2026 한국원자력학회 춘계학술발표회 원자력 인공지능 강습회

(기초) 선형회귀부터 Transformer까지

전준구

Assistant Professor, NINE Lab.,

Division of Advanced Nuclear Engineering,

POSTECH

Numerical
Investigation for
Nature &
Energy Lab.

My Background

[01] J. Jeon et al., *Ann. Nucl. Energy*, 2018.
[02] J. Jeon et al., *Nucl. Eng. Technol.*, 2019.
[03] J. Jeon et al., *Energies*, 2020.
[04] J. Jeon et al., *Int. J. Heat Mass. Transf.*, 2021.
[05] J. Jeon et al., *Nucl. Eng. Technol.*, 2019.
[06] J. Jeon et al., *Nucl. Eng. Technol.*, 2021.
[07] J. Jeon et al., *Int. J. Energy Res.*, 2022.
[08] J. Jeon et al., *Int. J. Heat Mass. Transf.*, 2024
[09] J. Jeon et al., *Phys. Fluids*, 2025.
[10] S. Khanal et al., *Nucl. Eng. Technol.*, 2025.
[11] S. Jo et al., *Eng. Appl. Artif. Intell.*, *2026.*
[12] J. Shin et al., *Prog. Nucl. Energy, 2026.*
[13] S. Baral et al., *Comput. Fluids*., 2026.
[14] B. Cho et al., *Nucl. Eng. Technol*., 2026.
[15] J. Shin et al., *under review*.
[16] W. Jeong et al., *in preparation*.
[17] M. Lee et al., *in preparation*.
[18] S. Yu et al., *in preparation*.

LLM applications

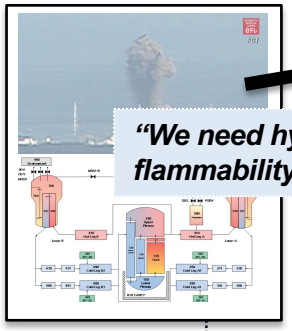
Fine-tuned **la**rge language model for **ME**LCOR **no-co**ding

2016 — **2018** — **2020** — **2022** — **2024** — **2026**

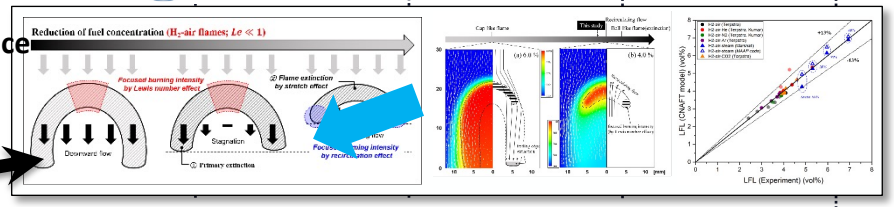
Master degree

Entrance CFD Reference

- ***Severe accident analysis*** *[1,2].*



"We need hydrogen flammability model!"

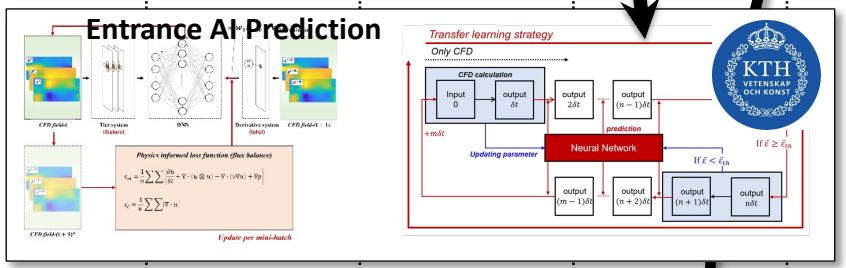


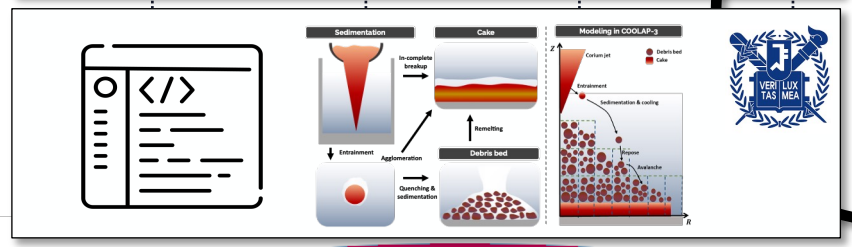
- ***Hydrogen flame extinction mechanism*** *with CFD [3, 4].*
- ***Hydrogen flammability limit model*** *[5, 6].*
- ***Hydrogen combustion risk prediction code*** *(HYCOR) (flammability, flame acceleration, DDT evaluation).*

"We need CFD acceleration using AI !"

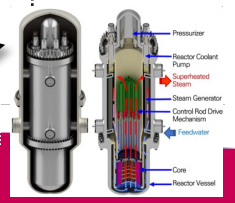
Ph.D. degree

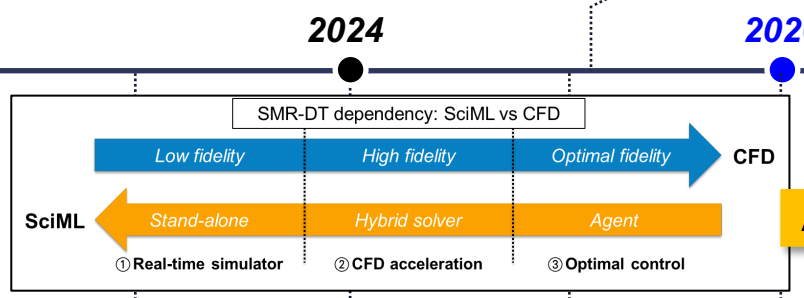
Entrance AI Prediction





"SMR innovation and licensing"





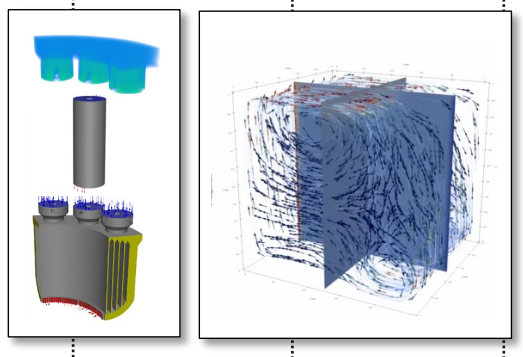
AI Act

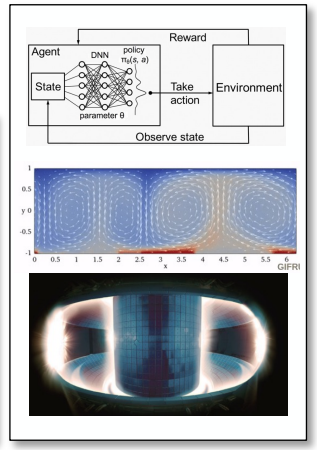
AI-driven SMR-DT

- ***DT level 1: Real-time simulator (AI surrogate) [11, 12, 14-17]***
- ***DT level 2: CFD acceleration (Hybrid solver) [10, 13]***
- ***DT level 3: optimal control strategy (reinforcement learning)***

"We can apply AI to SMR-Digital Twin!"

Postdoc





- ***Finite volume method network (FVMN)*** *model [7].*
- ***Physics-informed transfer learning (RePIT)*** *strategy [8].*
- ***Inductive-biased DRL for flow control*** [9]

- *SMART DT/ RePIT open-source software/ DRL for flow control*

AI-driven advanced severe accident code

- *Development of in-house code for severe accident phenomena*
- *Coupling of MELCOR code with in-house code*
- ***Development framework for severe accident in-house code***

- *SMR regulatory technology*
- ***Hydrogen risk assessment for i-SMR [18]***
- *i-SMR MELCOR modeling*

AI is super powerful!

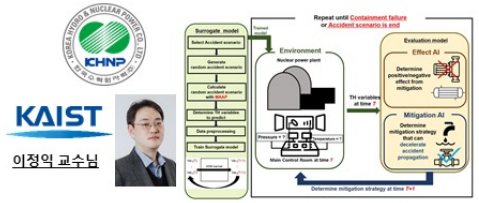
산업/연구/규제에서의 원자력 AI

산업

- **(한국전력기술)** 안전해석코드 입력문 AI 에이전트 개발

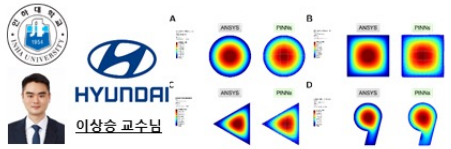
이수영 교수님

- **(한국수력원자력)** 안전해석코드 써로게이트 AI 모델 개발



이정익 교수님

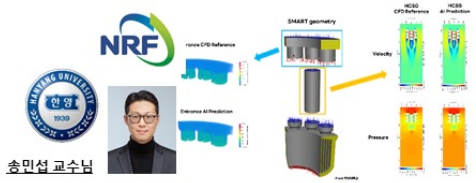
- **(현대자동차)** 구조해석 가속용 PINN 솔버 개발 (완료)



이상승 교수님

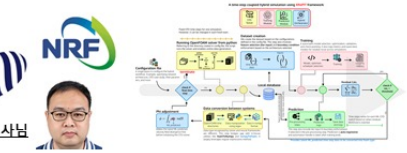
연구

- **(한국연구재단)** SMR 열수력 AI 디지털트윈 기술 개발



송민섭 교수님

- **(한국연구재단)** SMR 고정밀·고속 AI 솔버 개발



박동규 이사님

- (한국연구재단) 원자로 설계가속화 PSA Agentic AI 개발

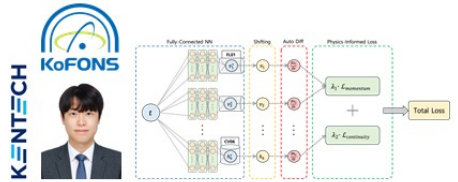
이승준 교수님

- **(한국연구재단)** 핵융합 MHD 유동제어 기술 개발

서재민 교수님

규제

- **(원자력안전재단)** PINN 기반 안전해석솔버 개발



이윤희 교수님

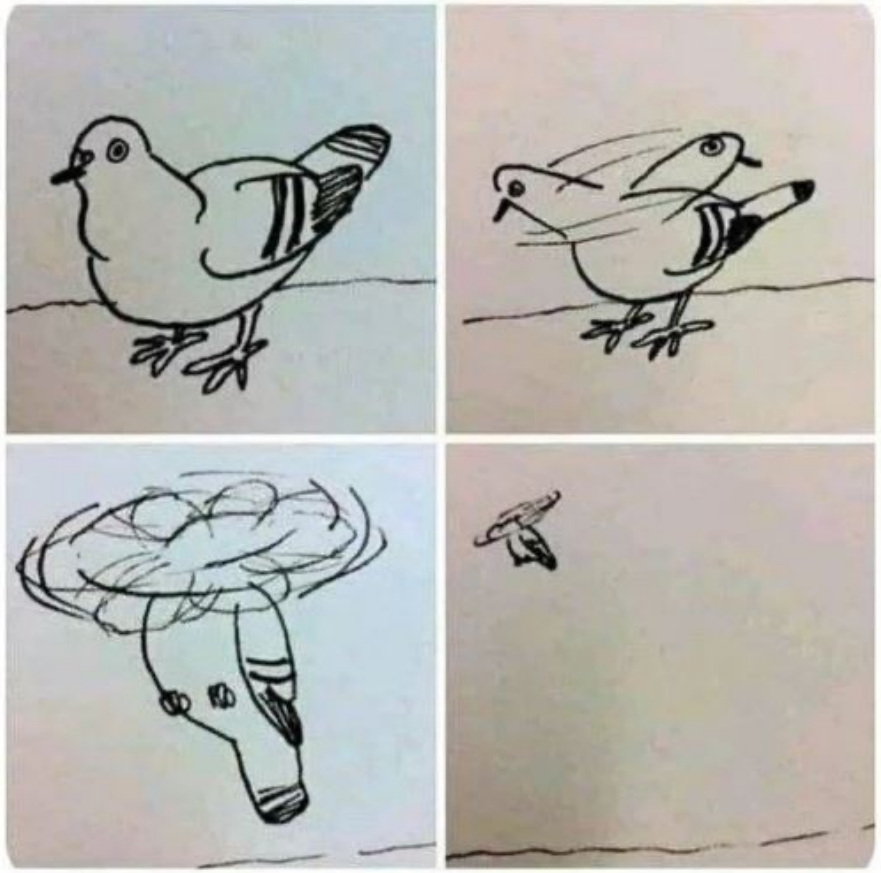
- **(원자력안전재단)** 안전해석코드 AI 요소 규제방법론 개발

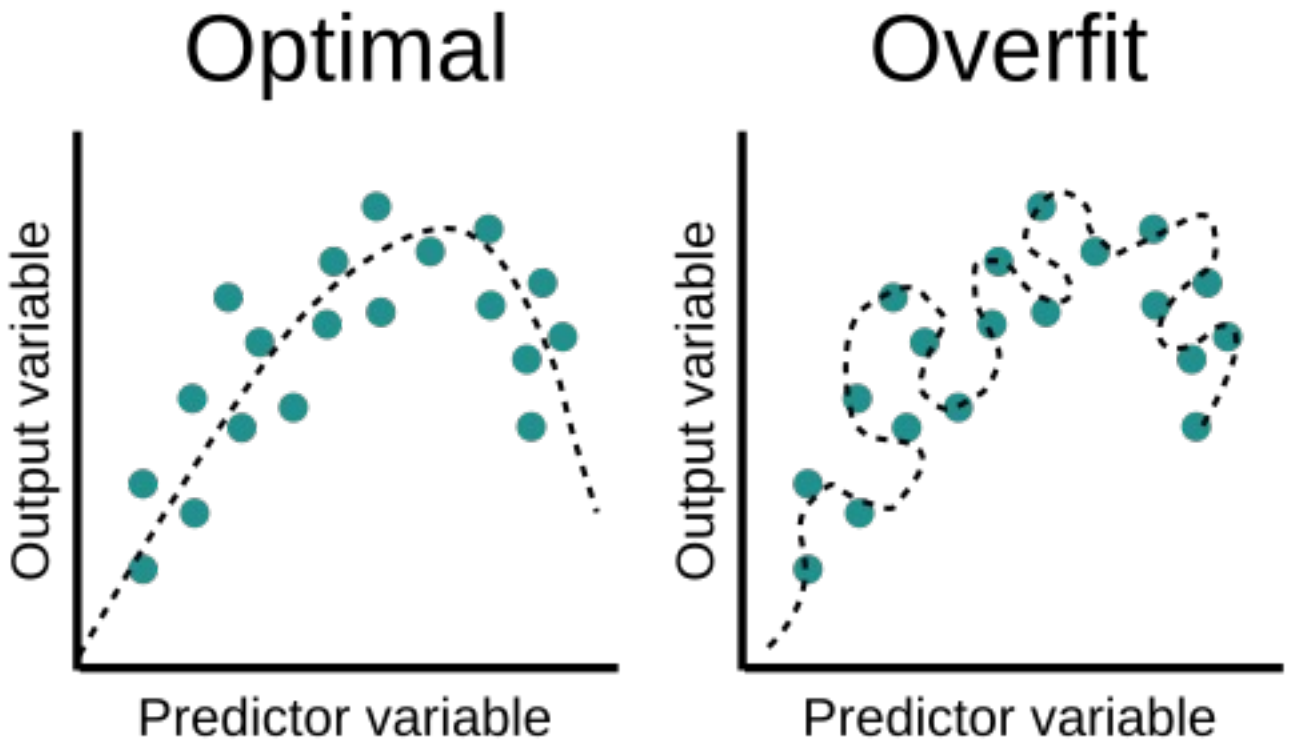
김경모 교수님

- **(소형모듈원자로규제연구추진단)** i-SMR 진공격납용기 가연성 규제방법론 개발

김형대 교수님

But not always…





AI Act (since 2026)

'세계 최초 시행' AI 기본법 주요 내용

2026년 1월 22일 시행

산업 기반 조성

- **인공지능 기본계획 수립**: 과학기술정보통신부 장관이 3년마다 수립
- 국가인공지능위원회 실효성 확보, AI 안전연구소 설립 등

'고영향 AI' 개념 도입

- **고영향 AI:** 의료·에너지·채용·대출심사 등 국민의 생명이나 권리에 큰 영향을 주는 분야의 AI
- 고영향 AI 이용한 제품·서비스 제공 시 이용자에게 **사전 고지 필요**
- 안전성 관리 의무, 과태료 부과 책임 등 법적 규제 대상

AI 투명성 확보

- **AI 워터마크 표시 의무:** 생성형 AI 활용한 결과물에 이용자가 AI 제작물임을 알 수 있도록 가시적·비가시적 워터마크 부착 등 표시

해외 빅테크 국내 대리인 지정

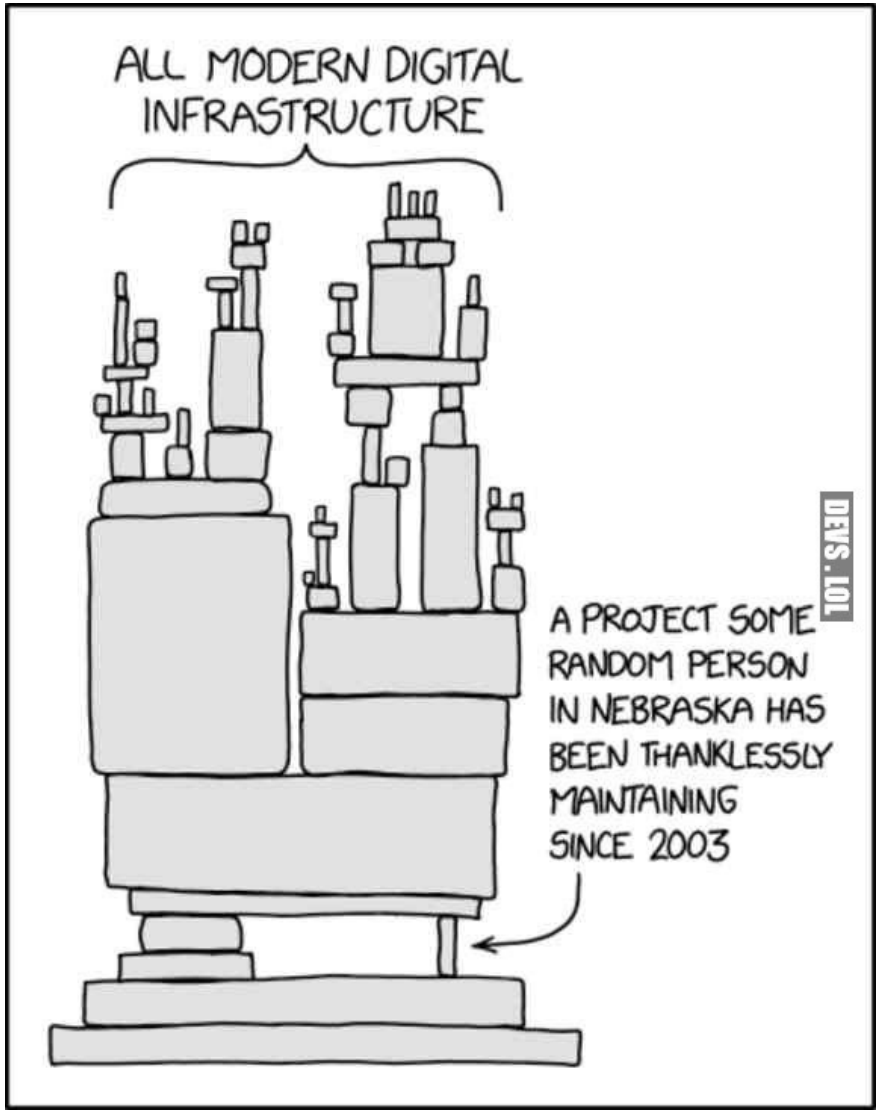
- 전년도 글로벌 매출액 1조 원 이상, 국내 AI 서비스 매출 100억 원 이상, 국내 일평균 이용자 100만 명 이상 조건 중 **하나라도 해당 시 국내 대리인 지정**
- 현재 **오픈AI·구글** 해당

과태료 부과

- AI 기본법상 의무 위반 시 **최대 3,000만 원** 과태료 부과, 1년 이상 계도 기간 운영

연합뉴스 자료: 국가법령정보센터 김민지 기자 = minfo@yna.co.kr/20260120

That’s the reason why we need to lean AI fundamentals.



Source of lecture materials

- 대한기계학회 기계인공지능연구회강습회 자료
- 서강대학교 김낭중 교수님 2024 강습회 강의자료
- 숙명여자대학교 심주영 교수님 2024 강습회 강의자료
- 인하대학교 고승찬 교수님 2025 강습회 강의자료
- 중앙대학교 이수영 교수님 2025 강습회 강의자료
- 한국전기연구원 김태완 박사님 2025 강습회 강의자료
- Andew Ng et al., CS229 Lecture Notes (Stanford University)
- Illustrations from various columns (noted separately)

Tips for Beginners

Stanford Online ‘CS229’ Andrew Ng

대한기계학회-기계인공지능연구회 ‘AI Bootcamp’

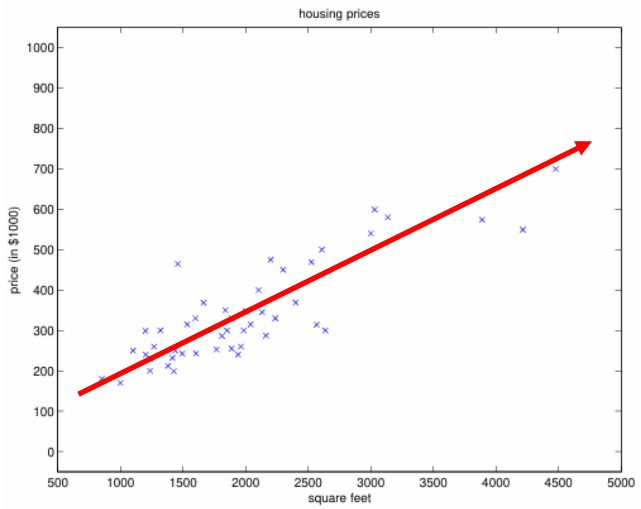
https://sites.google.com/view/aiksme/home?authuser=0

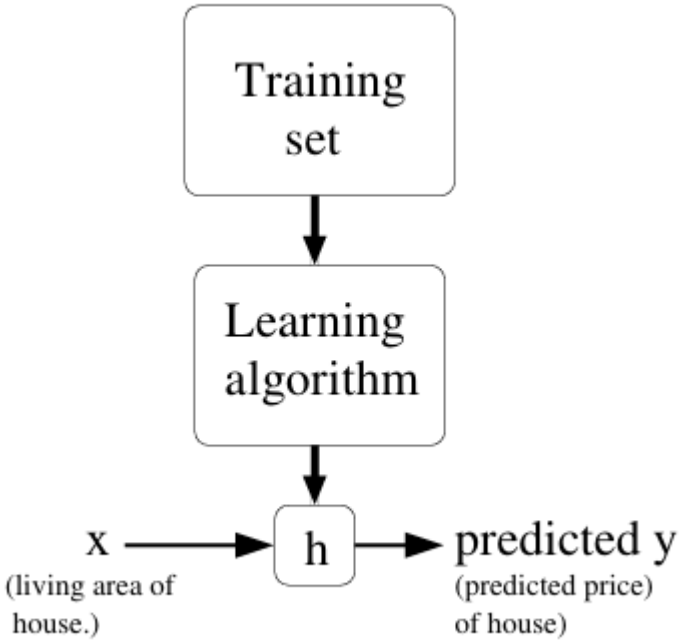
Topics	Colab	Slides	Youtube
Python Programming		pdf#00	
Introduction to Artificial Intelligence (AI)		pdf#01	iYoutube#01
End-to-End Machine Learning	iColab#02	pdf#02	iYoutube#02
Regression, Classification	iColab#03	pdf#03, pdf#04	iYoutube#03
Dimension Reduction	iColab#05	pdf#05	iYoutube#04
Clustering	iColab#06	pdf#06	iYoutube#05
Artificial Neural Networks (ANN)	iColab#07	pdf#07	iYoutube#06, iYoutube#07
Autoencoder	iColab#08	pdf#08	iYoutube#08
Convolutional Neural Networks (CNN)	iColab#09	pdf#09	iYoutube#09
Long Short-Term Memory (LSTM)	iColab#10	pdf#10	iYoutube#10
Reinforcement Learning (RL)		pdf#11	iYoutube#11
Value-based RL		pdf#12	iYoutube#12
Policy-based RL	iColab#13	pdf#13	iYoutube#13
기계공학에서 LLM 활용	pdf_ChatGPT4	pdf#14	iYoutube#14
Language-based Robotics		pdf#15	iYoutube#15
ChatGPT 활용 화성 탐사선 제어		pdf#16	iYoutube#16

Table of Contents

Linear regression

Living area ($feet^2$)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮





Linear regression

- How can we PREDICT the target?

x: input(feature)
$h_\theta(x)$: output(target) θ_i: parameters (also called weights)

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

area # bedrooms

- How can we EVALUATE the error of our function?

→ Cost function (loss function):

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^i) - y^i)^2$$

mean squared error (MSE)

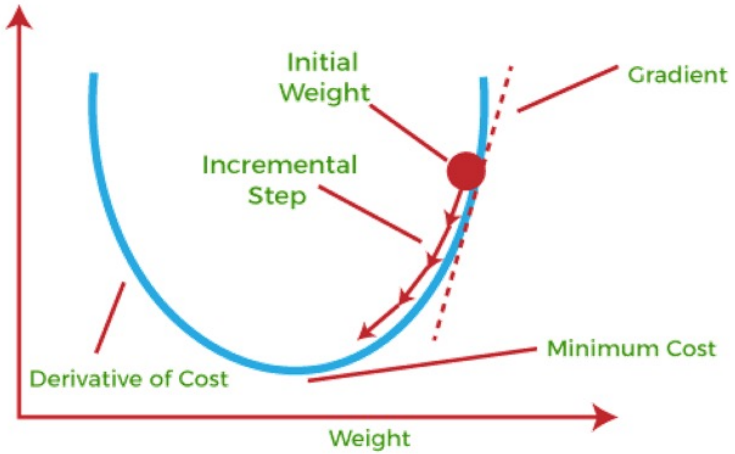
Linear regression

- How can we FIND the optimal values for the parameters?

  → Gradient descent (GD) algorithm

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

α: learning rate (hyperparameter)



- In MSE loss function (least mean square)

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\
&= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (\theta_0 x_0 \cdot \cdot \theta_j x_j \cdot \cdot) \\
&= (h_\theta(x) - y) x_j
\end{aligned}$$

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} (y^i - h_\theta(x^i)) x_j^i, (for\ every\ j)$$

i for number of training samples

j for number of parameters

Partial differential equations

- **general PDE form**:

$$\mathcal{F}\left(\mathbf{u}(\mathbf{x},\mathrm{t}),\nabla\mathbf{u}(\boldsymbol{x},t),\nabla^2\mathbf{u}(\boldsymbol{x},t)\right)=0,\qquad(\mathbf{x},t)\in\Omega\times[0,T]$$

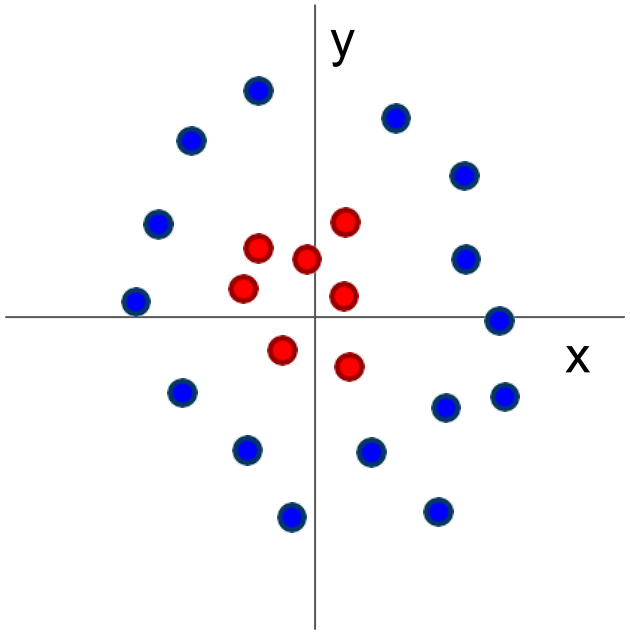
- example 1) **Navier-Stokes equation**: $\mathcal{F}(\mathbf{u},\nabla\mathbf{u},\nabla^2\mathbf{u})=\frac{\partial\mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}+\nabla p-\nu\nabla^2\mathbf{u}$

- example 2) **Fourier's Law**: $\mathcal{F}(T,\nabla^2T)=\frac{\partial T}{\partial t}-\alpha\nabla^2T$

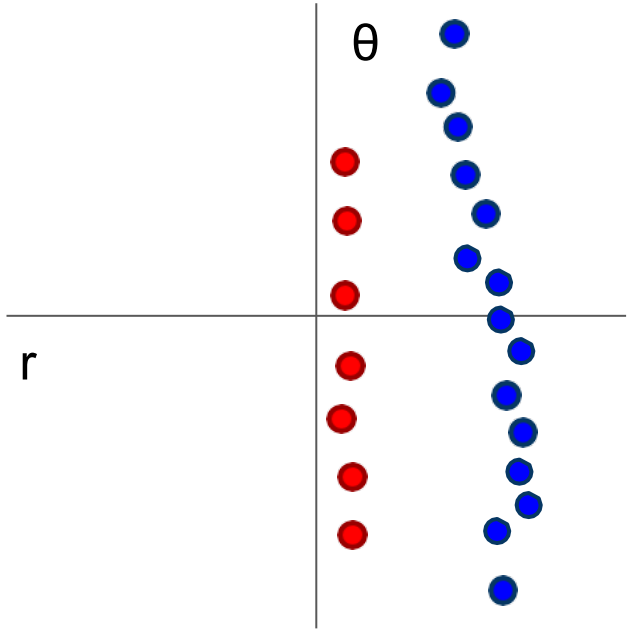
- example 3) **Faraday's Law**: $\mathcal{F}(B,\nabla\times E)=\frac{\partial B}{\partial t}+\nabla\times E$

https://www.youtube.com/watch?v=65yw9klPklk

Kernel method



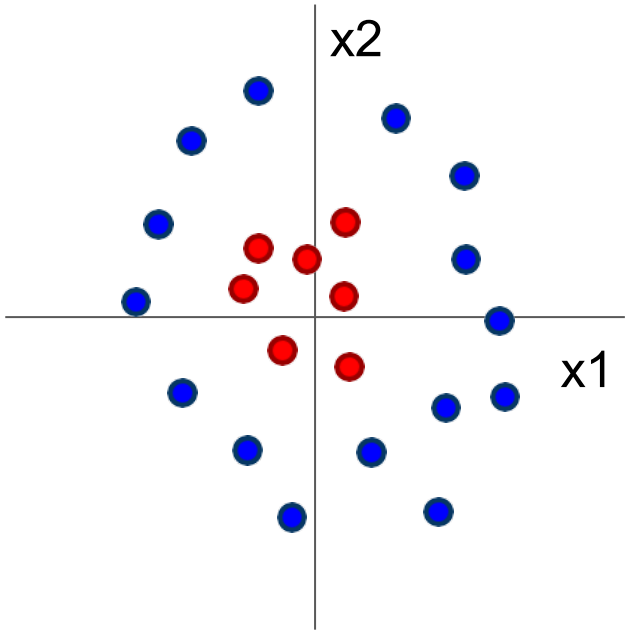
$$f(x, y) = (r(x, y), \theta(x, y))$$



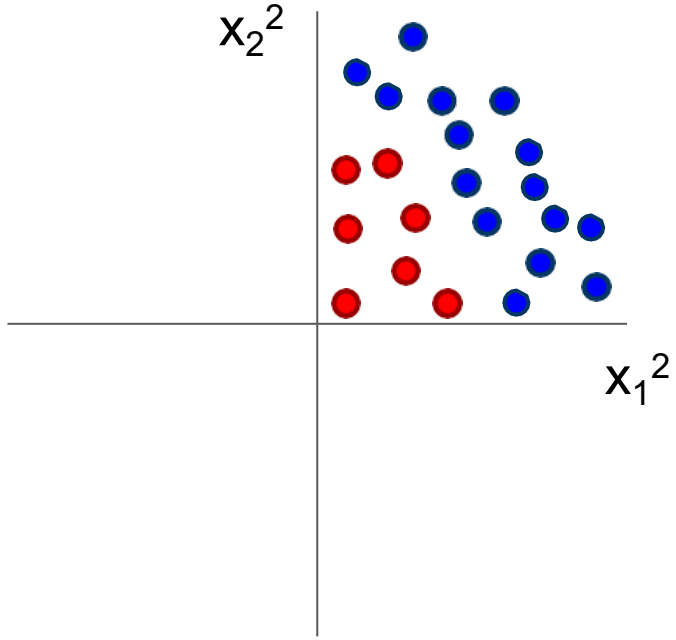
Cannot separate red and blue points with linear classifier

After applying feature transform, points can be separated by linear classifier

Kernel method



$(x_1, x_2) \rightarrow (x_1^2, x_2^2)$



Cannot separate red and blue points with linear classifier

After applying feature transform, points can be separated by linear classifier

Polynomial regression

- For a given target function $f(x)$, we aim to approximate it as a polynomial centered at x_0 and parametrized by the coefficients $a_0, a_1, \cdots, a_n$.

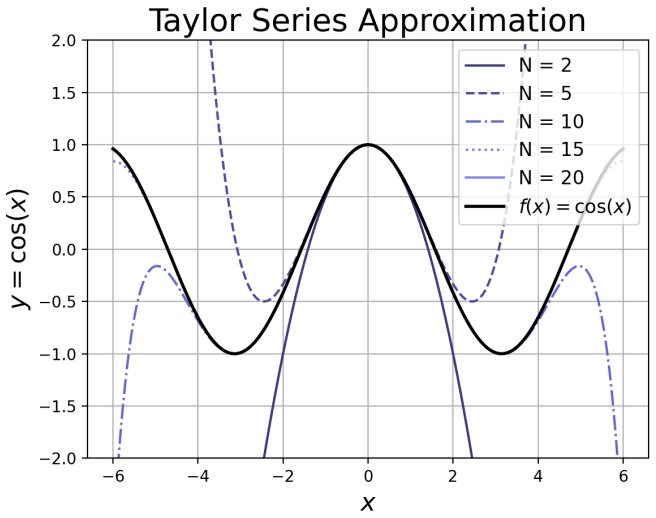
$$f(x) \approx a0 + a_1(x - x_0) + a_2(x - x_0)^2 + a_3(x - x_0)^3 + \cdots + a_n(x - x_0)^n$$

- How to determine the parameters (coefficients) which best approximate the target function?

$$a_k = \frac{f^k(x_0)}{k!}$$

- Is this a reliable approximation?

  Convergence of the Talyor Series



Trigonometric regression

- For a given target function $f(x)$, we aim to approximate it as a trigonometric function parametrized by the coefficients $a_1, \cdots, a_n$ and $b_1, \cdots, b_n$.

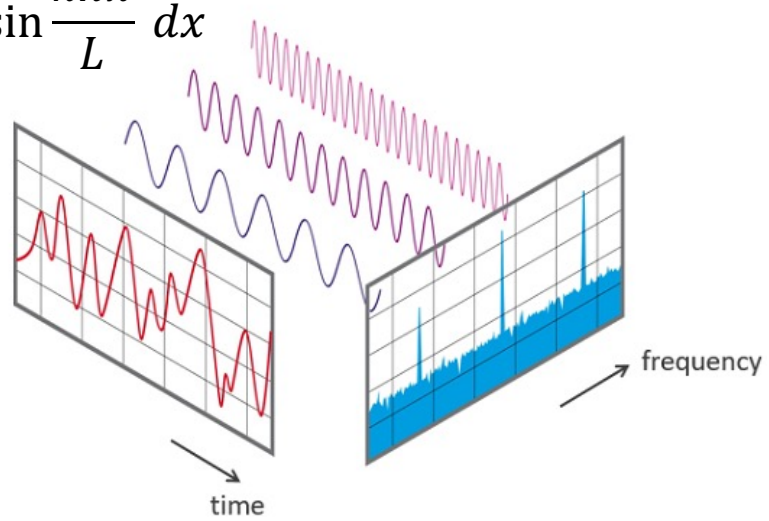
$$f(x) \approx \frac{a_0}{2} + \sum_{k=1}^{n} \left(a_k \cos\frac{k\pi x}{L} + b_k \sin\frac{k\pi x}{L}\right)$$

- How to determine the parameters (coefficients) which best approximate the target function?

$$a_k = \frac{1}{L}\int_{-L}^{L} f(x)\cos\frac{k\pi x}{L}\,dx\,, b_k = \frac{1}{L}\int_{-L}^{L} f(x)\sin\frac{k\pi x}{L}\,dx$$

- Is this a reliable approximation?

  Convergence of the Fourier Series



Things to study

- Assumption of linear regression
- Probabilistic interpretation of linear regression
- Generalized linear model

Table of Contents

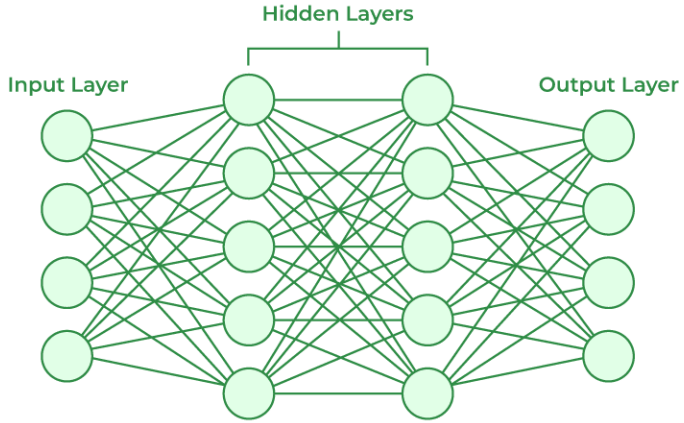
Machine learning (neural networks)

- For a given target function $f(x)$, we aim to approximate it as a an L-layer feed-forward neural network parametrized by θ (weight and bias).

$$f^1(x) = W^1 x + b^1$$
$$f^\ell(x) = W^\ell \sigma\left(f^{\ell-1}(x)\right) + b^\ell \text{for } 2 \le \ell \le L \implies f(x) \approx f^L$$

- How to determine the parameters (coefficients) which best approximate the target function?

  Set a loss function and minimize it ('learning' as well as 'approximation')



- Is this a reliable approximation?

  Universal approximation theorem

Neural networks: universal nonlinear function approximator

Universal Approximation Theorem

For a continuous function f, there exists a sequence of growing neural networks $\{f_n\}$ such that

$$\lim_{n\to\infty} \max_x |f_n(x) - f(x)| = 0.$$

- Arbitrary width case for two-layer neural networks ($L = 2$, $n_1 \to \infty$).

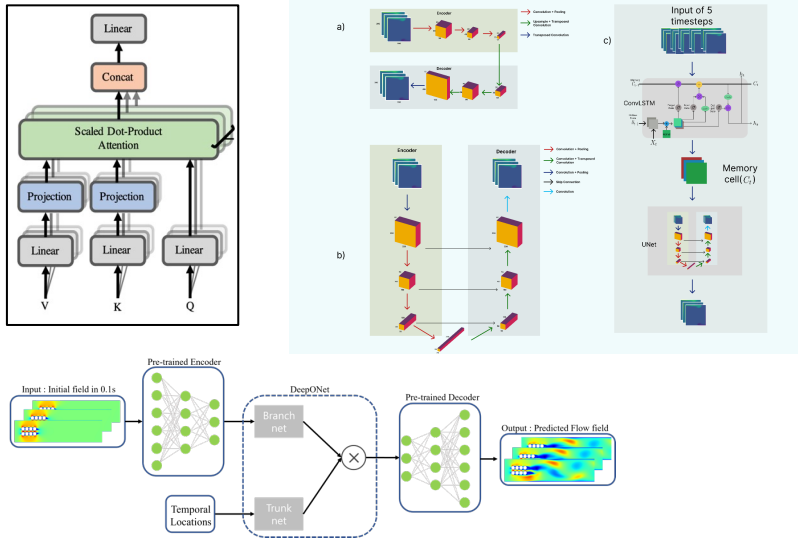
  G. Cybenko,
  Approximation by superpositions of a sigmoidal function,
  Math. Control Signals Systems, 2(4):303–314, 1989.

  K. Hornik,
  Approximation capabilities of multilayer feedforward networks,
  Neural Networks, 4(2):251–257, 1991.

- Bounded width and arbitrary depth case ($n_i < \infty$, $L \to \infty$).

  P. Kidger and T. Lyons,
  Universal Approximation with Deep Narrow Network,
  Proceedings of Machine Learning Research, 2306–2327, 2020.

Deep learning architectures

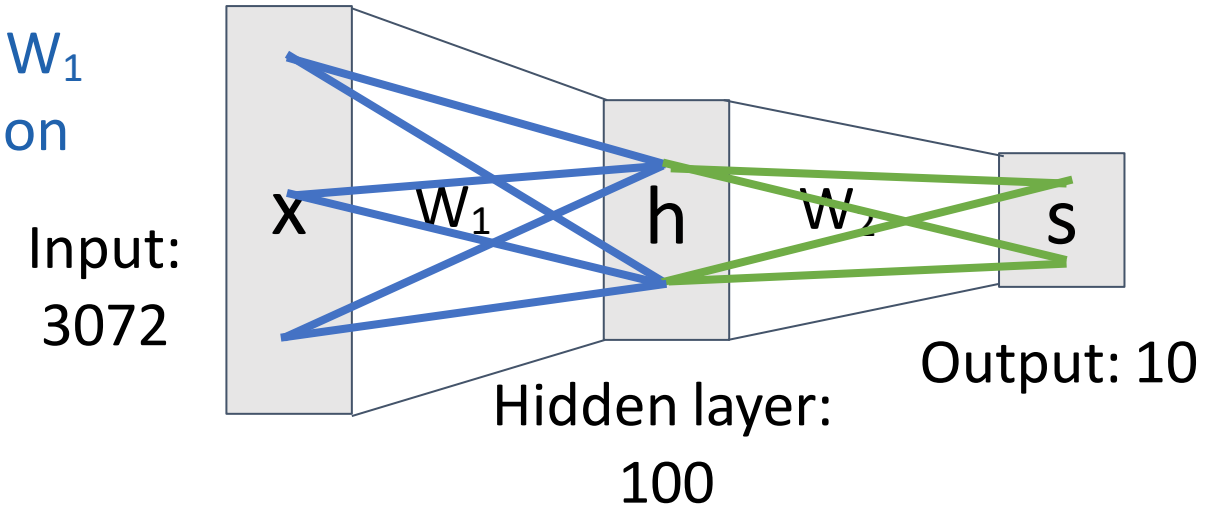


MLP

Before: Linear classifier $f(x) = Wx + b$

Now: 2-layer Neural Network $f(x) = W_2 \max(0, W_1 x + b_1) + b_2$

Element (i, j) of W_1 gives the effect on h_i from x_j



Element (i, j) of W_2 gives the effect on s_i from h_j

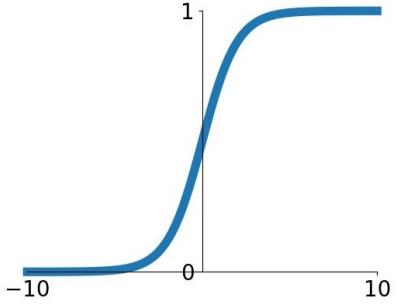
Fully-connected neural network

a.k.a. Multi-Layer Perceptron (MLP)

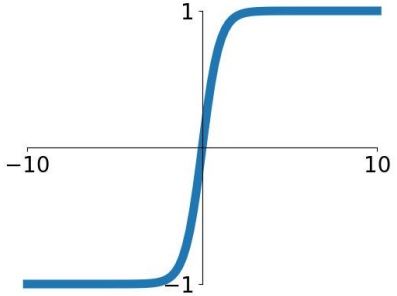
Activation Functions

Sigmoid

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

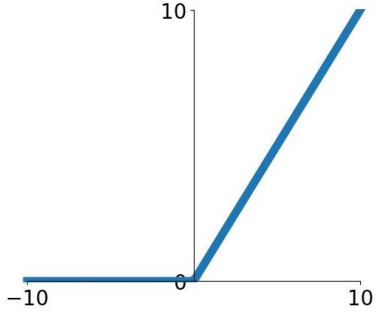


$$\tanh(x)$$



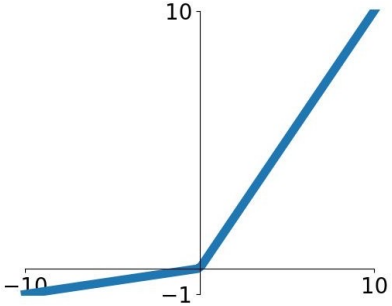
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

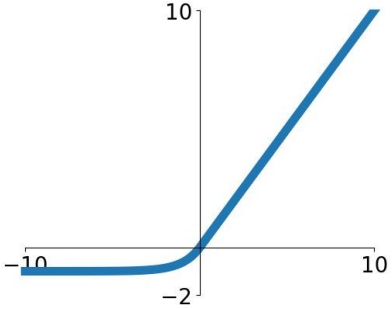


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

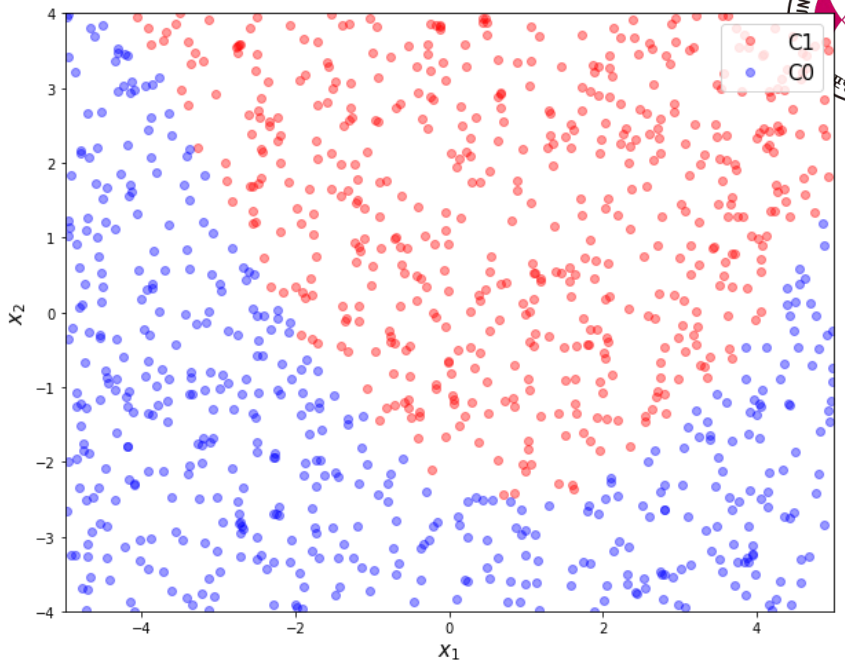
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Nonlinearly distributed data

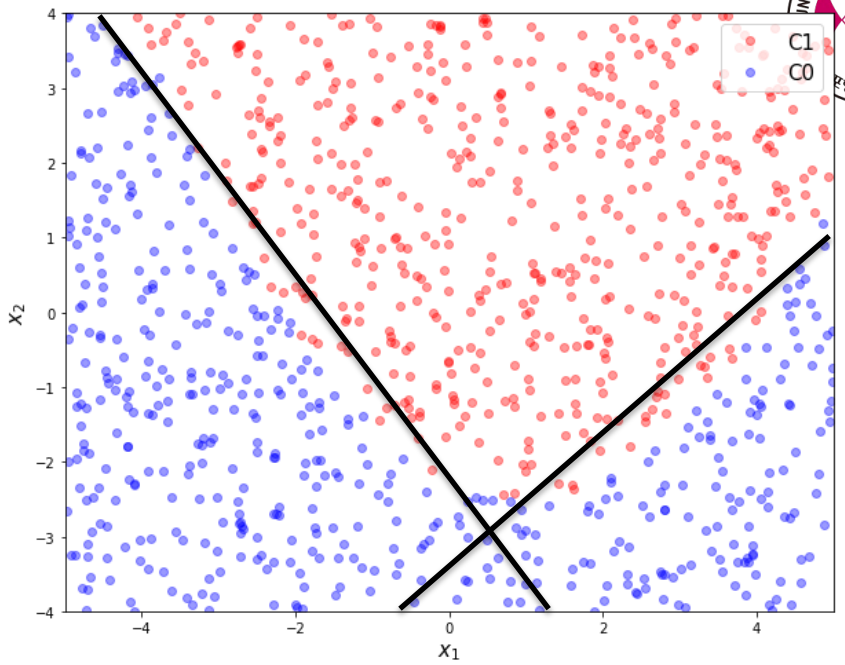
- Example to understand network's behavior
  - Include a hidden layer

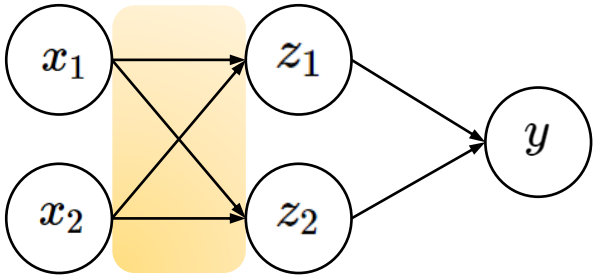


Nonlinearly distributed data

- x space

- Nonlinear classifier in original space
- Piecewise linear classification boundaries

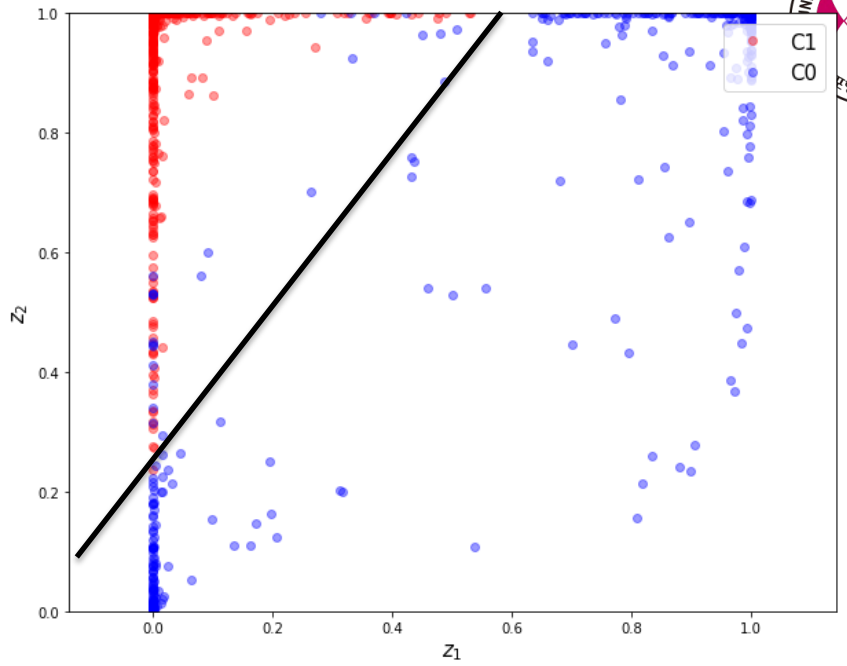


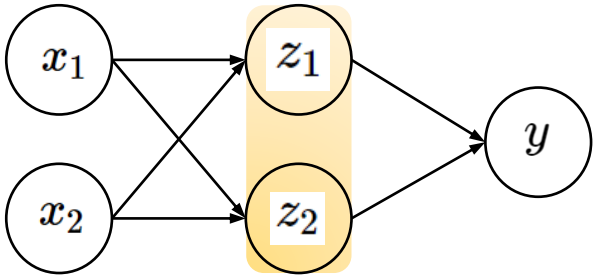


Nonlinearly distributed data

- z space

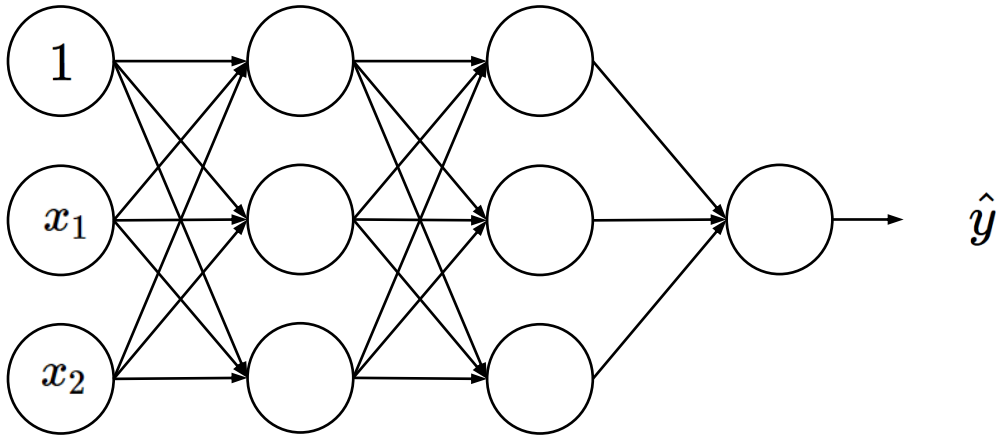
- Kernel
- Linearly separable

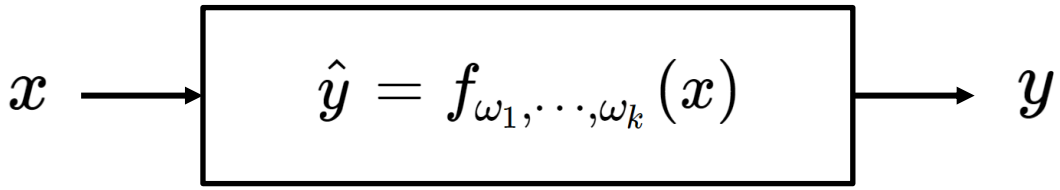




Summary of neural networks

- Universal function approximator
- Nonlinear mapping can be represented by another neurons



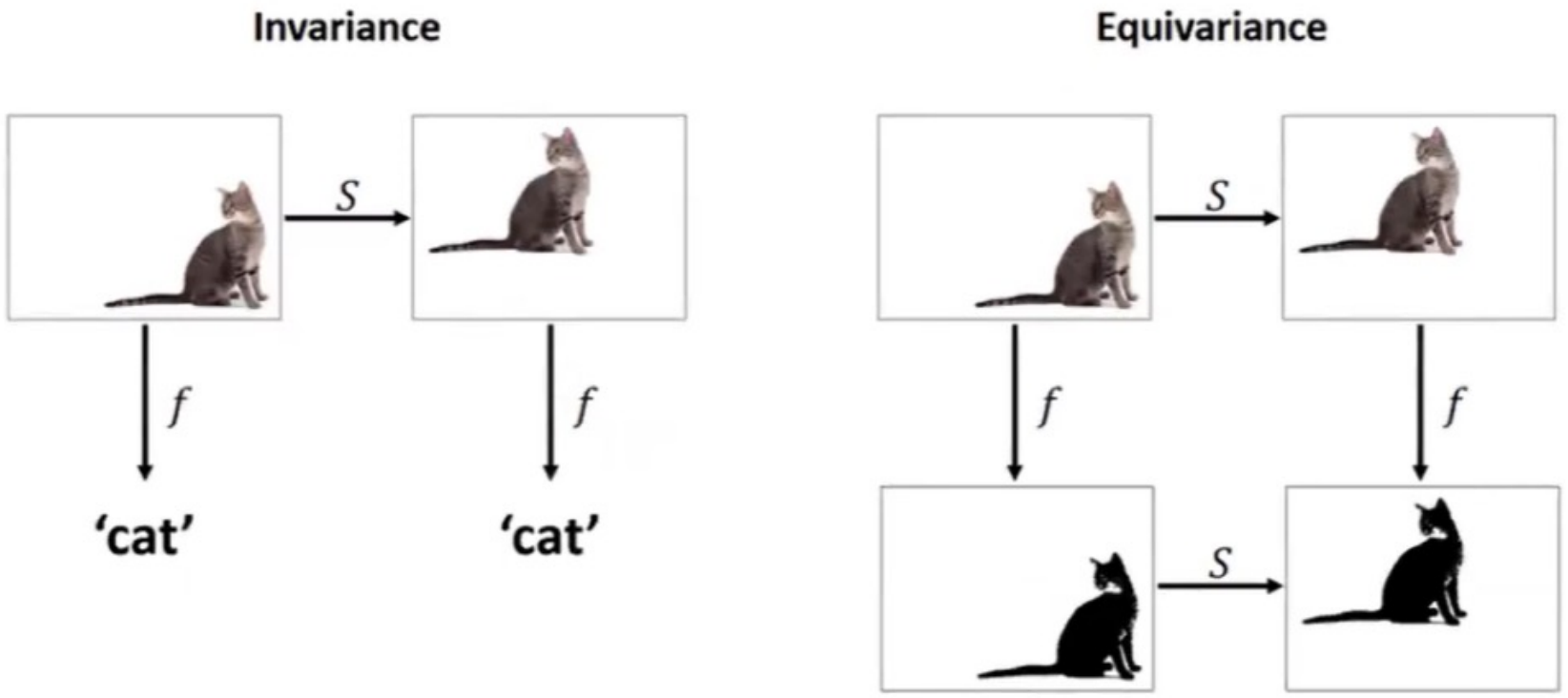


Things to study

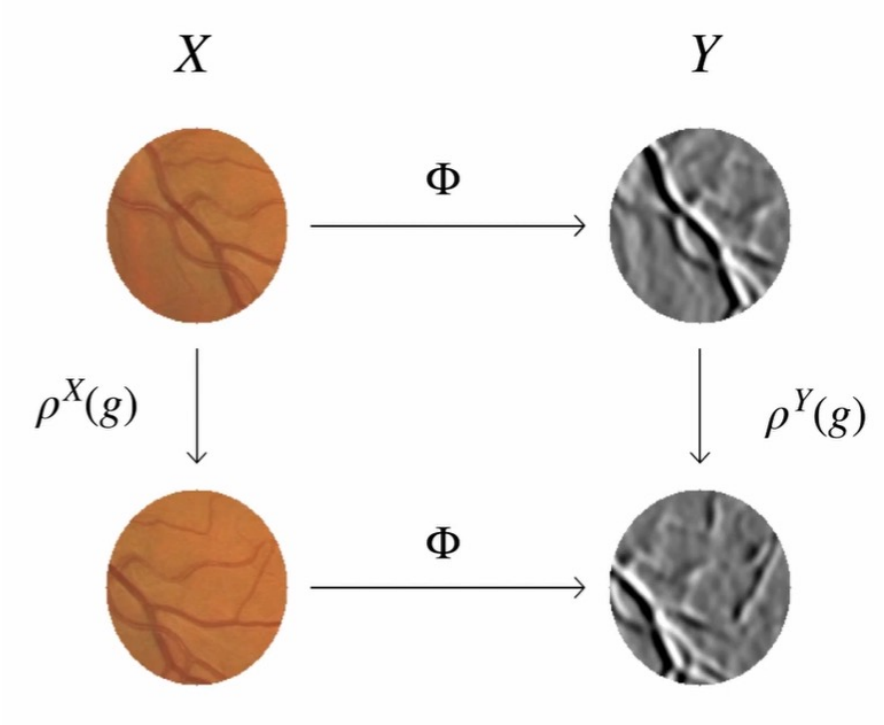
- Feed-forward propagation
- Backpropagation and chain rule
- Generalized linear model

Table of Contents

Invariance vs Equivariance



Invariance vs Equivariance



$\mathbf{\Phi}: X \to Y$ (neural network layer)

Equivariance: $\mathbf{\Phi} \circ \rho^X(\mathrm{g}) = \rho^Y(\mathrm{g}) \circ \mathbf{\Phi}$

Invariance: $\mathbf{\Phi} \circ \rho^X(\mathrm{g}) = \mathbf{\Phi}$

Invariance vs Equivariance

① Fully connected layer (or MLP) is invariant or equivariant?

② Convolutional layer is invariant or equivariant?

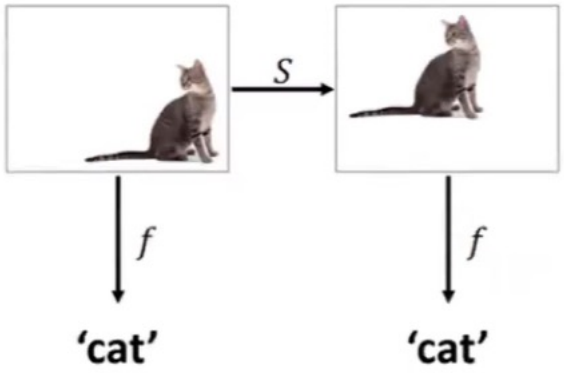
Invariance vs Equivariance

① Fully connected layer (or MLP) is invariant or equivariant?

: Neither

② Convolutional layer is invariant or equivariant?

: Equivariant



Invariance vs Equivariance

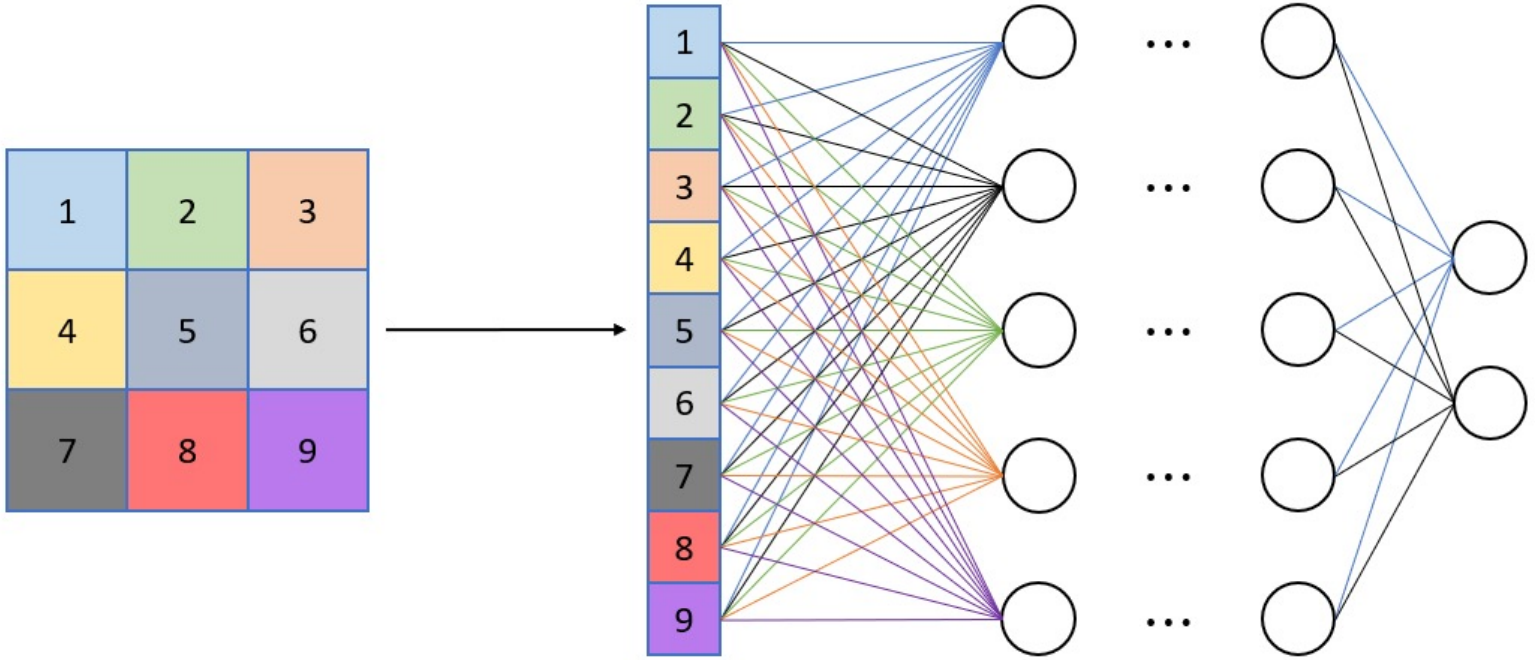
① MLP: no equivariant

② CNN: spatial equivariant

③ RNN: temporal equivariant

④ Transformer: permutation equivariant

Limitation of FCN (classification)



It's all mixed up. It can learn invariant properties, but **it is inefficient** for large size image.

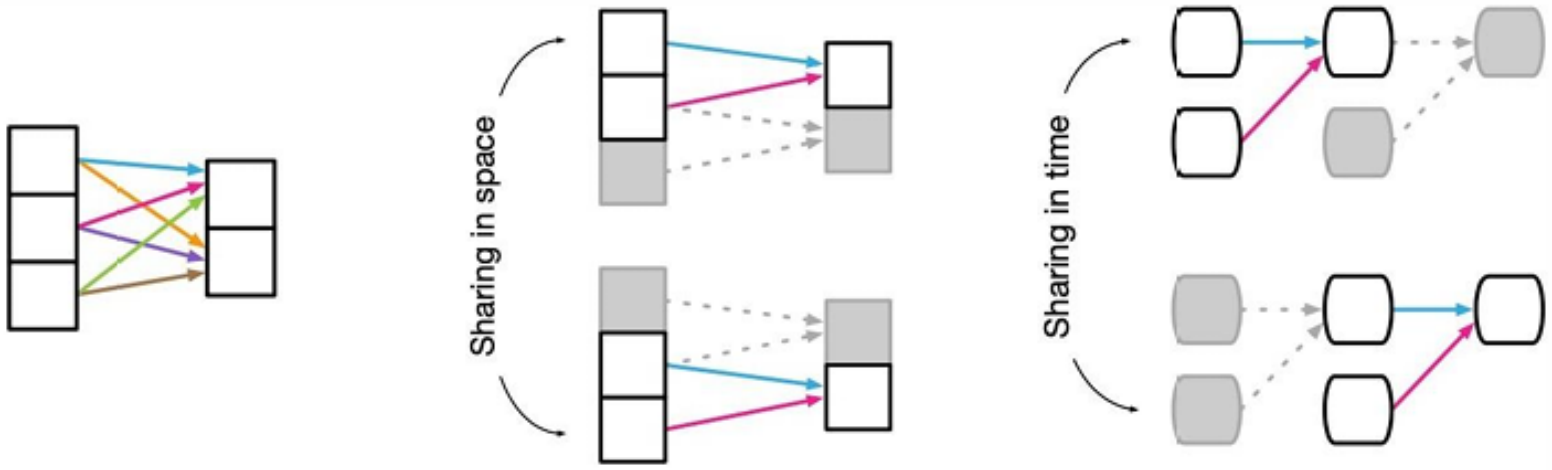
T. Son et al. (2023), *Sensors*

Model Types	Number of Parameters
MLP	64.08 M
CNN	2.3 M

medium.com, T. Son et al. (2023), *Sensors*

Table of Contents

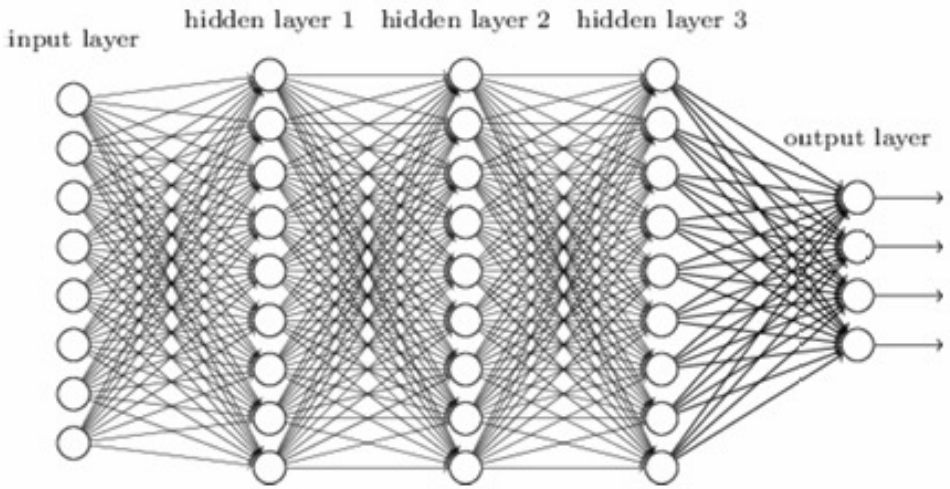
Relational inductive bias



(a) Fully connected (b) Convolutional (c) Recurrent

- Inductive bias is the set of assumptions that learning model uses to predict output for generalization.
- Relational inductive bias is the assumptions on relationships in input data.
- Relational inductive biases are one of the keys to modeling architectures such as FCN, ConvNet, RNN, GNN, or etc..

Relational inductive bias

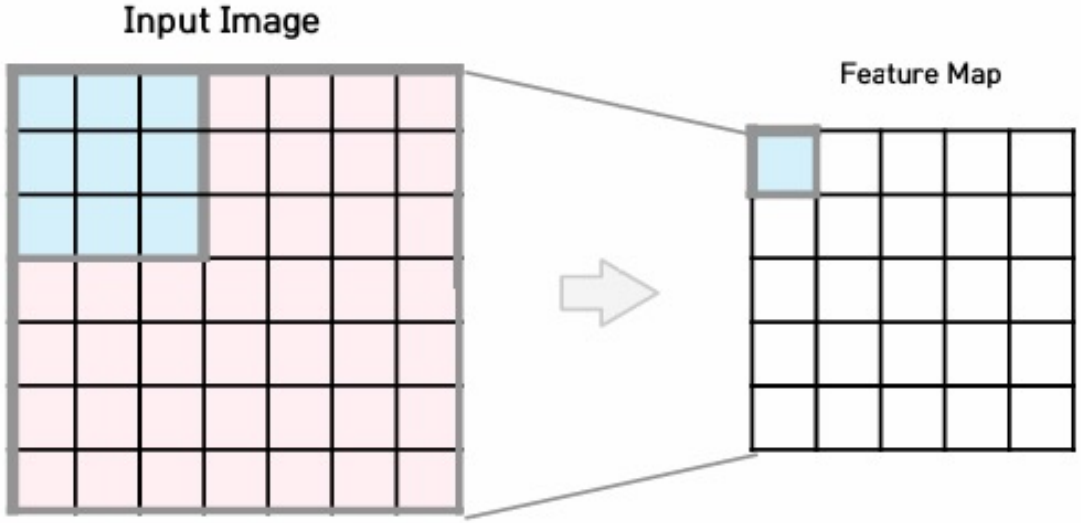


= Equivariant

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2.

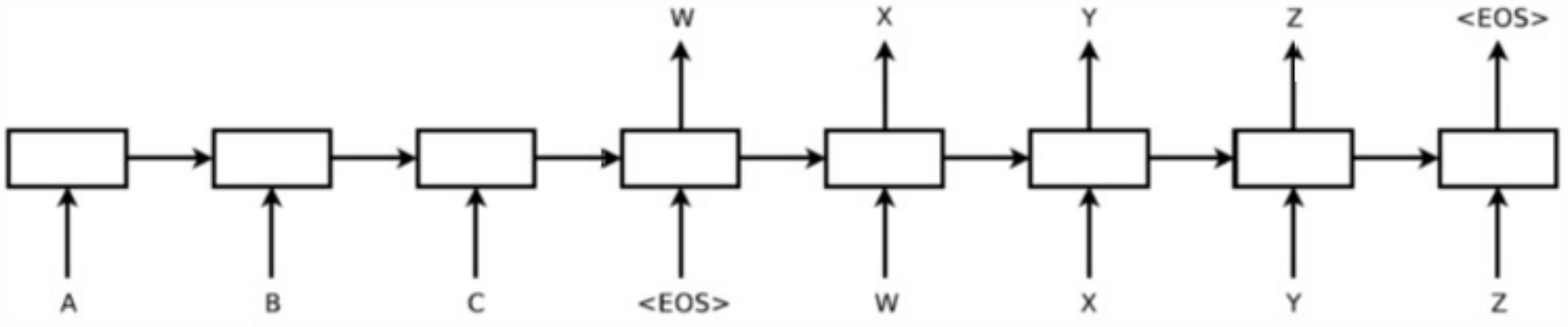
Relational inductive bias



Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2.

Relational inductive bias

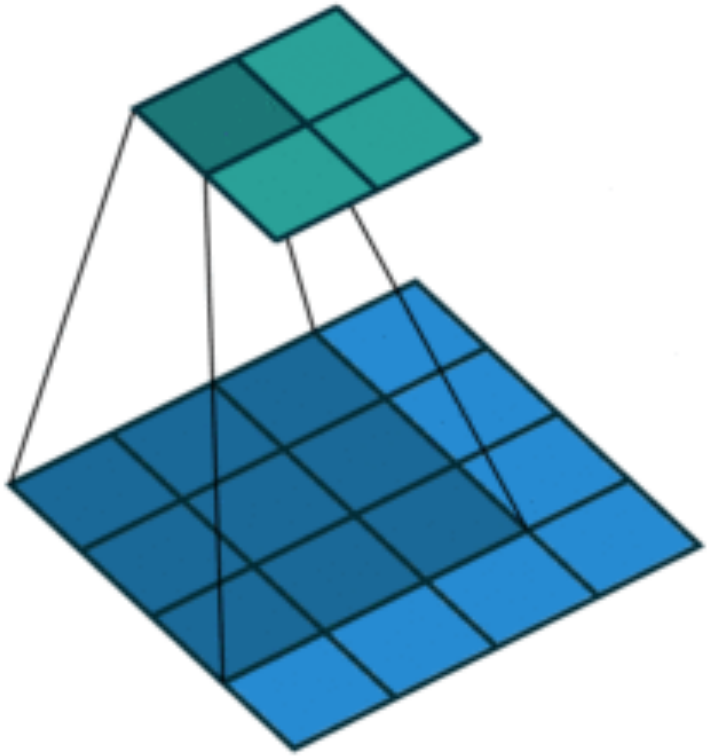


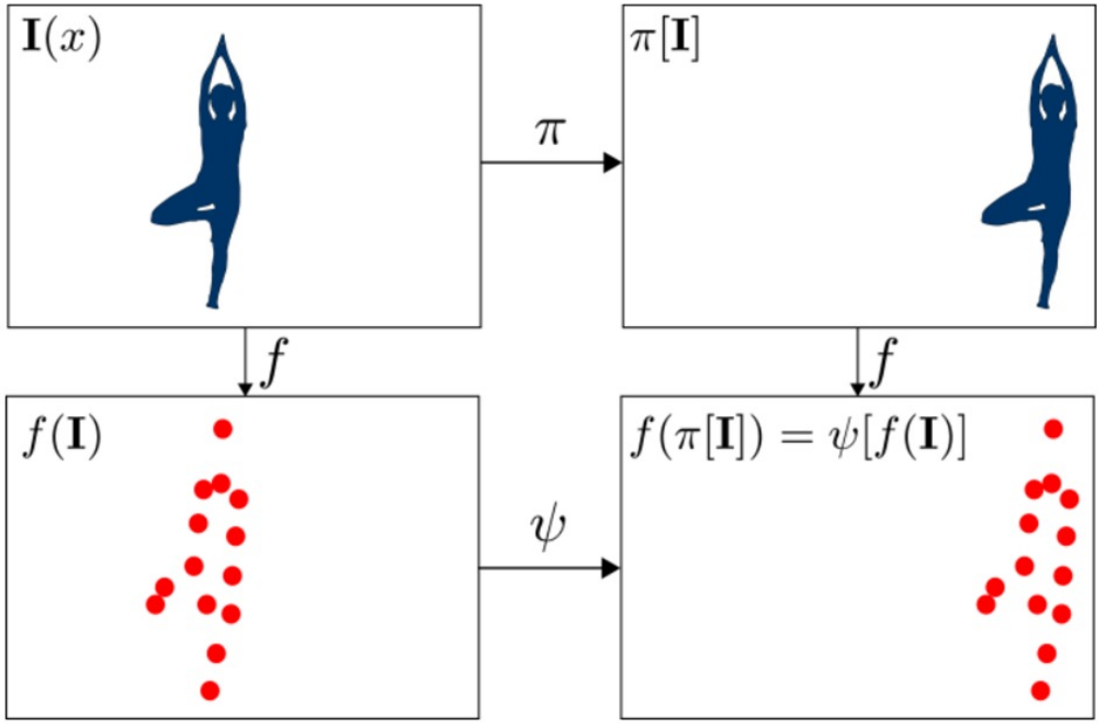
Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2.

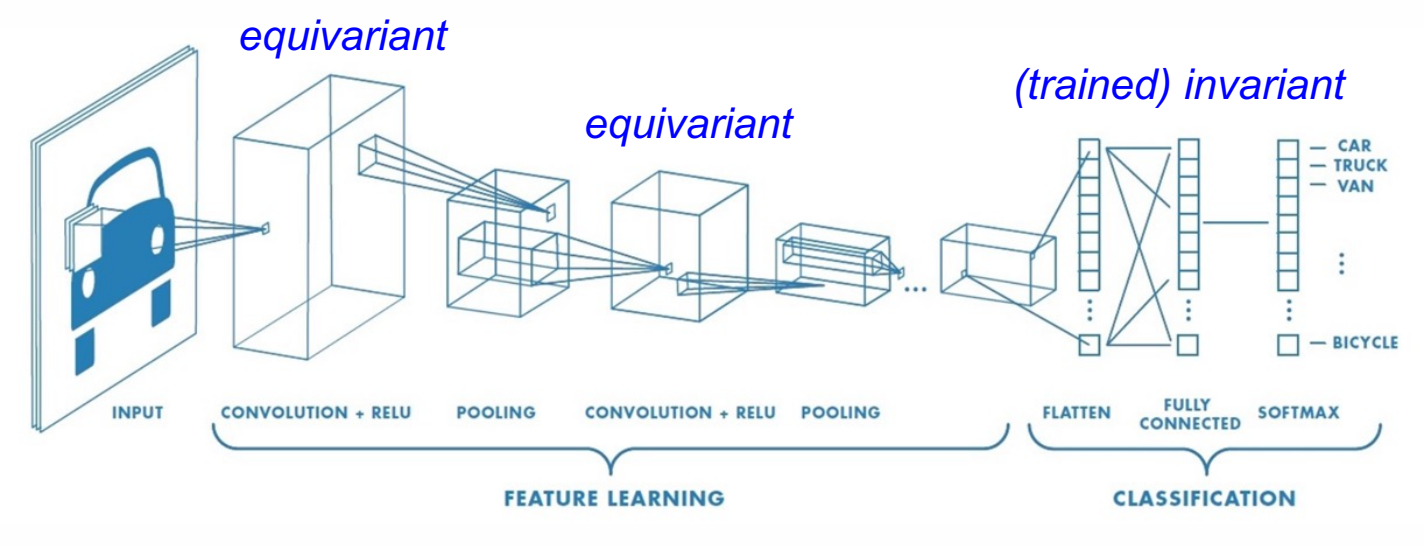
Convolutional layer

- Convolutional layer is spatial equivariant!





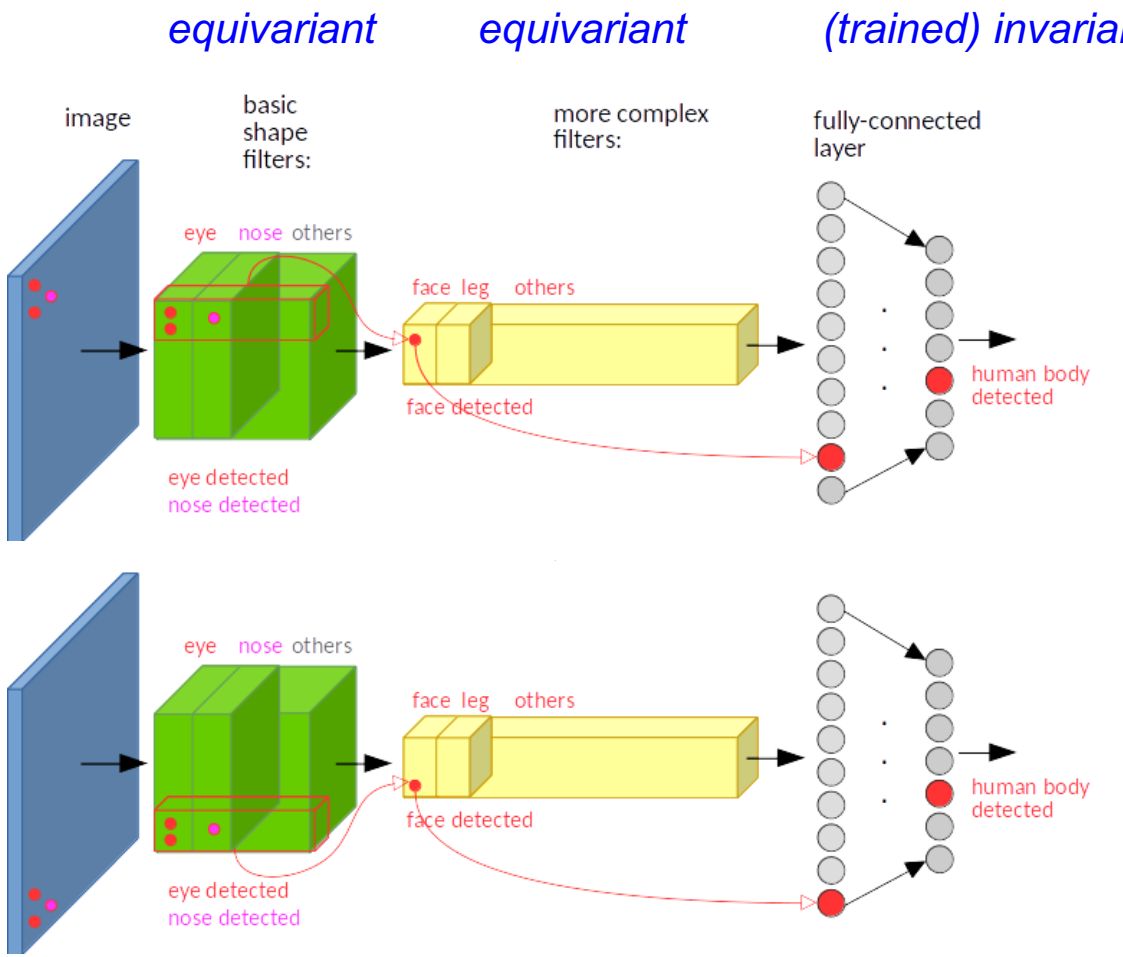
Principle of convolutional neural network (classification)



By preserving **equivariance longer**, the network retains **more detailed information**, enhancing its overall **performance**

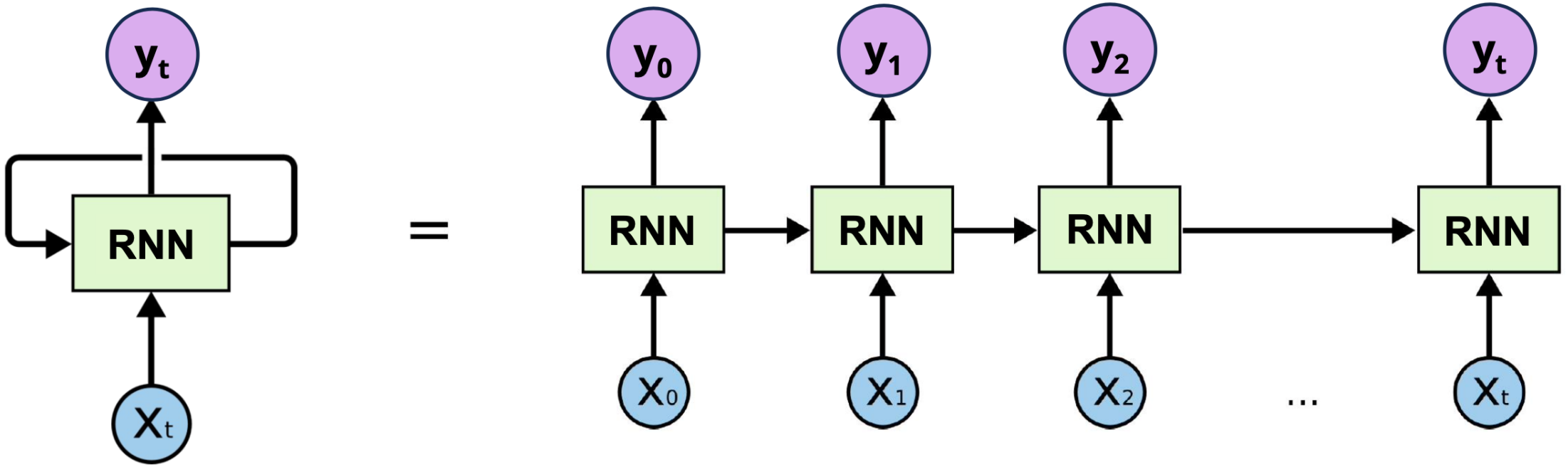
T. Cohen (2021), Equivariant convolutional networks, Doctoral dissertation.

Principle of convolutional neural network (classification)

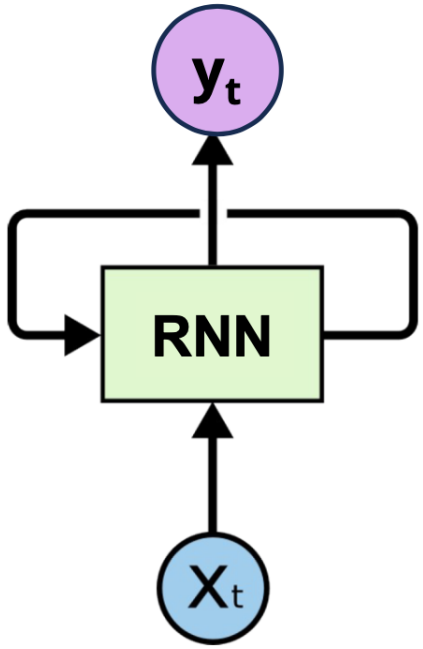


If we reduce the image size while **maintaining equivariant**, MLP **can learn invariant**.

Recurrent Neural Network (RNN)



Recurrent Neural Network (RNN)



We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

new state

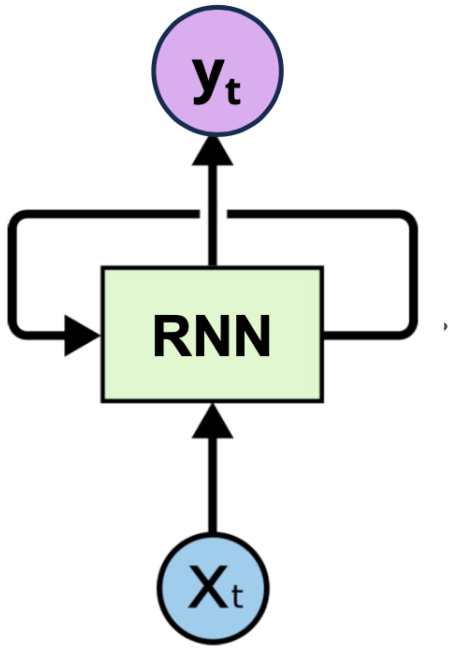
some function with parameters W

old state

input vector at some time step

Notice: the same function and the same set of parameters are used at every time step.

Recurrent Neural Network (RNN)



The state consists of a single *"hidden"* vector **h**:

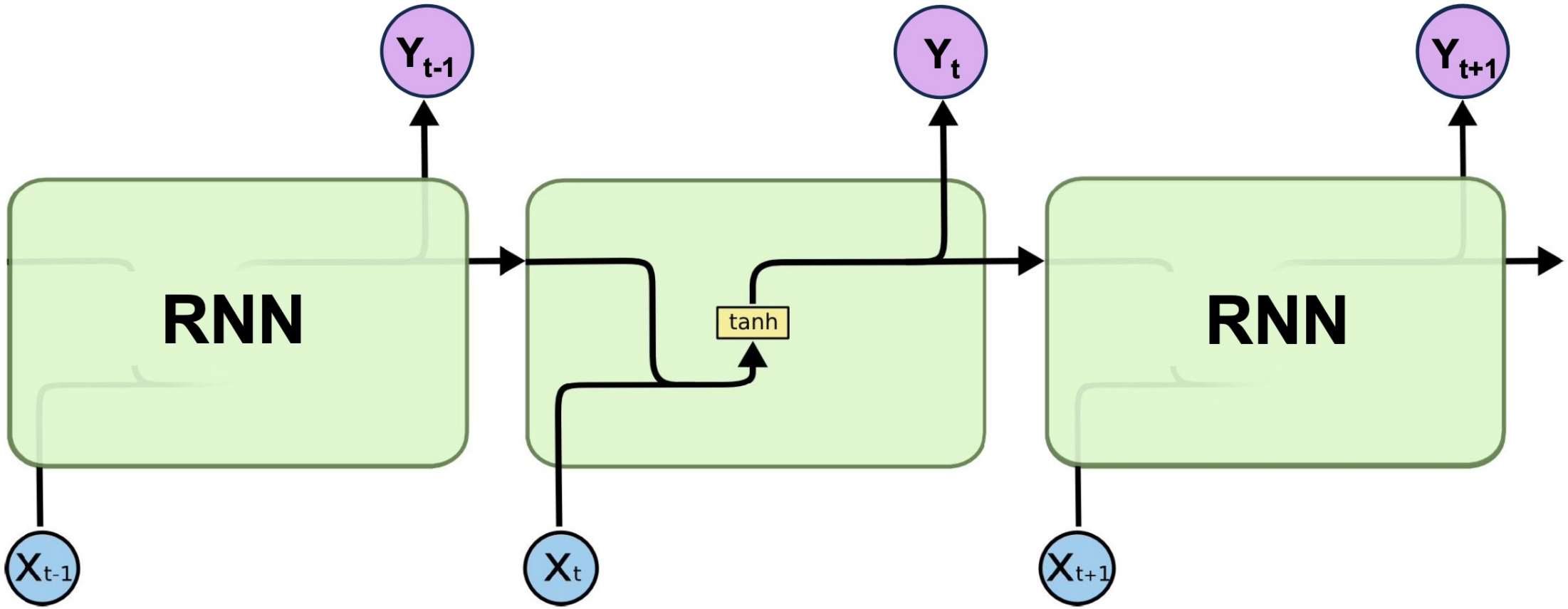
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

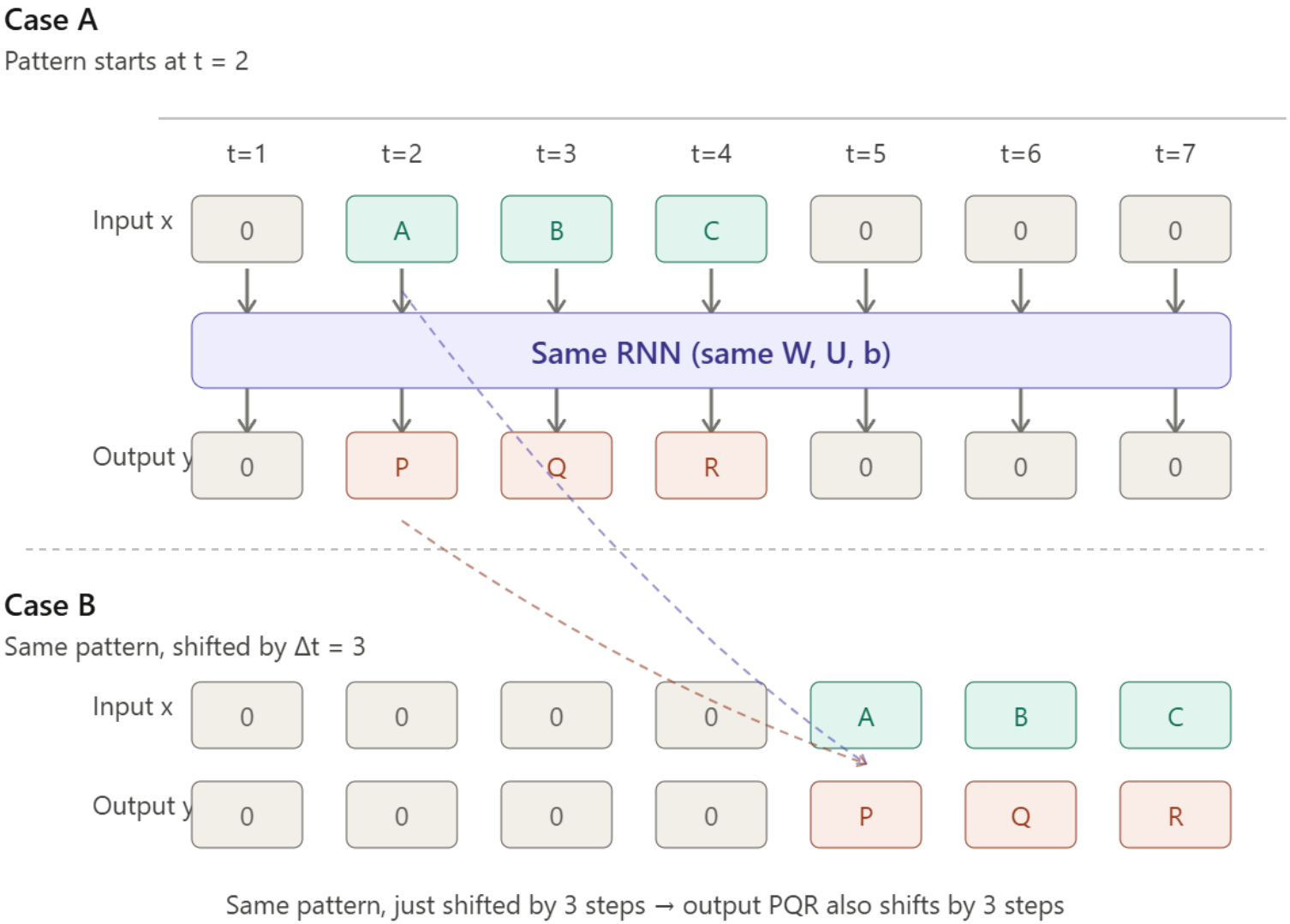
$$y_t = W_{hy}h_t + b_y$$

Sometimes called a "Vanilla RNN"

Recurrent Neural Network (RNN)



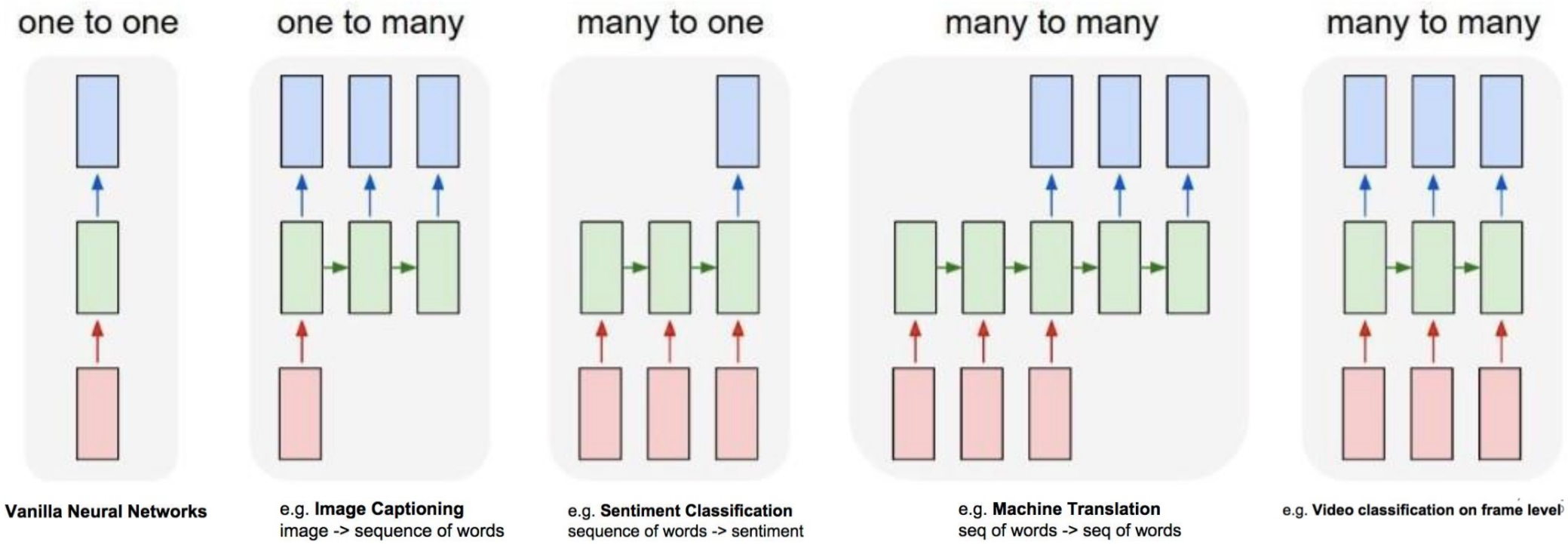
RNN is temporal equivariant!



Same pattern, just shifted by 3 steps → output PQR also shifts by 3 steps

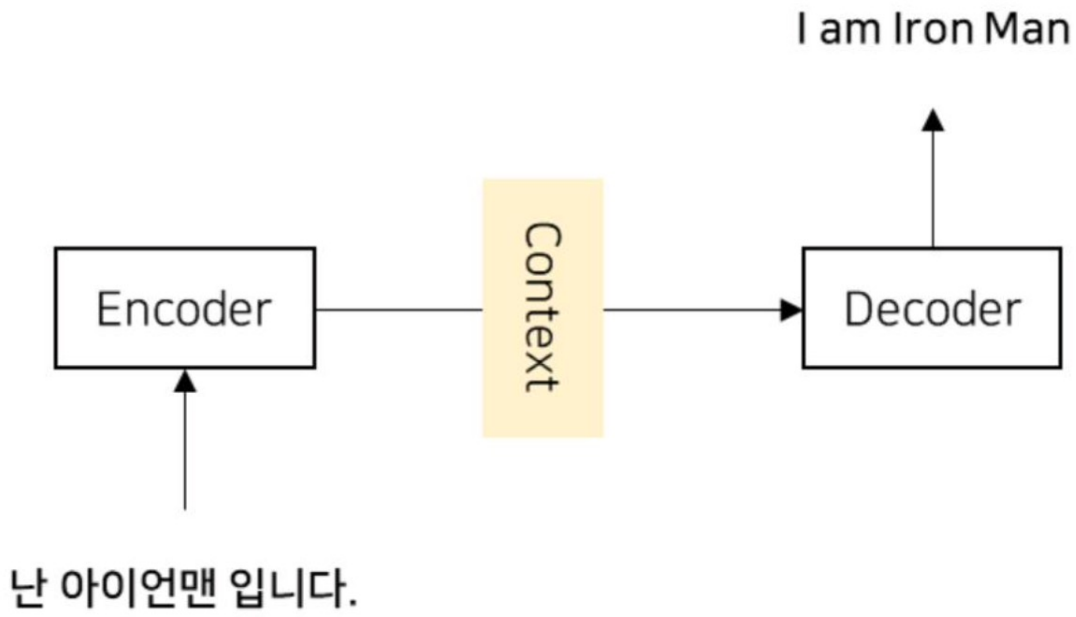
Recurrent Neural Network (RNN)

- Variable length of sequence can be input in RNN, because of shared weights at each time step. RNN has various structure according to task types



Sequence to Sequence (seq2seq): Many to One + One to Many

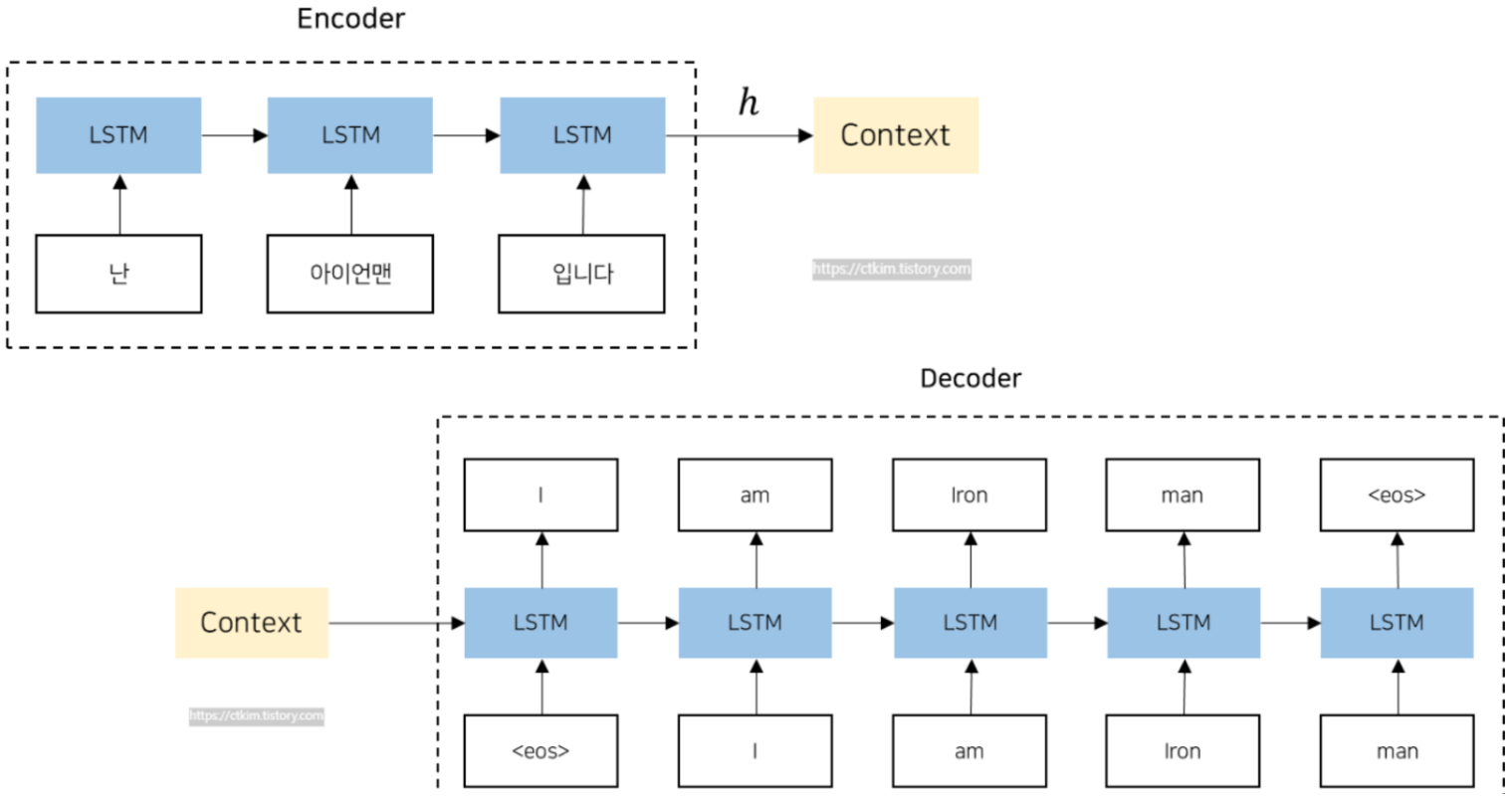
- Seq2seq is a model for training to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French).



https://ctkim.tistory.com/entry/RNN-seq2seq란 -무엇인가

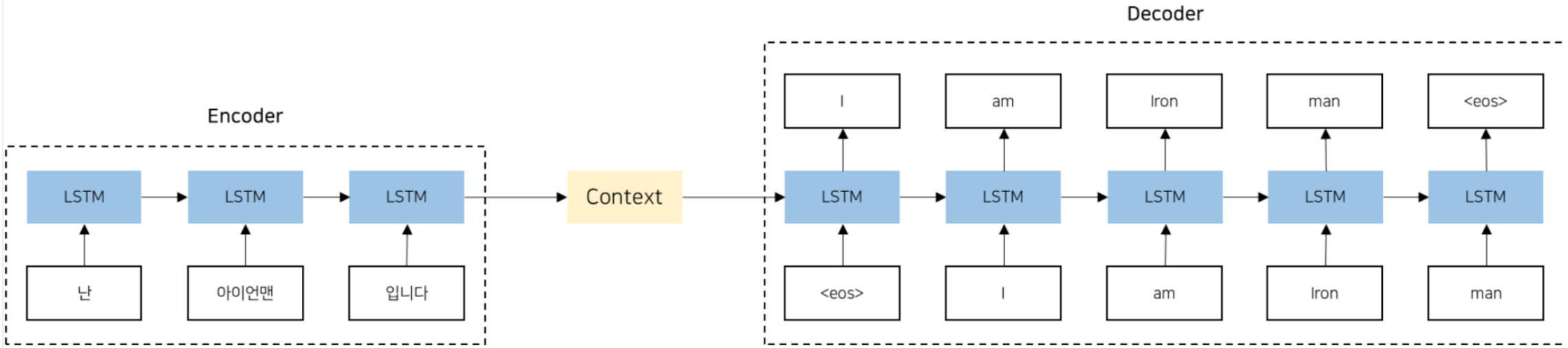
Sequence to Sequence (seq2seq): Many to One + One to Many

- Seq2seq is a model for training to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French).



Sequence to Sequence (seq2seq): Many to One + One to Many

- Seq2seq is a model for training to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French).



Problem 1) fixed size of context vector
Problem 2) gradient vanishing

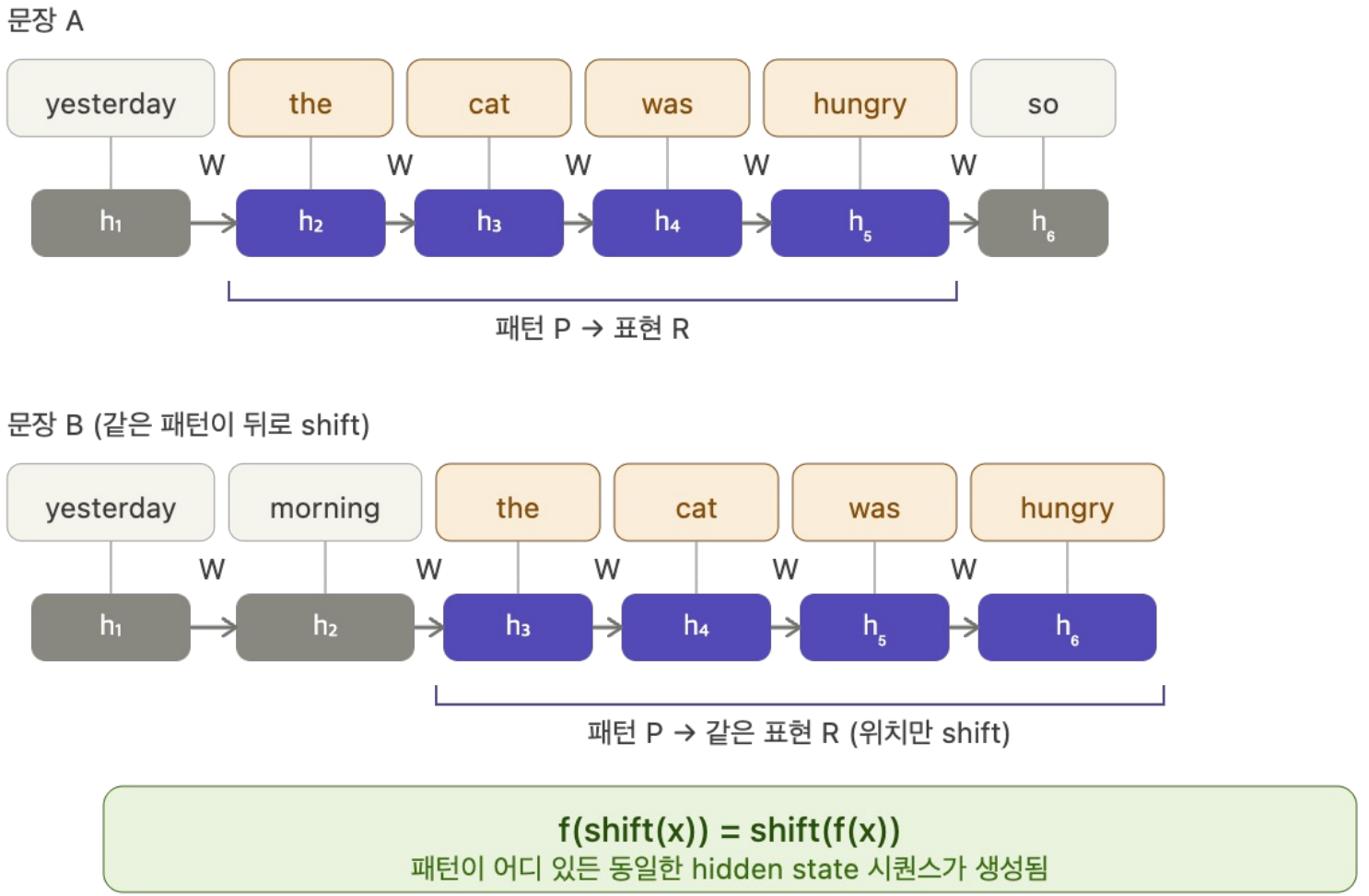
Things to study

- Feed-forward/backpropagation of CNN
- Feed-forward/backpropagation of RNN
- Feed-forward/backpropagation of LSTM
- Gradient vanishing problem

Table of Contents

Limitation of RNN

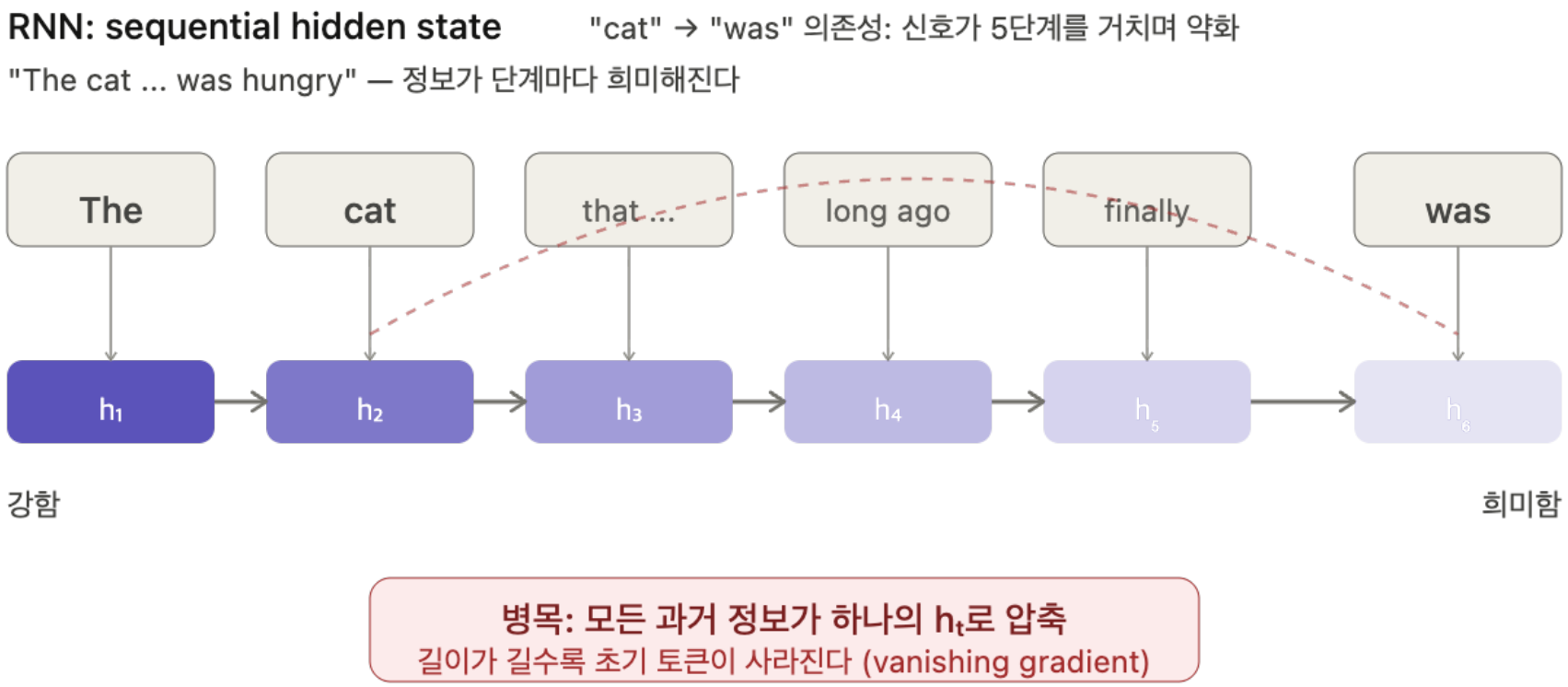
1) Limited long-range dependency modeling



f(shift(x)) = shift(f(x))
패턴이 어디 있든 동일한 hidden state 시퀀스가 생성됨

Limitation of RNN

1) Limited long-range dependency modeling



Limitation of RNN

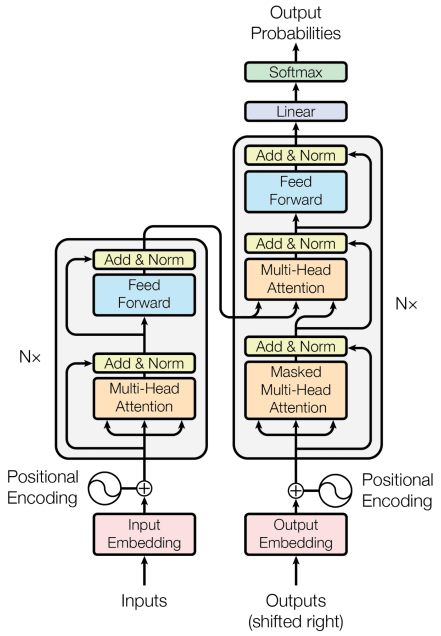
2) Weak contextualization

*“I swam across the river to get to the other **bank**.”*

*“I walked across the road to get cash from the **bank**.”*

Attention Mechanism: Attention is All You Need

- The fundamental concept that underpins a transformer is *attention*.
  - Originally developed as an enhancement to RNNs for machine translation.
  - Later showed significantly improved performance by eliminating the recurrence structure and focusing exclusively on the attention mechanism.



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Vaswani *et al.* (2017)

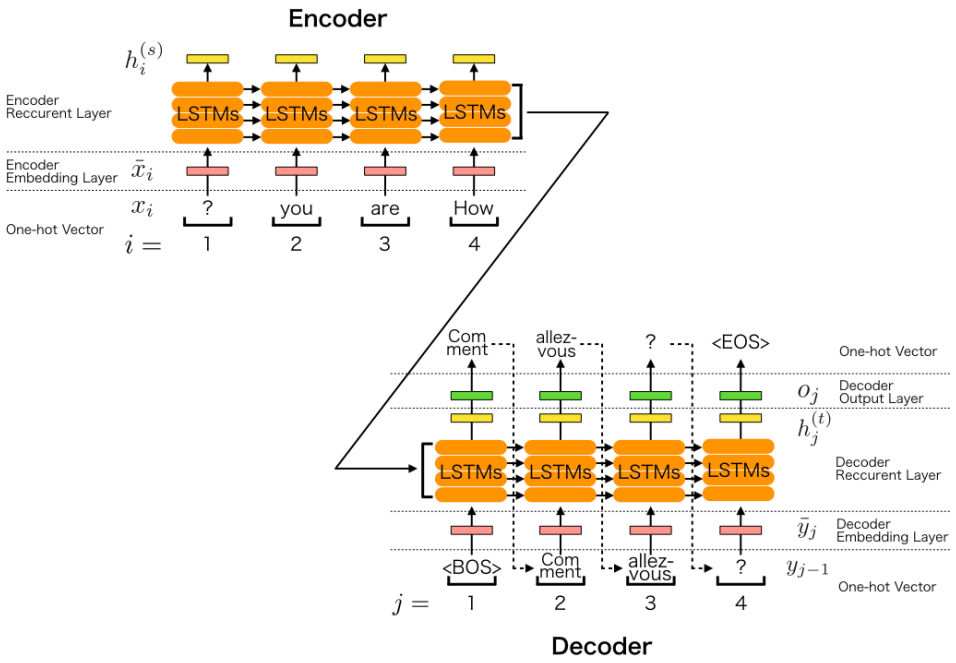
Recurrent Neural Network (RNN)

RNN

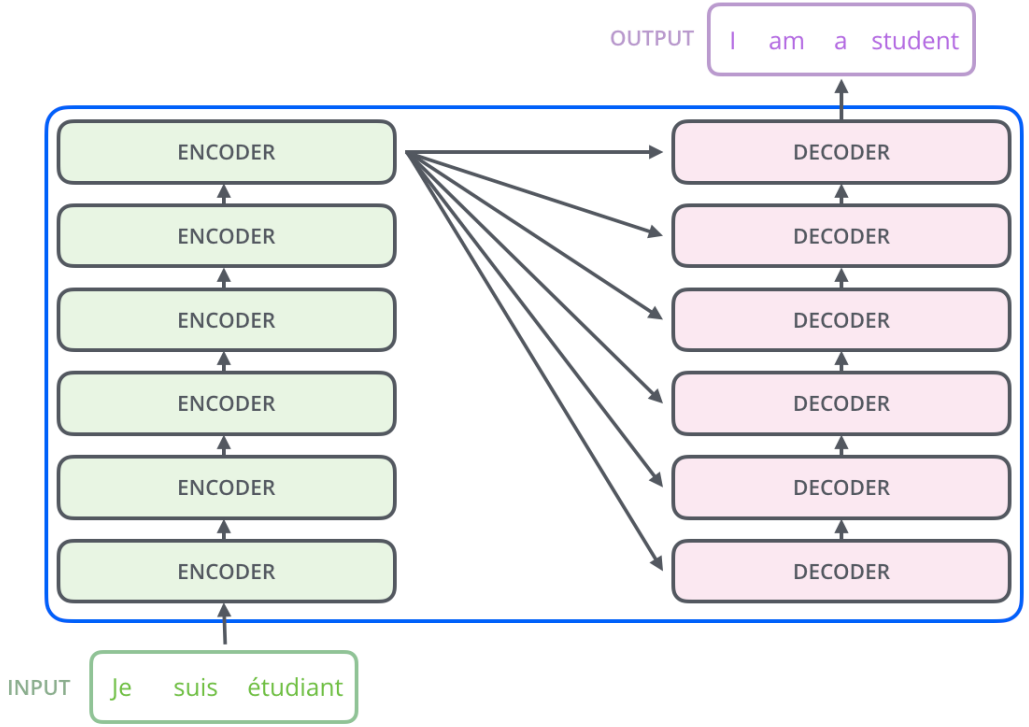
LSTM

Transformer

Transformer Overview

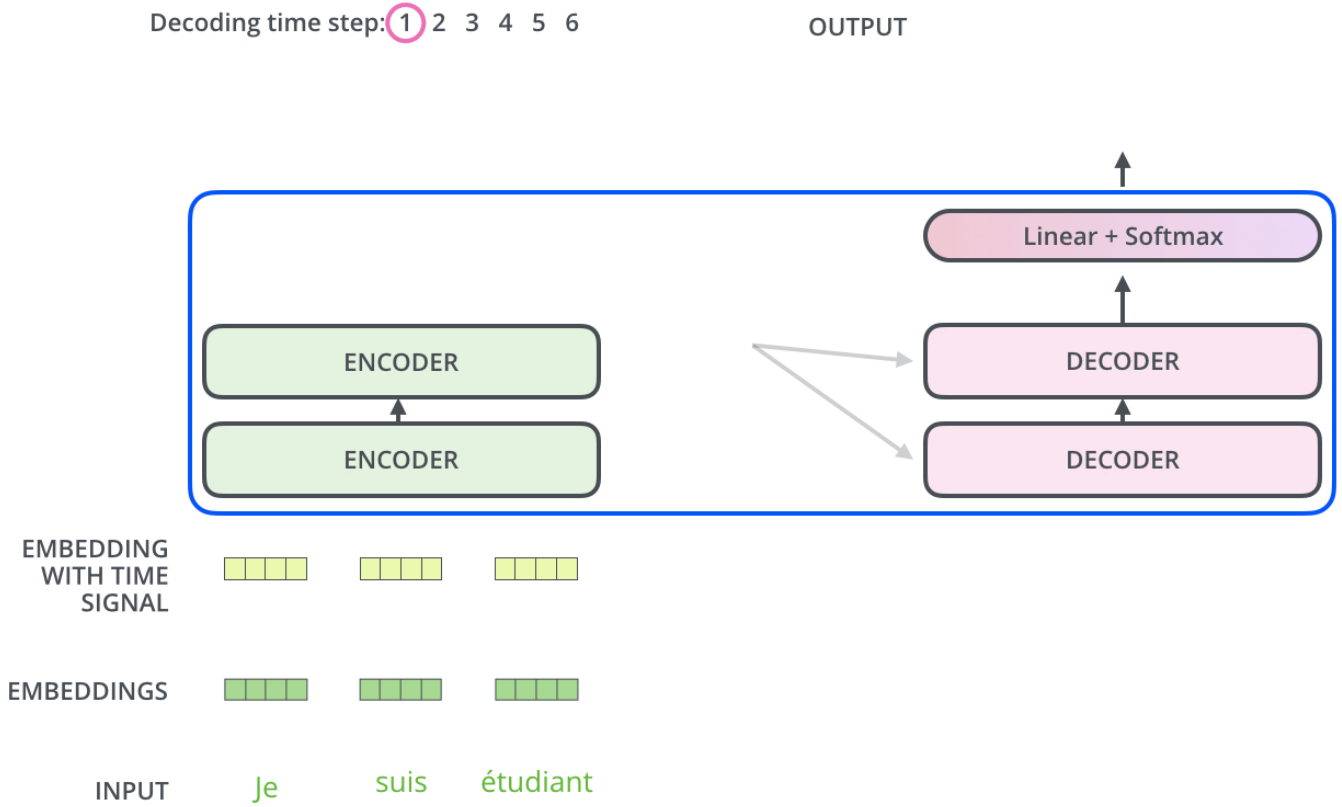


RNN architecture

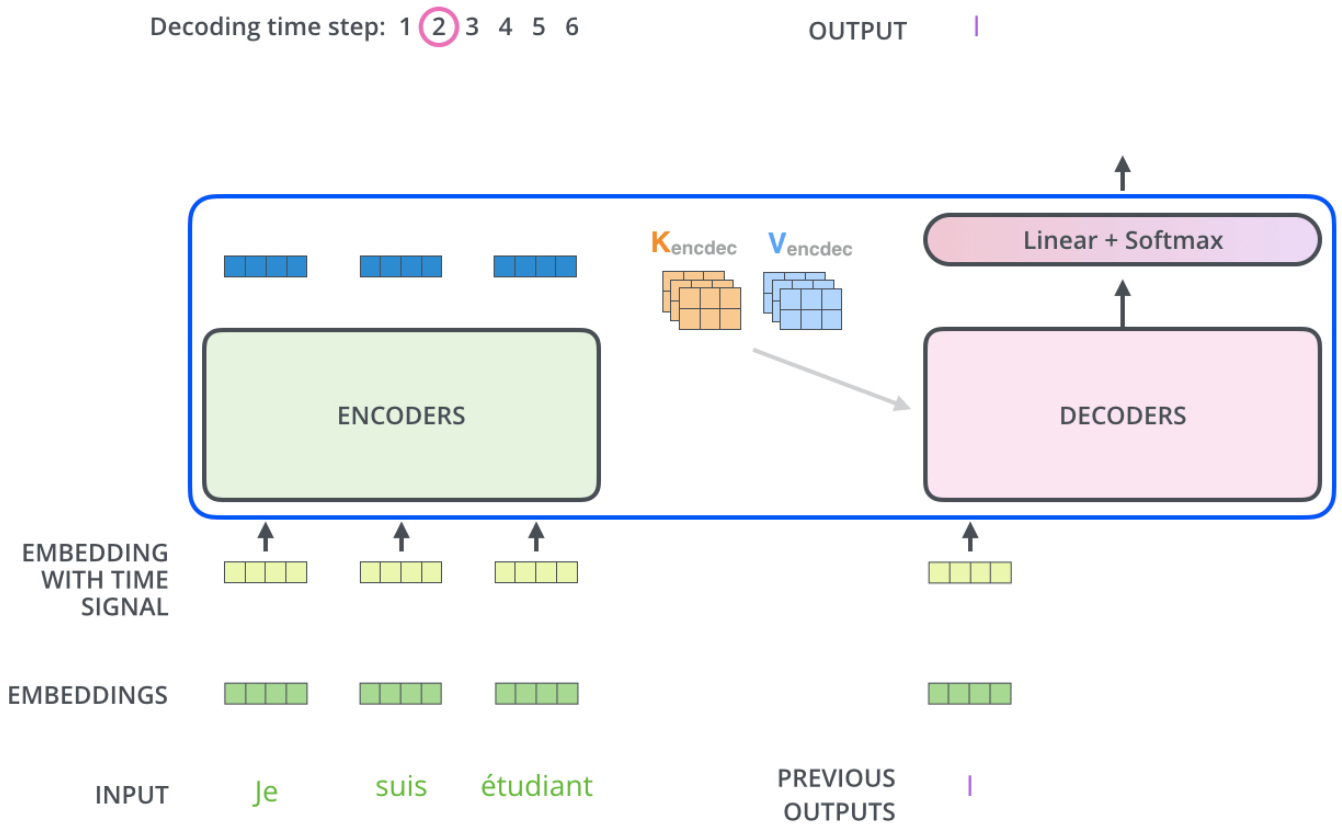


Transformer architecture

Transformer Overview

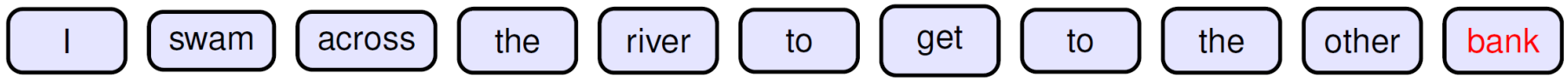


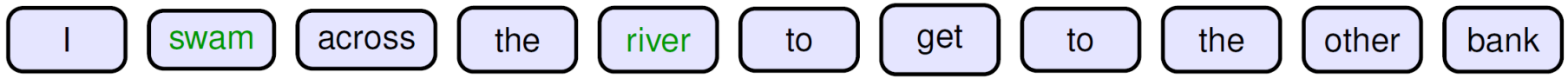
Transformer Overview



https://jalammar.github.io/illustrated-transformer/

Key Concepts in Attention





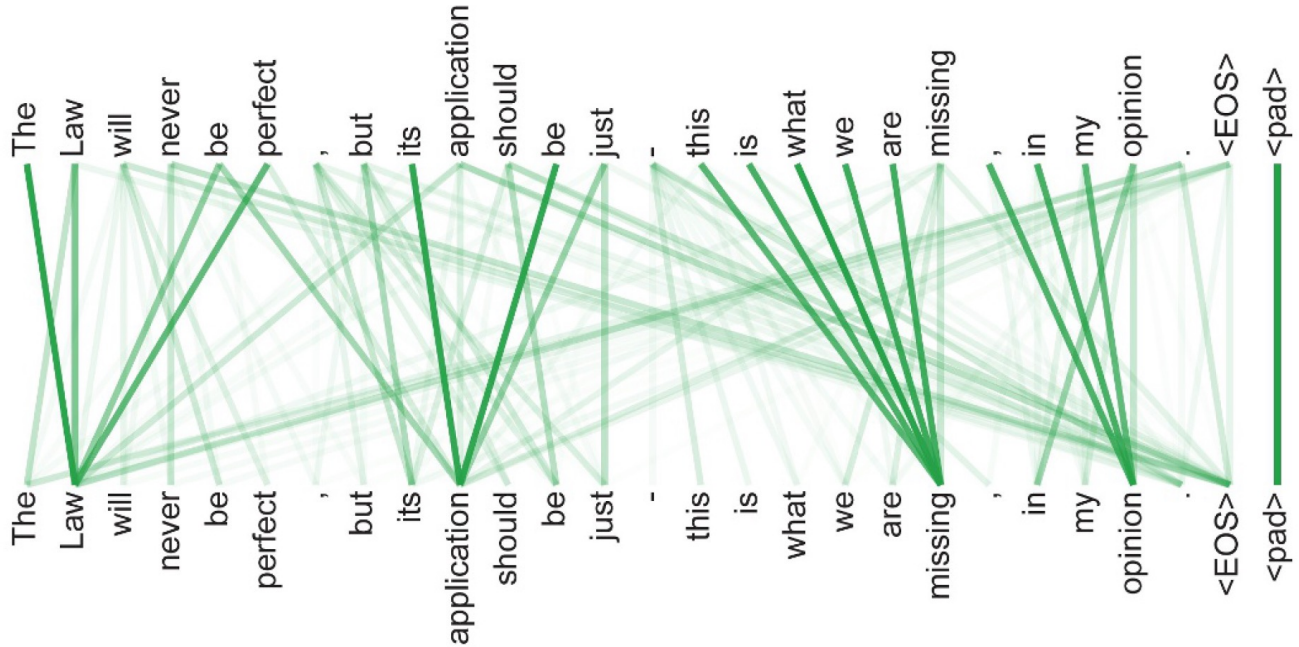
Two Different Sentences

- Particular locations that should receive more attention depend on the input sequence itself

*"I swam across the river to get to the other **bank**."*

*"I walked across the road to get cash from the **bank**."*

  - Once the network is trained, the weights for their associated inputs are generally fixed.
  - By contrast, attention uses weighting factors whose values depend on the specific input data.



Institution Behind Self-Attention

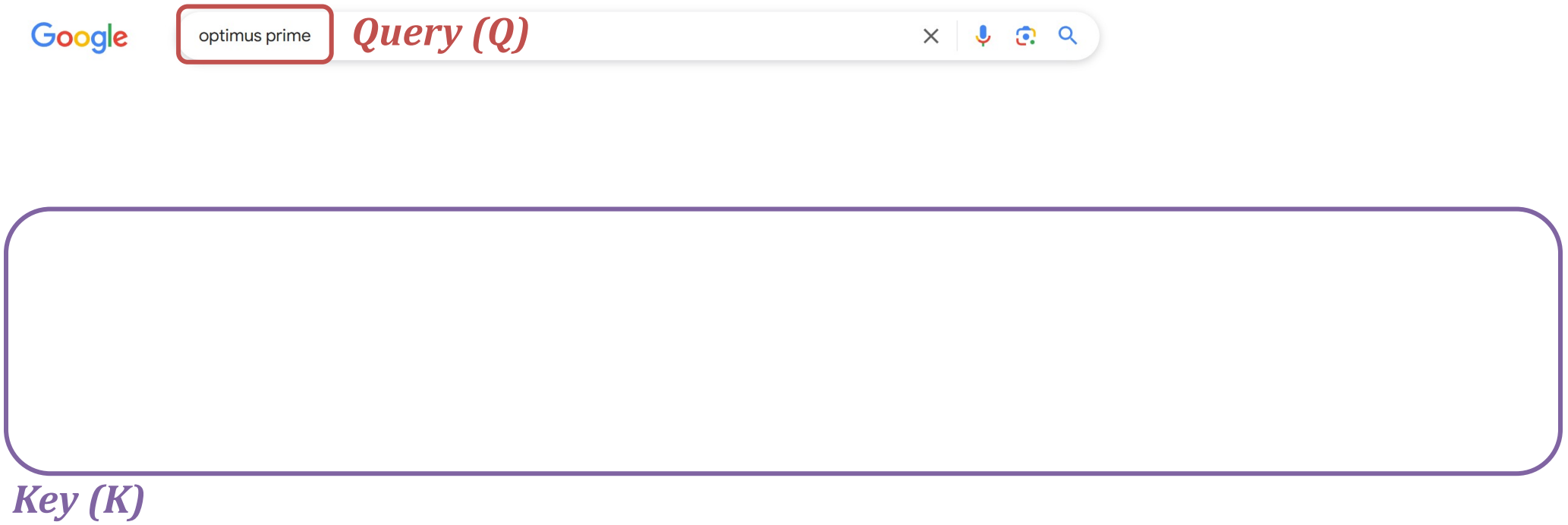
- Let's attend to the most important parts of an input

Institution Behind Self-Attention

- Our brain behaves like
  1. Identify which parts to attend to
  2. Extract the features with high attention

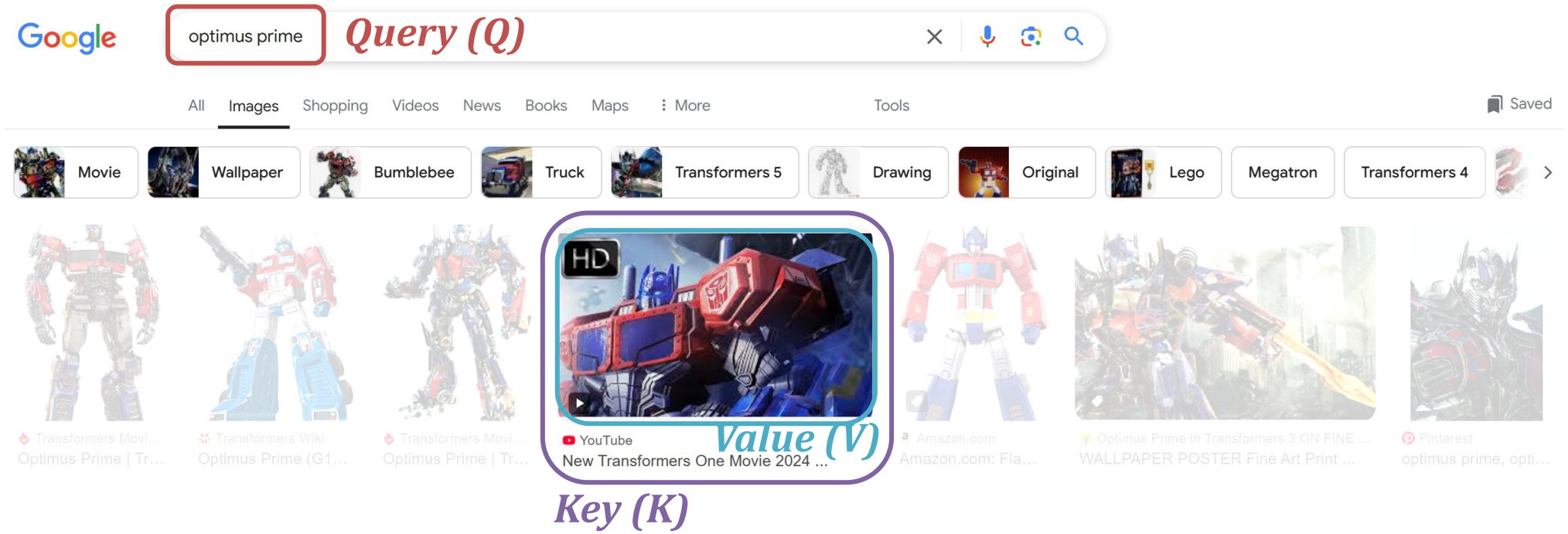
Understanding Self-Attention with Search

- Compute attention masks
  - How similar is each key to the desired query?

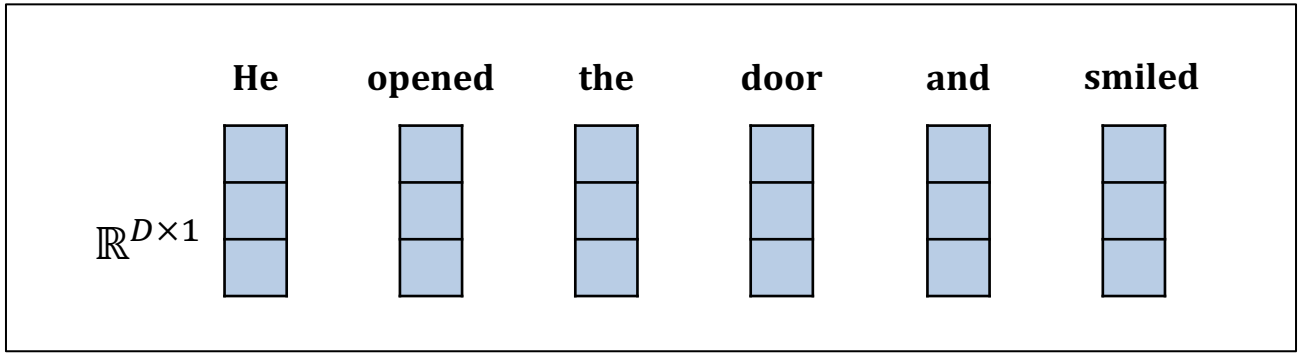


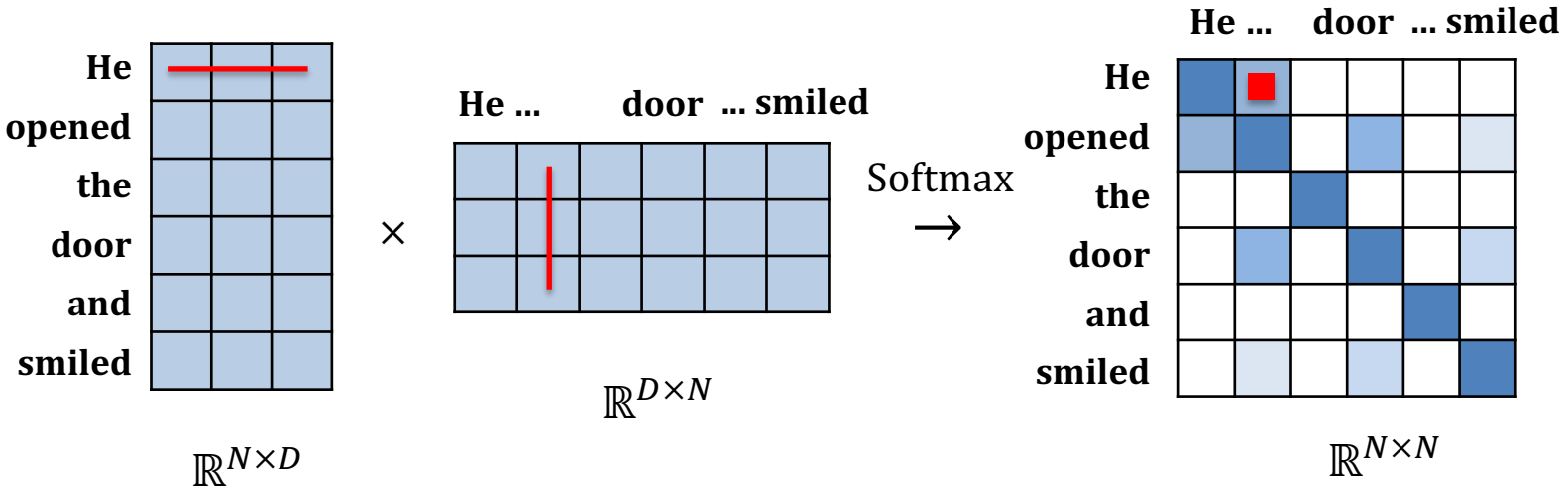
Understanding Self-Attention with Search

- Extract values based on attention
  - Return the values highest attention
  - C.f.) Hard attention vs. Soft attention

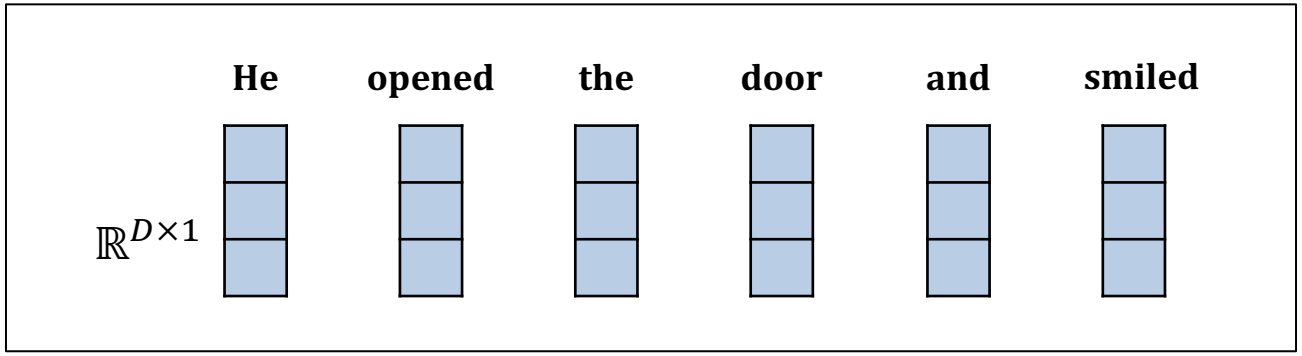


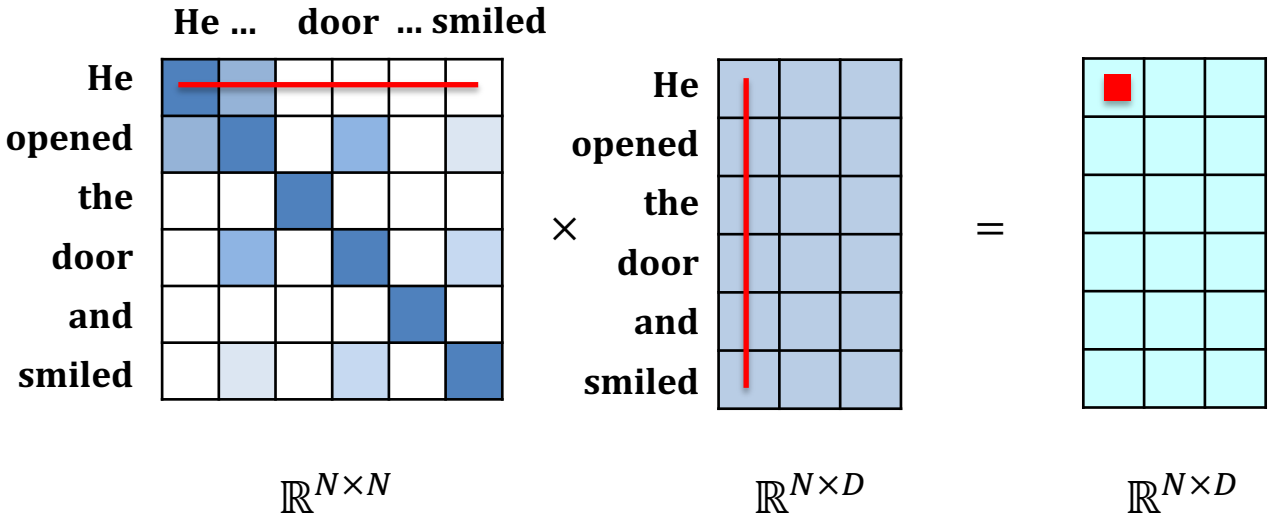
Learning Self-Attention with Neural Networks



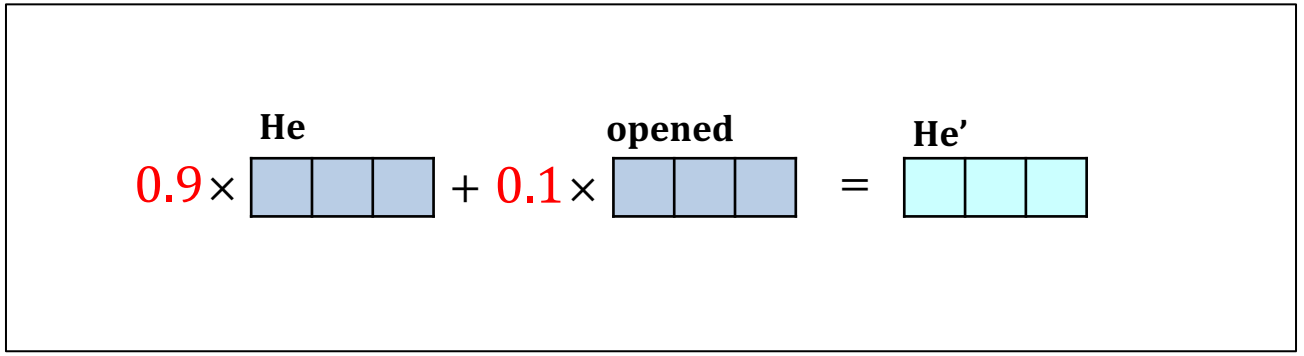


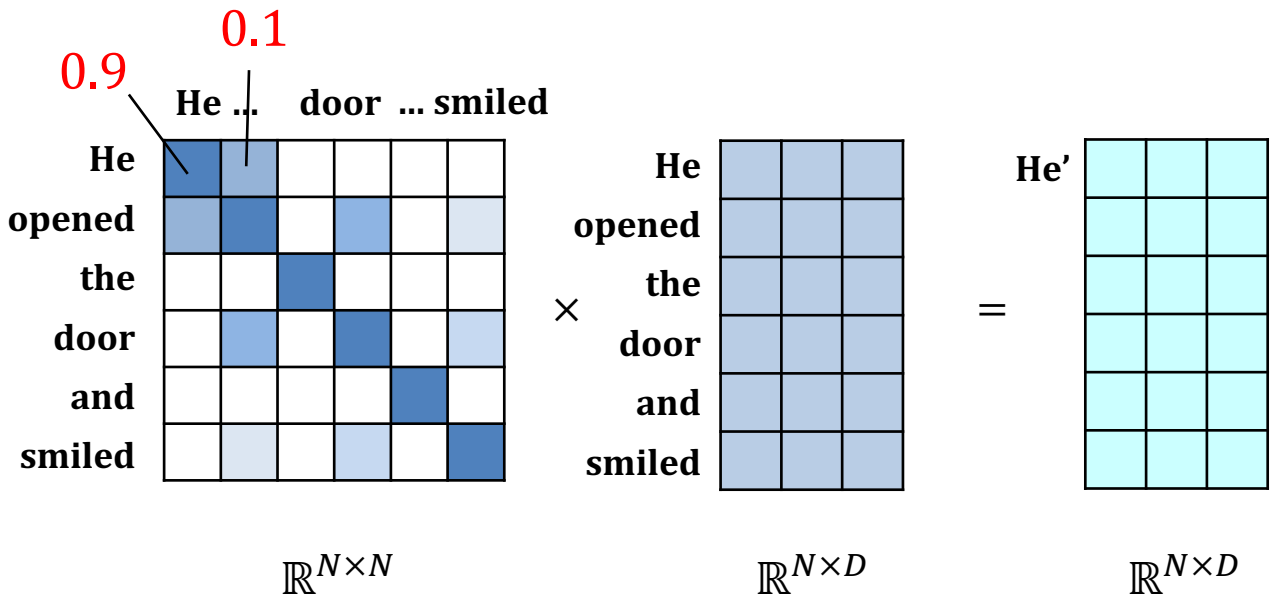
Learning Self-Attention with Neural Networks



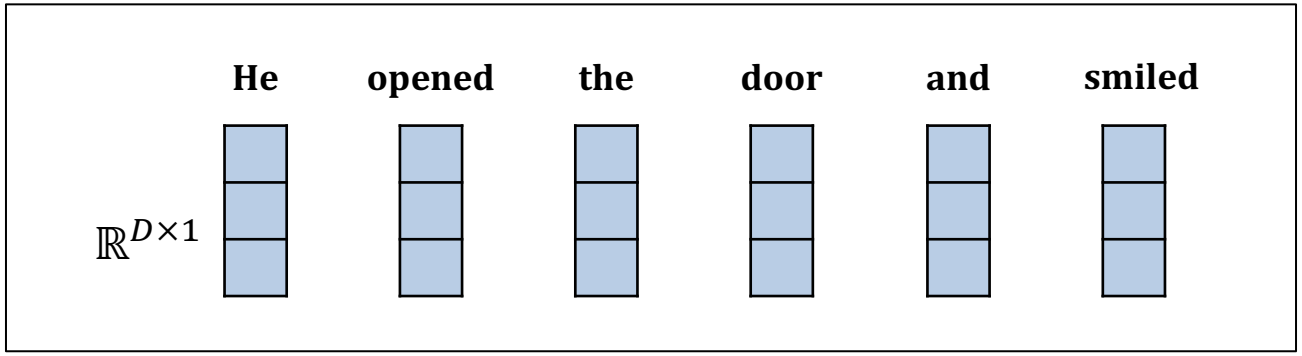


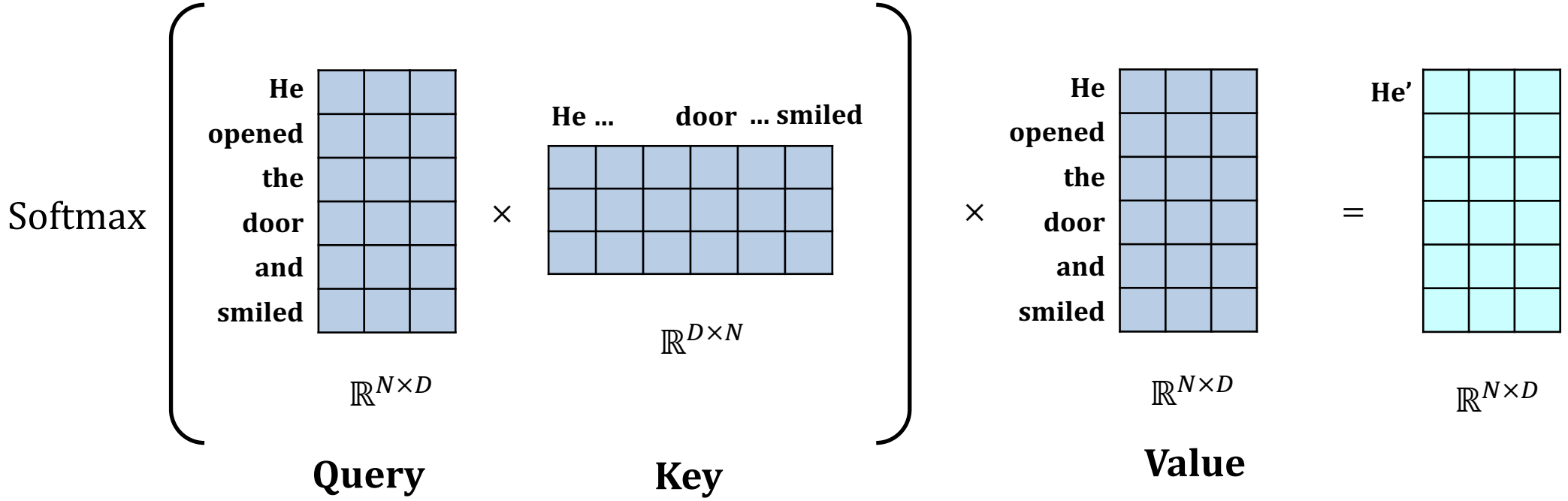
Learning Self-Attention with Neural Networks





Learning Self-Attention with Neural Networks

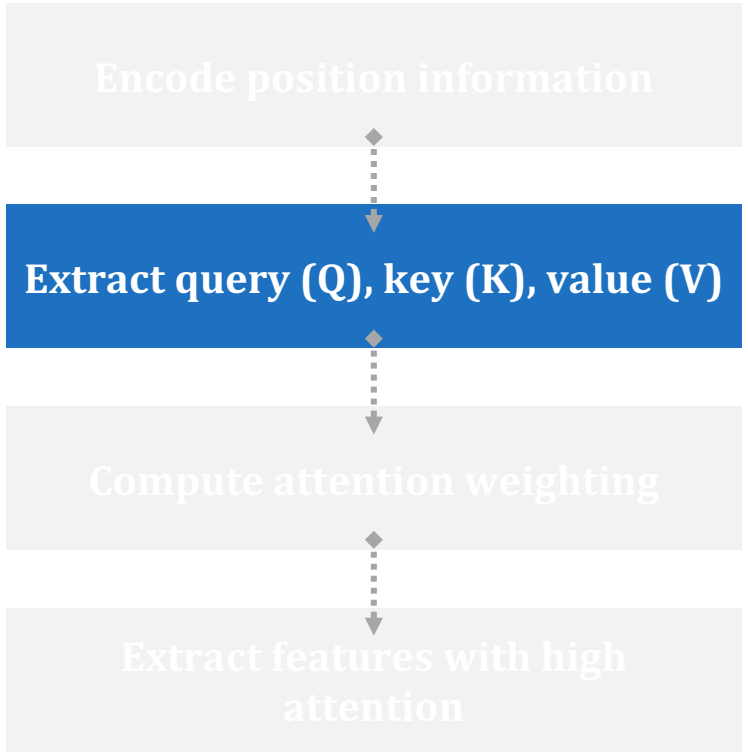


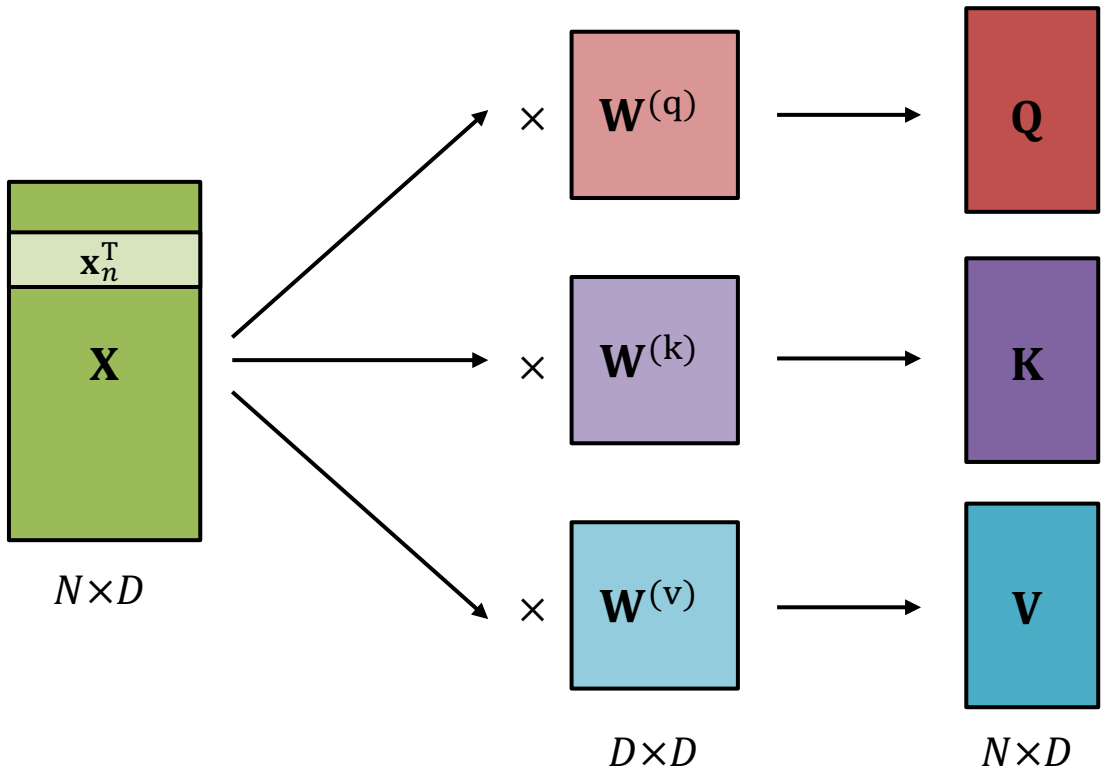


Learning Self-Attention with Neural Networks

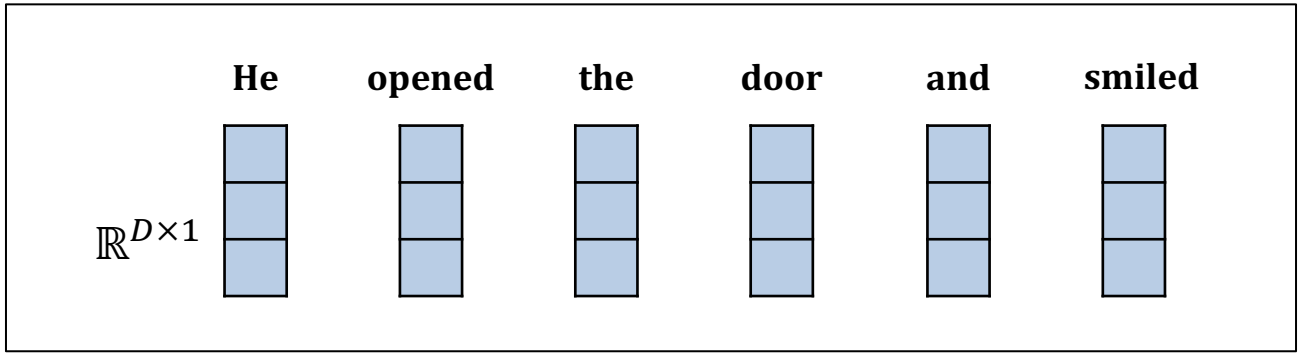
Learning Self-Attention with Neural Networks

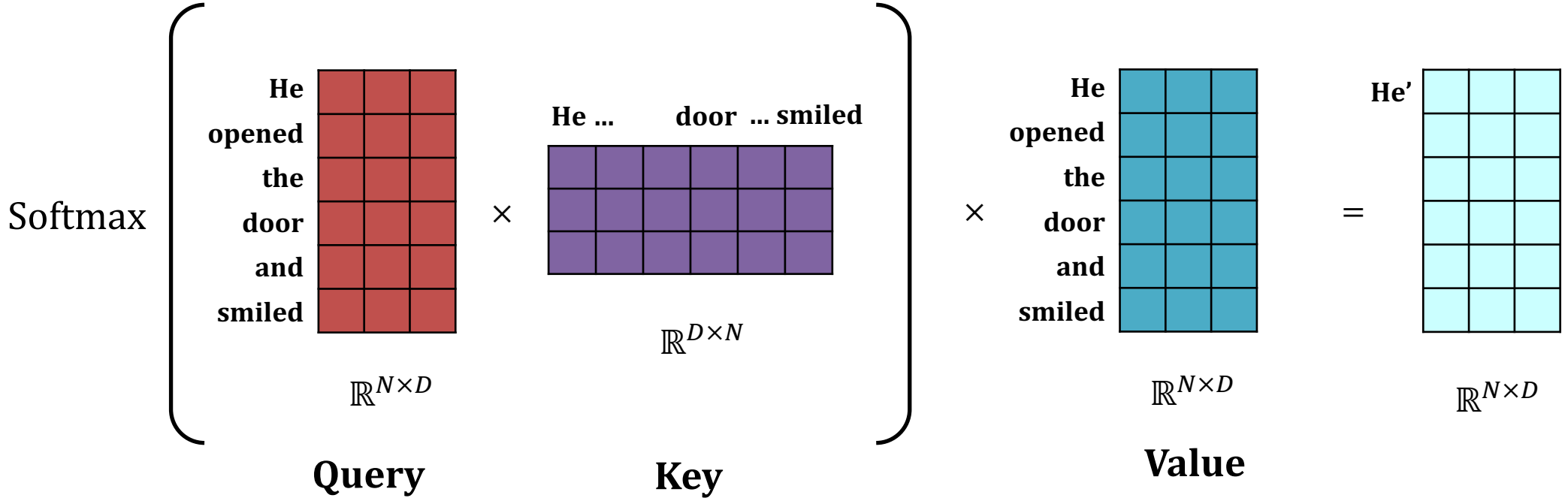
- Goal is to identify and attend to most important features in input data.





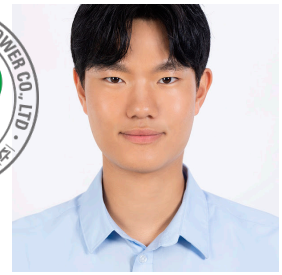
Learning Self-Attention with Neural Networks

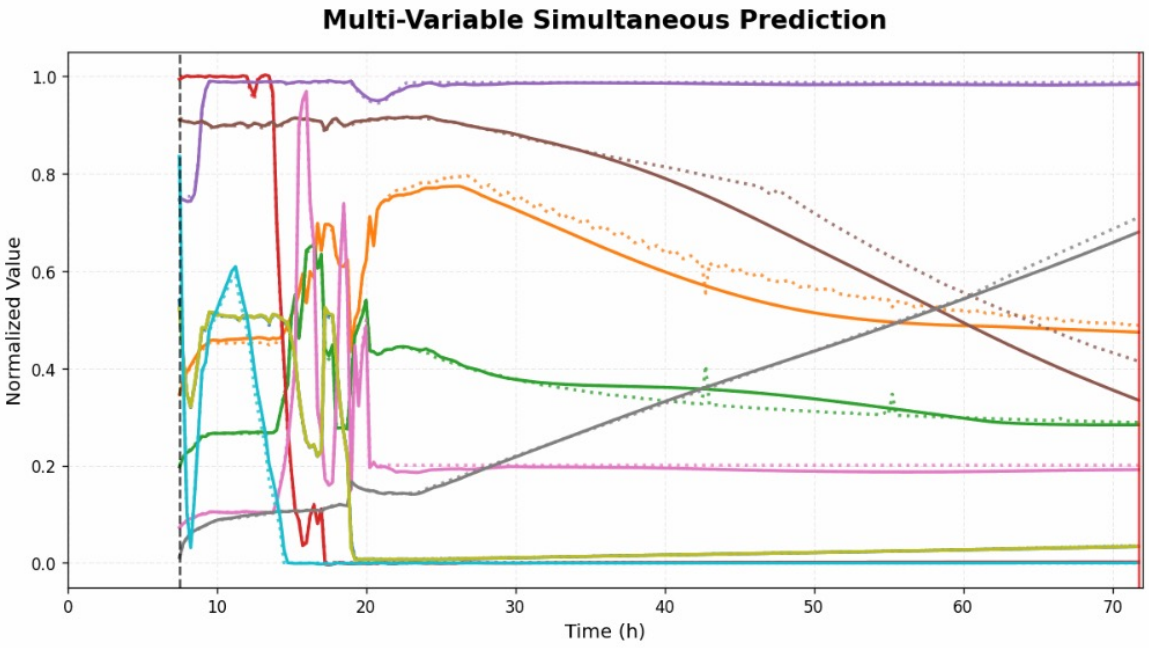


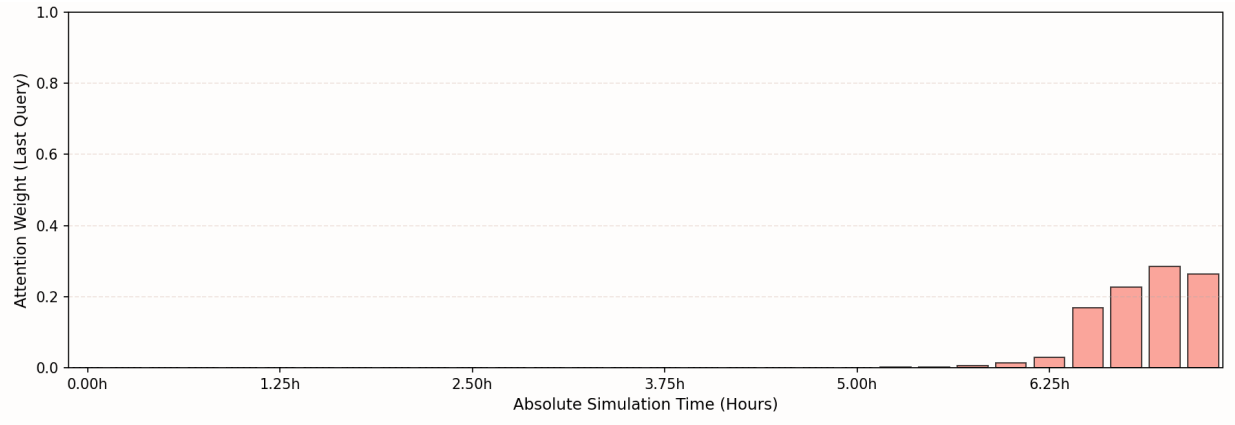


Learning Self-Attention with Neural Networks

- Example of Self-attention layer (timeseries prediction)
  - Self-attention weights reveal which past timesteps are more important

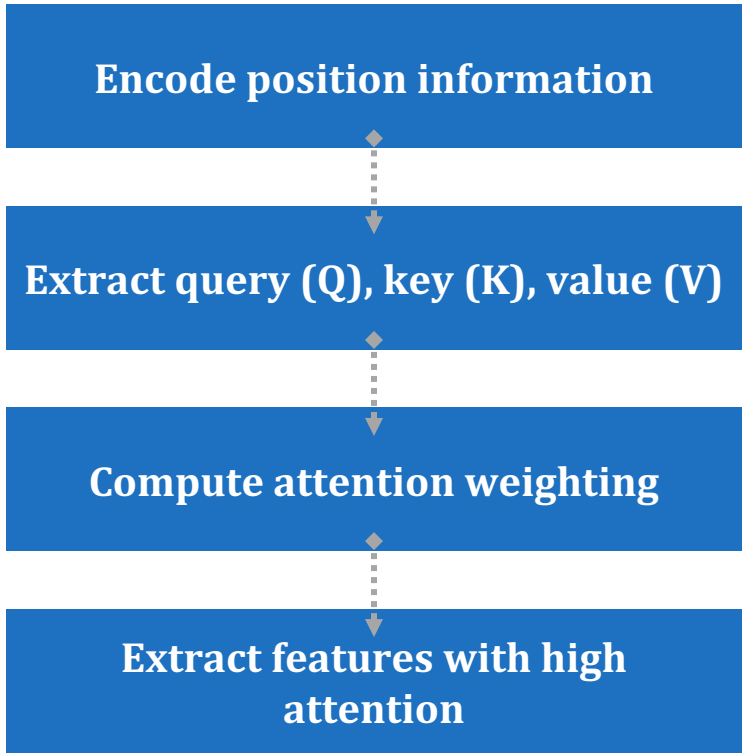






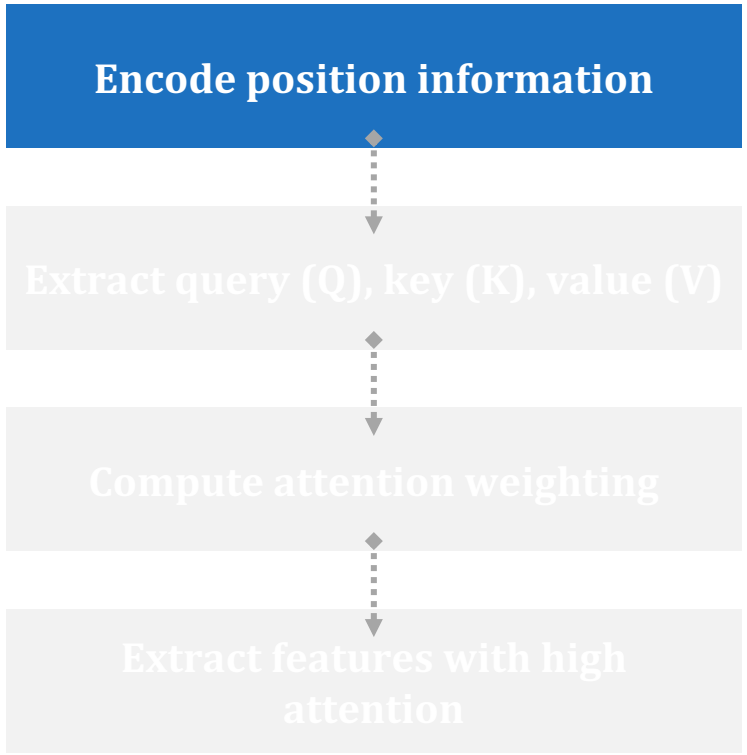
Learning Self-Attention with Neural Networks

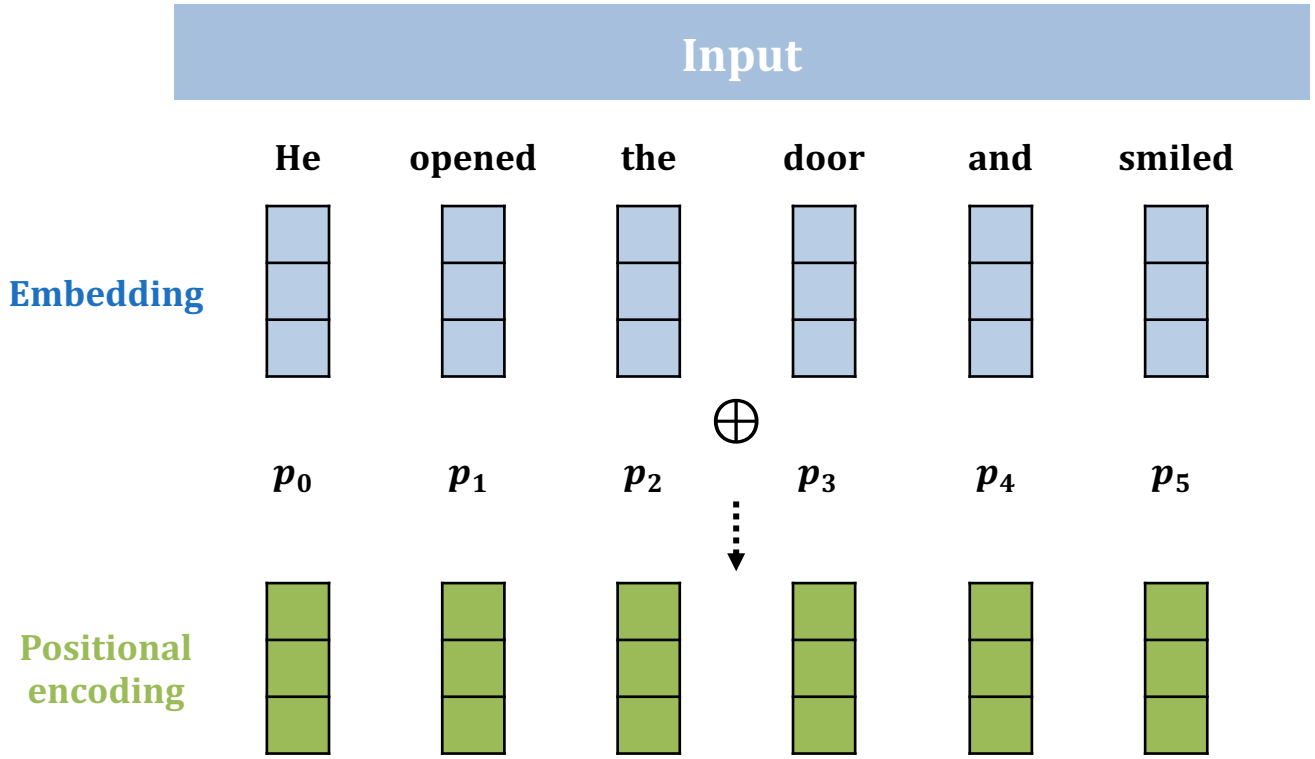
- Goal is to identify and attend to most important features in input data.



Learning Self-Attention with Neural Networks

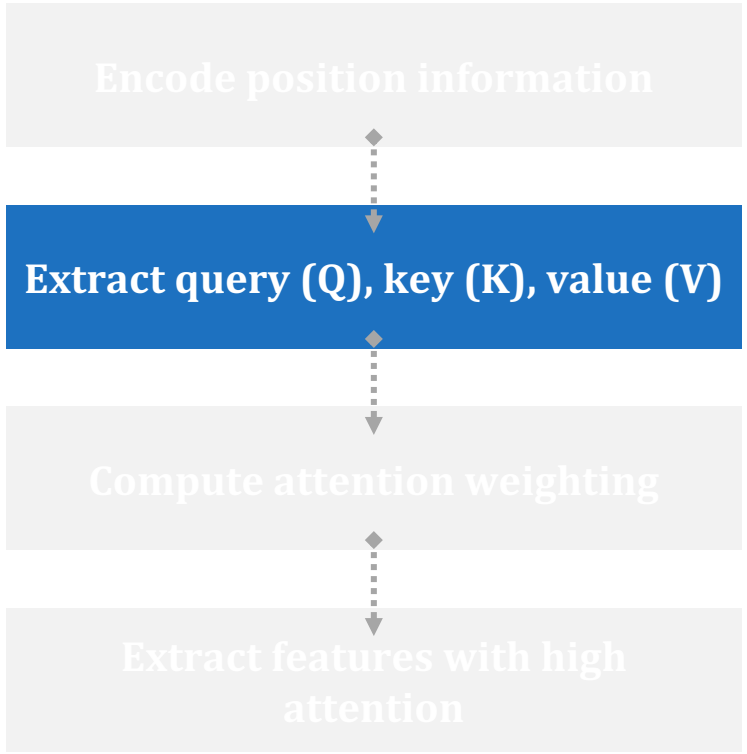
- Goal is to identify and attend to most important features in input data.

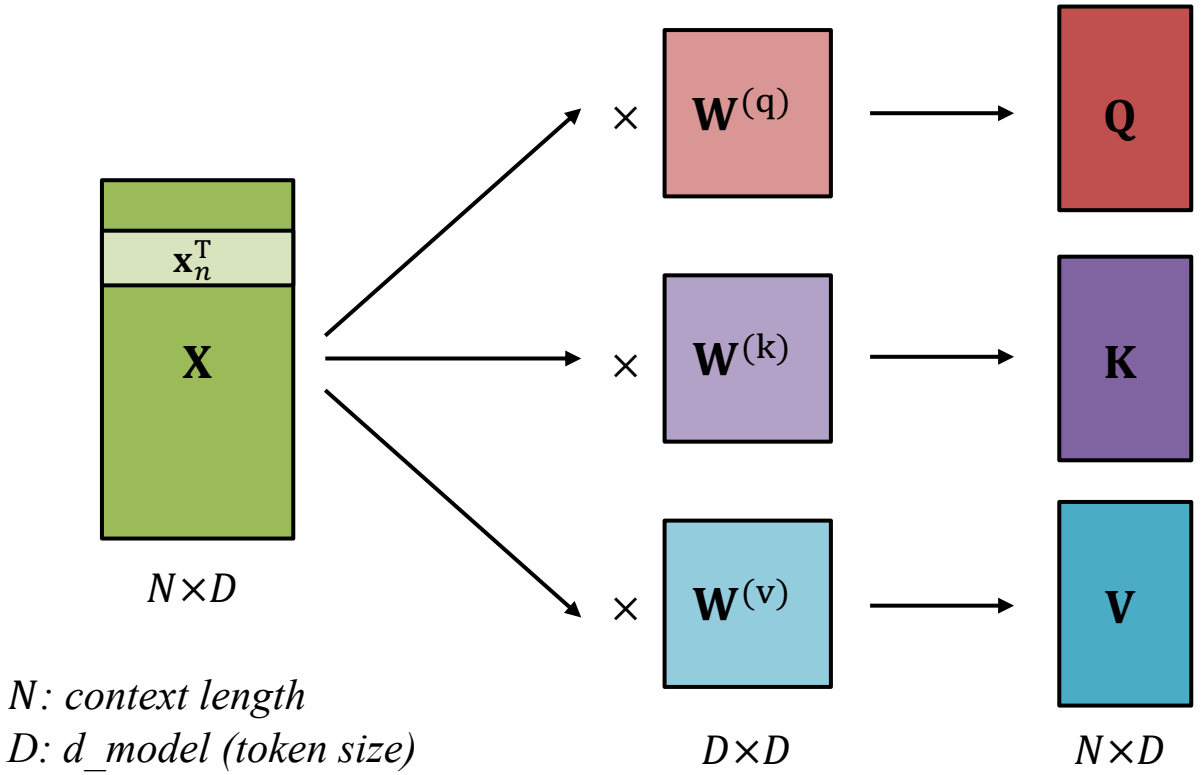




Learning Self-Attention with Neural Networks

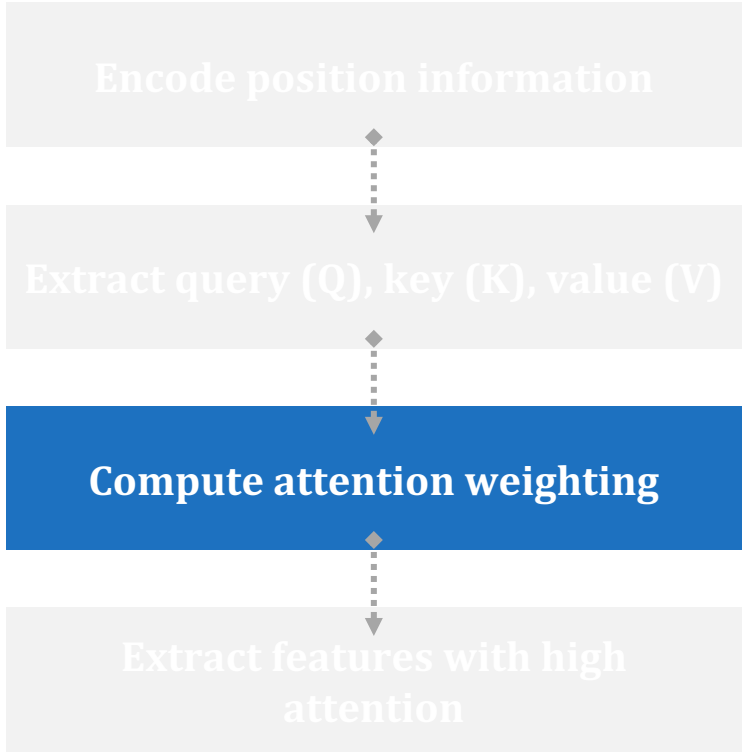
- Goal is to identify and attend to most important features in input data.

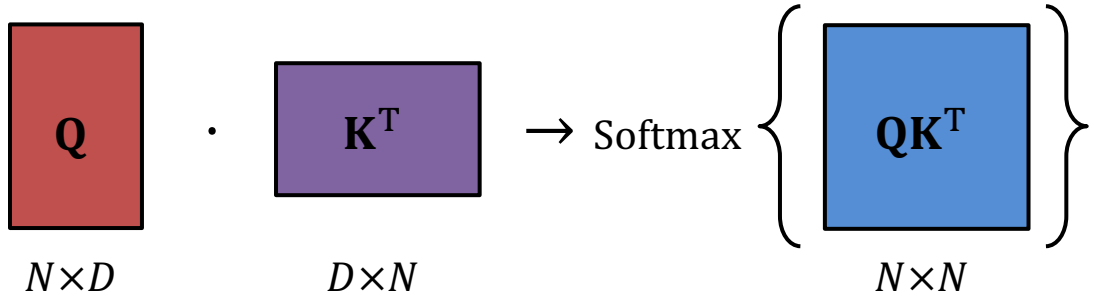




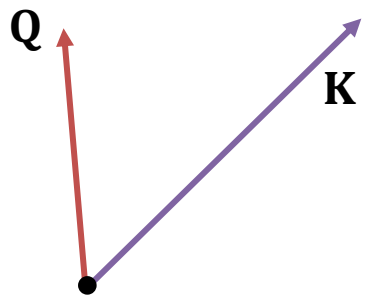
Learning Self-Attention with Neural Networks

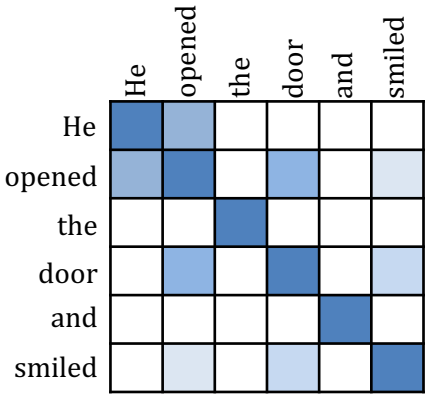
- Goal is to identify and attend to most important features in input data.





Attention score by computing similarity between Q and K





Learning Self-Attention with Neural Networks

- Goal is to identify and attend to most important features in input data.

Encode position information

↓

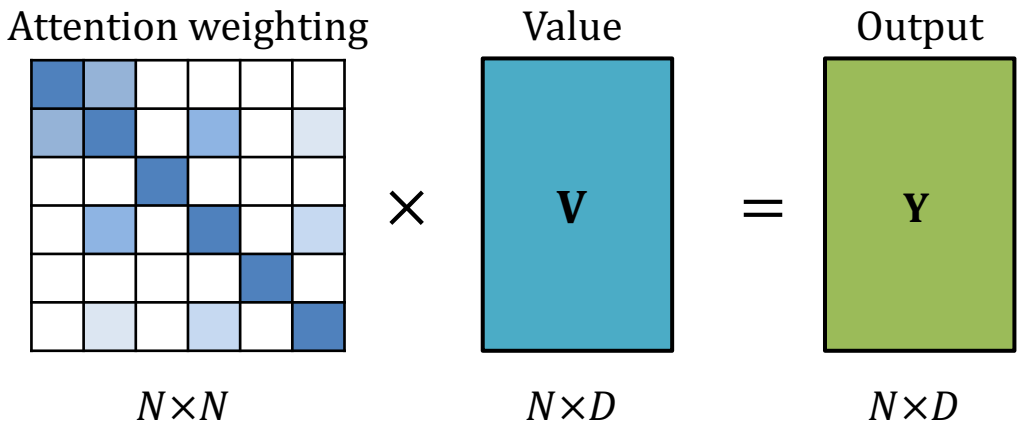
Extract query (Q), key (K), value (V)

↓

Compute attention weighting

↓

Extract features with high attention



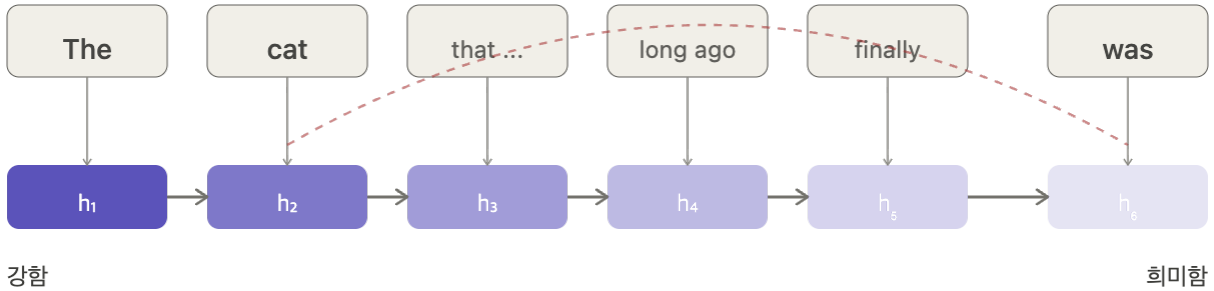
$$\mathbf{Y} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \equiv \text{Softmax}\left[\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{D_{\mathrm{k}}}}\right]\mathbf{V}$$

Last-token method

RNN: sequential hidden state

"cat" → "was" 의존성: 신호가 5단계를 거치며 약화

"The cat ... was hungry" — 정보가 단계마다 희미해진다



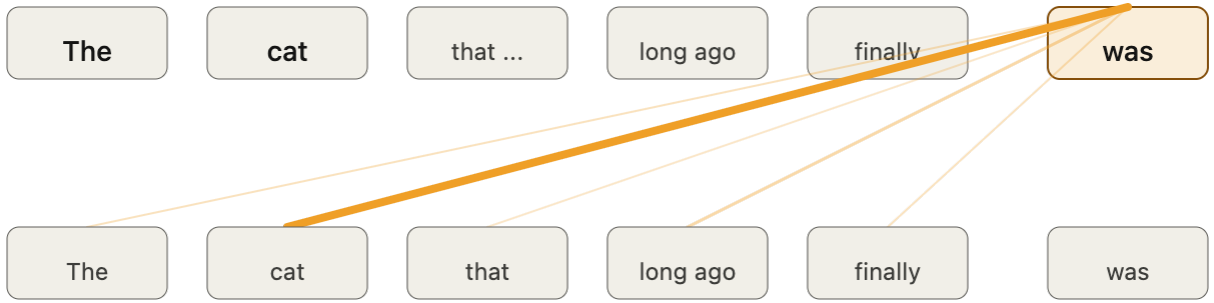
강함 희미함

병목: 모든 과거 정보가 하나의 h_t로 압축

길이가 길수록 초기 토큰이 사라진다 (vanishing gradient)

Transformer: self-attention

모든 토큰이 모든 토큰을 직접 본다 — 거리는 O(1)



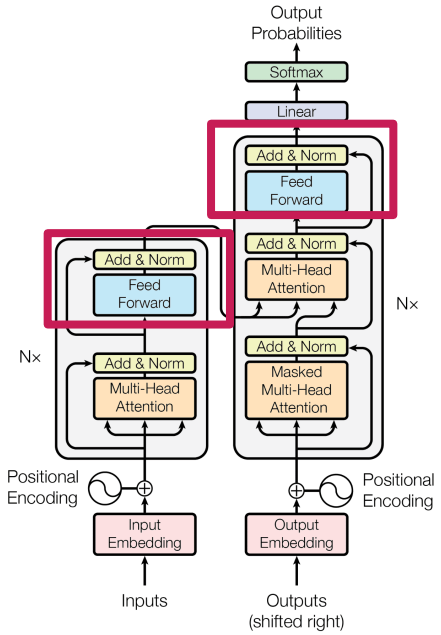
선의 굵기 = attention weight. "was"가 "cat"을 직접 강하게 참조

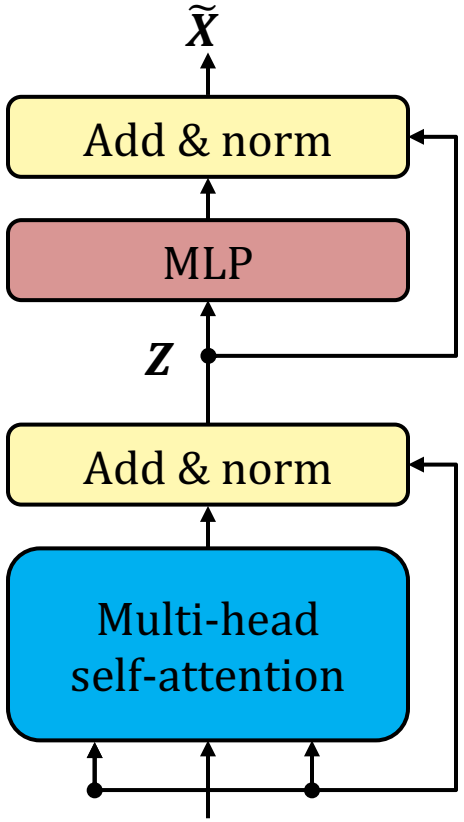
거리에 무관: 어떤 토큰 쌍이든 1-step

병렬 계산 가능, 장거리 의존성 손실 없음

Last-token method

- Residual connections that bypass the multi-head structure
- Layer normalization to improve training efficiency





$$\tilde{X} = WZ + b$$

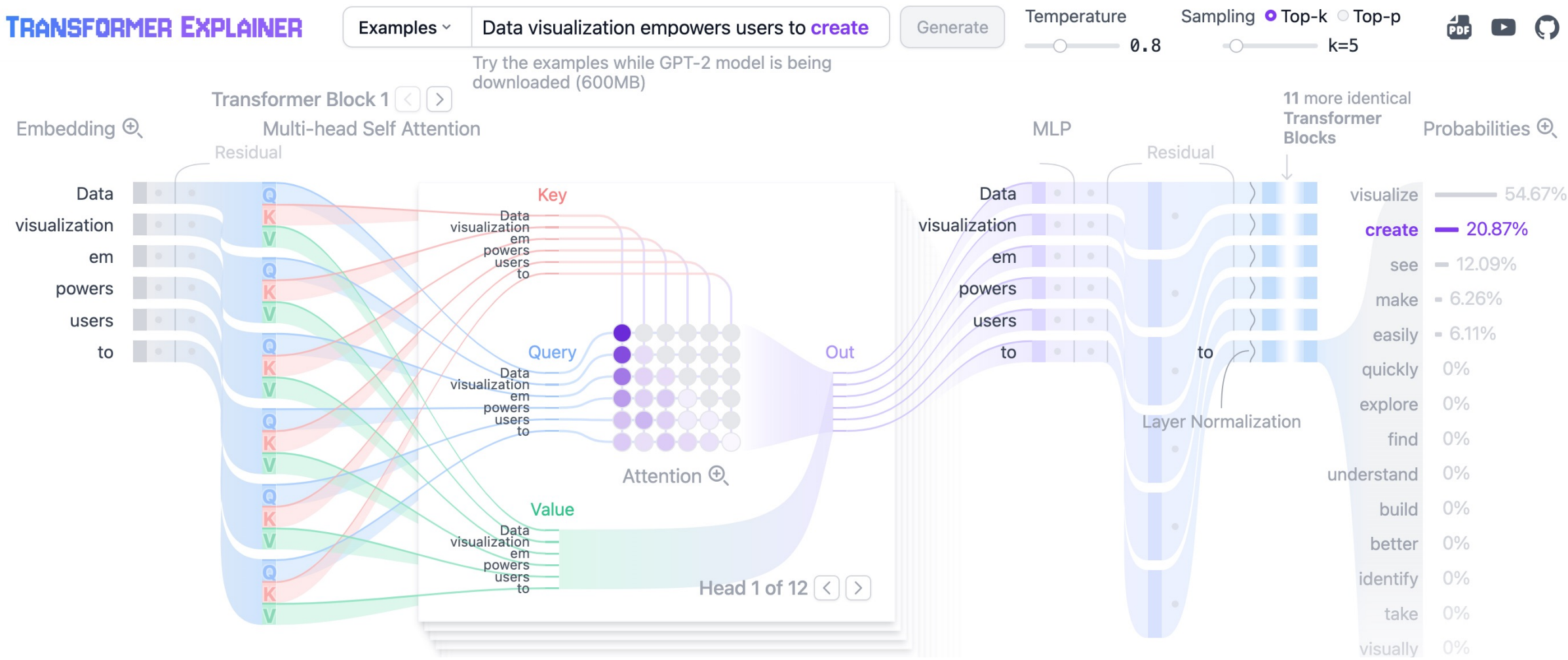
$Z \in \mathbb{R}^d$

$W \in \mathbb{R}^{V \times d}$

$\tilde{X} \in \mathbb{R}^V$

d: model size
V: number of vocabularies

Softmax and Output Probabilities

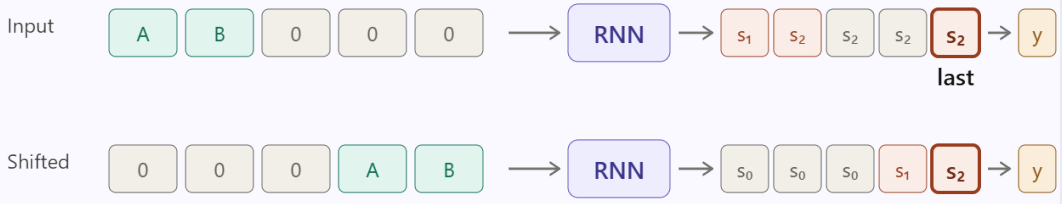


https://poloclub.github.io/transformer-explainer/

Why transformer?

RNN

Temporally equivariant → invariant via last hidden state

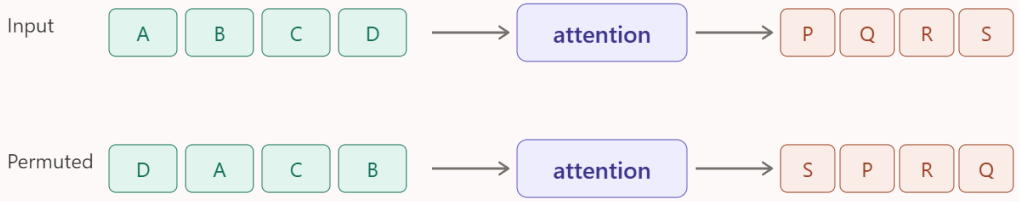


hidden states slide along time (s_0 stays idle, then s_1 s_2) → equivariant

but the final state s_2 has absorbed the same content → same y → invariant

Transformer

Permutation equivariant — treats input as a set, no order assumed



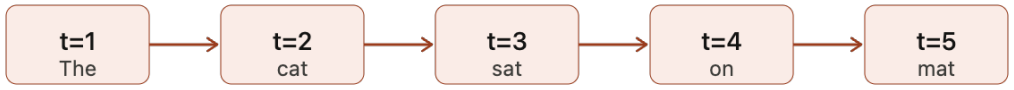
outputs permute exactly the same way as inputs → permutation equivariant

a much stronger symmetry than shift — order must be added back via positional encoding

→ for invariance: pool over tokens, or read out a special [CLS] token

RNN / LSTM — 순차 처리

시점 t의 계산은 t-1이 끝나야 시작됨



hidden state가 한 칸씩 전달 — 5단계를 차례대로

GPU 코어 1개만 일하는 중 · 나머지는 대기

Transformer — 병렬 처리

모든 토큰이 동시에 서로를 참조 (self-attention)

한 번의 거대한 행렬 곱(QK^T)으로 모든 관계를 동시에

GPU 코어 수천 개가 동시에 일하는 중

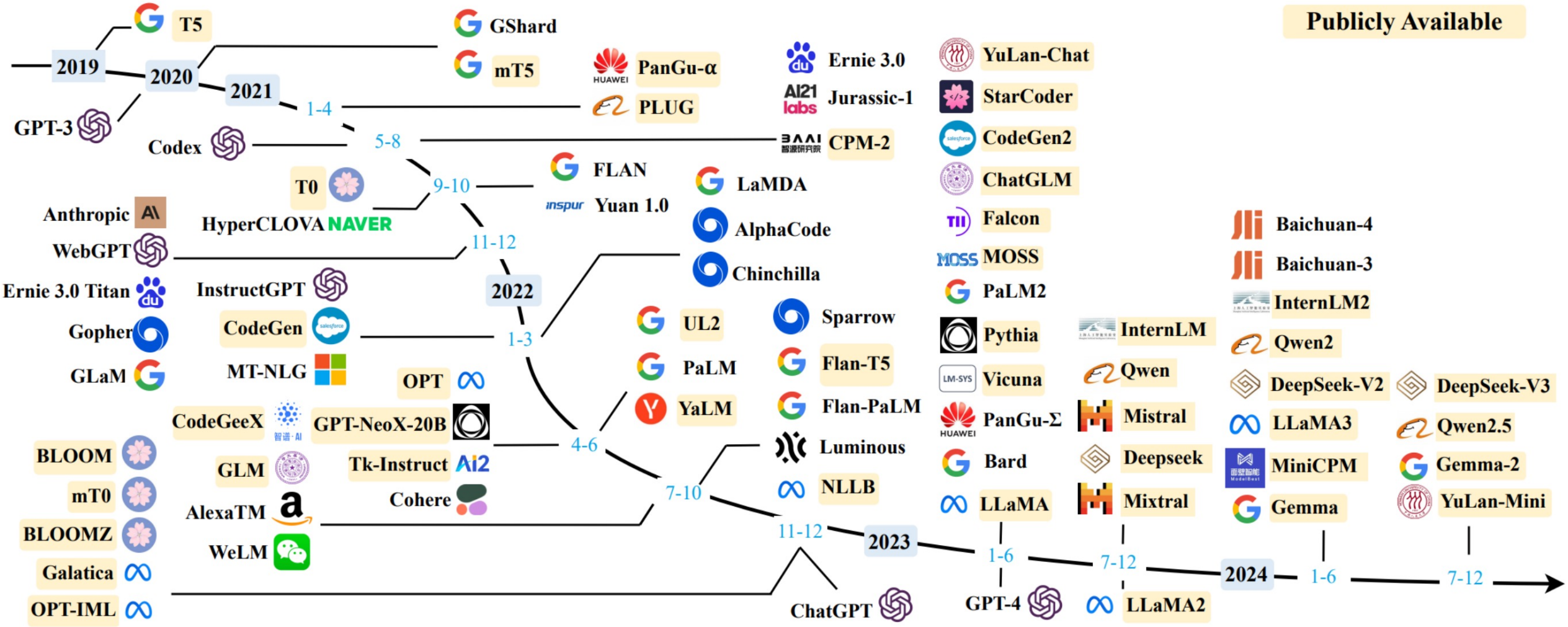
결론

병렬화	짧은 gradient 경로	현실적 학습 시간
GPU 활용률 ↑↑	깊게 쌓아도 학습 안정	100B+ 모델도 가능

$d_{model}: > 10K$

$context\ length > 10\text{K}$

Transformer 이후 언어모델의 발전흐름



- 이 모든 것들이 다 Transformer 기반 언어모델

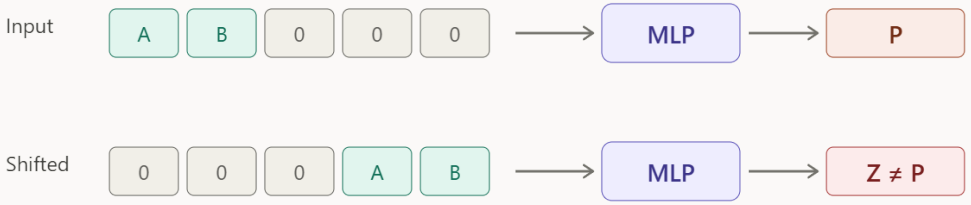
Things to study

- Tokenization
- Word embedding
- Positional encoding
- Cross-attention
- Masked-attention
- Multi-head attention

Overview

MLP

Neither equivariant nor invariant

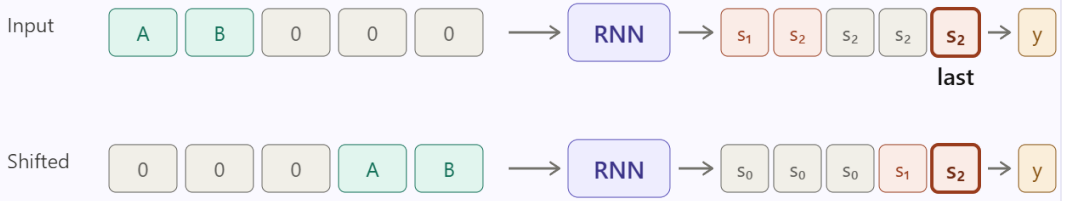


Each input slot has its own weights — moving the pattern breaks everything

f(shift(x)) ≠ f(x) and f(shift(x)) ≠ shift(f(x))

RNN

Temporally equivariant → invariant via last hidden state

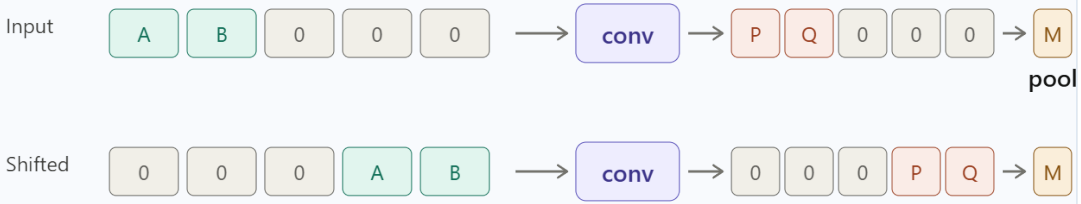


hidden states slide along time (s_0 stays idle, then s_1 s_2) → equivariant

but the final state s_2 has absorbed the same content → same y → invariant

CNN

Spatially equivariant → invariant via global pooling

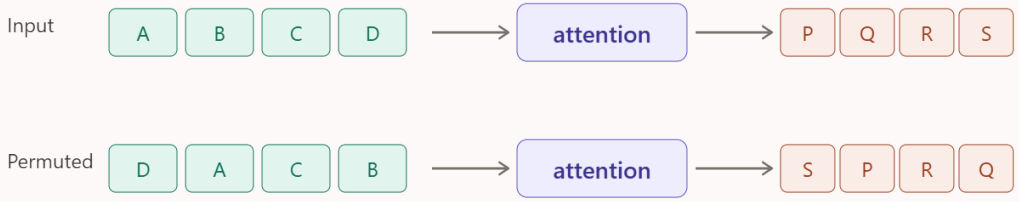


conv output PQ slides with the input → equivariant: f(shift(x)) = shift(f(x))

global pooling collapses all positions → same M either way → invariant

Transformer

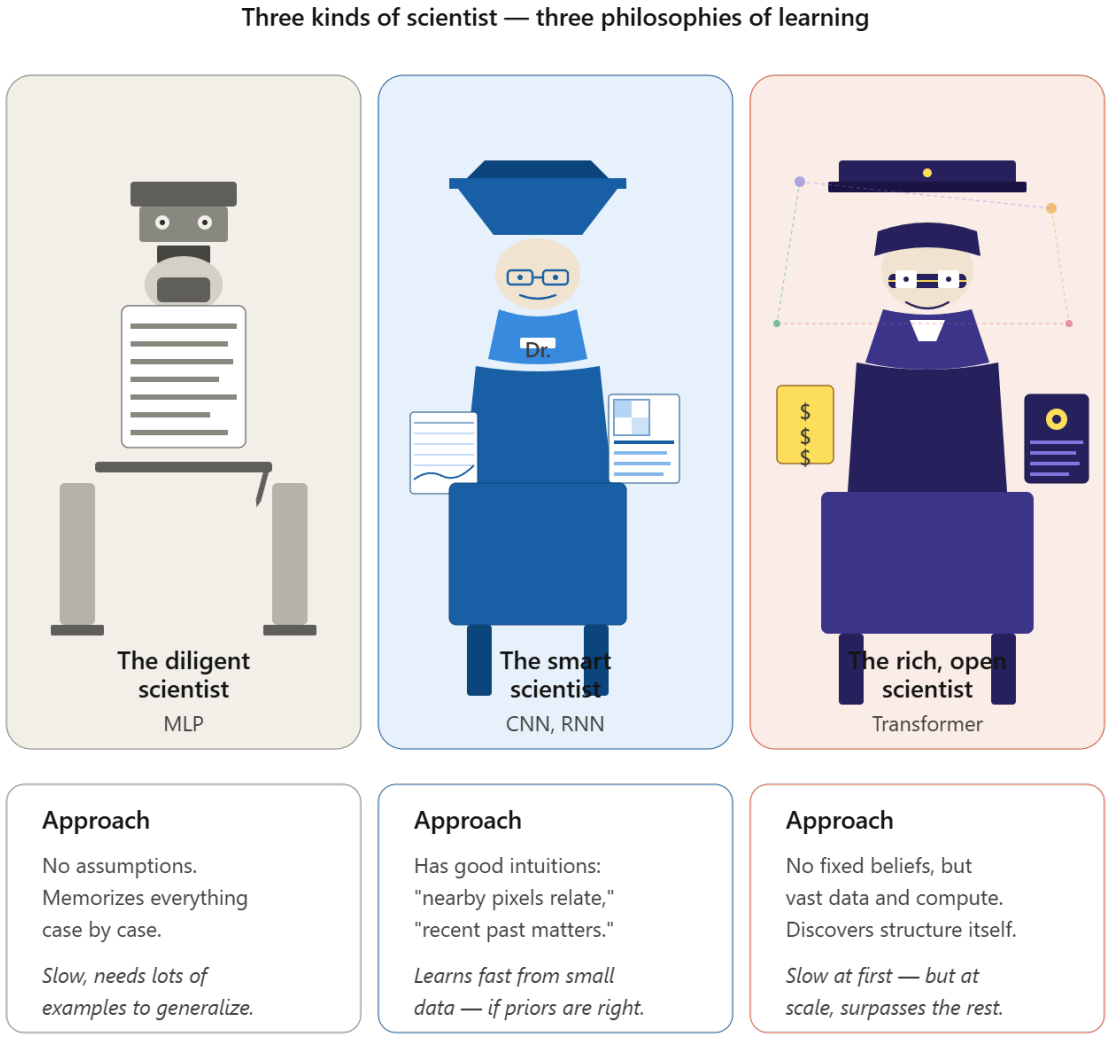
Permutation equivariant — treats input as a set, no order assumed



outputs permute exactly the same way as inputs → permutation equivariant

a much stronger symmetry than shift — order must be added back via positional encoding

→ for invariance: pool over tokens, or read out a special [CLS] token



Thank You.
Any Questions?

jgjeon41@postech.ac.kr