The Surprising Power of Graph Neural Networks with Random Node Initialization

Ralph Abboud[1] , **İsmail İlkan Ceylan**[1] , **Martin Grohe**[2] and **Thomas Lukasiewicz**[1]
[1]University of Oxford
[2]RWTH Aachen University

Abstract

Graph neural networks (GNNs) are effective models for representation learning on relational data. However, standard GNNs are limited in their expressive power, as they cannot distinguish graphs beyond the capability of the Weisfeiler-Leman graph isomorphism heuristic. In order to break this expressiveness barrier, GNNs have been enhanced with *random node initialization (RNI)*, where the idea is to train and run the models with randomized initial node features. In this work, we analyze the expressive power of GNNs with RNI, and prove that these models are *universal*, a first such result for GNNs not relying on computationally demanding higher-order properties. This universality result holds even with partially randomized initial node features, and preserves the *invariance* properties of GNNs in expectation. We then empirically analyze the effect of RNI on GNNs, based on carefully constructed datasets. Our empirical findings support the superior performance of GNNs with RNI over standard GNNs.

1 Introduction

Graph neural networks (GNNs) [Scarselli *et al.*, 2009; Gori *et al.*, 2005] are neural architectures designed for learning functions over graph domains, and naturally encode desirable properties such as permutation invariance (resp., equivariance) relative to graph nodes, and node-level computation based on message passing. These properties provide GNNs with a strong inductive bias, enabling them to effectively learn and combine both local and global graph features [Battaglia *et al.*, 2018]. GNNs have been applied to a multitude of tasks, ranging from protein classification [Gilmer *et al.*, 2017] and synthesis [You *et al.*, 2018], protein-protein interaction [Fout *et al.*, 2017], and social network analysis [Hamilton *et al.*, 2017], to recommender systems [Ying *et al.*, 2018] and combinatorial optimization [Bengio *et al.*, 2021].

While being widely applied, popular GNN architectures, such as message passing neural networks (MPNNs), are limited in their expressive power. Specifically, MPNNs are at most as powerful as the Weisfeiler-Leman (1-WL) graph isomorphism heuristic [Morris *et al.*, 2019; Xu *et al.*, 2019], and thus cannot discern between several families of non-isomorphic graphs, e.g., sets of regular graphs [Cai *et al.*, 1992]. To address this limitation, alternative GNN architectures with provably higher expressive power, such as k-GNNs [Morris *et al.*, 2019] and invariant (resp., equivariant) graph networks [Maron *et al.*, 2019b], have been proposed. These models, which we refer to as *higher-order GNNs*, are inspired by the generalization of 1-WL to $k-$tuples of nodes, known as k-WL [Cai *et al.*, 1992]. While these models are very expressive, they are computationally very demanding. As a result, MPNNs, despite their limited expressiveness, remain the standard for graph representation learning.

In a rather recent development, MPNNs have achieved empirical improvements using *random node initialization* (RNI), in which initial node embeddings are randomly set. Indeed, RNI enables MPNNs to detect *fixed* substructures, so extends their power beyond 1-WL, and also allows for a better approximation of a class of combinatorial problems [Sato *et al.*, 2021]. While very important, these findings do not explain the overall theoretical impact of RNI on GNN learning and generalization for *arbitrary* functions.

In this paper, we thoroughly study the impact of RNI on MPNNs. Our main result states that MPNNs enhanced with RNI are *universal*, and thus can approximate every function defined on graphs of any fixed order. This follows from a logical characterization of the expressiveness of MPNNs [Barceló *et al.*, 2020] combined with an argument on order-invariant definability. Importantly, MPNNs enhanced with RNI preserve the *permutation-invariance* of MPNNs in expectation, and possess a strong inductive bias. Our result strongly contrasts with 1-WL limitations of deterministic MPNNs, and provides a foundation for developing expressive and memory-efficient MPNNs with strong inductive bias.

To verify our theoretical findings, we carry out a careful empirical study. We design EXP, a synthetic dataset requiring 2-WL expressive power for models to achieve above-random performance, and run MPNNs with RNI on it, to observe *how well* and *how easily* this model can learn and generalize. Then, we propose CEXP, a modification of EXP with partially 1-WL distinguishable data, and evaluate the same questions in this more variable setting. Overall, the contributions of this paper are as follows:

- We prove that MPNNs with RNI are universal, while being permutation-invariant in expectation. This is a significant improvement over the 1-WL limit of standard MPNNs and, to our knowledge, a first universality result for memory-efficient GNNs.

- We introduce two carefully designed datasets, EXP and CEXP, based on graph pairs only distinguishable by 2-WL or higher, to rigorously evaluate the impact of RNI.
- We analyze the effects of RNI on MPNNs on these datasets, and observe that (i) MPNNs with RNI closely match the performance of higher-order GNNs, (ii) the improved performance of MPNNs with RNI comes at the cost of slower convergence, and (iii) partially randomizing initial node features improves model convergence and accuracy.
- We additionally perform the same experiments with analog sparser datasets, with longer training, and observe a similar behavior, but more volatility.

The proof of the main theorem, as well as further details on datasets and experiments, can be found in the long version of this paper: http://www.arxiv.org/abs/2010.01179.

2 Graph Neural Networks

Graph neural networks (GNNs) [Gori *et al.*, 2005; Scarselli *et al.*, 2009] are neural models for learning functions over graph-structured data. In a GNN, graph nodes are assigned vector representations, which are updated iteratively through series of *invariant* or *equivariant* computational layers. Formally, a function f is *invariant* over graphs if, for isomorphic graphs $G, H \in \mathcal{G}$ it holds that $f(G) = f(H)$. Furthermore, a function f mapping a graph G with vertices $V(G)$ to vectors $\boldsymbol{x} \in \mathbb{R}^{|V(G)|}$ is *equivariant* if, for every permutation π of $V(G)$, it holds that $f(G^{\pi}) = f(G)^{\pi}$.

2.1 Message Passing Neural Networks

In MPNNs [Gilmer *et al.*, 2017], node representations aggregate *messages* from their neighboring nodes, and use this information to iteratively update their representations. Formally, given a node x, its vector representation $v_{x,t}$ at time t, and its neighborhood $N(x)$, an update can be written as:

$$v_{x,t+1} = combine\Big(v_{x,t}, aggregate\big(\{v_{y,t} |\, y \in N(x)\}\big)\Big),$$

where *combine* and *aggregate* are functions, and *aggregate* is typically permutation-invariant. Once message passing is complete, the final node representations are then used to compute target outputs. Prominent MPNNs include graph convolutional networks (GCNs) [Kipf and Welling, 2017] and graph attention networks (GATs) [Velickovic *et al.*, 2018].

It is known that standard MPNNs have the same power as the 1-dimensional Weisfeiler-Leman algorithm (1-WL) [Xu *et al.*, 2019; Morris *et al.*, 2019]. This entails that graphs (or nodes) cannot be distinguished by MPNNs if 1-WL does not distinguish them. For instance, 1-WL cannot distinguish between the graphs G and H, shown in Figure 1, despite them being clearly non-isomorphic. Therefore, MPNNs cannot learn functions with different outputs for G and H.

Another somewhat trivial limitation in the expressiveness of MPNNs is that information is only propagated along edges, and hence can never be shared between distinct connected components of a graph [Barceló *et al.*, 2020; Xu *et al.*, 2019]. An easy way to overcome this limitation is by adding *global readouts*, that is, permutation-invariant functions that aggregate the current states of all nodes. Throughout the paper, we therefore focus on MPNNs with global readouts, referred to as *ACR-GNNs* [Barceló *et al.*, 2020].

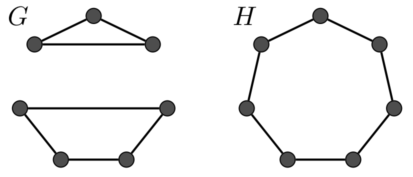


Figure 1: G and H are indistinguishable by 1-WL

2.2 Higher-order Graph Neural Networks

We now present the main classes of higher-order GNNs.

Higher-order MPNNs. The k-WL hierarchy has been directly emulated in GNNs, such that these models learn embeddings for *tuples* of nodes, and perform message passing between them, as opposed to individual nodes. This higher-order message passing approach resulted in models such as k-GNNs [Morris *et al.*, 2019], which have $(k-1)$-WL expressive power.[1] These models need $O(|V|^k)$ memory to run, leading to *excessive memory requirements*.

Invariant (resp., equivariant) graph networks. Another class of higher-order GNNs is invariant (resp., equivariant) graph networks [Maron *et al.*, 2019b], which represent graphs as a tensor, and implicitly pass information between nodes through invariant (resp., equivariant) computational blocks. Following intermediate blocks, *higher-order* tensors are typically returned, and the order of these tensors correlates directly with the expressive power of the overall model. Indeed, invariant networks [Maron *et al.*, 2019c], and later equivariant networks [Keriven and Peyré, 2019], are shown to be universal, but with tensor orders of $O(|V|^2)$, where $|V|$ denotes the number of graph nodes. Furthermore, invariant (resp., equivariant) networks with intermediate tensor order k are shown to be equivalent in power to $(k-1)$-WL [Maron *et al.*, 2019a], which is strictly more expressive as k increases [Cai *et al.*, 1992]. Therefore, universal higher-order models require *intractably-sized intermediate tensors* in practice.

Provably powerful graph networks. A special class of invariant GNNs is provably powerful graph networks (PPGNs)[Maron *et al.*, 2019a]. PPGNs are based on "blocks" of multilayer perceptrons (MLPs) and matrix multiplication, which theoretically have 2-WL expressive power, and only require memory $O(|V|^2)$ (compared to $O(|V|^3)$ for 3-GNNs). However, PPGNs theoretically require *exponentially many samples* in the number of graph nodes to learn necessary functions for 2-WL expressiveness [Puny *et al.*, 2020].

[1]In the literature, different versions of the Weisfeiler-Leman algorithm have inconsistent dimension counts, but are equally expressive. For example, $(k+1)$-WL and $(k+1)$-GNNs in [Morris *et al.*, 2019] are equivalent to k-WL of [Cai *et al.*, 1992; Grohe, 2017]. We follow the latter, as it is the standard in the literature on graph isomorphism testing.

3 MPNNs with Random Node Initialization

We present the main result of the paper, showing that RNI makes MPNNs universal, in a natural sense. Our work is a first positive result for the universality of MPNNs. This result is not based on a new model, but rather on random initialization of node features, which is widely used in practice, and in this respect, it also serves as a theoretical justification for models that are empirically successful.

3.1 Universality and Invariance

It may appear somewhat surprising, and even counter-intuitive, that randomly initializing node features on its own would deliver such a gain in expressiveness. In fact, on the surface, random initialization no longer preserves the invariance of MPNNs, since the result of the computation of an MPNN with RNI not only depends on the structure (i.e., the isomorphism type) of the input graph, but also on the random initialization. The broader picture is, however, rather subtle, as we can view such a model as computing a random variable (or as generating an output distribution), and this random variable would still be invariant. This means that the outcome of the computation of an MPNN with RNI does still *not* depend on the specific representation of the input graph, which fundamentally maintains invariance. Indeed, the mean of random features, in expectation, will inform GNN predictions, and is identical across all nodes, as randomization is i.i.d. However, the variability between different samples and the variability of a random sample enable graph discrimination and improve expressiveness. Hence, in expectation, all samples fluctuate around a unique value, preserving invariance, whereas sample variance improves expressiveness.

Formally, let $\mathcal{G}_n$ be the class of all n-vertex graphs, i.e., graphs that consist of at most n vertices, and let $f : \mathcal{G}_n \to \mathbb{R}$. We say that a randomized function $\mathcal{X}$ that associates with every graph $G \in \mathcal{G}_n$ a random variable $\mathcal{X}(G)$ is an (ϵ, δ)*-approximation* of f if for all $G \in \mathcal{G}_n$ it holds that $\Pr\left(|f(G) - \mathcal{X}(G)| \le \epsilon\right) \ge 1 - \delta$. Note that an MPNN $\mathcal{N}$ with RNI computes such functions $\mathcal{X}$. If $\mathcal{X}$ is computed by $\mathcal{N}$, we say that $\mathcal{N}$ (ϵ, δ)*-approximates* f.

Theorem 1 (Universal approximation)**.** *Let $n \ge 1$, and let $f : \mathcal{G}_n \to \mathbb{R}$ be invariant. Then, for all $\epsilon, \delta > 0$, there is an MPNN with RNI that (ϵ, δ)-approximates f.*

For ease of presentation, we state the theorem only for real-valued functions, but note that it can be extended to equivariant functions. The result can also be extended to weighted graphs, but then the function f needs to be continuous.

3.2 Result Overview

To prove Theorem 1, we first show that MPNNs with RNI can capture arbitrary Boolean functions, by building on the result of [Barceló *et al.*, 2020], which states that any logical sentence in C^2 can be captured by an MPNN (or, by an ACR-GNN in their terminology). The logic C is the extension of first-order predicate logic using counting quantifiers of the form $\exists^{\ge k} x$ for $k \ge 0$, where $\exists^{\ge k} x \varphi(x)$ means that there are at least k elements x satisfying φ, and C^2 is the two-variable fragment of C.

We establish that any graph with identifying node features, which we call *individualized graphs*, can be represented by a sentence in C^2. Then, we extend this result to sets of individualized graphs, and thus to Boolean functions mapping these sets to True, by showing that these functions are represented by a C^2 sentence, namely, the disjunction of all constituent graph sentences. Following this, we provide a construction with node embeddings based on RNI, and show that RNI individualizes input graphs w.h.p. Thus, RNI makes that MPNNs learn a Boolean function over individualized graphs w.h.p. Since all such functions can be captured by a sentence in C^2, and an MPNN can capture any Boolean function, MPNNs with RNI can capture arbitrary Boolean functions. Finally, the result is extended to real-valued functions via a natural mapping, yielding universality.

The concrete implications of Theorem 1 can be summarized as follows. First, MPNNs with RNI can distinguish individual graphs with an embedding dimensionality polynomial in the inverse of desired confidence δ (namely, $O(n^2\delta^{-1})$, where n is the number of graph nodes). Second, universality also holds with partial RNI, and even with only one randomized dimension. Third, the theorem is adaptive and tightly linked to the descriptive complexity of the approximated function. That is, for a more restricted class of functions, there may be more efficient constructions than the disjunction of individualized graph sentences, and our proof does not rely on a particular construction. Finally, our construction provides a *logical characterization*for MPNNs with RNI, and substantiates how randomization improves expressiveness. This construction therefore also enables a more logically grounded theoretical study of randomized MPNN models, based on particular architectural or parametric choices.

Similarly to other universality results, Theorem 1 can potentially result in very large constructions. This is a simple consequence of the generality of such results: Theorem 1 applies to families of functions, describing problems of *arbitrary* computational complexity, including problems that are computationally hard, even to approximate. Thus, it is more relevant to empirically verify the formal statement, and test the capacity of MPNNs with RNI relative to higher-order GNNs. Higher-order GNNs typically suffer from prohibitive space requirements, but this not the case for MPNNs with RNI, and this already makes them more practically viable. In fact, our experiments demonstrate that MPNNs with RNI indeed combine expressiveness with efficiency in practice.

4 Datasets for Expressiveness Evaluation

GNNs are typically evaluated on real-world datasets [Kersting *et al.*, 2016], which are not tailored for evaluating expressive power, as they do not contain instances indistinguishable by 1-WL. In fact, higher-order models only marginally outperform MPNNs on these datasets [Dwivedi *et al.*, 2020], which further highlights their unsuitability. Thus, we developed the synthetic datasets EXP and CEXP. EXP explicitly evaluates GNN expressiveness, and consists of graph instances $\{G_1, \ldots, G_n, H_1, \ldots, H_n\}$, where each instance encodes a propositional formula. The classification task is to determine whether the formula is satisfiable (SAT). Each pair

(G_i, H_i) respects the following properties: (i) G_i and H_i are non-isomorphic, (ii) G_i and H_i have different SAT outcomes, that is, G_i encodes a satisfiable formula, while H_i encodes an unsatisfiable formula, (iii) G_i and H_i are 1-WL indistinguishable, so are *guaranteed* to be classified in the same way by standard MPNNs, and (iv) G_i and H_i are 2-WL distinguishable, so *can* be classified differently by higher-order GNNs.

Fundamentally, every (G_i, H_i) is carefully constructed on top of a basic building block, the *core pair*. In this pair, both cores are based on propositional clauses, such that one core is satisfiable and the other is not, both *exclusively* determine the satisfiability of G_i (resp., H_i), and have graph encodings enabling all aforementioned properties. Core pairs and their resulting graph instances in EXP are *planar* and are also carefully constrained to ensure that they are 2-WL distinguishable. Thus, core pairs are key substructures within EXP, and distinguishing these cores is essential for a good performance.

Building on EXP, CEXP includes instances with varying expressiveness requirements. Specifically, CEXP is a standard EXP dataset where 50% of all satisfiable graph pairs are made 1-WL distinguishable from their unsatisfiable counterparts, only differing from these by a small number of added edges. Hence, CEXP consists of 50% "corrupted" data, distinguishable by MPNNs and labelled CORRUPT, and 50% unmodified data, generated analogously to EXP, and requiring expressive power beyond 1-WL, referred to as $\overline{\text{EXP}}$. Thus, CEXP contains the same core structures as EXP, but these lead to different SAT values in $\overline{\text{EXP}}$ and CORRUPT, which makes the learning task more challenging than learning $\overline{\text{EXP}}$ or CORRUPT in isolation.

5 Experimental Evaluation

In this section, we first evaluate the effect of RNI on MPNN expressiveness based on EXP, and compare against established higher-order GNNs. We then extend our analysis to CEXP. Our experiments use the following models:

1-WL GCN (1-GCN). A GCN with 8 distinct message passing iterations, ELU non-linearities [Clevert *et al.*, 2016], 64-dimensional embeddings, and deterministic learnable initial node embeddings indicating node type. This model is guaranteed to achieve 50% accuracy on EXP.

GCN - Random node initialization (GCN-RNI). A 1-GCN enhanced with RNI. We evaluate this model with four initialization distributions, namely, the standard normal distribution $\mathcal{N}(0,1)$ (N), the uniform distribution over $[-1,1]$ (U), Xavier normal (XN), and the Xavier uniform distribution (XU) [Glorot and Bengio, 2010]. We denote the respective models GCN-RNI(D), where $D \in \{\text{N}, \text{U}, \text{XN}, \text{XU}\}$.

GCN - Partial RNI (GCN-x%RNI). A GCN-RNI model, where $\lfloor \frac{64x}{100} \rfloor$ dimensions are initially randomized, and all remaining dimensions are set deterministically from one-hot representation of the two input node types (literal and disjunction). We set x to the extreme values 0 and 100%, 50%, as well as near-edge cases of 87.5% and 12.5%, respectively.

PPGN. A higher-order GNN with 2-WL expressive power [Maron *et al.*, 2019a]. We set up PPGN using its original implementation, and use its default configuration of eight 400-dimensional computational blocks.

Model	Test Accuracy (%)
GCN-RNI(U)	97.3 ± 2.55
GCN-RNI(N)	**98.0** ± **1.85**
GCN-RNI(XU)	97.0 ± 1.43
GCN-RNI(XN)	96.6 ± 2.20
PPGN	50.0
1-2-3-GCN-L	50.0
3-GCN	**99.7** ± **0.004**

Table 1: Accuracy results on EXP.

1-2-3-GCN-L. A higher-order GNN [Morris *et al.*, 2019] emulating 2-WL on 3-node tuples. 1-2-3-GCN-L operates at increasingly coarse granularity, starting with single nodes and rising to 3-tuples. This model uses a *connected* relaxation of 2-WL, which slightly reduces space requirements, but comes at the cost of some theoretical guarantees. We set up 1-2-3-GCN-L with 64-dimensional embeddings, 3 message passing iterations at level 1, 2 at level 2, and 8 at level 3.

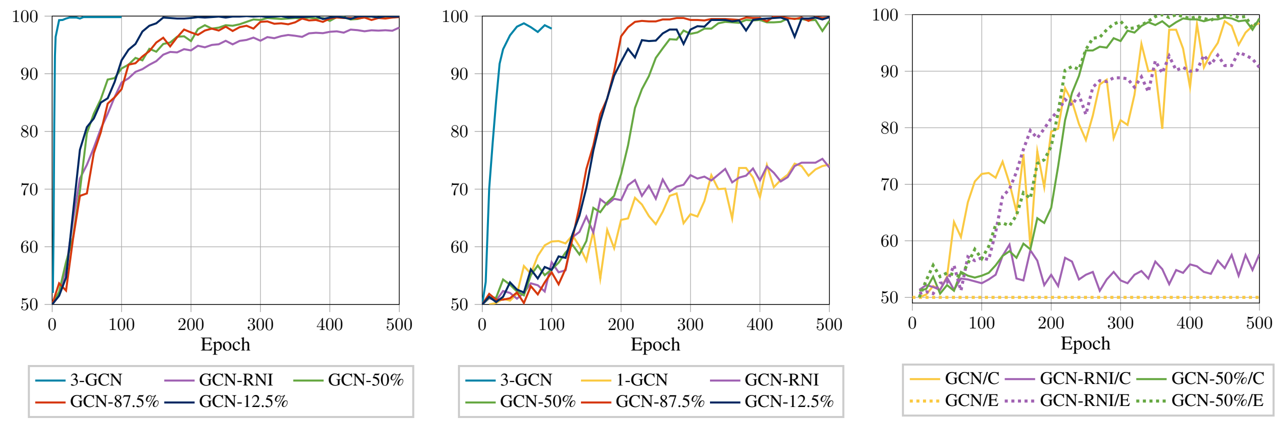
3-GCN. A GCN analog of the *full* 2-WL procedure over 3-node tuples, thus preserving all theoretical guarantees.

5.1 How Does RNI Improve Expressiveness?

In this experiment, we evaluate GCNs using different RNI settings on EXP, and compare with standard GNNs and higher-order models. Specifically, we generate an EXP dataset consisting of 600 graph pairs. Then, we evaluate all models on EXP using 10-fold cross-validation. We train 3-GCN for 100 epochs per fold, and all other systems for 500 epochs, and report *mean test accuracy* across all folds.

Full test accuracy results for all models are reported in Table 1, and model convergence for 3-GCN and all GCN-RNI models are shown in Figure 2a. In line with Theorem 1, GCN-RNI achieves a near-perfect performance on EXP, substantially surpassing 50%. Indeed, GCN-RNI models achieve above 95% accuracy with all four RNI distributions. This finding further supports observations made with rGNNs [Sato *et al.*, 2021], and shows that RNI is also beneficial in settings beyond structure detection. Empirically, we observed that GCN-RNI is highly sensitive to changes in learning rate, activation function, and/or randomization distribution, and required delicate tuning to achieve its best performance.

Surprisingly, PPGN does not achieve a performance above 50%, despite being theoretically 2-WL expressive. Essentially, PPGN learns an approximation of 2-WL, based on power-sum multi-symmetric polynomials (PMP), but fails to distinguish EXP graph pairs, despite extensive training. This suggests that PPGN struggles to learn the required PMPs, and we could not improve these results, both for training and testing, with hyperparameter tuning. Furthermore, PPGN requires exponentially many data samples in the size of the input graph [Puny *et al.*, 2020] for learning. Hence, PPGN is likely struggling to discern between EXP graph pairs due to the smaller sample size and variability of the dataset. 1-2-3-GCN-L also only achieves 50% accuracy, which can be



(a) EXP. (b) CEXP. (c) $\overline{\text{EXP}}$ (/E) and CORRUPT (/C).

Figure 2: Learning curves across all experiments for all models.

attributed to theoretical model limitations. Indeed, this algorithm only considers 3-tuples of nodes that form a connected subgraph, thus discarding disconnected 3-tuples, where the difference between EXP cores lies. This further highlights the difficulty of EXP, as even relaxing 2-WL reduces the model to random performance. Note that 3-GCN achieves near-perfect performance, as it explicitly has the necessary theoretical power, irrespective of learning constraints, and must only learn appropriate injective aggregation functions for neighbor aggregation [Xu *et al.*, 2019].

In terms of convergence, we observe that 3-GCN converges significantly faster than GCN-RNI models, for all randomization percentages. Indeed, 3-GCN only requires about 10 epochs to achieve optimal performance, whereas GCN-RNI models all require over 100 epochs. Intuitively, this slower convergence of GCN-RNI can be attributed to a harder learning task compared to 3-GCN: Whereas 3-GCN learns from deterministic embeddings, and can naturally discern between dataset cores, GCN-RNI relies on RNI to discern between EXP data points, via an artificial node ordering. This implies that GCN-RNI must leverage RNI to detect structure, then subsequently learn robustness against RNI variability, which makes its learning task especially challenging.

Our findings suggest that RNI practically improves MPNN expressiveness, and makes them competitive with higher-order models, despite being less demanding computationally. Indeed, for a 50-node graph, GCN-RNI only requires 3200 parameters (using 64-dimensional embeddings), whereas 3-GCN requires 1,254,400 parameters. Nonetheless, GCN-RNI performs comparably to 3-GCN, and, unlike the latter, can easily scale to larger instances. This increase in expressive power, however, comes at the cost of slower convergence. Even so, RNI proves to be a promising direction for building scalable yet powerful MPNNs.

5.2 How Does RNI Behave on Variable Data?

In the earlier experiment, RNI practically improves the expressive power of GCNs over EXP. However, EXP solely evaluates expressiveness, and this leaves multiple questions open: How does RNI impact learning when data contains instances with varying expressiveness requirements, and how does RNI affect generalization on more variable datasets? We experiment with CEXP to explicitly address these questions.

We generated CEXP by generating another 600 graph pairs, then selecting 300 of these and modifying their satisfiable graph, yielding CORRUPT. CEXP is well-suited for holistically evaluating the efficacy of RNI, as it evaluates the contribution of RNI on $\overline{\text{EXP}}$ conjointly with a second learning task on CORRUPT involving very similar core structures, and assesses the effect of different randomization degrees on overall and subset-specific model performance.

In this experiment, we train GCN-RNI (with varying randomization degrees) and 3-GCN on CEXP, and compare their accuracy. For GCN-RNI, we observe the effect of RNI on learning $\overline{\text{EXP}}$ and CORRUPT, and the interplay between these tasks. In all experiments, we use the normal distribution for RNI, given its strong performance in the earlier experiment.

The learning curves of all GCN-RNI and 3-GCN on CEXP are shown in Figure 2b, and the same curves for the $\overline{\text{EXP}}$ and CORRUPT subsets are shown in Figure 2c. As on EXP, 3-GCN converges very quickly, exceeding 90% test accuracy within 25 epochs on CEXP. By contrast, GCN-RNI, for all randomization levels, converges much slower, around after 200 epochs, despite the small size of input graphs (~70 nodes at most). Furthermore, fully randomized GCN-RNI performs worse than partly randomized GCN-RNI, particularly on CEXP, due to its weak performance on CORRUPT.

First, we observe that partial randomization significantly improves performance. This can clearly be seen on CEXP, where GCN-12.5%RNI and GCN-87.5%RNI achieve the best performance, by far outperforming GCN-RNI, which struggles on CORRUPT. This can be attributed to having a better inductive bias than a fully randomized model. Indeed, GCN-12.5%RNI has mostly deterministic node embeddings, which simplifies learning over CORRUPT. This also applies to GCN-87.5%RNI, where the number of deterministic dimensions, though small, remains sufficient. Both models also benefit from randomization for $\overline{\text{EXP}}$, similarly to a fully randomized GCN. GCN-12.5%RNI and GCN-87.5%RNI effectively achieve the best of both worlds on CExp, leveraging inductive bias from deterministic node embeddings, while harnessing the power of RNI to perform strongly on $\overline{\text{Exp}}$. This is best shown in Figure 2c, where standard GCN fails to learn $\overline{\text{Exp}}$, fully randomized GCN-RNI struggles to learn Corrupt, and the semi-randomized GCN-50%RNI achieves perfect performance on both subsets. We also note that partial RNI, when applied to several real datasets, where 1-WL power is sufficient, did not harm performance [Sato *et al.*, 2021], and thus at least preserves the original learning ability of MPNNs in such settings. Overall, these are surprising findings, which suggest that MPNNs can viably improve across all possible data with partial and even small amounts of randomization.

Second, we observe that the fully randomized GCN-RNI performs substantially worse than its partially randomized counterparts. Whereas fully randomized GCN-RNI only performs marginally worse on Exp (cf. Figure 2a) than partially randomized models, this gap is very large on CExp, primarily due to Corrupt. This observation concurs with the earlier idea of inductive bias: Fully randomized GCN-RNI loses all node type information, which is key for Corrupt, and therefore struggles. Indeed, the model fails to achieve even 60% accuracy on Corrupt, where other models are near perfect, and also relatively struggles on $\overline{\text{Exp}}$, only reaching 91% accuracy and converging slower.

Third, all GCN-RNI models, at all randomization levels, converge significantly slower than 3-GCN on both CExp and Exp. However, an interesting phenomenon can be seen on CExp: All GCN-RNI models fluctuate around 55% accuracy within the first 100 epochs, suggesting a struggle jointly fitting both Corrupt and $\overline{\text{Exp}}$, before they ultimately improve. This, however, is not observed with 3-GCN. Unlike on Exp, randomness is not necessarily beneficial on CExp, as it can hurt performance on Corrupt. Hence, RNI-enhanced models must additionally learn to isolate deterministic dimensions for Corrupt, and randomized dimensions for $\overline{\text{Exp}}$. These findings consolidate the earlier observations made on Exp, and highlight that the variability and slower learning for RNI also hinges on the complexity of the input dataset.

Finally, we observe that both fully randomized GCN-RNI, and, surprisingly, 1-GCN, struggle to learn Corrupt relative to partially randomized GCN-RNI. We also observe that 1-GCN does not "struggle", and begins improving consistently from the start of training. These observations can be attributed to key conceptual , but very distinct hindrances impeding both models. For 1-GCN, the model is jointly trying to learn both $\overline{\text{Exp}}$ and Corrupt, when it provably cannot fit the former. This joint optimization severely hinders Corrupt learning, as data pairs from both subsets are highly similar, and share identically generated UNSAT graphs. Hence, 1-GCN, in attempting to fit SAT graphs from both subsets, knowing it cannot distinguish $\overline{\text{Exp}}$ pairs, struggles to learn the simpler difference in Corrupt pairs. For GCN-RNI, the model discards key type information, so must only rely on structural differences to learn Corrupt, which impedes its convergence. All in all, this further consolidates the promise of partial RNI as a means to combine the strengths of both deterministic and random features.

6 Related Work

MPNNs have been enhanced with RNI [Sato *et al.*, 2021], such that the model trains and runs with partially randomized initial node features. These models, denoted rGNNs, are shown to near-optimally approximate solutions to specific combinatorial optimization problems, and can distinguish between 1-WL indistinguishable graph pairs based on fixed local substructures. Nonetheless, the precise impact of RNI on GNNs for learning arbitrary functions over graphs remained open. Indeed, rGNNs are only shown to admit parameters that can detect a *unique, fixed* substructure, and thus tasks requiring *simultaneous* detection of multiple combinations of structures, as well as problems having no locality or structural biases, are not captured by the existing theory.

Our work improves on Theorem 1 of [Sato *et al.*, 2021], and shows *universality* of MPNNs with RNI. Thus, it shows that arbitrary real-valued functions over graphs can be learned by MPNNs with RNI. Our result is distinctively based on a logical characterization of MPNNs, which allows us to link the size of the MPNN with the descriptive complexity of the target function to be learned. Empirically, we highlight that the power of RNI in a significantly more challenging setting, using a target function (SAT) which does not rely on local structures, is *hard* to approximate.

Similarly to RNI, random pre-set color features have been used to disambiguate between nodes [Dasoulas *et al.*, 2020]. This approach, known as CLIP, introduces randomness to node representations, but explicitly makes graphs distinguishable by construction. By contrast, we study random features produced by RNI, which (i) are not designed a priori to distinguish nodes, (ii) do not explicitly introduce a fixed underlying structure, and (iii) yield potentially infinitely many representations for a single graph. In this more general setting, we nonetheless show that RNI adds expressive power to distinguish nodes with high probability, leads to a universality result, and performs strongly in challenging problem settings.

7 Summary and Outlook

We studied the expressive power of MPNNs with RNI, and showed that these models are universal and preserve MPNN invariance in expectation. We also empirically evaluated these models on carefully designed datasets, and observed that RNI improves their learning ability, but slows their convergence. Our work delivers a theoretical result, supported by practical insights, to quantify the effect of RNI on GNNs. An interesting topic for future work is to study whether polynomial functions can be captured via efficient constructions; see, e.g., [Grohe, 2021] for related open problems.

Acknowledgments

This work was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, by the AXA Research Fund, and by the EPSRC grants EP/R013667/1 and EP/M025268/1. Ralph Abboud is funded by the Oxford-DeepMind Graduate Scholarship and the Alun Hughes Graduate Scholarship. Experiments were conducted on the Advanced Research Computing (ARC) cluster administered by the University of Oxford.

References

[Barceló *et al.*, 2020] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *ICLR*, 2020.

[Battaglia *et al.*, 2018] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[Bengio *et al.*, 2021] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[Cai *et al.*, 1992] Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Comb.*, 12(4):389–410, 1992.

[Clevert *et al.*, 2016] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR*, 2016.

[Dasoulas *et al.*, 2020] George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. In *IJCAI*, 2020.

[Dwivedi *et al.*, 2020] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *CoRR*, abs/2003.00982, 2020.

[Fout *et al.*, 2017] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben - Hur. Protein interface prediction using graph convolutional networks. In *NIPS*, pages 6530–6539, 2017.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272, 2017.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

[Gori *et al.*, 2005] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IJCNN*, volume 2, pages 729–734, 2005.

[Grohe, 2017] Martin Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*, volume 47 of *Lecture Notes in Logic*. Cambridge University Press, 2017.

[Grohe, 2021] Martin Grohe. The logic of graph neural networks. In *LICS*, 2021.

[Hamilton *et al.*, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017.

[Keriven and Peyré, 2019] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *NeurIPS*, pages 7090–7099, 2019.

[Kersting *et al.*, 2016] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. http://graphkernels.cs.tu-dortmund.de.

[Kipf and Welling, 2017] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Maron *et al.*, 2019a] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *NeurIPS*, pages 2153–2164, 2019.

[Maron *et al.*, 2019b] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *ICLR*, 2019.

[Maron *et al.*, 2019c] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *ICML*, pages 4363–4371, 2019.

[Morris *et al.*, 2019] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI*, pages 4602–4609, 2019.

[Puny *et al.*, 2020] Omri Puny, Heli Ben-Hamu, and Yaron Lipman. From graph low-rank global attention to 2-FWL approximation. *CoRR*, abs/2006.07846v1, 2020.

[Sato *et al.*, 2021] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *SDM*, 2021.

[Scarselli *et al.*, 2009] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, pages 974–983, 2018.

[You *et al.*, 2018] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay S. Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *NeurIPS*, pages 6412–6422, 2018.