VilLain: Self-Supervised Learning on Homogeneous Hypergraphs without Features via Virtual Label Propagation (Online Appendix)

Geon Lee
KAIST
Seoul, South Korea
geonlee0325@kaist.ac.kr

Soo Yong Lee
KAIST
Seoul, South Korea
syleetolow@kaist.ac.kr

Kijung Shin
KAIST
Seoul, South Korea
kijungs@kaist.ac.kr

ABSTRACT

This document provides supplementary information to the main paper "VilLain: Self-Supervised Learning on Homogeneous Hypergraphs without Features via Virtual Label Propagation."

A ADDITIONAL EXPERIMENTAL RESULTS

In this section, we provide additional experimental results that are not covered in the main context.

A.1 Higher-Order Homogeneity

We examine the number of steps of label propagation required for the average entropy of the hyperedges in the real-world hypergraphs to reach ϵ of that of the hyperedges in the randomized hypergraphs. For example, it requires 3,409 steps of label propagation to reach 0.999 of the average entropy of random hypergraphs, as shown in Table 1. These results support Observation 2, i.e., real-world hypergraphs exhibit not only the hyperedge-level label homogeneity but also the higher-order homogeneity.

Table 1: The number of label propagation steps required for the average entropy of the hyperedges in the real-world hypergraphs to reach ϵ of that of the hyperedges in the randomized hypergraphs.

	PR	HG	CS	CR	PM	DB	TV	AZ
$\epsilon = 0.9$	6	15	140	31	16	832	812	22
$\epsilon = 0.99$	15	34	456	102	43	2,813	2,234	47
$\epsilon = 0.999$	22	47	∞	161	70	4,046	3,409	59

A.2 Full Results

We provide the full results on the three considered downstream tasks: node classification (Table 11), hyperedge prediction (Table 12), node clustering (Table 13), and node retrieval (Table 14). In these tables, we include the results of the space-efficient version, VilLain$_{\text{B}}$ (see Appendix B). For VilLain$_{\text{B}}$, we consider two variants, VilLain$_{\text{B}}^{128}$ and VilLain$_{\text{B}}^{256}$, which generate binary embeddings that cost 128 and 256 bits, respectively, for each embedding vector. We set the number of v-labels in each subspace to 4 for both variants. Note that VilLain$_{\text{B}}^{128}$ and VilLain$_{\text{B}}^{256}$ require only **1/32** and **1/16** of the space used by the other methods, respectively. We include the standard deviation in the tables. In node classification, hyperedge prediction, and node retrieval tasks, on average, VilLain and VilLain$_{\text{B}}$ show the best performance. In the node clustering task, VilLain show the second-best performance. Notably, VilLain$_{\text{B}}^{128}$ and VilLain$_{\text{B}}^{256}$, which require substantially less number of bits for embeddings than the other, highly rank on average. Moreover, it is worthwhile to notice that the proposed methods outperform (semi-)supervised methods (e.g., HGNN and AllSet), which are trained specifically for the node classification task. We conjecture that v-label propagation inherits rich structural properties and also potential higher-order structure-label relationships, generating high-quality representations of nodes.

A.3 Connections to Contrastive Loss

We investigate the relevance of the loss functions utilized by VilLain, particularly examining alignment and uniformity, which are essential properties of contrastive learning-based methods.

Conceptual analysis. We examine the relevance of the loss functions employed by VilLain and alignment and uniformity, which are key properties of contrastive losses. We discuss how VilLain incorporates both alignment and uniformity properties inherent in contrastive learning methods.

- While **alignment** is implemented in a specific way in contrastive learning, alignment, in general, refers to the property of assigning similar features to similar samples. VilLain ensures this property by aiming to minimize $\mathcal{L}_{\text{local}}$. It aims to reduce the entropy of v-label assignment probability vectors of each hyperedge (or node), where these probability vectors are computed by averaging the probability distributions of nodes (resp. hyperedges) incident to the hyperedge (resp. node) being considered. By minimizing the entropy of the averaged probability vectors, the model encourages v-label assignments of nodes (or hyperedges) incident to the same hyperedges (resp. nodes) to be close to each other.
- **Uniformity** generally refers to uniformly distributing features in the latent space. In VilLain, this is ensured by aiming to minimize $\mathcal{J}_{\text{cls}}$ and $\mathcal{J}_{\text{dst}}$. Minimizing ensures the diversity in v-label distributions while minimizing ensures distinctiveness among the distributed v-labels. This prevents the v-label distributions of nodes from becoming excessively aligned and thus preserves maximal information within the latent space.

Distinction from contrastive losses. Furthermore, we clarify how our loss functions distinctly differ from contrastive losses.

WWW '24, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0171-9/24/05.
https://doi.org/10.1145/3589334.3645454

Table 2: Dual loss functions of VilLain, $\mathcal{L}_{\text{local}}$ and $\mathcal{L}_{\text{global}}$ improve the alignment and uniformity of the embeddings.

Method	Alignment ($-\mathbb{E}_{(u,v)\sim p_{\text{pos}}}\|\tilde{z}_u - \tilde{z}_v\|^2$)								Uniformity ($-\log \mathbb{E}_{(u,v)\sim p_{\text{data}}} e^{-2\|\tilde{z}_u - \tilde{z}_v\|^2}$)							
	DB	TV	AZ	PM	HG	CS	CR	PM	DB	TV	AZ	PM	HG	CS	CR	PM
Random	-1.998	-2.000	-2.000	-2.001	-2.005	-2.002	-1.996	-1.998	3.924	3.963	3.926	3.926	3.949	3.930	3.939	3.949
$\mathcal{L}_{\text{local}}$	-0.054	-0.031	-0.098	-0.703	-0.274	-0.084	-0.192	-0.179	2.644	3.633	1.442	1.258	1.377	2.295	2.536	1.417
$\mathcal{L}_{\text{local}} + \mathcal{L}_{\text{global}}$	-0.060	0.032	-0.091	-0.778	-0.319	-0.094	-0.215	-0.207	2.926	3.652	1.348	1.761	1.712	2.442	2.824	1.898

Table 3: Introducing $\mathcal{L}_{\text{global}}$ prevents the dominance and entropy of v-label distributions from approaching 1 and 0, respectively, which is a scenario where a single v-label dominates the entire hypergraph.

		Primary	High	Citeseer	Cora	Primary	DBLP	Trivago	Amazon
$\mathcal{L}_{\text{local}}$	Dominance (↓)	0.994	0.852	0.676	0.789	0.873	0.782	0.539	0.992
	Entropy (↑)	0.010	0.289	0.598	0.447	0.254	0.483	0.688	0.026
$\mathcal{L}_{\text{local}} + \mathcal{L}_{\text{global}}$	Dominance (↓)	**0.577**	**0.600**	**0.532**	**0.546**	**0.619**	**0.584**	**0.510**	**0.580**
	Entropy (↑)	**0.676**	**0.665**	**0.690**	**0.686**	**0.656**	**0.671**	**0.692**	**0.668**

Table 4: Density (i.e., $|E|/|V|$) and overlapness (i.e., $\sum_{e\in E}|e|/|V|$) of each dataset. Primary exhibits exceptionally high density and overlapness compared to other datasets.

	PR	HG	CS	CR	PM	DB	TV	AZ
Density	**52.495**	23.908	0.803	1.130	2.079	0.522	1.350	0.122
Overlapness	**126.979**	55.633	2.755	3.457	9.049	2.510	4.207	1.622

- Contrastive loss is typically employed to ensure alignment and uniformity at the node level. However, in VilLain 's approach, $\mathcal{L}_{\text{local}}$ and $\mathcal{L}_{\text{global}}$ operate on different dimensions. Specifically, minimizing $\mathcal{L}_{\text{local}}$ focuses on embedding alignment at the node or hyperedge level, aiming to assign similar v-labels to structurally similar nodes or hyperedges. On the other hand, minimizing $\mathcal{L}_{\text{global}}$ focuses on embedding uniformity at the v-label level, aiming to distribute v-labels diversely and distinctly across the entire hypergraph.
- Contrastive loss is applied to ensure alignment between different views. However, VilLain 's loss functions aim to align v-labels between nodes or hyperedges that are structurally similar.
- Most importantly, VilLain outperformed the contrastive learning-based method, TriCL across different tasks.

Empirical analysis. To numerically measure the degree of alignment and uniformity within the node embeddings generated by VilLain, we computed the following [5]:

$$-\mathbb{E}_{(u,v)\sim p_{\text{pos}}}\|\tilde{z}_u - \tilde{z}_v\|^2 \quad \text{and} \quad -\log \mathbb{E}_{(u,v)\sim p_{\text{data}}} e^{-2\|\tilde{z}_u - \tilde{z}_v\|^2}$$

where p_{pos} is the distribution of positive node pairs sampled from the same hyperedge, and p_{data} is the distribution of nodes in general. In addition, we used L2 normalized representations, denoted as $\tilde{z}_u$, derived from the embedding z_u of node u.

Table 2 presents a comparison of the measured alignment and uniformity among random embeddings, a variant of VilLain utilizing only $\mathcal{L}_{\text{local}}$, and VilLain employing both $\mathcal{L}_{\text{local}}$ and $\mathcal{L}_{\text{global}}$. In the table, higher values imply greater alignment and uniformity among the embeddings. While minimizing $\mathcal{L}_{\text{local}}$ might result in improved alignment, it could lead to unsatisfactory uniformity. If $\mathcal{L}_{\text{global}}$ is minimized together with $\mathcal{L}_{\text{local}}$, the uniformity is significantly improved at the cost of slight reduction of alignment.

Table 5: VilLain benefits from input node features in node classification. When utilizing node features, it ranks highest on average among its feature-requiring baselines across four datasets where node features are provided.

Method	DBLP	Citeseer	Cora	Pubmed	Rank
GCN	84.45 ± 1.25	64.60 ± 3.00	76.06 ± 2.29	74.92 ± 2.90	5.25 ± 2.62
GAT	77.07 ± 1.63	50.39 ± 3.40	59.79 ± 2.08	73.96 ± 2.13	11.00 ± 1.15
DGI	85.64 ± 1.13	68.53 ± 2.91	77.50 ± 2.04	75.62 ± 2.82	3.50 ± 2.38
GRACE	85.63 ± 1.05	61.33 ± 2.78	71.16 ± 1.81	77.47 ± 1.57	5.75 ± 3.30
GMI	80.85 ± 1.49	57.09 ± 2.68	74.73 ± 1.69	76.38 ± 2.21	8.00 ± 2.44
HGNN	84.36 ± 1.70	64.28 ± 2.53	75.63 ± 1.39	76.63 ± 2.44	5.00 ± 0.81
HNHN	74.44 ± 1.98	58.53 ± 3.31	67.87 ± 3.51	69.38 ± 3.47	10.75 ± 1.25
AllSet	83.67 ± 1.53	57.88 ± 3.14	70.07 ± 3.23	75.24 ± 2.93	9.00 ± 1.15
UniGNN	84.22 ± 1.57	63.79 ± 3.72	74.44 ± 2.50	76.99 ± 2.82	5.75 ± 1.89
HyperGCL	76.12 ± 6.04	63.30 ± 2.11	73.01 ± 3.68	**82.62 ± 3.25**	6.75 ± 4.19
TriCL	**86.59 ± 0.88**	64.53 ± 3.17	**79.03 ± 0.63**	76.60 ± 1.71	2.75 ± 2.06
VilLain	85.68 ± 0.85	**68.77 ± 1.82**	76.54 ± 1.44	78.25 ± 2.41	**2.00 ± 0.81**

A.4 When $\mathcal{L}_{\text{global}}$ is Important

VilLain outperforms VilLain-L in most datasets. Notably, this performance advantage is particularly significant in Primary, and in this subsection, we analyze the reasons behind this improvement and explore when the inclusion of $\mathcal{L}_{\text{global}}$ is particularly beneficial. We hypothesize that VilLain-L faces difficulty in learning distinctive v-label distributions, with a single v-label accounting for nearly 100% of nodes in Primary, regardless of the predefined number of v-labels. This challenge may arise due to the dataset's unique characteristic of densely connected nodes. This is supported by the measured density (i.e., $|E|/|V|$) and overlapness (i.e., $\sum_{e\in E}|e|/|V|$) of the hypergraphs in Table 4.

Furthermore, we empirically validate the effectiveness of $\mathcal{L}_{\text{global}}$ by evaluating how v-labels are distributed with and without it. Specifically, we computed the followings:

- **Dominance** of the v-label is defined as the ratio of the most frequently assigned v-label.
- **Entropy** is derived from the ratio of each v-label assigned din the hypergraph.

Intuitively, when dominance and entropy approach 1 and 0, respectively, it indicates that a single v-label is assigned to all nodes, which

Table 6: The average accuracy over all feature-requiring methods (e.g., GCN, HGNN, and TriCL) using different input features. Hyper2vec is the most useful input feature, compared to learnable embeddings and Node2vec.

Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
Learnable	21.87 ± 3.72	6.75 ± 6.64	13.88 ± 5.73	61.71 ± 24.20	74.19 ± 31.24	41.73 ± 9.26	44.80 ± 13.02	55.11 ± 11.28	2.87 ± 0.33
Node2vec	36.21 ± 5.07	25.11 ± 11.47	**26.63 ± 5.75**	**81.87 ± 10.21**	**95.97 ± 4.92**	53.37 ± 4.18	54.81 ± 6.48	71.32 ± 5.34	1.62 ± 0.48
Hyper2vec	**63.63 ± 6.06**	**55.51 ± 23.03**	OOT	81.79 ± 11.54	92.60 ± 6.48	**57.94 ± 3.27**	**70.05 ± 3.93**	**74.60 ± 4.85**	**1.28 ± 0.45**

Table 7: HGNN and TriCL yield unsatisfactory performance with learnable features and random features, while input features learned by Hyper2vec demonstrate significantly better accuracy. VilLain outperforms them by a large margin.

Method		DBLP	Primary	High	Citeseer	Cora	Pubmed	Rank
HGNN	Learnable Features	21.47 ± 2.28	76.71 ± 3.36	79.58 ± 3.48	42.34 ± 2.11	43.02 ± 2.33	54.66 ± 2.76	4.67 ± 0.47
	Random Features	21.24 ± 1.91	78.43 ± 2.36	83.12 ± 3.65	43.04 ± 3.39	43.26 ± 2.44	54.41 ± 3.46	4.33 ± 0.47
	Hyper2vec	66.60 ± 2.18	88.28 ± 5.02	92.19 ± 3.84	60.91 ± 2.32	72.90 ± 2.00	76.58 ± 2.86	2.67 ± 0.45
TriCL	Learnable Features	19.31 ± 1.11	31.86 ± 2.64	30.34 ± 3.75	24.94 ± 1.62	25.10 ± 2.22	38.74 ± 2.25	6.50 ± 0.50
	Random Features	18.96 ± 1.20	31.84 ± 3.49	38.33 ± 4.42	25.89 ± 2.28	24.29 ± 1.74	39.69 ± 1.93	6.50 ± 0.50
	Hyper2vec	68.18 ± 1.36	92.67 ± 2.50	98.10 ± 1.02	59.17 ± 3.35	72.35 ± 1.53	78.57 ± 1.88	2.33 ± 0.45
	VilLain	**77.16 ± 1.26**	**93.66 ± 3.93**	**99.19 ± 0.41**	**61.53 ± 3.17**	**75.03 ± 1.38**	**78.82 ± 1.47**	**1.00 ± 0.00**

is undesirable. As shown in Table 3, Primary and Amazon exhibit notably high dominance (0.992 and 0.994) and low entropy (0.026 and 0.010) when only $\mathcal{L}_{\text{local}}$ is used to learn v-label distributions. In addition, we can also find that $\mathcal{L}_{\text{global}}$ effectively mitigates this issue, reducing the dominance and increasing the entropy.

A.5 Improvements from Node Features

External node features, if available, are useful and typically enhance method performance. VilLain can be extended to incorporate node features by introducing $|V|$ additional hyperedges, where each hyperedge is a group of the k-nearest neighbors of each node based on cosine similarity between node features. Then, it learns v-label distributions on an augmented hypergraph with $|V|$ nodes and $|V| + |E|$ hyperedges. As shown in Table 5, VilLain benefits from using node features, outperforming its feature-requiring baselines in terms of average ranks when using $k = 3$.

We would like to emphasize that our simple approach to utilizing external node features is distinguished from how other baseline methods utilize them (i.e., projecting and propagating them through edges), potentially making it a suboptimal choice. However, it is crucial to note that VilLain is primarily designed for scenarios where node features are unavailable and thus is tailored to perform best in such cases. Furthermore, it is important to note that in our other experiments, we used topological node features obtained through Hyper2vec, instead of external features for the baselines that require input node features.

A.6 Usefulness as Input Features

We evaluate the usefulness of the methods as an input of the feature-requiring methods (i.e., GCN, GAT, DGI, GRACE, GMI, HGNN, HNHN, AllSet, UniGNN, HyperGCL, and TriCL). Specifically, we train these models using three different input features including a learnable one, which is trained together with the models. As shown in Table 6, using Hyper2vec yields the best accuracy in node classification, and thus we use their embeddings for input features of feature-requiring methods.

In addition, in Table 7, we present a comparison of node classification accuracies of HGNN and TriCL, which are semi-supervised and self-supervised GNN methods for hypergraphs, respectively. We utilize different input features across the considered datasets, except for those that result in out-of-memory issues. We can see that GNNs with learnable features and random features yield unsatisfactory performance, while input features learned by Hyper2vec demonstrate significantly better accuracy. Most importantly, VilLain outperforms them by a large margin.

A.7 Effects of Long-Range V-label Propagation

To examine the effects of the long-range propagation of v-labels, we test how the number of steps k (during training) and k' (during inference) affect the performance of VilLain in the three considered tasks in Tables 15 and 16, respectively. Except for Primary and High, which are the smallest datasets, adopting long-range propagation of v-labels is beneficial for node classification, node retrieval, and hyperedge prediction. In particular, we can see that large datasets (e.g., DBLP, Trivago, and Amazon) benefit from large ks and k's.

A.8 Aggregation for Embedding Generation

In VilLain, embeddings are generated by aggregating embeddings obtained with various numbers of v-labels. While the aggregation method is flexible, we concatenate embeddings obtained using different numbers of v-labels, specifically, for each number $\lceil \frac{d}{D} \rceil \in \{2, 3, \cdots, 8\}$ of v-labels, we learn D subspaces and then perform PCA to ensure that the final embedding is of the target dimension d. In Table 8, we compare the performance with VilLain when applying mean-pooling, instead of PCA, to the embeddings from different v-label numbers, for the embedding aggregation. Across three different downstream tasks, the concatenate-then-PCA outperforms mean-pooling on average.

Table 8: To aggregate embeddings obtained from various numbers of v-labels in each subspace, concatenating the embeddings and applying PCA (PCA) outperforms averaging the embeddings (mean) in the three considered downstream tasks.

	Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
NCS	Mean	74.56 ± 1.14	77.23 ± 1.35	56.36 ± 2.23	**93.88 ± 3.94**	98.95 ± 0.70	**62.70 ± 2.78**	74.38 ± 1.31	**79.03 ± 1.64**	1.62 ± 0.48
	PCA	**77.16 ± 1.26**	**79.43 ± 1.63**	**57.95 ± 2.47**	93.66 ± 3.93	**99.19 ± 0.41**	61.53 ± 3.17	**75.03 ± 1.38**	78.82 ± 1.47	**1.37 ± 0.48**
HP	Mean	80.37 ± 0.97	95.11 ± 0.55	94.81 ± 0.37	82.40 ± 0.89	87.21 ± 0.67	**82.66 ± 0.95**	**79.44 ± 0.57**	**83.10 ± 0.70**	1.62 ± 0.48
	PCA	**81.61 ± 0.52**	**95.12 ± 0.37**	**94.91 ± 0.36**	**83.19 ± 0.56**	**87.79 ± 0.68**	82.08 ± 1.42	78.95 ± 0.79	82.79 ± 0.79	**1.37 ± 0.48**
NCT	Mean	46.32 ± 1.36	65.77 ± 0.32	34.77 ± 0.50	**85.90 ± 1.30**	**98.72 ± 0.00**	34.04 ± 0.86	48.38 ± 0.95	32.62 ± 0.02	1.75 ± 0.43
	PCA	**46.58 ± 0.62**	**69.35 ± 0.32**	**35.24 ± 0.48**	85.67 ± 1.88	**98.72 ± 0.00**	**34.53 ± 0.45**	**50.38 ± 2.25**	**32.73 ± 0.00**	**1.12 ± 0.22**
NR	Mean	49.35 ± 0.00	43.84 ± 0.00	51.26 ± 0.72	86.68 ± 0.00	98.78 ± 0.00	43.95 ± 0.10	52.76 ± 0.30	63.12 ± 0.37	2.00 ± 0.00
	PCA	**60.15 ± 0.55**	**67.23 ± 0.72**	**53.64 ± 0.47**	**91.26 ± 0.00**	**98.99 ± 0.00**	**46.37 ± 0.00**	**57.96 ± 0.07**	**64.43 ± 0.07**	**1.00 ± 0.00**

Table 9: To obtain the embedding of each hyperedge, maxmin-pooling is more effective than mean-pooling in all datasets in both VilLain and TriCL (the strongest considered baseline method).

	Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
VilLain	Mean	52.55 ± 1.13	59.36 ± 1.52	64.99 ± 2.34	57.54 ± 1.79	56.83 ± 1.81	54.03 ± 2.20	56.89 ± 2.38	57.56 ± 0.99	2.00 ± 0.00
	Maxmin	**81.61 ± 0.52**	**95.12 ± 0.37**	**94.91 ± 0.36**	**83.19 ± 0.56**	**87.79 ± 0.68**	**82.08 ± 1.42**	**78.95 ± 0.79**	**82.79 ± 0.79**	**1.00 ± 0.00**
TriCL	Mean	53.60 ± 0.89	OOM	OOM	63.84 ± 0.97	59.29 ± 1.08	61.33 ± 2.40	68.28 ± 1.50	59.69 ± 1.34	2.00 ± 0.00
	Maxmin	**77.40 ± 0.68**	OOM	OOM	**83.00 ± 0.63**	**87.78 ± 0.52**	**81.96 ± 0.91**	**76.69 ± 0.70**	**80.45 ± 0.75**	**1.00 ± 0.00**

A.9 Hyperedge Embedding Method for Hyperedge Prediction

To obtain the embedding of each hyperedge, we apply maxmin pooling, i.e., elementwise max pooling - elementwise min pooling, to the embeddings of the nodes in it. In Table 9, we test the effectiveness of maxmin pooling compared to mean pooling for hyperedge prediction. For both VilLain and TriCL [4], which is the strongest baseline, maxmin pooling is more effective than mean pooling across all datasets.

A.10 Graph-Modeling-Based Methods

In Section **??**, we applied GNNs to pairwise graphs which are transformed from hypergraphs. For this transformation, we adopted clique expansion, which is a popular approach to transform hypergraphs into graphs [6–8]. However, such clique-expanded graphs are often different from graphs conventionally used for GNN benchmarks. Specifically, for the citation datasets (e.g., Cora, Pubmed, and Citeseer), each edge joins co-cited graphs in clique-expanded graphs, while each edge in GNN-benchmark graphs joints a pair of citing and cited papers. Thus, we evaluate the performance of well-established GNN models, specifically GATv2 [1], GCNII [2], and GPRGNN [3], when applied to the original structures of the graph datasets. We consider the setting without features, which our paper focuses on and thus use embeddings obtained from Node2vec as their input features. As shown in Table 10, VilLain outperforms GNN competitors in node classification, even when they use graphs modeled with the same semantics as hypergraphs, instead of clique expansion. This demonstrates the effectiveness of employing hypergraph modeling and VilLain for learning embeddings from its structure.

A.11 Loss of VilLain

We examine how losses of VilLain decrease with training epochs. As discussed in Section **??**, VilLain optimizes two losses $\mathcal{L}_{\text{local}}$ and $\mathcal{L}_{\text{global}}$ that aim to capture the local and global structural information of the hypergraph, respectively. As shown in Figure **??**, the two losses $\mathcal{L}_{\text{local}}$ and $\mathcal{L}_{\text{global}}$ jointly decrease as VilLain is trained. In terms of the number $L = \lceil d/D \rceil$ of v-labels in each subspace, the decrease of $\mathcal{L}_{\text{local}}$ is facilitated by smaller L. Intuitively, a smaller number of v-labels is more likely to lead to homogeneous hyperedges. On the other hand, $\mathcal{L}_{\text{global}}$ decreases faster with a larger number of v-labels in each subspace since more diverse v-labels are more likely to be distinctive from each other.

B VILLAIN$_{\text{B}}$: SPACE-EFFICIENT BINARY EMBEDDING

As hypergraphs grow in size, so does the space required to store the embeddings. Specifically, a continuous d-dimensional vector consisting of d real numbers requires $32d$ bits if float-32 is used to represent each real number. To reduce the space requirement, we propose VilLain$_{\text{B}}$, a space-efficient version of VilLain that produces binary node embeddings for hypergraphs. Specifically, we binarize the continuous vector $\mathbf{Z}_i^{\langle t \rangle}$ of node v_i in each t^{th} subspace obtained by VilLain, which is a probabilistic distribution over d/D v-labels, to a one-hot vector $\widehat{\mathbf{Z}}_i^{\langle t \rangle} \in \{0,1\}^{d/D}$ as:

$$\widehat{\mathbf{Z}}_i^{\langle t \rangle} = \text{one-hot}\left(\arg\max_j \left(\mathbf{Z}_{i,j}^{\langle t \rangle}\right)\right).$$

Then, the final binarized embedding $\widehat{\mathbf{Z}}_i \in \{0,1\}^d$ is obtained by concatenating the binarized embeddings from the D subspaces.

To encode a d/D-dimensional one-hot vector in each subspace, $\lceil \log_2 \frac{d}{D} \rceil$ bits are required. Hence, encoding a final binarized vector, which is the concatenation of D one-hot vectors, requires

Table 10: VilLain, applied to hypergraphs, outperforms the recent graph-based baseline methods (i.e., GATv2, GCNII, and GPRGNN) across benchmark graph datasets.

Graph Type	Model	Cora	Citeseer	Pubmed	Avg. Rank
Graph	GATv2	70.58 ± 1.4	52.76 ± 2.4	72.92 ± 2.9	3.33 ± 0.47
	GCNII	71.37 ± 2.4	56.47 ± 1.6	74.63 ± 2.0	2.00 ± 0.00
	GPRGNN	69.60 ± 2.0	56.38 ± 2.2	71.44 ± 1.6	3.66 ± 0.47
Hypergraph	VilLain	**75.03** ± 1.38	**61.53** ± 3.17	**78.82** ± 1.47	**1.00** ± 0.00

$\lceil D \log_2 \frac{d}{D} \rceil$ bits. Note that, $D \log_2 \frac{d}{D} < 32d$ always holds for any positive integers d and D ($\leq d$).

In Tables 11, 12, 13, and 14, we include the performance of VilLain_B. While requiring a substantially smaller number of bits to encode embeddings, VilLain_B outperforms baseline methods in the considered four downstream tasks.

REFERENCES

[1] Shaked Brody, Uri Alon, and Eran Yahav. 2021. How Attentive are Graph Attention Networks?. In *ICLR*.
[2] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *ICML*.
[3] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive universal generalized pagerank graph neural network. In *ICLR*.
[4] Dongjin Lee and Kijung Shin. 2023. I'm me, we're us, and I'm us: Tri-directional contrastive learning on hypergraphs. In *AAAI*.
[5] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*.
[6] Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. 2020. Nhp: Neural hypergraph link prediction. In *CIKM*.
[7] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with hypergraphs: Clustering, classification, and embedding. In *NeurIPS*.
[8] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. 2016. Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing* 216 (2016), 150–162.

Table 11: Full results on node classification (in terms of accuracy). VilLain and VilLain$_B$ outperform the existing (hyper)graph representation learning methods.

Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
GCN	49.65 ± 2.91	18.53 ± 3.61	19.08 ± 2.43	89.54 ± 2.49	89.82 ± 4.99	53.35 ± 3.76	63.71 ± 2.73	70.89 ± 1.60	12.62 ± 2.64
GAT	OOM	OOM	OOM	58.48 ± 7.12	76.94 ± 9.60	51.06 ± 4.29	62.74 ± 3.07	61.66 ± 5.27	16.60 ± 1.35
Deepwalk	29.03 ± 1.43	16.85 ± 0.45	25.43 ± 1.72	84.89 ± 3.67	99.31 ± 0.48	45.10 ± 3.18	56.58 ± 1.88	68.58 ± 2.60	13.00 ± 5.04
Node2vec	29.21 ± 1.89	16.88 ± 0.44	25.27 ± 2.36	83.53 ± 3.09	**99.38 ± 0.45**	45.37 ± 3.17	59.15 ± 1.84	69.05 ± 3.00	12.37 ± 5.09
DGI	62.37 ± 3.32	73.46 ± 1.22	31.80 ± 1.45	86.66 ± 4.51	92.49 ± 0.60	61.36 ± 2.91	71.23 ± 2.04	77.51 ± 1.38	8.37 ± 3.87
GRACE	71.86 ± 2.51	OOM	OOM	63.78 ± 5.12	99.03 ± 0.30	61.16 ± 2.78	73.43 ± 1.81	77.70 ± 1.81	6.50 ± 4.85
GMI	64.19 ± 1.63	OOM	OOM	80.10 ± 4.94	96.61 ± 2.63	58.67 ± 2.68	71.31 ± 1.69	75.51 ± 2.77	10.66 ± 2.05
HGNN	66.60 ± 2.18	OOM	OOM	88.28 ± 5.02	92.19 ± 3.84	60.91 ± 2.32	72.90 ± 2.00	76.58 ± 2.86	8.66 ± 3.19
HNHN	63.99 ± 2.21	59.52 ± 1.64	28.99 ± 2.63	91.31 ± 2.47	96.83 ± 1.25	59.02 ± 1.63	68.81 ± 1.26	75.33 ± 1.77	8.87 ± 2.08
AllSet	63.67 ± 1.89	36.58 ± 0.93	21.75 ± 1.67	85.94 ± 3.02	95.70 ± 1.66	56.08 ± 1.95	67.73 ± 1.81	74.11 ± 2.04	11.75 ± 1.19
UniGNN	67.16 ± 2.15	69.98 ± 1.60	33.77 ± 3.22	88.88 ± 3.58	95.12 ± 3.97	59.10 ± 2.76	71.44 ± 1.03	74.37 ± 2.10	8.37 ± 2.91
HyperGCL	58.72 ± 1.54	74.99 ± 1.23	22.86 ± 2.01	74.07 ± 6.06	85.79 ± 8.92	57.54 ± 1.61	74.99 ± 1.33	78.44 ± 3.33	9.37 ± 5.67
Hyper2vec	67.18 ± 1.78	75.82 ± 1.45	OOT	92.52 ± 2.45	96.34 ± 1.34	61.50 ± 2.60	71.79 ± 1.63	77.04 ± 1.51	5.85 ± 2.84
LBSN	22.63 ± 2.20	47.99 ± 0.82	11.56 ± 0.90	86.71 ± 3.71	95.87 ± 2.28	45.43 ± 2.15	59.70 ± 1.31	54.89 ± 2.38	13.62 ± 3.27
TriCL	68.18 ± 1.36	OOM	OOM	92.67 ± 2.50	98.10 ± 1.02	59.17 ± 3.35	72.35 ± 1.53	78.57 ± 1.88	5.44 ± 2.13
VilLain$_B^{128}$	67.99 ± 1.16	64.93 ± 1.76	52.37 ± 1.82	**95.63 ± 0.28**	99.32 ± 0.17	60.83 ± 2.82	74.40 ± 1.38	77.57 ± 1.61	4.25 ± 1.98
VilLain$_B^{256}$	70.39 ± 1.76	69.26 ± 1.45	52.40 ± 2.03	95.15 ± 2.04	99.14 ± 0.29	60.27 ± 2.97	74.46 ± 1.88	78.00 ± 1.20	4.00 ± 1.73
VilLain	**77.16 ± 1.26**	**79.43 ± 1.63**	**57.95 ± 2.47**	93.66 ± 3.93	99.19 ± 0.41	**61.53 ± 3.17**	**75.03 ± 1.38**	**78.82 ± 1.47**	**1.62 ± 1.11**

Table 12: Full results on hyperedge prediction (in terms of accuracy). VilLain and VilLain$_B$ (see Appendix B) outperform the existing (hyper)graph representation learning methods.

Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
Deepwalk	63.90 ± 0.94	61.27 ± 1.14	69.36 ± 0.74	83.79 ± 0.68	85.87 ± 0.73	69.55 ± 1.63	67.18 ± 1.32	65.90 ± 0.62	8.00 ± 2.87
Node2vec	64.20 ± 0.79	61.43 ± 0.81	69.29 ± 0.70	83.15 ± 0.86	85.36 ± 0.64	70.35 ± 1.44	66.94 ± 1.57	65.75 ± 0.77	7.87 ± 2.36
DGI	**86.05 ± 0.60**	83.83 ± 0.70	90.82 ± 0.65	79.06 ± 0.99	84.38 ± 0.77	79.15 ± 0.94	76.33 ± 0.97	80.92 ± 0.74	5.37 ± 2.95
GRACE	85.43 ± 0.76	OOM	OOM	80.32 ± 0.77	87.42 ± 0.46	77.88 ± 1.31	74.52 ± 0.78	79.05 ± 0.75	5.00 ± 1.82
GMI	75.60 ± 0.71	OOM	OOM	82.43 ± 0.68	85.90 ± 0.60	74.41 ± 1.25	69.40 ± 1.38	72.34 ± 0.70	7.16 ± 1.21
Hyper2vec	71.19 ± 1.01	72.36 ± 1.08	OOT	76.41 ± 0.92	79.57 ± 0.85	78.05 ± 1.76	71.65 ± 1.54	71.48 ± 0.88	8.14 ± 1.95
LBSN	48.68 ± 1.08	89.08 ± 0.68	63.65 ± 1.60	79.43 ± 0.80	87.05 ± 0.60	74.29 ± 1.64	69.63 ± 0.98	66.10 ± 0.77	7.62 ± 2.34
TriCL	77.40 ± 0.76	OOM	OOM	**83.99 ± 0.70**	87.78 ± 0.44	81.96 ± 1.42	76.69 ± 0.79	80.45 ± 0.67	3.66 ± 1.69
VilLain$_B^{128}$	79.39 ± 0.78	93.49 ± 0.66	92.97 ± 0.60	79.76 ± 0.54	86.05 ± 0.51	82.48 ± 1.29	78.92 ± 1.27	81.42 ± 0.79	3.87 ± 2.08
VilLain$_B^{256}$	79.44 ± 0.68	93.90 ± 0.75	92.64 ± 0.52	79.81 ± 0.75	86.35 ± 0.65	**83.03 ± 1.18**	**79.95 ± 1.35**	80.69 ± 0.71	3.37 ± 1.93
VilLain	81.61 ± 0.52	**95.12 ± 0.37**	**94.91 ± 0.36**	83.19 ± 0.56	**87.79 ± 0.68**	82.08 ± 1.42	78.95 ± 0.79	**82.79 ± 0.79**	**1.87 ± 0.92**

Table 13: Full results on node clustering (in terms of normalized mutual information). VilLain and VilLain$_B$ (see Appendix B) outperform the existing (hyper)graph representation learning methods.

Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
Deepwalk	0.70 ± 0.05	16.70 ± 0.21	7.75 ± 0.03	85.15 ± 0.39	**100.00 ± 0.00**	14.62 ± 1.73	23.87 ± 1.83	**34.35 ± 0.33**	6.50 ± 3.60
Node2vec	0.91 ± 0.03	16.96 ± 0.54	7.73 ± 0.13	83.47 ± 0.70	**100.00 ± 0.00**	14.52 ± 0.82	23.80 ± 1.21	32.83 ± 0.07	7.50 ± 3.20
DGI	16.63 ± 0.01	44.50 ± 1.68	13.01 ± 0.95	84.43 ± 1.68	73.88 ± 0.95	29.09 ± 0.61	32.07 ± 0.79	31.27 ± 0.00	7.50 ± 2.06
GRACE	42.96 ± 0.11	OOM	OOM	67.59 ± 1.12	98.17 ± 0.18	33.04 ± 1.33	46.04 ± 2.44	31.55 ± 0.11	6.16 ± 3.02
GMI	27.80 ± 2.61	OOM	OOM	84.08 ± 0.54	93.10 ± 0.26	25.33 ± 1.72	42.60 ± 3.56	18.71 ± 0.01	8.50 ± 1.25
Hyper2vec	43.40 ± 0.94	66.33 ± 0.27	OOT	**92.48 ± 0.35**	99.34 ± 0.30	34.28 ± 0.30	45.53 ± 1.05	33.62 ± 0.13	**2.57 ± 0.90**
LBSN	1.05 ± 0.00	39.41 ± 0.12	2.68 ± 0.33	85.53 ± 0.55	97.80 ± 0.24	12.14 ± 0.43	28.96 ± 0.30	4.62 ± 0.59	8.50 ± 1.87
TriCL	38.00 ± 0.02	OOM	OOM	87.83 ± 1.22	98.74 ± 0.00	34.41 ± 0.02	44.75 ± 0.30	33.74 ± 0.01	3.66 ± 1.37
VilLain$_B^{128}$	35.77 ± 1.92	56.51 ± 0.41	31.94 ± 0.13	89.40 ± 0.04	98.72 ± 0.00	31.60 ± 0.73	44.99 ± 1.84	32.40 ± 0.00	4.75 ± 1.56
VilLain$_B^{256}$	35.90 ± 0.92	58.85 ± 0.40	31.23 ± 0.16	89.76 ± 1.18	98.72 ± 0.00	32.34 ± 1.79	49.08 ± 1.23	33.43 ± 0.02	3.62 ± 1.21
VilLain	**46.58 ± 0.62**	**69.35 ± 0.32**	**35.24 ± 0.48**	85.67 ± 1.88	98.72 ± 0.00	**34.53 ± 0.45**	**50.38 ± 2.25**	32.73 ± 0.00	2.62 ± 2.11

Table 14: Full results on node retrieval (in terms of mean average precision). VilLain and VilLain$_B$ (see Appendix B) outperform the existing (hyper)graph representation learning methods.

Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
Deepwalk	21.34 ± 0.24	7.47 ± 0.11	27.71 ± 0.17	81.62 ± 0.00	98.72 ± 0.00	27.61 ± 0.07	29.24 ± 0.14	48.95 ± 0.19	8.37 ± 1.99
Node2vec	21.61 ± 0.20	7.09 ± 0.07	27.78 ± 0.17	81.07 ± 0.00	98.58 ± 0.00	27.29 ± 0.05	29.42 ± 0.15	49.39 ± 0.21	8.50 ± 1.65
DGI	36.06 ± 0.37	37.32 ± 0.13	31.13 ± 0.38	89.73 ± 0.00	97.81 ± 0.00	43.78 ± 0.10	50.64 ± 0.30	61.65 ± 0.36	4.75 ± 1.29
GRACE	50.22 ± 0.60	OOM	OOM	61.41 ± 0.00	**99.49 ± 0.00**	41.09 ± 0.08	54.17 ± 0.36	60.94 ± 0.36	5.33 ± 3.19
GMI	34.63 ± 0.33	OOM	OOM	80.00 ± 0.00	97.78 ± 0.00	35.89 ± 0.07	41.62 ± 0.28	55.10 ± 0.31	8.33 ± 0.74
Hyper2vec	35.47 ± 0.41	43.11 ± 0.48	OOT	85.74 ± 0.00	90.70 ± 0.00	41.21 ± 0.09	46.67 ± 0.27	55.62 ± 0.19	6.57 ± 2.44
LBSN	21.01 ± 0.09	19.05 ± 0.14	29.11 ± 0.37	81.30 ± 0.00	93.22 ± 0.00	30.60 ± 0.09	40.09 ± 0.26	43.50 ± 0.39	8.62 ± 2.05
TriCL	45.11 ± 0.56	OOM	OOM	89.91 ± 0.00	97.64 ± 0.00	42.37 ± 0.10	54.98 ± 0.26	61.94 ± 0.25	4.66 ± 2.13
VilLain$_B^{128}$	47.27 ± 0.54	35.39 ± 0.76	50.34 ± 0.53	89.65 ± 0.00	99.21 ± 0.00	43.33 ± 0.10	53.57 ± 0.29	62.50 ± 0.35	3.87 ± 1.05
VilLain$_B^{256}$	53.78 ± 0.58	42.36 ± 0.62	50.61 ± 0.52	90.03 ± 0.00	99.22 ± 0.00	44.35 ± 0.09	56.11 ± 0.30	61.51 ± 0.35	2.50 ± 1.00
VilLain	**60.15 ± 0.55**	**67.23 ± 0.72**	**53.64 ± 0.47**	**91.26 ± 0.00**	98.99 ± 0.00	**46.37 ± 0.10**	**57.96 ± 0.27**	**64.43 ± 0.36**	**1.37 ± 0.99**

Table 15: Effects of ks in node classification (NCS), hyperedge prediction (HP), and node clustering (NCT) node retrieval (NR). VilLain benefits from the long-range propagation during training.

	Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
NCS	$k=1$	74.25 ± 1.05	78.14 ± 1.89	52.16 ± 2.36	**94.67 ± 2.47**	**99.51 ± 0.39**	60.48 ± 3.17	74.96 ± 1.04	78.21 ± 2.18	3.00 ± 1.22
	$k=2$	75.76 ± 1.41	78.44 ± 1.84	55.09 ± 2.55	93.43 ± 3.84	99.29 ± 0.39	60.17 ± 3.69	**75.15 ± 1.30**	78.97 ± 1.38	2.62 ± 0.85
	$k=4$	77.16 ± 1.26	79.43 ± 1.63	57.95 ± 2.47	93.66 ± 3.93	99.19 ± 0.41	61.53 ± 3.17	75.03 ± 1.38	78.82 ± 1.47	2.25 ± 0.43
	$k=8$	**78.22 ± 1.20**	**80.24 ± 1.89**	**59.12 ± 2.58**	92.47 ± 4.00	98.78 ± 0.69	**62.05 ± 3.52**	74.24 ± 1.49	**79.22 ± 1.73**	**2.12 ± 1.45**
HP	$k=1$	80.72 ± 0.68	94.60 ± 0.56	93.81 ± 0.48	**83.51 ± 0.64**	87.54 ± 0.63	81.51 ± 1.28	77.23 ± 0.91	81.80 ± 0.79	3.50 ± 1.00
	$k=2$	81.22 ± 0.61	94.82 ± 0.48	94.79 ± 0.44	83.36 ± 0.72	87.69 ± 0.55	82.02 ± 1.22	78.80 ± 0.90	82.29 ± 0.59	2.75 ± 0.43
	$k=4$	81.61 ± 0.52	95.12 ± 0.37	94.91 ± 0.36	83.19 ± 0.56	**87.79 ± 0.68**	82.08 ± 1.42	78.95 ± 0.79	82.79 ± 0.79	2.00 ± 0.50
	$k=8$	**81.97 ± 0.72**	**95.24 ± 0.49**	**95.27 ± 0.26**	82.99 ± 0.56	87.39 ± 0.47	**82.62 ± 1.13**	**79.13 ± 0.82**	**82.94 ± 0.49**	**1.75 ± 1.29**
NCT	$k=1$	43.36 ± 1.66	65.28 ± 0.28	32.66 ± 0.22	**90.40 ± 1.86**	**98.72 ± 0.00**	32.29 ± 2.06	44.14 ± 2.38	**33.96 ± 0.24**	2.87 ± 1.45
	$k=2$	45.50 ± 1.10	67.06 ± 0.52	33.13 ± 0.44	87.70 ± 1.65	**98.72 ± 0.00**	34.16 ± 1.78	47.38 ± 2.13	33.95 ± 0.27	2.50 ± 0.70
	$k=4$	46.58 ± 0.62	69.35 ± 0.32	35.24 ± 0.48	85.67 ± 1.88	**98.72 ± 0.00**	34.53 ± 0.45	50.38 ± 2.25	32.73 ± 0.00	2.12 ± 0.59
	$k=8$	**48.35 ± 0.06**	**71.44 ± 0.28**	**36.97 ± 0.07**	84.61 ± 1.19	**98.72 ± 0.00**	**35.16 ± 0.42**	**50.52 ± 0.94**	32.52 ± 0.00	**1.75 ± 1.29**
NR	$k=1$	55.55 ± 0.63	57.90 ± 0.74	48.44 ± 0.52	91.07 ± 0.00	**99.30 ± 0.00**	43.78 ± 0.11	53.84 ± 0.27	62.94 ± 0.33	3.50 ± 1.00
	$k=2$	58.42 ± 0.61	63.12 ± 0.70	51.61 ± 0.42	**91.35 ± 0.00**	99.10 ± 0.00	45.07 ± 0.11	55.98 ± 0.27	63.60 ± 0.35	2.62 ± 0.69
	$k=4$	60.15 ± 0.55	67.23 ± 0.72	53.64 ± 0.47	91.26 ± 0.00	98.99 ± 0.00	46.37 ± 0.10	**57.96 ± 0.27**	**64.43 ± 0.36**	**1.87 ± 0.59**
	$k=8$	**62.75 ± 0.48**	**69.08 ± 0.58**	**56.01 ± 0.64**	90.69 ± 0.00	98.70 ± 0.00	**47.68 ± 0.10**	57.72 ± 0.25	63.64 ± 0.37	2.00 ± 1.22

Table 16: Effects of k's in node classification (NCS), hyperedge prediction (HP), and node clustering (NCT) node retrieval (NR). VilLain benefits from the long-range propagation at inference (i.e., embedding generation).

	Method	DBLP	Trivago	Amazon	Primary	High	Citeseer	Cora	Pubmed	Rank
NCS	$k'=1$	64.71 ± 1.98	60.60 ± 1.54	48.46 ± 3.45	**96.74 ± 0.79**	**99.58 ± 0.20**	60.62 ± 2.94	74.68 ± 1.57	77.94 ± 1.84	5.62 ± 2.86
	$k'=2$	65.29 ± 1.91	61.59 ± 1.62	49.42 ± 2.76	96.36 ± 2.19	99.57 ± 0.21	60.44 ± 3.03	74.70 ± 1.58	78.18 ± 1.75	5.50 ± 2.29
	$k'=4$	66.64 ± 2.19	63.25 ± 1.63	50.52 ± 2.45	96.33 ± 1.99	99.39 ± 0.21	60.54 ± 3.28	74.77 ± 1.75	78.29 ± 2.14	5.00 ± 1.58
	$k'=8$	67.88 ± 1.79	65.08 ± 1.55	53.22 ± 3.74	93.91 ± 2.57	99.26 ± 0.38	61.29 ± 3.30	**75.06 ± 1.44**	78.75 ± 1.91	4.50 ± 1.41
	$k'=16$	70.83 ± 1.70	68.28 ± 1.38	54.65 ± 3.63	94.49 ± 3.05	98.86 ± 0.79	61.62 ± 3.28	74.86 ± 1.34	79.12 ± 1.44	3.75 ± 0.82
	$k'=32$	73.20 ± 1.60	72.31 ± 1.65	55.80 ± 2.41	92.57 ± 3.78	98.59 ± 1.42	61.96 ± 3.50	74.68 ± 1.30	79.22 ± 1.69	4.00 ± 1.65
	$k'=64$	76.47 ± 1.30	76.77 ± 1.71	56.42 ± 2.59	94.21 ± 3.34	98.50 ± 2.31	62.42 ± 2.93	74.25 ± 1.99	78.98 ± 1.68	4.00 ± 2.29
	$k'=128$	**77.62 ± 1.26**	**80.63 ± 1.36**	**57.46 ± 2.04**	88.68 ± 4.90	98.09 ± 1.90	**63.67 ± 3.26**	74.41 ± 1.76	**79.37 ± 1.92**	**3.50 ± 3.24**
HP	$k'=1$	77.77 ± 0.57	91.46 ± 0.57	93.11 ± 0.56	82.05 ± 0.92	87.45 ± 0.65	82.14 ± 0.97	79.63 ± 0.84	82.09 ± 0.70	6.75 ± 1.92
	$k'=2$	77.86 ± 0.57	91.53 ± 0.61	93.39 ± 0.75	82.30 ± 0.56	**87.74 ± 0.61**	82.39 ± 1.59	79.29 ± 0.87	82.27 ± 0.56	5.25 ± 1.98
	$k'=4$	78.58 ± 0.86	92.51 ± 0.82	93.77 ± 0.77	82.89 ± 0.84	87.53 ± 0.55	81.79 ± 1.24	78.28 ± 0.79	82.12 ± 0.82	5.87 ± 1.83
	$k'=8$	79.06 ± 0.83	92.31 ± 0.61	94.07 ± 0.51	82.86 ± 0.79	87.69 ± 0.66	82.23 ± 1.19	78.35 ± 0.93	82.46 ± 0.65	5.00 ± 1.11
	$k'=16$	79.87 ± 0.63	92.97 ± 0.44	94.59 ± 0.39	**83.06 ± 0.71**	87.55 ± 0.62	82.16 ± 0.97	78.87 ± 0.89	82.49 ± 0.75	4.12 ± 1.45
	$k'=32$	80.39 ± 0.53	93.80 ± 0.60	95.15 ± 0.38	82.89 ± 0.84	87.28 ± 0.45	82.74 ± 1.15	79.29 ± 0.84	82.92 ± 0.65	3.25 ± 1.56
	$k'=64$	81.06 ± 0.47	94.58 ± 0.61	95.15 ± 0.42	82.78 ± 0.85	87.70 ± 0.44	83.03 ± 0.92	79.51 ± 0.78	82.75 ± 0.55	**2.62 ± 0.99**
	$k'=128$	**81.89 ± 0.87**	**95.15 ± 0.49**	**95.21 ± 0.36**	81.91 ± 0.77	87.04 ± 0.70	**83.42 ± 1.19**	**80.23 ± 0.85**	**83.00 ± 0.55**	2.75 ± 3.03
NCT	$k'=1$	29.62 ± 1.44	48.38 ± 0.50	31.74 ± 0.05	93.08 ± 0.05	**98.72 ± 0.00**	33.04 ± 1.07	48.93 ± 2.07	32.36 ± 0.00	5.25 ± 2.72
	$k'=2$	30.30 ± 1.57	49.05 ± 0.33	31.74 ± 0.40	**93.09 ± 0.01**	**98.72 ± 0.00**	33.79 ± 0.53	49.78 ± 1.39	32.30 ± 0.00	4.87 ± 2.75
	$k'=4$	30.26 ± 1.88	50.24 ± 0.32	31.81 ± 0.45	91.95 ± 1.25	**98.72 ± 0.00**	33.95 ± 0.78	48.40 ± 1.85	32.31 ± 0.48	4.87 ± 1.76
	$k'=8$	30.39 ± 1.45	51.75 ± 0.75	33.62 ± 0.03	86.48 ± 1.53	**98.72 ± 0.00**	34.78 ± 0.54	**50.74 ± 1.88**	32.77 ± 0.01	**3.50 ± 1.73**
	$k'=16$	31.90 ± 1.66	54.59 ± 0.27	34.29 ± 0.05	84.45 ± 1.26	**98.72 ± 0.00**	34.97 ± 0.68	48.65 ± 1.49	32.61 ± 0.00	3.62 ± 1.11
	$k'=32$	35.63 ± 1.87	58.75 ± 0.66	34.05 ± 0.46	83.87 ± 0.32	98.42 ± 0.19	35.54 ± 1.93	47.49 ± 0.47	**32.98 ± 0.00**	4.25 ± 2.10
	$k'=64$	44.85 ± 1.29	64.91 ± 0.36	35.40 ± 0.58	84.87 ± 0.32	97.65 ± 0.00	**37.41 ± 0.76**	45.56 ± 1.05	32.65 ± 0.00	3.87 ± 2.52
	$k'=128$	**47.62 ± 0.05**	**70.88 ± 0.30**	**35.66 ± 0.61**	83.92 ± 0.38	98.09 ± 0.00	36.68 ± 0.22	44.35 ± 0.87	32.12 ± 0.00	4.37 ± 3.15
NR	$k'=1$	44.77 ± 0.54	31.08 ± 0.79	51.00 ± 0.42	91.94 ± 0.00	**99.41 ± 0.00**	45.47 ± 0.11	57.46 ± 0.26	63.04 ± 0.33	5.50 ± 2.39
	$k'=2$	45.34 ± 0.55	30.95 ± 0.82	50.25 ± 0.41	**91.95 ± 0.00**	99.35 ± 0.00	45.62 ± 0.11	57.51 ± 0.26	63.15 ± 0.34	5.25 ± 2.33
	$k'=4$	46.06 ± 0.60	33.43 ± 0.83	48.84 ± 0.45	91.72 ± 0.00	99.22 ± 0.00	45.05 ± 0.10	56.59 ± 0.27	63.21 ± 0.34	5.75 ± 2.04
	$k'=8$	48.00 ± 0.57	36.56 ± 0.77	51.32 ± 0.40	91.41 ± 0.00	99.06 ± 0.00	46.25 ± 0.10	57.79 ± 0.26	63.46 ± 0.35	4.25 ± 0.96
	$k'=16$	50.65 ± 0.58	40.19 ± 0.65	51.19 ± 0.39	91.01 ± 0.00	98.79 ± 0.00	46.77 ± 0.10	58.04 ± 0.26	**63.50 ± 0.36**	4.00 ± 1.22
	$k'=32$	54.32 ± 0.59	46.72 ± 0.76	**54.67 ± 0.37**	90.89 ± 0.00	98.49 ± 0.00	47.09 ± 0.10	58.23 ± 0.25	63.31 ± 0.37	**3.37 ± 1.65**
	$k'=64$	58.34 ± 0.58	60.01 ± 0.77	53.54 ± 0.48	90.88 ± 0.00	98.33 ± 0.00	**47.10 ± 0.10**	**58.29 ± 0.26**	62.97 ± 0.36	3.62 ± 2.64
	$k'=128$	**61.23 ± 0.53**	**70.75 ± 0.79**	52.60 ± 0.42	90.88 ± 0.00	98.29 ± 0.00	46.80 ± 0.10	58.28 ± 0.26	62.76 ± 0.36	4.12 ± 2.84