On the Extrapolation of Graph Neural Networks and Beyond – Group 4 (Rolling a Dice)

Yihe Deng
yihedeng@g.ucla.edu
704638248

Yu Yang
yuyang@cs.ucla.edu
904682607

Qing Li
liqing@ucla.edu
905224203

Shuwen Qiu
s.qiu@ucla.edu
005061173

ABSTRACT

Graph Neural Networks have been commonly used for modeling structured data and witnessed great success. With its surging development in different fields of applications, researchers are increasingly interested in the abilities of GNNs including its expressive power, generalization ability and robustness. Most works that study the generalization ability of GNNs focus on evaluating how well GNNs could adapt to unseen data drawn from the same distribution as the training data. In this project, we focus on the less-studied extrapolation ability of GNNs instead of it generalization (interpolation) ability. In other words, we are interested in how well GNNs could obtain higher-dimensional insights outside the support of the training distribution. We follow the work [40] and expands its empirical analysis by introducing our new evaluation tasks. With a more comprehensive investigation and analysis, we aim to provide a deeper insight into the extrapolation ability of different baseline GNN architectures. Our code and presentation slides are hosted at https://github.com/liqing-ustc/CS249-GNN-Final.

KEYWORDS

graph neural networks, extrapolation, out-of-distribution

ACM Reference Format:
Yihe Deng, Yu Yang, Qing Li, and Shuwen Qiu. 2021. On the Extrapolation of Graph Neural Networks and Beyond – Group 4 (Rolling a Dice). In *Proceedings of (CS-249-GNN).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the expressive power of graph as a unique type of data structure, graph neural networks have been gaining significant momentum in the last few years. Different GNN architectures have been proposed [21, 27, 32, 38] and applied to diverse areas of interesting problems [3, 25, 34]. The increased interests naturally advance the studies in multiple aspects of graph neural networks properties, including the expressive power [38], generalization ability [11, 22], and robustness [36, 42, 44] of the networks.

In this paper, we investigate a specific ability of graph neural networks—extrapolation—and make a distinction between extrapolation and the commonly-studied generalization. We thus give the informal definitions for the two different aspect of studies as follows:

- **Generalization/Interpolation** refers to the model's ability to adapt to new, unseen data drawn from the same distribution as the training data.
- **Extrapolation** refers to the model's ability to obtain higher-dimensional insights from lower-dimensional training, where the test data comes from a different distribution than training data.

In other words, extrapolation can be a much harder task than generalization/interpolation even for humans, where one is expected to extend from known data to higher-level knowledge and learn outside the training distribution.

[40] provide a comprehensive analysis on the extrapolation power of both multilayer perceptrons (MLPs) and GNNs. With both theoretical and empirical evidences, they discover that GNNs could extrapolate better to new data distributions. Following their work, we take a step further to more specifically study the extrapolation ability of GNNs and expand their experiments by introducing new extrapolation tasks for evaluation. We use our proposed extrapolation tasks where the test graph structure is different from that of the training graph, and evaluate the performances of multiple baseline graph models to provide our insights and analysis.

2 PROBLEM FORMALIZATION

2.1 Interpolation and Extrapolation

Given a distribution of interest $\mathcal{D}$ with domain $\mathcal{X}$ and target domain y, there is an underlying function $g : \mathcal{X} \to y$. The goal is then to learn the underlying function g with the support of a given set of training data $\{(\mathbf{x}_i, y_i)_{i \in \mathbf{I}}\} \subset \mathcal{D}'$ with $\mathcal{D}'$ as the distribution that training data draws from. The difference between interpolation task and extrapolation task then lies in the following different conditions:

- Interpolation: $\mathcal{D} = \mathcal{D}'$
- Extrapolation: $\mathcal{D} \supset \mathcal{D}'$

Then, for a model $f : \mathcal{X} \to \mathrm{R}$ trained on the training dataset, the extrapolation error of f follows the definition in [40] as:

$$\mathrm{E}_{x \sim \mathcal{P}}[l(f(x), g(x))]$$

CS-249-GNN, 2021 Winter, UCLA, Los Angeles, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

with $\mathcal{P} = \mathcal{X} \setminus \mathcal{D}$ as the distribution outside the training distribution and l the loss function.

2.2 Graph Neural Networks

Let a graph $G = (V, E)$ with each node $v \in V$ and link $(u, v) \in E$ and let its node feature vectors be X_v. A general framework for GNN architecture commonly computes a node representation by aggregating the representations of its neighboring nodes and combining the aggregation with the original representation of that node. Formally as in [38], for a given node $v \in V$, the computation for the node representation at the k-th layer of a GNN can be boiled down to

$$h_v^k = COMBINE^{(k)}(h_v^{(k-1)}, AGGREGATE^{(k)}(\{h_u^{k-1} : u \in \mathcal{N}(v)\}))$$

with h_v^k as the representation vector of node v at layer k and $\mathcal{N}(v)$ as the set of nodes that are adjacent to v. Different GCN models would then have a different selection for $COMBINE()$ and $AGGREGATE()$ functions.

For node classification tasks where $\mathcal{X}$ is the node feature vector space, the node representation vectors at the final layers could be directly used for prediction and the learned mapping f would be

$$f(x_v) = MLP^{(k+1)}(h_v^k)$$

For graph classification, $\mathcal{X}$ is then the graph feature vector space and the node representation vectors at the final layers would be aggregated to obtain the entire graph's representation for prediction:

$$f(x_G) = MLP^{(k+1)}(READOUT(\{h_v^k : v \in G\}))$$

with $READOUT$ as an aggregating function chosen by the particular GNN architecture.

3 RELATED WORKS

3.1 Systematic Generalization

The central question in systematic generalization is: How well can a learning agent perform in unseen scenarios given limited exposure to the underlying configurations [13]? This question is also connected to the Language of Thought Hypothesis [9]: The systematicity, productivity, and inferential coherence characterize compositional generalization of concepts [24]. As a prevailing property of human cognition, systematicity poses a central argument against connectionist models [10]. Recently, there have been several works to explore the systematic generalization of deep neural networks in different tasks [1, 12, 20, 23, 35]. For example, [23] study systematic generalization in a synthetic translation task and reveal the incompetence of modern recurrent neural networks [7, 16] in generalizing to unseen scenarios with systematic differences, including unseen primitives, unseen compositions, and longer sequences.

3.2 Extrapolation of Graph Neural Networks

GNNs have been shown to extrapolate well in some non-linear algorithmic tasks, such as intuitive physics [3, 29, 40] and graph algorithms [2, 34], where the testing graphs are larger than training graphs. While dynamic programming [5] has been shown to be capable of solving such extrapolation tasks, a similar computation structure is observed in GNNs [39]. Following this observation, the authors of [40] proved theoretically why GNNs are successful in extrapolating algorithmic tasks and investigated empirically the extrapolation ability of GNNs on some simple tasks including max degree and shortest path problems. Our project follows the study of [40] and introduce more evaluation tasks to comprehensively and specifically study the extrapolation ability of different GNN architectures.

3.3 Generalization of Graph Neural Networks

Many works study the generalization ability of GNNs to unseen data of the given data distribution that training data draws from and tries to improve the performance of GNNs under different settings.

Pre-training [17, 18, 33] draws large amount of unlabeled data to improve the generalization ability of GNNs on the test data. It helps GNNs capture the structural and semantic properties of a input graph, so that they can easily generalize to any downstream tasks on the graphs within the same domain. Domain adaptaion and transfer learning of GNNs [26, 43] focus on the generalization of trained GNN to a specific target domain using the targer domain distributions. GNN meta-learning [6, 19, 28, 41] then focus on schemes of training GNNs to quickly learn never-before-seen labels and relations using only a handful of labeled data points. And robustness of GNNs [8, 30, 37, 44] target the ability of GNNs to generalize well to small adversarial pertubations on the original data distribution.

4 EXPERIMENTS AND RESULTS

4.1 Max Degree and Shorted Path

In this section, we expand the empirical experiments in [40] to study how the model architecture can affect a GNN's extrapolation ability.

4.1.1 *Task Definition.*

Max Degree. We consider the task of finding the maximum degree on a graph. Given any input graph $G = (V, E)$, the label is computed by the underlying function

$$y = g*(G) = \max_{u \in G} \sum_{v \in N(u)} 1. \tag{1}$$

Shortest Path. We consider the task of finding the length of the shortest path on a graph, from a given source to target nodes. Given any graph $G = (V, E)$, the node features, besides regular node features, encode whether a node is source s, and whether a node is target t. The edge features are a scalar representing the edge weight. For unweighted graphs, all edge weights are 1. Then the label $y = g(G)$ is the length of the shortest path from s to t on G.

Shortest path can be solved by the Bellman-Ford(BF) algorithm [4] with the following update:

$$d[k][u] = \min_{v \in N(u)} d[k-1][v] + w(v, u), \tag{2}$$

where $w(v, u)$ is the weight of edge (v, u), and $d[k][u]$ is the shortest distance to node u within k steps.

4.1.2 *Training and Evaluation.*

For the training, validation and test sets, we consider the general graphs with continuous features and use the networkx [14] library to generate graphs.

For the **max degree** task, we perform the extrapolation on the number of nodes and node features.

- The number of vertices of graphs $|V|$ for training and validation sets are sampled uniformly from $[20...30]$. The number of vertices of graphs $|V|$ for test set is sampled uniformly from $[50..100]$.
- Node features in training and validation sets are sampled uniformly from $[-5.0, 5.0]^3$. For test sets, we extrapolate node features by sampling node features uniformly from $[-10.0, 10.0]^3$.

For the **shortest path** task, we perform the extrapolation on the number of nodes and edge weights.

- The number of vertices of graphs $|V|$ for training and validation sets are sampled uniformly from $[20...40]$. The number of vertices of graphs $|V|$for test set is sampled uniformly from $[50..70]$.
- Edge weights for training and validation graphs are sampled from $[1.0, 5.0]$. For test sets, we extrapolate the edge weights by sampling the edge weights uniformly from $[1.0, 10.0]$.
- All node features are $[h, I(v = s), I(v = t)]$ with h sampled from $[-5.0, 5.0]$.

While [40] only considers two combinations of the graph-level pooling (READOUT) and neighbor-level pooling (AGGREGATE) functions for GNNs on each task (underlined in Table 1 and Table 2), we want to fully explore and study the extrapolation performance of GNNs that apply different pooling functions, including the max-, sum-, mean- and min-pooling.

4.1.3 Results. In Table 1 and Table 2, we report the mean absolute percentage error (MAPE) loss for the two tasks, max degree and shortest path, across different AGGREGATE functions and READOUT functions for interpolation and extrapolation. The experiment settings appear in the previous work [40] are underlined in the tables, and the settings with the best performance are highlighted in the bold font.

For the **max degree** task, according to Equation 1, we expect using the graph-level max-pooling and neighbor-level sum-pooling can help GNNs better extrapolate on this task. This assumption is confirmed by the empirical results shown in Table 1. Our results are consistent with the conclusion drawn in [40] that

(1) GNNs with sum-aggregation and sum-readout do not extrapolate well in Max Degree.
(2) If we can encode appropriate non-linearities into the model architecture, the resulting neural network can extrapolate well.

For the **shortest path** task, the observations are a little different. Similarly, according to Equation 2, we expect using the min-pooling for both the READOUT and AGGREGATE can help GNNs better extrapolate on this task. Our results in Table 2 show that this architecture can indeed work well comparing to other combinations of pooling functions.

However, we also notice that GNNs with min-aggregation and max-readout can also achieve comparable or even better performance. [40] didn't consider this architecture as the max-readout is a counter-intuitive choice for the shortest path problem. If we compare the performance of the max-readout and min-readout in general, we can still see that their results are quite close.

To conclude our findings about how the architecture of a GNN affects its extrapolation, according to our empirical results, there isn't a universal best pooling function that works well in every task. GNNs can better extrapolate if we can encode the appropriate non-linearity. However, doing so often requires domain expertise or model search. This requirement suggests that extrapolation is harder in general.

Table 1: The performance comparison of different graph-level and neighbor-level pooling functions for the max degree problem. Settings that are included in [40] are underlined.

Graph (READOUT)	Neighbor (AGGREGATE)	MAPE	
		Interpolate	Extrapolate
Max-pooling	Max-pooling	12.48	51.85
	Sum-pooling	**0.16**	**1.09**
	Mean-pooling	13.77	17.99
	Min-pooling	11.47	16.76
Sum-pooling	Max-pooling	10.59	107.54
	Sum-pooling	10.64	159.94
	Mean-pooling	47.74	58.08
	Min-pooling	46.95	69.00
Mean-pooling	Max-pooling	7.65	15.08
	Sum-pooling	8.71	12.74
	Mean-pooling	47.77	49.97
	Min-pooling	8.74	36.03
Min-pooling	Max-pooling	10.49	25.50
	Sum-pooling	9.97	14.83
	Mean-pooling	11.37	35.66
	Min-pooling	19.45	35.72

Table 2: The performance comparison of different graph-level and neighbor-level pooling functions for the shortest path problem. Settings that are included in [40] are underlined.

Graph (READOUT)	Neighbor (AGGREGATE)	MAPE	
		Interpolate	Extrapolate
Max-pooling	Max-pooling	25.37	142.15
	Sum-pooling	25.83	5209.29
	Mean-pooling	25.53	38.01
	Min-pooling	**25.55**	**36.64**
Sum-pooling	Max-pooling	54.84	81.43
	Sum-pooling	54.64	77.86
	Mean-pooling	54.77	80.74
	Min-pooling	54.85	80.95
Mean-pooling	Max-pooling	27.69	846.99
	Sum-pooling	55.57	23516.80
	Mean-pooling	27.33	183.37
	Min-pooling	28.23	324.95
Min-pooling	Max-pooling	25.43	59.82
	Sum-pooling	25.38	3982.14
	Mean-pooling	25.49	136.12
	Min-pooling	25.63	37.51

4.2 N-body Problem

4.2.1 Task Definition. In an n-body system as shown in Figure 1, there are n objects that exerted gravitational forces on each other, which were a function of the objects' pairwise distances and masses, $F_{ij} = \frac{Gm_i m_j (x_i - x_j)}{||x_i - x_j||^3}$ from object i to j. Given the initial states of the objects, including mass, coordinates, velocity, etc, the model is expected to predict the objects' state of the next time step.

4.2.2 Training and Evaluation. We consider two extrapolation schemes.

(1) The distances between stars are out-of-distribution for the test set.
We ensure all pairwise distances of stars in a frame are between 1 to 20 meters, but distance in the training set is greater than 30m
(2) The masses of stars are out-of-distribution for the test set.
We set the mass of the center star is 200kg for testing and 100kg for training. The masses of others stars are sampled between $[0.04, 18]kg$ for testing and $[0.02, 9]kg$ for training

4.2.3 Models. The object features including the mass, current position and velocity of the star and the relation attributes are concatenated and fed into a relation-centric function. The output of the function shows the effects of the relations between objects. And the output effects are combined with the object features and the external force and fed into an object-centric function. Detailed illustration of the framework can be found in [3].

4.2.4 Input representation. The node features include object mass, position and velocity of stars. For edge features, we consider two kinds of representations.

(1) $w_{ij} = 0$. Since only the gravitational force will be considered, the edge attributes are an all-zero vector with dimension $1 \times n(n-1)$. This is the initial edge attribute of the interaction networks.
(2) $w_{ij} = \frac{m_j}{||x_i^t - x_j^t||_2^3} \cdot (x_j^t - x_i^t)$. Given the input $m_i, m_j, x_i^t, x_j^t, v_i^t$, we can see the physical laws below that the interaction network needs to learn are non-linear. Therefore, we consider putting the explicit form of gravitational force as the edge input, which can help the MLP only learn linear relations so as to help the GNN do better extrapolation.

$$F_i^t = G \sum_{j \neq i} \frac{m_j \times m_i}{||x_i^t - x_j^t||_2^3} \cdot (x_j^t - x_i^t)$$

$$a_i^t = F_i^t / m_i$$

$$v_i^{t+1} = v_i^t + a_i^t \cdot dt$$

4.2.5 Results. We use Mean Absolute Percentage Error(MAPE) to evaluate the model performance. As we can see in table 3, the input with updated edge attributes can have better performance.

Table 3: The performance comparison of different edge input.

Input	MAPE		
	Extra Dist	Extra Mass	Interpolate
$w_{ij} = 0$	8.776	6.632	1.938
$w_{ij} = \frac{m_j}{||x_i^t - x_j^t||_2^3} \cdot (x_j^t - x_i^t)$	3.448	3.600	1.274

4.3 Handwritten arithmetic with INTegers

Inspired by humans' remarkable ability to master arithmetic and generalize to unseen problems, we present a new dataset, Handwritten arithmetic with INTegers (HINT), to study the generalization capability of graph neural networks at three different levels: *perception*, *syntax*, and *semantics*, as exemplified by Fig. 2.

4.3.1 Task Definition. The task of HINT is intuitive and straightforward: It is tasked to predict the final results of handwritten arithmetic expressions in a weakly-supervised manner. Only the final results are given as supervision; all intermediate values and representations are latent, including symbolic expressions, parse trees, and execution traces.

4.3.2 Data Generation. The data generation process follows three steps; see Fig. 3 for an illustration. First, we extract handwritten images from CROHME[1] to obtain primitive concepts, including digits $0 \sim 9$, operators $+, -, \times, \div$, and parentheses $(,)$. Second, we randomly sample *prefix* expressions and convert them to *infix* expressions with necessary parentheses based on the operator precedence; we only allow single-digit numbers in expressions. These symbolic expressions are fed into a solver to calculate the final results. Third, we randomly sample handwritten images for symbols in an expression and concatenate them to construct final handwritten expressions. We only keep the handwritten expressions as input and the corresponding final results as supervision; all intermediate results are discarded.

4.3.3 Train and Evaluation. To rigorously evaluate how well the learned concepts are systematically generalized, we replace the typical i.i.d. train/test split with a carefully designed evaluation scheme: (i) all handwritten images in the test set are unseen in training, (ii) at most 1,000 samples are generated for each number of operators in expressions, (iii) limit the maximum number of operators to 10 and the maximum values to 100 in the training set:

$$D_{train} \subset \mathcal{D}_{train} = \{(x, y) : |x| \leq 10, \max(v) \leq 100\}, \tag{3}$$

where x is the handwritten expression, $|x|$ its number of operators, y the final result, and v all the intermediate values generated when calculating the final result.

We carefully devise the test set to evaluate different generalization capabilities (*i.e.*, interpolation and extrapolation) on different levels of meanings (*i.e.*, perception, syntax and semantics). Specifically, the test set is composed of five subsets, formally defined as:

$$\begin{aligned}
&D_{test} = D_{test}^{(1)} \cup D_{test}^{(2)} \cup D_{test}^{(3)} \cup D_{test}^{(4)} \cup D_{test}^{(5)}, \text{where} \\
&D_{test}^{(1)} = D_{train}, \\
&D_{test}^{(2)} \subset \mathcal{D}_{train} \setminus D_{train}, \\
&D_{test}^{(3)} \subset \{(x, y) : |x| \leq 10, \max(v) > 100\}, \\
&D_{test}^{(4)} \subset \{(x, y) : |x| > 10, \max(v) \leq 100\}, \\
&D_{test}^{(5)} \subset \{(x, y) : |x| > 10, \max(v) > 100\}.
\end{aligned} \tag{4}$$

All above subsets requires generalization on perception of learned concepts. $D_{test}^{(1)}$ requires no generalization on either syntax or semantics, $D_{test}^{(2)}$ requires interpolation on both syntax and semantics,

[1] https://www.cs.rit.edu/~crohme2019/

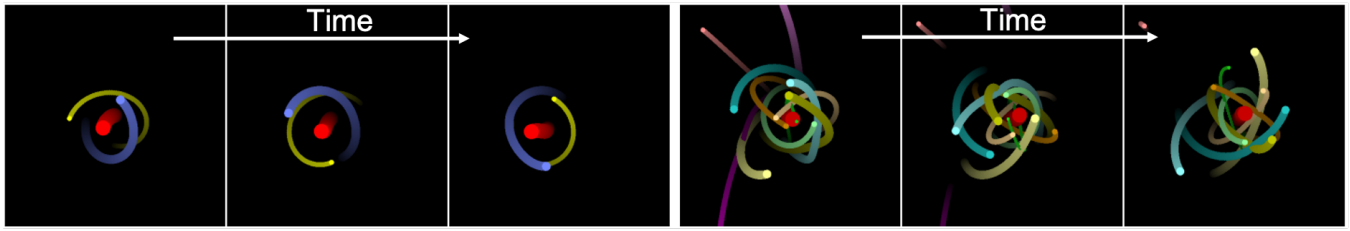


Figure 1: Example 3-body and 12-body systems simulated through time

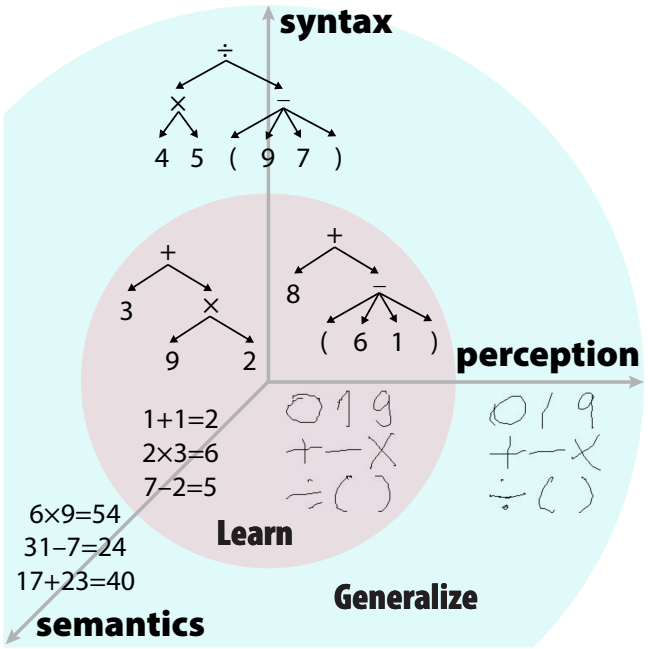


Figure 2: Learning and generalization at three different levels. A learning agent needs to simultaneously master (i) *perception*, how concepts are perceived from raw signals such as images, (ii) *syntax*, how multiple concepts are structurally combined to form a valid expression, and (iii) *semantics*, how concepts are realized to afford various reasoning tasks.

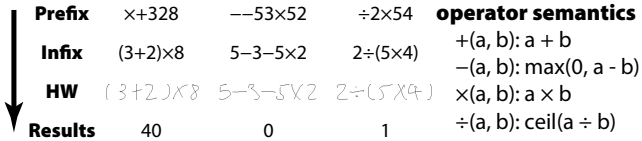


Figure 3: Illustrations of the data generation pipeline.

$D^{(3)}_{test}$ requires interpolation on syntax and extrapolation on semantics, $D^{(4)}_{test}$ requires extrapolation on syntax and interpolation on semantics, and $D^{(5)}_{test}$ requires extrapolation on both syntax and semantics.

In total, the training and test set includes 11,170 and 48,910 samples, respectively. Subsets in the test set are balanced to be 23%, 23%, 22%, 16%, and 16%. Fig. 4 visualizes several randomly selected examples from the proposed HINT dataset.

4.3.4 Models. In the GNN model, the task of HINT is formulated as a graph-to-sequence problem. The input is the dependency graph of an handwritten expression, which is generated by the shunting-yard algorithm[2], and the output is a sequence of digits, which is then converted to an integer as the predicted result. We adopt a graph neural network with 6 layers and each layer has 128 hidden units. We use the max-pooling as the graph and neighbor aggregation function.

We also build a baseline model using LSTM, formulating the task as a sequence-to-sequence problem: The input is an expression sequence, and the output is a sequence of digits. In the LSTM model, the encoder is a bi-directional LSTM [16] with three layers, and the decoder is a one-layer LSTM.

Before being fed into these models, the handwritten expressions are processed by a standard ResNet-18 [15] as the perception module. Since learning such an neural network from scratch is prohibitively challenging, the ResNet-18 is pre-trained by a unsupervised clustering method [31] on unlabeled handwritten images. We test models with varied numbers of layers and report ones with the best results. To speed up the convergence, we train all models with a simple curriculum from short expressions to long ones:

(1) Epoch 0 ∼ 20: max length = 3
(2) Epoch 20 ∼ 40: max length = 7
(3) Epoch 40 ∼ 60: max length = 11
(4) Epoch 60 ∼ 80: max length = 15
(5) Epoch 80 ∼ 100: max length = ∞

4.3.5 Results. We compare the performance of the GNN and LSTM models on HINT. As shown in Table 4, both GNN and LSTM obtain high accuracy on the test subset 1, which indicates that they can generalize over perception very well. In the test subsets 2 and 4, which requires interpolation and extrapolation over syntax, the GNN model outperforms the LSTM model by a large margin. However, their performances drop significantly on the test subsets 3 and 5, which require systematic generalization over semantics. Notably, when using image input, their accuracy is less than 10% on test subsets 3 and 5 that involve larger numbers compared to the training set. This result indicates that the pure neural models do not learn the semantics of concepts in a generalizable way and fail to extrapolate to large numbers. Fig. 5 shows several examples of the GNN predictions on each test subset.

4.3.6 How do models extrapolate? Among the generalization capability, we are particularly interested in extrapolation. Based on the experimental results, we believe that the key of successful extrapolation is *recursion*. Although the graph structure in the hidden cells of the GNN model serves as a recursive prior on syntax, no such prior in its representation for semantics. This deficiency explains

[2]https://en.wikipedia.org/wiki/Shunting-yard_algorithm

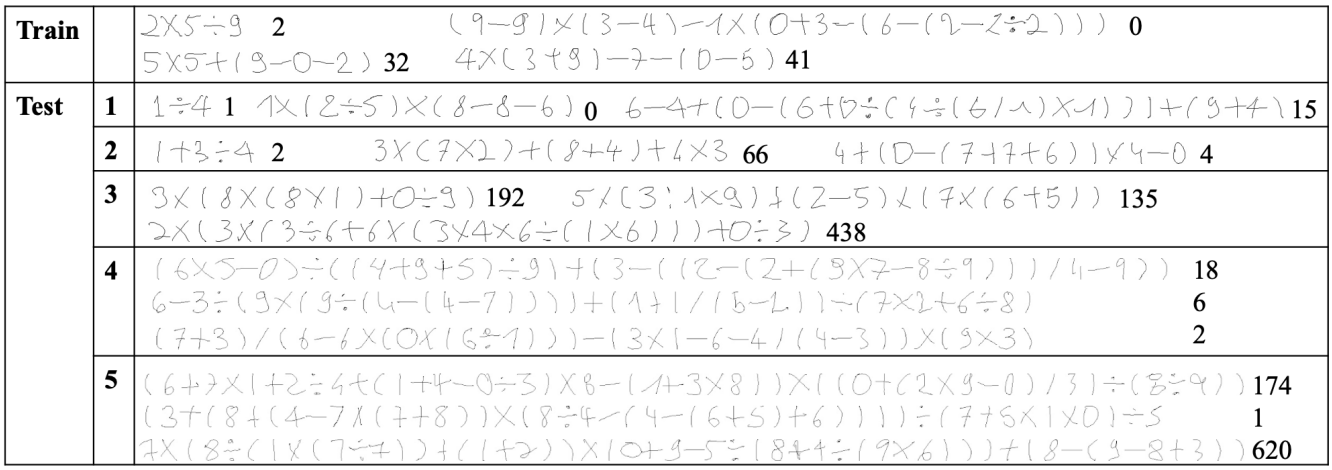


Figure 4: Randomly selected examples from the training set and each subset of the test set.

Table 4: The performance comparison of GNN and LSTM.

Input	Model	Test Accuracy (%)					
		Overall	1	2	3	4	5
Symbol (Embedding)	GNN	49.71	97.05	63.67	11.58	52.41	12.57
	LSTM	34.58	98.31	29.79	2.91	26.39	2.76
Image (ResNet-18)	GNN	39.39	87.02	46.17	6.51	40.44	6.47
	LSTM	32.95	87.31	30.74	2.67	31.17	2.55

why GNN would achieve a decent accuracy (40.44%) on the test subset 3 (extrapolation only on syntax) but a much lower accuracy (6.51%) on the test subset 4 (extrapolation only on semantics). Taken together, these observations strongly imply that the recursive prior on task-specific representations is the crux of extrapolation, which is also in line with the recent analysis of graph neural networks, which successfully extrapolate algorithmic tasks due to the *task-specific non-linearities* in the architecture or features [39, 40].

5 CONCLUSION

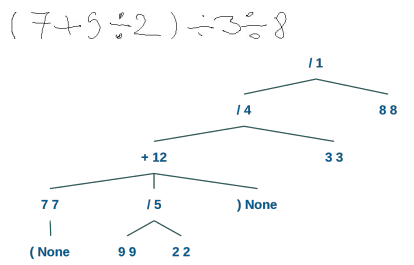
In this project, we study the extrapolation ability of graph neural networks instead of it generalization (interpolation) ability. We follow the work [40] and expands its empirical analysis by introducing our new evaluation tasks. With a more comprehensive investigation and analysis, we provide a deeper insight into the power and limitations of GNN in reasoning tasks requiring extrapolation. We hope our project can broadly inspire more future studies on the extrapolation power of neural networks beyond GNNs.

6 TASK DISTRIBUTION FORM

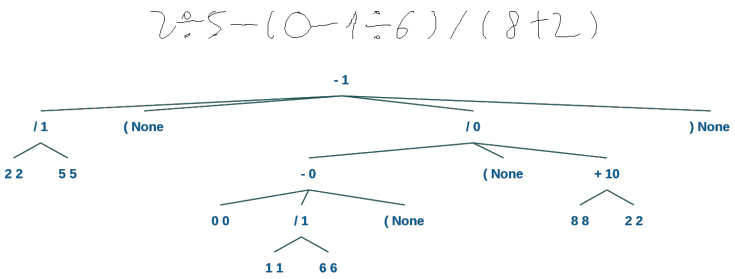
Task	People
Idea proposal	Qing Li
Formulation and discussion	Everyone
Experiments (max-degree and shortest path)	Yu Yang
Experiments (n-body problem)	Shuwen Qiu
Experiments (HINT)	Qing Li
Writing report (section 1-3)	Yihe Deng
Revising report	Everyone

Test subset 1

GT: (7+9/2)/3/8 = 1 PD: (7+9/2)/3/8 = 1

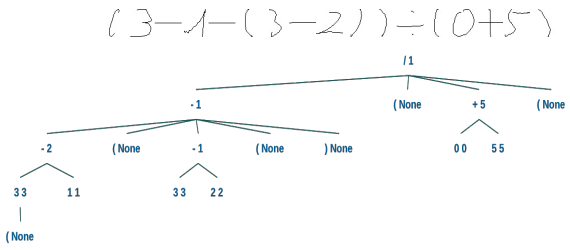


GT: 2/5-(0-1/6)/(8+2) = 1 PD: 2/5-(0-1/6(/(8+2) = 1

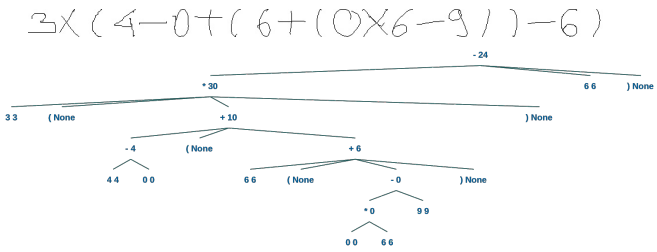


Test subset 2

GT: (3-1-(3-2))/(0+5) = 1 PD: (3-1-(3-2()/(0+5(= 1

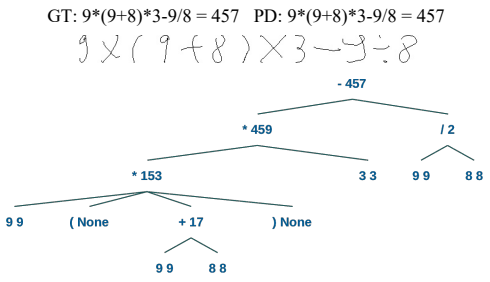


GT: 3*(4-0+(6+(0*6-9))-6) = 12 PD: 3*(4-0+(6+(0*6-9))-6) = 24

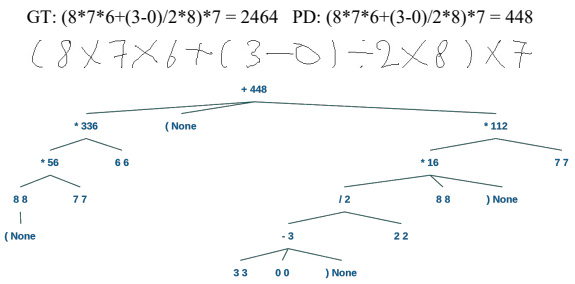


Test subset 3

GT: 9*(9+8)*3-9/8 = 457 PD: 9*(9+8)*3-9/8 = 457

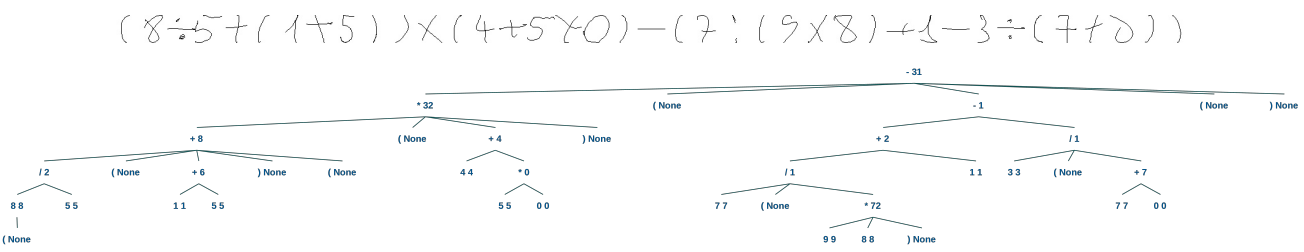


GT: (8*7*6+(3-0)/2*8)*7 = 2464 PD: (8*7*6+(3-0)/2*8)*7 = 448



Test subset 4

GT: (8/5+(1+5))*(4+5*0)-(7/(9*8)+1-3/(7+0)) = 31 PD: (8/5+(1+5)(*(4+5*0)-(7/(9*8)+1-3/(7+0() = 31



Test subset 5

GT: (8*7-5/5)*(3-(2-1)+1)/(9*1*(8+1)+(9+3)-0) = 2 PD: (8*7-5/5)*(3-(2-1)+1)/(9*1*(8+1)/(9+3)-0) = 24

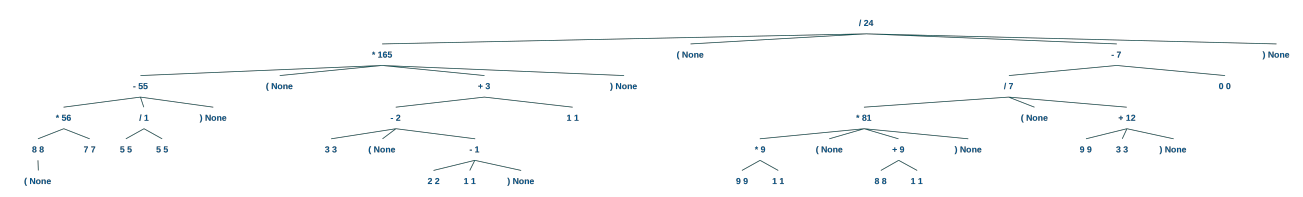


Figure 5: Examples of the GNN predictions on the test set. "GT" and "PD" denote "ground-truth" and "prediction," respectively. Each node in the solution tree is a tuple of (symbol, value).

REFERENCES

[1] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: what is required and can it be learned? 2018.

[2] Peter Battaglia, Jessica Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[3] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 4502–4510. Curran Associates, Inc., 2016.

[4] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[5] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[6] Jatin Chauhan, Deepak Nathani, and Manohar Kaul. Few-shot learning on graphs via super-classes based on graph spectral measures. In *International Conference on Learning Representations*, 2020.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[8] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1115–1124, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[9] Jerry A Fodor. *The language of thought*, volume 5. Harvard university press, 1975.

[10] Jerry A Fodor, Zenon W Pylyshyn, et al. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

[11] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3419–3430. PMLR, 13–18 Jul 2020.

[12] Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. Permutation equivariant models for compositional generalization in language. 2019.

[13] Ulf Grenander. *General pattern theory-A mathematical study of regular structures*. Clarendon Press, 1993.

[14] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2016.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

[18] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867, 2020.

[19] Kexin Huang and Marinka Zitnik. Graph meta learning via local subgraphs. *Advances in Neural Information Processing Systems*, 33, 2020.

[20] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. 2019.

[21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[22] Boris Knyazev, Graham Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. 10 2019.

[23] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. 2018.

[24] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[25] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

[26] Jaekoo Lee, Hyunjae Kim, Jongsun Lee, and Sungroh Yoon. Transfer learning for deep learning on graph-structured data. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2154–2160. AAAI Press, 2017.

[27] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, April 2016.

[28] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning to propagate for graph meta-learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[29] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018.

[30] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S. Yu, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

[31] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. 2020.

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster.

[33] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019.

[34] Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *International Conference on Learning Representations*, 2020.

[35] Sirui Xie, Xiaojian Ma, Peiyu Yu, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Halma: Humanlike abstraction learning meets affordance in rapid problem solving. *arXiv preprint arXiv:2102.11344*, 2021.

[36] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3961–3967. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[37] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[39] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020.

[40] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon Shaolei Du, Ken-Ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *International Conference on Learning Representations*, 2021.

[41] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh Chawla, and Zhenhui Li. Graph few-shot learning via knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6656–6663, 2020.

[42] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1399–1407, 2019.

[43] Qi Zhu, Yidan Xu, Haonan Wang, Chao Zhang, Jiawei Han, and Carl Yang. Transfer learning of graph neural networks with ego-graph information maximization, 2021.

[44] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.