한국컴퓨터그래픽스학회
Korea Computer Graphics Society Vol. 27, No. 5, P. 73~79

다양한 동작 학습을 위한 깊은신경망 구조 비교

박수환° 이제희*

서울대학교

{shpark, jehee}@mrl.snu.ac.kr

A Comparison of Deep Neural Network Structures for Learning Various Motions

Soohwan Park° Jehee Lee*

Seoul National University

요 약

최근 컴퓨터 애니메이션 분야에서는 기존의 유한상태기계나 그래프 기반의 방식들에서 벗어나 딥러닝을 이용한 동작 생성 방식이 많이 연구되고있다. 동작 학습에 요구되는 네트워크의 표현력은 학습해야하는 동작의 단순한 길이보다는 그 안에 포함된 동작의 다양성에 더 큰 영향을 받는다. 본 연구는 이처럼 학습해야하는 동작의 종류가 다양한 경우에 효율적인 네트워크 구조를 찾는것을 목표로 한다. 기본적인 fully-connected 구조, 여러개의 fully-connected 레이어를 병렬적으로 사용하는 mixture of experts구조, seq2seq처리에 널리 사용되는 순환신경망(RNN), 그리고 최근 시퀀스 형태의 데이터 처리를 위해 자연어 처리 분야에서 사용되고있는 transformer구조의 네트워크들을 각각 학습하고 비교한다.

Abstract

Recently, in the field of computer animation, a method for generating motion using deep learning has been studied away from conventional finite-state machines or graph-based methods. The expressiveness of the network required for learning motions is more influenced by the diversity of motion contained in it than by the simple length of motion to be learned. This study aims to find an efficient network structure when the types of motions to be learned are diverse. In this paper, we train and compare three types of networks: basic fully-connected structure, mixture of experts structure that uses multiple fully-connected layers in parallel, recurrent neural network which is widely used to deal with seq2seq, and transformer structure used for sequence-type data processing in the natural language processing field.

키워드: 딥러닝, 동작 생성, 컴퓨터 애니메이션, 시뮬레이션
Keywords: Deep Learning, Motion Generation, Computer Animation, Simulation

1. 서론

인간 혹은 동물 형태의 캐릭터가 등장하는 게임에서 그 캐릭터들의 동작의 품질은 게임 전체의 품질에 매우 큰 영향을 미친다. 게임의 볼륨이 커지고 품질이 높아질 수록, 게임 내의 캐릭터가 표현해야하는 동작의 다양성이 증가하며, 요구되는 동작의 자연스러움의 수준 역시 높아지게 된다. 최근 이런 AAA급 게임의 개발이 많이 이루어짐에 따라, 이에 맞는 고품질, 고효율의 모션 생성기술에 대한 연구도 많이 이루어지고있다.

기존의 게임개발에는 유한상태기계 기반의 모션 생성 방식이 많이 사용되었다. 이 방식은 개발자의 의도대로 정확한 동작을 생성 할 수 있지만, 모든 동작들과 그 동작의 연결과정들을 일일히 아티스트가 손으로 지정하고 생성해야했다. 따라서 캐릭터가 수행해야하는 동작의 종류가 많아지게되면 그 동작들을 처리하기 위해 필요한 노력이 감당하기 어려울 정도로 커지게 된다. 비교적 최근의 게임들에는 모션매칭이라는 그래프 기반의 동작 생성 방식이 사용되었다. 이 방식은 동작간의 연결을 일일히 처리해줄 필요는 없지만 여전히 정교한 시스템 디자인이 필요하며, 동작의 양이 증가하면 그에 따라 메모리 요구치가 늘어난다.

*corresponding author: Jehee Lee/Seoul National University(jehee@mrl.snu.ac.kr)

Received : 2021.11.19./ Review completed : 1st 2021.11.25. / Accepted : 2021.11.26.
DOI : 10.15701/kcgs.2021.27.5.73
ISSN : 1975-7883(Print)/2383-529X(Online)

최근 딥 러닝은 많은 양의 데이터를 효율적으로 처리 할 수 있다는 장점을 바탕으로 다양한 분야에서 응용되고 있다. 많은 연구들이 이런 장점을 동작 생성 방식에도 적용하여 좋은 결과를 거두고 있다. 이 연구들은 RNN, transformer와 같은 시퀀스형태의 데이터 처리에 적합한 네트워크 구조나 PFNN이나 MANN와 같이 다양한 종류의 동작을 처리하기에 적합한 mixture of experts 모델 기반의 구조들을 주로 사용하였다.

본 연구에서는 이런 여러가지 형태의 네트워크 구조들을 비교, 분석하는것을 목표로한다. 그 중에서도 모션의 다양성이 큰 피격 반응 동작들을 활용하여 네트워크의 효율성을 비교하고자 한다. 피격 동작은 여러 피격 조건들에 영향을 받는데, 공격의 세기, 방향, 부위, 피격 자세 등 여러 조건들이 변화함에 따라서 그에 따른 피격 반응 역시 다양하게 변화하게 된다. 이런 피격 동작을 생성하도록 학습하여 동작이 다양한 경우에 네트워크 구조에 따른 결과를 비교하고 효율적은 네트워크 구조를 찾고자 한다.

2. 관련 연구

과거의 많은 연구들이 실제 사람의 움직임을 캡쳐한 모션 캡쳐 데이터를 바탕으로 원하는 동작을 생성하는 데이터 기반 동작 생성 방식을 연구했었다. 그래프 기반 방식 [1, 2]은 주어진 모션 데이터들로부터 각 모션 프레임을 꼭지점, 그 프레임간의 전이를 변으로 하는 그래프를 구성하고, 그 그래프를 순회하면서 모션을 생성하는 방식이다. 사용자의 목적에따라 적절한 꼭지점과 변들을 탐색함으로써 적절한 모션을 생성하는것이 가능하다. 모션 매칭 [3, 4]은 직접적인 그래프를 구성하는 것이 아니라 모션간의 전이를 결정하는 특징 벡터와 비용 함수를 정의함으로써 모션을 생성한다. 일정 주기마다 현재 캐릭터의 상태와 사용자의 제어 목적에 가장 적합한 모션 프레임을 데이터베이스에서 탐색하고 조건을 만족할경우 그 프레임으로 전이하는 것을 반복하는 방식으로 모션을 생성한다.

딥러닝의 등장 이후에는 이런 알고리즘들을 네트워크에 학습시키려는 많은 연구들이 있었다. Lee et al. [5]은 순환 신경망(recurrent neural network)형태의 네트워크 구조를 기반으로 다양한 목적에 따라 적절한 동작을 생성하는 시스템을 학습하였다. 연속적으로 주어지는 사용자의 입력에 따라 다양한 동작을 수행하도록 캐릭터를 제어 할 수 있으며, 생성되는 동작들이 데이터 속에 포함되어있는 다양한 법칙들을 만족하도록 네트워크를 학습했다. 그 후속 연구 [6]에서는 정해진 시간안에 사용자가 입력한 제어 목적을 수행하도록 학습했다. Phase-functioned neural network(PFNN) [7]은 사람의 보행 사이클에서의 진행 정도를 나타내는 페이즈를 기반으로 네트워크의 파라미터를 합성하는 네트워크 구조이다. 이를 통해 주기적으로 반복되는 사람의 동작들을 잘 학습 할 수 있었다. Zhang et al. [8]은 페이즈 변수 또한 네트워크가 학습해서 생성하도록 mixture of experts 형태의 구조를 제안했다. MANN이라 불리는 네트워크 구조를 통해 직접적인 페이즈 계산이 어려운 4족보행 동작도 잘 학습 할 수 있도록 했다.

Figure 1: motion generation system

Holden et al. [9]은 end-to-end의 형태로 디자인되는 위의 네트워크 구조들과 달리 기존 모션 매칭의 구조를 유지한 채로, 각각의 주요 파트들을 네트워크로 대체하는 형태로 시스템을 구성하였다. 이를 통해 한번 학습이 완료되면 수정이 어렵다는 네트워크 기반 모델의 단점을 최소화 할 수 있었다.

모션 생성은 시퀀스의 형태로 구성되며, 과거의 동작이 현재의 동작에 영향을 미친다는 점에서 자연어 처리와 유사하다고 할 수 있다. 최근 자연어 처리 분야에서는 Transformer [10]라는 attention 매커니즘 기반의 네트워크 구조가 좋은 결과를 거두고 있다. attention 매커니즘은 모든 히스토리를 누적해서 사용하던 기존의 방식과 다르게, 현재 시점에 중요한 영향을 미치는 과거에만 집중하도록 네트워크를 구성하는 방식이다. 본 연구에서는 이와 유사하게 attention 매커니즘을 기반으로 하는 네트워크 구조를 설계하고 다른 모델들과 비교할 것이다.

3. 피격 반응 생성 시스템

피격 반응 동작은 동작에 영향을 주는 조건의 종류가 많고, 조건의 작은 변화에 따라서도 그 결과가 크게 달라 질 수 있기 때문에 반응 동작의 종류가 매우 다양하다. 이런 동작을 모두 모션캡쳐를 통해 모두 확보하는것은 어렵고, 따라서 우리는 다양한 피격 반응을 합성할 수 있는 피격 반응 생성 시스템을 디자인하였다.

이 시스템은 주어진 피격 반응 모션 캡쳐 데이터를 바탕으로 물리시뮬레이션을 활용해 다양하게 변화하는 피격 반응 동작들을 생성한다. 실제 사람이 피격 당할 때, 사람이 피격 사실을 인지하고 반응하기까지 약간의 짧은 딜레이 [11, 12]가 존재한다. 이 시간동안은 사람의 의지가 아니라 공격에 의한 물리적 충격이 동작에 더 큰 영향을 미치게 되며, 그 이후 사람이 피격 사실을 인지하고 몸을 제어하면서 부터 사람의 특성에 따른 피격 반응 동작이 나타나게 된다. 우리는 이런 사람의 특성을 모방하여 피격 반응 동작을 생성하도록 시스템을 구성하였다[Figure 1].

시스템은 물리 시뮬레이션을 통해 동작을 생성하는 순간 피격 반응 구간과 주어진 모션 데이터로부터 모션 매칭 알고리즘을 통해 동작을 생성하는 후속 피격 반응 구간으로 나눠진다. 어떤 상태에서 피격이 발생할 경우 짧은 시간(0.1s)동안 캐릭터는 주어진 피격 조건 c에 따라 피격 순간 캐릭터의 자세 p로부터 가상의 물리 환경속에서 시뮬레이션 된다. 피격 조건 $c = \{f, \theta, b\}$로 구성된며, f은 공격의 강도, θ는 공격의 방향, b는 공격 당한 신체 부위를 의미한다. 물리 시뮬레이션이 종료 되면, 종료된 순간의

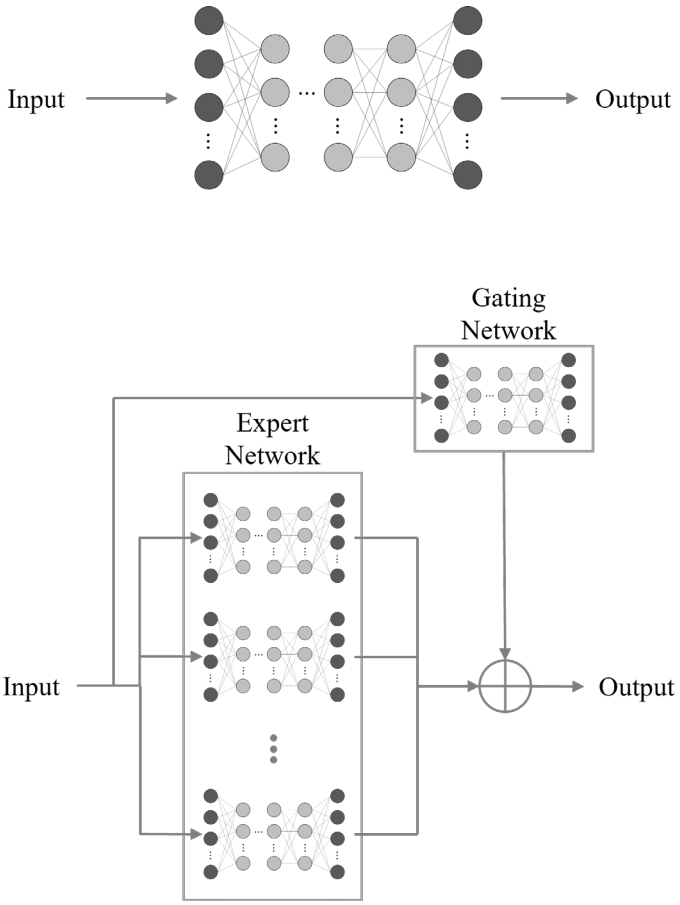


Figure 2: Network structure, top : fully-connected, bottom : mixture of experts

캐릭터의 상태 $\tilde{p}$와 피격 조건 c을 기반으로 모션 매칭을 통해 적절한 프레임을 모션 데이터베이스에서 탐색하고 그 프레임으로 전환한다. 이후 캐릭터는 모션매칭을 통해 기본 상태로 복귀하게 되며, 도중에 다시 피격이 발생할 경우 위의 과정을 반복하게 된다.

4. 네트워크 학습

4.1 네트워크 종류

본 연구에서는 3장의 시스템으로부터 다양한 피격 반응을 포함하는 학습 데이터를 생성하고, 이를 이용해 네트워크를 학습한 뒤 그 결과를 비교한다. 공통된 학습 데이터와 손실 함수 하에서, 네트워크 구조와 파라미터를 변화시키며 실험한다. 실험에 사용된 네트워크의 종류는 네가지이며, 각각 fully-connected, mixture of experts, RNN, transformer이다.

Fully-Connected Fully-connected(fc) 레이어는 가장 기본적인 신경망 형태로 레이어의 모든 입력과 출력이 서로 연결되어있는 형태이다[Figure 2, 위]. Fully-connected 모델은 이런 fc 레이어를 여러번 연결해서 만든 형태의 네트워크 구조이다.

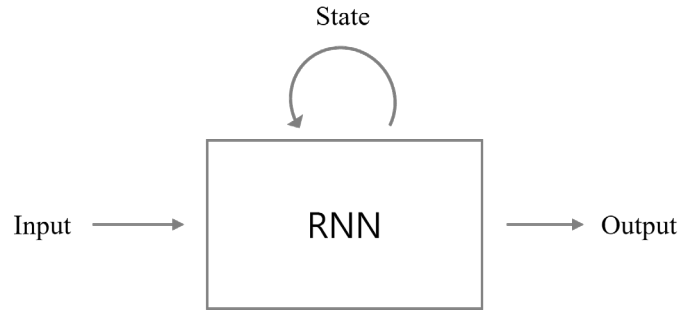


Figure 3: Network structure, recurrent neural network

$$o_t = W o_{t-1} + b \tag{1}$$

$o, b \in \mathbb{R}^n, W \in \mathbb{R}^{n \times n}$ 이며 n은 레이어의 노드 개수를 의미한다. 본 연구에서는 $n = 1024$개의 노드로 이루어진 3개의 레이어를 연속적으로 배치하여 네트워크를 구성하였다.

Mixture of Experts 여러개의 expert 네트워크들을 병렬적으로 계산한뒤, gating 네트워크의 출력값으로 expert 네트워크들의 출력값을 합성해서 최종 결과물을 계산하는 네트워크 구조를 mixture of experts 모델이라 한다[Figure 2, 아래]. 랜덤하게 정해지는 초기 파라미터에 따라 각각의 expert는 서로 다른 방향으로 학습되게 되며, 다양한 종류의 데이터를 학습하는데 효율적이다. 본 연구에서는 각각의 expert와 gating 네트워크는 512개의 노드로 구성된 fc 레이어 세개로 이루어 지며, 총 8개의 expert를 사용하였다. 각각의 expert별로 서로 다른 태스크에 적합하게 미리 학습을 시키는 경우도 있지만, 본 연구에서는 전체 네트워크 구조가 동시에 학습을 시작하도록 디자인하였다.

RNN 순환신경망(recurrent neural network, RNN)은 시퀀스 투 시퀀스 분야에서 널리 사용되는 네트워크 구조이다. 이전까지 들어온 입력에 기반하여 네트워크 내부에 스테이트를 보존하고 있으며 이 스테이트를 통해 현재의 출력을 생성하기 위해 과거의 입력을 참조 할 수 있도록 한다[Figure 3]. 본 연구에서는 Long Short-Term Memory(LSTM) [13]를 기반으로 RNN을 구성했다. 512개의 노드로 이루어진 4개의 LSTM 유닛을 이용해 네트워크를 디자인하였다.

Transformer 트랜스포머는 어텐션 메커니즘을 기반으로 디자인 된 시퀀스 투 시퀀스 모델이다. 순차적으로 입력을 받아서 처리하던 기존의 시퀀스 투 시퀀스 모델과 달리, 어텐션 매커니즘은 현재 프레임과 관련이 높은 과거에 집중(attention)해서 높은 가중치를 주고 이로 부터 결과를 생성한다. 어텐션은 쿼리 Q, 키 K, 밸류 V의 함수로 구성된다.

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V \tag{2}$$

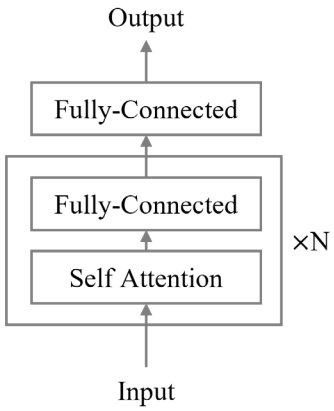


Figure 4: Network structure, transformer

K와 V는 서로 대응되는 쌍으로 구성된다. Q와 K의 내적을 통해 각 V의 가중치를 구하고, 이 가중치를 바탕으로 V에 가중합연산을 취해 최종 결과를 구하는 연산이다. 트랜스포머에는 입력값 간에 서로 어텐션 연산을 하는 셀프 어텐션과 디코더의 출력 목표 값을 쿼리로 인코더의 결과값에 어텐션을 수행하는 인코더-디코더 어텐션이 사용된다. 하나의 시퀀스로를 인코딩해서 다른 종류의 시퀀스로 디코딩하는 트랜스포머와 달리, 본 연구는 연속적으로 하나의 모션 시퀀스를 생성해나가는 것이 목표이기 때문에, 셀프 어텐션만을 이용해서 모델을 구성하였다[Figure 4]. 각 레이어의 노드는 512개를 사용하였으며, 셀프 어텐션과 fully-connected로 구성된 인코딩 블럭의 수는 4개, 그리고 어텐션 헤드의 수는 8개를 사용하였다.

위의 설명에서는 각 네트워크 구조 별 기본적인 특성과 형태에 대해 설명하였다. 실제 구현상에서는 이 외에도 학습 성능향상을 위해 leaky-relu와 같은 활성 함수와 dropout 레이어, residual 레이어등이 사용되었다.

4.2 학습 데이터 생성

학습 데이터는 3장의 피격 반응 생성 시스템으로부터 생성된다. 매 프레임마다 일정한 확률(3%)로 랜덤한 피격이 발생하며, 이를 정해진 프레임만큼 반복하여 모션 시퀀스 $M = \{m_t\}$를 생성한다. 과도한 피격을 방지하기 위해 피격이 한번 발생한 후에는 일정 시간(10프레임)동안은 피격이 발생하지 않으며, $f \in [300N, 900N]$, $\theta \in [-\pi, \pi]$의 조건으로 피격 세기와 방향을 선택하고, 피격 부위는 상체 부위중 랜덤하게 선택하였다.

생성된 모션 시퀀스 M은 네트워크 학습을 위한 포즈 시퀀스 $Y = \{y_t\}$와 컨트롤 파라미터 시퀀스 $X = \{x_t\}$의 형태로 변환된다. 포즈 $y = \{y_{\text{root}}, y_{\text{joint}}, y_c\}$로 표현되며, $y_{root} \in \mathbb{R}^6$는 이전 프레임의 루트 대비 현재 프레임의 루트 위치이고 $y_{joint} \in \mathbb{R}^{132}$는 각 조인트의 로테이션이며 $y_c \in \mathbb{R}^2$는 각 발의 접촉 정보이다.. 컨트롤 파라미터 x는 피격에 대한 정보를 네트워크의 입력으로 사용 할 수 있도록 다시 표현한 벡터이다. $x = \{\alpha, t, f, \cos(\theta), \sin(\theta), l, \beta\}$로 정의된다. $\alpha \in \mathbb{R}^2$는 피격 상태와 아닌 경우를 구분하는 원 핫 벡터이고, t는 피격이 지속되는 시간이며, $l \in \mathbb{R}^6$은 캐릭터 루트 기준 피격 위치와 피격당한 부위 기준의 피격 위치이다. $\beta \in \mathbb{R}^{22}$는 피격 당한 부위정보를 원 핫 벡터의 형태로 표현한 값이다. 피격이 발생한 순간부터 10프레임간 피격 정보를 담은 x가 입력으로 주어지며, 나머지 구간에는 $\alpha = \{1, 0\}$이고 이를 제외한 모든 값이 0인 기본 상태의 x가 주어진다.

생성된 X와 Y로부터 네트워크는 과거 히스토리 H_t와 현재 프레임의 x_t, y_t를 입력으로 받아 다음 프레임의 $\hat{y}_{t+1}$을 출력하도록 학습된다.

$$\hat{y}_{t+1} = G(H_t, x_t, y_t). \tag{3}$$

4.3 손실 함수

네트워크 학습을 위한 손실 함수는 아래와 같이 정의된다.

$$L = L_r + L_j + L_{fk} + L_v + L_{\text{foot}} + L_{\text{adv}}. \tag{4}$$

L_r과 L_j는 각각 루트와 조인트 로테이션의 차이를 최소화 하기 위한 항이다. y는 학습 데이터에 있는 실제 참 값이고 $\hat{y}$ 는 네트워크에 의해 출력된 결과이다.

$$\begin{aligned} L_r &= \sum_t \|y_{\text{root},t+1} - \hat{y}_{\text{root},t+1}\|^2, \\ L_j &= \sum_t \|y_{\text{joint},t+1} - \hat{y}_{\text{joint},t+1}\|^2. \end{aligned} \tag{5}$$

L_{fk}와 L_v는 각각 캐릭터 루트 좌표계 에서의 각 조인트의 위치와 속도의 차이를 최소화 하는 항이다. 포워드 키네마틱스 연산을 통해 조인트 로테이션 값들 루트 좌표계의 포지션으로 변환해서 계산한다.

$$\begin{aligned} L_{fk} &= \sum_t \|\text{FK}(y_{t+1}) - \text{FK}(\hat{y}_{t+1})\|^2, \\ L_v &= \sum_t \|(\text{FK}(y_{t+1}) - \text{FK}(y_t)) - (\text{FK}(\hat{y}_{t+1}) - \text{FK}(\hat{y}_t))\|^2. \end{aligned} \tag{6}$$

L_{foot}은 캐릭터의 발이 미끄러지는것을 보정해주기 위한 항이다.

$$L_{\text{foot}} = \sum_t \sum_{k \in \text{Feet}} \hat{y}^k_{c,t} \hat{y}^k_{c,t+1} \|\text{FK}(\hat{y}_{t+1})^k - \text{FK}(\hat{y}_t)^k\|^2, \tag{7}$$

$\hat{y}^k_{c,t}$는 k번째 조인트가 땅에 닿았는지 여부를 판단하는 값으로, 0.5보다 작을 경우 0, 클경우 1이된다. $\text{FK}(\hat{y}_t)^k$는 k번째 조인트의 루트 좌표계에서의 위치를 의미한다. 즉, 두 프레임 연속으로 발이 땅에 닿아있을 경 우, 두 발의 위치 차이를 최소화하도록 설계된 함수이다.

직접적인 손실 함수 설계로는 포함할 수 없는 특성들을 비교하

Table 1: The number of parameters and performance of networks

Network type	#Parameters (M)	Performance(ms)
fully-connected 1024	4.91	1.08
fully-connected 2048	16.12	3.04
mixture of experts	8.47	1.39
RNN	7.78	1.91
transformer	6.96	2.49

기 위해 우리는 LSGAN [14]의 형태로 디자인된 적대적 손실 항 L_{adv}을 추가하였다.

$$\begin{aligned} L_{\text{adv}} &= \sum_t \|1 - D(H_t, x_t, y_t, \hat{y}_{t+1})\|^2, \\ L_{\text{disc}} &= \sum_t \|1 - D(H_t, x_t, y_t, y_{t+1})\|^2 \\ &\quad + \sum_t \|D(H_t, x_t, y_t, \hat{y}_{t+1})\|^2. \end{aligned} \tag{8}$$

5. 실험 결과

피격 반응 생성 시스템은 C++환경에서 제작되었으며, 물리 시뮬레이션은 Dynamic Animation and Robotics Toolkit(DART)라이브러리를 활용하여 구현하였다. 딥 러닝을 위한 네트워크과 관련 코드는 Python환경에서 Tensorflow를 기반으로 제작하였다. 피격 반응 생성 시스템으로부터 약 8시간 분량(83만 프레임, 30Hz)의 데이터를 학습용으로 생성하였으며, 하나의 동일한 학습 데이터셋으로 모든 네트워크를 약 24시간동안 학습하였다. 실험은 총 5가지의 네트워크로 수행되었다. 각 레이어의 수가 1024개인 fully-connected 모델과 2048개인 fully-connected 모델, mixture of experts 모델, RNN 모델 그리고 transformer 모델이다.

Figure 5는 학습된 네트워크로 부터 생성된 피격 동작 예제의 스크린 샷이다. 동일한 피격 조건에 대한 피격 반응을 실험했을 때 모든 네트워크가 입력에 반응해 적절한 피격 반응을 생성할 수 있었으며, 육안으로 확인 할때는 큰 차이를 구분하는 것이 어려웠다. 그러나 같은 조건임에도 완벽히 같은 반응이 생성되지 않고, 조금씩 달라지는 차이가 있었다. 피격 조건에 따라서 원본 데이터와의 유사도가 달라서 눈으로 보는 것만으로는 네트워크 간의 우열을 명확히 판단하기가 어려웠다.

Table 1는 각 네트워크 별 파라미터 개수와 연산 성능을 나타낸 표이고, Figure 6는 네트워크 모델별 러닝 커브 그래프이다. x축은 학습 시간이고, y축은 그에 따른 손실 값을 의미한다. 우선 fully-connected 모델 두개를 비교해보면, 노드의 수가 두배로 늘었을때, 거의 그 제곱에 비례해서 파라미터의 수가 증가하고 연산 속도가 느려지는것을 확인 할 수 있다. 이는 파라미터 수 증가에 따른 당연한 결과로써, Figure 6을 보면 대신 파라미터 수가 많은 fc 2048모델이 보다 빨리 그리고 정확히 학습하는 것을 확인 할 수 있다.

다른 세 종류의 모델을 비교해보면, mixture of experts 모델같

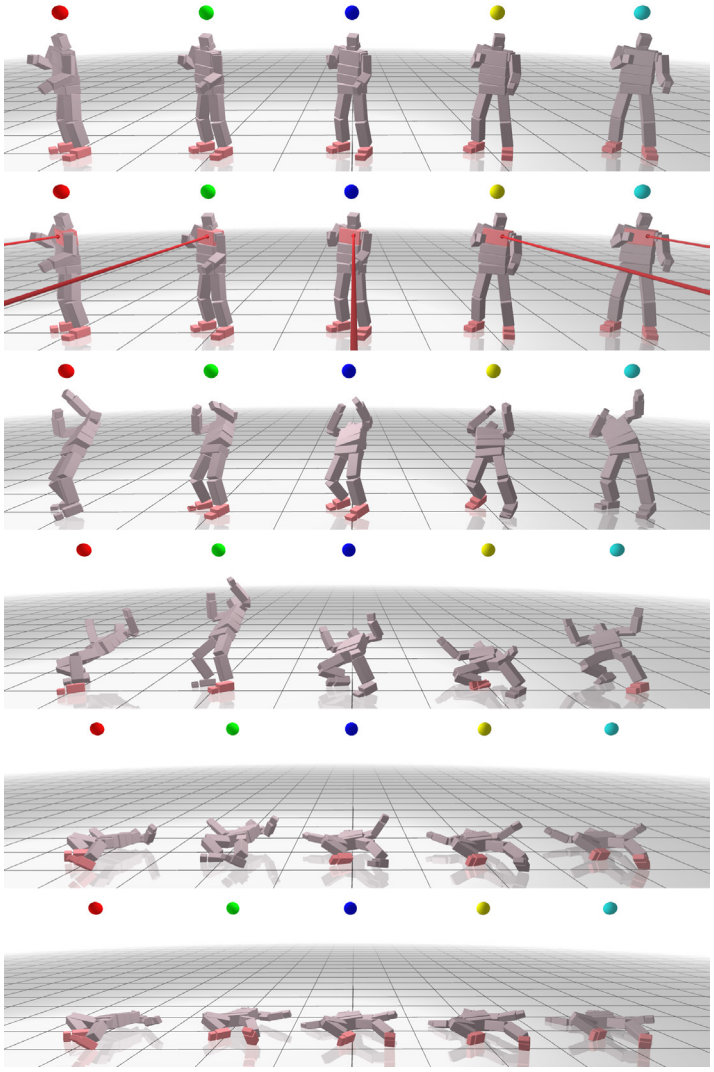


Figure 5: Generated motions from same hit conditions from each network, red : fc 1024, green : fc 2048, blue : moe, yellow : RNN, cyan : transformer.

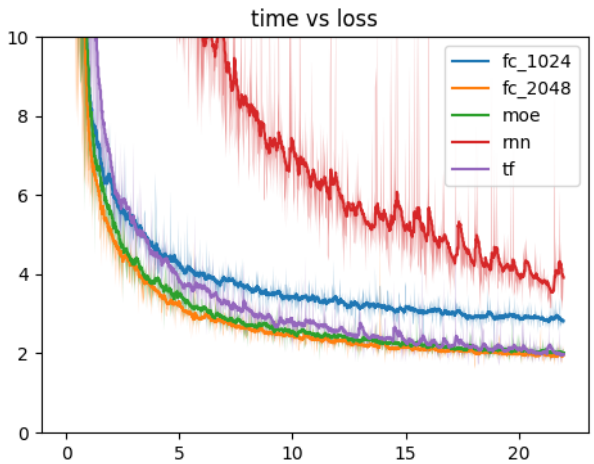


Figure 6: Learning curves, x-axis : time(h). y-axis : loss

은경우 fc를 제외한 세 모델중 가장 많은 파라미터가 필요하지만 가장 빠른 연산이 가능하며, RNN은 중간, transformer는 반대로 가장 적은 파라미터를 사용하지만 가장 속도가 느린것을 확인 할 수 있다. 이는 RNN과 transformer가 단순히 레이어간의 행렬곱 연산 이외에도 다른 부가적인 연산들을 많이 사용하고 있기 때문인 것으로 보인다. 학습 성능을 보면, mixture of experts 모델의 경우 fc 2048과 거의 유사한 학습 성능을 보이는 반면에, transformer모델의 경우 그보다 살짝 느린 학습 성능, rnn의 경우 매우 느린 학습 성능을 보이는 것을 확인 할 수 있다. 학습 정도와 성능, 파라미터 수를 종합적으로 고려하면, mixture of experts 모델이 파라미터 수 대비 성능과 학습 속도가 가장 뛰어나다고 할 수 있다.

6. 결론 및 향후 연구

본 연구에서는 다양한 모션을 생성 할 수 있는 피격 반응 생성 시스템을 구현하고, 이로부터 다양한 모션을 학습해야 하는 경우에 각 네트워크의 성능에 대한 비교를 진행했다. 총 네 가지 구조의 모델에 대해서 다섯 가지 네트워크로 실험을 진행하였고, 각 구조의 차이점과 장단점을 비교하였다.

네트워크의 성능은 전반적인 형태나 구조 뿐만 아니라 세부적인 다양한 종류의 파라미터들에도 큰 영향을 받는다. 본 논문에서는 이를 고정하고 특정한 5개의 케이스에 대해서 실험을 진행한 결과 mixture of experts 모델이 상대적으로 좋은 성능을 보인다는 결과를 얻었다. 하지만 각 네트워크에 맞게 파라미터 튜닝을 진행 할 경우, 결과가 달라 질 수도 있다. 예를 들어 본 연구에서 사용한 적대적 손실 항의 경우 디스크리미네이터와 우리의 네트워크 모델의 학습이 동시에 진행되기 때문에, 둘간의 학습 균형이 중요하다. 본 연구에서는 네트워크간의 일관된 비교를 위해 디스크리미네이터의 학습 속도(learning rate)를 고정한채로 실험을 진행하였고, 그 결과 RNN의 경우에는 디스크리미네이터의 학습 속도가 너무 빨라서 해당 손실 항이 적절한 역할을 하지 못하였다. 이처럼 네트워크 구조와 그 특성에 따라 적절한 파라미터 튜닝이 필요 할 수 있으며, 이점을 보완한다면 좀 더 정확하고 좋은 결과를 얻을 수 있을 것이라 기대한다.

본 연구에서는 네트워크의 학습 곡선과 파라미터 수, 그리고 연산 성능을 바탕으론 네트워크들을 비교하였다. 그러나 네트워크의 목적인 동작 생성을 기준으로 보다 정확히 평가하기 위해서는 생성된 동작의 자연스러움을 평가 할 수 있어야한다. 실제로 Figure 5과 같이 실제 네트워크를 이용해 동작을 생성해 볼 경우, 모든 네트워크가 전반적으로 자연스러운 동작을 생성하는 것을 확인 할 수 있었다. 피격 조건에 따라 특정한 네트워크가 상대적으로 부자연스럽거나 원본과 다른 동작을 생성하는 경우도 있었지만, 이 역시 피격 조건에 따라 그 부자연스러운 네트워크가 달라졌기 때문에 이를 바탕으로 네트워크를 비교하기가 어려웠다. 실제 게임속에서 네트워크를 사용하기 위해서는 결국 눈으로 보이는 품질이 중요하고, 따라서 이를 객관적으로 평가 할 수 있는 기준이 있다면 보다 실제 목적에 맞는 비교 결과를 얻을 수 있을 것이라 생각한다.

적절한 네트워크 모델을 고르는것은 딥러닝 기반 연구를 시작함에 있어서 매우 중요하다. 하지만, 네트워크 종류별로 최적의 성능을 내기 위해서는 많은 튜닝과 실험이 필요하고, 이로 인해 네트워크를 선택하는데에는 많은 노력이 필요하다. 이 논문이 딥러닝 기반 모션 생성을 위해 네트워크를 설계하고자 하는 연구자들에게 많은 도움이 될 수 있을 것이라 생각한다.

감사의 글

이 논문은 NCSOFT AI Center(NCSOFT2021-0767-K)와 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2017-0-00878, SW컴퓨팅산업원천기술개발산업(SW스타랩)).

References

[1] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard, “Interactive control of avatars animated with human motion data,” *ACM Trans. Graph.*, vol. 21, no. 3, pp. 491–500, 2002.

[2] L. Kovar, M. Gleicher, and F. Pighin, “Motion graphs,” *ACM Trans. Graph.*, vol. 21, no. 3, pp. 473–482, July 2002.

[3] M. Büttner and S. Clavet, “Motion matching - the road to next gen animation,” in *Proc. of Nucl.ai 2015*, 2015.

[4] S. Clavet, “Motion matching and the road to next-gen animation,” in *Proc. of GDC 2016*, 2016.

[5] K. Lee, S. Lee, and J. Lee, “Interactive character animation by learning multi-objective control,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–10, 2018.

[6] S. Lee, S. Lee, Y. Lee, and J. Lee, “Learning a family of motor skills from a single motion clip,” *ACM Trans. Graph.*, vol. 40, no. 4, 2021.

[7] D. Holden, T. Komura, and J. Saito, “Phase-functioned neural networks for character control,” *ACM Trans. Graph.*, vol. 36, no. 4, 2017.

[8] H. Zhang, S. Starke, T. Komura, and J. Saito, “Mode-adaptive neural networks for quadruped motion control,” *ACM Trans. Graph.*, vol. 37, no. 4, 2018.

[9] D. Holden, O. Kanoun, M. Perepichka, and T. Popa, “Learned motion matching,” *ACM Trans. Graph.*, vol. 39, no. 4, 2020.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[11] M. Flanders and P. Cordo, “Kinesthetic and visual control of a bimanual task: specification of direction and amplitude,” *Journal of Neuroscience*, vol. 9, no. 2, 1989.

[12] A. P. Georgopoulos, J. F. Kalaska, and J. T. Massey, “Spatial trajectories and reaction times of aimed movements: effects of practice, uncertainty, and change in target location,” *Journal of neurophysiology*, vol. 46, no. 4, 1981.

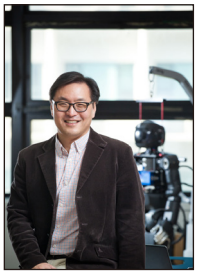
[13] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2794–2802.

〈저자소개〉

박수환

- 2015 : 서울대학교 전기정보공학부 학사
- 2015~현재 :서울대학교 컴퓨터공학부 석박통합과정
- 관심분야 : 물리 기반 캐릭터 제어, 데이터 기반 애니메이션
- https://orcid.org/0000-0002-0531-5859



이제희

- 1993 : 한국과학기술원 전산학 학사
- 1995 : 한국과학기술원 전산학 석사
- 2000 : 한국과학기술원 전산학 박사
- 2003~현재 : 서울대학교 컴퓨터공학부 교수
- 관심 분야 : 컴퓨터 그래픽스, 애니메이션, 생체역학, 로보틱스, 의료보행분석
- https://orcid.org/0000-0003-1023-1868