Coloring Graph Neural Networks for Node Disambiguation

George Dasoulas[1,2] , **Ludovic Dos Santos**[1] , **Kevin Scaman**[1] and **Aladin Virmaux**[1]
[1]Huawei Noah's Ark Lab
[2]École Polytechnique, Paris, France
{george.dasoulas1, ludovic.dos.santos, kevin.scaman, aladin.virmaux}@huawei.com

Abstract

In this paper, we show that a simple coloring scheme can improve, both theoretically and empirically, the expressive power of Message Passing Neural Networks (MPNNs). More specifically, we introduce a graph neural network called Colored Local Iterative Procedure (CLIP) that uses colors to disambiguate identical node attributes, and show that this representation is a *universal approximator* of continuous functions on graphs with node attributes [1]. Our method relies on *separability*, a key topological characteristic that allows to extend well-chosen neural networks into universal representations. Finally, we show experimentally that CLIP is capable of capturing structural characteristics that traditional MPNNs fail to distinguish, while being state-of-the-art on benchmark graph classification datasets.

1 Introduction

Learning good representations is seen by many machine learning researchers as the main reason behind the tremendous successes of the field in recent years [Bengio *et al.*, 2013].

Despite a large literature and state-of-the-art performance on benchmark graph classification datasets, graph neural networks yet lack a similar theoretical foundation [Xu *et al.*, 2019]. Universality for these architectures is either hinted at via equivalence with approximate graph isomorphism tests (k-WL tests in [Xu *et al.*, 2019; Maron *et al.*, 2019a]), or proved under restrictive assumptions (finite node attribute space in [Murphy *et al.*, 2019]).

In this paper, we introduce Colored Local Iterative Procedure (CLIP), which tackles the limitations of current Message Passing Neural Networks (MPNNs) by showing, both theoretically and experimentally, that adding a simple coloring scheme can improve the flexibility and power of these graph representations. More specifically, our contributions are: 1) we provide a precise mathematical definition for universal graph representations, 2) we present a general mechanism to design universal neural networks using separability, 3) we propose a novel node coloring scheme leading to CLIP, the first provably universal extension of MPNNs, 4) we show that CLIP achieves state of the art results on benchmark datasets while significantly outperforming traditional MPNNs as well as recent methods on graph property testing.

The rest of the paper is organized as follows: Section 2 gives an overview of the graph representation literature and related works. Section 3 provides a precise definition for universal representations, as well as a generic method to design them using *separable* neural networks. In Section 4, we show that most state-of-the-art representations are not sufficiently expressive to be universal. Then, using the analysis of Section 3, Section 5 provides CLIP, a provably universal extension of MPNNs. Finally, Section 6 shows that CLIP achieves state-of-the-art accuracies on benchmark graph classification taks, as well as outperforming its competitors on graph property testing problems.

2 Related Works

The first works investigating the use of neural networks for graphs used recurrent neural networks to represent directed acyclic graphs [Sperduti and Starita, 1997; Frasconi *et al.*, 1998]. More generic graph neural networks were later introduced by [Gori *et al.*, 2005; Scarselli *et al.*, 2009], and may be divided into two categories. 1) *Spectral methods* [Bruna *et al.*, 2014; Henaff *et al.*, 2015; Defferrard *et al.*, 2016; Kipf and Welling, 2017] that perform convolution on the Fourier domain of the graph through the spectral decomposition of the graph Laplacian. 2) *Message passing neural networks* [Gilmer *et al.*, 2017], sometimes simply referred to as *graph neural networks*, that are based on the aggregation of neighborhood information through a local iterative process. This category contains most state-of-the-art graph representation methods such as [Duvenaud *et al.*, 2015; Grover and Leskovec, 2016; Lei *et al.*, 2017; Ying *et al.*, 2018; Verma and Zhang, 2019], DeepWalk [Perozzi *et al.*, 2014], graph attention networks [Velickovic *et al.*, 2018], graphSAGE [Hamilton *et al.*, 2017] or GIN [Xu *et al.*, 2019].

Recently, [Xu *et al.*, 2019] showed that MPNNs were, at most, as expressive as the Weisfeiler-Lehman (WL) test for graph isomorphism [Weisfeiler and Lehman, 1968]. This suprising result led to several works proposing MPNN extensions to improve their expressivity, and ultimately tend towards *universality* [Maron *et al.*, 2019a; Maron *et al.*, 2019b; Maron *et al.*, 2019c; Murphy *et al.*, 2019; Chen *et al.*, 2019].

[1]A longer version of the paper with detailed proofs of propositions and theorems is available at https://arxiv.org/pdf/1912.06058.pdf.

However, these graph representations are either as powerful as the k-WL test [Maron *et al.*, 2019a], or provide universal graph representations under the restrictive assumption of finite node attribute space [Murphy *et al.*, 2019]. Other recent approaches [Maron *et al.*, 2019c] implies quadratic order of tensors in the size of the considered graphs. Some more powerfull GNNs are studied and benchmarked on real classical datasets and on graph property testing [Kriege *et al.*, 2018; Murphy *et al.*, 2019; Chen *et al.*, 2019]: a set of problems that classical MPNNs cannot handle. Our work thus provides a more general and powerful result of universality, matching the original definition of [Cybenko, 1989] for MLPs.

3 Universal Representations via Separability

In this section we present the theoretical tools used to design our universal graph representation. More specifically, we show that *separable* representations are sufficiently flexible to capture all relevant information about a given object, and may be extended into universal representations.

3.1 Notations and Basic Assumptions

Let $\mathcal{X}, \mathcal{Y}$ be two topological spaces, then $\mathcal{F}(\mathcal{X}, \mathcal{Y})$ (resp. $\mathcal{C}(\mathcal{X}, \mathcal{Y})$) denotes the space of all functions (resp. continuous functions) from $\mathcal{X}$ to $\mathcal{Y}$. Moreover, for any group G acting on a set $\mathcal{X}$, $\mathcal{X}/G$ denotes the set of orbits of $\mathcal{X}$ under the action of G. Finally, $\|\cdot\|$ is a norm on $\mathbb{R}^d$, and $\mathcal{P}_n$ is the set of all permutation matrices of size n. In what follows, we assume that all the considered topological spaces are *Hausdorff* (see e.g. [Bourbaki, 1998] for an in-depth review): each pair of distinct points can be separated by two disjoint open sets. This assumption is rather weak (e.g. all metric spaces are Hausdorff) and is verified by most topological spaces commonly encountered in the field of machine learning.

3.2 Universal Representations

Let $\mathcal{X}$ be a set of objects (e.g. vectors, images, graphs, or temporal data) to be used as input information for a machine learning task (e.g. classification, regression or clustering). In what follows, we denote as *vector representation* of $\mathcal{X}$ a function $f : \mathcal{X} \to \mathbb{R}^d$ that maps each element $x \in \mathcal{X}$ to a d-dimensional vector $f(x) \in \mathbb{R}^d$. A standard setting for supervised representation learning is to define a class of vector representations $\mathfrak{F}_d \subset \mathcal{F}(\mathcal{X}, \mathbb{R}^d)$ (e.g. convolutional neural networks for images) and use the target values (e.g. image classes) to learn a *good* vector representation in light of the supervised learning task (i.e. one vector representation $f \in \mathfrak{F}_d$ that leads to a good accuracy on the learning task). In order to present more general results, we will consider neural network architectures that can output vectors of any size, i.e. $\mathfrak{F} \subset \cup_{d \in \mathbb{N}^*} \mathcal{F}(\mathcal{X}, \mathbb{R}^d)$, and will denote $\mathfrak{F}_d = \mathfrak{F} \cap \mathcal{F}(\mathcal{X}, \mathbb{R}^d)$ the set of d-dimensional vector representations of $\mathfrak{F}$. A natural characteristic to ask from the class $\mathfrak{F}$ is to be generic enough to approximate any vector representation, a notion that we will denote as *universal representation* [Hornik *et al.*, 1989].

Definition 1. A class of vector representations $\mathfrak{F} \subset \cup_{d \in \mathbb{N}^*} \mathcal{F}(\mathcal{X}, \mathbb{R}^d)$ is called a *universal representation* of $\mathcal{X}$ if for any compact subset $K \subset \mathcal{X}$ and $d \in \mathbb{N}^*$, $\mathcal{F}$ is uniformly dense in $\mathcal{C}(K, \mathbb{R}^d)$.

In other words, $\mathfrak{F}$ is a universal representation of a normed space $\mathcal{X}$ if and only if, for any continuous function $\phi : \mathcal{X} \to \mathbb{R}^d$, any compact $K \subset \mathcal{X}$ and any $\varepsilon > 0$, there exists $f \in \mathfrak{F}$ such that

$$\forall x \in K,\ \|\phi(x) - f(x)\| \leq \varepsilon\,. \tag{1}$$

One of the most fundamental theorems of neural network theory states that one hidden layer MLPs are universal representations of the m-dimensional vector space $\mathbb{R}^m$.

Theorem 1 ([Pinkus, 1999])**.** *Let $\varphi : \mathbb{R} \to \mathbb{R}$ be a continuous non polynomial activation function. For any compact $K \subset \mathbb{R}^m$ and $d \in \mathbb{N}^*$, two layers neural networks with activation φ are uniformly dense in the set $\mathcal{C}(K, \mathbb{R}^d)$.*

However, for graphs and structured objects, universal representations are hard to obtain due to their complex structure and invariance to a group of transformations (e.g. permutations of the node labels). We show in this paper that a key topological property, *separability*, may lead to universal representations of those structures.

3.3 Separability is (Almost) All You Need

Loosely speaking, universal representations can approximate any vector-valued function. It is thus natural to require that these representations are *expressive* enough to separate each pair of dissimilar elements of $\mathcal{X}$.

Definition 2 (Separability)**.** A set of functions $\mathfrak{F} \subset \mathcal{F}(\mathcal{X}, \mathcal{Y})$ is said to *separate* points of $\mathcal{X}$ if for every pair of distinct points x and y, there exists $f \in \mathfrak{F}$ such that $f(x) \neq f(y)$.

For a class of vector representations $\mathfrak{F} \subset \cup_{d \in \mathbb{N}^*} \mathcal{F}(\mathcal{X}, \mathbb{R}^d)$, we will say that $\mathfrak{F}$ is *separable* if its 1-dimensional representations $\mathfrak{F}_1$ separates points of $\mathcal{X}$. Separability is rather weak, as we only require the existence of different outputs for every pair of inputs. Unsurprisingly, we now show that it is a necessary condition for universality.

Proposition 1. *Let $\mathfrak{F}$ be a universal representation of $\mathcal{X}$, then $\mathfrak{F}_1$ separates points of $\mathcal{X}$.*

While separability is necessary for universal representations, it is also key to designing neural network architectures that can be extended into universal representations. More specifically, under technical assumptions, separable representations can be composed with a universal representation of $\mathbb{R}^d$ (such as MLPs) to become universal.

Theorem 2. *For all $d \geq 0$, let $\mathcal{M}_d$ be a universal approximation of $\mathbb{R}^d$. Let $\mathfrak{F}$ be a class of vector representations of $\mathcal{X}$ such that:*

(i) ***Continuity:*** *every $f \in \mathfrak{F}$ is continuous,*

(ii) ***Stability by concatenation:*** *for all $f, g \in \mathfrak{F}$, $x \mapsto (f(x), g(x)) \in \mathfrak{F}$,*

(iii) ***Separability:*** *$\mathfrak{F}_1$ separates points of $\mathcal{X}$.*

Then $\{\psi \circ f\ :\ \exists d \geq 1$ s.t. $\psi \in \mathcal{M}_d, f \in \mathfrak{F}\}$ is a universal representation of $\mathcal{X}$.

Stability by concatenation is verified by most neural networks architectures, as illustrated for MLPs in Figure 1. The proof of Theorem 2 relies on the Stone-Weierstrass theorem (see e.g. [Rudin, 1987]) whose assumptions are continuity,

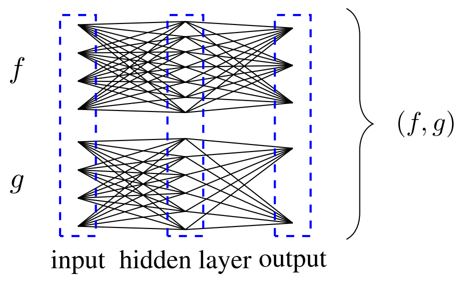

Figure 1: Concatenation of two MLPs f and g.

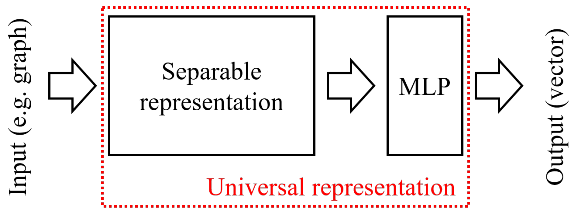

Figure 2: Universal representations can easily be created by combining a separable representation with an MLP.

separability, and the fact that the class of functions is an algebra. Fortunately, composing a separable and concatenable representation with a universal representation automatically leads to an algebra, and thus the applicability of the Stone-Weierstrass theorem and the desired result. Since MLPs are universal representations of $\mathbb{R}^d$, Theorem 2 implies a convenient way to design universal representations of more complex object spaces: create a separable representation and compose it with a simple MLP (see Figure 2).

Corollary 1. *A continuous, concatenable and separable representation of $\mathcal{X}$ composed with an MLP is universal.*

Note that many neural networks of the deep learning literature have this two steps structure, including classical image CNNs such as AlexNet [Krizhevsky *et al.*, 2012] or Inception [Szegedy *et al.*, 2016]. In this paper, we use Corollary 1 to design universal graph and neighborhood representations, although the method is much more generic and may be applied to other objects.

4 Limitations of Existing Representations

In this section, we first provide a proper definition for graphs with node attributes, and then show that message passing neural networks are not sufficiently expressive to be universal.

4.1 Graphs with Node Attributes

Consider a dataset of n interacting objects (e.g. users of a social network) in which each object $i \in [\![1, n]\!]$ has a vector attribute $v_i \in \mathbb{R}^m$ and is a node in an undirected graph G with adjacency matrix $A \in \mathbb{R}^{n\times n}$.

Definition 3. The space of graphs of size n with m-dimensional node attributes is the quotient space

$$\mathbf{Graph}_{m,n} = \left\{(v, A) \in \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times n}\right\} / \mathcal{P}_n\,, \qquad (2)$$

where A is the adjacency matrix of the graph, v contains the m-dimensional representation of each node in the graph and the set of permutations matrices $\mathcal{P}_n$ is acting on (v, A) by

$$\forall P \in \mathcal{P}_n, \quad P \cdot (v, A) = (Pv, PAP^\top)\,. \qquad (3)$$

Moreover, we limit ourselves to graphs of maximum size $n_{\max}$, where $n_{\max}$ is a large integer. This allows us to consider functions on graphs of different sizes without obtaining infinite dimensional spaces and infinitely complex functions that would be impossible to learn via a finite number of samples. We thus define $\mathbf{Graph}_m = \bigcup_{n\le n_{\max}} \mathbf{Graph}_{m,n}$.

4.2 Message Passing Neural Networks

A common method for designing graph representations is to rely on local iterative procedures. Following the notations of [Xu *et al.*, 2019], a *message passing neural network* (MPNN) [Gilmer *et al.*, 2017] is made of three consecutive phases that will create intermediate node representations $x_{i,t}$ for each node $i \in [\![1, n]\!]$ and a final graph representation x_G as described by the following procedure: 1) **Initialization:** All node representations are initialized with their node attributes $x_{i,0} = v_i$. 2) **Aggregation and combination:** T local iterative steps are performed in order to capture larger and larger structural characteristics of the graph. 3) **Readout:** This step combines all final node representations into a single graph representation: $x_G = \text{READOUT}(\{x_{i,T}\}_{i\in[\![1,n]\!]})$, where READOUT is permutation invariant.

Unfortunately, while MPNNs are very efficient in practice and proven to be as expressive as the Weisfeiler-Lehman algorithm [Weisfeiler and Lehman, 1968; Xu *et al.*, 2019], they are not sufficiently expressive to construct isomorphism tests or separate all graphs (for example, consider k-regular graphs without node attributes, for which a small calculation shows that any MPNN representation will only depend on the number of nodes and degree k [Xu *et al.*, 2019]). As a direct application of Proposition 1, MPNNs are thus not expressive enough to create universal representations.

5 Extending MPNNs Using a Simple Coloring Scheme

In this section, we present Colored Local Iterative Procedure (CLIP), an extension of MPNNs using colors to differentiate identical node attributes, that is able to capture more complex structural graph characteristics than traditional MPNNs. This is proved theoretically through a universal approximation theorem in Section 5.3 and experimentally in Section 6. CLIP is based on three consecutive steps: 1) graphs are colored with several different colorings, 2) a neighborhood aggregation scheme provides a vector representation for each colored graph, 3) all vector representations are combined to provide a final output vector. We now provide more information on the coloring scheme.

5.1 Colors to Differentiate Nodes

In order to distinguish non-isomorphic graphs, our approach consists in coloring nodes of the graph with identical attributes.

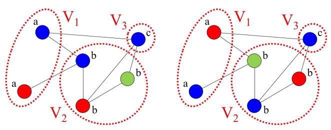

Figure 3: Example of two valid colorings of the same attributed graph. Note that each V_k contains nodes with identical attributes.

This idea is inspired by classical graph isomorphism algorithms that use colors to distinguish nodes [McKay, 1981], and may be viewed as an extension of one-hot encodings used for graphs without node attributes [Xu *et al.*, 2019].

For any $k \in \mathbb{N}$, let C_k be a finite set of k colors. These colors may be represented as one-hot encodings (C_k is the natural basis of $\mathbb{R}^k$) or more generally any finite set of k elements. At initialization, we first partition the nodes into groups of identical attributes $V_1, ..., V_K \subset [\![1, n]\!]$. Then, for a subset V_k of size $|V_k|$, we give to each of its nodes a distinct color from C_k (hence a subset of size $|V_k|$). For example, Figure 3 shows two colorings of the same graph, which is decomposed in three groups V_1, V_2 and V_3 containing nodes with attributes a, b and c respectively. Since V_1 contains only two nodes, a coloring of the graph will attribute two colors ($(1, 0)$ and $(0, 1)$, depicted as *blue* and *red*) to these nodes. More precisely, the set of colorings $\mathcal{C}(v, A)$ of a graph $G = (v, A)$ are defined as

$$\mathcal{C}(v, A) = \Big\{(c_1, ..., c_n) : \forall k \in [\![1, K]\!], (c_i)_{i \in V_k} \text{ is a permutation of } C_{|V_k|}\Big\}. \quad (4)$$

5.2 The CLIP Algorithm

In the CLIP algorithm, we add a coloring scheme to an MPNN in order to distinguish identical node attributes. This is achieved by modifying the initialization and readout phases of MPNNs as follows.

1. **Colored initialization:** We first select a set $\mathcal{C}_k \subseteq \mathcal{C}(v, A)$ of k distinct colorings uniformly at random (see Eq. (4)). Then, for each coloring $c \in \mathcal{C}_k$, node representations are initialized with their node attributes concatenated with their color: $x_{i,0}^c = (v_i, c_i)$.
2. **Aggregation and combination:** This step is performed for all colorings $c \in \mathcal{C}_k$ using a universal set representation as the aggregation function: $x_{i,t+1}^c = \psi^{(t)}\big(x_{i,t}^c, \sum_{j \in \mathcal{N}_i} \varphi^{(t)}(x_{j,t}^c)\big)$, where ψ and φ are MLPs with continuous non-polynomial activation functions and $\psi(x, y)$ denotes the result of ψ applied to the concatenation of x and y. The aggregation scheme we propose is closely related to DeepSet [Zaheer *et al.*, 2017], and a direct application of Corollary 1 proves the universality of our architecture.
3. **Colored readout:** This step performs a maximum over all possible colorings in order to obtain a final *coloring-independent* graph representation. In order to keep the stability by concatenation, the maximum is taken coefficient-wise
$$x_G = \psi\left(\max_{c \in \mathcal{C}_k} \sum_{i=1}^{n} x_{i,T}^c\right), \quad (5)$$
where ψ is an MLP with continuous non polynomial activation functions.

We treat k as a hyper-parameter of the algorithm and call k-CLIP (resp. ∞-CLIP) the algorithm using k colorings (resp. all colorings, i.e. $k = |\mathcal{C}(v, A)|$). Note that, while our focus is graphs with node attributes, the approach used for CLIP is easily extendable to similar data structures such as directed or weighted graphs with node attributes, graphs with node labels, graphs with edge attributes or graphs with additional attributes at the graph level.

5.3 Universal Representation Theorem

As the colorings are chosen at random, the CLIP representation is itself random as soon as $k < |\mathcal{C}(v, A)|$, and the number of colorings k will impact the variance of the representation. However, ∞-CLIP is deterministic and permutation invariant, as MPNNs are permutation invariant. The separability is less trivial and is ensured by the coloring scheme.

Theorem 3. *The ∞-CLIP algorithm with one local iteration ($T = 1$) is a universal representation of the space* $\mathbf{Graph}_m$ *of graphs with node attributes.*

The proof of Theorem 3 relies on showing that ∞-CLIP is separable and applying Corollary 1. This is achieved by fixing a coloring on one graph and identifying all nodes and edges of the second graph using the fact that all pairs (v_i, c_i) are dissimilar. Similarly to the case of MLPs, only one local iteration is necessary to ensure universality of the representation. This rather counter-intuitive result is due to the fact that all nodes can be identified by their color, and the readout function can aggregate all the structural information in a complex and non-trivial way. However, as for MLPs, one may expect poor generalization capabilities for CLIP with only one local iteration, and deeper networks may allow for more complex representations and better generalization. This point is addressed in the experiments of Section 6. Moreover, ∞-CLIP may be slow in practice due to a large number of colorings, and reducing k will speed-up the computation. Fortunately, while k-CLIP is random, a similar universality theorem still holds even for $k = 1$.

Theorem 4. *The 1-CLIP algorithm with one local iteration ($T = 1$) is a random representation whose expectation is a universal representation of the space* $\mathbf{Graph}_m$ *of graphs with node attributes.*

The proof of Theorem 4 relies on using ∞-CLIP on the augmented node attributes $v_i' = (v_i, c_i)$. As all node attributes are, by design, different, the max over all colorings in Eq. (5) disappears and, for any coloring, 1-CLIP returns an ε-approximation of the target function.

Remark 1. Note that the variance of the representation may be reduced by averaging over multiple samples. Moreover, the proof of Theorem 4 shows that the variance can be reduced to an arbitrary precision given enough training epochs, although this may lead to very large training times in practice.

5.4 Computational Complexity

As the local iterative steps are performed T times on each node and the complexity of the aggregation depends on the number of neighbors of the considered node, the complexity is proportional to the number of edges of the graph E and the number of steps T. Moreover, CLIP performs this iterative aggregation for each coloring, and its complexity is also proportional to the number of chosen colorings $k = |\mathcal{C}_k|$. Hence the complexity of the algorithm is in $O(kET)$.

Note that the number of all possible colorings for a given graph depends exponentially in the size of the groups $V_1, ..., V_K$,

$$|\mathcal{C}(v, A)| = \prod_{k=1}^{K} |V_k|!\,, \tag{6}$$

and thus ∞-CLIP is practical only when most node attributes are dissimilar. This worst case exponential dependency in the number of nodes can hardly be avoided for universal representations. Indeed, a universal graph representation should also be able to solve the graph isomorphism problem. Despite the existence of polynomial time algorithms for a broad class of graphs [Luks, 1982; Bodlaender, 1990], graph isomorphism is still quasi-polynomial in general [Babai, 2016]. As a result, creating a universal graph representation with polynomial complexity for all possible graphs and functions to approximate is highly unlikely, as it would also induce a graph isomorphism test of polynomial complexity and thus solve a very hard and long standing open problem of theoretical computer science.

6 Experiments

In this section we show empirically the practical efficiency of CLIP and its relaxation. We run two sets of experiments to compare CLIP w.r.t. state-of-the-art methods in supervised learning settings: i) on 5 real-world graph classification datasets and ii) on 4 synthetic datasets to distinguish structural graph properties and isomorphism. Both experiments follow the same experimental protocol as described in [Xu *et al.*, 2019]: 10-fold cross validation with grid search hyperparameter optimization.

6.1 Classical Benchmark Datasets

We performed experiments on five common benchmark datasets extracted from standard social networks (IMDBb and IMDBm) and bio-informatics databases (MUTAG, PROTEINS and PTC). Following standard practices for graph classification on these datasets, we use one-hot encodings of node degrees as node attributes for IMDBb and IMDBm [Xu *et al.*, 2019], and perform single-label multi-class classification on all datasets. We compared CLIP with six state-of-the-art baseline algorithms: 1) **WL:** Weisfeiler-Lehman subtree kernel [Shervashidze *et al.*, 2011], 2) **AWL:** Anonymous Walk Embeddings [Ivanov and Burnaev, 2018], 3) **DCNN:** Diffusion-convolutional neural networks [Atwood and Towsley, 2016], 4) **PS:** PATCHY-SAN [Niepert *et al.*, 2016], 5) **DGCNN:** Deep Graph CNN [Zhang *et al.*, 2018] and 6) **GIN:** Graph Isomorphism Network [Xu *et al.*, 2019]. WL and AWL are representative of unsupervised methods coupled with an SVM classifier, while DCNN, PS, DGCNN and GIN are four deep learning architectures. As the same experimental protocol as that of [Xu *et al.*, 2019] was used, we present their reported results on Table 1.

As Table 1 shows, CLIP can achieve state-of-the-art performance on the five benchmark datasets. Moreover, CLIP is consistent across all datasets, while all other competitors have at least one weak performance. This is a good indicator of the robustness of the method to multiple classification tasks and dataset types. Finally, the addition of colors does not improve the accuracy for these graph classification tasks, except on the MUTAG dataset. This may come from the small dataset sizes (leading to high variances) or an inherent difficulty of these classification tasks, and contrasts with the clear improvements of the method for property testing (see Section 6.2).

Remark 2. In three out of five datasets, none of the recent state-of-the-art algorithms have statistically significantly better results than older methods (e.g. WL). We argue that, considering the high variances of all classification algorithms on classical graph datasets, graph property testing may be better suited to measure the expressiveness of graph representation learning algorithms in practice.

6.2 Graph Property Testing

We now investigate the ability of CLIP to identify structural graph properties, a task which was previously used to evaluate the expressivity of graph kernels and on which the Weisfeiler-Lehman subtree kernel has been shown to fail for bounded-degree graphs [Kriege *et al.*, 2018]. The performance of our algorithm is evaluated for the binary classification of four different structural properties: 1) connectivity, 2) bipartiteness, 3) triangle-freeness, 4) circular skip links [Murphy *et al.*, 2019] against three competitors: a) GIN, arguably the most efficient MPNN variant yet published [Xu *et al.*, 2019], b) Ring-GNN, a permutation invariant network that uses the ring of matrix addition and multiplication [Chen *et al.*, 2019], c) RP-GIN, the Graph Isomorphism Network combined with Relational Pooling, as described by [Murphy *et al.*, 2019], which is able to distinguish certain cases of non-isomorphic regular graphs.

The Connectivity, Bipartiteness and Triangle-Freeness datasets consist each of 1000 (20-node) graphs with 500 positive samples and 500 negative ones. Regarding the connectivity, the positive samples correspond to disconnected graphs with two 10-node connected components selected among randomly generated graphs and then we constructed negative samples by adding to positive samples a random edge between the two connected components. Regarding the bipartiteness, the positive samples correspond to bipartite graphs, while for the negative samples (non-bipartite graphs) we chose the positive samples and for each of them we added an edge between randomly selected nodes from the same partition, in order to form odd cycles.[2]. Regarding the triangle-freeness, the positive samples correspond to triangle-free graphs selected among randomly generated graphs and then we constructed negative samples by randomly adding new edges to positive samples until it creates at least one triangle. In all three datasets, we generated random graphs using the

[2]Having an odd cycle in a graph makes the graph non bipartite.

Dataset	PTC	IMDBb	IMDBm	PROTEINS	MUTAG
WL	59.9±4.3	73.8±3.9*	50.9±3.8*	75.0±3.1*	90.4±5.7
DCNN	56.6	49.1	33.5	61.3	67.0
PS	60.0±4.8	71.0±2.2	45.2±2.8	75.9±2.8*	92.6±4.2*
DGCNN	58.6	70.0	47.8	75.5*	85.8
AWL	-	74.5±5.9*	51.5±3.6*	-	87.9±9.8
GIN	64.6±7.0*	75.1±5.1*	52.3±2.8*	76.2±2.8*	89.4±5.6
0-CLIP	65.9±4.0*	75.4±2.0*	**52.5±2.6**	77.0±3.2*	90.0±5.1
CLIP	**67.9±7.1**	**76.0±2.7**	**52.5±3.0**	**77.1±4.4**	**93.9±4.0**

Table 1: Classification accuracies on benchmark datasets. The best performer w.r.t. the mean is highlighted in bold. We perform an unpaired t-test with asymptotic significance of 0.1 w.r.t. the best performer and highlight with an asterisk the ones for which the difference is not statistically significant. 0-CLIP is the CLIP architecture without any colorings.

Property	Connectivity	Bipartiteness	Triangle-freeness	Circular skip links		
	mean ± std	mean ± std	mean ± std	mean ± std	max	min
GIN	55.2 ± 4.4	53.1 ±4.7	50.7±6.1	10.0 ± 0.0	10.0	10.0
Ring-GNN	-	-	-	(?) ± 15.7	80.0	10.0
1-RP-GIN	66.1±5.2	66.0±5.1	63.0±3.6	20.0 ± 7.0	28.6	10.0
16-RP-GIN	83.3±7.9	64.9±4.1	65.7±3.3	37.6 ± 12.9	53.3	10.0
0-CLIP	56.5 ± 4.0	55.4 ± 5.7	59.6 ± 3.8	10.0 ± 0.0	10.0	10.0
1-CLIP	73.3 ± 2.2	63.3 ±1.9	63.5 ±7.3	61.9 ±11.9	80.7	36.7
16-CLIP	**99.7 ± 0.5**	**99.2 ± 0.9**	**94.2±3.4**	**90.8 ± 6.8**	**98.7**	**76.0**

Table 2: Classification accuracies of the synthetic datasets. k-RP-GIN refers to a relational pooling averaged over k random permutations. We report Ring-GNN results from [Chen *et al.*, 2019].

Erdös-Rényi model with probabilities $p = 0.5, 0.5, 0.1$ respectively. Lastly, the Circular skip links dataset consists of 150 graphs of 41 nodes as described in [Murphy *et al.*, 2019; Chen *et al.*, 2019]. The Circular Skip Links graphs are undirected regular graphs with node degree 4. We denote a Circular skip link graph by $G_{n,k}$ an undirected graph of n nodes, where $(i, j) \in E$ holds if and only if $|i - j| \equiv 1$ or $k(\mod n)$ This is a 10-class multiclass classification task whose objective is to classify each graph according to its isomorphism class.

Table 2 shows that CLIP is able to capture the structural information of connectivity, bipartiteness, triangle-freeness and circular skip links, while MPNN variants fail to identify these graph properties. Furthermore, we observe that CLIP outperforms RP-GIN, that was shown to provide very expressive representations for regular graphs [Murphy *et al.*, 2019], even with a high number of permutations (the equivalent of colors in their method is set to $k = 16$). Moreover, both for k-RP-GIN and k-CLIP, the increase of permutations and colorings respectively lead to higher accuracies. In particular, CLIP can capture almost perfectly the different graph properties with as little as $k = 16$ colorings.

7 Conclusion

In this paper, we showed that a simple coloring scheme can improve the expressive power of MPNNs. Using such a coloring scheme, we extended MPNNs to create CLIP, the first universal graph representation. Universality was proven using the novel concept of separable neural networks, and our experiments showed that CLIP is state-of-the-art on both graph classification datasets and property testing tasks. The coloring scheme is especially well suited to hard classification tasks that require complex structural information to learn. The framework is general and simple enough to extend to other data structures such as directed, weighted or labeled graphs. Future work includes more detailed and quantitative approximation results depending on the parameters of the architecture such as the number of colors k, or number of hops of the iterative neighborhood aggregation.

References

[Atwood and Towsley, 2016] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[Babai, 2016] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697. ACM, 2016.

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *PAMI*, 2013.

[Bodlaender, 1990] Hans L Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11(4):631–643, 1990.

[Bourbaki, 1998] N. Bourbaki. *General Topology: Chapters 1-4*. Number vol. 4 in Addison-Wesley series in mathematics. Springer, 1998.

[Bruna *et al.*, 2014] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. *ICLR*, 2014.

[Chen *et al.*, 2019] Z. Chen, S. Villar, L. Chen, and J. Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *NeurIPS 2019*, 2019.

[Cybenko, 1989] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 1989.

[Defferrard *et al.*, 2016] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[Duvenaud *et al.*, 2015] D. K Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.

[Frasconi *et al.*, 1998] Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *IEEE transactions on Neural Networks*, 1998.

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

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

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *SIGKDD*, 2016.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[Hornik *et al.*, 1989] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[Ivanov and Burnaev, 2018] Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. *ICML*, 2018.

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

[Kriege *et al.*, 2018] N. M. Kriege, C. Morris, A. Rey, and C. Sohler. A property testing framework for the theoretical expressivity of graph kernels. In *IJCAI*, 2018.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[Lei *et al.*, 2017] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving neural architectures from sequence and graph kernels. *ICML*, 2017.

[Luks, 1982] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42 – 65, 1982.

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

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

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

[McKay, 1981] Brendan D McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[Murphy *et al.*, 2019] R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro. Relational pooling for graph representations. In *ICML*, 2019.

[Niepert *et al.*, 2016] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, 2016.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, 2014.

[Pinkus, 1999] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

[Rudin, 1987] Walter Rudin. *Real and Complex Analysis, 3rd Ed.* McGraw-Hill, Inc., New York, NY, USA, 1987.

[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*, 2009.

[Shervashidze *et al.*, 2011] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 2011.

[Sperduti and Starita, 1997] Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 1997.

[Szegedy *et al.*, 2016] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

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

[Verma and Zhang, 2019] Saurabh Verma and Zhi-Li Zhang. Graph capsule convolutional neural networks. *ICLR*, 2019.

[Weisfeiler and Lehman, 1968] Boris Weisfeiler and AA Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 1968.

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

[Ying *et al.*, 2018] R. Ying, J. You, C. Morris, X. Ren, W. L Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NIPS*, pages 4805–4815, 2018.

[Zaheer *et al.*, 2017] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. Deep sets. In *NIPS*, 2017.

[Zhang *et al.*, 2018] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.