

# **Representational Capabilities of Feed-forward and Sequential Neural Architectures**

**Clayton Sanford**  
Ph.D. Thesis Defense

**Committee:** Daniel Hsu, Rocco Servedio,  
Christos Papadimitriou, Joan Bruna, Matus Telgarsky

April 17, 2024



# Public talk

- Machine learning and neural networks background
- Dissertation contributions overview
- Vignette #1: Random feature models
- Vignette #2: Transformers and sequential models
- Acknowledgements

# What is machine learning?

Algorithms that make decisions based on data,  
without explicit programming.

# What is machine learning?

## Examples

- Object recognition



# What is machine learning?

## Examples

- Object recognition
- Content recommendation

The image shows a grid of 12 video thumbnails, likely from a search results page, demonstrating various applications of machine learning:

- MATRIX MARVEL** (13:00): How AI Discovered a Faster Matrix Multiplication Algorithm by Quanta Magazine. A thumbnail showing a 4x4 matrix multiplication diagram.
- Map of Computer Science** (10:58): Map of Computer Science by Domain of Science. A thumbnail showing a colorful map of computer science concepts.
- How to lie using visual proofs** (18:49): 3Blue1Brown. A thumbnail showing geometric diagrams used in visual proofs.
- But what is a convolution?** (23:01): 3Blue1Brown. A thumbnail showing examples of convolution operations on pixelated images.
- You Should Learn to Program: Christian Genco at TEDxSMU** (10:26): TEDx Talks. A thumbnail showing Christian Genco speaking at a TEDx event.
- STEW'S U.S HIGH SPEED RAIL NEWS APRIL 2024** (13:56): Stew's U.S. High Speed Rail News April 2024 | Brightline... by Lucid Stew. A thumbnail showing a high-speed train.
- It's never too late.** (14:44): How Washington DC fixed their Metro's biggest problem by The Flying Moose. A thumbnail showing a subway train.
- Bullet Train | Progress video - March 2024 | MAHSR | NCSRCL** (5:20): Bullet Train | Progress video - March 2024 | MAHSR | NCSRCL by NCSRCL INDIA. A thumbnail showing a bullet train on a track.
- Vancouver's Ideal Transit System - Capstone Project** (16:22): Vancouver's Ideal Transit System - Capstone Project by The Greater Discussions. A thumbnail showing a transit map with a yellow arrow pointing to a train.
- HS2 Birmingham's Coleshill going towards curdworth area...** (3:44): HS2 Birmingham's Coleshill going towards curdworth area... by Paul C G Burrows. A thumbnail showing an aerial view of a construction site.

# What is machine learning?

## Examples

- Object recognition
- Content recommendation
- Large language models

The screenshot shows a conversation in ChatGPT 4. The user asks, "How do I prepare for my PhD thesis defense?". ChatGPT responds by providing a general overview and then listing two main steps: 1. Understand the Format and 2. Prepare Your Presentation, each with specific sub-points.

You  
How do I prepare for my PhD thesis defense?

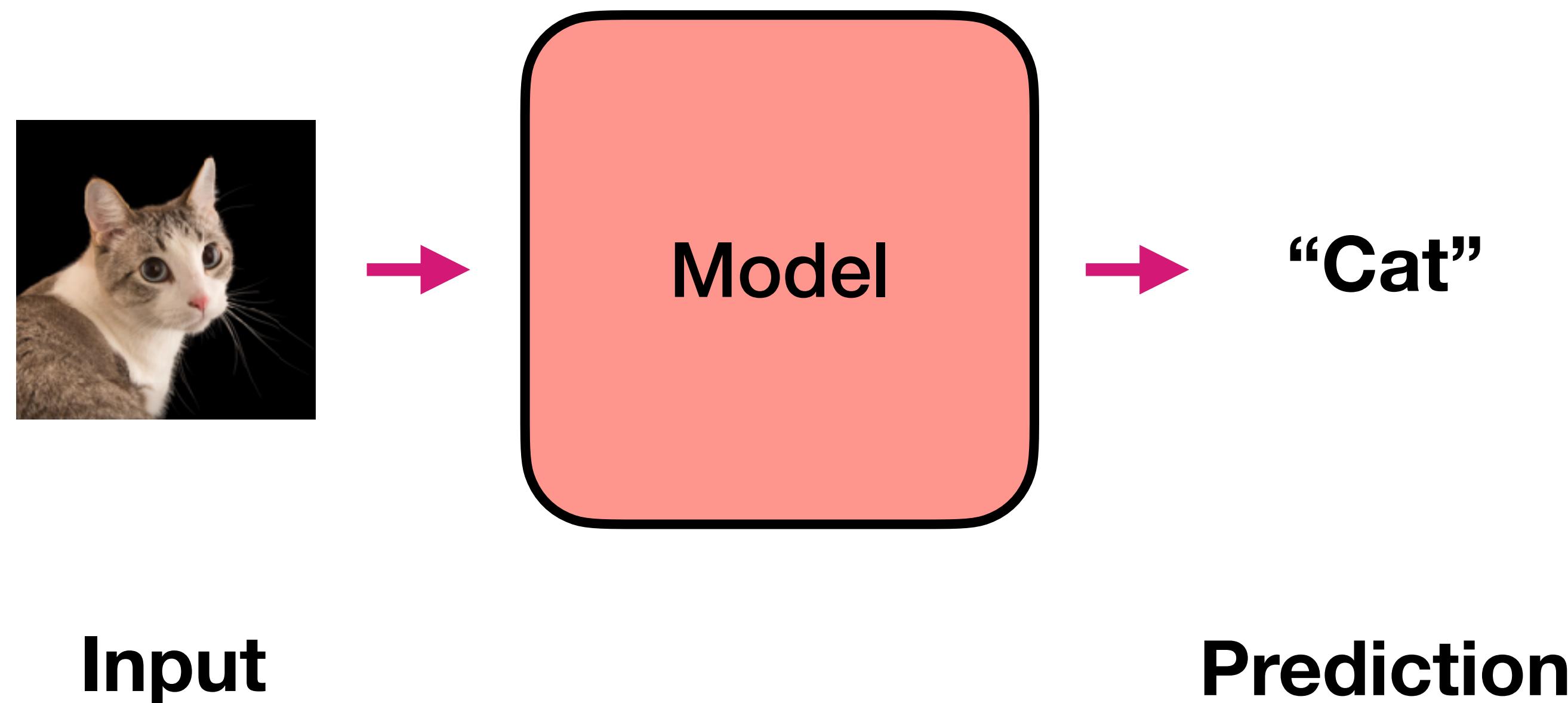
ChatGPT  
Preparing for a PhD thesis defense is a significant milestone, and it's essential to be thoroughly prepared. Here are some steps to help you get ready for this important event:

1. **Understand the Format:** Different institutions and departments have specific formats for thesis defenses. Make sure you understand the structure of the event, including the length of your presentation and the sequence of events.
2. **Prepare Your Presentation:**
  - **Content:** Summarize the key points of your research, including your hypotheses, methodology, results, and conclusions. Make sure to highlight the significance of your findings and how they contribute to your field.
  - **Slides:** Create clear, professional slides to support your presentation. Use visuals like charts, graphs, and images to make complex information easier to understand.
  - **Practice:** Rehearse your presentation multiple times. You might also record yourself to critique your performance or present in front of friends or colleagues for feedback.

Message ChatGPT...

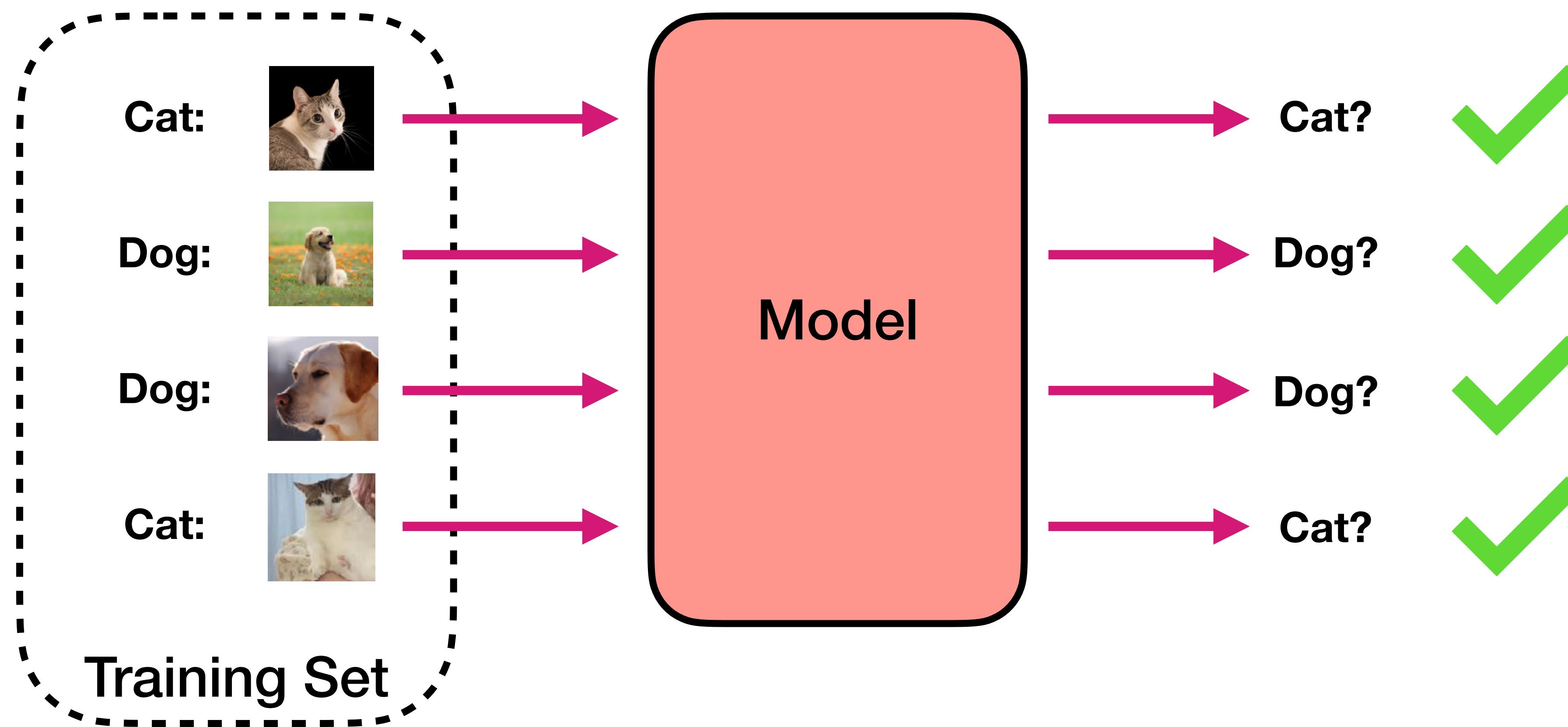
# How does ML work?

**Goal:** Model that correctly maps inputs to labels.



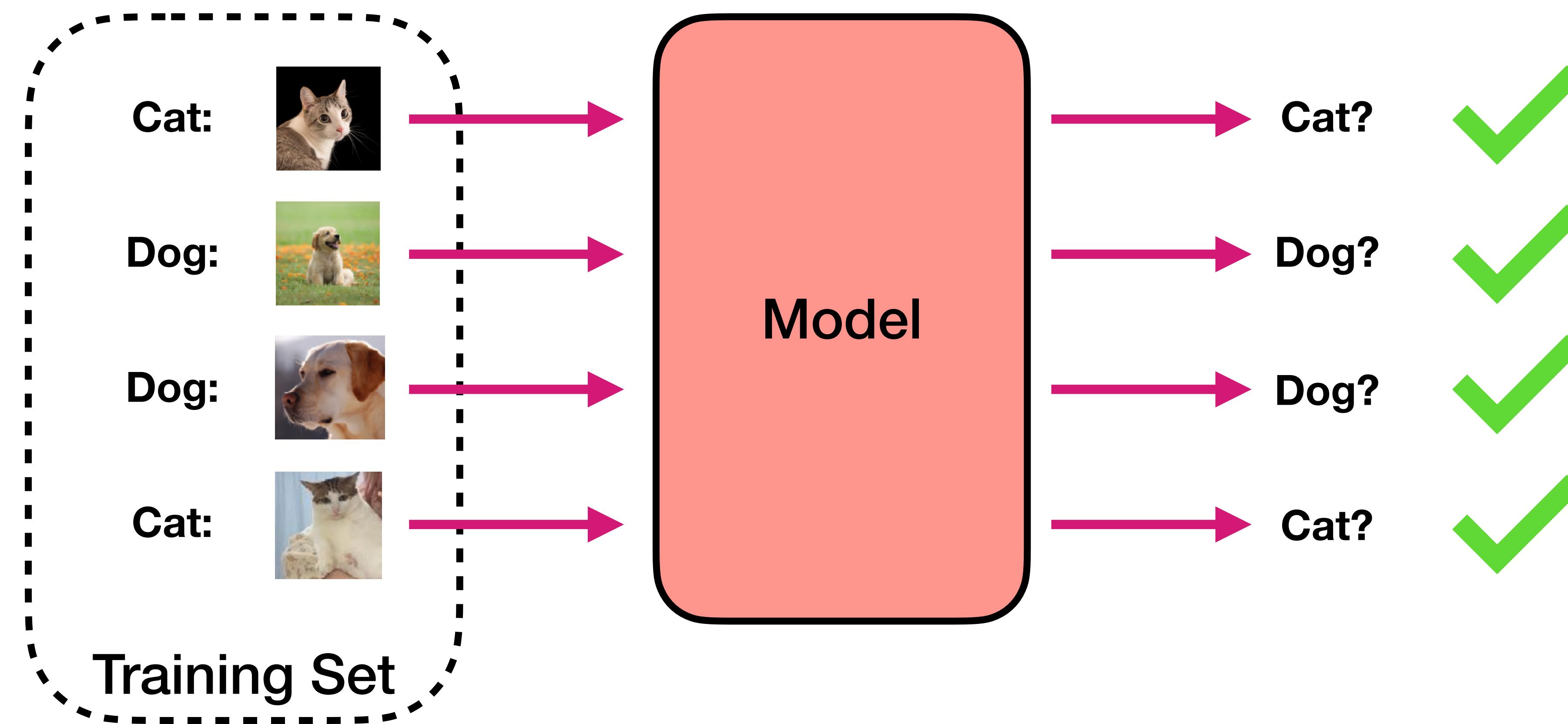
# How does ML work?

**Approach:** Choose a model that correctly classifies training set.



# How does ML work?

**Challenge:** How to find a “pixels to cat/dog” mathematical function?

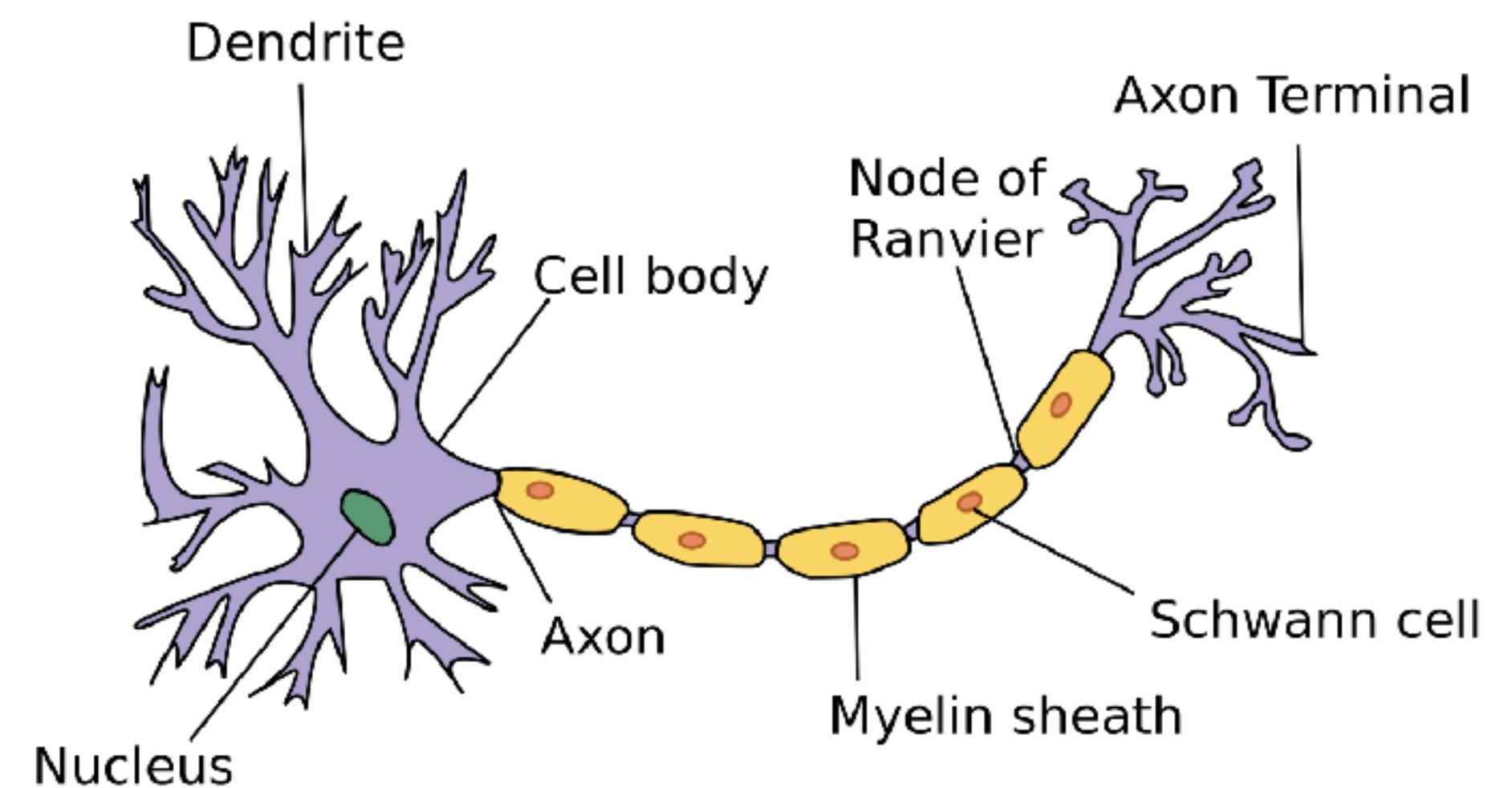


# Neural network models

Model

A **neural network**  $f$  is a biologically inspired mathematical function.

An **architecture**  $\mathcal{F}$  is a family of neural networks with similar structure.



# Neural network models

Architecture  $\mathcal{F}$  of networks  $f$ .

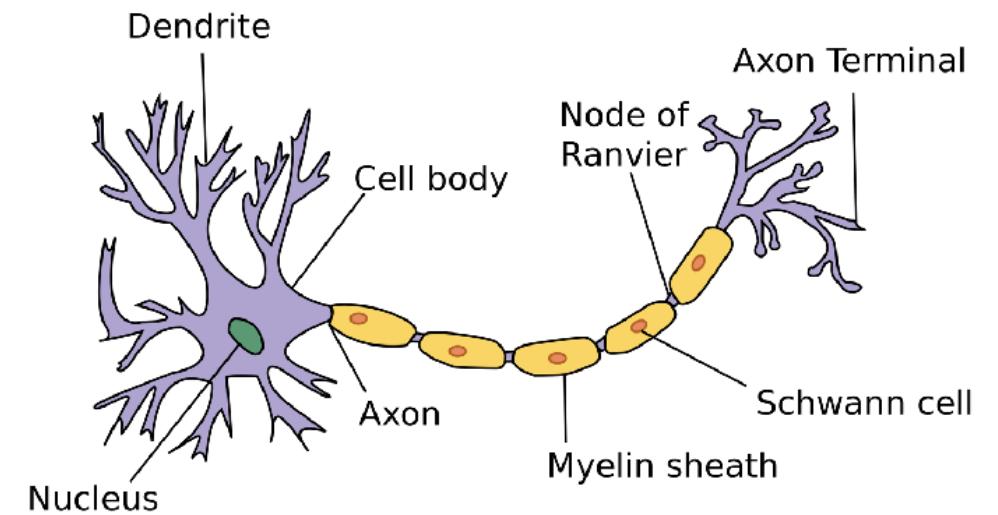
**Perceptron:**  $f(x) = \sigma(w^\top x + b)$ .

- Parameters:  $w, b$ .
- Threshold activation  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ .

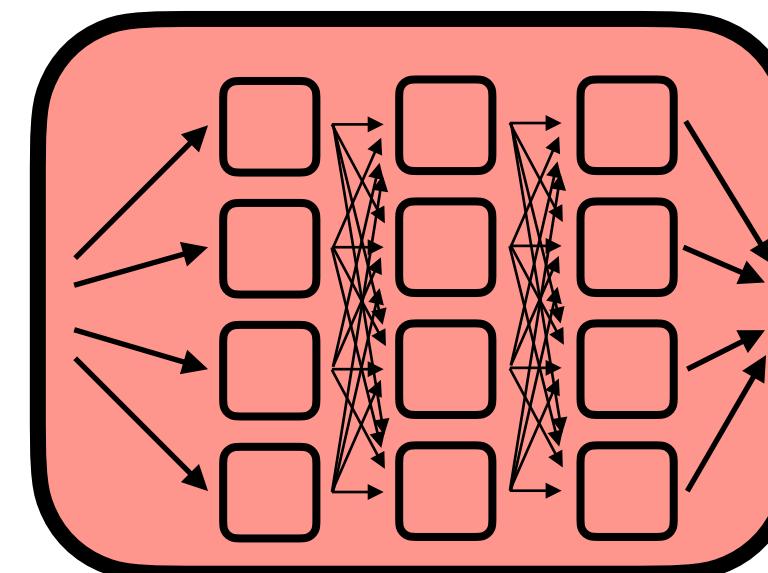
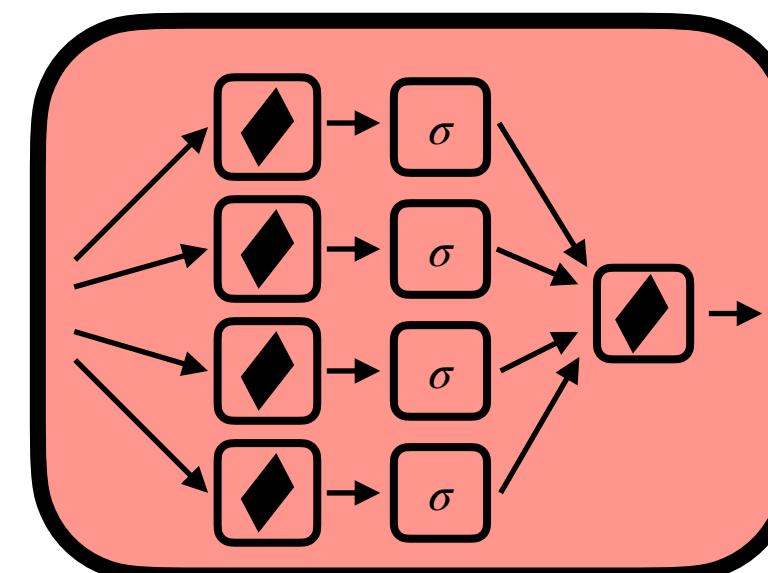
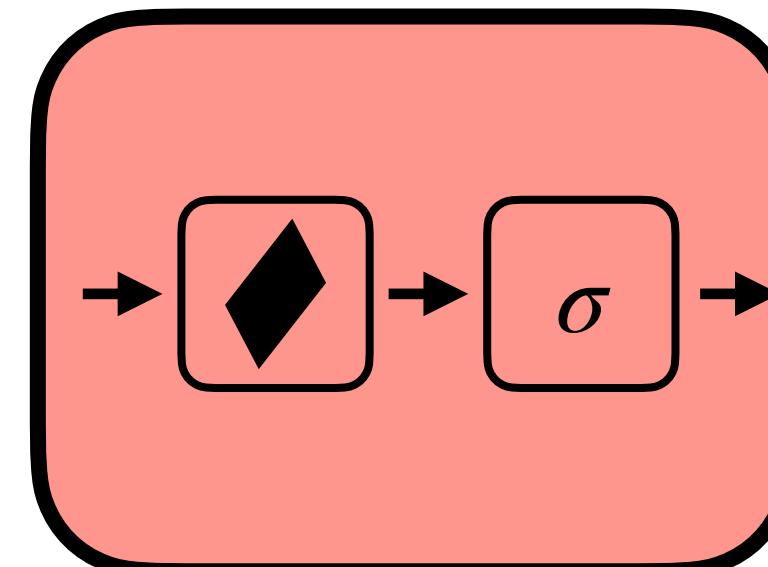
**Two-layer perceptron:**  $f(x) = \sum_{i=1}^m u_i \sigma(w_i^\top x + b_i)$

- Parameters:  $w, b, u$ .

**Multi-layer perceptron:**

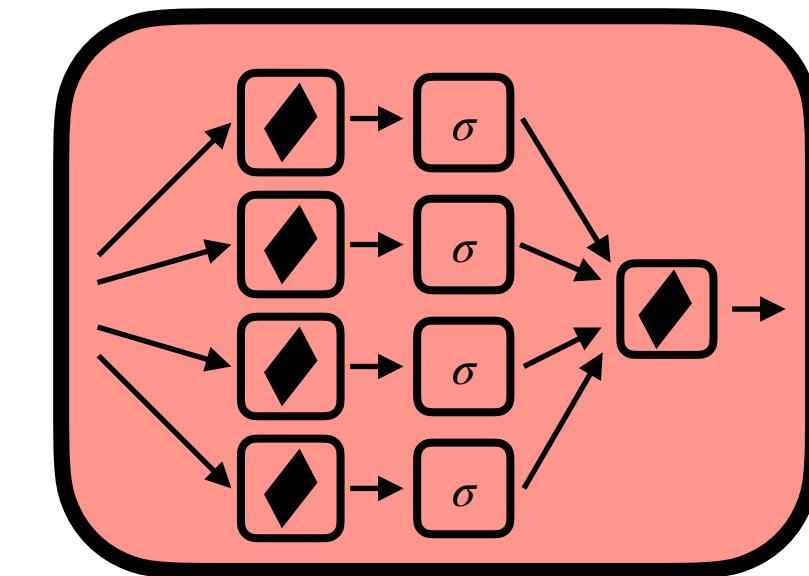


Model



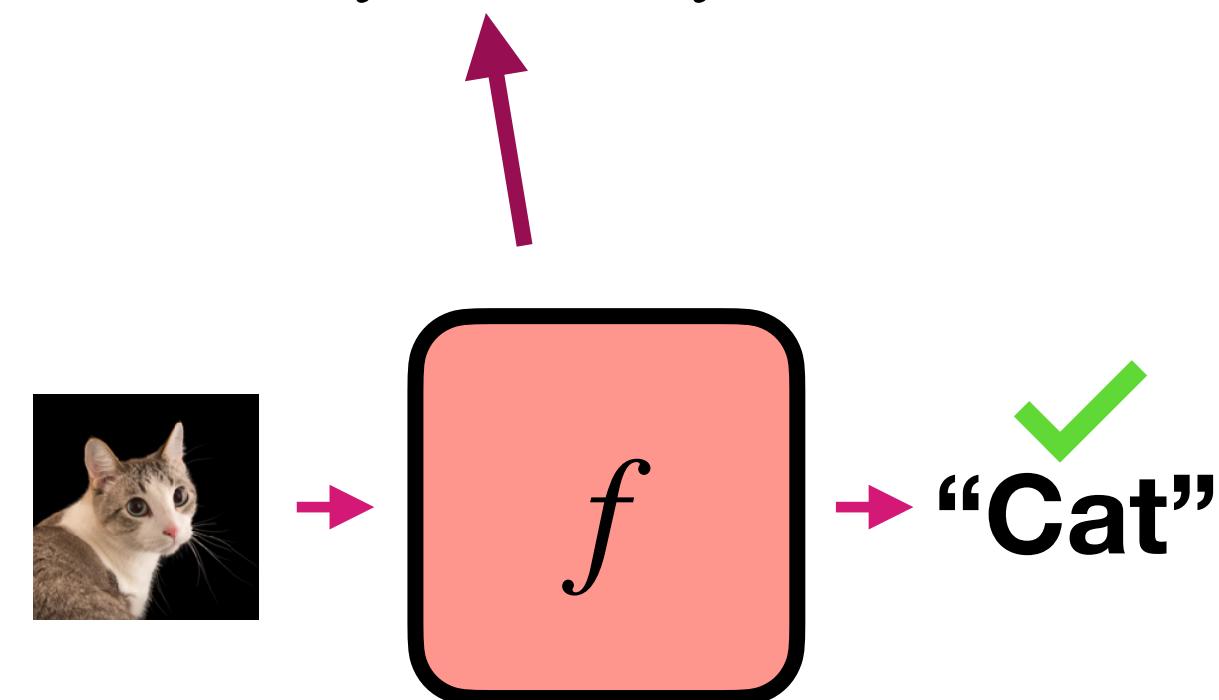
# Complete ML pipeline

1. Choose an architecture  $\mathcal{F}$ .



# Complete ML pipeline

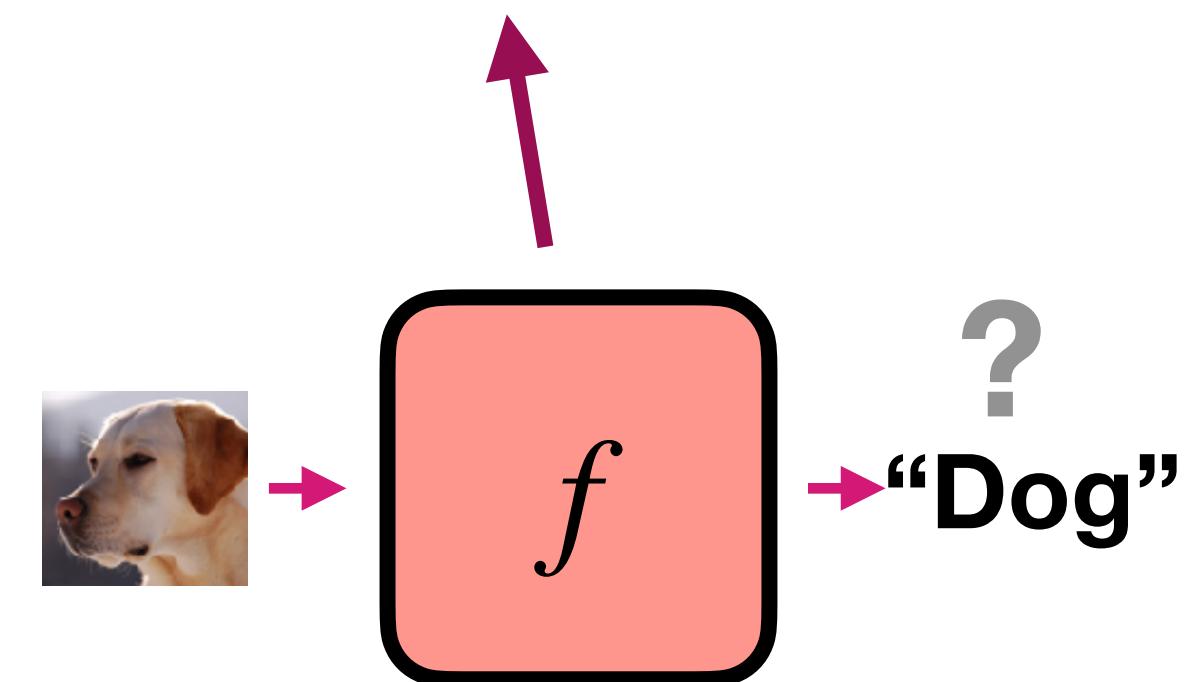
1. Choose an architecture  $\mathcal{F}$ .
2. Find network  $f \in \mathcal{F}$  that fits labeled dataset  $(x_1, y_1), \dots, (x_n, y_n)$ , i.e. choose  $f$  such that  $f(x_i) \approx y_i$ .



“Cat”

# Complete ML pipeline

1. Choose an architecture  $\mathcal{F}$ .
2. Find network  $f \in \mathcal{F}$  that fits labeled dataset  $(x_1, y_1), \dots, (x_n, y_n)$ , i.e. choose  $f$  such that  $f(x_i) \approx y_i$ .
3. Deploy trained network  $f$  to new data, use  $f(x_{\text{test}})$  as prediction.



# Theoretical questions about ML

1. Choose an architecture  $\mathcal{F}$ .
2. Find network  $f \in \mathcal{F}$  that fits labeled dataset  $(x_1, y_1), \dots, (x_n, y_n)$ , i.e. choose  $f$  such that  $f(x_i) \approx y_i$ .
3. Deploy trained network  $f$  to new data, use  $f(x_{\text{test}})$  as prediction.

# Theoretical questions about ML

1. Choose an architecture  $\mathcal{F}$ .  
Is  $\mathcal{F}$  expressive enough for the data? **[Representation]**
2. Find network  $f \in \mathcal{F}$  that fits labeled dataset  $(x_1, y_1), \dots, (x_n, y_n)$ ,  
i.e. choose  $f$  such that  $f(x_i) \approx y_i$ .
3. Deploy trained network  $f$  to new data, use  $f(x_{\text{test}})$  as prediction.

# Theoretical questions about ML

1. Choose an architecture  $\mathcal{F}$ .  
Is  $\mathcal{F}$  expressive enough for the data? **[Representation]**
2. Find network  $f \in \mathcal{F}$  that fits labeled dataset  $(x_1, y_1), \dots, (x_n, y_n)$ ,  
i.e. choose  $f$  such that  $f(x_i) \approx y_i$ .  
Can  $f$  be found efficiently? **[Optimization]**
3. Deploy trained network  $f$  to new data, use  $f(x_{\text{test}})$  as prediction.

# Theoretical questions about ML

1. Choose an architecture  $\mathcal{F}$ .  
Is  $\mathcal{F}$  expressive enough for the data? **[Representation]**
2. Find network  $f \in \mathcal{F}$  that fits labeled dataset  $(x_1, y_1), \dots, (x_n, y_n)$ ,  
i.e. choose  $f$  such that  $f(x_i) \approx y_i$ .  
Can  $f$  be found efficiently? **[Optimization]**
3. Deploy trained network  $f$  to new data, use  $f(x_{\text{test}})$  as prediction.  
Does high training accuracy mean high test accuracy? **[Generalization]**

# Theoretical questions about ML

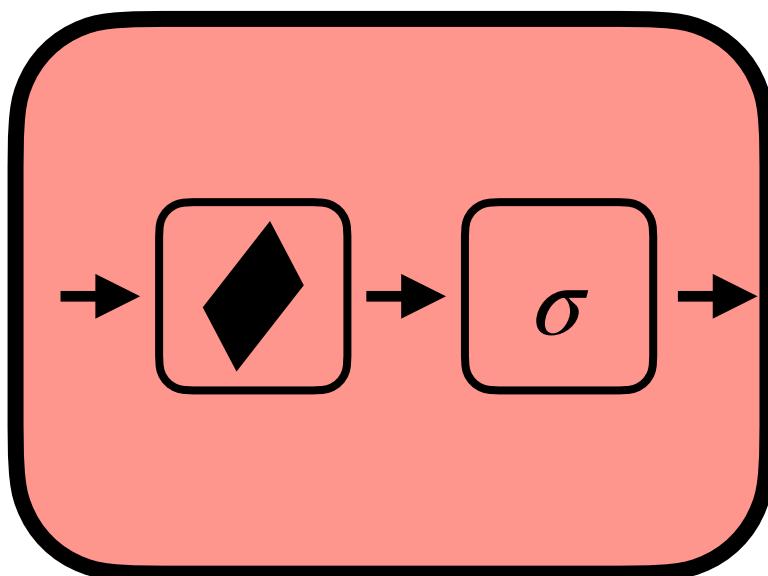
1. Choose an architecture  $\mathcal{F}$ .  
Is  $\mathcal{F}$  expressive enough for the data? **[Representation]**
2. Find network  $f \in \mathcal{F}$  that fits labeled dataset  $(x_1, y_1), \dots, (x_n, y_n)$ ,  
i.e. choose  $f$  such that  $f(x_i) \approx y_i$ ..  
Can  $f$  be found efficiently? **[Optimization]**
3. Deploy trained network  $f$  to new data, use  $f(x_{\text{test}})$  as prediction.  
Does high training accuracy mean high test accuracy? **[Generalization]**

# Vignette #0

**Perceptron [Rosenblatt '58]:**

$$f(x) = \sigma(w^T x + b).$$

- Parameters:  $w, b$ .
- Threshold activation  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ .



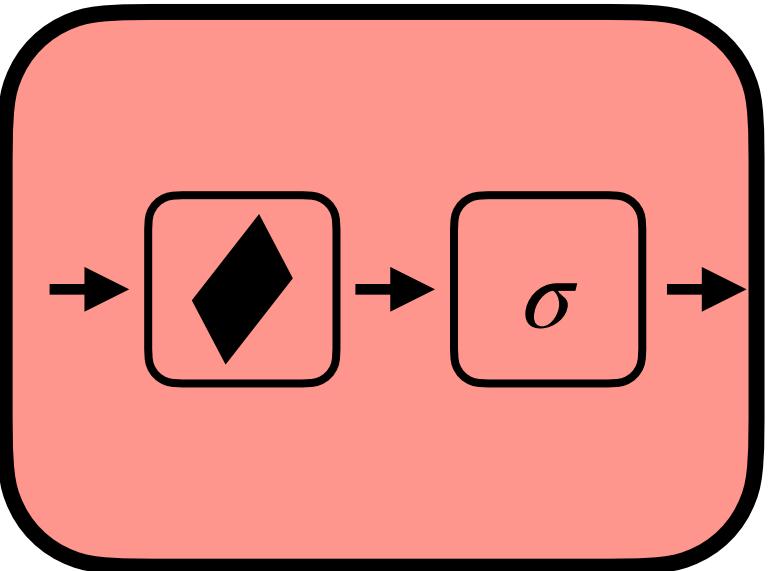
“walk, talk, see,  
write, reproduce itself  
and be conscious of its  
existence”

“learn, make  
decisions, and  
translate languages”



Frank Rosenblatt

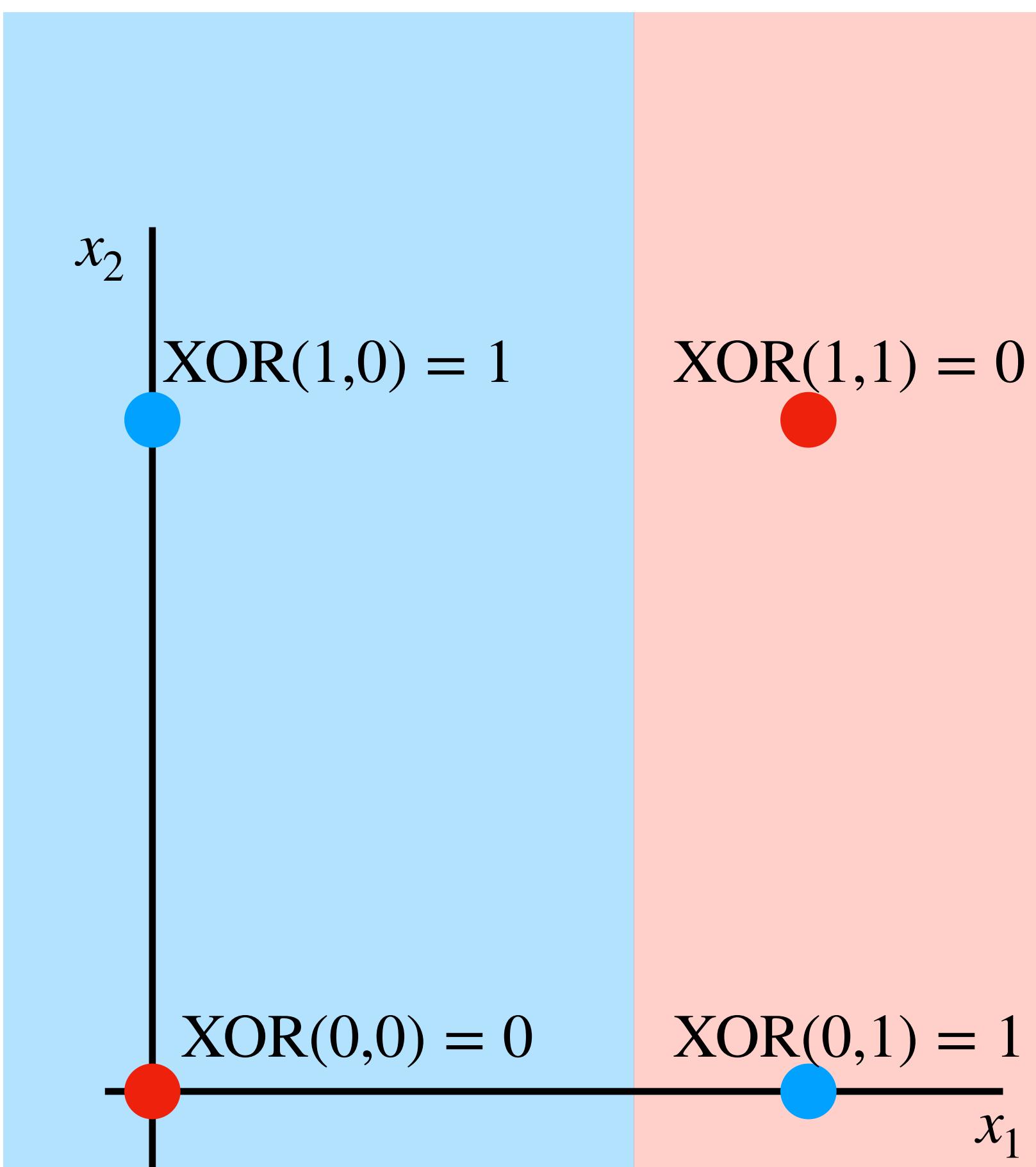
# Vignette #0: Perceptrons



$$f(x) = \sigma(w^T x + b).$$

**Papert and Minsky '69:**

- Perceptrons only work for linearly separable data.
- **Negative result:** No Perceptron  $f_\theta$  computes XOR.

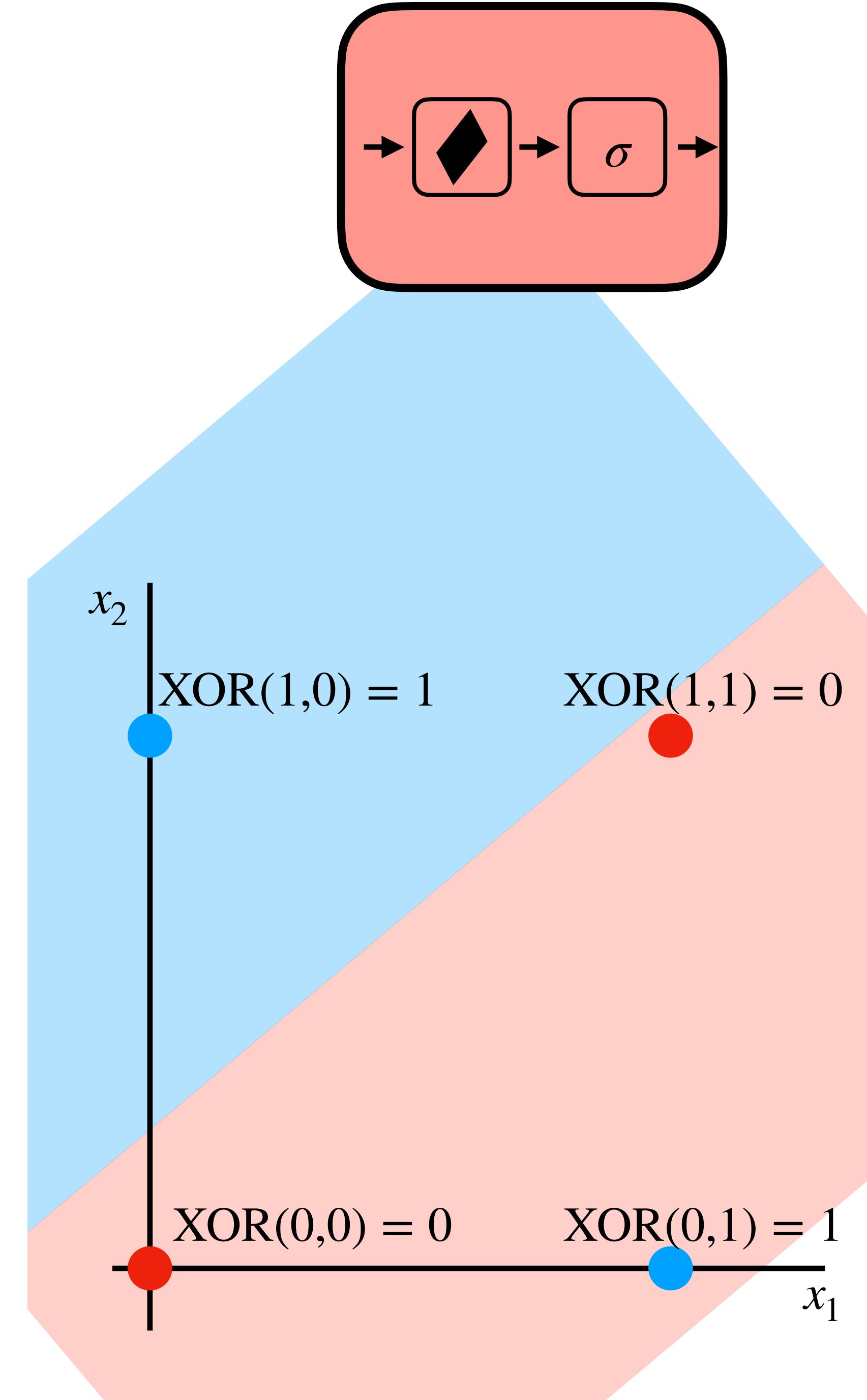


# Vignette #0: Perceptrons

$$f(x) = \sigma(w^T x + b).$$

**Papert and Minsky '69:**

- Perceptrons only work for linearly separable data.
- **Negative result:** No Perceptron  $f_\theta$  computes XOR.

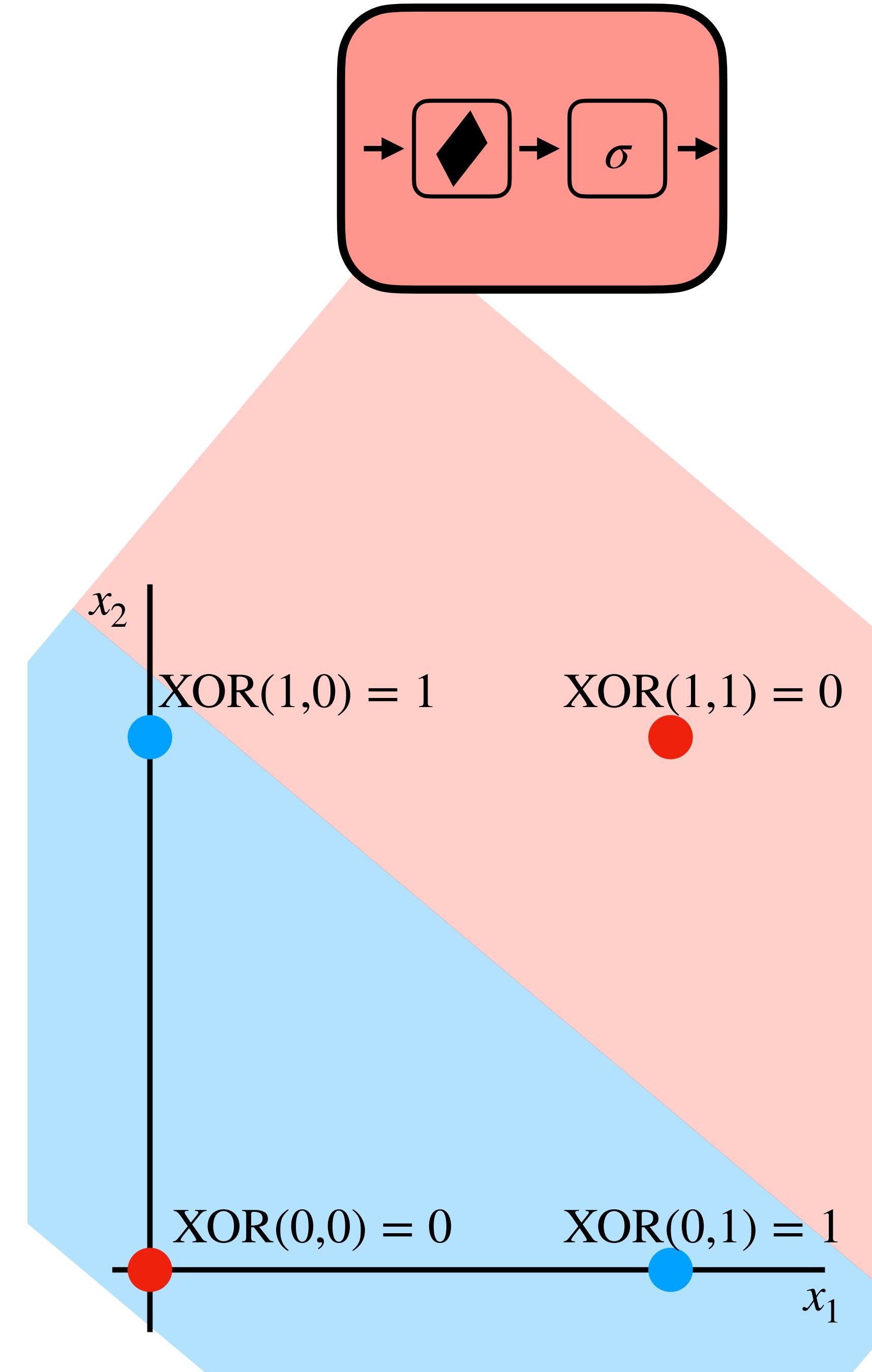


# Vignette #0: Perceptrons

$$f(x) = \sigma(w^T x + b).$$

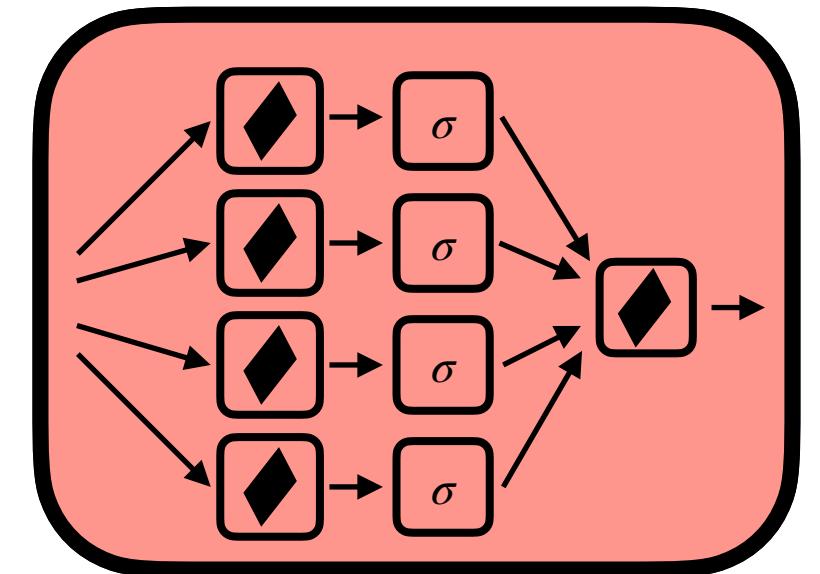
**Papert and Minsky '69:**

- Perceptrons only work for linearly separable data.
- **Negative result:** No Perceptron  $f_\theta$  computes XOR.  
➡ Fundamental limitation of Perceptron.



# Universal Approximation Theorem

Funahashi et al; Hornik et al; Cybenko, '89



**Two-layer NN:**  $f(x) = \sum_{i=1}^m u_i \sigma(w_i^\top x + b_i)$

**Positive result:** For continuous  $h : \mathbb{R}^d \rightarrow \mathbb{R}$ , accuracy  $\epsilon > 0$ , and compact  $\mathcal{X} \subset \mathbb{R}^d$ , there exists a 2-layer network  $f$  with

$$\max_{x \in \mathcal{X}} |f(x) - h(x)| \leq \epsilon.$$

- [+] Rules out impossible XOR-like targets for neural networks of depth at least 2.
- [-] No optimization guarantees—how do you find it?
- [-] No size guarantees—how large is the width  $m$ ?

# Thesis questions

**Positive results:**

Which tasks can an architecture **efficiently** solve?

**Negative results:**

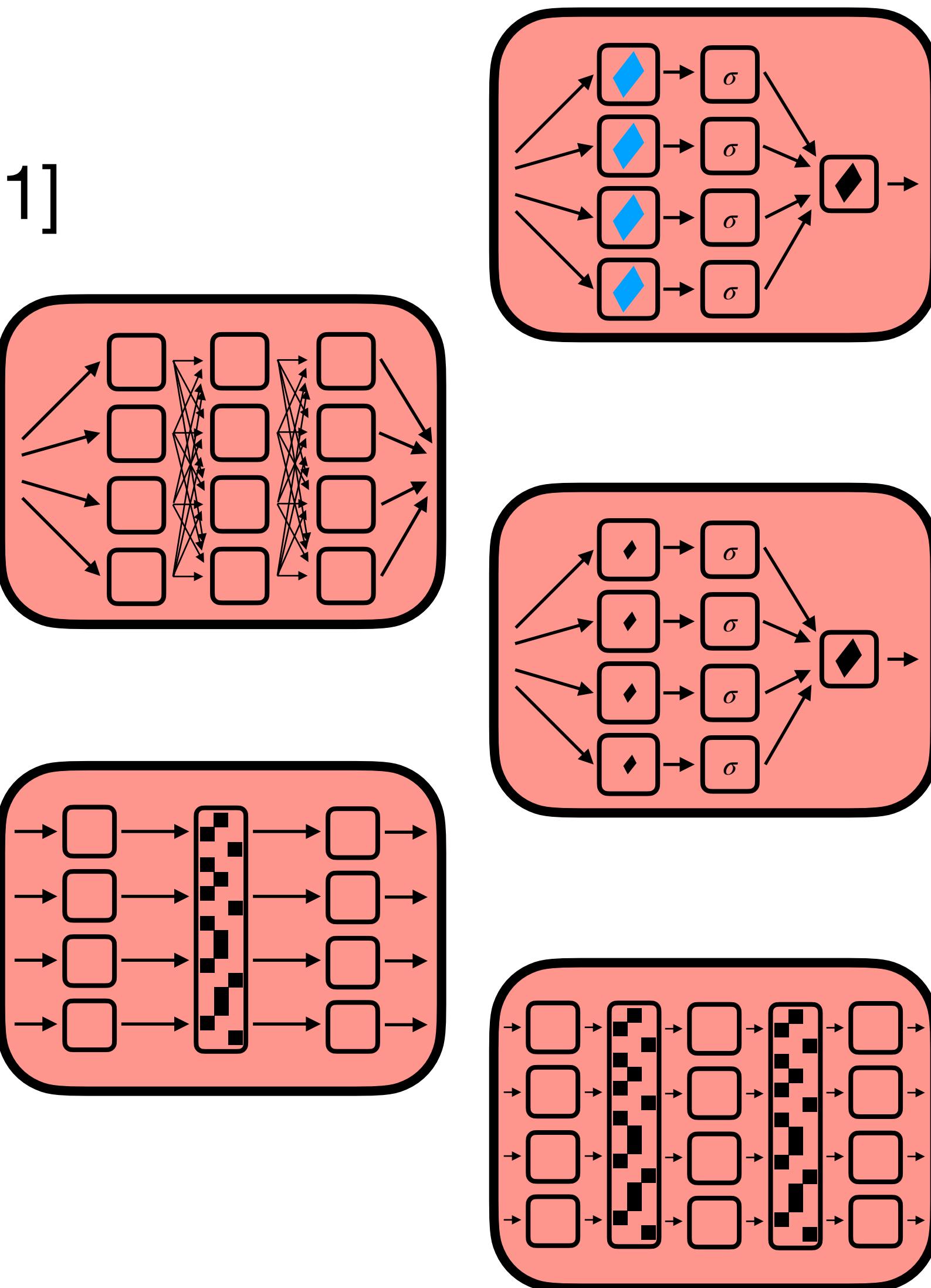
Which tasks **cannot** be efficiently solved by an architecture?

**Contrasts:**

Which tasks distinguish the powers of two architectures?

# Dissertation chapters

1. 2-layer random feature NNs  
[Hsu, **S**, Servedio, Vlatakis-Gkaragkounis '21]
2. Shallow vs deep NNs  
[**S**, Chatziafratis '22]
3. 2-layer regularized NNs  
[Ardeshir, Hsu, **S** '23]
4. Single-layer transformers  
[**S**, Hsu, Telgarsky '23]
5. Multi-layer transformers  
[**S**, Hsu, Telgarsky '24]

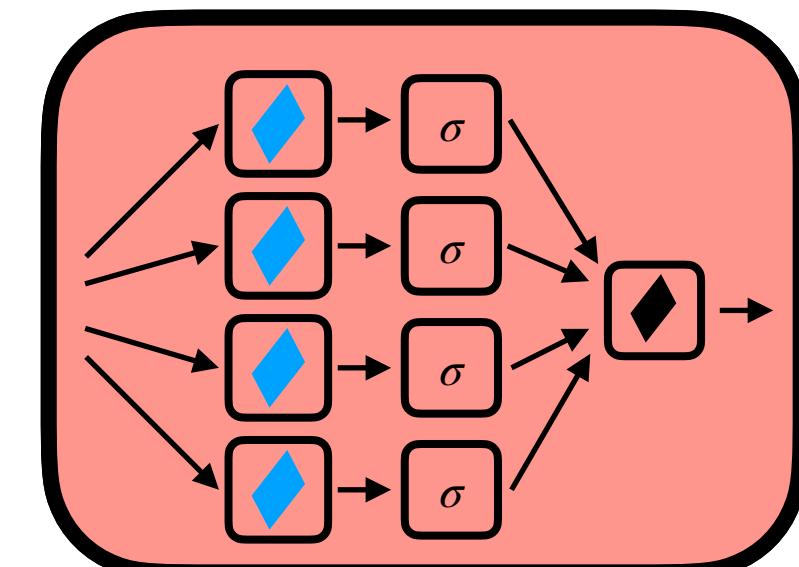


# Dissertation chapters

## Vignette #1

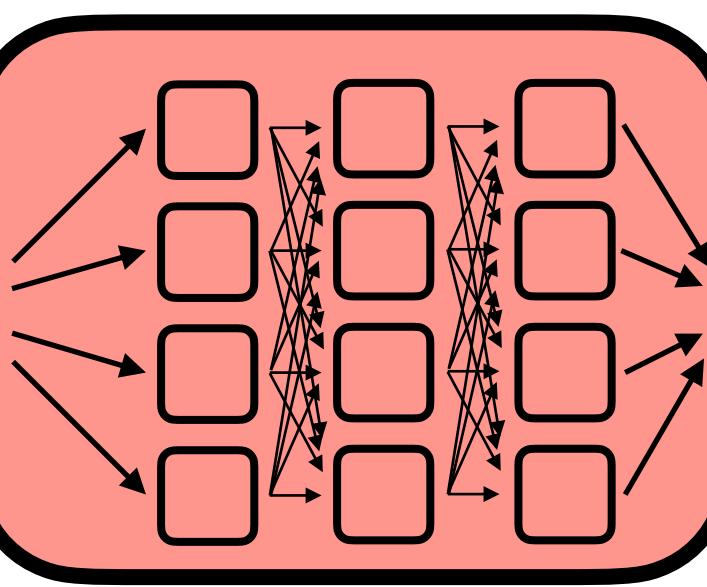
1. 2-layer random feature NNs

[Hsu, **S**, Servedio, Vlatakis-Gkaragkounis '21]



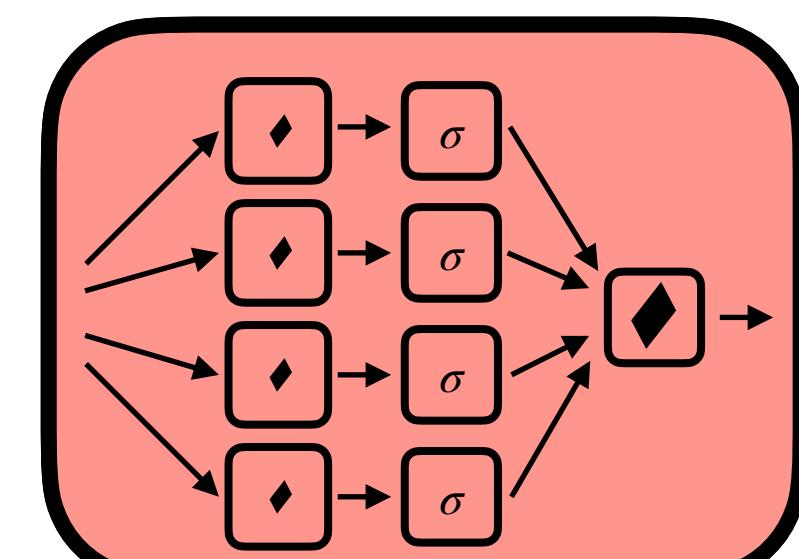
2. Shallow vs deep NNs

[**S**, Chatziafratis '22]



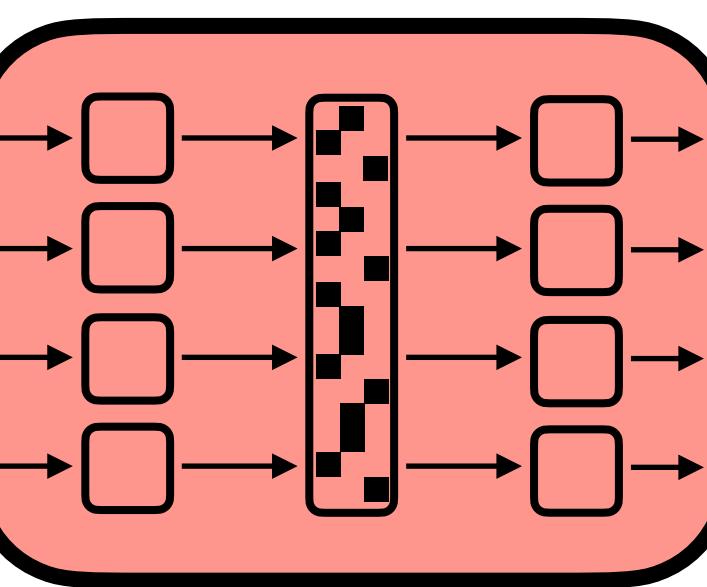
3. 2-layer regularized NNs

[Ardeshir, Hsu, **S** '23]



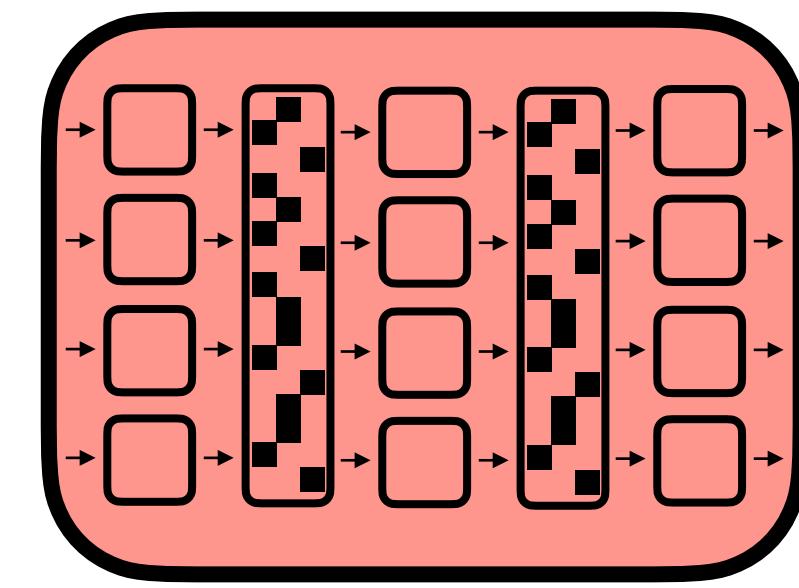
4. Single-layer transformers

[**S**, Hsu, Telgarsky '23]



5. Multi-layer transformers

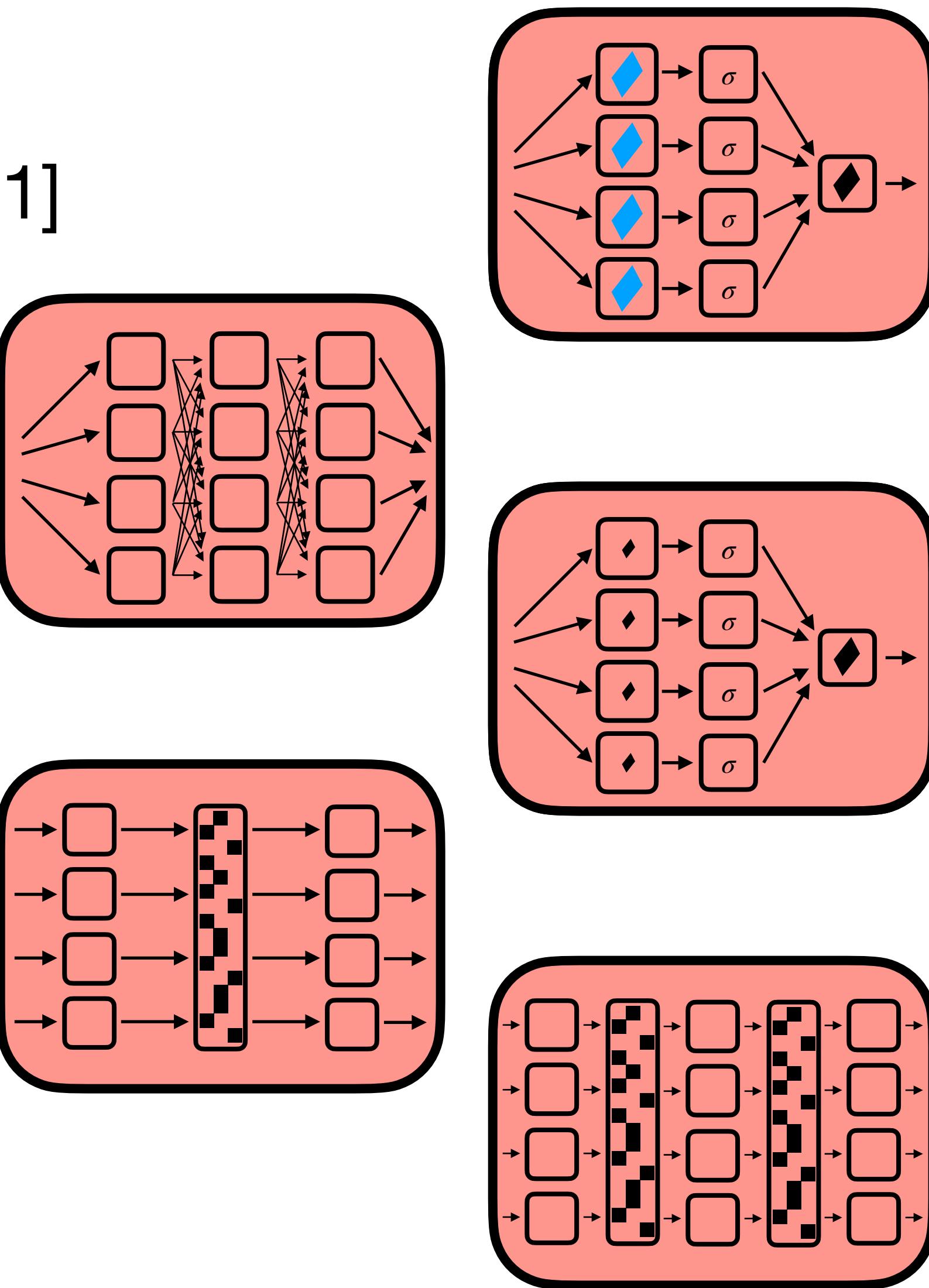
[**S**, Hsu, Telgarsky '24]

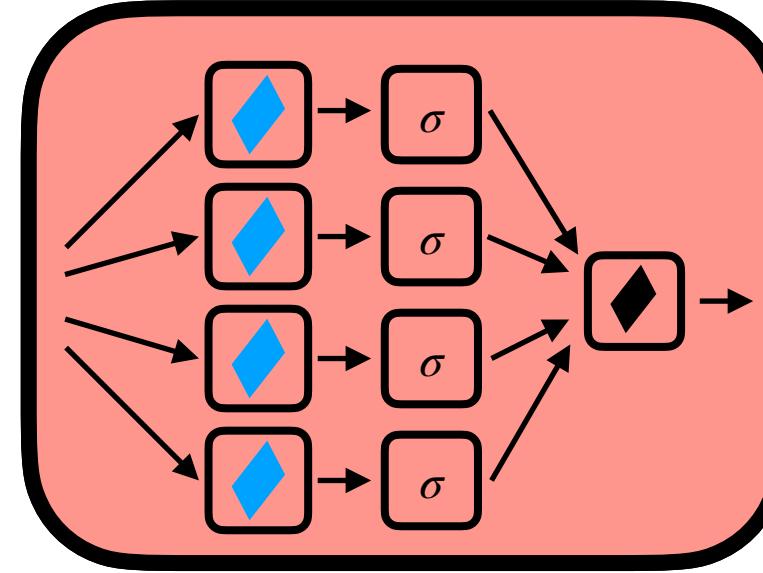


# Dissertation chapters

1. 2-layer random feature NNs  
[Hsu, **S**, Servedio, Vlatakis-Gkaragkounis '21]
2. Shallow vs deep NNs  
[**S**, Chatziafratis '22]
3. 2-layer regularized NNs  
[Ardeshir, Hsu, **S** '23]
4. Single-layer transformers  
[**S**, Hsu, Telgarsky '23]
5. Multi-layer transformers  
[**S**, Hsu, Telgarsky '24]

Vignette #2

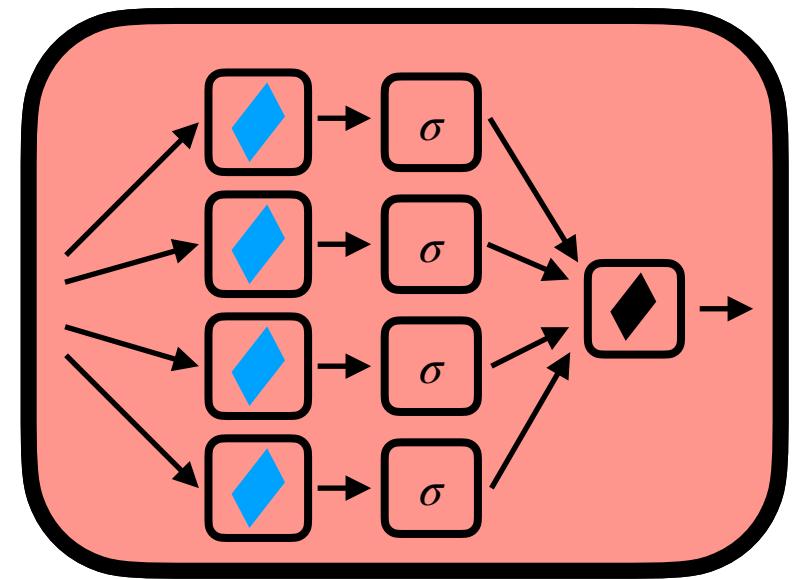




# Vignette #1: Random Feature Models

# Two-layer random feature models (RFMs)

**Architecture:**  $f(x) = \sum_{i=1}^m u_i \sigma(\mathbf{w}_i^\top x + \mathbf{b}_i).$



- Bottom-layer weights fixed, sampled at random:  $(\mathbf{w}_i, \mathbf{b}_i) \sim \mathcal{P}$ .
- Top-layer weights  $u = (u_1, \dots, u_m)$  chosen based on  $(\mathbf{w}, \mathbf{b})$ .

## Questions:

- Representational capabilities and limitations of RFMs?
- Comparison with standard 2-layer NNs?

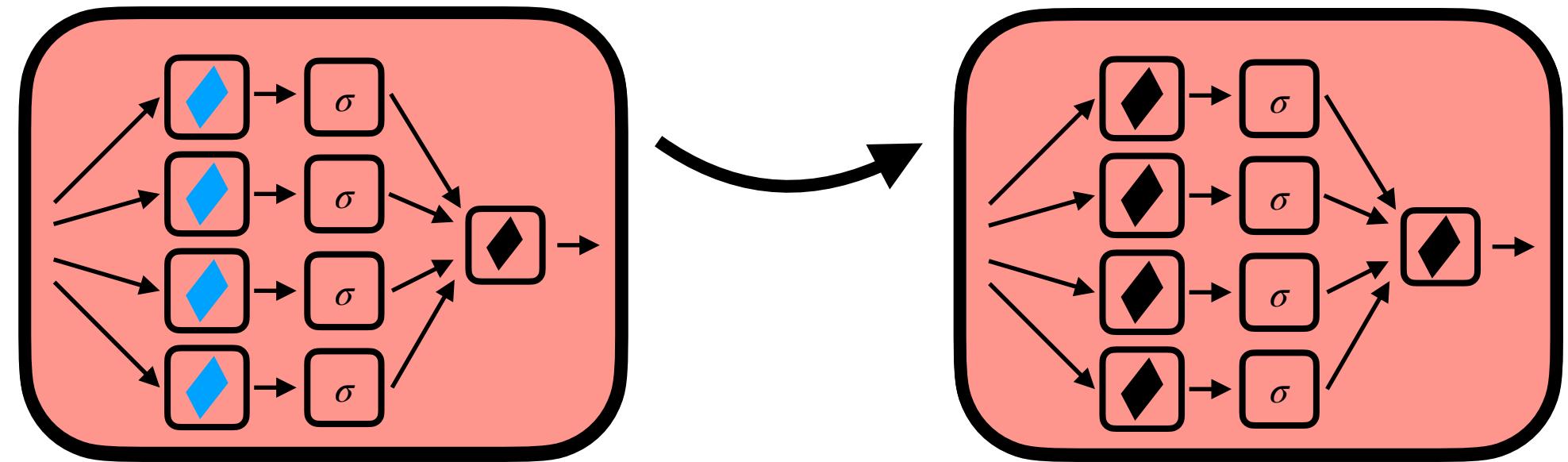
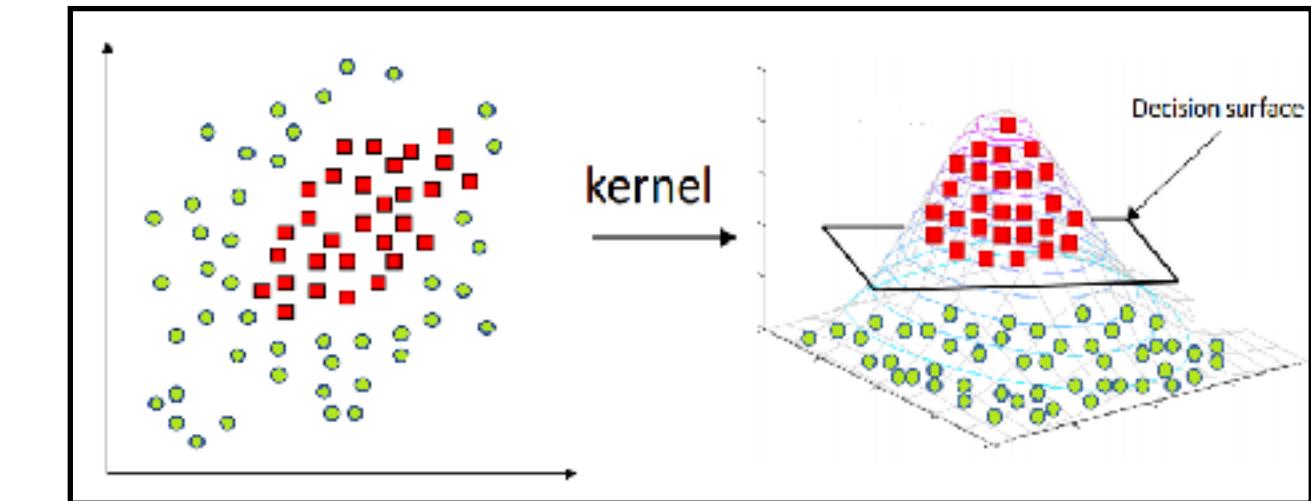
# RFMs, kernels, and neural networks

**Motivating question:** Can neural networks be accurately analyzed as kernel machines?

- Wish: Obtain NN optimization and generalization guarantees from kernels.

**Connections:**

- RFMs are NNs.
- Kernel machines are approximable by RFMs [Rahimi, Recht '07].
- Certain NN initializations train like kernels (NTK) [Jacot, et al '19].



# Contributions

## 1. Positive result:

RFMs (and two-layer NNs) can approximate low-dimensional or smooth targets.

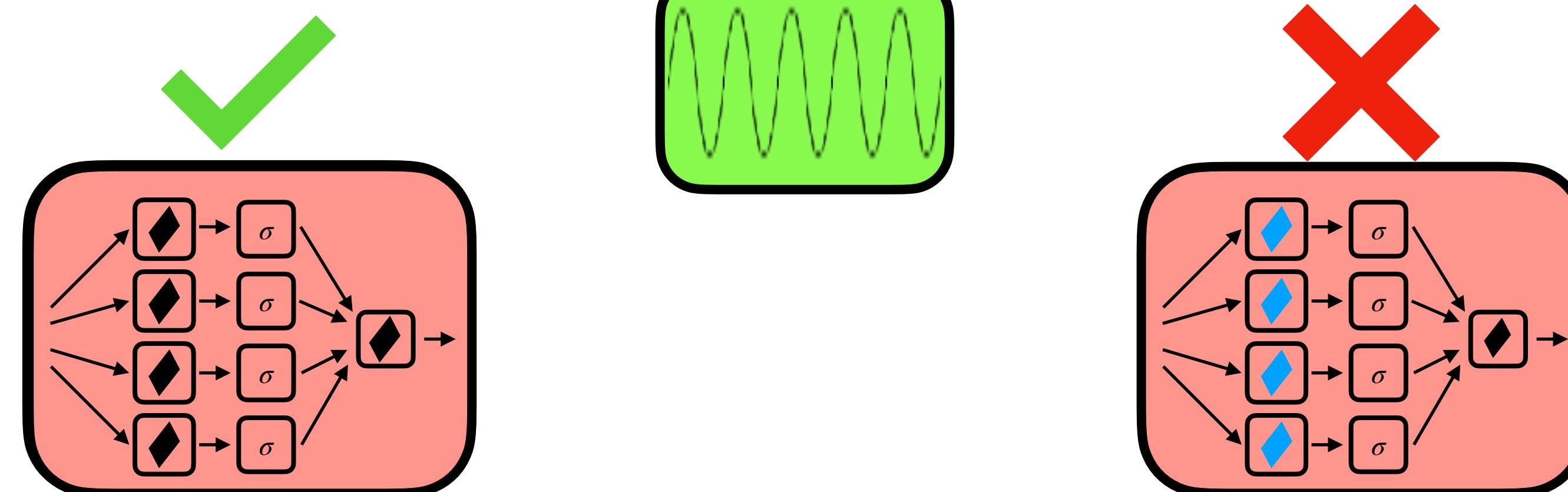
## 2. Negative result:

RFMs cannot approximate high-dimensional “jagged” targets.

## 3. Contrast:

Sinusoidal targets are efficiently approximable by NNs, but not by RFMs.

→ Flaw in NN-kernel analogy.



# Theoretical results

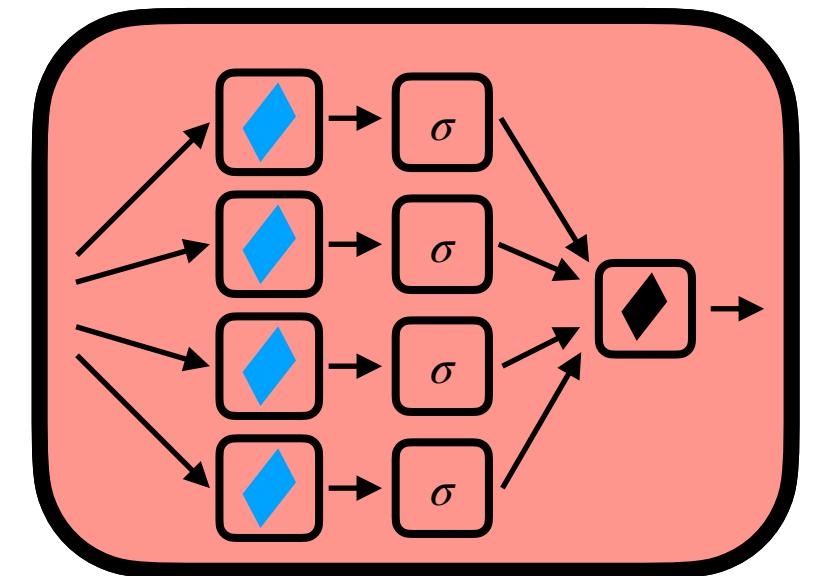
**Two-layer RFM:**  $f(x) = \sum_{i=1}^m u_i \sigma(\mathbf{w}_i^\top x + \mathbf{b}_i)$ .

- $h$  is  **$L$ -Lipschitz** if  $|h(x) - h(x')| \leq L\|x - x'\|_2$ .

**Theorem:** The minimum width  $m$  of an RFM  $f$  needed to approximate any  $d$ -dimensional  $L$ -Lipschitz target satisfies:

$$m = \min\{d^{\Theta(L^2)}, L^{\Theta(d)}\}$$

- **Theoretical significance:** First nearly-tight results for all choices of  $L$  and  $d$ .



# Theoretical results

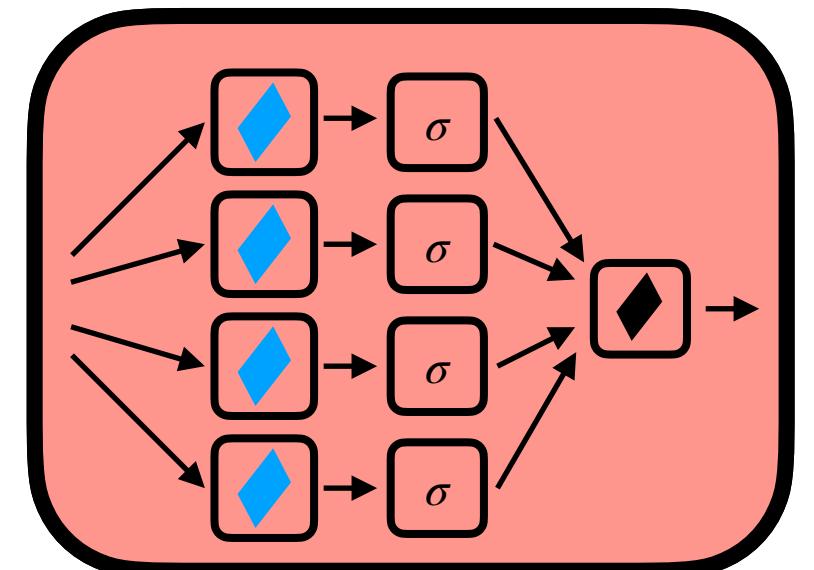
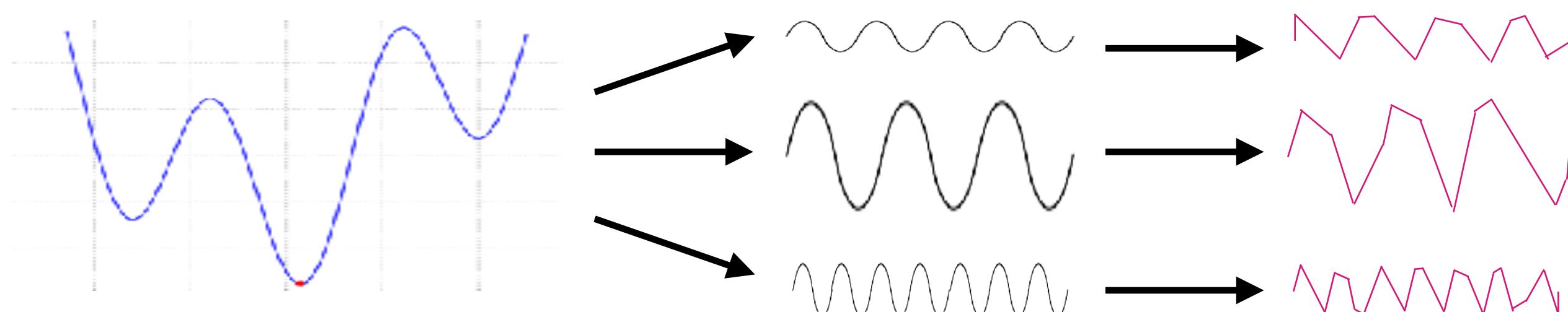
**Two-layer RFM:**  $f(x) = \sum_{i=1}^m u_i \sigma(\mathbf{w}_i^\top x + \mathbf{b}_i)$ .

- $h$  is  **$L$ -Lipschitz** if  $|h(x) - h(x')| \leq L\|x - x'\|_2$ .

**Theorem [positive]:** There exists a width- $m$  RFM  $f$  that approximates any  $d$ -dimensional  $L$ -Lipschitz target  $h$  when:

$$m \leq \min\{d^{O(L^2)}, L^{O(d)}\}$$

**Proof idea:** Approx.  $h$  with sinusoids, approx. sinusoids with random neurons.



# Theoretical results

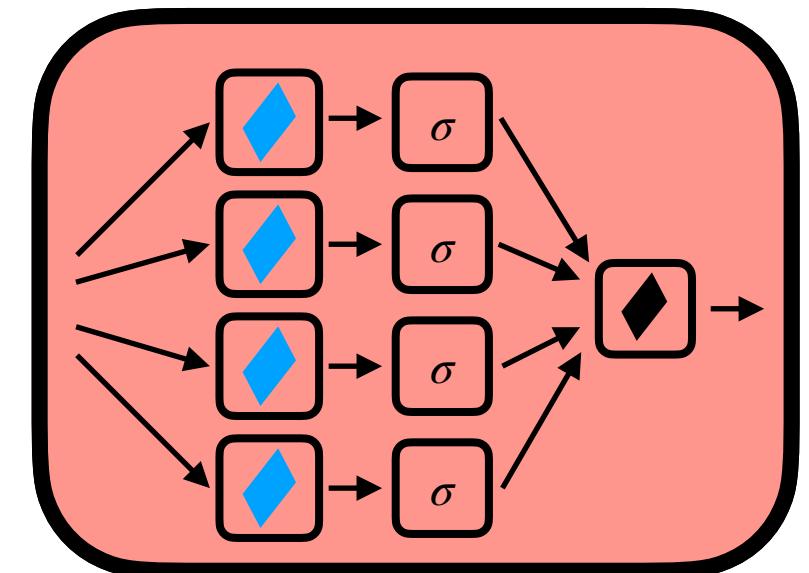
**Two-layer RFM:**  $f(x) = \sum_{i=1}^m u_i \sigma(\mathbf{w}_i^\top x + \mathbf{b}_i)$ .

- $h$  is  **$L$ -Lipschitz** if  $|h(x) - h(x')| \leq L\|x - x'\|_2$ .

**Theorem [negative]:** For  $d$ -dimensional  $L$ -Lipschitz target  $h$ , every width- $m$  RFM  $f$  that approximates it has:

$$m \geq \min\{d^{\Omega(L^2)}, L^{\Omega(d)}\}$$

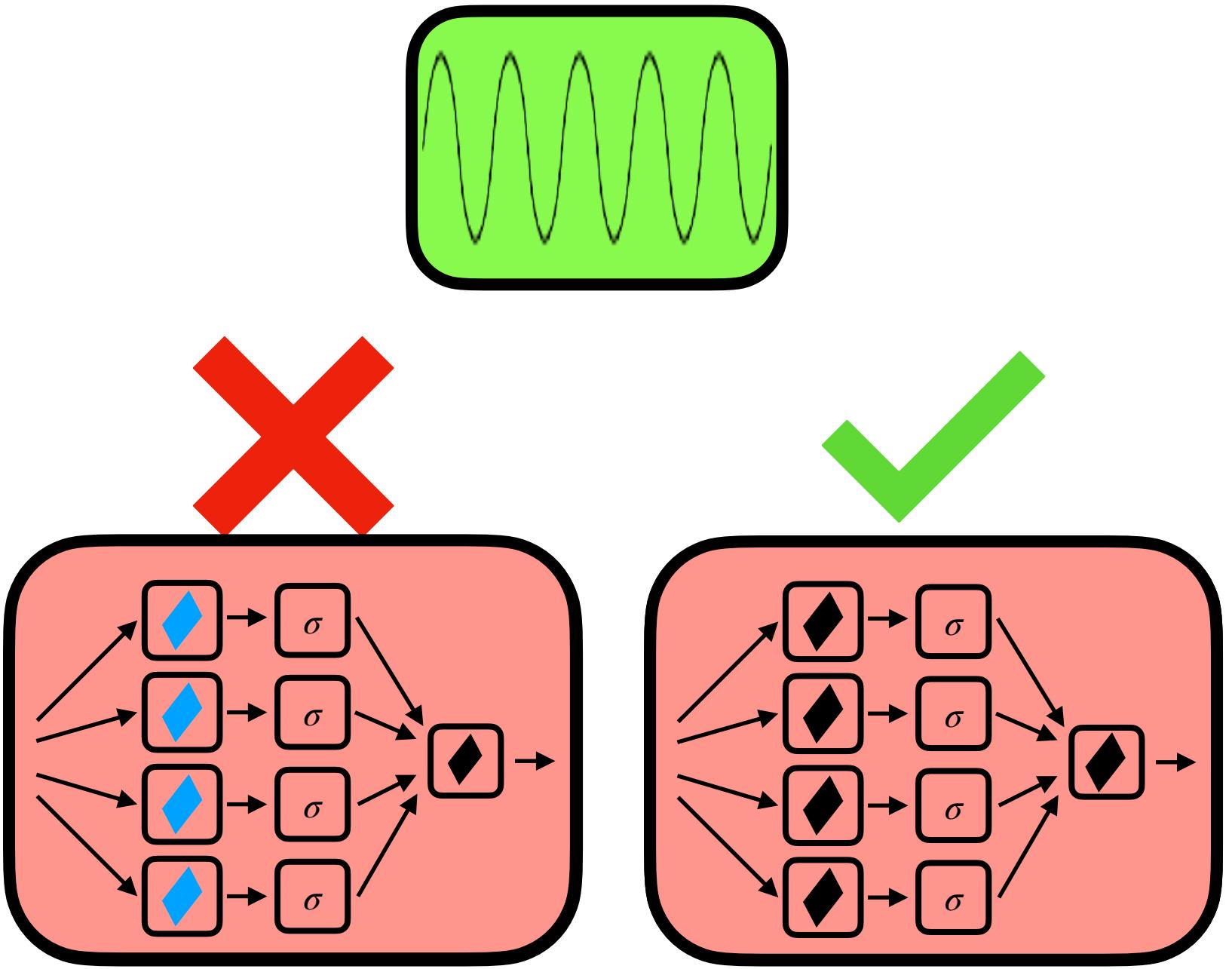
**Proof idea:** Kernel-based dimensionality argument.



# Separation with 2-layer NNs

**Contrast:** Sinusoidal target  $h(x) = \sin(\sqrt{d}\nu^\top x)$ .

- All RFMs that approximate  $h$  have large width:  $m = \exp(d)$ .
- But, 2-layer NN can approximate  $h$  with small width:  $m = \text{poly}(d)$ .



**Implication:** NNs can represent “simple” targets; RFMs cannot.

- Similar optimization conclusions [e.g. Damian, Lee, Soltanolkotabi '22].

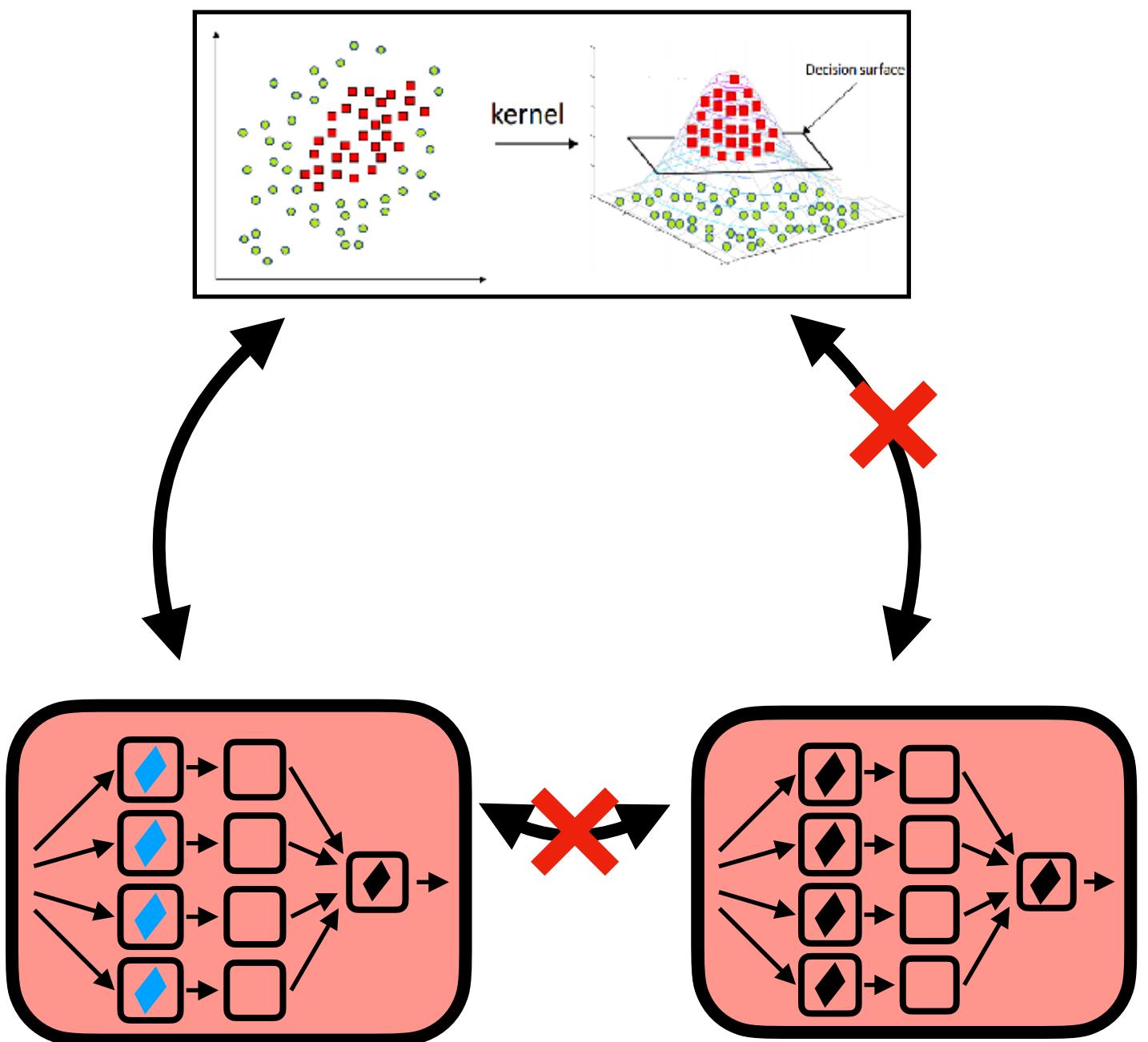
# Separation with 2-layer NNs

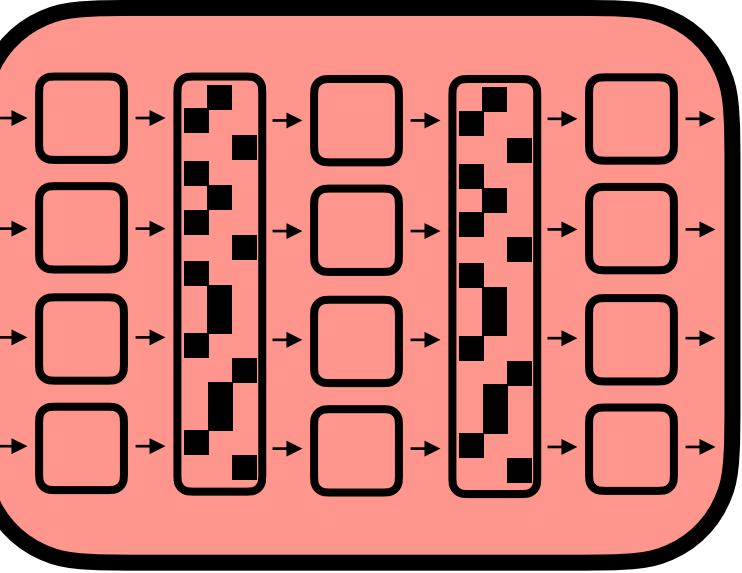
**Contrast:** Sinusoidal target  $h(x) = \sin(\sqrt{d}v^\top x)$ .

- All RFMs that approximate  $h$  have large width:  $m = \exp(d)$ .
- But, 2-layer NN can approximate  $h$  with small width:  $m = \text{poly}(d)$ .

**Implication:** NNs can represent “simple” targets; RFMs cannot.

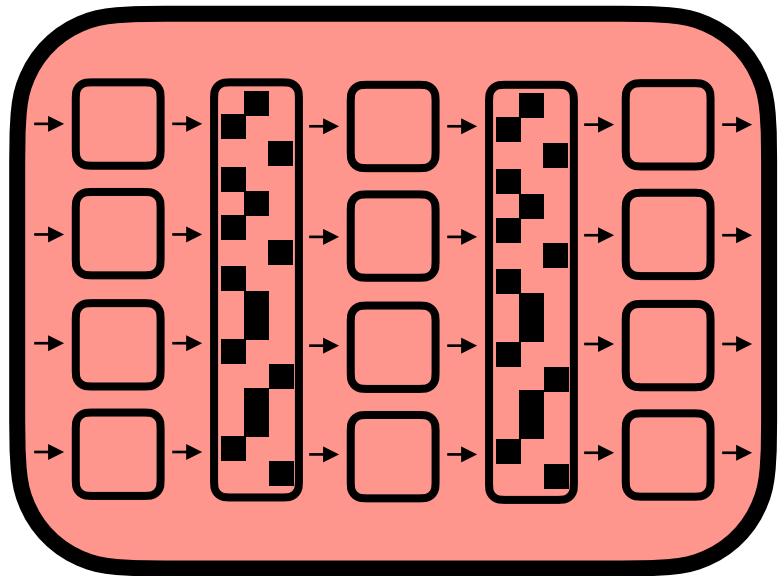
- Similar optimization conclusions [e.g. Damian, Lee, Soltanolkotabi '22].



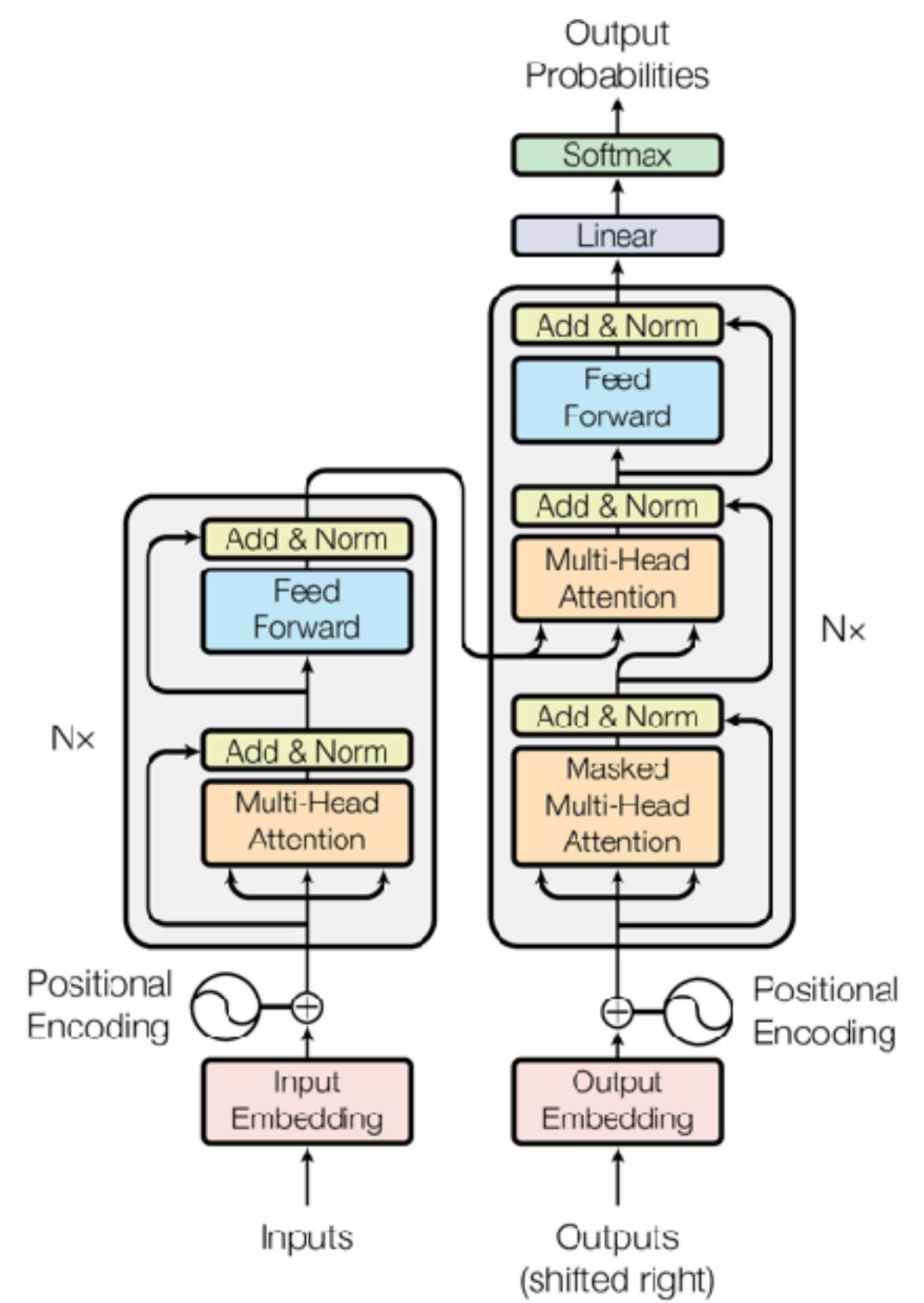


# Vignette #2: Transformers and Sequential Architectures

# Transformer architecture

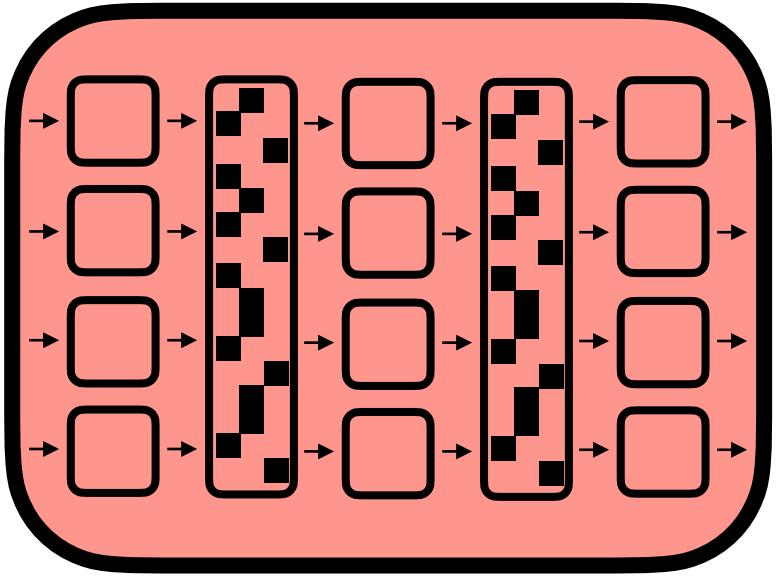


- Backbone of modern large language models.
- **Sequence-to-sequence** architecture.
  - Sequence length  $N \gg$  embedding dim  $m$ .
  - Large sequence length  $N$  in practice:  
32K GPT-4, 100K Claude, 1M Gemini.



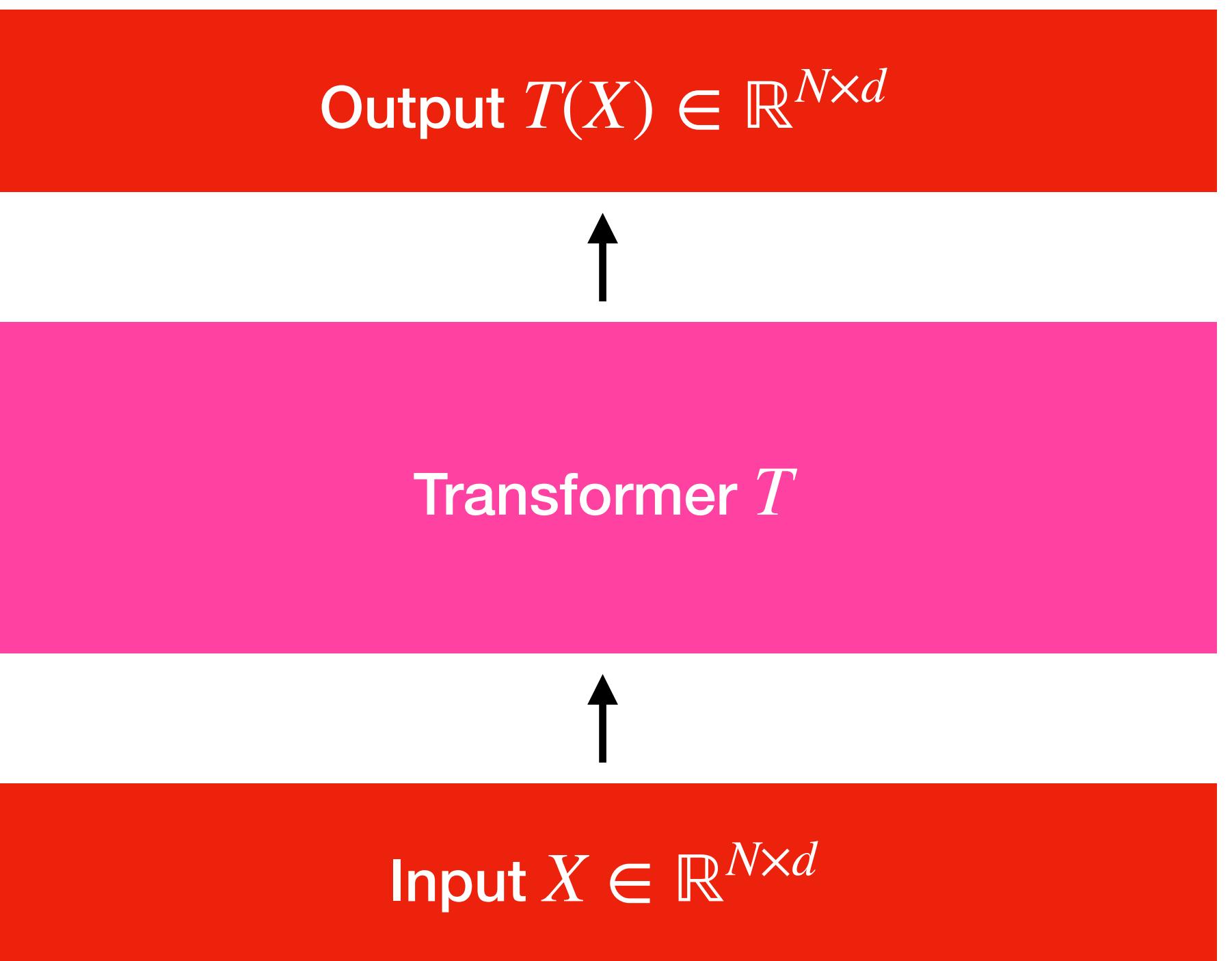
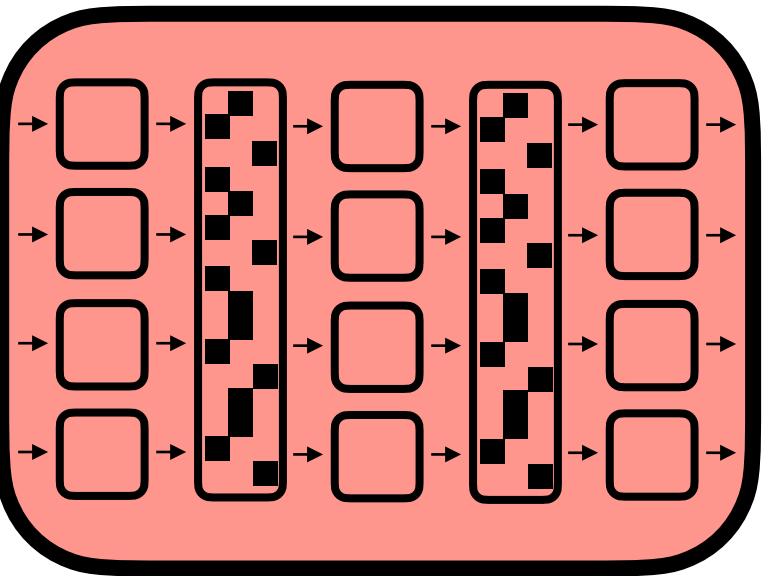
Vaswani et al '17

# Motivating questions

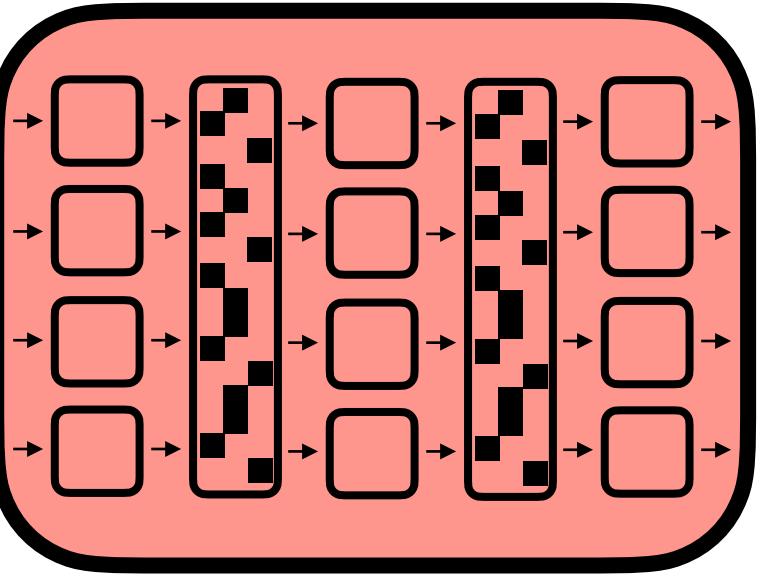


1. Are RNNs and “shortcut transformers” as powerful as standard transformers?
2. How to scale transformers (width and depth)?

# Transformer architecture



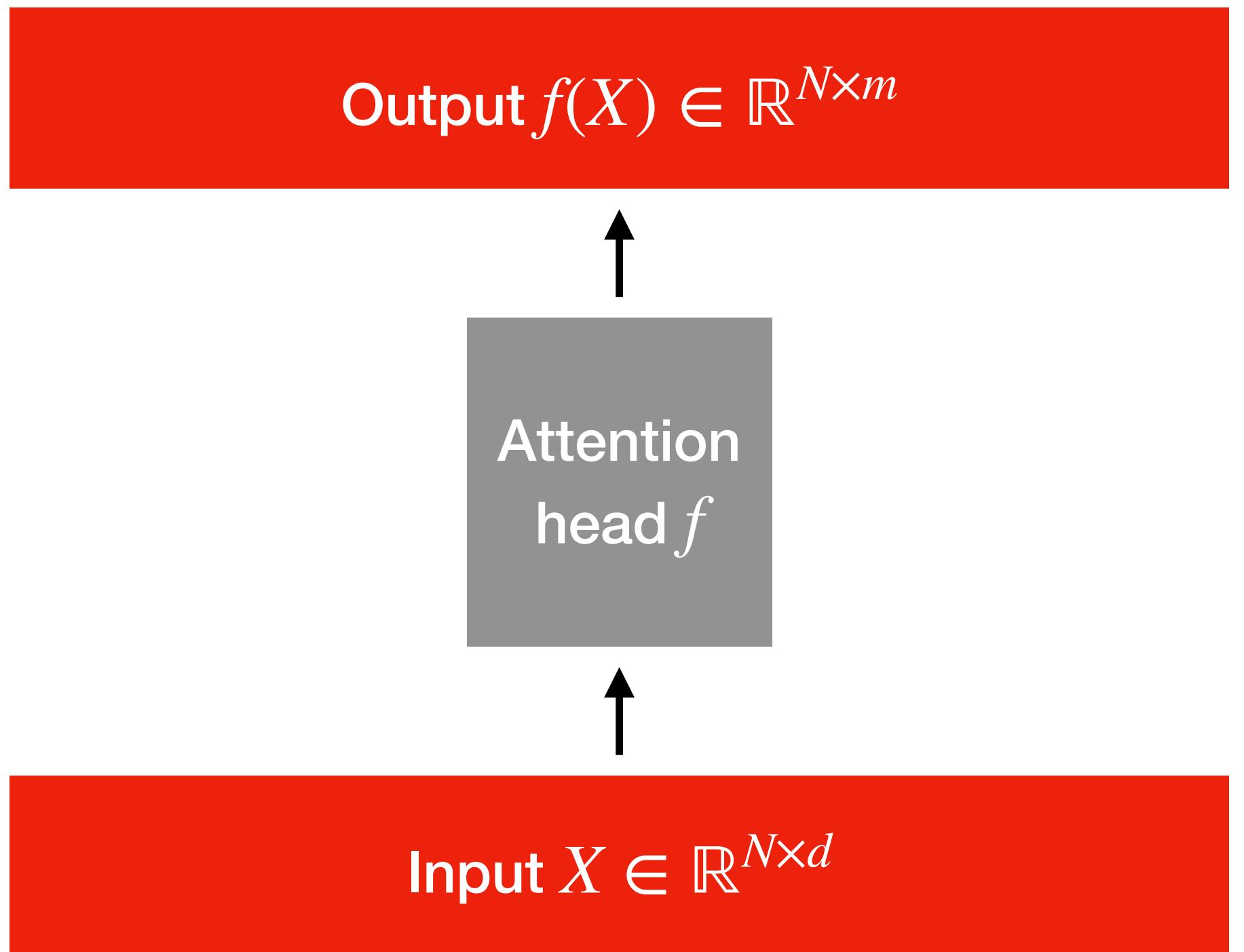
# Transformer architecture



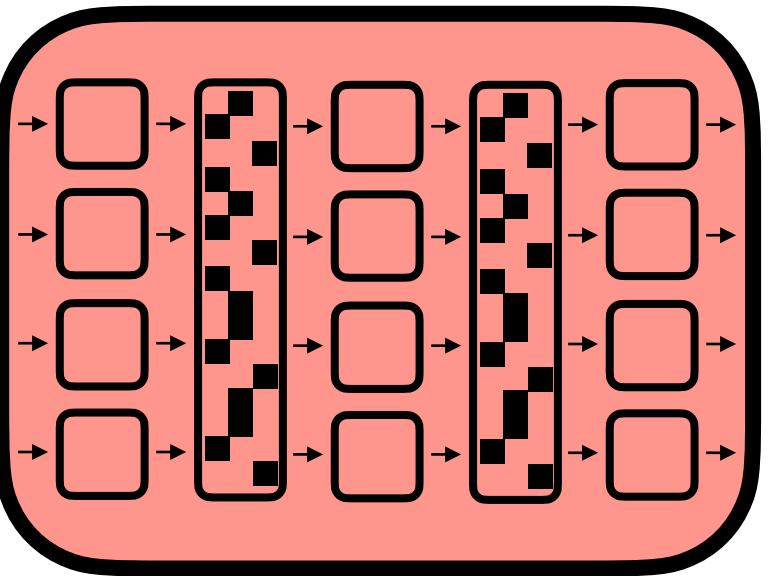
**Attention head:**

$$f(X) = \text{softmax}(XQK^TX^T)XV.$$

Parameters:  $Q, K, V \in \mathbb{R}^{d \times m}$ .



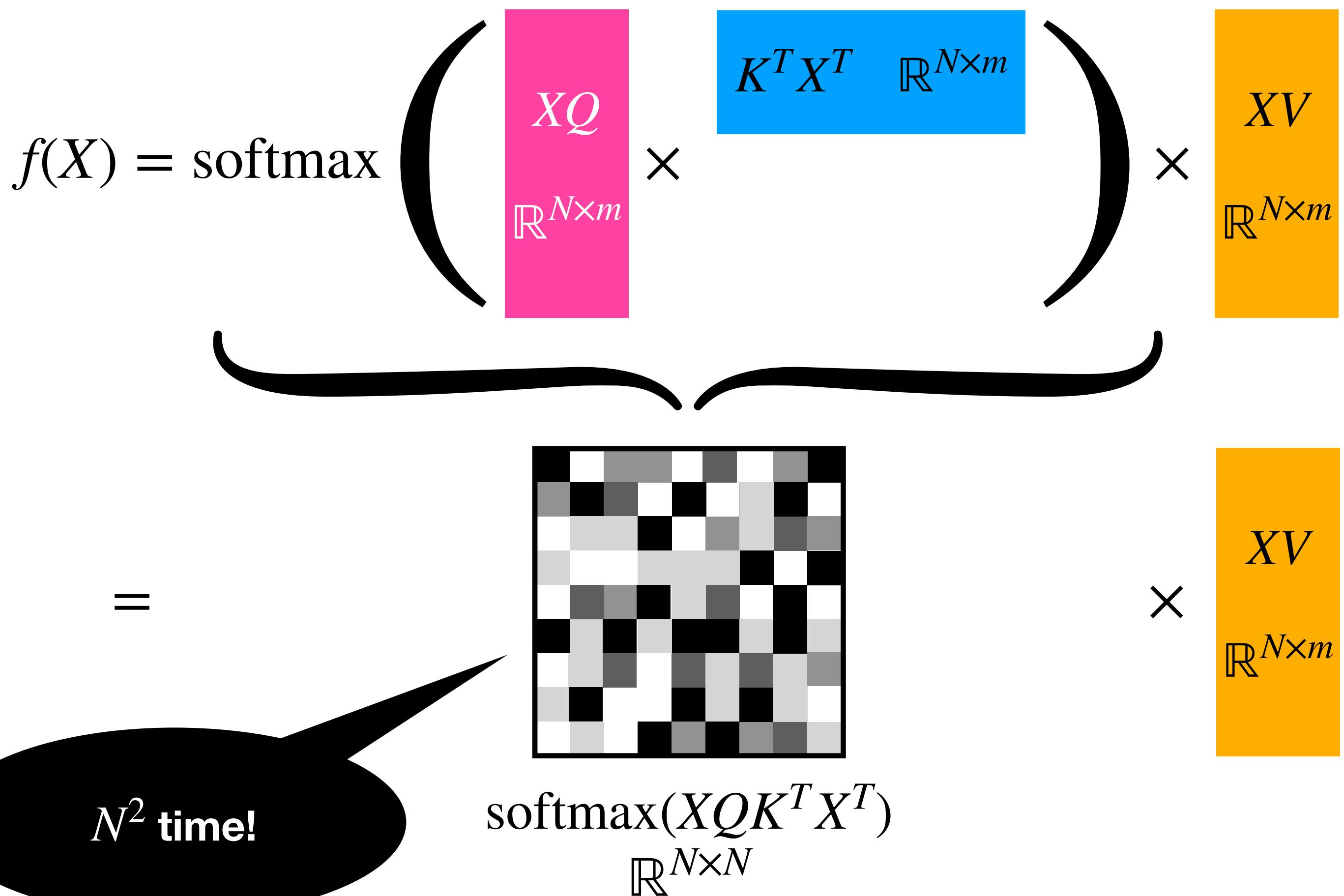
# Transformer architecture



**Attention head:**

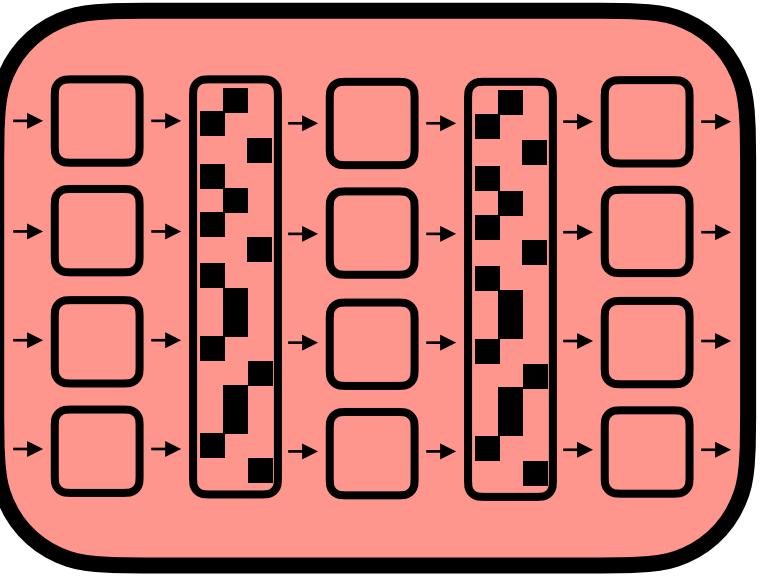
$$f(X) = \text{softmax}(XQK^TX^T)XV.$$

Parameters:  $Q, K, V \in \mathbb{R}^{d \times m}$ .



$N^2$  time!

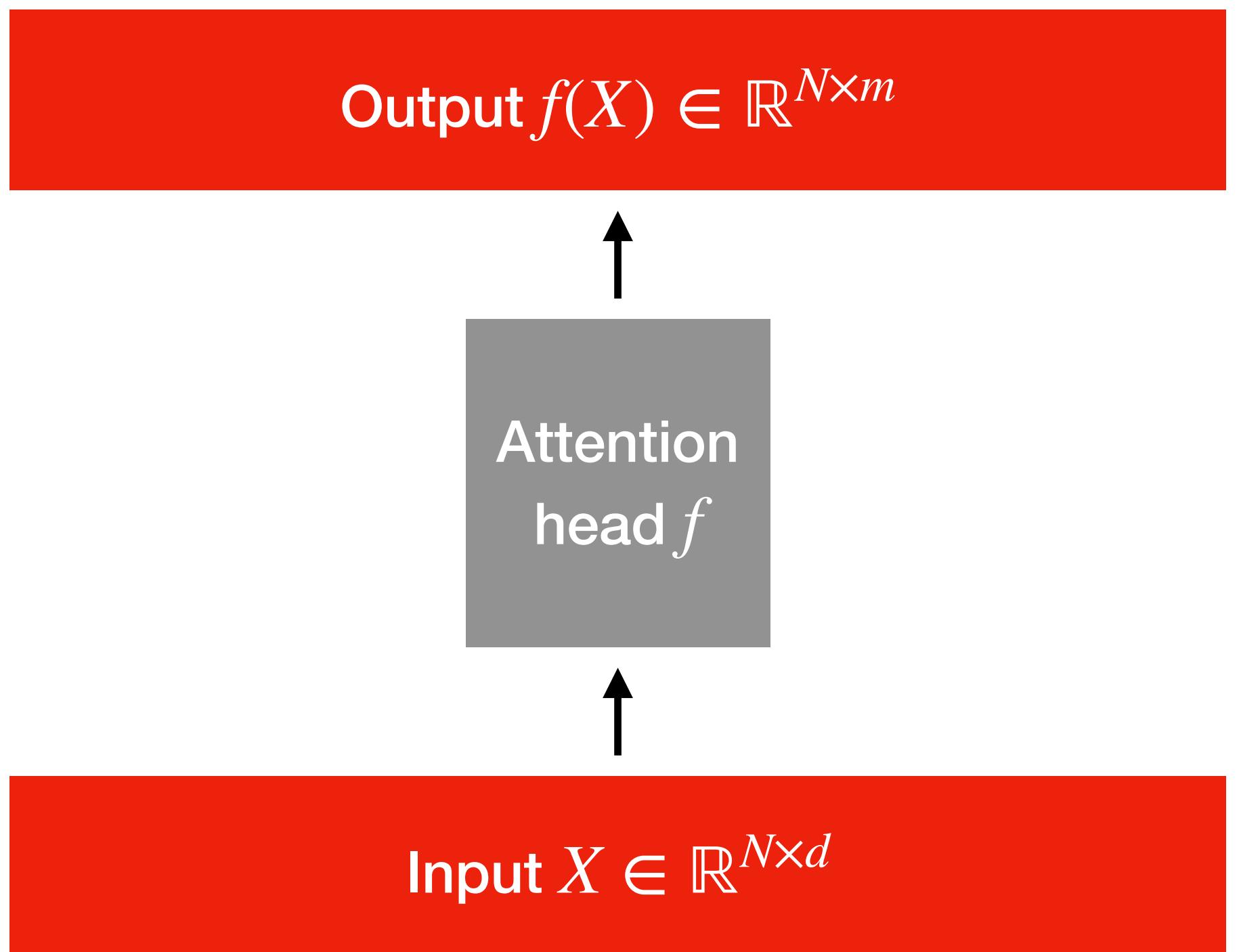
# Transformer architecture



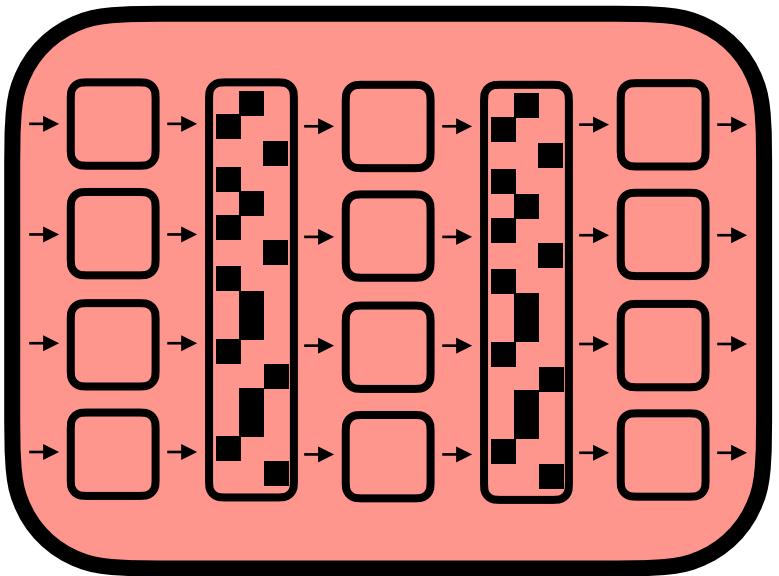
**Attention head:**

$$f(X) = \text{softmax}(XQK^TX^T)XV.$$

Parameters:  $Q, K, V \in \mathbb{R}^{d \times m}$ .



# Transformer architecture



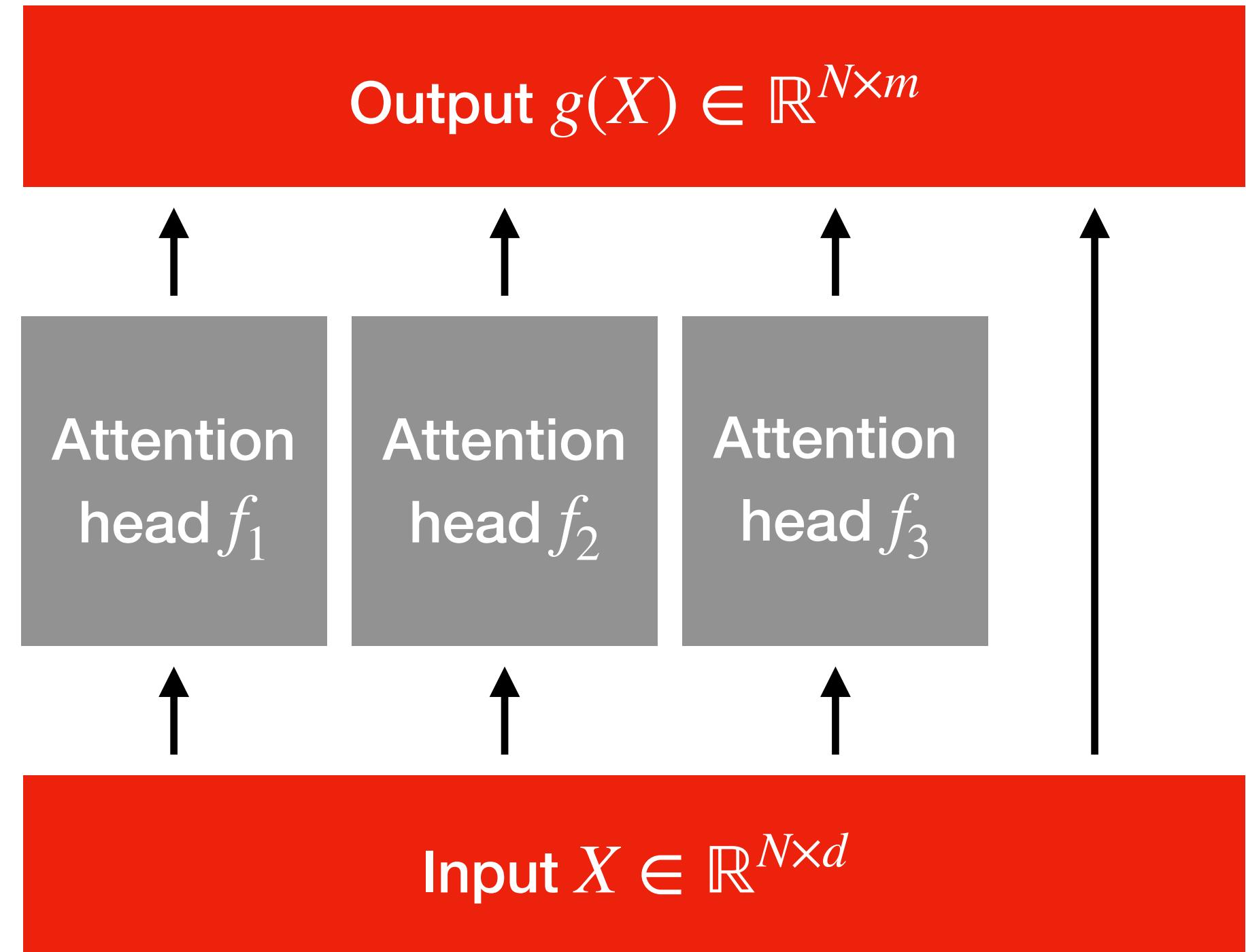
**Attention head:**

$$f(X) = \text{softmax}(XQK^TX^T)XV.$$

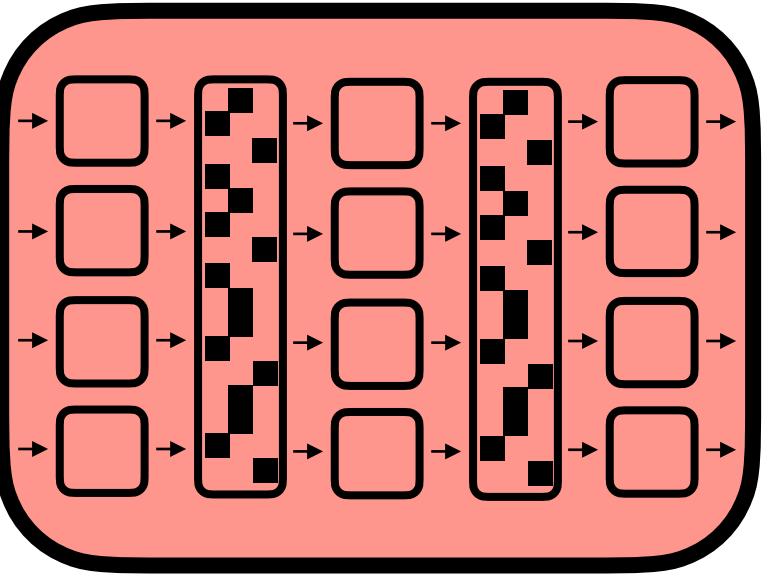
Parameters:  $Q, K, V \in \mathbb{R}^{d \times m}$ .

**Multi-headed attention:**

$$g(X) = X + \sum_{h=1}^H f_h(X).$$



# Transformer architecture



**Attention head:**

$$f(X) = \text{softmax}(XQK^TX^T)XV.$$

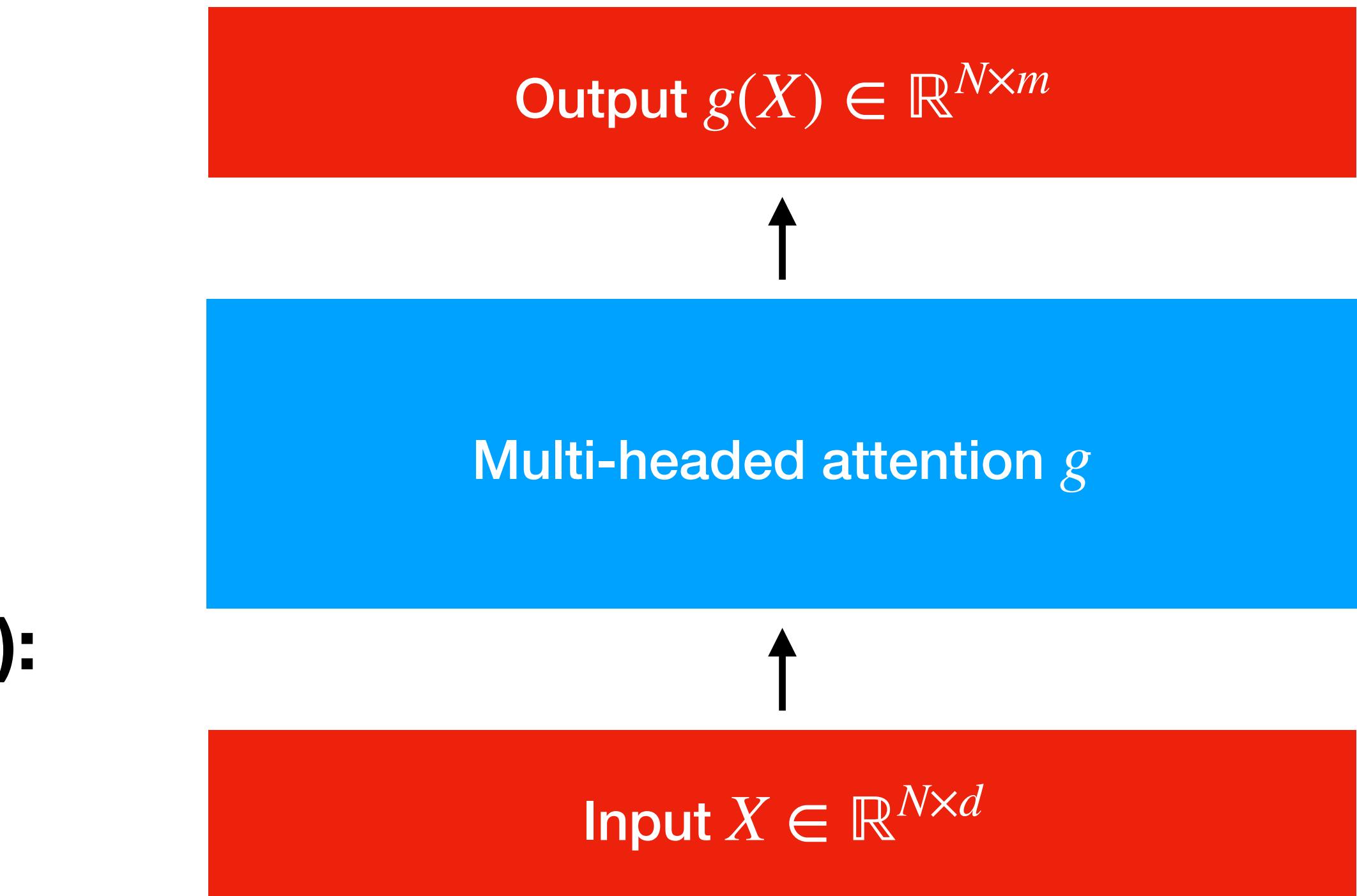
Parameters:  $Q, K, V \in \mathbb{R}^{d \times m}$ .

**Multi-headed attention:**

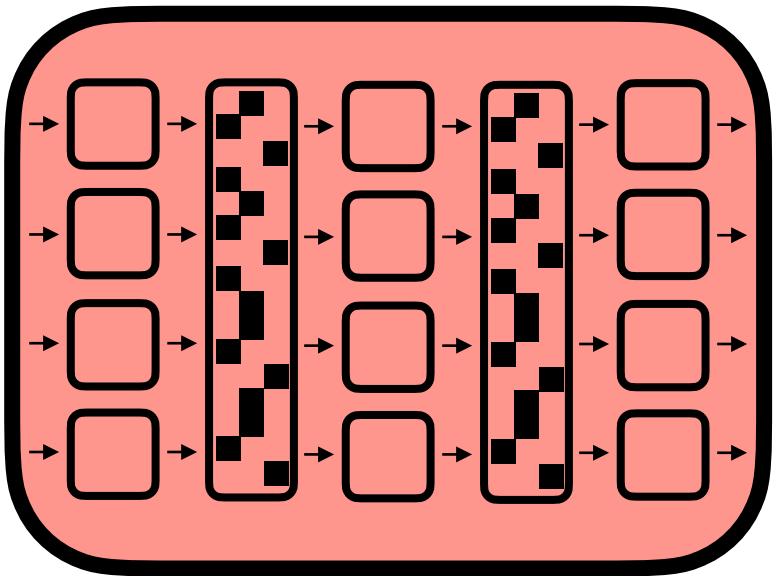
$$g(X) = X + \sum_{h=1}^H f_h(X).$$

**Element-wise multi-layer perceptron (MLP):**

$$\phi(X) = (\phi(x_1), \dots, \phi(x_N)).$$



# Transformer architecture



**Attention head:**

$$f(X) = \text{softmax}(XQK^TX^T)XV.$$

Parameters:  $Q, K, V \in \mathbb{R}^{d \times m}$ .

**Multi-headed attention:**

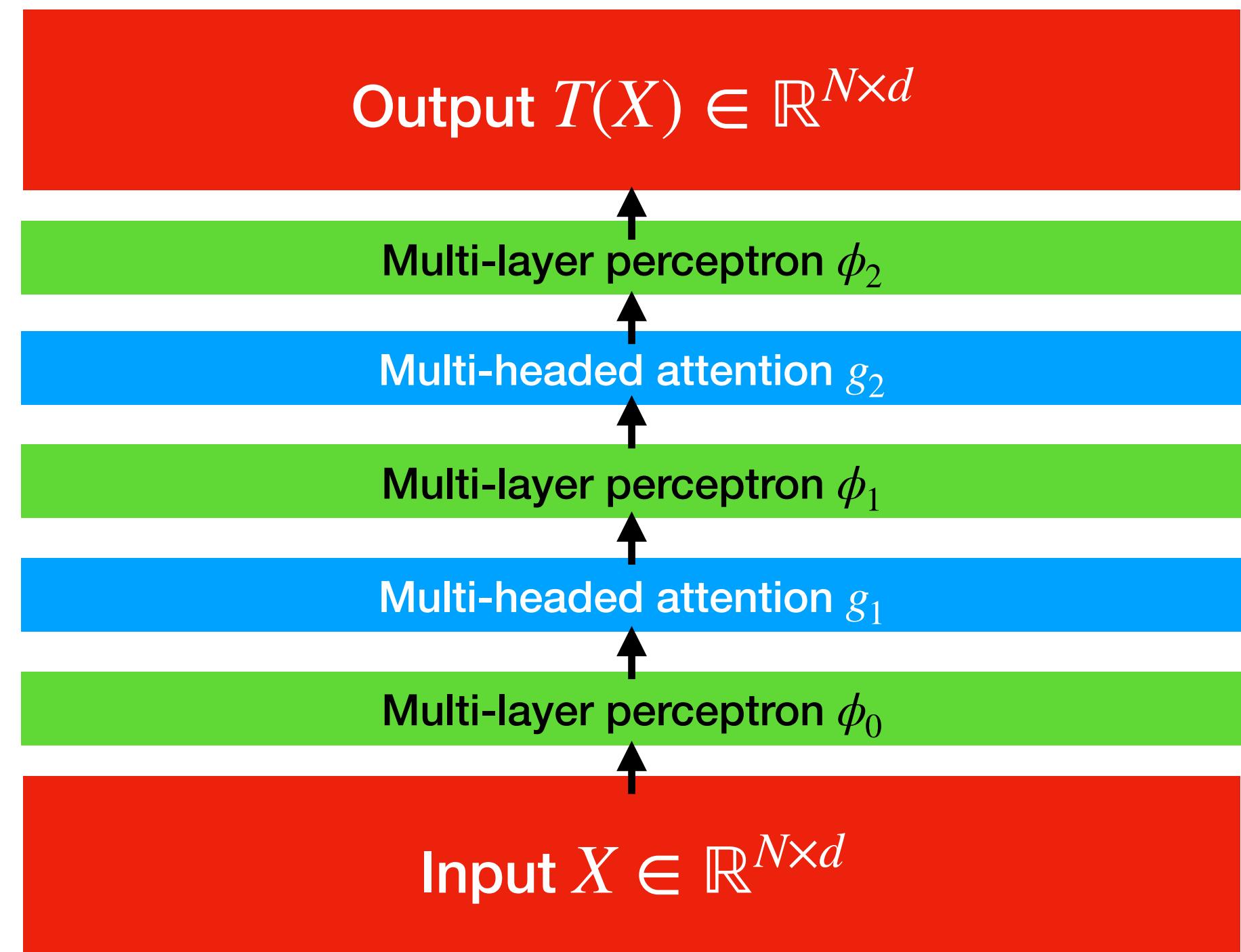
$$g(X) = X + \sum_{h=1}^H f_h(X).$$

**Element-wise multi-layer perceptron (MLP):**

$$\phi(X) = (\phi(x_1), \dots, \phi(x_N)).$$

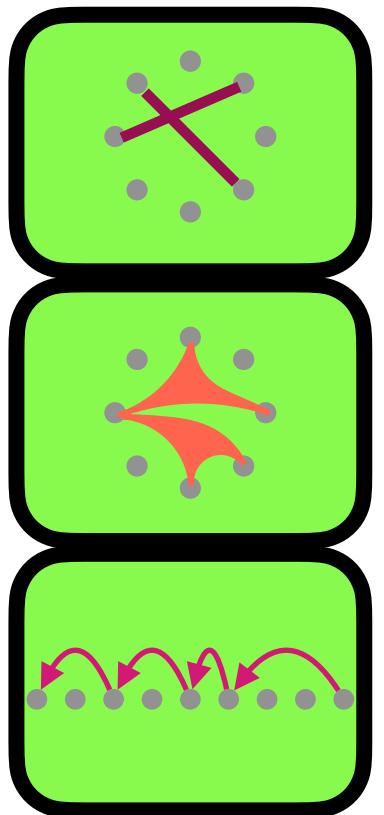
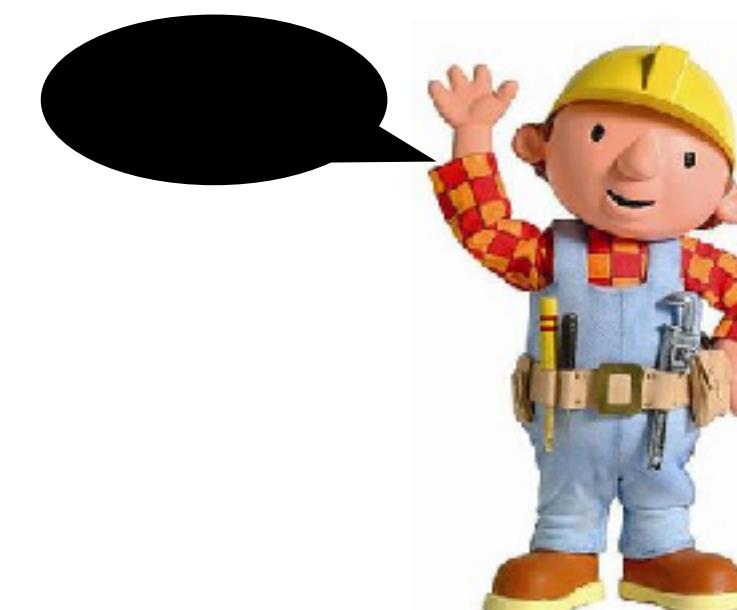
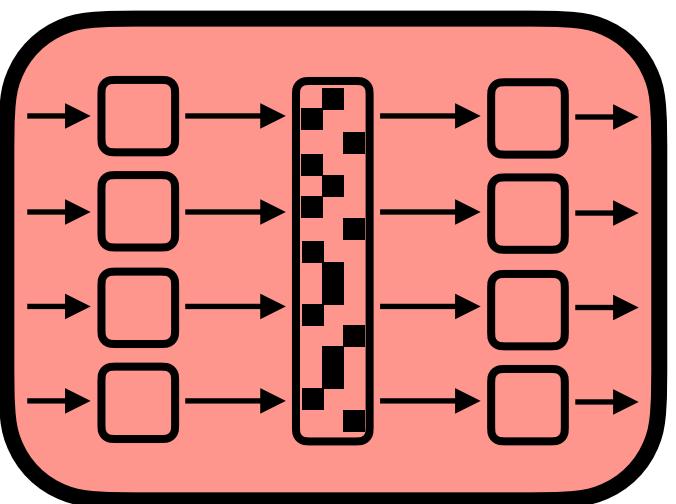
**Full transformer:**

$$T(X) = (\phi_L \circ g_L \circ \dots \circ g_1 \circ \phi_0)(X).$$

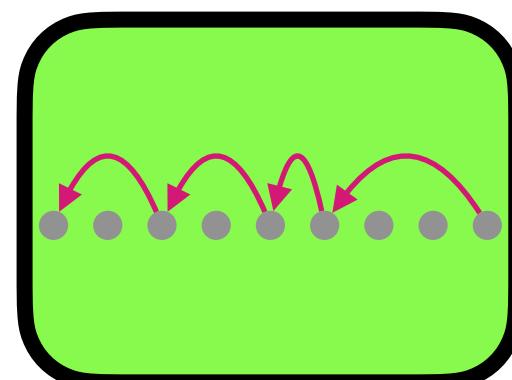
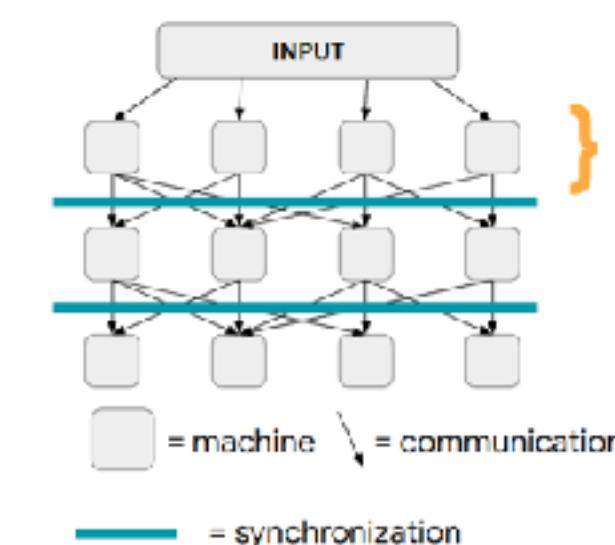
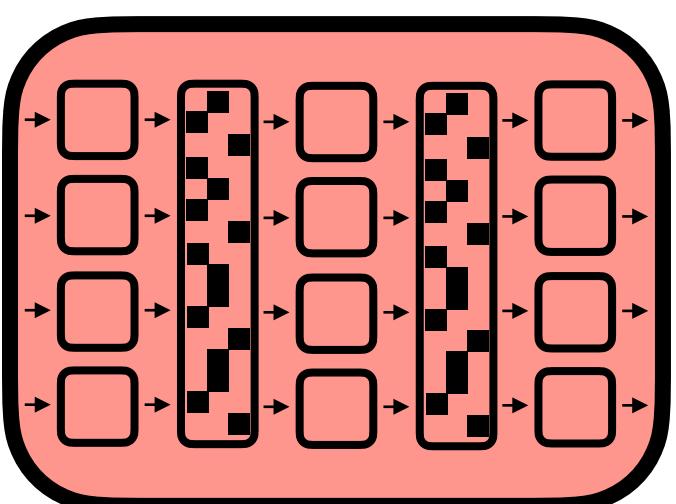


# Transformers and communication

**Relationship #1:** 1-layer  $m$ -width transformers are  $m$ -bit Alice-Bob communication protocols.



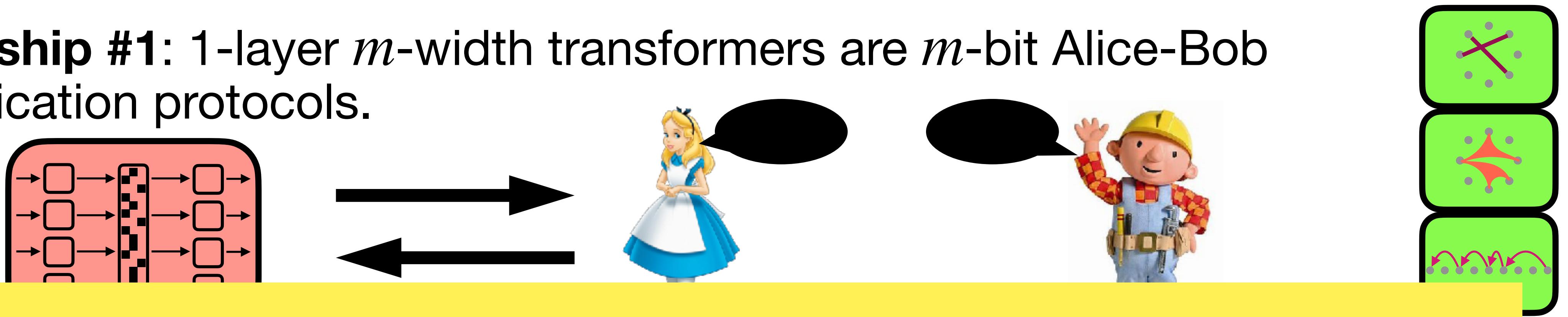
**Relationship #2:**  $L$ -layer  $m$ -width transformers are  $L$ -round Massively Parallel Computation (MPC) protocols with  $m$  local memory,



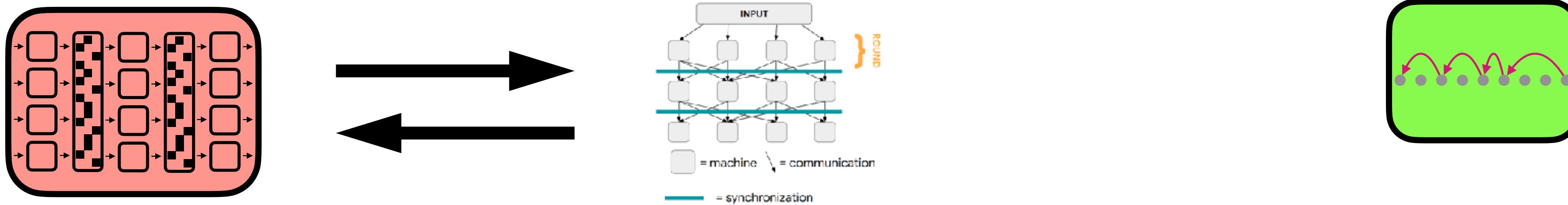
# Contributions

Connections between transformers and distributed computational models.

**Relationship #1:** 1-layer  $m$ -width transformers are  $m$ -bit Alice-Bob communication protocols.



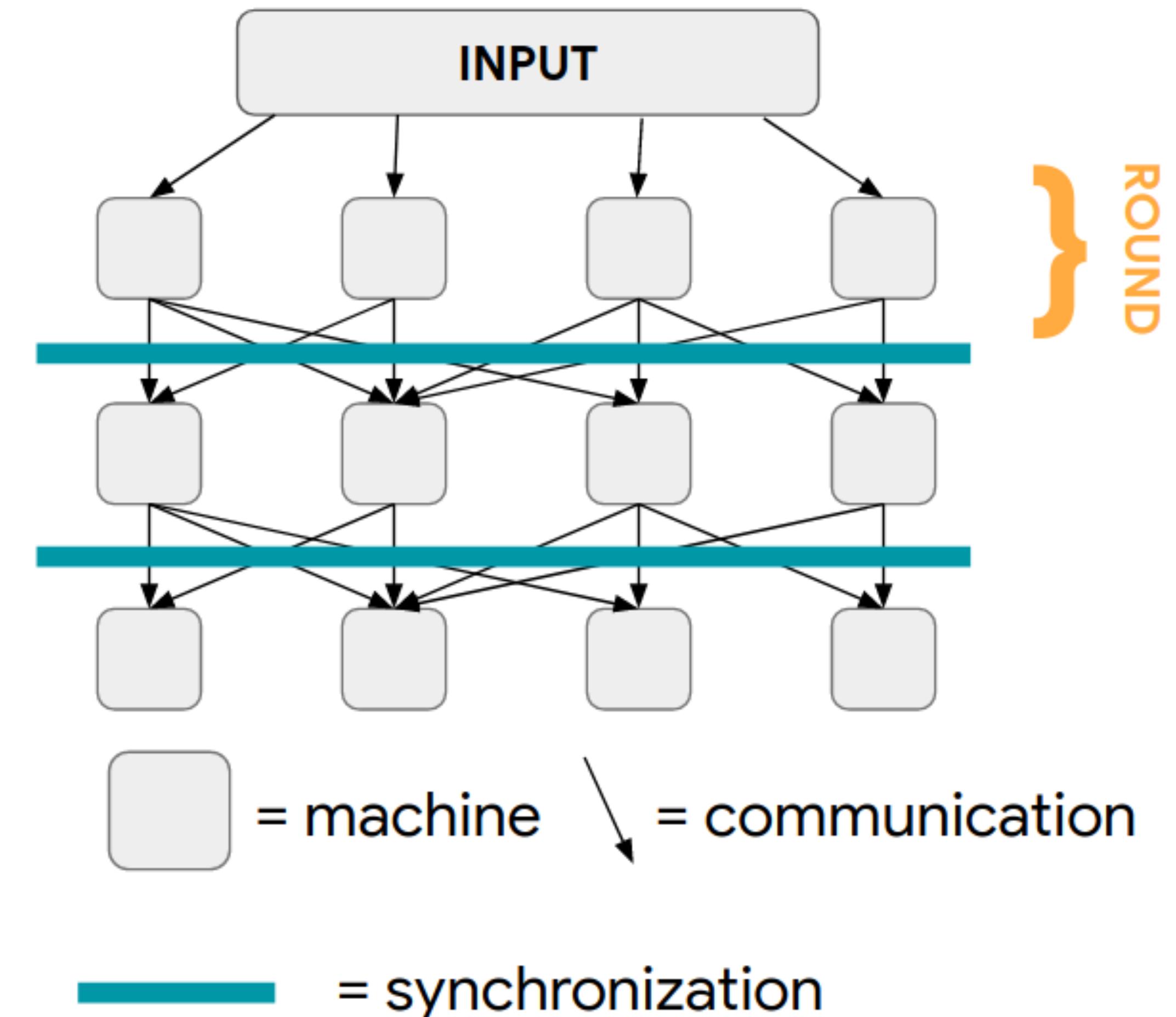
**Relationship #2:**  $L$ -layer  $m$ -width transformers are  $L$ -round Massively Parallel Computation (MPC) protocols with  $m$  local memory,



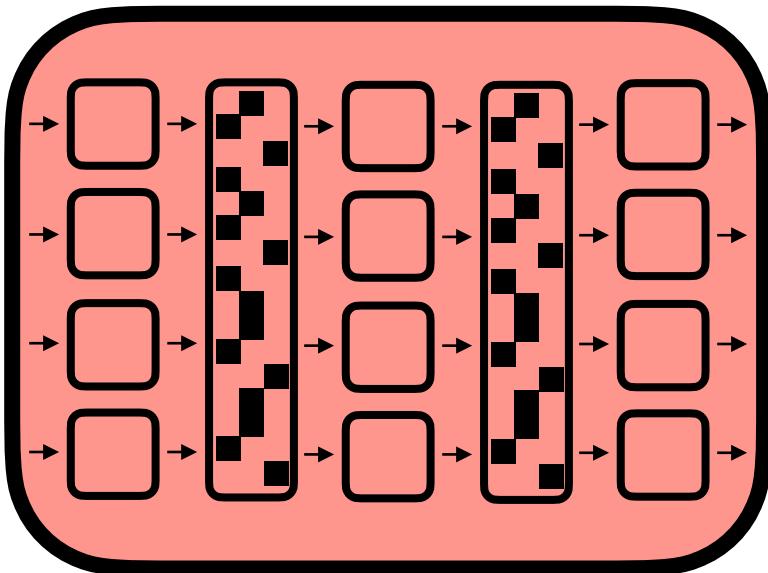
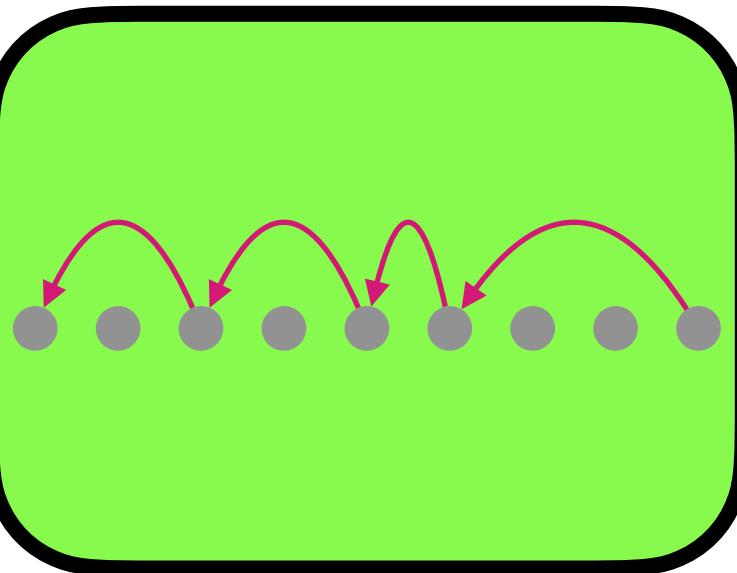
# Massively Parallel Computation model (MPC)

## Computational model of MapReduce

- Input divided among  $q$  machines with local memory  $s$ .
- Round  $r = 1, \dots, R$ :
  - Each machine performs computations on local memory.
  - Each machine sends and receives  $\leq s$  bits of information.

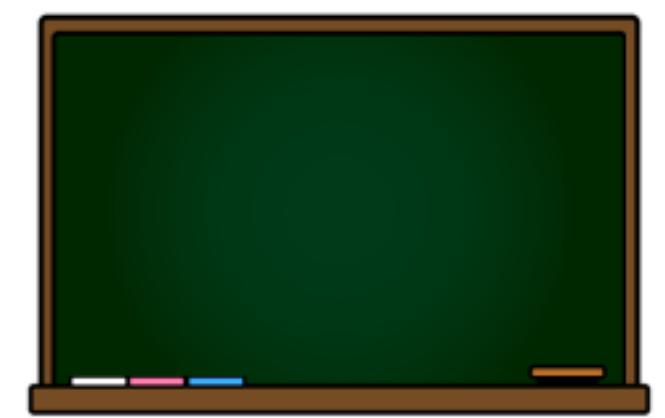
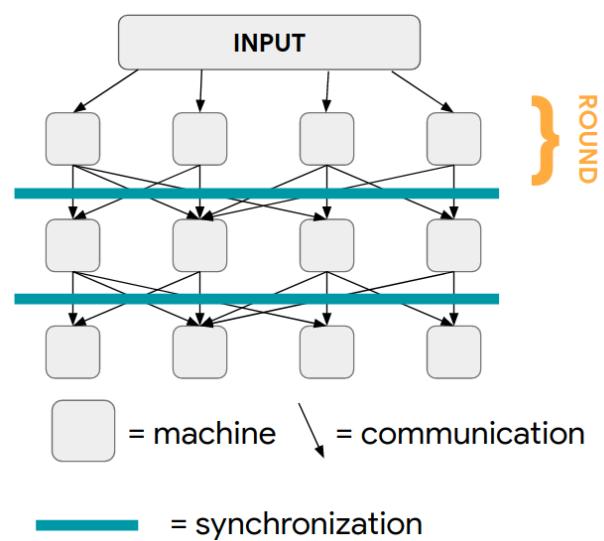
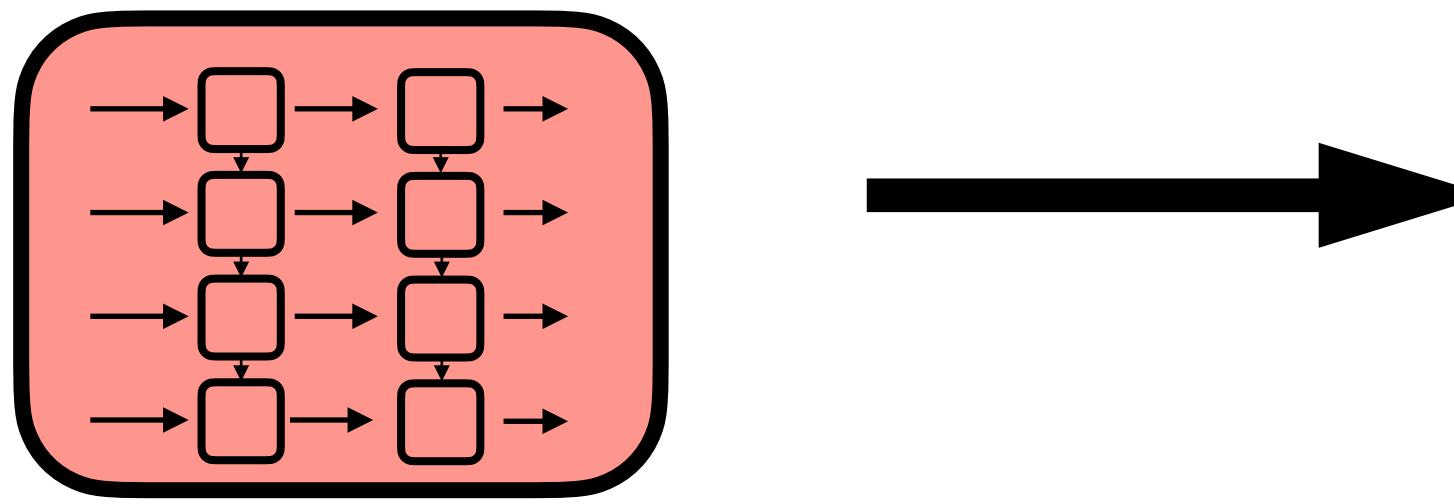
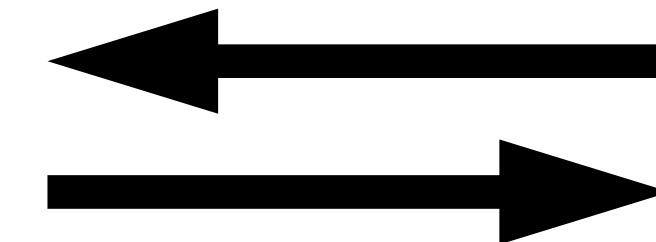
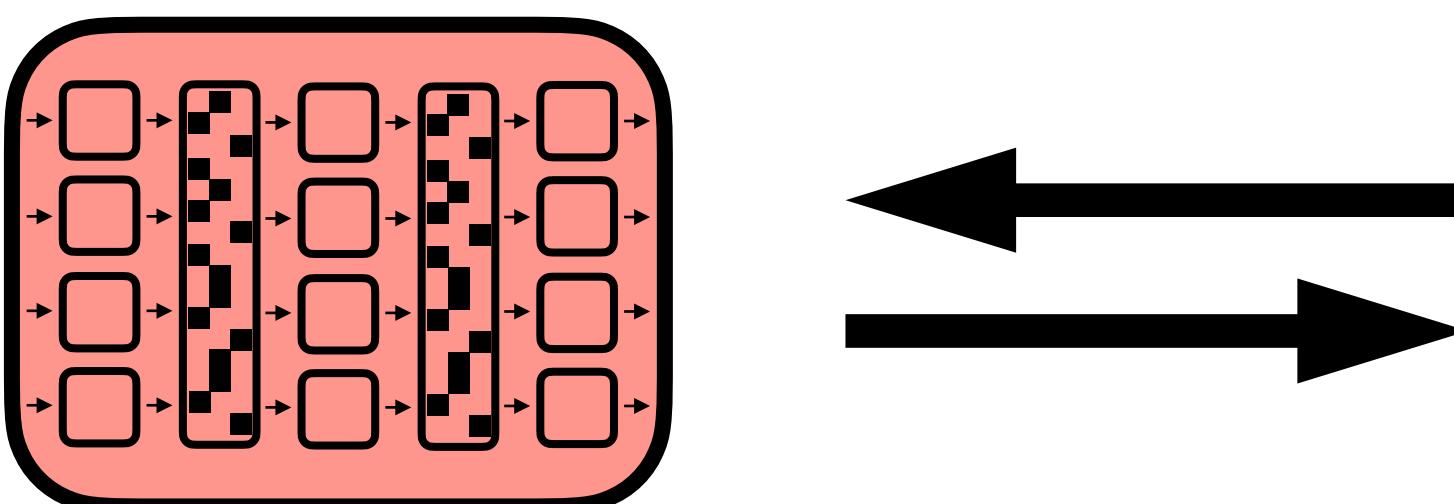


# Transformers and distributed computing

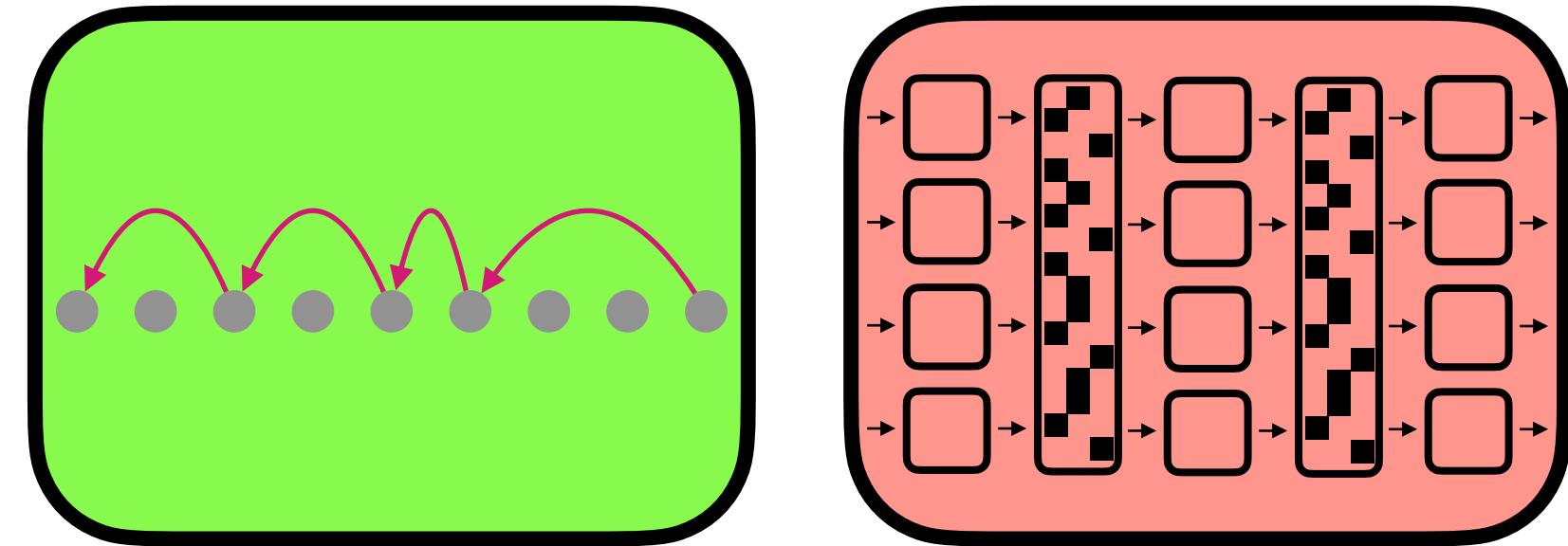


## Broader equivalences

- Transformers can simulate MPC algorithms.
  - MPC algorithms can simulate transformers.
  - Serial blackboard model can simulate alternative models (RNNs, sub-quadratic transformers).
- ➡ Alternatives are not parallelizable!



# Depth separation (1 vs 2)



**Induction Heads:** Most recent bigram match.

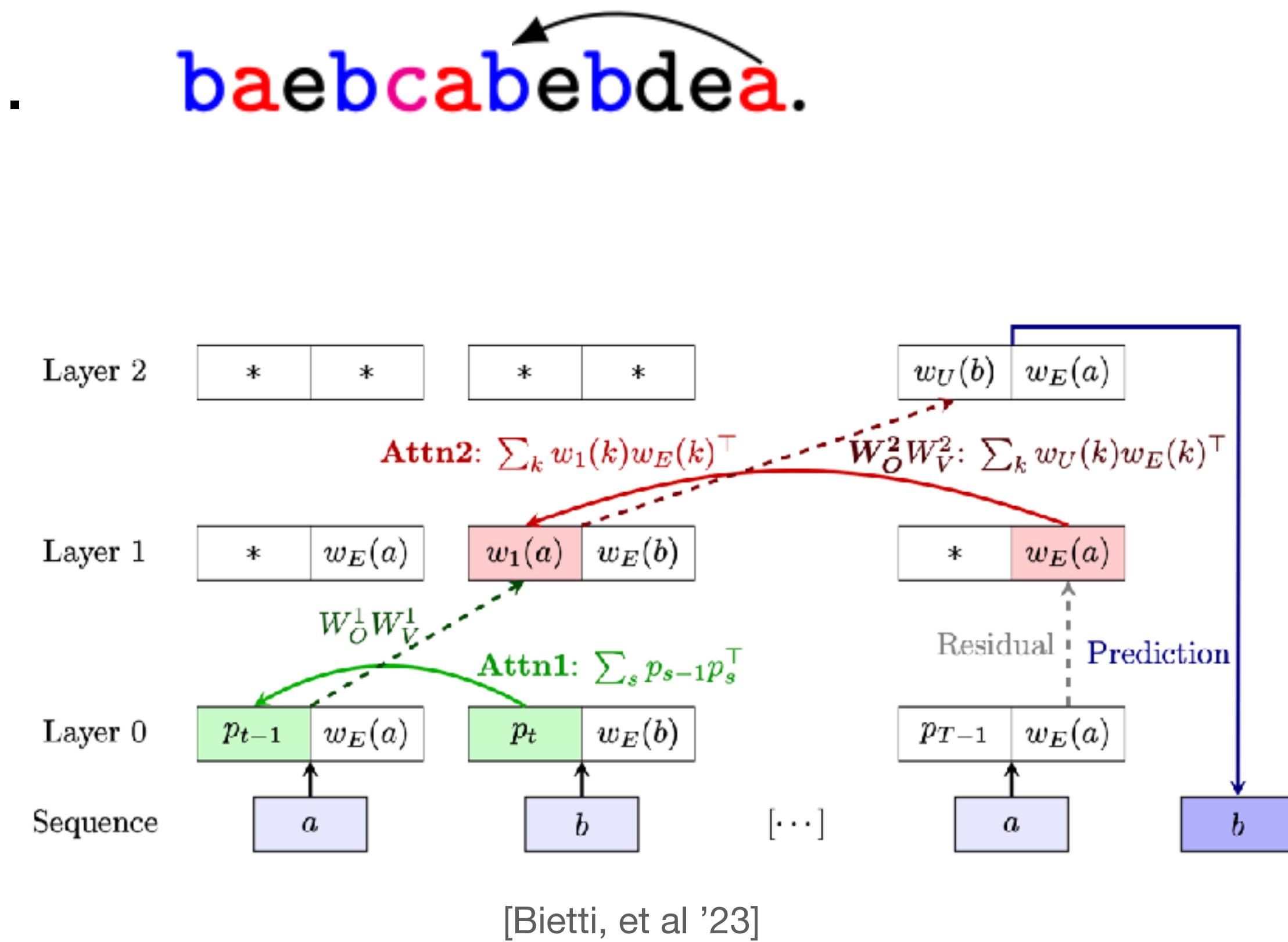
$$\text{IH}(\dots ab\dots a)_N = b$$

**Positive result** [Bietti, et al '23]:  
A 2-layer transformer efficiently solves IH.

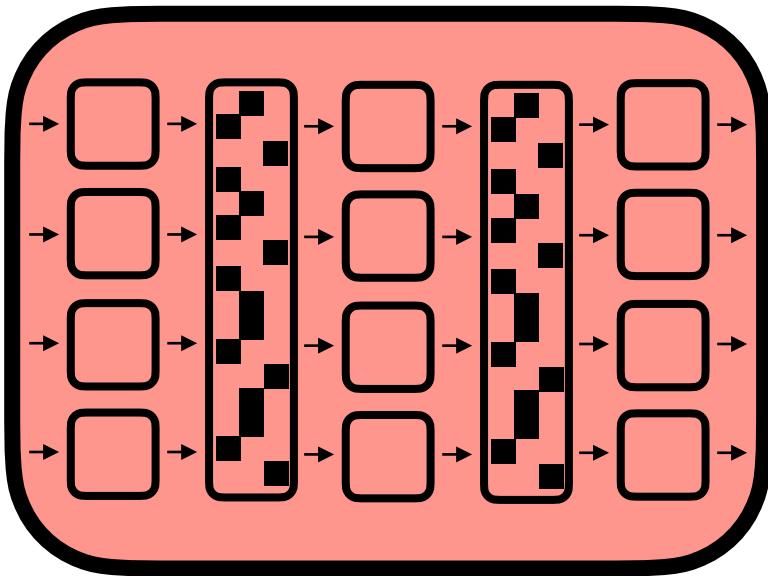
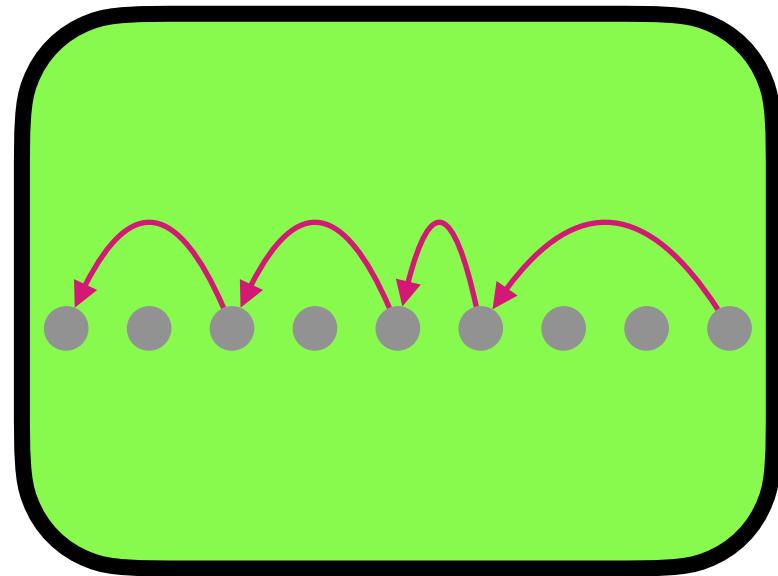
**Negative result:**

1-layer transformers that solve IH  
have width  $m \geq N^{\Omega(1)}$ .

- Proof idea: Communication complexity



# Depth separation ( $\log k$ )



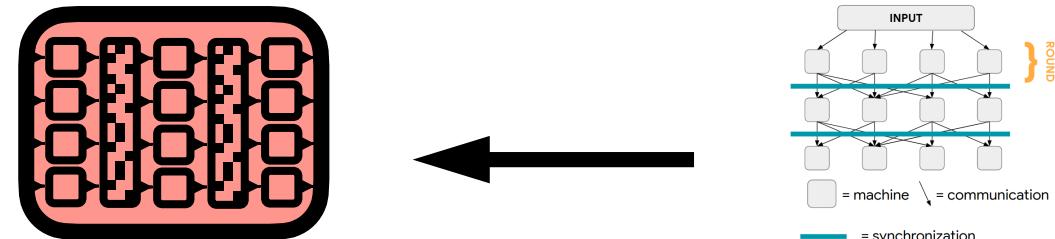
**$k$ -Induction Heads:** Composition of IH.

**baeb**c**abeb**d**ea.**

$$k\text{-IH}(\dots a_k a_{k+1} \dots a_{k-1} a_k \dots \dots a_2 a_3 \dots a_1 a_2 \dots a_1) = a_{k+1}$$

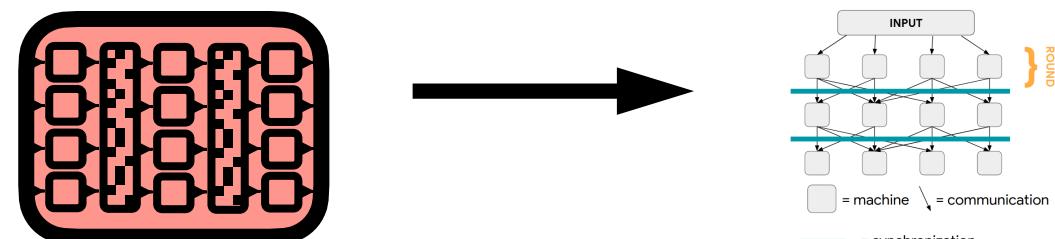
**Positive result:** A transformer efficiently solves  $k$ -IH and has depth  $L \leq O(\log k)$ .

- Proof idea: MPC algorithm that combines partial solutions.



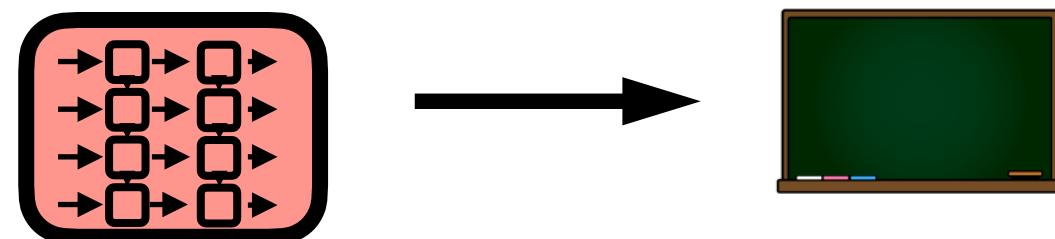
**Negative result:** Transformers that efficiently solve  $k$ -IH have depth  $L \geq \Omega(\log k)$ .

- Proof idea: MPC cycle identification lower bound conjecture.

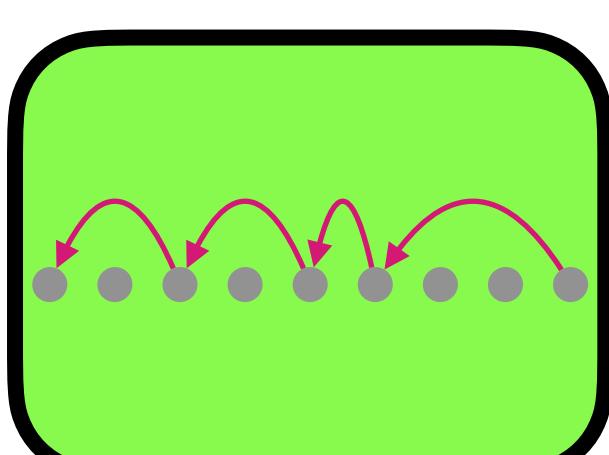
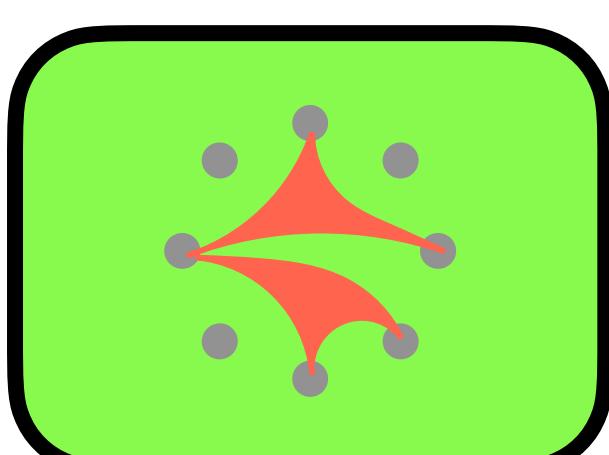
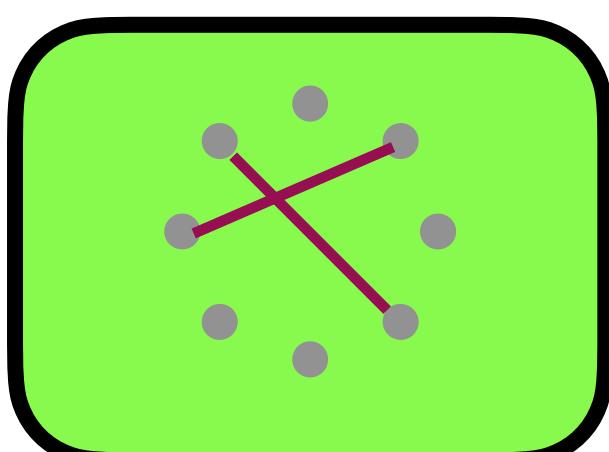
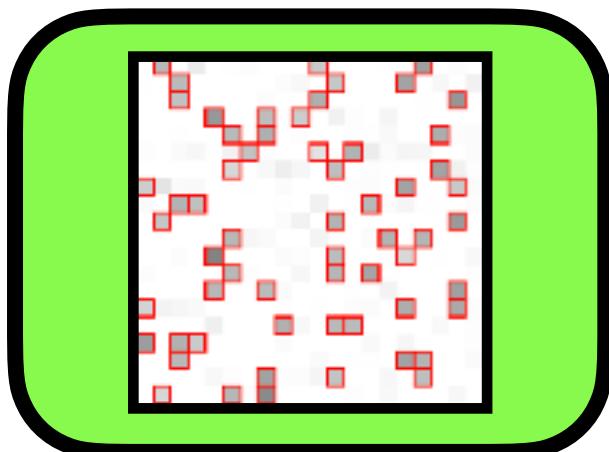


**Contrast:** RNNs and sub-quad. transformers that efficiently solve  $k$ -IH have depth  $L \geq k$ .

- Proof idea: Pointer chasing communication lower bounds.



# Tasks



# Takeaways

Transformers vs RNNs/  
sub-quadratic separations

Contrast

Expressivity of  
embedding dimension  $m$

Positive result  
Negative result

Transformer layers are  
inherently pairwise

Negative result  
Contrast

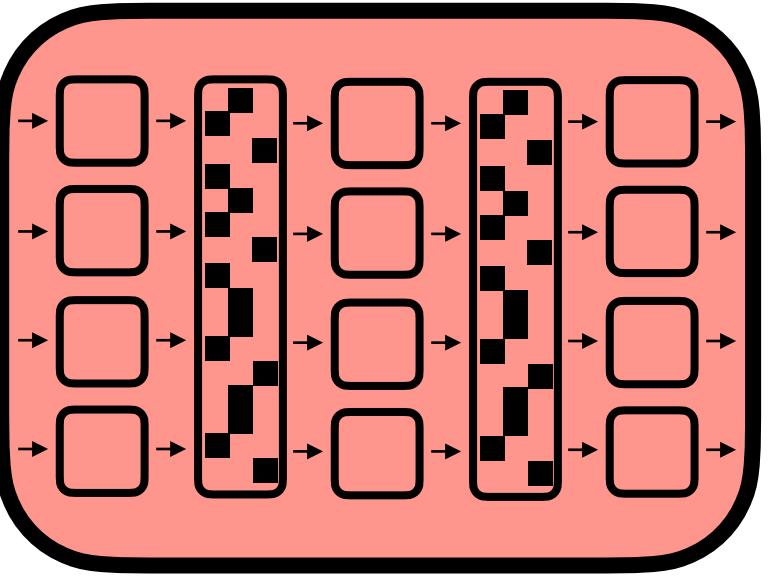
Expressivity of  
depth  $L$

Positive result  
Negative result

Efficient computation of  
compositional tasks

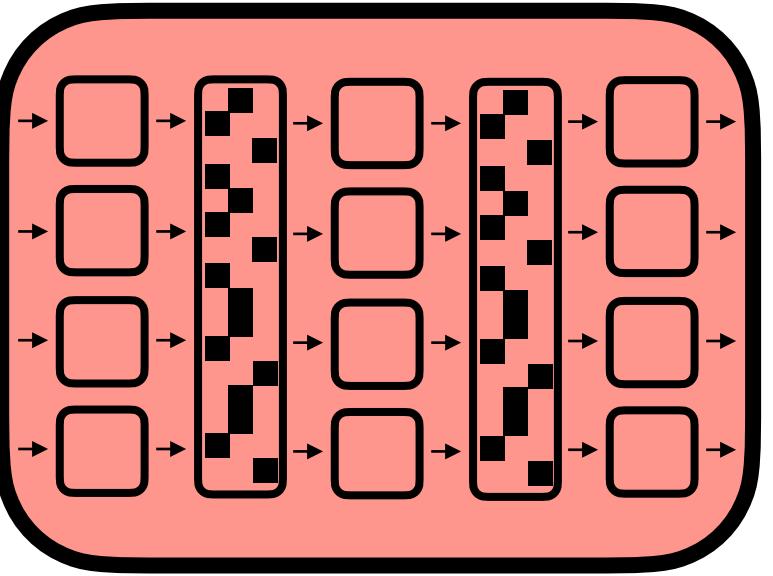
Positive result

# Implications



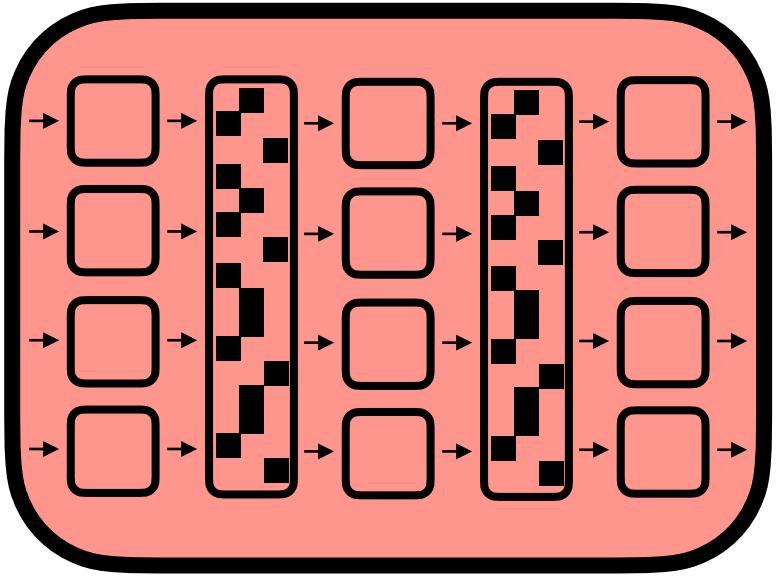
1. Are RNNs and sub-quadratic models as powerful as standard transformers?
2. How to scale transformers (width and depth)?

# Implications



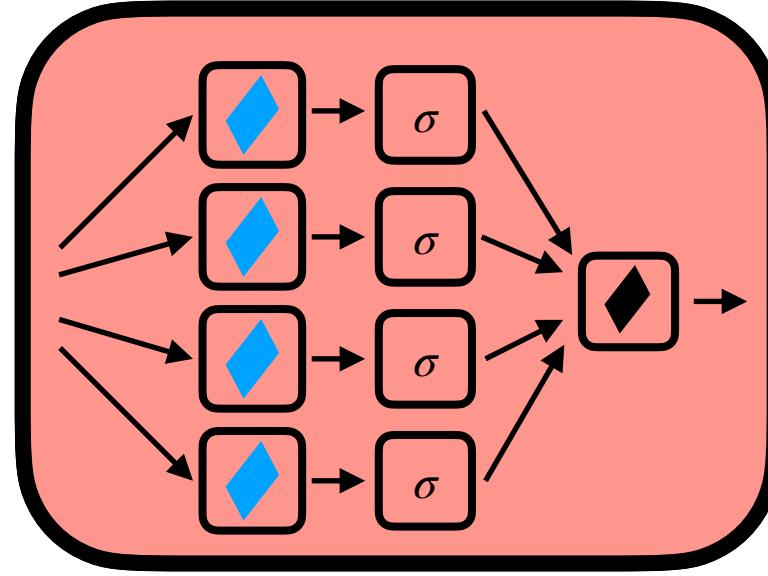
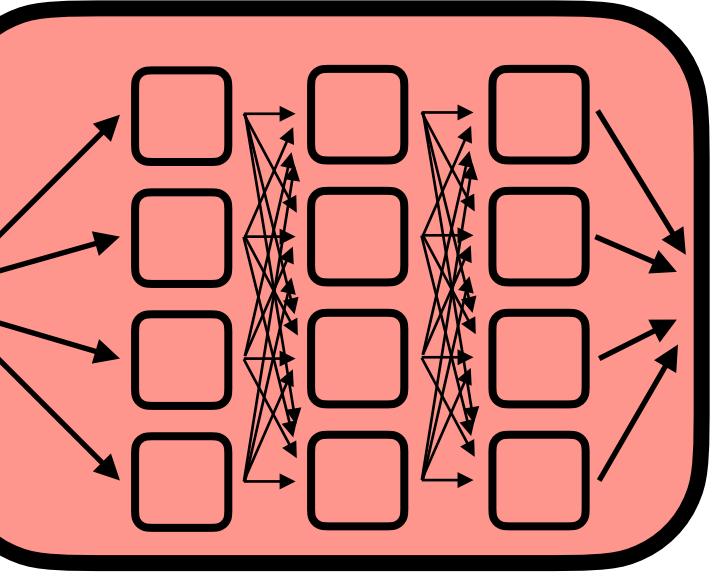
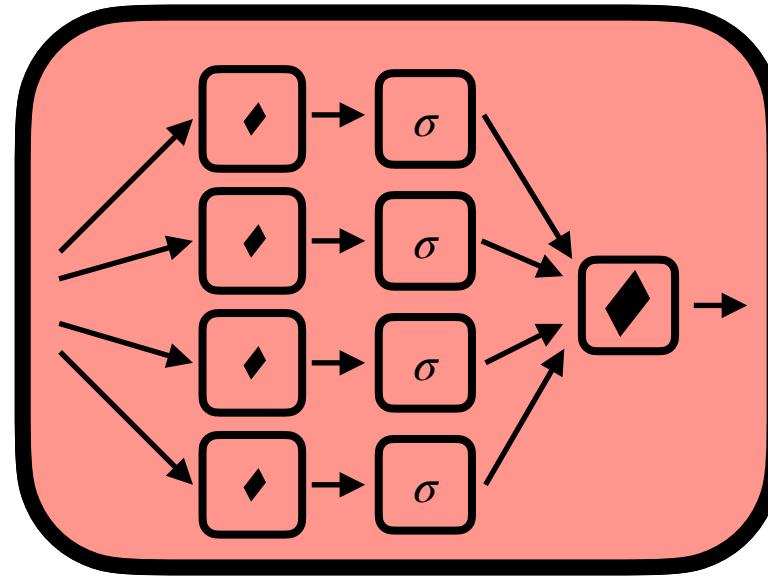
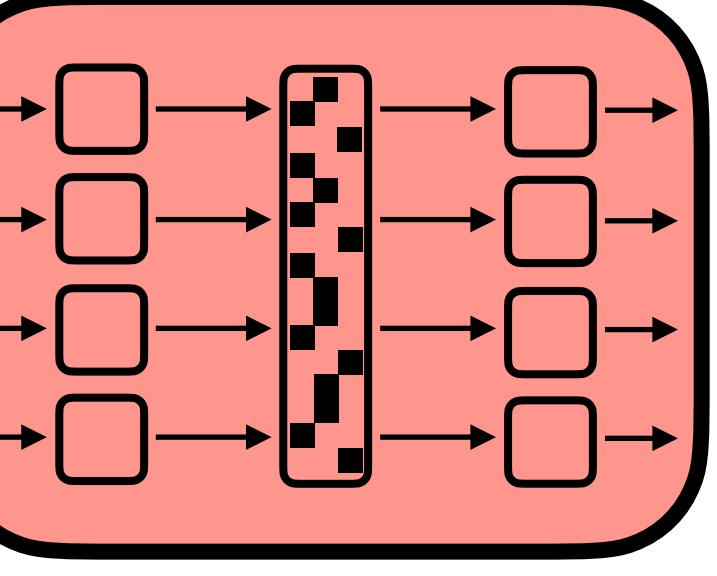
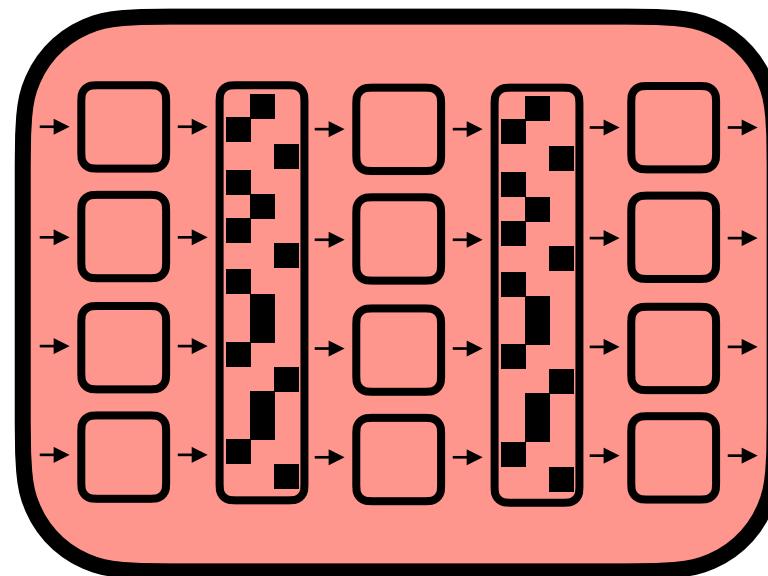
1. Are RNNs and sub-quadratic models as powerful as standard transformers?  
No, evidenced by parallel and pointer chasing ( $k$ -IH) algorithms.
2. How to scale transformers (width and depth)?

# Implications



1. Are RNNs and sub-quadratic models as powerful as standard transformers?  
No, evidenced by parallel and pointer chasing ( $k$ -IH) algorithms.
2. How to scale transformers (width and depth)?  
Communication/distributed computing bounds on both.

# Recap

1. 2-layer random feature NNs  
[Hsu, **S**, Servedio, Vlatakis-Gkaragkounis '21]  

2. Shallow vs deep NNs  
[**S**, Chatziafratis '22]  

3. 2-layer regularized NNs  
[Ardeshir, Hsu, **S** '23]  

4. Single-layer transformers  
[**S**, Hsu, Telgarsky '23]  

5. Multi-layer transformers  
[**S**, Hsu, Telgarsky '24]  


# Final notes

**Primary contributions:** Fundamental trade-offs between architectures.

**Technical contributions:** Novel connections between NNs and rich theoretical CS subfields.

- Dynamical systems, communication complexity, distributed computing, ...

**Beyond representation:**

- Other work about generalization and optimization.
- Future work: Optimization and generalization of transformers.

# Acknowledgements

# Advisors: Daniel Hsu and Rocco Servedio

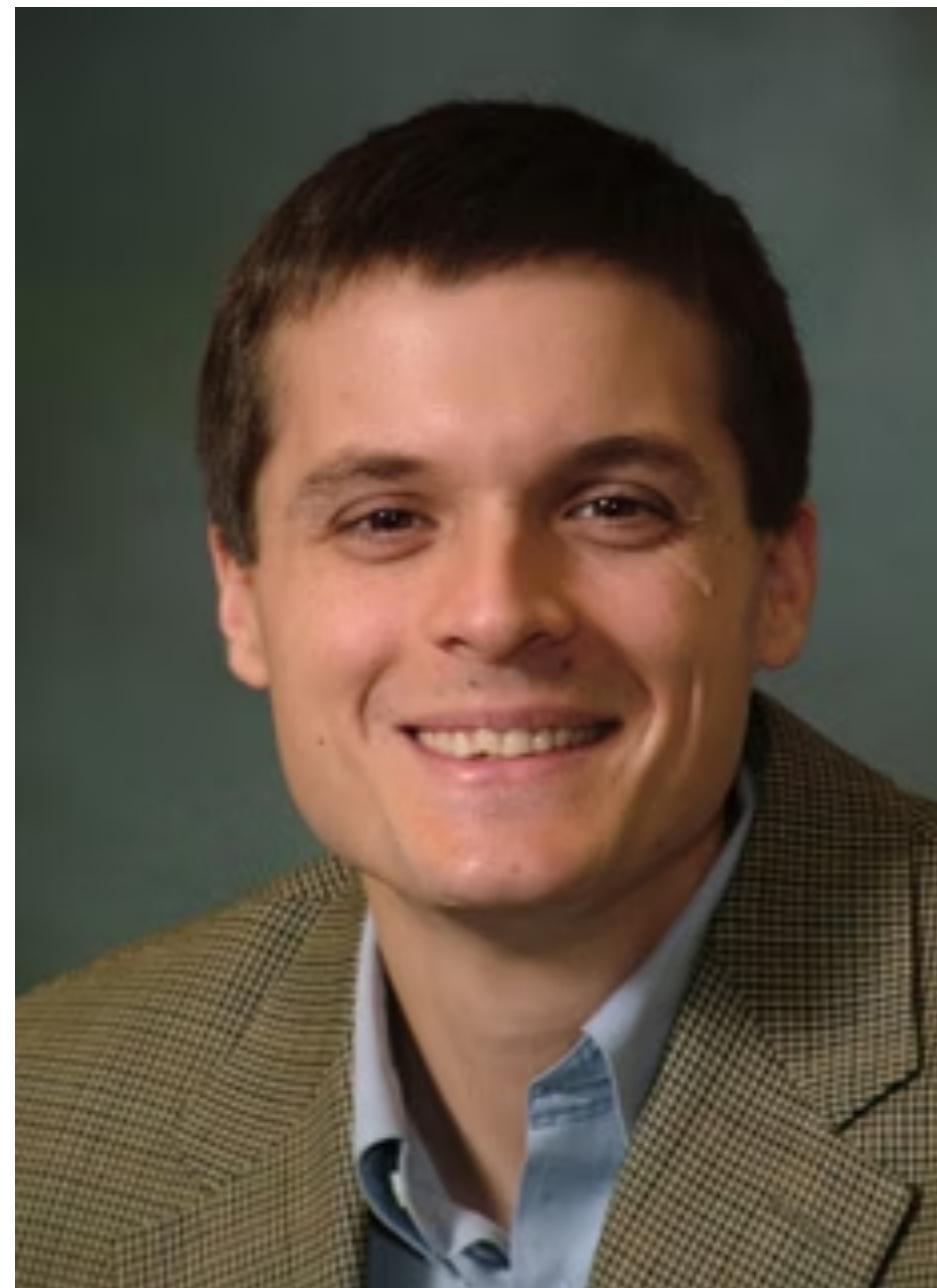
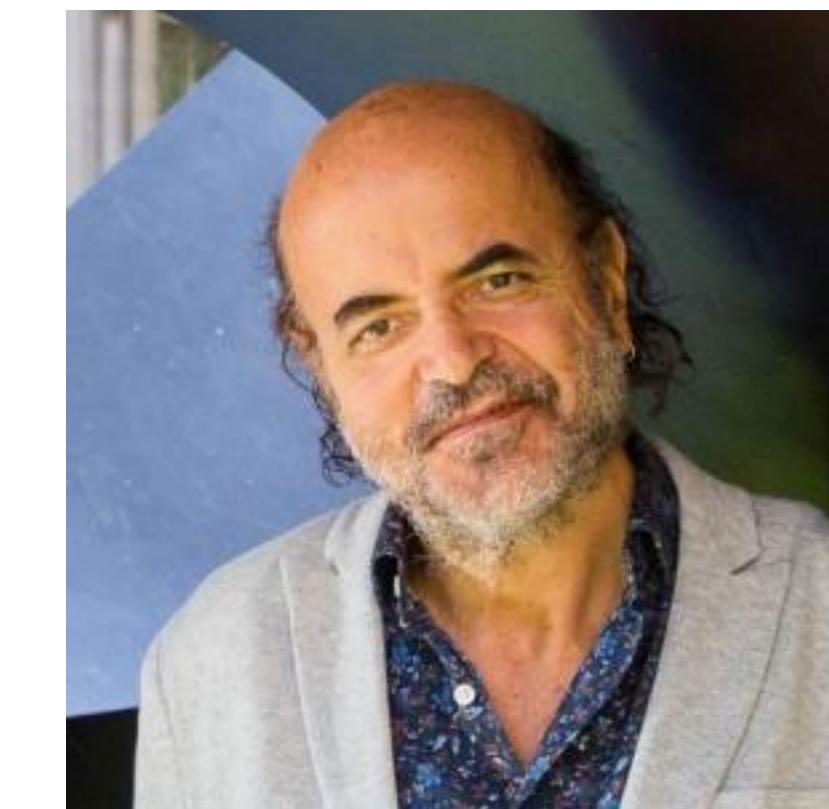


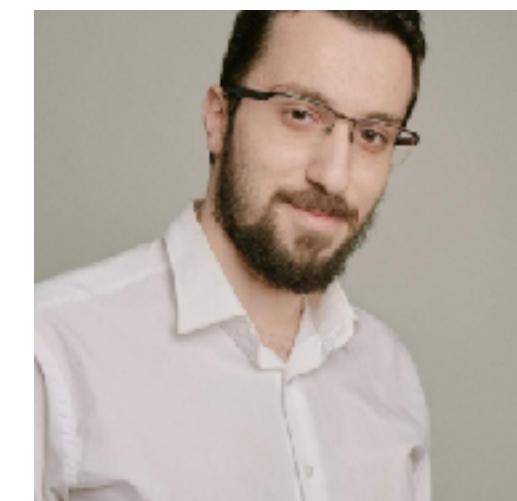
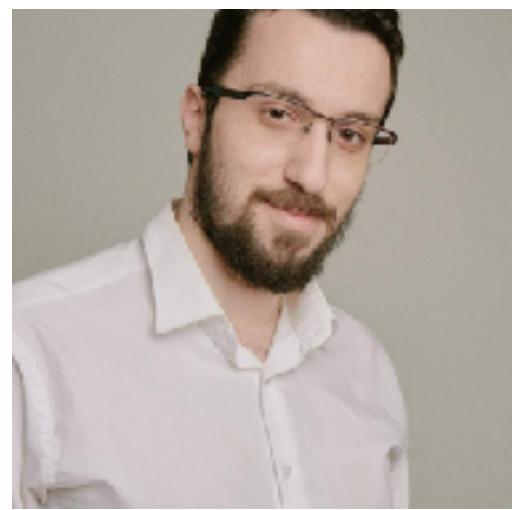
Photo: Manolis Vlatakis-Gkaragkounis

# Committee members

**Joan Bruna, Daniel Hsu, Christos Papadimitriou, Rocco Servedio, Matus Telgarsky**



# Research collaborators



# Columbia CS Theory



# Teachers, Professors, Mentors



COLUMBIA  
UNIVERSITY



BROWN

# Friends



# Shuai



# Family



# Dr. Barbara Hugus (1927 - 2021)



# Mom and Barbara



**Thank you.**