Graph ConvNets for Molecule Generation and Travelling Salesman Problem

Xavier Bresson

School of Computer Science and Engineering
Data Science and AI Research Centre
Nanyang Technological University (NTU), Singapore

Joint work with T. Laurent (LMU) and C. Joshi (NTU)

IPAM Workshop
Deep Geometric Learning of Big Data and Applications
May 21st 2019

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

NATIONAL RESEARCH FOUNDATION
PRIME MINISTER'S OFFICE
SINGAPORE

Outline

- Graph ConvNets
- Molecule Generation
- Travelling Salesman Problem (TSP)
- Conclusion

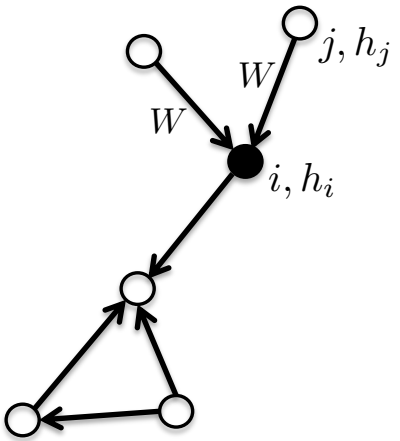
Outline

- Graph ConvNets
- Molecule Generation
- Travelling Salesman Problem (TSP)
- Conclusion

Graph Neural Networks[1]

- Spatial NN techniques to deal with arbitrary graphs.
- Minimal inner structures :
  - Invariant by vertex re-indexing (no graph matching is required)
  - Locality/local reception field (only neighbors are considered)
  - Weight sharing (convolutional operations)
  - Independence w.r.t. graph size

$$h_i = f_{\text{GNN}}\left(\{h_j : j \to i\}\right)$$



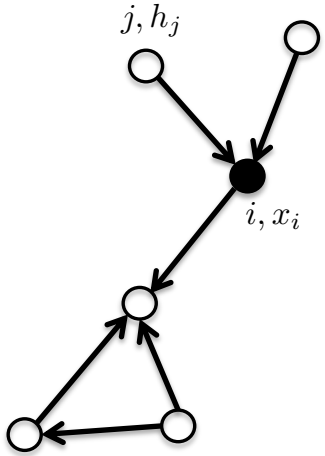
- What instantiation of f ?

[1] Scarselli, Gori, Tsoi, Hagenbuchner, Monfardini, The Graph Neural Network Model, 2009

Graph Recurrent Neural Networks

- Graph RNN with Multi-Layer Perceptron (MLP)[1] :

$$h_i = \sum_{j \to i} \mathcal{C}_{\text{G-MLP}}(x_i, h_j) = \sum_{j \to i} A\sigma(B\sigma(Ux_i + Vh_j))$$



- Graph GRU[2,3] (Gated Recurrent Unit) :

$$h_i = \mathcal{C}_{\text{G-GRU}}(x_i, \sum_{j \to i} h_j)$$

Fixed-point iterative scheme needed :

$$\begin{aligned}
\bar{h}_i^t &= \sum_{j \to i} h_j^t, \quad h_i^{t=0} = x_i \\
z_i^{t+1} &= \sigma(U_z h_i^t + V_z \bar{h}_i^t) \\
r_i^{t+1} &= \sigma(U_r h_i^t + V_r \bar{h}_i^t) \\
\tilde{h}_i^{t+1} &= \tanh\big(U_h(h_i^t \odot r_i^{t+1}) + V_h \bar{h}_i^t\big) \\
h_i^{t+1} &= (1 - z_i^{t+1}) \odot h_i^t + z_i^{t+1} \odot \tilde{h}_i^{t+1}
\end{aligned}$$

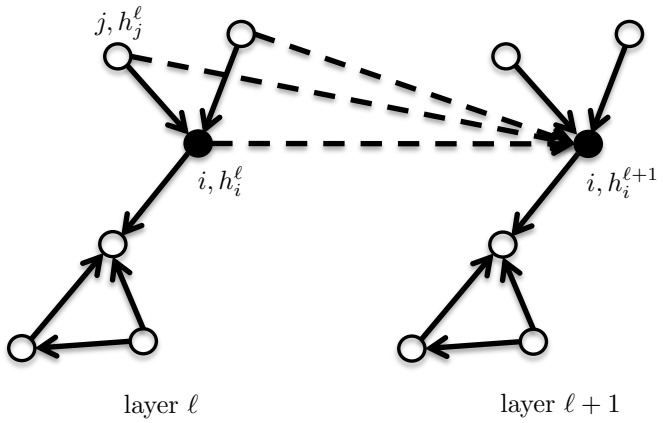
[1] Scarselli, Gori, Tsoi, Hagenbuchner, Monfardini, The Graph Neural Network Model, 2009
[2] Li, Tarlow, Brockschmidt, Zemel, Gated Graph Sequence Neural Networks, 2015
[3] Cho, Merrienboer, Gulcehre, Bahdanau, Bougares, Schwenk, Bengio, Learning Phrase Representations using RNN for Statistical Machine Translation, 2014

Graph ConvNets

- Vanilla graph ConvNets[1], GCN[2] (with ReLU) and GraphSAGE[3] (with max) :

$$h_i^{\ell+1} = \mathcal{C}_{\text{G-VCN}}\Big(h_i^\ell, \sum_{j \to i} h_j^\ell\Big), \quad h_i^{\ell=0} = x_i$$

$$= \text{ReLU}\Big(U^\ell h_i^\ell + V^\ell \sum_{j \to i} h_j^\ell\Big), \quad h_i^{\ell=0} = x_i$$



[1] Sukhbaatar, Szlam, Fergus, Learning Multiagent Communication with Backpropagation, 2016
[2] Kipf, Welling, Semi-Supervised Classification with Graph Convolutional Networks, 2017
[3] Hamilton, Ying, Leskovec, Inductive representation learning on large graphs, 2017

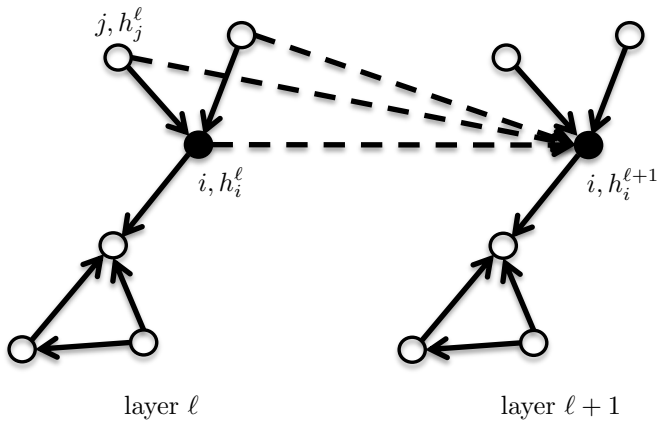
Residual Gated Graph ConvNets[1]

- Graph ConvNets architecture with edge gating mechanism leveraging[2-4], residuality[5] and batch normalization[6] :

$$h_i^{\ell+1} = h_i^\ell + \mathrm{ReLU}\Big(\mathrm{BN}\Big(U^\ell h_i^\ell + \sum_{j\to i} \eta_{ij}^\ell \odot V^\ell h_j^\ell\Big)\Big)$$

edge gates (anisotropic property)

$$\eta_{ij} = \sigma\left(A^\ell h_i^\ell + B^\ell h_j^\ell\right)$$



The idea is to design the simplest learnable anisotropic and multiscale diffusion operator on graphs [Perona-Malik'87 inspiration]

[1] Bresson, Laurent, Residual gated graph convnets, 2017
[2] Sukhbaatar, Szlam, Fergus, Learning Multiagent Communication with Backpropagation, 2016
[3] Hamilton, Ying, Leskovec, Inductive representation learning on large graphs, 2017
[4] Marcheggiani, Titov, Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling, 2017
[5] He, Zhang, Ren, Sun, Deep Residual Learning for Image Recognition, 2016
[6] Ioffe, Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015

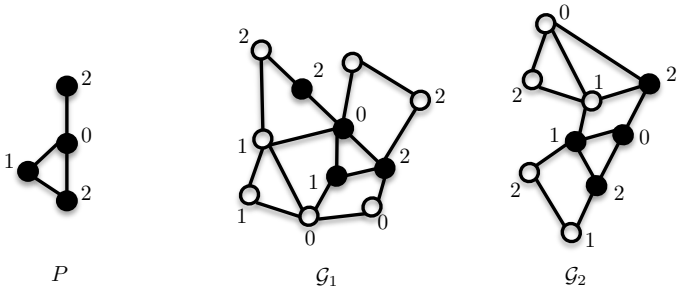
Graph RNNs vs Graph ConvNets

- Graph RNNs vs graph ConvNets
- Numerical study to compare both graph architectures[1] on two basic and representative graph problems:
  - Sub-graph matching[2]
  - Semi-supervised classification

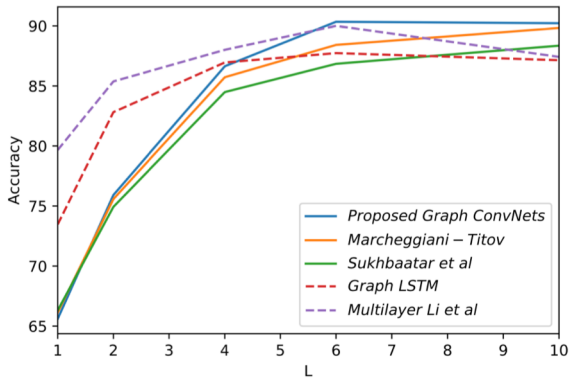
[1] Bresson, Laurent, Residual gated graph convnets, 2017
[2] Scarselli, Gori, Tsoi, Hagenbuchner, Monfardini, The Graph Neural Network Model, 2009

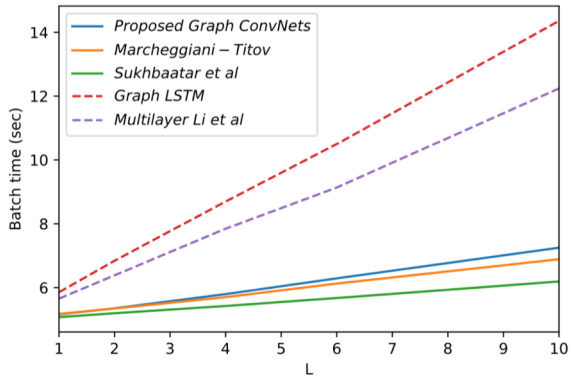
Numerical Experiments

- Graph learning problem :
  - Pattern matching



- Experimental results:

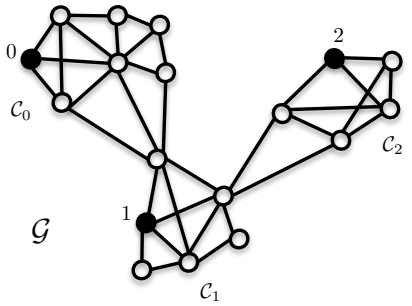




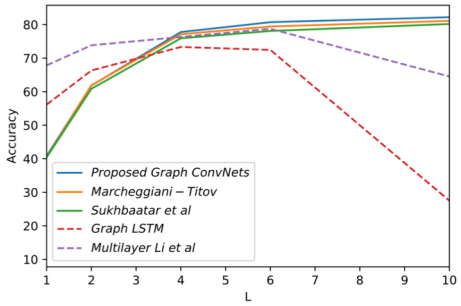
- All graph NNs are upgraded with residuality and batch normalization (offers 10% improvement).

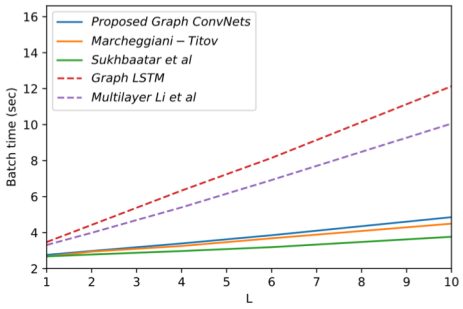
Numerical Experiments

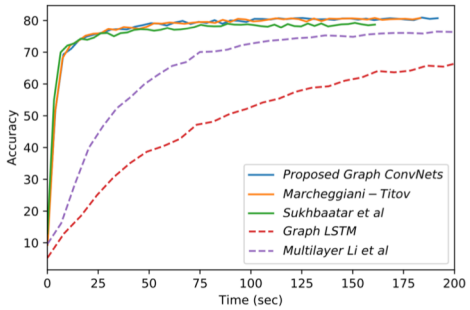
- Graph learning problem :
  - Semi-supervised clustering



- Experimental results :

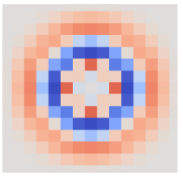






- ConvNets architectures that can be deep (by stacking many layers) offer competitive performances for graphs with variable sizes.

Anisotropy vs Isotropy



- Standard ConvNets produce anisotropic filters because Euclidean grids have directional structure.
- Graph ConvNets compute isotropic filters because there is no notion of directions on arbitrary graphs.
- How to get anisotropy back for graphs ?
  - Edge gates[1]/attention[2] information to treat neighbors differently.
  - Differentiate graph edges and graph vertices[3] (e.g. different atoms and atom connections)

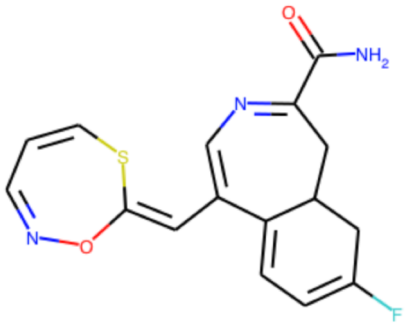
[1] Bresson, Laurent, Residual gated graph convnets, 2017
[2] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph Attention Networks, 2018
[3] Gilmer, Schoenholz, Riley, Vinyals, Dahl, Neural message passing for quantum chemistry, 2017

Outline

- Graph ConvNets
- **Molecule Generation**
- Travelling Salesman Problem (TSP)
- Conclusion

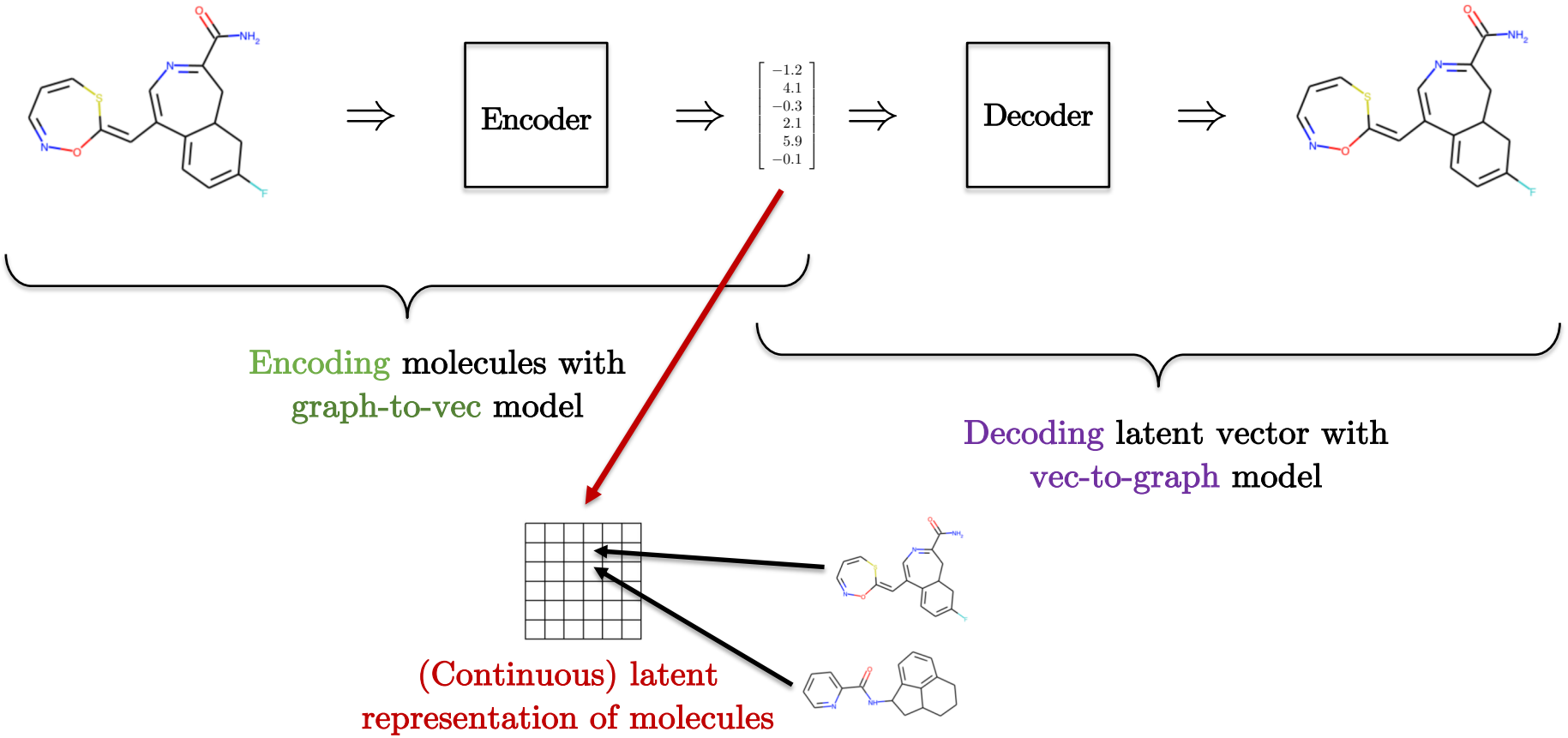
Molecule Generation

- Goal is to design a neural network that can
  - Auto-encode molecules,
  - Generate novel molecules,
  - Produce molecules with optimized chemical property.



Graph Auto-Encoder

- Graph-to-Graph Model :



Graph Encoder

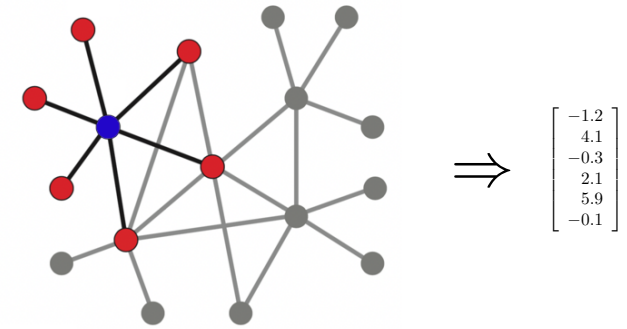
- Graph NNs often used to encode molecules into a continuous vectorial space.
  - GNNs used for regression s.a. Duvenaud-et-al[1], Gilmer-et.al[2] to predict molecular properties (1-2 orders of magnitude faster than solving Schrodinger equation w/ DFT).

Graph RNNs, Graph ConvNets, Graph Attention Nets

$$h_i = f_{\text{node}}(\{h_j\}, j \in \mathcal{N}(i))$$

$$z = g_{\text{graph}}(\{h_i\}, i \in V)$$

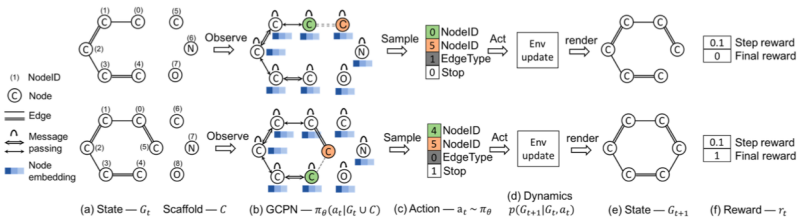
Reduce function : Sum or Mean



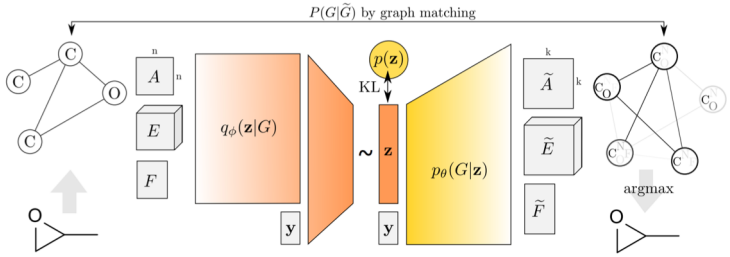
[1] Duvenaud, Maclaurin, Iparraguirre, Bombarell, Hirzel, Aspuru-Guzik, Adams, Convolutional networks on graphs for learning molecular fingerprints, 2015
[2] Gilmer, Schoenholz, Riley, Vinyals, Dahl, Neural message passing for quantum chemistry, 2017

Decoder & Graph Generation

- Encoding is easy. Decoding is more challenging !
- Two approaches :
  - Auto-regressive models : Sequential generation of molecules (atom-by-atom).
    - Jin-et.al, 2018[1], You-Leskovec-et.al, 2018[2], etc
  - One-shot models : Generation of all atoms in a single pass.
    - Simonovsky, Komodakis, 2018[3], De Cao, Kipf, 2018[4], etc



You-Leskovec-et.al, 2018

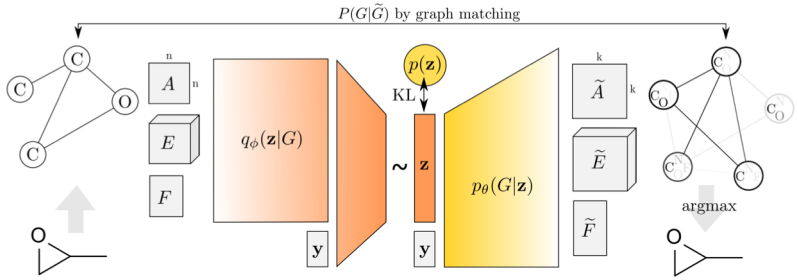


Simonovsky, Komodakis, 2018

[1] Jin, Barzilay, Jaakkola, Junction Tree Variational Autoencoder for Molecular Graph Generation, 2018
[2] You, Liu, Ying, Pande, Leskovec, Graph convolutional policy network for goal-directed molecular graph generation, 2018
[2] Simonovsky, Komodakis, GraphVAE: Towards generation of small graphs using variational autoencoders, 2018
[4] De Cao, Kipf, MolGAN: An implicit generative model for small molecular graphs, 2018

One-Shot Decoder

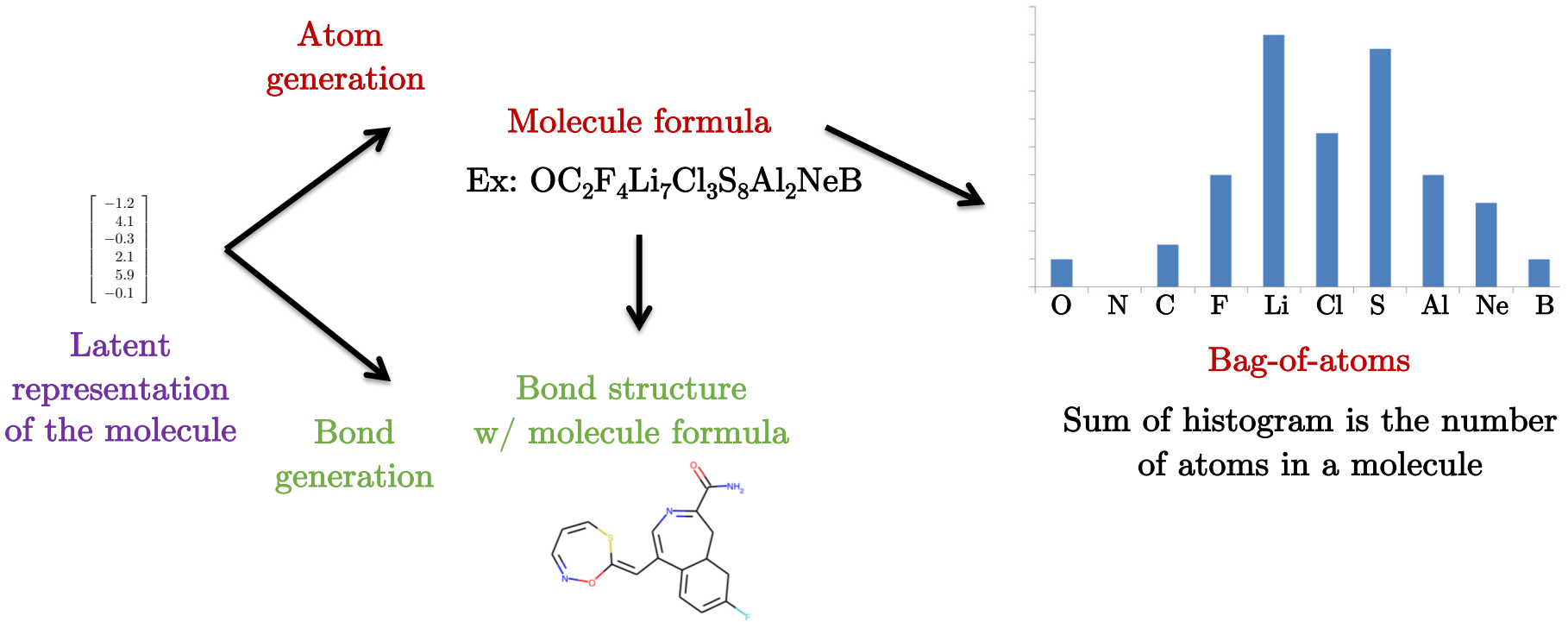
- A challenge with one-shot decoder is to generate molecules of different sizes.
  - It is hard to generate simultaneously :
    - The number of atoms,
    - The bond structure between the atoms.
  - Authors[1,2] generated molecules with a fixed size (the size of the largest molecule).



[1] Simonovsky, Komodakis, GraphVAE: Towards generation of small graphs using variational autoencoders, 2018
[2] De Cao, Kipf, MolGAN: An implicit generative model for small molecular graphs, 2018

Our Decoder

- We propose to disentangle these 2 problems :

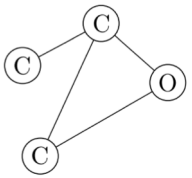


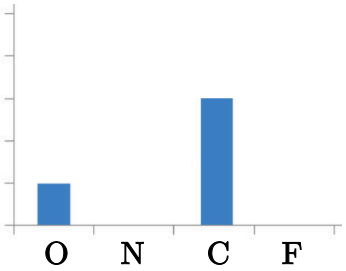
Atom Decoder

- We decode the latent representation of the molecule with a Multi-Layer Perceptron (MLP) to produce the histogram over the atoms in a molecule :

$$\begin{bmatrix} -1.2 \\ 4.1 \\ -0.3 \\ 2.1 \\ 5.9 \\ -0.1 \end{bmatrix}$$

MLP





Latent representation of the molecule

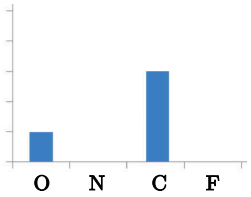
Bag-of-atoms

OC_3

Bond Decoder

- The "IKEA" model :
  - The bag-of-atoms indicates what atoms are in the molecule (IKEA pieces),
  - The atoms are assembled with a graph NN (IKEA assembly instructions).

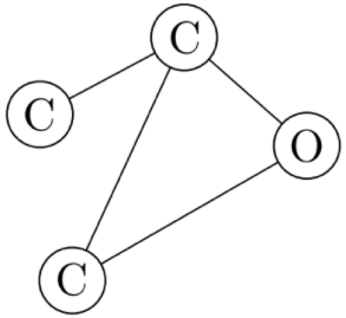
Bag-of-atoms
OC_3



Latent
representation
of the molecule

$$\begin{bmatrix} -1.2 \\ 4.1 \\ -0.3 \\ 2.1 \\ 5.9 \\ -0.1 \end{bmatrix}$$

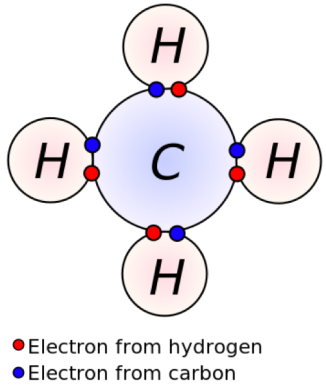
Graph
Net



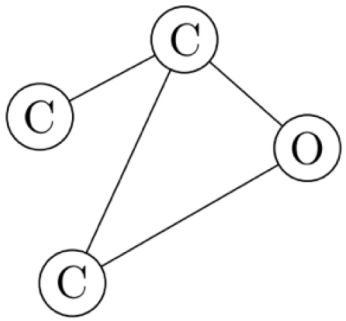
Decoded
molecule

Beam Search

- The one-shot model may not produce a chemically valid molecule.
  - Violation of atom valency (maximum number of electrons in the outer shell of the atom that can participate of a chemical bond).

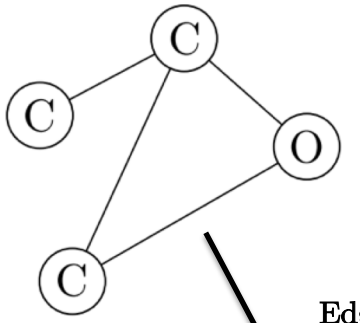


- We use beam search to produce a valid molecule.

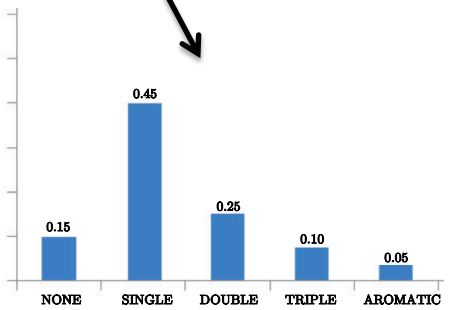


Beam Search

- Beam search :
  - Start with a random edge.
  - Select the next edge that
    - has the largest probability (or Bernouilli sampling),
    - is connected to selected edges,
    - does not violate valency.
  - Repeat for a number of different initializations.
  - Select the molecule that maximizes
    - The product of edge probabilities or,
    - The chemical property to be optimized s.a. druglikeness (QED), constrained solubility (logP), etc.

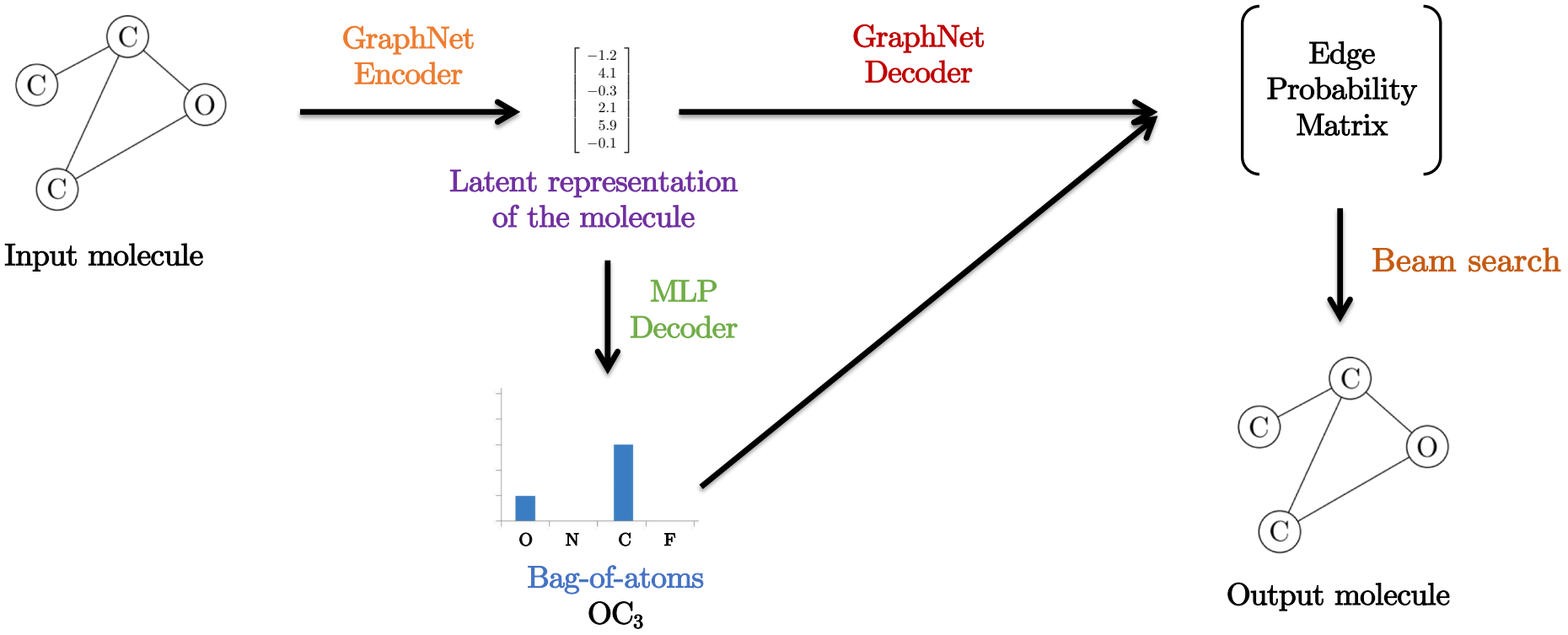


Edge probability map given by the output of the graph NN.



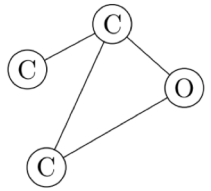
Summary

- Molecule auto-encoder system :



Encoder Description

- We use graph ConvNet[1] :



Input molecule

Graph ConvNet Encoder

$$\begin{bmatrix} -1.2 \\ 4.1 \\ -0.3 \\ 2.1 \\ 5.9 \\ -0.1 \end{bmatrix}$$

Latent representation z of the molecule

Node and edge representations

$$\begin{cases} h_i^{\ell+1} = h_i^\ell + \mathrm{ReLU}\Big(\mathrm{BN}\Big(W_1^\ell h_i^\ell + \sum_{j\sim i} \eta_{ij}^\ell \odot W_2^\ell h_j^\ell\Big)\Big) \\ e_{ij}^{\ell+1} = e_{ij}^\ell + \mathrm{ReLU}\Big(\mathrm{BN}\Big(V_1^\ell e_{ij}^\ell + V_2^\ell h_i^\ell + V_3^\ell h_j^\ell\Big)\Big) \end{cases} \quad \text{with} \quad \eta_{ij}^\ell = \frac{\sigma(e_{ij}^\ell)}{\sum_{j'\sim i}\sigma(e_{ij'}^\ell) + \varepsilon}$$

Dense attention

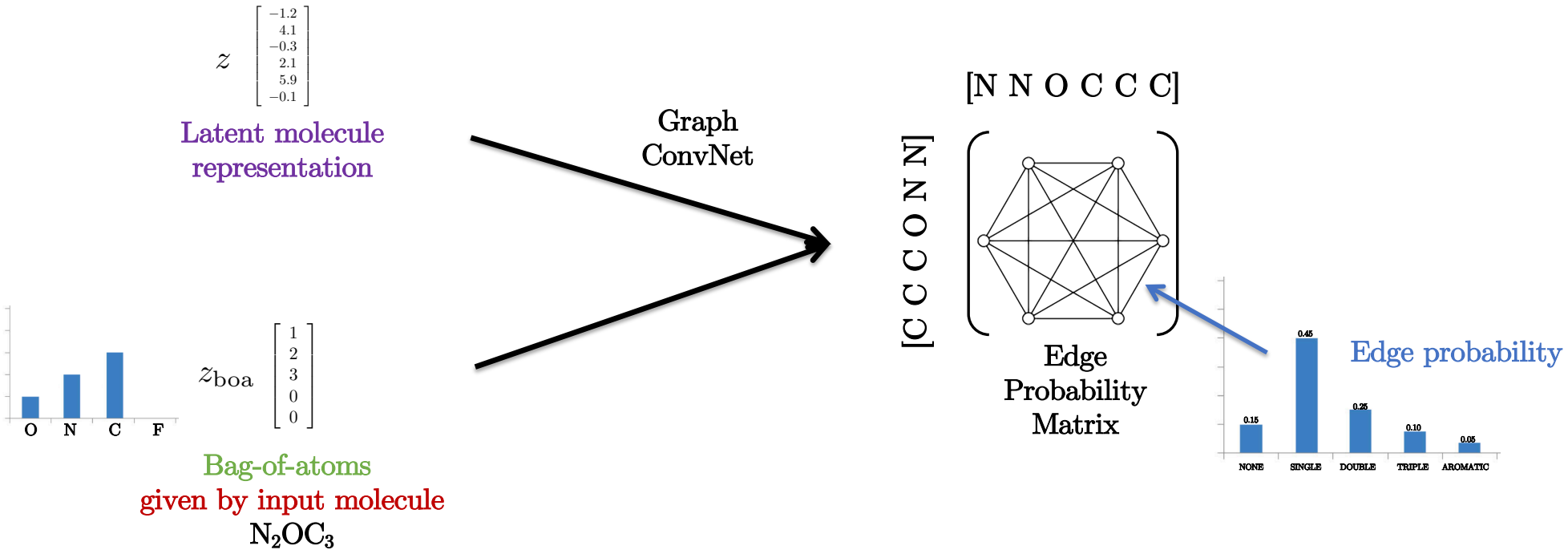
Graph representation

$$z = \sum_{i,j=1}^{N} \sigma\left(Ae_{ij}^L + Bh_i^L + Ch_j^L\right) \odot We_{ij}^L$$

[1] Bresson, Laurent, Residual gated graph convnets, 2017

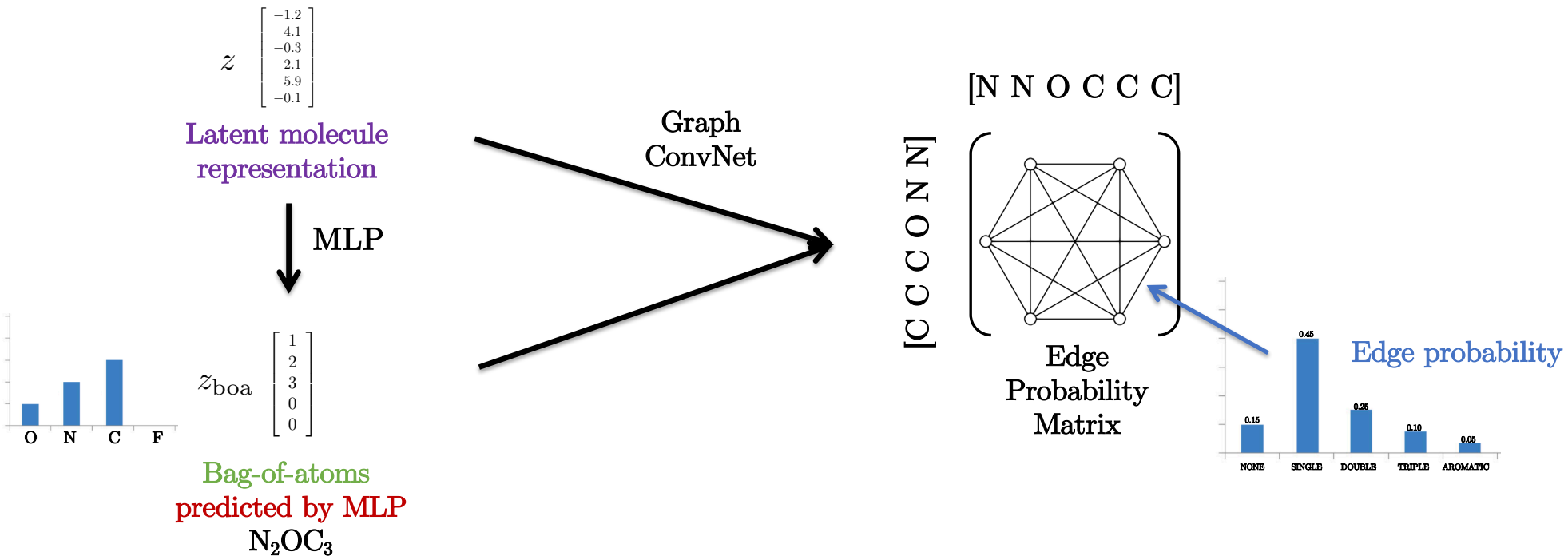
Bond Decoding during Training

- Given the latent encoding z of the molecule and the bag-of-atoms z_{boa} , we use a graph ConvNet to decode the bonds between the atoms :



Bond Decoding at Test Time

- The bag-of-atoms of the input molecule is predicted by a MLP :



Breaking symmetry

- The bond decoder starts with a fully connected graph with the atom type z_{ato} on each node.

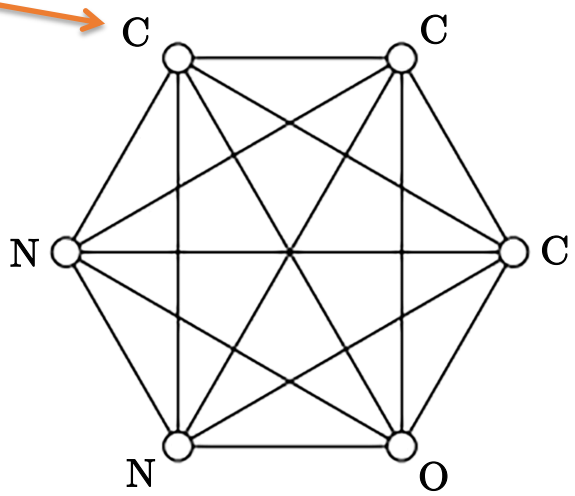
- This is not enough for the graph NN to be able to differentiate the 3 atoms of Carbon and the 2 atoms of Nitrogen !

  - We break the symmetry by introducing positional features z_{pos} , which will differentiate several atoms of the same type.

  - We concatenate this positional feature with the atom type z_{ato} to form the input node feature of the decoder.

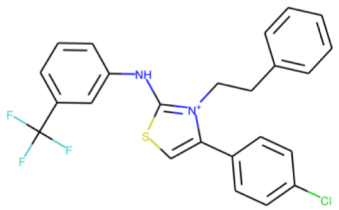
$$h_i^{\ell=0} = \begin{bmatrix} z_{\text{ato}\,i} \\ z_{\text{pos}_i} \end{bmatrix}$$

$$h_i^{\ell=0} = \begin{bmatrix} z_{\text{ato}\,i} \end{bmatrix}$$



Positional Features

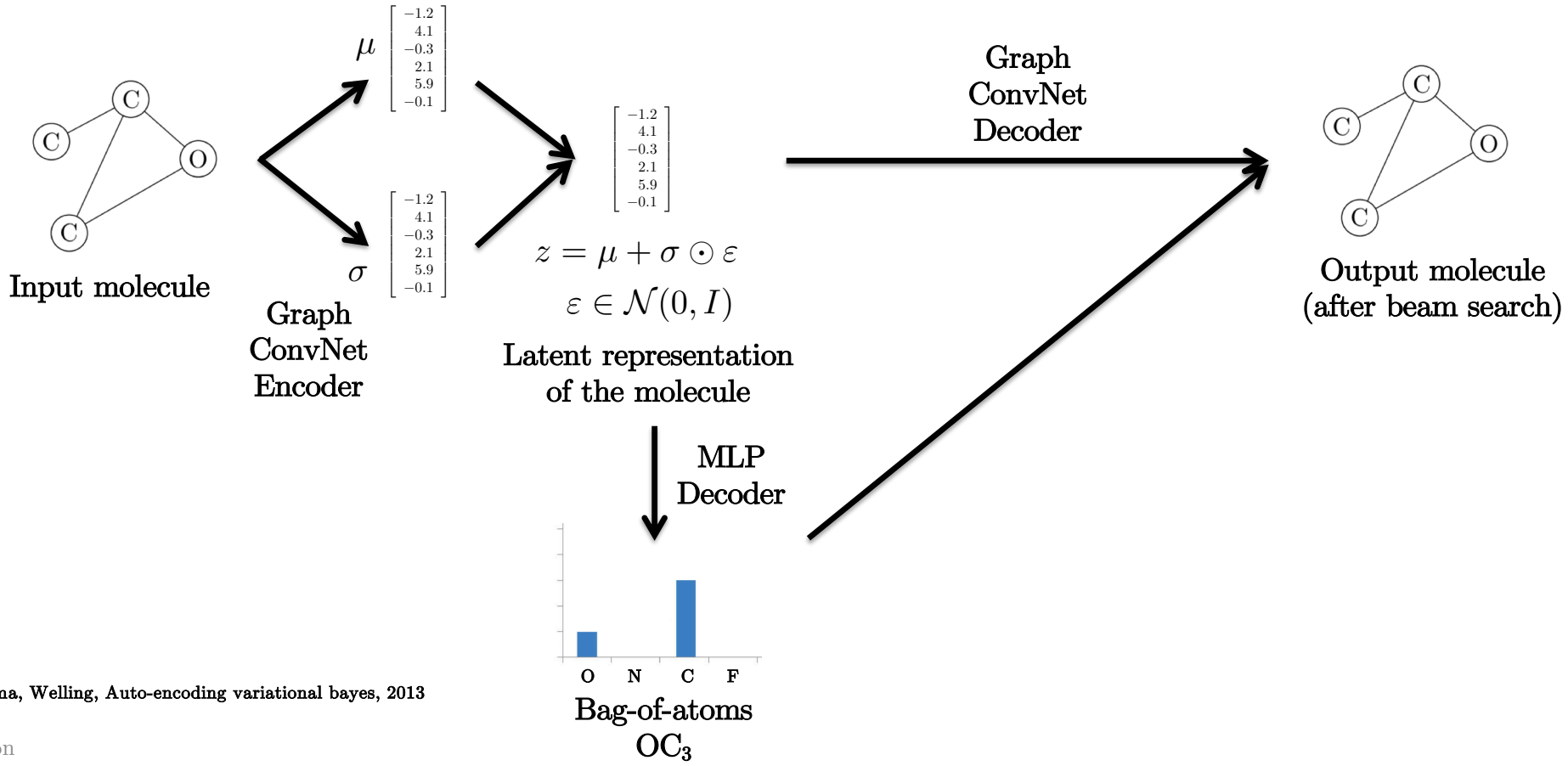
- We need to order the atoms.
- We use the SMILE representation of molecules to order the atoms.
  - A SMILE is a sequence of string characters that encodes atoms and bonds of a molecule.

SMILE () = FC(F)(F)C1=CC=CC(NC2=[N+](CCC3=CC=CC=C3)...
...C(C3=CC=C(Cl)C=C3)=CS2)=C1

$(F, 1)$ $(C, 1)$ $(F, 2)$ $(F, 3)$ $(C, 2) \ldots$

Variational Auto-Encoder

- Finally, we use a VAE formulation[1] to improve molecule generation "by filling the latent space" :



[1] Kingma, Welling, Auto-encoding variational bayes, 2013

Loss

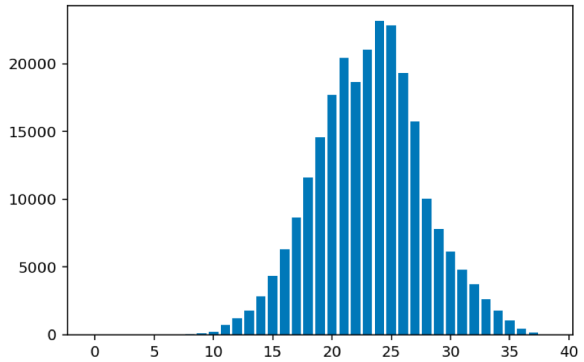
- Final loss is composed of
  - Cross-entropy loss for edge probability,
  - Cross-entropy loss for bag-of-atoms probability,
  - Kullback–Leibler divergence for the VAE Gaussian distribution.

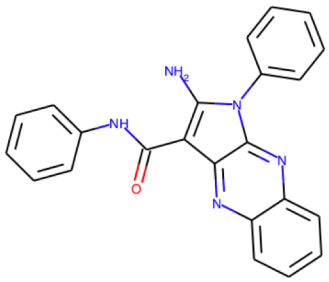
$$L = \lambda_e \sum_e \hat{p}_e \log p_e + \lambda_a \sum_a \hat{p}_a \log p_a - \frac{\lambda_{vae}}{2} \sum_k (1 + \log \sigma_k^2 - \mu_k^2 - \sigma_k^2)$$

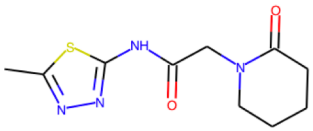
- No matching process necessary between input and output molecules because the same atom ordering is used (with the SMILE representation).

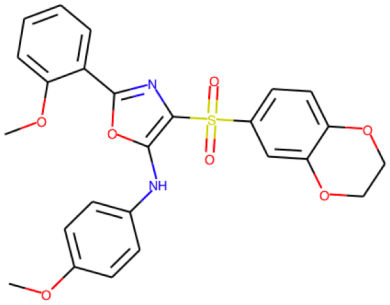
Dataset

- ZINC :
  - 250k drug like molecules,
  - Up to 38 heavy atoms (excluded Hydrogen).









Training

- Mini-batch of 50 molecules
- Learning rate is decreased by 1.25 after each epoch if training loss does not decrease by 1%.
- Learning stops when LR is less than 10^{-6}.
- Training takes 28 hours on a single Nvidia 1080Ti GPU.

Numerical Experiments

- Molecule reconstruction
  - How many molecules are correctly decoded?
- Molecule novelty
  - Beyond memorization – how many molecules sampled from the learned distribution are not in the training set?
- Molecule optimization
  - How much property improvement can we obtain when optimizing the latent space?
  - The chemical property is here the constrained solubility of molecules.

Main Baseline Techniques

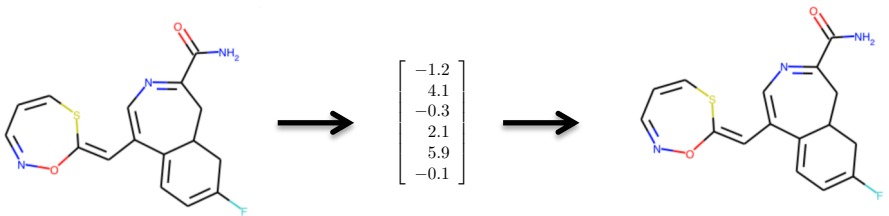
- VAE + SL :
  - JT-VAE : Jin, Barzilay, Jaakkola, Junction Tree Variational Autoencoder for Molecular Graph Generation, 2018

- GAN + RL :
  - GCPN : You, Liu, Ying, Pande, Leskovec, Graph convolutional policy network for goal-directed molecular graph generation, 2018

Molecule Reconstruction

Method	Reconstruction	Validity
CVAE [Gomez-Bombarelli et al., 2016]	44.6%	0.7%
GVAE [Kusner et al., 2017]	53.7%	7.2%
SD-VAE [Dai et al, 2018]	76.2%	43.5%
GraphVAE [Simonovsky, Komodakis, 2018]	-	13.5%
JT-VAE (SL) [Jin et al, 2018]	76.7%	**100.0%**
GCPN (GAN+RL) [You et al, 2018]	-	-
OURS (VAE+SL)	**90.5%**	**100.0%**

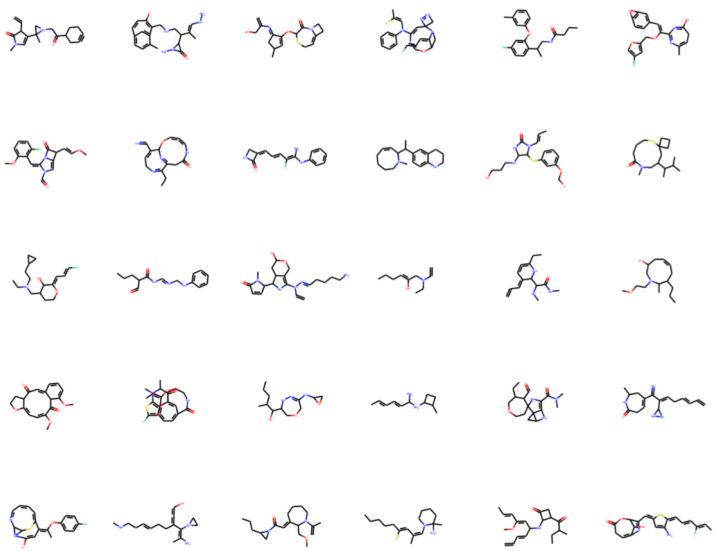
Table 1: Percentage of successful reconstruction of 250k ZINC molecules.



Molecule Novelty

Method	Novelty	Uniqueness
JT-VAE (SL) [Jin et al, 2018]	**100.0%**	**100.0%**
GCPN (GAN+RL) [You et al, 2018]	-	-
OURS (VAE+SL)	**100.0%**	**100.0%**

Table 2: Sample 5000 molecules from learned prior distribution.



Molecule Optimization #1

- Constrained optimization :
  - Goal is to maximize the constrained solubility of the training molecules.
  - Optimization is done by gradient ascent in the latent space of molecules.
  - Following JT-VAE, we report the top 3 optimized molecules :

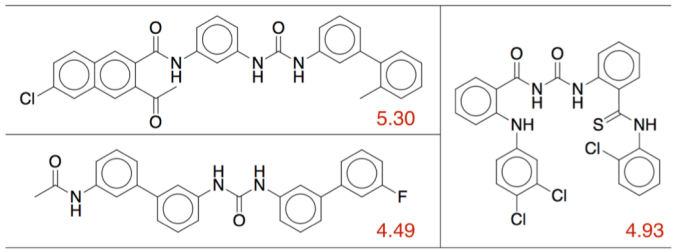
Method	1st	2nd	3rd
ZINC	4.52	4.30	4.23
CVAE [Gomez-Bombarelli et al., 2016]	1.98	1.42	1.19
GVAE [Kusner et al., 2017]	2.94	2.89	2.80
SD-VAE [Dai et al, 2018]	4.04	3.50	2.96
JT-VAE (SL) [Jin et al, 2018]	5.30	4.93	4.49
OURS (VAE+SL)	5.24	5.14	5.06
GCPN (GAN+RL) [You et al, 2018]	**7.98**	**7.85**	**7.80**

Table 3: Generative performance of the top three molecules for the constrained solubility.

Molecule Optimization #1

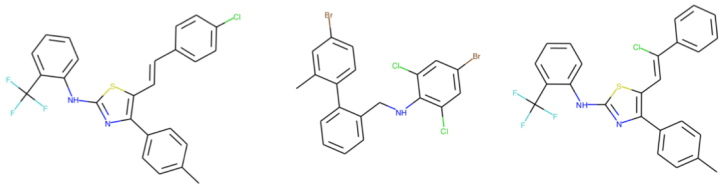
- Top 3 optimized molecules :

JT-VAE (VAE+SL)



Mean is 4.90

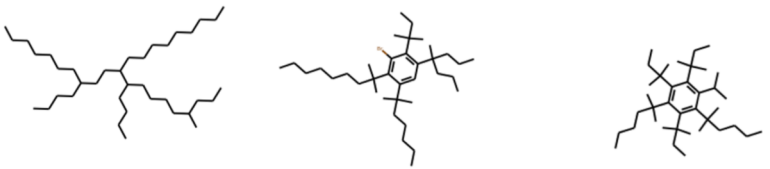
OURS (VAE+SL)



5.24 5.14 5.06

Mean is 5.14

GCPN (GAN+RL)



7.98 7.48 7.12

Mean is 7.52

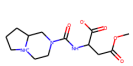
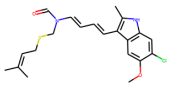
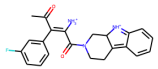
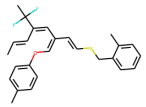
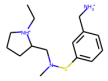
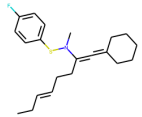
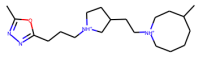
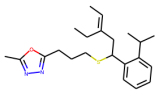
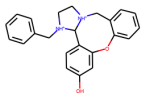
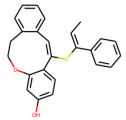
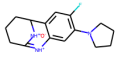
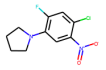
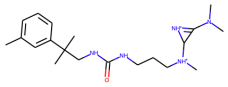
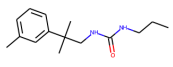
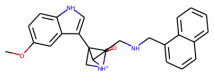
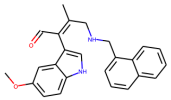
Molecule Optimization #2

- Constrained optimization :
  - Goal is to maximize the constrained solubility of the 800 test molecules with the lowest value.
  - The optimization of the chemical property is constrained by the similarity between the original molecule and the new generated molecule.
  - Following JT-VAE, we report property improvements w.r.t. molecule similarity δ :

	JT-VAE [Jin et al, 2018] (SL)			GCPN [You et al, 2018] (GAN+RL)			OURS (VAE+SL)		
δ	Improvement	Similarity	Success	Improvement	Similarity	Success	Improvement	Similarity	Success
0.0	1.91 ± 2.04	0.28 ± 0.15	97.5%	4.20 ± 1.28	**0.32 ± 0.12**	**100.0%**	**5.24 ± 1.55**	0.18 ± 0.12	**100.0%**
0.2	1.68 ± 1.85	0.33 ± 0.13	97.1%	4.12 ± 1.19	**0.34 ± 0.11**	**100.0%**	**4.29 ± 1.57**	0.31 ± 0.12	98.6%
0.4	0.84 ± 1.45	**0.51 ± 0.10**	83.6%	2.49 ± 1.30	0.47 ± 0.08	**100.0%**	**3.05 ± 1.46**	**0.51 ± 0.10**	84.0%
0.6	0.21 ± 0.71	**0.69 ± 0.06**	46.4%	0.79 ± 0.63	0.68 ± 0.08	**100.0%**	**2.46 ± 1.27**	0.67 ± 0.05	40.1%

Table 7: Molecule optimization results.

Molecule Optimization #2

Molecule similarity 0.0				
	4: -8.38	4: 2.19	77: -5.81	77: 4.75
Molecule similarity 0.2				
	143: -5.01	143: 3.54	136: -5.06	136: 3.10
Molecule similarity 0.4				
	604: -2.94	604: 3.39	103: -5.40	103: 0.88
Molecule similarity 0.6				
	89: -5.64	89: 0.94	782: -2.57	782: 2.44

Conclusion

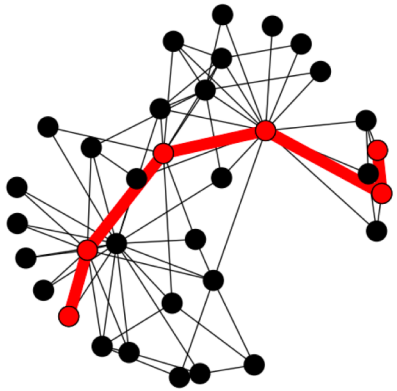
- We propose a simple and efficient VAE for atoms and bonds decoding.
- We report highest VAE accuracy on ZINC dataset for
  - Molecule reconstruction,
  - Molecule optimization of constrained solubility property.
- Comparing VAE+SL vs GAN+RL :
  - GAN+RL generates better molecules (outside the training statistics),
  - VAE+SL generates better optimized molecules similar to original ones,
  - GAN+RL generates optimized molecules with 100% success.

Outline

- Graph ConvNets
- Molecule Generation
- **Travelling Salesman Problem (TSP)**
- Conclusion

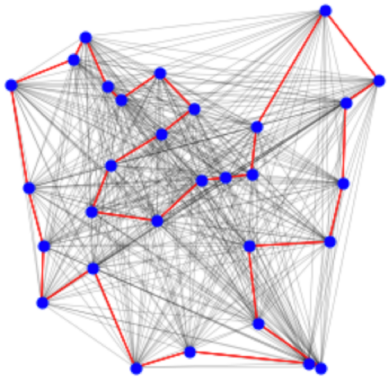
Graph combinatorial problems

- Shortest path problem :
  - Find a path between two vertices in a graph with shortest distance.
  - Dijkstra's algorithm : Exact solver in polynomial time $O(E+V\log V)$ with Fibonacci heap.
  - Applications : Road Navigator, P2P network, etc



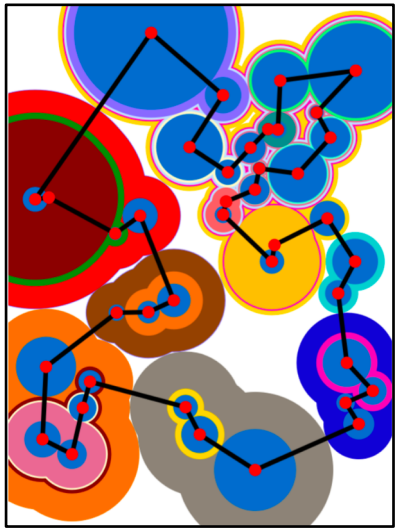
Graph combinatorial problems

- Travelling salesman problem (TSP) :
  - Find the shortest tour that visits each node and returns to the starting node.
  - NP-hard problem in combinatorial optimization.
    - Exact solution in factorial time $O(V!)$
  - Long history : Von Neumann'51, Dantzig'54, Bellman-Held-Karp'62, Edmonds'67, etc.
    - William Cook (Waterloo U.), MAA Invited Talk on TSP, Jan. 2018 : https://www.youtube.com/watch?v=q8nQTNvCrjE
  - Applications : Transportation, scheduling, hardware design, etc.



Operational Research

- State-of-the-art solution
  - Concorde solver developed by Applegate, Bixby, Chvatal, Cook, 2006 (test-of-time).
  - Code available :
    - http://www.math.uwaterloo.ca/tsp/concorde
  - Leveraged 30 years of theoretical developments + data structures and heuristics from computer science.
  - Branch-and-Cut approach :
    - Cutting plane algorithms (Dantzig et al., 1954) to iteratively solve linear relaxations of TSP.
    - Branch-and-bound approach to reduce solution search space.



Operational Research vs. Neural Networks

- Effective OR algorithms require :
  - Significant specialized knowledge,
  - Years of research work.
- "Can we use deep neural networks to learn better combinatorial optimization algorithms instead?" Bengio et al., 2018[1].
- The last three years, multiple efforts to develop a learning algorithm for TSP (proof-of-concept).
- Goal : Design learning algorithms for tackling previously un-encountered NP-hard problems, especially those that are non-trivial to design heuristics for, Bello et al., 2016[2].

[1] Bengio, Lodi, Prouvost, Machine learning for combinatorial optimization: a methodological tour d'horizon, 2018
[2] Bello, Pham, V Le, Norouzi, Bengio, Neural combinatorial optimization with reinforcement learning, 2016

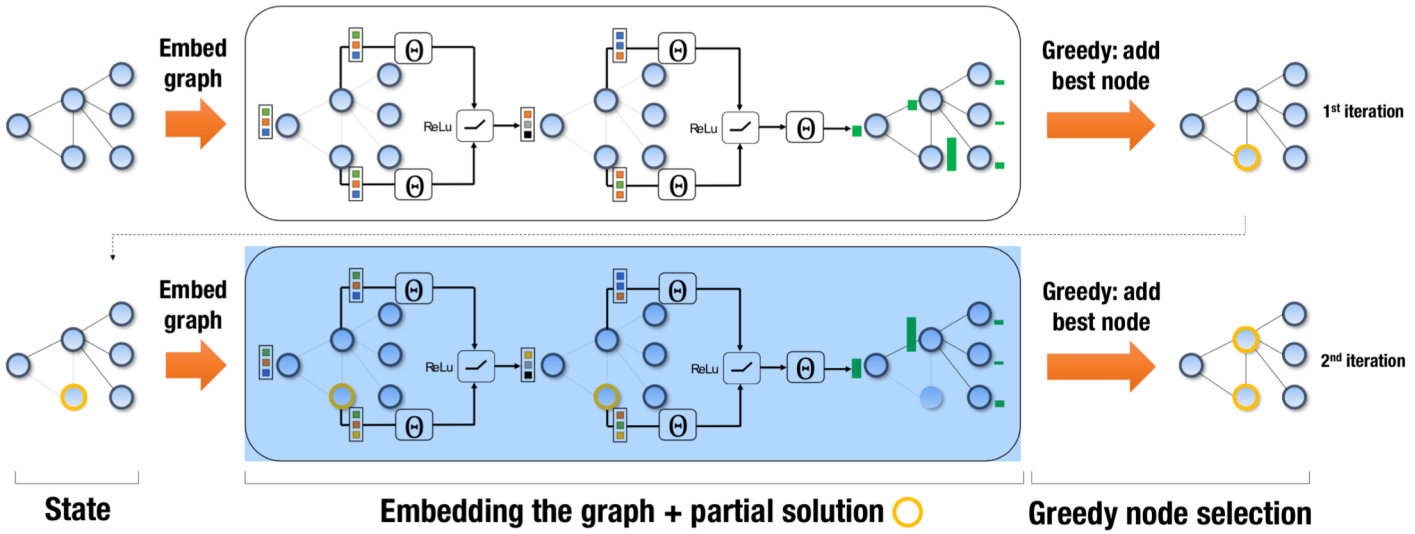
Neural Networks

- Sequence-to-sequence approach for TSP :
  - Introduced by Vinyals et al., 2015[1] and extended by Bello et al. 2016[2] :
    - Neural Network : Pointer Network (Seq2Seq model with attention mechanism), no use of graph structure.
    - Training Scheme : Reinforcement Learning to minimize the length of the tour.
    - Solution Search : Probabilistic sampling from learned policy.
    - Output Type : Autoregressive, step-by-step generation of the next node.

[1] Vinyals, Fortunato, Jaitly, Pointer networks, 2015
[2] Bello, Pham, V Le, Norouzi, Bengio, Neural combinatorial optimization with reinforcement learning, 2016

Neural Networks

- Graph neural network approach for TSP :
  - Introduced by Dai et al., 2017[1], and extended by Kool et al., 2019[2].



[1] Dai, Khalil, Zhang, Dilkina, Song, Learning combinatorial optimization algorithms over graphs, 2017
[2] Kool, van Hoof, Welling, Attention, learn to solve routing problems!, 2019

Neural Networks

- Kool et al., 2019[1] :
  - Neural Network : Graph Attention Network (GAT)[2]
  - Training Scheme : Reinforcement Learning to minimize the length of the tour.
  - Solution Search : Probabilistic sampling from learned policy.
  - Output Type : Autoregressive, step-by-step generation of the next node.
  - SOTA technique

[1] Kool, van Hoof, Welling, Attention, learn to solve routing problems!, 2019
[2] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph attention networks, 2017

Our Approach

- Introduced by Nowak et al., 2017[1] and extended by us.
  - Neural Network : Graph ConvNet[2]
  - Training Scheme : Supervised Learning using pairs of problems and solutions generated with Concorde.
  - Solution Search : Beam search over edge probability matrix.
  - Output Type : One-shot solution of nodes.

[1] Nowak, Villar, Bandeira, Bruna, A note on learning algorithms for quadratic assignment with graph neural networks, 2017
[2] Bresson, Laurent, Residual gated graph convnets, 2017

Baseline Techniques

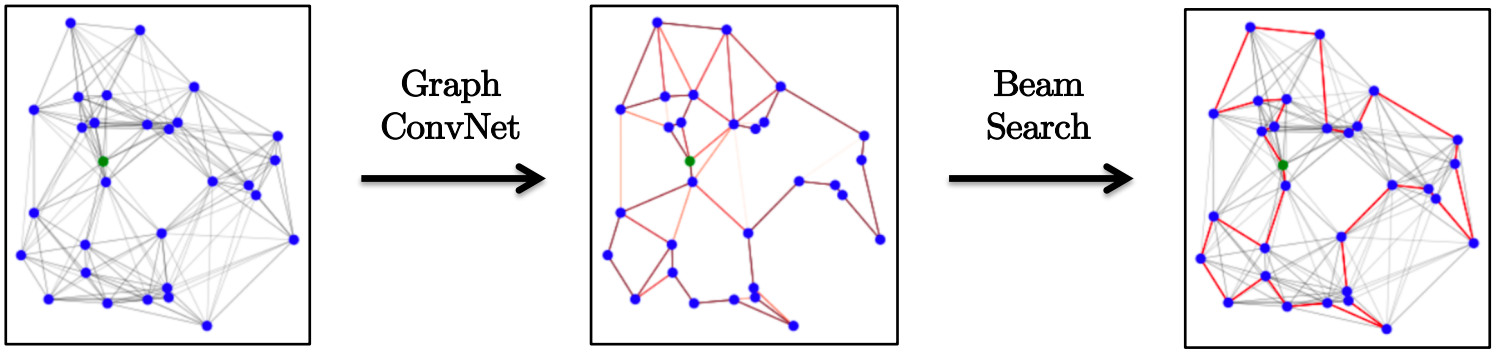
- OR techniques :
  - Concorde (TSP heuristics)
  - Gurobi (exact LP/QP solver)
  - Google OR Tools (general heuristics)

- Neural networks :

Method	Neural Network	Model Type	Training Setting	Solution Search Type
Bello et al. (2016)	Pointer Network	Autoregressive	RL	Sample from policy
Kool et al. (2019)	Graph Attention Network	Autoregressive	RL	Sample from policy
Ours	Graph ConvNet	Non-autoregressive	SL	Beam search

Model Overview

- Taking a 2D Euclidean graph as input,
- Graph ConvNet model outputs an edge probability matrix,
- Probability map is converted to a valid tour using beam search.



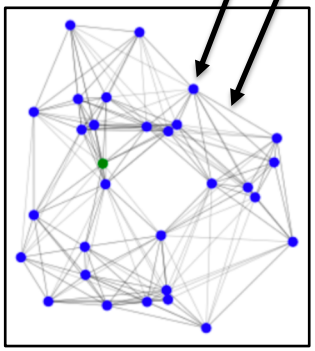
Graph ConvNet

Node and edge representations[1]

$$h_i^{\ell+1} = h_i^\ell + \mathrm{ReLU}\Big(\mathrm{BN}\Big(W_1^\ell h_i^\ell + \sum_{j\sim i} \eta_{ij}^\ell \odot W_2^\ell h_j^\ell\Big)\Big) \quad \text{with} \quad \eta_{ij}^\ell = \frac{\sigma(e_{ij}^\ell)}{\sum_{j'\sim i}\sigma(e_{ij'}^\ell) + \varepsilon}$$

Dense attention

$$e_{ij}^{\ell+1} = e_{ij}^\ell + \mathrm{ReLU}\Big(\mathrm{BN}\Big(V_1^\ell e_{ij}^\ell + V_2^\ell h_i^\ell + V_3^\ell h_j^\ell\Big)\Big)$$



$$h_i^{\ell=0} = x_i \in \mathbb{R}^2$$

$$e_{ij}^{\ell=0} = \|x_i - x_j\|_2$$

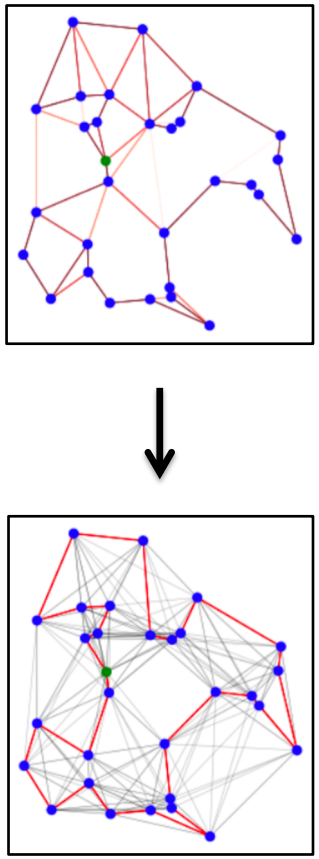
Edge classifier $\quad p_{ij} = \mathrm{sigmoid}(\mathrm{MLP}(e_{ij}^L))$

Loss is the logistic regression loss on edges.

[1] Bresson, Laurent, Residual gated graph convnets, 2017

Beam Search Decoder

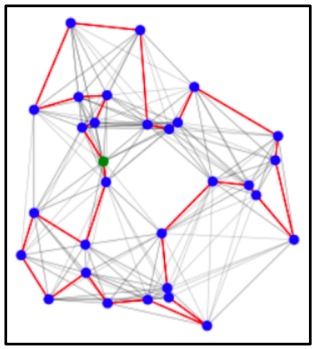
- Output of GNN :
  - Probabilistic heat-map over the adjacency matrix of tour connections.
- Beam search decoder :
  - Starting from a random node, expand b most probable edge connections among the node's neighbors.
  - Keep expanding the top-b partial tours at each stage until all nodes are visited.
  - Final prediction is the tour with the smallest length among the b tours at the end of search.



Dataset Generation

- Current paradigm :
  - Training and evaluating models on TSP instances of fixed sizes, that are 20, 50 and 100 nodes.

- Training sets :
  - 1 Million training pairs of problem instances and solutions,
  - 10K pairs for validation/test sets,
  - For each TSP, n node locations sampled randomly in the unit square. Optimal tour is found with Concorde.



Measuring Performance

- Metrics :
  - Predicted tour length : Average predicted tour length over 10,000 test instances.

$$l = \frac{1}{m} \sum_{i=1}^{m} l_m$$

  - Optimality gap : Average percentage ratio of predicted tour length relative to optimal solution over 10,000 test instances.

$$\text{gap} = \frac{1}{m} \sum_{i=1}^{m} \left(l_m / \hat{l}_m - 1 \right)$$

  - Evaluation time : Total wall clock time taken to solve 10,000 instances, either on single GPU (Nvidia 1080Ti) or 32 instances in parallel on a 32 virtual CPU system (2 × Xeon E5-2630).

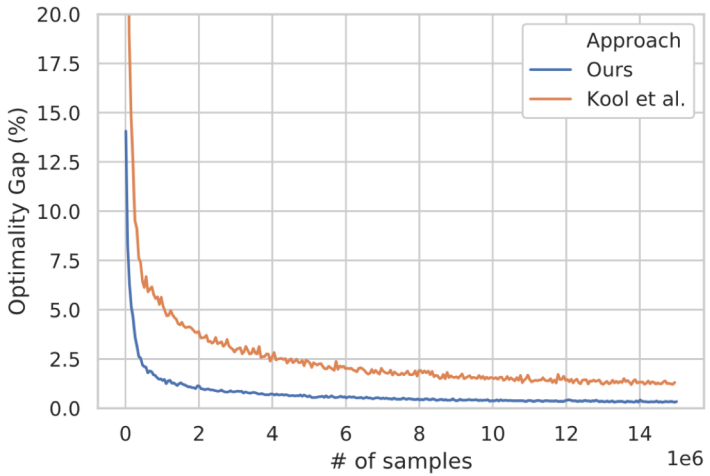
Performance on Fixed-size TSPs

- The proposed system
  - Outperforms previous NNs in terms of both optimality gap and evaluation time (graph ConvNet and beam search are parallelizable).
  - Does not outperform Concorde.

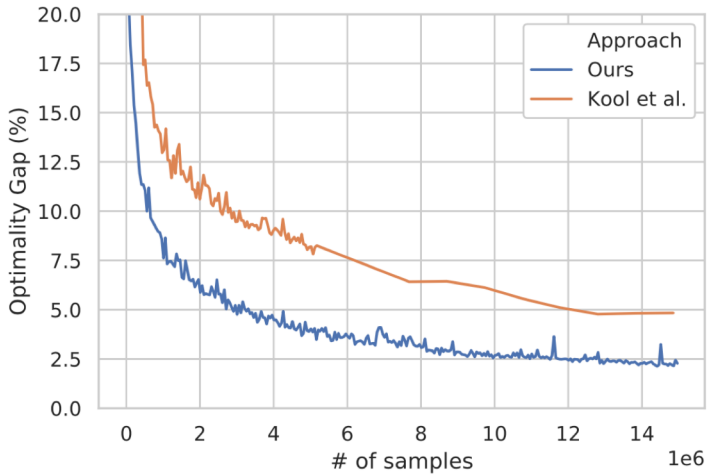
Method	Type	TSP20			TSP50			TSP100		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
Concorde	Solver	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
Gurobi	Solver	3.84	0.00%	(7s)	5.70	0.00%	(2m)	7.76	0.00%	(17m)
OR Tools	H, S	3.85	0.37%		5.80	1.83%		7.99	2.90%	
GNN [Nowak et al., 2017]	SL, BS	3.93	2.46%			-			-	
PtrNet [Bello et al., 2016]	RL, S		-		5.75	0.95%		8.00	3.03%	
GAT [Kool et al., 2019]	RL, S	3.84	0.08%	**(5m)**	5.73	0.52%	(24m)	7.94	2.26%	(1h)
GCN (Ours)	**SL, BS***	**3.84**	**0.01%**	(12m)	**5.70**	**0.01%**	**(18m)**	**7.87**	**1.39%**	**(40m)**

Performance on Fixed-size TSPs

- Sample efficiency :



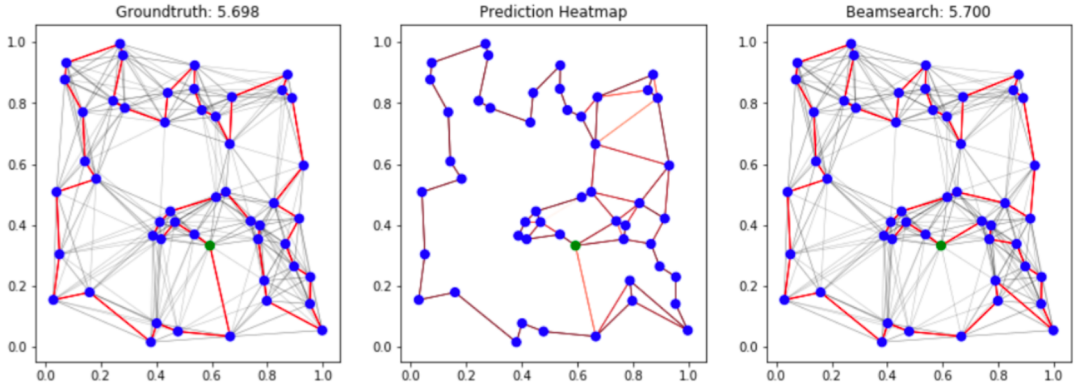
TSP50



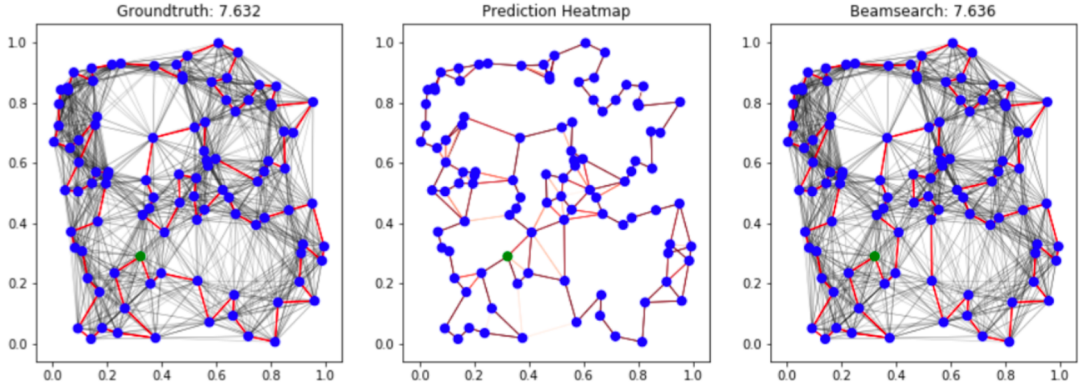
TSP100

Visualization

TSP50



TSP100



Trade-offs

- Supervised learning :
  - Better learning : The model is given more information to learn faster.
  - Limited data : Generating labelled datasets for instances beyond hundreds of nodes is computationally expensive.
- Reinforcement Learning :
  - Sparse learning : The model is only given a reward ⇒ Less informative for learning.
  - Infinite data : Does not require the creation of labelled datasets.

Transfer to Different TSP Sizes

- Zero-shot generalization to different problem sizes :

Problem	TSP20 Model	TSP50 Model	TSP100 Model
TSP20	0.08%	2%	22%
TSP50	1%	0.52%	3%
TSP100	4%	2.5%	2.26%

Optimality gap for Kool et al., 2019

Problem	TSP20 Model	TSP50 Model	TSP100 Model
TSP20	0.01%	34%	70%
TSP50	38%	0.01%	65%
TSP100	28%	31%	1.39%

Optimality gap for our model

Conclusion

- We propose a simple and fast NN technique for solving TSPs for fixed sizes.
- Future work :
  - Generalization to different TSP sizes with zero-shot learning or fine-tuning.
  - Being faster than Concorde/OR solvers due to parallelization (for large TSP sizes).
  - Beyond TSP : Expand our system to RL to solve combinatorial problems that are non-trivial to design heuristics[1,2].

[1] Bengio, Lodi, Prouvost, Machine learning for combinatorial optimization: a methodological tour d'horizon, 2018
[2] Bello, Pham, V Le, Norouzi, Bengio, Neural combinatorial optimization with reinforcement learning, 2016

Outline

- Graph ConvNets
- Molecule Generation
- Travelling Salesman Problem (TSP)
- **Conclusion**

Conclusion

- We have solved two different problems, molecule generation and TSP, with the same technique :
  - One-shot graph ConvNet + Supervised Learning + Beam Search
  - Simple and fast solution (GPU parallelizable)
- An alternative to auto-regressive graph NN methods.

Questions?