Quantum Neural Network

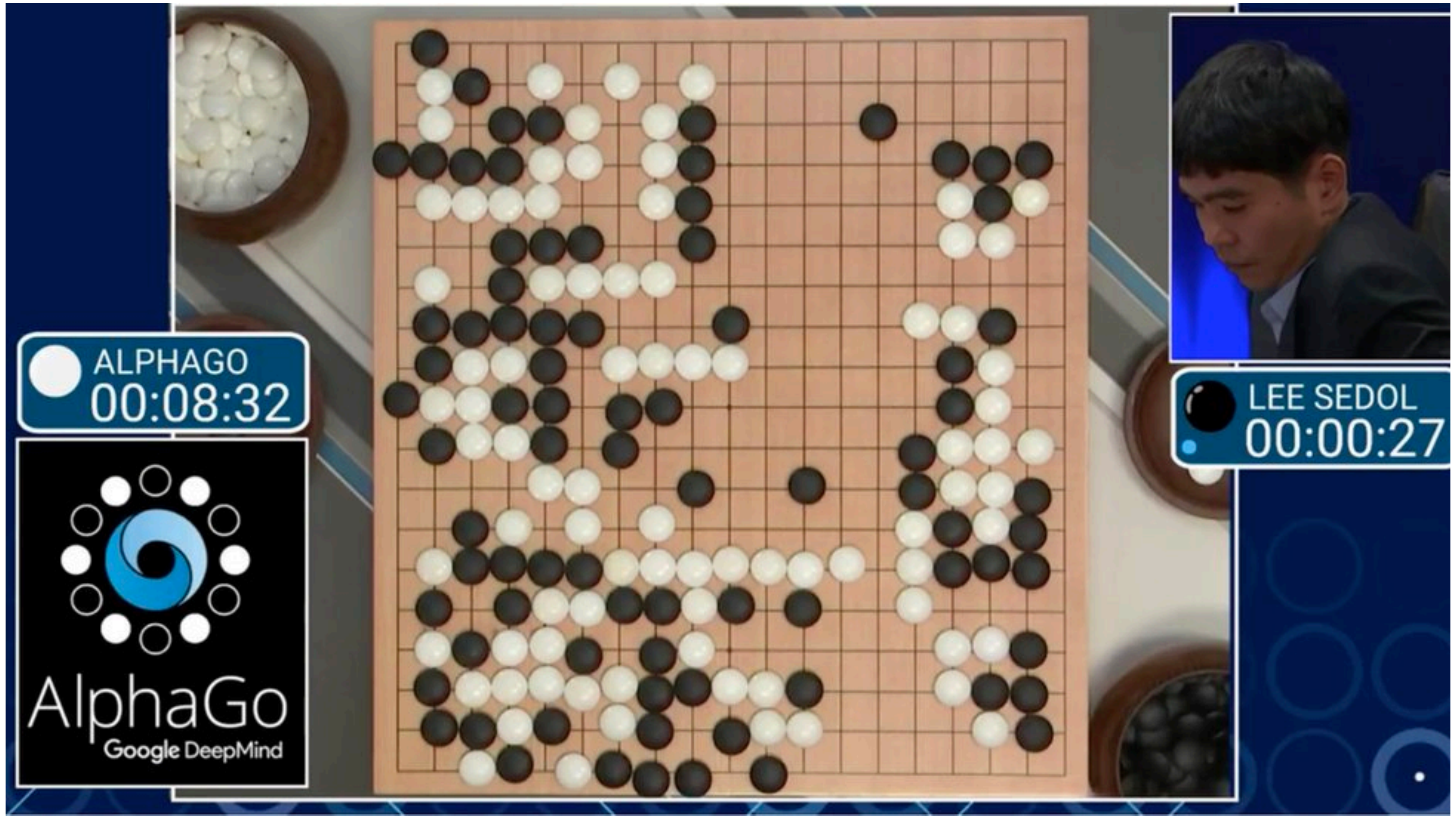
Shinwoo Park 2023.12.13

Index

- The power of modern AI
- Neural Networks
  - Perceptron
  - Feed Forward Neural Networks
  - Training Neural Networks
- Quantum Neural Networks
  - [Training deep quantum neural networks] Beer, Kerstin, et al. Nature communications 2020

The power of modern AI

AlphaGo



The power of modern AI

ChatGPT

ChatGPT

You

Explain me about quantum computing

ChatGPT

Quantum computing is a cutting-edge area of computing that leverages principles of quantum mechanics to perform operations on data. Traditional computers, based on classical physics, use bits—binary units of information represented as either 0 or 1—to process and store information. Quantum computing, on the other hand, operates using quantum bits or qubits.

Key principles of quantum computing include:

1. **Superposition:** Qubits can exist in multiple states simultaneously, unlike classical bits which can only be in one state (0 or 1) at a time. This property allows quantum computers to perform many calculations at once.
2. **Entanglement:** Qubits can be interconnected or entangled in a way that the state of one qubit is instantly correlated with the state of another, regardless of the distance between them. This phenomenon enables faster communication and computation.
3. **Quantum Interference:** Qubits can interfere with each other positively or negatively, increasing computational power when many qubits are involved.

Quantum computing's potential lies in its ability to solve certain complex problems exponentially faster than classical computers. Some of these problems include:

- **Factorization:** Breaking down large numbers into their prime factors, which is fundamental to cryptography (e.g., RSA encryption).
- **Optimization:** Finding the best solution among a vast number of possibilities in fields like logistics, finance, and resource allocation.
- **Molecular Simulation:** Modeling complex molecules and chemical reactions, which can aid drug discovery and material science.

However, quantum computing faces significant challenges such as qubit stability (as they are highly sensitive to environmental interference), error correction, scalability (building larger quantum computers), and the need for specialized quantum algorithms to fully utilize their power.

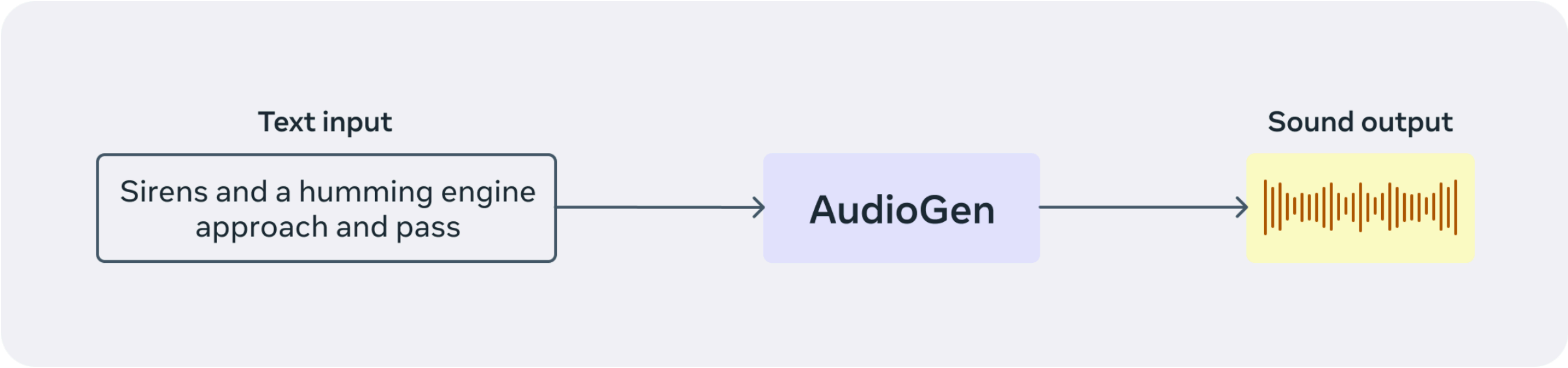
Several companies, including IBM, Google, Microsoft, and startups like Rigetti and IonQ, are actively researching and developing quantum computing hardware, software, and algorithms. While practical, large-scale quantum computers are still in the early stages of development, progress in this field holds enormous promise for revolutionizing various industries and solving problems that are currently intractable for classical computers.

The power of modern AI

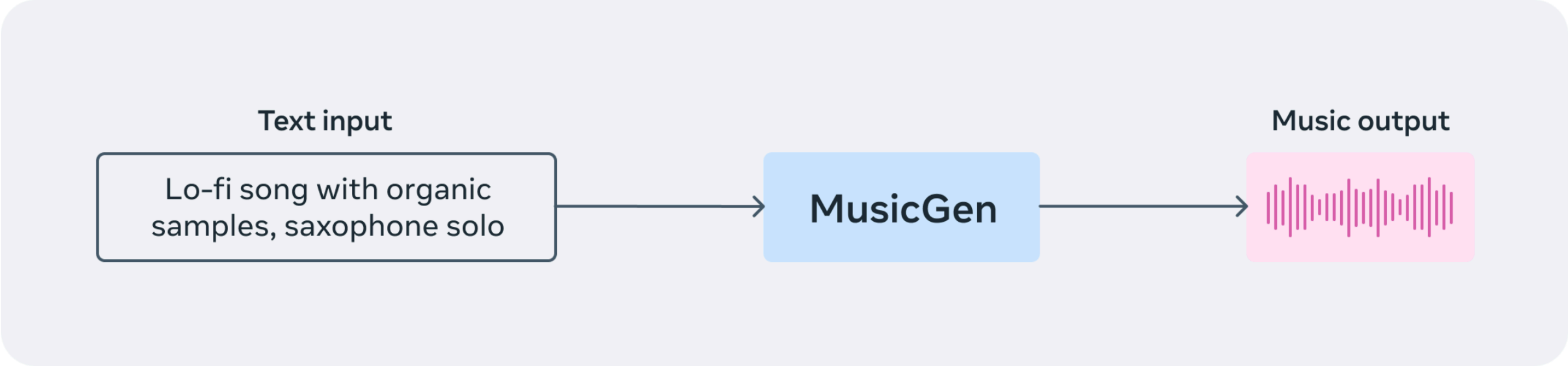
AudioCraft

Meta

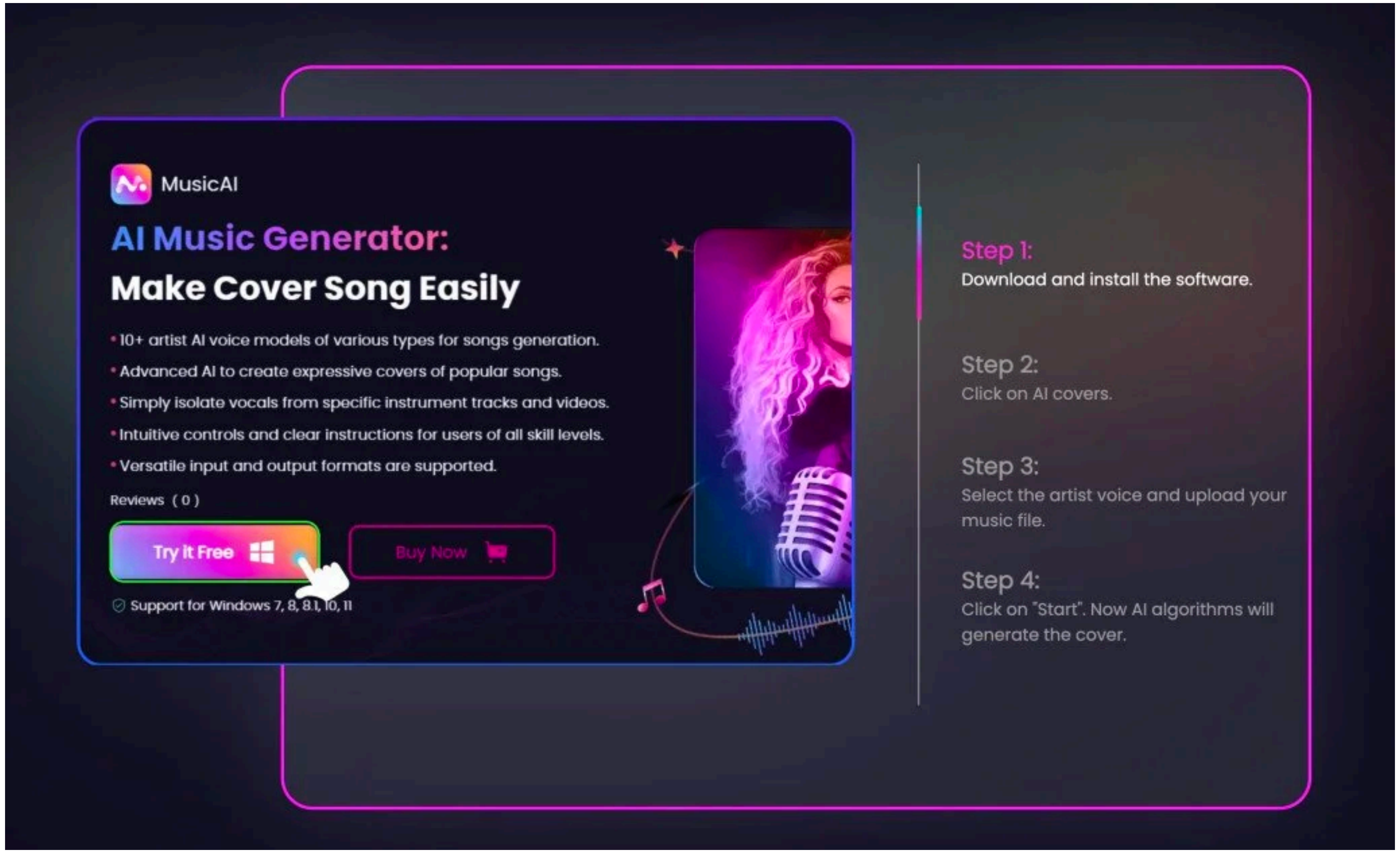
Text-to-sound generation



Text-to-music generation

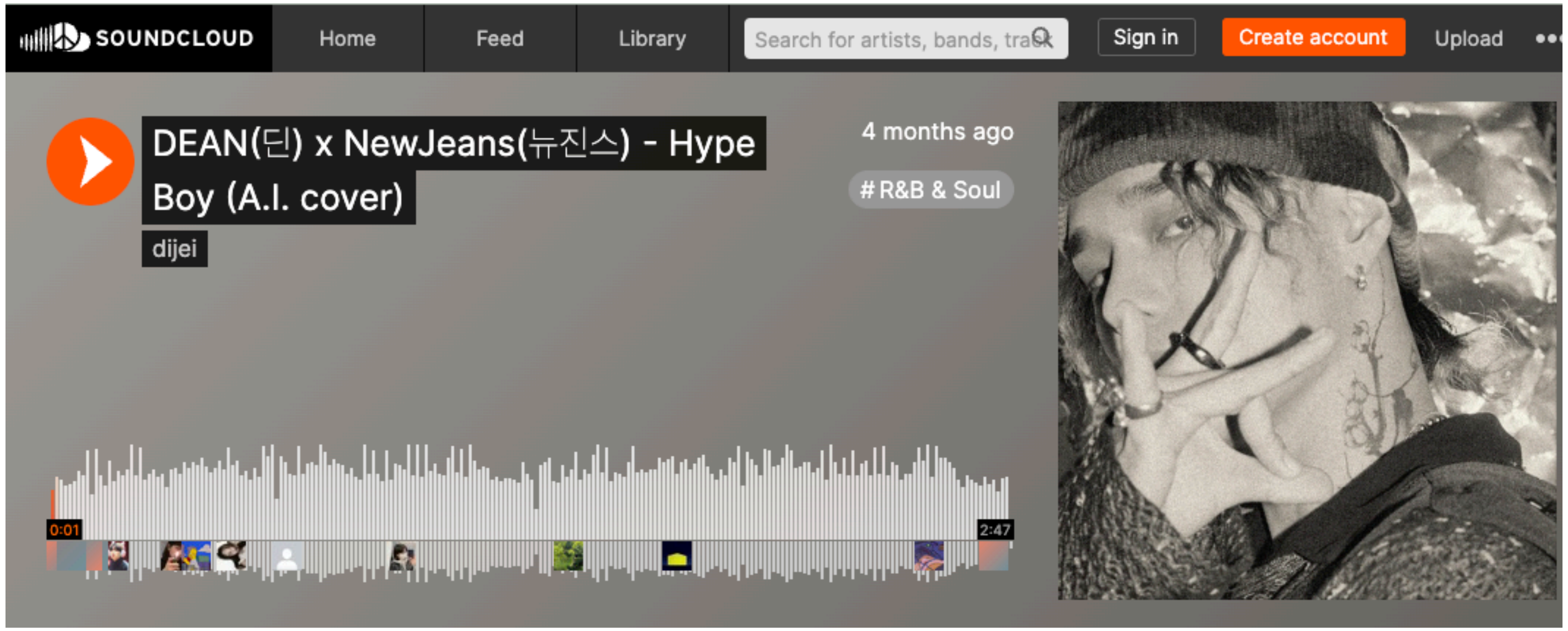


The power of modern AI
MusicAI



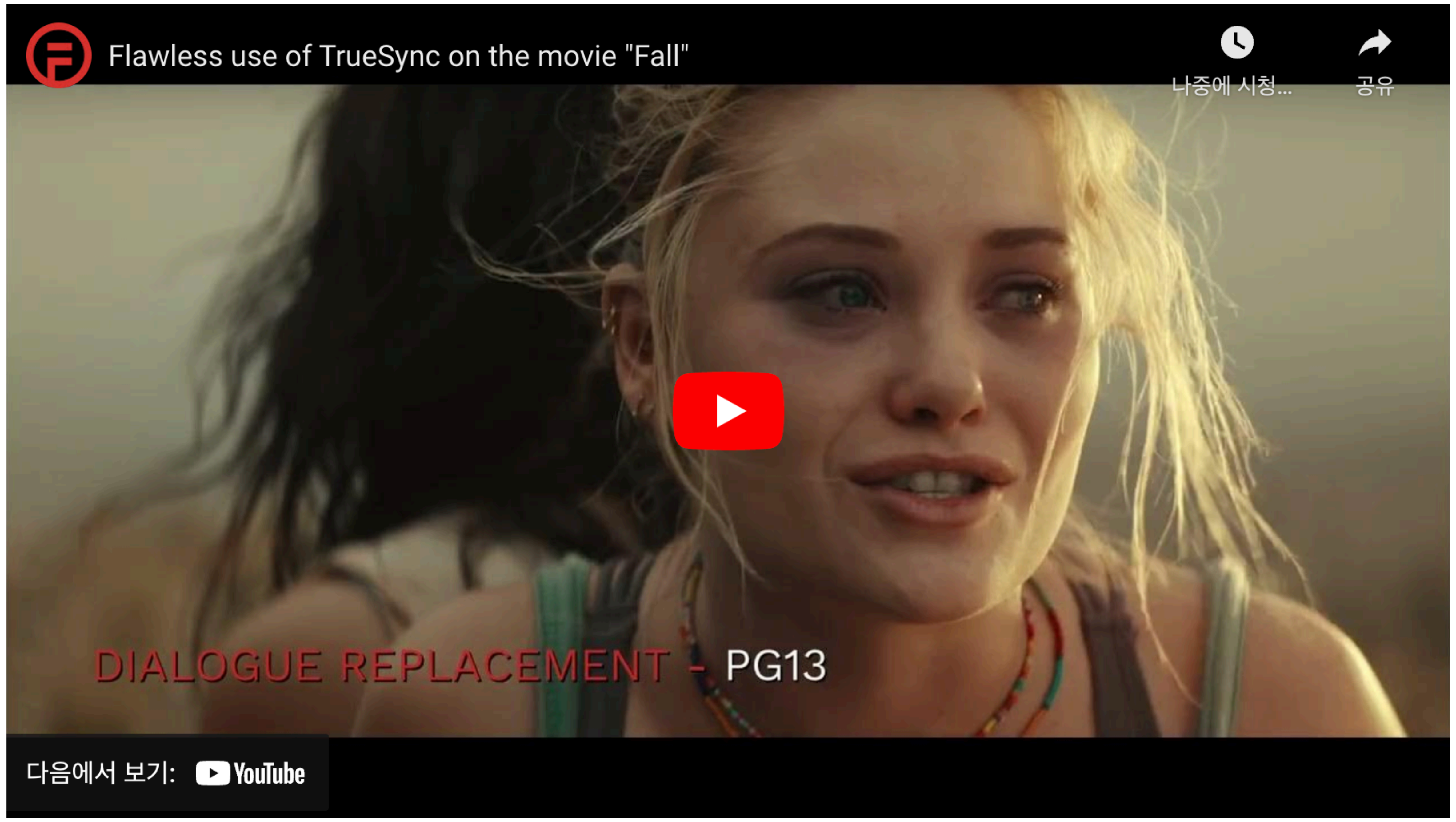
The power of modern AI

AI cover



The power of modern AI

Flawless AI



The power of modern AI
Midjourney

Jason Allen won first place in the digital art category at the Colorado State Fair art competition

by exhibiting 'Space Opera Theater' created through Midjourney.

The power of modern AI

Stable Diffusion

An image generated by Stable Diffusion

based on the text prompt

“a photograph of an astronaut riding a horse”

The power of modern AI

DALL-E + ChatGPT

MI My 5 year old keeps talking about a "super-duper sunflower hedgehog" -- what does it look like?

The power of modern AI

DALL-E + ChatGPT

MI Could you design some stickers?

The power of modern AI

Runway Gen-2

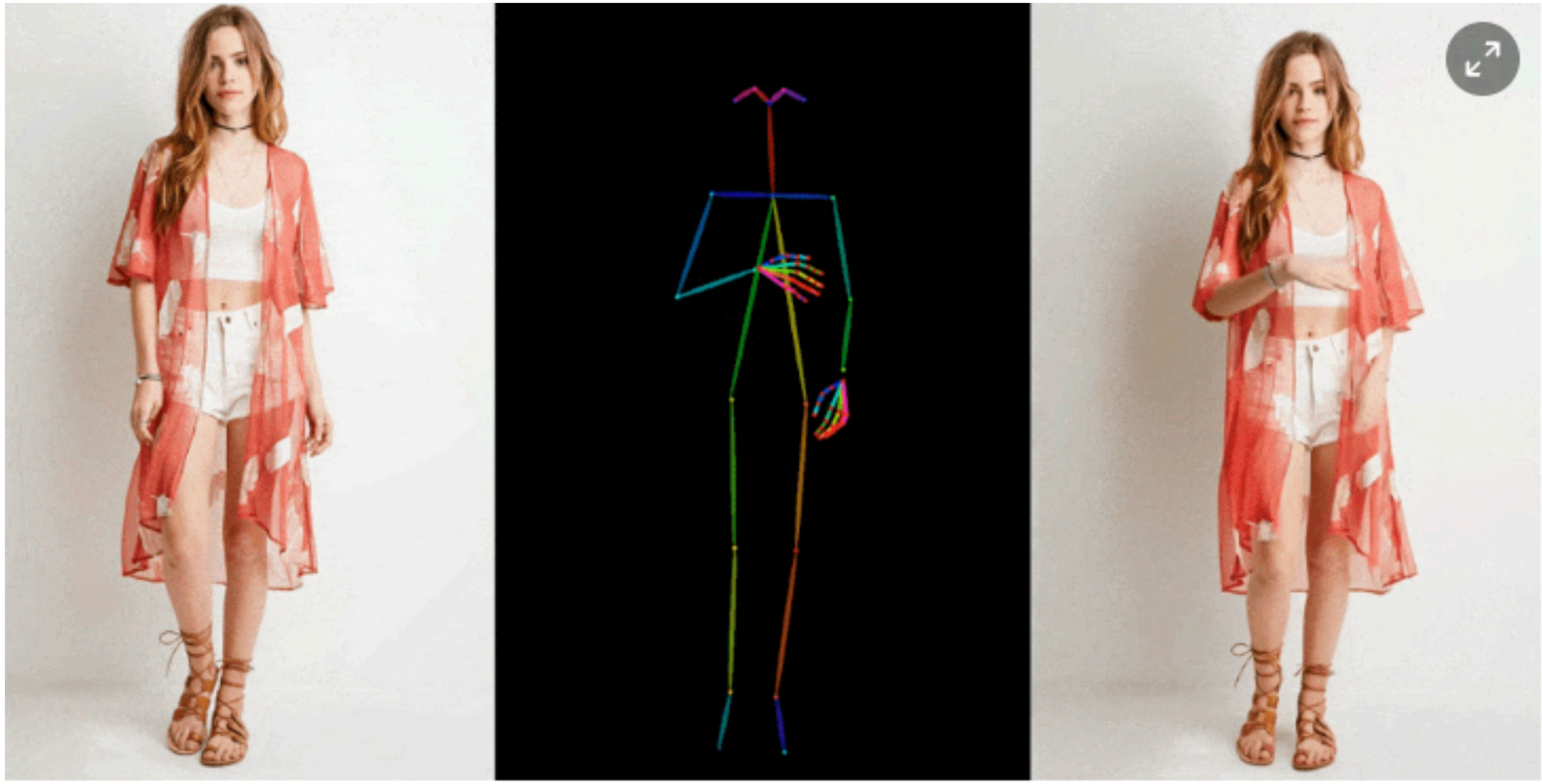
Mode 01: *Text to Video*

Synthesize videos in any style you can imagine using nothing but a text prompt. If you can say it, now you can see it.

The late afternoon sun peeking through the window of a New York City loft.

The power of modern AI

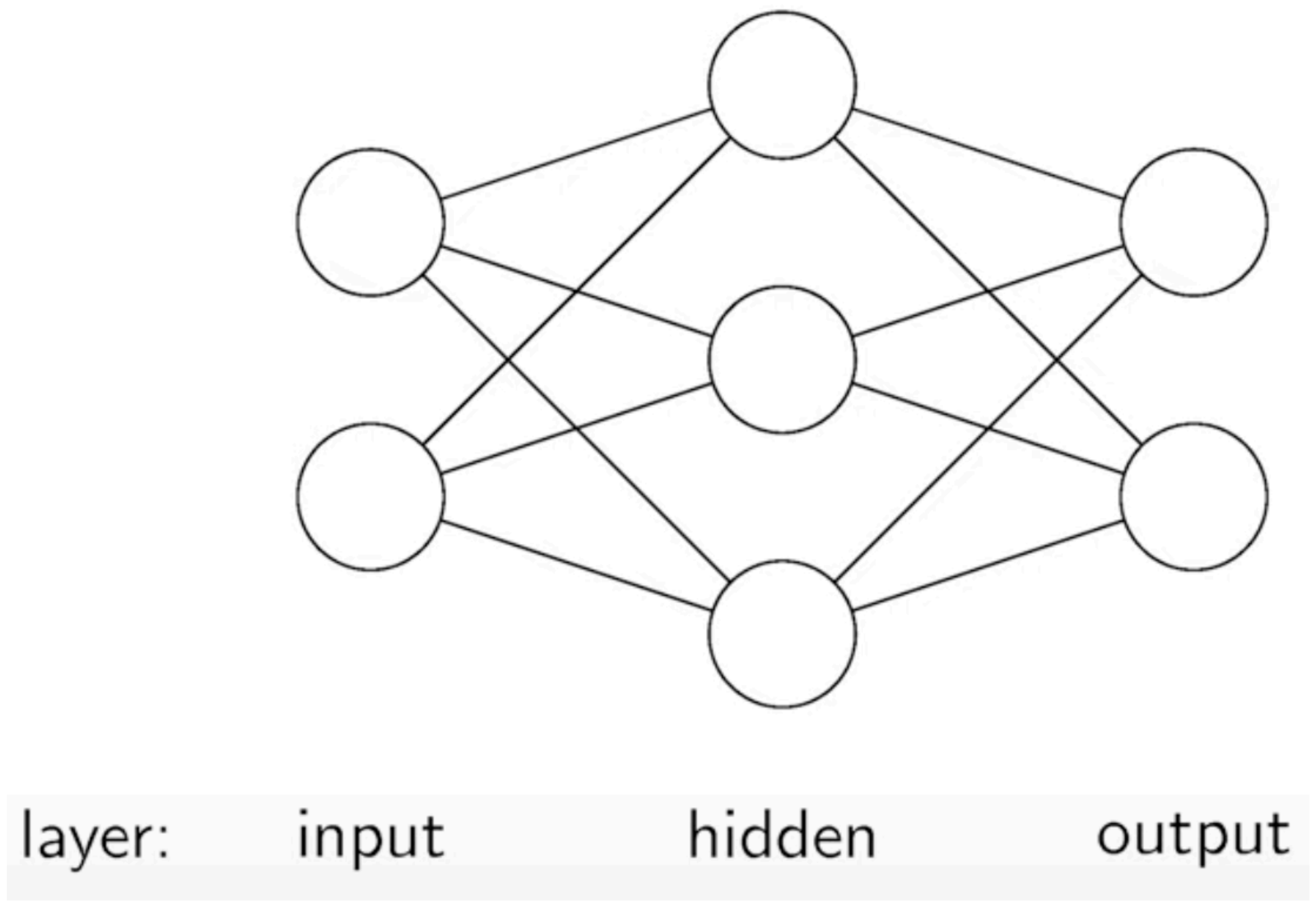
Animate Anyone



The power of modern AI

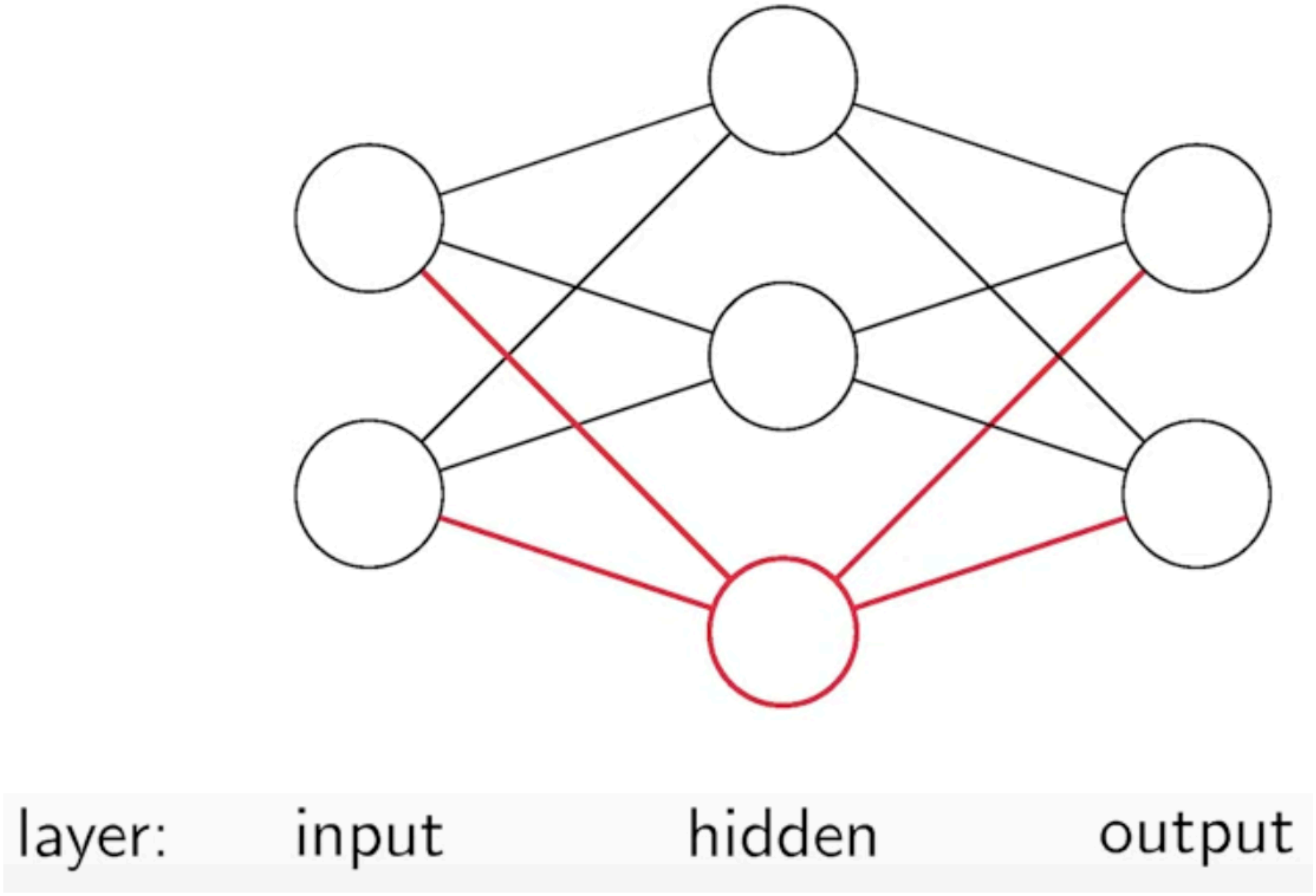
And many mores!

Neural Networks



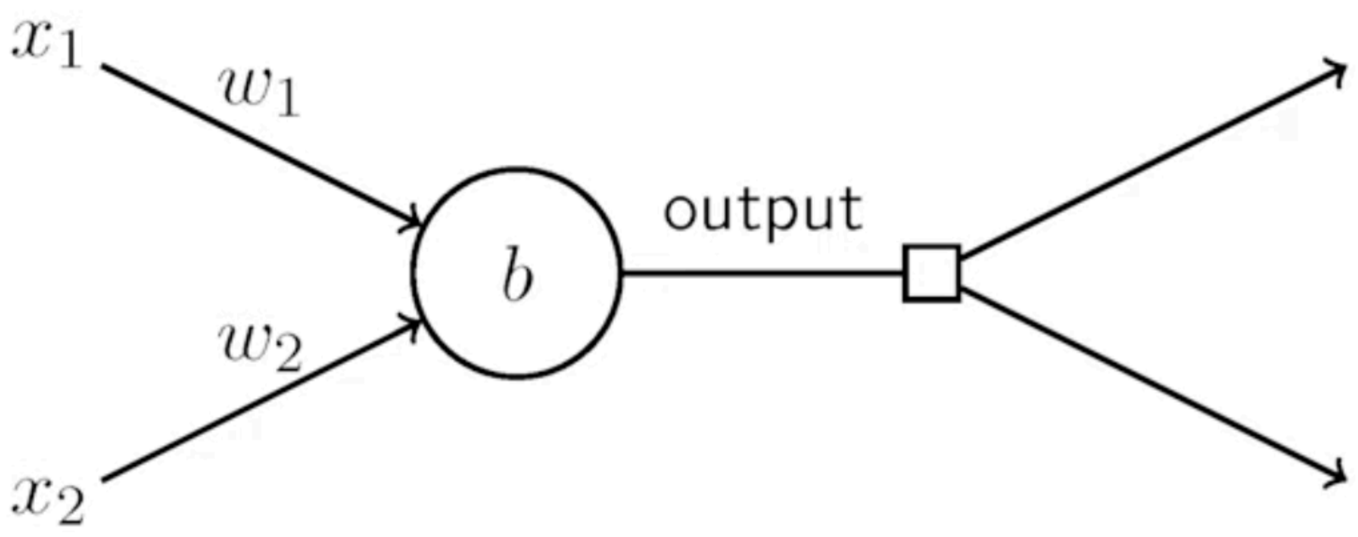
Neural Networks

Perceptron



Neural Networks

Perceptron



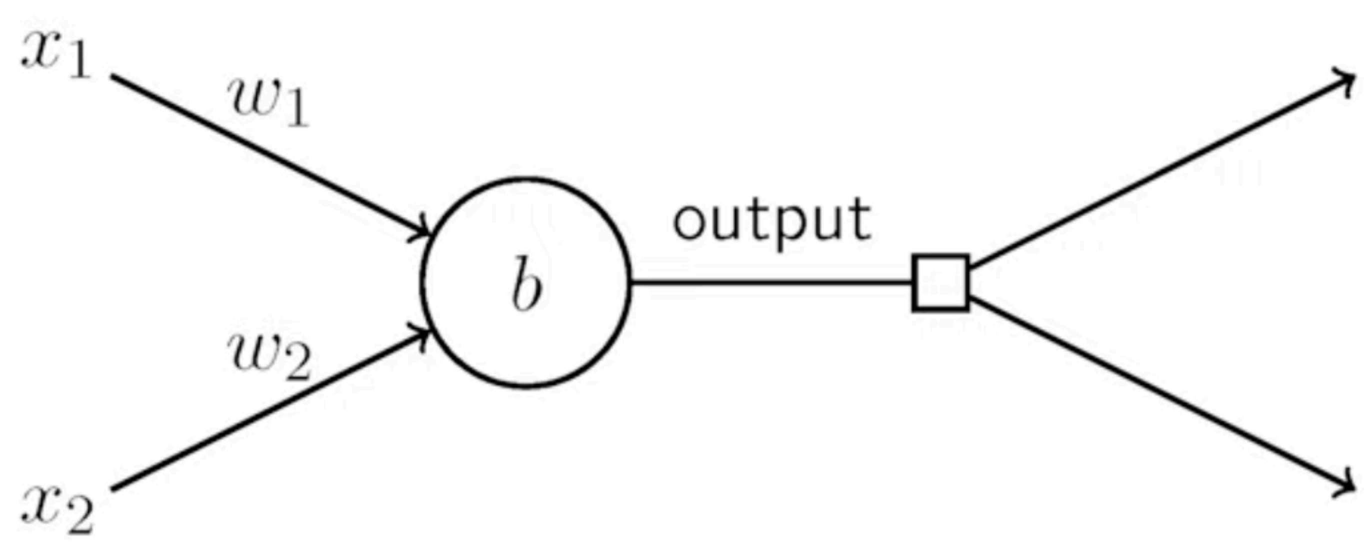
$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq b \\ 1 & \text{if } \sum_j w_j x_j > b \end{cases}$$

The perceptron receives input values (x1, x2) from the previous layer and assigns specific weights (w1, w2).

The perceptron also has a threshold (b) that determines its activation.

Neural Networks

Perceptron



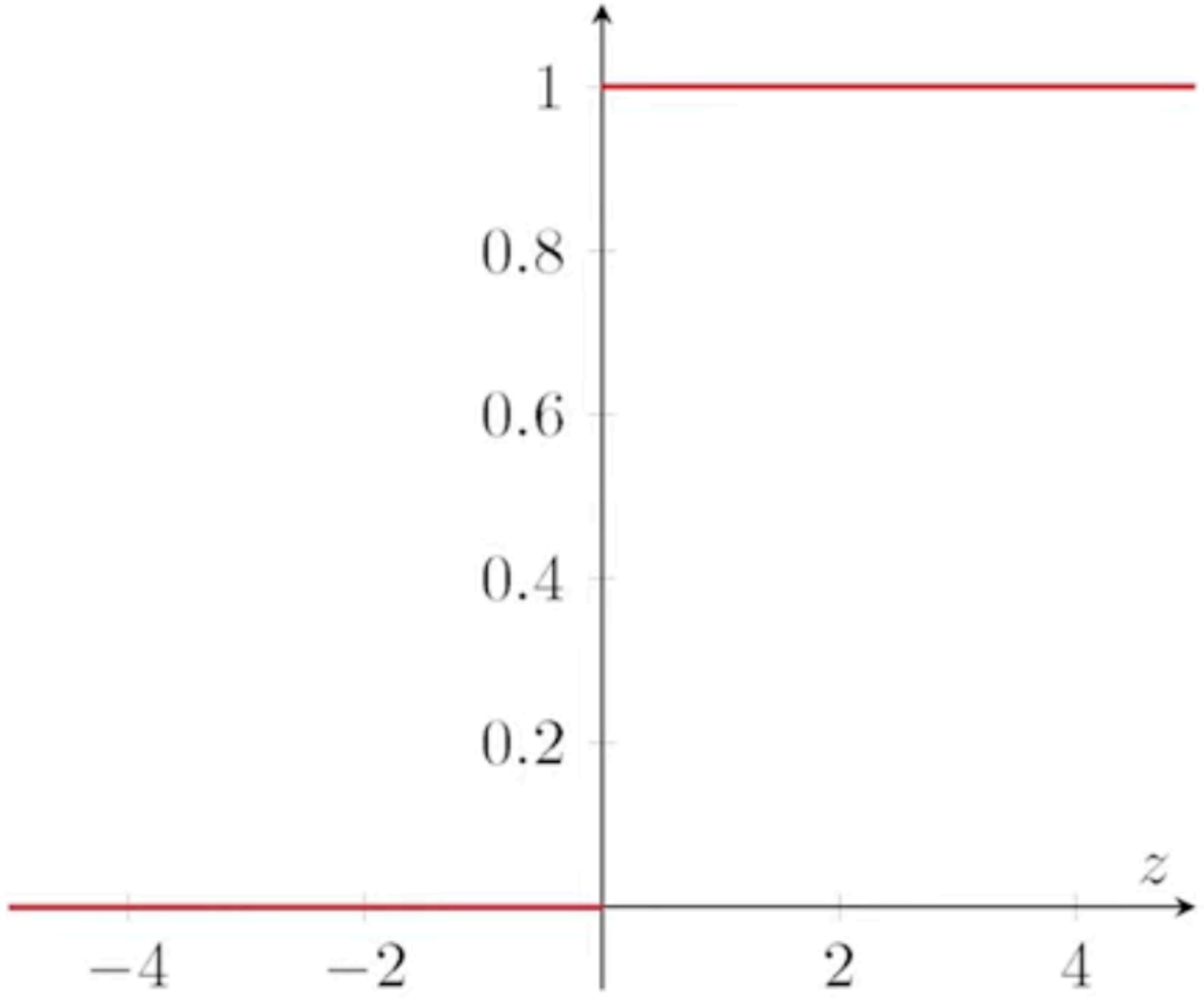
$$\text{output} = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

$$z \equiv \sum_j w_j x_j - b$$

w: weight, b: bias

Neural Networks

Perceptron - Activation Function

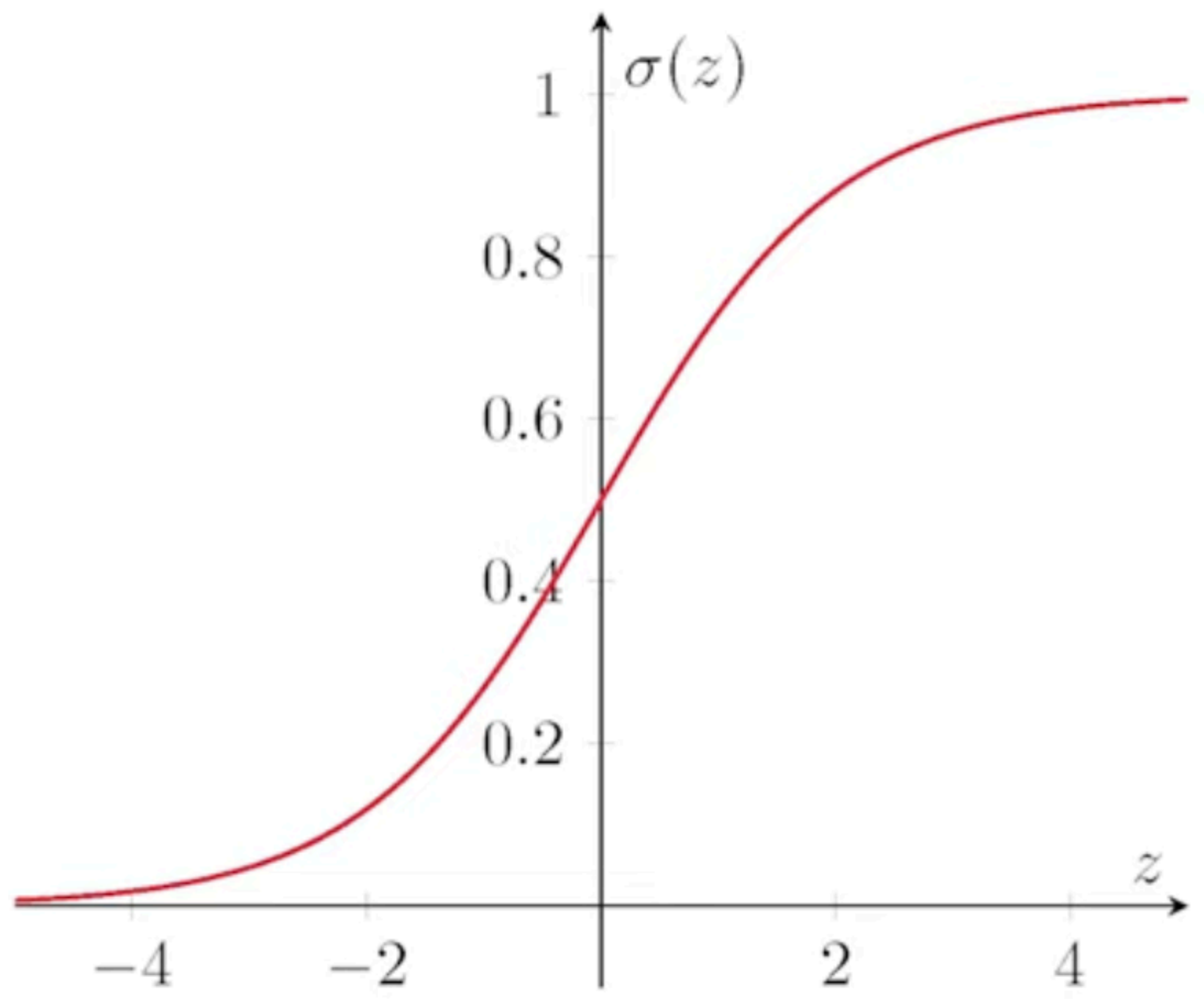


Perceptron uses step function as its activation function

1. Small changes in the weight/bias -> Big changes in the output
2. It outputs only 0 or 1 as the output.

Neural Networks

Perceptron - Activation Function



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

Use sigmoid function!

1. Small changes in the weight/bias -> Small changes in the output
2. It outputs a real number between 0 and 1 as the output.

Neural Networks

Perceptron - Nonlinear Activation Function

A linear function represents a relationship between input and output as a straight line,

whereas a nonlinear function does not.

Let's consider a 3-layer network using the linear function $h(x) = cx$ (where c is a constant) as the activation function.

If $y(x) = h(h(h(x)))$, it becomes $y(x) = c * c * c * x$, which eventually can be expressed as $y(x) = ax$ when $a = c^3$.

In other words, when using a linear function as the activation function,

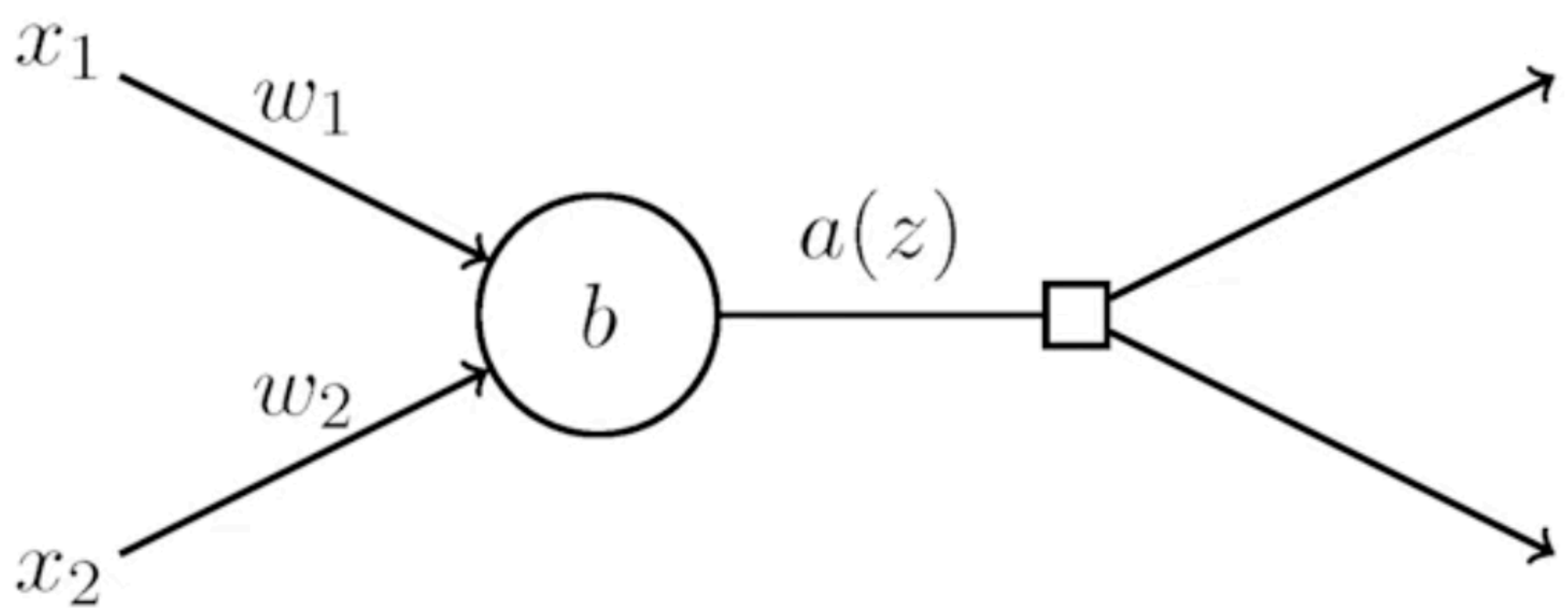
we cannot take advantage of building the network into multiple layers.

Hence, it's not possible to solve difficult problems that require modeling diverse and complex relationships.

The sigmoid function we looked at earlier is a nonlinear function.

Neural Networks

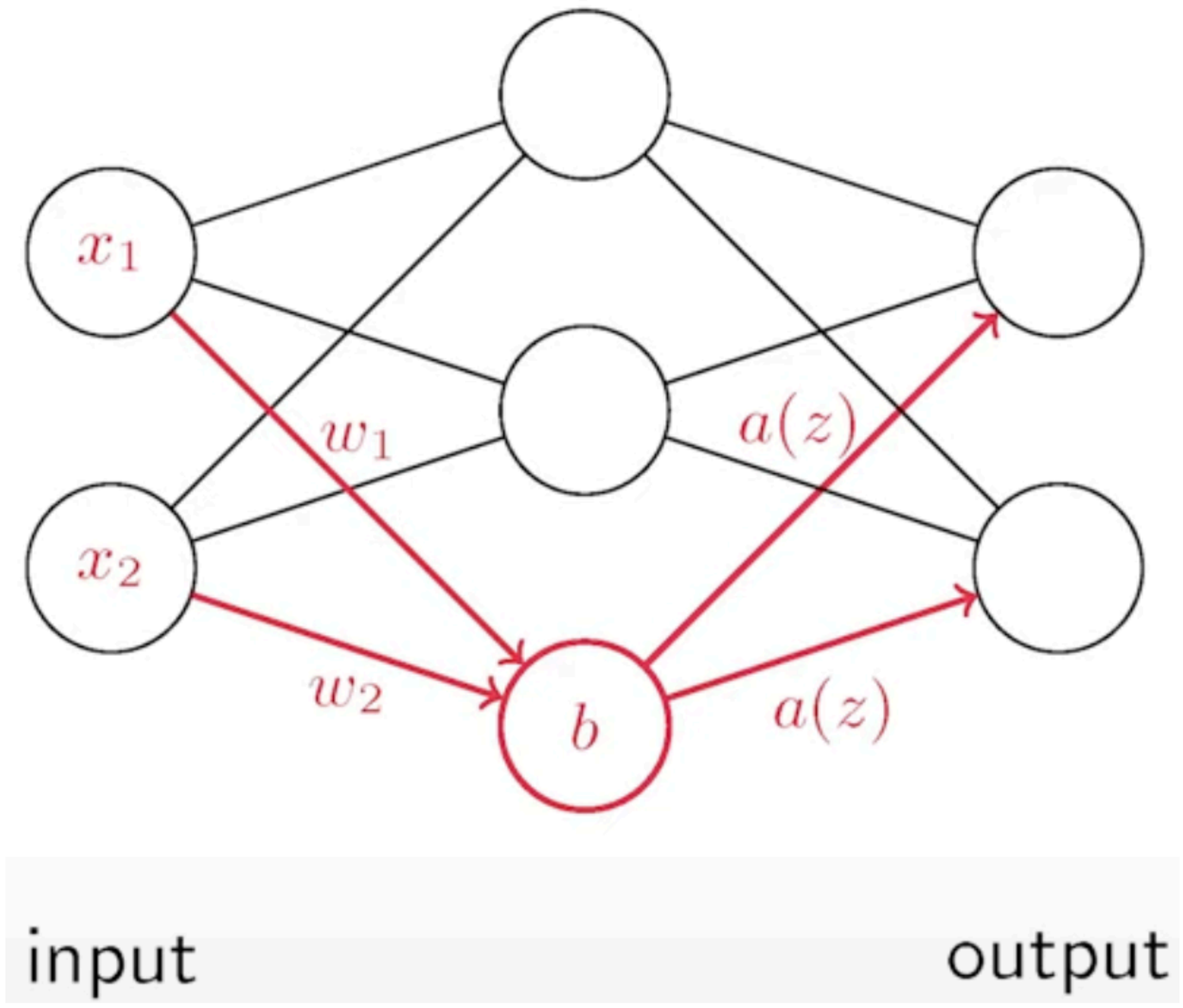
Perceptron



$$a(z) \equiv \sigma(z) = \frac{1}{1 + e^{-z}}$$

Neural Networks

Feed Forward Neural Networks (Multi-layered Perceptron)



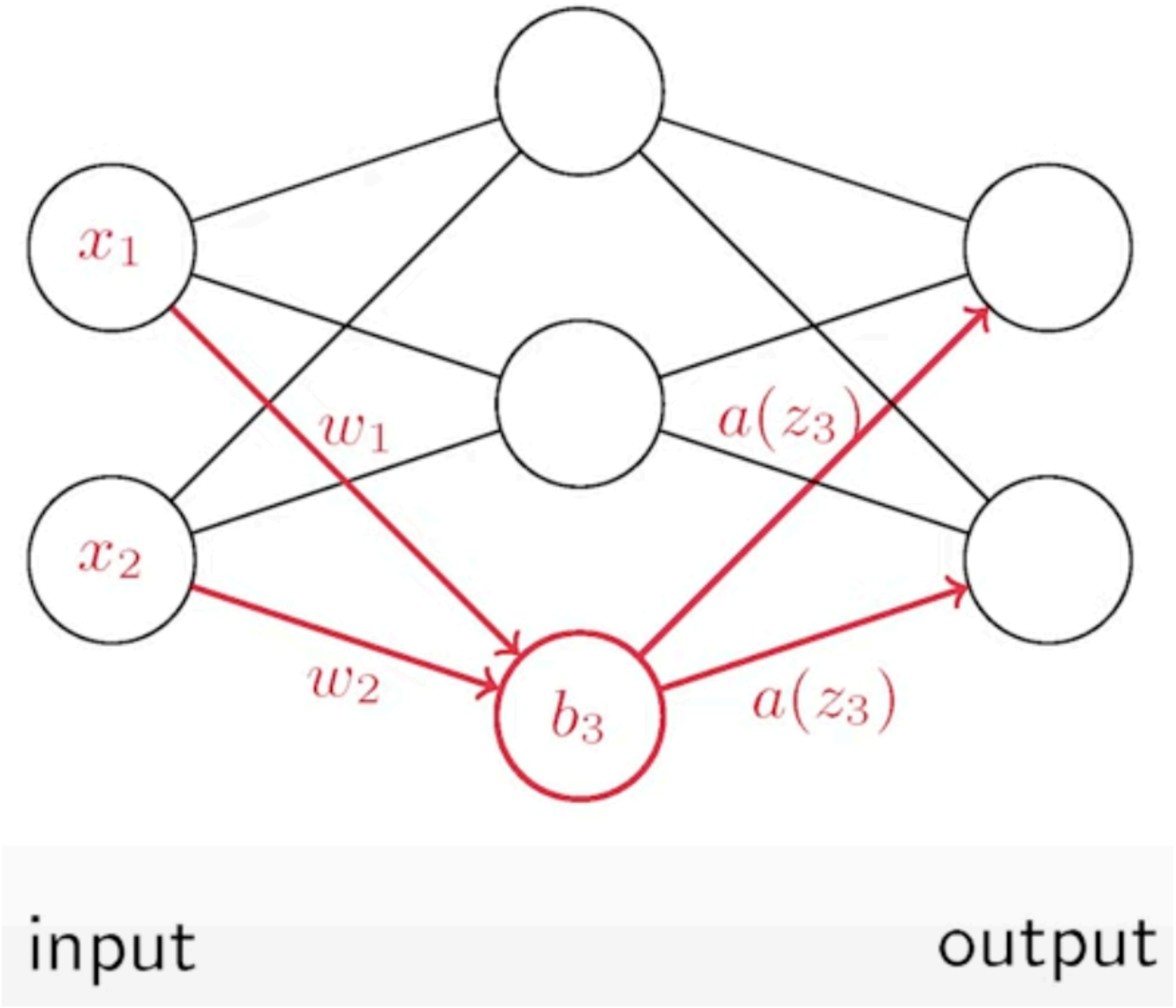
A **feedforward neural network** (**FNN**) is characterized by direction of the flow of information between its layers.

Its flow is uni-directional, meaning that the information in the model flows in only one direction—forward—

from the input nodes, through the hidden nodes (if any) and to the output nodes, without any cycles or loops.

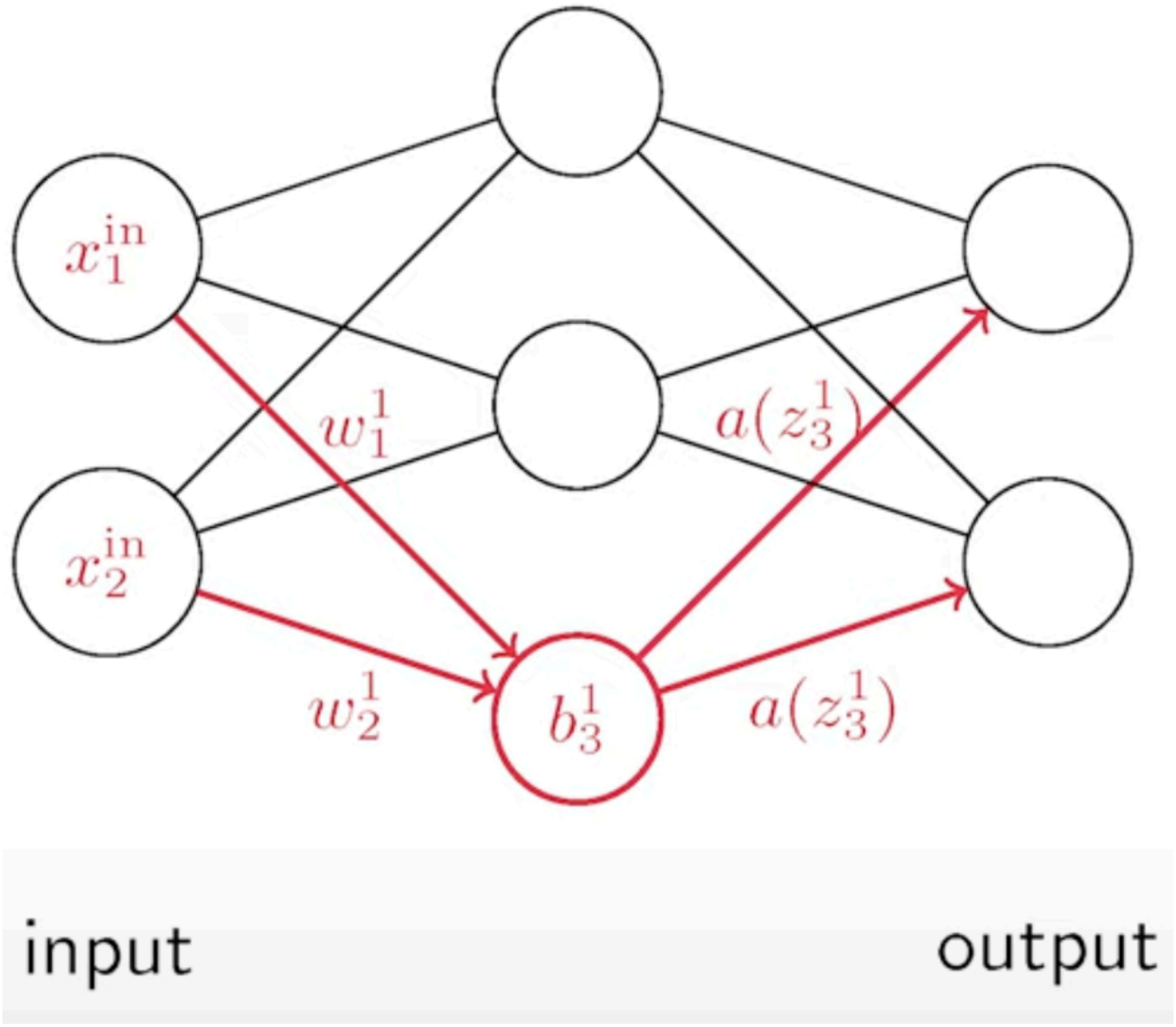
Neural Networks

Feed Forward Neural Networks (Multi-layered Perceptron)



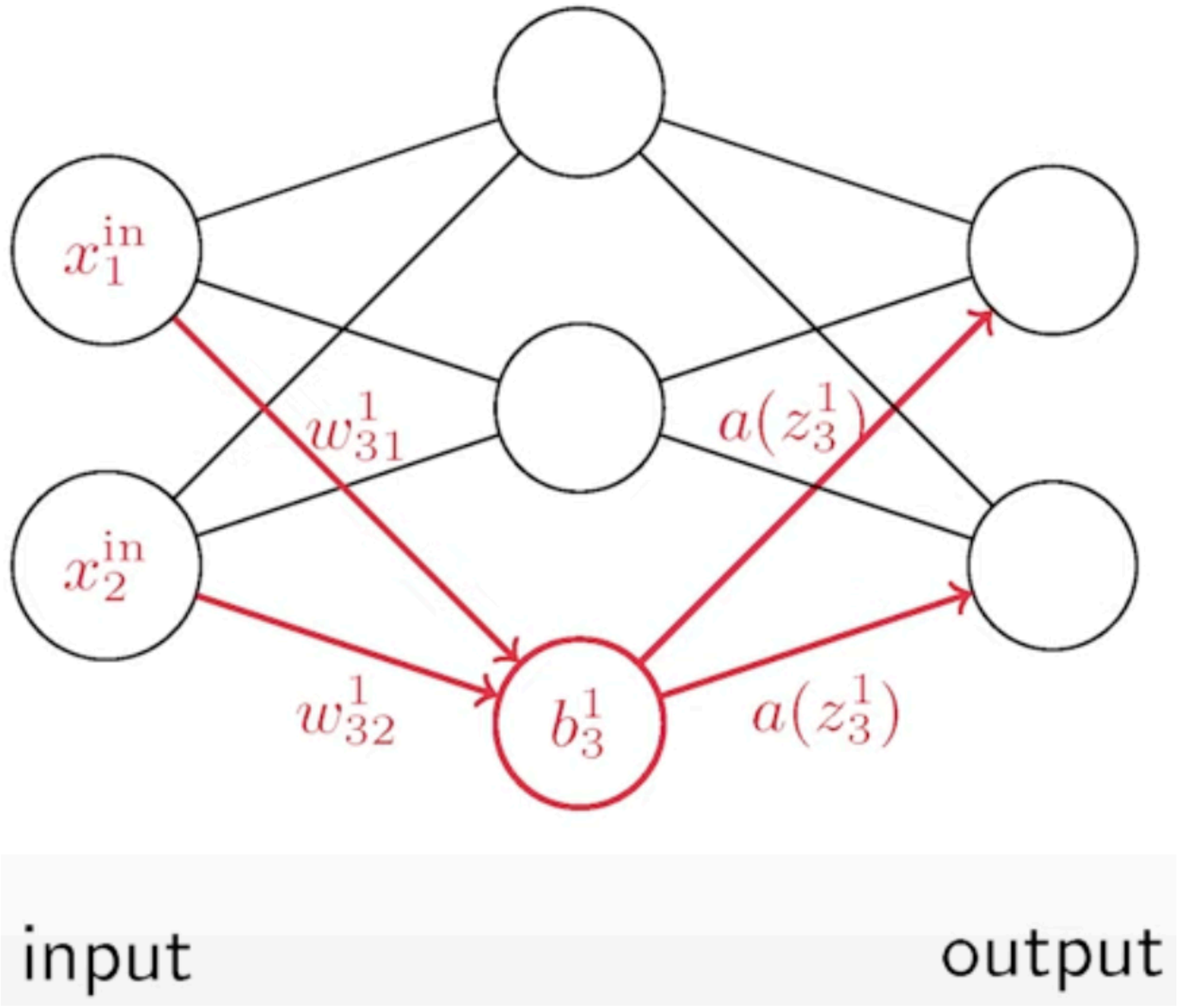
Neural Networks

Feed Forward Neural Networks (Multi-layered Perceptron)



Neural Networks

Feed Forward Neural Networks (Multi-layered Perceptron)



Neural Networks

Training Neural Networks

Training data: N pairs $(x, y(x))$

Cost function:

$$C(w,b) \equiv \frac{1}{N} \sum_x \frac{||y(x) - a^{\text{out}}(x)||^2}{2}$$

Neural Networks

Training Neural Networks

Training data: N pairs $(x, y(x))$

Cost function:

$$C(w, b) \equiv \frac{1}{N} \sum_{x} \frac{||y(x) - a^{\text{out}}(x)||^2}{2}$$

Goal: Minimize $C(w, b) \rightarrow$ **Gradient descent**

Neural Networks

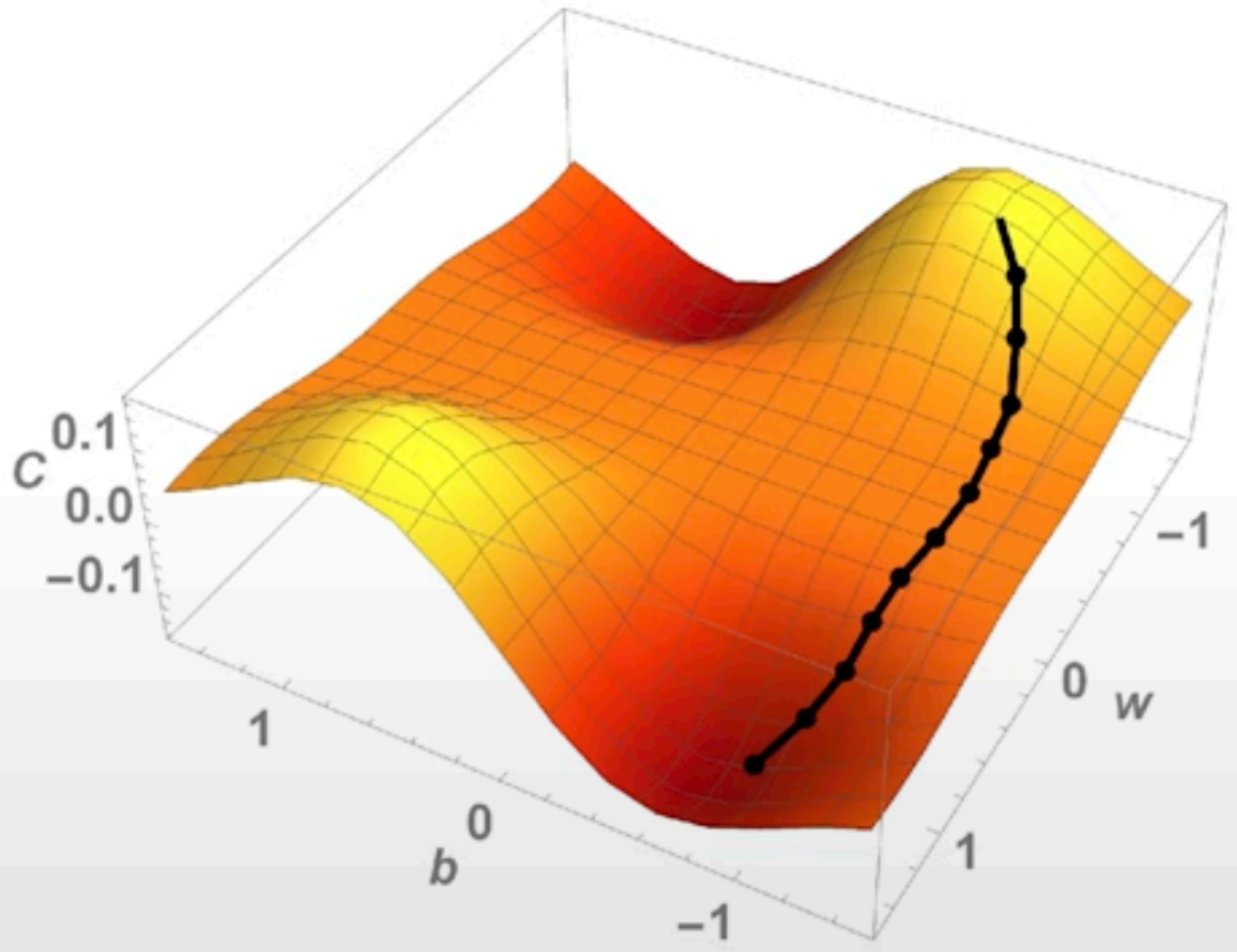
Training Neural Networks

Training data: N pairs $(x, y(x))$

Cost function:

$$C(w,b) \equiv \frac{1}{N}\sum_x \frac{||y(x) - a^{\text{out}}(x)||^2}{2}$$

Goal: Minimize $C(w,b) \rightarrow$ **Gradient descent**



$$w^l_{jk} \rightarrow w^l_{jk} - \eta \frac{\partial C}{\partial w^l_{jk}}$$

$$b^l_j \rightarrow b^l_j - \eta \frac{\partial C}{\partial b^l_j}$$

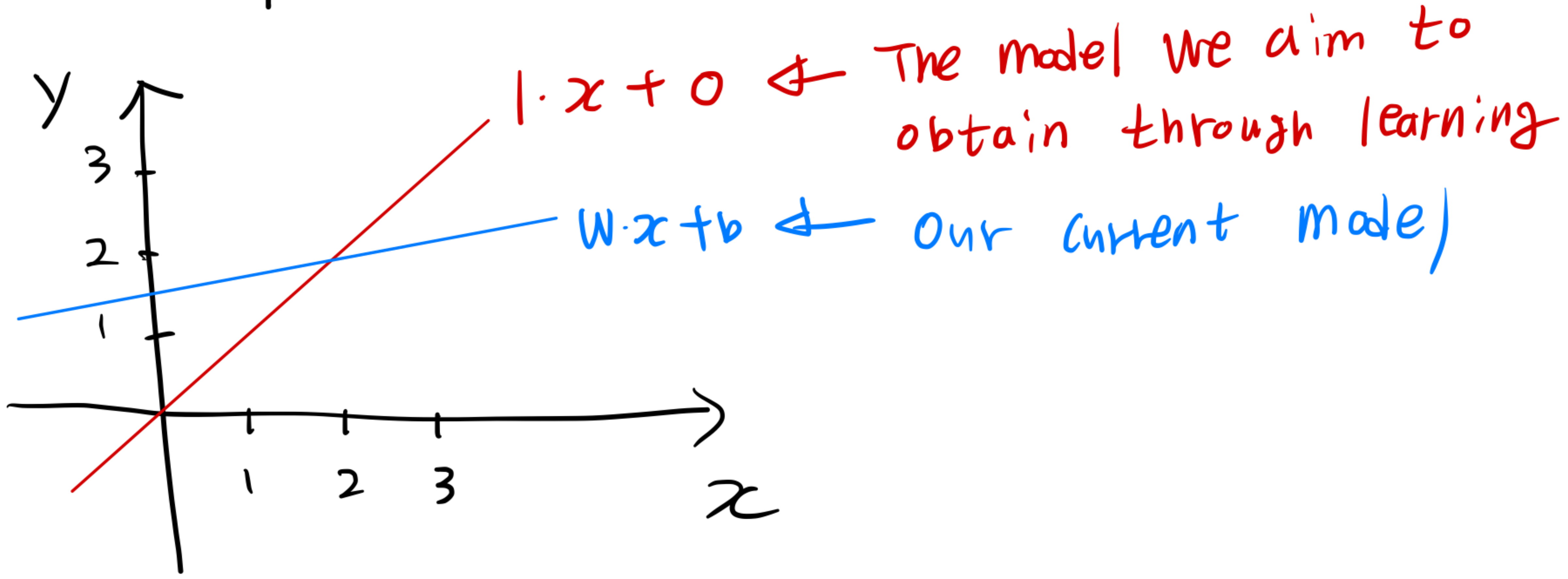
Neural Networks

Training Neural Networks

Linear Regression

x	$y(x)$
1	1
2	2
3	3

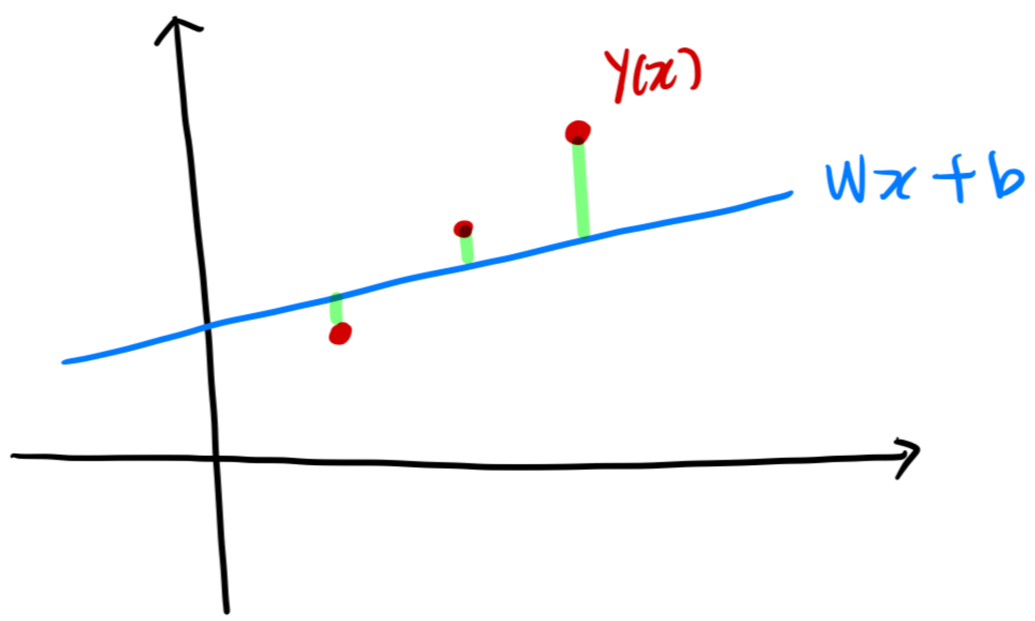
← Training Data



Neural Networks

Training Neural Networks

Cost function



Difference (Cost) $Y(x) - a^{out}(x)$

To apply a greater penalty as the difference grows larger, the error value is squared.
Also, since there can be positive/negative error values, squaring is done to prevent cancellation when added together.

$$Cost \equiv \frac{1}{N} \sum_x \left(Wx + b - Y(x) \right)^2$$

Neural Networks

Training Neural Networks

Since the input and the ground-truth are fixed, what influences the cost are the weight (w) and bias (b).

To obtain the w and b values that minimize the cost, adjustments need to be made to w and b using gradient descent.

For simplicity, let's disregard b for a moment.

→ $W \cdot x$ /

x	$Y(x)$
1	1
2	2
3	3

→ Let's calculate the cost for various values of W

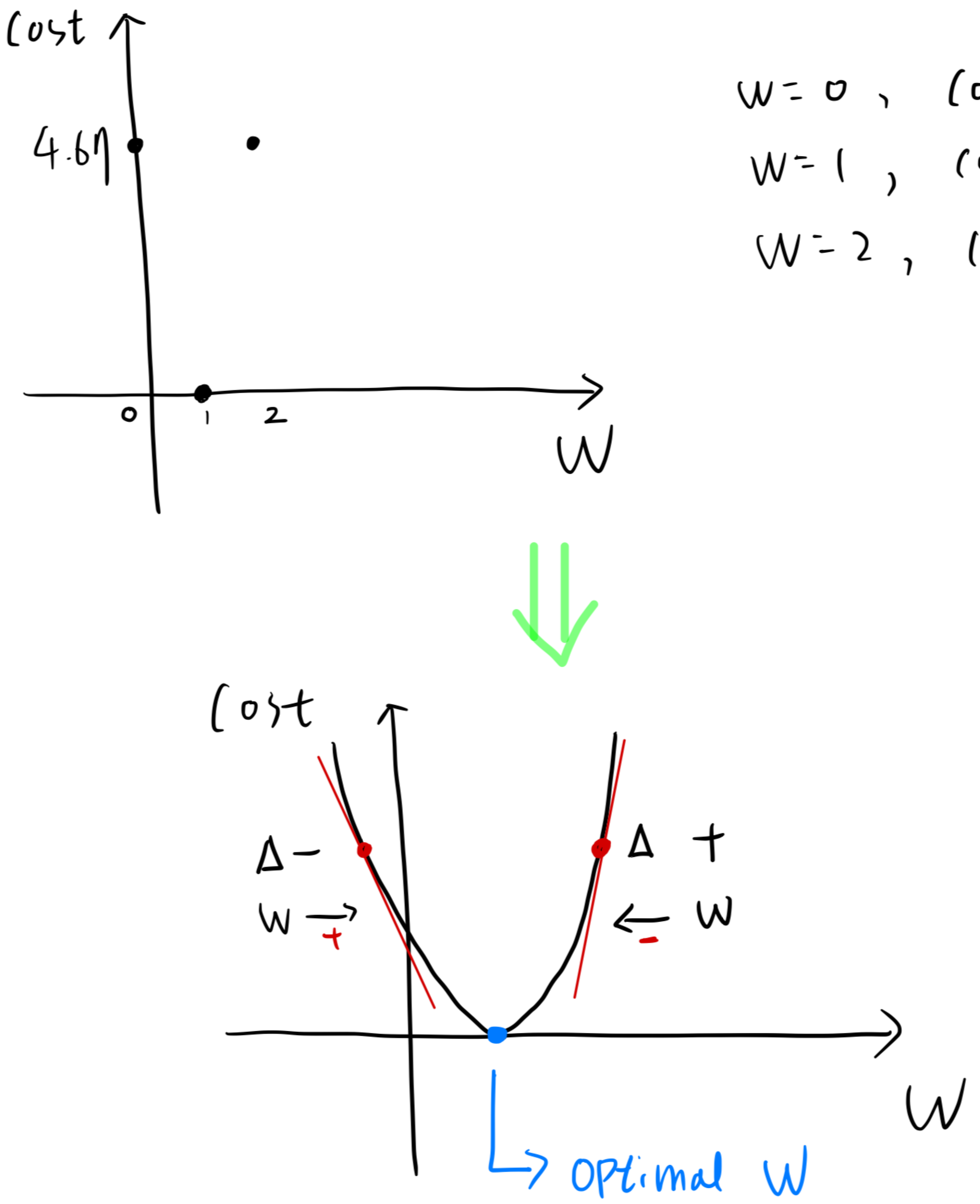
$W=1$, $Cost = \frac{1}{3}\{(1\cdot1-1)^2+(1\cdot2-2)^2+(1\cdot3-3)^2\}=0$

$W=0$, $Cost = \frac{1}{3}\{(0\cdot1-1)^2+(0\cdot2-2)^2+(0\cdot3-3)^2\}=4.67$

$W=2$, $Cost = \frac{1}{3}\{(2\cdot1-1)^2+(2\cdot2-2)^2+(2\cdot3-3)^2\}=4.67$

Neural Networks

Training Neural Networks



w = 0 , cost = 4.67
w = 1 , cost = 0
w = 2 , cost = 4.67

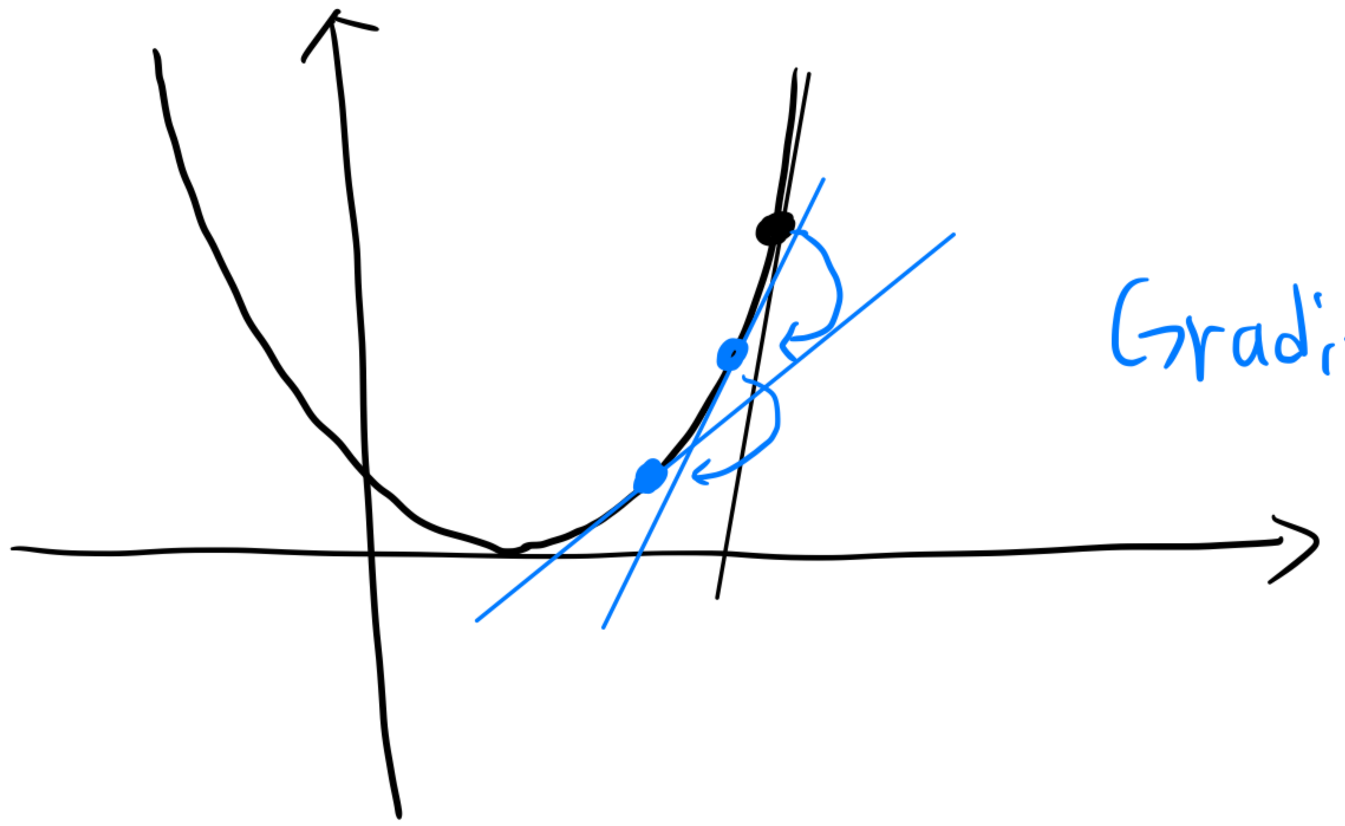
Neural Networks

Training Neural Networks

$$Cost = \frac{1}{2N} \sum_{x} (Wx - Y(x))^2$$

$$W_{update} = W - \alpha \cdot \frac{\partial Cost}{\partial W}$$

$$= W - \alpha \cdot \frac{1}{N} \sum_{x} (Wx - Y(x)) \cdot x$$



Gradient descent

Quantum Neural Networks

nature COMMUNICATIONS

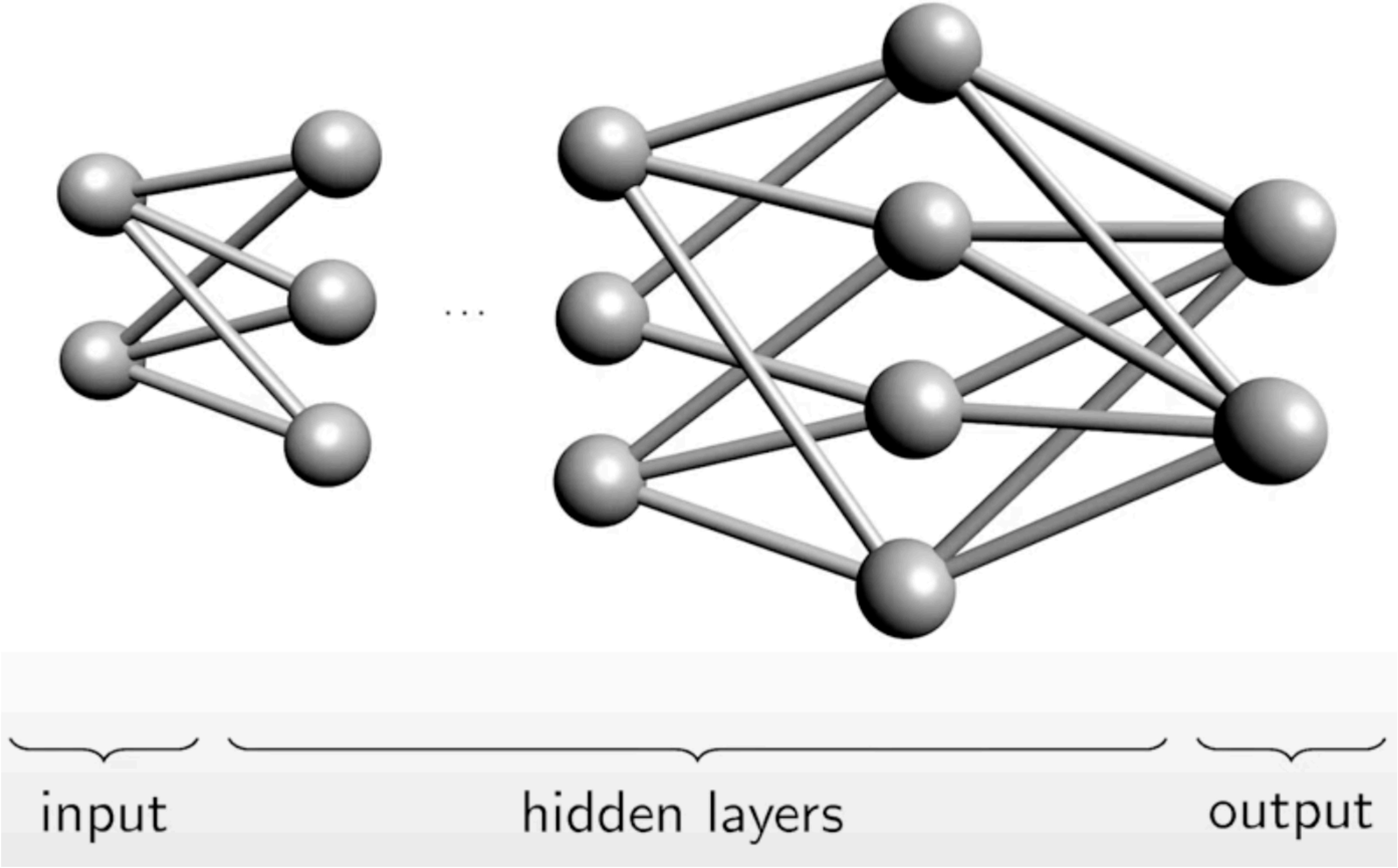
ARTICLE

https://doi.org/10.1038/s41467-020-14454-2 OPEN

Training deep quantum neural networks

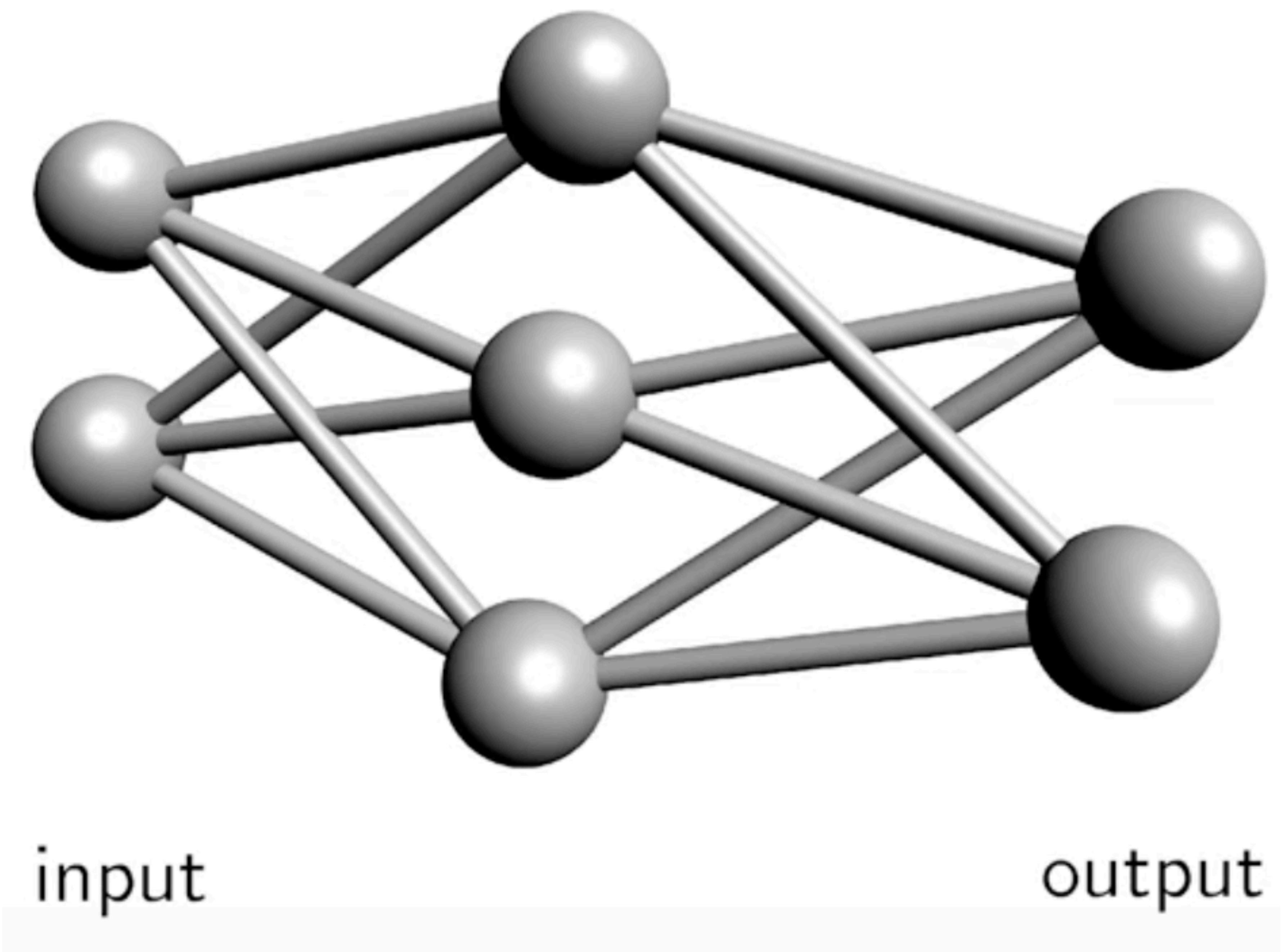
Kerstin Beer[1*], Dmytro Bondarenko[1], Terry Farrelly[1,2], Tobias J. Osborne[1], Robert Salzmann[1,3], Daniel Scheiermann[1] & Ramona Wolf[1]

Quantum Neural Networks



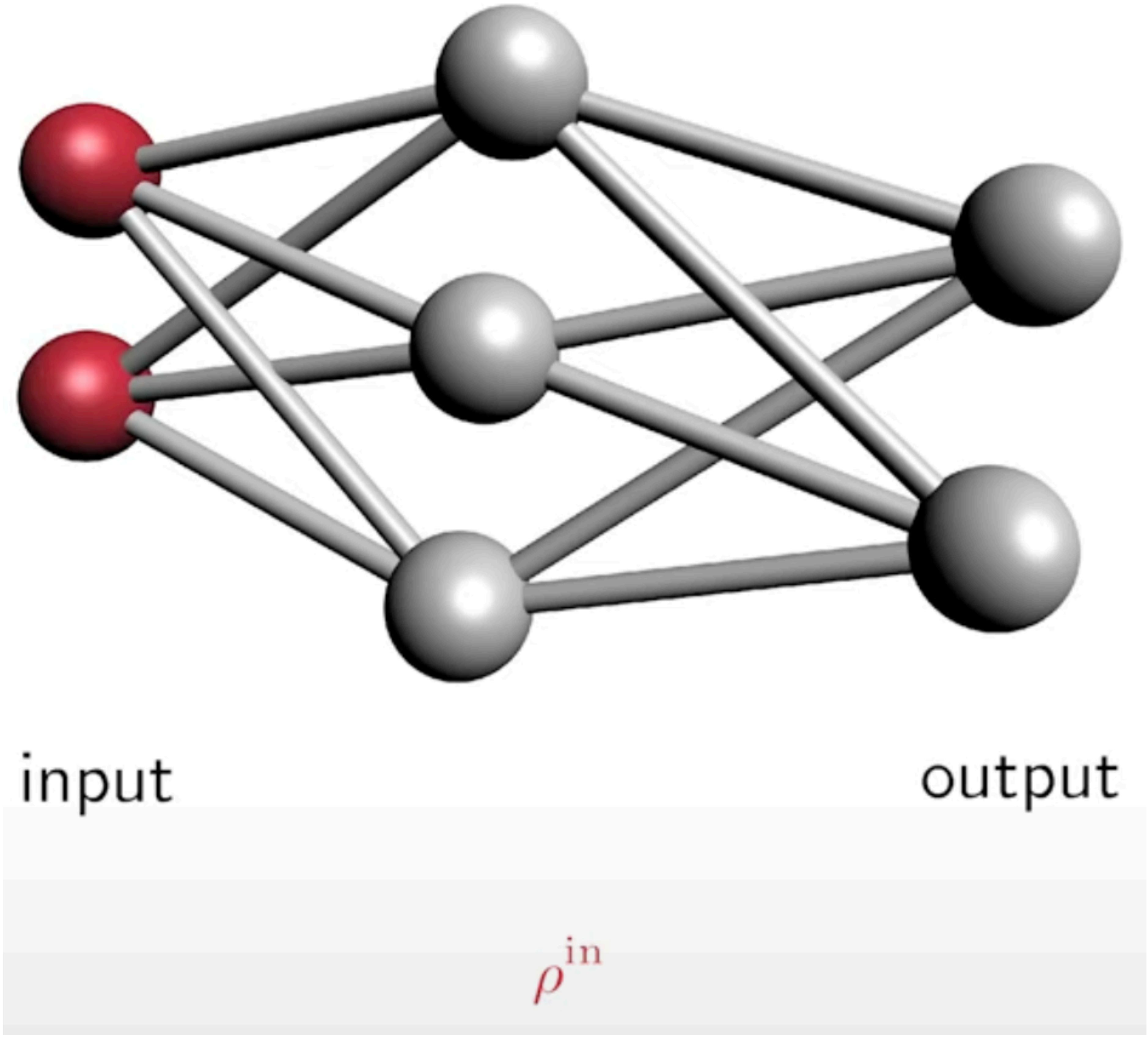
Quantum Neural Networks

Feed Forward



Quantum Neural Networks

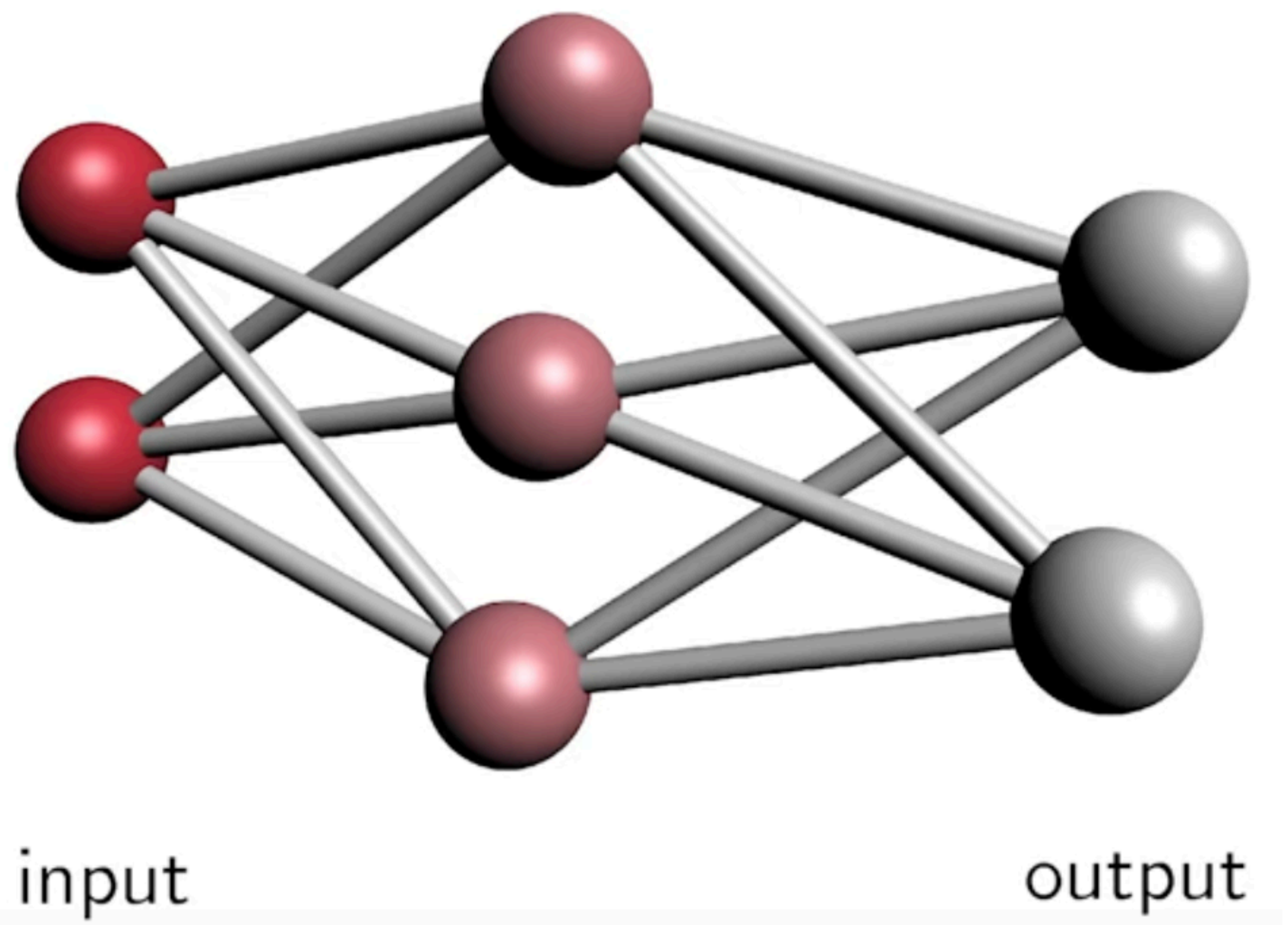
Feed Forward



State of input qubit

Quantum Neural Networks

Feed Forward

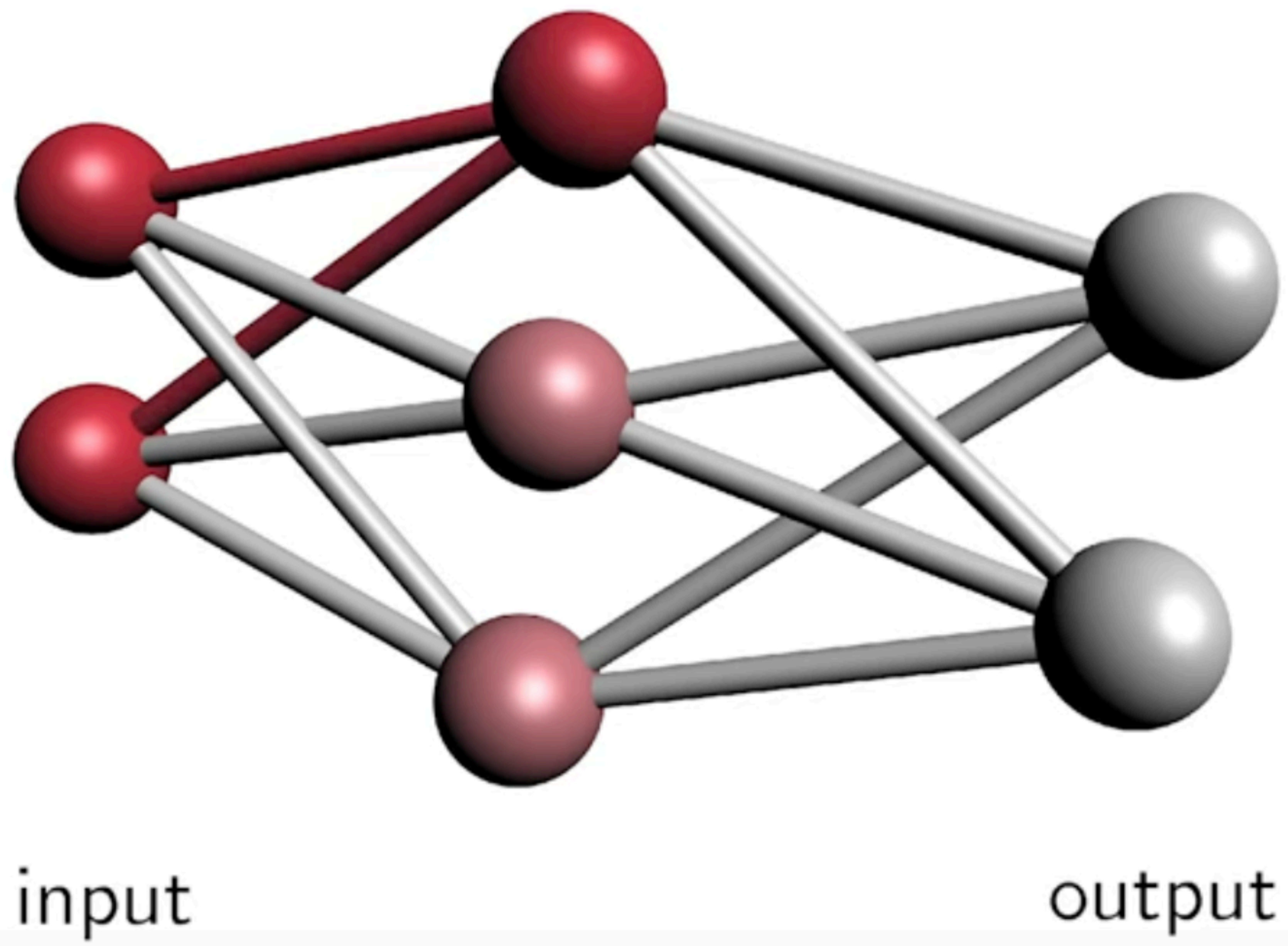


$$\rho^{\mathrm{in}} \otimes |000\rangle_1\langle 000|$$

Initialize hidden state

Quantum Neural Networks

Feed Forward

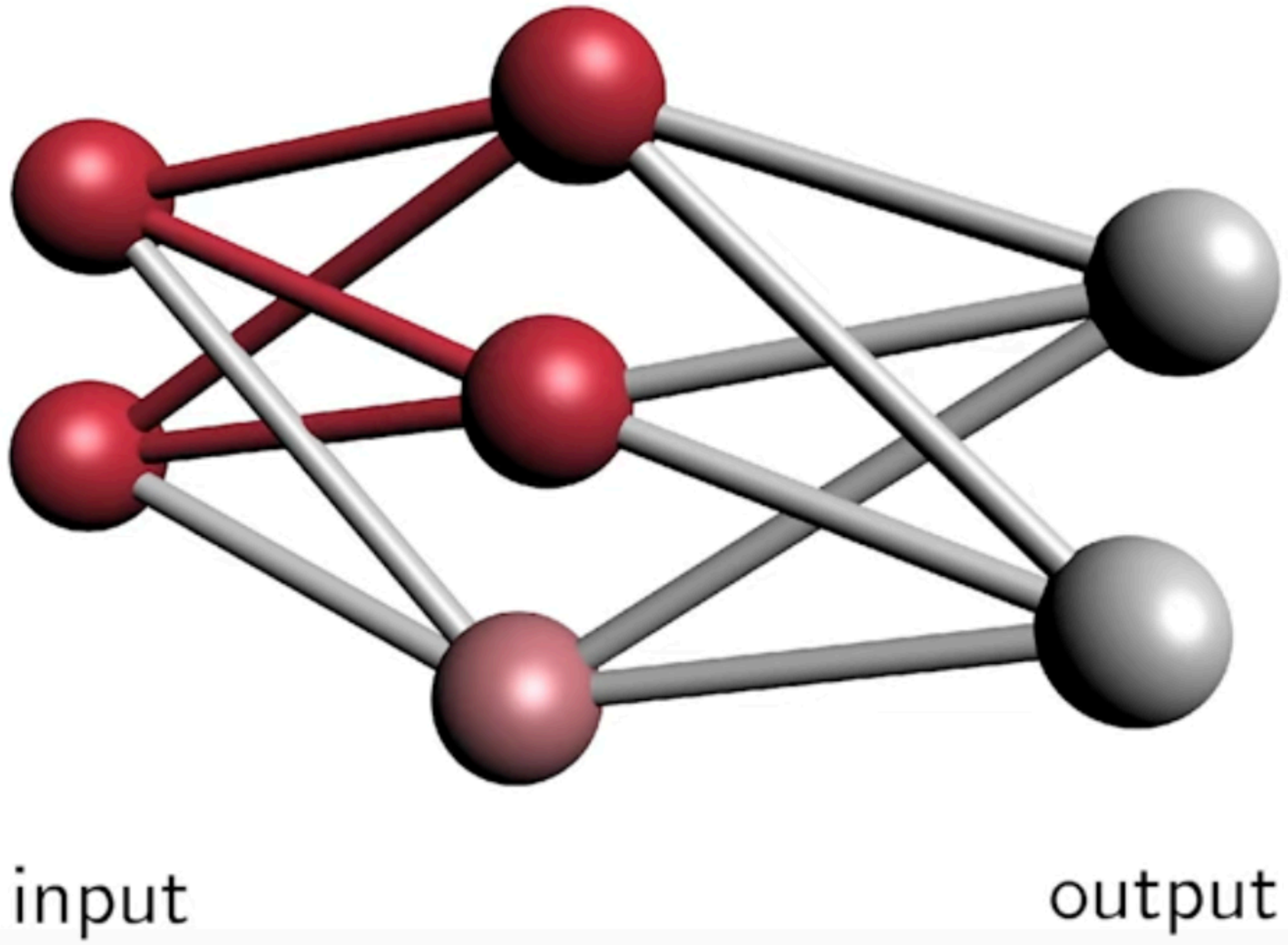


$$U_1^1\left(\rho^{\text{in}} \otimes |000\rangle_1\langle 000|\right)U_1^{1\dagger}$$

Apply unitary matrix

Quantum Neural Networks

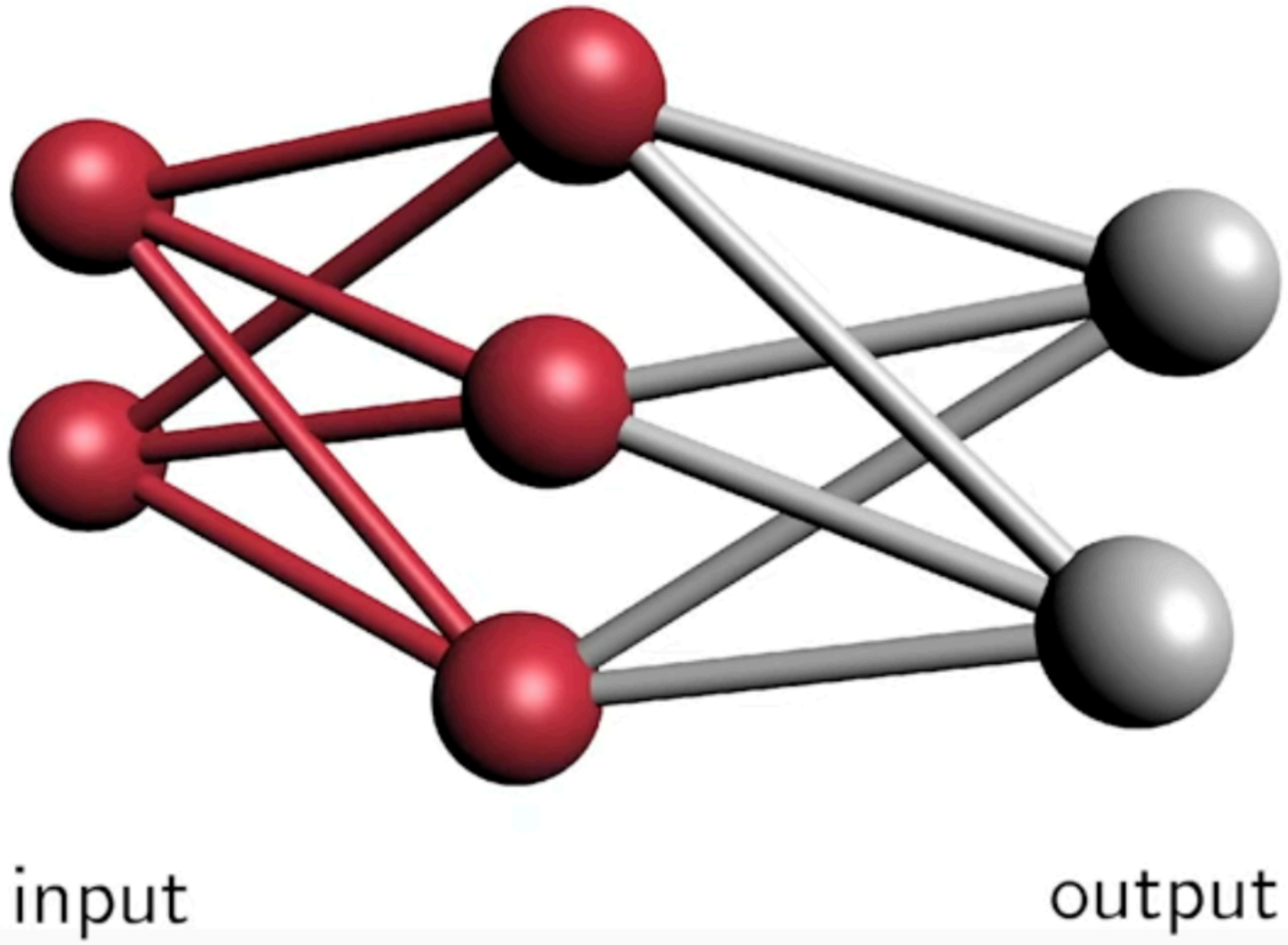
Feed Forward



$$U_2^1 U_1^1 \left(\rho^{\text{in}} \otimes |000\rangle_1 \langle 000|\right) U_1^{1\dagger} U_2^{1\dagger}$$

Quantum Neural Networks

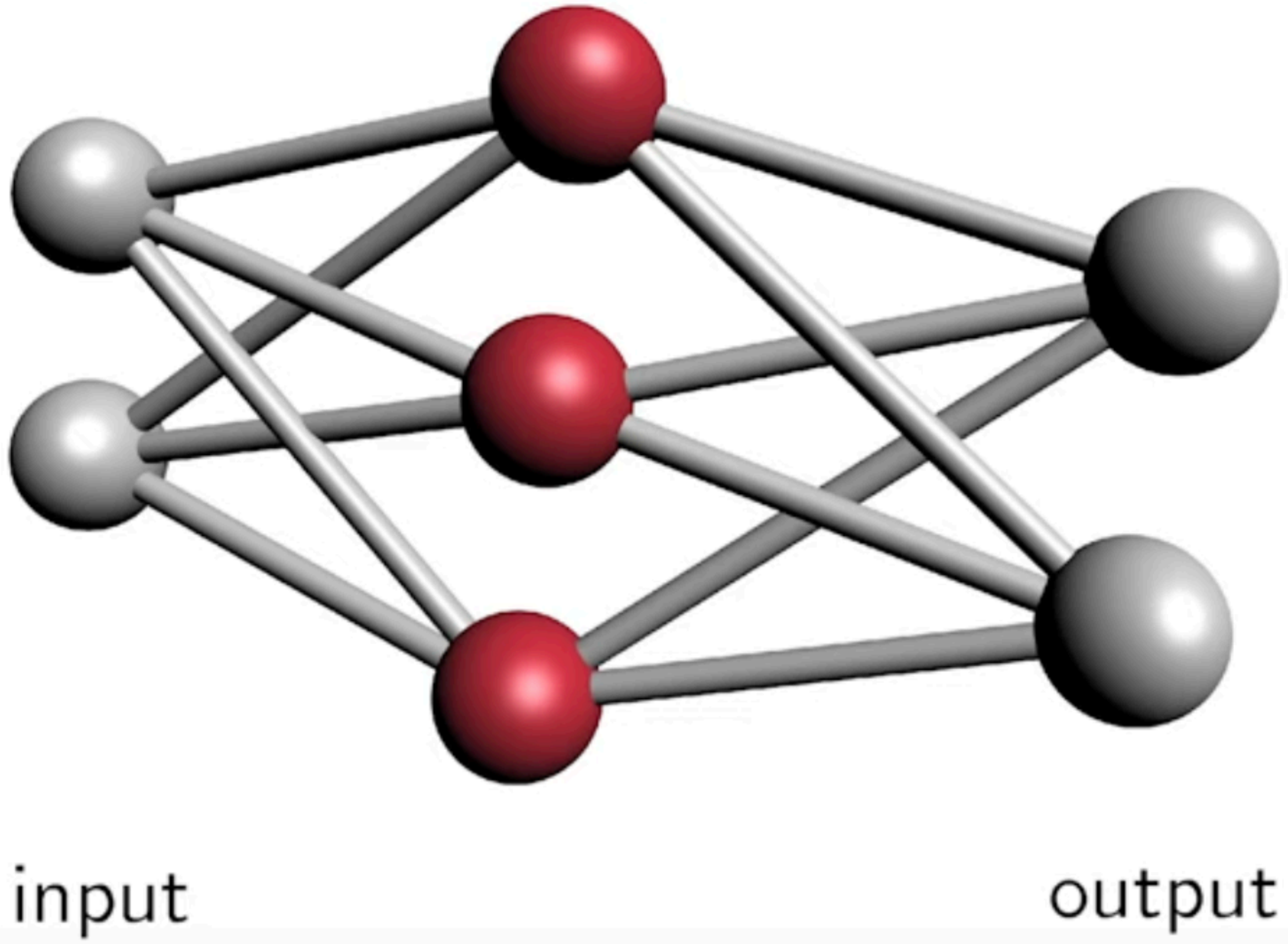
Feed Forward



$$U_3^1 U_2^1 U_1^1 \left(\rho^{\mathrm{in}} \otimes |000\rangle_1 \langle 000|\right) U_1^{1\dagger} U_2^{1\dagger} U_3^{1\dagger}$$

Quantum Neural Networks

Feed Forward

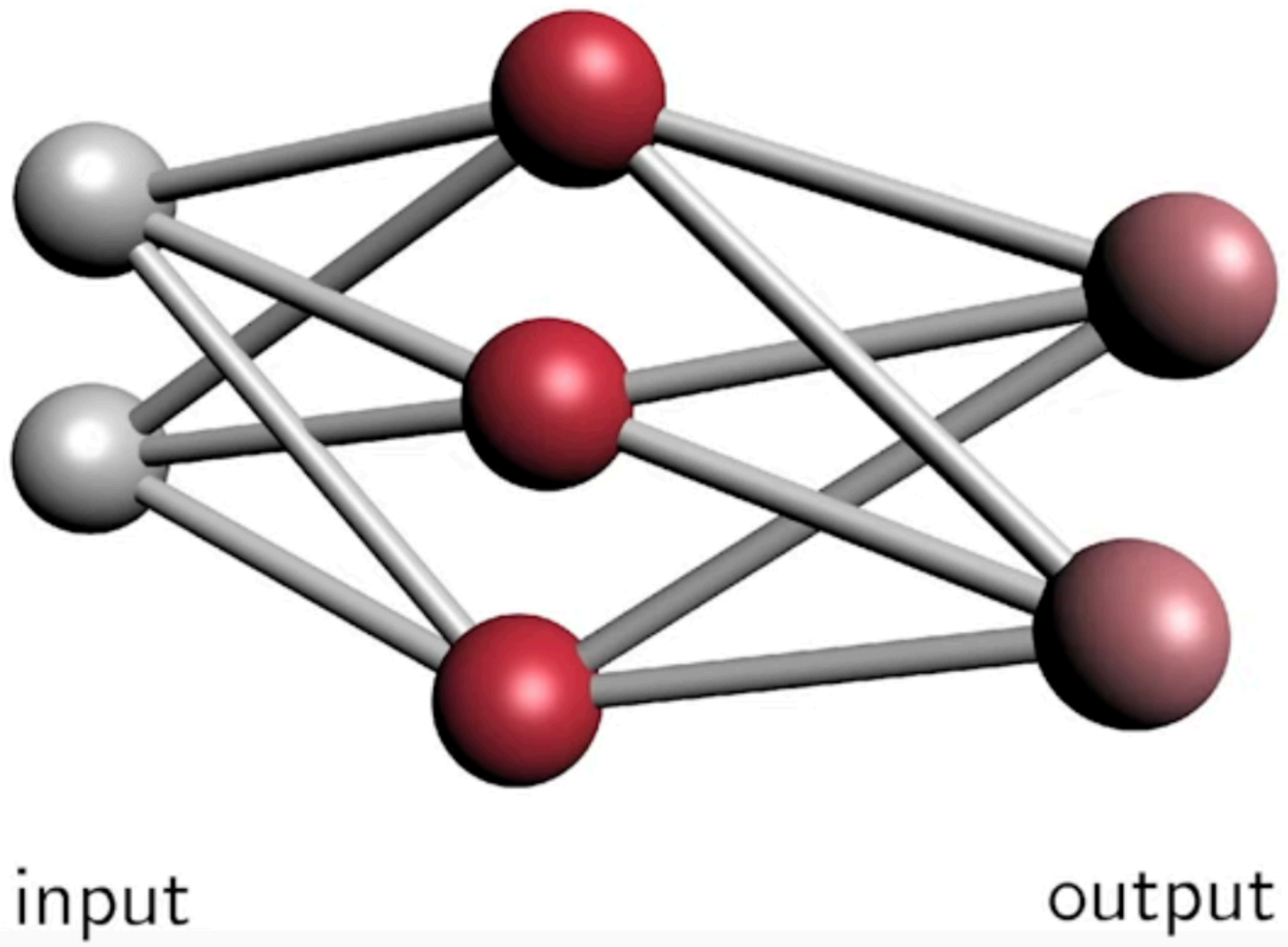


$$\mathrm{tr}_{\mathrm{in}}\left(U_3^1 U_2^1 U_1^1 \left(\rho^{\mathrm{in}} \otimes |000\rangle_1\langle 000|\right) U_1^{1\dagger} U_2^{1\dagger} U_3^{1\dagger}\right)$$

Keep only the state of the hidden layer (trace out the input layer)

Quantum Neural Networks

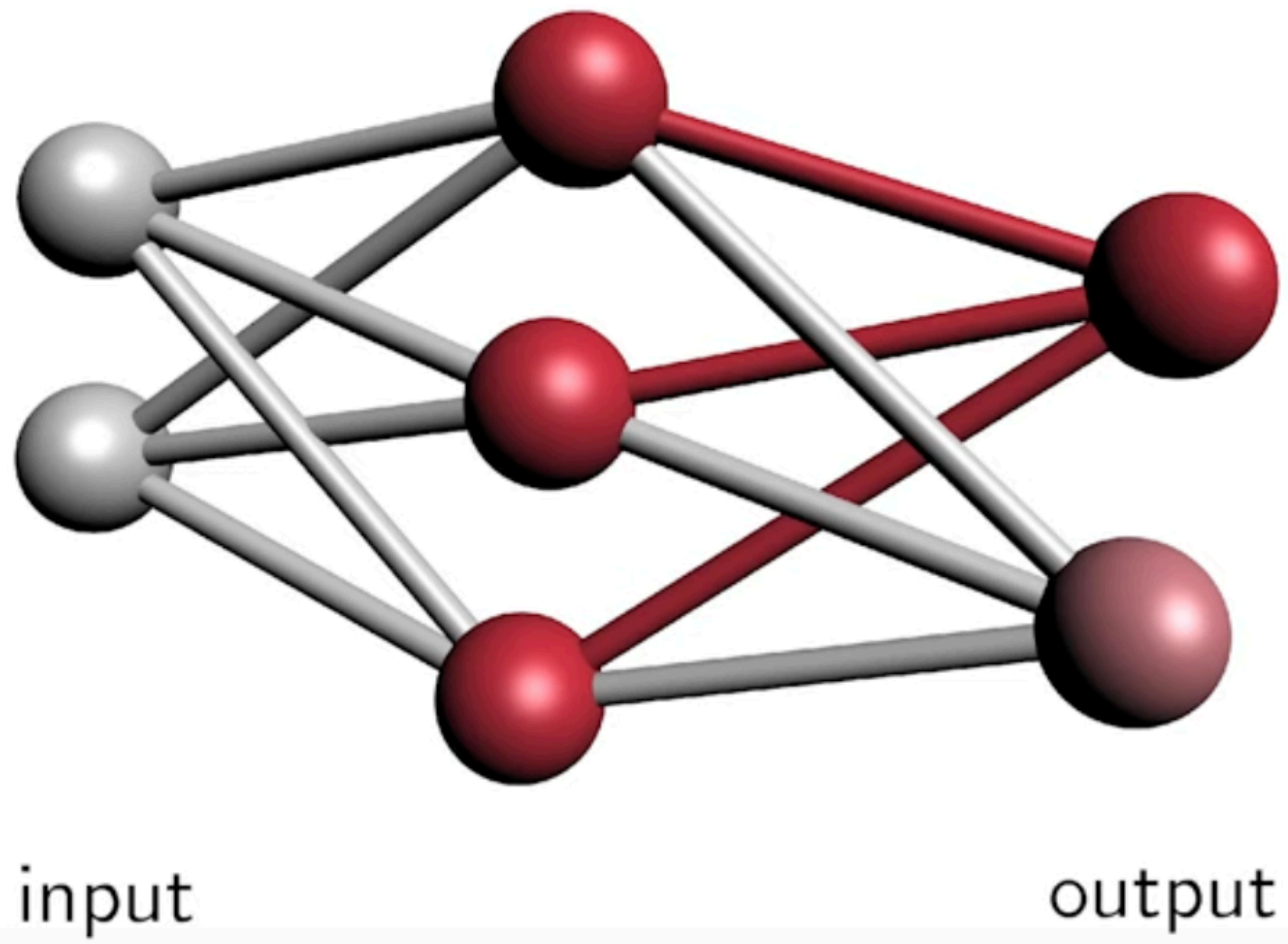
Feed Forward



$$\mathrm{tr}_{\mathrm{in}}\left(U_3^1 U_2^1 U_1^1 \left(\rho^{\mathrm{in}} \otimes |000\rangle_1 \langle 000|\right) U_1^{1\dagger} U_2^{1\dagger} U_3^{1\dagger}\right) \otimes |00\rangle_{\mathrm{out}} \langle 00|$$

Quantum Neural Networks

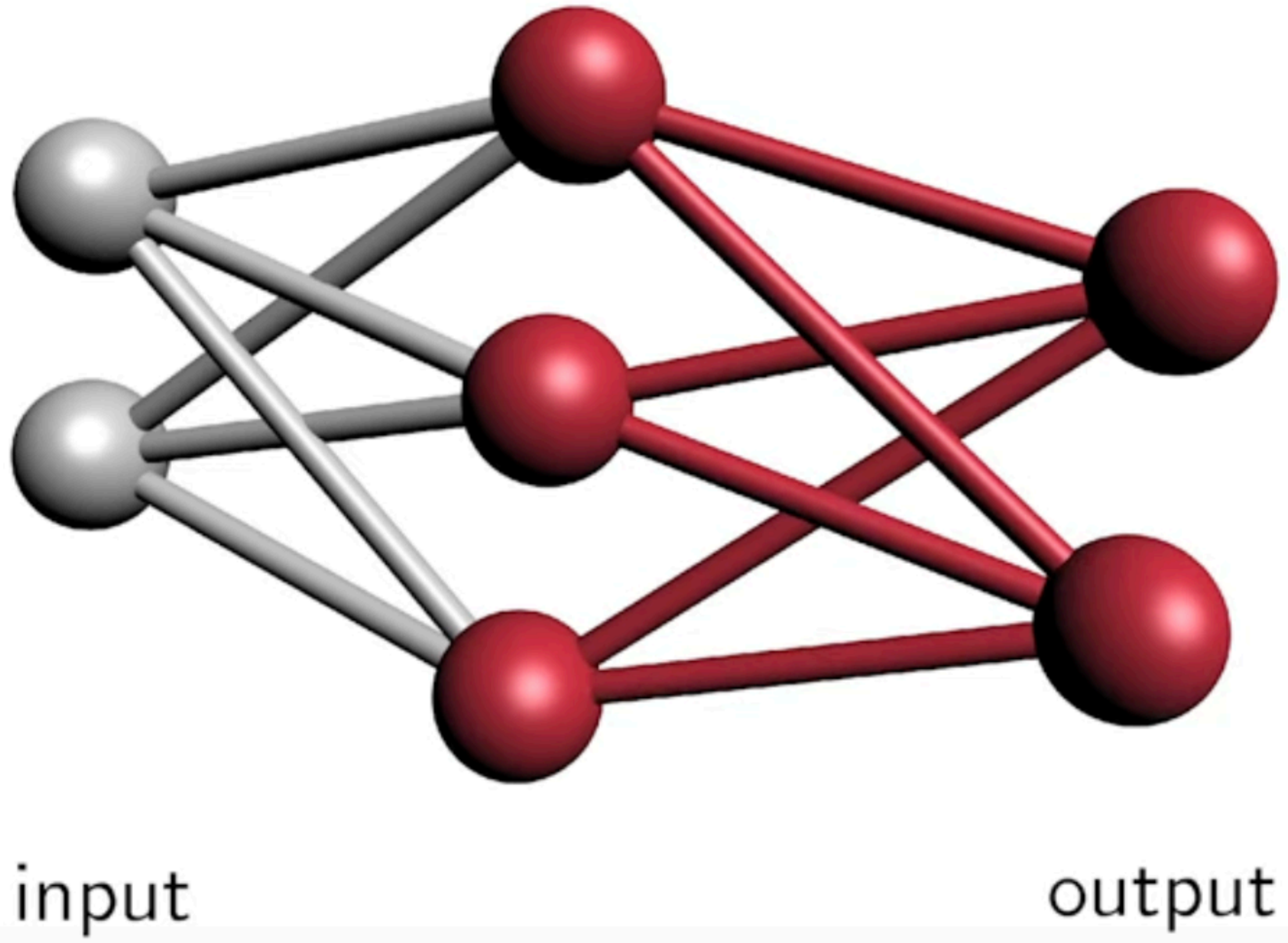
Feed Forward



$$U_1^{\mathrm{out}}\left(\mathrm{tr}_{\mathrm{in}}\left(U_3^1 U_2^1 U_1^1\left(\rho^{\mathrm{in}} \otimes |000\rangle_1\langle 000|\right)U_1^{1\dagger} U_2^{1\dagger} U_3^{1\dagger}\right) \otimes |00\rangle_{\mathrm{out}}\langle 00|\right)U_1^{\mathrm{out}\dagger}$$

Quantum Neural Networks

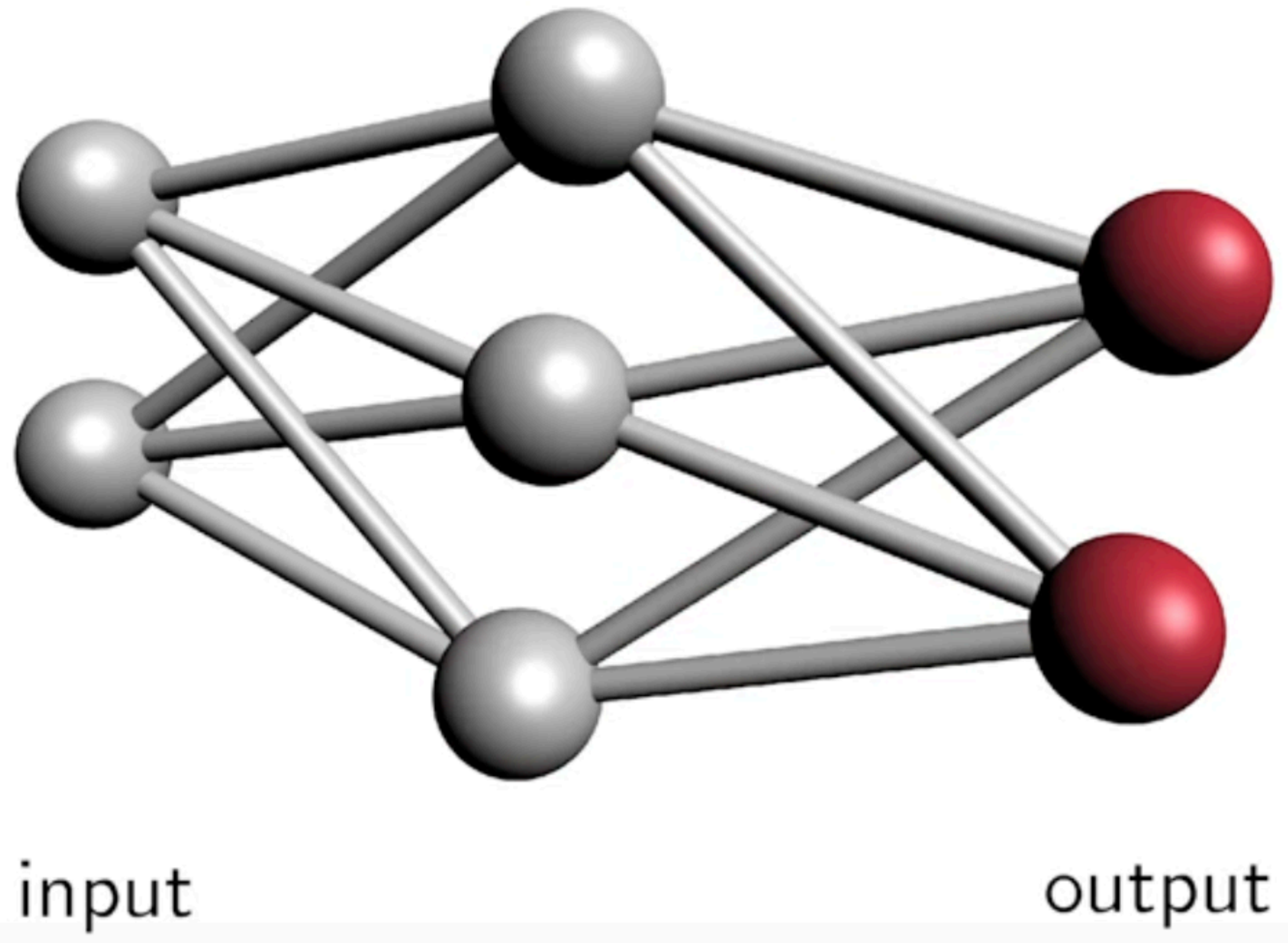
Feed Forward



$$U_2^{\text{out}} U_1^{\text{out}} \left(\text{tr}_{\text{in}} \left(U_3^1 U_2^1 U_1^1 \left(\rho^{\text{in}} \otimes |000\rangle_1 \langle 000| \right) U_1^{1\dagger} U_2^{1\dagger} U_3^{1\dagger} \right) \otimes |00\rangle_{\text{out}} \langle 00| \right) U_1^{\text{out}\dagger} U_2^{\text{out}\dagger}$$

Quantum Neural Networks

Feed Forward



$$\mathrm{tr}_1\left(U_2^{\mathrm{out}}U_1^{\mathrm{out}}\left(\mathrm{tr}_{\mathrm{in}}\left(U_3^1U_2^1U_1^1\left(\rho^{\mathrm{in}}\otimes|000\rangle_1\langle000|\right)U_1^{1\dagger}U_2^{1\dagger}U_3^{1\dagger}\right)\otimes|00\rangle_{\mathrm{out}}\langle00|\right)U_1^{\mathrm{out}\dagger}U_2^{\mathrm{out}\dagger}\right)$$

Keep only the state of the last layer (trace out the hidden layer)

Quantum Neural Networks

Cost Function

Task: Learning an unknown unitary V

Training data: N pairs $(|\phi_x\rangle, V|\phi_x\rangle)$

Cost function: Fidelity

$$C = \frac{1}{N}\sum_{x=1}^{N}\langle\phi_x|V^\dagger\rho_x^{\text{out}}V|\phi_x\rangle$$

Fidelity measures the closeness of two quantum states.

Quantum Neural Networks

Cost Function

Task: Learning an unknown unitary V

Training data: N pairs $(|\phi_x\rangle, V|\phi_x\rangle)$

Cost function: Fidelity

$$C = \frac{1}{N}\sum_{x=1}^{N}\langle\phi_x|V^\dagger\rho_x^{\text{out}}V|\phi_x\rangle$$

Output states

Desired outputs

We want to maximize Fidelity!

Quantum Neural Networks

Training Quantum Neural Networks

$$U_j^l \rightarrow e^{i\epsilon K_j^l} U_j^l$$

Update the unitary matrix

Quantum Neural Networks

Training Quantum Neural Networks

$$U_j^l \rightarrow e^{i\epsilon K_j^l} U_j^l$$

Update the unitary matrix

Quantum Neural Networks

Training Quantum Neural Networks

$$U_j^l \to e^{i\epsilon K_j^l} U_j^l$$

$$K_j^l = \eta \frac{2^{m_{l-1}}}{N} \sum_{x=1}^{N} \mathrm{tr}_{\mathrm{rest}} \Big[U_j^l \ldots U_1^l \left(\rho_x^{l-1} \otimes |0\ldots0\rangle_l \langle 0\ldots0| \right) U_1^{l\,\dagger} \ldots U_j^{l\,\dagger}, \\ U_{j+1}^{l}{}^{\dagger} \ldots U_{m_l}^{l}{}^{\dagger} \left(\mathbb{I}_{l-1} \otimes \sigma_x^l \right) U_{m_l}^l \ldots U_{j+1}^l \Big]$$

Thank You

Reference

1. Training deep quantum neural networks - https://youtu.be/_M2GQAknykg?si=SsUGTe2Zl5BZMqcu
2. Training deep quantum neural networks - https://www.nature.com/articles/s41467-020-14454-2