EECE 571F: Advanced Topics in Deep Learning

Lecture 1: Introduction to Deep Learning

Renjie Liao

University of British Columbia

Winter, Term 1, 2024

Course Information

- Course website: https://lrjconan.github.io/UBC-EECE571F-DL-Structures/
- Cutting-edge topics in deep learning with structures (not an introduction!!!)
- Assumes basic knowledge about machine learning, deep learning
  - ➢ View relevant textbooks/courses on the website
- Assumes basic knowledge about linear algebra, calculus, probability
- Assumes proficiency in deep learning libraries: PyTorch, JAX, Tensorflow
  - ➢ Self-learning through online tutorials, e.g. https://pytorch.org/tutorials/

Course Information

- Two sections: Mon. & Wed. 13:30 to 3:00pm,

  Room 4018, Orchard Commons (ORCH)

  Office hour: 1:30pm to 2:30pm, Tue., KAIS 3047 (Ohm)

- TA: Yuanpei Gao (yuanpeig@student.ubc.ca)

Course Information

- Two sections: Mon. & Wed. 13:30 to 3:00pm,

  Room 4018, Orchard Commons (ORCH)

  Office hour: 1:30pm to 2:30pm, Tue., KAIS 3047 (Ohm)

- TA: Yuanpei Gao (yuanpeig@student.ubc.ca)

- All lectures will be delivered in person without recording unless notified otherwise

- Use Piazza for discussion & questions (actively answering others' questions get you bonuses) and Canvas for submitting reports

  https://piazza.com/ubc.ca/winterterm12024/eece571f

Course Information

- Expectation & Grading (More info on the website)
  - [15%] One paper reading report, due Sep. 27
  - [15%] Project proposal, due Oct. 11
  - [15%] Project presentations, around last two weeks
  - [15%] Peer-review report of project presentations, due Dec. 6
  - [40%] Project report and code, due Dec. 11
- You are encouraged to team up (up to 4 members) for projects

Course Information

- How to get free GPUs for your course project?

  1. **Google Colab**: https://research.google.com/colaboratory/

     Google Colab is a web-based iPython Notebook service that has access to a free Nvidia K80 GPU per Google account.

  2. **Google Compute Engine**: https://cloud.google.com/compute

     Google Compute Engine provides virtual machines with GPUs running in Google's data center. You get $300 free credit when you sign up.

- Strategy of using GPUs

  1. Debug models on small datasets (subsets) using CPUs or low-end GPUs until they work

  2. Launch batch jobs on high-end GPUs to tune hyperparameters

Course Scope

- Brief Intro to Deep Learning

Course Scope

- Brief Intro to Deep Learning

- Geometric Deep Learning

  - Deep Learning Models for Sets and Sequences: Deep Sets & Transformers
  - Deep Learning Models for Graphs: Message Passing & Graph Convolution GNNs
  - Group Equivariant Deep Learning

Course Scope

- Brief Intro to Deep Learning

- Geometric Deep Learning

  - Deep Learning Models for Sets and Sequences: Deep Sets & Transformers
  - Deep Learning Models for Graphs: Message Passing & Graph Convolution GNNs
  - Group Equivariant Deep Learning

- Probabilistic Deep Learning

  - Auto-regressive models, Large Language Models (LLMs)
  - Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs)
  - Flow models
  - Diffusion/Score based models

Outline

- Brief Introduction & History & Application
- Basic Deep Learning Models
  - Multi-Layer Perceptron (MLP)
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Objective Function
- Learning Algorithm: Back-propagation
- Limitations

Outline

- **Brief Introduction & History & Application**
- Basic Deep Learning Models
    - Multi-Layer Perceptron (MLP)
    - Convolutional Neural Network (CNN)
    - Recurrent Neural Network (RNN)
- Objective Function
- Learning Algorithm: Back-propagation
- Limitations

What is Deep Learning?

- Definition from Wikipedia:

  Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning.

- Key Aspects:

  Data: Large (supervised) datasets, e.g., ImageNet (14 million+ annotated images)

  Model: Deep (i.e., many layers) neural networks, e.g., ResNet-152

  Learning algorithm: Back-propagation (BP), i.e., stochastic gradient descent (SGD)

Brief History of Deep Learning (Connectionism)

- Artificial Neurons (McCulloch and Pitts 1943)
- Hebbian Rule: Cells that fire together wire together (Donald Hebb 1949)
- Perceptron (Frank Rosenblatt 1958)
- Discovery of orientation selectivity and columnar organization in the visual cortex (Hubel and Wiesel, 1959)
- Neocognitron (first Convolutional Neural Network, Fukushima 1979)
- Hopfield networks (Hopfield 1982)
- Boltzmann machines (Hinton, Sejnowski 1983)
- Backpropagation (Linnainmaa 1970, Werbos 1974, Rumelhart, Hinton, Williams 1986)
- First application of BP to Neocognitron-like CNNs (LeCun et al. 1989)
- Long-short term memory (Hochreiter, Schmidhuber 1997)

Brief History of Deep Learning (Connectionism)

- Deep belief networks (DBN) (Hinton et al., 2006)
- Breakthrough in speech recognition (Dahl et al. 2010)
- Breakthrough in computer vision: AlexNet (Krizhevsky et al. 2012), ResNet (He et al. 2016)
- Breakthrough in games: DQN (Minh, 2015), AlphaGO (2016)
- Breakthrough in natural language processing: Seq2seq (Sutskever et al. 2014), Transformers (Vaswani et al. 2017), GPT-3 (Brown et al. 2020)
- Breakthrough in protein structure prediction: AlphaFold (2020)

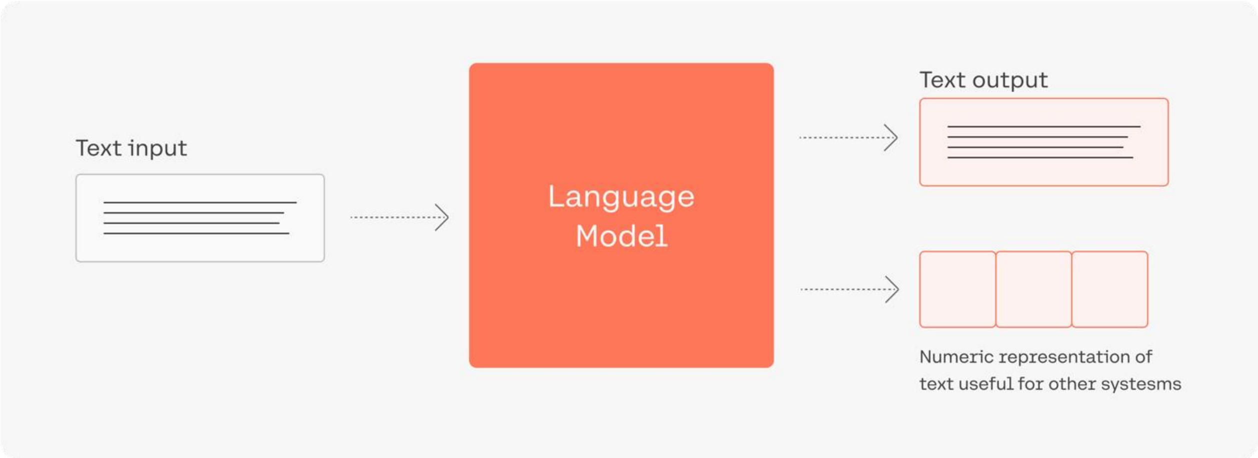
……

The future depends on some graduate student who is deeply suspicious of everything I have said.

- Geoffrey Hinton

Applications of Deep Learning

Large Language Models (LLMs)



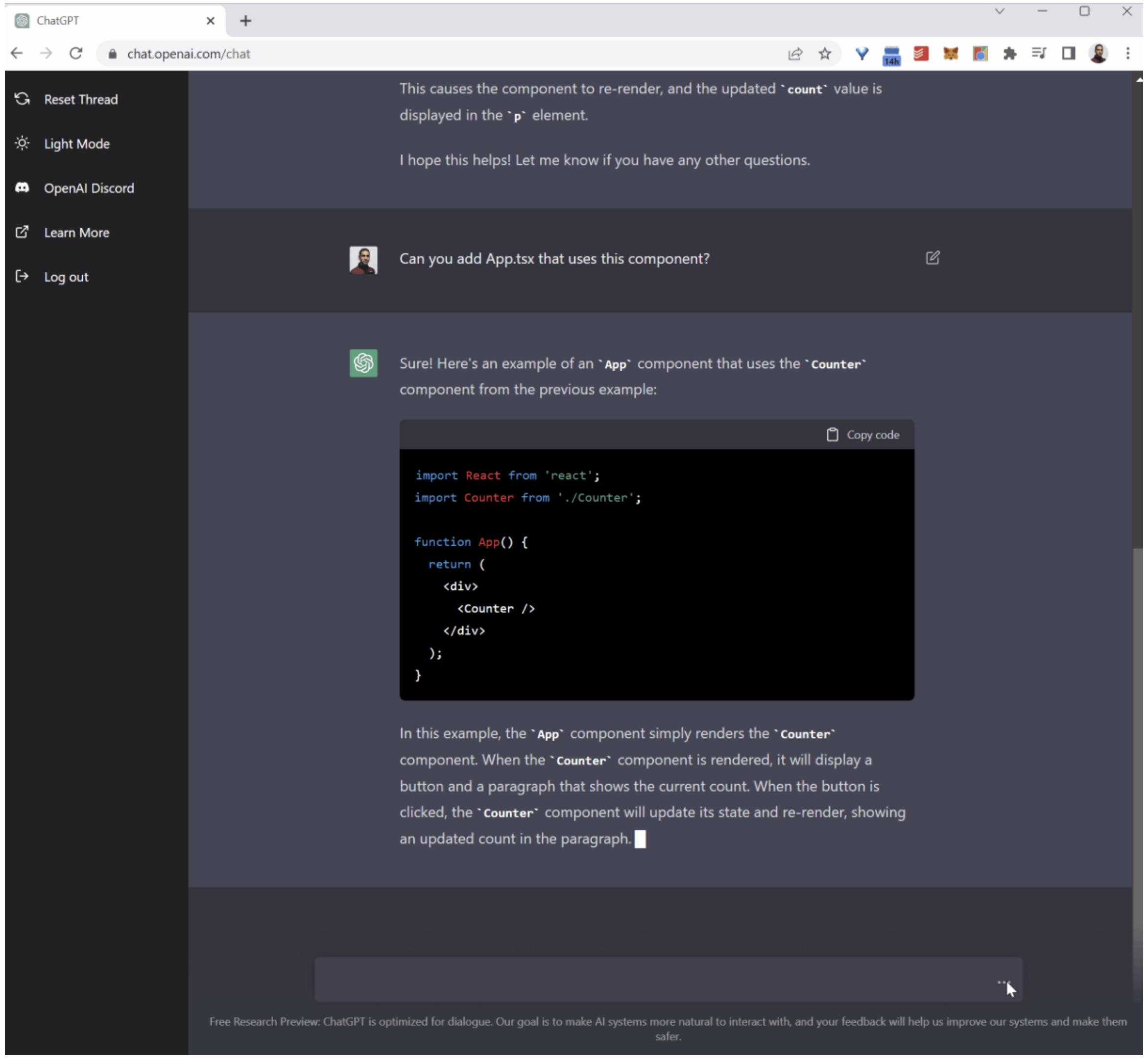


Image Credit: https://docs.cohere.com/docs/introduction-to-large-language-models
https://medium.com/geekculture/6-chatgpt-mind-blowing-extensions-to-use-it-anywhere-db6638640ec7

Applications of Deep Learning

Text/Program Generation

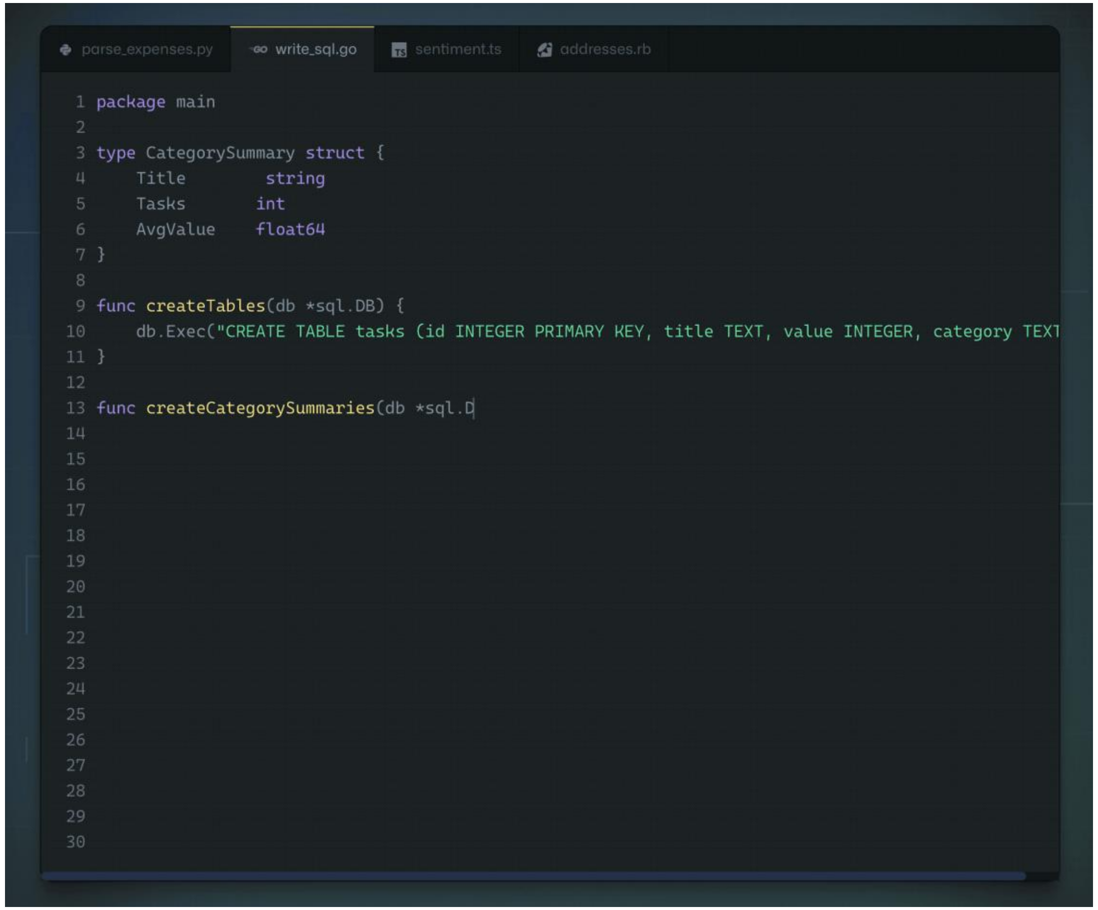


Image Credit: https://techcrunch.com/2020/11/12/othersideai-raises-2-6m-to-let-gpt-3-write-your-emails-for-you/
https://techcrunch.com/2021/06/29/github-previews-new-ai-tool-that-makes-coding-suggestions/

Applications of Deep Learning

Speech Recognition, Personal Assistants

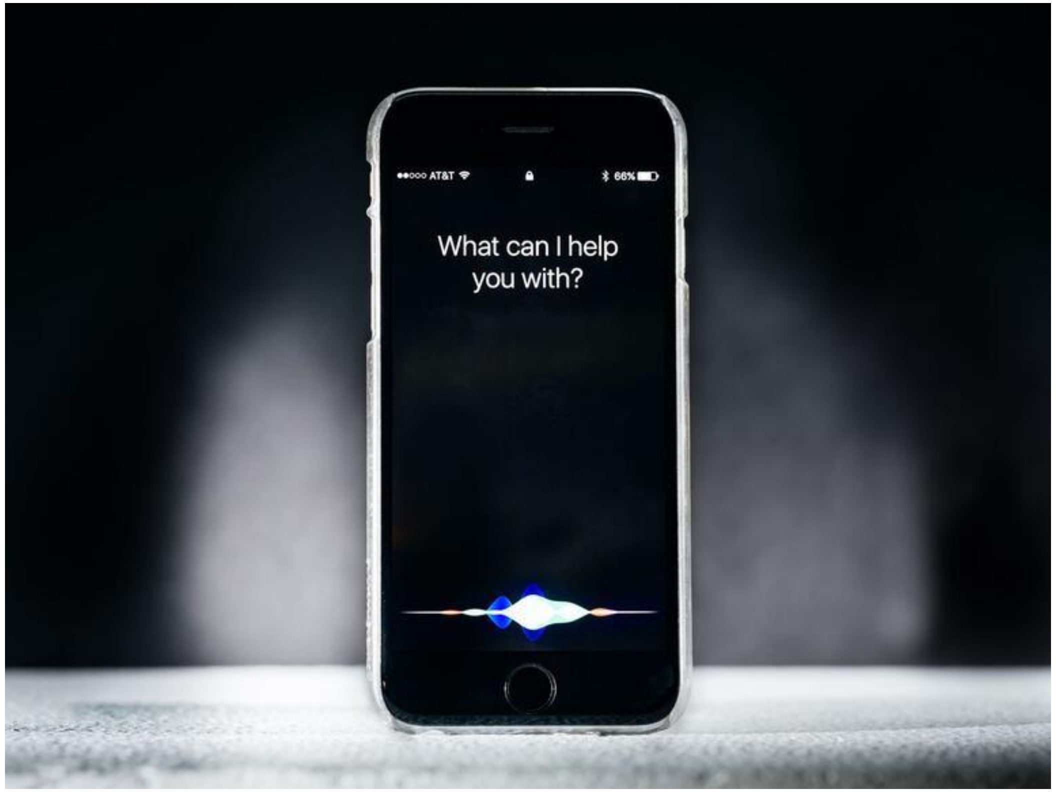
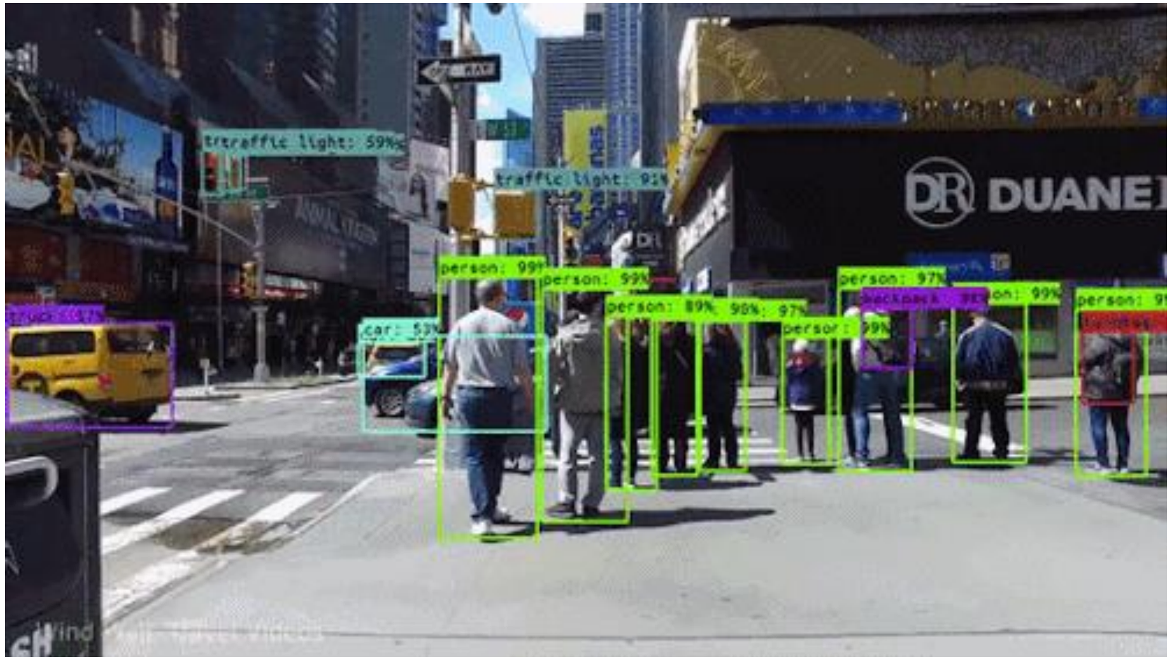


Image Credit: https://www.techrepublic.com/article/apples-siri-the-smart-persons-guide/
https://www.pcmag.com/news/amazon-echo-vs-google-home-which-smart-speaker-is-best

Applications of Deep Learning

Computer Vision/Graphics, e.g., Object detection, Rendering



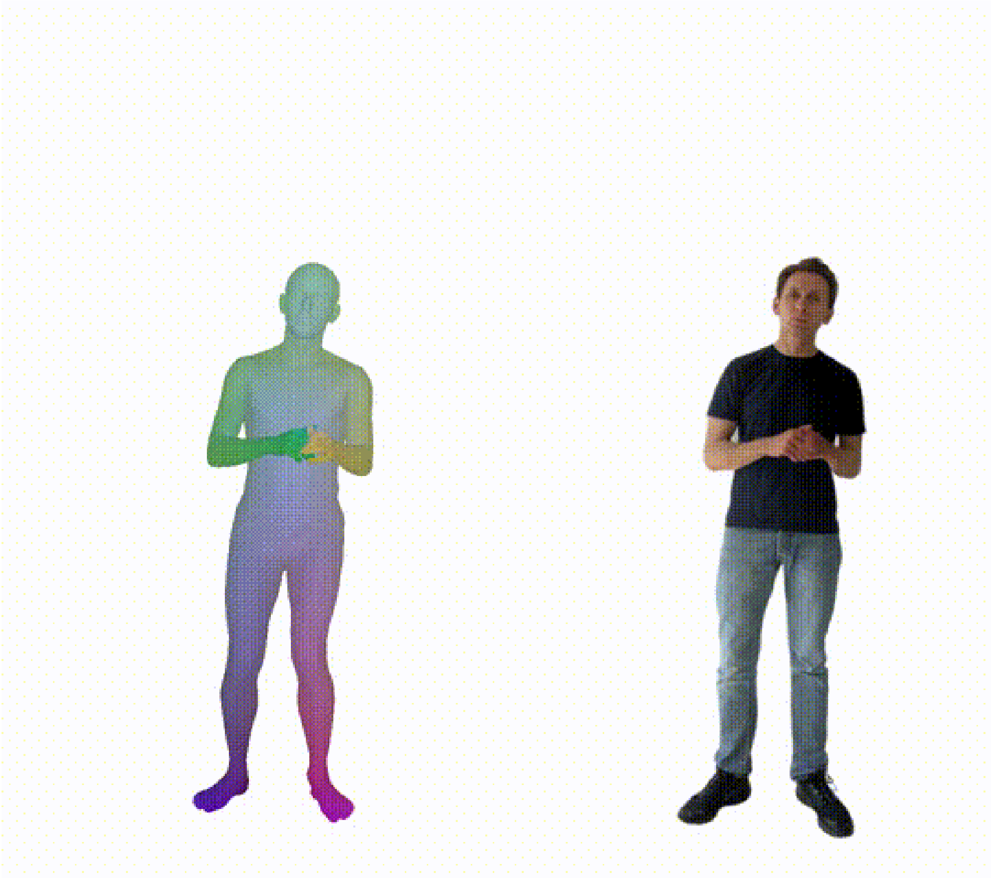


Image Credit: https://github.com/sergeyprokudin/smplpix
https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

Applications of Deep Learning

Virtual/Augmented Reality

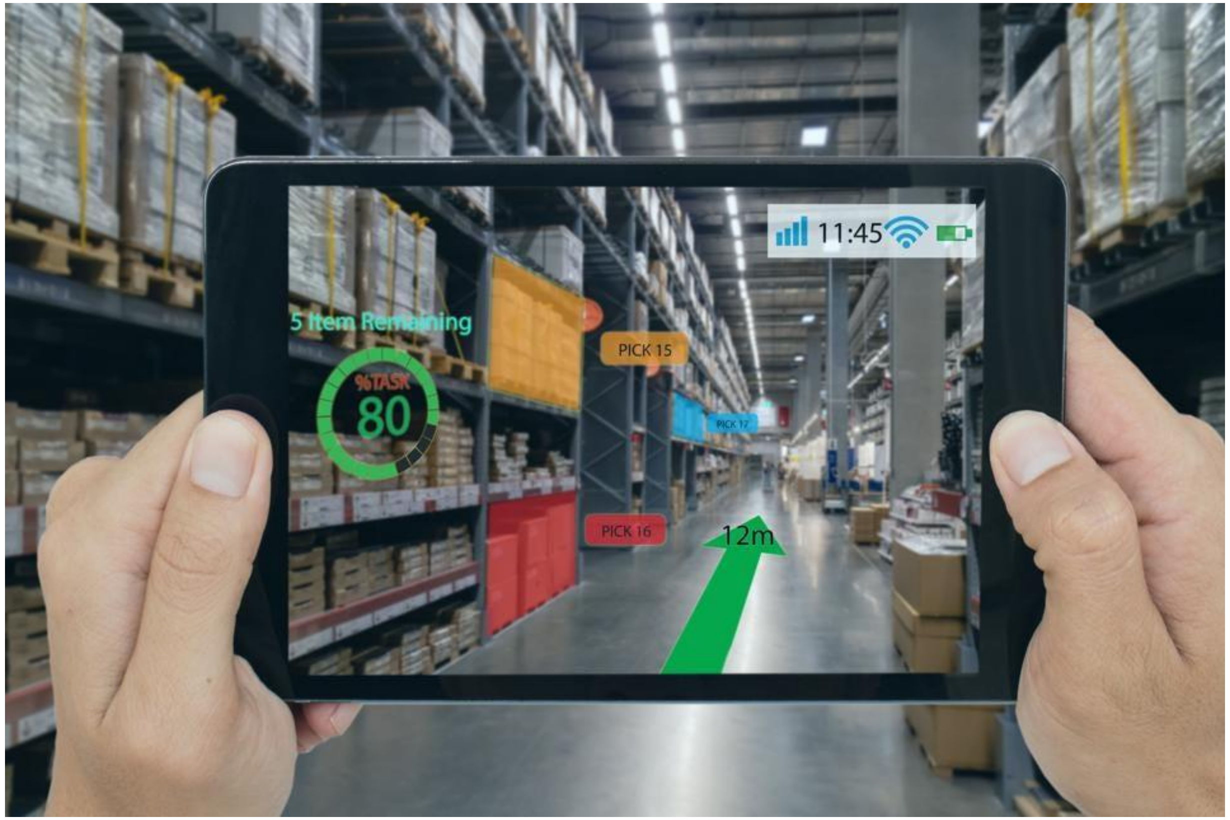


Image Credit: https://www.businessinsider.com/scott-galloway-metaverse-future-not-facebook-2021-8
https://www.forbes.com/sites/theyec/2019/02/06/augmented-reality-in-business-how-ar-may-change-the-way-we-work/?sh=36011cd851e5

Applications of Deep Learning

Robotics, Autonomous Driving

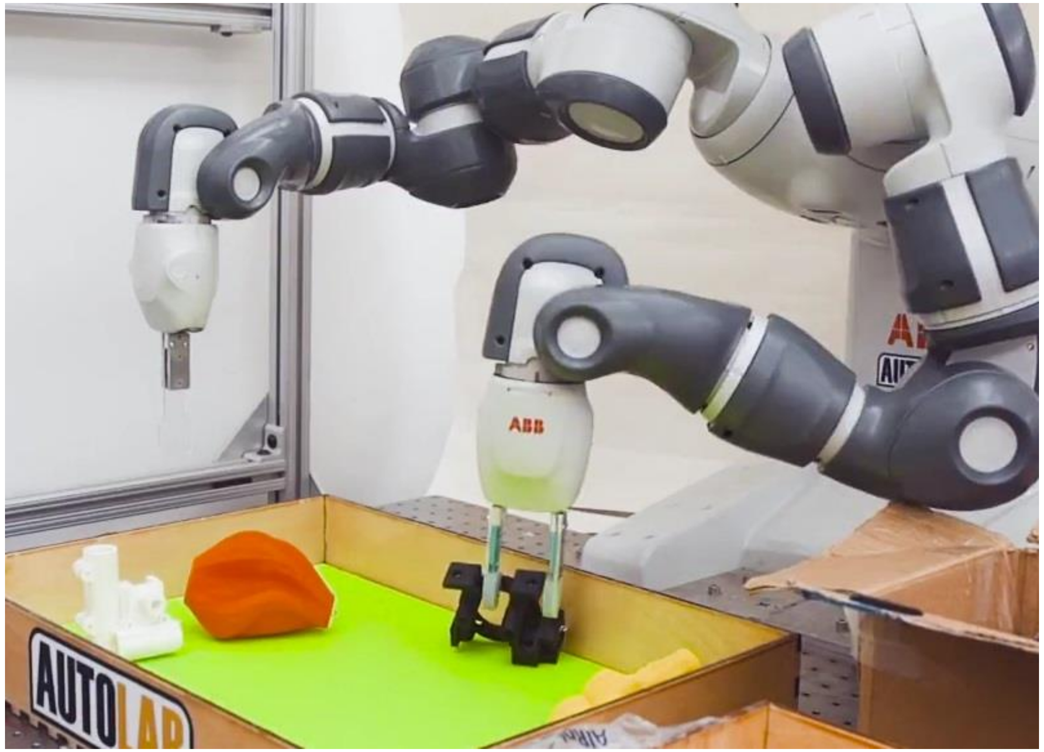


Image Credit: https://techcrunch.com/2017/05/26/this-robot-arms-ai-thinks-like-we-do-about-how-to-grab-something/
https://techcrunch.com/2018/10/30/waymo-takes-the-wheel-self-driving-cars-go-fully-driverless-on-california-roads/

Applications of Deep Learning

Protein structure prediction, Drug discovery

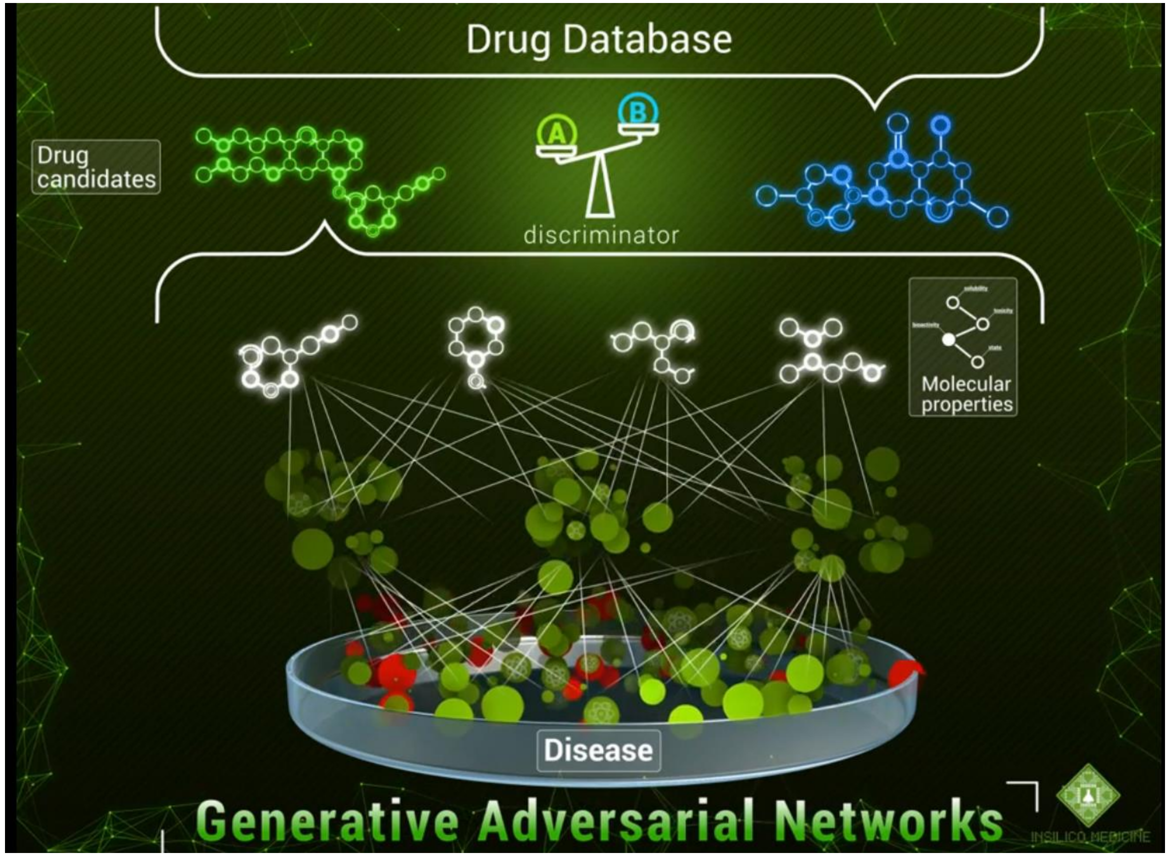
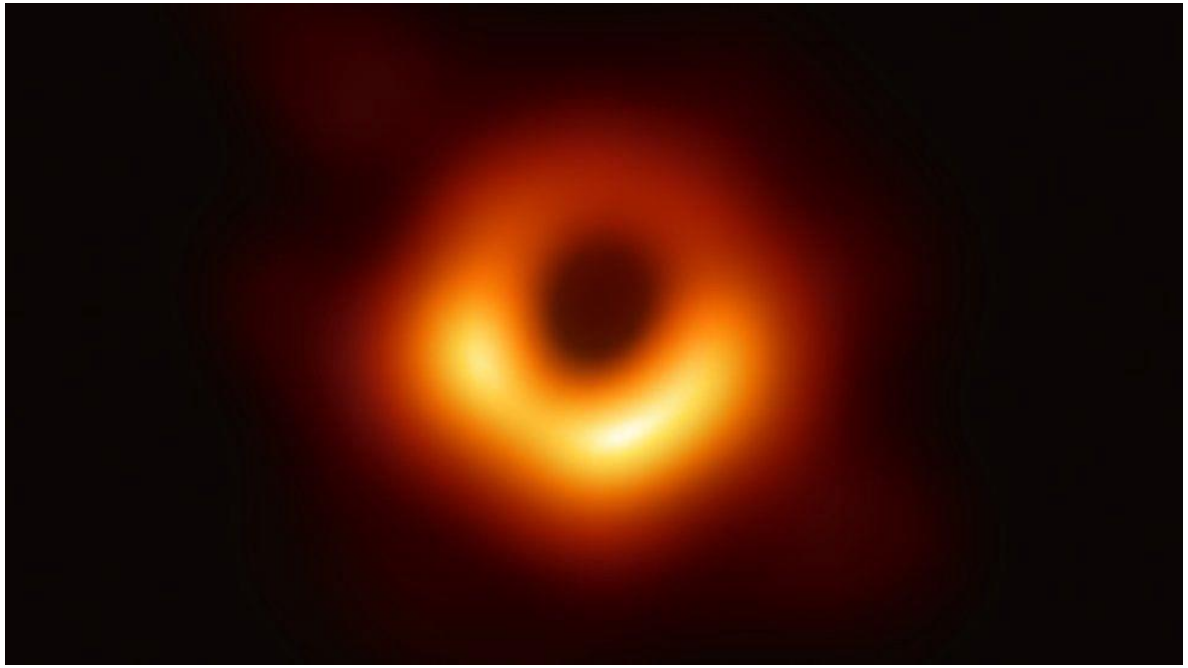


Image Credit: https://www.nature.com/articles/d41586-020-03348-4
https://medium.com/neuromation-blog/creating-molecules-from-scratch-i-drug-discovery-with-generative-adversarial-networks-9d42cc496fc6

Applications of Deep Learning

Black Holes, Physics Simulation



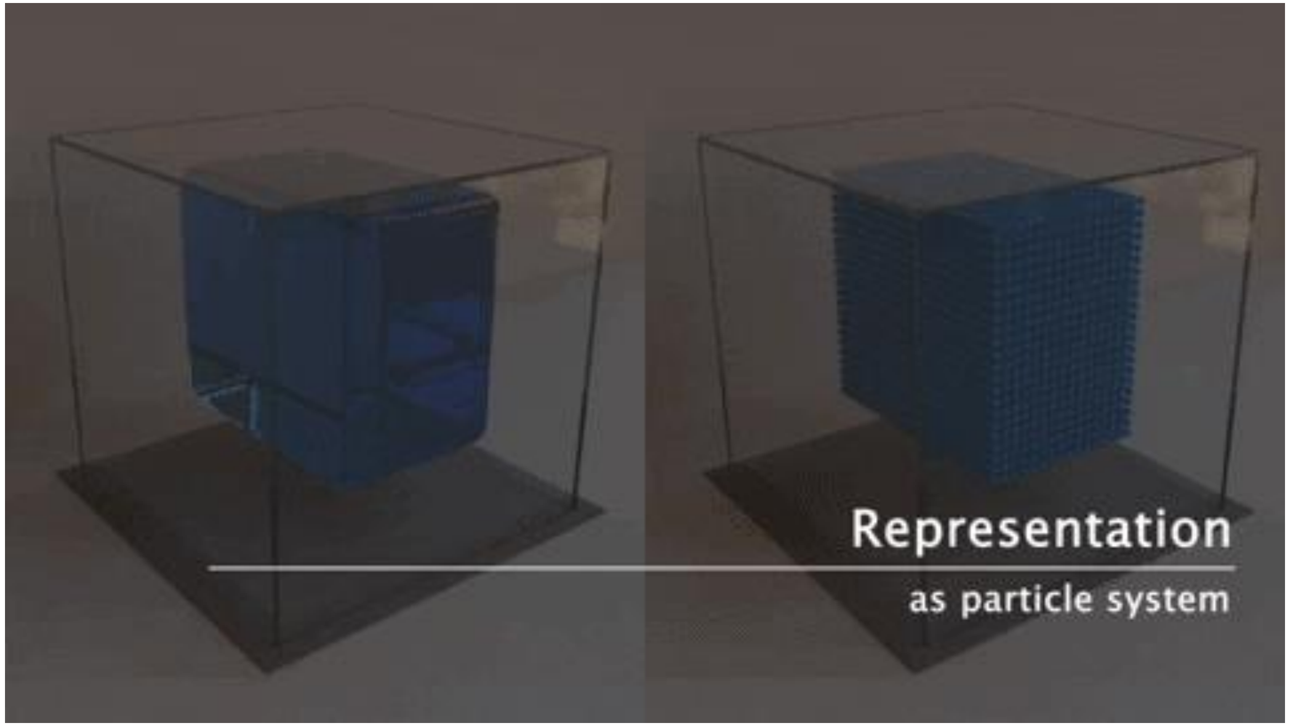


Image Credit: https://medium.com/analytics-vidhya/when-neural-networks-saw-the-first-image-of-black-hole-3205e28b6578
https://meerasridhar23.medium.com/lets-10x-physics-simulations-38ad6fc5cae5

Outline

- Brief Introduction & History & Application
- **Basic Deep Learning Models**
    - Multi-Layer Perceptron (MLP)
    - Convolutional Neural Network (CNN)
    - Recurrent Neural Network (RNN)
- Objective Function
- Learning Algorithm: Back-propagation
- Limitations

Basic Deep Learning Models

Multi-Layer Perceptron (MLP)

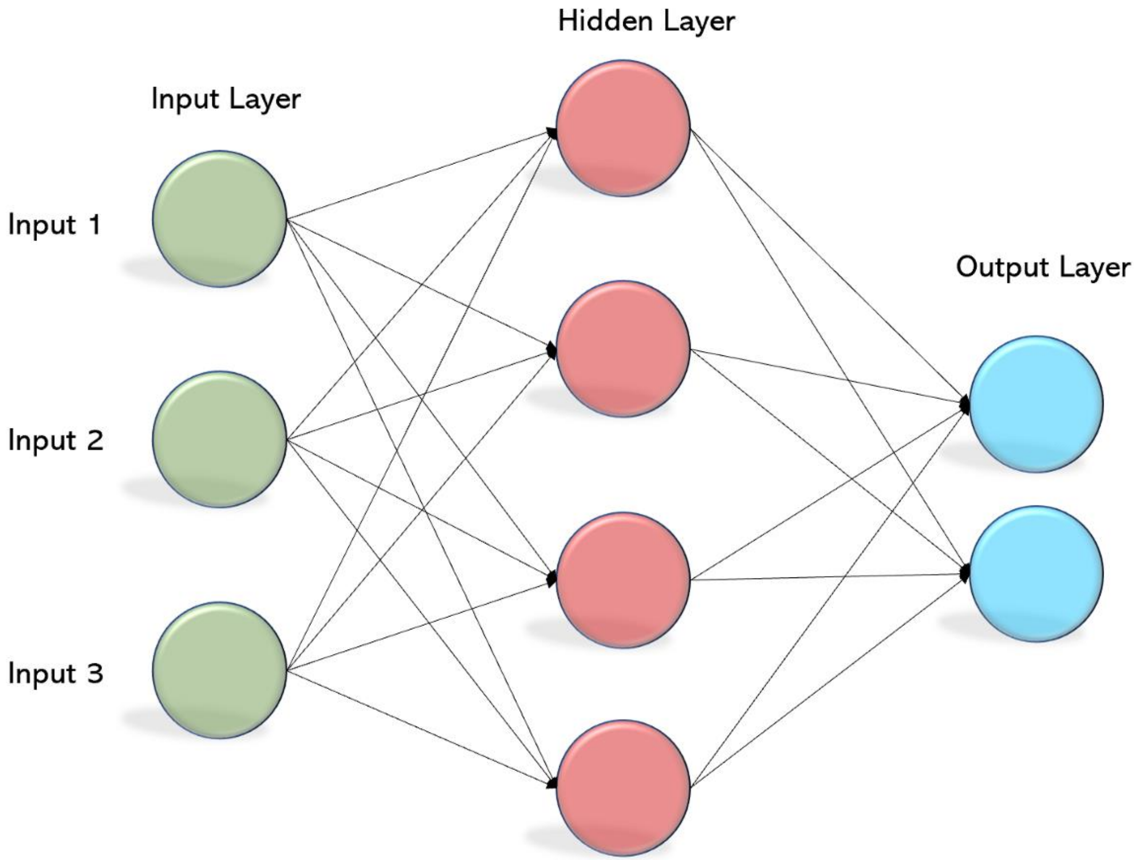


Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Basic Deep Learning Models

Multi-Layer Perceptron (MLP)

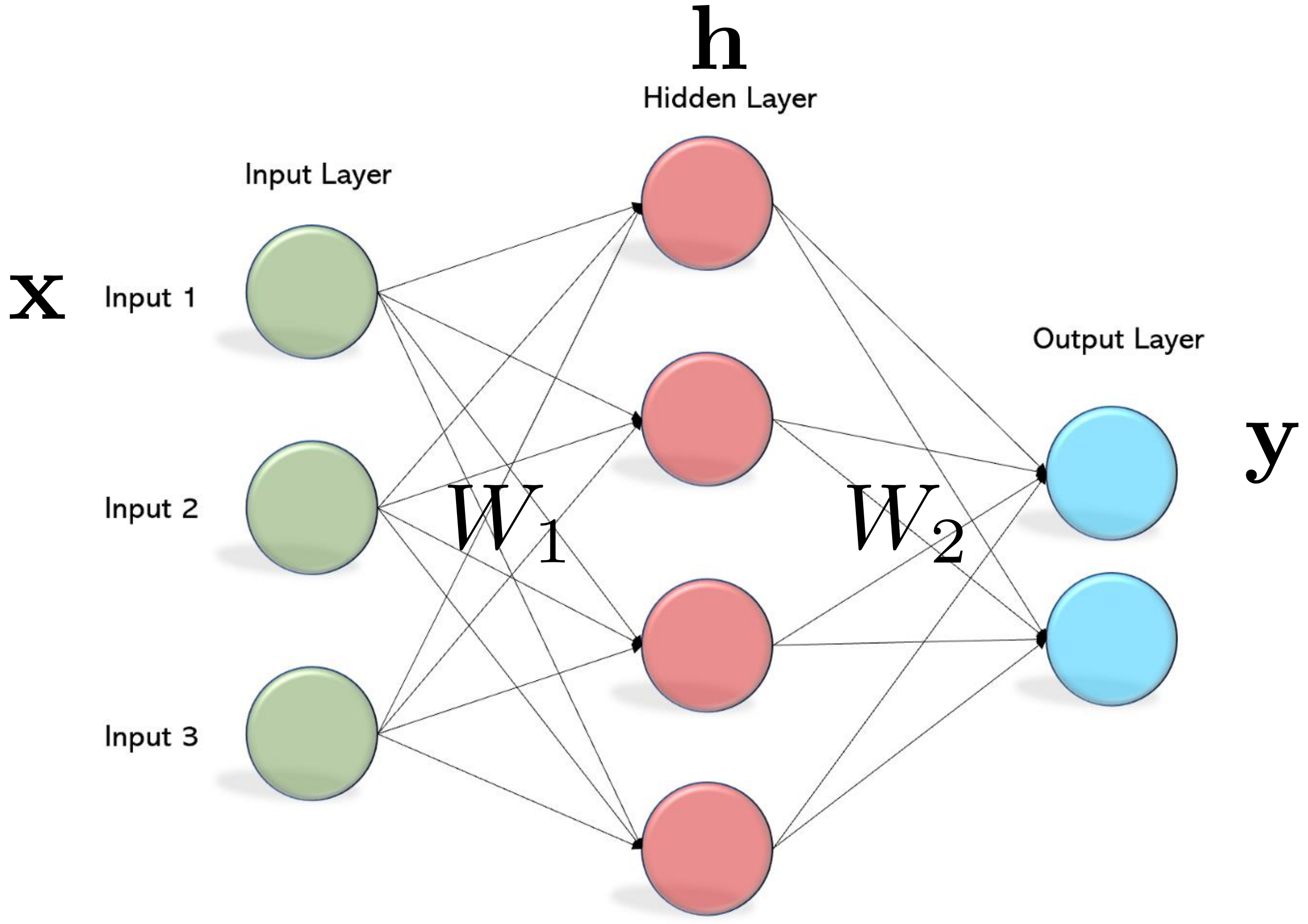
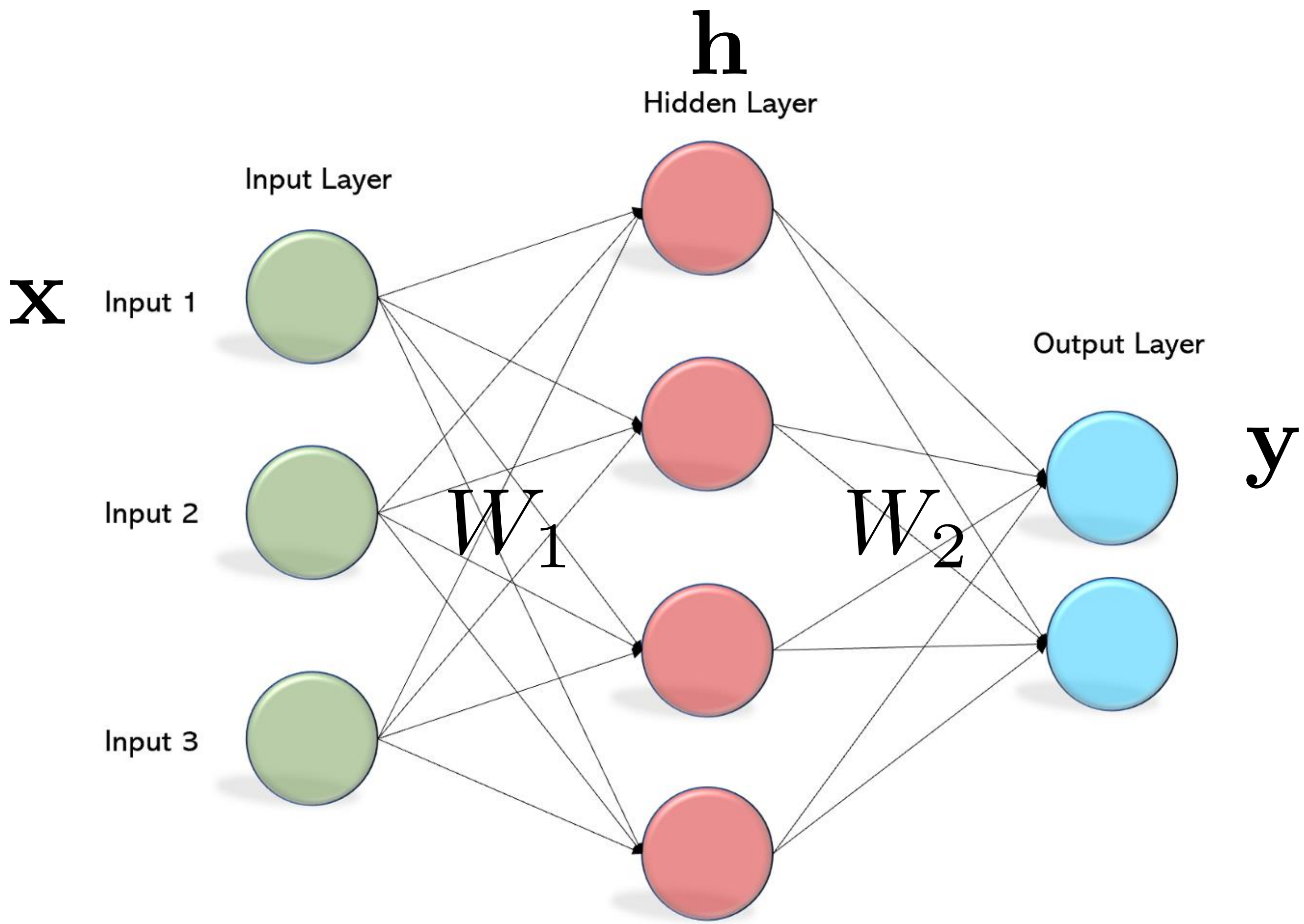


Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Basic Deep Learning Models

Multi-Layer Perceptron (MLP)



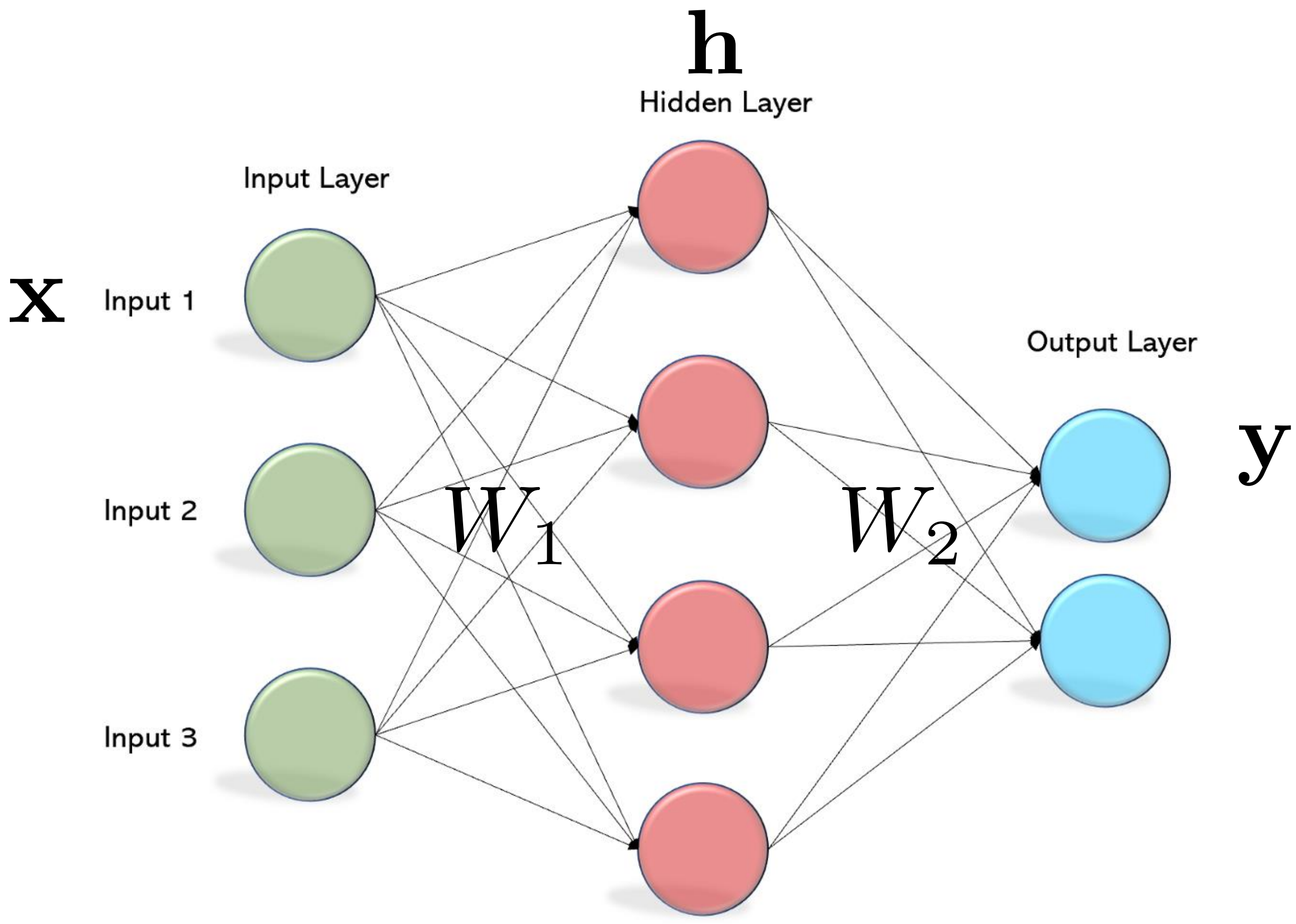
$$\mathbf{h} = \sigma(W_1 \mathbf{x})$$

$$\mathbf{y} = W_2 \mathbf{h}$$

Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Basic Deep Learning Models

Multi-Layer Perceptron (MLP)



$$\mathbf{h} = \sigma(W_1 \mathbf{x})$$

$$\mathbf{y} = W_2 \mathbf{h}$$

ReLU: $\sigma(\mathbf{h}) = \max(\mathbf{h}, 0)$

Sigmoid: $\sigma(\mathbf{h}) = \frac{1}{1 + \exp(-\mathbf{h})}$

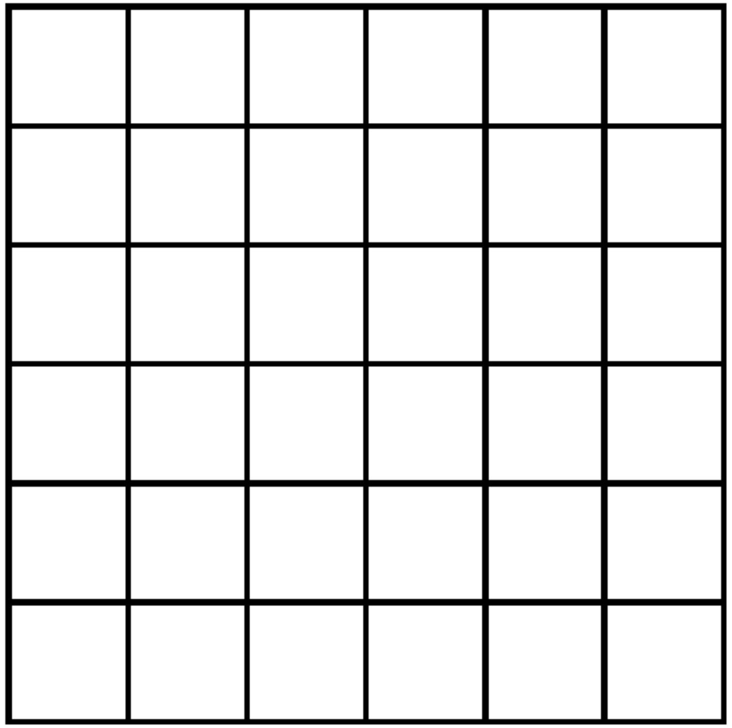
Tanh, Softplus, ELU, …

Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

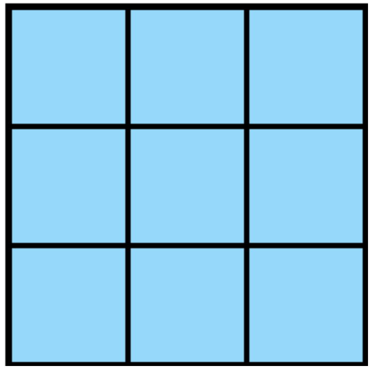
Basic Deep Learning Models

Convolutional Neural Network (CNN)

Convolution (Discrete)



Image



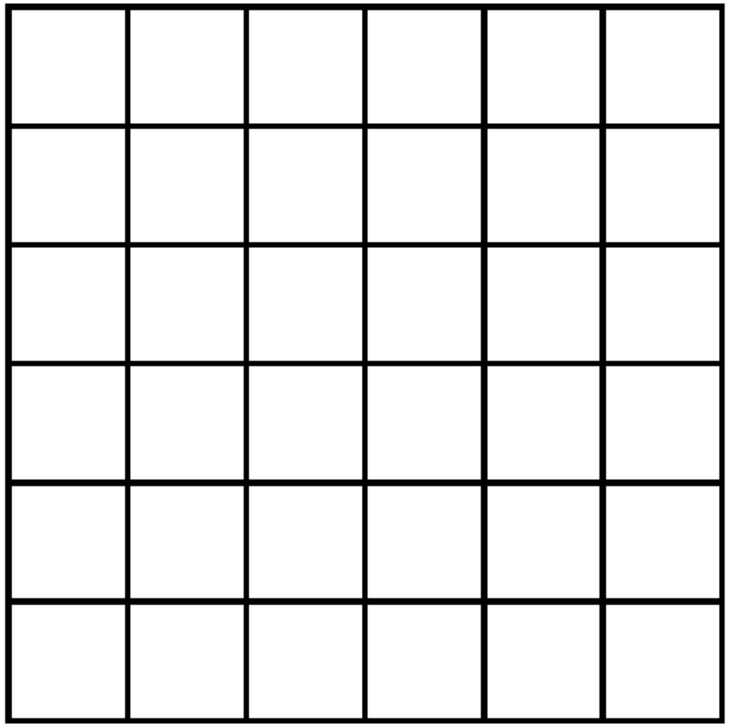
Convolutional Filter

Basic Deep Learning Models

Convolutional Neural Network (CNN)

Convolution (Discrete)

$$\mathbf{y}_{i,j} = \sum_{m=1}^{K} \sum_{n=1}^{K} W_{m,n} \mathbf{x}_{i+m-\lceil K/2 \rceil, j+n-\lceil K/2 \rceil}$$

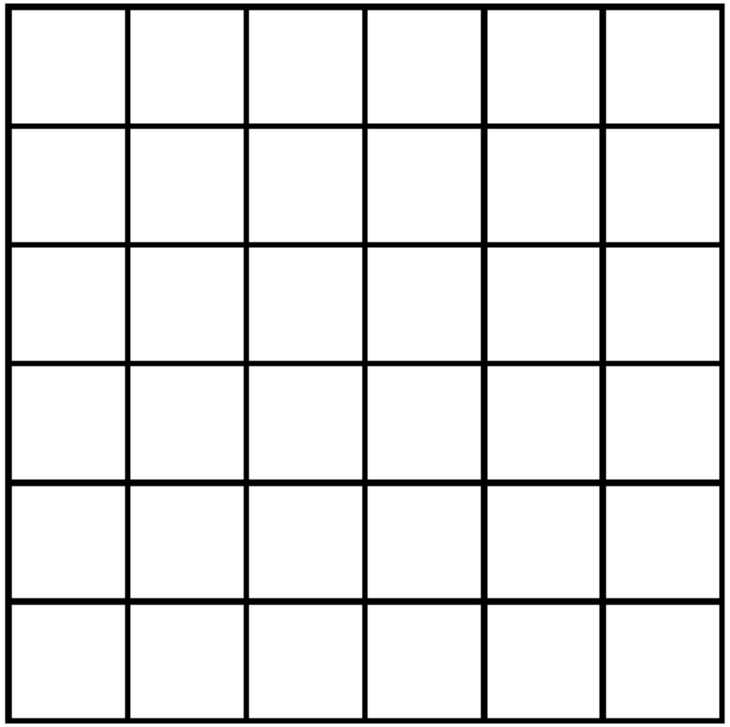


Basic Deep Learning Models

Convolutional Neural Network (CNN)

Convolution (Discrete)

$$\mathbf{y}_{i,j} = \sum_{m=1}^{K} \sum_{n=1}^{K} W_{m,n} \mathbf{x}_{i+m-\lceil K/2 \rceil, j+n-\lceil K/2 \rceil}$$



Convolution ⬄ Matrix Multiplication

Matrix Multiplication View I

1D Convolution (Discrete) ⇔ Matrix Multiplication

Filter => Toeplitz matrix (diagonal-constant)

Matrix Multiplication View I

1D Convolution (Discrete) ⬄ Matrix Multiplication

Filter => Toeplitz matrix (diagonal-constant)

$$y = h * x = \begin{bmatrix} h_{\lfloor m/2 \rfloor +1} & h_{\lfloor m/2 \rfloor +2} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ h_{\lfloor m/2 \rfloor} & h_{\lfloor m/2 \rfloor +1} & \cdots & h_{m-1} & h_m & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & 0 & \cdots & 0 \\ 0 & h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & h_1 & h_2 & \cdots & h_{\lfloor m/2 \rfloor +2} \\ 0 & 0 & \cdots & \cdots & & \cdots\cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor +1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

Matrix Multiplication View I

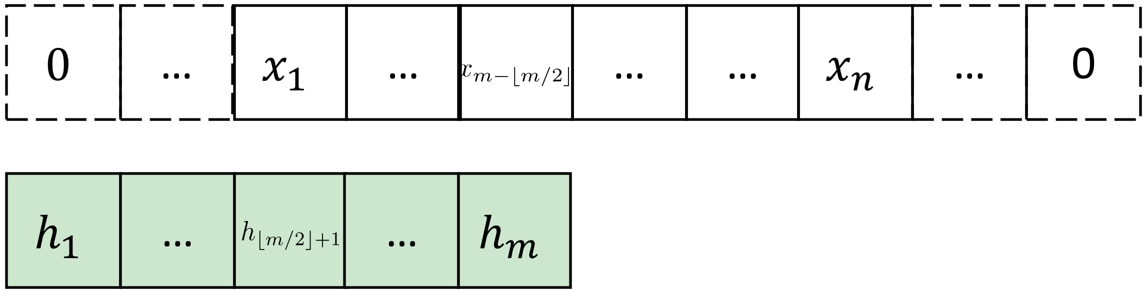
1D Convolution (Discrete) ⇔ Matrix Multiplication

Filter => Toeplitz matrix (diagonal-constant)

$$y = h * x = \begin{bmatrix} h_{\lfloor m/2 \rfloor+1} & h_{\lfloor m/2 \rfloor+2} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ h_{\lfloor m/2 \rfloor} & h_{\lfloor m/2 \rfloor+1} & \cdots & h_{m-1} & h_m & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & 0 & \cdots & 0 \\ 0 & h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & h_1 & h_2 & \cdots & h_{\lfloor m/2 \rfloor+2} \\ 0 & 0 & \cdots & \cdots & & \cdots\cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor+1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

Input x

Filter h

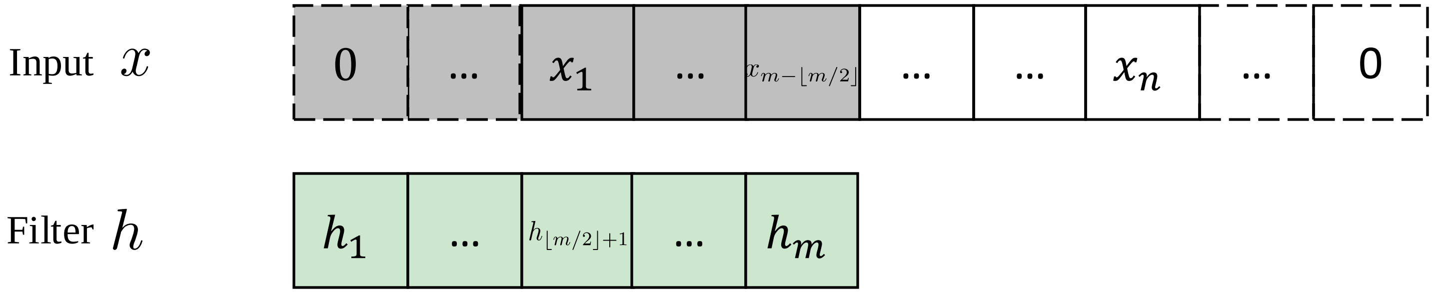


Matrix Multiplication View I

1D Convolution (Discrete) ⇔ Matrix Multiplication

Filter => Toeplitz matrix (diagonal-constant)

$$
y = h * x = \begin{bmatrix}
h_{\lfloor m/2 \rfloor+1} & h_{\lfloor m/2 \rfloor+2} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\
h_{\lfloor m/2 \rfloor} & h_{\lfloor m/2 \rfloor+1} & \cdots & h_{m-1} & h_m & 0 & \cdots & \cdots & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & 0 & \cdots & 0 \\
0 & h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & \cdots & \cdots & 0 & h_1 & h_2 & \cdots & h_{\lfloor m/2 \rfloor+2} \\
0 & 0 & \cdots & \cdots & & \cdots\cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor+1}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
$$



Matrix Multiplication View I

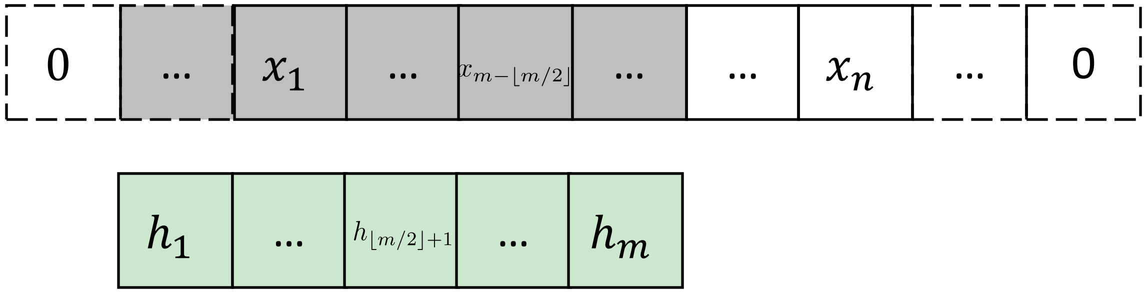
1D Convolution (Discrete) ⇔ Matrix Multiplication

Filter => Toeplitz matrix (diagonal-constant)

$$
y = h * x = \begin{bmatrix}
h_{\lfloor m/2 \rfloor+1} & h_{\lfloor m/2 \rfloor+2} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\
h_{\lfloor m/2 \rfloor} & h_{\lfloor m/2 \rfloor+1} & \cdots & h_{m-1} & h_m & 0 & \cdots & \cdots & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & 0 & \cdots & 0 \\
0 & h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & \cdots & \cdots & 0 & h_1 & h_2 & \cdots & h_{\lfloor m/2 \rfloor+2} \\
0 & 0 & \cdots & \cdots & & \cdots\cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor+1}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
$$

Input x

Filter h



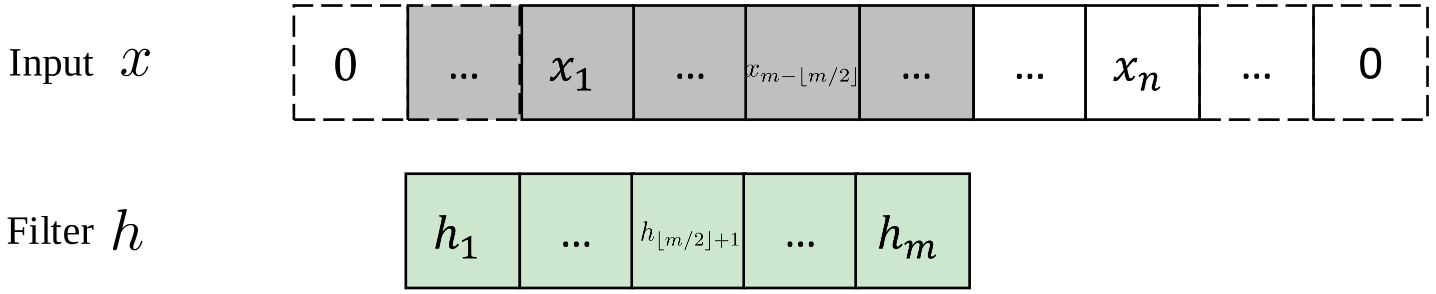
Matrix Multiplication View I

1D Convolution (Discrete) ⬄ Matrix Multiplication

Filter => Toeplitz matrix (diagonal-constant)

It could be very sparse (e.g., when n >> m)!

$$
y = h * x = \begin{bmatrix}
h_{\lfloor m/2 \rfloor+1} & h_{\lfloor m/2 \rfloor+2} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\
h_{\lfloor m/2 \rfloor} & h_{\lfloor m/2 \rfloor+1} & \cdots & h_{m-1} & h_m & 0 & \cdots & \cdots & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & 0 & \cdots & 0 \\
0 & h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & h_m & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & \cdots & \cdots & 0 & h_1 & h_2 & \cdots & h_{\lfloor m/2 \rfloor+2} \\
0 & 0 & \cdots & \cdots & & \cdots\cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor+1}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
$$



Matrix Multiplication View II

1D Convolution (Discrete) ⇔ Matrix Multiplication

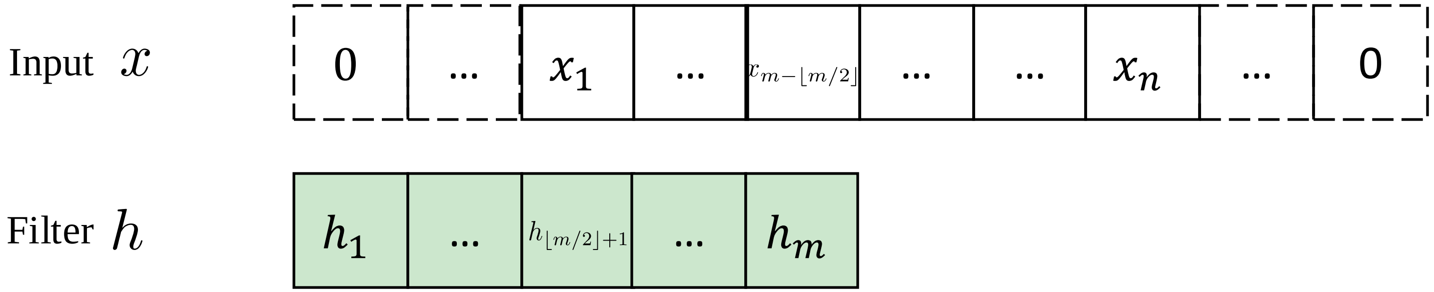
Data => Toeplitz matrix (diagonal-constant)

Matrix Multiplication View II

1D Convolution (Discrete) ⇔ Matrix Multiplication

Data => Toeplitz matrix (diagonal-constant)

$$y^\top = (h * x)^\top = \begin{bmatrix} h_m & h_{m-1} & \cdots & h_3 & h_2 & h_1 \end{bmatrix} \begin{bmatrix} x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor+1} & \cdots & x_m & x_{m+1} & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & x_{m-1} & x_m & \cdots & \vdots & \vdots \\ x_1 & x_2 & \cdots & \vdots & x_{m-1} & \cdots & x_n & 0 \\ 0 & x_1 & \cdots & \vdots & \vdots & \cdots & x_{n-1} & x_n \\ \vdots & 0 & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & x_1 & x_2 & \cdots & x_{n-\lfloor m/2 \rfloor+1} & x_{n-\lfloor m/2 \rfloor} \end{bmatrix}$$

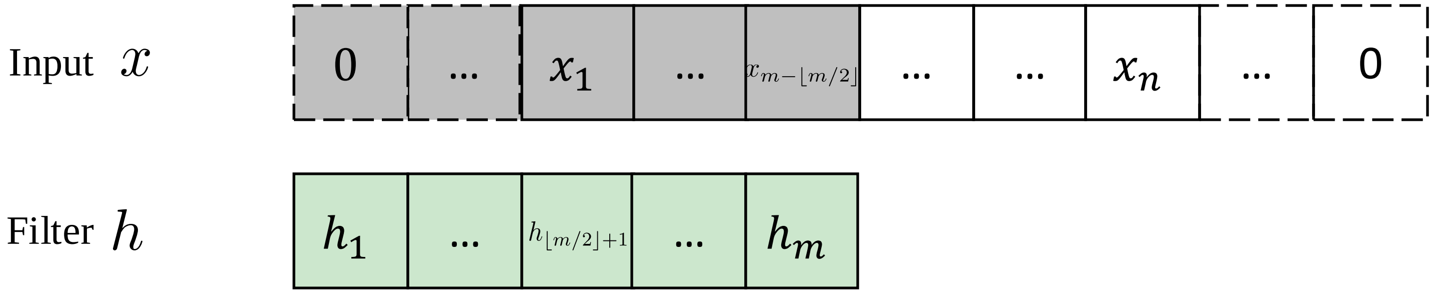


Matrix Multiplication View II

1D Convolution (Discrete) ⬄ Matrix Multiplication

Data => Toeplitz matrix (diagonal-constant)

$$y^\top = (h * x)^\top = \begin{bmatrix} h_m & h_{m-1} & \cdots & h_3 & h_2 & h_1 \end{bmatrix} \begin{bmatrix} x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor+1} & \cdots & x_m & x_{m+1} & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & x_{m-1} & x_m & \cdots & \vdots & \vdots \\ x_1 & x_2 & \cdots & \vdots & x_{m-1} & \cdots & x_n & 0 \\ 0 & x_1 & \cdots & \vdots & \vdots & \cdots & x_{n-1} & x_n \\ \vdots & 0 & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & x_1 & x_2 & \cdots & x_{n-\lfloor m/2 \rfloor+1} & x_{n-\lfloor m/2 \rfloor} \end{bmatrix}$$

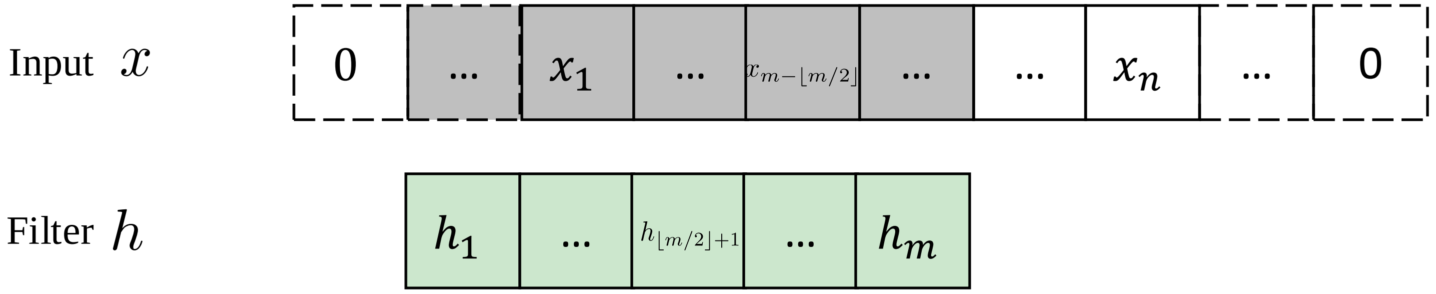


Matrix Multiplication View II

1D Convolution (Discrete) ⬄ Matrix Multiplication

Data => Toeplitz matrix (diagonal-constant)

$$y^\top = (h * x)^\top = \begin{bmatrix} h_m & h_{m-1} & \cdots & h_3 & h_2 & h_1 \end{bmatrix} \begin{bmatrix} x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor+1} & \cdots & x_m & x_{m+1} & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & x_{m-1} & x_m & \cdots & \vdots & \vdots \\ x_1 & x_2 & \cdots & \vdots & x_{m-1} & \cdots & x_n & 0 \\ 0 & x_1 & \cdots & \vdots & \vdots & \cdots & x_{n-1} & x_n \\ \vdots & 0 & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & x_1 & x_2 & \cdots & x_{n-\lfloor m/2 \rfloor+1} & x_{n-\lfloor m/2 \rfloor} \end{bmatrix}$$



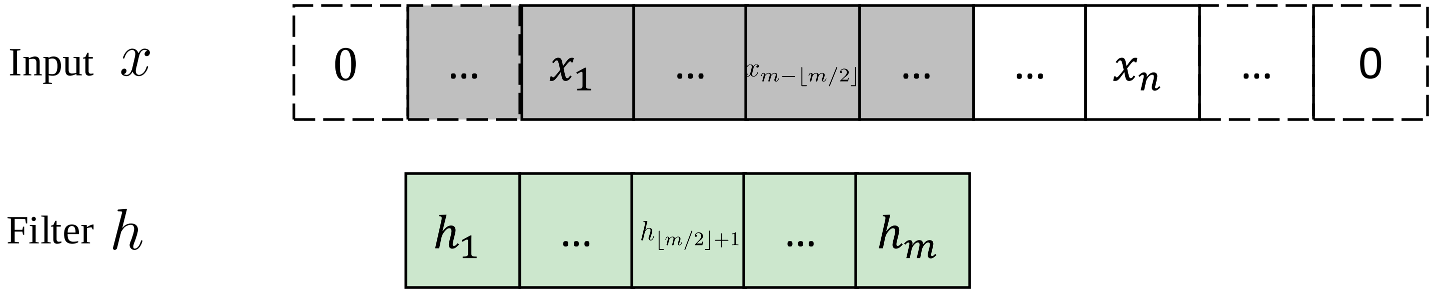
Matrix Multiplication View II

1D Convolution (Discrete) ⇔ Matrix Multiplication

Data => Toeplitz matrix (diagonal-constant)

It could be dense (e.g., when n >> m)!

$$
y^\top = (h * x)^\top = \begin{bmatrix} h_m & h_{m-1} & \cdots & h_3 & h_2 & h_1 \end{bmatrix}
\begin{bmatrix}
x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor+1} & \cdots & x_m & x_{m+1} & \cdots & 0 & 0 \\
\vdots & \vdots & \cdots & x_{m-1} & x_m & \cdots & \vdots & \vdots \\
x_1 & x_2 & \cdots & \vdots & x_{m-1} & \cdots & x_n & 0 \\
0 & x_1 & \cdots & \vdots & \vdots & \cdots & x_{n-1} & x_n \\
\vdots & 0 & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\
0 & 0 & \cdots & x_1 & x_2 & \cdots & x_{n-\lfloor m/2 \rfloor+1} & x_{n-\lfloor m/2 \rfloor}
\end{bmatrix}
$$



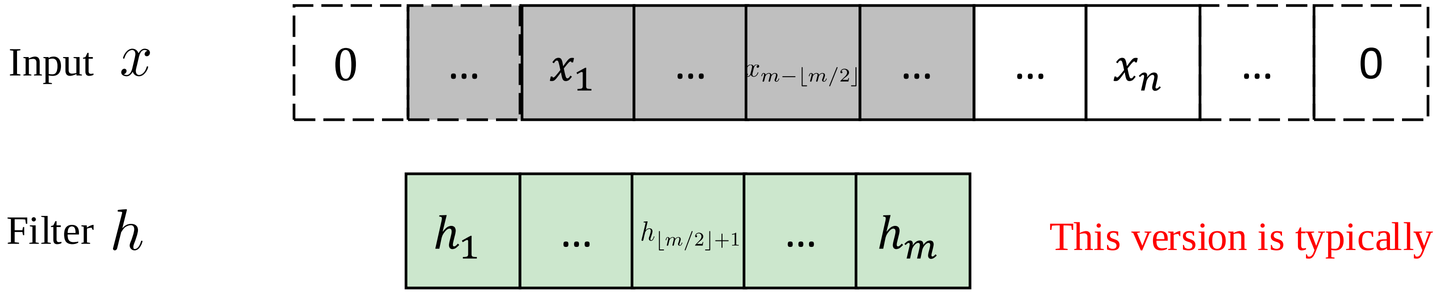
Matrix Multiplication View II

1D Convolution (Discrete) ⇔ Matrix Multiplication

Data => Toeplitz matrix (diagonal-constant)

It could be dense (e.g., when n >> m)!

$$y^\top = (h * x)^\top = \begin{bmatrix} h_m & h_{m-1} & \cdots & h_3 & h_2 & h_1 \end{bmatrix} \begin{bmatrix} x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor+1} & \cdots & x_m & x_{m+1} & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & x_{m-1} & x_m & \cdots & \vdots & \vdots \\ x_1 & x_2 & \cdots & \vdots & x_{m-1} & \cdots & x_n & 0 \\ 0 & x_1 & \cdots & \vdots & \vdots & \cdots & x_{n-1} & x_n \\ \vdots & 0 & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & x_1 & x_2 & \cdots & x_{n-\lfloor m/2 \rfloor+1} & x_{n-\lfloor m/2 \rfloor} \end{bmatrix}$$



This version is typically implemented on GPUs!

Matrix Multiplication View II

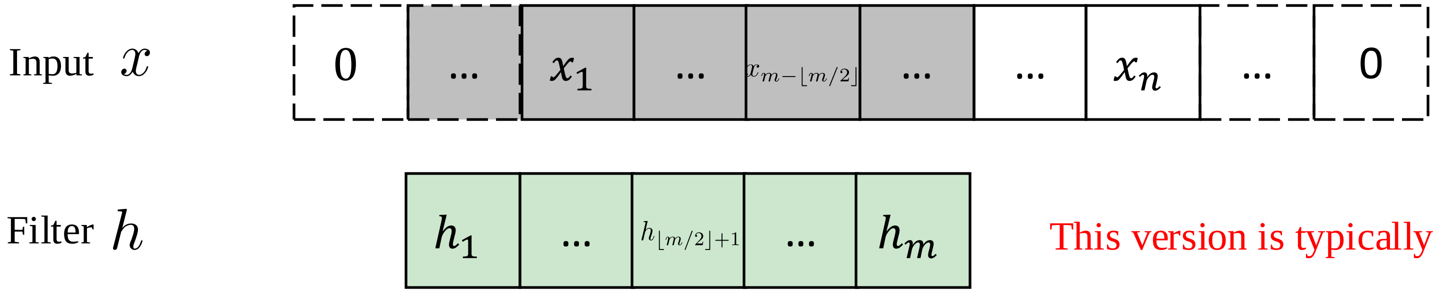
1D Convolution (Discrete) ⬄ Matrix Multiplication

This equivalence holds for 2D and other higher-order convolutions!

Data => Toeplitz matrix (diagonal-constant)

It could be dense (e.g., when n >> m)!

$$y^\top = (h * x)^\top = \begin{bmatrix} h_m & h_{m-1} & \cdots & h_3 & h_2 & h_1 \end{bmatrix} \begin{bmatrix} x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor+1} & \cdots & x_m & x_{m+1} & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & x_{m-1} & x_m & \cdots & \vdots & \vdots \\ x_1 & x_2 & \cdots & \vdots & x_{m-1} & \cdots & x_n & 0 \\ 0 & x_1 & \cdots & \vdots & \vdots & \cdots & x_{n-1} & x_n \\ \vdots & 0 & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & x_1 & x_2 & \cdots & x_{n-\lfloor m/2 \rfloor+1} & x_{n-\lfloor m/2 \rfloor} \end{bmatrix}$$

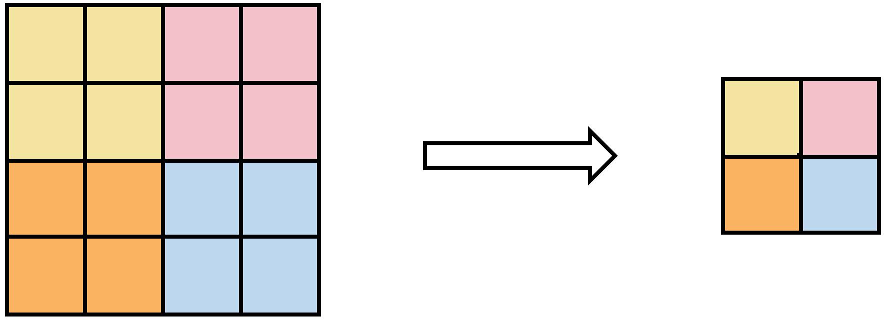


This version is typically implemented on GPUs!

Basic Deep Learning Models

Convolutional Neural Network (CNN)

Pooling (e.g., 2X2)



Basic Deep Learning Models

Convolutional Neural Network (CNN)

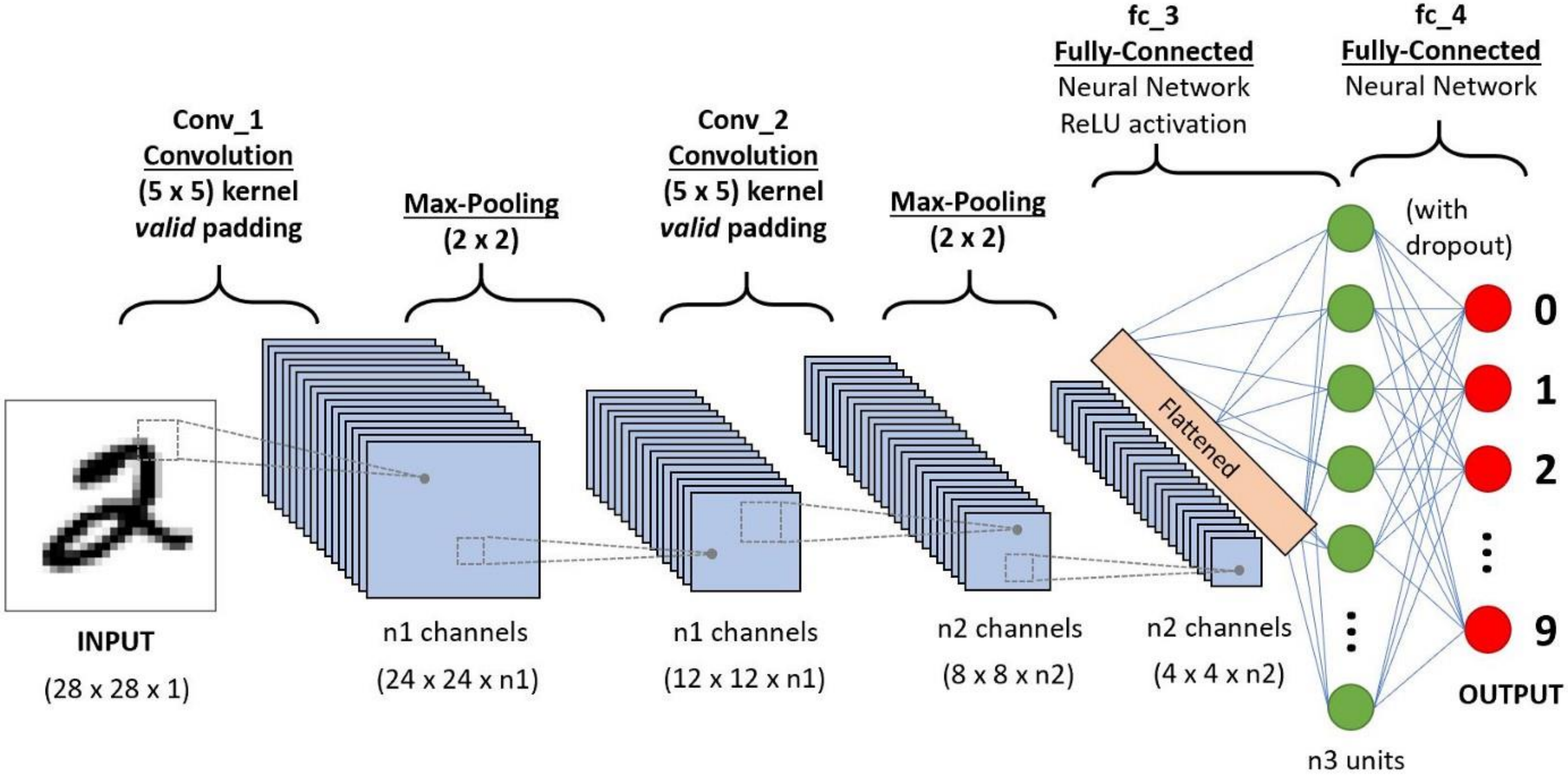


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Basic Deep Learning Models

Recurrent Neural Network (RNN)

Same neural network gets reused many times!

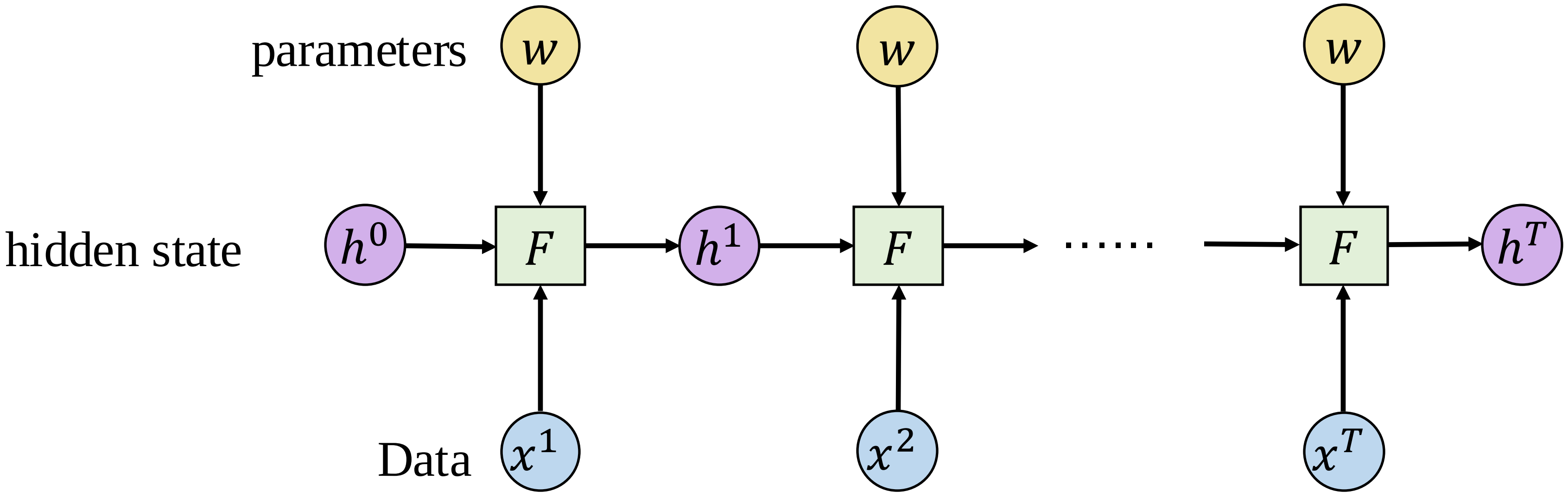
$$\mathbf{h}^t = F(\mathbf{x}^t, \mathbf{h}^{t-1}, W)$$

Basic Deep Learning Models

Recurrent Neural Network (RNN)

Same neural network gets reused many times!

$$\mathbf{h}^t = F(\mathbf{x}^t, \mathbf{h}^{t-1}, W)$$

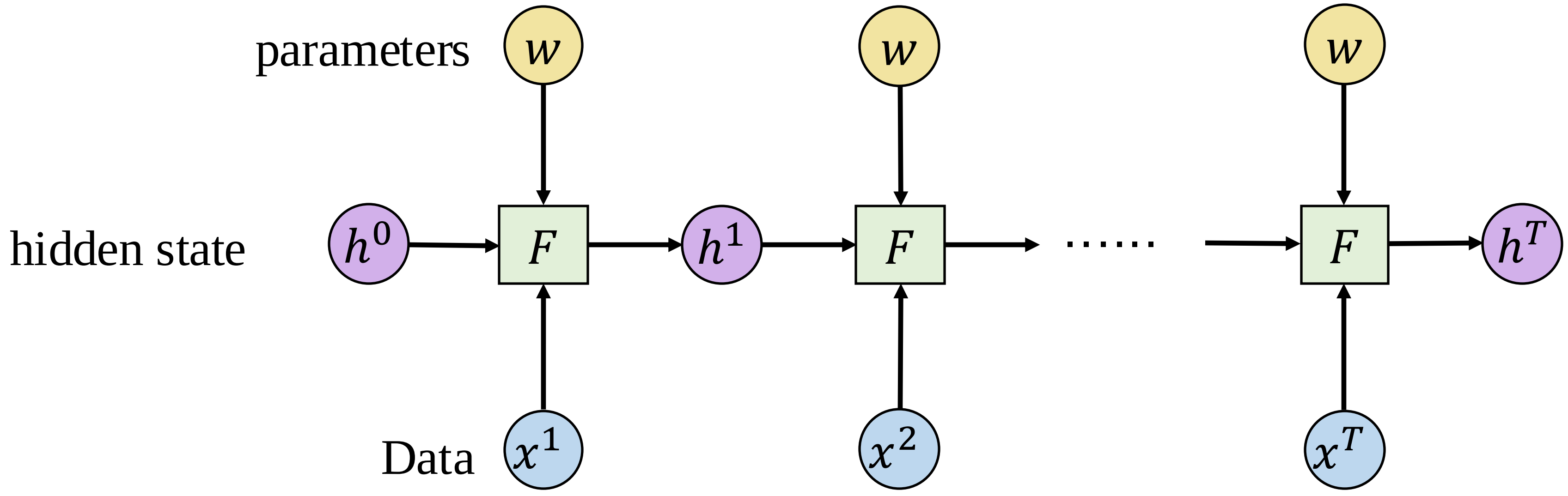


Basic Deep Learning Models

Recurrent Neural Network (RNN)

Same neural network gets reused many times!

$$\mathbf{h}^t = F(\mathbf{x}^t, \mathbf{h}^{t-1}, W)$$



F could be any neural network!

Outline

- Brief Introduction & History & Application
- Basic Deep Learning Models
  - Multi-Layer Perceptron (MLP)
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- **Objective Function**
- Learning Algorithm: Back-propagation
- Limitations

Objective (Loss) Function

- Supervised Learning

Given (data, label), we want to minimize empirical risk/loss

Loss = Function(label, model(data))

Objective (Loss) Function

- Supervised Learning

Empirical Risk Minimization (ERM)!

Given (data, label), we want to minimize empirical risk/loss

Loss = Function(label, model(data))

Objective (Loss) Function

- Supervised Learning

Empirical Risk Minimization (ERM)!

Given (data, label), we want to minimize empirical risk/loss

Loss = Function(label, model(data))

- Classification

Cross-Entropy Loss:

$$\ell(p, q) = -\sum_{i=1}^{K} p_i \log q_i$$

Objective (Loss) Function

- Supervised Learning

Empirical Risk Minimization (ERM)!

Given (data, label), we want to minimize empirical risk/loss

Loss = Function(label, model(data))

- Classification

Cross-Entropy Loss:

$$\ell(p, q) = -\sum_{i=1}^{K} p_i \log q_i$$

- Regression

Mean-Squared Error (MSE):

$$\ell(\mathbf{x}, \mathbf{y}) = \frac{1}{K} \|\mathbf{x} - \mathbf{y}\|_2^2$$

Objective (Loss) Function

Unsupervised/Self-supervised Learning

Only data is given

Objective (Loss) Function

Unsupervised/Self-supervised Learning

Only data is given

- Likelihood (Autoregressive models)
- Reconstruction Loss (Auto-encoders)
- Contrastive Loss (noise contrastive estimation, self-supervised learning)
- Min-max Loss (Generative adversarial networks)

……

Objective (Loss) Function

Unsupervised/Self-supervised Learning

Only data is given

- Likelihood (Autoregressive models)
- Reconstruction Loss (Auto-encoders)
- Contrastive Loss (noise contrastive estimation, self-supervised learning)
- Min-max Loss (Generative adversarial networks)

……

Designing a good objective function itself is a challenging research question!

Objective (Loss) Function

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**

- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**

- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Image Credit: Yann Lecun, NIPS 2016

Outline

- Brief Introduction & History & Application
- Basic Deep Learning Models
  - Multi-Layer Perceptron (MLP)
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Objective Function
- **Learning Algorithm: Back-propagation**
- Limitations

Learning Algorithm

Learning algorithm is about **credit assignment**

Assign credits based on contribution ⇔ Adjust parameters based on loss

Learning Algorithm

Learning algorithm is about **credit assignment**

Assign credits based on contribution ⇔ Adjust parameters based on loss

The most successful learning algorithm so far is **gradient based learning**!

Learning Algorithm

Learning algorithm is about **credit assignment**

Assign credits based on contribution ⇔ Adjust parameters based on loss

The most successful learning algorithm so far is **gradient based learning**!

Representative method: stochastic gradient descent (SGD), Robbins and Monro, 1951

Learning Algorithm

Learning algorithm is about **credit assignment**

Assign credits based on contribution ⇔ Adjust parameters based on loss

The most successful learning algorithm so far is **gradient based learning**!

Representative method: stochastic gradient descent (SGD), Robbins and Monro, 1951

Back-propagation (BP) = SGD in the context of deep learning

Back-Propagation

Multi-Layer Perceptron (MLP)

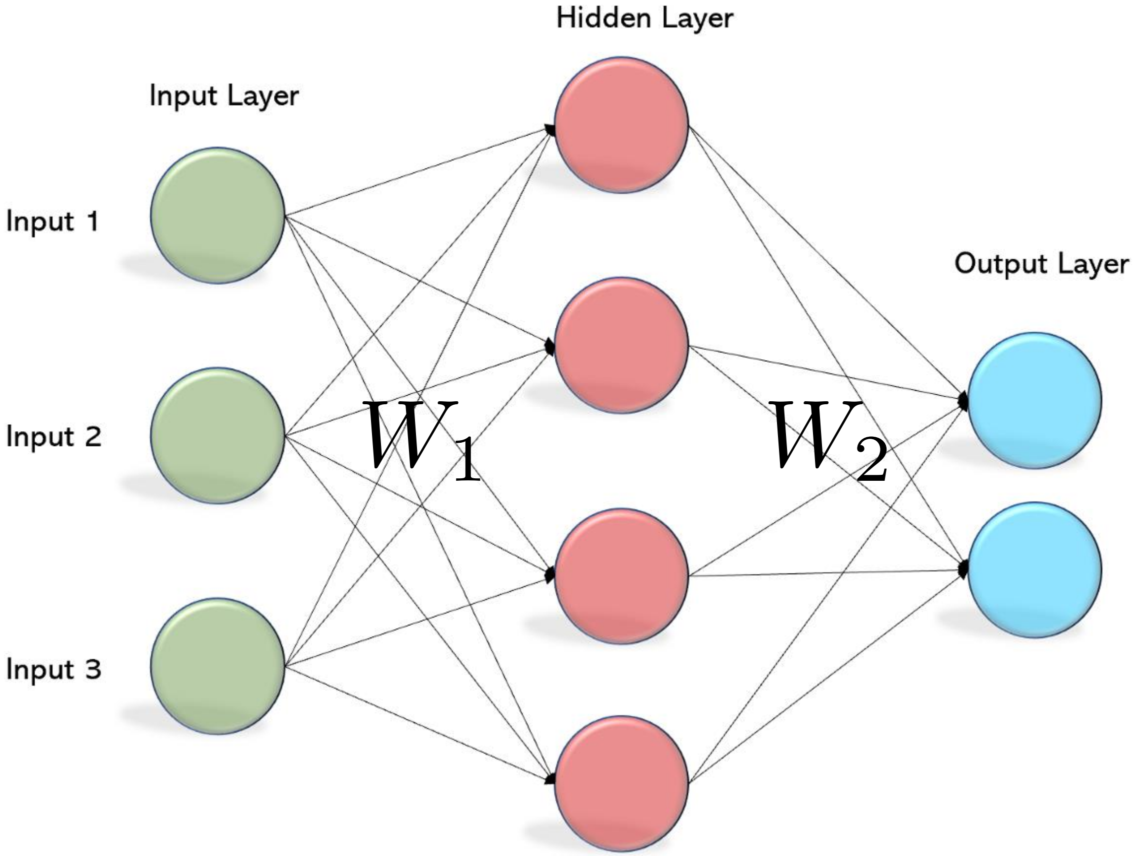


Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Back-Propagation

Forward Pass / Inference :

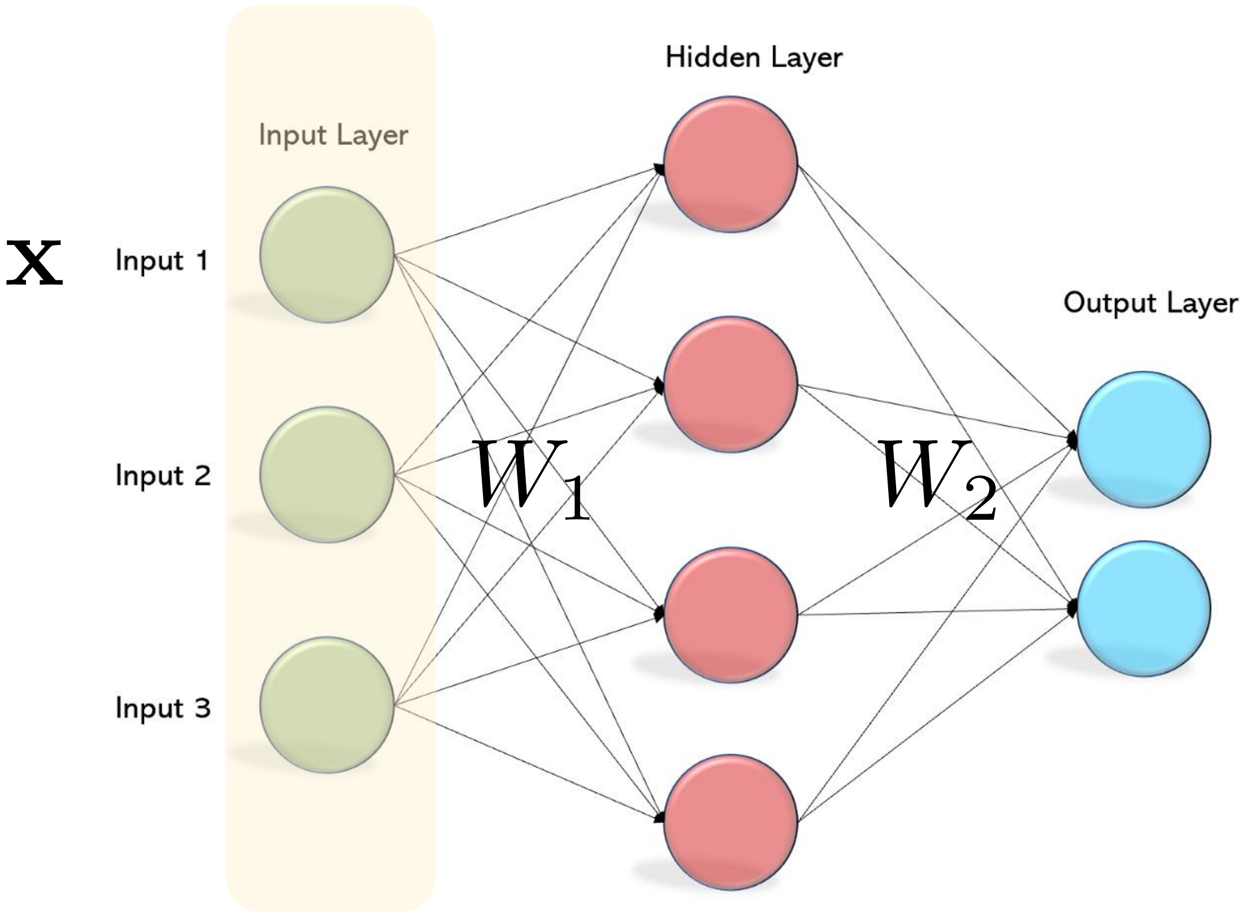


Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Back-Propagation

Forward Pass / Inference :

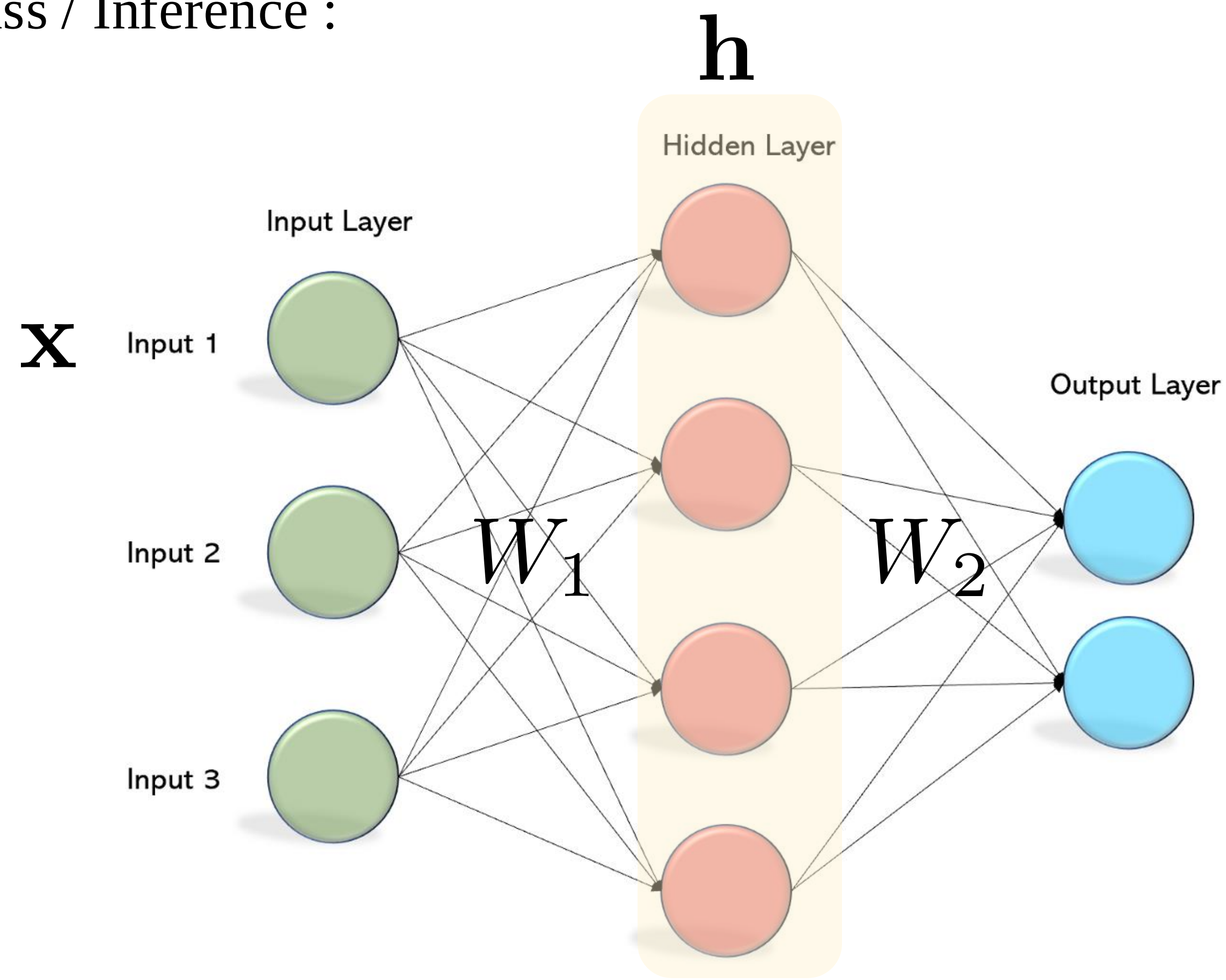


Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Back-Propagation

Forward Pass / Inference :

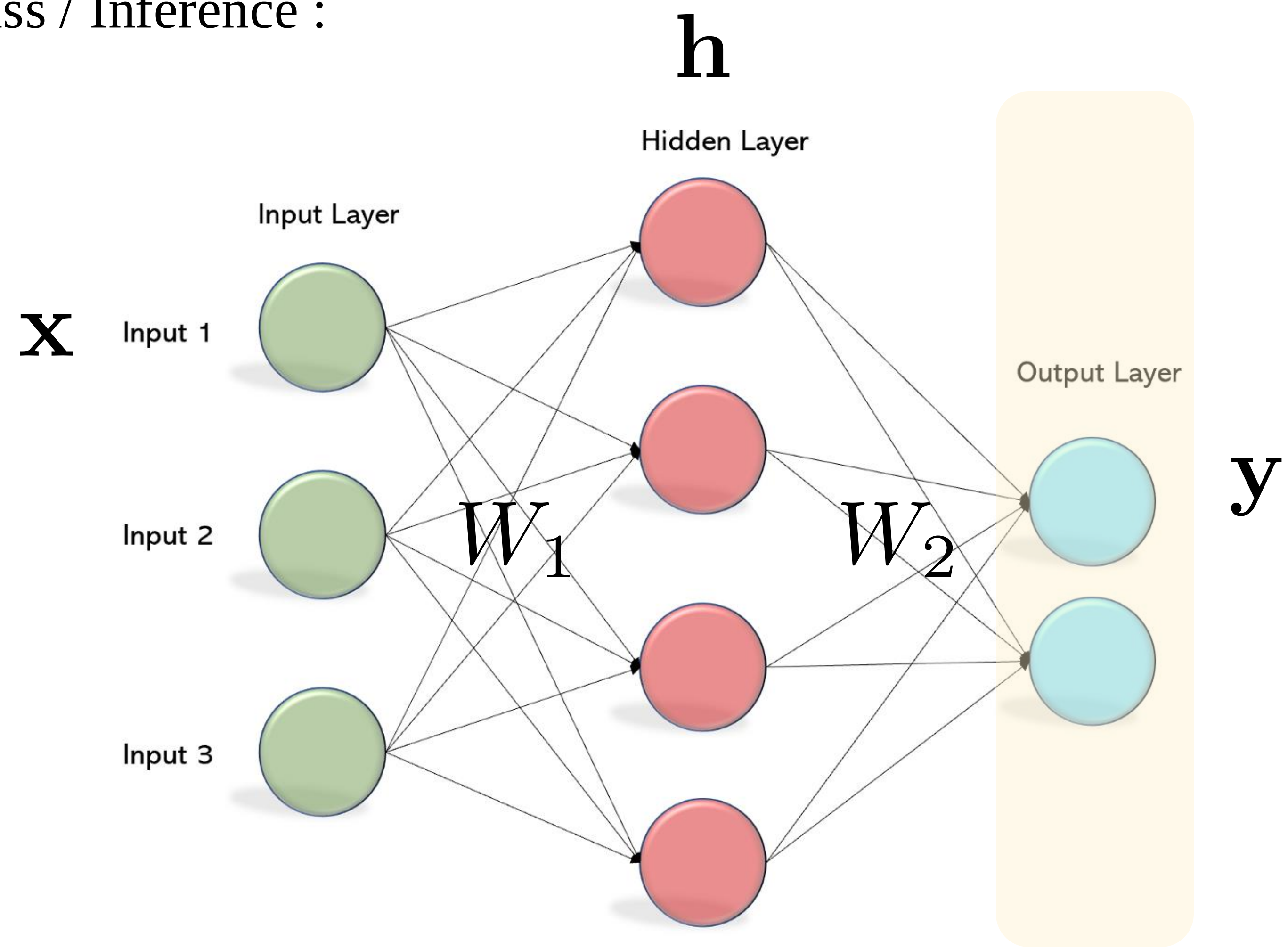
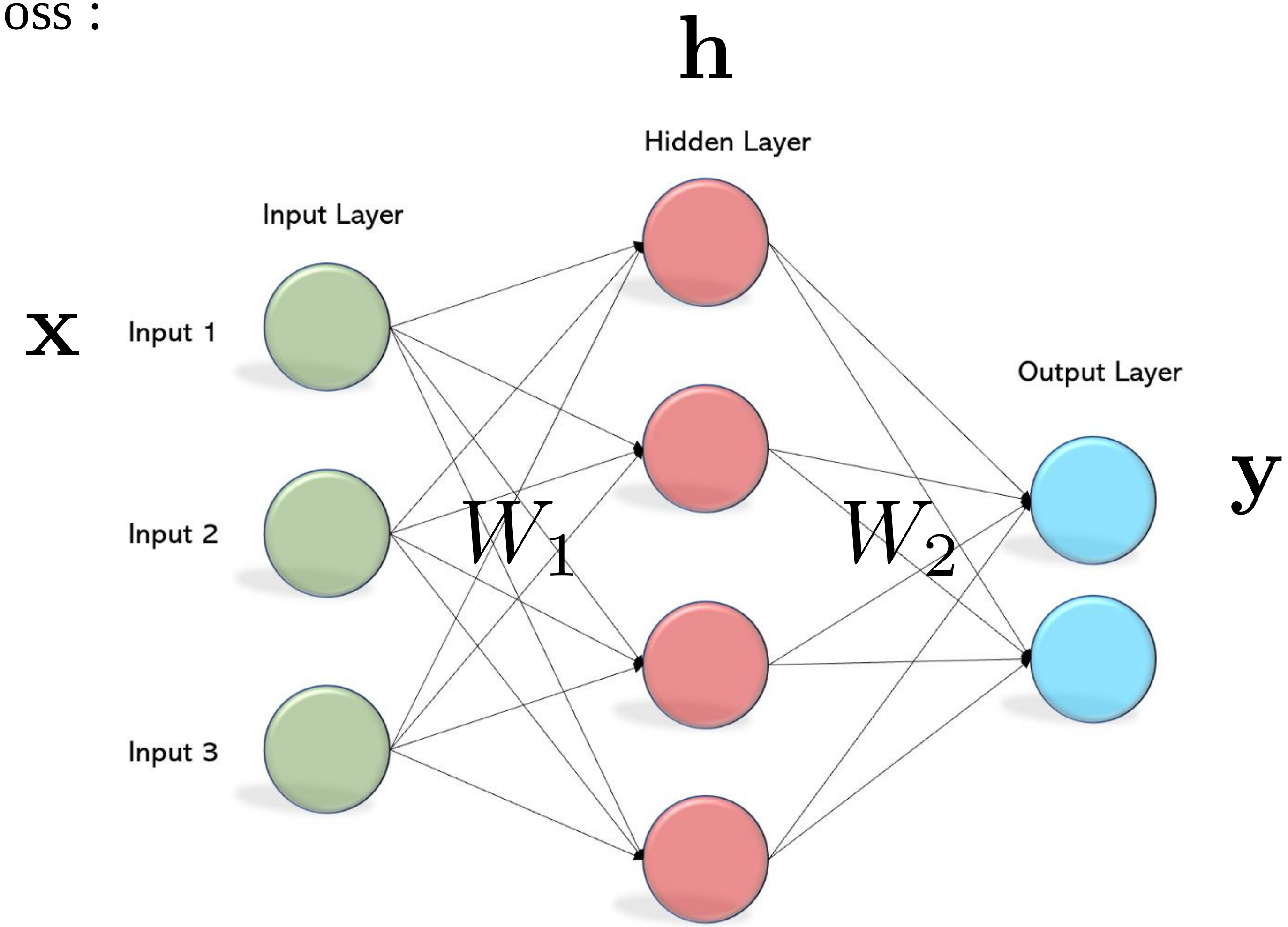


Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Back-Propagation

Compute Loss :



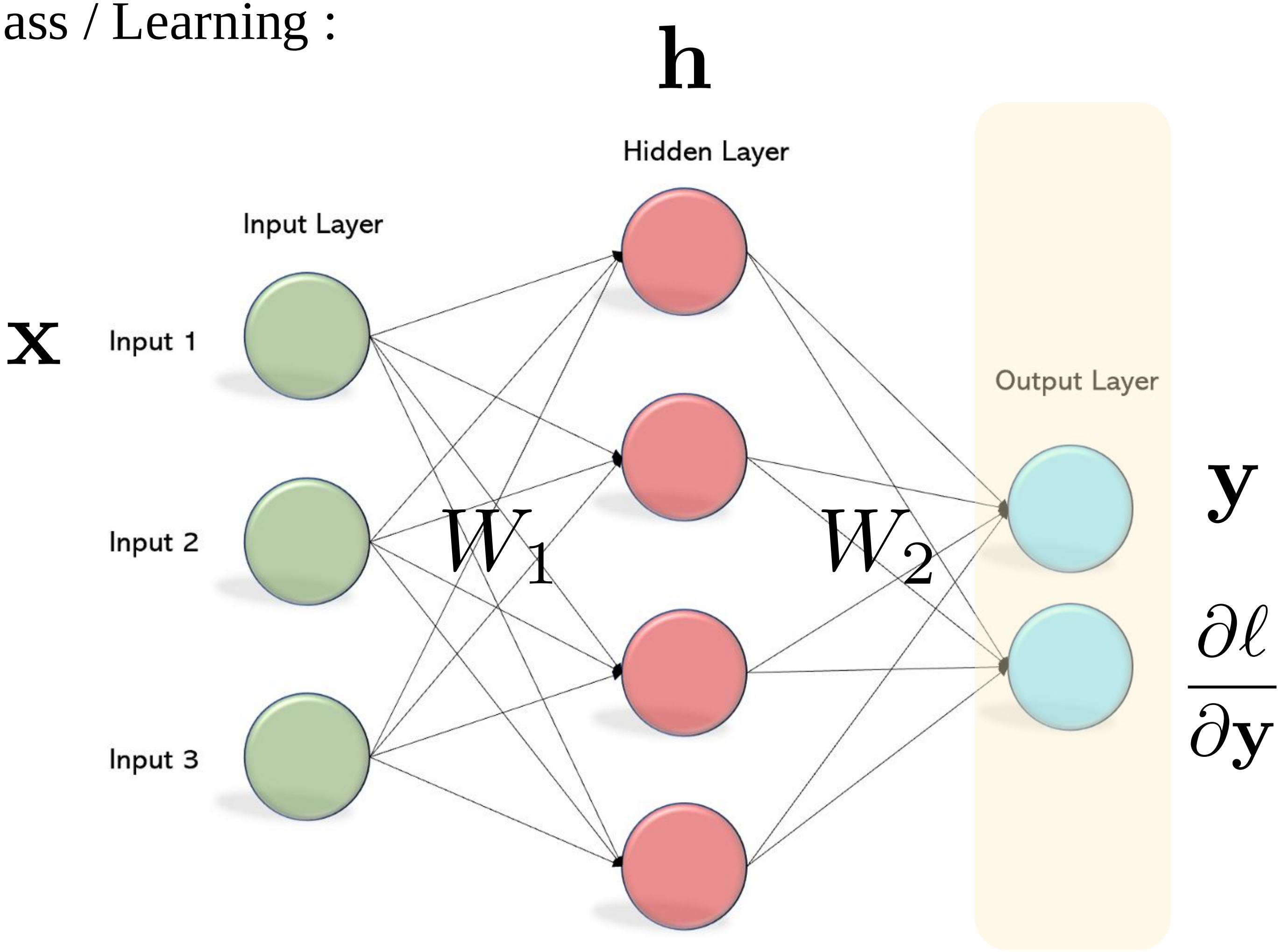
Loss

$$\ell(\mathbf{y}, \mathbf{y}_{\text{label}})$$

Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Back-Propagation

Backward Pass / Learning :



Loss

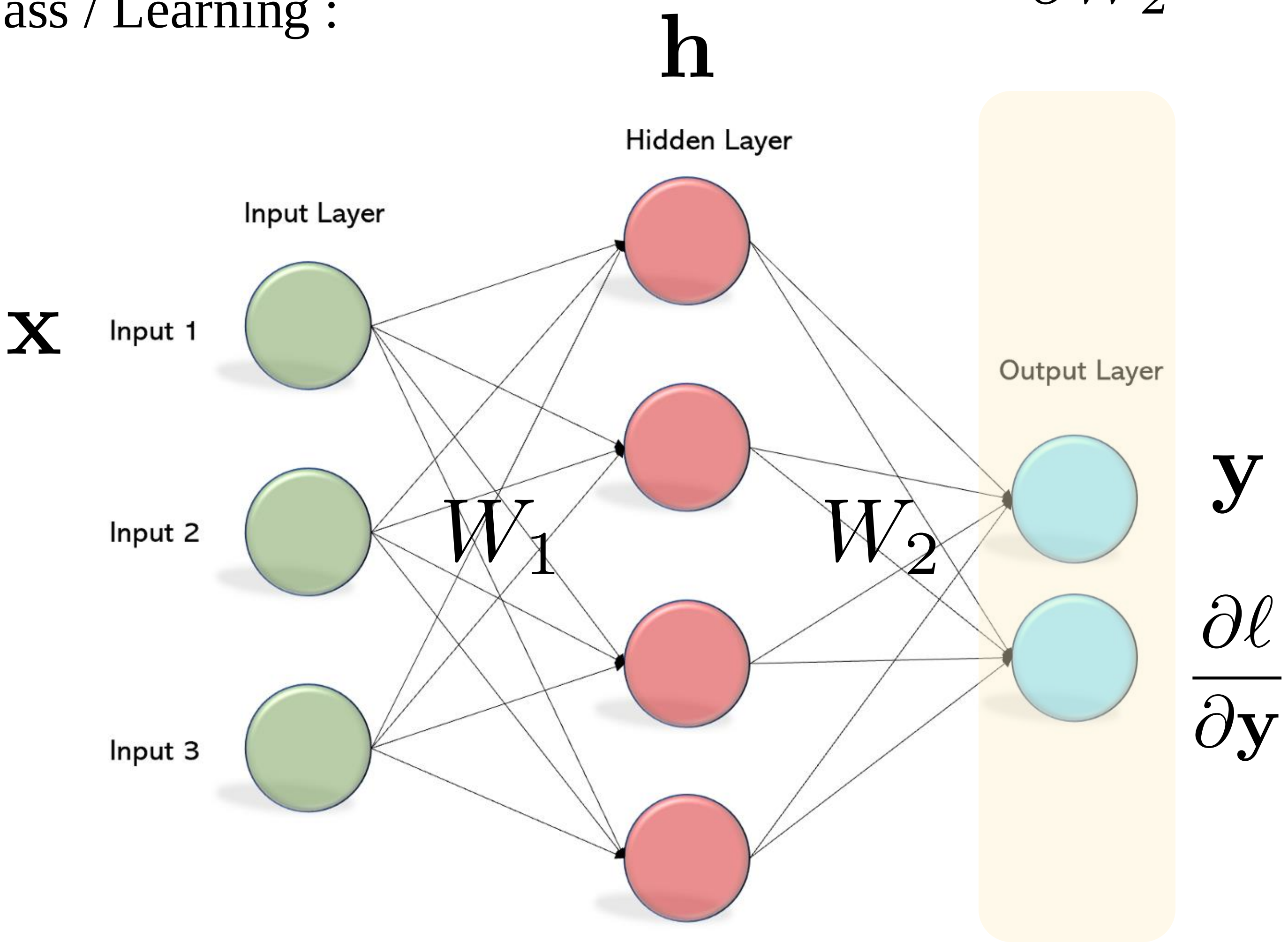
$\ell(\mathbf{y}, \mathbf{y}_{\text{label}})$

Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Back-Propagation

$$\frac{\partial \ell}{\partial W_2} = \left(\frac{\partial \mathbf{y}}{\partial W_2}\right)^\top \frac{\partial \ell}{\partial \mathbf{y}}$$

Backward Pass / Learning :



Loss

$\ell(\mathbf{y}, \mathbf{y}_{\text{label}})$

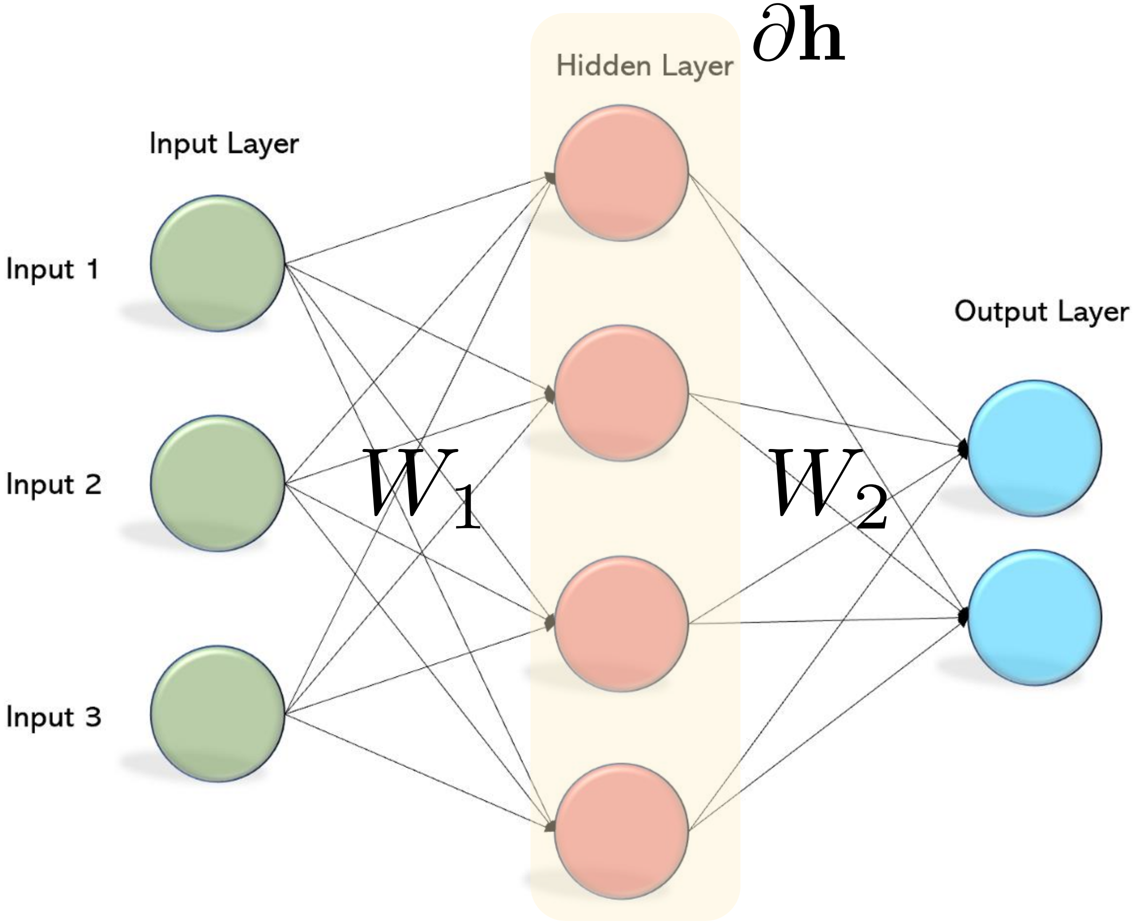
Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Back-Propagation

$$\frac{\partial \ell}{\partial \mathbf{h}} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{h}}\right)^\top \frac{\partial \ell}{\partial \mathbf{y}}$$

Backward Pass / Learning :

$\mathbf{h}$ $\quad \frac{\partial \ell}{\partial \mathbf{h}}$



$\mathbf{x}$ $\quad W_1 \quad W_2 \quad \mathbf{y} \quad \frac{\partial \ell}{\partial \mathbf{y}}$

Loss

$\ell(\mathbf{y}, \mathbf{y}_{\text{label}})$

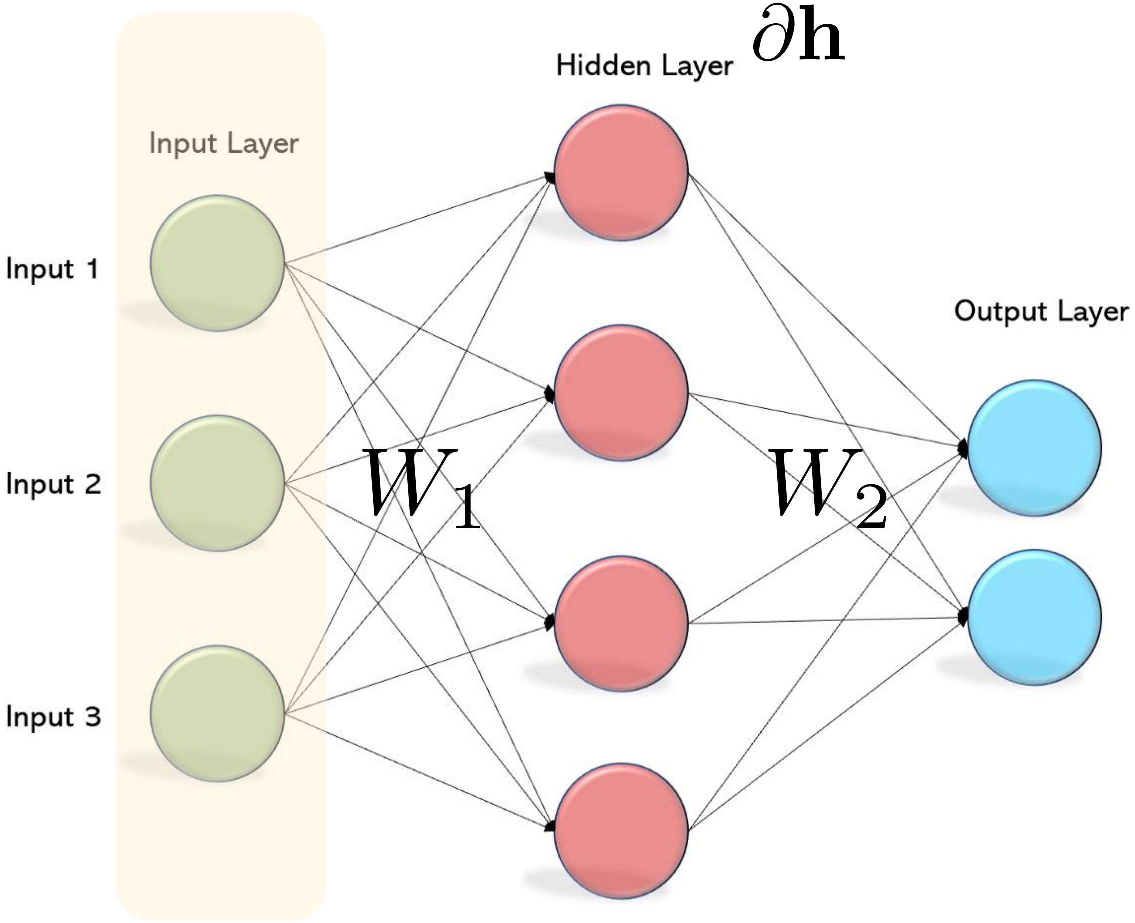
Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Back-Propagation

$$\frac{\partial \ell}{\partial W_1} = \left(\frac{\partial \mathbf{h}}{\partial W_1}\right)^{\top} \left(\frac{\partial \mathbf{y}}{\partial \mathbf{h}}\right)^{\top} \frac{\partial \ell}{\partial \mathbf{y}}$$

Backward Pass / Learning :

$\mathbf{h}$ $\frac{\partial \ell}{\partial \mathbf{h}}$



$\mathbf{x}$ $\frac{\partial \ell}{\partial \mathbf{x}}$

W_1 W_2

$\mathbf{y}$ $\frac{\partial \ell}{\partial \mathbf{y}}$

Loss
$\ell(\mathbf{y}, \mathbf{y}_{\text{label}})$

Image Credit: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Outline

- Brief Introduction & History & Application
- Basic Deep Learning Models
  - Multi-Layer Perceptron (MLP)
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Objective Function
- Learning Algorithm: Back-propagation
- **Limitations**

Limitations

- MLPs/CNNs are restricted to data with fixed size
  - Each sample needs to have the same size

Limitations

- MLPs/CNNs are restricted to data with fixed size
  - Each sample needs to have the same size
- RNNs can deal with varying-size data
  - Only presented as sequences

Limitations

- MLPs/CNNs are restricted to data with fixed size
  - Each sample needs to have the same size
- RNNs can deal with varying-size data
  - Only presented as sequences
- Learned representations do not explicitly encode structures of data
  - Hard to interpret and manipulate

Questions?