Appendices

A Proof of Theorems 1 and 2

We extend Theorem 1 in [20] to a general diffusion operator L in the following lemma.

Lemma 1. Suppose that a graph $\mathcal{G}$ has k connected components $\{C_i\}_{i=1}^k$ and the diffusion operator L is defined as that in (2). L has k linearly independent eigenvectors $\{\boldsymbol{v_1}, \ldots, \boldsymbol{v_k}\}$ corresponding to its largest eigenvalue λ_{max}. If $\mathcal{G}$ has no bipartite components, then for any $\boldsymbol{x} \in \mathbb{R}^N$

$$\lim_{m\to\infty} (\frac{1}{\lambda_{max}} L)^m \boldsymbol{x} = [\boldsymbol{v_1}, \ldots, \boldsymbol{v_k}]\boldsymbol{\theta}, \tag{11}$$

for some $\boldsymbol{\theta} \in \mathbb{R}^k$.

Lemma 2. Suppose we randomly sample $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$ under a continuous distribution. Suppose we have pointwise function $\mathrm{ReLU}(z) = \max(0, z)$, we have

$$\mathbb{P}(\mathrm{rank}\,(\mathrm{ReLU}([\boldsymbol{x}, \boldsymbol{y}])) \leq \mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]) \mid \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N) = 1$$

Proof The main point of this lemma is to show that, for all vector pairs $[\boldsymbol{x}, \boldsymbol{y}]$ in the vector space $\mathbb{R}^N$, the pairs with two linearly independent vectors that will be transformed to be linearly dependent after ReLU transformation cover a nonzero area and the pairs with two linearly dependent vectors that will be transformed to be independent after ReLU transformation cover a zero area. The details are as follows.

We generalize ReLU onto multi-dimensional case by applying it element-wise on every element of the matrix. Any $\boldsymbol{x} \in \mathbb{R}^N$ can be represented as $\boldsymbol{x} = \boldsymbol{x_+} + \boldsymbol{x_-}$, where $\boldsymbol{x_+}$ and $\boldsymbol{x_-}$ are the nonnegative and nonpositive components of $\boldsymbol{x}$, respectively. We can see that $\mathrm{ReLU}(\boldsymbol{x}) = \boldsymbol{x}_+$. It is trivial when $\boldsymbol{y} = 0$. We only discuss for all nonzero $\boldsymbol{y} \in \mathbb{R}^N$.

Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ are linearly independent (which happens with probability 1), then $\nexists\, c \neq 0$ that $\boldsymbol{x} = c\boldsymbol{y}$. If $\exists\, d \neq 0$ that $\mathrm{ReLU}(\boldsymbol{x}) = d\ \mathrm{ReLU}(\boldsymbol{y})$, then $\boldsymbol{x_+} = d\boldsymbol{y_+}$ and $\boldsymbol{x_-} \neq d\boldsymbol{y_-}$ and the existence of this kind of $\boldsymbol{x}, \boldsymbol{y}$ has a nonzero probability $\frac{1}{2^N}\frac{1+N}{2^N}$ (See Lemma 3); other than these cases, the independency will be kept after the ReLU transformation.

Suppose $\boldsymbol{y}$ is linearly dependent of $\boldsymbol{x}$ (which happens with probability 0), *i.e.*, $\exists\, c \neq 0$ that $\boldsymbol{x} = c\boldsymbol{y}$. If $c > 0$, we have $\boldsymbol{x_-} = c\boldsymbol{y_-}$ and $\boldsymbol{x_+} = c\boldsymbol{y_+}$. Since $\mathrm{ReLU}(\boldsymbol{x}) = \boldsymbol{x_+}$, $\mathrm{ReLU}(\boldsymbol{y}) = \boldsymbol{y_+}$, then $\mathrm{ReLU}(\boldsymbol{x}) = c\mathrm{ReLU}(\boldsymbol{y})$. The dependency will be kept. If $c < 0$, we have $\boldsymbol{x_-} = c\boldsymbol{y_+}$ and $\boldsymbol{x_+} = c\boldsymbol{y_-}$. If $\exists\, d \neq 0$ that $\mathrm{ReLU}(\boldsymbol{x}) = d\ \mathrm{ReLU}(\boldsymbol{y})$, this means $\boldsymbol{x_+} = d\boldsymbol{y_+} = \frac{d}{c}\boldsymbol{x_-}$. This holds only when $x = 0$. This happens with probability 0. Then, $\mathrm{ReLU}(\boldsymbol{x})$ and $d\ \mathrm{ReLU}(\boldsymbol{y})$ will be independent when $c < 0$. So whether or not ReLU will keep the dependency between to vectors depends on the sign of c. Thus, under the assumption, we have probability $\frac{1}{2}$ that ReLU will keep the dependency and probability $\frac{1}{2}$ will not.

According to the discussion above, we have

$$\mathbb{P}\left(\mathrm{rank}\,(\mathrm{ReLU}([\boldsymbol{x}, \boldsymbol{y}])) \leq \mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]) \mid \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N\right) = \frac{\mathbb{P}\left(\mathrm{rank}\,(\mathrm{ReLU}([\boldsymbol{x}, \boldsymbol{y}])) \leq \mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]), \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N\right)}{\mathbb{P}\left(\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N\right)}$$

$$= \mathbb{P}\left(\mathrm{rank}\,(\mathrm{ReLU}([\boldsymbol{x}, \boldsymbol{y}])) \leq \mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]) \mid \mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]) = 1, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N\right)\mathbb{P}\left(\mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]) = 1, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N\right) +$$

$$\mathbb{P}\left(\mathrm{rank}\,(\mathrm{ReLU}([\boldsymbol{x}, \boldsymbol{y}])) \leq \mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]) \mid \mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]) = 2, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N\right)\mathbb{P}\left(\mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}]) = 2, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N\right)$$

$$= 0 \times \frac{1}{2} + 1 \times 1 = 1.$$

Lemma proved. □

Lemma 3. Suppose we randomly sample $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$ under a continuous distribution, then

$$\mathbb{P}(\boldsymbol{x_+} = d\boldsymbol{y_+}, \boldsymbol{x_-} \neq d\boldsymbol{y_-} \mid x, y \in \mathbb{R}^N, d \neq 0) = \frac{1}{2^N}\frac{1+N}{2^N}$$

Proof

$$\begin{aligned}
&\mathbb{P}(\boldsymbol{x_+} = d\boldsymbol{y_+}, \boldsymbol{x_-} \neq d\boldsymbol{y_-}) \\
&= \mathbb{P}(\boldsymbol{x_+} = d\boldsymbol{y_+}, \boldsymbol{x_-} \neq d\boldsymbol{y_-} \mid df(\boldsymbol{x_+}) \leq 1)\ \mathbb{P}(df(\boldsymbol{x_+}) \leq 1) + \mathbb{P}(\boldsymbol{x_+} = d\boldsymbol{y_+}, \boldsymbol{x_-} \neq d\boldsymbol{y_-} \mid df(\boldsymbol{x_+}) > 1)\ \mathbb{P}(df(\boldsymbol{x_+}) > 1) \\
&= \frac{1}{2^N}\frac{1+N}{2^N} + 0 \cdot \frac{2^N - 1 - N}{2^N} = \frac{1}{2^N}\frac{1+N}{2^N}
\end{aligned}$$

where df denotes degree of freedom. $df(\boldsymbol{x_+}) \leq 1$ means that $\boldsymbol{x}$ can at most have one dimension to be positive and there are $1 + N$ out of 2^N hyperoctants that satisfies this condition. The set of $\boldsymbol{y}$ that can make $\boldsymbol{x_+} = d\boldsymbol{y_+}, \boldsymbol{x_-} \neq d\boldsymbol{y_-}$ hold has an area of $\frac{1}{2^N}$, *i.e.* when $\boldsymbol{y}$ is in the same hyperoctant as $\boldsymbol{x}$. If $\boldsymbol{x}$ lies in other hyperoctants, $df(y_-) \leq N - 2$. And since $\boldsymbol{x_+} = d\boldsymbol{y_+}$, $\boldsymbol{y}$ is just a low dimensional surface in $\mathbb{R}^N$ with area 0. □

Theorem 1. Suppose that $\mathcal{G}$ has k connected components and the diffusion operator L is defined as that in (2). Let $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ be any block vector that randomly sampled under a continuous distribution and $\{W_0, W_1, \ldots, W_n\}$ be any set of parameter matrices, if $\mathcal{G}$ has no bipartite components, then in (4), as $n \to \infty$, $\text{rank}(\boldsymbol{Y'}) \leq k$ almost surely.

Proof Upon the conclusions in Lemma 2-3, we have $\text{rank}(\text{ReLU}(L\boldsymbol{X})) \leq \text{rank}(L\boldsymbol{X})$ with probability 1 and it is obvious $\text{rank}(L\boldsymbol{X}W_0) \leq \text{rank}(L\boldsymbol{X})$. Using these two inequality iteratively for (4), we have $\text{rank}(\boldsymbol{Y'}) \leq \text{rank}(L^{n+1}\boldsymbol{X})$. Based on Lemma 1, we have probability 1 to get

$$\lim_{n\to\infty} \text{rank}(\boldsymbol{Y'}) \leq \lim_{n\to\infty} \text{rank}(L^{n+1}\boldsymbol{X}) = \text{rank}([\boldsymbol{v_1}, \ldots, \boldsymbol{v_k}][\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_F}]) \leq \text{rank}([\boldsymbol{v_1}, \ldots, \boldsymbol{v_k}]) = k,$$

where $\boldsymbol{\theta_i} \in \mathbb{R}^k, i = 1, \ldots, F$. Thus, $\text{rank}(\boldsymbol{Y'}) \leq k$ □

Theorem 2. Suppose we randomly sample $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$ under a continuous distribution and we have the pointwise function $\text{Tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, we have

$$\mathbb{P}(\text{rank}\,(\text{Tanh}([\boldsymbol{x}, \boldsymbol{y}])) \geq \text{rank}([\boldsymbol{x}, \boldsymbol{y}]) \mid \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N) = 1$$

Proof We first prove it for $N = 2$. It is trivial when x, y have 0 elements.

Suppose $\boldsymbol{x} = [x_1, x_2]$, $\boldsymbol{y} = [y_1, y_2]$ are linearly dependent and all of their elements are nonzero. If $\text{Tanh}(\boldsymbol{x})$ and $\text{Tanh}(\boldsymbol{y})$ are still linearly dependent, we must have

$$\frac{\text{Tanh}(x_1)}{\text{Tanh}(y_1)} = \frac{\text{Tanh}(x_2)}{\text{Tanh}(y_2)},\ s.t.\ \frac{x_1}{y_1} = \frac{x_2}{y_2}$$

This equation only have one solution

$$x_1 = \frac{1033977}{9530},\ x_2 = -\frac{929}{10},\ y_1 = -\frac{1113}{10},\ y_2 = \frac{953}{10}$$

which means pointwise Tanh transformation will break the dependency of $\boldsymbol{x}, \boldsymbol{y}$ with probability 1.

If x, y are linearly independent, suppose $\text{Tanh}(\boldsymbol{x}) = \boldsymbol{x'} = [x'_1, x'_2]$ is a vector in $\mathbb{R}^2$, and the set of vectors in $\mathbb{R}^2$ that can be transformed by pointwise Tanh to the same line as $\boldsymbol{x'}$ covers an area of probability 0. That is for any fixed $\boldsymbol{x'} \in \mathbb{R}^2$, the solution $[y_1, y_2]$ of

$$\frac{\text{Tanh}(y_1)}{\text{Tanh}(y_2)} = \frac{x'_1}{x'_2}$$

covers an area of 0. Thus,

$$\mathbb{P}(\text{rank}\,(\text{Tanh}([\boldsymbol{x}, \boldsymbol{y}])) \geq \text{rank}([\boldsymbol{x}, \boldsymbol{y}]) \mid \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N) = 1$$

which means pointwise Tanh transformation will increase the independency between vectors in $\mathbb{R}^2$.

Similar to $\mathbb{R}^2$, suppose $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]$, $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]$ are linearly dependent, if all elements in $\boldsymbol{x}, \boldsymbol{y}$ are nonzero and $\mathrm{Tanh}(\boldsymbol{x})$ and $\mathrm{Tanh}(\boldsymbol{y})$ are still linearly dependent, we must have

$$\frac{\mathrm{Tanh}(x_1)}{\mathrm{Tanh}(y_1)} = \frac{\mathrm{Tanh}(x_2)}{\mathrm{Tanh}(y_2)} = \cdots = \frac{\mathrm{Tanh}(x_N)}{\mathrm{Tanh}(y_N)}, \ s.t. \ \frac{x_1}{y_1} = \frac{x_2}{y_2} \cdots = \frac{x_N}{y_N}$$

The solution of any pair of equations covers an area of probability 0 in $\mathbb{R}^N$. Actually, this still holds when $\boldsymbol{x}, \boldsymbol{y}$ have some 0 elements, *i.e.* for any subset of the above equations, the area of the solution is still 0.

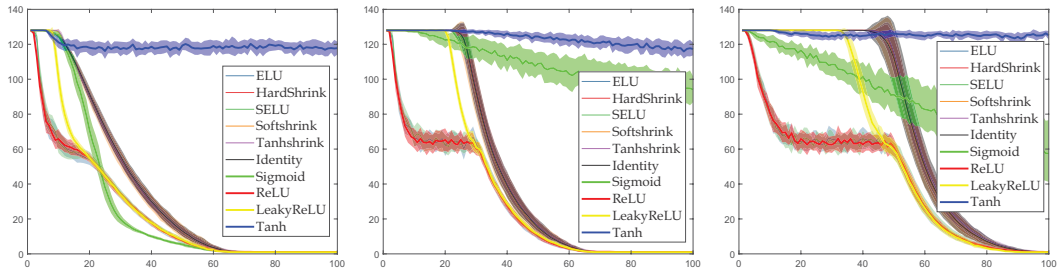
If $\boldsymbol{x}, \boldsymbol{y}$ are linearly independent, suppose $\mathrm{Tanh}(\boldsymbol{x}) = \boldsymbol{x}'$ is a vector in $\mathbb{R}^N$, and the space in $\mathbb{R}^N$ that can be transformed by pointwise Tanh to the same line as $\boldsymbol{x}'$ covers an area of probability 0 which is the same as the case of $\mathbb{R}^2$. Therefore, Lemma 2 still holds in $\mathbb{R}^N$. □

B Numerical Experiments on Synthetic Data

The goal of the experiments is to test which network structure with which kind of activation function has the potential to be extended to deep architecture. We measure this potential by the numerical rank of the output features in each hidden layer of the networks using synthetic data. The reason of choosing this measure can be explained by Theorem 2.2. We build the certain networks with depth 100 and the data is generated as follows.

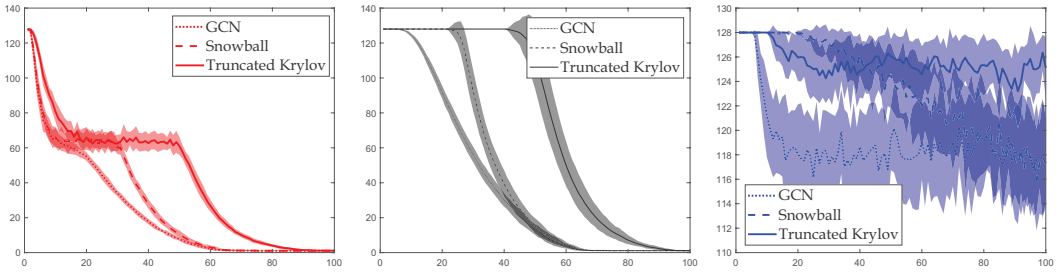
We first randomly generate edges of an Erdős-Rényi graph $G(1000, 0.01)$, *i.e.* the existence of the edge between any pair of nodes is a Bernoulli random variable with $p = 0.01$. Then, we construct the corresponding adjacency matrix A of the graph which is a $\mathbb{R}^{1000\times1000}$ matrix. We generate a $\mathbb{R}^{1000\times500}$ feature matrix $\boldsymbol{X}$ and each of its element is drawn from $N(0, 1)$. We normalize A and $\boldsymbol{X}$ as [17] and abuse the notation $A, \boldsymbol{X}$ to denote the normalized matrices. We keep 3 blocks in each layer of truncated block Krylov network. The number of input channel in each layer depends on the network structures and the number of output channel is set to be 128 for all networks. Each element in every parameter matrix W_i, $i = 1, \ldots, 100$ is randomly sampled from $N(0, 1)$ and the size is $\mathbb{R}^{\#input\times\#output}$. With the synthetic $A, \boldsymbol{X}, W_i$, we simulate the feedforward process according to the network architecture and collect the numerical rank (at most 128) of the output in each of the 100 hidden layers. For each activation function under each network architecture, we repeat the experiments for 20 times and plot the mean results with standard deviation bars.

C Rank Comparison of Activation Functions and Networks



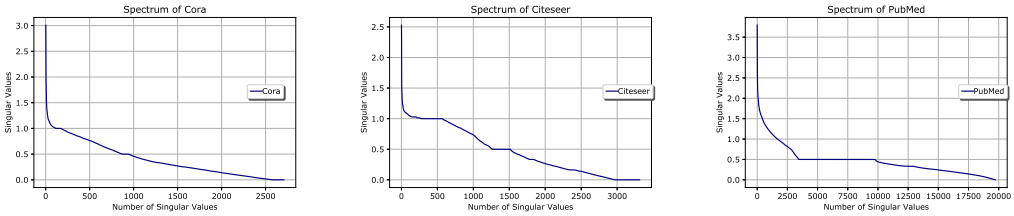
(a) GCN (b) Snowball (c) Truncated Block Krylov

Figure 4: Column ranks of different activation functions with the same architecture



(a) ReLU (b) Identity (c) Tanh

Figure 5: Column ranks of different architectures with the same activation function



(a) Cora (b) Citeseer (c) PubMed

Figure 6: Spectrum of the renormalized adjacency matrices for several datasets

D Spectrum of the Datasets

E Experiment Settings and Hyperparameters

The so-called public splits in [24] and the setting that randomly sample 20 instances for each class as labeled data in [36] is actually the same. Most of the results for the algorithms with validation are cited from [24], where they are reproduced with validation. However, some of them actually do not use validation in original papers and can achieve better results. In the paper, We compare with their best results.

We use NVIDIA apex amp mixed-precision plugin for PyTorch to accelerate our experiments. Most of the results were obtained from NVIDIA V100 clusters on Beluga of Compute-Canada, with minor part of them obtained from NVIDIA K20, K80 clusters on Helios Compute-Canada. The hyperparameters are searched using Bayesian optimization.

A useful tip is the smaller your training set is, the larger dropout probability should be set and the larger early stopping you should have.

Table 5 and Table 4 shows the hyperparameters to achieve the performance in the experiments, for cases without and with validation, respectively. When conducting the hyperparameter search, we encounter memory problems: current GPUs cannot afford deeper and wider structures. But we do observe better performance with the increment of the network size. It is expected to achieve better performance with more advanced deep learning devices.

Table 4: Hyperparameters for Tests with Validation

Architecture	Dataset	Split	Accuracy		Corresponding Hyperparameters					
			Our Best	SOTA	learning rate	weight decay	width	depth/blocks	dropout	optimizer
linear Snowball	**Cora**	0.5%	71.69	60.8	3.0911E-05	4.7732E-02	500	17	0.37106	RMSprop
		1%	75.20	67.5	4.0685E-04	8.9625E-03	100	12	0.67302	RMSprop
		3%	81.50	77.7	1.0256E-05	2.2773E-02	1900	9	0.8031	RMSprop
		5.2% (public)	83.65	83.0	7.2385E-05	2.4428E-02	1400	12	0.91054	RMSprop
	CiteSeer	0.5%	61.21	53.8	2.0858E-03	1.6430E-02	2800	3	0.98144	RMSprop
		1%	67.26	63.3	1.7237E-03	4.0416E-02	5000	2	0.98661	RMSprop
		3.6% (public)	73.39	72.5	1.1486E-03	4.7276E-02	2800	2	0.98167	RMSprop
	Pubmed	0.03%	71.74	61.0	2.5377E-03	1.0830E-02	1700	2	0.98733	RMSprop
		0.05%	73.17	68.8	4.7712E-03	1.8898E-03	400	2	0.89331	RMSprop
		0.1%	75.68	73.4	3.7998E-04	1.6507E-02	100	11	0.17388	RMSprop
		0.3% (public)	79.22	79.0	8.1370E-04	4.0998E-02	2400	3	0.98527	RMSprop
Snowball	**Cora**	0.5%	72.90	60.8	1.7430E-04	2.4378E-02	100	23	0.6086	RMSprop
		1%	76.16	67.5	3.4202E-03	1.5751E-03	3900	2	0.98907	RMSprop
		3%	81.12	77.7	3.4523E-05	1.1752E-02	3800	5	0.16643	RMSprop
		5.2% (public)	83.32	83.0	2.4800E-05	3.5866E-02	3800	6	0.91983	RMSprop
	CiteSeer	0.5%	62.41	53.8	2.3734E-03	2.2992E-02	2500	2	0.97315	RMSprop
		1%	67.04	63.3	1.9889E-03	2.2401E-02	700	4	0.82512	RMSprop
		3.6% (public)	73.23	72.5	4.1985E-03	8.9302E-03	3400	1	0.96857	RMSprop
	Pubmed	0.03%	70.87	61.0	1.4998E-03	2.4265E-02	500	11	0.93233	RMSprop
		0.05%	73.03	68.8	1.4754E-03	3.0558E-02	400	5	0.72253	RMSprop
		0.1%	76.09	73.4	4.2362E-04	3.3066E-02	400	4	0.090822	RMSprop
		0.3% (public)	79.51	79.0	4.8091E-03	1.3221E-03	2800	1	0.98994	RMSprop
truncated Krylov	**Cora**	0.5%	74.11	60.8	1.4387E-04	9.4404E-03	3700	86	0.94346	RMSprop
		1%	77.55	67.5	3.0239E-03	1.8363E-03	4500	32	0.98817	RMSprop
		3%	81.81	77.7	1.5997E-03	1.8666E-04	2200	14	0.9814	RMSprop
		5.2% (public)	83.51	83.0	8.9107E-04	3.4034E-03	500	30	0.049966	RMSprop
	CiteSeer	0.5%	65.28	53.8	3.3864E-03	4.9753E-02	4600	34	0.9842	RMSprop
		1%	68.21	63.3	1.5359E-03	1.1404E-02	3700	22	0.91228	RMSprop
		3.6% (public)	74.18	72.5	2.8980E-03	4.2862E-02	2200	19	0.98689	RMSprop
	Pubmed	0.03%	71.45	61.0	4.5592E-04	3.7077E-03	5000	7	0.98677	RMSprop
		0.05%	73.24	68.8	3.8475E-03	9.1710E-03	1600	7	0.98375	RMSprop
		0.1%	77.01	73.4	3.8877E-03	1.5528E-02	3200	6	0.97849	RMSprop
		0.3% (public)	80.12	79.0	1.4288E-03	1.6897E-02	4200	7	0.017084	RMSprop

Table 5: Hyperparameters for Tests without Validation

Architecture	Dataset	Percentage	Accuracy Ours	Accuracy SOTA	Correspondong Hyperparameters lr	weight_decay	hidden	layers/n_blocks	dropout	Optimizer
linear Snowball	**Cora**	0.5%	69.53	61.5	4.4438E-05	1.7409E-02	550	12	0.007753	Adam
		1%	74.12	69.9	1.0826E-03	3.3462E-03	1250	3	0.50426	Adam
		2%	79.43	75.9	2.4594E-06	9.6734E-03	1650	12	0.34073	Adam
		3%	80.41	78.5	2.8597E-05	3.4732E-02	900	15	0.039034	Adam
		4%	81.3	80.4	3.6830E-05	1.5664E-02	3750	4	0.93797	Adam
		5%	82.19	81.7	5.8323E-06	8.5940E-03	2850	5	0.14701	Adam
	CiteSeer	0.5%	56.76	56.1	4.5629E-03	2.0106E-03	300	3	0.038225	Adam
		1%	65.44	62.1	3.5530E-05	4.9935E-02	600	6	0.03556	Adam
		2%	68.78	68.6	6.1176E-06	3.0101E-02	1950	3	0.040484	Adam
		3%	71	70.3	2.1956E-05	4.3569E-02	3350	3	0.30207	Adam
		4%	72.23	70.8	9.1952E-05	4.6407E-02	3350	2	0.018231	Adam
		5%	72.21	71.3	3.7173E-03	1.9605E-03	2950	1	0.96958	Adam
	Pubmed	0.03%	64.133	62.2	1.0724E-03	8.1097E-03	64	4	0.8022	RMSProp
		0.05%	69.48	68.3	1.5936E-03	3.0236E-03	6	10	0.73067	RMSProp
		0.1%	72.93	72.7	4.9733E-03	1.3744E-03	128	3	0.91214	RMSProp
		0.3%	79.33	79.2	1.7998E-03	9.6753E-04	512	1	0.97483	RMSProp
Snowball	**Cora**	0.5%	67.15	61.5	9.8649E-04	1.0305E-02	1600	3	0.92785	Adam
		1%	73.47	69.9	1.4228E-04	1.3472E-02	100	13	0.68601	Adam
		2%	78.54	75.9	5.7111E-06	1.5544E-02	600	13	0.022622	Adam
		3%	79.97	78.5	4.0278E-05	2.7287E-02	4350	5	0.57173	Adam
		4%	81.49	80.4	1.4152E-05	2.3359E-02	2500	13	0.018578	Adam
		5%	81.82	81.7	1.2621E-03	1.5323E-02	3550	2	0.87352	Adam
	CiteSeer	0.5%	56.39	56.1	2.6983E-03	2.5370E-02	300	6	0.82964	Adam
		1%	65.04	62.1	1.6982E-03	1.5473E-02	2150	2	0.98611	Adam
		2%	69.48	68.6	9.7299E-05	4.9675E-02	2150	3	0.71216	Adam
		3%	71.09	70.3	1.7839E-04	3.0874E-02	2150	2	0.16549	Adam
		4%	72.32	70.8	5.6575E-05	3.5949E-02	4800	2	0.012576	Adam
		5%	72.8	71.3	2.8643E-04	1.6399E-02	2000	2	0.37308	Adam
	Pubmed	0.03%	62.94	62.2	1.2700E-03	1.4159E-03	128	4	0.76848	RMSProp
		0.05%	68.31	68.3	1.1224E-03	9.9166E-05	256	3	0.85496	RMSProp
		0.1%	73.29	72.7	6.0506E-04	1.0303E-03	256	2	0.97988	RMSProp
		0.3%	79.63	79.2	1.1416E-03	6.1543E-04	128	1	0.989	RMSProp
truncated Krylov	**Cora**	0.5%	72.96	61.5	3.3276E-03	1.0496E-04	128	18	0.76012	RMSProp
		1%	75.52	69.9	7.4797E-04	9.1736E-03	2048	20	0.98941	RMSProp
		2%	80.31	75.9	1.7894E-04	1.1079E-02	4096	16	0.97091	RMSProp
		3%	81.54	78.5	4.3837E-04	2.6958E-03	512	17	0.96643	RMSProp
		4%	82.47	80.4	3.6117E-03	4.1040E-04	64	25	0.021987	RMSProp
		5%	83.36	81.7	1.0294E-03	5.3882E-04	256	23	0.028392	RMSProp
	CiteSeer	0.5%	59.6	56.1	1.9790E-03	4.0283E-04	16	20	0.007761	RMSProp
		1%	65.95	62.1	7.8506E-04	8.2432E-03	64	24	0.28159	RMSProp
		2%	70.23	68.6	5.4517E-04	1.0818E-02	256	12	0.27027	RMSProp
		3%	71.81	70.3	1.4107E-04	5.0062E-03	1024	9	0.57823	RMSProp
		4%	72.36	70.8	4.8864E-06	1.8038E-02	4096	12	0.11164	RMSProp
		5%	72.24	71.3	2.1761E-03	1.1753E-02	5000	8	0.71473	Adam
	Pubmed	0.03%	69.07	62.2	6.8475E-04	2.8822E-02	4096	7	0.97245	RMSProp
		0.05%	71.77	68.3	2.3342E+04	2.2189E-03	1024	8	0.93694	RMSProp
		0.1%	76.07	72.7	4.2629E-04	4.1339E-03	2048	8	0.98914	RMSProp
		0.3%	80.04	79.2	2.2602E-04	3.3626E-02	2000	7	0.070573	Adam