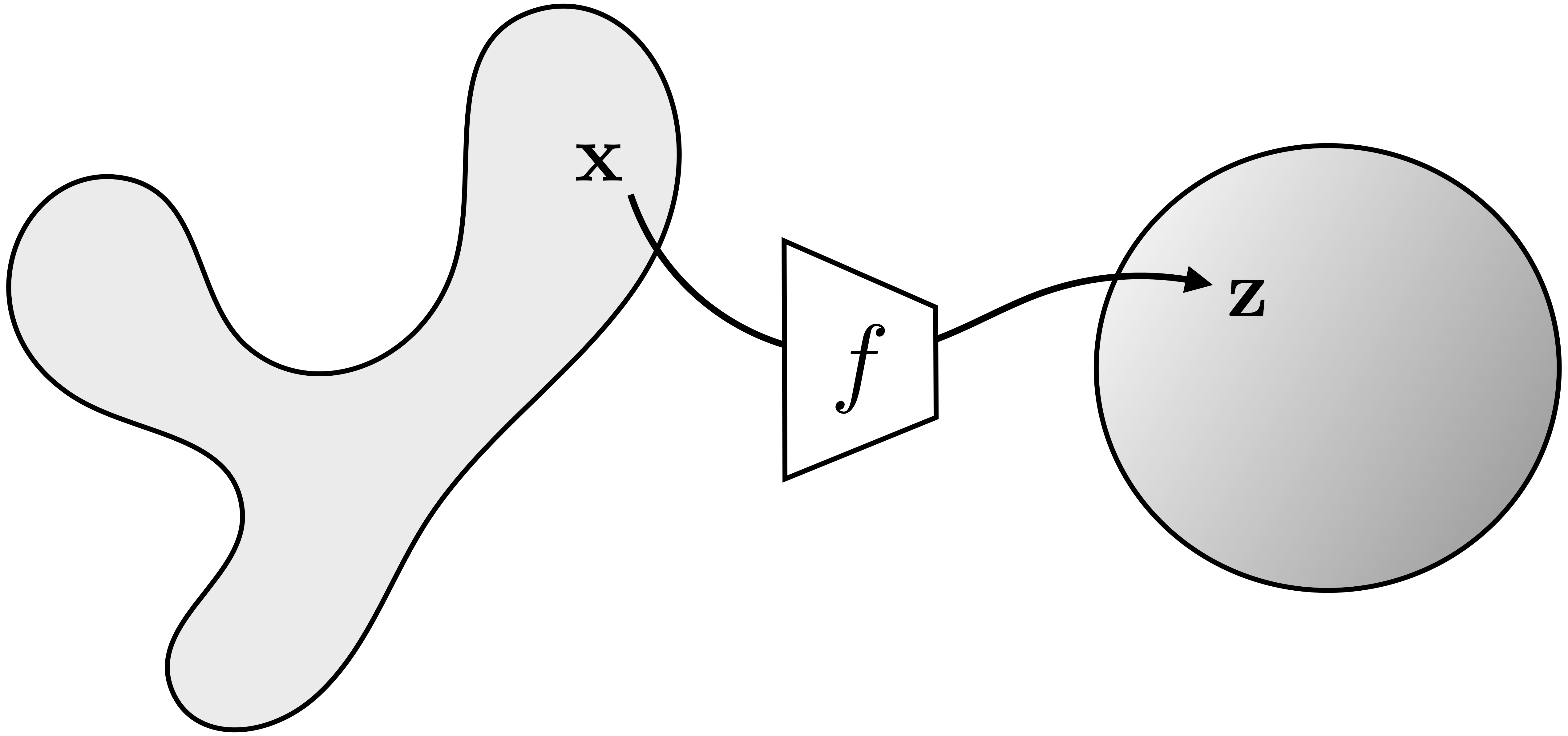
Lecture 11: Representation Learning I

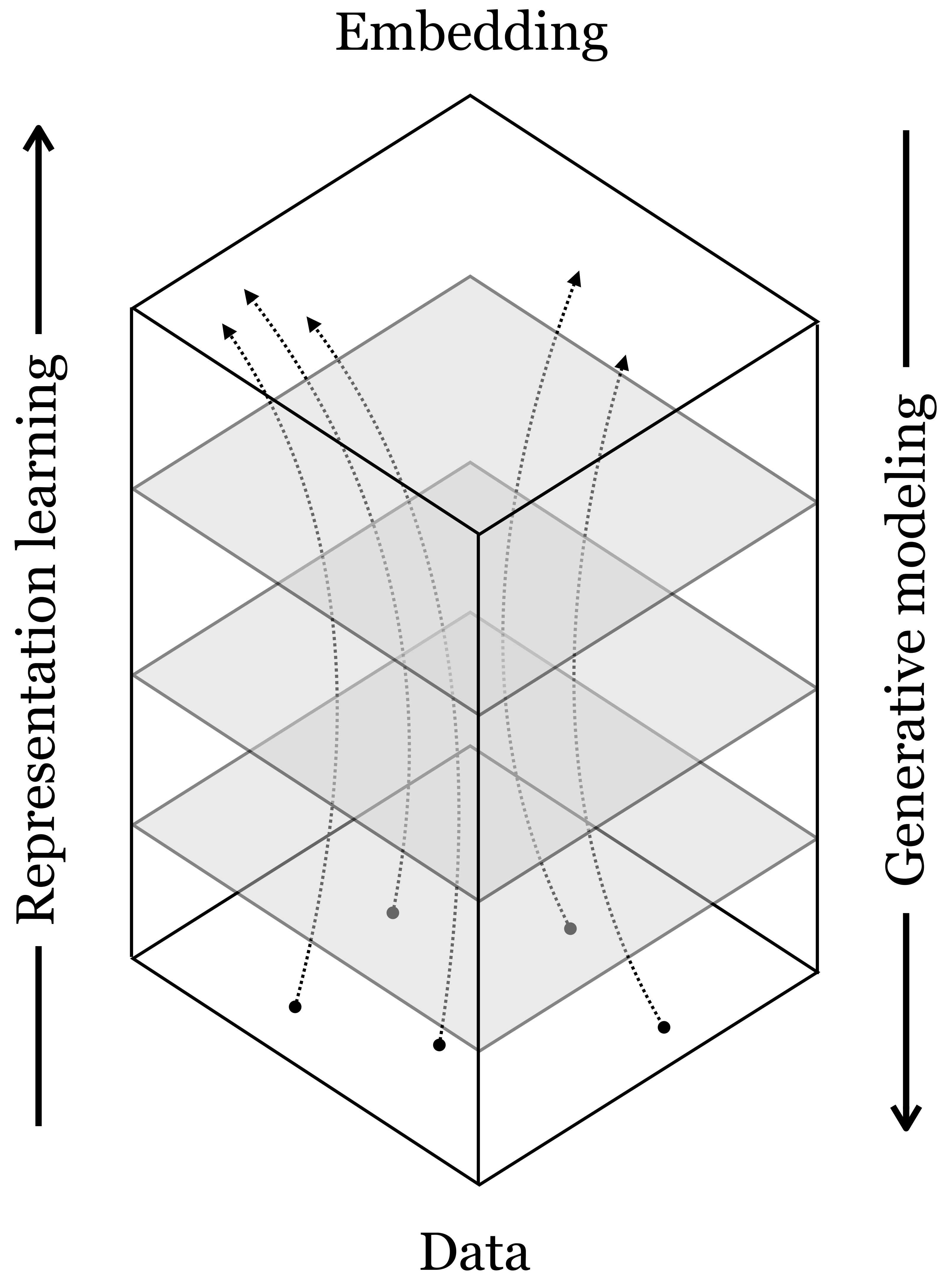
Speaker: Phillip Isola



12. Representation Learning I

- Nets learn representations
- Why learn representations?
- Autoencoders
- Clustering and VQ
- Self-supervised learning by reconstruction

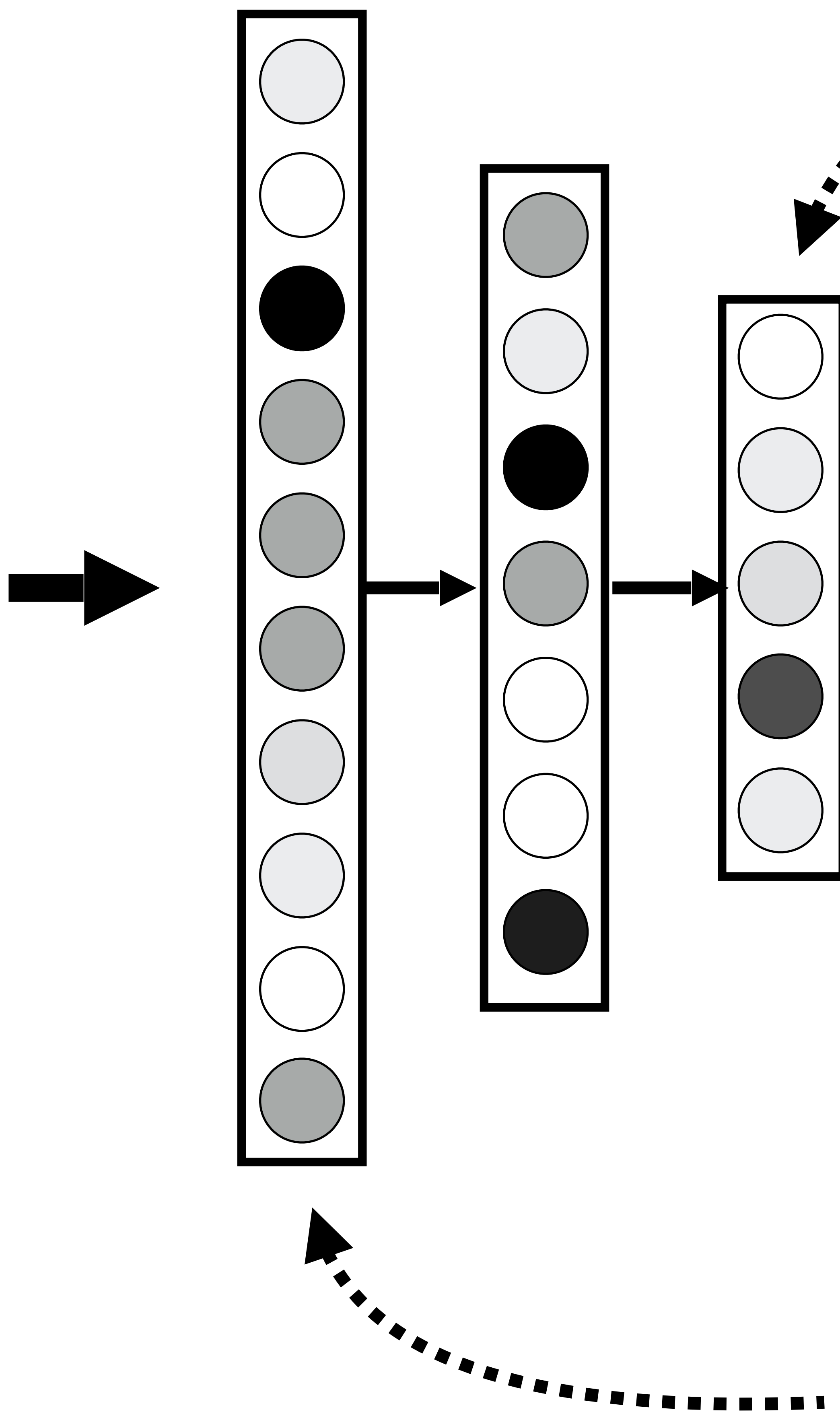
- Deep nets transform datapoints, layer by layer
- Each layer is a different *representation* of the data
- In the forward direction, the mapping goes from observed data to latent embeddings — this direction is called **representation learning**
- In the reverse direction, the mapping goes from latent embeddings to observed data — this direction is called **generative modeling**



x2vec

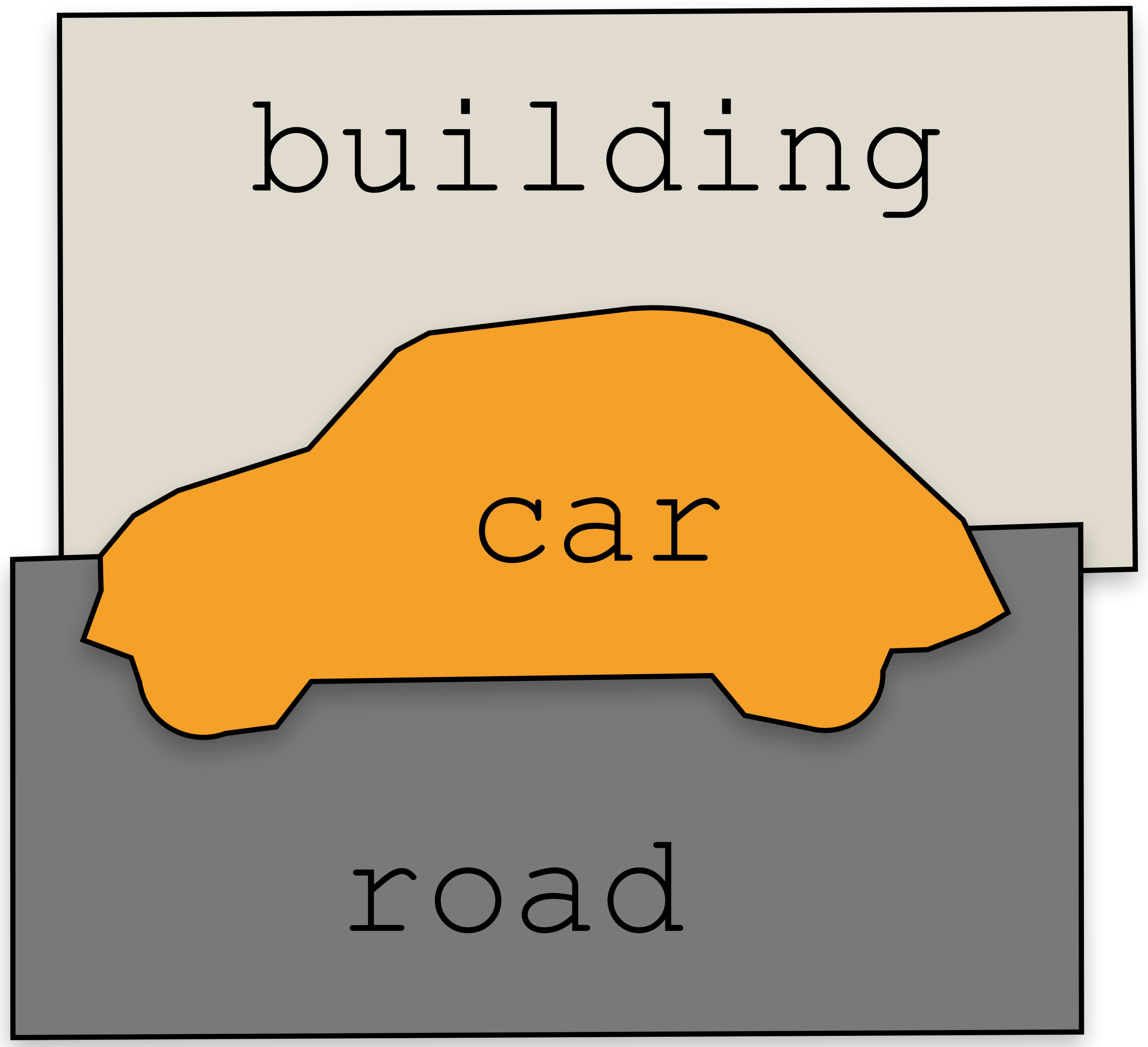
x

Image



layer 3 representation of image

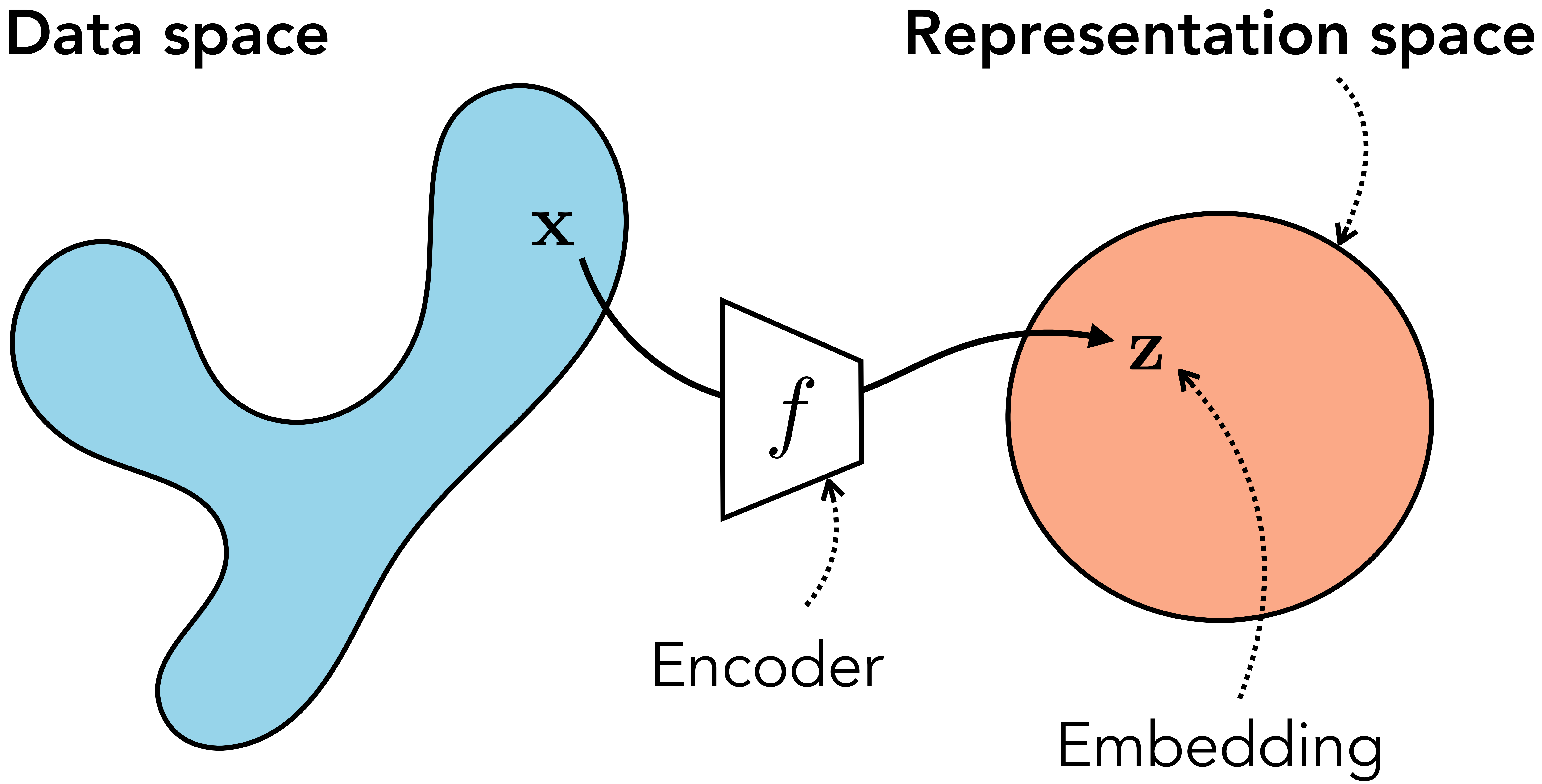
layer 1 representation of image



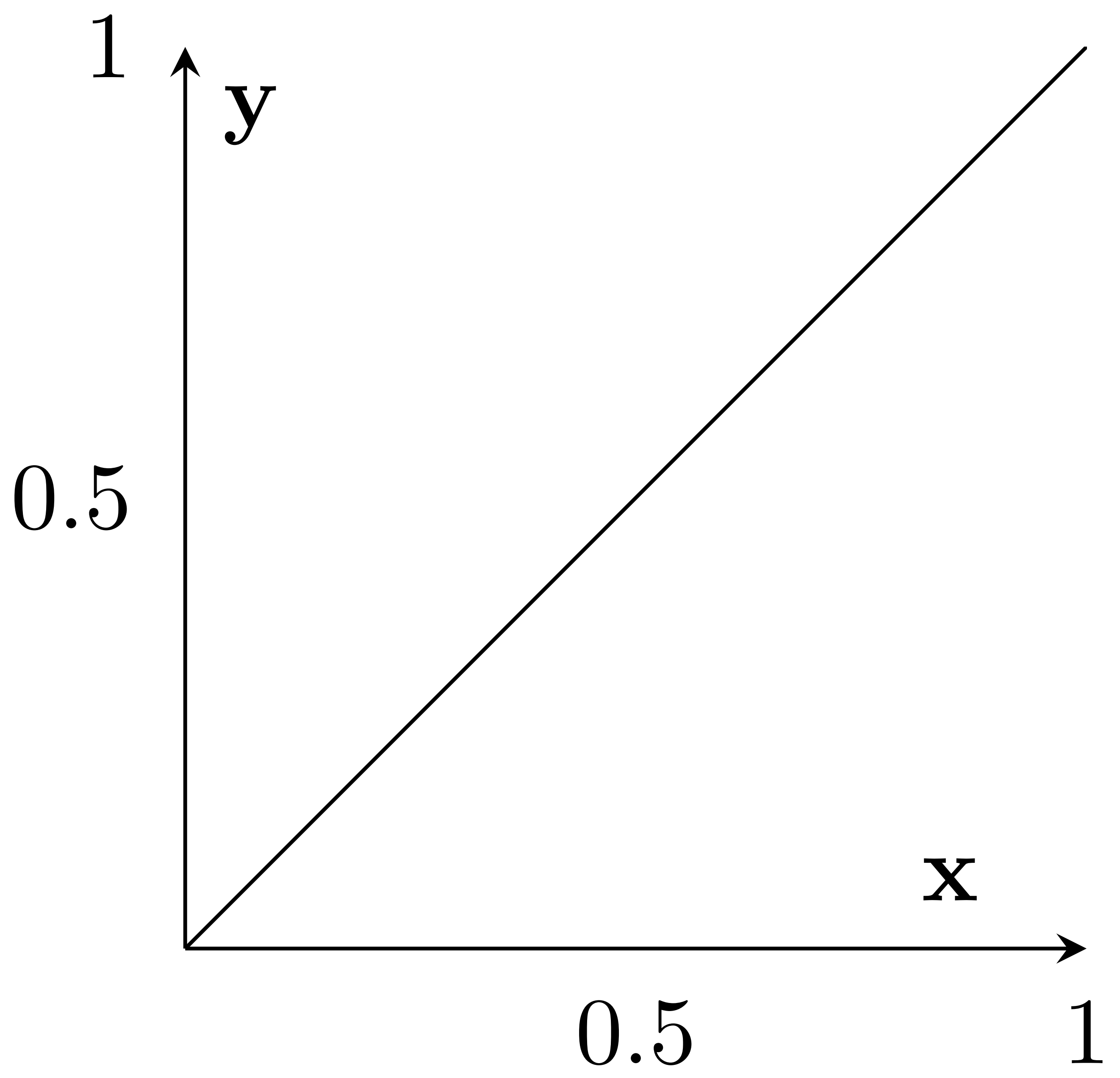
Represent data as a neural **embedding** — a vector/tensor of neural activations

(perhaps representing a vector of detected texture patterns or object parts)

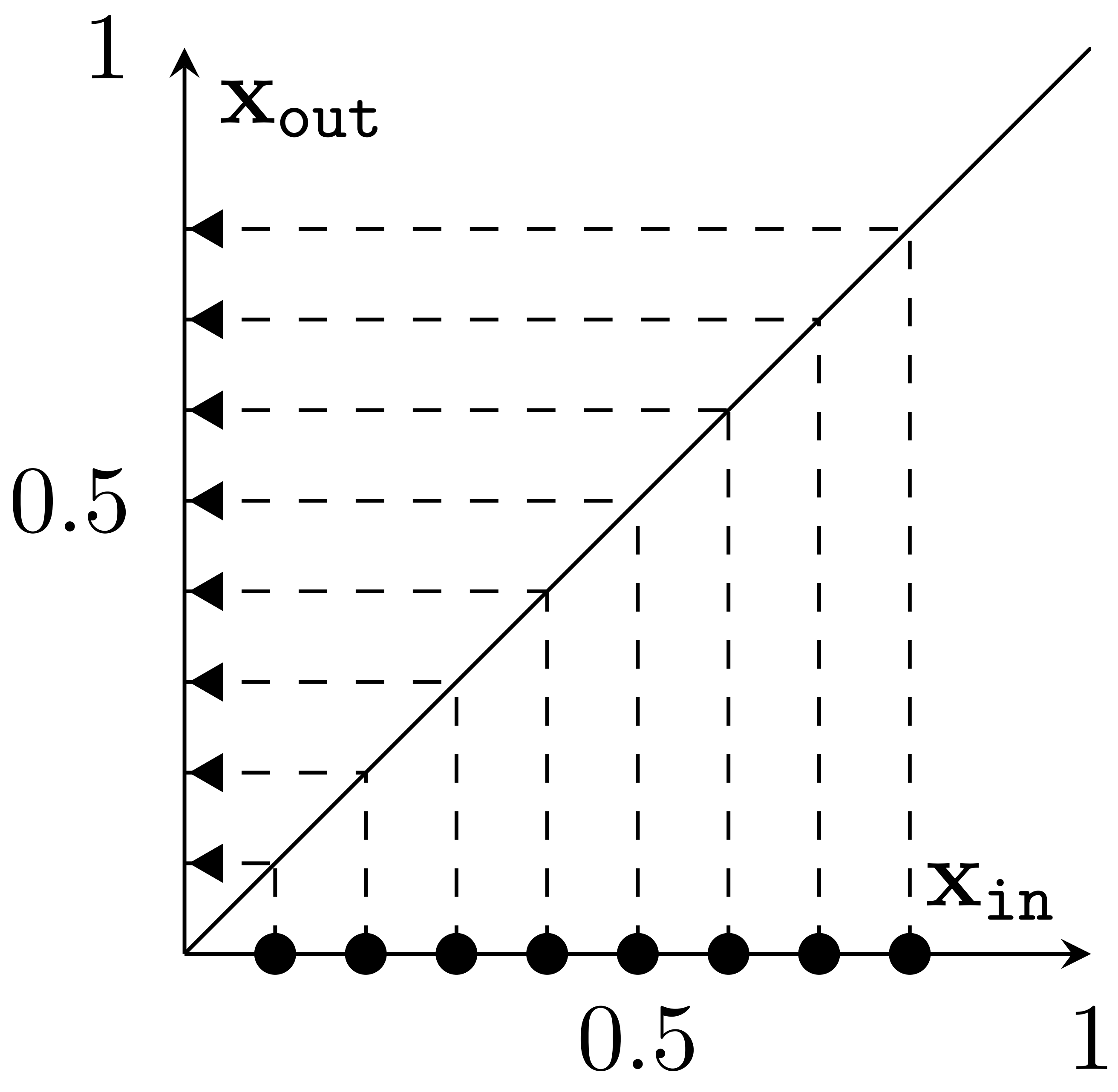
x2vec



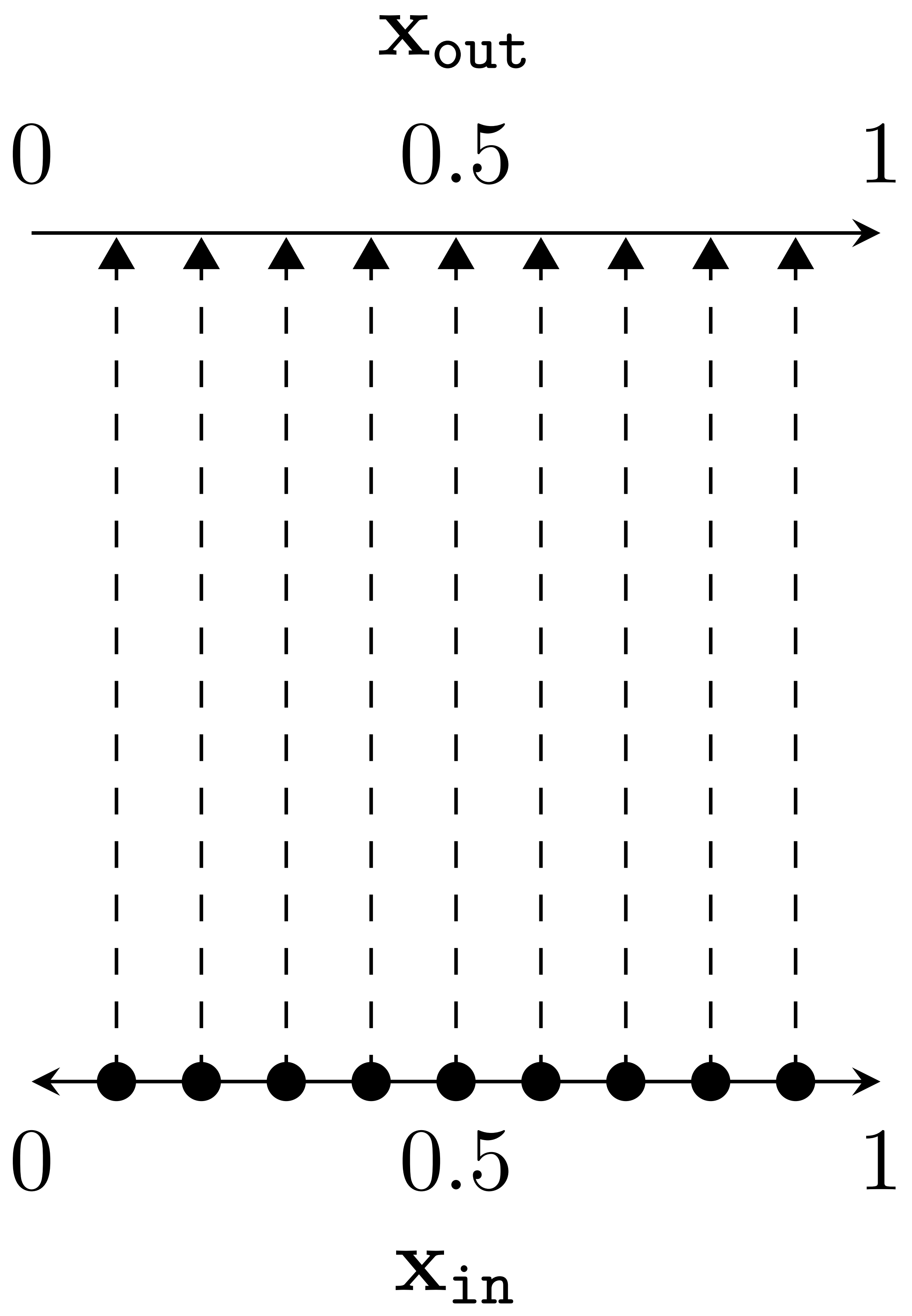
Two different ways to represent a function



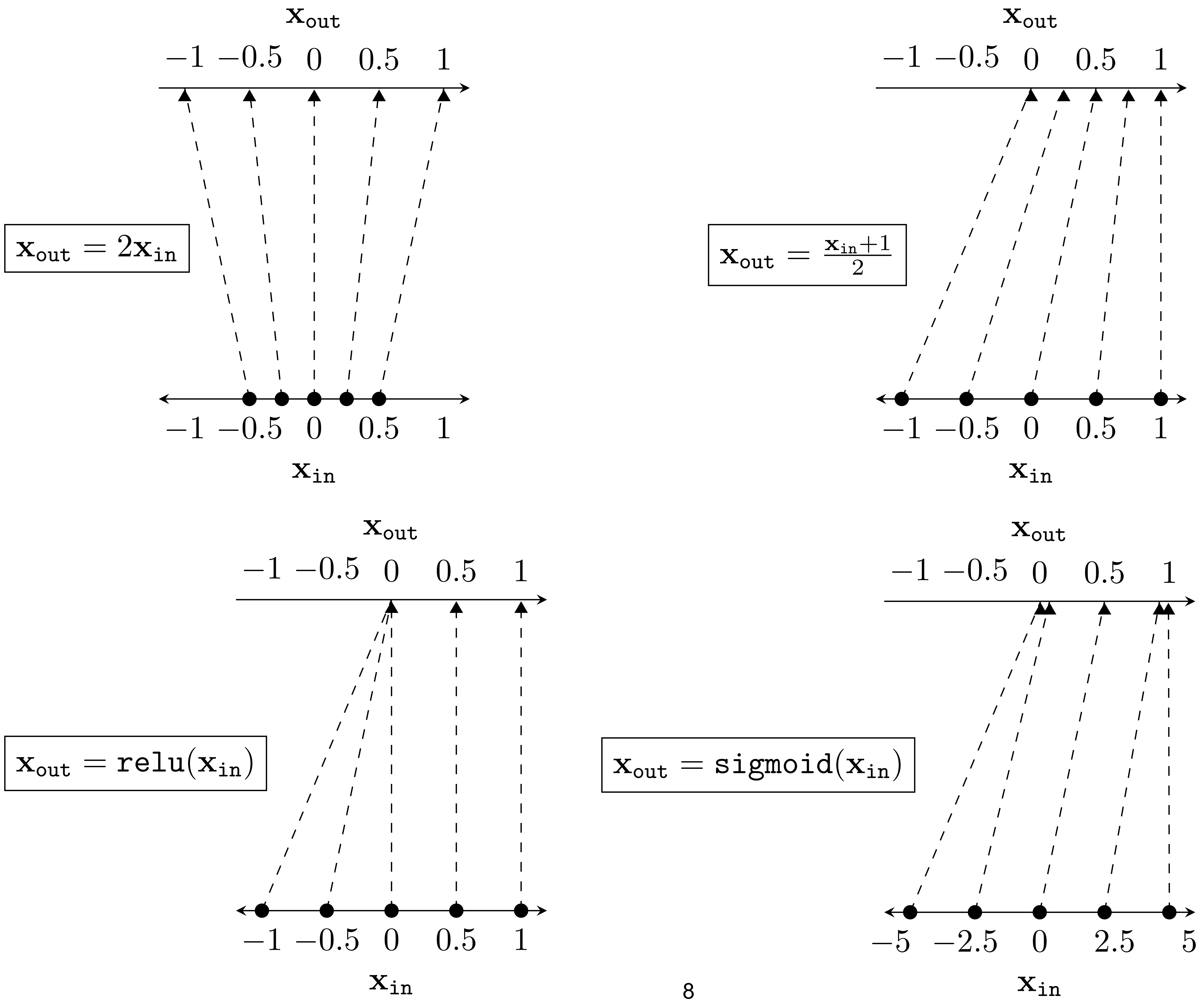
Two different ways to represent a function



$\Rightarrow$



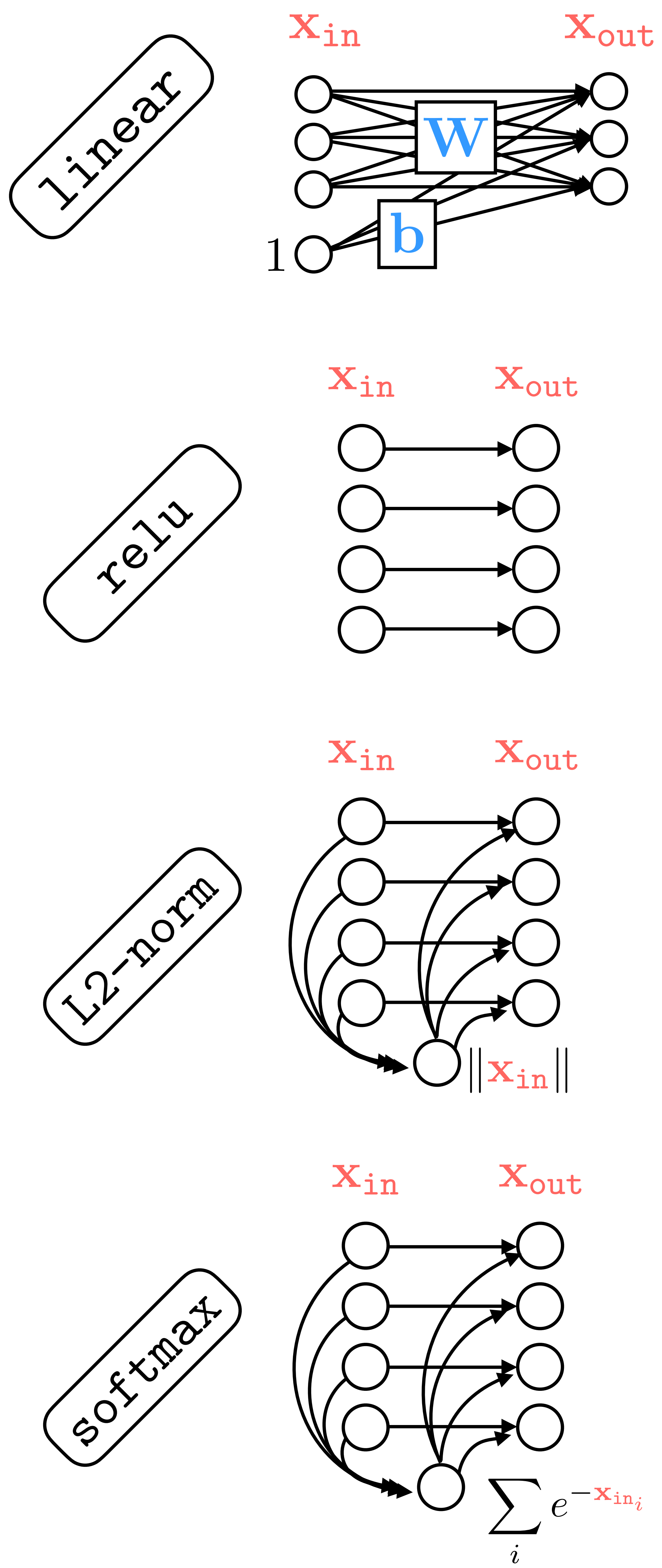
Data transformations for a variety of neural net layers



Activations

Parameters

Wiring graph | Equation | Mapping

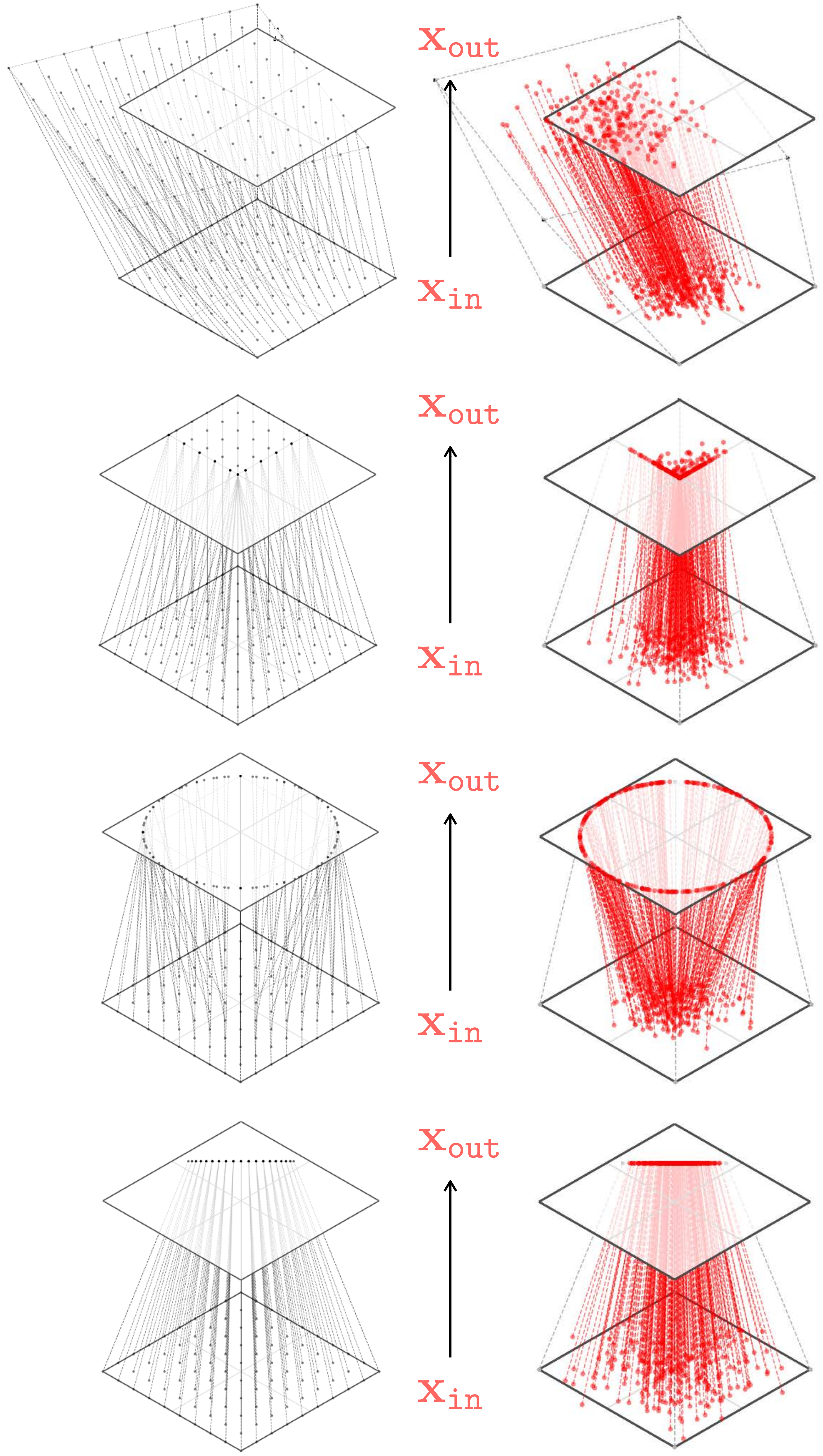


$$\mathbf{x}_{\text{out}} = \mathbf{W}\mathbf{x}_{\text{in}} + \mathbf{b}$$

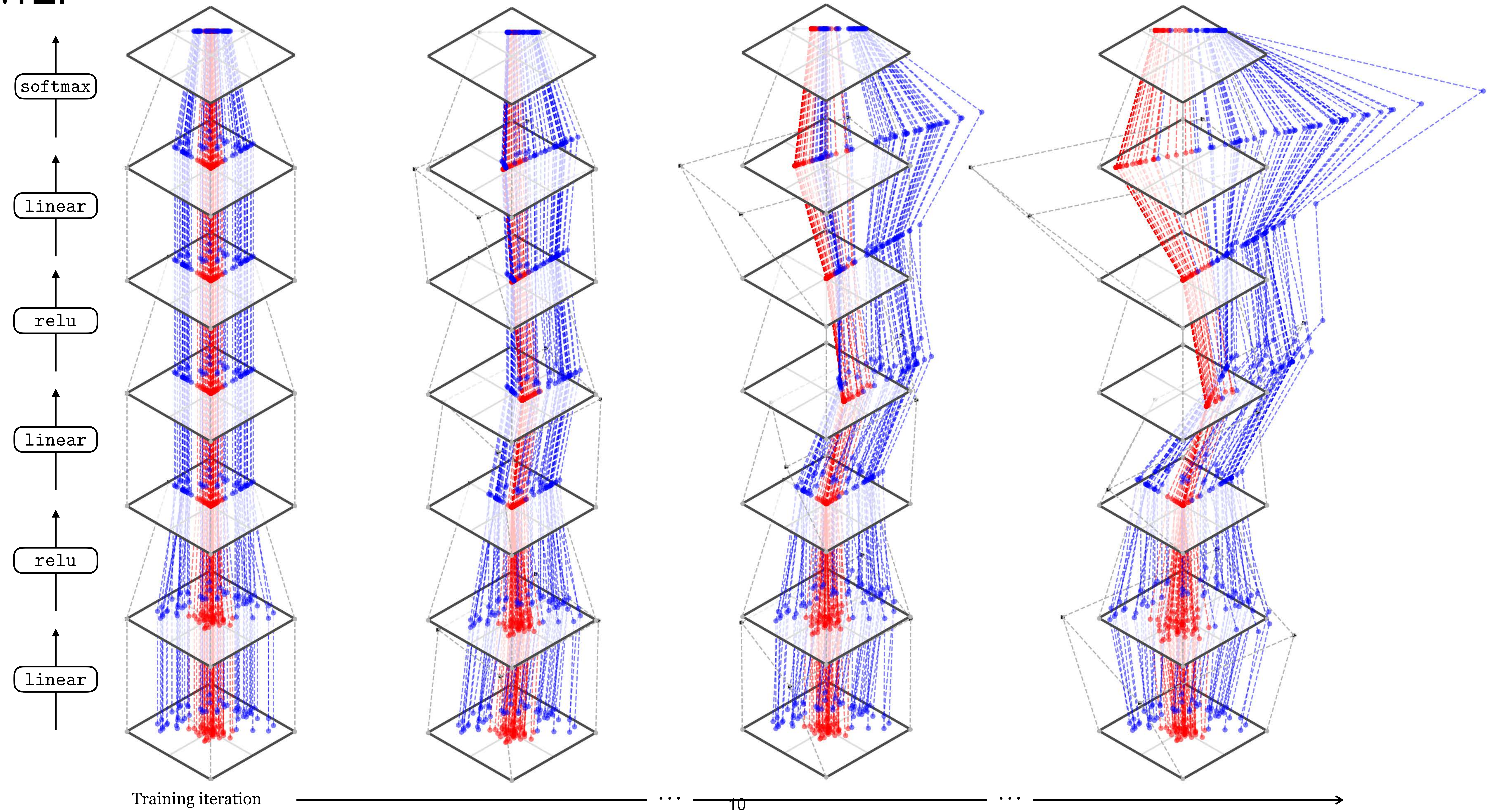
$$x_{\text{out}}[i] = \max(x_{\text{in}}[i], 0)$$

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

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

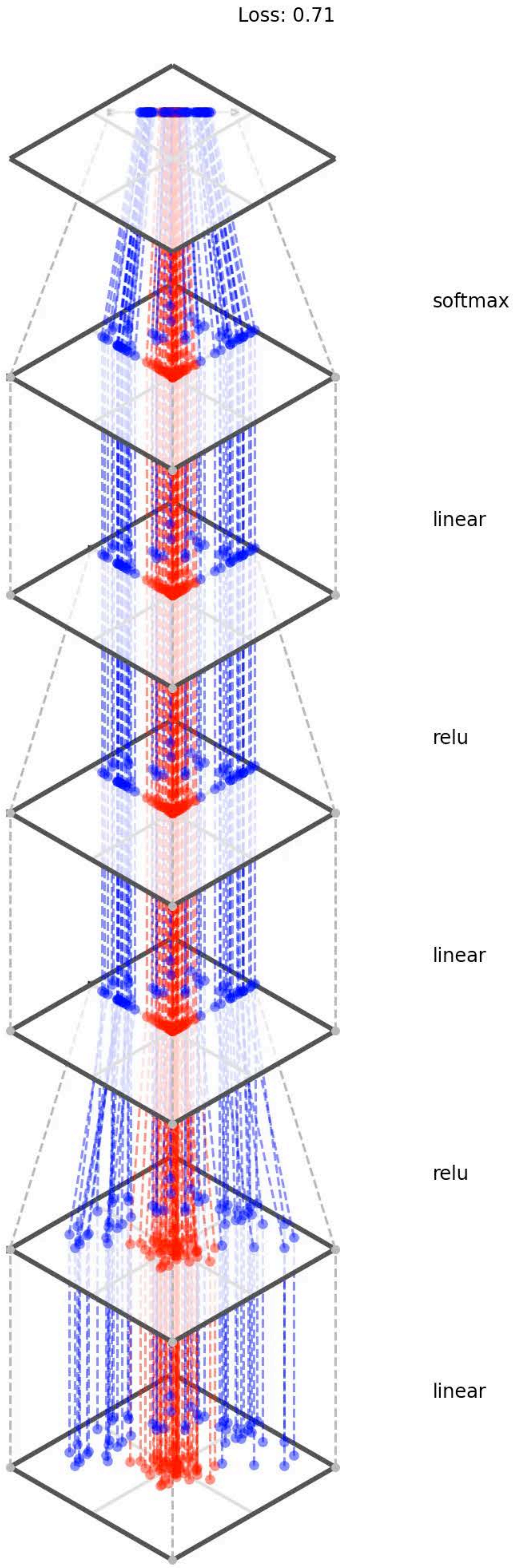


MLP

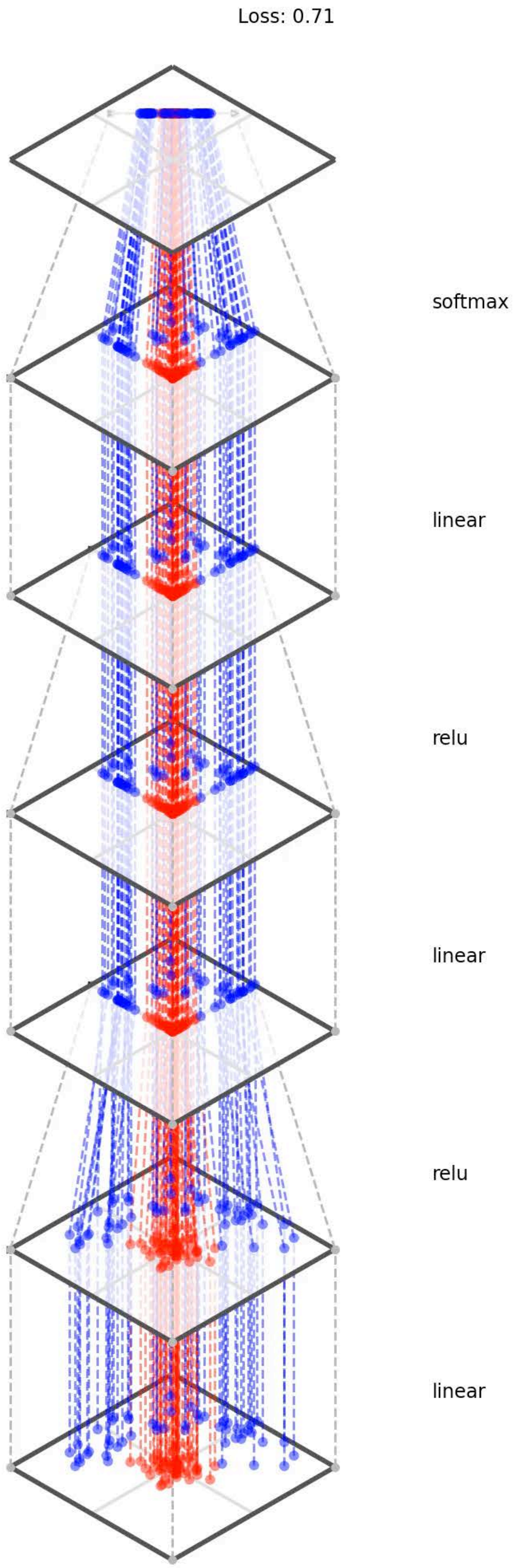


softmax
linear
relu
linear
relu
linear
Training iteration
...
...

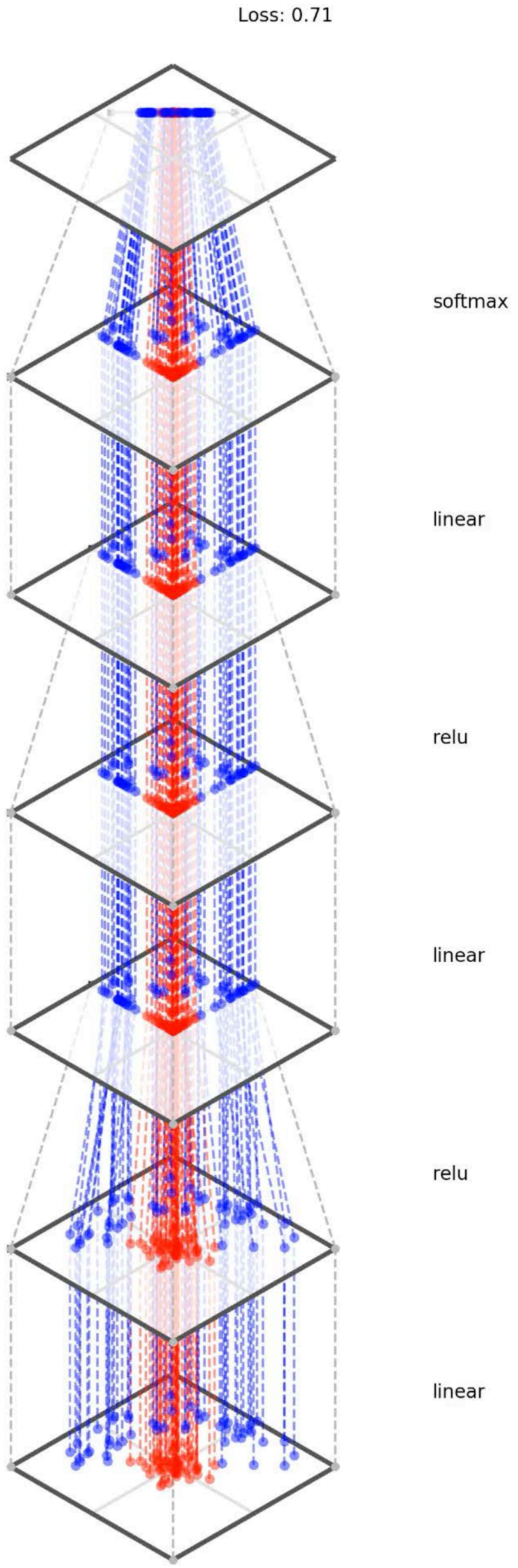
MLP



SGD
(lr=0.01)



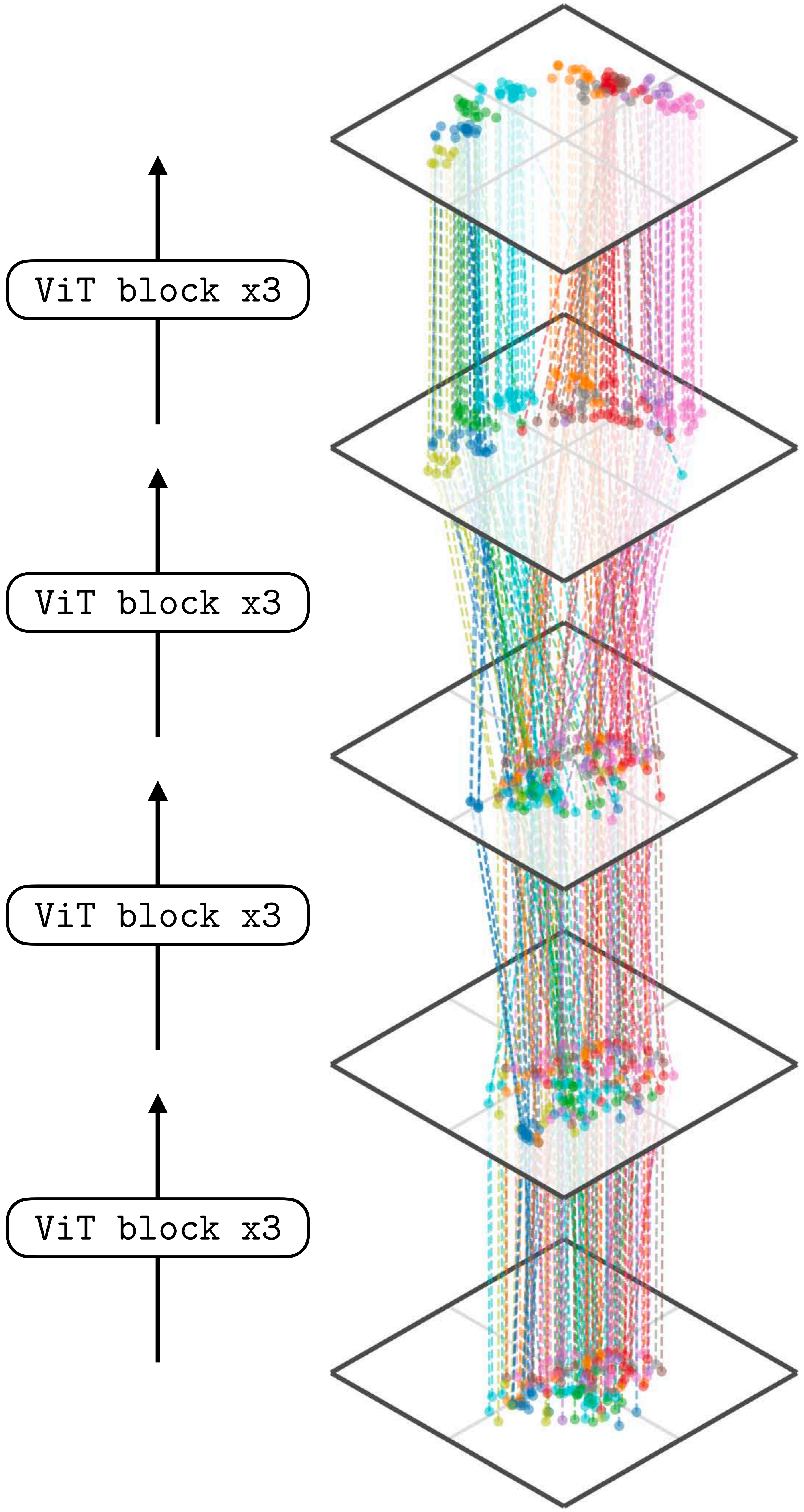
Steepest descent
in spectral norm
(see pset 2)
(lr=0.002)

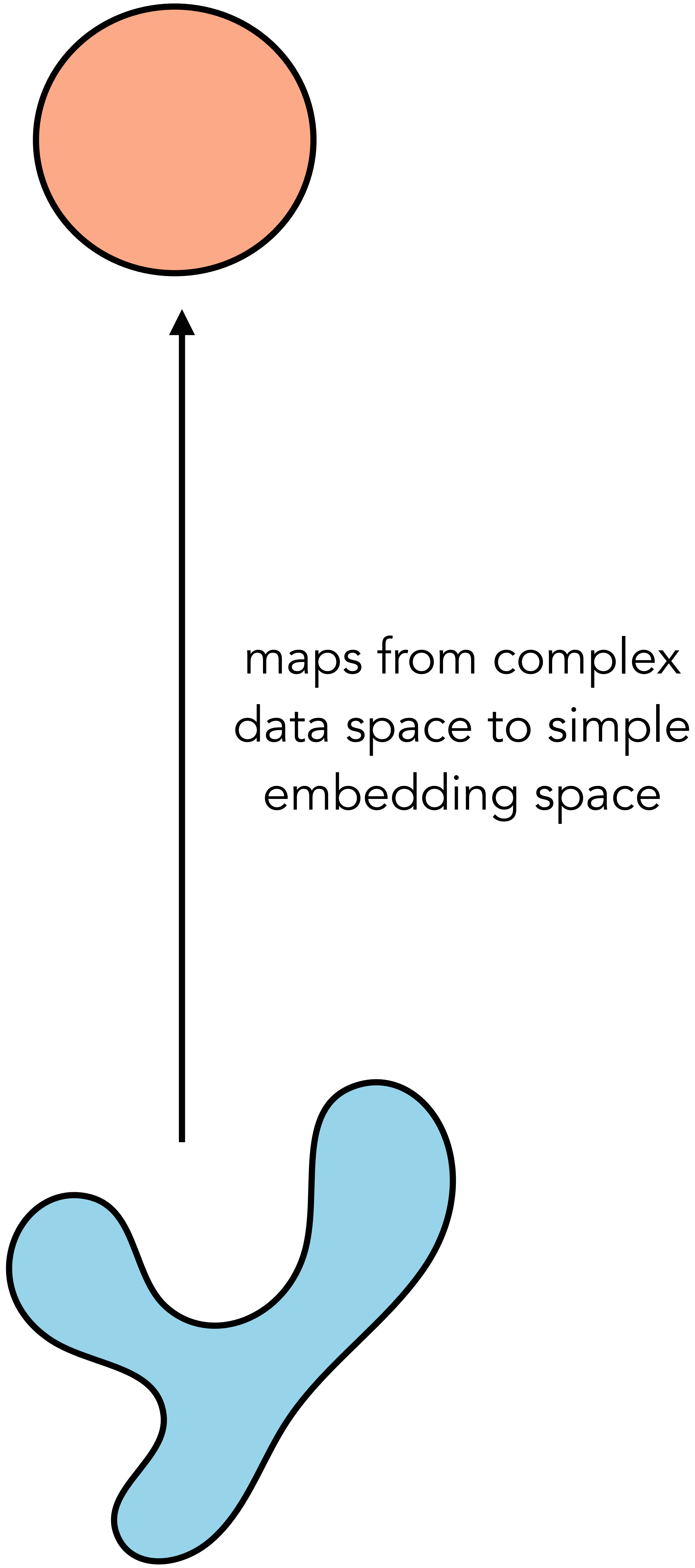


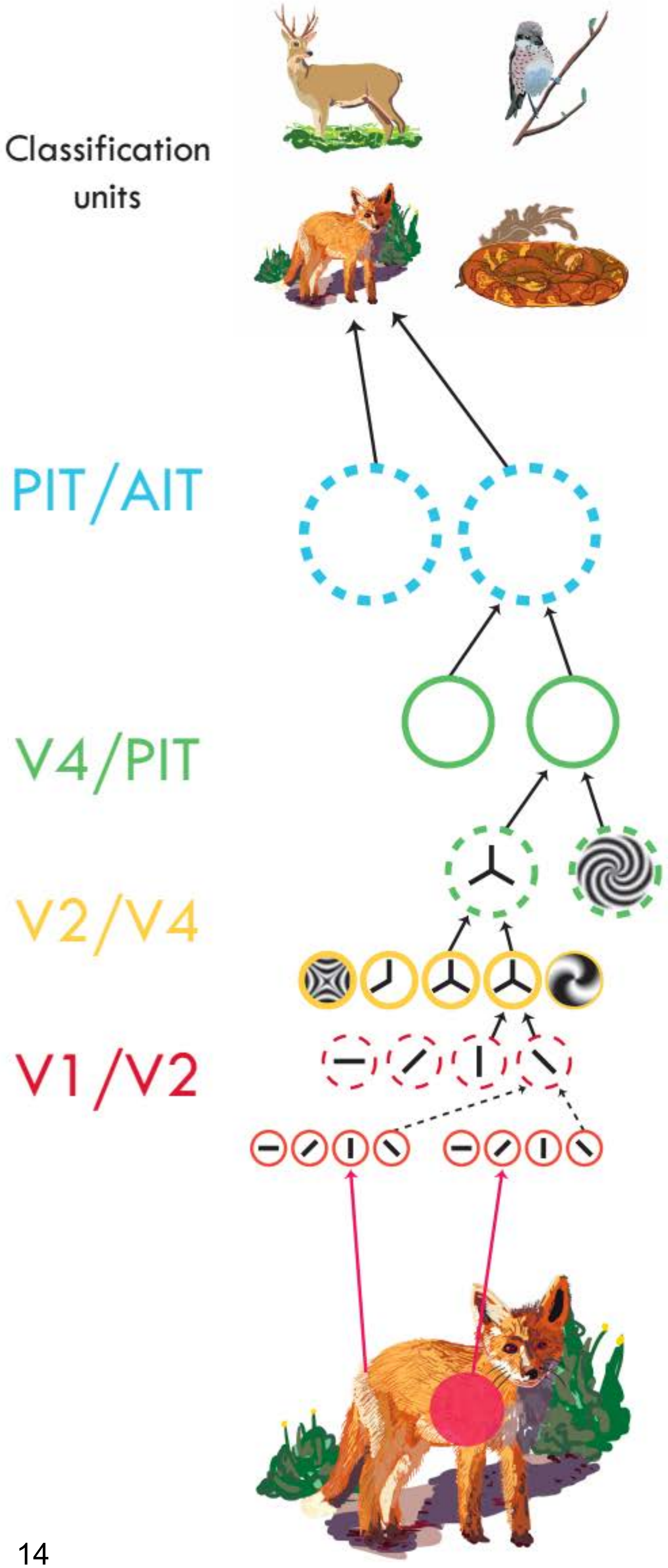
Code to make these: https://colab.research.google.com/drive/1VBw_HOQg6J2HCgozEO-ktUM_KSFaYVUD?usp=sharing

CLIP

[Radford*, Kim* et al., ICML 2021]





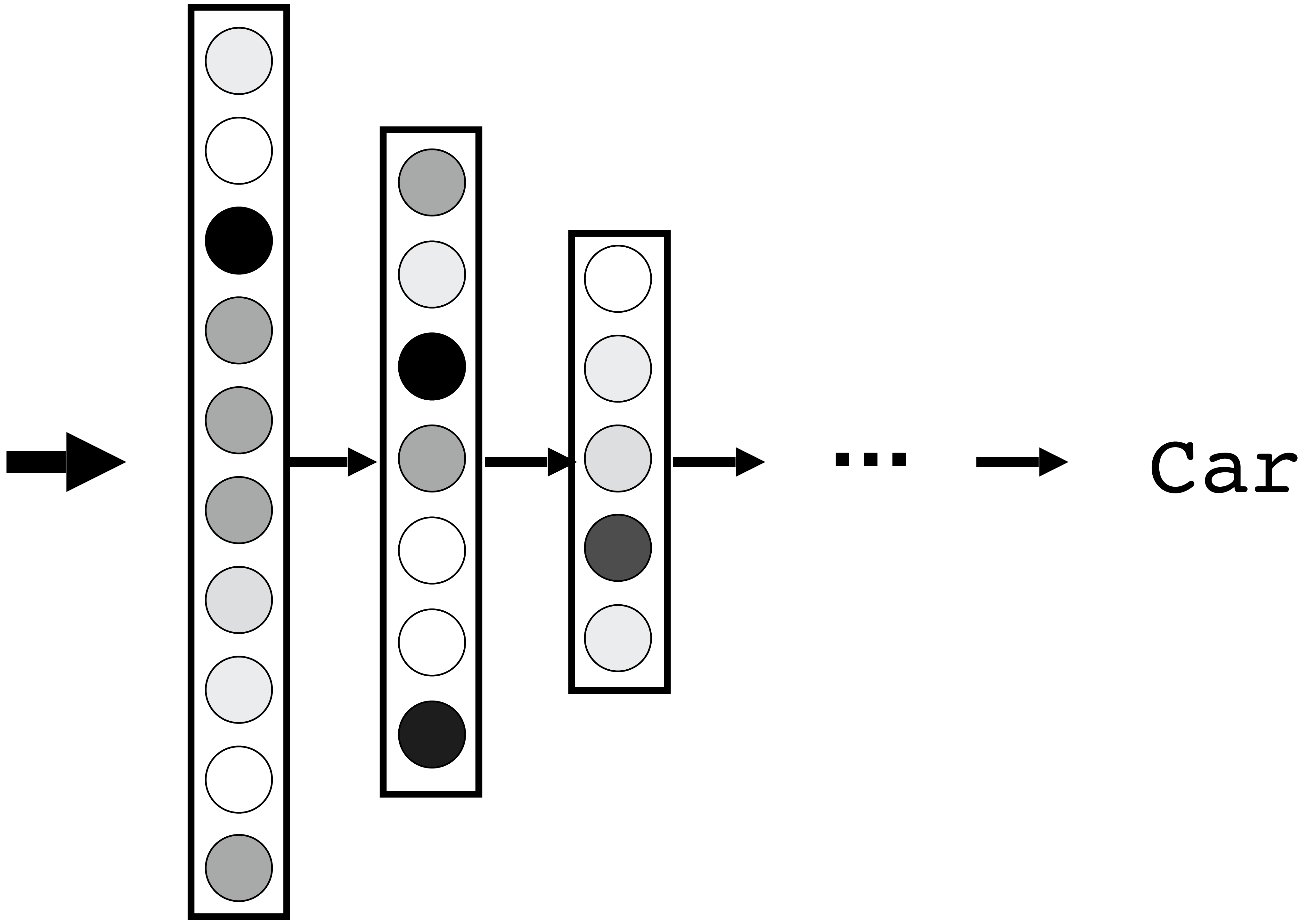


[Serre, 2014]

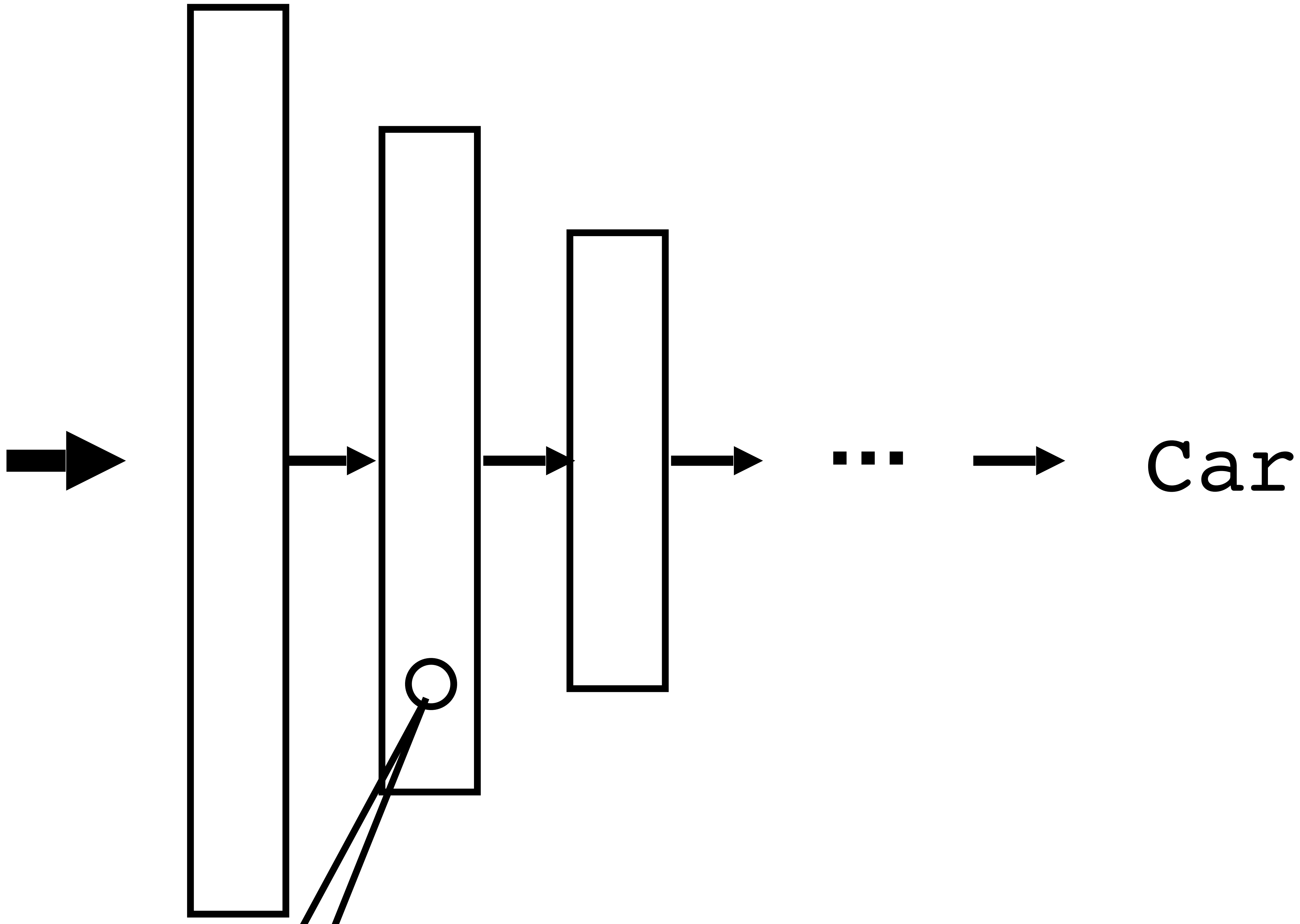
What do deep nets internally learn?

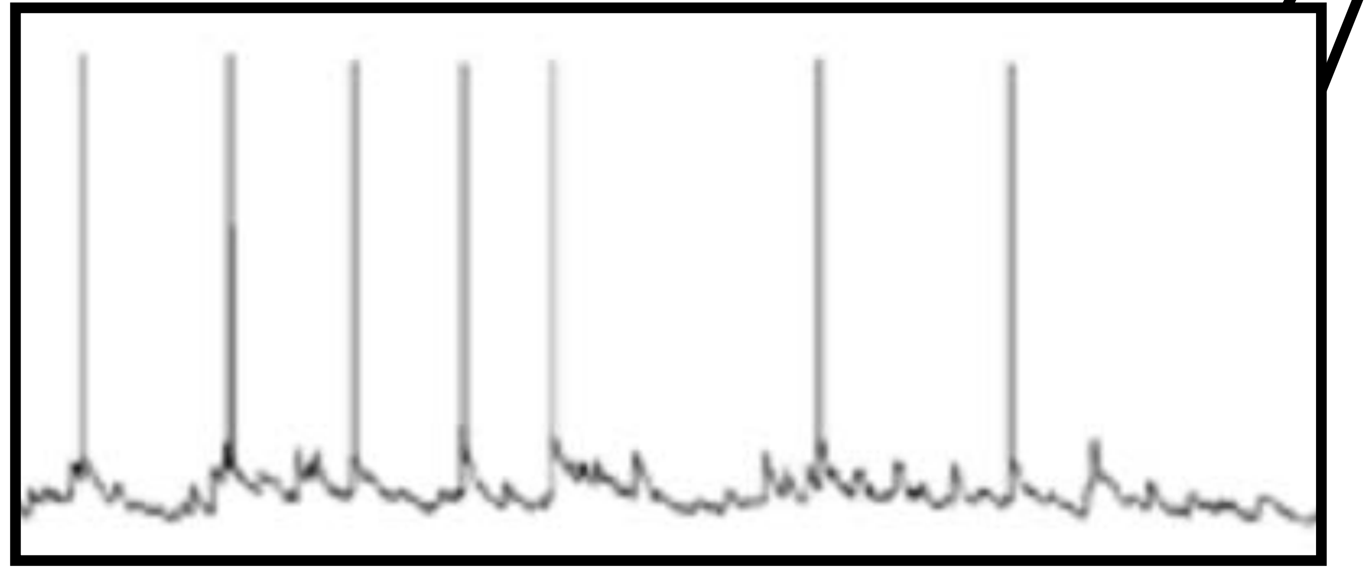
$\mathbf{x}$

Image



Deep Net "Electrophysiology"





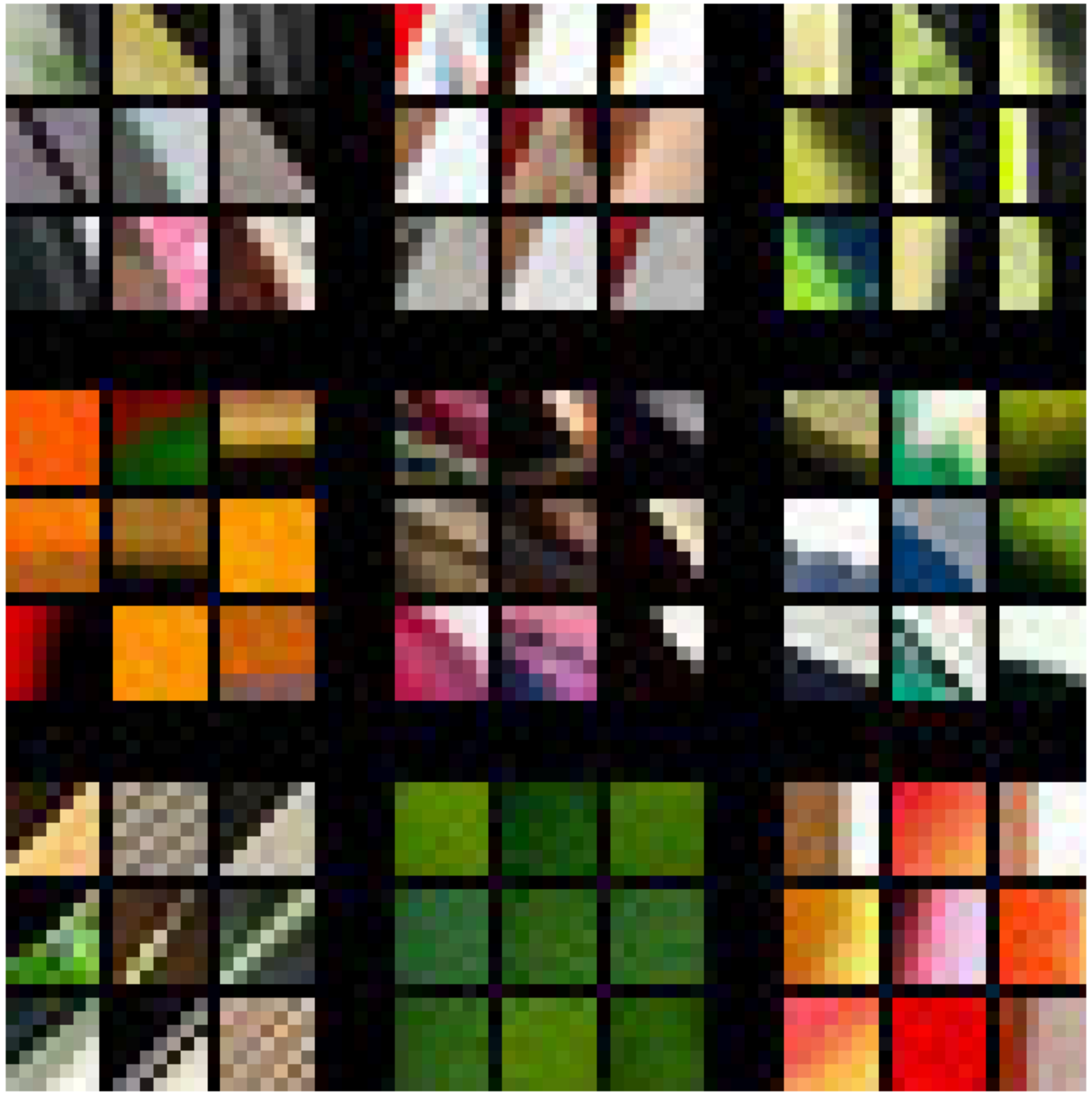
[Zeiler & Fergus, ECCV 2014]

[Zhou, Khosla, Lapedriza, Oliva, Torralba., ICLR 2015]

Visualizing and Understanding CNNs

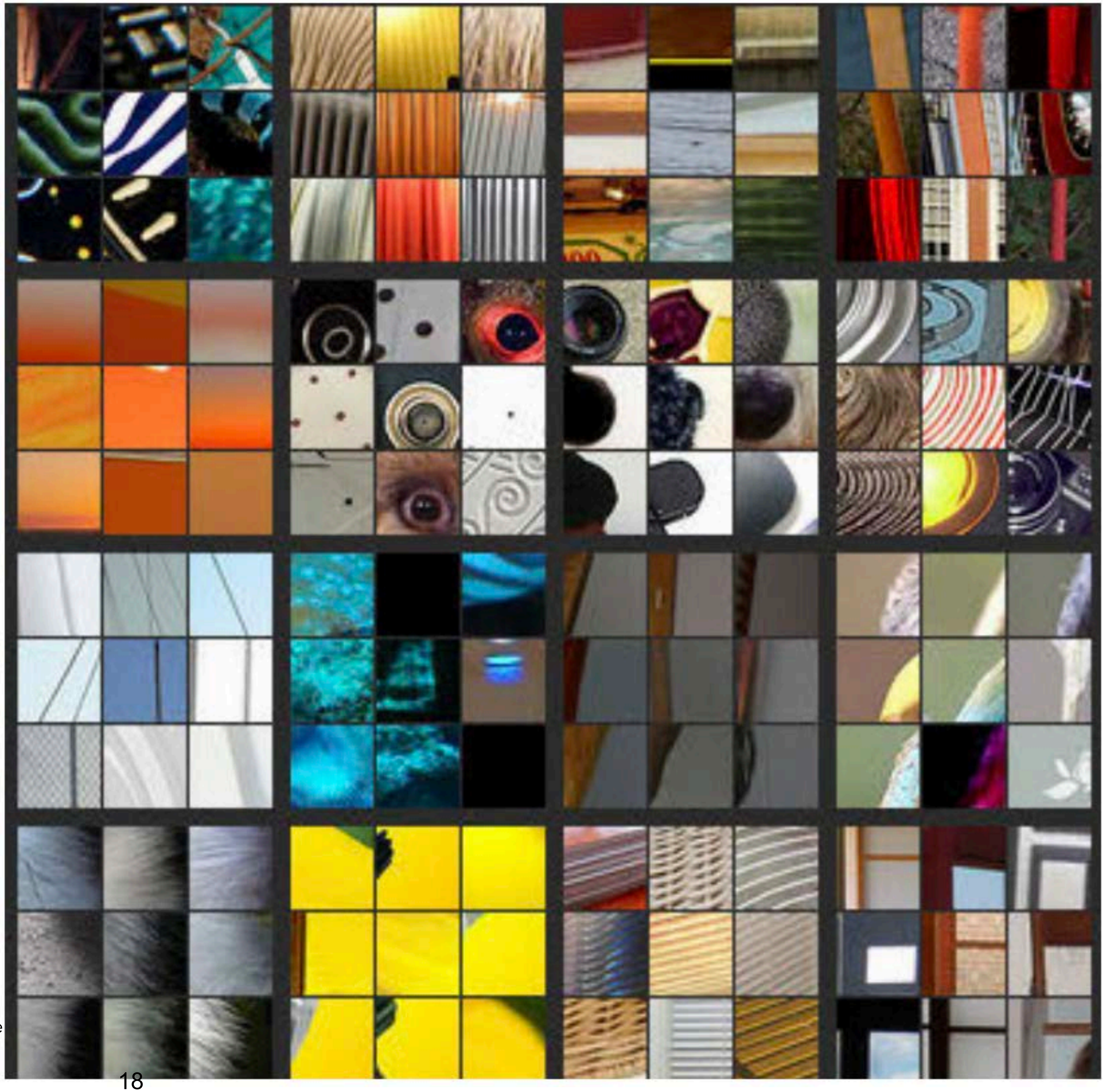
[https://arxiv.org/pdf/1311.2901]

Image patches that activate several of the **layer 1** neurons most strongly



[Zeiler and Fergus, 2014]

Image patches that activate several of the **layer 2** neurons most strongly

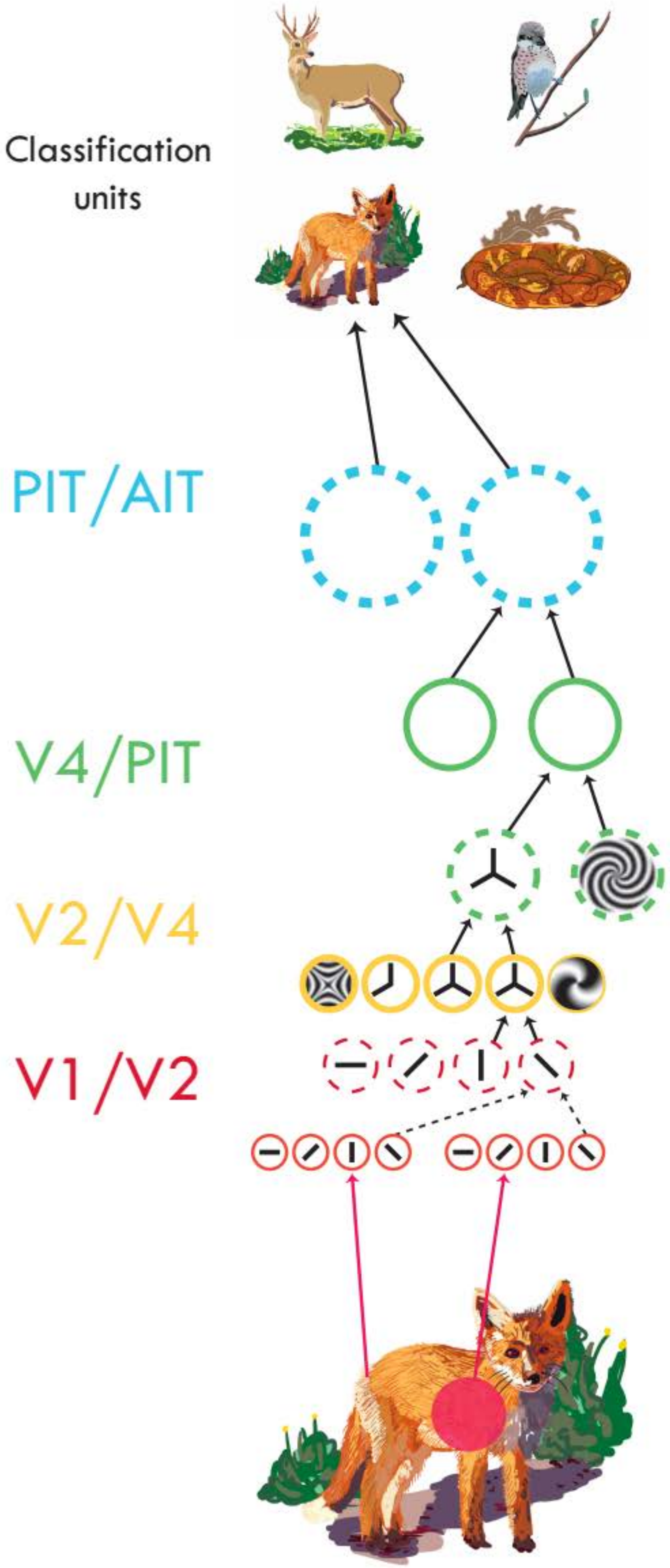


[Zeiler and Fergus, 2014]

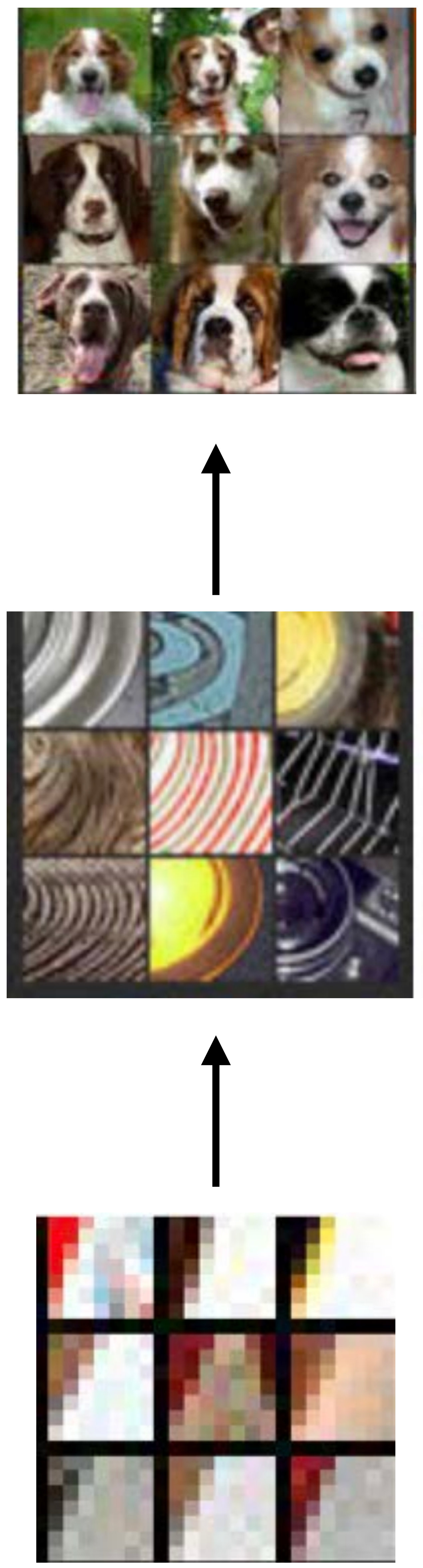
Image patches that activate several of the **layer 3** neurons most strongly

[Zeiler and Fergus, 2014]

Image patches that activate several of the **layer 5** neurons most strongly



[Serre, 2014]



[Zeiler & Fergus, ECCV 2014]

What is a representation?

Mainly, we will restrict our attention to **vector embeddings**

A representation of a data domain $\mathscr{X}$ is a function $f : \mathscr{X} \to \mathbb{R}^d$ that assigns a feature vector to each input in that domain. This function is called an **encoder**.

A representation of a datapoint $\mathbf{x}$ is a vector $\mathbf{z} \in \mathbb{R}^d$ with $\mathbf{z} = f(\mathbf{x})$.

Why learn representations?

To do more learning! (aka **Transfer learning**)

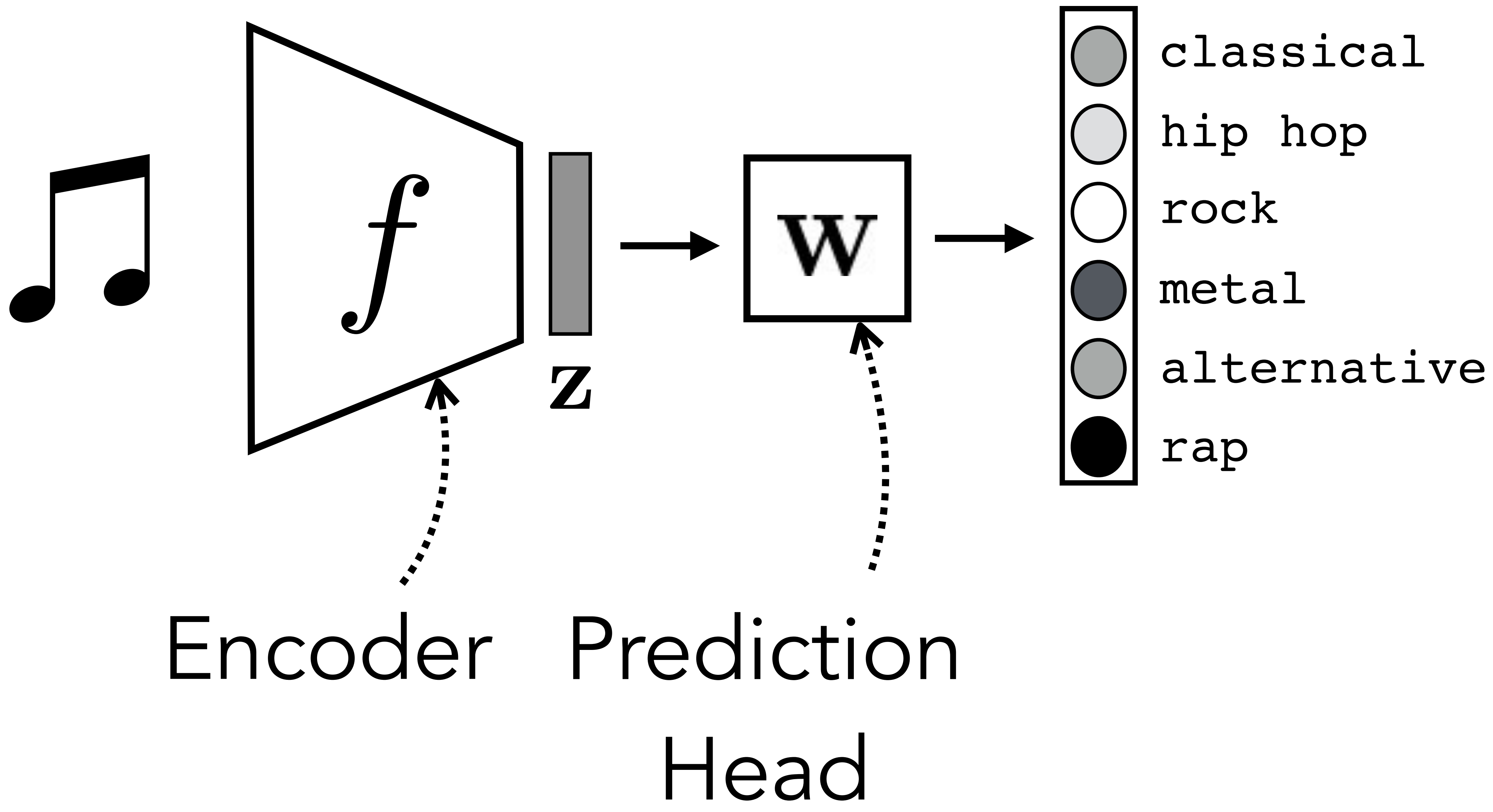
"Generally speaking, a good representation is one that makes a subsequent learning task easier." — *Deep Learning*, Goodfellow et al. 2016

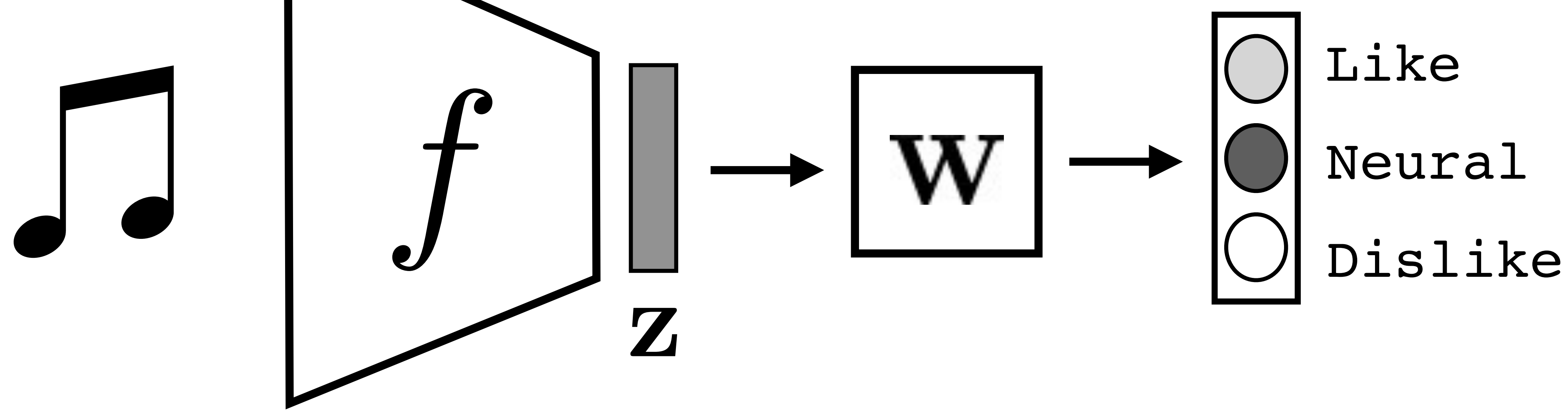
Training

Testing

Genre recognition

Preference prediction

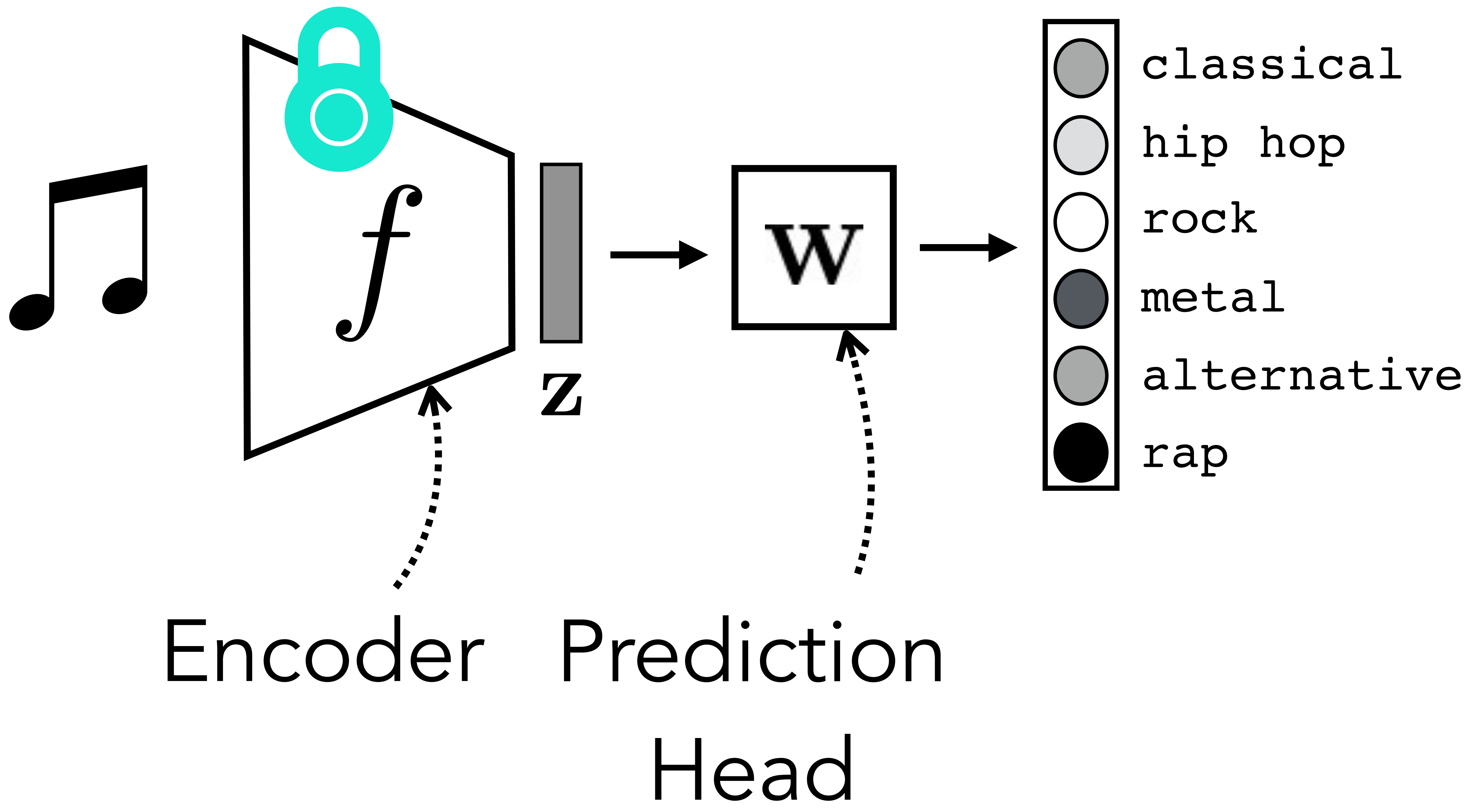




Often, what we will be "tested" on is not what we were trained on.

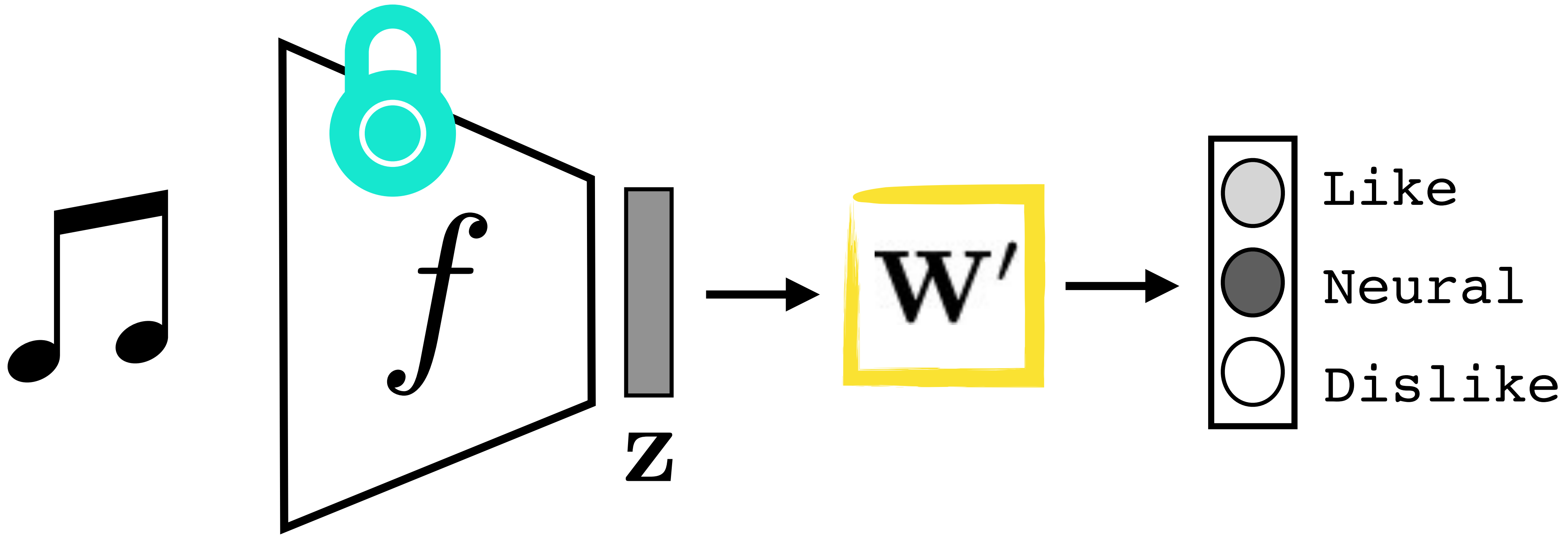
Training

Genre recognition



Adapting

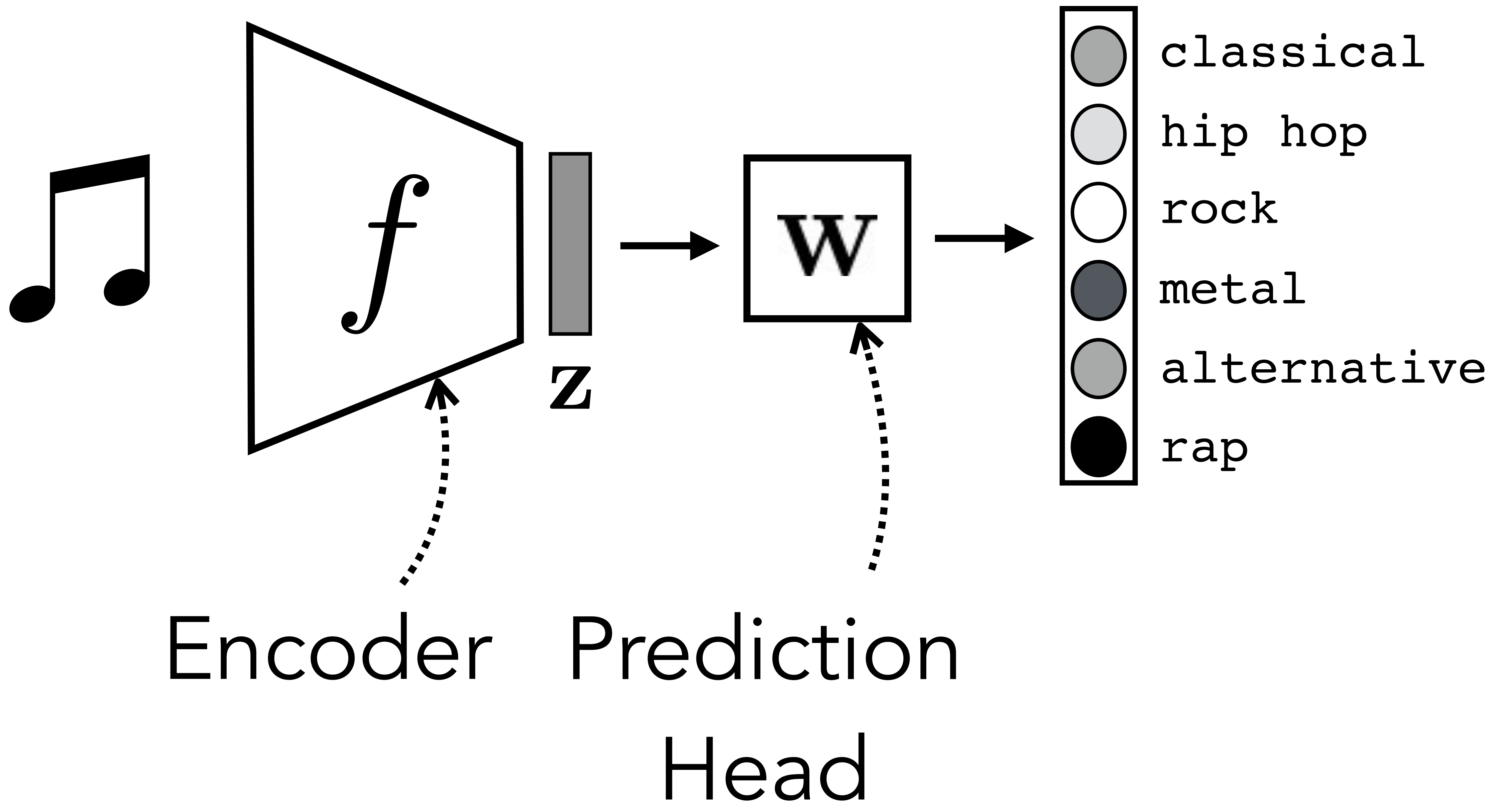
Preference prediction



Linear adaptation: freeze f, train a new linear map to new target data

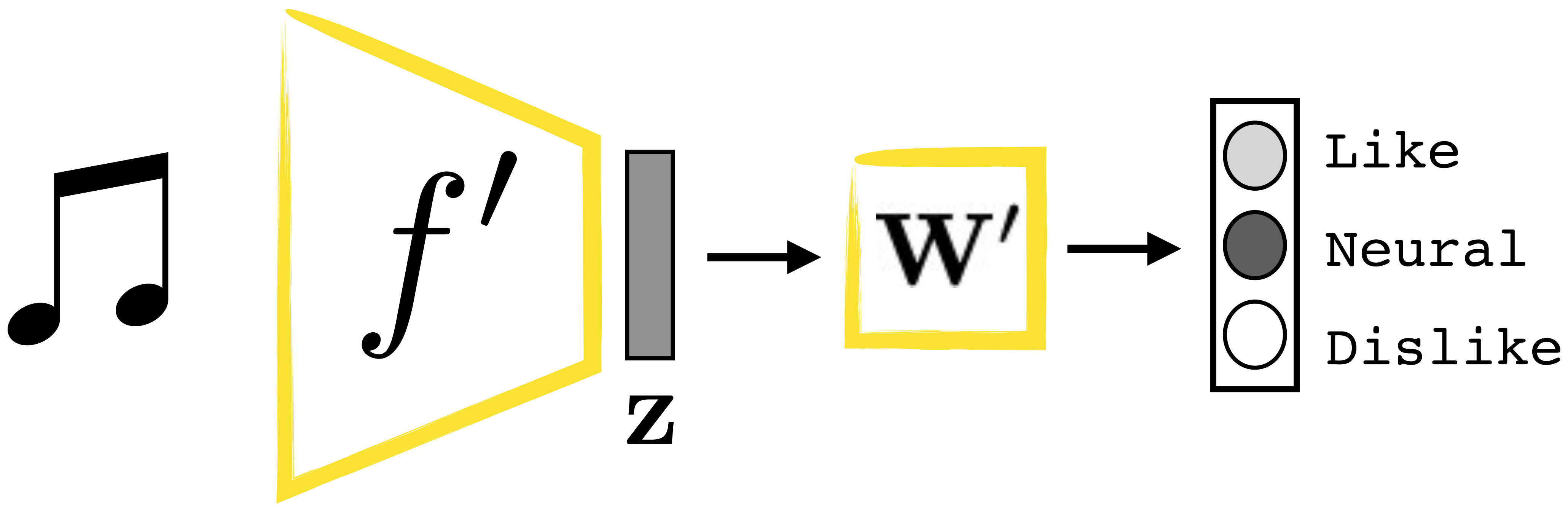
Training

Genre recognition



Adapting

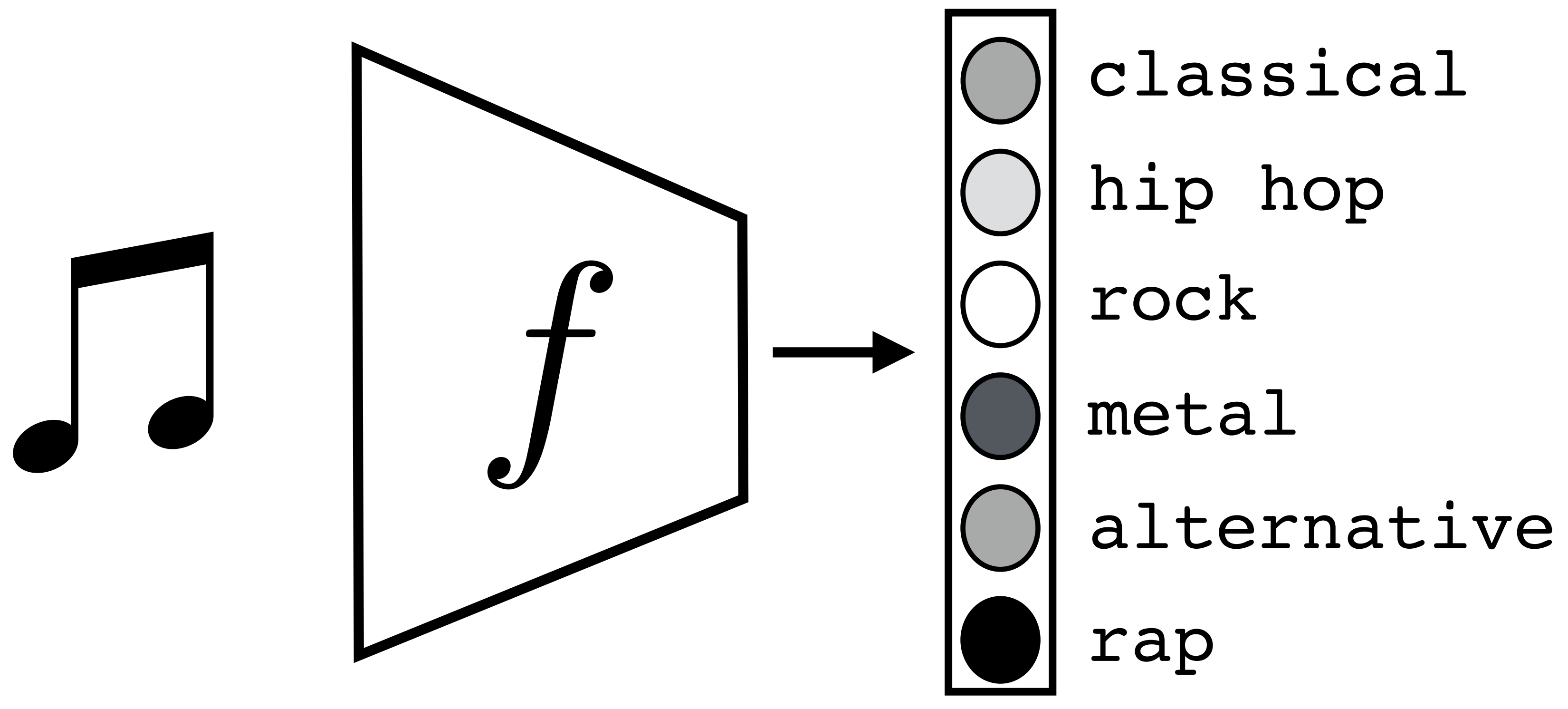
Preference prediction



Finetuning: initialize f′ as f, then continue training on new target data

Pretraining

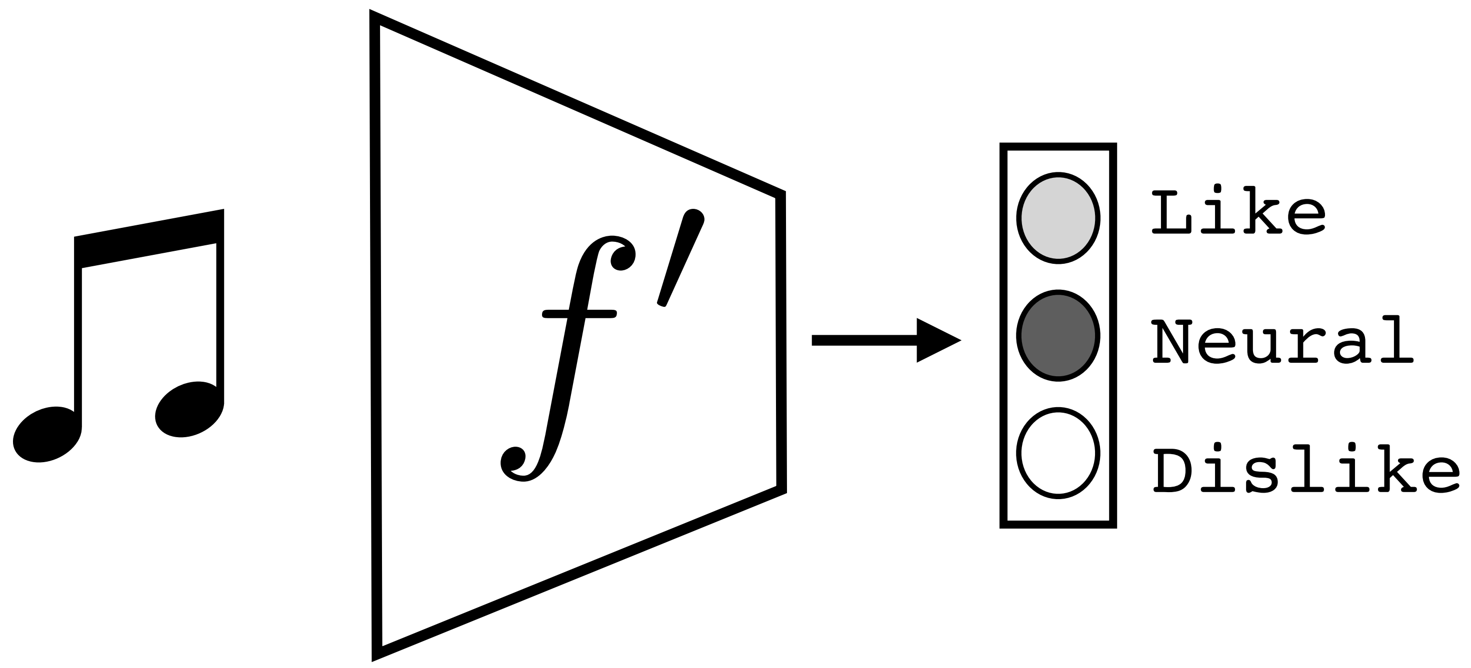
Genre recognition



A lot of data

Adapting

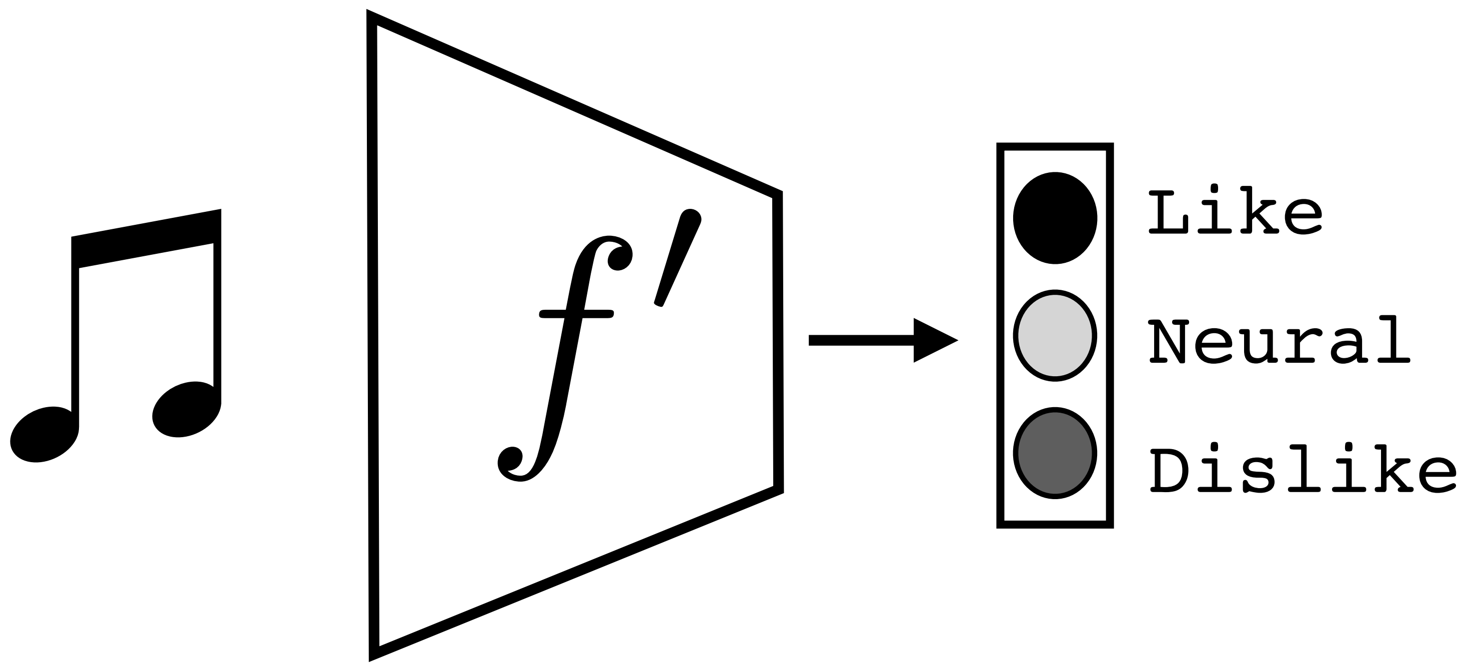
Preference prediction



A little data

Testing

Preference prediction



Finetuning

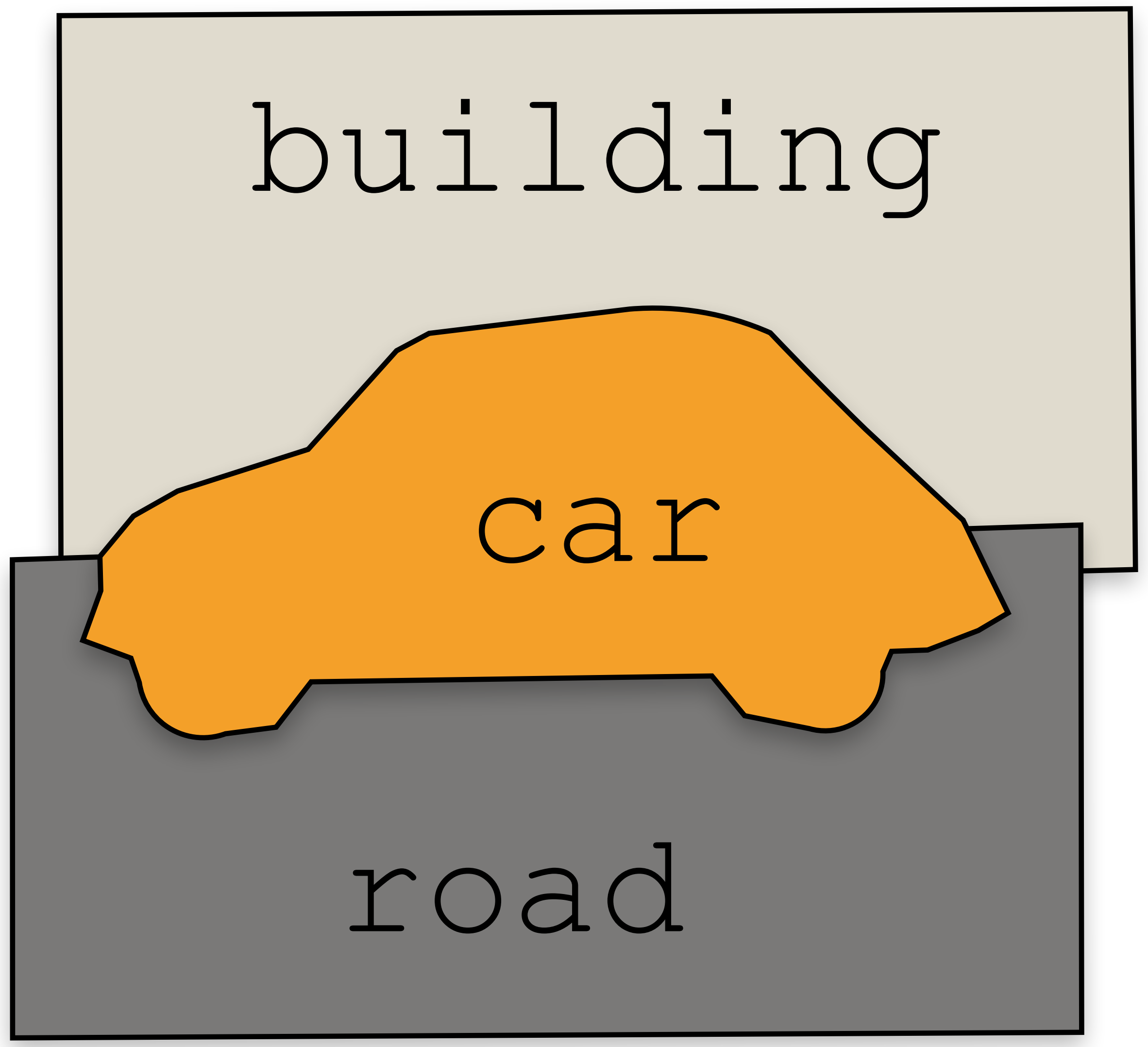
- Pretrain a network on task A, resulting in parameters **W** and **b**
- Initialize a second network with some or all of **W** and **b**
- Train the second network on task B, resulting in parameters **W'** and **b'**

How do you learn a good representation?

Representation learning

Good representations are:

1. Compact (*minimal*)
2. Explanatory (*sufficient*)
3. Disentangled (*independent factors*)
4. Interpretable
5. *Make subsequent problem solving easy*
6. …?



[See "Representation Learning", Bengio 2013, for more commentary]

Learning from examples

(aka **supervised learning**)

Training data

$$\{x^{(1)}, y^{(1)}\}$$
$$\{x^{(2)}, y^{(2)}\} \quad \rightarrow \quad \boxed{\text{Learner}} \quad \rightarrow \quad f : X \rightarrow Y$$
$$\{x^{(3)}, y^{(3)}\}$$
$$\ldots$$

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}^{(i)}), \mathbf{y}^{(i)})$$

Learning without examples

(includes **unsupervised learning** / **self-supervised learning**)

Data

$$\{x^{(1)}\}$$
$$\{x^{(2)}\} \quad \rightarrow \quad \text{Learner} \quad \rightarrow \quad ?$$
$$\{x^{(3)}\}$$
$$\ldots$$

Representation Learning

Data

$\{x^{(1)}\}$

$\{x^{(2)}\}$ $\rightarrow$ Learner $\rightarrow$

$\{x^{(3)}\}$

$\dots$

embeddings

clusters

metrics

...

Two basic approaches: 1) compression, 2) prediction

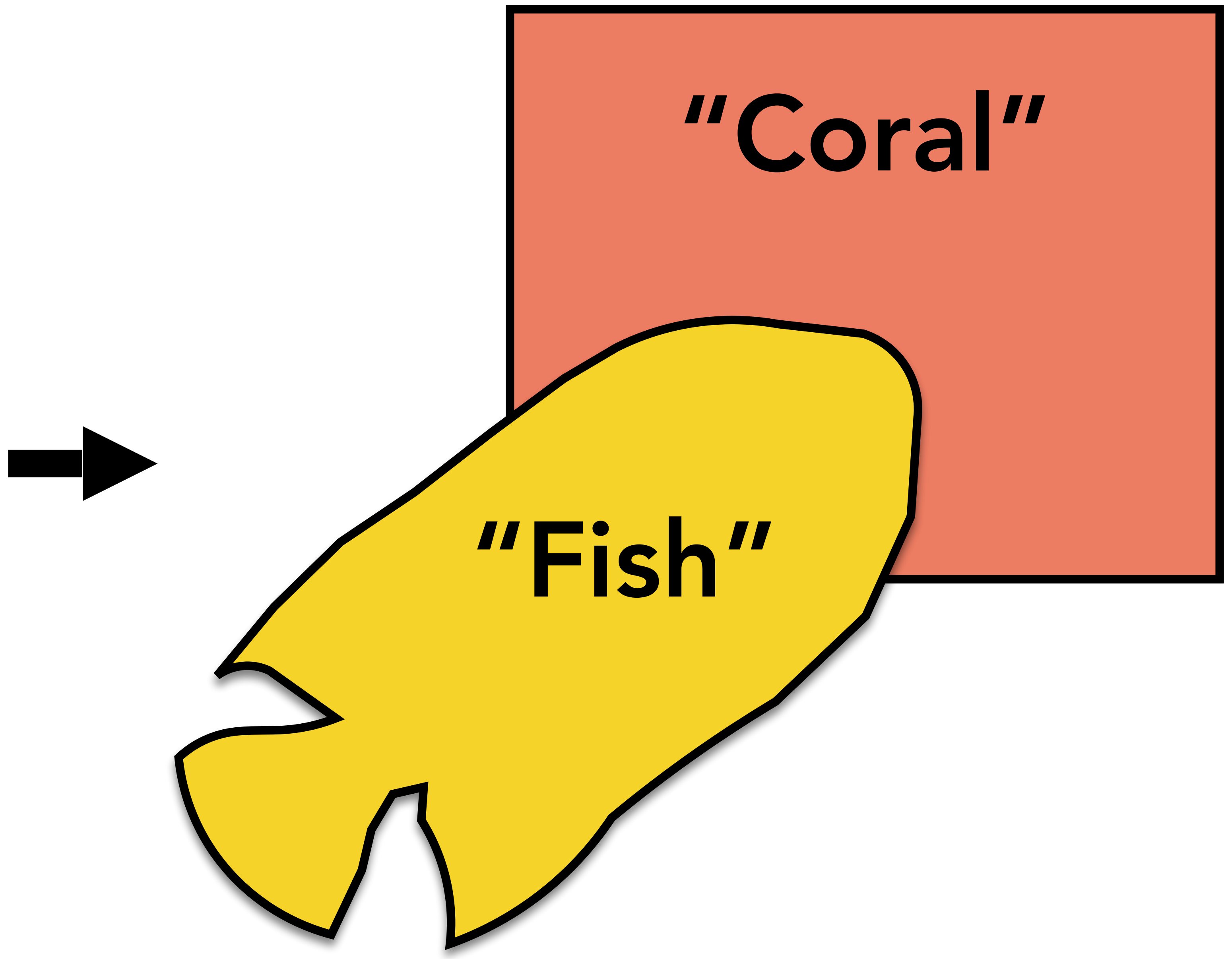
Learning Method	Learning Principle	Short Summary
Autoencoding	Compression	Remove redundant information
Contrastive	Compression	Achieve invariance to viewing transformations
Clustering	Compression	Quantize continuous data into discrete categories
Future prediction	Prediction	Predict the future
Imputation	Prediction	Predict missing data
Pretext tasks	Prediction	Predict abstract properties of your data

(Question: are these actually different?)

Learning via compression

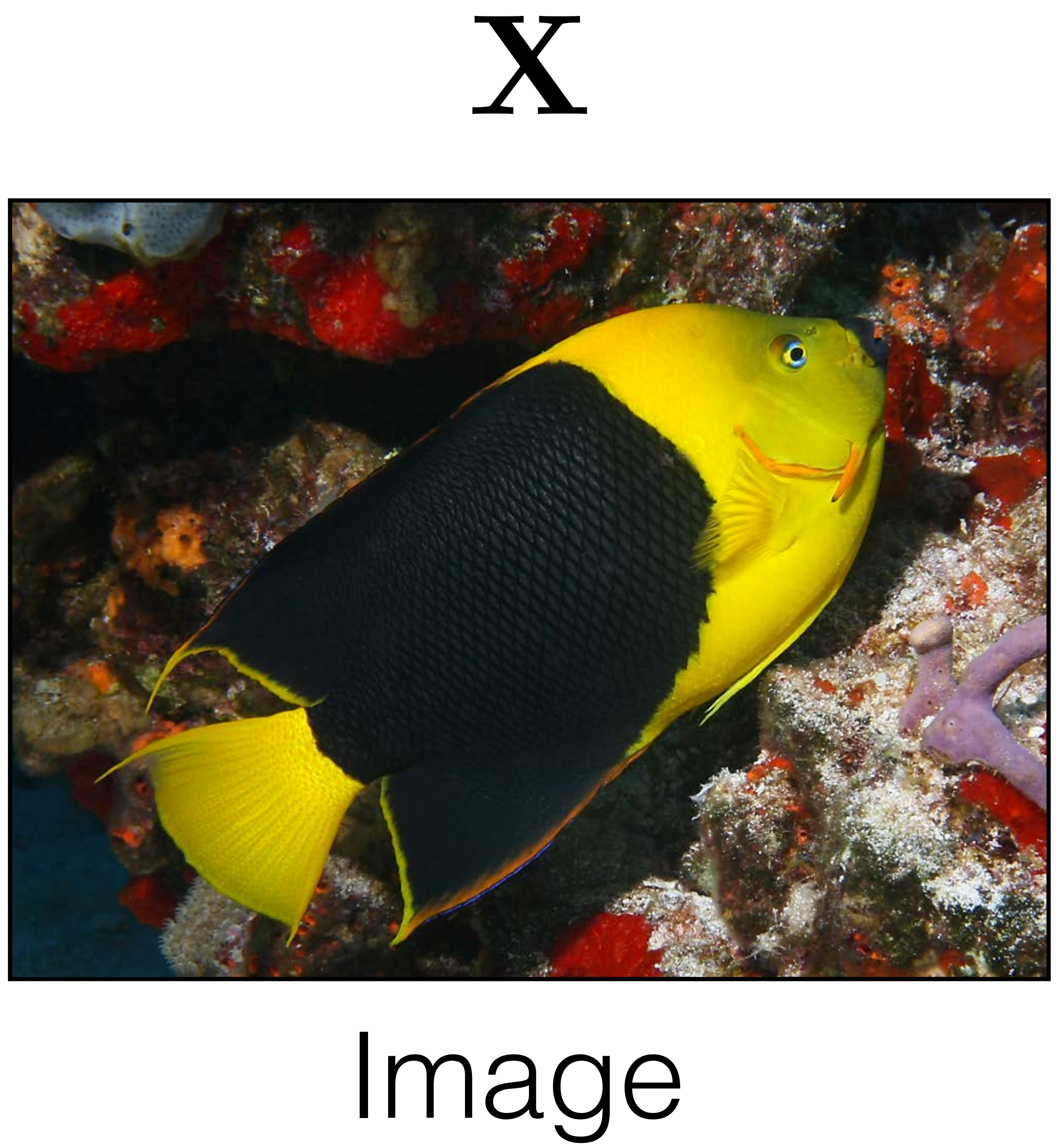
$\mathbf{x}$

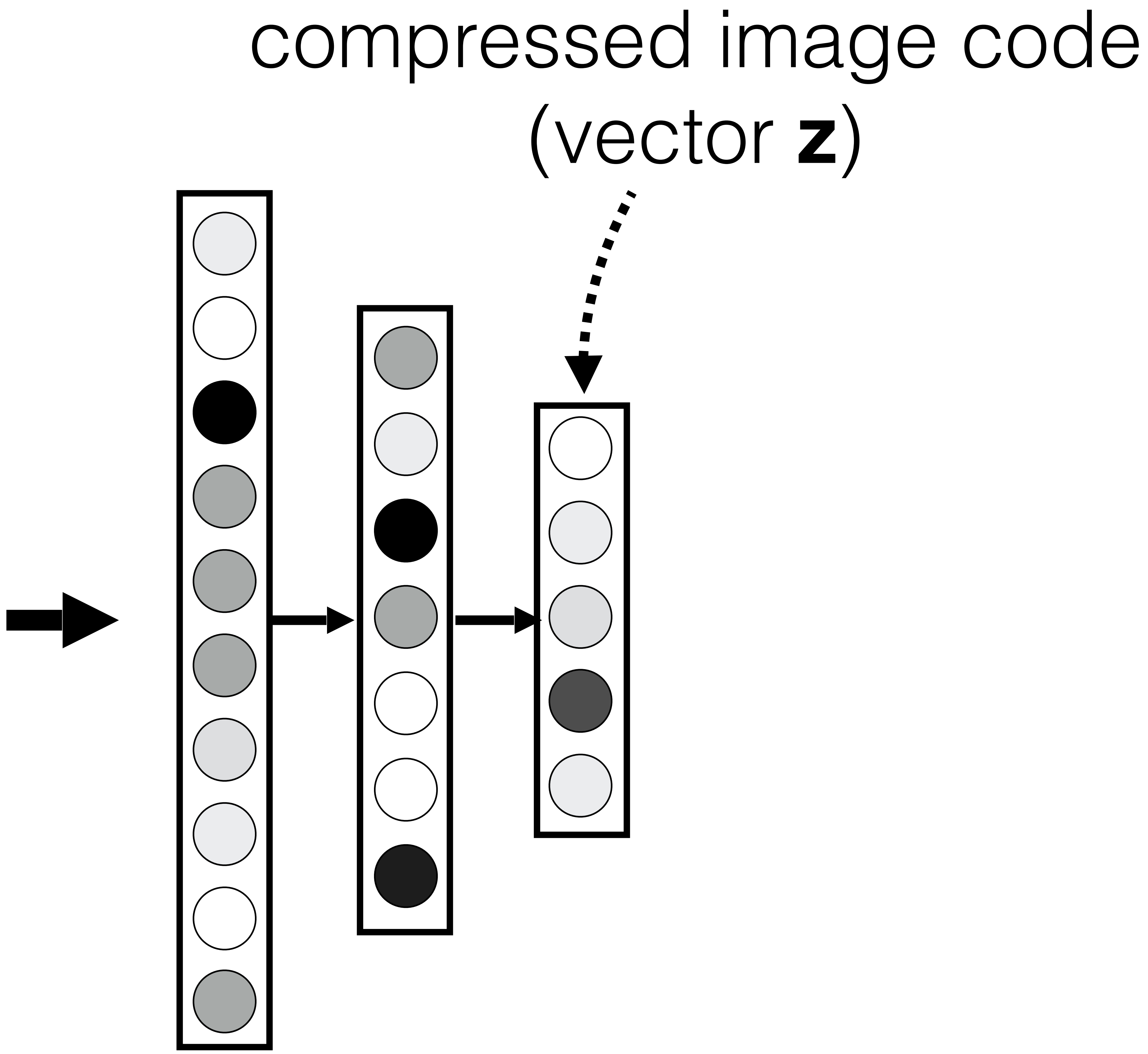
Image



Compact mental representation

Learning via compression





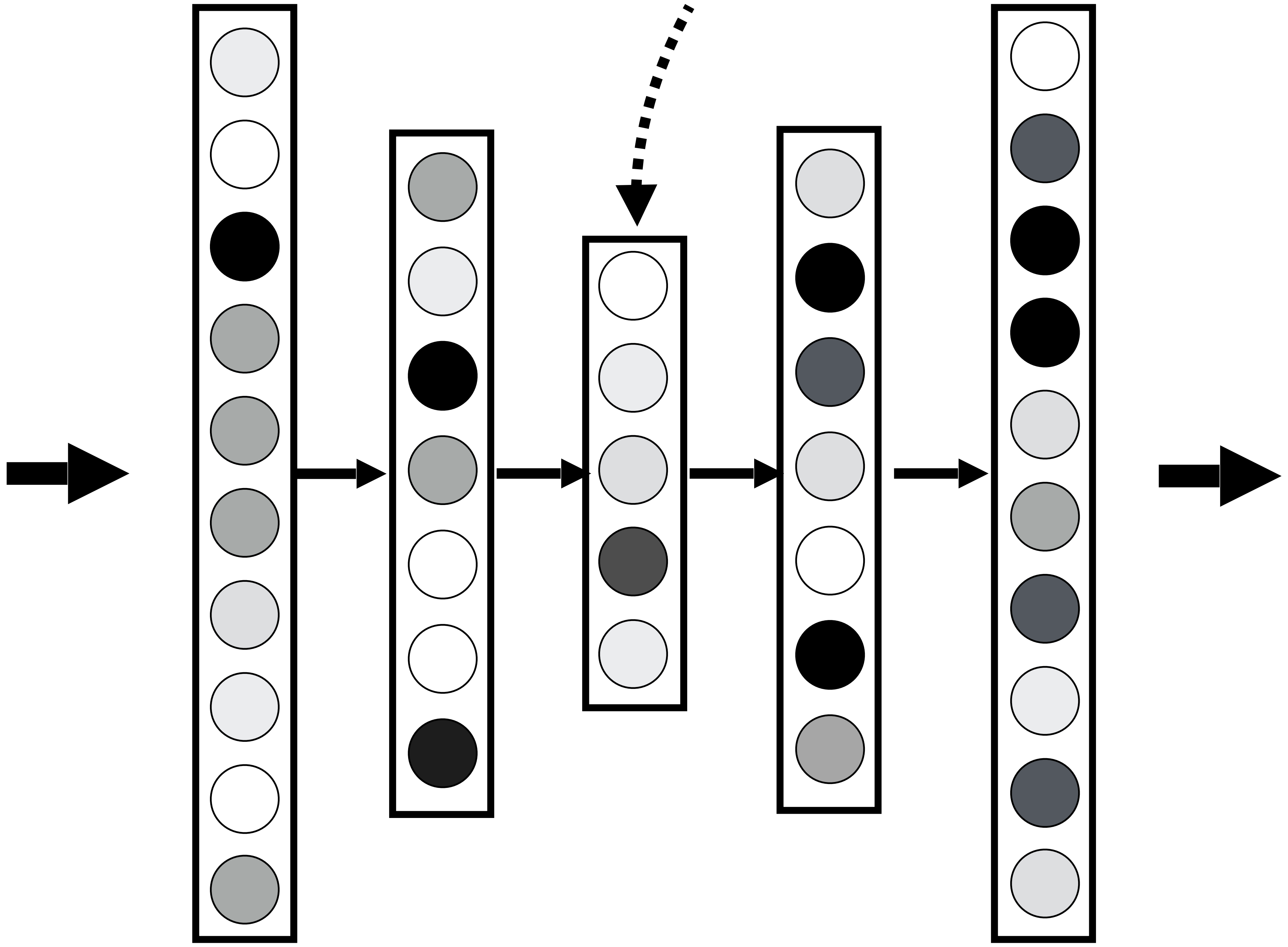
Learning via compression

$\mathbf{x}$

Image

compressed image code

(vector $\mathbf{z}$)



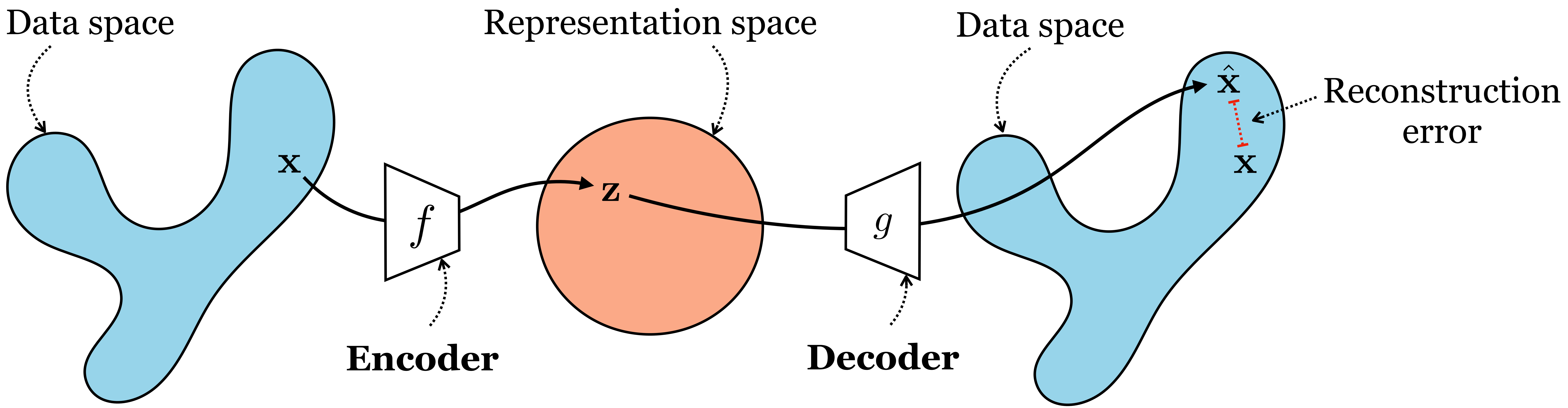
"Autoencoder"

$\hat{\mathbf{x}}$

Reconstructed image

[e.g., Hinton & Salakhutdinov, Science 2006]

Autoencoder



$$f^*, g^* = \arg\min_{f,g} \mathbb{E}_{\mathbf{x}} \left\| \mathbf{x} - g(f(\mathbf{x})) \right\|_2^2$$

Data

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

L_2 autoencoder

Objective

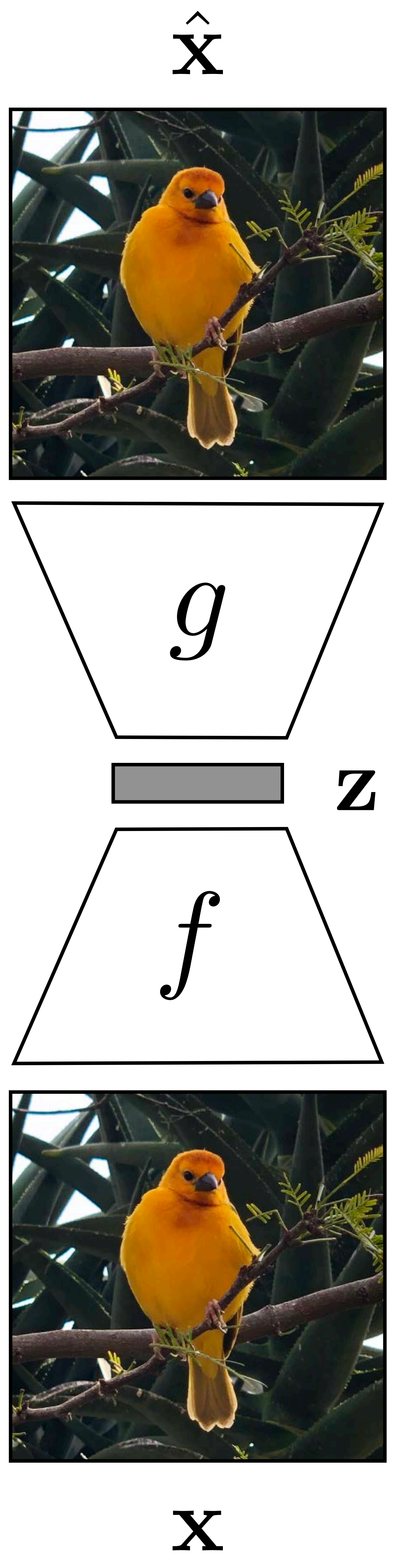
$$\mathcal{L}(F(\mathbf{x}), \mathbf{x}) = \|F(\mathbf{x}) - \mathbf{x}\|_2^2$$

Hypothesis space

$$F = g \circ f : \mathbb{R}^N \rightarrow \mathbb{R}^M \rightarrow \mathbb{R}^N$$

$\rightarrow f$

Typically, M<N



Data

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

L_2 autoencoder

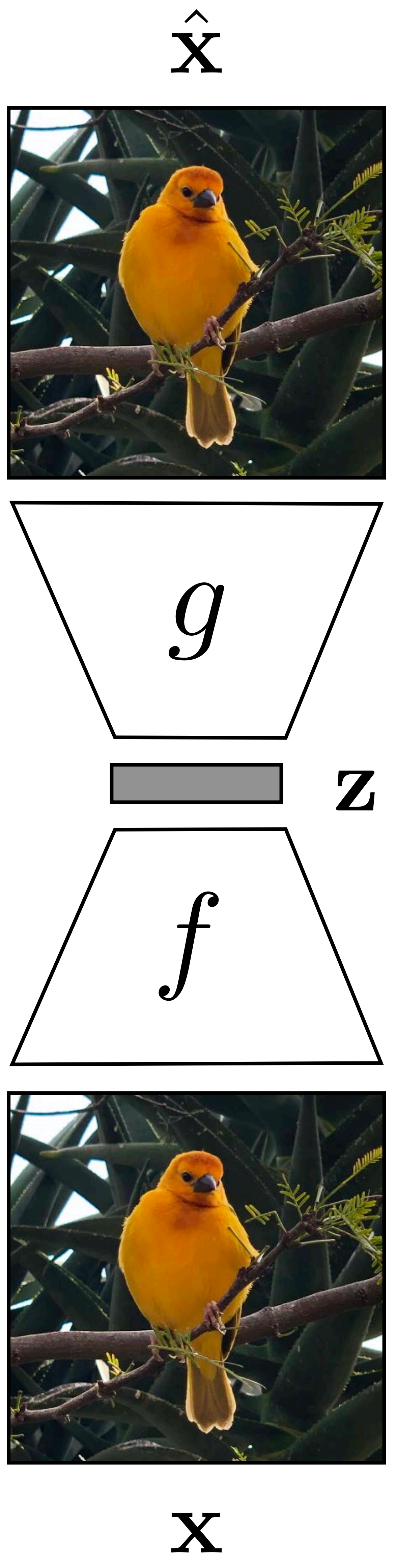
Objective

$$\mathcal{L}(F(\mathbf{x}), \mathbf{x}) = \|F(\mathbf{x}) - \mathbf{x}\|_2^2$$

Hypothesis space

$$F = g \circ f : \mathbb{R}^N \rightarrow \mathbb{R}^M \rightarrow \mathbb{R}^N$$

$\rightarrow f$

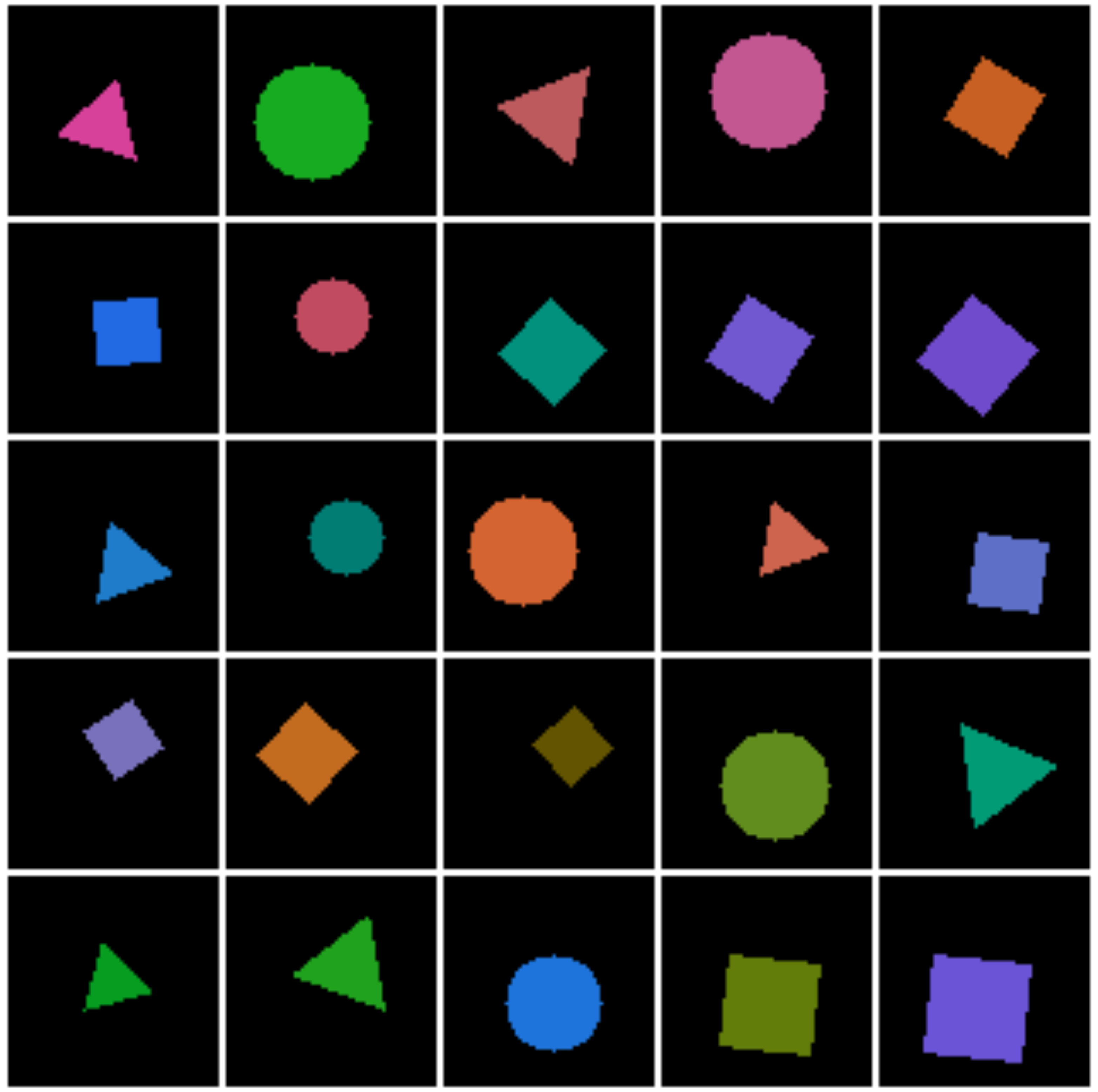


What if f and g are both linear?

Then the embedding spans the same M-dimensional subspace as PCA

Quick experiment

Data



$\rightarrow$

L_2 autoencoder

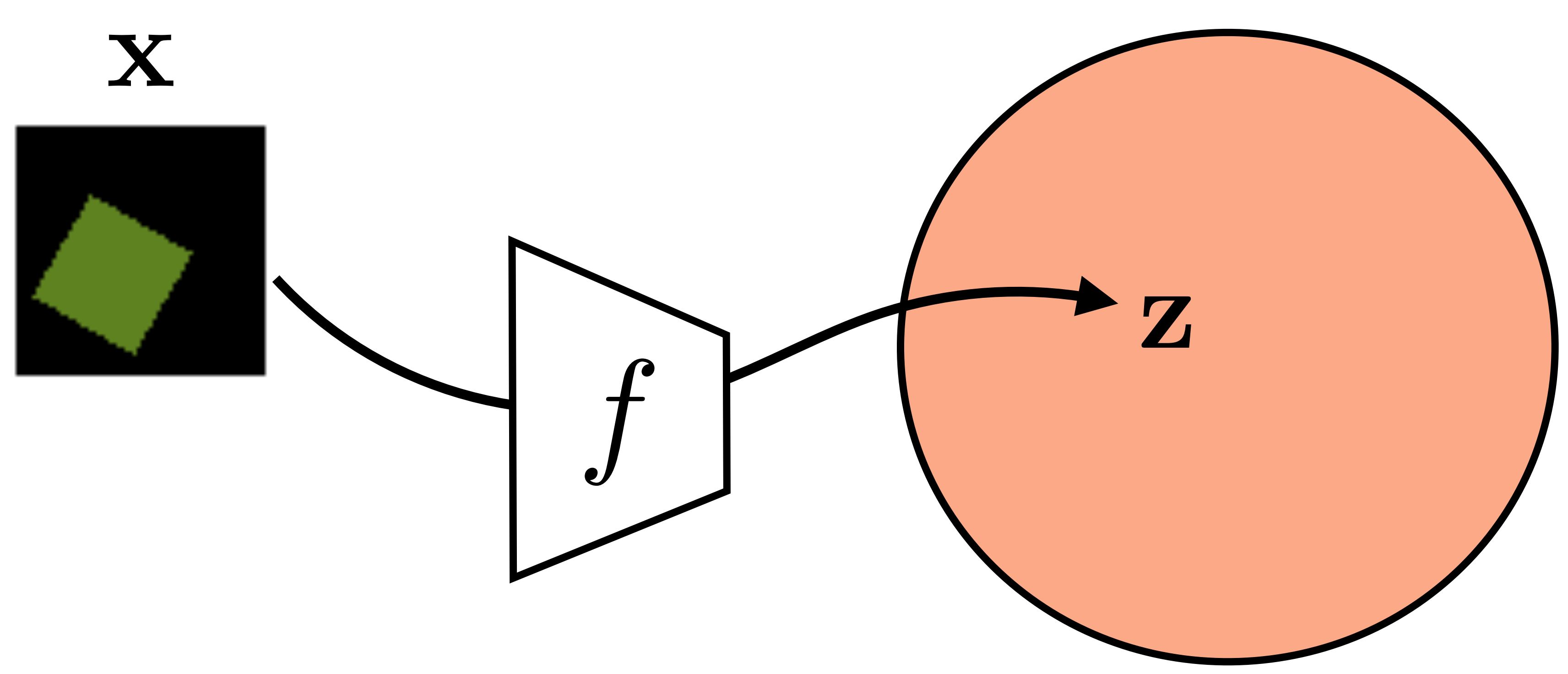
Objective

$$\mathcal{L}(F(\mathbf{x}), \mathbf{x}) = \|F(\mathbf{x}) - \mathbf{x}\|_2^2$$

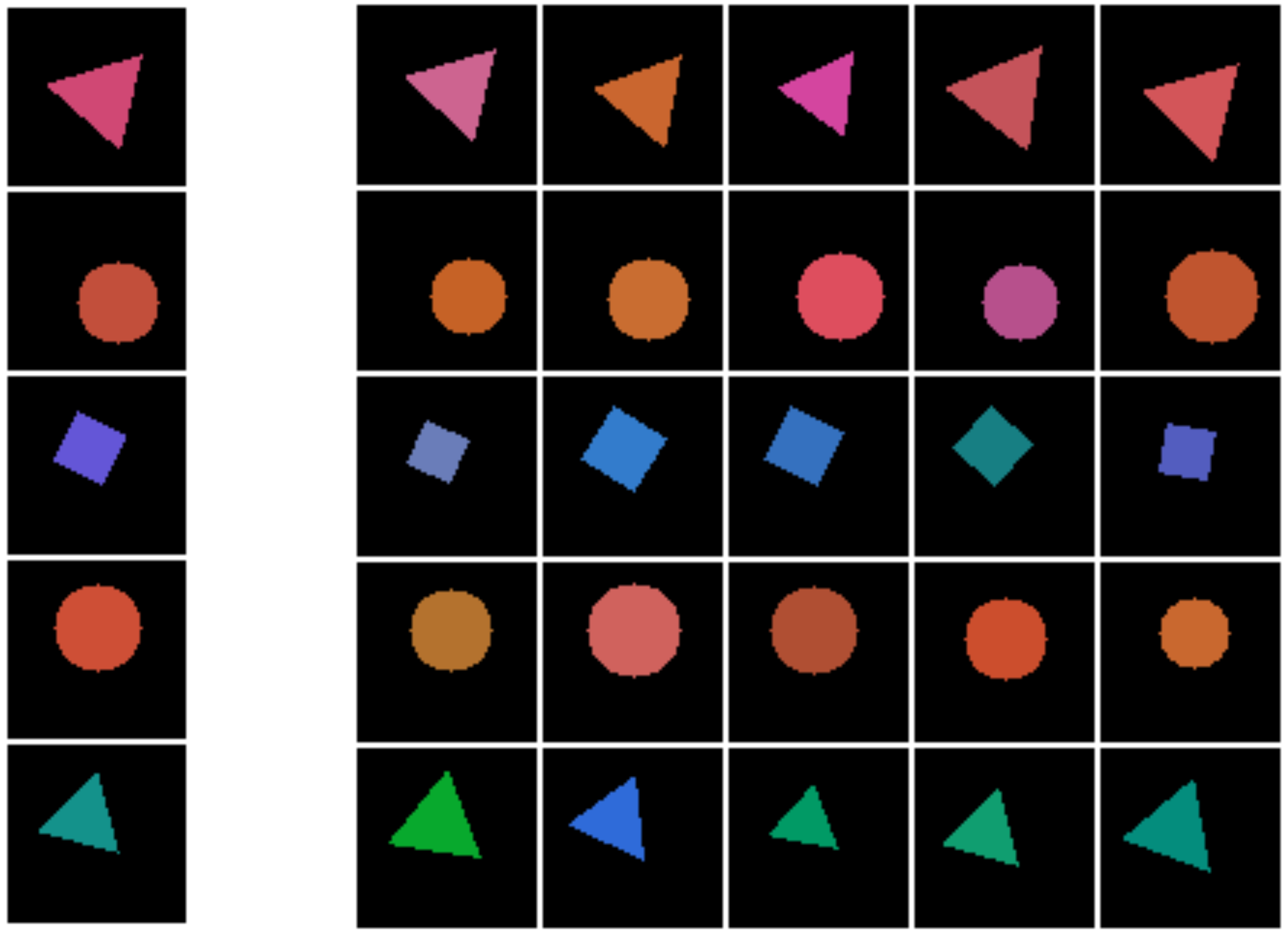
Hypothesis space

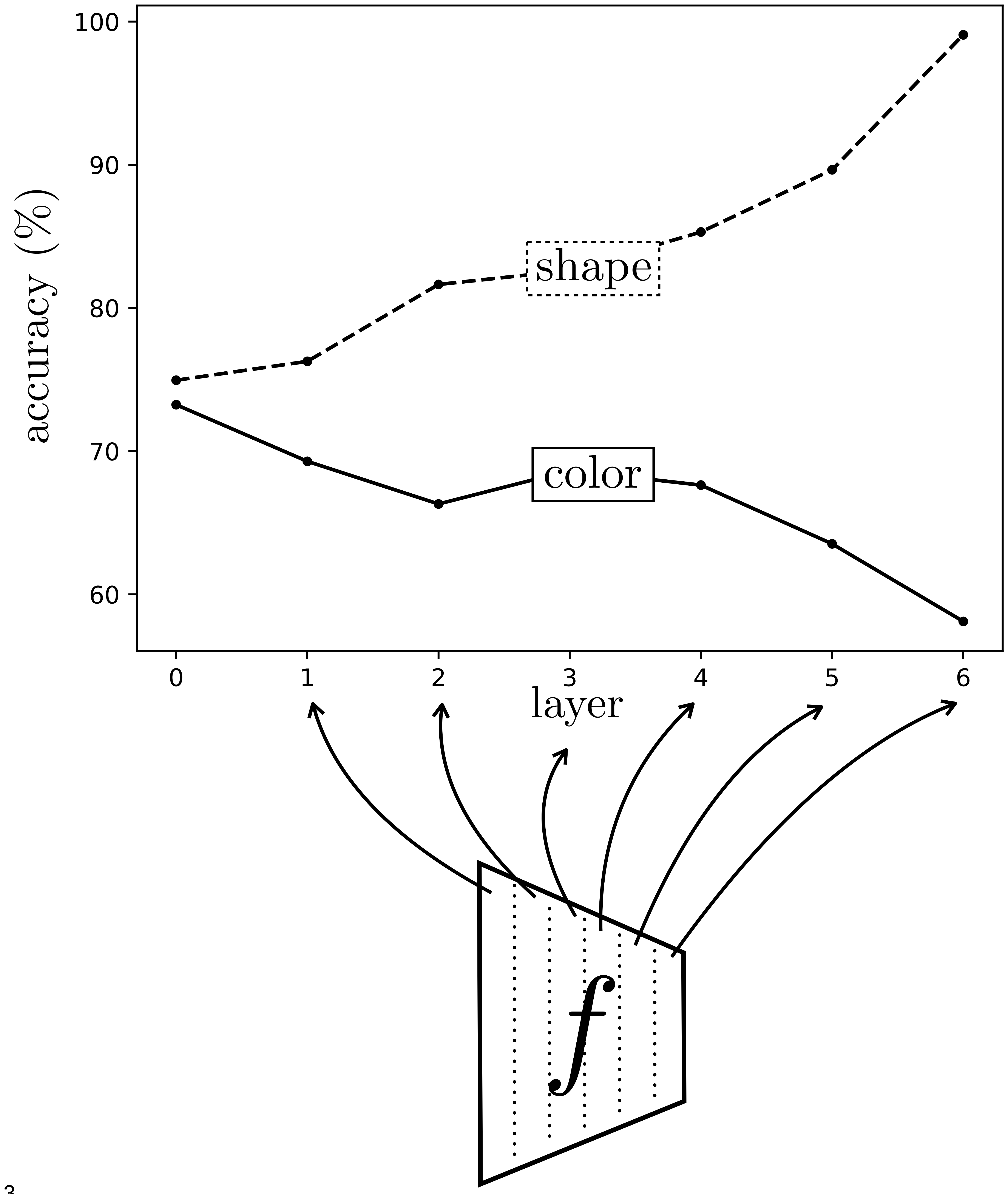
$$F = g \circ f : \mathbb{R}^N \rightarrow \mathbb{R}^M \rightarrow \mathbb{R}^N$$

$\rightarrow f$

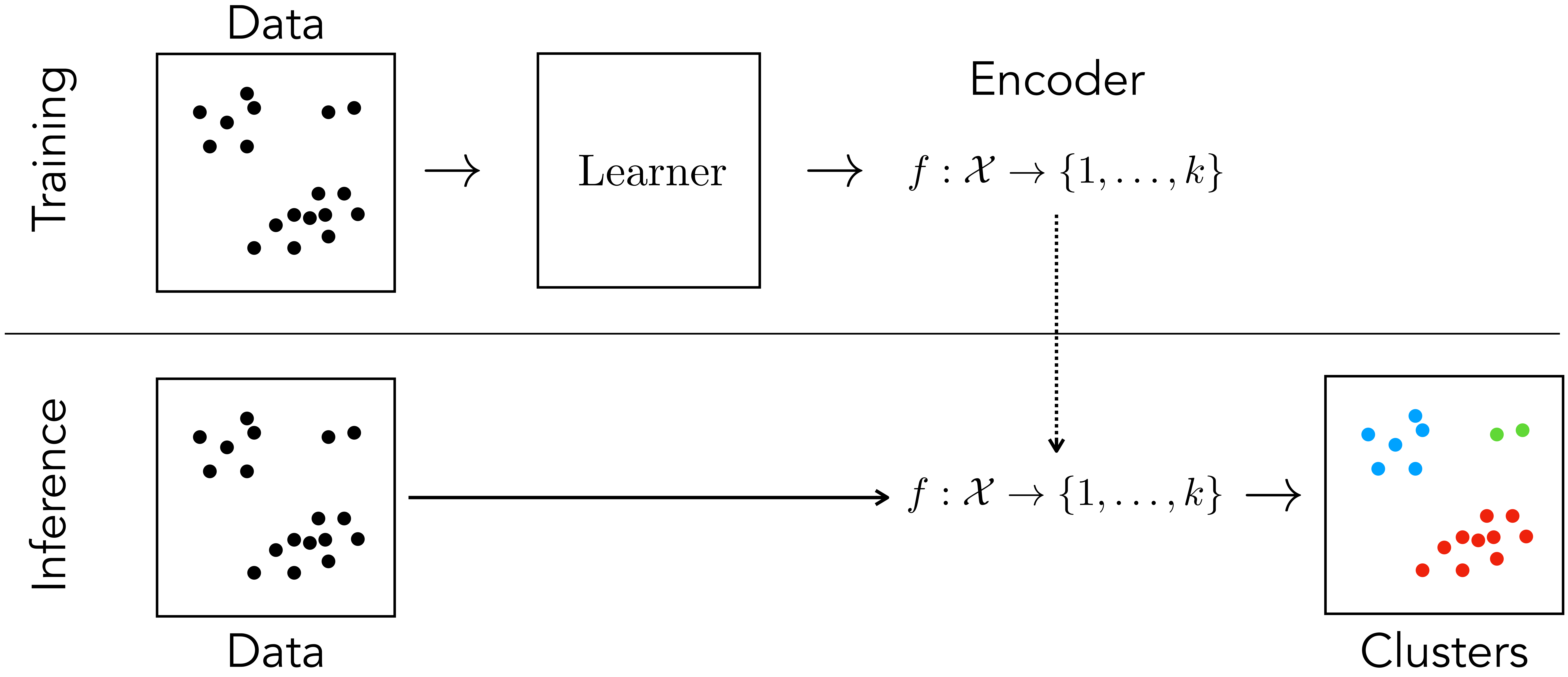


query nearest-neighbors in z-space



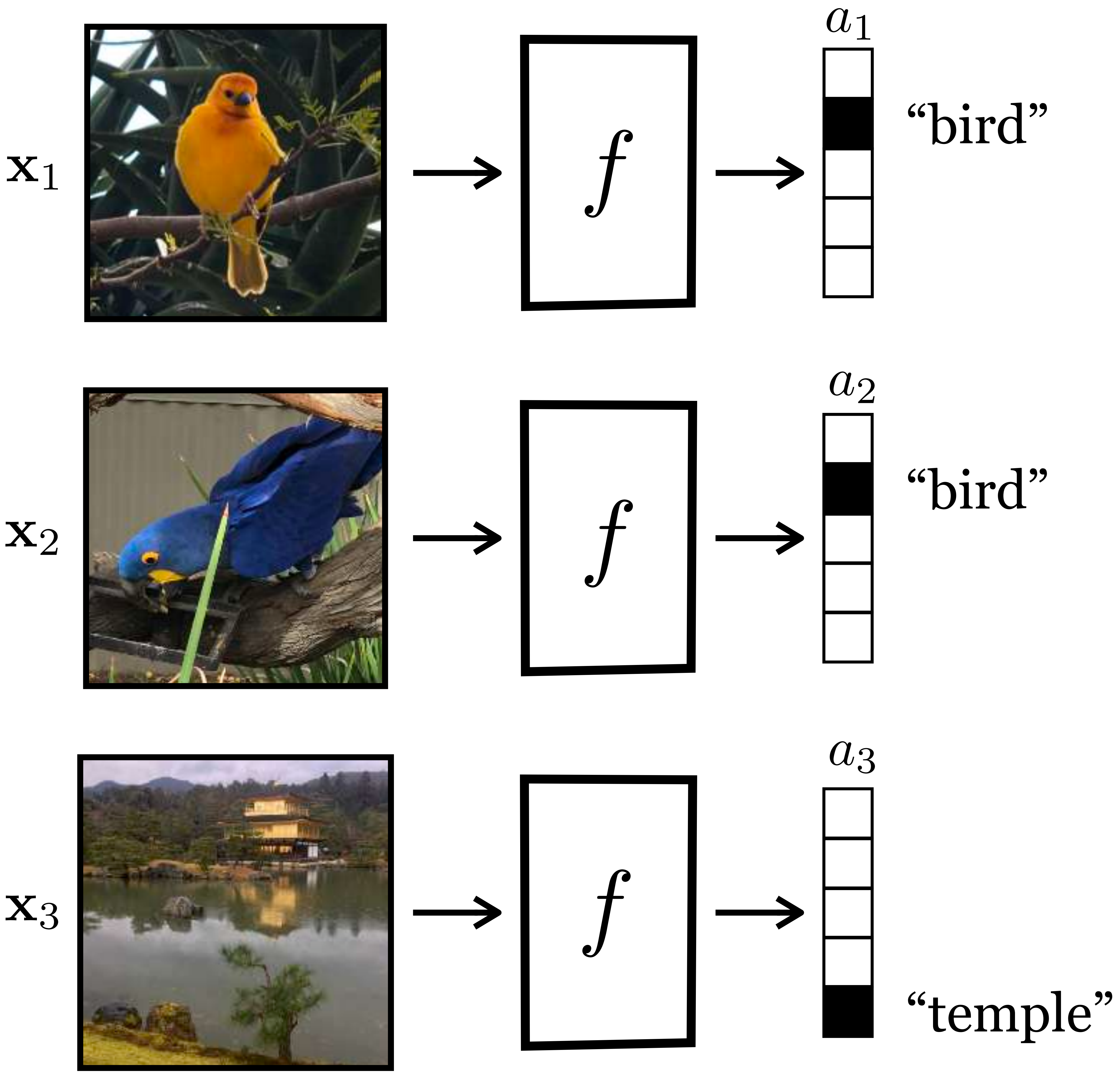


Clustering



Training
Data
Learner
Encoder
$f : \mathcal{X} \rightarrow \{1, \ldots, k\}$
Inference
Data
$f : \mathcal{X} \rightarrow \{1, \ldots, k\}$
Clusters

Clustering — a rep learning perspective

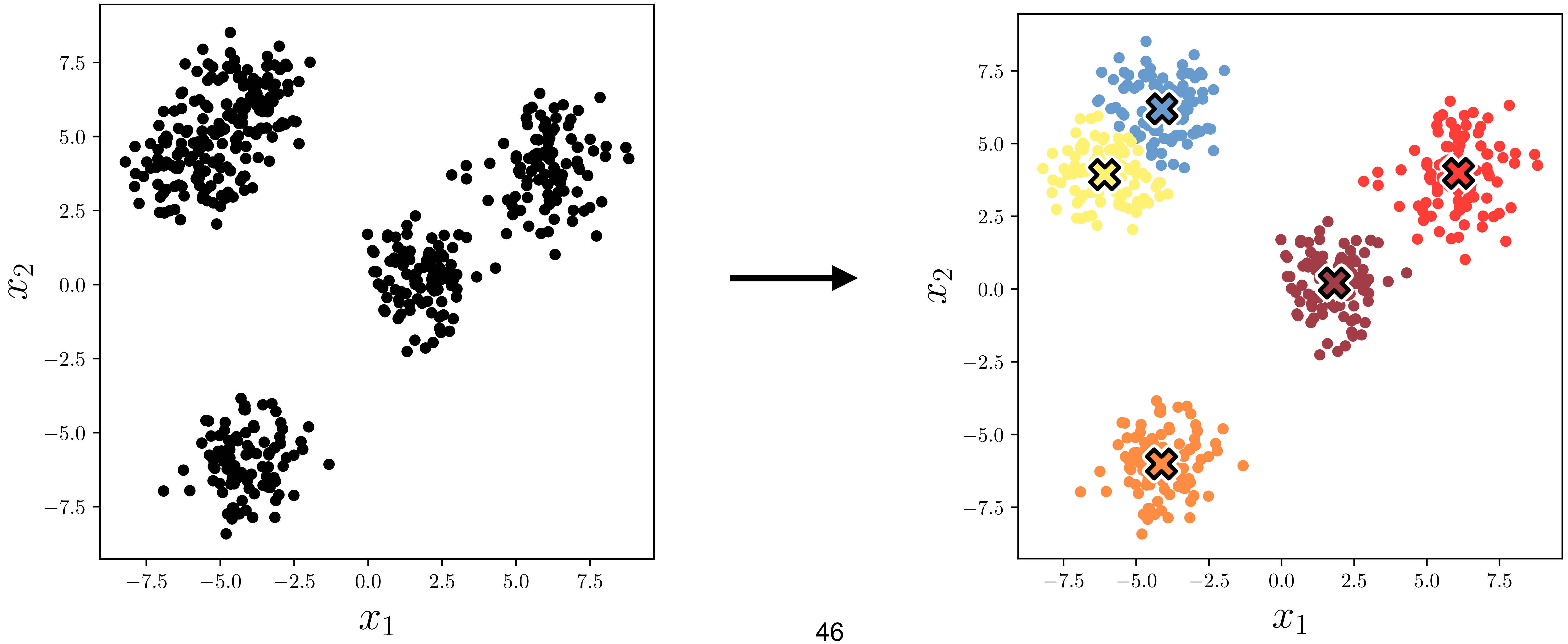


- What's the best representation that humans have come up with so far?
- Language!
- Words are the atoms of language
- Clustering is the problem of making up new words for things

k-means

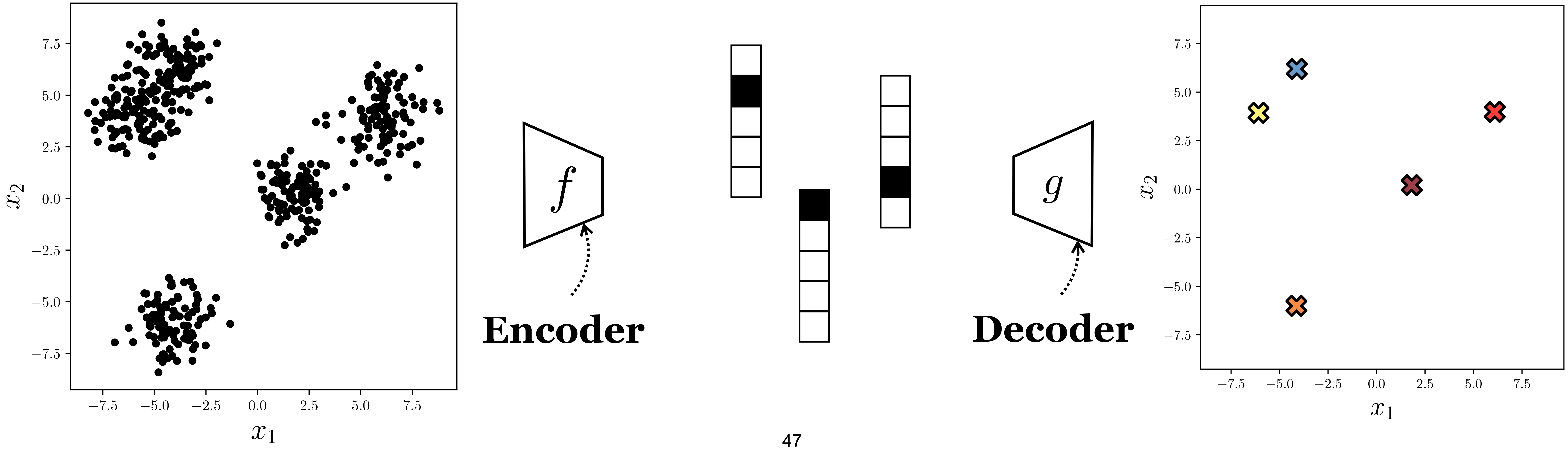
- Map datapoints to integers (i.e. cluster)

- In such a way that each datapoint is as close as possible to the mean of the cluster it is assigned to



k-means

- Map datapoints to integers (i.e. cluster)
- In such a way that each datapoint is as close as possible to its cluster's code mean



k-means

Data

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

k-means (L_2)

Objective

$$\mathcal{L}(F(\mathbf{x}), \mathbf{x}) = \|F(\mathbf{x}) - \mathbf{x}\|_2^2$$

Hypothesis space

$$F = g \circ f : \{\mathbf{x}\}_{i=1}^{N} \rightarrow \{1, \ldots, k\} \rightarrow \mathbb{R}^M$$

Optimizer

Block coordinate descent

$\longrightarrow f$

f and g are both lookup tables

VQ nets

Data

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

VQ Autoencoder (L_2)

Objective

$$\mathcal{L}(F(\mathbf{x}), \mathbf{x}) = \|F(\mathbf{x}) - \mathbf{x}\|_2^2 \; + \ldots$$

Hypothesis space

$$F = g \circ f : \{\mathbf{x}\}_{i=1}^{N} \rightarrow \{1, \ldots, k\} \rightarrow \mathbb{R}^M$$

Optimizer

Backprop w/ approximations

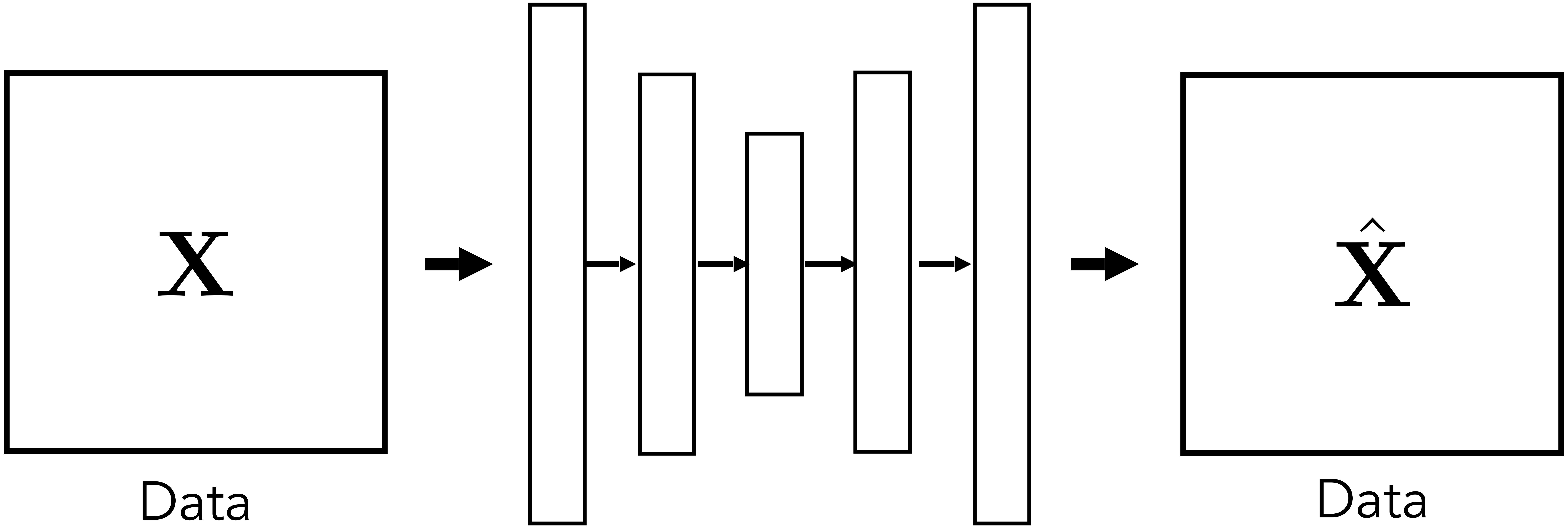
$\longrightarrow \; f$

What if f and g are both deep nets?

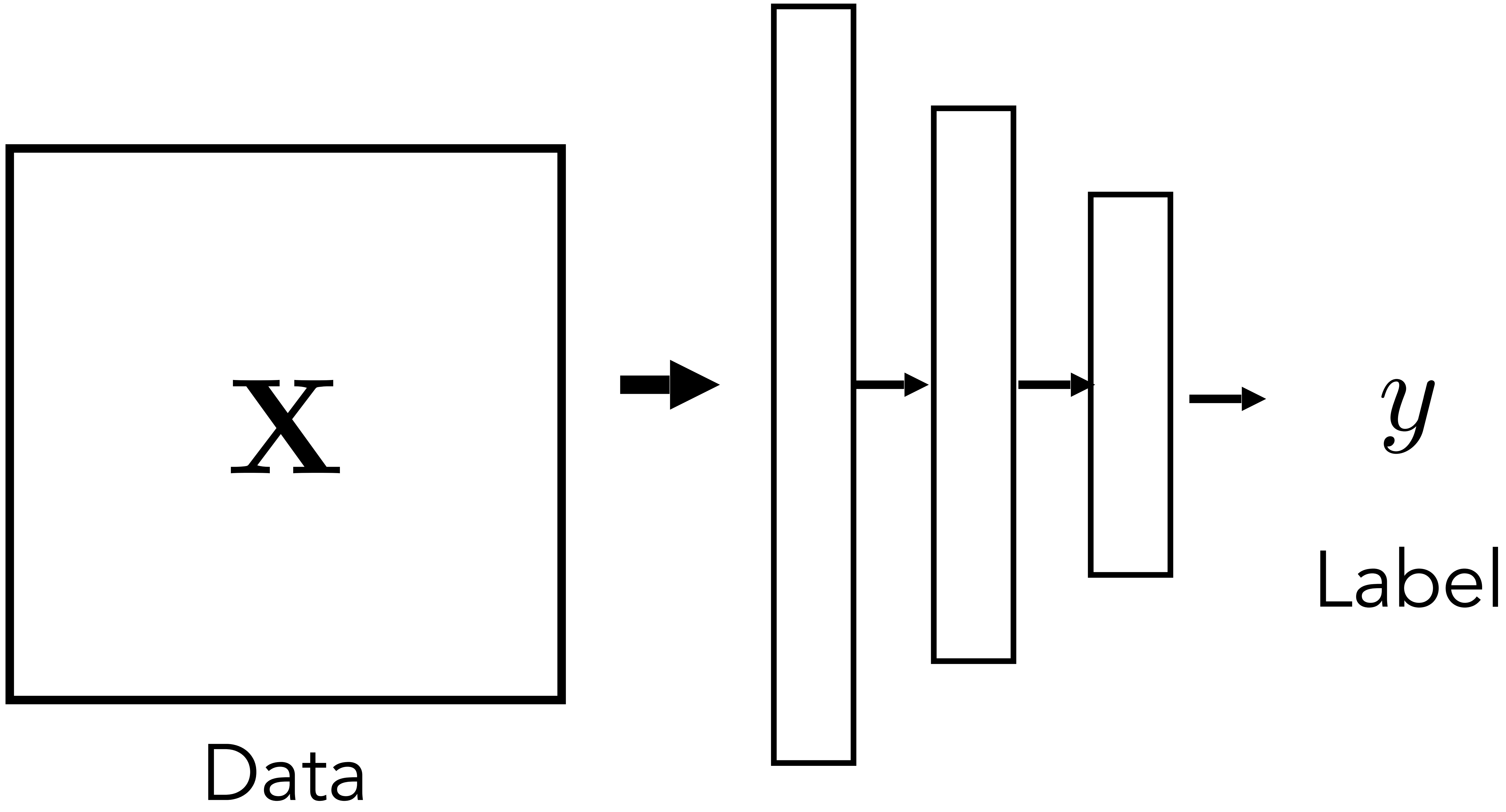
Then we call this a **"Vector Quantized" Autoencoder**
(e.g., VQVAE, VQGAN)

[see e.g., Oord, Vinyals, Kavukcuoglu, 2017]

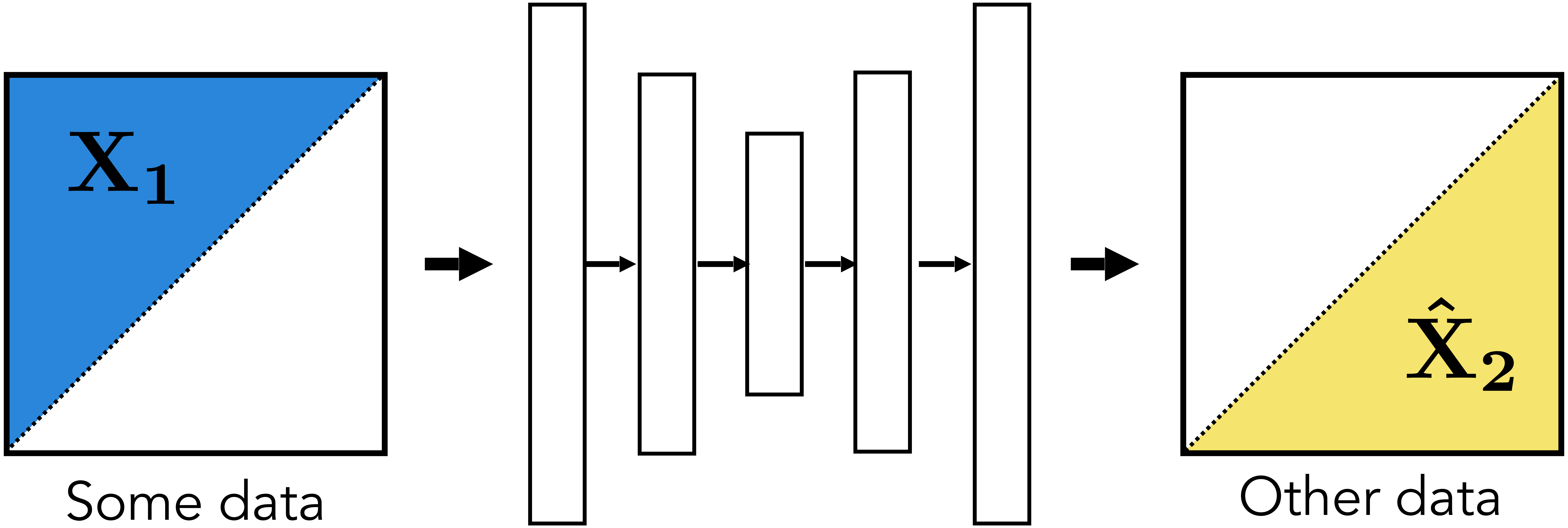
Data compression

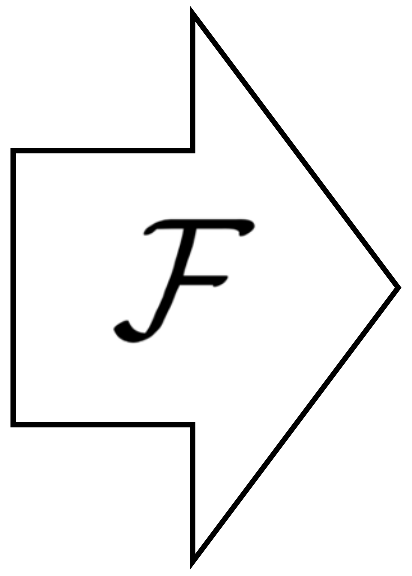


Label prediction



Data prediction
aka "self-supervised learning"



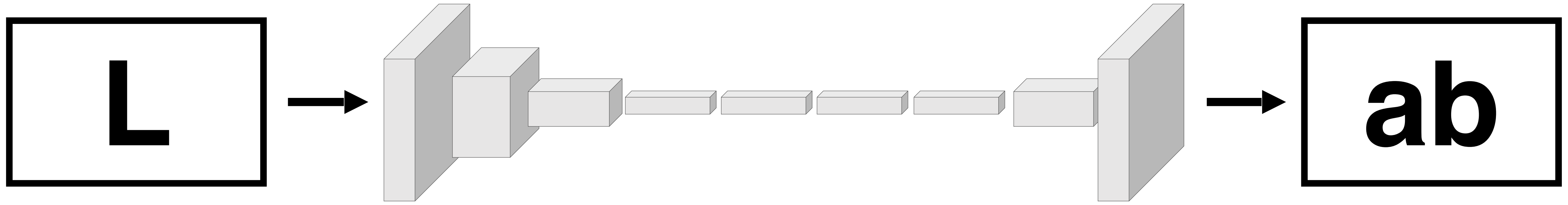


Grayscale image: L channel

$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$

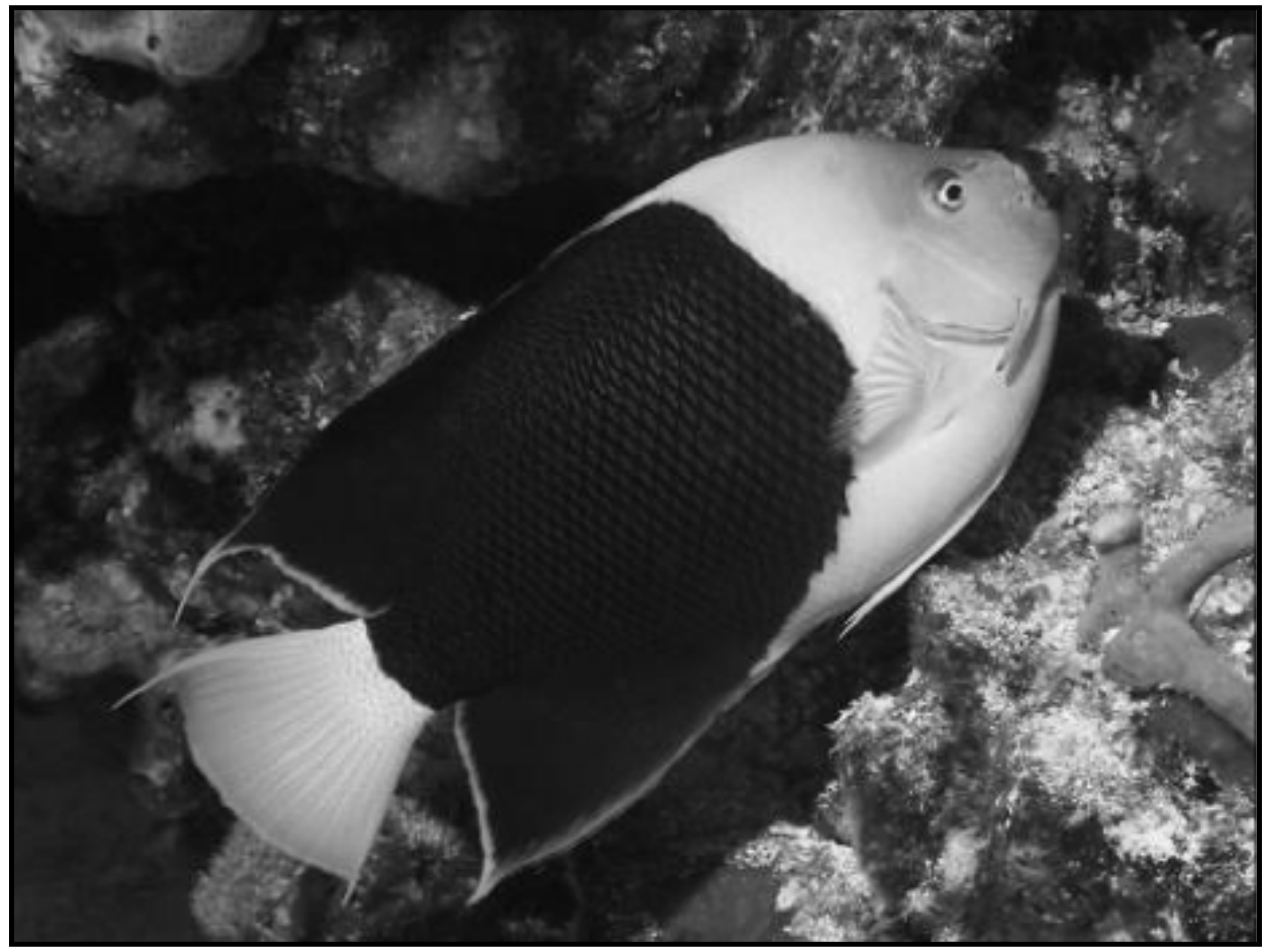
Color information: ab channels

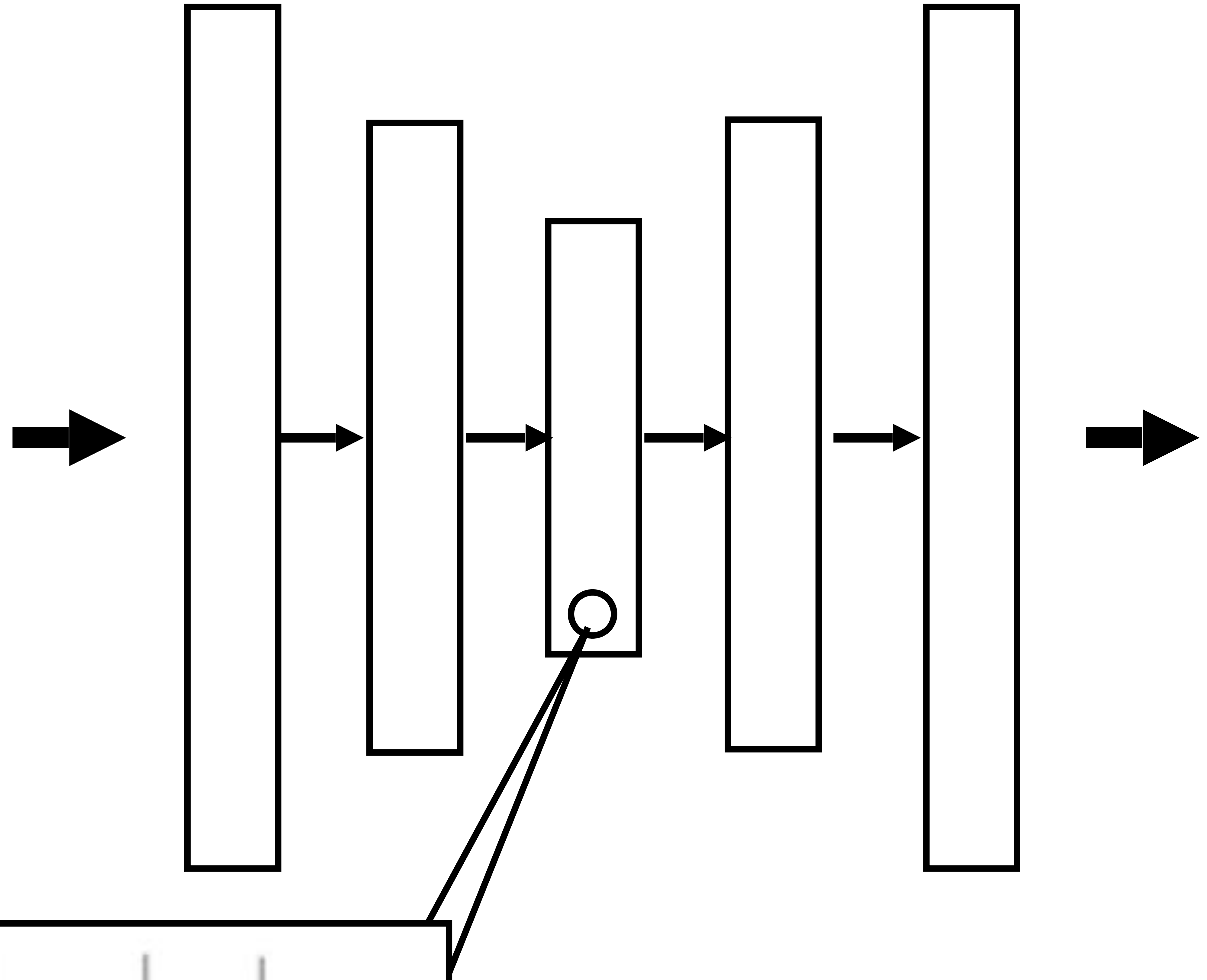
$$\widehat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 2}$$

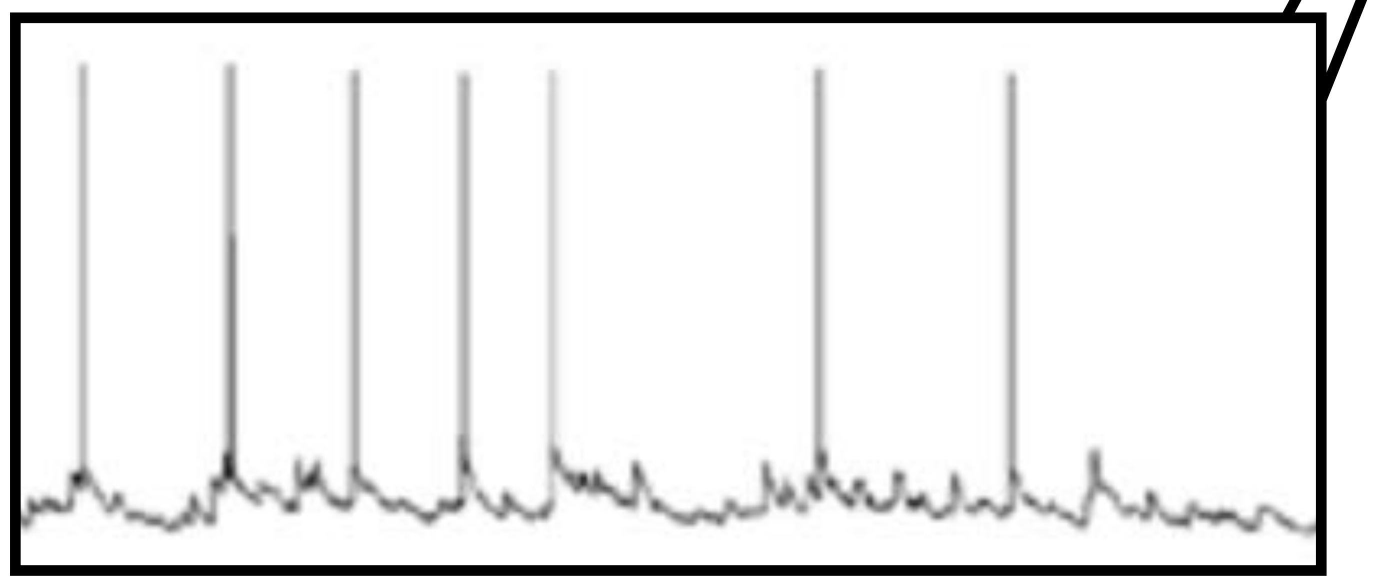


[Zhang, Isola, Efros, ECCV 2016]

Deep Net "Electrophysiology"







[Zeiler & Fergus, ECCV 2014]

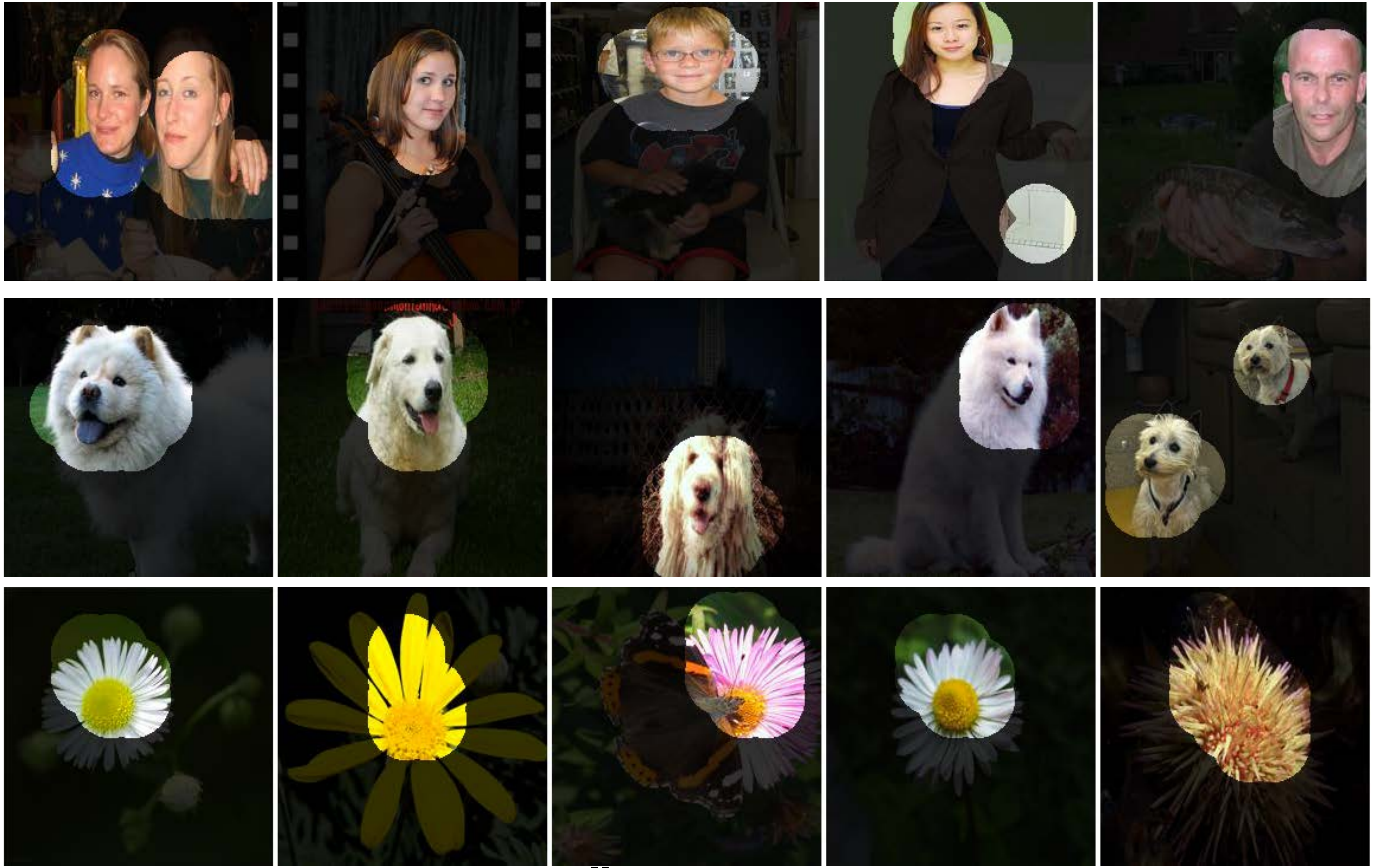
[Zhou, Khosla, Lapedriza, Oliva, Torralba., ICLR 2015]

Stimuli that drive selected neurons (conv5 layer)

faces

dog faces

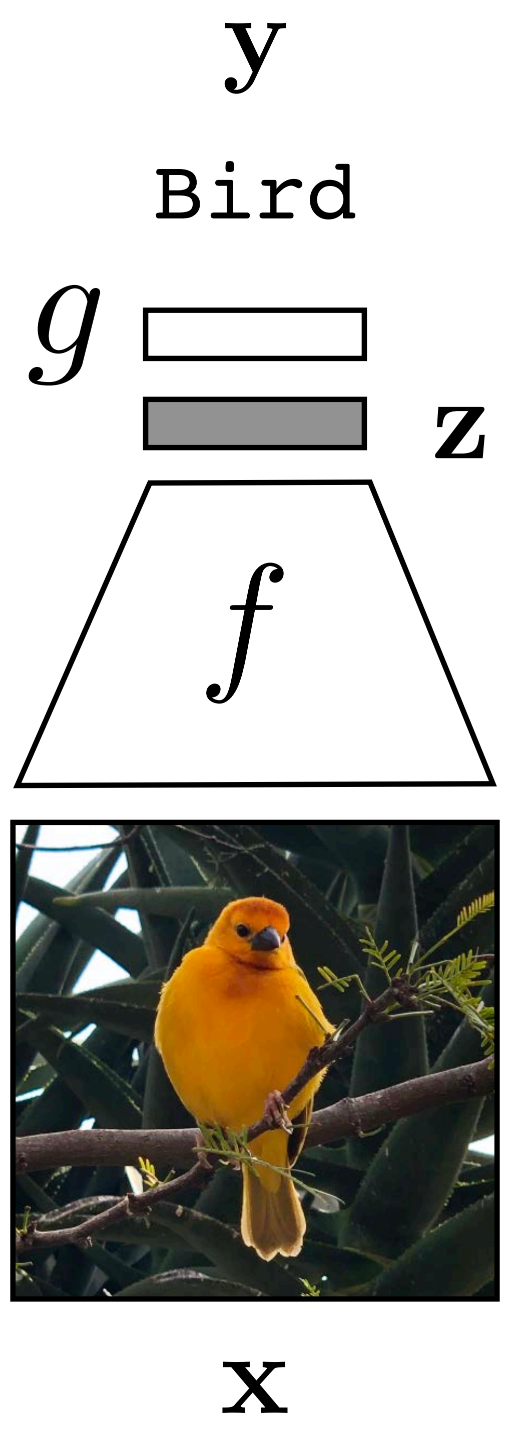
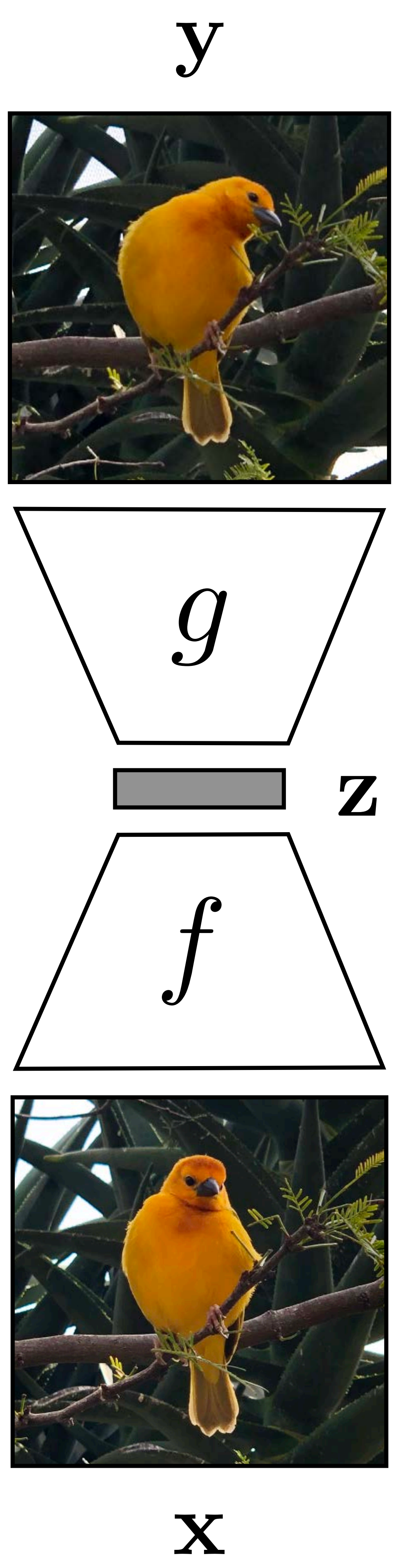
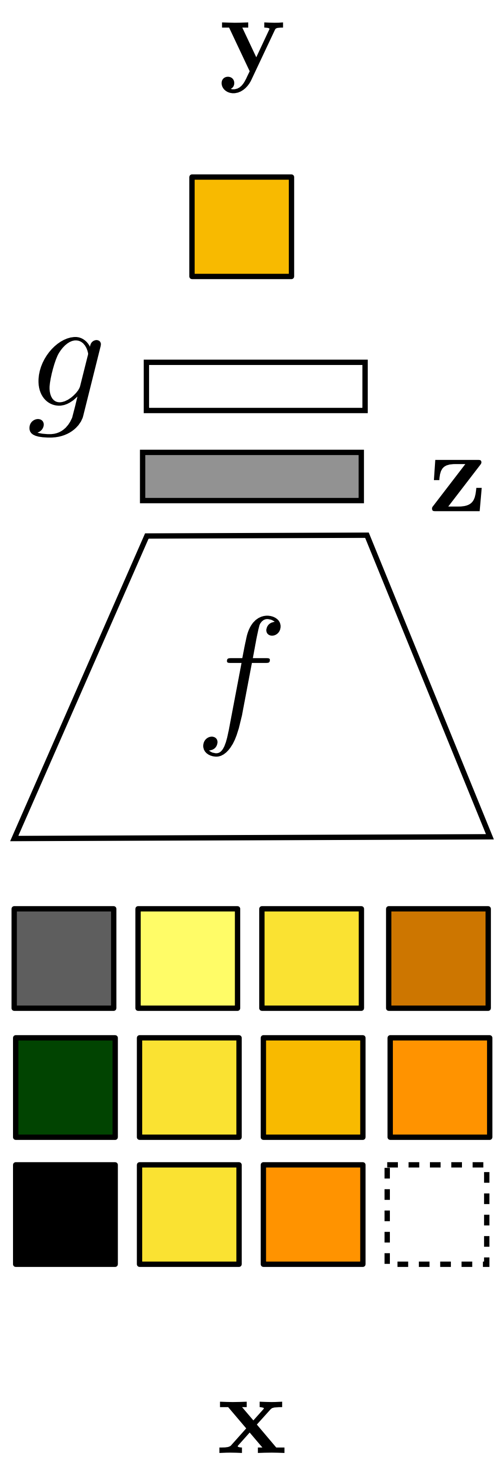
flowers



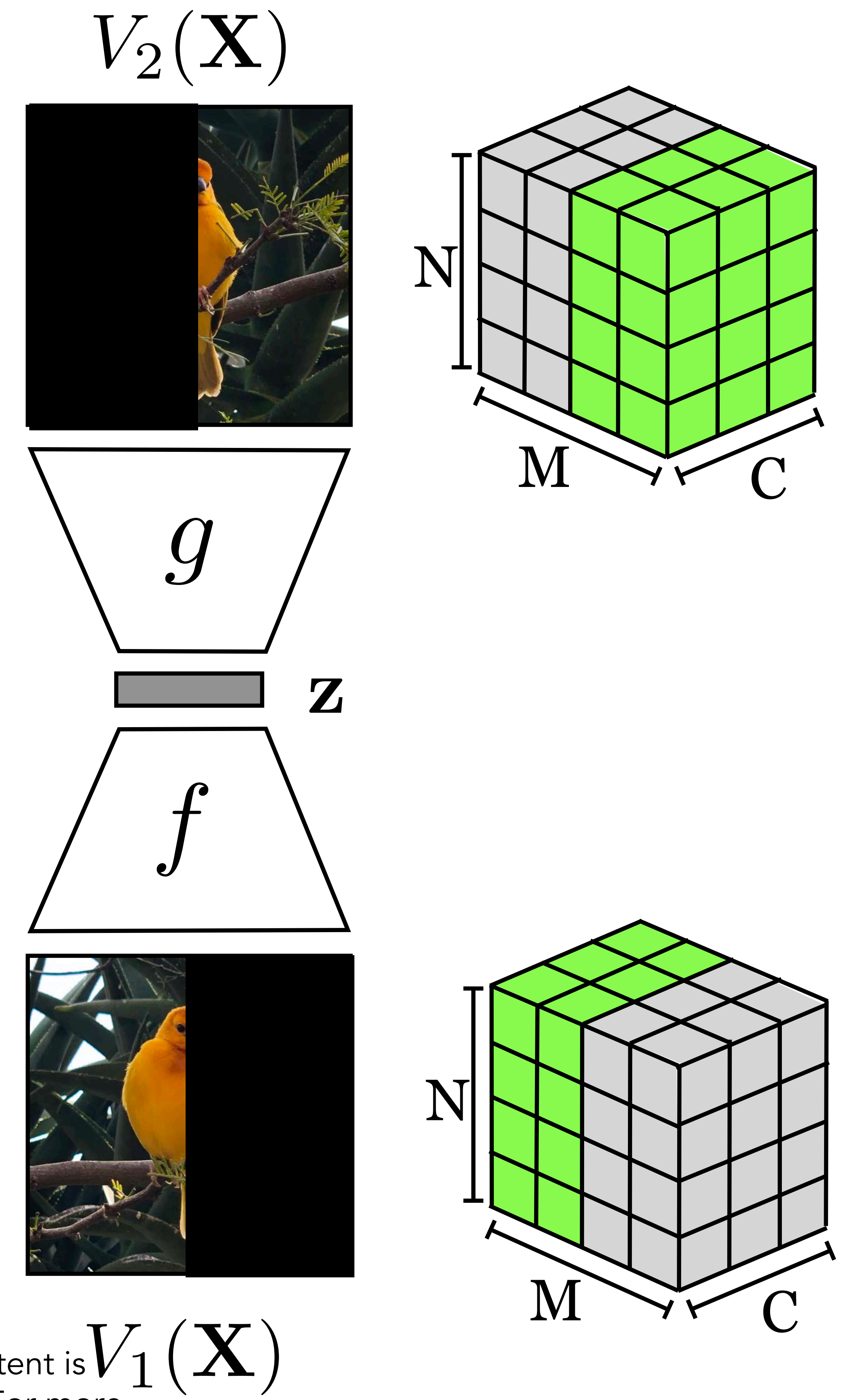
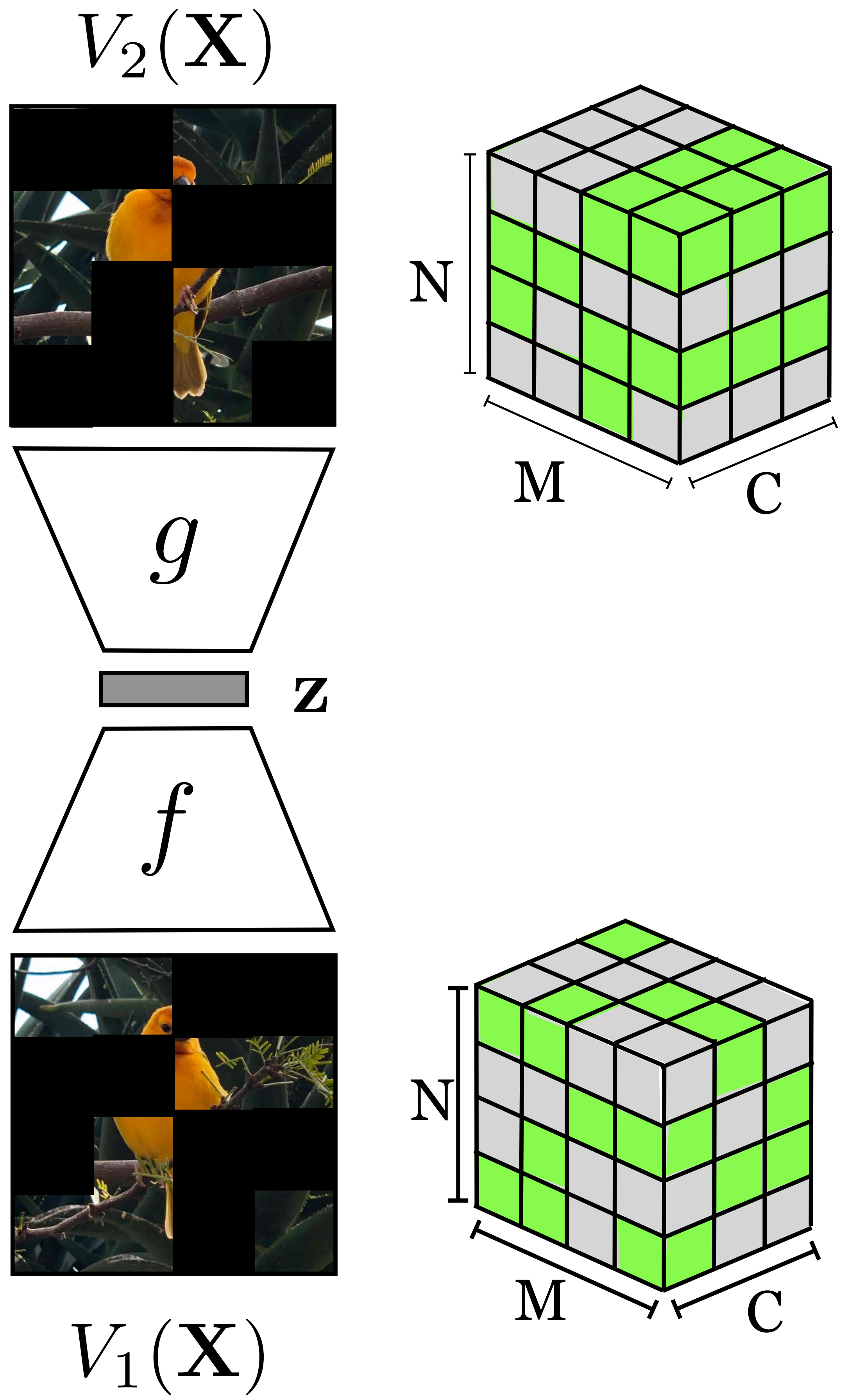
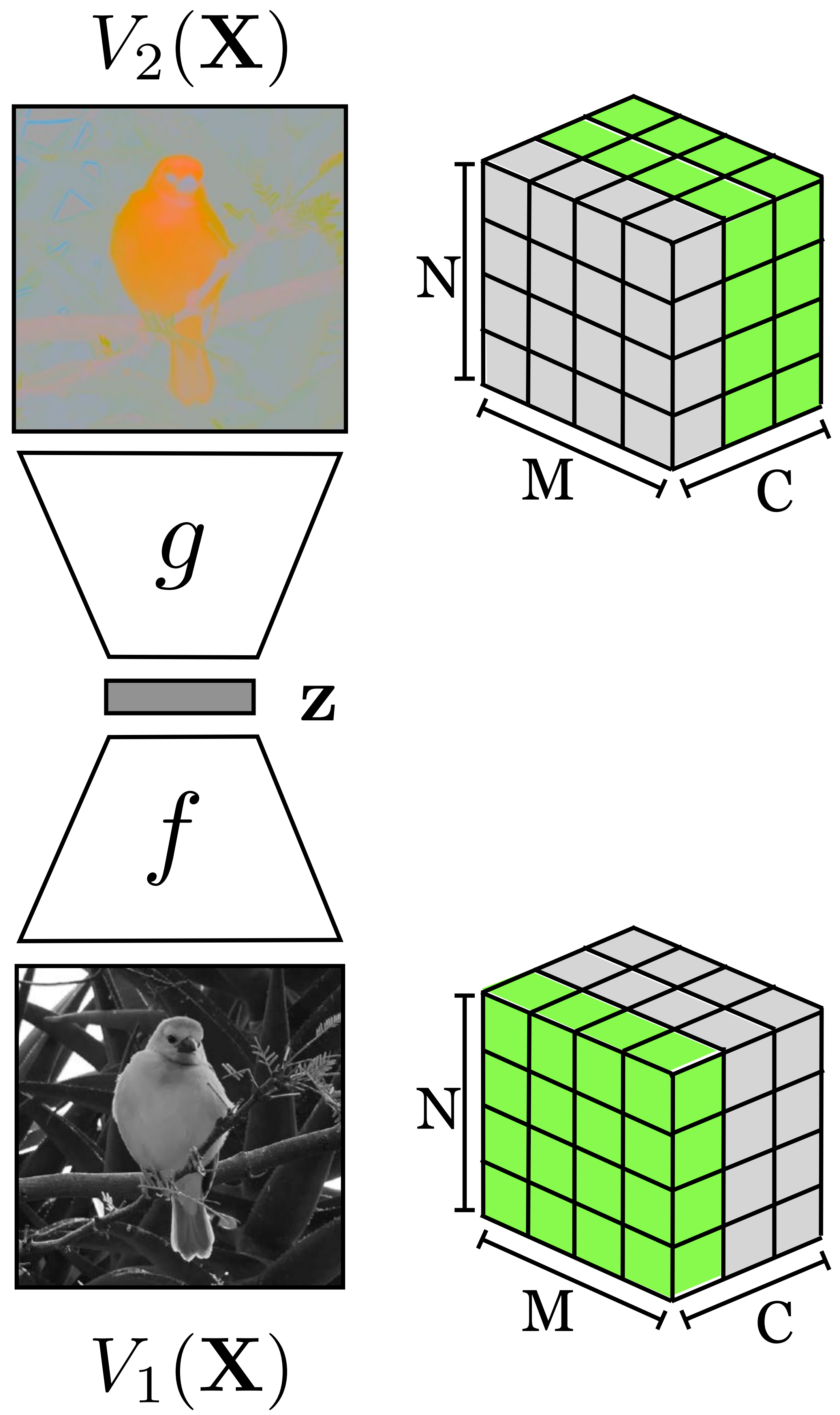
Self-supervised learning

Common trick:

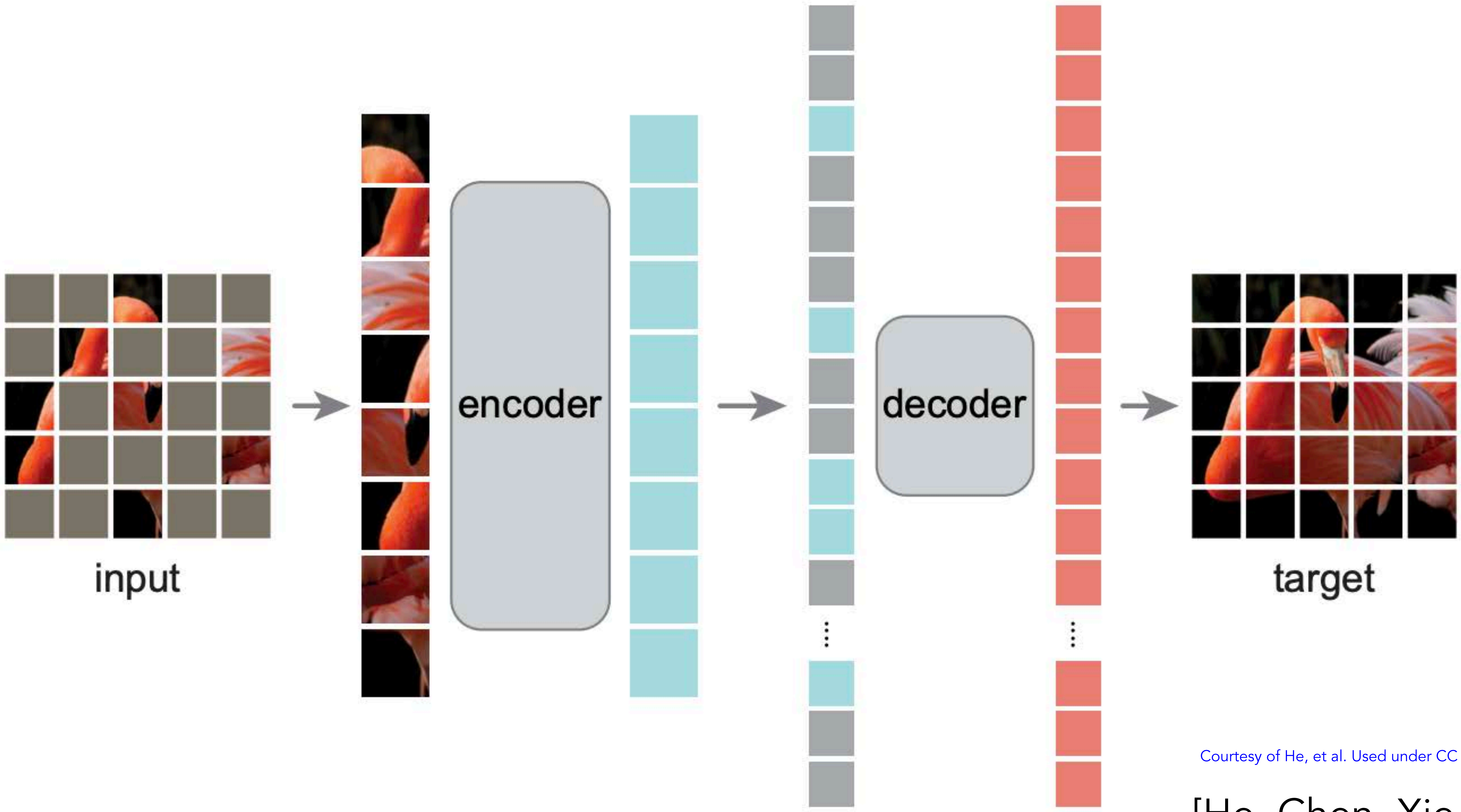
- Convert "unsupervised" problem into "supervised" empirical risk minimization
- Do so by cooking up "labels" (prediction targets) from the raw data itself — called **pretext task**

Pretext task:	Class prediction	Future frame prediction	Next pixel prediction
Model schematic:			

Imputation: one pretext task to rule them all?

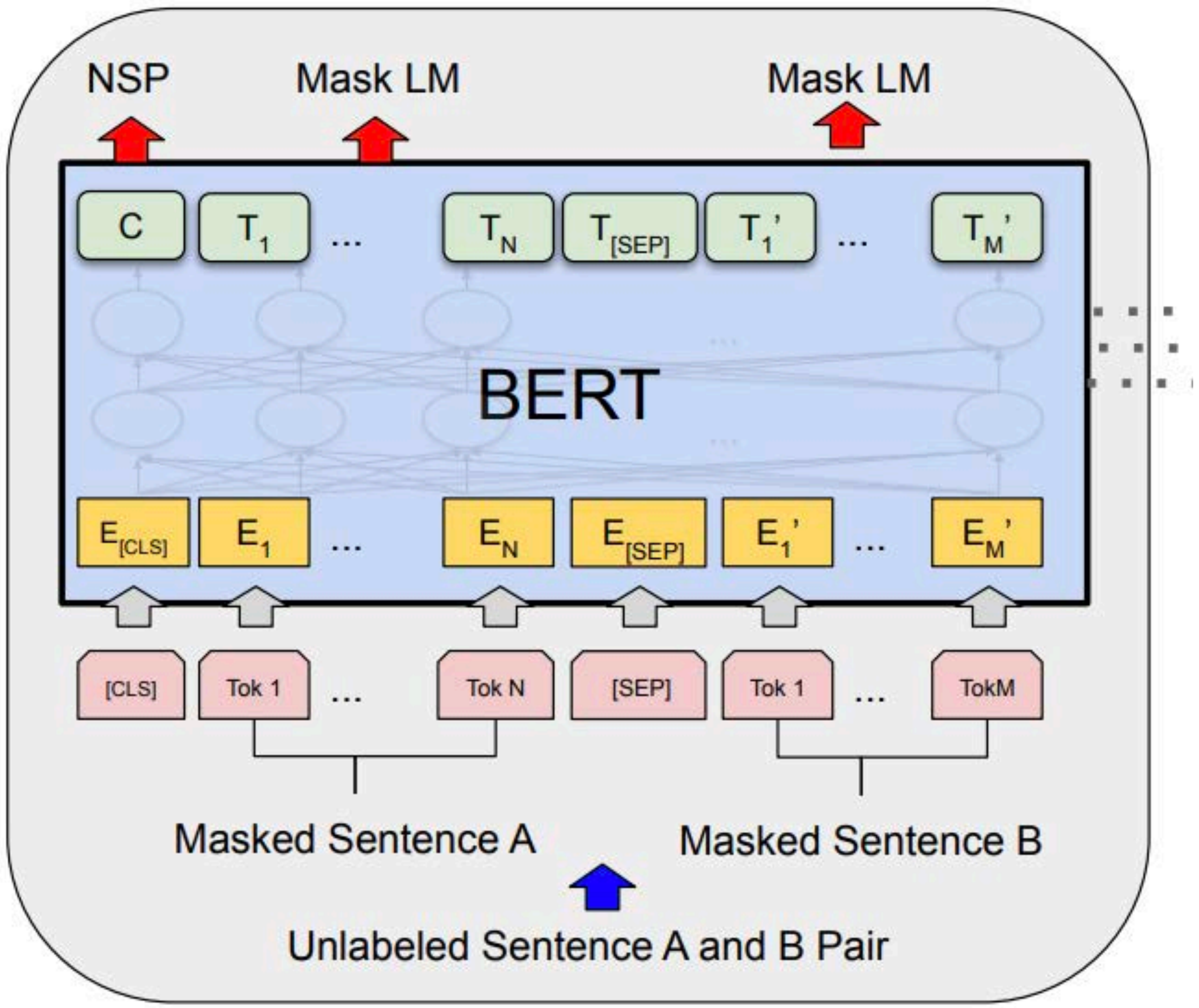
Pretext task:	Spatial imputation	Spatial imputation	Channel imputation
Model schematic: Observed Masked			

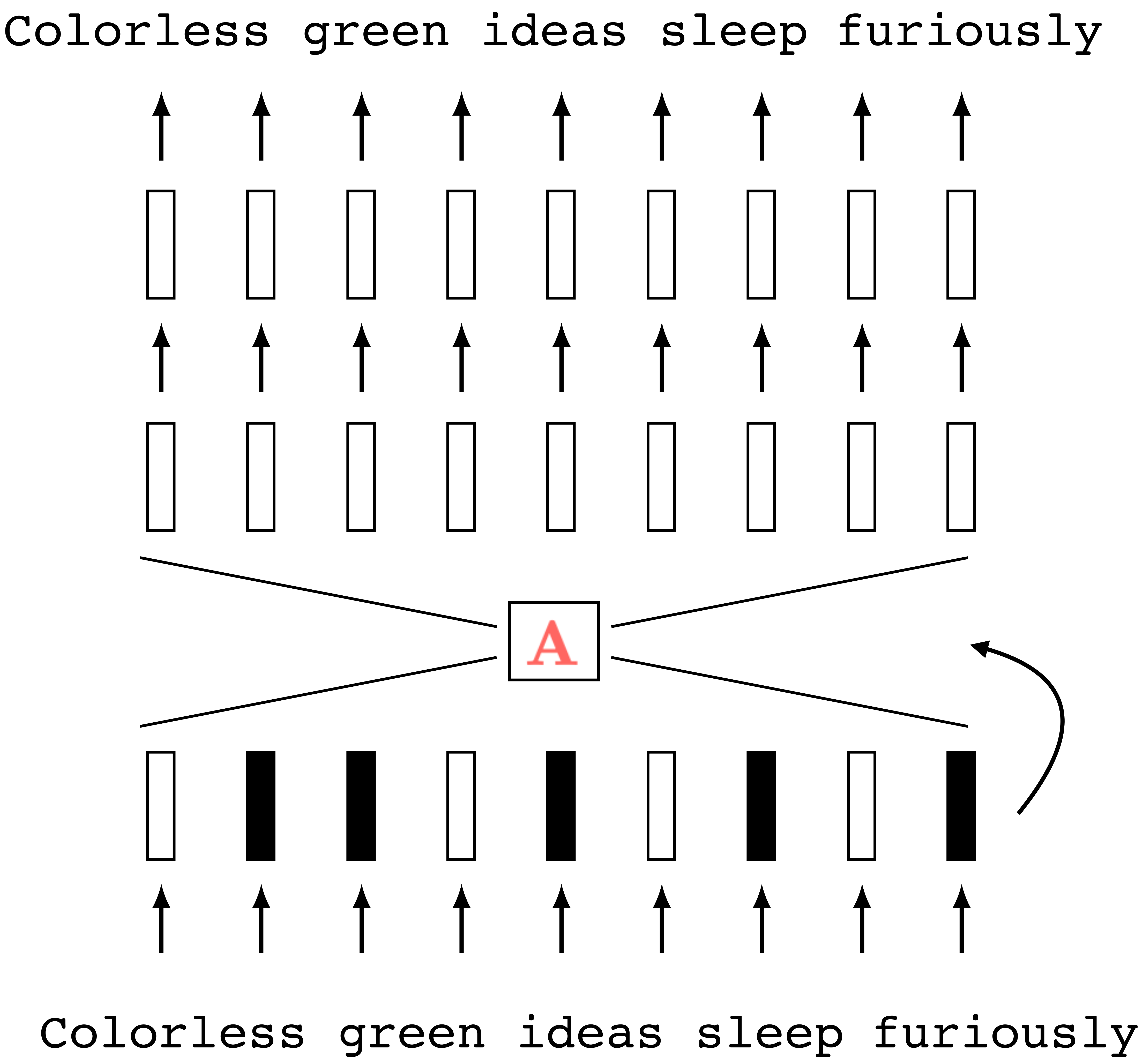
Masked Autoencoder (MAE)



[He, Chen, Xie, et al. 2021]

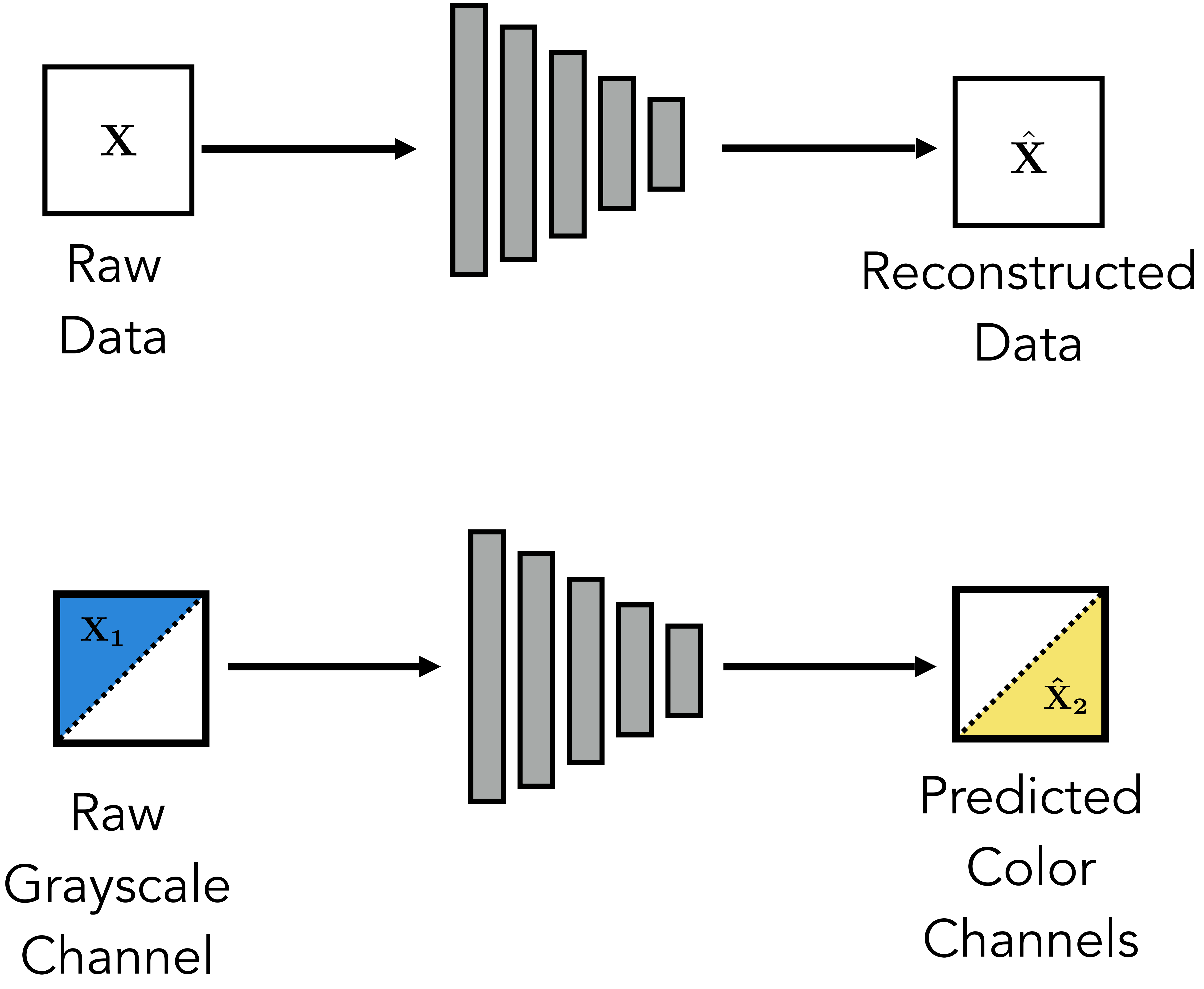
Bidirectional Transformers (BERT)

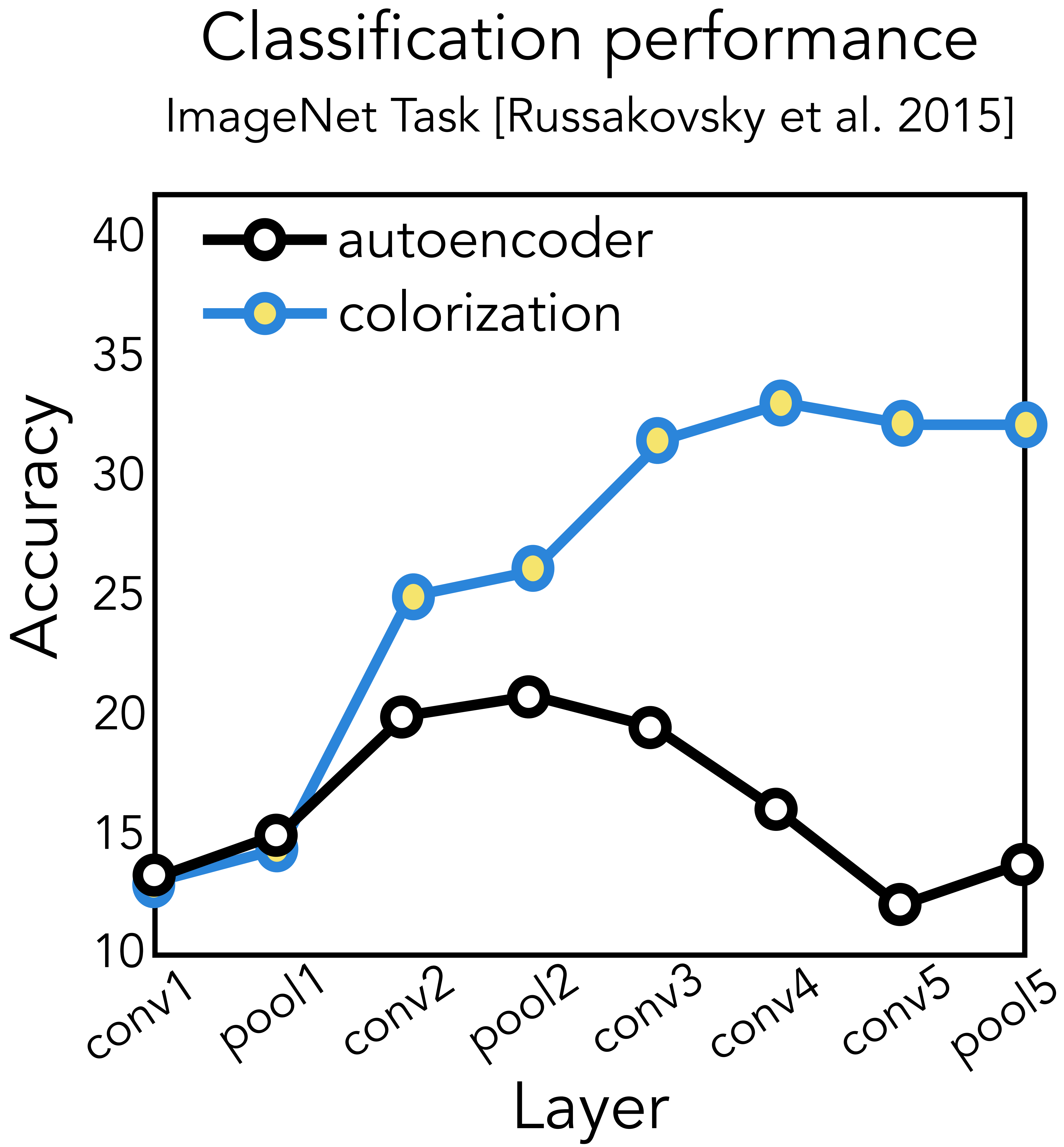




[He, Chen, Xie, et al. 2021]

Masked prediction often works better than autoencoding





[Zhang, Isola, Efros, ECCV 2016]

Masked prediction often works better than autoencoding

Ongoing science!

Why?

- Hypothesis 1: It's hard to control compression via a dimensional bottleneck. Requires fiddling with the architecture. Low-dimensional embeddings have bad properties in terms of optimization, etc.

- Hypothesis 2: Autoencoders have shortcuts where they can copy part of the input and get a decent loss. They fall into these traps (local minima) even if global minimizer is in fact good.

- Hypothesis 3: Masked prediction is closer to the downstream problems we care about, which are mainly about prediction.

- Still an open question!

How Much Information is the Machine Given during Learning?

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

[Slide Credit: Yann LeCun]

Summary

1. Deep nets learn *representations*, just like our brains do
2. This is useful because representations transfer — they act as prior knowledge that enables quick learning on new tasks
3. Representations can also be learned without labels, which is great since labels are expensive and limiting
4. Without labels there are many ways to learn representations. We saw:
    1. representations as compressed codes
    2. representations as predictions of missing data