# Zero-Cost Operation Scoring in Differentiable Architecture Search

**Lichuan Xiang[1*], Łukasz Dudziak[2*], Mohamed S. Abdelfattah[3*]**
**Thomas Chau[2], Nicholas D. Lane[2,4], Hongkai Wen[1,2]**
[1] University of Warwick, UK
[2] Samsung AI Center Cambridge, UK
[3] Cornell University, USA
[4] University of Cambridge, UK
{l.xiang.2, hongkai.wen}@warwick.ac.uk
{l.dudziak, thomas.chau, nic.lane}@samsung.com
mohamed@cornell.edu

## Abstract

We formalize and analyze a fundamental component of differentiable neural architecture search (NAS): local "operation scoring" at each operation choice. We view existing operation scoring functions as inexact proxies for accuracy, and we find that they perform poorly when analyzed empirically on NAS benchmarks. From this perspective, we introduce a novel *perturbation-based zero-cost operation scoring* (Zero-Cost-PT) approach, which utilizes zero-cost proxies that were recently studied in multi-trial NAS but degrade significantly on larger search spaces, typical for differentiable NAS. We conduct a thorough empirical evaluation on a number of NAS benchmarks and large search spaces, from NAS-Bench-201, NAS-Bench-1Shot1, NAS-Bench-Macro, to DARTS-like and MobileNet-like spaces, showing significant improvements in both search time and accuracy. On the ImageNet classification task on the DARTS search space, our approach improved accuracy compared to the best current training-free methods (TE-NAS) while being over $10\times$ faster (total searching time 25 minutes on a single GPU), and observed significantly better transferability on architectures searched on the CIFAR-10 dataset with an accuracy increase of 1.8 pp. Our code is available at: https://github.com/zerocostptnas/zerocost_operation_score.

## Introduction

One of the biggest problems in neural architecture search (NAS) is the computational cost – even training a single deep network can require enormous computational resources, and many NAS methods need to train tens, if not hundreds, of networks in order to converge to a good architecture (Real et al. 2019; Luo et al. 2018; Dudziak et al. 2020). A related problem concerns search space size—a larger NAS search space would typically contain better architectures but requires a longer searching time (Real et al. 2019). Differentiable architecture search (DARTS) was first proposed to tackle those challenges, showcasing promising results when searching for a network in a set of over $10^{18}$ possible variations (Liu, Simonyan, and Yang 2019). Unfortunately, DARTS has significant robustness issues, as demonstrated through many recent works (Zela et al. 2020; Shu, Wang, and Cai 2020; Yu et al. 2020). It also requires a careful selection of hyperparameters, making it somewhat difficult to adapt to a new task. Recently, (Wang et al. 2021) showed that operation selection in DARTS based on the magnitude of architectural parameters ($\alpha$) is fundamentally wrong and will always simply select skip connections over other more meaningful operations. They proposed an alternative operation selection method based on *perturbation*, where the importance of an operation is determined by the decrease of the supernet's validation accuracy when it is removed. Then the most important operations are selected by exhaustively comparing them with other alternatives on each single edge of the supernet until the final architecture is found.

In a parallel line of work that aims to speed up NAS, *proxies* are often used instead of training accuracy to quickly obtain an indication of performance without expensive full training for each searched model. Conventional proxies typically consist of a reduced form of training with fewer epochs, less data or a smaller DNN architecture (Zhou et al. 2020). Most recently, *zero-cost proxies*, which are extreme types of NAS proxies that do not require any training, have gained interest with empirically outperforming conventional training-based proxies and deliver outstanding results on common NAS benchmarks (Abdelfattah et al. 2021; Mellor et al. 2021). However, their efficient usage on a large search space, typical for differentiable NAS, has been shown to be more challenging and remains an open problem (Mellor et al. 2021).

The objective of our paper is to shed some light onto the implicit proxies that are used for operation scoring in differentiable NAS, and to discover new proxies in this setting that have the potential of improving both search speed and quality. We decompose differentiable NAS into its two constituent parts: (1) supernet training and (2) operation scoring. We focus on the second component and formalize the concept of "operation scoring" that happens during local operation selection at each edge in a supernet. Through this lens, we are able to empirically compare the efficacy of existing differentiable NAS operation scoring functions. We find that existing methods act as a proxy for accuracy and perform poorly on NAS benchmarks. Consequently, we propose new operation scoring functions based on zero-cost proxies that outperform

*These authors contributed equally.

existing methods on both search speed and accuracy. Our main contributions are:

- Formalize *operation scoring* in differentiable NAS and perform a first-of-its-kind analysis of the implicit proxies that are present in existing methods.
- Propose, evaluate and compare *perturbation-based zero-cost operation scoring* (Zero-Cost-PT) for differentiable NAS building upon recent work on training-free NAS proxies.
- Perform a thorough empirical evaluation of Zero-Cost-PT in multiple search spaces and datasets, including DARTS, DARTS subspaces S1-S4, MobileNet-like space, and 3 popular NAS benchmarks: NAS-Bench-201, NAS-Bench-1shot1 and NAS-Bench-Macro.

## Related work

**Classic NAS and Proxies.** Zoph & Lee were among the first to propose an automated method to search neural network architectures, using a reinforcement learning agent to maximize rewards coming from training different models (Zoph and Le 2017). Since then, a number of alternative approaches have been proposed in order to reduce the significant cost introduced by training each proposed model. In general, reduced training can be found in many NAS works (Pham et al. 2018; Zhou et al. 2020), and different proxies have been proposed, e.g. searching for a model on a smaller dataset and then transferring the architecture to the larger target dataset (Real et al. 2019; Mehrotra et al. 2021), or incorporating a predictor into the search process (Wei et al. 2020; Dudziak et al. 2020; Wu et al. 2021; Wen et al. 2019).

**Zero-cost Proxies.** Recently, zero-cost proxies (Mellor et al. 2021; Abdelfattah et al. 2021) for NAS emerged from pruning-at-initialisation literature (Tanaka et al. 2020; Wang, Zhang, and Grosse 2020; Lee, Ajanthan, and Torr 2019; Turner et al. 2020). Such proxies can be formulated as architecture scoring functions $S(A)$ that evaluate the "saliency" of a given architecture $A$ in achieving accuracy at initialization without the expensive training process. In this paper, we adopt the recently proposed zero-cost proxies (Abdelfattah et al. 2021; Mellor et al. 2021), namely `grad_norm`, `snip`, `grasp`, `synflow`, `fisher` and `nwot`. Those metrics either aggregate the saliency of model parameters to compute the score of an architecture (Abdelfattah et al. 2021), or use the overlapping of activations between different samples within a minibatch of data as a performance indicator (Mellor et al. 2021). In a similar vein, (Chen, Gong, and Wang 2021) proposed the use of training-free scoring for operations based on the neural tangent kernel (Jacot, Gabriel, and Hongler 2021) and number of linear regions; operations with the lowest score are *pruned* from the supernet iteratively until a subnetwork is found.

**Differentiable NAS and Operation Perturbation.** Liu et al. first proposed to search for a neural network's architecture by parameterizing it with continuous values (called architectural parameters $\alpha$) in a differentiable way. Their method constructs a *supernet*, i.e., a superposition of all networks in the search space, and optimizes the architectural parameters ($\alpha$) together with supernet weights ($w$). The final architecture is extracted from the supernet by preserving operations with the largest $\alpha$. Despite the significant reduction in searching time, the stability and generalizability of DARTS have been challenged, e.g., it may produce trivial models dominated by skip connections (Zela et al. 2020). SDARTS (Chen and Hsieh 2020) proposed to overcome such issues by smoothing the loss landscape, while SGAS (Li et al. 2020) considered a greedy algorithm to select and prune operations sequentially. The recent DARTS-PT (Wang et al. 2021) proposed a perturbation-based operation selection strategy, showing promising results on DARTS space. In DARTS-PT operations are no longer selected by optimizing architectural parameters ($\alpha$), but via a scoring function evaluating the impact on a supernet's validation accuracy when they are removed.

## Rethinking Operation Scoring

In the context of differentiable NAS, a supernet would contain multiple candidate operations on each edge as shown in Figure 1. Operation scoring functions assign a score to rank operations and select the best one. In this section, we empirically quantify the effectiveness of existing operation scoring methods in differentiable NAS, with a specific focus on DARTS (Liu, Simonyan, and Yang 2019) and the recently-proposed DARTS-PT (Wang et al. 2021). Concretely, we view these scoring functions as proxies for final subnetwork accuracies and we evaluate them on that basis to quantify how well these functions perform. We challenge many assumptions made in previous work and show that we can outperform existing methods with lightweight alternatives.

### Operation Scoring Preliminaries

For a supernet $A$ we want to be able to start *discretizing* edges in order to derive a subnetwork. When discretizing we replace an edge composed of multiple candidate operations and their respective (optional) architectural parameters $\alpha$ with only one operation selected from the candidates. We will denote the process of discretization of an edge $e$ with operation $o$, given a model $A$, as: $A + (e, o)$. Analogously, the *perturbation* of a supernet $A$ by removing an operation $o$ from an edge $e$ will be denoted as $A - (e, o)$. Figure 1 illustrates discretization and perturbation. Furthermore, we will use $\mathcal{A}$, $\mathcal{E}$ and $\mathcal{O}$ to refer to the set of all possible network architectures, edges in the supernet and candidate operations, respectively. More details about notation can be found in Appendix A.1.

NAS can then be performed by iterative discretization of edges in the supernet, yielding in the process a sequence of partially discretized architectures: $A_0, A_1, ..., A_{|\mathcal{E}|}$, where $A_0$ is the original supernet, $A_{|\mathcal{E}|}$ is the final fully-discretized subnetwork (result of NAS), and $A_t$ is $A_{t-1}$ after discretizing a next edge, i.e., $A_t = A_{t-1} + (e_t, o_t)$ where $t$ is an iteration counter. The problem of finding the sequence of $(e_t, o_t)$ that maximizes the performance of the resulting network $A_{|\mathcal{E}|}$ has an optimal substructure and can be reduced to the problem of finding the optimal policy $\pi : \mathcal{A} \times \mathcal{E} \to \mathcal{O}$ that is used to decide on an operation to assign to an edge at each iteration, given current model (state). This policy function is defined by means of an analogous scoring function $f : \mathcal{A} \times \mathcal{E} \times \mathcal{O} \to \mathbb{R}$, that assigns scores to the possible values of the policy function, and then taking $\arg\max$ or $\arg\min$ over $f$,

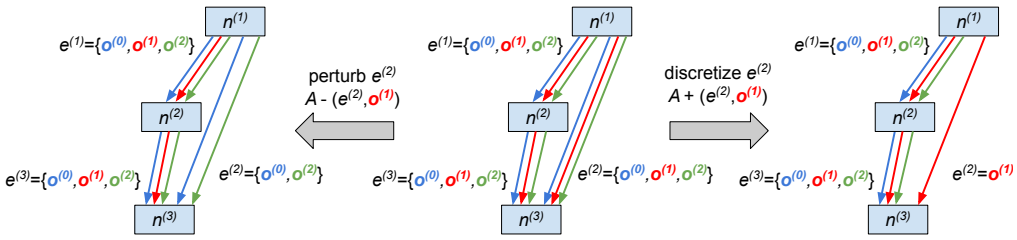


Figure 1: Visualization of perturbation and discretization of an edge in a supernet. Middle: a supernet is composed of three edges $\{e^{(i)}\}_{i=1,2,3}$, each consisting of three possible operations $\{o^{(i)}\}_{i=1,2,3}$ which are applied in parallel to the same input. Left: edge $e^{(2)}$ is perturbed by removing $o^{(1)}$ from the set of candidate operations assigned to this edge. Right: the same edge $e^{(2)}$ is discretized with operation $o^{(1)}$ by removing all other candidate operations leaving $o^{(1)}$ as the only choice left.

depending on the type of scores produced by $f$. [1]

We begin by defining the optimal scoring function that we will later use to assess the quality of different empirical approaches. For a given partially-discretized model $A_t$, let us denote the set of all possible fully-discretized networks that can be obtained from $A_t$ after a next edge $e$ is discretized with an operation $o$ as $\mathcal{A}_{t,e,o}$. Our optimal scoring function can then be defined as:

$$\pi_{\text{best-acc}}(A_t, e) = \arg\max_{o \in \mathcal{O}_e} \max_{A_{|\mathcal{E}|} \in \mathcal{A}_{t,e,o}} V^*(A_{|\mathcal{E}|}) \quad (1)$$

where $V^*$ is the validation accuracy of a network after converged (we will use $V$ to denote validation accuracy without training), and $\mathcal{O}_e \subseteq \mathcal{O}$ is the subset of candidate operations that are considered for edge $e$. It is easy to see that this policy meets Bellman's principle of optimality (Bellman 1957) – the definition follows directly from it and therefore is the optimal solution to our problem. However, it might be more practical to consider the expected achievable accuracy when an operation is selected, instead of the best. Therefore we define the function $\pi_{\text{avg-acc}}$:

$$\pi_{\text{avg-acc}}(A_t, e) = \arg\max_{o \in \mathcal{O}_e} \mathop{\mathbb{E}}_{A_{|\mathcal{E}|} \in \mathcal{A}_{t,e,o}} V^*(A_{|\mathcal{E}|}) \quad (2)$$

In practice, we are unable to use either $\pi_{\text{best-acc}}$ or $\pi_{\text{avg-acc}}$ since we would need to have the final validation accuracy $V^*$ of all the networks in the search space. Here we consider the following practical alternatives from DARTS (Liu, Simonyan, and Yang 2019) and the recent DARTS-PT (Wang et al. 2021):

$$\pi_{\text{darts}}(A_t, e) = \arg\max_{o \in \mathcal{O}_e} \alpha_{e,o} \quad (3)$$

$$\pi_{\text{disc-acc}}(A_t, e) = \arg\max_{o \in \mathcal{O}_e} V^*(A_t + (e, o)) \quad (4)$$

$$\pi_{\text{darts-pt}}(A_t, e) = \arg\min_{o \in \mathcal{O}_e} V(A_t - (e, o)) \quad (5)$$

where $\alpha_{e,o}$ is the architectural parameter assigned to operation $o$ on edge $e$ as presented in DARTS (Liu, Simonyan, and Yang 2019). $\pi_{\text{disc-acc}}$ uses accuracy of a supernet after an operation $o$ is assigned to an edge $e$ – this is referred to as "discretization accuracy" in DARTS-PT and is assumed to be a good operation scoring function (Wang et al. 2021), it could approximate $f_{\text{avg-acc}}$. $\pi_{\text{darts-pt}}$ is the perturbation-based approach used by DARTS-PT – it is presented as a practical and lightweight alternative to $\pi_{\text{disc-acc}}$ (Wang et al. 2021).

**Zero-Cost Operation Scoring.** We argue that the scoring functions 3-5 are merely proxies for the best achievable accuracy (Eq. 1). As such, we see an opportunity to use a new class of training-free proxies that are very fast to compute and have been shown to work well within multi-trial NAS, albeit not in differentiable NAS, nor within large search spaces. We present the following scoring functions that use a zero-cost proxy $S$ instead of validation accuracy when discretizing an edge or perturbing an operation. Note that the supernet is randomly-initialized and untrained.

$$\pi_{\text{disc-zc}}(A_t, e) = \arg\max_{o \in \mathcal{O}_e} S(A_t + (e, o)) \quad (6)$$

$$\pi_{\text{zc-pt}}(A_t, e) = \arg\min_{o \in \mathcal{O}_e} S(A_t - (e, o)) \quad (7)$$

In the rest of this paper, we consider the following proxies that have been proposed in recent zero-cost NAS literature: `grad_norm` (Abdelfattah et al. 2021), `snip` (Lee, Ajanthan, and Torr 2019), `grasp` (Wang, Zhang, and Grosse 2020), `synflow` (Tanaka et al. 2020), `fisher` (Theis et al. 2018), `zen_score` (Lin et al. 2021), `tenas` (Chen, Gong, and Wang 2021) and `nwot` (Mellor et al. 2021). Detailed metrics descriptions are included in Appendix A.2. Note that in most existing work (Abdelfattah et al. 2021; Mellor et al. 2021), zero-cost metrics are not used to score operations but to select architectures based on their end-to-end scores, their effectiveness on operation selection remains to discover, and we are going to show that building operation scoring function and algorithm upon on them is trivial, while TE-NAS (Chen, Gong, and Wang 2021) also uses them to score operations. However, as opposed to *selecting* the optimal operations (via either *discretization* or *perturbation*), the `tenas` metric is used to iteratively *prune* the weakest operations from a supernet (Chen, Gong, and Wang 2021).

[1] Since a scoring function clearly defines a relevant policy function, we will sometimes talk about a scoring function even though the context might be directly related to a policy function – in those cases, it should be understood as the policy function that follows from the relevant scoring function (and vice versa).

## Empirical Analysis on Operation Scoring

In this subsection, we investigate the performance of different operation scoring methods. Because we want to compare with the optimal best-acc and avg-acc, we conduct experiments on two popular NAS benchmarks: NAS-Bench-201 (Dong and Yang 2020) and NAS-Bench-1Shot1 (Zela, Siems, and Hutter 2020). In the following, we discuss our findings using NAS-Bench-201, while results on NAS-Bench-1Shot1 can be found in Appendix A.10. We conduct our investigation in two settings, *initial* and *progressive*. The first setting compares operation scoring functions while making their first decision (iteration 0) during NAS. The second, *progressive*, setting takes into account retraining of a partially discretized supernet $A_t$ and a subsequent rescoring of operations, that might occur between iterations of different algorithms that we consider like darts-pt (Wang et al. 2021).

**Initial Operation Scoring.** For the supernet $A_0$ we compute the operation scores for all operations on all edges, at the first iteration (iteration 0) of NAS, that is, $f(A_0, e, o)\ \forall\ e \in \mathcal{E},\ o \in \mathcal{O}_e$. In our first experiment, we collect the scores produced by different scoring methods, per operation, per edge, then compute the Spearman rank correlation for operations on each edge, and finally average the rank correlation coefficient over all edges (details of our experiments and illustrative examples are provided in Appendix A.3). The resulting averaged rank correlation is indicative of how well an operation scoring method would do when making the first discretization decision, relative to a perfect "oracle" search. We plot the rank correlation coefficients in Figure 2, showing many surprising findings. First, the original darts $\alpha$ score is weakly and inversely correlated with the oracle scores, further supporting arguments in prior work that this is not an effective operation scoring method. Second, disc-acc is inversely correlated to best-acc. This refutes the claim in the DARTS-PT paper that disc-acc is a reasonable operation score (Wang et al. 2021) – these findings are aligned with prior work that has already shown that the supernet accuracy is unrelated to the final subnetwork accuracy (Li et al. 2020). Third, the darts-pt score does not track disc-acc, in fact, it is inversely-correlated to it as well, meaning that the darts-pt score is not a good approximation of disc-acc. However, darts-pt is weakly correlated to the "oracle" best-acc and avg-acc scores which supports (empirically) why it works well. Fourth, tenas (Chen, Gong, and Wang 2021), which also utilizes training-free operation scoring, performs fairly well, with Spearman-$\rho$ =0.44, but still falls short of the performance of the two zc-pt variants ($\rho$ =0.77 and 0.63). Finally, our zc-pt, when using either `synflow` or `nwot` metric, is strongly correlated with both the best-acc and avg-acc metrics, indicating that there could be huge promise when using this scoring function within NAS. Note that disc-zc, in particular when using `nwot` metric, is only weakly correlated with the oracle scores, suggesting that *perturbation* is a more robust scoring paradigm than *discretization*. We provide more analysis on disc-zc *vs.* later, and compare NAS results when using either scoring method in Appendix A.6.

In Table 1, we show the discovered NAS-Bench-201 architecture when applying the seven scoring functions (Eq. (1) – (7)) for operation selection on all edges. As expected, best-acc chooses the best subnetwork, while avg-acc selects a very good model but not the best one, likely due to the large variance of accuracies in NAS-Bench-201. zc-pt(`nwot`) selected one of the top models in NAS-Bench-201 as expected from the strong correlation with the oracle best-acc function; while tenas selected a good model, in the top 15% of the NAS-Bench-201 dataset, commensurate with the average correlation shown in Figure 2. The remaining operation scoring functions failed to produce a good model in this experiment, suggesting that these metrics do not make a good initial choice of operations at iteration 0 of differentiable NAS. A similar analysis of NAS-Bench-1shot1 search space can be found in Appendix A.10. We provide a more detailed look into the failures of certain scoring methods below.

| | **best-acc** | **avg-acc** | **disc-acc** | **darts-pt** |
|---|---|---|---|---|
| Avg. Error[1][%] | 5.63 | 6.24 | 13.55 | 19.43 |
| Rank[2] | 1 | 166 | 12,744 | 13,770 |
| | **zc-pt(`nwot`)** | **disc-zc(`nwot`)** | **darts** | **tenas** |
| Avg. Error[1][%] | 5.81 | 22.96 | 45.7 | 7.19 |
| Rank[2] | 14 | 14,274 | 15,231 | 1,817 |

[1] Computed as the average of all available seeds for the selected model in NAS-Bench-201 CIFAR-10 dataset.
[2] Rank in NAS-Bench-201

Table 1: Model selected based on maximizing each operation strength independently.

**Analysis of the darts-pt and disc-acc scoring.** As mentioned before, our zc-pt operation scoring function outperforms both darts-pt and disc-acc, despite the latter methods relying directly on accuracy. This may sound counter-intuitive, but it becomes clearer if we note that the accuracy used by these methods (supernet accuracy) is not directly relevant to the NAS objective (subnet accuracy). Regarding darts-pt, we argue that the unrolled estimation performed by a supernet, as described by (Wang et al. 2021), might lead to the observed preference towards selecting skip connections (see Table A12). This is because under this hypothesis, convolutional operations on an edge perform only refinement of the input, while most of the information is carried directly from the input through a skip connection. Therefore, it can be expected that removal of the skip connection should have much severe effects on the supernet's performance. More information can be found in Appendix A.3.

Regarding disc-acc, we perform an additional case study of how the supernet's accuracy changes after discretizing an edge with different operations, as a function of training epochs. The experiment details are in Appendix A.3. From the results we can see that as retraining progresses, lighter operations (`none` in particular) converge much faster than heavy, but potentially more meaningful, operations like `conv_1x1`. What is more, even after sufficiently long training, all choices converge to roughly the same point. We argue that this is caused by the fact that in the early NAS iterations the supernet is heavily overparamtrized, which hinders our ability to faithfully measure each operation's contribution based simply on the accuracy of the supernet.

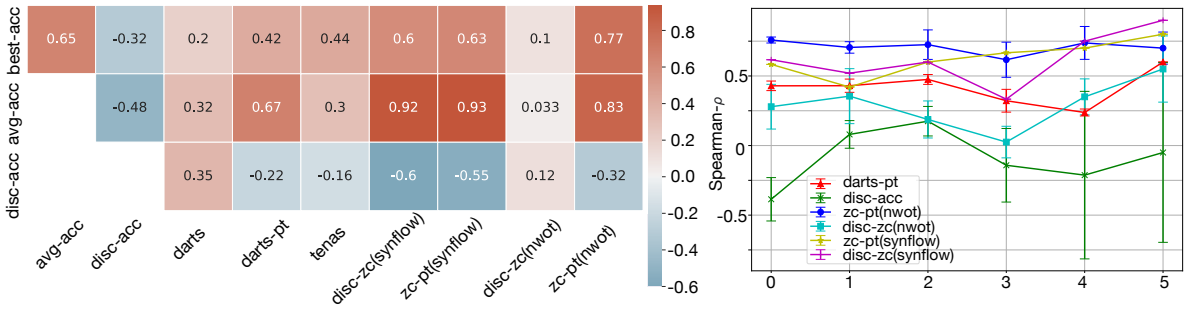


Figure 2: Left: Rank correlation coefficient between different operation scoring metrics at the first iteration of NAS. Right: Rank correlation coefficient of different operation scoring functions vs. best-acc when invoked iteratively for each edge. In iteration $i$, only edge $i$ is discretized then all scores for all operations on the remaining edges is computed and correlated against best-acc.

**Analysis of the zc-pt and disc-zc scoring.** We further perform detailed experiments on comparing both *pertubation* and *discretization* policies, especially when using `nwot` zero-cost metric. Specifically, we create a toy model that allows us to observe how `disc-zc(nwot)` and `zc-pt(nwot)` behave in a simplified setting, as we vary the model's depth. We observe that while both approaches behave similarly when a network is shallow, as we increase its depth `disc-zc` quickly degrades and becomes biased towards selecting skip connection, which is not the case for `zc-pt`. This suggests that `nwot` might in fact prefer shallower networks – however, unlike discretization, perturbation paradigm does not introduce the ability to reduce the supernet's depth (as long as each edge includes at least two meaningful operations among their candidates), which seems to robustify `nwot` significantly. More detailed can be found in Appendix A.4.

While the above signals some major weaknesses of different proxies used in differentiable NAS, when used to perform initial scoring of operations, it's worthwhile to further analyze them in the *progressive* setting which would show what happens in later NAS iterations.

**Progressive Operation Scoring.** Until now, we have only investigated the performance of operation scoring functions in the first iteration of NAS. This approach is relevant for methods like DARTS, where operation scoring function $f$ does not depend on $A_t$ in any way (only $A_0$), but is not truly representative of other methods that work iteratively. Because of that, we extend our analysis to investigate what happens in later iterations of NAS. To do that, we calculate the correlation of scoring functions in the *progressive* setting by performing the following steps: (1) score operations on all undiscretized edges, (2) discretize edge $i$, (3) retrain for 5 epochs (darts-pt and disc-acc only), (4) increment $i$ and repeat from step 1 until all edges are discretized. At each iteration $i$, we calculate the scores for the operations on all remaining undiscretized edges and compute their Spearman-$\rho$ rank correlation coefficients with respect to best-acc. This is plotted in Figure 2, averaged over 4 seeds.

Our results confirm many of our *initial* (iteration-0) analysis. zc-pt(`nwot`) continues to be the best operation scoring function, and darts-pt is the second-best, improving in correlation from 0.4 to 0.6 between the first and last iterations, indeed showing that retraining and/or progressive discretization helps. However, disc-acc continues to be unrepresentative of operation strength even when used in the iterative setting. This is not what we expected, especially in the very last iteration when disc-acc is supposed to match a subnetwork exactly. As Figure 2 shows, the variance in the last iteration is quite large – we believe this happens because we do not train to convergence every time we discretize an edge, and instead we only train for 5 epochs. Our progressive analysis provided further empirical evidence that supernet discretization accuracy should not be used as a proxy for subnetwork accuracy, contradicting (Wang et al. 2021). However, we have confirmed that darts-pt does in fact improve when retraining is performed between NAS iterations, but could still be improved upon with zc-pt – it performed exceptionally well as a proxy for accuracy and has the potential to make differentiable NAS both much faster and of higher accuracy.

## Zero-Cost-PT Neural Architecture Search

Based on our analysis of operation scoring, in this section, we propose a NAS algorithm called Zero-Cost-PT using zero-cost perturbation and perform ablation studies to find the best set of heuristics for our NAS methodology, including: edge discretization order, number of search and validation iterations, and the choice of the zero-cost metric.

### Architecture Search with Zero-cost Proxies

Our algorithm contains two stages: *architecture proposal* and *validation*. It begins with an untrained supernet $A_0$ which contains a set of edges $\mathcal{E}$, the number of proposal iterations `N`, and the number of validation iterations `V`. In each proposal iteration $i$, we discretize the supernet $A_0$ based on our proposed zero-cost-based perturbation function $f_{\text{zc-pt}}$ that achieved promising results in the previous section. After all edges have been discretized, the final architecture is added to the set of candidates and we begin the process again for $i+1$ starting with the original $A_0$. After `N` candidate architectures have been constructed, the validation stage begins. We score the candidate architectures again using a selected zero-cost metric (the same which is used in $f_{\text{zc-pt}}$), but this time computing their end-to-end score rather than using the perturbation paradigm. We calculate the zero-cost metric for

| Method | CIFAR-10 | CIFAR-100 | ImageNet-16 |
|---|---|---|---|
| Zero-Cost-PT$_{\texttt{random}}$ with different proxies (Section ) | | | |
| tenas | $70.07_{\pm 39.87}$ | $83.04_{\pm 31.93}$ | $90.57_{\pm 17.21}$ |
| fisher | $10.64_{\pm 1.27}$ | $38.48_{\pm 1.96}$ | $82.85_{\pm 12.63}$ |
| grad_norm | $10.55_{\pm 1.11}$ | $38.43_{\pm 2.10}$ | $80.71_{\pm 12.10}$ |
| grasp | $9.81_{\pm 3.42}$ | $36.52_{\pm 6.33}$ | $64.27_{\pm 8.82}$ |
| snip | $8.32_{\pm 2.02}$ | $34.00_{\pm 4.03}$ | $65.35_{\pm 11.04}$ |
| zen_score[1] | $6.24_{\pm 0.00}$ | $28.89_{\pm 0.00}$ | $58.56_{\pm 0.00}$ |
| synflow[1] | $6.24_{\pm 0.00}$ | $28.89_{\pm 0.00}$ | $58.56_{\pm 0.00}$ |
| nwot | **5.97**$_{\pm 0.17}$ | **27.47**$_{\pm 0.28}$ | **53.82**$_{\pm 0.77}$ |
| **Baselines and SOTA approaches** (Section ) | | | |
| Random | $13.39_{\pm 13.28}$ | $39.17_{\pm 12.58}$ | $66.87_{\pm 9.66}$ |
| DARTS | $45.70_{\pm 0.00}$ | $84.39_{\pm 0.00}$ | $83.68_{\pm 0.00}$ |
| DARTS-PT [1] | $11.89_{\pm 0.00}$ | $45.72_{\pm 6.26}$ | $69.60_{\pm 4.40}$ |
| DARTS-PT (fix $\alpha$) [1, 2] | $6.20_{\pm 0.00}$ | $34.03_{\pm 2.24}$ | $61.36_{\pm 1.91}$ |
| NASWOT(synflow) [3] | $6.54_{\pm 0.62}$ | $29.53_{\pm 2.13}$ | $58.22_{\pm 4.18}$ |
| NASWOT(nwot) [3] | $7.04_{\pm 0.80}$ | $29.97_{\pm 1.16}$ | $55.57_{\pm 2.07}$ |
| TE-NAS | *6.10*$_{\pm 0.47}$ | $28.76_{\pm 0.56}$ | $57.62_{\pm 0.46}$ |
| Zero-Cost-Disc(nwot) | $6.22_{\pm 0.84}$ | *28.18*$_{\pm 2.01}$ | *55.14*$_{\pm 1.77}$ |

[1] Only 1 model was selected across all 4 seeds in both cases.
[2] Results on CIFAR-10 taken from (Wang et al. 2021). Results on other datasets computed using official code in (Wang et al. 2021) across 4 seeds.
[3] Using N=1000 for both proxies and averaged over 500 runs as in (Mellor et al. 2021).

Table 2: Comparison in test error (%) with SOTA perturbation-based and zero-cost NAS on NAS-Bench-201 (best in bold, 2nd best in italic, applies throughout).

each subnetwork using V different minibatches of data. The final architecture is selected by the best total score during the validation stage. The full algorithm is outlined as Algorithm 1 in Appendix A.5 and the flowchart of our algorithms is in Figure 3 . Our algorithm contains four main hyperparameters: N, V, ordering of edges to follow when discretizing, and the zero-cost metric to use ($S$). In the following we present detailed ablations to decide on the best possible configuration.

## Ablation Study

We conduct ablations of the proposed Zero-Cost-PT approach on NAS-Bench-201 (Dong and Yang 2020). More results on additional NAS search spaces and benchmarks are reported later in Section  and Appendix A.7. NAS-Bench-201 constructed a unified cell-based search space, where each architecture has been trained on three different datasets, CIFAR-10, CIFAR-100 and ImageNet-16-120[2]. In our experiments, we take a randomly initialised supernet for this search space and apply our Zero-Cost-PT algorithm to search for architectures without any training. We search with four random seeds (0, 1, 2, 3) and report the average and standard deviation of test errors of the obtained architectures. All searches are performed on CIFAR-10, and obtained architectures are then additionally evaluated on the other two datasets.

**Different Zero-cost Metrics.** Since our focus is to understand how existing zero-cost metrics can be successfully applied to a large-space NAS, we begin our investigation by analysis how different metrics behave when used in the proposed combination with perturbation-based search. It is also worth noting that our formulation and analysis are general and can be extended to new zero-cost proxies that may emerge in the future. For now we only consider random edge discretization order (Zero-Cost-PT$_{\texttt{random}}$), and more details on edge discretization order will be presented later. Table 2 compares the average test errors of architectures selected by different proxies on NAS-Bench-201. We see that nwot, synflow and zen_score perform considerably better across the three datasets than the others, where nwot offers around 0.27% improvement over synflow. On the other hand, we notice tenas fails in this case, as the metric was designed for *pruning* operations rather than *selecting* them (more details are discussed in Appendix A.6). Other than that, even the naive grad_norm outperforms the state-of-the-art DARTS-PT on this benchmark. This confirms it is the appropriate combination of zero-cost metrics and perturbation-based NAS paradigms as in Zero-Cost-PT that could become promising proxies to the actual trained accuracy. We also observed that the ranking of those metrics are quite stable on the three datasets (descending order in terms of error as in Table 2), indicating that architectures discovered by our Zero-Cost-PT have good transferability. In particular nwot consistently performs best, reducing test errors on all datasets by a considerable margin.

**Architecture Proposal vs. Validation.** We then study the impact of different architecture proposal iterations N and validation iterations V when Zero-Cost-PT uses nwot metric and random edge discretization order. Intuitively, larger N leads to more architecture candidates being found, while V indicates the amount of data used to rank the search candidates. As shown in Figure 4, we see larger N does lead to more architectures discovered, but not proportional to the value of N on NAS-Bench-201 space. For N=100 we discover 27.8 distinct architectures on average, but when increased to N=1000 the number only roughly doubles. We also see that even with N=10, Zero-Cost-PT$_{\texttt{random}}$ can already discover top models in the space, demonstrating desirable balance between search quality and efficiency. On the other hand, as shown in Figure 4, larger V tends to reduce the performance variance, especially for smaller N. This is also expected as more validation iterations could stabilise the ranking of selected architecture candidates, helping Zero-Cost-PT to retain the most promising ones with a manageable overhead of V mini-batches. To further justify our finding on NAS-Bench-201, we performed similar ablations on DARTS-CNN space. We study the impact of different architecture proposal iterations N and validation iterations V when Zero-Cost-PT uses random as the search order and nwot metric. The further experiment details are in Appendix A.7

We first consider an extreme case, setting architecture proposal iteration N=1, where Zero-Cost-PT only proposes one candidate (with random edge discretization order), and with no validation stage performed. And then, in order to maximize the performance of our method, we balance exploration (higher N + random edge order) and exploitation (higher V) in the searching and validation phases respectively.

Admittedly, the interplay between those two phases is crucial for our method. To further showcase how the valida-

[2] We use the three random seeds available in NAS-Bench-201: 777, 888, 999.

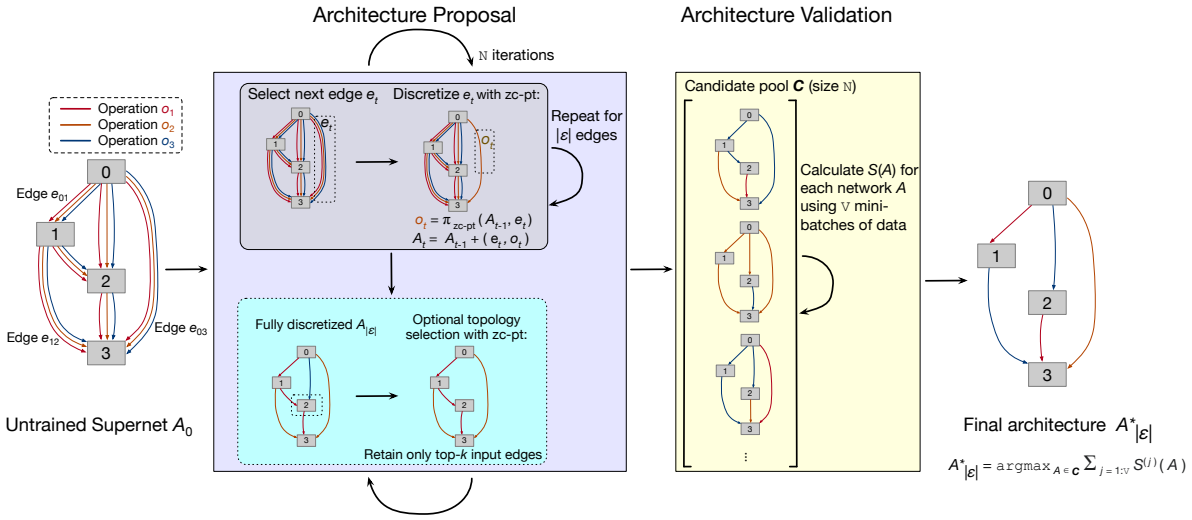


Figure 3: Flowchart of the proposed Zero-Cost-PT algorithm.

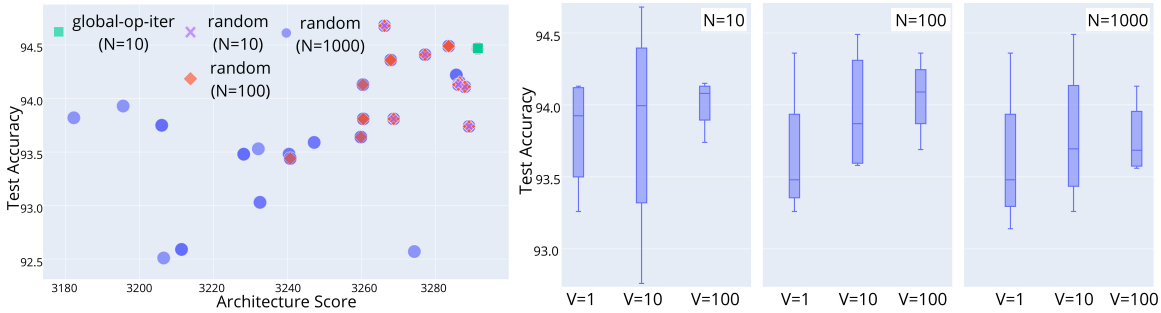


Figure 4: (left) Accuracy vs. score of architectures discovered on CIFAR-10 by Zero-Cost-PT with different N. (right) Accuracy distribution of discovered architectures with different N and V.

| N | V | Test Error(%) | |
|---|---|---|---|
| | | Avg. | Best |
| 1 | 0 | $2.81_{\pm 0.29}$ | 2.43 |
| 10 | 1 | $2.93_{\pm 0.14}$ | 2.65 |
| | 10 | $2.88_{\pm 0.14}$ | 2.65 |
| | 100 | $2.64_{\pm 0.16}$ | 2.43 |

Table 3: Ablation overview, the performance of Zero-Cost-PT$_{\texttt{random}}$ with N={1, 10}, V={0, 1, 10, 100}, nwot metric on DARTS CNN space.

tion phase complements the searching phase, we run additional ablations on the DARTS CNN space with N=10 and V={1,10,100}, the results are shown in Table 3. The results are consistent with what is shown in the NAS-Bench-201.

**Edge Discretization Order.** Finally, we investigate how different edge discretization orders impact our approach, when the best performing nwot metric, N=10 and V=100 are used. We consider the following edge discretization orders:

- `fixed`: discretizes the edges in a fixed order, where our experiments discretize from the input towards the output;
- `random`: discretizes the edges in a random order;
- `GOI(global-op-iter)`: iteratively evaluates $S(A-(e,o))$ for all operations on all edges in $\mathcal{E}$, selects the edge $e$ containing the operation $o^*$ with globally best score. Discretizes $e$ with $o^*$, then repeats to decide on the next edge (re-evaluating scores) until all edges have been discretized;
- `GEI(global-edge-iter)`: similar to the above but iteratively selects edge $e$ from $\mathcal{E}$ based on the average score of all operations on each edge;
- `GOO(global-op-once)`: only evaluates $S(A-(e,o))$ for all operations once to obtain a ranking order of the operations and decide the edge order upfront based on it, then starts following the algorithm as usual, calculating scores of operations at each edge iteratively;
- `GEO(global-edge-once)`: similar to the above but uses the average score of operations on edges to obtain the edge discretization order.

Table. 5 shows the performance of and # of perturbations required by our Zero-Cost-PT approach. We observe that `global-op-iter` consistently performs best across all three datasets since it iteratively explores the search space of remaining operations while greedily selecting the current best. On the other hand, we see that the perfor-

| Method | Error [%] Avg. | Best | Params [M] | Cost [GPU-days] |
|---|---|---|---|---|
| DARTS | $3.00_{\pm 0.14}$ | - | 3.3 | 0.4 |
| SDARTS-RS | $2.67_{\pm 0.03}$ | - | 3.4 | 0.4 |
| SGAS | $2.66_{\pm 0.24}$ | - | 3.7 | 0.25 |
| DARTS-PT | $\underline{\mathbf{2.61}_{\pm 0.08}}$ | *2.48* | 3.0 | 0.8 |
| DARTS-PT$_{+\texttt{none}}$[1] | $2.73_{\pm 0.13}$ | 2.67 | 3.2 | 0.8 |
| TE-NAS | $2.63_{\pm 0.064}$ | - | 3.8 | *0.05* |
| max-param-random | $2.94_{\pm 0.098}$ | 2.83 | 5.14 | - |
| NASWOT(2500) | $2.99_{\pm 0.22}$ | 2.66 | - | 0.018 |
| NASWOT(20000) | $2.73_{\pm 0.09}$ | 2.58 | - | 0.083 |
| NASWOT(50000) | $2.72_{\pm 0.09}$ | 2.52 | - | 0.208 |
| Zero-Cost-EVO | $2.94_{\pm 0.14}$ | 2.72 | - | 0.018 |
| Zero-Cost-PT$_{\texttt{synflow}}$ | $3.88_{\pm 0.56}$ | 3.38 | 5.1 | - |
| Zero-Cost-PT$_{\texttt{zen_score}}$ | $3.06_{\pm 0.31}$ | 2.68 | 2.9 | - |
| Zero-Cost-PT$_{\texttt{random}}$ | $2.64_{\pm 0.16}$ | **2.43** | 4.7 | **0.018** |
| Zero-Cost-PT$_{\texttt{global-op-iter}}$ | *2.62*$_{\pm 0.09}$ | 2.49 | 4.6 | 0.17 |

[1] Results obtained by re-enabling `none` operation in DARTS-PT (Wang et al. 2021).

Table 4: Comparison with s SOTA differentiable NAS methods on the DARTS CNN search space (CIFAR-10).

| Order[1] | # of Perturbations[2] | C10 | C100 | ImageNet-16 |
|---|---|---|---|---|
| fixed | $\lvert\mathcal{O}\rvert\lvert\mathcal{E}\rvert$ | $5.98_{\pm 0.50}$ | $27.60_{\pm 1.63}$ | $54.23_{\pm 0.93}$ |
| GOI | $\frac{1}{2}\lvert\mathcal{O}\rvert\lvert\mathcal{E}\rvert(\lvert\mathcal{E}\rvert+1)$ | $\mathbf{5.69}_{\pm 0.19}$ | $\mathbf{26.80}_{\pm 0.51}$ | $\mathbf{53.64}_{\pm 0.40}$ |
| GOO | $2\lvert\mathcal{O}\rvert\lvert\mathcal{E}\rvert - \lvert\mathcal{O}\rvert$ | $6.30_{\pm 0.57}$ | $28.96_{\pm 1.66}$ | $55.04_{\pm 1.47}$ |
| GEI | $\frac{1}{2}\lvert\mathcal{O}\rvert\lvert\mathcal{E}\rvert(\lvert\mathcal{E}\rvert+1)$ | $6.23_{\pm 0.45}$ | $28.42_{\pm 0.59}$ | $54.39_{\pm 0.47}$ |
| GEO | $2\lvert\mathcal{O}\rvert\lvert\mathcal{E}\rvert - \lvert\mathcal{O}\rvert$ | $6.30_{\pm 0.57}$ | $28.96_{\pm 1.66}$ | $55.04_{\pm 1.47}$ |
| random | $\lvert\mathcal{O}\rvert\lvert\mathcal{E}\rvert$ | *5.97*$_{\pm 0.17}$ | *27.47*$_{\pm 0.28}$ | *53.82*$_{\pm 0.77}$ |

[1] All methods use `nwot` metric, with `N`=10 and `V`=100.
[2] Number of perturbations per search iteration.

Table 5: Test error (%) of Zero-Cost-PT when using different search orders on NAS-Bench-201.

mance of `global-op-once` is inferior since it determines the order of perturbation by assessing the importance of operations once and for all at the beginning, which may not be appropriate as discretization continues. We observe similar behaviour in `global-edge-iter` and `global-edge-once`, both of which use the average importance of operations on edges to decide search order, leading to suboptimal performance. It is also worth pointing out that `fixed` performs relatively well comparing to the other variants, offering comparable performance with `random`. This shows that Zero-Cost-PT is generally robust to the edge discretization order. In the following experiments, we use Zero-Cost-PT with `random` order with a moderate setting in architecture proposal iterations (`N`=10) to balance exploration and exploitation during search, while maintaining efficiency.

## Results

In this section, we perform extensive empirical comparisons of Zero-Cost-PT with the state-of-the-art differentiable and zero-cost NAS algorithms on a number of search spaces.

| Method | Error [%] Top-1 | Top-5 | Params [M] | Cost [GPU-days] |
|---|---|---|---|---|
| DARTS | 26.7 | 8.7 | 4.7 | 0.4 |
| SDARTS-RS | 25.6 | 8.2 | - | 0.4 |
| DARTS-PT | 25.5 | 8.0 | 4.6 | 0.8 |
| PC-DARTS | 25.1 | 7.8 | 5.3 | 0.1 |
| SGAS | **24.1** | 7.3 | 5.4 | 0.25 |
| TE-NAS(C10) | 26.2 | 8.3 | 6.3 | *0.05* |
| TE-NAS | 24.5 | 7.5 | 5.4 | 0.17 |
| Zero-Cost-PT[1](best) | *24.4* | 7.5 | 6.3 | **0.018** |
| Zero-Cost-PT[1](4 seeds) | $24.6_{\pm 0.13}$ | $7.6_{\pm 0.09}$ | 6.3 | **0.018** |

[1] We use the same training pipeline DARTS (Liu, Simonyan, and Yang 2019).

Table 6: Comparison with SOTA differentiable NAS methods on the DARTS CNN search space (ImageNet).

Due to space limit, in the following, we present results on NAS-Bench-201 (Dong and Yang 2020), DARTS CNN space (Liu, Simonyan, and Yang 2019) and the practical large search space MobileNet-like space. Results on NAS-Bench-1shot1 (Zela, Siems, and Hutter 2020), NAS-Bench-Macro (Su et al. 2021) and the four DARTS subspaces S1-S4 (Zela et al. 2020), together with detailed experimental settings and more baselines are in Appendix A.9-A.12.

### Tabular NAS Benchmarks

Table 2 shows the average test error (%) of the competing approaches and our Zero-Cost-PT on the three datasets in NAS-Bench-201. Here we include the naive random search and original DARTS as baselines, and compare our approach with the recent zero-cost NAS algorithm NASWOT (Mellor et al. 2021), TE-NAS (Chen, Gong, and Wang 2021), as well as the perturbation-based NAS approaches DARTS-PT and DARTS-PT (fix $\alpha$) (Wang et al. 2021). As in all competing approaches, we perform a search on CIFAR-10 and evaluate the final model on all three datasets. We see that on all datasets, our Zero-Cost-PT (with `nwot`) consistently offers superior performance, especially on CIFAR-100 and ImageNet-16. On the other hand, the best existing perturbation-based algorithm, DARTS-PT (fix $\alpha$), fails on those two datasets, producing suboptimal results with small improvements compared to random search, suggesting that architectures discovered by DARTS-PT might not transfer well to other datasets. TE-NAS is second best on CIFAR but as we show in the later section, performance deteriorates on larger datasets like ImageNet.

We compare the performance of Zero-Cost-DISC and our proposed Zero-Cost-PT on NAS-Bench-201 (Dong and Yang 2020), as shown in Table 2. Zero-Cost-DISC results in inferior performance compared to the proposed perturbation-based approach (Zero-Cost-PT) on all datasets, confirming our previous analysis on their oracle metric correlations.

### DARTS CNN Search Space

We use the same settings as in DARTS-PT (Wang et al. 2021), but instead of pre-training the supernet and fine-tuning it after each perturbation, we take an untrained supernet and directly perform our algorithm as in previous section. Additional

details, baselines, ablations and discovered architectures can be found in Appendix A.7, A.9, and A.13.

**Results on CIFAR-10.** As shown in Table 4 the proposed Zero-Cost-PT approaches can achieve a much better average test error than the DARTS and are comparable to its newer variants SDARTS-RS (Chen and Hsieh 2020) and SGAS (Li et al. 2020) at a much lower searching cost (especially when using `random` ordering). There is a significant search cost reduction compared to DARTS-PT. While DARTS-PT needs to perform retraining between iterations, Zero-Cost-PT only evaluates the score of the perturbed supernet with zero-cost proxies ($S_{\texttt{nwot}}$), requiring less than a minibatch of data.

**Different Zero-cost Searching Approaches.** We additionally compare our Zero-Cost-PT to several alternative ways of performing zero-cost NAS, to further show its efficiency in utilising zero-cost proxies and assert its efficiency as a searching methodology. We start with the simplest baseline of maximizing number of parameters, which is based on the observation that our method tends to select slightly larger models than some of the baselines in Table 4 and Table 6. We include details in Appendix A.7 and summarize our findings here. Overall, the test error (%) of this baseline is $2.93_{\pm 0.23}$ (avg.) and 2.78 (min), vs. our $2.64_{\pm 0.16}$ (avg.) and 2.43 (min). This confirms that simply selecting models with maximum FLOPs/Params is not an appropriate searching method in general and that our methods perform more meaningful architecture selection than simply maximising model size.

In Section Tabular NAS Benchmarks, we compared our method to sampling-based zero-cost NAS in Table 2. Our results are empirically better on all three datasets. Additionally, our method computes the operation score per edge in a supernet, whereas the sampling-based approach computes the end-to-end network score. The relationship between the number of subnetworks and the number of operations is exponential. Therefore, we anticipate having to sample exponentially many networks in sample-based NASWOT (Mellor et al. 2021) compared to our proposed Zero-Cost-PT.

In order to extend the comparison between zero-cost NAS (NASWOT) and our Zero-Cost-PT to the DARTS CNN search space, we have conducted further experiments similar to NASWOT on NAS-Bench-201, the details of these experiments can be found in Appendix A.7 and the results are presented in Table 4. In sum, for a similar time budget to ours (25 min), the average performance of the baseline is actually closer to the random search ($3.29_{\pm 0.15}$) (Liu, Simonyan, and Yang 2019) than to our method, with significant variance.

In addition to the random sampling baseline presented above, here we further extend our study by performing an evolution-based search for a model maximizing the `nwot` metric (instead of sampling randomly as above) which is then trained. We denote this baseline as Zero-Cost-EVO. We allow a similar search budget (2500 sample size, $\sim$25min on a single 2080ti GPU), and follow the same settings as in our experiments (searching with 4 random seeds, each of the discovered models is trained with 4 random seeds). The results are shown in Table 4. We can see that when given a similar search budget, evolution-based search performs significantly worse than Zero-Cost-PT (avg. of 2.94 vs. 2.64), confirming the efficacy of the proposed NAS algorithm.

| Architecture | Error [%] | | Params | Cost |
|---|---|---|---|---|
| | Top-1. | Top-5 | [M] | [GPU-days] |
| MobileNet-V3(1.0) | *24.8* | - | 5.3 | 288 |
| GreedyNAS | 25.1 | - | 3.8 | *7.6* |
| SPOS | 25.3 | - | - | 12.4 |
| ProxylessNAS (GPU) | 24.9 | 7.5 | 7.1 | 8.3 |
| Zero-Cost-PT(best) | **23.6** | 6.8 | 8.0 | **0.041** |
| Zero-Cost-PT(avg) | $23.8_{\pm 0.08}$ | $6.93_{\pm 0.09}$ | 8.1 | 0.041 |

Table 7: Comparison on MobileNet search space (ImageNet).

**Results on ImageNet.** Table 6 shows the ImageNet classification accuracy for architectures searched on CIFAR-10. Our Zero-Cost-PT$_{\texttt{random}}$ algorithm is able to find architectures with a comparable accuracy much faster than previous work, further reinforcing its efficacy in this setting. While TE-NAS results on CIFAR-10 were very close to Zero-Cost-PT, a much larger difference is observed on ImageNet with an accuracy drop of 1.8 pp and a search time that is $\sim 2.5\times$ slower than Zero-Cost-PT.

## MobileNet-like Search Space

It is well known that most of the existing NAS algorithms designed for MobileNet-like perform constrained NAS. However, our method has not been designed for such a context. To the best of our knowledge, the necessity to consider both scores of operations and their potential contribution to the sum of #FLOPS/Params of the final model would result in a potentially NP-hard problem. Therefore, we do not enforce such constraints at this point as it is less relevant.

**Results on ImageNet.** Table 7 shows the performance (error %) of the architectures discovered by the proposed Zero-Cost-PT algorithm on ImageNet, using a MobileNet-like search space from ProxylessNAS (Cai, Zhu, and Han 2019), we, therefore, compare only to results using the same setting. We see that compared to the existing train-based approaches, our approach allows for finding even better models, but also larger ones, much faster (at least $190\times$ speed up). Please note, that because our method performs unconstrained NAS, unlike existing baselines, the results should not be interpreted as being objectively better. Instead, we simply use them as reference points to put our results in perspective – the goal was to show that our method works well in this type of search space and the results support this claim; as expected, more accurate models, but with a larger footprint, can be found faster compared to the constrained baselines.

# Conclusion

In this paper, we formalized the implicit operation scoring proxies that are present within differentiable NAS algorithms to both analyze existing methods and propose new ones. We showed that lightweight operation scoring methods based on zero-cost proxies empirically outperform existing operation scoring functions. We also found that perturbation is more effective than discretization when scoring an operation, leading to our lightweight NAS algorithm, Zero-Cost-PT.

## References


Abdelfattah, M. S.; Mehrotra, A.; Dudziak, Ł.; and Lane, N. D. 2021. Zero-Cost Proxies for Lightweight NAS. In *International Conference on Learning Representations (ICLR)*.

Bellman, R. 1957. *Dynamic Programming*. Dover Publications. ISBN 9780486428093.

Cai, H.; Zhu, L.; and Han, S. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations (ICLR)*.

Chen, W.; Gong, X.; and Wang, Z. 2021. Neural Architecture Search on ImageNet in Four GPU Hours: A Theoretically Inspired Perspective. In *International Conference on Learning Representations*.

Chen, X.; and Hsieh, C.-J. 2020. Stabilizing differentiable architecture search via perturbation-based regularization. In *International Conference on Machine Learning (ICML)*, 1554–1565. PMLR.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dong, X.; and Yang, Y. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Dudziak, Ł.; Chau, T.; Abdelfattah, M. S.; Lee, R.; Kim, H.; and Lane, N. D. 2020. BRP-NAS: Prediction-based NAS using GCNs. In *Neural Information Processing Systems (NeurIPS)*.

Jacot, A.; Gabriel, F.; and Hongler, C. 2021. Neural Tangent Kernel: Convergence and Generalization in Neural Networks (Invited Paper). In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, 6. New York, NY, USA: Association for Computing Machinery. ISBN 9781450380539.

Lee, N.; Ajanthan, T.; and Torr, P. H. 2019. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*.

Li, G.; Qian, G.; Delgadillo, I. C.; Muller, M.; Thabet, A.; and Ghanem, B. 2020. SGAS: Sequential greedy architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1620–1630.

Lin, M.; Wang, P.; Sun, Z.; Chen, H.; Sun, X.; Qian, Q.; Li, H.; and Jin, R. 2021. Zen-NAS: A Zero-Shot NAS for High-Performance Deep Image Recognition. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Luo, R.; Tian, F.; Qin, T.; Chen, E.; and Liu, T.-Y. 2018. Neural Architecture Optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 7827–7838. Red Hook, NY, USA: Curran Associates Inc.

Mehrotra, A.; Ramos, A. G.; Bhattacharya, S.; Łukasz Dudziak; Vipperla, R.; Chau, T.; Abdelfattah, M. S.; Ishtiaq, S.; and Lane, N. D. 2021. NAS-Bench-ASR: Reproducible Neural Architecture Search for Speech Recognition. In *International Conference on Learning Representations (ICLR)*.

Mellor, J.; Turner, J.; Storkey, A.; and Crowley, E. J. 2021. Neural Architecture Search without Training. In *International Conference on Machine Learning (ICML)*.

Pearlmutter, B. A. 1993. Fast Exact Multiplication by the Hessian. *Neural Computation*.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient Neural Architecture Search via Parameters Sharing. In *International Conference on Machine Learning (ICML)*, 4095–4104.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized Evolution for Image Classifier Architecture Search. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Shu, Y.; Wang, W.; and Cai, S. 2020. Understanding Architectures Learnt by Cell-based Neural Architecture Search. In *International Conference on Learning Representations*.

Su, X.; Huang, T.; Li, Y.; You, S.; Wang, F.; Qian, C.; Zhang, C.; and Xu, C. 2021. Prioritized Architecture Sampling with Monto-Carlo Tree Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10968–10977.

Tanaka, H.; Kunin, D.; Yamins, D. L. K.; and Ganguli, S. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Neural Information Processing Systems (NeurIPS)*.

Theis, L.; Korshunova, I.; Tejani, A.; and Huszár, F. 2018. Faster gaze prediction with dense networks and Fisher pruning. ArXiv:1801.05787.

Turner, J.; Crowley, E. J.; O'Boyle, M.; Storkey, A.; and Gray, G. 2020. BlockSwap: Fisher-guided Block Substitution for Network Compression on a Budget. In *International Conference on Learning Representations (ICLR)*.

Wang, C.; Zhang, G.; and Grosse, R. 2020. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *International Conference on Learning Representations (ICLR)*.

Wang, R.; Cheng, M.; Chen, X.; Tang, X.; and Hsieh, C.-J. 2021. Rethinking Architecture Selection in Differentiable NAS. In *International Conference on Learning Representations (ICLR)*.

Wei, C.; Niu, C.; Tang, Y.; and min Liang, J. 2020. NPENAS: Neural Predictor Guided Evolution for Neural Architecture Search. *arXiv:2003.12857*.

Wen, W.; Liu, H.; Li, H.; Chen, Y.; Bender, G.; and Kindermans, P.-J. 2019. Neural Predictor for Neural Architecture Search. *arXiv:1912.00848*.

Wu, J.; Dai, X.; Chen, D.; Chen, Y.; Liu, M.; Yu, Y.; Wang, Z.; Liu, Z.; Chen, M.; and Yuan, L. 2021. Weak NAS Predictors Are All You Need. *arXiv:2102.10490*.

Yu, K.; Sciuto, C.; Jaggi, M.; and Claudiu Musat, M. S. 2020. Evaluating The Search Phase of Neural Architecture Search. In *International Conference on Learning Representations*.

Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; and Hutter, F. 2020. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, volume 3, 7.

Zela, A.; Siems, J.; and Hutter, F. 2020. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. *arXiv preprint arXiv:2001.10422*.

Zhou, D.; Zhou, X.; Zhang, W.; Loy, C. C.; Yi, S.; Zhang, X.; and Ouyang, W. 2020. EcoNAS: Finding Proxies for Economical Neural Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.