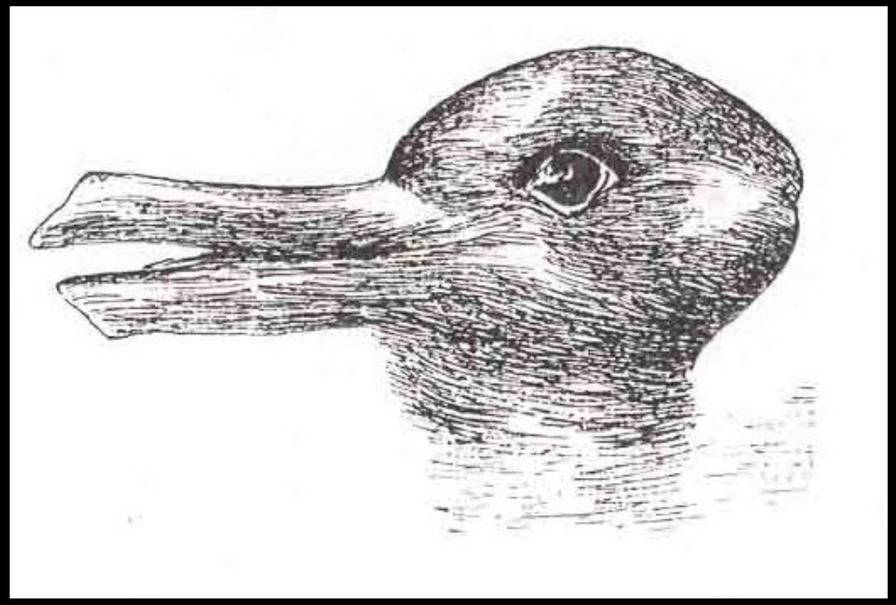
Advanced Computer Vision: Representation Learning, Generative Models

MLMI17

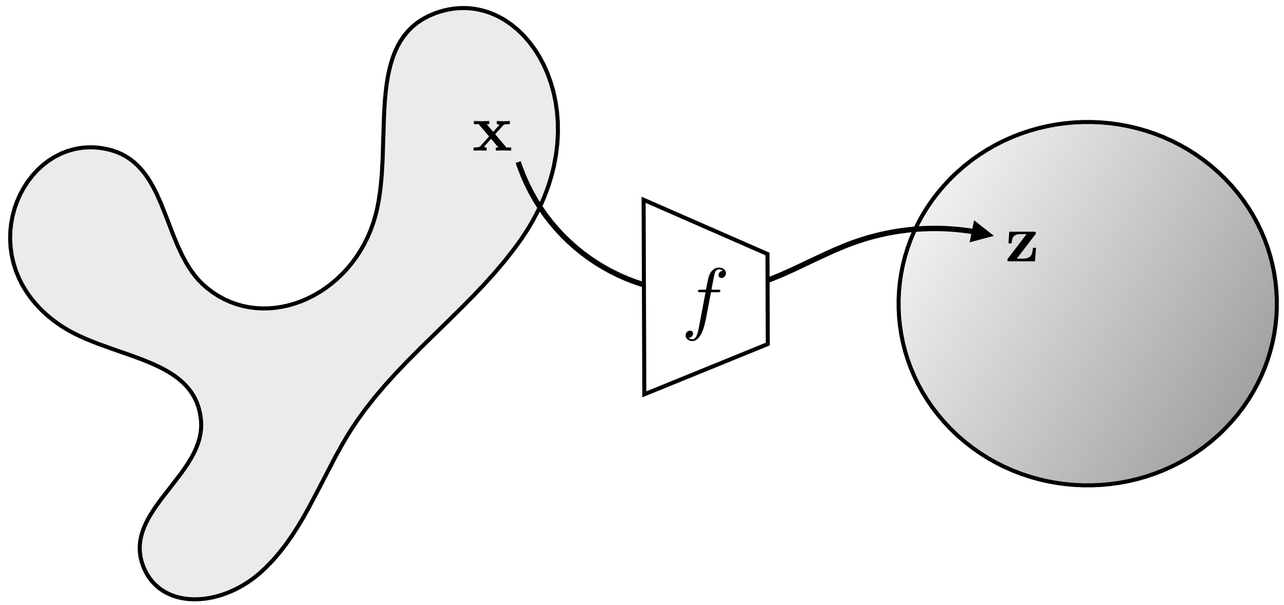
Ayush Tewari

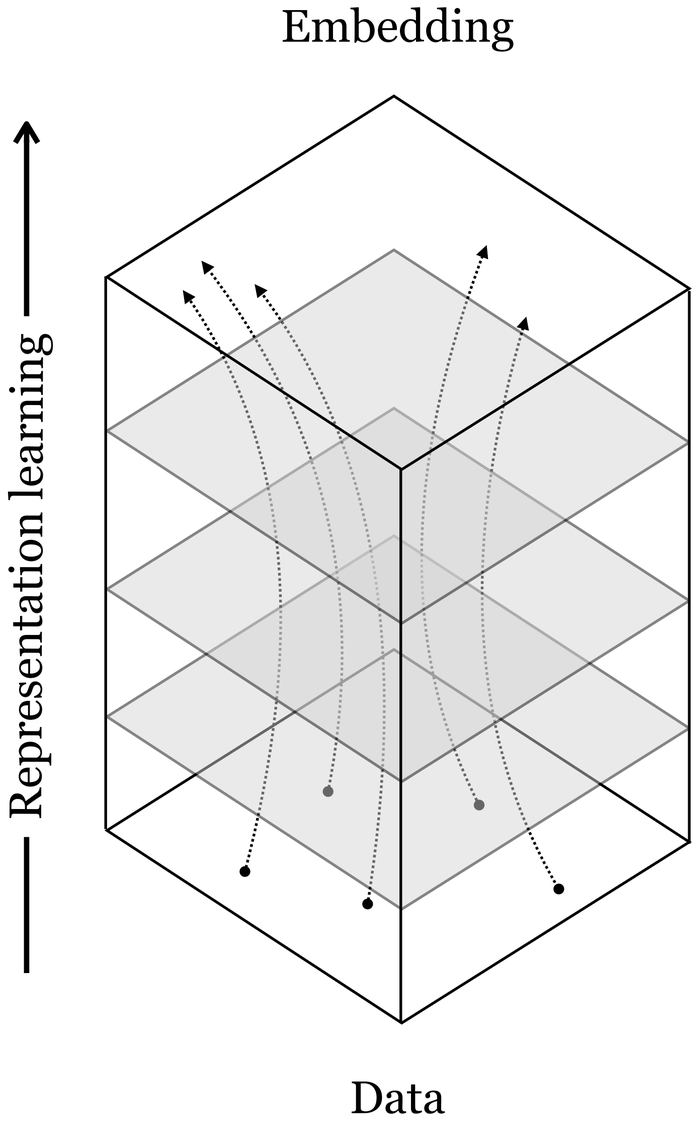


UNIVERSITY OF CAMBRIDGE

What is representation learning?

- A general recipe:
  - Collect large quantities of unlabelled data
  - Define an auxiliary task
  - Train a model so solve this task
  - **Pray**



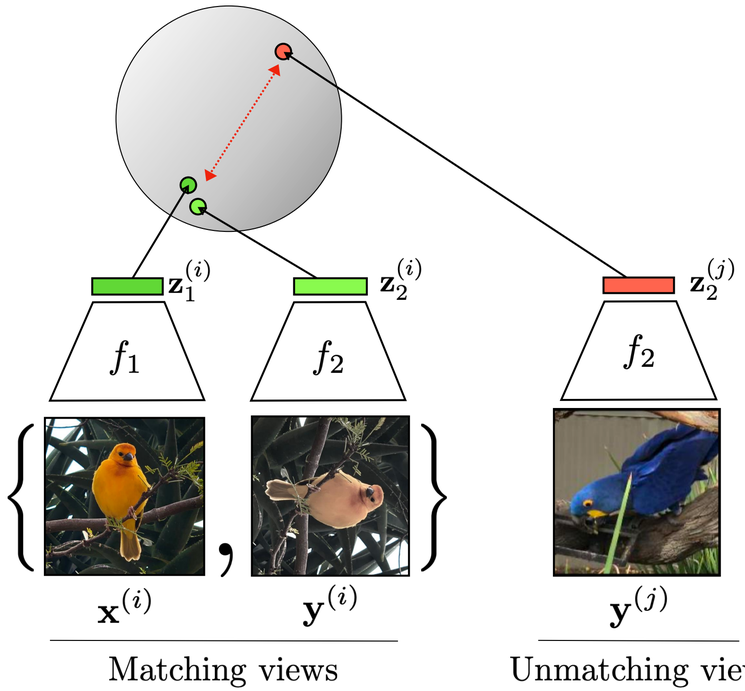


Three classes of self-supervision

- Generative
- Contrastive
- Distillation

Contrastive

- Do we need to preserve all information in the image?
  - What if I do not care about low-level details?
- Contrastive learning!
  - Augmentations of an image should have similar representations



Data

$\{\mathbf{x}^{(i)}\}_{i=1}^{N}, T \rightarrow$

Constrastive learning (transformations)

Objective

$\sum_{i,j} D(f(T(\mathbf{x}^{(i)})), f(\mathbf{x}^{(i)}))$

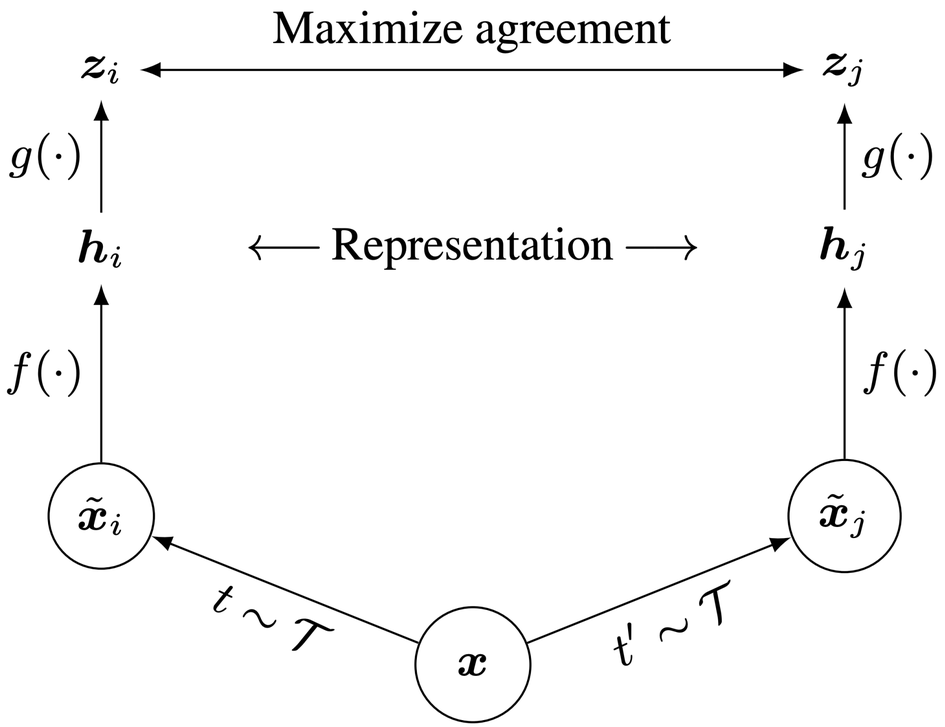
$-D(f(\mathbf{x}^{(i)}), f(\mathbf{x}^{(j)}))$

Hypothesis space

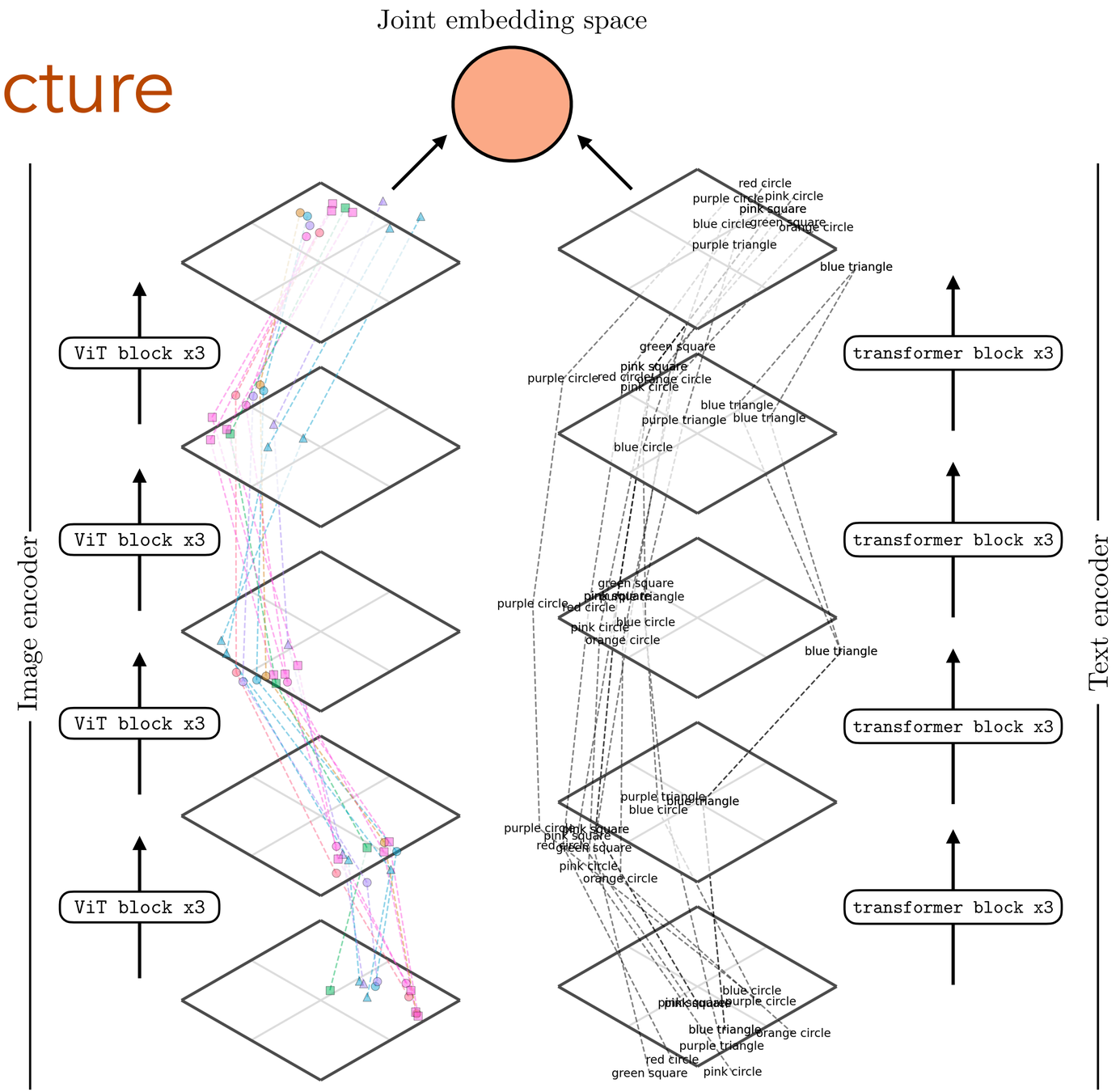
$f : \mathbb{R}^N \rightarrow \mathbb{R}^M$

$\rightarrow f$

SimCLR - Architecture



CLIP - Architecture



Three classes of self-supervision

- Generative
- Contrastive
- **Distillation**

Distillation-based Self-Supervision

- Starts from knowledge distillation

Distilling the Knowledge in a Neural Network

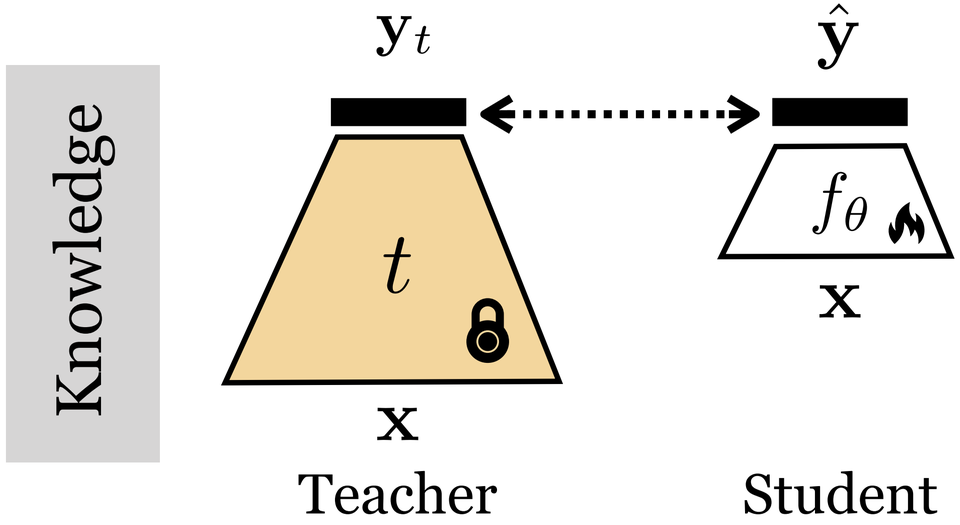
Geoffrey Hinton[*†]
Google Inc.
Mountain View
geoffhinton@google.com

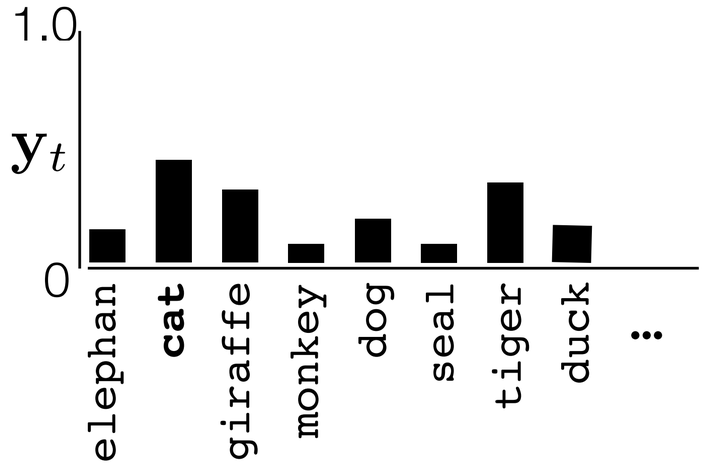
Oriol Vinyals[†]
Google Inc.
Mountain View
vinyals@google.com

Jeff Dean
Google Inc.
Mountain View
jeff@google.com

Many insects have a larval form that is optimized for extracting energy and nutrients from the environment and a completely different adult form that is optimized for the very different requirements of traveling and reproduction. In large-scale machine learning, we typically use very similar models for the training stage and the deployment stage despite their very different requirements: For tasks like speech and object recognition, training must extract structure from very large, highly redundant datasets but it does not need to operate in real time and it can use a huge amount of computation. Deployment to a large number of users, however, has much more stringent requirements on latency and computational resources. The analogy with insects suggests that we should be willing to train very cumbersome models if that makes it easier to extract structure from the data. The cumbersome model could be an ensemble of separately trained models or a single very large model trained with a very strong regularizer such as dropout [9]. Once the cumbersome model has been trained, we can then use a different kind of training, which we call "distillation" to transfer the knowledge from the cumbersome model to a small model that is more suitable for deployment. A version of this

Knowledge Distillation





$$\{\mathbf{x}^{(i)}\}_{i=1}^{N}$$
$$t : \mathcal{X} \to \triangle^{K-1}$$

$\longrightarrow$

Knowledge Distillation

Cross-entropy Objective

$$H(t(\mathbf{x}), f_\theta(\mathbf{x}))$$

Hypothesis space

$$f_\theta : \mathcal{X} \to \triangle^{K-1}$$

$\longrightarrow$ f_θ

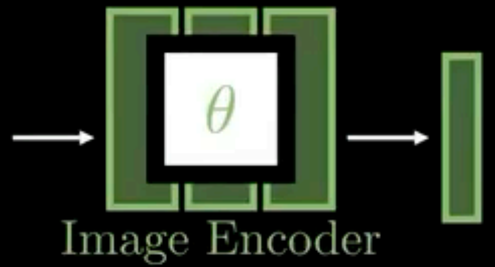
Knowledge **di**stillation *with **no** labels* (DINO)

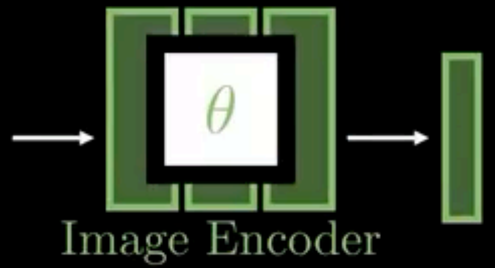
Unsupervised Learning of Visual Features by Contrasting Cluster Assignments

Mathilde Caron[1,2] **Ishan Misra**[2] **Julien Mairal**[1]

Priya Goyal[2] **Piotr Bojanowski**[2] **Armand Joulin**[2]

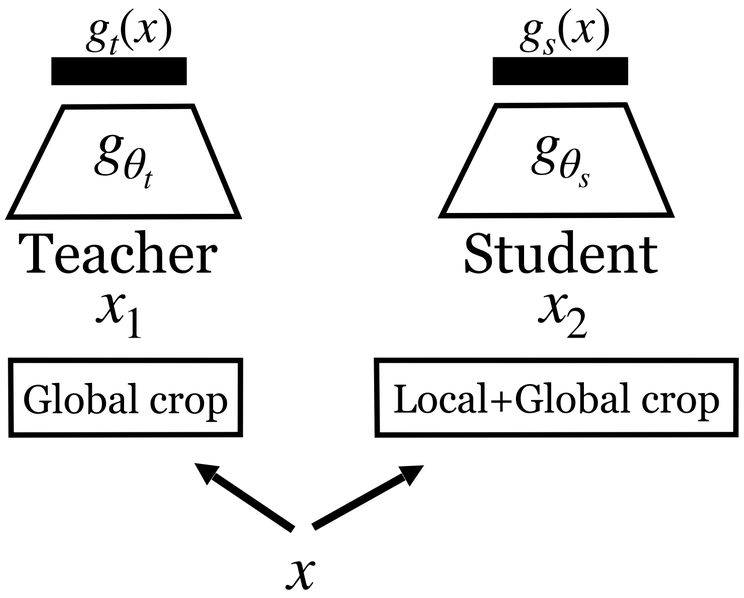
[1] Inria* [2] Facebook AI Research


θ
Image Encoder


θ
Image Encoder

Knowledge **di**stillation *with **no** labels* (DINO)

- Teacher and student have identical architecture
- Crops
  - Two global (>50%) and several local views (<50%)
  - Teacher only sees global views

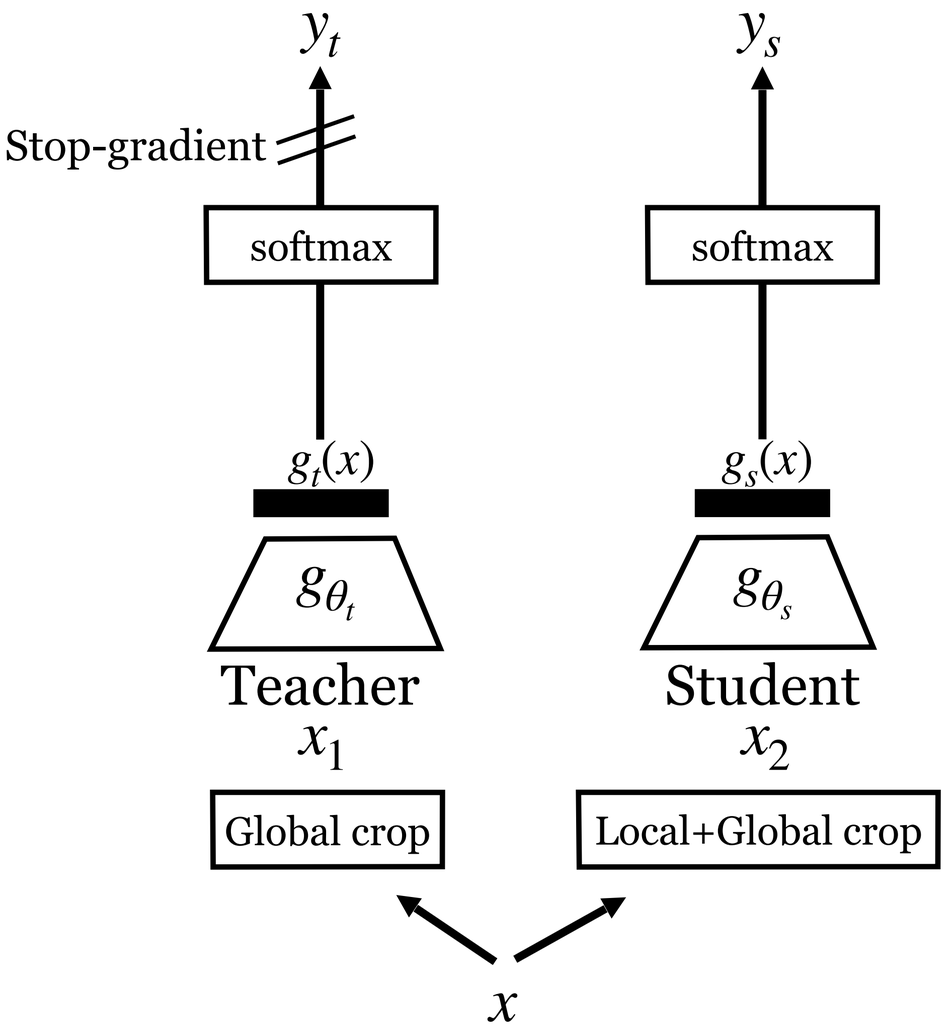



Orig. image


Global crop

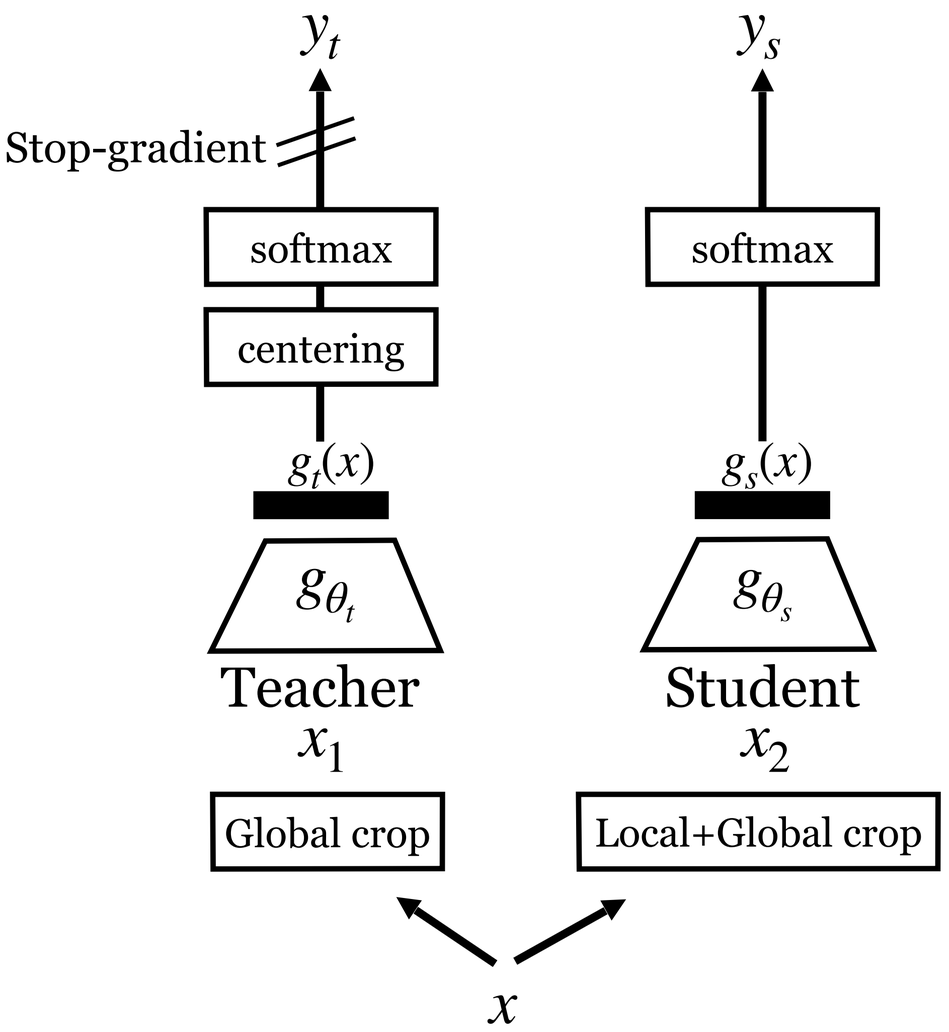

Local crop

Knowledge **di**stillation *with **no** labels* (DINO)



- Cross-entropy loss $H(y_t, y_s)$
  - Softmax applied before loss
- We do not have a teacher?!
  - Teacher is an exponential moving average (EMA) of the student $\theta_t = \lambda\theta_t + (1-\lambda)\theta_s$

Knowledge **di**stillation *with **no** labels* (DINO)



- What if output labels all collapse?
- Centering (all images are in the same class)

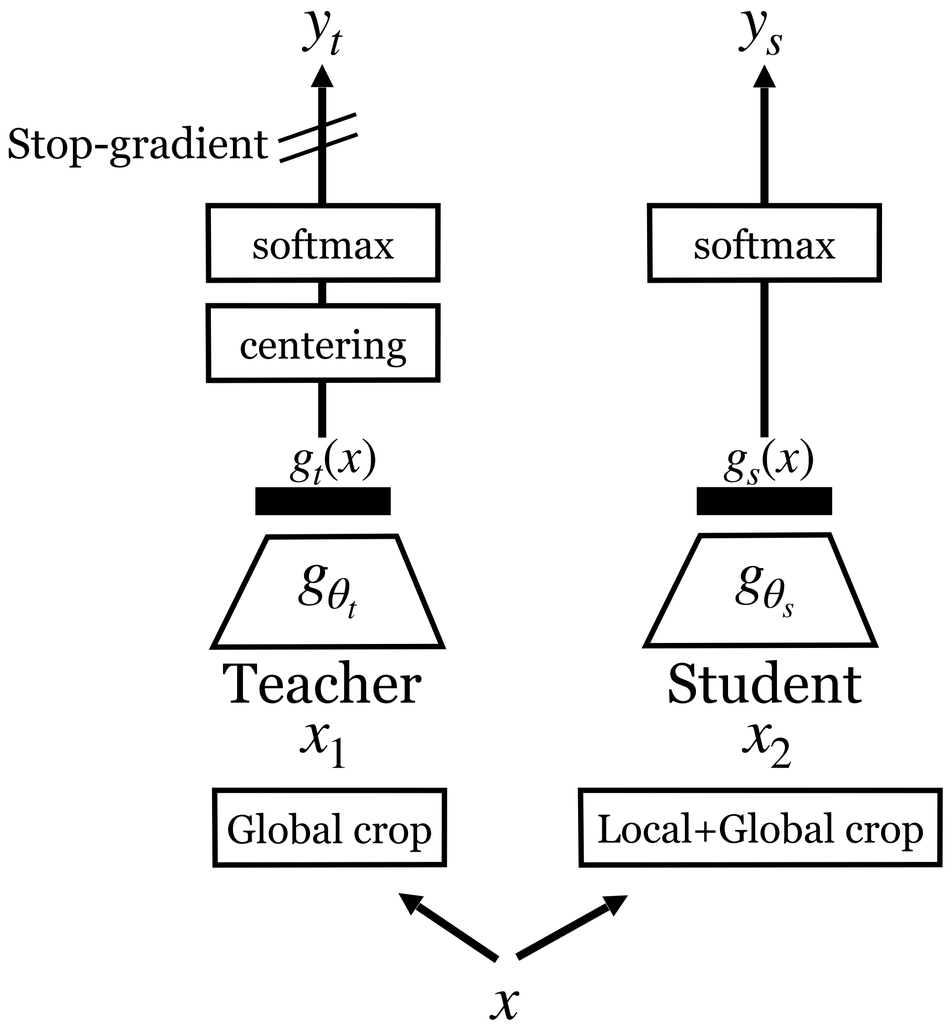
$$g_t(x) = g_t(x) - c$$

$$c = mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_t(x)$$

- Sharpening (all uniform probability)
  - low temperature in softmax

$$\frac{exp(y_t^{(i)}/\tau)}{\sum_k exp(y_t^{(k)}/\tau)}$$

Knowledge **di**stillation *with* ***no*** *labels* (DINO)



Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Knowledge **di**stillation *with **no** labels* (DINO)

Why does it learn local information?

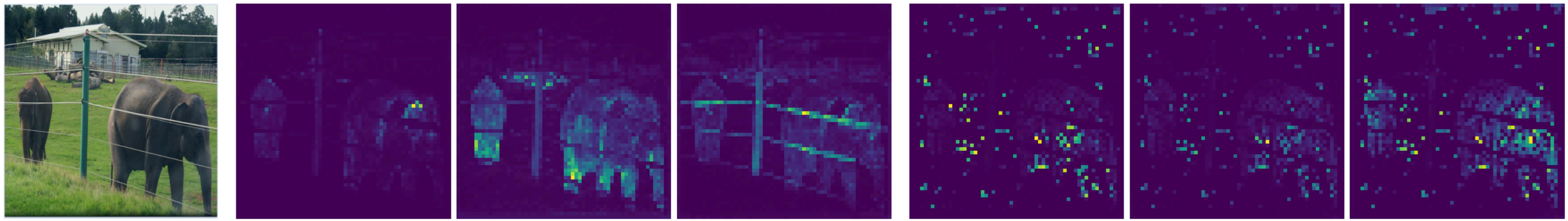
Knowledge **di**stillation *with **no** labels* (DINO)

Knowledge **di**stillation *with **no** labels* (DINO)

Knowledge **di**stillation *with **no** labels* (DINO)

DINO

Supervised



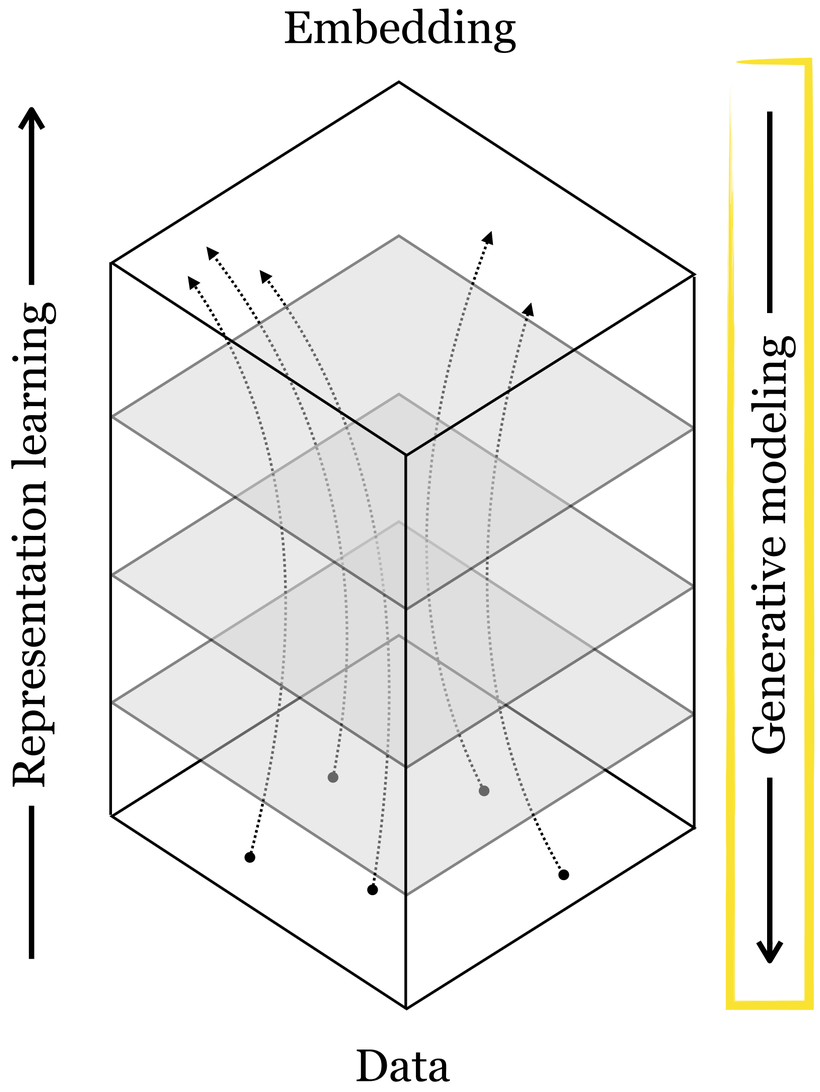
Three classes of self-supervision

- Generative
- Contrastive
- Distillation

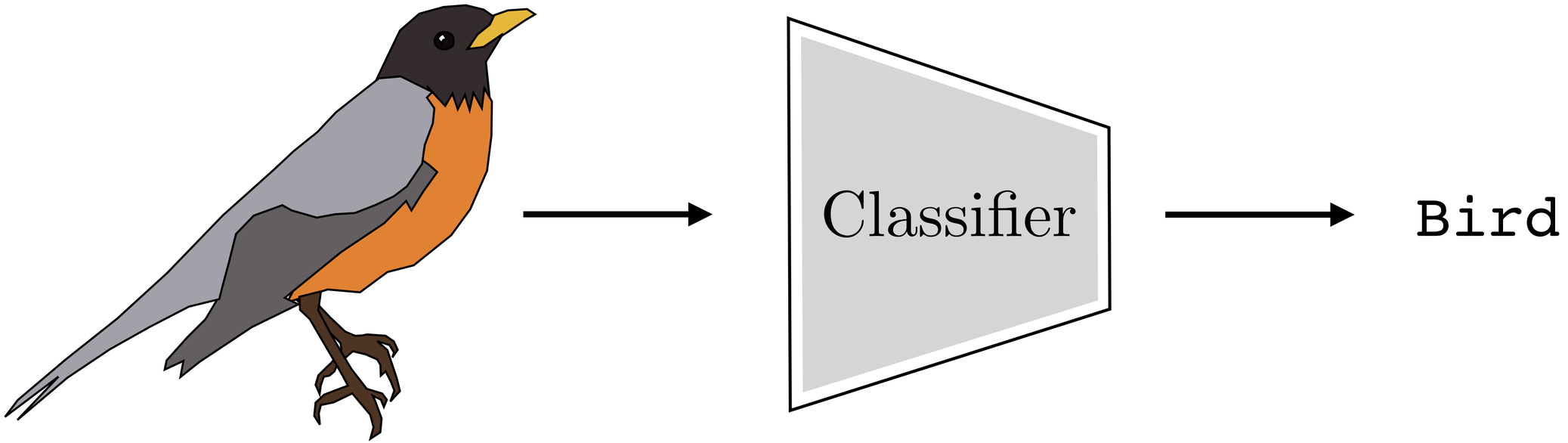
Representation learning methods can now extract very rich semantic features from images.

Surprisingly, there is no DINO for videos (V-JEPA is close). VideoMAE does not seem to be as impactful as MAE.

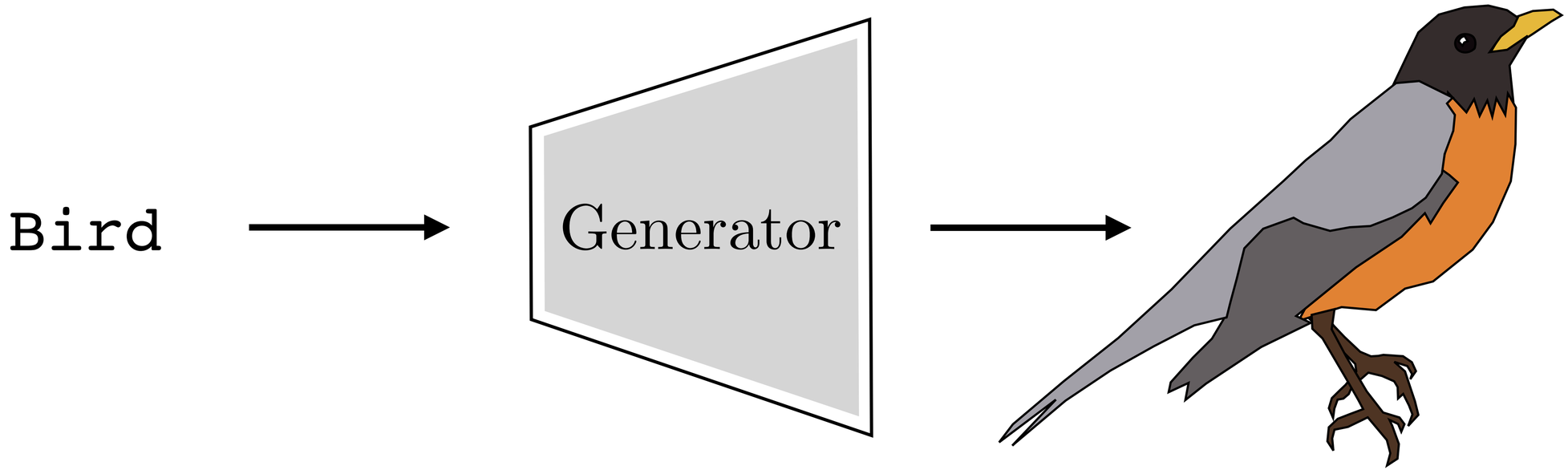
Generative models



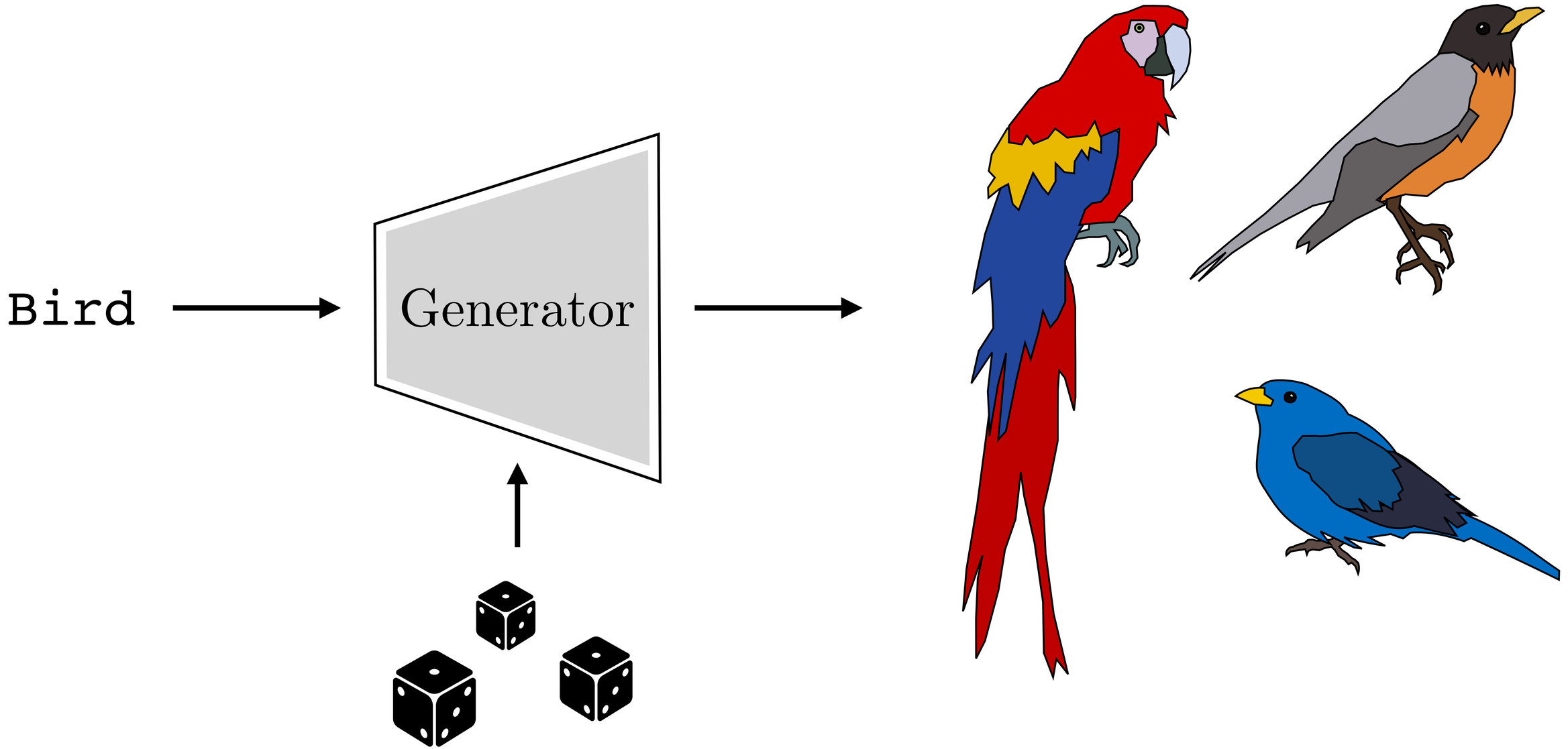
Generative models



Generative models

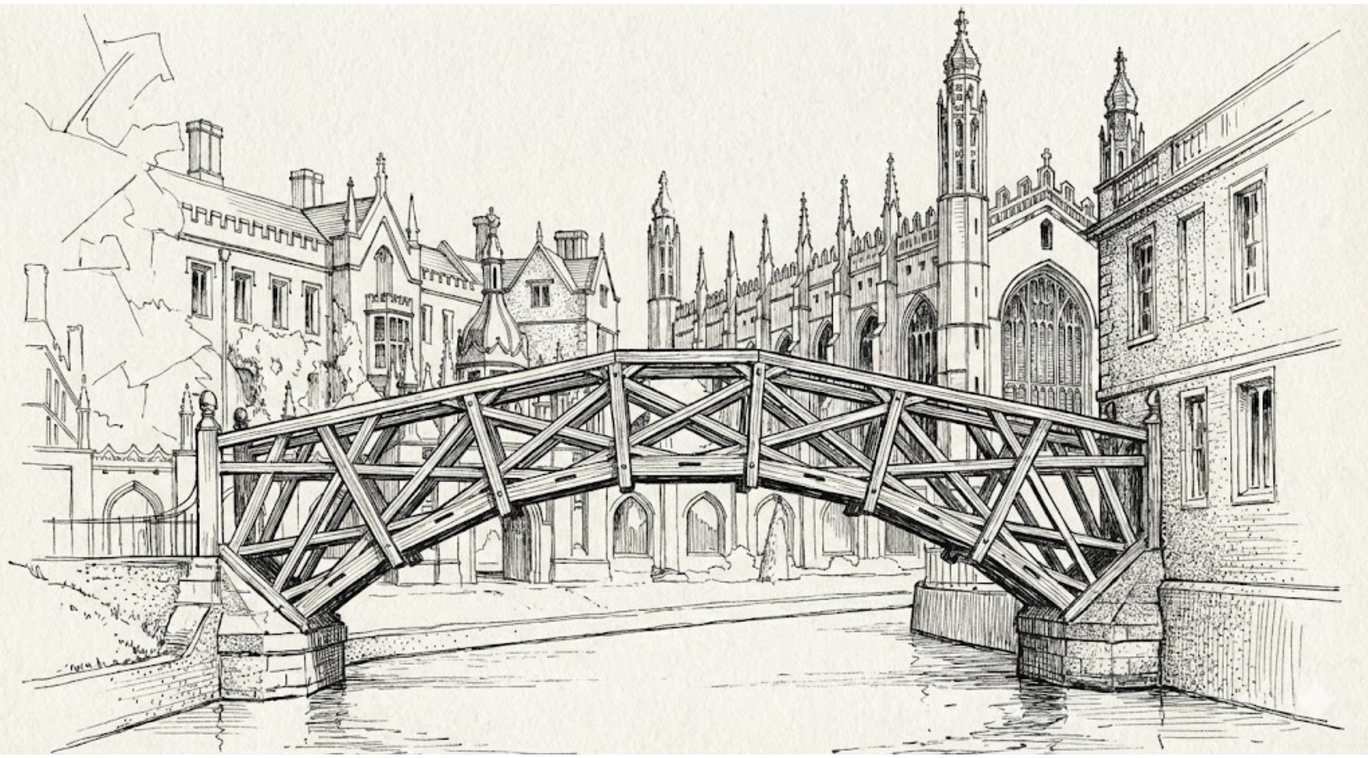


Generative models



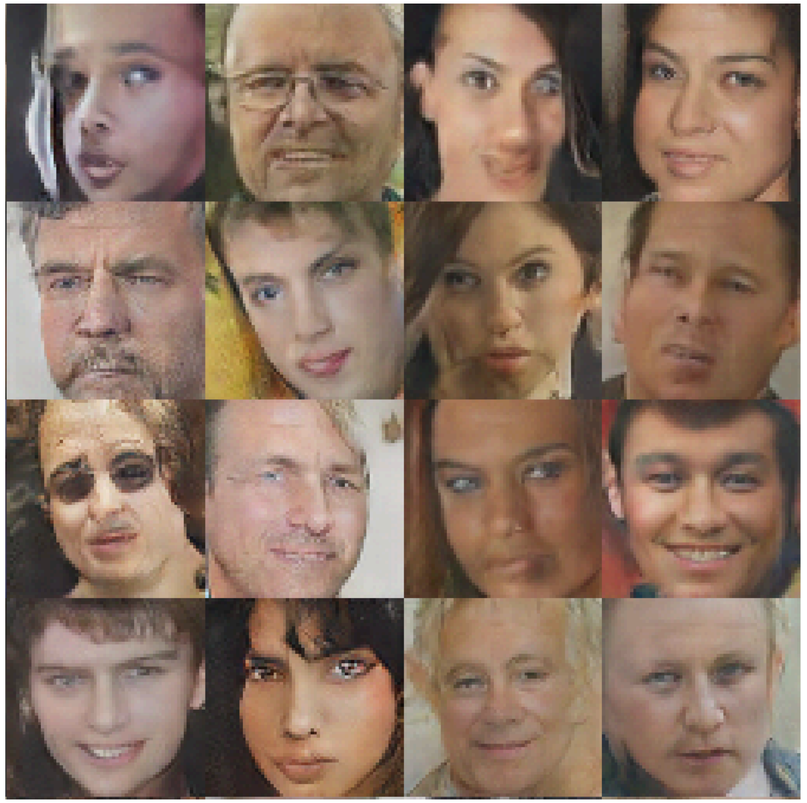
State of the art

Artistic Renditions of Cambridge (Gemini)



State of the art (once upon a time)

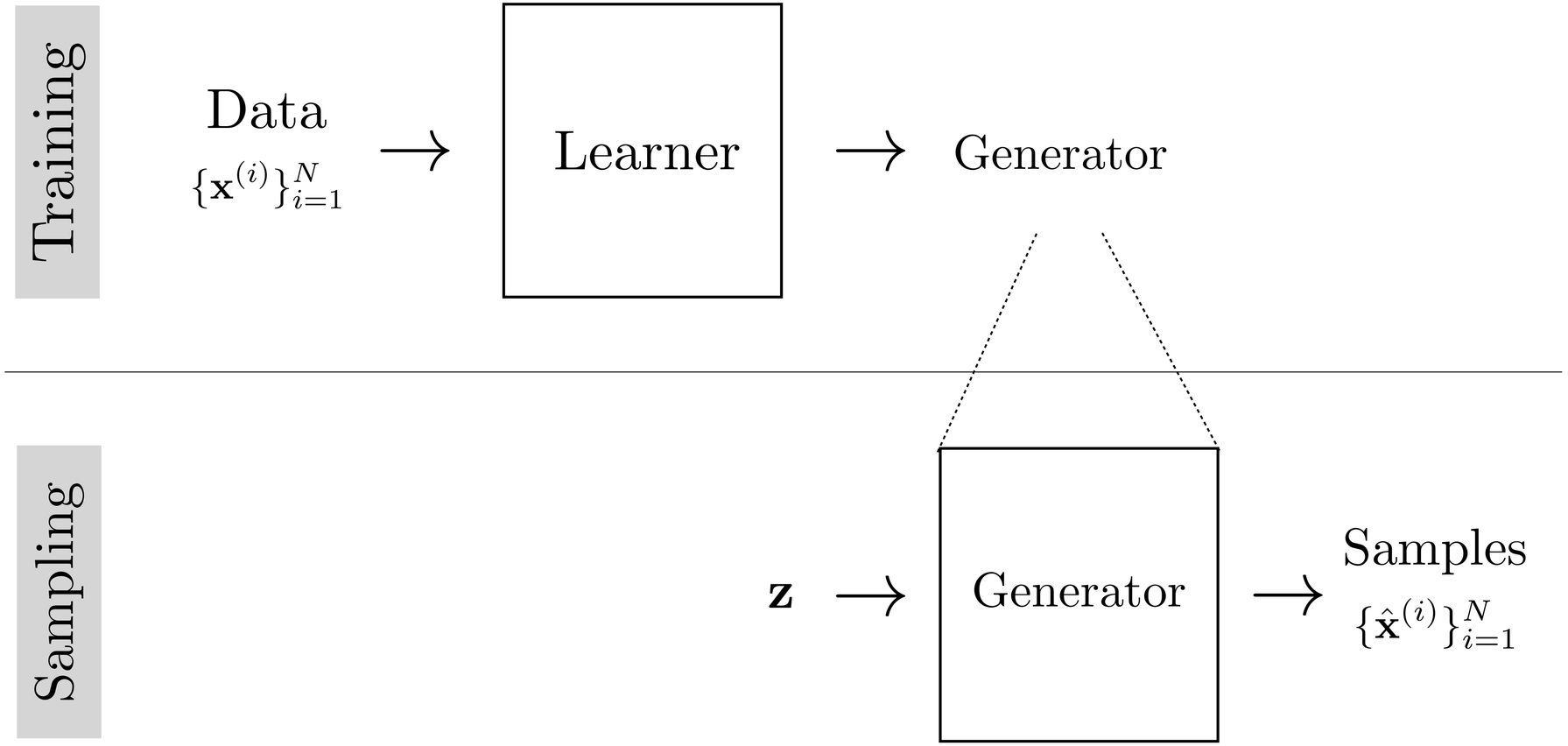
[Karras et al., ICLR 2018]



[Radford et al., ICLR 2016]

What is a generative model?

- **An algorithm that generates data**
- A statistical model of the joint distribution of some data, p(x,y,…)



Two types of generative models

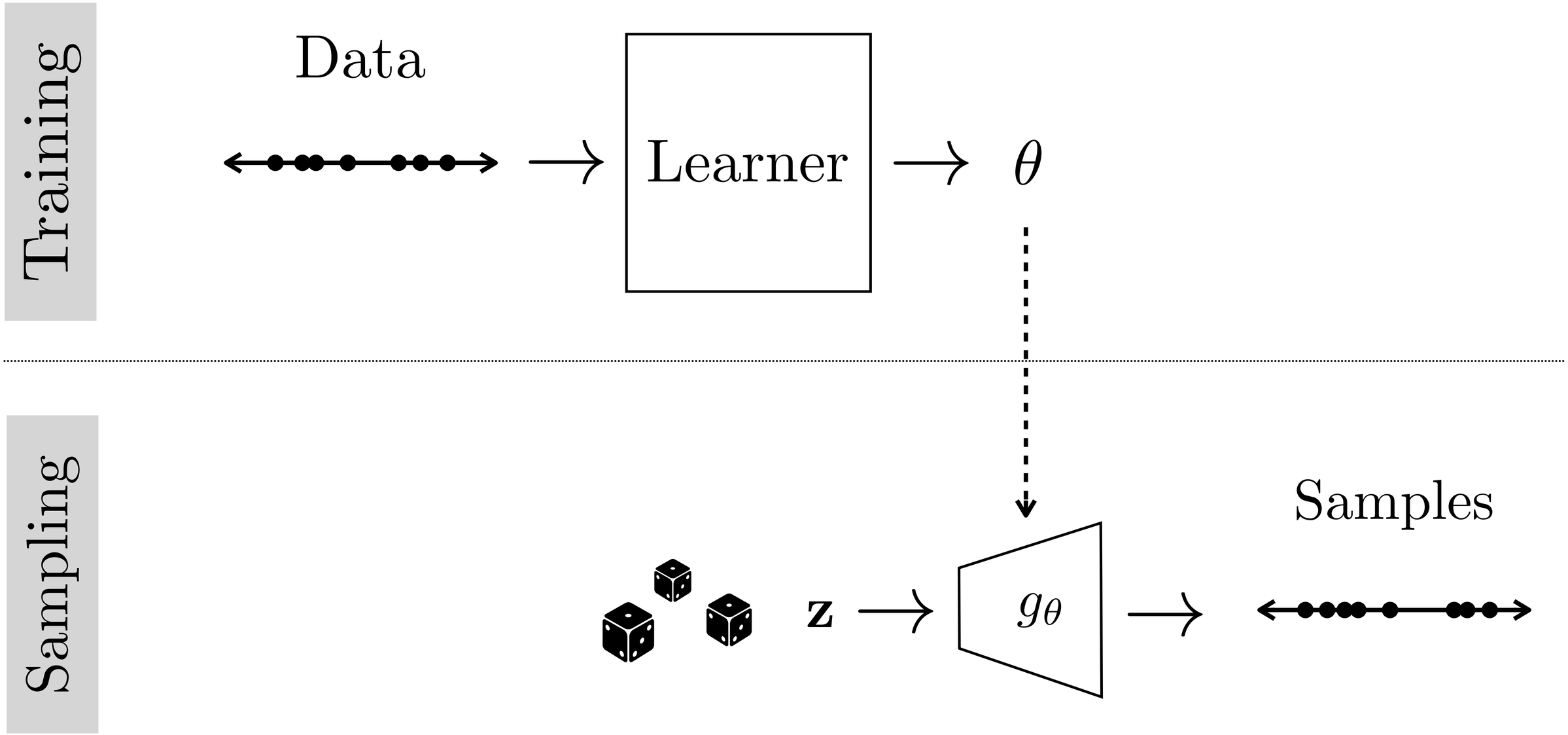
Two approaches:

confusingly, sometimes called an "implicit generative model"

1. **Direct approach**: learn a function that generates data directly $G : \mathcal{Z} \to \mathcal{X}$

2. **Indirect approach**: learn a function that scores data; generate data by finding points that score highly under this function $E : \mathcal{X} \to \mathbb{R}$

Direct Approach



Indirect approach

e.g., likelihood, energy, "score function"

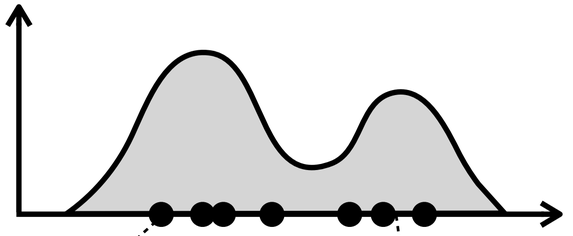
Training

Data

$\{\mathbf{x}^{(i)}\}_{i=1}^{N}$

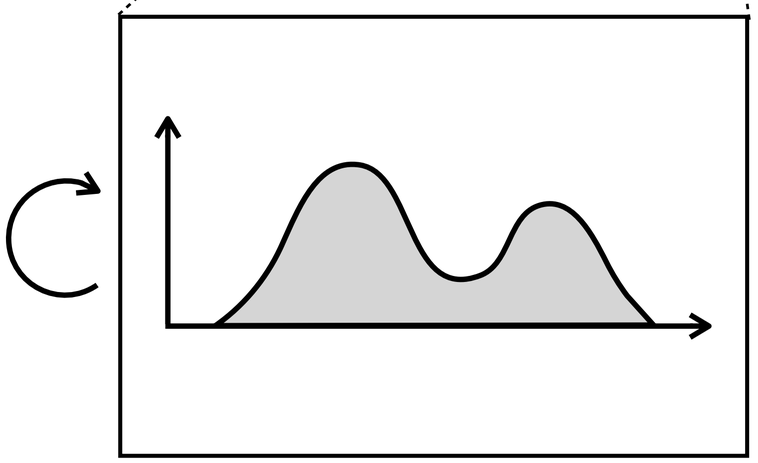
Learner

Scoring function



Sampling

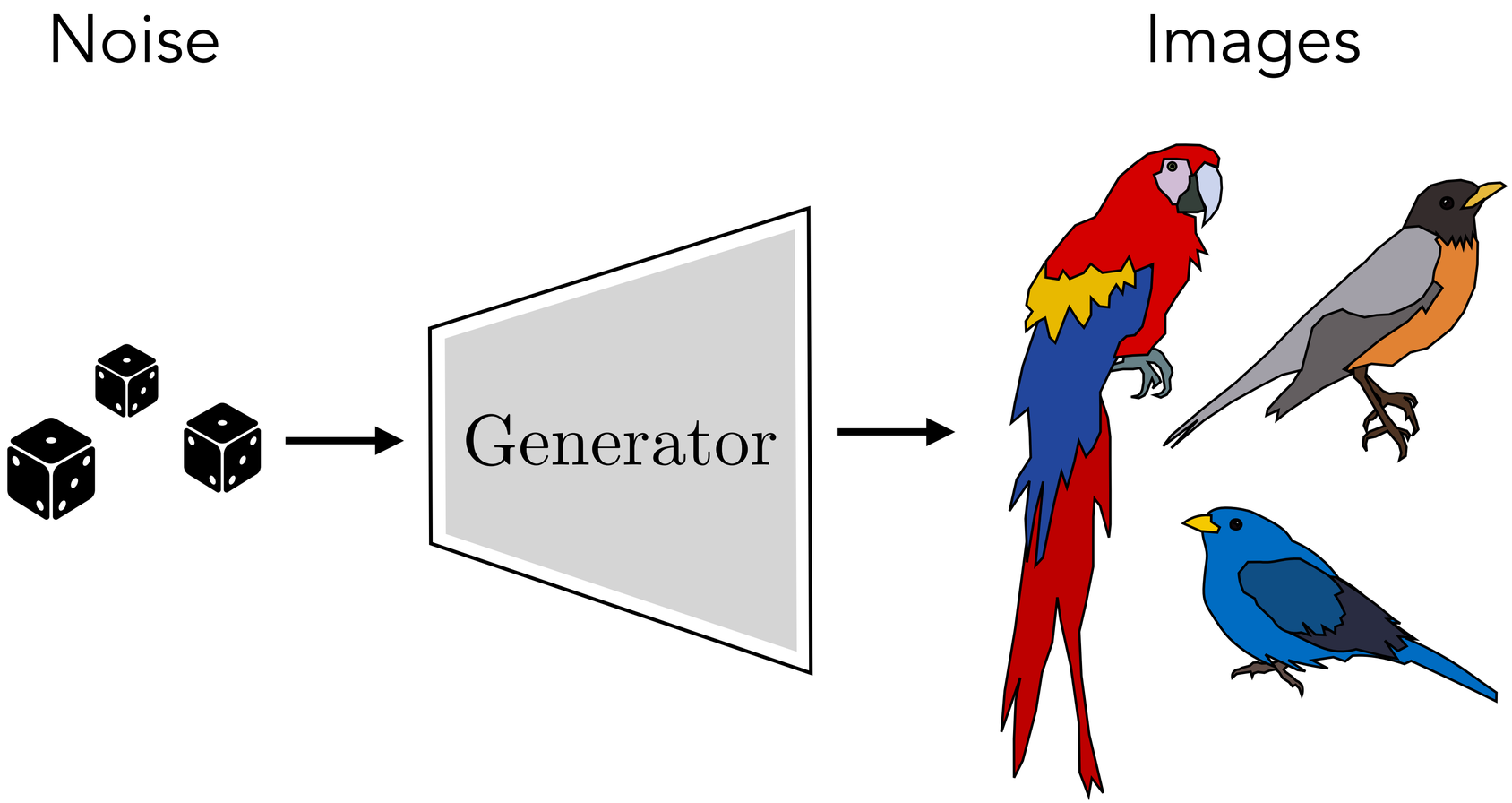
Sampling algorithm
(e.g., MCMC)



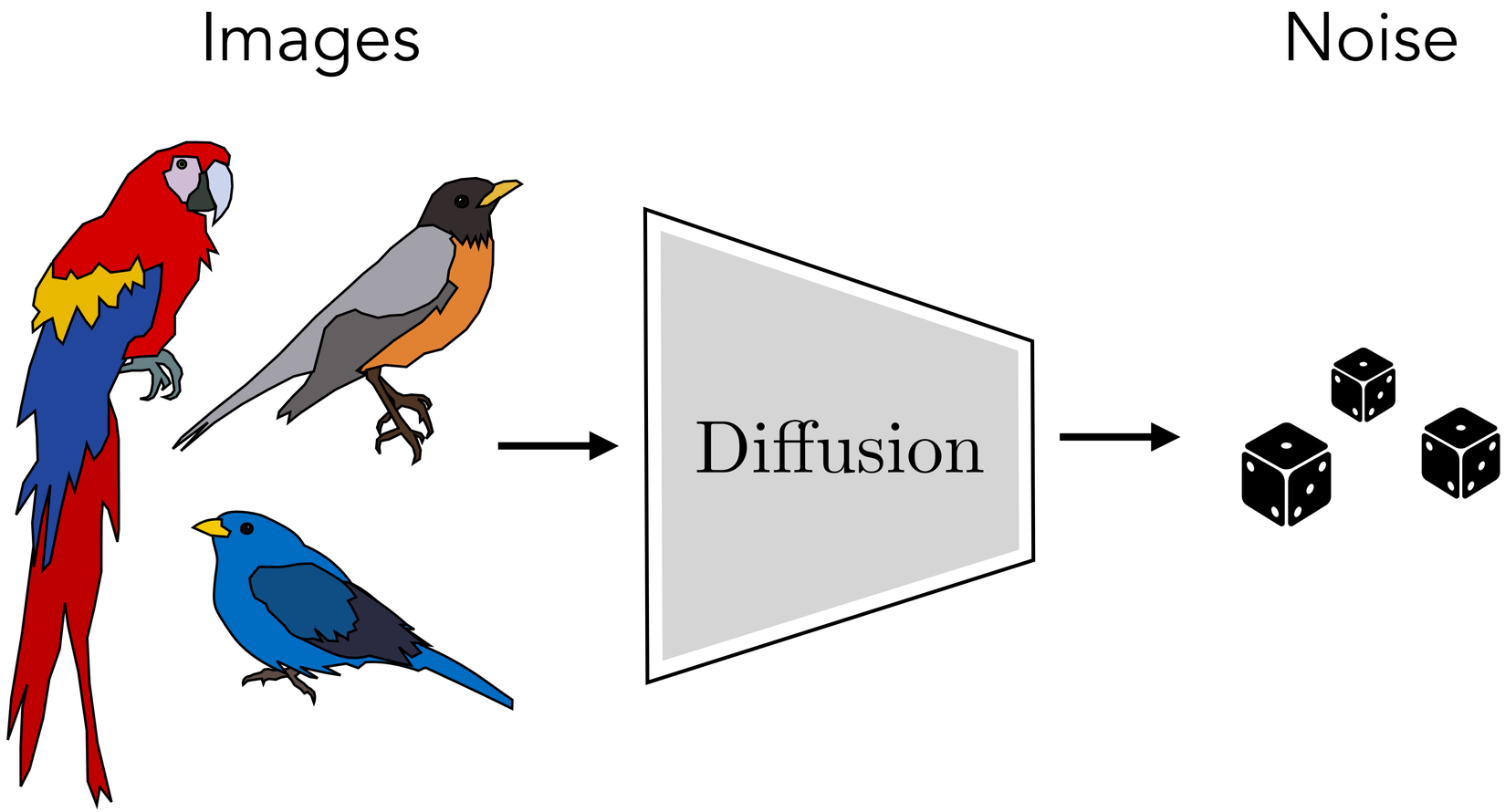
Samples

$\{\hat{\mathbf{x}}^{(i)}\}_{i=1}^{N}$

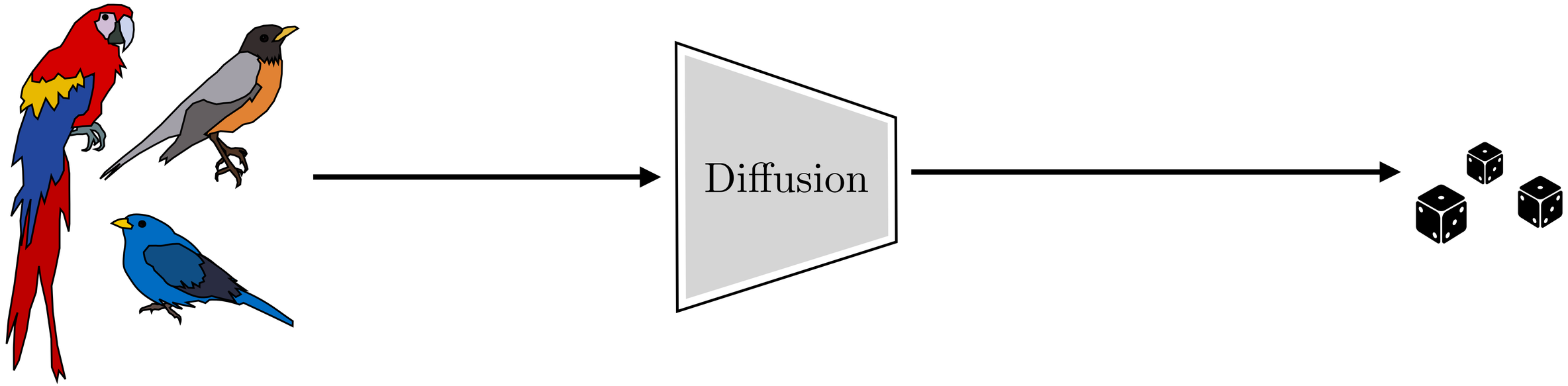
Diffusion Generative Models: Noise -> Data

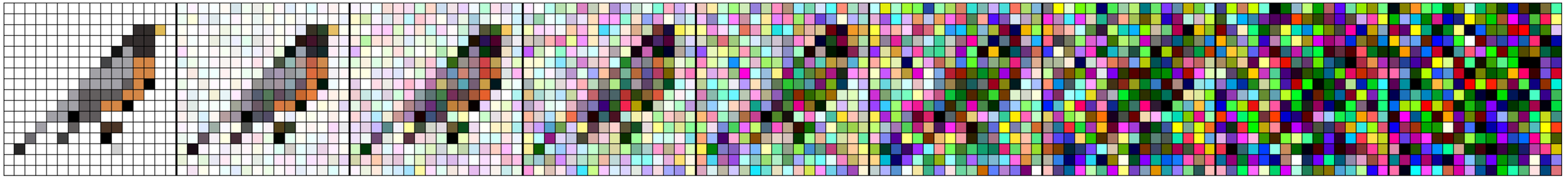


Diffusion Generative Models: Data -> Noise



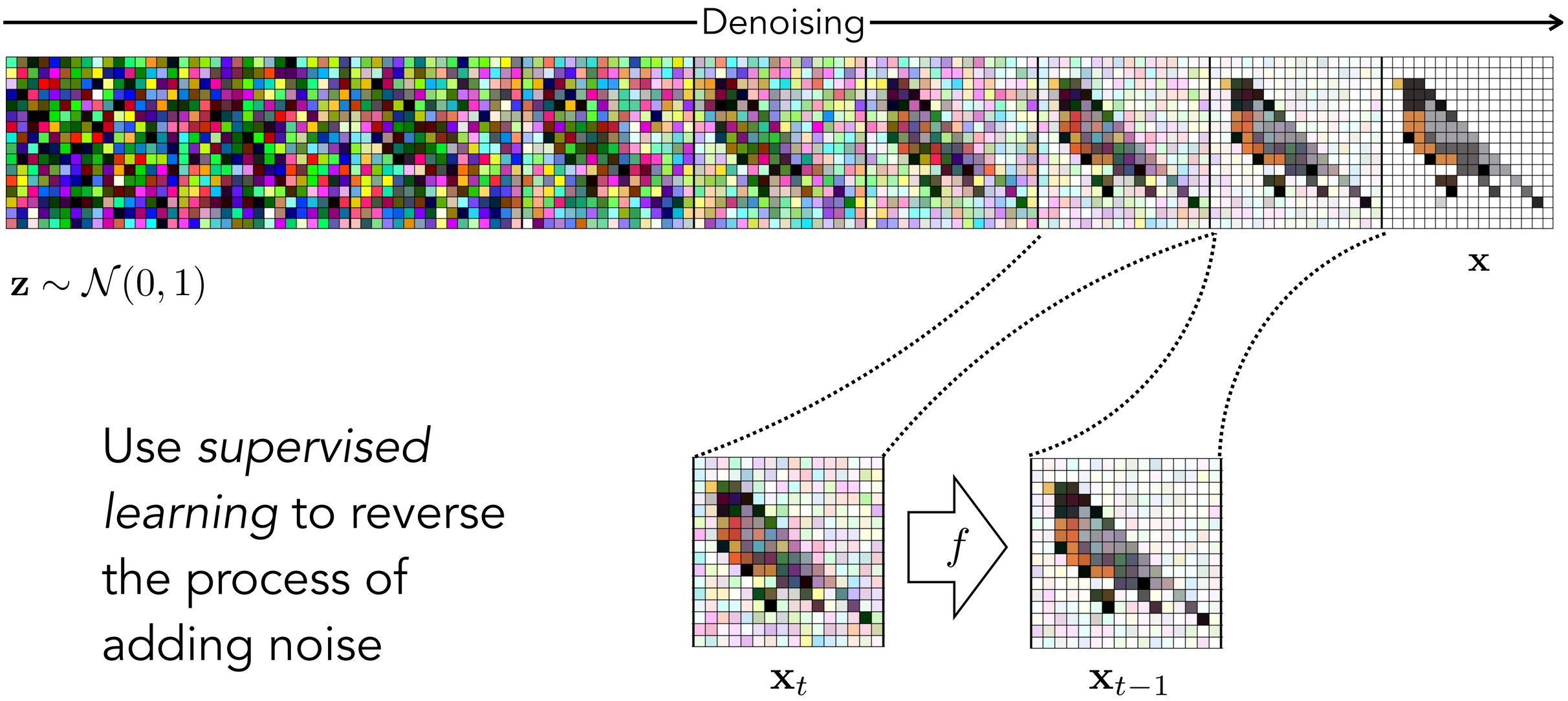
Diffusion Generative Models: Data -> Noise





Diffusion: Just add noise

Diffusion Generative Models

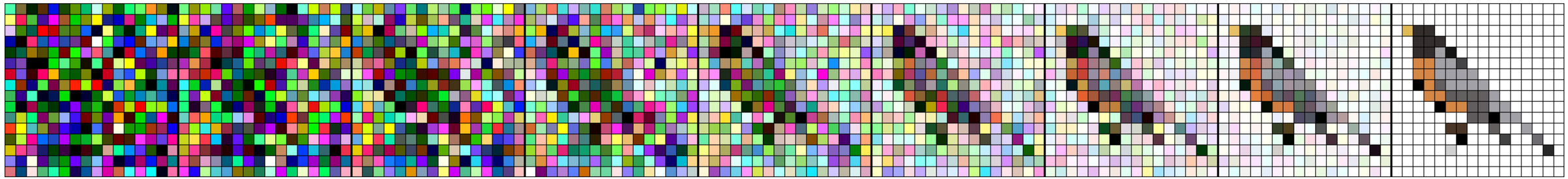


Use *supervised learning* to reverse the process of adding noise

Diffusion Generative Models

$\mathbf{z} \sim \mathcal{N}(0, 1)$

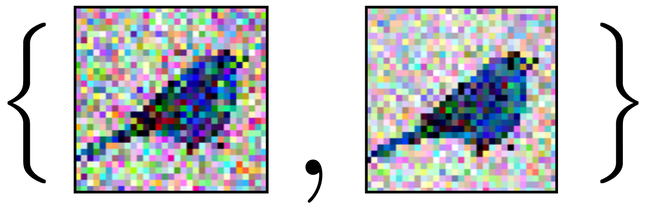
$\mathbf{x}$



Denoising

Training data

$\mathbf{x}_t$ $\mathbf{x}_{t-1}$

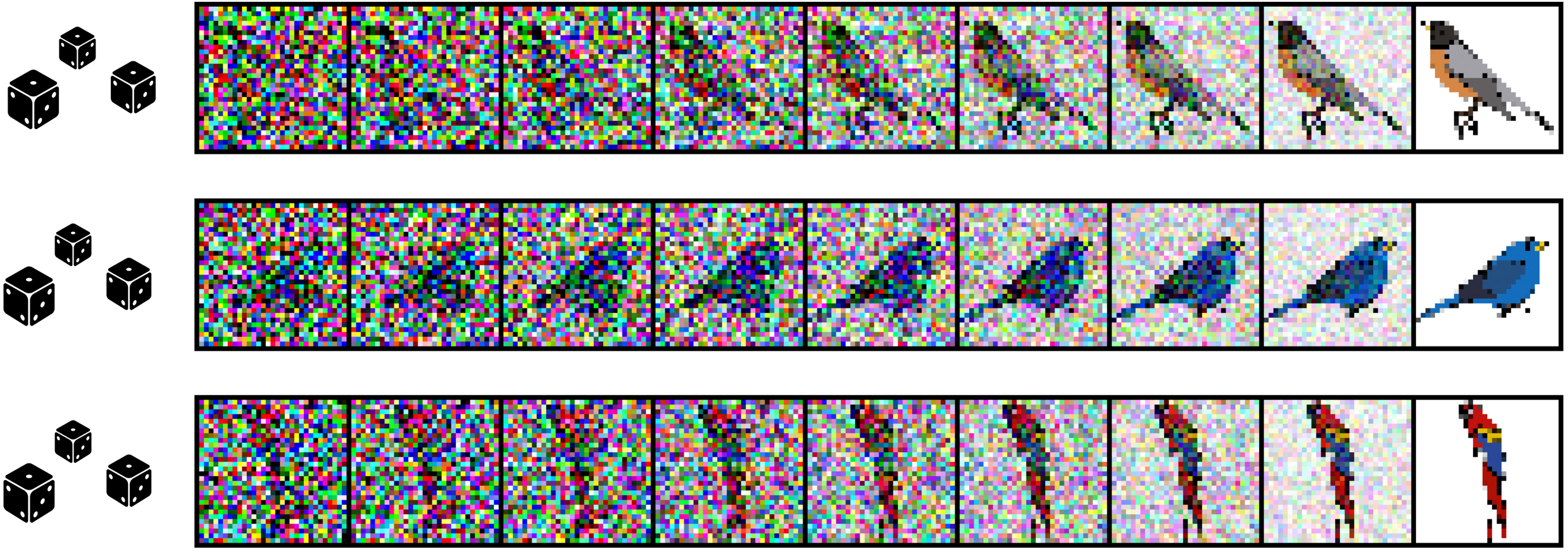


$$\arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_t), \mathbf{x}_{t-1})$$

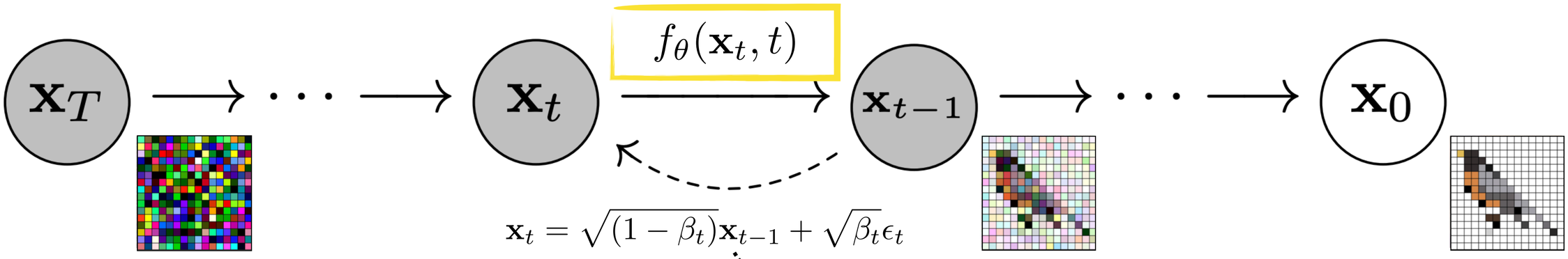
Converts generative modeling into a bunch of supervised prediction problems

Diffusion Generative Models

Different noise samples (dice rolls) result in different images



Gaussian Diffusion Models



Forward process:

$$\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad \mathbf{x}_t = \sqrt{(1-\beta_t)}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_t$$

Reverse process:

$$\mu = f_\theta(\mathbf{x}_t, t) \qquad \mathbf{x}_{t-1} \sim \mathcal{N}(\mu, \sigma^2)$$

The variances, beta and sigma, are modeling choices. See Ho, Jain, and Abbeel for details.

[Fig adapted from Ho, Jain, Abbeel, 2020]

Gaussian Diffusion Models

Algorithm 1.2: Training a diffusion model.

Input: training data $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$

Output: trained model f_θ

Generate training sequences via diffusion:

for $i = 1, \ldots, N$ **do**

  for $t = 1, \ldots, T$ **do**

    $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

    $\mathbf{x}_t^{(i)} \leftarrow \sqrt{(1-\beta_t)}\mathbf{x}_{t-1}^{(i)} + \sqrt{\beta_t}\epsilon_t$

Train denoiser f_θ **to reverse these sequences:**

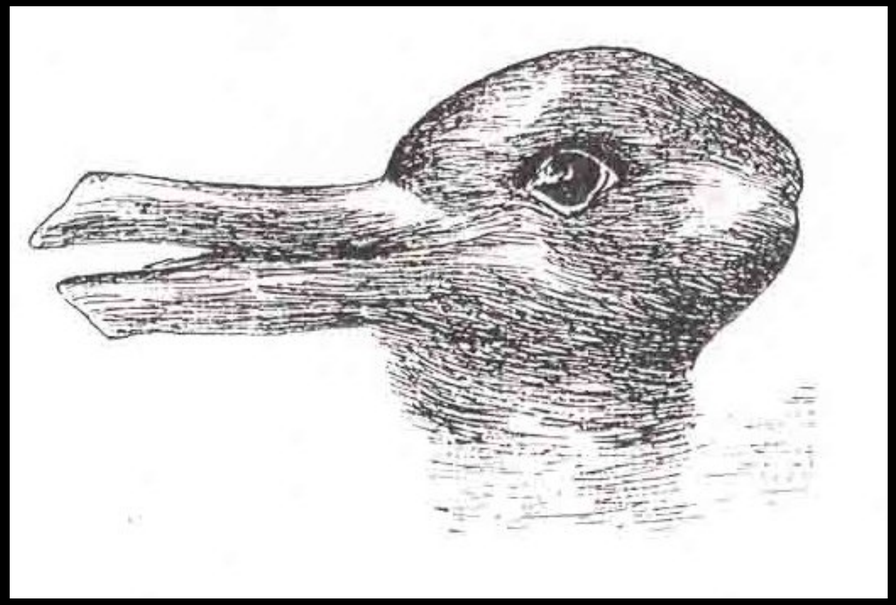
$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \sum_{t=1}^{T} \mathcal{L}(f_\theta(\mathbf{x}_t^{(i)}, t), \mathbf{x}_{t-1}^{(i)}))$

Return: f_{θ^*}

Advanced Computer Vision: Representation Learning, Generative Models

MLMI17

Ayush Tewari



UNIVERSITY OF CAMBRIDGE