Learning representation of graph layouts from graph layout images

Hammad Haleem

Agenda

- Aim of the project
- Overview of Current Progress
  - **Experiment 1:** Learning graph representations using deep learning
  - **Experiment 2:** Using latent graph features for layout prediction

Project AIM

- To explore how to make use of CNN models, for learning from graph images and use latent image features for graph rendering

- I designed the following two experiments:

    - **First:** To make use of graph images to predict specific features, and use an empirical formula to evaluate if the graph is correctly rendered or not.

    - **Second:** Make use of graph data, i.e., a set of nodes and edges and use the latent features to estimate a graph layout.

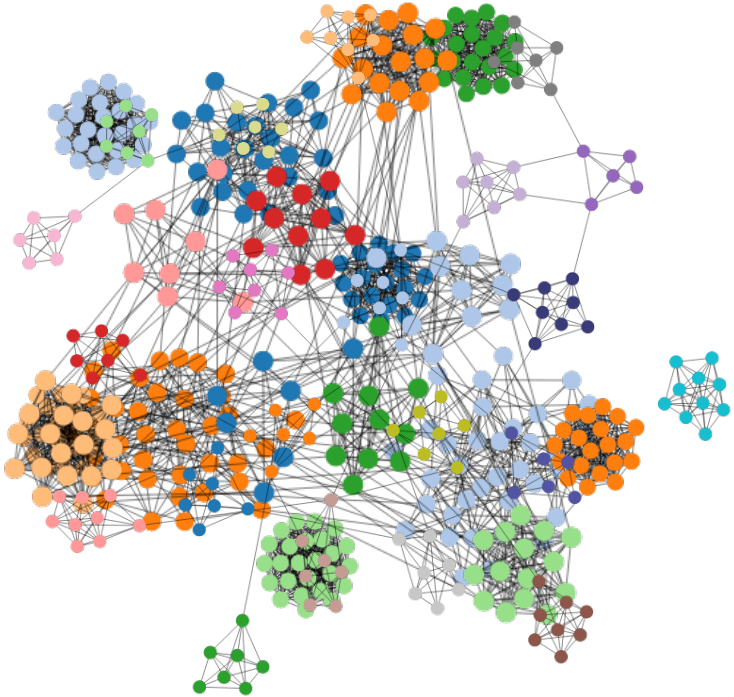
Overview : Experiment 1

I followed following steps in Experiment 1

1) Generate multiple synthetic graphs.
2) Use benchmarked rendering algorithms to render these graphs.
3) Label each graph image using extracted features.
4) Build and train a CNN model, to learn these features.
5) Design metric for finding a good graph layout, using image features.

I have discussed each step in detail in next few slides.

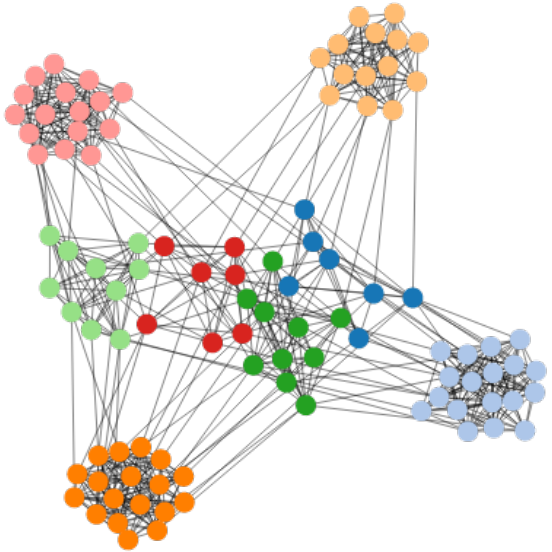
Sample Input images



We generated more than 80,000 such images, making use of force directed layout algorithm with different forces.

Features and their significance

- We labelled each image, using both aesthetic and structural features. An example from the dataset if given below.



node_overlap	4
number_of_edges	670
number_of_crossings	8238
number_of_nodes	100
number_of_edge_crossings_outside_community	1295
number_of_communities	8
average_node_distance_from_center	30.15
max_node_distance_from_center	36.13
min_node_distance_from_center	26.96
median_node_distance_from_center	29.27

Train a CNN on this data

- Create a model to learn the above-mentioned features from an input image.
- In the first step, I modified VGG (a very popular and efficient deep learning model) to train on this problem.
- VGG was too large for the task (didn’t converge after days of running) so I created a smaller model with a lesser number of weights to train.

Current CNN model

- Input : 324 x 270 pixel image
- Convolution layers
  - Conv 32 channels , 3 x 3 filter
  - Conv 32 channels , 3 x 3 filter
  - Conv 64 channels , 3 x 3 filter
  - Conv 128 channels , 3 x 3 filter
- Fully connected layers
  - FC 1 : 2048 neurons
  - FC 2 : 2048 neurons
  - FC3 : 1024 neurons
- Output layers
  - 4 output neurons

- **Evaluation metrics**
  - Model losses started from 10^6 and dropped to 4-5 points for four regression variables.
  - The models does a good prediction of all major input variables.

Design a metric for finding if the rendered image of the graph is good or not?

- Based on features for each image, design a metric to label, if the graph rendering is good or not?
- We need to take into account factors like
  - Node occlusions
  - Edge overlapping
  - Inter-community distance.
- I would like to have feedback on possible measures for a good graph layout. *How to quantitatively say if a graph layout is good or not ?*

Experiment 2

AIM

- Design a deep learning model to draw a graph layout given an input edge list, and model trained on images of similar graphs.

- Make use of both the latent features of a graph as well as available images to draw an approximate graph.

    - Extract latent features from an edge matrix, and use it for visualization.

Algorithm Stage I

Graph image

Connection information

Model (Node2Vec)

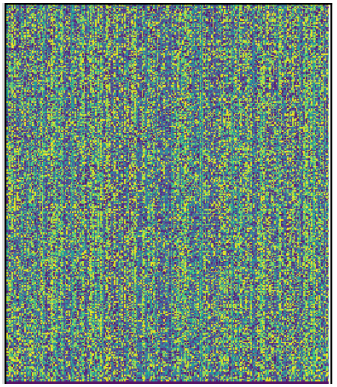
Matrix (M + x + y) x N

N is number of node
M is number of features
x,y position of each node.

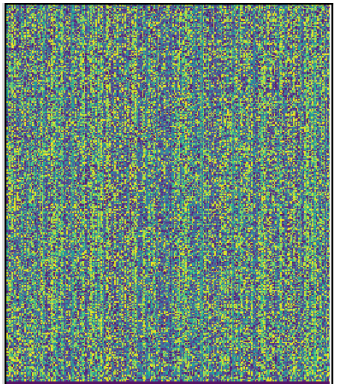
Converting this matrix to an image that can be input to a CNN

X1, Y1, color1
X2, Y2, color2
X3, Y3, color3
X4, Y4, color4

.
.
.
.
.
.
.
.
Xn, Yn, color n



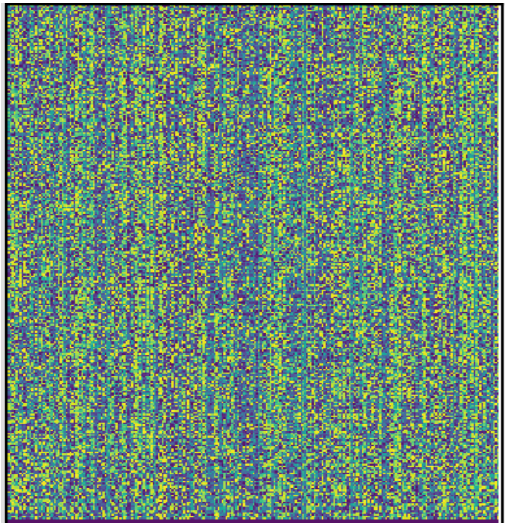
Augmented latent representations



- This image represents the latent information present in a graph.

- Augmented with X,Y co-ordinates that were extracted from a force directed layout of the same graph.
    - I used the same dataset; I generated before to construct these latent representations. With more than 70,000 such representations I was able to train a prediction model.

Algorithm stage II



X1, Y1, Z1
X2, Y2, Z2
X3, Y3, Z3
X4, Y4, Z4

.
.
.
.
.
.
.
.
.
.
.
.
.

Xn, Yn, Zn

Convolutional layers

FC layer 1

FC layer 2

X Y Z

Results & Evaluation

- The loss for this computation is still, quite high and I am still running a variety of variations of this model to find the best fit.

- As Yong, mentioned he is going to be experimenting on similar lines using an RNN model, RNN might be able to give better results.

- *I would love to have a feedback on how to improve this model.*

Thanks!

Any questions ?