ISSN No:-2456-2165

Graph Convolutional Networks: Adaptations and Applications

Sai Annanya Sree Vedala
Computer Science and Engineering
Chaitanya Bharathi Institute of Technology
Hyderabad, India

Pavan Kumar Dharmoju
Electrical and Electronics Engineering
Chaitanya Bharathi Institute of Technology
Hyderabad, India

Rida Malik Mubeen
Computer Science and Engineering
MJCET
Hyderabad, India

Abstract:- Graph Convolutional Networks, Graph Conventional Networks are a generalised version of Convolutional Neural Networks. They are an extension of the generic convolutional operation and have the ability to deal with non-Euclidean types of data and can easily work with nodes and graphs to get features to learn and train the networks. They have evolved over time and have been applied to various domains. The techniques have improved and the performance of the Graph Convolutional Networks has been a great tool in the domain of research. In this study, we present the transformations and improvements of Graph Convolutional Networks and analyse the variation of the contrast between the traditional convolutional neural network and the graph neural network. The different applications have been discussed, adaptations have been highlighted along with the limitations.

Keywords:- *Graph Convolutional Network.*

I. INTRODUCTION

Graph convolutional network [7] is a type of neural network that has a powerful architecture for machine learning on graphs. They are a variant of graph neural networks that can deal with the non-regularity of data structures. They consist of operations of multiplying input neurons with weights. This is the same as the convolutions operations in the convolution layers that present in Convolution neural networks. The set of weights are called filters and these filters act as sliding windows across the images and enable the neural network to learn features. Graph convolutional networks are an extension of Convolutional Neural Networks where the Convolutional Neural Networks are great at computer vision tasks and ability to train deep neural networks but fall short in their efficiency when it comes to variation in the order of the data. While on the other hand, Graph convolutional networks have the ability to work with the unordered data and can work directly on graphs and deal with structural information. One great advantage of using Graph convolutional networks is that it solves the problem of node classification. Each node provides feature information from all neighbours and the aggregate value from the features is fed into a neural network. They use both node features and structures for the learning and the training. These number of hops can be decided as to how fast the information from the entire graph can be covered. It is observed that the results obtained from a 2 to 3 layered Graph convolutional network are quite optimal. The problem of increasing the number of layers is the decrease in the performance of the network. This is one of the issues that will be addressed in the latter part of this paper.

II. UNDERSTANDING GCNS

A. The concept of Convolutional Neural Networks

Convolutional Neural Network [1] is similar to traditional Artificial Neural Networks where they are composed of neurons that tend to self-optimize through a process of self-learning. Every neuron will still receive input and perform an operation which is the very basis of ANNs. Throughout the process, that is, from the input raw image vectors, reaching the final output of the class score, the whole network would still show a single perceptive score function which is the weight. The final concluding layer will contain loss functions associated with the classes, and further, all of the regular tips and tricks built for a traditional ANN will still apply. The major notable difference between Convolutional Neural Networks [16] and traditional ANNs is that Convolutional Neural Networks are majorly used in scenarios involving images. This tends to allow us to encode features which are image-specific into the architecture so that it helps make the network more suited for image-focused tasks like pattern recognition within images. This is done while further reducing the parameters required to set up the model to not make the entire process extremely expensive, both in terms of time and space complexities which is one of the largest limitations of traditional forms of ANN i.e., that they tend to struggle with the computational complexity required to compute image data. The basic architecture of a Convolutional Neural Network can be broken down into the following:

- The Input layer: It will hold the pixel values of the input image.
- The Convolutional Layer: This will determine the output of neurons that are connected to the regions which are local to the input through the calculation of the scalar product of their weights with the region connected to the input volume. This aims to apply an elementwise activation function like the sigmoid to the output of the activation which is produced by the previous layer.

- The Pooling Layer: This will perform down sampling throughout the spatial dimensionality of the input that is given, which further reduces the number of parameters within that activation which contributes to lesser computation power.
- The Fully-Connected Layers: These will finally perform similar duties that are found in standard ANNs and they attempt to produce class scores from the aforementioned activations.

B. Graph Convolutional Network

Graph convolutional network is a type of neural network that has a powerful architecture for machine learning on graphs. They are a variant of graph neural networks that can deal with the non-regularity of data structures. They consist of operations of multiplying input neurons with weights. This is the same as the convolutions operations in the convolution layers that present in Convolution neural networks. The set of weights are called filters and these filters act as sliding windows across the images and enable the neural network to learn features.

Graph neural networks are a generalised version of the convolutional neural networks where the nodes are not ordered and the number of nodes connections vary. It operates on graphs with a matrix as the input. An input feature matrix and a matrix representation of the graph structure are both considered as input to the network. Graph Convolutional Networks are used for semi-supervised learning and the main idea is to take the weighted average of all the neighbours nodes features and passing the resulting feature vectors through a neural network for training. The node level output produced is a feature matrix that can be modelled by introducing pooling operations. The input matrix is typically not normalised and the scale is changed when any multiplication operation takes place. The matrix needs to be normalised in order to deal with the problem.

Graph Convolutional Networks are an extension of Convolutional Neural Networks where the Convolutional Neural Networks are great at computer vision tasks and ability to train deep neural networks but fall short in their efficiency when it comes to variation in the order of the data. While on the other hand, Graph Convolutional Networks have the ability to work with the unordered data and can work directly on graphs and deal with structural information. As mentioned above, it solves the problem of node classification. Each node provides feature information from all neighbours and the aggregate value from the features is fed into a neural network. They use both node features and structures for the learning and the training. Multiple layers are stacked on top of one another to get a deep network. The output of the previous layer is considered as the input for the next layer and so on and so forth. When the layers are stacked, the process of gathering information is repeated and the number of layers is the maximum number of hops each node can travel. This number of hops can be decided as to how fast the information from the entire graph can be covered. It is observed that the results obtained from a 2 to 3 layered Graph Convolutional Network are quite optimal. The problem of increasing the number of layers is the decrease in the performance of the network. Graph Convolutional Networks having the semi supervised learning ability with normalised propagation leads to an improvement in the efficiency in terms of the parameters and operations and better prediction.

III. IMPROVEMENTS ON CONVENTIONAL GRAPH CONVOLUTIONAL NETWORKS

One definite improvement [4] to Graph Convolutional Networks would be to be able to make them go deeper than the standard three to four layers and still not face issues like the vanishing gradient problem. Drawing from Convolutional Neural Networks, Graph Convolutional Networks aim to extract rich features at a vertex by cumulating features of vertices that are present in its neighbourhood. Most Graph Convolutional Networks only update the vertex features at each iteration and tend to have fixed graph structures. Recent work shows that dynamic graph convolution[8] where the graph structure changes in each layer, can learn better graph representations as compared to Graph Convolutional Networks with a fixed graph structure. We see that the dynamically changing neighbours [5] in Graph Convolutional Networks helps mitigate the over-smoothing problem. This also results in a comparatively larger receptive field in the case of Graph Convolutional Networks. The improvement that is suggested is to recompute the edges between the vertices with the help of a Dilated k-NN function in the feature space of each layer to increase the receptive field further. The following are three operations that can enable much deeper Graph Convolutional Networks to be trained:

A. Residual Connections

The ResGraph Convolutional Network is proposed to handle the vanishing gradient problem of Graph Convolutional Networks. The PlainGraph Convolutional Network, which is the baseline model that consists of three blocks: a PlainGraph Convolutional Network backbone block, a fusion block, and an MLP [17] prediction block. The backbone stacks 28 EdgeConv layers with dynamic k-NN, each of which is similar to the one used in DG Convolutional Neural Network. There are no skip connections used here. The ResGraph Convolutional Network is constructed by adding a dynamic dilated k-NN and residual graph connections to the aforementioned PlainGraph Convolutional Network.

B. Dense Connections

DenseNet was proposed to put to use the dense connectivity among the layers of a neural network, which improves information flow in the network. This allows efficient reuse of features amongst the layers. The DenseGraph Convolutional Network is proposed to handle the vanishing gradient problem of Graph Convolutional Networks. The DenseGraph Convolutional Network is built by adding dynamic dilated k-NN and dense graph connections to the PlainGraph Convolutional Network that was previously written about.

C. Dilated Aggregations

The Dilated wavelet convolution is an algorithm that comes from the wavelet processing domain . Dilated convolutions were introduced as an alternative to applying consecutive pooling layers for dense prediction tasks in order to mitigate spatial information loss caused by pooling operations. The experiments demonstrate that aggregating multi-scale contextual information using dilated convolutions can highly increase the accuracy of dense prediction tasks. The reason behind this is that the receptive field is enlarged by dilation without the loss of resolution. Dilation assists or helps the receptive fields of deep Graph Convolutional Networks.

Therefore, dilated aggregation [6] is introduced to Graph Convolutional Networks. Out of the many possible ways, a Dilated k-NN [9] is used to find dilated neighbours using a predefined distance metric after every Graph Convolutional Network layer and construct a Dilated Graph. Thus, by adding skip connections to Graph Convolutional Networks, the difficulty of training can be addressed, which is the major problem of Graph Convolutional Networks to go deeper. Additionally, dilated graph convolutions help to gain a larger receptive field without loss of resolution. Even using a small amount of nearest neighbours, deep Graph Convolutional Networks can achieve high performance.

Another way of letting Graph Convolutional Networks go deeper is to use a differentiable generalized message aggregation function. This defines a family of permutation invariant functions. The definition of such a generalized aggregation function provides a new view of the design of aggregation functions in Graph Convolutional Networks. A new variant of residual connections and message normalization layers are further introduced. The new generalized aggregation function is suitable for Graph Convolutional Networks, as it has a permutation invariant property. The generalized aggregation covers commonly used functions like mean and max in graph convolutions. Additionally, its parameters can be modified to improve the performance of diverse Graph Convolutional Network tasks. This method improves current state-of-the-art performance by 7.8%, 0.2%, 6.7% and 0.9% on the following datasets: ogbn-proteins, ogbn-arxiv, ogbg-ppa and ogbg-molhiv, respectively.

Self-supervision helps improve Graph Convolutional Networks as well. They help in generalizability and they boost Adversarial robustness as well. There are three schemes to incorporate self-supervision into Graph Convolutional Networks. Out of these, multi-task learning seems to work as the regularizer and consistently benefits Graph Convolutional Networks in generalizable standard performances with proper self-supervised tasks. Self-training is restricted in what are the assigned pseudo-labels and what data are used to assign pseudo-labels. We also see that the performance gain is more visible in few-shot learning methods and can diminish with slightly increasing labelling rates. In the case of the second, multi-task learning, self-supervised tasks provide informative and relevant priors which benefit Graph Convolutional Network in generalizable target performance. Node clustering and graph partitioning give priors on node features and graph structures; whereas graph completion with priors on both provides help to a Graph Convolutional Network in context-based feature representation. Third, multi-task self-supervision in adversarial training improves Graph Convolutional Networks robustness against various graph attacks. Node clustering, as well as graph partitioning, give priors on features and links, and thus they defend better against feature attacks and link attacks. Graph completion, with perturbation priors on both features and links, increase the robustness consistently and sometimes hugely for the most damaging feature and link attacks.

IV. APPLICATIONS OF GRAPH CONVOLUTIONAL NETWORKS

Lots of machine learning tasks require dealing with graph data which contains rich relation information among elements. Modelling physics systems, learning molecular fingerprints, predicting protein interface, and classifying diseases require a model to learn from graph inputs. In other domains such as learning from non-structural data like texts and images, reasoning on extracted structures, like the dependency tree of sentences and the scene graph of images, is an important research topic that also needs graph reasoning models. Graph Convolutional Networks (Graph Convolutional Networks) are connectionist models that capture the dependence of graphs via message passing between the nodes of graphs. Unlike standard neural networks, graph neural networks retain a state that can represent information from its neighbourhood with an arbitrary depth. Although the primitive graph neural networks have been found difficult to train for a fixed point, recent advances in network architectures, optimisation techniques, and parallel computation have enabled successful learning with them. On several of the tasks described above, systems based on graph convolutional networks (Graph Convolutional Network) and gated graph neural networks (GGNN) have recently exhibited ground-breaking performance. We present a comprehensive assessment of the applications of graph convolutional networks through adaptations and categorize those applications [3] while giving an in-depth overview of the process and comparison with state-of-the-art models.

A. Graph Convolutional Network approach for decoding EEG Motor Imagery Skills

Brain Control Interface(BCI) applications have been on the rise in the fields of medical engineering. BCIs decode the brain activity so that they can operate devices like artificial limbs and wheelchairs. Electroencephalogram(EEG) has been the go-to procedure for measuring brain activity due to its high resolution, portability and ease of use. EEG based on motor imagery(MI) mentally mimics a variety of motor activities, such as visualising hand or foot movements. The Euclidean structure of EEG electrodes may not effectively reflect and describe the interplay between signals. Traditional Convolutional Neural Network methods do the classification without considering the topological relationship among electrodes. Neuroscience presses on the need for analysing patterns of brain dynamics, Thus Graph Convolutional

Networks were employed to analyse the performance of raw EEG signals on different types of motor imagery tasks while giving equal importance to topological relationships of electrons[12]. The model was built on Pearson's matrix of overall signals, for representing the traditional topological relationship of EEG electrodes a graph Laplacian was built. The Graph Convolutional Networks-Net has an experimentally determined architecture with six convolution and six pooling layers, pooling layers are employed to reduce the dimensionality of the model, the soft plus function is used as activation function for convolution layers, the output layer uses the SoftMax activation function

This particular model used the same dataset as 'PhysioNet', the state of the art model in the given field, while the experiments have shown that at the hundredth subject level the Graph Convolutional Networks-Net model outperformed all the other studies. The Graph Convolutional Networks-Net was able to predict MI tasks with 99.18 maximum accuracy and 96.24 average accuracy, showing the robustness and effectiveness of the proposed model. It, on the other hand, accurately predicted all four MI tasks, the best of which was the two feet prediction, which had a 99.42 percent accuracy. It showed that the proposed technique could provide a generalised representation that was resistant to both personal and group-wise changes. It may be used to decode any EEG MI signals as well as other EEG-based graph-structured data in order to create more effective and efficient BCI systems.

B. BRP-NAS: Prediction-based NAS using Graph Convolutional Networks

In comparison to hand-crafted alternatives, neural architecture search (NAS) has shown remarkable effectiveness in automatically building competitive neural networks. NAS, but on the other hand, is computationally expensive, as it requires the training of models or introduces non-trivial complexity into the search process. Furthermore, in addition to being accurate, real-world deployment necessitates models that satisfy efficiency or hardware limitations (e.g., latency, memory, and energy consumption), yet obtaining different performance metrics of a model can be time consuming, irrespective of the cost of training it. It has been demonstrated empirically that an accurate latency predictor is crucial in NAS when latency on the target hardware is of interest, and conventional latency predictors are excessively error-prone. On a variety of devices, the research offers an end-to-end NAS latency prediction based on a Graph Convolutional Network and shows that it outperforms prior methods (proxy, layer-wise) [13].

A Graph Convolutional Network that learns models for graph-structure data is used in the proposed end-to-end latency predictor. The Graph Convolutional Network predictor comprises four layers of Graph Convolutional Networks, each with 600 hidden units, and a fully connected layer that provides a scalar latency prediction. All predictors are trained 100 times with a randomly sampled set of 900 models from the NAS-Bench-201 dataset each time. The remaining 14k models are utilized for testing, while 100 random models are used for validation.

TABLE I. PERFORMANCE OF LATENCY PREDICTORS ON THE NAS-BENCH-201: OUR GRAPH CONVOLUTIONAL NETWORK PREDICTOR OUTPERFORMS THE LAYER-WISE PREDICTOR ACROSS DIFFERENT DEVICES.

	Accuracy of Graph Convolutional Network Predictor			**Accuracy of Layer-wise predictor**		
Error Bound	Desktop CPU	Desktop GPU	Embedded GPU	Desktop CPU	Desktop GPU	Embedded GPU
±1%	36.0±3.5	36.7±4.0	24.3±1.4	3.5±0.2	4.2±0.2	6.1±0.3
±5%	85.2±1.8	85.9±1.9	82.5±1.5	18.2±0.4	17.1±0.3	29.7±0.8
±10%	96.4±0.7	96.9±0.8	96.3±0.5	29.6±1.1	32.6±1.2	54.0±0.8

Table 1 compares the performance of the proposed Graph Convolutional Network latency predictor to that of the layer-wise predictor on various devices. The percentage of models with predicted latency within the corresponding error bound relative to measured latency is shown by the values. We can observe that the excellent performance is consistent across a variety of devices with radically varying latency characteristics.

C. New Tricks of Node Classification with Graph Convolutional Networks

Methods based on the 3D Morphable Model (3DMM) have had a lot of success reconstructing 3D face forms from single-view pictures. However, the face textures reconstructed using these approaches do not have the same quality as the input pictures. Recent research shows that generative networks can recover high-quality facial textures from a large-scale collection of high-resolution UV maps of face textures, which is difficult to create and not publicly available. This research provides a method for reconstructing 3D facial forms with high-fidelity textures from single-view pictures captured in the wild, without the requirement for a large-scale face texture library, in this work. The basic concept is to use face features from the input image to improve the first texture created by a 3DMM-based technique. Instead of recreating the UV map, we suggest using graph convolutional networks to reconstruct the precise colours for the mesh vertices[14].

This system is made up of three modules, and it provides a coarse-to-fine method for 3D face reconstruction. A Regressor for regressing the 3DMM coefficients, face position, and lighting parameters, and a FaceNet for extracting image features for future detail refining and

identity-preserving are included in the feature extraction module. The texture refinement module is made up of three graph convolutional networks: a Graph Convolutional Network Decoder that decodes FaceNet features and generates detailed colours for mesh vertices, a Graph Convolutional Network Refiner that refines the vertex colours generated by the Regressor, and a combiner that combines the two colours to produce final vertex colours. The Discriminator uses adversarial training to try to enhance the texture refinement module's output.

The results obtained from this research are extremely promising and the Graph Convolutional Network was highly effective in predicting accurate colours and mesh vertices and outperformed other models by far.

D. Point Cloud Upsampling using Graph Convolutional Networks.

As seen above, adaptations like residual connections, dense connections, and dilated convolutions, Graph Convolutional Networks were made to go deeper by getting rid of the gradient problem. With slight adaptations, Graph Convolutional Networks can be used effectively in the task of point cloud upsampling. Deep learning-based approaches for point cloud upsampling do not rely on priors or hand-crafted features to learn how to upsample point clouds, unlike classic optimization-based methods. The use of point clouds to represent 3D data is becoming increasingly common. The rising availability of 3D sensors is contributing to this growing popularity. Such sensors are now an essential component of key robotics and self-driving automobile applications. But due to computational constraints, both in terms of time and space, these 3D sensors often produce sparse and noisy point clouds, which end up portraying evident limitations.

The upsampling modules and feature extractors employed greatly influence the efficacy of learning-based point cloud upsampling processes[10]. An efficient method is one that uses a Graph Convolutional Network (Graph Convolutional Network) to improve the encoding of the local point information from point neighbourhoods. This method has proved to extensively improve state-of-the-art upsampling methods. The other part that needs to be worked on in order to receive an efficient upsampling result would be an improved feature extraction. This is achieved through a multi-scale point feature extractor which is called the Inception DenseGraph Convolutional Network. This performs by aggregating features at multiple scales, leading to better final performance efficiency. When the Inception DenseGraph Convolutional Network is combined with the aforementioned approach, it results in the PU-Graph Convolutional Network. The above adaptation of a novel Graph Convolutional Network was compiled and experimented on a new large-scale dataset PU1K for point cloud upsampling and also a dataset that was 8 times larger than the PU1K dataset and both the results were concurrent. Extensive experiments demonstrate that our proposed PU-Graph Convolutional Network pipeline, which integrates Inception DenseGraph Convolutional Network and a system to better encode local point information from point neighbourhoods, outperforms state-of-the-art methods on PU1K and the other dataset while requiring fewer parameters and being more efficient in inference. It also produces a higher upsampling quality on real-scanned point clouds compared to other methods.

E. Temporal action identification using Graph Convolutional Networks

Temporal action identification [11] is a crucial yet difficult job in video comprehension. Although video context is a crucial signal for efficiently detecting activities, current research focuses mostly on temporal context, disregarding semantic context and other important context features. To attain good efficiency in the aforementioned Temporal action identification, a graph convolutional network (Graph Convolutional Network) model which adaptively incorporates multi-level semantic context into video features and casts temporal action detection as a sub-graph localization problem can be used. In this method, video snippets are defined as graph nodes, snippet-snippet correlations as edges, and context-associated actions are defined as target sub-graphs.

With graph convolutional being the base, a GCNeXt is designed, which learns the features of each node. It does this by aggregating its context and dynamically updating the edges in the graph. Each sub-graph must be localised. In order to do this, an SGAlign layer is introduced to embed each individual sub-graph into the Euclidean space. Experiments show that this method is capable of finding effective video context without extra supervision and achieves more efficient results than state-of-the-art performance at multiple instances. The SGAlign extracts sub-graph features using a set of anchors. SGAlign aligns node characteristics along temporal and semantic graphs and outputs a concatenation of both features. The order of nodes is maintained in the final representation when utilising the temporal graph. Since node features are represented by their feature neighbours, this isn't necessarily true for the semantic network. In the GCNeXt block, temporal and semantic networks of the same cardinality process the input feature. In each box, we show the (input channel, output channel). Both convolution streams use a split-transform-merge method with 32 pathways to boost transformation variety. The total of both streams and the input is the output of the module.

F. Graph Mining

Graph mining is a set of tools and techniques for analysing the properties of real-world graphs, forecasting how the structure and properties of a given graph might affect a given application, and creating models that can generate realistic graphs that match the patterns found in real-world graphs of interest. Graph mining techniques [2] are often used to find valuable structures for later tasks. Frequent subgraph mining, graph matching, graph classification, graph clustering, etc., are some traditional graph mining challenges. Although certain downstream tasks may be addressed directly using deep learning without the need for graph mining, the fundamental problems are worth investigating from the standpoint of GNNs.

Let us consider some challenges. The first challenge is graph matching. Traditional methods for graph matching usually suffer from high computational complexity, both in terms of time and space constraints. GNNs allow researchers to use neural networks to capture the structure of graphs, providing yet another answer to the challenge. A Siamese MPNN model was proposed to learn the graph editing distance. It consists of two parallel MPNNs with the same structure and weight sharing. The goal of the training is to embed a pair of graphs with a short editing distance into a small amount of latent space. While tests were carried out on more real-world circumstances, such as a similarity search in a control flow graph, several comparable approaches were created. Graph clustering is the second challenge, which involves grouping the vertices of a graph into clusters based on the graph topology and/or node characteristics. Various node representation learning works have been created, and the node representation may be given to classic clustering methods. Graph pooling, in addition to learning node embeddings, may be thought of as a form of clustering. They look at what makes a successful graph clustering technique desirable and offer ways to improve the spectral modularity metric, which is a very useful graph clustering metric. Graph Convolutional Networks, therefore, assist us in overcoming a variety of difficulties in addition to those mentioned above. This results in a significant boost in efficiency and productivity.

REFERENCES

[1]. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (references)
[2]. Li, Yujia et al. "Graph Matching Networks For Learning The Similarity Of Graph Structured Objects". PMLR, 2019.
[3]. Zhou, Jie et al. "Graph Neural Networks: A Review Of Methods And Applications". AI Open, vol 1, 2020, pp. 57-81. Elsevier BV, doi:10.1016/j.aiopen.2021.01.001.
[4]. Li, Guohao et al. "Deepgcns: Can Gcns Go As Deep As Cnns?". Arxiv.Org, 2019
[5]. Li, Guohao et al. "Deepergcn: All You Need To Train Deeper Gcns". Arxiv.Org, 2020, https://arxiv.org/abs/2006.07739v1. Accessed 30 June 2021.
[6]. Li, Guohao et al. "Deepgcns: Making Gcns Go As Deep As Cnns". IEEE Transactions On Pattern Analysis And Machine Intelligence, 2021, pp. 1-1. Institute Of Electrical And Electronics Engineers (IEEE),
[7]. Zhang, Si et al. "Graph Convolutional Networks: A Comprehensive Review". Computational Social Networks, vol 6, no. 1, 2019. Springer Science And Business Media LLC.
[8]. Manessi, Franco et al. "Dynamic Graph Convolutional Networks". Pattern Recognition, vol 97, 2020, p. 107000. Elsevier BV
[9]. "Dynamic K-Nearest-Neighbor With Distance And Attribute Weighted For Classification". Ieeexplore.Ieee.Org, 2021
[10]. Qian, Guocheng et al. "PU-Graph Convolutional Network: Point Cloud Upsampling Using Graph Convolutional Networks". Arxiv.Org, 2019, https://arxiv.org/abs/1912.03264
[11]. Zeng, Runhao et al. "Graph Convolutional Networks For Temporal Action Localization". Openaccess.Thecvf.Com, 2019
[12]. Lun, Xiangmin et al. "Gcns-Net: A Graph Convolutional Neural Network Approach For Decoding Time-Resolved EEG Motor Imagery Signals". Arxiv.Org, 2020,
[13]. Dudziak, Łukasz et al. "BRP-NAS: Prediction-Based NAS Using Gcns". Arxiv.Org, 2020.
[14]. Lin, Jiangke et al. "Towards High-Fidelity 3D Face Reconstruction From In-The-Wild Images Using Graph Convolutional Networks". Arxiv.Org, 2020.
[15]. Lin, Jiangke et al. "Towards High-Fidelity 3D Face Reconstruction From In-The-Wild Images Using Graph Convolutional Networks". Arxiv.Org, 2020
[16]. S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017
[17]. Singh and M. Sachan, "Multi-layer perceptron (MLP) neural network technique for offline handwritten Gurmukhi character recognition," 2014 IEEE International Conference on Computational Intelligence and Computing Research, 2014, pp. 1-5, doi: 10.1109/ICCIC.2014.7238334.