Multi-Label Image Recognition
with Graph Convolutional Networks

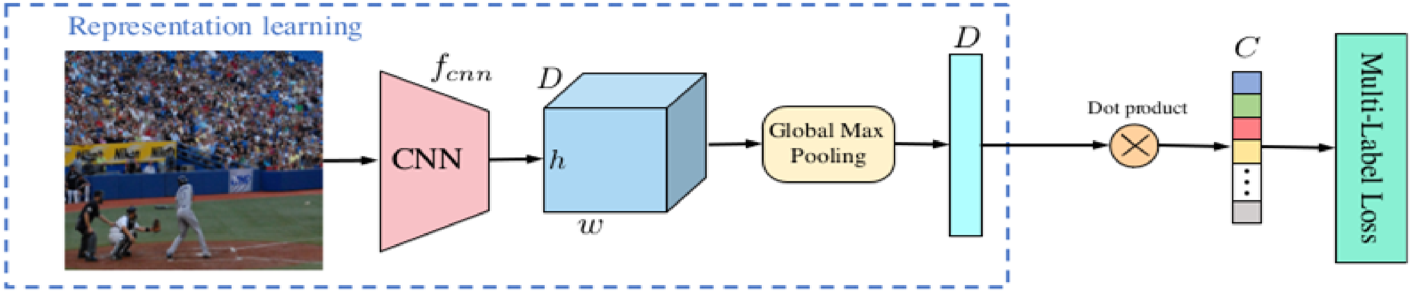
2020-21157 Cheeun Hong

Chen, Zhao-Min and Wei, Xiu-Shen and Wang, Peng and Guo, Yanwen.
"Multi-Label Image Recognition with Graph Convolutional Networks",
CVPR, 2019, pp. 5177-5186

Motivation

- Image 안의 Label 간 연관성 이용

Image $\xrightarrow{\text{ResNet}}$ Feature $\xrightarrow{\text{Classifier}}$ Label



Label graph 를 independent object classifier 로 **mapping 해주는 GCN**을 학습하자!

Background

- How to train GCN:
- GCN: **correlation matrix** 에 기반해 node 사이의 정보를 propagate
  correlation matrix 가 주어지지 않은 환경에서는 data를 통해 구하기
- **Data-driven matrix**

$$P_{ij} = P(L_j|L_i)$$

  - 학습 set 에 있는 label pair 이용
  - 어떤 label 들이 서로 얼마나 co-occur 하는 지 확률로
  - 문제: 희귀한 label pair가 noise가 될 수 있다 (outliers' effect)

Implementations

- **Binary matrix**

$$A_{ij} = \begin{cases} 0, & \text{if } P_{ij} < \tau \\ 1, & \text{if } P_{ij} \geq \tau \end{cases}$$

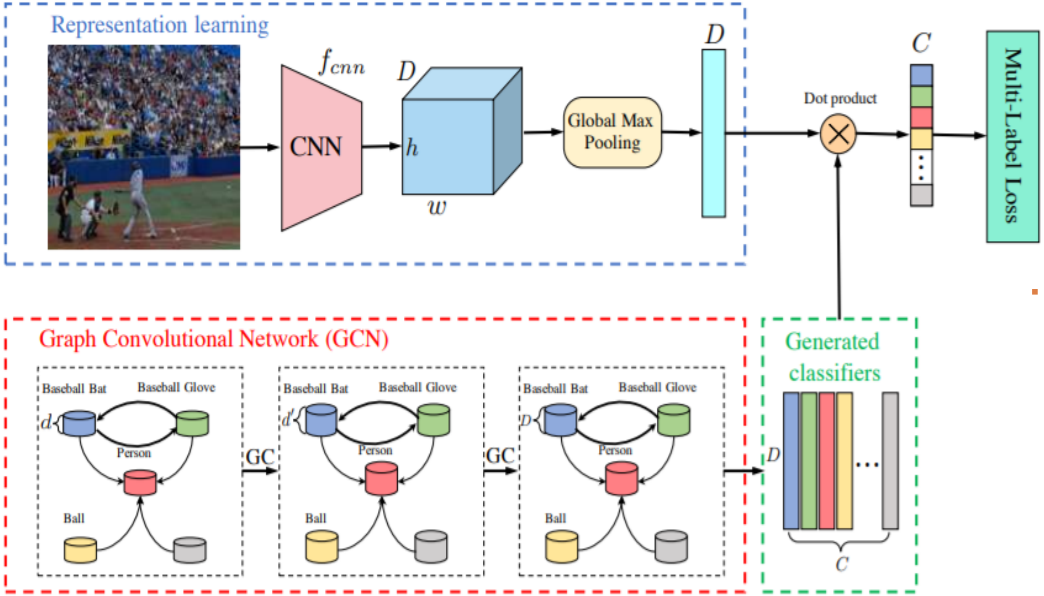
- noise 제거를 위해 threshold with tau
- noise 없어지면서, 자신까지 oversmoothing 되는 문제

- **Re-weighted matrix**

$$A'_{ij} = \begin{cases} p / \sum_{\substack{j=1 \\ i \neq j}}^{C} A_{ij}, & \text{if } i \neq j \\ 1 - p, & \text{if } i = j \end{cases},$$

- 자신이 oversmoothing 되지 않도록 fixed weight value
- 주변 : distribution 에 따라 weight 조절

Experiments



- **Representational Learning**
  - 입력: 448x448 영상
  - 모듈: ResNet101, ImageNet pretrained
  - 출력: 2048-dim feature vector
- **Graph Convolutional Network**
  - 입력: Cx300 word embedding features (pretrained, GLoVe)
  - 모듈: GCN 2개
  - 출력: Cx2048 interdependent object classifier

Experiments

Table 1. Comparisons with state-of-the-art methods on the MS-COCO dataset. The performance of the proposed ML-GCN based on two types of correlation matrices are reported."Binary" denotes that we use the binary correlation matrix, cf. Eq. (7). "Re-weighted" means the correlation matrix generated by the proposed re-weighted scheme is used, cf. Eq. (8).

Methods	All							Top-3					
	mAP	CP	CR	CF1	OP	OR	OF1	CP	CR	CF1	OP	OR	OF1
CNN-RNN [28]	61.2	–	–	–	–	–	–	66.0	55.6	60.4	69.2	66.4	67.8
RNN-Attention [29]	–	–	–	–	–	–	–	79.1	58.7	67.4	84.0	63.0	72.0
Order-Free RNN [1]	–	–	–	–	–	–	–	71.6	54.8	62.1	74.2	62.2	67.7
ML-ZSL [15]	–	–	–	–	–	–	–	74.1	**64.5**	69.0	–	–	–
SRN [36]	77.1	81.6	65.4	71.2	82.7	69.9	75.8	85.2	58.8	67.4	87.4	62.5	72.9
ResNet-101 [10]	77.3	80.2	66.7	72.8	83.9	70.8	76.8	84.1	59.4	69.7	89.1	62.8	73.6
Multi-Evidence [6]	–	80.4	70.2	74.9	85.2	72.5	78.4	84.5	62.2	70.6	89.1	64.3	74.7
ML-GCN (Binary)	80.3	81.1	70.1	75.2	83.8	74.2	78.7	84.9	61.3	71.2	88.8	65.2	75.2
ML-GCN (Re-weighted)	**83.0**	**85.1**	**72.0**	**78.0**	**85.8**	**75.4**	**80.3**	**89.2**	64.1	**74.6**	**90.5**	**66.5**	**76.7**

MS-COCO, VOC2007 : **SOTA < Binary < Re-weighted**

Ablation Studies

- Correlation Matrix

$\boldsymbol{\tau}$ (in binary matrix)

p (in re-weighted matrix)

- GCN

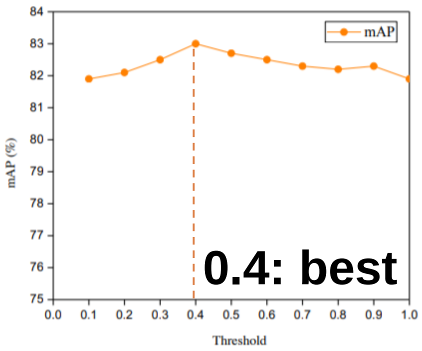
type of word embeddings

GCN layer depth

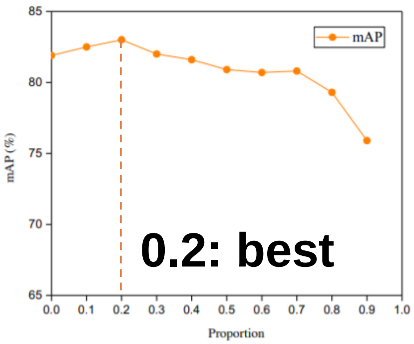
Ablation Studies

- Correlation Matrix

$\boldsymbol{\tau}$ (in binary matrix)	p (in re-weighted matrix)



(a) Comparisons on MS-COCO.



(a) Comparisons on MS-COCO.

Ablation Studies

- Correlation Matrix

$\boldsymbol{\tau}$ (in binary matrix)

p (in re-weighted matrix)

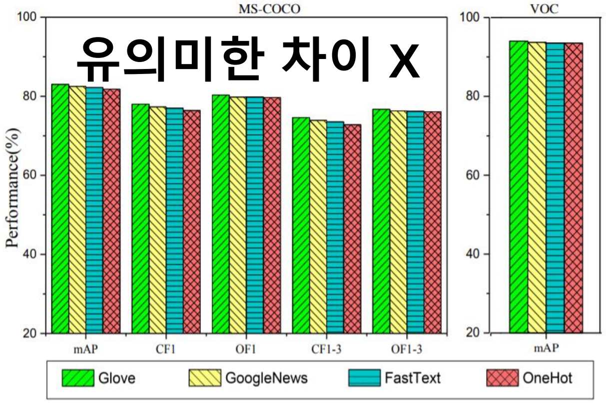
- GCN

type of word embeddings

GCN layer depth

Ablation Studies

- GCN



♯ Layer	MS-COCO				
	All			Top-3	
	mAP	CF1	OF1	CF1	OF1
2-layer	**83.0**	**78.0**	**80.3**	**74.6**	**76.7**
3-layer	82.1	76.9	79.7	73.7	76.2
4-layer	81.1	76.4	79.4	72.5	75.8

깊을수록 성능저하

type of word embeddings

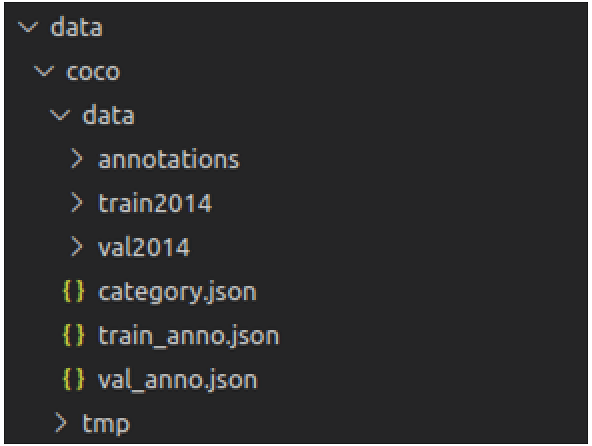
GCN layer depth

Conclusion

- Image Recognition 분야에 GCN을 적용함
- ML-GCN 을 통해 파악한 label dependencies를 이용해 더 성능 좋은 classifier 를 학습함
- Image retrieval 관점에서, image representation 자체의 학습에도 도움되는 것 확인함
    - ML-GCN + K-NN algorithm vs ResNet backbone

Code Implementations

- Existing code
  - fixed some deprecated functions (not supported by torch >0,4 version)
- Data
  - Save COCO2014 dataset
  - in directory data/coco/data
- Environment
  - saved conda environment file in environment.yaml
  - *conda env create -f environment.yml --name MLGCN*
  - *conda activate MLGCN*



* Detailed guide in my github repository

Training & Testing

- Train
  - <u>reweighted matrix</u>, batch_size = 8, epochs=40 (epoch_step = 20), training time = about 1.5 day
- Test
  - *bash demo.sh* # to see test results for COCO2014
- Reproduced Results
  - error rate all lower than 1.5% → **well reproduced!**

Methods	All							Top-3					
	mAP	OP	OR	OF1	CP	CR	CF1	OP	OR	OF1	CP	CR	CF1
ML-GCN (reweighted)	83.0	85.8	75.4	80.3	85.1	72.0	78.0	90.5	66.5	76.7	89.2	64.1	74.6
Reproduced	83.0	84.7	76.2	80.2	84.0	72.8	78.0	89.8	66.8	76.6	88.6	63.9	74.3
Error rate [%]	0	1.3	-1.1	0.1	1.3	-1.1	0	0.8	-0.5	0.1	0.7	0.3	0.4

* Error rate = expected_value - experimented_result / expected_value *100

Repository

- My Repository
  - https://github.com/Cheeun/ML-GCN
- Original
  - https://github.com/Megvii-Nanjing/ML-GCN