**Project Report – Structure**

1. **Title**

**Algorithm name, title definition**

Learning convolutional neural networks for graphs

A framework for learning convolutional neural networks for graphs

1. **Reference**

**A reference to the paper used**

Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov. "Learning convolutional neural networks for graphs." *International conference on machine learning*. 2016.

1. **Motivation**

**What is the problem the algorithm/architecture is trying to solve?**

The “PATCHY-SAN” algorithm and framework aim to solve classification and regression problems of unseen graphs. It can be used for datasets that can be represented as graphs with multiple attributes on their nodes and edges, for example, classification of chemical compounds and communities in social networks.

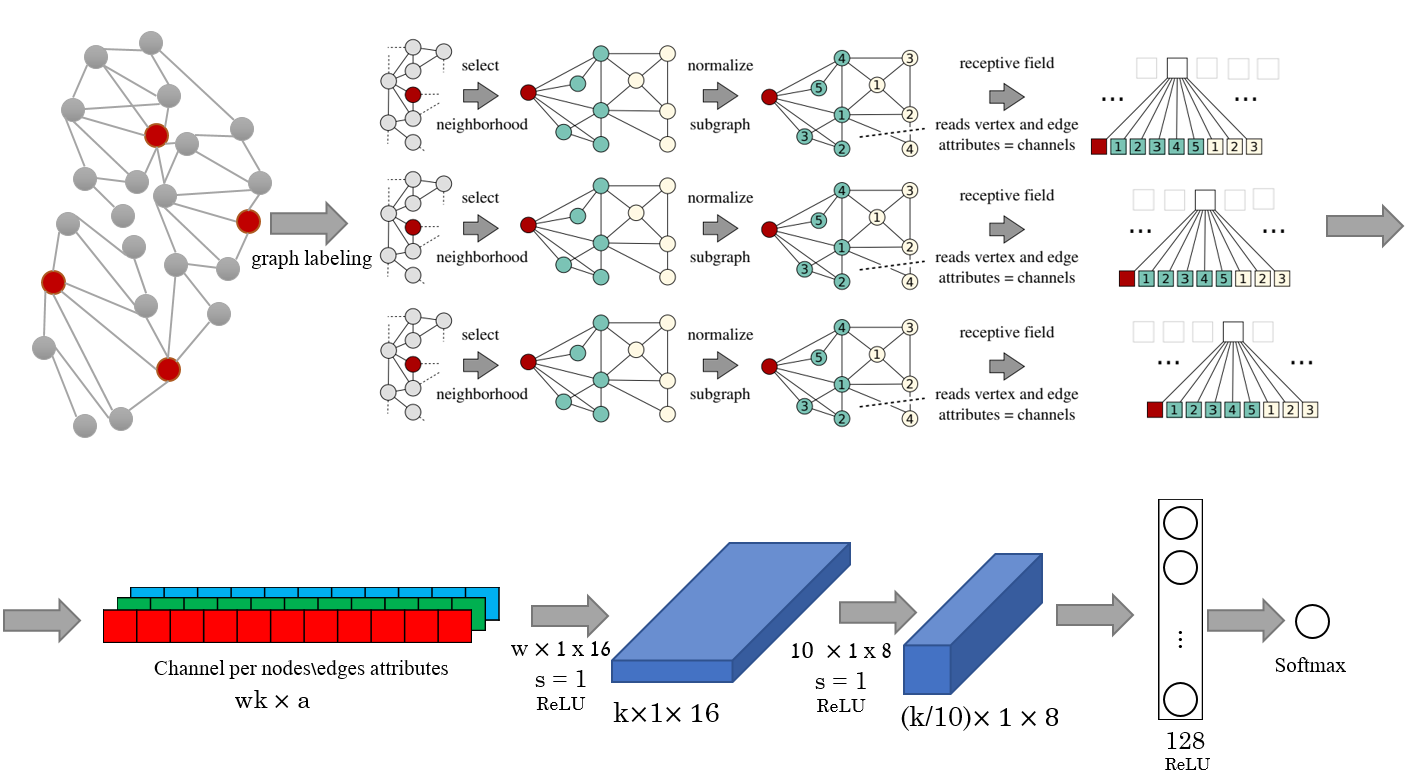
1. **Short description**

**Describe the algorithm/architecture. Description should be about a paragraph.**

The algorithm generalizes CNN for graphs problems. It firsts, deals with graphs normalizations, since unlike images, graphs do not have a natural order (like pixel locations). In order to build receptive fields (CNN filters), the algorithm selects a sequence of nodes to focus on, for each it extracts it’s graph’s neighborhood and normalize the extracted graph using graph canonization algorithm. Later, it learns this graph representation using convolutional neural network with two layers of 1-dimentional convolutions, two dense layers and a sigmoid.

1. **Architecture**

**Describe the architecture used in detail. Provide a diagram (or diagrams) describing the architecture structures. Describe the model hyper-parameters (both the ones stated in the paper and the ones used in your experiments, where there changes): initial learning rate, learning rate decline, non-linear functions, loss function, number of training epochs etc.**



* 1. Graphs preprocessing

The graph preprocessing is built from few steps that aims to capture the graphs structure across different examples. The steps are: Nodes sequence selection, Neighborhood assembling and Graph normalization

1. Node sequence selection – this step aims to identify for each graph, a sequence of nodes ordered by their importance, for which a receptive field are created.

The nodes are selected according to a given graph labeling function. A labeling function is a function from the set of vertices to an ordered set (sequence). For example, nodes degree and nodes centrality measures. The labeled sequence is than traversed using stride , until exactly receptive fields are created. If needed, zero padded receptive fields are created.

1. Neighborhood assembling – for each node that was selected in the previous step, a receptive field is created by performing BFS (breadth first search), and at each distance the nodes are added to a set , until size of is a least .
2. Graph normalization – each neighborhood graph is than normalized, i.e. ordered by a canonization algorithm such that nodes of two different graphs will have a similar relative position in the normalized order if and only if they have the same structural roles within the two different graphs. The graph canonization algorithm is 1-dim Weisfeiler-lehman.
   1. CNN architecture

For each input graph G, it applies normalized receptive fields, size for vertices or size edges that results and a tensors. and are the number of input channels, in case that each node or edge has more than one attribute.

In case of edges, we transform the nodes graph into edges graph (aka line graph)

Those tensors are now applied with 1-dimensional convolutional layer with stride and receptive field size to the first and to the nodes graphs and edges graph receptively.

Merge layers are used to combine between input of nodes graphs and edges graphs in needed. The rest of the architecture can be chosen arbitrarily.

* 1. Experiment

In our experiment, like in the paper, we built a CNN with 2 convolutional layers and 1 dense layer and a softmax for all experiments. First, receptive field size and . The number of receptive fields, , is set as the average number of nodes. 1-dimensional Weisfeiler-Lehman (1-WL) is used as the graph canonization algorithm. In addition, we ran experiments for k = 10 where we combined receptive fields for nodes and edges using a merge layer (k = 10E ). As part of our innovation in this project, we also ran additional experiment, where we built additional channels which represent different node’s graph kernels.

The first convolutional layer had 16 output channels. The second convolutional layer has 8 output channels, a stride of s = 1, and a field size of 10. With ReLU activation function. The dense layer has 128 neurons with ReLU and a dropout rate of 0.5. The Dropout and the small number of neurons are needed to avoid overfitting on the smaller data sets. The hyperparameter are number of epochs and batch size for the mini-batch RMSPROP algorithm.

1. **Dataset**

**A short description of the dataset used (size, number of classes, class ratios etc.).**

e 6 standard benchmark data sets to compare run-time and classification accuracy

**MUTAG -** (Debnath et al., 1991) is a data set of 188 nitro compounds where classes indicate whether the compound has a mutagenic effect on a bacterium.

**PTC -** consists of 344 chemical compounds where classes indicate carcinogenicity for male and female rats (Toivonen et al., 2003).

**NCI1 -**  (and **NCI109**) are chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell lines (Wale & Karypis, 2006).

**PROTEINS -** is a graph collection where nodes are secondary structure elements and edges indicate neighborhood in the amino-acid sequence or in 3D space. Graphs are classified as enzyme or non-enzyme.

**D&D -**  is a data set of 1178 protein structures (Dobson & Doig, 2003) classified into enzymes and non-enzymes.

additional experiments with on larger social network graph data sets (up to 12000 graphs each, with an average of 400 nodes), where normalized node degree as nodes attribute.

**COLLAB, IMDB-B, IMDB-M, RE-B, RE-M5k, RE-M10k**

1. Results

The results should include the following:

* The final results on the test set
* Training/validation set loss and accuracy (present ~30 data points for the training loss throughout the experiment). If the paper you’re implementing applies a different metric, use that. In case of uncertainty, contact the lecturer.
* Test set loss and accuracy for each training epoch.

1. Results analysis

Where did your model do well? Could you identify cases where the performance was good/poor? Were there differences between your results and the reported results? If so, what?

1. **Conclusions**

**What are the conclusions you can draw from your experiments?**

Generalizing convolutional neural network for graphs classification problems works very good and outperforms the graph kernels methods that were the state of the art when the paper was published.