GNN questions for PetarV (answered offline)

- Do you recommend any GNNs for heterogenous graphs? Namely graphs with different edge types. How to aggregate neighbor information when the edge types are different?
    - Both MPNNs and GATs are easily extendable to handle heterogeneous graphs -- simply learn one kind of attention mechanism / message function per edge type. In principle it gets more interesting than this: see **R-GCNs** for an extension of GCNs to such data -- many subsequently proposed architectures leverage similar construction ideas. One example I'm aware of recently, which extends a GAT-like model for knowledge graph reasoning, is the **GraIL** model.

- How to deal with really big graphs, when the adj. matrix is too big to be stored? Can we do GCN in a mini-batch fashion?
    - Yes! The **GraphSAGE** work presented in the second half of my lecture directly details one such approach (select a mini-batch of nodes, then randomly subsample neighbours and neighbours' neighbours as a subgraph to run a GCN over).

- Can you suggest any standard libraries and datasets for getting started with GNNs?
    - Of course! For libraries, if PyTorch is your thing I would highly recommend either **PyTorch Geometric** or the **Deep Graph Library**. Both of these have a very rich model zoo (especially PyG), and come with many standard or non-standard datasets pre-packaged. For TensorFlow, popular choices are **GraphNets** and **Spektral**. Regarding datasets, it is mostly a question of whether you'd like to just quickly get a GNN model up-and-running, or actually do meaningful benchmarking. For getting started quickly, the standard Cora/Citeseer/Pubmed datasets are now established as the "MNIST" of the graph domain, and are small enough to even be handled on CPU. The flipside is that their edges are not highly "interesting" for GNNs to process, hence most GNN layers proposed in recent times will perform more-or-less the same on them. For meaningful benchmarking, I would recommend either the **Benchmarking-GNNs** effort or the **Open Graph Benchmark**. The latter is well on its way to becoming the "ImageNet for graphs".

- Are there any backpropagation in Graph Attention Networks?
    - Yes, GATs perform backpropagation much like any other feedforward neural net -- all operations within it are fully differentiable w.r.t. node, graph or edge-level predictions. The computational / backpropagation graph of GATs is highly similar to that of masked Transformers.

- How about using GNNs for solving TSP?
    - A very **exciting** area, and one I'm highly enthusiastic about! Powerful GNN-TSP approaches exist both for **reinforcement learning** and **supervised learning** -- I recommend starting with those.

- How much harder (in terms of the dataset dimensions) is it to apply graph attention network to regression problems (ex. probability prediction) instead of classification tasks?

    - Hard to quantify, I’m afraid -- and likely to be task-specific :)

- How would you re-design your model in which most graph nodes represent attributes, while only in some of them we try to make predictions?
    - Easiest way to do this is to predict in **every** node, then **mask out** the loss function (at training time) or evaluation metrics (at test time) to only account for the nodes where you’d like to make predictions.

- Does graph structure has importance in GNNs? Or is it enough to set the graph to be complete and let the network deside the weights?
    - That would highly depend on the application. A fully-connected GNN can in principle model anything -- but it may get much harder for it to do so. For some tasks -- esp on small graphs -- there is no real difference. But sometimes your task _is_ structural (e.g. counting the number of triangles in the graph) -- for these tasks a fully-connected GNN would struggle. Further, fully-connected GNNs can suffer from *oversmoothing* (having too many neighbours means that it becomes much harder to isolate significant messages, and most node features then end up converging to more-or-less equal vectors), and more importantly, they require quadratic (O(n^2)) memory to store, making them prohibitively expensive for graphs in excess of 100,000 nodes or so.

- Why message passing is ubiquitous in GNN and not in other sub-fields?
    - The concept of message passing is not at all isolated to GNNs -- in fact, one major inspiration direction for GNNs came from **probabilistic graphical models** (PGMs) where the concept of message passing can be deeply traced back to. See, for example, the **belief propagation** algorithm.

- Wouldn't using hypergraphs make the network be better? Why stop at dim=2?
    - In principle, yes. But there are scalability issues around inducing hypergraphs (which most existing hyper-GNN models have to carefully account for) and it turns out that in practice, most hyper-edges can be effectively *decomposed* into a stacked sequence of binary relations, meaning that hyper-GNNs don’t often improve on standard GNNs in practice -- at least on the benchmarks popular so far.

- How GNNs and RL could be combined? Are there already some applications?
    - Yes, and this is a **very** exciting area! I’d like to partition it into methods that either represent the agent as a graph (e.g. **NerveNet**), the environment as a graph (e.g. applications to **TextWorld**), or even the task that needs to be executed as a graph (*[citation needed]* :)).

- Are there any methods that are designed to deal with (and exploit) particular graph structures e.g. small world graphs?
    - Yes -- recently, **substructure-counting methods** seem to be an interesting proposed method for improving graph neural net expressivity that directly count interesting substructures in the data (assuming the labels are somehow related to these). But the literature on this is vast, and I’m by no means an expert :)

- How is a graph net more powerful than an MLP? Can't an MLP learn the edge weights and thus the internal graph structure?
    - A graph neural net is _not_ more powerful than an MLP. However its inductive biases allow it to adapt to graph-structured data much more efficiently (requiring substantially fewer training data samples) than an MLP. For a thorough theoretical discussion of this, consider **What can neural networks reason about?** by Xu et al.

- How to apply Graph Attention Networks in a weighted graph?
    - I see two possible means of doing so: either scale your (unnormalised) attention weights by the given edge weights, or pass the weight as an additional parameter to the attention mechanism.

- What are some new cool applications of graph networks?
    - Hopefully covered within the lecture :)

- How to efficiently apply Graph Attention Layers, given that each node may have different number of neighbours? We don't wanna use cycles. Is masking utilized?
    - Under some assumptions on the attention function, GATs can be efficiently implemented using sparse matrix multiplication operations. See for example: https://github.com/PetarV-/GAT/blob/master/utils/layers.py#L36-L85

- How is input and processing conceptually separated in GNNs? Is there a clear destinction as there is for image processing, where the first layer is the input-image and the next layers are processing it?
    - Exactly. With some exceptions (such as spatio-temporal graphs), most commonly you can think of GNNs as exact analogues of CNNs -- the input layer is the original graph's features, and then the following layers serve to progressively process it, with varying degrees of flexibility.

- Could these models be improved by attending over larger neighborhoods in the graph? Looking at neighbors of your neighbors?
    - This used to be a very popular approach in the earlier years of GNN processing, but nowadays the more common approach is to stack multiple 1-hop GNN layers and thus extract similar kinds of "receptive fields" while not having to deal with increasing neighbourhood sizes.

- What do we get after training a MPNN model over a graph? Do we have an "encoder" that we can use later on to embed another graph?
    - Exactly, we gain a GNN-based encoder function which can provide us node/edge/graph embeddings for an entirely new graph.

- Can be a convolution applied in a spectral domain of a graph?
    - Exactly so! In fact, most of the earlier work on GNNs almost exclusively relied on the "graph fourier transform", which relies on the eigendecomposition of the graph Laplacian matrix, which can be used to redefine convolution as node-wise multiplication (as you would in classical DSP). Here's a few references:
        - https://arxiv.org/abs/1312.6203

        - https://arxiv.org/abs/1606.09375
        - https://arxiv.org/abs/1705.07664
        - https://arxiv.org/abs/1904.07785

- Can the same GNN architecture be used to learn multiple different topologies such that the number of nodes and connections are different? Or we need to have separate models?
    - If the GNN is **inductive**, then it can be applied across many topologies. GATs and MPNNs are examples of inductive models.

- Some graph algorithms, such as PageRank, are also work on passing messages between nodes. What's the added value of GNN comparing to them?
    - PageRank has historically been a very powerful way to obtain graph representations. However, its Markovian assumption and basis on random walk iterations mean that it is provably less expressive than a generic GNN. Many ideas from PageRank have made their way into GNN layers, however, with very successful outcomes either in predictive power or scalability. The most famous example I can think of is the **APPNP** model, which directly incorporates personalised PageRank into a GNN.

- Is message passing done for one time step in GAT? Or for k steps or until convergence? What are the implications of this choice?
    - Message passing is done for K steps, where K is a hyperparameter. It is analogous to the depth of a CNN, and accounts for how “deep” features we would like the GNN to compute. Older variants of GNNs (e.g. original work by Scarselli et al.) iterated until convergence, but this limited the applicability of the trained model.

- How can we interpret GNNs?
    - A very important and emerging topic recently! One way is to ***highlight*** the nodes and edges most relevant to making a particular prediction. See **GNNExplainer** and **GraphLIME** for two prominent examples.

- How do you do progressive negative sampling? How do you automatically determine which examples will be more difficult for the NN?
    - Please see the original **PinSage** paper for more details -- but in a nutshell, the **personalised PageRank score** is used as a proxy for “difficulty” of a sample.

- How might GNNs deal with hypergraphs, i.e. hyperedges?
    - Two simple ways (many others are proposed in literature) are:
        - Add a binary edge between all pairs of vertices incident to a hyper-edge
        - Have a “master node” for each hyper-edge, and connect it with binary links to the constituent nodes.

- Is GNN good for routing problems?
    - This would depend on what we refer to as “good”, but GNNs have recently seen a lot of investigations, for example, for the **travelling salesman problem**.

- Is there any method that works with Dynamic networks or evolving networks? maybe something like temporal GNNs? (and many related questions)
    - Yes! GNNs can be applied to a wide variety of these, **spatio-temporal, graphs**. The key idea (on which most existing research rests) is to have a *spatial GNN* that processes the graph at each point in time, with a *temporal* component that aggregates this information across time (to build up a sort of latent "history" of states). Popular models so far combine diffusion-convolutions with GRUs (**DCRNN**), GATs with GRUs (**GaAN**), GCNs with convolutions (**STGCN**), GCNs with dilated convolutions (**Graph WaveNet**) or even GATs with Transformers (**STGRAT**).

- transformers can be seen as GNN . Does this mean transformers with multiple head attention equivalent to representational power of GNN and used interchangeably
    - The **equivalence** is still an open problem (whether GATs are less expressive than MPNNs). GNNs are definitely a superset of Transformers, however. :)