요약

https://github.com/limaosen0/AS-GCN

- '(2018)Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition' 논문에서 취약점(관절의 주변 지역과의 연관성만 학습)을 보완하기 위한 후속논문
- 기존 Spatial Configuration의 hop 수를 늘리고(S-link) + 표현 모션에 따라 joint 간 연관성을 학습시켜서(A-link) S-GCN을 개선하고, 시간적인 연산은 T-CN으로 계산하도록 변경
  - 초기 학습(10 epoch 정도)에서 A-link를 별도의 loss function으로 학습한뒤
  - 그림 2와 같이 cross entropy loss / L2 loss를 사용하여 학습
- 다음 프레임을 예측하는 Prediction Head를 추가하여 행동 인식 정확도에 미미한 성능개선 보임(0.8%)
- 이전 논문과 같은 데이터셋으로 실험하여 ST-GCN보다 성능 개선(정확도 4~5% 향상)

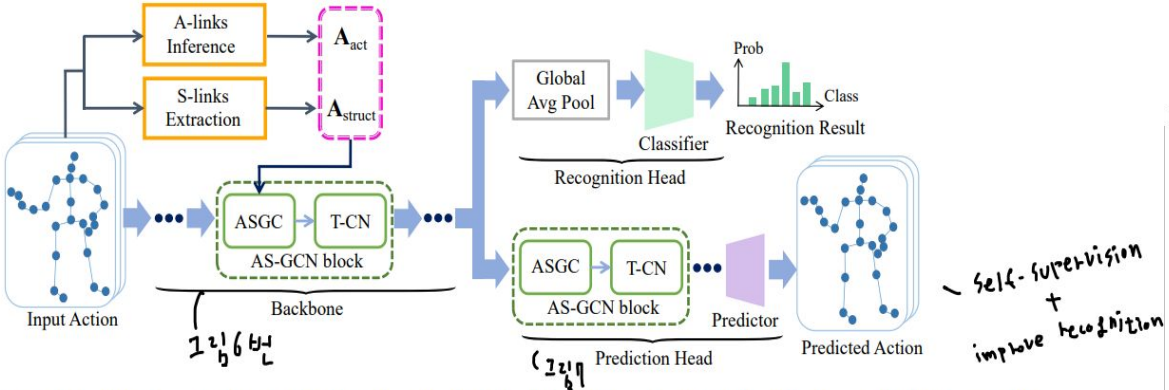


Figure 2: The pipeline of the proposed AS-GCN. The inferred actional graph *A-links* and extended structural graph *S-links* are fed to the AS-GCN blocks to learn spatial features. The last AS-SCN block is connect to two parallel branches, the recognition head and the prediction head, which are simultaneously trained.

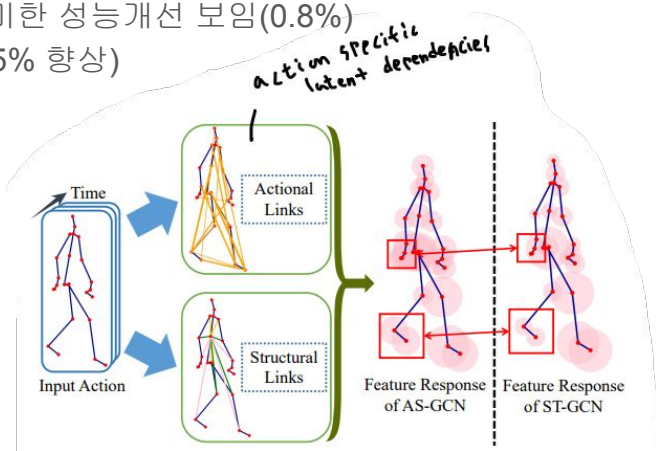


Figure 1: Feature learning with generalized skeleton graphs. the actional links and structural links capture dependencies between joints. For the action "walking", actional links denotes that hands and feet are correlated. The semilucent circles on the right bodies are the joint feature maps for recognition, whose areas are the response magnitudes. Compared to ST-GCN, AS-GCN obtains responses on collaborative moving joints (red boxes).

모델 블럭

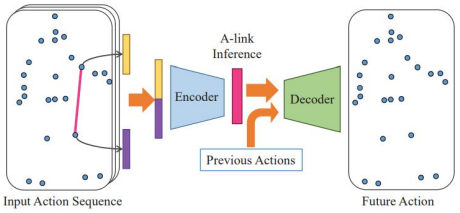


Figure 4. A-links inference module (AIM). To infer the A-link between two joints, the joint features are concatenated and fed into the encoder-decoder formed AIM. The encoder produces the inferred A-links and the decoder generates the future pose conditioned on the A-links and previous actions.

$$\mathbf{X}_{\text{act}} = \text{AGC}(\mathbf{X}_{\text{in}}) = \sum_{c=1}^{C} \hat{\mathbf{A}}_{\text{act}}^{(c)} \mathbf{X}_{\text{in}} \mathbf{W}_{\text{act}}^{(c)} \in \mathbb{R}^{n \times d_{\text{out}}}, \quad (4)$$

$$\mathbf{X}_{\text{struc}} = \text{SGC}(\mathbf{X}_{\text{in}}) = \sum_{l=1}^{L} \sum_{p \in \mathcal{P}} \mathbf{M}_{\text{struc}}^{(p,l)} \circ \hat{\mathbf{A}}^{(p)l} \mathbf{X}_{\text{in}} \mathbf{W}_{\text{struc}}^{(p,l)} \in \mathbb{R}^{n \times d_{\text{out}}}, \quad (5)$$

$$\mathbf{X}_{\text{out}} = \text{ASGC}(\mathbf{X}_{\text{in}}) = \mathbf{X}_{\text{struc}} + \lambda \mathbf{X}_{\text{act}} \in \mathbb{R}^{n \times d_{\text{out}}},$$

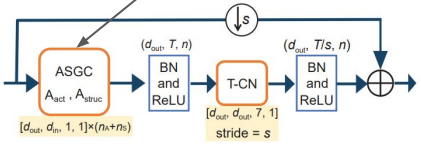
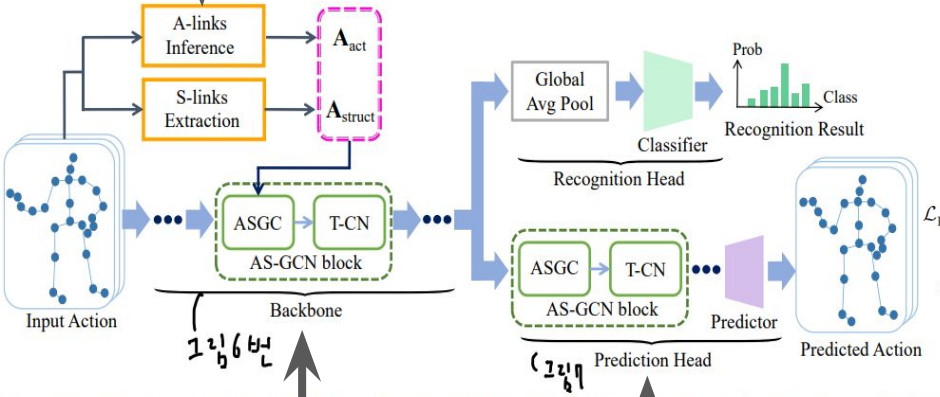


Figure 5. An AS-GCN block consists of ASGC, T-CN, and other operations: batch normalization (BN), ReLU and the residual block. The shapes of data are above the BN and ReLU blocks. The shapes of network parameters are under ASGC and T-CN.



$$\mathcal{L}_{\text{recog}} = -\mathbf{y}^{\text{T}} \log(\hat{\mathbf{y}})$$

$$\mathcal{L}_{\text{predict}} = \frac{1}{ndT'} \sum_{i=1}^{nd} \sum_{t=1}^{T'} \left\| \hat{\mathcal{X}}_{i,:,t} - \mathcal{X}_{i,:,t} \right\|_2^2$$

self-supervision + improve recognition

Figure 2: The pipeline of the proposed AS-GCN. The inferred actional graph *A-links* and extended structural graph *S-links* are fed to the AS-GCN blocks to learn spatial features. The last AS-SCN block is connect to two parallel branches, the recognition head and the prediction head, which are simultaneously trained.

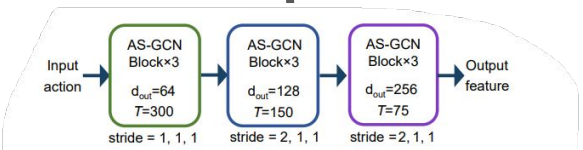


Figure 6: The backbone network of AS-GCN, which includes nine AS-GCN blocks. The feature dimensions are presented.

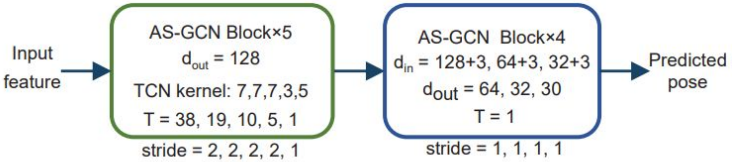


Figure 7. Future prediction head of AS-GCN.

Actional Link

$$\mathcal{A} = \text{encode}(\mathcal{X}) \in [0,1]^{n \times n \times C}, \quad (2)$$

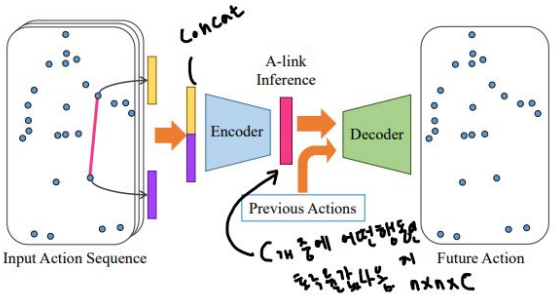


Figure 4: A-links inference module (AIM). To infer the A-link between two joints, the joint features are concatenated and fed into the encoder-decoder formed AIM. The encoder produces the inferred A-links and the decoder generates the future pose conditioned on the A-links and previous actions.

- **Structural Link**는 이전논문과 동일하게 partition strategies를 활용하고 **Actional Link**는 어떤 포즈 중인지에 따라 관절 별 인접 매트릭스(=edge)를 만드는 것
  그림 (4) - 그림 (2)에서의 A-link Inference 블록에 대한 상세 표현
- 초기 학습(총 학습 100 epoch 중 10 epoch)에서는 A-link를 학습하기위해 Encoder-Decoder 구조로 학습
  - 이후 나머지 모델 학습시 Encoder 영역만 학습 Decoder 사용 X
- encoder
  - 식 (2) 에 의해 [batch_size, 3d position, frame num ,joint] 을 입력으로 [joint, joint, Action-type]을 출력하는 모듈
  - 인코더는 Link features / Joint features 를 만드는 것의 반복으로 linear layer, concat, element-wise mean 같은 간단한 레이어로 구성
    - Link feature : 서로 다른 joint 간 관계성에 대한 feature [joint num * (joint num -1), feature size]
      - 서로다른 joint feature를 concat 하는 것으로 link feature 구현
    - Joint feature : joint에 대한 feature [joint num, feature size]
      - 얻고자하는 joint와 연결된 Link feature를 average하는 것으로 변환
  - 초기학습
    - 식(3)으로 action-type, i-j joint 별 linking probability를 관계성을 Action-type을 기준으로 softmax
    - Action-type C은 feature size로 모델 설계할 때 조작할 수 있는 hyper-parameter (논문은 C = 3을 사용)
    - C 값 너무 작으면 Action에 대한 학습이 힘들고, 값이 커도 너무 많은 관계성을 표현하여 학습에 문제가 됨
    - 학습 시 노드간의 관계성이 없는 상태를 우선시하기 위해 0번째 C를 Ghost Link로 정의하고 C0의 prior P를 높게 설정 및 실제 학습/연산은 1번째 C부터 사용하도록 설계함
  - 이후 학습에서는 encoder의 softmax 연산 전 값인 action-type, i-j joint 별 linking probability A_act 를 사용하여 식(4)로 Actional Graph Convolution 계산

$$\text{link features}: \mathbf{Q}_{i,j}^{(k+1)} = f_e^{(k)}(f_v^{(k)}(\mathbf{p}_i^{(k)}) \oplus (f_v^{(k)}(\mathbf{p}_j^{(k)})),$$
$$\text{joint features}: \mathbf{p}_i^{(k+1)} = \mathcal{F}(\mathbf{Q}_{i,:}^{(k+1)}) \oplus \mathbf{p}_i^{(k)},$$
$$\mathcal{A}_{i,j,:} = \text{softmax}\left(\frac{\mathbf{Q}_{i,j}^{(K)} + \mathbf{r}}{\tau}\right) \in \mathbb{R}^C, \quad (3)$$
$$\forall i,j,\ \mathcal{A}_{i,j,0} + \sum_{c=1}^{C} \mathcal{A}_{i,j,c} = 1,$$
$$\mathcal{A}_{:,:,0}^{(0)} = P_0 \text{ and } \mathcal{A}_{:,:,c}^{(0)} = P_0/C \text{ for } c = 1, 2, \cdots, C.$$

$$\mathbf{X}_{\text{act}} = \text{AGC}(\mathbf{X}_{\text{in}}) \quad (4)$$
$$= \sum_{c=1}^{C} \hat{\mathbf{A}}_{\text{act}}^{(c)} \mathbf{X}_{\text{in}} \mathbf{W}_{\text{act}}^{(c)} \in \mathbb{R}^{n \times d_{\text{out}}},$$

Actional Link(계속) + Structural Link

$$\mathcal{A}_{i,j,:} = \text{softmax}\left(\frac{\mathbf{Q}_{i,j}^{(K)} + \mathbf{r}}{\tau}\right) \in \mathbb{R}^C, \qquad (3)$$

- decoder
  - 식 (3)의 softmax값과 입력 X로 다음프레임에서의joint position을 예측하는 부분
  -
  - 디코더의 구조는 (a) ~ (d)로 구성됨
    - (a) : linking probability A로 weighted averaging하여 link feature 생성
    - (b) : (a)의 link feature를 joint feature로 변환(연결된 edge의 feature 평균)과 frame time t, joint i 의 x값과 concat
    - (c) GRU 태우기 (link feature GRU)
    - (d) guassian distribution에 따라 다음 프레임의 x값을 구함
  - Gaussian negative log likelihood Loss + KL Divergence Loss L_AIM 로 학습 진행

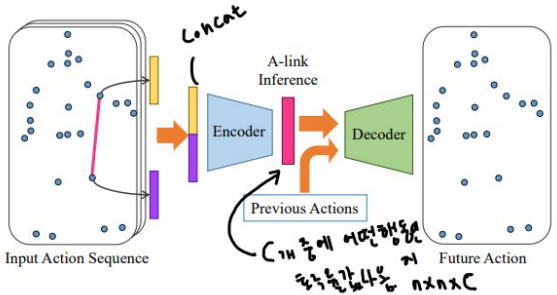


Figure 4: A-links inference module (AIM). To infer the A-link between two joints, the joint features are concatenated and fed into the encoder-decoder formed AIM. The encoder produces the inferred A-links and the decoder generates the future pose conditioned on the A-links and previous actions.

$$\mathbf{X}_{t+1} = \text{decode}(\mathbf{X}_t, \ldots, \mathbf{X}_1, \mathcal{A}) \in \mathbb{R}^{n\times 3},$$

$$\begin{aligned}
(a)\quad \mathbf{Q}_{i,j}^t &= \textstyle\sum_{c=1}^{C} \mathcal{A}_{i,j,c} f_e^{(c)}(f_v^{(c)}(\mathbf{x}_i^t) \oplus f_v^{(c)}(\mathbf{x}_j^t)) \\
(b)\quad \mathbf{p}_i^t &= \mathcal{F}(\mathbf{Q}_{i,:}^t) \oplus \mathbf{x}_i^t \\
(c)\quad \mathbf{S}_i^{t+1} &= \text{GRU}(\mathbf{S}_i^t, \mathbf{p}_i^t) \\
(d)\quad \hat{\mu}_i^{t+1} &= f_{\text{out}}(\mathbf{S}_i^{t+1}) \in \mathbb{R}^3,
\end{aligned}$$

$$\mathcal{L}_{\text{AIM}}(\mathcal{A}) = -\sum_{i=1}^{n}\sum_{t=2}^{T} \frac{\|\mathbf{x}_i^t - \hat{\mu}_i^t\|^2}{2\sigma^2} + \sum_{c=1}^{C} \log \frac{\mathcal{A}_{:,:,c}}{\mathcal{A}_{:,:,c}^{(0)}},$$

- Structural Link
  - 이전 논문의 partition strategies에 의해 hop에 따라
    인접한 부모 joint 그룹과 선택 joint + 자식 joint 그룹을 구분하여 (다른 labeling) S-link를 만듦
  - ex) hop을 4로 지정할 경우, hop=0일 때의 S-link 부터 hop=4일 때의 S-link까지 구성
  - 식 (5)를 통해 S-link 기반의 Graph convolution을 진행

- Actional-Structural Graph Convolution Block (ASGC Block)
  - AGC와 SGC를 각각 계산한 뒤 둘을 더하는 것으로 ASGC 구현

$$\begin{aligned}
\mathbf{X}_{\text{out}} &= \text{ASGC}(\mathbf{X}_{\text{in}}) \\
&= \mathbf{X}_{\text{struc}} + \lambda \mathbf{X}_{\text{act}} \in \mathbb{R}^{n\times d_{\text{out}}},
\end{aligned}$$

$$\begin{aligned}
\mathbf{X}_{\text{struc}} &= \text{SGC}(\mathbf{X}_{\text{in}}) \qquad (5) \\
&= \sum_{l=1}^{L}\sum_{p\in\mathcal{P}} \mathbf{M}_{\text{struc}}^{(p,l)} \circ \hat{\mathbf{A}}^{(p)l} \mathbf{X}_{\text{in}} \mathbf{W}_{\text{struc}}^{(p,l)} \\
&\in \mathbb{R}^{n\times d_{\text{out}}},
\end{aligned}$$

실험

- 이전 논문과 같은 NTU-RGB+D, Kinectics dataset 활용
- 표1 : NTU-RGB+D의 Cross-Subject로 S-link의 hop에 따른 성능 실험
- 표2 : A-link에서 C의 차원수와 Ghost link의 Prior 값 변화에 따른 성능실험
  그림8: A-link 인접행렬 [NxNxC]에서 0.9 이상의 edge를 가진 노드를 주황색으로 표현
- 표3 : Action Recognition 외에 next frame prediction head를 추가로 붙였을때의 변화
- 표4,5 : 타 모델과 성능비교

Table 1. The recognition accuracy on NTU-RGB+D Cross-Subject with various links: S-links, A-links and A- with S-links (AS-links). We tune the polynomial order in S-links from 1 to 4.

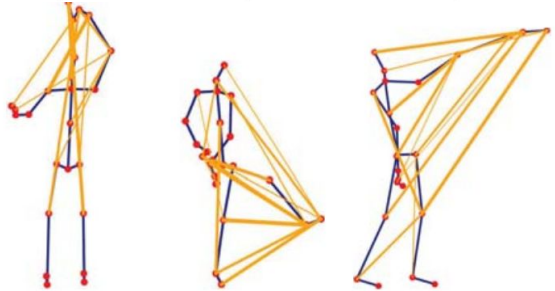
Polynomial order	S-links	A-links	AS-links
1	81.5%	83.2%	83.2%
2	82.2%		83.7%
3	83.4%		84.4%
4	84.2%		**86.1%**

Table 2. Recognition accuracy with various number of A-link types and different priors of the ghost links.

C	1	2	3	4	5
Acc	84.6%	86.5%	**86.8%**	85.8%	83.3%
P_0	0.99	0.95	0.50	0.20	0.00
Acc	86.0%	**86.8%**	84.3%	82.7%	81.1%

Table 3. The recognition results of models with/without prediction heads on NTU-RGB+D Cross-Subject are listed, where the models use AS-links. We tune the order of S-links from 1 to 4.

Polynomial order	AS-links	+ Pred
1	83.2%	84.0%
2	83.7%	84.3%
3	84.4%	85.1%
4	86.1%	**86.8%**



(a) Hand waving (b) Kick something (c) Taking a selfie

Figure 8. A-links in actions. We plot the A-links with probabilities larger than 0.9. The wider line denotes the larger probability.

Table 4. Comparison of action recognition performance on NTU-RGB+D. The classification accuracies on both Cross-Subject and Cross-View benchmarks are presented.

Methods	Cross Subject	Cross View
Lie Group [27]	50.1%	52.8%
H-RNN [6]	59.1%	64.0%
Deep LSTM [22]	60.7%	67.3%
PA-LSTM [22]	62.9%	70.3%
ST-LSTM+TS [20]	69.2%	77.7%
Temporal Conv [14]	74.3%	83.1%
Visualize CNN [21]	76.0%	82.6%
C-CNN+MTLN [13]	79.6%	84.8%
ST-GCN [29]	81.5%	88.3%
DPRL [26]	83.5%	89.8%
SR-TSL [23]	84.8%	92.4%
HCN [18]	86.5%	91.1%
AS-GCN (Ours)	**86.8%**	**94.2%**

Table 5. Comparison of action recognition performance on Kinetics. We list the top-1 and top-5 classification accuracies.

Methods	Top-1 Acc	Top-5 Acc
Feature Enc [8]	14.9%	25.8%
Deep LSTM [22]	16.4%	35.3%
Temporal Conv [14]	20.3%	40.0%
ST-GCN [29]	30.7%	52.8%
AS-GCN (Ours)	**34.8%**	**56.5%**