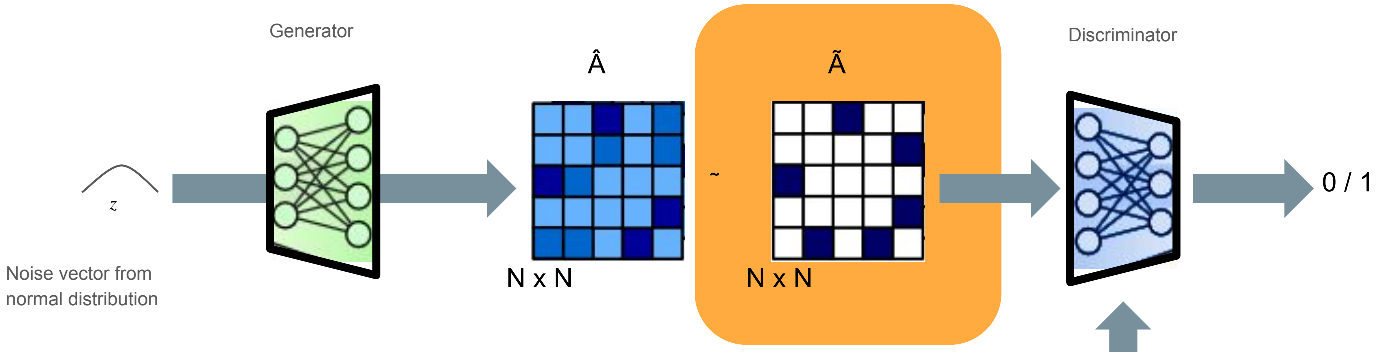
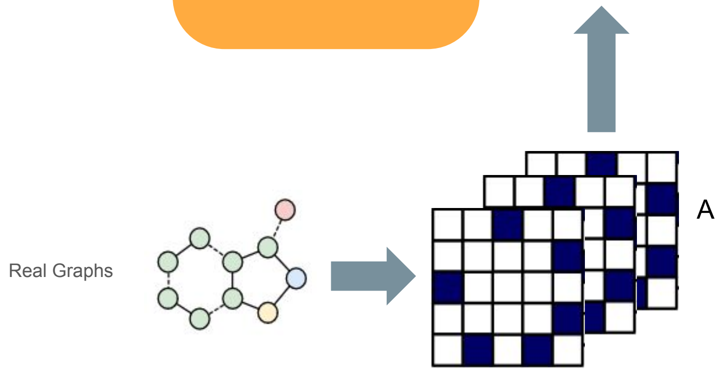
Updated Classifiers

Contamination (IF)	Datasets	GLADC	IF (400 trees)	RF (400 trees)	DT
0.17	MMP	**0.696 ± 0.042**	0.457 ± 0.007	0.371 ± 0.003	0.423 ± 5.551
0.05	HSE	**0.618 ± 0.110**	0.439 ± 0.007	0.555 ± 0.011	0.512 ± 0.000
0.08	p53	**0.649 ± 0.216**	0.483 ± 0.005	0.502 ± 0.005	0.589 ± 0.000
	BZR	**0.715 ± 0.067**			
	DHFR	**0.612 ± 0.041**			
	COX2	**0.615 ± 0.044**			
	ENZYMES	**0.583 ± 0.035**			
	IMDB	**0.656 ± 0.023**			
	AIDS	0.993 ± 0.005			
0.49	NCI1	0.683 ± 0.011	0.485 ± 0.014	0.693 ± 0.017	0.663 ± 0.013

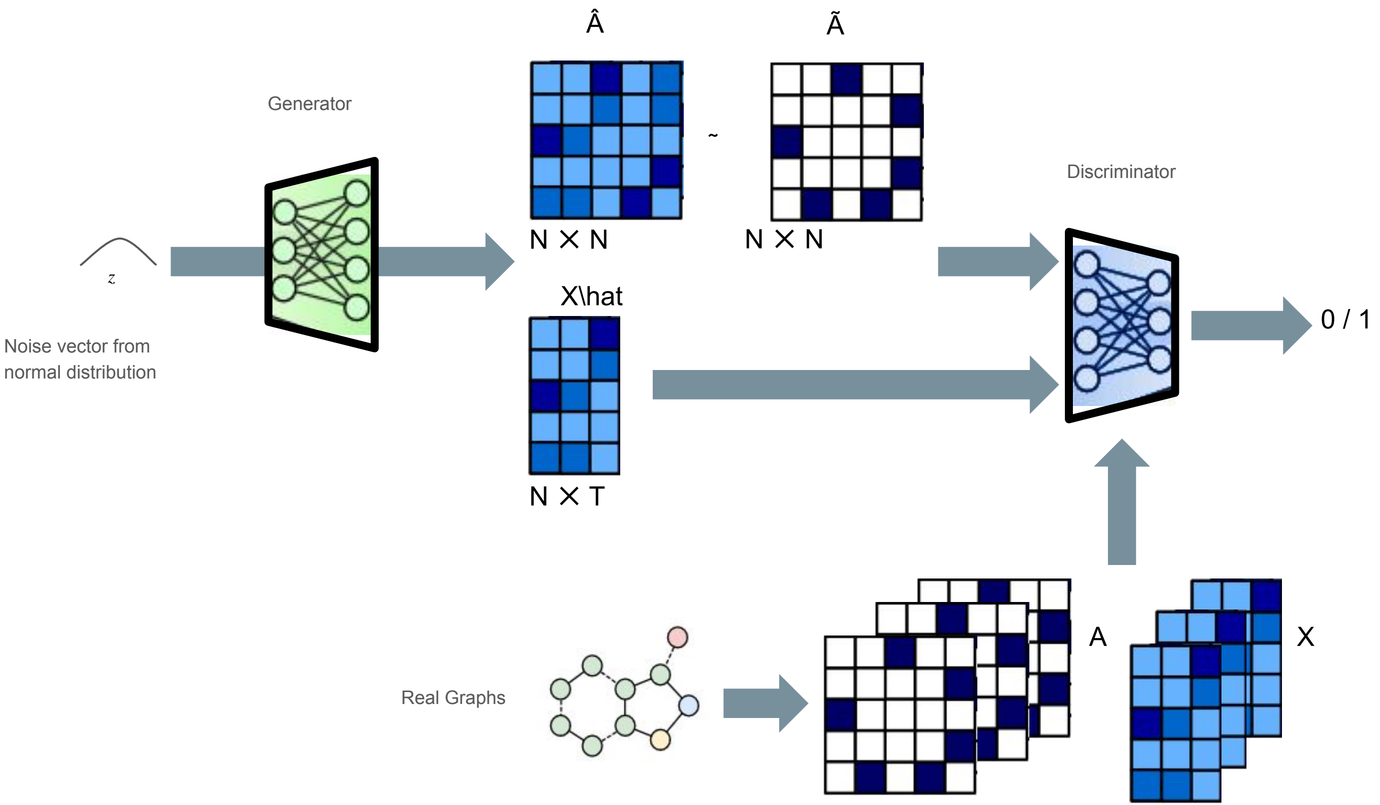
Generation Pipeline (plain graphs)





Continuous matrix not binarized in GLADC's pipeline. (may be skipped?)

Generation Pipeline (attributed graphs)



Pipeline (details)

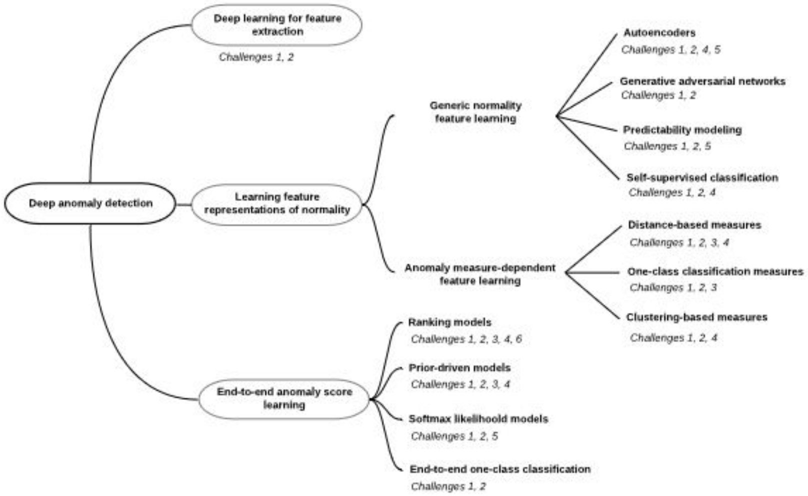
- Generator
  - Consists of a “multi dense” layer followed by one/two final linear layer(s).
    - Two if dealing with attributed graphs (one for the edges or A\hat and the other for the nodes X\hat).
    - One for plain graphs (edges or A\hat).
  - Dropout layers in between.
    - Prevents the model from becoming too dependent on particular neurons.
  - Loss: log(1 - D(G(z)))
- Discriminator
  - Possibly inspired from PatchGAN.
    - Classifies patches (local parts) of an image, or in this case a graph and its subgraphs.
  - Consists of graph convolution (for node representation), custom graph aggregation (for graph representation), “multi dense” layer and a final classification linear layer.
  - Fed both real and fake inputs for loss computation.
    - Loss obtained from sum of the losses for real and fake inputs.

Brainstorm

- As stated earlier, GLADC does not binarize the adjacency matrices before computing the loss.
  - Is this worth looking into?
  - MolGAN does suggest binarizing the adjacency matrices before feeding into discriminator.
  - Continuous adjacency matrices will simply tell the probability of an edge being present or not.
  - MolGAN applies categorical sampling.
    - This won’t work in our case since we don’t have differentiable categories of nodes or edges.
    - Will need to define a threshold for the probability of an edge being enough to consider it present.
- Try out discriminator as anomaly detection module?
- In fact, if we go for GAN generation, using the Discriminator as anomaly detection might be the only viable approach.
  - GLADC passes a graph as input to its auto encoder and returns a new graph from said input.
    - GANs only take noise.
  - GLADCs judges a graph’s anomaly potential by comparing the input with the generated fake graph.

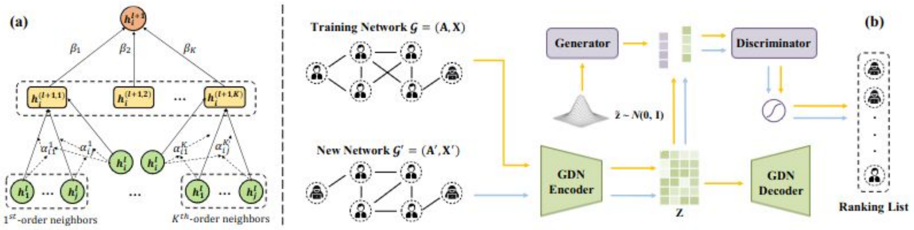
Brainstorm (cont.)

- Nice taxonomy of strategies for anomaly detection.
  - https://arxiv.org/abs/2007.02500



Brainstorm (cont.)

- Following previous taxonomy, GLADC lands in the “Generic normality feature learning”.
  - As it only uses normal graphs during learning.
  - In the Autoencoders sub-branch.
- Example of “Anomaly measure-dependent feature learning”
  - Inductive node level anomaly detection.
    - https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.ijcai.org/proceedings/2020/0179.pdf&ved=2ahUKEwjWvtO_gJqIAxWAg_0HHUt1CYEQFnoECBcQAQ&usg=AOvVaw3v-IfqYuDISW8ryHdNPHOa



Brainstorm (cont.)

- Going back three slides, are there other ways to utilize GANs' power?
- Latent space searching.
    - https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://kdd.org/exploration_files/p29-GAN_based_anomaly_detection_review_including_reviewer_suggestions.pdf&ved=2ahUKEwjX5NGSjJqIAxWbiv0HHbWtANEQFnoECBYQAQ&usg=AOvVaw05Yi7ISdjry2IvbgWnZ3FN
    - For any input data x, search your Generator's latent space Z for a z that closely resembles x.
        - If there is none, there is a chance you are looking at an anomalous graph.
    - This step can result particularly computationally expensive.
        - Some optimizations…
        - Usually used for image generation
            - Images are big matrices, we are dealing with small graph matrices A\hat and X\hat
    - Fast Ano-GAN.
        - https://pubmed.ncbi.nlm.nih.gov/30831356/
        - Additional network to optimize latent space searching.
        - Only applied for images?