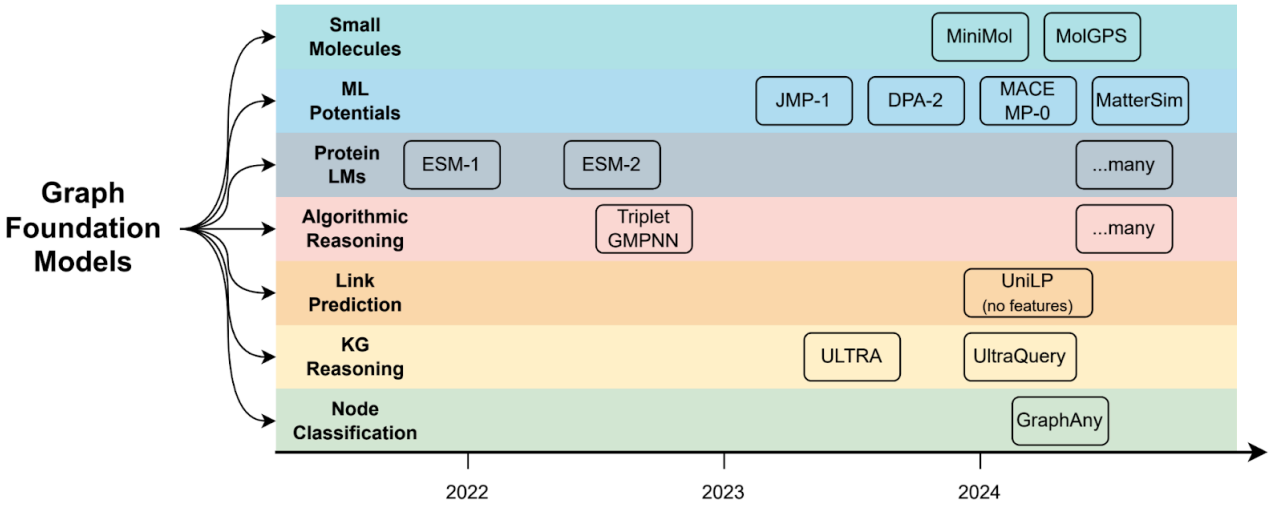
Foundation Models in Graph & Geometric Deep Learning

In this post, we argue that the era of Graph FMs has already begun and provide a few examples of how one can use them already today.

Michael Galkin
Jun 18, 2024 28 min read

Foundation Models in language, vision, and audio have been among the primary research topics in Machine Learning in 2024 whereas FMs for graph-structured data have somewhat lagged behind. In this post, we argue that the era of Graph FMs has already begun and provide a few examples of how one can use them already today.

This post was written and edited by Michael Galkin and Michael Bronstein with significant contributions from Jianan Zhao, Haitao Mao, Zhaocheng Zhu.



The timeline of emerging foundation models in graph- and geometric deep learning. Image by Authors.

Table of Contents

What are Graph Foundation Models and how to build them?

Since there is a certain degree of ambiguity in what counts as a "foundational" model, it would be appropriate to start with a definition to establish a common ground:

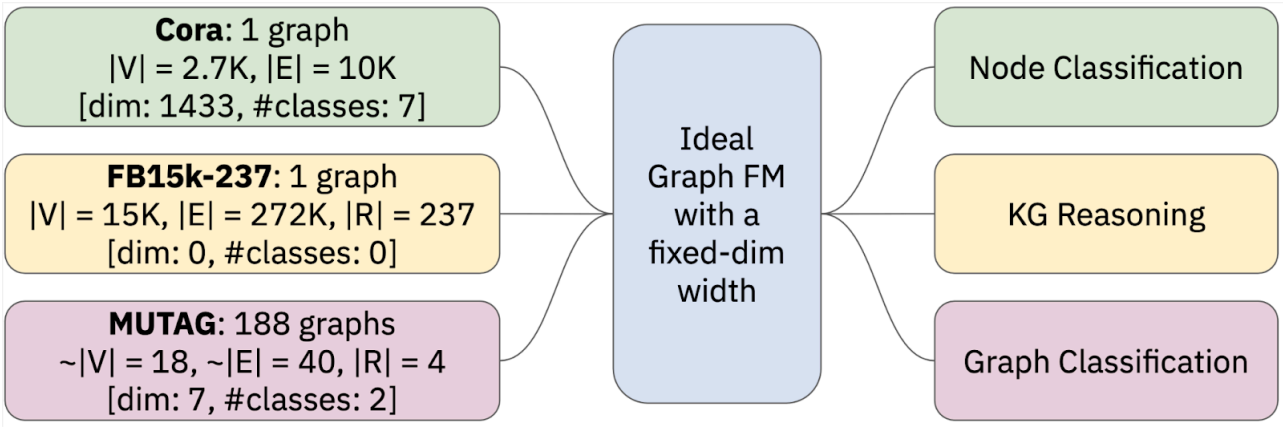
> "A Graph Foundation Model is a single (neural) model that learns transferable graph representations that can generalize to any new, previously unseen graph"

One of the challenges is that graphs come in all forms and shapes and their connectivity and feature structure can be very different. Standard Graph Neural Networks (GNNs) are not "foundational" because they can work in the best case only on graphs with the same type and dimension of features. Graph heuristics like Label Propagation or Personalized PageRank that can run on any graph can

neither be considered Graph FMs because they do not involve any learning. As much as we love Large Language Models, it is still unclear whether parsing graphs into sequences that can be then passed to an LLM (like in GraphText or Talk Like A Graph) is a suitable approach for retaining graph symmetries and scaling to anything larger than toy-sized datasets (we leave LLMs + Graphs to a separate post).

Perhaps the most important question for designing Graph FMs is transferable graph representations. LLMs, as suggested in the recent ICML 2024 position paper by Mao, Chen et al., can squash any text in any language into tokens from a fixed-size vocabulary. Video-Language FMs resort to patches that can always be extracted from an image (one always has RGB channels in any image or video). It is not immediately clear what could a universal featurization (à la tokenization) scheme be for graphs, which might have very diverse characteristics, e.g.:

- One large graph with node features and some given node labels (typical for node classification tasks)
- One large graph without node features and classes, but with meaningful edge types (typical for link prediction and KG reasoning)
- Many small graphs with/without node/edge features, with graph-level labels (typical for graph classification and regression)



🦄 An ideal graph foundation model that takes any graph with any node/edge/graph features and performs any node- / edge- / graph-level task. Such Graph FMs do not exist in pure form as of mid-2024. Image by Authors

So far, there is a handful of open research questions for the graph learning community when designing Graph FMs:

1️⃣ **How to generalize across graphs with heterogeneous node/edge/graph features?** For example, the popular Cora dataset for node classification is one graph with node features of dimension 1,433, whereas the Citeseer dataset has 3,703-dimensional features. How can one define a single representation space for such diverse graphs?

2️⃣ **How to generalize across prediction tasks?** Node classification tasks may have a different number of node classes (e.g., Cora has 7 classes and Citeseer 6). Even further, can a node classification model perform well in link prediction?

3️⃣ **What should the foundational model expressivity be?** Much research has been done on the expressive power of GNNs, typically resorting to the analogy with Weisfeiler-Lehman isomorphism tests. Since graph foundational models should ideally handle a broad spectrum of problems, the right expressive power is elusive. For instance, in node classification tasks, node features are important along with graph homophily or heterophily. In link prediction, structural patterns and breaking automorphisms are more important (node features often don't give a huge performance boost). In graph-level tasks, graph isomorphism starts to play a crucial role. In 3D geometric tasks like molecule generation, there is an additional complexity of continuous symmetries to take care of (see the Hitchhiker's Guide to Geometric GNNs).

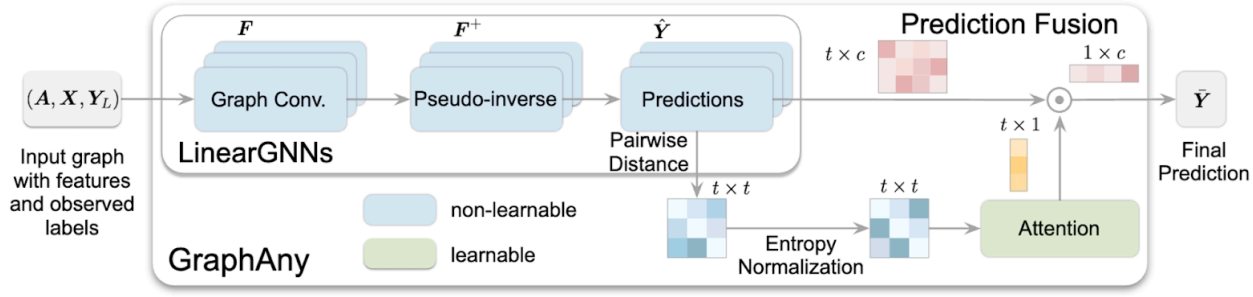
In the following sections, we will show that at least in some tasks and domains, Graph FMs are already available. We will highlight their design choices when it comes to transferable features and practical benefits when it comes to inductive inference on new unseen graphs.

📚 **Read more in references [1][2] and Github Repo**

Node Classification: GraphAny

For years, GNN-based node classifiers have been limited to a single graph dataset. That is, given e.g. the Cora graph with 2.7K nodes, 1433-dimensional features, and 7 classes, one has to train a GNN specifically on the Cora graph with its labels and run inference on the same graph. Applying a trained model on another graph, e.g. Citeseer with 3703-dimensional features and 6 classes would run into an unsurmountable difficulty: how would one model generalize to different input feature dimensions and a different number of classes? Usually, prediction heads are hardcoded to a fixed number of classes.

GraphAny is, to the best of our knowledge, the first Graph FM where a single pre-trained model can perform node classification on any graph with any feature dimension and any number of classes. A single GraphAny model pre-trained on 120 nodes of the standard Wisconsin dataset successfully generalizes to 30+ other graphs of different sizes and features and, on average, outperforms GCN and GAT graph neural network architectures trained from scratch on each of those graphs.



Overview of GraphAny: LinearGNNs are used to perform non-parametric predictions and derive the entropy-normalized distance features. The final prediction is generated by fusing multiple LinearGNN predictions on each node with attention learned based on the distance features. Source: Zhao et al.

Setup: Semi-supervised node classification: given a graph G, node features X, and a few labeled nodes from C classes, predict labels of target nodes (binary or multi-class classification). The dimension of

node features and the number of unique classes are not fixed and are graph-dependent.

What is transferable: Instead of modeling a universal latent space for all possible graphs (which is quite cumbersome or maybe even practically impossible), GraphAny bypasses this problem and focuses on the *interactions between predictions of spectral filters*. Given a collection of high-pass and low-pass filters akin to Simplified Graph Convolutions (for instance, operations of the form AX and (I-A)X, dubbed "LinearGNNs" in the paper) and known node labels:

0️⃣ GraphAny applies filters to all nodes;

1️⃣ GraphAny obtains optimal weights for each predictor from nodes with known labels by solving a least squares optimization problem in closed form (optimal weights are expressed as a pseudoinverse);

2️⃣ Applies the optimal weights to unknown nodes to get tentative prediction logits;

3️⃣ Computes pair-wise distances between those logits and applies entropy regularization (such that different graph- and feature sizes will not affect the distribution). For example, for 5 LinearGNNs, this would result in 5 x 4 = 20 combinations of logit scores;

4️⃣ Learns the inductive attention matrix over those logits to weight the predictions most effectively (e.g., putting more attention to high-pass filters for heterophilic graphs).

In the end, the only learnable component in the model is the parameterization of attention (via MLP), which *does not depend* on the target number of unique classes, but only on the number of LinearGNNs used. In the same vein, all LinearGNN predictors are non-parametric,

their updated node features and optimal weights can be pre-computed beforehand for faster inference.

📚 **Read more in references [3]**

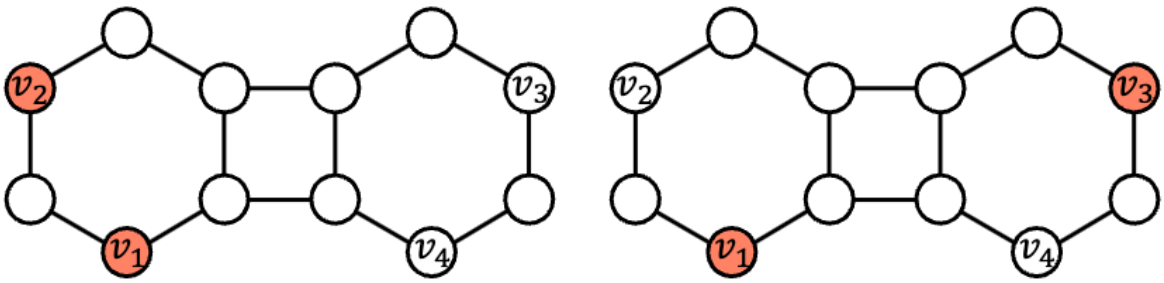
Link Prediction: Not yet

Setup: given a graph G, with or without node features, predict whether a link exists between a pair of nodes (v1, v2)

😢 For graphs with node features, we are not aware of any single transferable model for link prediction.

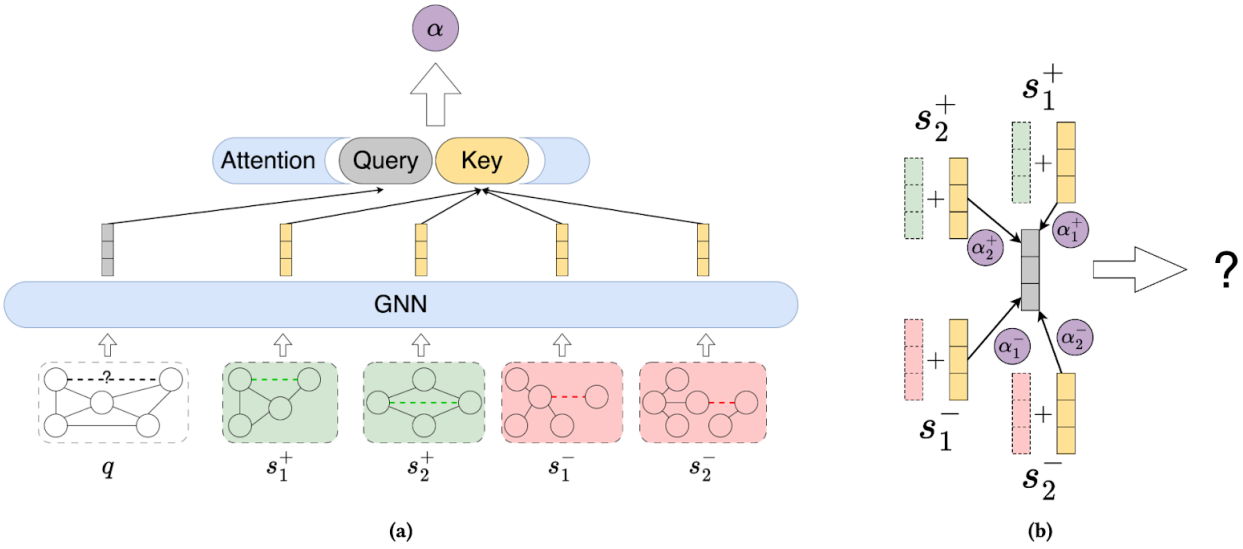
For non-featurized graphs (or when you decide to omit node features deliberately), there is more to say – basically, all GNNs with a labeling trick *potentially* can transfer to new graphs thanks to the uniform node featurization strategy.

It is known that in link prediction, the biggest hurdle is the presence of automorphic nodes (nodes that have the same structural roles) – vanilla GNNs assign them the same feature making two links (v1, v2) and (v1, v3) in the image below 👇 indistinguishable. Labeling tricks like Double Radius Node Labeling or Distance Encoding are such node featurization strategies that break automorphism symmetries.



V2 and v3 are automorphic nodes and standard GNNs score (v1,v2) and (v1,v3) equally. When we predict (v1, v2), we will label these two nodes differently from the rest, so that a GNN is aware of the target link when learning v1 and v2's representations. Similarly, when predicting (v1, v3), nodes v1 and v3 will be labeled differently. This way, the representation of v2 in the left graph will be different from that of v3 in the right graph, enabling GNNs to distinguish the non-isomorphic links (v1, v2) and (v1, v3). Source: Zhang et al.

Perhaps the only approach with a labeling trick (for non-featurized graphs) that was evaluated on link prediction on unseen graphs is UniLP. UniLP is an in-context, contrastive learning model that requires a set of positive and negative samples for each target link to be predicted. Practically, UniLP uses SEAL as a backbone GNN and learns an attention over a fixed number of positive and negative samples. On the other hand, SEAL is notoriously slow, so the first step towards making UniLP scale to large graphs is to replace subgraph mining with more efficient approaches like ELPH and BUDDY.



Overview of the Universal Link Predictor framework. (a) For predicting a query link q, we initially sample positive (s+) and negative (s-) in-context links from the target graph. Both the query link and these in-context links are independently processed through a shared subgraph GNN encoder. An attention mechanism then calculates scores based on the similarity between the query link and the in-context links. (b) The final representation of the query link, contextualized by the target graph, is obtained through a weighted summation, which combines the representations of the in-context links with their respective labels. Source: Dong et al.

What is transferable: structural patterns learned by labeling trick GNNs – it is proven that methods like Neural Bellman-Ford capture metrics over node pairs, eg, Personalized PageRank or Katz index (often used for link prediction).

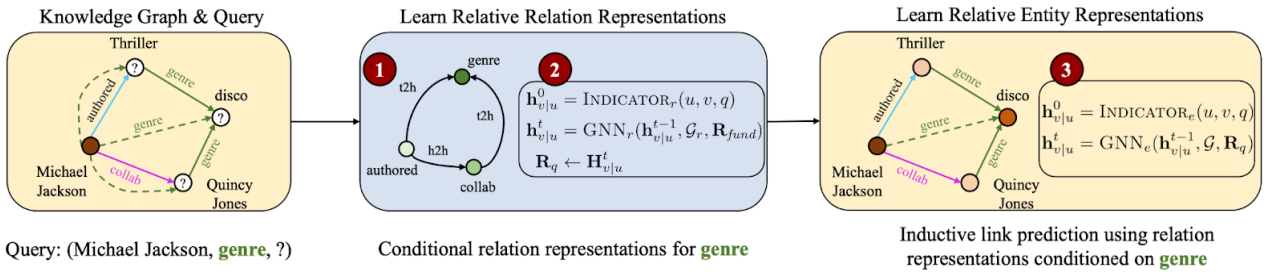
Now, as we know how to deal with automorphisms, the only step towards a single graph FM for link prediction would be to add a support for heterogeneous node features – perhaps GraphAny-style approaches might be an inspiration?

📚 **Read more in references [4][5][6][7]**

Knowledge Graph Reasoning: ULTRA and UltraQuery

Knowledge graphs have graph-specific sets of entities and relations, e.g. common encyclopedia facts from Wikipedia / Wikidata or biomedical facts in Hetionet, those relations have different semantics and are not directly mappable to each other. For years, KG reasoning models were hardcoded to a given vocabulary of relations and could not transfer to new, unseen KGs with completely new entities and relations.

ULTRA is the first foundation model for KG reasoning that transfers to any KG at inference time in the zero-shot manner. That is, a single pre-trained model can run inference on any multi-relational graph with any size and entity/relation vocabulary. Averaged over 57 graphs, ULTRA significantly outperforms baselines trained specifically on each graph. Recently, ULTRA was extended to UltraQuery to support even more complex logical queries on graphs involving conjunctions, disjunctions, and negation operators. UltraQuery transfers to unseen graphs and 10+ complex query patterns on those unseen graphs outperforming much larger baselines trained from scratch.



Given a query (Michael Jackson, genre, ?), ULTRA builds a graph of relations (edge types) to capture their interactions in the original graph conditioned on the query relation (genre) and derives relational representations from this smaller graph. Those features are then used as edge type features in the original bigger graph to answer the query.
Source: Galkin et al.

Setup: Given a multi-relational graph G with |E| nodes and |R| edge types, no node features, answer simple KG completion queries *(head,*

relation, ?) or complex queries involving logical operators by returning a probability distribution over all nodes in the given graph. The set of nodes and relation types depends on the graph and can vary.

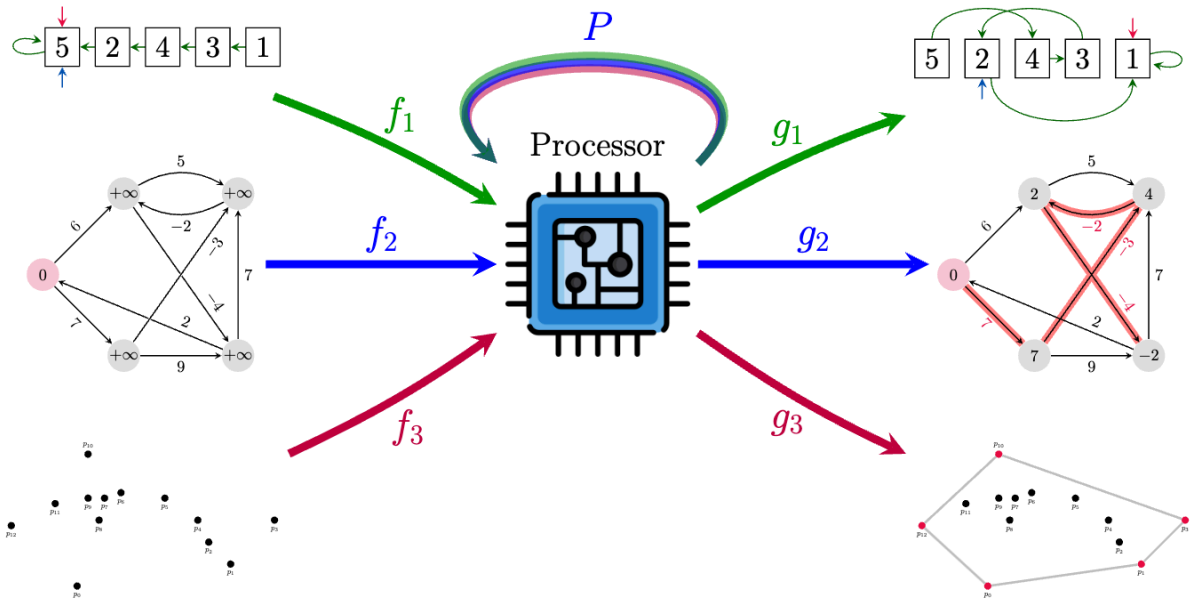
What is transferable: ULTRA relies on modeling relational interactions. Forgetting about relation identities and target graph domain for a second, if we see that relations "authored" and "collaborated" can share the same starting node, and relations "student" and "coauthor" in another graph can share a starting node, then the relative, structural representations of those two pairs of relations might be similar. This holds for any multi-relational graph in any domain, be it encyclopedia or biomedical KGs. ULTRA goes further and captures 4 such "fundamental" interactions between relations. Those fundamental interactions are transferable to any KG (together with learned GNN weights) – this way, one single pre-trained model is ready for inference on any unseen graph and simple or complex reasoning query.

Read more in the dedicated Medium post:

ULTRA: Foundation Models for Knowledge Graph Reasoning

📚 **Read more in references [8][9]**

Algorithmic Reasoning: Generalist Algorithmic Learner



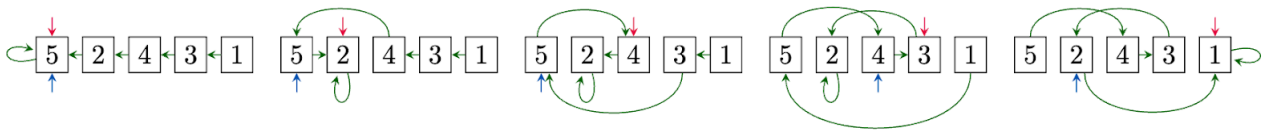
A generalist neural algorithmic learner is a single processor GNN P, with a single set of weights, capable of solving several algorithmic tasks in a shared latent space (each of which is attached to P with simple encoders/decoders f and g). Among others, the processor network is capable of sorting (top), shortest path-finding (middle), and convex hull finding (bottom). Source: Ibarz et al.

Setup: Neural algorithmic reasoning (NAR) studies the execution of standard algorithms (eg, sorting, searching, dynamic programming) in the latent space and generalization to the inputs of arbitrary size. A lot of such algorithms can be represented with a graph input and pointers. Given a graph G with node and edge features, the task is to simulate the algorithm and produce the correct output. Optionally, you can get access to hints – time series of intermediate states of the algorithm which can act as the intermediate supervised signal. Obviously, different algorithms require a different number of steps to execute, so length is not fixed here.

What is transferable: Homogeneous feature space and similar control flow for similar algorithms. For instance, Prim's and Dijkstra's algorithms share a similar structure, differing only in the choice of key function and edge relaxation subroutine. Besides, there are several proofs of a direct alignment between message passing and dynamic programming. This is the main motivation behind one "processor" neural network that updates latent states for all considered algorithms (30 classic algos from the CLRS book).

Triplet-GMPNN was the first such universal processor neural net (by 2024 it became rather standard in the NAR literature) – it is a GNN that operates on triples of nodes and their features (akin to Edge Transformers and triangular attention in AlphaFold). The model is trained in the multi-task mode on all algorithmic tasks in the benchmark with a handful of optimization and tricks. A single model bumps the average performance on 30 tasks by over 20% (in absolute numbers) compared to single-task specialist models.

Still, encoders and decoders are parameterized specifically for each task – one of the ways to unify the input and output formats might as well be text with LLM processors as done in the recent text version of CLRS.



```
insertion_sort:
key: [5 2 4 3 1], initial_trace: [5 2 4 3 1]
trace | pred:
[2 5 4 3 1], [2 4 5 3 1], [2 3 4 5 1] | [1 2 3 4 5]
```

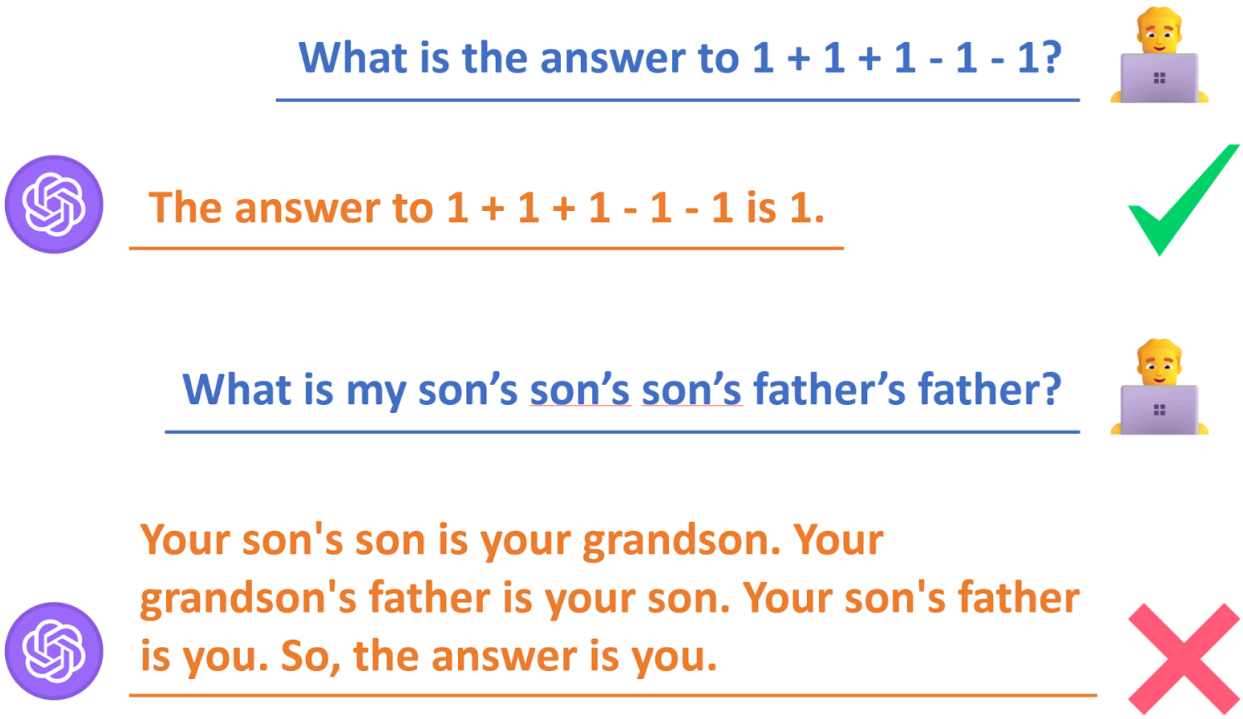
Top: The graph algorithmic trace of insertion sorting a list *[5, 2, 4, 3, 1]* in graph form. **Bottom**: The same algorithmic trace, represented textually, by using the CLRS-Text generator. The model receives as input (depicted in green) the input array (key) and the initial value of the sorting trace (initial_trace), using which it is prompted to predict the trace (depicted in blue) of gradually sorting the list, by inserting one element at a time into a partially sorted list, from left to right. At the end, the model needs to output the final sorted array (depicted in red), and it is evaluated on whether this array is predicted correctly. Source: Markeeva, McLeish, Ibarz, et al.

Perhaps the most interesting question of 2024 and 2025 in NAR is:

> *Can algorithmic reasoning ideas for OOD generalization be the key to generalizable LLM reasoning?*

LLMs notoriously struggle with complex reasoning problems, dozens of papers appear on arxiv every month trying a new prompting method to bump benchmarking performance another percentage-or-two, but most of them do not transfer across tasks of similar graph structures (see the

example below). There is a need for more principled approaches and NAR has the potential to fill this gap!



Failure of LLMs on reasoning problems with similar graph structures. Image by Authors.

📚 **Read more in references [10][11]**

Geometric and AI4Science Foundation Models

In the world of Geometric Deep Learning and scientific applications, foundation models are becoming prevalent as universal ML potentials, protein language models, and universal molecular property predictors. Although the universal vocabulary exists in most such cases (e.g., atom types in small molecules or amino acids in proteins) and we do not have to think about universal featurization, the main complexity lies in the real-world physical nature of atomistic objects – they have pronounced 3D structure and properties (like energy), which have theoretical justifications rooted in chemistry, physics, and quantum mechanics.

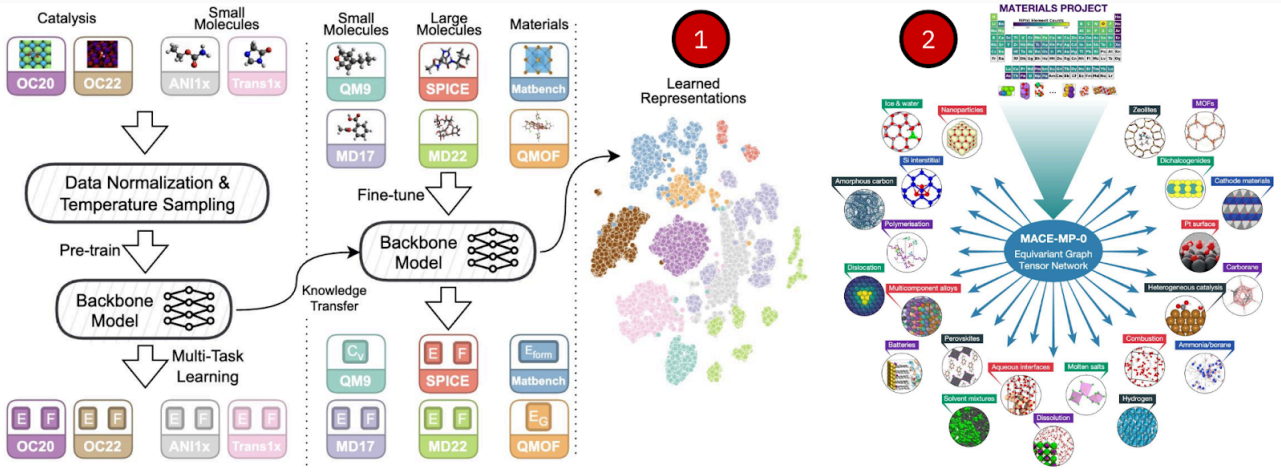
ML Potentials: JMP-1, DPA-2 for molecules, MACE-MP-0 and MatterSim for inorganic crystals

Setup: given a 3D structure, predict the energy of the structure and per-atom forces;

What is transferable: a vocabulary of atoms from the periodic table.

ML potentials estimate the potential energy of a chemical compound – like molecules or periodic crystals – given their 3D coordinates and optional input (like periodic boundary conditions for crystals). For any atomistic model, the vocabulary of possible atoms is always bound by the Periodic Table which currently includes 118 elements. The "foundational" aspect in ML potentials is to generalize to any atomistic structure (there can be combinatorially many) and be stable enough to be used in molecular dynamics (MD), drug- and materials discovery pipelines.

JMP-1 and DPA-2 released around the same time aim to be such universal ML potential models – they are trained on a sheer variety of structures – from organic molecules to crystals to MD trajectories. For example, a single pre-trained JMP-1 excels at QM9, rMD17 for small molecules, MatBench and QMOF on crystals, and MD22, SPICE on large molecules being on-par or better than specialized per-dataset models. Similarly, MACE-MP-0 and MatterSim are the most advanced FMs for inorganic crystals (MACE-MP-0 is already available with weights) evaluated on 20+ crystal tasks ranging from multicomponent alloys to combustion and molten salts. Equivariant GNNs are at the heart of those systems helping to process equivariant features (Cartesian coordinates) and invariant features (like atom types).



Sources: (1) Pre-training and fine-tuning of **JMP-1** for molecules and crystals, Shoghi et al (2) **MACE-MP-0** is trained only on the Materials Project data and transfers to molecular dynamics simulation across a wide variety of chemistries in the solid, liquid and gaseous phases, Batatia, Benner, Chiang, Elena, Kovács, Riebesell et al.

The next frontier seems to be ML-accelerated molecular dynamics simulations – traditional computational methods work at the femtosecond scale (10–15) and require millions and billions of steps to simulate a molecule, crystal, or protein. Speeding up such computations would have an immense scientific impact.

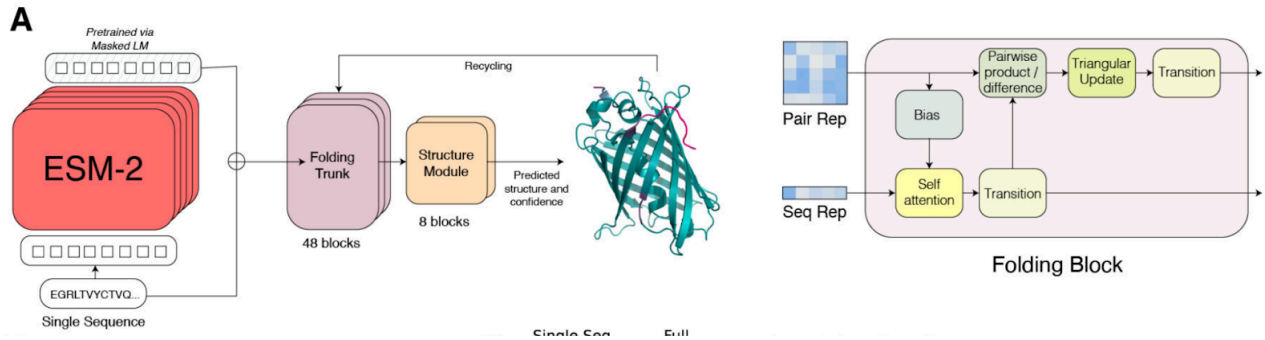
📚 **Read more in references [12][13][14][15]**

Protein LMs: ESM-2

Setup: given a protein sequence, predict the masked tokens akin to masked language modeling;

What is transferable: a vocabulary of 20 (22) amino acids.

Protein sequences resemble natural language with amino acids as tokens, and Transformers excel at encoding sequence data. Although the vocabulary of amino acids is relatively small, the space of possible proteins is enormous, so training on large volumes of known proteins might hint at the properties of unseen combinations. ESM-2 is perhaps the most popular protein LM thanks to the pre-training data size, a variety of available checkpoints, and informative features.



ESM2 as a masked LM and ESMFold for protein structure prediction. Source: Lin, Akin, Rao, Hie, et al.

ESM features are used in countless applications from predicting 3D structure (in ESMFold) to protein-ligand binding (DiffDock and its descendants) to protein structure generative models (like a recent FoldFlow 2). Bigger transformers and more data are likely to increase protein LMs' performance even further – at this scale, however, the data question becomes more prevalent (we also discuss the interplay between architecture and data in the dedicated section), eg, the ESM Metagenomic Atlas already encodes 700M+ structures including those seen outside humans in the soil, oceans, or hydrothermal vents. Is there a way to trillions of tokens as in common LLM training datasets?

📚 **Read more in references [16][17]**

2D Molecules: MiniMol and MolGPS

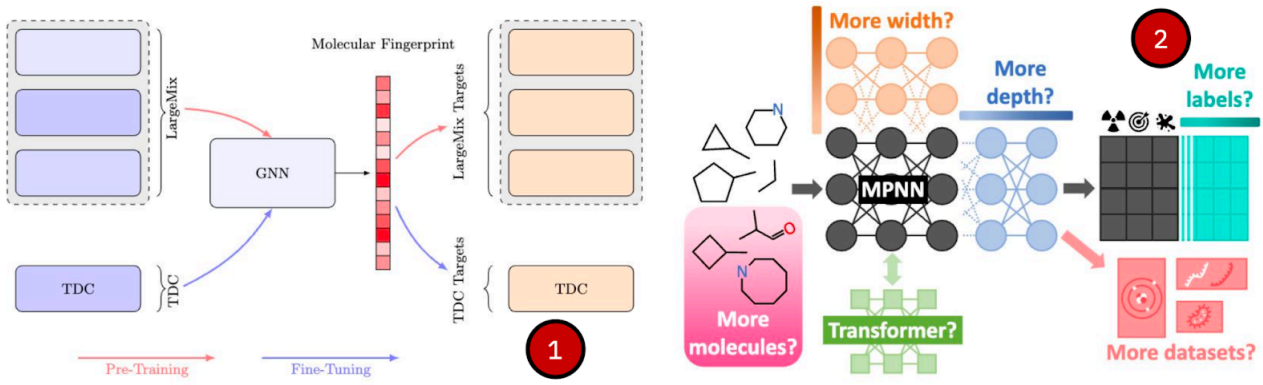
Setup: given a 2D graph structure with atom types and bond types, predict molecular properties

What is transferable: a vocabulary of atoms from the periodic table and bond types

With 2D graphs (without 3D atom coordinates) universal encoding and transferability come from a fixed vocabulary of atom and bond types which you can send to any GNN or Transformer encoder. Although molecular fingerprints have been used since 1960s (Morgan

fingerprints [18]), their primary goal was to evaluate similarity, not to model a latent space. The task of a single (large) neural encoder is to learn useful representations that might hint at certain physical molecular properties.

Recent examples of generalist models for learning molecular representations are MiniMol and MolGPS which have been trained on a large corpus of molecular graphs and probed on dozens of downstream tasks. That said, you still need to fine-tune a separate task-specific decoder / predictor given the models' representations – in that sense, one single pre-trained model will not be able to run zero-shot inference on all possible unseen tasks, rather on those for which decoders have been trained. Fine-tuning is still a good cheap option though since those models are orders of magnitude smaller than LLMs.



Source: (1) Workflow overview of the MiniMol pre-training and downstream task evaluation. (2) Criteria of the scaling study of MolGPS

📚 Read more in references [19][20]

Expressivity & Scaling Laws: Do Graph FMs scale?

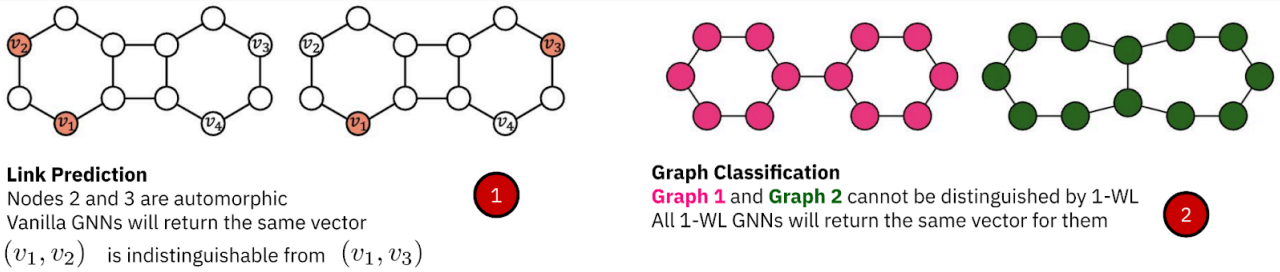
Transformers in LLMs and multi-modal frontier models are rather standard and we know some basic scaling principles for them. Do transformers (as an architecture, not LLMs) work equally well on

graphs? What are the general challenges when designing a backbone for Graph FMs?

If you categorize the models highlighted in the previous sections, only 2 areas feature transformers – protein LMs (ESM) with a natural sequential bias and small molecules (MolGPS). The rest are GNNs. There are several reasons for that:

- Vanilla transformers do not scale to any reasonably large graph larger than a standard context length (>4–10k nodes). Anything above that range requires tricks like feeding only subgraphs (losing the whole graph structure and long-range dependencies) or linear attention (that might not have good scaling properties). In contrast, GNNs are linear in the number of edges, and, in the case of sparse graphs (V ~ E), are linear in the number of nodes.
- Vanilla transformers without positional encodings are less expressive than GNNs. Mining positional encodings like Laplacian PEs on a graph with V nodes is $O(V^3)$.
- What should be a "token" when encoding graphs via transformers? There is no clear winner in the literature, e.g., nodes, nodes + edges, or subgraphs are all viable option

➡️ Touching upon **expressivity**, different graph tasks need to deal with different symmetries, e.g., automorphic nodes in link prediction lead to indistinguishable representations, whereas in graph classification/regression going beyond 1-WL is necessary for distinguishing molecules which otherwise might look isomorphic to vanilla GNNs.



Different tasks need to deal with different symmetries. Image by Authors. Sources of graphs: (1) Zhang et al, (2) Morris et al

This fact begs two questions:

> *How expressive should GFMs be? What is the trade-off between expressivity and scalability?*

Ideally, we want a single model to resolve all those symmetries equally well. However, more expressive models would lead to more computationally expensive architectures both in training and inference. We agree with the recent ICML'24 position paper on the future directions in Graph ML theory that the community should seek the balance between expressivity, generalization, and optimization.

Still, it is worth noting that with the growing availability of training data, it might be a computationally cheaper idea to defer learning complex symmetries and invariances directly from the data (instead of baking them into a model). A few recent good examples of this thesis are AlphaFold 3 and Molecular Conformer Fields that reach SOTA in many generative applications *without* expensive equivariant geometric encoders.

📚 Read more in references [21]

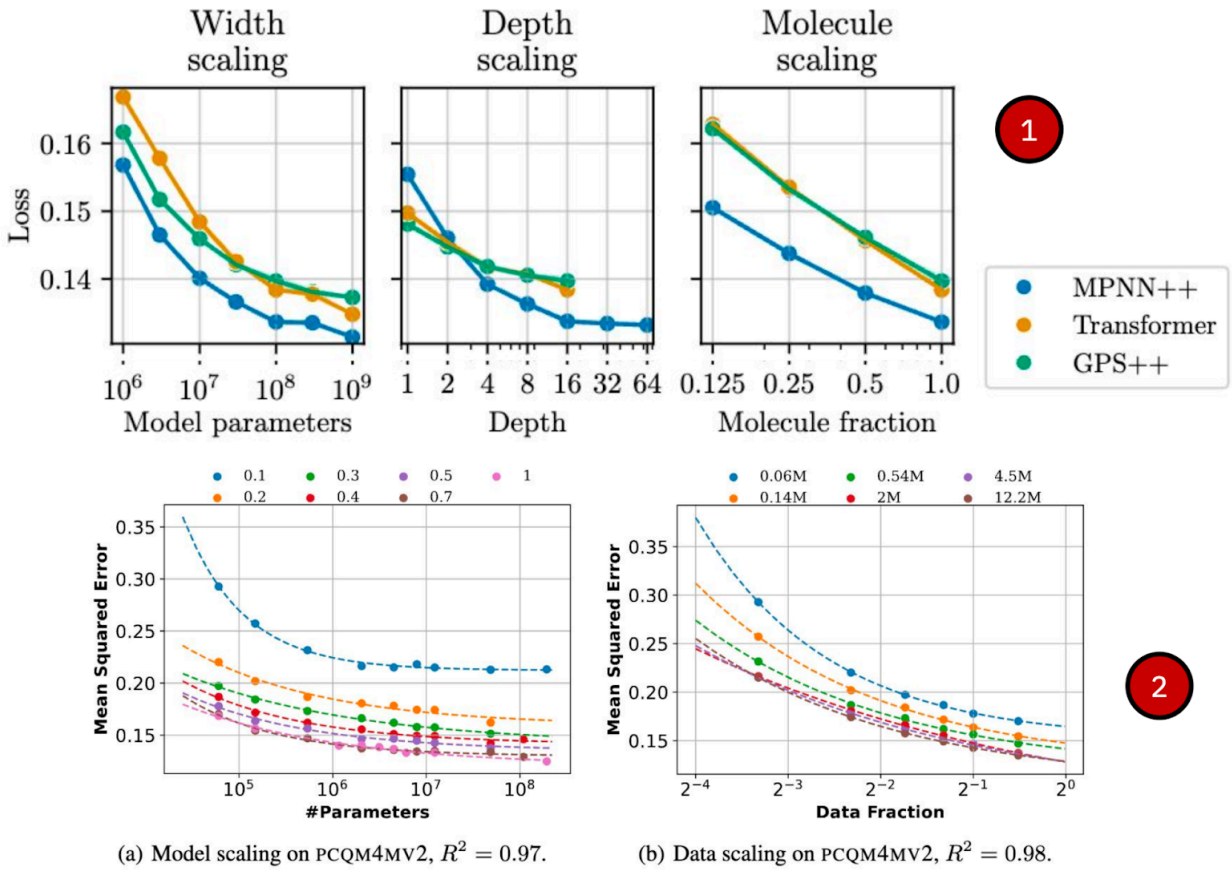
➡️ When it comes to **scaling**, both model and data should be scaled up. However:

❌ Non-geometric graphs: There is no principled study on scaling GNNs or Transformers to large graphs and common tasks like node

classification and link prediction. 2-layer GraphSAGE is often not very far away from huge 16-layer graph transformers. In a similar trend, in the KG reasoning domain, a single ULTRA model (discussed above) with <200k parameters outperforms million-sized shallow embedding models on 50+ graphs. Why is it happening? We'd hypothesize the crux is in 1️⃣ task nature – most of non-geometric graphs are noisy similarity graphs that are not bounded to a concrete physical phenomenon like molecules 2️⃣ Given rich node and edge features, models have to learn *representations of graph structures* (common for link prediction) or just *functions over given features* (a good example is node classification in OGB where most gains are achieved by adding an LLM feature encoder).

✅ Geometric graphs: There are several recent works focusing on molecular graphs:

- Frey et al (2023) study scaling of geometric GNNs for ML potentials;
- Sypetkowski, Wenkel et al (2024) introduce MolGPS and study scaling MPNNs and Graph Transformers up to 1B parameters on the large dataset of 5M molecules
- Liu et al (2024) probe GCN, GIN, and GraphGPS up to 100M parameters on molecular datasets up to 4M molecules.



(a) Model scaling on PCQM4MV2, $R^2 = 0.97$. (b) Data scaling on PCQM4MV2, $R^2 = 0.98$.

Scaling molecular GNNs and GTs. Sources: (1) Sypetkowski, Wenkel et al, (2) Liu et al

The Data Question: What should be scaled? Is there enough graph data to train Graph FMs?

1️⃣ **What should be scaled in graph data?** Nodes? Edges? The number of graphs? Something else?

There is no clear winner in the literature, we would rather gravitate towards a broader term ***diversity***, that is, a diversity of patterns in the graph data. For example, in node classification on large product graphs, it likely would not matter much if you train on a graph with 100M nodes or 10B nodes since it's the same nature of a user-item graph. However, showing examples with homophily and heterophily on different scales and sparsities might be quite beneficial. In **GraphAny**, showing examples of such graphs allowed to build a robust node classifier that generalizes to different graph distributions,

In KG reasoning with **ULTRA**, it was found that the ***diversity of relational patterns*** in pre-training plays the biggest role in inductive generalization, e.g., one large dense graph is worse than a collection of smaller but sparse, dense, few-relational, and many-relational graphs.

In molecular graph-level tasks, e.g., in **MolGPS**, scaling the number of unique molecules with different physical properties helps a lot (as shown in the charts above 👆).

Besides, UniAug finds that increased coverage of the structural patterns in pre-training data adds to the performance across different downstream tasks from various domains.

2️⃣ Is there enough data to train Graph FMs?

Openly available graph data is orders of magnitudes smaller than natural language tokens or images or videos, and it is fine. This very article includes thousands of language and image tokens and no explicit graphs (unless you try to parse this text to a graph like an abstract meaning representation graph). The number of 'good' proteins with known structures in PDB is small, the number of known 'good' molecules for drugs is small.

> Are Graph FMs doomed because of data scarcity?

Well, not really. The two open avenues are: (1) more sample-efficient architectures; (2) using more black-box and synthetic data.

Synthetic benchmarks like GraphWorld might be of use to increase the diversity of training data and improve generalization to real-world datasets. Black-box data obtained from scientific experiments, in turn, is likely to become the key factor in building successful foundation models in AI 4 Science – those who master it will prevail on the market.

The Road to Biology 2.0 Will Pass Through Black-Box Data

📚 **Read more in references [20][22][23]**

👉 Key Takeaways 👈

➡️ How to generalize across graphs with heterogeneous node/edge/graph features?

- Non-geometric graphs: Relative information transfers (such as prediction differences in *GraphAny* or relational interactions in *Ultra*), absolute information does not.
- Geometric graphs: transfer is easier thanks to the fixed set of atoms, but models have to learn some notion of physics to be reliable

➡️ How to generalize across prediction tasks?

- To date, there is no single model (among non-geometric GNNs) that would be able to perform node classification, link prediction, and graph classification in the zero-shot inference mode.
- Framing all tasks through the lens of one might help, eg, node classification can be framed as link prediction.

➡️ What is the optimal model expressivity?

- Node classification, link prediction, and graph classification leverage different symmetries.
- Blunt application of maximally expressive models quickly leads to exponential runtime complexity or enormous memory costs – need to maintain the *expressivity vs efficiency* balance.
- The link between expressivity, sample complexity (how much training data you need), and inductive generalization is still

unknown.

➡️ Data

- Openly available graph data is orders of magnitude smaller than text/vision data, models have to be sample-efficient.
- Scaling laws are at the emerging stage, it is still unclear what to scale – number of nodes? Edges? Motifs? What is the notion of a token in graphs?
- Geometric GNNs: there is much more experimental data available that makes little sense to domain experts but might be of value to neural nets.

---

1. Mao, Chen, et al. Graph Foundation Models Are Already Here. ICML 2024
2. Morris et al. Future Directions in Foundations of Graph Machine Learning. ICML 2024
3. Zhao et al. GraphAny: A Foundation Model for Node Classification on Any Graph. Arxiv 2024. Code on Github
4. Dong et al. Universal Link Predictor By In-Context Learning on Graphs, arxiv 2024
5. Zhang et al. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. NeurIPS 2021
6. Chamberlain, Shirobokov, et al. Graph Neural Networks for Link Prediction with Subgraph Sketching. ICLR 2023
7. Zhu et al. Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction. NeurIPS 2021
8. Galkin et al. Towards Foundation Models for Knowledge Graph Reasoning. ICLR 2024

9. Galkin et al. Zero-shot Logical Query Reasoning on any Knowledge Graph. arxiv 2024. Code on Github
10. Ibarz et al. A Generalist Neural Algorithmic Learner LoG 2022
11. Markeeva, McLeish, Ibarz, et al. The CLRS-Text Algorithmic Reasoning Language Benchmark. arxiv 2024
12. Shoghi et al. From Molecules to Materials: Pre-training Large Generalizable Models for Atomic Property Prediction. ICLR 2024
13. Zhang, Liu et al. DPA-2: Towards a universal large atomic model for molecular and material simulation, arxiv 2023
14. Batatia et al. A foundation model for atomistic materials chemistry, arxiv 2024
15. Yang et al. MatterSim: A Deep Learning Atomistic Model Across Elements, Temperatures and Pressures, arxiv 2024
16. Rives et al. Biological Structure and Function Emerge from Scaling Unsupervised Learning to 250 Million Protein Sequences. PNAS 2021
17. Lin, Akin, Rao, Hie, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. Science 2023. Code
18. Morgan HL (1965) The generation of a unique machine description for chemical structures – a technique developed at chemical abstracts service. J Chem Doc 5:107–113.
19. Kläser, Banaszewski, et al. MiniMol: A Parameter Efficient Foundation Model for Molecular Learning, arxiv 2024
20. Sypetkowski, Wenkel et al. On the Scalability of GNNs for Molecular Graphs, arxiv 2024
21. Morris et al. Future Directions in Foundations of Graph Machine Learning. ICML 2024
22. Liu et al. Neural Scaling Laws on Graphs, arxiv 2024

23. Frey et al. Neural scaling of deep chemical models, Nature Machine Intelligence 2023