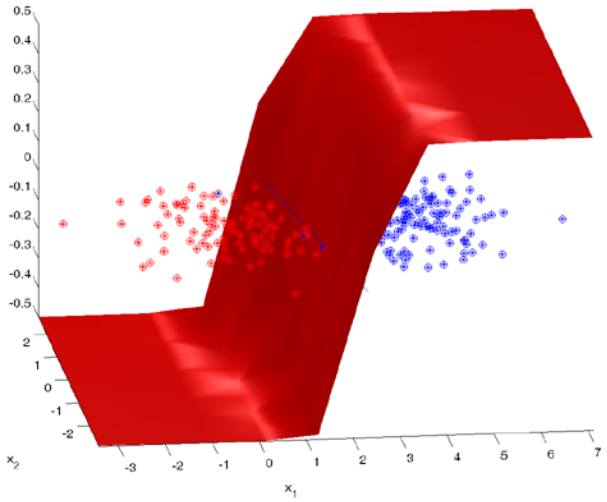
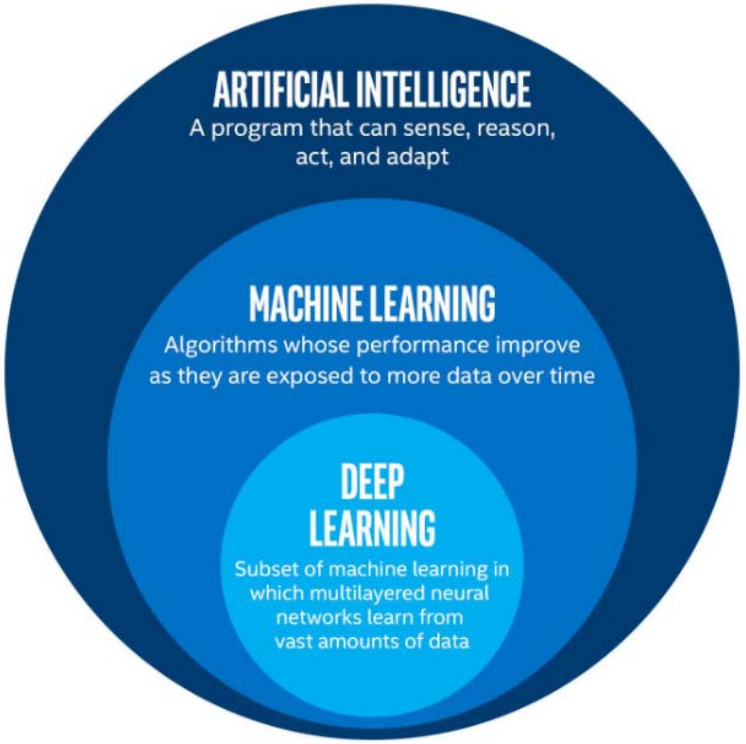
Introduction to Deep Learning

Introduction (2)

Prof. Songhwai Oh

ECE, SNU



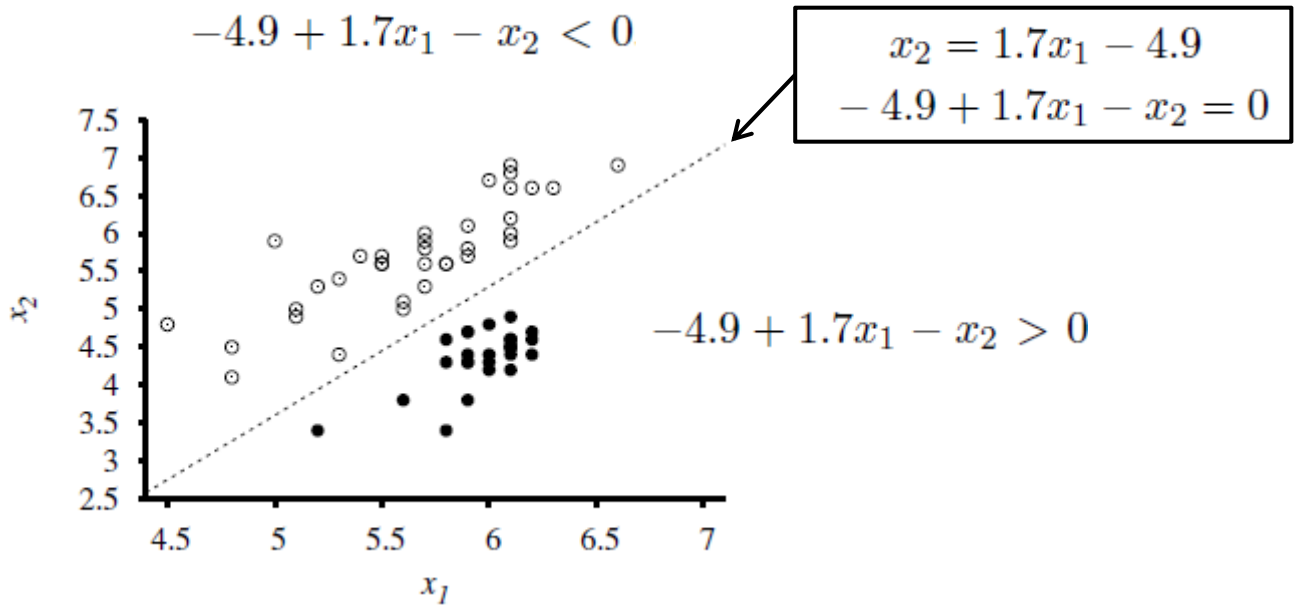


LINEAR CLASSIFICATION

Linear Classifiers

- Example
  - Classes: Earthquakes (0), Underground nuclear explosions (1)
  - Input values: Body wave magnitudes, Surface wave magnitudes
- **Decision boundary**: a line separating two classes.

Linearly Separable Case

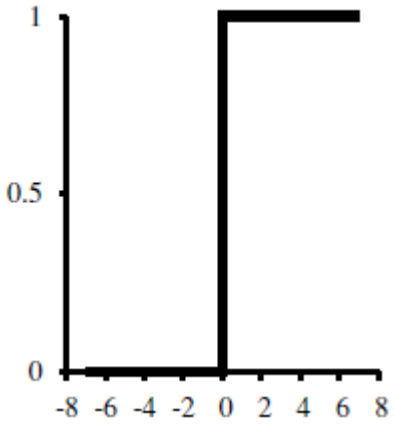


$$h_{\mathbf{w}}(\mathbf{x}) = 1 \text{ if } \mathbf{w} \cdot \mathbf{x} \geq 0 \text{ and } 0 \text{ otherwise.}$$

Perceptron Learning Rule

$$h_{\mathbf{w}}(\mathbf{x}) = \mathit{Threshold}(\mathbf{w} \cdot \mathbf{x}) \text{ where } \mathit{Threshold}(z) = 1 \text{ if } z \geq 0 \text{ and } 0 \text{ otherwise.}$$

Threshold function



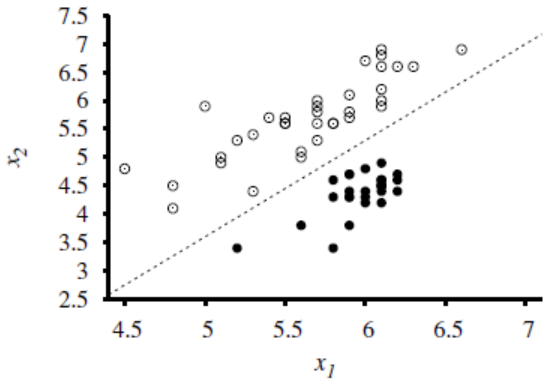
Update rule: $w_i \leftarrow w_i + \alpha\,(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$

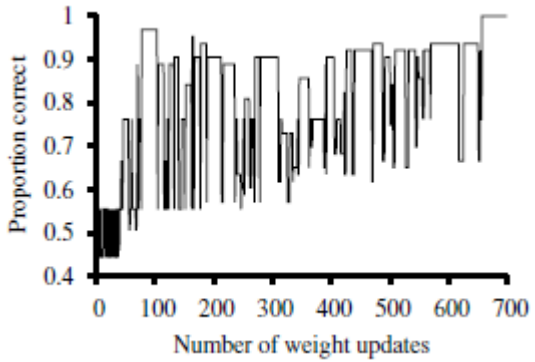
(converges if the problem is linearly separable.)

- If the output is correct, i.e., $y = h_{\mathbf{w}}(\mathbf{x})$, then the weights are not changed.
- If y is 1 but $h_{\mathbf{w}}(\mathbf{x})$ is 0, then w_i is *increased* when the corresponding input x_i is positive and *decreased* when x_i is negative. This makes sense, because we want to make $\mathbf{w} \cdot \mathbf{x}$ bigger so that $h_{\mathbf{w}}(\mathbf{x})$ outputs a 1.
- If y is 0 but $h_{\mathbf{w}}(\mathbf{x})$ is 1, then w_i is *decreased* when the corresponding input x_i is positive and *increased* when x_i is negative. This makes sense, because we want to make $\mathbf{w} \cdot \mathbf{x}$ smaller so that $h_{\mathbf{w}}(\mathbf{x})$ outputs a 0.

Learning Curve

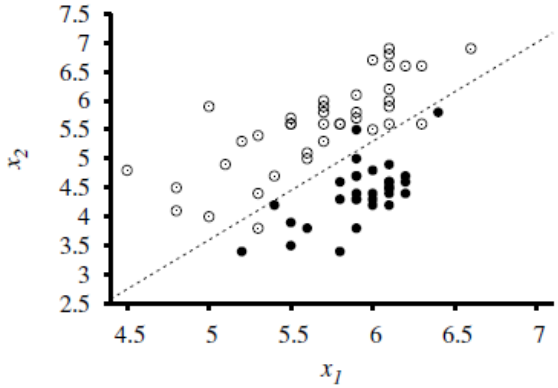
Separable case

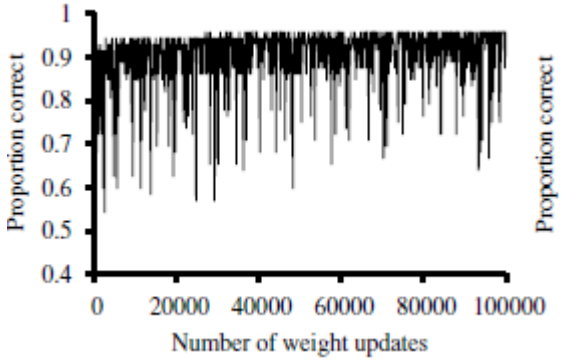




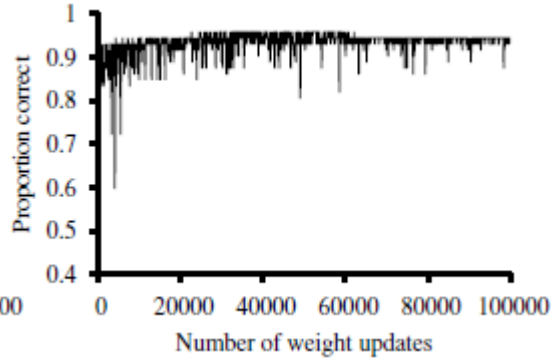
Learning curve

Non-separable case





Learning curve
(constant learning rate)

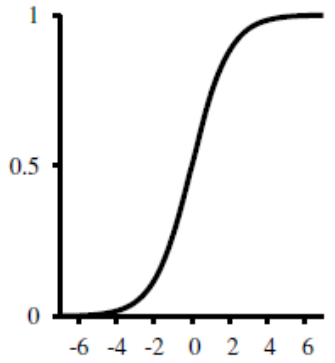


Learning curve
(decreasing learning rate)

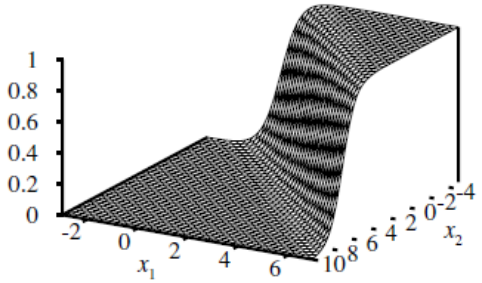
Logistic Regression

Logistic function

$$Logistic(z) = \frac{1}{1+e^{-z}}$$



Soft thresholding



$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w}\cdot\mathbf{x})$$

Logistic regression

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w}\cdot\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}$$

$$\begin{aligned}
\frac{\partial}{\partial w_i} Loss(\mathbf{w}) &= \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x}))^2 \\
&= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x})) \\
\text{(chain rule)} \quad &= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w}\cdot\mathbf{x}) \times \frac{\partial}{\partial w_i}\mathbf{w}\cdot\mathbf{x} \\
&= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w}\cdot\mathbf{x}) \times x_i \,.
\end{aligned}$$

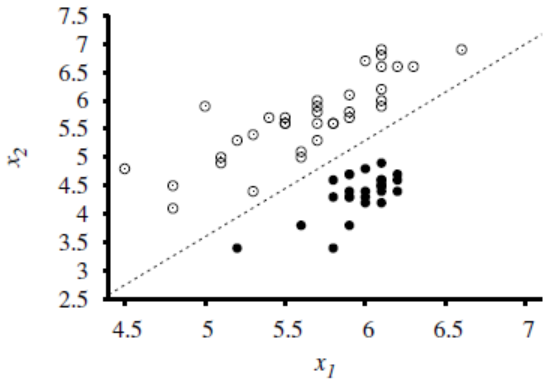
If g is a logistic function, then

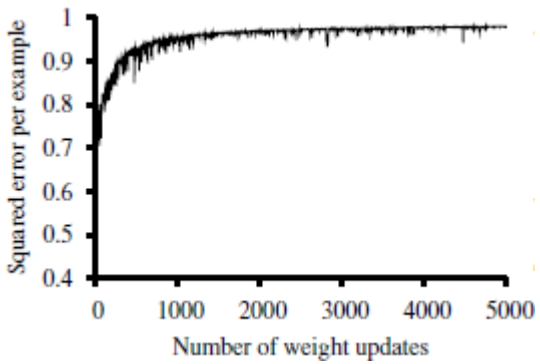
$$g'(z) = \frac{dg}{dz}(z) = g(z)(1-g(z)).$$

$$g'(\mathbf{w}\cdot\mathbf{x}) = g(\mathbf{w}\cdot\mathbf{x})(1-g(\mathbf{w}\cdot\mathbf{x})) = h_{\mathbf{w}}(\mathbf{x})(1-h_{\mathbf{w}}(\mathbf{x}))$$

$$w_i \leftarrow w_i + \alpha\,(y - h_{\mathbf{w}}(\mathbf{x})) \times h_{\mathbf{w}}(\mathbf{x})(1-h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

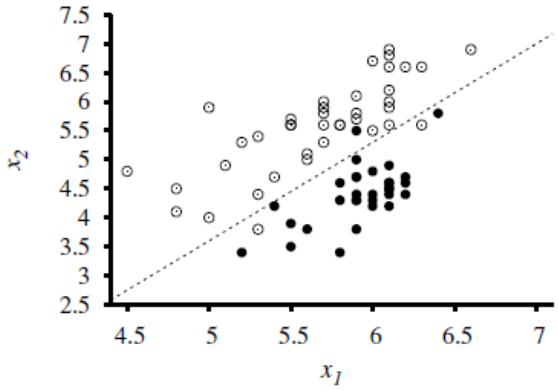
Separable case

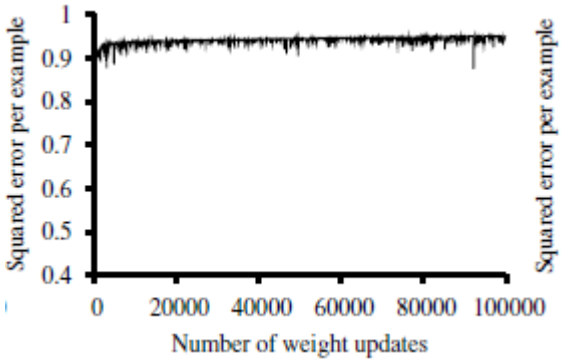




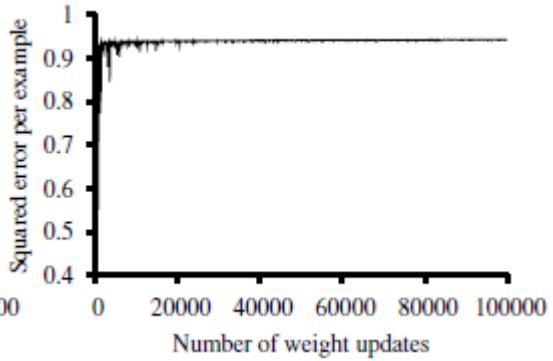
Learning curve

Non-separable case





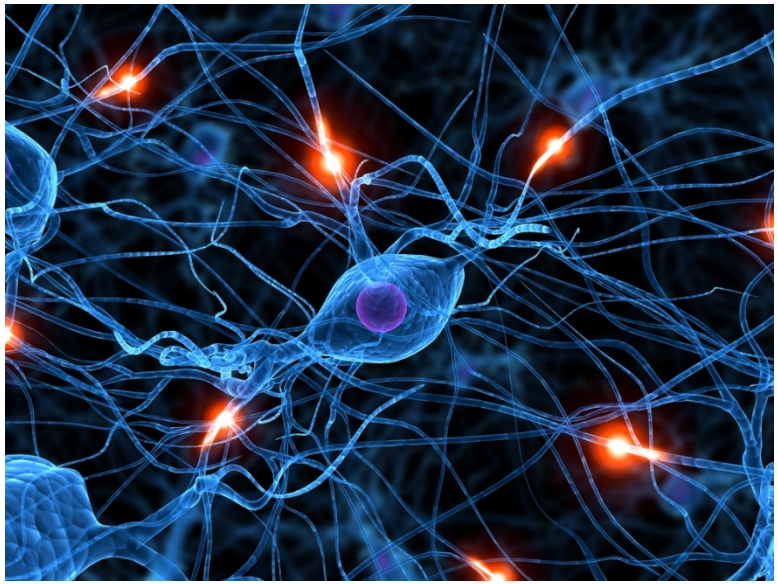
Learning curve
(constant learning rate)



Learning curve
(decreasing learning rate)

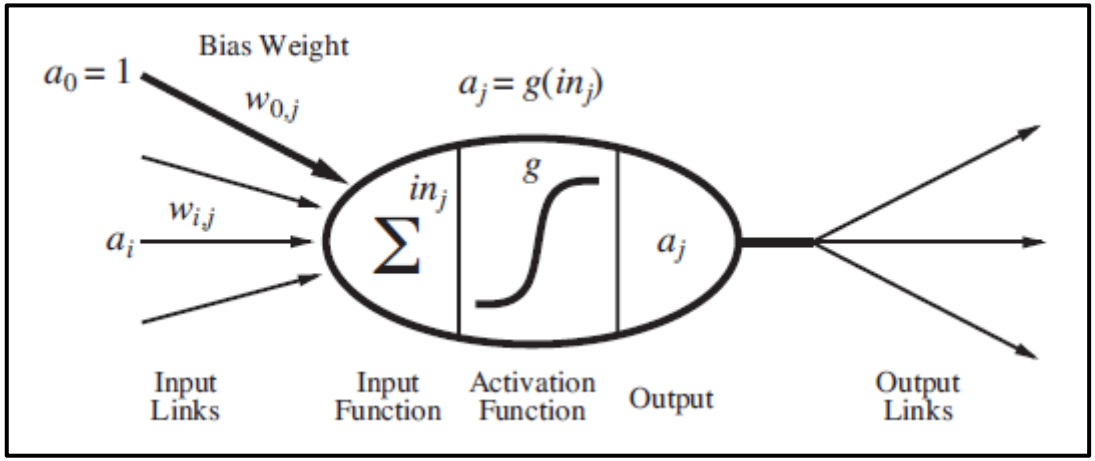
Human brain

- 100 billion neurons
- 100 to 500 trillion synapses



ARTIFICIAL NEURAL NETWORKS (ANN)

Neural Network Structure

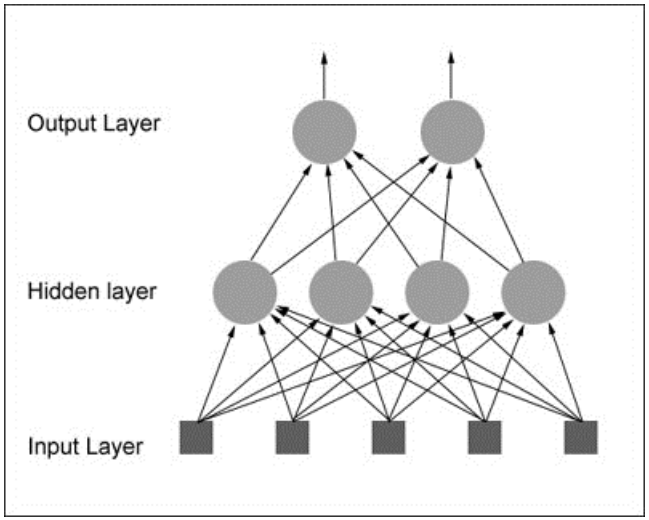


$$in_j = \sum_{i=0}^{n} w_{i,j} a_i$$

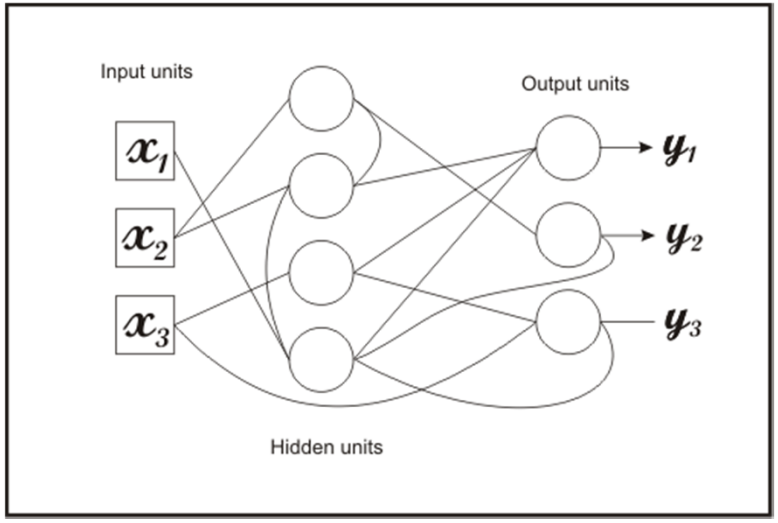
$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{i,j} a_i\right)$$

- Perceptron: hard thresholding
- Sigmoid perceptron: soft thresholding, e.g., logistic function

Feed-forward network

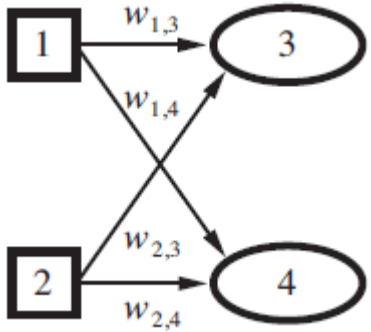


Recurrent network

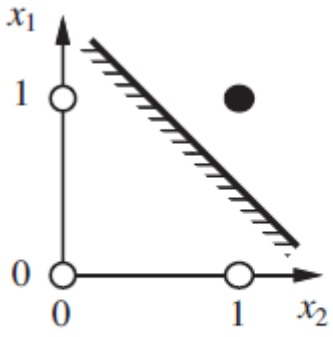


Single-Layer Feed-Forward Neural Networks

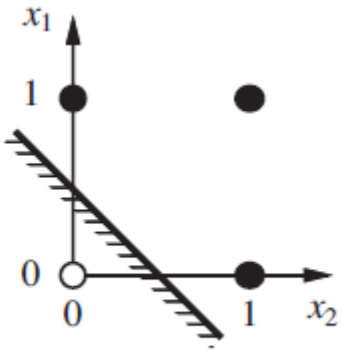
x_1	x_2	y_3 (carry)	y_4 (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



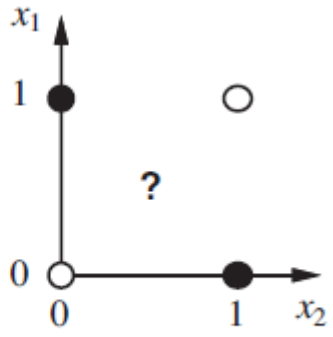
- Perceptron learning rule
- Logistic regression



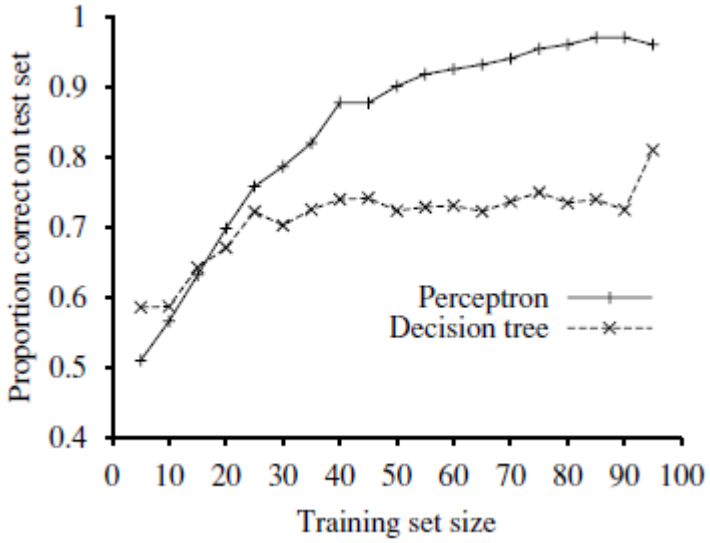
(a) x_1 **and** x_2



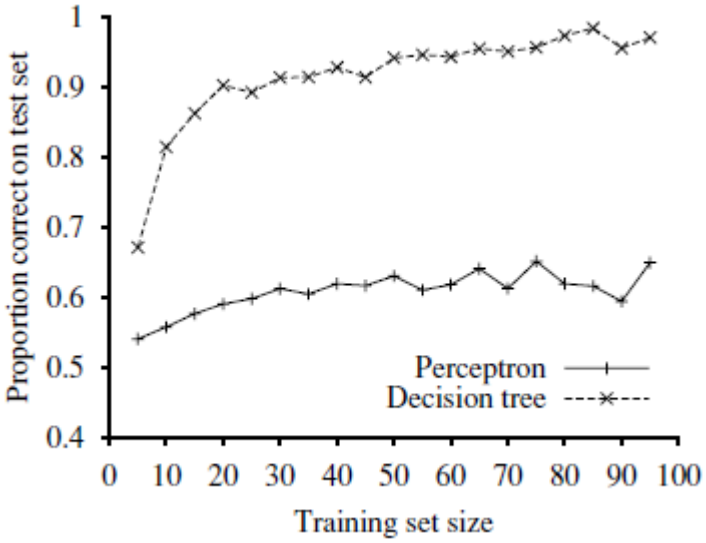
(b) x_1 **or** x_2



(c) x_1 **xor** x_2

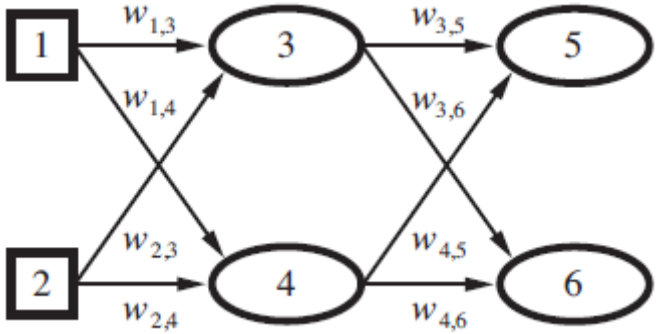


Majority function
(11 Boolean inputs)



WillWait
(Restaurant example)

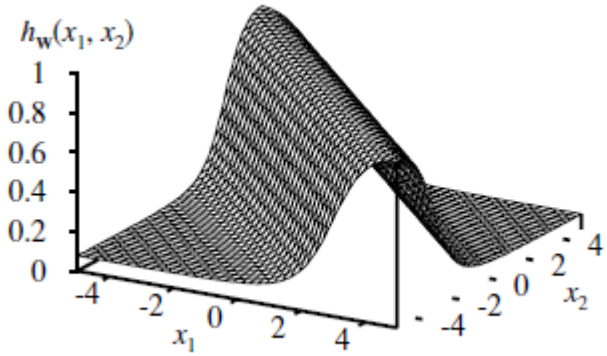
Multilayer Feed-Forward Neural Networks

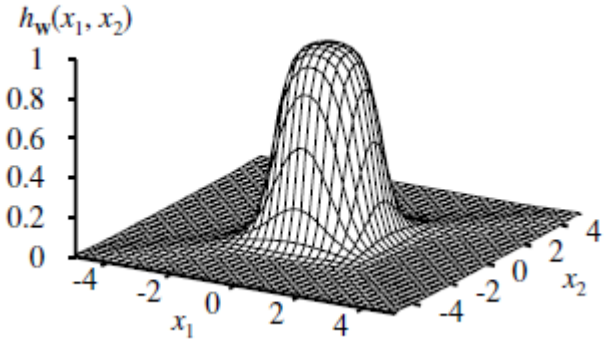


input units | hidden units | output units

Input units: $(a_1, a_2) = (x_1, x_2)$

$$
\begin{aligned}
a_5 &= g(w_{0,5,+} w_{3,5}\, a_3 + w_{4,5}\, a_4) \\
&= g(w_{0,5,+} w_{3,5}\, g(w_{0,3} + w_{1,3}\, a_1 + w_{2,3}\, a_2) \\
&\quad + w_{4,5}\, g(w_0 4 + w_{1,4}\, a_1 + w_{2,4}\, a_2)) \\
&= g(w_{0,5,+} w_{3,5}\, g(w_{0,3} + w_{1,3}\, x_1 + w_{2,3}\, x_2) \\
&\quad + w_{4,5}\, g(w_0 4 + w_{1,4}\, x_1 + w_{2,4}\, x_2))
\end{aligned}
$$





An ANN with a single (sufficiently large) hidden layer can represent any continuous function.

Back-Propagation

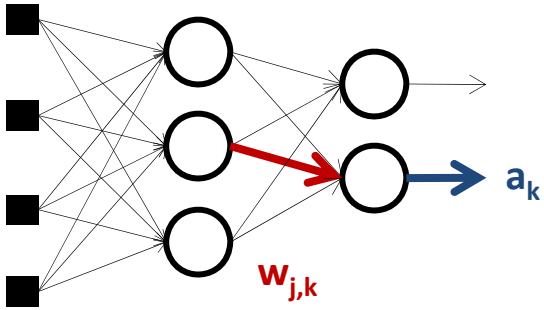
- Let $\mathbf{h_w}$ be a nonlinear function representing a multilayer feed-forward neural network, where $\mathbf{w}$ is a weight vector.
- We can learn a neural network by finding weights using a gradient descent algorithm.
- Gradient descent update rule:

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w}).$$

$$\frac{\partial}{\partial w} Loss(\mathbf{w}) = \frac{\partial}{\partial w} |\mathbf{y} - \mathbf{h_w}(\mathbf{x})|^2 = \frac{\partial}{\partial w} \sum_k (y_k - a_k)^2 = \sum_k \frac{\partial}{\partial w} (y_k - a_k)^2$$

from the j^{th} hidden unit to the k^{th} output

$$Loss_k = (y_k - a_k)^2$$



$$\frac{\partial Loss_k}{\partial w_{j,k}} = -2(y_k - a_k)\frac{\partial a_k}{\partial w_{j,k}} = -2(y_k - a_k)\frac{\partial g(in_k)}{\partial w_{j,k}}$$

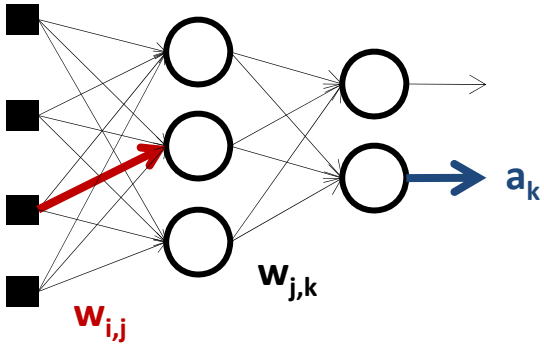
$$= -2(y_k - a_k)g'(in_k)\frac{\partial in_k}{\partial w_{j,k}} = -2(y_k - a_k)g'(in_k)\frac{\partial}{\partial w_{j,k}}\left(\sum_j w_{j,k}a_j\right)$$

$$= -2\underbrace{(y_k - a_k)g'(in_k)}_{\Delta_k}a_j = -a_j\Delta_k \ ,$$

from the i^{th} input to the j^{th} hidden unit

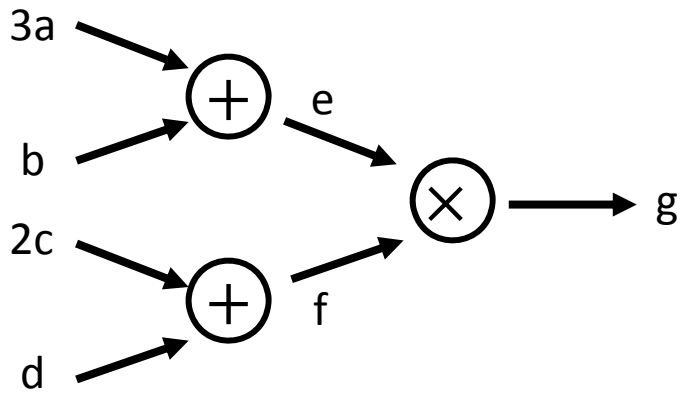
$$Loss_k = (y_k - a_k)^2$$

$$\begin{aligned}
\frac{\partial Loss_k}{\partial w_{i,j}} &= -2(y_k - a_k)\frac{\partial a_k}{\partial w_{i,j}} = -2(y_k - a_k)\frac{\partial g(in_k)}{\partial w_{i,j}} \\
&= -2(y_k - a_k)g'(in_k)\frac{\partial in_k}{\partial w_{i,j}} = -2\Delta_k \frac{\partial}{\partial w_{i,j}}\left(\sum_j w_{j,k}a_j\right) \\
&= -2\Delta_k w_{j,k}\frac{\partial a_j}{\partial w_{i,j}} = -2\Delta_k w_{j,k}\frac{\partial g(in_j)}{\partial w_{i,j}} \\
&= -2\Delta_k w_{j,k}g'(in_j)\frac{\partial in_j}{\partial w_{i,j}} \\
&= -2\Delta_k w_{j,k}g'(in_j)\frac{\partial}{\partial w_{i,j}}\left(\sum_i w_{i,j}a_i\right) \\
&= \underbrace{-2\Delta_k w_{j,k}g'(in_j)}_{\Delta_j}a_i = -a_i\Delta_j \ ,
\end{aligned}$$



Example

Computation graph



$$g = ef$$
$$e = 3a + b$$
$$f = 2c + d$$

$$g = (3a + b)(2c + d)$$

$$Loss = (y - g)^2$$

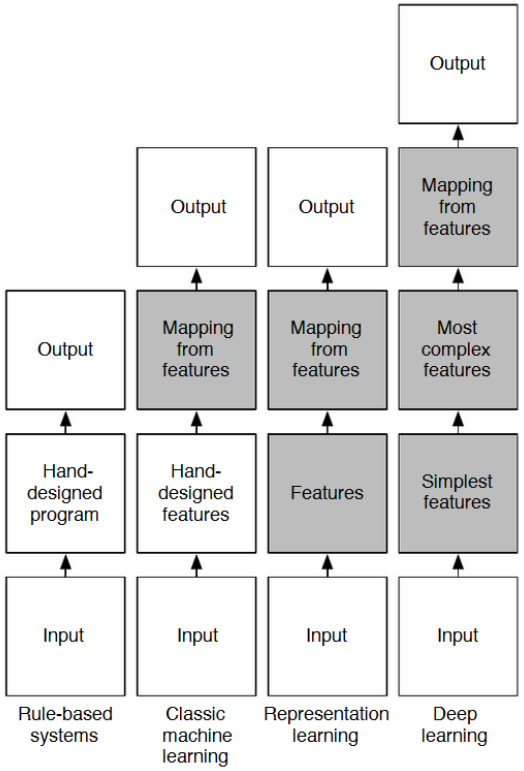
$$\frac{\partial Loss}{\partial a} = \frac{\partial}{\partial a}(y - g)^2$$

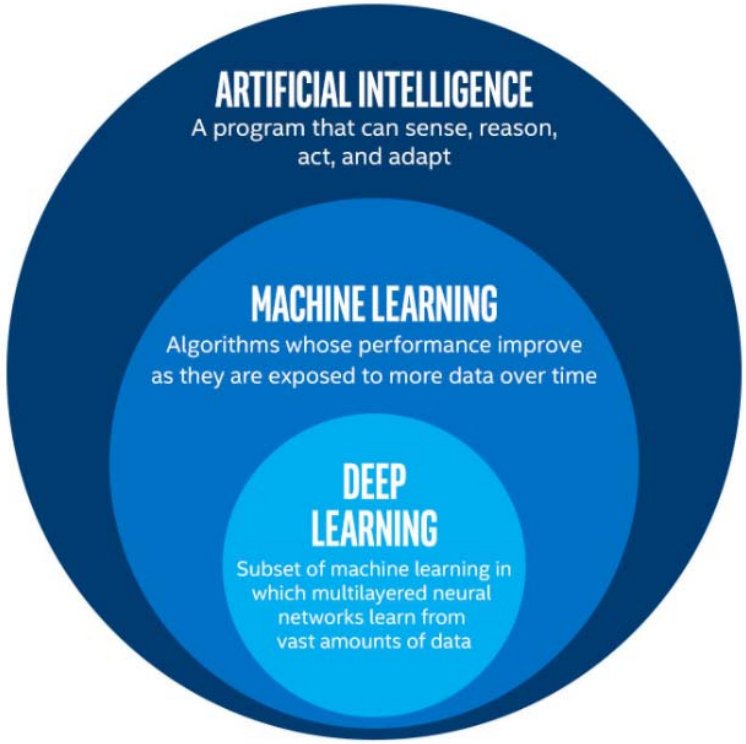
$$\frac{\partial Loss}{\partial a} = -2(y - g)\frac{\partial g}{\partial a} = -2(y - g)\frac{\partial g}{\partial e}\frac{\partial e}{\partial a} = -2(y - g) \cdot f \cdot 3$$
$$= -6(y - g)(2c + d)$$

Issues

- Overfitting
  - Complex model
  - Not enough data
- Vanishing/exploding gradient problem
  - Cannot train many layers of a network
- Other competing methods
  - Support vector machines
  - Bayesian networks

- Breakthroughs
  - Faster computers, GPUs
  - Cheap memory (enabling large data)
  - New techniques



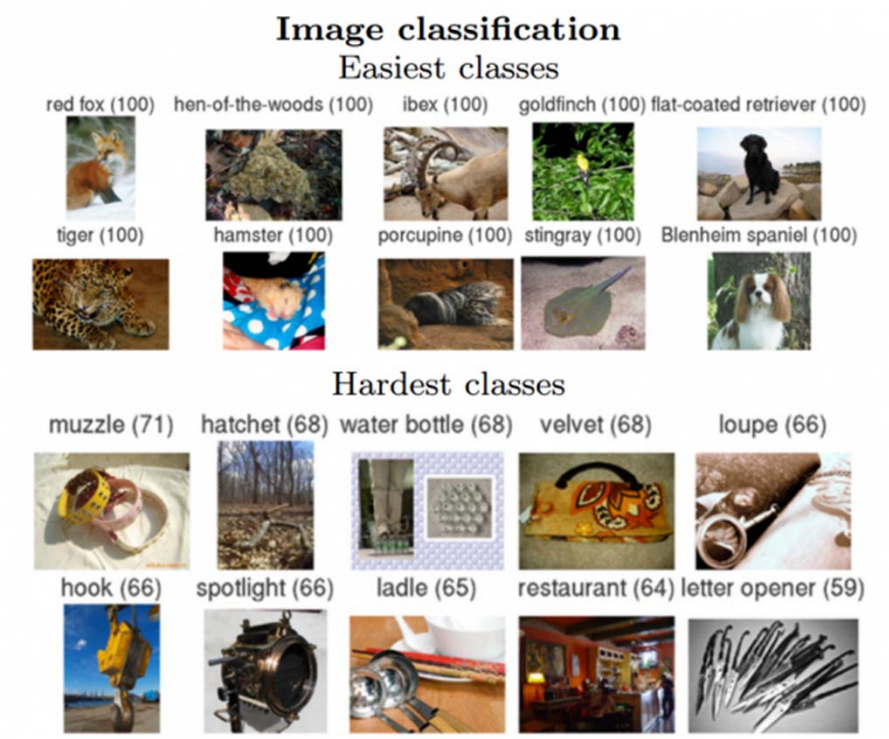


DEEP LEARNING

ImageNet Large-Scale Visual Recognition Challenge, 2012

Tasks:

- Decide whether a given image contains a particular type of object or not. For example, a contestant might decide that there are cars in this image but no tigers.
- Find a particular object and draw a box around it. For example, a contestant might decide that there is a screwdriver at a certain position with a width of 50 pixels and a height of 30 pixels.



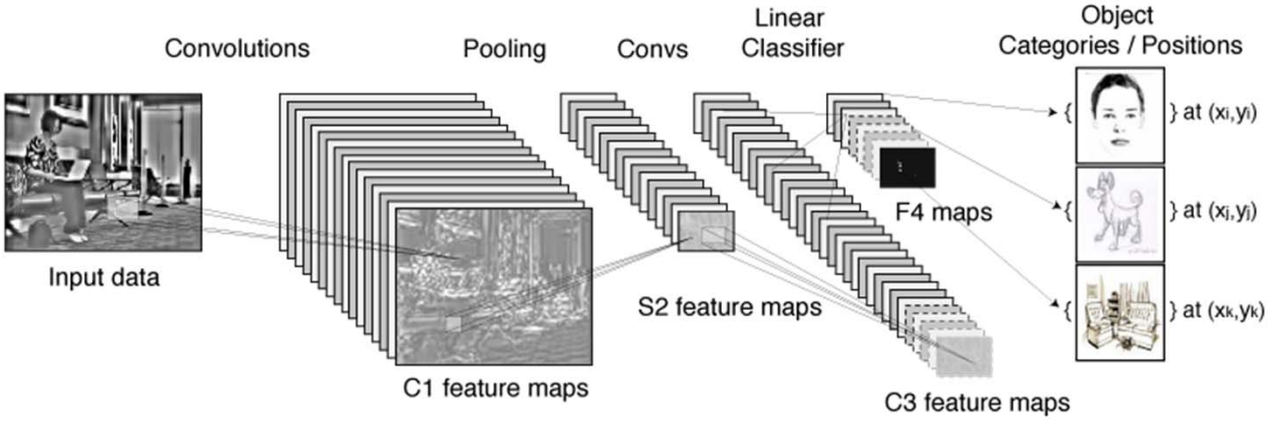
- 1000 different categories
- Over 1 million images
- Training set: 456,567 images

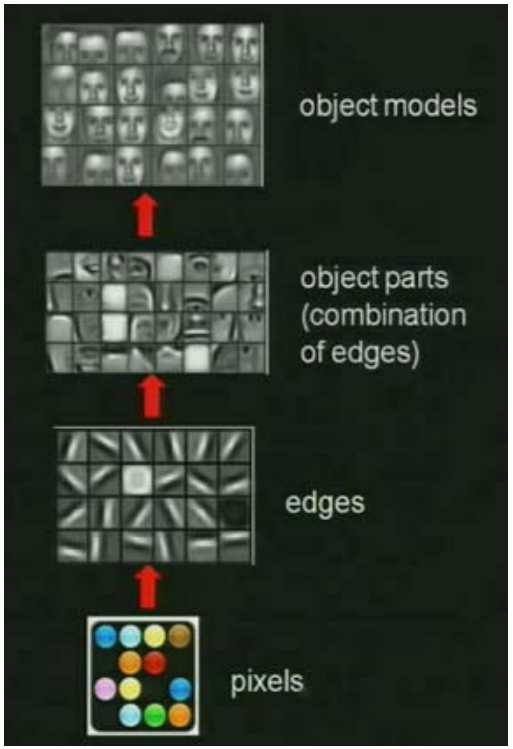
Year	Winning Error Rate
2010	28.2%
2011	25.8%
2012	16.4% (2nd 25.2%)
2013	11.2%
2014	6.7%
2015	3.57%
Human	**About 5.1%**

ImageNet Large Scale Visual Recognition Challenge. Russakovsky et al. *arXiv preprint arXiv:1409.0575*. URL: http://arxiv.org/abs/1409.0575v1

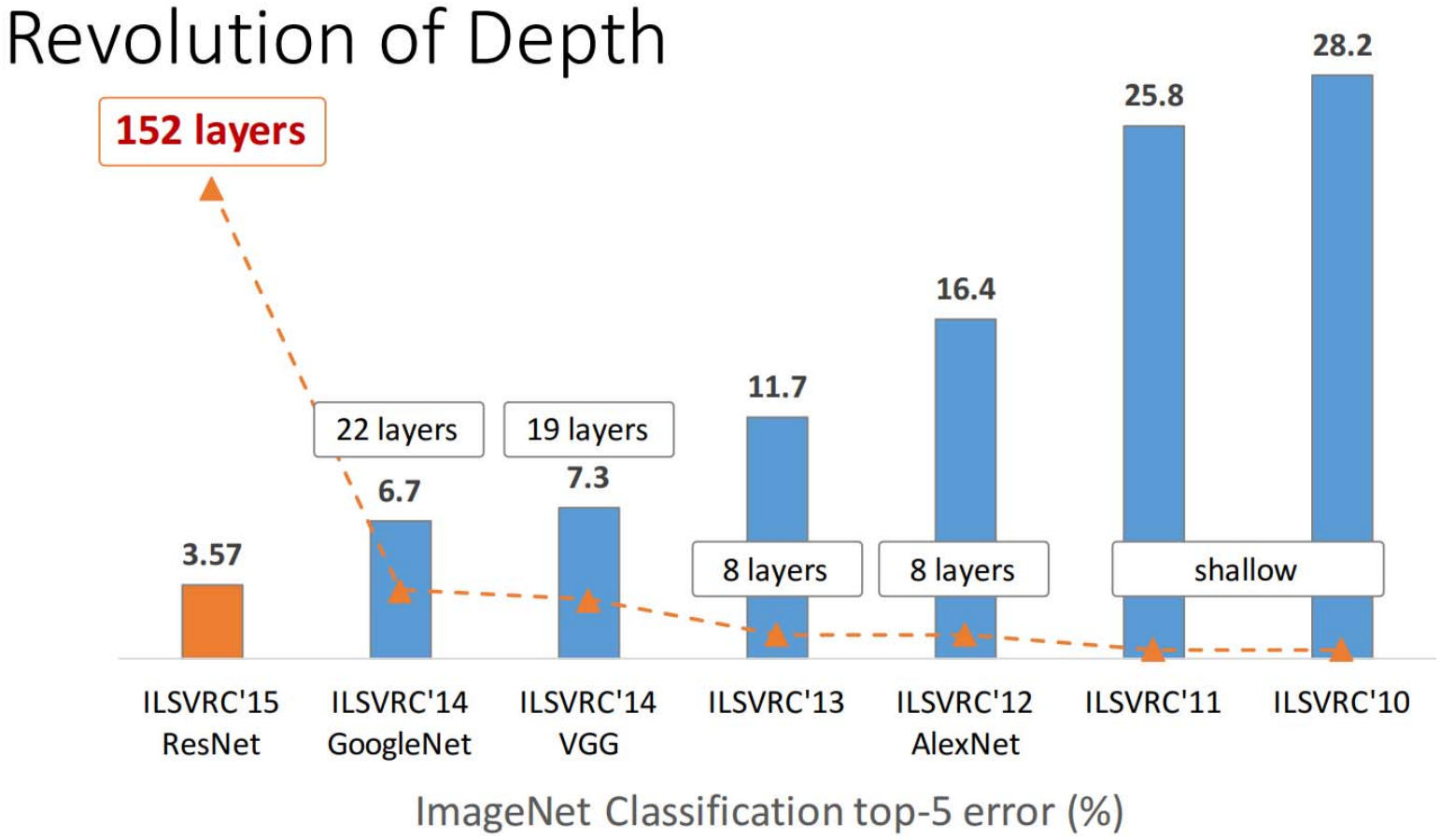
Convolutional Neural Networks (CNNs)

- SuperVision (2012)
  - Deep convolutional neural network
  - 650,000 neurons
  - 5 convolutional layers
  - Over 60 million parameters
- Clarifai (2013)
- GoogleLeNet (2014) – 22 layers
- ResNet (2015) – 152 layers



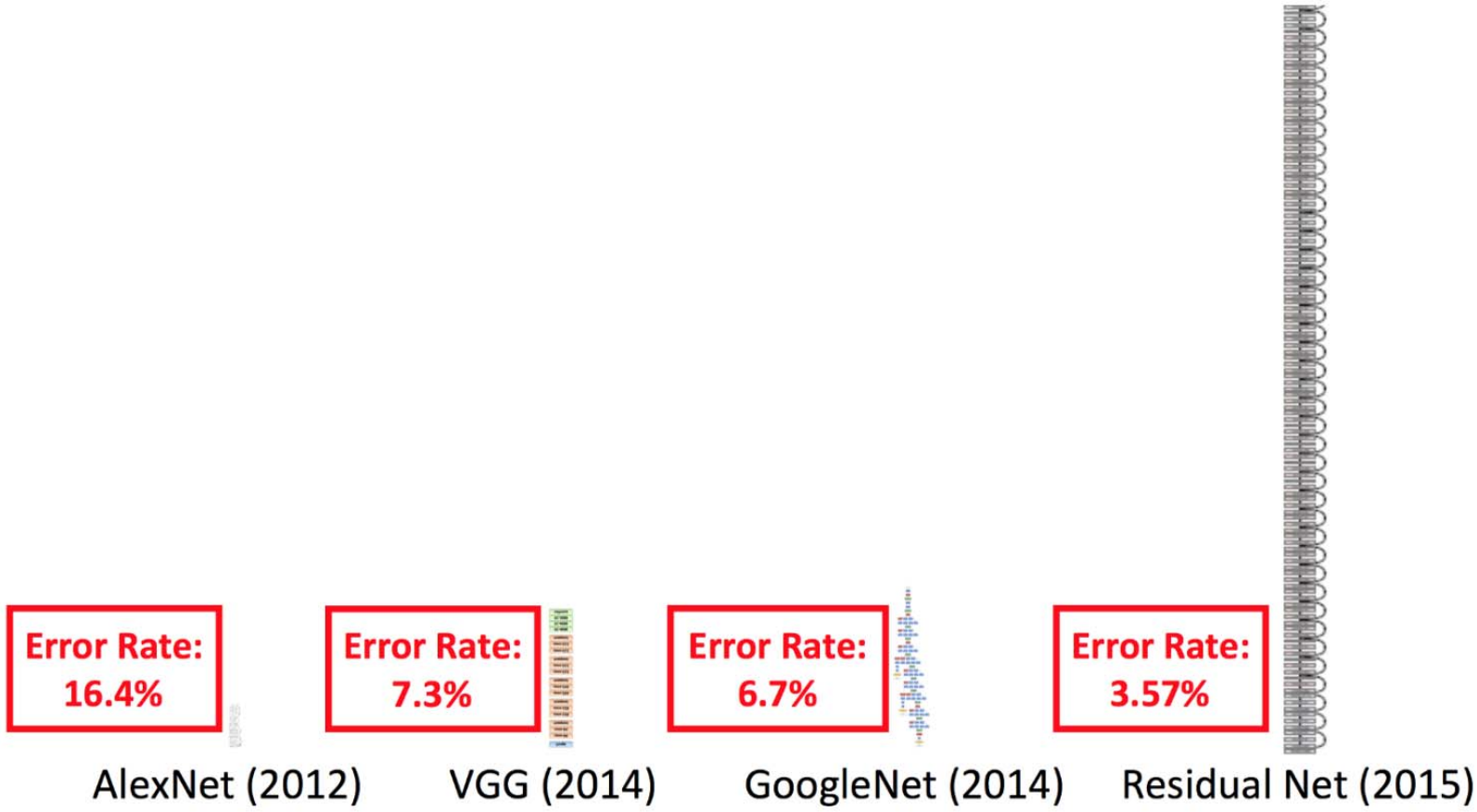


ImageNet Challenge



K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR, 2016.

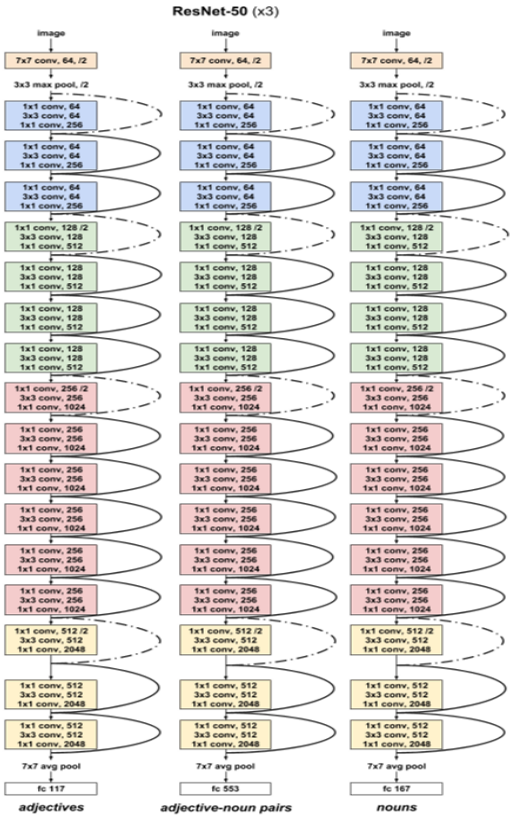
The Trend



https://medium.com/@k3083518729/tensorflow-image-recognition-58b0ac77c263

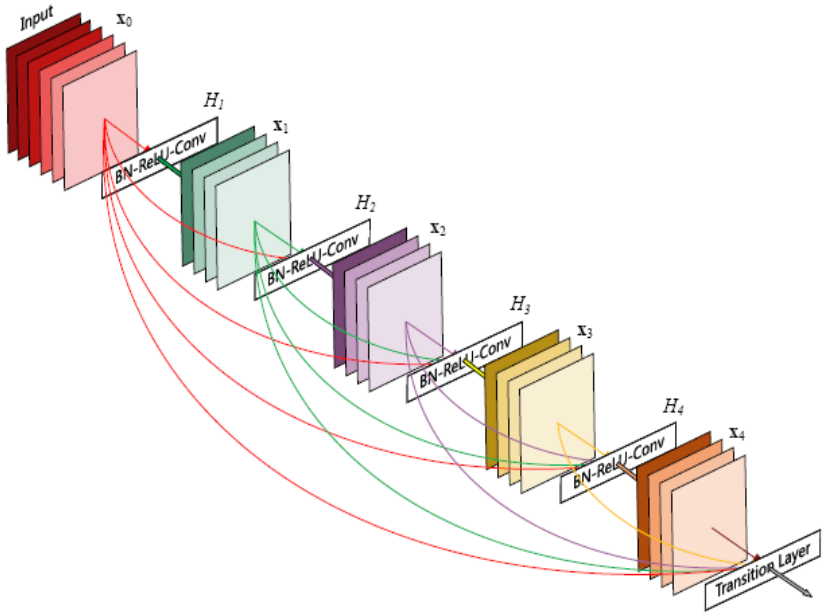
The Trend

Going deeper



ResNet (CVPR'16 Best Paper)

Going denser



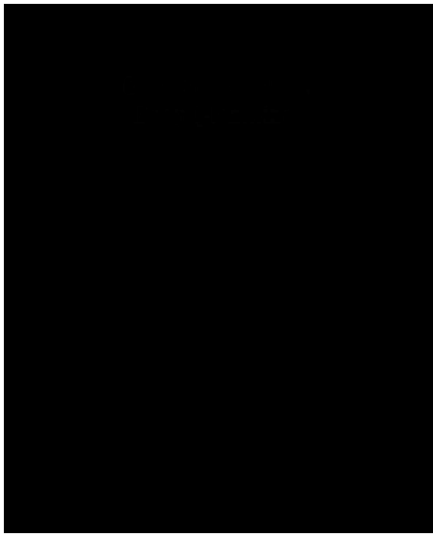
DenseNet (CVPR'17 Best Paper)

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR, 2016.
G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. CVPR, 2017.

DEEP REINFORCEMENT LEARNING

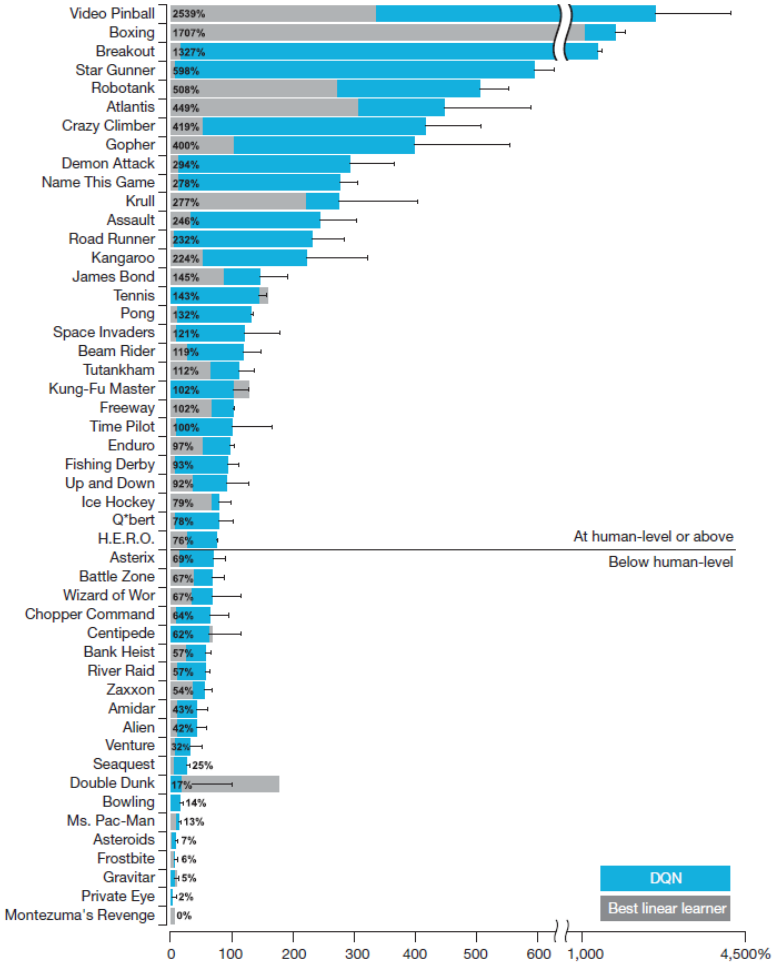
Deep Q-Network (DQN), 2015

- Playing Atari games
- Input: Game screen shots
- Output: Control (left, right, shoot, …)
- Convolutional neural networks (CNN)
- Reinforcement learning: Q-learning



Breakout

Space Invaders



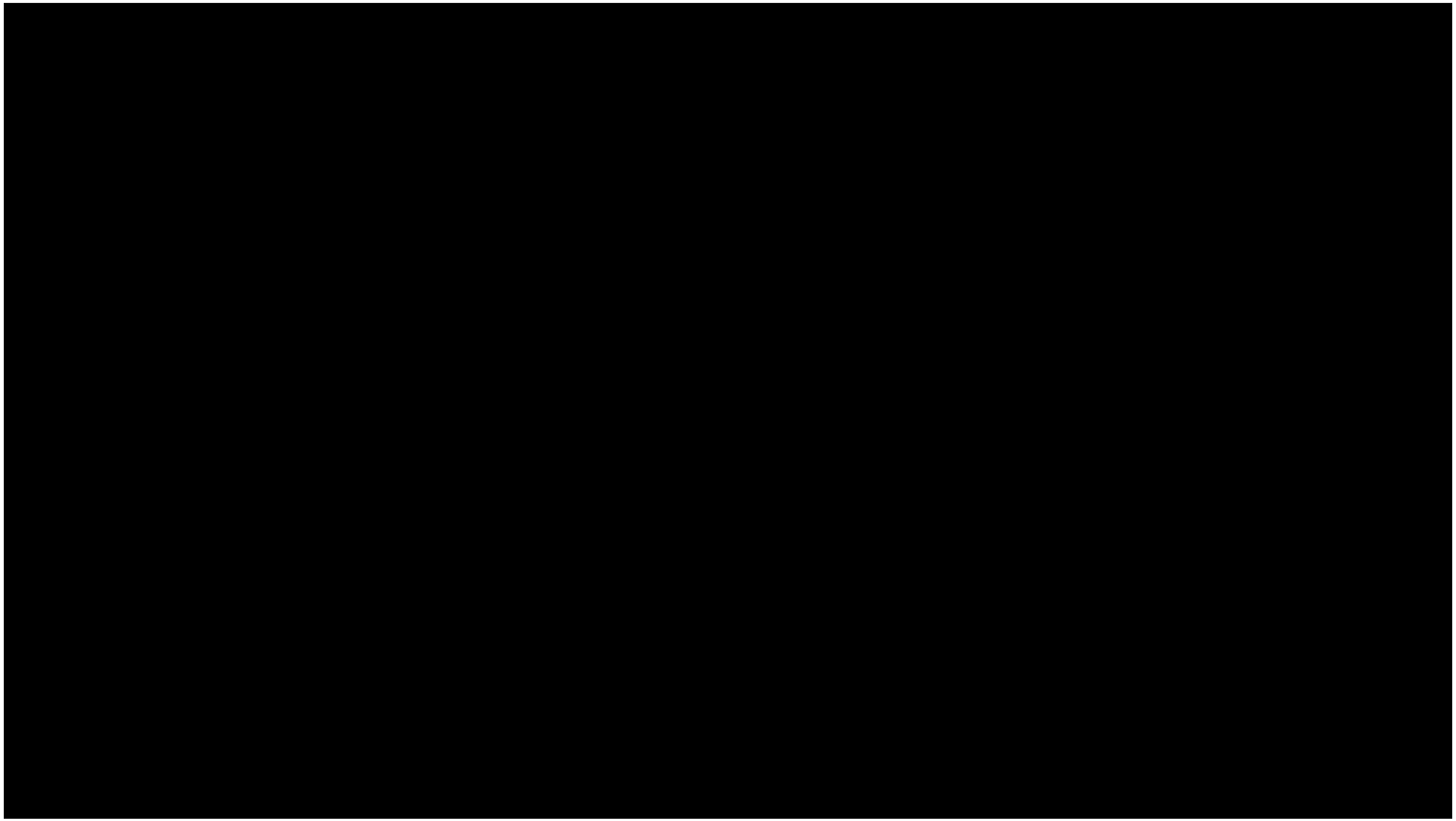
AlphaGo, 2016

Google DeepMind's AlphaGo vs. Lee Sedol, March 2016

- Possible board positions of Go: 10^{170}
  - cf. Chess: 10^{47}
- Monte Carlo tree search
- Deep neural networks:
  - Value network
  - Policy network
- Reinforcement learning
- Trained from
  - 30 million human moves
  - Playing against itself
- 1,202 CPUs, 176 GPUs

Game	Date	Black	White	Result	Moves
1	9 March 2016	Lee Sedol	**AlphaGo**	Lee Sedol resigned	186 Game 1
2	10 March 2016	**AlphaGo**	Lee Sedol	Lee Sedol resigned	211 Game 2
3	12 March 2016	Lee Sedol	**AlphaGo**	Lee Sedol resigned	176 Game 3
4	13 March 2016	AlphaGo	**Lee Sedol**	AlphaGo resigned	180 Game 4
5	15 March 2016	Lee Sedol[note 1]	**AlphaGo**	Lee Sedol resigned	280 Game 5
Result: **AlphaGo 4 – 1 Lee Sedol**					

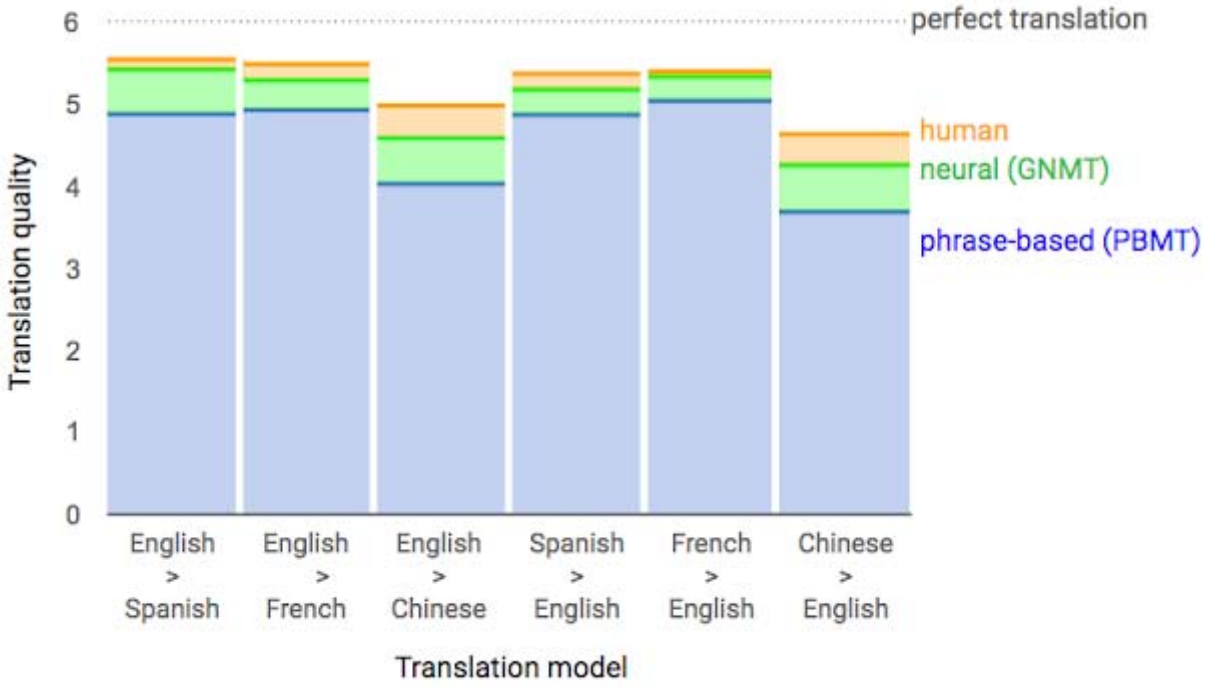
Robotics (OpenAI)



Deep Learning: Some Recent Applications

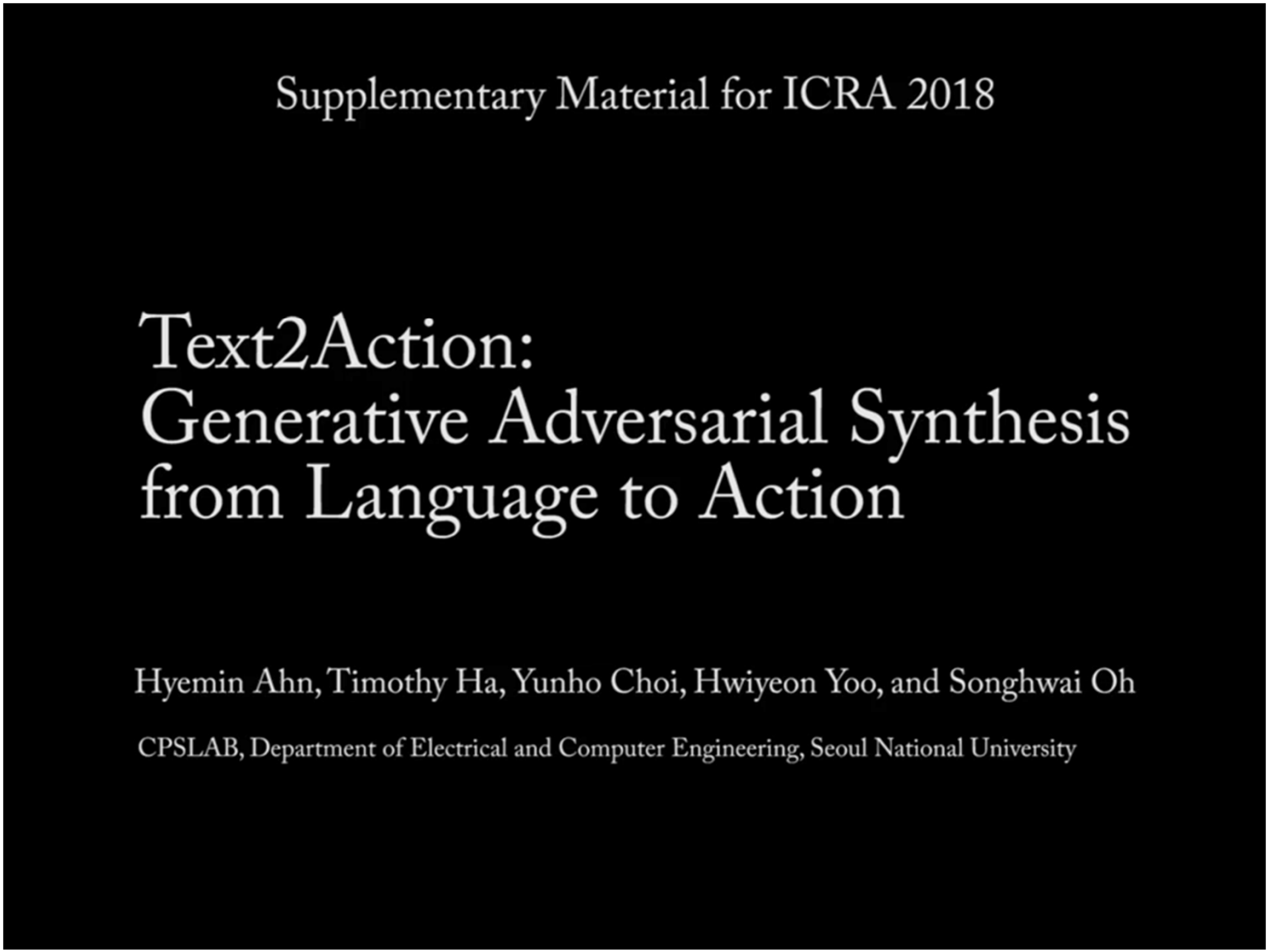
Language Translation

- Google Neural Machine Translation (GNMT) System



Source: https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

Language to Action (SNU)



Synthesis

- Speech (WaveNet, Google DeepMind)
- Images

[1]

[2]

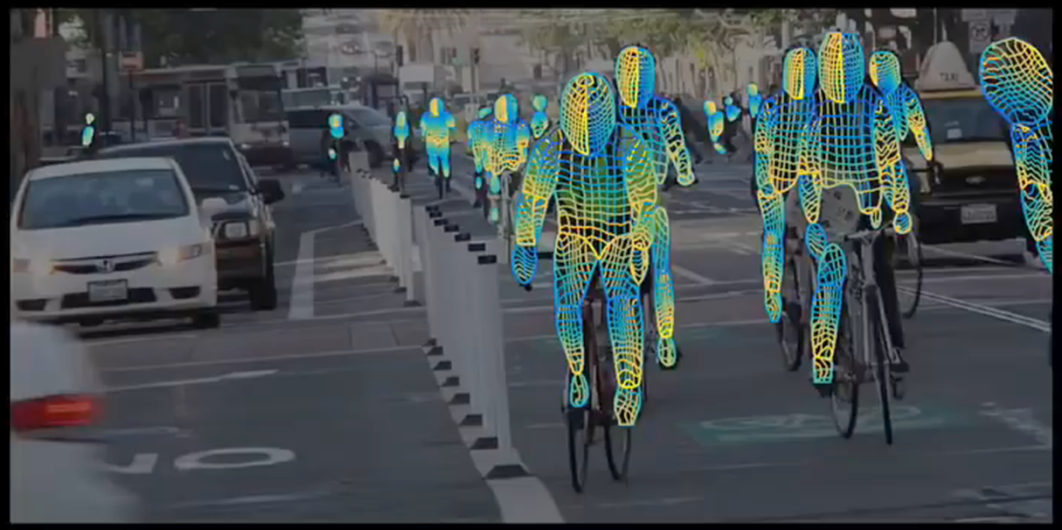
[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros , "**Image-to-Image Translation with Conditional Adversarial Nets**," CVPR 2017.
[2] Karras, T., Aila, T., Laine, S., & Lehtinen, J. **Progressive growing of GANs for improved quality, stability, and variation.** ICLR 2018.

Video Synthesis (University of Washington)

Pose Estimation (DensePose, Facebook)

DensePose:

Dense Human Pose Estimation In The Wild



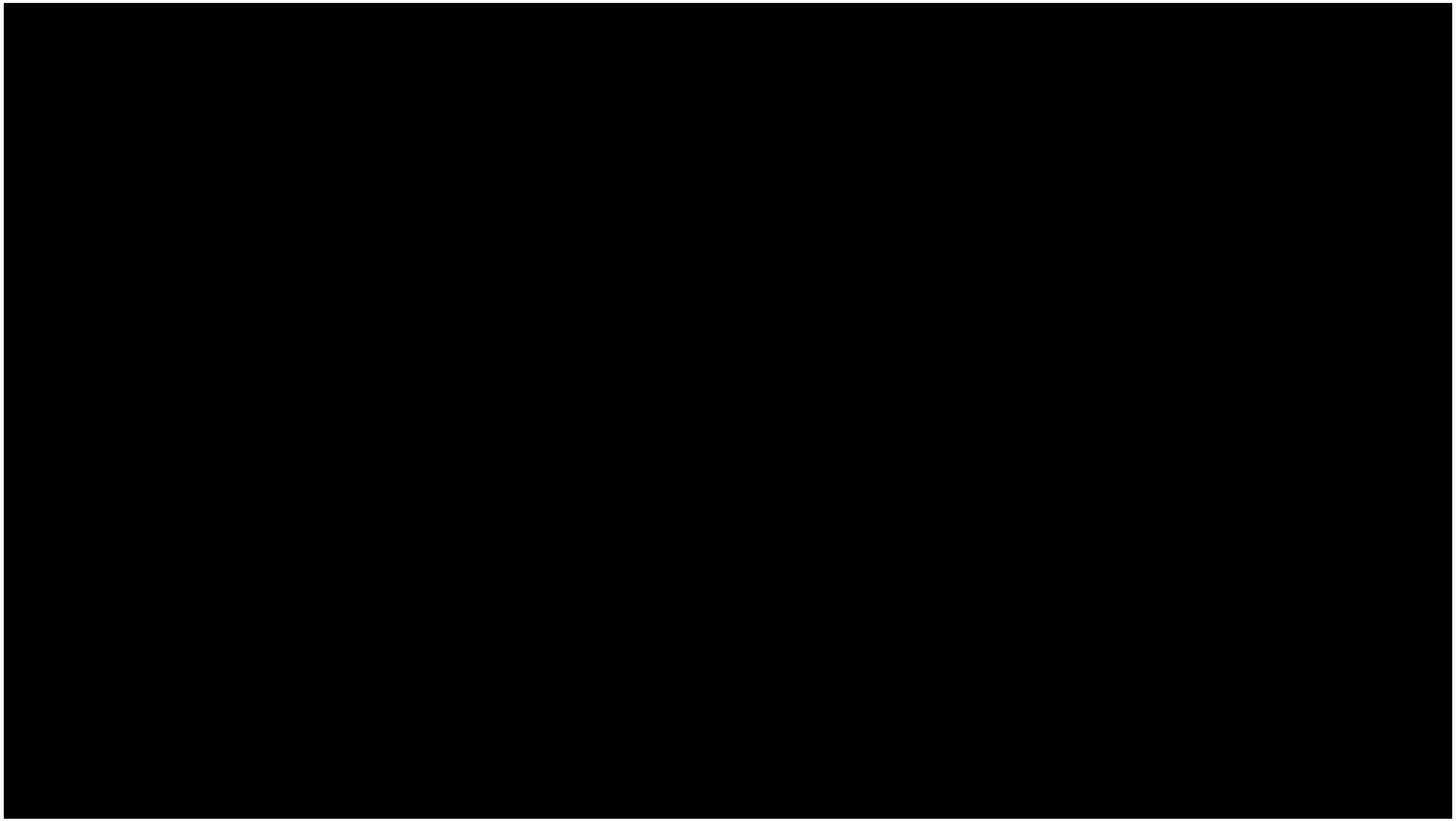
Rıza Alp Güler *
INRIA, CentraleSupélec

Natalia Neverova
Facebook AI Research

Iasonas Kokkinos
Facebook AI Research

* Rıza Alp Güler was with Facebook AI Research during this work.

Autonomous Driving (Wayve)



Wrap Up

- Linear classification
- Neural networks
  - Backpropagation
- Deep learning
- Deep reinforcement learning