A Framework for Exploring and Modelling Neural Architecture Search Methods

Pavlo Radiuk[1] [0000-0003-3609-112X] and Nadiia Hrypynska[2] [0000-0003-0103-976X]

[1,2] Khmelnytskyi National University, 11, Instytuts'ka str., Khmelnytskyi, 29016, Ukraine

[1]radiukpavlo@gmail.com, [2]grypynska@gmail.com

Abstract. For the past years, many researchers and engineers have been developing and optimising deep neural networks (DNN). The process of neural architecture design and tuning its hyperparameters remains monotonous, time-consuming, and do not always ensure optimal results. In his regard, the automatic design of machine learning (AutoML) has been widely utilised, and neural architecture search (NAS) has been actively developing in recent years. Despite meaningful advances in the field of NAS, a unified, systematic approach to explore and compare search methods has not been established yet. In this paper, we aim to close this knowledge gap by summarising search decisions and strategies and propose a schematic framework. It applies quantitative and qualitative metrics for prototyping, comparing, and benchmarking the NAS methods. Moreover, our framework enables categorising critical areas to search for better neural architectures.

Keywords: deep neural network, AutoML, neural architecture search, scheme modelling, efficient neural network.

1 Introduction

Nowadays, NAS studies focus on enhancing the viability of DNNs in three ways: increasing network accuracy, decreasing computational costs, and reducing network weight. For instance, some applications require high-level precision from neural networks [1]. For real-time applications, hardware requirements can be vital [2]. Some applied systems depend on swift yet straightforward methods [3]. According to [4,5], among the various optimisation methods, NAS shows the most noticeable outcomes. In general, the goal of NAS can be defined as a construction of an optimal architecture limited by certain conditions, iteratively searching for better architectures that satisfy those conditions. Considering the significant benefits of automation, AutoML has been rapidly developing in recent years. Unfortunately, the scientific community has not standardised a comprehensive approach to explore, compare and select efficient NAS methods. Consequently, we aim to suggest a new framework to promote this field of research.

We consider an N-dimensional architecture space to construct an optimal neural network. The search space covers the scope of all possible combinations of neural architectures. Every structure represents an N-tuple, where N stands for the number of parameters of the architecture search space. To search for a capable topology that meets all the constraints of a given problem, we must assign a set of cost conditions to those constraints. Restricted search narrows the search space and focuses on low-cost architectures.

2 Related Work

Several studies in the NAS area are based on either manual or automatic searching techniques. As stated in [6], a manual search is to modify the neural elements of DNNs by imposing sparse constraints on the target function during training. Another work [7] proposes to replace 3×3 convolutional blocks with individual building blocks. Such blocks embody a combination of 3×3 and 1×1 convolutional operators with a reduction in the number of input channels on each convolutional block. Furthermore, the authors of this work achieved increasing classification accuracy by implementing manual down-sampling. In [8], authors presented a universal cascade decoder, which is embedded into the network by sequential adding of each convolutional building block. Li et al. [9] combined dense skip connections as specific micro-blocks from a 2D DenseUNet with a 3D equivalent to hierarchically accumulate volumetric features so that their network can maximise the use of spatial information during the classification of 3D images. Dolz [10] enhanced the previous approach by applying an operator level to the building blocks of convolutional neural networks (CNN). Their HyperDense-Net utilises dense skip connections between convolutional layers in each building block. In [11], researchers proposed a hierarchical group convolution operation that can effectively compress the deep neural model. In [12], a hierarchical DNN consists of multiple CNNs on each network layer. The proposed network grows in the tree form with new data classes on each to identify previously trained ones. Kokiopoulou [13] presented a gradient-based approach to design effective DNN architectures. In this work, authors applied the modification called subnetwork level to decrease the inference time. The idea is that the proposed structure splits the DNN into several subnetworks, where each subnetwork specialises and runs only on a separate set of input classes.

Despite successful examples of manual methods, they are still highly dependent on the input conditions of a given task. The use of manual search requires comprehensive expertise and time-consuming work. To address these issues, researchers have further developed advanced methods to automate the design process of DNN [14]. In this study, the parameterised underlying architecture is utilised to search for a suitable architecture for every dataset separately. In [15], authors independently trained an ensemble of small building blocks with a homogeneous averaged output. In [16], researchers proposed a modified reinforcement learning method which selects different neural building blocks to construct DNNs. Guo [17] demonstrates how the inverse reinforcement learning approach can search for efficient network structures topologically inspired by manual search.

In order to resolve computational-cost issues, Yan et al. [18] presented a capable NAS based on the parameter sharing approach with a hierarchical search space. In [19], a novel hierarchical evolution representation scheme simulates the modularised design template commonly utilised by human experts. In [20], researchers suggested an automatic designing strategy that empowers particle vectors to encode building blocks of DNN easily. Chu [21] utilised a fixed hierarchical macro-architecture as a search space. In [22], authors applied a transfer learning approach to automatic NAS and proposed a generic search space, including fixed macro-architecture and hyperparameter choices. Zeng [23] implemented various sampling procedures into an automatic selection method which significantly reduced the search time for the architecture. Recently, researchers started to implement a directed acyclic graph (DAG) as a search space, as in [24], to store dozens of subgraphs, each of which denotes the type of sample architecture. In [25,26], authors suggested an evolutionary NAS. In their work, search space comprises a diverse set of hyperparameters, activation functions, and normalisation layers. Parkashi [27] applied multi-label classification into the cascade neural network for efficient hyperparameter tuning.

3 Evaluation Criteria

The main problem of NAS methods in terms of evaluation is their limitation by the performance of optimised architecture. In other words, the better a produced architecture performs (measured by quantitative criteria), the better the corresponding NAS technique is. The most used quantitative metrics [5,6,14,28] to measure search methods are described below.

- **Training and validation loss**. Training loss signals whether the network is improving in the search procedure and if so, shows the speed of improvement. The validation loss checks if the model is overfitting or underfitting. The goal is to measure how fast the output network converges and how well it will perform on new datasets.
- **The number of parameters**. This criterion records the number of parameters the output model contains. The more parameters and weights a model comprises, the heavier it is. The idea behind it is to estimate and compare the amount of physical memory the models require.
- **Amount of training data**. In the real world, training data could be a precious resource, depending on the expense of data collection and labelling. This criterion aims to assess whether the search method can find an optimal architecture on a limited dataset.
- **Training and test CPU time**. This metric records the total CPUs and GPUs time used to run all training and test epochs. The goal is to measure how much system resources the search method might consume.
- **Memory requirement**. The metric is vital when an output architecture is intended to be used in limited performance devices such as smartphones, digital cameras, or

quadcopters. The goal is to check whether a model is suitable for restricted systems.

Despite the simplicity and practicality of quantitative indicators, they do have significant disadvantages. The efficiency of neural modals crafted by either manual or automatic search techniques may vary significantly due to the influence of stochastic processes of a particular domain [29]. Currently, used metrics focus mostly on the performance of a network. Such criteria do not account the required level of expertise in a subject area and the level of effort needed to apply the search method to another issue or dataset. To this context, the evaluation of variance on performance might be considered a proper criterion to select a suitable candidate architecture. Another essential aspect is the level of manual changes to the search procedure. Even automatic search methods may depend on human interventions, although their principal purpose is to reduce its level. Besides, the reproducibility is a common pitfall in NAS. Therefore, within our framework, we consider the variance on performance and the level of human intervention as qualitative criteria. Below we describe a five-step evaluation strategy that must be followed successively.

- **Search space design**. Standard manual search spaces usually reduce the efficiency of the further search strategy. The goal here is to check the suggested heuristic search procedure for non-standard search placements.
- **Requirements**. Predetermine the signs of optimality of the underlying architecture. The purpose of this step is to reduce the likelihood of obtaining suboptimal architecture upon the completion of training.
- **Search process**. Like the type of defined search space, this step differentiates the model's ultimate performance from basic training strategies. The goal is to check whether a predetermined underlying architecture can be trained in a standard way or whether a specific training strategy should be employed.
- **Adaptation to changes**. The search procedure can be repeated either from scratch or using prior knowledge by applying additional methods. The aim is to consider in advance the computational cost of any human interventions.
- **Code availability**. NAS methods are especially challenging to replicate. Their effectiveness depends on the implementation details. The goal is to open source a programming code and ensure its reproducibility so that the exploring NAS method can be proved on other machines and datasets.

4 Algorithm

This paper proposes a solution that promotes exploring, comparing and estimating search methods, both quantitatively and qualitatively. Our approach is dedicated to identifying the strengths and weaknesses of search methods and suggest improvements that can be implemented to them. In total, the framework consists of four stages, plus the evaluation block. Below we briefly describe the algorithm, the consistent execution of which allows efficient modelling and assessing of any search method.

1. **Setup stage**. In this step, we define the search space and initialise the underlying architecture.
2. **Framing stage**. This step is devoted to measuring the computational and memory cost and configuring the architecture search techniques.
3. **Adaptation stage**. In the adaptation stage, we accommodate the various architectural improvements based on the decisions made in the previous step.
4. **Assessment stage**. Here we assess the performance of training strategy and evaluate output architectures. If the resulting model satisfies the initial conditions or expectations, then the search stops. Otherwise, adaptation techniques must be reused.

The third and fourth stages form a cycle during which the studied architecture is improved iteratively. Fig. 1 illustrates the above algorithm.

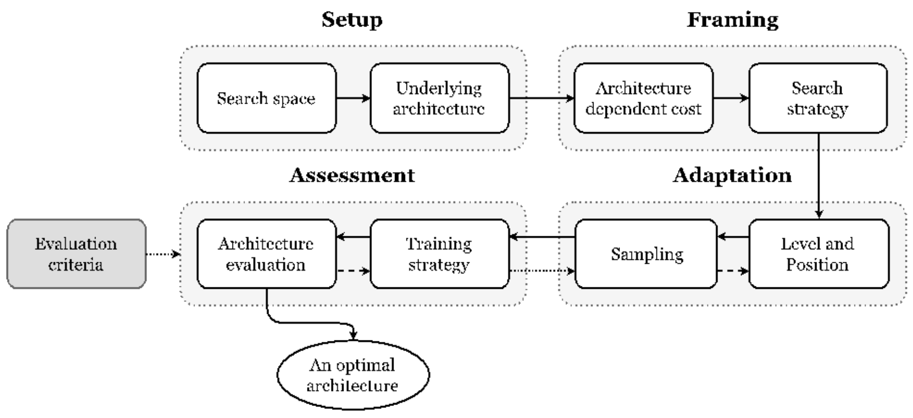

Fig. 1. The scheme of the framework

In the following sections, we provide more detailed instructions based on the analysis of the literature on how each step can be applied to real-world modelling.

5 Setup Stage

In this section, we describe the first stage of the framework. Setup stage specifies the search environment and comprises the initialisation of search space and the definition of underlying architecture. The metrics used in this stage are **the amount of training data**, **memory requirement**, **search space design** and **requirements**.

5.1 The Definition of Search space

The architecture search space depends primarily on the intended use since different datasets impose different constraints on the neural architecture. Also, the search space is affected by existing computational resources. To find an optimal architecture, we need to compromise between the network flexibility, target coverage and computational requirements during the search space.

It should be mentioned that automatic search requires a precise definition of the search space. Concurrently, such requirement is not crucial for handmade framing. In Fig. 2, we categorised search spaces into four groups.

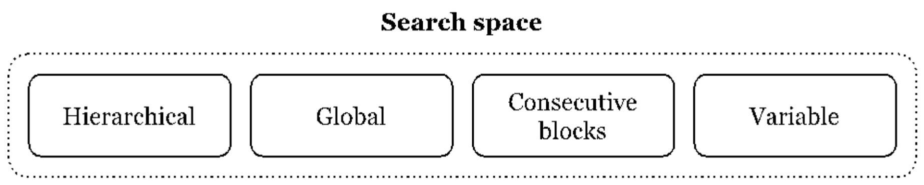

Fig. 2. Types of search space

Hierarchical search spaces consist of two search levels: micro- and macro-architecture search spaces. The macro-layer explains the architecture combination of micro-layer to create the whole network. Sometimes, hierarchical search space [11,12,18,19] can contain fixed micro- [9,24] and macro-architectures [21,22], as in, to provide the managed search. Global search space means that the architecture search is performed among the entire scope of the underlying network. Moreover, as described in [6,7,17], the search scope, in this case, is not limited by any architecture requirements. Search spaces with consecutive blocks apply similar blocks at different network levels. For instance, in [8,10,15,20], the final networks comprise repeatedly attached building blocks and subnetworks. According to [13,17,23,25-27], when input requirements are unknown in advance, a good practice is to utilise variable search spaces.

The Kendall coefficient τ [30] is a commonly utilised metric to calculate the correlation between two rankings. According to [28], the τ indicator is well suited for estimating the search space and can be calculated as

$$\tau = \frac{S_c - S_t}{\frac{1}{2}n(n-1)}, \tag{1}$$

where S_c and S_d stand for the number of concordant and discordant pairs, respectively, n is the number of elements that define search space. The Kendall coefficient is determined in $[-1;1]$, where -1 corresponds to a perfect negative correlation and 1 to a perfect positive correlation. If $\tau = 0$, the space rankings are completely independent. Thus, an ideal NAS method has

$$\tau = \left\lfloor \begin{array}{l} 1; \\ -1. \end{array} \right.$$

5.2 The Initialisation of Underlying Architecture

In this step, it is necessary to initialise the underlying architecture that fits into the previously defined search space. This architecture further serves as a starting point for experiments in both manual and automatic search techniques. Fig. 3 shows types of underlying architectures.

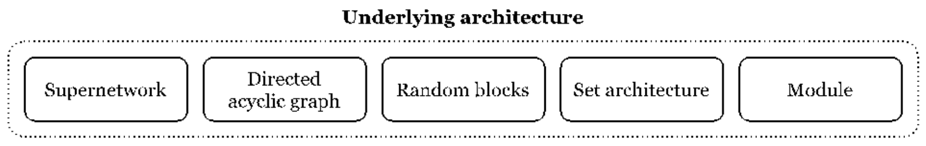

Fig. 3. Approaches for specifying underlying architecture

Different types of underlying architectures serve for various applications. In [12,18,21], all possible topologies organise a superposition called supernetwork. In DAG, each subgraph is assigned as a convolutional block or down-sampling operation [24]. Genetic and reinforced-based algorithms imply the underlying architecture as a randomising of the network blocks [20,23,25]. Sometimes, researches utilise set state-of-the-art architectures, for example, DenseNet [9,10,16], ResNet [17,19], or [26], as an underlying one and after that apply modifications to search for an optimal solution. In cases [6,7,11,13,22], the convolutional module serves as the underlying architecture, and the originating module, gradually expanding, performs the search for an optimal architecture.

6 Framing Stage

In this section, we determine a way to measure the cost of the explored architecture and select a search strategy. The criteria employed here are **the amount of training data**, **training and test CPU time**, **memory requirement** and **search process**.

6.1 The Establishment of Architecture Dependent Costs

Architecture cost designates the requirements which optimal architecture must fit. Having analysed the literature mentioned in the previous sections, we identified a set of cost terms that every NAS method encounters. We illustrate the defined costs and appropriate ways to eliminate them in Fig. 4.

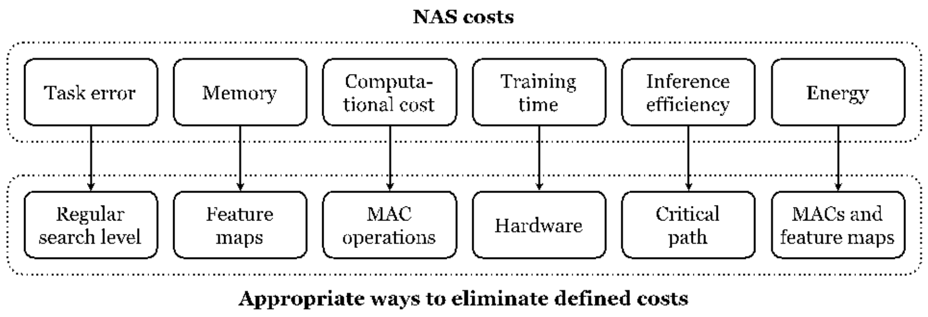

Fig. 4. NAS costs and corresponding ways to reduce them

Determining the expanses is necessary to select a search strategy in the following steps. Also, well-defined cost metrics allow comparing different architectures to choose the best one.

6.2 The Selection of Search Strategy

Here, based on the defined search space, we chose an appropriate search strategy to seek an optimal architecture. As stated in [28], the search space established from the set of all probable architectures is tremendously vast, sometimes endless. Therefore, the task of scanning all the topologies within such search space becomes practically impossible to solve. Besides, the architecture cost impacts the search strategy and is used to compare different architectures in the search space. Fig. 5 presents the most utilised NAS strategies.

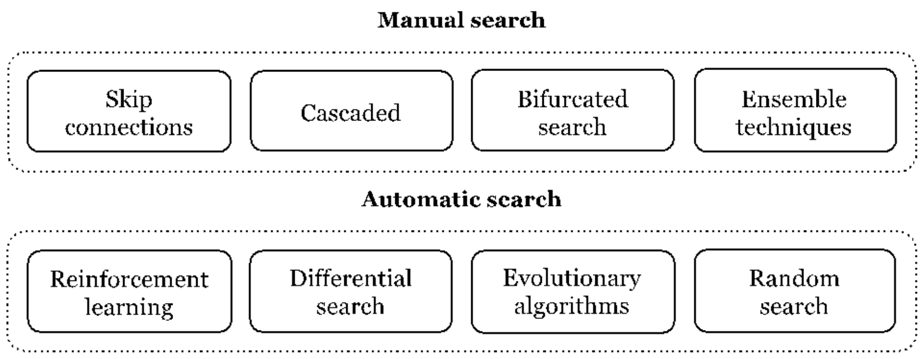

Fig. 5. Manual and automatic search strategies

Manual search serves as a guide or set of rules for designing architectures. An example rule can be implying additional architectural components such as cascades [8,27] or sequences [15]. One suggestion for manual search is a dense architecture by typing bandwidths or searching for compact architectures [9-11]. Some manual techniques depend on searching for bifurcated structures to enhance feature maps and weight sharing [6,7,12]. In other cases, researchers search for an ensemble of networks and calculate its average output to obtain the desired result [13,15,23].

In contrast to manual search, automated search methods are more advanced approaches. Reinforced-based search strategy assigns the reward by reducing the overall cost of architecture, which in turn leads to a better network configuration [16,17]. Random [18,19,22] and evolutionary [20,25,26,27] search strategies achieve significant results at a low cost compared to other search strategies. Nowadays, modern state-of-the-art technique – differentiable architecture search (DART) – uses gradient descent approaches to minimise the overall cost function of architecture [21,24].

7 Adaptation Stage

In this stage, we apply various improvements to exploring architecture. This procedure comprises establishing level, position and sampling techniques. The metrics utilised here are **the number of parameters** and **adaptation to changes**.

7.1 The Selection of Level and Position Modifications

Network design can be refined based on two criteria: the level and position of modifications. The level commonly corresponds to operator level [6,11,16,26], building block level [7,9,15,18], subnetwork level [18], cell-based level [22,27] or their connection between each other [10-12,16,19,21,24]. Some studies are based on global position [19,21], others use layer-based [8,10,17]. Different upgradings can also be applied as group-based [6,11,12] or hybrid positions [9]. Many methods apply pooling position and its varieties as an additional layer [10,12,17,20,21,25]. Fig. 6 depicts common instances of adaptation procedure.

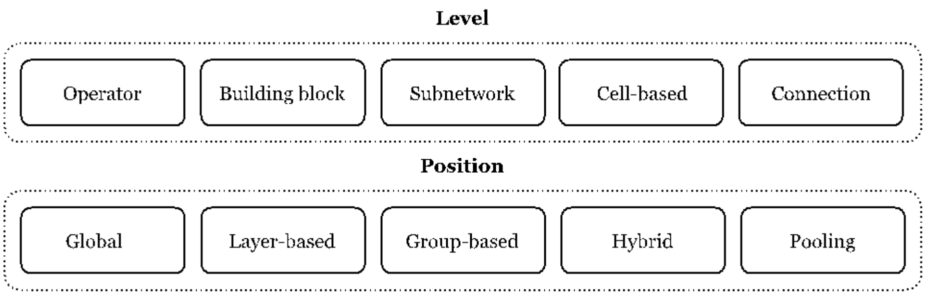

Fig. 6. Modification techniques commonly used in NAS

Manual search methods involve various enhancements through the lens of intuition and experience. Automated search methods instead define the level and position by the parameterised architecture of the search space. Choosing the right level and position depends on the computational constraints and the input datasets.

7.2 The Application of Sampling Techniques

In this step, we tune the architecture parameters. Implemented modifications result in the distribution of the architecture configuration. Such distribution allows obtaining a combination of architectures that best meet the requirements defined for the intended application. The sampling methods can differ from each other and usually dependent on specific search spaces and search strategies. In general, sampling techniques are the varieties of pruning. They allow removing redundant or unnecessary blocks from an architecture. Fig. 7 shows the instances of the sampling methods.

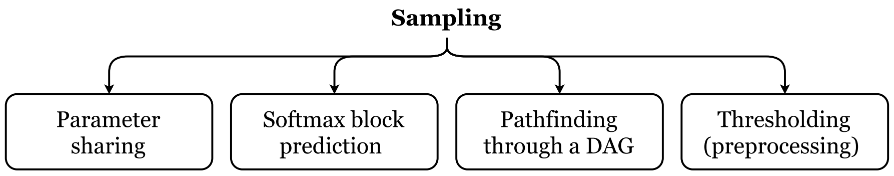

Fig. 7. Sampling techniques

In [22,24], researchers apply a parameter sharing technique, while [21] employs SoftMax block prediction to improve network robustness. Works [18,24] implemented pathfinding through a DAG to reduce computational time cost. Thresholding or pre-processing a network is a common technique to comprise all sparse architectures within the search space into one efficient neural network [18,21]. Manual methods do not contain parameterised architecture distribution that must be determined. For that reason, sampling procedures are not applied to a manual search but are widely used in an automated one.

8 Assessment Stage

In the last step, we train and evaluate the sampled architecture. Researchers should consider **the training and validation loss**, **the number of parameters**, **training and test CPU time**, **search process** and **code availability**. Fig. 8 illustrates the assessment stage.

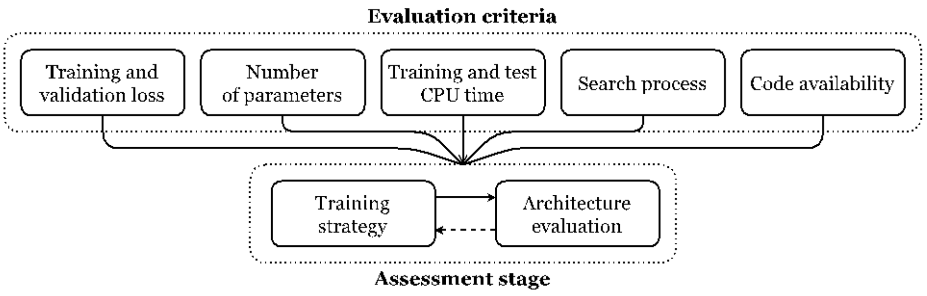

Fig. 8. The procedure of the evaluation of a search method

Practitioners can apply various training strategies, depending on the initial conditions of the task. For example, progressive training means the training of only the modified part of the network, while the rest of the network parameters are remaining fixed [23]. While performing a bifurcated exercise, one trains various subnetworks, connected to a shared subnetwork, in parallel [15,21]. Also, a practical approach is to combine different training strategies presenting them as training steps [5,19]. When the training finishes, the explored architecture is being evaluated on the validation set. If the found architecture satisfies the initial metrics, the search process stops, and the architecture is accepted as an expected result. Otherwise, it is needed to return to the previous step and implement additional improvements to find an architecture that satisfies optimisation criteria.

9 Framework Application

In this section, we demonstrate an attempt to summarise the NAS procedure using the proposed framework. In total, upon the suggested measures, we analysed and evaluated state-of-the-art search methods developed in recent years. Table 1 and Table 2 presents the results of the analysis.

Table 1. Analysis of NAS methods outlined by the proposed framework (part 1)

Study	NAS approach	Search space	Underlying architecture	Search strategy
[6]	Manual	Global	Module	Bifurcated search
[7]	Manual	Global	Module	Bifurcated search
[8]	Manual	Consecutive blocks	Set	Cascade
[9]	Manual	Hierarchical micro	Set	Skip connections
[10]	Manual	Consecutive blocks	Set	Skip connections
[11]	Manual	Hierarchical	Module	Skip connections
[12]	Manual	Hierarchical	Supernetwork	Bifurcated search
[13]	Manual	Variable	Module	Ensemble
[16]	Automatic	Global	Set	Reinforcement learning
[17]	Automatic	Variable	Set	Reinforcement learning
[18]	Automatic	Hierarchical	Supernetwork	Random search
[19]	Automatic	Hierarchical	Set	Random search
[21]	Automatic	Hierarchical macro	Supernetwork	Differential
[22]	Automatic	Hierarchical macro	Module	Random search
[24]	Automatic	Hierarchical micro	DAG	Differential
[26]	Automatic	Variable	Set	Evolutionary
[27]	Automatic	Variable	Set	Cascade, Evolutionary

Table 2. Analysis of NAS methods outlined by the proposed framework (part 2)

Study	Cost	Adaptation techniques	Sampling techniques	Code availability
[6]	Task error	Operator level, Group-based position	–	Not available
[7]	Task error	Building block level	–	Available and replicable
[8]	Task error	Layer-based position	–	Not available
[9]	Task error	Building block level Hybrid position	–	Available and replicable
[10]	Task error	Connection level, Layer-based, Pooling	–	Available and replicable
[11]	Computational cost, Memory	Operator level, Connection level, Group-based position	–	Not available
[12]	Task error, Inference efficiency	Connection level, Group-based position, Pooling	–	Not available
[13]	Task error, Computational cost	–	–	Available, not replicable
[16]	Memory, Energy	Operator level	Pathfinding	Not available
[17]	Task error, Training time	Layer-based position, Pooling	Pathfinding	Available and replicable
[18]	Task error	Subnetwork	Thresholding, Pathfinding	Not available
[21]	Task error	Connection level, Global position, Pooling	SoftMax, Thresholding	Available and replicable
[22]	Computational cost	Cell-based level	Parameter sharing	Not available
[24]	Task error, Training time	Connection level, Pooling	Parameter sharing, Pathfinding	Available and replicable
[26]	Inference efficiency, Energy	Operator level	–	Not available
[27]	Task error, Inference efficiency	Cell-based level	Pathfinding	Not available

The framework allows combining and analysing various processes that are occurring during the design of neural networks. According to Table 1, search spaces are usually represented as hierarchical structures. Set of handcrafted recognised networks remains the popular type of underlying architecture. Bifurcated and cascaded search strategies are widely prevalent among the manual NAS methods. In contrast, automatic search methods are mostly based on reinforcement and evolutionary strategies.

From table 2, we can see that the primary goal of most search methods remains task error reduction. Individual building blocks and divorce levels are commonly utilised in the adaptation stage. Among sampling techniques, pathfinding and parameter sharing are in favour. The number of works with open source code is approximately equal to the number of works without code.

10 Conclusion

Recently, there has been a variety of methods and technologies for searching optimal neural network architectures. Nevertheless, the issue of summarising and evaluating different procedures and approaches in NAS remains open. In this paper, we propose a new framework for exploring and modelling new NAS techniques. Our method is based on the standard quantitative metrics that measure both individual architectures and search methods. Within the proposed framework, we introduce five qualitative assessments to ensure the variance on performance and the level of human intervention. Our work describes all the critical aspects of state-of-the-art search methods and summarises known approaches. The proposed framework imparts a better understanding of recent and noticeable contributions made in the development of NAS methods. In future research, we plan to expand our framework by covering image and text classification tasks.

References

1. Ghavami, N., Hu, Y., Gibson, E., Bonmati, E., Emberton, M., Moore, C.M., Barratt, D.C.: Automatic segmentation of prostate MRI using convolutional neural networks: Investigating the impact of network architecture on the accuracy of volume measurement and MRI-ultrasound registration. Med. Image Anal. 58, (2019). doi:10.1016/j.media.2019.101558
2. Feng, X., Jiang, Y., Yang, X., Du, M., Li, X.: Computer vision algorithms and hardware implementations: A survey. Integration. Elsevier 69, 309–320 (2019). doi:10.1016/j.vlsi.2019.07.005
3. Park, D.S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E.D., Le, Q. V.: SpecAugment: A simple data augmentation method for automatic speech recognition. Proc. Interspeech 2019, 2613-2617 (2019). doi:10.21437/Interspeech.2019-2680
4. Bashar, A.: Survey on evolving deep learning neural network architectures. J. Artif. Intell. Capsul. Networks. 1, 73–82 (2019). doi:10.36548/jaicn.2019.2.003
5. Drozdal, J., Weisz, J., Wang, D., Dass, G., Yao, B., Zhao, C., Muller, M., Ju, L., Su, H.: Trust in AutoML: Exploring information needs for establishing trust in automated machine learning systems. (2020). doi:10.1145/3377325.3377501

6. Smithson, S.C., Yang, G., Gross, W.J., Meyer, B.H.: Neural networks designing neural networks: Multi-objective hyper-parameter optimisation. In: 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pp. 1–8 (2016). doi: 10.1145/2966986.2967058
7. Zoph, B., Vasudevan, V., Shlens, J., Le, Q. V: Learning transferable architectures for scalable image recognition. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8697–8710 (2018). doi:10.1109/CVPR.2018.00907
8. Liang, P., Chen, J., Zheng, H., Yang, L., Zhang, Y., Chen, D.Z.: Cascade decoder: A universal decoding method for biomedical image segmentation. In: 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019). pp. 339–342 (2019). doi: 10.1109/ISBI.2019.8759430
9. Li, X., Chen, H., Qi, X., Dou, Q., Fu, C.-W.W., Heng, P.-A.A.: H-DenseUNet: Hybrid densely connected UNet for liver and tumor segmentation from CT volumes. IEEE Trans. Med. Imaging. 37, 2663–2674 (2018). doi:10.1109/TMI.2018.2845918
10. Dolz, J., Gopinath, K., Yuan, J., Lombaert, H., Desrosiers, C., Ben Ayed, I.: HyperDense-Net: A hyper-densely connected CNN for multi-modal image segmentation. IEEE Trans. Med. Imaging. 38(5), 1116–1126 (2019). doi:10.1109/TMI.2018.2878669
11. Xie, X., Zhou, Y., Kung, S.-Y.: HGC: Hierarchical group convolution for highly efficient neural network. arXiv preprint arXiv:1906.03657 (2019)
12. Roy, D., Panda, P., Roy, K.: Tree-CNN: A hierarchical deep convolutional neural network for incremental learning. Neural Networks 121, 148–160 (2018). doi:10.1016/j.neunet.2019.09.010
13. Kokiopoulou, E., Hauth, A., Sbaiz, L., Gesmundo, A., Bartok, G., Berent, J.: Fast task-aware architecture inference. arXiv preprint arXiv:1902.05781 (2019)
14. Mendoza, H., Klein, A., Feurer, M., Springenberg, J.T., Hutter, F.: Towards automatically-tuned neural networks. In: Hutter, F., Kotthoff, L., and Vanschoren, J. (eds.) Proceedings of the Workshop on Automatic Machine Learning. pp. 58–65. PMLR, New York, New York, USA (2016). doi:10.1007/978-3-030-05318-5_7
15. Macko, V., Weill, C., Mazzawi, H., Gonzalvo, J.: Improving neural architecture search image classifiers via ensemble learning. arXiv preprint arXiv:1903.06236 (2019)
16. Zhong, G., Jiao, W., Gao, W., Huang, K.: Automatic design of deep networks with neural blocks. Cognit. Comput. 12, 1–12 (2020). doi:10.1007/s12559-019-09677-5
17. Guo, M., Zhong, Z., Wu, W., Lin, D., Yan, J.: IRLAS: Inverse reinforcement learning for architecture search. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9013–9021 (2019). doi: 10.1109/CVPR.2019.00923
18. Yan, S., Fang, B., Zhang, F., Zheng, Y., Zeng, X., Xu, H., Zhang, M.: HM-NAS: Efficient neural architecture search via hierarchical masking. CoRR. abs/1909.0, (2019)
19. Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K.: Hierarchical representations for efficient architecture search. In: International Conference on Learning Representations (ICLR). (2017)
20. Wang, B., Sun, Y., Xue, B., Zhang, M.: Evolving deep convolutional neural networks by variable-length particle swarm optimisation for image classification. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8 (2018). doi: 10.1109/CEC.2018.8477735
21. Chu, X., Zhou, T., Zhang, B., Li, J.: Fair DARTS: Eliminating unfair advantages in differentiable architecture search. arXiv preprint arXiv:1911.12126 (2019)
22. Wong, C., Houlsby, N., Lu, Y., Gesmundo, A.: Transfer learning with neural AutoML. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 8366–8375. Curran Associates Inc., Red Hook, NY, USA (2018)

23. Zeng, X., Luo, G.: Progressive sampling-based Bayesian optimisation for efficient and automatic machine learning model selection. Heal. Inf. Sci. Syst. 5, 2 (2017). doi:10.1007/s13755-017-0023-z
24. Dong, X., Yang, Y.: Searching for a robust neural architecture in four GPU hours. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1761–1770 (2019). doi: 10.1109/CVPR.2019.00186
25. Van Wyk, G.J., Bosman, A.S.: Evolutionary neural architecture search for image restoration. Proc. Int. Jt. Conf. Neural Networks. 2019-July (2018). doi:10.1109/IJCNN.2019.8852417
26. Suzuki, T., Takeshita, S., Ono, S.: Adversarial example generation using evolutionary multi-objective optimisation. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 2136–2144 (2019). doi:10.1109/CEC.2019.8790123
27. Pakrashi, A., Mac Namee, B.: CascadeML: An automatic neural network architecture evolution and training algorithm for multi-label classification. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 11927 LNAI, 3–17 (2019). doi:10.1007/978-3-030-34885-4_1
28. Yu, K., Sciuto, C., Jaggi, M., Musat, C., Salzmann, M.: Evaluating the search phase of neural architecture search. arXiv preprint arXiv:1902.08142 (2019)
29. Belli, M.R., Conti, M., Crippa, P., Turchetti, C.: Artificial neural networks as approximators of stochastic processes. Neural Networks. 12, 647–658 (1999). doi: 10.1016/S0893-6080(99)00017-9
30. Kendall, M.G.: A new measure of rank correlation. Biometrika. 30, 81–93 (1938). doi: 10.2307/2332226