Computer Visions
Generative Machine Learning Tools

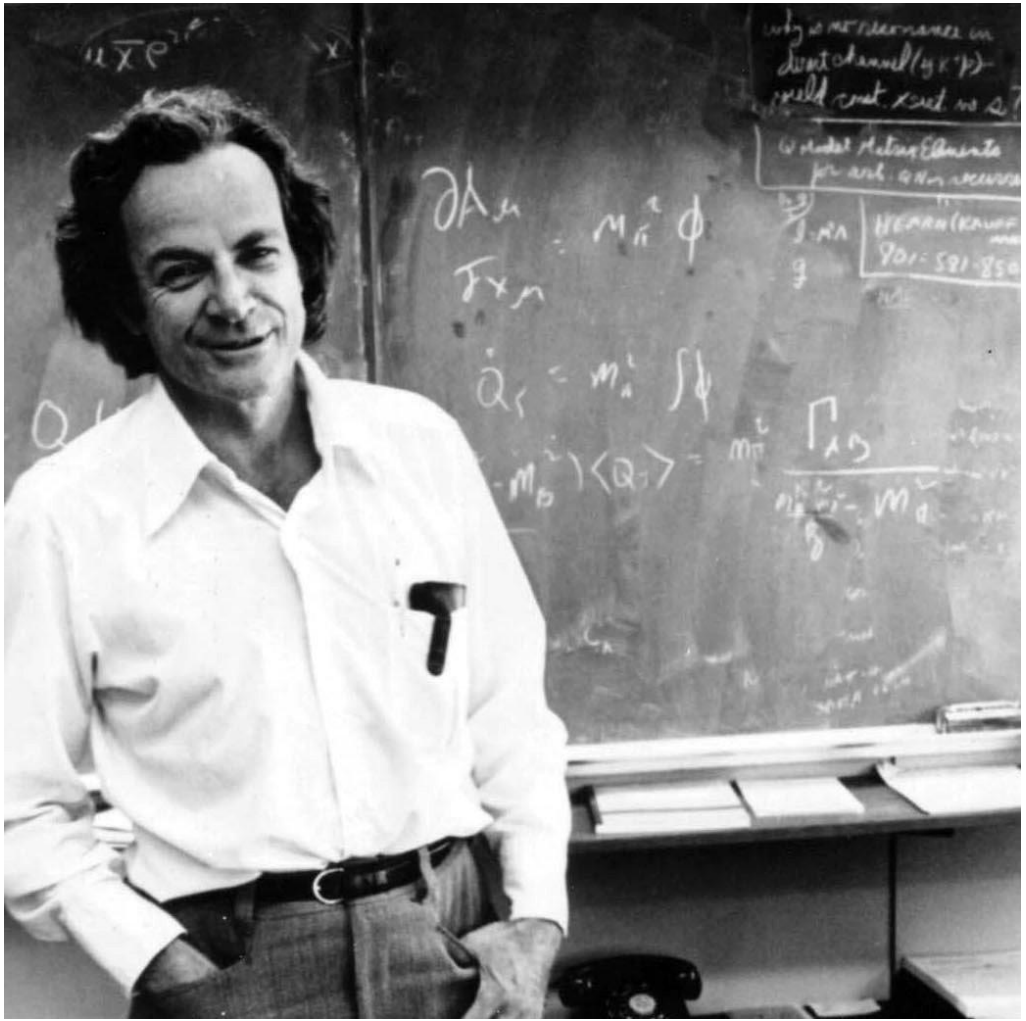
MAS.S68 F’19

Roy Shilkrot, Fluid Interfaces

Class 1: Introduction

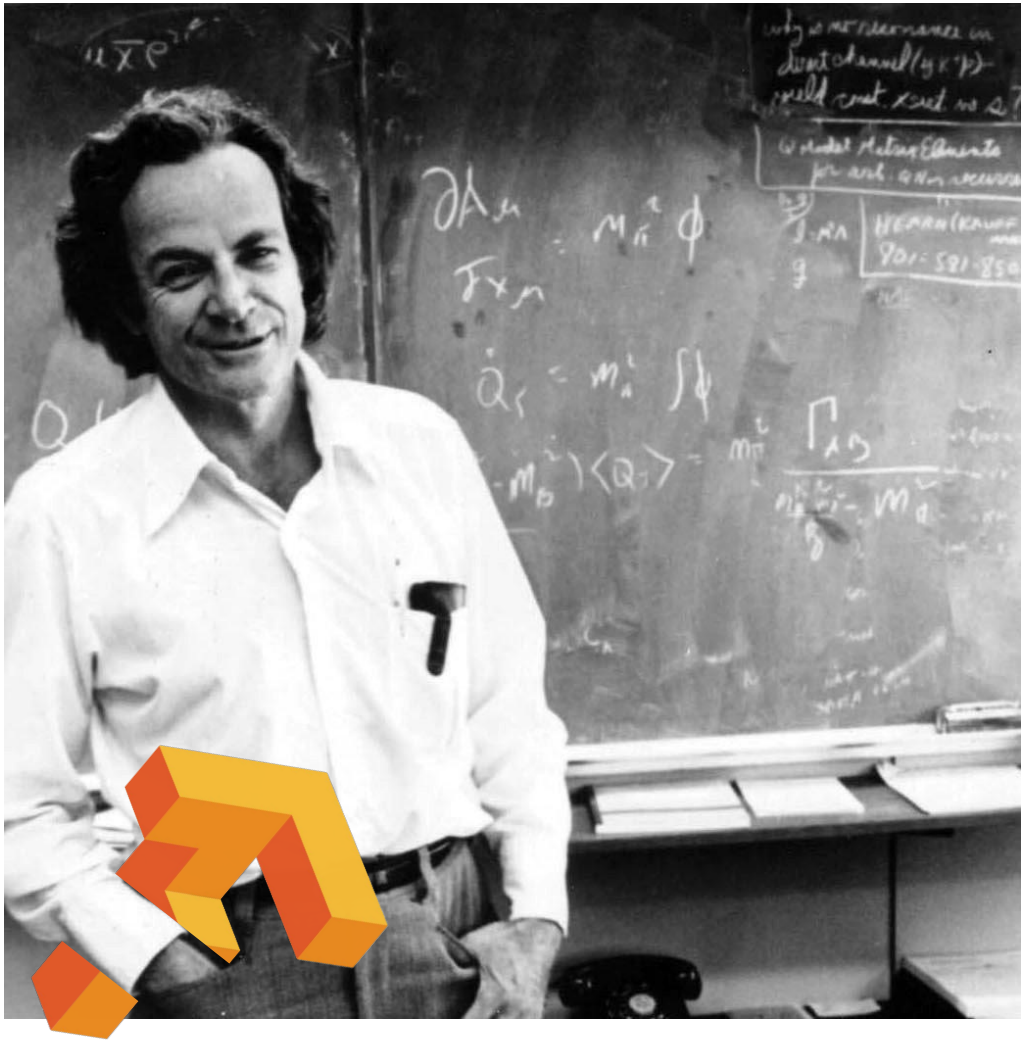
“What I cannot create, I do not understand.”

Richard Feynman



“What I cannot ~~create~~ generate, I do not understand.”

Richard fAInman



Today

Intro

- Logistics
- What will happen, What will not.
- Context: Art + [Tech, Sci, Math]

Content

- Machine Learning
- Linear Models and Artificial Neural Networks
- Deep Learning
- Generative Models and Unsupervised Learning

Application

- ~~Tooling: Python, Jupyter, Docker, TF/Ker./PT~~
- Style Transfer
- Deep Dream (aka Puppy Slugs)
- Neural Doodle

Course Logistics

Location: E15-466

Time: Thursdays from 1-3pm, on a once-every-two-weeks schedule.

Instructors: Prof. Pattie Maes, Dr. Roy Shilkrot

Units: 0-6-0

Assignments: 1 class 1 assignment

Final projects: Groups of 1-3

Grading scheme:

- Assignments: 50%
- Final project: 50%
- Extra credit: 5%

Course materials will be up online.

Timeline

	Date	Topic	Assignment / Model
1	Sep. 5th	Intro, Context, ML, NN, DL, Tools	ImageNet: Style, Deep Dream, Neural Doodle
2	Sep. 19th	Image: Autoencoders, GANs	VAE, DCGAN, C/Info GAN, BigGAN, Cycle
3	Oct. 3rd	Text: RNNs, Atten., Transformers	Word2Vec, BERT, GPT, ELMo, Txt2img
4	Oct. 17th	Sketch, Draw, Paint: more RNNs	Sketch RNN,
5	Oct. 31st	Music, Audio, Composition	WaveNet, Melody RNN
6	Nov. 14th	3D generative models, GraphNN	Image to 3D, 3D GAN, 3D
7	TBD	Project presentations	

What is this class about?

We Will Discuss	**We Will Not Discuss**
Deep Learning Models	Other statistical modeling
Generative Application	Classification, Regression, Detection, ...
Machine Learning Mechanics	Model Training strategies
Pre-Trained Models, Retraining	Training from scratch
Data ingest, ETL	Data pipelines engineering, Big Data
Gradient Descent, Batch GD	Broyden–Fletcher–Goldfarb–Shanno
Creativity Assistance	Art Philosophy

Generative Art

Nature: Chaos vs. Order

Math / Physics: capture form chaos a formula = explicit order

Algorithm, Simulation: Inverse Math (rules, formula) back to Nature

Art: Simplification, Amplification, Emotion/Expression, of Nature/Natural

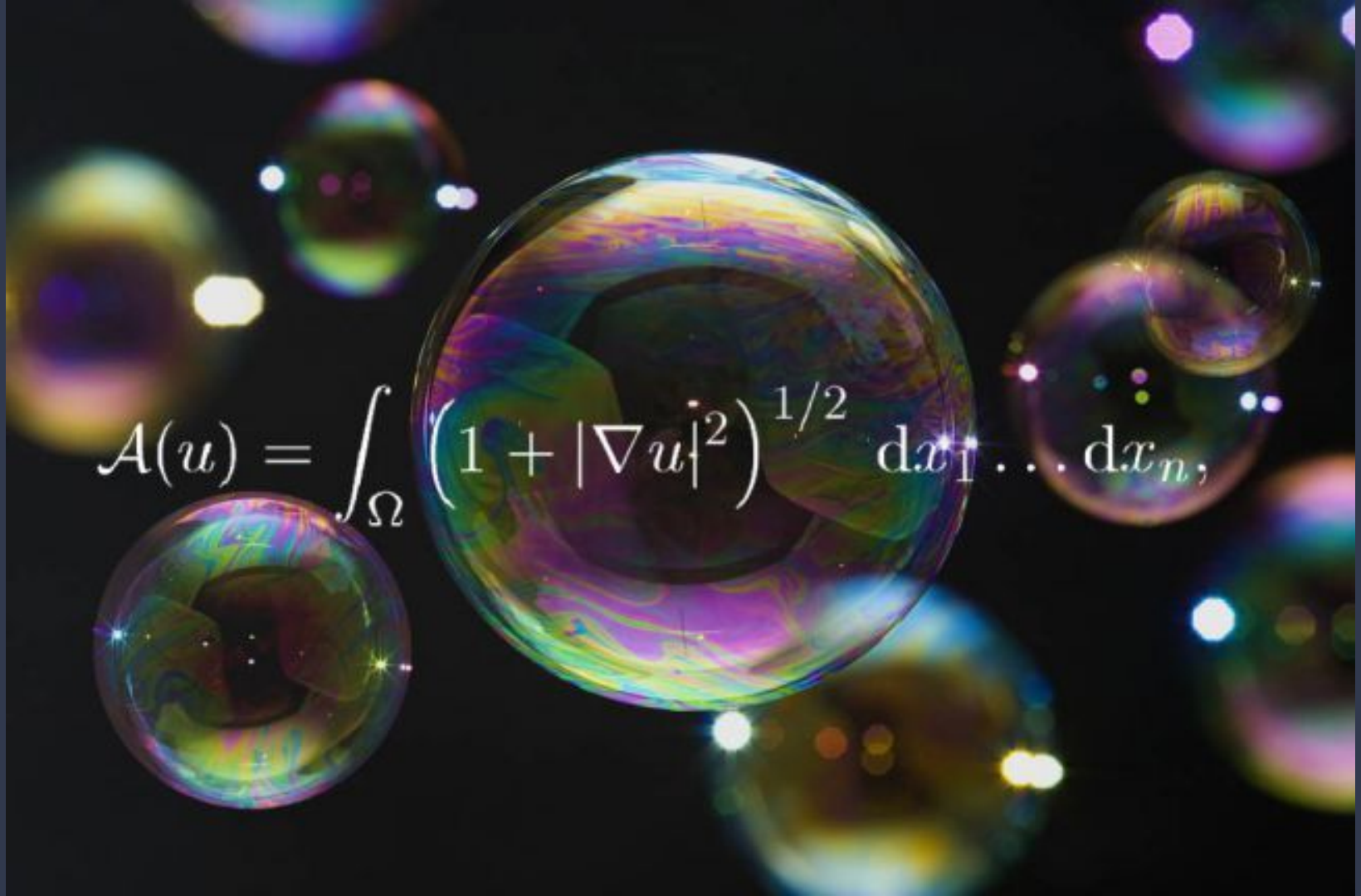
Generative Art: Simplistic Rules, Random input

Data: Opposite of Math, capture chaos with statistics/implicit order

Generative from: 1) Math, 2) Statistics

Generative **AI** Art: Real world data Input, generative statistical modeling, sampling

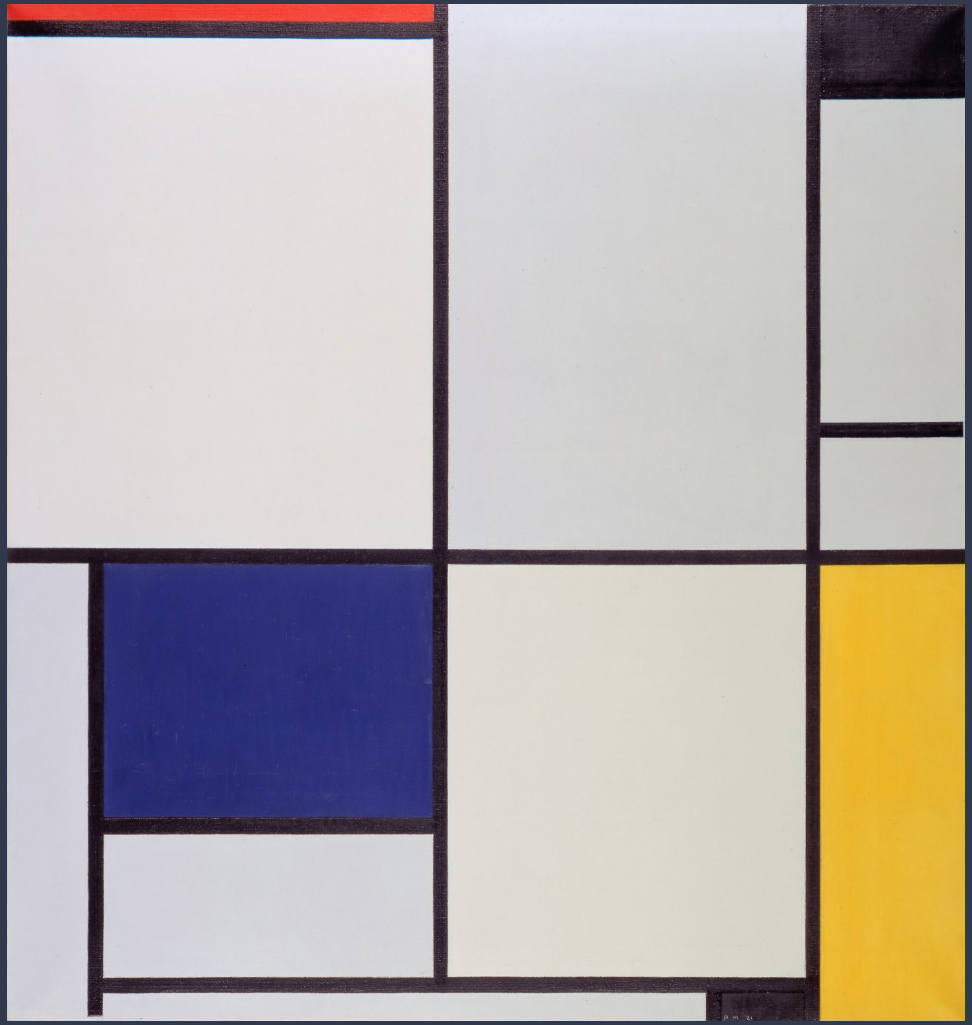

CHAOS
ORDER


$$\mathcal{A}(u) = \int_\Omega \left(1 + |\nabla u|^2\right)^{1/2} \mathrm{d}x_1 \ldots \mathrm{d}x_n,$$

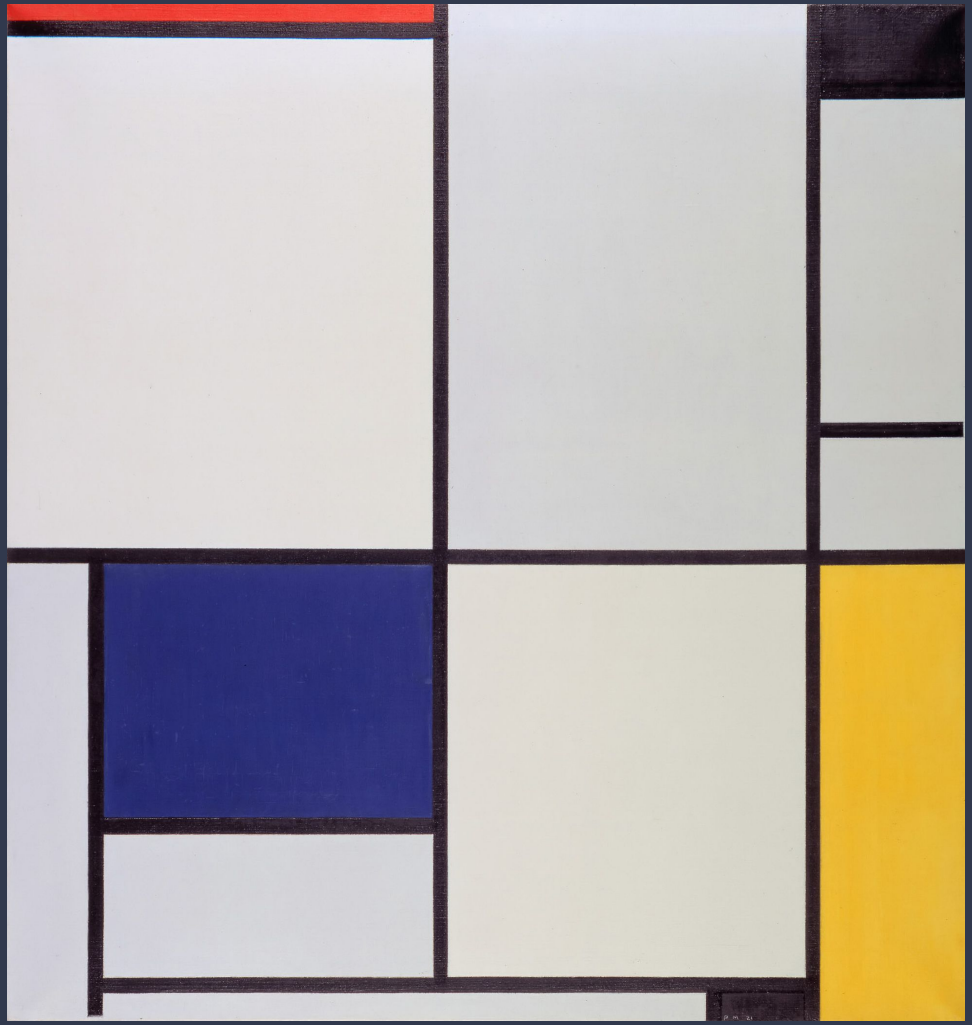
Lindenmayer System

(L-System)

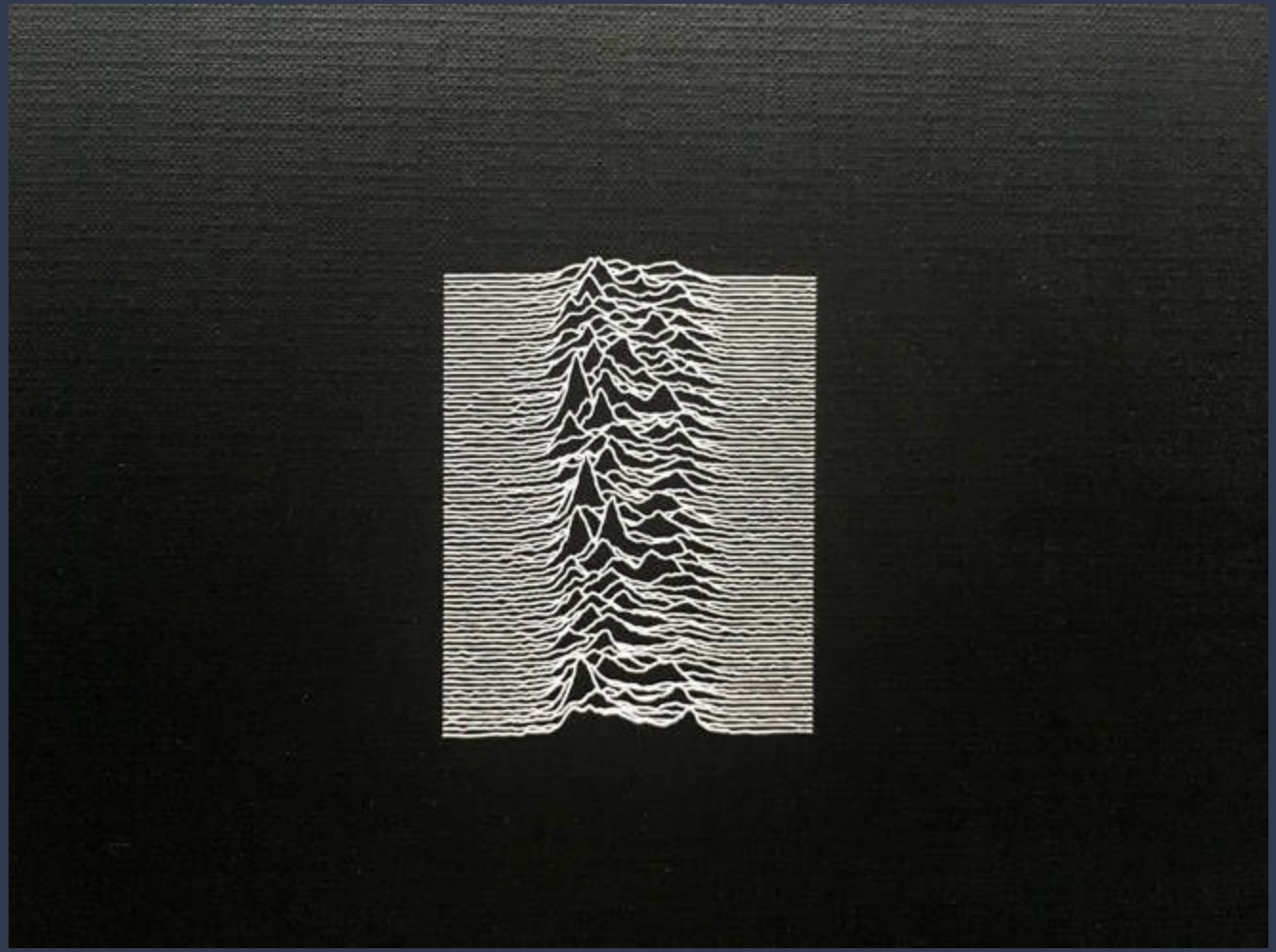
MUSIC AND MATH: THE GENIUS OF BEETHOVEN



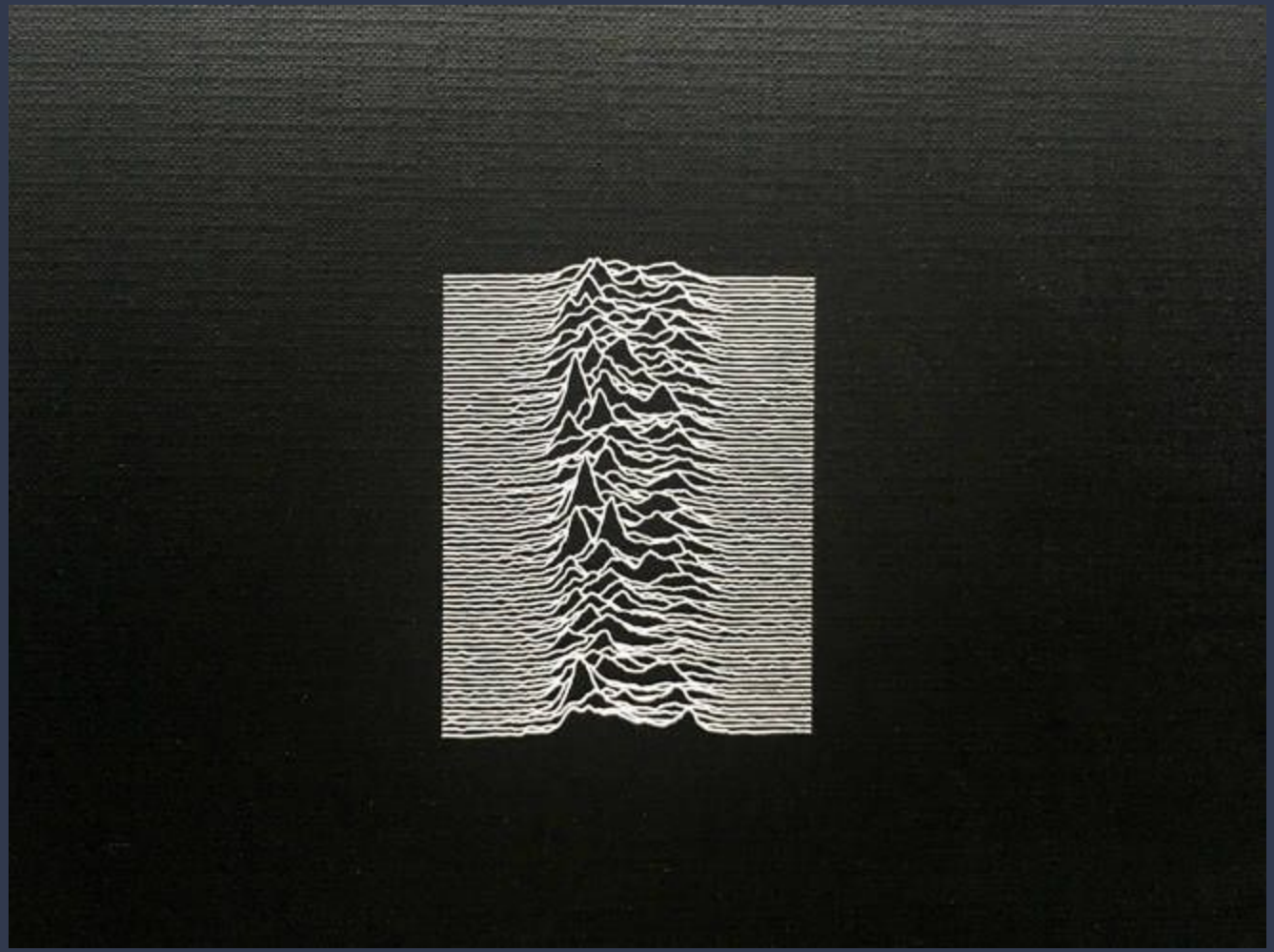
Piet Mondrian
1921



Feel?

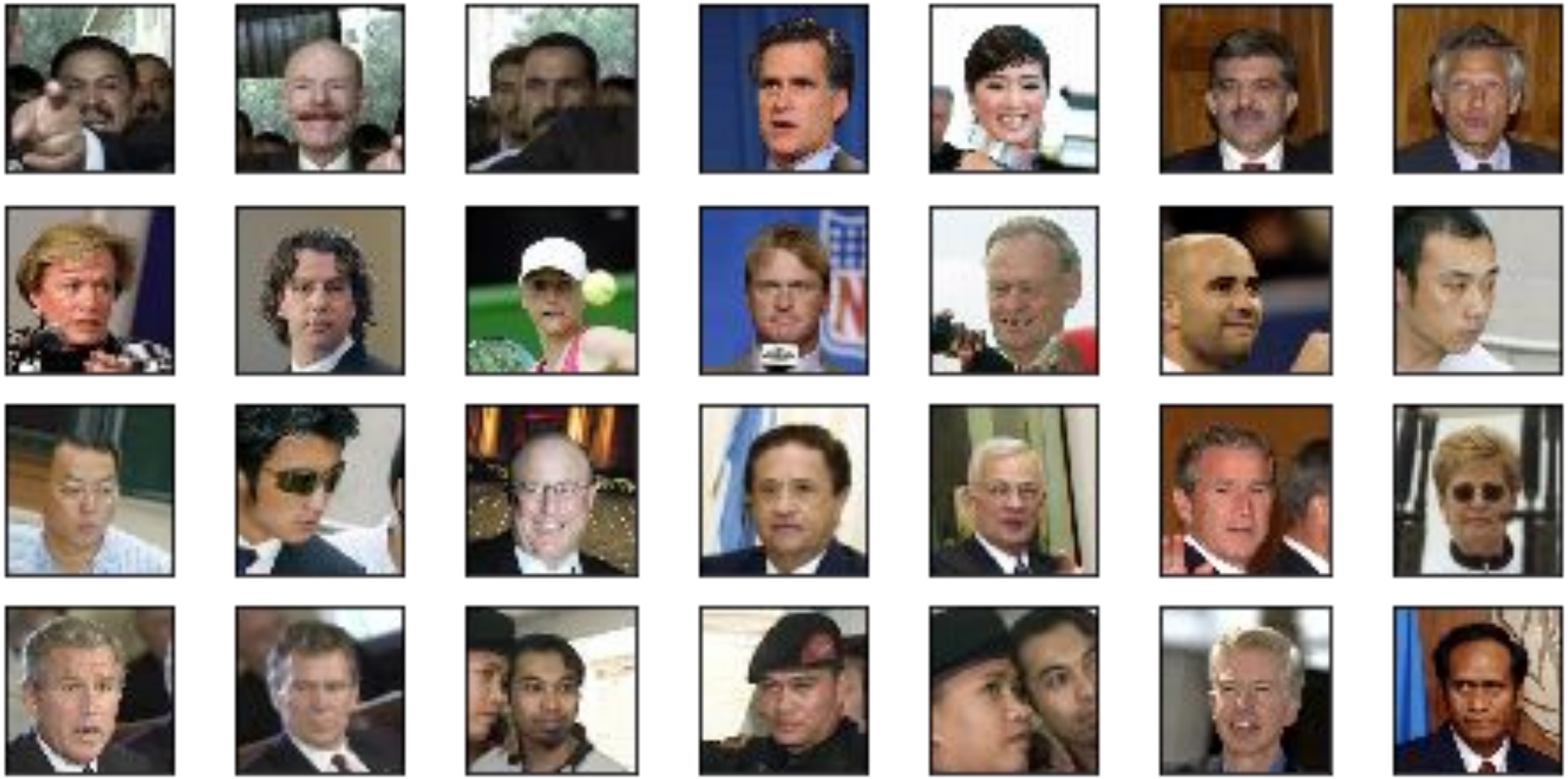


Joy Division
1971, 1979



Feel?

Art from Rules Data Random



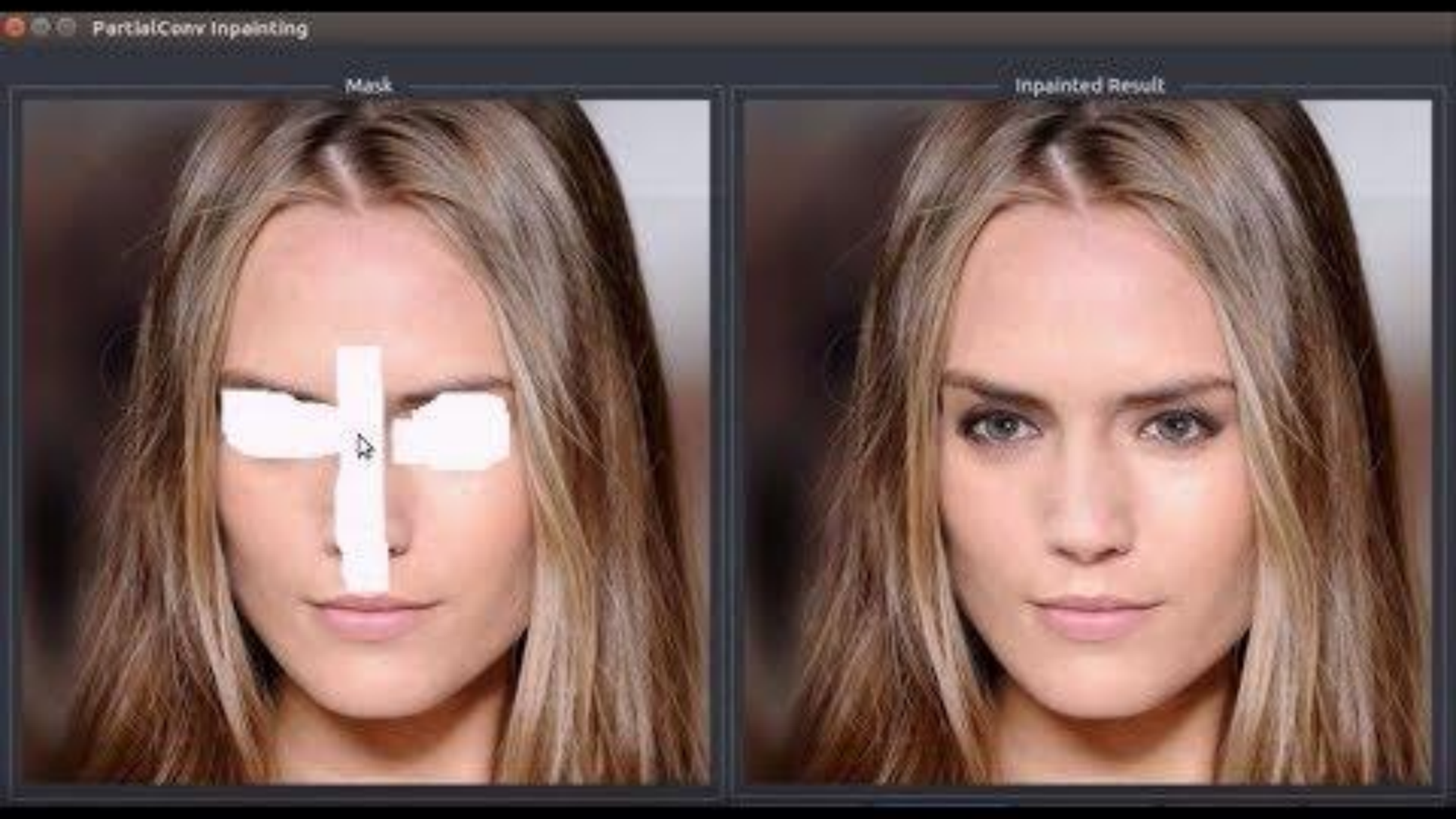
Implicit Order

Implicit Order

Coarse styles
($4^2 - 8^2$)

Middle styles
($16^2 - 32^2$)

Fine styles
($64^2 - 1024^2$)


PartialConv Inpainting
Mask
Inpainted Result






GANVAS STUDIO

COLLABORATIVE NEURAL NETWORK ART PRINTS

Make your Ganbreeder creations physical! Our posters are printed with UV-resistant ink on satin-finish paper. Our paintings are printed with latex ink on canvas on wood. We hand-paint details, edges, and textures onto the canvas to accent more detail and remove pixelation. Email us if you have any questions. Artwork takes 3-4 weeks to ship.

Image URL :

https://ganbreeder.app/i?k=762423a!

Enter a URL from ganbreeder.app or choose an example below.






18" Satin-Finish Poster ($30)

12" Canvas on Wood ($90)

24" Canvas on Wood ($200)

Order Now

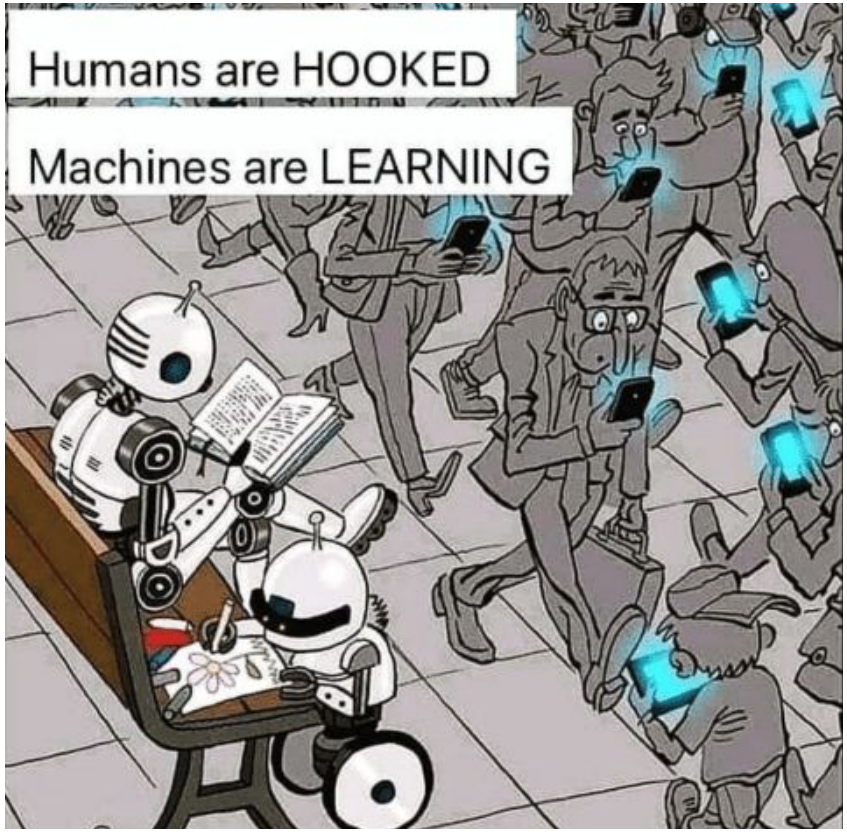
Let AI work for **you**!

😮 💸

https://magenta.tensorflow.org/music-transformer

Machine Learning

Intuition (visual problems)
Basic concepts
Linear models
Neural Networks
Backpropagation
Convolutional NNs
ILSVRC
Generative Models



Machine Learning for Visual Tasks

Algorithms that improve their prediction performance at some task with experience (or data).

Data

(experience)

Learning Algorithm

(task)

- Outside
- Street
- Day
- Crosswalk
- Junction
- ...

Understanding

(performance)

[Orabona]

Object Detection

Recognition | Tasks

Recognition | Tasks

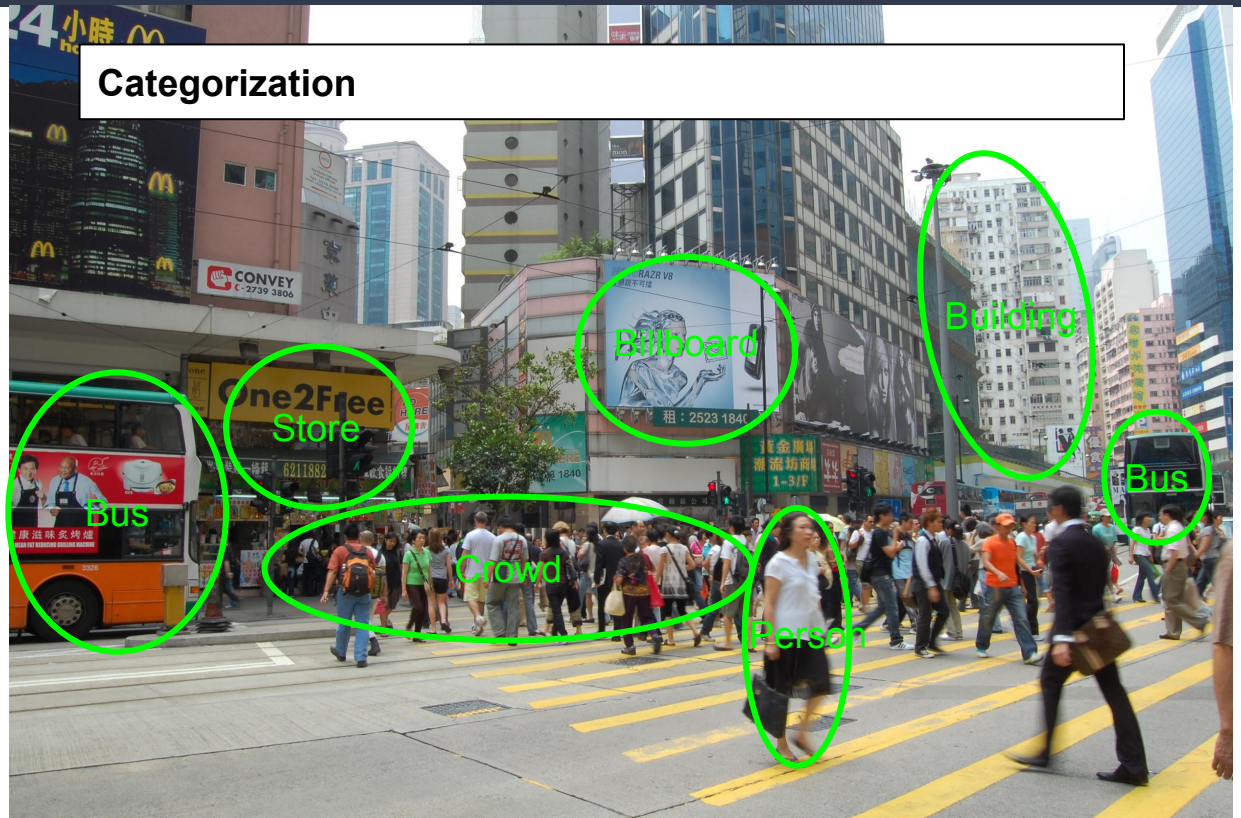
Recognition | Tasks

Recognition | Tasks

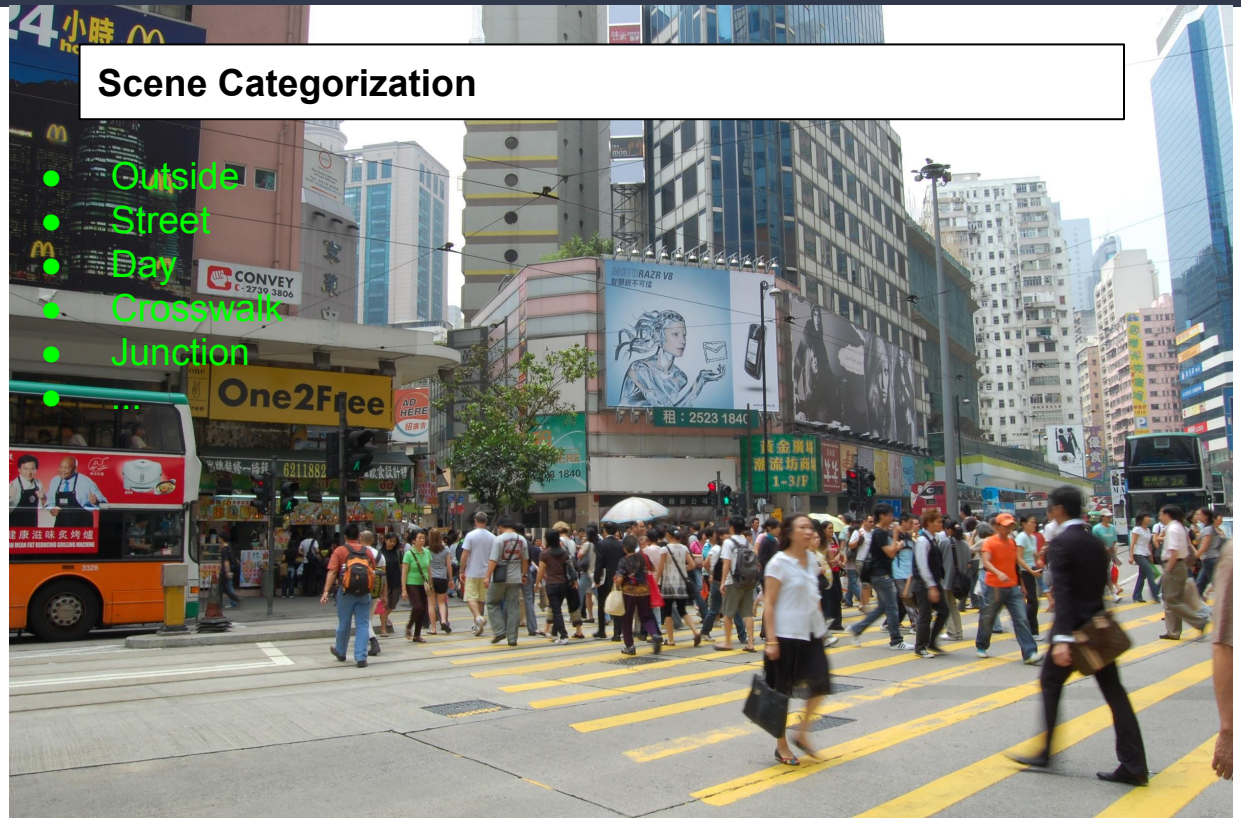
Recognition | Tasks

Recognition | Tasks

Recognition | Tasks



Recognition | Tasks

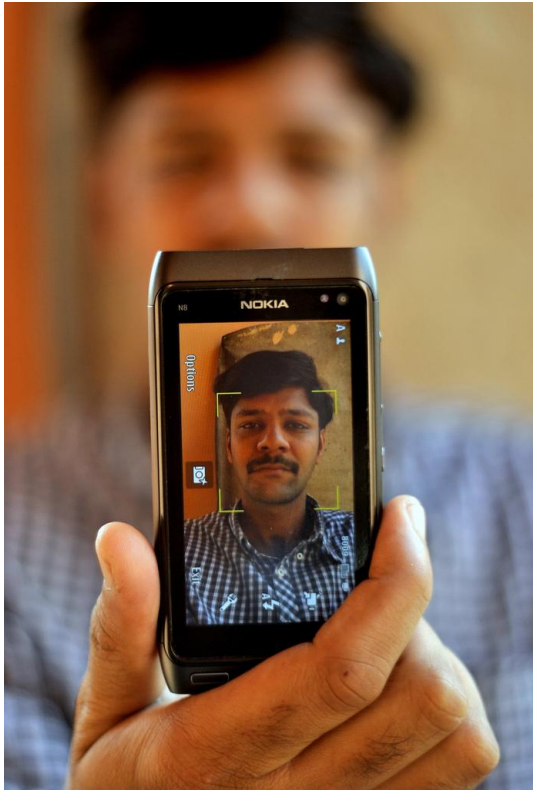


Applications

Autonomous Driving

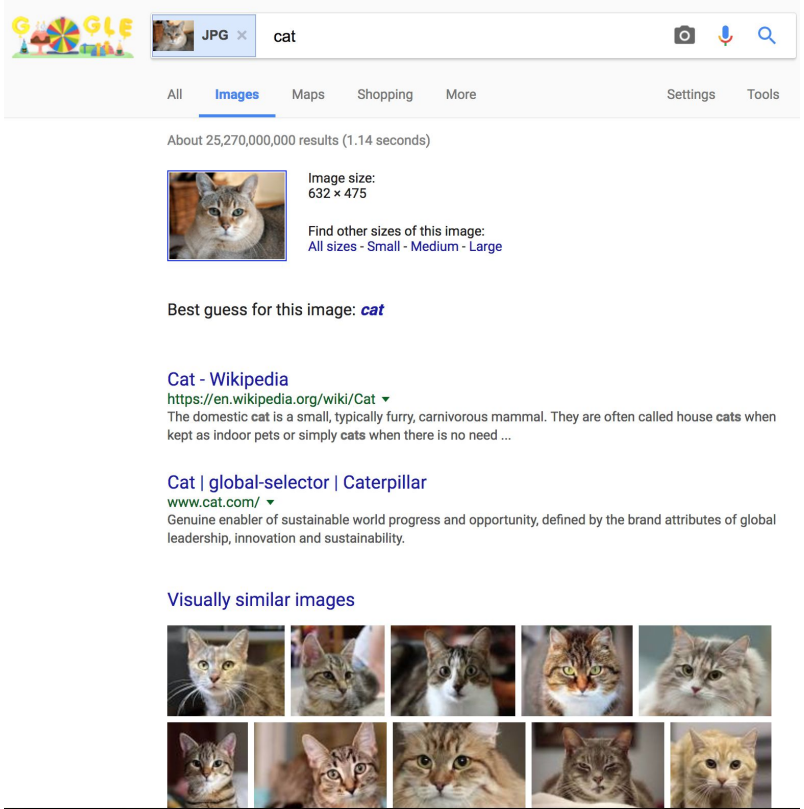
Applications

Photography



Applications

Image Search



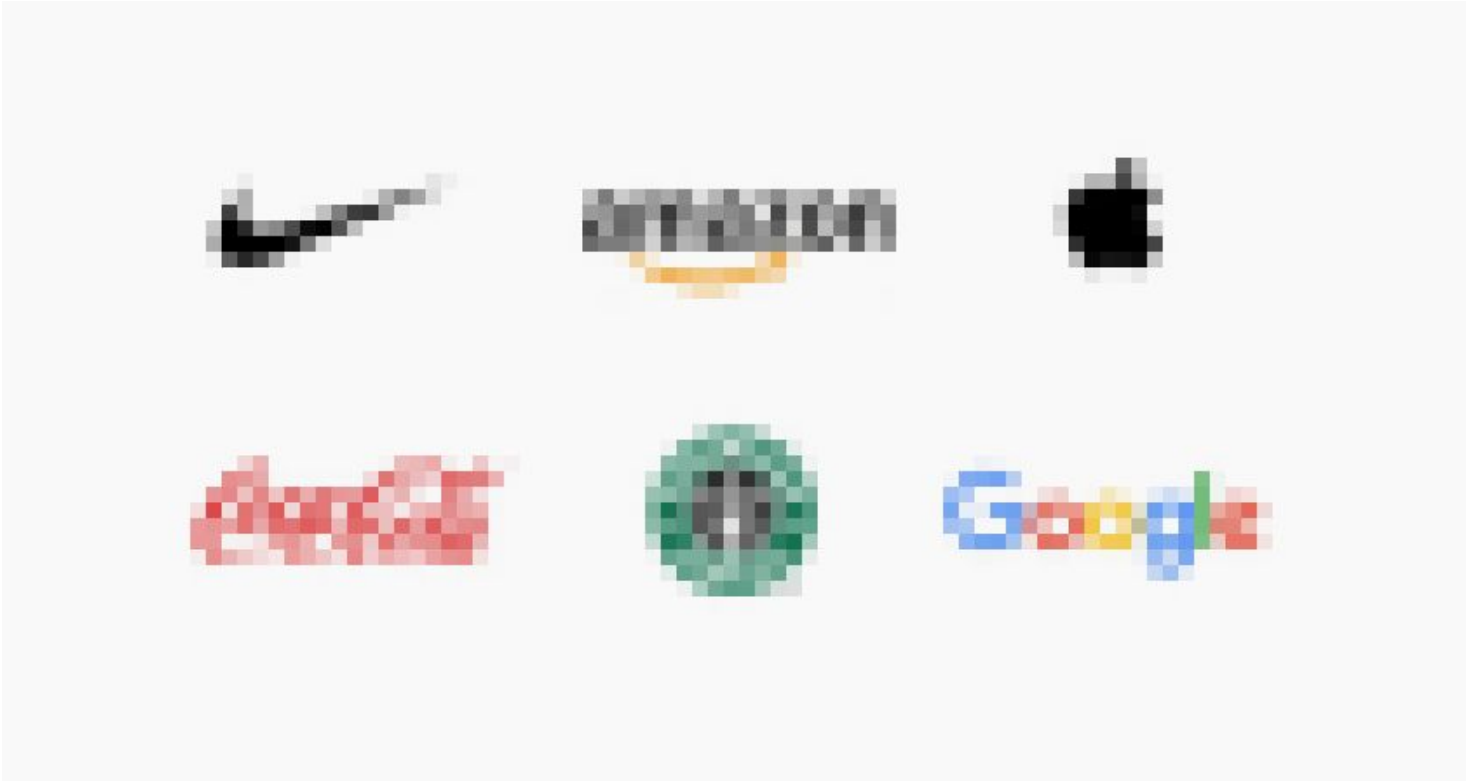
Face Recognition by Humans: Nineteen Results All Computer Vision Researchers Should Know About

Increased knowledge about the ways people recognize each other may help to guide efforts to develop practical automatic face-recognition systems.

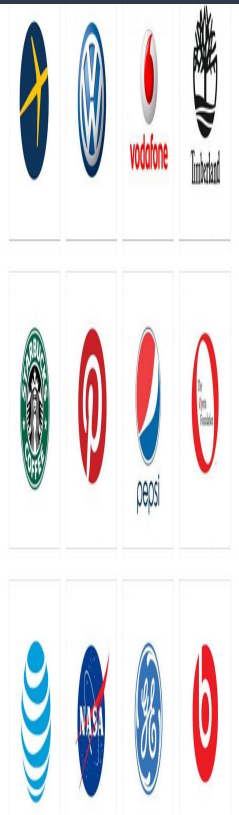
By Pawan Sinha, Benjamin Balas, Yuri Ostrovsky, and Richard Russell

[2006]

Human Vision | Brand Detection



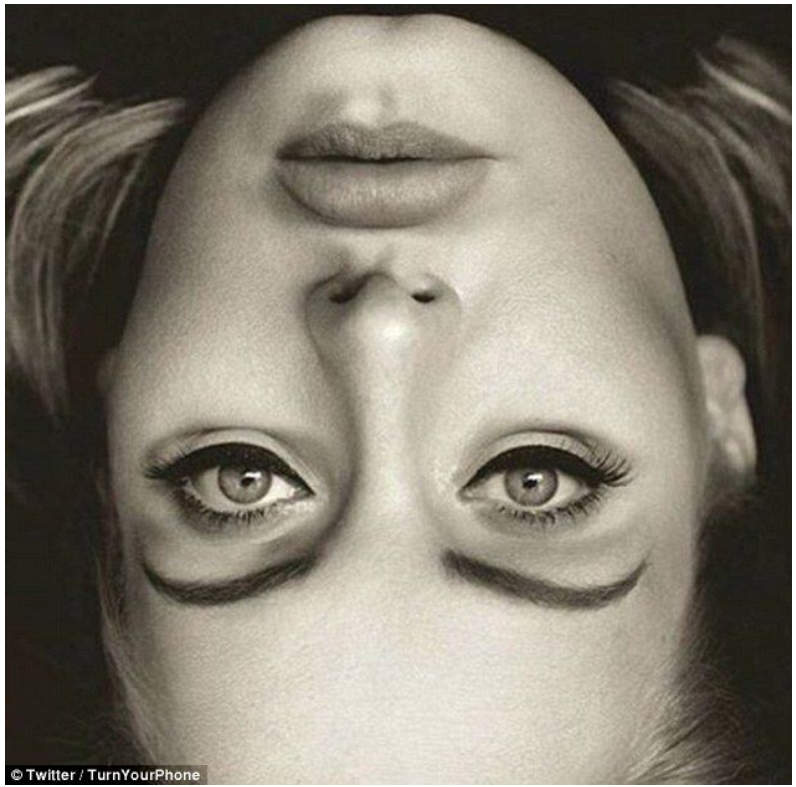
Human Vision | Brand Detection



Human Vision | Face Detection

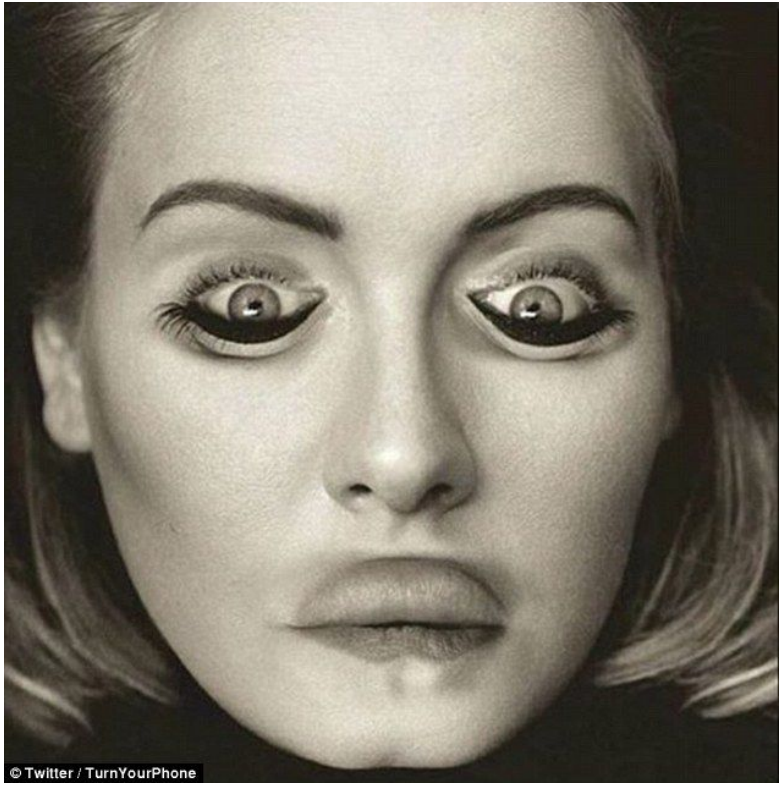
Human Vision | Face Detection

Human Vision | Face Detection



[Link]

Human Vision | Face Detection

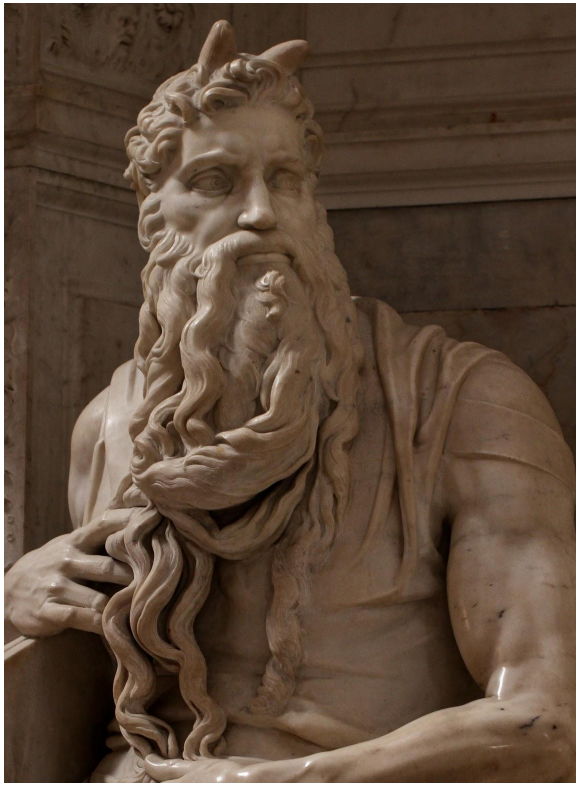


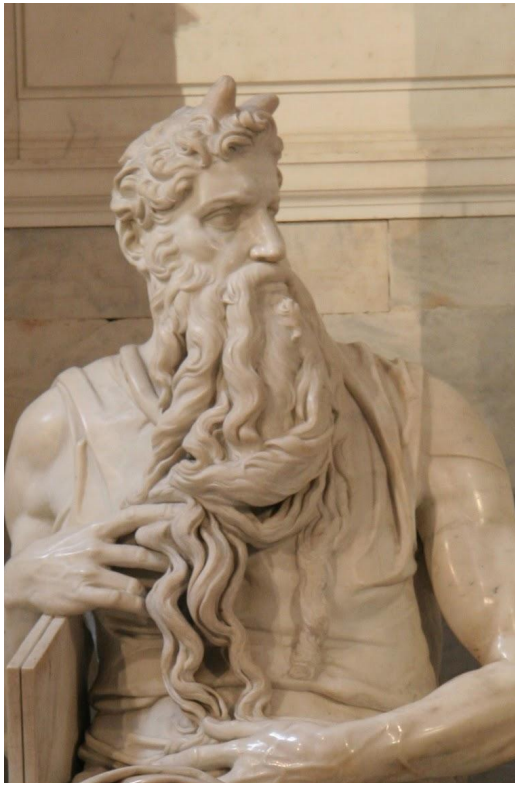
[Link]

Challenges | Scale of Domain

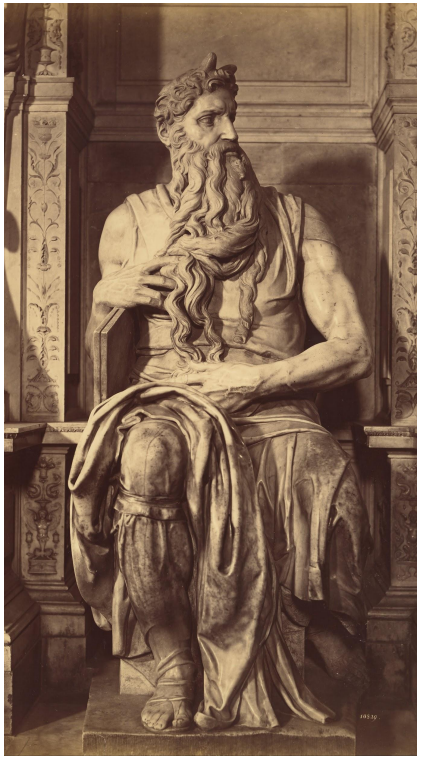
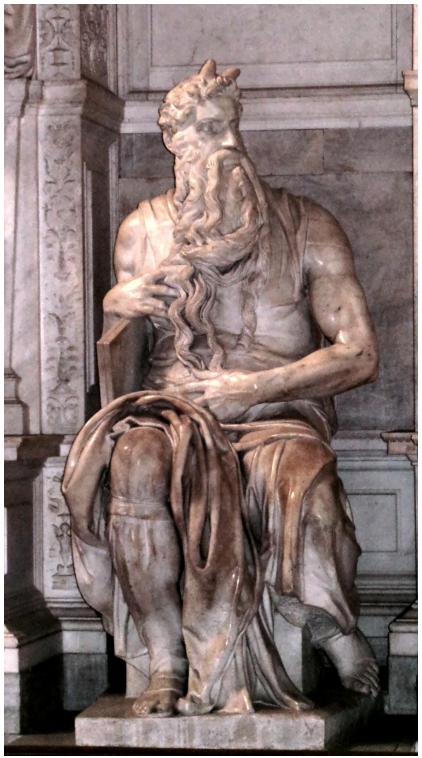
~100,000 nouns in the (spoken) english language
~21,000 object categories in ImageNet
Estimated 30,000-40,000 object types

Challenges | Viewpoint





Challenges | Lighting


Challenges | Scale

Challenges | Occlusion

Magritte, 1964

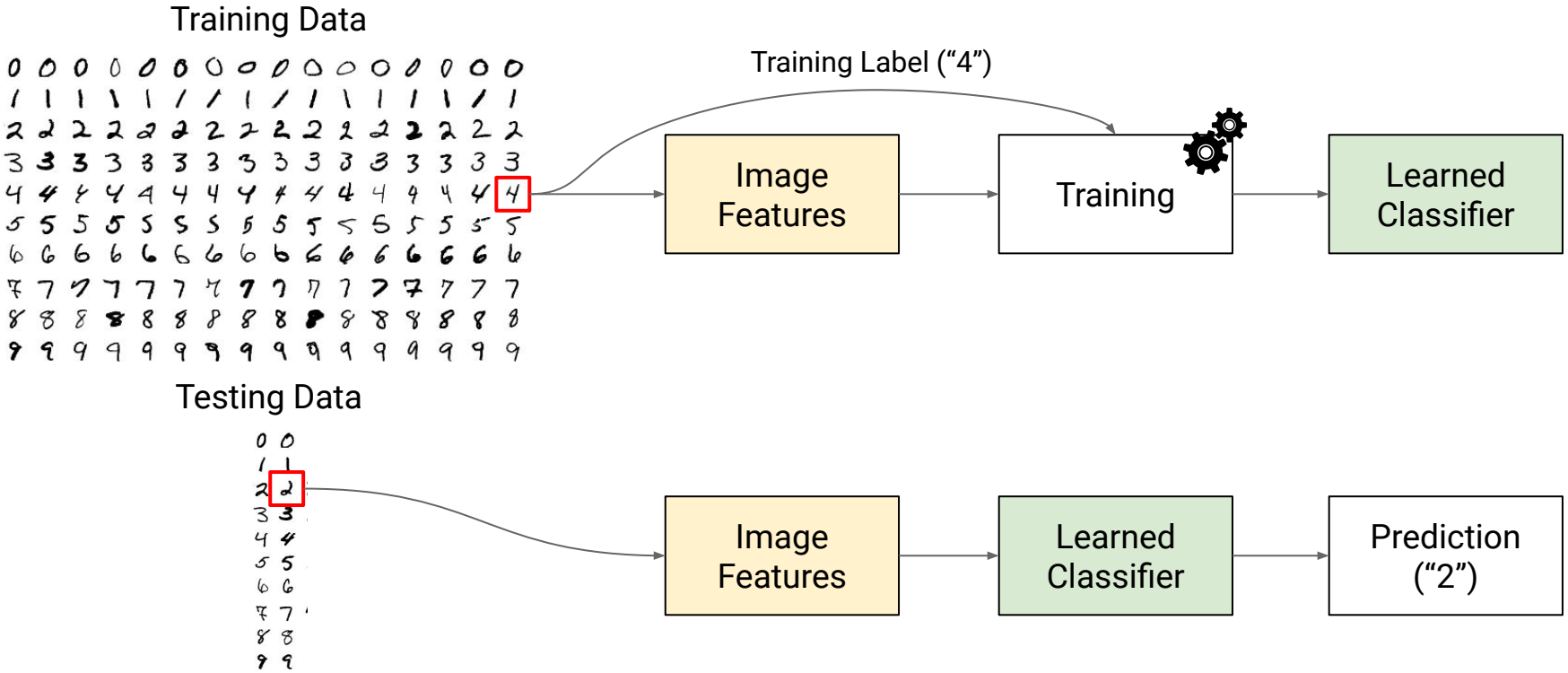
Challenges | Intra-Class Variation

Car

Car

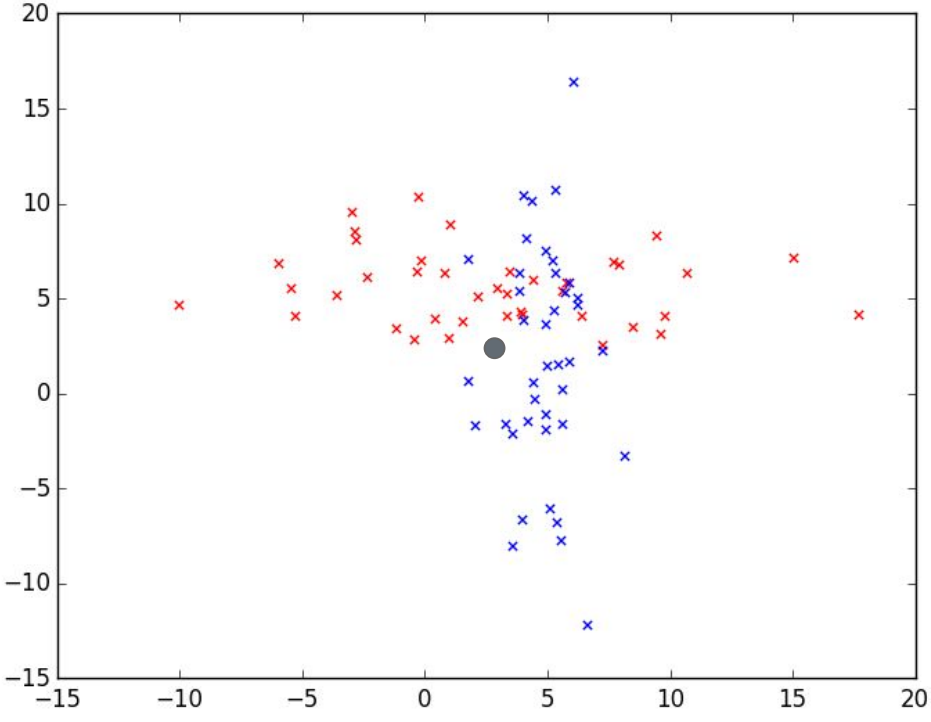
Car

Simple Machine Learning Pipeline



Simple Classifier

What's the simplest idea you can think of to classify a new sample to blue or red?

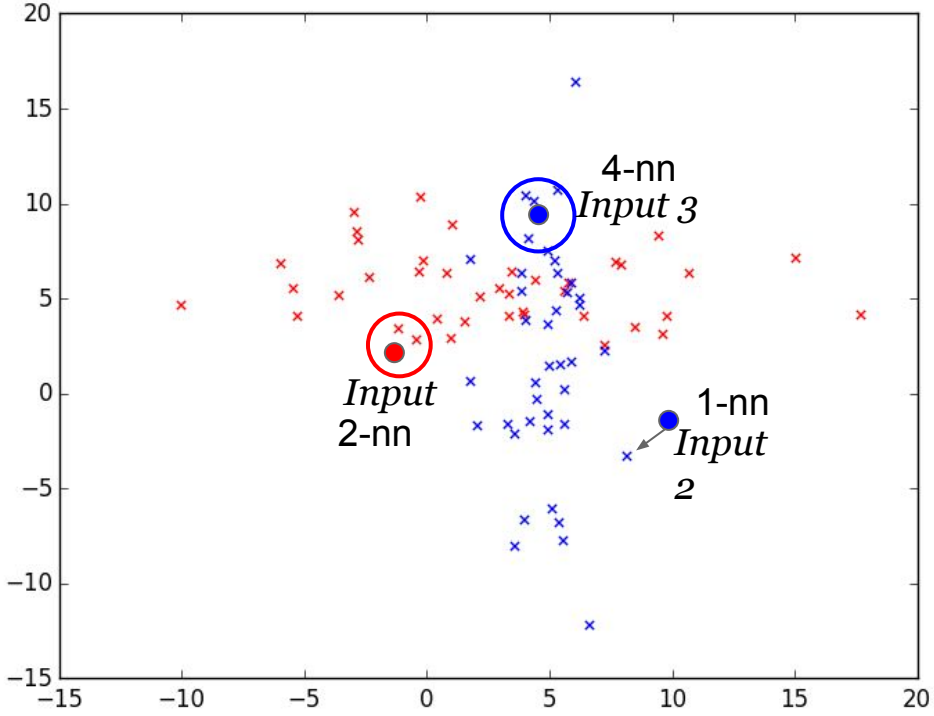


Nearest Neighbor Classifier

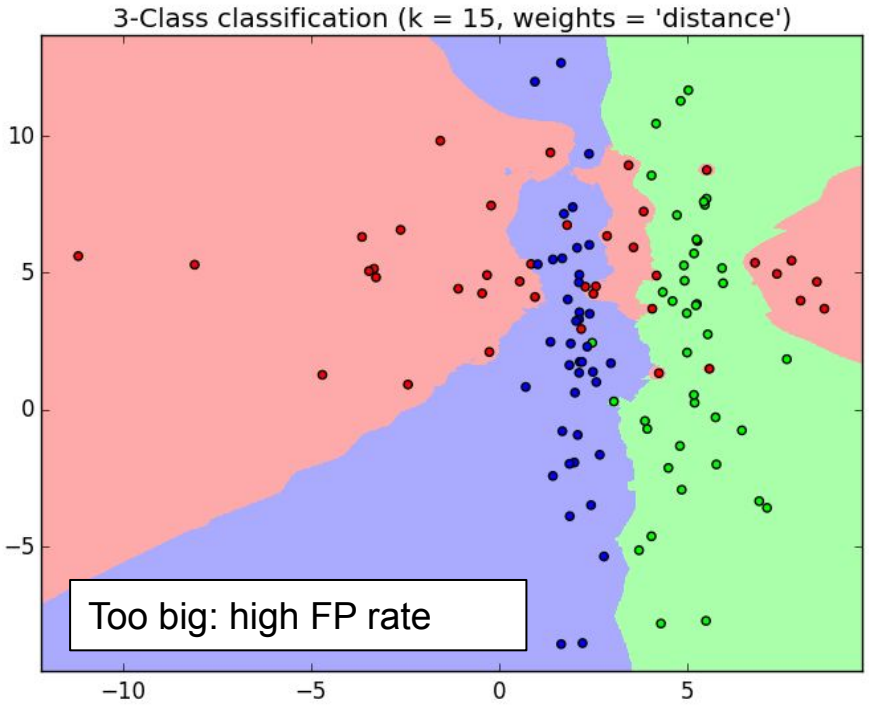
The simplest idea: Given an input image, find the closest prior (training) sample and return its label.

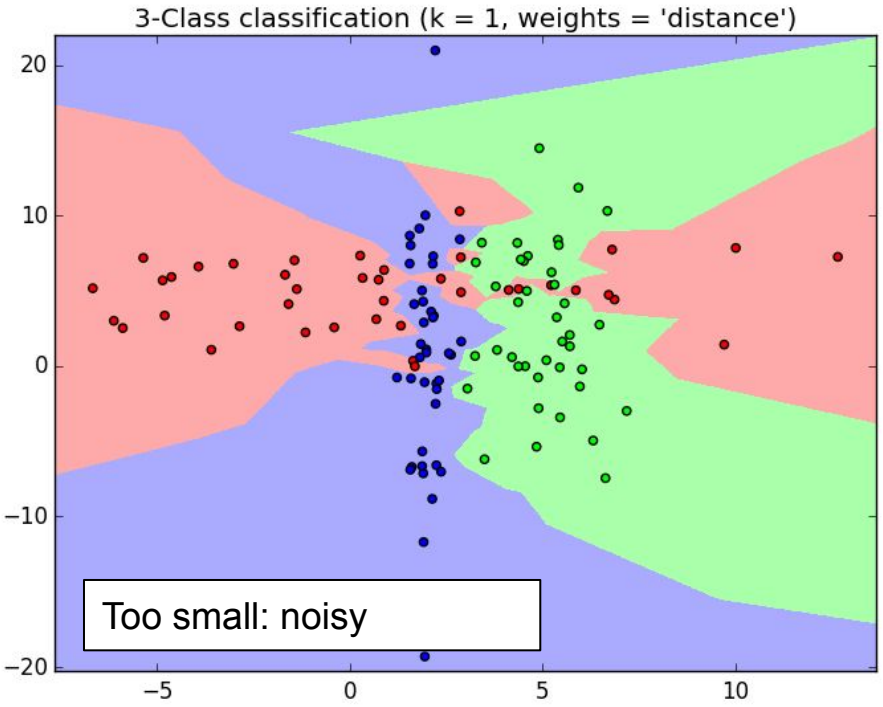
Distance metric can be chosen: L_2, L_1, K-L, cross-correlation, etc.

k-Nearest Neighbor: Find the closest k neighbors and make a decision (the mean, median, etc.)



Nearest Neighbor Classifier | Choose k





Naive Bayes Classifier

class Image features

$$p(C_k \mid x_1, \ldots, x_n)$$

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Bayes rule

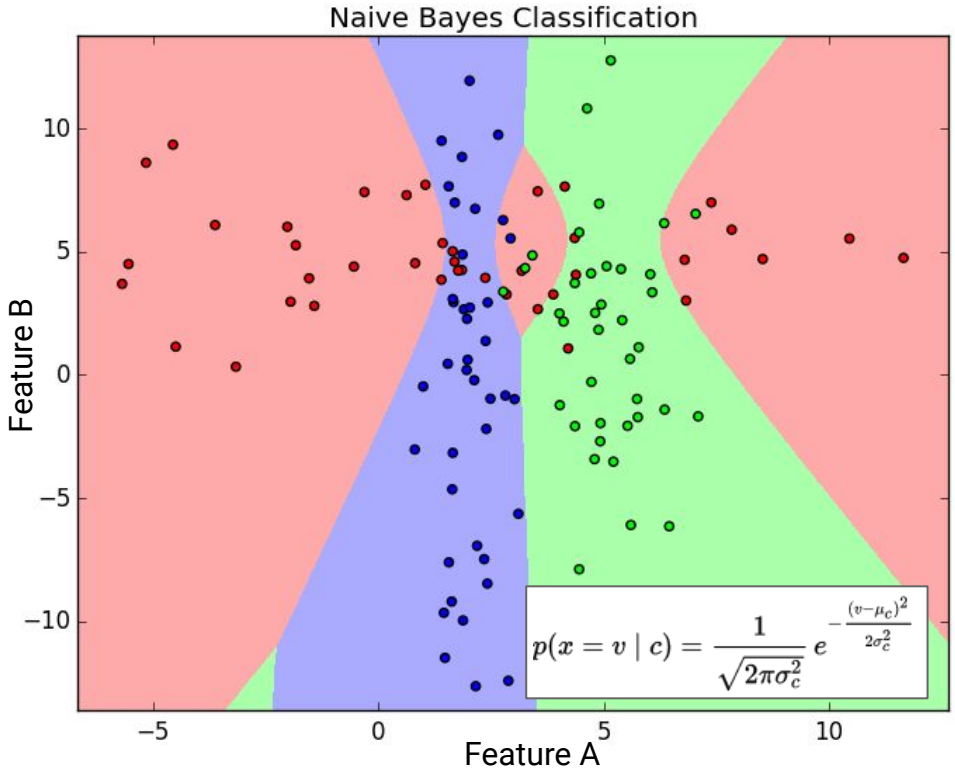
Assume feature independence.

$$p(x_i \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k)$$

Model the posterior using a multiplication of the features conditional probability:

$$p(C_k \mid x_1, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Calculate $p(x_i|C_k)$ from dataset, e.g. fit a gaussian.



Naive Bayes | In the Learning Pipeline

Training Data

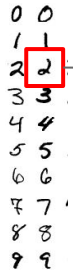
Training Label (“4”)

Image Features → Training → Learned Classifier

Training objective

$$\hat{\theta} = \arg\min_{\theta} \sum_{s}^{\#\text{samples}} \left[d\left(C_k^{(s)}, \arg\max_{C_k} p(C_k | \vec{x_s}, \theta) \right) \right]$$

Testing Data



prediction: $\hat{k} = \arg\max_{k} p(C_k|\vec{x},\theta) = \arg\max_{k} p(C_k) \prod_{i} p(x^{(i)}|C_k,\theta)$

Image Features → Learned Classifier → Prediction (“2”)

Regression vs. Classification

Computer vision models

- Observe measured data, **x**
- Draw inferences from it about state of world, **w**

Examples:

- Observe adjacent frames in video sequence
- Infer camera motion

- Observe image of face
- Infer identity

- Observe images from two displaced cameras
- Infer 3d structure of scene

Regression vs. Classification

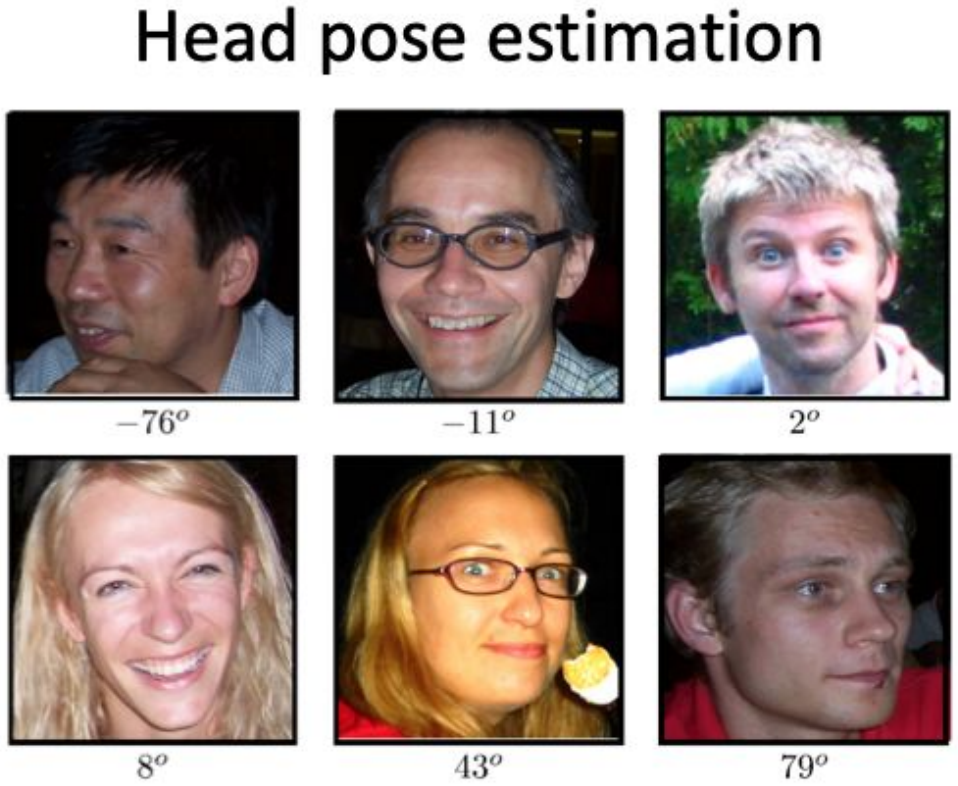
- Observe measured data, **x**
- Draw inferences from it about world, **w**

When the world state **w** is continuous we'll call this regression

When the world state **w** is discrete we call this classification

[Prince]

Regression Application

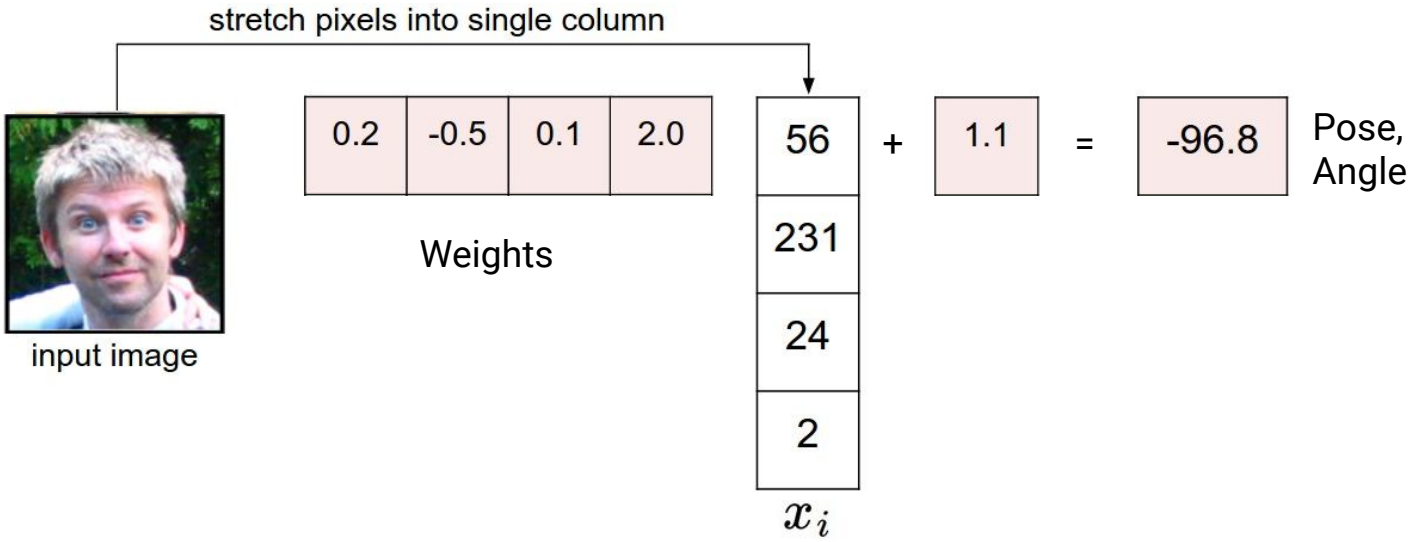


[Prince]

Linear Regression

Find correlation (linear relationship) between image pixels, or other image features, and head pose estimation, e.g. angle.

Goal function: Make Linear combination as close as possible to requested output: $L = \sum_i \|Wx - b\|^2$



[Karpathy]

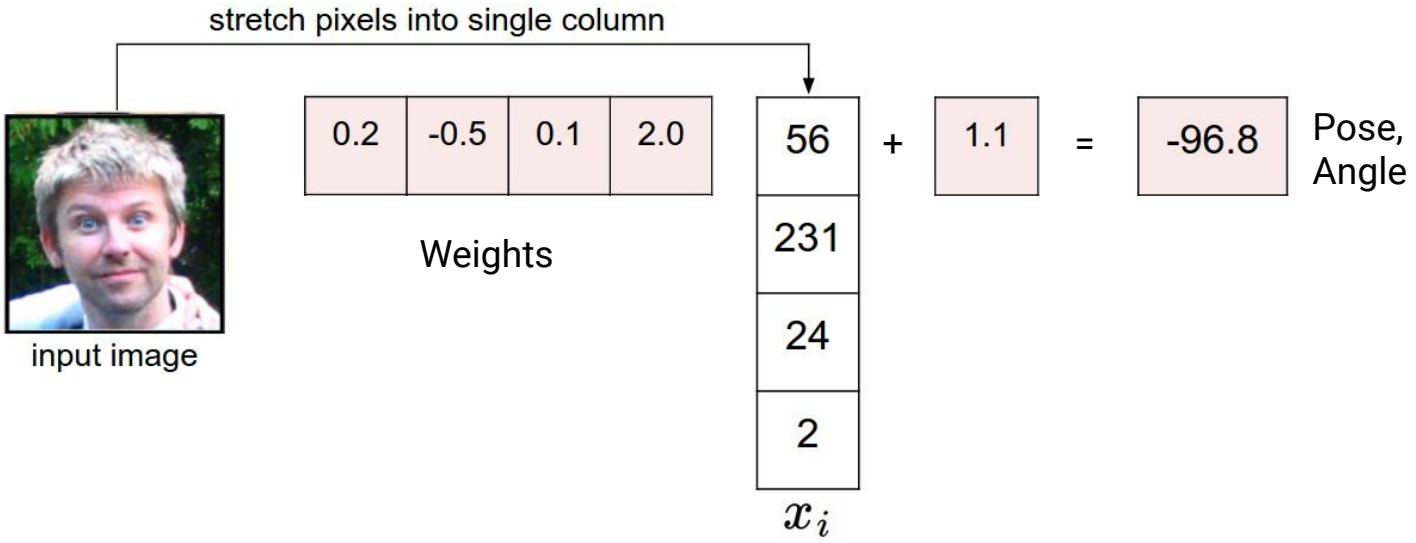
Linear Regression

$$\hat{W} = \arg\min_W \sum_i (Wx - b)^2$$

Derive, set to zero: $\hat{W} = (X^\top X)^{-1} X^\top b$

Major problem: X is very big, $X^\top X$ is enormous,

$(X^\top X)^{-1}$ is intractable



[Karpathy]

Linear Classifier

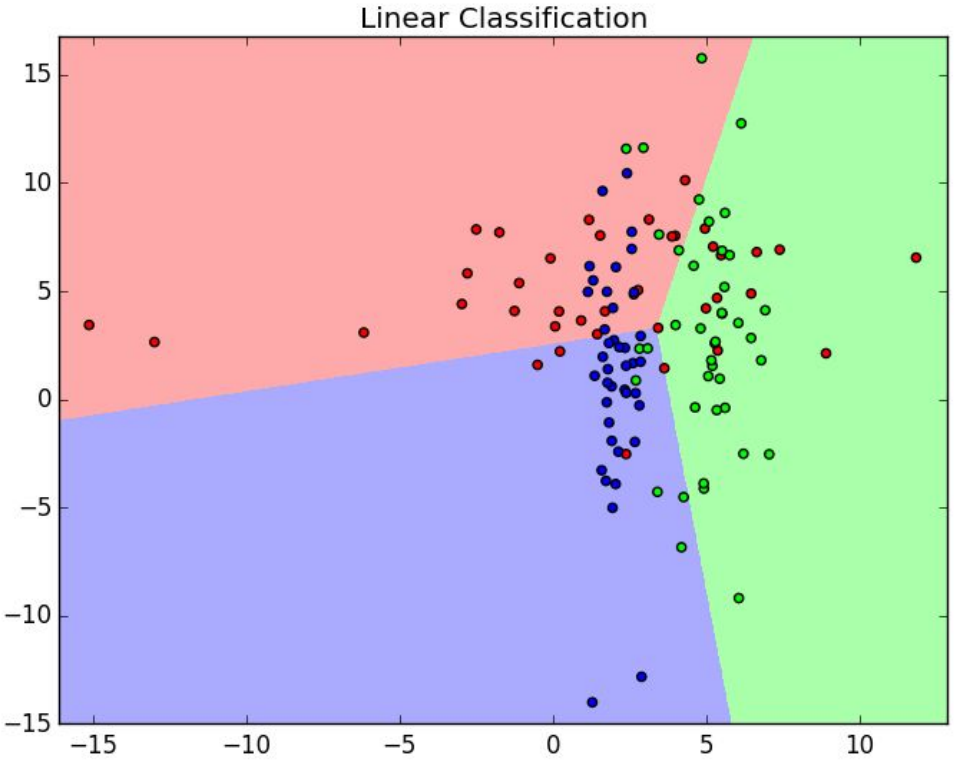
$$C_k = f(Wx + b)$$

Learning objective:

$$\hat{W}, \hat{b} = \underset{W,b}{\arg\min} \sum_{s}^{\#samples} d\left(C_k^{(s)}, f(Wx_s + b)\right)$$

For example, in binary classification:

$$f(m) = \begin{cases} 0 & m \leq 0 \\ 1 & m > 0 \end{cases}$$

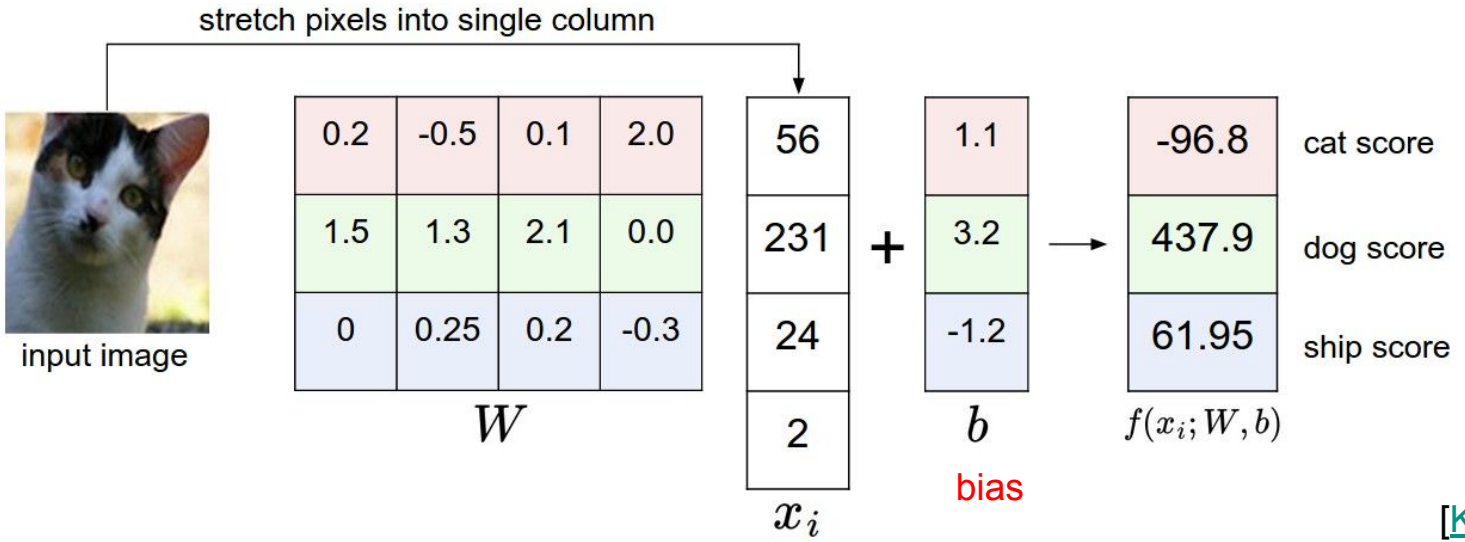


Linear Classifier

Simple image parametrization: Each pixel is a number in a very long vector.

Linear score function: $f(x_i, W, b) = Wx_i + b$

Need to learn: W and b



[Karpathy]

Neural Networks

Intro, Perceptron

MLP

Learning: Gradient Back-propagation

Biological Neuron

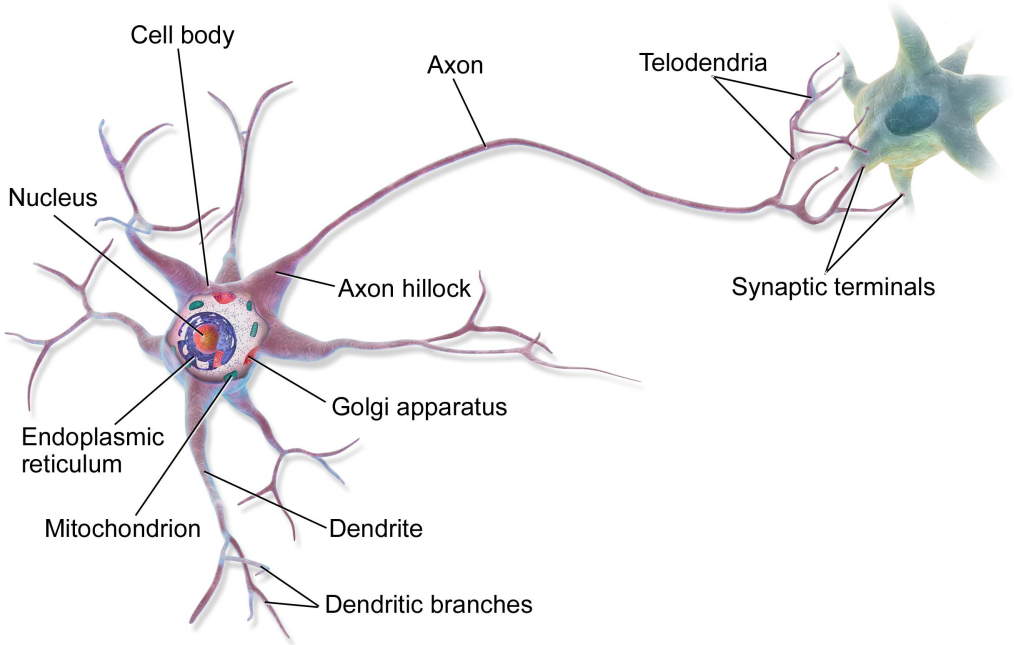
The elementary “computation unit” in the brain.

The brain may have 85 billion of them:

85,000,000,000

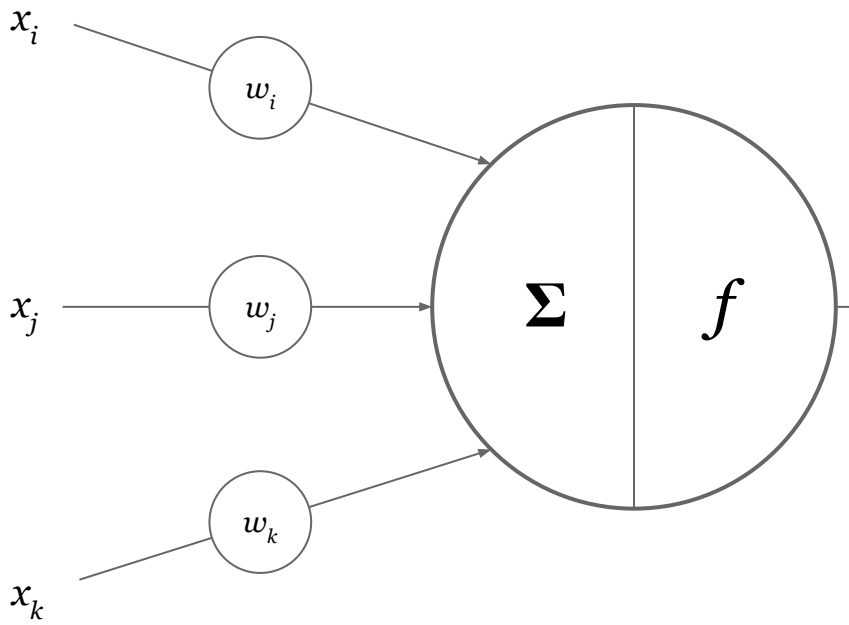
They respond to and emit (activate, fire) electrical signals.

They “learn” when to fire based on their inputs.



Artificial Neuron - Perceptron

[McCulloch & Pitts 1945]



Basically - what is this?

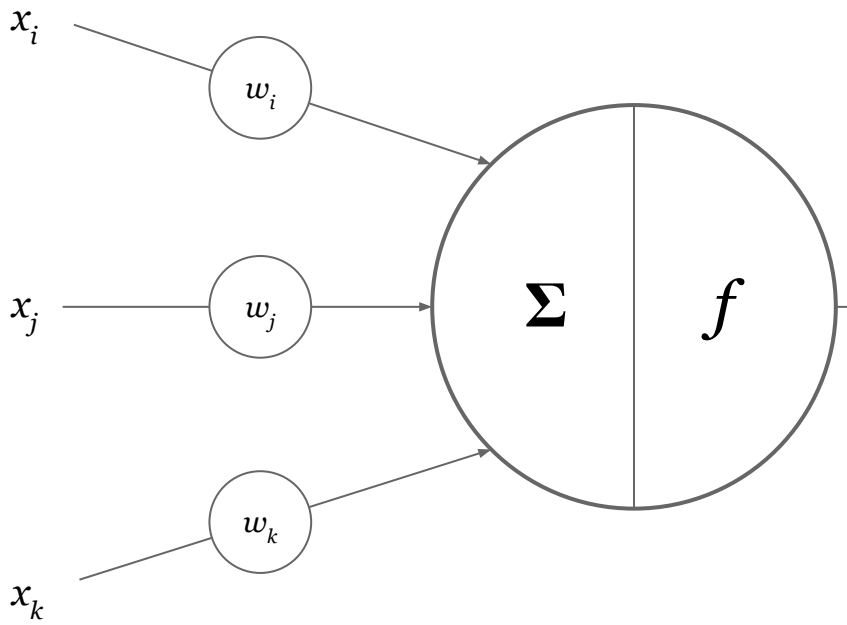
$f(w^{\mathrm{T}}x)$

f may be a Linear Threshold Unit, like:

$$heaviside(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Artificial Neuron - Perceptron

[McCulloch & Pitts 1945]



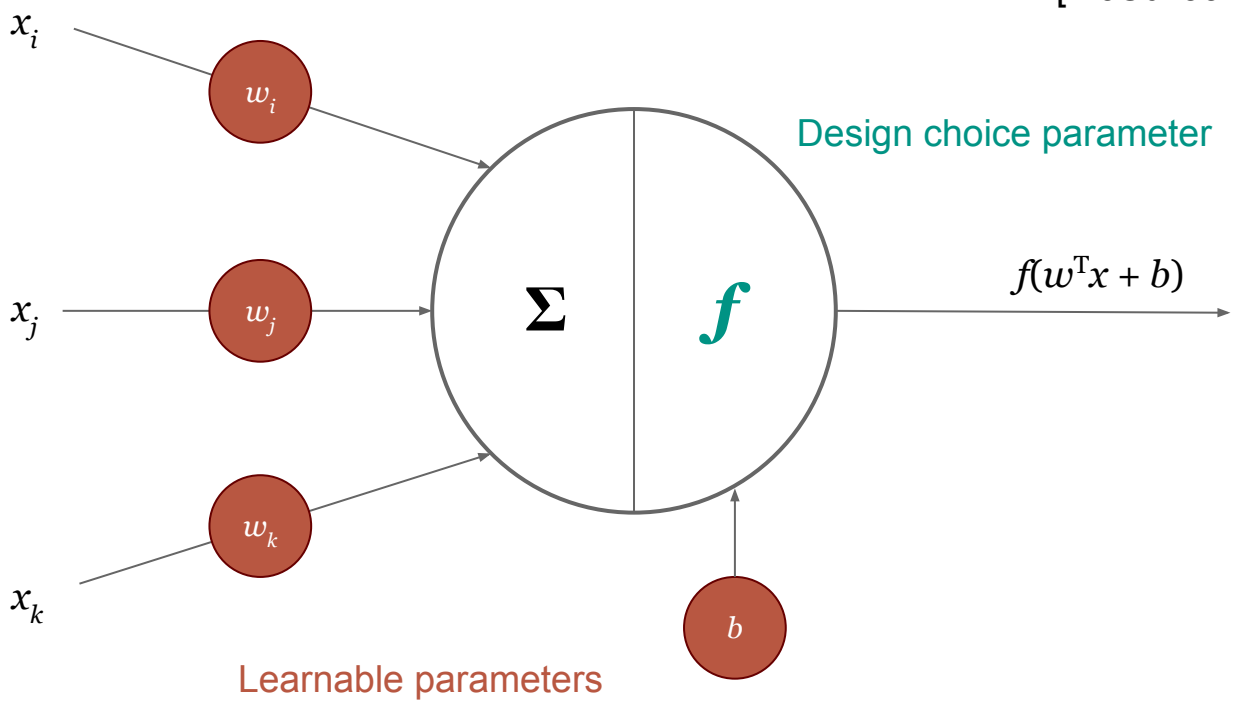
f may be a Linear Threshold Unit, like:

$$heaviside(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

A simple binary linear classifier...

Artificial Neuron - Perceptron

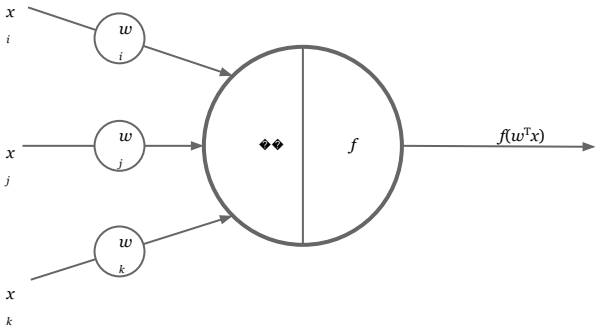
[McCulloch & Pitts 1945]

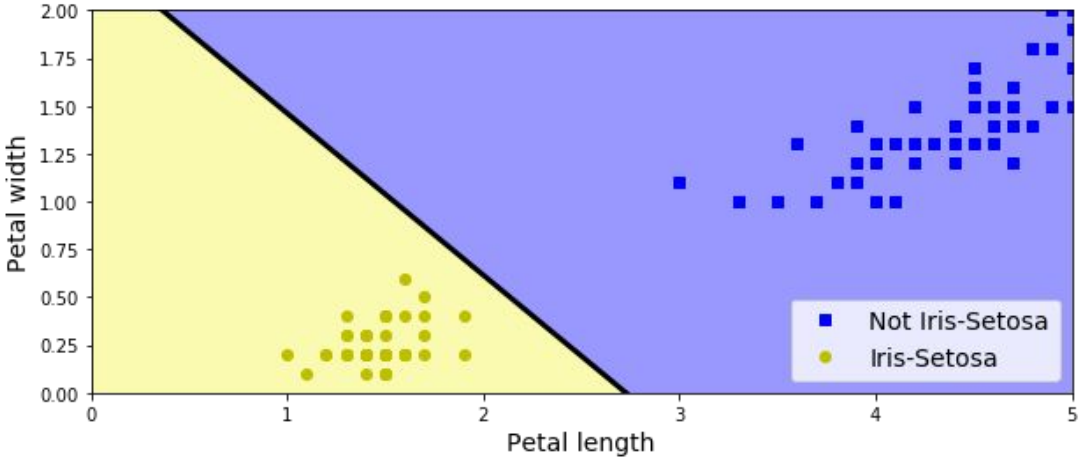


Single Perceptron

Using the loss function, can be a:

- Linear SVM (hinge, max-margin)
- Softmax (cross-entropy)
- … any linear classifier





Single perceptron with Sigmoid activation function

[Géron]

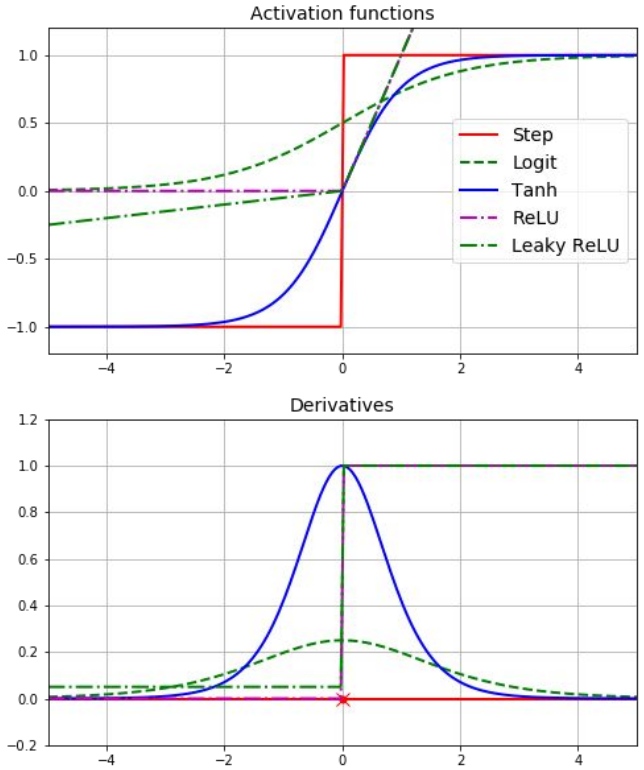
Activation Functions

Sigmoid: $\sigma(x) = 1 / (1+e^{-x})$
What is believed to be in the bio neurons. People used it for many years...

Tanh: $2\sigma(2x) - 1$. A better Sigmoid

ReLU: Rectified Linear Unit: $\max(0, x)$
Much faster to compute, converges faster.
Has a “dead zone” where gradient is 0 and can’t recover.

Leaky ReLU: $\mathbb{1}(x < 0)(\alpha x)+\mathbb{1}(x \geq 0)(x)$
A better ReLU, more robust to dying. Always has a non-zero subgradient.



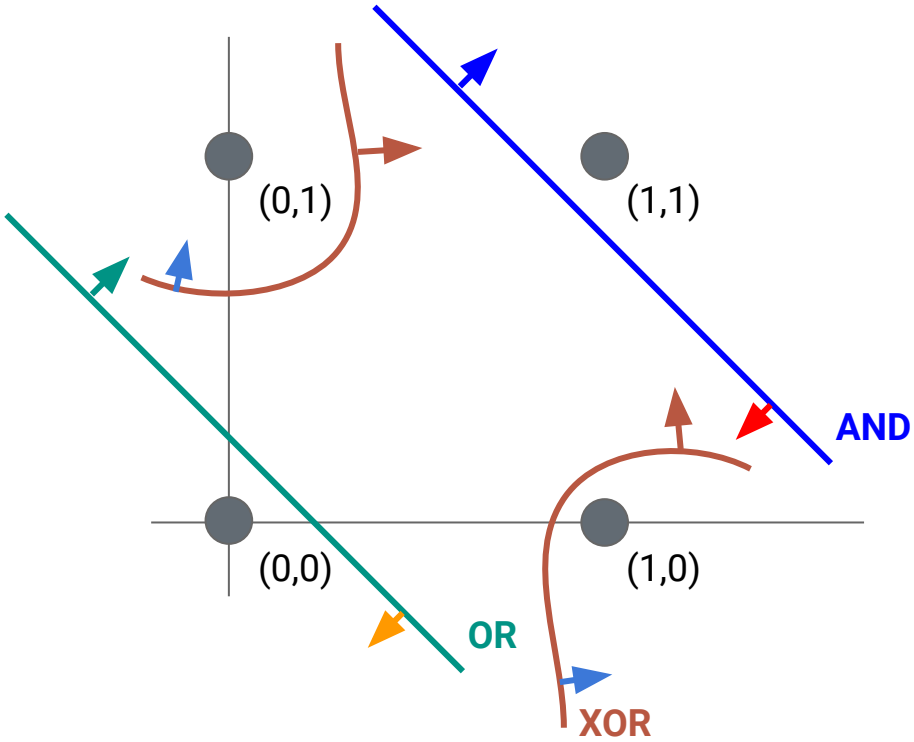
[Géron]

Neuron Problems

Marvin Minsky and **Seymour Papert** proved [1969] a single perceptron can't learn the XOR function, while it easily can learn AND and OR.

	AND	OR	XOR
0,0	0	0	0
0,1	0	1	1
1,0	0	1	1
1,1	1	1	0

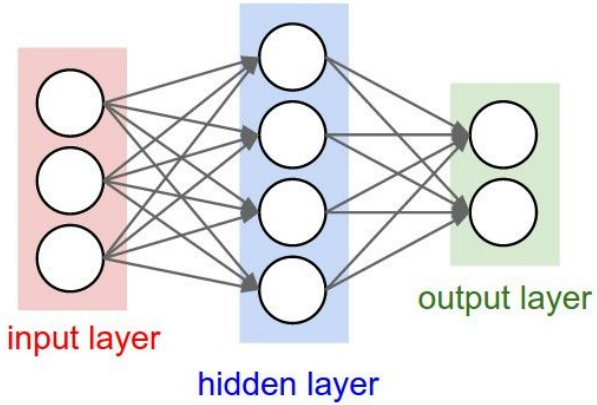
Adding **another layer of neurons** can easily solve the problem though!



Multi Layer Perceptron (MLP)

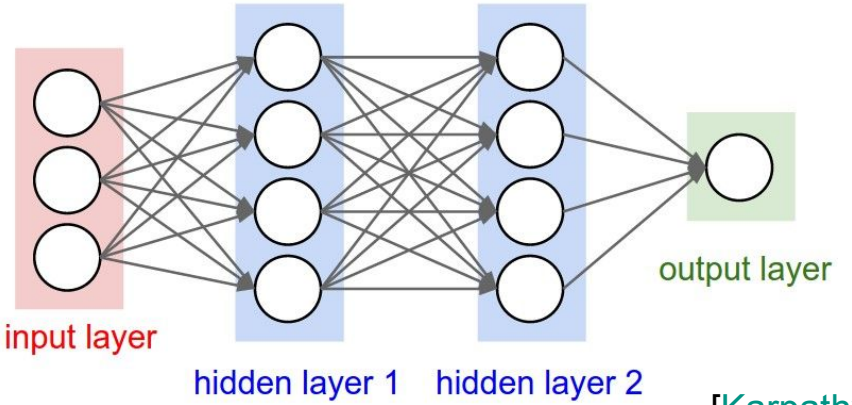
Perceptrons only start to become really powerful when they are strung together.

A single Perceptron can only model linearly separable data (like a linear SVM)



A single “hidden” layer can theoretically model any continuous function (*universal approximator*)

Multiple “hidden” layers (MLPs) are empirically better. But 4 layers is rarely better than 3 layers...



[Karpathy]

Questions?

Backpropagation Algorithm

[Werbos 1974]
Invented
Backpropagation.

[Hinton et al. 1986]

>55,000 Citations!!!

Maybe the most
Influential ML paper
Of our generation...

CHAPTER 8

Learning Internal Representations by Error Propagation

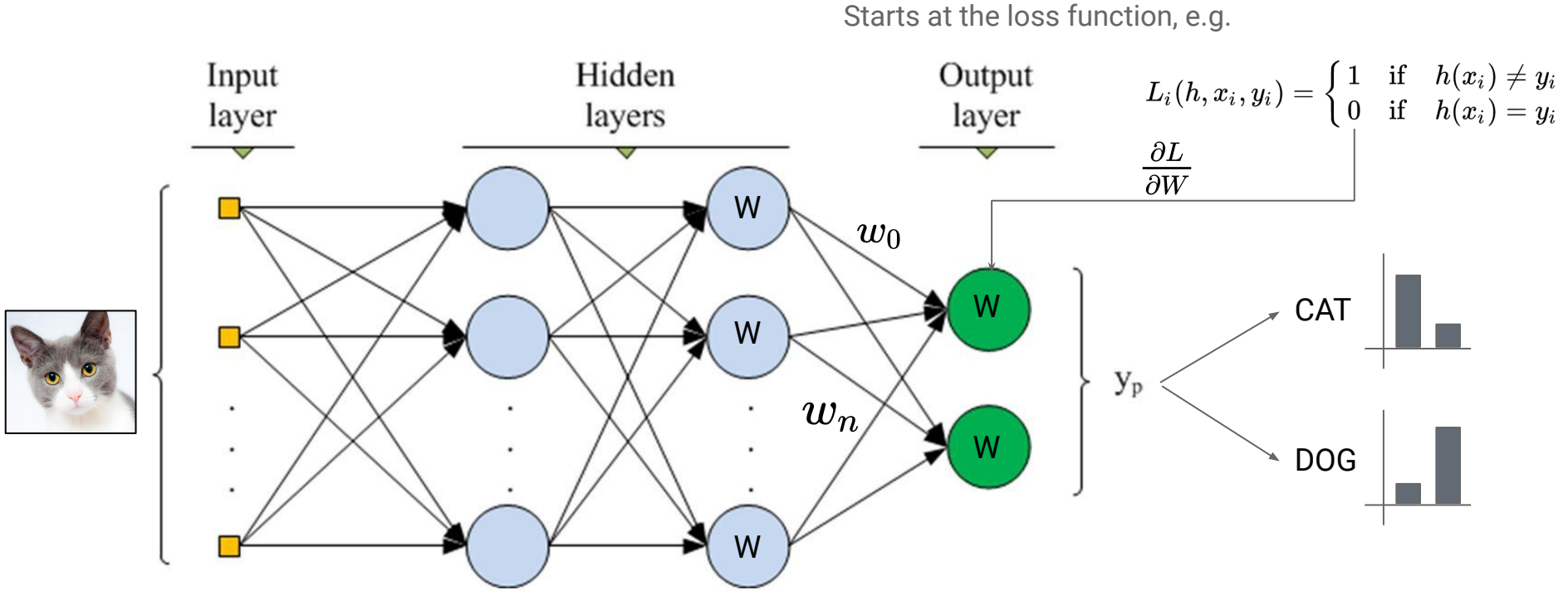
D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS

THE PROBLEM

We now have a rather good understanding of simple two-layer associative networks in which a set of input patterns arriving at an input layer are mapped directly to a set of output patterns at an output layer. Such networks have no *hidden* units. They involve only *input* and *output* units. In these cases there is no *internal representation*. The coding provided by the external world must suffice. These networks have proved useful in a wide variety of applications (cf. Chapters 2, 17, and 18). Perhaps the essential character of such networks is that they map similar input patterns to similar output patterns. This is what allows these networks to make reasonable generalizations and perform reasonably on patterns that have never before been presented. The similarity of patterns in a PDP system is determined by their overlap. The overlap in such networks is determined outside the learning system itself—by whatever produces the patterns.

The constraint that similar input patterns lead to similar outputs can lead to an inability of the system to learn certain mappings from input to output. Whenever the representation provided by the outside world is such that the similarity structure of the input and output patterns are very different, a network without internal representations (i.e., a

Gradient in Neural Networks

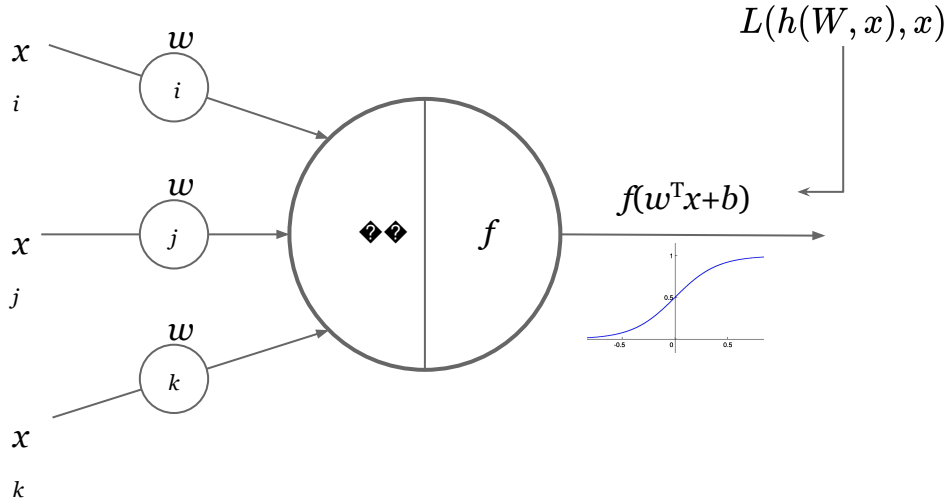


Learning Using The Gradient

Learnable parameters: W and b

We know the loss function, and we have training samples with labels x, y

But how to calculate the “derivative” of our Perceptron?

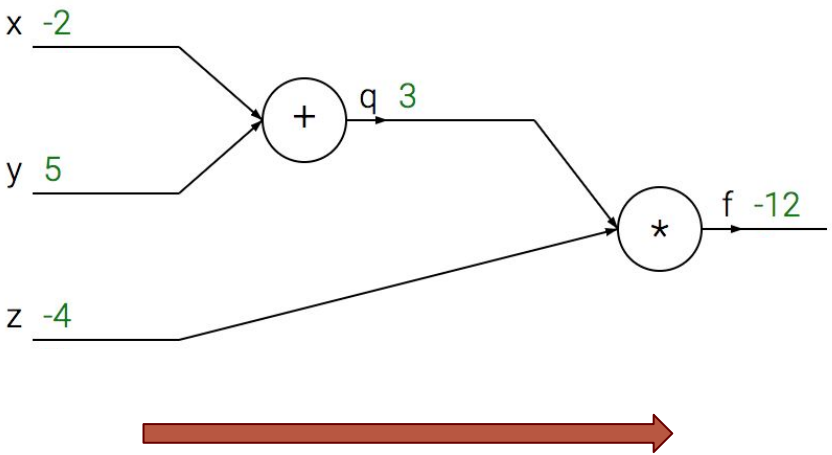


If we can spell out the perceptron derivative we can use **gradient descent** and update the weights:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \frac{1}{n} \sum_i \nabla_{\mathbf{w}} L(h_{\mathbf{w}^{(t)}}(x_i), y_i)$$

Backpropagation Algorithm

For example: $f(x, y, z) = (x + y)z$



Most if not **all** of the calculations we make can be expressed as these “computation graphs”, certainly MLPs and NNs.

A NN neuron is simply $f(\vec{w}, b, a, \vec{x}) = a(\vec{w} \cdot \vec{x} + b)$
For example, using a ReLU activation:

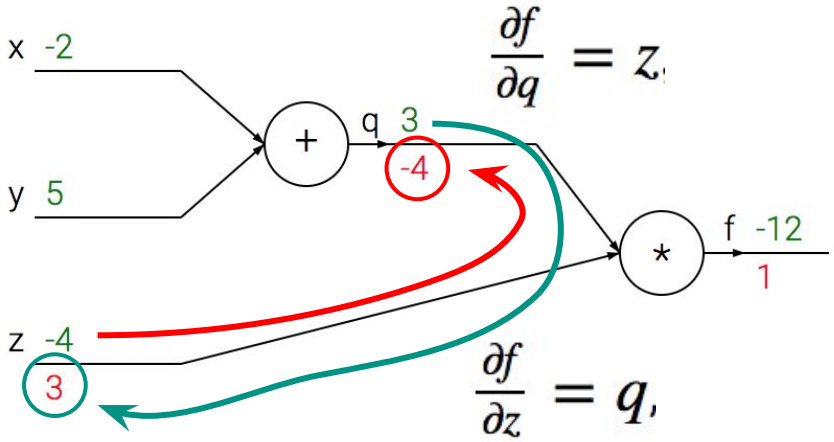
$$f(\vec{w}, b, \vec{x}) = \begin{cases} \vec{w} \cdot \vec{x} + b & \text{if} \quad \vec{w} \cdot \vec{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

First: forward pass.

[Karpathy]

Backpropagation Algorithm

For example:



Then start going backwards from the top.

Say e.g. the loss function determines f should change by 1, i.e. $\frac{\partial L}{\partial f} = 1$.

$f = qz$, so: $\frac{\partial (q \cdot z)}{\partial q} = z, \frac{\partial (q \cdot z)}{\partial z} = q$

Now how do we compound the gradient from q backwards to x and y ?

[Karpathy]

Chain Rule

Our perceptron is made of complex operations chained together.

For example: $f(x, y, z) = (x + y)z$

Break it down: $q = x + y \qquad f = qz$

If f is just a multiplication we can easily say:

$$\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

For the addition in q:

$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

The Chain Rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Chaining derivatives is just a multiplication of the partial derivatives.

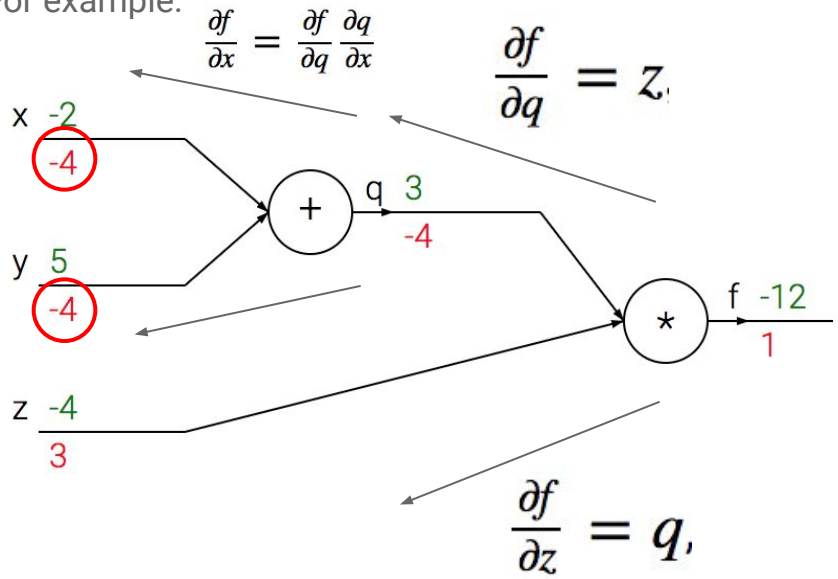
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Insight: If we can express the calculation forward in a computation graph, we can also express the gradient.

[Karpathy]

Backpropagation Algorithm

For example:



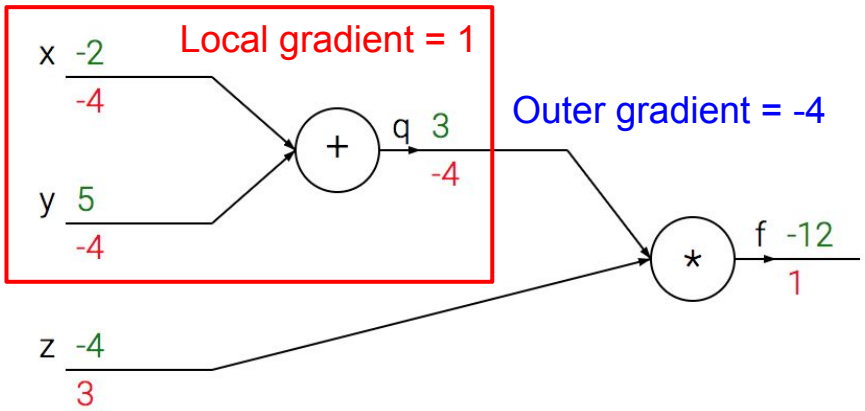
Then start going backwards from the top.

Use the chain rule to compound the gradients.

[Karpathy]

Backpropagation Algorithm

For example:



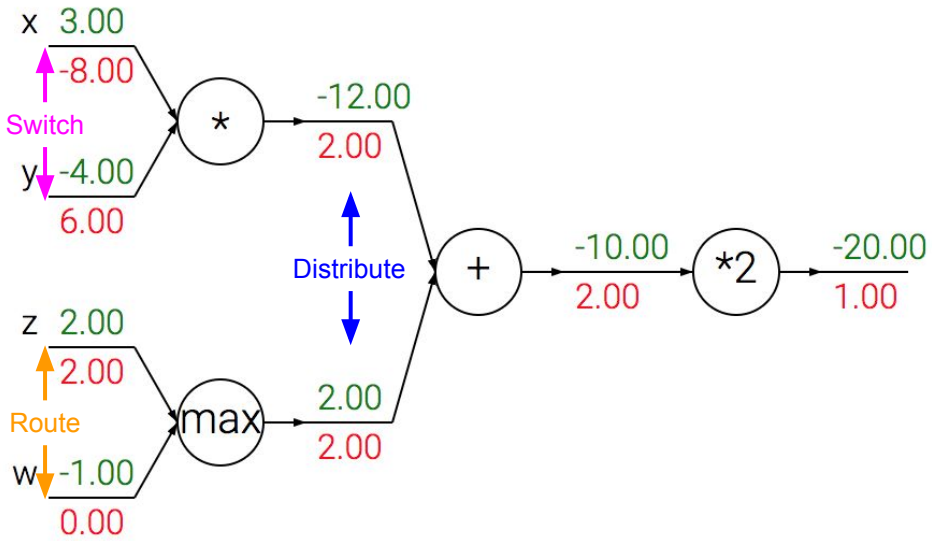
Notice that individual gates are self sufficient.

They can calculate their outputs, and also their *local gradients*.

The chain rule determines the global gradient - multiply the outer gradient with the local.

[Karpathy]

Backpropagation in Compute Graphs



add, **multiply** and **max** gates are the most common in neural networks.

And their gradients are very simple:

Add - Distribute the outer gradient

Multiply - Switch the inputs and chain the outer.

Max - Route the outer gradient to the bigger value.

[Karpathy]

Convolutional Neural Nets

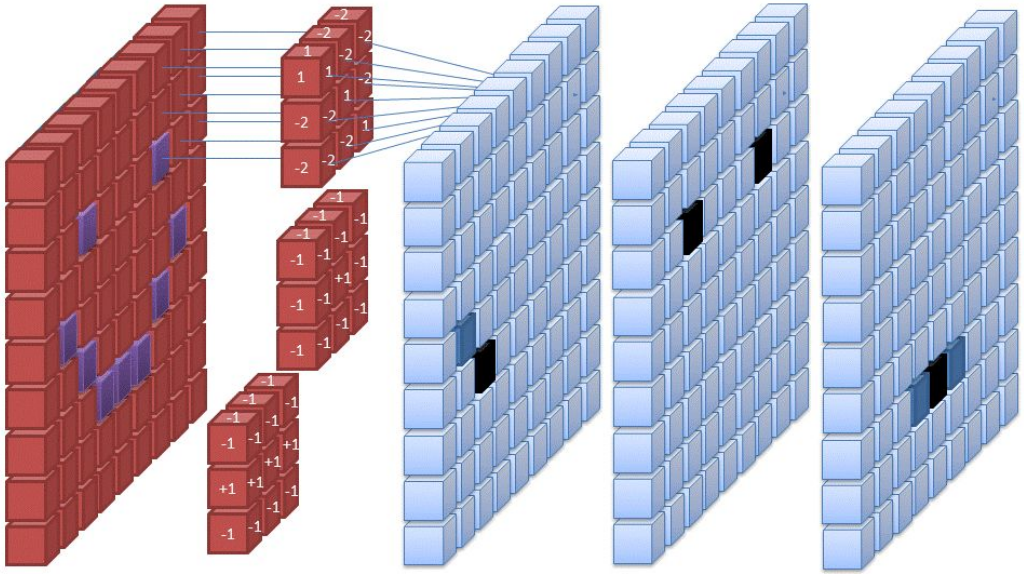
History, Intro

Convolutional Perceptrons

Pooling

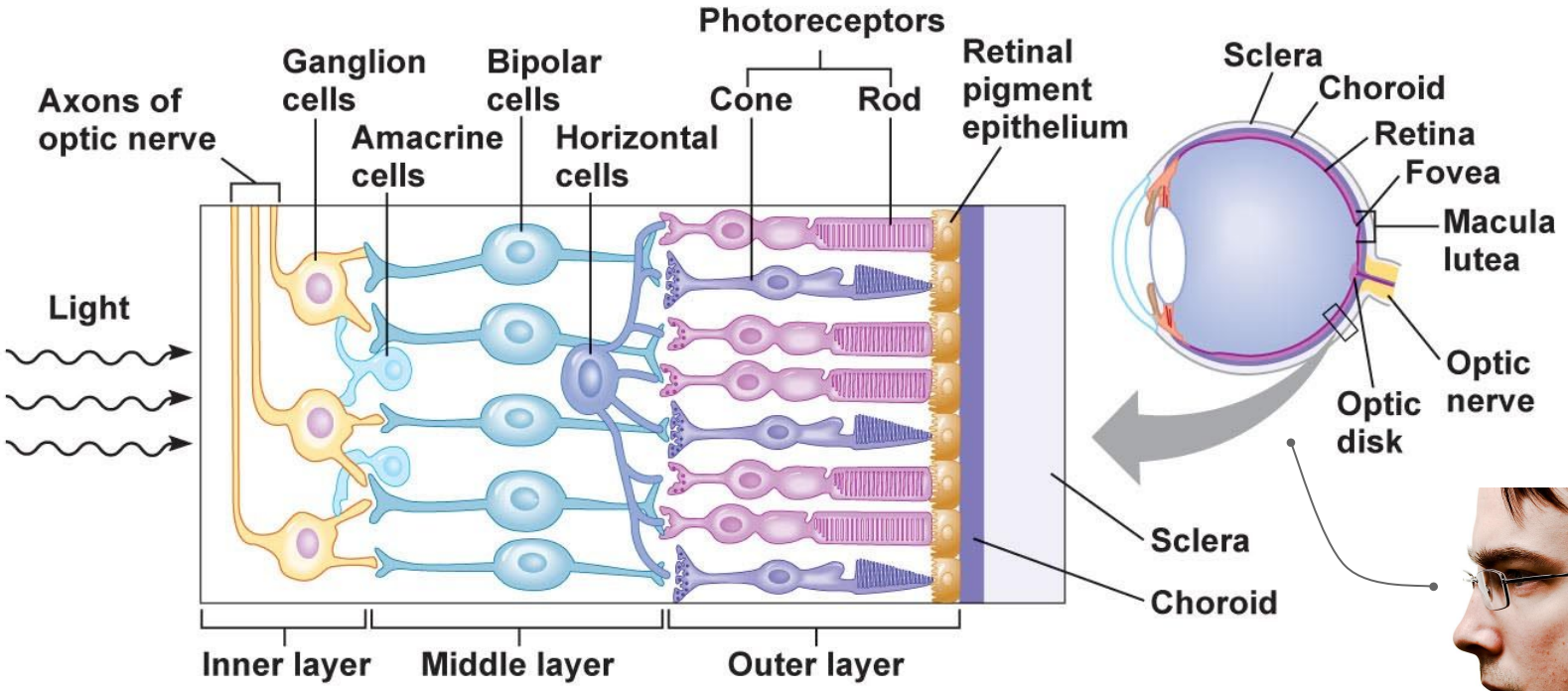
Activation maps

ConvNets changing the world



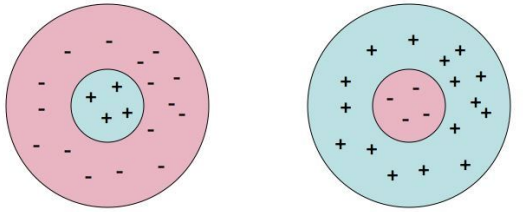
Retina | Layers, Rods and Cones

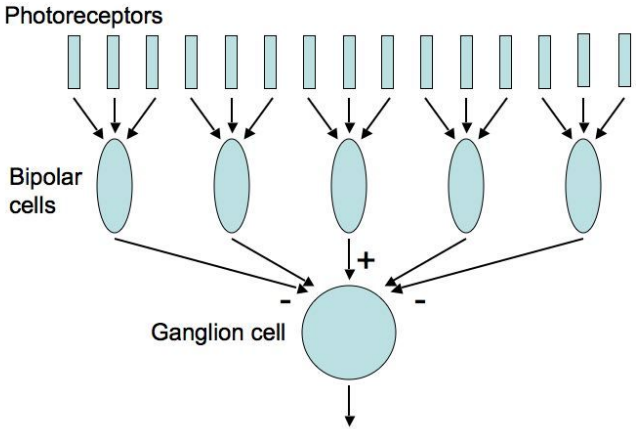
RECALL



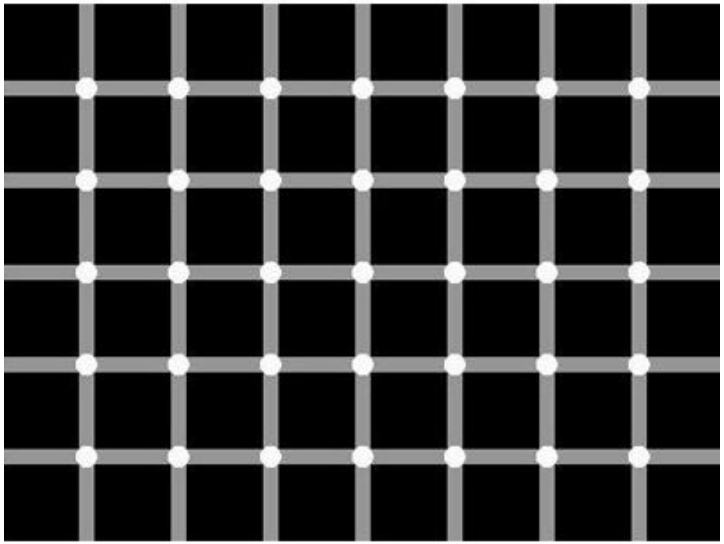
Lateral Inhibition | Receptor "AND Gates"

RECALL





Count all the black dots you can see.

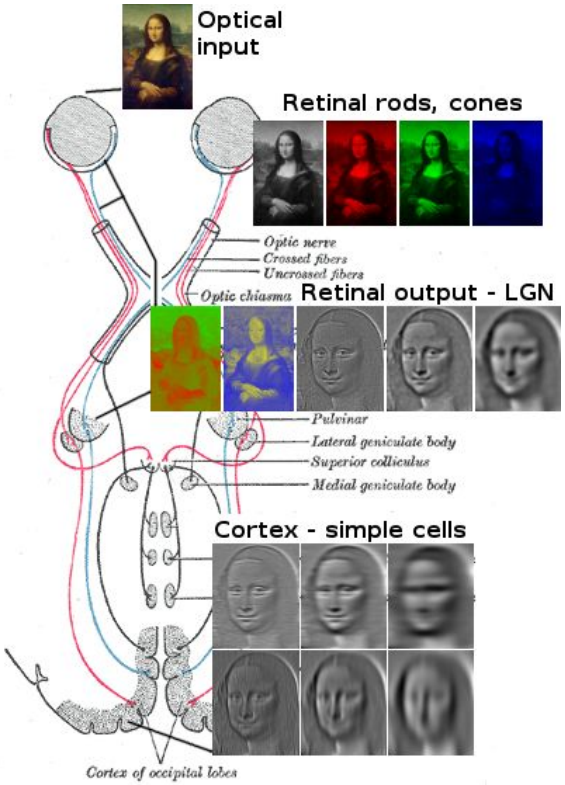


Answer: There are no black dots

If you focus directly on each dot, you'll see that all of them are white.

Visual Processing System | Hierarchy

RECALL



V1
- Speed
- Direction
- Spatial frequency
- Temporal frequency
- Orientation
- Color

V2
(thick stripes)
- Speed
- Direction
- Spatial frequency
- Temporal frequency

MT
- Speed
- Direction
- Spatial frequency
- Temporal frequency
- Local/Global motion

MST
- Expansions
- Contractions
- Rotations
- Translations
- Spirals
- Optic Flow

Parietal Regions
- Heading
- Optic flow
- Self motion
- Multi-modal integration

V2
(thin stripes, interstripes)
-Edges
-Illusory edges
- Border ownership
- Color

V4
- Angles
- Curvature
- Perceived color
- Kinetic contours
- Motion

TEO/PIT
- Simple shapes

TE/AIT
- Complex shapes/ body parts
- Object recognition
- Object invariance

Visual Cortex

Huber & Wiesel, 1950s

Researched the levels of the visual cortex organization in cats.
(Warning: cat experiment images may be disturbing)

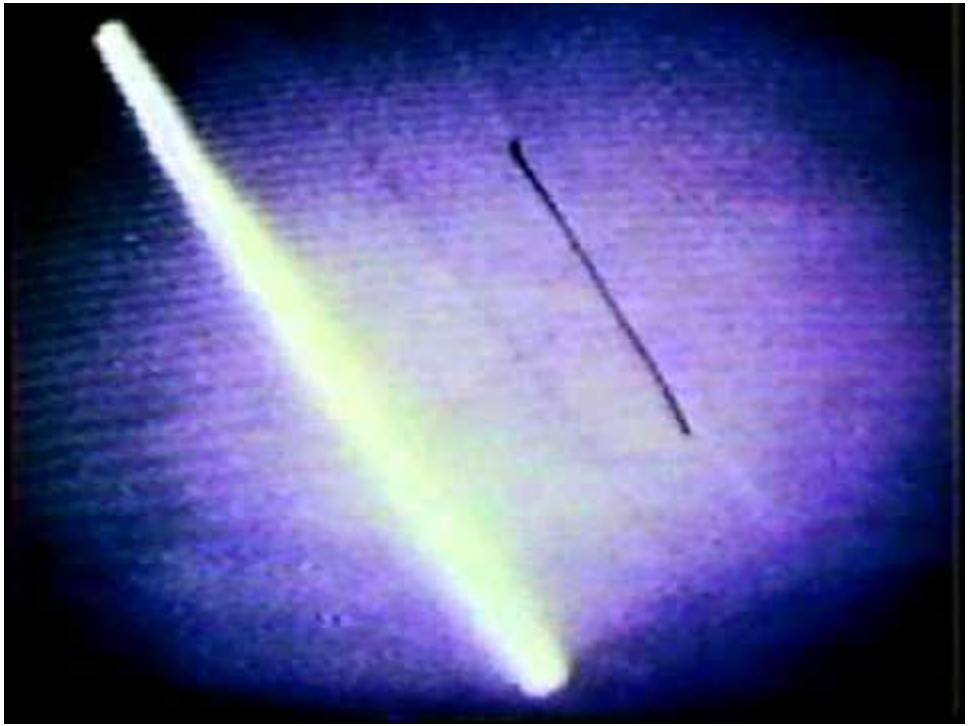
Found 3 types of cells:

- Simple cells: response to orientation
- Complex cells: Orientation and Motion
- Hypercomplex cells: Orientation, Motion and simple shape (circle, short line, etc.)

[Video]

Visual Cortex

They won a **nobel prize**!



[Video]

Visual Neural Networks

Y. LeCun and Y. Bengio, 1990s: researching the application of hierarchical organization of learnable convolutions in neural networks.

"Gradient-based learning applied to document recognition." [1998]

~17,500 citations!!!

“LeNet-5”·

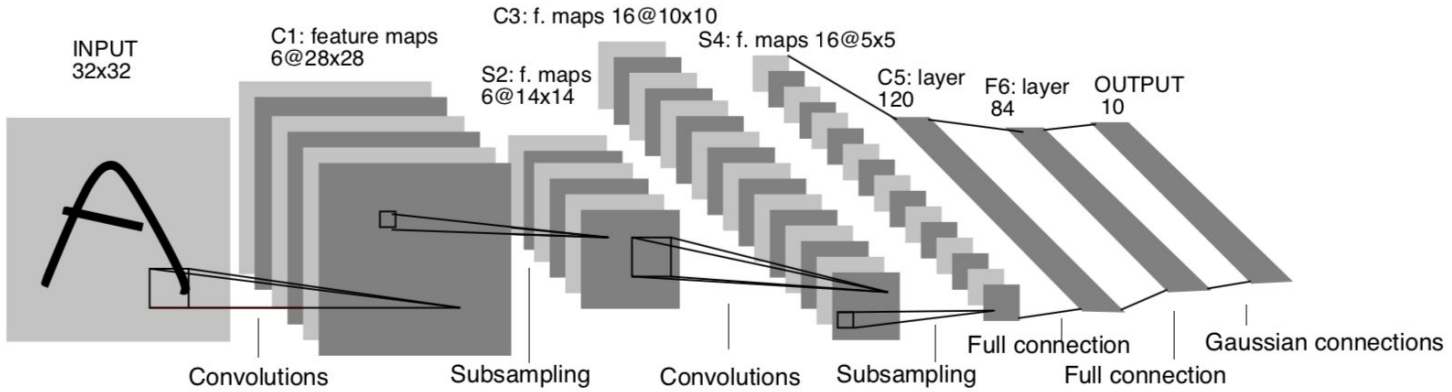


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Convolutional Neural Networks

The big ideas in [LeCun and Bengio 1998]:

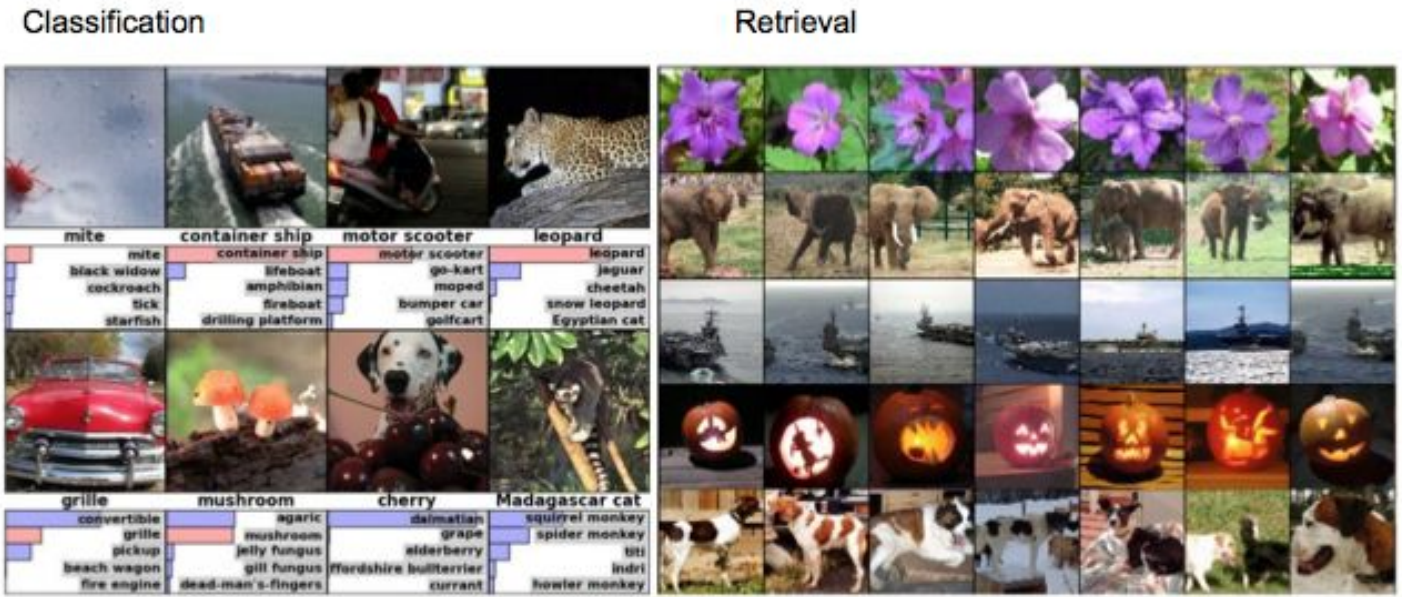
- From locally (fully) connected neurons to “shift invariant” convolutional neurons.
- “weight sharing”
- Spatial sub-sampling: “Pooling”

Similar initial ideas, e.g. in Fukushima’s “Neocognitron” [1980], predated backprop and gradient-based training was virtually impossible.

Convolutional Neural Networks

[Fei-Fei Li]

Fast-forward to today: ConvNets are everywhere



Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Convolutional Neural Networks

Fast-forward to today: ConvNets are everywhere

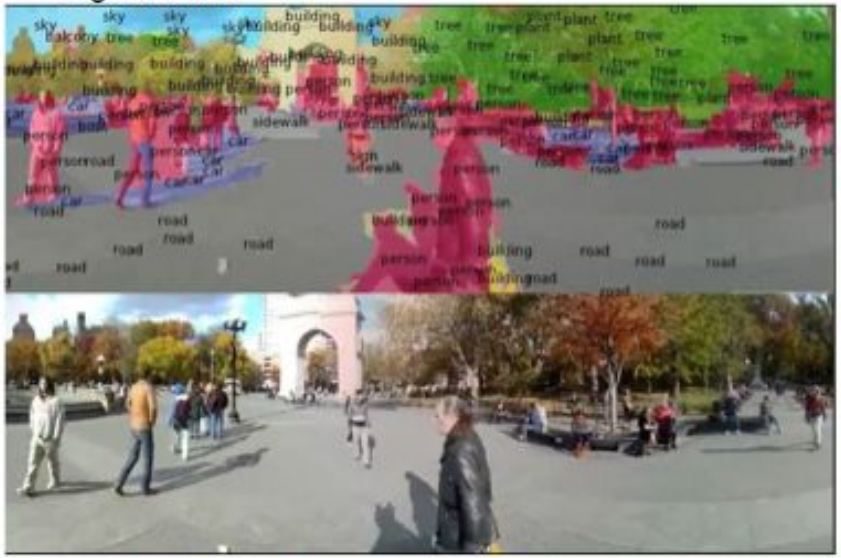
ei-Fei Li]

Detection

Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

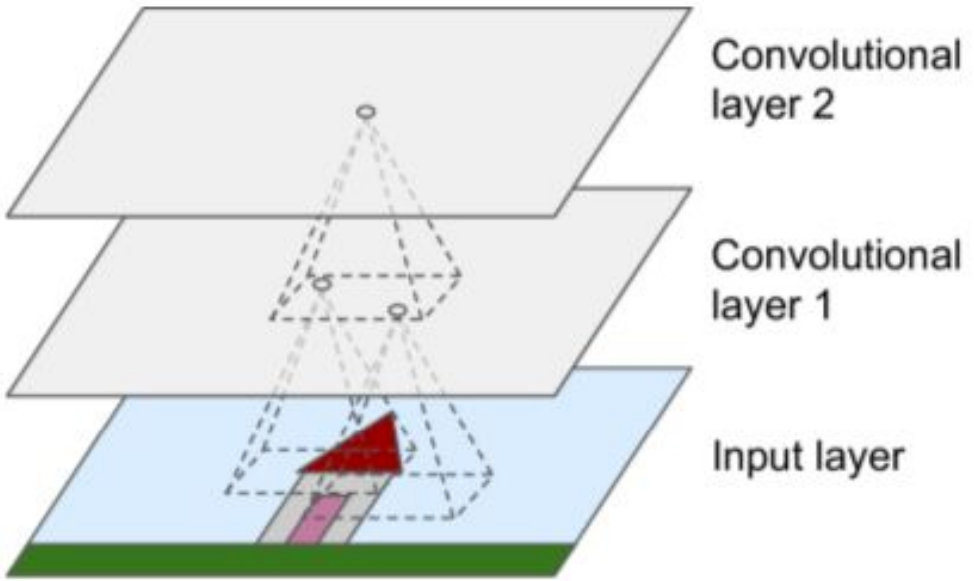
Convolutional Layers

"Fully Connected Layers": neurons are connected (Receptive Field) to all pixels in the input layer.

"Convolutional Layers": neurons have just a NxN receptive field, which is applied spatially across all the input pixels, **sharing the weights**.

"Shift invariance" of convolutions comes into play. Visual structures can be recognized *anywhere in the image*.

Conv layers reduce the number of trainable parameters by many orders of magnitude...



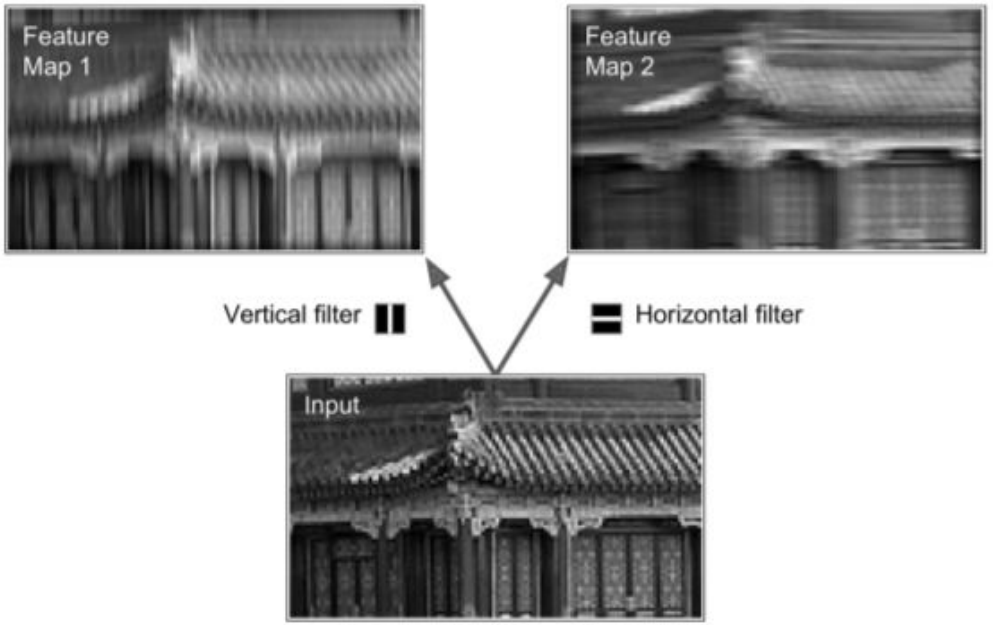
[Géron]

Learned Convolutions

Each Conv layer learns a set of weights - (multiple) convolution kernels.

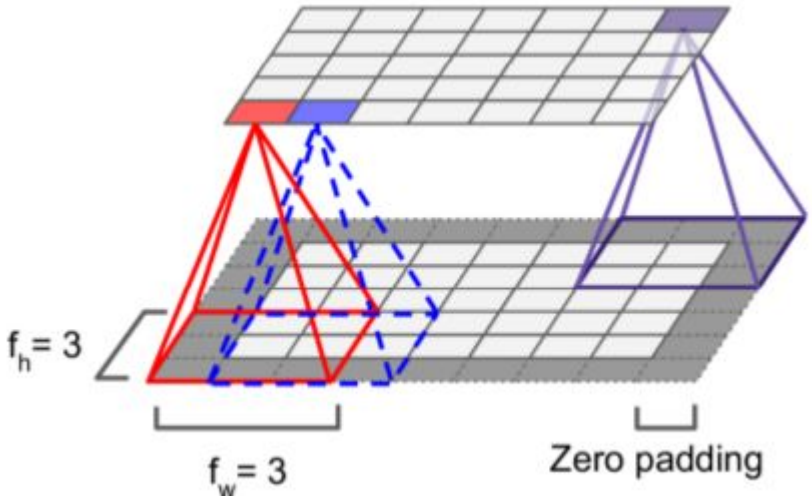
The convolution operation is same as the linear perceptrons: Wx_i

Backpropagation adjusts the weights, and the layer learns the best convolution to fit the loss function minimization.

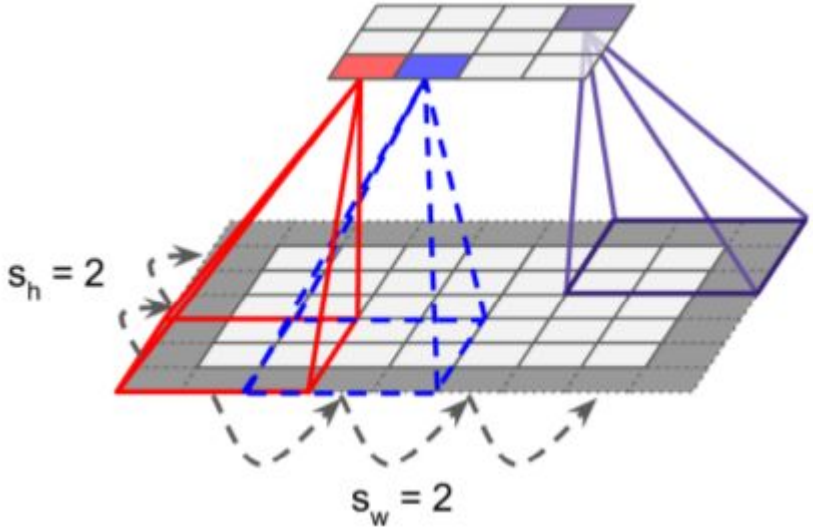


[Géron]

Padding and Stride



Border pixels are missing values for receptive field of 3.
Output layer will be smaller, or we add padding.



Stride can be used to reduce the size of the output of convolutional layers.

[Géron]

Feature Maps

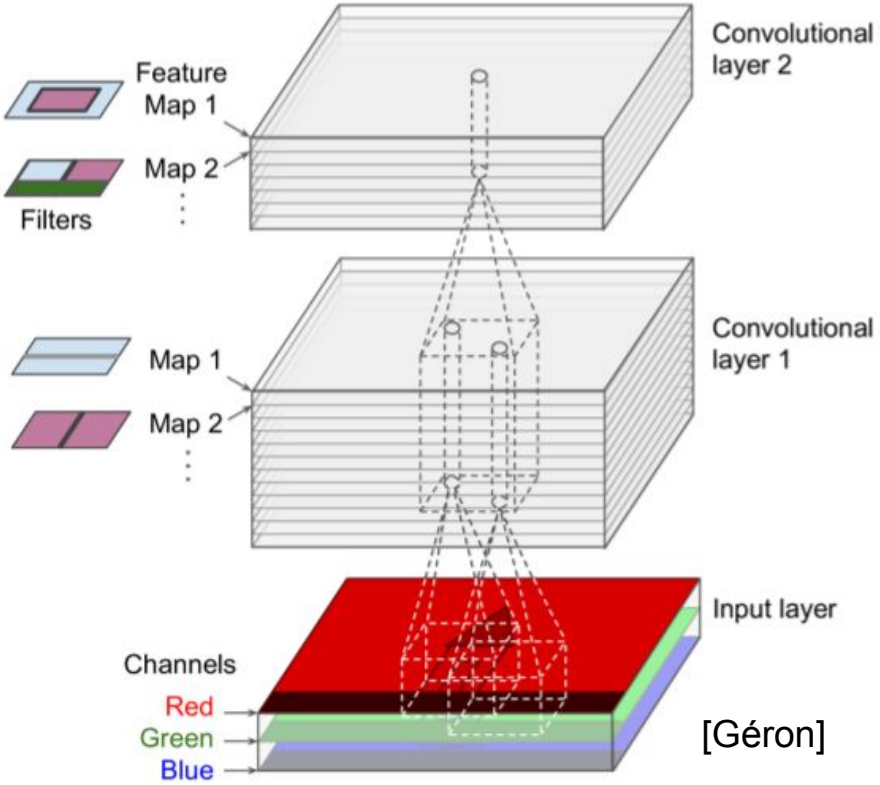
Each neuron contributes a convolved output image (local feature extraction).

The result of a convolutional layer is a stack of all those maps.

Result: A visual structure (e.g. “face”) can be detected anywhere in the image.

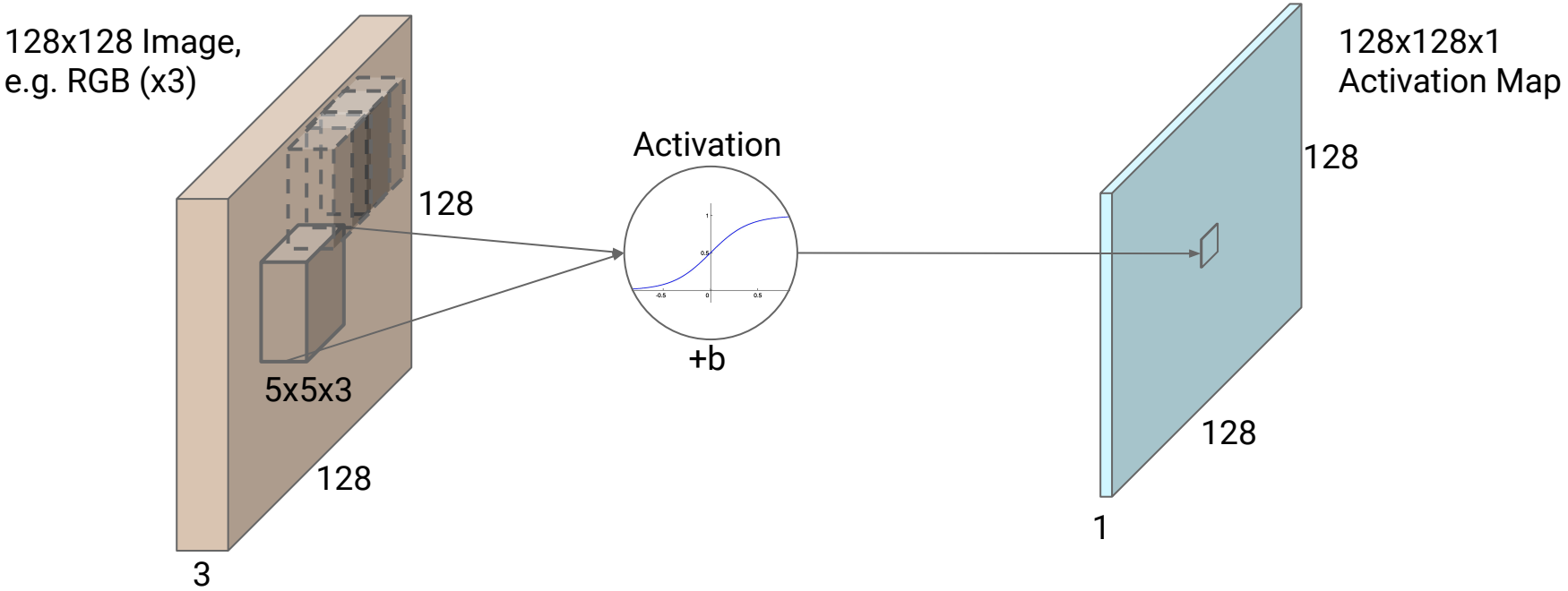
Successive layers apply to *all lower layers*.
This can quickly blow up in dimension!

“Activation map”: Applying activation (sigmoid, tanh, ReLU,...) on the feature map.



[Géron]

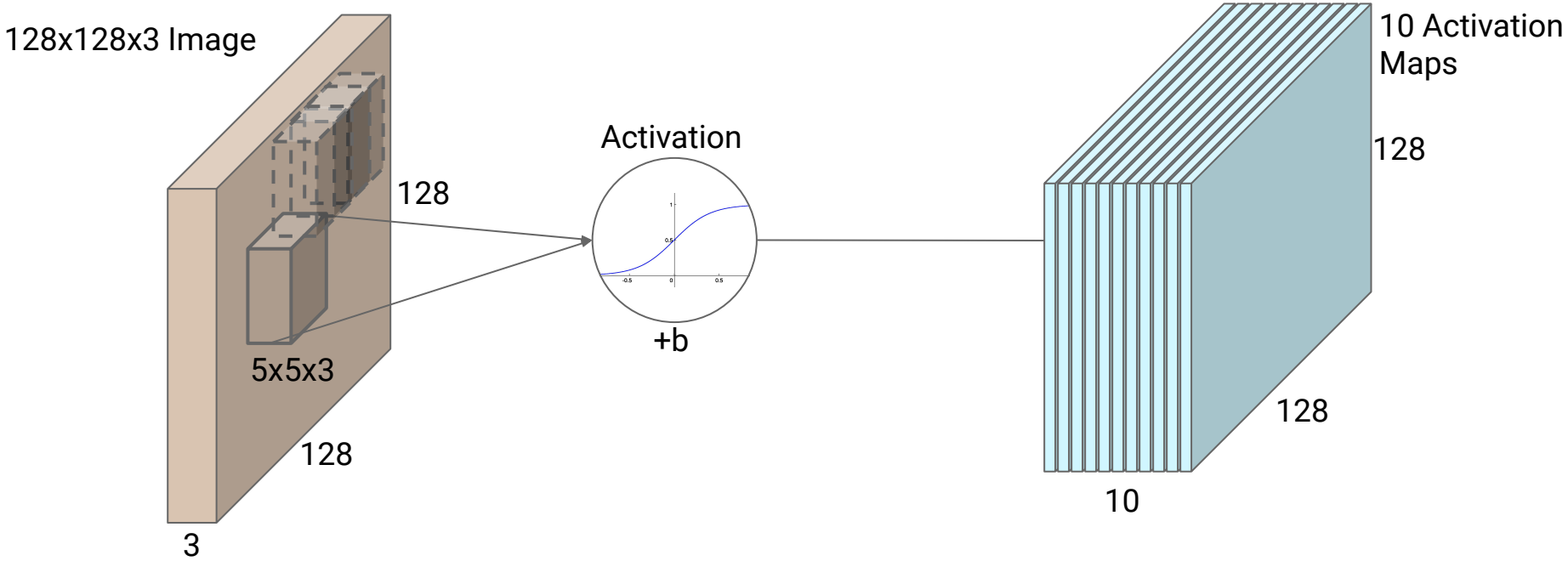
Convolutional Layer



Learnable parameters: 5x5x3+1 = 76

With “Same” padding, output size is maintained

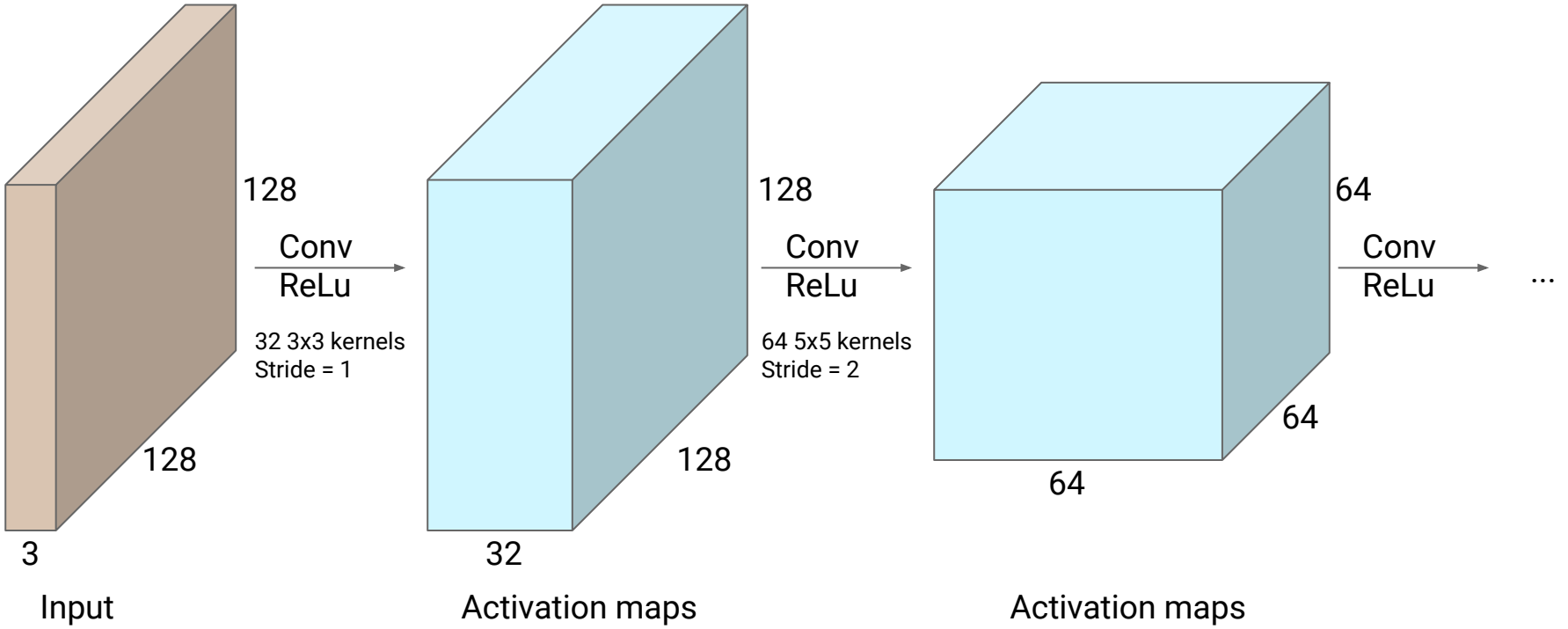
Convolutional Layer



Learnable parameters: (5x5x3+1)x10 = 760

Usually we learn many convolution kernels at each layer...

Fully Convolutional Networks



Questions?

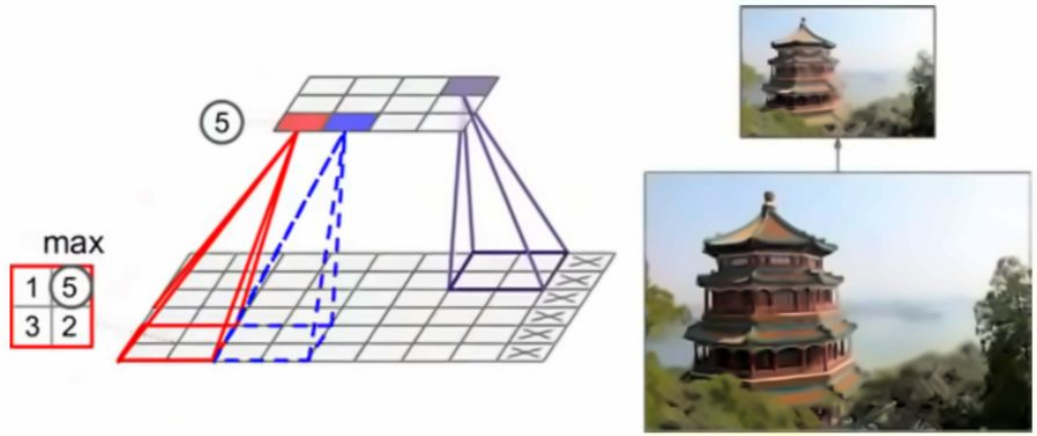
Pooling Layers

To prevent this blow up we apply “pooling”.

Every neuron is connected to a small receptive field and outputs a single output.

Many kinds: max, min, average, ...

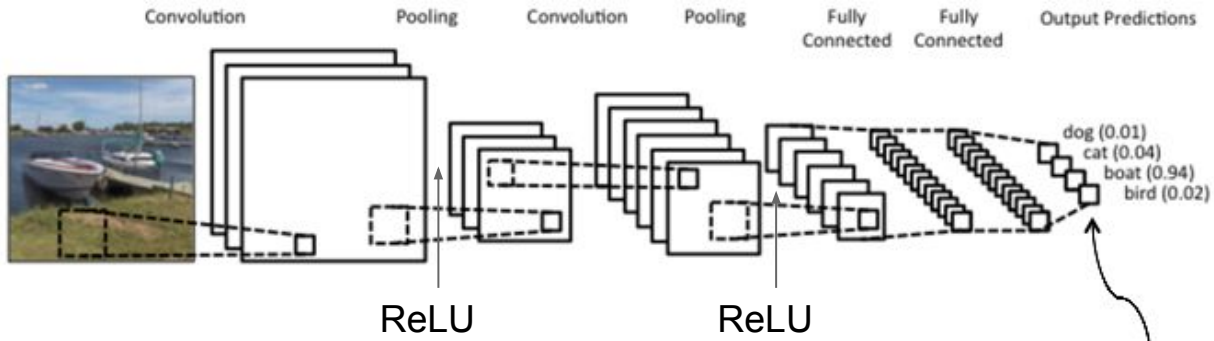
It also adds a little shift invariance.



max-pool layer

CNN Architectures

Typical Classification CNN:



The last layer applies the cross-entropy loss (Softmax).

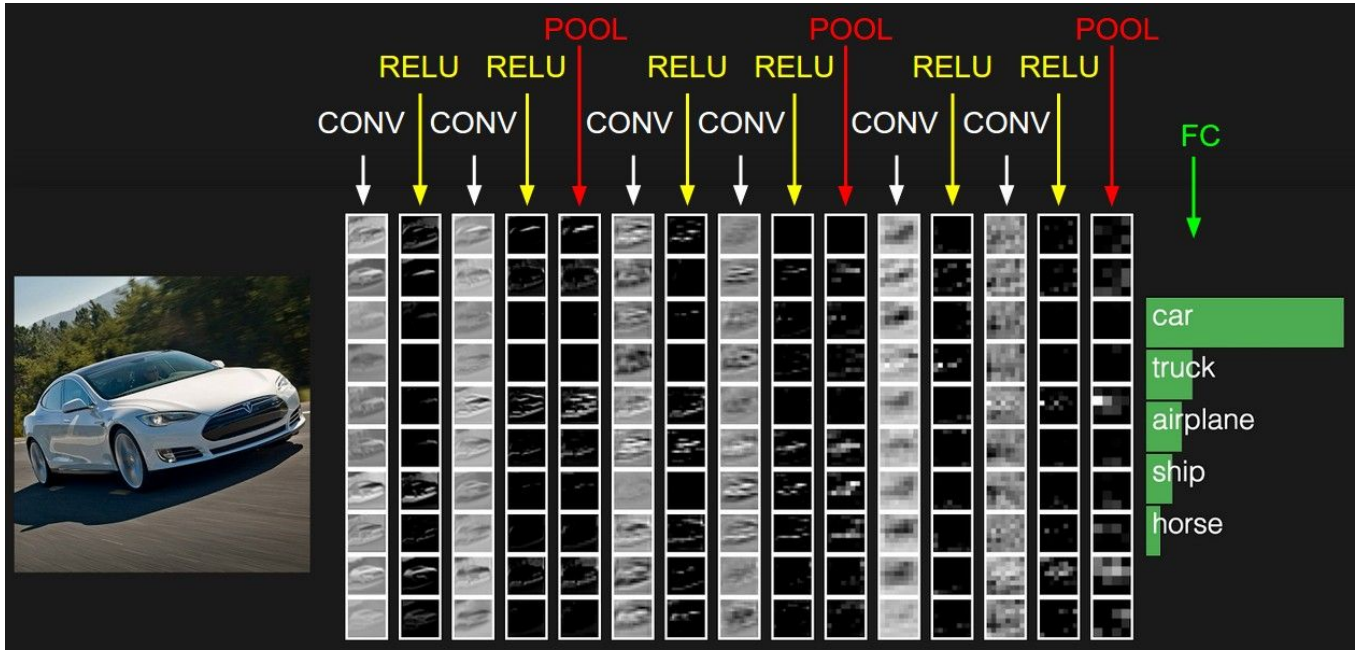
Softmax layer:

$$P(c \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_c \cdot \mathbf{x})}{\sum_{k=1}^{C} \exp(\mathbf{w}_k \cdot \mathbf{x})}$$

Each conv-pool pair makes the image smaller but deeper, learning more complicated visual constructs.

[Lazebnik]

What Does A CNN “See”?



[DEMO]

[Visualization]

[Activation Atlas]

Famous Architectures

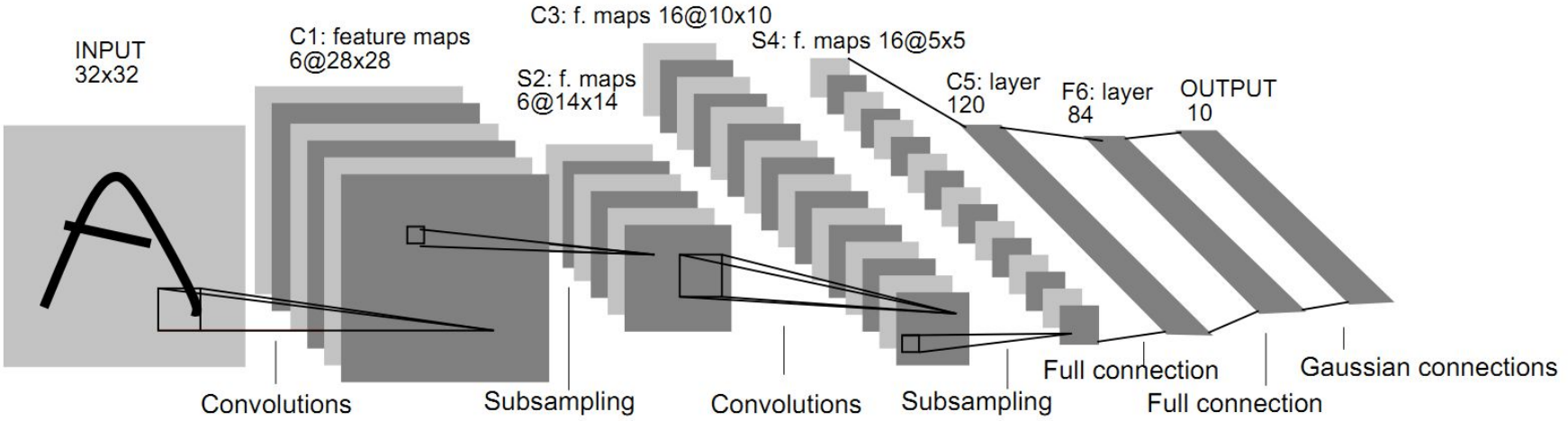
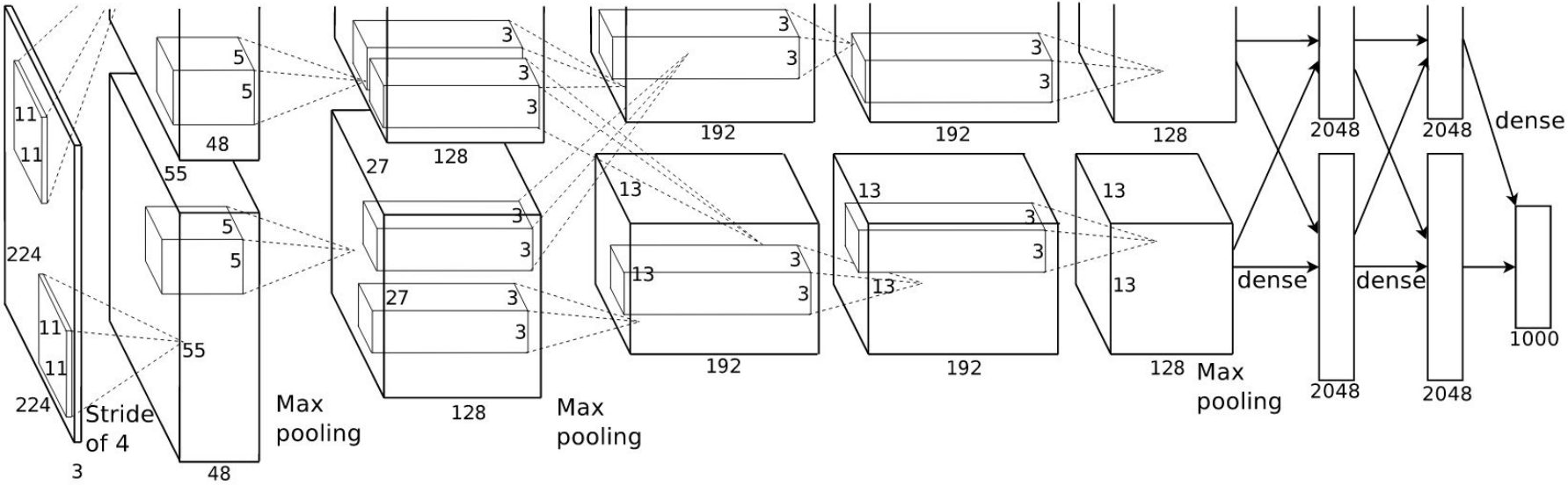


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

"LeNet", LeCun 1998. Achieved 98% accuracy on MNIST (handwritten digit classification)

Famous Architectures



"AlexNet", Krizhevsky + Hinton 2010. Achieved ~15% top-5 error rate on ImageNet Large Scale Visual Recognition challenge (ILSVRC). Beat the 2nd runner up by more than 10%!

ILSVRC Image Classification (CLS) Task

Steel drum

[ILSVRC 2017]

1000 object classes 1,431,167 images CLS-LOC

ILSVRC Image Classification (CLS) Task

Steel drum

[ILSVRC 2017]

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

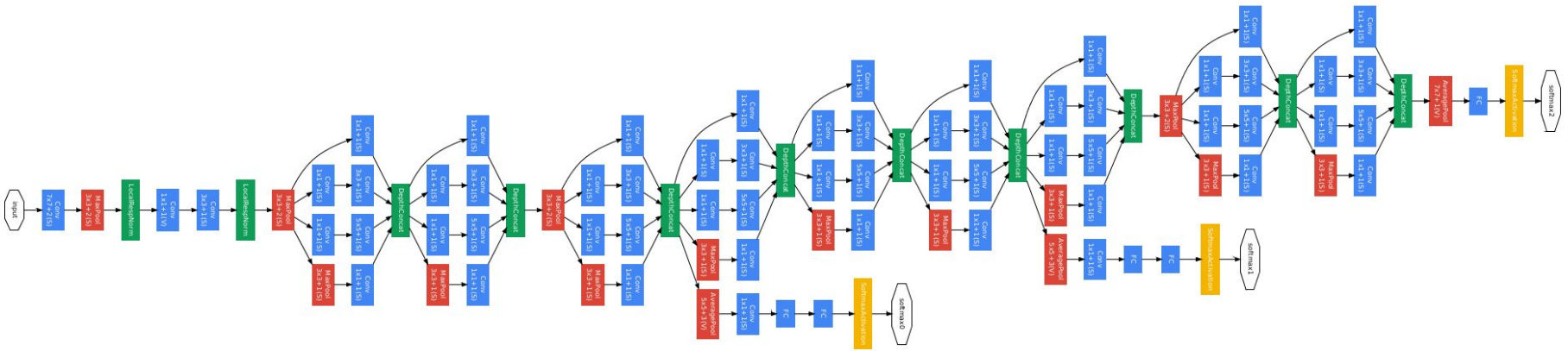
✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

$$\text{Error} = \frac{1}{100{,}000} \sum_{\substack{100{,}000 \\ \text{images}}} 1[\text{incorrect on image i}]$$

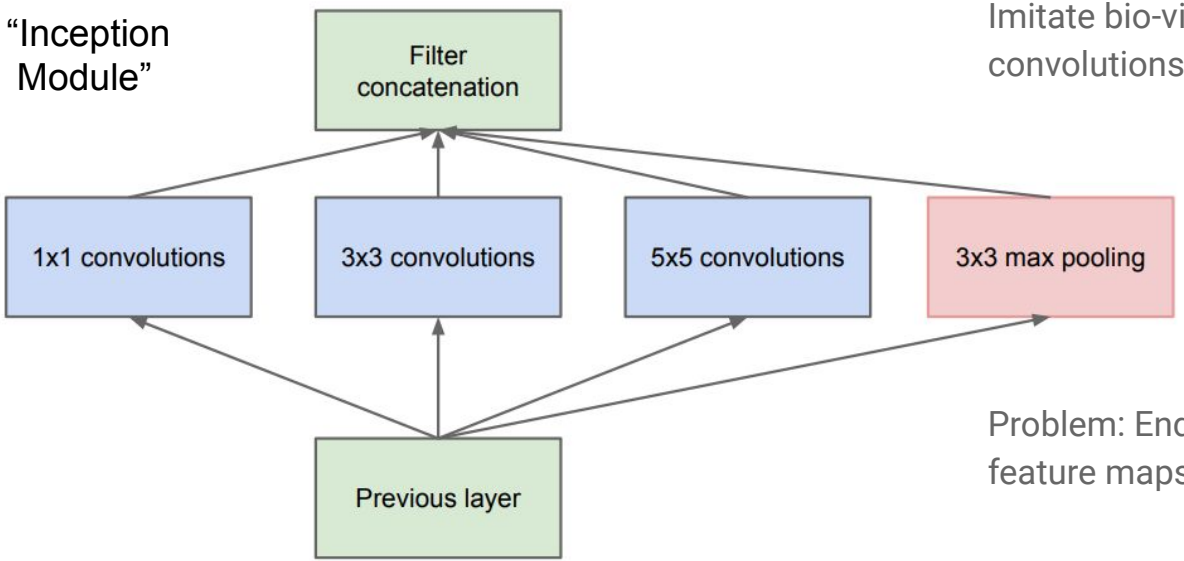
Famous Architectures



"GoogLeNet", Szegedy et al 2015. Has 22 layers. Won the ILSVRC 2014, achieved 6.67% top-5 accuracy.

Famous Architectures

"Inception Module"



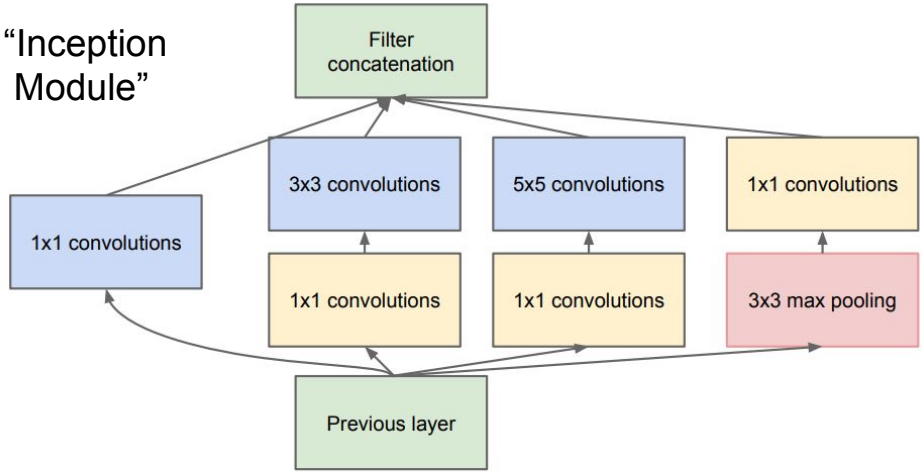
Imitate bio-vision by applying multiple sized convolutions to the same location.

Problem: Ends up with an enormous amount of feature maps.

"GoogLeNet", Szegedy et al 2015

Famous Architectures

“Inception Module”



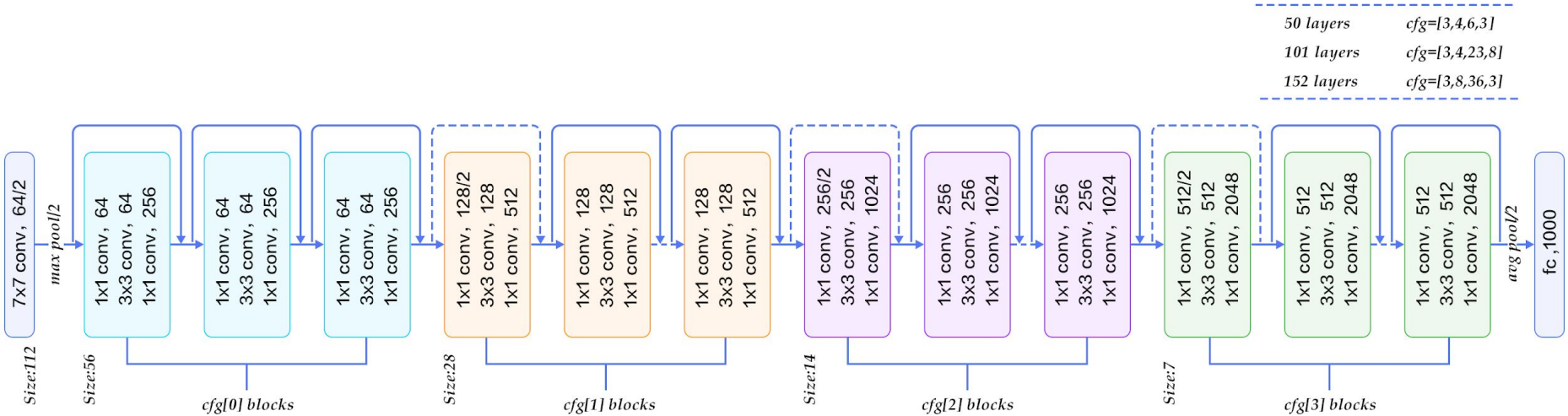
(b) Inception module with dimensionality reduction

Add 1x1 convolutions to reduce dimension before conv-ing more.

Remember 1x1 convolutions simply “squash” the stacked maps.

“GoogLeNet”, Szegedy et al 2015

Famous Architectures

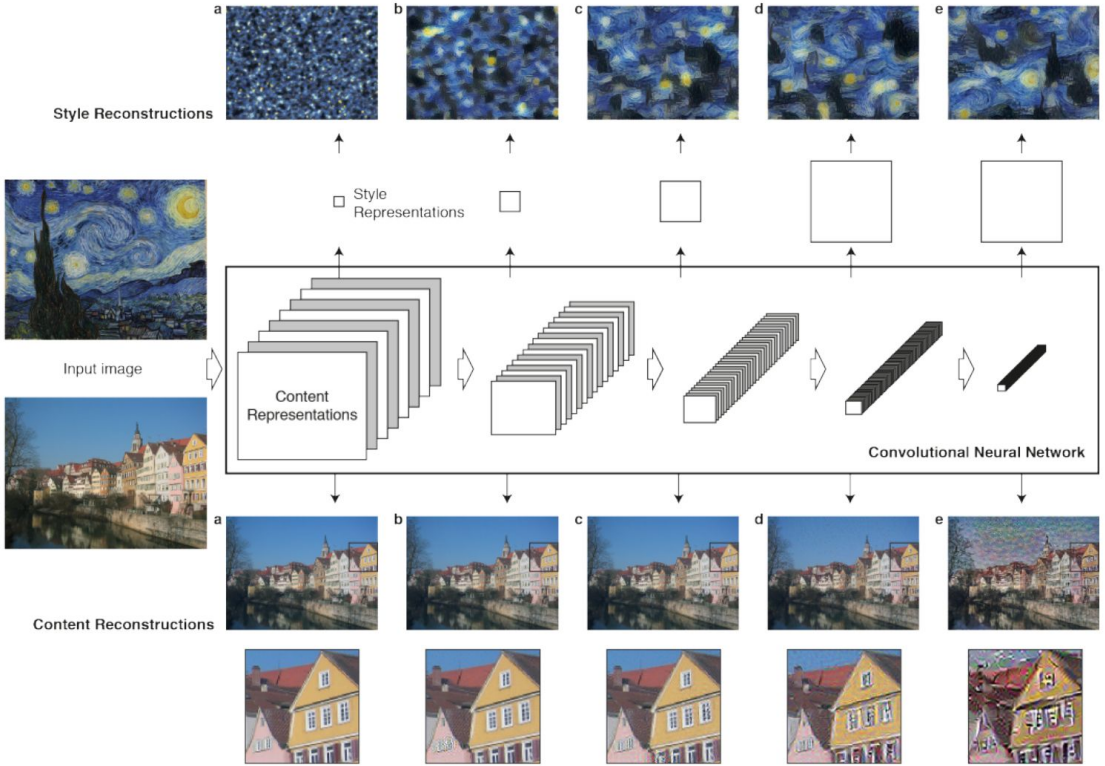


"ResNet" (Residual Network), He et al. 2016. Achieved **3.57%** accuracy on the ILSVRC 2015 with 152 layers.

Classification Results (CLS)

[ILSVRC 2017]

Style Transfer



ImageNet Model:
AlexNet
VGG
InceptionNet
ResNet
MobileNet
RetinaNet
DenseNet
...