

Advanced Machine Learning For Design

Lecture 5: Machine Learning for Image
Processing (part 2)

Evangelos Niforatos
19/10/2022

aml4d-ide@tudelft.nl
<https://aml4design.github.io/>

Admin

- Only 43/59 Peer Assessment forms completed for Group Assignment 1!
- We will publish Group Assignment feedback and aggregated peer assessment scores by next week.
- Preparation about the final exam
 - Content: Slides, Sources and Materials on <https://aml4design.github.io/>
 - Quizzes: Complete them at your own time
 - Example exam in the last week of the course
- Final Exam
 - Register for the final exam (Nov. 11 at 09:00–12:00)!
 - Registration for the final exam expires 14 calendar days before the day of the exam.
 - For more information, please check this website: <https://www.tudelft.nl/studenten/onderwijs/vakken-en-tentamens/tentamens/aanmelden-voor-tentamens>

How do humans see?

Hubel and Wiesel, 1959

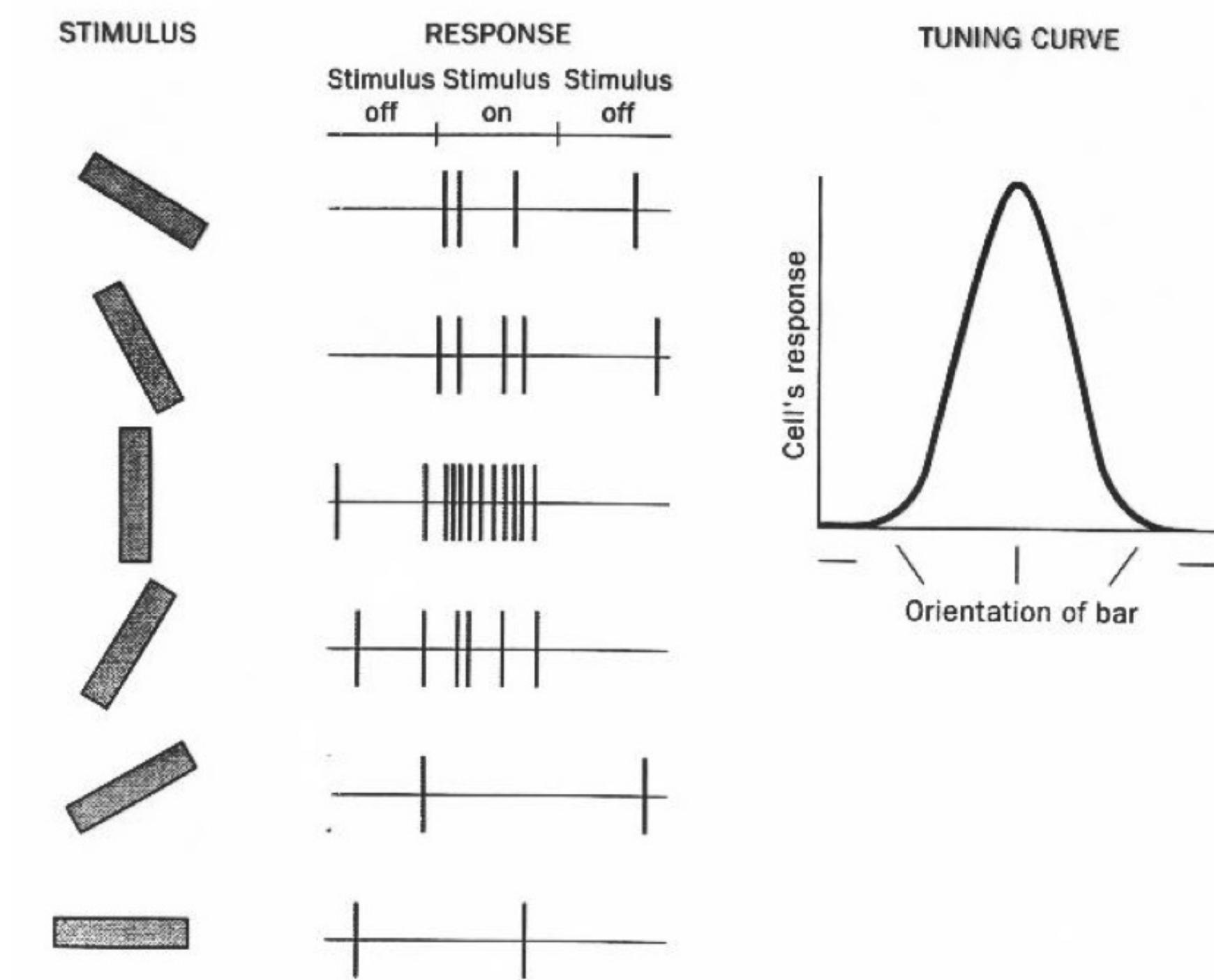
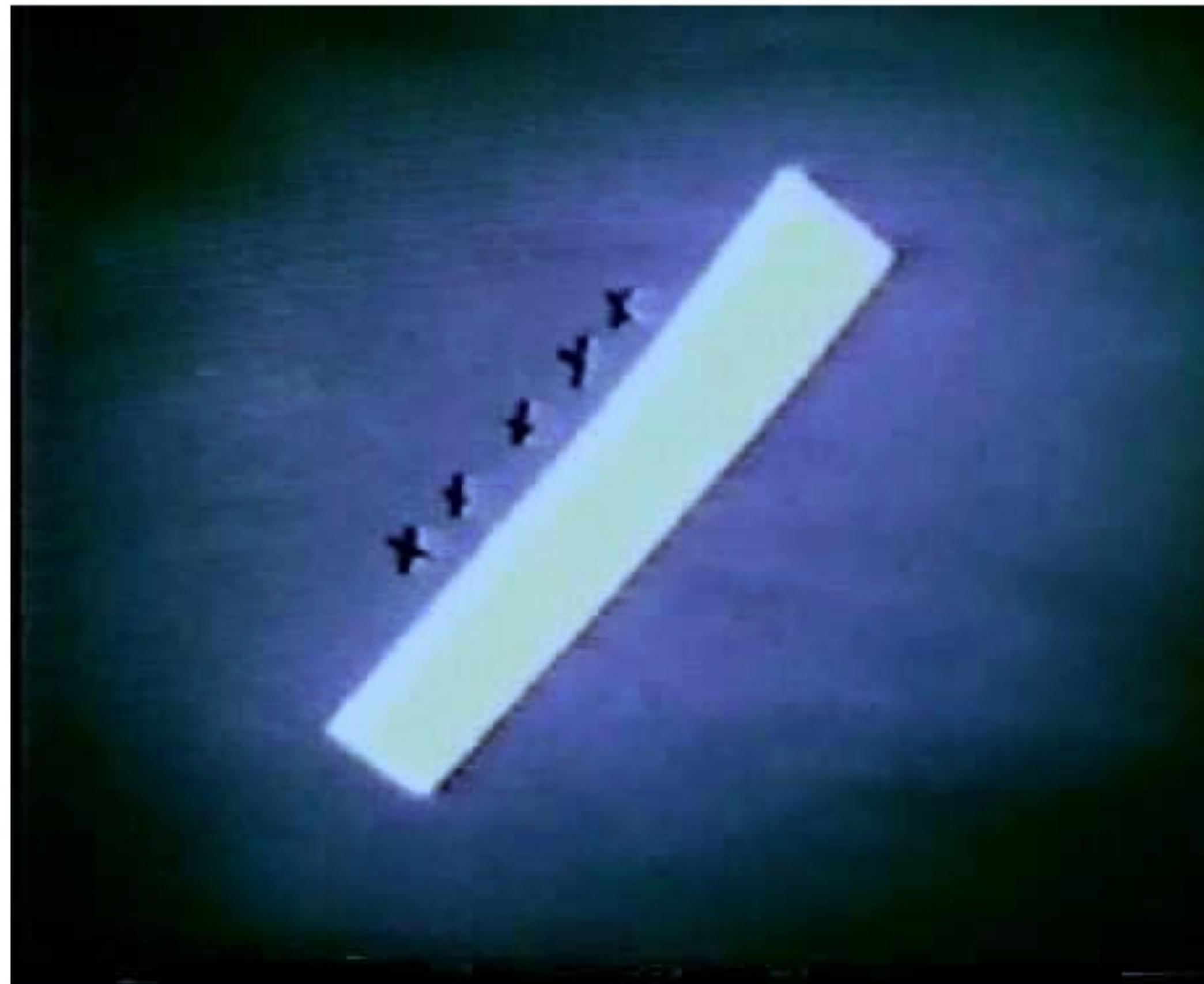
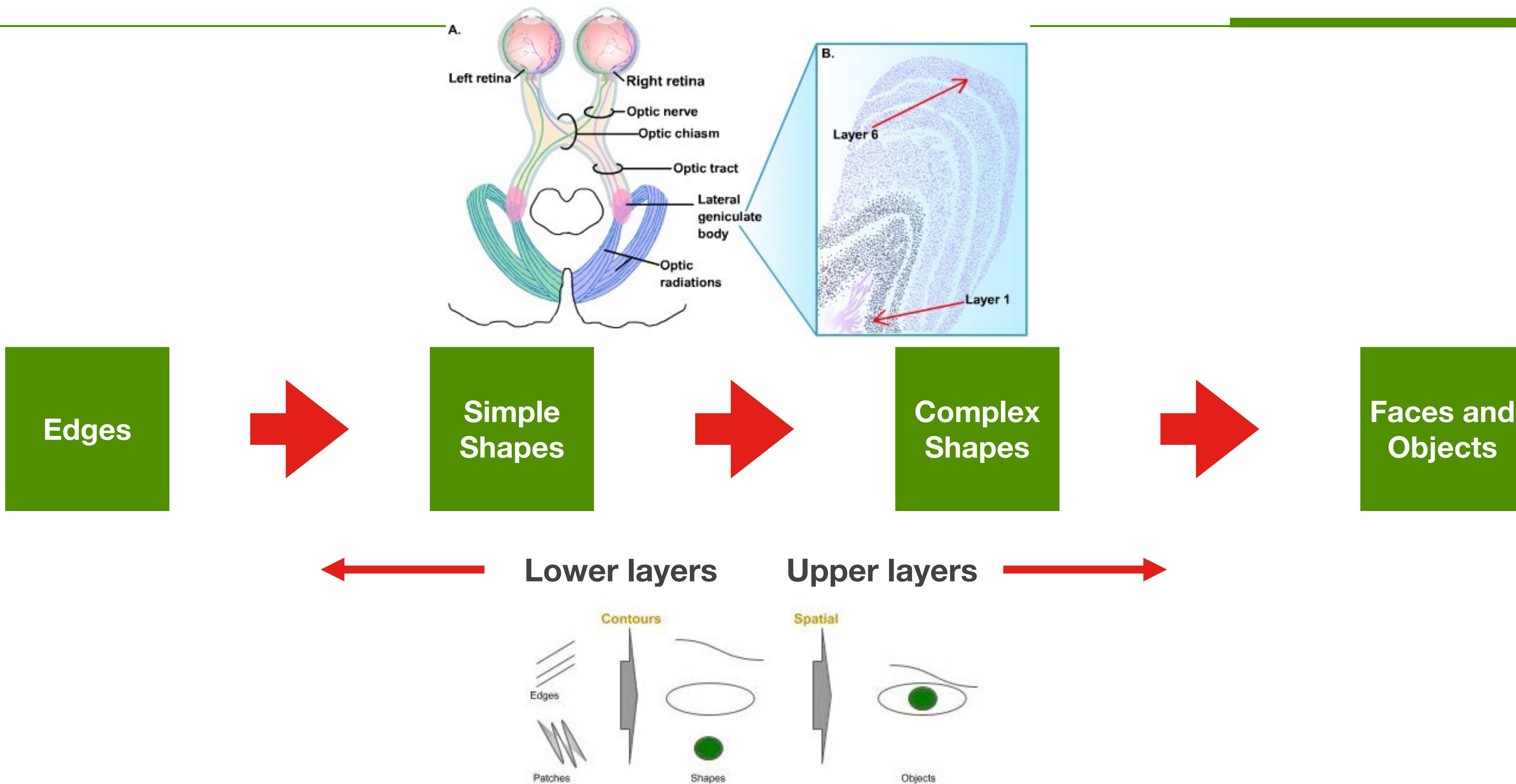


FIGURE 4.8 Response of a single cortical cell to bars presented at various orientations.

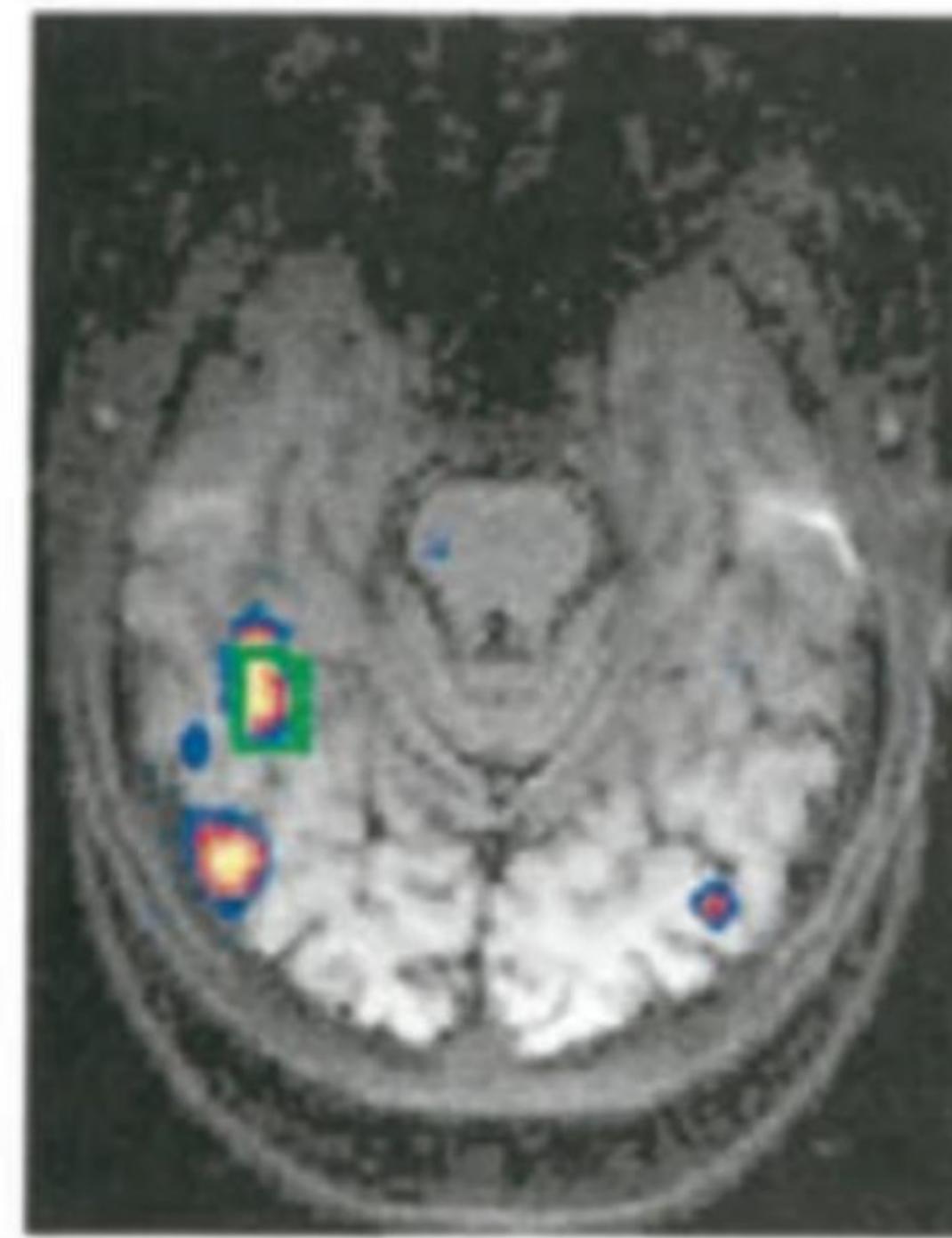
<https://www.youtube.com/watch?v=IOHayh06LJ4>

Neural Pathways



Neural Correlation of Objects & Scene Recognition

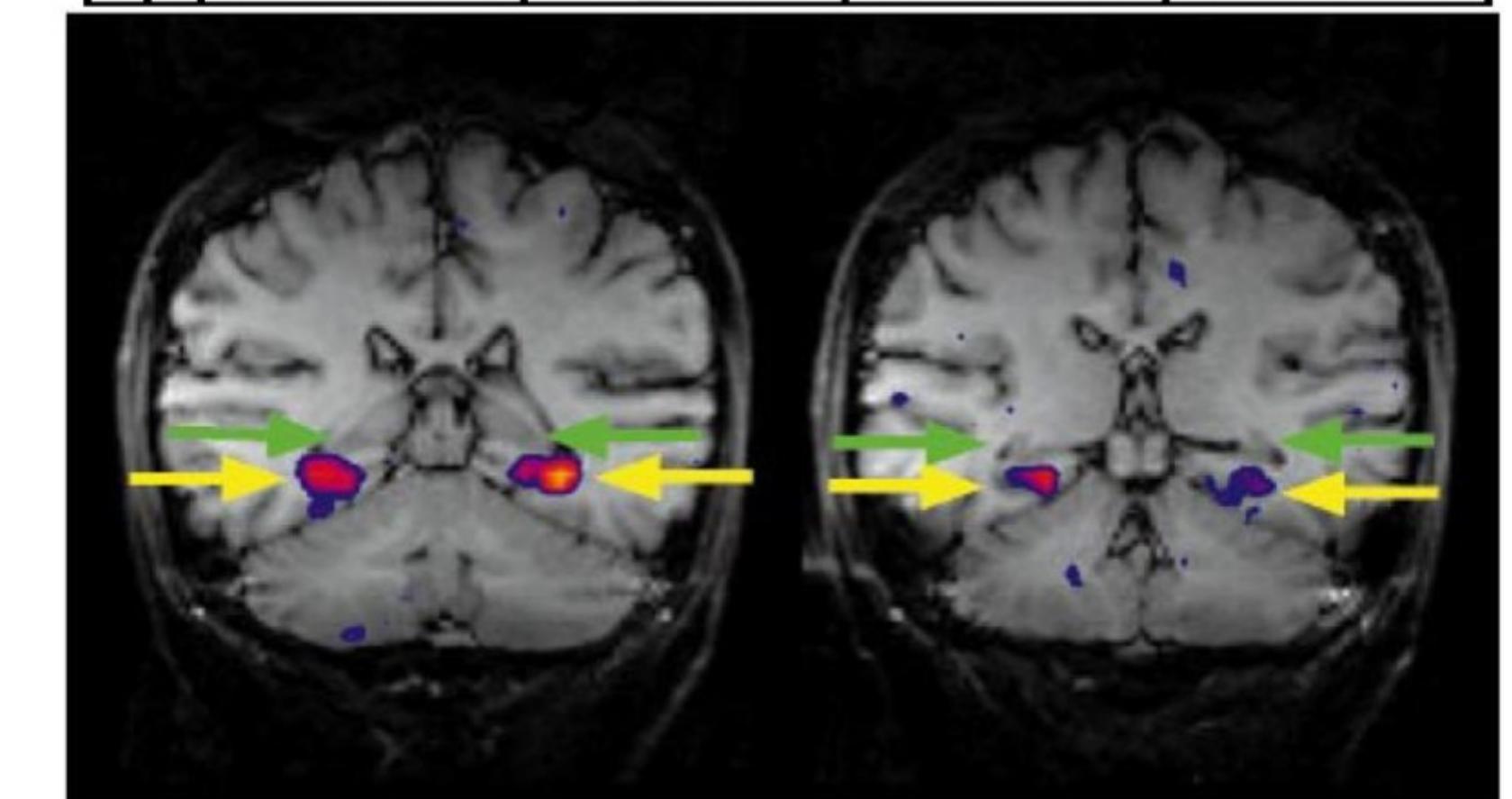
Faces > Houses



% signal change

a

	Faces	Objects	Houses	Scenes
Stimuli	Intact			
Scrambled				



Kanwisher et al. J. Neuro. 1997

Epstein & Kanwisher, Nature, 1998

Why is machine vision hard?

The deformable and truncated cat

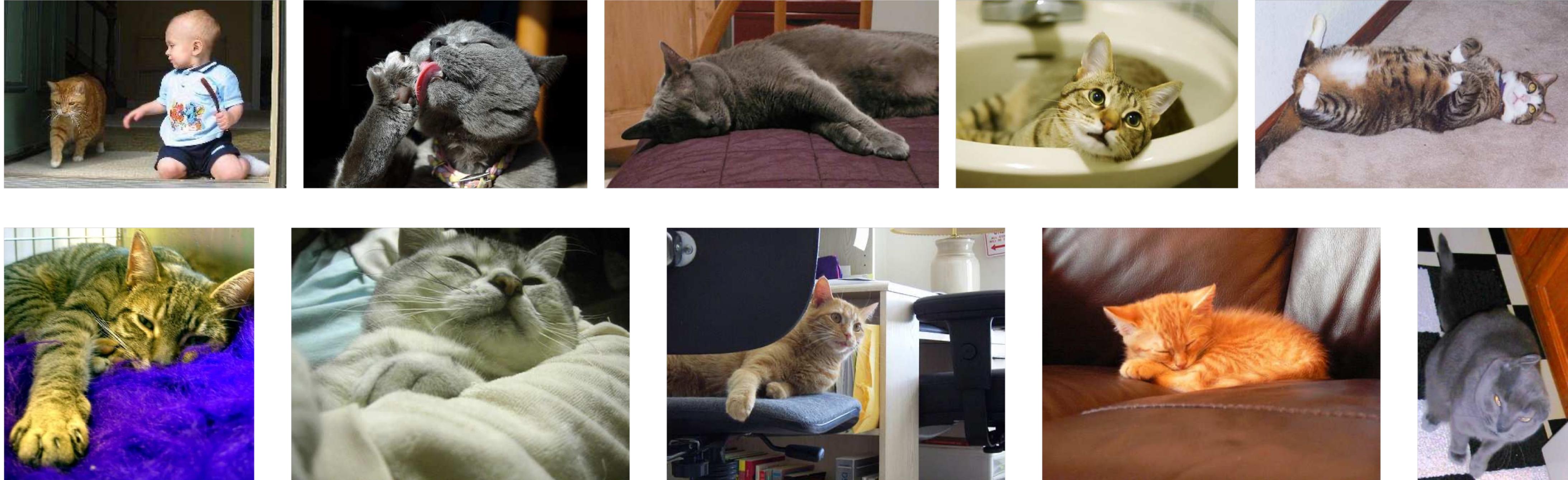
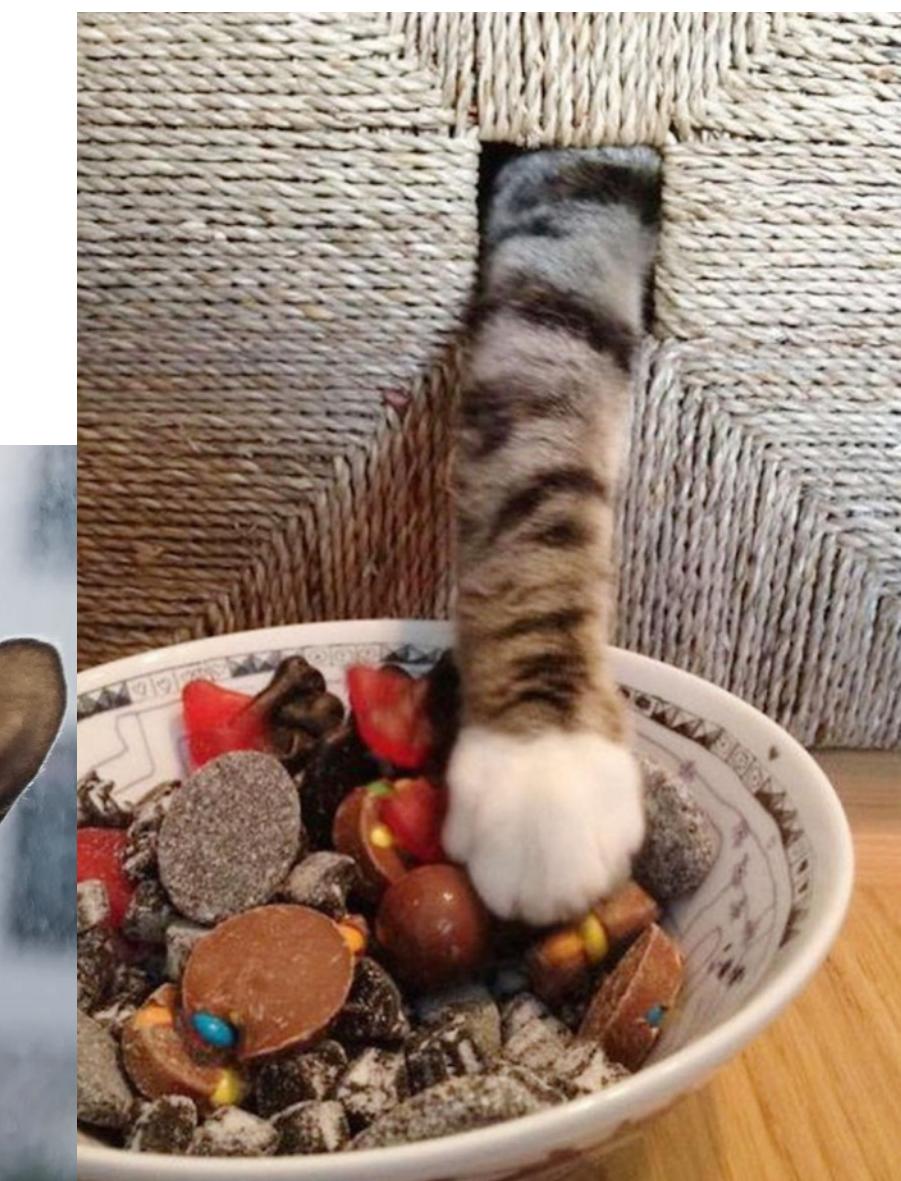
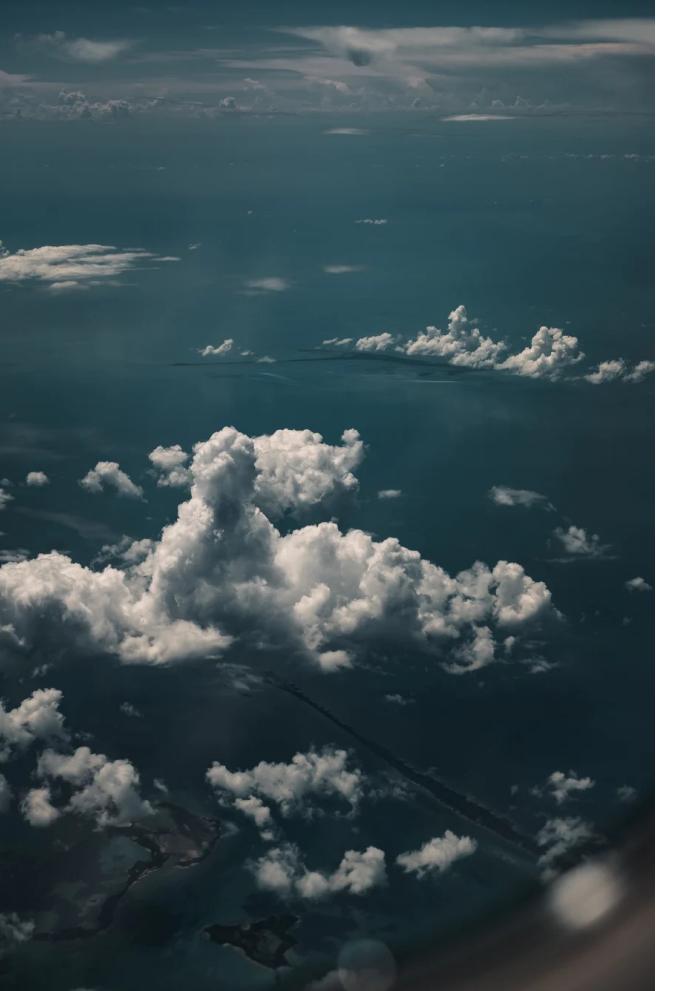
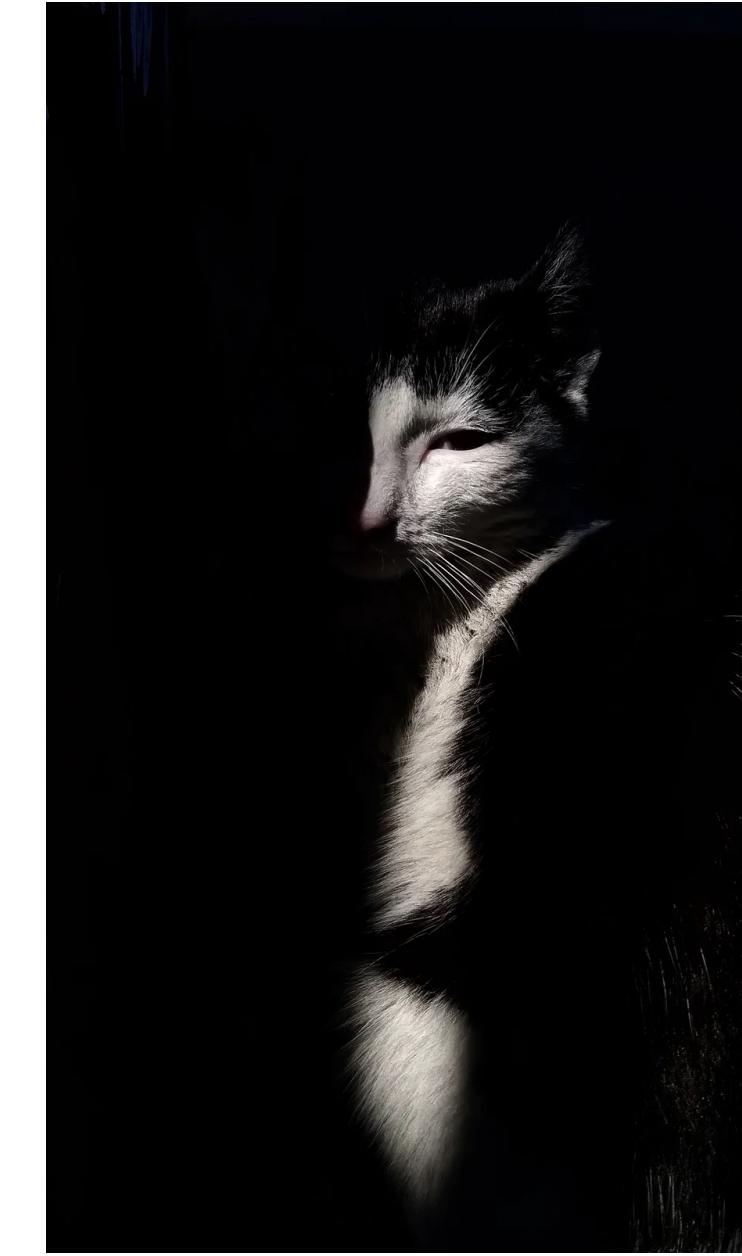
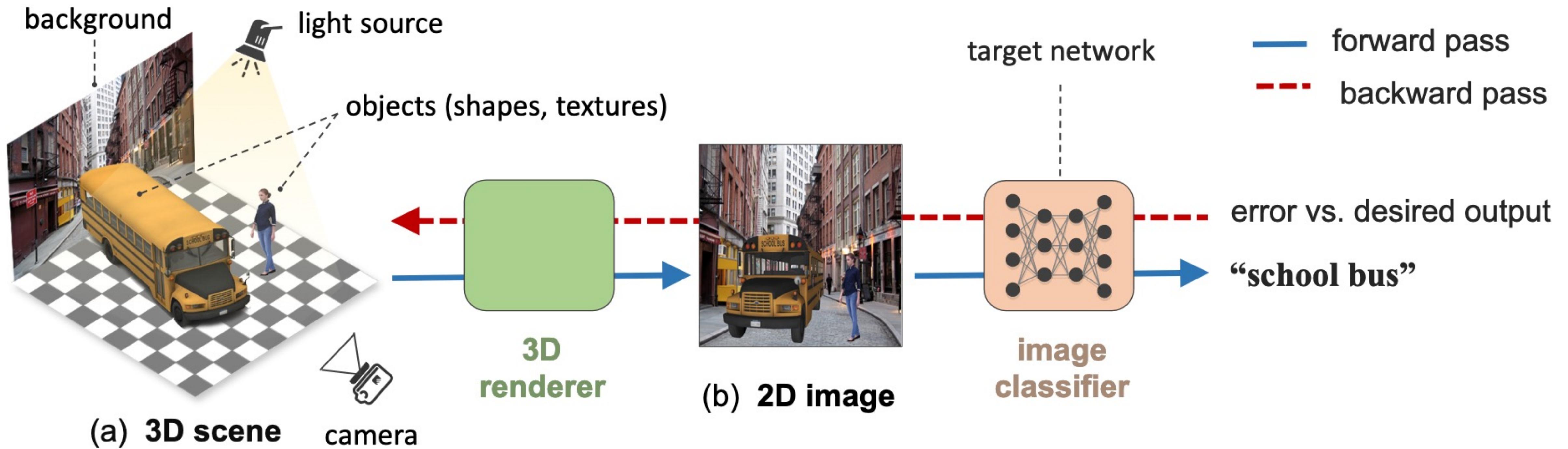


Figure 1. **The deformable and truncated cat.** Cats exhibit (almost) unconstrained variations in shape and layout.

Parkhi et al. *The truth about cats and dogs*. 2011





Strike (with) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects. Alcorn et al. 2019

Computer Vision Challenges

■ Viewpoint Variation

- A single instance of an object can be oriented in many ways with respect to the camera

■ Scale variation

- Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image)

■ Deformation

- Many objects of interest are not rigid bodies and can be deformed in extreme ways

■ Occlusion

- The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible

■ Illumination Condition

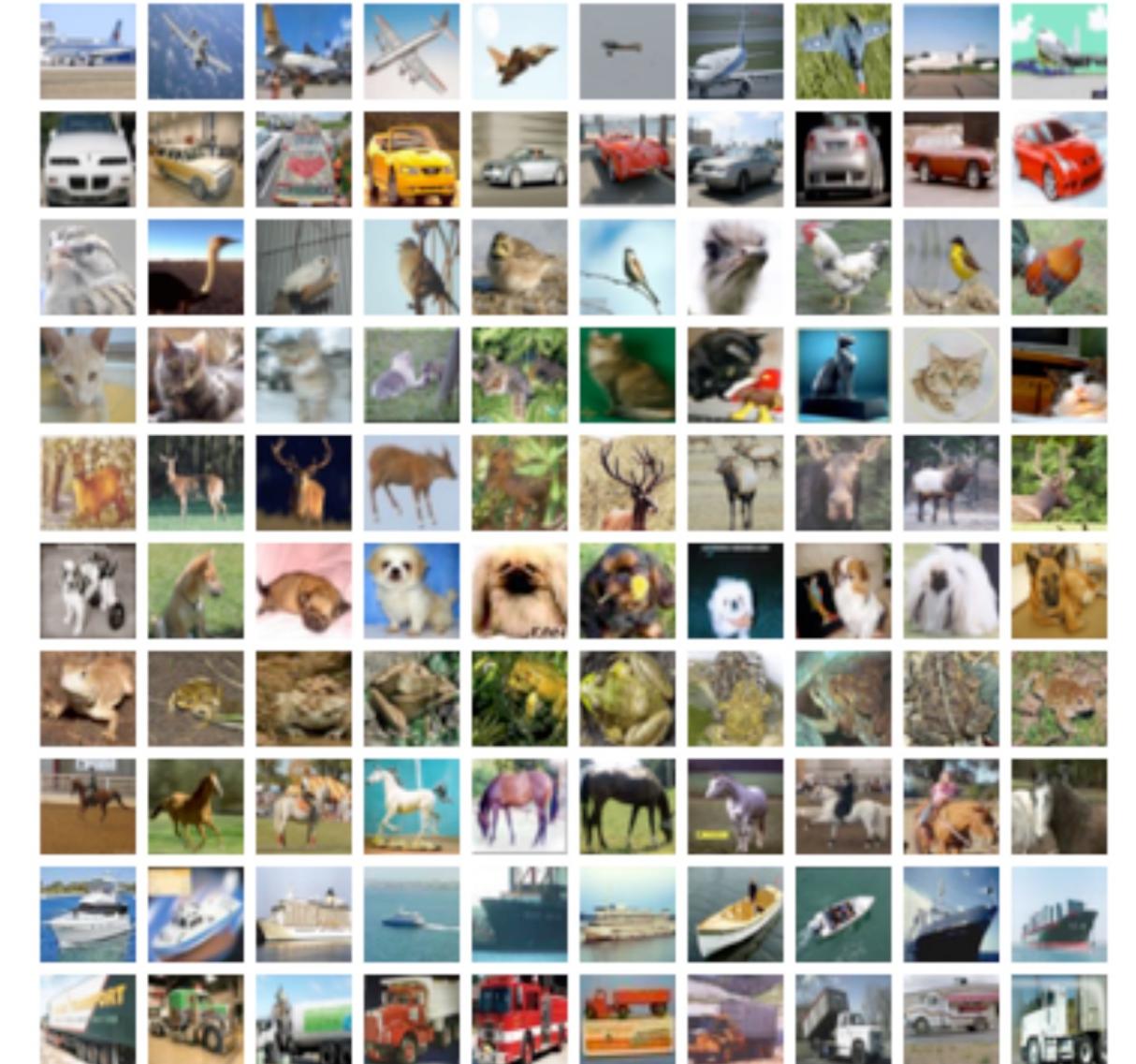
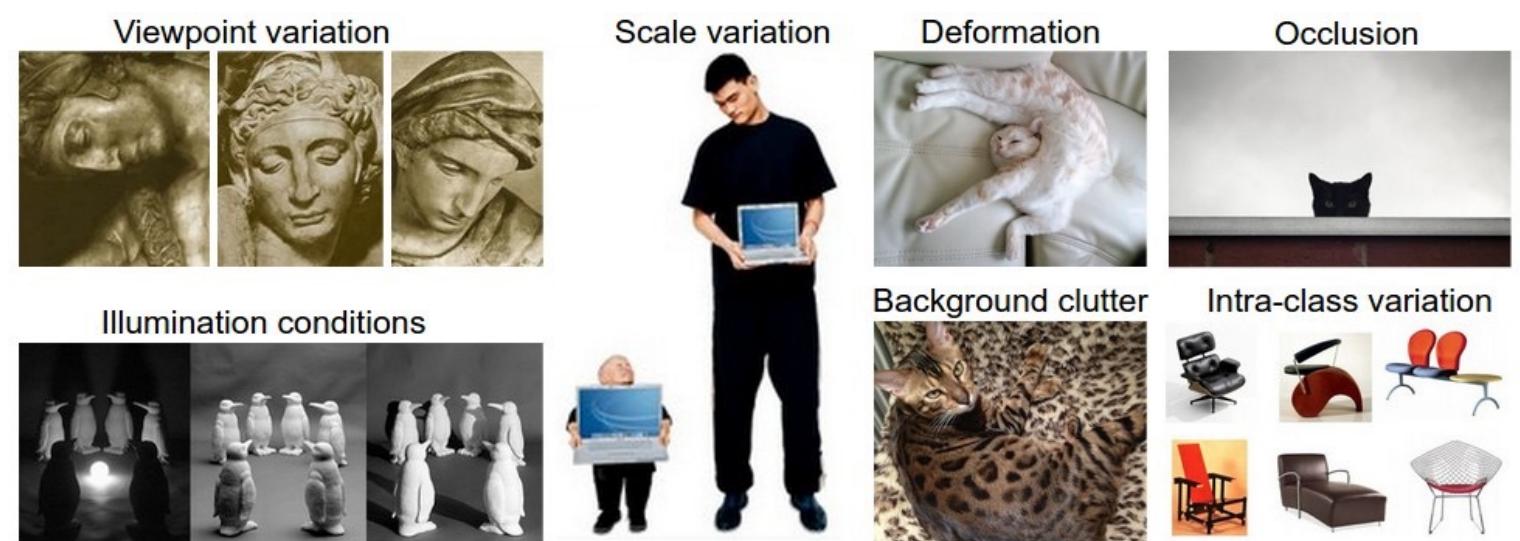
- The effects of illumination are drastic on the pixel level

■ Background clutter

- The objects of interest may blend into their environment, making them hard to identify

■ Intra-class variation

- The classes of interest can often be relatively broad, such as chair. There are many different types of these objects, each with their own appearance

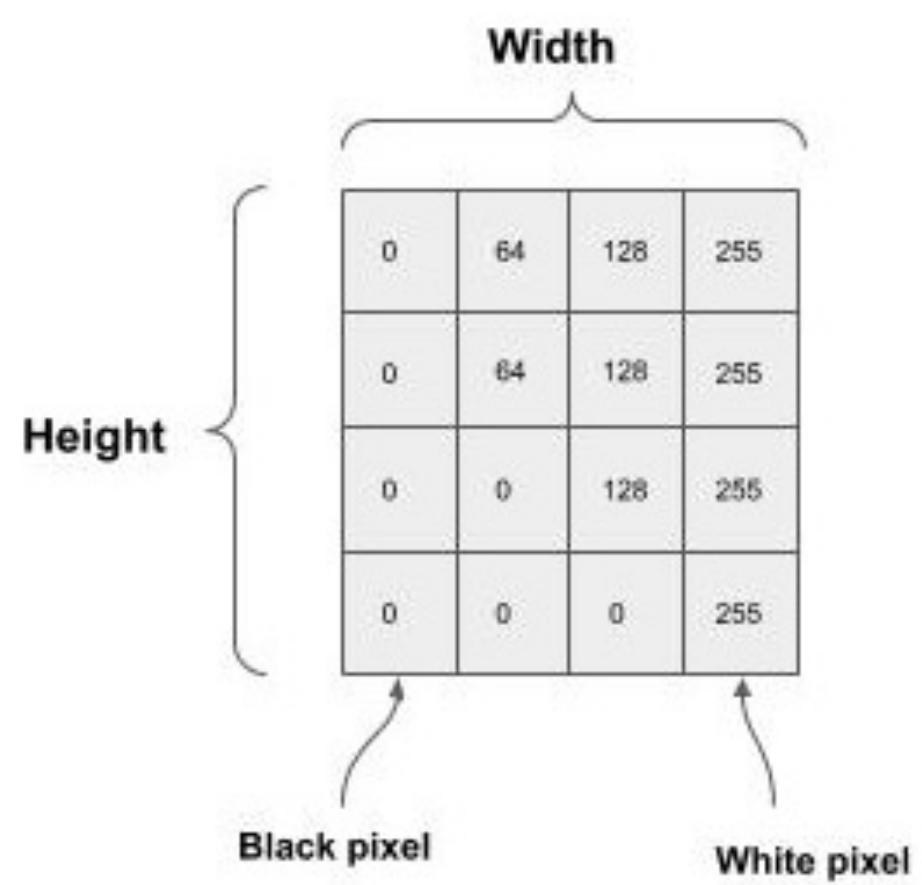
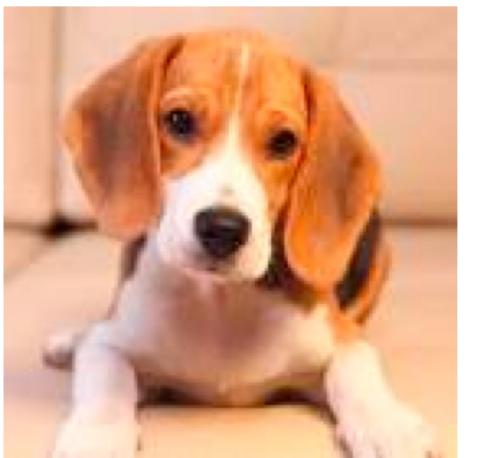


Let's see this in practice: Real-time Object Detection Demo

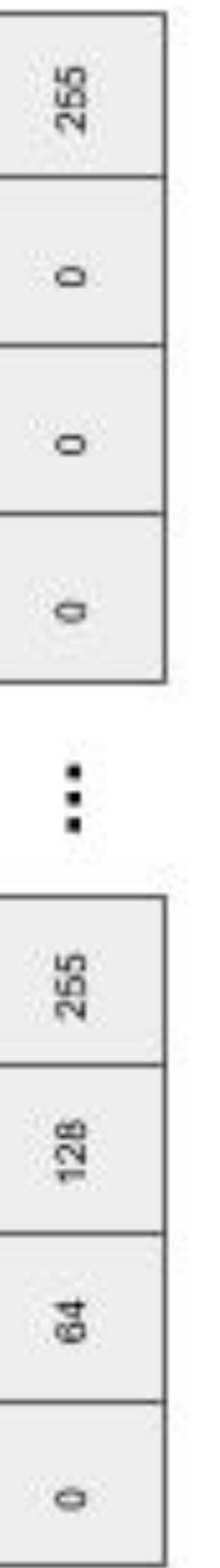
Model: [YOLOv5 in PyTorch](#) when / where does the model fail?
Find code on: <https://aml4design.github.io/code/>

How CV models work?

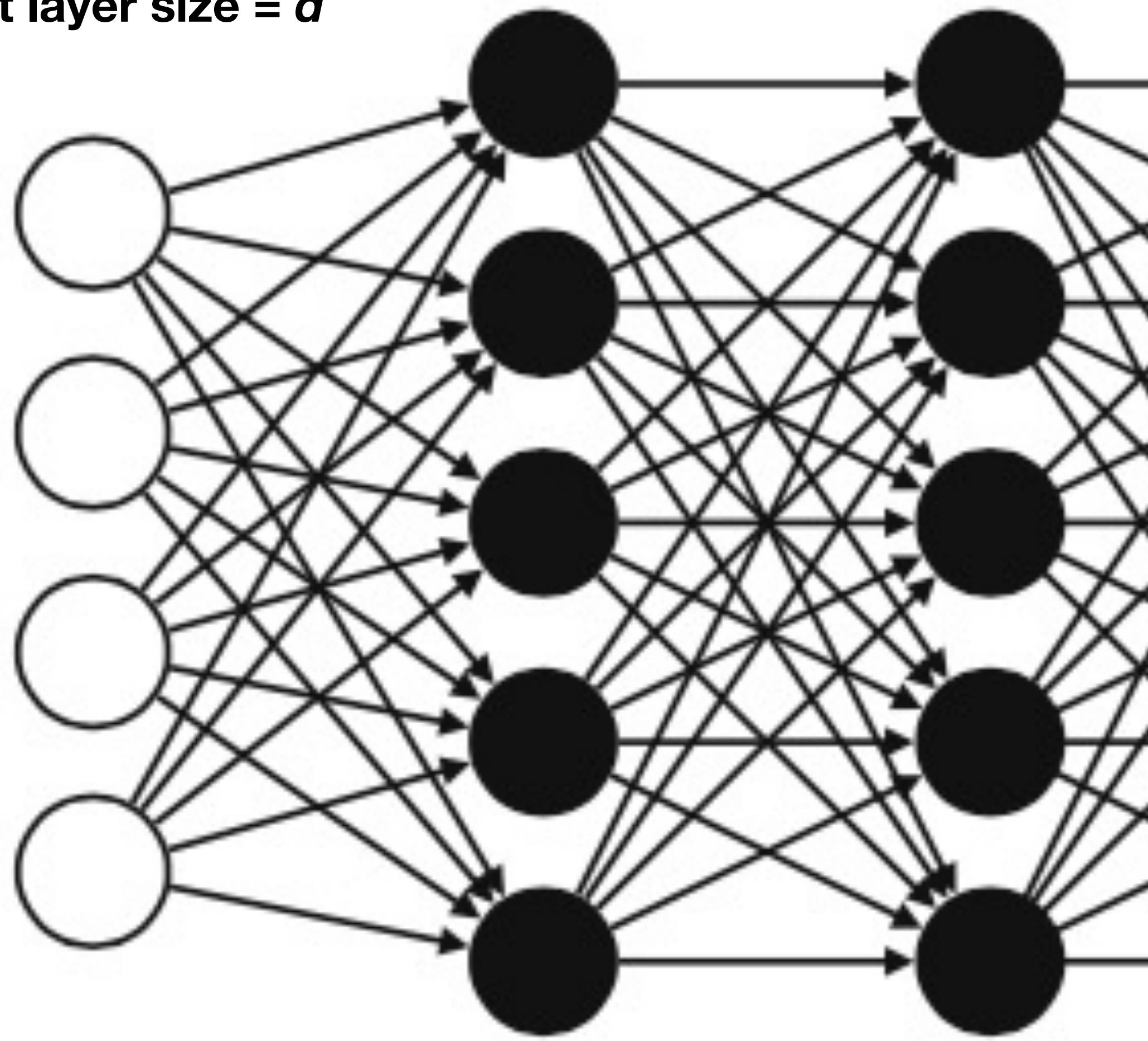
Flattening



$d = \text{width} \times \text{height}$

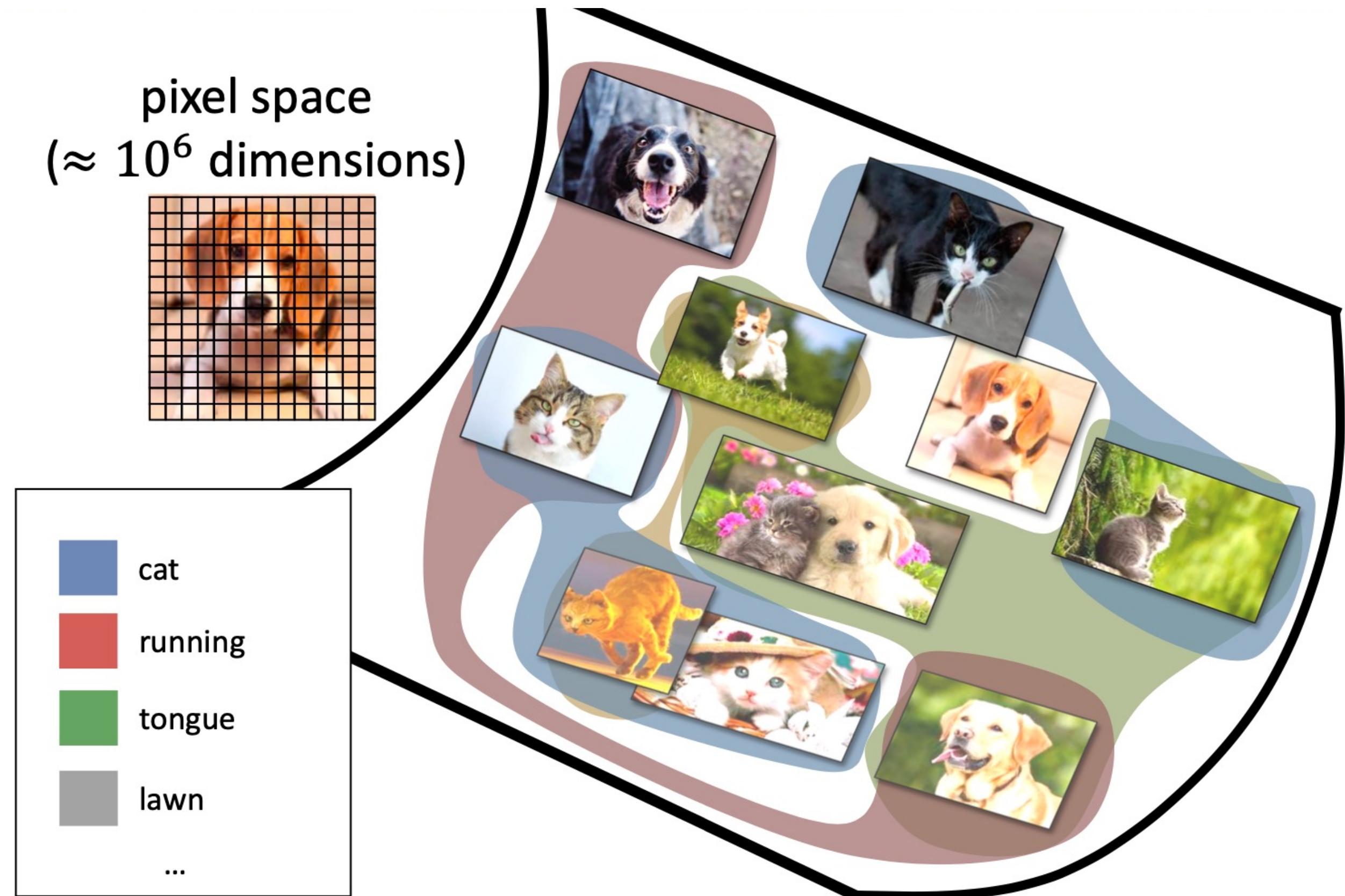


Input layer size = d



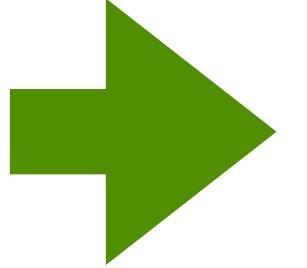
Curse of dimensionality

- High dimensionality
 - A 1024×768 image has $d = 786432!$
 - A tiny 32×32 image has $d = 1024$
- Decision boundaries in pixel space are extremely complex
- We will need “big” ML models with lots of parameters
 - For example, linear regressors need d parameters



Downsampling

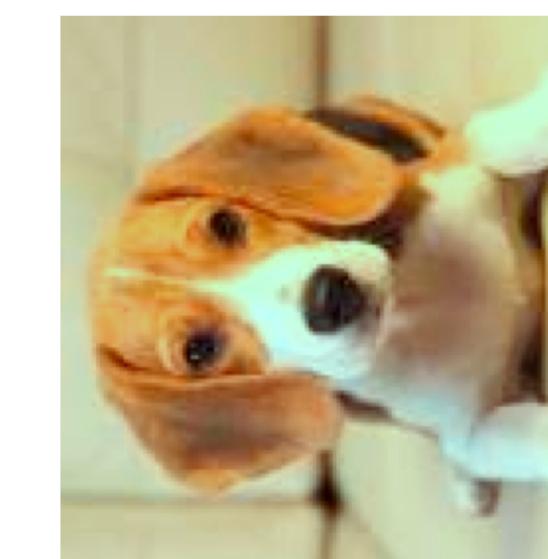
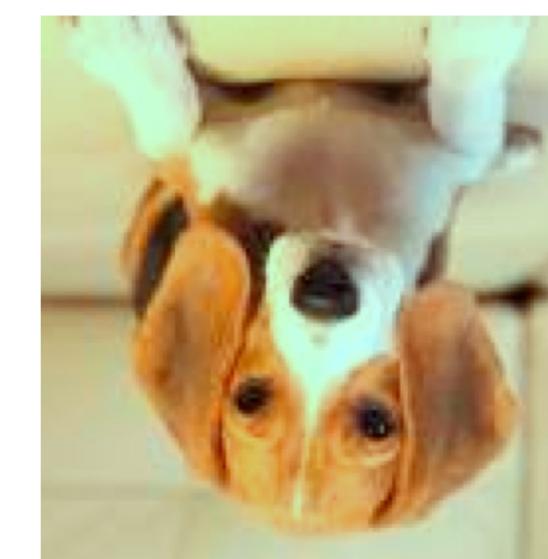
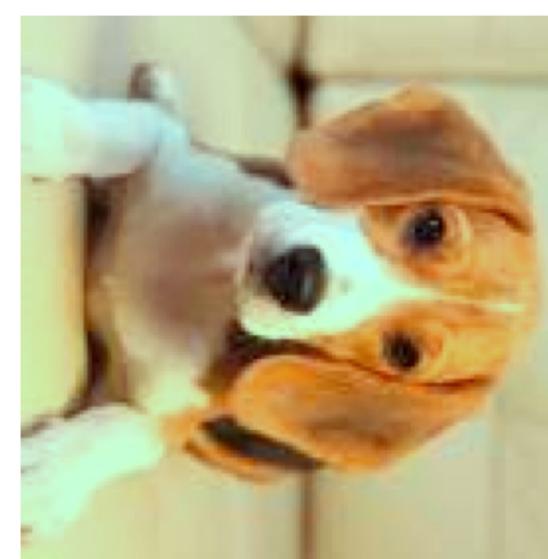
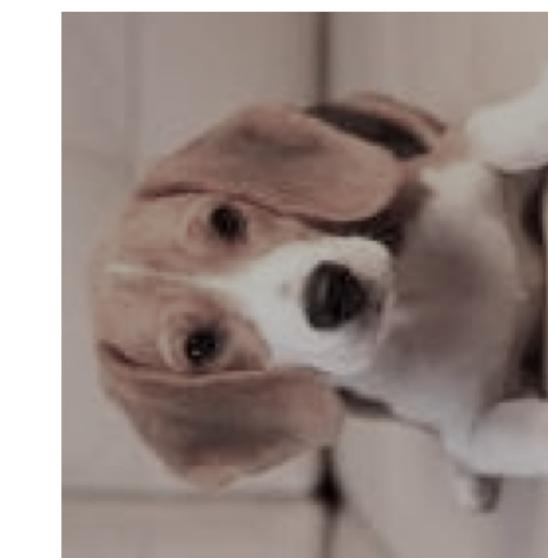
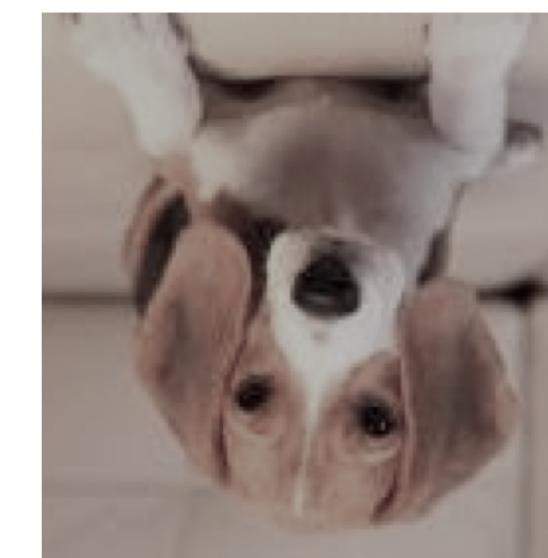
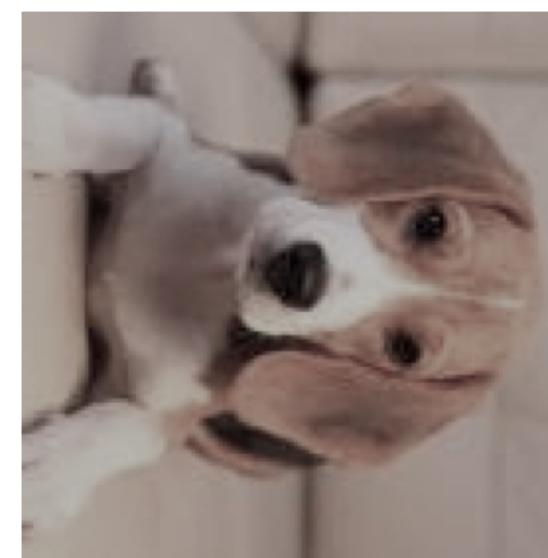
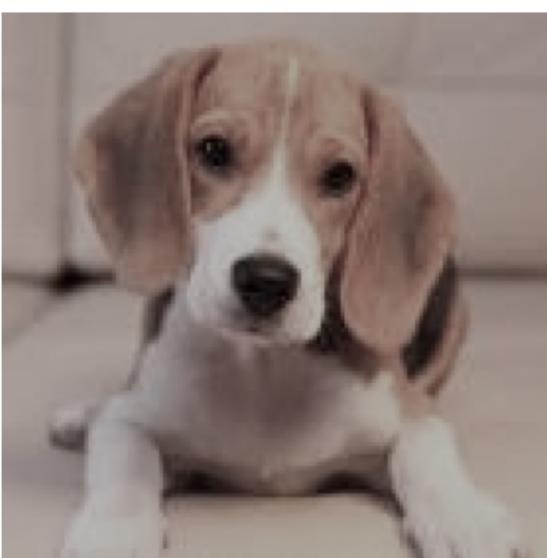
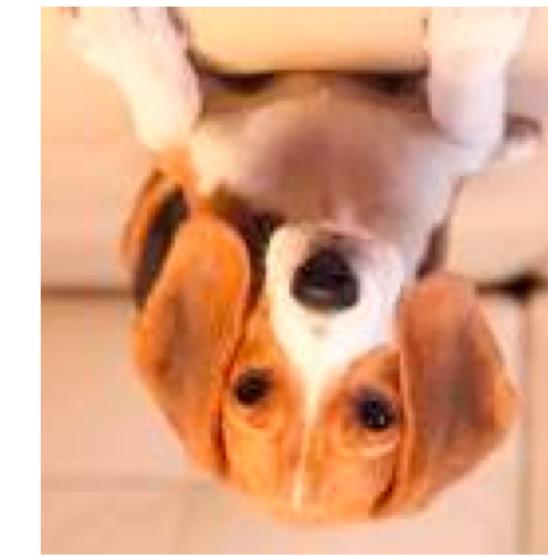
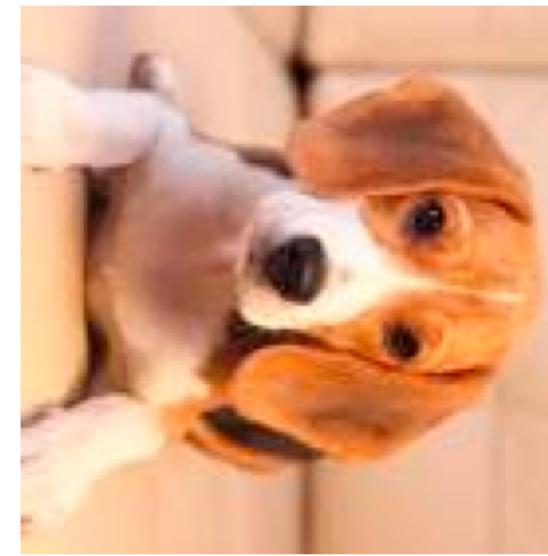
1024
1024



224
224



What about generalisation?



The “old days:” Feature Extraction

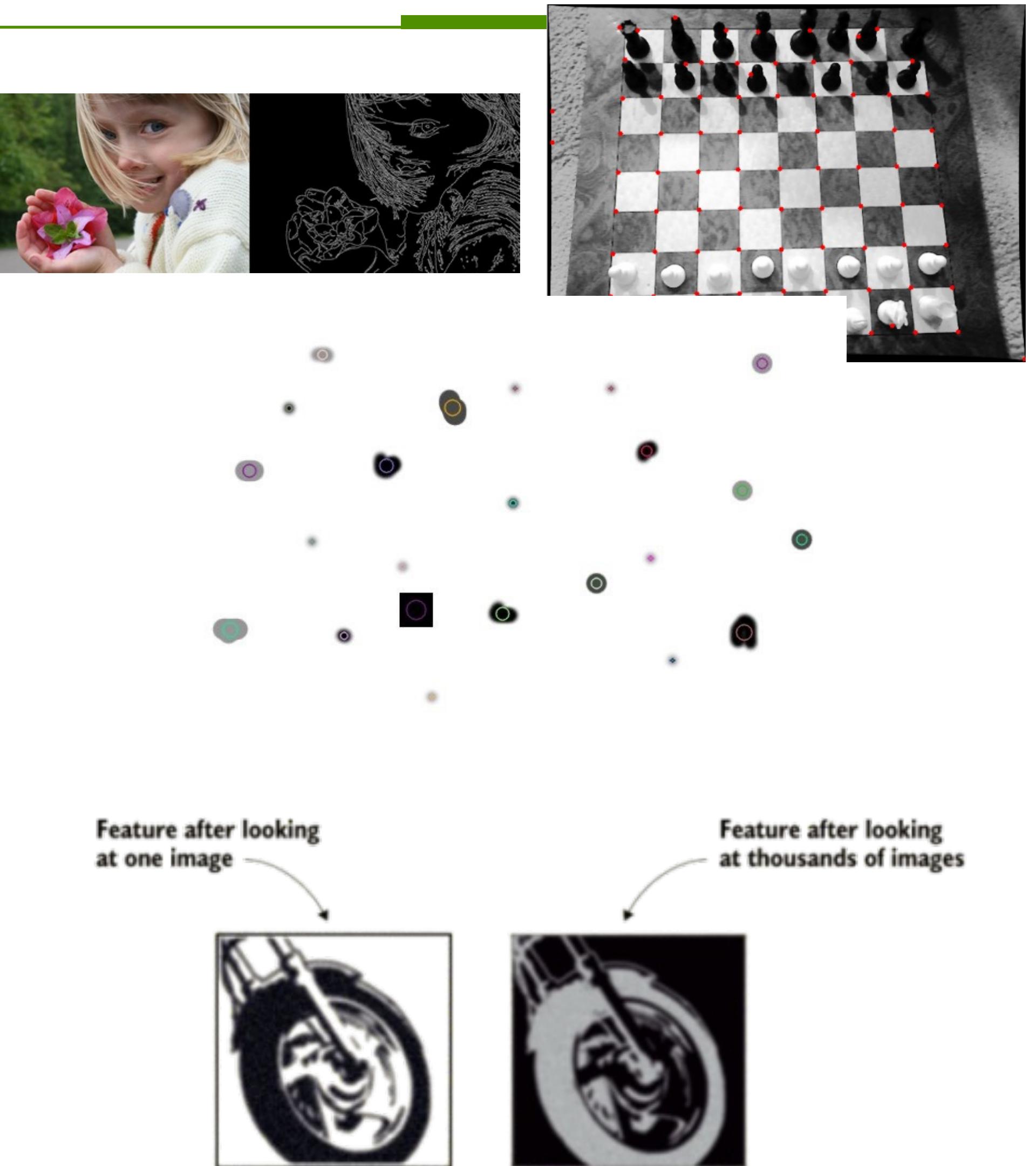
- **Feature**

- A relevant piece of information about the content of an image
 - e.g., edges, corners, blobs (regions), ridges



- A good feature:

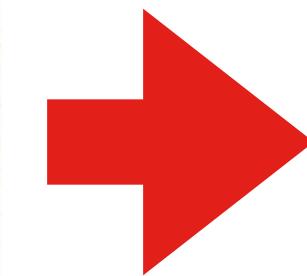
- is repeatable
- identifiable
- can be easily tracked and compared
- is consistent across different scales, lighting conditions, and viewing angles
- is still visible in noisy images or when only part of an object is visible
- can distinguish objects from one another



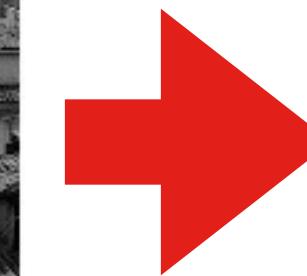
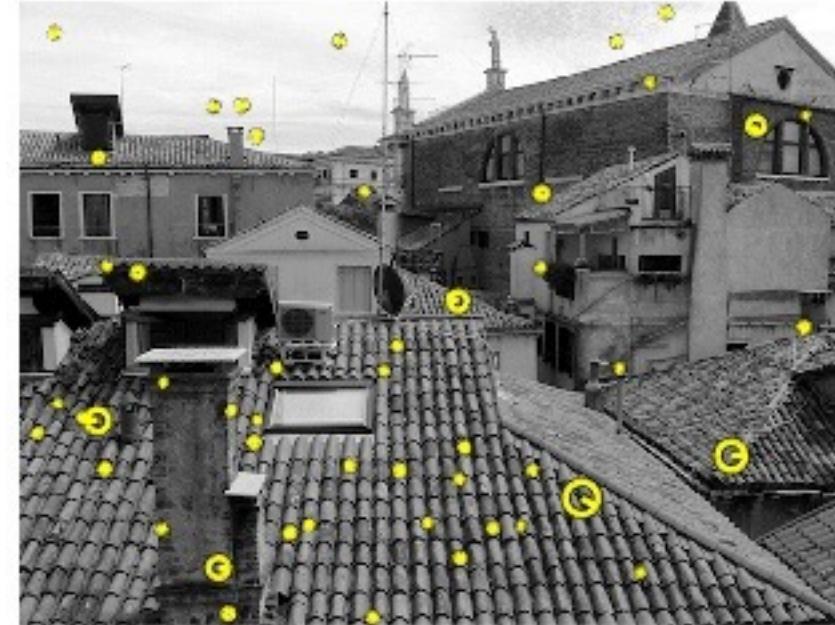
Feature Extraction Techniques

<https://www.vlfeat.org>

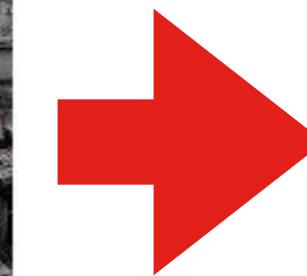
Scale-Invariant Feature Transform (SIFT)



Find “interest points”
in the scene



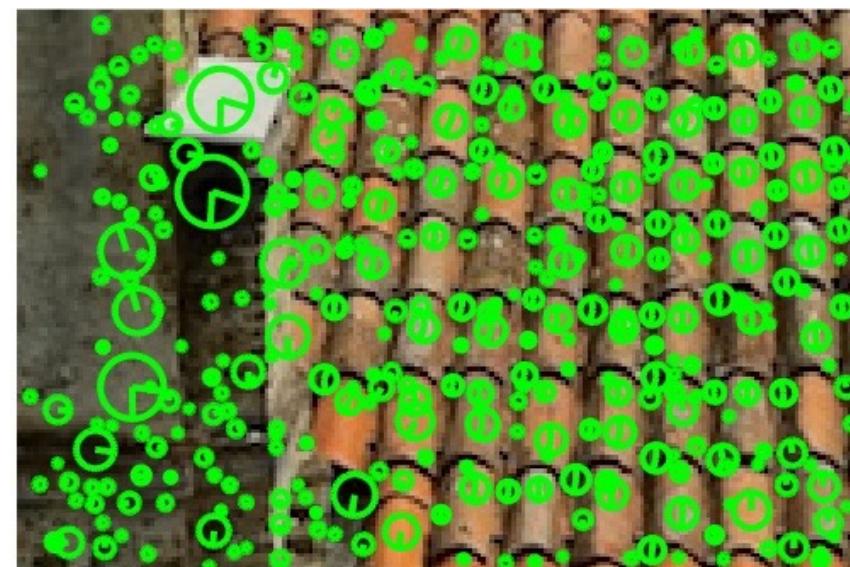
Compute features at
interest points



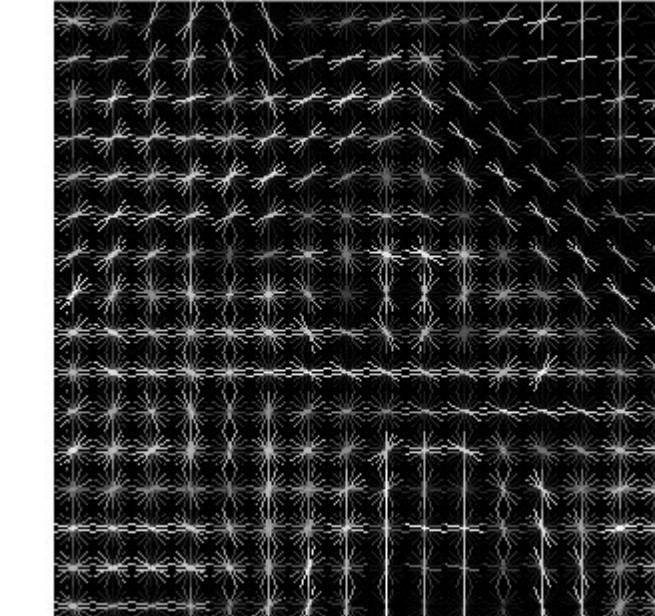
Convert to fixed-dimensional
feature vector
Histogram



Co-variant feature detector

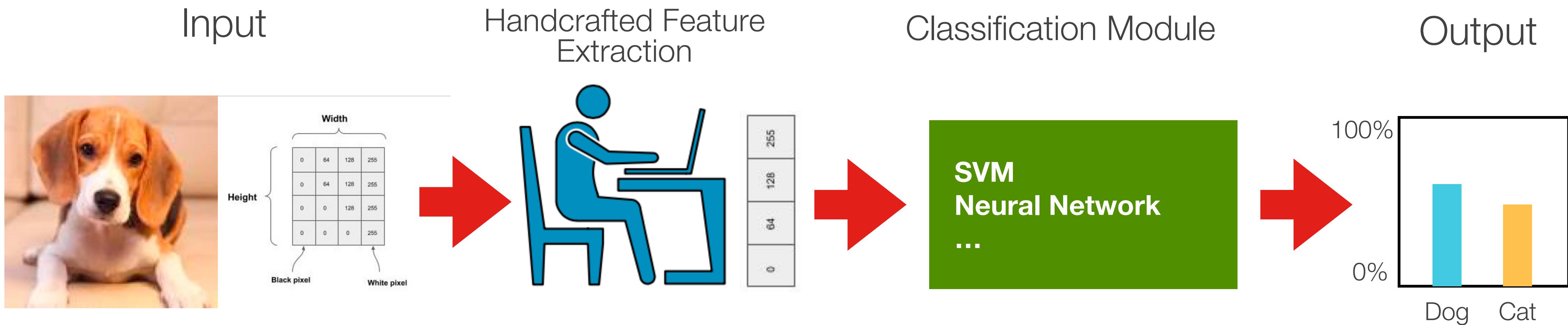


Histogram and oriented gradients



The “old days:” Feature Engineering

- Machine learning models are only as good as the features you provide them with
- To figure out which features you should use for a specific problem
 - rely on domain knowledge (or partner with domain experts)
 - experiment to create features that make machine learning algorithms work better



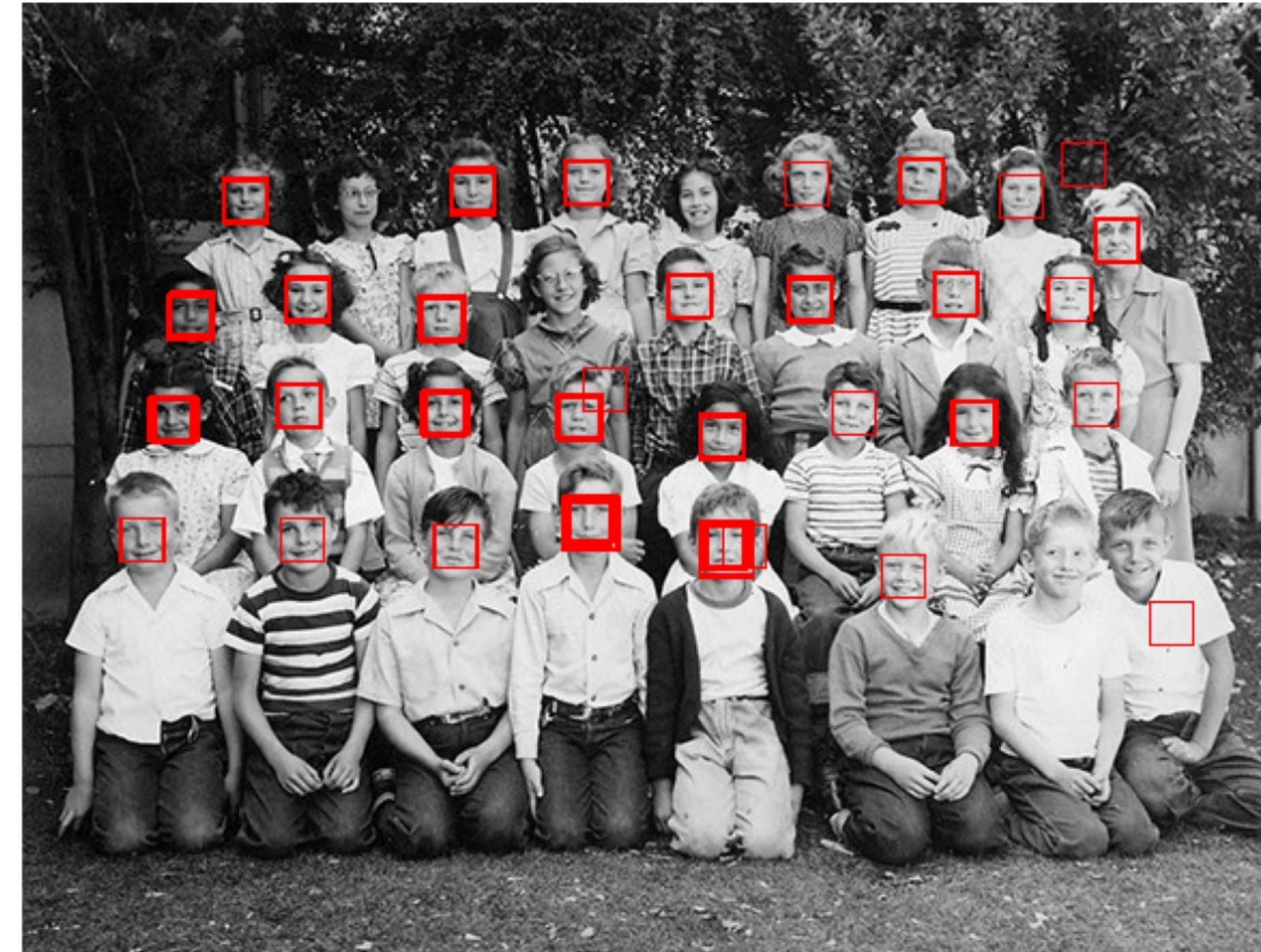
Performance

Object Detection (~2007)



Felzenszwalb, Ramanan, McAllester. A Discriminatively Trained, Multiscale, Deformable Part Model. CVPR 2008 ([DPM v1](#))

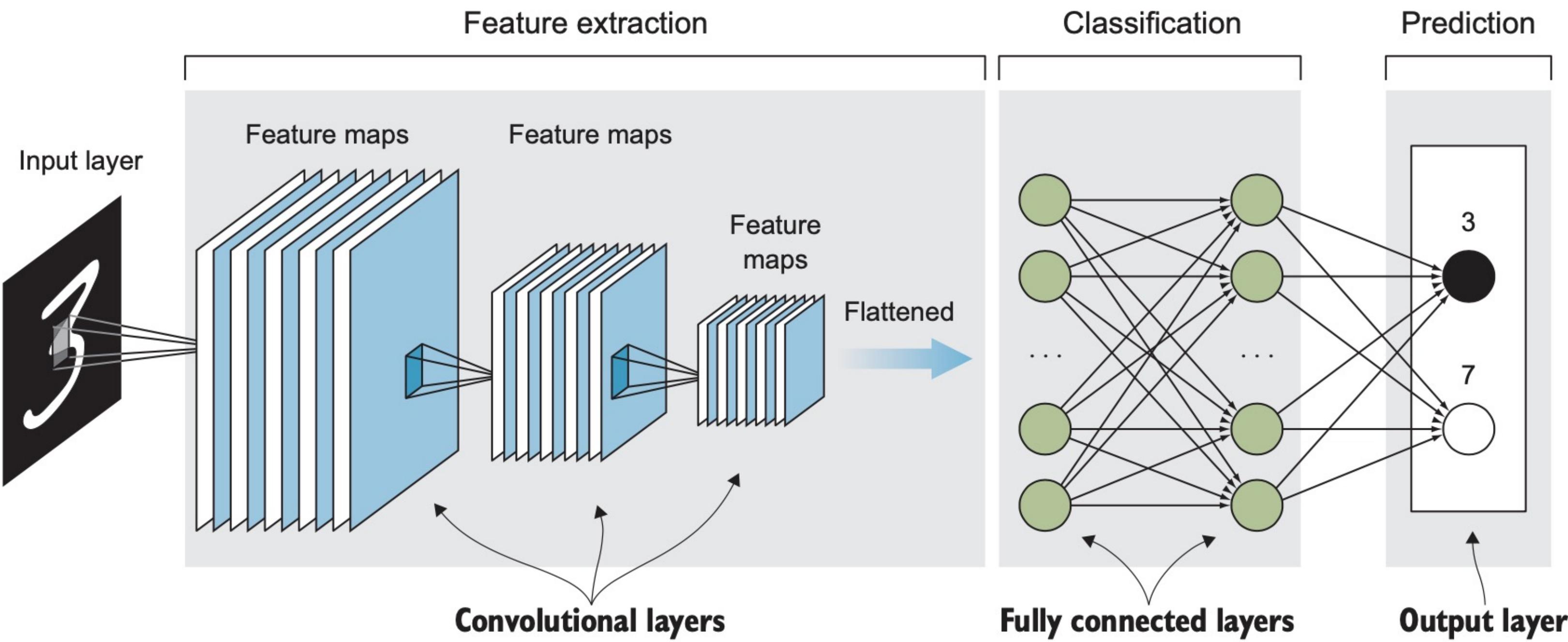
Face Detection (~2013)



<https://github.com/alexdemartos/ViolaAndJones>

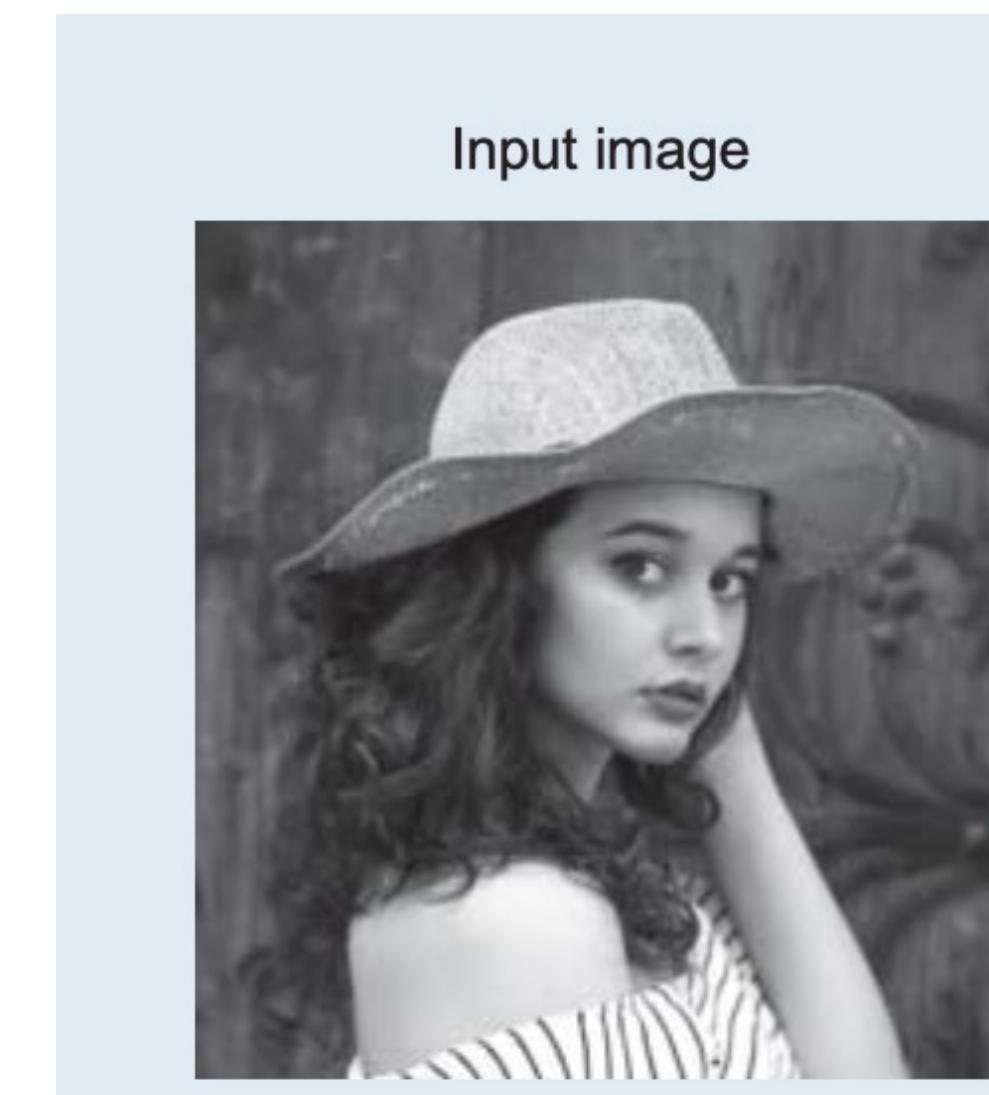
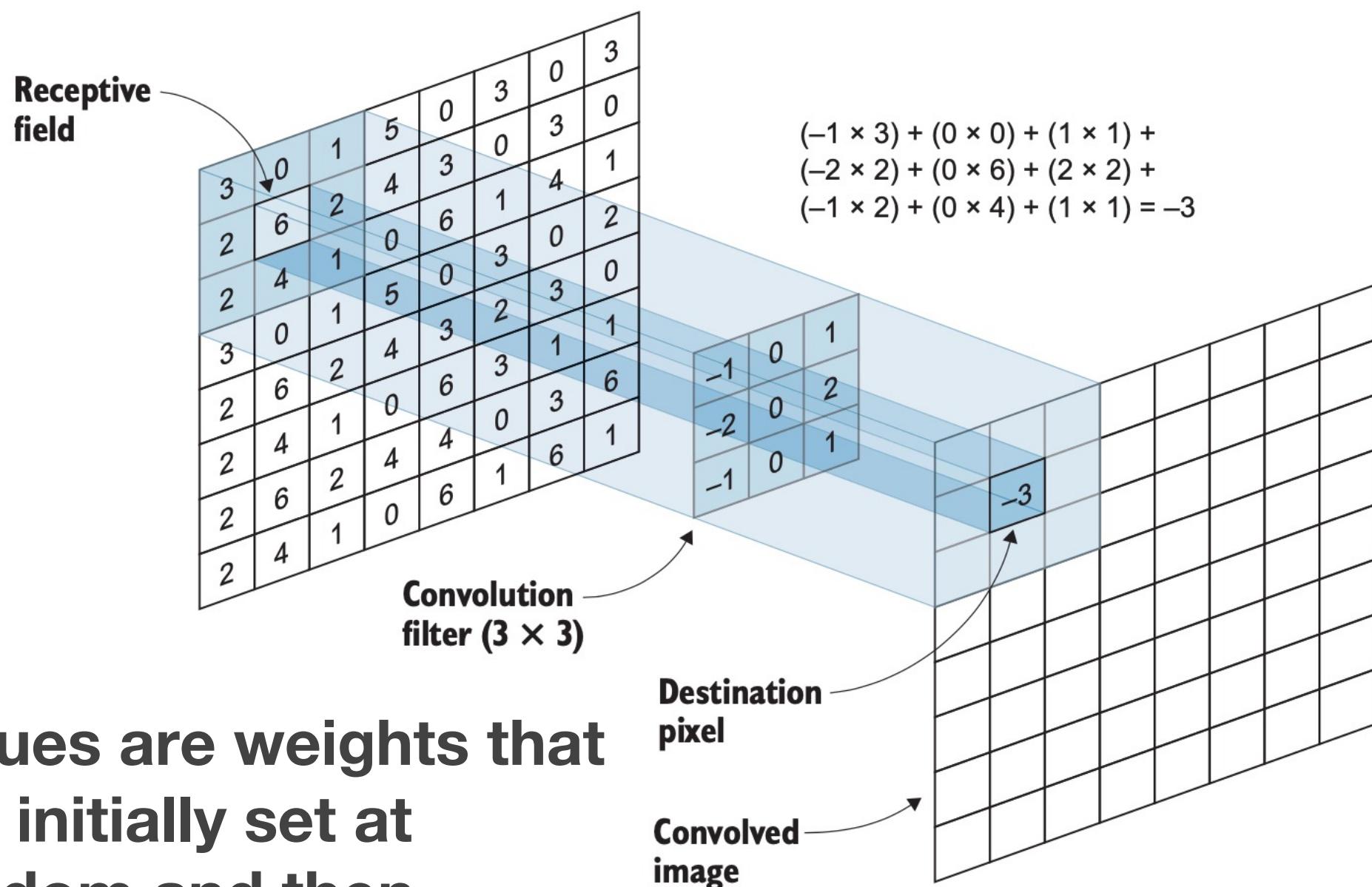
Credits: Ross Girshick (Facebook AI Research)

Convolutional Neural Networks



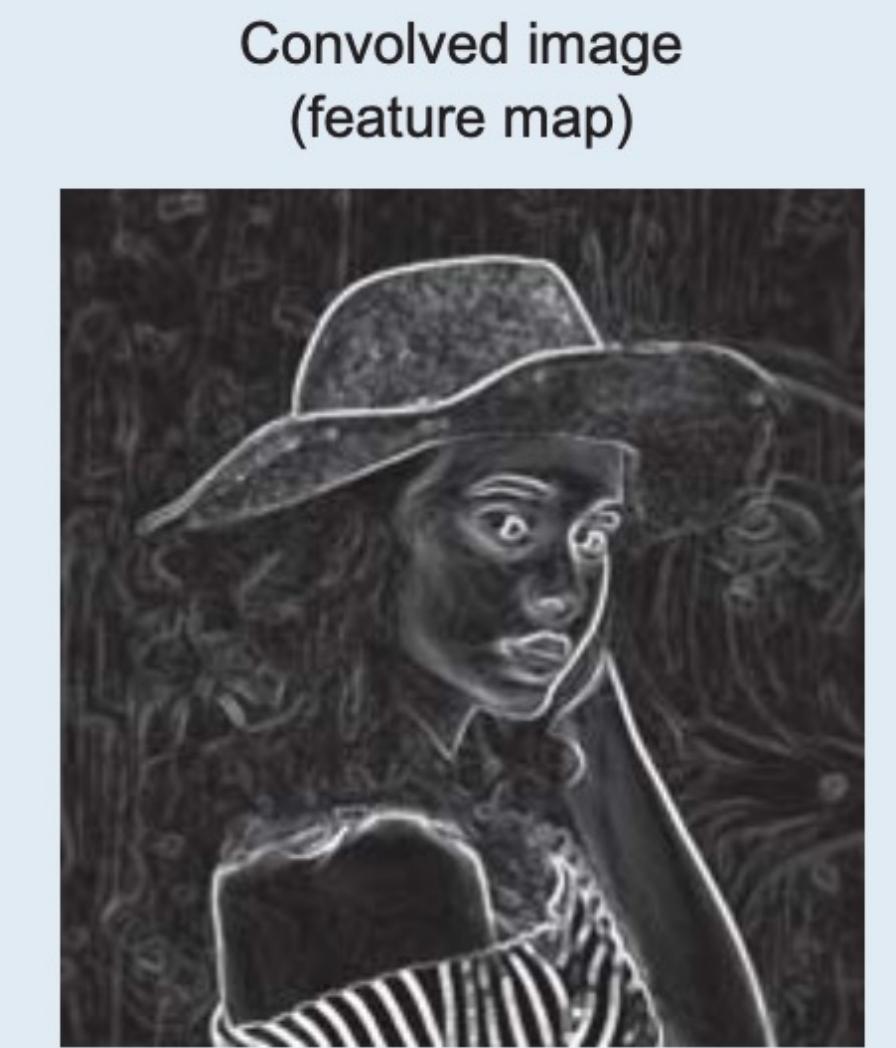
- CNNs exploit image properties to drastically reduce the number of model parameters
- Feature maps
 - **Automatically** extracted hierarchical
 - Retain spatial association between pixels
- **Translation invariance**
 - a dog is a dog even if its image is shifted by a few pixels
- Local interactions
 - all processing happens within very small image windows
 - within each layer, far-away pixels cannot influence nearby pixels

Convolution & Feature Maps



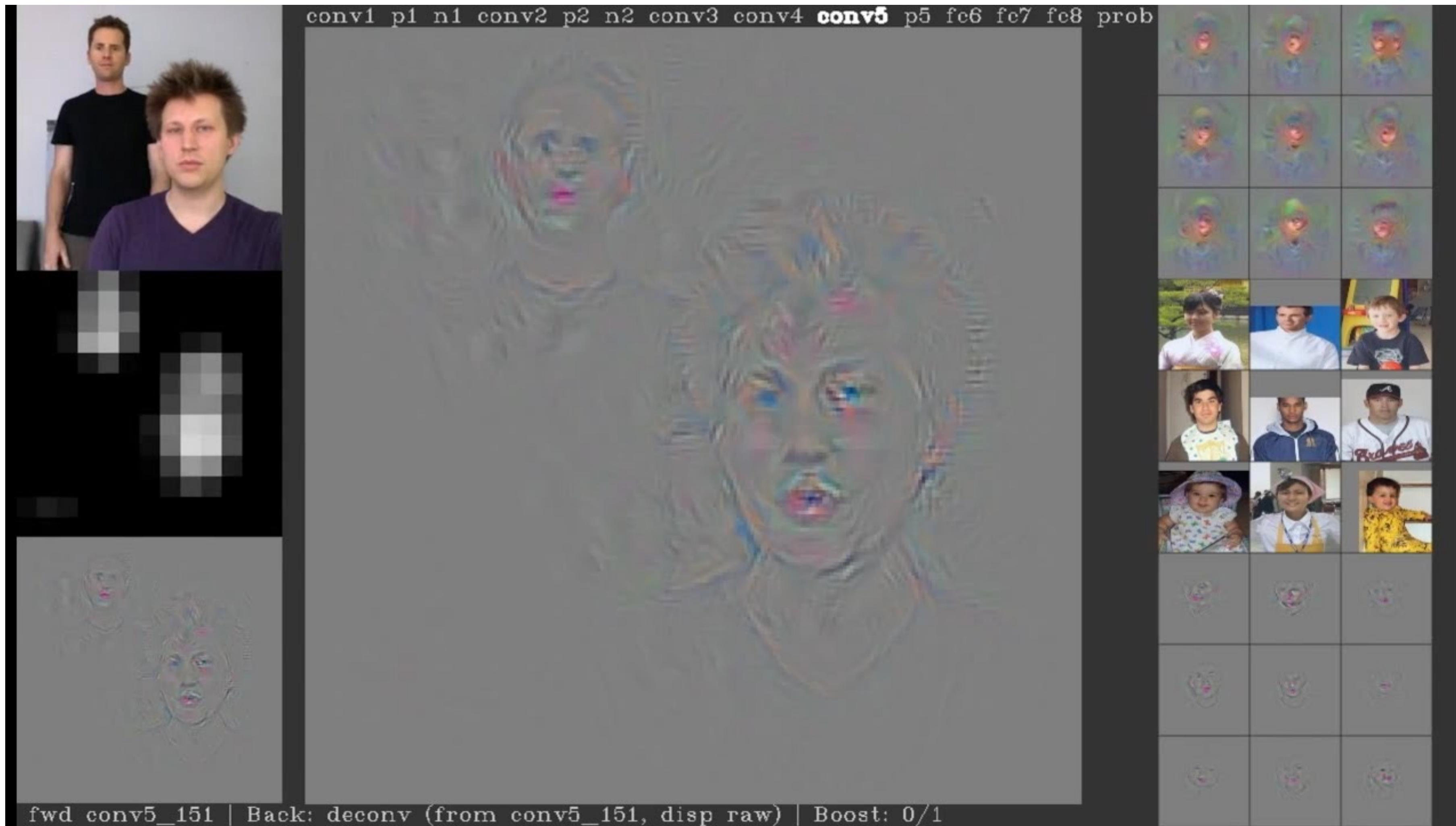
Convolution kernel with optimized weights

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix} * \begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix} =$$

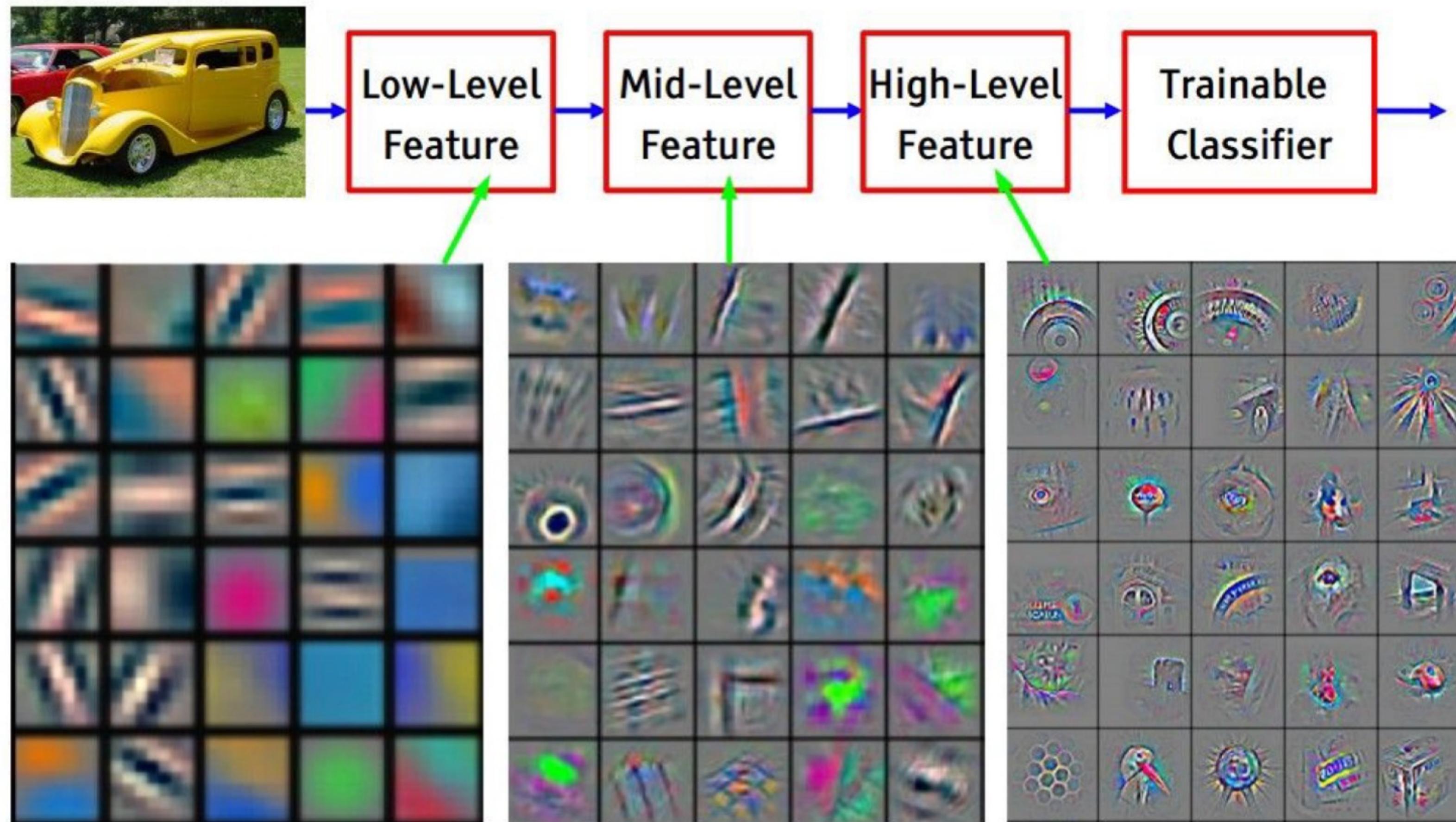


- Try this
<https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html>

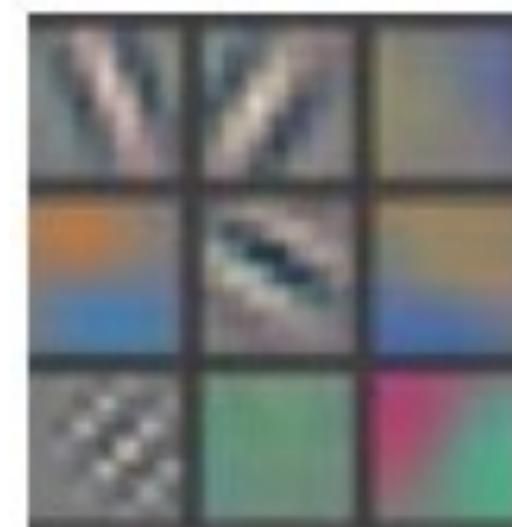
What do CNN learn?



Feature Visualisation



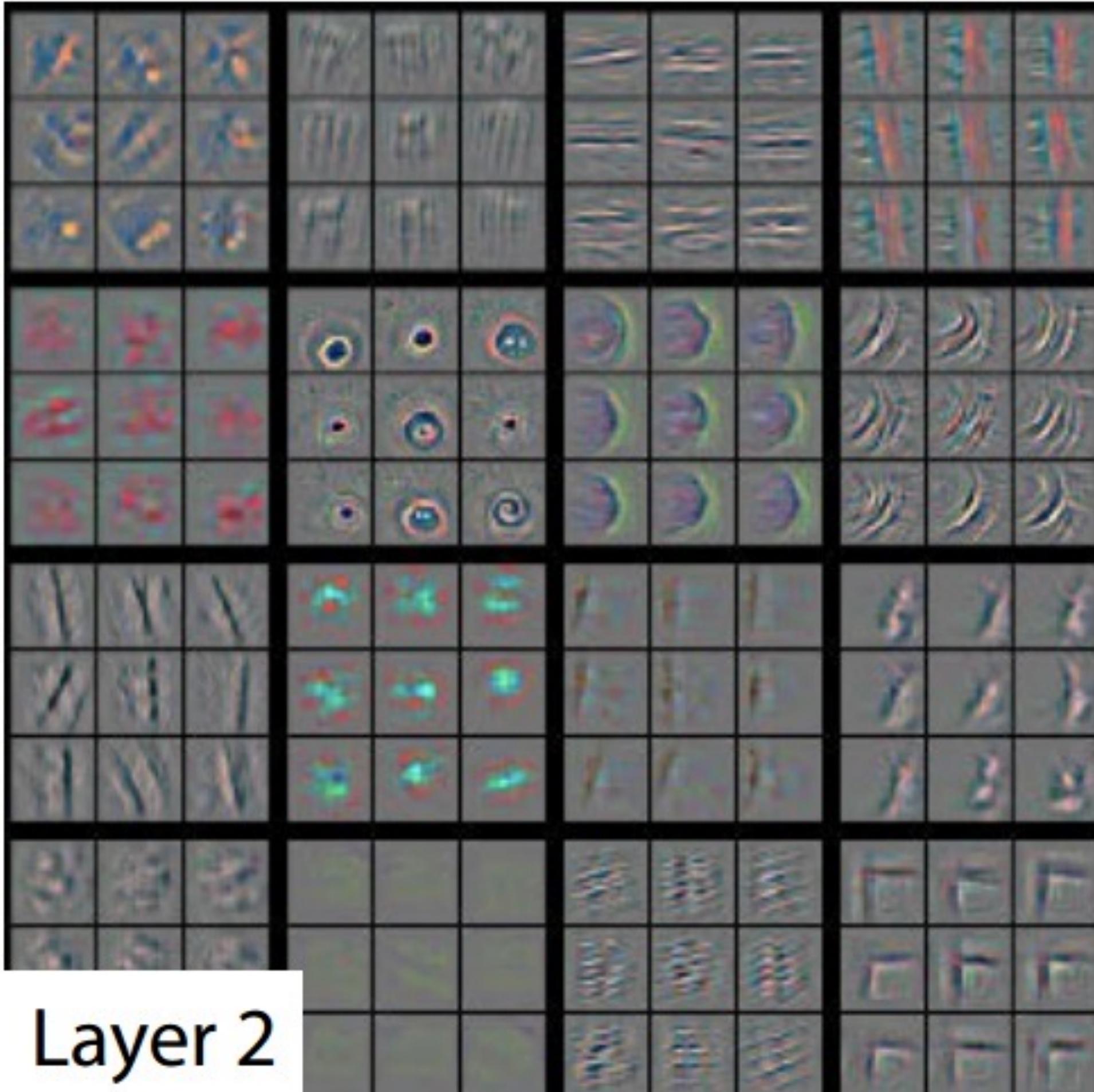
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



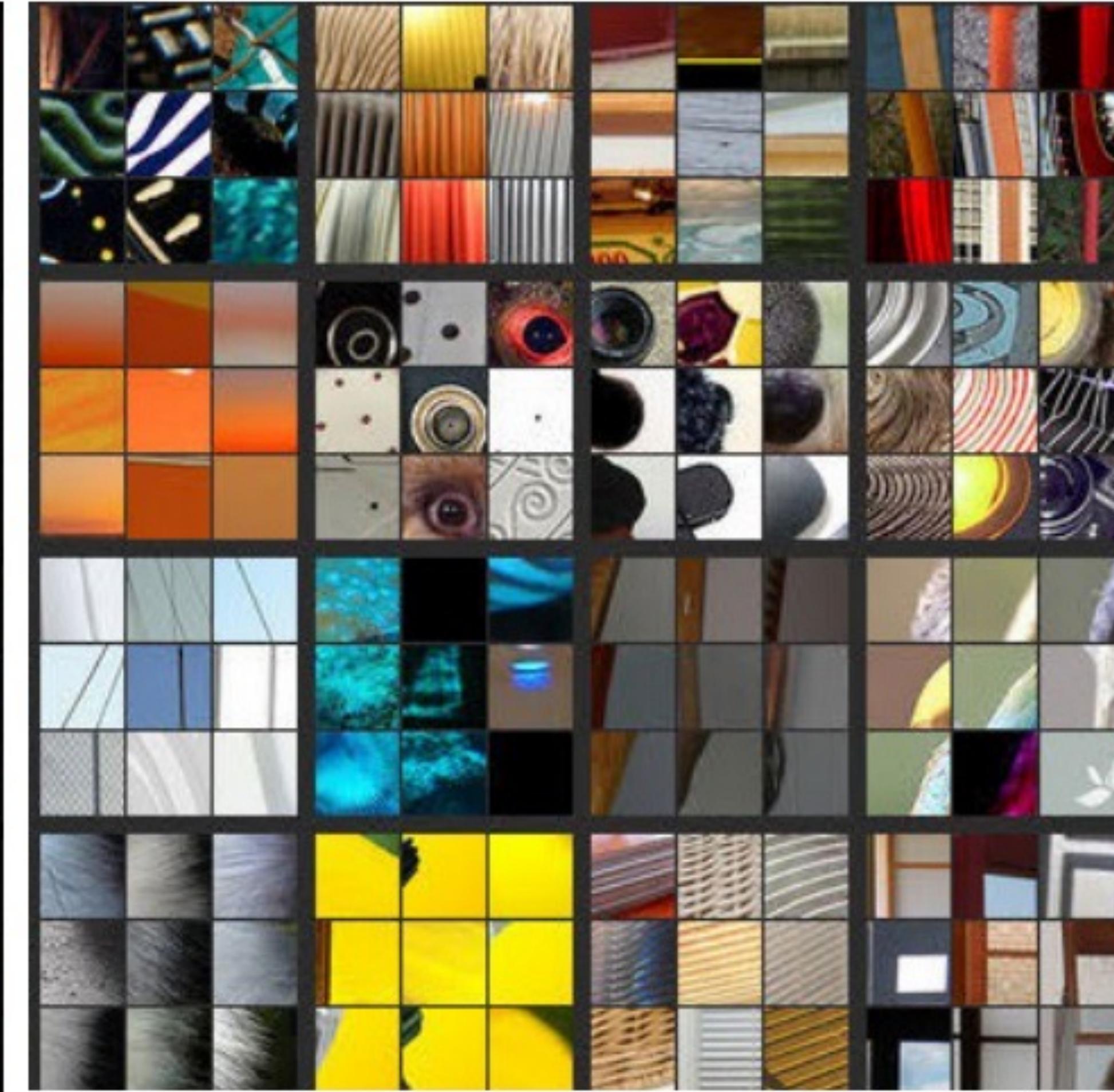
Layer 1



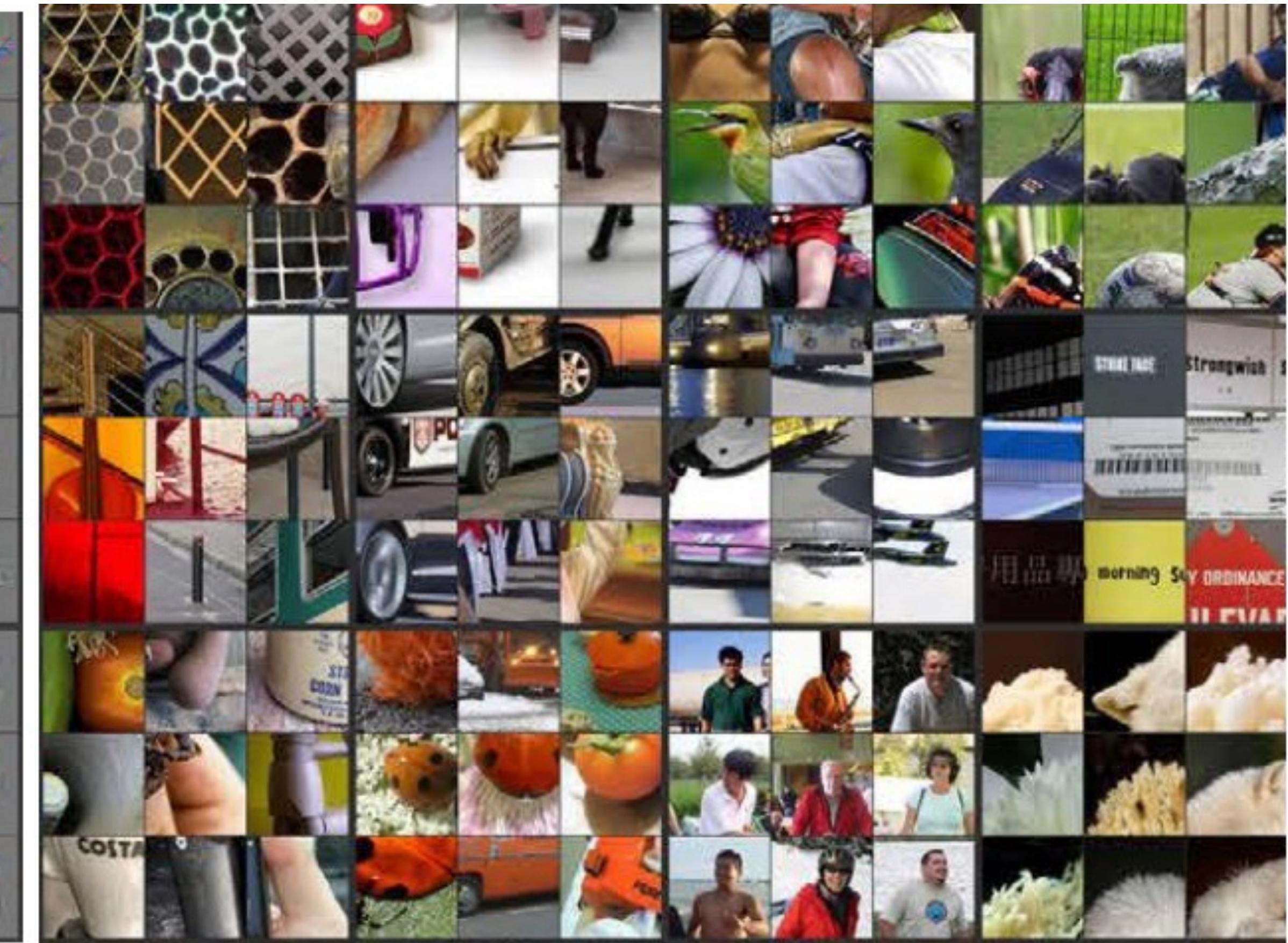
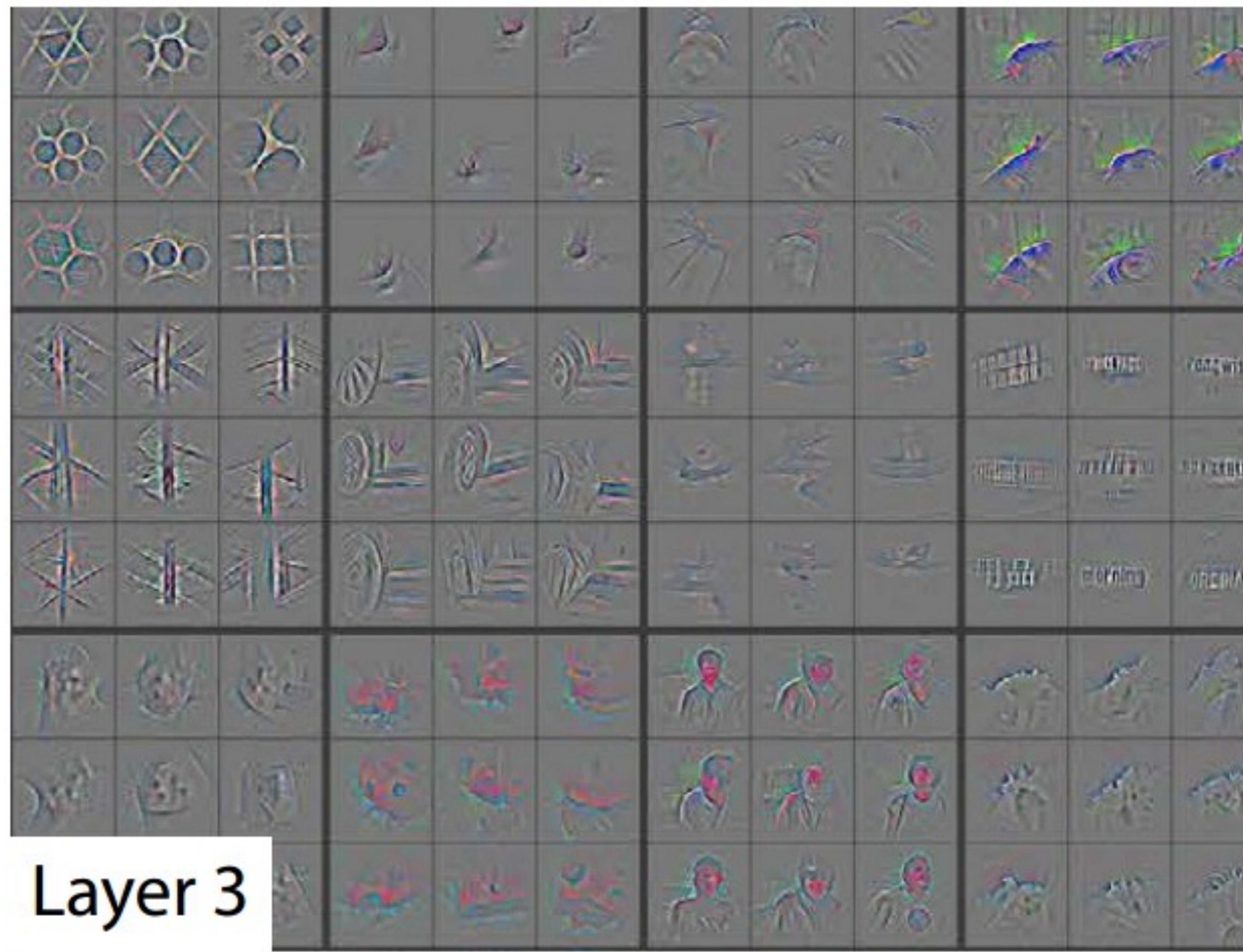
Visualizing and Understanding Convolutional Network. Zeiler and Fergus, ECCV 2014



Layer 2



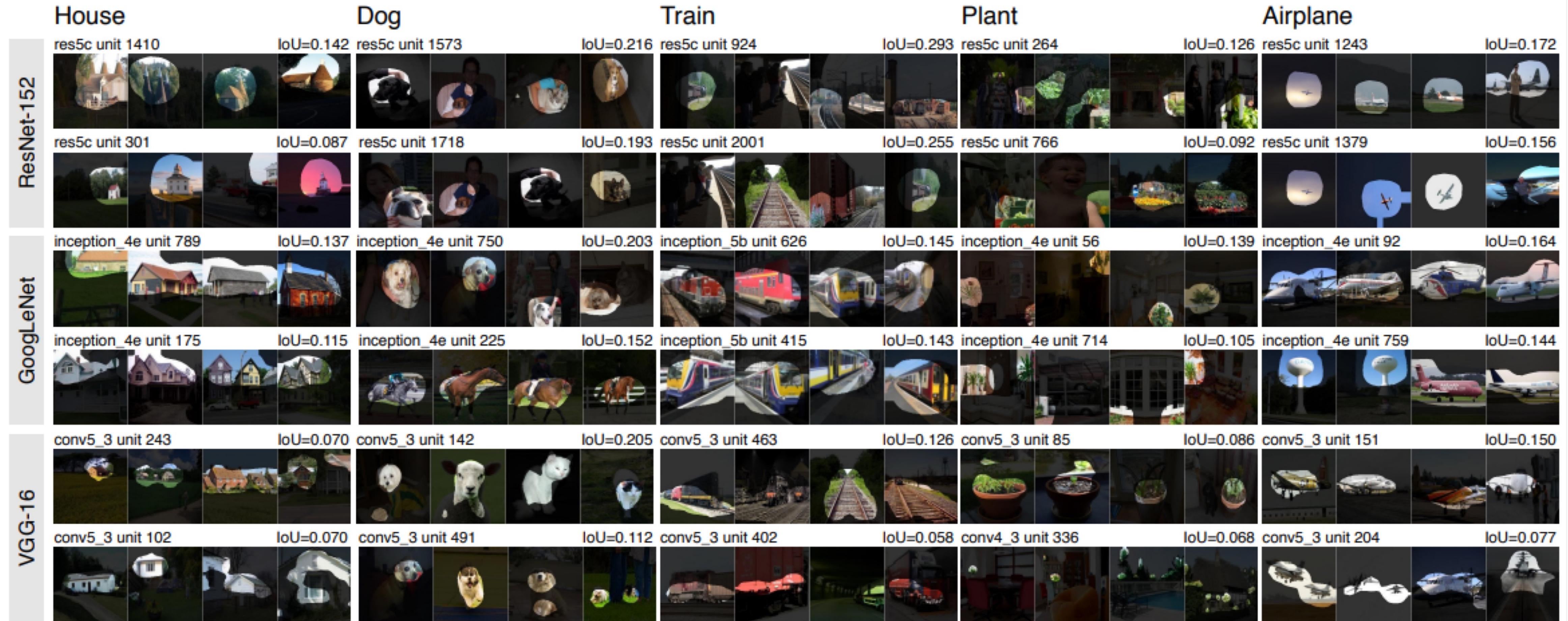
Visualizing and Understanding
Convolutional Network.
Zeiler and Fergus, ECCV 2014



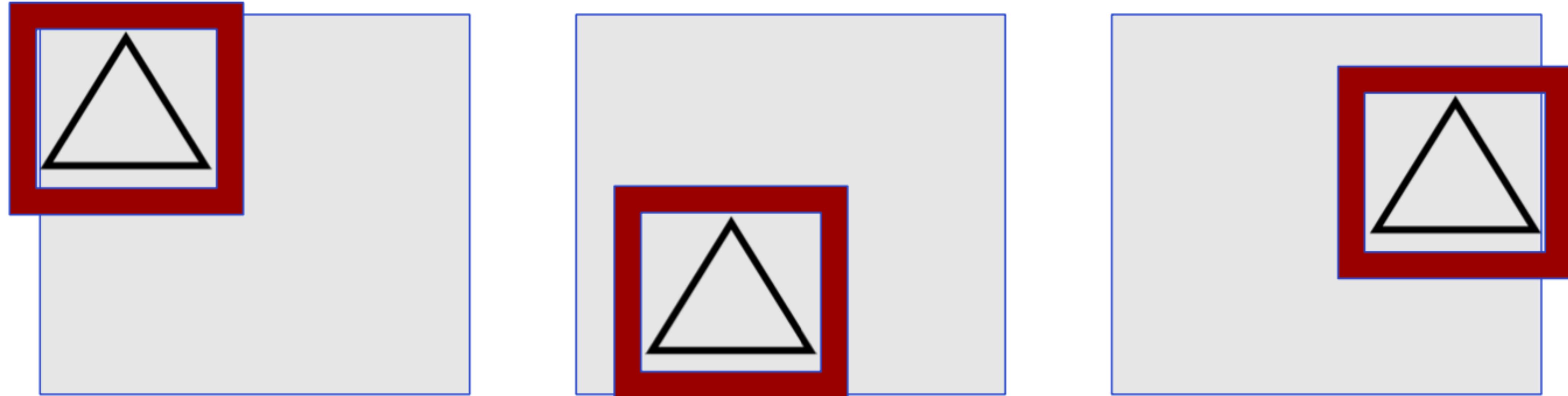
Layer 3

Visualizing and Understanding Convolutional Network. Zeiler and Fergus, ECCV 2014

Network dissection



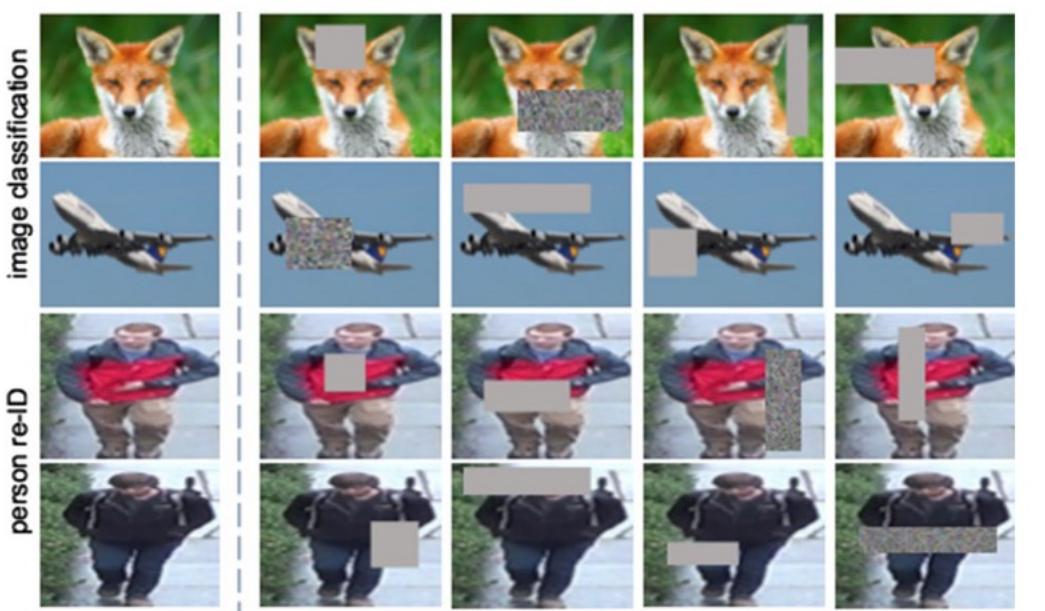
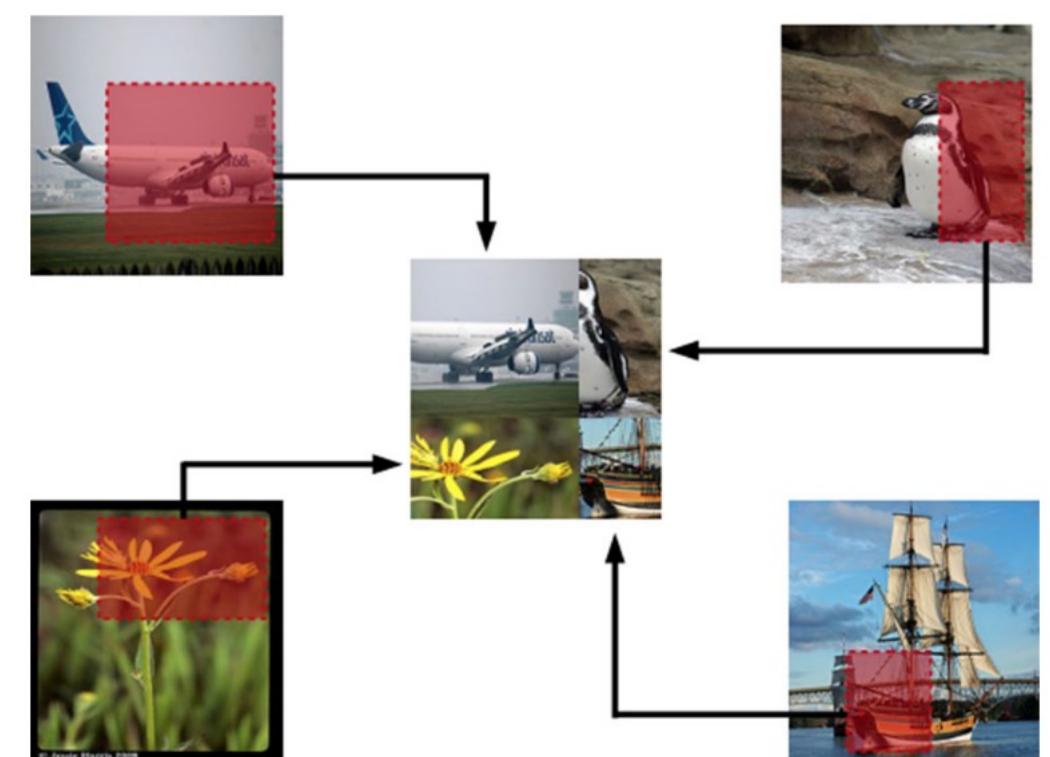
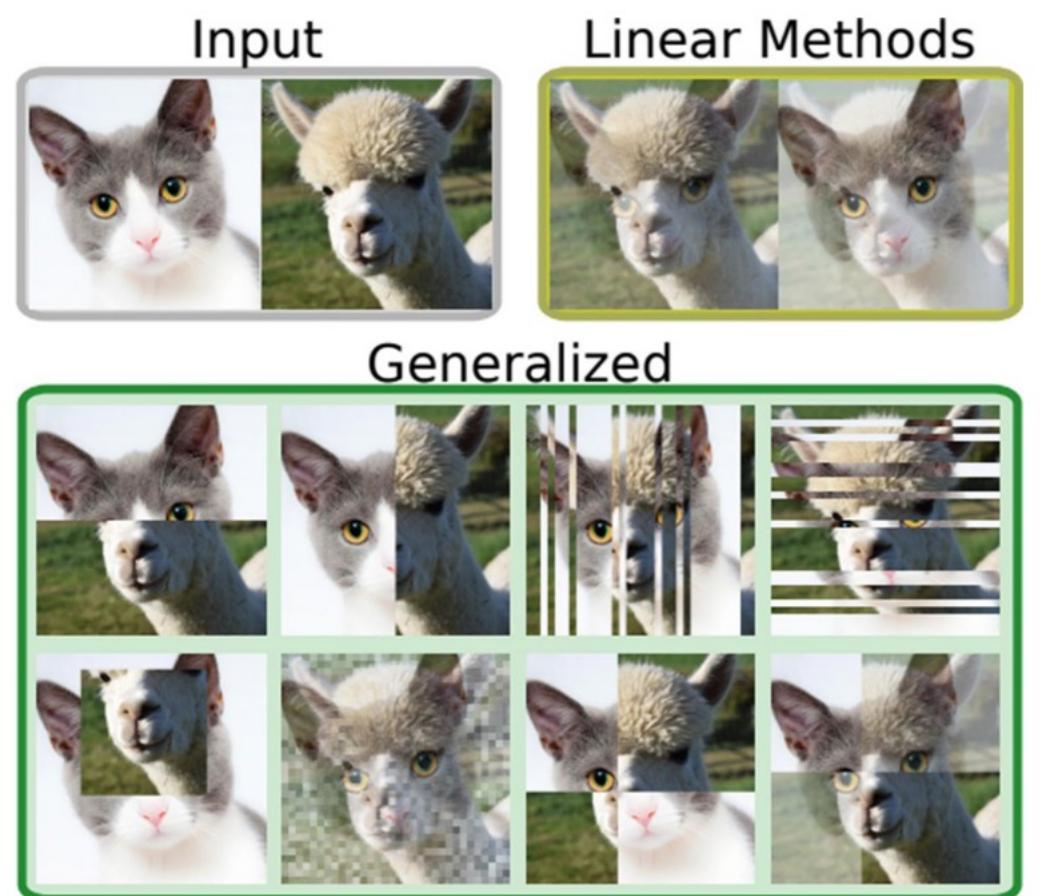
Translation Invariance



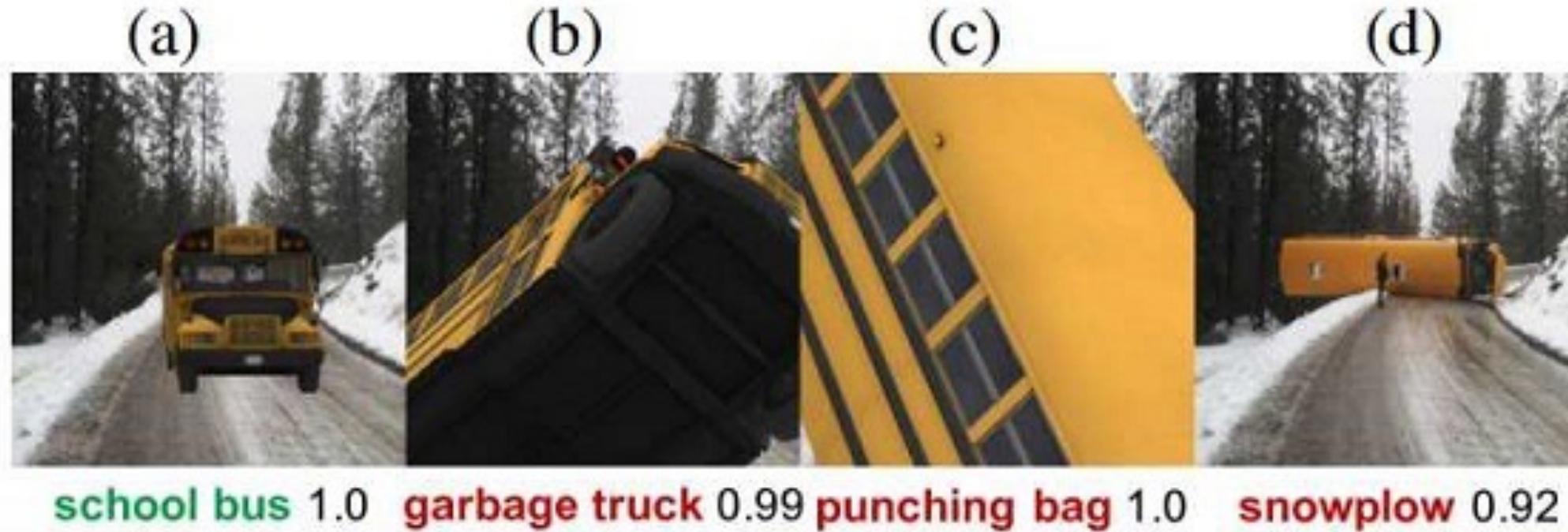
- But not rotation and scaling invariance!

Data Augmentation

- Generate variations of the input data, to improve generalisability (out of distribution inputs)
 - Improve invariance (rotation, scaling, distortion)
- Geometric
 - Flipping
 - Color space
 - Cropping
 - Rotation
 - Translation
 - Noise Injection
- Color space transformation
- Mixing Images
- Random erasing
- Adversarial training
- GAN-based image generation



Robustness to input variation

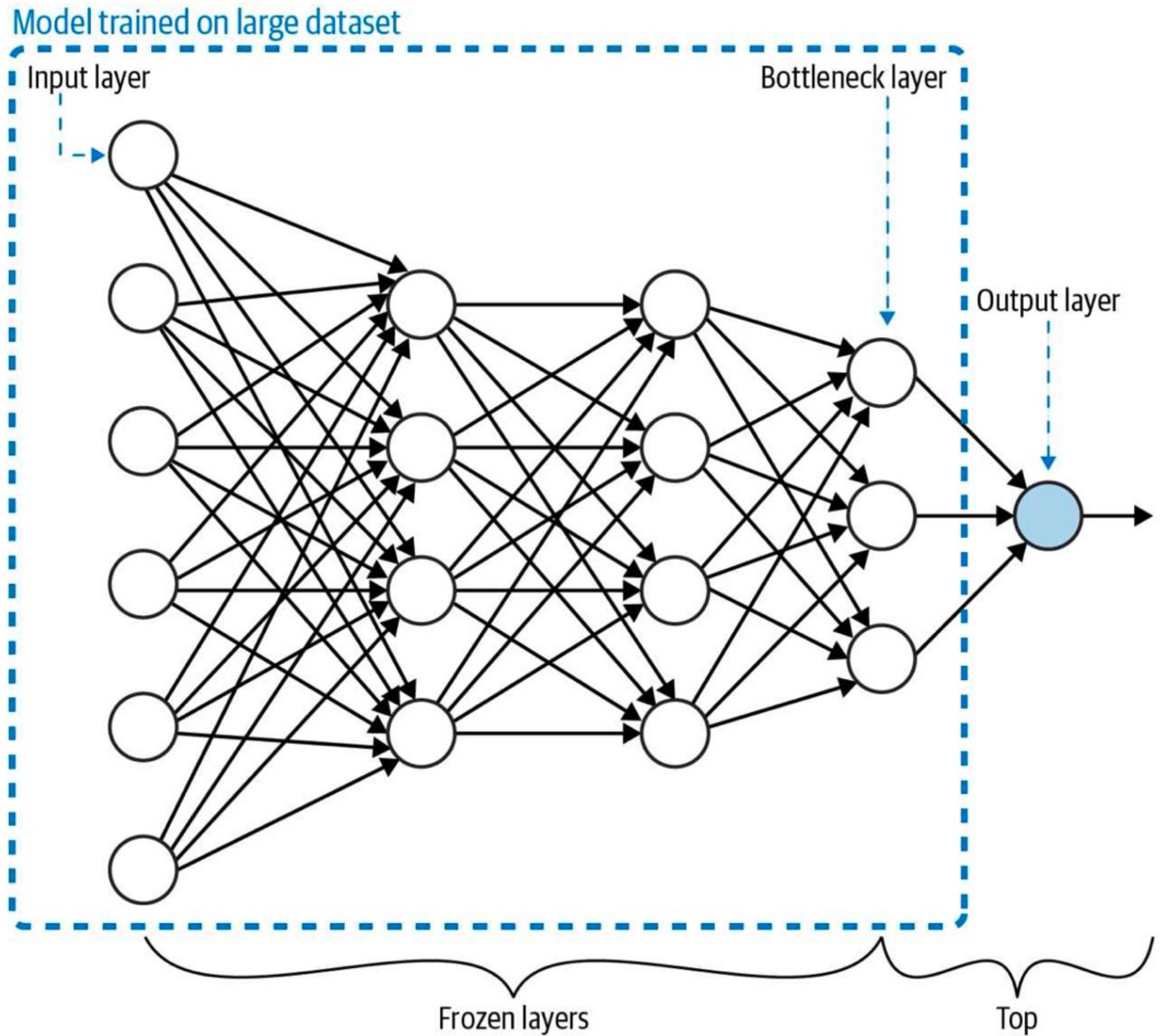


Strike (with) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects.
Alcorn et al. 2019

<https://arxiv.org/pdf/1811.11553.pdf>



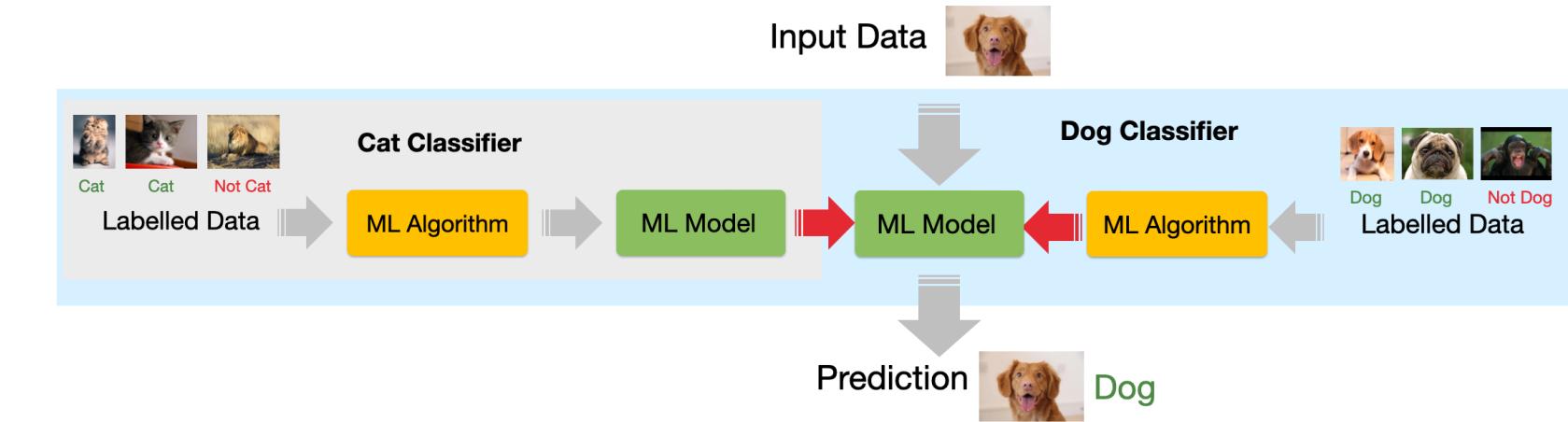
Transfer Learning



Transfer Learning

Reuse a model trained for one task is re-purposed (tuned) on a different but related task

Useful in tasks lacking abundant data

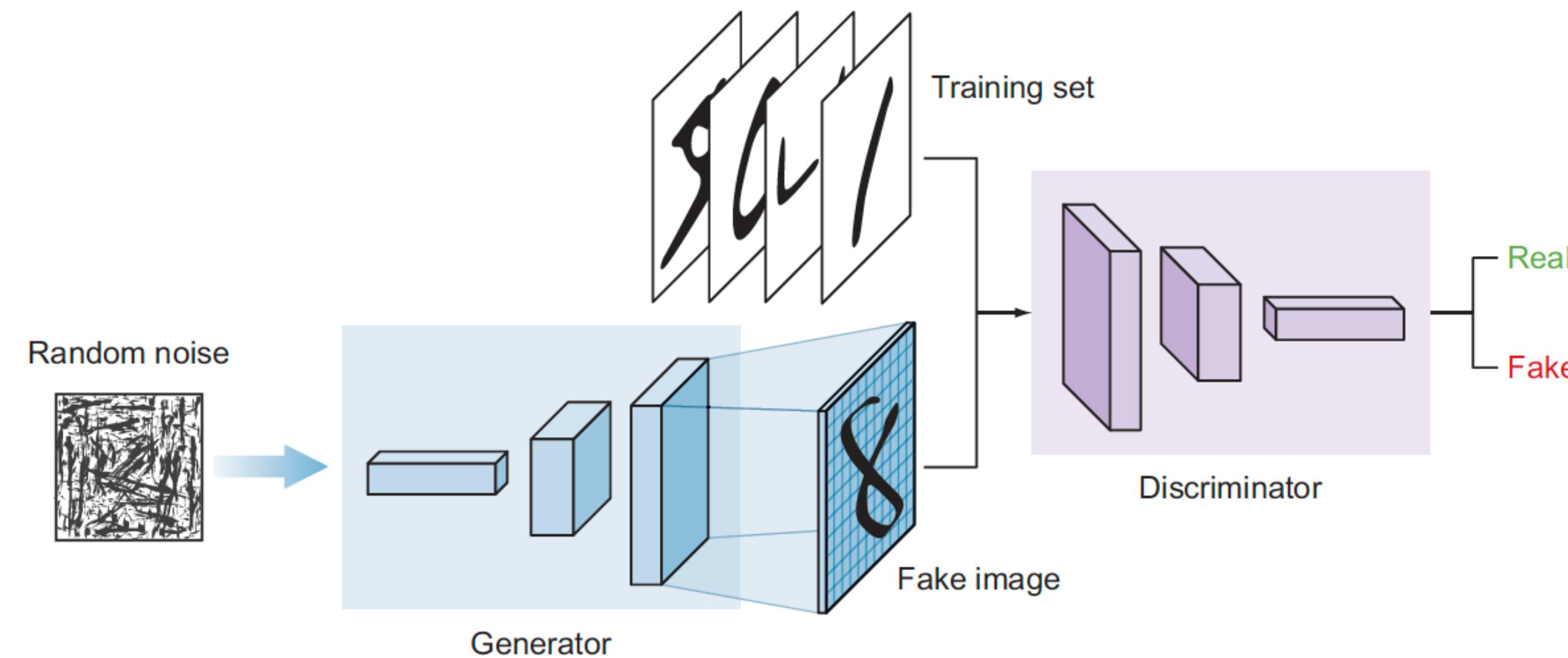


- **Problem:** training custom ML models requires extremely large datasets
- **Transfer learning:** take a model that has been trained on the **same type of data for a similar task** and apply it to a specialised task using our own custom data.
 - **Same data:** same data modality. same types of images (e.g. professional pictures vs. Social media pictures)
 - **Similar tasks:** if you need a new object classification model, use a model pre-trained for object classification

Advanced Computer Vision Techniques

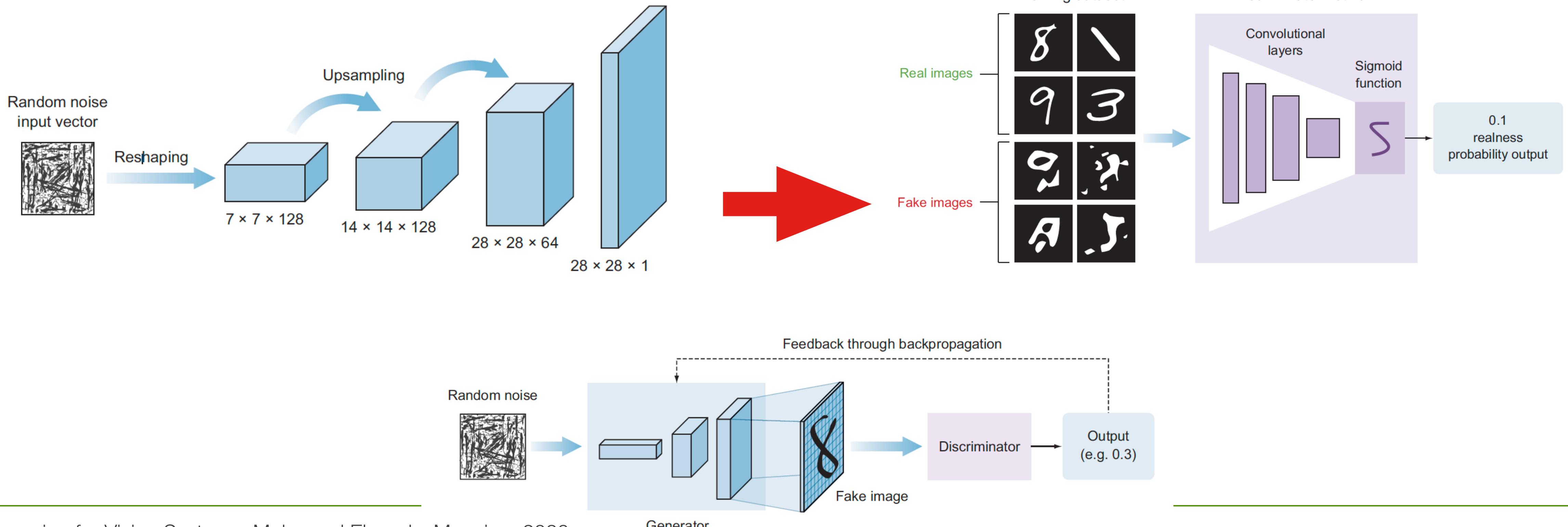
Generative Adversarial Networks

- Learn patterns from the training dataset and create new images that have a similar distribution of the training set
- Two deep neural networks that compete with each other
 - The generator tries to convert random noise into observations that look as if they have been sampled from the original dataset
 - The discriminator tries to predict whether an observation comes from the original dataset or is one of the generator's forgeries



Generative Adversarial Networks

- The generator's architecture looks like an inverted CNN that starts with a narrow input and is upsampled a few times until it reaches the desired size
- The discriminator's model is a typical classification neural network that aims to classify images generated by the generator as real or fake



Which face is real? - <https://www.whichfaceisreal.com/>

PLAY

ABOUT METHODS LEARN PRESS CONTACT BOOK CALLING BS

Click on the person who is real.



- Try this

<https://thispersondoesnotexist.com/>

Image super-resolution GAN

- A good technical summary

<https://blog.paperspace.com/image-super-resolution/>



<https://newatlas.com/super-resolution-weizmann-institute/23486/>

Synthetic Video Generation

The screenshot displays the Synthesia video generation interface. At the top, there is a large preview window showing a woman with long dark hair, wearing a light pink button-down shirt, standing against a plain white background. In the bottom left corner of this preview window, the Synthesia logo (a blue hexagon icon) and the word "synthesia" are visible. Below the preview window, a large text block reads: "Say goodbye to cameras, microphones and actors!". Underneath this text, a smaller line of text says: "Create professional AI videos from text in 60+ languages.". To the right of the preview window, the main interface shows a video player window with a woman's face, an "Unmute video" button at the bottom, and a "Type your video title here" input field above it. To the right of the video player, there are several sections: "Select avatar, size and alignment" with a grid of nine avatars (three rows of three), and categories for "Avatar", "Music", "Background", "Text", and "Shapes". On the far left, there are some icons and a small preview of the video frame.

Synthesia

Say goodbye to cameras, microphones and actors!

Create professional AI videos from text in 60+ languages.

Type your video title here

Cancel Generate video

Select avatar, size and alignment

Anna Mia Vincent

Gloria Jonathan Sofia

Avatar Music Background Text Shapes

Unmute video

Text-to-image Generation

TEXT PROMPT an illustration of a baby daikon radish in a tutu walking a dog

AI-GENERATED IMAGES



Edit prompt or view more images↓

TEXT PROMPT an armchair in the shape of an avocado....

AI-GENERATED IMAGES



Edit prompt or view more images↓

- <https://openai.com/blog/dall-e/>



- ML-generated painting sold for \$432,500
- The network trained on a dataset of 15,000 portraits painted between the fourteenth and twentieth centuries
- Network “learned” the style, and generated a new painting

Neural Style Transfer



Content Image

Style Image

Stylized Result

<https://replicate.com/rinongal/stylegan-nada>

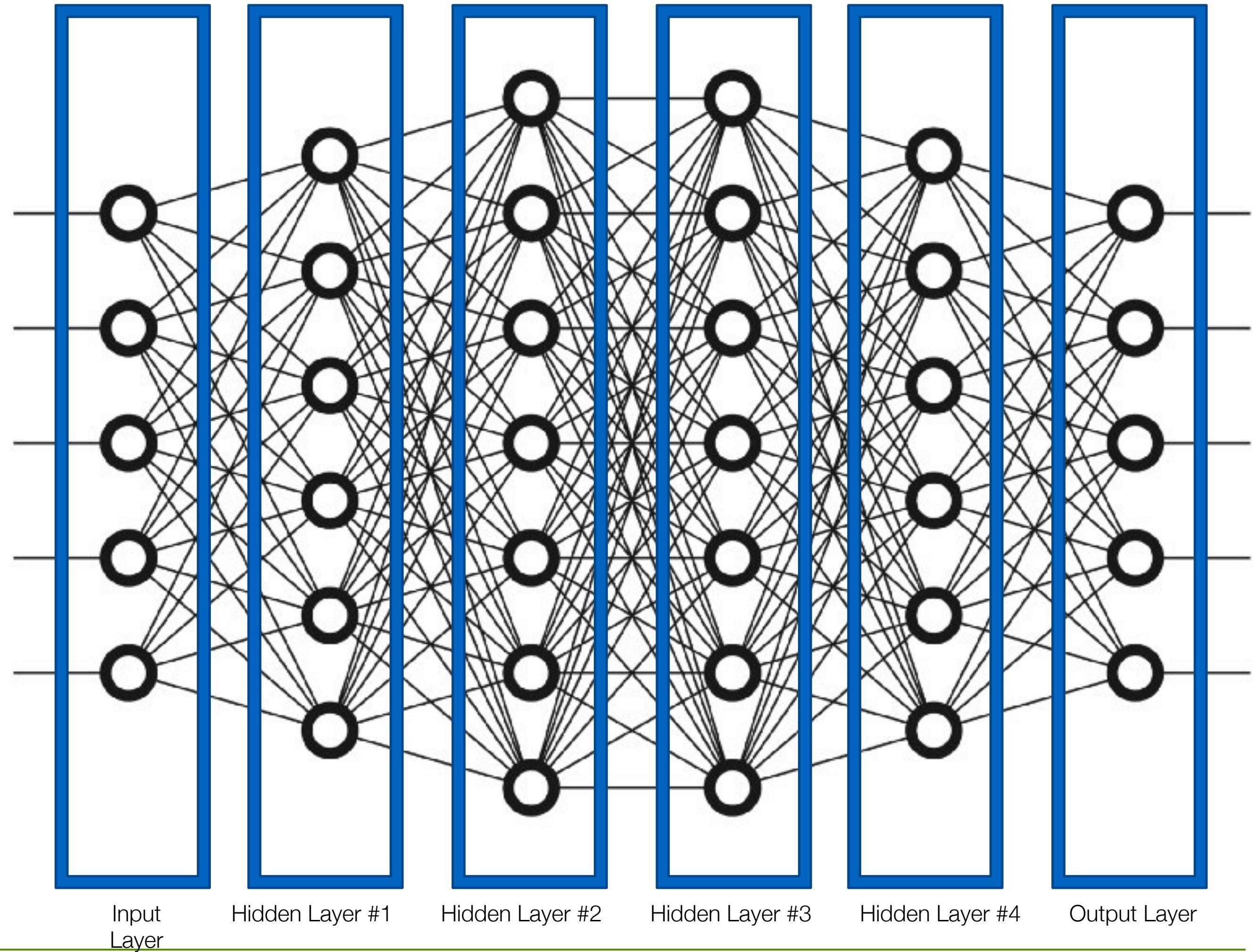
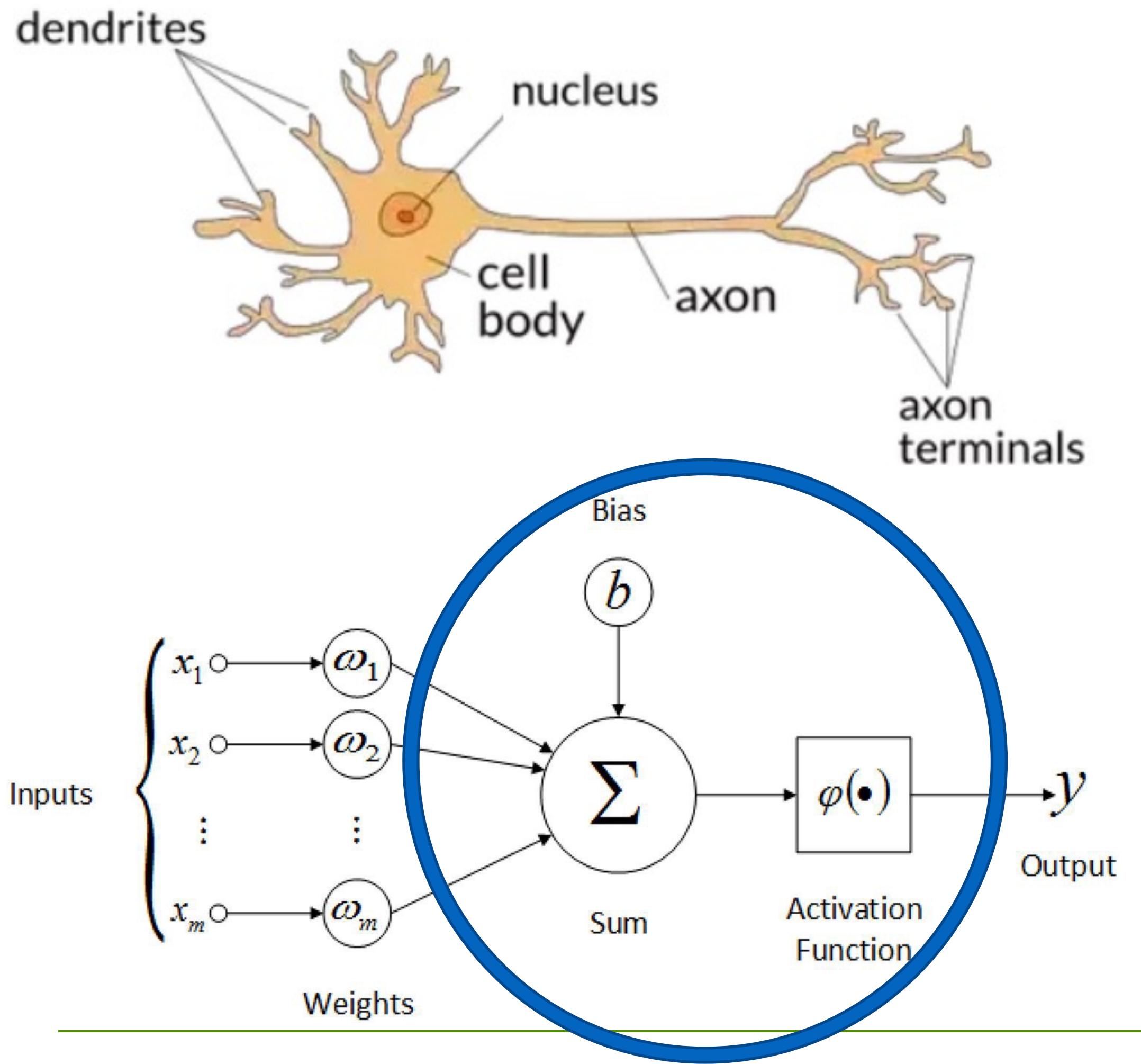


Deepfakes

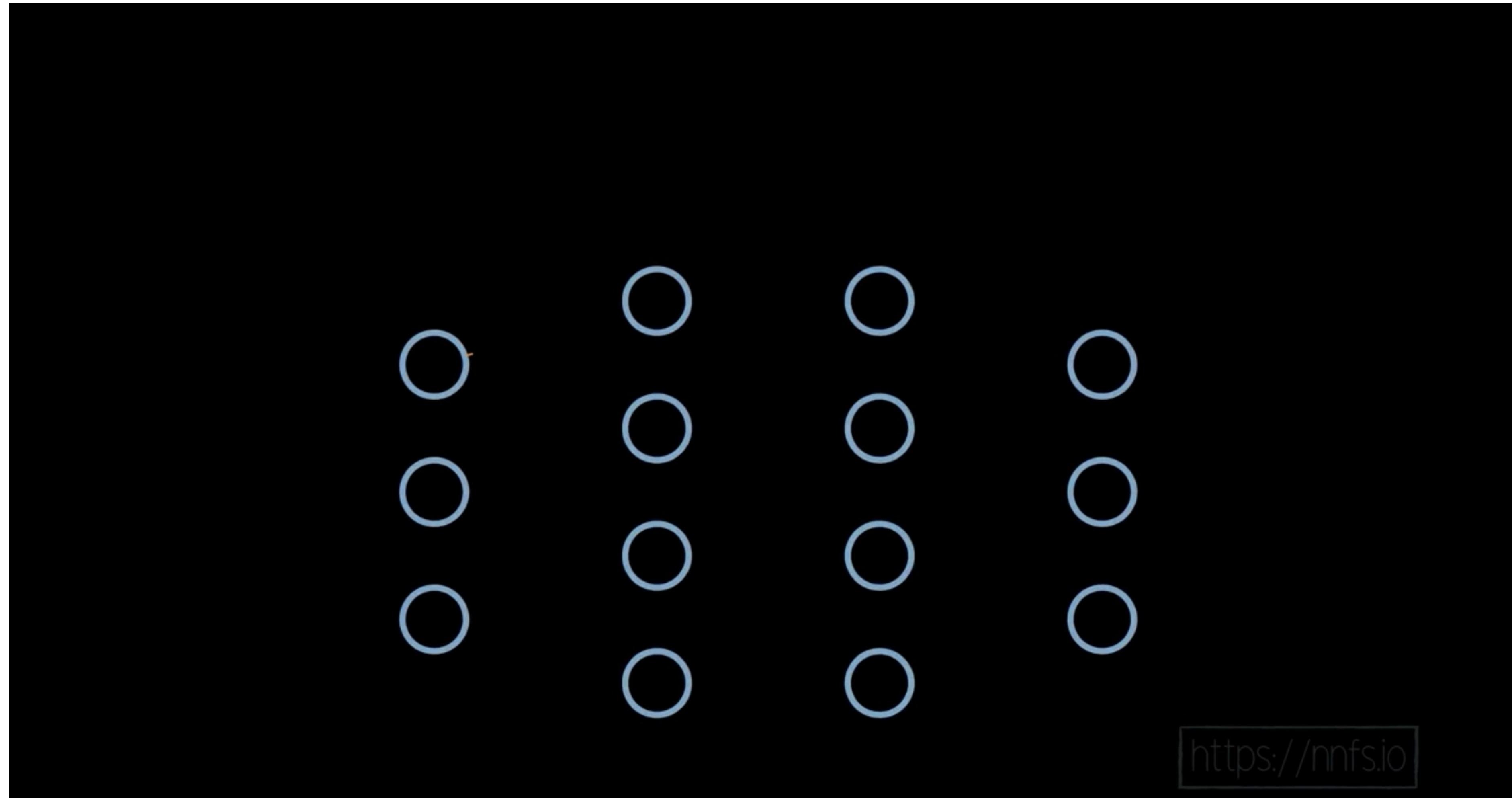


Advanced: Backpropagation in DNNs

From a Neuron to a Deep Neural Network (DNN)



How exactly does a Neuron in a DNN work?

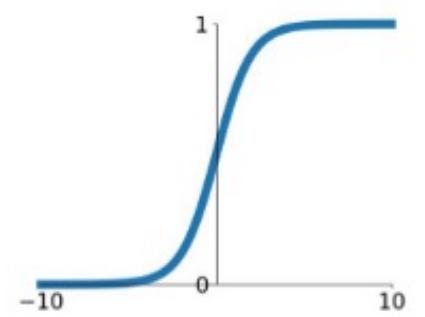


Source: “What is backpropagation really doing?”—YouTube walk-through (see last slide)

Activation Functions for Forward Propagation

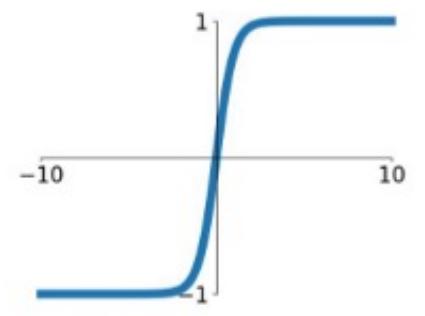
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



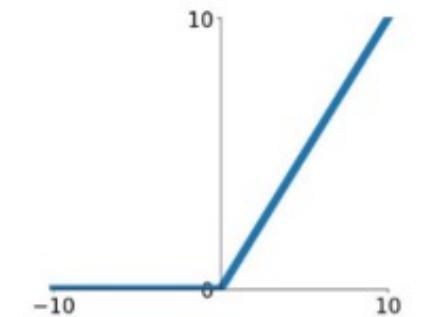
tanh

$$\tanh(x)$$



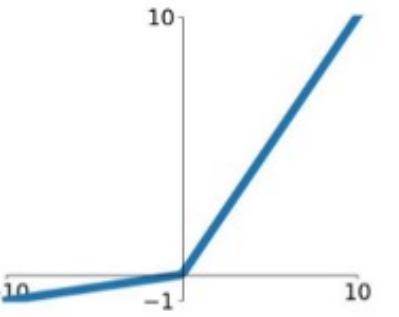
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

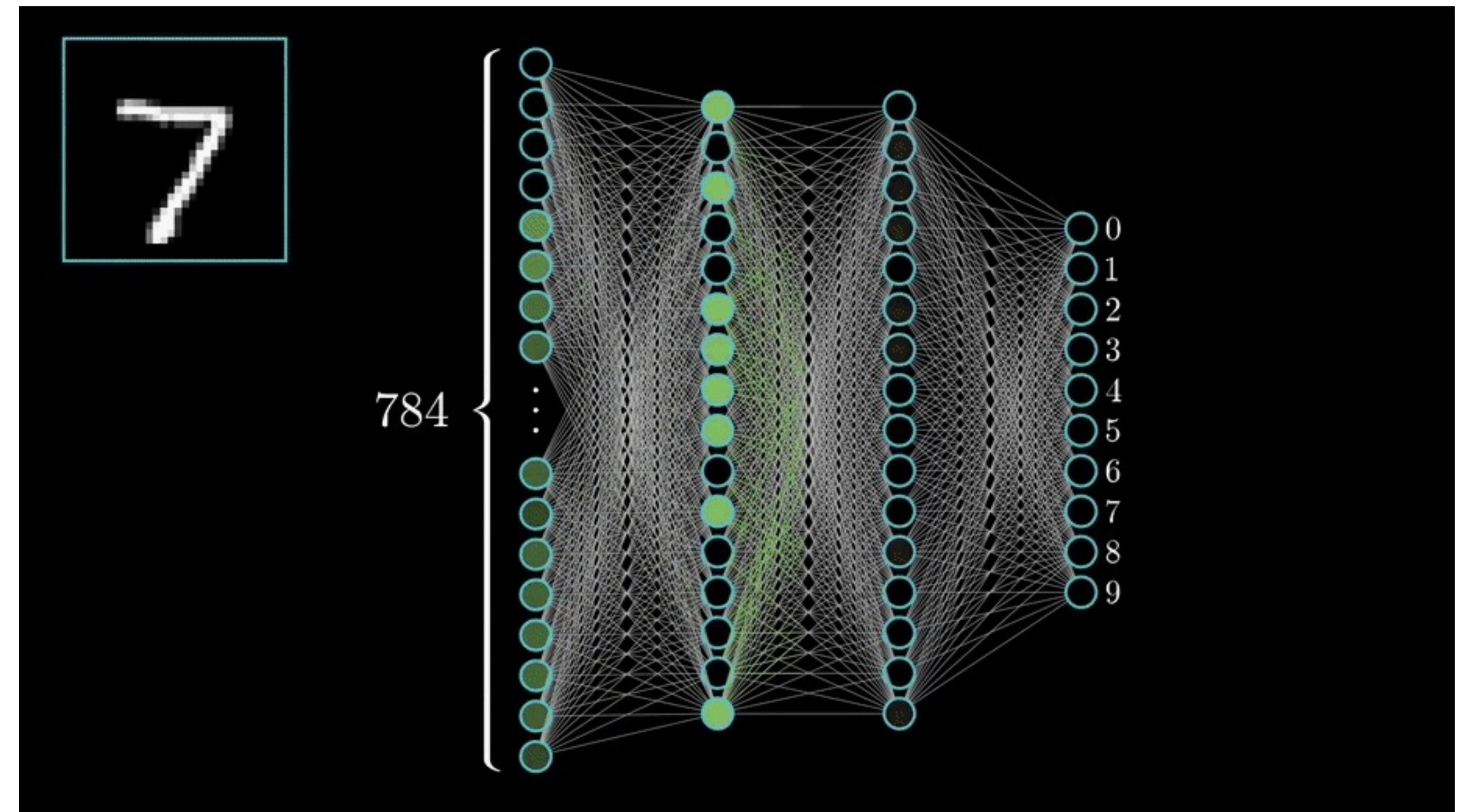
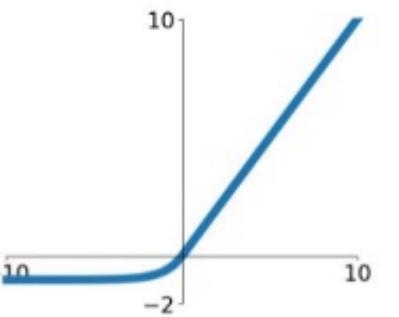


Maxout

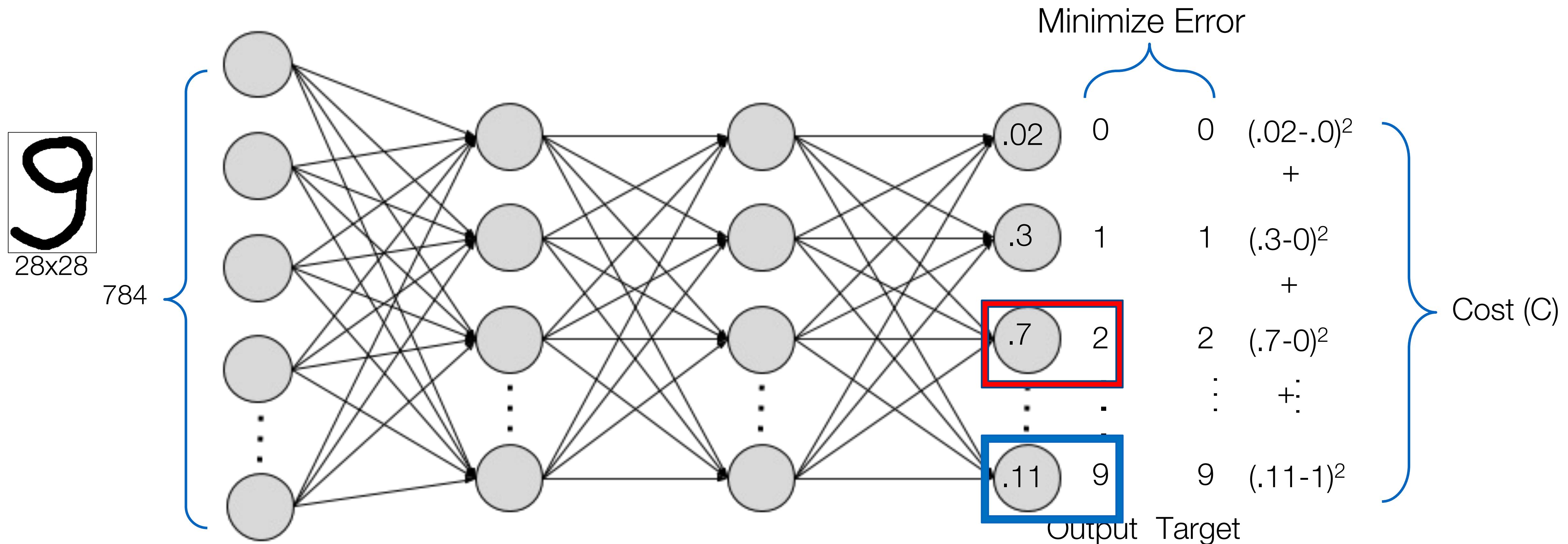
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



But how does a DNN learn?



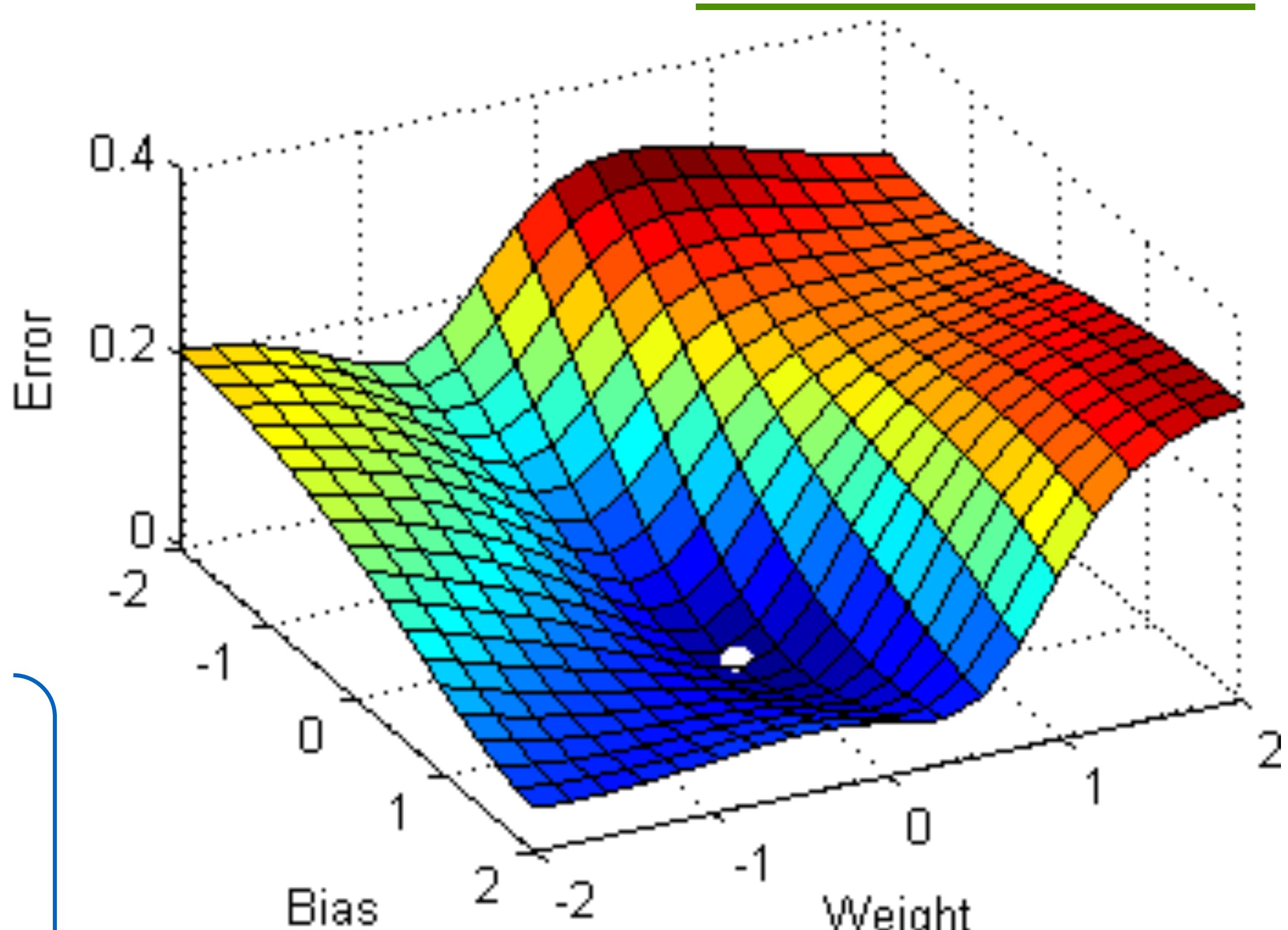
We want to minimize $C(W, B, S^r, E)$

Stochastic Gradient Descent

- 1. Compute ∇C (gradient direction of cost function)
- 2. Small step towards the $-\nabla C$ direction
- 3. Repeat

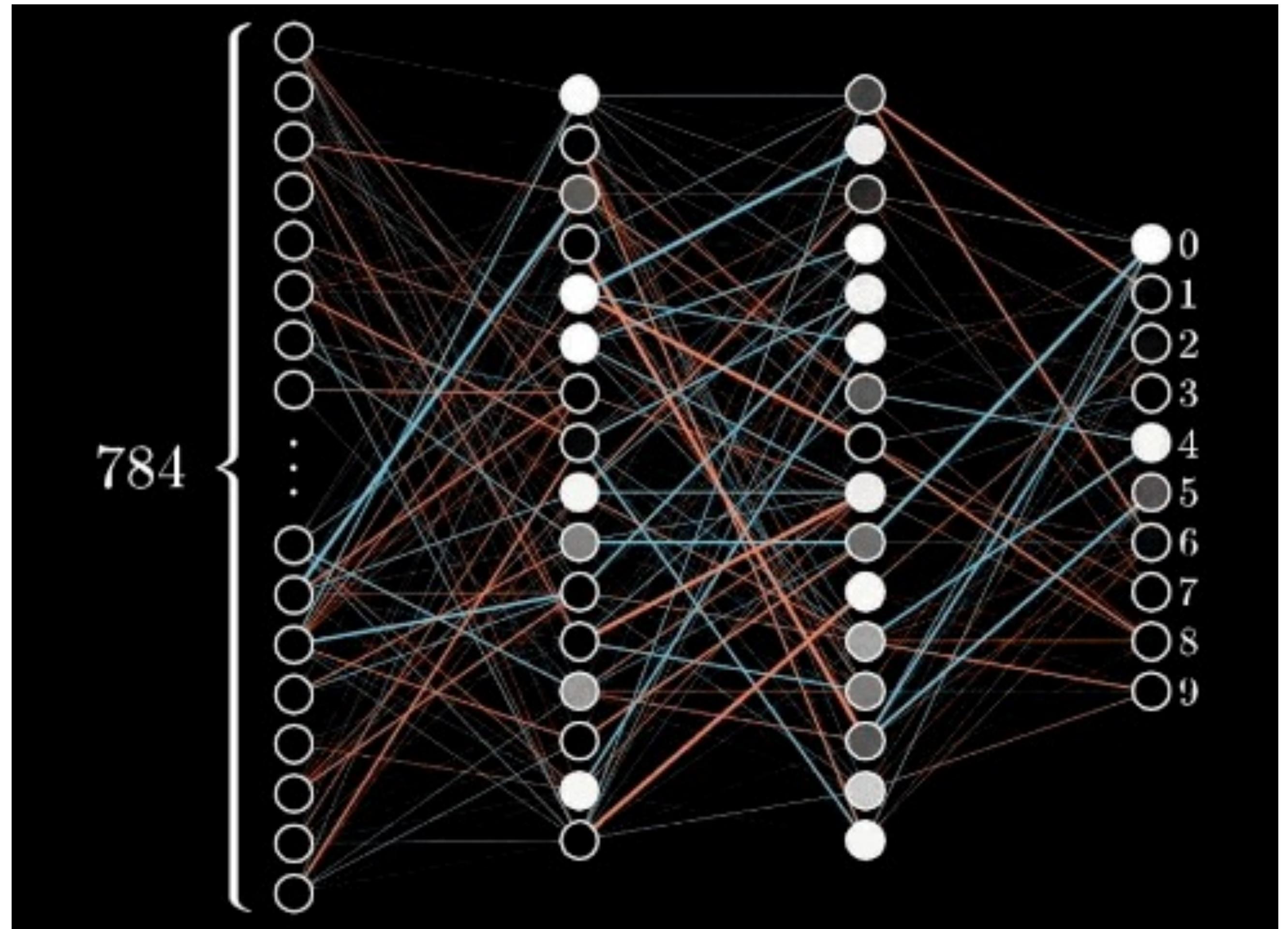
$$\vec{W} = \begin{bmatrix} 1.1 \\ 3.8 \\ -2.2 \\ \vdots \\ 1.4 \\ 2.12 \\ -1.4 \end{bmatrix}$$

$$-\nabla C(\vec{W}) = \begin{bmatrix} .1 \\ .2 \\ -.3 \\ \vdots \\ 1.4 \\ -.05 \\ .2 \end{bmatrix}$$

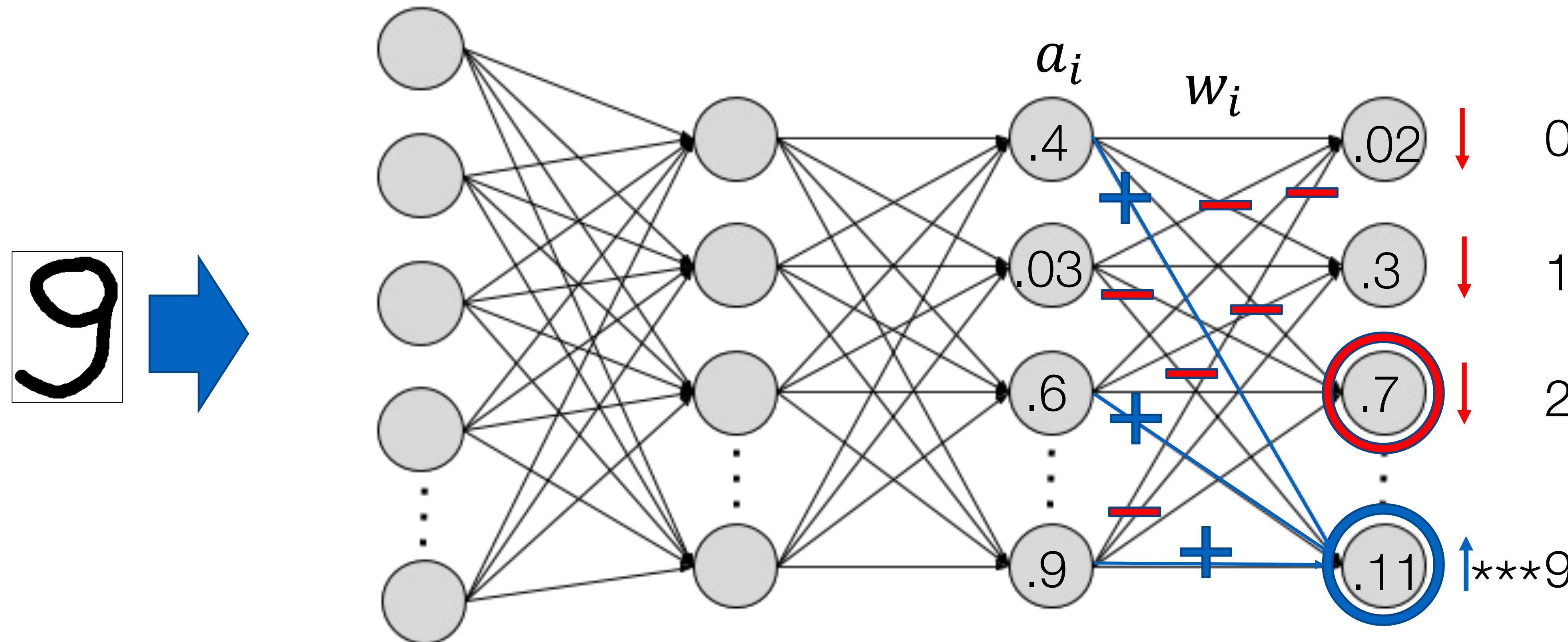


Backpropagation for computing Stochastic Gradient Descent

- Increase bias (b) and weights (w_i) in proportion to previous layer activations (a_i)
- Change activations from the previous layer (a_i) in proportion to weights (w_i)
- Repeat over the entire training dataset



1. Increase bias (b) and weights (w_i) in proportion to previous layer activations (a_i)



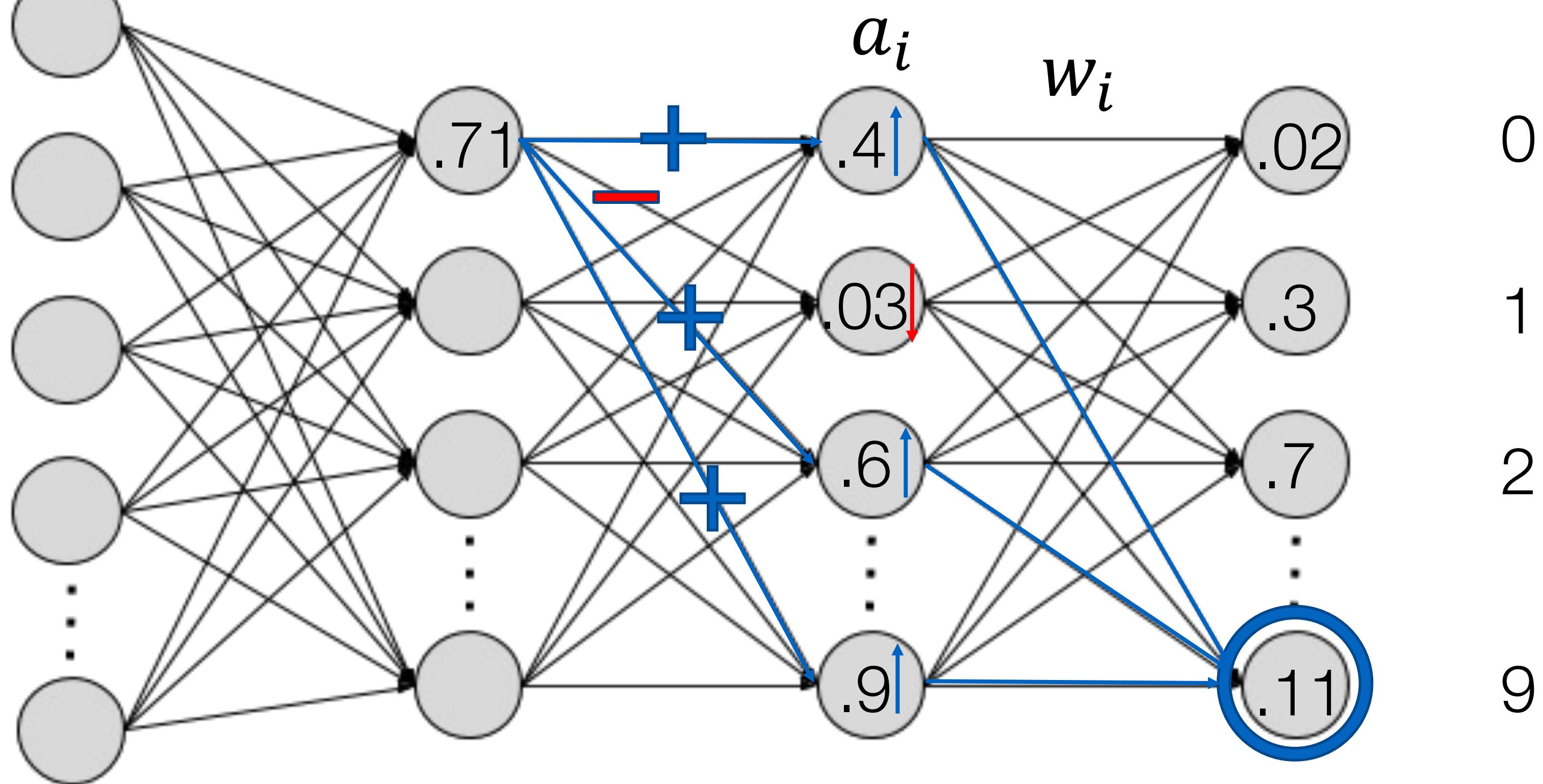
Focus on the activation of the neuron we want to increase:

$$.11 = \sigma(w_0 a_0 + w_1 a_1 + \dots + w_{n-1} a_{n-1} + b)$$

2. Change the activations from the previous layer (a_i) in proportion to weights (w_i)

9

- We can't directly alter the activations of neurons
- We seek to maximize magnitude
- We gradually produce a list of changes to apply per layer
- We recursively apply the same process backwards



“When 2 connected cell neurons fire simultaneously, the connection between them strengthens” – Donald Hebb (1949)

3. Repeat for the entire Training Dataset

	2	1	6	2	7	...	Average over all training dataset
w_0	-.02	+.5	-.21	-.1	+.4	...	\rightarrow +.01
w_1	+.3	-.31	-.07	+.2	-.12	...	\rightarrow +.09
w_2	+.1	+.03	-.09	+.1	+.02	...	\rightarrow -.03
:	:	:	:	:	:	...	:
w_{n-1}	+.05	-.01	+.04	-.03	-.17	...	\rightarrow -.19

$$= -\nabla C(w_1, w_2, \dots, w_{n-1})$$

Backpropagation with Batching

- Computationally expensive process if done case by case, instead:
 - Shuffle dataset
 - Divide in batches
 - Compute a gradient descent “step” per batch

$\{ 9 \ 3 \ 7 \ 1 \}$

Compute step with backprop for batch #1

$\{ 5 \ 4 \ 2 \ 9 \}$

Compute step with backprop for batch #2

$\{ 8 \ 2 \ 6 \ 0 \}$

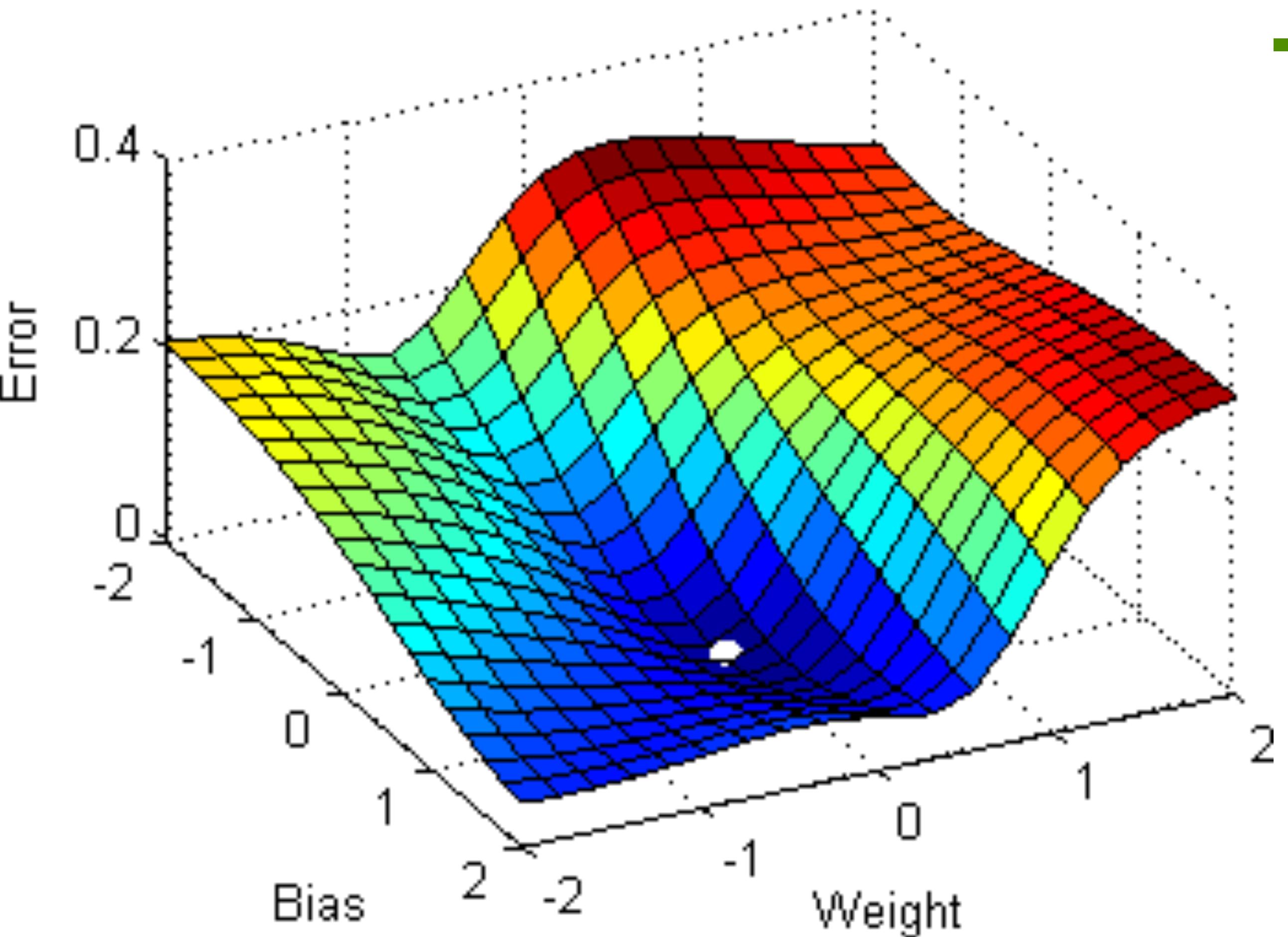
Compute step with backprop for batch #3

$\{ 6 \ 4 \ 7 \ 5 \}$

Compute step with backprop for batch #4

Recap

- Back propagation is the algorithm that determines how a single training example (case) would alter neurons' weights and biases for achieving the most rapid decrease to the cost.
- By dividing our training dataset in batches, the network will converge to a local minimum of the cost function (C) by making gradual adjustments.
- NOTE #1: The training dataset should be sufficiently large for backpropagation to work.
- NOTE #2: Ineffective with "noisy" training data



Admin

Credits

- CMU Computer Vision course - Matthew O'Toole. <http://16385.courses.cs.cmu.edu/spring2022/>
- CIS 419/519 Applied Machine Learning. Eric Eaton, Dinesh Jayaraman.
<https://www.seas.upenn.edu/~cis519/spring2020/>
- Deep Learning Patterns and Practices - Andrew Ferlitsch, Manning, 2021
- Machine Learning Design Patterns - Lakshmanan, Robinson, Munn, 2020
- Grokking Machine Learning. Luis G. Serrano. Manning, 2021
- Deep Learning for Vision Systems. Mohamed Elgendi. Manning, 2020

Advanced Machine Learning For Design

Lecture 5: Machine Learning for Image
Processing (part 2)

Evangelos Niforatos
19/10/2022

aml4d-ide@tudelft.nl
<https://aml4design.github.io/>