What are GNN and what are its possible applications?

Graph neural networks are special kind of deep neural networks that demonstrate real world problems into graphical(edges, nodes) domains and do retain information from its neighborhood

With arbitrary depth. GNN addresses several problems of traditional methods like arbitrary size, complex topology and data structures' shortcomings that are normally more suited to graph models. Wide range of GNN applications persist: Node classification(fraudulent customer), link prediction(social networks friendship), Face recognition, Recommender systems and many more.

What is the difference between GNN, GCN, CNN and FCNN?

In FCNN all the neurons or nodes of one layer are connected to every neuron to the next layer. A fully connected layer accepts weighted sum from each node in the layer. On the other hand, convolution neural nets extract the best features from the available input features hence does the classification task using much lower number of parameters than FCNN by using the operation convolution. GNN is the generic term of deep learning using graphical representations as mentioned in the first answer containing approaches like GCN, GAT, GraphSage and more. GCN is the strategy in GNN domains which embeds nodes by using symmetrically normalized graph Laplacian while methods like aggregation of neighborhood and combine information are in place.

How can we insert prior knowledge (i.e. filters) into the learning process?

From the exemplary implementation given, we can see that we can insert the prior knowledge in the form of an adjacency matrix which contains the features in terms of pixel value ranging from 0 to 1. It is just one method there are other methods as well to feed prior knowledge e.g. node embeddings via deepwalk, randomwalk, node2vec etc. After applying a simple Laplacian filter the output for the layer would be:

$$X^{(l+1)} = \widetilde{D}^{-1}\tilde{A}\widetilde{D}^{-1}X^{l}W^{l}$$

where $\widetilde{D}$ is the Degree matrix of the vertices and $\tilde{A}$ being the adjacency matrix after normalization.

What are the possible methods for implementing GCNs?

First of all graph structure is normalized while having node features in a usable form. Then the graph convolution layer which passes a filter over the nodes and extracts essential features. Finally some sort of activation function (Leaky ReLu) and dropout layers that perform a sort of pooling/downsampling over the first convolution.

How does your adjacency matrix work and why did you choose it?

I have chosen DeepWalk as a method of encoding prior knowledge in the learning process. DeepWalk empirically produces a low-rank transformation of a network's normalized adjacency matrix. DeepWalk uses short random walks to learn representations for vertices in graphs. Generic mathematical representation of deepwalk is as follows:

$$log\left(vol(G)\left(\frac{1}{T}\sum_{r=1}^{T}(D^{-1}A)^{r}\right)D^{-1}\right) - \log(b)$$

Where, T & b represents window size and number of negative sampling. D represents degree of the nodes, A is the adjacency matrix and vol(G) is the volume of the graph.