IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 February 2023, accepted 5 March 2023, date of publication 8 March 2023, date of current version 16 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3253818

SURVEY

Neural Architecture Search Benchmarks: Insights and Survey

KRISHNA TEJA CHITTY-VENKATA[1], MURALI EMANI[2], VENKATRAM VISHWANATH[2], AND ARUN K. SOMANI[1], (Life Fellow, IEEE)

[1]Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011, USA
[2]Argonne National Laboratory, Lemont, IL 60439, USA

Corresponding author: Krishna Teja Chitty-Venkata (krishnat@iastate.edu)

This work was supported in part by the Philip and Virginia Sproul Professorship; in part by the National Science Foundation at Iowa State University under Grant MRI 1726447 and Grant MRI 2018594; and in part by the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DEAC02-06CH11357.

ABSTRACT Neural Architecture Search (NAS), a promising and fast-moving research field, aims to automate the architectural design of Deep Neural Networks (DNNs) to achieve better performance on the given task and dataset. NAS methods have been very successful in discovering efficient models for various Computer Vision, Natural Language Processing, etc. The major obstacles to the advancement of NAS techniques are the demand for large computation resources and fair evaluation of various search methods. The differences in training pipeline and setting make it challenging to compare the efficiency of two NAS algorithms. A large number of NAS Benchmarks to simulate the architecture evaluation in seconds have been released over the last few years to ease the computation burden of training neural networks and can aid in the unbiased assessment of different search methods. This paper provides an extensive review of several publicly available NAS Benchmarks in the literature. We provide technical details and a deeper understanding of each benchmark and point out future directions.

INDEX TERMS Neural architecture search benchmarks, convolutional neural networks, hardware-aware neural architecture search, hyperparameter optimization, RNN, LSTM.

I. INTRODUCTION

Deep Neural Networks (DNNs) have progressed at a rapid pace over the past decade and deliver State-of-the-Art (SOTA) performance in many tasks. Convolutional Neural Networks (CNNs) [1], [2] are employed in many Computer Vision applications due to their highest level of correctness, while Recurrent Neural Networks (RNNs) and Transformers are predominantly used for language-based tasks.

Neural Architecture Search (NAS) is an efficient and rapidly evolving area which automates the design of neural networks for a given dataset and task. NAS refers to using a search algorithm to find the best-performing architecture from a predefined search space while being more accurate than the hand-crafted models. Over the past few years, several search strategies have been developed, such as Reinforcement Learning [3], One-shot/Differentiable [4], Evolutionary [5], Once-for-all [6], Random search [7], Distillation search [8], Bayesian Optimization [9], Low/zero cost proxy search [10]. This ensures finding more robust models while significantly reducing the human effort spent on tedious neural architecture design and tuning. Hardware-Aware Neural Architecture Search (HW-NAS) aims to search for a model such that the searched topology not only attains desirable accuracy but also efficiently executes on the target hardware.

The major challenges in developing efficient NAS algorithms are **(i)** reproducibility problem, due to variation in search space and experimental methodology, and hence comparison between methods is a very big issue, and **(ii)** the computation complexity of NAS is high as it requires training and evaluation of several networks on a large search space and dataset, and hence NAS is inaccessible to researchers who have limited computing resources. Therefore, in this context, several Neural Architecture Search Benchmarks

The associate editor coordinating the review of this manuscript and approving it for publication was Yizhang Jiang.

(NAS-Bench) such as NAS-Bench-101 [11], NAS-Bench-201 [12], and NAS-Bench-NLP [13], etc., have been developed over the last few years to alleviate the evaluation cost problem and improve reproducibility.

A typical NAS benchmark consists of precomputed evaluation metrics, such as the validation accuracy, FLOPs, number of parameters, and latency on the hardware of all the neural architectures present in the predefined search space. These benchmarks are built through exhaustive evaluation of enumerating and training every possible architecture from the search space. During the search process, the metrics of a model in the search space are queried from the benchmark to assess the quality (i.e., good or bad), allowing NAS researchers to focus solely on the design of efficient search algorithms without investing time and computing power on model training. On the other hand, these benchmarks also provide metadata for training within the predefined search space, as it is crucial for the NAS methods to adopt a common training and testing procedure for reproducibility, rapid prototyping, and comparison. We review several SOTA Neural Architecture Search Benchmarks and future directions to explore this fast-evolving area. In our paper, we refer to the benchmark dataset as the actual NAS benchmark, which provides evaluation metrics of several architectures in the search space, such as NAS-Bench-101 [11], while the term dataset is used for the actual dataset of images/text to train the neural network, such as CIFAR-10 [14] or ImageNet [15].

A. COMPARISON WITH OTHER PAPERS

In the past, several AutoML, NAS, and HW-NAS survey papers have been published. These papers focus on various aspects of search space, search algorithms, evaluation metrics and do not extensively discuss NAS benchmarks in detail. The existing surveys on Neural Architecture Search are outlined in Table 1.

TABLE 1. Summary of NAS surveys.

NAS Survey Type	References
AutoML/AutoDL	[16]–[21]
NAS Classification & Theory	[22]–[33]
NAS Method Evaluation	[34]
Hardware-aware NAS	[35]–[39]
Federated Learning NAS	[40], [41]
Transformer/BERT/ViT NAS	[42]
Generative Adversarial NAS	[43], [44]
Graph Neural Network NAS	[45], [46]
Quantum NAS	[47]

B. CONTRIBUTIONS

Our work is the first dedicated survey paper targeting Neural Architecture Search Benchmarks. The main contributions of this paper are as follows:

1) We investigate the necessity, types and challenges in a NAS benchmark design.
2) We compare several publicly available NAS Benchmarks and provide their technical details in terms of structure, datasets, applications and hardware platforms for the benefit of the research community. We also identify the limitations of current benchmarks.
3) We compile GitHub repositories of all the NAS benchmarks discussed in this paper and provide references for easy access for researchers needing to use them.
4) We create a GitHub repository of all NAS Benchmarks for quick access. This repository can be found at https://github.com/krishnateja95/Neural_Architecture_Search_Benchmark_Collection. The repository will be updated as new benchmarks are developed in the future.
5) Using the outcome of the above studies, we then propose guidelines and provide best practices to build a useful NAS Benchmark. The result of our study in the growing field will assist the research community in developing a better next generation of NAS benchmarks.

C. SCOPE OF THE PAPER

Our paper is focused on the broad area of NAS Benchmarks (till January 2023), where we study different search spaces, challenges, functionality, the scope of each such benchmark, and future steps. We assume that the readers are aware of basic CNN, RNN operations and terminologies such as Convolution, Depthwise Convolution, Max/Avg Pooling, learning rate, and filter/channel size.

D. PAPER ORGANIZATION

Section II describes the common search spaces used in NAS algorithms, followed by an overview of search techniques in Section III. Section IV introduces the necessity, challenges, attributes and recommendations of NAS Benchmarks. Section V presents the description of each NAS benchmark for Computer Vision (CV) applications, followed by non-vision tasks in Section VI. The HW-NAS Benchmarks are discussed in Section VII, while Section VIII describes NAS-HPO Benchmarks. In Section IX, we discuss the limitations of current benchmarks and provide future directions, while Section X concludes this paper by giving a high-level contribution of this paper. Figure 1 provides the classification of different NAS Benchmarks we review in this paper.

II. SEARCH SPACE

A NAS method typically consists of the following three components: **(1)** Search space, a manually-built primitive operation set and structure, whether cell-wise or layer-wise, **(2)** Search Strategy or the core search algorithm through which a network is searched on the search space, **(3)** Evaluation phase, where a predicted network is evaluated for its performance. The first step in a NAS algorithm is to choose the search space consisting of all possible neural architectures from which the NAS method finds a suitable model. The design of the search space neural for a network model is combinatorial, and the complexity of the search algorithm

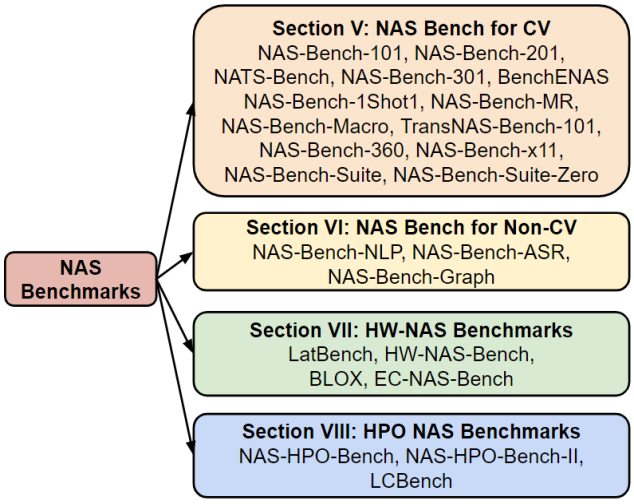

FIGURE 1. Classification of NAS Benchmarks.

increases with an increase in the number of operations. It typically consists of a manually designed set of operations such as Standard Convolution, Depthwise Convolution, etc. The search space can be divided into two types based on the construction of different operations within a single layer, as follows: **(i)** Micro/Cell-level, **(ii)** Macro/layer-wise.

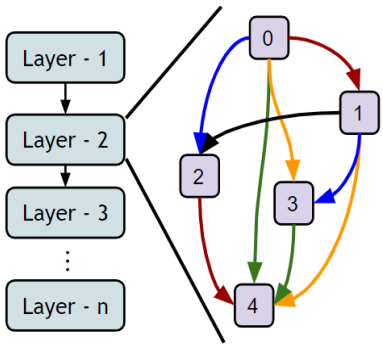

FIGURE 2. Cell-level Search Space.

A. MICRO/CELL SEARCH

The Micro search defines a cell-based structure that is arranged in the form of a Directed Acyclic Graph (DAG), as shown in Figure 2. Each edge in the graph represents an operation to be searched in the search process. The neural network is formed by stacking the searched DAG multiple times, thereby reusing the same architecture throughout the model. The operation on each edge is searched from the set of predefined search elements. There exist two types of cells based on the dimensions of input and output feature maps: a "Normal Cell" does not change the height and width of feature throughout the cell, and a "Reduction Cell" downsamples the feature map, i.e., the resolution of the output is halved compared the input activation. NASNet [3] and DARTS [4] are two examples of cell-based search spaces.

B. LAYER-WISE/MACRO SEARCH SPACE

The same cell structure throughout the network can have a varying effect on the validation accuracy and implementation on the hardware. The branched structure in the cell is not hardware-friendly due to the fragmented architecture, resulting in high inference time. This limitation is resolved in the layer-wise search space by searching different configurations at each layer of the model to attain efficient networks. MobileNetV2 [48], Facebook-Berkely-Net (FBnet) [49], and its variants are commonly used as the macroarchitecture in the layer-wise search space. These macroarchitectures typically consist of the following modules: **(1)** a 3×3 Convolution as head, **(2)** a series of MBConv/FBNet modules, **(3)** average pooling and FC layer as the tail.

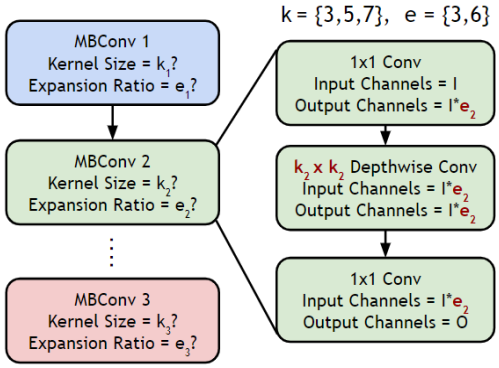

FIGURE 3. Layer-wise/Macro Search Space.

Figure 3 illustrates an example of the layer-wise MBConv/FBNet search, where the input and output filter sizes in the unit are kept constant while the filter expansion (e) and kernel (k) sizes of the module are searched. The search space in ProxylessNAS [50] can be generically represented as MBConv_e_k, where e indicates expansion ratio ($e \in \{3,6\}$) and k stands for kernel size ($y \in \{3,5,7\}$). Wu et al. [49] introduced FB-Network, a macroarchitecture similar to MobileNetV2, where the MBConv block is replaced with the FBNet block. The 1×1 Pointwise Convolution is replaced with the 1×1 Group Convolution. The search space is represented as FBNet_e_k_g, where e and k indicate the expansion ratio and kernel size, respectively, and g is for group size.

III. NEURAL ARCHITECTURE SEARCH STRATEGIES AND PERFORMANCE EVALUATION

Automated Machine Learning (AutoML) is a promising method to automate the design pipeline of developing a high-quality ML algorithm for a given task and resources. The technique significantly reduces human effort and enables non-ML experts to build models for their applications without worrying about ML and statistical knowledge. Hyperparameter Optimization (HPO) is a method to adjust the hyperparameter values during training, often the most difficult process which accounts for the success of Deep Learning. The approaches for HPO include manual tuning (based on

ML expertise), trial-and-error, random search, or grid search. Neural Architecture Search [51], [52], [53] is a process of automating the architecture optimization of DNNs, such as searching for optimal operations or parameters at each level of the network. Hardware-aware NAS (HW-NAS) methods aim to search for a neural architecture that is accurate on the given dataset and hardware-efficient, such as latency or energy, compared to the manually designed networks. The hardware performance metrics are incorporated into the NAS algorithm to guide the hardware-aware search process. This section classifies and summarizes several NAS methods. In this section, we provide a brief overview of several NAS algorithms. The best approach depends on the requirements and the resources available, as each of these methods has its own strengths and weaknesses. See White et al. [32] for an in-depth latest survey on NAS.

A. REINFORCEMENT LEARNING-NAS (RL-NAS)

The pioneering NAS algorithm proposed by Zoph and Le [3] is based on Reinforcement Learning (RL) principle to search for efficient neural architectures. The algorithm consists of an RNN model as a controller that interacts with the environment of all possible neural architectures. In each iteration, the controller predicts the architecture that is likely to generate good accuracy, and the predicted model is trained end-to-end to obtain its performance. The validation accuracy of the trained models is used as a reward/feedback signal to update the RNN such that it predicts the next architecture that can lead to higher accuracy. In RL terms, this iterative methodology allows the algorithm to move from one state (a given architecture) to another (a potentially better architecture) in which the accuracy increases. See the survey paper of Jaafra et al. [54] for detailed explanation on Reinforcement Learning-based NAS method.

B. ONE-SHOT/DIFFERENTIABLE/GRADIENT NAS

One-shot NAS methods reduce the computational burden by encoding all possible neural architectures in the form of a Supernetwork. This method leverages weight sharing notion, thereby reusing the same weights for multiple architecture combinations. This allows the search algorithm to conduct training and evaluation in a single network rather than training and evaluating individual models separately. DARTS [4] is one of the pathbreaking one-shot NAS algorithms that formulates the search process in a differentiable manner by initializing a learnable architectural parameter (α) for every operation in the search space. The Supernetwork is trained using gradient descent, and the final architecture is formed by sampling the operation at each level with the highest α parameter value. Although the One-shot method significantly reduces the computational burden, it suffers from a lack of interpretability and is sensitive to initialization. The differentiable search can sometimes be difficult to interpret in terms of the design principles. The Supernetwork can be sensitive to parameter initialization, and the optimization may be trapped in local minima. See the recent survey papers on Weight Sharing [28], SuperNetwork [29] and Gradient Descent [30] for more insights

C. EVOLUTIONARY NEURAL ARCHITECTURE SEARCH

The Evolutionary Learning search algorithms [5] use principles of natural evolution, such as selection and mutation, to search for good solutions to a problem. The Genetic Algorithm in Evolutionary Learning is an iterative process of evaluating selected individuals according to a fitness function and generating a new set of architectures using the characteristics of best-performing models from the previous generation. Initially, a population is randomly generated by sampling different architectures from a large pool of networks in the search space. Each individual is a specific neural architecture, which is trained on the target task to determine fitness. The weaker networks have less chance of surviving in the current generation as it competes with candidates of a higher fitness function. The next generation of top k networks is obtained by mutation or crossover of top individual models in the current generation of networks. Although this search process is very effective, it requires a large amount of computation time and resources. See the survey papers of Liu et al. [55] and Zhou et al. [33] for a detailed explanation of Evolutionary Learning-based NAS.

D. ONCE-FOR-ALL SEARCH

Once-For-All (OFA) [6] is a two-step method that combines One-shot Supernetwork training and Evolutionary search. OFA algorithm first trains an over-parameterized network of maximum dimensions along each of the hyperparameters, i.e., kernel and filter size. During the Evolutionary search process, the predicted networks are sampled directly from the Supernetwork and are evaluated to capture the validation metrics without finetuning. The main advantage of this method is avoiding retraining of the sampled networks as weights are directly retained from the Supernetwork. Several methods rely on the Once-for-all methodology by first training a large CNN/Transformer Supernetwork and applying Evolutionary search to find the optimal subnetwork.

E. RANDOM SEARCH

Random NAS [7] is a type of search algorithm in which the search space of possible architectures is explored randomly instead of a guided search process. In this method, a set of possible neural network architectures are defined, and the algorithm generates new models by randomly sampling the operations at each level in the network. The performance of each sampled architecture is evaluated on the target dataset, and the best-performing models are selected for further evaluation. Random NAS is simple, easy to implement, and does not require a complex learning and optimization pipeline. Nonetheless, it can be less effective than other well-defined NAS methods, such as Reinforcement or Evolutionary, as they do not establish relationships between

different neural architectures. Therefore, the random search method can sometimes require a large number of iterations to find an optimal solution.

F. ZERO-COST/LOW-COST PROXY-BASED NAS

The NAS methods with zero or low-cost proxies do not require training any candidate model during the search process [10]. It relies on a set of performance estimation techniques to quickly evaluate the performance of the chosen models. During the search process, a performance predictor is used to evaluate the performance of the candidate model without actually training them. For example, Zero-cost proxy [56] devises a method to score a neural architecture at initialization which is indicative of its final trained accuracy using just a single minibatch. Although these low-cost estimation strategies significantly reduce the compute burden, they may not always produce efficient models as compared to the traditional training-based methods, as the performance estimator may not truly reflect the performance of the trained models.

G. BAYESIAN OPTIMIZATION

Bayesian Optimization (BO) [9] uses statistical methods which construct a surrogate model that maps the hyperparameters to the performance of the model. It defines a probabilistic Bayesian inference model of the search space and uses this model to guide the search process in finding efficient architectures. The search algorithm generates new architectures in the search space region where the statistical model predicts good architectures that are likely to be effective. This process of constructing the model and finding the optimal hyperparameters is repeated iteratively until the optimal values are found, or the iteration reaches a predetermined stopping criterion. A benefit of the Bayesian technique is that it can efficiently search, even when it is large and the function is noisy or non-smooth.

H. KNOWLEDGE DISTILLATION NAS

Knowledge Distillation [8] is a process of transferring knowledge from a large complex model to a smaller and more efficient network. The student model is trained to minimize the difference between its predictions and the predictions from the teacher model for the same input minibatch. Distillation [57] is used to accelerate the NAS process by training the student models to mimic the behavior of the teacher model, which is a set of architectures that have been previously trained, allowing NAS algorithms to quickly evaluate the performance of a large number of candidate architectures.

I. HARDWARE-AWARE NAS (HW-NAS)

The core principle in the HW-NAS method [35] is to include the hardware performance metrics of the target hardware in the reward/loss/optimization/fitness functions in one of the NAS algorithms described previously. For a given model M, let Acc(M) and Lat(M) represent the accuracy and hardware latency of a predicted model, and "T" denote the target latency of the searched model. The multi-objective function can be written as $\text{Acc(M)} * \left(\frac{Lat(M)}{T}\right)^{\gamma}$. See HW-NAS survey papers [35], [36], [37], [38], [39] that describe NAS methods on different hardware platforms.

J. PERFORMANCE EVALUATION

As we observed in all the search strategies, the evaluation phase is the most critical step, which evaluates the performance of a predicted architecture. It compares different networks generated by search algorithms to guide the search algorithm to find optimal models. NAS Benchmarks play a pivotal role in curtailing the cost of this expensive performance estimation, which is typically done by full or partial training of the predicting network. Xie et al. [34] summarized several efficient evaluation methods for NAS.

IV. INTRODUCTION TO NAS BENCHMARKS

NAS Benchmarks are critical to aid the design process of DNNs. There are several reasons why benchmarks play a pivotal role in today's fast-evolving NAS field. This section provides an overview of NAS Benchmarks, the necessity of using these benchmarks, challenges, types, and guidelines to create such benchmarks.

A. NECESSITY OF NAS BENCHMARKS

(1) A NAS algorithm samples the top-performing architectures from the pool of networks and evaluates the performance metric of the sampled network, which is used to guide the search algorithm to find the next best architecture. The validation accuracy, typically obtained by training the network, is expensive on a large dataset. Therefore, the main limitation of NAS, in general, is the availability of computational resources and time for training, making it difficult to conduct experiments for researchers who do not have access to large-scale training systems. The benchmark dataset provides the ground-truth model performance metrics of all the candidate models in the search space either through a precomputed dictionary or by predicting using a surrogate model. This allows NAS methods to obtain a selected model's accuracy in a smaller amount of time by querying the dataset dictionary instead of fully training the architecture during the search process.

(2) NAS researchers can solely focus on quickly validating the effectiveness of their search algorithm in terms of relative strengths and weaknesses by avoiding expensive training. Therefore, the NAS Benchmarks greatly relieve the computational burden for researchers who do not have access to enough computing environments to accelerate these networks. The benchmarks are useful for comparing the performance of different search strategies and for understanding the trade-offs between different neural architectures.

(3) Although NAS methods tend to consistently improve to increase efficiency, the training setup, such as the learning rate and weight initialization, are different. A few studies [58] have shown that seed design plays a key role in NAS

methods, and yet many NAS papers do not discuss the role of seed and related observations. The NAS Benchmarks enable a fair comparison between NAS methods by providing standard training and evaluation protocols, datasets, and metrics.

(4) The inconsistency in the training of selected architecture could induce bias in the NAS algorithm to choose the optimal model. The discrepancies include different data augmentation, regularization, learning rate, batch size, etc. The improper dataset division for inference of trained models also leads to a difference in validation/testing accuracy. Hence, NAS Benchmarks provide an unbiased accuracy of the chosen architecture during the search process.

(5) NAS benchmarks can also be utilized to develop surrogate models to predict the accuracies of unseen networks and unknown search spaces. For example, the input of the surrogate model could be an untrained architecture in the benchmark and the output being the predicted accuracy. The researchers can use the predicted outputs of the surrogate model to identify promising architectures for further study on how they can be improved.

(6) The goal of HW-NAS Benchmarks is to democratize HW-aware architecture search research to non-hardware practitioners and make HW-NAS research reproducible and easily accessible [12]. This involves careful measurement of hardware performance metrics such as latency or energy of the network in the search space on the real or simulated hardware platform. Therefore, non-hardware experts can obtain the latency of a layer or entire model by querying the benchmark dataset. In a few cases where the implementation of Convolution/Recurrent/FC layers on hardware is semi-optimized or sub-optimal, the inappropriate measurement in latency leads to a bad search of hardware-efficient networks. Thus, the HW-NAS benchmarks can facilitate a fair comparison of different HW-NAS methods and searched models. Also, these benchmarks can be used to predict the latencies of unseen neural architectures.

B. TYPES OF NAS BENCHMARKS

There exist two different types of NAS Benchmarks based on the process of querying/estimating the performance metrics of the neural architecture. The two types of NAS Benchmarks are (1) Tabular NAS Benchmark and (2) Surrogate Benchmark. A detailed explanation of both kinds of benchmarks is detailed below:

1) TABULAR BENCHMARK

Tabular benchmark is the first kind and the most widely constructed benchmark dataset. This database is built by enumerating all possible neural architectures in the predefined search space and training every model end-to-end to capture the related performance metrics. The benchmark returns the table entry when a model's performance is queried. The process of creating this type of benchmark is expensive computationally, as it requires training every architecture from scratch. Siems et al. [59] motivated the issue with tabular benchmarks by discussing the stochasticity in these benchmarks. The authors claim that the randomness in mini-batch training and its appearance in the performance of a model makes the architecture a random variable. Consequently, the tabular dataset contains the results of only a few draws as the NAS benchmarks train the network not more than three times. The presence of such bias in the evaluation formulates the tabular benchmark into a simple estimator of performance metrics based on the previous evaluations only. Thus, we can assume that better estimators exist, outperforming the tabular benchmarks. An unintended outcome of NAS tabular benchmarks is that the existing ones are based on a relatively narrow architectural space, as building a benchmark on a large search space requires exhaustive training of all models. This leads to undesirable results when transferred to large and contrasting search spaces. Also, the tabular benchmarks require large storage space, and the size increases with an increase in the search space.

2) SURROGATE BENCHMARKS

The Surrogate Benchmarks estimate the performance of a neural architecture through an auxiliary ML model, thereby mimicking the functionality of the tabular benchmark. The surrogate models are trained on the data generated by the prior training of several architectures. Therefore, the validation accuracy of the untrained and unseen models can be predicted using the secondary ML models much faster, thus avoiding the need to store the entire tabular data. Although these models only approximate the real benchmark, if curated well, the surrogate models can perform better than the tabular benchmarks, which was shown by some previous works [59], [60]. A few examples of surrogate models are regression, LGBoost, XGBoost, Graph Isomorphism Network (GIN), NGBoost, μ-Support Vector Regression (SVR), Random Forests (RF), etc. The surrogate benchmarks can be particularly useful when searching for architectures on large datasets where the tabular benchmarks can be computationally expensive. Once-for-all [6] method employs an MLP network in the Evolutionary search process to predict the accuracy and latency of the network on the ImageNet dataset. There is often a risk factor associated with the surrogate benchmarks when there is a significant difference between the architecture under test and the architectures on which the surrogate benchmark is trained.

C. CHALLENGES/ATTRIBUTES OF EFFICIENT NAS BENCHMARKS

There are several challenges for creating an efficient and well-defined NAS benchmark as it involves many steps of carefully designing the search space, possessing computation resources, and training pipeline [61]. An efficient NAS algorithm should balance difficulty, expressive power, complexity, novelty, and achievable performance. The attributes and challenges in designing a benchmark are outlined below:

1) TASK DEFINITION

The main challenge is to define a diverse set of tasks and datasets which are to be used in the benchmark. It is also important to report the quality of the dataset, as well as any augmentation or preprocessing that may be necessary. Several NAS methods are evaluated on commonly available high-quality public datasets such as CIFAR or ImageNet. Hence, the datasets should also be easily available to researchers working in this domain.

2) SEARCH SPACE

The search space is a key component in a NAS algorithm, and any benchmark differs in terms of the search space as the rest of the pipeline, such as training and metric collection, remains almost the same. The search space should be well-curated for the corresponding search elements for the target application and datasets. A wrong search space that does not have good accuracy on the target dataset or an impractical search space on which the search algorithms do not work might not be helpful to the NAS research community. The search space often limits the model performance gain on a given task. The search algorithm should discover unusual designs to obtain validation accuracy beyond a threshold. Additionally, the search space should be flexible to allow for a wide range of performance levels that are challenging and require advanced search algorithms to find an efficient model.

3) ENUMERATION/SAMPLING

All the architectures in the predefined search space should be properly enumerated well in Tabular Benchmark, and it needs to be noted that repetitions do not happen. It is very common in the micro/cell-based search space to have *isomorphic* cells, networks with the same topology. NAS-Bench-101 [11] utilizes an iterative graph hashing method to detect whether two cell architectures are isomorphic. Also, for Surrogate Benchmarks, the networks need to be sampled in such a way that they are evenly distributed across a wide range of accuracies, the number of parameters, FLOPs, etc. The Surrogate Benchmarks, with more architectures clustered around single/multiple regions, introduce bias in the surrogate models.

4) EXPRESSIVENESS

A NAS Benchmark should be representative enough in such a way that it considers mixed types of tasks and neural architecture that are commonly encountered, such that a NAS algorithm can be evaluated effectively for different scenarios. Traditionally, the NAS algorithms limit their search elements to reduce the search computation time. Nonetheless, considering a search space with diverse elements would be very useful for building more robust models. It is also important to define the space to ensure that the NAS algorithm has enough freedom to explore a wide range of networks but not so much freedom that the search process becomes impractical. Additionally, depending only on a few benchmarks can lead to over-fitting [62], and therefore, NAS algorithms should be evaluated on a variety of benchmarks to understand the true potential of the algorithm.

5) COMPLEXITY

The difficulty level of the benchmarks implies how hard it is to attain acceptable performance on the given search space and respective metrics. It also indicates how complicated is the search space in terms of the number of operations allowed, the complicated structure of the pathways within a network, and the density of possible connections of nodes. A few SOTA benchmark datasets are easily designed in such a way that random search methods can also work well. It is important to create challenging benchmarks to obtain meaningful evaluations from the NAS algorithm, but not extremely difficult that they are impossible to find networks in a reasonable time. A challenging search space allows complex operations and connections within the network and includes search elements that are difficult to optimize, which leads to unconventional or novel architectures. A good benchmark should provide validation metrics for a wide range of complexity, such as different numbers of classes or input dimensions, to test the robustness of a NAS algorithm.

6) EVALUATION PROTOCOLS/METRICS

The main advantage of NAS Benchmarks is in assisting the search process by quickly evaluating the predicted architectures. Therefore, it is extremely important for NAS Benchmarks to clearly define the evaluation metrics of the task and dataset. For example, if a NAS Benchmark considers multiple tasks, such as Image Segmentation and Detection, all the corresponding validation metrics and protocols should be properly presented.

7) TRAINING RESOURCES

The main difficulty in creating an efficient NAS Benchmark lies in training every architecture on the target dataset to fetch the training and evaluation metrics. The networks in benchmarks require multiple runs as the performance of architecture can vary depending on the weight initialization and other factors. Hence, a single run of a network may not fully represent the true capabilities of a network. Therefore, owing large-scale computation resources for repeated training, such as GPUs or TPUs, is crucial. In the case of HW-NAS Benchmarks, as the typical process involves capturing the hardware metrics, the target device's availability and expertise in the efficient implementation of the network layers on the target system is a big challenge.

8) OLD BENCHMARKS

It is helpful for NAS researchers to use the existing and well-established benchmarks as a starting point to design when designing a NAS benchmark. These benchmarks have typically been widely used and validated in the literature and can provide a useful baseline for comparison. However, it may also be necessary to modify these benchmarks to

better fit the specific needs of the NAS algorithm. Although it requires the NAS algorithm to include competitive architecture spaces to increase performance on a given task, the benchmarks that have proven helpful for the research community should not be discarded.

9) DOCUMENTATION

The tabular benchmark datasets should be released in an easy-to-access format to easily query the metrics. The code associated with the benchmark generation should also be documented properly. The benchmarks should not be limited to just the validation accuracy and must also report others performance metrics such as the total number of weights, memory, FLOPs, etc. The papers reporting benchmarks using surrogate models should consider describing and documenting the code for the following: data collection, building surrogate models, validation, and training data of both architectures in the search space and surrogate model.

V. NAS BENCHMARKS FOR CNNs

This section provides an overview of prominent NAS Benchmarks related to vision tasks such as Image Classification, Object Detection, etc. We report the number of unique neural architectures, tasks, type of architecture (cell-based or layer-wise), training details, and the metrics reported for every model in each benchmark.

A. NAS-BENCH-101

NAS-bench-101 [11] is a pioneering work and the first rigorous dataset for NAS reproducibility research. It provides a tabular dataset of over 423k unique neural architectures mapped to their training time and validation accuracy on the CIFAR-10 dataset [14]. NAS-Bench-101 defines a graph search space Ω of all CNN architectures with a fixed backbone, comprising of a head (3×3 Convolution), body, and a tail (Average Pooling and an FC layer), as shown in Figure 4. The body is built by alternatively stacking a block three times between two down-sampling operations, where the stacked block is a feed-forward structure of the three repeated cells.

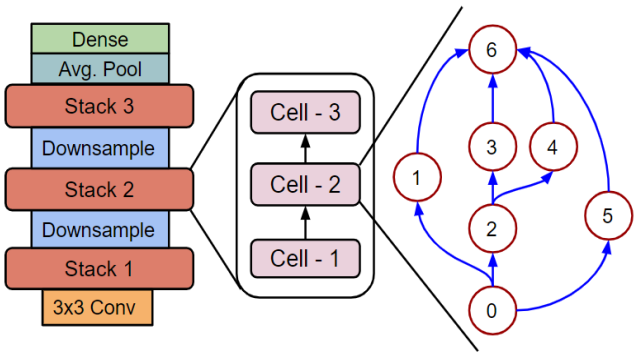

FIGURE 4. Search Space in NAS-Bench-101 [11].

Each cell in the stacked block is a Directed Acyclic Graph (DAG) with seven nodes and a maximum of nine edges that can take any operation from the following: {1×1 Convolution (Conv 1×1), 3×3 Convolution (Conv 3×3), Max Pooling (max-pool)}. Each cell is composed of one input node (Node 0), one output node (Node 6), and at most five internal nodes (Nodes 1-5) that can take any operation in the predefined search space. Therefore, this NAS benchmark dataset consists of 423,624 architectures obtained by enumerating all possible DAGs within the search space. Each architecture has been trained and evaluated three times (three different initializations) with the same experimental setting (like learning rate) in each run to prevent bias in the implementation. The networks are trained on the CIFAR-10 dataset for a total of 108 epochs, and their fitness is evaluated at various training epochs (4, 12, 36, and 108). The dataset returns 0 for a missing neural architecture, and the best test accuracy on CIFAR-10 out of all the networks is 94.23%. The benchmark is not diverse enough in a way that it does not contain diverse tasks and datasets. Although most networks are centered around 90%, it has a long tail towards the low accuracy cluster, i.e., around 10%. Despite limitations, NAS-Bench-101 is still a powerful dataset to study the performance of different NAS algorithms.

B. NAS-BENCH-201

NAS-bench-201 [12] is based on DARTS search space [4] and is different from NAS-Bench-101 in terms of search space, diagnostic information, and results on multiple datasets such as CIFAR-10, CIFAR-100 [14], and ImageNet-16-120 [64]. NAS-bench-201 is a cell-based search space with four intermediate nodes and five operations as follows: zeroize (none), skip connection, 1×1 Convolution, 3×3 Convolution, and Average Pooling. The macro-architecture contains three cell stages (Figure 5) with 16, 32, and 64 channels, respectively, in each stage and a residual module of stride two between the three cells. The cell architecture is designed only for normal cells, and the reduction cell is similar to the structure of ResNet [65] to restrict the search space size. There are 15,625 unique networks in the benchmark that are trained and evaluated on the three different image classification datasets to allow transfer learning. NAS-bench-201 benchmark dataset provides full training information about training, validation, and test losses/accuracies on the three mentioned datasets for a total of 200 epochs. The authors show the effectiveness of the benchmark by evaluating it on different NAS methods such as Regularized Evolution [66], Random Search [7], DARTS [4]. The limitation of NAS-Bench-201 is that it includes a relatively small number of models compared to many possible architectures that could be tested. Therefore, this benchmark may not be a full representation of a wide range of networks. NAS-Bench-201 is small and inexpressive, such as that simple search strategy like local [67] or random search works well on this benchmark. The search spaces of both NAS-Bench-101 and NAS-Bench-201 are fixed in terms of encoding spaces with a limited number of nodes and edge types in a cell. The results obtained on

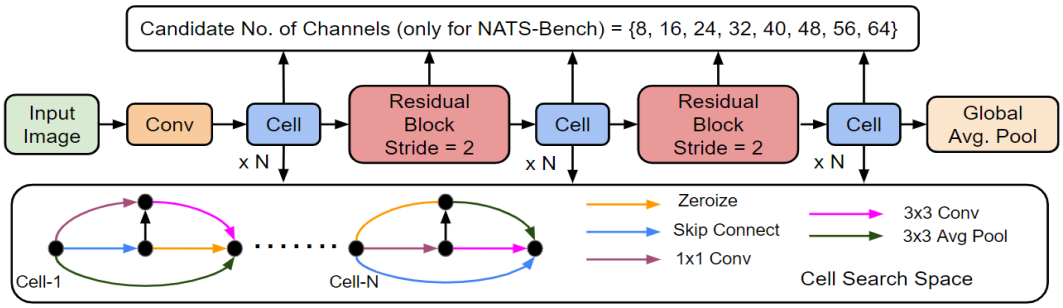

FIGURE 5. The Topology and Search Space in NAS-bench-201 [12] and NATS-Bench [63].

this benchmark cannot be generalized on larger benchmarks, datasets and other types of coding spaces.

C. NATS-BENCH

NATS-Bench [63] is a direct extension to NAS-bench-201 [12] with an enlarged search space and more architectures in the benchmark dataset. The search space (Figure 5) is equivalent to the macro skeleton in NAS-bench-201 benchmark [12], with the inclusion of the channel size in the cells as the search elements. NATS-Bench consists of two search spaces with different distributions: **(1)** NATS-Bench TSS: a topology search space consisting of 15,625 architectures which have varying operations at each edge of the cell and fixed channel/filter sizes (same as NAS-bench-201), **(2)** NATS-Bench SSS: a size search space of 32,768 networks where the number of channels is variable from the following list: {8, 16, 24, 32, 40, 48, 56, 64}.

D. NAS-BENCH-301

NAS-Bench-301 [59], [60] provides training time, accuracy (training, validation, and test), and the number of parameters for every randomly sampled 60k models in the cell-based search space on the CIFAR-10 dataset. The search space is based on the DARTS-based space with the following seven search elements: 3 × 3.5×5 Separable Convolution, 3 × 3.5×5 Dilation Convolution, Max Pooling, Avg Pooling, and Skip connection. Even though the total number of unique architectures possible with the DARTS-based space and the seven search elements is more than 10^{18}, only 60k models are trained as the authors use an auxiliary ML model to predict the performance of other networks in the search space. The surrogate model, made up of a Graph Isomorphism Network (GIN), is used to query the metrics of other unknown trained networks. The authors empirically show that the surrogate model trained on the subset of 60k trained networks can outperform the tabular benchmark. The three best-performing surrogate models are LGBoost, XGBoost and GIN, and they outperform other models such as Random Forests, Support Vector Regression and NGBoost. The main limitation of this method is the number of models sampled to train the surrogate model. The authors sample only 60k, which is $6*10^{-7}$% of the total 10^{18} networks. The models are sampled using Random search, which makes it hard to justify that the sampled architectures are representative of the large search space, and thus the generality is questionable.

E. NAS-BENCH-1Shot1

NAS-Bench-1Shot1 [68] is a benchmark dataset to bridge the gap between the discrete space in NAS-Bench-101 [11] and the continuous space of One-shot NAS such as the differentiable NAS. The NAS-Bench-101 is successful in the fair evaluation of different NAS techniques, but it cannot be directly utilized in the One-shot methods [4], [69] due to the benchmark's discrete nature. Figure 6 depicts the construction of a supernetwork in the One-shot NAS method where the weights of all discrete architectures are shared within a supermodel. On the other hand, Figure 4 illustrates an example of an individual architecture present in the NAS-Bench-101 dataset [11]. Therefore, the NAS-Bench-101 architecture space does not match the space present in the One-shot methods, requiring modifications to the former (NAS-Bench-101) to adapt to the latter (One-shot). NAS-Bench-1Shot1 includes a set of three benchmarks based on the number of parent nodes, containing 6240, 29160, and 363648 networks, respectively, to track the performance of the searched networks computationally cheaply. The process of querying validation metrics from the discrete space of NAS-Bench-101 is as follows: **(1)** First, the operation at each choice edge is chosen based on the highest architectural weight, **(2)** The parent node of each choice edge is chosen from the k choices, **(3)** A discrete cell structure is constructed from the operation and node list, and the performance metrics can be obtained from the benchmark.

F. NAS-BENCH-MR

Ding et al. proposed NAS-Bench-MR [70], a multi-task NAS benchmark specific to four different vision applications for learning task-transferable networks. The authors explored the possibility of building benchmarks for vision application for tasks other than the traditional Image Classification task, as most of the previous works focused on this single task. They pointed out that networks like ResNet [65] work well on Image Classification but fail to deliver expected performance on other vision tasks. Therefore,

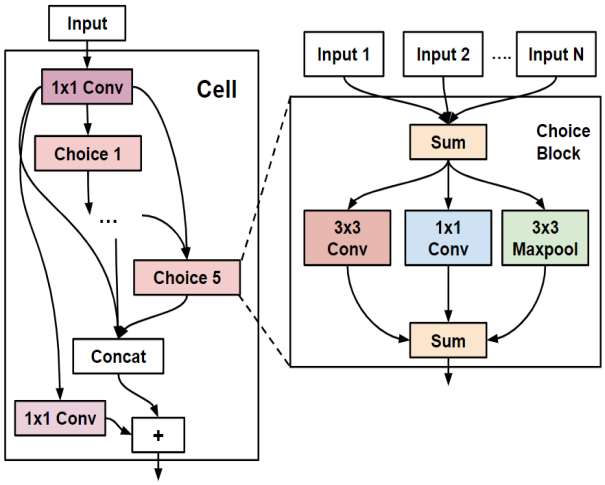

FIGURE 6. Supernetwork in One-Shot NAS [68].

manually designing a single architecture for numerous proposes is difficult in practice. The authors developed a methodology to learn a versatile model that is able fit multiple datasets and work better on following diverse vision tasks: **(i)** Image Classification on ImageNet [15], **(ii)** Segmentation on Cityscapes [71], **(iii)** 3D Detection on KITTI [72], **(iv)** Video Recognition on HMDB51 [73]. The search space is a multi-resolution universal search space that includes numbers of resolutions, blocks, and channels as the search elements. It contains mixed granularities of feature map representations for different applications, like high-resolution for segmentation and low-resolution maps for Image classification. NAS-Bench-MR randomly samples 2.5k networks from the multi-resolution search space and trains on the previously mentioned datasets under nine different settings. NAS-Bench-201 and NAS-Bench-MR differ in neural architecture size, search space, and training setting. The features, such as the number of channels, blocks, and resolutions, are fixed in the former benchmark, while the latter includes them in the search space. NAS-Bench-MR accommodates deeper, shallower, wider, and narrower models by including high and low-level feature maps.

G. NAS-BENCH-MACRO

NAS-Bench-Macro [74] provides the validation accuracy, number of parameters, and FLOPs of various networks on a layer-wise search space. MobileNetV2-like backbone network [48], consisting of a 3×3 Convolution layer, eight sequential searchable blocks, and an FC layer for classification is used as a search space. The choices in the search space for the searchable units are as follows: **(i)** Identity or skip connection, **(ii)** MBConv block of kernel size 3 and expansion ratio 3 (MBConv_3_3), **(iii)** MBConv block of kernel size 5 and expansion ratio 6 (MBConv_5_6). Therefore, the total number of unique architectures for eight searchable units and three choices in the NAS-Bench-Macro benchmark is $3^8 = 6561$. Each network architecture is trained thrice on different seeds on the CIFAR-10 dataset with a batch size of 256. Their mean validation accuracies in three runs are reported as the final validation accuracy. The best-performing network in the entire benchmark dataset has a test accuracy of 93.13%, and the average accuracy of all models is roughly 90.4% on the CIFAR-10 dataset.

H. TransNAS-BENCH-101

TransNAS-Bench-101 [75] is a tabular dataset for transfer learning across seven vision tasks, mainly for transferability and generalizability of NAS algorithms. The main goal is to find universal architectures across heterogeneous tasks, ease the comparability issue and curtail the computation hurdle of transferable NAS. TransNAS-Bench considers two search spaces with different search elements, defined as follows: **(i)** Cell-based search space similar to NAS-Bench-201 [12] with the following search elements: 1×1 Convolution, 3×3 Convolution, Zeroize and Skip Connection resulting in 4096 models, **(ii)** Macro search space of repeating residual blocks where the search space is composed of the number of blocks (network depth) and block operations at which the feature map is reduced to half, and the number of filters is doubled, resulting in 3256 unique models. A total of 7352 distinct neural architectures are trained on seven vision tasks such as Object Classification, Scene Classification, Room Layout, Jigsaw, Auto Encoder, Surface Normal, Semantic Segmentation, and diagnostic data such as task performance (Eg: Validation accuracy), inference execution time, and FLOPs are reported. The FLOPs of the individual model are calculated on the input image of size (224, 224, 3), and the inference latency is measured on one Nvidia V100 GPU on the image of dimension (1080, 720, 3).

I. NAS-BENCH-360

NAS-Bench-360 [76] is a benchmark suite for evaluating SOTA NAS algorithms on ten diverse application domains, datasets, task dimensionalities, and learning methods on an academic budget. Unlike other benchmarks, this benchmark does not provide a tabular dictionary of different metrics or evaluate architectures in the search space. Instead, this benchmark dataset organizes different tasks, search elements, search spaces, and datasets to help the NAS community quickly obtain information about variegated tasks for fast evaluation of their NAS methods. The tasks and associated datasets are as follows: **(1)** Standard Image Classification using DenseNet-BC search space [77] on CIFAR-100 dataset [14], **(2)** Classifying spherically projected images [78], **(3)** Classifying Electromyography signals on NinaPro DB5 dataset [79], **(4)** Labeling sound events FSD50K [80], **(5)** Solving partial differential equations (PDEs) [81], **(6)** Protein distance prediction [82], **(7)** Identifying cosmic ray contamination [83], **(8)** Detecting heart disease [84], **(9)** Satellite image time series analysis [85], **(10)** Predicting functional effects from genetic sequences on DeepSEA [86].

J. NAS-BENCH-x11

Single-fidelity is defined as accessing the training/validation loss/accuracy information of a network at only the final epoch, while multi-fidelity refers to obtaining the training metric at any arbitrary epochs during the end-to-end training. It is possible to integrate a few multi-fidelity methods, such as early stopping and learning curve exploration, into the NAS algorithms to make them more efficient. The earlier discussed NAS benchmarks allowed either single-fidelity or very limited multi-fidelity. For example, NAS-Bench-301 contains statistics only at epoch 100, while NAS-Bench-101 stores the validation accuracies only at the following epoch numbers: 4, 12, 36, and 108. NAS-Bench-x11 [87] resolves this issue by predicting the full training information at any random epoch through a surrogate model rather than just the best validation accuracy. The authors introduce three surrogate benchmarks, namely NAS-Bench-111, NAS-Bench-311, and NAS-Bench-NLP11, that build on top of NAS-Bench-101, NAS-Bench-301, and NAS-Bench-NLP to estimate the full learning curve information. The three surrogate benchmarks are generated from the original benchmarks using a method based on singular value decomposition and noise modeling. The authors show the effectiveness of the benchmark by applying on popular NAS methods, such as Regularized Evolution [66], Local Search [67], and BANANAS [9].

K. NAS-BENCH-SUITE

Mehta et al. [88] studied the existing benchmarks and made the followings assumptions about NAS methods: **(1)** If a NAS method performs well on a particular benchmark such as NAS-Bench-101 or NAS-Bench-201, it does not automatically work on other benchmarks with a different search space, **(2)** The default hyperparameters of the existing NAS algorithms may not be robust, and there is still room to tune them better, **(3)** The tuning of hyperparameters of a NAS algorithm on a specific benchmark or search space and these to other space could make the performance of the search method remarkably worse. Hence, it can be concluded that a NAS method that works on tiny subsets of search space or benchmarks does not transfer well across different tasks or datasets, and thus generalizing is a question. NAS-Bench-Suite [88] is a collection of several previously discussed benchmarks such as NAS-Bench-101 [11], NAS-Bench-201 [12], NAS-Bench-301 [59], NAS-Bench-1Shot1 [68], TransNAS-Bench-101 [70], NAS-Bench-MR [70], NAS-Bench-x11 [87]. The suite also consists of NAS-Bench-NLP [13] and NAS-Bench-ASR [89] that are reviewed in the upcoming section. NAS-Bench-Suite is an extensible collection of numerous benchmarks of 25 combinations of search spaces and datasets, queryable through a unified interface for reproducibility and generalizability. This work reduces the burden of researchers to gather information on different search space benchmarks and datasets to evaluate the NAS algorithms, thereby assisting the community in developing methods that are more generalizable on new and unseen applications.

L. NAS-BENCH-SUITE-ZERO

Zero-cost proxy is a technique to estimate the relative performance of an architecture from a single minibatch of input data. NAS-Bench-Suite-Zero [90], a benchmark collection of 13 Zero-cost proxies on 28 tasks, is used to analyze the generalizability and complementary information. The authors also present a concrete method to reduce bias to improve the efficiency of Zero cost proxy methods. The search space and tasks are directly acquired from NAS-Bench-101, NAS-Bench-201, NAS-Bench-301, and TransNAS-Bench-101. NAS-Bench-Suite-Zero extends NAS-Bench-Suite by adding two datasets from NAS-Bench-360, and four datasets from Taskonomy [91].

M. BenchENAS

BenchENAS [92] is a benchmark for a fair evaluation of Evolutionary Neural Architecture Search methods. The benchmark consists of nine representative Evolutionary search algorithms, which include LargeEvo [93], CGP-CNN [94], Generic CNN [95], HierarchicalCNN [96], AE-CNN [97], EvoCNN [98], NSGA-Net [99], RegularizedEvo [66], CNN-GA [100]. The metrics in the benchmark include accuracy, #FLOPS, #parameters on MNIST [101], CIFAR and ImageNet datasets for different optimizers, learning rates and batch sizes. The experiments on BenchNAS provide insights into the strengths and weaknesses of each ENAS algorithm.

VI. NAS BENCHMARKS FOR NON-VISION APPLICATIONS

In the previous section, we discussed benchmarks specific to Vision applications that are primarily built on Convolution based networks. In this section, we discuss the NAS benchmarks on non-CV tasks, such as Natural Language Processing (NLP), Automatic Speech Recognition (ASR) and architectures such ad Graph Neural Networks (GNNs).

A. NAS-BENCH-NLP

NAS-Bench-NLP [102] is the first tabular NAS benchmark for tasks in Natural Language Processing domain, such as next word prediction. The search space consists of a novel search space that is comprised of the standard RNN architecture and the modified LSTM and GRU cells. The search space resembles NAS-Bench-101 as the DAG structure has at most 25 nodes, and each cell can take one out of the seven elements. The search elements in the benchmark consist of the following seven choices: Linear: $f(x_1, x_2, \ldots x_n) = W_1x_1 + W_2x_2 + \ldots W_nx_n + b$, Blending (element-wise): $f(z, x, y) = z \odot x + (1 - z) \odot y$, Element-wise product and sum, Activation Functions: Tanh, Sigmoid, and Leaky ReLU. The benchmark generated around 14k models and their training performance on the Penn Tree Bank (PTB) dataset [103], 4k of them are trained thrice with different settings, and the rest are trained only once. The best performing 289 networks on the PTB dataset are additionally trained on the WikiText-2 dataset [104]. The dataset provides the following information about each network in the search space: the number of parameters,

training time, and validation perplexities at 5, 10, 25, and 50 epochs. This benchmark cannot be used for modern NAS research in NLP, as Transformer NAS research [42] is way ahead of LSTM and GRU. In fact, the future work of NAS-Bench-NLP [13] pointed out that their benchmarks are not as efficient as Transformer models. Therefore, the next direction to explore in NLP NAS benchmark design is to create efficient benchmarks on Transformers.

B. NAS-BENCH-ASR

Mehrotra et al. proposed NAS-Bench-ASR [89], the first benchmark for the Automatic Speech Recognition (ASR) task, evaluated on the TIMIT audio dataset [105]. The benchmark is a tabular dataset consisting of 8,242 unique network architectures trained for three predefined epochs, each with different initializations. The fundamental unit in the search space is a convolution-based cell or DAG topology in the overall macroarchitecture. The DAG consists of four nodes (T1, T2, T3, T4) and two types of edges: main and skip. The main edge is composed of six operations: linear, four convolutions with different choices of the kernel and dilation size $\in \{(5, 1), (5, 2), (7, 1), (7, 2)\}$. The skip edge is either an identity or a zero operation that generates a tensor of zeros of the input dimension. The benchmark provides validation accuracy per epoch, and the final test accuracy on the TIMIT audio dataset [105]. Additionally, it also gives the number of parameters, FLOPs, and latencies of these networks on diverse platforms such as desktop (Tesla 1080Ti) and embedded GPUs (Jetson Nano) for different batch sizes. The networks performing very well on the TIMIT dataset are transferred to the larger LibriSpeech dataset [106].

C. NAS-BENCH-GRAPH

The wide presence of graph data led to the development of Graph Neural Networks (GNNs) [107] for various tasks such as node classification, link prediction, and graph pattern recognition. Naturally, Graph Neural Architecture Search [108] has gained momentum in searching for efficient GNN models. NAS-Bench-Graph [109] is the first benchmark to support unified, reproducible, and efficient evaluation for Graph Neural Architecture Search. This benchmark features 26,206 unique GNN architectures, which are compact compared to the large-scale benchmarks but expressive in nature. The architectures in this benchmark are trained on nine diverse graph datasets and reported metrics such as train, validation, and test accuracy in every training epoch, the latency, the number of trainable parameters, etc. NAS-Bench-Graph also provides latency on Intel CPU and Nvidia GPU hardware platforms.

VII. HW-NAS BENCHMARKS

The hardware-agnostic NAS benchmarks discussed in the previous sections offer only the training-related metrics and do not necessarily provide the hardware performance metrics such as latency, energy, memory consumption, etc. Therefore, these benchmarks limit the usage in developing HW-aware NAS algorithms, especially for non-hardware researchers. The diverse methodologies in the latency measurement pipeline, programming environment, and limited hardware knowledge make it difficult for the search algorithms to find optimal architecture for real-time deployment and comparison of different HW-NAS methods. The Hardware-aware NAS Benchmarks can alleviate this burden by providing precomputed hardware performance metrics on multiple platforms for efficiency in the search process, reproducibility, and systematic comparison.

A. LatBench

Dudziak et al. proposed LatBench [110], a large-scale runtime latency dataset for multi-objective Neural Architecture Search on many devices, including desktop/mobile/embedded CPU/GPU/TPU/DSP. This benchmark differs from the earlier benchmarks, where the previous authors approximated the latency using proxies or summing up layer-wise latencies. The authors of LatBench run each network architecture in the NAS-Bench-201 search space to capture the latencies. The benchmark obtains latencies on the following device setting: **(i)** Desktop CPU: Intel Core i7-7820X, **(ii)** Desktop GPU: Nvidia GTX 1080 Ti, **(iii)** Embedded GPU: Nvidia Jetson Nano, **(iv)** Embedded TPU - Google EdgeTPU, **(v)** Mobile GPU - Qualcomm Adreno 612 GPU, **(vi)** Mobile DSP - Qualcomm Hexagon 690 DSP. The Tensorflow framework with the 1.15.0 version is used to run models on the Desktop CPU, Desktop GPU, and Embedded GPU, while TensorFlow Lite Runtime 2.1.0 is used for Embedded TPU. On the other hand, Snapdragon Neural Processing Engine (SNPE) software with version 1.36.0.746 is employed to accelerate networks on Mobile GPU and Mobile DSP. Each model in the NAS-Bench-201 dataset is run 1000 times on the aforementioned non-mobile device setting using a batch size of 1 and 32×32 patch size. Along with the latency dataset, the authors also proposed **Binary Relation Predictor-NAS** (BRP-NAS) to predict the end-to-end latency based on a Graph Convolution Network (GCN) that outperforms proxy metrics or layer-wise latencies on different devices.

B. HW-NAS-BENCH

Dong and Yang et al. proposed HW-NAS-Bench [12], a Hardware-aware architecture search benchmark to gather hardware cost metrics on a diverse set of platforms, including Nvidia Edge GPU Jetson TX2 (Edge GPU), Raspberry Pi 4, Edge TPU Dev Board (Edge TPU), Pixel 3 (Mobile Phone), Eyeriss (ASIC), XilinxZC706 (FPGA). It differs from the previous LatBench in terms of selected devices, hardware metrics, search space, hardware evaluation metric pipeline, and the verification of obtained benchmarks. LatBench is limited to very few commercial devices, while HW-NAS-BENCH extends the estimation on non-commercial accelerators such as Eyeriss and Xilinx FPGA, along with evaluation on commercial systems. LatBench considers latency evaluation only in the NAS-Bench-201 space, while HW-NAS-Bench evaluates latency and energy of every architecture

in the following search spaces: Cell-based (46875 models in NAS-Bench-201) and layer-wise (10^{21} models in FBNet) search spaces. Additionally, HW-NAS-Bench provides a comprehensive description for collecting the desired hardware cost on these devices and verifies that the platform-specific HW-NAS can lead to optimal network-cost solutions through their analysis. The authors performed the hardware cost collection in parallel and spent about a month to obtain the desired metrics on the NAS-Bench-201 space. The latency of models in the FBNet search space is estimated by summing the latencies of individual operations instead of measuring the end-to-end latency on the target hardware due to the presence of a large number of architectures in the search space. This leads to ignoring optimizations such as cache reuse, layer fusion etc. The authors show the importance of HW-NAS-Bench by evaluating using ProxylessNAS [50] on CIFAR-100.

C. BLOX

BLOX [111] provides validation metrics of 91,125 models on a macro search space trained on the CIFAR-100 dataset. The latency of every model in the search space is reported on Nvidia GTX 1080 Ti and Qualcomm Snapdragon 888 with Hexagon 780 DSP. The search elements include Vgg16-style 3×3 Convolution, ResNet-style bottleneck with 5×5 Depthwise Separable Convolution and EfficientNet-style Fused Inverted Bottleneck modules. BLOX enables the study of emerging blockwise knowledge distillation search algorithms, such as DONNA [112]. The benchmark only evaluates models with three stages, whereas the benchmark, like NAS-Bench-Macro, studies up to eight stages. Although BLOX is useful for emerging blockwise search and layer-wise search space, it provides evaluations only on a single dataset (CIFAR-100). The best accuracy is only 76.6%, whereas methods like P-DARTS [113] report an accuracy of more than 80%.

D. EC-NAS-BENCH

EC-NAS-Bench [114] is an energy consumption-aware tabular benchmark for multi-objective optimization NAS research. The Benchmark provides training energy consumption, power consumption of CPUs, GPUs, and DRAM, and carbon footprint for all architectures present in the NAS-Bench-101 dataset. The models are trained only for four epochs on a single Nvidia Quadro RTX 6000 GPU, following the same training hyperparameter setting and strategy of NAS-Bench-101. The surrogate time and energy measurements for all the runs are obtained through a surrogate linear scaling model. EC-NAS-Bench quantifies the resource costs specific to model training and reports the total resource costs and computational overhead. The Benchmark is evaluated on the multi-objective optimization NAS strategies, and the results showed that the Pareto-optimal solutions offer better trade-offs between the hardware performance metrics and the model training energy consumption.

VIII. NAS-HPO BENCHMARKS

The benchmarks discussed in the previous sections are based on neural architecture optimization. In addition to the neural operators, the hyperparameters during the training process affect the model performance. This section explores joint Hyperparameter Optimization and Neural Architecture Search Benchmarks for further NAS research.

A. NAS-HPO-BENCH

NAS-HPO-Bench [115] incorporates hyperparameter and architecture search elements in a single benchmark on a plain two-layer MLP network on four regression datasets. The benchmark consists of 62208 feed-forward configurations on the following datasets: Protein structure [116], Slice localization [117], Naval propulsion [118], and Parkinson's telemonitoring [119]. The initial learning rate, batch size, learning rate scheduler, activation functions, and neuron size in both FC layers are chosen from the following pool: {.0005,.001,.005,.01,.05,.1}, {8, 16, 32, 64}, {cosine, fix} and {relu, tanh}, {16, 32, 64, 128, 256, 512}, respectively.

B. NAS-HPO-BENCH-II

NAS-HPO-Bench-II [120] is built by combining the cell-based neural architecture search space and training-related hyperparameters such as the learning rate and batch size. The benchmark consists of 4k unique cell architectures built from the following search elements: 3×3 Convolution, 3×3 Average Pooling of stride 1, Skip connection, and a Zero operation. The batch size and learning rate are chosen from {16, 32, 64, 128, 256, 512} and {0.003125, 0.00625, 0.0125, 0.025, 0.5, 0.1, 0,2, 0.4}, respectively, resulting in a total of 192k configurations (4k*6*8 = 192k). Each model with a different setting is trained thrice for 12 epochs on the CIFAR-10 dataset and reports training/validation/testing accuracies, losses, and the time elapsed for every epoch. Finally, a surrogate model is built to estimate the performance of every architecture after 200 epochs. The dataset for training the surrogate model is built by training 100 randomly sampled networks from the search space on all the combinations of learning rate and batch size, thereby producing 4.8k pairs.

C. LCBench

LCBench [121] studies the multi-fidelity optimization with respect to the learning curves on the joint space of architectural and training hyperparameters across 35 OpenML datasets [122]. The MLP search space includes common architecture and training hyperparameters such as the number of layers, number of units, size, learning rate, L2 regularization, momentum, and dropout rate. The benchmark dataset provides train, test, and validation accuracy/loss, global, and layer-wise gradient statistics of 2000 models.

Apart from NAS-HPO Benchmarks, a few HPO-only are HPOBench [123], LassoBench [124], YAHPO Gym [125].

TABLE 2. Summary of various NAS benchmarks.

Benchmark	Total Size	Search Space	Tasks/ Datasets	Operations	Tab. or Surr.	Metrics Reported	GitHub
NAS-Bench-101 [11]	423k	Cell	CIFAR-10	1×1/3×3 Conv Max Pool	Tabular	Val Acc. & Train Time	[126]
NAS-Bench-201 [12]	15k	Cell	CIFAR-10 CIFAR-100 ImageNet-16	1×1/3×3 Conv Avg Pool	Tabular	Train, Val, Test loss & Accuracies	[127]
NATS-Bench [63]	6k+ 32k	Cell	CIFAR-10 CIFAR-100 ImageNet-16	1×1/3×3 Conv Avg Pool Filter Sizes	Tabular	Train, Val., Test loss & Accuracies	[128]
NAS-Bench-301 [59]	60k	Cell	CIFAR-10	3×3/5×5 Conv, Max/Avg Pool	Surrogate	Val Acc.	[129]
NAS-Bench-1Shot1 [68]	399k	Cell	CIFAR-10	1×1/3×3 Conv Max Pool	Tabular	Validation Accuracy	[130]
NAS-Bench-MR [70]	2.5k	Multi-Res.	ImageNet, KITTI Cityscapes	Varying Operations	Tabular	Val Acc., #Params, FLOPs	[131]
NAS-Bench-Macro [74]	6k	Layer-wise	CIFAR-10	Skip, MBC_3_3 MBC_5_6	Tabular	Val Acc., #Params, FLOPs	[132]
TransNAS-Bench-101 [75]	7k	Macro & Cell	7 Applications	1×1/3×3 Conv Zero, Skip	Tabular	Val Acc., FLOPs, Inference Time	[133]
NAS-Bench-360 [76]	N/A	Cell & Layer-wise	Ten Diverse Tasks and Applications	Varying operations	Tabular	N/A	[134]
NAS-Bench-x11 [87]	423k, 10^{18}, 10^{53}	Cell	CIFAR-10	NAS-Bench-101/301	Surrogate	Train Info	[135]
NAS-Bench-Suite [88]	A suite of NAS Benchmarks -						[136]
NAS-Bench-Suite-Zero [90]	A collection of 13 Zero-cost proxies on 28 tasks						[137]
BenchENAS [92]	Collection of Several Evolutionary Search Methods						[138]
NAS-Bench-NLP [13]	14k	RNN, LSTM, GRU	PTB and WikiText-2	Linear, Blending, product & sum, Tanh, Sigmoid, LeakyReLU	Tabular	#Params, Train Time & Test Perplexity	[139]
NAS-Bench-ASR [89]	8k	Cell	ASR on TIMIT	Linear, Conv, Identity, Zeroize	Tabular	Val./Test Acc., #Params, FLOPs	[140]
NAS-Bench-Graph [109]	26k	Graph Cell	9 Graph Datasets	7 GNN layers	Tabular	#Params, Latency Train & Test Accuracy	[141]
LatBench [110]	15k	Cell	ImageNet	Same as NAS-Bench-201	Surrogate	Latency of models on CPU, GPU, TPU, DSP	[142]
HW-NAS-Bench [12]	15k	Cell & Layer	ImageNet, CIFAR-10	NAS-Bench-201 & FBNet	Tabular	Latency of models on CPU, GPU, ASIC, FPGA	[143]
BLOX [111]	91k	Layer	CIFAR-100	Conv., Identity, MBConv, BConv	Tabular	Latency on Nvidia GPU and Qualcomm DSP	[144]
EC-NAS-Bench [114]	15k	Cell	CIFAR-10	Same as NAS-Bench-101	Surrogate	Energy, Power on CPU, GPU	[145]
NAS-HPO-Bench [115]	62k	FC	Four Datsets	FC Neurons, LR, Scheduler, Batch size, Activation func.	Tabular	Validation Metrics	[146]
NAS-HPO-Bench-II [120]	192k	Cell	CIFAR-10	3×3 Conv, Skip Avg Pool, Batch Size, LR, Filter Sizes	Both	Validation Metrics	[147]
LCBench [121]	2k	Layer	35 datasets	Training Hyperparameters	Tabular	Validation Metrics	[148]

IX. DISCUSSION, LIMITATIONS AND FUTURE WORK

In this section, we discuss the limitations of existing Neural Architecture Search benchmarks and provide possible future directions. In Table 2, we provide a brief overview of all the NAS benchmarks that summarizes search space, size, operations, tasks, and the GitHub repository URLs of every benchmark. The current Benchmarks are helpful for NAS researchers, and in the future, we believe specific attention is needed to the following exciting:

A. NAS BENCHMARKS FOR LARGE SCALE DATASET

The major limitation of the SOTA benchmarks is the availability of validation accuracy on large-scale datasets, such as ImageNet [15]. This is understandable due to the fact that training large networks multiple times on a large dataset requires massive computing resources. The current benchmarks on small-scale datasets are useful for research purposes, but many NAS works have shown that the results on small-scale datasets do not efficiently transfer to large-scale datasets. A direction to explore in creating NAS Benchmarks for a large-scale dataset is designing better Surrogate models. The majority of SOTA NAS Benchmarks are considered on a very narrow search space of cell-based architecture and consist of only a few hundred or thousand neural networks. However, layer-wise search spaces like FBNet consist of nearly 10^{21} unique models, making it very difficult to build a tabular dataset. Therefore, surrogate models are extremely crucial for large search spaces to estimate the validation metrics of all the neural architectures. Once-for-all methodology [6] can also be used to measure the validation accuracy, which samples the desired network from a Supernetwork. Although the sampled networks generate close to near true accuracy, they can be used in the search process to find efficient models. Given the large amount of computation required, research groups around the world who have access to large-scale computing clusters can come together and work in a coordinated fashion to develop Benchmarks for large datasets.

B. BUILDING A NAS BENCHMARK COMMUNITY

If possible, a consortium can be established for NAS Benchmarks in the future, similar to the MLperf community [149], [150], [151], [152], where all the code and benchmark datasets should be stored for quick access to researchers. A leaderboard should be established in the public domain indicating the most effective approaches on each of the benchmarks. Also, competitions in the leading ML conferences can be held, where people can submit their NAS algorithm for evaluation on NAS benchmarks.

C. SEARCH SPACE

The search elements and spaces limit the NAS methods to find robust and efficient neural architectures. Most of the NAS Benchmarks are based on micro-search space where a cell architecture is repeated throughout the network. However, several HW-NAS algorithms have shown that a cell-based network is inefficient on several hardware platforms and instead adopt layer-wise search space such as MobileNetV2 [48] and MobileNetV3 [153]. The accuracy margin of the networks on these search spaces is not significantly high compared to the manually designed models, requiring the development of benchmarks in the other search spaces like EfficientNet [154]. Besides the traditional CNN structures in ResNet [65] and MobileNet [155], NAS has been successful in finding efficient models on various other search spaces, such as Transformers [156], Vision Transformer [157], Mixture-of-Expert (MoE) [158], and Graph Neural Networks [159]. Hence, the NAS community needs benchmarks in these diverse and prominent search spaces to push the ceiling beyond CNNs, even for computer vision applications.

D. HW-NAS BENCHMARKS

Although there is quite a lot of work on NAS Benchmarks, there are only a handful of HW-NAS Benchmarks available, as described in Section VII. Although the current benchmarks on cell-based architectures are useful for research purposes, layer-wise search space network evaluation on real/simulated hardware platforms is extremely critical for real-time deployment. There exist diversified hardware platforms such as tiny MCUs, server CPUs, multi-GPUs, in-computing memory (ReRAM), and many layer-wise spaces such as ResNet, MobileNetV3, EfficientNet, and several tasks, datasets, and applications for which NAS benchmarks can be targeted. The AI chip companies can boost the HW-NAS research community by releasing the NAS benchmarks consisting of performance metrics of architectures on various search spaces on their hardware. This can guarantee the optimal implementation of DNNs on hardware and set a standard for network comparison. The SOTA HW-NAS Benchmarks include only a dense Convolution layer while excluding sparse multiplication in both cell and layer-wise search space. Also, the current HW-NAS benchmarks do not consider sparsity-supporting hardware and mixed precision Quantization. Hence, the NAS community needs sparse and mixed-precision quantized benchmarks to search for more robust models that are small in size and faster on hardware.

E. NAS BENCHMARKS FOR OTHER APPLICATIONS

As we reviewed several NAS Benchmarks, it was evident that most of the works targeted Image Classification applications and very few other Vision and language tasks. NAS has been very successful in discovering SOTA models for various other applications such as Object Detection [160], Semantic Segmentation [161], Machine Translation [162], etc. Therefore, building an optimized NAS Benchmark for other tasks can be crucial for NAS research.

F. NETWORK AND ACCELERATOR CO-SEARCH

The neural architecture and hardware accelerator co-search is a problem where the neural network and ASIC dimensions

are simultaneously searched to get the best performance in both worlds. The benchmarks related to co-search remain untouched or least explored. The authors of DANCE [163] measure the latency and energy metrics of 1.8 million network-hardware pairs from the MobileNetV2 search space and different sizes of the accelerator, but the open-source implementation is not available. Although the HW-NAS-Bench includes the evaluation of several architectures on accelerators, the results cannot be used directly for the co-search problem as the micro-architectural properties of the hardware, such as an array, register, and RAM size, are fixed. As a result, the ML-system community requires neural architectures with different search spaces to be evaluated on different shapes and sizes of the accelerator to boost the co-search research.

G. INTEGRATING WITH DL FRAMEWORKS

The significant development of NAS Benchmarks over the years led to the creation of several toolkits and frameworks for these benchmark datasets. Neural Network Intelligence (NNI) [164] is an AutoML toolkit for NAS, HPO, and model compression that includes NAS-Bench-101 and NAS-Bench-201. Archai [165], a platform for NAS to ensure reproducibility and fair comparison, includes NAS benchmarks from several search spaces and search methods. Hence, rapid inclusion of all the NAS benchmarks is crucial for fast querying and efficient comparison. NAS-Bench-Suite [88] is the first step towards including several NAS Benchmarks under a single framework.

As we emphasized the challenges and need for more efficient benchmarks to push the limits of NAS, in the coming years, we foresee a boom in NAS benchmarks that are more diverse with respect to search space, applications, hardware platforms, a blend of tabular and surrogate, training aspects with proper APIs and documentation to support easy access of several metrics.

X. CONCLUSION

NAS Benchmarks provide a set of precomputed metrics of several architectures and evaluation protocols, which allows NAS researchers to quickly evaluate and compare different methods. In recent years, these Benchmarks have significantly contributed to the development of novel NAS algorithms. We, in this paper, provided an overview of recent advances in this narrow and emerging field and summarized many SOTA NAS Benchmarks. We mainly discussed the structure of each Benchmark and provided URLs for every open-sourced work. Many search methods rely on NAS benchmarks to show the efficiency of their search algorithm. Going forward, NAS Benchmarks will be influential in the development of efficient search algorithms. The NAS Benchmarks have come a long way and become mature in the last few years, aiding the optimization of NAS algorithms. Hence, more well-defined, documented, and easy-to-use benchmarks are needed to assist this process.

ACKNOWLEDGMENT

All opinions, findings, and conclusions expressed are those of the authors.

REFERENCES

[1] A. A. Abdelhamid, E.-S.-M. El-Kenawy, B. Alotaibi, G. M. Amer, M. Y. Abdelkader, A. Ibrahim, and M. M. Eid, "Robust speech emotion recognition using CNN+LSTM based on stochastic fractal search optimization algorithm," *IEEE Access*, vol. 10, pp. 49265–49284, 2022.

[2] N. Lopac, F. Hrzic, I. P. Vuksanovic, and J. Lerga, "Detection of non-stationary GW signals in high noise from Cohen's class of time–frequency representations using deep learning," *IEEE Access*, vol. 10, pp. 2408–2428, 2022.

[3] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.

[4] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.

[5] Y. Chen, G. Meng, Q. Zhang, S. Xiang, C. Huang, L. Mu, and X. Wang, "RENAS: Reinforced evolutionary neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4787–4796.

[6] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 2019, *arXiv:1908.09791*.

[7] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 367–377.

[8] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, Mar. 2021.

[9] C. White, W. Neiswanger, and Y. Savani, "BANANAS: Bayesian optimization with neural architectures for neural architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 12, pp. 10293–10301.

[10] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 7588–7598.

[11] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-Bench-101: Towards reproducible neural architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7105–7114.

[12] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," 2020, *arXiv:2001.00326*.

[13] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, and E. Burnaev, "NAS-Bench-NLP: Neural architecture search benchmark for natural language processing," 2020, *arXiv:2006.07116*.

[14] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., 2009.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[16] D. Jacob Kedziora, K. Musial, and B. Gabrys, "AutonoML: Towards an integrated framework for autonomous machine learning," 2020, *arXiv:2012.12600*.

[17] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*. Cham, Switzerland: Springer, 2019.

[18] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106622.

[19] Q. Yao, M. Wang, Y. Chen, W. Dai, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu, "Taking human out of learning applications: A survey on automated machine learning," 2018, *arXiv:1810.13306*.

[20] E.-G. Talbi, "Automated design of deep neural networks: A survey and unified taxonomy," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–37, Mar. 2022.

[21] X. Dong, D. J. Kedziora, K. Musial, and B. Gabrys, "Automated deep learning: Neural architecture search is not the end," 2021, *arXiv:2112.09245*.

[22] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.

[23] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," 2019, *arXiv:1905.01392*.

[24] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–34, 2021.

[25] Y.-Q. Hu and Y. Yu, "A technical view on neural architecture search," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 4, pp. 795–811, Apr. 2020.

[26] G. Kyriakides and K. Margaritis, "An introduction to neural architecture search for convolutional networks," 2020, *arXiv:2005.11074*.

[27] D. Baymurzina, E. Golikov, and M. Burtsev, "A review of neural architecture search," *Neurocomputing*, vol. 474, pp. 82–93, Feb. 2022.

[28] L. Xie, X. Chen, K. Bi, L. Wei, Y. Xu, L. Wang, Z. Chen, A. Xiao, J. Chang, X. Zhang, and Q. Tian, "Weight-sharing neural architecture search: A battle to shrink the optimization gap," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–37, Dec. 2022.

[29] S. Cha, T. Kim, H. Lee, and S.-Y. Yun, "A survey of supernet optimization and its applications: Spatial and temporal optimization for neural architecture search," 2022, *arXiv:2204.03916*.

[30] S. Santra, J.-W. Hsieh, and C.-F. Lin, "Gradient descent effects on differential neural architecture search: A survey," *IEEE Access*, vol. 9, pp. 89602–89618, 2021.

[31] S. Liu, H. Zhang, and Y. Jin, "A survey on computationally efficient neural architecture search," 2022, *arXiv:2206.01520*.

[32] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter, "Neural architecture search: Insights from 1000 papers," 2023, *arXiv:2301.08727*.

[33] X. Zhou, A. K. Qin, Y. Sun, and K. C. Tan, "A survey of advances in evolutionary neural architecture search," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2021, pp. 950–957.

[34] X. Xie, X. Song, Z. Lv, G. G. Yen, W. Ding, and Y. Sun, "Efficient evaluation methods for neural architecture search: A survey," 2023, *arXiv:2301.05919*.

[35] K. T. Chitty-Venkata and A. K. Somani, "Neural architecture search survey: A hardware perspective," *ACM Comput. Surveys (CSUR)*, 2022.

[36] L. Sekanina, "Neural architecture search and hardware accelerator co-search: A survey," *IEEE Access*, vol. 9, pp. 151337–151362, 2021.

[37] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," 2021, *arXiv:2101.09336*.

[38] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "Hardware-aware neural architecture search: Survey and taxonomy," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 4322–4329.

[39] X. Zhang, W. Jiang, Y. Shi, and J. Hu, "When neural architecture search meets hardware implementation: From hardware awareness to co-design," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 25–30.

[40] H. Zhu, H. Zhang, and Y. Jin, "From federated learning to federated neural architecture search: A survey," *Complex Intell. Syst.*, vol. 7, no. 2, pp. 639–657, Apr. 2021.

[41] D. Liu and Y. Cao, "Federated neural architecture search evolution and open problems: An overview," in *Proc. Int. Conf. Bio-Inspired Comput., Theories Appl.* Cham, Switzerland: Springer, 2021, pp. 330–345.

[42] K. T. Chitty-Venkata, M. Emani, V. Vishwanath, and A. K. Somani, "Neural architecture search for transformers: A survey," *IEEE Access*, vol. 10, pp. 108374–108412, 2022.

[43] V. V. V. Ganepola and T. Wirasingha, "Automating generative adversarial networks using neural architecture search: A review," in *Proc. Int. Conf. Emerg. Smart Comput. Informat. (ESCI)*, Mar. 2021, pp. 577–582.

[44] V. U. Buthgamumudalige and T. Wirasingha, "Neural architecture search for generative adversarial networks: A review," in *Proc. 10th Int. Conf. Inf. Autom. Sustainability (ICIAfS)*, Aug. 2021, pp. 246–251.

[45] Z. Zhang, X. Wang, and W. Zhu, "Automated machine learning on graphs: A survey," 2021, *arXiv:2103.00742*.

[46] B. M. Oloulade, J. Gao, J. Chen, T. Lyu, and R. Al-Sabri, "Graph neural architecture search: A survey," *Tsinghua Sci. Technol.*, vol. 27, no. 4, pp. 692–708, Aug. 2022.

[47] S. Poornachandra and S. Prapulla, "Neural architecture search in classical and quantum computers: A survey," *Int. Res. J. Eng. Technol.*, vol. 7, no. 6, pp. 1–6, 2020.

[48] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[49] B. Wu, K. Keutzer, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, and Y. Jia, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10734–10742.

[50] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," 2018, *arXiv:1812.00332*.

[51] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–13.

[52] Y. Guo, Y. Zheng, M. Tan, Q. Chen, Z. Li, J. Chen, P. Zhao, and J. Huang, "Towards accurate and compact architectures via neural architecture transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6501–6516, Oct. 2022.

[53] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, Li-Jia Li, Li Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 19–34.

[54] Y. Jaafra, J. L. Laurent, A. Deruyver, and M. S. Naceur, "Reinforcement learning for neural architecture search: A review," *Image Vis. Comput.*, vol. 89, pp. 57–66, Sep. 2019.

[55] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 550–570, Feb. 2023.

[56] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight NAS," 2021, *arXiv:2101.08134*.

[57] C. Li, T. Tang, G. Wang, J. Peng, B. Wang, X. Liang, and X. Chang, "BossNAS: Exploring hybrid CNN-transformers with block-wisely self-supervised neural architecture search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12281–12291.

[58] A. Yang, P. M. Esperança, and F. M. Carlucci, "NAS evaluation is frustratingly hard," 2019, *arXiv:1912.12522*.

[59] A. Zela, J. Siems, L. Zimmer, J. Lukasik, M. Keuper, and F. Hutter, "Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks," 2020, *arXiv:2008.09777*.

[60] A. Zela, J. N. Siems, L. Zimmer, J. Lukasik, M. Keuper, and F. Hutter, "Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–7.

[61] M. Lindauer and F. Hutter, "Best practices for scientific research on neural architecture search," *J. Mach. Learn. Res.*, vol. 21, no. 243, pp. 1–18, 2020.

[62] B. Koch, E. Denton, A. Hanna, and J. G. Foster, "Reduced, reused and recycled: The life of a dataset in machine learning research," 2021, *arXiv:2112.01716*.

[63] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3634–3646, Jul. 2021.

[64] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of ImageNet as an alternative to the CIFAR datasets," 2017, *arXiv:1707.08819*.

[65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[66] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.

[67] C. White, S. Nolen, and Y. Savani, "Exploring the loss landscape in neural architecture search," 2020, *arXiv:2005.02960*.

[68] A. Zela, J. Siems, and F. Hutter, "NAS-Bench-1Shot1: Benchmarking and dissecting one-shot neural architecture search," 2020, *arXiv:2001.10422*.

[69] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[70] M. Ding, Y. Huo, H. Lu, L. Yang, Z. Wang, Z. Lu, J. Wang, and P. Luo, "Learning versatile neural architectures by propagating network codes," 2021, *arXiv:2103.13253*.

[71] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[72] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. (2015). *The KITTI Vision Benchmark Suite*. [Online]. Available: http://www.cvlibs.net/datasets/kitti

[73] D. S. Wishart et al., "HMDB: The human metabolome database," *Nucleic Acids Res.*, vol. 35, pp. D521–D526, Jan. 2007.

[74] X. Su, T. Huang, Y. Li, S. You, F. Wang, C. Qian, C. Zhang, and C. Xu, "Prioritized architecture sampling with Monto-Carlo tree search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10968–10977.

[75] Y. Duan, X. Chen, H. Xu, Z. Chen, X. Liang, T. Zhang, and Z. Li, "TransNAS-Bench-101: Improving transferability and generalizability of cross-task neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 5251–5260.

[76] R. Tu, N. Roberts, M. Khodak, J. Shen, F. Sala, and A. Talwalkar, "NAS-Bench-360: Benchmarking neural architecture search on diverse tasks," 2021, *arXiv:2110.05668*.

[77] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[78] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, "Spherical CNNs," 2018, *arXiv:1801.10130*.

[79] M. Atzori, A. Gijsberts, S. Heynen, A.-G.-M. Hager, O. Deriaz, P. van der Smagt, C. Castellini, B. Caputo, and H. Müller, "Building the Ninapro database: A resource for the biorobotics community," in *Proc. 4th IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatronics (BioRob)*, Jun. 2012, pp. 1258–1265.

[80] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "FSD50K: An open dataset of human-labeled sound events," 2020, *arXiv:2010.00475*.

[81] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," 2020, *arXiv:2010.08895*.

[82] B. Adhikari, "A fully open-source framework for deep learning protein real-valued distances," *Sci. Rep.*, vol. 10, no. 1, pp. 1–10, Aug. 2020.

[83] K. Zhang and J. S. Bloom, "DeepCR: Cosmic ray rejection with deep learning," *Astrophys. J.*, vol. 889, no. 1, p. 24, Jan. 2020.

[84] S. Hong, Y. Xu, A. Khare, S. Priambada, K. Maher, A. Aljiffry, J. Sun, and A. Tumanov, "HOLMES: Health OnLine model ensemble serving for deep learning models in intensive care units," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 1614–1624.

[85] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining Knowl. Discovery*, vol. 34, no. 5, pp. 1454–1495, Sep. 2020.

[86] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature Methods*, vol. 12, no. 10, pp. 931–934, Oct. 2015.

[87] S. Yan, C. White, Y. Savani, and F. Hutter, "Nas-Bench-x11 and the power of learning curves," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–12.

[88] Y. Mehta, C. White, A. Zela, A. Krishnakumar, G. Zabergja, S. Moradian, M. Safari, K. Yu, and F. Hutter, "NAS-Bench-Suite: NAS evaluation is (now) surprisingly easy," 2022, *arXiv:2201.13396*.

[89] A. Mehrotra, A. G. C. Ramos, S. Bhattacharya, Ł. Dudziak, R. Vipperla, T. Chau, M. S. Abdelfattah, S. Ishtiaq, and N. D. Lane, "NAS-Bench-ASR: Reproducible neural architecture search for speech recognition," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–12.

[90] A. Krishnakumar, C. White, A. Zela, R. Tu, M. Safari, and F. Hutter, "NAS-Bench-Suite-Zero: Accelerating research on zero cost proxies," 2022, *arXiv:2210.03230*.

[91] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3712–3722.

[92] X. Xie, Y. Liu, Y. Sun, G. G. Yen, B. Xue, and M. Zhang, "BenchENAS: A benchmarking platform for evolutionary neural architecture search," *IEEE Trans. Evol. Comput.*, vol. 26, no. 6, pp. 1473–1485, Dec. 2022.

[93] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2902–2911.

[94] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2017, pp. 497–504.

[95] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1379–1388.

[96] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," 2017, *arXiv:1711.00436*.

[97] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020.

[98] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, Apr. 2020.

[99] 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," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2019, pp. 419–427.

[100] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.

[101] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[102] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, A. Filippov, and E. Burnaev, "NAS-Bench-NLP: Neural architecture search benchmark for natural language processing," *IEEE Access*, vol. 10, pp. 45736–45747, 2022.

[103] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, 2010, vol. 2, no. 3, pp. 1045–1048.

[104] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016, *arXiv:1609.07843*.

[105] J. S. Garofolo, "Timit acoustic phonetic continuous speech corpus," Linguistic Data Consortium, Tech. Rep., 1993.

[106] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5206–5210.

[107] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2020.

[108] Y. Li and I. King, "AutoGraph: Automated graph neural network," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2020, pp. 189–201.

[109] Y. Qin, Z. Zhang, X. Wang, Z. Zhang, and W. Zhu, "NAS-Bench-Graph: Benchmarking graph neural architecture search," 2022, *arXiv:2206.09166*.

[110] Ł. Dudziak, T. Chau, M. S. Abdelfattah, R. Lee, H. Kim, and N. D. Lane, "BRP-NAS: Prediction-based NAS using GCNs," 2020, *arXiv:2007.08668*.

[111] T. C. P. Chau, Ł. Dudziak, H. Wen, N. D. Lane, and M. S. Abdelfattah, "BLOX: Macro neural architecture search benchmark and algorithms," 2022, *arXiv:2210.07271*.

[112] B. Moons, P. Noorzad, A. Skliar, G. Mariani, D. Mehta, C. Lott, and T. Blankevoort, "Distilling optimal neural networks: Rapid search in diverse spaces," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12229–12238.

[113] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1294–1303.

[114] P. Bakhtiarifard, C. Igel, and R. Selvan, "Energy consumption-aware tabular benchmarks for neural architecture search," 2022, *arXiv:2210.06015*.

[115] A. Klein and F. Hutter, "Tabular benchmarks for joint architecture and hyperparameter optimization," 2019, *arXiv:1905.04970*.

[116] P. Rana. (2013). *Physicochemical Properties of Protein Tertiary Structure Data Set*. UCI Machine Learning Repository. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Struct

[117] F. Graf, H.-P. Kriegel, M. Schubert, S. Pölsterl, and A. Cavallaro, "2D image registration in CT images using radial image descriptors," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2011, pp. 607–614.

[118] A. Coraddu, L. Oneto, A. Ghio, S. Savio, D. Anguita, and M. Figari, "Machine learning approaches for improving condition-based maintenance of naval propulsion plants," *Inst. Mech. Eng., M, J. Eng. Maritime Environ.*, vol. 230, no. 1, pp. 136–153, Jul. 2014.

[119] A. Tsanas, M. Little, P. McSharry, and L. Ramig, "Accurate telemonitoring of Parkinson's disease progression by non-invasive speech tests," *Nature Precedings*, vol. 2009, p. 1, Oct. 2009.

[120] Y. Hirose, N. Yoshinari, and S. Shirakawa, "Nas-HPO-Bench-II: A benchmark dataset on joint optimization of convolutional neural network architecture and training hyperparameters," in *Proc. Asian Conf. Mach. Learn.*, 2021, pp. 1349–1364.

[121] L. Zimmer, M. Lindauer, and F. Hutter, "Auto-PyTorch: Multi-fidelity MetaLearning for efficient and robust AutoDL," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 3079–3090, Sep. 2021.

[122] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo, "OPENML: Networked science in machine learning," *ACM SIGKDD Explor. Newslett.*, vol. 15, no. 2, pp. 49–60, 2014.

[123] K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, and F. Hutter, "HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO," 2021, *arXiv:2109.06716*.

[124] K. Šehić, A. Gramfort, J. Salmon, and L. Nardi, "LassoBench: A high-dimensional hyperparameter optimization benchmark suite for lasso," 2021, *arXiv:2111.02790*.

[125] F. Pfisterer, L. Schneider, J. Moosbauer, M. Binder, and B. Bischl, "YAHPO gym—An efficient multi-objective multi-fidelity benchmark for hyperparameter optimization," 2021, *arXiv:2109.03670*.

[126] (2021). *NAS-Bench-101*. [Online]. Available: https://github.com/google-research/nasbench

[127] (2021). *NAS-Bench-201*. [Online]. Available: https://github.com/D-X-Y/NAS-Bench-201

[128] (2021). *NATS-Bench*. [Online]. Available: https://github.com/D-X-Y/NATS-Bench

[129] (2021). *NAS-Bench-301*. [Online]. Available: https://github.com/automl/nasbench301

[130] (2021). *NAS-Bench-1Shot1*. [Online]. Available: https://github.com/automl/nasbench-1shot1

[131] (2021). *NAS-Bench-MR*. [Online]. Available: https://github.com/dingmyu/NCP

[132] (2021). *NAS-Bench-Macro*. [Online]. Available: https://github.com/xiusu/NAS-Bench-Macro

[133] (2021). *TransNAS-Bench-101*. [Online]. Available: https://github.com/kmdanielduan/TransNASBench

[134] (2021). *NAS-Bench-360*. [Online]. Available: https://github.com/rtu715/NAS-Bench-360

[135] (2021). *NAS-Bench-x11*. [Online]. Available: https://github.com/automl/nas-bench-x11

[136] (2021). *NAS-Bench-Suite*. [Online]. Available: https://github.com/automl/NASLib

[137] (2022). *NAS-Bench-Suite-Zero*. [Online]. Available: https://github.com/automl/naslib/tree/zerocost

[138] (2022). *Benchenas*. [Online]. Available: https://benchenas.com/

[139] (2021). *NAS-Bench-NLP*. [Online]. Available: https://github.com/fmsnew/nas-bench-nlp-release

[140] (2021). *NAS-Bench-ASR*. [Online]. Available: https://github.com/SamsungLabs/nb-asr

[141] (2022). *NAS-Bench-Graph*. [Online]. Available: https://github.com/THUMNLab/NAS-Bench-Graph

[142] (2021). *LatBench*. [Online]. Available: https://github.com/SamsungLabs/eagle

[143] (2021). *HW-NAS-Bench*. [Online]. Available: https://github.com/RICE-EIC/HW-NAS-Bench

[144] (2022). *BLOX*. [Online]. Available: https://github.com/SamsungLabs/blox

[145] (2022). *EC-NAS-Bench*. [Online]. Available: https://github.com/PedramBakh/EC-NAS-Bench

[146] (2021). *NAS-HPO-Bench*. [Online]. Available: https://github.com/automl/nas_benchmarks

[147] (2021). *NAS-HPO-Bench-II*. [Online]. Available: https://github.com/yoichii/nashpobench2api

[148] (2021). *LCBench*. [Online]. Available: https://github.com/automl/LCBench

[149] V. J. Reddi et al., "MLPerf inference benchmark," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA)*, May 2020, pp. 446–459.

[150] P. Mattson, V. J. Reddi, C. Cheng, C. Coleman, G. Diamos, D. Kanter, P. Micikevicius, D. Patterson, G. Schmuelling, H. Tang, G.-Y. Wei, and C.-J. Wu, "MLPerf: An industry standard benchmark suite for machine learning performance," *IEEE Micro*, vol. 40, no. 2, pp. 8–16, Mar. 2020.

[151] P. Mattson et al., "MLPerf training benchmark," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 336–349, Mar. 2020.

[152] C. Banbury et al., "MLPerf tiny benchmark," 2021, *arXiv:2106.07597*.

[153] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[154] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[155] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[156] D. So, Q. Le, and C. Liang, "The evolved transformer," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5877–5886.

[157] M. Chen, H. Peng, J. Fu, and H. Ling, "AutoFormer: Searching transformers for visual recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12270–12280.

[158] S. Latifi, S. Muralidharan, and M. Garland, "Efficient sparsely activated transformers," 2022, *arXiv:2208.14580*.

[159] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graph neural architecture search," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1403–1409.

[160] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, and Y. Zhang, "NAS-FCOS: Fast neural architecture search for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11943–11951.

[161] A. Shaw, D. Hunter, F. Landola, and S. Sidhu, "SqueezeNAS: Fast neural architecture search for faster semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1–11.

[162] Y. Fan, F. Tian, Y. Xia, T. Qin, X. Li, and T. Liu, "Searching better architectures for neural machine translation," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 28, pp. 1574–1585, 2020.

[163] K. Choi, D. Hong, H. Yoon, J. Yu, Y. Kim, and J. Lee, "DANCE: Differentiable accelerator/network co-exploration," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 337–342.

[164] Microsoft. *Neural Network Intelligence*. (2021). [Online]. Available: https://github.com/microsoft/nni

[165] Microsoft. (2022). *Archai*. [Online]. Available: https://github.com/microsoft/archai

KRISHNA TEJA CHITTY-VENKATA received the B.E. degree in electronics and communication from the University College of Engineering, Osmania University, Hyderabad, India, in 2017. He is currently pursuing the Ph.D. degree with Iowa State University, Ames, IA, USA. He interned with the Argonne National Laboratory, Intel Corporation, and AMD, where he worked on various problems on efficient deep learning. His research interests include designing network design algorithms and compression methods for neural network processing, such as pruning, quantization, neural architecture search, and general and special-purpose hardware platforms.

MURALI EMANI received the Ph.D. degree from the School of Informatics, The University of Edinburgh, U.K., in 2015. He was a Postdoctoral Research Staff Member with the Lawrence Livermore National Laboratory. He is currently an Assistant Computer Scientist with the Data Science Group, Argonne Leadership Computing Facility (ALCF), Argonne National Laboratory. He co-leads the AI Testbed with ALCF to evaluate novel AI accelerators for scientific machine learning applications. He has organized workshops and participated in tutorials that include benchmarking deep-learning workloads on emerging hardware, MLPerf-Bench at MLSys 2020, MLSys 2021, ISPASS 2020, ISPASS 2021, and ASPLOS 2021. His research interests include scalable machine learning, emerging HPC and AI architectures, and AI for science. He serves as the Co-Chair for the MLPerf HPC Group, MLCommons, to benchmark large-scale ML on HPC systems. He has also co-chaired the MLPerf birds-of-a-feather sessions at SC 2019, SC 2020, and SC 2021.

VENKATRAM VISHWANATH received the Ph.D. degree in computer science from the University of Illinois at Chicago, in 2009. He is currently a Computer Scientist with the Argonne National Laboratory. He is the Data Science Team Lead with the Argonne Leadership Computing Facility (ALCF). His current focus is on algorithms, system software, and workflows to facilitate data-centric applications on supercomputing systems. His research interests include scientific applications, supercomputing architectures, parallel algorithms and runtimes, scalable analytics, and collaborative workspaces. He received best paper awards at venues, including HPDC and LDAV, and a Gordon Bell Finalist.

ARUN K. SOMANI (Life Fellow, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from McGill University, Montreal, in 1983 and 1985, respectively. He was a Scientific Officer with the Government of India, New Delhi, and a Faculty Member with the University of Washington, Seattle, WA, USA. He is currently an Anson Marston Distinguished Professor of electrical and computer engineering with Iowa State University. He has delivered several keynote speeches, and distinguished and invited talks all over the world. His research interests include computer system design, architecture, fault-tolerant computing, computer interconnection networks, optical networking, and reconfigurable and parallel computer systems. He is a fellow of AAAS. He is a Distinguished Engineer of ACM and an Eminent Engineer of Tau Beta Pi. He has served as an IEEE Distinguished Visitor, an IEEE Distinguished Tutorial Speaker, and an IEEE Communication Society Distinguished Visitor.