Vision-Oriented Lightweight Neural Architecture Search with Budget-Adaptive Evaluation

Yi Fan Yu-Bin Yang*
State Key Laboratory of Novel Software Technology, Nanjing University
Nanjing 210023, China
fanyiplus@smail.nju.edu.cn yangyubin@nju.edu.cn

Abstract

In the deep-learning-based computer vision community, Neural Architecture Search (NAS) has become the de-facto tool for acquiring task-optimal network structures. Nevertheless, NAS methods are trapped in a fundamental accuracy-efficiency dilemma: training-based approaches deliver reliable performance but incur prohibitive search costs, whereas training-free strategies are ultra-fast but often yield relatively unreliable rankings. To reconcile this conflict, we propose a vision-oriented lightweight training-based NAS framework. We first design six micro vision tasks whose training time is negligible, yet together they probe a broad spectrum of representational capacities. Built upon these tasks, we introduce a budget-adaptive performance evaluator to produce the most accurate ranking attainable within the limit. Experiments on popular NAS benchmarks show that our method achieves a ranking correlation higher than existing methods. Furthermore, we construct a search space from prevalent neural blocks and run our method at a cost close to training-free methods; the discovered architecture surpasses the current state-of-the-art under identical training recipes. Our codes are available at https://github.com/fanyi-plus/tf-nas.

1. Introduction

With the rapid development of deep learning, neural architectures have come to dominate a wide range of computer-vision tasks [51]. The task of designing network architectures for different scenarios is becoming increasingly burdensome [22, 27, 43]. To free engineers from this drudgery, Neural Architecture Search (NAS) is proposed [42]. NAS automates the trial-and-error loop and introduces learning-based strategies that steer the search toward promising regions of the design space. Nevertheless, conventional NAS still demands prohibitive amounts of computation and wall-clock time [2]. To remove the training cost altogether, training-free NAS has subsequently emerged [56]. Instead of fully training each candidate, these approaches score architectures at (or close to) random initialization; the score is called a *proxy*. A good proxy is strongly correlated with the true test performance, yet it is obtained after only a handful of forward/backward passes plus some light-weight statistics on weights, gradients, or intermediate features. Evaluating one proxy therefore takes only seconds, and the entire search can finish on a single GPU within a few hours. To emphasise the contrast, earlier NAS methods are now referred to as *training-based* NAS.

This dramatic speed-up, however, comes at a price. First, the ranking accuracy of training-free proxies drops compared with training-based competitors. Some proxies are heuristics that measure the expressivity or trainability of an untrained network [17, 34, 45], but the behavior of the network may change unpredictably after training. Other proxies are supported by rigorous theory [10, 50, 57], yet the proofs invariably rely on strong assumptions that often fail to hold in practice. Empirically, architectures discovered by training-free NAS underperform those found by training-based methods in the same search space. Second, training-free proxies are tightly coupled to specific architectural families. As shown in [61], directly applying Convolutional Neural Network (CNN)-oriented proxies to Transformer search spaces yields catastrophic results. Several proxies even require particular layers such as ReLU [39]; without them the proxy cannot be computed. To date, most training-free methods target either CNNs or Transformers, while Mamba-like [20] or newly proposed architectures (*e.g.*, RWKV [15]) have no corresponding proxies. Even if specialised proxies were designed tomorrow, they would still be restricted to their respective families, falling short of a universal solution. NAS therefore seems to be trapped in a fundamental accuracy-efficiency dilemma: training-based NAS is accurate and general but prohibitively slow, whereas training-free NAS is fast but less accurate and highly specialised.

*Corresponding author.

To break this dilemma, we draw inspiration from [41], which proposes an efficient performance predictor based on a battery of synthetic token-level tasks that probe a model's ability to memorise, retrieve, and compress. We observe, however, that those tasks assume one-dimensional discrete tokens whose solutions can be guessed from positional indices—an assumption that is incompatible with the two-dimensional, geometrically sensitive nature of images. We therefore redesign six vision-oriented probing tasks, including Local-Global Jigsaw (LGJ), Occlusion In-Painting (OIP), Rotation Match (RM), Color-Shape Binding (CSB), Motion Forecast (MF), and Visual Memorization (VM). We aggregate scores across these six tasks to produce a comprehensive evaluation of candidate networks. Building upon this, we further observe that Probing Task Configuration (PTC) significantly influences both evaluation accuracy and computational overhead. Consequently, given a fixed search-time budget, determining how to program PTC to obtain the most accurate possible evaluation of candidate networks emerges as another question. We focus on two key PTCs, dataset size D and number of training epochs E, treating them as independent variables, with the best online score S on the validation set serving as the dependent variable. A quadratic surface is then fitted to these observations, and the optimal operating point under the time budget is found by constrained optimization. Experiments show that our method attains state-of-the-art ranking correlation and search results under various time consuming budgets compared to existing training-based and training-free NAS methods.

Our contributions are summarised as follows:

- We design six vision-specific probing tasks that can estimate model quality without full training.
- We propose a principled method to balance dataset size and number of epochs under a user-specified time budget, yielding the best possible proxy score.
- Empirically, our method achieves state-of-the-art ranking correlation and downstream task performance under various time consuming budgets.

2. Related work

Early NAS approaches train thousands of candidate architectures from scratch [3, 35, 44, 48], incurring prohibitive GPU costs. Subsequent research mitigates the burden through weight-sharing super-networks (one-shot NAS) [13, 21] and differentiable relaxation (DARTS-family) [36, 54], which amortise training cost and convert the discrete search into continuous optimisation. Nevertheless, a super-network still has to be fully converged, and the ranking bias introduced by shared weights remains an open issue [8]. To further boost the quality of the final model, OFA [4], BigNAS [59] and AttentiveNAS [53] introduce progressive shrinking, the sandwich rule or Pareto-sampling for additional re-training and multi-objective refinement. All of these methods, however, still follow the "train-validate-retrain" paradigm; when the dataset or the deployment constraint changes, the entire pipeline has to be repeated, leading to considerable human and computational expense. To completely eliminate training, recent studies advocate "zero-cost proxies" that estimate the potential of a randomly-initialized network. Parameter-level proxies such as Synflow [49], SNIP [32], and GraSP [52] sum the saliency scores borrowed from pruning literature; gradient/activation-level proxies such as Gradnorm [1], Fisher [37], and Jacobian determinant [39] quantify trainability or expressivity using statistics collected from a single forward-backward pass. Other training-free NAS methods leverage different criteria to assess network performance, such as the nuclear norm of specific matrices related to the network's state and input dynamics [17, 45]. While training-free NAS methods have shown promise in CNNs and Transformers, their application to Mamba networks remains underexplored. More notably, existing training-free NAS methods still face a pronounced trade-off between generality and accuracy.

3. Method

3.1. Evaluation of vision models

In this subsection, we first describe the six probing tasks for evaluating vision models, and then introduce the complete pipeline for the evaluation. For ease of understanding, we provide illustrations of each probing task and the entire evaluation process for a candidate network in Section A of the Supplementary Material.

3.1.1. Local-global jigsaw

A foundational principle in visual recognition is the integration of local fragments into a global whole [23]. To instantiate this, we design a task that partitions a fixed-size image into several patches, randomly shuffles their spatial order, and requires the model to output the correct permutation indices. Owing to the spatial continuity and semantic co-occurrence statistics of natural images, texture, color, and shape cues at patch edges provide alignment signals of varying strength, and the model that can capture these long-range dependencies will efficiently locate the unique solution in permutation space.

To construct the dataset, we sample images from ImageNet-1K. For each sample, we first resize it to a uniform resolution of $h \times w$ via bilinear interpolation, then divide the image into patches of size $h_p \times w_p$. If h_p does not divide h or w_p does not divide w, we pad the edges using mirror padding along the corresponding dimension. The number of patches is thus $p = \lceil h/h_p \rceil \lceil w/w_p \rceil$. We stack all patches along a new dimension, shuffle them, and obtain a tensor of shape $p \times h_p \times w_p$ that serves as the model input

$\boldsymbol{x}$. The model output $\boldsymbol{y}$ is required to be a vector of length p, where the i-th element denotes the original position of patch $\boldsymbol{x}_i$ in the source image.

Regarding the evaluation metric, we consider two aspects. First, using raw accuracy may overlook the model's sensitivity to local adjacency relationships. We therefore employ average error distance—the mean Euclidean distance between predicted patch locations and ground-truth locations. Second, some image regions may be intrinsically indistinguishable: for example, two patches that both contain only pure sky blue lack distinguishing features, and penalizing the model for failing to differentiate them is unreasonable. Consequently, we also compute the cosine similarity between the predicted patch and its ground-truth counterpart; if the similarity exceeds a threshold γ, the pair is deemed indistinguishable and excluded from the error calculation. The final per-sample metric is

$$\text{Score}_{\text{LGJ}} = \frac{\sum_{i=1}^{p} \mathbb{1}_{c\left(\boldsymbol{x}_i, \boldsymbol{x}_{g(i)}\right)<\gamma} \left\|\mathcal{P}\left(g\left(i\right)\right) - \mathcal{P}\left(\boldsymbol{y}_i\right)\right\|_2}{\sum_{i=1}^{p} \mathbb{1}_{c\left(\boldsymbol{x}_i, \boldsymbol{x}_{g(i)}\right)<\gamma}}, \tag{1}$$

where $g(i)$ denotes the original position index of the i-th patch after shuffling, $c(\cdot,\cdot)$ is the cosine similarity function, and $\mathcal{P}(\cdot)$ maps a patch index to its two-dimensional coordinates.

3.1.2. Occlusion in-painting

Real-world objects are frequently occluded by railings, text overlays, or sensor noise, compelling visual systems to make plausible inferences within information voids [28]. We therefore design a task that places a black mask of random size in an image, blends in an extraneous image patch as noisy occlusion, and asks the model to regress the original pixel values of the masked region.

To construct the dataset, we sample images from ImageNet-1K and resize them to $h \times w$ via the same procedure described in Sec. 3.1.1. We then randomly select a mask height o_h from $[o_{\text{h\_min}}, o_{\text{h\_max}}]$ and a mask width o_w from $[o_{\text{w\_min}}, o_{\text{w\_max}}]$. A rectangular region $\mathcal{M}$ of size $o_h \times o_w$ is chosen in the image and its pixel values are set to zero. Simultaneously, we extract a patch of size $o_h \times o_w$ from another randomly selected image, paste it into the mask region, and set its opacity to α to simulate noisy interference. The resulting image $\boldsymbol{x}$ serves as the model input; we require the model output $\boldsymbol{y}$ to approximate the original $h \times w$ image as closely as possible.

In many generative vision tasks, the loss function typically avoids L2 to prevent inflated scores caused by blurry averaging, with perceptual metrics from pre-trained models often used as alternatives [9]. We argue, however, that pre-trained models introduce substantial biases and are therefore unsuitable for inclusion in our probing tasks. Consequently, we retain the L2 loss and additionally introduce a variance-penalty term to encourage the variance of the model-generated image to match that of the original. The final per-sample metric is

$$\text{Score}_{\text{OIP}} = \|\boldsymbol{y} - \boldsymbol{o}\|_2 + \mu \left|\text{std}\left(\boldsymbol{y}\right) - \text{std}\left(\boldsymbol{o}\right)\right|, \tag{2}$$

where $\boldsymbol{o}$ denotes the tensor corresponding to the occluded region, $\text{std}(\cdot)$ computes the standard deviation, and μ is a manually specified hyperparameter.

3.1.3. Rotation match

Geometric invariance represents a fundamental hard requirement for visual systems [30]. In this task, we simultaneously present the model with an original image and a randomly rotated version, requiring it to output the rotation angle. Unlike the jigsaw task which focuses on spatial ordering, this examines directional equivariance—whether the network can decouple geometric transformations from content itself.

The dataset is also sourced from ImageNet-1K, but here no resolution adjustment is necessary. Because rotating an image by an arbitrary angle causes some pixels to fall outside the boundaries while introducing external regions into the image interior, we designate the valid sample region as circular. Specifically, we first extract a square region $\mathcal{S}$ of side length h from the original image using the method described in OIP. We then inscribe the largest possible circular mask $\mathcal{C}$ within $\mathcal{S}$; clearly, $\mathcal{C}$ is centered at the square's center with radius $h/2$. To eliminate the shortcut that "image content inherently possesses an upward orientation" and to better assess the model's geometric equivariance capability, we also rotate the original image. We randomly select two angles θ_1 and θ_2 from $[0^\circ, 360^\circ)$ and rotate $\mathcal{C}$ counterclockwise about its center by θ_1 and θ_2 to obtain $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, which serve as the pre- and post-rotation images, respectively. During rotation, for each pixel in the target image we compute its corresponding coordinates in the source image; since these coordinates may not be integers, we again use bilinear interpolation to determine their pixel values. After obtaining $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we concatenate them along the width dimension to produce $\boldsymbol{x}$, which serves as the model input. The model's output $\boldsymbol{y}$ is required to predict the rotation angle.

It should be noted that due to the periodicity of angular values, directly regressing the angle and using the distance between predicted and ground-truth values as a loss is problematic. For instance, when an image is rotated counterclockwise by 359°, a model output of 0° would incur a large loss despite being highly accurate. To address this, we map angle values onto the unit circle. Specifically, we map the ground-truth angle θ_g to a point $\hat{\boldsymbol{\theta}}_g = (\cos\theta_g, \sin\theta_g)$ on the unit circle in $\mathbb{R}^2$, and similarly normalize the model output $\boldsymbol{y} = (y_1, y_2)$ to the unit circle, *i.e.*, $\hat{\boldsymbol{y}} = \left(y_1/\sqrt{y_1^2+y_2^2},\ y_2/\sqrt{y_1^2+y_2^2}\right)$. The evaluation met-

ric is then the Euclidean distance between $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{\theta}}_g$:

$$\text{Score}_{\text{RM}} = \left\| \hat{\boldsymbol{y}} - \hat{\boldsymbol{\theta}}_g \right\|_2 . \tag{3}$$

3.1.4. Color-shape binding

In many visual scenes, the system must not only recognize object attributes but also correctly bind them to the corresponding objects [7]. To this end, we present the model with two simple geometric shapes, each painted a distinct color, and require it to output the correct binding between color and shape class.

Unlike previous tasks, here we eschew natural images and instead generate a minimalist geometric world programmatically. Specifically, on an $h \times w$ canvas we randomly place two simple shapes (*e.g.*, circles, rectangles, or triangles) and assign each a unique color. To prevent overlap, we conduct collision detection during placement, ensuring the Intersection-over-Union (IoU) of the two objects is zero. Moreover, to block texture shortcuts, all shapes are filled with solid colors and the background is set to a uniform gray level. This rendered image serves as the model input $\boldsymbol{x}$. Let c_1 denote the number of shape categories and c_2 the number of colors, yielding $c_1 c_2$ possible color-shape combinations. The task thus becomes a $c_1 c_2$-class classification problem, and we require the model to output the correct class $\boldsymbol{y}$.

The evaluation metric for this task is binary, taking values of 1 or 0. A correct classification receives a score of 1, otherwise 0:

$$\text{Score}_{\text{CSB}} = \mathbb{1}_{\boldsymbol{y}=\boldsymbol{y}_g}, \tag{4}$$

where $\boldsymbol{y}_g$ is the ground-truth color-shape binding. We deliberately avoid intermediate scores (*e.g.*, assigning 0.5 for partially correct predictions) because the task emphasizes simultaneous recognition and binding of both attributes; from its perspective, a network that can identify only partial attributes is no better than one that is entirely nonfunctional.

3.1.5. Motion forecast

For dynamic vision tasks such as video tracking and autonomous driving motion estimation, models are required not only to perceive the present but also to forecast future states [16]. To this end, we simulate the elastic bouncing of a white ball on a black background; after being shown a sequence of preceding frames, the model must predict the ball's center coordinates in the subsequent frame. Owing to the ball's initial velocity, boundary elasticity, and external force fields, this task demands that the network possess the capacity to model low- or high-order dynamics.

This task likewise eschews real-world data, instead generating trajectories programmatically. Specifically, on a black canvas of size $h \times w$, we randomly initialize the position and velocity of a white ball, randomly set the boundary's elastic coefficient, and prescribe a gravitational field for the system. A physics engine is then employed to simulate the ball's bouncing motion on the canvas, producing a sequence of consecutive frames. As the generation pipeline can be fully implemented in PyTorch, we omit a detailed description for brevity. Let each trajectory comprise T frames. The model input $\boldsymbol{x}$ is the image sequence of the first $T-1$ frames, which is a tensor of shape $(T-1) \times h \times w$, and we require the model to output the two-dimensional coordinates $\boldsymbol{y}$ of the ball's center in the T-th frame.

The evaluation metric for this task can naturally be the L2 error:

$$\text{Score}_{\text{MF}} = \left\| \boldsymbol{y} - \boldsymbol{y}_g \right\|_2 , \tag{5}$$

where $\boldsymbol{y}_g$ denotes the ground-truth ball center coordinates.

3.1.6. Visual memorization

Long-tail distributions are the norm in real-world vision deployments: rare categories often contain only a handful of samples, yet models must still memorize them accurately [63]. This task therefore quantifies the resulting memory-generalization trade-off via an extreme few-shot regime.

Specifically, we randomly draw k classes from ImageNet-1K and sample m_{train} and m_{test} instances from each class's training and test splits, respectively, forming a k-class classification task. Here m_{train} is set to a very small number, typically on the order of 10^2 or fewer, to simulate a few-shot learning scenario. To reduce evaluation overhead, m_{test} is also kept small. The first phase excludes any data augmentation to isolate bare memorization capacity; the second phase then permits standard augmentation to observe the improvement in generalization.

Test accuracy is the conventional metric for classification. However, for such few-shot tasks we especially wish to penalize overfitting: a severely overfitted model that achieves excellent training accuracy may still obtain an acceptable test score, yet such a model would likely fail in practical deployment. We therefore incorporate the gap between training and test accuracy as an overfitting penalty. The evaluation metric is

$$\begin{aligned} \text{Score}_{\text{VM}} =& p_{\text{test}} - (p_{\text{train}} - p_{\text{test}}) \\ =& 2p_{\text{test}} - p_{\text{train}}, \end{aligned} \tag{6}$$

where p_{train} and p_{test} denote accuracy on the training and test datasets, respectively.

3.1.7. The calculation of score

The scoring formulas above except Eq. (6) produce per-sample scores. During evaluation we compute the mean over a batch of samples to obtain the final score for each task. After acquiring the six task-specific scores, we compute an overall score, S. We observe that the scales differ across tasks and that higher scores are better for CSB and VM, whereas lower scores are better for the remaining four

tasks. We therefore normalize each task score to a common range and direction via subtracting the mean, dividing by the standard deviation, and taking the negative of LGJ, OIP, RM, and MF. Then we sum them to produce the final score that guides architecture selection.

Overall, our probing tasks rely solely on the ImageNet-1K dataset; no additional datasets, models, or manual annotations are required. Moreover, the tasks are intentionally simple: in OIP we restrict occlusions to rectangular masks, and in CSB we render only two planar shapes. This design enables promising architectures to demonstrate learning capability after minimal training, thereby distinguishing candidate networks with low computational overhead.

3.2. Model evaluation at the designated time

In the preceding subsection we present a method for efficiently evaluating a candidate network. However, as noted in Sec. 1, the PTC can substantially affect both the accuracy of evaluation and its time cost. Two critical variables influencing PTC are dataset size D and training epochs E. While larger D and E generally yield more accurate evaluations, they also increase time cost. Consequently, under a limited budget T_{max}, the question of how to allocate D and E to obtain the most accurate evaluation in the shortest time becomes paramount. Here we propose an automated protocol based on continuous response-surface approximation that treats both dataset size and number of epochs as differentiable resources, obtaining via closed-form saddle-point solution the configuration that maximizes per-unit-time gain.

We first consider a formal framework. Let the best online score S on the validation set be an unknown function $S(D, E)$ of dataset size D and epochs E. Given a total time budget T_{max}, the per-sample acquisition time t_d and per-parameter-update time t_e can be pre-calibrated via low-cost probe experiments. The total time cost thus obeys the linear constraint $D \cdot t_d + E \cdot t_e \leq T_{\text{max}}$. The objective is to find the (D, E) pair within this half-plane that maximizes S. Inspired by scaling-law studies for language models [25], we can assume that S is twice differentiable in the region of interest, and then approximate it with a low-order polynomial, reducing the number of measurements to a handful.

Specifically, we adopt the quadratic response surface $\hat{S}(D, E) = \beta_0 + \beta_1 D + \beta_2 E + \beta_3 D^2 + \beta_4 E^2 + \beta_5 DE$ as a local surrogate model. Estimating the coefficients $\beta_0 \sim \beta_5$ requires at least five anchor points. We use a Latin hypercube design to uniformly place $(D_{\text{low}}, E_{\text{low}})$, $(D_{\text{high}}, E_{\text{low}})$, $(D_{\text{low}}, E_{\text{high}})$, $(D_{\text{high}}, E_{\text{high}})$, and the center point $(D_{\text{mid}}, E_{\text{mid}})$ within the budget ellipse. The maxima D_{max} and E_{max} are back-calculated from T_{max} as $D_{\text{max}} = T_{\text{max}}/(2t_d)$ and $E_{\text{max}} = T_{\text{max}}/(2t_e)$, ensuring all anchors strictly lie within the feasible region. For each anchor, we perform three independent training runs and take the median as the response value to suppress variance due to random seeds. With the five tuples (D_i, E_i, S_i), we solve for $\beta_0 \sim \beta_5$ via weighted least squares, yielding an analytical surface. Next, by setting partial derivatives to zero, we obtain the zero-gradient point of $\hat{S}$: $\beta_1 + 2\beta_3 D + \beta_5 E = 0$ and $\beta_2 + 2\beta_4 E + \beta_5 D = 0$. Solving this linear system yields the candidate saddle point $(D_{\text{saddle}}, E_{\text{saddle}})$. If this point lies within the budget ellipse, it is adopted directly; if it falls outside, we invoke the Lagrange multiplier method to find the extremum on the boundary $D \cdot t_d + E \cdot t_e = T_{\text{max}}$, obtaining the closed-form solution

$$D^* = \frac{2\beta_4(T_{\text{max}} - \beta_5 D_{\text{saddle}} t_e) - \beta_5 t_d \beta_2}{\Delta}, \tag{7}$$

$$E^* = \frac{T_{\text{max}} - D^* t_d}{t_e}, \tag{8}$$

where $\Delta = 4\beta_3\beta_4 t_d t_e - \beta_5^2 t_d t_e$. This analytic solution avoids numerical iteration and completes in milliseconds, ensuring the entire decision loop remains within human-acceptable interaction latency.

The reliability of the surface approximation is assessed via lack-of-fit at the center point. If the relative error exceeds a preset threshold, we collect four additional sub-anchors around the saddle point, merge them with the existing data, and refit. This adaptive refinement step typically completes in one round, with the total number of anchor points never exceeding nine, thereby strictly bounding the total GPU time within T_{max}. Through this process, we obtain the optimal D and E that maximize utilization of the prescribed time budget.

3.3. Overall search process

Integrating the two subsections above yields the complete model architecture search procedure. First, a total time budget $T_{\text{max,all}}$ and a suitable search space must be specified. Based on the search space size, the number of candidate networks n is then determined empirically; experience suggests that $n = 500 \sim 1000$ is reasonable when the search space is on the order of 10^3 [18, 34]. The time budget for each candidate follows as $T_{\text{max}} = T_{\text{max,all}}/n$, and the optimal data amount D_i^* and training epochs E_i^* for each candidate are obtained via the method in Sec. 3.2. We then sample n distinct architectures from the search space, compute their scores S using the method in Sec. 3.1, and select the highest-scoring network as the final result.

In practical scenarios where the search result must meet specific resource constraints (*e.g.*, parameter number, FLOPs, or inference latency), we compute the relevant costs for each sampled network and discard any that violate the constraints, resampling until a sufficient number of compliant candidates is obtained. Furthermore, our approach is compatible with more sophisticated search strate-

Table 1. Ranking correlation of various training-free NAS methods. Best results in bold.

dataset	NAS-Bench-101				NAS-Bench-201				NATS-Bench				NAS-Bench-301			
top-k	all	20%	10%	5%	all	20%	10%	5%	all	20%	10%	5%	all	20%	10%	5%
#Param	80.73	61.58	68.49	52.00	79.76	64.53	52.65	58.67	81.18	60.65	49.22	50.00	76.77	52.57	54.94	53.33
Gradnorm [1]	78.18	51.88	49.39	43.33	76.36	63.03	61.47	30.00	77.39	54.71	46.94	26.67	77.89	49.98	52.65	47.33
Synflow [49]	77.20	44.81	50.04	58.00	78.40	43.96	51.18	56.00	75.79	56.00	52.98	29.33	73.16	49.41	43.67	26.67
GraSP [52]	76.58	51.03	44.16	37.33	77.29	47.15	41.39	52.00	74.02	48.97	36.49	36.67	72.35	32.32	56.41	52.00
Fisher [37]	72.71	45.66	43.02	29.33	72.23	49.98	47.92	45.33	73.14	40.32	56.24	54.00	72.68	36.40	36.49	48.00
Snip [32]	76.91	49.45	39.76	20.00	76.23	55.64	54.29	**64.67**	77.47	40.08	51.18	44.00	74.79	52.85	44.82	30.67
Zen-NAS [34]	84.83	63.56	73.55	68.00	84.34	65.66	67.67	55.33	83.39	65.90	59.35	**72.67**	80.85	57.17	**59.35**	62.00
ZiCo [33]	83.28	66.95	62.45	73.33	84.38	66.59	57.55	52.67	82.78	63.43	58.20	34.67	79.23	56.04	41.22	61.33
MeCo [29]	84.71	58.10	52.65	52.67	83.89	60.48	55.43	60.00	83.22	62.42	62.61	52.00	79.18	56.65	49.39	51.33
AZ-NAS [31]	85.12	60.48	34.37	16.00	83.28	58.99	56.57	60.00	82.89	55.68	50.86	67.33	81.90	64.53	58.04	49.33
SWAP-NAS [40]	79.92	58.51	46.29	46.00	78.83	58.55	48.57	28.67	77.63	50.59	50.53	39.33	76.51	49.05	23.76	32.00
Auto-Prox [55]	80.82	60.57	66.37	48.67	80.88	56.97	44.82	42.67	80.30	61.66	66.20	51.33	76.49	49.78	36.00	47.33
UP-NAS [26]	82.37	58.59	51.18	61.33	83.46	64.65	58.20	50.67	82.32	55.88	60.49	52.67	78.70	54.75	47.76	**69.33**
DSS++ [61]	67.25	44.73	40.41	19.33	70.57	43.07	27.84	28.00	71.06	40.89	36.16	38.00	67.13	39.68	43.02	41.33
AttnNAS [17]	58.36	22.91	23.76	34.67	57.10	27.92	27.35	10.00	57.28	37.90	31.10	12.67	53.26	23.19	19.51	11.33
ours	**92.10**	**78.51**	**75.51**	**79.33**	**93.31**	**81.33**	**76.82**	64.00	**89.27**	**76.32**	**68.82**	63.33	**86.00**	**68.12**	56.90	44.00
dataset	ViT-Bench-101-A				ViT-Bench-101-P				Vim				VMamba			
top-k	all	20%	10%	5%	all	20%	10%	5%	all	20%	10%	5%	all	20%	10%	5%
#Param	59.87	36.12	25.55	29.33	58.89	37.41	36.65	29.33	50.47	23.39	12.00	15.33	49.47	18.71	24.57	2.67
Gradnorm [1]	47.03	44.18	30.90	32.00	47.23	47.90	32.20	30.67	49.42	33.72	40.69	35.33	42.79	31.33	46.57	46.67
Synflow [49]	18.06	17.98	27.18	25.33	8.73	17.62	23.43	17.33	12.13	2.14	22.61	19.33	15.48	6.71	23.59	18.00
GraSP [52]	19.57	10.14	19.51	4.67	14.38	6.99	2.04	5.33	13.71	2.71	8.08	33.33	22.02	7.76	19.18	5.33
Fisher [37]	13.57	2.30	4.33	5.33	14.91	8.24	18.20	10.00	16.63	11.92	14.29	4.00	17.17	18.14	6.61	17.33
Snip [32]	21.01	9.74	31.10	16.67	19.55	16.93	16.73	7.33	19.85	6.30	6.12	16.00	17.87	0.81	7.10	16.00
Zen-NAS [34]	20.21	18.18	31.76	8.00	27.70	12.00	17.22	27.33	18.97	4.69	3.18	7.33	20.02	22.22	22.29	28.00
ZiCo [33]	20.75	18.06	17.71	28.67	21.58	4.81	12.98	20.67	21.30	10.38	17.39	11.33	20.86	22.10	16.90	27.33
MeCo [29]	27.95	0.24	4.65	9.33	26.54	14.67	13.96	8.00	24.28	11.19	2.37	4.00	25.66	28.48	12.33	18.00
AZ-NAS [31]	31.34	10.26	15.43	25.33	23.84	8.93	10.69	13.33	22.48	9.62	4.33	10.67	22.89	16.61	24.24	8.00
SWAP-NAS [40]	21.76	10.18	0.41	22.00	19.50	8.32	10.69	6.67	18.23	7.88	4.33	32.67	19.00	12.12	3.02	8.00
Auto-Prox [55]	20.38	9.09	4.33	5.33	21.03	2.55	0.08	21.33	20.04	15.88	7.27	15.33	19.72	18.95	19.35	10.67
UP-NAS [26]	19.21	14.99	6.45	0.00	20.04	20.81	20.65	4.00	29.22	1.70	13.47	10.00	23.72	15.64	2.20	15.33
AC [45]	69.62	48.20	42.86	23.33	64.55	41.82	38.12	17.33	-	-	-	-	-	-	-	-
HI [45]	78.53	53.45	38.12	37.33	74.66	43.52	40.24	43.33	-	-	-	-	-	-	-	-
HC [45]	81.14	49.66	52.49	51.33	78.41	55.15	45.96	**57.33**	-	-	-	-	-	-	-	-
DSS++ [61]	80.35	60.36	**64.57**	68.00	76.86	61.01	50.04	34.00	29.28	4.28	2.04	24.00	28.14	17.41	7.27	24.67
AttnNAS [17]	65.83	40.97	25.88	34.67	62.74	26.71	15.92	11.33	28.71	11.72	10.37	42.00	24.80	21.05	10.69	0.67
ours	**86.38**	**69.13**	64.24	**80.67**	**80.71**	**61.82**	**57.22**	38.67	**86.48**	**64.40**	**62.78**	**56.00**	**87.15**	**61.05**	**53.31**	**64.67**

gies such as evolutionary algorithms [61] or Bayesian optimization [5], which begin with a small initial sample and use the resulting scores to guide subsequent sampling.

4. Experiments

To verify the effectiveness of our method, we first compare the ranking correlation with existing training-free NAS methods, and then benchmark the search results. To demonstrate generality, all experiments are conducted on multiple network families. We provide all hyperparameter values we use in our experiments in Section B of the Supplementary Material.

4.1. Ranking correlation

CNN, Transformer and Mamba are three classic families widely used in vision tasks. We therefore compare the ranking correlation of existing training-free NAS methods on all three. To obtain the correlation, one should first fix a search space, uniformly samples candidate networks, trains and tests them to obtain ground-truth performace (top-1 accuracy in image classification), and finally computes the correlation between the proposed proxy and the performace. To benchmark competitors, we repeat the same procedure with their proxies.

For CNNs we adopt the four public benchmarks, NAS-Bench-101 [58], NAS-Bench-201 [12], NATS-Bench [14] and NAS-Bench-301 [46], which already contain the ground-truth accuracies, saving us from re-training and guaranteeing fairness. For Transformers we use ViT-Bench-101 [55] about AutoFormer [6] and ViT-Bench-101 about PiT [24] (ViT-Bench-101-A and ViT-Bench-101-P in short respectively) in the same way. Unfortunately, no public NAS benchmark exists for Mamba; we therefore hand-craft two spaces for Vim [62] and VMamba [38], sample 500 architectures from each, train them from scratch and record the best validation performace.

We use Kendall-τ as the correlation metric. Following prior work [11, 60], we also report correlations restricted to the top-20%, top-10% and top-5% highest-performace networks, since the correct ordering among good architectures is often more important than that among poor ones. The results are given in Tab. 1. We note that AC, HI, and HC

Table 2. Search results of various training-based NAS methods. S, P, F, L, T-1, T-5, SS, MS are abbreviations for search time (GPU hours), parameter number (M), FLOPs (G), latency (s), top-1 accuracy (%), top-5 accuracy (%), mIoU (SS), and mIoU (MS), respectively. For object detection, we adopt the Mask R-CNN 3× MS schedule. In our method, the labels S (short), M (median), and L (long) correspond to search times of 1, 2, and 3 GPU hours, respectively. Best results in bold.

Network	Method	S	P	F	L	CL		DT						SG	
						T-1	T-5	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	SS	MS
CNN	MOTE-NAS [60]	3.1	16	3.8	1.27	77.2	94.1	46.3	64.2	49.7	28.8	53.6	50.0	41.8	45.0
	QuantNAS [19]	17	16	3.4	1.60	79.3	96.1	47.6	65.8	50.6	29.7	55.0	50.9	42.0	45.7
	RobustDNAS [47]	179	16	4.0	1.40	79.1	96.5	47.6	65.9	51.4	29.5	55.5	51.1	43.7	46.1
	AttentiveNAS [53]	420	15	3.5	1.18	75.2	91.0	44.5	62.1	48.6	28.8	51.6	48.4	40.5	43.0
	OFA [4]	27	15	3.4	1.08	75.3	92.3	45.6	62.5	48.3	28.8	52.7	48.6	40.9	44.5
	ours (S)	1	17	3.8	1.14	78.7	96.7	47.5	66.4	52.0	30.3	55.4	51.3	42.7	47.3
	ours (M)	2	16	4.1	1.42	80.2	97.3	49.1	66.2	51.3	29.4	55.1	52.4	43.8	47.3
	ours (L)	3	17	4.1	1.10	**80.9**	**97.9**	**49.8**	**67.1**	**52.1**	**30.7**	**55.7**	**52.9**	**44.3**	**47.6**
Transformer	MOTE-NAS [60]	3.3	16	3.8	1.73	77.0	94.0	46.2	64.5	50.0	28.7	53.5	49.9	41.4	44.7
	QuantNAS [19]	18.2	17	4.2	1.16	78.7	95.6	47.7	65.5	51.0	29.7	54.5	50.6	42.5	46.0
	RobustDNAS [47]	190	16	3.7	1.63	79.2	96.9	46.6	66.1	52.0	29.6	54.9	51.4	43.2	46.5
	AttentiveNAS [53]	453	15	4.1	0.37	74.3	91.2	44.8	61.6	48.6	27.7	51.6	48.5	40.5	43.5
	OFA [4]	31	16	3.2	0.88	75.5	92.8	45.4	63.4	49.2	28.1	52.4	49.1	41.3	43.8
	ours (S)	1	17	4.2	1.05	79.2	97.2	47.9	65.9	51.3	28.9	55.1	50.9	43.4	46.2
	ours (M)	2	16	4.6	0.86	80.3	97.4	48.1	67.2	51.6	30.6	55.9	52.4	44.0	**47.6**
	ours (L)	3	17	4.4	1.25	**80.5**	**98.2**	**49.3**	**67.8**	**52.2**	**31.3**	**56.6**	**52.5**	**44.1**	**47.6**
Mamba	MOTE-NAS [60]	3.0	16	3.8	1.01	77.8	93.8	45.9	64.1	49.4	28.4	53.7	49.5	41.7	45.2
	QuantNAS [19]	18.1	16	4.0	2.20	78.5	95.9	47.4	65.5	50.8	29.7	54.9	51.4	42.4	46.4
	RobustDNAS [47]	209	17	3.9	1.49	78.8	95.8	47.3	65.9	51.9	29.7	54.9	51.8	43.1	46.2
	AttentiveNAS [53]	437	15	3.5	1.49	74.7	91.0	44.9	61.4	48.9	28.2	52.2	49.0	40.7	43.6
	OFA [4]	32	15	4.2	0.56	75.7	91.5	45.1	63.5	48.1	28.1	52.7	49.5	41.7	43.5
	ours (S)	1	17	4.1	1.20	80.1	97.0	47.7	66.3	50.9	29.2	55.4	51.0	43.4	46.5
	ours (M)	2	16	4.6	1.67	80.2	97.6	47.9	67.3	51.8	29.1	**56.0**	52.6	43.1	46.9
	ours (L)	3	17	3.7	0.68	**80.8**	**98.0**	**48.1**	**67.5**	**52.5**	**30.5**	**56.0**	**52.8**	**44.2**	**47.2**

require attention maps in their proxy computation, which are tensors unique to Transformers; therefore, we only report their results on ViT-Bench-101-A and ViT-Bench-101-P. Across the eight spaces our method consistently yields the highest Kendall-τ. Competitor training-free NAS methods not only trail almost behind in their native family, but collapse almost to zero when transferred to other families. For example, GraSP is a CNN-oriented method; therefore, it performs well on the four CNN-based benchmarks, while performing poorly on the other four benchmarks. This confirms that our method generalises well across CNNs, Transformers and Mambas.

We emphasise that training-based NAS methods are excluded from the correlation study. All contemporary training-based methods embed candidates in a supernetwork or allocate disproportionate training budget to promising candidates, so at any given search step different architectures have been trained for different lengths of time. Their intermediate accuracies are therefore not comparable and cannot be used to compute a meaningful ranking correlation; we compare with these methods only at the performace of the search results in the next subsection.

4.2. Performance of search results

In this subsection we modify the previous search spaces to create one large space for CNNs, one for Transformers, and one for Mamba. We describe the search space in detail in Section C of the Supplementary Material. We then perform search in each space. To verify performance under different time budgets, we run search with three distinct budgets—1, 2 and 3 GPU hours—in every space. The number of candidates is 500, and we also constrain the model's parameter number to no more than 17 M. The search results are trained from scratch and tested on three vision tasks—image CLassification (CL), object DeTection (DT), and semantic SeGmentation (SG)—and compared with multiple training-based NAS methods.

Table 2 shows the comparison with training-based NAS methods. Our method matches or surpasses all training-based NAS methods while using far less search time, demonstrating that it can obtain excellent search results with low time cost. Compared to the latest training-free NAS methods, our method still demonstrates excellent performance. The detailed results are presented in Section D of the Supplementary Material. So across multiple network families and time budgets, our method consistently achieves better results than existing NAS methods on several vision tasks, illustrating its strong effectiveness and generality.

4.3. Ablation study

Task-set completeness. To verify whether all six proposed tasks are necessary, we exhaustively evaluate every non-empty subset of tasks on NAS-Bench-301. Following the method in Sec. 4.1, we compute the Kendall-τ between proxy scores and ground-truth accuracies while keeping the total time budget identical for all subsets. Figure 1 sum-

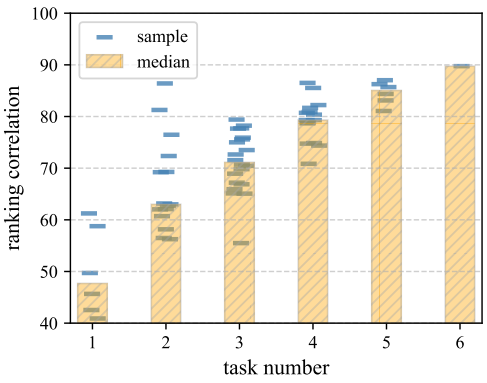

Figure 1. Ranking correlation of combining tasks.

Table 3. Top-1 accuracy of search results on CL using different PTCs under the time budget 1 GPU hour.

Variable	D	E	D & E
CNN	77.5	77.3	78.7
Transformer	78.0	77.2	79.2
Mamba	78.3	78.5	80.1

marises the results: ranking correlation monotonically increases with the number of tasks, and no subset of 1∼5 tasks ever reaches the correlation obtained with the full six-task combination. According to our method, when fewer tasks are used, the dataset or the number of epochs will be enlarged to reach the time budget; nevertheless, the correlation still lags behind, indicating that the additional data/episodes cannot substitute for the missing task diversity. This observation confirms that the six tasks are complementary and that each contributes unique information to the final proxy.

Dual-variable optimality. Section 3.2 introduces a quadratic-surface model that treats dataset size D and training epochs E as joint variables to maximise proxy quality under a time constraint. To justify the effects of both variables, we conduct an ablation that freezes one variable and optimises the other via a one-dimensional quadratic fit. We conduct the search process again on three search space used in Sec. 4.2 under the time budget 1 GPU hour, and then train the search results on CL and test their top-1 accuracy. As shown in Tab. 3, the single-variable settings consistently underperform the dual-variable regime, demonstrating that both D and E exert significant, non-redundant influence on the proxy reliability and that omitting either variable degrades the final ranking quality.

4.4. Discussion

The results of one-dimensional probing task to vision models. To substantiate our claim that the six probing tasks proposed in [41] are ill-suited for vision models, we replace our tasks with theirs and repeat the experiment on NAS-Bench-301, ViT-Bench-101-A and VMamba under exactly configuration in Sec. 4.1. As shown in Tab. 4, the resulting Kendall-τ is markedly lower than that obtained with our tasks, corroborating our analytical arguments.

Table 4. Ranking correlation on NAS-Bench-301 using the one-dimensional probing tasks in [41]. NB-301 and VB-101-A are abbreviations for NAS-Bench-301 and ViT-Bench-101-A respectively.

top-k	all	20%	10%	5%
NB-301	24.68	20.40	24.08	4.67
VB-101-A	68.81	46.91	40.90	40.00
VMamba	23.14	7.96	4.98	11.33

Table 5. Ranking correlation with and without averaging evaluation results on several LR/WD combinations. NB-301 and VB-101-A are abbreviations for NAS-Bench-301 and ViT-Bench-101-A respectively.

Benchmark	NB-301	VB-101-A	VMamba
w/o LR/WD	86.00	86.38	87.15
w/ LR/WD	85.91	86.72	85.99

Sensitivity to learning-rate and weight-decay scheduling. In our probing tasks we keep Learning Rate (LR) and Weight Decay (WD) fixed. By contrast, [41] averages scores of each task obtained with several (learning-rate, weight-decay) tuples. To see whether such a multi-schedule strategy benefits our method, we replicate the experiments in 4.1 on on NAS-Bench-301, ViT-Bench-101-A and VMamba, while sampling three LR/WD combinations per candidate and averaging the scores. Table 5 shows no statistically significant gain in final architecture quality; because the multi-schedule variant multiplies evaluation time by the same factor, we conclude that it is unnecessary for our proxy design.

5. Conclusion

We propose a lightweight NAS method tailored for vision models. Six deliberately engineered vision tasks serve as rapid proxies for architectural quality. Built upon these proxies, we further introduce a time-budget-aware performance estimator that adaptively allocates data and epochs to maximise ranking fidelity within any user-specified deadline. Experiments on several NAS benchmarks and search space demonstrate that our method achieves markedly higher ranking correlation and better search results. In the future we will explore automatic re-weighting of the six tasks according to the target visual application.

Acknowledgments

This work was supported by the Natural Science Foundation of China (Grant No. 62176119), and the Jiangsu Graduate Research Innovation Program (Grant No. KYCX24_0262).

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021. 2, 6

[2] Sarwat Ali and M Arif Wani. Gradient-based neural architecture search: A comprehensive evaluation. *Machine Learning and Knowledge Extraction*, 5(3):1176–1194, 2023. 1

[3] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, 2017. 2

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 2, 7

[5] Andrés Camero, Hao Wang, Enrique Alba, and Thomas Bäck. Bayesian neural architecture search using a training-free performance metric. *Applied Soft Computing*, 106: 107356, 2021. 6

[6] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12270–12280, 2021. 6

[7] Na Chen and Katsumi Watanabe. Color–shape associations affect feature binding. *Psychonomic bulletin & review*, 28 (1):169–177, 2021. 4

[8] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on computer vision*, pages 12239–12248, 2021. 2

[9] Steffen Czolbe, Oswin Krause, Ingemar Cox, and Christian Igel. A loss function for generative neural networks based on watson's perceptual model. *Advances in Neural Information Processing Systems*, 33:2051–2061, 2020. 3

[10] Tu Do and Ngoc Hoang Luong. Training-free multi-objective evolutionary neural architecture search via neural tangent kernel and number of linear regions. In *International Conference on Neural Information Processing*, pages 335–347. Springer, 2021. 1

[11] Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Zimian Wei, Qiang Wang, and Xiaowen Chu. Parzc: Parametric zero-cost proxies for efficient nas. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 16327–16335, 2025. 6

[12] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*. 6

[13] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3681–3690, 2019. 2

[14] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3634–3646, 2021. 6

[15] Yuchen Duan, Weiyun Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, Yu Qiao, Hongsheng Li, Jifeng Dai, and Wenhai Wang. Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures. *CoRR*, 2024. 1

[16] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9710–9719, 2021. 4

[17] Yi Fan and Yu-Bin Yang. Training-free neural architectural search on transformer via evaluating expressivity and trainability. In *2024 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2024. 1, 2, 6

[18] Yi Fan, Zhong-Han Niu, and Yu-Bin Yang. Data-aware zero-shot neural architecture search for image recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023. 5

[19] Tianxiao Gao, Li Guo, Shanwei Zhao, Peihan Xu, Yukun Yang, Xionghao Liu, Shihao Wang, Shiai Zhu, and Dajiang Zhou. Quantnas: quantization-aware neural architecture search for efficient deployment on mobile device. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1704–1713, 2024. 7

[20] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 1

[21] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision*, pages 544–560. Springer, 2020. 2

[22] Yunusa Haruna, Shiyin Qin, Abdulrahman Hamman Adama Chukkol, Abdulganiyu Abdu Yusuf, Isah Bello, and Adamu Lawan. Exploring the synergies of hybrid convolutional neural network and vision transformer architectures for computer vision: A survey. *Engineering Applications of Artificial Intelligence*, 144:110057, 2025. 1

[23] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15013–15022, 2021. 2

[24] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11936–11945, 2021. 6

[25] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030, 2022. 5

[26] Yi-Cheng Huang, Wei-Hua Li, Chih-Han Tsou, Jun-Cheng Chen, and Chu-Song Chen. Up-nas: Unified proxy for neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1675–1684, 2024. 6

[27] Li Hui, Ahmed Ibrahim, and Riyadh Hindi. Computer vision-based concrete crack identification using mobilenetv2 neural network and adaptive thresholding. *Infrastructures*, 10(2):42, 2025. 1

[28] Jie Jiang. Occlusion-aware stroke-based drawing, inbetweening, and painting. 2022. 3

[29] Tangyu Jiang, Haodi Wang, and Rongfang Bie. Meco: zero-shot nas with one data and single forward pass via minimum eigenvalue of correlation. *Advances in Neural Information Processing Systems*, 36:61020–61047, 2023. 6

[30] Julio A Kovacs and Willy Wriggers. Fast rotational matching. *Biological Crystallography*, 58(8):1282–1286, 2002. 3

[31] Junghyup Lee and Bumsub Ham. Az-nas: Assembling zero-cost proxies for network architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5893–5903, 2024. 6

[32] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018. 2, 6

[33] Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot nas via inverse coefficient of variation on gradients. *arXiv preprint arXiv:2301.11300*, 2023. 6

[34] 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, 5, 6

[35] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018. 2

[36] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2

[37] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, pages 7021–7032. PMLR, 2021. 2, 6

[38] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. *Advances in neural information processing systems*, 37:103031–103063, 2024. 6

[39] 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, 2

[40] Yameng Peng, Andy Song, Haytham M Fayek, Vic Ciesielski, and Xiaojun Chang. Swap-nas: Sample-wise activation patterns for ultra-fast nas. In *ICLR*, 2024. 6

[41] Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Re, et al. Mechanistic design and scaling of hybrid architectures. In *International Conference on Machine Learning*, pages 40908–40950. PMLR, 2024. 2, 8

[42] Matt Poyser and Toby P Breckon. Neural architecture search: A contemporary literature review for computer vision applications. *Pattern Recognition*, 147:110052, 2024. 1

[43] Ravi Raj and Andrzej Kos. An extensive study of convolutional neural networks: applications in computer vision for improved robotics perceptions. *Sensors*, 25(4):1033, 2025. 1

[44] 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 *International conference on machine learning*, pages 2902–2911. PMLR, 2017. 2

[45] Aaron Serianni and Jugal Kalita. Training-free neural architecture search for rnns and transformers. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023. 1, 2, 6

[46] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 4:14, 2020. 6

[47] Konstanty Subbotko, Wojciech Jablonski, and Piotr Bilinski. The devil is in discretization discrepancy. robustifying differentiable nas with single-stage searching protocol. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1665–1674, 2024. 7

[48] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2820–2828, 2019. 2

[49] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020. 2, 6

[50] Ondrej Tybl and Lukas Neumann. Training-free neural architecture search through variance of knowledge of deep network weights. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14881–14890, 2025. 1

[51] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018(1):7068349, 2018. 1

[52] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020. 2, 6

[53] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6418–6427, 2021. 2, 7

[54] Wenna Wang, Xiuwei Zhang, Hengfei Cui, Hanlin Yin, and Yannnig Zhang. Fp-darts: Fast parallel differentiable neural architecture search for image classification. *Pattern Recognition*, 136:109193, 2023. 2

[55] Zimian Wei, Peijie Dong, Zheng Hui, Anggeng Li, Lujun Li, Menglong Lu, Hengyue Pan, and Dongsheng Li. Auto-prox: Training-free vision transformer architecture search via automatic proxy discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 15814–15822, 2024. 6

[56] Meng-Ting Wu and Chun-Wei Tsai. Training-free neural architecture search: A review. *ICT Express*, 10(1):213–231, 2024. 1

[57] Tomomasa Yamasaki, Zhehui Wang, Tao Luo, Niangjun Chen, and Bo Wang. Rbflex-nas: Training-free neural architecture search using radial basis function kernel and hyperparameter detection. *IEEE Transactions on Neural Networks and Learning Systems*, 2025. 1

[58] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International conference on machine learning*, pages 7105–7114. PMLR, 2019. 6

[59] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pages 702–717. Springer, 2020. 2

[60] Yuming Zhang, Jun Hsieh, Xin Li, Ming-Ching Chang, Chun-Chieh Lee, and Kuo-Chin Fan. Mote-nas: Multi-objective training-based estimate for efficient neural architecture search. *Advances in Neural Information Processing Systems*, 37:100845–100869, 2024. 6, 7

[61] Qinqin Zhou, Kekai Sheng, Xiawu Zheng, Ke Li, Yonghong Tian, Jie Chen, and Rongrong Ji. Training-free transformer architecture search with zero-cost proxy guided evolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1, 6

[62] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024. 6

[63] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922, 2014. 4