Multi-objective Architecture Search for Real-time Radar Target Detection on Embedded Systems

Noé Lallouet
LAMSADE, Université Paris-Dauphine
Thales DMS
Paris, France
noe.lallouet@thalesgroup.com

Tristan Cazenave
LAMSADE, Université Paris-Dauphine
Paris, France
tristan.cazenave@lamsade.dauphine.fr

Cyrille Enderli
Thales DMS
Elancourt, France
cyrille-jean.enderli@fr.thalesgroup.com

Léo Monnier
Thales DMS
Elancourt, France
leo.monnier@thalesgroup.com

***Abstract*—Deep neural networks are powerful models for radar detection tasks, but their high computational cost limits suitability for real-time inference on CPUs. In this paper, we address this limitation by proposing an approach to automatically search for optimized neural network architectures tailored to airborne target detection. Our method relies on neural architecture search with multi-objective Monte Carlo search that optimizes the trade-off between detection accuracy and latency constrained by arithmetic intensity, guiding the search towards CPU-efficient networks. Experimental results show that the proposed architecture matches the baseline detector's performance while being up to four times faster on CPU.**

***Index Terms*—radar, deep learning, CNN, NAS, MCTS**

I. Introduction

Radar detection and remote sensing using deep neural networks is a topic that has recently attracted interest in the research community [1]. Departing from classical signal processing tools, neural networks are increasingly used, due to their remarkable representative power. More specifically, deep networks thrive in complex environments where classical models need additional specific processing (e.g. clutter power transition regions or interferences). The target detection problem can be expressed as solving the following decision problem:

$$\begin{cases} H_0 : y(t) = \nu(t) : \text{absence of target} \\ H_1 : y(t) = x(t) + \nu(t) : \text{presence of target} \end{cases} \quad (1)$$

where $y(t)$ is the received signal, $x(t)$ is the signal of the target and $\nu(t)$ is noise, including thermal noise, clutter or jamming.

Typically, the computational resources available to the radar system are limited. These systems are often embedded on edge devices, where GPU inference may not be available. In particular, the memory footprint and inference latency of the neural network are of particular interest for deployment in real-life environments. However, despite advances towards efficient inference on CPU [2], [3] modern neural networks are often computationally heavy, and may be intractable in scenarios where real-time inference is required. This highlights the importance of designing neural networks that achieve high detection performance and low latency.

In this paper, we focus on the automated design of computationally efficient neural networks for the problem of radar detection with a Pulse-Doppler radar. The rate of predictions depends on the radar's pulse repetition interval (PRI), which may require inference latencies on the order of a few milliseconds, motivating the use of architecture search directed towards extremely low-latency networks. The contributions of this paper are the following: first, we propose the design of an radar image segmentation architecture search space optimized for low-latency networks. Second, we leverage constrained multi-objective neural architecture search to discover a set of architectures corresponding to optimal trade-offs between inference latency and detection performance and propose a novel search constraint based on arithmetic intensity. Finally, we benchmark these architectures against a robust baseline and report their performances.

II. Related works

The popularity of deep neural networks for computer vision tasks has driven researchers from the radar community to develop new approaches for radar detection based on neural networks [4]. [5] proposes a radar detector based on the Faster R-CNN architecture, while [6] implements a detector based on the FCN architecture, constrained by the Neyman-Pearson criterion. Others have also proposed architectures based on CNNs [7], [8] or Transformers [9]. Recently, [10] proposed a detector for semantic segmentation on multi-view radar data, fusing information taken from Range-Doppler (RD) and Rang-Angle (RA) tensors. Although the vast majority of these works are inspired by the classic operators in computer vision, some convolution operators specifically tailored for radar segmentation have been proposed [11], [12]. While the excellent segmentation performance of networks developed recently is acknowledged, their slower inference speed renders

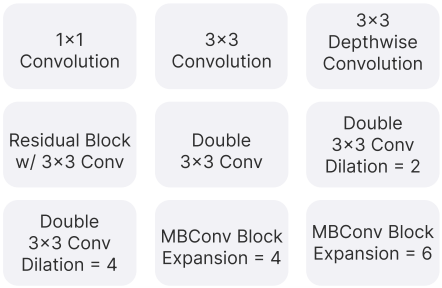


Fig. 1. Available operations for each stage

them intractable on CPU. Specifically, we benchmarked two networks with a very high segmentation performance on the CARRADA dataset [13]: MVNet [10] and AdaPKC [12]. The measured runtimes on CPU were respectively 52 ms and 115 ms, exceeding the 5 ms inference latency of the baseline used in this study, and thus falling outside the operational regime of interest.

Neural architecture search (NAS) [14] has been established as a popular framework for the automated design of novel architectures maximizing an objective, usually validation accuracy. NAS research, historically focused on delivering networks with high accuracy, has shifted towards the search for architectures satisfying requirements for performance and computational complexity [15]–[17]. A particular subfield of interest is multi-objective NAS, that aims to identify not only a single architecture, but the entire Pareto front of neural architectures with respect to some objectives [18]–[20]. Neural architecture search has been applied to automatically design neural networks for the task of radar target detection [21], classification [22] and to synthetic aperture radar segmentation and detection [23], [24].

III. Search space

In this work, we develop a neural architecture search space adapted to the radar target detection problem at hand. A large part of NAS approaches focus on cell-based search spaces, where the network consists of a searched cell (represented as a directed acyclic graph) stacked according to a predefined skeleton. Such approaches are capable of designing topologically novel, high performing architectures, but are ill-suited to extremely low-latency neural architecture search. Indeed, in such a frugal regime, the computation time of the cell is slowed by the frequent exchanges of data between the processor and the cache that are needed for sequentially computing $n(n-1)/2$ operations for n nodes. As such, we propose a macro search space consisting of only one operation per block of the neural network skeleton. We consider a lightweight U-Net [25] with the number of channels doubling at each stage, from 16 to 128, as the primary baseline for detection performance, as it has shown optimal detection performance on thermal noise and high robustness to clutter [26].

Here, the searched architecture remains topologically similar to the baseline U-Net, with a four-stage encoder and decoder separated by a bottleneck, and skip-connections between parallel encoder and decoder stages. For each stage, the agent chooses between nine convolution-based operators, visible in Figure 1. The number of convolutional channels for each stage is searchable, with values in [8, 16, 32, 48, 64, 128]. The size of the search space is thus 3.9×10^{15}. The search space is represented by a supernetwork S, a large architecture containing all possible subnetworks, as popularized by [27]. The search is performed by sequentially sampling subnetworks from S and training them via backpropagation for a small number of steps on the target task. We implement a weight-sharing approach where the sampled subnets inherit the weights of the supernet [28], allowing progressive training of all architectures contained in S.

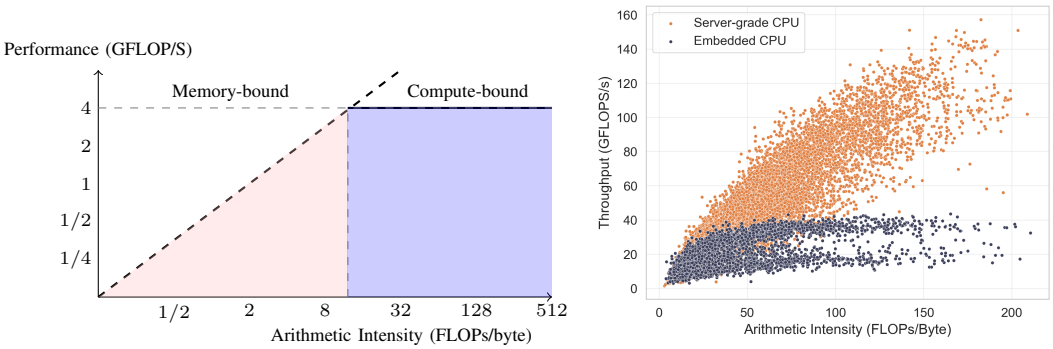
Due to the large dimension of the search space, finding the optimal Pareto front is computationally infeasible. As such, the proposed approach is a heuristic that asymptotically converges to the global Pareto front, but, due to a limited number of objective function evaluations, provides an approximation of the true Pareto front at the end of the search.

IV. Search strategy

The aim of the search is to identify a set of neural network architectures corresponding to different optimal values of the trade-off between detection performance and inference latency. To this end, we perform multi-objective neural architecture search. The majority of modern multi-objective neural architecture search methods aim to jointly optimize both the validation loss and the computational complexity of the sampled models. The computational complexity is often optimized through a proxy such as FLOPs [15], [19] or inference latency on target hardware [16]. However, recent works [29], [30] have shown that FLOPs or number of parameters are a flawed proxy for inference latency. Moreover, simply optimizing for inference latency allows subnets with low latency but with large structural inefficiencies to be found.

In order to guide the algorithm to the most efficient architecture, we leverage the roofline model [31] as a soft constraint applied to the objective functions during the search. The roofline model (Figure 2a) provides a performance estimate of an algorithm on a given device and is divided in two regions: a memory-bound region and a compute-bound region. An algorithm that lies in the compute-bound region is not limited by the transfer cost of data between the processor and the memory, but rather by the computational complexity of the operations on the processor. Figure 2b shows the arithmetic intensity against throughput of architectures randomly sampled from the search space implemented in Section III: the roofline diagram performance ceiling can be clearly seen on the resource-constrained consumer CPU, while more powerful server-grade CPUs are able to process operation-intensive architectures without reaching a throughput ceiling.

We use the concept of arithmetic intensity (AI) [31] to determine the structural efficiency of a neural network. Arithmetic intensity is calculated as the ratio of floating point operations (FLOPs) to memory traffic. During the search, if a sampled



(a) (b)

Fig. 2. (a) Roofline model illustration. (b) Roofline model visualized on the proposed search space, for random architectures in the search space where the latency was computed on a server-grade CPU (orange) and on a consumer-grade CPU (blue).

architecture possesses an AI value that lies in the memory-bound region of the target hardware's roofline diagram, a soft penalty is applied to the objective functions. Specifically, the multi-objective search optimizes the following objective functions:

$$\begin{cases} f_1(a) = & \mathcal{L}_{val}(a, X_{val}, Y_{val}, W) \\ f_2(a) = & \text{Latency}(a) + \lambda \max\left(0, \frac{\Gamma - AI(a)}{\Gamma}\right) \end{cases} \tag{2}$$

where $\mathcal{L}_{val}(a, X_{val}, Y_{val}, W)$ is the validation loss of architecture a inheriting weights W from the supernet, $AI(a)$ is the arithmetic intensity of a, λ = 0.01 and Γ is the AI necessary to lie in the compute-bound region of the target hardware. The value of Γ is calculated as:

$$\Gamma = \frac{n_{cores} \times f_c \times \text{vector\_size} \times 2}{B} \tag{3}$$

where n_{cores} is the number of cores of the processor, f_c is the clock frequency, vector_size is the number of operands per vector register, 2 is the factor needed to account for the one multiply and one add operation and B is the maximum memory bandwidth [32]. The scaling coefficient value $\lambda = 0.01$ was chosen to ensure that the magnitude of the AI constraint in this particular setting remains close to the magnitude of the latency during the search. While performing a grid search for the value of λ would be intractable due to the large size of the search space, this initialization correctly penalizes memory-bound networks without overpowering the initial latency minimization objective.

It is acknowledged that the multi-objective formulation in Equation 2 is dependent upon the target hardware's properties. However, the underlying framework is hardware-agnostic: running the multi-objective search on different hardware configurations only requires recalibration of the value of Γ and computation of the target latencies.

[16] proposes a NAS method that generates AI-optimized networks, but doesn't use AI as an objective during the search. To the best of our knowledge, AI has not been used as a search objective or a constraint in NAS outside of benchmarks. It is important to note that maximizing AI is not the goal of the search. Minimizing latency alone favors architectures with both low latency and low AI, which may result in networks underutilizing the computational capabilities of the processor and being bottlenecked by memory traffic. The constraint serves as a regularizer to enforce a minimal level of intensity, which results in higher-quality solutions. An ablation study (Section V-A) reveals that unconstrained search converges to memory-bound architectures that struggle to generalize, confirming the interest of the AI constraint.

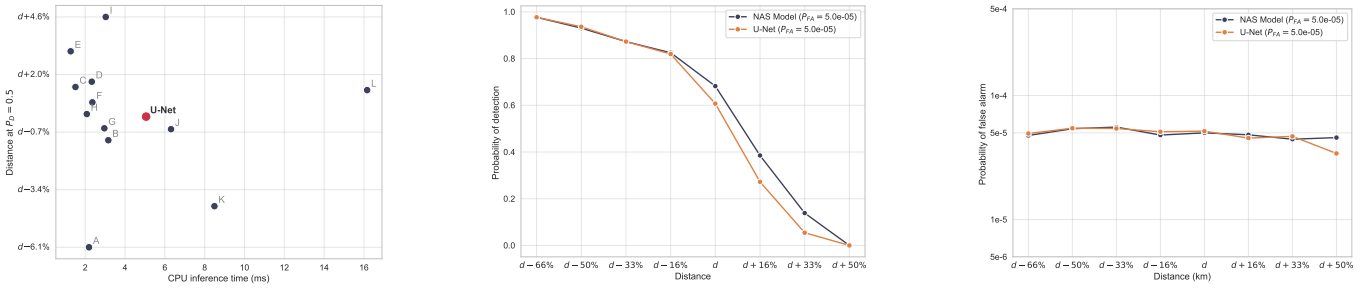
Monte Carlo search (MCS) has been used for neural architecture search with success [33], [34]. We propose to run the architecture search with a multi-objective variant of the UCT algorithm, Pareto-MCTS [35], to perform the search. The search runs for 25 000 objective function evaluations.

V. Experimental Results

TABLE I
METRICS OF THE BEST MODELS AFTER TRAINING. D@$P_D = 0.5$ (HIGHER IS BETTER, % OF BASELINE).

Model	N_{params}	CPU latency (ms)	D @ $P_D = 0.5$
A	59 585	2.19 ± 0.14	0.939
B	99 553	3.16 ± 0.15	0.989
C	100 047	1.51 ± 0.08	1.014
D	109 513	2.33 ± 0.11	*1.016*
E	169 593	$\mathbf{1.27 \pm 0.07}$	1.030
F	242 265	2.36 ± 0.11	1.006
G	264 177	2.96 ± 0.12	0.995
H	273 593	*2.08 ± 0.16*	1.001
I	296 153	3.03 ± 0.22	**1.046**
J	325 641	6.31 ± 0.49	0.994
K	410 841	8.49 ± 0.69	0.958
L	1 185 573	16.16 ± 0.71	1.012
Lightweight U-Net	1 964 888	5.06 ± 0.23	1.00

After performing the search, the non-dominated architectures found by the search algorithm are trained for 100 epochs from scratch on a dataset consisting of 80 000 range-Doppler maps generated by a realistic industrial simulation tool and



(a) (b) (c)

Fig. 3. (a) CPU inference time against distance @ $P_D = 0.5$ for each of the searched models. (b) Probability of detection P_D against target distance. (c) Probability of false alarm P_{FA} against target distance. Models were calibrated at $P_{FA} = 5 \times 10^{-5}$ on thermal noise only.

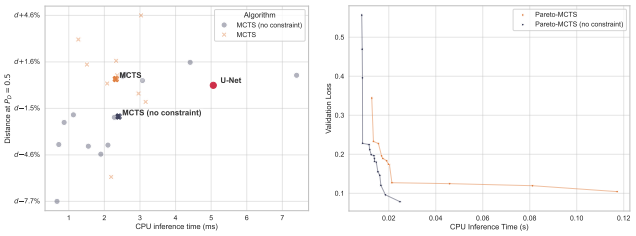
evaluated on a test set consisting of 2500 range-Doppler maps in clutter environments not seen during training. The final CPU inference latency of the searched models is calculated on a Intel® Core™ Ultra 7 165H CPU using ONNX runtime. In future works, we will benchmark the architectures on edge devices with more modest processor characteristics.

Detection performance can only be compared between models at a constant probability of false alarm P_{FA}, which is often a requirement of radar systems. For a model prediction $\hat{y}$, a detection is announced for every pixel $\hat{y}_{i,j}$ which value exceeds a threshold σ. We tune the threshold σ over which a detection is announced on a dataset consisting only of gaussian thermal noise in order to attain the required P_{FA}. Once the P_{FA} is identical for all models, we compare them using the Distance@$P_D = 0.5$ metric, which is the distance at which the detection probability is equal to 0.5.

Table I shows the metrics for some of the architectures found by the search algorithm. It is possible to see that the best models outperform the detection performance of the U-Net, while being as much as 4 times faster in terms of inference latency. The P_D-distance and P_{FA}-distance curve of Model I against the U-Net is shown in Figure 3b and 3c.

A. Ablation study: Constraint on Arithmetic Intensity

In order to validate the usefulness of the arithmetic intensity constraint, we performed a search with the constraint removed, meaning that no penalty is associated to sampling a neural network with low intensity. The unconstrained search converged to a Pareto front that strictly dominates the constrained Pareto front, as it is possible to see in Figure 4b. However, after training the obtained architectures, we observe that the majority of the architectures discovered by the unconstrained search have both an extremely low inference latency on CPU and poor detection performance. As such, these architectures are inferior to the ones found by the constrained search, as Figure 4a shows. We hypothesize that, during the search, these inefficient architectures benefited from the large-scale supernet training through weight inheritance. As these architectures are structurally weak, the same weight values are difficult to obtain using gradient descent during training, yielding a lower performance. These results warrant further investigation on the use of metrics related to arithmetic intensity in neural architecture search.



(a) (b)

Fig. 4. (a) Average D@0.5 against inference CPU time for the architectures found by our method (orange) and by MCTS without applying the constraint on arithmetic intensity (blue). (b) Pareto front obtained by the search with (orange) and without (blue) the arithmetic intensity penalty.

B. Ablation study: Discounted Pareto-MCTS

A key consideration to take into account is the nonstationarity of the weight-sharing NAS problem. Indeed, at the beginning of the search, all operation choices, including those who may reveal to be optimal, appear weak to the agent, as the supernetwork is not trained. As the search progresses, all choices appear gradually better ; it is thus important that the agent keeps exploring while the supernetwork is being trained. To this end, we experimented with modifying the UCB1 formula that is used to select nodes by adding a discounting term that weighs more significantly recent rewards, in a fashion similar to [36]. The score for a node a is then:

$$\text{Score}(a) = \frac{\sum_{k=0}^{N(a)} \gamma^{N(a)-k} R_k}{N(a)} + c\sqrt{\frac{\log t}{N(a)}} \tag{4}$$

where $N(a)$ is the number of times node a has been visited ; R_k is the reward obtained by following a playout from node a at time step k ; t is the total number of time steps so far and $\gamma \in [0, 1]$ is a discounting term.

We compare the search performance of the discounted Pareto-MCTS against the classic Pareto-MCTS algorithm. Figure 5 shows that, while using the discounted UCB formula speeds up the network convergence in the early stages of the

search, the two search algorithms eventually settle on a similar average supernet validation loss. Furthermore, visualizing the architectures produced by both search algorithms (Figures 6a and 6b) shows that the undiscounted version of Pareto-MCTS converged to a slightly better set of architectures than the discounted variant. As such, we decided to keep the search with the vanilla UCB formula.

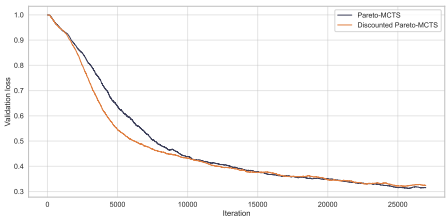
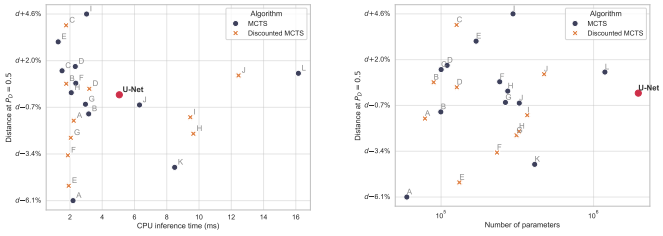


Fig. 5. Evolution of the average accuracy of sampled subnetworks during training



(a) (b)

Fig. 6. (a) CPU inference time against distance @ $P_D = 0.5$ for searched models, colored by search algorithm. (b) Number of trainable parameters against distance @ $P_D = 0.5$.

VI. Conclusion

In this work, we perform multi-objective neural architecture search to design efficient radar target detectors that reach optimal detection performance while minimizing CPU inference latency. We employ arithmetic intensity–constrained latency as a search objective, which proves particularly effective for the considered detection task and warrants further investigation for other problems. The best neural network architectures identified during the search achieve superior detection performance compared to a baseline U-Net in cluttered environments, while achieving a $10\times$ reduction in parameter count and $4\times$ speedup on CPU.

Future works will include benchmarking the automated neural network design approach on other radar detection and segmentation tasks, careful evaluation on real-world clutter environments, and extending the search to low-power processors.

References

[1] D. Cantorna, C. Dafonte, A. Iglesias, and B. Arcay, "Oil spill segmentation in SAR images using convolutional neural networks. A comparative analysis with clustering and logistic regression algorithms," *Applied Soft Computing*, vol. 84, p. 105716, Nov. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494619304971

[2] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-Aware Neural Architecture Search for Mobile," arXiv, Tech. Rep. arXiv:1807.11626, May 2019, arXiv:1807.11626 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1807.11626

[3] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," Nov. 2019, arXiv:1905.02244 [cs]. [Online]. Available: http://arxiv.org/abs/1905.02244

[4] J. Akhtar and K. E. Olsen, "GO-CFAR Trained Neural Network Target Detectors," in *2019 IEEE Radar Conference (RadarConf)*, Apr. 2019, pp. 1–5, iSSN: 2375-5318. [Online]. Available: https://ieeexplore.ieee.org/document/8835765

[5] D. Brodeski, I. Bilik, and R. Giryes, "Deep Radar Detector," in *2019 IEEE Radar Conference (RadarConf)*. Boston, MA, USA: IEEE, Apr. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8835792/

[6] Z. Baird, M. K. McDonald, S. Rajan, and S. Lee, "A Neyman-Pearson Criterion-Based Neural Network Detector for Maritime Radar," in *2021 IEEE 24th International Conference on Information Fusion (FUSION)*. Sun City, South Africa: IEEE, Nov. 2021, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9626944/

[7] C. Wang, J. Tian, J. Cao, and X. Wang, "Deep Learning-Based UAV Detection in Pulse-Doppler Radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–12, 2022.

[8] F. Yavuz, "Radar Target Detection with CNN," in *2021 29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1581–1585.

[9] J. Bai, L. Zheng, S. Li, B. Tan, S. Chen, and L. Huang, "Radar Transformer: An Object Classification Network Based on 4D MMW Imaging Radar," *Sensors*, vol. 21, no. 11, p. 3854, Jun. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/11/3854

[10] A. Ouaknine, A. Newson, P. Pérez, F. Tupin, and J. Rebut, "Multi-View Radar Semantic Segmentation," Aug. 2021, arXiv:2103.16214 [cs]. [Online]. Available: http://arxiv.org/abs/2103.16214

[11] L. Zhang, X. Zhang, Y. Zhang, Y. Guo, Y. Chen, X. Huang, and Z. Ma, "PeakConv: Learning Peak Receptive Field for Radar Semantic Segmentation," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 17 577–17 586, iSSN: 2575-7075. [Online]. Available: https://ieeexplore.ieee.org/document/10204655

[12] T. Li, L. Zhang, Y. Zhang, Z. Hu, P. Pi, Z. Lu, Q. Liao, and Z. Ma, "AdaPKC: PeakConv with Adaptive Peak Receptive Field for Radar Semantic Segmentation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 136 545–136 575, Dec. 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/hash/f6b22ac37beb5da61efd4882082c9ecd-Abstract-Conference.html

[13] A. Ouaknine, A. Newson, J. Rebut, F. Tupin, and P. Pérez, "CARRADA Dataset: Camera and Automotive Radar with Range-Angle-Doppler Annotations," May 2021, arXiv:2005.01456 [cs]. [Online]. Available: http://arxiv.org/abs/2005.01456

[14] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," 2016. [Online]. Available: https://arxiv.org/abs/1611.01578

[15] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," arXiv, Tech. Rep. arXiv:1905.11946, Sep. 2020, arXiv:1905.11946 [cs, stat] type: article. [Online]. Available: http://arxiv.org/abs/1905.11946

[16] A. Shaw, D. Hunter, F. Landola, and S. Sidhu, "SqueezeNAS: Fast Neural Architecture Search for Faster Semantic Segmentation," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 2014–2024. [Online]. Available: https://ieeexplore.ieee.org/document/9022227/

[17] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, and Y. Lin, "HW-NAS-Bench:Hardware-Aware Neural Architecture Search Benchmark," arXiv, Tech. Rep. arXiv:2103.10584, Mar. 2021, arXiv:2103.10584 [cs] type: article. [Online]. Available: http://arxiv.org/abs/2103.10584

[18] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution," arXiv, Tech. Rep. arXiv:1804.09081, Feb. 2019, arXiv:1804.09081 [cs, stat] type: article. [Online]. Available: http://arxiv.org/abs/1804.09081

[19] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm," arXiv, Tech. Rep. arXiv:1810.03522,

Apr. 2019, arXiv:1810.03522 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1810.03522

[20] S. Mao, M. Qin, W. Dong, H. Liu, and Y. Gao, "RAM-NAS: Resource-aware Multiobjective Neural Architecture Search Method for Robot Vision Tasks," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 2712–2719, arXiv:2509.20688 [cs]. [Online]. Available: http://arxiv.org/abs/2509.20688

[21] N. Lallouet, T. Cazenave, C. Enderli, and S. Gourdin, "Searching Efficient Deep Architectures for Radar Target Detection using Monte-Carlo Tree Search," in *2024 International Radar Conference (RADAR)*, Oct. 2024, pp. 1–6, arXiv:2506.21772 [eess]. [Online]. Available: http://arxiv.org/abs/2506.21772

[22] H. Zhu, S. Guo, W. Sheng, and L. Xiao, "SCM: A Searched Convolutional Metaformer for SAR Ship Classification," *Remote Sensing*, vol. 15, no. 11, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/15/11/2904

[23] A. Archet, F. Orieux, N. Ventroux, and N. Gac, "Exploration d'architectures de réseaux de neurones pour la segmentation sémantique d'images aériennes," Aug. 2023.

[24] W. Du, J. Chen, C. Zhang, P. Zhao, H. Wan, Z. Zhou, Y. Cao, Z. Huang, Y. Li, and B. Wu, "SARNas: A Hardware-Aware SAR Target Detection Algorithm via Multiobjective Neural Architecture Search," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–23, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10173551

[25] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[26] N. Lallouet, T. Cazenave, C. Enderli, and S. Gourdin, "Loss Function Design For Training Robust Radar Detectors Using Deep Learning," in *Conference on Artificial Intelligence for Defense*. Rennes, France: DGA Maîtrise de l'Information, Nov. 2023. [Online]. Available: https://hal.science/hal-04328554

[27] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search," arXiv, Tech. Rep. arXiv:1806.09055, Apr. 2019, arXiv:1806.09055 [cs, stat] type: article. [Online]. Available: http://arxiv.org/abs/1806.09055

[28] L. Xie, X. Chen, K. Bi, L. Wei, Y. Xu, Z. Chen, L. Wang, A. Xiao, J. Chang, X. Zhang, and Q. Tian, "Weight-Sharing Neural Architecture Search: A Battle to Shrink the Optimization Gap," Aug. 2020, arXiv:2008.01475 [cs]. [Online]. Available: http://arxiv.org/abs/2008.01475

[29] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search," arXiv, Tech. Rep. arXiv:1812.03443, May 2019, arXiv:1812.03443 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1812.03443

[30] M. Xiang, R. Ding, H. Liu, and X. Zhou, "Latency-Constrained Neural Architecture Search Method for Efficient Model Deployment on RISC-V Devices," *Electronics*, vol. 13, no. 4, Feb. 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/4/692

[31] S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures," Tech. Rep. 1407078, Sep. 2009. [Online]. Available: http://www.osti.gov/servlets/purl/1407078/

[32] A. Nakano, "Theoretical Peak Performance of a computer," 2023. [Online]. Available: https://aiichironakano.github.io/cs596/PeakFlops.pdf

[33] R. Negrinho and G. Gordon, "DeepArchitect: Automatically Designing and Training Deep Architectures," arXiv, Tech. Rep. arXiv:1704.08792, Apr. 2017, arXiv:1704.08792 [cs, stat] type: article. [Online]. Available: http://arxiv.org/abs/1704.08792

[34] L. Wang, Y. Zhao, Y. Jinnai, Y. Tian, and R. Fonseca, "AlphaX: eXploring Neural Architectures with Deep Neural Networks and Monte Carlo Tree Search," arXiv, Tech. Rep. arXiv:1903.11059, Oct. 2019, arXiv:1903.11059 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1903.11059

[35] W. Wang and M. Sebag, "Multi-objective Monte-Carlo Tree Search," 2012.

[36] L. Kocsis and C. Szepesvári, "Discounted UCB," Jan. 2006.