Novelty Driven Evolutionary Neural Architecture Search

Nilotpal Sinha
nilotpalsinha.cs06g@nctu.edu.tw
National Yang Ming Chiao Tung University
Hsinchu City, Taiwan R.O.C

Kuan-Wen Chen
kuanwen@cs.nctu.edu.tw
National Yang Ming Chiao Tung University
Hsinchu City, Taiwan R.O.C

ABSTRACT

Evolutionary algorithms (EA) based neural architecture search (NAS) involves evaluating each architecture by training it from scratch, which is extremely time-consuming. This can be reduced by using a supernet for estimating the fitness of an architecture due to weight sharing among all architectures in the search space. However, the estimated fitness is very noisy due to the co-adaptation of the operations in the supernet which results in NAS methods getting trapped in local optimum. In this paper, we propose a method called NEvoNAS wherein the NAS problem is posed as a multi-objective problem with 2 objectives: *(i)* maximize architecture novelty, *(ii)* maximize architecture fitness/accuracy. The novelty search is used for maintaining a diverse set of solutions at each generation which helps avoiding local optimum traps while the architecture fitness is calculated using supernet. NSGA-II is used for finding the *pareto optimal front* for the NAS problem and the best architecture in the pareto front is returned as the searched architecture. Exerimentally, NEvoNAS gives better results on 2 different search spaces while using significantly less computational resources as compared to previous EA-based methods. The code for our paper can be found here.

CCS CONCEPTS

• **Computing methodologies** → **Distributed artificial intelligence**; **Heuristic function construction**; **Computer vision**.

KEYWORDS

Neural architecture search, supernet, novelty search, multi-objective optimization

ACM Reference Format:
Nilotpal Sinha and Kuan-Wen Chen. 2022. Novelty Driven Evolutionary Neural Architecture Search. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528889

1 INTRODUCTION

In the recent years, convolutional neural networks (CNNs) have been very instrumental in solving various computer vision problems. However, the CNN architectures (such as ResNet [14], DenseNet [15] AlexNet [17], VGGNet [33]) have been designed mainly by humans, relying on their intuition and understanding of the specific problem. Searching the neural architecture automatically by using an algorithm, i.e. *Neural architecture search* (NAS), is an alternative to the architectures designed by humans, and in the recent years, these NAS methods have attracted increasing interest because of its promise of an automatic and efficient search of architectures specific to a task. Vanilla NAS methods [13][44][45] have shown promising results in the field of computer vision but most of these methods consume a huge amount of computational power as it involves training each architecture from scratch for its evaluation. Vanilla evolutionary algorithm (EA)-based NAS methods also suffers from the same huge computational requirement problem. For example, the method proposed in [30] required 3150 GPU days of evolution.

NAS Methods	Trial 1	Trial 2	Trial 3	Trial 4
DARTS†[24]	97.08	97.23	97.0	96.95
EvNAS‡[34]	97.19	97.39	96.93	97.04
Random Search†[21]	97.04	96.67	97.17	97.0

Table 1: Quality of the architectures found in 4 trials for different NAS methods using supernet. † represents results report in [21] while ‡ represents re-run.

Recently proposed gradient-based methods such as [24] [11][40] have reduced the search time by sharing weights among the architectures through the use of supernet. However, the supernet suffers from inaccurate performance estimation [2]. This results in premature convergence to the local optimum [4][41]. In order to mitigate this problem, NAS methods using supernet run the algorithm multiple times and select the best architecture out of the multiple runs. This can be thought of as running the algorithm multiple times in order to get a set of good quality neural architectures. This is illustrated in Table 1, which shows the quality of the searched architecture in terms of test accuracy on CIFAR-10 dataset for gradient based method DARTS[24], EA-based method EvNAS[34] and random search[21] in 4 trials. These multiple trials end up increasing the compuational costs.

In this paper, we propose a method called NEvoNAS (*Novelty Driven Evolutionary Neural Architecture Search*), in which the algorithm is run only once to get a set of good quality neural architecture solutions. This is achieved by posing the NAS problem as a multi-objective problem with 2 objectives: *(i)* maximize architecture novelty, and *(ii)* maximize architecture fitness/accuracy. Maximizing architecture novelty (i.e. *novelty search*) is used for maintaining a diverse set of solutions at each generation which helps avoiding local optimum traps while maximizing the architecture fitness using supernet guides the search towards potential solutions. We used NSGA-II for finding the *pareto optimal front* of the multi-objective

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3528889

NAS problem and the best architecture in the discovered pareto optimal front is returned as the searched architecture.

Our contributions can be summarized as follows:

- We propose a novelty metric called *architecture novelty metric* which determines how novel a neural architecture is from the already discovered neural architectures.
- We pose the NAS problem as a multi-objective problem with the objective of maximizing both the architecture novelty metric and architecture fitness.

2 PROPOSED METHOD

2.1 Search Space and Performance Estimation

We follow [24] to create the architecture by staking together 2 types of cells: *normal* cells which preserve the dimentionality of the input with a stride of one and *reduction* cells which reduce the spatial dimension with a stride of two. A cell in the architecture is represented by an *architecture parameter*, α which is a matrix with columns representing the weights of different operations $Op(.)s$ from the operation space O (i.e. the search space of NAS) and rows representing the edge between two nodes.

We used a *supernet* [24] to estimate the performance of an architecture in the search space. It shares the weights among all architectures in the search space by treating all the architectures as the subgraphs of a supergraph. This design choice allows us to skip the individual architecture training from scratch for its evaluation because of the weight-sharing nature of the supernet, thus resulting in a significant reduction of search time. The performance of an architecture is calculated by first selecting the architecture in the supernet and then calculating the performance of the supernet on the validation data, also known as the *fitness* of the architecture.

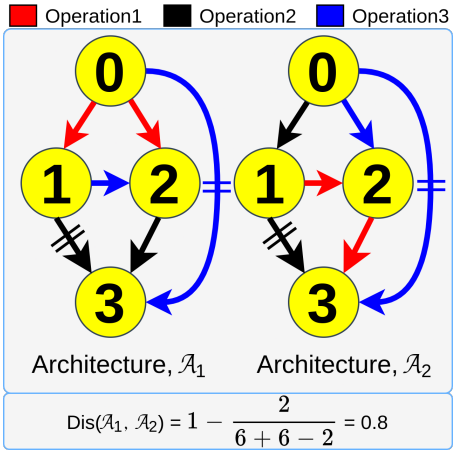

Figure 1: Illustration of architecture dissimilarity metric calculation of two architectures $\mathcal{A}_1$ and $\mathcal{A}_2$. The common edges between $\mathcal{A}_1$ and $\mathcal{A}_2$ is shown by two small parallel lines on the edges between the two nodes that are common in both architectures.

2.2 Architecture Novelty Metric

In order to create an EA algorithm that rewards novel architecture, we need a *novelty metric* that measures how different an architecture is from another architecture. This provides a constant pressure to generate new architecture. In the neural architecture space, we first define a *similarity metric*, $Sim(\mathcal{A}_1, \mathcal{A}_2)$, which measures how similar an architecture $\mathcal{A}_1$ to another architecture $\mathcal{A}_2$ and is given as follows:

$$Sim(\mathcal{A}_1, \mathcal{A}_2) = \frac{\cap(\mathcal{A}_1, \mathcal{A}_2)}{n(\mathcal{A}_1) + n(\mathcal{A}_2) - \cap(\mathcal{A}_1, \mathcal{A}_2)} \tag{1}$$

Where $\cap(\mathcal{A}_1, \mathcal{A}_2)$ refers to the number of common operations between 2 nodes present in the given architectures, $\mathcal{A}_1, \mathcal{A}_2$, $n(\mathcal{A}_1)$ and $n(\mathcal{A}_2)$ refer to total number of opration edges present between nodes in the $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. Note that $Sim(\mathcal{A}_1, \mathcal{A}_2)$ equals to 1 if $\mathcal{A}_1$ and $\mathcal{A}_2$ are the same architecture (i.e. $\mathcal{A}_1 = \mathcal{A}_2$) and $Sim(\mathcal{A}_1, \mathcal{A}_2)$ equals to 0 if $\mathcal{A}_1$ and $\mathcal{A}_2$ do not share any operations between 2 nodes (i.e. completely different architectures). Thus, $0 \leq Sim(\mathcal{A}_1, \mathcal{A}_2) \leq 1$. Now, we define an *dissimilarity metric*, $Dis(\mathcal{A}_1, \mathcal{A}_2)$, which is used for measuring how different an architecture $\mathcal{A}_1$ is from another architecture $\mathcal{A}_2$ and is given as follows:

$$Dis(\mathcal{A}_1, \mathcal{A}_2) = 1 - Sim(\mathcal{A}_1, \mathcal{A}_2) \tag{2}$$

Note that $Dis(\mathcal{A}_1, \mathcal{A}_2)$ equals to 0 if $\mathcal{A}_1$ and $\mathcal{A}_2$ are the same architecture (i.e. $\mathcal{A}_1 = \mathcal{A}_2$) and $Dis(\mathcal{A}_1, \mathcal{A}_2)$ equals to 1 if $\mathcal{A}_1$ and $\mathcal{A}_2$ do not share any operations between 2 nodes (i.e. completely different architectures). Thus, $0 \leq Dis(\mathcal{A}_1, \mathcal{A}_2) \leq 1$. For illustration, in Figure 1, the architectures $\mathcal{A}_1$ and $\mathcal{A}_2$ have two common edges between nodes $(0, 3)$ and $(1, 3)$, thus $\cap(\mathcal{A}_1, \mathcal{A}_2) = 2$, while both $n(\mathcal{A}_1)$ and $n(\mathcal{A}_2)$ are equal to 6. So, the dissimilarity metric comes out to be 0.8.

The novelty of a newly generated neural architecture is computed with respect to an archive of past generated neural architectures and current population of neural architecture. To get the novelty of neural architecture, we need a novelty metric [19] which characterizes how far the neural architecture is from its predecessors and the rest of the population in the neural architectural space. We define *architecture novelty metric* as the mean dissimilarity metric of the k-nearest neighbors, which is given as follows:

$$F_{nov}(\mathcal{A}) = \frac{1}{k} \sum_{i=1}^{k} Dis(\mathcal{A}, \mathcal{A}_i) \tag{3}$$

Where $\mathcal{A}_i$ is the ith-nearest neighbor of the neural architecture $\mathcal{A}$ in terms of the dissimilarity metric. The nearest neighbors are calculated from the archive of past neural architectures and the current population.

2.3 NEvoNAS

Multi-objective optimization is a popular branch of *evolutionary computation* (EC), which involves optimizing problems with more than one objective function simultaneously. NEvoNAS poses the NAS problem as a multi-objective problem with two objectives: *(i)* maximize architecture novelty, *(ii)* maximize architecture fitness. The architecture novelty is calculated using the architecture novelty metric (discussed in Section 2.2) and the fitness of the architecture is calculated using the supernet. In order to solve the multi-objective

Table 2: Comparison of NEvoNAS with other NAS methods in S1 in terms of test accuracy (higher is better) on CIFAR-10, CIFAR-100 and ImageNet.

Architecture	CIFAR-10			CIFAR-100			ImageNet				Search
	Top-1 (%) **Acc.** (%)	Params (M)	**GPU Days**	**Top-1** (%) **Acc.** (%)	Params (M)	**GPU Days**	**Test Accuracy (%)** **Top-1**	**Top-5**	Params (M)	+×	**Method**
ResNet[14]	95.39	1.7	-	77.90	1.7	-	-	-	-	-	manual
DenseNet-BC[15]	96.54	25.6	-	82.82	25.6	-	-	-	-	-	manual
ShuffleNet[43]	90.87	1.06	-	77.14	1.06	-	-	-	-	-	manual
PNAS[22]	96.59	3.2	225	80.47	3.2	225	74.2	91.9	5.1	588	SMBO
RSPS[21]	97.14	4.3	2.7	-	-	-	-	-	-	-	random
NASNet-A[45]	97.35	3.3	1800	-	-	-	74.0	91.6	5.3	564	RL
ENAS[29]	97.14	4.6	0.45	80.57	4.6	0.45	-	-	-	-	RL
DARTS[24]	97.24	3.3	4	-	-	-	73.3	91.3	4.7	574	gradient
GDAS[11]	97.07	3.4	0.83	-	-	-	74.0	91.5	5.3	581	gradient
SNAS[40]	97.15	2.8	1.5	-	-	-	72.7	90.8	4.3	522	gradient
SETN[10]	97.31	4.6	1.8	-	-	-	74.3	92.0	5.4	599	gradient
AmoebaNet-A[30]	96.66	3.2	3150	81.07	3.2	3150	74.5	92.0	5.1	555	EA
Large-scale Evo.[31]	94.60	5.4	2750	77.00	40.4	2750	-	-	-	-	EA
CNN-GA[38]	96.78	2.9	35	79.47	4.1	40	-	-	-	-	EA
AE-CNN[37]	95.7	2.0	27	79.15	5.4	36	-	-	-	-	EA
NSGANetV1-A2[27]	97.35	0.9	27	82.58	0.9	27	-	-	-	-	EA
AE-CNN+E2EPP[36]	94.70	4.3	7	77.98	20.9	10	-	-	-	-	EA
NSGA-NET[26]	97.25	3.3	4	79.26	3.3	8	-	-	-	-	EA
EN^2AS[42]	97.39	3.1	3	-	-	-	-	-	-	-	EA
NEvoNAS-C10A	**97.46**	**3.4**	**0.35**	-	-	-	74.8	92.1	4.9	541	**EA**
NEvoNAS-C100A	-	-	-	**83.95**	**3.9**	**0.3**	**75.7**	**92.7**	**5.4**	**598**	**EA**

problem, we used NSGA-II [6], a well-known Pareto-based Multi-objective Evolutionary Algorithm (MOEA).

The entire process is summarized in the supplementary. NEvoNAS starts with initializing the population randomly, the supernet with random weights and an empty *archive*. In each generation, the supernet is trained on the training data. During training, α of each individual architecture in the population is copied to the supernet in a round-robin fashion for each training batch. Then, the fitness of each individual architecture in the population, F_{acc}, is calculated using the supernet. Next, the novelty of each individual architecture in the population, F_{nov}, is calculated with respect to the *archive* of past neural architectures and the current population of neural architectures. Th *archive* is then updated to include the new individual architectures from the current population. NSGA-II is then used to generate the next generation population. The entire process runs for $\mathcal{G}$ generations. NEvoNAS returns a pareto optimal front, $P_{optimal}$, (i.e. set of possible neural architecture solution) and the best neural architecture in the front is returned as the searched architecture. Note that NEvoNAS runs for only once to get a set of possible solutions unlike other NAS methods using supernet [24][34][21].

3 EXPERIMENTS

In this section, we report the performance of NEvoNAS in terms of a neural architecture search on the search space used in [24] *Search space 1 (**S1**)*. We performed architecture searches on both CIFAR-10 and CIFAR-100 with different random number seeds; their results are provided in Table 2. The results show that the cells discovered by NEvoNAS on CIFAR-10 and CIFAR-100 achieve better results than those by human designed, RL based, gradient-based and EA-based methods. On comparing the computation time (or *search cost*) measured in terms of *GPU days*, we found that NEvoNAS performs the architecture search in significantly less time as compared to other EA-based methods while giving better search results. GPU days for any NAS method is calculated by multiplying the number of GPUs used in the NAS method by the execution time (reported in units of days). We followed [24] to compare the transfer capability of NEvoNAS with that of the other NAS methods, wherein the discovered architecture on a dataset was transferred to another dataset (i.e. ImageNet) by retraining the architecture from scratch on the new dataset. So, the discovered architectures from the architecture search on CIFAR-10 and CIFAR-100 (i.e. NEvoNAS-C10A and NEvoNAS-C100A) are then evaluated on the ImageNet dataset in mobile setting and the results are provided in Table 2. The results show that the cells discovered by NEvoNAS on CIFAR-10 and CIFAR-100 can be successfully transferred to ImageNet, achieving better results than those of human designed, RL based, gradient based and EA based methods while using significantly less computational resources.

4 CONCLUSION

The goal of this paper was to mitigate the noisy fitness estimation nature of the supernet which forces NAS methods using supernet to run multiple times to get a set of neural architecture solutions. We resolve this by posing the NAS problem as a multi-objective problem with two objectives of maximizing the architecture novelty (i.e. *novelty search*) and maximizing the architecture fitness. This results in a pareto optimal front which provides a set of good quality neural architecture solutions in a single run, thus, reducing the computational requirements. Experimentally, NEvoNAS reduced the search time of EA-based search methods significantly while achieving better results in S1 search space.

ACKNOWLEDGMENTS

This work was supported in part by the *Ministry of Science and Technology of Taiwan* (MOST 110-2628-E-A49-012-, MOST 110-2634-F-A49-006-, and MOST 111-2420-H-369-001-). Furthermore, we are grateful to the *National Center for High-performance Computing* for computer time and facilities.

REFERENCES

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2017. Designing Neural Network Architectures using Reinforcement Learning. *International Conference on Learning Representations* (2017).

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*. PMLR, 550–559.

[3] J. Blank and K. Deb. 2020. pymoo: Multi-Objective Optimization in Python. *IEEE Access* 8 (2020), 89497–89509.

[4] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*. 1294–1303.

[5] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. 2017. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819* (2017).

[6] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[7] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. 2007. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th annual conference on genetic and evolutionary computation*. 1187–1194.

[8] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: a large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255.

[9] Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (2017).

[10] Xuanyi Dong and Yi Yang. 2019. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE International Conference on Computer Vision*. 3681–3690.

[11] Xuanyi Dong and Yi Yang. 2019. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*. 1761–1770.

[12] Xuanyi Dong and Yi Yang. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HJxyZkBKDr

[13] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2018. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377* (2018).

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4700–4708.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

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

[18] Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.

[19] Joel Lehman and Kenneth O Stanley. 2011. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 211–218.

[20] Joel Lehman, Kenneth O Stanley, et al. 2008. Exploiting open-endedness to solve problems through the search for novelty.. In *ALIFE*. Citeseer, 329–336.

[21] Liam Li and Ameet Talwalkar. 2020. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*. PMLR, 367–377.

[22] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Fei-Fei Li, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 19–34.

[23] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. 2018. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJQRKzbA-

[24] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1eYHoC5FX

[25] Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=Skq89Scxx

[26] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 419–427.

[27] Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. 2020. Multi-objective evolutionary design of deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation* (2020).

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[29] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 4095–4104. http://proceedings.mlr.press/v80/pham18a.html

[30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4780–4789.

[31] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2902–2911.

[32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[33] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[34] Nilotpal Sinha and Kuan-Wen Chen. 2021. Evolving neural architecture using one shot model. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 910–918.

[35] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2017. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the genetic and evolutionary computation conference*. 497–504.

[36] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G Yen, and Mengjie Zhang. 2019. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Transactions on Evolutionary Computation* 24, 2 (2019), 350–364.

[37] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. 2019. Completely automated CNN architecture design based on blocks. *IEEE transactions on neural networks and learning systems* 31, 4 (2019), 1242–1254.

[38] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Jiancheng Lv. 2020. Automatically designing CNN architectures using the genetic algorithm for image classification. *IEEE transactions on cybernetics* 50, 9 (2020), 3840–3854.

[39] Lingxi Xie and Alan Yuille. 2017. Genetic cnn. In *Proceedings of the IEEE International Conference on Computer Vision*. 1379–1388.

[40] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2019. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rylqooRqK7

[41] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. 2020. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1gDNyrKDS

[42] Miao Zhang, Huiqi Li, Shirui Pan, Taoping Liu, and Steven W Su. 2020. One-Shot Neural Architecture Search via Novelty Driven Sampling.. In *IJCAI*. 3188–3194.

[43] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6848–6856.

[44] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

[45] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8697–8710.