CS6208 : Advanced Topics in Artificial Intelligence
Graph Machine Learning

Lecture 7 : Graph Transformer & Graph ViT/MLP-Mixer

Semester 2 2022/23

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science
National University of Singapore (NUS)

Outline

- Review of graph network architectures
- Graph positional encoding
- Transformers for sequences
- Graph transformers
- ViT/MLP-Mixer for images
- Graph ViT/MLP-Mixer
- Conclusion

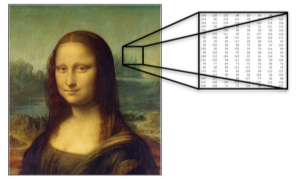
Outline

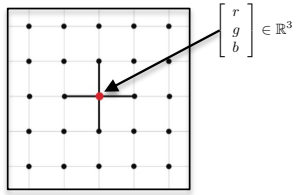
- Review of graph network architectures
- Graph positional encoding
- Transformers for sequences
- Graph transformers
- ViT/MLP-Mixer for images
- Graph ViT/MLP-Mixer
- Conclusion

Neural network architectures

- To enable DL+X, we need three key ingredients :
  - Data, GPU and neural network

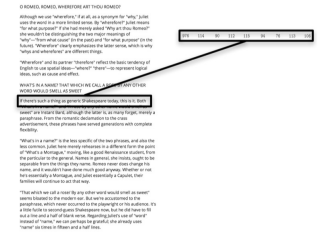
- There are four classes of networks
  - CNNs/RNNs/Transformers are designed for grids, sequences and sets.
    - GPUs/DL libraries PyTorch[1], TensorFlow[2] are optimized for these architectures.
  - Graph Neural Networks (GNNs) are more universal and broadly applicable architectures.
    - GPUs are not optimized for sparse linear algebra.

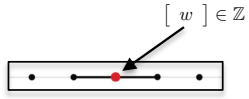




grid

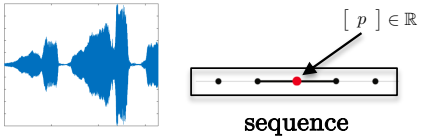
Computer Vision (CV)





sequence

Natural Language Processing (NLP)



sequence

Speech Recognition (SR)

Graph Analysis

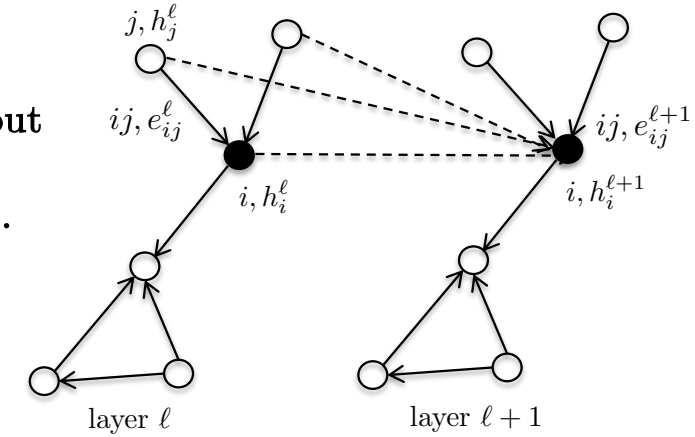
[1] Paszke et-al, Automatic differentiation in pytorch, 2017
[2] Abadi et-al, TensorFlow: a system for Large-Scale machine learning, 2016

GNN architectures

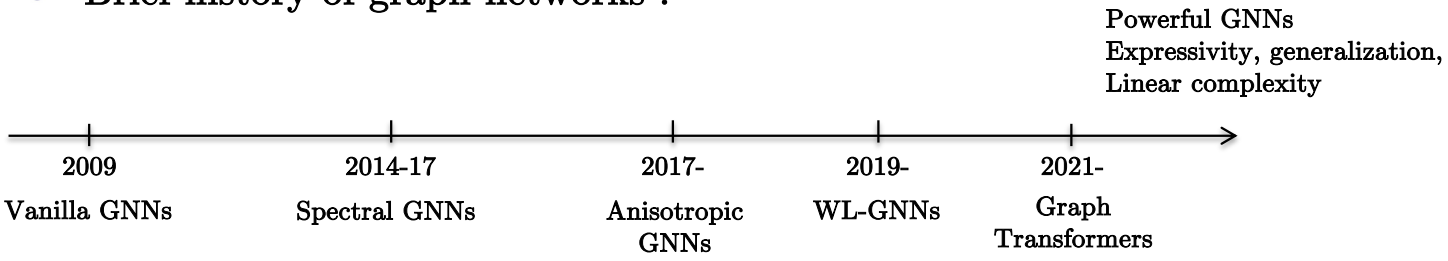
- What is a graph network?
  - Compositional parametric equivariant function that maps an input pair (graph, feature) to continuous vectorial representation for node/edge/graph which can be used for downstream graph tasks.
  - Often defined as a message-passing process[1] :

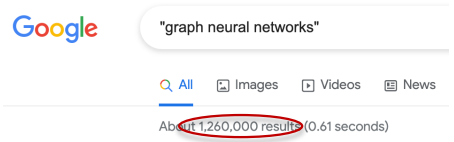
$$\text{Node-update}: \quad h_i^{\ell+1} = f_{\text{node}}\left(h_i^{\ell}, \{h_j^{\ell}, e_{ij}^{\ell} : j \in \mathcal{N}_i\}\right) \in \mathbb{R}^d$$

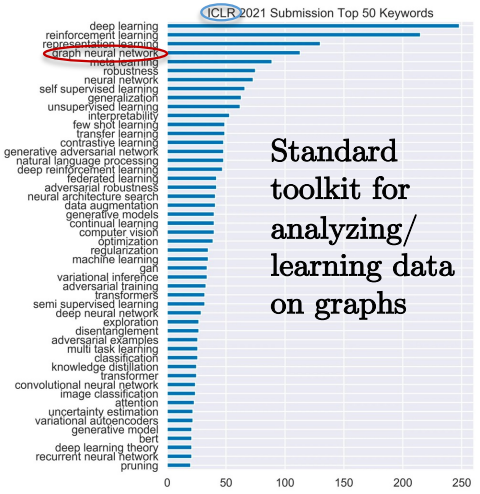
$$\text{Edge-update}: \quad e_{ij}^{\ell+1} = f_{\text{edge}}\left(e_{ij}^{\ell}, h_i^{\ell}, h_j^{\ell}\right) \in \mathbb{R}^d$$



- Brief history of graph networks :







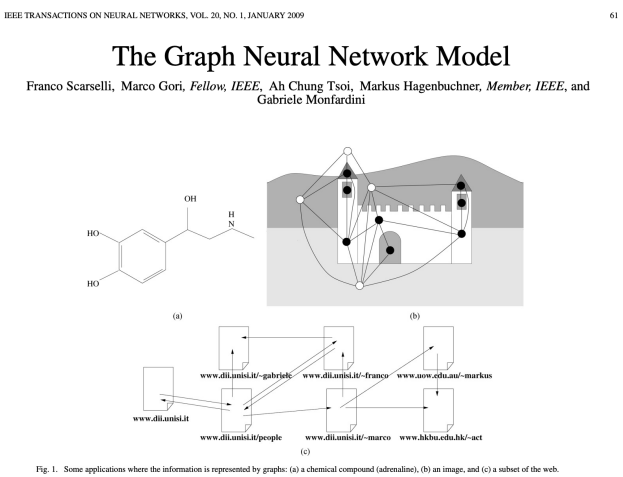
Standard toolkit for analyzing/ learning data on graphs

[1] Gilmer, Schoenholz, Riley, Vinyals, Dahl, Neural message passing for quantum chemistry, 2017

Vanilla GNNs[1]

- Contributions
  - First NN with layers equivariant/invariant w.r.t. index permutation, independent to neighborhood and graph size, local reception field, weight sharing.
  - Linear complexity O(N+E)
- Limitations
  - Simple model (vanilla RNN w/ aggregation of neighbors)
  - Vanishing gradient problem

$$
\begin{aligned}
h_i^{\ell+1} &= f_{\text{VGNN}}\big(h_i^{\ell}, \{h_j^{\ell} : j \in \mathcal{N}_i\}\big) \\
&= \sum_{j \to i} \sigma(U x_i + V h_j^{\ell}) \in \mathbb{R}^d
\end{aligned}
$$

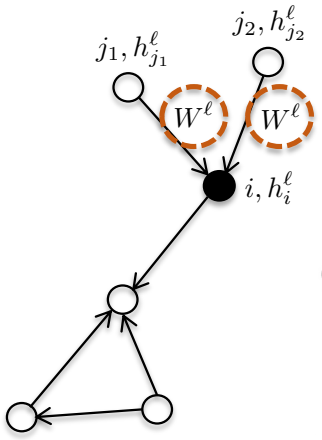


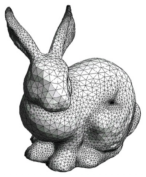
[1] Scarselli, Gori, Tsoi, Hagenbuchner, Monfardini, The Graph Neural Network Model, 2009

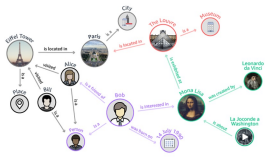
Spectral GNNs[1,2,3]

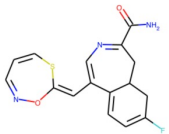
- Contributions
  - Leverage graph spectral theory to define convolution on graphs
  - Stable, robust to perturbation, exact k-hop kernel support
  - Linear complexity O(N+E)
  - GCN[3] is the most popular GNN technique
- Limitation
  - Isotropic kernels

$$
\begin{aligned}
h^{\ell+1} &= \sigma(\ s^\ell *_{\mathcal{G}} h^\ell\ ) \in \mathbb{R}^{N\times d} \\
&= \sigma(\ U\hat{s}^\ell(\Lambda)U^T h^\ell\ ) \\
&= \sigma(\ \hat{s}^\ell(\Delta)h^\ell\ ),\ \Delta = U\Lambda U^T \\
&= \sigma(\ \sum_{k=0}^{K-1} \underbrace{T_k(\Delta)}_{\substack{\text{Chebyshev}\\ \text{functions}}} h^\ell W_k^\ell\ ) \\
&= \sigma(\ D^{-1/2}AD^{-1/2}h^\ell W^\ell\ ) \\
&\text{1st order approx} \\
h_i^{\ell+1} &= \sigma(\ \frac{1}{\sqrt{d_i}}\sum_{j\in\mathcal{N}_i}\frac{1}{\sqrt{d_j}}A_{ij}W^\ell h_j^\ell\ ) \in \mathbb{R}^d
\end{aligned}
$$





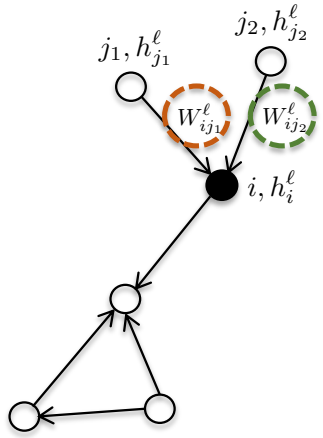




Most data is anisotropic

[1] Bruna, Zaremba, Szlam, LeCun, Spectral networks and locally connected networks on graphs, 2013
[2] Defferrard, Bresson, Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, 2016
[3] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017

Anisotropic GNNs



- Contributions
  - Design anisotropic equivariant mechanisms that treat neighbors differently.
  - GatedGCNs[1] with anisotropic diffusion[2], GAT[3] with softmax attention[4], Mesh CNNs[5] with anisotropic gauge equivariant kernels
  - Stable, robust, interpretable, linear complexity O(N+E)

$$h_i^{\ell+1} = h_i^\ell + \mathrm{ReLU}\Big(\mathrm{BN}\Big(W_1^\ell h_i^\ell + \sum_{j\in\mathcal{N}_i} e_{ij}^\ell \odot W_2^\ell h_j^\ell\Big)\Big)$$

$$e_{ij}^\ell = \frac{\sigma(\hat{e}_{ij}^\ell)}{\sum_{j'\in\mathcal{N}_i}\sigma(\hat{e}_{ij'}^\ell)+\varepsilon}$$

$$\hat{e}_{ij}^\ell = \hat{e}_{ij}^{\ell-1} + \mathrm{ReLU}\Big(\mathrm{BN}\big(V_1^\ell h_i^{\ell-1} + V_2^\ell h_j^{\ell-1} + V_3^\ell \hat{e}_{ij}^{\ell-1}\big)\Big)$$

GatedGCNs[1]

(based on Perona-Malik's anisotropic PDE[2] generalized on graphs)

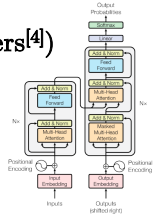
$$h_i^{\ell+1} = \mathrm{Concat}_{k=1}^{K}\Big(\mathrm{ELU}\Big(\sum_{j\in\mathcal{N}_i} e_{ij}^{k,\ell}\ W_1^{k,\ell}\ h_j^\ell\Big)\Big)$$

$$e_{ij}^{k,\ell} = \mathrm{Softmax}_{\mathcal{N}_i}(\hat{e}_{ij}^{k,\ell}) = \frac{\exp(\hat{e}_{ij}^{k,\ell})}{\sum_{j'\in\mathcal{N}_i}\exp(\hat{e}_{ij'}^{k,\ell})}$$

$$\hat{e}_{ij}^{k,\ell} = \mathrm{LeakyReLU}\Big(W_2^{k,\ell}\ \mathrm{Concat}\big(W_1^{k,\ell}h_i^\ell,\ W_1^{k,\ell}h_j^\ell\big)\Big)$$

GAT[3]

(based on Transformers[4])



- Limitations
  - Low expressivity, over-squashing issue

[1] Bresson, Laurent, Residual gated graph convnets, 2017
[2] Perona, Malik, Scale-space and edge detection using anisotropic diffusion, 1987
[3] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph Attention Networks, 2017
[4] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017
[5] De Haan, Weiler, Cohen, Welling, Gauge equivariant mesh CNNs: Anisotropic convolutions on geometric graphs, 2020

Low expressivity

- Most GNNs s.a. GCN[1], GAT[2], GatedGCNs[3] have theoretically low expressivity/ representation power to
  - Distinguish (simple) non-isomorphic graphs[4,5].
  - Identify/count elementary sub-structures like cycles and cliques[6].

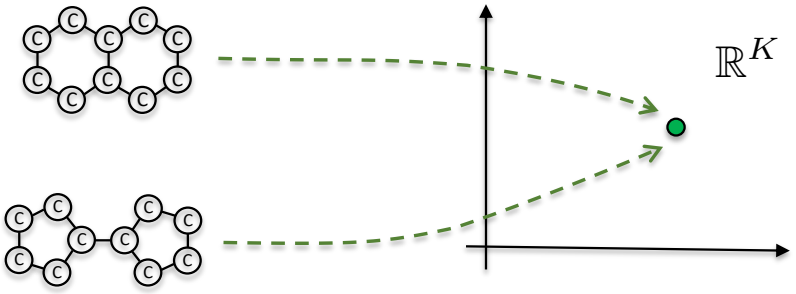


Fig 1. Vectorial graph representation

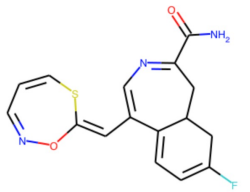


Fig 2. Graph with cycles (rings)

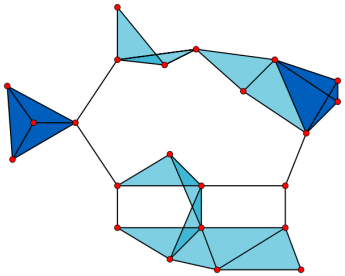


Fig 3. Graph with
19 × 3-vertex cliques (light and dark blue triangles)
and 2 × 4-vertex cliques (dark blue areas)

[1] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[2] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph Attention Networks, 2018
[3] Bresson, Laurent, Residual gated graph convnets, 2017
[4] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks? 2019
[5] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph networks, 2019
[6] Chen, Chen, Villar, Bruna, Can graph neural networks count substructures? 2020

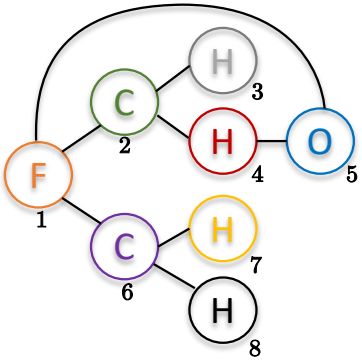
Graph isomorphism

- Graph isomorphism : Two graphs are isomorphic if there exists an index permutation that preserves node neighboors.
- Determining whether two graphs are isomorphic is NP-intermediate. It is not known if a polynomial time algorithm exists, or the problem is NP-hard.
- Weisfeiler-Lehman test[1] provides a necessary (but not sufficient) condition to guarantee that two graphs are isomorphic.
  - Design an injective coloring function f_{WL} that takes a pair (node, its neighborhood) as input, and outputs a new node color :

iteration

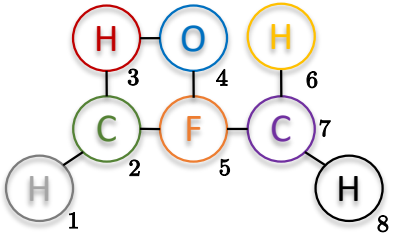
$$f_{\mathrm{WL}}\left(c_i^{\ell}, \{c_j^{\ell}\}_{j \in \mathcal{N}_i}\right) = c_i^{\ell+1}$$

Multiset (set of unordered and repetitive elements)



Graph 1

Isomorphic graphs

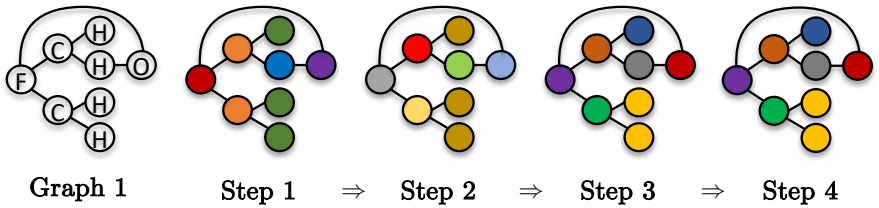


Graph 2

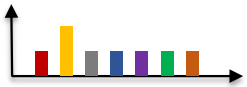
[1] Weisfeiler, Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

Weisfeiler-Lehman test[1]

- WL algorithm iteratively applies the coloring function f_{WL} until no new colors are created :

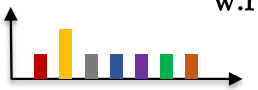


No new colors created, algorithm stops.

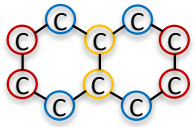


Isomorphic graphs w.r.t. WL test

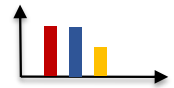
No new colors created, algorithm stops.



- However the 1-WL test can fail to distinguish (simple) non-isomorphic graphs :

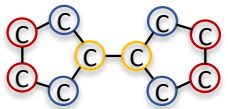


Graph 1 (Decalin)

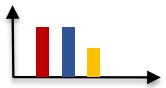


Histogram of colors

Isomorphic graphs w.r.t. WL test ⇔



Graph 2 (Bicyclopentyl)

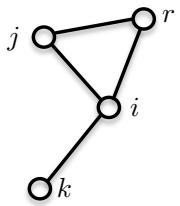


Histogram of colors

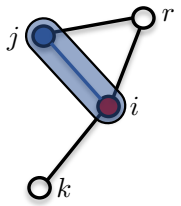
[1] Weisfeiler, Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

WL-GNNs

- GIN[1] designed to be as maximally expressive as the original 1-WL test[2].
- k-WL tests : Original test uses 2-tuple of nodes. To improve expressivity power of WL test, higher-order interactions between nodes with k-tuple of nodes with $k \geq 3$ can be used.
  - k-order equivariant GNNs[3] are theoretically more expressive but
    - These networks require $O(N^k)$ memory/speed complexities, with at least k=3 to be more powerful than GIN, which means $O(N^3)$ and thus not practical.
    - 3-WL/Ring/2-FGNN GNNs[4,5,6] have $O(N^2)$-memory but $O(N^3)$-speed complexities.
    - Expressivity does not necessarily imply generalization[7].

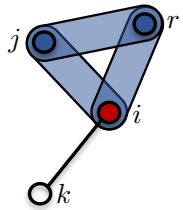


$\mathcal{G} = (V, E)$



Edges $= \{i, j\}, \{i, r\}, \{i, k\}$

2-tuple of nodes



Hyper-edges $= \{i, j, r\}, \{i, k, r\}, \{i, j, k\}$

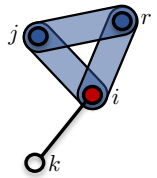
3-tuple of nodes

[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[2] Weisfeiler, Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968
[3] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[4] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[5] Chen, Villar, Chen, Bruna, On the equivalence between graph isomorphism testing and function approximation with gnns, 2019
[6] Azizian, Lelarge, Expressive power of invariant and equivariant graph neural networks, 2020
[7] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020

Outline

- Review of graph network architectures
- **Graph positional encoding**
- Transformers for sequences
- Graph transformers
- ViT/MLP-Mixer for images
- Graph ViT/MLP-Mixer
- Conclusion

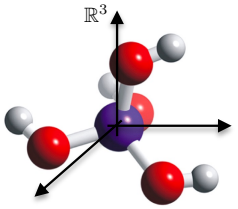
Higher expressivity power



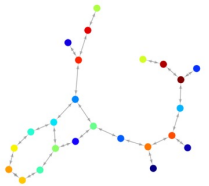
Hyper-edges $=\{i,j,r\}$, $\{i,k,r\},\{i,j,k\}$

- How to improve the limited expressivity of MP-GNNs ?
  - Consider higher-order node interactions (k-tuples with k>2) with WL-GNNs[1,2,3,4].
    - Theorem[5] (universal approximator) : Any continuous function invariant by permutation can be arbitrarily approximated by WL-GNNs, with the necessary condition the network has higher-order tensor of order k=poly(N)= N(N-1)/2.
    - These networks are computationally expensive, at least $O(N^3)$ for 3-WL expressivity.
  - Provide a unique ID for each node, i.e. positional encoding (PE).
    - Theorem[6] : MP-GNNs are provable more expressive than the 1-WL test when considering node positional encoding.
    - Theorem[7] : MP-GNNs are Turing-complete when depth $d \geq \delta_G$ layers (graph diameter), width is unbounded, and each node is uniquely identified.

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[2] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[3] Chen, Chen, Villar, Bruna, Can graph neural networks count substructures? 2020
[4] Azizian, Lelarge, Expressive power of invariant and equivariant graph neural networks, 2020
[5] Maron Fetaya, Segol, Lipman, On the universality of invariant networks, 2019
[6] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, 2019
[7] Loukas, What graph neural networks cannot learn: depth vs width, 2019

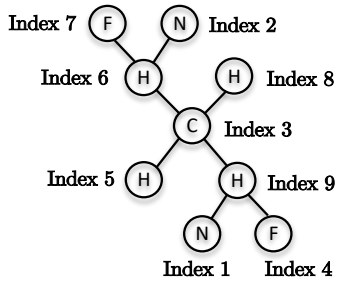


Geometric PE
(e.g. 3D coordinates)


Structural PE
(next slide)

Which structural positional encoding?

- Theory does not provide guidance on the choice of structural PE for the class of graphs and task.
- The simplest PE is an (arbitrary) indexing of the nodes[1], among N! possible indexings.
  - During training, indexing are uniformly sampled from the N! possible choices in order for the network to learn to be independent to these arbitrary choices.



- What are properties for designing meaningful PEs?
  - Uniqueness of representation for each node
  - Permutation-invariance
  - Distance-sensitivity : Nodes far apart on the graph should have different positional features whereas nodes nearby have similar positional features.

[1] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, 2019

Laplacian Positional Encoding[1]

- Laplacian eigenvectors[2] :
  - Spectral techniques that embed graphs into an Euclidean space.
  - Form a meaningful local coordinate system, while preserving the global graph structure.
  - Define via factorization (EVD) of the graph Laplacian matrix.
  - Computational complexity is $O(E^{3/2})$ and $O(N)$ with approximate Nystrom method[3].
- Laplacian eigenvectors as PE[1] :
  - Hybrid positional and structural encodings (invariant by index permutation).
  - Unique and distance-sensitive : two nodes far away on graph have large PE distance, and inversely.
  - LapPE are graph generalizations of Transformer's PE[4].

$$\Delta = I - D^{-1/2} A D^{-1/2} = U^T \Lambda U$$

Graph Laplacian — Degree matrix — Adjacency matrix — Eigenvalue matrix — (Laplacian) Eigenvector matrix

[1] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[2] Belkin, Niyogi, Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering, 2001
[3] Fowlkes, Belongie, Chung, Malik, Spectral grouping using the nystrom method, 2004
[4] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017

Limitation of Laplacian PE

- LapPE have natural symmetries with the arbitrary sign of eigenvectors (after being normalized to have unit length).
  - The number of possible sign flips is 2^K, k being the number of eigenvectors.
  - In practice, we choose $K \ll N$, and therefore 2^K is much smaller N! (the number of possible ordering of the nodes). Smaller sampling space than index PEs, and therefore smaller amount of ambiguities to be resolved by the network.
  - During the training, eigenvectors will be uniformly sampled at random between the 2^K possibilities[1].
- Despite this limitation, LapPE have good performance in practice[1].
- LapPE are used in GraphTransformer[2], Directed-GNN[3], SAN[4].

[1] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[2] Dwivedi, Bresson, A generalization of transformer networks to graphs, 2020
[3] Beani, Passaro, Letourneau, Hamilton, Corso, Lio, Directional graph networks, 2020
[4] Kreuzer, Beaini, Hamilton, Letourneau, Tossou, Rethinking Graph Transformers with Spectral Attention, 2021

Random Walk Positional Encoding[1]

- Diffusion-based PE :
  - Diffusion techniques capture graph geometry.
  - Popular graph diffusion processes :
    - Diffusion kernel[2,3], Random walk s.a. pageRank[4]
  - Computational complexity is O(N^2)
- Diagonal of K-step of RW matrix as PE[1] :
  - Memory complexity is O(N).
  - RWPE is not limited with the sign ambiguity like LapPE[5], i.e. the network is not required to learn additional invariance.
  - RWPE does not guarantee unique node ID and distance-sensitive (no manifold assumption[6]).
    - Uniqueness of node representation is under the condition that each node has a unique K-hop topological neighborhood for a sufficient large K.

$$p_i = \begin{bmatrix} \mathrm{RW}_{ii} \\ \mathrm{RW}_{ii}^2 \\ \vdots \\ \mathrm{RW}_{ii}^K \end{bmatrix} \in \mathbb{R}^K, \qquad \mathrm{RW} = AD^{-1}$$

Random Walk Matrix

Adjacency matrix

Degree matrix

[1] Dwivedi, Luu, Laurent, Bengio, Bresson, Graph neural networks with learnable structural and positional representations, 2021
[2] Kondor, Vert, Diffusion kernels, 2004
[3] Talmon, Cohen, Gannot, Coifman, Diffusion maps for signal processing, 2013
[4] Page, Brin, Motwani, Winograd, The PageRank citation ranking: Bringing order to the web, 1999
[5] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[6] Belkin, Niyogi, Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering, 2001

Learning structural and positional encodings[1]

- Standard MP-GNN inject the positional information into the input layer of the GNN[2] :

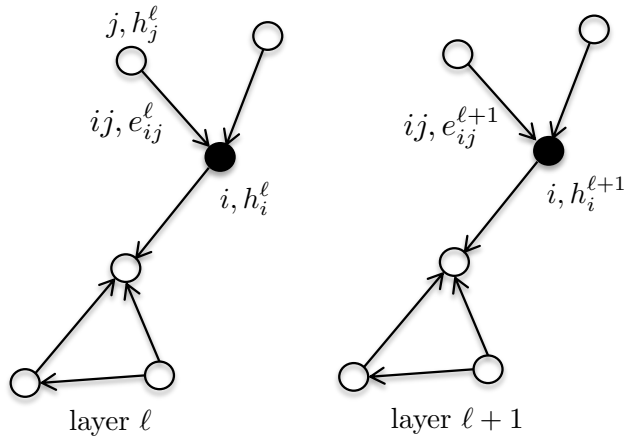
$$\text{Node-update}: \quad h_i^{\ell+1} = f_h(h_i^\ell, \{h_j^\ell\}_{j\in\mathcal{N}_i}, e_{ij}^\ell)$$

$$\text{Edge-update}: \quad e_{ij}^{\ell+1} = f_e(h_i^\ell, h_j^\ell, e_{ij}^\ell)$$

$$\text{with } h_i^{\ell=0} = \text{LE}(\text{Concat}(h_i^0, p_i))$$

graph PE

Input node feature



- We can decouple and learn structural and positional encodings separately :

$$\text{Node-update}: \quad h_i^{\ell+1} = f_h(h_i^\ell, p_i^\ell, \{h_j^\ell, p_j^\ell\}_{j\in\mathcal{N}_i}, e_{ij}^\ell)$$

$$\text{Edge-update}: \quad e_{ij}^{\ell+1} = f_e(h_i^\ell, h_j^\ell, e_{ij}^\ell)$$

$$\text{PE-update}: \quad p_i^{\ell+1} = f_p(p_i^\ell, \{p_j^\ell\}_{j\in\mathcal{N}_i})$$

[1] Dwivedi, Luu, Laurent, Bengio, Bresson, Graph neural networks with learnable structural and positional representations, 2021
[2] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020

GatedGCN[1] with learnable PE

- **Augmented architecture[2] :**

$$h^{\ell+1}, e^{\ell+1}, p^{\ell+1} = \text{GatedGCN+LearnPE}(h^\ell, e^\ell, p^\ell), \;\; h \in \mathbb{R}^{n\times d}, e \in \mathbb{R}^{E\times d}, p \in \mathbb{R}^{n\times K}$$

defined as

$$h_i^{\ell+1} = h_i^\ell + \text{ReLU}\Big(\text{BN}\Big(A_1^\ell \begin{bmatrix} h_i^\ell \\ p_i^\ell \end{bmatrix} + \sum_{j\in\mathcal{N}_i} \eta_{ij}^\ell \odot A_2^\ell \begin{bmatrix} h_j^\ell \\ p_j^\ell \end{bmatrix}\Big)\Big)$$

$$e_{ij}^{\ell+1} = e_{ij}^\ell + \text{ReLU}\Big(\text{BN}\Big(B_1^\ell h_i^\ell + B_2^\ell h_j^\ell + B_3^\ell e_{ij}^\ell\Big)\Big)$$

Positional representation that augments any MP-GNN

$$\eta_{ij}^\ell = \frac{\sigma(e_{ij}^\ell)}{\sum_{j'\in\mathcal{N}_i} \sigma(e_{ij'}^\ell) + \varepsilon}$$

$$p_i^{\ell+1} = p_i^\ell + \tanh\Big(C_1^\ell p_i^\ell + \sum_{j\in\mathcal{N}_i} \eta_{ij}^\ell \odot C_2^\ell p_j^\ell\Big) \;\; \text{with} \;\; \text{mean}(p_{:,k}^{\ell+1}) = 0, \;\; \|p_{:,k}^{\ell+1}\| = 1, \;\; \forall k$$

[1] Bresson, Laurent, Residual gated graph convnets, 2017
[2] Dwivedi, Luu, Laurent, Bengio, Bresson, Graph neural networks with learnable structural and positional representations, 2021

Positional encoding loss

- Positional loss can be considered.
  - Laplacian eigenvector loss[1,2] can be used to enforce the PE to form a coordinate system constrained by the graph topology.
- Final loss is composed of two terms :

$$L = L_{\text{Task}}(\begin{bmatrix} h^L \\ p^L \end{bmatrix}) + \alpha \; L_{\text{LapEig}}(p^L)$$

$$L_{\text{LapEig}}(p) = \text{trace}(p^T \Delta p) + \lambda \|p^T p - \text{I}_K\|_{\text{F}}^2$$

[1] Belkin, Niyogi, Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering, 2001
[2] Lai, Osher, A splitting method for orthogonality constrained problems, 2014

Outline

- Review of graph network architectures
- Graph positional encoding
- Transformers for sequences
- Graph transformers
- ViT/MLP-Mixer for images
- Graph ViT/MLP-Mixer
- Conclusion

RNNs for sequences

- RNNs like LSTM[1] are designed for sequences.
- Contributions
  - They learn representation of sequences independently of their length using a recurrence formula with weight sharing across time (translation invariance).
  - They process ordered sequences, i.e. 1D regular lattice, by design.
- Limitations
  - Hard to train because they are non-linear dynamical systems.
    - Any small perturbation can amplify or vanish.
  - Slow to train because of their sequential nature (unlike CNNs).
    - Important limitation with large-scale datasets.
  - RNNs cannot learn long-term dependencies (no more than 50 steps) between words because of over-squashing.

[1] Hochreiter, Schmidhuber, Long short-term memory, 1997

Over-squashing

- Most neural networks s.a. CNNs, RNNs, GNNs are message-passing techniques.
  - New representation is computed by aggregating (with f_{NN}) the neighborhood information $j \to i$:

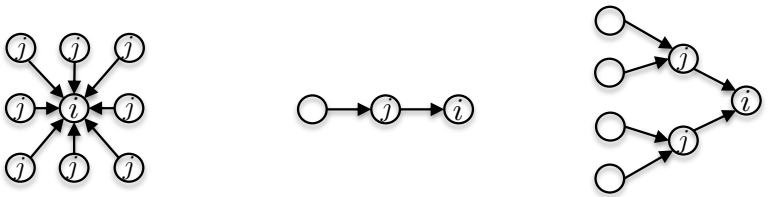
$$h_i^{\ell+1} = f_{\text{NN}}\left(h_i^{\ell}, \{h_j^{\ell}\}_{j \in \mathcal{N}_i \text{ or } j \to i}\right) \in \mathbb{R}^d$$

layer

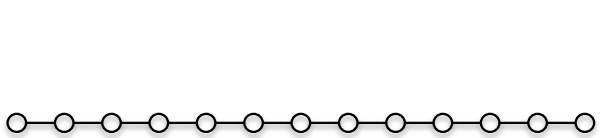
Learnable parametric function w/ inductive bias

Domain structure

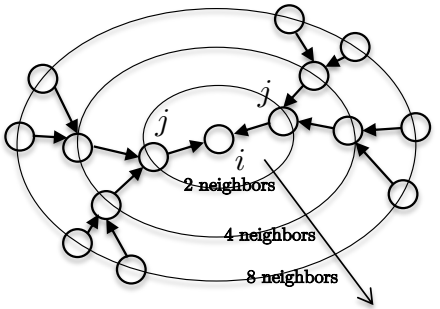
$j \to i$:



  - As the MP mechanism is local for CNNs/RNNs/GNNs, they require to stack multiple layers to propagate information over long-range distances (ideally over the whole sequence/graph).



Over-squashing

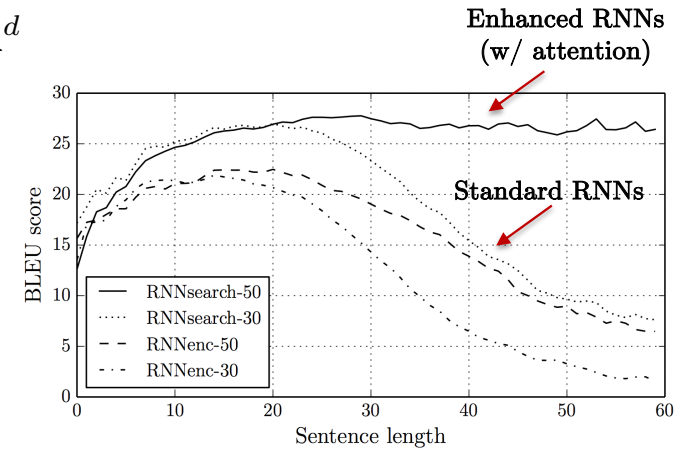


Nb of nodes in the reception field grows exponentially with nb of layers.

- Stacking layers to propagate information though the domain can be a major issue.
  - The reception field grows as $O(L^2)$ for CNNs, $O(L)$ for RNNs and $O(2^{L+1})$ for GNNs (for tree graphs), L being the number of layers.
- The reception field size gives the number of neighbors used to update the representation.
  - For RNNs, this requires aggregating $O(L)$ number of vectors and $O(2^{L+1})$ for GNNs.

$$h_i^{\ell+1} = F_{\text{RNN}}\left(h_i^0, h_{i-1}^0, h_{i-2}^0, ..., h_{i-L}^0\right) \in \mathbb{R}^d$$
$$= F_{\text{GNN}}\left(h_i^0, h_{i-1}^0, h_{i-2}^0, ..., h_{i-2^{L+1}}^0\right) \in \mathbb{R}^d$$

- This issue is known as over-squashing.
  - NNs w/ local reception field cannot learn long-range dependency.
  - Vanilla RNNs cannot process sequences more than 20 words[1].
  - This means that MP-GNNs cannot stack many layers.



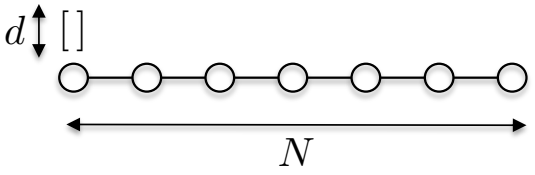
Figure[1]. Accuracy of translated sequences (higher is better).

[1] Bahdanau, Cho, Bengio, Neural machine translation by jointly learning to align and translate, 2014

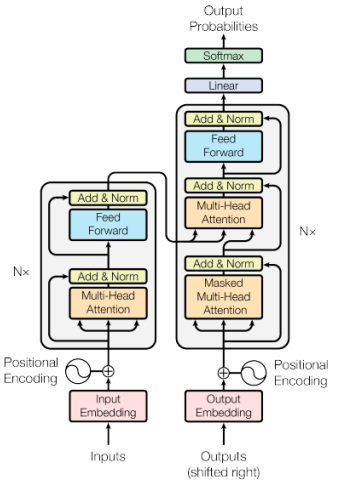
Transformers for sequences[1]

- Key ingredients of original Transformers
  - Attention mechanism (self/cross attention)
  - Layer normalization + residual connection + MLP
  - Cos/sin functions as node positional encoding

$h \in \mathbb{R}^{N \times d}$



- Contributions
  - Attention layers fixes the over-squashing issue by connecting all words together with pair-wise matching (attention scores).
  - Attention layers produce a dynamic representation of a word depending on the context (e.g. the vase broke, the news broke, Sandy broke the law, etc) ≠ word2vec.



$$h^{\ell+1} = \text{Softmax}(\frac{QK^T}{\sqrt{d}})V \in \mathbb{R}^{N \times d}$$

$$Q = h^\ell W_Q^\ell \in \mathbb{R}^{N \times d}$$

$$K = h^\ell W_K^\ell \in \mathbb{R}^{N \times d}$$

$$V = h^\ell W_V^\ell \in \mathbb{R}^{N \times d}$$

[1] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017

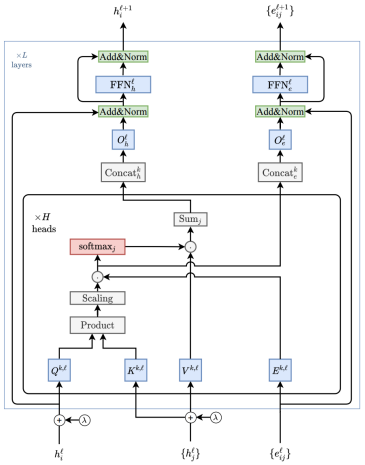
Limitations of Transformers

- Limitations :
  - They fix the issues of over-squashing and long-range dependency by "connecting everything to everything" but the attention mechanism is costly :
    - The reception field is the size of the domain O(N).
    - Memory/speed complexity is $O(N^2 d)$.
    - This limits Transformers on GPUs to only process short sequences, $N \leq 1{,}000$ for $d=512$.
  - They cannot process ordered sequences, i.e. 1D regular lattice.
    - Transformers are set neural networks – they are designed to process sets of data points.
    - They require additional data structure s.a. cos/sin functions as node positional encoding to represent explicitly the position of the words in the sequence.

Outline

- Review of graph network architectures
- Graph positional encoding
- Transformers for sequences
- **Graph transformers**
- ViT/MLP-Mixer for images
- Graph ViT/MLP-Mixer
- Conclusion

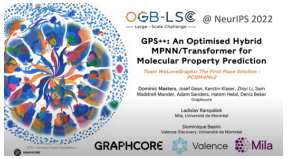
Graph Transformers[1]

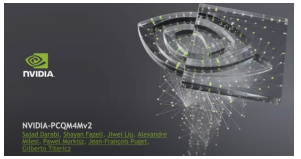


- Contributions
  - Generalize Transformers to graphs to overcome over-squashing and poor long-range dependency (same idea from NLP[2,3]).
  - Leverage graph as topological inductive bias.
  - Generalize cos/sin positional encoding to graphs (node ordering) with Laplacian eigenvectors.
  - Introduce edge features through bi-linear product (for e.g. molecular bound or relationships between entities in KG).
  - Popular class of architectures in biology (winner and runner-up at NeurIPS'22 OGB Large-Scale Challenge competition)[7]
- Limitations
  - Quadratic complexity $O(N^2+E)$, limited to small (molecular) graphs.

OGB-LSC @ NeurIPS 2022
Large - Scale Challenge

- 10:50PM–11:40PM, UTC: Graph-Level Track (PCQM4Mv2)
  - Intro to the task (5min) [Video]
  - Live presentation by 1st place winner: **WeLoveGraphs** (10min) [Video]
  - Live presentation by 2nd place winner: **NVIDIA-PCQM4Mv2** (10min) [Video]
  - Live presentation by 2nd place winner: **ViSNet** (10min) [Video]
  - Live Q&A and discussion (15min) [Video]





[1] Dwivedi, Bresson, A generalization of transformer networks to graphs, AAAI 2021
[2] Bahdanau, Cho, Bengio, Neural machine translation by jointly learning to align and translate, 2014
[3] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017
[4] Kreuzer, Beaini, Hamilton, Letourneau, Tossou, Rethinking Graph Transformers with Spectral Attention, 2021
[5] Mialon, Chen, Selosse, Mairal, GraphiT: Encoding Graph Structure in Transformers, 2021
[6] Ying, Cai, Luo, Zheng, Ke, He, Shen, Liu, Do Transformers Really Perform Bad for Graph Representation? 2021
[7] https://ogb.stanford.edu/neurips2022/workshop

Transformers[1]

- **Architecture**

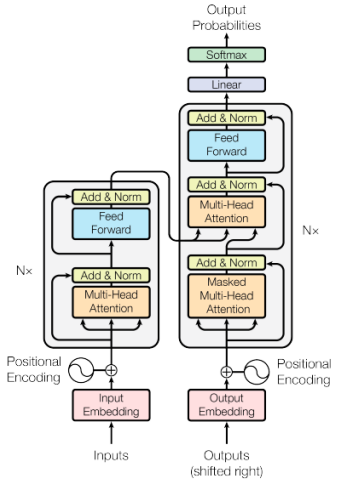
$$h^{\ell+1} = \mathrm{TR}(h^\ell),\ h^{\ell=0} = \mathrm{LE}(h^0) + \mathrm{PE},\ h^{\ell+1}, h^\ell \in \mathbb{R}^{n \times d}$$

defined as

$$\begin{cases} h^{\ell+1} & = \mathrm{LN}(\hat{h}^{\ell+1} + \mathrm{MLP}(\hat{h}^{\ell+1})) \\ h^{\ell+1} & = \mathrm{LN}(h^\ell + \mathrm{MHA}(h^\ell) \end{cases}$$

with

$$\mathrm{MHA}(h) = (\|_{m=1}^{M} X^m) W_o \in \mathbb{R}^{n \times d}$$



with X defined as (omitting m)

$$X = \mathrm{Softmax}(\frac{QK^T}{\sqrt{d/M}})V \in \mathbb{R}^{n \times \frac{d}{M}}$$

$$\text{and} \begin{cases} Q & = hW_Q \in \mathbb{R}^{n \times \frac{d}{M}} \\ K & = hW_K \in \mathbb{R}^{n \times \frac{d}{M}} \\ V & = hW_V \in \mathbb{R}^{n \times \frac{d}{M}} \end{cases}$$

and X in point-wise representation

$$x_i = \sum_{j \in V} \frac{w_{ij}}{\sum_{j' \in V} w_{ij'}} V_j$$

$$\text{with} \begin{cases} w_{ij} & = \exp(A_{ij}) \\ A_{ij} & = q_i k_j^T / \sqrt{d/M} \quad \text{(attention score)} \end{cases}$$

[1] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017

Graph Transformers[1]

- **Architecture**

$$h^{\ell+1}, e^{\ell+1} = \text{GraphTransformers}(h^\ell, e^\ell),\ h^{\ell+1}, h^\ell \in \mathbb{R}^{n\times d},\ e^{\ell+1}, e^\ell \in \mathbb{R}^{E\times d}$$

$$h^{\ell=0} = \text{LE}(h^0) + \text{LapPE}\ \ e^{\ell=0} = \text{LE}(e^0)$$

defined as

$$\begin{cases} h^{\ell+1} &= \text{LN}(\hat{h}^{\ell+1} + \text{MLP}(\hat{h}^{\ell+1})) \\ h^{\ell+1} &= \text{LN}(h^\ell + \text{G-MHA}(h^\ell, e^\ell) \\ e^{\ell+1} &= f_e(h^\ell, e^\ell)\ //\ e^{\ell+1} = e^{\ell=0} \end{cases}$$

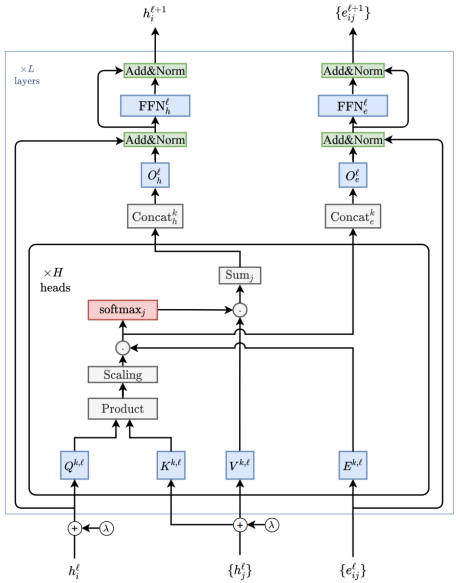
with

$$\text{G-MHA}(h) = (\|_{m=1}^{M} X^m) W_o \in \mathbb{R}^{n\times d}$$

with X in point-wise representation

$$x_i = \sum_{j\in\mathcal{N}_i} \frac{w_{ij}}{\sum_{j'\in\mathcal{N}_i} w_{ij'}} V_j$$

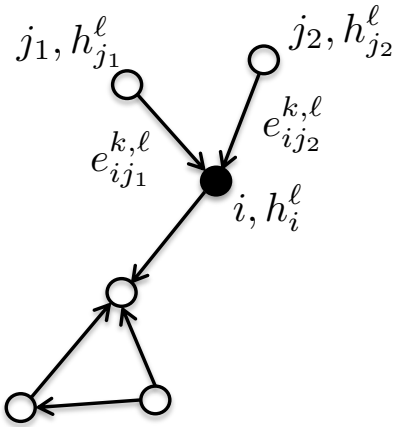
with $\begin{cases} w_{ij} &= \exp(A_{ij}) \\ A_{ij} &= q \cdot \text{diag}(c_{ij}) k_j^T / \sqrt{d/M} \end{cases}$ (attention score)



[1] Dwivedi, Bresson, A generalization of transformer networks to graphs, 2020

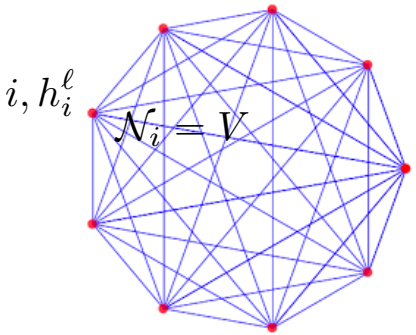
Consistency

- Graph Transformers[1] reduces to (original) Transformers[2] when
  - The graph is fully connected, i.e. $\mathcal{N}_i = V$
  - The value of edge feature is 1, i.e. $c_{ij} = 1$
  - The LapPE are cos/sin functions for sequences



Sparsely connected graph/
Graph Transformers

$\Rightarrow$



Fully connected graph/
Transformers

[1] Dwivedi, Bresson, A generalization of transformer networks to graphs, 2020
[2] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017

Improvements

- SAN[1]
  - Separate parameters for (existing) edges and complementary (non-existing) edges.
  - Weighted additive contribution between edges and non-existing edges.
- GraphiT[2]
  - Diffusion geometry captures short-range and long-range graph topology.
  - Multiplicative and pair-wise contribution.
  - Unique set of parameters for edges and non-existing edges (unlike SAN).

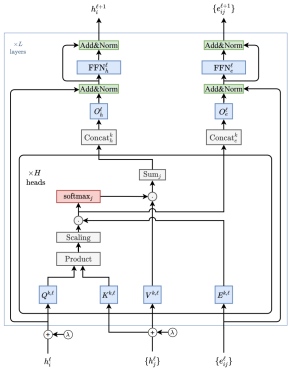
$$w_{ij} = \exp(\frac{q_i k_j^T}{\sqrt{d/M}}).K_{ij} \quad (K_{ij} \text{ is the diffusion distance})$$

- Graphormer[3]
  - Graph geometry captures short-range and long-range graph information with shortest paths.
  - Additive and pair-wise contribution for the attention score.

$$A_{ij} = q_i k_j^T / \sqrt{d/M} + c_{ij} \quad (c_{ij} \text{ is based on shortest path})$$

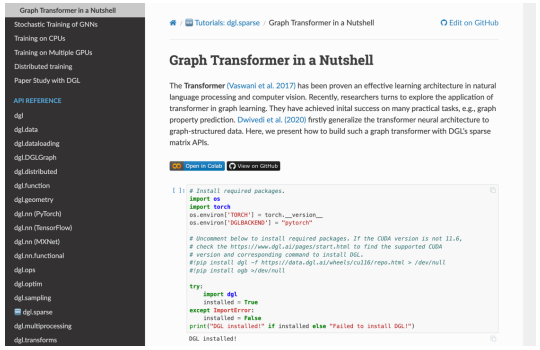
[1] Kreuzer, Beaini, Hamilton, Letourneau, Tossou, Rethinking Graph Transformers with Spectral Attention, 2021
[2] Mialon, Chen, Selosse, Mairal, GraphiT: Encoding Graph Structure in Transformers, 2021
[3] Ying, Cai, Luo, Zheng, Ke, He, Shen, Liu, Do Transformers Really Perform Bad for Graph Representation?, 2021

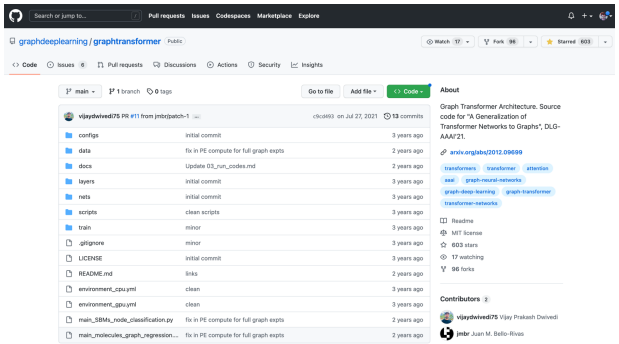
Graph Transformers



- Material
  - GitHub repo (600*+) : https://github.com/graphdeeplearning/graphtransformer
  - Blogpost (Vijay Dwivedi) : https://towardsdatascience.com/graph-transformer-generalization-of-transformers-to-graphs-ead2448cff8b
  - Andrew Ng's article : https://www.deeplearning.ai/the-batch/issue-151
  - DGL tutorial (10 lines of codes): https://docs.dgl.ai/en/latest/notebooks/sparse/graph_transformer.html

DeepLearning.AI
THE BATCH





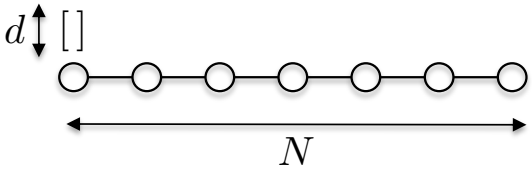
Outline

- Review of graph network architectures
- Graph positional encoding
- Transformers for sequences
- Graph transformers
- **ViT/MLP-Mixer for images**
- Graph ViT/MLP-Mixer
- Conclusion

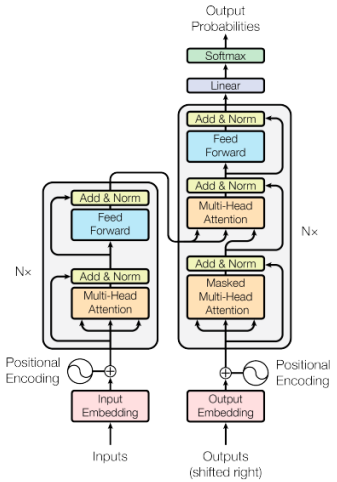
Transformers for long sequences

- Transformers[1] have been a breakthrough in NLP but the attention mechanism is computationally expansive since the memory/speed complexity is $O(N^2 d)$.
- This limits Transformer on GPUs to short sequences,

  $N \leq 1{,}000$ for $d=512$.
- How to scale-up to long sequences?
  - Linear approximations of the attention function[2,3].



$$h \in \mathbb{R}^{N \times d}$$



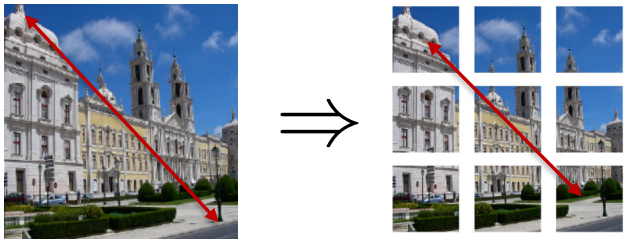
[1] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017
[2] Choromanski et-al, Rethinking attention with performers, 2020
[3] Jaegle, Gimeno, Brock, Vinyals, Zisserman, Carreira, Perceiver: General perception with iterative attention, 2021

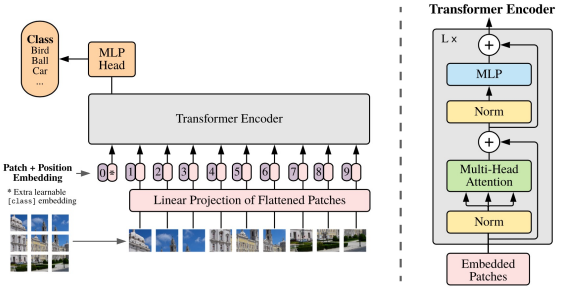
ViT/MLP-Mixer for images

- How to scale Transformer to image grids (CV)?

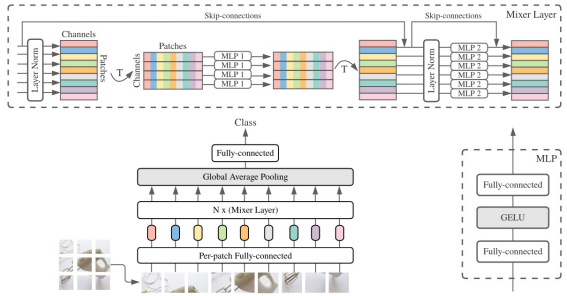
  E.g. N = 1,000 x 1,000 pixels = 1M pixels/nodes

- ViT architecture[1] :
  - Perform attention not on image pixels but on image patches : $O(N^2d) \searrow O(P^2d)$ w/ $P \ll N$ patches
  - Embed patches with MLP (or CNN with stride/kernel = patch size)
  - Same layer normalization + residual connection + MLP
  - Raster ordering of patches as node positional encoding
- MLP-Mixer architecture[2] :
  - Replacing the costly attention function with simple MLPs reduces complexity from $O(P^2d) \searrow O(Pd)$ with comparable performance.
  - Token embedding + token mixer layer is enough to capture long-range dependency in images.





Vision Transformer (ViT)



MLP-Mixer

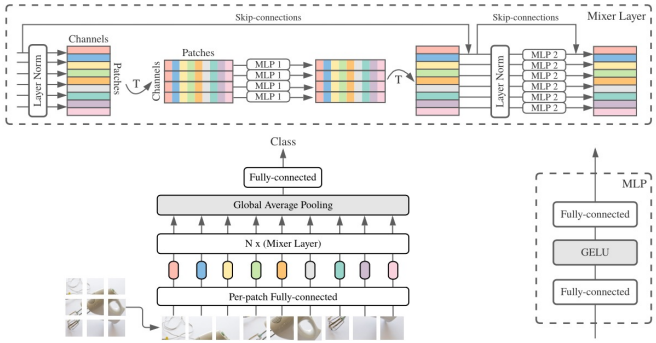
[1] Dosovitskiy et-al, An image is worth 16x16 words: Transformers for image recognition at scale, 2020
[2] Tolstikhin et-al, MLP-mixer: An all-MLP architecture for vision, 2021

Outline

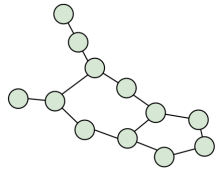
- Review of graph network architectures
- Graph positional encoding
- Transformers for sequences
- Graph transformers
- ViT/MLP-Mixer for images
- **Graph ViT/MLP-Mixer**
- Conclusion

Why MLP-Mixer for graphs?

- Standard MLP-Mixer[2] captures long-range interaction in images with low complexity.
- Our goal is to transfer these advantages to GNNs and design a new network that simultaneously
  - captures long-range dependency (mitigating the over-squashing issue),
  - keeps linear speed/memory complexity (similar to standard MP-GNNs),
  - achieves isomorphism expressivity (good representation power).
- However, generalizing MLP-Mixer is challenging due to the variable nature of graphs.



MLP-Mixer for images[1]



MLP-Mixer for graphs

[1] Tolstikhin et-al, MLP-mixer: An all-MLP architecture for vision, 2021

Generalization challenges

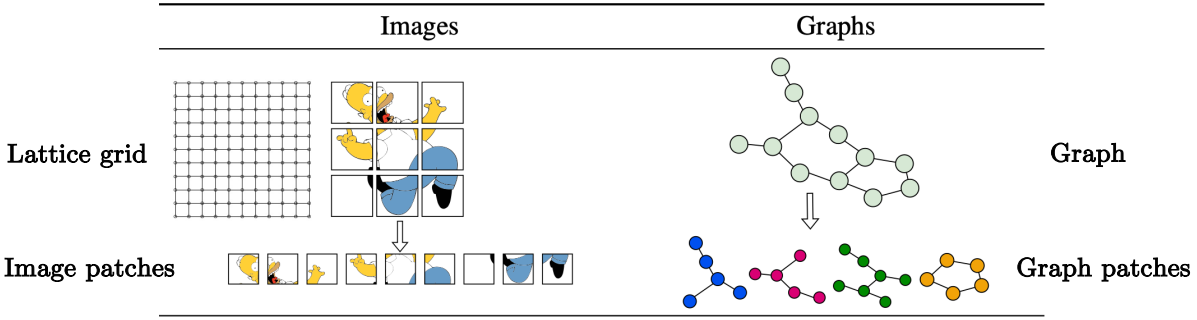
We identify four key challenges to generalize MLP-Mixer from grids to graphs.

(1) How to define and quickly extract graph patches(/tokens)?

- Images are supported by a regular lattice, easily split into same-size grid-like patches via fast pixel reordering.
- Graphs are irregular and cannot be divided into similar patches.

(2) How to encode graph patches into a vectorial representation?

- Same-size patches can be encoded with MLP.
- Graph patches have different topological structures (variable #nodes and #edges)



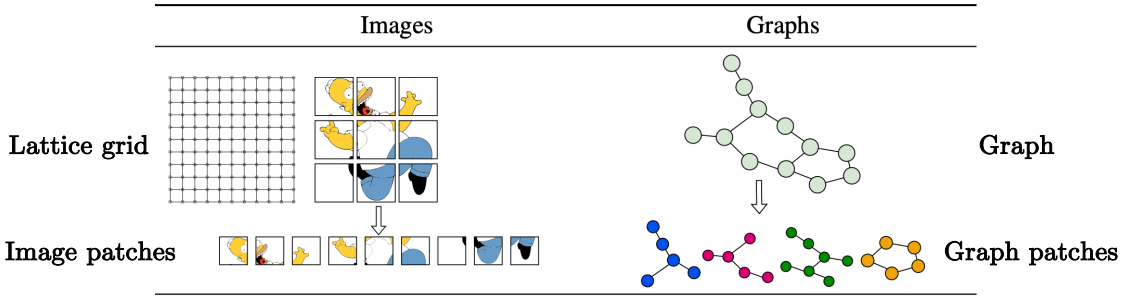
Generalization challenges

(3) How to preserve node and patch positional information?

- Image patches have implicit positions since images are always ordered the same way, but graphs are naturally not-aligned and the set of graph patches are therefore unordered.
- Pixels in each patch are ordered the same way, but nodes in graph tokens are generally unordered.

(4) How to reduce over-fitting for graphs?

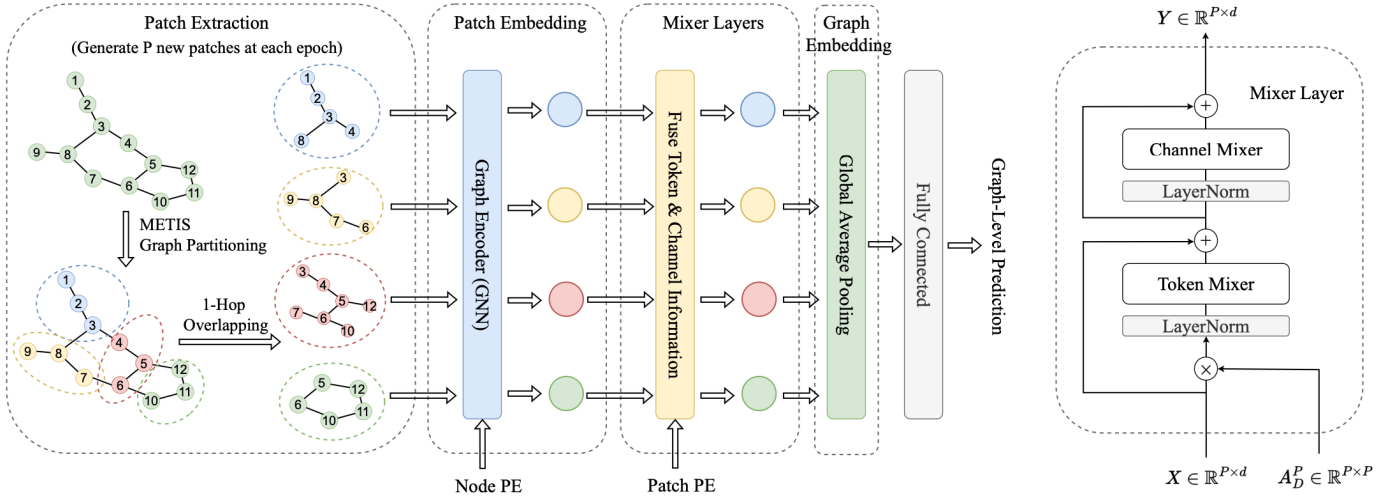
- MLP-Mixer architectures are strong over-fitters. Images have a rich set of data augmentation and regularization techniques, e.g. cropping, flipping, RandAugment[1], mixup[2].
- Graph data augmentation methods[3] are not yet as effective.



[1] Cubuk, Zoph, Shlens, V Le, Randaugment: Practical automated data augmentation with a reduced search space, 2020
[2] Zhang, Cisse, Dauphin, Lopez-Paz, mixup: Beyond empirical risk minimization, 2017
[3] Zhao, Liu, Neves, Woodford, Jiang, Shah, Data augmentation for graph neural networks, 2020

Proposed architecture[1]

- Generic architecture of Graph MLP-Mixer is illustrated below.
- Implementation of each building block is arbitrary.
- Our choices lead to a simple framework that provides speed and accuracy.

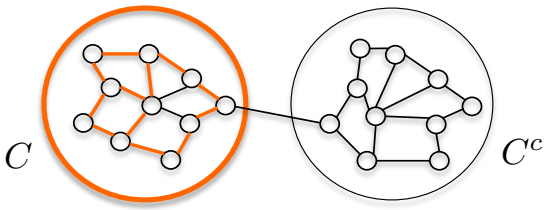


Graph MLP-Mixer

[1] He, Hooi, Laurent, Perold, LeCun, Bresson, A Generalization of ViT/MLP-Mixer to Graphs, 2022

Patch extraction

- Graph patch extraction algorithm must satisfy the following conditions :
  - The clustering algorithm can be applied to any arbitrary graph.
  - The nodes in the patch must be more connected than for those outside the patch (s.a. compound of atoms).
  - The extraction must be fast, i.e. complexity at most linear w.r.t. the number of edges O(E).
- Graph partitioning algorithms have a long and rich development[1,2].
- They are NP-hard combinatorial problems and approximations are required.
- We select Metis[3] that
  - Partitions a graph by maximizing the number of within-cluster links.
  - Provides one of the best trade-off accuracy and speed.



$$\max_C \frac{\text{Assoc}(C,C)}{\text{Vol}(C)} + \frac{\text{Assoc}(C^c,C^c)}{\text{Vol}(C^c)}$$

$$\text{Assoc}(C,C) = \sum_{i\in C, j\in C} A_{ij} \quad \text{(Adj. matrix)}$$

$$\text{Vol}(C) = \sum_{i\in C} d_i \quad \text{(degree)}$$

Normalized Association

[1] Von Luxburg, A tutorial on spectral clustering, 2007
[2] Buluc, Meyerhenke, Safro, Sanders, Schulz, Recent advances in graph partitioning, 2016
[3] Karypis, Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, 1998

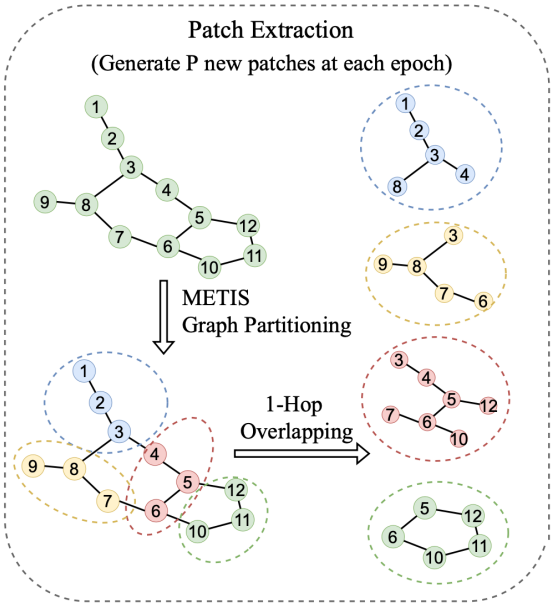
Metis limitations

- Two major limitations

  (1) Metis produces non-overlapping patches and loses the cutting edges. We expand the graph patches with a k-hop neighborhood.

  (2) Metis is a deterministic process, it always generates the same clusters which leads to over-fitting.

- Data augmentation
  - At each epoch, Metis is applied on a perturbed graph (by randomly dropping a small set of edges) to get slightly different partitions.
  - Then graph patches are extracted from the original graph (not the perturbed one) to retain the original nodes and edges.
  - This new dropout technique produces distinct graph patches at each epoch that significantly improves the results at a small additional computational time.



Patch encoder

- Image patch embedding is fast and well-defined with a simple MLP.
- Graph patch embedding must handle heterogeneous structure, variable patch size, index permutation invariance, weight sharing and capable of distinguishing isomorphic patches.
- As a result, the graph patch encoder is a MP-GNN (e.g. GCN[1], GAT[2], GT[3]) which can map a graph token into a fixed-size representation into 3 steps :

(1) Raw node and edge embedding :

$$h_i^0 = T^0 p_i + U^0 \alpha_i + u^0 \in \mathbb{R}^d, \quad e_{ij}^0 = V^0 \beta_{ij} + v^0 \in \mathbb{R}^d$$

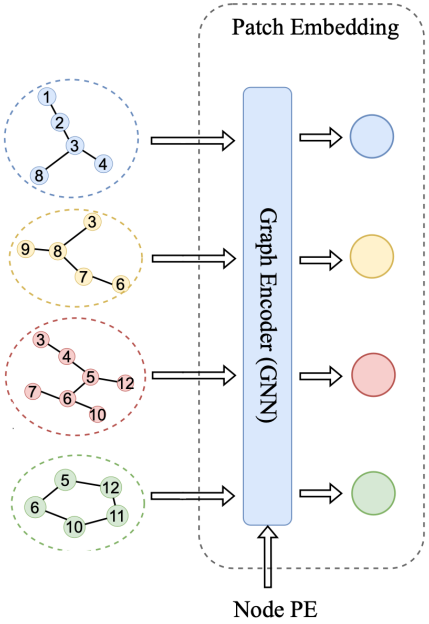
Positional encoding — Raw node features — Raw edge features

(2) Graph convolutional layers with MP-GNN :

$$h_{i,p}^{\ell+1} = f_{\text{node}}(h_{i,p}^\ell, \{h_{j,p}^\ell, e_{ij,p}^\ell | j \in \mathcal{N}(i)\}) + g_{\text{patch-n}}(h_p^\ell) \in \mathbb{R}^d$$
$$e_{ij,p}^{\ell+1} = f_{\text{edge}}(h_{i,p}^\ell, h_{i,p}^\ell, e_{ij,p}^\ell) + g_{\text{patch-e}}(e_p^\ell) \in \mathbb{R}^d$$

(3) Pooling and readout : $x_{G_p} = \text{MLP}(h_p) \in \mathbb{R}^d, \quad h_p = \frac{1}{|\mathcal{V}_p|} \sum_{i \in \mathcal{V}_p} h_{i,p}^{\ell=L} \in \mathbb{R}^d$

Patch embedding



- Note that the MP-GNN is applied to small patch graphs, which are not affected by poor long-range interaction and over-squashing.

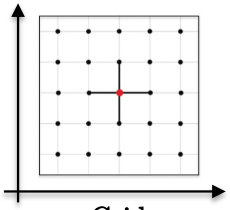
[1] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[2] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph Attention Networks, 2018
[3] Dwivedi, Bresson, A generalization of transformer networks to graphs, 2021

Positional information

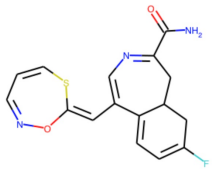
- Regular grid offers an implicit arrangement for the sequence of image patches and for the pixels inside the image patch.
- General graphs do not have a canonical ordering of nodes and patches. This lack of positional information reduces the expressivity of the network.
- We use two explicit positional encoding (PE) :
  - Absolute node PE : We use random-walk structural encoding[1] for molecular data and Laplacian eigenvectors[2,3] encodings for synthetic graph datasets.
  - Relative patch PE : We use a n-step random walk diffusion process[4] on the the adjacency matrix of the patch graphs (computed from the original graph adjacency matrix and the cluster extracted by Metis) :

$$A_{\mathcal{D}}^{P} = \left(D^{-1}A^{P}\right)^{n} \in \mathbb{R}^{P\times P}$$

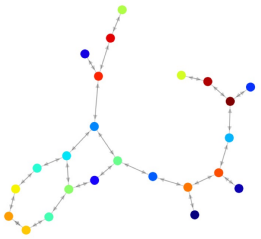
$$A_{ij}^{P} = |\mathcal{V}_i \cap \mathcal{V}_j| = \mathrm{Cut}(\mathcal{V}_i, \mathcal{V}_j) = \sum_{k\in\mathcal{V}_i}\sum_{l\in\mathcal{V}_j} A_{kl}$$



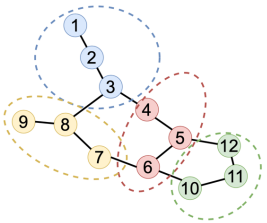
Grid



Graphs



Node PE



Patch PE

[1] Dwivedi, Luu, Laurent, Bengio, Bresson, Graph neural networks with learnable structural and positional representations, 2021
[2] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[3] Dwivedi, Bresson, A generalization of transformer networks to graphs, 2021
[4] Kondor, Vert, Diffusion kernels, 2014

Mixer layer and graph embedding

- The original mixer layer[1] is a network that alternates channel and token mixing steps with two MLPs. These interleaved steps fuse token and channel information.
  - The simplicity of the mixer layer was important to understand the attention mechanism is not the only key component to get good performance.
  - This has led to a significant reduction in the computational cost, from $O(P^2d)$ to $O(Pd)$, with almost the same performance.
- Let $X \in R^{P \times d}$ be the patch embedding matrix, then the graph mixer layer is defined as

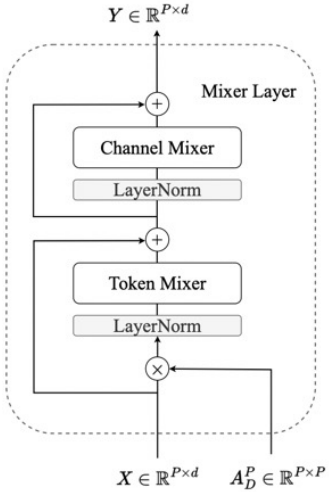
Patch PE

$$U = X + (W_2\sigma(W_1 \text{ LayerNorm}(A_{\mathcal{D}}^{P}X))) \in \mathbb{R}^{P \times d} \quad \text{Token mixer}$$

$$Y = U + (W_4\sigma(W_3\text{LayerNorm}(U)^T))^T \in \mathbb{R}^{P \times d} \quad \text{Channel mixer}$$

- The final graph-level representation is given by mean pooling all the (non-empty) patches and using a small MLP :

$$y_G = \text{MLP}(h_G) \in \mathbb{R}/\mathbb{R}^{n_c}, \quad h_G = \sum_p m_p \cdot x_{G_p} / \sum_p m_p \ \in \mathbb{R}^d$$



[1] Tolstikhin et-al, MLP-mixer: An all-MLP architecture for vision, 2021

Numerical experiments

- We evaluate Graph MLP-Mixer on a range of benchmark graph datasets :

  1) Simulated datasets : CSL[1], EXP[2], SR25[3] and TreeNeighbourMatch[4] datasets.

  2) Small molecular datasets : ZINC from Benchmarking GNNs[5] (for solubility) and MolTOX21 (for toxicity) and MolHIV from Open Graph Benchmark (OGB)[6] (for HIV inhibition).

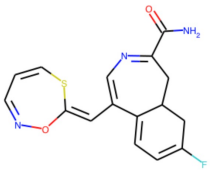
  3) Large molecular datasets : Peptides-func (for antibacterial, antiviral properties etc) and Peptides-struct (for 3D properties) from Long Range Graph Benchmark (LRGB)[7]

[1] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, 2019
[2] Abboud, Ceylan, Grohe, Lukasiewicz, The surprising power of graph neural networks with random node initialization, 2020
[3] Balcilar, Heroux, Gauzere, Vasseur, Adam, Honeine, Breaking the limits of message passing graph neural networks, 2021
[4] Alon, Yahav, On the bottleneck of graph neural networks and its practical implications, 2020
[5] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[6] Hu et-al, Open graph benchmark: Datasets for machine learning on graphs, 2020
[7] Dwivedi, Rampavek, Galkin, Parviz, Wolf, Luu, Beaini, Long range graph benchmark, 2022

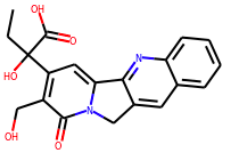
Benchmark datasets

- Statistics of molecular and synthetic datasets :

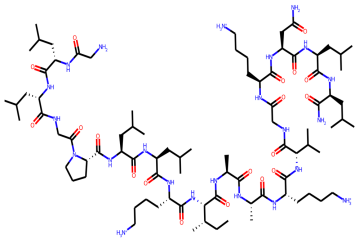
	Dataset	#Graphs	#Nodes	Avg. #Nodes	Avg. #Edges	Task	Metric
Simulated datasets	CSL	150	41	41	164	10-class classif.	Accuracy
	EXP	1,200	32-64	44.4	110.2	2-class classif.	Accuracy
	SR25	15	25	25	300	15-class classif.	Accuracy
	TreeNeighbourMatch (r=2)	96	7	7	6	4-class classif.	Accuracy
	TreeNeighbourMatch (r=3)	32,000	15	15	14	8-class classif.	Accuracy
	TreeNeighbourMatch (r=4)	64,000	31	31	30	16-class classif.	Accuracy
	TreeNeighbourMatch (r=5)	128,000	63	63	62	32-class classif.	Accuracy
	TreeNeighbourMatch (r=6)	256,000	127	127	126	64-class classif.	Accuracy
	TreeNeighbourMatch (r=7)	512,000	255	255	254	128-class classif.	Accuracy
	TreeNeighbourMatch (r=8)	640,000	511	511	510	256-class classif.	Accuracy
Small molecular datasets	ZINC	12,000	9-37	23.2	24.9	regression	MAE
	MolTOX21	7,831	1-132	18.57	38.6	12-task classif.	ROCAUC
	MolHIV	41,127	2-222	25.5	54.9	binary classif.	ROCAUC
Large molecular datasets	Peptides-func	15,535	8-444	150.9	307.3	10-class classif.	Avg. Precision
	Peptides-struct	15,535	8-444	150.9	307.3	regression	MAE



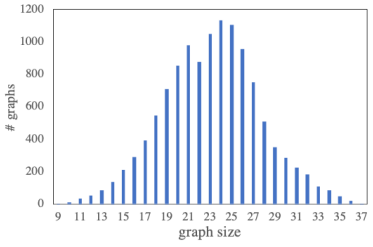
ZINC



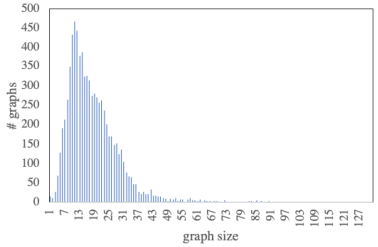
MolHIV



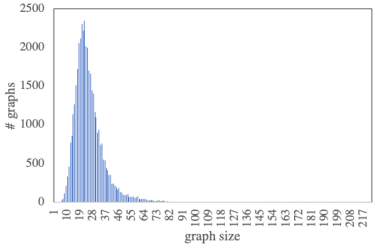
Peptide



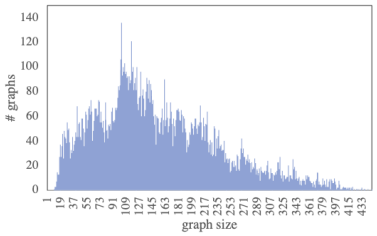
ZINC



MolTOX21



MolHIV



Peptides-func/struct

Comparison with standard MP-GNNs

- Graph MLP-Mixer lifts the performance of all base MP-GNNs across all datasets, which include GCN[1], GatedGCN[2], GINE[3] and GraphTransformer[4].
- We augmented all base models with the same PEs as Graph MLP-Mixer to ensure fair comparison.
- These promising results demonstrate the generic nature of our architecture which can be applied to any MP-GNN in practice.

Model	ZINC	MolTOX21	MolHIV	Peptide-func	Peptide-struct
	MAE ↓	ROCAUC ↑	ROCAUC ↑	Avg. Precision ↑	MAE ↓
GCN	0.1952 ± 0.0057	0.7525 ± 0.0031	0.7813 ± 0.0081	0.6328 ± 0.0086	0.2758 ± 0.0012
GCN-MLP-Mixer	**0.1323 ± 0.0026**	**0.7829 ± 0.0058**	**0.7912 ± 0.0121**	**0.6810 ± 0.0067**	**0.2492 ± 0.0011**
GatedGCN	0.1577 ± 0.0046	0.7641 ± 0.0057	0.7874 ± 0.0119	0.6300 ± 0.0029	0.2778 ± 0.0017
GatedGCN-MLP-Mixer	**0.1249 ± 0.0020**	**0.7861 ± 0.0048**	**0.8073 ± 0.0037**	**0.6856 ± 0.0044**	**0.2484 ± 0.0016**
GINE	0.1072 ± 0.0037	0.7730 ± 0.0064	0.7885 ± 0.0034	0.6405 ± 0.0077	0.2780 ± 0.0021
GINE-MLP-Mixer	**0.0745 ± 0.0014**	**0.7866 ± 0.0022**	**0.7951 ± 0.0077**	**0.6921 ± 0.0054**	**0.2485 ± 0.0004**
GraphTrans	0.1230 ± 0.0018	0.7646 ± 0.0055	0.7884 ± 0.0104	0.6313 ± 0.0039	0.2777 ± 0.0025
GraphTrans-MLP-Mixer	**0.0798 ± 0.0032**	**0.7874 ± 0.0044**	**0.7892 ± 0.0116**	**0.6795 ± 0.0063**	**0.2475 ± 0.0015**

Table: Results are averaged over 4 runs with 4 different seeds.

[1] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[2] Bresson, Laurent, Residual gated graph convnets, 2017
[3] Hu, Liu, Gomes, Zitnik, Liang, Pande, Leskovec, Strategies for pre-training graph neural networks, 2019
[4] Dwivedi, Bresson, A generalization of transformer networks to graphs, 2021

Comparison with SOTA results

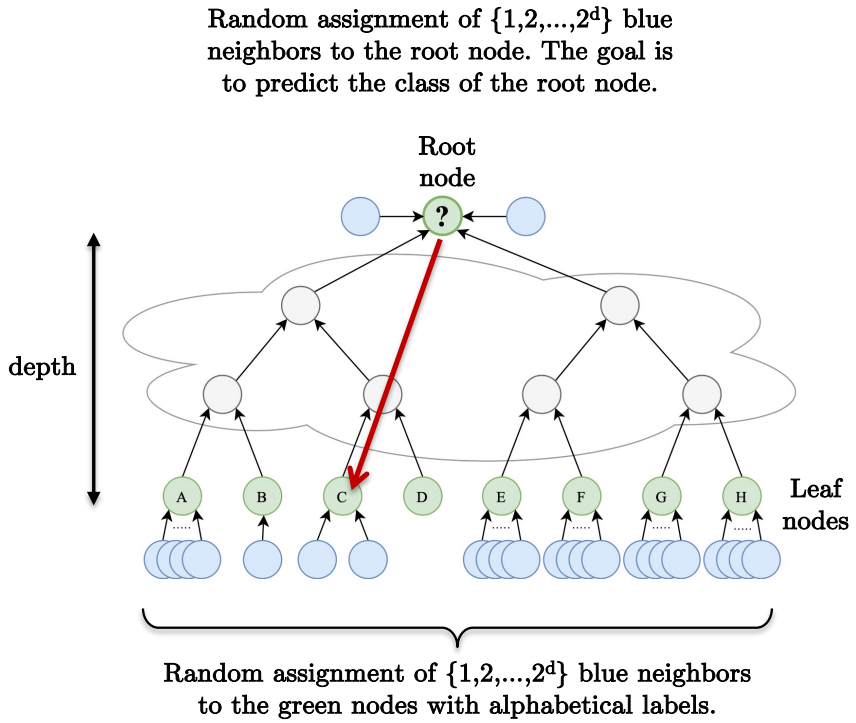
- We compare against GNN models with SOTA results
  - Graph Transformers: GraphiT, GPS, SAN, etc
  - Expressive GNNs: GNN-AK$_+$ and SUN
- For small molecular graphs, our model achieved competitive results on ZINC and MolHIV.
- For larger molecular graphs, our model sets SOTA performance on Peptides-fun/struct.
- Graph MLP-Mixer offers better space-time complexity and scalability.
  - SAN+LapPE and SUN require 7.5× and 41× training time per epoch, and 12× and 18× memory respectively, compared to our model.

Model	ZINC	MolHIV	Peptides-func			Peptides-strcut		
	MAE ↓	ROCAUC ↑	Avg. Precision ↑	Time/Epoch	Memory	MAE ↓	Time/Epoch	Memory
GT [36]	0.226 ± 0.014	–	–	–	–	–	–	–
GraphiT [53]	0.202 ± 0.011	–	–	–	–	–	–	–
Graphormer [56]	0.122 ± 0.006	–	–	–	–	–	–	–
GPS [57]	**0.070 ± 0.004**	0.7880 ± 0.0101	0.6562 ± 0.0115	1.3×	6.9×	0.2515 ± 0.0012	1.1×	6.6×
SAN+LapPE [38]	0.139 ± 0.006	0.7775 ± 0.0061	0.6384 ± 0.0121	8.8×	12.4×	0.2683 ± 0.0043	7.4×	11.8×
SAN+RWSE [38]	–	–	0.6439 ± 0.0075	7.5×	19.6×	0.2545 ± 0.0012	6.5×	11.7×
GNN-AK+ [31]	0.080 ± 0.001	0.7961 ± 0.0119	0.6480 ± 0.0089	2.5×	7.8×	0.2736 ± 0.0007	2.1×	7.3×
SUN [32]	0.084 ± 0.002	0.8003 ± 0.0055 [2]	0.6730 ± 0.0078	41.3×	18.8×	0.2498 ± 0.0008	35.7×	16.6×
Graph MLP-Mixer	0.075 ± 0.001	**0.8073 ± 0.0037**	**0.6921 ± 0.0054**	1.0×	1.0×	**0.2475 ± 0.0015**	1.0×	1.0×

Table: Results are averaged over 4 runs with 4 different seeds.

Synthetic dataset for over-squashing

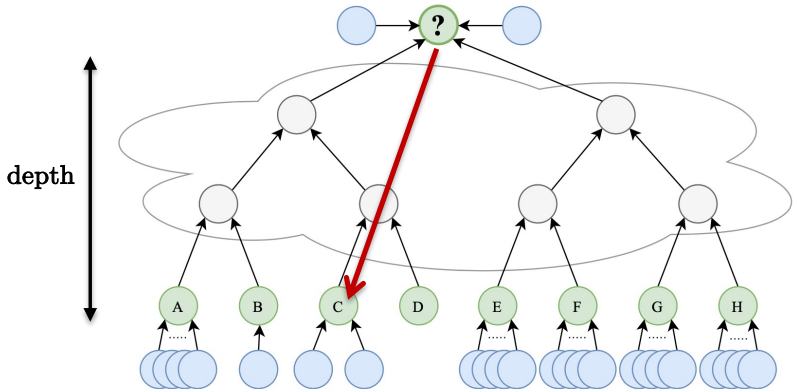
- TreeNeighboursMatch[1] is a synthetic dataset to simulate the exponentially-growing receptive field in GNNs and controls the intensity of over-squashing with the tree depth.
- The experiment consists in classifying the root node with the letter of the green labeled node with the same number of blue neighbors than the root node.
- To succeed, the network must be able to communicate across long distance (from the root to the leaf) and extract the information from $O(2^{L+1})$ aggregated nodes.



[1] Alon, Yahav, On the bottleneck of graph neural networks and its practical implications, 2020

Graph MLP-Mixer mitigates over-squashing[1]

- Standard MP-GNNs s.a. GCN, GGCN, GAT, GIN fail to generalize for depth ≥ 4 because of over-squashing that squeezes too much info from the exponential growth of the tree reception field.
- Our model is able to mitigate over-squashing and generalize until depth = 7 since it transmits the long-distance information directly with the mixer layer.



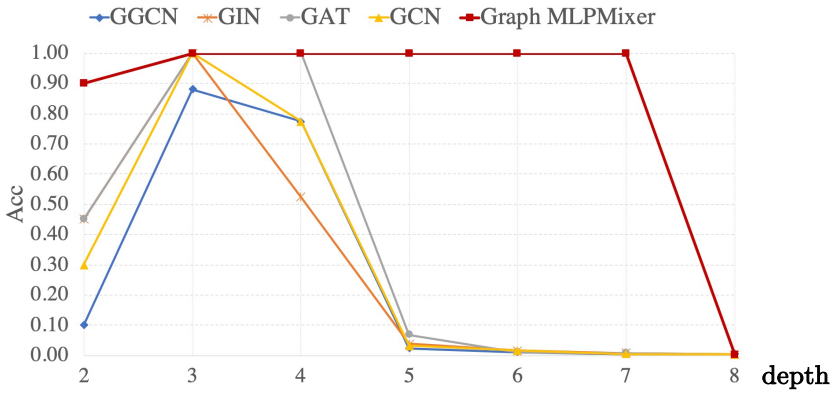
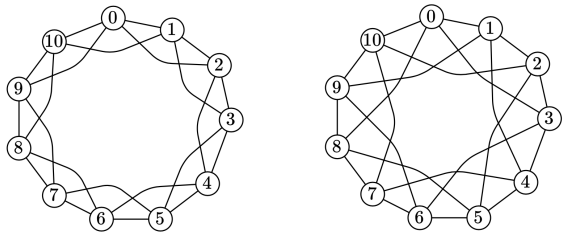


Figure. Test Accuracy w.r.t. the tree depth in the NEIGHBORSMATCH problem.

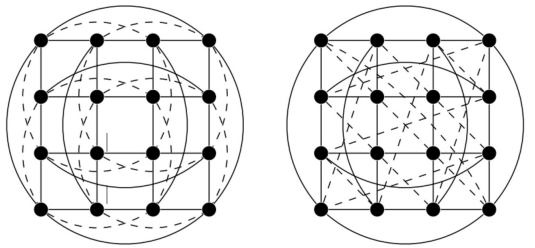
[1] Alon, Yahav, On the bottleneck of graph neural networks and its practical implications, 2020

Synthetic datasets for expressivity

- Circular Skip Link (CSL) dataset[1]
  - Classify isomorphic graphs that differ by the skip link value.
  - CSL contains 150 4-regular graphs (1-WL failed) divided into 10 isomorphism classes.
- EXP dataset[2]
  - EXP contains 1200 1&2-WL failed graphs that are split into two classes.
- SR25 dataset[3]
  - SR25 has 15 strongly regular 3-WL failed graphs with 25 nodes each with the goal of classifying them.



Example of non-isomorphic graphs where 1-WL test failed to distinguish (w/ skip-link 2 and 3).



Example of non-isomorphic graphs where 3-WL test failed to distinguish (strongly regular graphs known as 4×4-Rook and Shrikhande graphs)

[1] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, 2019
[2] Abboud, Ceylan, Grohe, Lukasiewicz, The surprising power of graph neural networks with random node initialization, 2020
[3] Balcilar, Heroux, Gauzere, Vasseur, Adam, Honeine, Breaking the limits of message passing graph neural networks, 2021

Graph MLP-Mixer achieves high expressivity

- We show empirically that graphs that cannot be distinguished by 1-WL[1], 2-WL[2] and 3-WL[3] tests are easily distinguished by the proposed Graph MLP-Mixer (where all standard MP-GNNs fail).

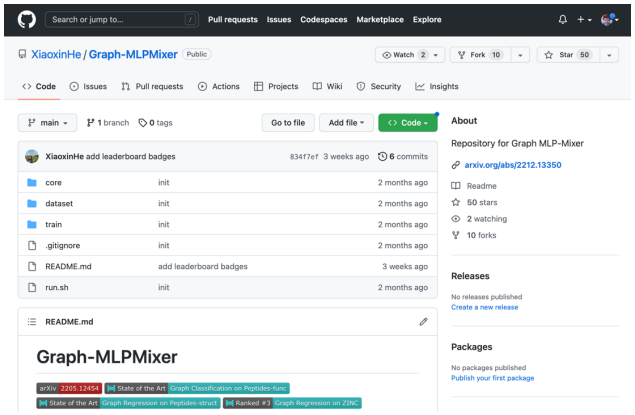
Model	CSL (ACC)	EXP (ACC)	SR25 (ACC)
GCN	10.00 ± 0.00	51.90 ± 1.96	6.67 ± 0.00
GatedGCN	10.00 ± 0.00	51.73 ± 1.65	6.67 ± 0.00
GINE	10.00 ± 0.00	50.69 ± 1.39	6.67 ± 0.00
GraphTrans	10.00 ± 0.00	52.35 ± 2.32	6.67 ± 0.00
GCN-MLP-Mixer	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
GatedGCN-MLP-Mixer	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
GINE-MLP-Mixer	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
GraphTrans-MLP-Mixer	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00

Table: Results are averaged over 4 runs with 4 different seeds.

[1] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, 2019
[2] Abboud, Ceylan, Grohe, Lukasiewicz, The surprising power of graph neural networks with random node initialization, 2020
[3] Balcilar, Heroux, Gauzere, Vasseur, Adam, Honeine, Breaking the limits of message passing graph neural networks, 2021

Graph MLP-Mixer

- Graph ViT/MLP-Mixer
  - Offer an architecture at the boundary between standard MP-GNNs and Graph Transformers.
  - Improve MP-GNN limitations (poor long-range dependency and low expressivity power) and have better complexity than Graph Transformers (linear complexity).
- GitHub repo : https://github.com/XiaoxinHe/Graph-MLPMixer



Outline

- Review of graph network architectures
- Graph positional encoding
- Transformers for sequences
- Graph transformers
- ViT/MLP-Mixer for images
- Graph ViT/MLP-Mixer
- **Conclusion**

Conclusion

- Developments of GNNs
  - Vanilla GNNs ⇒ Spectral GNNs ⇒ Anisotropic GNNs ⇒ WL-GNNs ⇒ Graph Transformers ⇒ Graph ViT/MLP-Mixer
- What are the next potential architecture improvements?
  - Large-scale GNNs (with hierarchical property?)
  - Computationally efficient GNNs, i.e. as fast as CNNs/Transformers (with standard hardware for dense linear algebra or new hardware for fast sparse linear algebra?)

Questions?