Designing Real-Time Neural Networks by Efficient Neural Architecture Search

Zitong Bo[1,2], Yilin Li[1,2], Ying Qiao[1](✉), Chang Leng[1], and Hongan Wang[1]

[1] Institute of Software, Chinese Academy of Sciences, Beijing, China
{zitong2019,qiaoying}@iscas.ac.cn

[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Convolutional Neural Networks (CNNs) are extensively used in cyber-physical systems for tasks like object detection and semantic segmentation. Given the critical nature of these systems, the ability of CNNs to meet stringent timing constraints is as crucial as their accuracy. However, variability in CNN execution times can lead to time constraint violations, affecting system reliability. To address this, we introduce RetNAS, an efficient neural architecture search framework that combines a rapid Worst-Case Execution Time (WCET) estimator and a constraint schedule to ensure timely performance. We utilize extreme value theory to estimate WCET, leveraging a generalized Pareto distribution based on limited execution time samples. Furthermore, RetNAS employs a constraint schedule to accelerate the search efficiency, which pursues a gradual search trajectory. RetNAS significantly enhances search efficiency and achieves a success rate of approximately 99.2% in meeting time constraints, outperforming other methods. Our experiments also show that CNNs designed by RetNAS surpass the accuracy of manually designed CNNs and visual transformers by 0.4% to 5%.

Keywords: Convolutional Neural Network · Neural Architecture Search · Real-time Systems

1 Introduction

Convolutional Neural Networks (CNNs) have significantly advanced in recent decades, impacting various fields which demand high accuracy and strict adherence to time constraints. In critical applications such as autonomous driving, failing to meet these constraints can have severe consequences [7]. However, the inherent variability in CNN execution times makes it challenging to design architectures that achieve both high accuracy and meet stringent timing requirements [19].

Manually designed lightweight CNNs offer a balance between execution time and accuracy but lack customization for specific time constraints. Neural Architecture Search (NAS) [1, 12] automates CNN design, optimizing both accuracy and execution time [17, 26]. However, these approaches often fail to meet real-time application requirements where strict time constraints are crucial. Real-time systems require programs that can guarantee Worst-Case Execution Time (WCET) within set limits to prevent catastrophic failures [24]. Existing NAS algorithms often struggle to consistently ensure that each execution adheres to time constraints due to considerable variations in CNN execution times. Accurately determining the WCET of a CNN typically demands extensive testing, leading to substantial time overhead [18]. Additionally, NAS involves searching through an immensely large search space, where evaluating the WCET for each sampled architecture incurs prohibitive time costs.

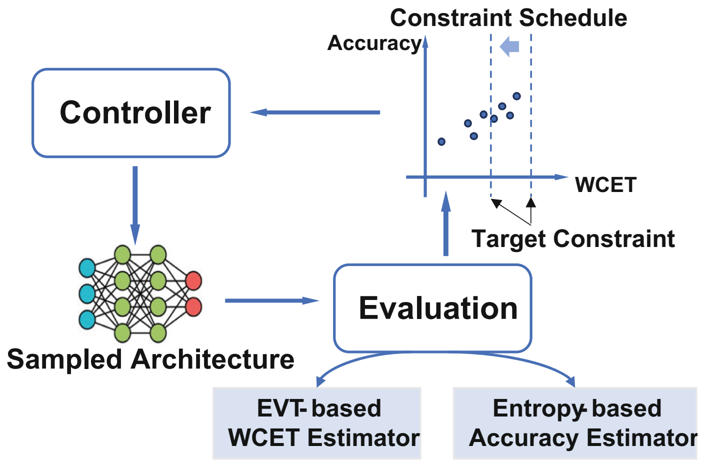


Fig. 1. Overview of our RetNAS framework. RetNAS integrates an EVT-based WCET estimator and a constraint schedule method to meet the time constraints of real-time applications.

To address this problem, we propose an efficient NAS framework named RetNAS (Real-time Neural Architecture Search), specifically tailored for designing real-time CNNs for time-critical systems. To the best of our knowledge, this is the first initiative to focus on neural network architecture design under stringent time constraints. Figure 1 presents an overview of RetNAS. Building upon the entropy-based NAS framework [23], RetNAS innovatively redefines the evaluation process by incorporating a novel WCET estimator based on Extreme Value Theory (EVT). This approach enables rapid WCET estimation without exhaustive testing, which significantly reduces the computational demand in NAS. RetNAS also integrates a constraint schedule, instrumental in guiding the search process toward architectures that meet these strict time constraints. Experimental results demonstrate that RetNAS can identify CNN architectures that not only achieve the desired accuracy levels but also adhere to specified time constraints. This dual-objective strategy is essential for enhancing the functionality of autonomous systems, ensuring both high accuracy and reliability in real-time applications. The contributions of this paper are summarized as follows:

- We propose an EVT-based WCET estimator for rapid and accurate WCET estimation for CNNs.
- We introduce a constraint schedule in steering the search process towards meeting aggressive time constraints.

– Through comprehensive experimental evaluations, we validate the effectiveness of RetNAS, demonstrating its superior performance in meeting time constraints and achieving high accuracy.

The remainder of this paper is organized as follows: Sect. 2 reviews related work in neural network design and time analysis. Section 3 presents the problem formulation for designing real-time CNNs. Section 4 details the RetNAS framework, while Sect. 5 describes extensive experiments to evaluate RetNAS's effectiveness. Conclusions are finally drawn in Sect. 6.

2 Related Work

2.1 Neural Network Design

CNN design is predominantly manual, discovering new design choices like residual connections [6] and vision transformer [13] to enhance accuracy and reduce complexity. The shift from manual to automatic design has been facilitated by NAS [1]. However, NAS are usually very time-consuming in evaluating accuracy and time. Predictor-based methods encode architectures into high-dimensional vectors to streamline the process [16]. One-shot methods reduce training costs by leveraging a large supernet [12]. Moreover, zero-shot methods, such as those utilizing Zen-Score [11] and Entropy [23], replace traditional accuracy metrics with alternative measures to further minimize training costs. For time-critical system designs, studies aim to optimize accuracy and latency, often by minimizing or imposing constraints on FLOPs [12] or real execution time [17, 26]. ZenNAS [11] and DeepMAD [23] focus on managing average execution time, while MnasNet [26] and EdgeNAS [17] use a weighted product of accuracy and execution time. These methods do not ensure each CNN execution meets specific time constraints, highlighting the need for neural architecture design methodologies specifically tailored for real-time applications.

2.2 Time Analysis of Neural Networks

Time is a pivotal factor in real-time applications, where delays can impact user experience or result in failures in safety-critical systems [24]. The execution times of CNNs can vary due to factors like input data, network dynamics, and memory contention [7, 19]. This variability necessitates rigorous time analysis to ensure CNNs operate within the strict time constraints of real-time environments. WCET analysis is essential for maintaining temporal accuracy, requiring CNNs to stay within predefined execution limits [21]. Static methods often face practical limitations and can be overly pessimistic, typically requiring access to proprietary code. Statistical methods provide empirical insights but lack the certainty needed for strict WCET compliance [5].

3 Problem Formulation

The design of real-time CNNs is framed as an optimization problem with dual objectives: maximizing accuracy and adhering to strict time constraints. We define the architecture space $\mathcal{A}$and seek an architecture a that optimizes accuracy while satisfying a time constraint C, which reflects the specific requirements of real-time applications. The optimization problem can be expressed formally as follows:

$$\underset{a \in \mathcal{A}}{maximize} \quad Accuracy(a) \tag{1}$$

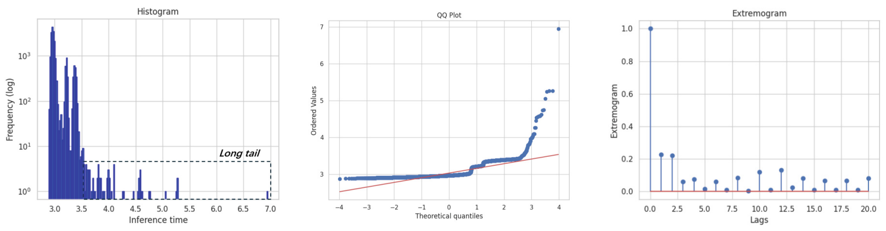
$$s.t. \;\; WCET(a) \leq C \tag{2}$$

Unlike previous approaches that focus on average execution time, our method emphasizes strict adherence to time constraints, a necessity in real-time systems. The variability in execution times and the complexity of determining a CNN's WCET make finding optimal architectures challenging[1]. Accurately obtaining $WCET(a)$ is resource-intensive, requiring extensive inference time tests and potentially consuming thousands of GPU-days. The narrow portion of the search space meeting stringent time constraints complicates the NAS process, making feedback from architectures that do not meet time constraints ineffective and the search nearly random and inefficient.

4 RetNAS Framework

4.1 EVT-Based WCET Estimator

Accurately estimating the WCET of a CNN architecture is crucial when applying NAS to design CNN. To delve into the nature of CNN execution times, we analyzed CNN execution times by conducting experiments with ResNet18 on an NVIDIA A100 GPU, which revealed a significant long-tail distribution. This observation led us to apply EVT [3], traditionally used in meteorology and finance, to model the extreme events in CNN execution times. After confirming the independence of extreme value data with extremogram tests, we employed the Pickands-Balkema-de Haan Theorem to more precisely model the tail distributions using a Generalized Pareto Distribution (GPD) [20] (Fig. 2).



(a) Histogram (b) Quantile-Quantile Plot (c) Extremogram

Fig. 2. Statistical analysis of execution time for the ResNet18.

[1] Given the challenges in achieving a strict WCET, minor deviations are permissible, aligning with the principles of soft real-time systems.

Theorem 1 (Pickands-Balkema-de Haan Theorem). *Let X be a random variable with distribution function F. If F belongs to a broad class of distribution functions, for any $y > 0$, the conditional excess distribution function over a threshold u converges to the GPD as u approaches the upper endpoint of F:*

$$\lim_{u \to x_F} \Pr\left(\frac{X-u}{\sigma(u)} \le y | X > u\right) = G(y), \tag{3}$$

where x_F is the right endpoint of the distribution of X, and $G(y)$ represents the GPD function.

This theoretical foundation supports our approach using EVT to estimate the WCET for CNNs. The probability density function (PDF) of the GPD captures the behavior of data beyond high thresholds efficiently, providing a robust framework for our estimations:

$$f(x; \mu, \xi, \sigma) = \begin{cases} \frac{1}{\sigma}(1+\xi(\frac{x-\mu}{\xi}))^{-\frac{1}{\xi}-1}, & \xi \neq 0 \\ \frac{1}{\sigma}e^{-\frac{x-\mu}{\sigma}}, & \xi = 0 \end{cases} \tag{4}$$

where μ is the location parameter, $\sigma > 0$ is the scale parameter, and ξ is the shape parameter.

Algorithm 1 EVT-based WCET Estimator

```
Require: Sorted CNN execution time $X = \{x_1, \dots, x_n\}$, threshold $\mu$, desired confidence level α
Ensure: Estimated WCET
  $x_{threshold} = x_{\lfloor \mu \times n \rfloor}$
  $X_{tail} = \{x_i \in X \mid x_i > x_{threshold}\}$
  $\hat{\xi}, \hat{\sigma} = \arg\max_{\xi,\sigma} \sum_{i=1}^{n} \log f\left(x_j; \mu, \xi, \sigma\right), x_j \in X_{tail}$
  $WCET = \mu + \frac{\hat{\sigma}}{\hat{\xi}}\left(\left(\frac{1}{1-\alpha}\right)^{\hat{\xi}} - 1\right)$
  Return $WCET$
```

Our proposed EVT-based WCET estimator begins by collecting execution time samples from different CNN architectural elements, which are then fitted to a GPD. This process involves running forward propagation *n* times per CNN architecture with a batch size of 1, producing *n* execution time recordings that are sorted from smallest to largest. The tail of this distribution is analyzed by setting a threshold for tail data and fitting the GPD using the maximum likelihood method for all samples exceeding this threshold. The WCET is estimated based on the scale parameter $\hat{\sigma}$ and shape parameter $\hat{\xi}$ of the GPD, which calculates the execution time corresponding to a specified confidence interval. This efficient method requires a minimal number of test samples, making it ideal for rapid WCET estimation in NAS contexts. The procedure is summarized in **Algorithm 1**.

4.2 Constraint Schedule

Designing efficient CNNs to meet aggressive time constraints is challenging because only a minor segment of the architecture search space may satisfy these stringent requirements. Ineffective feedback during the search process occurs when sampled architectures do not meet the time constraints. To address this, we introduce a constraint schedule method that progressively directs the search towards these constraints. Initially, constraints are set to a lenient level, allowing for broad exploration of architectural possibilities, fostering creativity and innovation. As the search progresses, constraints are gradually tightened, guiding the exploration towards architectures that increasingly align with the target constraints. This dynamic adjustment ensures that by the final iterations, the architecture not only meets but is optimized for the initial constraints.

The constraint schedule divides the search process into M segments, assuming a total of T NAS iterations. Each segment, denoted by the t^{th} round, corresponds to the $\lfloor t \cdot M/T \rfloor^{th}$ segment. The initial time constraint is C_0, and the final target constraint is C. We explore two types of constraint schedules: linear and exponential. The linear schedule provides a steady progression of constraints:

$$C_{linear}(t) = C_0 + \left\lfloor \frac{t \cdot M}{T} \right\rfloor \cdot \frac{1}{M} \cdot (C - C_0) \tag{5}$$

The exponential schedule adjusts constraints more rapidly at first and then more slowly, offering a nuanced control:

$$C_{exp}(t) = C_0 \cdot \left(\frac{C}{C_0} \right)^{\left\lfloor \frac{t \cdot M}{T} \right\rfloor \cdot \frac{1}{M}} \tag{6}$$

4.3 Maximum Entropy-Based Search

Efficiently computing accuracy is crucial in NAS for ranking architectures [11, 26]. Recent studies [23] have demonstrated a positive correlation between the entropy of a CNN and its accuracy. Hence, we utilize CNN entropy as a proxy for estimating accuracy.

Algorithm 2 RetNAS

```
Require: Time constraint $C$, total number of iterations $T$, population size $N$, initial architecture $a_0$
Ensure: The best architecture $a^*$
 Initialize population $P = \{a_0\}$.
 for $t = 1, 2, \ldots, T$ do
     Randomly select $a_t \in P$.
     Compute $C(t)$.
     Uniformly select a block $h$ in $a_t$.
     Uniformly alter the kernel size, width, depth within some range.
     if $WCET(\widehat{a_t}) \leq C(t)$ then
         Compute entropy $H(\widehat{a_t})$.
         Append $\widehat{a_t}$ to $P$.
     end if
     if $|P| > N$ then
         Remove the architecture with the lowest entropy.
     end if
 end for
 Return $a^*$, the architecture with the highest entropy in $P$.
```

Definition 1. *For a CNN layer i with input channels c_i, output channels c_{i+1}, kernel size k_i, and group g_i, the CNN operator performs matrix multiplication represented as $W_i \in R^{c_i \times c_{i+1} \times k_i^2 / g_i}$. For an L-layer network parameterized by $\{c_i, k_i, g_i, r_i\}_{i=1}^{L}$, its entropy is defined as:*

$$H \triangleq \log\left(r_{L+1}^2 c_{L+1}\right) \sum_{i=1}^{L} \log\left(c_i k_i^2 / g_i\right) \tag{7}$$

RetNAS's search space includes residual and bottleneck blocks as defined in ResNet [6, 22], with each block searched separately to optimize layer configurations. Each block's search space comprises input and output channel counts, kernel size, and number of layers. Following maximum entropy principles, RetNAS aims to maximize the overall entropy of the architecture to enhance accuracy.

The step-by-step description of RetNAS, detailed in **Algorithm 2**, utilizes an evolutionary algorithm as the architecture search controller, following the SOTA NAS framework [11, 23]. Alternative search controllers such as reinforcement learning [1, 26] or differentiable search methods [12, 17] could also be employed. During each iteration t, an architecture a_t within the population P is randomly selected and mutated. The mutation involves adjusting the attributes of the selected layer within a specified range, set to [0.5,2.0]. The newly mutated architecture $\widehat{a_t}$, if valid within the search space $\mathcal{A}$, is added to the population. The population is managed by removing architectures with the lowest entropy to maintain its size. After T iterations, the architecture with the highest entropy and a WCET that meets the time constraint C is selected as the output architecture.

5 Experiments

This section presents a series of experiments conducted on the CIFAR-100 (fine-label) [10] and ImageNet-1K [4] datasets to demonstrate the effectiveness of our proposed approach. The experiments aim to evaluate the performance of the EVT-based WCET estimator, compare RetNAS with manually designed ResNet architectures and other NAS methods, and assess the impact of individual components of RetNAS.

5.1 Experimental Settings

All architecture searches and evaluations were performed on an NVIDIA Tesla A100 GPU. The evolutionary population size was set at 512, with 20,000 evolutionary iterations. The threshold for the WCET estimation algorithm was set to 0.9 with a confidence level of 0.92. The search process was divided into ten segments by the constraint schedule, with initial time constraints set at 2.5 times the target constraints. For training the CNNs, we used the SGD optimizer with a momentum of 0.9 and a weight decay of 5e−4. The initial learning rate was 0.1, with a batch size of 256, employing cosine learning rate decay [15] with a 5-epoch warm-up. The training involved 1,440 epochs.

5.2 Evaluation for WCET Estimator

The initial part of our experiment validates the accuracy of the EVT-based WCET prediction method. We collected execution time data from various CNN architectures during the search process and assessed the EVT model fit to this data. Although statistical methods do not provide a strict WCET, we adopted the 99^{th} percentile of 10,000 sampling results as the WCET, following prior work [18]. We compared our method with two baseline techniques: the generalized extreme value (GEV) modeling [2] and the 99%-Observed method [18], using different sample sizes to evaluate mean absolute error (MAE) and average time cost. Table 2 demonstrates that our EVT-based WCET estimator outperforms the baselines in accuracy, achieving the precision of the 99%-Observed method with 4,000 samples using only 1,000 samples. This indicates our method's efficiency in gathering execution time data, underscoring its potential to enhance NAS evaluation processes significantly (Table 1).

5.3 Comparison with Hand-Crafted Neural Architecture

We evaluated RetNAS against established architectures including the ResNet [6], MobileNet(V3) [8], DenseNet [9], and GoogleNet [25] families. Specific models tested were ResNet18, ResNet34, ResNet50, ResNet101, MobileNet(V3)-S, MobileNet(V3)-L, DenseNet121, DenseNet-161. This evaluation focused on WCET and top-1 accuracy. For fair comparison, CNNs designed by RetNAS with linear scheduling were matched against ResNets with similar WCETs. Results on CIFAR-100 (Fig. 4) show RetNAS models achieving accuracies between 76.65% and 83.25% across WCETs from 3.9 ms to 16.8 ms, outperforming equivalent ResNet models in accuracy under similar WCET conditions (Fig. 3).

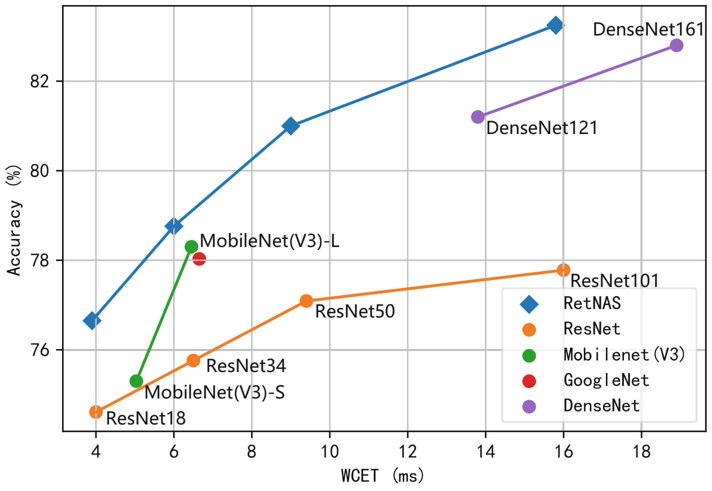


Fig. 3. WCET vs. top-1 accuracy of each model on CIFAR-100.

Table 1. Performance comparison of different WCET estimation methods.

	Number of Samples	MAE	Time Cost (s)
GPD	500	0.62	2.10
	1,000	0.32	4.20
GEV	500	1.03	2.10
	1,000	0.85	4.21
99%-Observed	500	0.85	2.09
	1,000	0.70	4.19
	2,000	0.52	8.39
	4,000	0.31	16.75

RetNAS also excelled in designing state-of-the-art CNNs for ImageNet-1K classification, compared against contemporary CNN and ViT models, as shown in Table 2. Despite the high performance of ViTs, their substantial computational demands make them less suitable for real-time applications. RetNAS-designed CNNs achieve superior accuracy with WCETs comparable to those of ResNets, highlighting RetNAS's efficiency in crafting high-accuracy, real-time neural network architectures.

5.4 Comparative Analysis with SOTA NAS Methods

To assess RetNAS's effectiveness in real-time neural network architecture design, we compared it against modified versions of two SOTA NAS methods, DeepMAD and MnasNet, adapted for real-time constraints. DeepMAD treats time constraints as rigid limits, while MnasNet integrates them into its optimization, seeking Pareto-optimal

Table 2. Comparison with SOTA ViT and CNN models on ImageNet-1K.

Model	# Param	WCET(ms)	Top-1 Accuracy(%)
ResNet50 [6]	26M	10.1	77.5
PVT-S [27]	25M	18.3	79.8
Swin-T [13]	29M	17.4	81.3
ConvNext-T [14]	29M	11.0	82.1
RetNAS	28M	**10.0**	**82.2**
ResNet101 [6]	45M	16.0	78.3
PVT-L [27]	61M	37.2	81.8
Swin-S [13]	50M	24.5	83.0
ConvNext-S [14]	50M	16.2	83.1
RetNAS	50M	**16.0**	**83.5**

solutions. Our comparative experiment targeted a WCET of 5 ms, evaluating metrics such as entropy (proxy for accuracy), time satisfiability, and search cost, defined as GPU-hours consumed by the NAS. Table 3 illustrates RetNAS's performance against DeepMAD and MnasNet, showcasing its superior ability to adhere to time constraints and optimize search efficiency. While MnasNet shows comparable entropy values, it significantly underperforms in time satisfiability, indicating inconsistencies in meeting time constraints. RetNAS, available in linear and exponential variants, demonstrates nuanced differences in handling constraints: the linear variant allows broader architectural exploration, whereas the exponential variant achieves slightly better time satisfiability due to more aggressive constraint intensification. A notable result is the high time satisfiability achieved by RetNAS, demonstrating the effectiveness of the EVT-based WCET estimator in real-time network design, surpassing other SOTA NAS methods in both efficiency and accuracy.

Table 3. Comparison with SOTA NAS methods on CIFAR-100.

Method	Entropy	Time Satisfiability(%)	Search Cost(hour)
DeepMAD-1k [23]	1057.0	86.5	19.6
DeepMAD-2k [23]	1047.3	94.2	38.3
MnasNet-1k [26]	1087.1	74.4	20.1
MnasNet-2k [26]	1085.3	85.3	39.8
RetNAS-linear	**1090.9**	99.0	21.2
RetNAS-exp	1088.1	**99.2**	20.5

6 Conclusion

RetNAS, our real-time neural architecture search framework, effectively balances high accuracy and strict time constraints in real-time applications. Our extensive analysis uncovers a significant long-tail characteristic in CNN execution times, addressed by our novel EVT-based WCET estimator utilizing GPD. Additionally, RetNAS employs a constraint schedule that strategically guides the search process. Validated through rigorous experiments, RetNAS produces CNN architectures that meet WCET constraints and maintain high accuracy. Future work will extend entropy estimation methods to other types of layers and improve RetNAS's hyperparameter sensitivity to enhance its practical application.

Acknowledgments. This work was supported by the Youth Innovation Promotion Association CAS (Grant No.2021103) and the CAS Project for Young Scientists in Basic Research (Grant No. YSBR-040).

References

1. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. In: International Conference on Learning Representations (2016)
2. Berezovskyi, K., Santinelli, L., Bletsas, K., Tovar, E.: WCET measurement-based and extreme value theory characterisation of CUDA kernels. In: Proceedings of the 22nd International Conference on Real-Time Networks and Systems, pp. 279–288. ACM, Versaille France (2014)
3. Cazorla, F.J., Kosmidis, L., Mezzetti, E., Hernandez, C., Abella, J., Vardanega, T.: Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey. ACM Comput. Surv. **52**(1), 1–35 (2019)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009)
5. Edgar, S., Burns, A.: Statistical analysis of WCET for scheduling. In: Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001) (Cat. No.01PR1420), pp. 215–224 (2001)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
7. Heo, S., Cho, S., Kim, Y., Kim, H.: Real-time object detection system with multi-path neural networks. In: 2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 174–187. IEEE, Sydney, Australia (2020)
8. Howard, A., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1314–1324 (2019)
9. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2261–2269. IEEE, Honolulu, HI (2017)
10. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
11. Lin, M., et al.: Zen-NAS: a zero-shot NAS for high-performance image recognition. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 337–346. IEEE, Montreal, QC, Canada (2021)
12. Liu, H., Simonyan, K., Yang, Y.: DARTS: differentiable architecture search. In: International Conference on Learning Representations (2019)

13. Liu, Z., et al.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012–10022 (2021)
14. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11976–11986 (2022)
15. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
16. Luo, R., Tan, X., Wang, R., Qin, T., Chen, E., Liu, T.Y.: Semi-supervised neural architecture search. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20, Curran Associates Inc., Red Hook, NY, USA (2020)
17. Luo, X., Liu, D., Kong, H., Liu, W.: EdgeNAS: discovering efficient neural architectures for edge systems. In: 2020 IEEE 38th International Conference on Computer Design (ICCD), pp. 288–295. IEEE, Hartford, CT, USA (2020)
18. Nigade, V., Bauszat, P., Bal, H., Wang, L.: Jellyfish: timely inference serving for dynamic edge networks. In: 2022 IEEE Real-Time Systems Symposium (RTSS), pp. 277–290 (2022)
19. Pang, W., Jiang, X., Lv, M., Gao, T., Liu, D., Yi, W.: Towards the predictability of dynamic real-time DNN inference. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **41**(9), 2849–2862 (2021)
20. Pickands III, J.: Statistical inference using extreme order statistics. The Annals of Statistics, pp. 119–131 (1975)
21. Puschner, P., Koza, C.: Calculating the maximum execution time of real-time programs. Real-Time Syst. **1**(2), 159–176 (1989)
22. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollar, P.: Designing network design spaces. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10425–10433. IEEE, Seattle, WA, USA (2020)
23. Shen, X., et al.: Deepmad: mathematical architecture design for deep convolutional neural network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6163–6173 (2023)
24. Stankovic, J.A., Ramamritham, K.: What is predictability for real-time systems? Real-Time Syst. **2**(4), 247–254 (1990)
25. Szegedy, C., et al.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
26. Tan, M., et al.: MnasNet: platform-aware neural architecture search for mobile. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2815–2823. IEEE, Long Beach, CA, USA (2019)
27. Wang, W., et al.: Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 568–578 (2021)