# Graph Neural Networks (GNNs)

## Why is it Hard to Analyze a Graph?

Graph-based data structures have drawbacks, and data scientists must understand them before developing graph-based solutions.

1. A graph exists in non-euclidean space. It does not exist in 2D or 3D space, which makes it harder to interpret the data. To visualize the structure in 2D space, you must use various dimensionality reduction tools.
2. Graphs are dynamic; they do not have a fixed form. There can be two visually different graphs, but they might have similar adjacency matrix representations. It makes it difficult for us to analyze data using traditional statistical tools.
3. Large size and dimensionality will increase the graph's complexity for human interpretations. The dense structure with multiple nodes and thousands of edges is harder to understand and extract insights.

## What is a Graph Neural Network (GNN)?

Graph Neural Networks are special types of neural networks capable of working with a graph data structure. They are highly influenced by Convolutional Neural Networks (CNNs) and graph embedding. GNNs are used in predicting nodes, edges, and graph-based tasks.

- **CNNs** are used for image classification. Similarly, GNNs are applied to graph structure (grid of pixels) to predict a class.
- **Recurrence Neural Networks** are used in text classification. Similarly, GNNs are applied to graph structures where every word is a node in a sentence.

GNNs were introduced when Convolutional Neural Networks failed to achieve optimal results due to the arbitrary size of the graph and complex structure.

The input graph is passed through a series of neural networks. The input graph structure is converted into graph embedding, allowing us to maintain information on nodes, edges, and global context.

Then the feature vector of nodes **A** and **C** is passed through the neural network layer. It aggregates these features and passes them to the next layer -

## Types of Graph Neural Networks

There are several types of neural networks, and most of them have some variation of Convolutional Neural Networks. In this section, we will be learning about the most popular GNNs.

- G**raph Convolutional Networks (GCNs)** are similar to traditional CNNs. It learns features by inspecting neighboring nodes. GNNs aggregate node vectors, pass the result to the dense layer, and apply non-linearity using the activation function. In short, it consists of Graph convolution, linear layer, and non-learner activation function. There are two major types of GCNs: Spatial Convolutional Networks and Spectral Convolutional Networks.
- **Graph Auto-Encoder Networks** learn graph representation using an encoder and attempt to reconstruct input graphs using a decoder. The encoder and decoders are joined by a bottleneck layer. They are commonly used in link prediction as Auto-Encoders are good at dealing with class balance.
- **Recurrent Graph Neural Networks(RGNNs)** learn the best diffusion pattern, and they can handle multi-relational graphs where a single node has multiple relations. This type of graph neural network uses regularizers to boost smoothness and eliminate over-parameterization. RGNNs use less computation power to produce better results. They are used in generating text, machine translation, speech recognition, generating image descriptions, video tagging, and text summarization.
- **Gated Graph Neural Networks (GGNNs)** are better than the RGNNs in performing tasks with long-term dependencies. Gated Graph Neural Networks improve Recurrent Graph Neural Networks by adding a node, edge, and time gates on long-term dependencies. Similar to Gated Recurrent Units (GRUs), the gates are used to remember and forget information in different states.

## Types of Graph Neural Networks Tasks

Below, we’ve outlined some of the types of GNN tasks with examples:

- **Graph Classification**: we use this to classify graphs into various categories. Its applications are social network analysis and text classification.
- Node Classification: this task uses neighboring node labels to predict missing node labels in a graph.
- **Link Prediction**: predicts the link between a pair of nodes in a graph with an incomplete adjacency matrix. It is commonly used for social networks.
- **Community Detection**: divides nodes into various clusters based on edge structure. It learns from edge weights, and distance and graph objects similarly.
- **Graph Embedding**: maps graphs into vectors, preserving the relevant information on nodes, edges, and structure.
- **Graph Generation**: learns from sample graph distribution to generate a new but similar graph structure.