Searching Efficient Deep Architectures for Radar Target Detection using Monte-Carlo Tree Search

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

Stéphanie Gourdin
Thales DMS
Elancourt, France
stephanie.gourdin@fr.thalesgroup.com

***Abstract*—Recent research works establish deep neural networks as high performing tools for radar target detection, especially on challenging environments (presence of clutter or interferences, multi-target scenarii...). However, the usually large computational complexity of these networks is one of the factors preventing them from being widely implemented in embedded radar systems. We propose to investigate novel neural architecture search (NAS) methods, based on Monte-Carlo Tree Search (MCTS), for finding neural networks achieving the required detection performance and striving towards a lower computational complexity. We evaluate the searched architectures on endoclutter radar signals, in order to compare their respective performance metrics and generalization properties. A novel network satisfying the required detection probability while being significantly lighter than the expert-designed baseline is proposed.**

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

I. Introduction

In recent years, artificial neural networks (ANN) applied to the problem of radar target detection have been the subject of keen interest from the research community. The representative power of neural networks as well as their generalization properties establish them as viable candidates to achieve superior performance compared to classical detection tools such as CFAR (Constant False Alarm Rate) detectors, especially on endoclutter environments.

The amount of computational resources available in radar systems is typically limited (e.g. only a CPU) and the need for real-time processing is an significant constraint. However, modern deep convolutional neural network (CNN) architectures are often large and prohibitionally computationally expensive, such as in the work of [1]. One can thus understand that designing efficient neural networks is of the utmost importance.

This paper focuses on the problem of air-air radar target detection. The radar, mounted on an airborne platform, is subject to unwanted signals, such as ground clutter. Targets of interest are typically aerial platforms with a small radar cross-section (RCS). The problem of radar target detection reduces to 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
- $\nu(t)$ is noise (thermal noise, ground clutter, ...)

The matched filter is an optimal solution under the assumption that the target is unique and $\nu(t)$ corresponds to thermal noise only. Space-Time Adaptive Processing further takes into account ground clutter through a correlated noise model, however its performance is limited when the target has low radial velocity, and the single target assumption is still required. An alternative way to treat this problem without using somewhat restrictive signal models is to follow a data-based approach. Here an a priori statistical model with many parameters, in the form of a neural network, is trained to detect multiple targets in various situations. The statistical model can then find an approximate solution to problem 1 by expressing it as a binary image segmentation problem, i.e. predict 1 for a pixel associated to a target signal, and 0 for a pixel associated to thermal noise or clutter.

In recent years, Neural Architecture Search (NAS) has been a popular approach for automatically finding neural architectures with good performance. We propose to investigate NAS for the design of radar target detector architectures, while introducing novel search methods.

Most recent NAS algorithms favour weight-sharing approaches, at the expense of traditional search algorithms such as MCTS (Monte-Carlo Tree Search). However, we motivate our use of Monte-Carlo methods by their remarkable search efficiency when the search space is large. The initial drawback of such approaches, namely the necessity to train an architecture from scratch at each random playout, is mitigated with the rise

of novel training-free metrics, on which we shall expand in Section II.

The contributions of this paper are the following :

- The efficiency of MCTS-based neural architecture search for the problem of radar target detection under network complexity constraints is investigated.
- The GRAVE algorithm and Nested Monte-Carlo Search are evaluated for the first time for NAS.
- Training-free metrics are used for the first time for Monte-Carlo NAS, and their adequacy is validated.
- A neural network that is more frugal than the original baseline while achieving the expected detection performance on endoclutter environments is proposed.

II. Related works

Deep learning has, for the past few years, been the subject of interest from the radar signal processing community. The research work of [2] introduces a radar detector inspired from the Faster R-CNN architecture. The work of [3] introduces a neural network based on a Fully Convolutional Network (FCN) to train a radar detector constrained by the Neyman-Pearson criterion, while [4] proposes a radar detector implementing the U-Net architecture. CNN architectures are also proposed by [5] and [6]. The vast majority of the radar detectors introduced in the literature posseses a very large (e.g. millions) number of parameters. Indeed, it has been shown that deep architectures (with a large number of convolutional layers), while tricky to train, exhibit very good segmentation performances and generalization capacity. However, in a bid to develop hardware-friendly radar detectors, we must try to identify light networks with a smaller number of parameters and detection performance on par, or superior, to their heavier counterparts.

Since the work of [7], NAS has sparked great interest in the research community. NAS aims to find the neural network architecture that minimizes the evaluation loss. This optimization problem can be expressed by Equation 2:

$$a^* = \arg\min_{a \in S} \mathcal{L}_{val}(X, Y, W_a) \tag{2}$$

where:

- a^* is the optimal neural network architecture
- S is the search space of all architectures
- W_a are the weights associated to architecture a
- $\mathcal{L}_{val}(X, Y, W)$ is the loss function computed on an validation dataset.

Even though early approaches often use reinforcement learning to train a controller which generates the architecture, later works, inspired by [8], primarily focus on supernet-based approaches. A *supernet* is an overparametrized network which contains all possible architectures. It is trained once and candidate architectures are sampled from it. An example of a recent supernet-based approach, [9], presents a one-shot approach for finding architectures using a supernet.

A subfield of NAS that is of particular interest is training-free NAS. Indeed, most NAS methods typically exhibit long training times for candidate architectures. Supernet techniques mitigate those long training times by training a very large network only once, but they also suffer from some drawbacks (e.g. deep coupling between architecture parameters and supernetwork weights). Recently, drawing from the neural network pruning literature [10] [11], NAS research efforts investigate training-free metrics in order to score architecture at initialization, without training the candidate architecture. Notable research works tackling this problem include [12], [13] and [14].

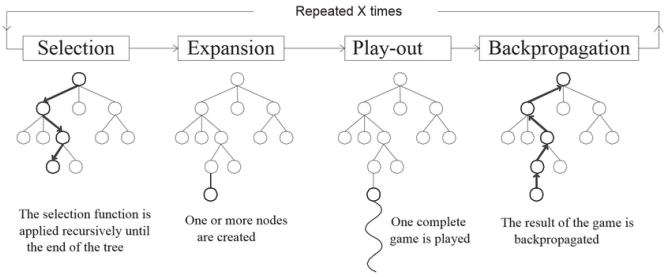

Fig. 1. Monte-Carlo tree search

Neural architecture search has been leveraged for the development of neural radar target detectors. Evolutionary algorithms have been investigated for automotive radar object classification [15] and for SAR image segmentation [16]. Supernet approaches have also been proposed for 3D object detection [17] and for SAR ship classification [18]. These research works demonstrate the interest of NAS for designing deep radar detectors.

Monte-Carlo Tree Search (MCTS) is a popular set of tree exploration techniques. It uses random playouts to estimate the value of a node in the search tree and implements a node selection algorithm for and efficient search. The four steps of the search procedure (selection, expansion, playout, backpropagation) are illustrated in Figure 1 [19]. MCTS has proved to be a reliable method for game playing [20] and numerous other applications [21] [22]. In recent years, some research works have focused on implementing MCTS techniques for neural architecture search. The seminal work of [23] introduces a MCTS agent coupled with a search space definition language for efficient tree traversal. [24] builds on this by proposing a search policy based on RAVE [25]. More recently, [26] introduces a value network, dubbed *Meta-DNN*, that aims to predict the value of a state without training the associated network architecture. Building on these works, we propose to evaluate more search algorithms based on Monte-Carlo Tree Search. To the best of our knowledge, MCTS methods have not been investigated for designing radar detection model architectures.

III. Methodology

A. Search space

One of the key components of NAS is the definition of an adequate architecture search space. In line with numerous research works, we decide to use the NASNet search space [27]. The NASNet search space reduces the search of an

Scenario	Aircraft speed (m/s)	Elevation (ft)
1	250	5000
2	250	1000
3	250	10 000
4	500	5000
5	500	1000
6	500	10 000
7	1000	5000
8	1000	1000
9	1000	10 000
10 (thermal noise)	N/A	N/A

TABLE I
RADAR SIGNAL GENERATION PARAMETERS

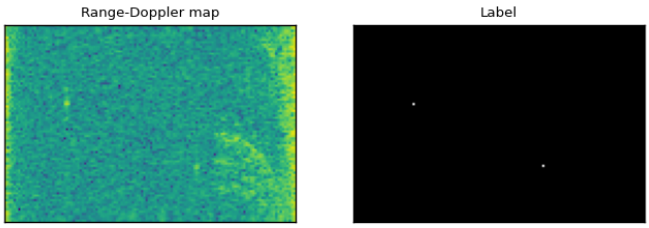

Fig. 2. A training dataset range-Doppler map

architecture to the search of cells, which are then stacked to produce the final architecture. Each cell is composed of N blocks, which have searchable inputs and operations. Even though [27] recommends $N = 5$, we choose to search a single block, for search efficiency purposes. The NASNet search space has been designed for the search of image classification networks, and thus only search *Normal Cells*, which preserve the input dimension, and *Reduction Cells*, which halve the input dimension. We propose the extension of NASNet to a third type of cell, that we name *Upsample Cell*, doubling the input dimension. This enables us to create U-Net-like [28] architectures and perform image segmentation. Furthermore, the searched architectures can be directly compared to the network proposed by [4], which reaches state-of-the-art target detection probability on thermal noise.

B. Data

The data consists of 80 000 range-Doppler maps. A range-Doppler map is simulated by a realistic radar signal generator, and includes ground clutter drawn from 10 different scenarios, which are explicited in Table I. Each map has a number of targets drawn uniformly between 0 and 6. An example of a training range-Doppler map with its associated label is given in Figure 2. As the dimensions of a training example are dependent on the range resolution and the number of FFT points, we decide to resize all images to 128x128 dimension using zero padding. The data is separated into a training and a validation set using a 80%-20% split. The test set is composed of 2000 novel range-Doppler maps that have not been encountered during training. This allows one to evaluate a neural network's generalization capabilities.

C. Algorithms

The most popular MCTS algorithm, UCT (Upper Confidence bound applied to Trees) [29], uses the reward statistics of a node s to select a child node according to the following formula:

$$\text{node} = \arg\max_{i \in C} \mu_i + k\sqrt{\frac{\ln(n_s)}{n_i}}$$

where:

- C are the children nodes of node s
- $\mu_i = \frac{r_i}{n_i}$ is the average reward after child i is selected
- n_s is the number of visits of node s
- n_i is the number of visits of the child node i.
- k is a tunable exploration constant.

UCT has been successfully used in neural architecture search [23], [24]. It is thus a good baseline to which we can compare the following algorithms. A node in the search tree represents an architecture, and a move allows the agent to select an operation, input, combination or network hyperparameter in the NASNet action space. A node is terminal when there are no more available moves, i.e. the architecture is complete and can be trained.

A straightforward improvement of UCT is RAVE (Rapid Action Value Estimation) [20]. RAVE leverages the statistics of a node as well as its AMAF (All Moves As First heuristic) value. The AMAF value corresponds to the reward statistics of a move, regardless of when it has been played during the game. This allows one to gather a larger amount of data for the same number of random playouts. However, RAVE makes the assumptions that the order of played moves doesn't matter, i.e. moves are interchangeable between one another. In the MC-RAVE algorithm, a child node is selected in the following manner:

$$\text{node} = \arg\max_{i \in C}(1-\beta)\mu_i + \beta\tilde{\mu}_i + k\sqrt{\frac{\ln(n_s)}{n_i}}$$

where:

- $\mu_i = \frac{r_i}{n_i}$ is the average reward after child i is selected
- $\tilde{\mu}_i = \frac{\tilde{r}_i}{\tilde{n}_i}$ s the average reward after the move associated with child i is played anytime during the game
- $\beta = \frac{\tilde{n}_i}{n_i + \tilde{n}_i + 4n_i\tilde{n}_i\tilde{b}^2}$, where $\tilde{b}$ is a bias constant.

In the case of neural architecture search, the moves represent the choice of an operation, input, combination or other hyperparameter. It follows that selecting one move before or after another is of no consequence. As such, the RAVE hypothesis holds, and the implementation of this search algorithm becomes possible.

It is possible to bring some improvements to RAVE. One of such improvements, called GRAVE (Generalized Rapid Action Value Estimation) [30], uses the AMAF statistics of a node s's ancestor if the number of visits of s is inferior to a value $tref$. GRAVE has shown to be an improvement over RAVE in several

board games ; this motivates the evaluation of the algorithm on the task of NAS. One of the interesting advantages that GRAVE possesses over RAVE is that GRAVE draws information from a node s's ancestor nodes when the current state s does not have enough playouts to provide reliable estimates. This brings stability to the tree search. In our experiments, the value $tref$ is set to 30 node visits.

Nested Monte-Carlo Search (NMCS) [31] is a different way of exploring a search tree using random playouts. The algorithm selects a move by recursively applying a nested search on lower node levels. Nested Monte-Carlo Search has also shown good performance on various games. To the best of our knowledge, GRAVE and Nested Monte-Carlo Search have not been applied to neural architecture search.

It is important to set an upper bound on the complexity of the network, to avoid the possibility that the search algorithm finds solutions with high performance but unable to ensure real-time detection. A good and relatively hardware-agnostic proxy for network latency is the number of parameters of the neural network. During a MCTS playout, if the search algorithm samples a network with a number of parameters that is superior to a fixed value α, the reward 0 is returned. This enables the algorithm to discount solutions which violate the network complexity bound. In our experiments, α is chosen as the number of parameters of the baseline U-Net.

MCTS needs a way to evaluate the value of a terminal state when performing a random playout. It is common, in NAS applications, to train the sampled network on the training dataset and assign the validation loss to the value of the leaf node corresponding to the architecture. However, training a neural network from scratch is computationally expensive, and quickly becomes intractable when evaluating a large number of states, especially when constrained by hardware or time. Furthermore, the random nature of MCTS entails that the algorithm must perform a large number of playouts before reaching a solution. It is thus clear that MCTS is not the most viable approach if one needs to train every candidate architecture. Motivated by the recent success of training-free approaches, we decide to use the metric proposed by [13] for scoring candidate architectures at initialization. When the tree search reaches a terminal node, the reward obtained by the associated architecture is computed in the following manner:

$$\text{score} = \log|K_H| \tag{3}$$

$$K_H = \begin{pmatrix} N_A - d_H(c_1, c_1) & \dots & N_A - d_H(c1, c_n) \\ \dots & & \dots \\ N_A - d_H(c_n, c_1) & \dots & N_A - d_H(cn, c_n) \end{pmatrix}$$

where N_A is the number of ReLU activations in the network, and $d_H(c_a, c_b)$ is the Hamming distance between the binary codes of training examples a and b computed in the ReLU activations. In short, this metric captures the correlations between two inputs in a minibatch of data, and scores highly a network that, at initialization, is able to differentiate the two inputs. We shall refer the reader to [13] for a detailed explanation of the metric.

Search algorithm	N_{param}	**Test loss**	$P_{FA}(\times 10^{-4})$
Baseline (U-Net)	120441	0.57	0.30
Random search	100041	0.88	2.01
UCT	71336	0.67	0.4
RAVE	63916	0.70	1.02
GRAVE	87016	0.67	0.70
NMCS	**48209**	**0.54**	**0.29**

TABLE II
ARCHITECTURE COMPARISON

Our MCTS methods implement leaf parallelization, as introduced by [32]. Here, instead of performing a single random playout after the expansion phase, we run 8 parallel playouts starting from the same node. This does not accelerate the search but provides improved stability as the value estimates for a node are much more reliable.

Finally, we implement a basic random search policy, which randomly samples architectures for a number of iterations k and returns the one with the highest score. We allow 25 minutes for all MCTS algorithms to search for a move and set the number of random search iterations k as the number of move explorations during this time. At the end of the procedure, the best performing architecture for all algorithms is returned and trained for 3 hours on a NVIDIA A4000 GPU.

IV. RESULTS

We shall compare the neural networks returned by each algorithm with the two following metrics : detection probability P_D at a fixed false alarm probability P_{FA}, and network complexity, through the number of parameters N_{param} of the architecture. We use the following proxies for P_D and P_{FA} :

$$\tilde{P_D} = \frac{\sum_i^N \hat{y}_i y_i}{\sum_i^N y_i}$$

$$\tilde{P_{FA}} = \frac{\sum_i^N \hat{y}_i (1 - y_i)}{\sum_i^N (1 - y_i)}$$

where $\hat{y}_i$ are the predicted pixels and y_i are ground truth pixels. Table II displays the network complexity of the architectures returned by each search algorithm.

The detection probabilities associated with these architectures, evaluated on the test set described in Section III, can be appreciated in Figure 3. Table II shows that the architecture chosen by Nested Monte-Carlo search slightly outperforms the baseline U-Net, both in detection probability and false alarm probability, while having 60% fewer parameters than the U-Net. We can see from Table II that, with the exception of the architecture returned by the Nested Monte-Carlo Search algorithms, all models have a false alarm probability P_{FA} higher than the baseline U-Net. This indicates that the scoring metric used during the search is possibly ill-suited to evaluating the false alarm probability at initialization. It can be noticed that random search failed to propose a competitive architecture for the detection problem, but we believe that, with a longer

search time, a reasonably well-performing architecture could be found, as [27] shows that random search on NASNet is a good baseline. The superiority of GRAVE over RAVE is consistent with the conclusions of [30], and indicates that GRAVE provides better node value estimates during the search, especially when the search time is short.

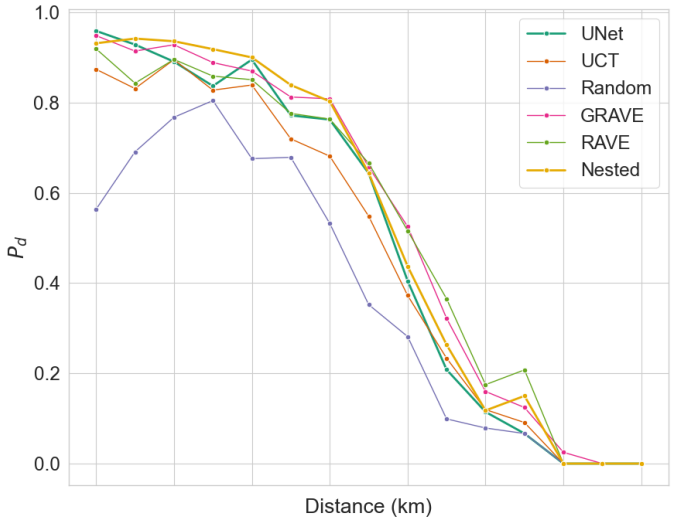

Fig. 3. Detection probability P_D

The normal, reduction and upsample cells returned by the most effective search algorithm, Nested Monte-Carlo Search, are shown in Figure 4.

V. Discussion

Among all models produced by the different search algorithms, we find that the model yielded by Nested Monte-Carlo Search performs the best on the test set. Interestingly, this model is not the one with the largest number of parameters. Intuitively, it may be claimed that there exists a trade-off between network complexity and performance ; however, we show that, on the task of radar target detection, light architectures are able to perform as well as more complex ones, such as the baseline U-Net, provided that the cell design be sound.

In addition to cell design, we also attempted to add more broad architecture hyperparameters, such as the number of convolutional blocks or the initial number of channels, in a bid to search a more expressive space. The algorithms proved efficient for finding good architectures with these added hyperparameters in an unconstrained space, but failed

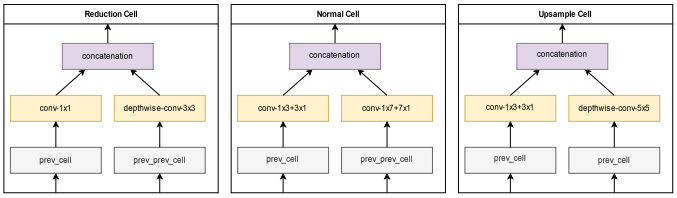

Fig. 4. Best performing architecture cells

to produce high-performing networks with the severe network complexity constraint introduced in Section III. Indeed, the MCTS-based algorithms fell in local minima, represented by hyperparameters choices associated to overly shallow networks, because of the negative reward obtained when exceeding the parameter constraint. More generally, this tendency to undershoot the number of parameters is encountered with most search algorithms, with the exception of random search, which is a notoriously hard baseline to beat on the NASNet search space. Avoiding these pitfalls is the object of current research, as we believe that Monte-Carlo based methods are able, with proper problem design, to assimilate the constraint as to return strong networks in the extended search space.
The limitations of our findings are the following :

- In order to validate the generalization performances of the searched network, evaluating them on additional scenarios with various differing clutter profiles is necessary.
- Additionally, the training time of the networks can be extended for optimal convergence. However, the scope of this work is not necessarily achieving the best possible performance via training hyperparameter tuning, but rather identifying efficient architectures that can be considered instead of the handmade baseline neural network.

VI. Conclusion and Future Works

In this paper, we propose a novel architecture for radar target detection using deep learning. We present innovative training-free Neural Architecture Search (NAS) methods, based on Monte-Carlo Tree Search algorithms. The best architecture found by these methods exhibits comparable detection performance to the current state of the art on endoclutter environments, while possessing fewer parameters (40% of the parameters of the baseline model).

Our findings validate the interest of Monte-Carlo methods for the design of neural radar detectors, but they also open the way for a large number of improvements. Future research will include investigations on higher-level Nested Monte-Carlo search, additional training-free metrics (including metrics adapted to P_{FA} estimation) and improvement through self-play for finding the best fitting architectures. We will also focus our attention on ways to satisfy a network complexity constraint without falling into local minima.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[2] 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/

[3] 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/

[4] 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

[5] 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.

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

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

[8] 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

[9] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single Path One-Shot Neural Architecture Search with Uniform Sampling," arXiv, Tech. Rep. arXiv:1904.00420, Jul. 2020, arXiv:1904.00420 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1904.00420

[10] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," arXiv, Tech. Rep. arXiv:2006.05467, Nov. 2020, arXiv:2006.05467 [cond-mat, q-bio, stat] type: article. [Online]. Available: http://arxiv.org/abs/2006.05467

[11] C. Wang, G. Zhang, and R. Grosse, "Picking Winning Tickets Before Training by Preserving Gradient Flow," arXiv, Tech. Rep. arXiv:2002.07376, Aug. 2020, arXiv:2002.07376 [cs, stat] type: article. [Online]. Available: http://arxiv.org/abs/2002.07376

[12] M. S. Abdelfattah, A. Mehrotra, Dudziak, and N. D. Lane, "Zero-Cost Proxies for Lightweight NAS," arXiv, Tech. Rep. arXiv:2101.08134, Mar. 2021, arXiv:2101.08134 [cs] type: article. [Online]. Available: http://arxiv.org/abs/2101.08134

[13] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural Architecture Search without Training," arXiv, Tech. Rep. arXiv:2006.04647, Jun. 2021, arXiv:2006.04647 [cs, stat] type: article. [Online]. Available: http://arxiv.org/abs/2006.04647

[14] Y. Shu, S. Cai, Z. Dai, B. C. Ooi, and B. K. H. Low, "NASI: Label- and Data-agnostic Neural Architecture Search at Initialization," arXiv, Tech. Rep. arXiv:2109.00817, Apr. 2022, arXiv:2109.00817 [cs] type: article. [Online]. Available: http://arxiv.org/abs/2109.00817

[15] A.-E. Cozma, L. Morgan, M. Stolz, D. Stoeckel, and K. Rambach, "DeepHybrid: Deep Learning on Automotive Radar Spectra and Reflections for Object Classification," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 2682–2687.

[16] 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.

[17] O. T.-C. Chen, Y.-X. Chang, Y.-W. Jhao, C.-Y. Chung, Y.-L. Chang, and W.-H. Huang, "3D Object Detection of Cars and Pedestrians by Deep Neural Networks from Unit-Sharing One-Shot NAS," in *2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2022, pp. 1–8.

[18] 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

[19] J. Mańdziuk, "MCTS/UCT in solving real-life problems," in *Studies in Computational Intelligence*, Jan. 2018, pp. 277–292.

[20] S. Gelly, Y. Wang, and O. Teytaud, "Modification of UCT with Patterns in Monte-Carlo Go," Jan. 2006.

[21] G. Best, O. Cliff, T. Patten, R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, p. 027836491875592, Mar. 2018.

[22] A. Rimmel, F. Teytaud, and T. Cazenave, "Optimization of the Nested Monte-Carlo Algorithm on the Traveling Salesman Problem with Time Windows," in *Evostar*, Turin, Italy, Apr. 2011. [Online]. Available: https://inria.hal.science/inria-00563668

[23] 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

[24] M. Wistuba, "Practical Deep Learning Architecture Optimization," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, Oct. 2018. [Online]. Available: http://dx.doi.org/10.1109/DSAA.2018.00037

[25] S. Gelly and D. Silver, "Monte-Carlo tree search and rapid action value estimation in computer Go," *Artificial Intelligence*, vol. 175, pp. 1856–1875, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000437021100052X

[26] 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

[27] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," arXiv, Tech. Rep. arXiv:1707.07012, Apr. 2018, arXiv:1707.07012 [cs, stat] type: article. [Online]. Available: http://arxiv.org/abs/1707.07012

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

[29] L. Kocsis and C. Szepesvári, "Bandit Based Monte-Carlo Planning," in *Machine Learning: ECML 2006*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293.

[30] T. Cazenave, "Generalized Rapid Action Value Estimation," in *24th International Conference on Artificial Intelligence*, Buenos Aires, Argentina, Jul. 2015, pp. 754–760. [Online]. Available: https://hal.science/hal-01436522

[31] ——, "Nested Monte-Carlo Search," 2009.

[32] T. Cazenave and N. Jouandeau, "On the Parallelization of UCT," in *Computer Games Workshop*, Amsterdam, Netherlands, Jun. 2007. [Online]. Available: https://hal.science/hal-02310186