The Journal of the Convergence on Culture Technology (JCCT)
Vol. 10, No. 1, pp.471-476, January 31, 2024. pISSN 2384-0358, eISSN 2384-0366

http://dx.doi.org/10.17703/JCCT.2024.10.1.471
JCCT 2024-1-56

딥러닝 기술을 적용한 그래프 알고리즘 성능 연구

Research on Performance of Graph Algorithm using Deep Learning Technology

노기섭*

Giseop Noh*

요 약 다양한 스마트 기기 및 컴퓨팅 디바이스의 보급에 따라 빅데이터 생성이 광범위하게 일어나고 있다. 기계학습은 데이터의 패턴을 학습하여 추론을 수행하는 알고리즘이다. 다양한 기계학습 알고리즘 중에서 주목을 받는 알고리즘은 신경망 기반의 딥러닝 학습이다. 딥러닝은 다양한 응용이 발표되면서 빠른 성능 향상을 달성하고 있다. 최근 딥러닝 알고리즘 중에서 그래프 구조를 활용하여 데이터를 분석하려는 시도가 증가하고 있다. 본 연구에서는 그래프 구조를 활용하여 딥러닝 네트워크에 전달하기 위한 그래프 생성 방법을 제시한다. 본 논문은 그래프 생성 과정에서 노드의 속성과 간선의 가중치를 일반화하고 행렬화 과정을 제시하여 딥러닝 입력에 필요한 구조로 전환하는 방법을 제시한다. 그래프 생성 과정에서 속성과 가중치 정보를 보전할 수 있는 선형변환 매트릭스 적용 방법을 제시한다. 마지막으로 일반 그래프의 딥러닝 입력 구조를 제시하고 성능 분석을 위한 접근법을 제시한다.

주요어 : 딥러닝, 그래프 네트워크, 그래프 행렬화, 그래프 생성

Abstract With the spread of various smart devices and computing devices, big data generation is occurring widely. Machine learning is an algorithm that performs reasoning by learning data patterns. Among the various machine learning algorithms, the algorithm that attracts attention is deep learning based on neural networks. Deep learning is achieving rapid performance improvement with the release of various applications. Recently, among deep learning algorithms, attempts to analyze data using graph structures are increasing. In this study, we present a graph generation method for transferring to a deep learning network. This paper proposes a method of generalizing node properties and edge weights in the graph generation process and converting them into a structure for deep learning input by presenting a matricization We present a method of applying a linear transformation matrix that can preserve attribute and weight information in the graph generation process. Finally, we present a deep learning input structure of a general graph and present an approach for performance analysis.

Key words : Deep Learning, Graph Networks, Graph Matricization, Graph Generation

Ⅰ. 서 론

다양한 스마트 기기 및 컴퓨팅 디바이스 보급이 급격히 증가하면서 빅데이터 생성이 광범위하고 활발하게 일어나고 있다. 이렇게 생성된 빅데이터를 활용하여 현상을 분석하고

*정회원, 청주대학교 소프트웨어융합학부 교수 (교신저자)
접수일: 2023년 10월 5일, 수정완료일: 2023년 10월 20일
게재확정일: 2023년 11월 5일

Received: October 5, 2023 / Revised: October 20, 2023
Accepted: November 5, 2023
*Corresponding Author: kafa46@cju.ac.kr
Dept. of Software Convergence, Cheongju Univ, Korea

예측하는 연구도 빠르게 발전하고 있다. 기계학습(Machine Learning, ML)은 수집된 데이터를 기계가 학습하고 새로운 입력에 대한 추론을 수행하는 알고리즘이다. 기계학습은 다양한 알고리즘과 응용(application) 방법이 존재한다. 기계학습 알고리즘 중에서 가장 높은 성능을 보이는 것은 딥러닝(deep learning)이다.

딥러닝은 데이터를 추상화 과정 및 표현(representation)을 다수의 은닉 계층 (hidden layer)를 통해 학습하는 알고리즘이다[1]. 딥러닝의 기본이 되는 신경망 구조는 DNN(Deep Neural Network)이다. 최근 딥러닝의 높은 예측 성능이 보고되면서 다양한 분야로 적용 범위가 확대되고 있다. 딥러닝의 대표적인 연구 분야로는 자연어처리(natural language processing), 영상처리, 음성처리 등이 있다. 대표적인 알고리즘으로는 입력 데이터의 패턴을 추출하여 예측하는 CNN(Convolutional Neural Network)[2], 시간의 흐름에 따른 변화 패턴을 추론하는 RNN(Recurrent Neural Network)[3] 등이 있다. 최근에는 그래프 자료구조를 이용한 GNN(Graph Neural Network)[4] 연구도 활발하게 진행되고 있다. 딥러닝 애플리케이션(application) 또한 군사 분야[5], 장애인 복지 분야[6], 의료 분야[7] 등으로 확대되고 있다.

그래프(graph)는 데이터를 노드(node)와 간선(edge)로 데이터를 표현하는 자료구조를 의미한다. 그래프로 표현할 수 있는 데이터의 예로는 소셜 네트워크, 물류망, 통신 네트워크, 논문 인용 네트워크 등 다양하다(그림 1 참조). 그래프 표현의 특징은 노드에 속성을 부여해 개별 특성 정보를 추가할 수 있다.

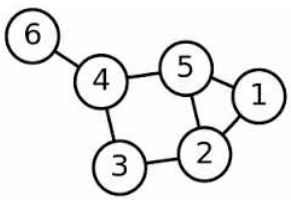


그림 1. 그래프 자료구조
Figure 1. Graph data structure

또한, 그래프 구조의 간선은 방향성(단방향, 양방향, 무방향)을 선택하거나 가중치(weight)를 추가하여 데이터 자체의 표현력을 증강시킬 수 있다는 점이 특징이다. 그래프 자료구조는 링크드 리스트(linked list) 또는 행렬로 표현 가능하다. 행렬 구조는 인간이 보기에 직관적이고 컴퓨터가 빠르게 처리할 수 있는 다양한 라이브러리가 제공되어 자주 활용된다.

그래프와 신경망 네트워크를 결합한 기본 모델을 GNN이라고 부른다. 기본 GNN은 신경망에 컨볼루션(convolution) 연산을 적용한 네트워크를 GCN(Graph Convolution Networks)[8], 순환(recurrent) 구조를 이용하여 기억력을 추가하는 GRN(Graph Recurrent Network)[9], Attentin mechanism을 적용한 GAN(Graph Attention Network)[10] 등으로 확장할 수 있다.

그래프 알고리즘에 딥러닝 기술을 적용하는 것은 그래프 생성을 어떻게 할 것이냐가 관건이다. 본 연구는 그래프가 가지는 특성에 따른 수치화 및 그래프 구조를 딥러닝 은닉층에서 학습 가능하도록 하는 행렬화(matricization) 및 벡터화(vectorization)에 대한 연구에 초점을 맞춘다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 본 논문에서 적용할 딥러닝 구조에 대하여 간략히 설명하고, Ⅲ장에서는 그래프를 딥러닝 구조에 입력하기 위한 그래프 생성에 대하여 살펴본다. 딥러닝을 적용한 그래프 알고리즘 및 성능분석은 Ⅳ장에서 제시한다. Ⅴ장에서 결론을 제시하며 논문을 마무리한다.

Ⅱ. 딥러닝 구조와 입력

서론에서 언급한 바와 같이 딥러닝 구조는 다양하게 구현될 수 있다. 본 논문에서는 그래프 구조와 딥러닝을 적용한 그래프 알고리즘 성능에 관심을 두고 있으므로, 그래프 자료구조를 딥러닝 입력으로 연결하는 방안이 핵심이다. 따라서 다양한 딥러닝 구조 중에서 가장

기본이 되는 DNN를 간략히 설명하고 Ⅲ장에서 그래프 생성구조를 설명한다. DNN은 1개의 입력(input) 계층, 1개의 출력(output) 계층, 그리고 2개 이상의 은닉(hidden) 계층으로 구성된다.

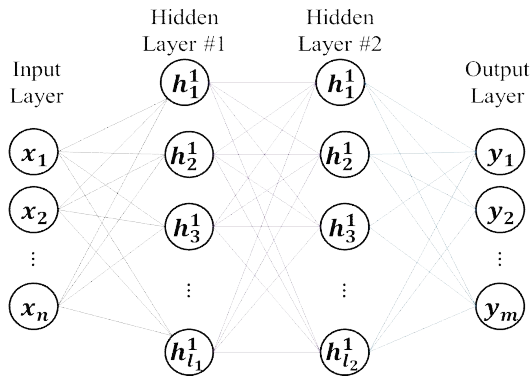


그림 2. DNN 신경망 구조
Figure 2. Neural network structure of DNN

그림 2에서 입력 레이어는 n차원, 출력 레이어는 m차원 이므로, DNN 입력은 $X \in R^n$ 이고, 출력은 $Y \in R^m$ 이다. 여기서 R은 실수 집합이며, R의 윗첨자는 카테시안 곱(Cartesian product)이다. 은닉 계층의 차원은 DNN 설계자에 따라 다양하게 구성할 수 있다. 입력 X 에 따라 내부 신경망이 전진/후진 학습을 통해 출력의 정확도를 높인다. 본 논문에서는 그래프를 입력 레이어에 효과적으로 전달하는 방법에 중심을 두고 있으므로 딥러닝 입력 X 에 집중한다.

Ⅲ. 그래프 생성 구조 분석

1. 그래프 데이터와 그래프 구조 생성

그래프 이론(graph theory)에서 그래프는 객체의 집합에서 객체 사이에서 관련성이 있는 객체들을 연결하는 조합론 구조이다. 그래프를 구성하는 객체는 속성(attribute)을 가지며 노드(node)라고 부른다. 객체와 객체 사이의 관계는 연결선으로 표현하며 간선(edge)라고 부른다. 모든 간선은 가중치(weight)를 가질수도 있고 그렇지 않을 수도 있다. 그래프 G는 식 (1)과 같이 표현할 수 있다.

$$G=(V,E), where \quad (1)$$
$$V=\{n_i \mid i \in N, i < \infty\} \text{ and}$$
$$E=\{e_{i,j}=(n_i, n_j) \mid n_i \in N, n_j \in N\}$$

n_i는 속성값 v를 갖는다. v를 갖는 n_i를 n_i^p라고 하자. v는 다수 값을 가질 수 있으므로 벡터로 해석할 수 있다. 따라서 $v \in R^p$라고 표현할 수 있다. p는 v의 차원(dimension)이다. n_i와 유사하게 $e_{i,j}$도 다수의 가중치(weight) w를 가질 수 있으므로 $e_{i,j}^w$ 라고 표기하며, w는 벡터로 처리한다. $w \in R^q$이며, R은 실수 집합이고 w는 q 차원이다.

임의의 행렬(matrix) M은 r개의 행(row)과 l개의 열(column)을 가지며, $M_{r,l}$ 로 표현하기로 한다. 행렬 표현을 편리하게 다루기 위하여 $M_{r,r}$인 행렬(행의 개수와 열의 개수가 같은 경우)을 M_r로 표현하기로 한다.

우선 $p=q=1$인 특수 경우에 한정하여 살펴보도록 한다. $p=q=1$인 상황을 $G_{1,1}$ 이라고 하자. $G_{1,1}$은 일반적인 행렬 구조이다. 그림 2의 상황을 살펴보면 임의의 노드 n_i는 하나의 속성값을 가진다.

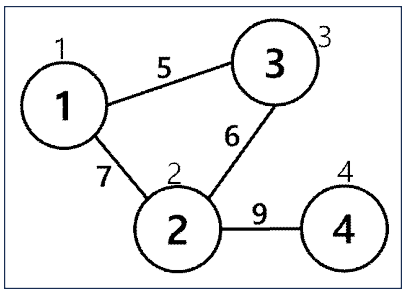


그림 3. $G_{1,1}$ 상황의 그래프 예시
Figure 3. An example of matrix with $G_{1,1}$

설명의 편의를 위해 노드의 인덱스를 속성값으로 지정하였다. 그림 3을 간선 가중치에 대하여 행렬로 표시하면 식 (2)와 같다.

$$M_4 = \begin{pmatrix} 1 & 7 & 5 & 0 \\ 7 & 1 & 6 & 9 \\ 5 & 6 & 1 & 0 \\ 0 & 9 & 0 & 1 \end{pmatrix} \quad (2)$$

식 (2)에서 $e_{i,i}$인 경우(즉 어떤 노드 n_i 자신에 대한 관계)에는 $e_{i,i}^{w}=1$로 표현하였다. 식 (2)는 그래프 네트워크의 특정 상태(state)을 표현한 것으로 해석할 수 있다.

그래프의 상태는 시간이 흐름에 따라 얼마든지 변할 수 있다. 어떤 시점 t에서의 그래프 상태를 행렬로 전환했을 경우를 M_r^t라고 하면, 시간을 고려한 M_r^t는 수식 (3)과 같이 표현할 수 있다.

$$M_r^t = \left\{ M_r^i \mid i = 1, 2, ..., t \right\} \quad (3)$$

그래프를 이용해 문제를 해결하는 방법은 크게 2가지로 구조화(modeling)할 수 있다.

- 유형 1: 각각의 시점($i=1,2,...,t$)에서 그래프가 의미하는 상태(state)가 무엇인지를 알고 있는 상태에서 $t+1$ 시점에서 M_i^{t+1}의 상태를 예측
- 유형 2: M_i^t 가 dataset으로 주어지고 $t+1$ 시점에서 n_i^p 또는 $e_{i,j}^{w}$를 예측

본 논문에서는 유형 1에 국한하여 접근하도록 한다. 유형 2는 후속 연구를 통해 추가 접근법을 제시할 예정이다.

2. 그래프 데이터의 차원 변환

유형 1을 학습하기 위해 확보한 데이터세트를 $D=\left\{\left(M_r^i, y_i\right)\right\}_{i=1}^{n}$이라고 정의하도록 한다. D에서 y_i는 $t=i$ 시점에서의 M_r^i 상태의 레이블(label)이다. 딥러닝 입력 레이어 차원은 n차원 벡터이고, 입력 M_r^i은 $r \times r$ 행렬이므로 차원 변환이 필요하다. 차원 변환 맵핑 함수를 f 라고 하면 그래프 구조의 딥러닝 입력은 수식 (4)와 같다.

$$f: R^{r \times r} \rightarrow R^n \quad (4)$$

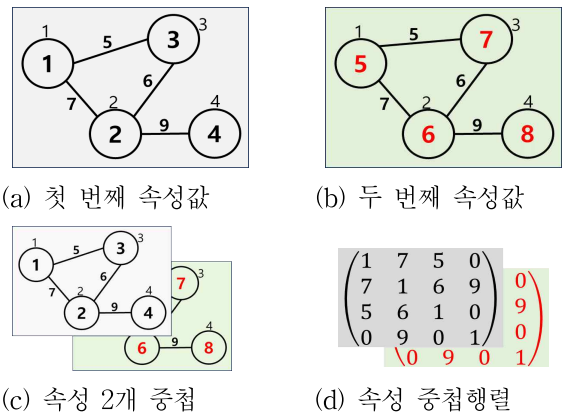
식 (4) 구현 방법으로 선형 변환 방식을 통해 가능하다. 여기서 주의할 점은 선형 변환을 위해 도입하는 변환 행렬은 딥러닝 학습과정에서 참여하는 학습 파라미터로 관리되어야 한다는 점이다.

3. 속성 및 가중치 정보의 고차원 확장

Ⅲ장 1절에서 언급한 바와 같이 그래프 G의 노드 n_i는 1개 이상의 속성을 가질 수 있으므로 식 (4)를 고차원 공간으로 확장해야 한다. 고차원의 개념을 명확히 하기 위해 n_i^p 크기(demensionality)는 식 (5)와 같이 정의하기로 한다.

$$1 \leq \left| n_i^v \right| < \infty,$$
$$where \left| \bullet \right| is\ cardinality\ function. \quad (5)$$

특정 G에서 모든 n_i의 속성의 개수는 동일하다고 가정하고, 본 논문에서는 p로 표현한다. 따라서 $\forall i,\ \left| n_i^v \right| = p$ 이다. 그림 3에 $p=2$인 경우를 적용하면 특정 시간 t_i에서 그래프 구조는 2개의 속성값을 갖는 별개의 행렬로 표현할 수 있다.



(a) 첫 번째 속성값 (b) 두 번째 속성값

(c) 속성 2개 중첩 (d) 속성 중첩행렬

그림 4. 특정 시점에서 $p=2$를 갖는 G의 속성값 예시
Figure 4. An example of a graph with $p=2$ attributes.

그림 4에서 t_i 시점에서 $p=2$인 G의 첫 번

째 속성값은 (a)에, 두 번째 속성값은 (b)에 표시하고, (c)에는 2개의 속성을 중첩한 내용을, 그리고 (d)에는 속성값을 중첩 행렬로 표시하였다. 그림 4를 일반화하여 노드 속성값을 일반화한 데이터는 3차원 텐서 $N^{attr} \in R^{|V| \times |V| \times p}$ 이다. 간선의 가중치도 N과 유사한 방식으로 일반화 할 수 있다. 어떤 G에서 간선 가중치 값은 3차원 텐서 $E^{wegith} \in R^{|V| \times |V| \times q}$ 이다.

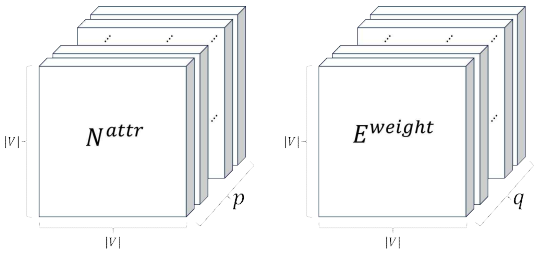


그림 5. 노드 속성 및 간선 가중치 구조
Figure 5. The structures of node attributes and edge weights

그림 5에서 N^{attr}은 간선의 가중치를 표현할 수 없으며, E^{wegith}는 노드 속성값을 표현할 수 없다. N^{attr}과 E^{wegith} W 모든 성질을 보존하면서 하나의 정보로 표현하기 위하여 하나의 행렬로 표현한다. 이를 위해 텐서N을 각 축에 행렬화(matricization)를 진행한다. 행렬화는 각 축에 대한 행렬을 이어 붙이기(concatenate)하여 N^1, N^2, N^3 행렬을 생성한다.

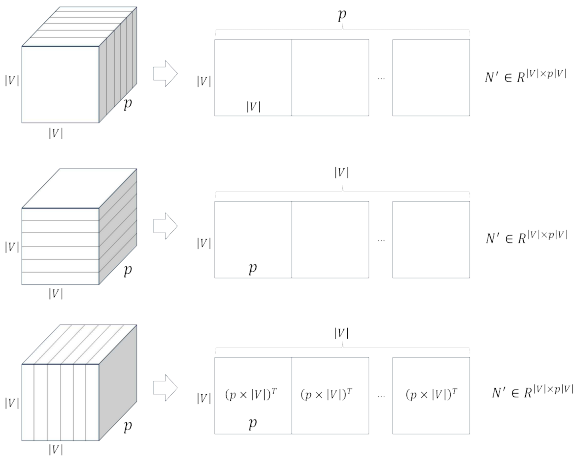


그림 6. 노드 가중치 텐서의 행렬화
Figure 6. Matricization of node weights tensor

그림 6은 노드 가중치 텐서를 행렬화를 표현한 그림이다. 그림 6의 최하단 그림에서 $R^{p \times |V|}$ 행렬을 전치(transpose)하면 3개 행렬의 크기를 $R^{|V| \times p|V|}$로 동일하게 조절할 수 있다. 그래프 구조를 표현할 수 있다. 식 (6)을 적용하여 모든 방향의 정보를 하나로 통합하고 최종적으로 $|V| \times |V|$ 행렬을 생성한다.

$$N^f = W_N \bullet concat(N^1; N^2; N^3)^T, \quad (6)$$
$$where\ N^f \in R^{|V| \times |V|},\ W_N \in R^{|V| \times p|V|}$$

간선의 가중치는 그림 5의 오른쪽 그림과 같이 표현되며 그림 6과 동일한 방식으로 가중치 텐서 W로부터 행렬화를 통해 W^1, W^2, W^3를 생성한다. 간선 가중치는 식 (7)을 이용하여 최종 행렬을 생성한다.

$$E^f = W_E \bullet concat(E^1; E^2; E^3)^T, \quad (7)$$
$$where\ E^f \in R^{|V| \times |V|},\ W_E \in R^{|V| \times q|V|}$$

Ⅳ. 딥러닝 적용 및 그래프 성능

딥러닝 적용은 Ⅲ장에서 도출한 N^f와 E^f를 결합하여 특정 시간 $t=i$에서의 입력 행렬 M_i는 식 (8)을 이용하여 구한다.

$$M_i = W_X^{|V| \times 2|V|} \bullet concat(N^f; E^f)^T \quad (8)$$

딥러닝 입력계층 전달을 위한 입력 벡터 X_i는 M_i를 직렬화(serialize)하여 사용한다. 그래프 구조를 딥러닝에 적용하는 방법은 식 (8)을 구하는 과정을 전처리 과정 또는 임베딩(embedding) 과정으로 포함하는 방법과 학습 파라미터로 적용하여 딥러닝 입력 구조로 포함하는 것이 가능하다. 전자는 기존 임베딩 알고리즘을 변형해 적용할 수 있다. 후자의 경우 선형 변환 행렬 W_N 및 W_E 를 학습에 참여시키는 방법이 가능하다. 속성과 가중치를 학

습하는 방법을 적용하기 위해서는 속성과 가중치를 별개의 입력으로 처리한 이후 연결하는 구조를 적용한다 (그림 7 참조).

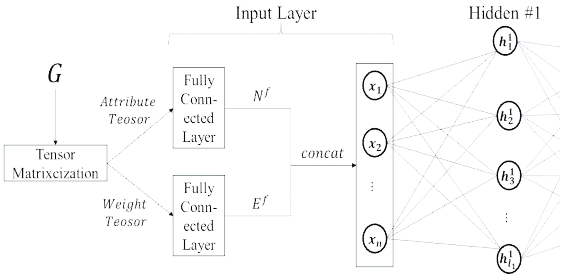


그림 7. 그래프의 딥러닝 적용

Figure 7. Applying graph into deep learning

딥러닝의 최종 성능은 임베딩 성능(α)과 파라미터의 학습 참여를 통한 예측 성능(β)을 비교하여 판단할 수 있다. 그래프 입력 방식에 따른 비교 우위(α/β)는 데이터 수집 분야, 딥러닝 Task 등에 따라 바뀔 수 있다. 딥러닝의 특성상 학습 진행을 추적하기 어려워 실험적 입증 추가적으로 필요하다. 향후 실험적 연구를 통해 입력 방식에 대한 추가 연구를 진행할 예정이다.

V. 결 론

본 연구는 그래프 구조를 정형화하여 딥러닝에 적용하기 위한 접근법을 연구하였다. 속성과 가중치를 텐서로 해석하여 차원별 행렬화를 어떻게 진행할 것인지에 대한 접근법을 제시하였다. 이를 위해 노드 속성과 간선 가중치의 모든 정보를 포함하는 최종 행렬을 도출하여 딥러닝의 입력으로 전달하기 위한 과정을 수학적으로 제시하였다. 최종적으로 그래프의 적용 방안을 제시하고 성능 연구에 대한 방향을 제안하였다. 향후 연구로는 속성과 가중치 값을 예측 방안에 대한 연구를 진행할 예정이다.

References

[1] Kamilaris, A., & Prenafeta-Boldú, F. X., “Deep learning in agriculture: A survey.” Computers and Electronics in Agriculture, Vol. 147, pp. 70-90, April 2018. DO: 10.1016/j.compag.2018.02.016

[2] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," IEEE Transactions on NNLS, 2021. DOI: 10.1109/TNNLS.2021.3084827

[3] S. Grossberg, "Recurrent neural networks," Scholarpedia, Vol. 8, No. 2, p. 1888, 2013. DOI: 10.4249/scholarpedia.1888

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," IEEE Transactions on NNLS, Vol. 32, No. 1, pp. 4-24, 2020. DOI: 10.1109/TNNLS.2020.2978386

[5] P. GunWoo, "CNN and SVM-Based Personalized Clothing Recommendation System : Focused on Military Personnel," JCCT, Vol. 9, No. 1, pp. 347-353, 2023.

[6] S.-H. Choi, J.-H. Kim, J.-D. Oh, and K.-S. Kong, "A Smart Closet Using Deep Learning and Image Recognition for the Blind," JIIBC, Vol. 20, No. 6, pp. 51-58, 2020. DOI : 10.7236/JIIBC.2020.20.6.51

[7] K.-H. Ann and S.-Y. Ohm, "A COVID-19 Chest X-ray Reading Technique based on Deep Learning," JCCT, Vol. 6, No. 4, pp. 789-795, 2020. DOI: 10.17703/JCCT.2020.6.4.789

[8] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," Computational Social Networks, Vol. 6, No. 1, pp. 1-23, 2019.

[9] Z. Liu and J. Zhou, "Graph Recurrent Networks," Introduction to Graph Neural Networks Springer, pp. 33-37, 2020.

[10] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," stat, Vol. 1050, No. 20, pp. 10-48550, 2017. DOI: 10.48550/arXiv.1710.10903

※ 이 논문은 2022. 3. 1.~ 2024. 2. 28. 학년도에 청주대학교 산업과학연구소가 지원한 학술연구 조성비(특별연구과제)에 의해 연구되었음.