Machine Learning

Reinforcement Learning & Transformer

Dr. Shuang LIANG

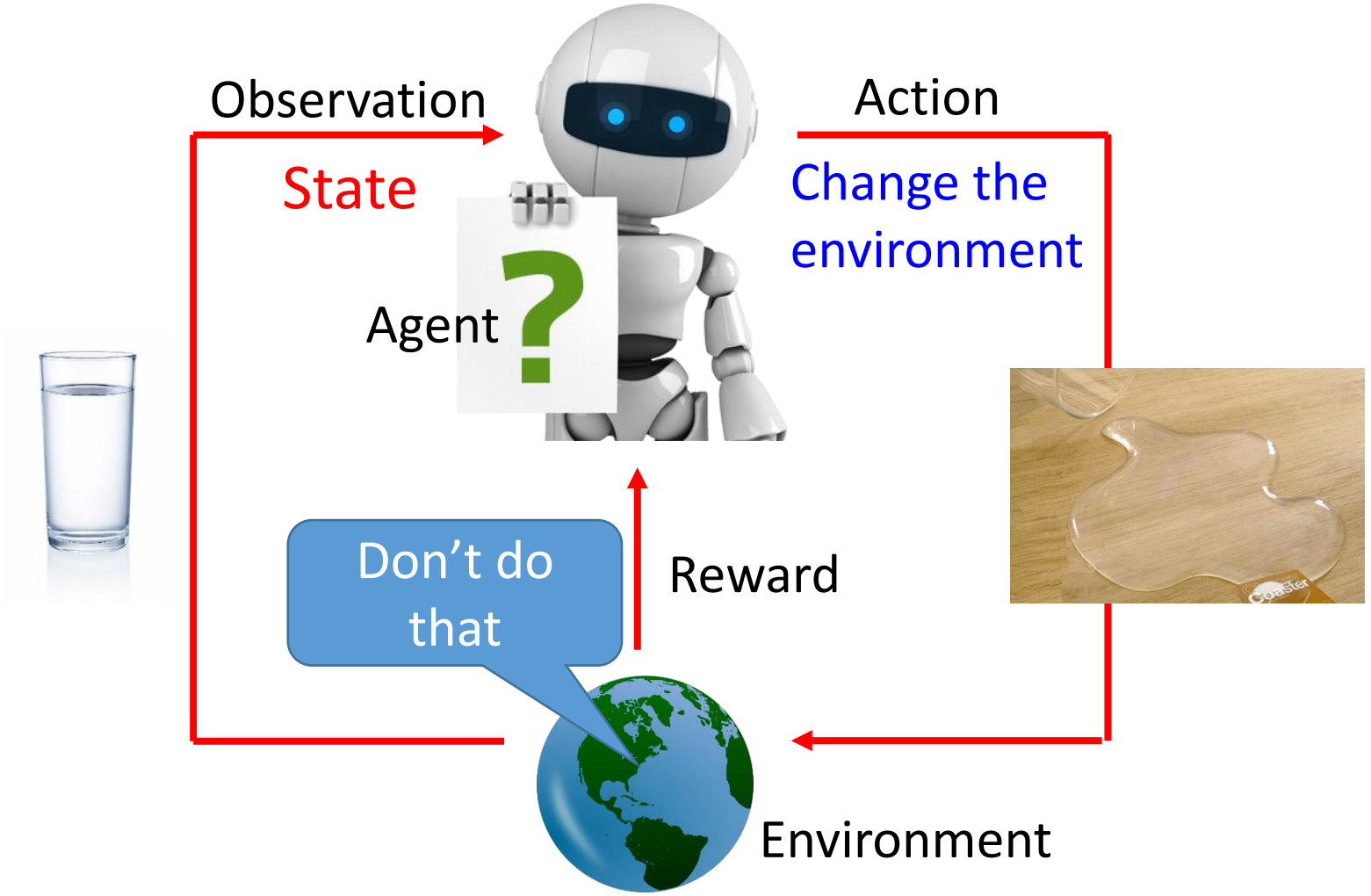
Today's Topics

- Reinforcement Learning
- Transformer

Today's Topics

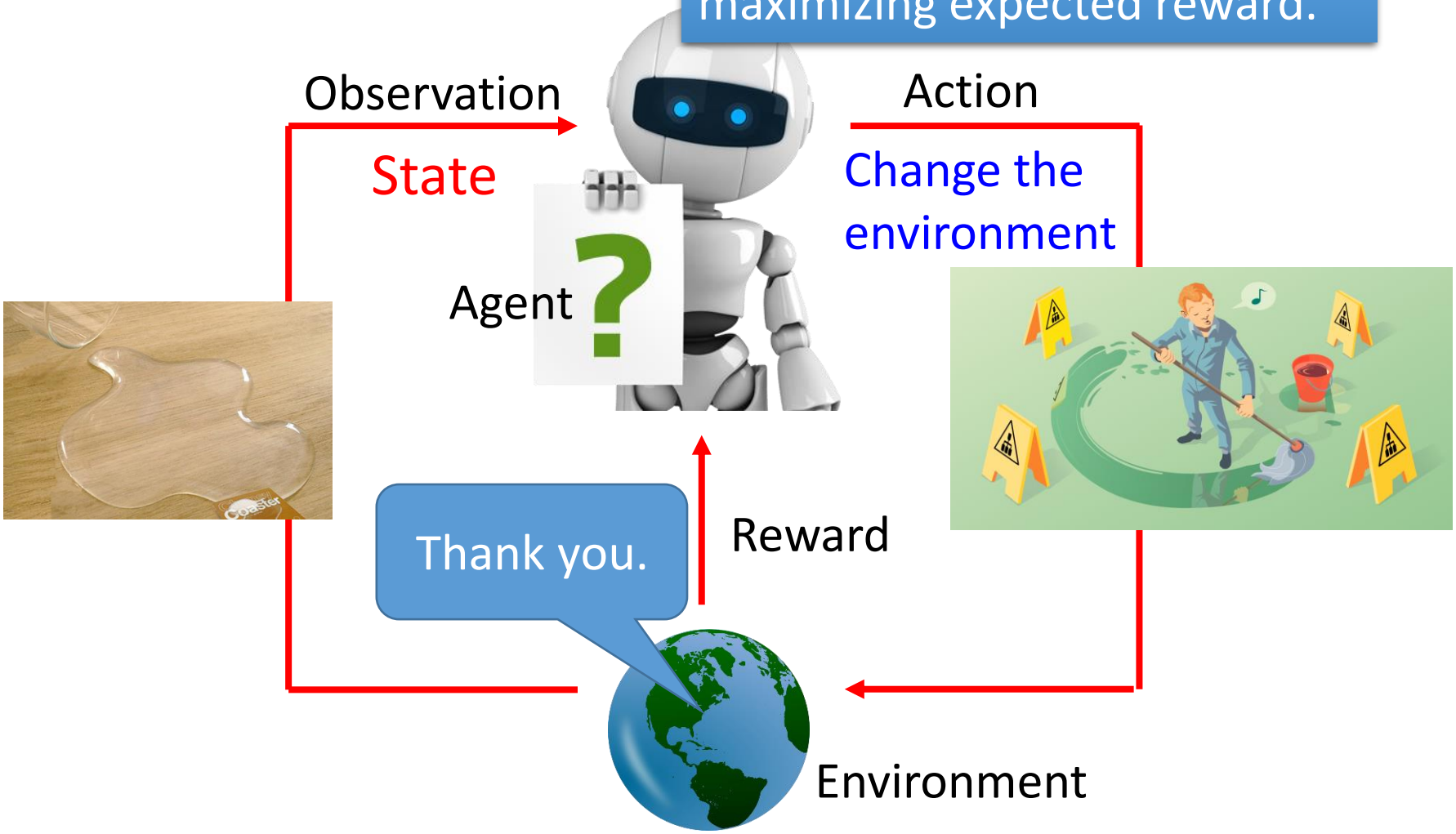
- ***Reinforcement Learning***
- Transformer

Scenario of Reinforcement Learning

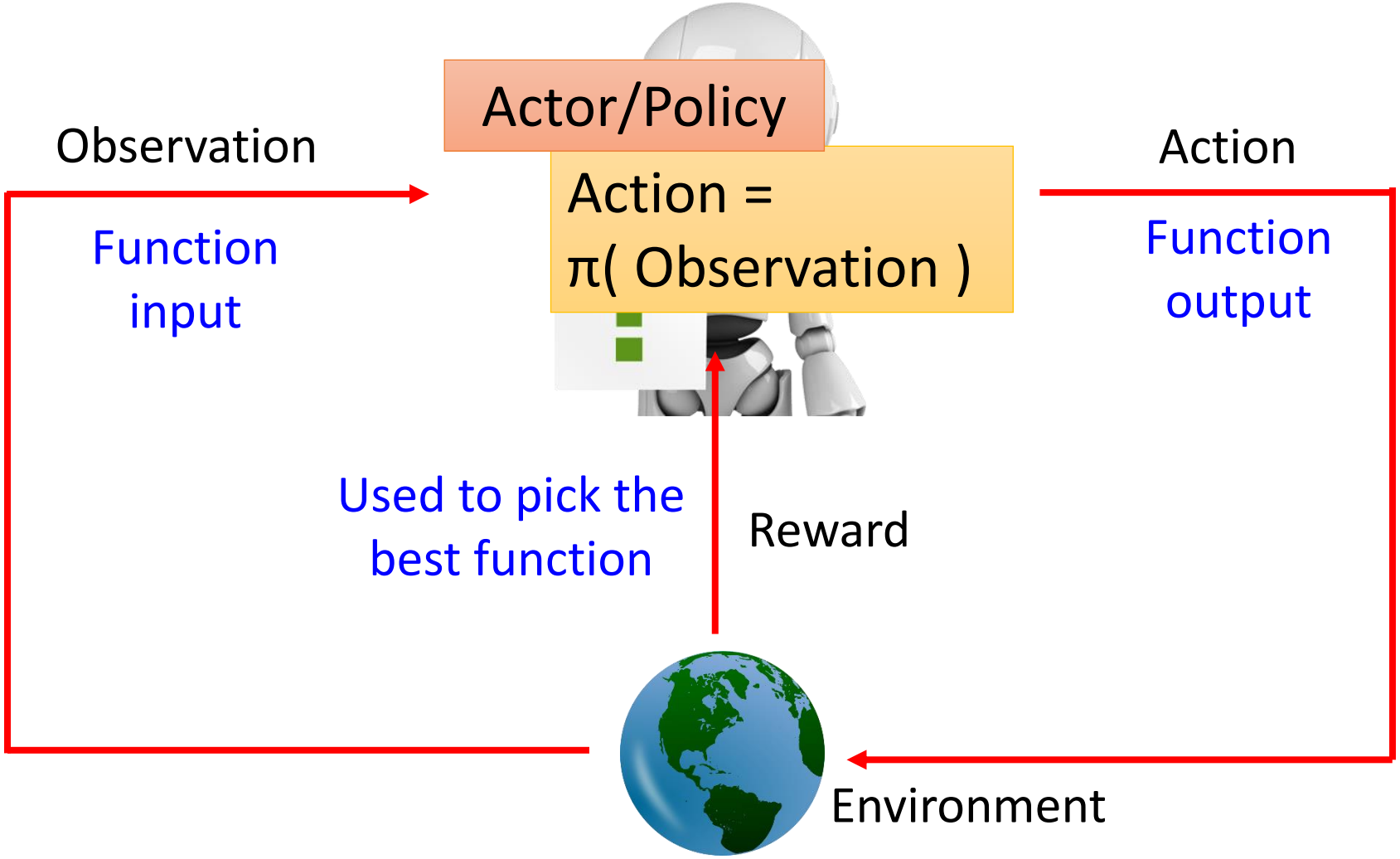


Scenario of Reinforcement Learning

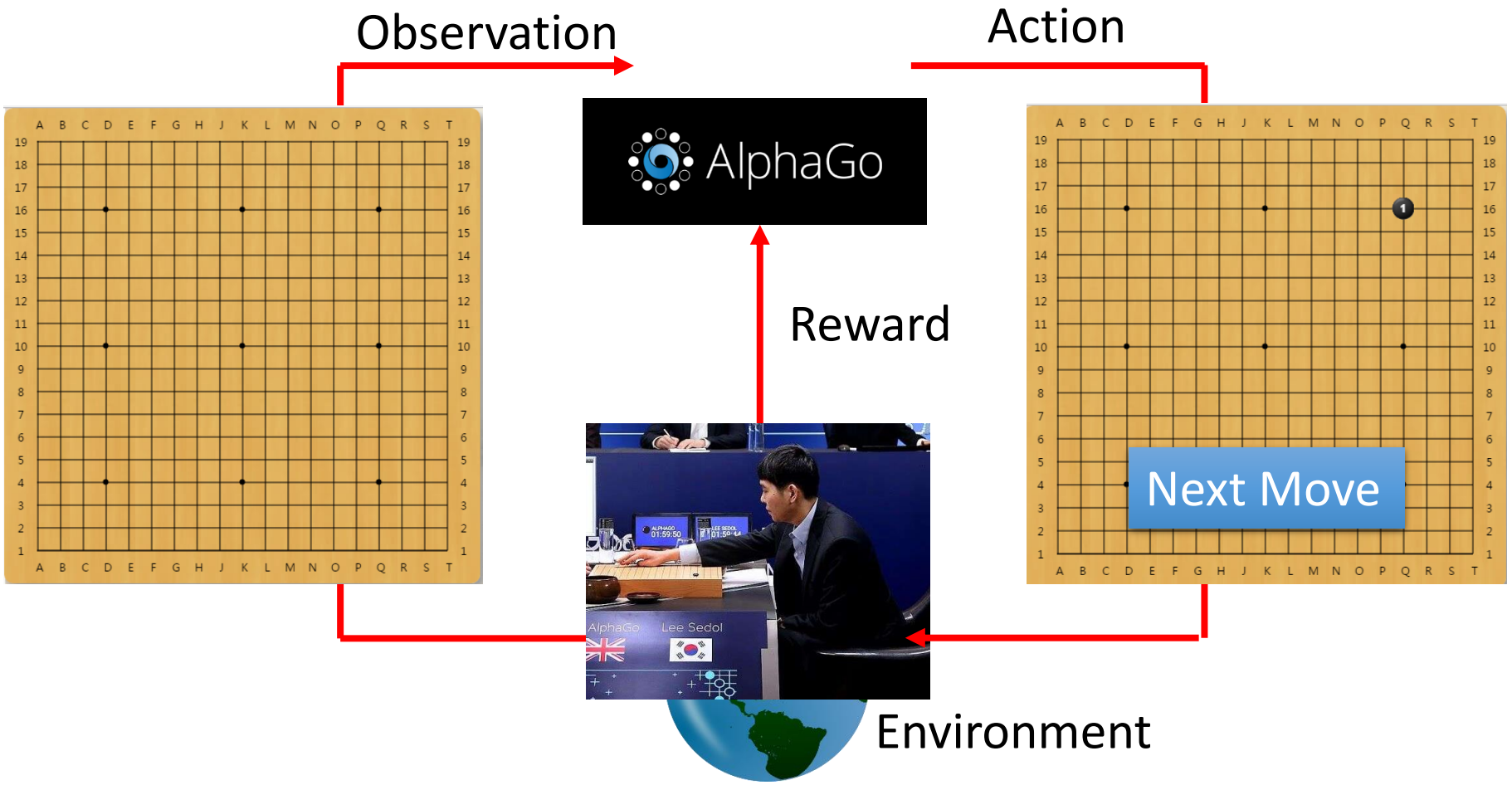
Agent learns to take actions maximizing expected reward.



Machine Learning
≈ Looking for a Function

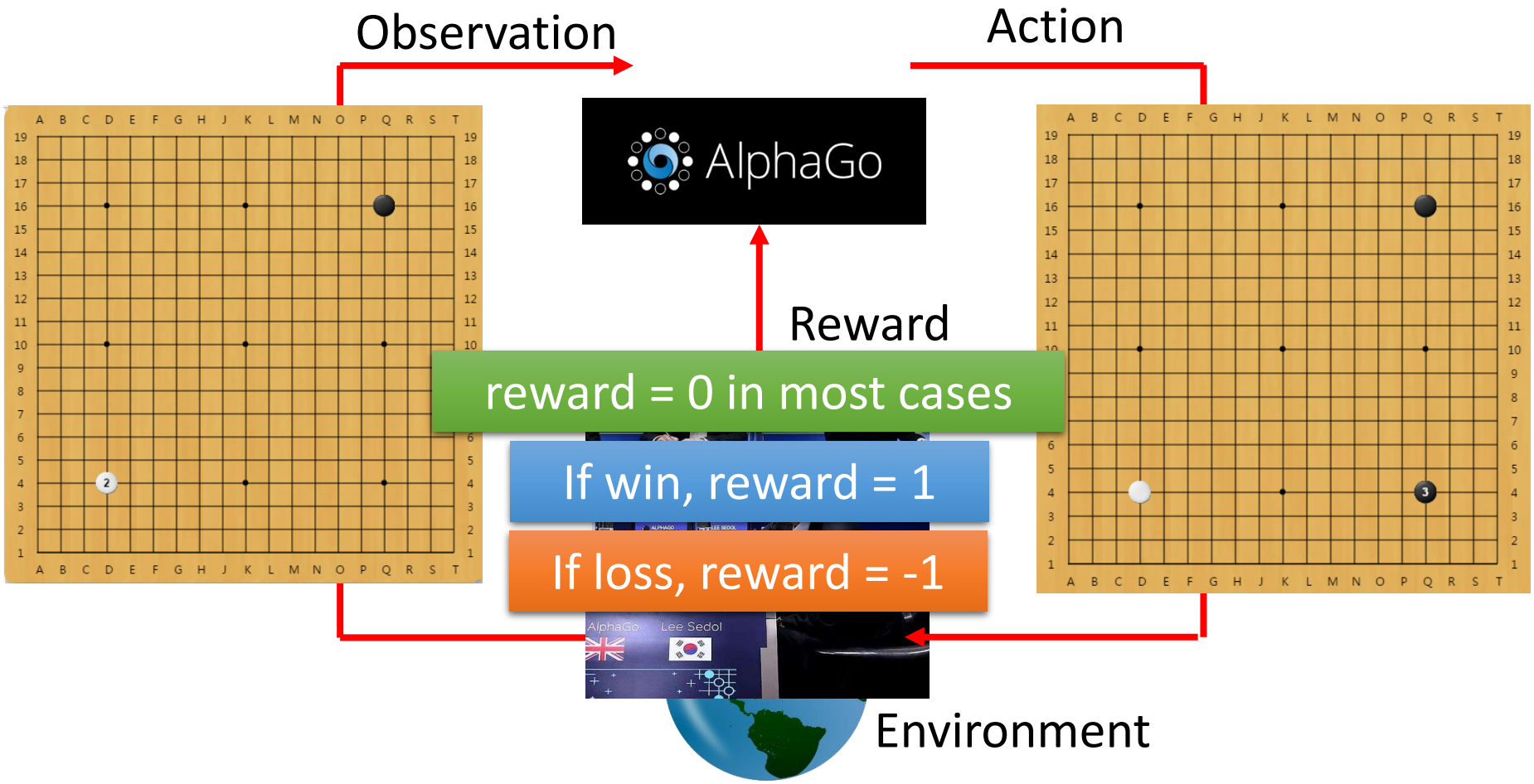


Learning to play Go



Learning to play Go

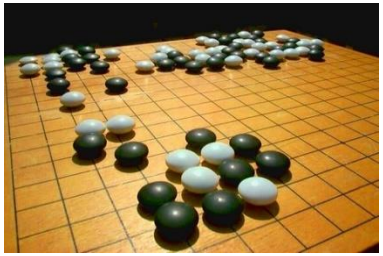
Agent learns to take actions maximizing expected reward.



Learning to play Go

- Supervised: Learning from teacher

Next move: "5-5"



Next move: "3-3"

- Reinforcement Learning Learning from experience

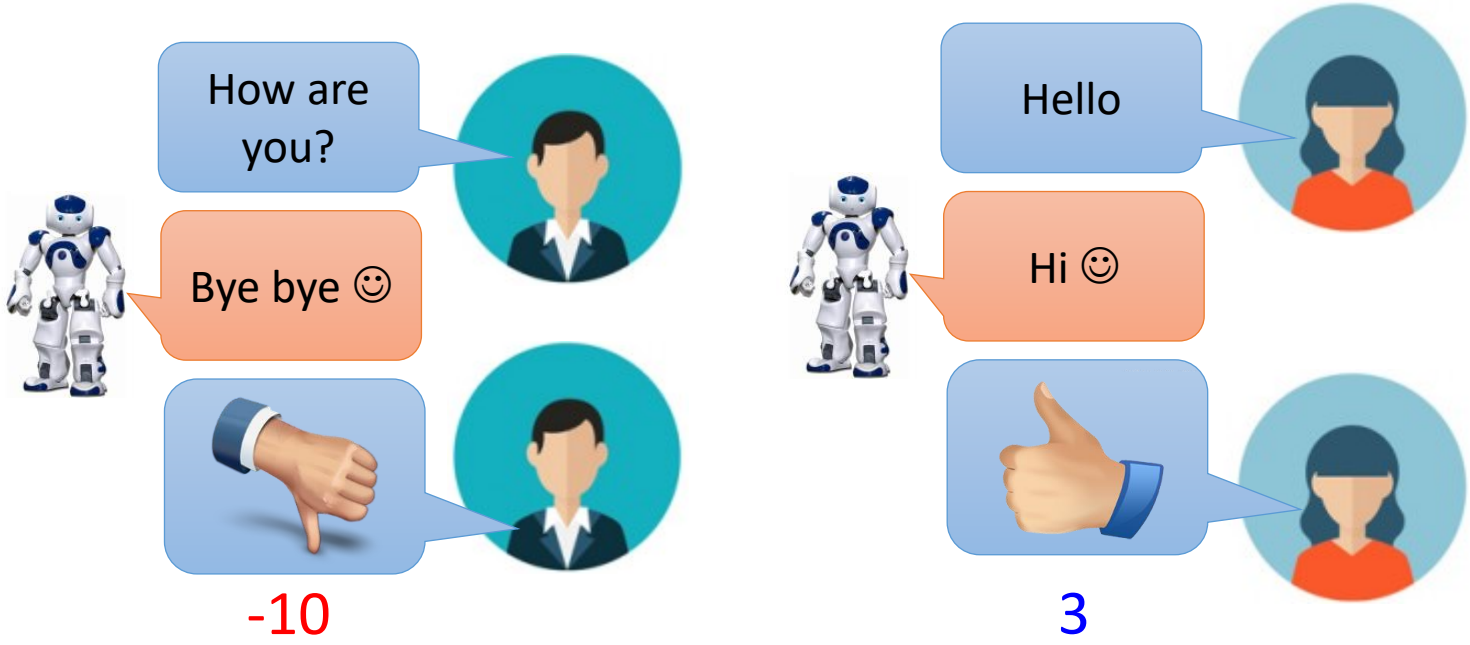
First move → …… many moves …… → Win!

(Two agents play with each other.)

Alpha Go is supervised learning + reinforcement learning.

Learning a chat-bot

- Machine obtains feedback from user



- Chat-bot learns to maximize the ***expected reward***

Learning a chat-bot

- Let two agents talk to each other (sometimes generate good dialogue, sometimes bad)


How old are you?


See you.


See you.


See you.


How old are you?


I am 16.


I thought you were 12.


What make you think so?

Learning a chat-bot

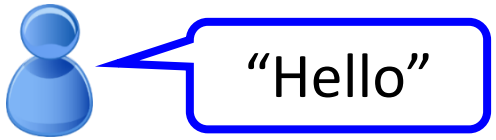
- By this approach, we can generate a lot of dialogues.
- Use some pre-defined rules to evaluate the goodness of a dialogue

Machine learns from the evaluation

Deep Reinforcement Learning for Dialogue Generation https://arxiv.org/pdf/1606.01541v3.pdf

Learning a chat-bot

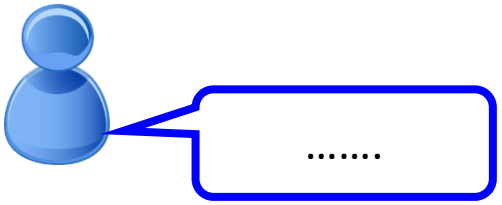
- Supervised

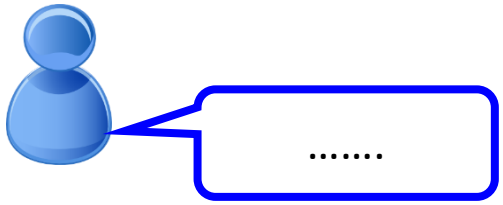


Say “Hi”

Say “Good bye”

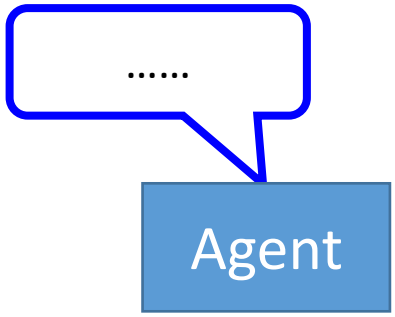
- Reinforcement





.....

Bad



Reinforcement Learning Involves

- Optimization
- Delayed consequences
- Exploration
- Generalization

Optimization

- Goal is to find an optimal way to make decisions
  - Yielding best outcomes or at least very good outcomes
- Explicit notion of utility of decisions
- Example: finding minimum distance route between two cities given network of roads

Delayed consequences

- Decisions now can impact things much later...
  - Saving for retirement
  - Finding a key in video game Montezuma's revenge
- Introduces two challenges
  - When planning: decisions involve reasoning about not just immediate benefit of a decision but also its longer term ramifications
  - When learning: temporal credit assignment is hard (what caused later high or low rewards?)

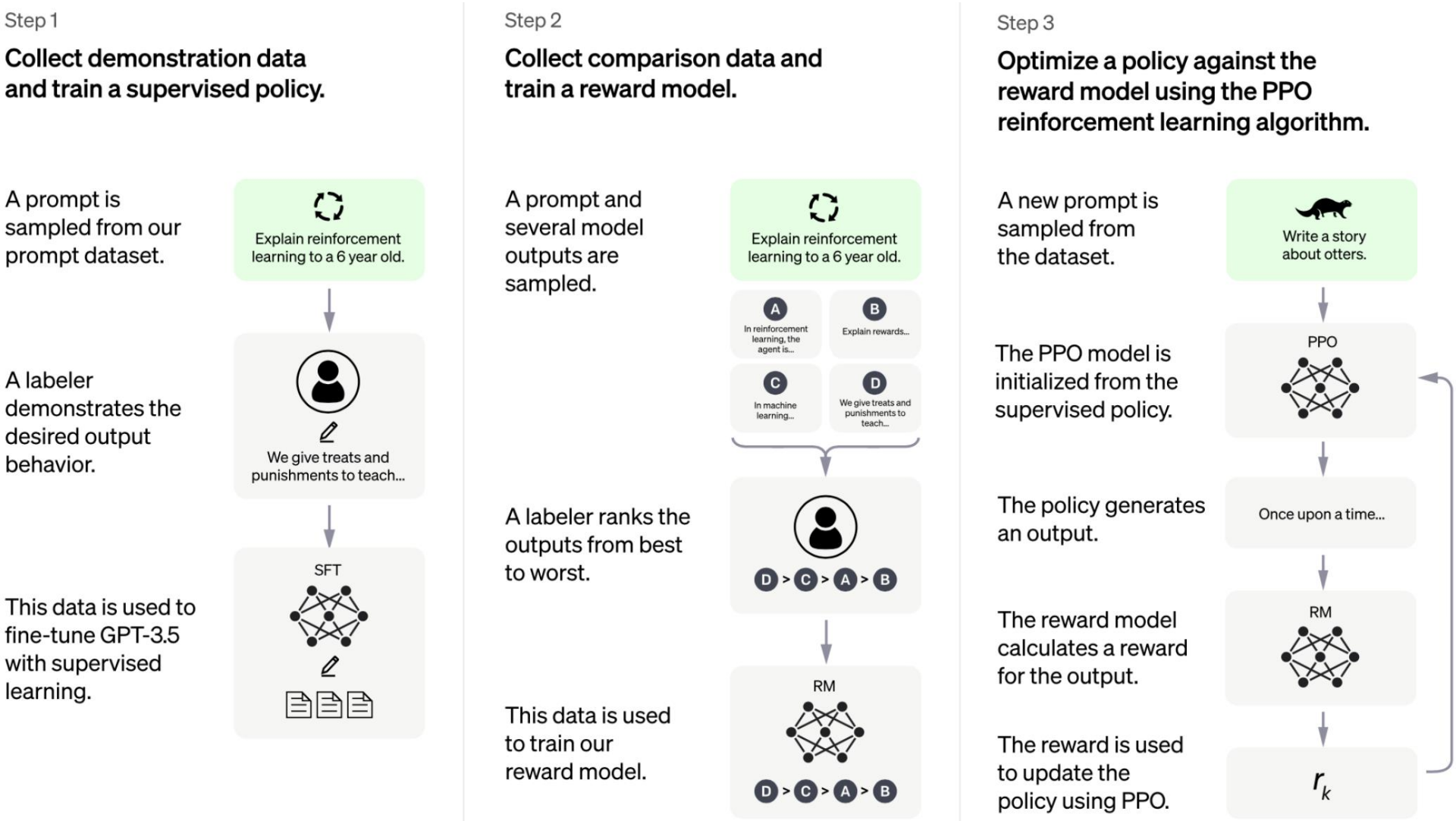
Exploration

- Learning about the world by making decisions
  - Agent as scientist
  - Learn to ride a bike by trying (and failing)
  - Finding a key in Montezuma's revenge
- Censored data（删失数据）
  - Only get a reward (label) for decision made
  - Don't know what would have happened if we had taken red pill instead of blue pill (Matrix movie reference)
- Decisions impact what we learn about
  - If we choose to go to SJTU instead of TongjiU, we will have different later experiences...

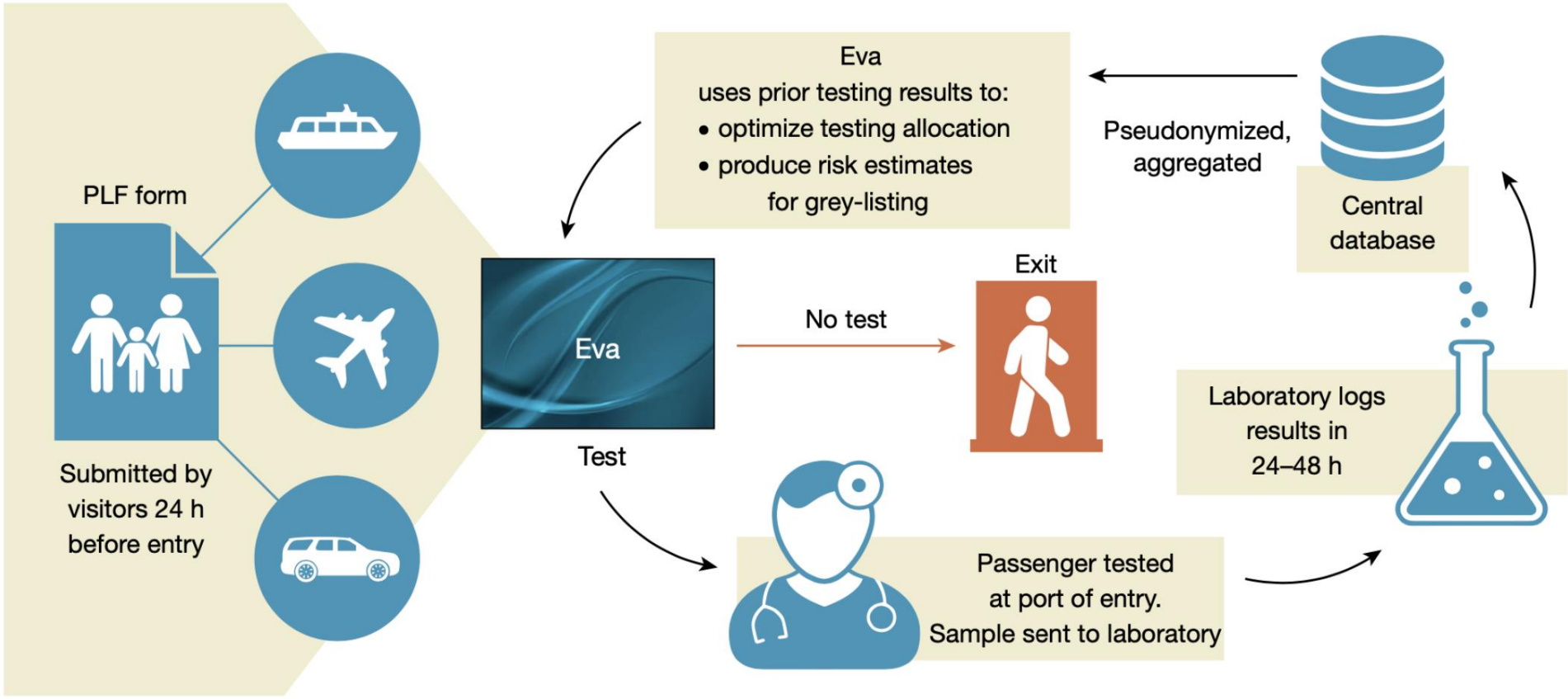
Generalization

- Policy is mapping from past experience to action
- Why not just pre-program a policy?

Example: ChatGPT



Example: COVID-19 border testing



Today's Topics

- Reinforcement Learning
- ***Transformer***

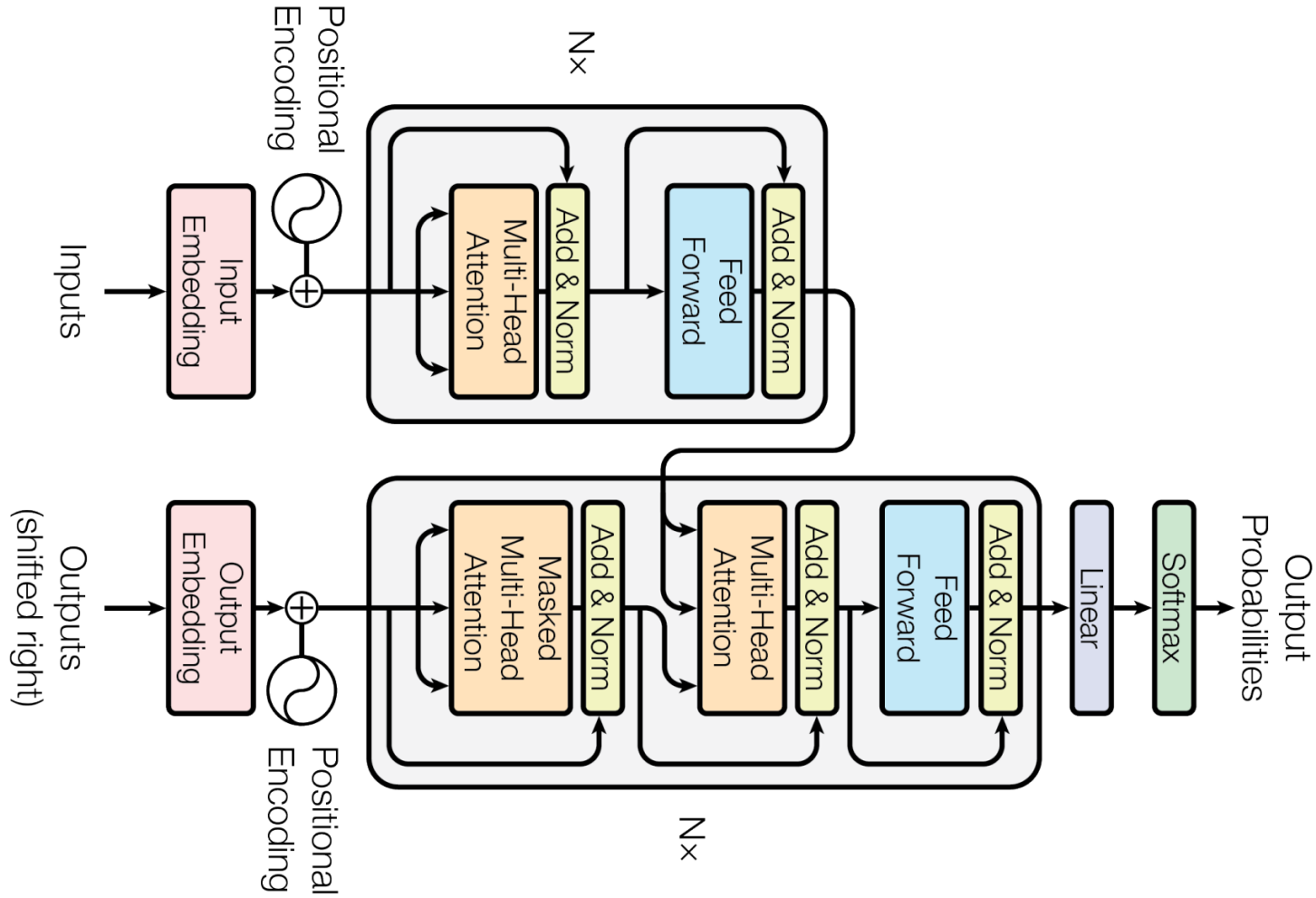
What is a Transformer model?

- A transformer model is a neural network that learns **context** and thus meaning by tracking relationships in sequential data like the words in this sentence.
- Transformer models apply an evolving set of mathematical techniques, called attention or **self-attention**, to detect subtle ways even distant data elements in a series influence and depend on each other.

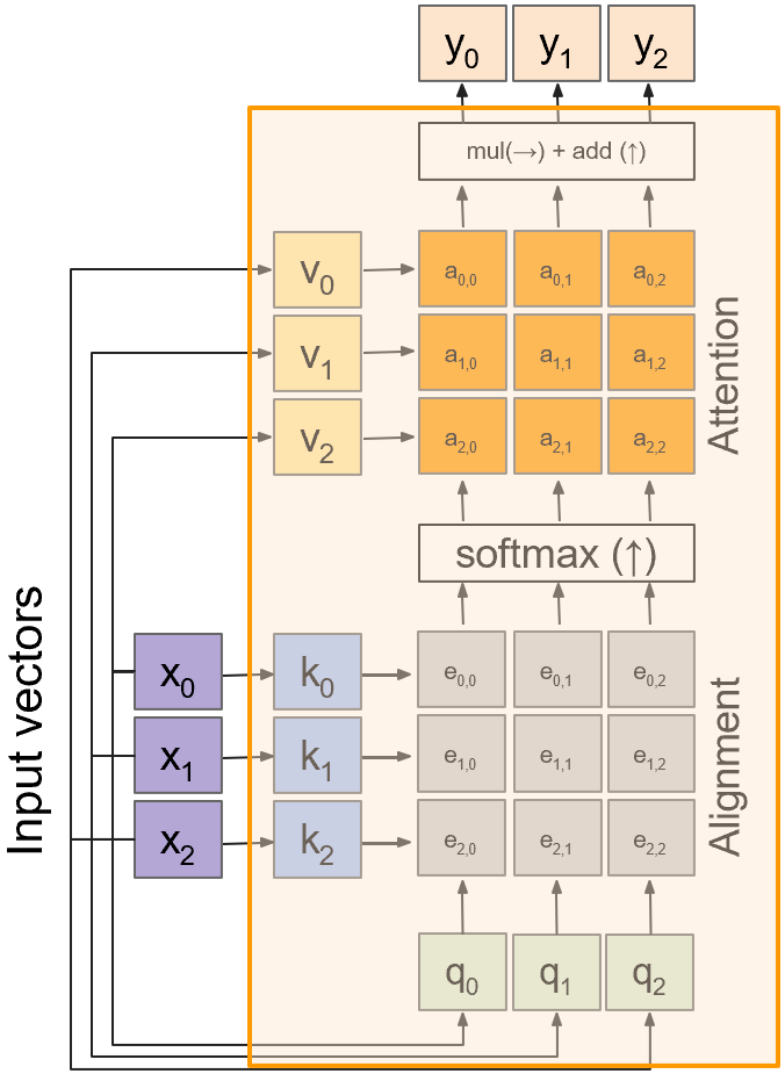
Architecture

[1706.03762.pdf (arxiv.org)]

Core: Multi-Head Attention, Self-attention, Positional Encoding



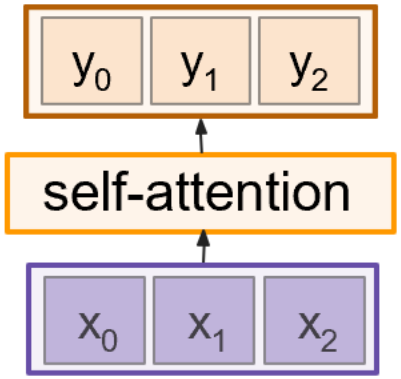
Self attention layer



Outputs:
context vectors: **y** (shape: D_v)

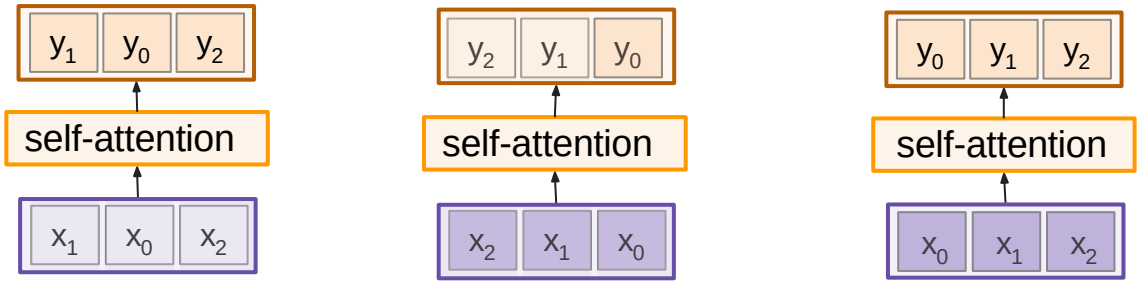
Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Query vectors: $\mathbf{q} = \mathbf{x}W_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Inputs:
Input vectors: **x** (shape: N x D)

Self attention layer

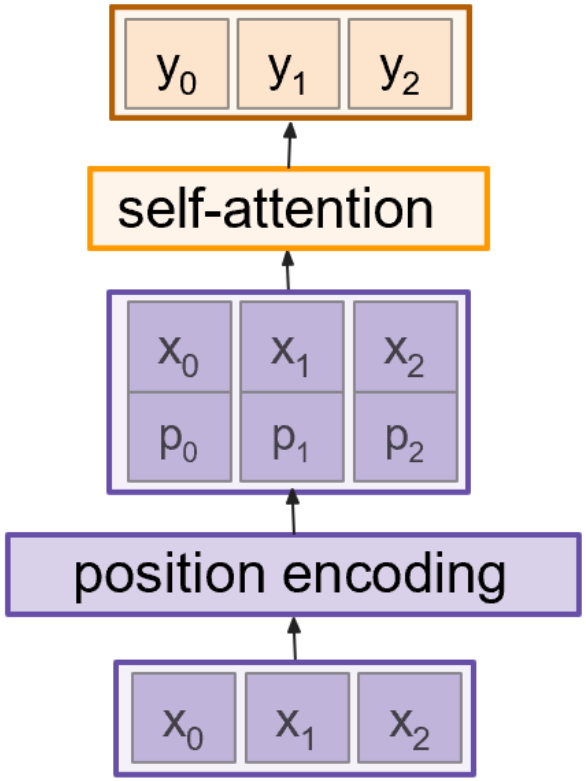


Permutation equivariant

Self-attention layer **doesn't care about the orders of the inputs**!

Problem: How can we encode ordered sequences like language or spatially ordered image features?

Positional encoding

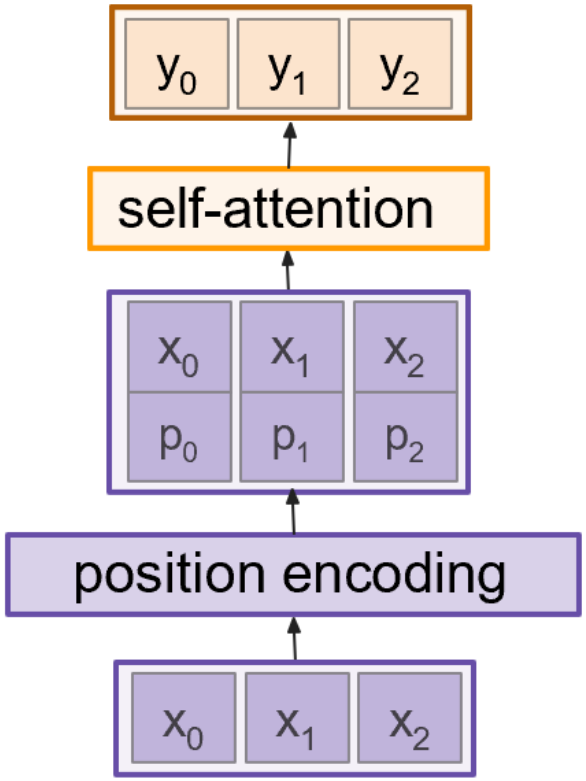


Concatenate special positional encoding p_j to each input vector x_j

We use a function *pos*: N $\rightarrow R^d$ to process the position j of the vector into a d-dimensional vector

So, $p_j = pos(j)$

Positional encoding



Desiderata of *pos*(.) :

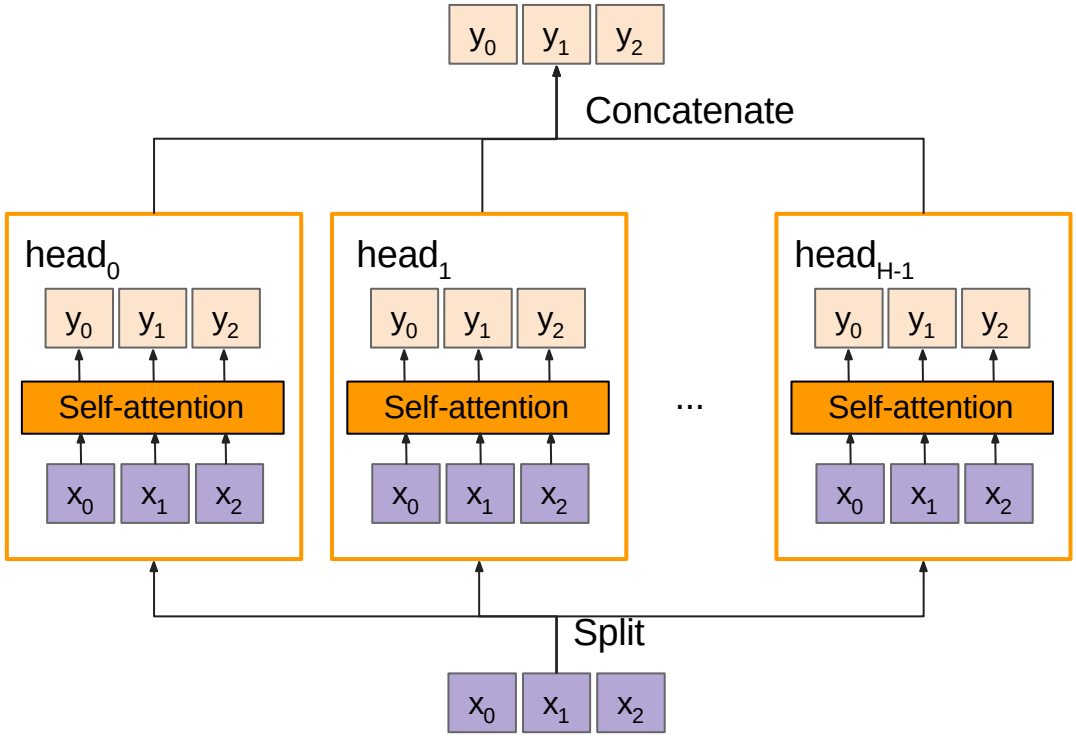
- It should output a **unique** encoding for each time-step (word's position in a sentence)
- **Distance** between any two time-steps should be consistent across sentences with different lengths.
- Our model should generalize to **longer** sentences without any efforts. Its values should be bounded.
- It must be **deterministic**.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

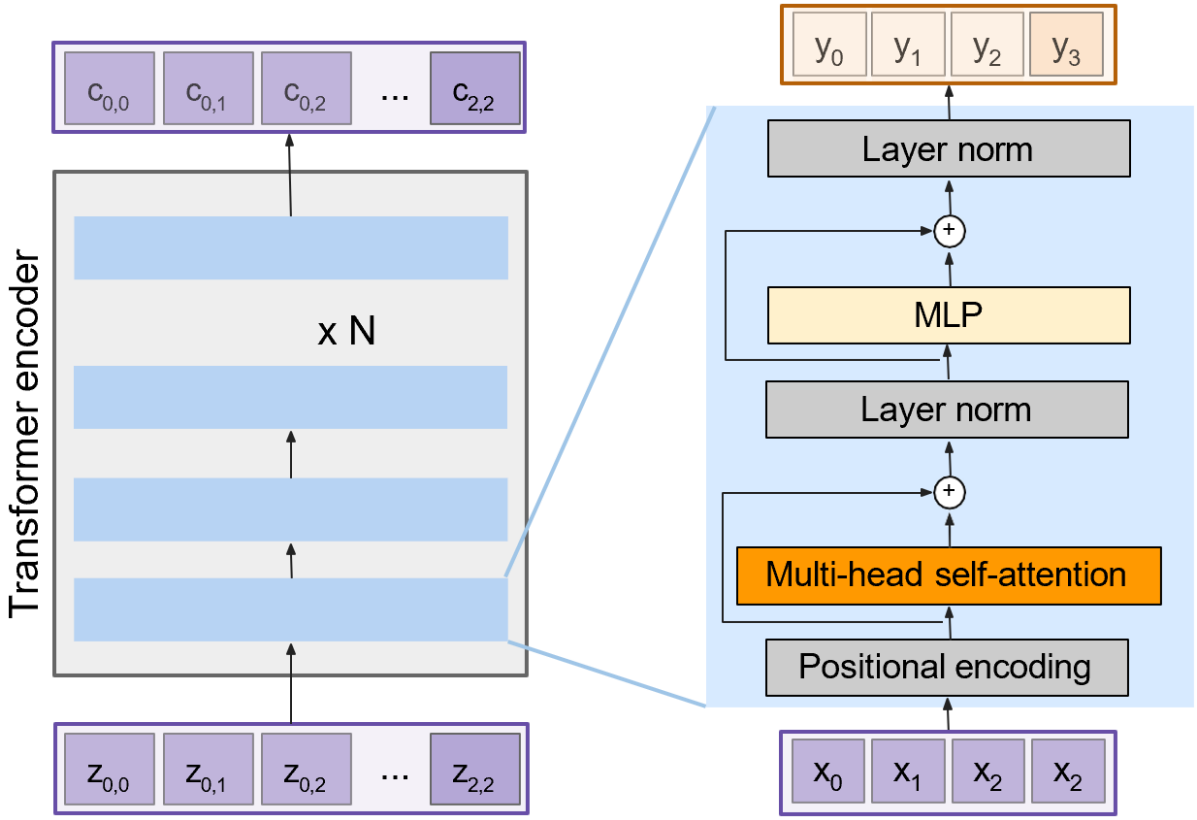
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Multi-head self-attention layer

- Multiple self-attention heads in parallel



The Transformer encoder block



Transformer Encoder Block:

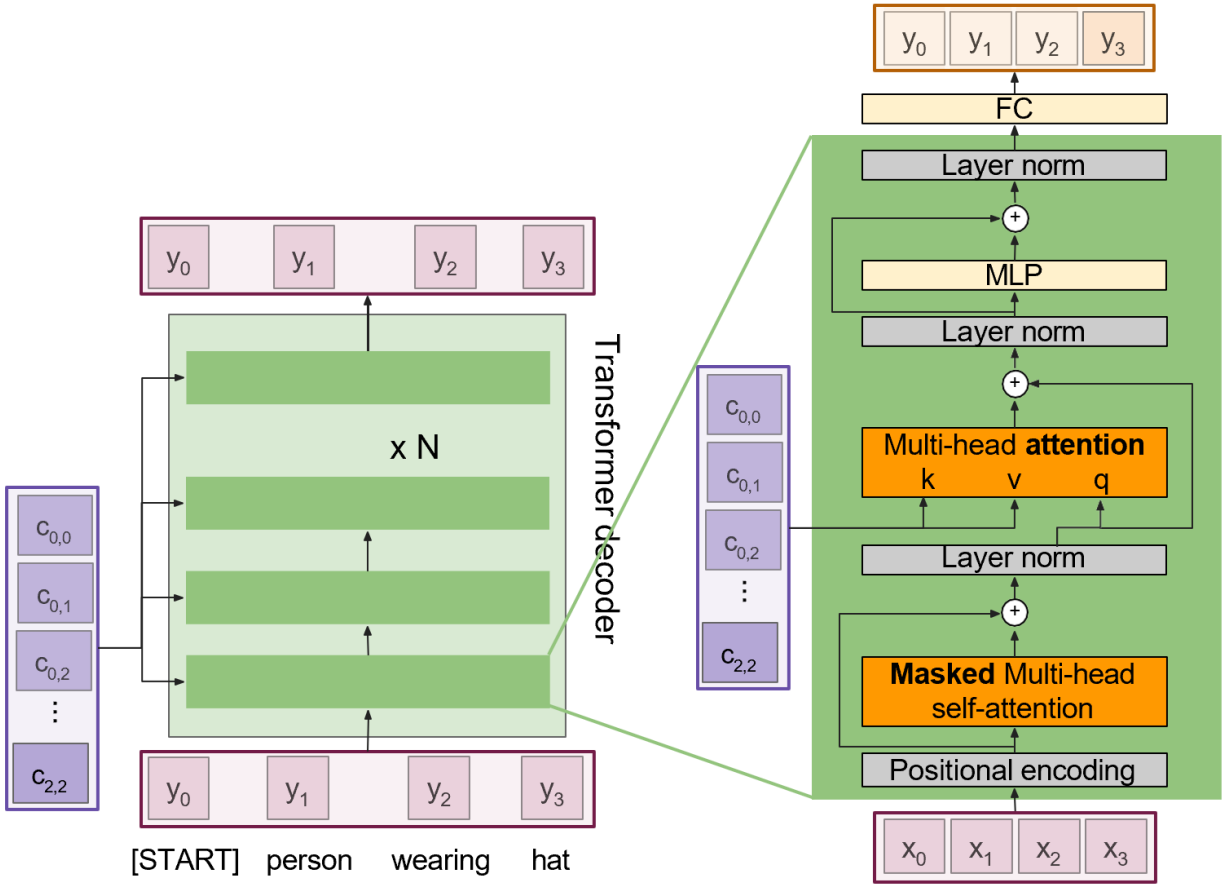
Inputs: Set of vectors **x**
Outputs: Set of vectors **y**

Self-attention is the only interaction between vectors.

Layer norm and MLP operate independently per vector.

Highly scalable, highly parallelizable, but high memory usage.

The Transformer decoder block



Transformer Decoder Block:

Inputs: Set of vectors **x** and Set of context vectors **c.**
Outputs: Set of vectors **y.**

Masked Self-attention only interacts with past inputs.

Multi-head attention block is NOT self-attention. It attends over encoder outputs.

Highly scalable, highly parallelizable, but high memory usage.

Vision Transformers vs. ResNets

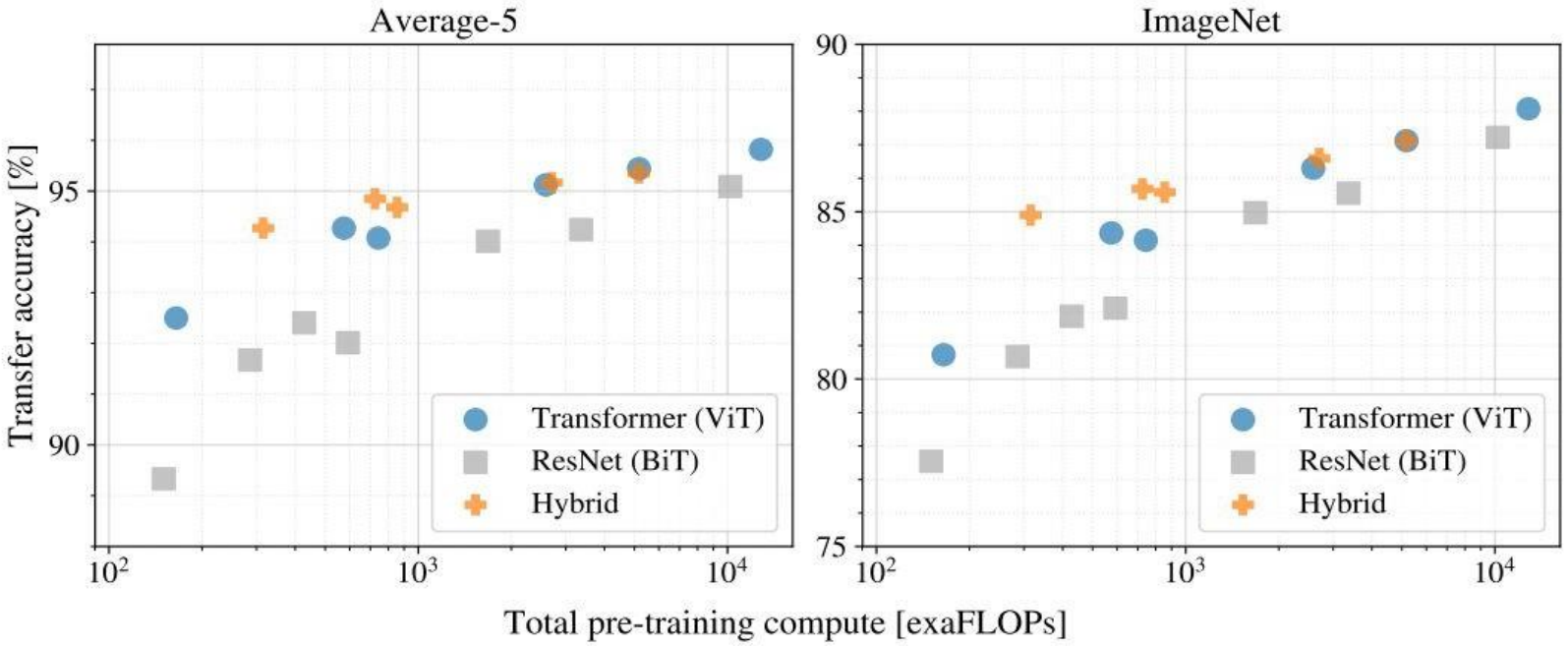
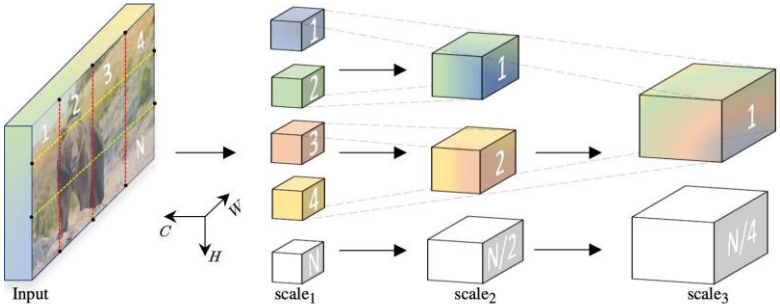


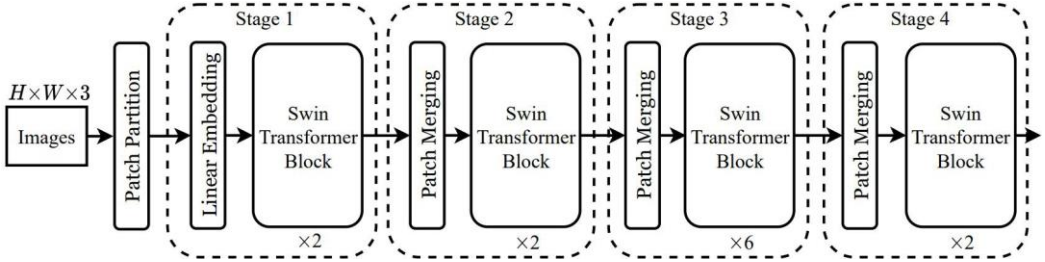
Figure 5: Performance versus cost for different architectures: Vision Transformers, ResNets, and hybrids. Vision Transformers generally outperform ResNets with the same computational budget. Hybrids improve upon pure Transformers for smaller model sizes, but the gap vanishes for larger models.

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

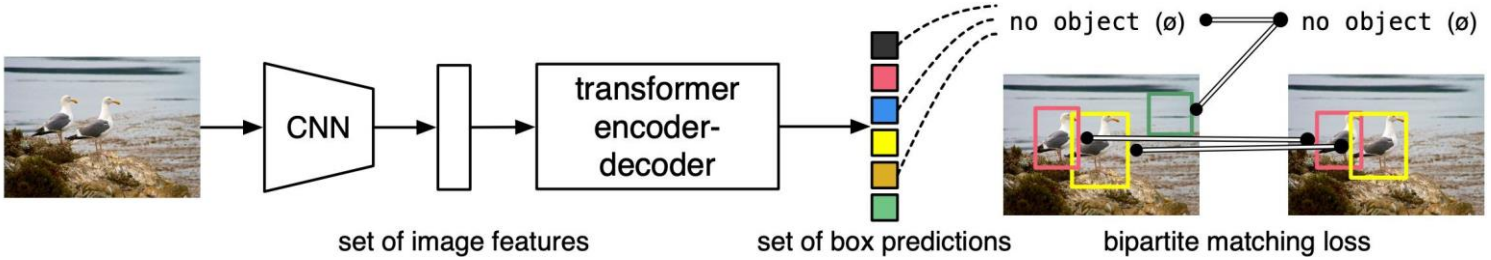
Vision Transformers



Fan et al, “Multiscale Vision Transformers”, ICCV 2021



Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021



Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

Transformer Summary

- **Transformers** are a type of layer that uses **self-attention** and layer norm.
  - It is highly **scalable** and highly **parallelizable**
  - **Faster** training, **larger** models, **better** performance across vision and language tasks
  - They are quickly replacing RNNs, LSTMs, and may(?) even replace convolutions.