Deep Visual Computing – A Primer

Daniel G. Aliaga

Fall 2023

Deep Visual Computing

- Since the beginning, it turns out **visual computing** and **machine learning** have been **deeply** connected
- Do you know why?
- Lets see… (get it: lets "see")

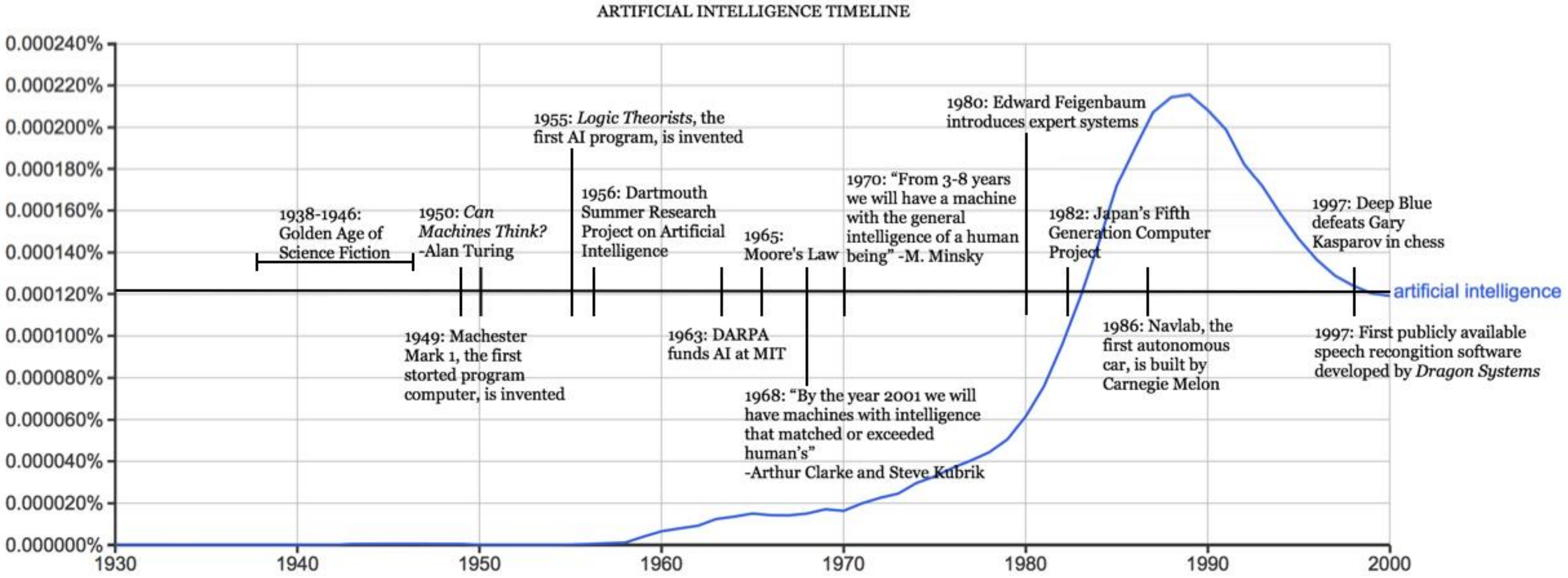
A long time ago in a computer far, far inferior to your phone, it all began…

-Daniel Aliaga, August 25, 2020

Logic Theorist (1956)

- A program designed to mimic the problem solving skills of a human
- From 1957-1974, AI flourished and failed and flourished…
- In 1968, A. Clarke and S. Kubrik said “by the year 2001 we will have machines with intelligence that matches or exceeded humans’s”
- In 1970, Marvin Minsky (MIT) said that in 3-8 years “we will have a machine with the general intelligence of an average human being”

AI Timeline



ARTIFICIAL INTELLIGENCE TIMELINE
0.000240%
0.000220%
0.000200%
0.000180%
0.000160%
0.000140%
0.000120%
0.000100%
0.000080%
0.000060%
0.000040%
0.000020%
0.000000%
1930
1940
1950
1960
1970
1980
1990
2000
1938-1946: Golden Age of Science Fiction
1950: *Can Machines Think?* -Alan Turing
1955: *Logic Theorists*, the first AI program, is invented
1956: Dartmouth Summer Research Project on Artificial Intelligence
1965: Moore's Law
1970: "From 3-8 years we will have a machine with the general intelligence of a human being" -M. Minsky
1980: Edward Feigenbaum introduces expert systems
1982: Japan's Fifth Generation Computer Project
1997: Deep Blue defeats Gary Kasparov in chess
artificial intelligence
1949: Machester Mark 1, the first stored program computer, is invented
1963: DARPA funds AI at MIT
1968: "By the year 2001 we will have machines with intelligence that matched or exceeded human's" -Arthur Clarke and Steve Kubrik
1986: Navlab, the first autonomous car, is built by Carnegie Melon
1997: First publicly available speech recognition software developed by *Dragon Systems*

1980s

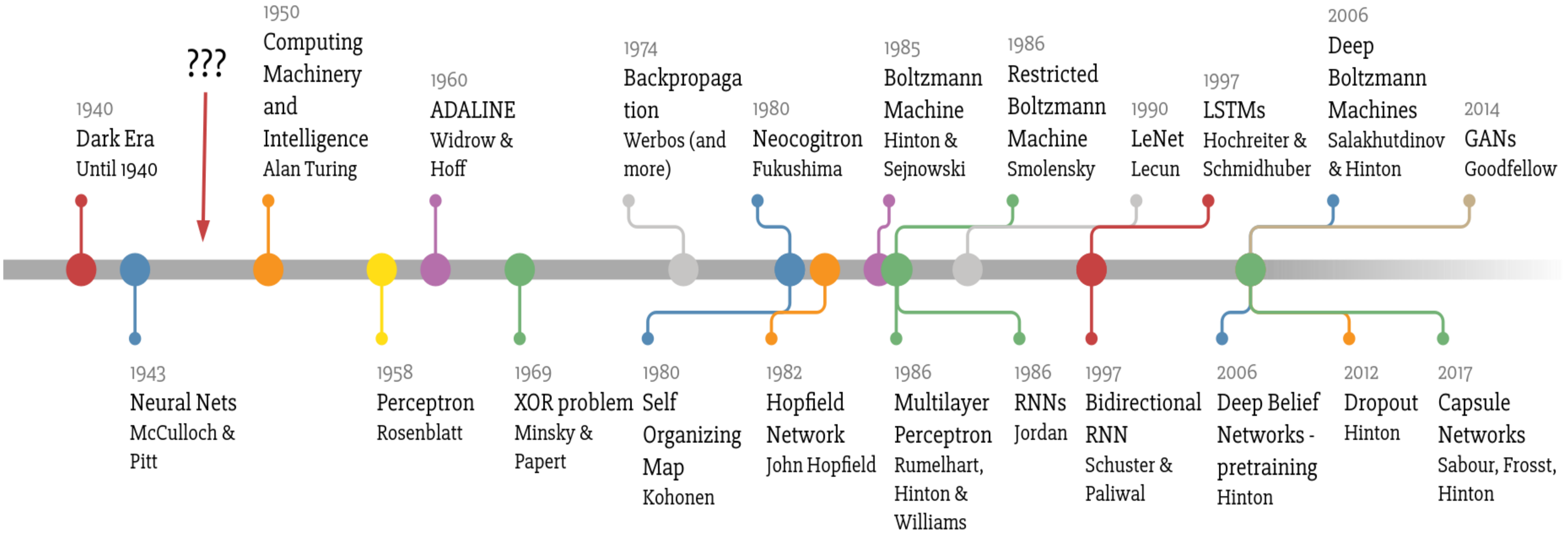
- Expert systems became popular: dedicated systems

- "Deep learning techniques" was a coined phrase but with diverse meanings…

- I was around then, and even a paid undergraduate researcher in a major AI lab

  - our job was to create a robot that could be programmed remotely and could execute algorithms for navigating and deciding how to avoid obstacles (e.g., walls and boxes)

Deep Learning Timeline



Made by Favio Vázquez

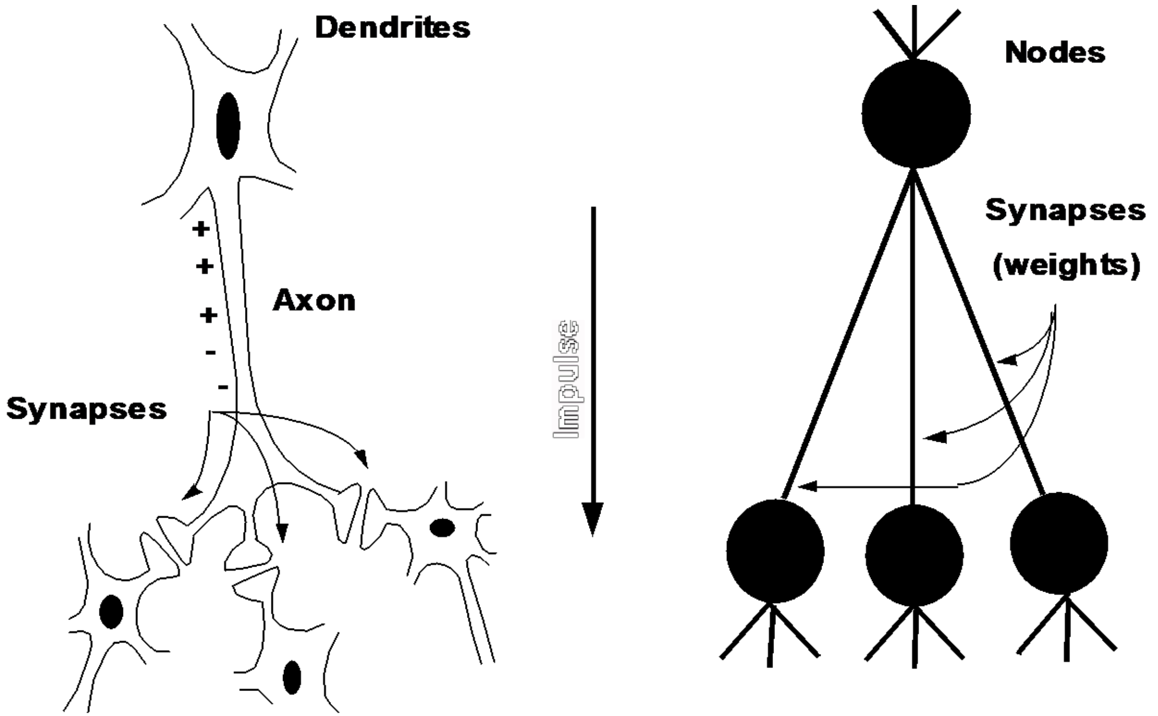
(Single Layer) Perceptron

- The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, F. Rosenblatt, Psychological Review, 65(6), 1958.

- Model based on the human visual system

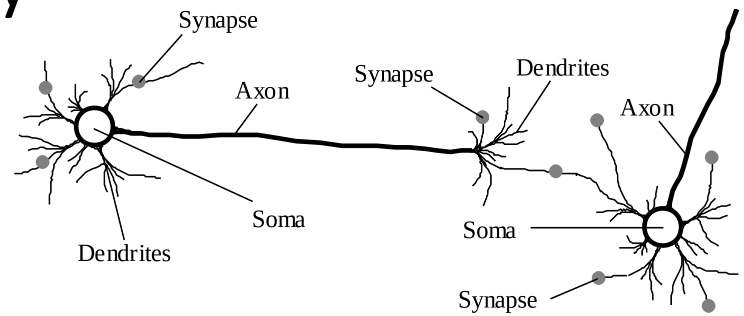
Biology 101

- In human brain:
  - Neuron switching time
    ~ 0.001 second
  - Number of neurons
    ~ 10^{10}
  - Connections per neuron
    ~ $10^{4\text{-}5}$
  - Scene recognition time
    ~ 0.1 second
  - Huge amount of parallel computation
    → 100 inference steps is not enough

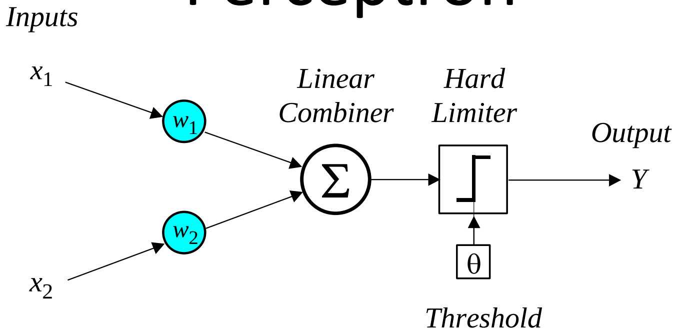


From Biology to Computers...

- Biology

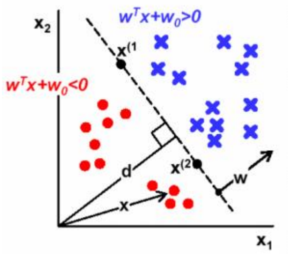


Perceptron



- Activation function

$$X = \sum_{i=1}^{n} x_i w_i \qquad Y = \begin{cases} +1, & \text{if } X \geq \omega_0 \\ -1, & \text{if } X < \omega_0 \end{cases}$$



Perceptron

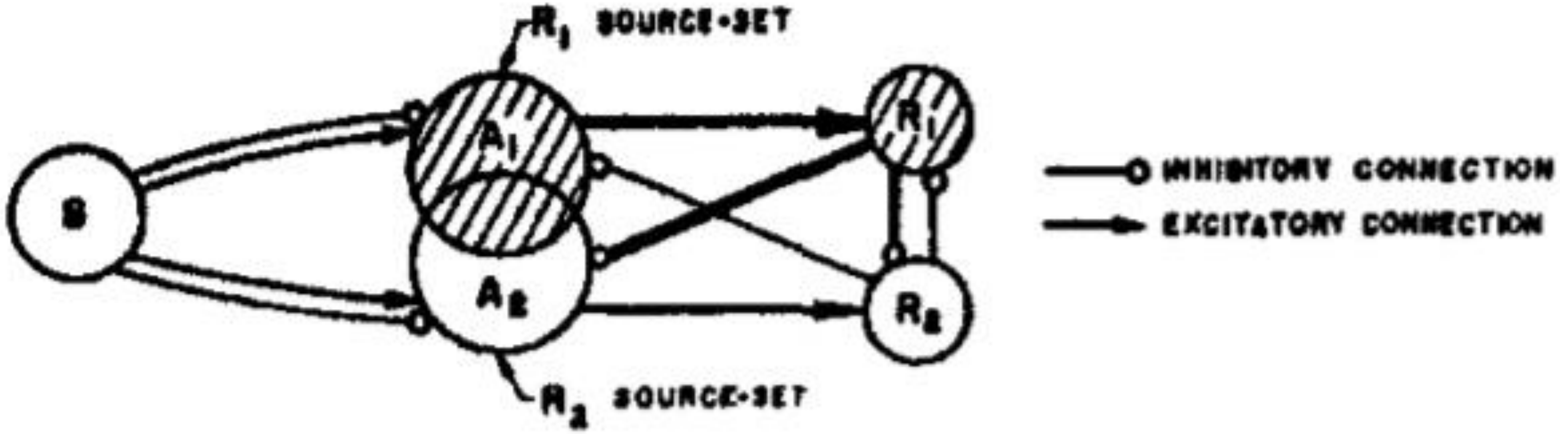
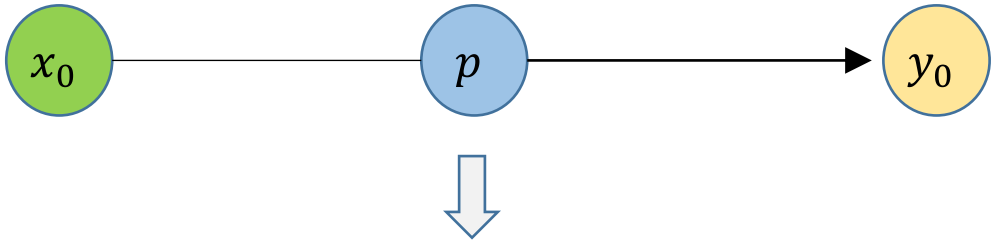


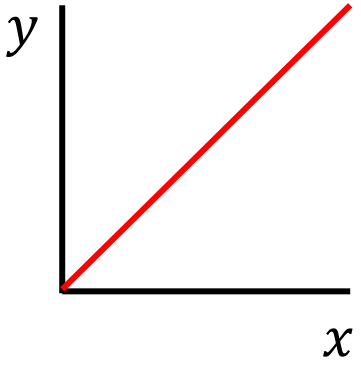
FIG. 2B. Venn diagram of the same perceptron (shading shows active sets for R_1 response).

Perceptron

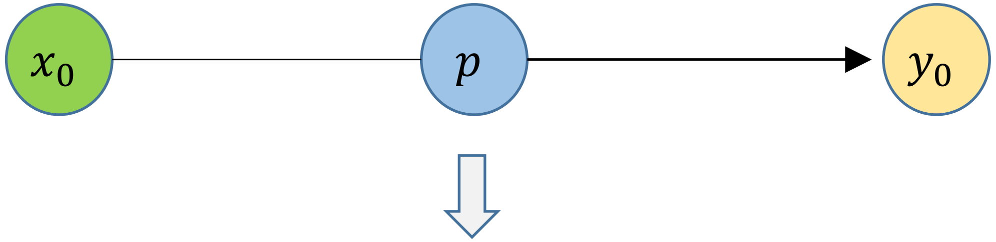


$$y = mx + b$$

Example: $b = 0, m = 1 \rightarrow\ y = x$



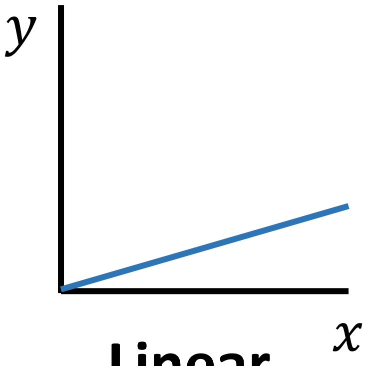
Perceptron



$$\boxed{y = mx + b}$$

Activation Function

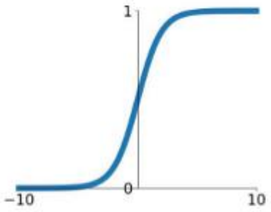
Activation Functions



Linear

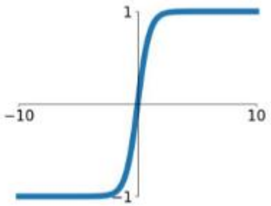
Sigmoid

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



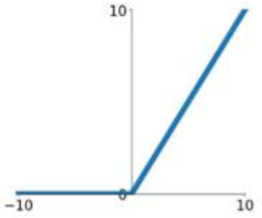
tanh

$\tanh(x)$



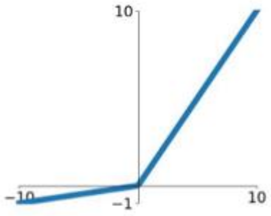
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

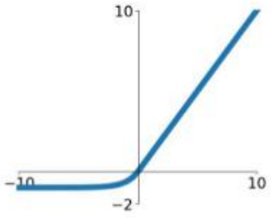


Maxout

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

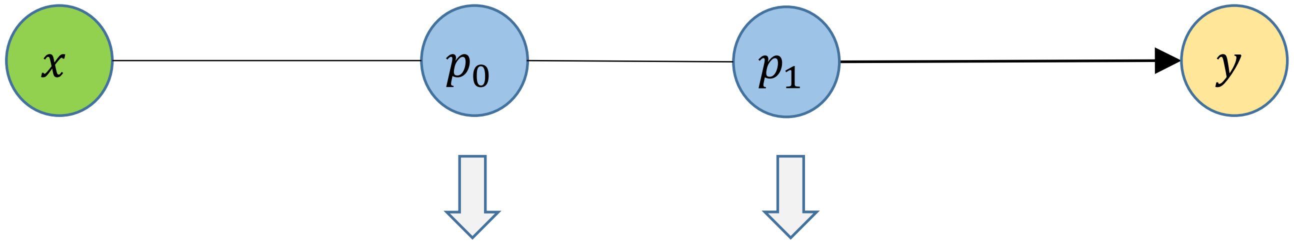
ELU

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



NOTE: ReLU = Rectified Linear Unit, ELU = Exponential Linear Unit

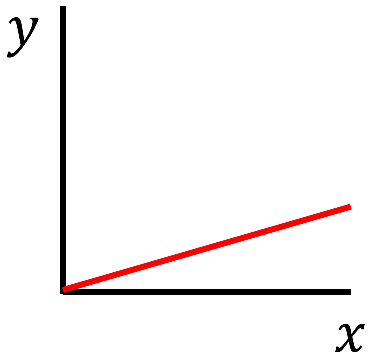
Multilayer Perceptron



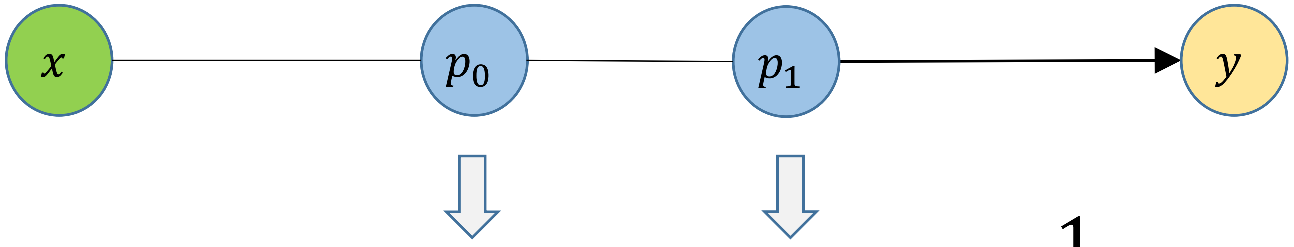
$$h = m_0 x + b_0 \qquad y = m_1 h + b_1$$

$$y = m_1(m_0 x + b_0) + b_1$$

Example: $\mathrm{b}_0 = \mathrm{b}_1 = 0, m_0 = m_1 = 0.5$ → $y = 0.25x$



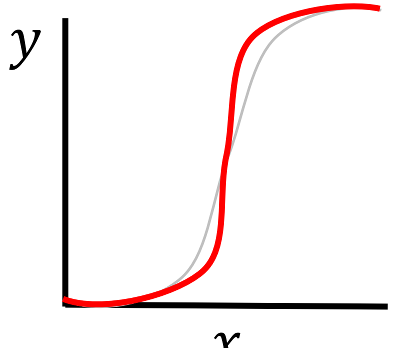
Multilayer Perceptron



$$h = m_0 x + b_0 \qquad y = \frac{1}{1 + e^{-m_1(h+b_1)}}$$

Example: $\mathrm{b}_0 = \mathrm{b}_1 = 0, m_0 = 2, m_1 = 1$

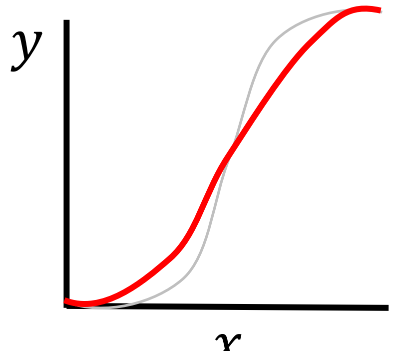
$$y = \frac{1}{1 + e^{-2x}}$$



Intuitively: y will be "high" for <u>smaller</u> values of x

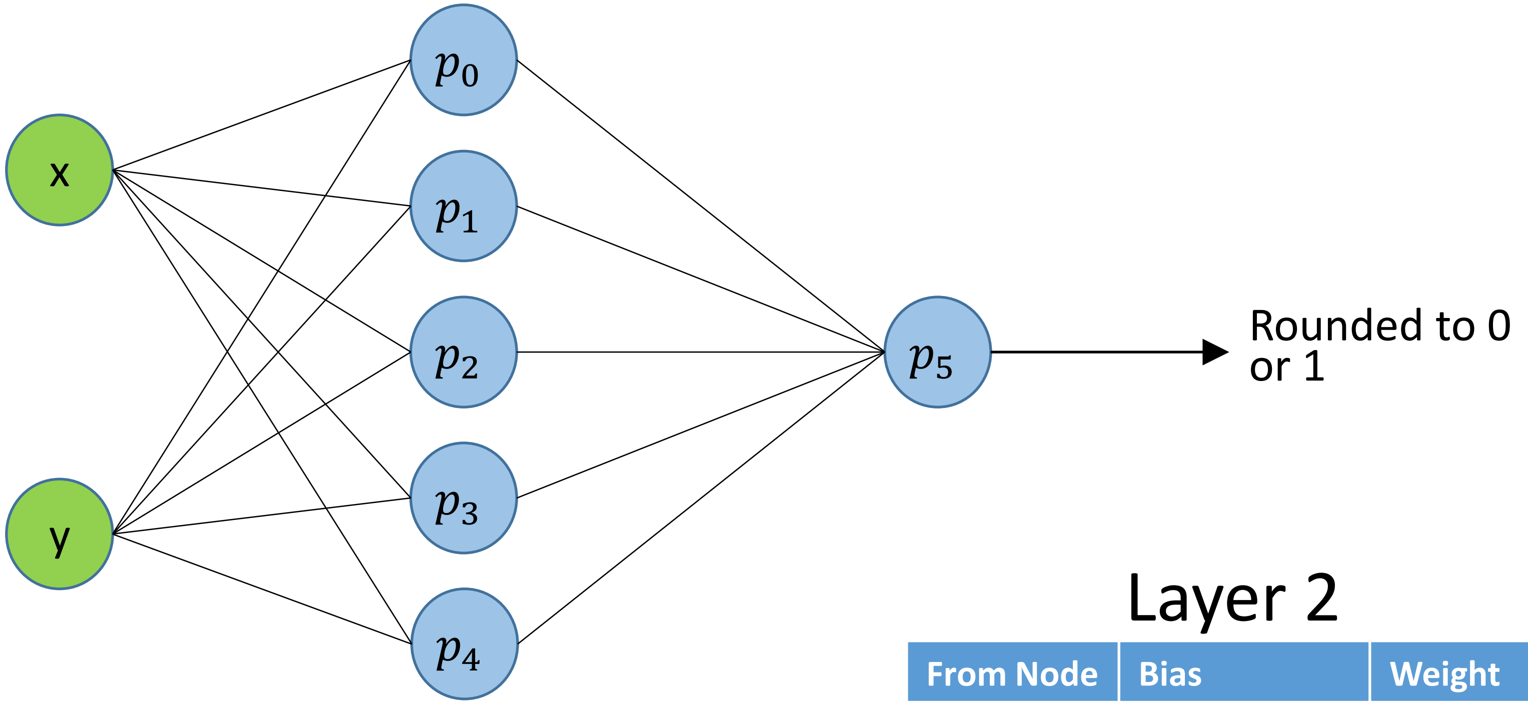
Example: $\mathrm{b}_0 = \mathrm{b}_1 = 0, m_0 = 0.5, m_1 = 1$

$$y = \frac{1}{1 + e^{-0.5x}}$$



Intuitively: y will be "high" for <u>larger</u> values of x

Multilayer Perceptron



(Sigmoid activation functions)

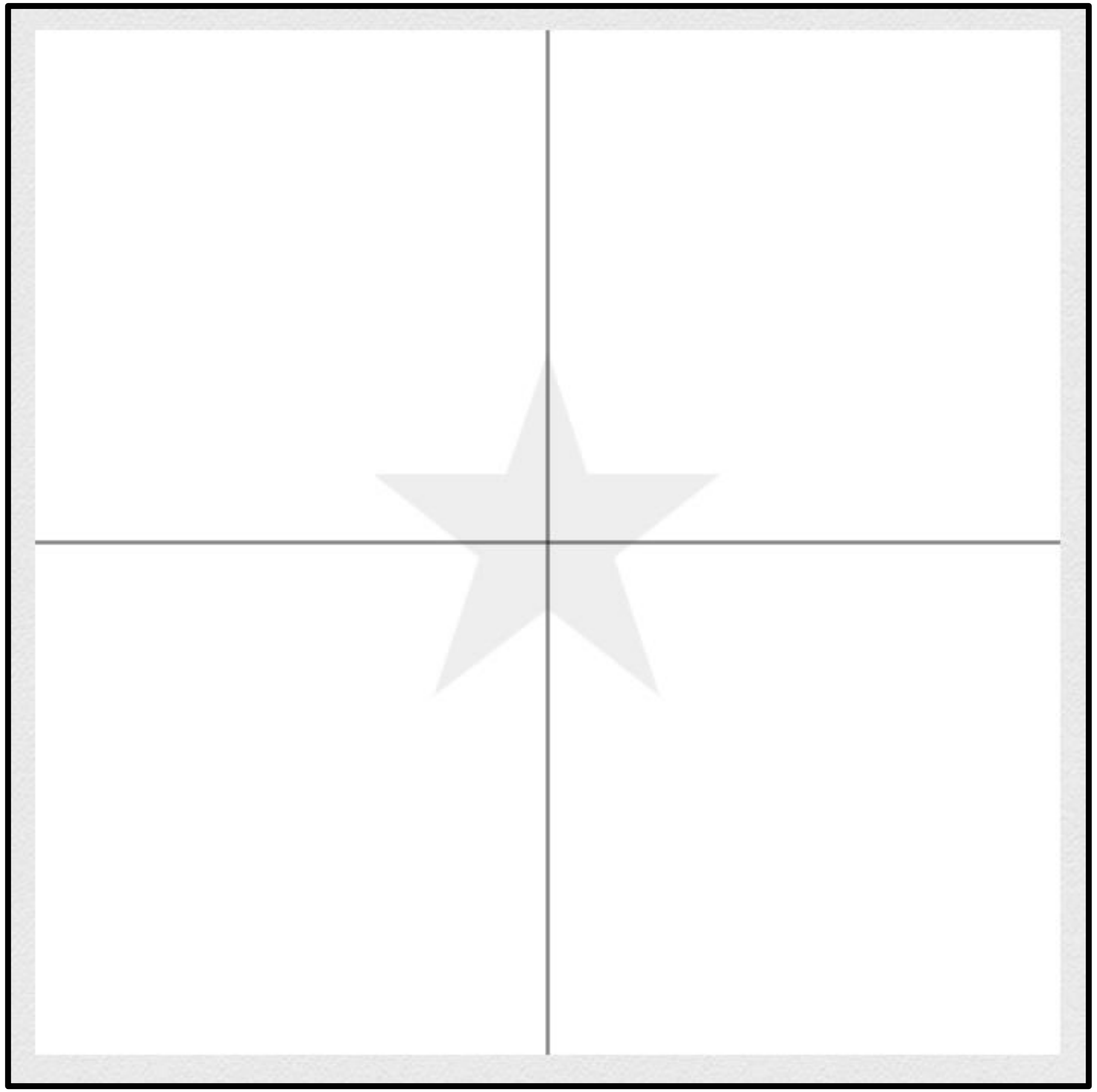
Layer 1

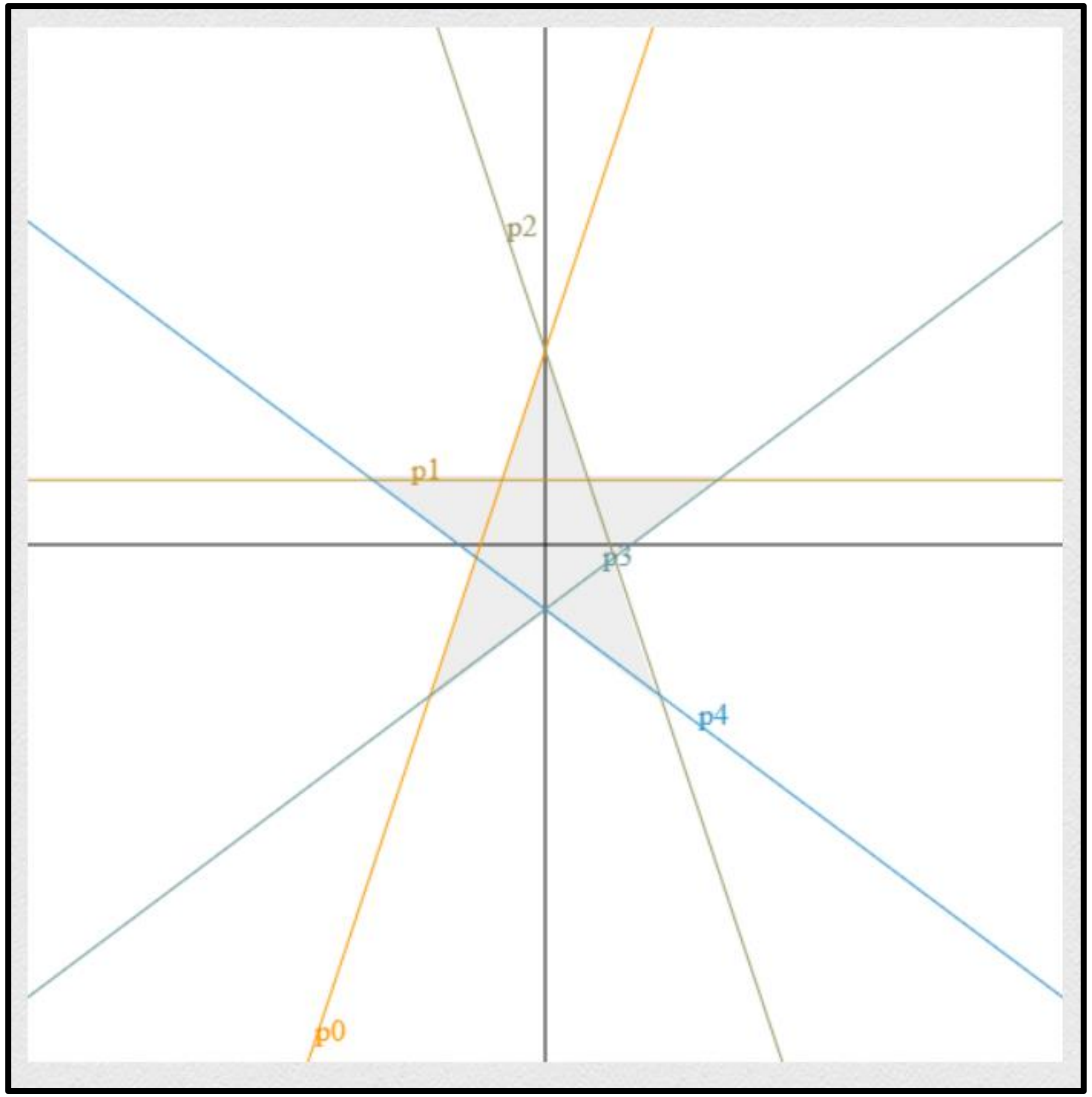
Node	Bias	x-Weight	y-Weight
0	-0.375	-3	1
1	-0.125	0	1
2	-0.375	3	1
3	0.125	-0.75	1
4	0.125	0.75	1

Layer 2

From Node	Bias	Weight
0	-0.2	1
1	-0.2	1
2	-0.2	1
3	-0.2	1
4	-0.2	1

Star Classifier: https://www.cs.utexas.edu/~teammco/misc/mlp





Perceptron

Algorithm 1: Perceptron Learning Algorithm

Input: Training examples $\{\mathbf{x}_i, y_i\}_{i=1}^{m}$.

Initialize $\mathbf{w}$ and b randomly.

while *not converged* **do**

  # # # Loop through the examples.
  for $j = 1, m$ **do**

    # # # Compare the true label and the prediction.
    $error = y_j$ - $\sigma(\mathbf{w}^T \mathbf{x}_j + b)$

    ### If the model wrongly predicts the class, we update the weights and bias.
    if *error != 0* **then**

      ### Update the weights.
      $\mathbf{w} = \mathbf{w} + error \times x_j$

      ### Update the bias.
      $b = b + error$

  Test for convergence

Output: Set of weights $\mathbf{w}$ and bias b for the perceptron.

Perceptrons

- Book by M. Minsky and S. Papert (1969)
- Was actually “An Introduction to Computational Geometry” – thus visual as well
- Commented on the limited ability of perceptrons and on the difficulty in training multi-layer perceptrons
- (Back propagation appeared in 1986 and helped a lot!)

Reprise: Computer Vision

- In 1959, Russell Kirsch and colleagues developed an image scanner: transform an image into a grid of numbers so that a machine can understand it!

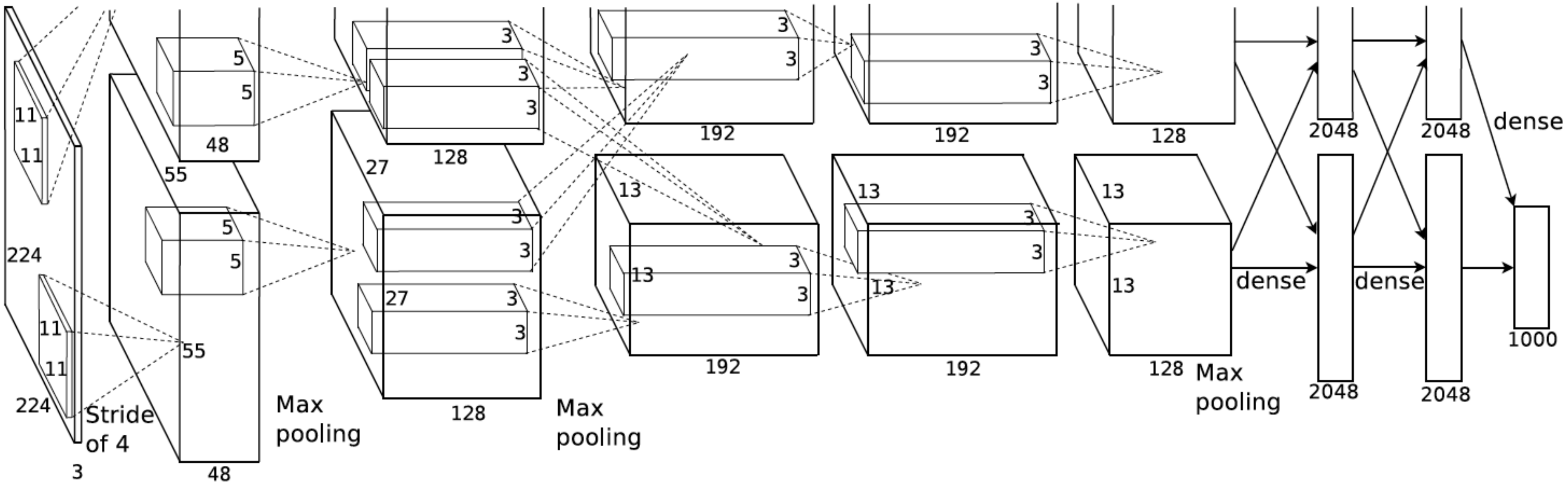
- One of the first scanned images:
  (176x176 pixels)

2010

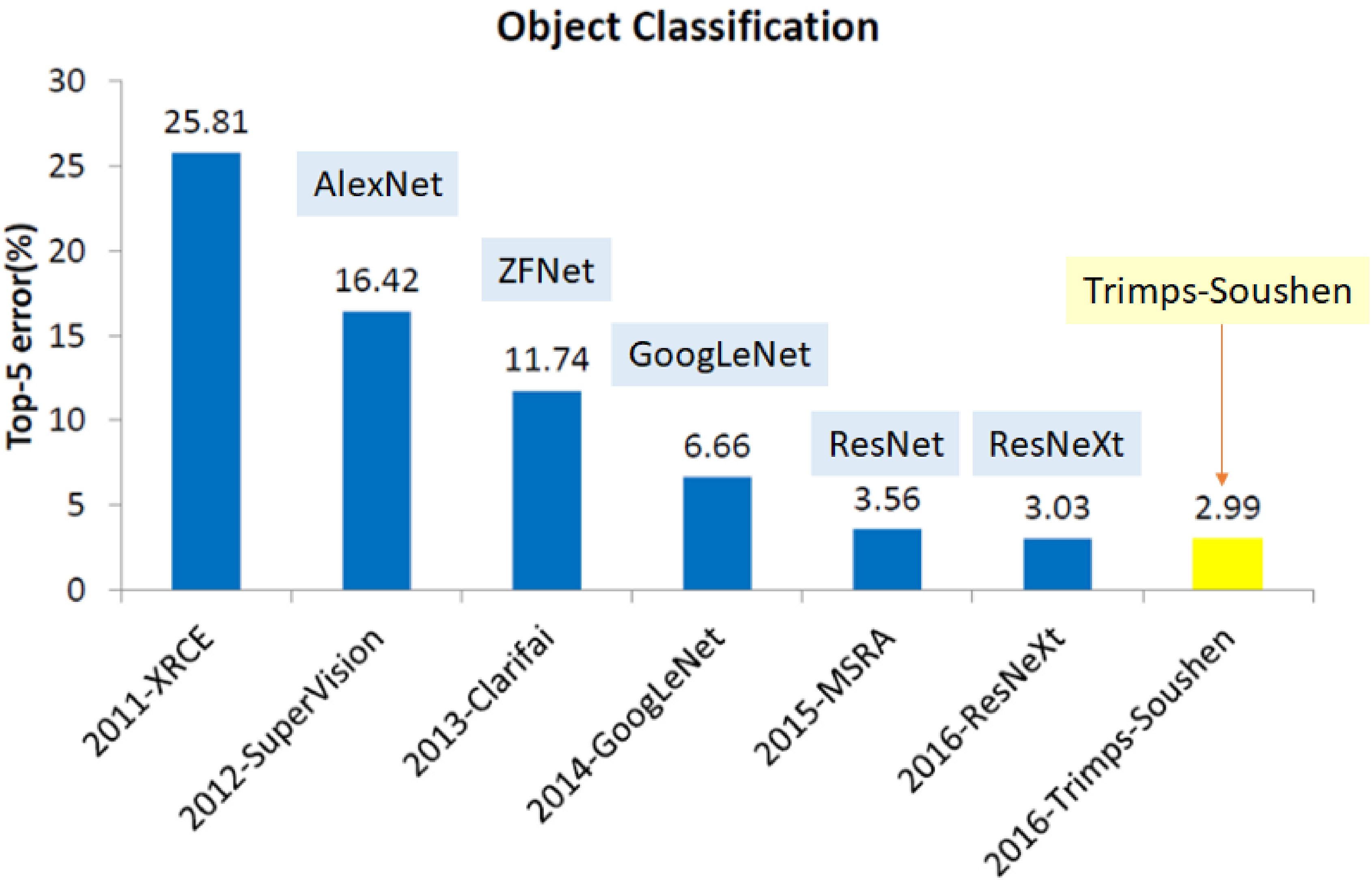
- ImageNet Large Scale Visual Recognition Competition (ILSVRC) runs annually
  - 2010/2011: error rates were around 26%
  - 2012: the beginning of a new beginning – AlexNet – reduced errors to 16%!

AlexNet

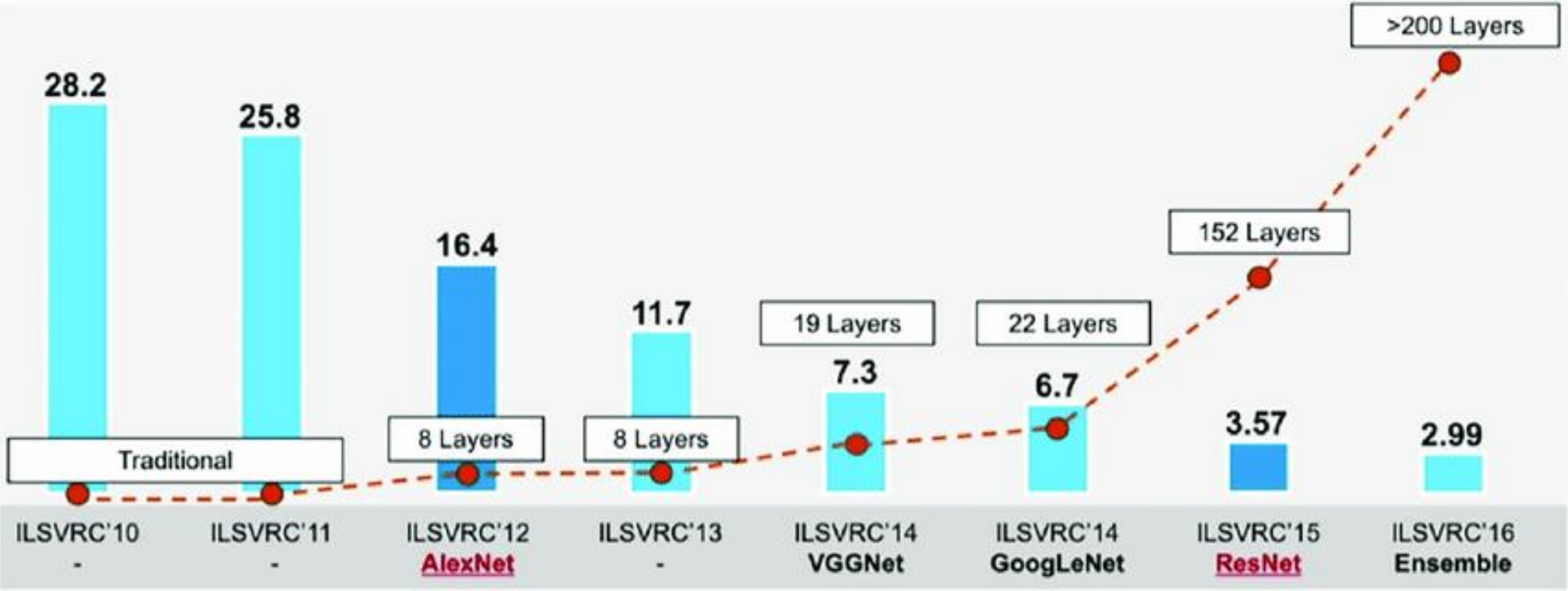
- University of Toronto created a CNN model (AlexNet) that changed everything (Krizhevsky et al. 2012)



ILSVRC (2011-2017)



ILSVRC (2010-2017)



Reprise: Computer Graphics

- First graphics **visual** image:
  - Ben Laposky used an oscilloscope in 1950s

(note: one of my undergrad senior projects was an oscilloscope based graphics engine)

Whirlwind Computer @ MIT

- Video display of real-time data:

1960s

- Ivan Sutherland used vector displays (=oscilloscope), light pens, and interaction

1965: The Ultimate Display…

- Fred Brooks using one of Ivan's displays….the birth of VR/AR

- NOTE: Fred Brooks was on my PhD committee, I worked in his research group and my MS and PhD revolved around VR/AR and graphics.

Deep Learning in Computer Graphics

- Like in computer vision, since 2010'ish **deep learning** has revolutionized computational **imaging** and computational **photography, rendering, and more**
- However, hand-crafted methods have significantly improved other domains such as **geometry processing, rendering and animation, video processing, and physical simulations**

Basic Machine Learning Recipe

1. Obtain training data
2. Choose decision and loss functions
3. Define goal
4. Optimize!

1. Training Data

$\{x_i, y_i\}$ for $i \in [1, N]$

Fundamental categories:

1. Synthetic data
2. Real data (annotated)
3. Real data (unannotated) <- tricky!

Properties:

1. Data should span/populate the distribution of expected input values
2. Data should be plenty – kinda same as above
3. Data should have low errors/noise (ideally)

2. Decision and Loss Functions

$$\hat{y} = \mathrm{f}_{\theta}(\mathrm{x}_{\mathrm{i}})$$

The function you wish to “decide” that given the inputs, and the parameters θ, yields an output $\hat{y}$ that is equal or close to desired values; thus, you seek

$$l(\hat{y}, y_i) \to 0$$

Properties:

1. Decision should be “doable” so that convergence is possible
2. Loss function should exploit as much as possible of domain knowledge

3. Define (Training) Goal

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$$

Define a function to find parameters θ^* that minimize the loss function for the entire training data set; i.e., find network weights and biases that make the network “learn” the desired (high-dimensional) function

4. Optimize!

- Perform small steps (opposite the gradient)…

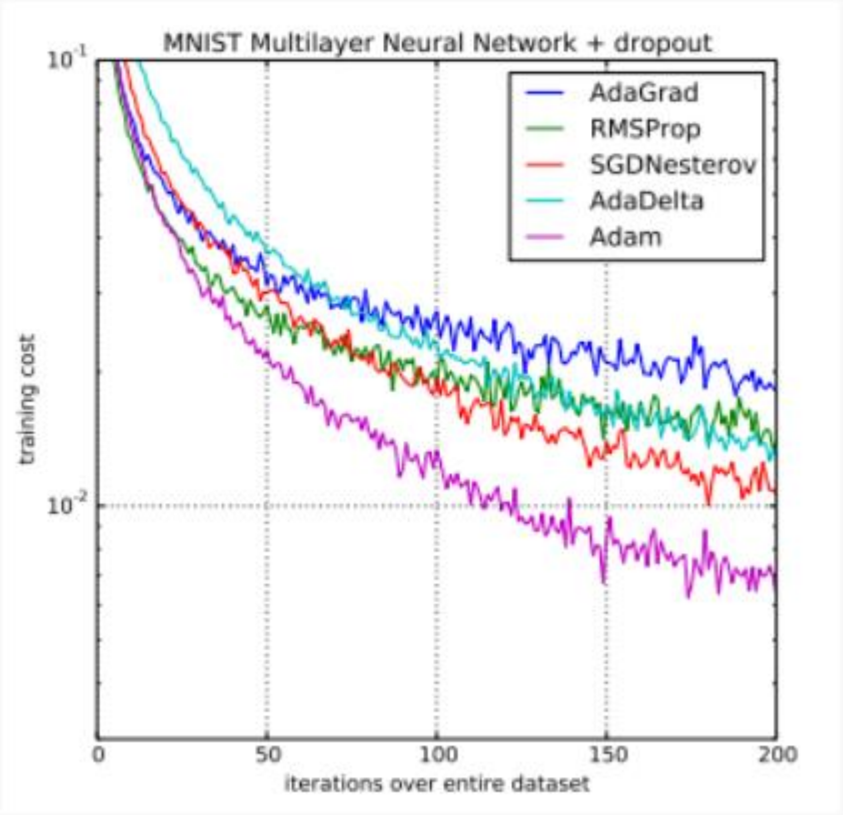
$$\theta^{t+1} = \theta^{t} - \alpha_t \nabla l(\mathrm{f}_\theta(x_i), y_i)$$

Move a small step against the gradient to eventually reach a set of network parameters that minimize the loss function

4. Optimize!

- Methods:
  - Stochastic Gradient Descent (SGD),
  - Adam, or
  - Others

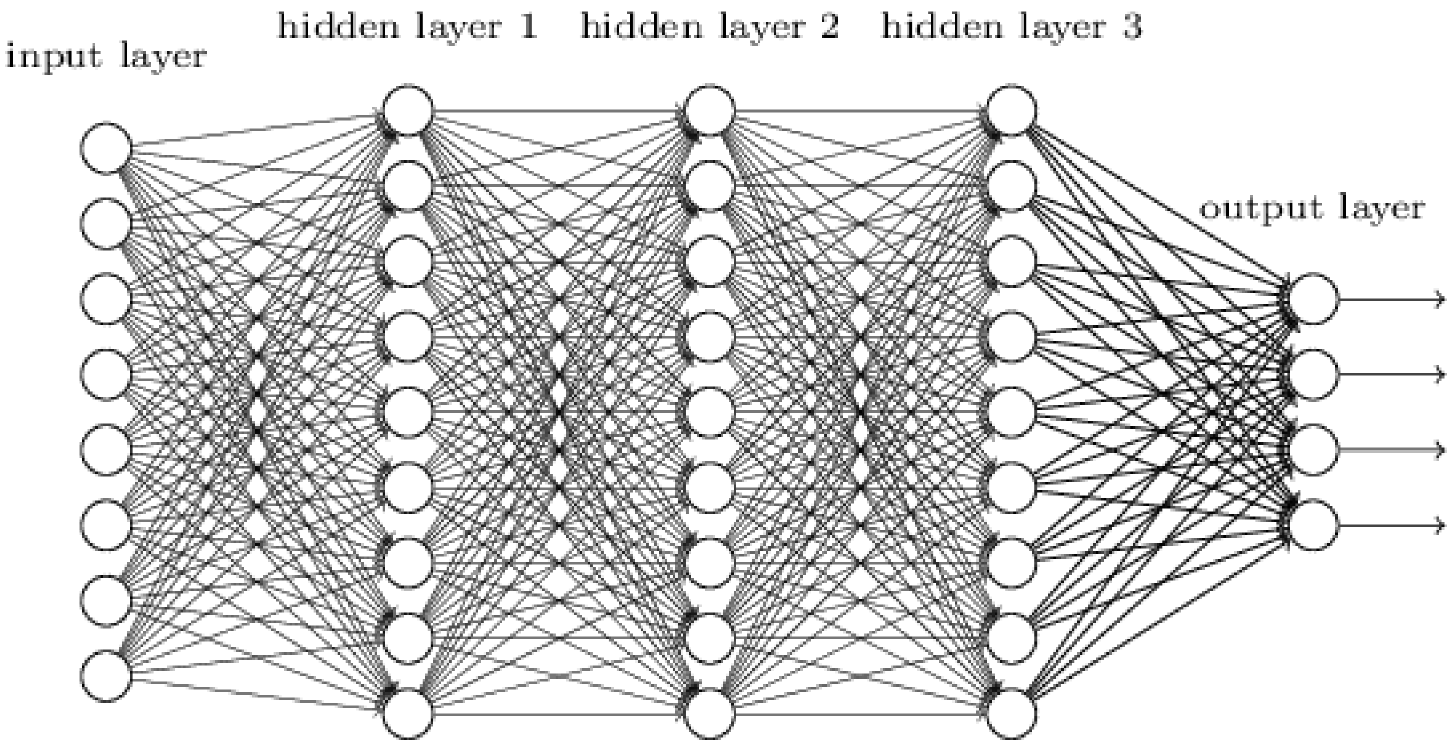
- Adam: an adaptive moment estimation based optimization – the learning rate changes during the optimization [Kingma and Ba, 2015]

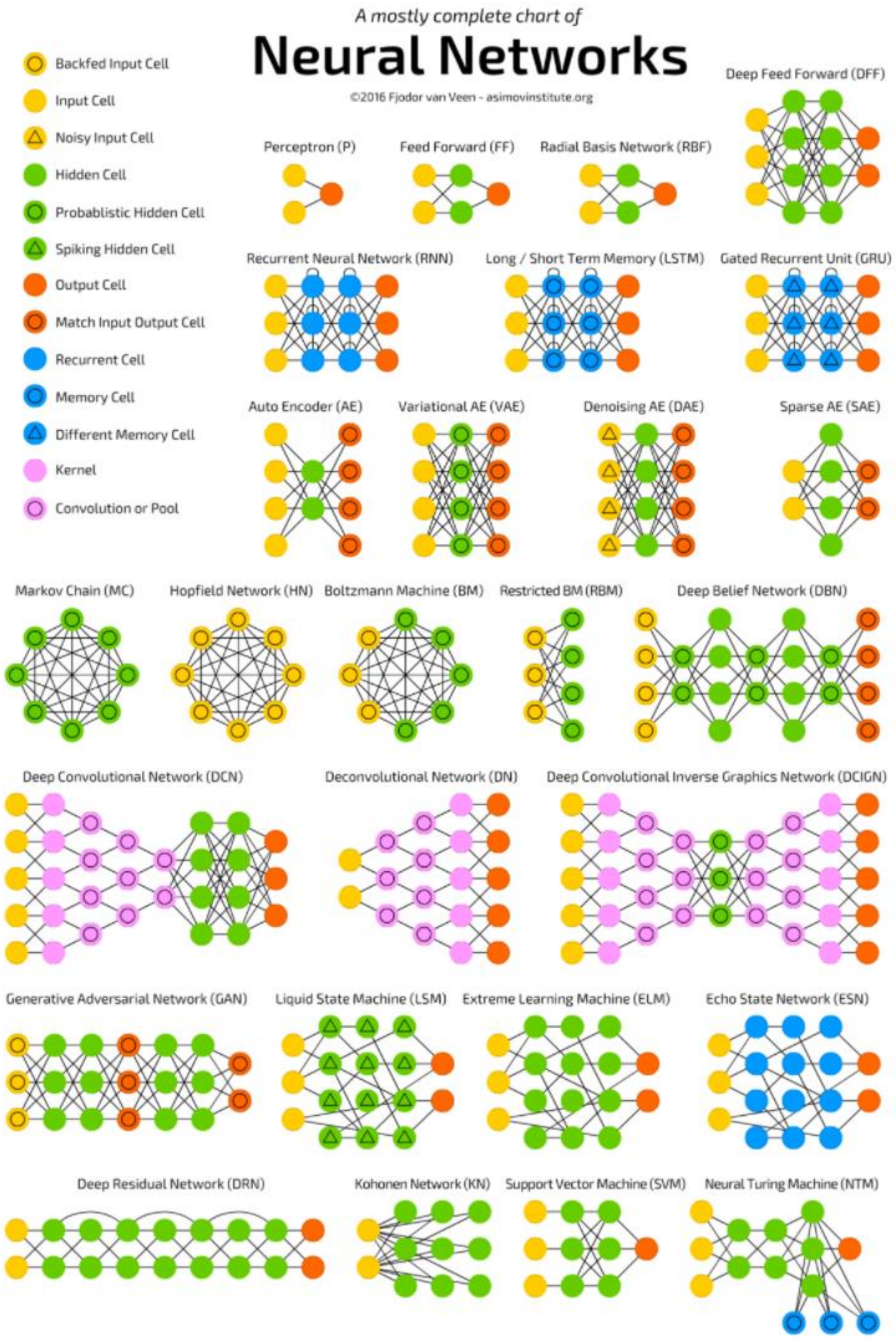


Comparison of Adam to Other Optimization Algorithms Training a Multilayer Perceptron
Taken from Adam: A Method for Stochastic Optimization, 2015.

Multilayer Perceptron: Fully Connected

- Fully Connected (FC) Network has lots of weights and biases to learn
  - 1 MP image has $Lx10^{12}$ parameters for L layers (or several billion parameters)





https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

Can we reduce the number of parameters to learn with our training data?

- Yes! Convolutional Neural Networks (CNN)
- Uses:
  - Spatial locality
  - Kernel reuse
  - Weight sharing
- Example result:
  - Instead of “billions of parameters”, using 100 kernels of 10x10 pixels with weight sharing needs only **10,000 parameters**

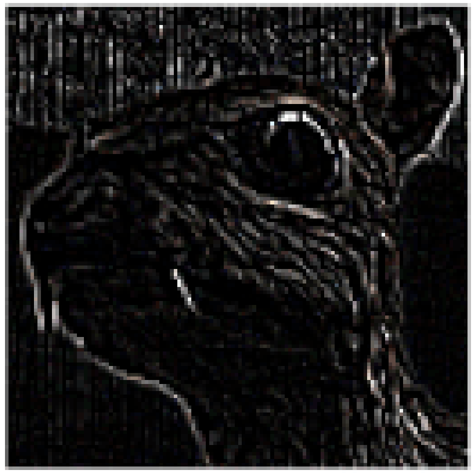
(Image) Convolution

Input image

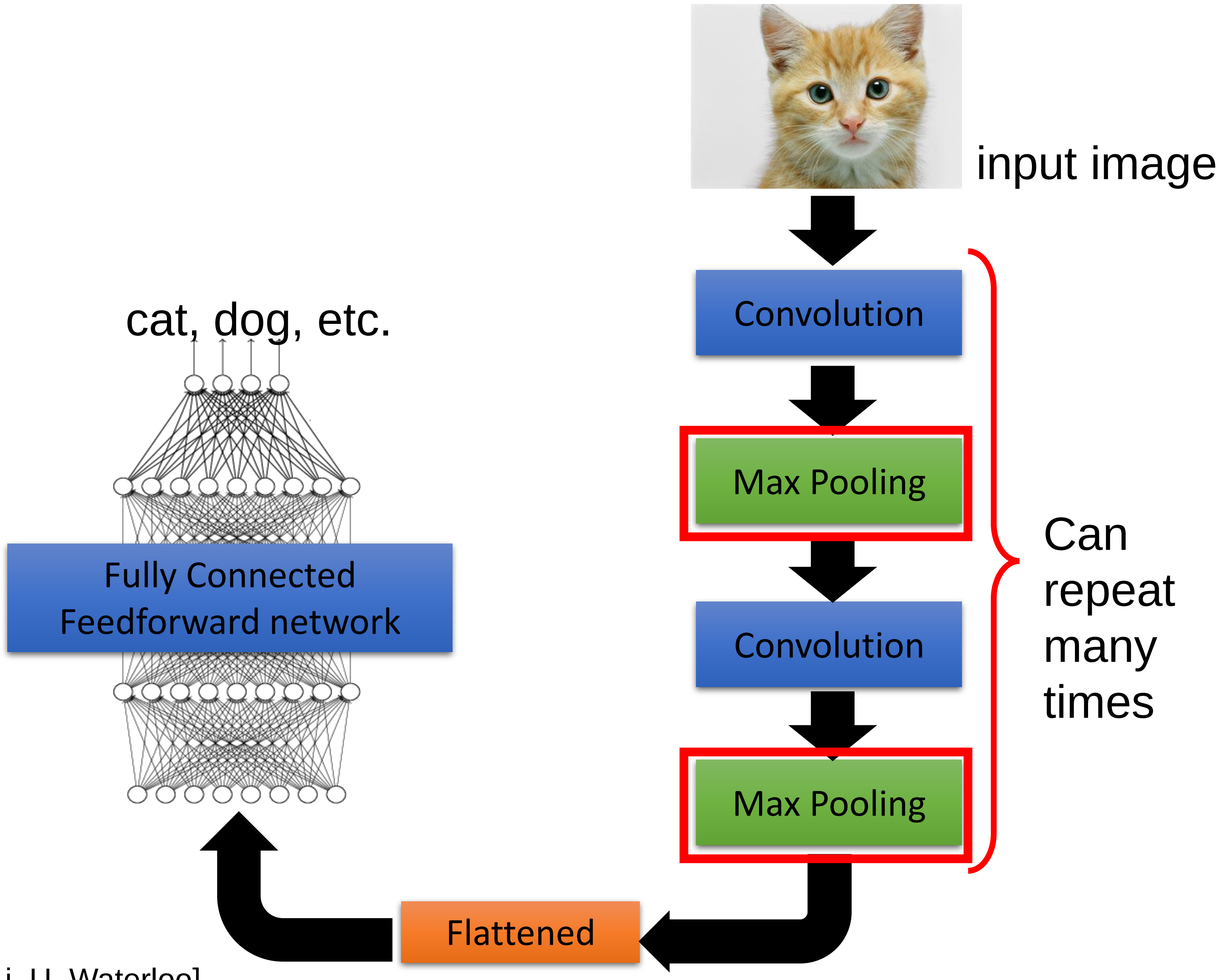
Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map



CNN



[Slides based on Ming Li, U. Waterloo]

CNN: Convolution Layer

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

[Slides based on Ming Li, U. Waterloo]

CNN: Convolution Layer

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

[Slides based on Ming Li, U. Waterloo]

CNN: Convolution Layer

-1	1	-1
-1	1	-1
-1	1	-1

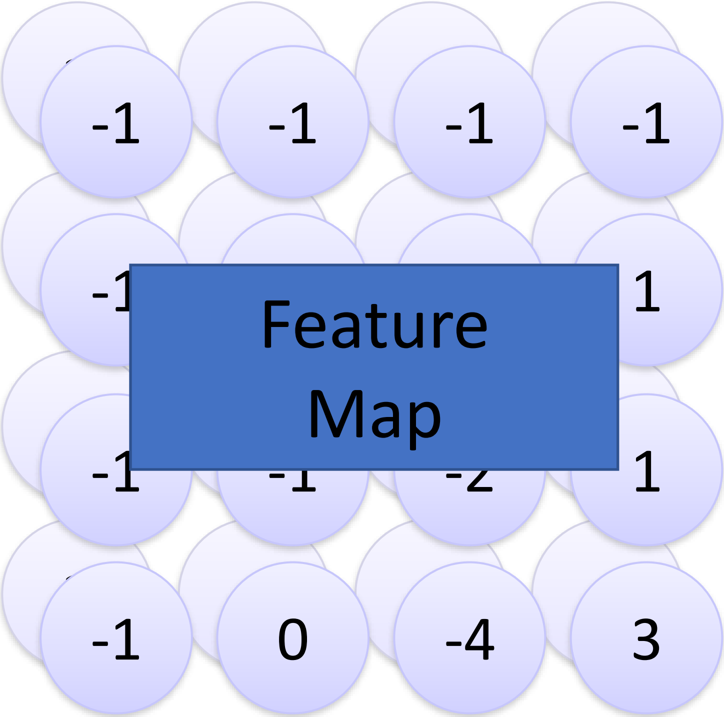
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Repeat this for some number of filters



Two 4 x 4 images
Forming 2 x 4 x 4 matrix

[Slides based on Ming Li, U. Waterloo]

CNN: Max Pooling

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1

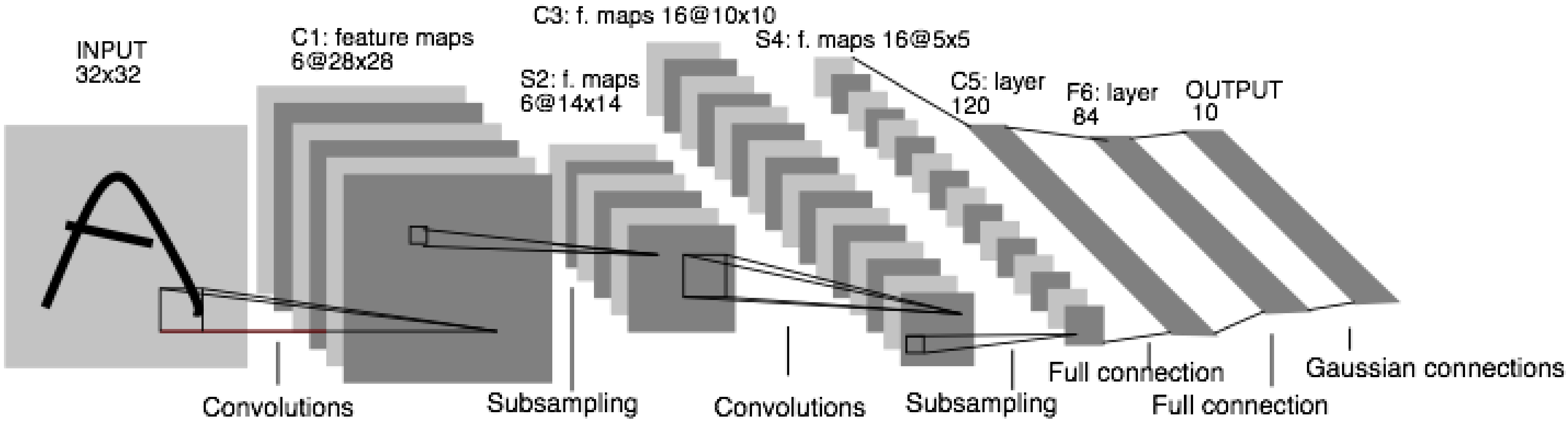
3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

[Slides based on Ming Li, U. Waterloo]

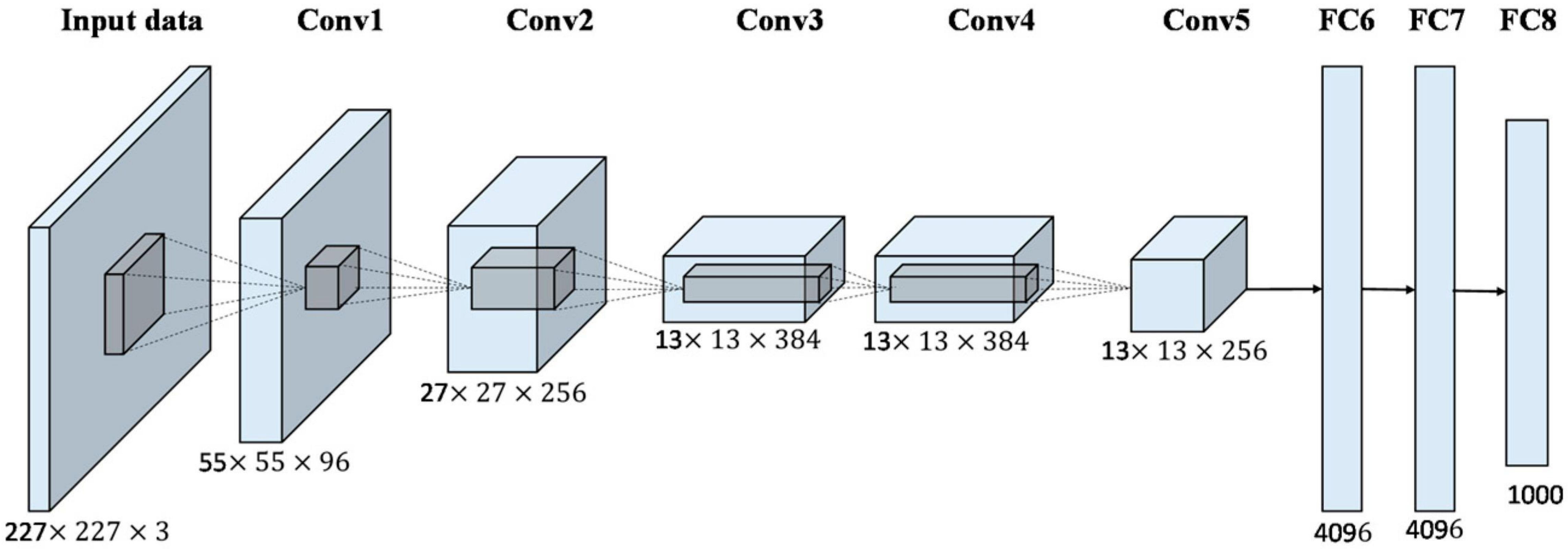
LeNet (1998)

- 32x32 image using CPU



LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [LeNet]

AlexNet (2012) -- diagrammatic



AlexNet: First Convolution Layer

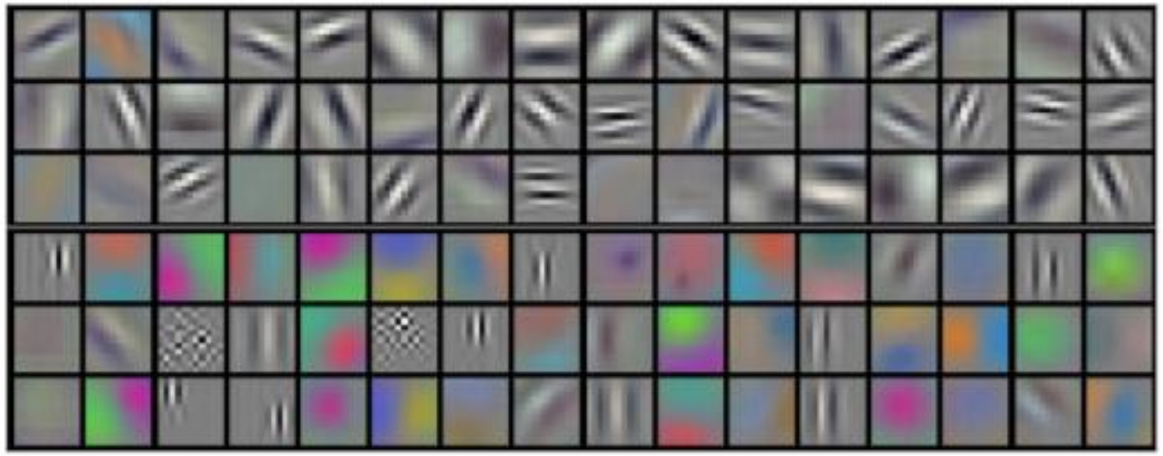


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

Comparison

LeNet

- 32*32*1
- 7 layers
- 2 conv and 4 classification
- **60 thousand parameters**
- Only two complete convolutional layers
  - Conv, nonlinearities, and pooling as one complete layer

AlexNet

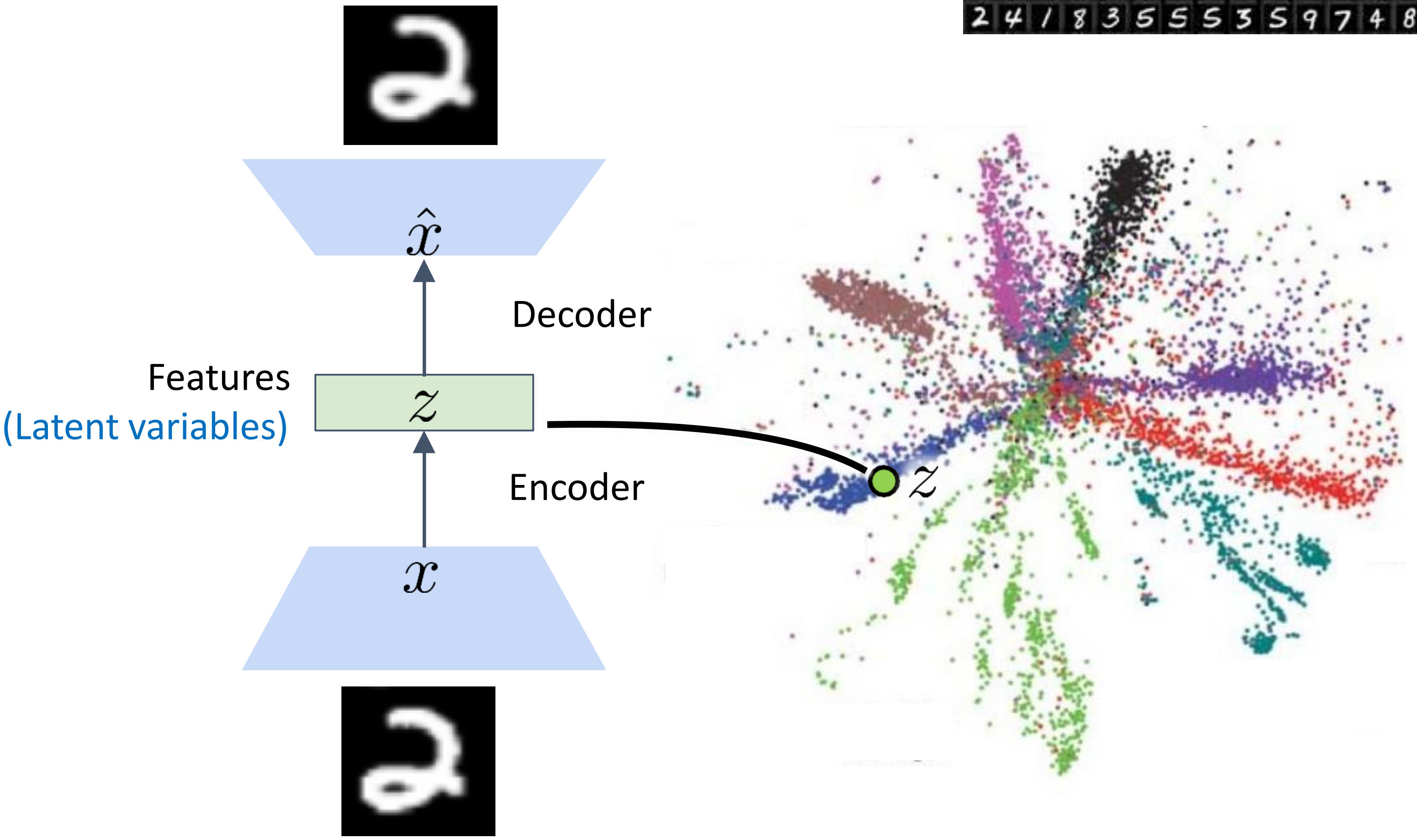
- 224*224*3
- 8 layers
- 5 conv and 3 fully classification
- 5 convolutional layers, and 3,4,5 stacked on top of each other
- Three complete conv layers
- **60 million parameters**
- **Since** insufficient data, did data augmentation:
  - Patches (224 from 256 input), translations, reflections
  - PCA, simulate changes in intensity and colors

CNN Demo

- https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

Autoencoder (AE)

Learn a data distribution for MNIST:

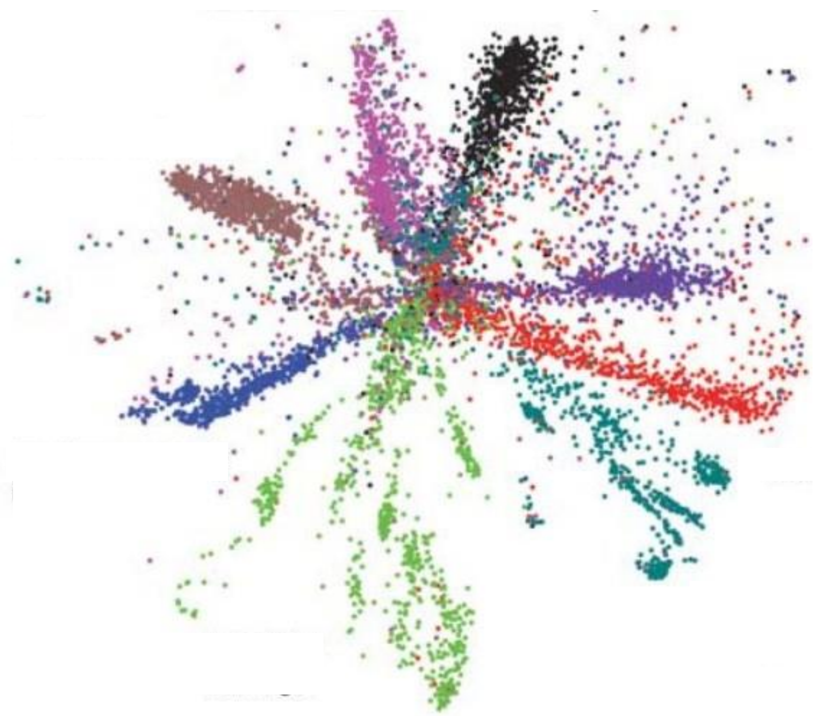


[CreativeAI – SIGGRAPH Course]

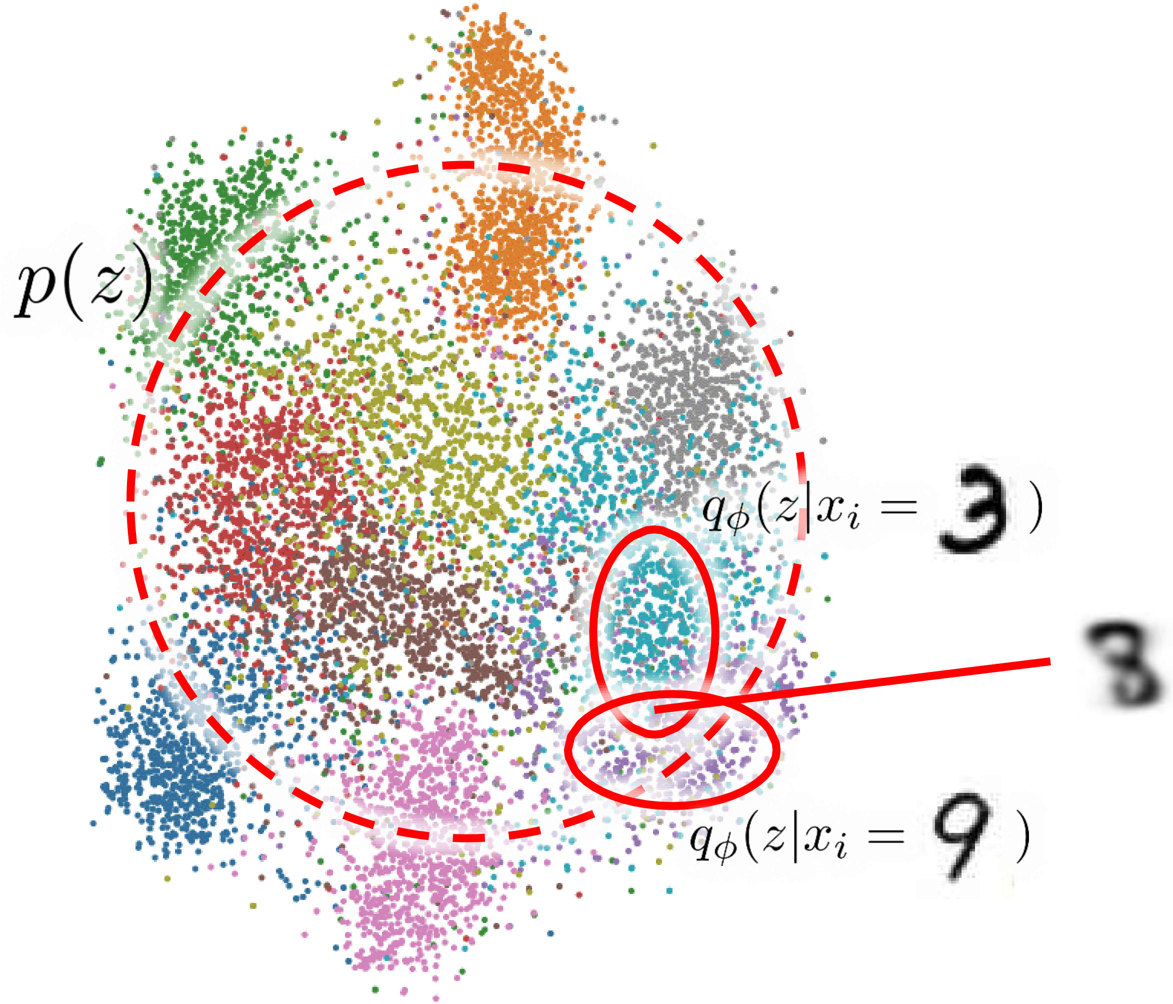
Image Credit: *Reducing the Dimensionality of Data with Neural Networks*, Hinton and Salakhutdinov

AE Improvement: Variational Autoencoder

Autoencoder



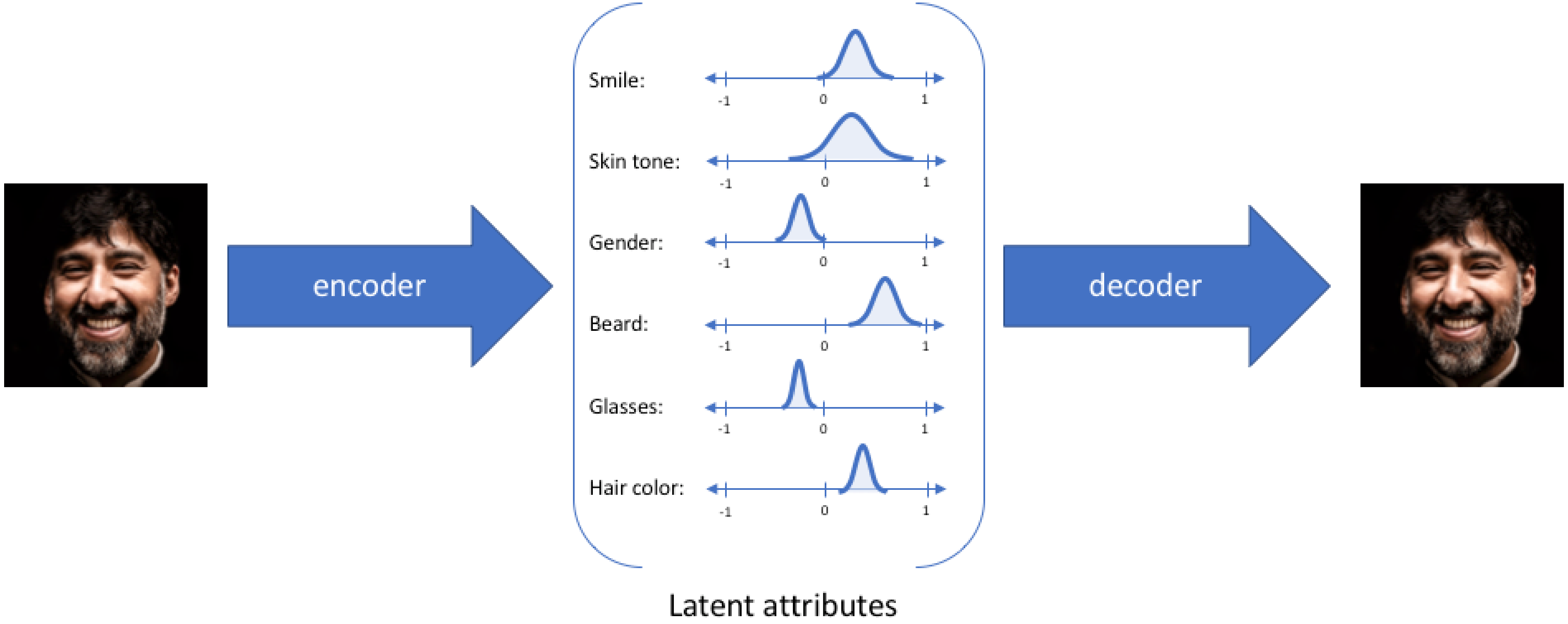
VAE



[CreativeAI – SIGGRAPH Course]

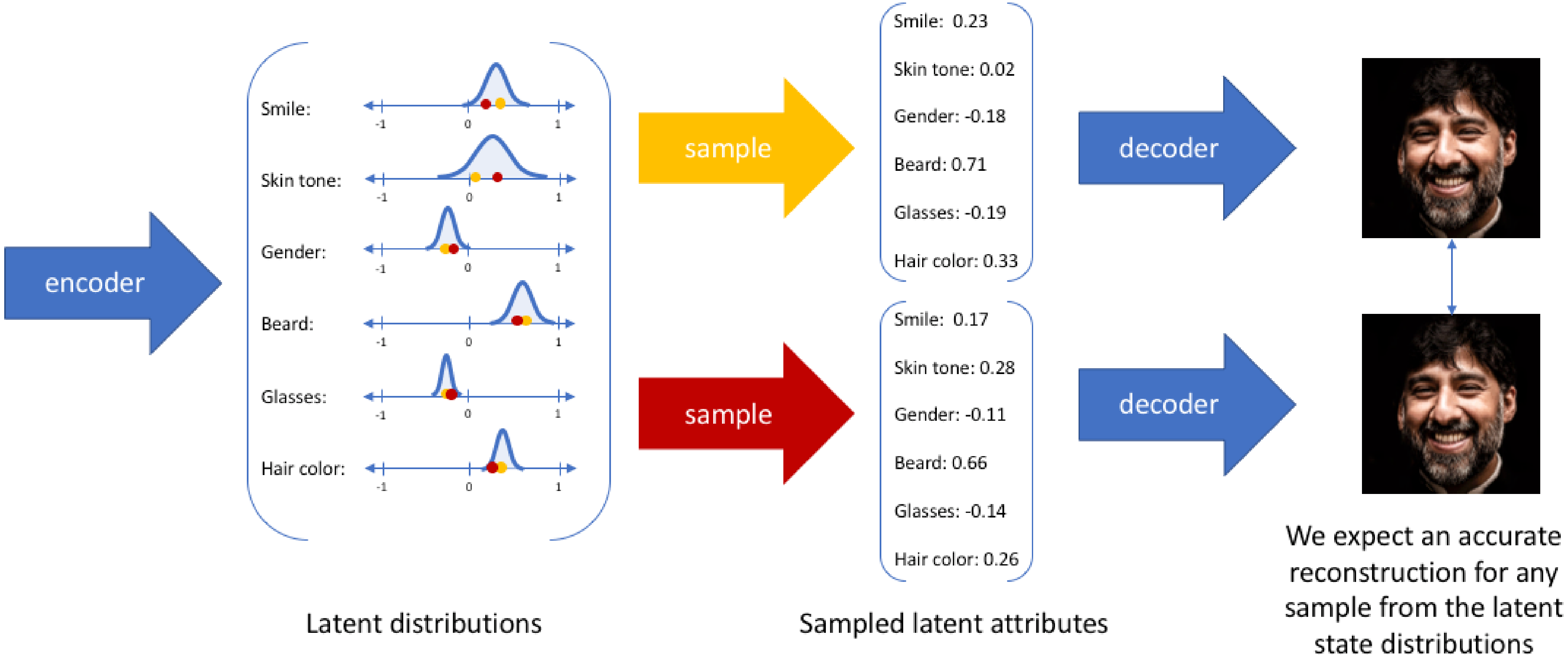
Image Credit: Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov

Variational Autoencoder Latent Space



[CreativeAI – SIGGRAPH Course]
https://www.jeremyjordan.me/variational-autoencoders/

Now, can ask for samples!



[CreativeAI – SIGGRAPH Course]

https://www.jeremyjordan.me/variational-autoencoders/

Generative Adversarial Networks (GANs)

[Goodfellow et al. 2014]

z → **Player 1: generator**

Scores if discriminator can't distinguish output from real image

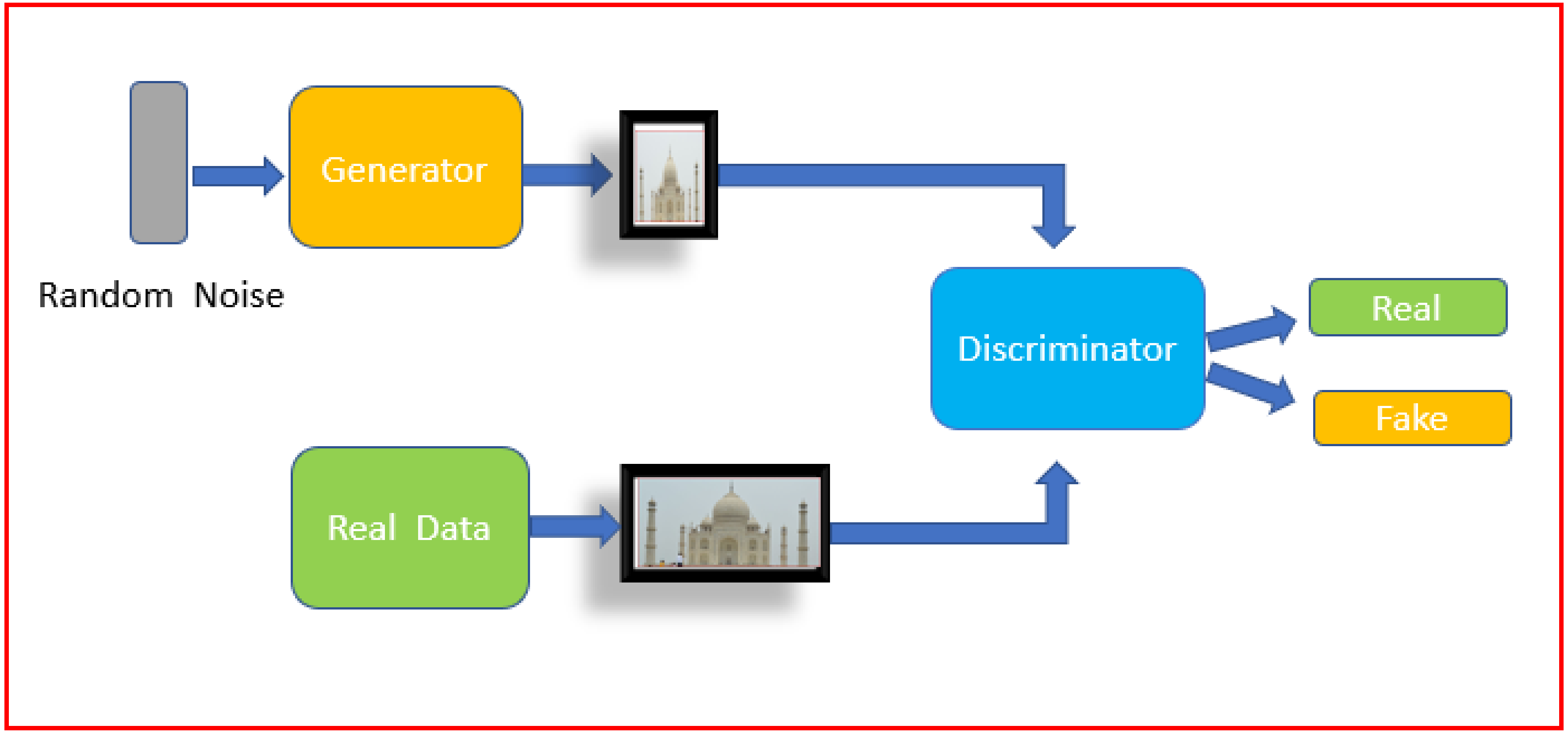
from dataset

Player 2: discriminator → real/fake

Scores if it can distinguish between real and fake

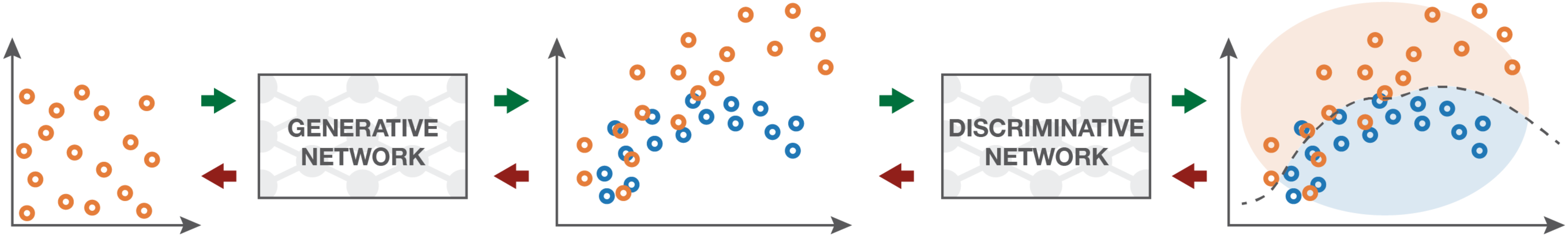
Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks*, Karras et al.

Generative Adversarial Networks (GANs)



GAN Information Flow

Forward propagation (generation and classification)   Backward propagation (adversarial training)



Input random variables.	The generative network is trained to **maximise** the final classification error.	The generated distribution and the true distribution are not compared directly.	The discriminative network is trained to **minimise** the final classification error.	The classification error is the basis metric for the training of both networks.

[CreativeAI – SIGGRAPH Course]

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Example of the Progression in the Capabilities of GANs From 2014 to 2017. Taken from [The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation](), 2018.

https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

StyleGAN

Additional Tricks:

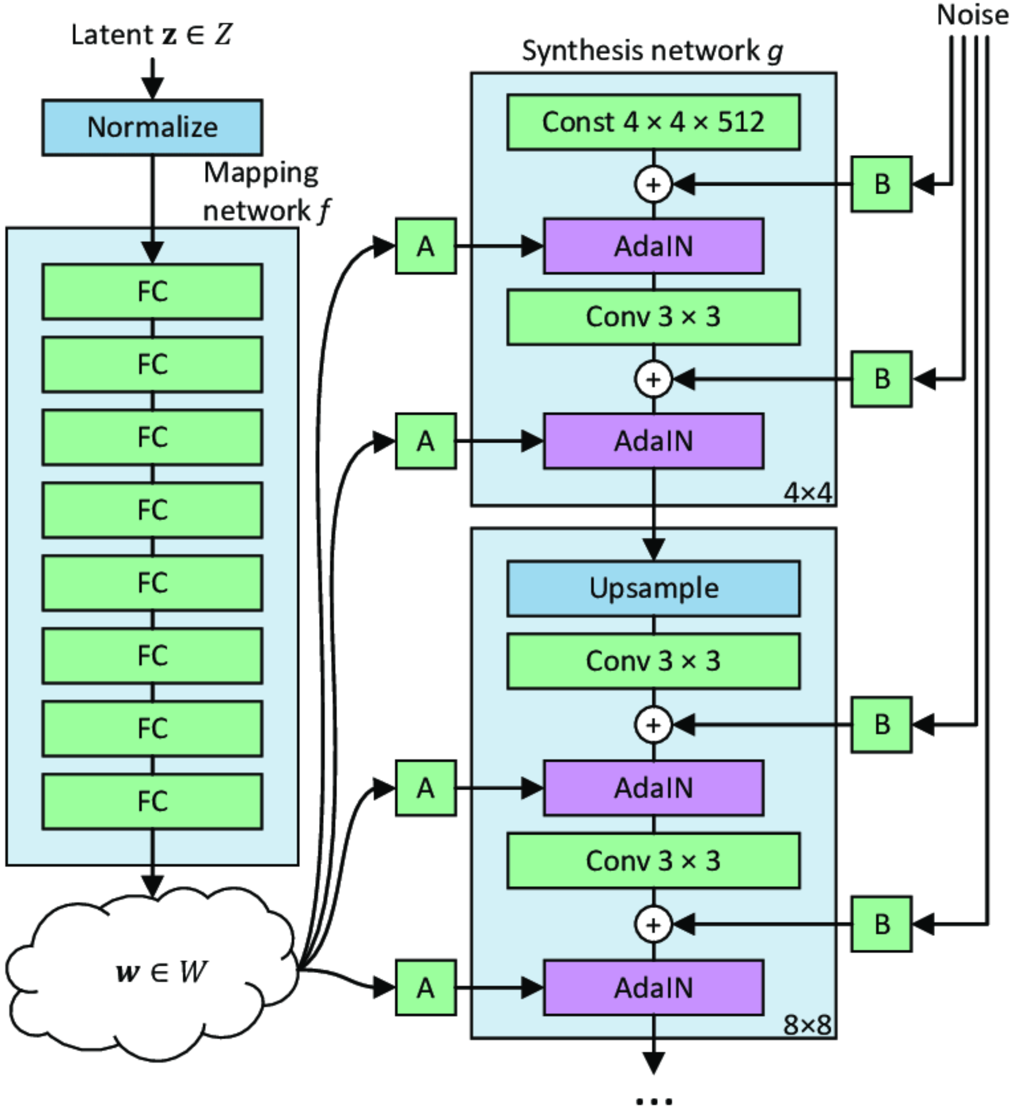
- Coarse-to-fine training
- Transformation of $p(z)$ to a more complex distr.
- …

content

style

Image credit: *A Style-Based Generator Architecture for Generative Adversarial Networks,* Karras et al.

StyleGAN



Latent $\mathbf{z} \in Z$
Normalize
Mapping network f
FC
FC
FC
FC
FC
FC
FC
FC
$\boldsymbol{w} \in W$
Synthesis network g
Noise
Const 4 × 4 × 512
A
AdaIN
Conv 3 × 3
A
AdaIN
4×4
B
B
Upsample
Conv 3 × 3
B
A
AdaIN
Conv 3 × 3
B
A
AdaIN
8×8
...

StyleGAN Demo

- https://thispersondoesnotexist.com/

Conditional GAN: Pix2Pix

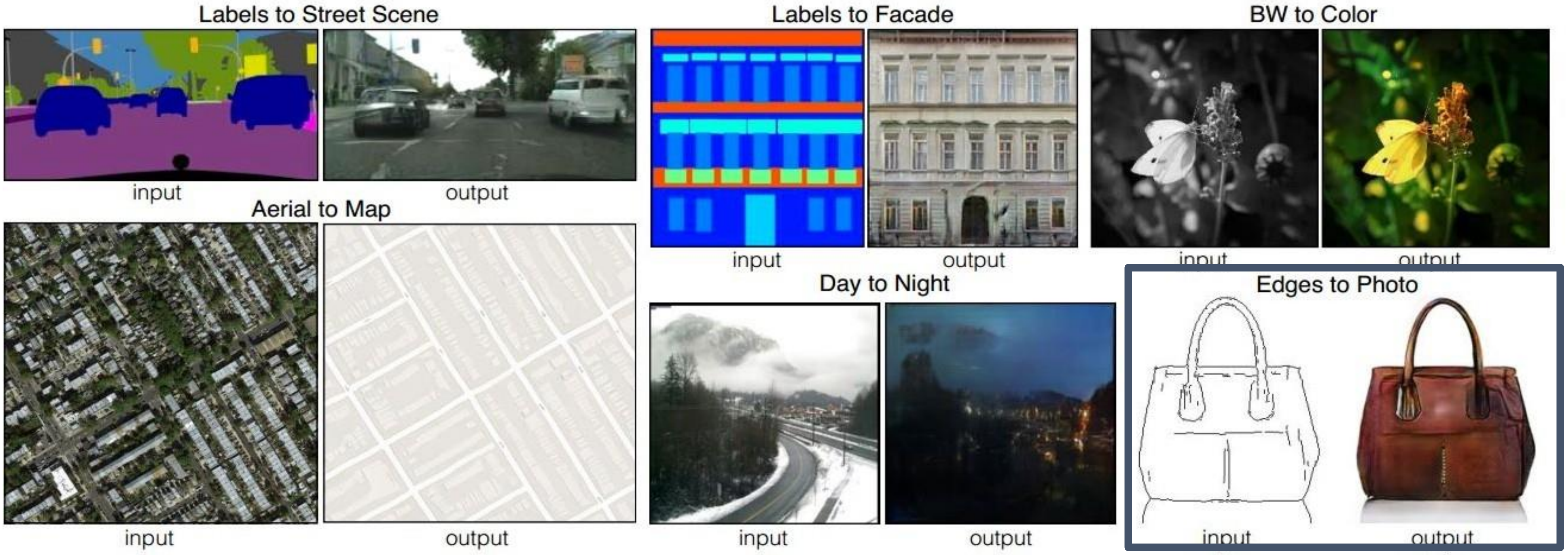


Image-to-image Translation with Conditional Adversarial Nets

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. CVPR 2017

[CreativeAI – SIGGRAPH Course]

slide credit: Phillip Isola & Jun-Yan Zhu

Edges → Images

Input Output Input Output Input Output

Edges from [Xie & Tu, 2015]

Sketches → Images

Input Output Input Output Input Output

Trained on Edges → Images

Data from [Eitz, Hays, Alexa, 2012]

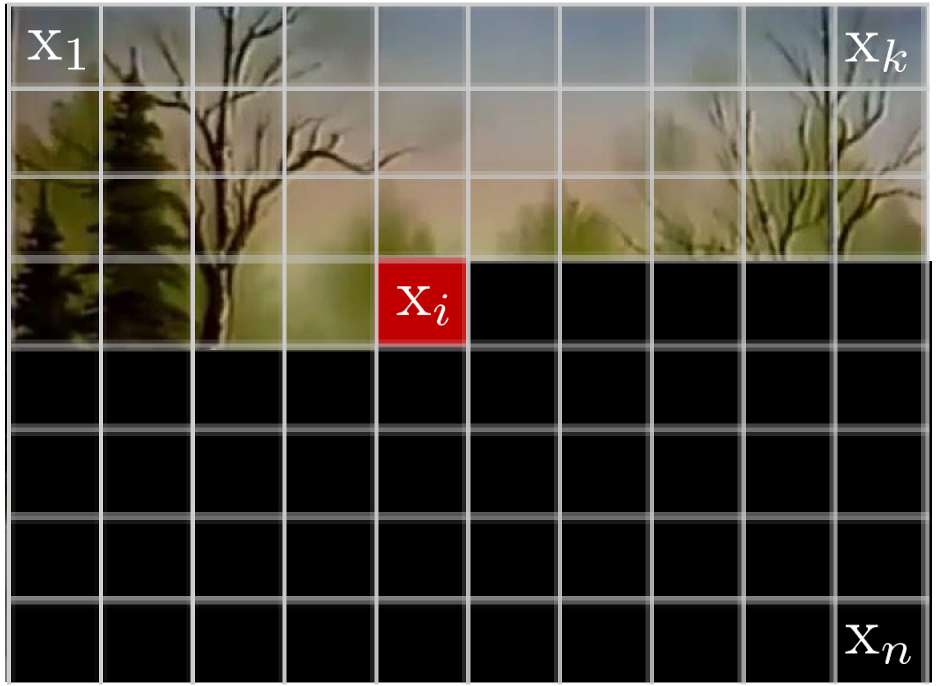
slide credit: Phillip Isola & Jun-Yan Zhu

Pix2Pix Demo

- https://affinelayer.com/pixsrv/

Autoregressive Models

- Create output step-by-step
- Each step depends on the output of all previous steps



$$x = [x_1, x_2, \ldots, x_n]$$

$$p_\theta(x) = p_\theta(x_1, x_2, \ldots, x_n)$$

Chain rule of probability:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, x_2, \ldots, x_{i-1})$$

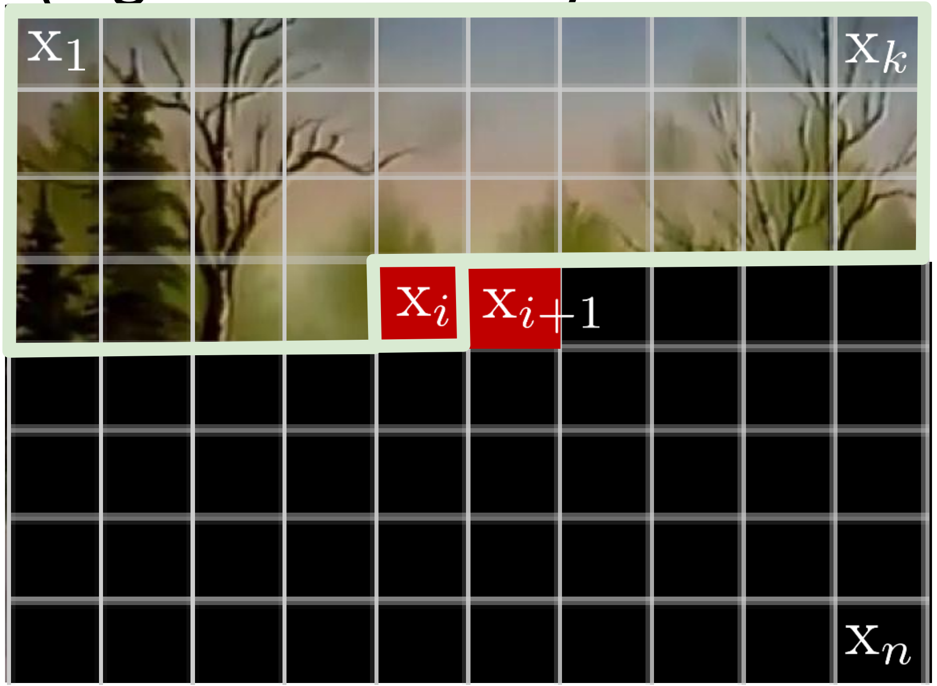
[CreativeAI – SIGGRAPH Course]

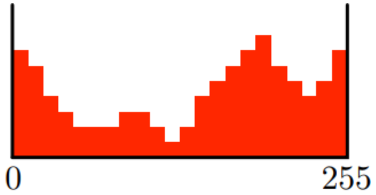
Video Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

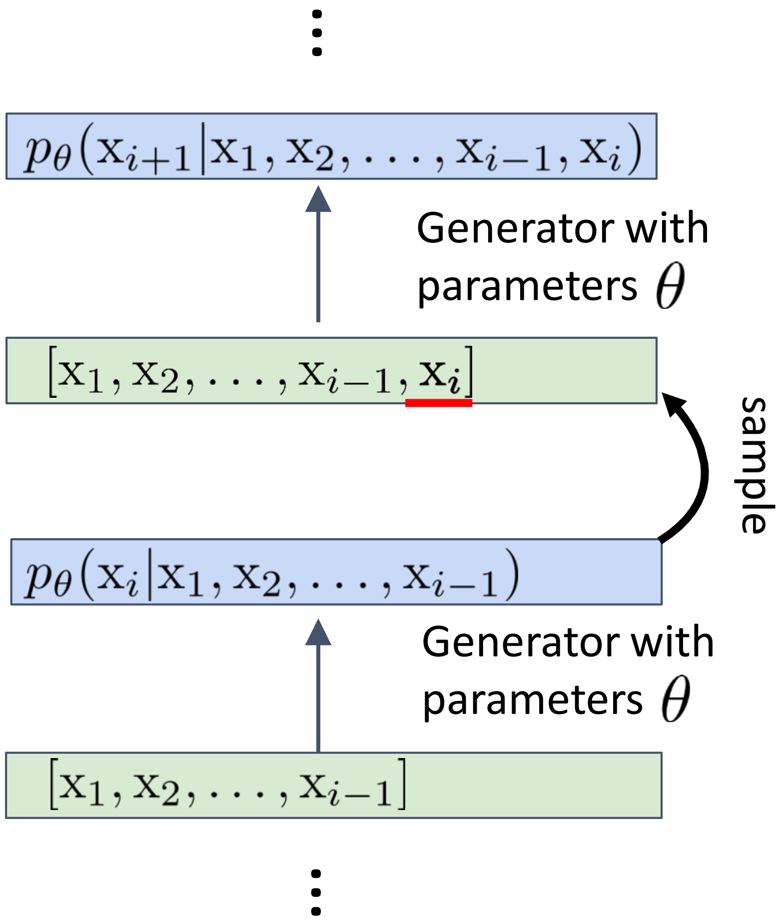
Autoregressive Models

Sample?
Evaluate?

- In each step, the model outputs a low-dimensional prob. distribution (e.g. over intensity values for one pixel)







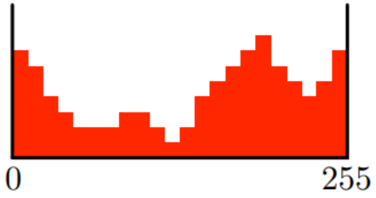
[CreativeAI – SIGGRAPH Course]

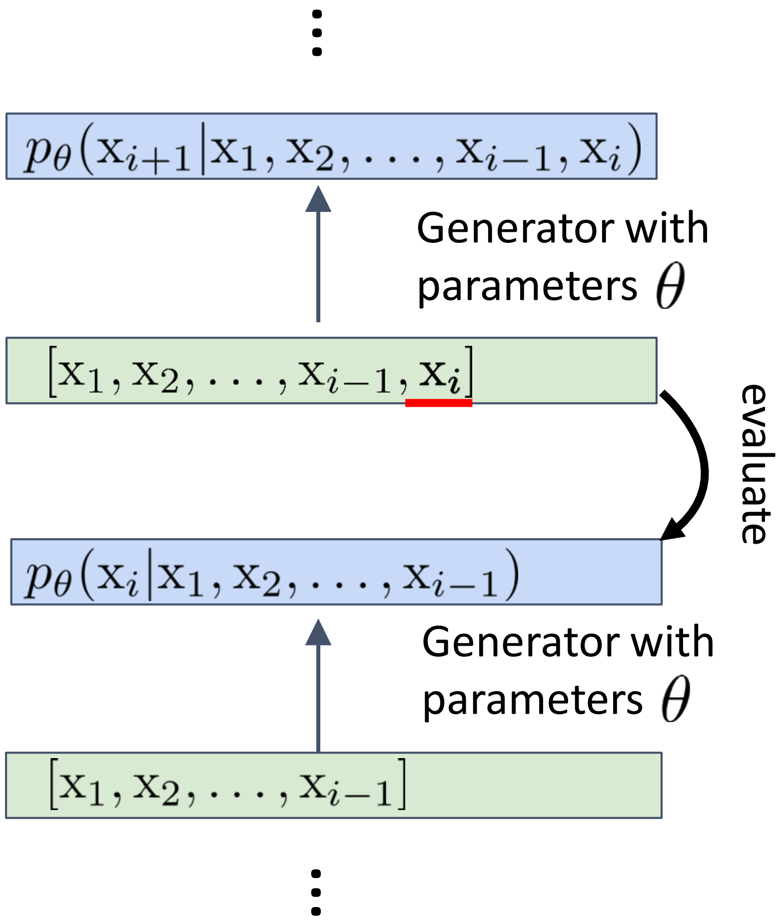
Video Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

Autoregressive Models

Sample?
Evaluate?

- In each step, the model outputs a low-dimensional prob. distribution (e.g. over intensity values for one pixel)



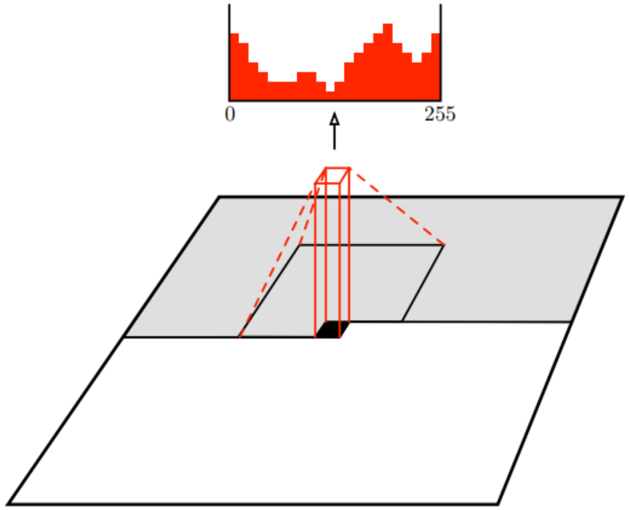


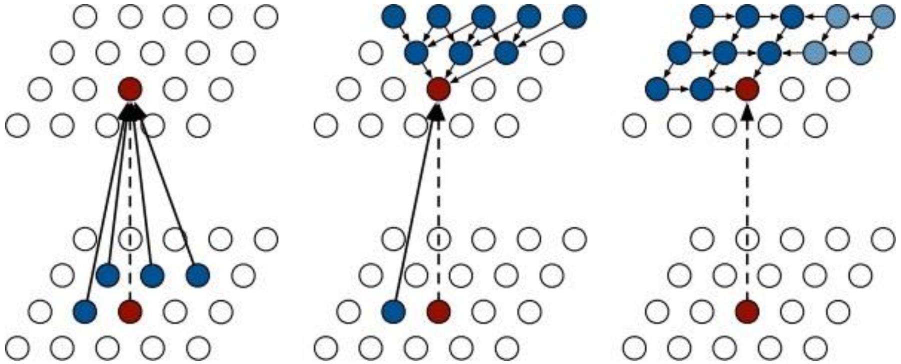
[CreativeAI – SIGGRAPH Course]

Image Credit: YouTube user *karwan kalary, Bob Ross Time Lapse*

Example: PixelRNN and PixelCNN

- Recursive network that has an **input** and a **state** (LSTM)
- Only recent steps are used as **input**, the **state** summarizes older steps





Sandbar

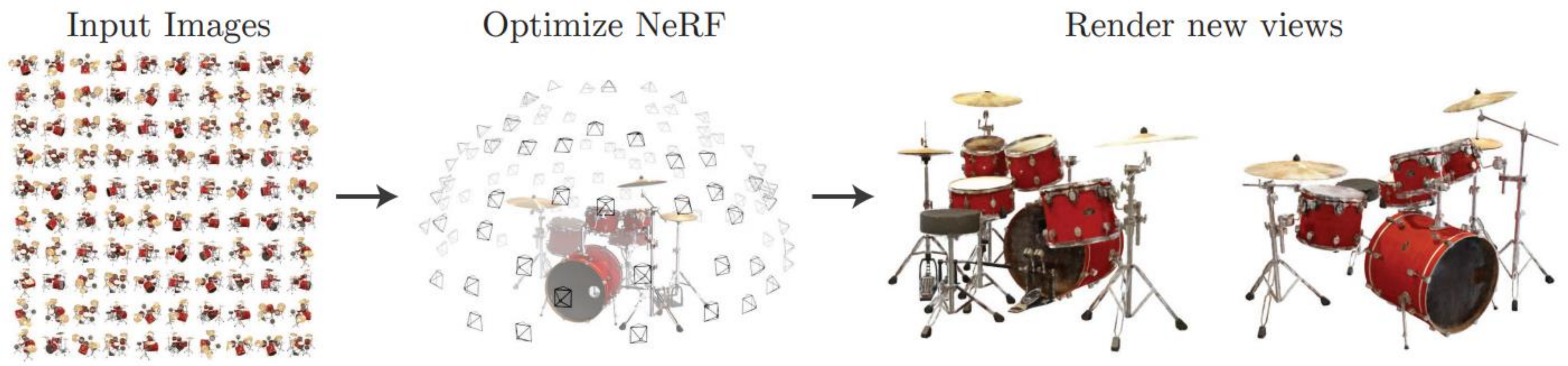
Lhasa Apso (dog)

Brown bear

[CreativeAI – SIGGRAPH Course]

Neural Reflectance Field (NERF)

- A neural radiance field (NeRF) is a fully-connected neural network that can generate novel views of complex 3D scenes, based on a partial set of 2D images



- (deep learning version of “Lightfields” – see other slides)

NERF

Instant-NERF: https://blogs.nvidia.com/blog/2022/03/25/instant-nerf-research-3d-ai/

Other NERFs: https://datagen.tech/guides/synthetic-data/neural-radiance-field-nerf/

Diffusion Models

- From noise to data…

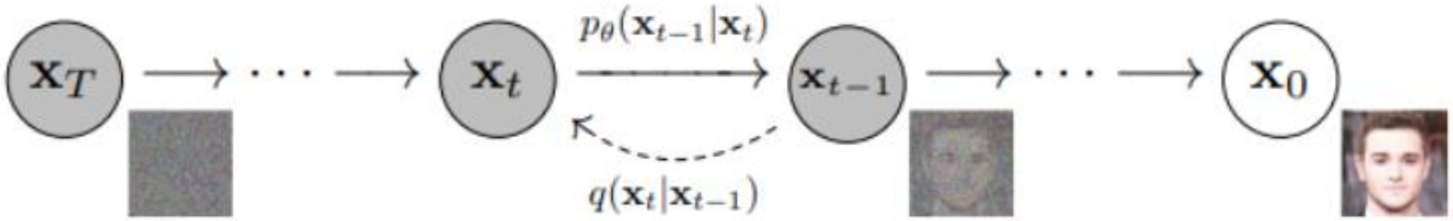
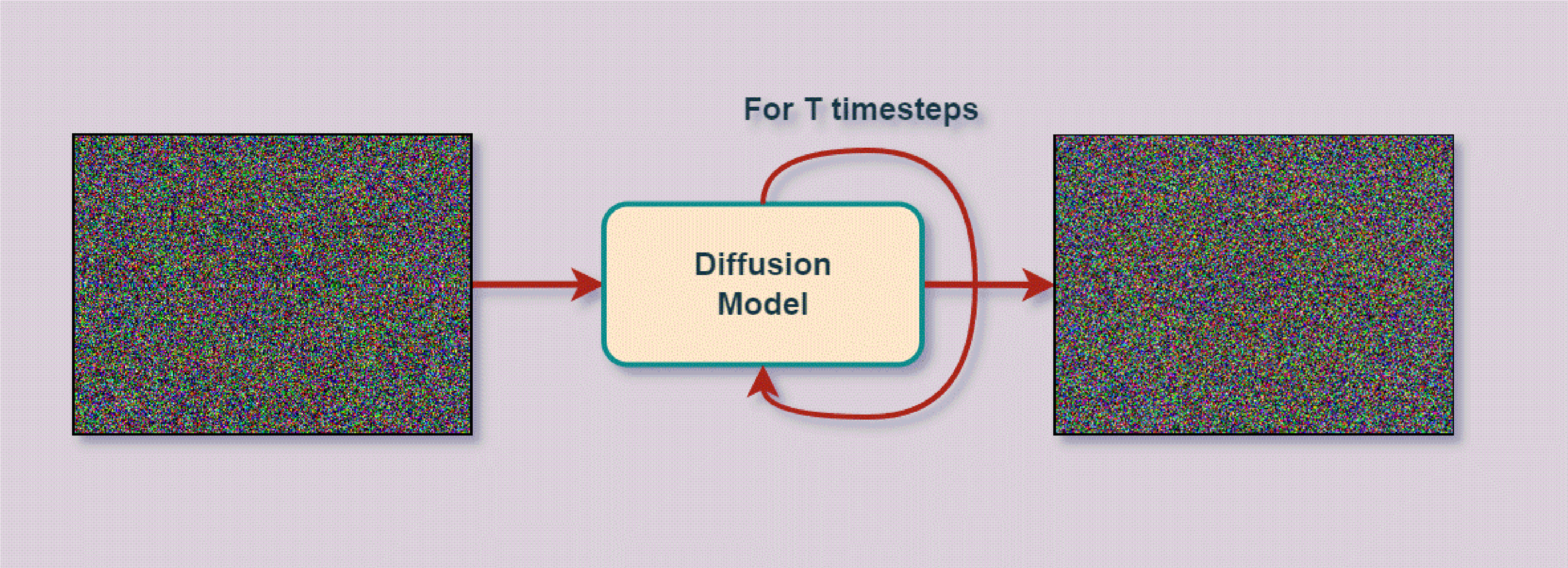


Figure 2: The directed graphical model considered in this work.

- Four popular diffusion models:
  - OpenAI’s Dall-E 2
  - Google’s Imagen
  - StabilityAI’s Stable Diffusion
  - Midjourney

Diffusion Models



[https://learnopencv.com/image-generation-using-diffusion-models/]