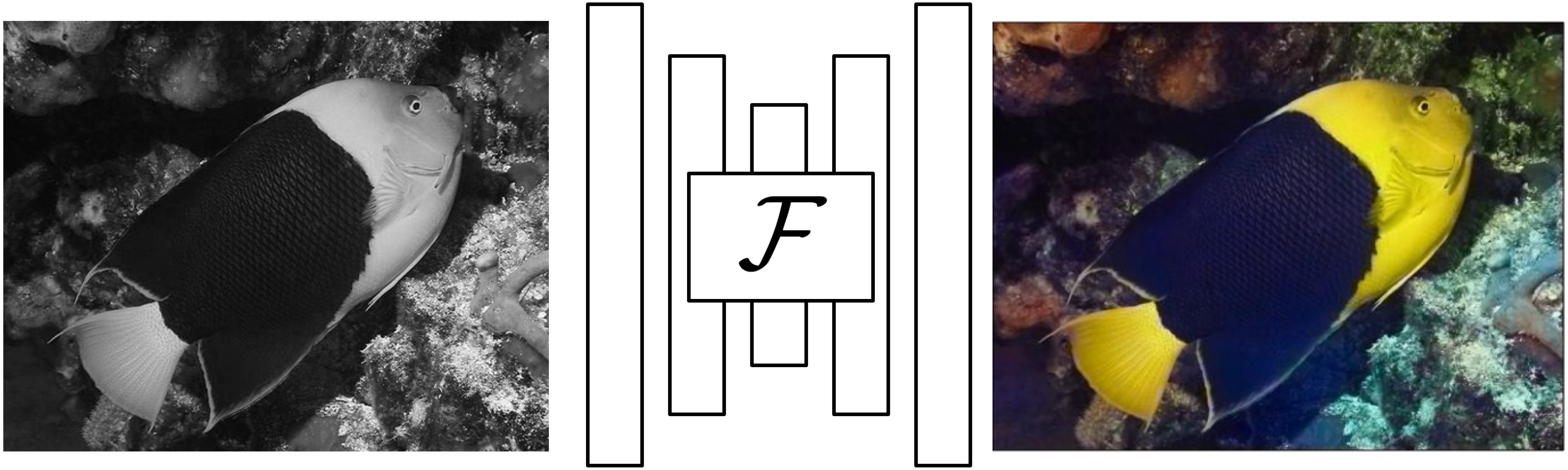


Convolutional Network for Image Synthesis

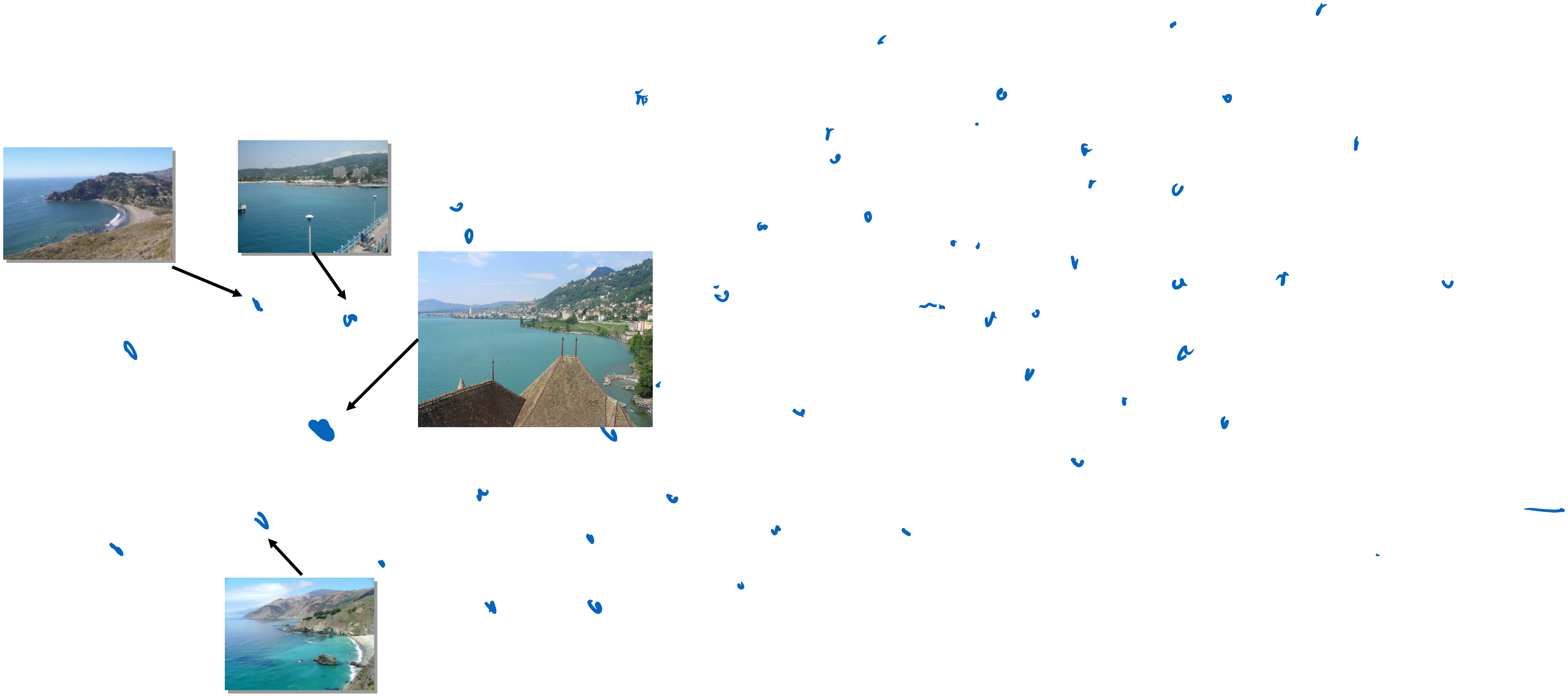
Jun-Yan Zhu

16-726 Learning-based Image Synthesis, Spring 2022

many slides from Alyosha Efros, Phillip Isola, Richard Zhang, James Hays, and Andrea Vedaldi, Jitendra Malik.

© Zhang et al., ECCV 2016

Review (data-driven graphics)

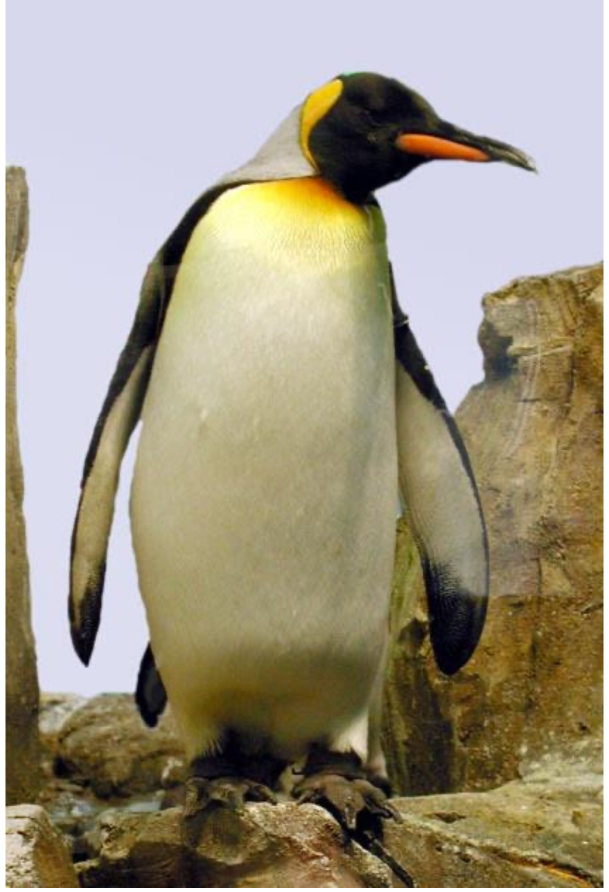


Review (data-driven graphics)

Nearest neighbor methods:

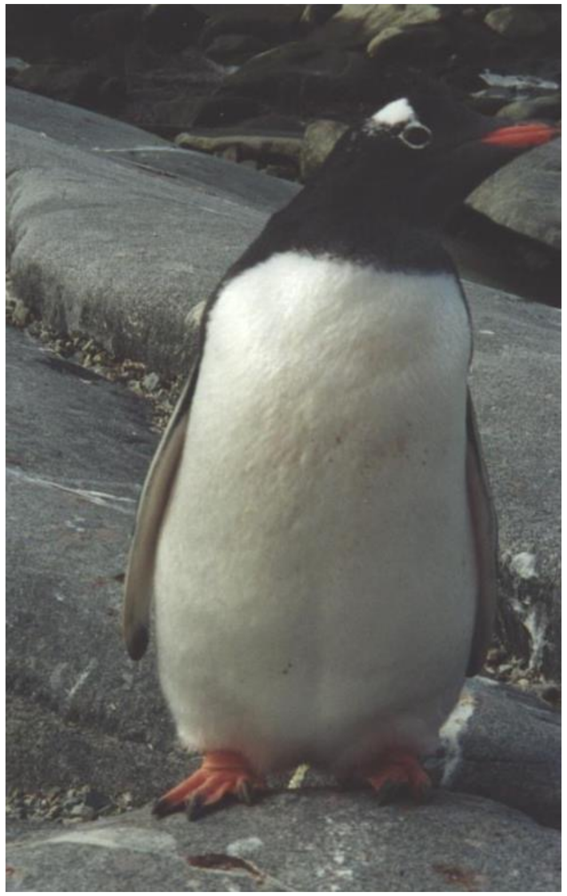
1. Stored examples
2. Calculate distance between two examples
3. Voting (label transfer): image blending/averaging

Visual similarity via labels



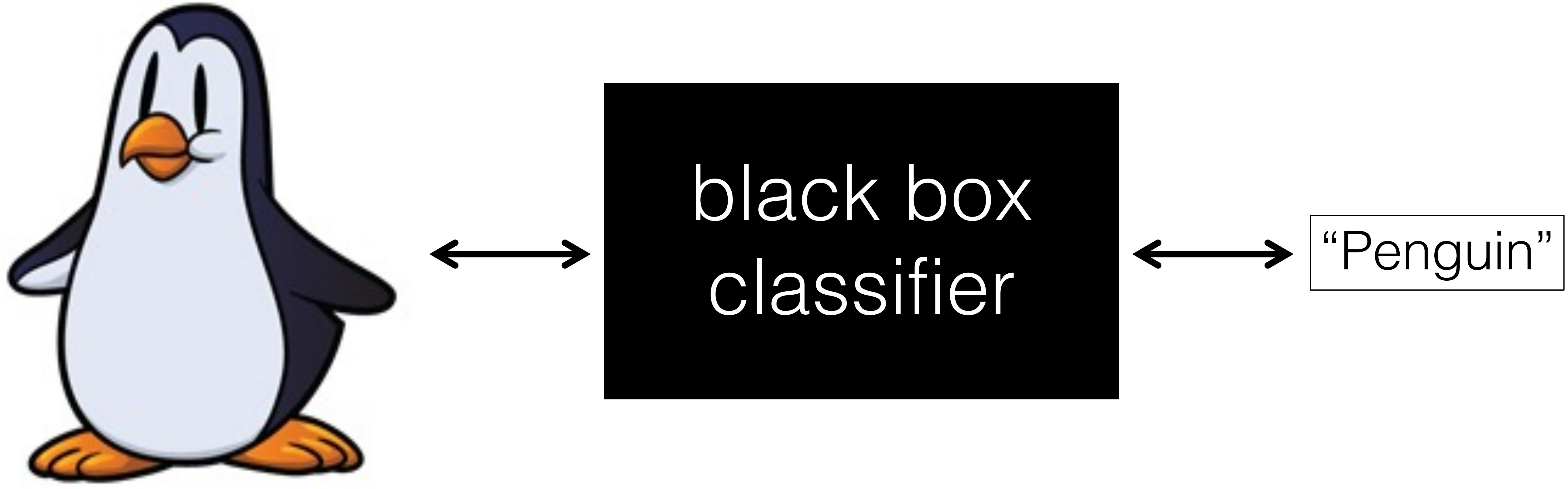
“Penguin”

$\overset{?}{==}$



“Penguin”

Machine Learning as data association



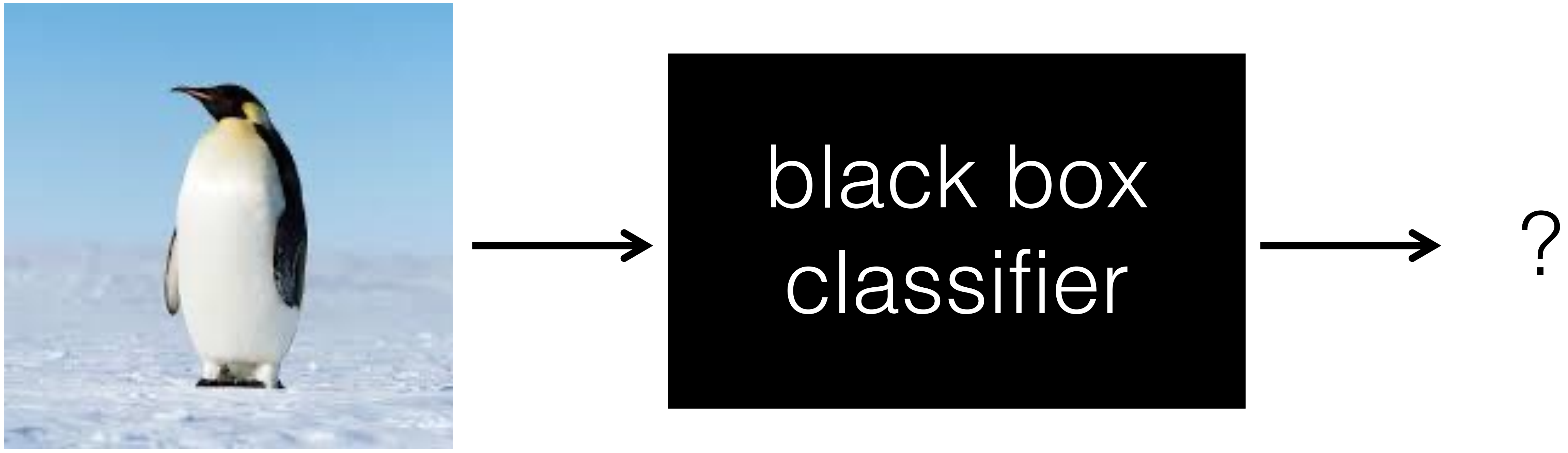
black box classifier

“Penguin”

image X

label Y

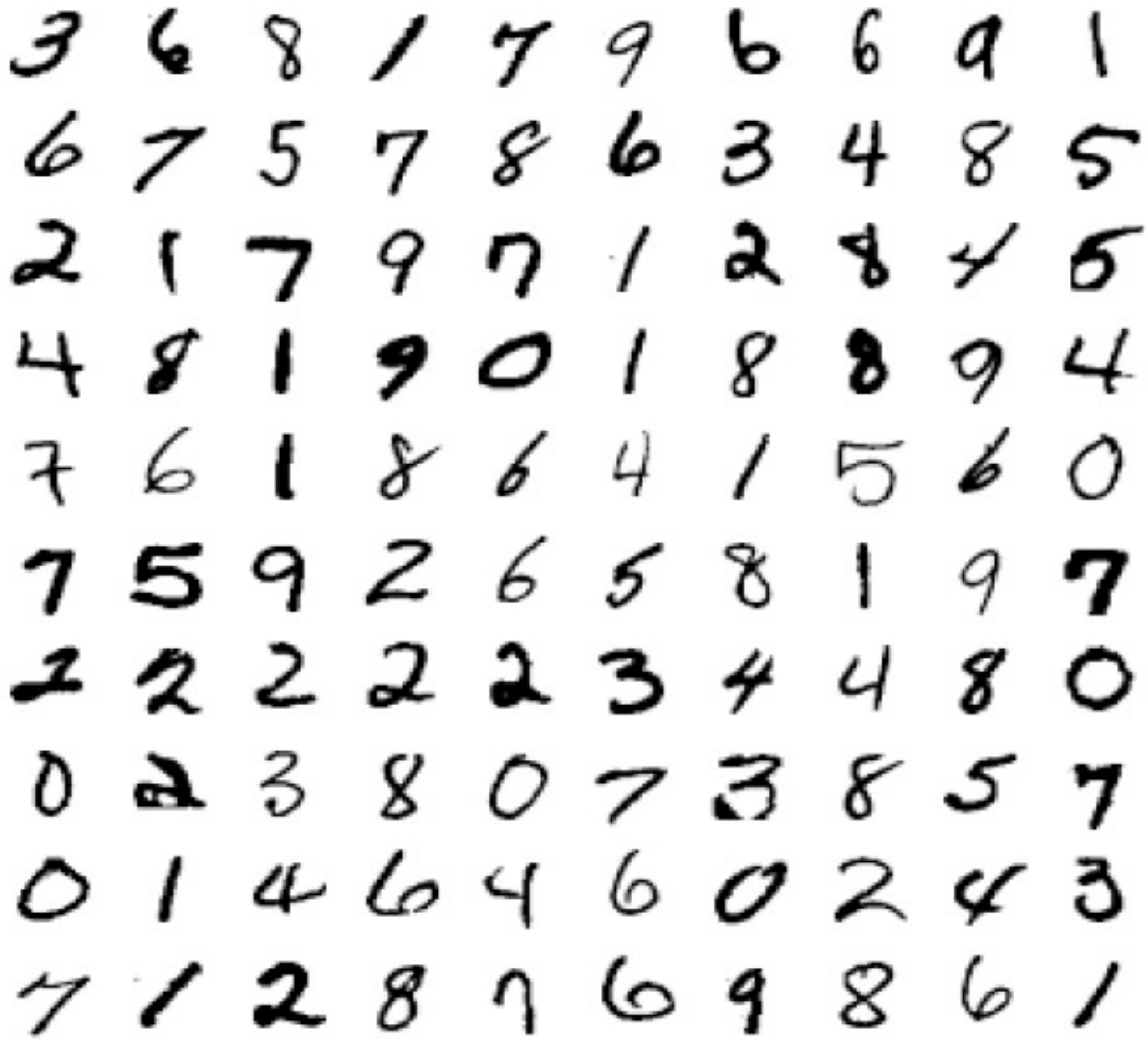
At test time...



black box classifier

?

image X

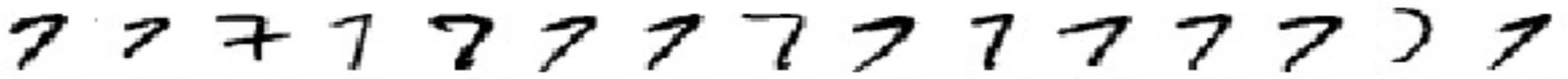
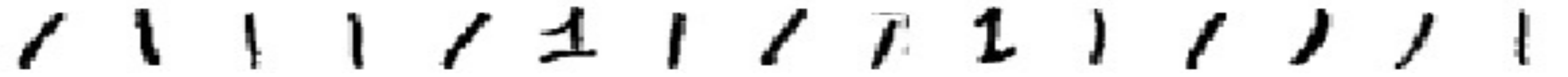


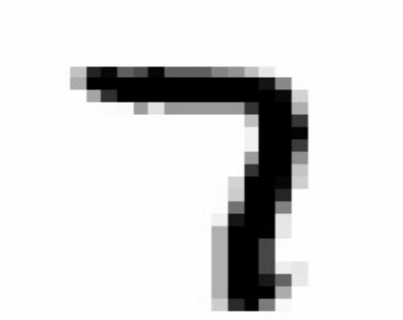
Examples from MNIST dataset [LeCun. 1998]

Warm-up Example:
Binary Digit Classification

vs.

Learning Approach to Digit Recognition

- **Collect Training Images**
  - Positive: 
  - Negative: 
- **Training Time**
  - Compute feature vectors for positive and negative example images
  - Train a classifier
- **Test Time**
  - Compute feature vector on new test image:
  - Evaluate classifier

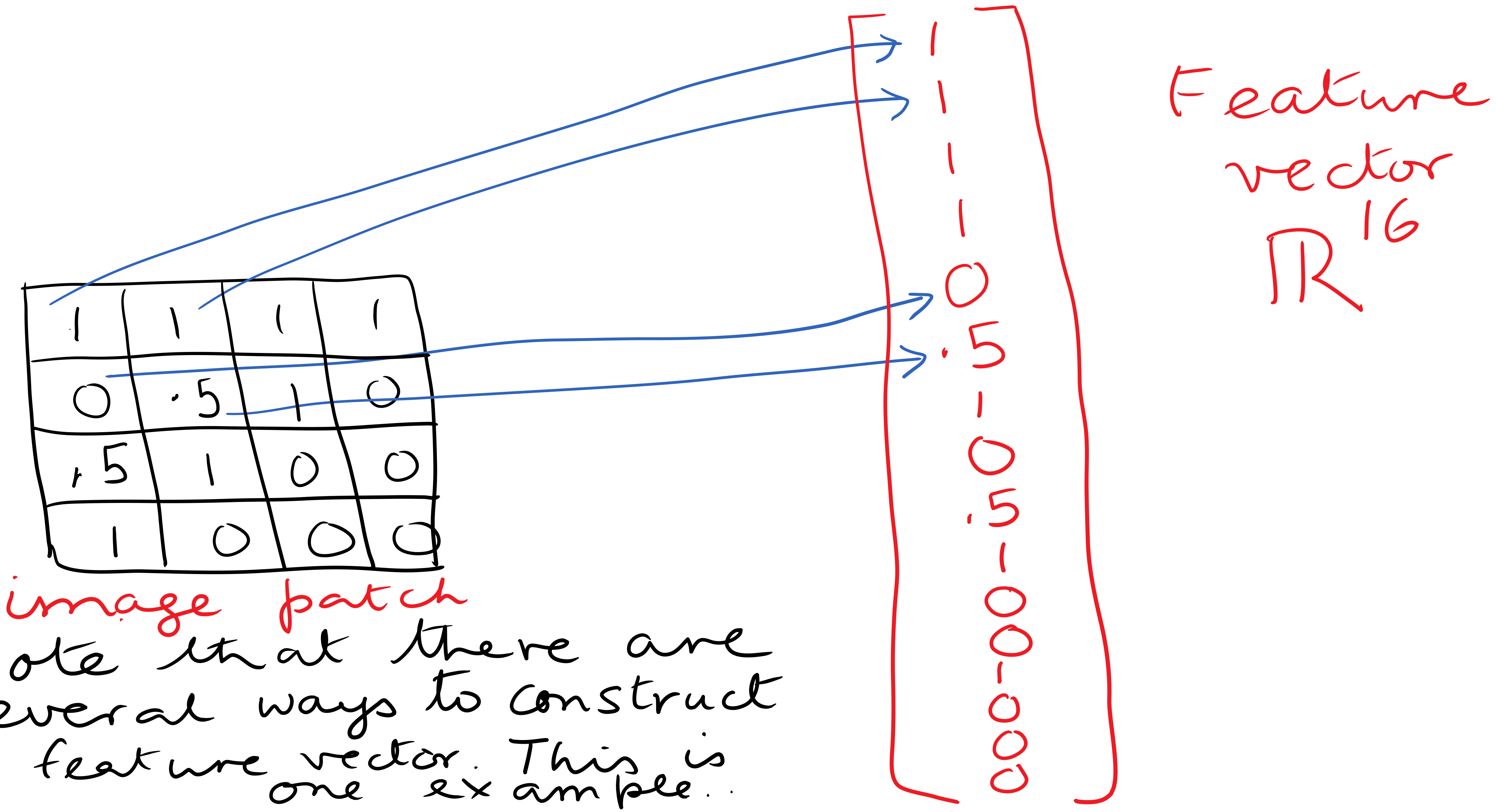


Let us take an example…

1	1	1	1
0	.5	1	0
.5	1	0	0
1	0	0	0

image patch

Let us take an example…



1	1	1	1
0	.5	1	0
.5	1	0	0
1	0	0	0

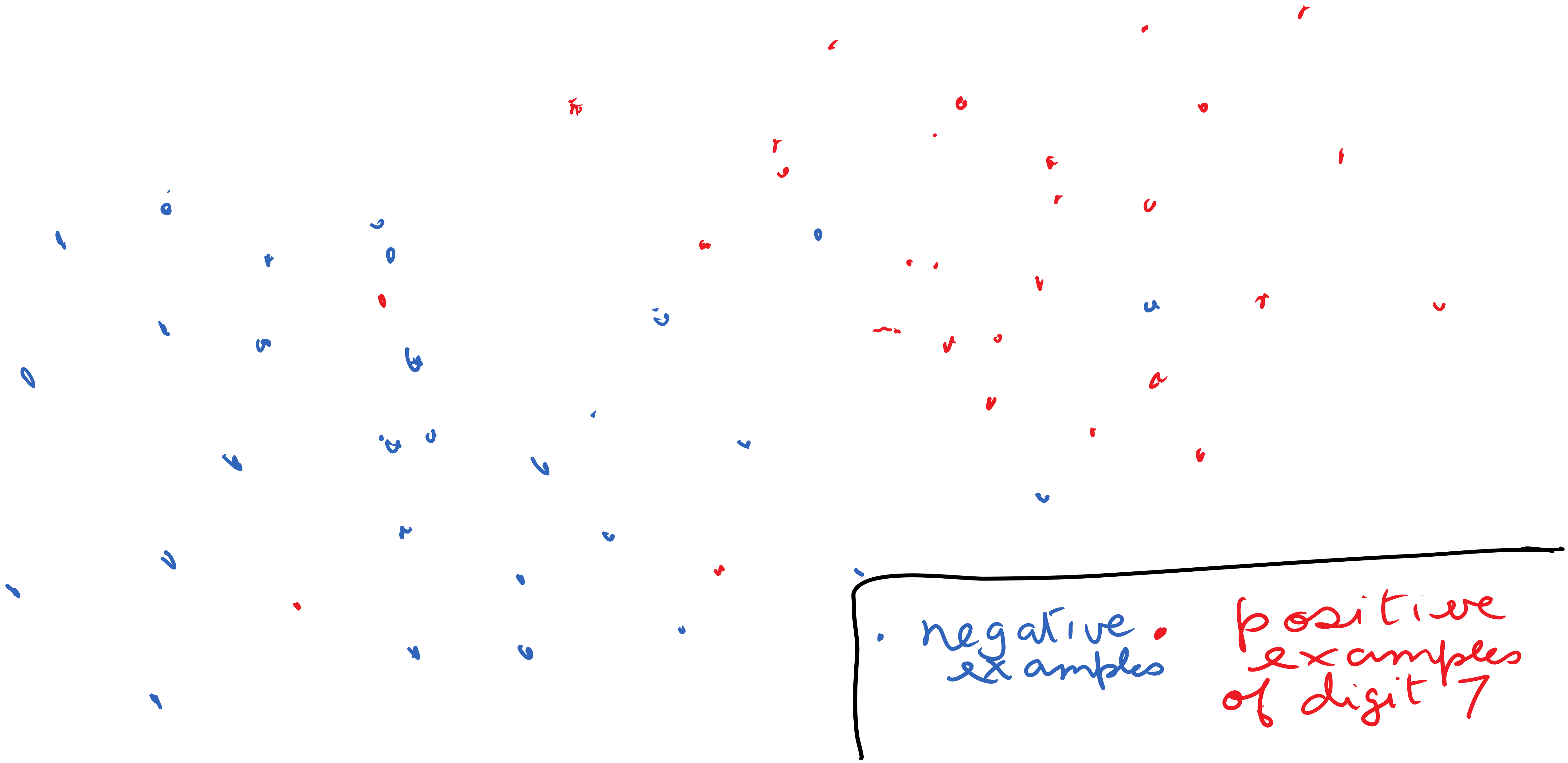
image patch

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ .5 \\ 1 \\ 0 \\ .5 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

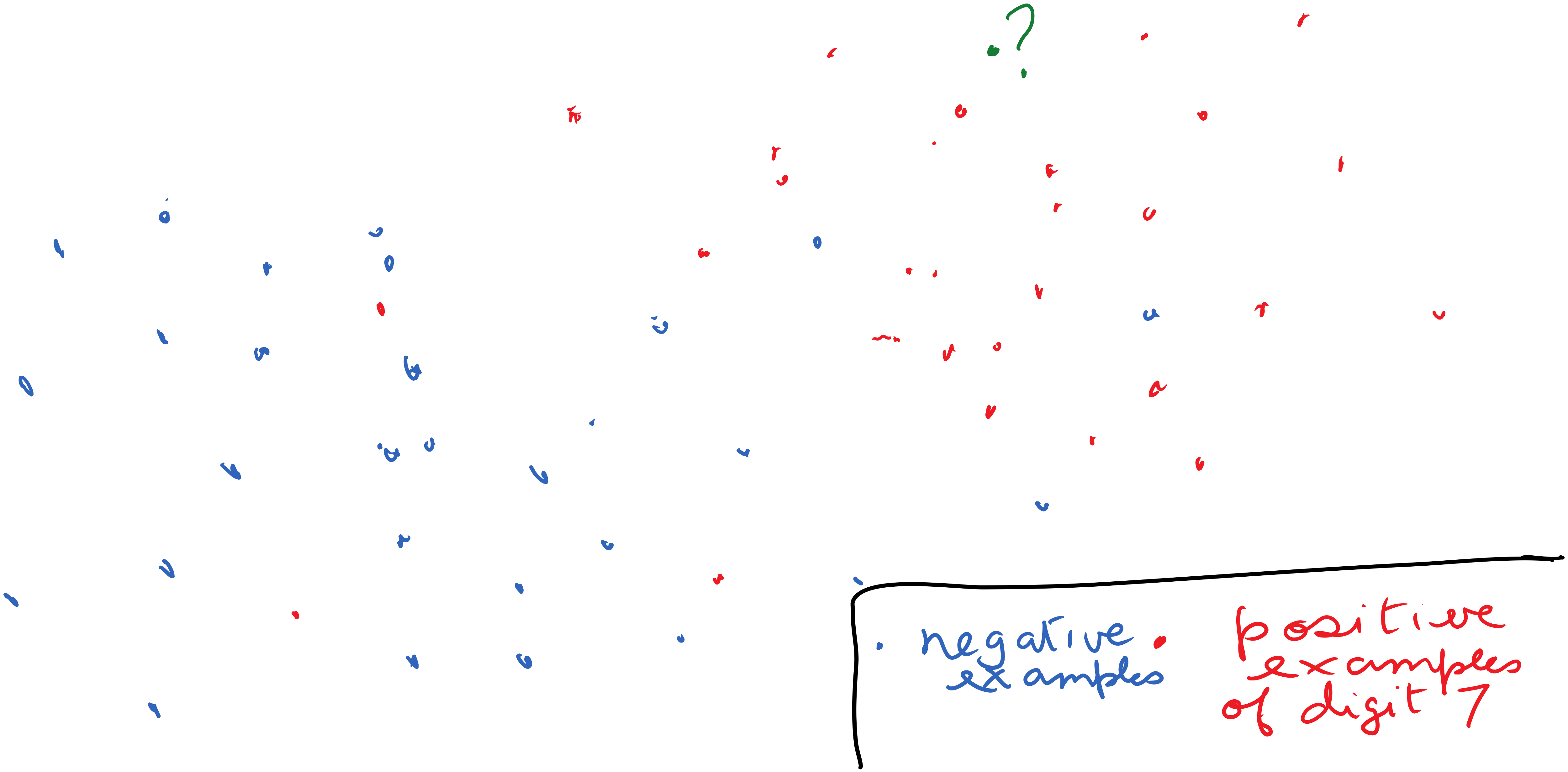
Feature vector $\mathbb{R}^{16}$

note that there are several ways to construct a feature vector. This is one example..

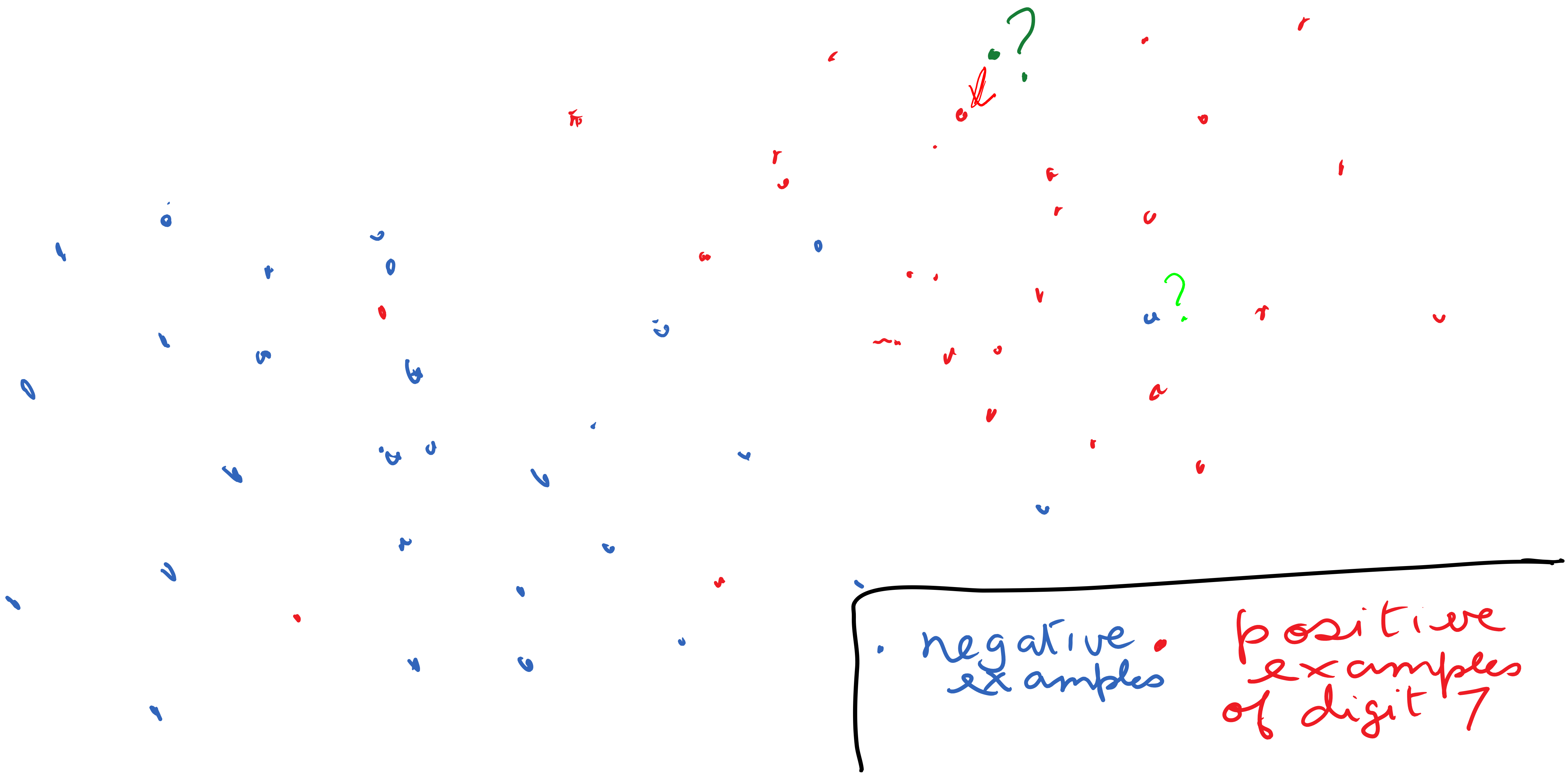
In feature space, positive and negative examples are just points…



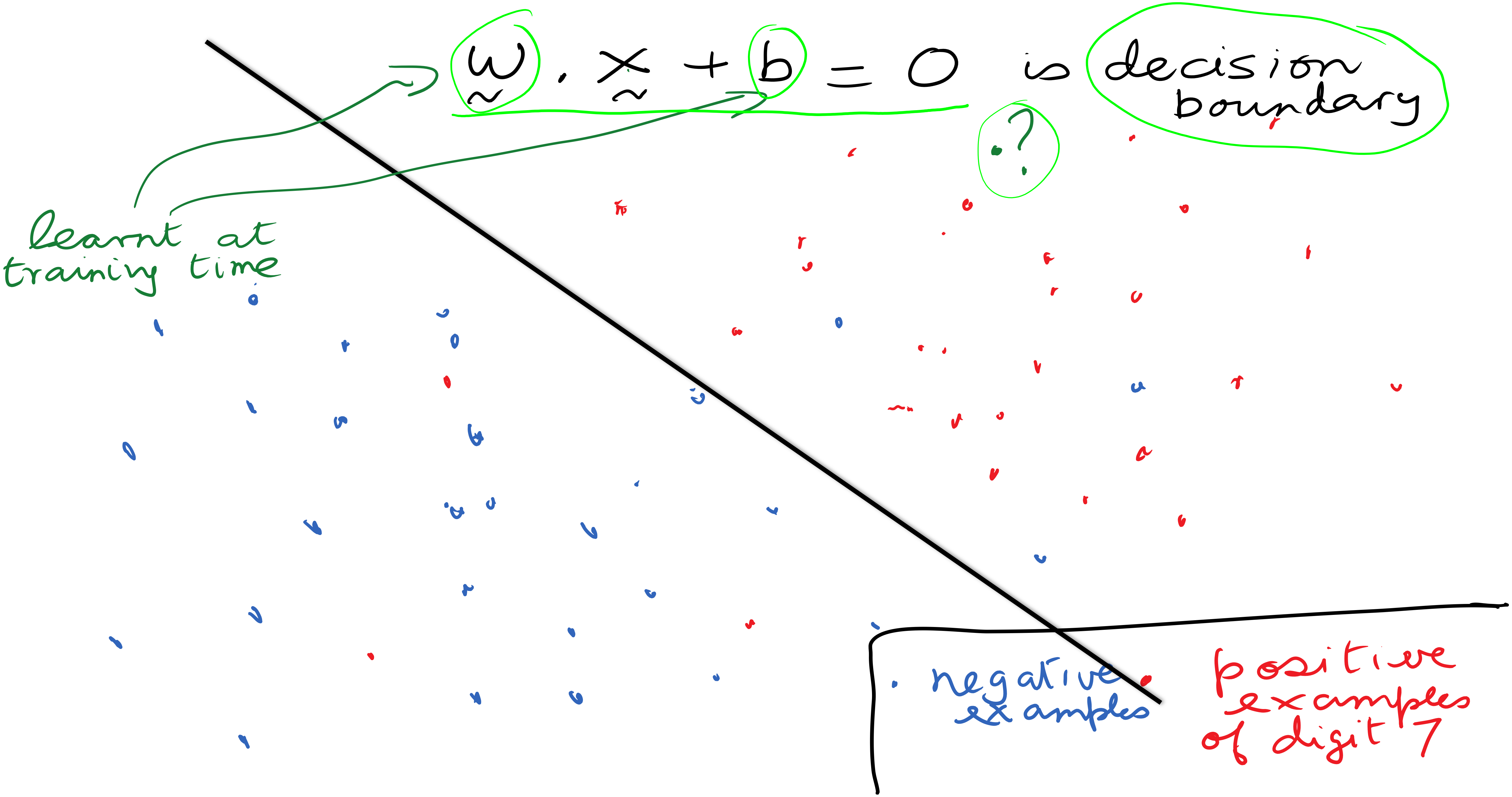
How do we classify a new point?



Nearest neighbor rule
“transfer label of nearest example”



Linear classifier rule



$\underset{\sim}{w} \cdot \underset{\sim}{x} + b = 0$ is decision boundary

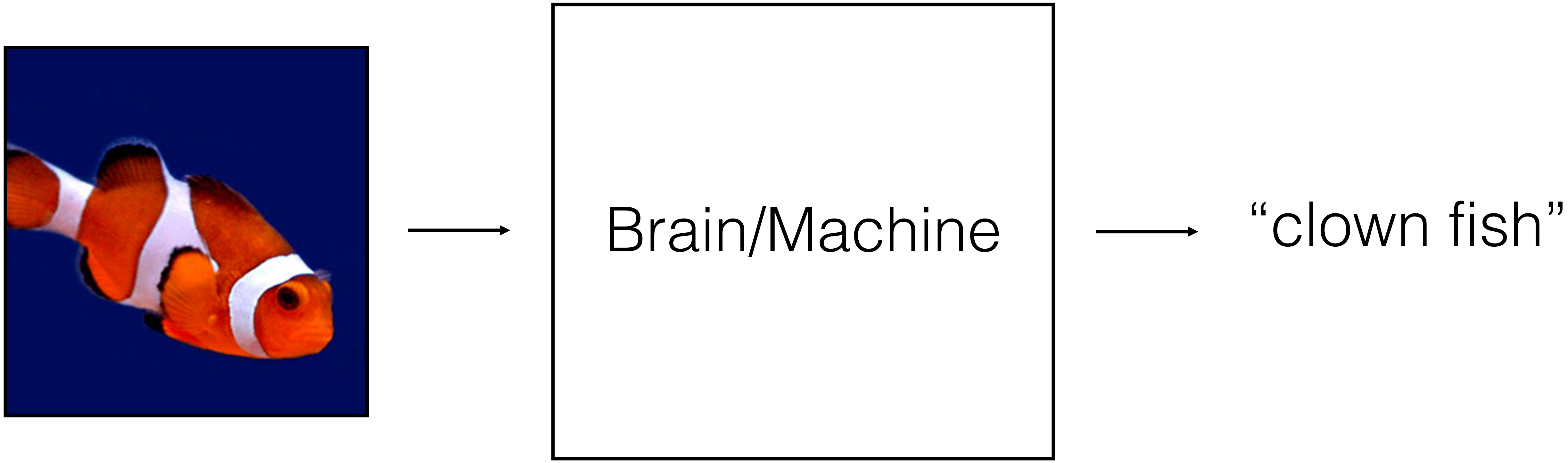
?

learnt at training time

· negative examples

· positive examples of digit 7

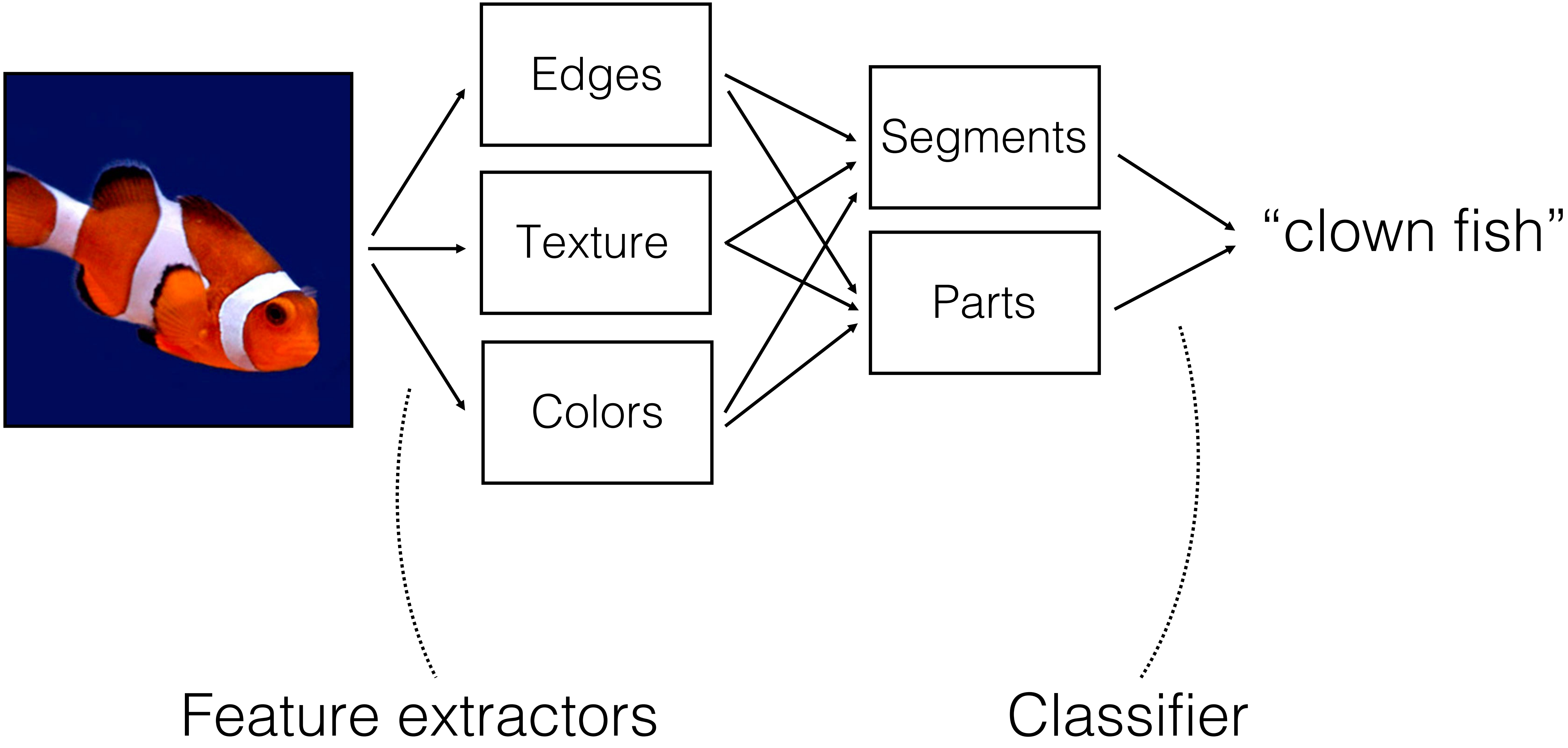
Basic idea



Brain/Machine

“clown fish”

Object recognition



Edges

Texture

Colors

Segments

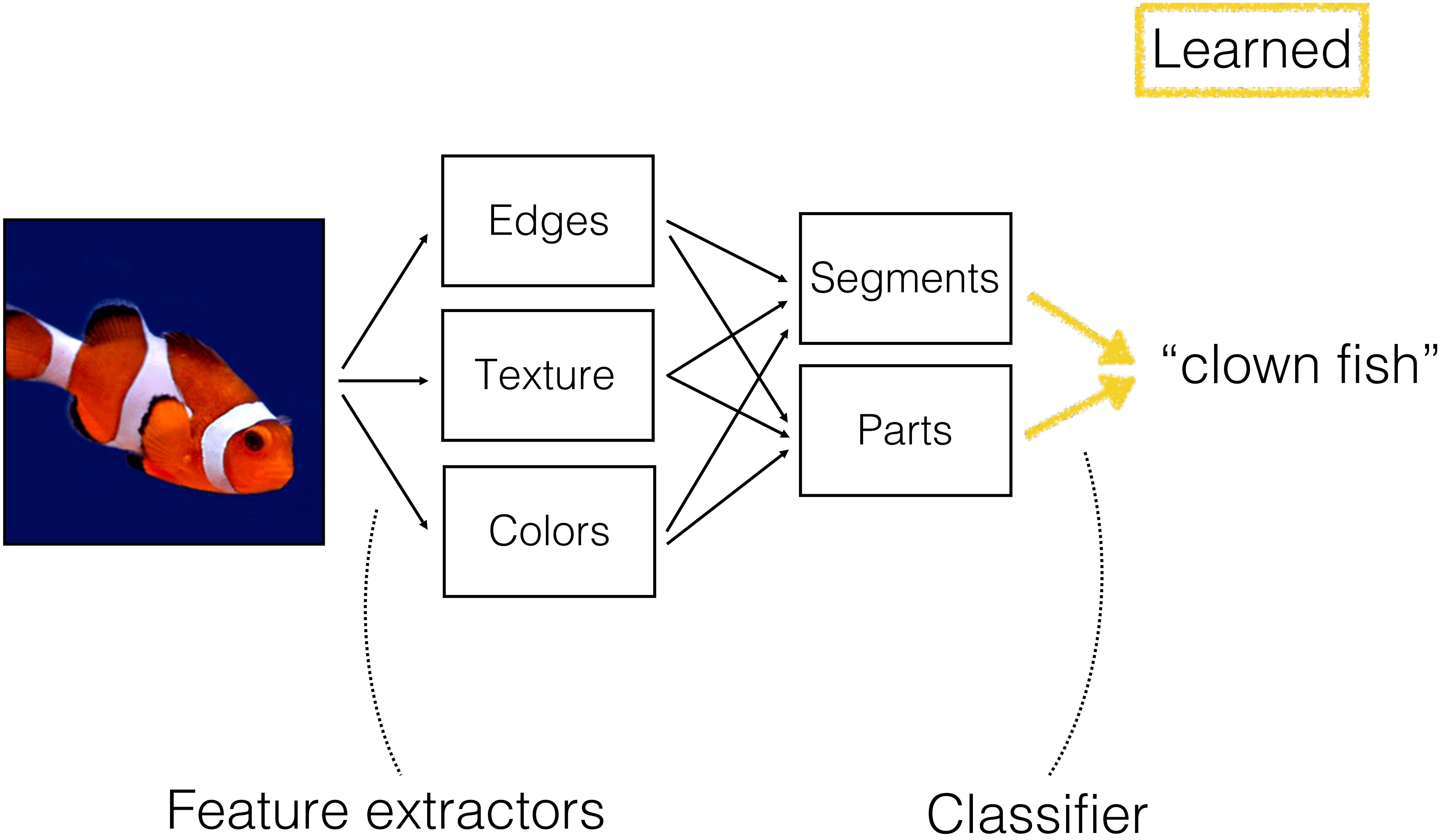
Parts

“clown fish”

Feature extractors

Classifier

Object recognition



Learned

Edges

Texture

Colors

Segments

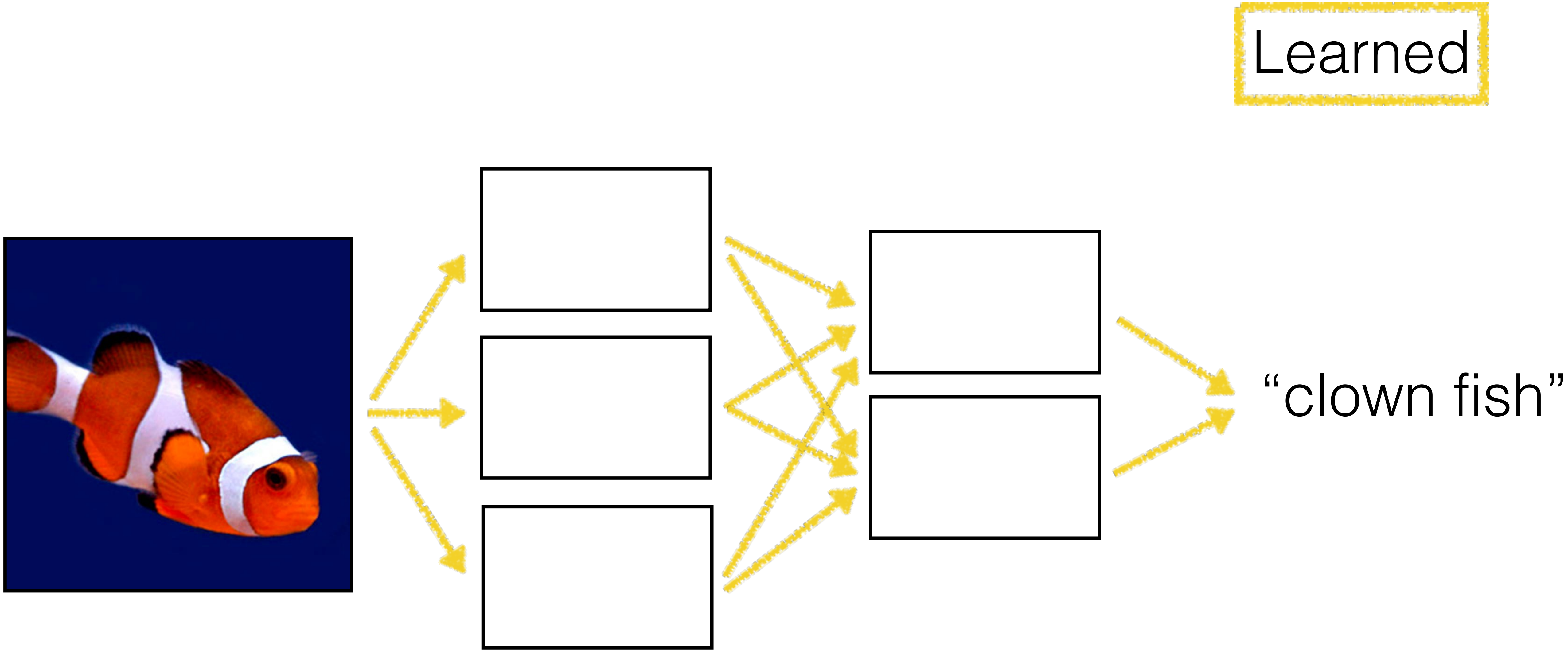
Parts

“clown fish”

Feature extractors

Classifier

Neural network

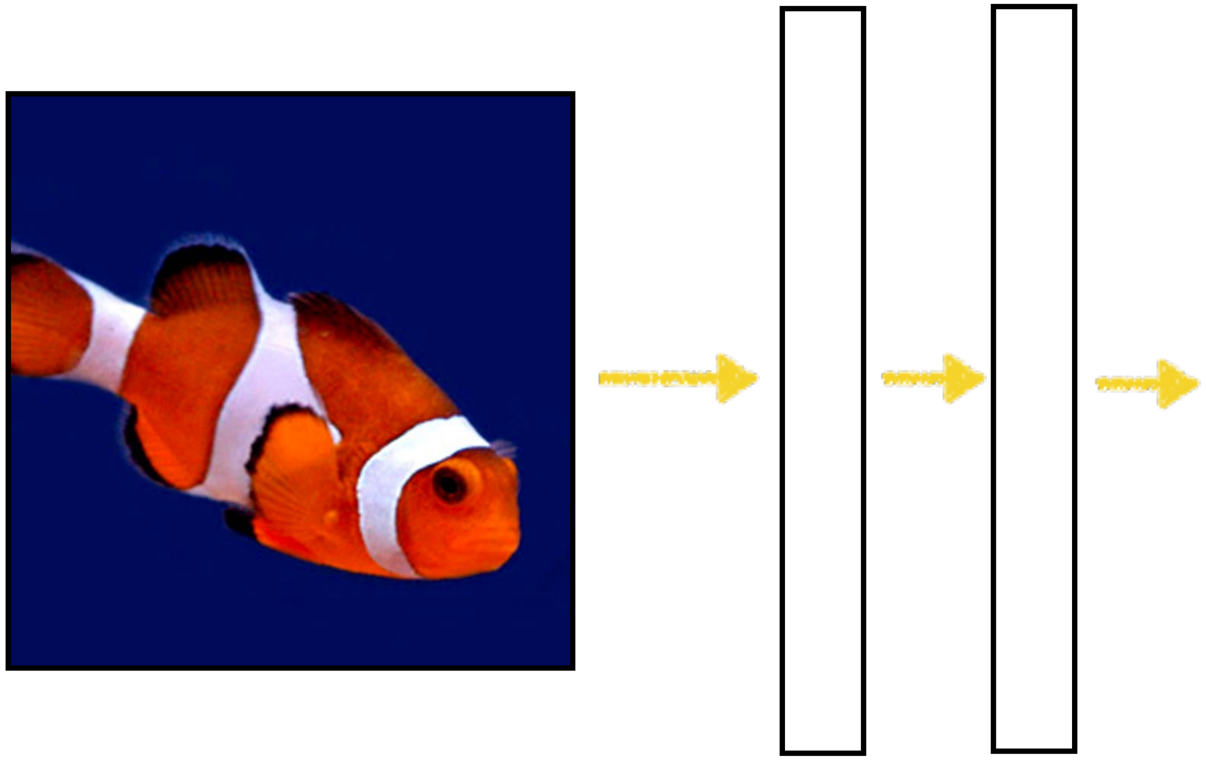


Learned

“clown fish”

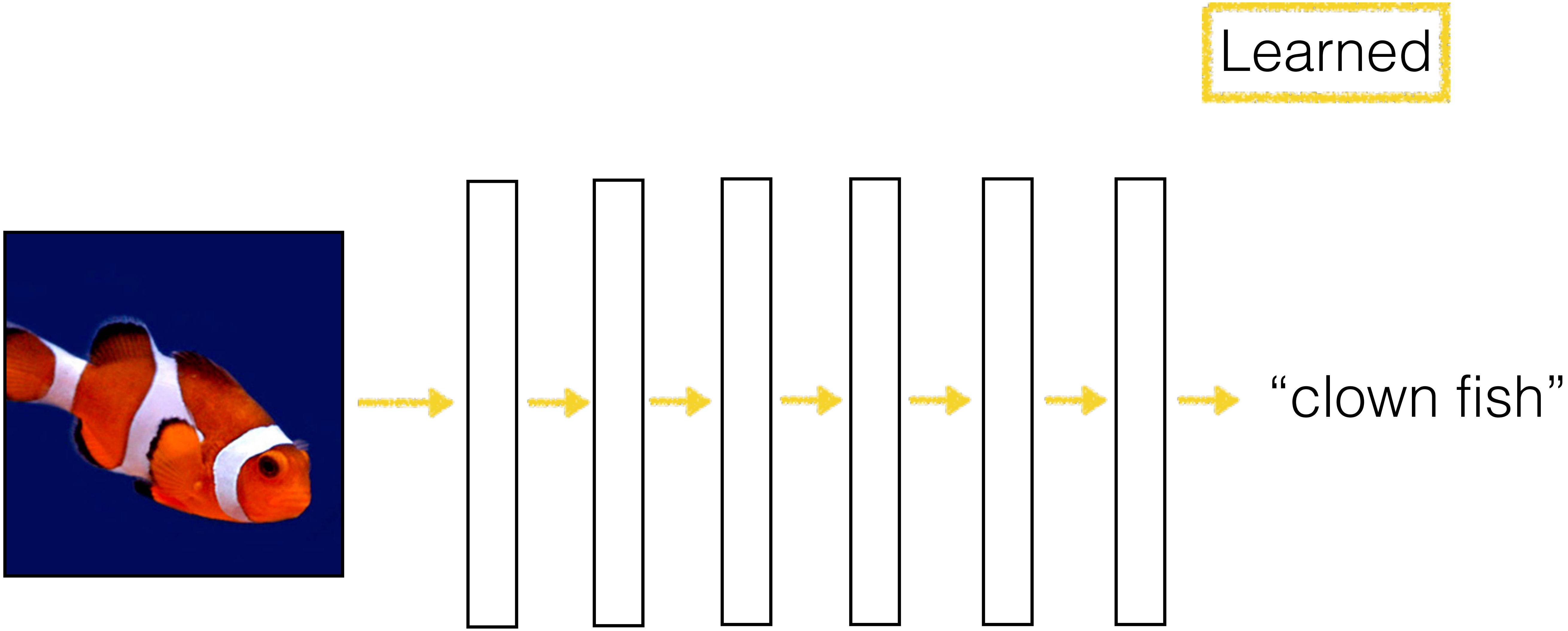
Neural network

Learned



“clown fish”

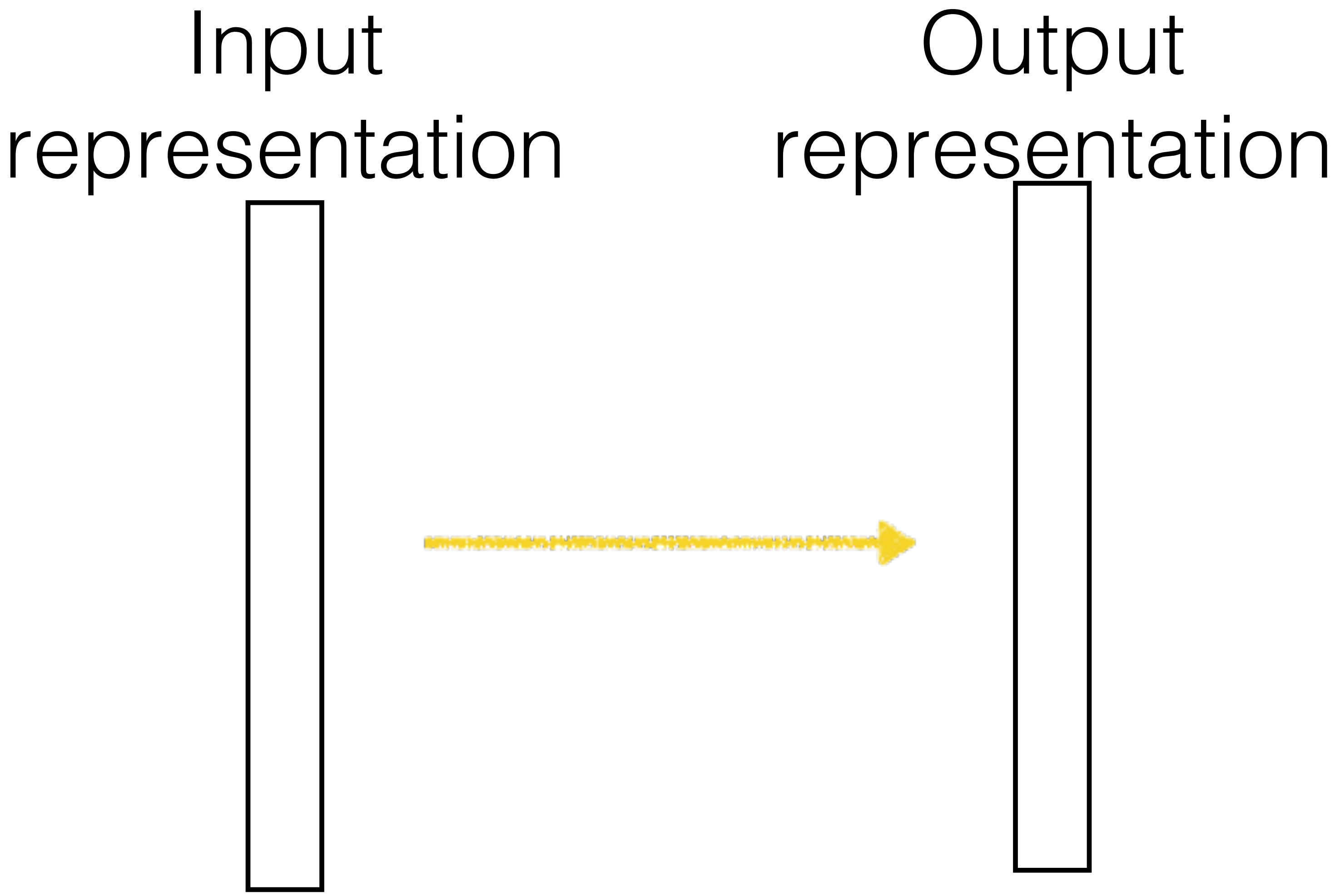
Deep neural network



Learned

“clown fish”

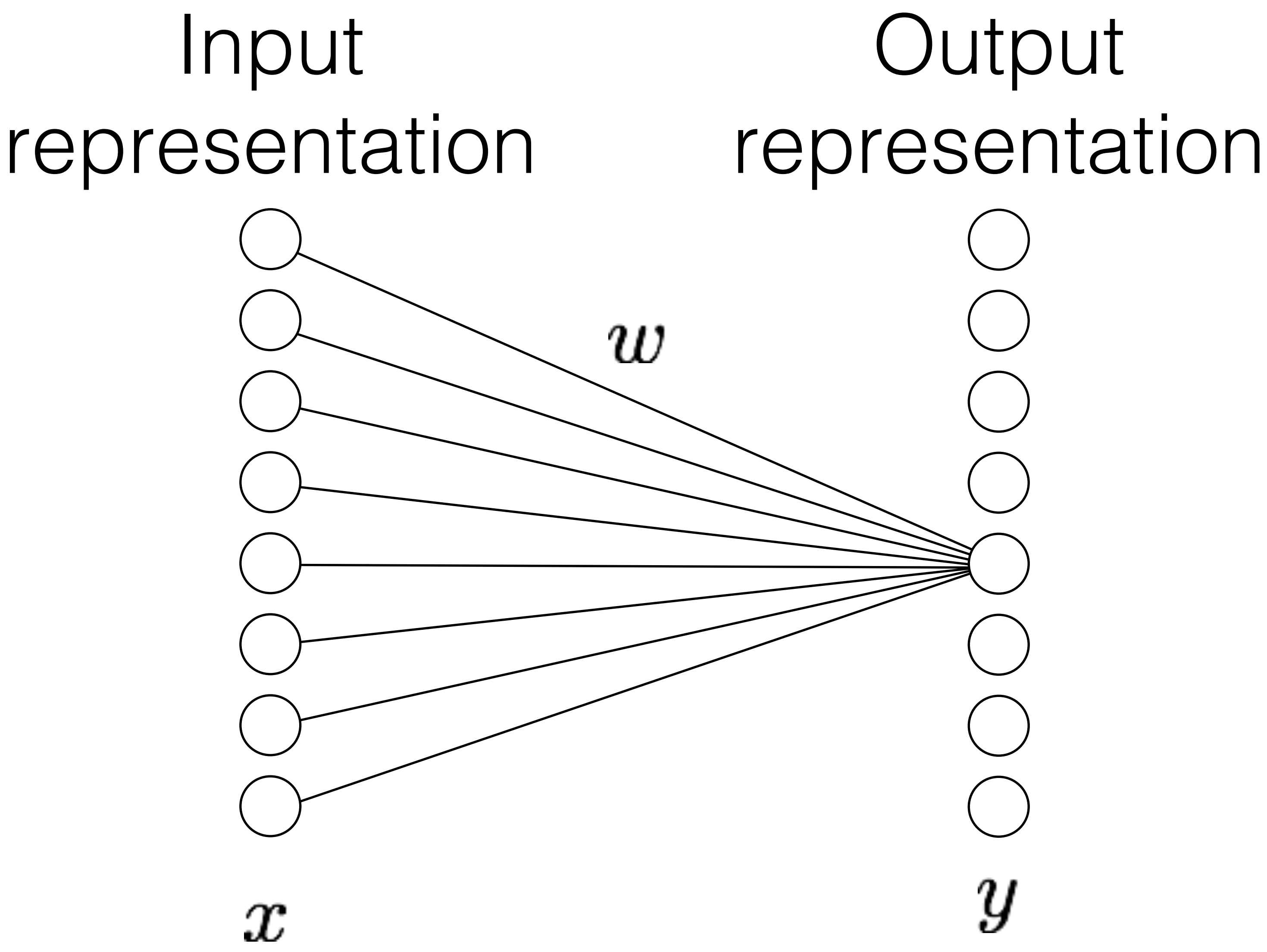
Computation in a neural net



Input representation

Output representation

Computation in a neural net



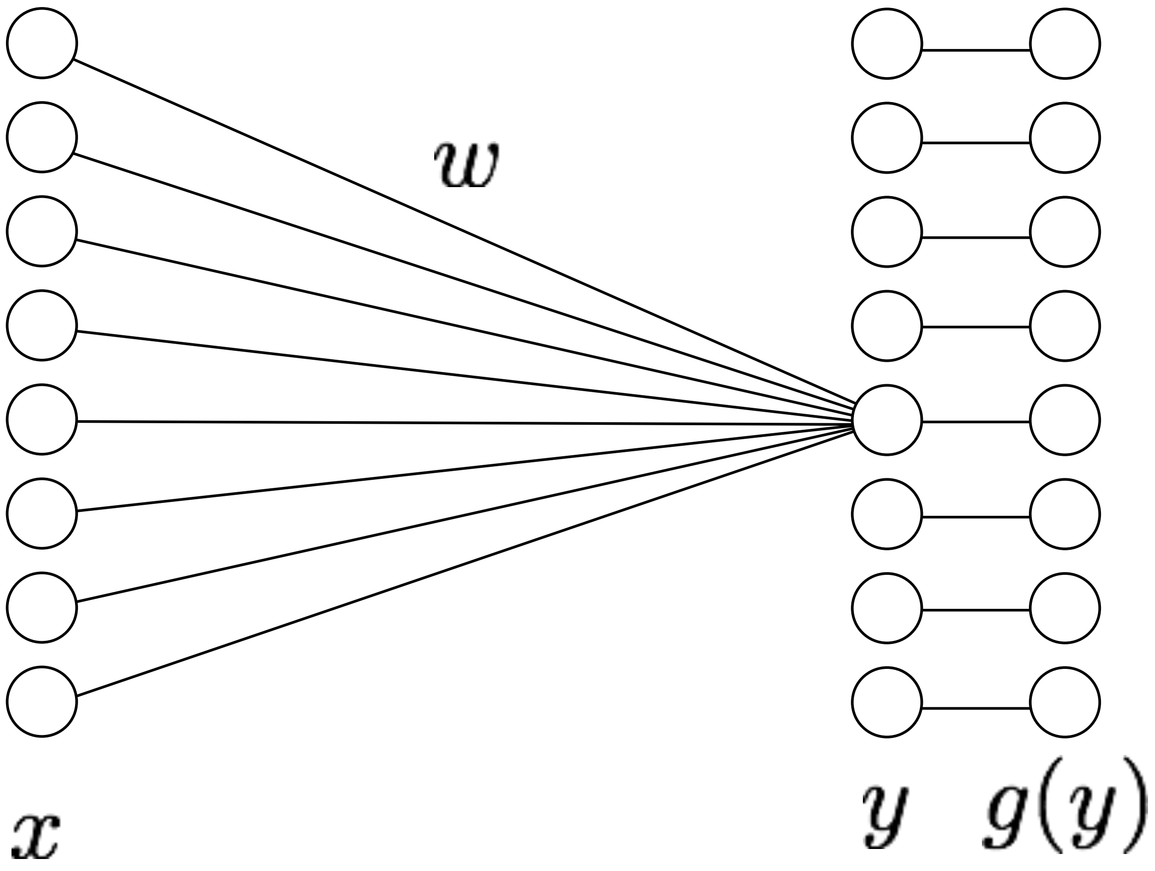
Input representation

Output representation

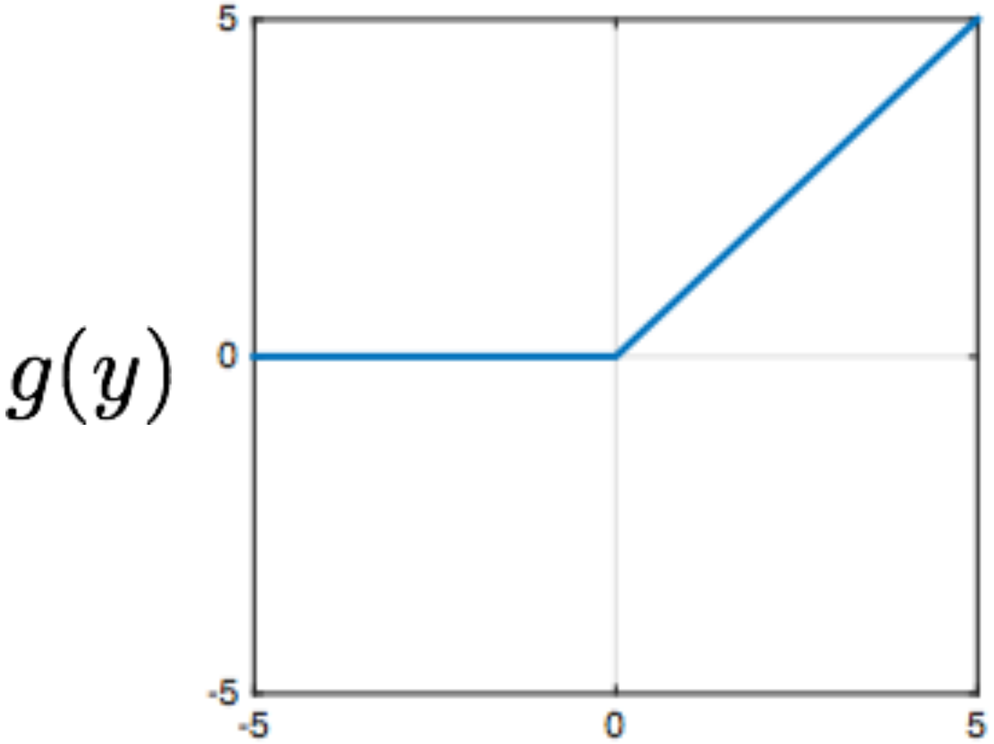
$$y_j = \sum_i w_{ij} x_i$$

i: the $\boldsymbol{i^{th}}$ dimension of $\boldsymbol{x}$, j; the $\boldsymbol{j^{th}}$ dimension of $\boldsymbol{y}$

Computation in a neural net



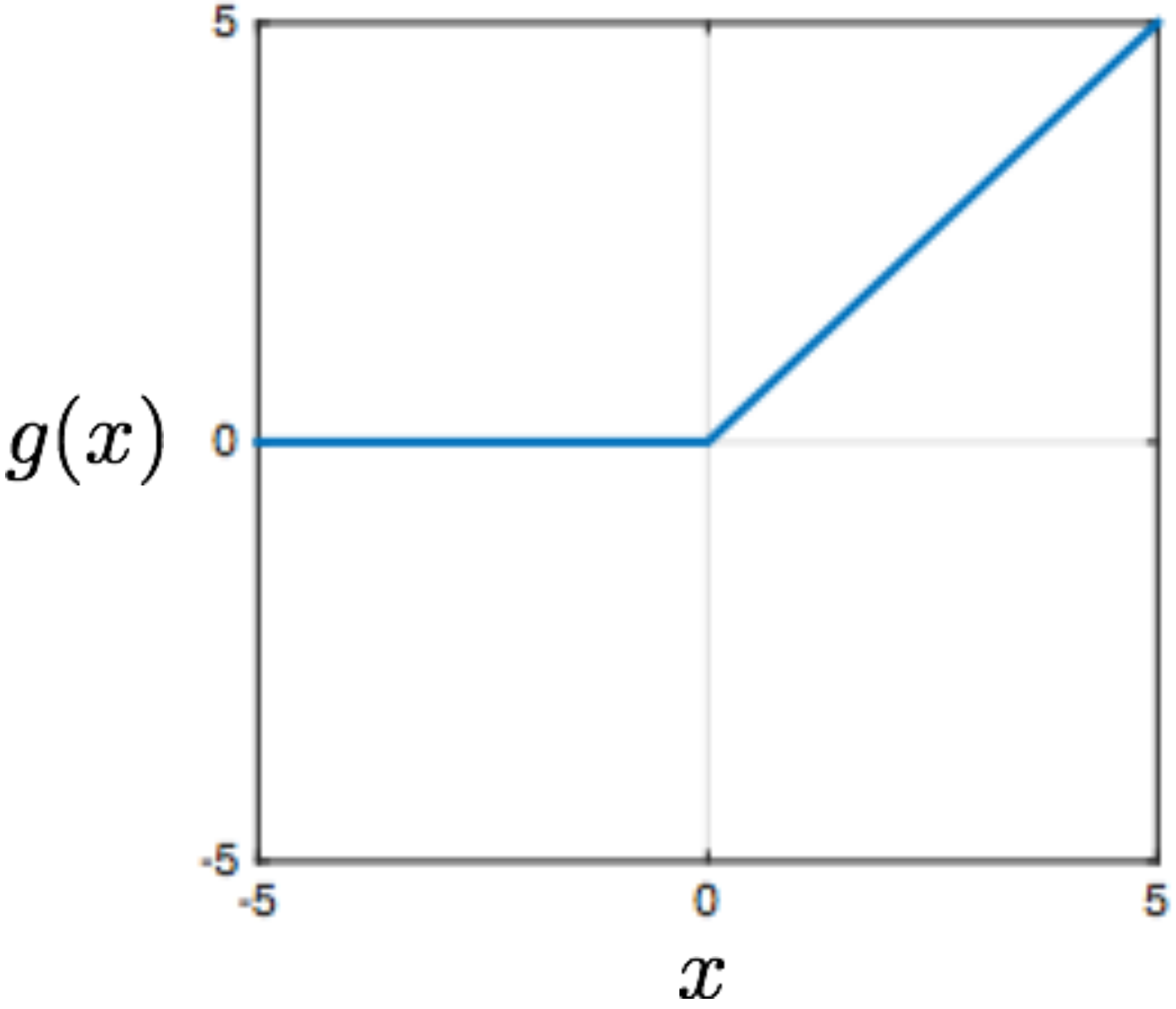
Rectified linear unit (ReLU)



$$g(y) = \max(0, y)$$

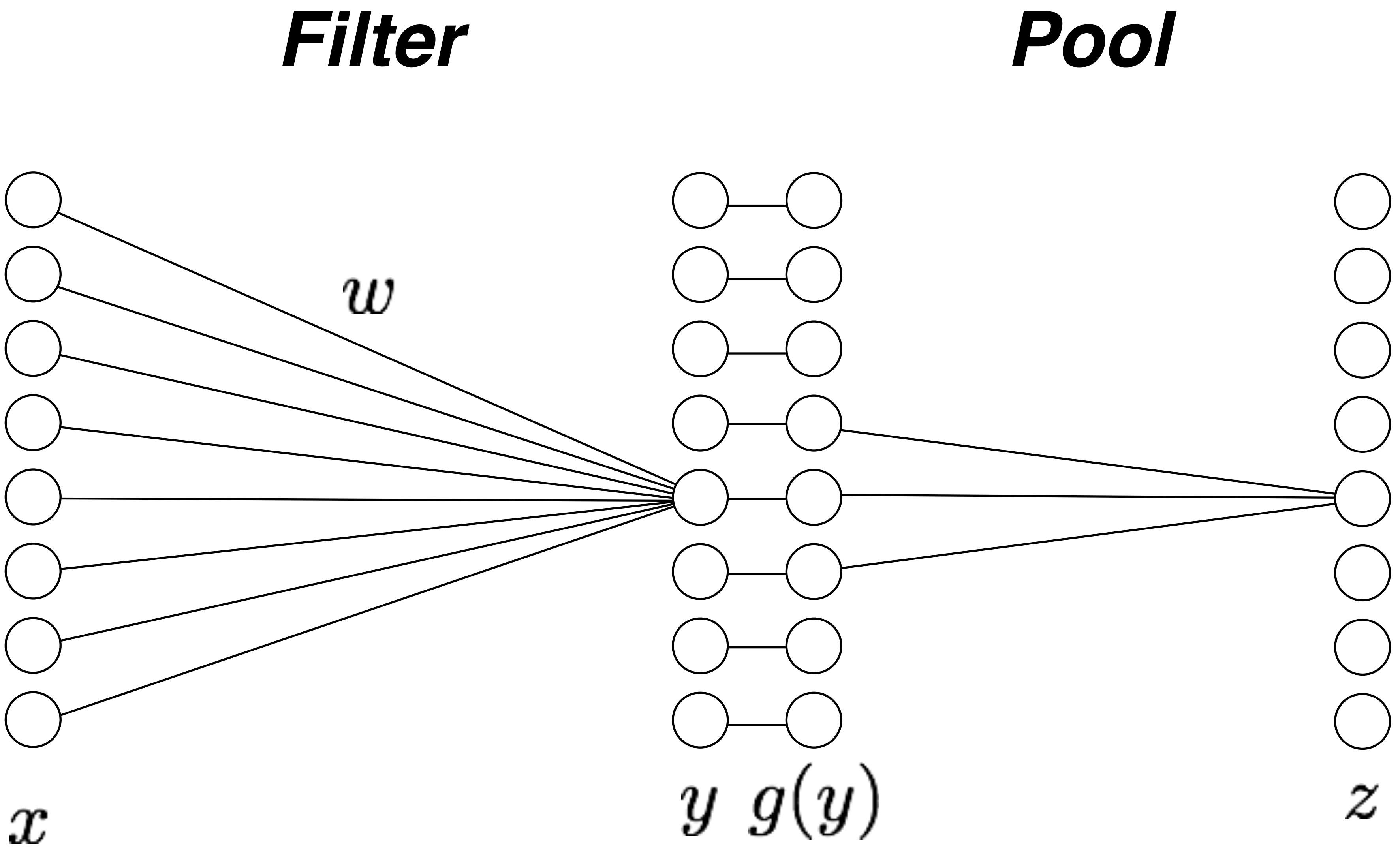
Computation in a neural net

Rectified linear unit (ReLU)



$$g(x) = \max(0, x)$$

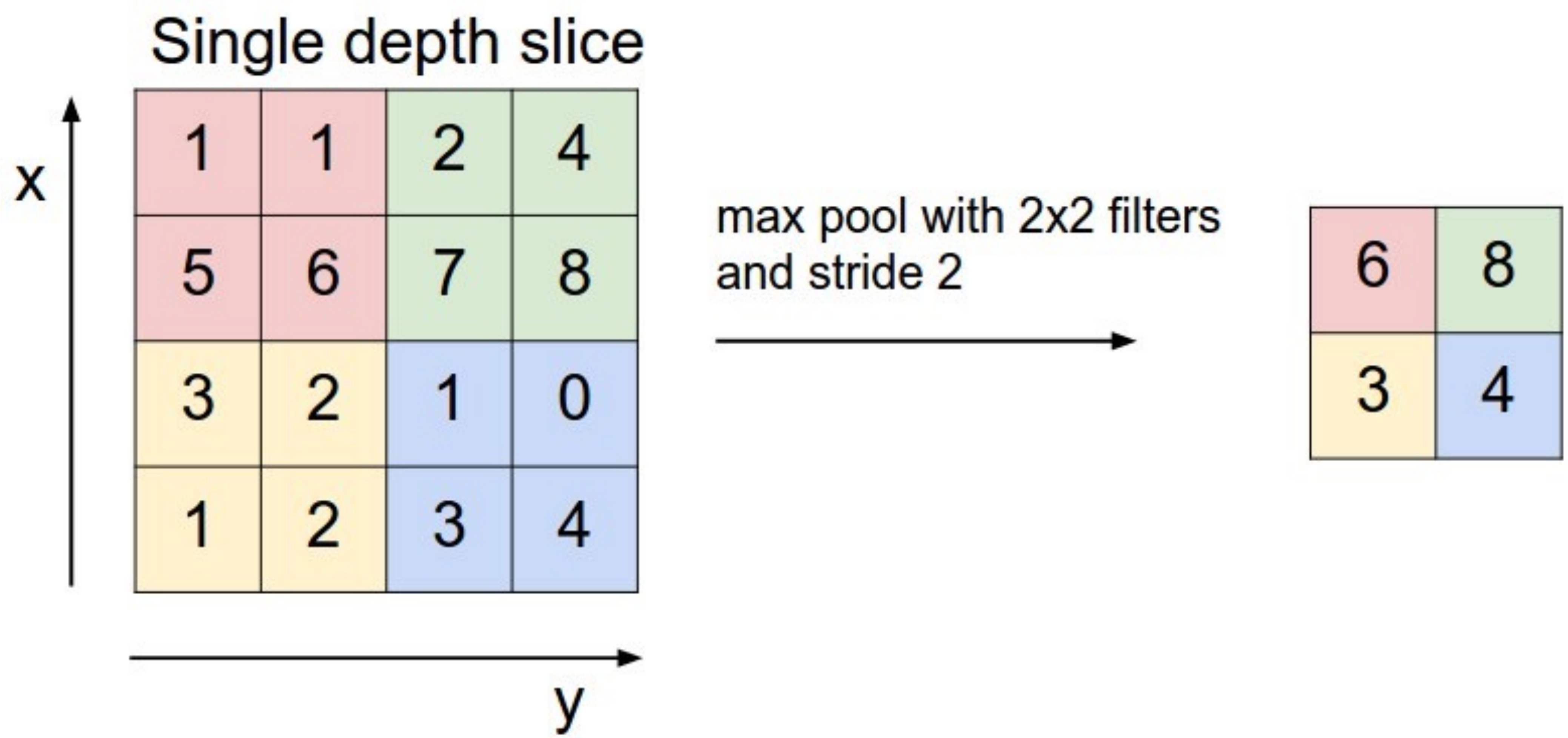
Computation in a neural net



$$y_j = \sum_i w_{ij} x_i \qquad z_k = \max_{j \in \mathcal{N}(j)} g(y_j)$$

i: the i^{th} dimension of x, j; the j^{th} dimension of y

Computation in a neural net

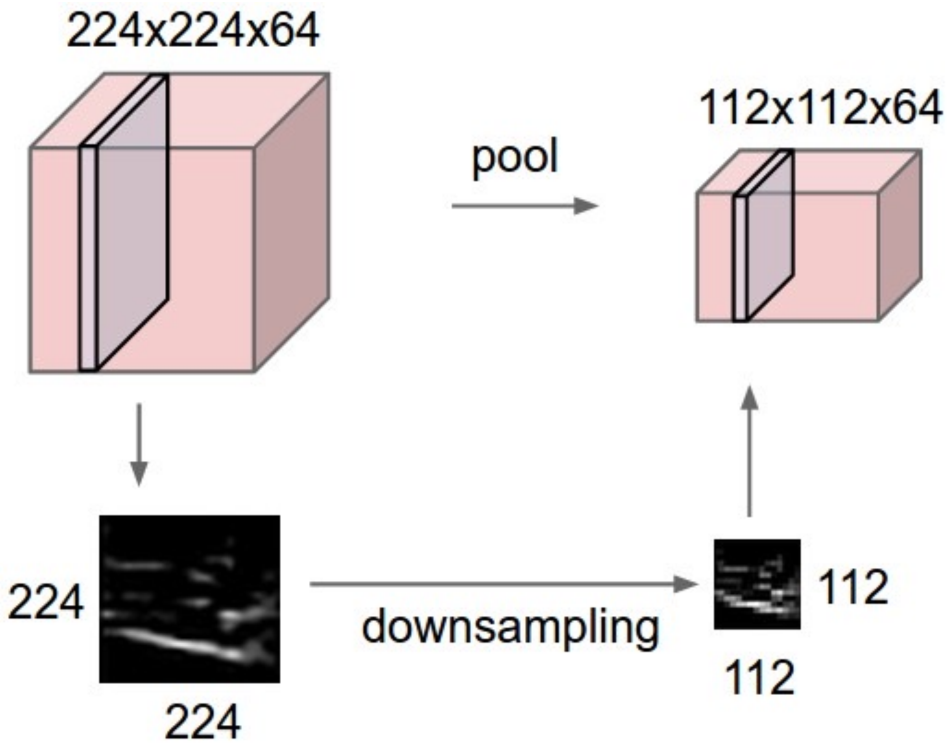


Single depth slice

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

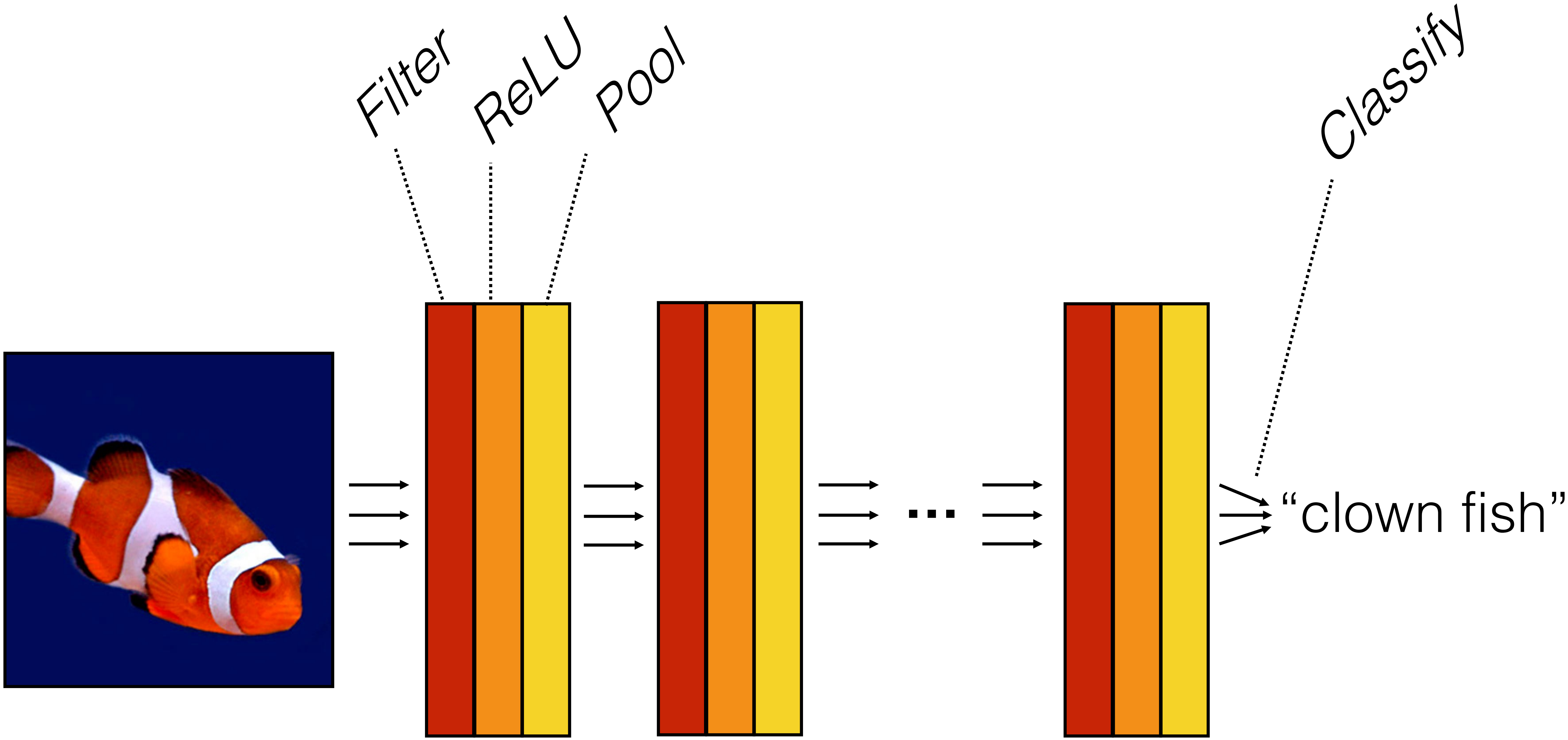
max pool with 2x2 filters and stride 2

6	8
3	4



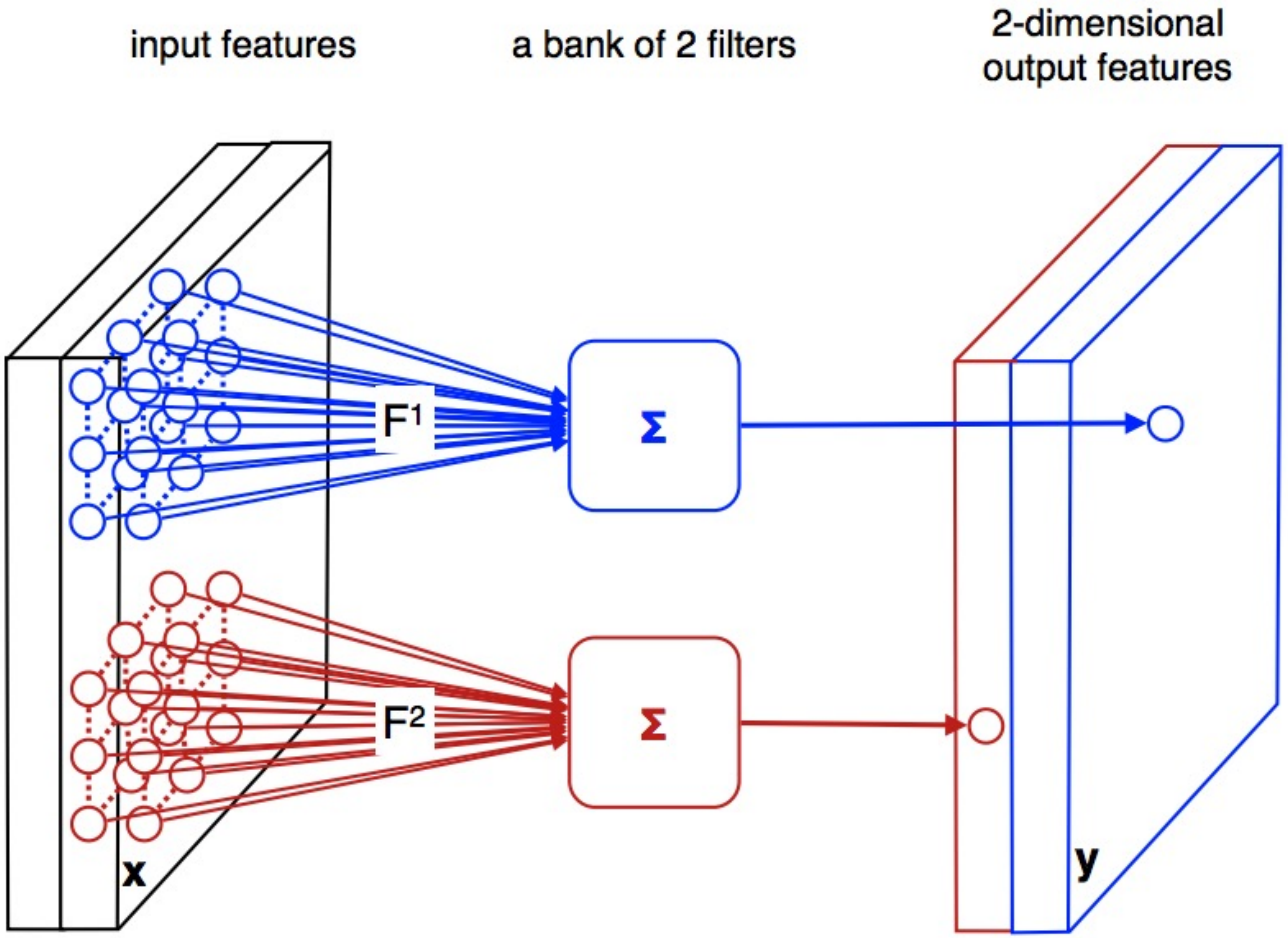
CS231n at Stanford

Computation in a neural net



$$f(\mathbf{x}) = f_L(\ldots f_2(f_1(\mathbf{x})))$$

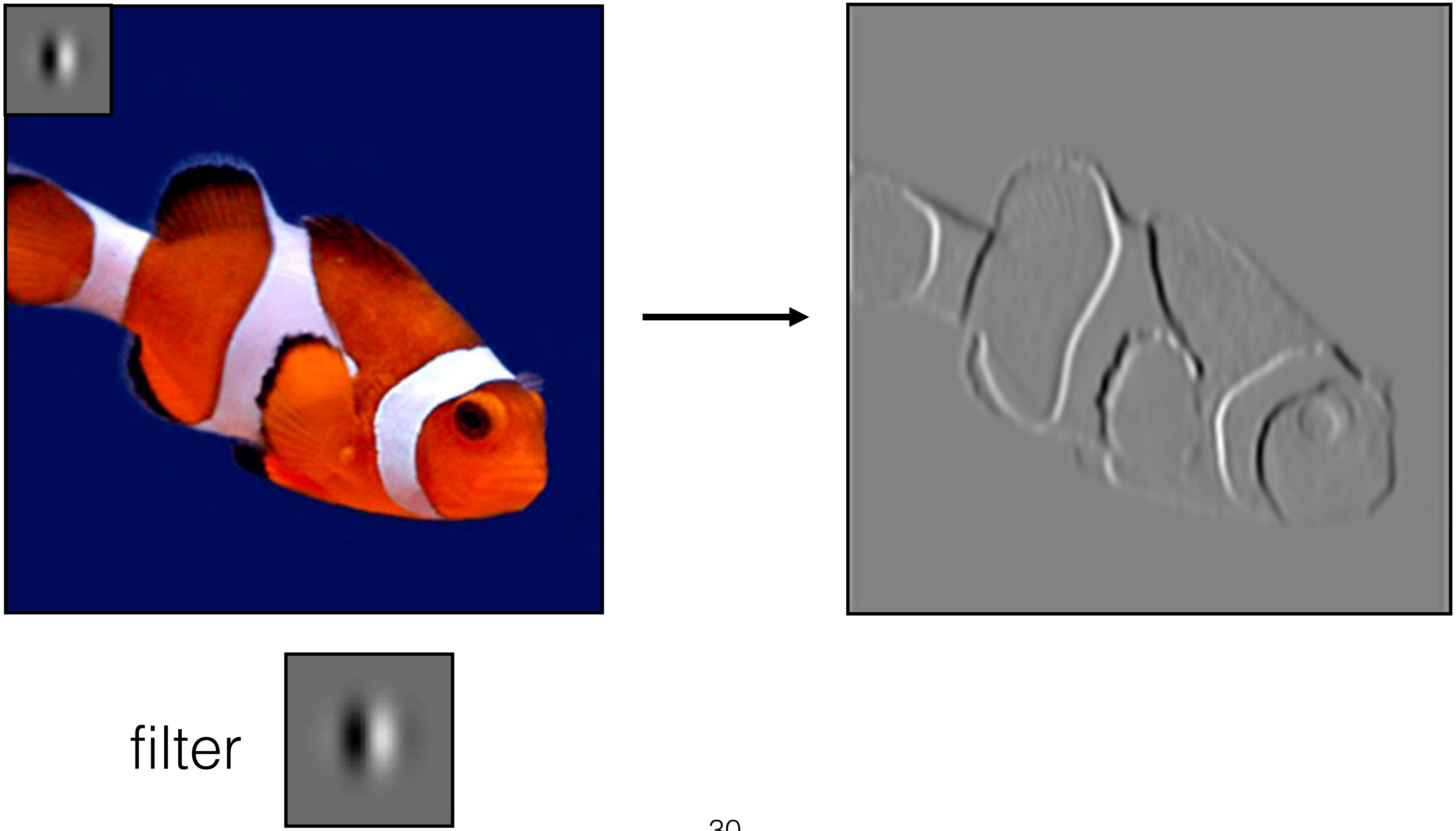
Convolutional Neural Nets



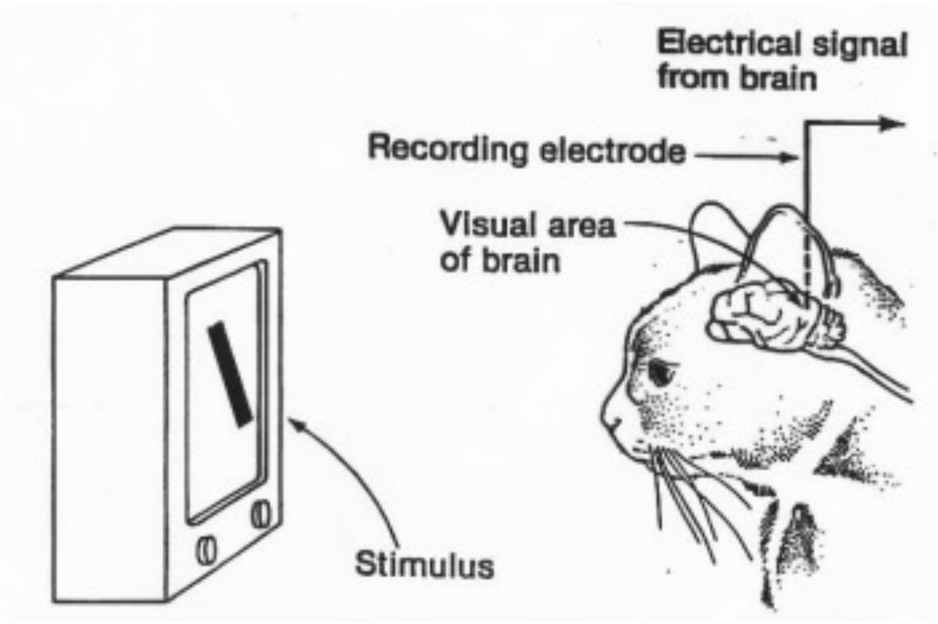
Slide from Andrea Vedaldi

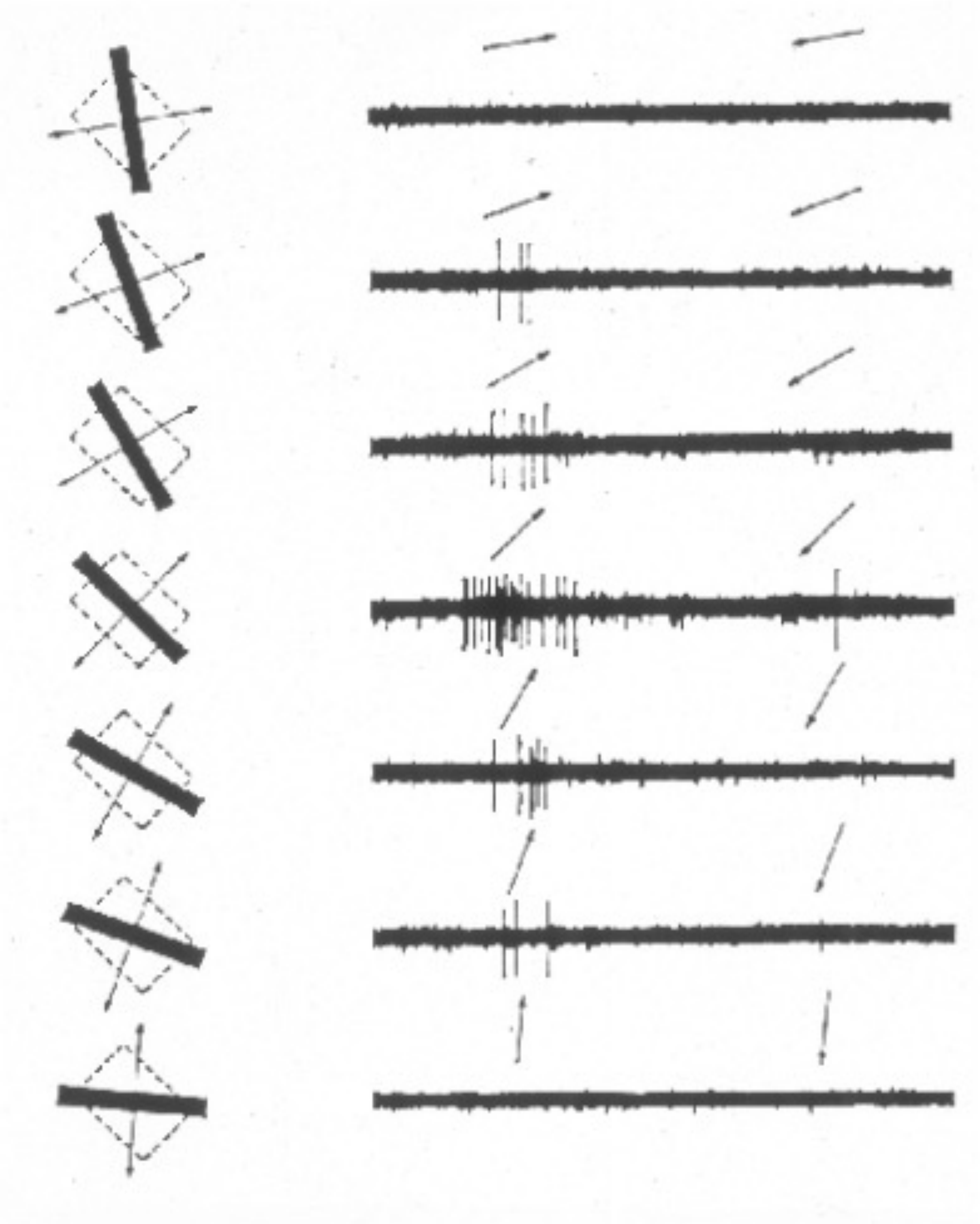
Convolutional Neural Nets

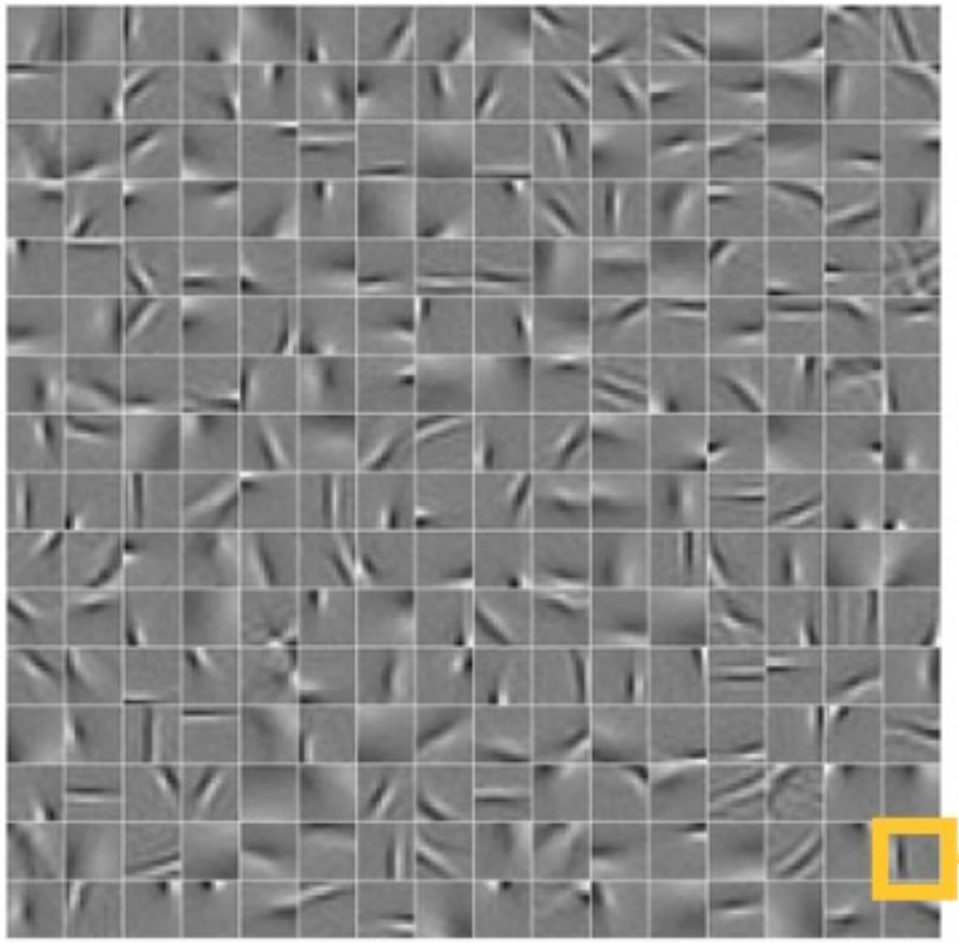
Convolution

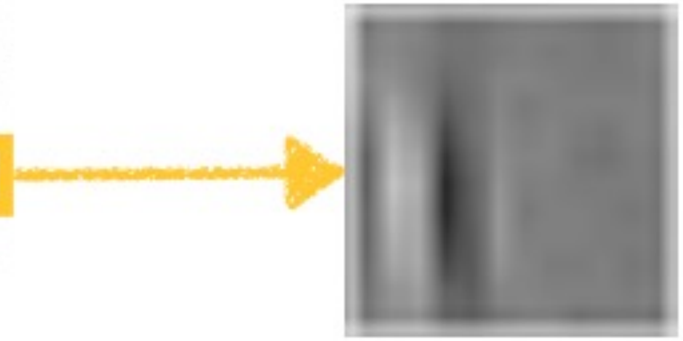


[Hubel and Wiesel 59]





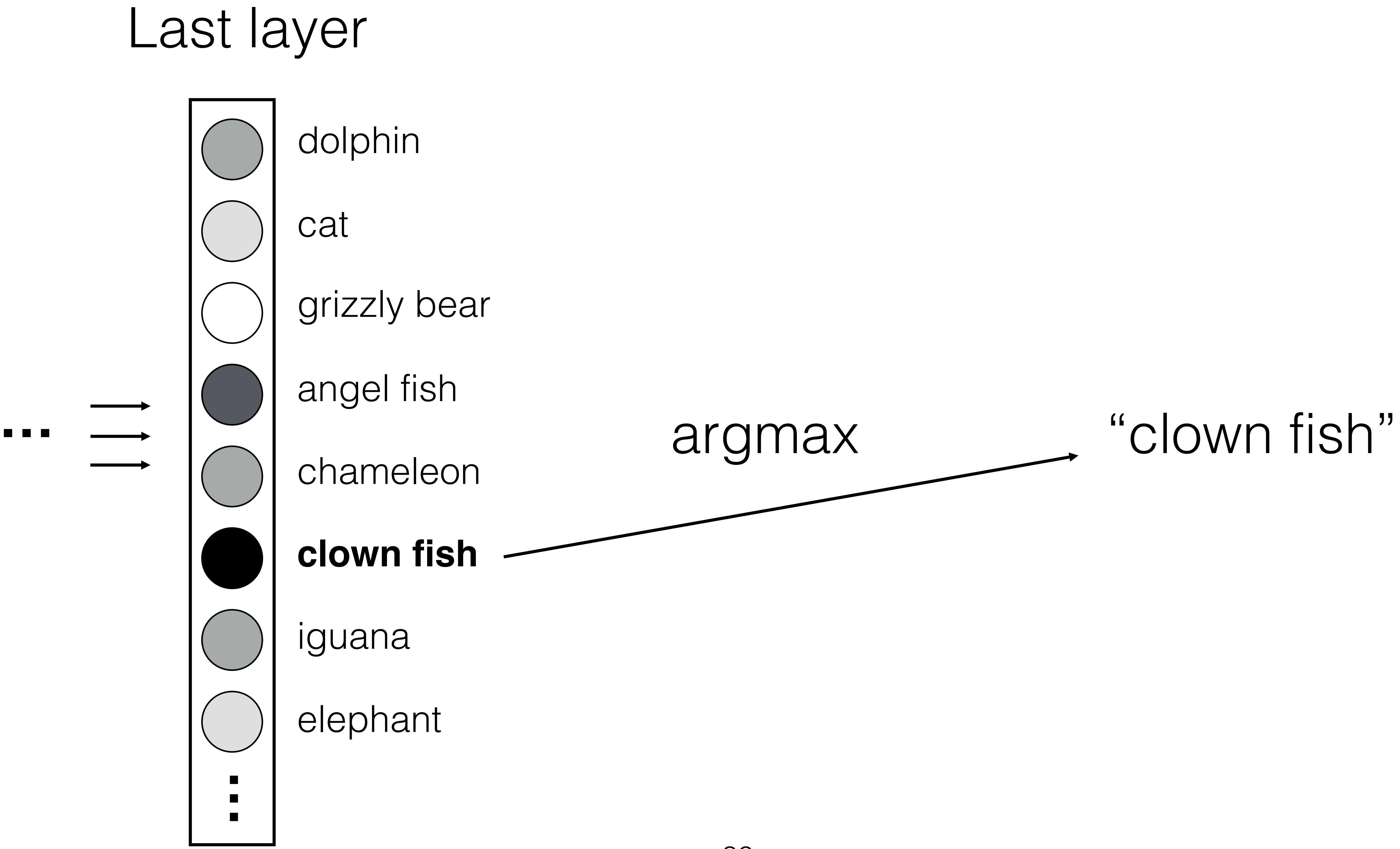




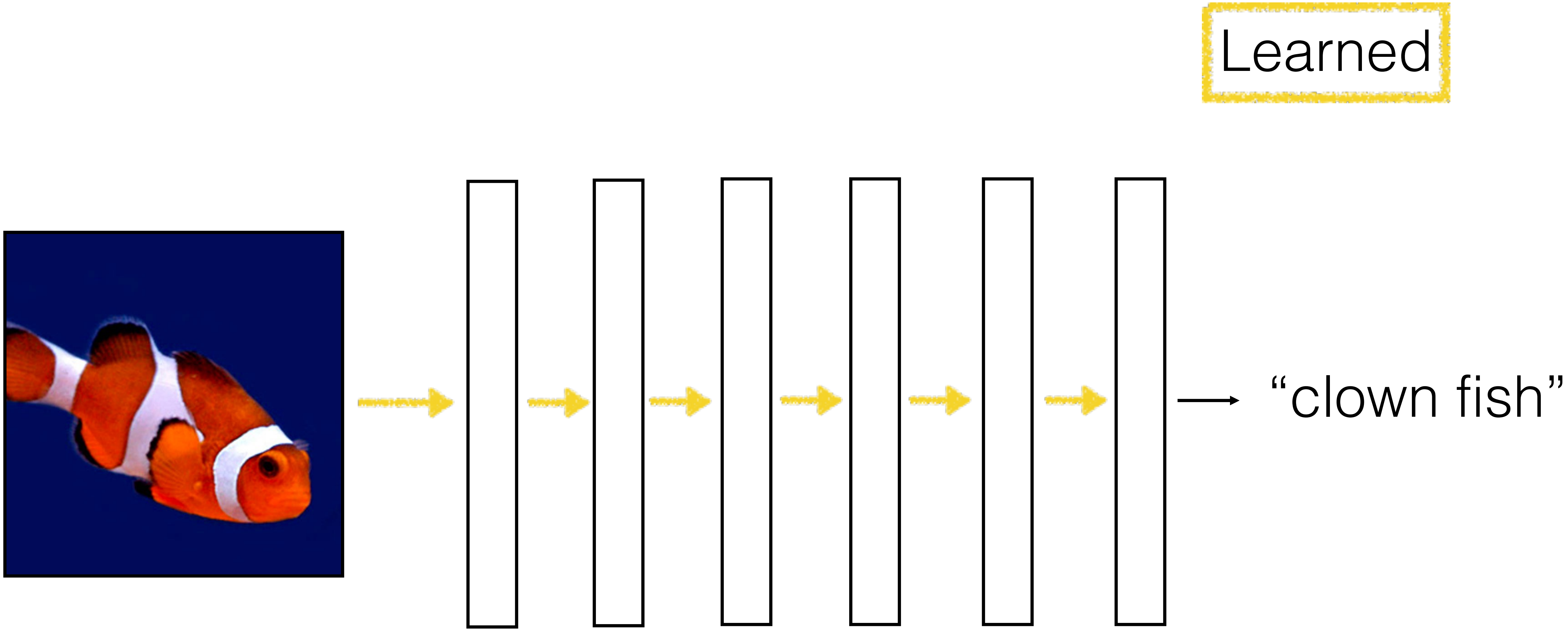
oriented filter

Slide from Andrea Vedaldi

Computation in a neural net



Learning with deep nets



Learned

“clown fish”

Learning with deep nets

→ “clown fish”

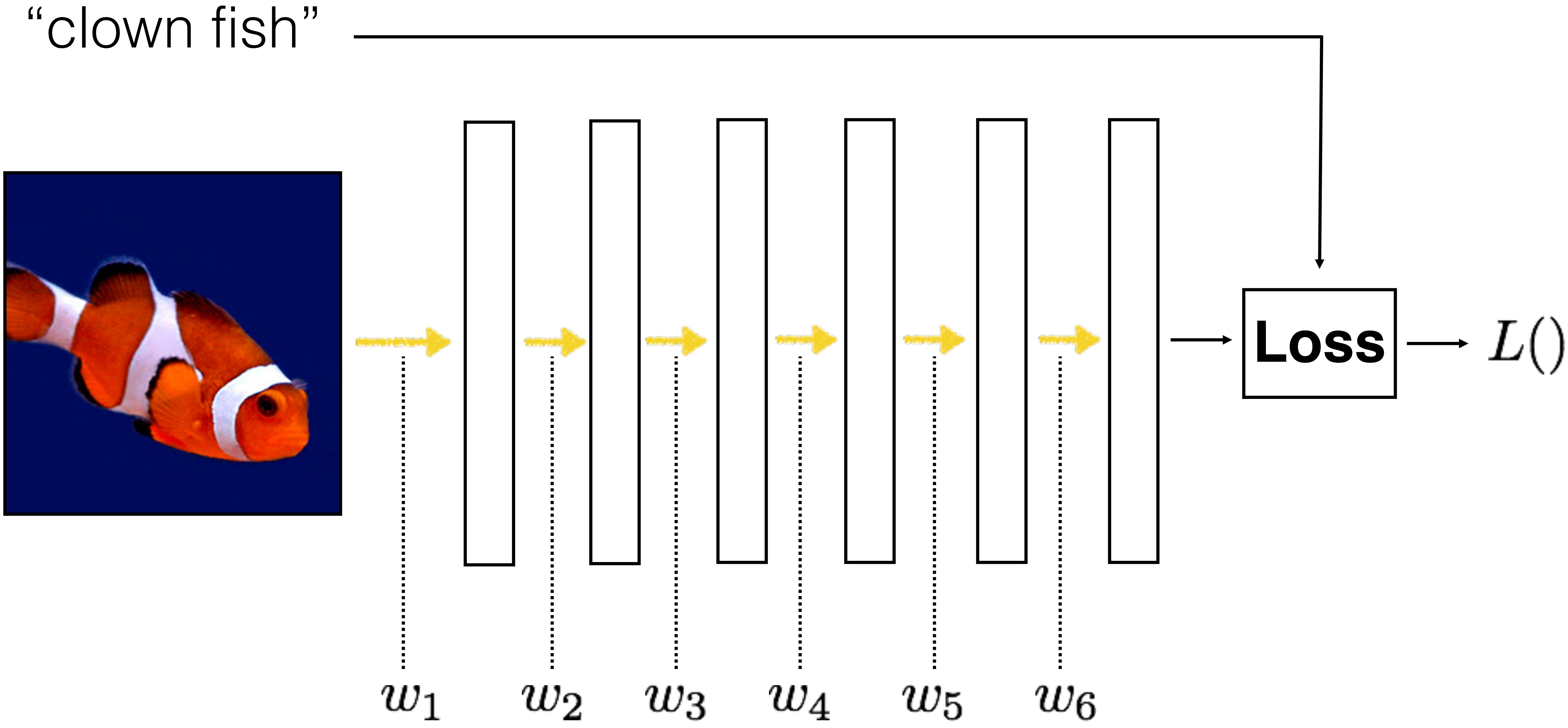
→ “grizzly bear”

→ “chameleon”

Train network to associate the right label with each image

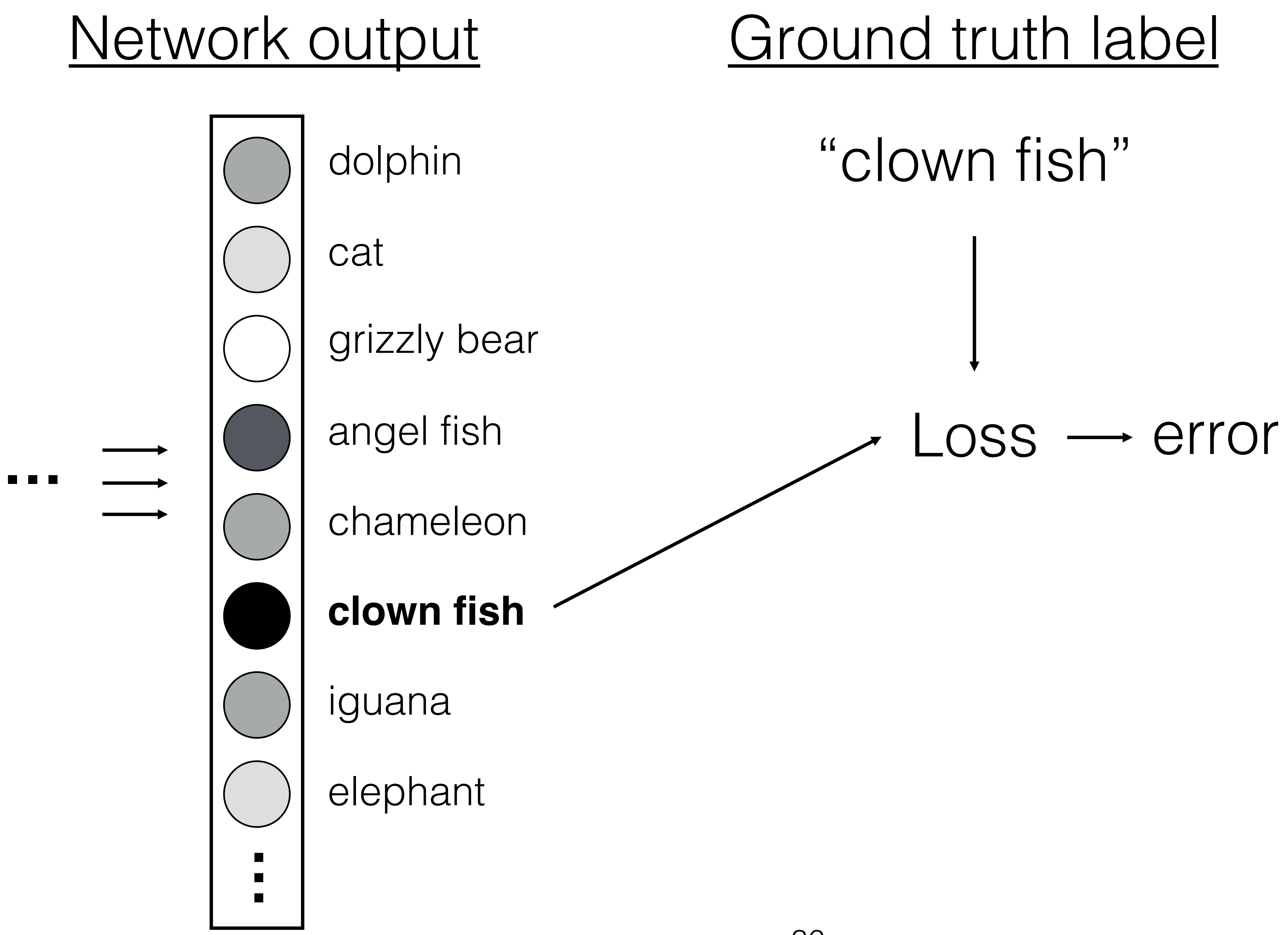
Learning with deep nets

Learned

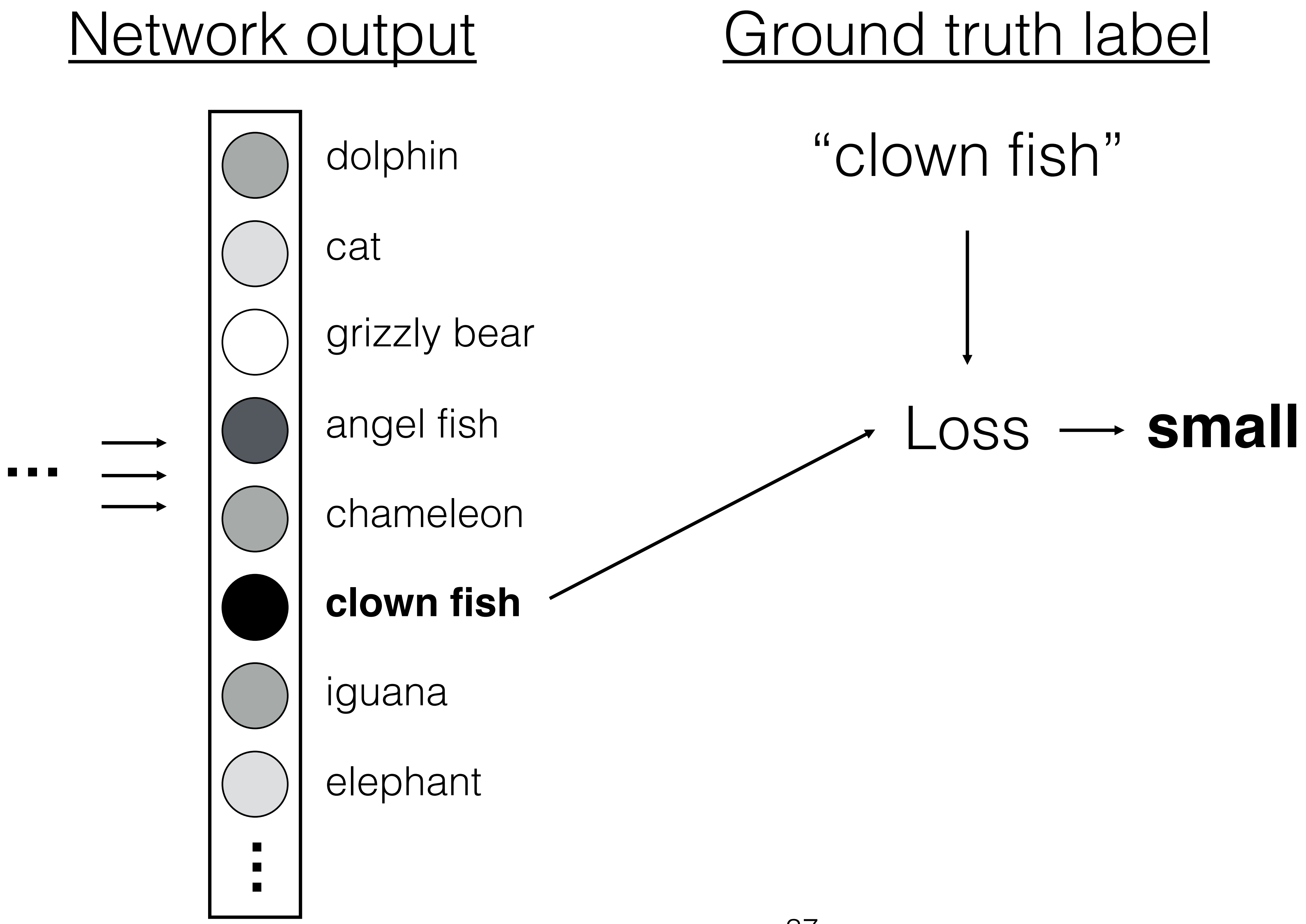


$$\operatorname*{argmin}_{\mathbf{w}} \quad L(w_1, \ldots, w_6)$$

Loss function



Loss function



Network output

Ground truth label

dolphin

cat

grizzly bear

angel fish

chameleon

clown fish

iguana

elephant

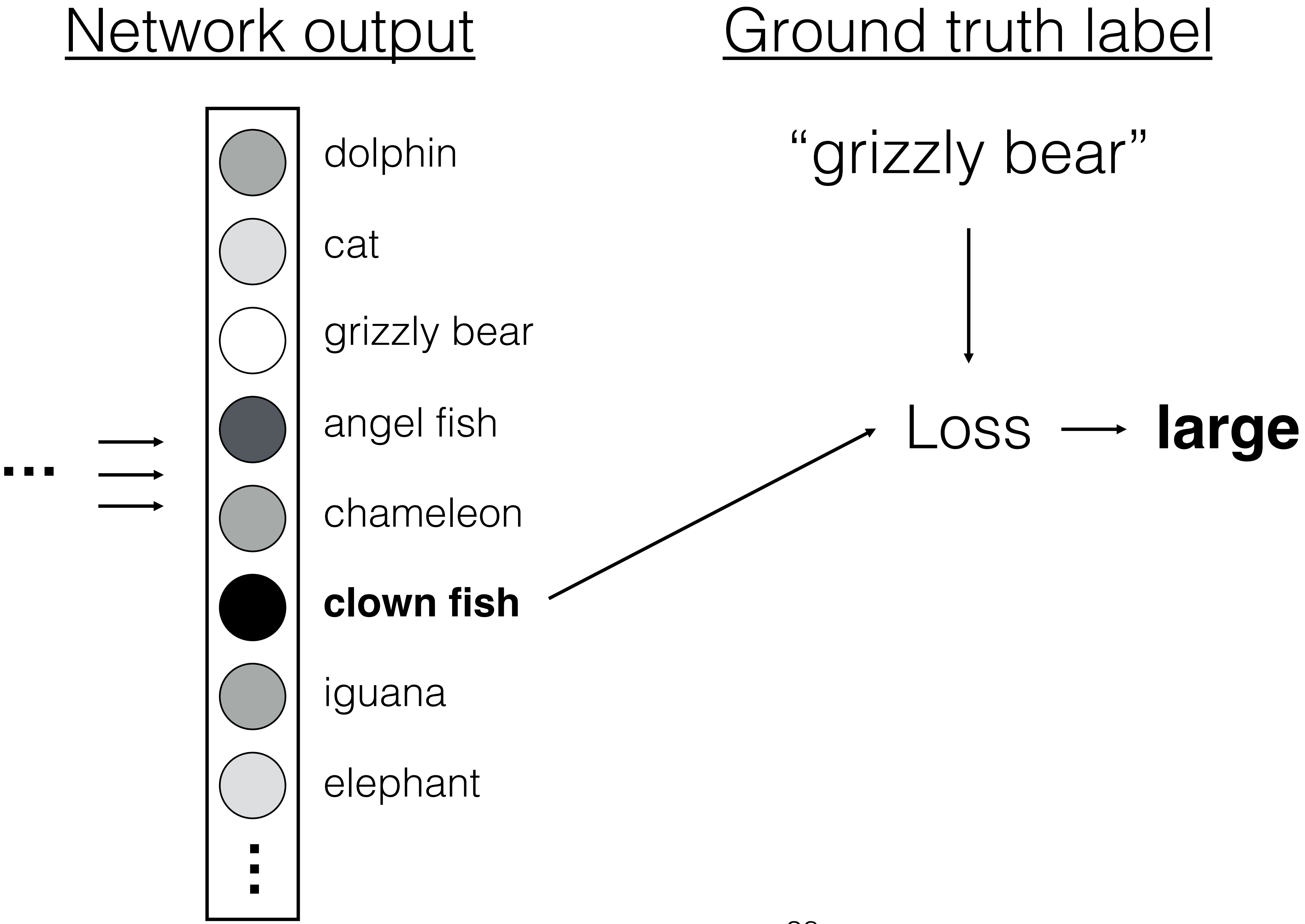
⋮

…

“clown fish”

Loss → **small**

Loss function



Network output

Ground truth label

dolphin

cat

grizzly bear

angel fish

chameleon

clown fish

iguana

elephant

⋮

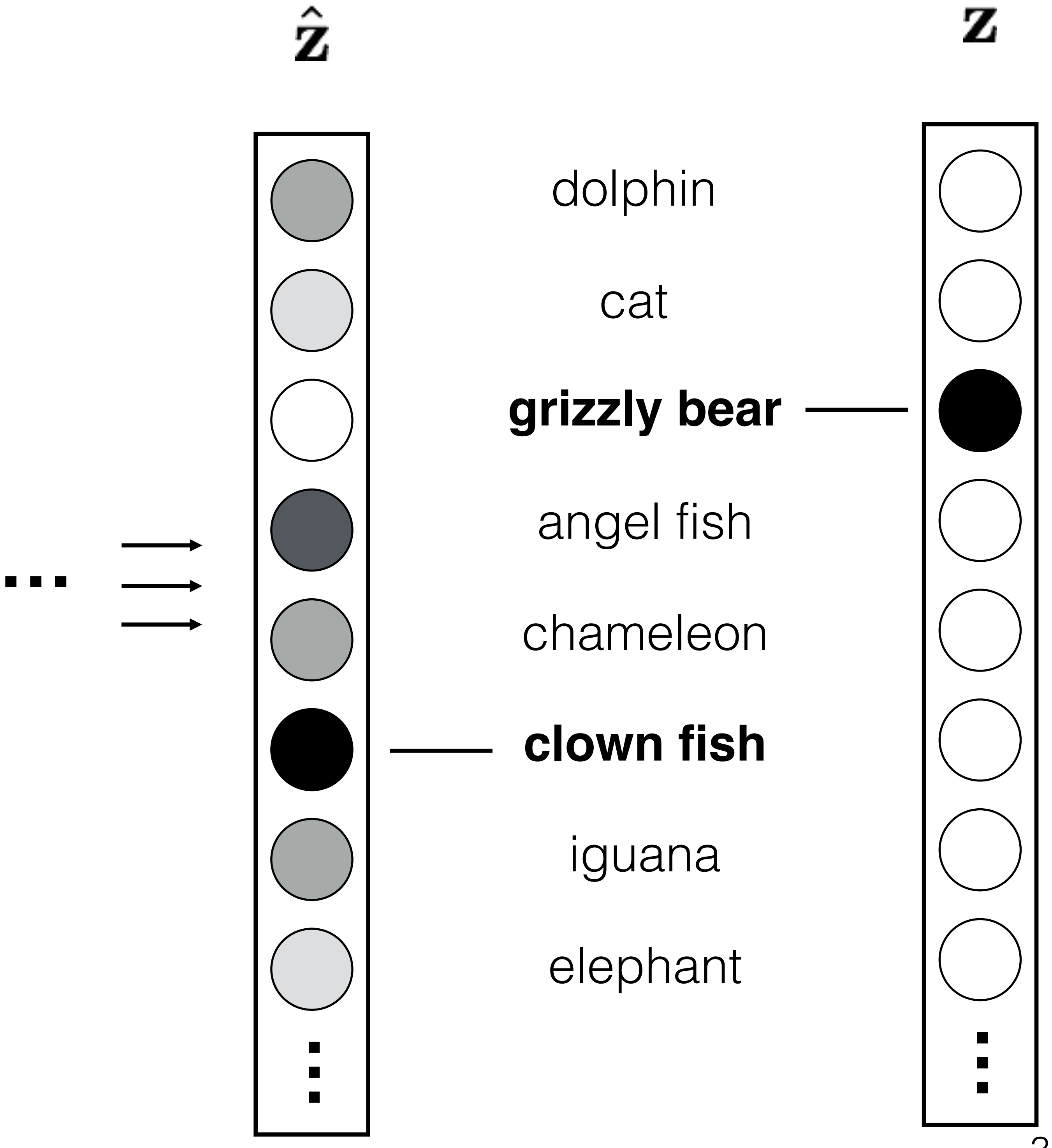
…

“grizzly bear”

Loss → **large**

Loss function for classification

Network output   Ground truth label



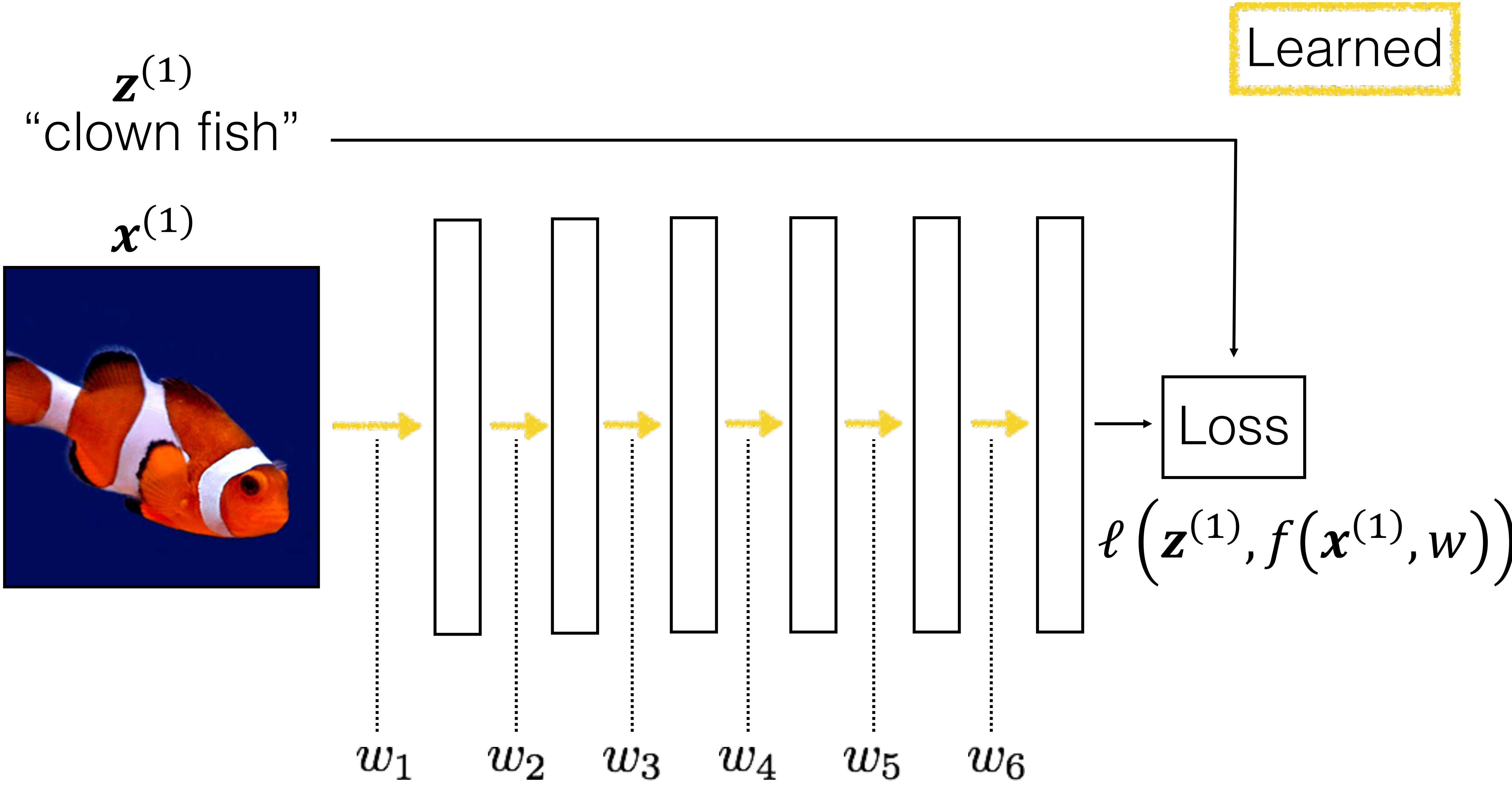
Probability of the observed data under the model

$$H(\hat{z}, z) = -\sum_{c} z_c \log \hat{z}_c$$

Cross-entropy loss

c is the c^{th} class in the output

Learning with deep nets



Learned

$\boldsymbol{z}^{(1)}$

“clown fish”

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

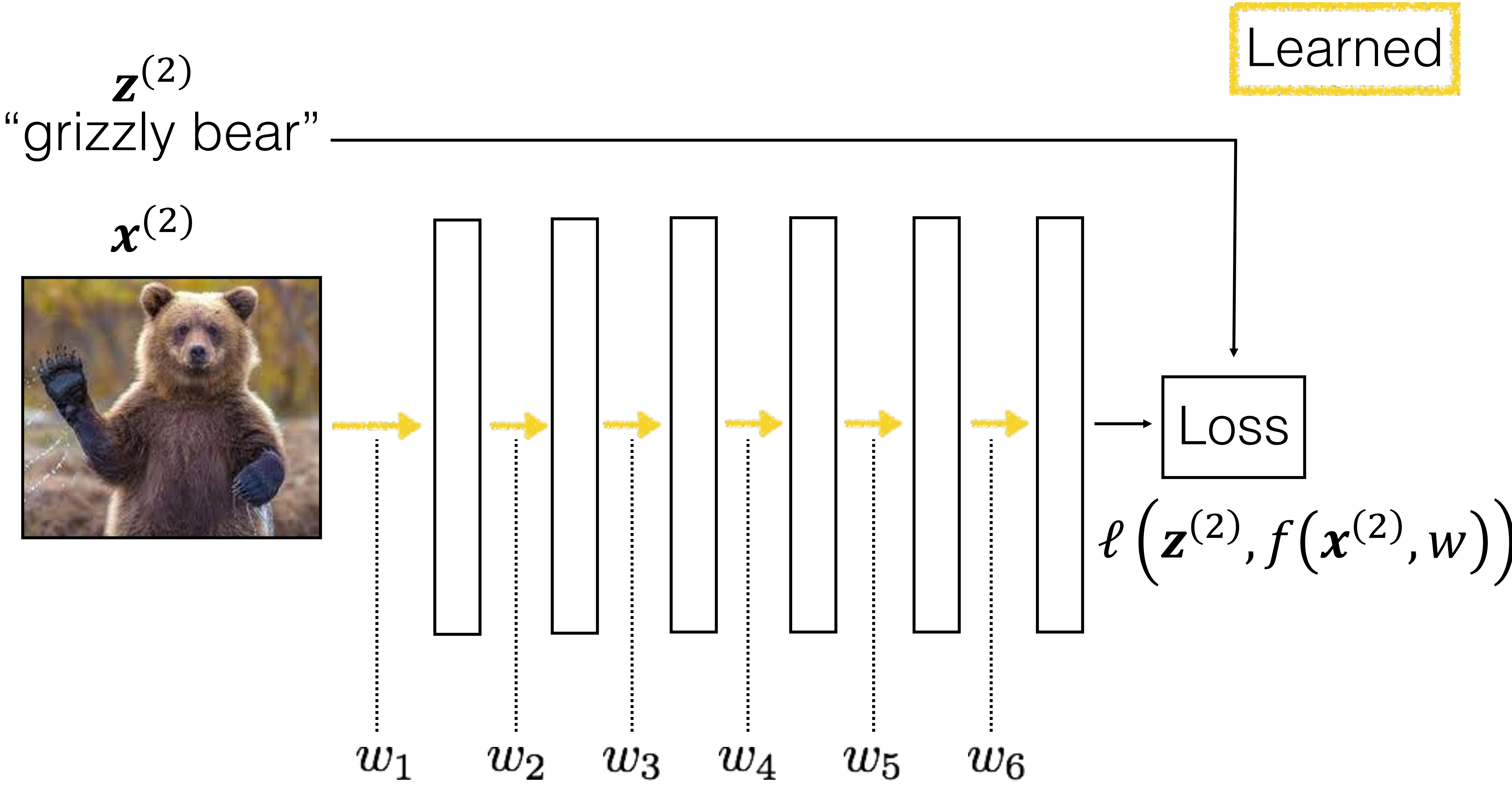
Loss

$\ell\left(\boldsymbol{z}^{(1)}, f\left(\boldsymbol{x}^{(1)}, w\right)\right)$

w_1 w_2 w_3 w_4 w_5 w_6

$\boldsymbol{x}^{(1)}, \boldsymbol{z}^{(1)}$ is the input and label of the 1st training image

Learning with deep nets



Learned

$\boldsymbol{z}^{(2)}$

“grizzly bear”

$\boldsymbol{x}^{(2)}$

Loss

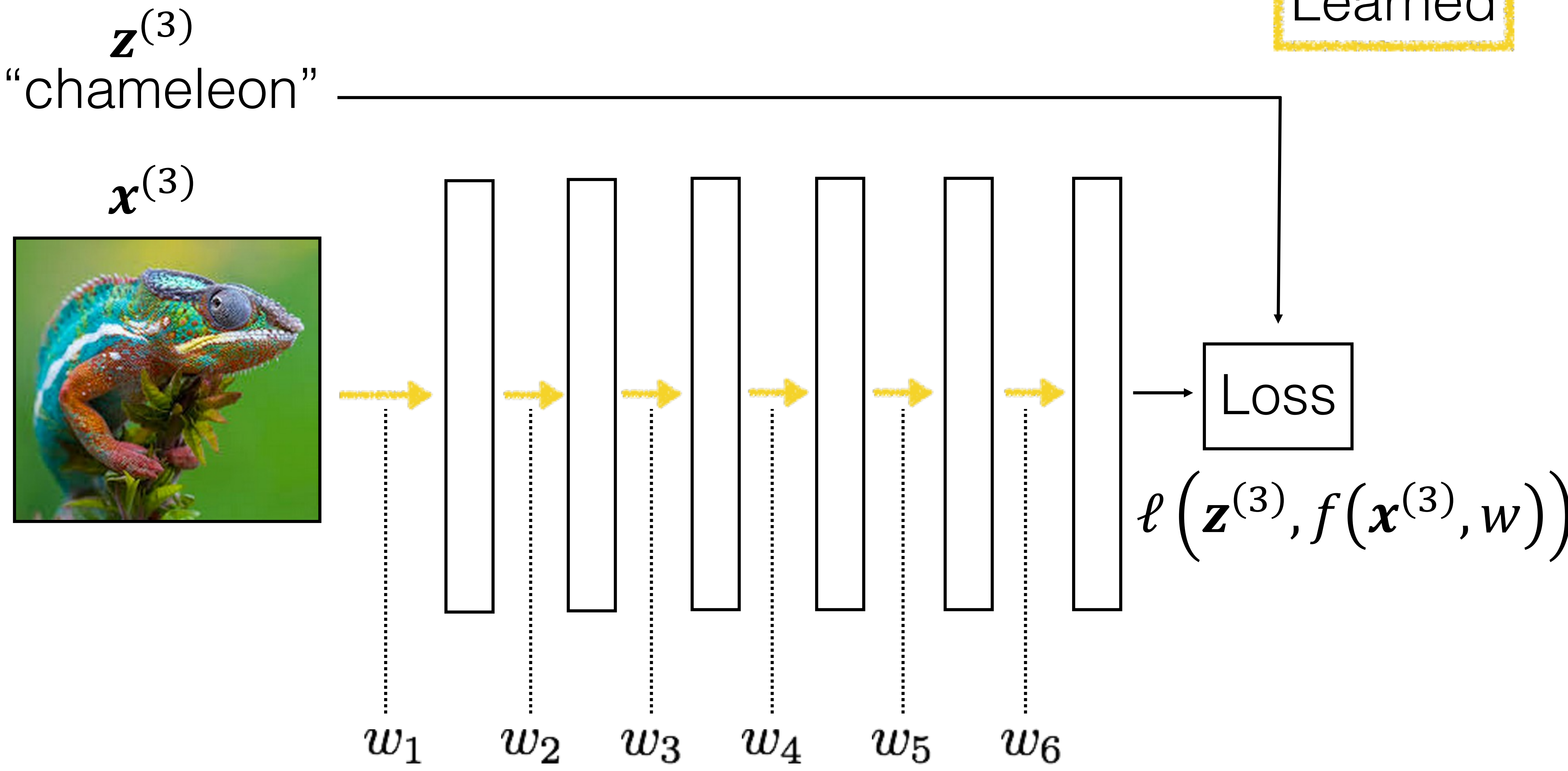
$\ell\left(\boldsymbol{z}^{(2)}, f\left(\boldsymbol{x}^{(2)}, w\right)\right)$

w_1 w_2 w_3 w_4 w_5 w_6

$\boldsymbol{x}^{(2)}, \boldsymbol{z}^{(2)}$ is the input and label of the 2nd training image

Learning with deep nets

Learned



$\boldsymbol{z}^{(3)}$

“chameleon”

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

Loss

$\ell\left(\boldsymbol{z}^{(3)}, f\left(\boldsymbol{x}^{(3)}, w\right)\right)$

w_1 w_2 w_3 w_4 w_5 w_6

$$argmin_w \sum_i \ell\left(z^{(i)}, f\left(x^{(i)}, w\right)\right)$$

Gradient descent

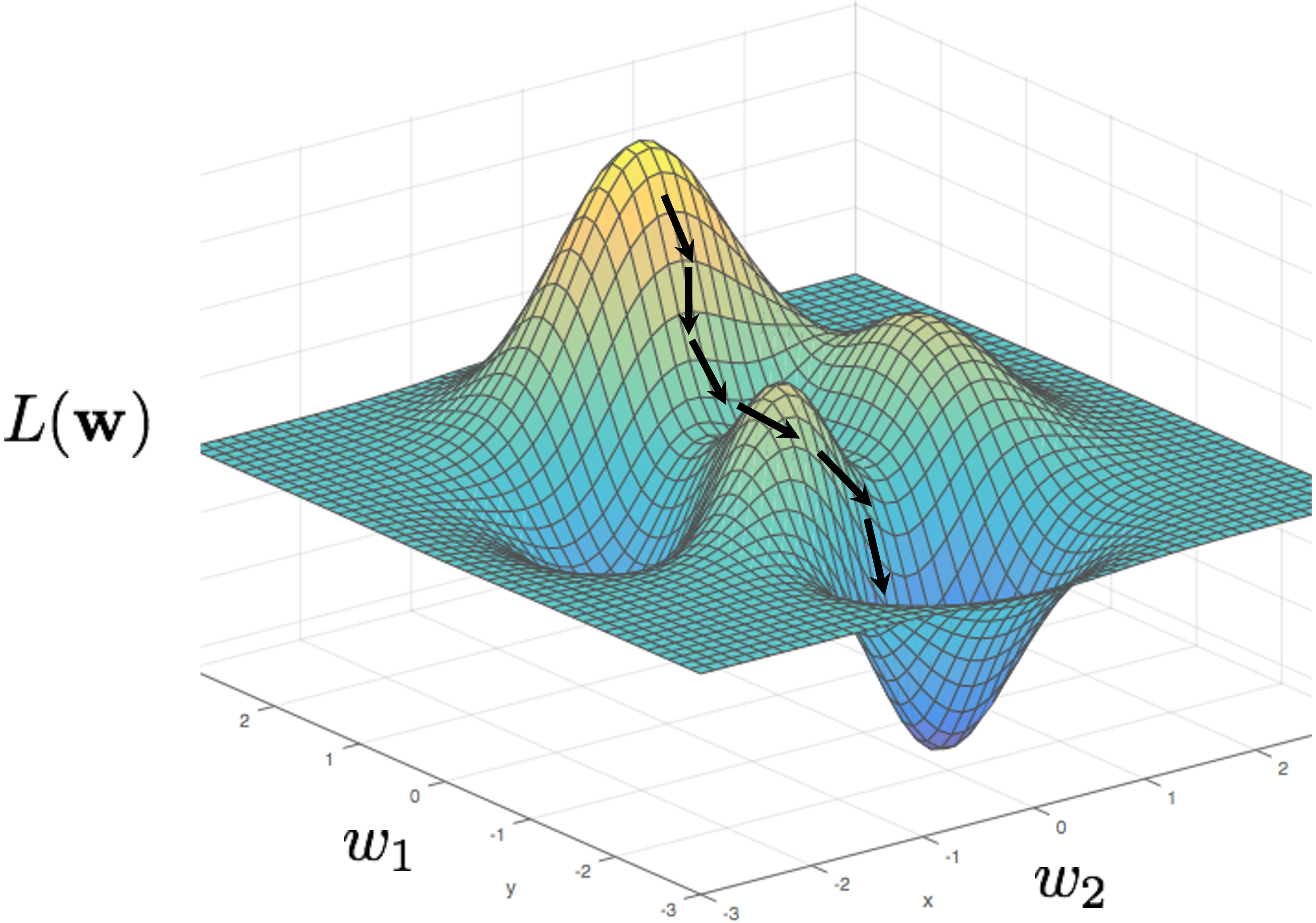
$$\boxed{argmin_w \sum_i \ell\left(z^{(i)}, f(x^{(i)}, w)\right) = argmin_w L(w)}$$

One iteration of gradient descent:

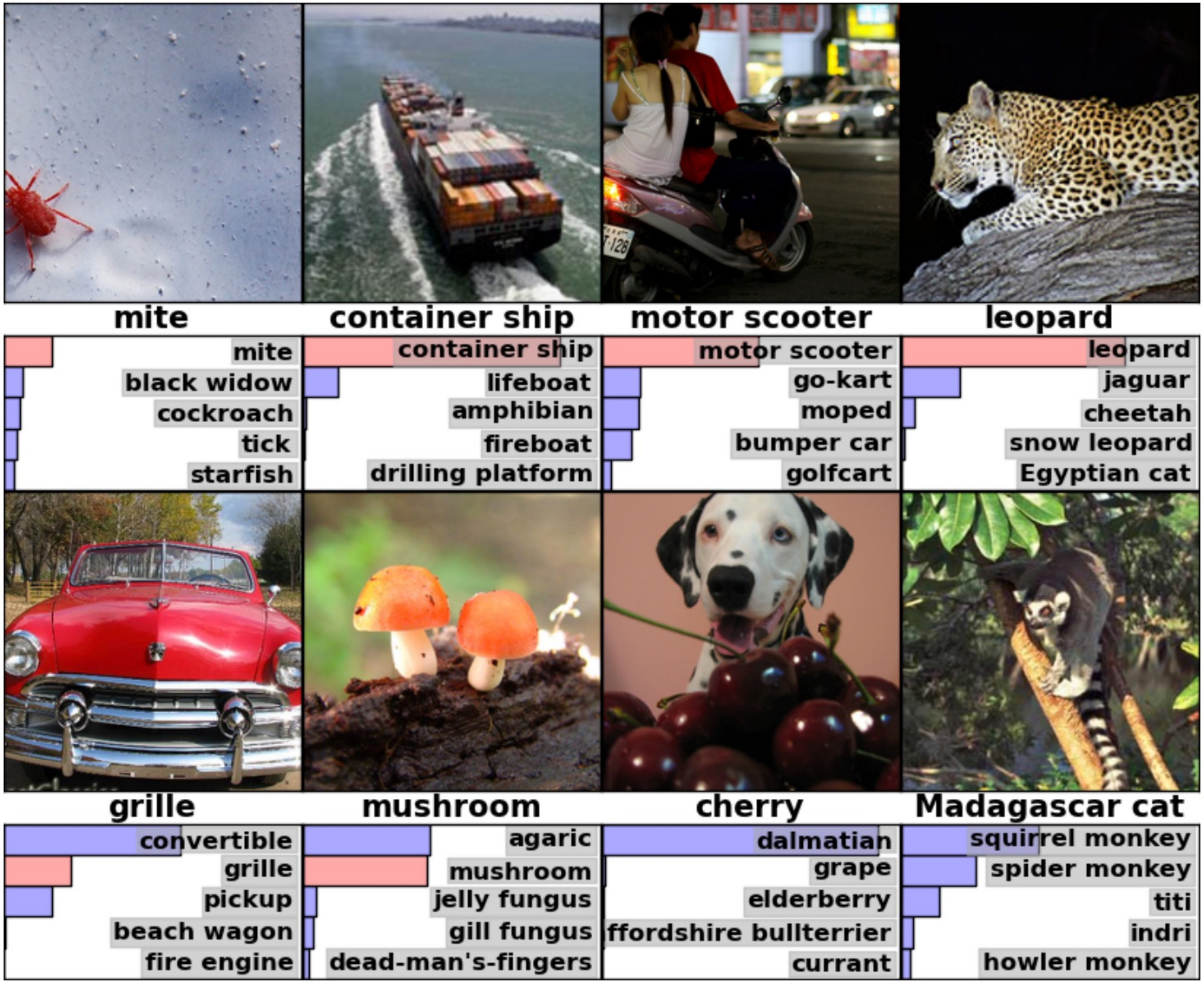
$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta_t \frac{\partial L(\mathbf{w}^{\mathbf{t}})}{\partial \mathbf{w}}$$

learning rate

Gradient descent

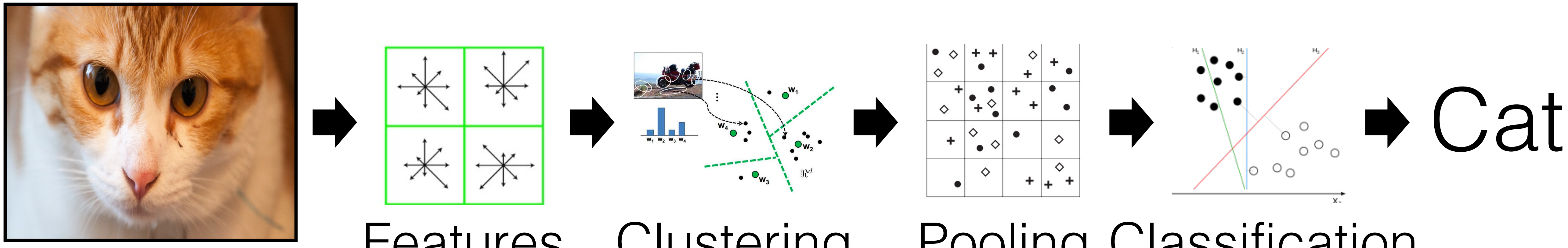


$p(c|\mathbf{x})$



[Krizhevsky et al. NIPS 2012]

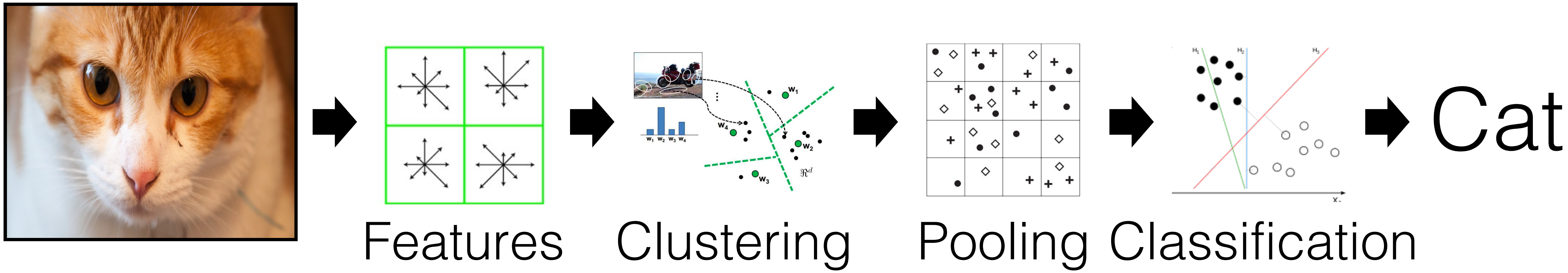
Computer Vision before 2012



Features Clustering Pooling Classification

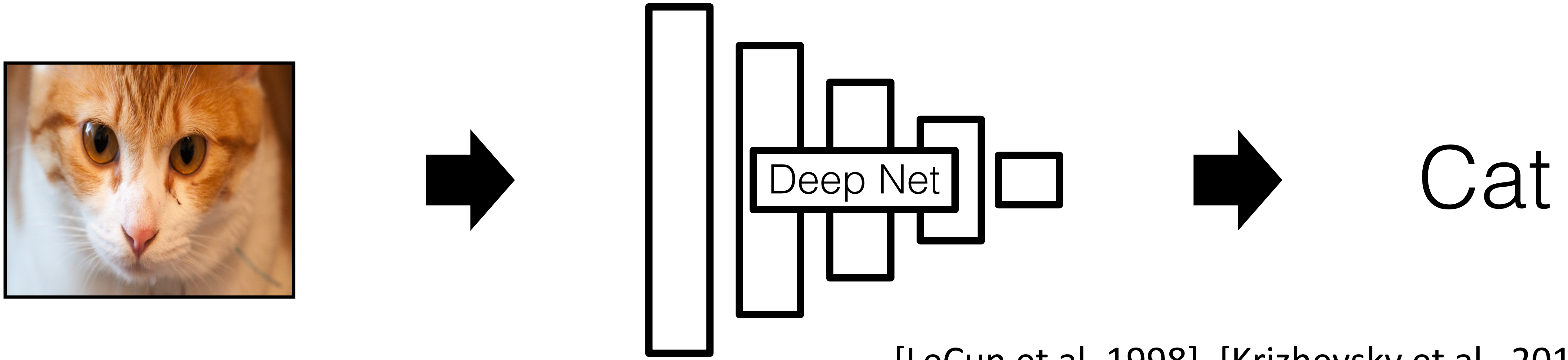
Cat

Computer Vision Now



Features Clustering Pooling Classification

Cat



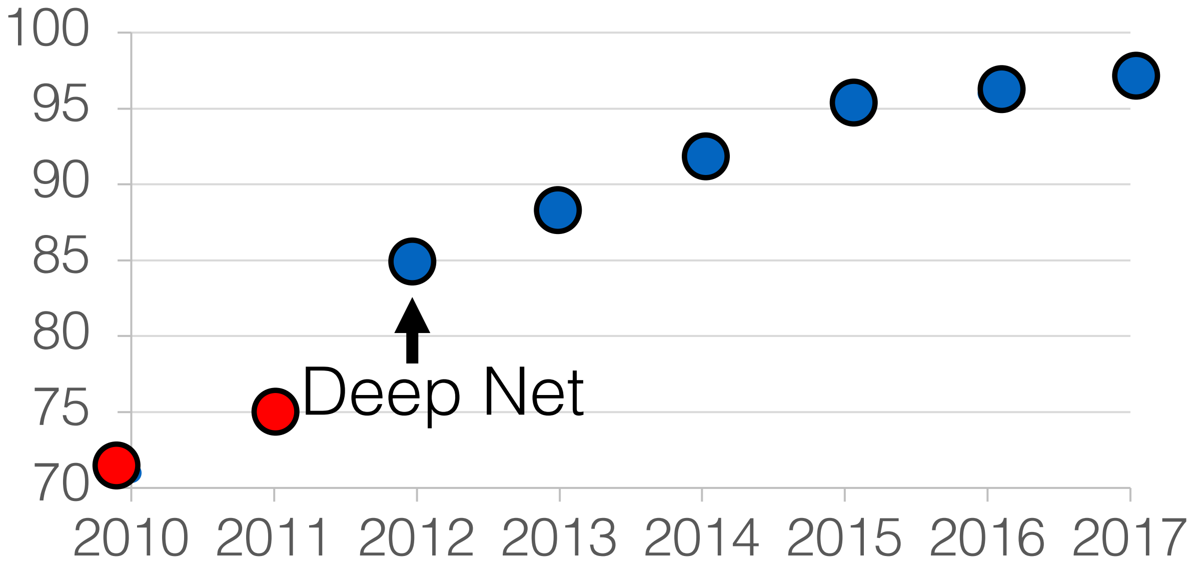
Deep Net

Cat

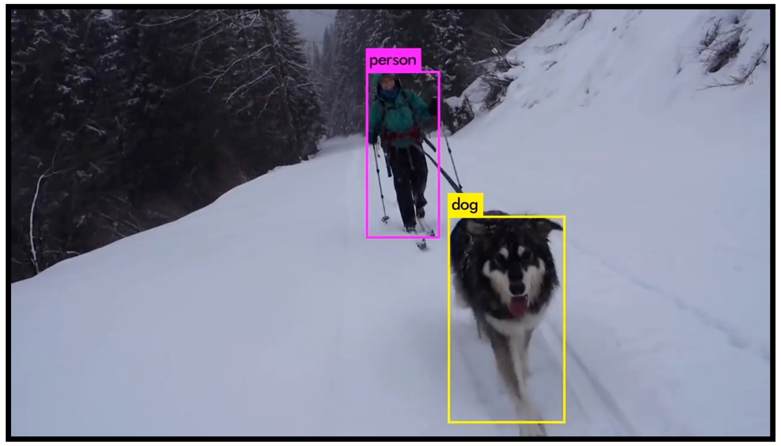
[LeCun et al, 1998], [Krizhevsky et al, 2012]

Deep Learning for Computer Vision

[Deng et al. 2009]

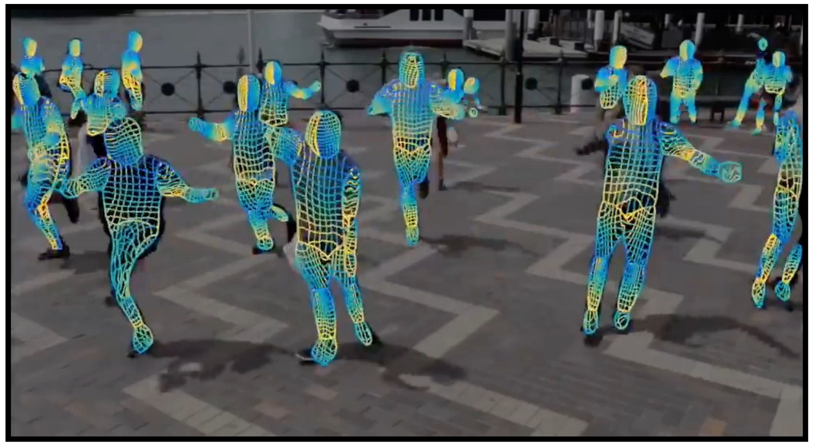


Top 5 **accuracy** on ImageNet benchmark



[Redmon et al., 2018]

Object detection



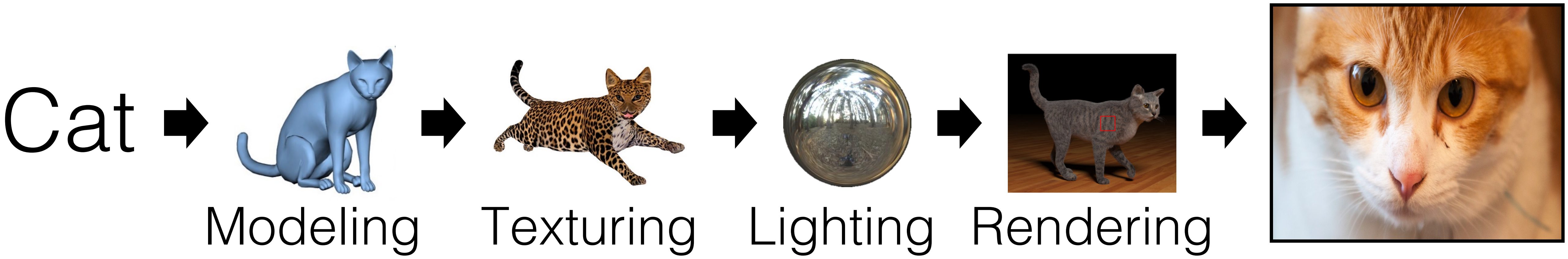
[Güler et al., 2018]

Human understanding

[Zhao et al., 2017]

Autonomous driving

Can Deep Learning **Help** Graphics?



Cat

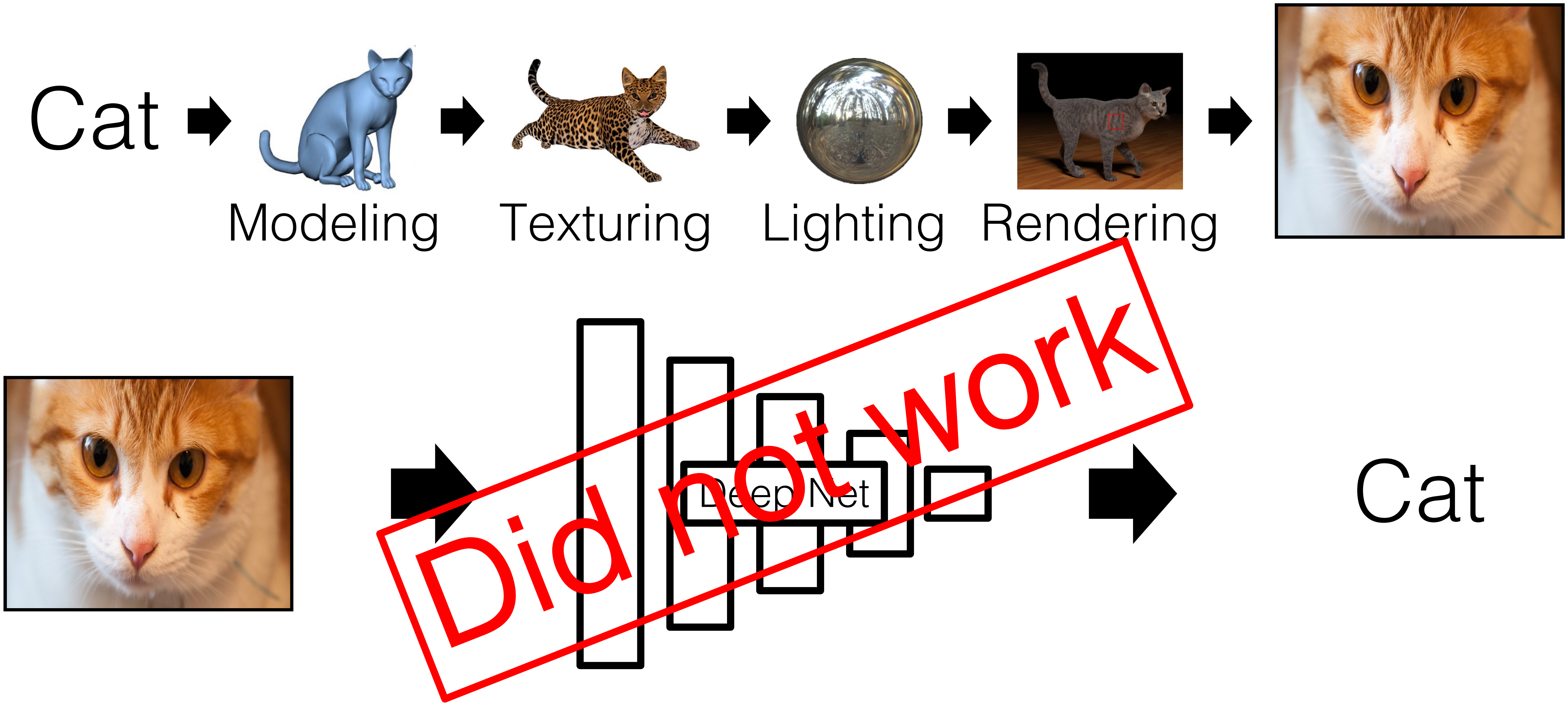
Modeling

Texturing

Lighting

Rendering

Can Deep Learning **Help** Graphics?

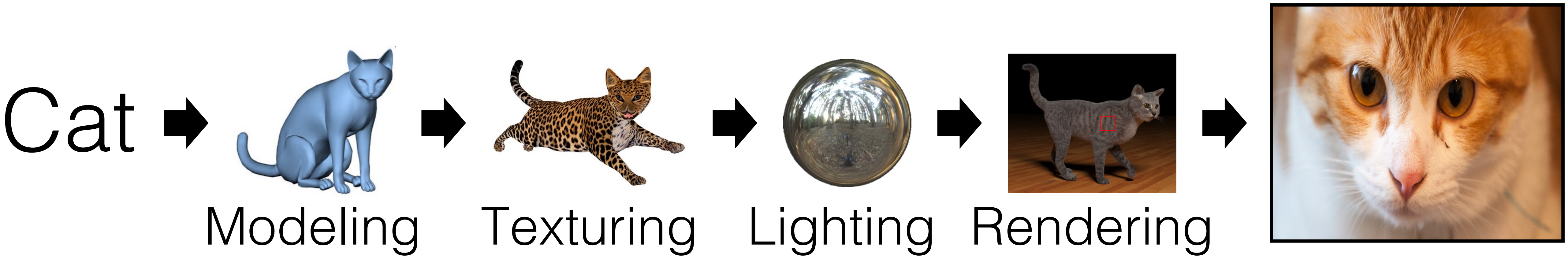


Cat → Modeling → Texturing → Lighting → Rendering →

→ Deep Net → Cat

Did not work

Generating images is hard!



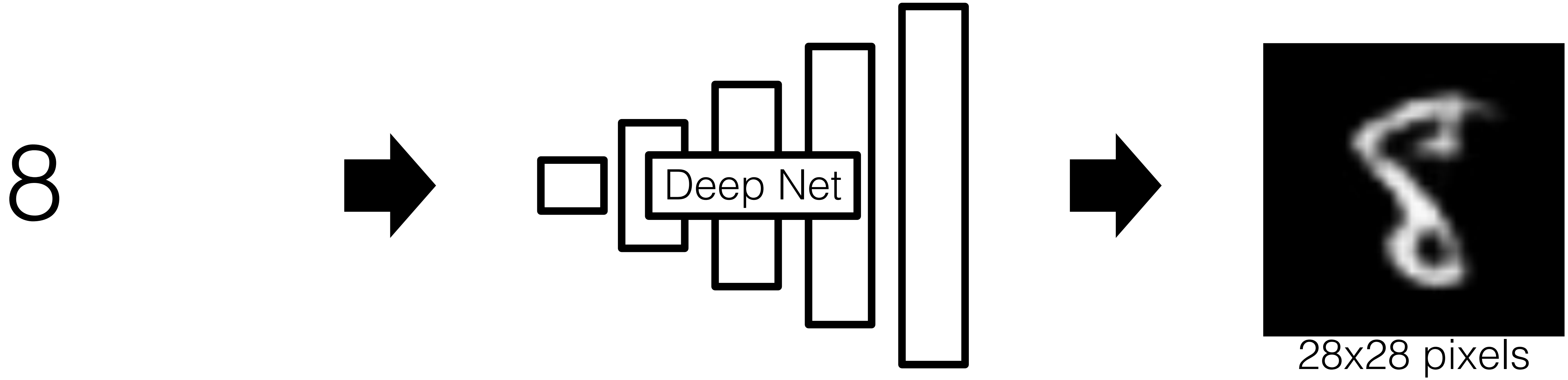
Cat

Modeling

Texturing

Lighting

Rendering

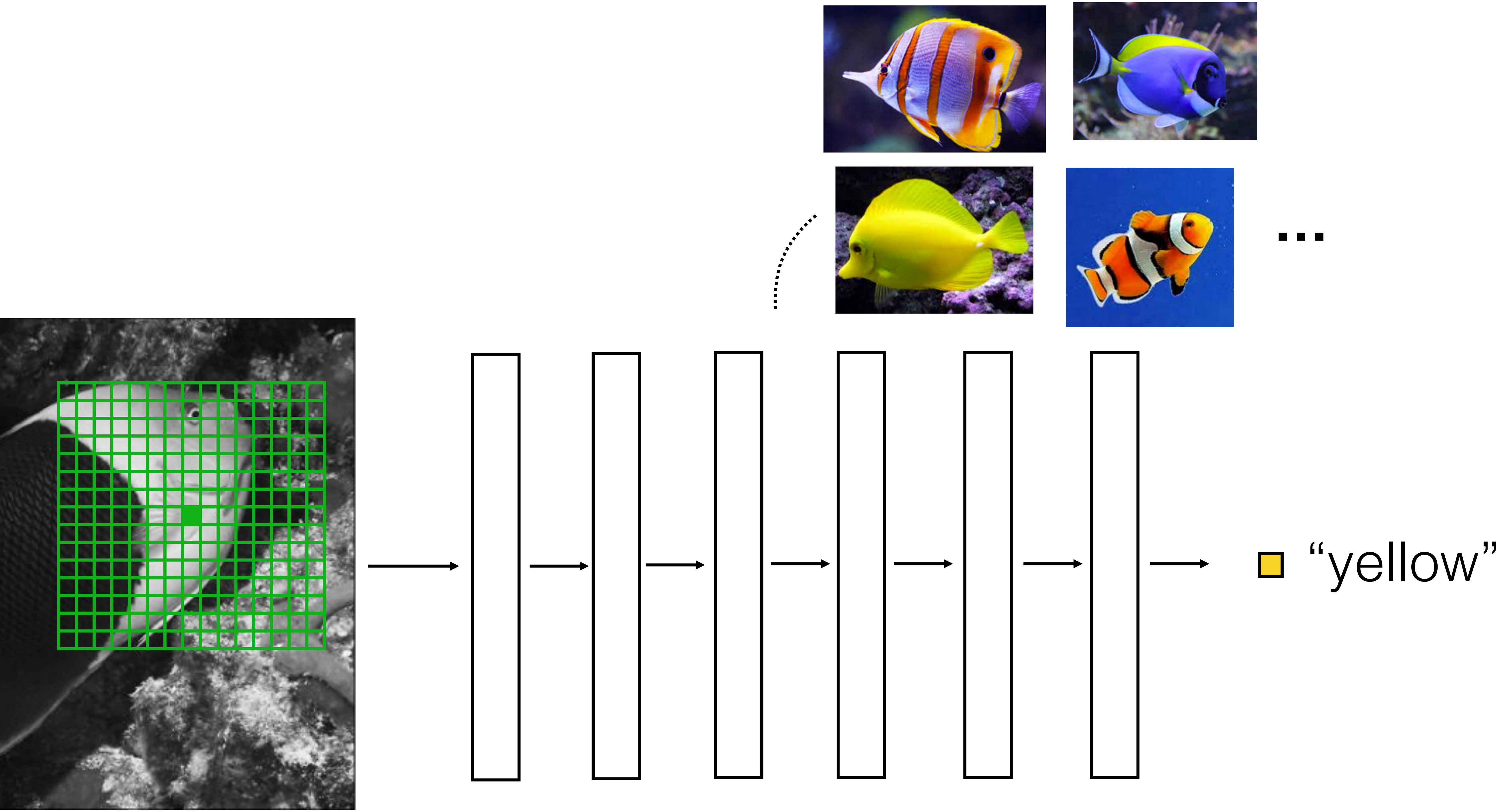


8

Deep Net

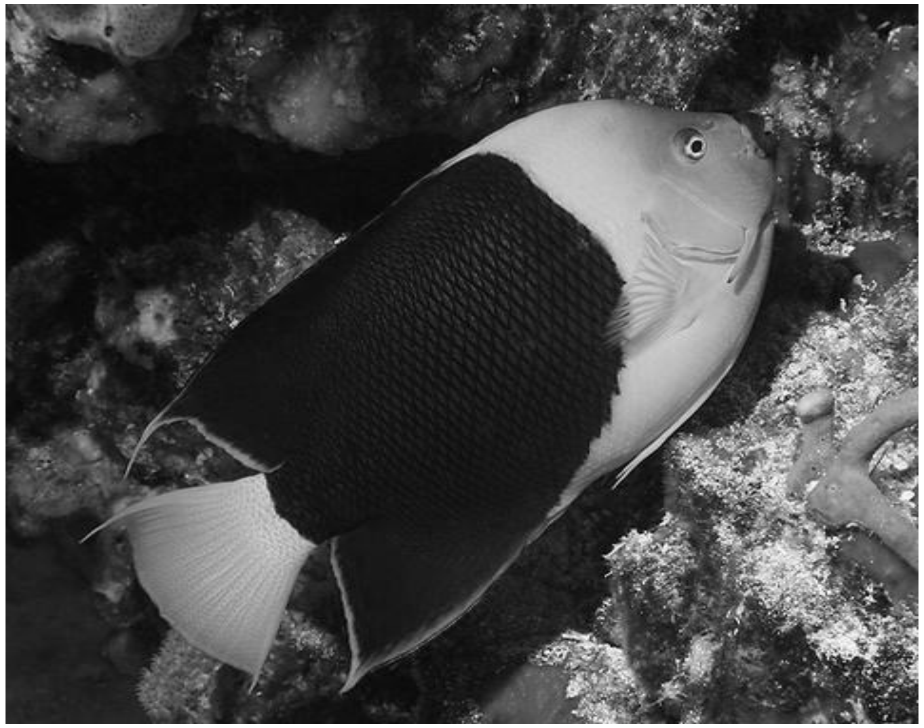
28x28 pixels

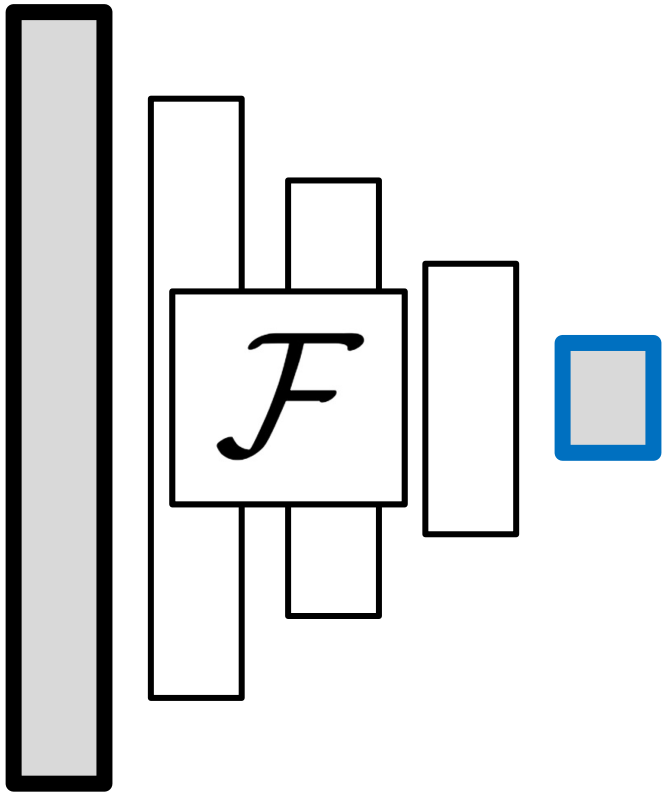
from Classification to Generation



redicting the color value of an output pixel given a patch

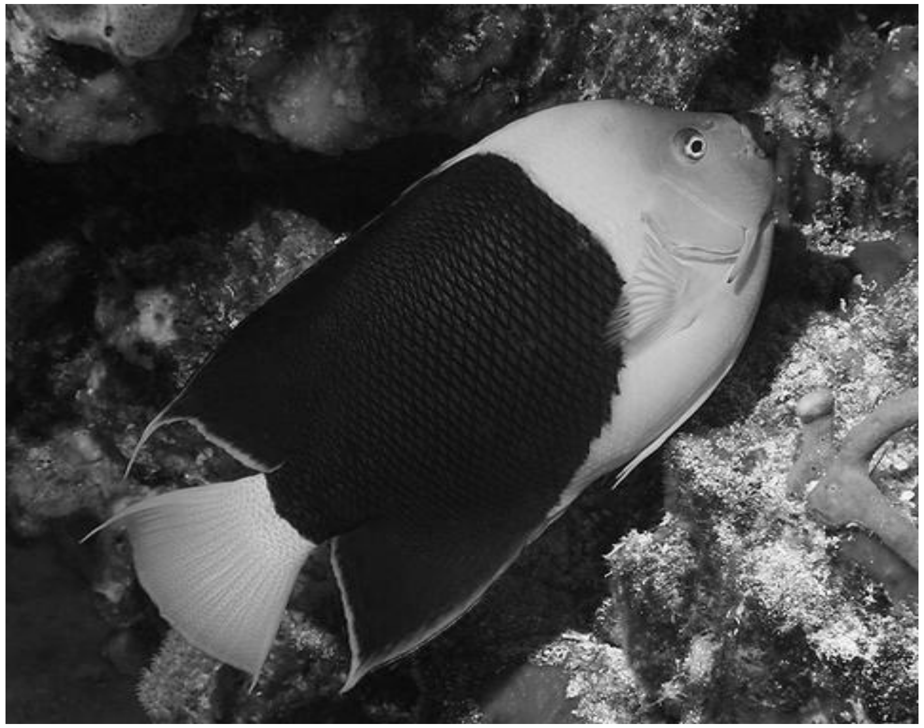
Discriminative Deep Networks

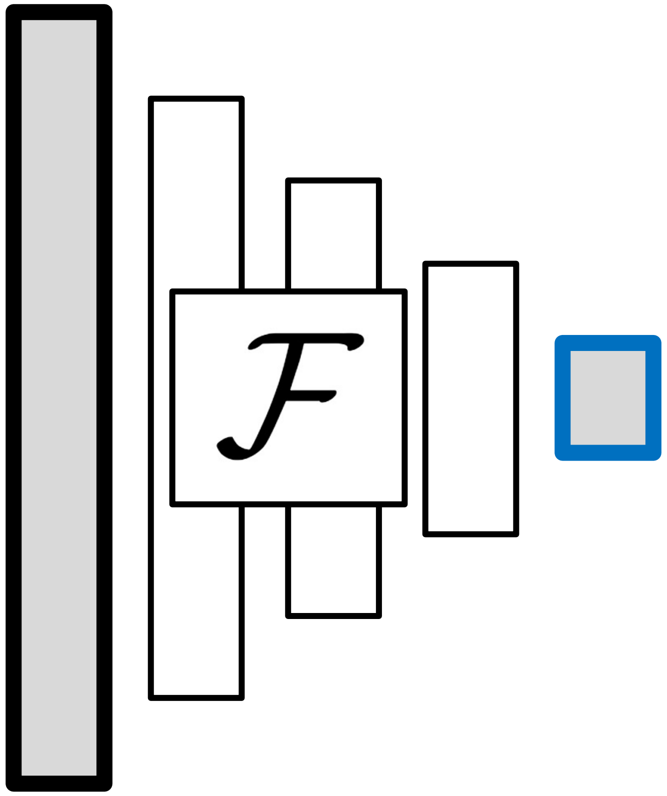




“Rockfish”

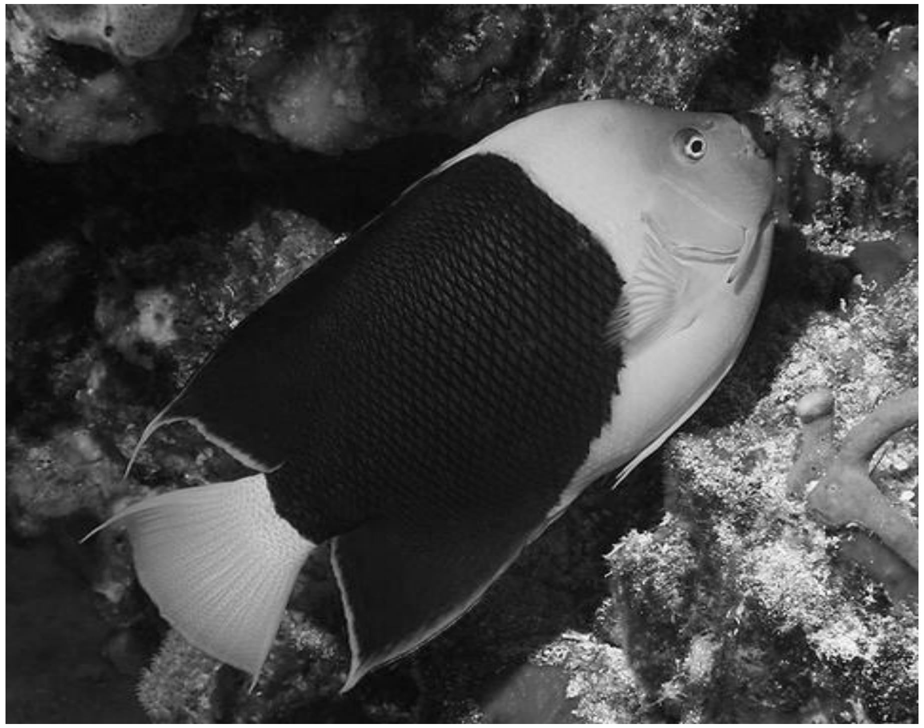
Discriminative Deep Networks

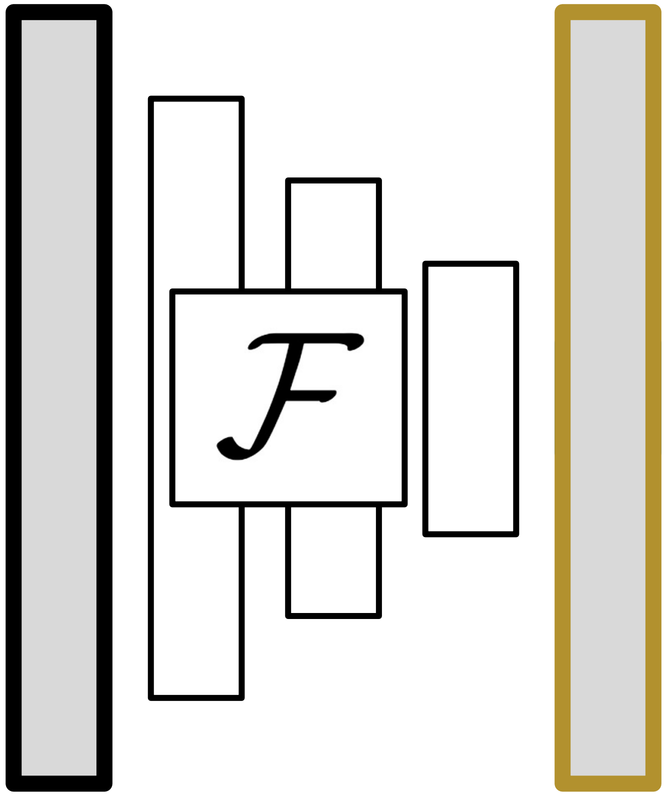




Raw, Unlabeled Pixels

Generative Deep Networks

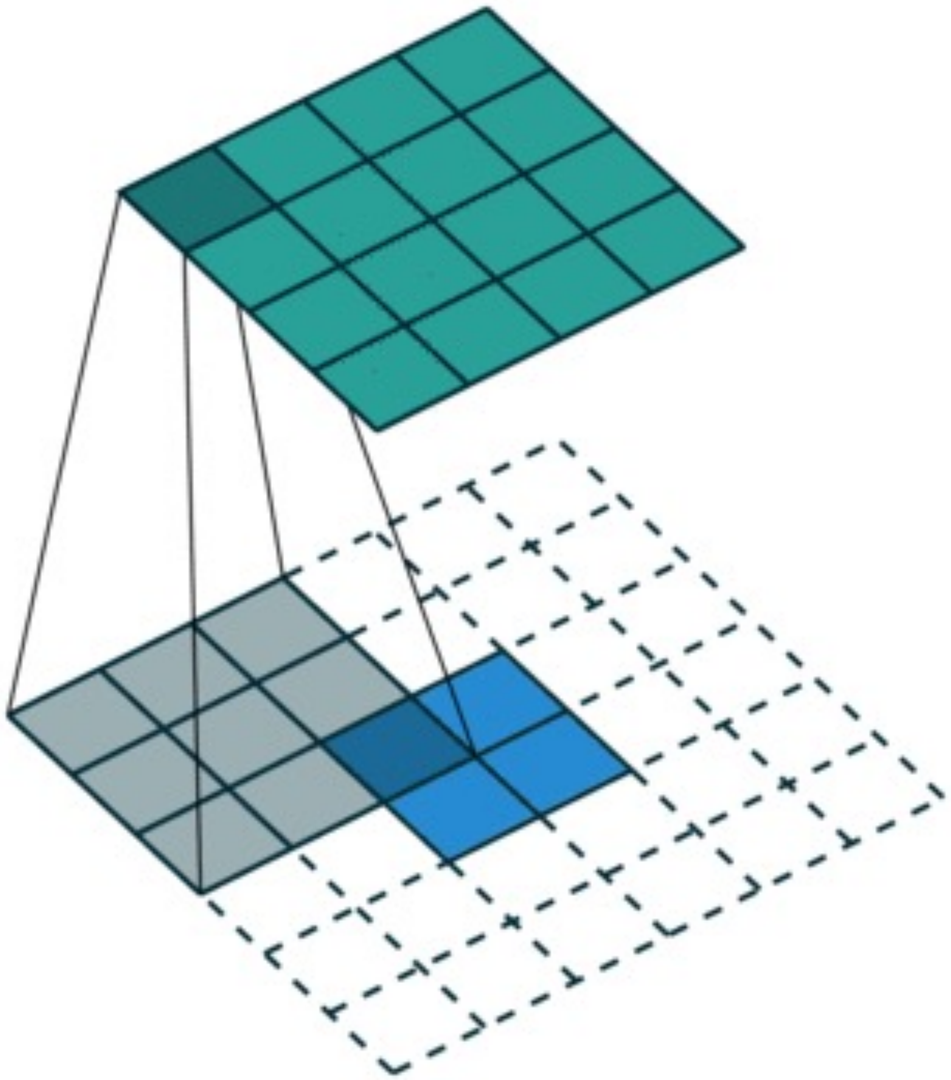




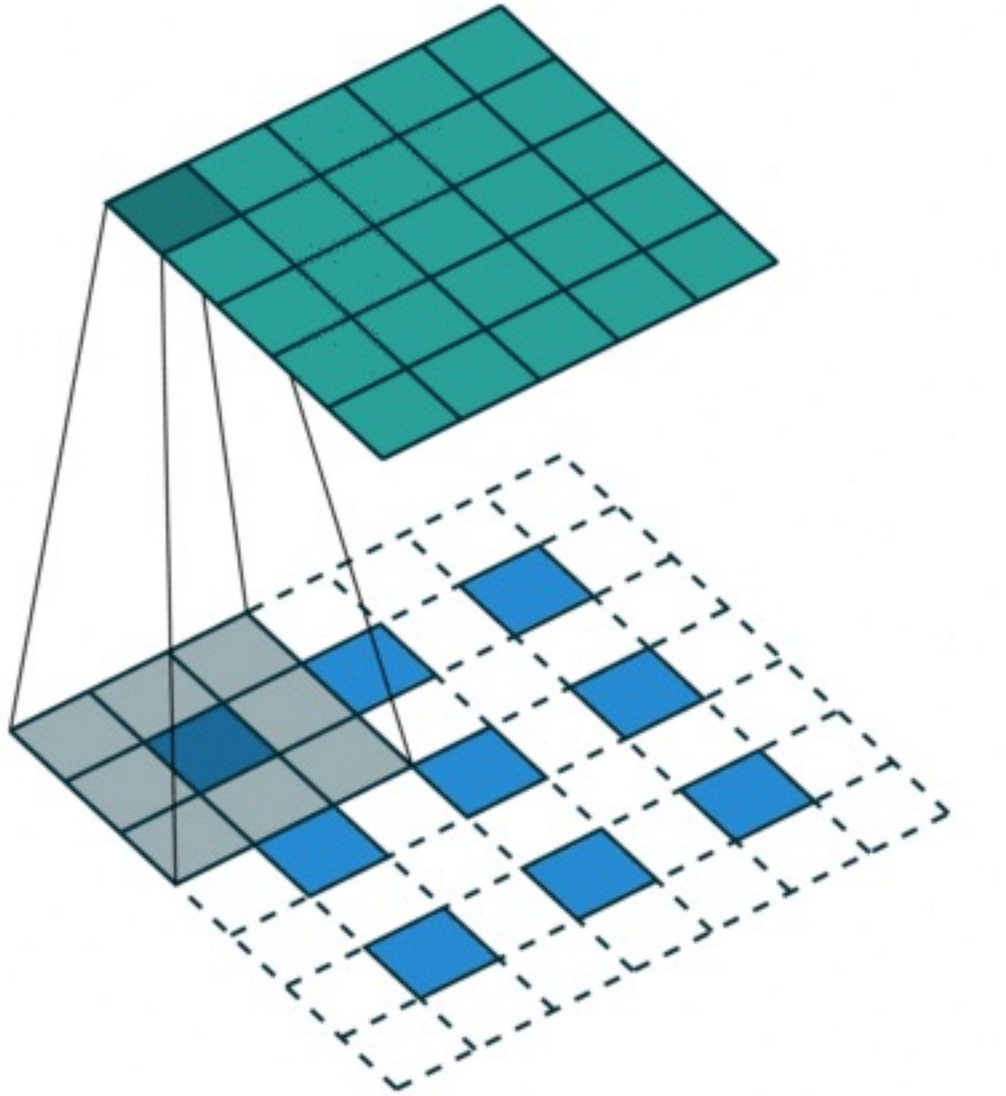
Raw, Unlabeled Pixels

Better Architectures

Fractionally-strided Convolution

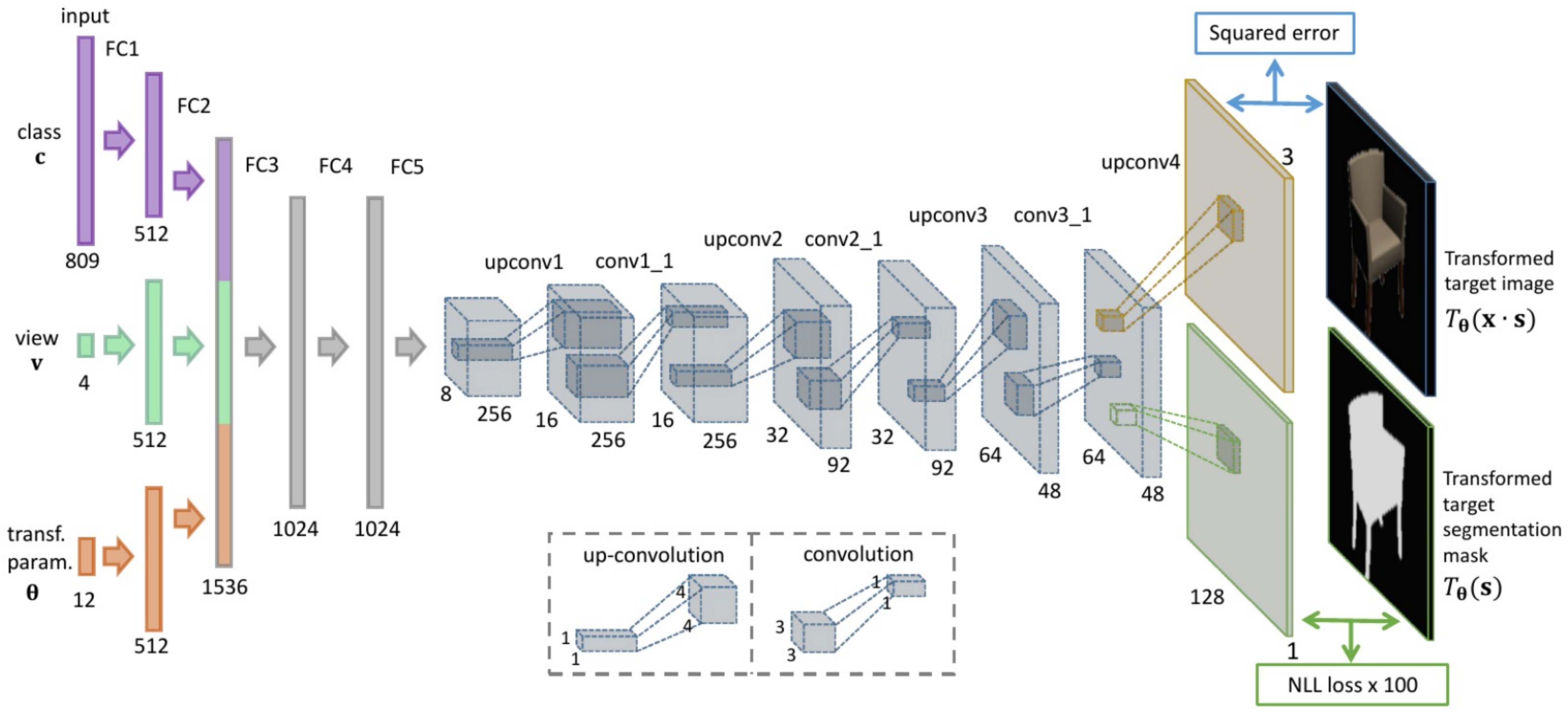


Regular conv



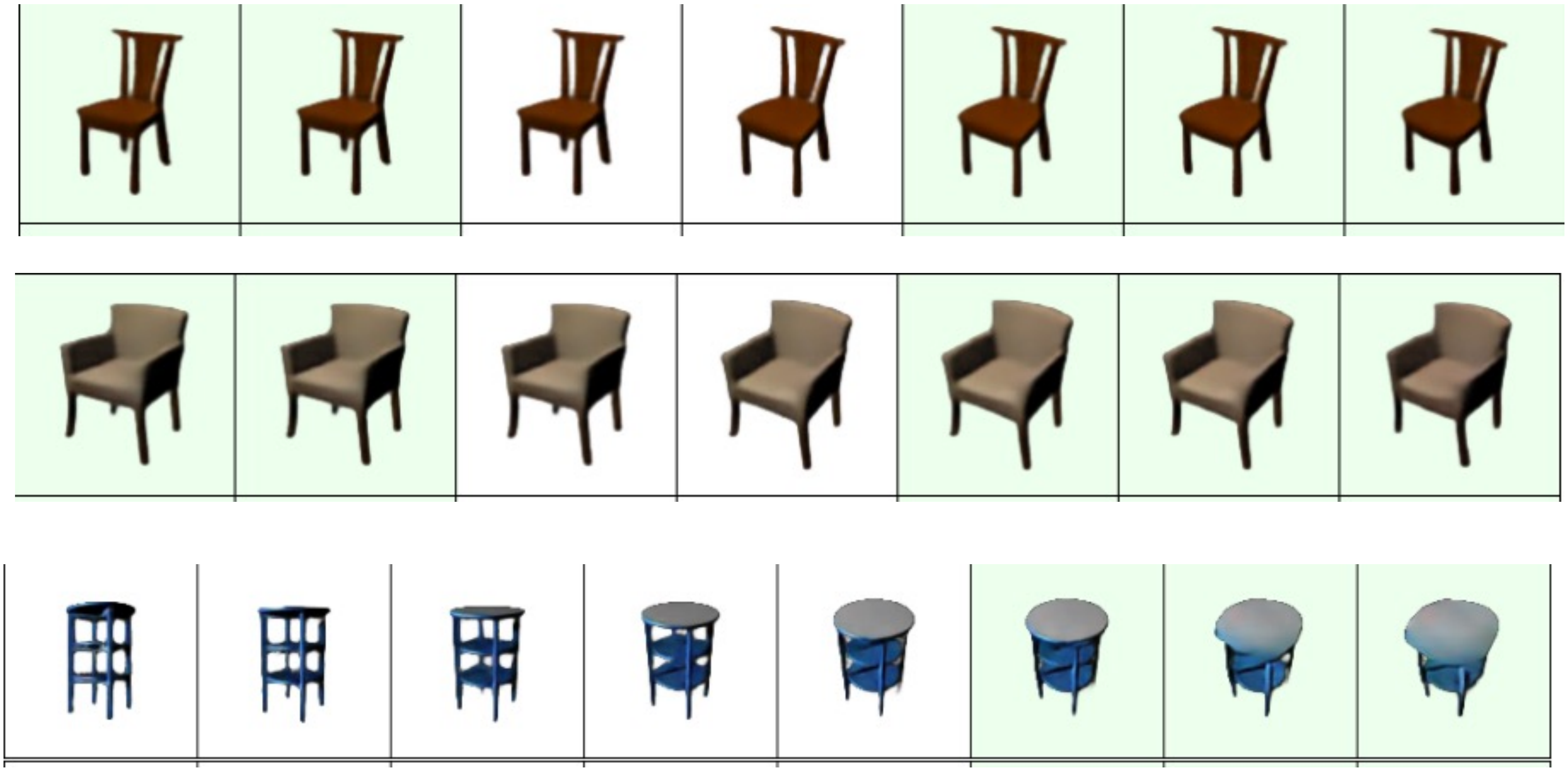
Fractiaionally-strided conv

Generating chairs conditional on chair ID, viewpoint, and transformation parameters



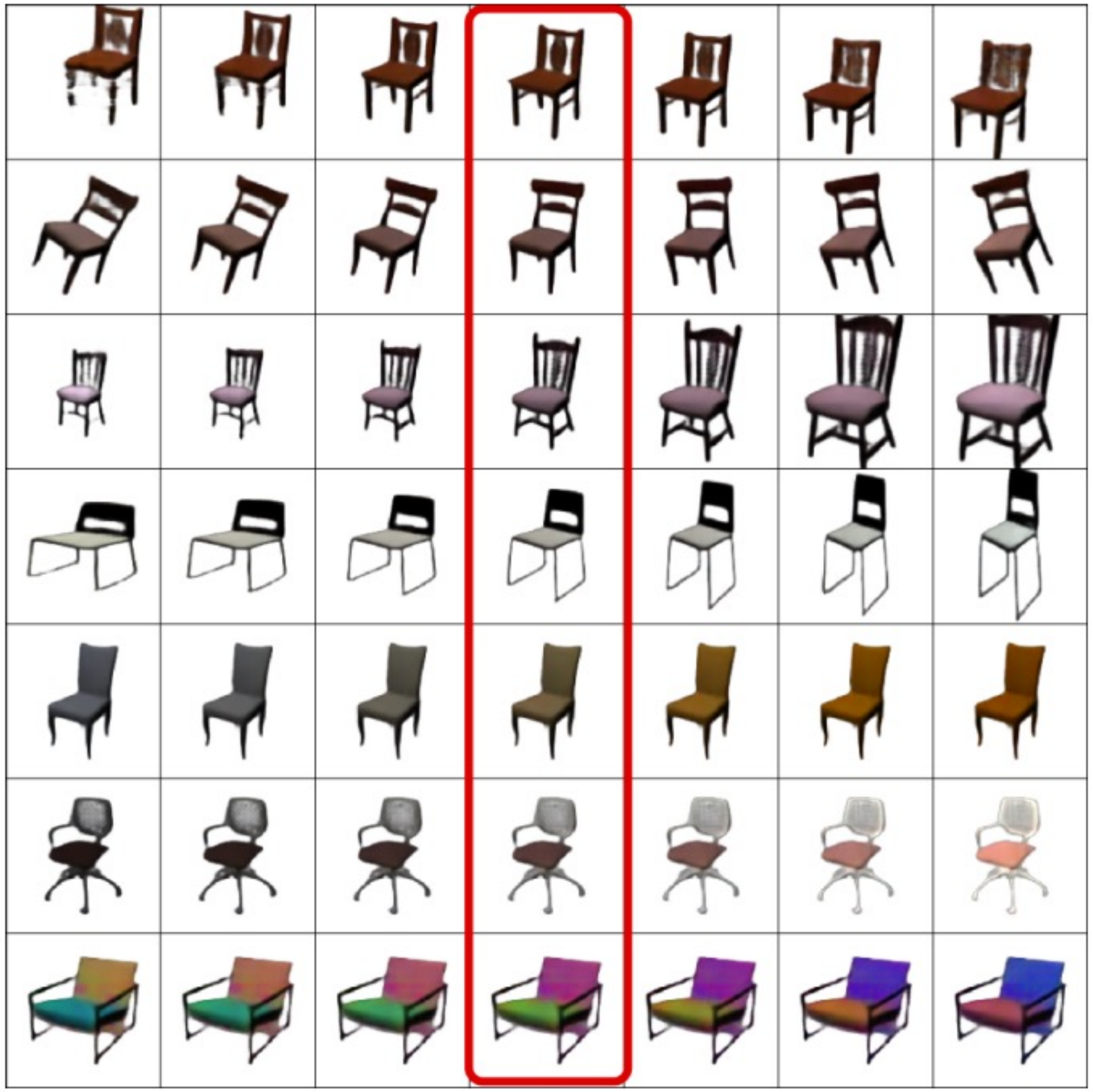
Dosovitskiy et al. Learning to Generate Chairs, Tables and Cars with Convolutional Networks
PAMI 2017 (CVPR 2015)

With Varying Viewpoints



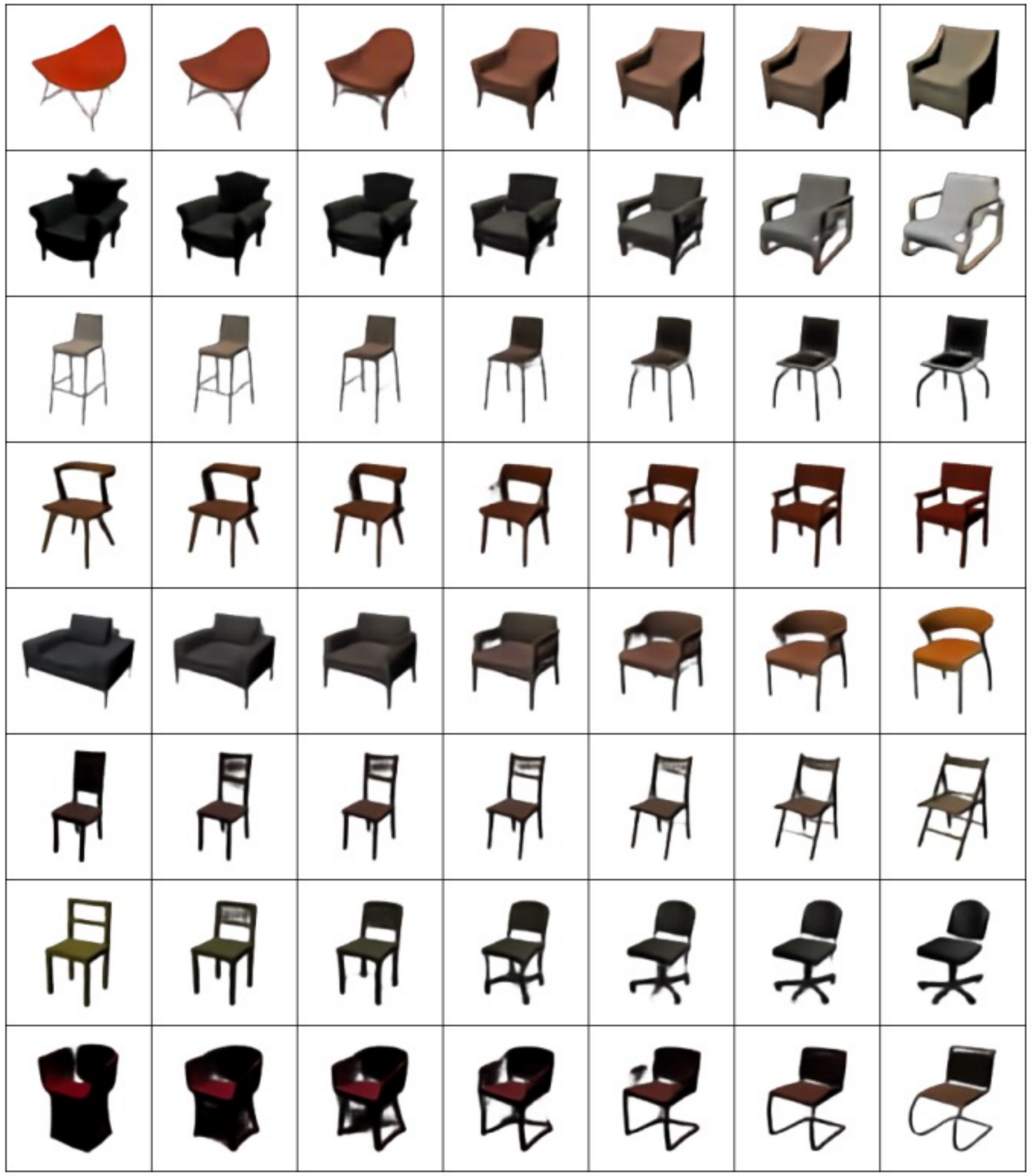
Dosovitskiy et al. Learning to Generate Chairs, Tables and Cars with Convolutional Networks
PAMI 2017 (CVPR 2015)

With Varying Transformation Parameters



Dosovitskiy et al. Learning to Generate Chairs, Tables and Cars with Convolutional Networks
PAMI 2017 (CVPR 2015)

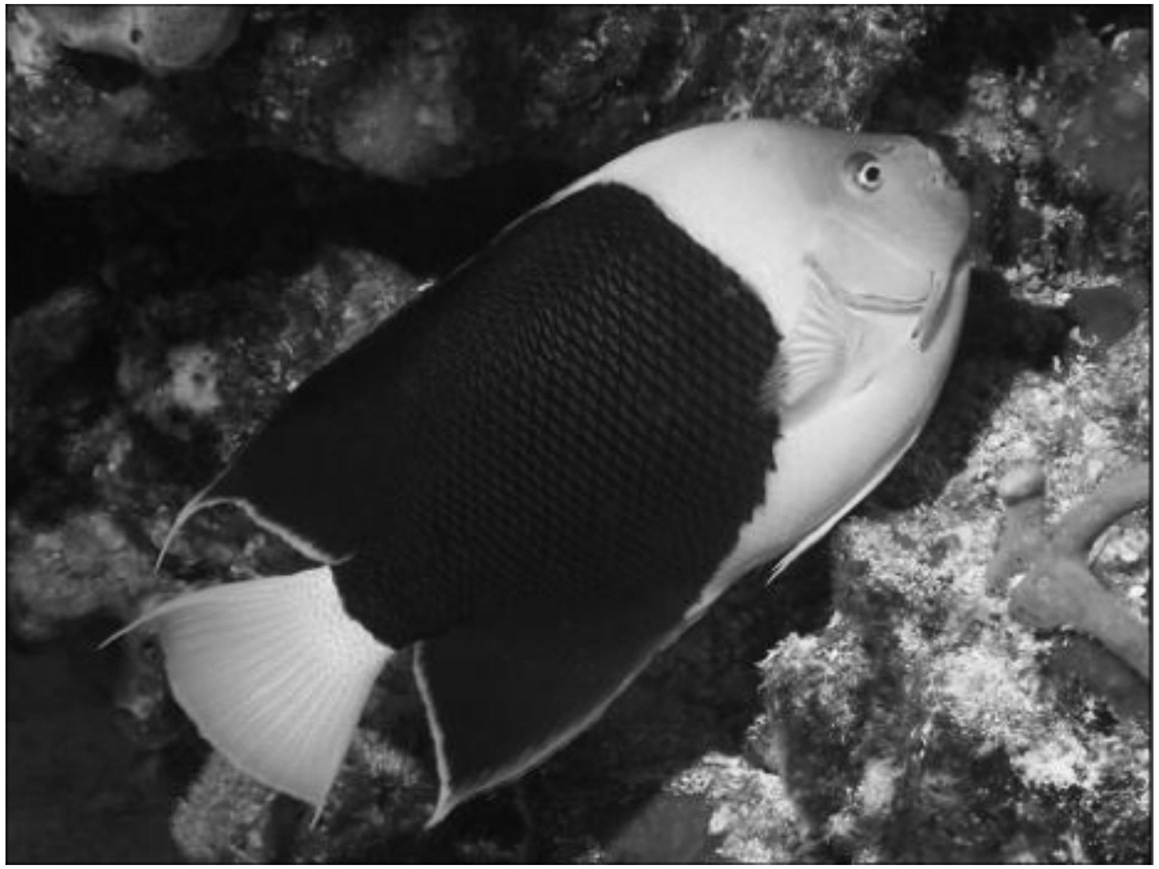
Interpolation between Two Chairs

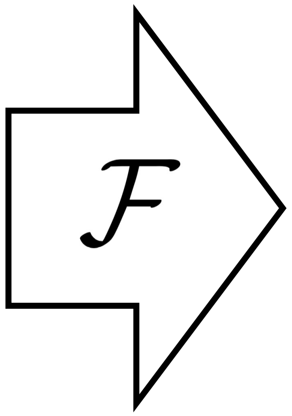


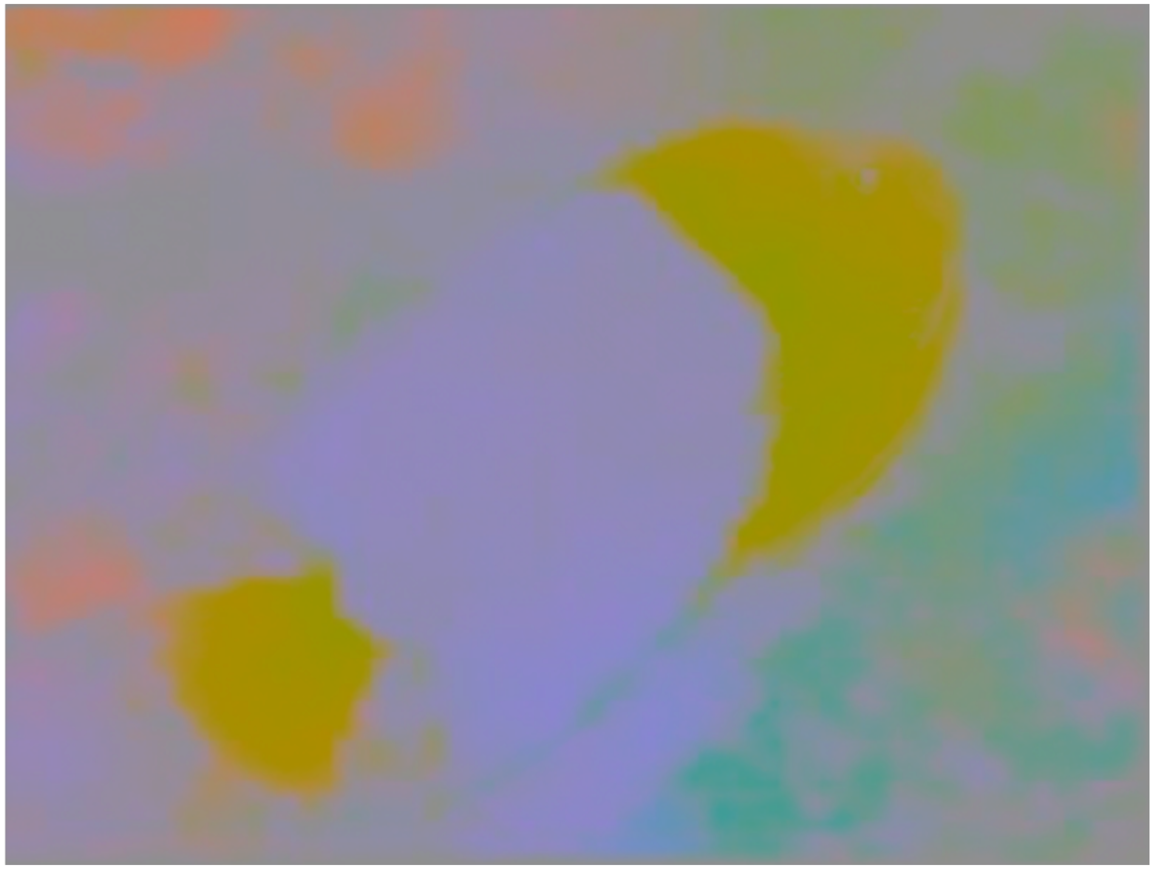
Dosovitskiy et al. Learning to Generate Chairs, Tables and Cars with Convolutional Networks PAMI 2017 (CVPR 2015)

Better Loss Functions

Ansel Adams. ***Yosemite Valley Bridge.***





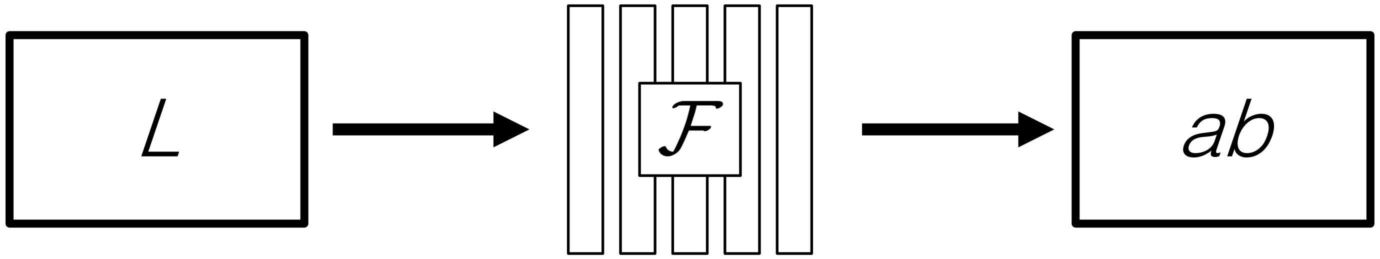


Grayscale image: *L* channel

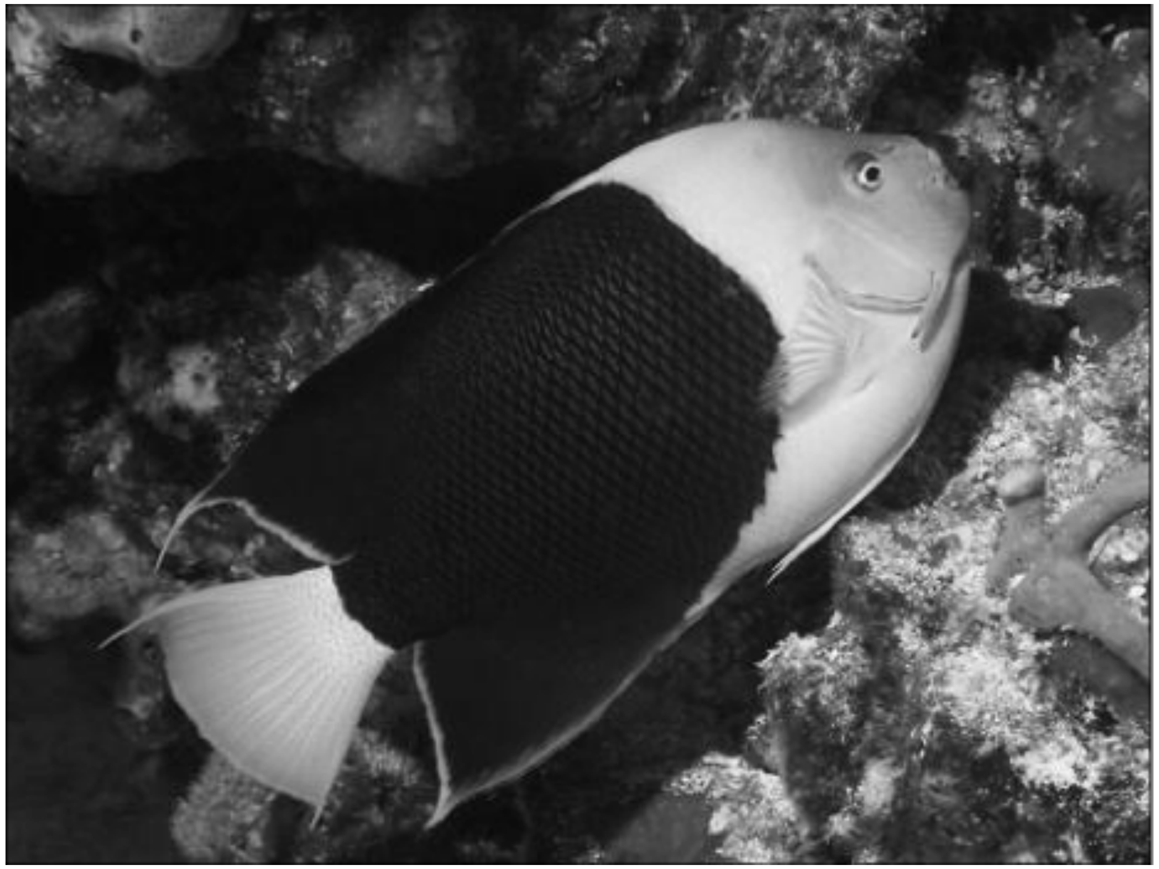
$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$

Color information: *ab* channels

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

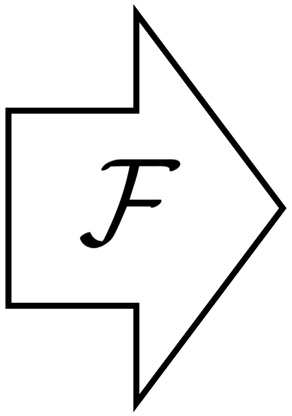


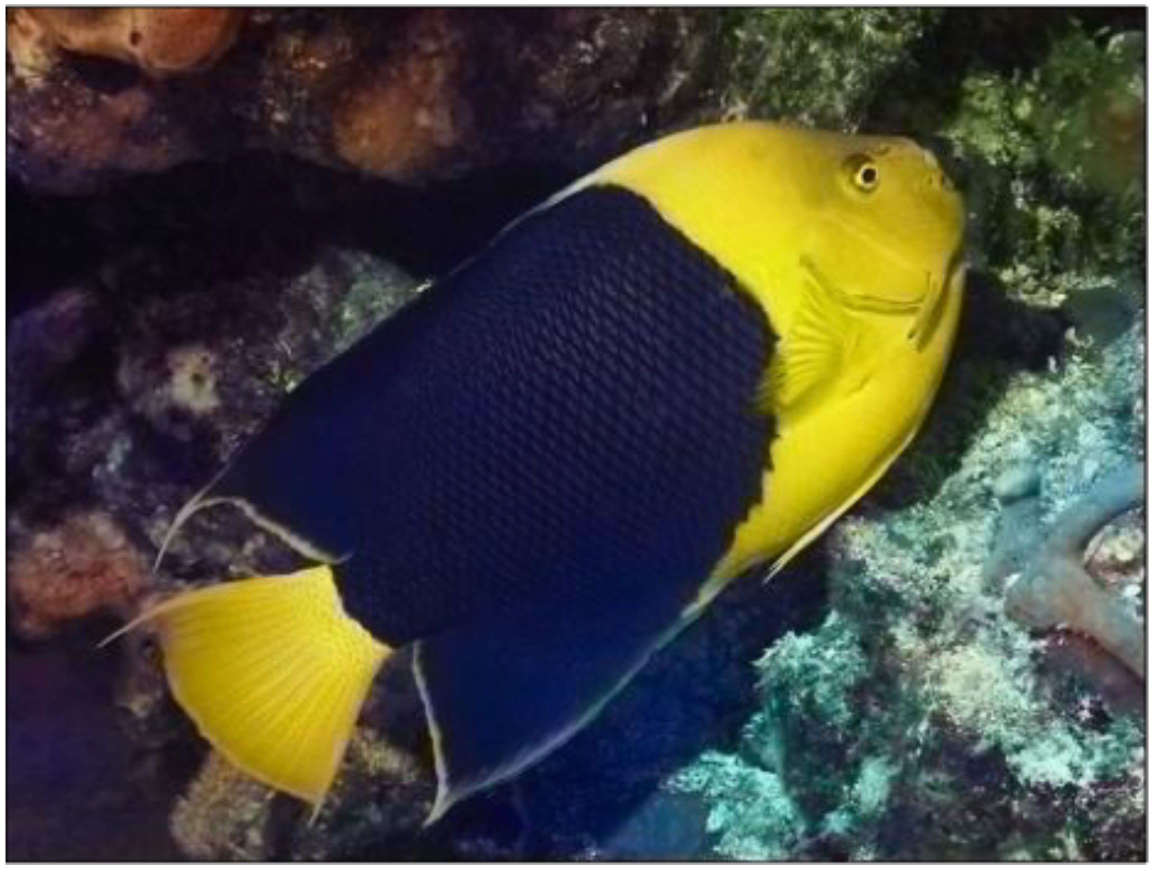
Zhang, Isola, Efros. *Colorful Image Colorization.* In *ECCV*, 2016.



Grayscale image: *L* channel

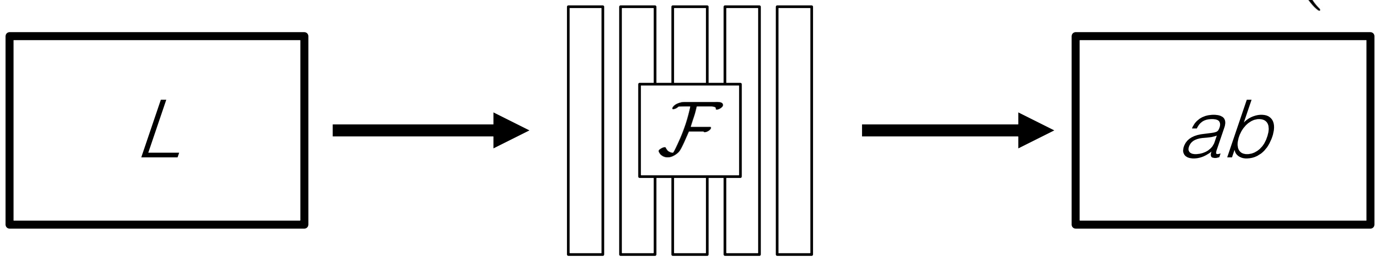
$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$





Concatenate (*L*,*ab*) channels

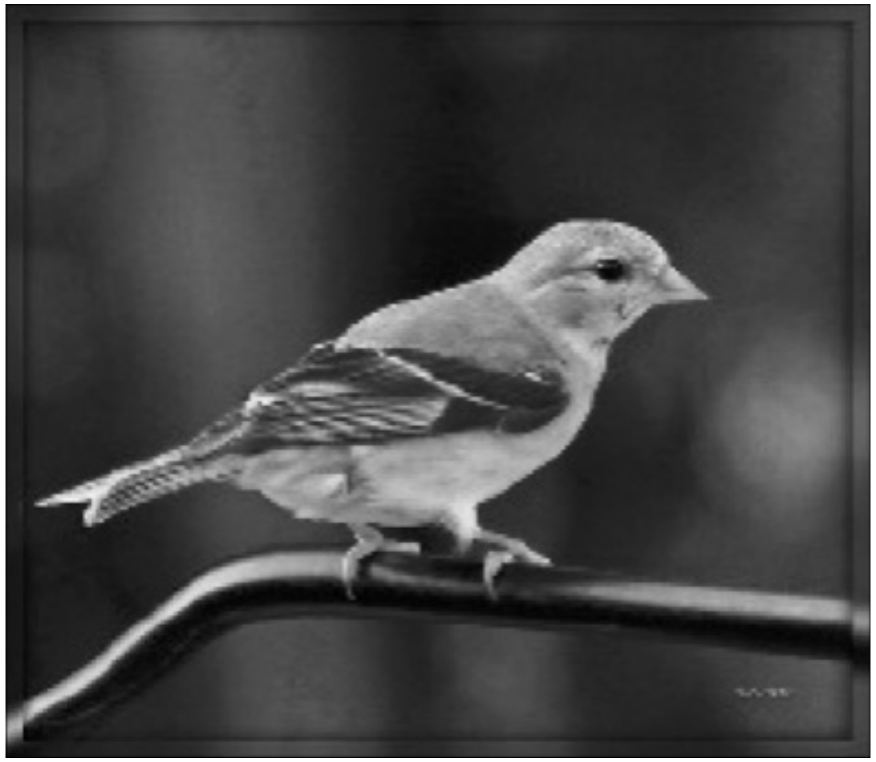
$$(\mathbf{X}, \widehat{\mathbf{Y}})$$



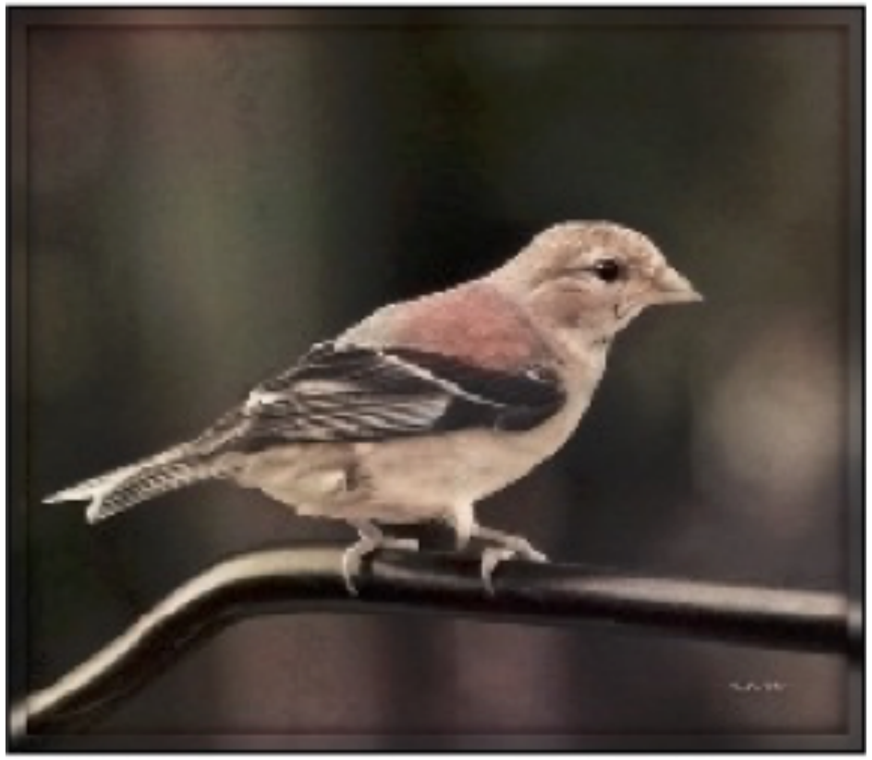
Zhang, Isola, Efros. *Colorful Image Colorization.* In *ECCV*, 2016.

Simple L2 regression doesn't work ☹

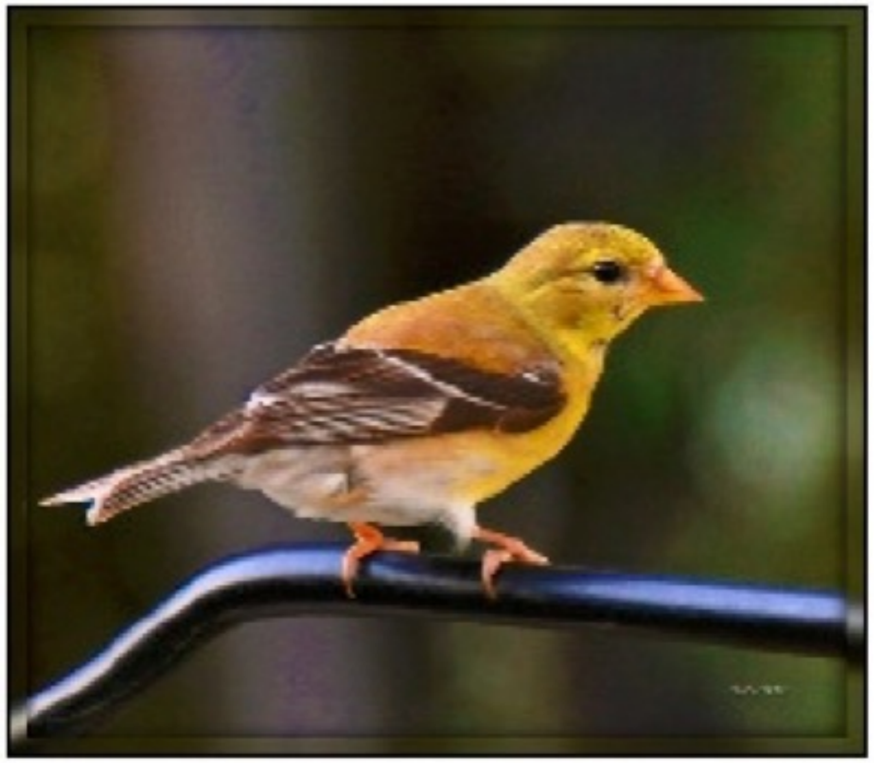
Input



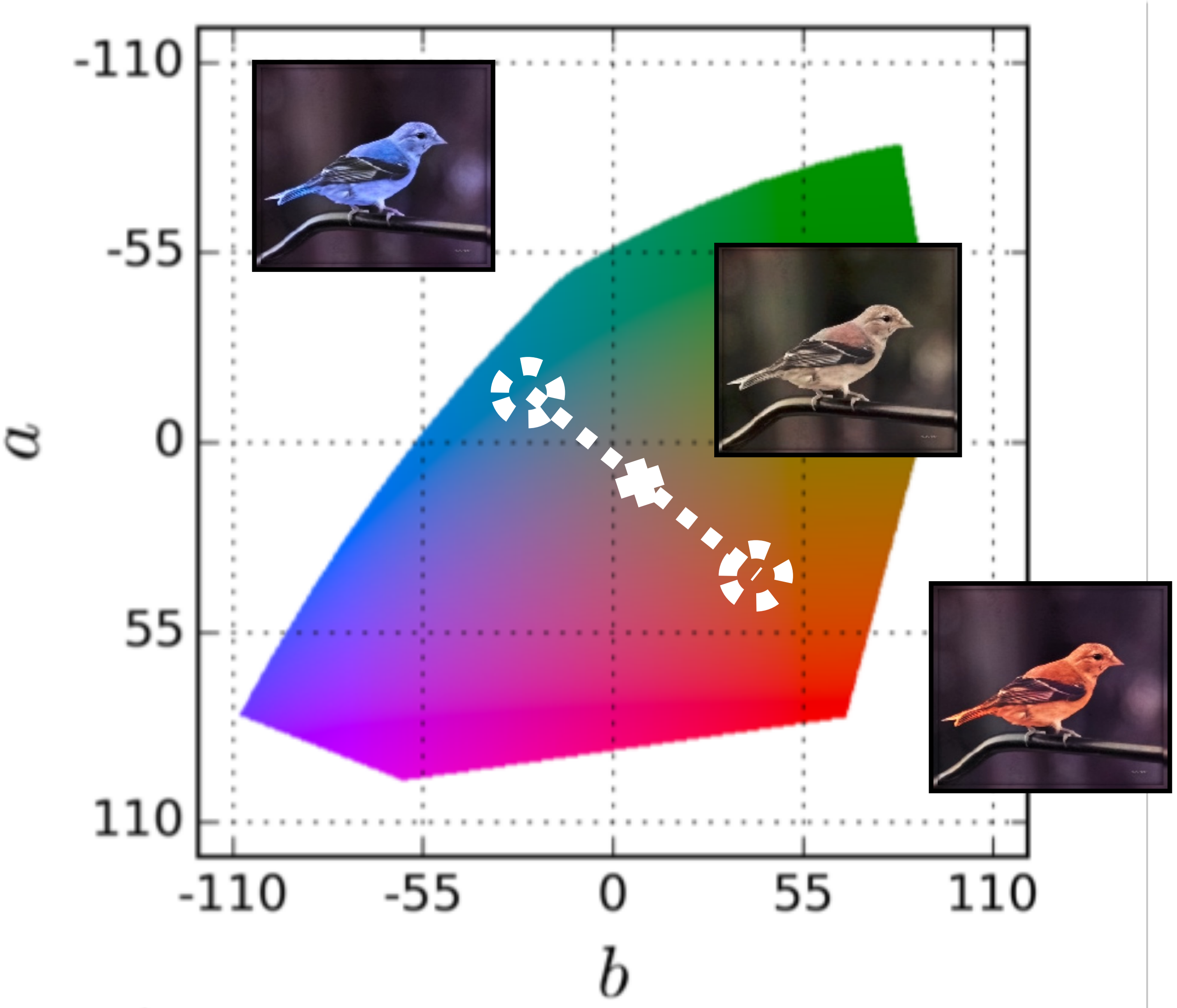
Output



Ground truth



$$\mathrm{L}_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2}\sum_{h,w} \|\mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}\|_2^2$$



$$\mathrm{L}_2(\widehat{\mathbf{Y}},\mathbf{Y}) = \frac{1}{2}\sum_{h,w}\|\mathbf{Y}_{h,w}-\widehat{\mathbf{Y}}_{h,w}\|_2^2$$

Better Loss Function

$$\theta^* = \arg\min_\theta \ell(\mathcal{F}_\theta(\mathbf{X}), \mathbf{Y})$$

- Regression with L2 loss inadequate

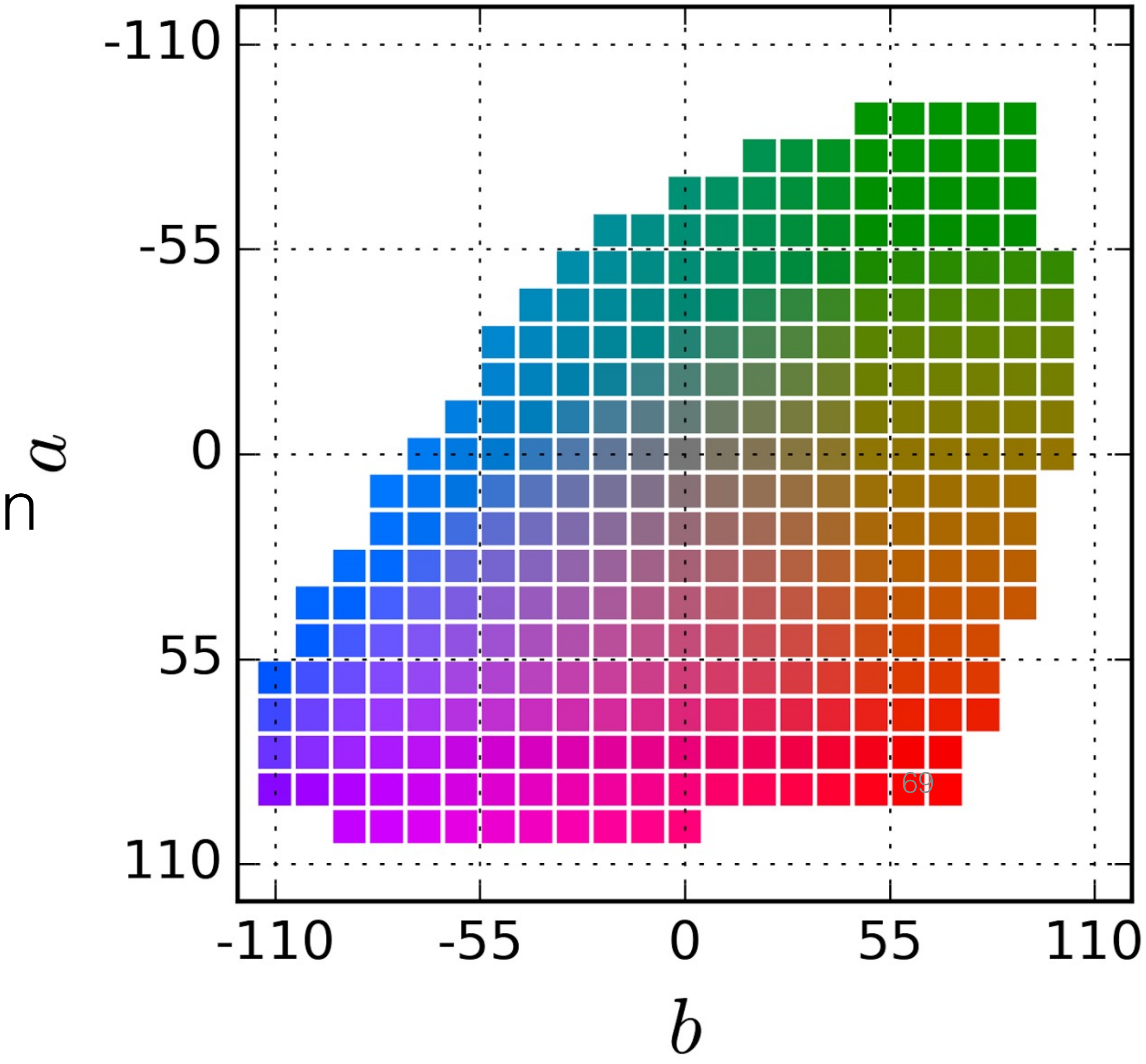
$$\mathrm{L}_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2}\sum_{h,w}\|\mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}\|_2^2$$

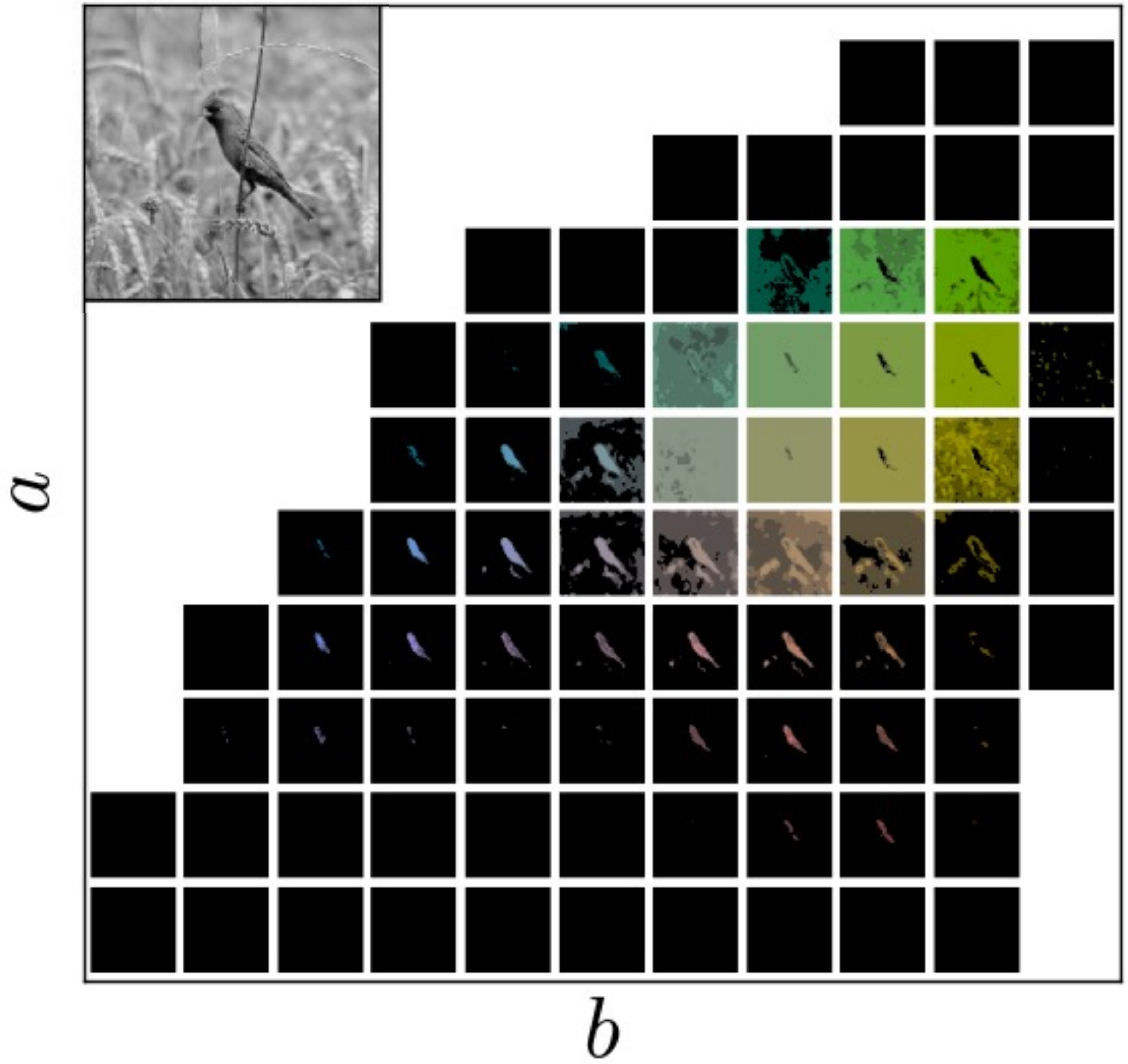
- Use per-pixel multinomial classification

$$\mathrm{L}(\widehat{\mathbf{Z}}, \mathbf{Z}) = -\frac{1}{HW}\sum_{h,w}\sum_q \mathbf{Z}_{h,w,q}\log(\widehat{\mathbf{Z}}_{h,w,q})$$

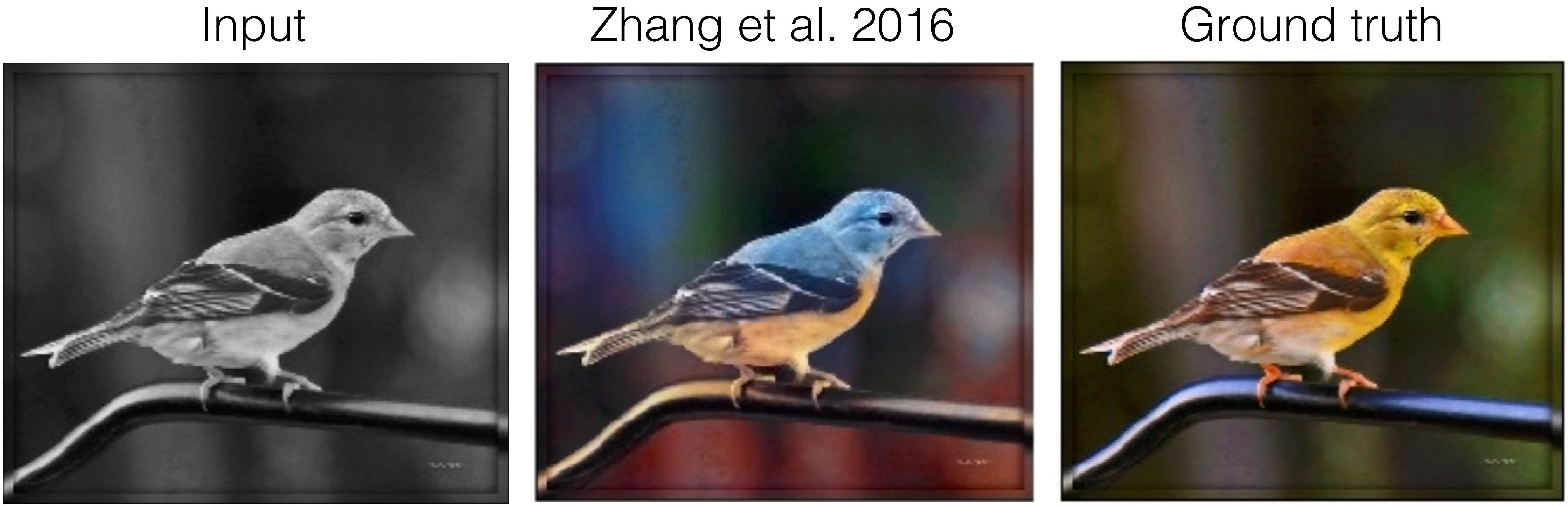
Colors in *ab* space

(discrete)




a
b

Designing loss functions



Input

Zhang et al. 2016

Ground truth

Color distribution cross-entropy loss with colorfulness enhancing term.

[Zhang, Isola, Efros, ECCV 2016]

Thank You!

16-726, Spring 2022

https://learning-image-synthesis.github.io/sp22/