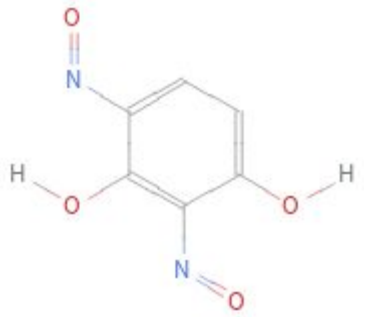
Graph neural networks for molecular property prediction

HIV replication inhibition and PCBA multiple assay prediction

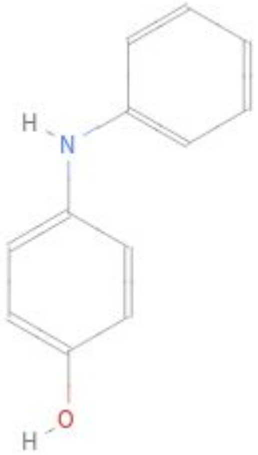
William Bruns
Stanford XCS224W student

Which molecule inhibits HIV replication?

(I’ll make it easy by giving you a choice between 2 molecules, guess and you will be right 50% of the time)



O=Nc1ccc(O)c(N=O)c1O

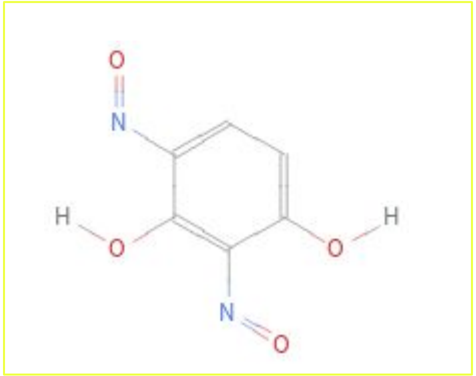


Oc1ccc(Nc2ccccc2)cc1

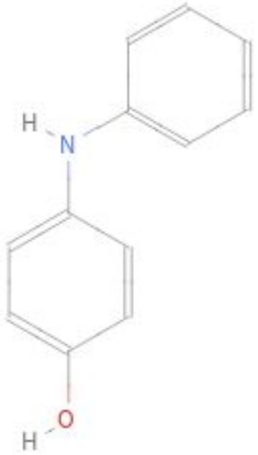
Molecule graphics from wolframalpha.com generated from Simplified Molecular Input Line Entry System (SMILES) strings from OGBG molhiv dataset
Example SMILES from https://snap.stanford.edu/ogb/data/graphproppred/csv_mol_download/hiv.zip
One of the first SMILES two class adjacent examples in hiv/mapping/mol.csv.gz (mapped to train split using train.csv, unmarked in this file) (during training OGB loader and official splits are used instead).
The very first example pair I was going to use, O=C(O)Cc1ccc(SSc2ccc(CC(=O)O)cc2)cc1 vs O=C(O)c1ccccc1SSc1ccccc1C(=O)O, was actually ambiguous, see https://pubchem.ncbi.nlm.nih.gov/compound/8409#section=Biological-Test-Results and search “aids”

Which molecule inhibits HIV replication?

(I'll make it easy by giving you a choice between 2 molecules, guess and you will be right 50% of the time)



O=Nc1ccc(O)c(N=O)c1O
https://pubchem.ncbi.nlm.nih.gov/bioassay/179#sid=68320

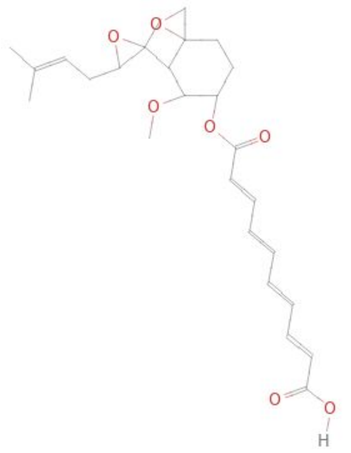


Oc1ccc(Nc2ccccc2)cc1
https://pubchem.ncbi.nlm.nih.gov/bioassay/179#sid=68322

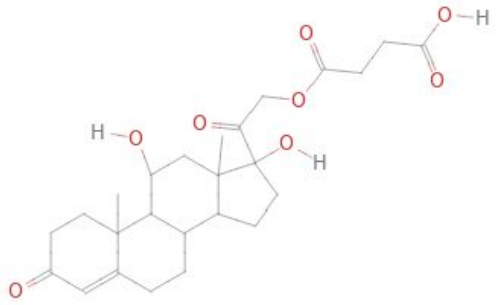
Molecule graphics from wolframalpha.com generated from Simplified Molecular Input Line Entry System (SMILES) strings from OGBG molhiv dataset
Example SMILES from https://snap.stanford.edu/ogb/data/graphproppred/csv_mol_download/hiv.zip
One of the first SMILES two class adjacent (original: CM vs CI) examples in hiv/mapping/mol.csv.gz (mapped to train split using train.csv, unmarked in this file) (during training OGB loader and official splits are used instead).
The very first example pair I was going to use, O=C(O)Cc1ccc(SSc2ccc(CC(=O)O)cc2)cc1 vs O=C(O)c1ccccc1SSc1ccccc1C(=O)O, was actually ambiguous, see https://pubchem.ncbi.nlm.nih.gov/compound/8409#section=Biological-Test-Results and search "aids"

Which molecule inhibits HIV replication?

(I’ll make it easy by giving you a choice between 2 molecules, guess and you will be right 50% of the time)



COC1C(OC(=O)C=CC=CC=CC=CC(=O)O)CCC2(CO2)C1C1(C)OC1CC=C(C)C

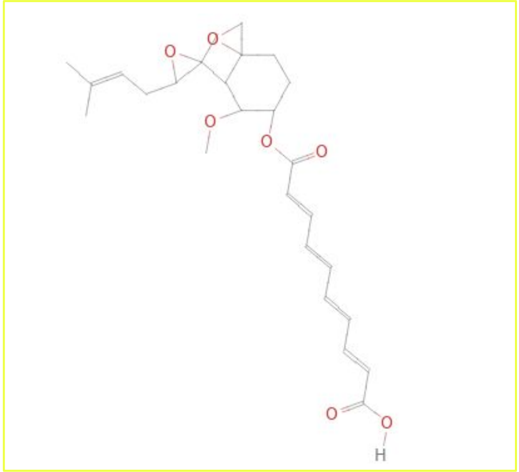


CC12CCC(=O)C=C1CCC1C2C(O)CC2(C)C1CCC2(O)C(=O)COC(=O)CCC(=O)O

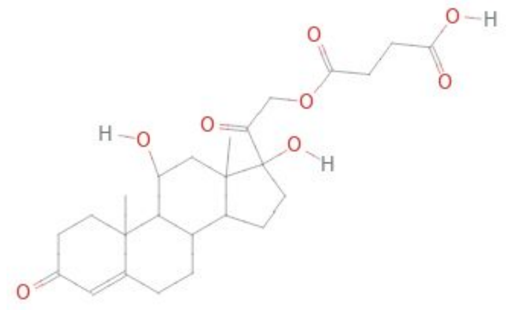
Molecule graphics from wolframalpha.com generated from SMILES strings from OGBG molhiv dataset
Example SMILES from https://snap.stanford.edu/ogb/data/graphproppred/csv_mol_download/hiv.zip
2 random SMILES, 1 from each class from hiv/mapping/mol.csv.gz (mapped to train split using train.csv, unmarked in this file) (during training OGB loader and official splits are used instead)

Which molecule inhibits HIV replication?

(I'll make it easy by giving you a choice between 2 molecules, guess and you will be right 50% of the time)



Aka Fumagillin, which is not only a HIV replication inhibitor, btw: "Originally isolated from the fungus Aspergillus fumigatus, it is used for the control of Nosema infection in honey bees. It has a role as an angiogenesis inhibitor, an antibacterial drug, an antiprotozoal drug, a methionine aminopeptidase 2 inhibitor, an antimicrobial agent and a fungal metabolite."



COC1C(OC(=O)C=CC=CC=CC=CC(=O)O)CCC2(CO2)C1C1(C)OC1CC=C(C)C
https://pubchem.ncbi.nlm.nih.gov/bioassay/179#sid=74694

CC12CCC(=O)C=C1CCC1C2C(O)CC2(C)C1CCC2(O)C(=O)COC(=O)CCC(=O)O
https://pubchem.ncbi.nlm.nih.gov/bioassay/179#sid=539584

Molecule graphics from wolframalpha.com generated from SMILES strings from OGBG molhiv dataset
Example SMILES from https://snap.stanford.edu/ogb/data/graphproppred/csv_mol_download/hiv.zip
2 random SMILES, 1 from each class (original CA vs CI) from hiv/mapping/mol.csv.gz (mapped to train split using train.csv, unmarked in this file) (during training OGB loader and official splits are used instead)

Can a computer predict this?
Why do we care?

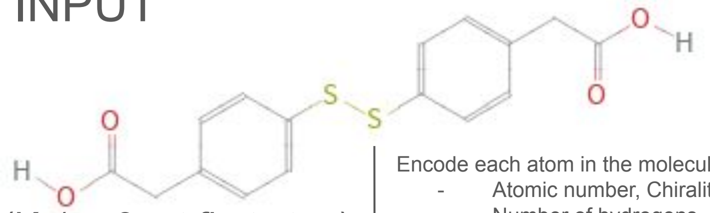
“Time and money are precious resources when the vast majority of compounds fail to reach FDA approval and those that do cost $1.2 billion on average to research and develop.
When searching for lead molecules, it **costs about $100 to purchase a single compound in a commercially available library;** in the lead optimization phase, it costs about $2500 to synthesize a proposed derivative; up to another $2500 for functional assays of candidate ligands; and the subsequent mouse-model and human studies that follow a successful lead optimization campaign cost exponentially more.
A simple back-of-the-envelope calculation shows that experimentally testing all 100 million purchasable compounds in the ZINC small molecule database is financially intractable for even the best funded laboratories. Even then, the ZINC database is a small portion of the vast combinatorial expanse that is drug-like chemical space.”

- Evan Feinberg of Stanford’s Pande Lab 2018
  https://medium.com/@pandelab/ai-for-drug-discovery-in-two-stories-49d7b1f019f3

(The Pande Lab owns the MoleculeNet benchmark datasets (https://arxiv.org/abs/1703.00564) whose molHIV data is basis for OGB’s molhiv dataset shown earlier.
OGB/SNAP data is at https://snap.stanford.edu/ogb/data/graphproppred/csv_mol_download/hiv.zip (with train/valid/test splits),
derived from Pande Lab’s https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/HIV.csv ,
which is from a public dataset originally https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data , see also https://pubchem.ncbi.nlm.nih.gov/bioassay/179
. Note the predictions are made on “Active”/”Inactive” binary classification but moderately vs confirmed active are available in original dataset and numerical EC50 values are available in the AID 179 dataset.)

Tl;dr - We can predict: Use a Graph Neural Network

INPUT



(K-1 = 0, at first step)

Encode each atom in the molecule by 9 features:

- Atomic number, Chirality, Degree, Formal Charge
- Number of hydrogens, Number of radical electrons, Hybridization
- Aromatic, In ring

Multiply 9 features by Linear layer to get node embedding

OUTPUT

Classification
POSITIVE

MLP

G

Sum all atom (node) embeddings to get molecule (graph) embed

for each atom

$h_v^{(k-1)}$

S

k := k + 1
repeat loop

k < # layers?

yes

no

S

final atom embedding

S

,

Aggregate with encoded neighbors

S

C

$\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}$

Multi-layer perceptron (MLP)

S

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right)$$

(This is a GIN, see Xu et al 2019 in references)

Just a preview! We will get here!

Molecule shown is
https://pubchem.ncbi.nlm.nih.gov/bioassay/179#sid=67143

Inspiration: GNNs in the news

-

Healthcare:

- "Discovery of a structural class of antibiotics with explainable deep learning"
(Wong et al 2023, https://www.nature.com/articles/s41586-023-06887-8) (uses Chemprop, GNN library)
(molecular property prediction is NOT specific to antibiotics or antibacterials)

- "Massively Multitask Networks for Drug Discovery" (Ramsundar et al 2015, https://arxiv.org/abs/1502.02072 ; team that introduced MoleculeNet which is the basis of some OGB datasets including ogbg-molhiv)

- "Modeling Polypharmacy Side Effects with Graph Convolutional Networks" Zitnik et al 2018, https://arxiv.org/pdf/1802.00543
(via XCS224W)

- See also many references (separate from above) in Leskovec's CS224W lecture 1.2 "Applications of Graph ML"

- Still machine learning on graphs but predicting protein structures:
AlphaFold, RoseTTAFold

- Designing amino acid sequences that fold to a specified structure: ProteinMPNN

SOTA Weather Forecasting:
- GraphCast uses GNN architecture Graph Isomorphism Network (GIN) to make global 10 day weather forecasts computable in 1 minute on a single machine that rival 6 hour national supercomputer forecasts
https://deepmind.google/discover/blog/graphcast-ai-model-for-faster-and-more-accurate-global-weather-forecasting/

Science generally:
- Artificial Intelligence for Science in Quantum, Atomistic, and Continuum Systems
https://arxiv.org/abs/2307.08423

-

Objective: Stanford OGB benchmarks

+ Stretch goal of predicting PCBA-577-WNV (open data, not benchmark)

Open Graph Benchmark has multiple task types:

- Node attribute prediction
- Edge prediction
- Graph property prediction
    - Single-task
        - OGBG MoleculeNet MolHIV replication inhibition challenge
            - 41,127 molecules, 80/10/10 train/val/test splits, metric ROCAUC
            - Started with this
    - Multi-task
        - OGBG MoleculeNet PubChem BioAssay 128 multitask challenge
            - 437,929 molecules, metric AP
            - Working on this now

MoleculeNet

A Benchmark for Molecular Machine Learning

A work by Pande Group at Stanford

uses

OPEN GRAPH BENCHMARK

uses

PubChem

NIH National Library of Medicine

National Center for Biotechnology Information

Stretch goal later:

- non-OGB Single task -> Predict PubChem BioAssay #577 West Nile Virus NS2bNS3 Proteinase inhibition
        - No current approved antivirals for West Nile Virus available

Tools

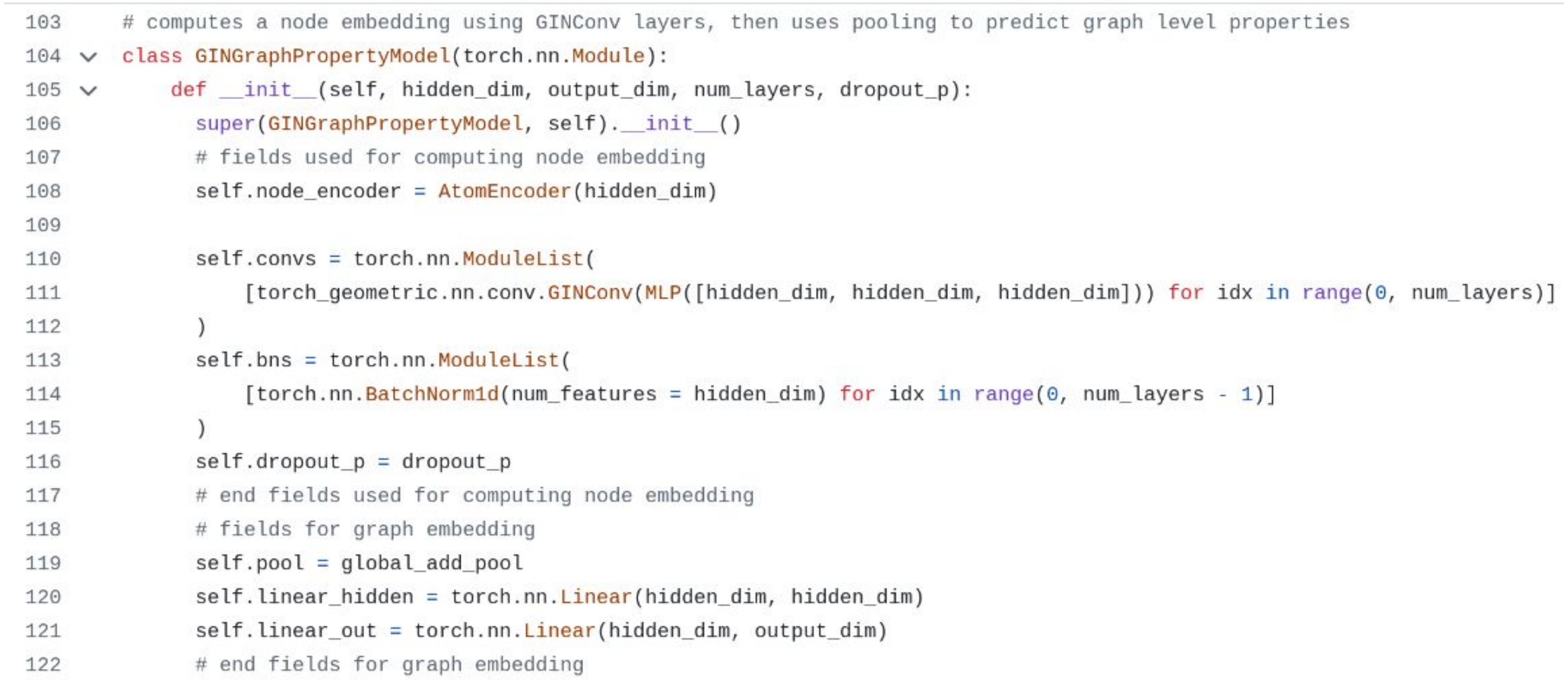
- Data: OGB + MoleculeNet
    - Hu, Weihua and Fey, Matthias and Zitnik, Marinka and Dong, Yuxiao and Ren, Hongyu and Liu, Bowen and Catasta, Michele and Leskovec, Jure. Open Graph Benchmark: Datasets for Machine Learning on Graphs. arXiv preprint arXiv:2005.00687, 2020.
    - Wu, Zhenqin and Ramsundar, Bharath and Feinberg, Evan N and Gomes, Joseph and Geniesse, Caleb and SPappu, Aneesh and Leswing, Karl and Pande, Vijay. Moleculenet: a benchmark for molecular machine learning. Chemical Science, 9(2):513–530, 2018.
- Modeling: PyG + GIN
    - PyTorch Geometric (https://pyg.org/):
      Fey, Matthias and Lenssen, Jan E. Fast Graph Representation Learning with PyTorch Geometric. ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019. (Graph Isomorphism Network (GIN) implementation used)
    - Graph Isomorphism Network:
      Xu, Keyulu and Hu, Weihua and Leskovec, Jure and Jegelka, Stefanie. How Powerful Are Graph Neural Networks? International Conference on Learning Representations, 2019. https://openreview.net/forum?id=ryGs6iA5Km , https://arxiv.org/pdf/1810.00826 . (Graph Isomorphism Network (GIN) original paper)

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum\nolimits_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right)$$

Approach: Tiny GIN

(32K parameters vs OGB team 1.8M parameter model)

https://github.com/willy-b/tiny-GIN-for-ogbg-molhiv



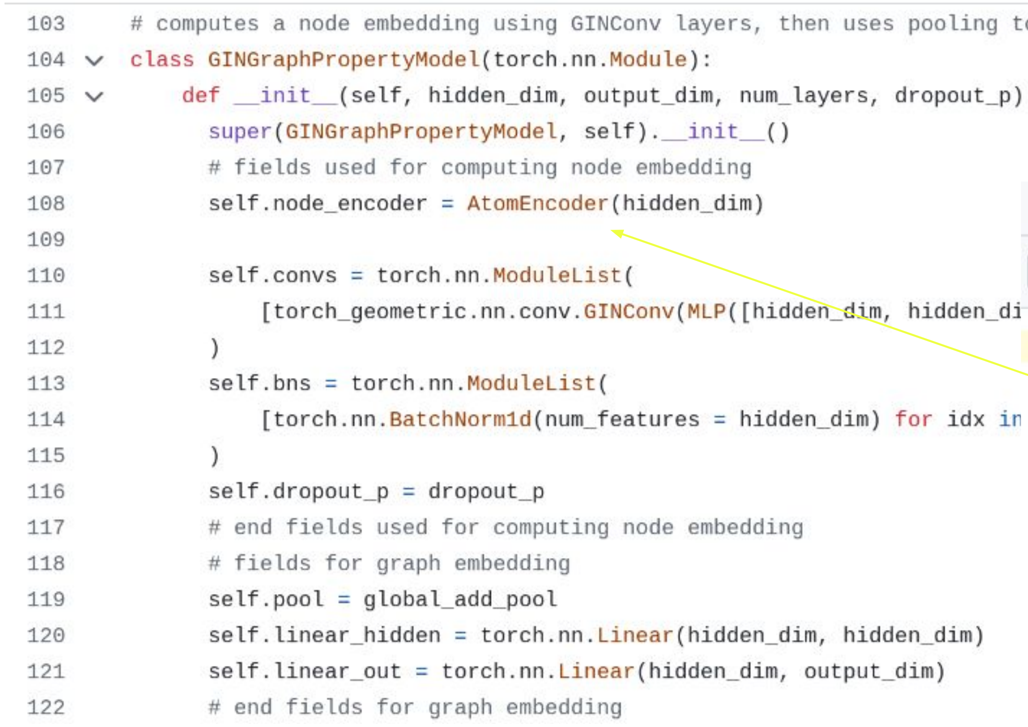
```python
# computes a node embedding using GINConv layers, then uses pooling to predict graph level properties
class GINGraphPropertyModel(torch.nn.Module):
    def __init__(self, hidden_dim, output_dim, num_layers, dropout_p):
      super(GINGraphPropertyModel, self).__init__()
      # fields used for computing node embedding
      self.node_encoder = AtomEncoder(hidden_dim)

      self.convs = torch.nn.ModuleList(
          [torch_geometric.nn.conv.GINConv(MLP([hidden_dim, hidden_dim, hidden_dim])) for idx in range(0, num_layers)]
      )
      self.bns = torch.nn.ModuleList(
          [torch.nn.BatchNorm1d(num_features = hidden_dim) for idx in range(0, num_layers - 1)]
      )
      self.dropout_p = dropout_p
      # end fields used for computing node embedding
      # fields for graph embedding
      self.pool = global_add_pool
      self.linear_hidden = torch.nn.Linear(hidden_dim, hidden_dim)
      self.linear_out = torch.nn.Linear(hidden_dim, output_dim)
      # end fields for graph embedding
```

Approach: Tiny GIN

(32K parameters vs OGB team 1.8M parameter model)

https://github.com/willy-b/tiny-GIN-for-ogbg-molhiv

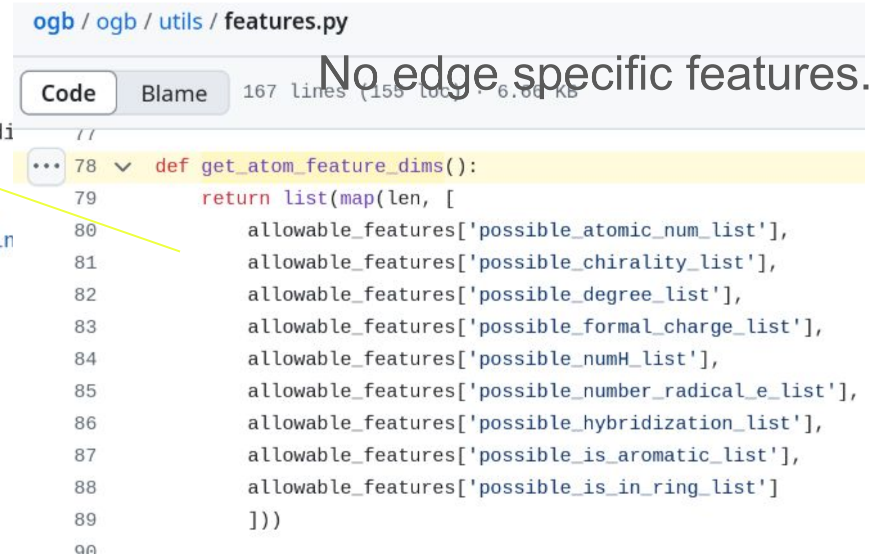


```
# computes a node embedding using GINConv layers, then uses pooling to predict graph level properties
class GINGraphPropertyModel(torch.nn.Module):
    def __init__(self, hidden_dim, output_dim, num_layers, dropout_p):
        super(GINGraphPropertyModel, self).__init__()
        # fields used for computing node embedding
        self.node_encoder = AtomEncoder(hidden_dim)

        self.convs = torch.nn.ModuleList(
            [torch_geometric.nn.conv.GINConv(MLP([hidden_dim, hidden_di
        )
        self.bns = torch.nn.ModuleList(
            [torch.nn.BatchNorm1d(num_features = hidden_dim) for idx in
        )
        self.dropout_p = dropout_p
        # end fields used for computing node embedding
        # fields for graph embedding
        self.pool = global_add_pool
        self.linear_hidden = torch.nn.Linear(hidden_dim, hidden_dim)
        self.linear_out = torch.nn.Linear(hidden_dim, output_dim)
        # end fields for graph embedding
```

Using OGB AtomEncoder 9 feature Atom representation.

No edge specific features.



ogb / ogb / utils / features.py

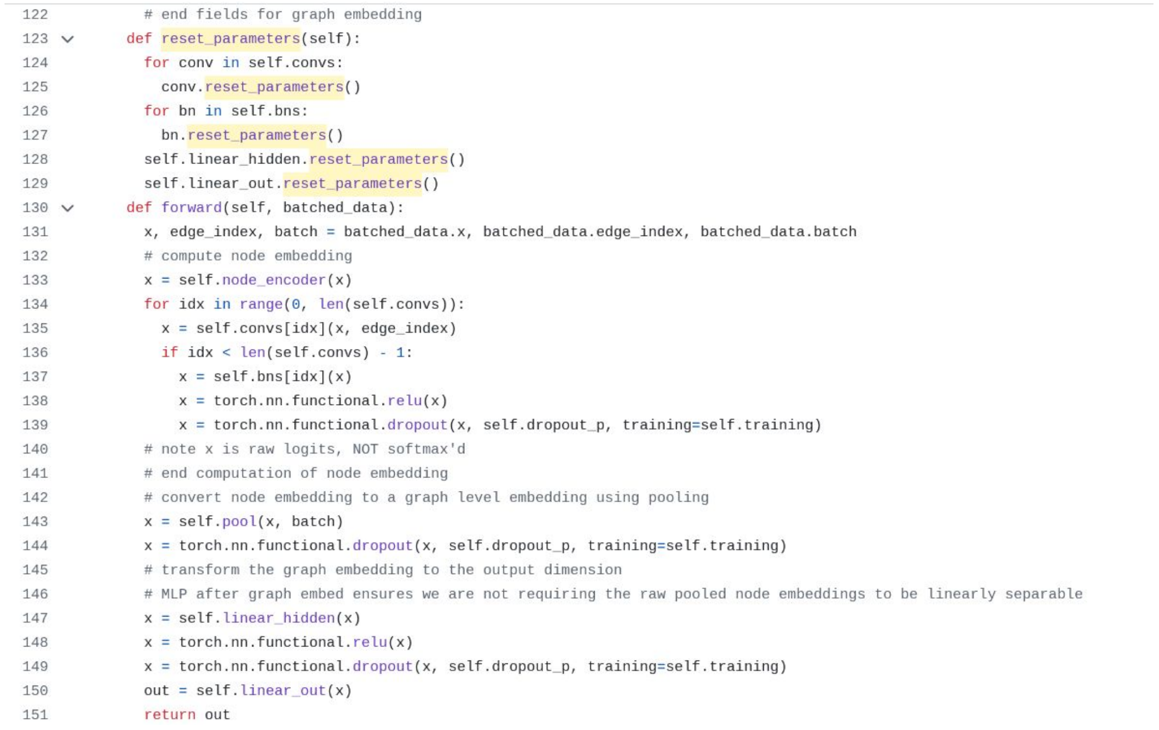
Code | Blame | 167 lines (155 loc) · 6.66 KB

```
def get_atom_feature_dims():
    return list(map(len, [
        allowable_features['possible_atomic_num_list'],
        allowable_features['possible_chirality_list'],
        allowable_features['possible_degree_list'],
        allowable_features['possible_formal_charge_list'],
        allowable_features['possible_numH_list'],
        allowable_features['possible_number_radical_e_list'],
        allowable_features['possible_hybridization_list'],
        allowable_features['possible_is_aromatic_list'],
        allowable_features['possible_is_in_ring_list']
        ]))
```

Approach: Tiny GIN

(32K parameters vs OGB team 1.8M parameter model)

https://github.com/willy-b/tiny-GIN-for-ogbg-molhiv

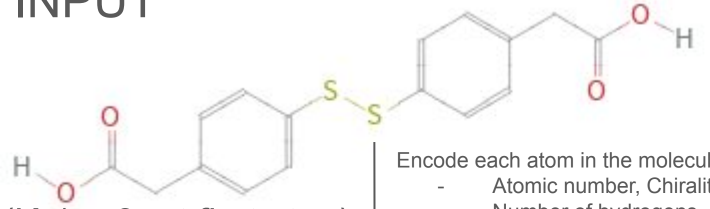


```
    # end fields for graph embedding
  def reset_parameters(self):
    for conv in self.convs:
      conv.reset_parameters()
    for bn in self.bns:
      bn.reset_parameters()
    self.linear_hidden.reset_parameters()
    self.linear_out.reset_parameters()
  def forward(self, batched_data):
    x, edge_index, batch = batched_data.x, batched_data.edge_index, batched_data.batch
    # compute node embedding
    x = self.node_encoder(x)
    for idx in range(0, len(self.convs)):
      x = self.convs[idx](x, edge_index)
      if idx < len(self.convs) - 1:
        x = self.bns[idx](x)
        x = torch.nn.functional.relu(x)
        x = torch.nn.functional.dropout(x, self.dropout_p, training=self.training)
    # note x is raw logits, NOT softmax'd
    # end computation of node embedding
    # convert node embedding to a graph level embedding using pooling
    x = self.pool(x, batch)
    x = torch.nn.functional.dropout(x, self.dropout_p, training=self.training)
    # transform the graph embedding to the output dimension
    # MLP after graph embed ensures we are not requiring the raw pooled node embeddings to be linearly separable
    x = self.linear_hidden(x)
    x = torch.nn.functional.relu(x)
    x = torch.nn.functional.dropout(x, self.dropout_p, training=self.training)
    out = self.linear_out(x)
    return out
```

(continued from last slide)

Tl;dr - We can predict: Use a Graph Neural Network

INPUT



(K-1 = 0, at first step)

Encode each atom in the molecule by 9 features:

- Atomic number, Chirality, Degree, Formal Charge
- Number of hydrogens, Number of radical electrons, Hybridization
- Aromatic, In ring

Multiply 9 features by Linear layer to get node embedding for each atom

OUTPUT

Classification POSITIVE

MLP

G

Sum all atom (node) embeddings to get molecule (graph) embed

$h_v^{(k-1)}$ s

k := k + 1 repeat loop

k < # layers?

yes

no

S,

C

s

Aggregate with encoded neighbors

$$\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}$$

Multi-layer perceptron (MLP)

s

s final atom embedding

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right)$$

(This is a GIN, see Xu et al 2019 in references)

Molecule shown is
https://pubchem.ncbi.nlm.nih.gov/bioassay/179#sid=67143

Approach: Tiny GIN

(32K parameters vs OGB team 1.8M parameter model)

https://github.com/willy-b/tiny-GIN-for-ogbg-molhiv

Hyperparameter values used:

(results in 32,385 model parameters per `sum(p.numel() for p in model.parameters())`, the advised way to count model parameters per https://web.archive.org/web/20240324175343/https://ogb.stanford.edu/docs/leader_overview/)

num_layers: 2 (vs 5 layers in OGB team solution)

hidden_dim: 64

dropout: 0.5

learning_rate: 0.001

epochs: 50

batch_size: 32

weight_decay: 1e-6

per e.g. "Keeping Neural Networks Simple by Minimizing the Description Length of the Weights" (Hinton et al 1993, https://www.cs.toronto.edu/~fritz/absps/colt93.pdf)

add/sum pooling

MLP after node->graph embed pooling

9 atom features used, all edge features ignored

Choice of 2 layers is based on experiment and justified by e.g. GCN GNN layers/hops discussion in Xu et al 2018 "Representation Learning on Graphs with Jumping Knowledge Networks" https://arxiv.org/pdf/1806.03536 . Avoids over-smoothing.

Noting that the depth of network for GNN is not the same as depth of network for non-GNN deep neural networks, as it also controls the number of hops in the graph considered for the embedding of each node; one could also make the network used to compute node embedding based on each hop deeper without changing the number of GNN layers (hops)**.**

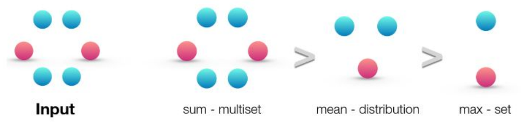
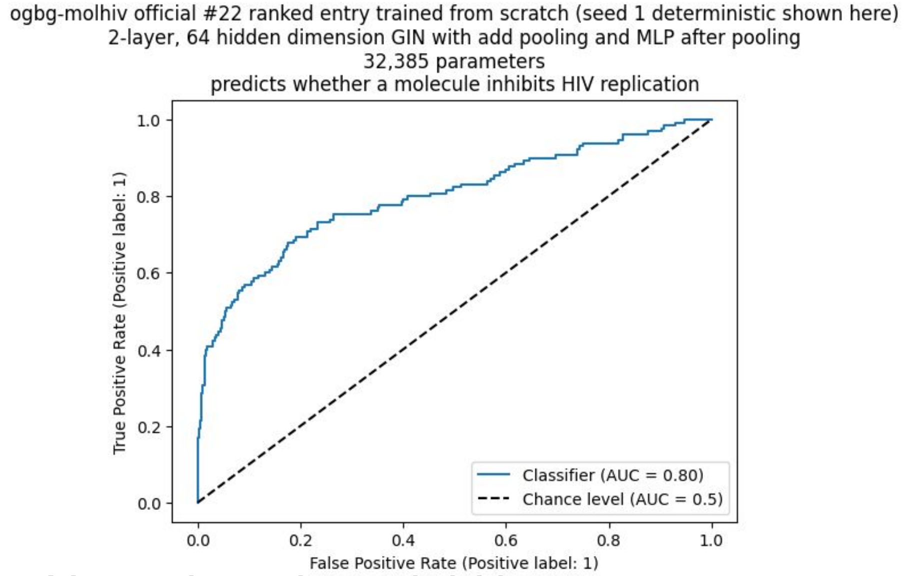
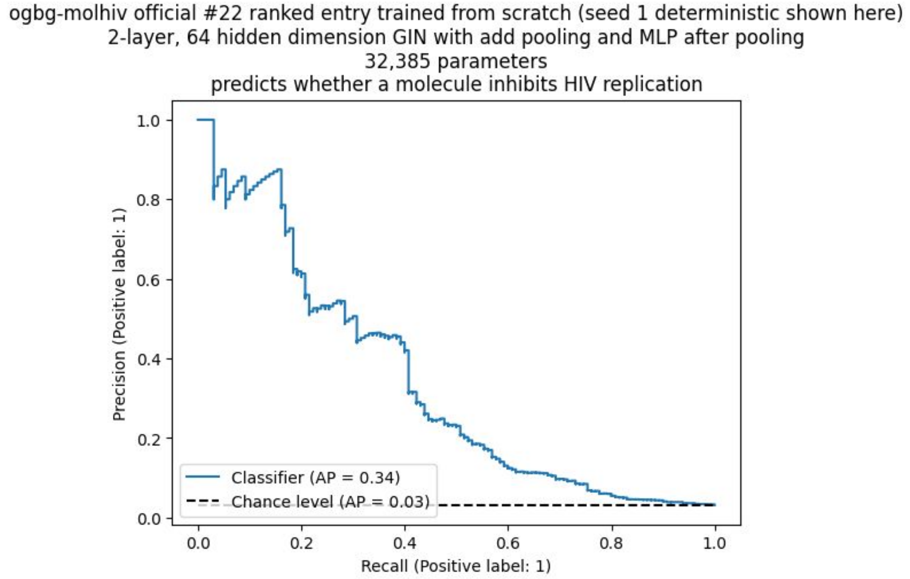


Figure 2: **Ranking by expressive power for sum, mean and max aggregators over a multiset.** Left panel shows the input multiset, *i.e.*, the network neighborhood to be aggregated. The next three panels illustrate the aspects of the multiset a given aggregator is able to capture: sum captures the full multiset, mean captures the proportion/distribution of elements of a given type, and the max aggregator ignores multiplicities (reduces the multiset to a simple set).

Results:
Receiver Operating Characteristic Area Under Curve and Precision-Recall Curve

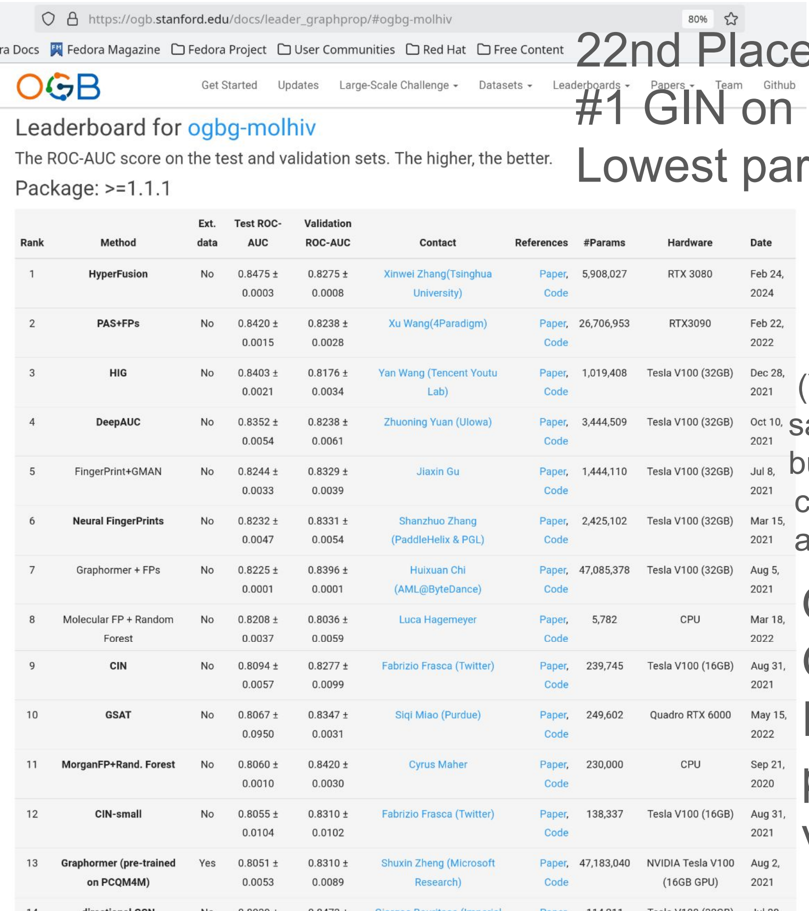




Note, there is variability.
This is seed 1, reported values were over 10 seeds and are similar but slightly worse than this on average, such that mean ROCAUC was 0.7835 +/- 0.0125 (mean +/- sample std, n=10) not 0.80 as shown above in detail.

Results (leaderboard)

22nd Place overall
#1 GIN on leaderboard
Lowest parameter count for a GNN



https://ogb.stanford.edu/docs/leader_graphprop/#ogbg-molhiv 80%

ra Docs · Fedora Magazine · Fedora Project · User Communities · Red Hat · Free Content

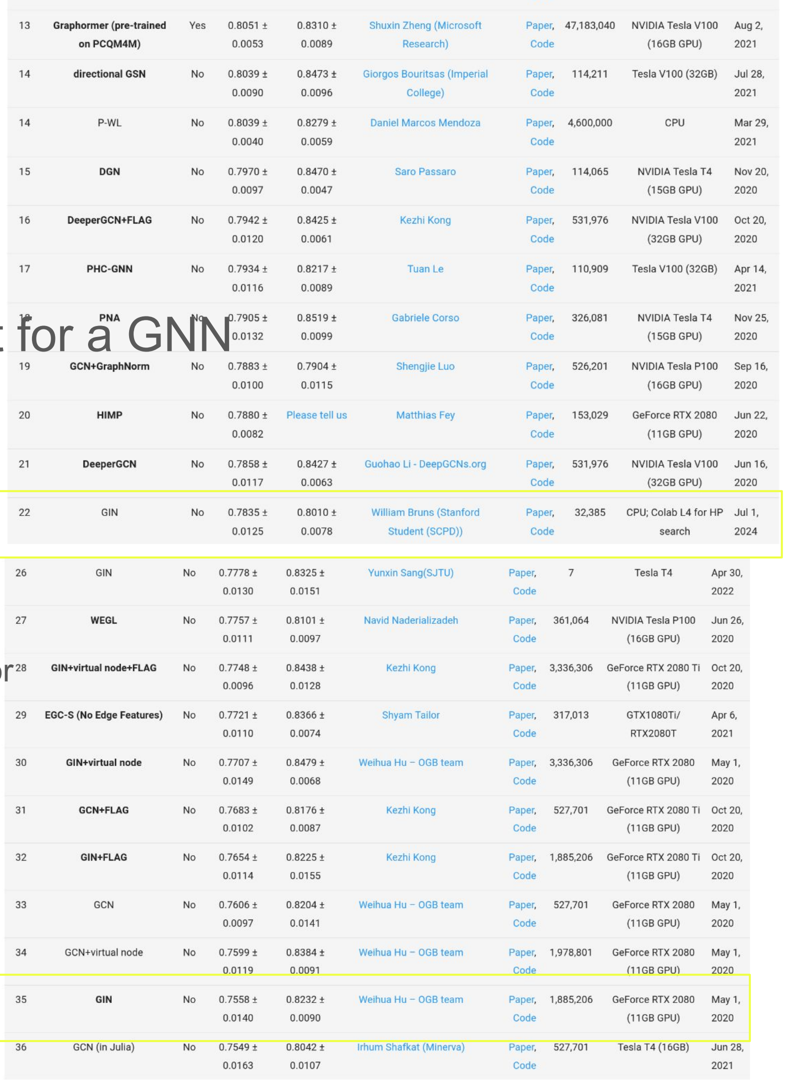
OGB · Get Started · Updates · Large-Scale Challenge · Datasets · Leaderboards · Papers · Team · Github

Leaderboard for ogbg-molhiv

The ROC-AUC score on the test and validation sets. The higher, the better.

Package: >=1.1.1

Rank	Method	Ext. data	Test ROC-AUC	Validation ROC-AUC	Contact	References	#Params	Hardware	Date
1	HyperFusion	No	0.8475 ± 0.0003	0.8275 ± 0.0008	Xinwei Zhang(Tsinghua University)	Paper, Code	5,908,027	RTX 3080	Feb 24, 2024
2	PAS+FPs	No	0.8420 ± 0.0015	0.8238 ± 0.0028	Xu Wang(4Paradigm)	Paper, Code	26,706,953	RTX3090	Feb 22, 2022
3	HIG	No	0.8403 ± 0.0021	0.8176 ± 0.0034	Yan Wang (Tencent Youtu Lab)	Paper, Code	1,019,408	Tesla V100 (32GB)	Dec 28, 2021
4	DeepAUC	No	0.8352 ± 0.0054	0.8238 ± 0.0061	Zhuoning Yuan (UIowa)	Paper, Code	3,444,509	Tesla V100 (32GB)	Oct 10, 2021
5	FingerPrint+GMAN	No	0.8244 ± 0.0033	0.8329 ± 0.0039	Jiaxin Gu	Paper, Code	1,444,110	Tesla V100 (32GB)	Jul 8, 2021
6	Neural FingerPrints	No	0.8232 ± 0.0047	0.8331 ± 0.0054	Shanzhuo Zhang (PaddleHelix & PGL)	Paper, Code	2,425,102	Tesla V100 (32GB)	Mar 15, 2021
7	Graphormer + FPs	No	0.8225 ± 0.0001	0.8396 ± 0.0001	Huixuan Chi (AML@ByteDance)	Paper, Code	47,085,378	Tesla V100 (32GB)	Aug 5, 2021
8	Molecular FP + Random Forest	No	0.8208 ± 0.0037	0.8036 ± 0.0059	Luca Hagemeyer	Paper, Code	5,782	CPU	Mar 18, 2022
9	CIN	No	0.8094 ± 0.0057	0.8277 ± 0.0099	Fabrizio Frasca (Twitter)	Paper, Code	239,745	Tesla V100 (16GB)	Aug 31, 2021
10	GSAT	No	0.8067 ± 0.0950	0.8347 ± 0.0031	Siqi Miao (Purdue)	Paper, Code	249,602	Quadro RTX 6000	May 15, 2022
11	MorganFP+Rand. Forest	No	0.8060 ± 0.0010	0.8420 ± 0.0030	Cyrus Maher	Paper, Code	230,000	CPU	Sep 21, 2020
12	CIN-small	No	0.8055 ± 0.0104	0.8310 ± 0.0102	Fabrizio Frasca (Twitter)	Paper, Code	138,337	Tesla V100 (16GB)	Aug 31, 2021
13	Graphormer (pre-trained on PCQM4M)	Yes	0.8051 ± 0.0053	0.8310 ± 0.0089	Shuxin Zheng (Microsoft Research)	Paper, Code	47,183,040	NVIDIA Tesla V100 (16GB GPU)	Aug 2, 2021



Rank	Method	Ext. data	Test ROC-AUC	Validation ROC-AUC	Contact	References	#Params	Hardware	Date
13	Graphormer (pre-trained on PCQM4M)	Yes	0.8051 ± 0.0053	0.8310 ± 0.0089	Shuxin Zheng (Microsoft Research)	Paper, Code	47,183,040	NVIDIA Tesla V100 (16GB GPU)	Aug 2, 2021
14	directional GSN	No	0.8039 ± 0.0090	0.8473 ± 0.0096	Giorgos Bouritsas (Imperial College)	Paper, Code	114,211	Tesla V100 (32GB)	Jul 28, 2021
14	P-WL	No	0.8039 ± 0.0040	0.8279 ± 0.0059	Daniel Marcos Mendoza	Paper, Code	4,600,000	CPU	Mar 29, 2021
15	DGN	No	0.7970 ± 0.0097	0.8470 ± 0.0047	Saro Passaro	Paper, Code	114,065	NVIDIA Tesla T4 (15GB GPU)	Nov 20, 2020
16	DeeperGCN+FLAG	No	0.7942 ± 0.0120	0.8425 ± 0.0061	Kezhi Kong	Paper, Code	531,976	NVIDIA Tesla V100 (32GB GPU)	Oct 20, 2020
17	PHC-GNN	No	0.7934 ± 0.0116	0.8217 ± 0.0089	Tuan Le	Paper, Code	110,909	Tesla V100 (32GB)	Apr 14, 2021
18	PNA	No	0.7905 ± 0.0132	0.8519 ± 0.0099	Gabriele Corso	Paper, Code	326,081	NVIDIA Tesla T4 (15GB GPU)	Nov 25, 2020
19	GCN+GraphNorm	No	0.7883 ± 0.0100	0.7904 ± 0.0115	Shengjie Luo	Paper, Code	526,201	NVIDIA Tesla P100 (16GB GPU)	Sep 16, 2020
20	HIMP	No	0.7880 ± 0.0082	Please tell us	Matthias Fey	Paper, Code	153,029	GeForce RTX 2080 (11GB GPU)	Jun 22, 2020
21	DeeperGCN	No	0.7858 ± 0.0117	0.8427 ± 0.0063	Guohao Li - DeepGCNs.org	Paper, Code	531,976	NVIDIA Tesla V100 (32GB GPU)	Jun 16, 2020
22	GIN	No	0.7835 ± 0.0125	0.8010 ± 0.0078	William Bruns (Stanford Student (SCPD))	Paper, Code	32,385	CPU; Colab L4 for HP search	Jul 1, 2024
26	GIN	No	0.7778 ± 0.0130	0.8325 ± 0.0151	Yunxin Sang(SJTU)	Paper, Code	7	Tesla T4	Apr 30, 2022
27	WEGL	No	0.7757 ± 0.0111	0.8101 ± 0.0097	Navid Naderializadeh	Paper, Code	361,064	NVIDIA Tesla P100 (16GB GPU)	Jun 26, 2020
28	GIN+virtual node+FLAG	No	0.7748 ± 0.0096	0.8438 ± 0.0128	Kezhi Kong	Paper, Code	3,336,306	GeForce RTX 2080 Ti (11GB GPU)	Oct 20, 2020
29	EGC-S (No Edge Features)	No	0.7721 ± 0.0110	0.8366 ± 0.0074	Shyam Tailor	Paper, Code	317,013	GTX1080Ti/ RTX2080T	Apr 6, 2021
30	GIN+virtual node	No	0.7707 ± 0.0149	0.8479 ± 0.0068	Weihua Hu – OGB team	Paper, Code	3,336,306	GeForce RTX 2080 (11GB GPU)	May 1, 2020
31	GCN+FLAG	No	0.7683 ± 0.0102	0.8176 ± 0.0087	Kezhi Kong	Paper, Code	527,701	GeForce RTX 2080 Ti (11GB GPU)	Oct 20, 2020
32	GIN+FLAG	No	0.7654 ± 0.0114	0.8225 ± 0.0155	Kezhi Kong	Paper, Code	1,885,206	GeForce RTX 2080 Ti (11GB GPU)	Oct 20, 2020
33	GCN	No	0.7606 ± 0.0097	0.8204 ± 0.0141	Weihua Hu – OGB team	Paper, Code	527,701	GeForce RTX 2080 (11GB GPU)	May 1, 2020
34	GCN+virtual node	No	0.7599 ± 0.0119	0.8384 ± 0.0091	Weihua Hu – OGB team	Paper, Code	1,978,801	GeForce RTX 2080 (11GB GPU)	May 1, 2020
35	GIN	No	0.7558 ± 0.0140	0.8232 ± 0.0090	Weihua Hu – OGB team	Paper, Code	1,885,206	GeForce RTX 2080 (11GB GPU)	May 1, 2020
36	GCN (in Julia)	No	0.7549 ± 0.0163	0.8042 ± 0.0107	Irhum Shafkat (Minerva)	Paper, Code	527,701	Tesla T4 (16GB)	Jun 28, 2021

(Yunxin Sang's says 7 parameters but is >50K confirmed with author and reported)

OGB team GIN Is 1.8M parameters vs our 32K

Future directions

In progress: OGBG molpcba 128-multitask challenge.

But what I'm excited about:

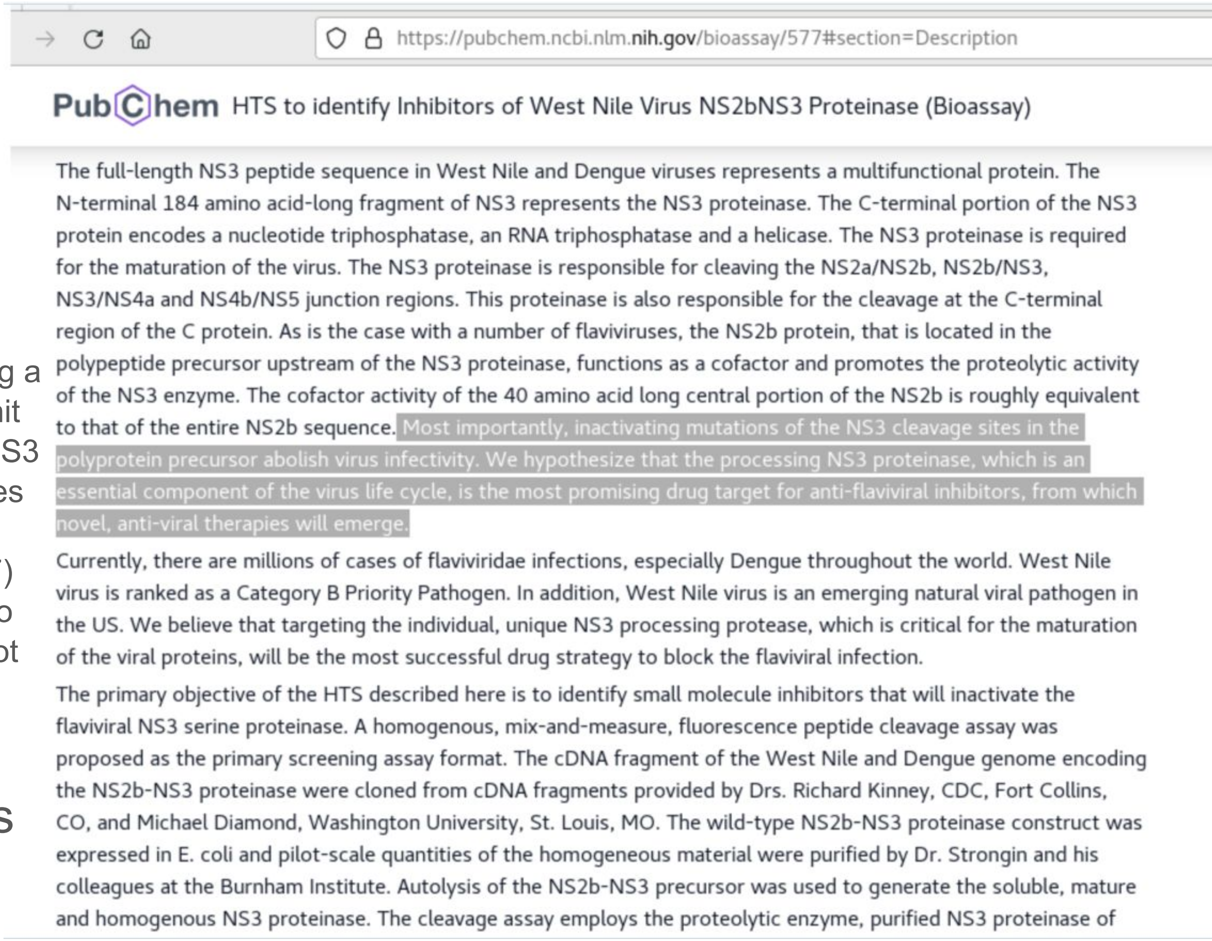
West Nile Virus doesn't have any approved antivirals! (unlike HIV which has many)

Could we speed up antiviral discovery by training a graph neural network to predict molecules that hit targets expected to inhibit the virus (e.g. NS2bNS3 proteinase) and then screen millions of molecules in e.g. the ZINC database for candidates?

I converted some PCBA data available (AID 577) into OGB format and started testing (not ready to release any results yet but gets some traction not SO dissimilar to say ogbg-molhiv benchmark).

If you are interested in collaborating on these problems please contact me at adde.animulis@gmail.com

or https://github.com/willy-b



https://pubchem.ncbi.nlm.nih.gov/bioassay/577#section=Description

PubChem HTS to identify Inhibitors of West Nile Virus NS2bNS3 Proteinase (Bioassay)

The full-length NS3 peptide sequence in West Nile and Dengue viruses represents a multifunctional protein. The N-terminal 184 amino acid-long fragment of NS3 represents the NS3 proteinase. The C-terminal portion of the NS3 protein encodes a nucleotide triphosphatase, an RNA triphosphatase and a helicase. The NS3 proteinase is required for the maturation of the virus. The NS3 proteinase is responsible for cleaving the NS2a/NS2b, NS2b/NS3, NS3/NS4a and NS4b/NS5 junction regions. This proteinase is also responsible for the cleavage at the C-terminal region of the C protein. As is the case with a number of flaviviruses, the NS2b protein, that is located in the polypeptide precursor upstream of the NS3 proteinase, functions as a cofactor and promotes the proteolytic activity of the NS3 enzyme. The cofactor activity of the 40 amino acid long central portion of the NS2b is roughly equivalent to that of the entire NS2b sequence. Most importantly, inactivating mutations of the NS3 cleavage sites in the polyprotein precursor abolish virus infectivity. We hypothesize that the processing NS3 proteinase, which is an essential component of the virus life cycle, is the most promising drug target for anti-flaviviral inhibitors, from which novel, anti-viral therapies will emerge.

Currently, there are millions of cases of flaviviridae infections, especially Dengue throughout the world. West Nile virus is ranked as a Category B Priority Pathogen. In addition, West Nile virus is an emerging natural viral pathogen in the US. We believe that targeting the individual, unique NS3 processing protease, which is critical for the maturation of the viral proteins, will be the most successful drug strategy to block the flaviviral infection.

The primary objective of the HTS described here is to identify small molecule inhibitors that will inactivate the flaviviral NS3 serine proteinase. A homogenous, mix-and-measure, fluorescence peptide cleavage assay was proposed as the primary screening assay format. The cDNA fragment of the West Nile and Dengue genome encoding the NS2b-NS3 proteinase were cloned from cDNA fragments provided by Drs. Richard Kinney, CDC, Fort Collins, CO, and Michael Diamond, Washington University, St. Louis, MO. The wild-type NS2b-NS3 proteinase construct was expressed in E. coli and pilot-scale quantities of the homogeneous material were purified by Dr. Strongin and his colleagues at the Burnham Institute. Autolysis of the NS2b-NS3 precursor was used to generate the soluble, mature and homogenous NS3 proteinase. The cleavage assay employs the proteolytic enzyme, purified NS3 proteinase of

References

Hu, Weihua and Fey, Matthias and Zitnik, Marinka and Dong, Yuxiao and Ren, Hongyu and Liu, Bowen and Catasta, Michele and Leskovec, Jure. Open Graph Benchmark: Datasets for Machine Learning on Graphs. arXiv preprint arXiv:2005.00687, 2020.

Wu, Zhenqin and Ramsundar, Bharath and Feinberg, Evan N and Gomes, Joseph and Geniesse, Caleb and SPappu, Aneesh and Leswing, Karl and Pande, Vijay. Moleculenet: a benchmark for molecular machine learning. Chemical Science, 9(2):513–530, 2018.

Fey, Matthias and Lenssen, Jan E. Fast Graph Representation Learning with PyTorch Geometric. ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019. (Graph Isomorphism Network (GIN) implementation used)

Xu, Keyulu and Hu, Weihua and Leskovec, Jure and Jegelka, Stefanie. How Powerful Are Graph Neural Networks? International Conference on Learning Representations, 2019. https://openreview.net/forum?id=ryGs6iA5Km , https://arxiv.org/pdf/1810.00826 . (Graph Isomorphism Network (GIN) original paper)