비디오 트랜스포머 연구 동향

조 현 중

고려대학교

Ⅰ. 서 론

2017년에 소개된 트랜스포머(transformer)는 자연어처리 분야에서 일반적으로 사용되어오던 RNN, LSTM 등 기존의 순차적인 계산 모델을 병렬화한 모델로 해당 분야에서 주목할만한 성능 향상을 보였다[1]. 이후 트랜스포머는 BERT, GPT 등 LLM(large language models)의 핵심 요소로 도입되어 자연어처리 분야의 혁신적 성과를 이끌어 오고 있다[2],[3]. 이러한 트랜스포머의 성공적 확산은 트랜스포머의 핵심 요소인 셀프 어텐션(self-attention, SA) 구조가 자연어의 원격 의존성(long-term dependency)의 특성을 기존의 순차적인 계산 모델에 비해 효과적으로 표현할 수 있기 때문이다. 최근에는 이러한 원격 의존성은 자연어뿐만 아니라, 이미지의 픽셀 사이에서도 발견될 수 있는 특징임에 주목하여 트랜스포머를 비전 분야에 활용하려는 시도가 자연스럽게 시작되고 있다.

한편, 컴퓨터 비전 분야에서는 2012년 AlexNet이 소개된 이후로 CNN 기반의 모델들이 지배적으로 사용되어 왔다[4]. CNN의 컨볼루션 필터는 비전 데이터의 지역성(locality)과 입력 데이터의 평행 이동에도 출력 라벨의 변화가 없는 특성(translation invariance)을 효과적으로 표현할 수 있는 구조로 알려져 있다. 하지만 컨볼루션 필터의 좁은 수용 영역(receptive field)은 입력 데이터의 지역적인 특성을 효과적으로 추출할 수 있지만, 넓은 영역으로부터 입력 특성을 추출하는데 불리하게 작용한다. 이 때문에 많은 연구자들이 CNN 모델의 한계를 넘어서기 위한 하나의 방법으로 트랜스포머를 주목하고 있다. 즉, 자연어 처리를 위해 고안되었던 트랜스포머를 컴퓨터 비전 데이터 처리를 위한 CNN 모델의 대안으로 활용하고자 하는 시도가 시작되고 있다.

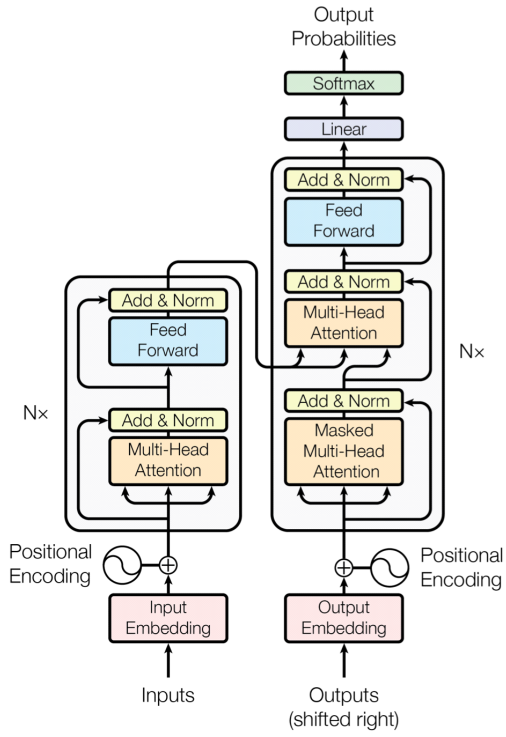
컴퓨터 비전 데이터 중 비디오 데이터는 가로축과 세로축으로 이루어진 2차원 공간에 픽셀값을 가지는 이미지가 시간축에 따라 변형되는 시공간(spatio-temporal) 데이터이다. 비디오 데이터는 시간에 따라 순차적으로 변화하고, 인접한 프레임 사이의 중복성이 높으며, 가로 세로뿐만 아니라, 시간축을 가지는 특성이 있다. 즉, 비디오 데이터는 2차원 이미지가 가지는 지역성과 함께 시간축을 따라 변화하는 순차적 데이터 특성도 갖고 있어서, 이미지 처리에 사용되는 딥러닝 모델을 단순 확장하여 적용하는 것을 넘어 트랜스포머를 변형하여 활용하고자 하는 다양한 시도들이 발표되고 있다.

본고에서는 비디오 데이터 처리를 위한 최근 비디오 트랜스포머(VT) 기술들을 살펴보고, 장단점을 비교한다. 본고에서 논의하는 주요 기술로는 TimeSformer, ViViT, Deformable Video Transformer, MotionFormer들이 있다. 또, 비디오 데이터와 같이 매우 큰 입력 데이터를 처리하기 위해 제안된 트랜스포머를 살펴보고, 비디오 데이터 적용 가능성을 고찰한다. 특히 트랜스포머의 핵심 기술인 각 모델의 셀프 어텐션 구조를 구체적으로 살펴보고, 향후 비디오 트랜스포머의 발전 방향을 전망하고자 한다.

Ⅱ. 트랜스포머 개요

트랜스포머는 RNN, LSTM 등 기존의 순차적인 모델을 병렬화한 모델로 셀프 어텐션, SA을 주요 요소 기술로 포함한다. SA는 트랜스포머의 입력인 토큰(Token) 배열에 포함된 개별 토큰의 특징 벡터를 표현하기 위해 다른 토큰들과 원격 의존성을 고려하기 위한 수단이다. 트랜스포머는 인코더와 디코더로 구성되고, 인코더는 다수의 인코더 레이어의 계층 구조로, 디코더는 다수의 디코더 레이어의 계층 구조로 이루어져 있다. [그림 1]은 트랜스포머의 구조를 나타낸다.

인코더 레이어는 멀티헤드 셀프 어텐션(multi-head self attention, MHSA), 레이어 정규화(layer norm, LN), 피드포워드 신경망(feed forward network, FFN)으로 구성되어 있다. 디코더 레이어는 인코더의 구성 요소와 함께 마스크 멀티헤드 셀프 어텐션(masked MHSA)를 포함하고 있다. [그림 2]는 하나의 인코더 레이어를 수식화한 것이다.



[그림 1] 트랜스포머 구조[1]

[A Single Layer] $f_\theta(\boldsymbol{x}) = \boldsymbol{z}$, $f_\theta: R^{t_{max} \times d} \rightarrow R^{t_{max} \times d}$

- $\boldsymbol{x} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_{t_{max}}}\}$, $t_{max}: max.seq.length$
- $d: in/out\ dim.$
- $d_h: hidden\ dim.$
- $d_f: hidden\ dim.\ of\ linear\ layer$

[MHSA]

- $Q^{(h)}(\boldsymbol{x_i}) = W_q^{(h)T} \boldsymbol{x_i}$
- $K^{(h)}(\boldsymbol{x_i}) = W_k^{(h)T} \boldsymbol{x_i}$,
- $V^{(h)}(\boldsymbol{x_i}) = W_v^{(h)T} \boldsymbol{x_i}$, $W_q^{(h)}, W_k^{(h)}, W_v^{(h)} \in R^{d \times d_h}$

- $\alpha_{i,j}^{(h)} = softmax_j(\frac{<Q^{(h)}(x_i),\ K^{(h)}(x_j)>}{\sqrt{d_h}})$.
- $\boldsymbol{u_i}' = \sum_{h=1}^{H} W_c^{(h)T} \left(\sum_{j=1}^{n} \alpha_{i,j}^{(h)} V^{(h)}(\boldsymbol{x_j}) \right)$, $W_c^{(h)} \in R^{d_h \times d}$,

[Add&LN]

- $\boldsymbol{u_i} = LayerNorm(\boldsymbol{x_i} + \boldsymbol{u_i'}: \gamma_1, \beta_1)$

[Point-wise FFN]

- $\boldsymbol{z_i'} = W_2^T Relu(W_1^T \boldsymbol{u_i})$, $W_1 \in R^{d \times d_f}, W_2 \in R^{d_f \times d}$,

[Add&LN]

- $\boldsymbol{z_i} = LayerNorm(\boldsymbol{u_i} + \boldsymbol{z_i'}: \gamma_2, \beta_2)$

[그림 2] 트랜스포머의 인코더 수식

입력 배열 $\boldsymbol{x} = \{\boldsymbol{x_1}, \cdots, \boldsymbol{x_{t_{max}}}\}$의 각 입력 $\boldsymbol{x_i}$은 MHSA 단계에서 셀프 어텐션 연산을 통해 다른 토큰과의 원격 의존성에 따른 벡터$\boldsymbol{u'_i}$로 변환된다. $\boldsymbol{u'_i}$는 LN, FFN 등을 거쳐 인코더 레이어의 출력인 특징 벡터$\boldsymbol{z_i}$로 변환된다.

Ⅲ. 비디오 트랜스포머

트랜스포머를 비디오 데이터에 적용하기 위해서는 공간축과 시간축을 가지는 비디오 데이터를 위한 SA를 어떻게 설계해야 하는지에 대한 고민이 필요하다. 본 장에서는 VT의 입력을 위한 비디오 데이터의 전처리 방법과 함께 다양한 VT 설계 사례를 MHSA를 중심으로 설명한다.

3-1 전처리

VT의 입력을 위해 비디오 데이터의 적절한 전처리가 필요하다. 일반적으로 비디오 데이터는 Tokenization, Embedding, 그리고 Positional Embedding(PE)의 전처리를 거친다[5].

Tokenization은 입력 배열을 만들기 위해 비디오 데이터를 토큰이라는 작은 단위로 나누는 것을 의미하는데, 토큰을 만드는 방법에 따라 Patch-wise, Instance-wise, Frame-wise, Clip-wise로 분류할 수 있다. 일반적으로 사용되는 방법은 Patch-wise 방법으로 비디오 데이터를 2D Patch나 3D Patch 단위로 나누어 입력 배열을 구성한다. 2D Patch는 비디오 데이터의 각 프레임을 예를 들어 16 × 16 해상도의 타일로 잘게 나누는 것을 말하고, 3D Patch는 시간축의 변화를 함께 고려하기 위해서 16 × 16 × 2 등으로 잘게 나누는 것을 말한다.

Embedding은 각 토큰들을 특징 벡터로 변환하는 것을 말한다. Embedding 방법은 공간적인 특징을 벡터화하는 방법(spatial embedding)과 시공간적인 특징을 벡터화하는 방법(spatio-temporal embedding), 차원 변형 등 최소한의 변환만을 하는 방법(minimal embedding)으로 분류할 수 있다. Spatial Embedding을 위해 2D CNN을, Spatio-temporal Embedding을 위해 3D CNN을, Minimal Embedding을 위해 선형 변환 등을 사용할 수 있다.

Tokenization과 Embedding을 거친 입력 배열에 각 요소의 순서 정보를 담기 위해 PE을 적용한다. 순서 정보는 일반적으로 Sinusoidal 신호를 바탕으로 표현되며, 특별히 비디오 데이터를 위해 2D 혹은 3D로 확장되기도 한다. 절대적(absolute)으로 표현되는 이러한 PE와는 달리, 토큰끼리의 상대적인

(relative) 거리를 학습을 통해 결정하려는 PE 방법도 제안된 바 있다. 이러한 방법은 pixel 기반의 이미지뿐만 아니라, 그래프 등 다양한 입력 형태에 적용이 가능하다[6]([그림 3]).

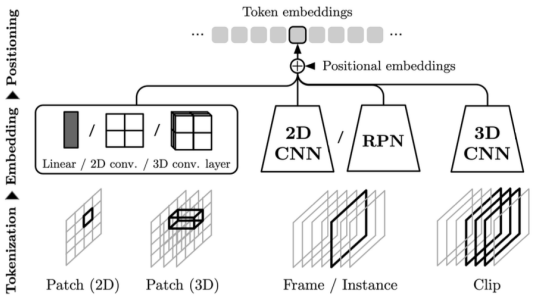
3-2 TimeSformer

TimeSformer는 비디오 데이터를 CNN 구조 없이 트랜스포머 기반만으로 처리하기 위해 제안되었다[7]. 이미지 데이터를 처리하기 위해 고안된 ViT와 마찬가지로 비디오 데이터의 각 프레임을 $P\times P$크기의 Patch로 서로 겹치지 않도록 Tokenization을 수행한다. 즉, 하나의 비디오 프레임은 $N=HW/P^2$개의 Patch로 나누어진다(H:높이, W:넓이). 공간 좌표 p와 시간 좌표 t의 Patch $x_{(p,t)}$는 아래 식과 같이 선형 변환 E와 PE $e^{pos}_{(p,t)}$로 위치 정보를 담은 후에 TimeSformer에 입력된다.

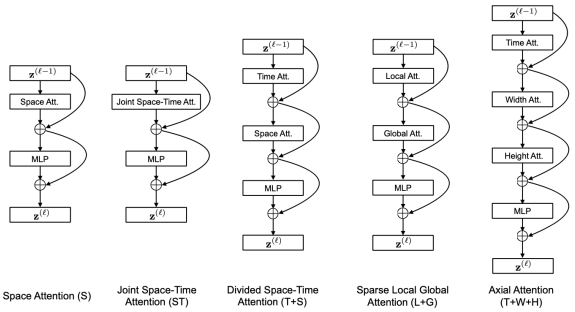
$$z_{(p,t)}=Ex_{(p,t)}+e^{pos}_{(p,t)} \quad (1)$$

TimeSformer는 인코더 레이어 (l)층에 $z^{(l-1)}_{(p,t)}$를 입력하고, 비디오 데이터를 위해 설계된 인코더 레이어는 적절한 Self-Attention을 적용한 후 $z^{(l)}_{(p,t)}$을 출력한다. TimeSformer는 다섯 가지의 SA 방법을 [그림 4]와 같이 제안하고 있다.

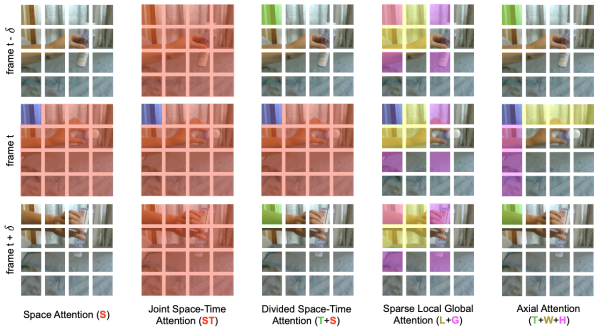
또, [그림 5]는 각 SA 방법이 시공간 축에 따라 어떤 형태로 Patch 사이의 의존성을 계산하는지를 직관적으로 나타내고 있다. 특히, TimeSformer의 대표적인 SA 방식은



[그림 3] 트랜스포머 입력을 위한 Tokenization, Embedding, PE 전처리 과정[5]



[그림 4] TimeSformer가 제안하는 다섯가지 SA 방법[7]



[그림 5] TimeSformer SA의 도식화. 파란색은 Query Patch, 그 밖에 색은 Attention을 위해 짝을 이루는 Patch를 나타냄.

Joint Space-Time(ST) Attention과 Divided Space-Time(T+S) Attention이다. ST Attention 방법은 일정 시간 범위 안에 있는 모든 프레임의 모든 Patch들 간의 의존성을 계산하는 방법인 반면, T+S Attention 방법은 일정 시간 범위 안에 있는 공간 p위치의 모든 Patch들의 의존성을 계산하고, 순차적으로 시간 t 프레임의 모든 Patch들에 대한 의존성을 계산한다.

ST Attention 방식은 일정 시간 범위 안에 있는 모든 Pacth들을 고려하므로 데이터가 충분히 있을 경우 TimeSformer가 적절한 Attention을 수행할 가능성이 높지만, SA의 계산량이 입력 데이터 길이 n에 대해 $O(n^2)$임을 고려하면 모든 Patch들을 고려하기 위해 많은 계산량이 필요함을 알 수 있다. 반면, T+S Attention은 계산량을 효과적으로 줄일 수 있지만, 두 프레임에서 다른 위치에 있는 Patch들 간의 의존성을 알 수 없다는 단점이 있다. TimeSformer 이후의 VT들은 연산 복잡도를 적절히 줄이면서도 비디오 데이터에 알맞은 다양한 SA 방법을 제안하고 있다.

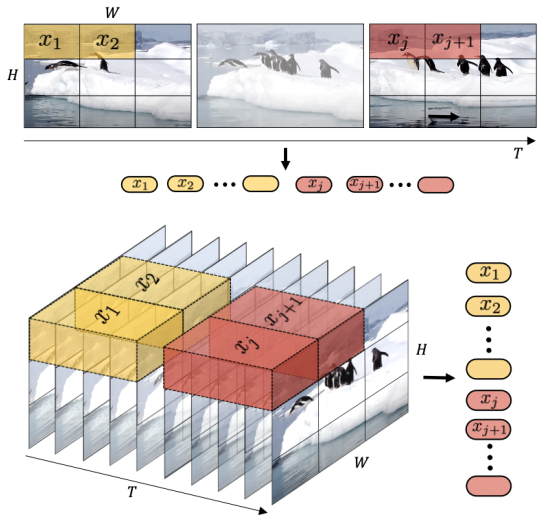
3-3 ViViT

ViViT은 TimeSformer의 2D Tokenization과는 달리 시공간 정보를 함께 담기 위한 Tubelet Embedding을 [그림 6]과 같이 제안했다[8]. Tubelet Embedding은 3D Tube 모양을 따라 Tokenization 후 선형 변환을 통해 Embedding을 완성한다. ViViT 저자들은 Tubelet Embedding이 TimeSformer의 2D Patch Embedding보다 좋은 성능을 보임을 실험적으로 증명하였다.

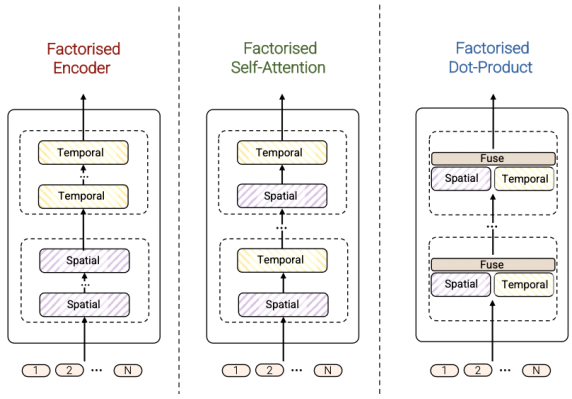
ViViT는 [그림 7]과 같이, Factorized Encoder, Factorized SA, Factorized Dot-Product 등, 비디오 데이터를 위한 세 가지 SA 모델을 제안하였다. 그리고 실험을 통해 Factorized Encoder 모델이 다른 제안 기술보다 성능이 우수함을 보였다.

Factorized Encoder 모델은 [그림 8]과 같이 두 개의 개별 인코더(공간 및 시간)로 구성된다. 비디오 데이터의 각 프레임은 Spatial Transformer를 통해 특징 벡터로 변환되고, 시간 축으로 배열된 특징 벡터들은 Temporal Transformer로 전달된다. 이 모델은 TimeSformer의 ST Attention 방법보다 트랜스포머 레이어가 더 많지만(따라서 파라미터가 더 많지만), $O(n^2)$복잡도를 가지는 SA 연산의 입력을 시공간을 분리해 진행함으로써 부동 소수점 연산(FLOP)을 줄일 수 있다.

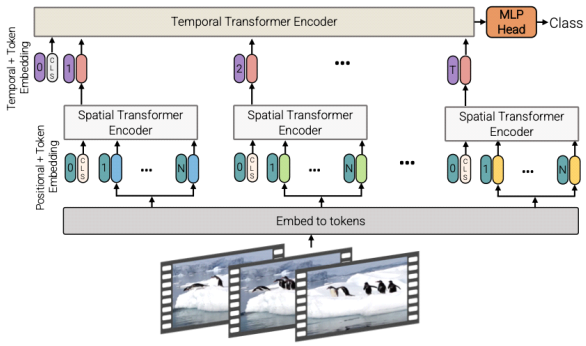
[그림 9]는 ViViT 저자들이 Kinetics-600 데이터셋을 사용하여 실험한 성능 측정값으로 ViViT와 TimeSformer가 좋은 성능을 보임을 알 수 있다[9].



[그림 6] ViViT의 2D Embedding과 3D Tubelet Embedding 비교[8]



[그림 7] ViViT의 세 가지 SA 방법[8]



[그림 8] ViViT의 Factorized Encoder SA[8]

Method	Top 1	Top 5
AttentionNAS [76]	79.8	94.4
LGD-3D R101 [51]	81.5	95.6
SlowFast R101-NL [21]	81.8	95.1
X3D-XL [20]	81.9	95.5
TimeSformer-L [4]	82.2	**95.6**
ViViT-L/16x2 FE	**82.9**	94.6
ViViT-L/16x2 FE (JFT)	84.3	94.9
ViViT-H/14x2 (JFT)	**85.8**	**96.5**

[그림 9] ViViT와 TimeSformer의 성능 비교(Kinetics-600)

3-4 Motionformer

트랜스포머는 CNN에 비해 Inductive Bias가 없기 때문에 충분한 성능을 얻기 위해 매우 많은 양의 데이터가 필요하

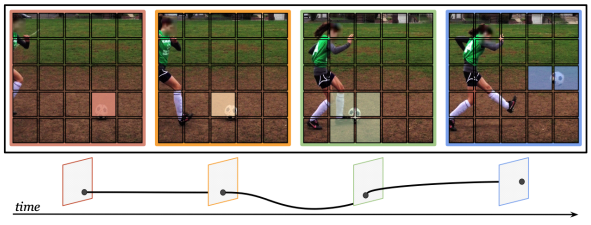
다고 알려져 있다. 비디오 데이터를 처리하기 위한 VT의 경우에 이와 같은 트랜스포머의 특성은 많은 양의 데이터를 처리하기 위해 높은 연산량을 요구함을 의미한다.

트랜스포머의 연산량을 줄이기 위한 방편으로 비디오 데이터의 독특한 특성을 주목한 논문들이 있는데, 그중 하나가 Motionformer이다[10]. 비디오는 풍부한 시각적 정보를 담고 있지만, 인접 프레임간의 중복된 정보가 많은 편이다. 따라서, 모든 비디오 프레임의 모든 Pixel 혹은 모든 Patch를 고려하기보다 움직임이 발생하는 Pixel 혹은 Patch만을 고려하는 것이 연산량 경감에 있어 효율적일 수 있다.

[그림 10]은 Kinetics-400 데이터 중 '축구공 차기' 동작으로 프레임 배열에서 공은 카메라에 대해 고정된 상태로 있지 않고 프레임마다 다른 위치로 이동함을 알 수 있다. Motionformer는 프레임간의 변화 즉, 공의 움직임 경로를 따라 SA을 적용하는 것, 즉 Trajectory Attention (TA)을 목표로 한다. 이러한 방법은 TimeSformer의 ST Attention 보다 비디오 데이터에 대해 자연스러운 Inductive Bias를 제공할 수 있다.

Motionformer은 3D Tubelet Embedding을 사용하고, 시간축과 공간축의 PE를 별도로 적용하여 인코더의 입력을 구성한다. Motionformer의 TA는 두 프레임 간의 Attention을 구하여 Trajectory Token을 구하는 1단계와 구해진 Trajectory Token들의 1D Attention을 구하는 2단계로 구성된다.

1단계에서 Trajectory Token $\tilde{y}_{stt'}$는 프레임 t의 s공간에 있는 특징 벡터와 프레임 t'의 모든 특징 벡터와의 의존성을 나타내며, 이를 구하기 위한 수식은 아래와 같다.



[그림 10] Trajectory Attention의 예시. Kinectics-400 데이터셋에 포함되어 있는 'kicking soccer ball" 동작으로 인접한 프레임들 사이에서 공의 위치가 공간적으로 고정되어 있지 않음을 알 수 있음[10].

$$\tilde{y}_{stt'} = \sum_{s'} v_{s't'} \frac{\exp < q_{st}, k_{s't'} >}{\sum_{\bar{s}} \exp < q_{st}, k_{\bar{s}t'} >} \quad (2)$$

위 식에서 $\boldsymbol{q_{st}}$는 프레임 t의 s위치에 있는 특징 벡터의 Query, $\boldsymbol{k_{s't'}}$와 $\boldsymbol{v_{s't'}}$는 프레임 t'의 s'위치에 있는 특징 벡터의 Key와 Value를 나타낸다. 이렇게 구해진 $\tilde{y}_{stt'}$는 t'축을 따라 1D 배열을 이루게 되는데, 이 배열에 대해 2단계에서 1D Attention을 아래 식과 같이 적용한다.

$$y_{st} = \sum_{s'} \tilde{v}_{stt'} \frac{\exp < \tilde{q}_{st}, \tilde{k}_{stt'} >}{\sum_{\bar{t}} \exp < \tilde{q}_{st}, \tilde{k}_{st\bar{t}} >} \quad (3)$$

위 식에서 $\tilde{\boldsymbol{q}}_{\boldsymbol{st}}$는 $\tilde{y}_{stt}$의 선형변환으로 구한 Query, $\tilde{k}_{stt'}$와 $\tilde{v}_{stt'}$는 $\tilde{y}_{stt'}$의 선형변환으로 구한 Key와 Value를 의미한다. 하지만 Trajectory Attention은 계산 복잡도가 TimeSformer의 ST Attention과 같은 단점이 있다.

Motionformer의 저자들은 Trajectory Attention의 연산 복잡도를 경감하기 위해 이를 근사하는 Orthoformer 알고리즘도 함께 제안했다. Orthoformer의 SA는 다음과 같은 근사식으로 설명할 수 있다.

$$softmax(QK^T/\sqrt{d_h}) \approx softmax(QP^T/\sqrt{d_h})softmax(PK^T/\sqrt{d_h}) \quad (4)$$

이 식에서 $Q, K, V \in R^{N \times d_h}$, $P \in R^{R \times d_h}$이다. 즉, N 크기의 입력 배열을 대표할 수 있는 Prototype R개를 선정함으로써 기존 SA 계산 복잡도를 $O(n^2)$에서 $O(n)$으로 감소시킨다. Orthoformer의 성능은 Prototype R개를 선정하는 방식에 의존적인데, 저자는 새롭게 선정하는 Prototype이 기존의 Prototype들과 최대한 직교할 수 있도록 선택하여 선정된 Prototype들이 최대한 겹치지 않도록 하였다.

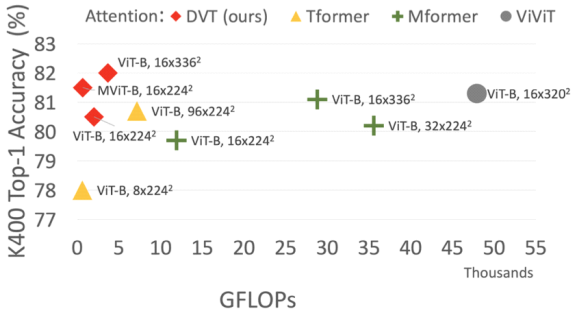
3-5 Deformable Video Transformer

Deformable Video Transformer (DVT) 역시 MotionFormer

와 마찬가지로 VT의 높은 계산 복잡도에 주목하고, 비디오 데이터의 동적 특성을 이용하여 특정 영역에 Attention을 제한적으로 줌으로써 계산 복잡도 경감을 시도하였다[11]. DVT는 MotionFormer와는 달리 비디오 압축 기술에서 사용되는 Motion Displacement 정보를 활용한다. Motion Displacement는 인접한 비디오 프레임사이의 차이를 말하고, 이는 두 프레임 사이의 움직임을 나타낸다. 즉, 기존 비디오 압축 기술에서 이미 계산된 Motion Displacement를 활용하여 추가적인 연산 처리 없이 모션 영역을 구하고, 이를 SA에 활용하고자 하는 것이 목적이다. 저자들은 MotionFormer에 비해 추가적인 계산없이 Trajectory Attention을 할 수 있음을 주장하면서, Kinetics-400 데이터셋을 활용한 동작 인식 실험에서 계산량 대비 인식 정확도가 다른 VT에 비해 높음을 실험적으로 [그림 11]과 같이 보였다.

구체적으로 DVT는 Deformable Space-Time Attention (D-ST-A)을 제안하였다. D-ST-A는 t시간 s위치의 Query $\boldsymbol{q}_s^t$에 대해 t' 시간 프레임마다 N개 위치 $\{s(n)\}_{n=1}^{N}$에 Attention을 적용한다. D-ST-A의 계산 복잡도는 기존 ST Attention $O(n^2)=O(S^2T^2)$에서 $O(SNT^2)$로 감소한다. 여기서 S는 프레임별 Token의 개수, T는 프레임의 개수를 나타낸다. 다음은 t시간 s위치의 Token z_s^t에 대한 특징 벡터 z'^t_s를 구하기 위한 식이다.

$$\boldsymbol{z'}^t_s = \boldsymbol{z}_s^t + \sum_{t'=1}^{T}\sum_{n=1}^{N} \alpha_{s(n)}^{t,t'} \boldsymbol{v}_{\boldsymbol{s(n)}}^{\boldsymbol{t'}} \tag{5}$$



[그림 11] Kinetics-400 데이터셋을 활용한 성능 비교. DVT는 Deformable Video Transformer, Tformer는 TimeSformer, Mformer는 MotionFormer를 나타냄[11].

위 식에서 $\alpha_{s(n)}^{t,t'}$는 D-ST-A로 구해진 Attention Score를 의미한다. $\alpha_{s(n)}^{t,t'}$는 Query $\boldsymbol{q}_s^t$와 Motion Displacement $\boldsymbol{m}_s^{t,t'}$의 아래 두 식과 같은 선형결합으로 계산된다.

$$\hat{\alpha}_s^{t,t'} = W_\alpha(\boldsymbol{q}_s^t + \boldsymbol{m}_s^{t,t'}) \tag{6}$$

$$\alpha_{s(n)}^{t,t'} = \frac{\exp\hat{\alpha}_{s(n)}^{t,t'}}{\sum_{t'=1}^{T}\sum_{n'=1}^{N}\hat{\alpha}_{s(n')}^{t,t'}} \tag{7}$$

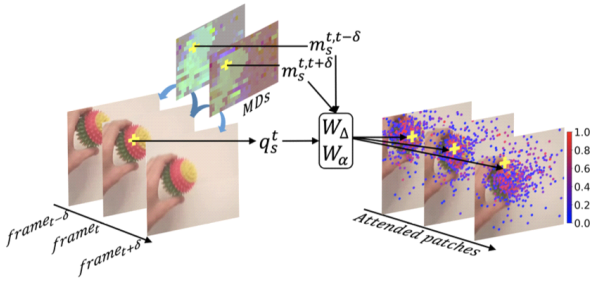
또 $\boldsymbol{v}_{\boldsymbol{s(n)}}^{\boldsymbol{t'}}$역시 Motion Displacement $\boldsymbol{m}_s^{t,t'}$를 활용하여 아래 두 식과 같이 계산된다.

$$\boldsymbol{p}(s(n)) = \boldsymbol{p}(s) + \Delta_s^{t,t'}(n) \tag{8}$$

$$\Delta_s^{t,t'}(n) = W_\Delta(\boldsymbol{q}_s^t + \boldsymbol{m}_s^{t,t'}) \tag{9}$$

위 식에서 $\boldsymbol{p}(s)$는 Token s의 위치이다. 즉, $\Delta_s^{t,t'}(n)$는 t 프레임의 s위치와 t'프레임의 $s(n)$ 사이의 위치 차이를 나타낸다. $\boldsymbol{v}_{\boldsymbol{s(n)}}^{\boldsymbol{t'}}$는 $\boldsymbol{p}(s(n))$을 얻은 후 bilinear interpolation을 통해 계산된다. [그림 12]는 D-ST-A을 나타낸다.

종합하면 D-ST-A는 Query $\boldsymbol{q}_s^t$와 비디오 압축 기술로 이미 계산된 Motion Displacement $\boldsymbol{m}_s^{t,t'}$로 Attention Score와 Value



[그림 12] D-ST-A의 시각화. 노란색 표시는 Query Patch를 나타내고, 각 프레임으로부터 motion displacements 정보를 활용하여 N=8개의 Patches를 추출하여 활용함[11].

값을 구한 후 해당하는 특징 벡터를 계산하는 방식이다. 저자들은 이러한 방식으로 Motionformer보다 적은 연산량으로 비디오 데이터에 효율적인 SA를 적용할 수 있다고 주장한다.

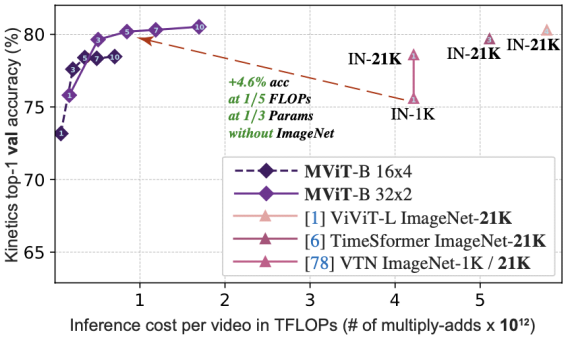
3-6 Multiscale Vision Transformer

Multiscale Vision Transformer(MViT)는 다양한 Scale로 계층화된 특징 벡터, 즉 특징 벡터 피라미드(Pyramid)를 트랜스포머에서 사용하기 위한 목적으로 설계되었다[14]. 특징 벡터 피라미드를 사용하는 목적은 두 가지인데, 첫째, 낮은 해상도의 특징 벡터를 활용하여 연산량을 줄이고, 둘째, 낮은 해상도의 특징 벡터를 통해 데이터의 맥락을 추출하고, 이를 이용하여 높은 해상도의 특징 벡터를 처리하기 위한 가이드를 얻기 위함이다.

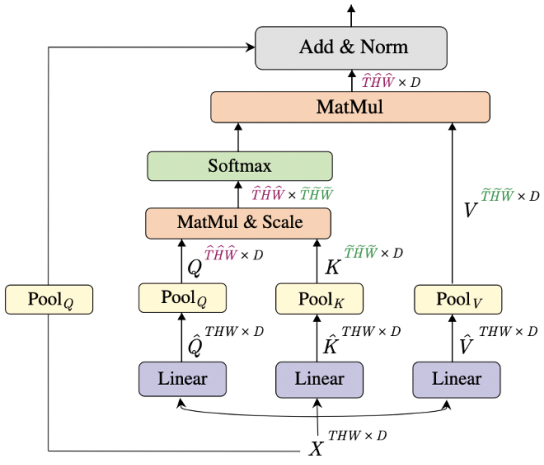
MViT는 입력이 출력으로 변환하는 동안 Channel의 수는 늘리고, 해상도는 줄이는 방향으로 동작하도록 설계되었다. [그림 13]은 MViT를 Kinectics-400 동작 인식 데이터셋에 적용했을 때 동작인식 성능을 나타내고 있는데, ViViT 및 TimeSformer에 비해 적은 연산량으로도 높은 정확도를 보이고 있음을 알 수 있다.

MViT은 Multi-Head Pooling Attention (MHPA)를 [그림 14]와 같이 수행한다. 입력 X는 먼저 선형변환과 Pooling을 거쳐 Query, Key, Value로 아래 식과 같이 변환된다.

$$
\begin{aligned}
Q &= Pool(XW_Q;\Theta_Q),\ X\in R^{L\times D},\ Q\in R^{\hat{L}\times D}\\
K &= Pool(XW_K;\Theta_K),\ K\in R^{\tilde{L}\times D}\\
V &= Pool(XW_V;\Theta_V),\ V\in R^{\tilde{L}\times D}
\end{aligned}
\tag{10}
$$



[그림 13] Kinectics-400 데이터셋을 활용한 정확도와 연산량의 Tradeoff 그래프[14]



[그림 14] Pooling Attention 연산 과정[14]

위 식에서 $L=THW$로 입력 배열의 길이를 나타내고, Θ는 Pooling 연산에 필요한 파라메터로 커널 크기, 패딩, 스트라이드를 포함한다. Pooling 연산으로 인해 MHPA의 연산량은 기존 MHSA에 비해 효과적으로 감소한다.

변환된 Q, K, V는 아래 식을 통해 Pooling Attention을 계산하는데 사용된다.

$$
PA = softmax(QK^T/\sqrt{D})V \tag{11}
$$

MViT은 다수의 MHPA와 MLP(Multi-Layer Perceptron)을 포함한 Stage를 직렬로 연결하여 전체 네트워크를 구성한다. 입력 X는 Stage를 거칠 때마다 Channel 수는 증가하고, 해상도는 줄어드는 특징 벡터로 변환된다.

Ⅳ. 긴 입력 배열을 위한 트랜스포머

Ⅲ장의 논의에서 다양한 VT들의 가장 큰 도전은 연산량을 줄이면서도 데이터의 원격 의존성을 효과적으로 표현할 수 있는 SA의 설계임을 알 수 있다. 특히 비디오 데이터는 응용 분야에 따라 시간(hour) 단위로 길어질 수 있으므로 효과적인 SA의 설계는 매우 중요하다.

사실 매우 긴 데이터 배열을 위한 트랜스포머의 설계는 트랜스포머 자체의 숙제이기도 하다. 만약 매우 긴 데이터 배열을 효과적으로 다룰 수 있는 트랜스포머가 있다면, 이

를 비디오 데이터 처리를 위해 적용하는 것은 어려운 일이 아닐지도 모른다. 본 장에서는 매우 긴 입력 배열을 다루기 위한 트랜스포머를 살펴봄으로써 비디오 데이터로의 적용 가능성을 타진해 본다.

4-1 Nyströmformer

Nyströmformer의 저자들은 기존 트랜스포머의 SA 수식을 주목하고, 이를 수학적으로 근사하려는 시도를 하였다[12]. 기존 SA식에서 연산량이 가장 많은 부분이 Attention Score를 구하는 부분인데 이를 아래 식과 같이 분할 행렬로 나타낸다.

$$S = softmax(\frac{QK^T}{\sqrt{d}}) = \begin{bmatrix} A_s & B_s \\ F_s & C_s \end{bmatrix} \tag{12}$$

분할 행렬 중 부분 행렬의 차원은 $A_s \in R^{m \times m}$, $B_s \in R^{m \times (n-m)}$, $F_s \in R^{(n-m) \times m}$이다. 분할 행렬은 아래와 같이 Nyström 근사식으로 표현이 가능하다.

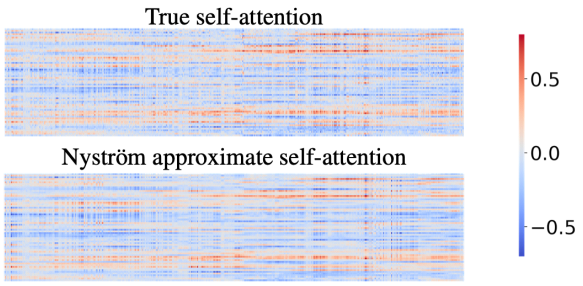
$$\hat{S} = \begin{bmatrix} A_s & B_s \\ F_s & F_s A_s^+ B_s \end{bmatrix} = \begin{bmatrix} A_s \\ F_s \end{bmatrix} A_S^+ \begin{bmatrix} A_s & B_s \end{bmatrix} \tag{13}$$

위 식에서 A_s^+는 A_s의 Moore-Penrose Inverse에 해당한다. 위 식을 이용하여 Attention Score 식을 아래와 같이 근사할 수 있다.

$$\hat{S} = \left[softmax(\frac{Q\tilde{K}^T}{\sqrt{d}}) \right] \left[softmax(\frac{\tilde{Q}\tilde{K}^T}{\sqrt{d}}) \right]^+ \left[softmax(\frac{\tilde{Q}K^T}{\sqrt{d}}) \right] \tag{14}$$

위 식에서 $\tilde{Q} = [\tilde{q}_1; \cdots; \tilde{q}_m] \in R^{m \times d}$, $\tilde{K} = [\tilde{k}_1; \cdots; \tilde{k}_m] \in R^{m \times d}$를 나타낸다. 즉, 기존 Q에서 m개의 landmark 행을 선택하여 $\tilde{Q}$를 구성하고, 기존 K에서 m개의 landmark 행을 선택하여 $\tilde{K}$를 구성한다.

저자에 따르면 단순히 행렬을 64개의 행 그룹으로 나누고, 각 그룹의 평균을 landmark로 선택하는 것만으로 만족할 만한 성능을 얻을 수 있다고 한다. [그림 15]는 실제 SA값과 근사값의 비교를 시각화한 것으로 두 값들이 비슷하게 계산됨을 보인다. 연산량도 만약 m이 n보다 현저하게 작다면 $O(n)$임을 보였다.



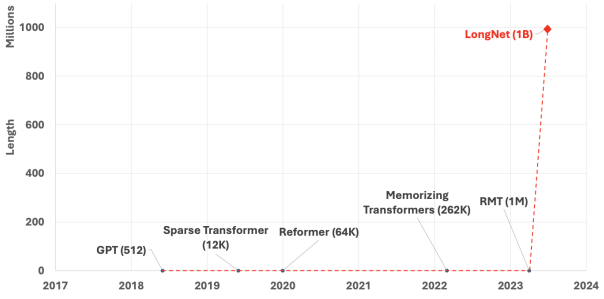
[그림 15] Ground Truth SA Score와 Nyström 근사값의 비교 시각화[12]

4-2 LongNet

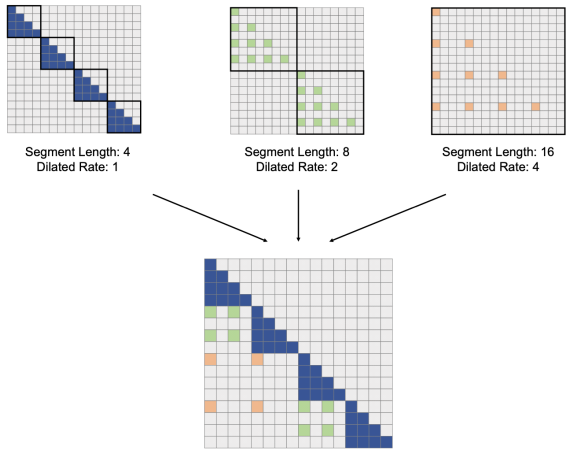
최근 매우긴 길이를 갖는 입력 배열을 처리할 수 있다고 주장하는 트랜스포머 LongNet이 arxiv에 소개되었다[13].

LongNet은 Dilated Attention을 사용하여 SA 연산 복잡도를 $O(n)$으로 줄였다. LongNet의 저자는 [그림 16]처럼 LongNet이 10억 개의 입력 배열을 처리할 수 있다고 주장한다. LongNet의 Dilated Attention은 [그림 17]로 직관적으로 이해할 수 있다. 그림에서처럼 Dilated Rate 값에 따라 Attention의 패턴의 촘촘함이 달라진다.

Dilated Attention은 Attention의 밀도는 토큰 사이의 거리에 기하급수적으로 반비례한다라는 저자들의 원칙에 따라 설계되었다. 또, 저자들은 LongNet의 선형적인 연산 복잡도



[그림 16] 트랜스포머의 입력 길이의 변화 추이[13]



[그림 17] LontNet의 SA 패턴[13]

를 가지는 Dilated Attention 구조로 인해 병렬화가 용이하여 10억 개에 달하는 입력 Token을 처리할 수 있다고 주장한다.

Ⅴ. 시사점

비디오 데이터는 공간적으로는 이미지의 특성을, 그리고 시간적으로는 시계열의 특성을 함께 갖는 독특한 데이터이다. 기본적으로 데이터의 크기가 크므로 $O(n^2)$연산 복잡도를 가지는 트랜스포머의 SA을 그대로 적용하기는 어렵다. 따라서 비디오 데이터의 특성을 반영하면서 트랜스포머의 혁신적인 성능을 활용하기 위한 시도들이 있어 왔고, 앞으로도 계속될 것으로 생각된다.

비디오 데이터에 있어 시간축과 공간축을 분리하여 연산 복잡도를 줄이려는 시도는 직관적이면서도 자연스럽다. 이러한 시도들 중 프레임별로 Embedding을 먼저 수행하고 Embedding 벡터 배열을 트랜스포머로 처리하는 방식은 비교적 널리 사용되어온 방식이다. 각각의 응용 목적에 따라 Embedding 모델이 다를 수 있으므로 Embedding 모델의 성능이 전체 성능에 많은 영향을 미칠 수 있다. 또, 시간축과 공간축을 분리함으로써 연산 복잡도를 줄일 수 있는 반면, 시공간을 동시에 고려하지 못하는 점은 비디오 데이터의 시공간적 특징을 모델링하지 못하는 단점이 될 수 있다.

한편 3D Tublet 방식의 Tokenization은 비디오 데이터의 시공간적 특징을 좀더 효율적으로 모델링하는 것으로 알려져 있는 반면, 많은 입력 토큰을 다루어야 한다는 점에서 연산 복잡도를 줄이기 위한 노력이 동반되어야 한다. 앞에서 살펴본 MotionFormer와 DVT는 비디오 데이터의 Motion에 해당하는맥락(Semantic) 특성을 활용하여 SA의 연산 복잡도를 줄이고자 했다. 두 기술의 한계는 MotionFormer의 성능은 Prototype 벡터를 선정하는 방식에 의존적이리라는 점과, DVT는 프레임마다 고정된 수의 Attention만을 가질 수 있다는 점이다.

긴 입력 배열을 다루기 위한 트랜스포머로 Nyströmformer와 LongNet도 살펴보았다. Nyströmformer는 성능이 Landmark 벡터 선정 방식에 따라 의존적으로 결정된다는 점, LongNet은 Dilated Attention이 입력 배열의 맥락을 고려하지 않고 설계된 점이 한계점으로 보인다.

앞으로의 VT는 비디오 데이터의 맥락을 적절히 활용하면서, SA의 연산 복잡도를 낮춰 입력 배열의 길이를 더 늘일 수 있는 다양한 방법을 시도할 것으로 예상된다.

Ⅵ. 결 론

본고에서는 비디오 데이터를 효과적으로 다루기 위한 최근 트랜스포머의 발전 현황에 대해 살펴보았다. 대체로 매우 긴 비디오 데이터에 대해 트랜스포머의 핵심인 SA를 활용하기 위해서는 $O(n^2)$의 연산 복잡도를 효과적으로 줄이는 방법이 필요하다. 최근 VT 기술은 비디오 데이터의 공간적인 특성이나 시간적인 중복성을 고려하여 SA를 적은 연산량으로 근사하는 방향으로 발전하고 있다. 비디오 데이터의 넓은 응용 가능성을 고려할 때 이러한 VT의 발전은 계속될 것으로 보이고, 머지않은 미래에 실제 응용에 널리 활용될 수 있는 기술 발전이 이루어 질것이라 기대한다.

참고문헌

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need”, *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “Bert:

Pre-training of deep bidirectional transformers for language understanding", *arXiv:1810.04805*, 2018.

[3] L. Floridi, M. Chiriatti, "GPT-3: Its nature, scope, limits, and consequences", *Minds and Machines*, vol. 30, pp. 681-694, 2020.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[5] J. Selva, A. S. Johansen, S. Escalera, K. Nasrollahi, T. B. Moeslund, and A. Clapes, "Video transformers: A survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[6] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations" *arXiv preprint arXiv:1803.02155*, 2018.

[7] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?", In *ICML*, vol. 2, no. 3, p. 4, Jul. 2021.

[8] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer", In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6836-6846, 2021.

[9] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, and M. Suleyman, "The kinetics human action video dataset", *arXiv preprint arXiv:1705.06950*, 2017.

[10] M. Patrick, D. Campbell, Y. Asano, I. Misra, F. Metze, C. Feichtenhofer, A. Vedaldi, and J. F. Henriques, "Keeping your eye on the ball: Trajectory attention in video transformers", *Advances in Neural Information Processing Systems*, vol. 34, pp.12493-12506, 2021.

[11] J. Wang, L. Torresani, "Deformable video transformer", In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14053-14062, 2022.

[12] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, "Nystromformer: A nystrom-based algorithm for approximating self-attention", In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, pp. 14138-14148, May 2021.

[13] J. Ding, S. Ma, L. Dong, X. Zhang, S. Huang, W. Wang, and F. Wei, "Longnet: Scaling transformers to 1,000,000,000 tokens", *arXiv preprint arXiv:2307.02486*, 2023.

[14] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale vision transformers", In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6824-6835, 2021.

≡ 필자소개 ≡

조 현 중

1996년 2월: 경북대학교 전자공학과 (공학사)
1998년 2월: 포항공대 전자공학과 (공학석사)
2006년 9월: Virginia Tech. 컴퓨터공학과 (공학박사)
2009년 3월～현재: 고려대학교 컴퓨터융합소프트웨어학과 교수
[주 관심분야] 딥러닝, 컴퓨터 비전, 행동인식, 상황인식, 영상합성