Improve Neural Architecture Search in Graph Neural Networks with Sparse Coding

Peng, XU

June 4, 2022

Department of Computer Science and Engineering
The Chinese University of Hong Kong

Outline

Introduction & Related work

Background

NAS in Graph Neural Networks

- Sample-based
  - Prohibitive computational costs

  1. Reinforcement Learning-based [ZL17]
  2. Evolutionary Algorithm-based [JZS15]
- Weight-Sharing-based
  - Reduce the search effort by reusing the neural weights

  1. **Differential-based** [LSY19; Wu+19]

Differential-based NAS (DARTS)

Problem Formulation. DARTS concurrently optimizes for the best network architecture and the neural weights of the network.

Concretely, consider a classical DARTS framework defined by $\{\mathcal{A}, w, \alpha\}$, where $\mathcal{A}$, w, and α denote the search space, weights of neural networks, and the architecture parameter, respectively.

Differential-based NAS (DARTS)

DARTS addresses *two* subproblems of weights and architectures alternatively using a bi-level optimization framework [LSY19], including weight optimization (top) and architecture optimization (bottom) as shown in the following,

$$\begin{cases} w = \underset{w}{\operatorname{argmin}} \mathcal{L}_{train}(\alpha, w) \\ \alpha^* = \underset{\alpha}{\operatorname{argmax}} \mathcal{L}_{val}\left(w^*(\alpha), \alpha\right), \end{cases} \tag{1}$$

where $\mathcal{L}$ denotes a loss function (e.g., cross-entropy loss) on training and validating dataset w.r.t w, and α is the architecture parameter.

Differential-based NAS (DARTS)

DARTS [LSY19] further converts the discrete search space ($\mathcal{A}$) into a differentiable one with the help of the softmax,

$$\frac{\exp\left(\alpha_{\boldsymbol{o}}^{(i,j)}\right)}{\sum_{\boldsymbol{o}' \in \mathcal{O}} \exp\left(\alpha_{\boldsymbol{o}'}^{(i,j)}\right)} \tag{2}$$

where $\mathcal{O}$ represents the search space consisting of all operations $\boldsymbol{o}$. Therefore, we can get the importance of each operator in the format of probability.

Differential-based NAS (DARTS)

More formally, DARTS builds an aggregated representation of operators and then use it to output a prediction. Given such a representation, Eq. 1 can be reformulated to minimize Eq. 3.

$$\min_{w,\alpha} \sum_{i=1}^{N} \mathcal{L}\left\{y_i, o^*(w, x_i)\right\} + \rho_1 ||w||_2, \quad \text{where} \quad o^* = \sum_{i=1}^{K} \alpha_i \mathcal{O}_i, \tag{3}$$

where ρ_1 is a balance parameter for regularization and α_i is a scalar indicating the importance of operator i. Also, the total number of operators is denoted by K. [1]

[1] By optimizing alternatively, DARTS avoids the potential biased caused by imbalance dimensionality between neural weights w and architecture parameters—the former often has more elements.

Differential-based NAS (DARTS)

Optimization Difficulty. Under the general gradient descent framework, DARTS yields a result by optimizing Eq. 3. However, two optimization challenges cause instability of DARTS:

- **Optimization challenge due to BLO**: It is difficult to reach optimal state for objectives in the Eq. 1 simultaneously under the bi-level optimization framework;
- **Inaccurate estimation**: The softmax operation will keep the unimportant architecture parameter as non-zero values.

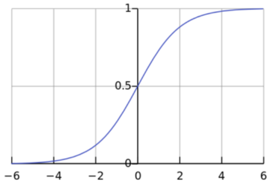


Figure 1: Softmax

Method

Motivation

In this work, we attempt to address the optimization difficulty in DARTS from a new perspective.

- It is easy to see that the ultimate goal of NAS is to assign the coefficients of unimportant operators as zeros and the important ones as non-zeros;
- Whether the neural weights reaches optimality may be less important if thinking outside the bi-level framework;
- Towards this, *searching the optimal architecture equals to find the optimal sparse combination of operators*, i.e., α.

Motivation

Notably, one of the most successful algorithms for this purpose is sparse coding [OF97], which has strong performance guarantees.

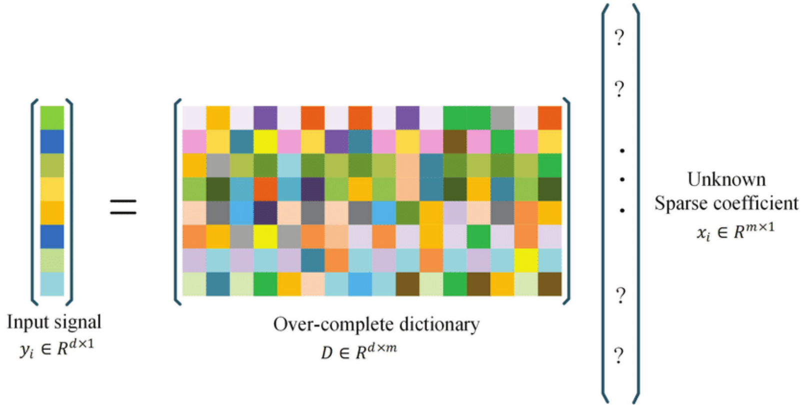


Figure 2: Sparse Coding

Motivation

Besides, we ask two key questions

- *Does updating weights contribute to optimal architectures searching?*
- *Could we instead not update the weights?*

Architecture Searching via Coding

Taking inspiration from sparse coding, we propose a new NAS algorithm called *neural architecture coding* (NAC) that updates model parameters as following:

$$\min_{w,\alpha} \sum_{i=1}^{N} \mathcal{L}\left\{y_i, \sum_{j=1}^{K} \mathcal{O}_j(w_j, \alpha_j, x_i)\right\} + \rho_1||w||_2 + \rho_2||\alpha||_1, \tag{4}$$

where ρ_1 and ρ_2 are balancing hyperparameter. [2]

[2] It is important to note that, other than computing the optimal $\boldsymbol{o}^*$ in classical DARTS explicitly (Eq. 3), we use $\sum_{j=1}^{K} \mathcal{O}_j(w_j, \alpha_j, x_i)$ as a proxy to approximate it and focus on optimizing the combination coefficient α, i.e. architecture parameters.

Architecture Searching via Coding

Three are threee benefits of NAC:

1. **Sparse Coding**: The sparsity regularization $||\alpha||_1$ allows us to rank the importance of operators directly without distortion as opposed to softmax that caused inaccurate gradient estimation.
2. **One-level Optimization**: By descending the dictionary and coding parameters on the same dataset, it can prevent the over-fitting issue due to the absence of validation processes as in a bi-level optimization.
3. **Interpretable Weight Sharing**: The perspective of dictionary learning and sparse coding makes the weight-sharing mechanism interpretable, where the weights can be viewed as bases in the dictionary to approximate a target signal.

Theoretical Analysis: Dictionary Orthogonality in NAC

What matters most when we reformulate the NAS problem as a sparse coding problem over a *dictionary* defined on operators? According to research in sparse coding (SC) [AEB06; Tro04; CRT06], its performance depends on the *quality* of the dictionary.

In particular, one essential property is mutual coherence or orthogonality.

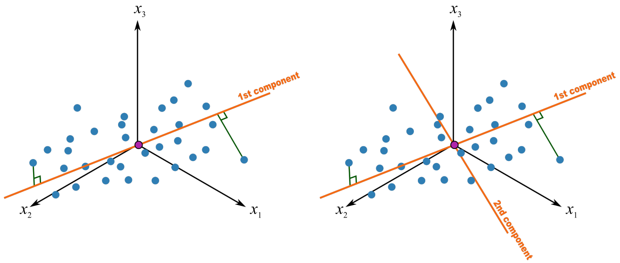


Figure 3: Orthogonality

Theoretical Analysis: Dictionary Orthogonality in NAC

We argue dictionaries that defined on deep neural networks naturally provides orthogonality. We first state our main theorem as below.

Theorem (NAC Theorem)
Suppose that for all atoms in a dictionary yielded by a deep neural network. Such a dictionary is often orthogonal.

This theorem can be proved by Central Limit Theorem and weak Law of Large Numbers. Please see Appendix for more details.

Theoretical Analysis: Dictionary Orthogonality in NAC

Our main theorem implies that the dictionaries from deep neural networks are qualified to learn architectures due to the built-in high dimensionality in deep neural networks, such as GNNs.

We now provide two versions of the dictionaries under this scheme.

- Case 1: Fixed random dictionary;
- Case 2: Adaptive dictionary;

Case 1: Fixed random dictionary

Since it is hard to guarantee the bias from updating weights to be eliminated, performance may suffer inevitably. Then, during training, rather than updating the weights to evaluate architectures, we directly measure their importance by implementing sparse coding on a fixed random dictionary, which is based on randomly initialized networks.

The objective is written as follows,

$$\min_{\alpha} \sum_{i=1}^{N} \mathcal{L} \left\{ y_i, \sum_{j=1}^{K} f[\mathcal{O}_j(w_j, \alpha_j, x_i)] \right\} + \rho_2 ||\alpha||_1. \tag{5}$$

Case 2: Adaptive dictionary

In this case, we update the dictionary during the training and then update sparse coefficients as a comparison to the above case.

The overall objective is as below:

$$\begin{cases} \min_w \sum_{i=1}^{N} \mathcal{L}\left\{y_i, \sum_{j=1}^{K} f[\mathcal{O}_j(w_j, \alpha_j, x_j)]\right\} + \rho_1 ||w||_2 \quad (a) \\ \min_\alpha \sum_{i=1}^{N} \mathcal{L}\left\{y_i, \sum_{j=1}^{K} f[\mathcal{O}_j(w_j, \alpha_j, x_j)]\right\} + \rho_2 ||\alpha||_1 \quad (b), \end{cases} \tag{6}$$

where these two are updated alternatively. Note that all parameters are updated in one dataset as opposed to two separated ones as in previous NAS methods. [3]

[3] Often the case, the dimension of w is way much greater than that of α, which is likely causing biased optimization. We therefore impose regularization on both variables to avoid this.

Realization and Details of LNAC and XNAC

In this work, we propose two realizations of NAC, namely LNAC and XNAC, where Fig. 4 outlines the core idea of both methods in one layer. [4] [5]

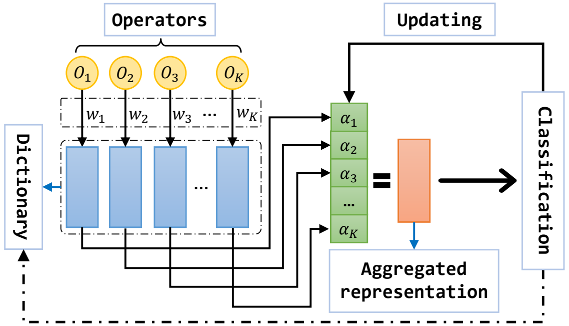


Figure 4: NAC framework

[4]LNAC directly learns the architecture α with a fixed dictionary.
[5]XNAC updates the dictionary in training, showing the additional process with a dash line.

Realization and Details of LNAC and XNAC

Algorithm 1 NAC algorithm

```
Require: The search space $\mathcal{A}$, dictionary initialization based on $w$.
Ensure: The architecture $a$
1: while $t = 1, \ldots, T$ do
2:     Compute the training loss $L_{train}$ on training data;
3:     Update $\alpha$ by taking gradient descending on $L_{train}(w, \alpha)$ w.r.t $\alpha$ under the
       fixed dictionary as in Eq.5;
4:     if use XNAC algorithm then
5:         Update the dictionary via updating $w$ by taking gradient descending
           on $L_{train}(w, \alpha)$ as in Eq.6 (a);
6:     end if
7: end while
8: Derive the final architecture $\{\alpha^*\}$ from the trained $\alpha$;
```

Experiments

Experiments

We evaluate the performance and properties of the proposed NAC to answer the following research questions:

- How does NAC perform in comparison to the baselines regarding the design of GNNs?
- How stable is NAC during the searching process?
- How does the convergence of NAC behave?

Experiment Setup

Datasets

We performed experiments on three benchmark datasets: Cora, CiteSeer, PubMed, following the data partition setting (training/validation/testing) as in [ZYT21].

Search space

We select a list of node aggregators (i.e., operators) in GNNs to form the search space. More specifically, we have the following: multi-layer perceptrons (MLP), GCN [KW17], GAT [Vel+18], GIN [XHLJ19], Geniepath [Liu+19], Graphsage [HYL17], ChebNet [DBV16]. [6] [7]

[6] Note that MLP is a special aggregator with fixed frequency when viewing GNNs from a spectrum perspective [Bal+20].

[7] Besides, Graphsage and GAT have different aggregation functions, such as maximum and LSTM, we apply all these variations as operators in the experiments.

Experiment Setup

Baselines
We include multiple NAS approaches with different search strategies (i.e., reinforcement learning methods, SOTA DARTS methods in our experiment. Specifically, the baselines include:

- SANE [ZYT21]
- GraphNAS [Gao+20]
- GraphNAS-WS [ZYT21]

Experiment Setup

Evaluation Metrics

The proposed approaches aim to boost the ability to learn efficacious architecture for downstream tasks. For each method, we report the accuracy and running time, which are widely used in NAS research [Zha+21].

- We use classification accuracy as the evaluation metric. [8]
- We report the running time to compare the efficiency of different methods. [9] [10]

[8]Note that all reported results are from the top-1 ranked architectures.
[9]We set the running time of NAC, SANE, and GraphNAS as the time they take to obtain the final architectures with 100 epochs.
[10]The running time of RS and Bayesian is fixed to the time of 100 times trial-and-error processes.

The comparisons between NAC and state-of-the-art methods

Table 1: The comparisons between state-of-the-art NAS-GNN models and the proposed NAC, including linear NAC (LNAC) and non-linear NAC (XNAC). Our method demonstrates competitive performance with much less time consumed, achieving one to two orders of magnitude faster than others. (RL: reinforcement learning; WS: weight sharing.)

		CiteSeer		Cora		PubMed	
		Acc(%)	Time (h)	Acc(%)	Time (h)	Acc(%)	Time (h)
RL	GraphNAS	66.82	4.23	69.93	4.09	83.80	22.73
	GraphNAS-WS	69.67	1.34	70.11	2.69	85.24	16.09
DARTS	SANE	73.34	0.07	84.26	0.17	87.80	1.79
NAC	LNAC	**75.00**	**0.02**	**88.52**	**0.02**	**89.01**	**0.15**
	XNAC	74.40	0.07	87.59	0.06	88.94	0.43

Performance Analysis

As shown in Table. 1, [11] [12] [13]

- NAC is superior to all baselines regarding both *accuracy* and *speed*.
  - More specifically, LNAC has about $2\% - 4\%$ improvement over the best baseline, i.e., SANE, and about $6\% - 18\%$ improvement over GraphNAS, in terms of the classification accuracy.
  - Our superiority in speed is more striking, which is about $10\times$ faster than SANE and about 200 time faster than the reinforcement learning-based methods.

[11]We compare NAC with the state-of-the-art baselines on several benchmark datasets.
[12]Due to the flexible definition of dictionaries, we try three different initializations, including normal, uniform and orthogonal style.
[13]We report the best results of NAC methods under these three different initialization strategies.

Performance Analysis

It is also assured that regardless of the initialization, NAC consistently outperforms the baselines. We present the result of different initialization strategies in Table. 2 for ablation.

Table 2: All three initializations yield close performance (i.e., below 1% difference), showing that NAC is robust to initialization.

		CiteSeer Acc(%)	Cora Acc(%)	PubMed Acc(%)
Normal	LNAC	**74.70**	**87.96**	**88.56**
	XNAC	74.09	87.59	85.50
Uniform	LNAC	73.64	**88.52**	**89.01**
	XNAC	**74.40**	87.04	88.20
Orthogonal	LNAC	**75.00**	**88.33**	**89.01**
	XNAC	74.40	87.59	88.94

Performance Analysis

LNAC outperforms XNAC in all cases.

- LNAC demonstrates consistent advantages over XNAC in both accuracy and speed.
  - The non-updating strategy allows LNAC to have much fewer parameters to optimize and thus has a much faster speed.
  - Updating weight inevitably introduces biased information due to the challenge to reach optimality in deep neural networks, which misleads the optimization of architecture learning in XNAC.[14]

Table 3: The comparisons between NAC (LNAC) and non-linear NAC (XNAC).

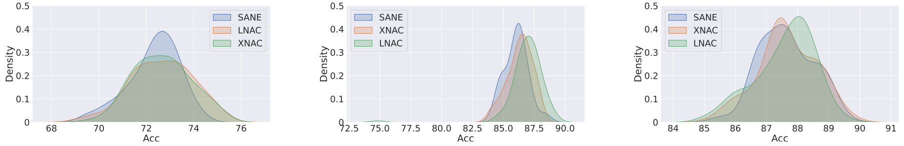
		CiteSeer		Cora		PubMed	
		Acc(%)	Time (h)	Acc(%)	Time (h)	Acc(%)	Time (h)
NAC	LNAC	**75.00**	**0.02**	**88.52**	**0.02**	**89.01**	**0.15**
	XNAC	74.40	0.07	87.59	0.06	88.94	0.43

[14]Thus, we recommend the use of LNAC over XNAC.

Search Space Analysis

We analyze methods' behaviors in searching to better understand the difference between ours and others.

To illustrate this, we apply kernel density estimation (KDE) [DLP11] to describe the quality of searching. The probability is given by the area under the density function but above the horizontal axis and within a given range. [15]



(a) CiteSeer (b) Cora (c) PubMed

Figure 5: The search space quality distribution comparison of NAC and SANE on benchmark datasets.

[15]In other words, larger areas indicate higher probabilities in the same range.

Analysis of Convergence

A notable benefit of the NAC framework is its guaranteed convergence from a sparse coding perspective. To verify this, Fig. 6 offers a convergence comparison between NAC and SANE from the Pubmed dataset, including the loss in the training and testing stages.

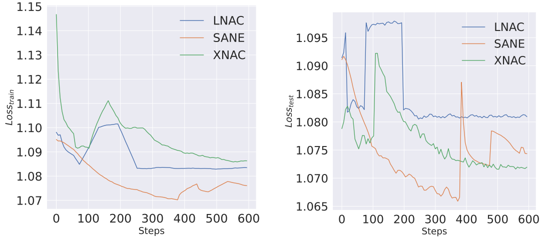


Figure 6: Comparison of SANE and NAC on loss values from training (left) and testing (right) on Pubmed data.

Conclusion

Contributions

- **Problem Formulation:**
  We show how to reformulate the NAS problem into a sparse coding problem, leading to the *first linear search complexity(backward) NAS algorithm* with theoretical guarantee.

- **Effectiveness and Efficiency**
  Our proposed method, NAC, shows superior performance in both accuracy and speed when compared with state-of-the-art baselines. NAC brings up to 6% improvement in terms of accuracy and is $100\times$ faster than baselines.

Reference I

[1] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in *ICLR*, 2017.

[2] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures,” in *ICML*, 2015.

[3] H. Liu, K. Simonyan, and Y. Yang, “DARTS: differentiable architecture search,” in *ICLR*, 2019.

[4] B. Wu *et al.*, “Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search,” in *CVPR*, 2019.

Reference II

[5] B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by v1?” *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997, ISSN: 0042-6989. DOI: `https://doi.org/10.1016/S0042-6989(97)00169-7`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0042698997001697`.

[6] M. Aharon, M. Elad, and A. Bruckstein, “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006. DOI: `10.1109/TSP.2006.881199`.

Reference III

[7] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004. DOI: 10.1109/TIT.2004.834793.

[8] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006. DOI: 10.1109/TIT.2005.862083.

[9] H. Zhao, Q. Yao, and W. Tu, "Search to aggregate neighborhood for graph neural network," in *ICDE*, 2021.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

Reference IV

[11] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[12] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" In *ICML*, 2019.

[13] Z. Liu *et al.*, "Geniepath: Graph neural networks with adaptive receptive paths," in *AAAI*, 2019.

[14] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.

[15] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NeurIPS*, 2016.

Reference V

[16] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine, "Analyzing the expressive power of graph neural networks in a spectral perspective," in *ICLR*, 2020.

[17] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graph neural architecture search.," in *IJCAI*, 2020.

[18] X. Zhang, Z. Huang, N. Wang, S. Xiang, and C. Pan, "You only search once: Single shot neural architecture search via direct sparse optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2891–2904, 2021.

[19] R. A. Davis, K.-S. Lii, and D. N. Politis, "Remarks on some nonparametric estimates of a density function," in *Selected Works of Murray Rosenblatt*, 2011, pp. 95–100.

Reference VI

[20] H. Fischer, *A history of the central limit theorem. From classical to modern probability theory*. 2011, ISBN: 978-0-387-87856-0. DOI: `10.1007/978-0-387-87857-7`.

[21] D. ter Haar, “Mathematical foundations of statistical mechanics. a. i. khinchin.,” *Science*, vol. 110, no. 2865, pp. 570–570, 1949. DOI: `10.1126/science.110.2865.570.b`.

Appendix

Appendix

Theorem (NAC Theorem)
Suppose that for all atoms in a dictionary yielded by a deep neural network.
Such a dictionary is often orthogonal.

Theoretical Analysis: Dictionary Orthogonality in NAC

Proof.
Given a dictionary $\boldsymbol{H} \in R^{n \times K}$, we have it mutual coherence computed as follows,

$$\varphi = \max_{\boldsymbol{h}_i, \boldsymbol{h}_j \in \boldsymbol{H}, i \neq j} \left| \left\langle \frac{\boldsymbol{h}_i}{\|\boldsymbol{h}_i\|_2}, \frac{\boldsymbol{h}_j}{\|\boldsymbol{h}_j\|_2} \right\rangle \right|, \tag{7}$$

where $\varphi \in [0, 1]$, and $\langle \cdot \rangle$ denotes inner product. The minimum of φ is reached for an orthogonal dictionary, and the maximum for a dictionary containing at least two collinear atoms (columns), which are 0 and 1, respectively.

Let $E_i = \frac{\boldsymbol{h}_i}{\|\boldsymbol{h}_i\|_2}$ and $E_j = \frac{\boldsymbol{h}_j}{\|\boldsymbol{h}_j\|_2}$, by Central Limit Theorem [Fis11], we know that $\langle E_i, E_j \rangle / \sqrt{n}$ converges to a normal distribution, i.e.,

$$\langle E_i, E_j \rangle = \lim_{n \to \infty} \sqrt{n} Z, \tag{8}$$

where Z is a standard normal distribution. Consider $\bar{E}$ as the mean value of all $\langle E_i, E_j \rangle$.

□

Theoretical Analysis: Dictionary Orthogonality in NAC

Proof.
With weak law of large numbers (a.k.a Khinchin's law) [Haa49], for any positive number ε, the probability that sample average $\bar{E}$ greater than ε converges 0 is written as

$$\lim_{n \to \infty} \Pr\left(\left|\bar{E}\right| \geq \varepsilon\right) = 0 \tag{9}$$

This implies that the probability that the inner product of E_i and E_j is greater than ε close to zero if $n \to \infty$. In other words, the probability that E_i and E_j are nearly orthogonal goes to 1 when their dimensionality is high. Therefore, the coherence of this dictionary reaches the minimum at a high dimensionality that holds for deep neural networks naturally. $\square$