Graph Neural Networks
Architectures, Fundamental Properties and Applications

Navid NaderiAlizadeh, Alejandro Ribeiro, Luana Ruiz, Zhiyang Wang

Web: gnn.seas.upenn.edu/aaai-2025/

Feb 26 2025

Graph Neural Network Applications

Navid NaderiAlizadeh

Dept. of Biostatistics & Bioinformatics
Duke University

navid.naderi@duke.edu
sites.duke.edu/navid

TH15: Graph Neural Networks: Architectures, Fundamental Properties and Applications
gnn.seas.upenn.edu/aaai-2025/

Feb 26 2025

Applications of GNNs

- GNNs enable scalable machine learning on graph-structured data in a variety of systems.
  - ⇒ Learning ratings in recommendation systems
  - ⇒ Resource allocation in communication systems
  - ⇒ Federated learning in distributed systems
  - ⇒ Protein property prediction in biological systems

Learning Ratings in Recommendation Systems

- Formulate recommendation systems as ML problems that predict ratings that users give to items

- In a recommendation system, we want to predict the rating a user would give to an item
- Collect ratings that some users give to some items $\Rightarrow$ These are rating histories
- Exploit product similarities to predict ratings of unseen user-item pairs
- Example 1 $\Rightarrow$ In an online store items are products and users are customers
- Example 2 $\Rightarrow$ In a movie repository items are movies and users are watchers

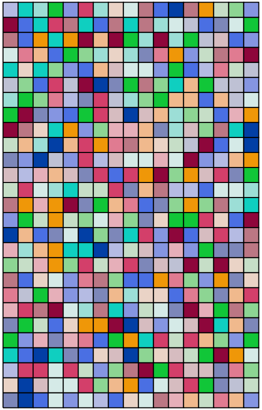
- For all items i and users u there exist ratings $\Rightarrow y_{ui}$

  $\Rightarrow$ User rating vector $\mathbf{y}_u$ has entries y_{ui}

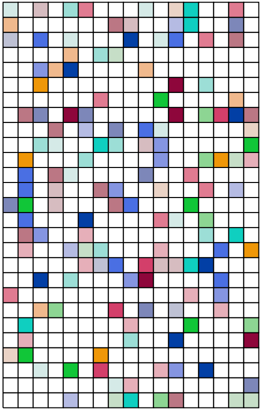
- We only observe a subset of ratings $\Rightarrow x_{ui}$

  $\Rightarrow$ Observed user rating vector $\mathbf{x}_u$ has entries x_{ui}

  $\Rightarrow$ We assume $x_{ui} = 0$ if item i is unrated by user u



- For all items i and users u there exist ratings $\Rightarrow y_{ui}$
  - $\Rightarrow$ User rating vector $\mathbf{y}_u$ has entries y_{ui}
- We only observe a subset of ratings $\Rightarrow x_{ui}$
  - $\Rightarrow$ Observed user rating vector $\mathbf{x}_u$ has entries x_{ui}
  - $\Rightarrow$ We assume $x_{ui} = 0$ if item i is unrated by user u



- Construct product similarity graph with weights w_{ij} represent likelihood of similar scores
- Interpret vector of ratings $\mathbf{y}_u$ of user u as a graph signal supported on the product similarity graph
- The observed ratings $\mathbf{x}_u$ of user u are a subsampling of this graph signal.
- Our goal is to learn to reconstruct the rating graph signal $\mathbf{y}_u$ from the observed ratings $\mathbf{x}_u$
- Build similarity graph using available ratings. Use of expert knowledge is common as well

- Consider pair of products i and j. Restrict attention to set of users that rated both products $\Rightarrow \mathcal{U}_{ij}$

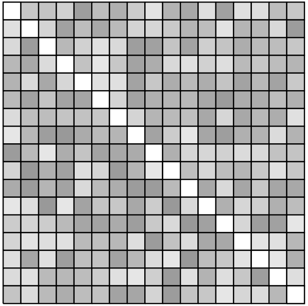
- Mean ratings restricted to users that rated products i and j

$$\mu_{ij} = \frac{1}{\#(\mathcal{U}_{ij})} \sum_{u \in \mathcal{U}_{ij}} x_{ui} \qquad \mu_{ji} = \frac{1}{\#(\mathcal{U}_{ij})} \sum_{u \in \mathcal{U}_{ji}} x_{uj}$$

- Similarity score = correlation restricted to users in $\mathcal{U}_{ij}$

$$\sigma_{ij} = \frac{1}{\#(\mathcal{U}_{ij})} \sum_{u \in \mathcal{U}_{ij}} \Big(x_{ui} - \mu_{ij} \Big) \Big(x_{uj} - \mu_{ji} \Big)$$

- Weights = normalized correlations $\Rightarrow w_{ij} = \sigma_{ij} \Big/ \sqrt{\sigma_{ii}\sigma_{jj}}$



- Given observed ratings $\mathbf{x}_u$ the AI produces estimates $\Phi(\mathbf{x}_u)$. We want $\Phi(\mathbf{x}_u)$ to approximate $\mathbf{y}_u$

$$\ell\Big(\mathbf{y}_u, \Phi(\mathbf{x}_u)\Big) = \frac{1}{2}\Big\|\, \mathbf{y}_u - \Phi(\mathbf{x}_u)\,\Big\|^2$$

- In reality, we want to predict the rating of specific item i

$$\ell\Big(\mathbf{y}_u, \Phi(\mathbf{x}_u)\Big) = \frac{1}{2}\Big(\mathbf{e}_i^T\mathbf{y}_u - \mathbf{e}_i^T\Phi(\mathbf{x}_u)\Big)^2$$

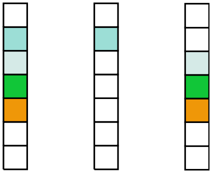
- Where $\mathbf{e}_i$ is a vector in the canonical basis $\Rightarrow (\mathbf{e}_i)_i = 1$, $(\mathbf{e}_i)_j = 0$ for $j \neq i$

Training Set

- For each item i let $\mathcal{U}_i$ be the set of users that have rated i. Construct training pairs $(\mathbf{x}, \mathbf{y})$ with

$$\mathbf{y} = \left(\mathbf{e}_i^T \mathbf{x}_u\right)\mathbf{e}_i \qquad \mathbf{x} = \mathbf{x}_u - \mathbf{y} \qquad \text{for all } u \in \mathcal{U}_i, \text{ for all } i$$

- Extract the rating x_{ui} of item i. Record into graph signal y
- Remove rating x_{ui} from $\mathbf{x}_u$. Record to graph signal $\mathbf{x}$
- Repeat for all users in the set $\mathcal{U}_i$ of users that rated i
- Repeat for all items $\Rightarrow$ Training set $\mathcal{T}$



- Parameterized AI $\Phi(\mathbf{x}_u) = \Phi(\mathbf{x}_u; \mathcal{H})$. We want to find solution of the supervised learning problem

$$\mathcal{H}^* = \underset{\mathcal{H}}{\text{argmin}} \sum_{(x,y)\in\mathcal{T}} \Big(\mathbf{e}_i^T \mathbf{y} - \mathbf{e}_i^T \Phi(\mathbf{x}; \mathcal{H}) \Big)^2$$

- Two bad ideas $\Rightarrow$ Linear regression. Fully connected neural networks

- Two good ideas $\Rightarrow$ Graph filters. Graph neural networks

Learning Ratings with Graph Filters and GNNs

- We use graph filters and graph neural networks to learn ratings in recommendation systems
- We contrast with the use of linear regression and fully connected neural networks

- Use MovieLens-100k as benchmark $\Rightarrow 10^6$ ratings given by $U = 943$ users to $M = 1,682$ movies

- The ratings for each movie are between 1 and 5. From one star to five starts

- Train and test several machine learning parametrizations.

Empirical Risk Minimization

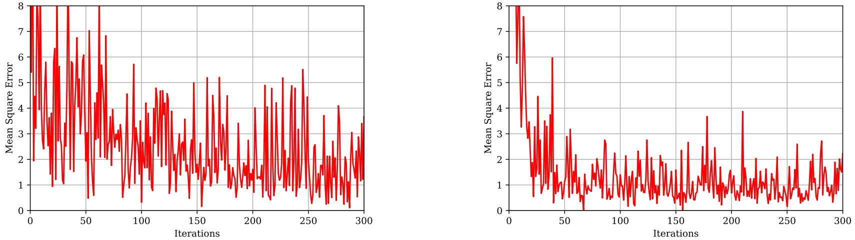
- We predict ratings using AI that results from solving the ERM problem

$$\mathcal{H}^* = \underset{\mathcal{H}}{\operatorname{argmin}} \sum_{(x,y)\in\mathcal{T}} \left(\mathbf{e}_i^T \mathbf{y} - \mathbf{e}_i^T \Phi(\mathbf{x}; \mathcal{H}) \right)^2$$

- Parameterizations that ignore data structure $\Longrightarrow$ Linear regression. Fully connected NNs

- Parameterizations that leverage data structure $\Longrightarrow$ Graph filters. Graph NNs

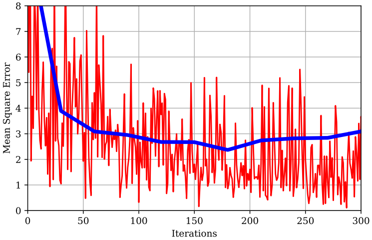
- Linear regression reduces training MSE to about 2. Quite bad for ratings that vary from 0 to 5
- Graph filter reduces training MSE to about 1. Not too good. Humans are not that predictable

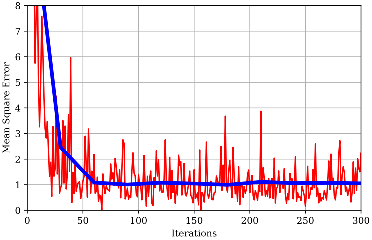


- Graph filter outperforms linear regression $\Rightarrow$ Leverages underlying permutation symmetries

- Linear regression works even worse in the test set

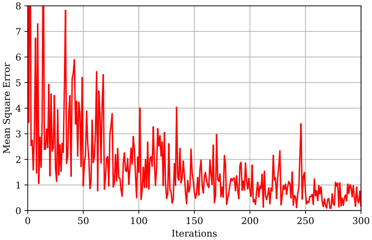
- The test MSE of the graph filter is about the same as the training MSE. It generalizes

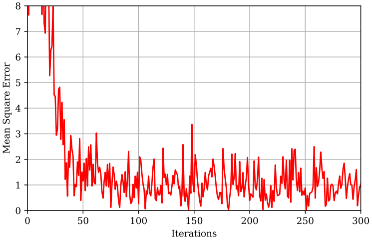




- Graph filter outperforms linear regression $\Rightarrow$ Leverages underlying permutation symmetries

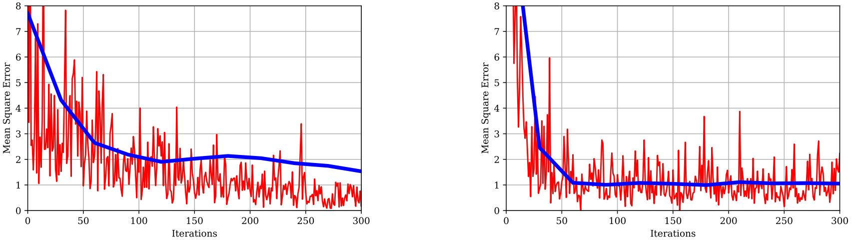
- The fully connected NN reduces the MSE to about 0.8. Looks like a great accomplishment.
- Graph NN reduces test MSE to about 0.9. Not bad. But not as good as the fully connected NN





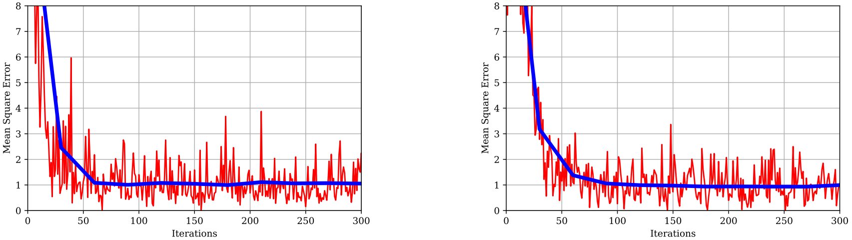
- Graph NN outperforms fully connected NN $\Rightarrow$ Leverages underlying permutation symmetries

- But the fully connected NN does not do well in the test set. It does not generalize
- The test MSE of the graph NN is about the same as the training MSE. It generalizes



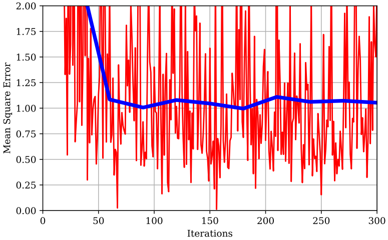
- Graph NN outperforms fully connected NN $\Rightarrow$ Leverages underlying permutation symmetries

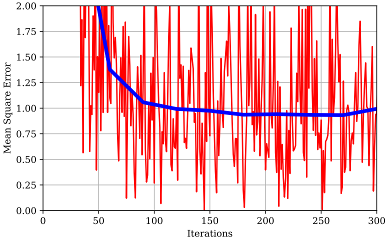
- The graph filter and the GNN do well in the training and test set. They generalize well
- The GNN does a little better. Not by much. But an extra 10% is not irrelevant



- GNN outperforms graph filter $\Rightarrow$ The GNN has a better stability-discriminability tradeoff

- The graph filter and the GNN do well in the training and test set. They generalize well
- The GNN does a little better. Not by much. But an extra 10% is not irrelevant

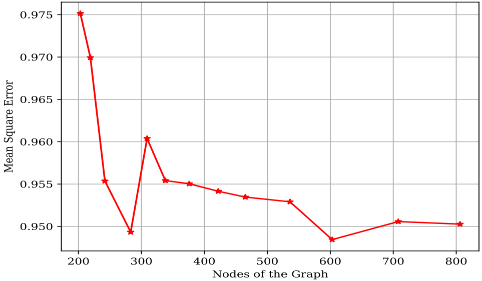




- GNN outperforms graph filter $\Rightarrow$ The GNN has a better stability-discriminability tradeoff

- A GNN can be trained on a graph with a small number of nodes ...

  $\Rightarrow$ And transferred to a graph with a (much) larger number of nodes. Without retraining



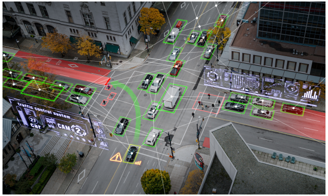
- In this recommendation system, transference incurs no MSE degradation $\Rightarrow$ MSE is further reduced

Wireless Resource Management with GNNs

- GNNs can enable scalable resource management in autonomous wireless communication networks.

Towards Next-Generation Wireless Networks

- Wireless networks are growing beyond humans' ability to design and manage them → 5G, WiFi 6

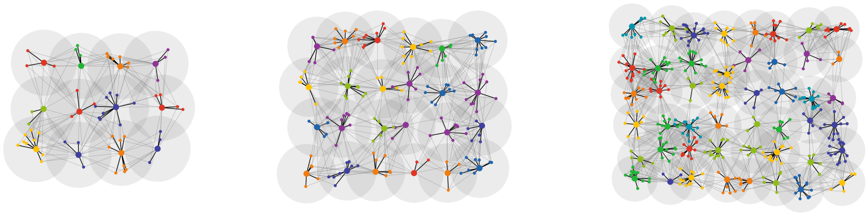


- To address increasing complexity of wireless networks, we will make them autonomous → 6G, WiFi 7

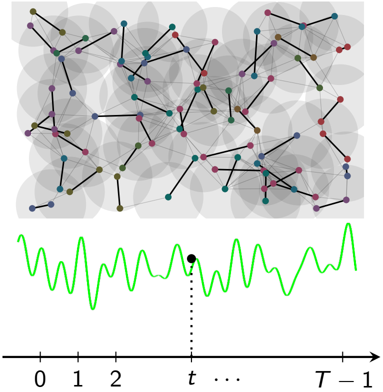
⇒ An autonomous wireless network makes (at least some) decisions without human intervention.

- Making operational decisions in wireless networks entails solving large-scale constrained optimization problems.

- Solving these problems is very challenging, leading to the design and use of heuristic methods.

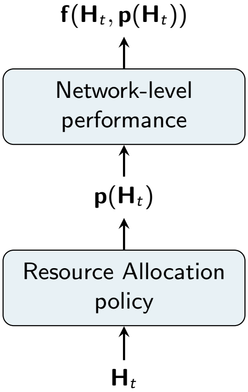


- We can leverage data to learn better autonomous network management policies using machine learning.



$$
\max_{\{\mathbf{p}(\mathbf{H}_t)\}_{t=0}^{T-1}} \quad \mathcal{U}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}(\mathbf{H}_t,\mathbf{p}(\mathbf{H}_t))\right)
$$

$$
\text{s.t.} \quad \mathbf{g}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}(\mathbf{H}_t,\mathbf{p}(\mathbf{H}_t))\right) \geq \mathbf{0}
$$



- In this classical formulation, resource allocation decisions must be recalculated for any given network state $\mathbf{H}$.
  - $\Rightarrow$ This makes learning and deploying such a policy infeasible in practice.
- We parameterize the resource allocation policy, replacing $\mathbf{p}(\mathbf{H})$ with $\mathbf{p}(\mathbf{H};\boldsymbol{\theta})$.
- The advantage of parameterization is that we do not need to solve the problem online to find the decisions.

Unparameterized Formulation

$$\max_{\{\mathbf{p}(\mathbf{H}_t)\}_{t=0}^{T-1}} \ \mathcal{U}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}(\mathbf{H}_t,\mathbf{p}(\mathbf{H}_t))\right)$$
$$\text{s.t.} \quad \mathbf{g}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}(\mathbf{H}_t,\mathbf{p}(\mathbf{H}_t))\right)\geq \mathbf{0}$$

$\longrightarrow$

Parameterized Formulation

$$P^\star = \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \ \mathcal{U}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}(\mathbf{H}_t,\mathbf{p}(\mathbf{H}_t;\boldsymbol{\theta}))\right)$$
$$\text{s.t.} \quad \mathbf{g}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}(\mathbf{H}_t,\mathbf{p}(\mathbf{H}_t;\boldsymbol{\theta}))\right)\geq \mathbf{0}$$

Empirical Risk Minimization

$$\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \quad -\frac{1}{N} \sum_{i=0}^{N-1} \ell\left(\psi\left(\mathbf{x}_i; \boldsymbol{\theta}\right)\right)$$

Parameterized Resource Allocation

$$\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \quad \mathcal{U}\left(\frac{1}{T} \sum_{t=0}^{T-1} \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t; \boldsymbol{\theta}))\right)$$

$$\text{s.t.} \quad \mathbf{g}\left(\frac{1}{T} \sum_{t=0}^{T-1} \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t; \boldsymbol{\theta}))\right) \geq \mathbf{0}$$

- Sequential decision making over a time series sequence $\{\mathbf{H}_t\}_{t=0}^{T-1}$ without access to ground-truth labels.
- Inclusion of the constraints makes this problem fundamentally different from a regular learning problem.

- We move to the Lagrangian dual domain, and associate a set of non-negative dual variables $\boldsymbol{\mu}$ to the constraints.
- The Lagrangian function can then be written as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\mu}) = \mathcal{U}\left(\frac{1}{T}\sum_{t=0}^{T-1} \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t; \boldsymbol{\theta}))\right) + \boldsymbol{\mu}^T \mathbf{g}\left(\frac{1}{T}\sum_{t=0}^{T-1} \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t; \boldsymbol{\theta}))\right).$$

- We then seek to maximize the Lagrangian over $\boldsymbol{\theta}$, while minimizing it over $\boldsymbol{\mu}$, i.e.,

$$D^\star = \min_{\boldsymbol{\mu} \geq \mathbf{0}} \max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\mu}).$$

Iterative Unsupervised Primal-Dual Updates

- The primal model parameters $\boldsymbol{\theta}$ and the dual variables $\boldsymbol{\mu}$ can be iteratively updated using a primal-dual method.

- We define an iteration duration T_0 between consecutive updates, and an iteration index k.

$$\boldsymbol{\theta}_k = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left[\mathcal{U}\left(\frac{1}{T_0} \sum_{t=kT_0}^{(k+1)T_0-1} \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t; \boldsymbol{\theta})) \right) + \boldsymbol{\mu}_k^T \mathbf{g}\left(\frac{1}{T_0} \sum_{t=kT_0}^{(k+1)T_0-1} \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t; \boldsymbol{\theta})) \right) \right]$$

$k \leftarrow k+1$

$$\boldsymbol{\mu}_{k+1} = \left[\boldsymbol{\mu}_k - \eta_{\boldsymbol{\mu}} \mathbf{g}\left(\frac{1}{T_0} \sum_{t=kT_0}^{(k+1)T_0-1} \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t; \boldsymbol{\theta}_k)) \right) \right]_+$$

- Constraint slacks are the gradient or a subgradient of the Lagrangian with respect to the dual variables.

Theorem (NaderiAlizadeh-Eisen-Ribeiro)

The sequence of decisions made by the primal-dual updates is both feasible, i.e.,

$$\lim_{T\to\infty} \mathbf{g}\left(\frac{1}{T}\sum_{t=0}^{T-1} \mathbf{f}\left(\mathbf{H}_t, \mathbf{p}\left(\mathbf{H}_t; \boldsymbol{\theta}_{\lfloor t/T_0 \rfloor}\right)\right)\right) \geq \mathbf{0}, \quad a.s.$$

and near-optimal, i.e.,

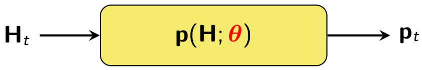
$$\lim_{T\to\infty} \mathbb{E}\left[\mathcal{U}\left(\frac{1}{T}\sum_{t=0}^{T-1} \mathbf{f}\left(\mathbf{H}_t, \mathbf{p}\left(\mathbf{H}_t; \boldsymbol{\theta}_{\lfloor t/T_0 \rfloor}\right)\right)\right)\right] \geq P^\star - \frac{c\eta_\mu G^2}{2}.$$

- c denotes the number of constraints, η_μ denotes the dual step size, G upper-bounds the constraint magnitudes.
- There are no restrictions on the convexity of $\mathbf{f}$ and the parameterization $\mathbf{p}(\cdot; \boldsymbol{\theta})$.
- Issue: Training cannot be stopped at a finite iteration!

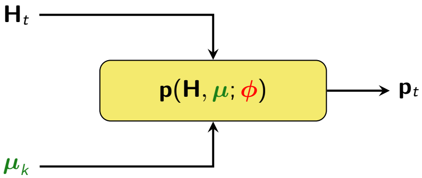
NaderiAlizadeh-Eisen-Ribeiro, *State-Augmented Learnable Algorithms for Resource Management in Wireless Networks*, IEEE TSP, arxiv.org/abs/2207.02242

- We propose to use both network state $\mathbf{H}$ and dual variables $\boldsymbol{\mu}$ as input to the resource allocation policy.
- We leverage a revised state-augmented parameterization $\mathbf{p}(\mathbf{H}, \boldsymbol{\mu}; \boldsymbol{\phi})$ to replace $\mathbf{p}(\mathbf{H}; \boldsymbol{\theta})$.

Regular Parameterization



State-Augmented Parameterization



- The revised parameterization leads to the augmented Lagrangian

$$\mathcal{L}_{\mu}(\phi) = \mathcal{U}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \mu; \phi))\right) + \mu^T \mathbf{g}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \mu; \phi))\right).$$

- The optimal state-augmented policy parameters are found during training as

$$\phi^{\star} = \arg\max_{\phi \in \mathbf{\Phi}} \mathbb{E}_{\mu}\left[\mathcal{L}_{\mu}(\phi)\right].$$

- This resolves the challenge of re-optimizing the model parameters for any given set of dual variables.
- The dual variables are updated during execution as

$$\mu_{k+1} = \left[\mu_k - \eta_{\mu}\, \underbrace{\mathbf{g}\left(\frac{1}{T_0}\sum_{t=kT_0}^{(k+1)T_0-1}\mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t, \mu_k; \phi^{\star}))\right)}_{\text{Constraint satisfaction over the } k^{\text{th}} \text{ iteration}}\right]_{+}.$$

Theorem (NaderiAlizadeh-Eisen-Ribeiro)

The sequence of decisions made by the proposed state-augmented algorithm is both feasible, i.e.,

$$\lim_{T\to\infty} \mathbf{g}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}\left(\mathbf{H}_t, \mathbf{p}\left(\mathbf{H}_t, \boldsymbol{\mu}_{\lfloor t/T_0\rfloor}; \boldsymbol{\phi}^\star\right)\right)\right) \geq \mathbf{0}, \quad a.s.$$

and near-optimal, i.e.,

$$\lim_{T\to\infty} \mathbb{E}\left[\mathcal{U}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{f}\left(\mathbf{H}_t, \mathbf{p}\left(\mathbf{H}_t, \boldsymbol{\mu}_{\lfloor t/T_0\rfloor}; \boldsymbol{\phi}^\star\right)\right)\right)\right] \geq P^\star - \frac{c\eta_\mu G^2}{2} - M\epsilon.$$

- ϵ-universal parameterization $\mathbf{p}(\mathbf{H}, \boldsymbol{\mu}; \boldsymbol{\phi})$: For any $\mathbf{H}$ and $\boldsymbol{\theta}(\cdot)$, there exists $\boldsymbol{\phi}$ s.t.

$$\mathbb{E}\left\|\mathbf{p}(\mathbf{H}, \boldsymbol{\mu}; \boldsymbol{\phi}) - \mathbf{p}(\mathbf{H}; \boldsymbol{\theta}(\boldsymbol{\mu}))\right\|_\infty \leq \epsilon.$$

- M-Lipschitz continuity of $\mathbf{f}$: For any $\mathbf{H}$, $\mathbf{p}_1$ and $\mathbf{p}_2$, $\mathbb{E}\left\|\mathbf{f}(\mathbf{H}, \mathbf{p}_1) - \mathbf{f}(\mathbf{H}, \mathbf{p}_2)\right\|_\infty \leq M\mathbb{E}\left\|\mathbf{p}_1 - \mathbf{p}_2\right\|_\infty$.
- The decisions made by our method are close to those made by the original primal-dual iterations.

NaderiAlizadeh-Eisen-Ribeiro, *State-Augmented Learnable Algorithms for Resource Management in Wireless Networks*, IEEE TSP, arxiv.org/abs/2207.02242

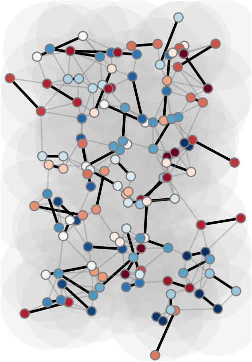
- We focus on multi-user interference channels with m transmitter-receiver pairs.
- The performance function for the i^{th} receiver represents its Shannon capacity,

$$f_i(\mathbf{H}_t, \mathbf{p}) = \log_2\left(1 + \frac{p_i \left|h_{ii,t}\right|^2}{\frac{N}{P_{\max}} + \sum_{j=1,j\neq i}^{m} p_j \left|h_{ji,t}\right|^2}\right).$$

- Considering a sum-rate utility and minimum-rate constraints leads to

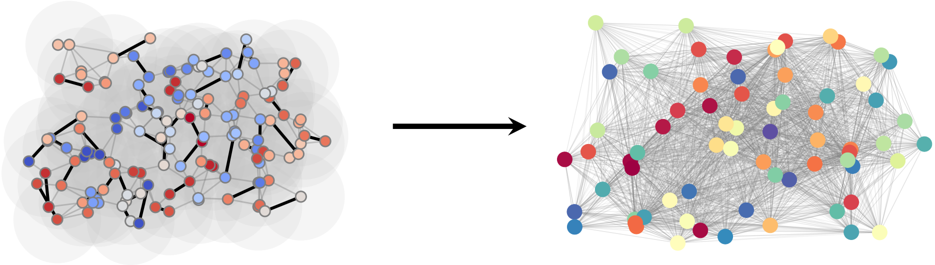
$$\max_{\{\mathbf{p}(\mathbf{H}_t)\}_{t=0}^{T-1}} \quad \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{m} f_i(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t)),$$

$$\text{s.t.} \quad \frac{1}{T}\sum_{t=0}^{T-1} \mathbf{f}(\mathbf{H}_t, \mathbf{p}(\mathbf{H}_t)) \geq f_{\min}\mathbf{1}_m.$$

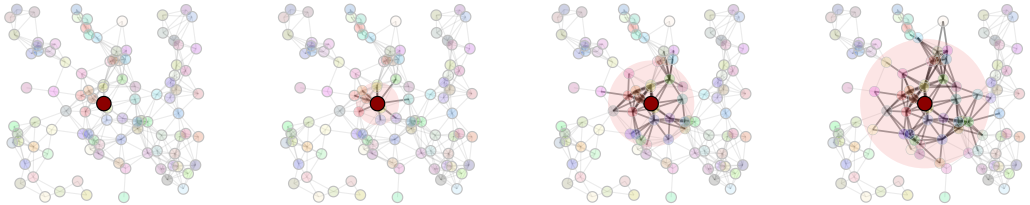


Modeling Interference Channels as Graphs

- We model the interference channel at each time step t as a graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}, \mathbf{Y}_t, w_t)$.
  - $\Rightarrow$ $\mathcal{V} = \{1, 2, \ldots, m\}$ denotes the set of transceiver nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges.
  - $\Rightarrow$ $\mathbf{Y}_t \in \mathbb{R}^{m \times 1}$ denotes the initial node features, which we set to the dual variables: $\mathbf{Y}_t = \boldsymbol{\mu}_{\lfloor t/T_0 \rfloor}$.
  - $\Rightarrow$ $w_t : \mathcal{E} \to \mathbb{R}$ denotes the edge weight function, which we define as $w_t(i,j) \propto \log\left(P_{\max}|h_{ij,t}|^2/N\right)$.

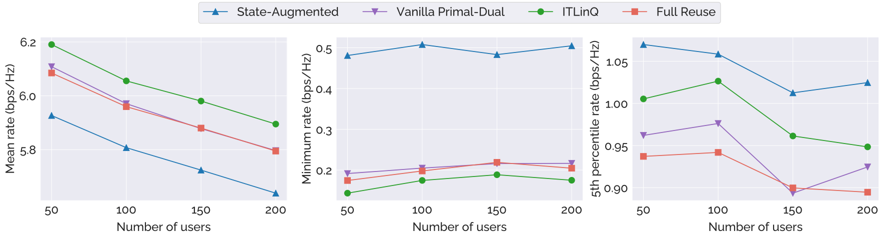


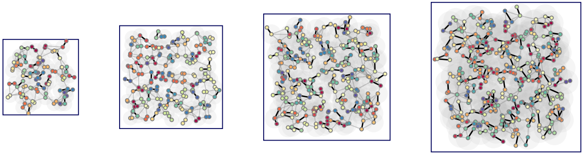
- We leverage GNN architectures to parameterize the resource allocation policies.
- Final node features at the output of the GNN are converted to resource allocation decisions.



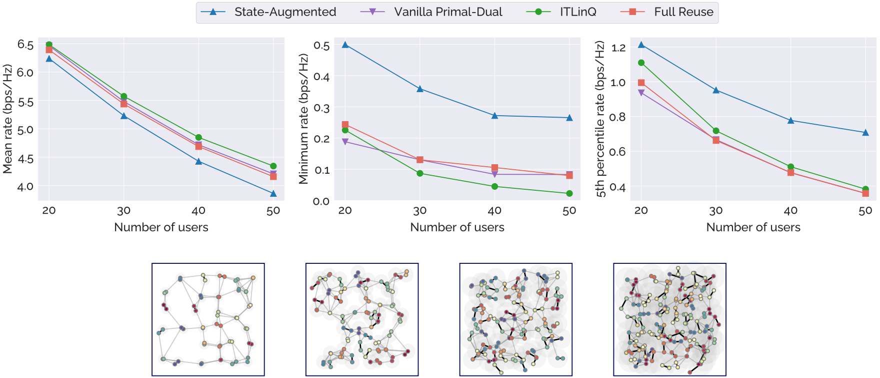
Scalability With Constant Network Density

- The network area size increases proportionally to the number of transmitter-receiver pairs.
- Policies are evaluated on the same network size that they have been trained on.

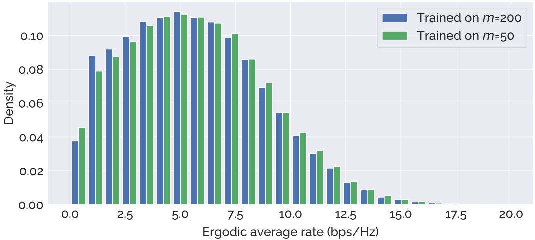


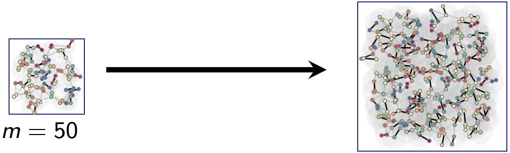


- The network area size is fixed regardless of the number of transmitter-receiver pairs.
- Policies are evaluated on the same network size that they have been trained on.



Policies are evaluated on a family of networks with $m = 200$ transmitter-receiver pairs.



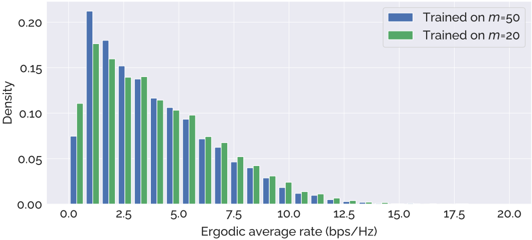


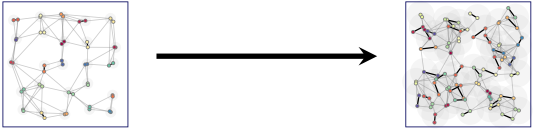
$m = 50$

$m = 200$

Transferability With Variable Network Density

Policies are evaluated on a family of networks with $m = 50$ transmitter-receiver pairs.





$m = 20$ $m = 50$

Federated Learning with GNNs

- GNNs can enable distributed training of models in a federated learning scenario.

- A group of agents attempt to learn a shared model $\mathbf{w}^\star$ with minimium average loss across agents:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{(\mathbf{x},y)\sim\mathfrak{D}_i}[\ell(f_{\mathbf{w}}(\mathbf{x}), y)].$$

- Considering a graph structure, we can have a constrained formulation:

$$\min_{\mathbf{w}_1,\dots,\mathbf{w}_n\in\mathbb{R}^d} \quad g(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{(\mathbf{x},y)\sim\mathfrak{D}_i}[\ell(f_{\mathbf{w}_i}(\mathbf{x}), y)],$$

$$\text{s.t.} \quad \mathbf{w}_i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} \mathbf{w}_j, \quad \text{for all } i = 1,\dots,N.$$

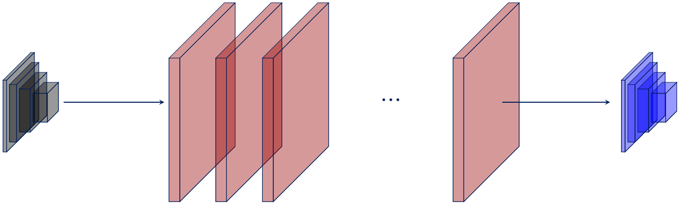
- A major challenge: High communication cost between the agents (and a central server).

- Instead of training the model $\mathbf{W}$ directly, we train a meta model $\mathbf{\Phi}(\mathbf{W}_0, \mathcal{D}; \theta)$, whose output is $\mathbf{W}^\star$:

$$\mathbf{W}^\star = \mathbf{\Phi}(\mathbf{W}_0, \mathcal{D}; \theta^\star) \quad \text{where} \quad \theta^\star = \arg\min_{\theta \in \mathbb{R}^P} \mathbb{E}\left[g(\mathbf{\Phi}(\mathbf{W}_0, \mathcal{D}; \theta))\right].$$

- The meta model takes as input the initial model $\mathbf{W}_0$ and a set of local datasets $\mathcal{D}$.
- We parameterize the meta model using L layers to mimic update rules of an iterative algorithm:

$$\mathbf{W}_l = \phi_l(\mathbf{W}_{l-1}, \mathcal{D}; \boldsymbol{\theta}_l), \quad l = 1, \ldots, L.$$



- Instead of the whole datasets $\mathcal{D}$, we feed stochastic batches of data $\mathcal{B}_l$ to the meta model:

$$\mathbf{W}_l = \phi_l(\mathbf{W}_{l-1}, \mathcal{D}; \boldsymbol{\theta}_l) \quad \rightarrow \quad \mathbf{W}_l = \phi_l(\mathbf{W}_{l-1}, \mathcal{B}_l; \boldsymbol{\theta}_l).$$

- We encourage the model parameters to improve after every layer using descending constraints:

$$\begin{aligned} \min_{\boldsymbol{\theta} \in \mathbb{R}^p} \quad & \mathbb{E}\left[g(\boldsymbol{\Phi}(\mathbf{W}_0, \mathcal{B}; \theta))\right] \\ \text{s.t.} \quad & \mathbb{E}\left[\|\nabla g(\mathbf{W}_l)\| - (1-\epsilon)\,\|\nabla g(\mathbf{W}_{l-1})\|\right] \leq 0, \text{ for all } l = 1, \dots, L, \\ & \mathbf{W}_l = \phi_l(\mathbf{W}_{l-1}, \mathcal{B}_l; \boldsymbol{\theta}_l), \quad \text{for all } l = 1, \dots, L. \end{aligned}$$

Hadou-NaderiAlizadeh-Ribeiro, *Stochastic Unrolled Federated Learning*, arxiv.org/abs/2305.15371

- Distributed gradient descent (DGD) is a distributed iterative algorithm with the update rule:

$$\mathbf{w}_i(l) = \sum_{j \in \mathcal{N}_i} s_{ij} \mathbf{w}_j(l-1) - \beta \nabla g_i(\mathbf{w}_i(l-1)), \quad i = 1, \dots, N.$$

- DGD relies on communication among agents, and local updates of the model using local data.
- We replace the first term with a GNN layer and the second term with a local FCNN:

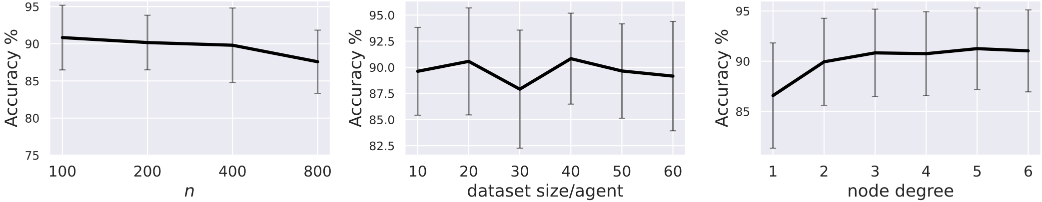
$$\mathbf{W}_l = \sum_{k=0}^{K-1} h_{kl} \mathbf{S}^k \mathbf{W}_{l-1} - \sigma\left(\left[\mathbf{W}_{l-1}, \mathcal{B}_l\right] \mathbf{M}_l + \mathbf{b}_l\right)$$

Hadou-NaderiAlizadeh-Ribeiro, *Stochastic Unrolled Federated Learning*, arxiv.org/abs/2305.15371

- Accuracy levels evaluated over randomly selected 3-class subsets of CIFAR-10 with 100 agents.

Training Algorithm	Accuracy	#Layers/Iterations
Centralized	25.81 ± 13.92	10
FedAvg	15.53 ± 12.29	10
SURF + DGD + GNN	$\mathbf{90.83 \pm 04.35}$	10
Centralized	$\mathbf{92.71 \pm 03.26}$	300
FedAvg	90.35 ± 03.69	300

- The trained meta-GNN transfers to different numbers of agents, dataset sizes, and topologies.

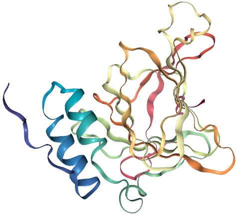


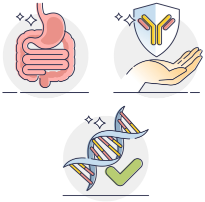
Protein Property Prediction with GNNs

- GNNs can enable learning over protein structures in biological systems.

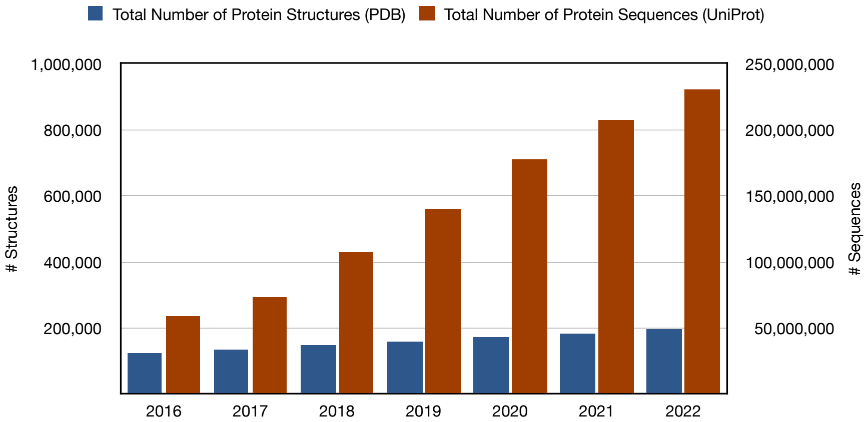
Protein Sequence – Structure – Function

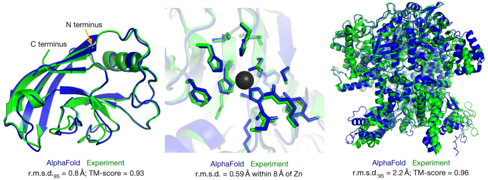
MEQTEVL...SGSLENN

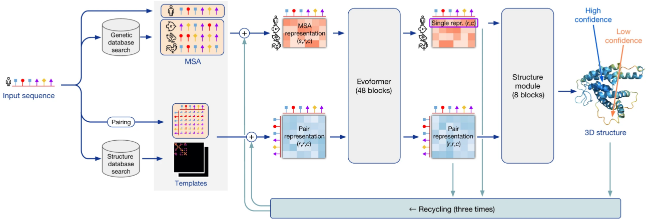


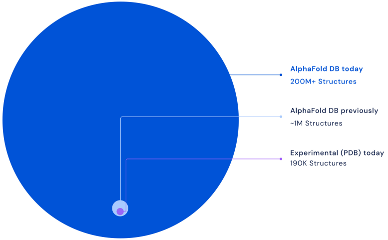


Protein Structure vs. Sequence Data Growth





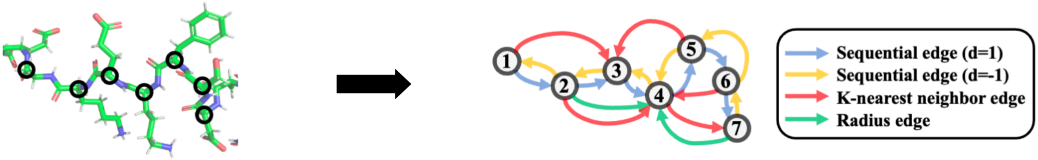




Jumper et al., *Highly accurate protein structure prediction with AlphaFold*, Nature, doi.org/10.1038/s41586-021-03819-2

Modeling Protein Structures as Graphs

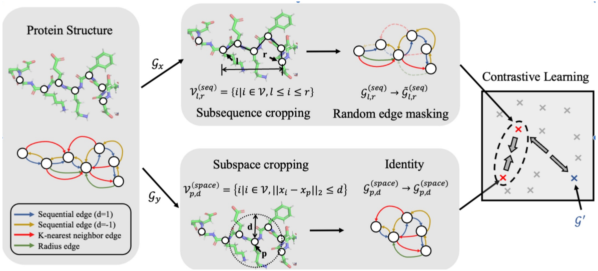
- Each node in the protein graph represents the Carbon-α atom of a residue (i.e., amino acid).
- 3D node coordinates given by $\mathbf{X} \in \mathbf{R}^{n \times 3}$ could be used as input node features.
- Graph adjacency matrix $\mathbf{S}$ can be derived via proximity in the sequence and/or structure.



Zhang et al., *Protein Representation Learning by Geometric Structure Pretraining*, ICLR, arxiv.org/abs/2203.06125

GNNs Trained as Foundation Models for Protein Structures

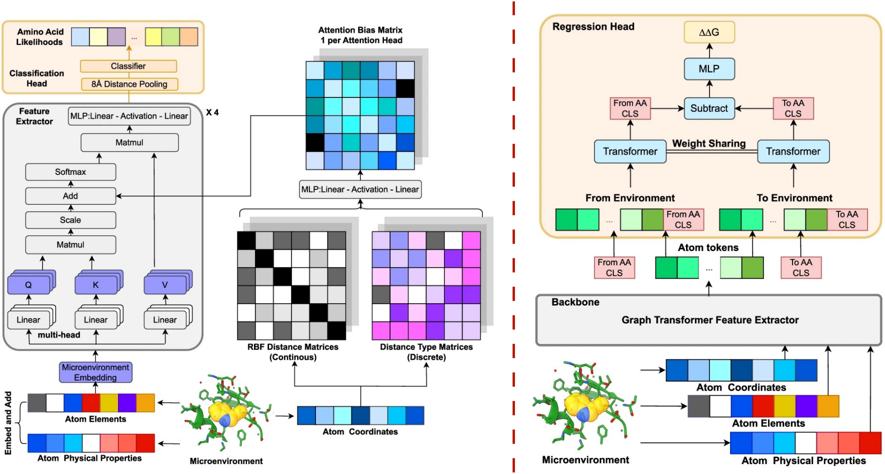
- A GNN $\Phi(\mathbf{X}; \mathbf{S}, \mathbf{H})$ can be pre-trained to minimize a contrastive loss on protein graph embeddings.



Method	Pretraining Dataset (Size)	EC	GO			Fold Classification				Reaction
			BP	MF	CC	Fold	Super.	Fam.	Avg.	
CNN (Shanehsazzadeh et al., 2020)	-	0.545	0.244	0.354	0.287	11.3	13.4	53.4	26.0	51.7
ResNet (Rao et al., 2019)	-	0.605	0.280	0.405	0.304	10.1	7.21	23.5	13.6	24.1
LSTM (Rao et al., 2019)	-	0.425	0.225	0.321	0.283	6.41	4.33	18.1	9.61	11.0
Transformer (Rao et al., 2019)	-	0.238	0.264	0.211	0.405	9.22	8.81	40.4	19.4	26.6
GCN (Kipf & Welling, 2017)	-	0.320	0.252	0.195	0.329	16.8*	21.3*	82.8*	40.3*	67.3*
GAT (Veličković et al., 2018)	-	0.368	0.284†	0.317†	0.385†	12.4	16.5	72.7	33.8	55.6
GVP (Jing et al., 2021)	-	0.489	0.326†	0.426†	0.420†	16.0	22.5	83.8	40.7	65.5
3DCNN_MQA (Derevyanko et al., 2018)	-	0.077	0.240	0.147	0.305	31.6*	45.4*	92.5*	56.5*	72.2*
GraphQA (Baldassarre et al., 2021)	-	0.509	0.308	0.329	0.413	23.7*	32.5*	84.4*	46.9*	60.8*
New IEConv (Hermosilla & Ropinski, 2022)	-	0.735	0.374	0.544	0.444	47.6*	70.2*	99.2*	72.3*	**87.2***
DeepFRI (Gligorijević et al., 2021)	Pfam (10M)	0.631	0.399	0.465	0.460	15.3*	20.6*	73.2*	36.4*	63.3*
ESM-1b (Rives et al., 2021)	UniRef50 (24M)	0.864	0.452	**0.657**	0.477	26.8	60.1	97.8	61.5	83.1
ProtBERT-BFD (Elnaggar et al., 2021)	BFD (2.1B)	0.838	0.279†	0.456†	0.408†	26.6*	55.8*	97.6*	60.0*	72.2*
LM-GVP (Wang et al., 2022b)	UniRef100 (216M)	0.664	0.417†	0.545†	**0.527**†	-	-	-	-	-
New IEConv (Hermosilla & Ropinski, 2022)	PDB (476K)	-	-	-	-	50.3*	**80.6***	99.7*	76.9*	**87.6***
Multiview Contrast	AlphaFoldDB (805K)	**0.874**	**0.490**	**0.654**	0.488	54.1	**80.5**	**99.9**	**78.1**	**87.5**

Zhang et al., *Protein Representation Learning by Geometric Structure Pretraining*, ICLR, arxiv.org/abs/2203.06125

- Graph Transformers enable learning on multiple protein graph structures simultaneously.

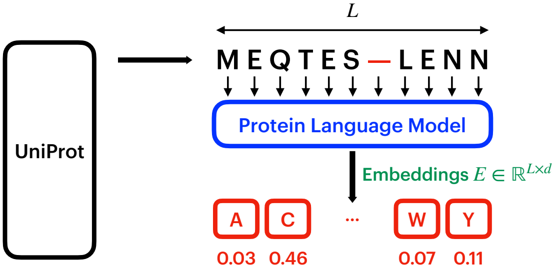


Diaz et al., *Stability Oracle: a structure-based graph-transformer framework for identifying stabilizing mutations*, Nature, doi.org/10.1038/s41467-024-49780-2

- PLM architectures are pre-trained using millions of sequences via the unsupervised masking objective

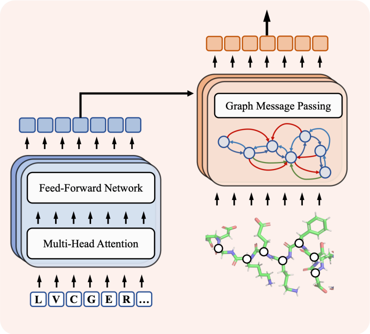
$$\mathcal{L}_{\text{MLM}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j\in\mathcal{M}_i}\log p_\theta(s_{ij}|s_{i,\setminus\mathcal{M}_i})$$

- This leads to intermediate embeddings $\mathbf{E} \in \mathbf{R}^{L\times d}$ that can be used for downstream tasks.



Lin et al., *Evolutionary-scale prediction of atomic-level protein structure with a language model*, Science, science.org/doi/10.1126/science.ade2574
Elnaggar et al., *ProtTrans: Toward understanding the language of life through self-supervised learning*, IEEE TPAMI, arxiv.org/abs/2007.06225

- PLM-generated embeddings can be used as input graph signals for subsequent GNN models.

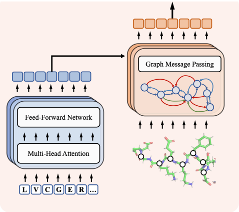


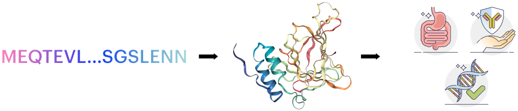
Method	PLM	Struct. Info.	EC	GO-BP	GO-MF
			F_{max}	F_{max}	F_{max}
ProtBERT-BFD[1]	✓	✗	0.838	0.279	0.456
ESM-2-650M[1]	✓	✗	<u>0.880</u>	<u>0.460</u>	<u>0.661</u>
GearNet	✗	✓	0.730	0.356	0.503
ESM-GearNet	✓	✓	<u>**0.890**</u>	<u>**0.488**</u>	<u>**0.681**</u>

Zhang et al., *A Systematic Study of Joint Representation Learning on Protein Sequences and Structures*, ICLR MLDD, arxiv.org/abs/2303.06275

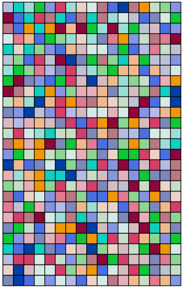
- Unsupervised GNN-based losses can be used for enforcing structural constraints on PLMs.

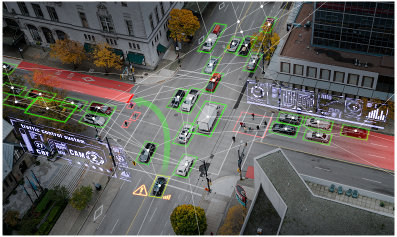
$$\min_{\theta_{\text{PLM}}, \mathbf{H}_{\text{GNN}}} \quad \mathcal{L}_{\text{Seq}}(\theta_{\text{PLM}}),$$
$$\text{s.t.} \quad \mathcal{L}_{\text{Str}}(\mathbf{X}_i, \mathbf{S}_i; \theta_{\text{PLM}}, \mathbf{H}_{\text{GNN}}) \leq \epsilon_i, \quad \text{for all } i = 1, \ldots, N.$$





Wang-Heinzinger-NaderiAlizadeh, *Fusing Protein Structures and Sequences: A Constrained Learning Approach*, In Preparation.




traffic control system
CAM 2