Trajectory Transformer

윤승제

Trajectory transformer

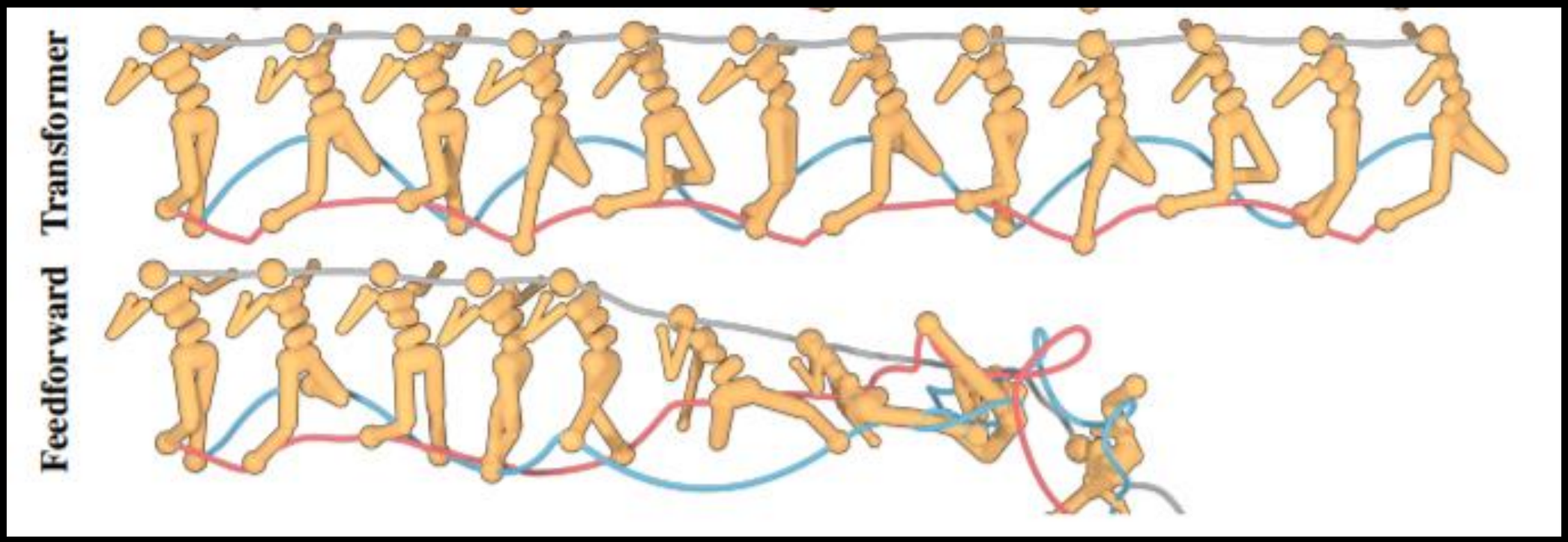
Offline Reinforcement Learning as One Big Sequence Modeling Problem

Offline Reinforcement Learning as One Big Sequence Modeling Problem

Michael Janner **Qiyang Li** **Sergey Levine**
University of California at Berkeley
{janner, qcli}@berkeley.edu svlevine@eecs.berkeley.edu

요약

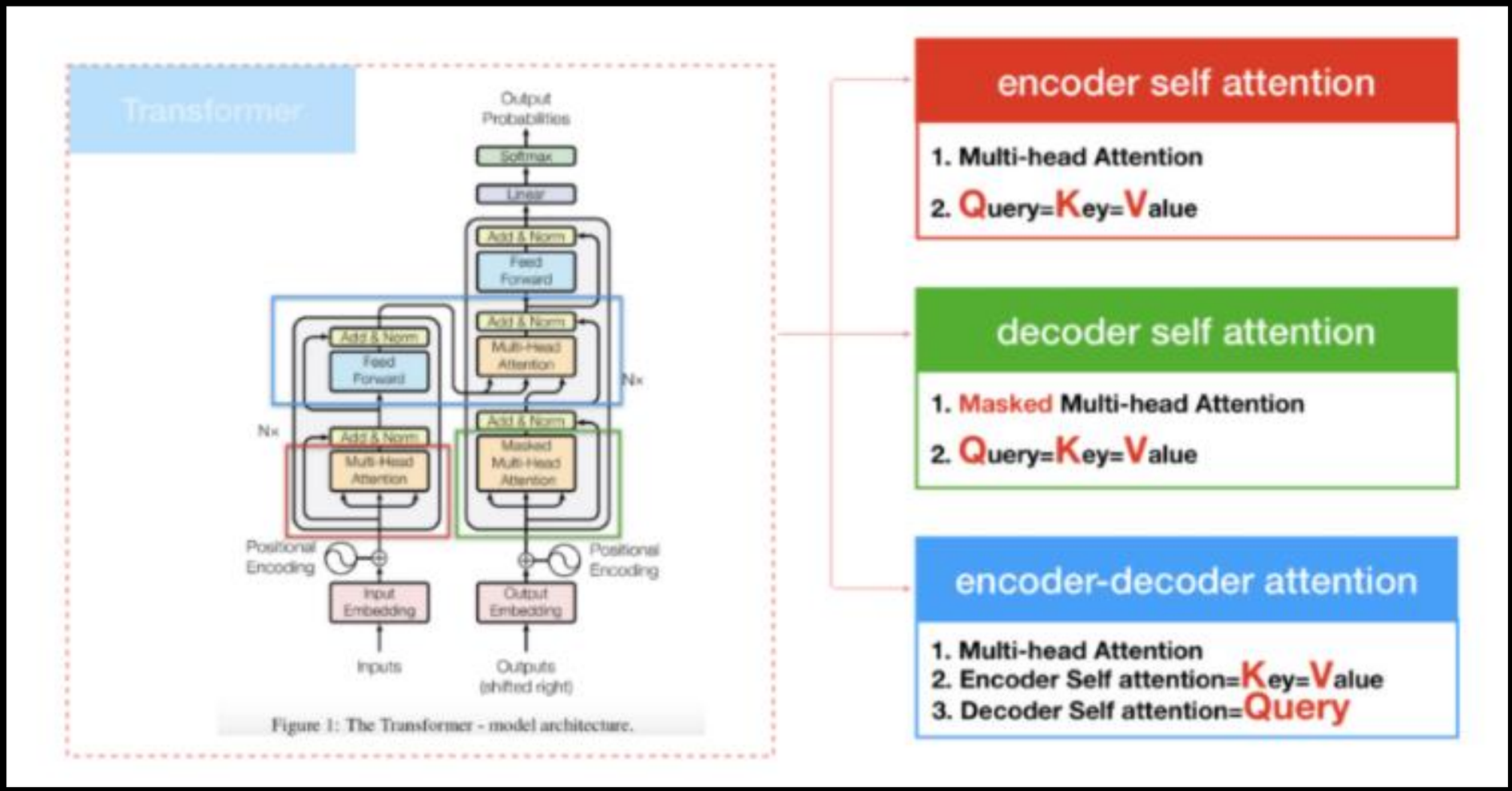
- 특징
  - One step prediction + factorize 하는 MDP 기반의 RL 대신, Long term horizon prediction 문제로 RL을 접근
  - NLP 문제에서 LSTM 보다 좋은 성능을 낸 Transformer를 적용한 RL
  - State, action, reward의 시퀀스를 예측해서 총 reward가 높게 나오도록 planning
  - 기존 Offline RL에 썼던 uncertainty estimation, Q value regularization 등이 필요 x
  - Decision Transformer와 하루 차이로 나와서 주목을 덜 받음



Transformer application

Self Attention

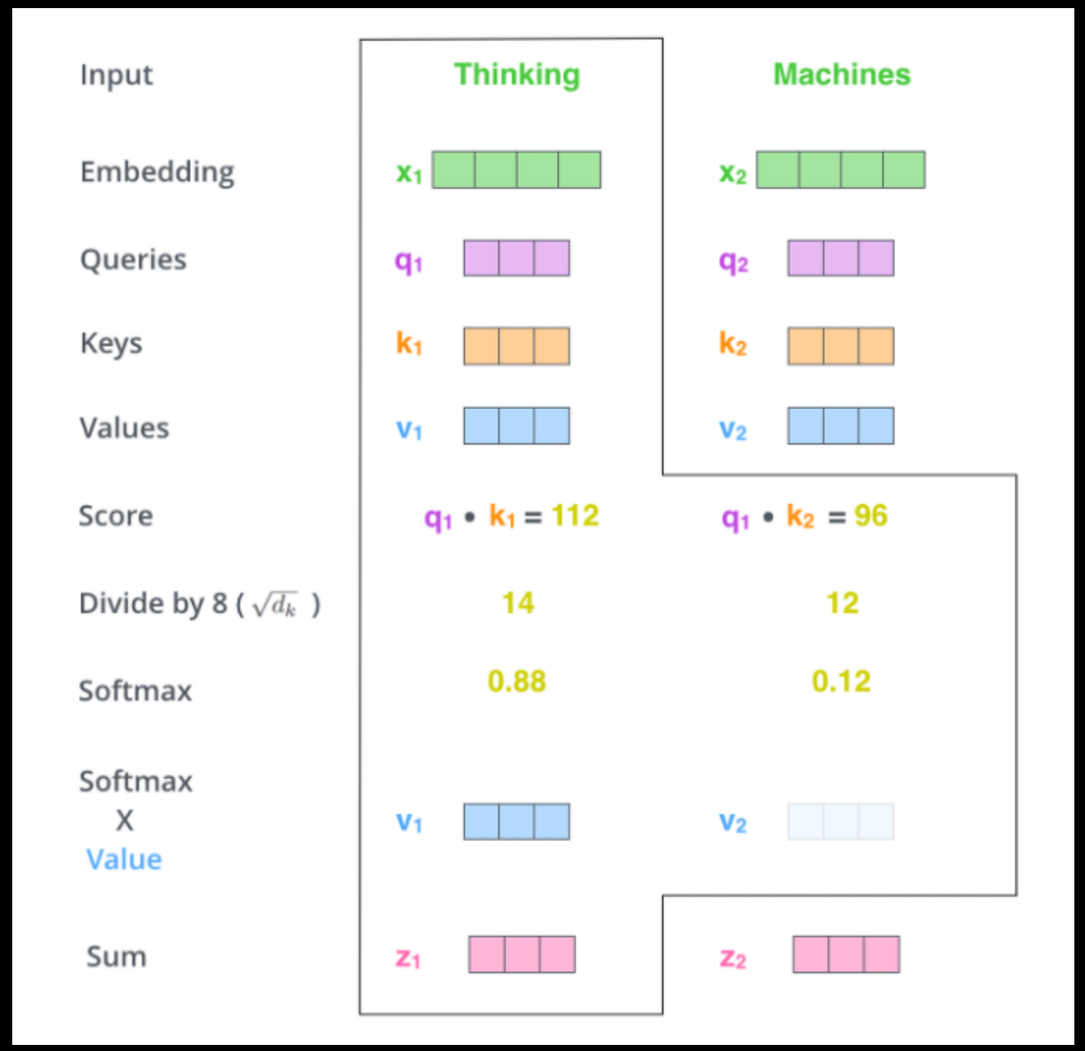
- Self attention
  - Attention is all you need (2017)
    - 언어모델에서 LSTM RNN을 밀어내고 기본 프레임이 됐다.
    - GPT3, BERT 등 모든 언어모델의 베이스
    - Input – input 간 관계 추정 가능



Self Attention

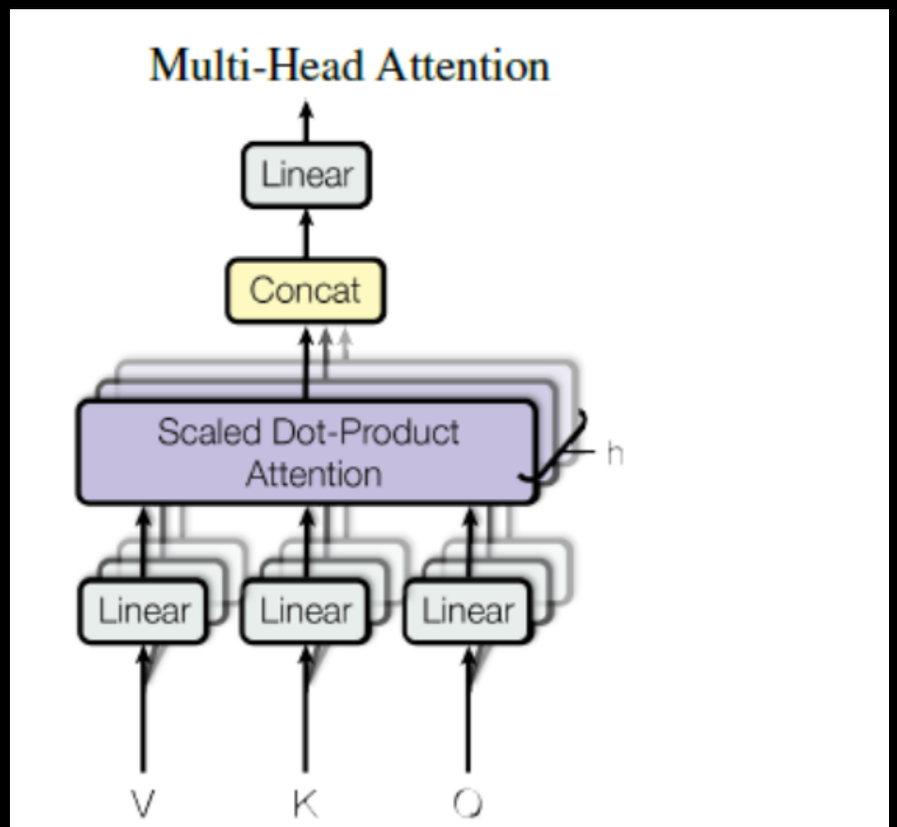
- Scaled dot product attention
  - Query(Q) : 영향을 받는 단어 A를 나타내는 변수
  - Key(K) : 영향을 주는 단어 B를 나타내는 변수
  - Value(V) : 그 영향에 대한 가중치

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



Self Attention

- Multi Head Attention
  - Scaled Dot-product attention 결과들을 붙여서, linear layer 통과시킨다.
  - res net 형태로 skip connection 추가한다.

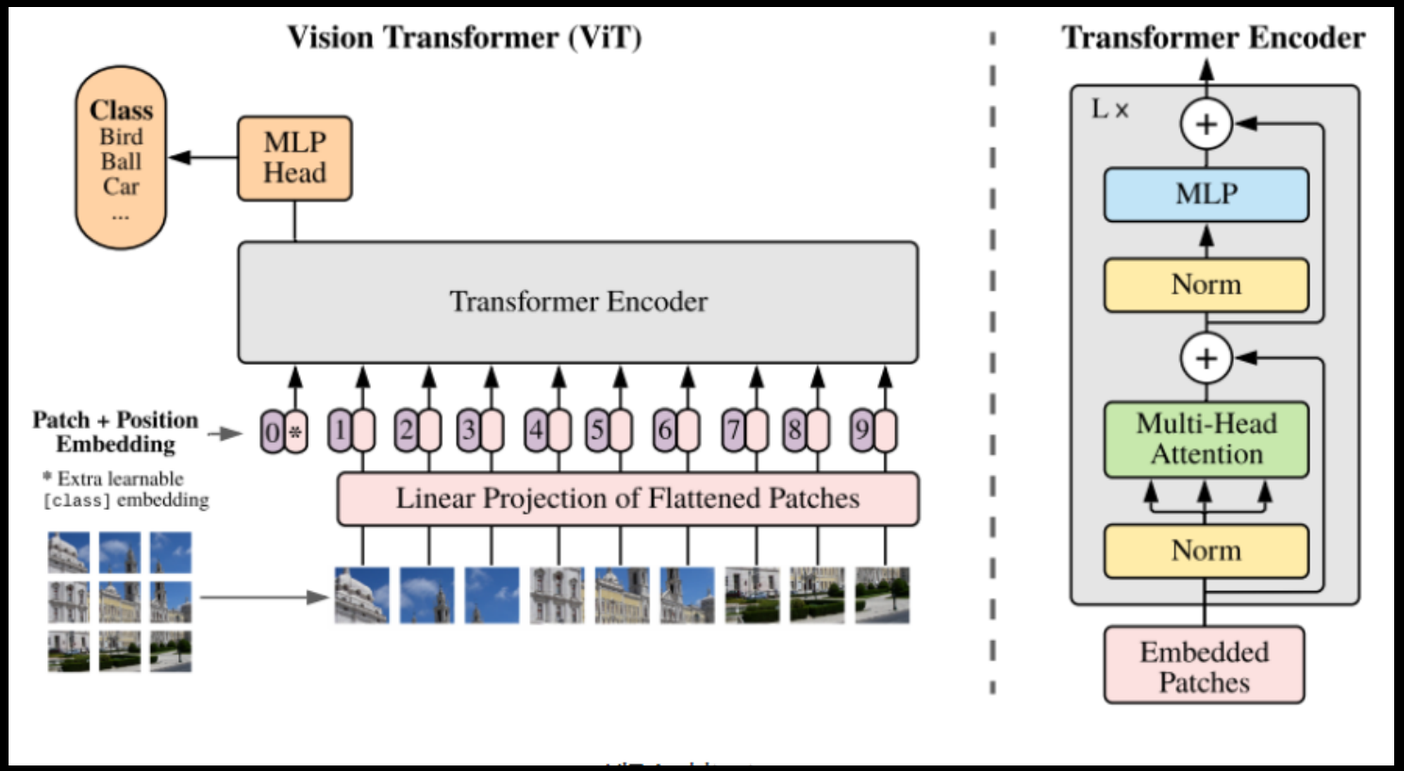


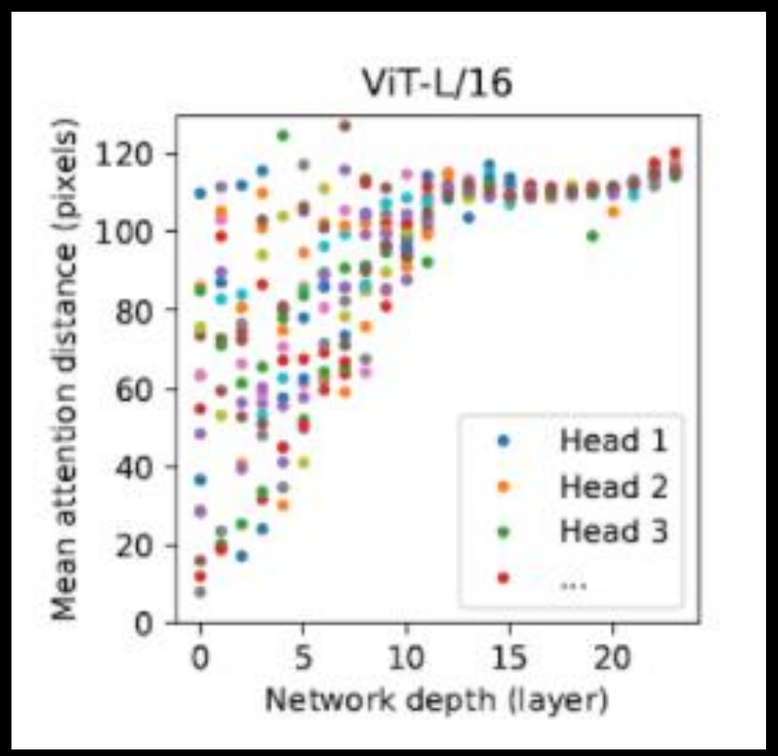
- 위의 그림을 수식으로 나타내면 다음과 같습니다.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

Transformer application

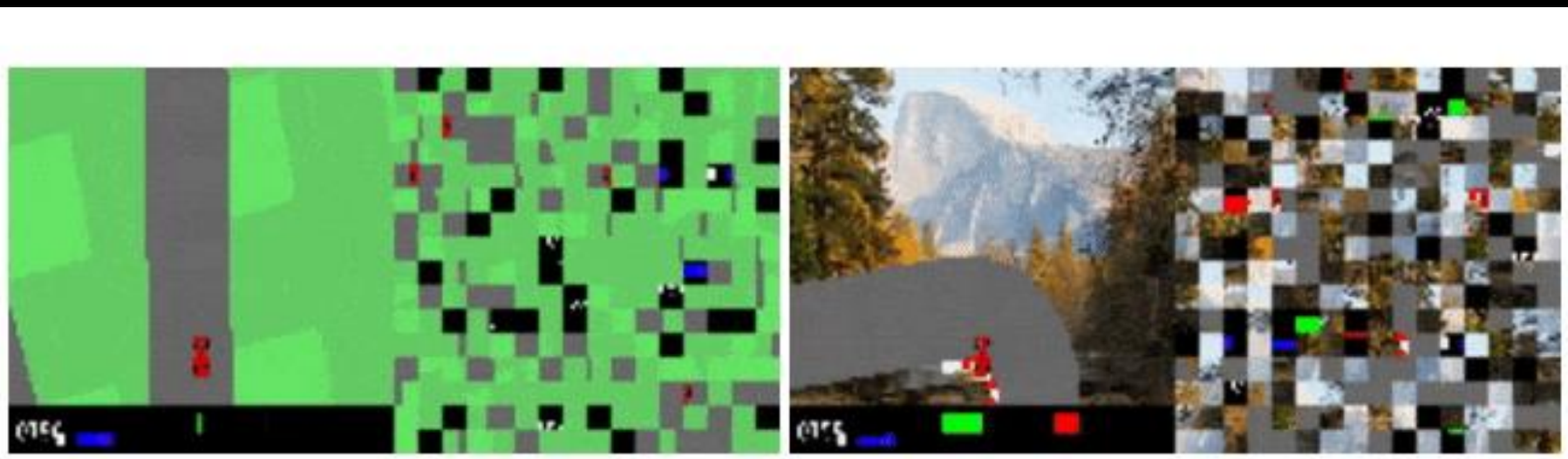
- Vision transformer
  - Convolution이란 inductive bias 없이 image classification 가능





Transformer application

- Attention neuron
  - Permutation invariant
  - Suffled input에서도 잘 동작
  - Redundant input에 대해서도 robust
  - https://attentionneuron.github.io/
  - 즉, 재학습이 필요 없음



We partition the visual input from CarRacing into a 2D grid of small patches, and shuffled their ordering. Without any additional training, our agent still performs even when the original training background (**left**) is replaced with new images (**right**).

Trajectory Transformer

Framework of Trajectory Transformer

- Structure
  - N차원 state, M차원 action, scalar reward 로 이루어진 trajectory τ
  - Continuous space 를 절대 그대로 쓰지 않고 tokenize (discretize)

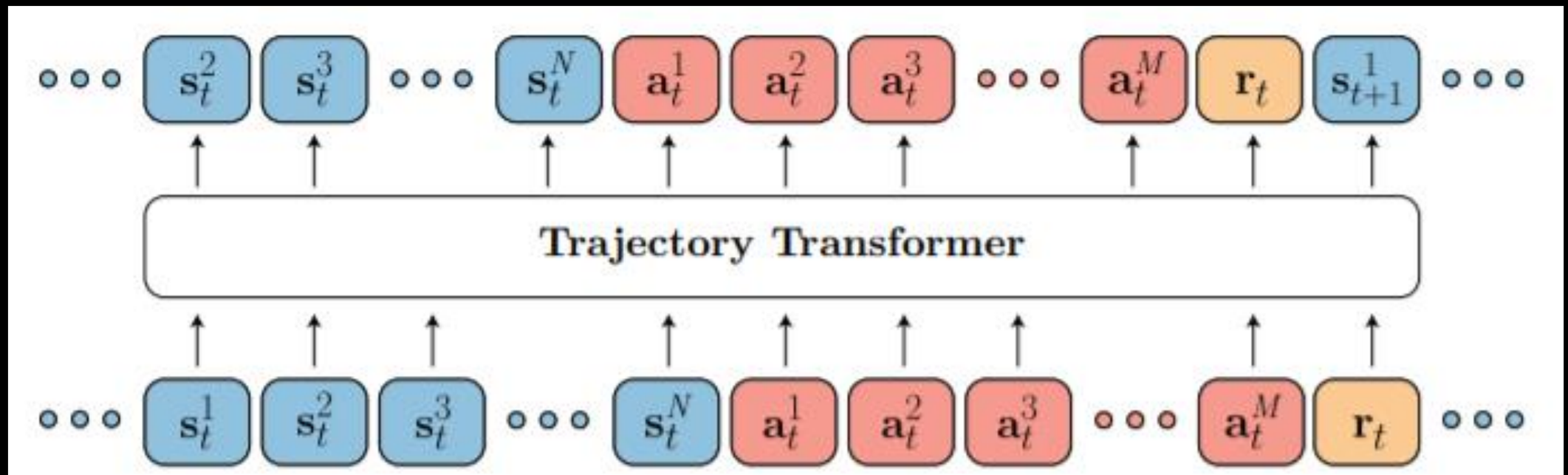


Figure 1 (Architecture) The Trajectory Transformer trains on sequences of (autoregressively discretized) states, actions, and rewards. Planning with the Trajectory Transformer mirrors the sampling procedure used to generate sequences from a language model.

$$\boldsymbol{\tau} = \left(\ldots, s_t^1, s_t^2, \ldots, s_t^N, a_t^1, a_t^2, \ldots, a_t^M, r_t, \ldots \right) \qquad t = 1, \ldots, T.$$

Framework of Trajectory Transformer

- Tokenization
  - N차원 State 에서 i번째 차원마다 사이즈 V 만큼 tokenization (논문에서 V=100)

$$(\max \mathbf{s}^i - \min \mathbf{s}^i)/V$$

  - Min-max 사이의 uniform 혹은 quantile 로 정의

  - Gaussian transition 같은 단순화 가정 없이 expressive 하게 trajectory를 정의하는 게 가능
    - 아마 가정으로 인해 trajectory transformer의 inductive bias가 들어가는 걸 막기 위해?

  - Action과 reward 도 동일하게 적용

Framework of Trajectory Transformer

- Loss term 의미
  - 1. state의 각 차원에 대해서 예측
  - 2. action에 대해서 예측
  - 3. reward에 대한 예측
  - 그냥 단순히 시퀀스에 대한 예측 = 사실상 supervised learning (RvS라고 많이 부름)
  - 최대 길이 512

$$\mathcal{L}(\tau) = \sum_{t=1}^{T} \Big(\sum_{i=1}^{N} \log P_\theta\big(s_t^i \mid \mathbf{s}_t^{<i}, \boldsymbol{\tau}_{<t}\big) + \sum_{j=1}^{M} \log P_\theta\big(a_t^j \mid \mathbf{a}_t^{<j}, \mathbf{s}_t, \boldsymbol{\tau}_{<t}\big) + \log P_\theta\big(r_t \mid \mathbf{a}_t, \mathbf{s}_t, \boldsymbol{\tau}_{<t}\big) \Big)$$

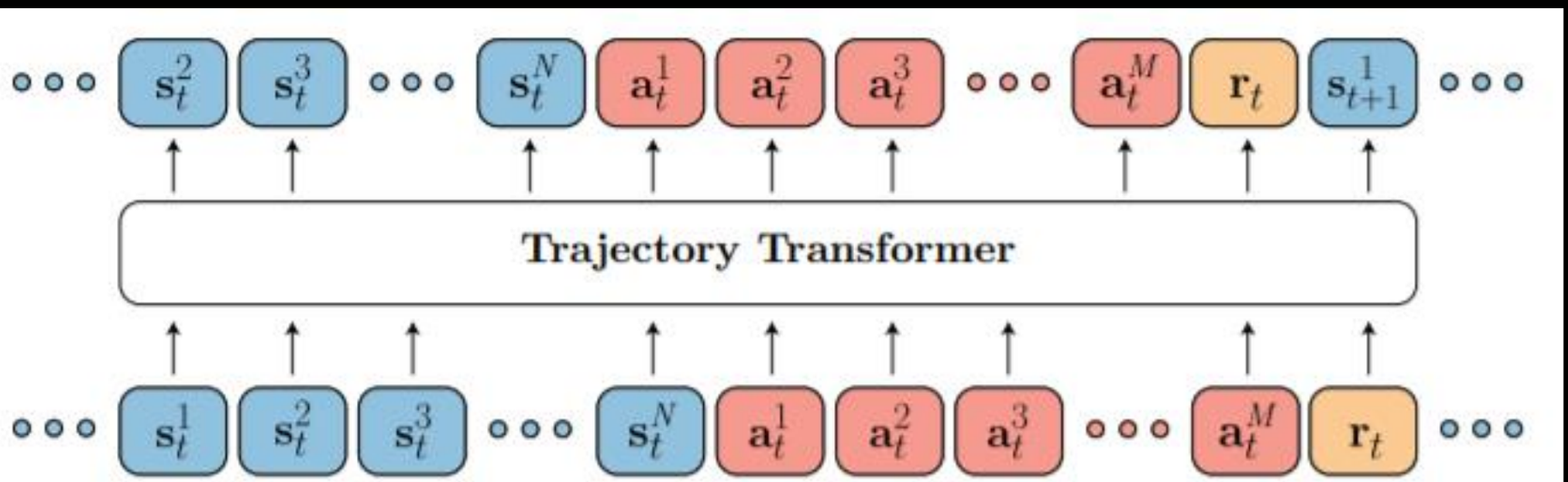


Figure 1 (Architecture) The Trajectory Transformer trains on sequences of (autoregressively discretized) states, actions, and rewards. Planning with the Trajectory Transformer mirrors the sampling procedure used to generate sequences from a language model.

Framework of Trajectory Transformer

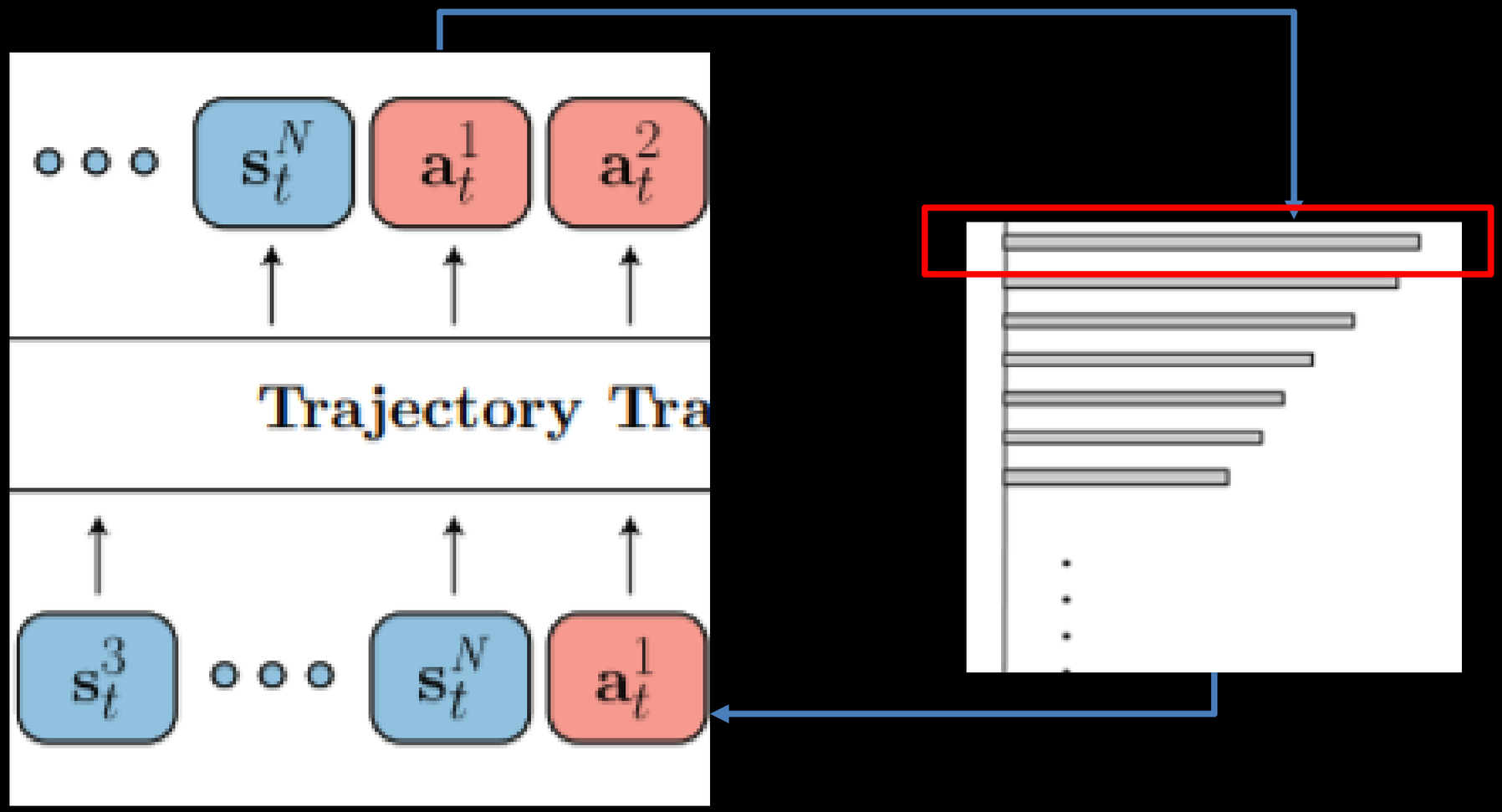
- Planning
  - Planning 으로는 beam search 사용
  - Seq2Seq에서 sequence 를 샘플링할 때, log probability Top 1 예측값만 쓰는 greedy decoding 만 하면 중간에 틀린 예측으로 전체 디코딩 결과가 무너짐.

Algorithm 1 Beam search

Require Input sequence $\mathbf{x}$, vocabulary $\mathcal{V}$, sequence length T, beam width B
Initialize $Y_0 = \{ ( ) \}$
for $t = 1, \ldots, T$ **do**
  $\mathcal{C}_t \leftarrow \{\mathbf{y}_{t-1} \circ y \mid \mathbf{y}_{t-1} \in Y_{t-1} \text{ and } y \in \mathcal{V}\}$ // candidate single-token extensions
  $Y_t \leftarrow \underset{Y \subseteq \mathcal{C}_t,\ |Y|=B}{\operatorname{argmax}} \log P_\theta(Y \mid \mathbf{x})$ // B most likely sequences from candidates
end for
Return $\underset{\mathbf{y} \in Y_T}{\operatorname{argmax}} \log P_\theta(\mathbf{y} \mid \mathbf{x})$

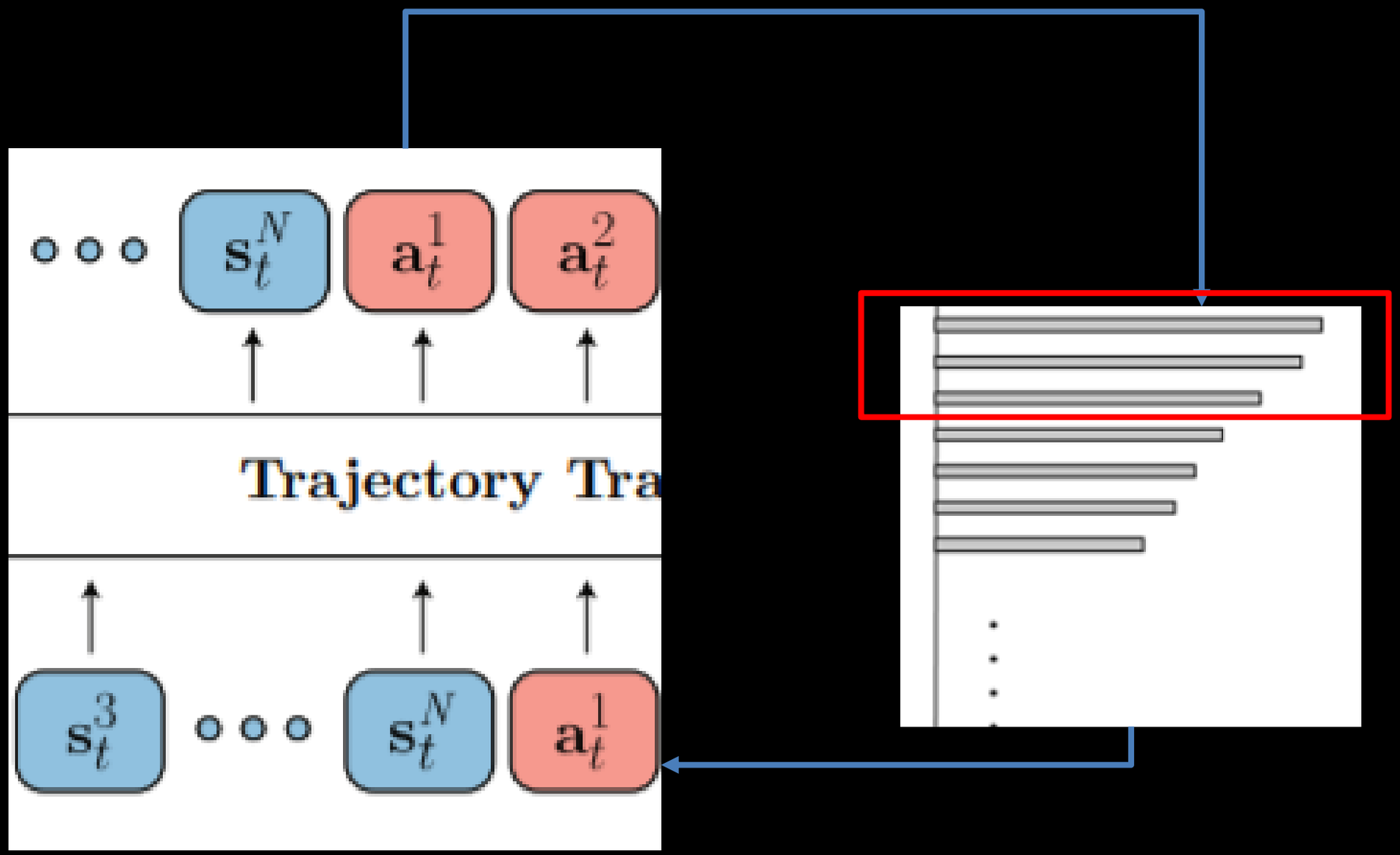
Framework of Trajectory Transformer

- Planning
  - Seq2Seq에서 sequence 를 샘플링할 때, Top 1 예측값만 쓰는 greedy decoding 만 하면 중간에 틀린 예측으로 전체 디코딩 결과가 무너짐.



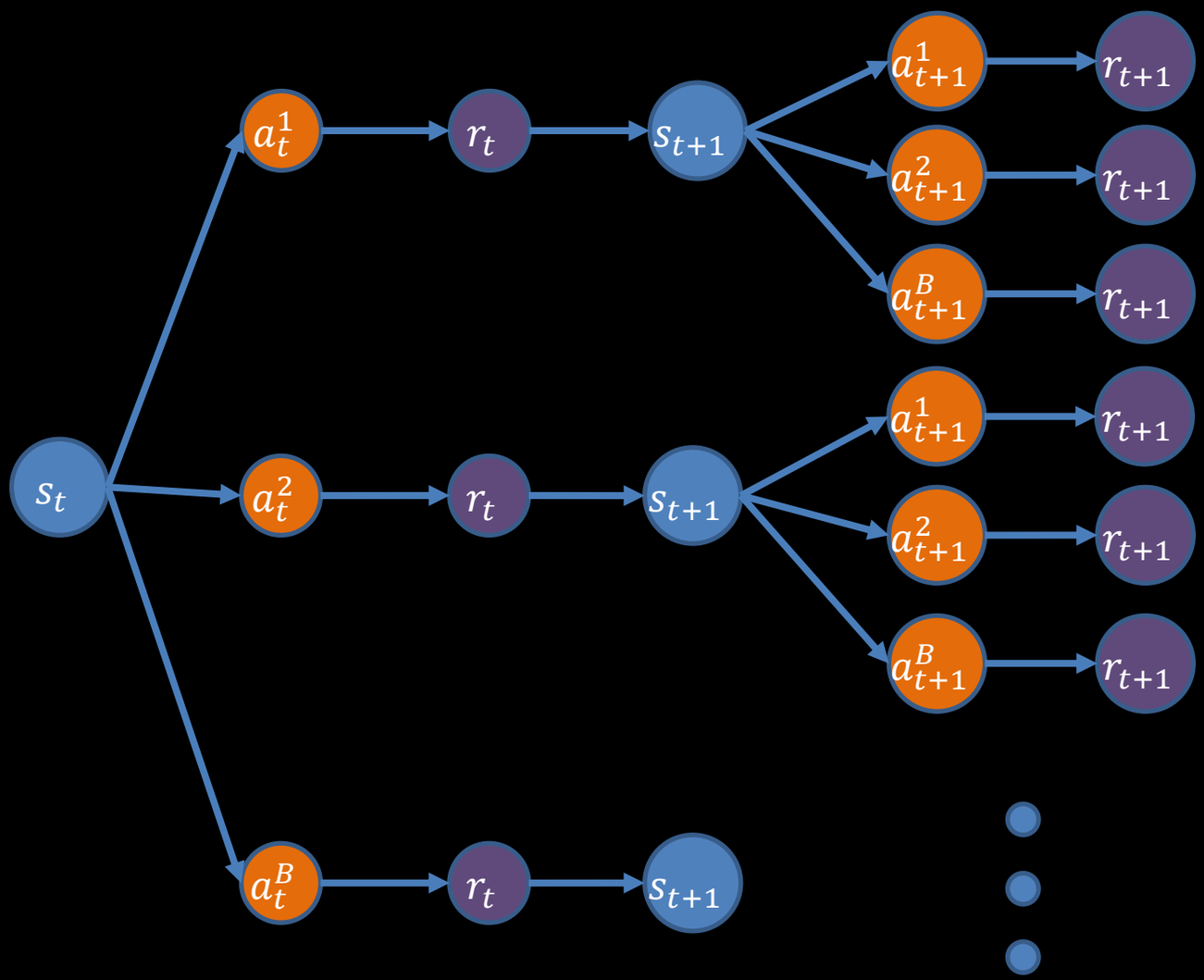
Framework of Trajectory Transformer

- Planning
  - Top 1 대신 Top K (beam width B) 를 고르고 decoding
  - B^2만큼 state token이 증가하지만, 거기서 logP 상위 B만큼만 고르고 나머지 버려서 반복.



Framework of Trajectory Transformer

- Planning
  - Top 1 대신 Top K (beam width B) 를 고르고 decoding
  - B^2만큼 state token이 증가하지만, 거기서 logP 상위 B만큼만 고르고 나머지 버려서 반복.
  - 최종적으로 후보 sequence 중 Rt가 제일 큰 seqeunce의 가장 최근 액션을 고른다
  - 보다 여러 후보군으로 search를 하기에 offline rl의 conservatism과 같은 효과를 내는 듯하다.

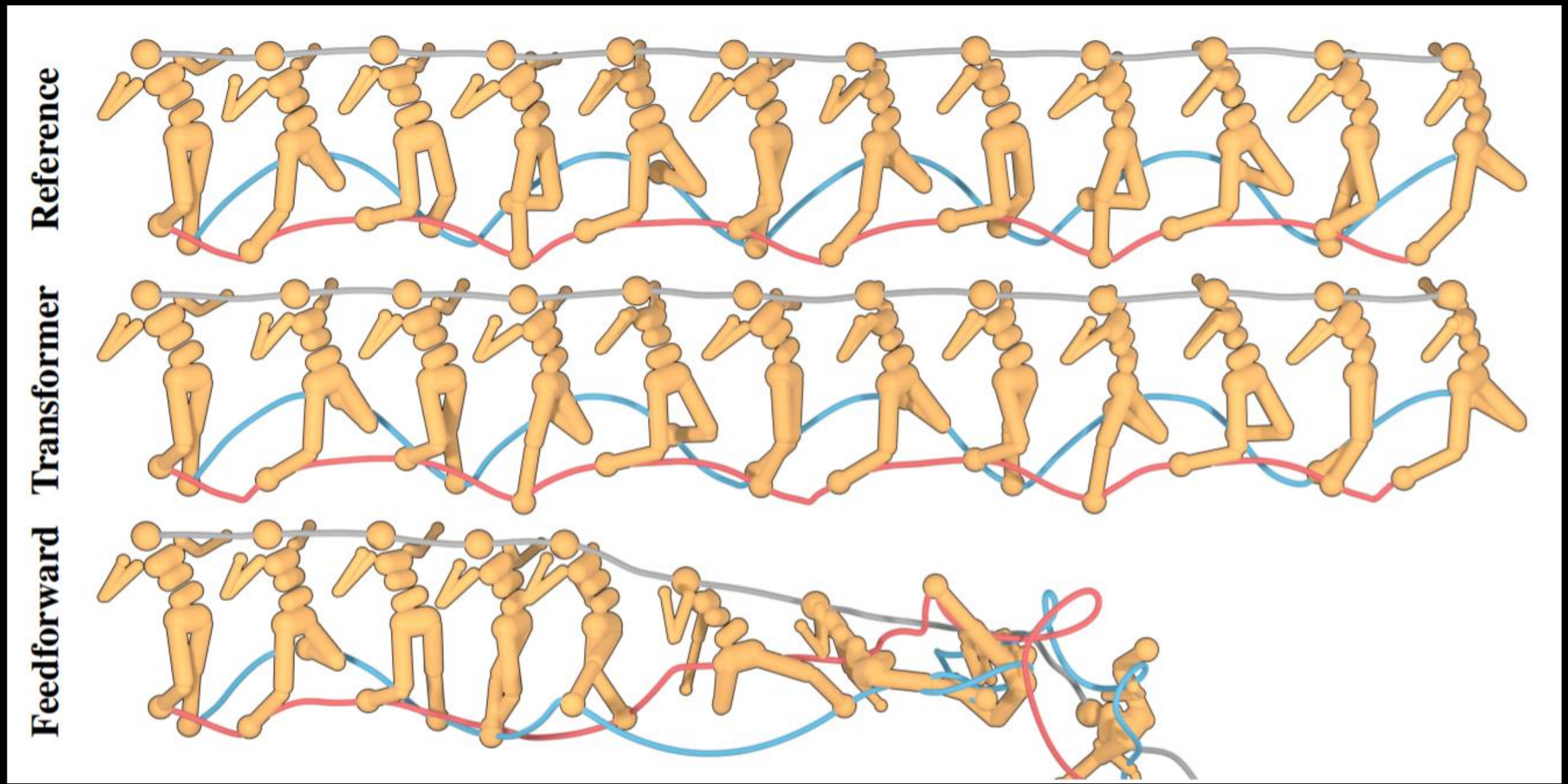


$$R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$$

Trajectory Transformer results

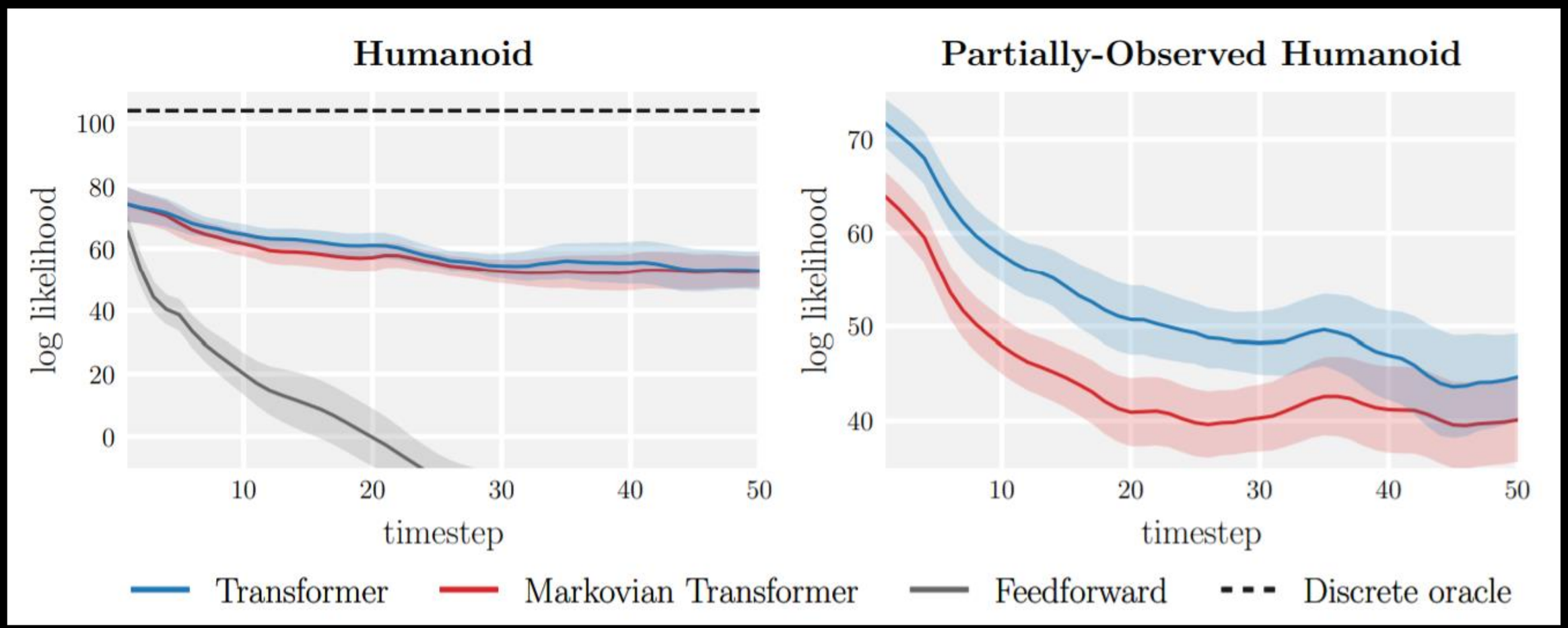
Trajectory Transformer results

- Trajectory에 대한 예측 성능



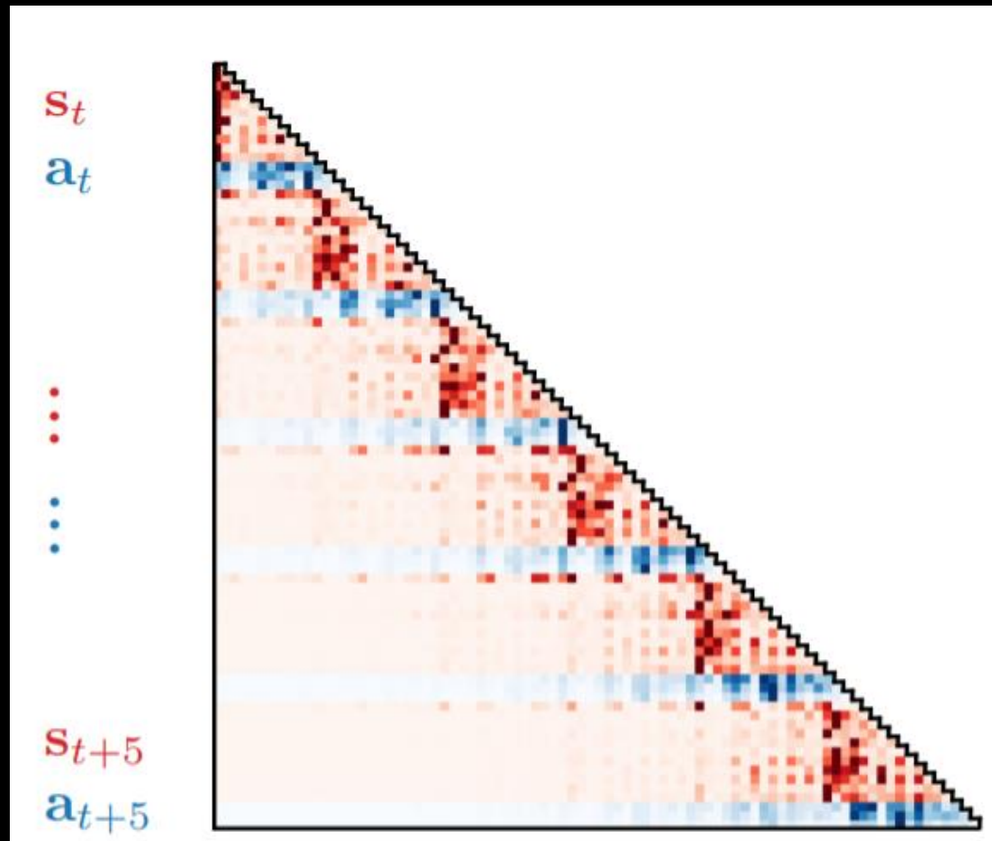
Trajectory Transformer results

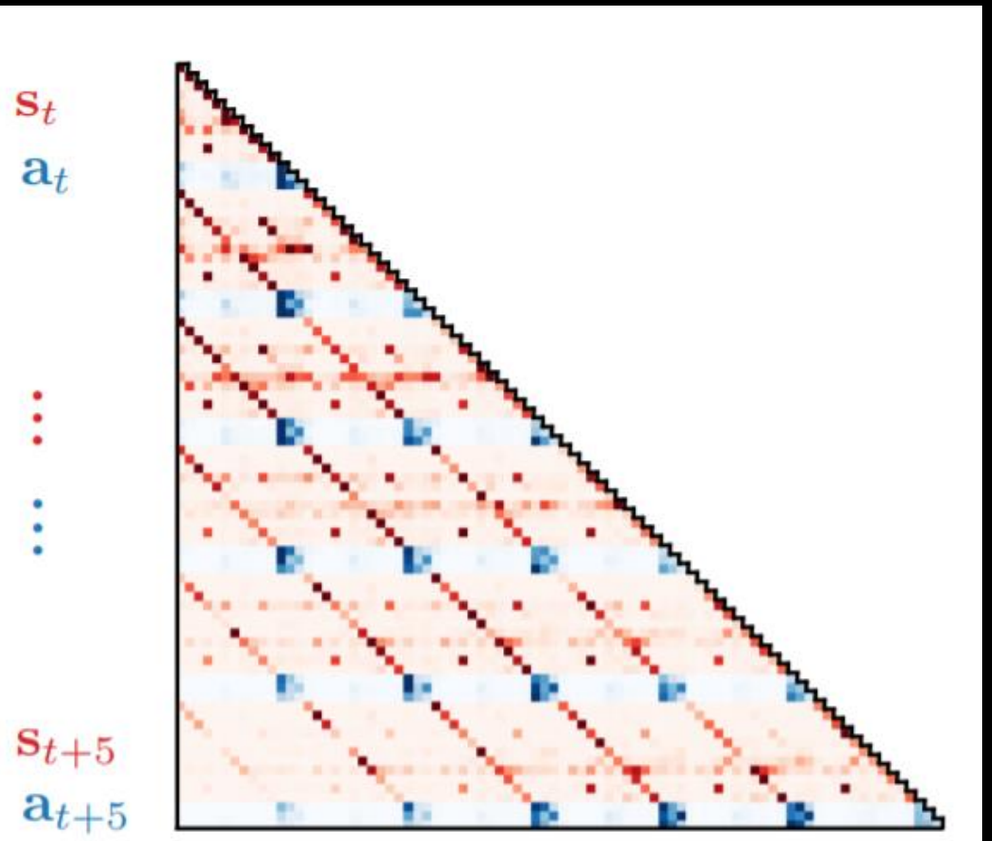
- Trajectory에 대한 예측 성능
  - 직전 state와 action 만 보게 하는 markovian transformer와 본래 TTO가 유사한 성능을 보임. -> 모델 스트럭쳐, expressive한 state tokenization이 long term prediction 에 영향을 미쳤을 거라고 추정
  - 일부 state 차원을 masking 하는 partially observable 실험에서는 본래 TTO가 근소하게 성능이 우위 -> POMDP 환경에서 one step이 아닌 long term condition이 정확도 향상에 영향을 주지 않았나.



Trajectory Transformer results

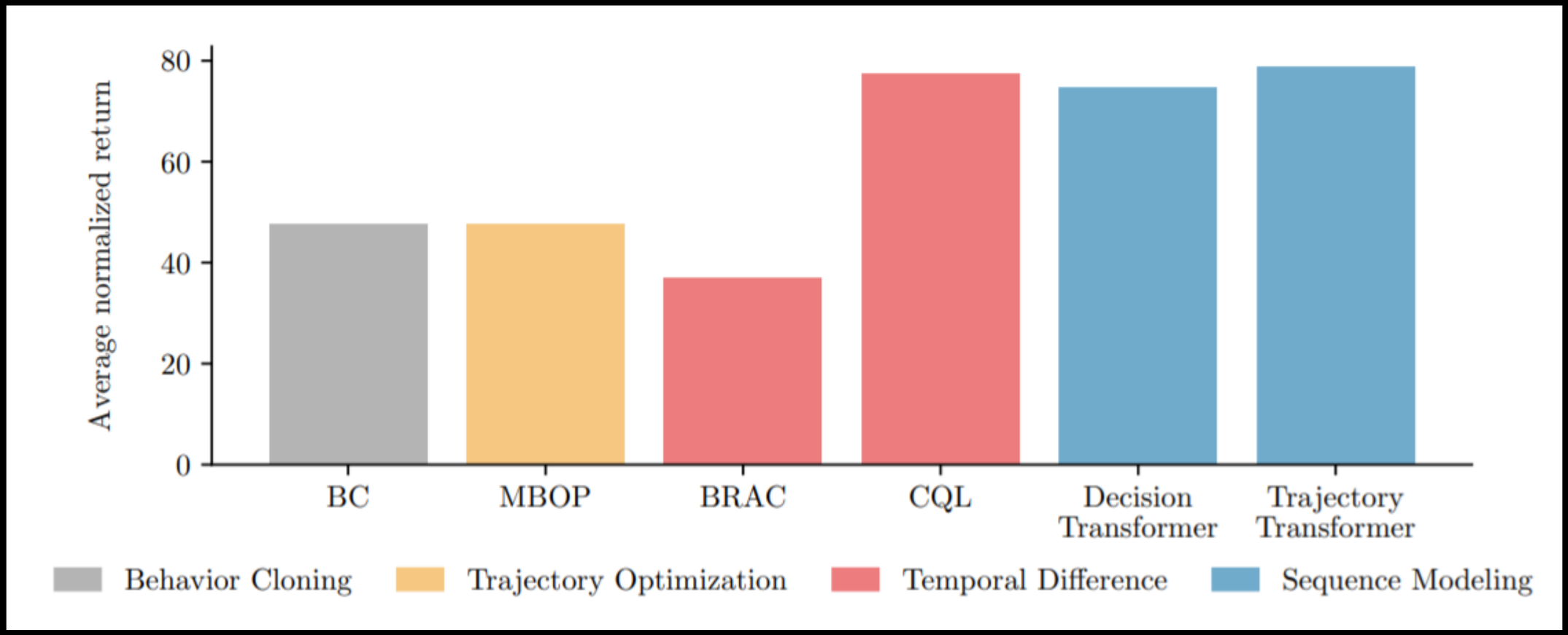
- Attention pattern
  - 첫번째 : s_t, a_t를 예측하는데 직전의 a_{t-1}, s_t만 보고 나머지는 attention을 두지 않는다.
  - 첫번째는 factorized single step prediction 즉 mdp 와 매우 유사
  - 두번째 : 첫번째와 다르게 과거 액션들에 대해 집중하고 있다.
  - Non Markov에다가 action filtering에 유사.
  - MDP 라는 inductive bias를 뺀 대신 모델의 capacity, 한계점이 올라가지 않았나 생각 (지극히 개인적 견해)





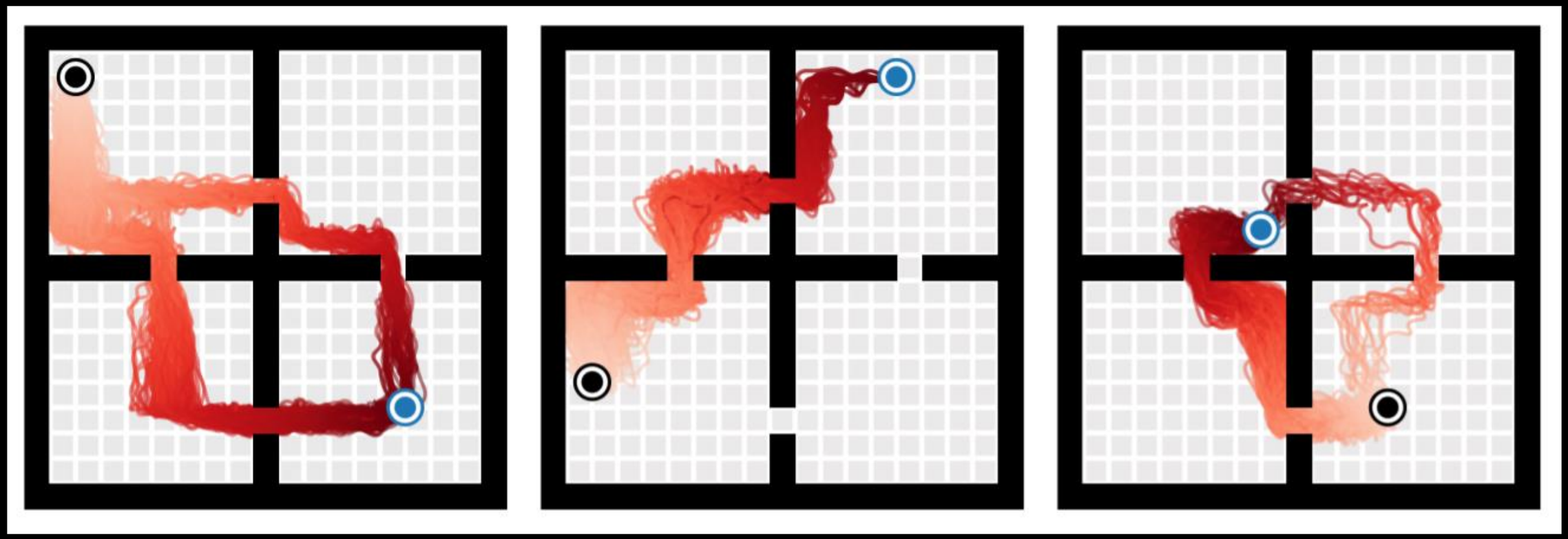
Trajectory Transformer results

- Offline RL
  - 다른 모델 대비 상대적으로 준수한 성능



Trajectory Transformer results

- Imitation learning & navigation
  - 파란색 goal만 바꾸어도 trajectory planning이 수행됨



END