ResNet & Transformer & Diffusion Policy

1. ResNet

1. vs CNN
   ↳ gradient vanishing.exploding
   degradation : 깊이 ↑ 성능 ↓

2. $H(x)$ 대신 $H(x) = F(x) + x$를 학습

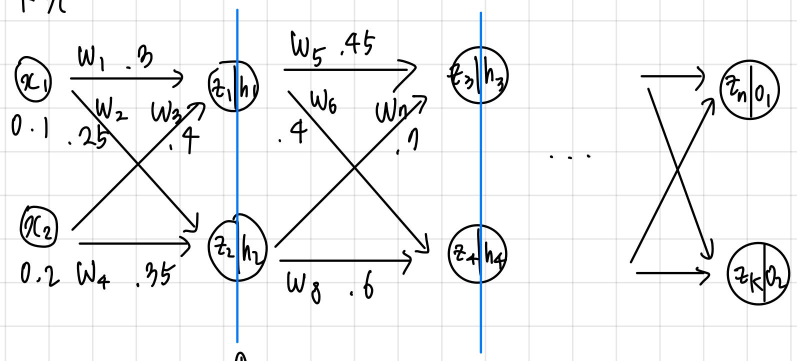
   shortcut
   (skip-connection)
   : 가중치 X, 복잡도 X

   ($F(x), x$ 차원 같지 X : $H(x) = F(x) + \underline{\underline{W}} x$)

   down-sampling

   mapping 방식
   - identity : zero-padding
   - projection : 1x1 conv ✓

3. "$+x$"



$$h = \mathrm{ReLU}\left\{\left(x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) + \left(z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right)\right\} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

element-wise addition

$$\mathrm{ReLU}\left\{\left(x_2 = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}\right) + \left(z_2 = \begin{bmatrix} z_3 \\ z_4 \end{bmatrix}\right)\right\} = \begin{bmatrix} h_3 \\ h_4 \end{bmatrix}$$

→ 장점?

1. 해당 레이어 필요 X → 입력으로 유지

↳

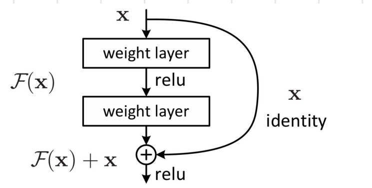


Figure 2. Residual learning: a building block.

항등매핑: $H(x) = x + F(x) \approx x$

2. 역전파 gradient 계산 학습 안정성 (gradient vanishing, exploding X)

↳ $\dfrac{\partial L}{\partial x} = \dfrac{\partial L}{\partial y} \cdot \dfrac{\partial y}{\partial x}$

Plain

$y = F(x)$

$\rightarrow \dfrac{\partial F(x)}{x}$

Residual

$y = F(x) + x$

$\rightarrow \dfrac{\partial \{F(x) + x\}}{\partial x} = \dfrac{\partial F(x)}{\partial x} + I$

2. Transformer (Attention is all you need)

1. vs RNN류

recurrent

↳ 순차 의존성 : $h_t = f(h_{t-1}, x_t)$

병렬화 X

2. Encoder & Decoder (with formula)

2.1 multi-head attention

입력 seq

↓

Encoder: (Encoder self-attention

FFN : $W_2 \cdot \mathrm{ReLU}(xW_1 + b_1) + b_2$

↓

$z = (z_1 \cdots z_n)$

↓

Decoder: (Decoder self-attention (Masked Multi-head attention)

Encoder-Decoder attention

FFN

↓

$y = (y_1 \cdots y_n)$

↳ Residual Add & Norm

" $+x$ "

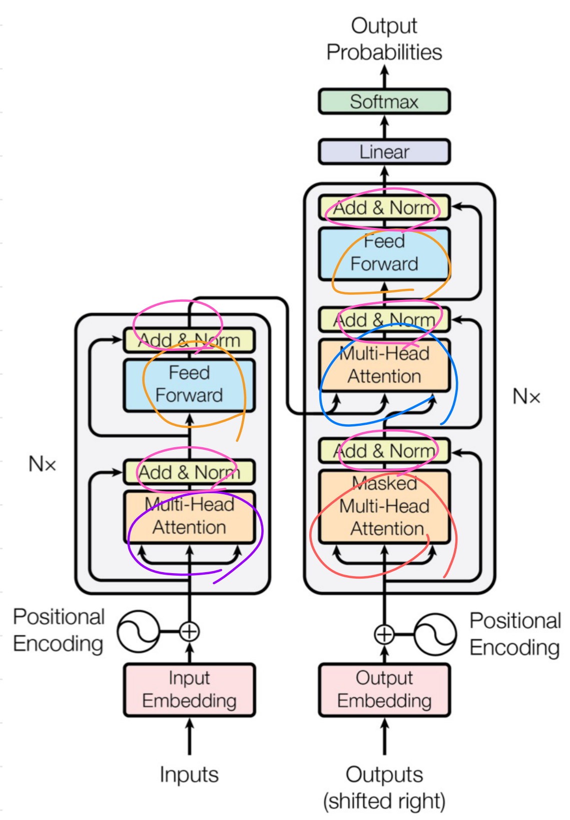


Figure 1: The Transformer - model architecture.

2.1 attention (Q,K,V)

1) X : token embedding ($T \times d_{model}$ dim) + Positional Encoding

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

→

pos 0 1 2 3

ex) 나는 오늘 밥을 먹었다.

$d_{model} = 10$ (토큰의 임베딩 차원)

→ "오늘" : token embedding $= [e_1, \cdots, e_{10}]$

+

position embedding $= [P_1, \cdots, P_{10}] = [\sin(1/10000^{\frac{0}{10}}), \cos(1/10000^{\frac{0}{10}}), \sin(1/10000^{\frac{1}{10}}), \cos(1/10000^{\frac{1}{10}}), \cdots]$

2) $Q = X \cdot W^Q$
$K = X \cdot W^K$
$V = X \cdot W^V$ → $\frac{d_{model}}{h}$ (head)

3)

$$\Sigma \left\{ \mathrm{Attention}(Q,K,V) = \overbrace{\mathrm{softmax}(\underbrace{\frac{QK^T}{\sqrt{d_k}}}_{\text{attention score } S})}^{\text{weight}} V \right\}$$

= weighted-sum (head 개수만큼) → Concatenate → Linear

3. Diffusion Policy

1. training : DDPM
   testing : DDIM
   + Policy

↳ <training>

샘플 행동 x_0 → 노이즈 $x_k = \sqrt{\bar{\alpha}_k}\, x_0 + \underbrace{\sqrt{1-\bar{\alpha}_k}}_{=\sigma_k}\, \varepsilon$ → $L = \| \varepsilon - \varepsilon_\theta(x_k, k, c) \|^2$ 최적화 (gradient)

k : k_{train}

$\bar{\alpha}_k$: hyper parameter

⇕

$$-\frac{1}{\sigma_k}\varepsilon_\theta(x_k, k, c) = \nabla_{x_k} \log p(x_k \mid c)$$ 최적화 (학습 시 기울기 최적화)

<testing> : Langevin 동역학 (Denoising)

: $x_{k-1} = x_k + \underbrace{\alpha_k}_{\text{gradient 적용 정도}} \cdot S_\theta(x_k, k, c) \left(+ \underbrace{\sigma_k \xi_k}_{\text{K 마다 붙는 gaussian noise}} \right)$

$k = k_{infer}$

: denoising 결과가 평균이므로 다양한 해를 얻기위해

2. Explicit / Implicit / Diffusion policy

1) Explicit

관측 O $\xrightarrow[F_\theta(O)]{}$ $\hat{a}$: 바로 행동 예측

2) Implicit

o , a $\xrightarrow[E_\theta(o,a)]{}$ arg min E → $\hat{a}$

3) Diffusion policy

O → $\varepsilon_\theta(O, a)$ → ∇E & k_{infer} → $\hat{a}$

3. 다중모달 행동분포의 표현 O
고차원 출력 공간
안정적 학습

4. ① Closed-loop action seq (receding)

→ 예측 길이 : T_p

사용 길이

T_a

↑

관찰 : 가장 최근 horizon

= T_o

② Visual conditioning

→ O_t는 C 로만 하고 출력으로 X (Visual Encoder 사용 : ResNet18)

③ CNN vs transformer

↓ ↓

over-smoothing "토큰화"