Appendices

A1. Related Works

A1.1. Neural Architecture Search

In early NAS, evolutionary algorithm (EA) [36,37] or reinforcement learning (RL) [4, 49, 50] were commonly-used search algorithms. Although these early works were successful in proving the potential of NAS, their immediate application was challenging due to the enormous search cost, easily mounting up to tens of thousands of GPU hours to search for a single architecture. Most of the computational overhead in early NAS algorithms occurred as the result of having to train each candidate architecture to convergence and evaluate it.

Through weight sharing [34], recent NAS works were able to achieve a noticeable acceleration in the architecture search process. Weight sharing utilizes a super-network, whose sub-networks correspond to candidate architectures belonging in the pre-defined search space. The sub-networks are evaluated with the weights inherited from the super-network, and thus, the sub-networks end up sharing a common set of weights. Modern NAS algorithms that exploit such performance approximation techniques can be categorized into differentiable NAS [9–11, 13, 14, 24, 28, 43, 45, 48], and one-shot NAS [5, 7, 17, 33, 46]; while the search and evaluation processes are entangled in the former, the latter disentangles them into separate processes.

Performance predictors [15], which take an encoded neural architecture as an input and output the accuracy of the corresponding architecture, are another promising direction for reducing the architecture evaluation cost. The main challenge in performance prediction is minimizing the number of architecture-accuracy pairs required to obtain a performance predictor that generalizes well to the rest of the search space. Because the problem of architecture performance prediction is by definition a regression task, [15, 25, 40] aimed to predict the exact value of accuracy by minimizing the mean squared error loss between the predicted and true accuracy. Recently, the concept of architecture comparators [12, 42], is rising in popularity. Instead of estimating the exact accuracy, comparators take two architectures as an input and use a ranking loss or a contrastive learning framework to predict which is more likely to rank higher in terms of accuracy.

Apart from weight sharing and performance predictors, there also exist works that aim to search for more general proxy settings for architecture evaluation. EcoNAS [47] explores four common reduction factors - the number of channels, the resolution of input images, the number of training epochs, and the sample ratio of the full training set - and determine which one of these proxies can be used to reliably estimate the final test accuracy. Na *et al.* [32] show that it is possible to only use a subset of the target dataset for execute NAS and propose a novel proxy dataset selection algorithm.

A1.2. NAS at initialization

Evaluating neural architectures without any amount of training is surely an interesting and attractive research direction that has potential to NAS. Mellor *et al*. [30] use the feature separability in the linear regions of a neural architecture as a metric to score architectures. Abdelfattah *et al.* [1] attempt to identify which one of the pruning-at-initialization techniques is most useful for NAS. Zen-NAS [26] analyzes the activation patterns in a neural architecture to quantify its expressivity. KNAS [41] and TE-NAS [8] have previously proposed to use the NTK framework to score neural architectures and thus are most closely-related to this work. As mentioned in the main paper, KNAS and TE-NAS use the MEAN and the CN metrics, respectively. In addition to CN, TE-NAS utilizes another at-initialization score, derived from the number of linear regions in a neural architecture.

A1.3. Neural Tangent Kernel

The NTK framework is based on the observation that for certain initialization schemes, the infinite width limit of many neural architectures can be exactly characterized using kernel tools [22]. Provided that this assumption holds, many of the questions in deep learning theory can be addressed through the study of linear methods and convex analyses [38]. The intuitiveness of the NTK framework led to important results regarding the generalization and optimization of deep neural networks [2, 6, 18, 23, 27, 51]. Such advances in the NTK framework subsequently led researchers to study how the NTK can be leveraged in various applications: prediction of the generalization performance and the training speed, explanation of inductive biases in deep neural networks, design of new classifiers, Despite these limitations, the intuitiveness of the NTK, which allows to use a powerful set of theoretical tools to exploit it, has led to a rapid increase in the amount of research that successfully leverages the NTK in applications, such as predicting generalization [16] and training speed [44], explaining certain inductive biases [31, 39] or designing new classifiers [3, 29]. Despite the proliferation of the NTK framework in the deep learning theory, there still exist doubts on whether the assumptions in the NTK framework truly holds for deep neural networks that are used in real life [19, 20].

A2. Metrics Summary

In Table A1, we provide an overview of the NTK-based metrics studied in the main paper, along with the direction of their rank correlation with the final test accuracy of a neural architecture.

Table A1. Overview of the NTK-based metrics studied in the main paper. "Rank" refers to whether the metric should be positively (+) or negatively (−) correlated with the final test accuracy.

Metric	Equation	Rank
F-Norm	$\|\|\Theta_{\theta_t}\|\|_F$	+
Mean	$\mu(\Theta_{\theta_0})$	+
NCN	$\frac{-(\lambda_{\max}(\Theta_{\theta_0}))}{(\lambda_{\min}(\Theta_{\theta_0}))}$	+
LGA	$\frac{(\Theta_{\theta_0}-\mu(\Theta_{\theta_0}))\cdot(L_{\mathcal{Y}}-\mu(L_{\mathcal{Y}}))}{\|\|\Theta_{\theta_0}-\mu(\Theta_{\theta_0})\|\|_2\|\|L_{\mathcal{Y}}-\mu(L_{\mathcal{Y}})\|\|_2}$	+

Table A2. Summary of differences among different NDS search spaces. "# Ops." and "# nodes" correspond to the number of opearations and the number of nodes in each cell. "Output" refers to which node(s) are concatenated for the output (= A if all nodes are are concatenate, = L if there are nodes that are not used as input to other nodes). "# cells" refers to the number possible cells, without considering the redundancy, that exist in each search space.

Benchmarks	# Ops.	# Nodes	Output	# Cells (B)
NASNet	13	5	L	71,465,842
Amoeba	8	5	L	556,628
PNAS	8	5	A	556,628
ENAS	5	5	L	5,063
DARTS	8	4	A	242

A3. NAS Benchmarks & Image Datasets

NAS-Bench-101 (cont'd from the main paper) Each convolution operator in NAS-Bench-101 follows the Conv-BN-ReLU pattern, and naïve convolutions are used instead of separable convolutions, such that the resulting architectures closely match the designs of ResNet and Inception. Each cell is stacked 3 times, followed by a max-pooling layer, which halves the image height and width and doubles the number of channels. In the resulting DNN, the above pattern is repeated 3 times, and lastly, a glabal average pooling and a final classification layer with the Softmax function are inserted.

NAS-Bench-201 (cont'd from the main paper) The convolution operator in NAS-Bench-201 follows the operation sequence of ReLU-Conv-BN. The macro-architecture of NAS-Bench-201 starts with one 3-by-3 convolution with 16 output channels and a batch normalization layer [21]. Each cell is stacked 5 times, followed by a residual block. In the final DNN, this pattern is repeated 3 times. The number of output channels in the first, second, and third stages is set to be 16, 32 and 64, respectively. The residual block serves to downsample the spatial size and double the channels of an input feature map. The shortcut path in this residual block consists of a 2-by-2 average pooling layer with stride of 2 and a 1-by-1 convolution. Lastly, a global average pooling and a final classification layer with the Softmax function are inserted for classification.

NDS Benchmark [35] Please refer to Table A2 for the summary of differences among the search spaces in the NDS benchmark [35]

Image Datasets Please refer to Table A4 for the summary of image datasets utilized in NAS benchmarks.

A4. Experimental Details

Section 3.2 The NTK computation involves per-sample gradients, which are computationally intractable to obtain from high-dimensional datasets that contain tens of thousands images. Therefore, in this paper, we instead use a single minibatch, randomly sampled from the train set, to compute the NTK. For CIFAR-10 and CIFAR-100 datasets, a minibatch of size 256 is used, while for ImageNet16-120, a minibatch of size 512 is used. For a fair comparison, we use the same set of image samples to construct the minibatch used for evaluation across all benchmarks. A single NVIDIA V100 GPU is used for the experiments in this section.

Sections 3.3. & 3.4 For rank correlation evaluation on CIFAR-10 and CIFAR-100, we use a minibatch of size 256, while for that on ImageNet16-120, we use a minibatch of size 512. For NCN, we use the Eval mode BN on PyTorch, and for F-Norm and Mean, we use the Train mode BN. The choice of BN usage is set based on the results from Section 3.2; for all metrics, we choose the BN setting that yields the highest rank correlation for each metric. A single NVIDIA V100 GPU is used for the experiments in these sections.

Section 4.1 Measurements are done on NAS-Bench-201. We use a single image sampled from the validation set and Eval mode BN for NTK computation in this section. A single NVIDIA V100 GPU is used for the experiments.

Section 4.3 To train candidate architectures, we use the momentum SGD optimizer with a learning rate of 0.025, momentum of 0.9, a weight decay factor of 3e-4. These are standard settings used to train architectures in NAS benchmarks. The train set of each dataset is used for training, and a single minibatch from the validation set is used to derive NTK-based metrics. A batch size of 1024 is used to train architectures. For NTK computation, a batch size of 256 is used for CIFAR-10 and CIFAR-100, and a batch size of 512 is used for ImageNet16-120. A single NVIDIA V100 GPU is used for the experiments in this section. For F-Norm and Mean, we use the Train mode BN, and for NCN and LGA, we use the Eval mode BN.

Section 5 We use the same experimental settings as those used in **Section 4.3** to train architectures and derive LGA_3 and LGA_5. A single NVIDIA A40 GPU is used to execute both random and evolutionary search algorithms.

Table A3. CIFAR-10 Search results on additional search spaces.

Metric	NB101	NDS-DARTS	NDS-ENAS
F-Norm	89.17	91.83	85.19
Mean	88.05	88.40	91.14
NCN	90.81	91.99	92.52
LGA	**92.57**	**93.61**	**93.26**
Metric	NDS-Amoeba	NDS-NASNet	NAS-Macro
F-Norm	91.35	84.71	87.12
Mean	90.89	89.89	87.25
NCN	89.14	87.21	90.62
LGA	**94.35**	**94.30**	**92.24**

A5. Full Benchmark Evaluation Results

In Figure A1, the rank correlation evaluation results of existing at-initialization NTK-based metrics on all benchmarks are visualized.

A6. Fine-grained Rank Correlation Evaluation

This experiment is repeated for 20 different runs, and the evaluation results in box-and-whisker plots are presented in Figure A2, A3, and A4. P1 contains 100 architectures sampled from Top-10% of architectures, whereas P10 contains the same number of architectures sampled from Bottom-10% of architectures. Ticks on the x-axis correspond to accuracy deciles in descending order (P1 $\rightarrow$ P10). "Total" refers to the rank correlation evaluation results over all 1,000 architectures.

A7. Search Results on Other Benchmarks

We conducted random search with 500 sampled architectures with four compared metrics. Except for LGA, which is obtained after 3 epochs, others are measured at initialization. The search results are presented in Table A3. LGA is the only metric that searches for a successful architecture across all search spaces.

A8. Pseudocode for Search Algorithms

The pseudocode for random search is presented in Algorithm 1. In random search, N corresponds to the total number of candidate architectures evaluated during search. In this work, we set N in random search to be 100. The pseudocode for evolutionary search is presented in Algorithm 2. In evolutionary search, N corresponds to the number of architectures kept in the parent pool (or population). In this work, we set N in evolutionary search to be 10.

A9. Limitations & Societal Impact

Limitations: Even though we demonstrate that LGA exhibits a high predictive performance with only few epochs of training, extensive theoretical advances are required to fundamentally address the unreliability of the NTK. To become a truly reliable theoretical framework for architecture selection, the NTK must be able to encompass diverse operations, normalization types, and weight initializations.
Societal Impact: With the help of at-initialization metrics, NAS was able to greatly reduce its environmental cost. Unfortunately, the need for training to stabilize the NTK will inevitably increase the environmental cost of NAS.

Algorithm 1: Random Search

```
sampler = RandomSampler()
best_arch, best_LGA = None, 0
for i = 1 : N do
  cand_arch = sampler()
  for Epoch = 1 : t do
    Train(cand_arch)
  end for
  LGA_t = cand_arch.LGA()
  if LGA_t > best_LGA then
    best_arch, best_LGA = cand_arch, LGA_t
  end if
end for
chosen_arch = best_arch
```

Algorithm 2: Evolutionary Search

```
parent_pool = []
lga_hist = []
sampler = RandomSampler()
parent_arch, child_arch = None, None
for i = 1 : N do
  cand_arch = sampler()
  parent_pool.append(cand_arch)
  for Epoch = 1 : t do
    Train(cand_arch)
  end for
  lga_hist.append(cand_arch.LGA())
end for
while search budget not exceeded do
  Based on lga_hist, choose the architecture with the
  highest LGA_t from the parent pool as parent_arch
  child_arch = Mutate(parent_arch)
  for Epoch = 1 : t do
    Train(child_arch)
  end for
  parent_pool.popleft()
  lga_hist.popleft()
  parent_pool.append(child_arch)
  lga_hist.append(child_arch.LGA())
```

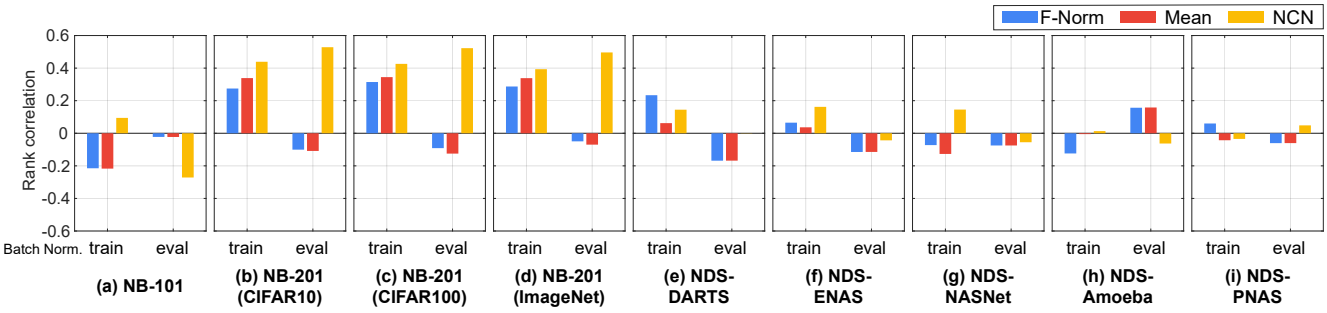

Figure A1. Rank correlation evaluation results on various NAS benchmarks. For F-Norm and Mean, the evaluation results based on the Train mode BN are reported, whereas for NCN and LGA, those based on the Eval mode BN are reported. The scale and the range of y-axes are set to be the same across all search spaces. NB is an abbreviation for NAS-Bench.

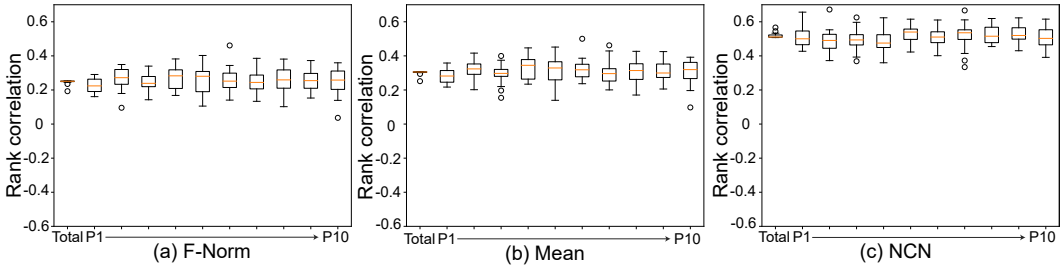

Figure A2. Rank correlation evaluation results on NAS-Bench-201 CIFAR-10 for every accuracy decile.

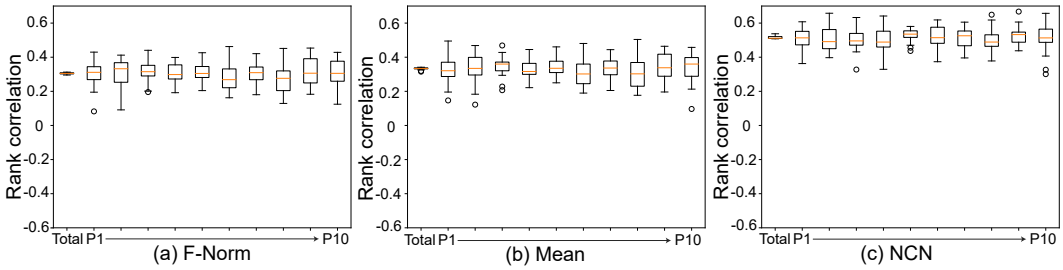

Figure A3. Rank correlation evaluation results on NAS-Bench-201 CIFAR-100 for every accuracy decile.

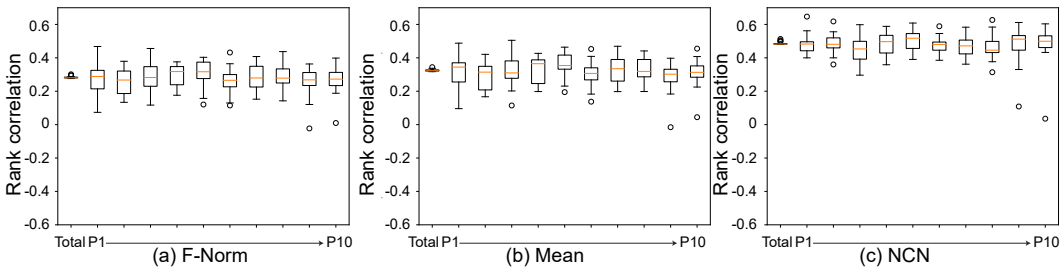

Figure A4. Rank correlation evaluation results on NAS-Bench-201 ImageNet-16-120 for every accuracy decile.

Table A4. Summary of image datasets used to construct NAS Benchmarks.

Dataset	# of Train Data	# of Validation Data	# of Test Data	# of Classes	Image Size
CIFAR-10	50,000	-	10,000	10	(32×32)
CIFAR-100	50,000	-	10,000	100	(32×32)
ImageNet-16-120	151,700	3,000	3,000	120	(16×16)

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight nas. In *International Conference on Learning Representations*, 2020. 1

[2] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019. 1

[3] Sanjeev Arora, Simon S Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2019. 1

[4] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, 2017. 1

[5] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *Proceedings of the 35th International Conference on Machine Learning*, pages 550–559, 2018. 1

[6] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *Advances in Neural Information Processing Systems*, 32:12893–12904, 2019. 1

[7] Andrew Brock, Theo Lim, JM Ritchie, and Nick Weston. Smash: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2018. 1

[8] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *International Conference on Learning Representations*, 2020. 1

[9] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. 2020. 1

[10] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Drnas: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2020. 1

[11] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019. 1

[12] Yaofo Chen, Yong Guo, Qi Chen, Minli Li, Wei Zeng, Yaowei Wang, and Mingkui Tan. Contrastive neural architecture search with neural architecture comparators. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9502–9511, 2021. 1

[13] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: Robustly stepping out of performance collapse without indicators. In *International Conference on Learning Representations*, 2020. 1

[14] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. In *European Conference On Computer Vision*, 2020. 1

[15] Boyang Deng, Junjie Yan, and Dahua Lin. Peephole: Predicting network performance before training. *arXiv preprint arXiv:1712.03351*, 2017. 1

[16] Aditya Deshpande, Alessandro Achille, Avinash Ravichandran, Hao Li, Luca Zancato, Charless Fowlkes, Rahul Bhotika, Stefano Soatto, and Pietro Perona. A linearized framework and a new benchmark for model selection for fine-tuning. *arXiv preprint arXiv:2102.00084*, 2021. 1

[17] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019. 1

[18] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019. 1

[19] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[20] Micah Goldblum, Jonas Geiping, Avi Schwarzschild, Michael Moeller, and Tom Goldstein. Truth or backpropaganda? an empirical investigation of deep learning theory. In *International Conference on Learning Representations*, 2019. 1

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 2

[22] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8580–8589, 2018. 1

[23] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32:8572–8583, 2019. 1

[24] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1620–1630, 2020. 1

[25] Zhihang Li, Teng Xi, Jiankang Deng, Gang Zhang, Shengzhao Wen, and Ran He. Gp-nas: Gaussian process based neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11933–11942, 2020. 1

[26] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 347–356, 2021. 1

[27] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 1

[29] Wesley Maddox, Shuai Tang, Pablo Moreno, Andrew Gordon Wilson, and Andreas Damianou. Fast adaptation with linearized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 2737–2745. PMLR, 2021. 1

[30] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021. 1

[31] Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. Self-distillation amplifies regularization in hilbert space. volume 33, 2020. 1

[32] Byunggook Na, Jisoo Mok, Hyeokjun Choe, and Sungroh Yoon. Accelerating neural architecture search via proxy data. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021. 1

[33] Houwen Peng, Hao Du, Hongyuan Yu, QI LI, Jing Liao, and Jianlong Fu. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[34] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4095–4104, 2018. 1

[35] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1882–1890, 2019. 2

[36] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019. 1

[37] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2902–2911. JMLR. org, 2017. 1

[38] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 1

[39] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. volume 33, 2020. 1

[40] Yehui Tang, Yunhe Wang, Yixing Xu, Hanting Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. A semi-supervised assessor of neural architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1810–1819, 2020. 1

[41] Jingjing Xu, Liang Zhao, Junyang Lin, Rundong Gao, Xu Sun, and Hongxia Yang. Knas: Green neural architecture search. In *International Conference on Machine Learning*, pages 11613–11625. PMLR, 2021. 1

[42] Yixing Xu, Yunhe Wang, Kai Han, Yehui Tang, Shangling Jui, Chunjing Xu, and Chang Xu. Renas: Relativistic evaluation of neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4411–4420, 2021. 1

[43] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2019. 1

[44] Luca Zancato, Alessandro Achille, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Predicting training time without training. volume 33, 2020. 1

[45] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2019. 1

[46] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven Su. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7809–7818, 2020. 1

[47] Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11396–11404, 2020. 1

[48] Pan Zhou, Caiming Xiong, Richard Socher, and Steven Hoi. Theory-inspired path-regularized differential network architecture search. In *Neural Information Processing Systems*, 2020. 1

[49] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. 1

[50] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018. 1

[51] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. *Advances in neural information processing systems*, 2019. 1