ONE-SHOT NEURAL ARCHITECTURE SEARCH VIA COMPRESSIVE SENSING

Minsu Cho[1], Mohammadreza Soltani[2], and Chinmay Hegde[1]
[1] New York University, [2] Duke University
mc8065@nyu.edu, mohammadreza.soltani@duke.edu, chinmayh@nyu.edu

ABSTRACT

Neural Architecture Search remains a very challenging meta-learning problem. Several recent techniques based on parameter-sharing idea have focused on reducing the NAS running time by leveraging proxy models, leading to architectures with competitive performance compared to those with hand-crafted designs. In this paper, we propose an iterative technique for NAS, inspired by algorithms for learning low-degree sparse Boolean functions. We validate our approach on the DARTs search space (Liu et al., 2018b) and NAS-Bench-201 (Yang et al., 2020). In addition, we provide theoretical analysis via upper bounds on the number of validation error measurements needed for reliable learning, and include ablation studies to further in-depth understanding of our technique.

1 INTRODUCTION

The choice of a suitable neural network architecture for complex prediction tasks such as image classification often requires substantial trial-and-error. Recently, there has been a growing interest to *automatically* select the architecture of neural networks that can achieve competitive (or better) results over hand-designed architectures. Neural architecture search (NAS) tries to automate this hand-design process by constructing competitive architectures with as small computational budgets as possible (Zoph et al., 2018; Liu et al., 2018a;b; Xie et al., 2018; Bender et al., 2018; Li & Talwalkar, 2019).

In a departure from traditional methods, we approach the NAS problem via the lens of *compressive sensing*. The field of compressive sensing (or sparse recovery), introduced by the seminal works of Candes et al. (2006); Donoho et al. (2006), has received significant attention in ML community over the last decade and has influenced advances in nonlinear and combinatorial optimization. Here, we develop a new NAS method called CoNAS (Compressive sensing-based Neural Architecture Search), which merges ideas from sparse recovery with the so-called "one-shot" architecture search methods (Bender et al., 2018; Li & Talwalkar, 2019). Our contribution is twofold. First, CoNAS uses a new *search space* that permits exploration of a large(r) number of diverse candidate architectures. Second, it utilizes a new *search strategy* that borrows ideas from the recovery of Boolean functions from their (sparse) Fourier expansions. We show how a combination of these two ideas leads to improved NAS performance.

2 CONAS: A NEW NAS APPROACH

Overview. Our proposed algorithm, Compressive sensing-based Neural Architecture Search (CoNAS), combines ideas from learning a sparse graph (Boolean Fourier analysis) and one-shot NAS. As we mentioned above, CoNAS consists of two components: a newly defined search space, and a more practical search strategy. Before discussing these two parts, we review some basics.

Fourier analysis of Boolean functions. We follow the treatment given in O'Donnell (2014). A real-valued Boolean function is one that maps n-bit binary vectors (i.e., vertices of a hypercube) to a real number: $f : \{-1, 1\}^n \to \mathbb{R}$. Such functions can be represented in a basis comprising real multilinear polynomials called the *Fourier* basis, defined as follows. (We denote the vectors with bold letters. Also, $[n]$ denotes the set $\{1, 2 \ldots, n\}$. Hence, the power set of $[n]$ is denoted by $2^{[n]}$.

Definition 2.1. *For $S \subseteq [n]$, define the parity function $\chi_S : \{-1,1\}^n \to \{-1,1\}$ such that $\chi_S(\boldsymbol{\alpha}) = \prod_{i \in S} \alpha_i$. Then, the Fourier basis is the set of all 2^n parity functions $\{\chi_S\}$.*

The key fact is that the basis of parity functions forms an K-bounded orthonormal system (BOS) with $K = 1$, therefore satisfying two properties:

$$\langle \chi_S, \chi_T \rangle = \begin{cases} 1, & \text{if } S = T \\ 0, & \text{if } S \neq T \end{cases} \qquad \text{and} \qquad \sup_{\boldsymbol{\alpha} \in \{-1,1\}^n} |\chi_S(\boldsymbol{\alpha})| \leq 1 \quad \text{for all } S \subseteq [n], \tag{2.1}$$

Due to the orthonormality, any Boolean function f has a unique Fourier representation, given by $f(\boldsymbol{\alpha}) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(\boldsymbol{\alpha})$, with Fourier coefficients $\hat{f}(S) = \mathbb{E}_{\boldsymbol{\alpha} \in \{-1,1\}^n}[f(\boldsymbol{\alpha}) \chi_S(\boldsymbol{\alpha})]$ where expectation is taken with respect to the uniform distribution over the vertices of a hypercube.

A modeling assumption is that the Fourier spectrum of the function is concentrated on monomials of small degree ($\leq d$). This corresponds to the case where f is a decision tree (Hazan et al., 2017), and allows us to simplify the Fourier expansion by limiting its support. Let $\mathcal{P}_d \subseteq 2^{[n]}$ be a fixed collection of Fourier basis such that $\mathcal{P}_d \coloneqq \{\chi_S : S \subseteq 2^{[n]}, |S| \leq d\}$ Then $\mathcal{P}_d$ induces a function space consisting of all functions of order d or less, denoted by $\mathcal{H}_{\mathcal{P}_d} \coloneqq \{f : Supp[\hat{f}] \subseteq \mathcal{P}_d\}$. For example, $\mathcal{P}_2$ allows us to express the function f with at most $\sum_{l=0}^{d} \binom{n}{l} \equiv \mathcal{O}(n^2)$ Fourier coefficients.

Algorithm 1 Pseudocode: CoNAS

Inputs: Number of measurements m, number of coefficients s, lasso parameter λ
Train one-shot model (f) with parameter sharing (RSPS)
while not enough measurements ($k < m$) **do**
  Randomly sample a sub-architecture binary encoded vector α_k
  Collect test loss, $\mathbf{y} = \{L(f(\alpha_1)), \ldots, L(f(\alpha_k))\}$
  $k \leftarrow k + 1$
end while
Construct graph sampling matrix $\mathbf{A}$ from $\{\alpha_1, \ldots, \alpha_m\}$
Solve $\mathbf{u}^* = \arg\min_{\mathbf{u}} \|\mathbf{y} - \mathbf{A}\mathbf{u}\|_2^2 + \lambda \|\mathbf{u}\|_1$.
Approximate $g \approx f$ with s absolutely largest coefficients from $\mathbf{u}^*$
Compute minimizer $\mathbf{z} = \arg\min_{\alpha} g(\alpha)$ via brute force.
return Constructed cell from $\mathbf{z}$

Lastly, if we have a prior knowledge of some set of bits J, we use an operation called *restriction*.

Definition 2.2. *Let $f : \{-1,1\}^n \to \mathbb{R}$, $(J, \overline{J})$ be a partition of $[n]$, and $z \in \{-1,1\}^{\overline{J}}$. The restriction of f to J using z denoted by $f_{J|z} : \{-1,1\}^J \to \mathbb{R}$ is the subfunction of f given by fixing the coordinates in $\overline{J}$ to the bit values z.*

We now discuss two main components of CoNAS.

1-Search Space. Following the approach of DARTs (Liu et al., 2018b), we define a directed acyclic graph (DAG) with all predecessor nodes are connected to every intermediate node with all possible operations. We represent any sub-graph of the DAG using a binary string $\boldsymbol{\alpha}$ called the *architecture encoder*. Its length is the total number of edges in the DAG, excluding the edges connect to the output node. A 1 (resp. -1) in $\boldsymbol{\alpha}$ indicates an active (resp. inactive) edge.

Figure 1 gives an example of how the architecture encoder $\boldsymbol{\alpha}$ samples the sub-architecture of the fully-connected model in the case of a convolutional neural network. The goal of CoNAS is to find the "best" encoder $\boldsymbol{\alpha}^*$, which is "close enough" to the best achievable validation accuracy by constructing the final model with $\boldsymbol{\alpha}^*$ encoded sub-graph.

2-Search Strategy. We propose a compressive measuring strategy to approximate the one-shot model with a Fourier-sparse Boolean function. Let $f : \{-1,1\}^n \to \mathbb{R}$ map the sub-graph of the one-shot pre-trained model encoded by $\boldsymbol{\alpha}$ to its validation performance. We collect a small number of function evaluations of f and reconstruct the Fourier-sparse function $g \approx f$ via a sparse recovery algorithm with randomly sampled measurements (test loss). Then, we solve $\arg\min_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha})$ by exhausting enumeration over all coordinates, supports of the Boolean function g. We obtain a cell to construct the final architecture by activating the edges such that $\alpha_i^* = 1$. If the solution does not return enough edges, restricting the hypercube allows the iterative process to find more edges. From Definition 2.2, we restrict the approximate function g by fixing bit values found in the previous solution and repeat the sparse recovery by randomly sampling sub-graph in the remaining edges.

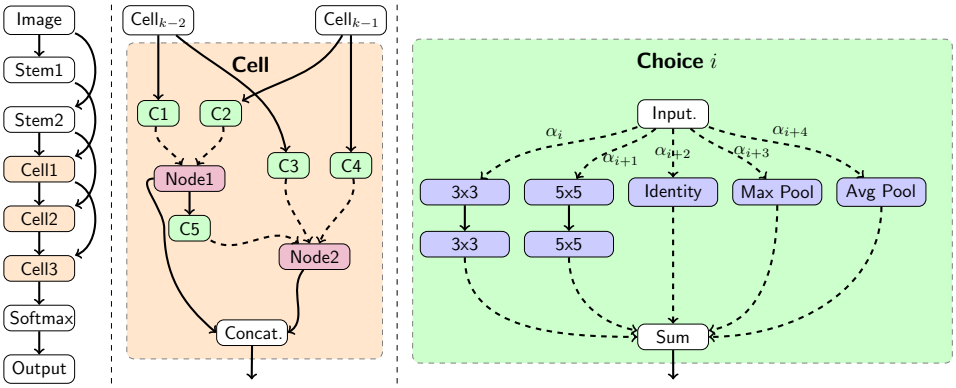

Figure 1: *Diagram inspired by Bender et al. (2018). The example architecture encoder $\boldsymbol{\alpha}$ samples a sub-architecture for $N = 5$ nodes (two intermediate nodes) with five different operations. Each component in $\boldsymbol{\alpha}$ maps to the edges one-to-one in all* Choice *blocks in a cell. Since the CNN search space finds both* normal cell *and* reduce cell*, the length of $\boldsymbol{\alpha}$ is equivalent to $(2 + 3) \cdot 5 \cdot 2 = 50$.*

Full Algorithm. We now describe CoNAS in detail, with pseudocode provided in Algorithm 1.

We first train a parameter-shared one-shot model (Li & Talwalkar, 2019) with standard backpropagation; however, we only update the weights corresponding to the randomly sampled sub-graph edges for each minibatch. Then, we randomly sample sub-graphs by generating architecture encoder strings $\boldsymbol{\alpha} \in \{-1, 1\}^n$ from some distribution. (e.g., a $Bernoulli(p)$ distribution for each bit of $\boldsymbol{\alpha}$ independently).

In the second stage, we collect m measurements of randomly sampled sub-architecture performance denoted by $\mathbf{y} = (L(f(\boldsymbol{\alpha}_1)), L(f(\boldsymbol{\alpha}_2)), \ldots, L(f(\boldsymbol{\alpha}_m)))^T$. Next, we construct the *graph-sampling matrix* $\mathbf{A} \in \{-1, 1\}^{m \times |\mathcal{P}_d|}$ with entries

$$\mathbf{A}_{l,k} = \chi_{S_k}(\boldsymbol{\alpha}_l), \qquad l \in [m], k \in [|\mathcal{P}_d|], S \subseteq [n], |S| \leq d, \tag{2.2}$$

where d is the maximum degree of monomials in the Fourier expansion, and S_k is the index set corresponding to the k$^{\text{th}}$ Fourier basis. We solve the familiar Lasso problem (Tibshirani, 1996):

$$\mathbf{u}^* = \arg\min_{\mathbf{u} \in \mathbb{R}^{|\mathcal{P}_d|}} \|\mathbf{y} - \mathbf{A}\mathbf{u}\|_2^2 + \lambda \|\mathbf{u}\|_1, \tag{2.3}$$

to (approximately) recover the global optimizer $\mathbf{u}^*$, the vector contains the Fourier coefficients corresponding to $\mathcal{P}_d$. We define an approximate function $g \approx f$ with Fourier coefficients with the top-s (absolutely) largest coefficients from $\mathbf{u}^*$, and solve $\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha})$ by computing all the possible points (brute force) in the subcube defined by the support of g (this computation is feasible if s is small). Finally, we obtain a cell to construct the final architecture by activating the edges corresponding to all $i \in [n]$ such that $\alpha_i^* = 1$.

Theoretical support for CoNAS. We first note that a system of linear equations given by $\mathbf{y} = \mathbf{A}\mathbf{u}$ with the graph-sampling matrix $\mathbf{A} \in \{-1, 1\}^{m \times O(n^d)}$, measurements $\mathbf{y} \in \mathbb{R}^m$, and Fourier coefficient vector $\mathbf{u} \in \mathbb{R}^{O(n^d)}$ is an ill-posed problem when $m \ll O(n^d)$ for large n. However, if the graph-sampling matrix satisfies *Restricted Isometry Property (RIP)*, the sparse coefficients, $\mathbf{u}$ can be uniquely recovered.

Definition 2.3. *A matrix $\mathbf{A} \in \mathbb{R}^{m \times \mathcal{O}(n^d)}$ satisfies the restricted isometry property of order s with some constant δ if for every s-sparse vector $\mathbf{u} \in \mathbb{R}^{\mathcal{O}(n^d)}$ (i.e., only s entries are non-zero) the following holds:*

$$(1 - \delta)\|\mathbf{u}\|_2^2 \leq \|\mathbf{A}\mathbf{u}\|_2^2 \leq (1 + \delta)\|\mathbf{u}\|_2^2.$$

To the best of our knowledge, the best known result with mild dependency on δ (i.e., δ^{-2}) is due to Haviv & Regev (2017), which we can apply for our setup. It is easy to check that the graph-sampling matrix $\mathbf{A}$ in our proposed CoNAS algorithm satisfies BOS for $K = 1$ (equation 2.2). We defer the proof in Appendix A.3.

Theorem 2.4. *Let the graph-sampling matrix $A \in \{-1, 1\}^{m \times \mathcal{O}(n^d)}$ be constructed by taking m rows (random sampling points) uniformly and independently from the rows of a square matrix $\mathbf{M} \in$*

Table 1: ***Comparison with existing NAS methods on CIFAR-10 DARTs search space***

Method	Test Error (%)	Params (M)	Multi-Add (M)	Search (GPU day)
NASNet-A	2.65	3.3	-	2000
GDAS	2.82	2.5	-	0.17
SNAS	2.85 ± 0.02	2.8	-	1.5
ENAS	2.89	4.6	-	0.45
DARTs	2.76 ± 0.09	3.3	548	4
Random Search	3.29 ± 0.15	3.1	-	4
RSPS	$2.71 \vert 2.85 \pm 0.08$	3.7	634	2.7
CoNAS (1)	$2.59 \vert 2.67 \pm 0.06$	3.2	450	0.35
CoNAS (2)	$2.73 \vert 2.85 \pm 0.08$	2.7	430	0.35

Table 2: ***Comparison with existing NAS methods on NAS-Bench-201***

Method	Search	CIFAR-10		CIFAR-100		ImageNet-16-120	
	(seconds)	validation	test	validation	test	validation	test
DARTs	35781.80	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
GDAS	31609.80	89.89±0.08	93.61±0.09	71.34±0.04	70.70±0.30	41.59±1.33	41.71±0.98
ENAS	14058.80	37.51±3.19	53.89±0.58	13.37±2.35	13.96±2.33	15.06±1.95	14.84±2.10
RSPS	8007.13	80.42±3.58	84.07±3.61	52.12±5.55	52.31±5.77	27.22±3.24	26.28±3.09
CoNAS	8222.80	88.40±2.79	91.22±3.08	65.82±5.72	66.39±5.51	39.51±6.95	38.82±7.01

$\{-1, 1\}^{\mathcal{O}(n^d) \times \mathcal{O}(n^d)}$. *Then the normalized matrix* $\mathbf{A}$ *with* $m = \mathcal{O}(\log^2(\frac{1}{\delta})\delta^{-2} s \log^2(\frac{s}{\delta}) d \log(n))$ *with probability at least* $1 - 2^{-\Omega(d \log n \log(\frac{s}{\delta}))}$ *satisfies the restricted isometry property of order* s *with constant* δ*; as a result, every* s*-sparse vector* $\mathbf{u} \in \mathbb{R}^{\mathcal{O}(n^d)}$ *can be recovered from the sample* y_i*'s* $\mathbf{y} = \mathbf{A}\mathbf{u} = \left(\sum_{j=1}^{|\mathcal{O}(n^d)|} u_j \mathbf{A}_{i,j}\right)_{i=1}^{m}$ *by solving the LASSO problem in (equation 2.3).*

In essence, Theorem 2.4 provides a successful guarantee for recovering the optimal sub-network of a given size given a sufficient number of performance measurements.

3 Experiments and Results

In this section, we illustrate the efficacy of our proposed approach through some experimental results. We start by experimenting on two popular NAS benchmarks: (i) a CNN search on CIFAR-10 on DARTs (Liu et al., 2018b) search space, (ii) a CNN search on NAS-Bench-201 (Yang et al., 2020) search space. We compare CoNAS with the following NAS algorithms in DARTs search space: NASNet (Zoph et al., 2018), SNAS (Xie et al., 2018), ENAS (Pham et al., 2018), DARTs (Liu et al., 2018b), random search from Liu et al. (2018b), and RSPS (Li & Talwalkar, 2019). Our evaluation setup for training the final architecture is the same as one reported by DARTs and NAS-Bench-201. We defer the search/evaluation setup details and ablation studies to the section **??** in the Appendix. The implementation of CoNAS is available from this link[1].

DARTs Search Space We run CoNAS in two different settings: (1) cells from the optimization problem (equation 2.3) and (2) the sub-graph from (1) to strictly match DARTs cell by manually dropping some operations (up to 2 edges for each intermediate nodes). We conduct the final training with exact hyperparameter setups used in DARTs. The average test error of our experiment uses five random seeds. Figure 2 in the Appendix includes the cells found from CoNAS. Our approach achieves the test error of 2.59% surpassing the RSPS, DARTs, and random search.

NAS-Bench-201 Search Space We evaluate our algorithm on NAS-Bench-201, a NAS benchmark designed for all cell-based NAS methods. We implement our algorithm on top of the existing Random Search with Parameter Sharing (RSPS) implementation in the NAS-Bench-201 library. We randomly sample sub-graphs from a trained one-shot model and solve the optimization problem (equation 2.3). If the solution picks more than one operation (an edge) between the nodes, we randomly choose an operation uniformly from the given solution. The average test error on NAS-Bench-201 uses three different random seeds. Table 2 compares cell-based NAS methods on NAS-Bench-201. We observe that both approaches consistently find a better cell structure than RSPS.

[1] https://github.com/chomd90/CoNAS_release

ACKNOWLEDGEMENTS

MC and CH are supported in part by NSF grants CCF-2005804 and CCF-1815101.

REFERENCES

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. *Proc. Int. Conf. Machine Learning*, 2018.

Jean Bourgain. An improved estimate in the restricted isometry problem. *Geometric Aspects of Functional Analysis: Israel Seminar (GAFA)*, pp. 65, 2014.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *Proc. Int. Conf. Learning Representations*, 2019.

Emmanuel Candes, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 2006.

Emmanuel J Candes. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, 346(9-10):589–592, 2008.

Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inform. Theory*, 2006.

Mahdi Cheraghchi, Venkatesan Guruswami, and Ameya Velingker. Restricted isometry of fourier matrices and list decodability of random linear codes. *SIAM Journal on Computing*, 2013.

David L Donoho et al. Compressed sensing. *IEEE Trans. Inform. Theory*, 2006.

Simon Foucart and Holger Rauhut. A mathematical introduction to compressive sensing. *Bull. Am. Math*, 54:151–165, 2017.

Ishay Haviv and Oded Regev. The list-decoding size of fourier-sparse boolean functions. *ACM Transactions on Computation Theory (TOCT)*, 8(3):10, 2016.

Ishay Haviv and Oded Regev. The restricted isometry property of subsampled fourier matrices. *Geometric Aspects of Functional Analysis*, pp. 163–179, 2017.

Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: a spectral approach. *arXiv preprint arXiv:1706.00764*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conf. Comp. Vision and Pattern Recog*, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. *IEEE Conf. Comp. Vision and Pattern Recog*, 2017.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pp. 240–248, 2016.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*, 2019.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *Euro. Conf. Comp. Vision*, 2018a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *Proc. Int. Conf. Machine Learning*, 2018b.

Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *Proc. Int. Conf. Machine Learning*, 2018.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *Proc. Assoc. Adv. Art. Intell. (AAAI)*, 2019.

Mark Rudelson and Roman Vershynin. On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 2008.

Peter Stobbe and Andreas Krause. Learning fourier sparse set functions. *Proc. Int. Conf. Art. Intell. Stat. (AISTATS)*, 2012.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. Nas evaluation is frustratingly hard. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HygrdpVKvr`.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *Proc. Int. Conf. Learning Representations*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *IEEE Conf. Comp. Vision and Pattern Recog*, 2018.

A Appendix

A.1 Search/Evaluation Protocols and Hyperparameter Setup

We conduct the exact equivalent evaluation training protocol of DARTs for the fair comparison. We adopt a NAS-Bench-201 API for the NAS-Bench-201 search space, which provides all available cells' precomputed performance, including test loss and accuracy.

Search Protocol in DARTs Search Space We train a one-shot architecture equivalent to RSPS with a cell containing $N = 7$ nodes with two nodes as inputs and one node as output. We randomly sample sub-graphs by generating architecture encoder strings $\alpha \in \{-1, 1\}^n$ using a $Bernoulli(p)$ distribution for each bit of α independently (we set $p = 0.5$). We set the threshold for gathering measurements (loss) to remove abnormally high loss values. We use seven operations: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated convolutions, 3×3 average pooling, 3×3 max pooling, and Identity. We equally divide the 50,000-sample training set into training and validation sets, following Li & Talwalkar (2019) and Liu et al. (2018b).

Search Protocol in NAS-Bench-201 Search Space NAS-Bench-201 provides the predefined search space with four operations (five operations including a zero operation) and four nodes. We uniformly random sample the sub-graph from all possible cells in the search space.

Table 3 and Table 4 include the training protocol and hyperparameters used in CoNAS on DARTs search space and NAS-Bench-201, respectively.

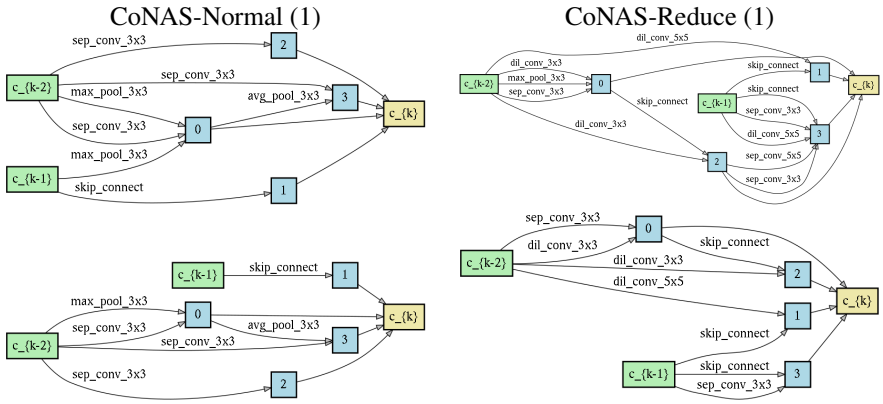

Figure 2: The cell in the first (resp. second) row corresponds to the CoNAS(1) (resp. CoNAS(2)) results in Table 1. CoNAS(2) is the sub-graph of CoNAS(1) which strictly meets the requirement for DARTs search space (only up to two edges allowed for intermediate nodes).

Table 3: DARTs search space searching hyperparameter set.

optimizer	SGD	initial LR	0.025
Nesterov	Yes	ending LR	0.001
momentum	0.9	LR schedule	cosine
weight decay	0.0003	epoch	100
batch size	64	initial channel	16
cells #	8	cutout	No
ops #	7	nodes #	7
random flip	p=0.5	random crop	Yes
normalization	Yes	threshold	1.5
measurements	3000	lasso (λ)	10.0
coefficient # (s)	12	degree (d)	2

Table 4: NAS-Bench-201 search space searching hyperparameter set.

optimizer	SGD	initial LR	0.025
Nesterov	Yes	ending LR	0.001
momentum	0.9	LR schedule	cosine
weight decay	0.0005	epoch	100
batch size	64	initial channel	16
cells #	5	cutout	No
ops #	4	nodes #	4
random flip	p=0.5	random crop	Yes
normalization	Yes	threshold	1.8
measurements	300	lasso (λ)	5.0
coefficient # (s)	14	degree (d)	2

Evaluation Protocol in DARTs Search Space We evaluate the cell found from CoNAS with exactly equivalent training protocol to DARTs. We use either NVIDIA Quadro RTX 8000 or NVIDIA TITAN RTX for the final architecture training process (1 GPU training). Table 5 shows the final evaluation protocol details. We set random seeds from 0 to 4 for the average test error on Table 1.

Table 5: DARTs search space final evaluation hyperparameter set.

optimizer	SGD	initial LR	0.025
Nesterov	Yes	ending LR	0
momentum	0.9	LR schedule	cosine
weight decay	0.0003	epoch	600
batch size	96	initial channel	36
random flip	p=0.5	random crop	Yes
normalization	Yes	cutout	Yes
drop-path	0.2	cutout length	16
cells #	20	aux weight	0.4
grad clip	5.0	parallel training	No

A.2 NAS Literature

Neural Architecture Search. Early NAS approaches used RL-based controllers (Zoph et al., 2018; Pham et al., 2018), evolutionary algorithms (Real et al., 2019), or sequential model-based optimization (SMBO) (Liu et al., 2018a), and showed competitive performance with manually-designed

architectures such as deep ResNets (He et al., 2016) and DenseNets (Huang et al., 2017). However, these approaches required substantial computational resources, running into thousands of GPU-days. Subsequent NAS works have focused on boosting search speeds by proposing novel search strategies, such as differentiable search technique via gradient-based optimization (Cai et al., 2019; Liu et al., 2018b; Xie et al., 2018) and random search via sampling sub-networks from a one-shot super-network (Bender et al., 2018; Li & Talwalkar, 2019). To the best of our knowledge, no NAS method yet reported has explored compressive sensing techniques.

Differentiable Neural Architecture Search (DARTs). Our CoNAS approach can be viewed as a refinement to DARTs ((Liu et al., 2018b)) which performs bilevel optimization by relaxing the (discrete) architecture search space to a differentiable search space via softmax operations. The choice of alternative optimization on differentiable multi-objective formulation substantially speeds up the search by orders of magnitude while achieving competitive performance compared to previous works (Zoph & Le, 2017; Zoph et al., 2018; Real et al., 2019; Liu et al., 2018a).

One-Shot Neural Architecture Search. Bender et al. (2018) provide an extensive experimental analysis on one-shot architecture search based on weight-sharing. Bender et al. (2018) statistically showed the correlation between the one-shot model (super-graph) and stand-alone model (sub-graph) through the experiments. Li & Talwalkar (2019) proposes simplified training procedures without stabilizing techniques (e.g., path dropout schedule on a direct acyclic graph (DAG) and ghost batch normalization) from Bender et al. (2018). As the final performance of the discovered architecture heavily relies on hyperparameter settings, Li & Talwalkar (2019) exactly accords hyperparameters and data augmentation techniques to DARTs for their experiments. This combination of random search via one-shot models with weight-sharing provides the best competitive baseline results reported in the NAS literature. Our CoNAS approach improves upon these reported results.

Learning Sub-Networks. Stobbe & Krause (2012) propose learning sparse sub-networks from a small number of random cuts; they also leverage ideas from compressive sensing and provide theoretical upper bounds for successful recover. Our CoNAS approach is directly inspired from their seminal work. However, we emphasize essential differences: while Stobbe & Krause (2012) emphasize *linear* measurements, CoNAS takes a different perspective by focusing on measurements that map sub-networks to performance, which are fundamentally *nonlinear*. Moreover, our theoretical bounds use better Fourier-RIP bounds, and lead to improved results in terms of measurement complexity.

Hyperparameter optimization. Building upon the approach of Stobbe & Krause (2012), Hazan et al. (2017) develop a spectral approach called *Harmonica* for hyperparameter optimization (HPO) by encoding hyperparameters as binary strings. CoNAS also follows the same path, albeit for NAS. While NAS and HPO are sister meta-learning problems, we emphasize that our focus is exclusively on NAS, while Hazan et al. (2017) exclusively focus on HPO.

Moreover, the techniques of Hazan et al. (2017) cannot be directly applied to the NAS problem. We need to define our search space, encode our search problem in terms of Boolean variables, and propose how to gather measurements. All these are new to our paper: in particular, CoNAS proposes gathering measurements within tractable sampling time via top of RSPS, while Harmonica naively gathers the approximated measurements by training the model for each randomly sampled hyperparameter choice. Finally, Harmonica requires invocation of a baseline hyperparameter optimization method (such as random search, successive halving (Jamieson & Talwalkar, 2016), or Hyperband (Li et al., 2017)), which CoNAS does not require.

A.3 THEORETICAL SUPPORT FROM COMPRESSIVE SENSING

The system of linear equations $\mathbf{y} = \mathbf{A}\mathbf{u}$ with the graph-sampling matrix $\mathbf{A} \in \{-1, 1\}^{m \times O(n^d)}$, measurements $\mathbf{y} \in \mathbb{R}^m$, and Fourier coefficient vector $\mathbf{u} \in \mathbb{R}^{O(n^d)}$ is an ill-posed problem when $m << O(n^d)$ for large n. However the sparse coefficients $\mathbf{u}$ can be recovered if the graph-sampling matrix satisfies *Restricted Isometry Property* from Definition 2.3.

There has been significant research during the last decade in proving upper bounds on the number of rows of bounded orthonormal dictionaries (matrix $\mathbf{A}$) for which $\mathbf{A}$ is guaranteed to satisfy the restricted isometry property with high probability. One of the first BOS results was established by Candes & Tao (2006), where the authors proved an upper bound scales as $\mathcal{O}(sd^6 \log^6 n)$ for a

subsampled Fourier matrix. While this result is seminal, it is only optimal up to some *polylog* factors. In fact, the authors in chapter 12 of Foucart & Rauhut (2017) have shown a necessary condition (lower bound) on the number of rows of BOS which scales as $\mathcal{O}(sd \log n)$. In an attempt to achieve to this lower bound, the result in Candes & Tao (2006) was further improved by Rudelson & Vershynin (2008) to $\mathcal{O}(sd \log^2 s \log(sd \log n) \log n)$. Motivated by this result, Cheraghchi et al. (2013) has even reduced the gap further by proving an upper bound on the number of rows as $\mathcal{O}(sd \log^3 s \log n)$. The best known available upper bound on the number of rows appears to be $\mathcal{O}(sd^2 \log s \log^2 n)$; however with worse dependency on the constant δ, i.e., δ^{-4} (please see Bourgain (2014)). To the best of our knowledge, the best known result with mild dependency on δ (i.e., δ^{-2}) is due to Haviv & Regev (2017), and is given by $\mathcal{O}(sd \log^2 s \log n)$. We have used this result for proving Theorem 2.4.

Theorem A.1. *Let the graph-sampling matrix $A \in \{-1, 1\}^{m \times \mathcal{O}(n^d)}$ be constructed by taking m rows (random sampling points) uniformly and independently from the rows of a square matrix $\mathbf{M} \in \{-1, 1\}^{\mathcal{O}(n^d) \times \mathcal{O}(n^d)}$. Then the normalized matrix $\mathbf{A}$ with $m = \mathcal{O}(\log^2(\frac{1}{\delta})\delta^{-2} s \log^2(\frac{s}{\delta}) d \log(n))$ with probability at least $1 - 2^{-\Omega(d \log n \log(\frac{s}{\delta}))}$ satisfies the restricted isometry property of order s with constant δ; as a result, every s-sparse vector $\mathbf{u} \in \mathbb{R}^{\mathcal{O}(n^d)}$ can be recovered from the sample y_i's $\mathbf{y} = \mathbf{A}\mathbf{u} = \left(\sum_{j=1}^{|\mathcal{O}(n^d)|} u_j \mathbf{A}_{i,j} \right)_{i=1}^{m}$ by LASSO (equation 2.3).*

Proof. First, we note that the graph-sampling matrix A is a BOS matrix with $K = 1$; hence, directly invoking Theorem 4.5 of Haviv & Regev (2016) to our setting, we can see that matrix A satisfies RIP. Now according to Theorem 1.1 of Candes (2008), letting $\delta < \sqrt{2} - 1$, the ℓ_1 minimization or LASSO will recover exactly the s sparse vector u. For instance, in our experiments, we have selected $m = 3000$ which is consistent with our parameters, $d = 2, s = 14, n = 196$. □

A.4 Ranking Correlation of RSPS.

We leverage Random Search with Parameter Sharing (RSPS) to collect measurements quickly for sparse recovery with the Fourier basis of Boolean functions. We examine the correlation between performance estimations from parameter-shared models and performance from isolated training from scratch with concatenated cells We first randomly sample 32 cells from a 100-epoch trained one-shot model (same search setup described in Appendix A.1) with RSPS method on the CIFAR-10 dataset. Then we collect two sets of test losses to observe ranking correlation via Kendall tau correlation between one-shot model and macro-skeleton trained with 160 epochs for each randomly sampled cell. We matched the initial number of channels (16) and depth (8) of micro skeleton (equivalent to number of cells) to the one-shot model. We observed the correlation between one-shot model and macro-skeleton is weak with Kendall tau $\tau = 0.15$ with the two-sided p-value 0.23. Figure 3 visually represents the rank correlation between one-shot model prediction and actual trained model in test loss. We point out that other factors that exist weakening the correlation, such as the same architecture training with different seeds, affect the ranking correlation (Yang et al., 2020) and high variance of CIFAR-10 results even with exact training protocol (Liu et al., 2018a).

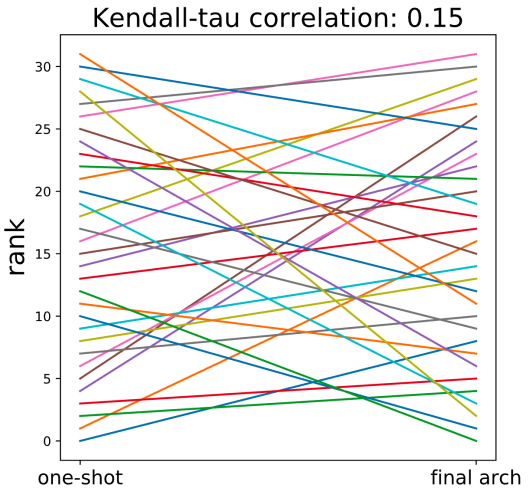

Figure 3: Ranking difference between one-shot model and actual architecture. The higher the Kendall-tau correlation, the more each line parallel to the x-axis.

A.5 Stability on Lasso Parameters

We check our algorithm's stability on lasso parameter by observing the solution given exact same measurements. Denote $\alpha^*_{\lambda=l}$ as the architecture encoded output from CoNAS given $\lambda = l$. We compare the hamming distance and the test error between $\alpha^*_{\lambda=1}$ and other λ values ($\lambda = 0.5, 2, 5, 10$).

We train a parameter-shared one-shot network with five operations resulting the encoder length $(2 + 3 + 4 + 5) * 5 * 2 = 140$. The average support of the solution from one sparse recovery is 15 out of the 140 length. The average hamming distance between two randomly generated binary strings with $\text{supp}(\alpha^*) = 15$ from $100,000$ samples was 27.58 ± 1.82. Our experiment shows a stable performance under various lasso parameters with small hamming distances regards to various λ. Also we measure the average test error with 150 training epochs on different λ values as shown in Table 6. For the baseline comparison, we compare CoNAS solutions with the randomly chosen architecture with 15 operations.

Table 6: Lasso Parameter Stability Experiment.

Criteria	$\lambda = 0.5$	$\lambda = 2.0$	$\lambda = 5.0$	$\lambda = 10.0$	Random
Hamming Distance	0	0	8	12	29
Test Error (%)	3.74 ± 0.07	3.74 ± 0.07	3.51 ± 0.06	3.62 ± 0.04	4.43 ± 0.08
Param (M)	2.3	2.3	2.6	2.6	2.7
Multiply-Add (M)	386	386	455	449	444