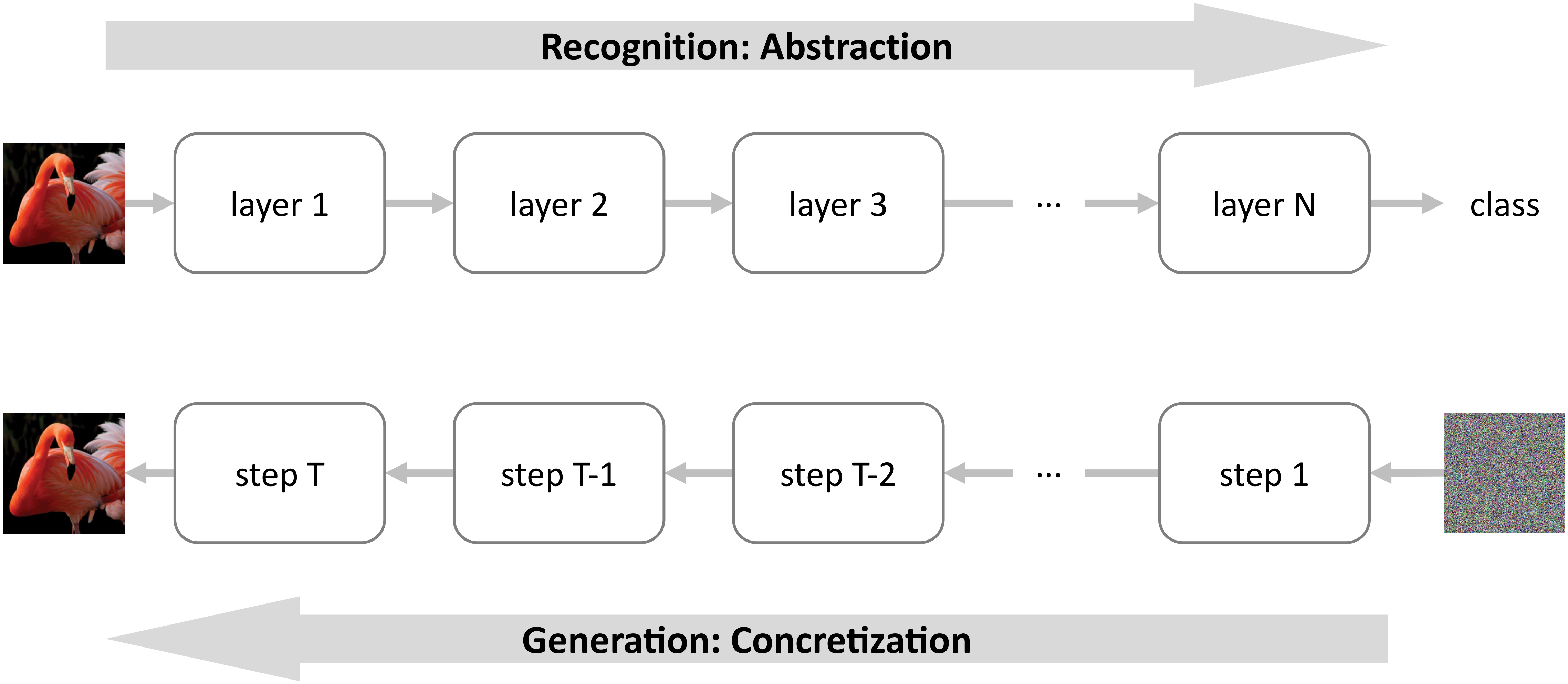


History Repeating in Generative Models?

- Today's generative models are conceptually like "layer-wise training"



Recognition vs. Generation: Two Sides of the Same Coin?



Recognition vs. Generation: Two Sides of the Same Coin?

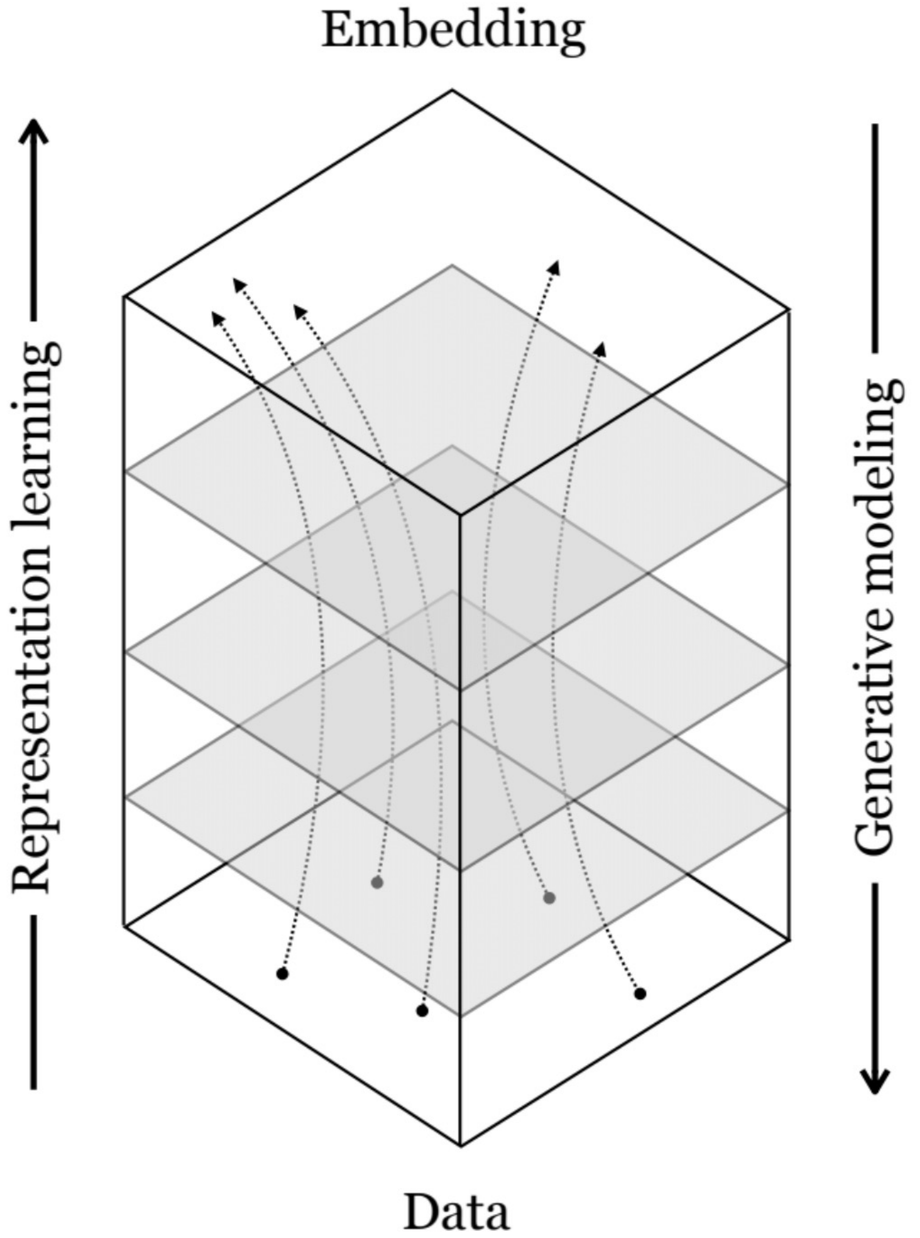
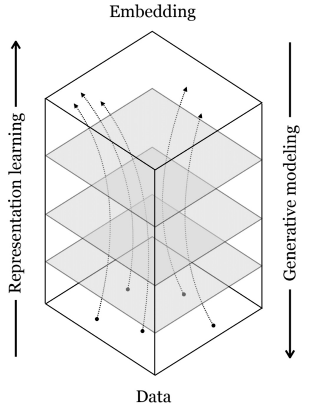


Illustration Credit: Phillip Isola

linear



$$\mathbf{x}_{\text{out}} = \mathbf{W}\mathbf{x}_{\text{in}} + \mathbf{b}$$

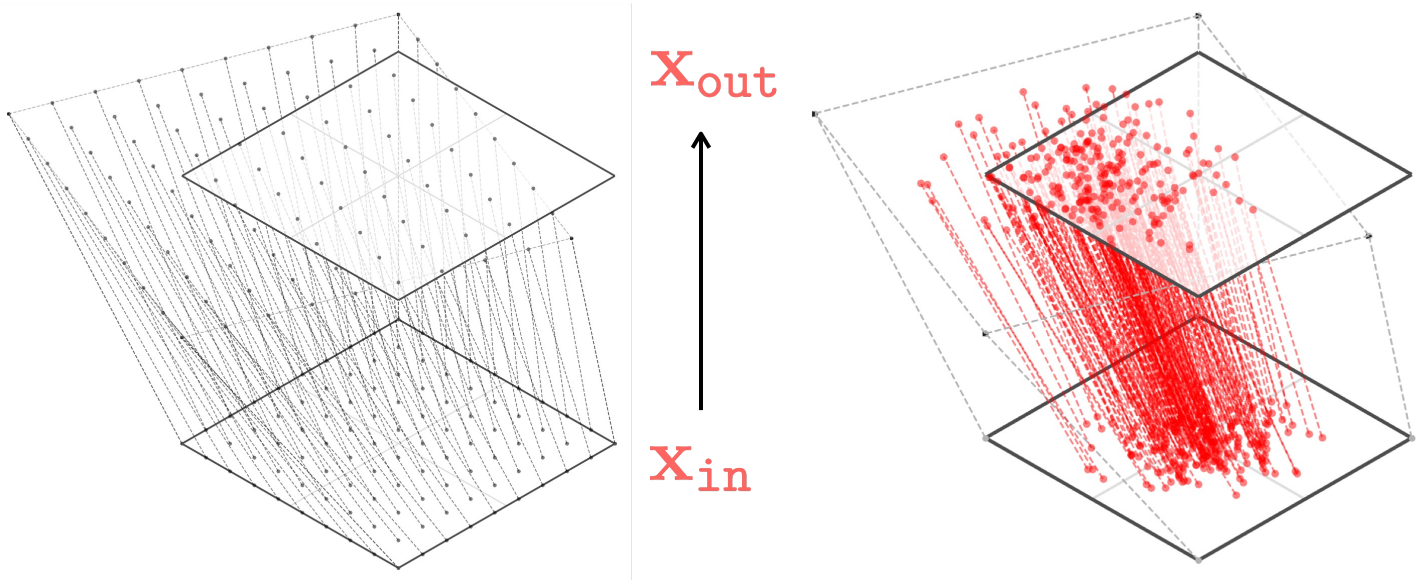
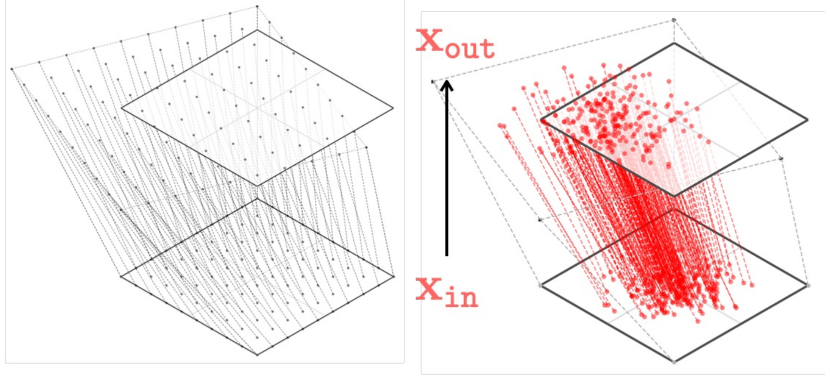
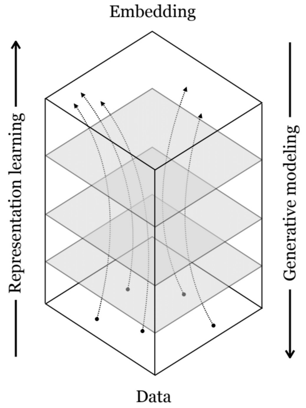


Illustration Credit: Phillip Isola

linear

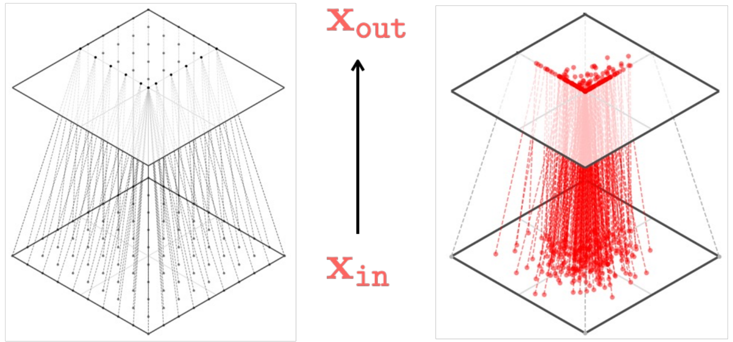
$$\mathbf{x}_{\text{out}} = \mathbf{W}\mathbf{x}_{\text{in}} + \mathbf{b}$$





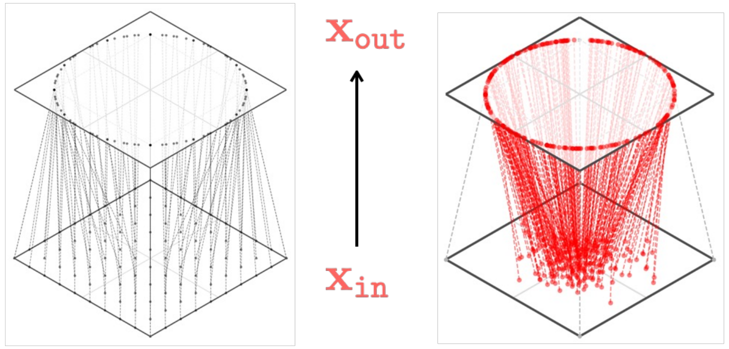
relu

$$x_{\text{out}}[i] = \max(x_{\text{in}}[i], 0)$$



L2-norm

$$x_{\text{out}}[i] = \frac{x_{\text{in}}[i]}{\|\mathbf{x}_{\text{in}}\|_2}$$



softmax

$$x_{\text{out}}[i] = \frac{e^{-\tau x_{\text{in}}[i]}}{\sum_{k=1}^{K} e^{-\tau x_{\text{in}}[k]}}$$

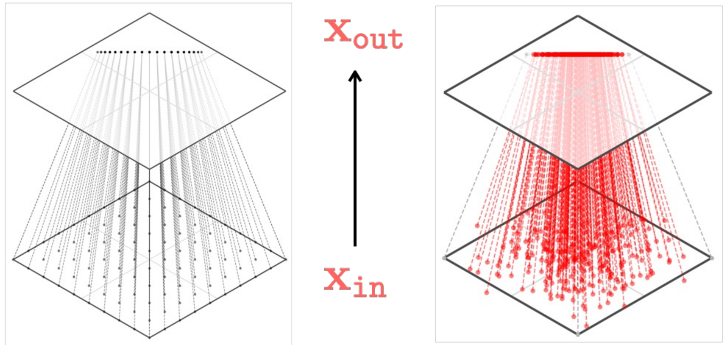
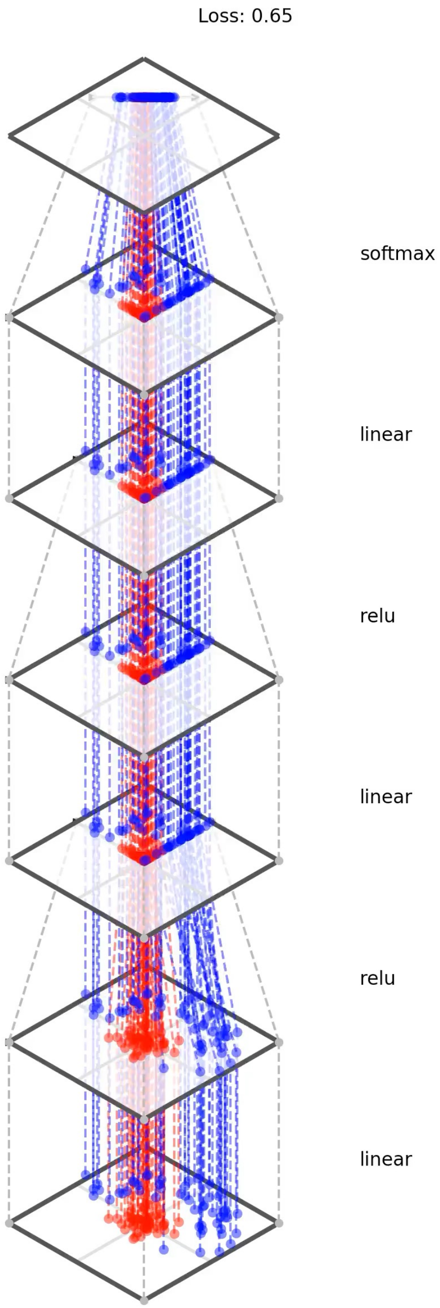


Illustration Credit: Phillip Isola



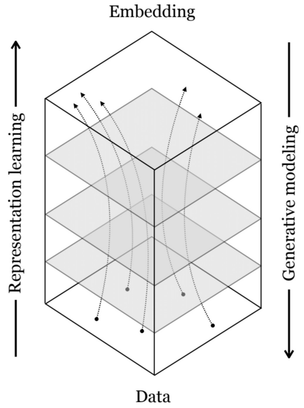
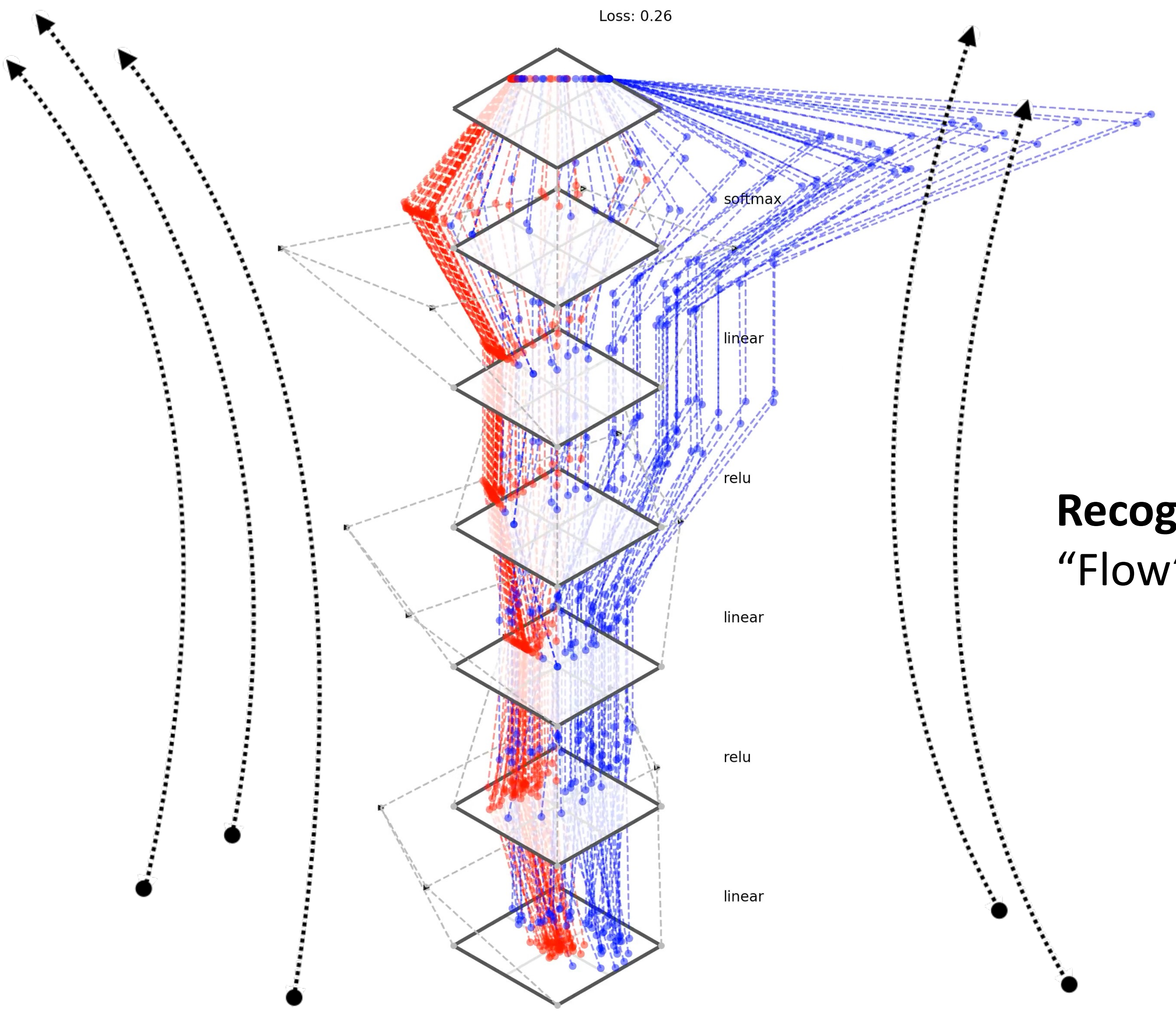
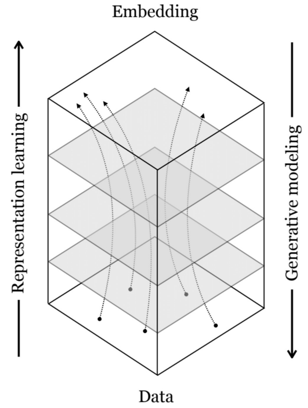


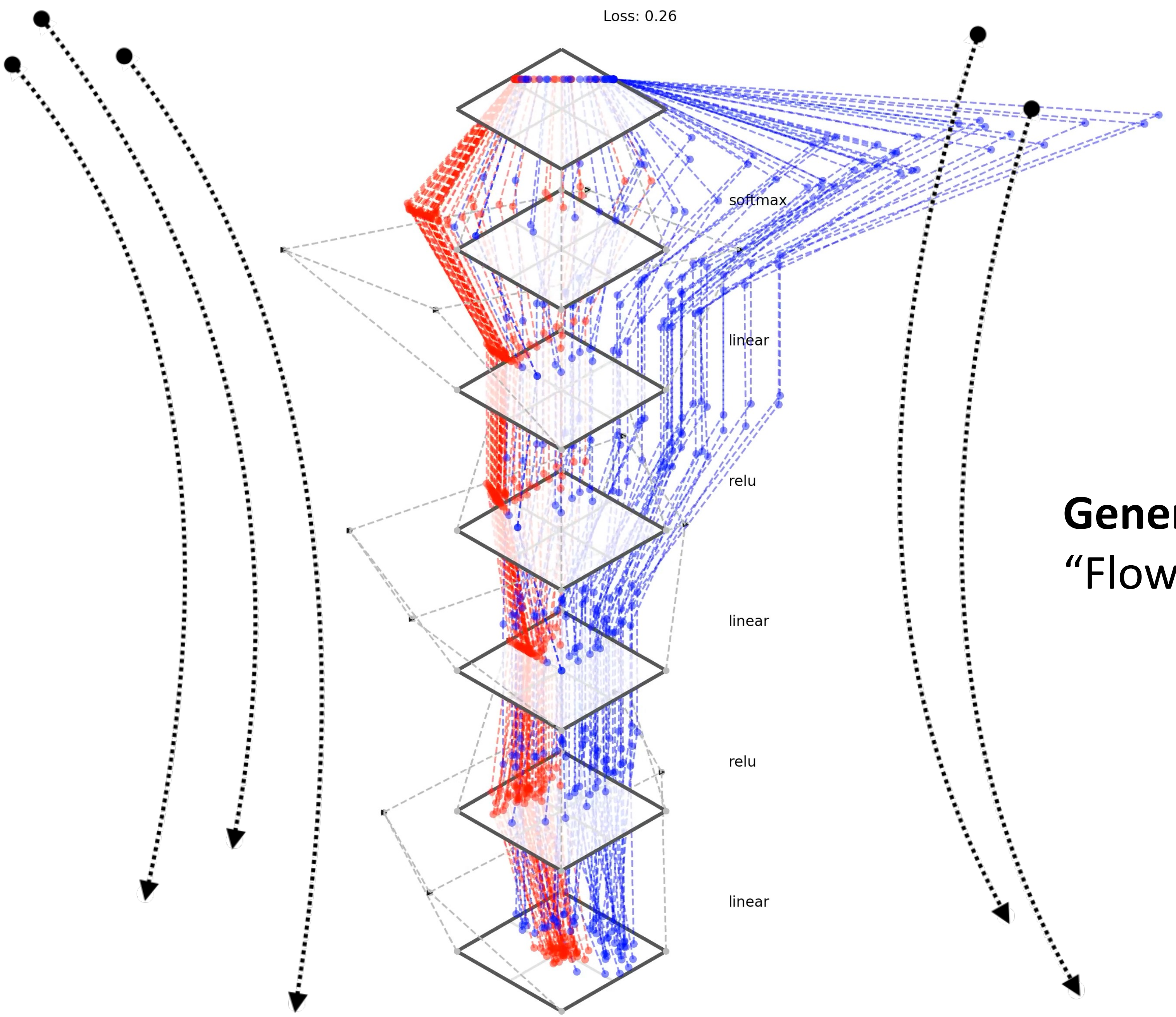
Illustration Credit: Phillip Isola

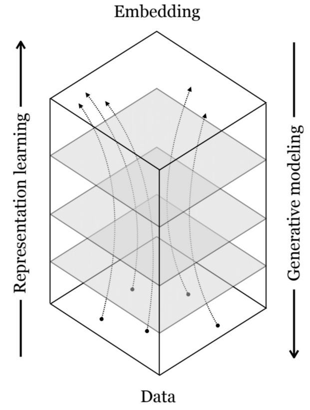




Recognition:
"Flow" from data to embeddings

Illustration Credit: Phillip Isola





Generation:
“Flow” from embeddings to data

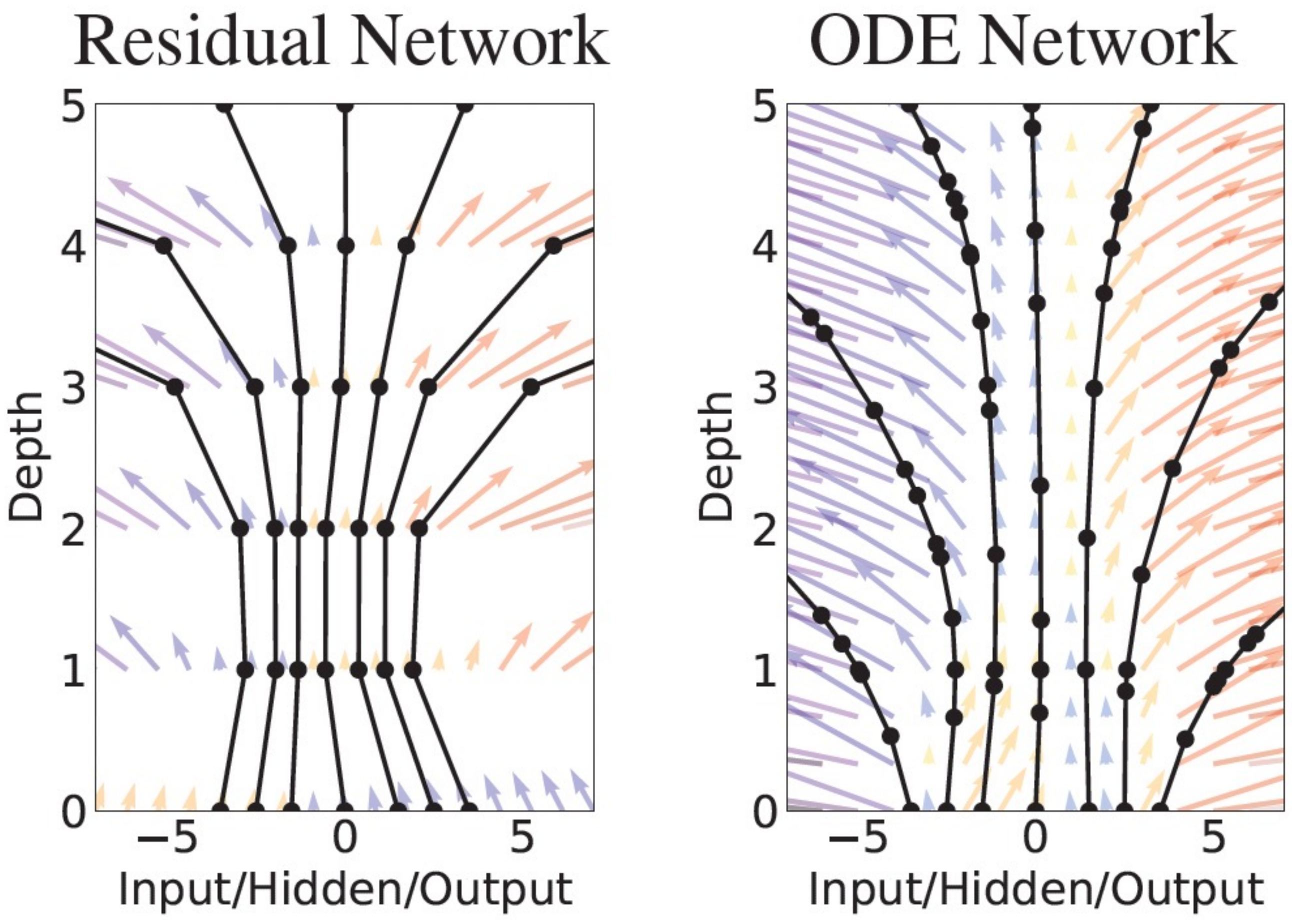
Illustration Credit: Phillip Isola

Neural ODE [Chen et al, NeurIPS 2018]

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$$

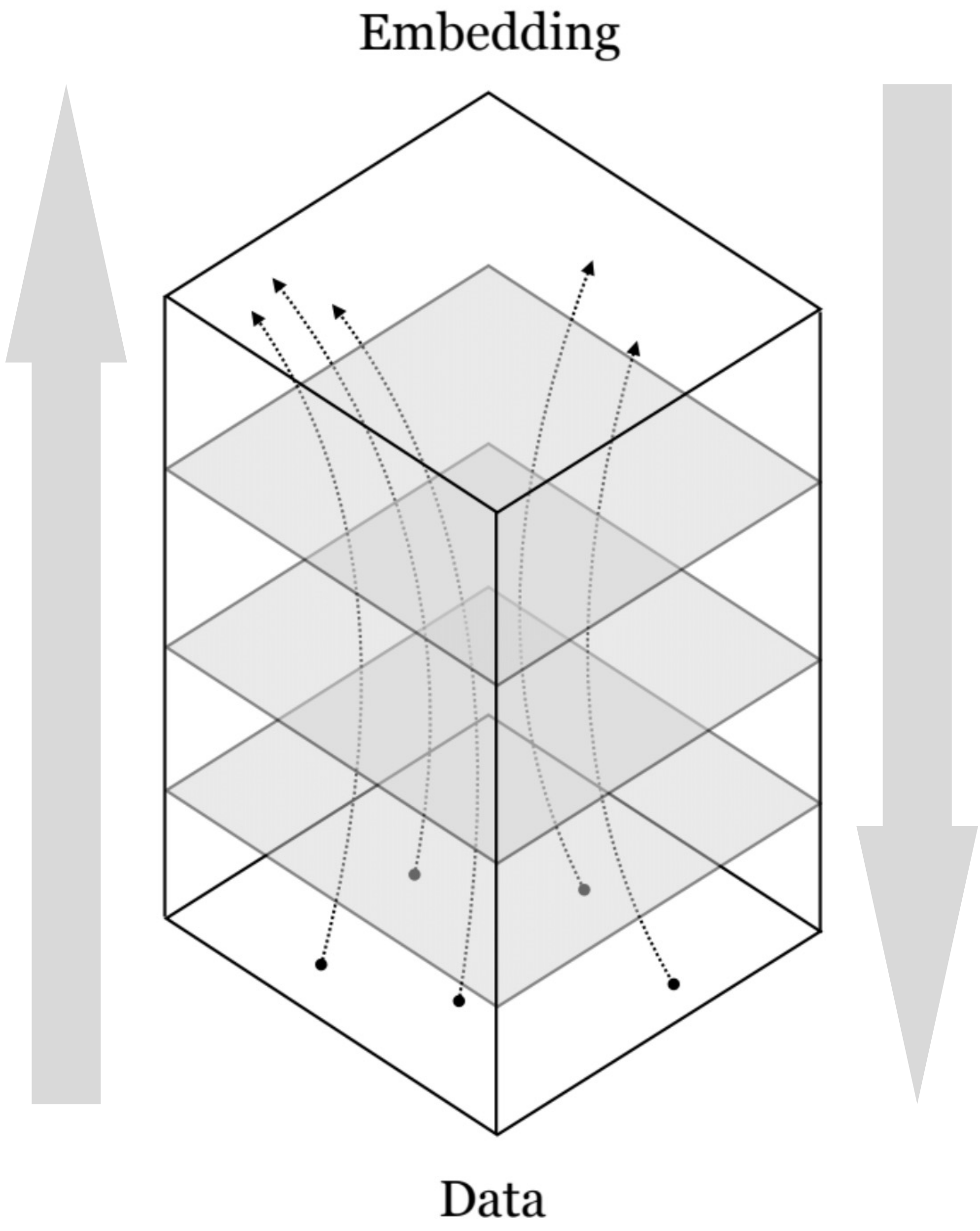
$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

- discrete time
- time-dependent parameterization



- continuous time
- time-shared parameterization
- f is often a ResNet

Recognition:
determined
data-to-label mapping



Generation:
unknown
"noise"-to-data mapping
(infinite possibilities)

Construct the mapping?

- Continuous Normalizing Flow (in Neural ODE)
- Flow Matching

Flow Matching

Flow Matching for Generative Modeling

Yaron Lipman[1,2] **Ricky T. Q. Chen**[1] **Heli Ben-Hamu**[2] **Maximilian Nickel**[1] **Matt Le**[1]
[1]Meta AI (FAIR) [2]Weizmann Institute of Science

Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow

Xingchao Liu*
University of Texas at Austin
xcliu@utexas.edu

Chengyue Gong*
University of Texas at Austin
cygong@cs.utexas.edu

Qiang Liu
University of Texas at Austin
lqiang@cs.utexas.edu

Building Normalizing Flows with Stochastic Interpolants

Michael S. Albergo
Center for Cosmology and Particle Physics
New York University
New York, NY 10003, USA
albergo@nyu.edu

Eric Vanden-Eijnden
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA
eve2@cims.nyu.edu

Flow Matching

Generative models

∪

Flows

∪

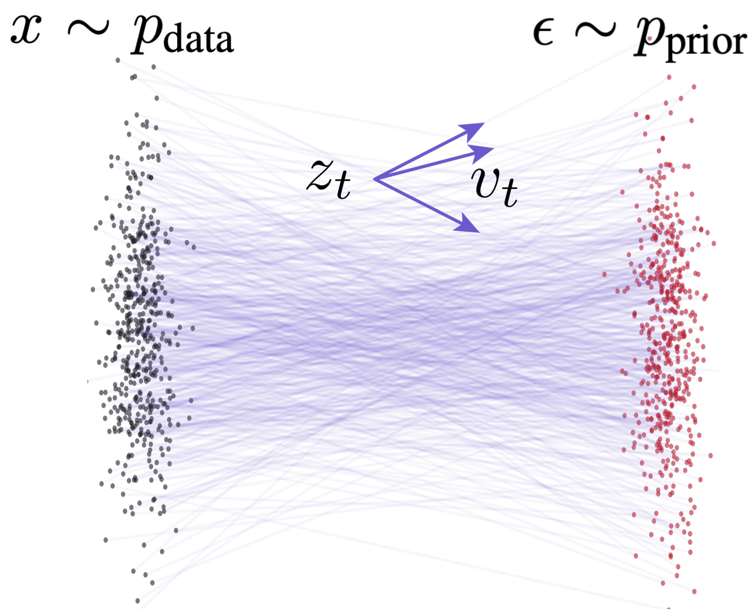
Flow Matching

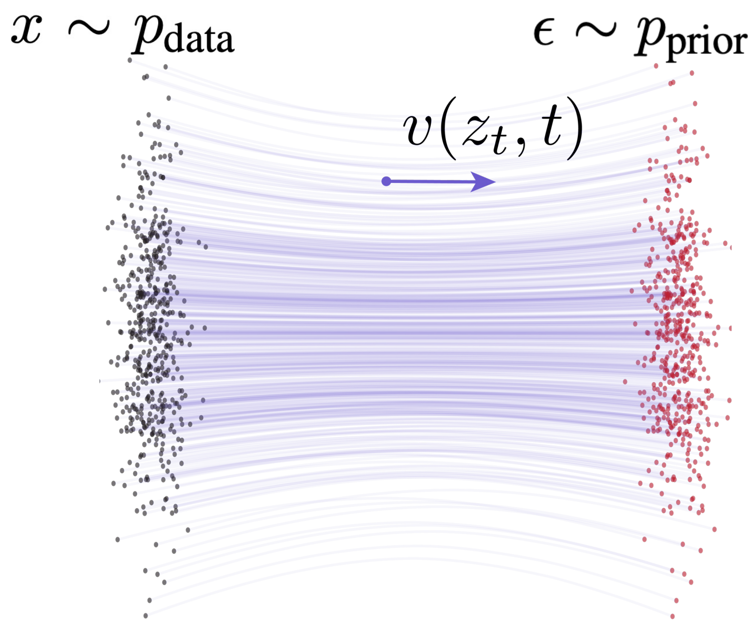
∪

Diffusion Models

Credit: Yaron Lipman, “Flow Matching: Simplifying and Generalizing Diffusion Models”

Flow Matching





$$z_t = (1 - t)x + t\epsilon$$

conditional velocity: $v_t = \epsilon - x$

marginal velocity: $v(z_t, t) \triangleq \mathbb{E}_{p_t(v_t|z_t)}[v_t]$

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}\|v_\theta(z_t, t) - v_t\|^2$$

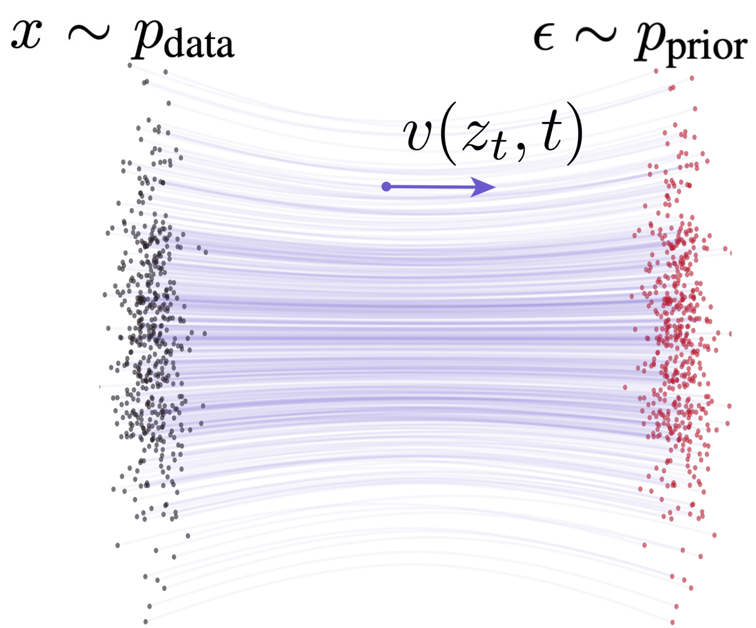
$$\mathcal{L}_{\text{FM}} = \mathbb{E}\|v_\theta(z_t, t) - v(z_t, t)\|^2$$

Illustration inspired by: Fjelde, Mathieu, Dutordoir, “An Introduction to Flow Matching”
https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html

Flow Matching

Solve ODE:

$$\frac{d}{dt}z_t = v(z_t, t)$$



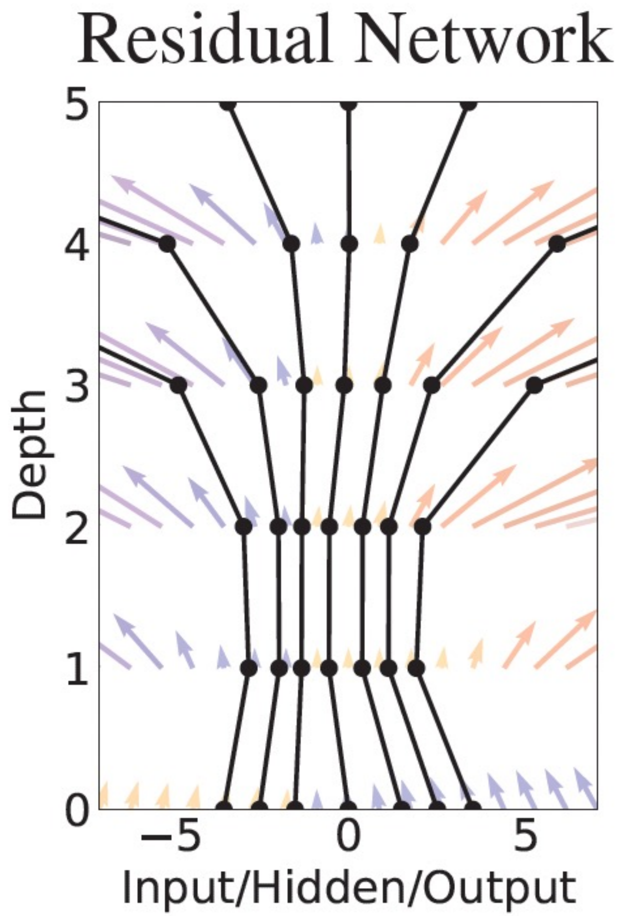
- In principle, w/ **ground-truth** field $v(z_t, t)$
- In practice, approximate by $v_\theta(z_t, t)$
- Ideally, trajectory given by **integral**: $z_r = z_t - \int_r^t v(z_\tau, \tau) d\tau$
- In reality, approximate by finite sum: $z_r = z_t + (r - t)v(z_t, t)$

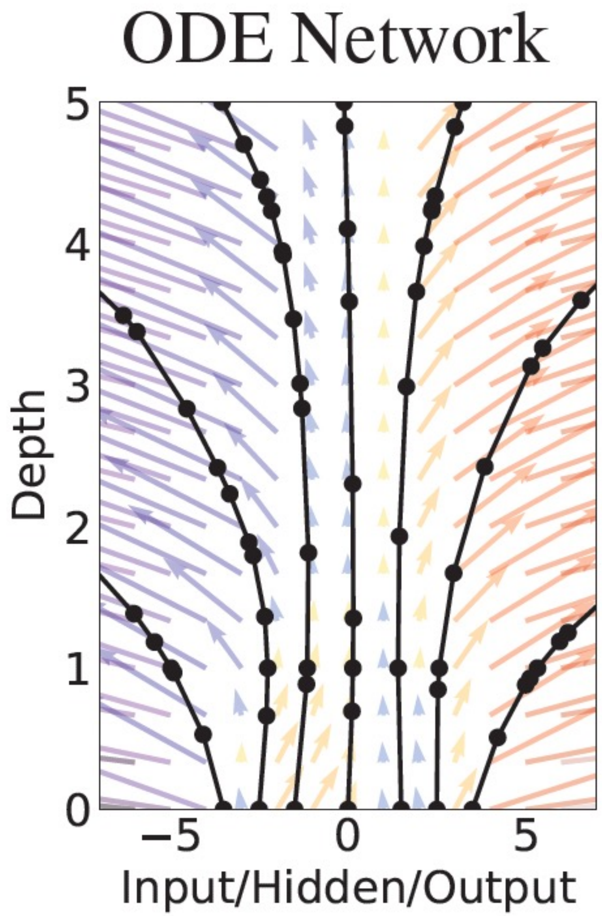
What we do:

$$z_r = z_t + (r - t)v(z_t, t)$$

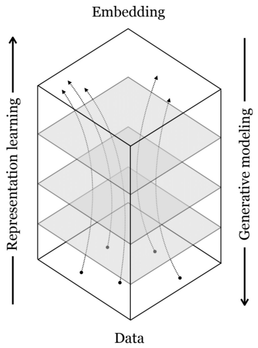
What we want:

$$\frac{d}{dt}z_t = v(z_t, t) \quad \text{or} \quad z_r = z_t - \int_r^t v(z_\tau, \tau)d\tau$$

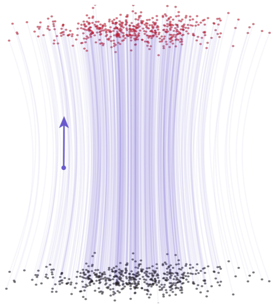




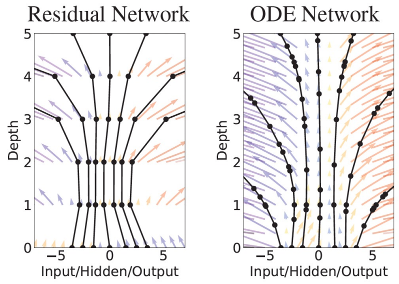
Key takeaways so Far ...



- Recognition vs. Generation: **flows** between distributions



- Flow Matching: builds **ground-truth** fields for training
  - implicit, pre-exist
  - network-independent



- We want **integral**, but in practice we do **finite sum**
  - ResNet-like discretization
  - numerical ODE solvers
- Towards **feedforward**, **end-to-end** generative modeling?

Mean Flows for One-step Generative Modeling

Zhengyang Geng[1] Mingyang Deng[2] Xingjian Bai[2]
J. Zico Kolter[1] Kaiming He[2]

[1]CMU [2]MIT

arXiv, May 2025

Average Velocity

What we want: $$z_r = z_t - \int_r^t v(z_\tau, \tau) d\tau$$

What we do: $$z_r = z_t - (t - r)\cancel{v(z_t, t)}$$

$$\Downarrow$$

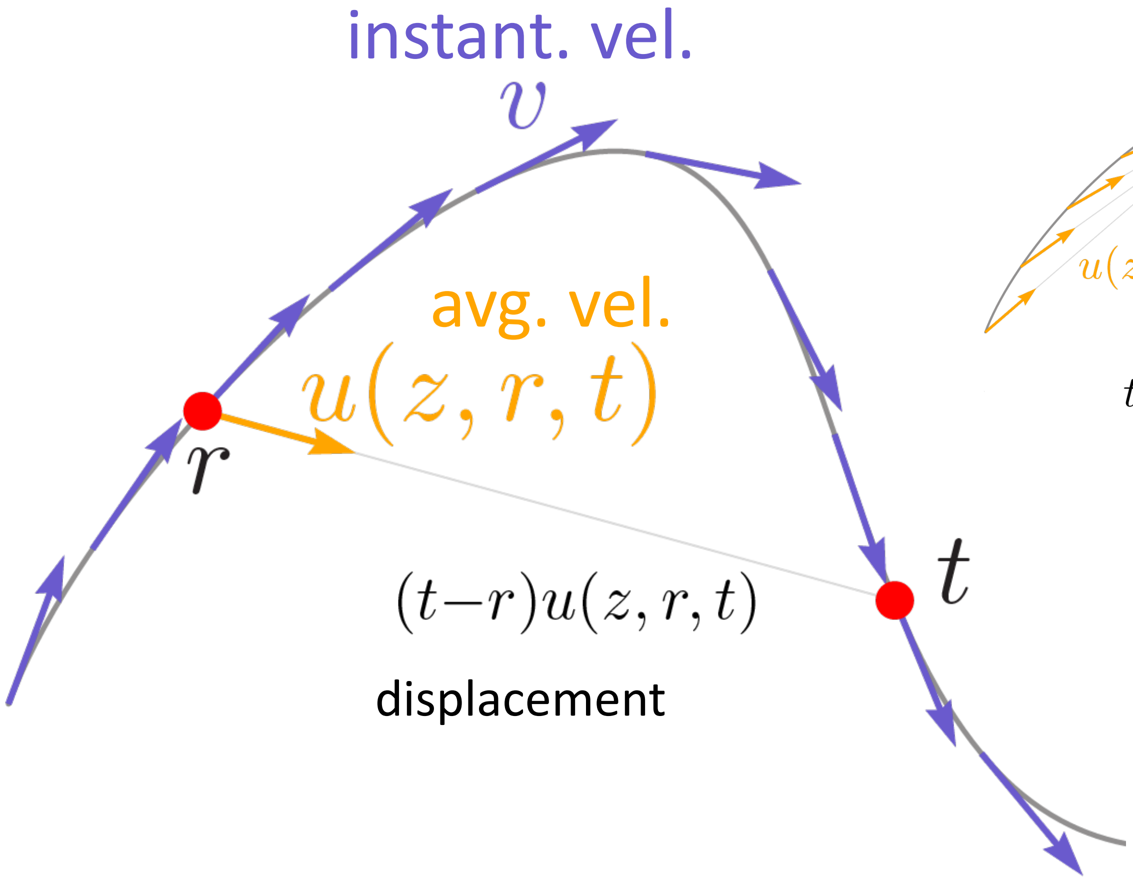
$$u(z_t, r, t) \triangleq \frac{1}{t - r} \int_r^t v(z_\tau, \tau) d\tau$$

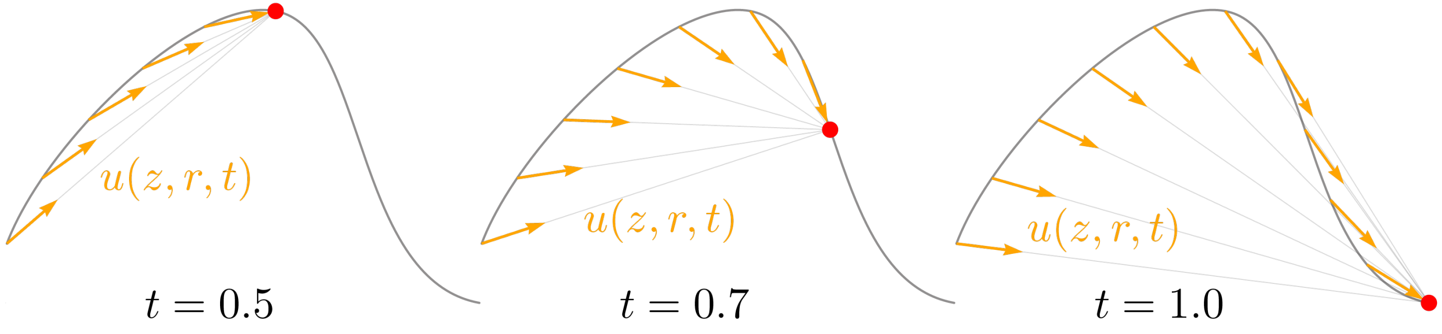
u: average velocity

v: instantaneous velocity

Average Velocity

$$u(z_t, r, t) \triangleq \frac{1}{t-r}\int_r^t v(z_\tau, \tau)d\tau$$





Properties:

- condition on **two time** variables
- network **independent**
- **ground-truth** field that pre-exists

The MeanFlow Identity

- **Integral** is intractable. **Differentiate** it instead.

$$u(z_t, r, t) \triangleq \frac{1}{t-r}\int_r^t v(z_\tau, \tau)d\tau$$

$$\Rightarrow \quad (t-r)u(z_t, r, t) = \int_r^t v(z_\tau, \tau)d\tau$$

differentiate wrt t $\Rightarrow$

$$\frac{d}{dt}(t-r)u(z_t, r, t) = \frac{d}{dt}\int_r^t v(z_\tau, \tau)d\tau$$

$\Rightarrow$ lhs: product rule

$$u(z_t, r, t) + (t-r)\frac{d}{dt}u(z_t, r, t) = v(z_t, t)$$

rhs: fundamental theorem of calculus

$$\Rightarrow \quad u(z_t, r, t) = v(z_t, t) - (t-r)\frac{d}{dt}u(z_t, r, t)$$

MeanFlow Identity

The MeanFlow Identity

$$u(z_t, r, t) = v(z_t, t) - (t - r)\frac{d}{dt}u(z_t, r, t)$$

avg. vel.

instant. vel.

two time variables

t-derivative

Computing the time derivative

$$\frac{d}{dt}u(z_t, r, t) = \partial_z u \frac{dz_t}{dt} + \partial_r u \frac{dr}{dt} + \partial_t u \frac{dt}{dt}$$

$= 0$ $= 1$

$$= \left[\partial_z u, \partial_r u, \partial_t u\right] \begin{bmatrix} v(z_t, t) \\ 0 \\ 1 \end{bmatrix}$$

Jacobian matrix

The ODE we are solving

$$\frac{d}{dt}z_t = v(z_t, t)$$

- Jacobian-vector product (**JVP**): `jvp(fn, (z, r, t), (v, 0, 1))`
- *c.f.* vector-Jacobian product (**VJP**): "backpropagation"

See: https://docs.jax.dev/en/latest/notebooks/autodiff_cookbook.html#how-it-s-made-two-foundational-autodiff-functions

Training MeanFlow Models

$$u(z_t, r, t) = v(z_t, t) - (t - r)\frac{d}{dt}u(z_t, r, t)$$

avg. vel. instant. vel. t-derivative

No neural net up till now; only about the **ground-truth** field

$$\mathcal{L}(\theta) = \mathbb{E}\left\|u_\theta(z_t, r, t) - \mathrm{sg}(u_{\mathrm{tgt}})\right\|_2^2$$

parameterize u directly target w/ stopgrad

$$u_{\mathrm{tgt}} = v(z_t, t) - (t - r)\left(v(z_t, t)\partial_z u_\theta + \partial_t u_\theta\right)$$

instant. vel. computed by JVP

Training MeanFlow Models

- if u_θ has **zero loss**, it satisfies the **MeanFlow Identity**
- **no integral**; only derivatives. (proven equivalent; see paper)
- **stopgrad** prevents **higher-order** gradients
- a **single-time** function u_θ is **insufficient**

$$\mathcal{L}(\theta) = \mathbb{E}\left\| \underbrace{u_\theta(z_t, r, t)}_{\text{parameterize } u \text{ directly}} - \underbrace{\mathrm{sg}(u_{\text{tgt}})}_{\text{target w/ stopgrad}} \right\|_2^2$$

$$u_{\text{tgt}} = \underbrace{v(z_t, t)}_{\text{instant. vel.}} - (t - r)\underbrace{\left(v(z_t, t)\partial_z u_\theta + \partial_t u_\theta\right)}_{\text{computed by JVP}}$$

Training MeanFlow Models

- **marginal** velocity is not explicitly accessible
- use **conditional** velocity instead (as in Flow Matching)

$$\mathcal{L}(\theta) = \mathbb{E}\left\| u_\theta(z_t, r, t) - \mathrm{sg}(u_{\mathrm{tgt}}) \right\|_2^2$$

parameterize u directly

target w/ stopgrad

$$u_{\mathrm{tgt}} = v(z_t, t) - (t - r)\left(v(z_t, t)\partial_z u_\theta + \partial_t u_\theta\right)$$

$$v_t = \epsilon - x$$

CFG can be handled similarly (see paper):

$$\tilde{v}_t \triangleq \omega\, v_t + (1 - \omega)\, u_\theta(z_t, t, t)$$

Algorithm 1 MeanFlow: Training.

Note: in PyTorch and JAX, jvp returns the function output and JVP.

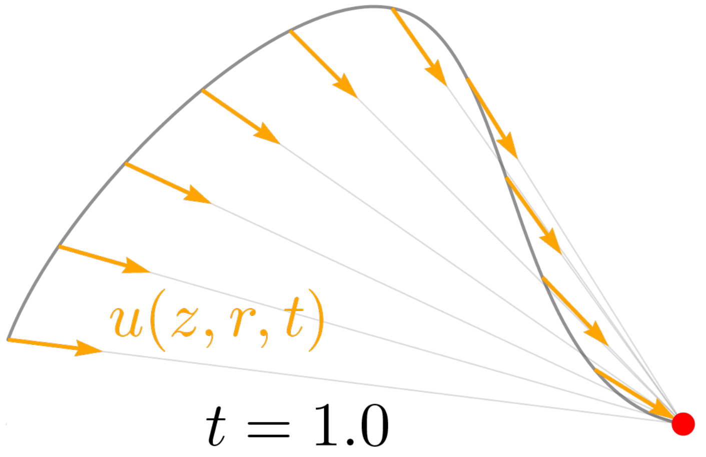
```
# fn(z, r, t): function to predict u
# x: training batch

t, r = sample_t_r()
e = randn_like(x)

z = (1 - t) * x + t * e
v = e - x

u, dudt = jvp(fn, (z, r, t), (v, 0, 1))

u_tgt = v - (t - r) * dudt
error = u - stopgrad(u_tgt)

loss = metric(error)
```

the main changes over Flow Matching

Sampling with MeanFlow

What we want: $$z_r = z_t - \int_r^t v(z_\tau, \tau) d\tau$$

What we do: $$z_r = z_t - (t - r)\underbrace{u(z_t, r, t)}_{\text{avg. vel.}}$$

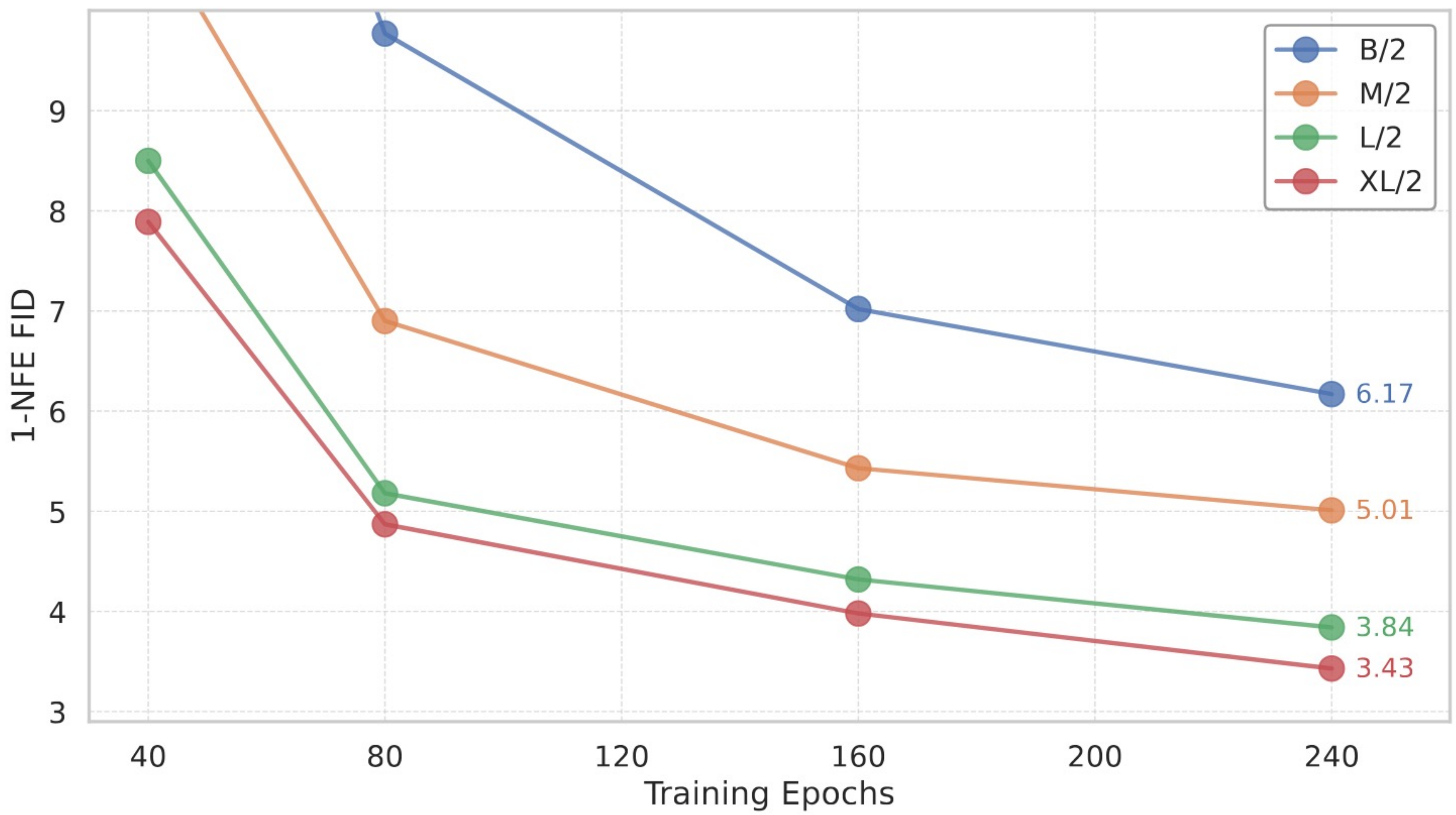


Algorithm 2 MeanFlow: 1-step Sampling

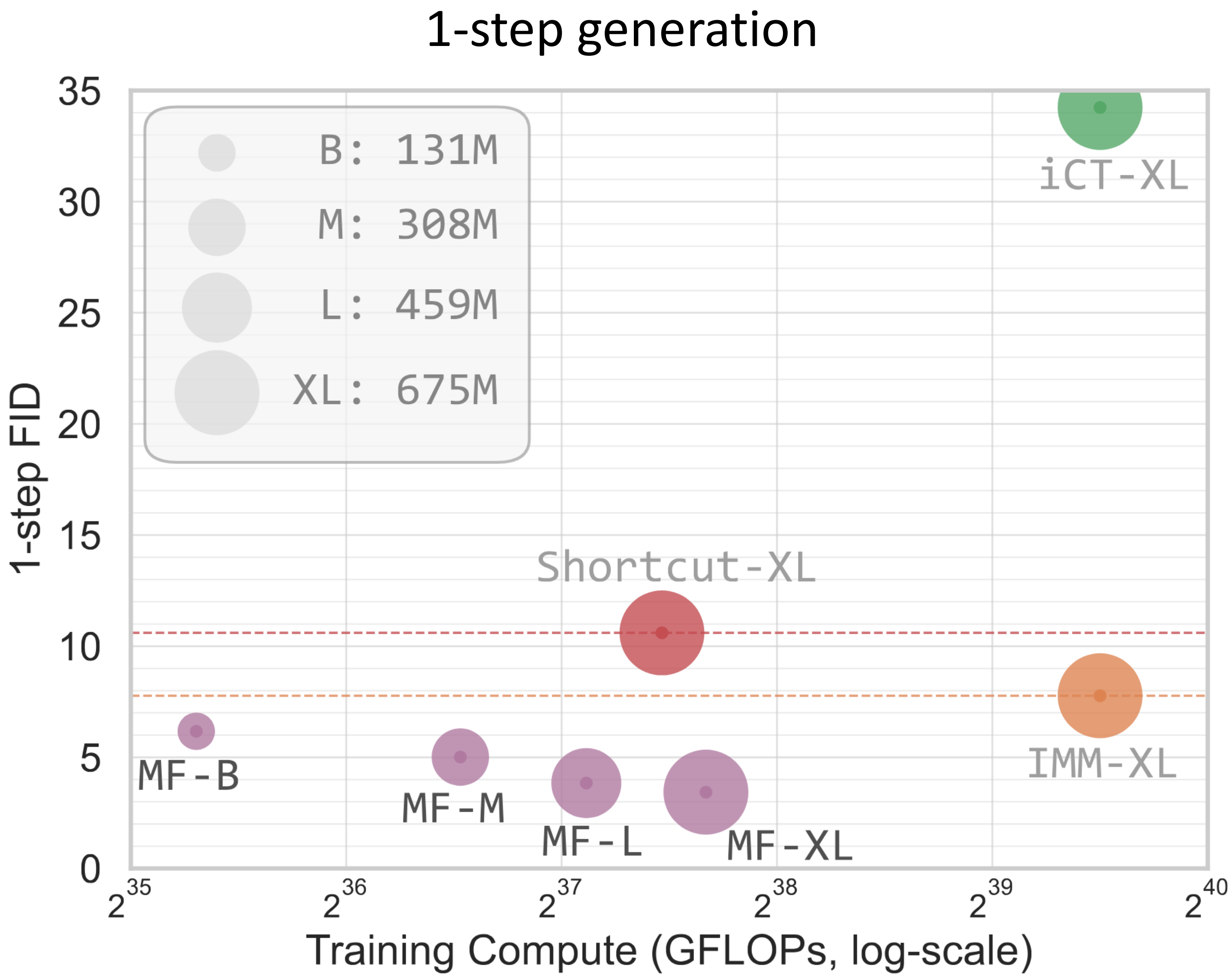
```
e = randn(x_shape)
x = e - fn(e, r=0, t=1)
```

Results: ImageNet 256x256

1-NFE (number of function evaluation) generation



Results: ImageNet 256x256



Results: ImageNet 256x256

method	params	NFE	FID
1-NFE diffusion/flow from scratch			
iCT-XL/2 [44]†	675M	1	34.24
Shortcut-XL/2 [13]	675M	1	10.60
MeanFlow-B/2	131M	1	6.17
MeanFlow-M/2	308M	1	5.01
MeanFlow-L/2	459M	1	3.84
MeanFlow-XL/2	676M	1	**3.43**

1-NFE: 70% improvement

Results: ImageNet 256x256

method	params	NFE	FID
1-NFE diffusion/flow from scratch			
iCT-XL/2 [44]†	675M	1	34.24
Shortcut-XL/2 [13]	675M	1	10.60
MeanFlow-B/2	131M	1	6.17
MeanFlow-M/2	308M	1	5.01
MeanFlow-L/2	459M	1	3.84
MeanFlow-XL/2	676M	1	**3.43**
2-NFE diffusion/flow from scratch			
iCT-XL/2 [44]†	675M	2	20.30
iMM-XL/2 [53]	675M	1×2	7.77
MeanFlow-XL/2	676M	2	2.93
MeanFlow-XL/2+	676M	2	**2.20**

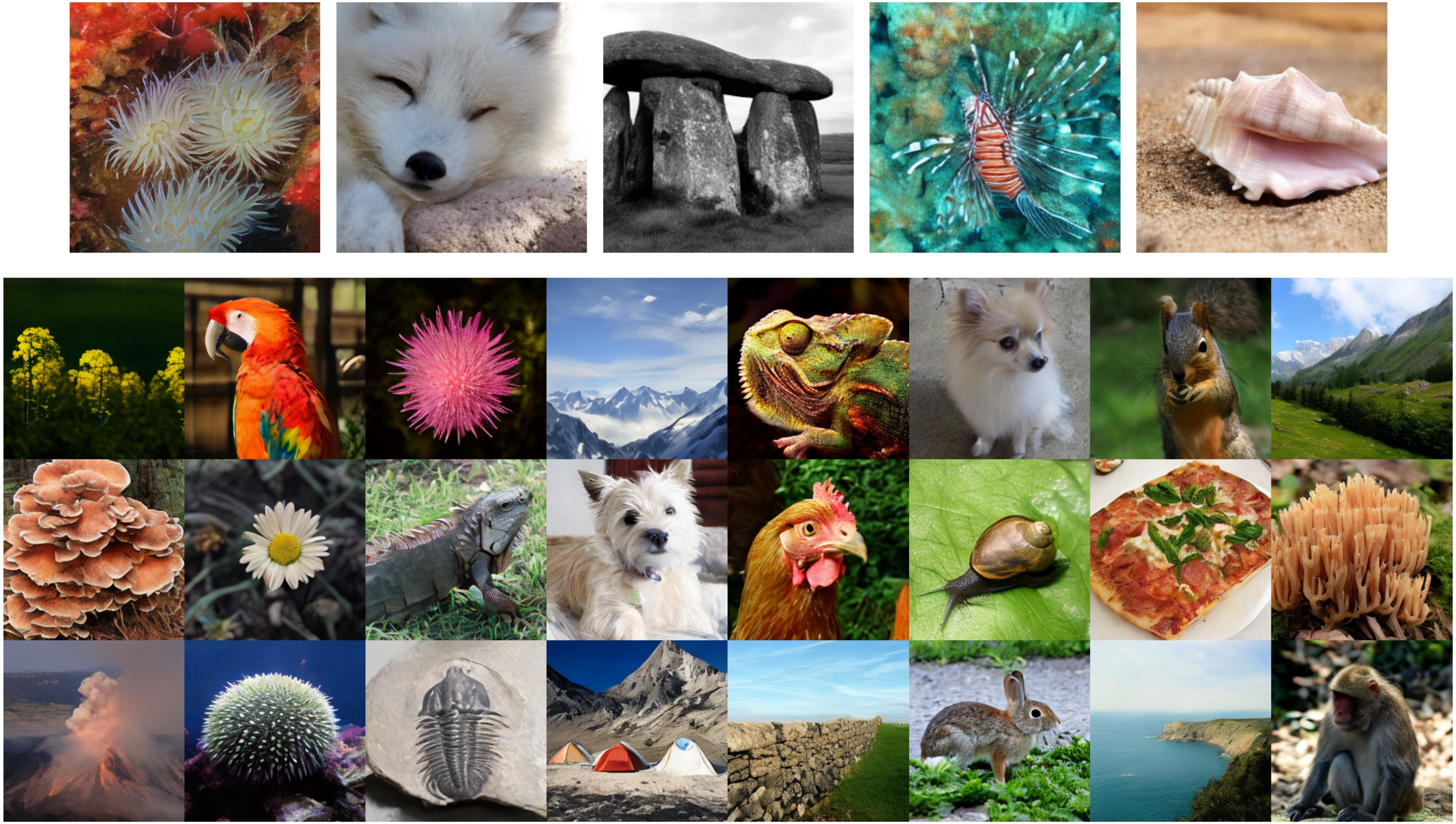
2-NFE: 70% improvement

Results: ImageNet 256x256

method	params	NFE	FID
1-NFE diffusion/flow from scratch			
iCT-XL/2 [44]†	675M	1	34.24
Shortcut-XL/2 [13]	675M	1	10.60
MeanFlow-B/2	131M	1	6.17
MeanFlow-M/2	308M	1	5.01
MeanFlow-L/2	459M	1	3.84
MeanFlow-XL/2	676M	1	**3.43**
2-NFE diffusion/flow from scratch			
iCT-XL/2 [44]†	675M	2	20.30
iMM-XL/2 [53]	675M	1×2	7.77
MeanFlow-XL/2	676M	2	2.93
MeanFlow-XL/2+	676M	2	**2.20**

method	params	NFE	FID
GANs			
BigGAN [5]	112M	1	6.95
GigaGAN [22]	569M	1	3.45
StyleGAN-XL [41]	166M	1	2.30
autoregressive/masking			
AR w/ VQGAN [10]	227M	1024	26.52
MaskGIT [6]	227M	8	6.18
VAR-*d*30 [48]	2B	10×2	1.92
MAR-H [28]	943M	256×2	1.55
diffusion/flow			
ADM [8]	554M	250×2	10.94
LDM-4-G [38]	400M	250×2	3.60
SimDiff [21]	2B	512×2	2.77
DiT-XL/2 [35]	675M	250×2	2.27
SiT-XL/2 [34]	675M	250×2	2.06
SiT-XL/2+REPA [52]	675M	250×2	**1.42**

narrows gap w/
multi-step counterparts



Qualitative result, 1-NFE generation (FID 3.43)

The Community Effort ...

- **Consistency Models**
  - Consistency Models (**CM**) [Song+ 2023]
  - improved Consistency Training (**iCT**) [Song & Dhariwal 2024]
  - Easy Consistency Training (**ECT**) [Geng+ 2024]
  - simple/stable/scalable Consistency Models (**sCM**) [Lu & Song 2024]
- **Two-time-variable Models**
  - Consistency Trajectory Models (**CTM**) [Kim+ 2023]
  - Flow Map Matching [Boffi+ 2024]
  - Shortcut Models [Frans+ 2024]
  - Inductive Moment Matching [Zhou+ 2025]
- **Revisiting Normalizing Flows**
  - TarFlow [Zhai+ ’24]
- ...

Looking ahead...

- Are we still in the **pre-AlexNet** era of generative modeling?

- MeanFlow is still driven by **iterative** Flow Matching (and diffusion)

- MeanFlow network plays two roles:
  - **construct** noise-to-data trajectories (pre-exist, but implicit)
  - **summarize** the fields via coarsening

- What’s a good formulation for **end-to-end** generative modeling?