AutoML: A Systematic Review on Automated Machine Learning with Neural Architecture Search

Article *in* Journal of Information and Intelligence · October 2023

DOI: 10.1016/j.jiixd.2023.10.002

7 authors, including:

Imrus Salehin
Pusan National University
31 PUBLICATIONS **311** CITATIONS
SEE PROFILE

Md Shamiul Islam
Bangladesh University of Business and Technology
15 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Pritom Saha
Daffodil International University
6 PUBLICATIONS **26** CITATIONS
SEE PROFILE

S M Noman
Frankfurt University of Applied Sciences
8 PUBLICATIONS **59** CITATIONS
SEE PROFILE

Contents lists available at ScienceDirect

Journal of Information and Intelligence

journal homepage: www.journals.elsevier.com/journal-of-information-and-intelligence

AutoML: A systematic review on automated machine learning with neural architecture search

Imrus Salehin [a,b,*], Md. Shamiul Islam [c], Pritom Saha [b], S.M. Noman [b,d], Azra Tuni [e], Md. Mehedi Hasan [f], Md. Abu Baten [g]

[a] *Department of Computer Engineering, Dongseo University, 47 Jurye-ro, Sasang-gu, Busan 47011, Republic of Korea*
[b] *Department of Computer Science and Engineering, Daffodil International University, Dhaka 1216, Bangladesh*
[c] *Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka 1216, Bangladesh*
[d] *Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences, Frankfurt 60318, Germany*
[e] *Department of Computer Science and Engineering, University of Asia Pacific, Dhaka 1205, Bangladesh*
[f] *Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka 1208, Bangladesh*
[g] *Department of Computer Science and Engineering, Northern University Bangladesh, Dhaka 1215, Bangladesh*

ARTICLE INFO

Keywords:
AutoML
Neural architecture search
Advance machine learning
Search space
Hyperparameter optimization

ABSTRACT

AutoML (Automated Machine Learning) is an emerging field that aims to automate the process of building machine learning models. AutoML emerged to increase productivity and efficiency by automating as much as possible the inefficient work that occurs while repeating this process whenever machine learning is applied. In particular, research has been conducted for a long time on technologies that can effectively develop high-quality models by minimizing the intervention of model developers in the process from data preprocessing to algorithm selection and tuning. In this semantic review research, we summarize the data processing requirements for AutoML approaches and provide a detailed explanation. We place greater emphasis on neural architecture search (NAS) as it currently represents a highly popular sub-topic within the field of AutoML. NAS methods use machine learning algorithms to search through a large space of possible architectures and find the one that performs best on a given task. We provide a summary of the performance achieved by representative NAS algorithms on the CIFAR-10, CIFAR-100, ImageNet and well-known benchmark datasets. Additionally, we delve into several noteworthy research directions in NAS methods including one/two-stage NAS, one-shot NAS and joint hyperparameter with architecture optimization. We discussed how the search space size and complexity in NAS can vary depending on the specific problem being addressed. To conclude, we examine several open problems (SOTA problems) within current AutoML methods that assure further investigation in future research.

1. Introduction

In recent times, deep learning has been widely implemented across various domains and has demonstrated remarkable success in solving challenging AI problems. This technology has been leveraged in areas like image classification [1,2] object detection [3] and language modeling [4,5] to achieve remarkable results. After the outstanding performance of AlexNet [6] in the 2012 imageNet large

[*] Corresponding author.
E-mail address: imrus15-8978@diu.edu.bd (I. Salehin).

https://doi.org/10.1016/j.jiixd.2023.10.002
Received 24 June 2023; Received in revised form 12 October 2023; Accepted 12 October 2023
Available online 18 October 2023

scale visual recognition challenge (ILSVRC) [7] surpassing all other conventional manual methods, more intricate and deeper neural networks have been introduced. For example, a standard ResNet-50 [8] model which has 50 layers, has approximately 25.6 million parameters. The initial stage utilizes 64 filters followed by the second stage with 128 filters, the third stage with 256 filters, the fourth stage with 512 filters and finally, the fifth stage with 1024 filters. Within the model, every residual block consists of two 3 × 3 convolutional layers accompanied by a shortcut connection. Each residual block in the model has two 3 × 3 convolutional layers and a shortcut connection. Now, the models prior to ResNet were designed manually through a trial-and-error process by experts which indicates that the development of high-performing models demands a considerable number of resources and time even for seasoned experts.

The excessive costs associated with model development have led to the emergence of a new concept which is the automation of the entire machine learning (ML) pipeline. This approach known as automated machine learning [9] aims to reduce the burden of development costs. Nowadays, the AutoML system has the capability to dynamically merge different techniques to establish a simple and user-friendly end-to-end ML pipeline system. To create a multitude of personalized AutoML methods, our objective is to develop the most sophisticated AI system.

The AutoML workflow involves several stages including feature engineering, data preparation, model generation and model evaluation. The process of model generation can be further categorized into search space exploration and optimization methods. The fundamental design principles of ML models are determined by the search space which can be classified into two categories: traditional ML models and neural architectures. In terms of classification, the optimization techniques can be categorized as hyperparameter optimization (HPO) [10] and architecture optimization (AO) [11]. HPO focuses on optimizing the parameters related to training such as the learning rate and batch size while AO focuses on optimizing the parameters associated with the model itself. The number of layers for neural architectures and the number of neighbors for KNN are examples of parameters that are optimized in architecture optimization (AO) techniques.

The main idea of neural architecture search [12] is to automate the process of designing and optimizing the architecture of neural networks. This approach aims to alleviate the burden of manual design and to improve the performance of neural networks by searching the space of architectures in an automated and efficient manner. NAS employs various methodologies including reinforcement learning, evolutionary algorithms, and gradient-based methods to explore and discover optimal architectures that can achieve high accuracy with minimal computational resources. The objective of a neural architecture search is to identify a neural architecture that exhibits both robustness and high performance. This is achieved by selectively choosing and combining fundamental operations from a predefined search space.

AutoML has emerged as a significant advancement in the field of robotics. It offers a streamlined and efficient way to develop machine learning models and algorithms for robotic systems, reducing the manual effort and expertise required in the traditional approach. AutoML in robotics involves automating various stages of the machine learning pipeline, including data preprocessing, feature engineering, model selection, hyperparameter tuning, and even deployment. For details about this, we have discussed it in the following section. While there are numerous exceptional reviews related to AutoML, our review stands out as it covers a wider range of AutoML methods beyond just NAS techniques. This paper provides a condensed overview of AutoML methods utilizing the complete AutoML according to the SoTA (State-of-the-Art) pipeline in Fig. 1 as a framework for the organization. The main purpose of this comprehensive overview is to offer beginners a thorough introduction to the ML field.

2. Systematic literature review approach

A systematic literature review is a rigorous and methodical process that involves analyzing with integrating existing research on a specific subject. It entails a structured approach of identifying relevant papers, evaluating the strength of evidence and synthesizing the findings to address a particular research issue. Adhering to established guidelines like PRISMA ensures a clear and iterative execution of this process. Data extraction and quality evaluation are performed to gather relevant data and assess the reliability and quality of the selected studies. Through further analysis, key conclusions are categorized and summarized. Subsequently, the results are interpreted in the context of the research topic, highlighting implications, limitations and recommendations for future studies. The systematic

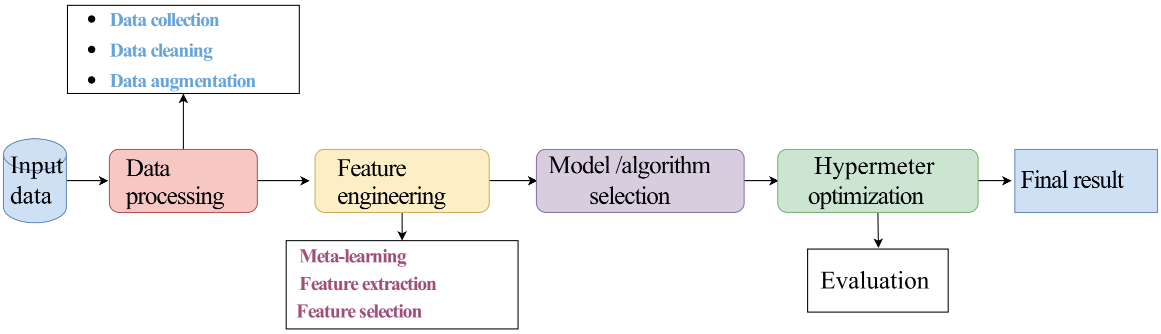

Fig. 1. An overview of AutoML pipeline with all stages.

evaluation of the literature facilitates a comprehensive and transparent assessment of available information sources fostering the advancement of scientific knowledge and evidence-based decision-making.

Step 1: Data processing search and selection

The data processing phase in a systematic literature review consists of a series of steps. These steps aim to identify relevant studies and select those that align with the predefined inclusion criteria ensuring the coherence between the chosen studies and the research objectives. The following link provides an overview of this data search process:

(1) Google Scholar (https://scholar.google.com).
(2) HAL Open Science (https://inria.hal.science/hal).
(3) Springer Link (https://link.springer.com).
(4) IEEE Xplore digital library (https://ieeexplore.ieee.org).
(5) Scopus (https://www.scopus.com).
(6) Elsevier ScienceDirect (https://www.sciencedirect.com).
(7) arXiv Paper (https://arxiv.org).

During the data processing phase of the systematic literature review, we employed a range of reputable databases and platforms. While Google Scholar is widely recognized and frequently used, it is important to consider the research question and subject area when selecting the database. To ensure a thorough search and selection process, it is expedient to utilize multiple databases taking into account the specific needs and requirements of the SLR. In a systematic literature review, search categories and keywords play a crucial role in effectively identifying relevant studies.

- **AutoML:** Automated Machine Learning, AutoML, Automatic Model Selection, Automated Model Training, AutoML Frameworks.
- **NAS:** Neural architecture search, NAS Algorithms, Automated Neural Network Design, Architecture Search Space, Neural Network Optimization, Neural Network Architecture Optimization.
- **Feature engineering:** Meta-learning.
- **Architecture Optimization:** Evolutionary Algorithm, Reinforcement Learning, Bayesian Optimization, Gradient-based Optimization.
- **Model Evaluation:** Weight sharing, Early stopping.

These keywords are intended to capture the core concepts related to AutoML, NAS, feature engineering, architecture optimization and model evaluation. By exploring the literature using these keywords, we read a wide range of research papers that address these key areas, contributing to a comprehensive understanding of AutoML, NAS and related topics. When searching for random keywords, we uncover fascinating developments in the field of AutoML and NAS such as the automated process of feature engineering, optimizing neural network architectures and evaluating model performance.

Step 2: Requisites for inclusion and exclusion

When considering the inclusion and exclusion of 175 papers from a dataset of 487 scientific literature papers, specific requisites were established to ensure the selection of relevant and suitable studies. These requisites typically involve criteria or characteristics that are used to determine whether a review paper should be included or excluded. The following are the criteria for inclusion and exclusion.

Requisites for inclusion:

- Relevance to the core concepts: The review paper directly addresses one or more of the core concepts related to AutoML, NAS, feature engineering, architecture optimization and model evaluation.
- Comprehensive coverage: The review paper provides a comprehensive analysis and synthesis of the literature on the selected core concepts, exploring relevant methodologies, algorithms, techniques and applications.
- Methodological rigor: The review paper demonstrates a sound methodology, such as a systematic or comprehensive approach, to ensure a thorough examination and evaluation of the literature on the chosen core concepts.
- Recent publication: Preference papers published within a specific timeframe to prioritize the inclusion of the most up-to-date research on AutoML, NAS, feature engineering, architecture optimization and model evaluation.

Requisites for Exclusion:

- Irrelevance to the core concepts: Review papers that do not directly relate to the core concepts of AutoML, NAS, feature engineering, architecture optimization and model evaluation.
- Limited scope or depth: Papers that provide a superficial or narrow analysis of the core concepts excluded in favor of those offering more comprehensive coverage.
- Methodological flaws: Papers with significant methodological limitations or flaws that may impact the validity or reliability of the findings related to the core concepts.
- Poor quality or inadequate reporting: Papers with inadequate reporting, incomplete data and low-quality content related to the core concepts are excluded to ensure the overall quality of the selected papers.

By applying these requisites for inclusion and exclusion based on the core concepts of AutoML, NAS, feature engineering, architecture optimization and model evaluation, the selection process ensures that the chosen 175 papers provide relevant, comprehensive and high-quality insights into these specific areas for further analysis and synthesis in the research industry.

Step 3: Literature selection process

The systematic literature review process for the selected keywords related to AutoML, NAS, feature engineering, architecture optimization and model evaluation involved four stages of screening: keyword search, title screening, abstract screening and full-text screening. The following provides a breakdown of each stage:

- **Keyword search:** Initially, a comprehensive search was conducted using the selected keywords. This search yielded a total of 487 papers that were potentially relevant to the review.
- **Title screening:** In this stage, the titles of the 287 papers were carefully screened to determine their relevance to the review topic. After the title screening process, 337 papers met the predetermined inclusion criteria.
- **Abstract screening:** The abstracts of the 337 papers were then screened to further narrow down the selection. Based on the screening process, 245 papers were found to be relevant and aligned with the review's objectives.
- **Full-text screening:** In the final stage, the full texts of the 245 papers were thoroughly assessed. This step involved a comprehensive evaluation of each paper to ensure it met all the inclusion criteria and contained the necessary information for the review. After this evaluation, a final set of 175 papers were selected for inclusion in the systematic literature review.

Following these four screening stages, the systematic literature review identified and included 175 papers (Table 1) deemed most relevant to the selected keywords and met the predetermined inclusion criteria, ensuring that only high-quality papers were included in the systematic literature review.

2.1. AutoML

AutoML technologies optimize the process of designing along with selecting machine learning models leading to more efficient with effective solutions. These advancements have the potential to drive significant progress and innovation in AI-related fields. Extracting both cutting-edge evolutionary algorithms (EAs) and distributed computing frameworks, LEAF (learning algorithms for earth observation fusion) uses these technologies which leverage evolutionary algorithms to automatically discover the best possible neural network architecture as well as hyperparameters for deep learning tasks [13]. AutoML-Zero, an open-source AutoML benchmark, aims to offer a comprehensive and accessible platform for the development and evaluation of automated machine learning techniques. Comparing four AutoML systems across 39 datasets this study explores the performance of automated machine learning methods [14].

The application of automated machine learning (AutoML) methods on approximately 300 datasets sourced from OpenML offers valuable insights into the performance and efficiency of these tools across a wide range of machine learning applications [15]. The decathlon benchmark designed for the analysis of medical images provides a comprehensive evaluation of multiple AutoML algorithm's effectiveness in classifying diverse medical image categories across all 10 datasets of the MedMNIST classification task [16]. VolcanoML is an AutoML framework designed to accelerate the end-to-end AutoML process by enhancing search space decomposition. It enables efficient exploration of the deep learning model design space [17]. A novel level-based taxonomy for AutoML systems discusses the potential opportunities and including democratizing machine learning and improving the accessibility of AI [18].

AutoML methods for multi-label categorization are one of the most difficult machine learning complications. On benchmark datasets, providing valuable insights into a grammar-based best-first search empirical evaluation of these approaches [19]. In order to evaluate and compare different AutoML systems three scenarios involving general machine learning, XGBoost and deep learning algorithm selection and tuning are employed as benchmarks [20]. AutoML techniques for unsupervised anomaly detection identify the key challenges of hyperparameters tuning, data cleaning and propose research directions for future developments [21]. Meta-learning techniques for identifying dataset features utilizing OpenML CC-18 benchmark with MetaBu meta-features improves the function of contemporary AutoML systems [22]. FLAML is a simple and inexpensive AutoML library that optimizes hyperparameters as well as model selection for other machine learning applications enabling efficient model training and deployment [23].

Table 1
Number of literature selection by database.

Data Sources	Number of papers according to keyword	Number of papers according to title	Number of papers according to abstract	Number of papers according to full text
Google Scholar	167	117	92	67
HAL Open Science	27	18	12	8
Springer Link	78	49	36	27
IEEE Xplore digital library	106	78	51	34
Scopus	41	30	21	15
Elsevier ScienceDirect	33	19	15	11
arXiv Paper	35	26	18	13
Total	487	337	245	175

Neural AutoML's transfer learning promises to speed up model training and reduce the time for vast volumes of labeled data utilizing RL-based architectural search [24]. Oracle AutoML enables fast and efficient creation of highly accurate machine learning models using an automated pipeline approach compared to cutting-edge open source AutoML solutions like H2O and auto-sklearn [25]. AutoML-Zero is a groundbreaking research effort that uses evolutionary algorithms to automatically design and optimize machine learning models from scratch without any human intervention [26]. AutoML for model compression (AMC) explores automated models for deployment on mobile devices to reduce computation costs, achieving significant compression and acceleration on VGG-16 models for ImageNet while maintaining high accuracy [27].

2.2. Neural architecture search (NAS)

Neural architecture search (NAS) holds the potential to revolutionize machine learning by automatically optimizing deep learning models for diverse applications paving the way for advancements in effectiveness and productivity. In their work, Marius Lindauer et al. present neural architecture search as a research approach that enables the comparison of various algorithms. NAS holds the potential to facilitate scientific advancements and revolutionize the field of machine learning. This underscores the importance of ongoing research in this domain to further improve the capability of deep learning models [28]. Jaehong Kim et al. emphasize the importance of meta-learning in the context of neural architecture search and outline a robotic strategy for generating meta-learners that can enhance the optimization of NAS algorithms [29]. In their research, Ziwei Zhang et al. talked about the challenges involved in designing efficient graph-based machine learning models exploring various approaches to automated machine learning on graphs such as graph neural networks, reinforcement learning and evolutionary algorithms [30]. V. Khoa et al. review multiple neural architecture search (NAS) methods applied in medical imaging, covering tasks like classification, segmentation, detection and reconstruction. They also investigate challenges, task integration and the use of meta-learning for few-shot learning in NAS [31]. Javad Artin et al. recommends the use of neural architecture search and linear regression for traffic forecasting considering current weather conditions. Their research suggests a novel approach that leverages NAS to automatically search for the optimal neural network design aiming to enhance prediction accuracy [32].

Youhei Akimoto et al. proposed a new optimization algorithm that combines natural gradient and adaptive learning rate methods with an adaptive step-size mechanism. The approach is designed to overcome the limitations of existing one-shot NAS algorithms that rely on weight sharing and parameter regularization [33]. For enhancing the differentiable architecture search (DARTS) technique, Xiangning Chen et al. suggest perturbation-based regularization Smooth-DARTS and point out the approach's limitations, specifically regarding instability and sensitivity to the learning rate [34].

ScaleNAS is a multi-path one-shot neural architecture search method that Hsin-Pai Cheng et al. describes to develop scale-aware high-resolution representations. The authors use a multi-path architecture and an efficient search space to increase the computational efficiency and accuracy of the NAS process, emphasizing scale awareness for high-resolution picture representation applications [35]. Santanu Santra et al. focus on discussing the implications of gradient descent on differential neural architecture search techniques as their main topic of research. Emphasizing the importance of optimizing the gradient descent process, the author highlights its significance in achieving efficient and effective neural architecture search. The author further explores different gradient-based optimization techniques and their applications in NAS providing valuable insights into this field [36]. NAS-HPO-Bench-II is a test dataset using neural architecture search and hyperparameter optimization techniques providing a valuable resource for evaluating and comparing their performance [37]. HardCoRe-NAS is a new approach for neural architecture search proposed by Niv Nayman et al. that applies hard constraints to the search space and introduces a differentiable relaxation of those constraints [38]. Joshua C. O. Koh et al. highlight the potential benefits of NAS in improving the efficiency and accuracy of plant phenotyping tasks such as plant growth monitoring and disease detection [39]. Wei Jia et al. automate the optimization of neural network designs for each modality, ensuring optimal performance. They conduct a comparative analysis of the suggested method against contemporary techniques to assess its performance. The research emphasizes the potential of NAS in enhancing the precision and effectiveness of biometric identification systems based on 2D and 3D palmprint and palm vein images [40]. Felipe Petroski proposes generative teaching networks (GTNs) which are neural networks that generate synthetic training data to accelerate the process of NAS. GTNs can reduce the number of actual training iterations required by generating realistic and diverse training data for the neural network, leading to faster and more efficient NAS [41]. KNAS, a green neural architecture search approach proposed by Jingjing Xu et al., attempts to conserve energy and CO_2 emissions throughout the search process to lessen the environmental effect of NAS algorithms [42]. A few-shot neural architecture search technique is suggested by Yiyang Zhao et al. that learns to effectively look for high-performing designs on a small number of training samples [43]. Hongpeng Zhou introduces BayesNAS, a novel Bayesian method for neural architecture search which has been developed recently. In this approach, a Gaussian learning process is utilized to forecast the anticipated validation accuracy of a certain design while the search is steered by Bayesian optimization [44]. Zhichao Lu et al. highlights the importance of using appropriate performance metrics and benchmark datasets to evaluate the effectiveness of NAS algorithms using multiobjective optimization (EMO) algorithms [45].

NAS-Bench-NLP offers an extensive evaluation of NLP model performance, training and inference efficiency, as well as the influence of various design choices on these metrics. This comprehensive analysis supports the advancement of automated NAS methods enabling the development of more efficient and effective NLP models [46]. The BLOX benchmark along with its associated algorithms, offers a holistic framework for macro neural architecture search facilitating the efficient exploration of the design space for deep learning models. This framework considers important factors such as model accuracy, inference speed and parameter efficiency employing the CIFAR-100 dataset for evaluation [47].

2.3. Industrial use of NAS and AutoML

Neural architecture search is an automated technology that finds extensive utilization in the industrial realm for optimizing deep neural applications on memristive platforms, effectively decreasing the necessity for high-resolution analog-to-digital converters (ADCs) [48]. In recent times, automated and efficient transformer models have gained significant prominence in both industry and academia. These models employ neural architecture search (NAS) techniques to craft the most suitable architecture, achieving the highest possible model accuracy while adhering to constraints related to latency, energy consumption and peak power usage [49–51].

Conventional machine learning methods often prove unsuitable for addressing the challenges posed by contemporary technologies such as low accuracy, detection rates, and data imbalances. In the present era, AutoML stands out as one of the most effective methodologies for rapidly processing vast amounts of data and efficiently detecting network intrusions [52,53]. AutoML has gained recognition for achieving high accuracy in various applications. It is utilized in predicting water quality, to address soil liquefaction potential issues, real-time predictions in news impacts on financial markets and healthcare in diseases predictions [54,55].

3. Motivation of the review study

This field is relatively new and holds significant appeal for enthusiastic researchers involved in developing AI and its subfields such as machine learning, deep learning and reinforcement learning. In current times, the primary objective of AutoML research is to automate the laborious and time-consuming aspects of constructing machine learning models including data preprocessing, algorithm selection and hyperparameter tuning. AutoML aims to increase productivity and efficiency in the machine learning workflow, allowing researchers and practitioners to focus on higher-level tasks such as problem formulation and result interpretation. Furthermore, AutoML aims to democratize machine learning by enabling individuals and organizations with limited machine learning expertise to utilize it effectively. This has the potential to drive innovation and progress across a broad spectrum of fields and industries.

However, in order to address the open issues and limitations, we mainly focus on determining certain requirements in this review paper. There are four main important requirements for AutoML performance.

- **Accuracy:** AutoML systems should strive to achieve high accuracy in their predictions or models. This involves minimizing errors and maximizing the correctness of the outputs generated by the automated processes.
- **Efficiency:** AutoML should be efficient in terms of time and computational resources. The automated processes should be designed to optimize the use of resources and reduce the overall time required to train models and make predictions.
- **Scalability:** AutoML systems should be capable of scaling up to handle large datasets and complex machine learning tasks. They should be able to handle increasing amounts of data and computational demands without compromising performance or efficiency.
- **Flexibility:** AutoML should offer flexibility in terms of supporting various machine learning algorithms, techniques and models. It should allow users to experiment with different approaches and customize the automated processes according to their specific requirements and preferences.

These four requirements play a crucial role in assessing the performance of AutoML systems and determining their effectiveness in automating the machine learning workflow. In detail, in our last open issue discussion section, we will base our discussion on these four items specifically focusing on performance and accuracy.

4. Data preparation

Data preparation stands as the foundational step in the ML and NAS pipelines. The act of data preparation involves collecting, merging, formatting and arranging data in a manner that facilitates its utilization for ML projects. Data preparation can be accomplished through three primary methods: data collection, data cleaning and data augmentation and we illustrated in Fig. 2.

The process of collecting data is considered essential. In this process, work-related data is collected from different sources. Data cleaning involves the identification of errors and issues in data followed by their correction and the filtering of faulty data to establish complete and accurate datasets. By generating new and diverse training datasets, data augmentation plays an important role in enhancing the performance as well as the outcomes of deep learning models. We will discuss the following topics in-depth in the subsection.

4.1. Data collection

Data collection is undeniably an intricate process. When combining datasets from diverse source systems, it is highly probable to encounter various challenges related to data quality, accuracy and consistency that require resolution. The collected data must undergo manipulation to ensure its usability and impertinent data needs to be filtered out. Recognizing the essentiality of datasets, a multitude of open datasets have emerged.

At the beginning of the machine learning study, researchers developed a handwritten digital dataset known as MNIST [56]. As time passed many large and valuable datasets like CIFAR-10 [57], CIFAR-100, ImageNet [58] and COCO [59] have developed to use in NAS as they are widely available and cover a wide range of tasks. However, finding a suitable dataset for any project is the most challenging part. There are several proposed methods two solve this problem as follows.

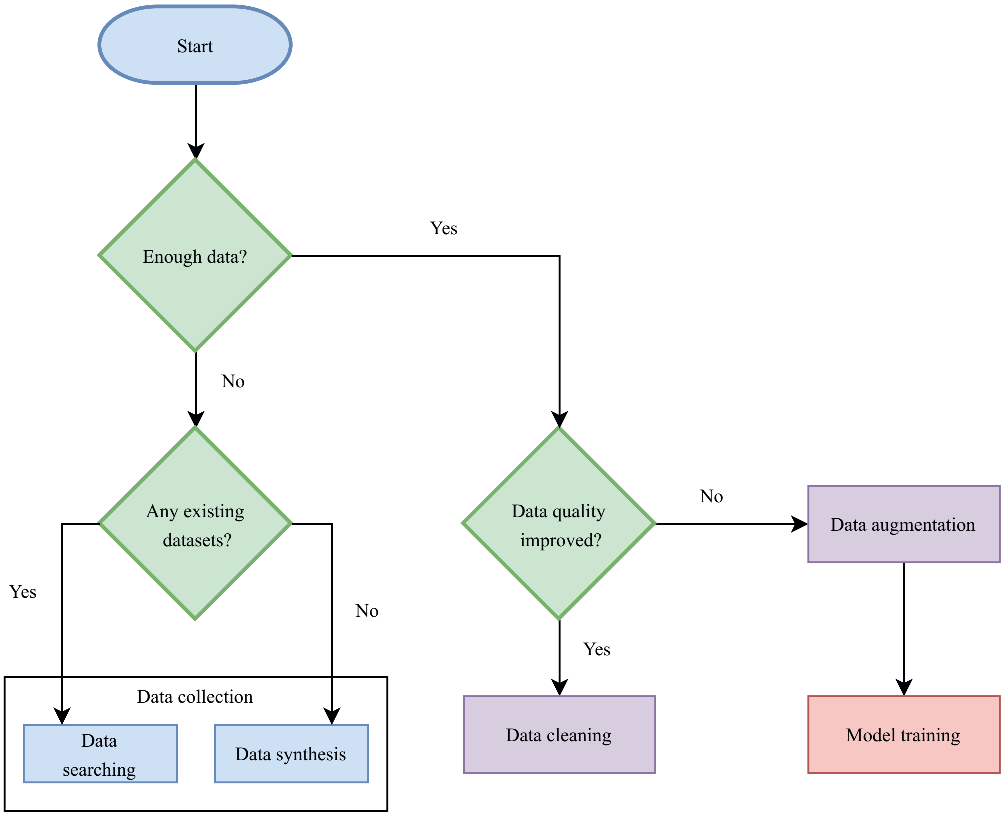

Fig. 2. Image preparation techniques.

4.1.1. Data searching

There is an enormous amount of data we find as we search on the internet. However, the issue lies in encountering unwanted data that is irrelevant to the projects. There are several challenges associated with data searching in AutoML and NAS. One challenge is that the gathered data must be compatible with the specific task we aim to perform. For instance, the data should include images and their related labels if the task involves classifying images. Another difficulty is ensuring that the data is indicative of the work in real life which can be challenging in some circumstances. Researchers can utilize a variety of methods for data searching in NAS to get around these issues. Using open datasets that have already been vetted and tagged for particular purposes like ImageNet or CIFAR-10 is one strategy. These datasets have been demonstrated to be successful in identifying high-performing neural architectures and are frequently utilized in the NAS community.

4.1.2. Data filtering

While searching for data in the data repository, the obtained response may not precisely correspond to the data we are looking for. It is possible that the data obtained may include undesired, irrelevant and inconsequential information. Hence, this data must be filtered for further use. Removing and excluding data that does not fulfill the criteria or standards from the dataset is referred to as data filtering. It is recognized as one of the most crucial steps in data preparation for AutoML and NAS as we can achieve accuracy, relevancy and perfectness for our dataset. By employing various processes, we can effectively accomplish the task of filtering. It is necessary to remove unnecessary data to guarantee the precision and quality of the dataset employed for training and evaluating machine learning models, particularly those produced using neural architecture search.

Using keyword-based searches to reduce the dataset to only contain information pertinent to the task is one method of removing extraneous data. After using keyword-based searches, the dataset may present irrelevant data. To overcome this situation, additional filtering techniques may be used. For example, an image can potentially appear in search results for multiple categories. However, we can overcome this problem using the approach proposed by Krause et al. [60] which allows us to distinguish and separate inaccurate results caused by cross-domain or cross-category noise.

Re-ranking relevant search results and presenting them linearly based on keywords is another strategy as proposed by Vo et al. [61]. This technique can ensure that only the most relevant data is added to the dataset used for NAS and AutoML.

4.1.3. Data labeling

To achieve the highest accuracy with AutoML and NAS algorithms, having a high-quality labeled dataset is crucial. The data labeling process includes identifying raw data and adding one or more relevant and informative labels that help the machine learning algorithms make decisions. Incorporating human-in-the-loop (HITL) approaches in AutoML enables the involvement of humans in data labeling when necessary. In these situations, a subset of the data is labeled by humans and utilized to train a fundamental model. The rest of the data is then predicted using the model and the predictions are subsequently verified by human annotators. The model is then retrained using the resultant labeled data and the procedure is continued until the required level of accuracy is achieved.

Automated data labeling is essential for reducing the amount of manual labor required in the data labeling process. Roh et al. [61] proposed a comprehensive review of self-labeling methods in semi-supervised learning, including self-training [62], co-training [63] and co-learning [64]. These approaches aim to reduce the dependency on human annotators and enhance efficiency in the learning process. These methods leverage the unlabeled data to enhance the model's performance.

The proposed method by Yang et al. [65] involves assigning multiple labels to an image under two conditions: when the confidence scores of these labels are closely similar or when the label with the highest score coincides with the original label of the image. This can lead to a more diverse and informative training dataset. This iterative algorithm integrates model training and web data filtering in order to enhance the performance of the model specifically on the target dataset. The algorithm chooses a subset of web data that has potential value for training the model and refines the model using both the original labeled data and the selected web data. The iterative process continues until the model achieves a satisfactory level of performance on the target dataset.

Dataset imbalance stands as another frequent challenge encountered in many datasets, which is solved using the synthetic minority over-sampling technique (SMOTE). According to Adi et al. [66], class imbalance is a frequent issue in poverty classification due to the small number of individuals living in poverty compared to those who do not. The authors propose using SMOTE to oversample the minority class (individuals living in poverty) to achieve a balanced dataset and to enhance the performance of machine learning models employed for poverty classification.

4.1.4. Data synthesis

The process of creating new data samples depending on the ones that already exist is known as data synthesis. Simulators can be employed to develop synthetic data that resembles the characteristics of real-world data. For instance, in the area of self-driving cars, simulators can be used to produce abundant artificial data for testing and training machine learning models for object detection, navigation and other tasks. The OpenAI Gym [67] toolkit is widely utilized for generating diverse simulation environments, allowing developers to focus on algorithm design rather than grappling with the complexities of obtaining real-world data.

Simulators may also be used to assess how well machine learning models hold up under different conditions. Simulators, for instance, may be used to produce fictitious data to evaluate the resistance of machine learning models to assaults and other security concerns in the field of cybersecurity. This can enhance the model's general functionality and security by pointing out model flaws.

Another approach for generating synthetic data is generative adversarial networks (GANs) [68]. GANs can generate realistic images that can be used in a variety of applications, such as art design and gaming. For example, NVIDIA's StyleGAN can generate high-resolution images of human faces that are nearly indistinguishable from real photos. GANs can generate realistic videos by generating individual frames and then combining them into a sequence. This can be used in applications such as video editing and special effects. Li et al. [69] generated videos from text with the help of a text encoder, a video decoder and a GAN-based discriminator to produce videos that match a given input text description. Text and music generation can also be possible with the help of GANs. Guy et al. [70] propose a method that generates quality text and is also capable of distinguishing between real and fake text.

4.2. Data cleaning

Data cleaning is a fundamental step in data processing, encompassing various techniques such as error detection, data normalization, duplicate removal and missing value imputation. It guarantees the precision, uniformity as well as dependability of the data leading to enhanced excellence and dependability in subsequent analyses of machine learning models. Therefore, this data has to be filtered and cleaned for further use. Several data-cleaning approaches [71] must be applied to clean the data if necessary. Examples of traditional data cleaning methods could be Potter's wheel [72] and IntelliClean [73]. However, manual data cleaning approaches require the involvement of human experts and are also time-consuming and labor-intensive. Hence, Chu et al. proposed "Katara" [74], a data-cleaning tool that automates the data-cleaning process and also reduces the time to clean the dataset.

However, as every aspect of ML is automated, data cleaning is also being updated from manual to automated. Katara uses the co-ordination of rule-based and machine-learning-based approaches to automate the data-cleaning process. The platform provides a visual interface for users to specify cleaning rules which are then applied to the data using a combination of machine learning and data mining algorithms. On the other hand, this method does not provide excellent efficiency. To overcome this problem and improve efficiency Krishnan et al. [75] propose dividing the big dataset into small, non-overlapping subsets and cleaning each subset independently. The percentage of dirty records in each subgroup and the percentage of clean records required to attain the specified level of accuracy are then estimated by the system using statistical sampling. Nevertheless, this method also necessitates the involvement of a human specialist to design the specific data-cleaning operations that should be applied to the dataset. Hence, it is not a fully automatic process. Boost-Clean [76] is designed to clean data automatically by detecting and repairing errors in the training dataset before training the machine learning model. Boost-Clean identifies potential errors in the training data using a set of pre-defined error detection rules. The system then uses a combination of statistical modeling and rule-based repairs to correct the errors. This method is designed to be flexible and can handle a variety of different types of errors including missing values, outliers and inconsistencies. AlphaClean [77] can also automate the cleaning process by automatically generating data-cleaning pipelines that can be easily executed by non-experts.

In order to find any potential flaws or inconsistencies, AlphaClean first analyzes the incoming data. The system then generates a list of potential cleaning pipelines made to fix these issues. The pipelines are created by AlphaClean using a combination of machine learning and rule-based techniques. AlphaClean may also learn from user comments to increase the caliber of the pipelines it generates. Several available approaches to clean data include DeepClean, Trifacta, OpenRefine, Data Wrangler and Google Refine. All the methods of data cleaning that are mentioned above actually work on a fixed dataset. If the dataset changes the whole process has to be repeated again which is time-consuming. As the real world makes numerous numbers of data every day the continuous process of data cleaning has gained significant importance in the machine learning field. Ilyas et al. [78] propose a method that allows for dynamic monitoring and cleaning of data as it is generated. The author suggests a framework for continuous evaluation that has a number of essential elements such as a data quality model, a cleaning plan generator, a cleaning executor and a feedback loop. The framework can be modified to meet the unique requirements of various applications because it is made to be flexible.

4.3. Data augmentation

Data augmentation (DA) has become a popular technique in machine learning for improving the effectiveness of the models, especially when the size of the available dataset is limited.

The objective of data augmentation is to expand the training dataset's size and enhance the model's quality by reducing overfitting

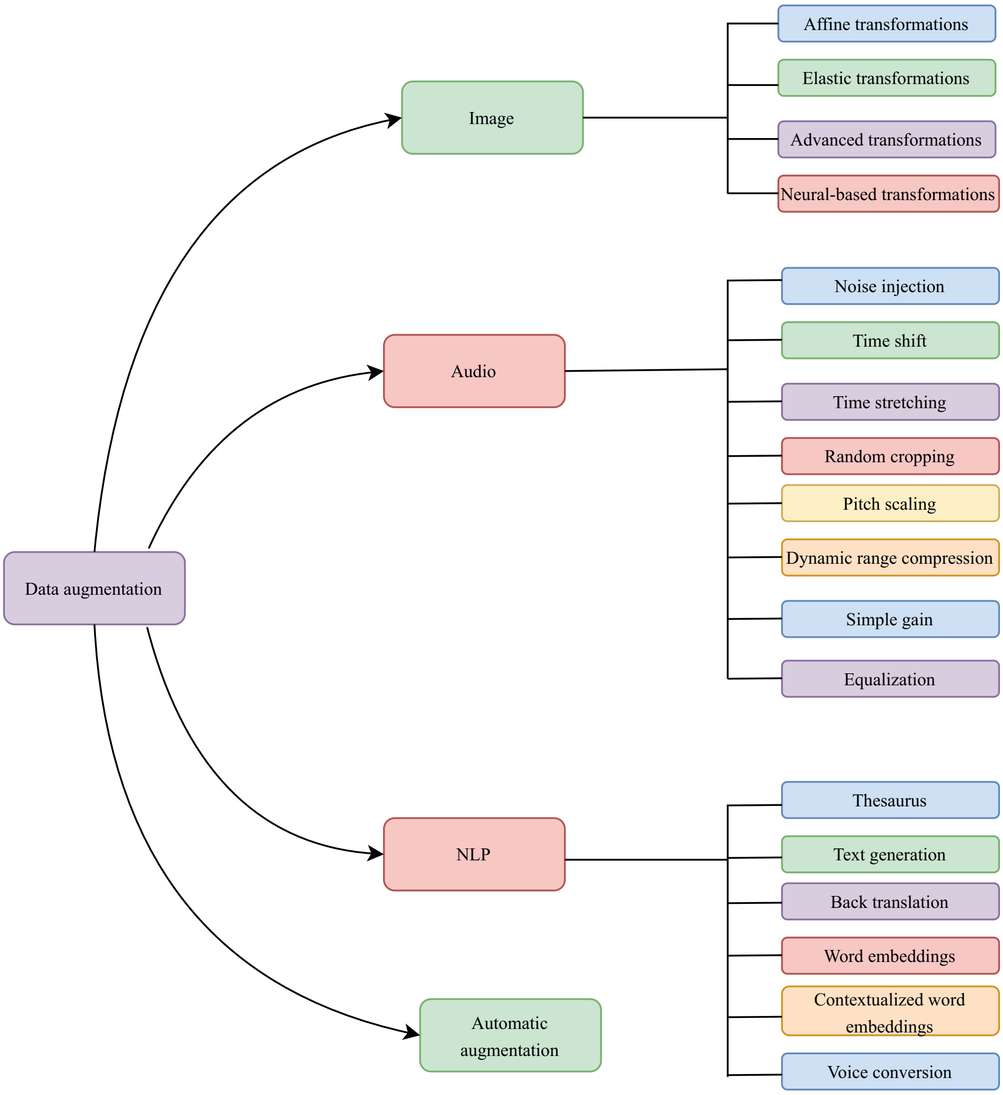

Fig. 3. Classification of data-augmentation techniques.

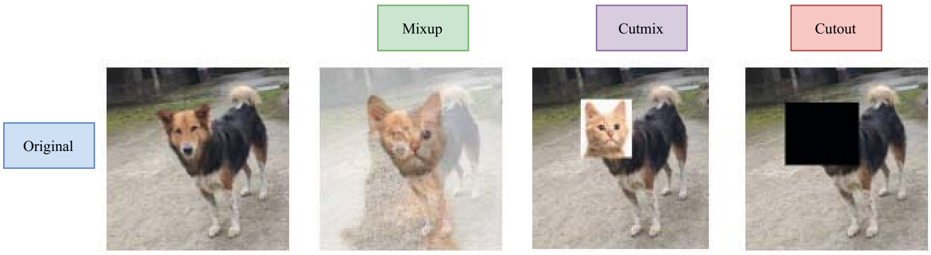

Fig. 4. Image augmentation techniques.

and improving generalization capabilities. Data augmentation can be applied to various types of data such as images, audio, text and tabular data. Figs. 3 and 4 show the different DA techniques on different types of data [79].

For image data, several DA techniques can be applied. The affine transformation includes scaling, rotation, translating and cropping an image while preserving the parallel lines. Affine transformations are generally applied for image processing and computer vision tasks as they can change the perspective of an object or scene. Elastic transformations, on the other hand, include operations like brightness shifting, contrast shifting, channel shifting and blurring. Elastic transformations initiate applying a smooth, nonlinear deformation to an image, which can stretch or compress the image locally. The advanced transformations encompass techniques such as random erasing, image blending, cutout [80], mixup [81], cutmix [82], etc. There are several open-source image transformation tools available for example, Torchvision [56], ImageAug [57], Albumentations [83], OpenCV [84]. Neural-based transformation can be categorized into three main categories: adversarial noise [85], neural style transfer [86] and GAN technique [87] in Fig. 5.

Textual data augmentation techniques can expand the variety of text and enhance training data. Some commonly used techniques include synonym replacement: Which involves replacing words with their synonyms while preserving the meaning of the sentence; random replacement: Inserting new words or phrases at random places; random deletion: Deleting words randomly from the sentence; back translation: Translating the text to a different language and then back to the original language; random swap: Shuffles the positions of words in a sentence; OCR noise: Introduces errors similar to those made during optical character recognition. Wong et al. [88] suggested two approaches, namely data warping and synthetic over-sampling, to generate additional training examples. Jiaao et al. [89] proposed a method named MixText for improving semi-supervised text classification tasks. MixText employs a linguistic interpolation approach in the hidden space of a neural network model. Yu et al. [90] introduced a technique that utilizes back-translation for DA with the aim of enhancing reading comprehension. The authors address the challenge of reading comprehension by designing a model that can effectively capture both local and global dependencies in text. Several open-source libraries are available for textual augmentation like TextBlob, TAugment, easy data augmentation (EDA), etc. NLPAug [58] an open-source toolkit, integrates a wide range of augmentation techniques catering to both textual and audio input data. The techniques described above are all manual DA techniques. It is important to note that human involvement is still required to select the appropriate augmentation operations and define specific DA policies appropriate to the specific tasks. This process demands significant expertise and time investment.

AutoAugment [91] is a groundbreaking study that utilizes reinforcement learning to automate the search for optimal data augmentation policies. However, the original implementation of AutoAugment suffers from inefficiency, requiring approximately 500 GPU hours to complete a single augmentation search.

To enhance search efficiency, several subsequent works have proposed improved algorithms with different search strategies for automating the search for optimal data augmentation policies. These strategies include gradient descent-based approaches [92,93], Bayesian-based optimization [94], online hyperparameter learning [95], greedy-based search [96] and random search [97]. In addition to these search-based methods, LingChen et al. [98] introduced a search-free DA method called UniformAugment. This method operates under the assumption that the augmentation space exhibits approximate distribution invariance, thereby eliminating the requirement for an explicit search process. These advancements in search strategies and the introduction of search-free methods have aimed to improve the efficiency and effectiveness of finding optimal DA policies, reducing the computational resources and time required for the augmentation search process.

5. Feature engineering

In the machine learning pipeline, feature engineering is a crucial stage where we convert raw data into a format that machine learning algorithms can use. For the models to perform better and be easier to understand it entails developing new features, choosing pertinent features and modifying current ones [99]. A more thorough search and optimization of the feature space are made possible by

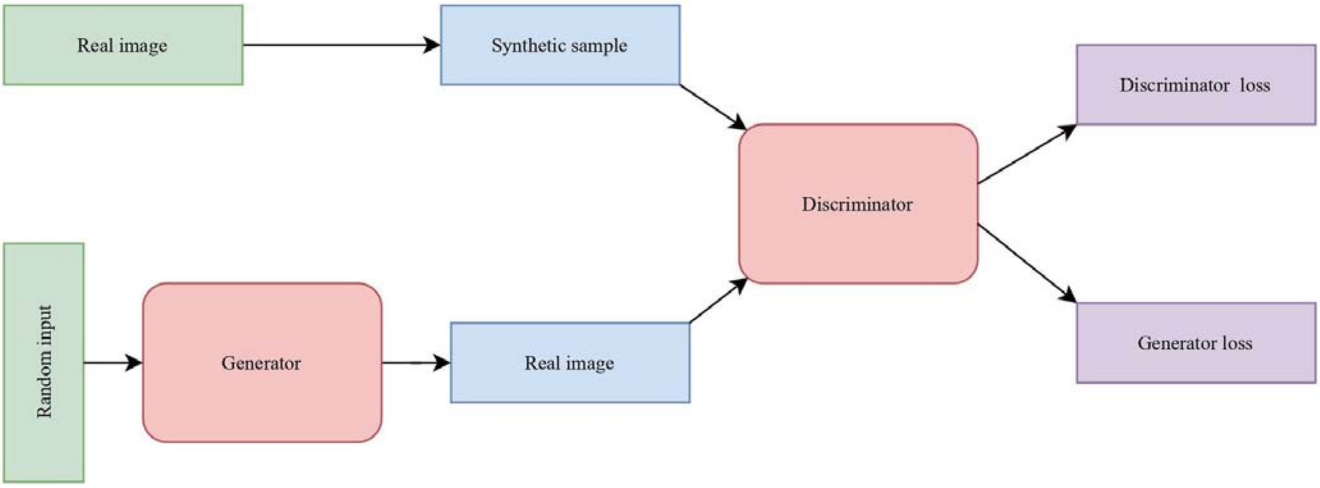

Fig. 5. GAN technique for image augmentation.

the platform's ability to automatically investigate and experiment with various feature engineering strategies [100].

Automated feature engineering is a part of the whole process offered by platforms that employ AutoML. To determine the best-performing model, these platforms test out different combinations of machine learning algorithms, feature engineering approaches and data preprocessing procedures. Through the use of AutoML tools, optimization methods like evolutionary algorithms or Bayesian optimization are used to search the universe of viable feature engineering transformations and combinations [101,102]. The integration of feature engineering into the AutoML workflow is depicted in Fig. 6 as follows:

It's crucial to keep in mind that the precise impact of feature engineering varies based on the dataset, issue area and baseline feature quality. Iterative for experimentation and validation are necessary for effective feature engineering in order to make sure the engineered features offer valuable insights and enhance model performance.

To assess the effects of dimensionality reduction, common metrics are as follows:

- **Reduce the number of characteristics:** The dataset's decreased dimensionality.
- **The variance was described:** Using methods like principal component analysis (PCA), the proportion of the data's variation can be explained by the smaller collection of attributes.
- **Information storage:** Retaining significant patterns or correlations in the data with a smaller feature set.

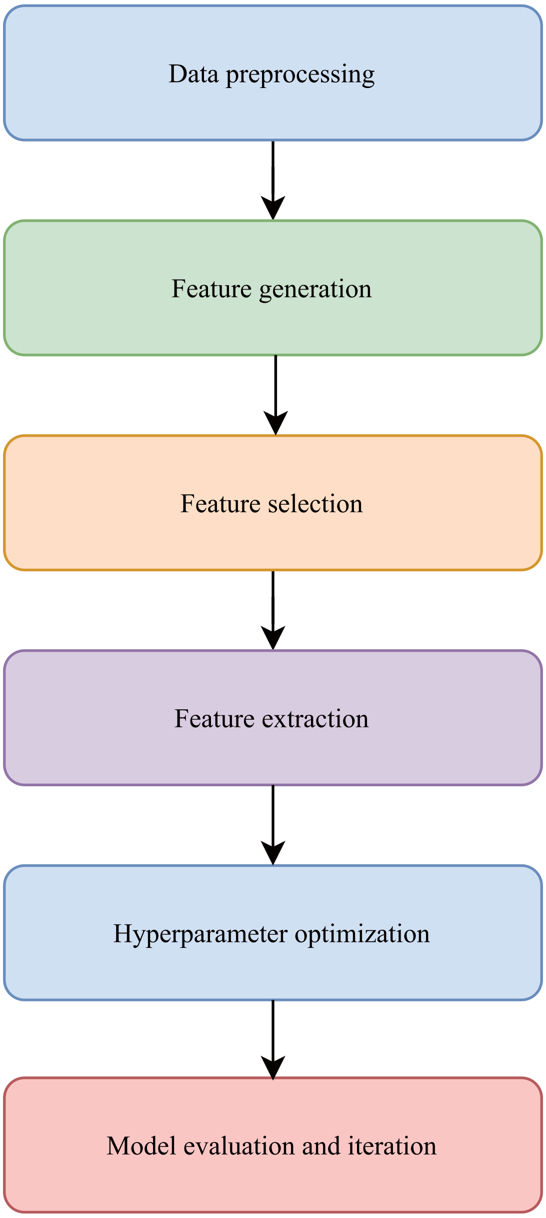

Fig. 6. This figure illustrates how feature engineering has been included into the AutoML approach.

5.1. Feature selection

A subset of pertinent features is chosen from a broader pool of accessible features through feature selection. Putting the most critical features front and center will boost model performance, decrease complexity and increase interpretability. Feature selection may lessen noise, overfitting and the curse of dimensionality by choosing the most pertinent characteristics [103]. This frequently results in greater generalization and predictive capability for machine learning models. In particular, for large dataset or computationally demanding methods working with a reduced collection of features might minimize computing resources and training time.

The AutoML pipeline includes a critical phase called feature selection. The goal of AutoML systems is to automate the entire machine learning process which includes data preparation, feature engineering, model selection and hyperparameter tuning. To locate the most informative features, AutoML systems make use of a variety of feature selection algorithms [104]. These strategies may combine filter methods, wrapper methods, embedded methods and so on. Depending on the AutoML platform's available algorithms and the particular needs of the challenge the feature selection strategy may be chosen.

By picking the most useful feature at each stage, the sequential forward selection method creates a feature subset sequentially. Though it is an easy and straightforward method, it does not necessarily provide the best feature subset as above in Fig. 7. To enhance the feature selection procedure, it is also possible to investigate different iterations of this approach, such as sequential backward selection (SBS) and sequential floating forward selection (SFFS). There are benefits and drawbacks to each feature selection strategy. The selection of an algorithm is influenced by a number of variables including the dataset's properties, the problem's specifications, available computing power and the final model's desired interpretability [105]. Finding the best solution for a particular issue frequently requires testing out a variety of algorithms and comparing their effectiveness.

AutoML's iterative feature selection process enables the investigation of various feature combinations and selection strategies, continuously honing the feature subset to improve model performance. The most useful features are found, overfitting is decreased and the prediction power of the model is enhanced through this iterative process [106]. It is crucial to keep in mind that the precise number of iterations and the stopping criteria for the iterative process may vary based on elements like the existing computational resources, time restraints and the required degree of model performance.

5.2. Feature extraction

The process of turning raw data into a collection of meaningful and representative features that capture the pertinent facts for a given activity or problem is known as feature extraction [106]. In order to make a subset of features from the original data that are more insightful, less redundant and appropriate for further analysis or modeling, a subset must be created or chosen. Reducing the dimensionality of the data while preserving essential information is the aim of feature extraction. It aids in streamlining data representation, enhancing computing effectiveness and improving the effectiveness of machine learning algorithms. Scale-invariant feature transform (SIFT) and histogram of oriented gradients (HOG) are two feature extraction techniques that are used to extract descriptive qualities from images.

Methods like autoencoders, bag-of-words (BoW) and term frequency-inverse document frequency (TF-IDF) are used to extract attributes from textual data. To extract characteristics from signals, such as audio or biological data, methods like fast Fourier transform (FFT) and wavelet transform are utilized [107]. Domain-specific feature extraction should take into consideration the features of the data and the task's needs. Data pretreatment features for handling missing values, encoding categorical variables and initial data cleaning are frequently included in AutoML platforms. This stage makes sure the data is formatted properly for feature extraction. For the purpose of creating fresh features from the original data AutoML systems make use of several automatic feature extraction approaches. These methods can include wavelet transforms, Fourier transforms, dimensionality reduction algorithms (such as PCA, LDA) or other domain-specific methods illustrate in Fig. 8. The goal of the automated feature extraction method is to identify significant patterns, cut down on repetition and provide a more insightful feature representation. By automating feature extraction, AutoML frees people to concentrate on more complex activities like analyzing the extracted features comprehending the underlying patterns in the data and making defensible judgments based on the transformed feature representation [108].

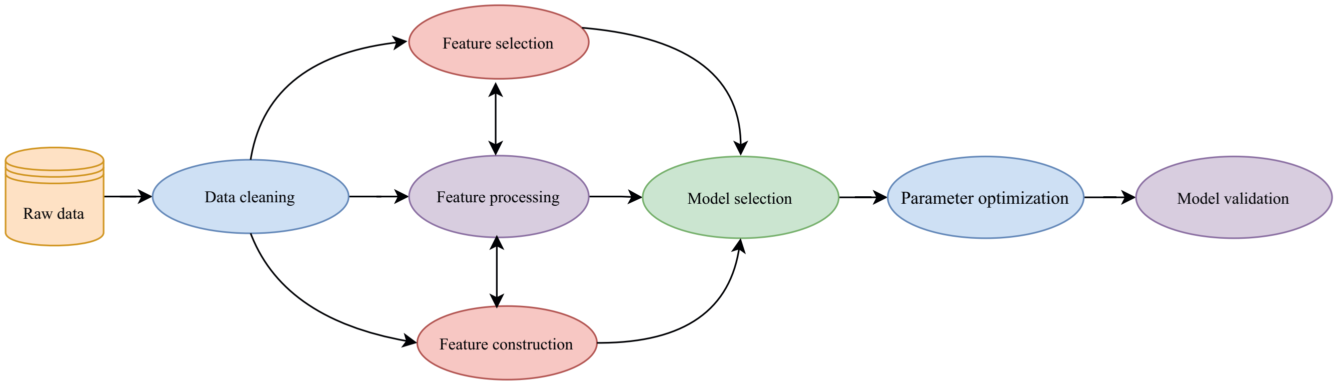

Fig. 7. Post-data cleaning for the ML pipeline is automated by AutoML capability. The pipeline's specific automated processes include model selection, parameter optimization, feature selection and preprocessing.

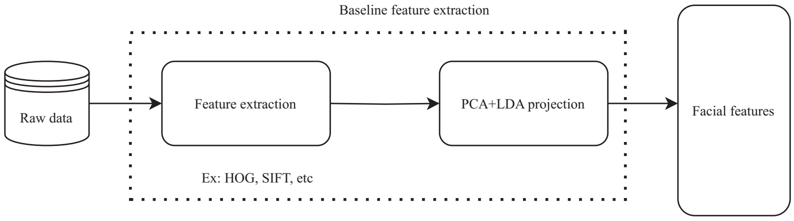

Fig. 8. Image-based feature extraction.

5.3. *Meta learning*

A branch of machine learning called "learning to learn" or "meta-learning" is concerned with creating frameworks and algorithms that can learn from different tasks or domains in order to themselves become better learners [109]. Meta-learning's objective is to make it possible for models to learn from their mistakes and build on their existing knowledge in order to better learn and adapt to new tasks.

Beyond this paradigm above in Fig. 9, meta-learning trains models to efficiently learn from a variety of tasks or datasets, enabling speedy adaptation and strong generalization to new, untried tasks. Model selection, tuning of hyperparameters and feature engineering are processes that AutoML automates and optimizes by utilizing meta-learning approaches to learn from prior machine-learning tasks or datasets [110]. Meta-learning in feature engineering may be used in a variety of fields including image processing, time series analysis, natural language processing and other areas where efficient feature engineering is essential for performance enhancement. The trained meta-learning model can produce suggestions or transformations for fresh data during the application phase based on its comprehension of useful attributes. The model can recommend certain feature engineering strategies, transformations or feature combinations that are likely to enhance performance on the intended job.

The process of feature engineering may be made more automated and adaptable by using meta-learning approaches. The meta-learning models can capture the correlations between features, transformations and task performance which allows for more effective feature engineering on tasks [111]. By automatically looking for the best feature representations and transformations, can save time and effort. By learning from previous feature engineering jobs, meta-learning in AutoML may automate the feature engineering process. The meta-learning component can find effective feature representations, transformations or combinations that work well on tasks or datasets that are comparable. Then, based on this information, suitable characteristics may be automatically generated for future assignments.

6. Model selection

Model selection is the most important part of NAS. We divided this into two stages-search space and architecture optimization.

6.1. *Search space*

For design model in AutoML Search space helps to identify optimum architecture, minimize complexity and boost the efficiency of architecture search. The search space determines which architectures can be theoretically represented. The NAS search space encompasses a set of fundamental neural network operations and the possible connections between nodes to construct robust network

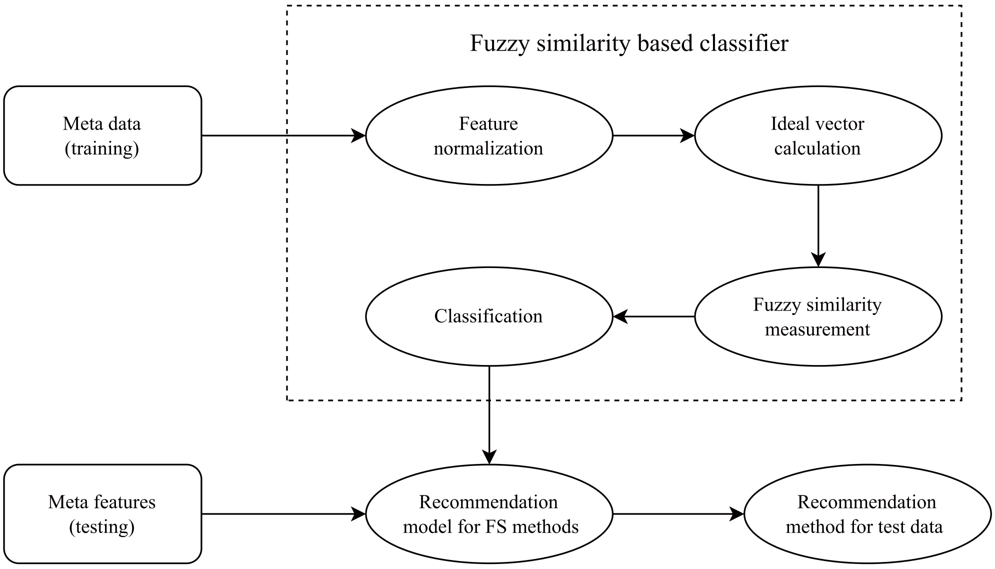
Fuzzy similarity based classifier
Meta data (training)
Feature normalization
Ideal vector calculation
Fuzzy similarity measurement
Classification
Meta features (testing)
Recommendation model for FS methods
Recommendation method for test data

Fig. 9. Overall procedure of meta-learning approach.

topologies. The most commonly used search space in NAS such as cell-based search space [112,113], entire structured search space [114–116], hierarchical search space [117–122], morphism based search space [123–125].

For large amount of dataset hierarchical search space perform better than cell-based search space, and entire structured search space. In this section, we delve into the hierarchical search space providing a comprehensive discussion on its characteristics and implications. A hierarchical genetic representation system [122] inspired by the modularized design pattern frequently used by human specialists and an expressive search space that accommodates complicated topologies. First, they present a flat architecture representation and then they construct a hierarchical architecture. In this architecture, there are multiple patterns distributed across different levels of the hierarchy.

$$O_m{}^{(i)} \;=\; assem(H_n{}^{(i)}, o^{(i-1)}), \nabla i = 2, \cdots, I. \tag{1}$$

Here in equation (1), I is hierarchy levels, ith level contains M_i motifs, highest-level $i = I$, network structure H_n^l. They recursively use $o_n^{(l)}$, nth motif in level l according to its structure. Level-one cells are the building blocks of level-two cells and can be some simple operations like 1 × 1 and 3 × 3 convolution and 3 × 3 max-pooling. Fig. 10 shows a basic hierarchical architecture.

Most super-resolution (SR) models follow a hierarchical structure [126]. Design a hierarchical SR search space including cell-level and network-level search spaces named hierarchical neural architecture search (HNAS). HNAS utilizes two types of cells to achieve its objectives. The normal cell manages the model's capacity and spatial resolution of feature maps while the up-sampling cell enhances the resolution. Mobile neural architecture search (MNAS) introduces a brand-new factorized hierarchy search space for CNN models where the operations and connections of each block are independently searched [119]. Global local image transformer (GLiT) represents Hierarchical neural architecture search consists of two primary phases: firstly, searching for the optimal distribution and secondly, finding the precise architecture to establish the distribution of the global and local sub-modules within each block [127].

6.2. Architecture optimization

The procedure of selecting the most favorable neural network architecture for a specific dataset is known as architecture optimization in deep learning. Designing a neural network for a new dataset requires experts with advanced proficiency in deep learning [128]. Deep learning architecture optimization uses an optimizer to modify the neural network features such as learning rate and weight. This optimizer improves precision and reduces overall loss.

Deep learning models commonly have millions of attributes which make choosing the correct weights for a model extremely challenging [129].

Some widely used deep learning architecture optimization algorithms are:

Gradient-based optimization: Gradient-based optimization is the method of iteratively establishing the upper or lower bound of a function by progressing in the direction of the steepest descent or ascent. Some common gradient-based optimization algorithms include gradient descent, gradient ascent and min-max gradient.

Bayesian optimization: Bayesian optimization is a progressive design method for comprehensive optimization on black-box systems that considers the lack of operational structures.

Reinforcement learning: Reinforcement learning as a specialized branch of machine learning is engrossed in how intelligent agents are expected to behave and take actions to maximize the cumulative reward in an environment. In reinforcement learning, the agents are rewarded upon action on desired behaviors and penalized for negative behaviors.

Evolutionary algorithm: Evolutionary algorithm (EA), categorized under evolutionary computation, is an optimization algorithm. By utilizing mechanisms such as reproduction, mutation, recombination and selection, the evolutionary algorithm (EA) emulates the natural selection process to discover problem solutions.

6.2.1. Evolutionary algorithm

Evolutionary algorithm (EA) are optimization techniques inspired by the principles of natural evolution. An array of prospect solutions is randomly created in order to maximize the quality function. The quality function is subsequently implemented in the problem

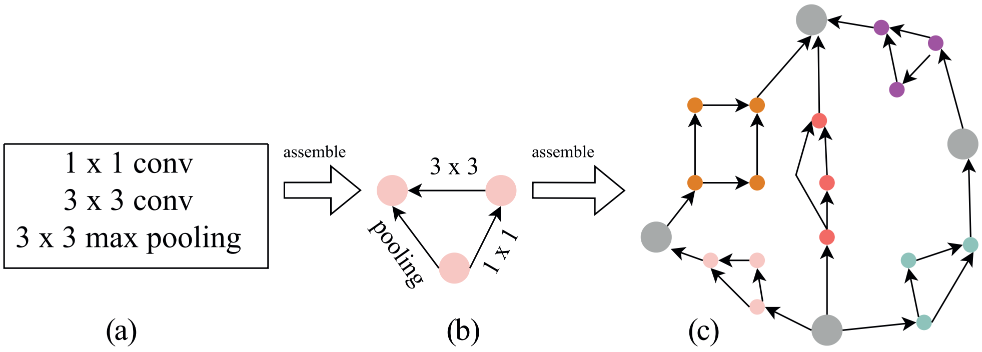

Fig. 10. (a) Demonstrates how level-1 primitive operations are built into a level-2 motif. (b) Illustrates how level-2 motifs are further integrated into a level-3 motif (c).

domain as an abstract fitness function. Some superior candidates are selected for the following generation based on their fitness function. Recombination which is depicted by the binary operator and/or mutation processes is applied to attain this after which a new set of candidates based on their fitness function is generated and this process is iterative [130].

The genetic algorithm (GA) is one of the most widely used and oldest optimization approaches. It shows the most intelligible mapping on a computer of the natural evolution process [130,131]. A population of individuals or chromosomes is taken to characterize prospective solutions to the problem in GA. The objective function defines the solution to the problem on the basis of how well a chromosome fits the objective function. There are three operators in GA: selection which defines the process of creating a new population on chromosomes according to the fitness values of the prior generation; mutation which is the process of changing values of particular genes at random and crossover, typically performed during the genetic algorithm process involves exchanging segments of chromosomes between selected individuals for the purpose of crossover [131].

EAs have a few fundamental operations which are: random initialization, selection, mutation and crossover. These operations and relevant parameters such as the count of neurons in each layer, the number of layers in architecture, activation function, learning rate and dropout rate must be specified along with implemented carefully in the automated process [132].

6.2.2. *Reinforcement learning*

Reinforcement learning (RL) is an area of study in machine learning that focuses on how an agent learns and adapts in an unfamiliar environment through interactions. The learning is usually guided by a particular objective in a “trial-and-error” approach. The agent receives a reward or penalty as feedback from the environment based on its actions. Depending on the feedback, the agent collects experience as well as knowledge of the environment and trains itself to reach its goal efficiently [133]. Fig. 11 shows how an agent interacts with the environment based on the rewards, policies and learning algorithm.

The primary model of RL entails an intelligent agent perceiving the environment through interactions and selecting actions that yield the highest reward value. The responsive interface of the intelligent agent and environment comprises reward, action and state. Every time when the RL system interacts with the environment, at first it receives the inputs from the environment state E_1 and then in accordance with the central interface process, the output of action Q acts the environment. Consequently, the environment acknowledges the action and transitions to a new state E'. The input of the new state E' gets accepted by the system and feedback signal Z is obtained as either reward or punishment of the environment to the system. The main goal of an RL system is to learn a strategy for action $\Pi E \to Q$. The strategy allows the action for the system to acquire the biggest accumulative reward value from the environment, this can be defined as the equation (2):

$$\sum_{i=0}^{\infty} \gamma^i Q_{t+i} , \ (0 < \gamma \le 1). \tag{2}$$

In the equation (2), γ denotes the discount factor. The primary notion of RL is that if any action of a system generates a positive reward for the environment, the system will enhance the trend, otherwise, the system will eliminate this trend [134].

When using the RL pipeline, it is mandatory to specify the reward function first. It might also be necessary to select what should be provided to the agent as an observation. Selecting hyperparameters is another challenge considering the influence of hyperparameters such as the time scale of the return γ, exploration trade-off, etc. in the RL algorithms. Usually, most of the systems can be placed in combination with an inner loop consisting of a standard RL pipeline and an outer loop for optimizing the agent arrangement [135]. To automate the RL pipelines, the hyperparameters and functions will need to be updated according to the positive or negative feedback of the environment based on the agent's actions.

6.2.3. *Bayesian optimization*

Bayesian optimization (BO) is a method designed to address challenges encountered in optimization problems such as high dimensionality, mixed parameter types, constraints, etc. It exclusively supports point-wise evaluations without providing function derivatives and lacks a closed-form representation, earning it the designation of a black-box method [136].

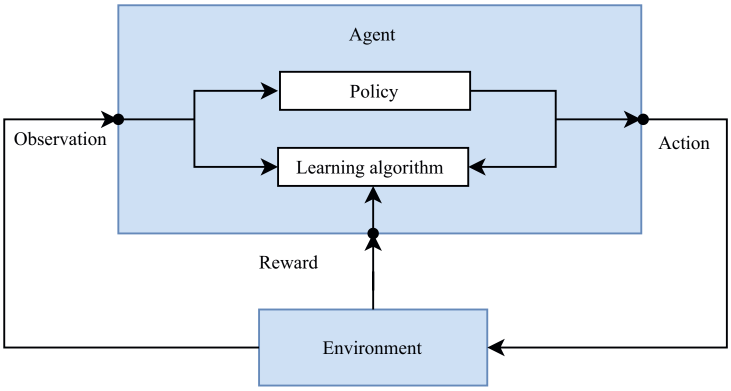

Fig. 11. Reinforcement learning agent and environment interaction.

The Bayesian optimization algorithm (BOA) works in such a way that prior information is not necessary. The first population of strings is randomized by BOA. A set of promising strings are chosen from the current population using any selection method. The selected set of strings is used to construct a Bayesian network. For maximizing the value of the implemented metric any metric can serve as an indicator of the quality of the networks and to search over the networks any search algorithm can be used. Joint distribution encoding techniques are used to generate new strings which are added to the old populations and some of the old strings are replaced by some of the new ones [137]. Bayesian optimization incorporates two fundamental methods.

Gaussian process (GP) models the epistemic uncertainty σ_t (y) and the predicted mean μ_t(y) for any point y in the input space, given a set of observations in equation (3),

$$P_{1:t} = \{(y_1, z_1), (y_2, z_2), \cdots, (y_t, z_t)\}. \tag{3}$$

At time t, y_t is the process input and z_t in the corresponding output.

Acquisition function (AF) deliberates the most favorable situation for the next demonstration, based on the uncertainty σ_t (y) and predicted mean μ_t(y) [136].

The covariance function k (y, y′) and mean function m (y) specify a GP completely in equation (4):

$$f(y) \sim \text{GP}(m(y), k(y, y')). \tag{4}$$

BO is excellent as an optimization system but to automate it, hyperparameters need to be selected carefully. It needs probabilistic surrogate models such as GP for every iteration. In every iteration, AF is also created from the GP or other such models. The determination of a probabilistic surrogate model and the AF is a vital issue in automating BO [138].

6.2.4. Gradient-based optimization

Gradient-based optimization works by moving in the direction of the positive gradient for maximization or the negative gradient for minimization by calculating the gradient of the function concerning its parameters. The step size is determined by a learning rate parameter. In Fig. 12 shows the Gradient-based optimization.

Gradient descent: Gradient descent is a method of finding the local minimum determined by the learning rate by minimizing an objective function f(y) which is defined by a model's parameters $y \in R$. The method works by modifying the model parameters in the direction counter to the gradient of the objective function $\nabla_y f(y)$ (which is a vector that signifies the direction of the utmost rate of increase of the function f) with respect to the parameters [139].

By following the negative gradient in equation (5), the function can be minimized with y_0 denoting the starting point and α denoting the positive distance moved in the direction of the negative gradient, the new y_1 will be considered as:

$$y_1 = y_0 - \alpha \nabla f(y_0). \tag{5}$$

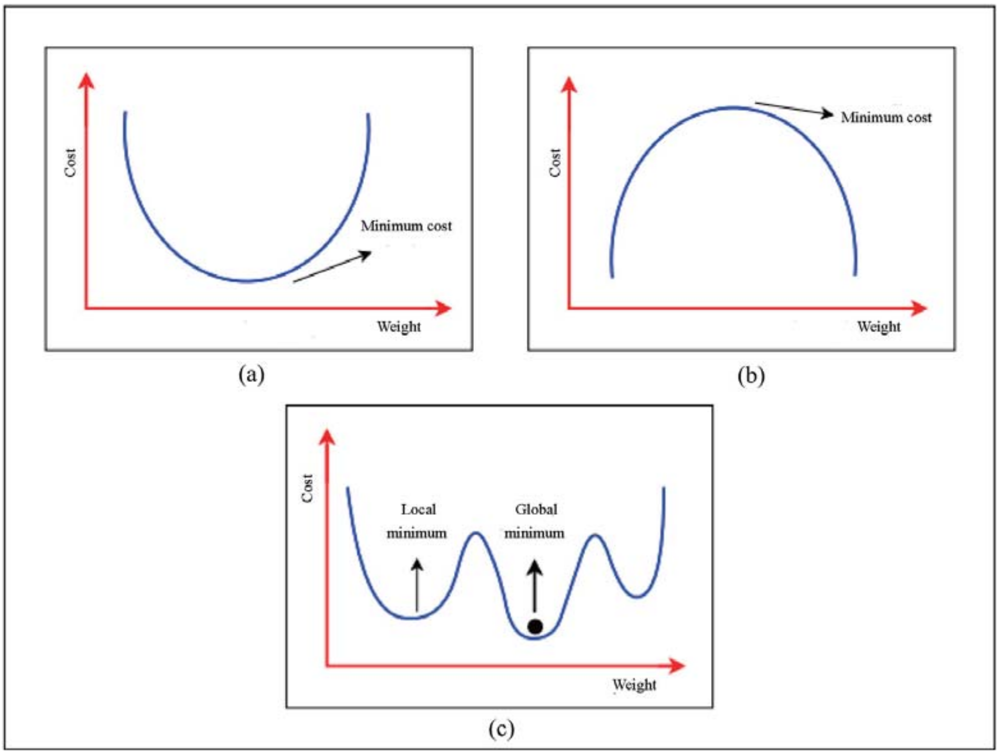

Fig. 12. Gradient-based optimization. (a) Gradient descent. (b) Gradient ascent. (c) Gradient min-max.

The above formula can also be written turning y_n into $y_{(n+1)}$ in equation (6):

$$y_{(n+1)} = y_n - \alpha \nabla f(y_n). \tag{6}$$

The process starts with the initial guess y_0 and after many iterations improving gradually it finds a local minimum. Gradient descent works in such a way that determining a local minimum assures the finding of a global minimum in convex scenarios. Nevertheless, non-convex situations frequently occur in neural networks which creates the obstacle of avoiding getting stuck in a local minimum that is not optimal [139].

Gradient descent optimization technique has been widely researched and used for decades [139,140]. The learning rate is the most critical hyperparameter in implementing neural networks [140,141]. If the learning rate is too large, the model will arrive at an inferior solution in a short amount of time and can even miss the optimal solution which increases the average loss [141]. Another core factor that has a significant impact on gradient descent is gradient estimation. Hence, the two common gradient descent optimization categories are, learning rate adjustment methods and gradient estimation optimization methods which respectively improve the stability and the gradient of the current iteration of a model [140].

Gradient ascent: Gradient ascent is an iterative method to identify a local maximum of a differentiable function by taking progressive steps proportional to the positive gradient of the function. The gradient ascent equation for function f characterizing a value to be maximized [142],

$$f\ (y)\ \sim\ \mathrm{GP}\ (m\ (y), k\ (y,\ y')). \tag{7}$$

In the equation (7), $y_{(n+1)}$ denotes the consequent position, the present position is denoted by y_n, α is the gradient ascent step size and Δf is the gradient of the current position [142].

Gradient min-max: Gradient min-max is an optimization method to solve the min-max problems. The objective of a min-max problem is to identify the minimum value of a function concerning a specific set of parameters while simultaneously maximizing it regarding another set of variables. This type of problem arises in game theory and machine learning [143].

In the gradient min-max optimization, parameters are iteratively updated using gradient ascent and descent steps. The descent step minimizes the function with respect to a set of parameters while the ascent step maximizes the function with respect to another set of variables. There are several variations of this method, including simultaneous gradient ascent and descent where both sets of parameters are updated concurrently and alternating gradient ascent and descent where the sets of parameters are updated alternately [144]. Fig. 12 shows cost versus weight graphs of the mentioned gradient-based optimization algorithms. When making the process automated through the use of AutoML methods, a modification for the hyperparameters can be determined. The primary objective of accelerating hyperparameter tuning is to reduce the number of training iterations, resulting in decreased training time. The learner's learning curve can be tracked as they advance through their training [145].

7. Model evaluation

Model evaluation is the most essential part of NAS for AutoML when checking effectiveness of the model on train and test dataset when a neural network has been created. The evaluation of NAS models can be a challenging task that demands meticulous attention to multiple factors. Evaluating a model is an expensive and lengthy process. Certain advanced evaluation methods can speed up the process, but they may compromise accuracy and precision. For example, ENAS is a new approach by distributing parameters between child models during the architecture search phase to NAS that significantly accelerates the process, reducing GPU hours by over 1000 times [114], DARTS+ uses an "early stopping" paradigm which shows the best result on CIFAR-10 and CIFAR-100 dataset with test error rate [146], Random search, with early stopping and weight sharing techniques, has been proven to achieve state-of-the-art results for neural architecture search (NAS) on Penn Treebank dataset and a highly competitive result on the CIFAR-10 dataset [147]. Here we introduce weight sharing and an early stopping strategy to accelerate the model.

7.1. Weight sharing

Weight sharing is incredibly useful for NAS's model evaluation process. Weight sharing technique is use to reduce computational cost of model in AutoML. FairNAS uses uniform sampling and multiple back-propagation procedures are involved in a single supernet step, which is followed by a single parameter change [148], shared convolutional kernel parameters method is used by Single-Path NAS to cut the cost of the search [149], construct a parameter sharing to reduces the computation cost [150]. AutoHAS use of weight sharing across all architectures significantly improves accuracy on ImageNet and achieves better accuracy with less compute cost such as random search or Bayesian methods [151]. Using shared parameters between convolutions with different numbers of input/output channels has been found to outperform random search for ImageNet. Each time, one of the six paths is chosen at random, and the shared model weights are updated [152]. To operate efficiently on big networks, share the weights of convolutional layers between residual blocks when using convolutions [153], and use weight sharing NAS to recommender systems domain [154]. Different layers frequently become functionally comparable after training as a result of parameter sharing, allowing for their collapse into recurrent blocks [150].

7.2. Early stopping

To prevent overfitting in deep learning models, early stopping is utilized as a regularization technique. The early stopping algorithm avoids the overfitting problem by monitoring the error trend of the training set, validation set and test set during training. Fig. 13 shows

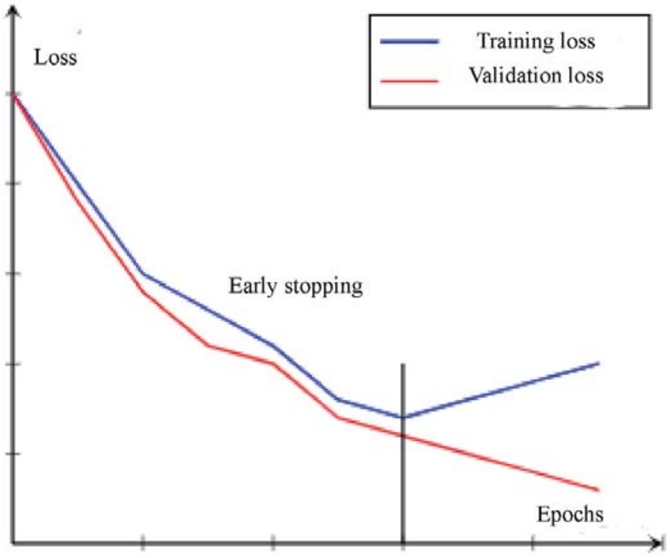

Fig. 13. Early stopping graphical visualization.

basic early stopping process. They introduce early stopping to accelerate the learning process [155]. MiLeNAS [156] found the optimal range of parameter numbers for their model architecture, they stopped the search process. By using this early stopping strategy, they were able to complete the search for the optimal architecture on the CIFAR-10 dataset. Using the early-stopping method can reduce search costs by up to 22.3% [157]. To avoid collapse DARTS introduced the DARTS+ "early stopping" procedure where they achieved 2.32% on CIFAR-10, 14.87% on CIFAR-100 and 23.7% on ImageNet test error [146]. When the number of search epochs becomes large HiNAS is often observed to collapse to avoid this they use early stopping search strategy [158].

8. NAS with AutoML

Neural architecture search (NAS) is an innovative approach in machine learning (ML) that searches for task-specific structures autonomously. As a result, the process of developing ML systems is significantly simplified thereby furthering the trend of democratizing ML. However, there is limited knowledge about the potential security risks associated with NAS which is concerning considering the growing utilization of NAS-generated models in increasingly critical domains [159]. NAS automates the design of neural network architectures. It entails determining the optimal architecture for the accuracy and cost of computation of a given task. AutoML automates the process of data gathering, feature design, model selection and optimization of hyperparameter settings for enhanced functionality. Automate the construction and optimization of neural networks by utilizing NAS and AutoML. NAS generates a set of potential architectures autonomously while AutoML trains and evaluates each design to determine the optimal solution. NAS-RL combines NAS and AutoML into a unified framework. This technique iteratively generates and evaluates neural network topologies using a reinforcement learning algorithm. NAS and AutoML decrease the amount of time and skill required to create high-performing neural network models. It requires computer resources and machine learning and deep learning expertise. When NAS and AutoML collaborate, the entire process of developing and optimizing neural networks for a specific task is automated.

Specifically, NAS generates a pool of potential architectures, followed by AutoML to train and evaluate each one in order to determine the optimal one. These methods automate the time-consuming tasks of preliminary data processing, constructing features, picking a model and hyperparametric standardization. This enables academics and data scientists to concentrate on model analysis and results rather than model construction.

DARTS, a gradient-based optimization method, aims to discover the optimal neural network architecture. DARTS discovers architectures with superior performance even with limited computational resources. Optimize model performance, development time, accessibility, scalability and resource utilization. These methodologies have the potential to facilitate and revolutionize machine learning. We use NAS to automate the process of designing optimal neural network architectures. NAS-Bench dataset is employed as a benchmark dataset for evaluating and comparing different NAS methods. The NAS-Bench dataset evaluates NAS techniques:

NAS-Bench dataset.

In this regard, NAS-Bench-101 [160] is a ground-breaking effort toward better consistency. Tabular data for 423,624 distinct neural network structures are provided, After being constructed from a preset graph-based search space, each of them underwent training and evaluation on CIFAR-10. After being formed from a predefined graph-based search space. As an extension of NAS-Bench-101, Dong et al. [161] created the NAS-Bench-201, which includes new search space findings on several other datasets (including CIFAR-10, CIFAR-100 and ImageNet-16-120) [162], and additional data for diagnostic purposes.

There is also a NAS-Bench for the work of the NLP team by Klyuchnikov et al. [46]. Scholars in the field of NAS use these datasets to validate the effectiveness and operational economy of their AO algorithms without having to conduct 20 cycles of recurrent training on selected designs. The CIFAR-10 classification dataset is commonly utilized by most NAS methods for training models due to its small image size, which facilitates rapid neural network training [57]. Moreover, when scaled up models that perform well on CIFAR-10 also demonstrate good performance on ImageNet. Hence, the fundamental NAS-Bench-101 is built upon CNN training on CIFAR-10 [6].

8.1. One-shot NAS

One-shot NAS methods generate all feasible designs from the overparameterized supernet embedded in the search space. Different architectures are represented by circles of varying sizes, but all architectures in one-shot NAS approaches hold the same relative importance [148]. There are two categories of one-shot NAS techniques, characterized by their distinct approaches to architecture optimization (AO) and parameter training. The optimization process can be performed in both a decoupled setting and an uncoupled setting [163]. One-shot NAS converts a search space (A) into a supernet (W_A) that swallows all candidates. All candidate architectures have inherent weights from the supernet, which is trained once. Only one neural network needs to be trained during the architectural search which reduces the duration of the search process [164]. The best architecture uses validation performance with supernet weights in equation (8):

$$\begin{cases} \min_{\alpha \in A} L_{val}(W_A \times (\alpha)), \\ s.t.\ \ (W_A \times (\alpha)) = \arg\ \min L_{train}(W_A(\alpha)). \end{cases} \tag{8}$$

NAS is fundamentally identical to conventional search techniques disregarding the fact that the architecture of the neural network or its hyperparameters is the target of the search. NAS can be defined in a manner suitable for beginners, including infants in equation (9):

$$\Lambda : DxB \rightarrow N, \tag{9}$$

where Λ specific deep learning algorithm is denoted by A, the dataset is denoted by D, the architectural search space is denoted by A and the model space is denoted by M. As a consequence of this, we as a species use a deep learning method to map the space for searching architectures to the space for building models [165].

The goal of operating the NAS is defined by the following equation (10):

$$\Lambda(\alpha, d) = \arg\ \min L(m_{\alpha,\theta}, d_{train}) + K(\theta), m_{\alpha,\theta} \in N_\alpha, \tag{10}$$

where L is the loss function, $d \in D$, $\alpha \in B$, is a parameter $m_{\alpha,\theta}$; $\theta \in N_\alpha$, $K(\theta)$ regularization method.

8.2. Zero-shot NAS

Neural architecture search is extensively utilized to systematically find the optimal neural network from several alternative architectures. Zero-shot NAS designs training-free proxies to anticipate architecture performance evaluations to save search time [166].

Algorithm Zen-NAS.

Require search area D, inference spending limit H, maximum depth J, the total number of iterations *E*, population size *N* for the evolutionary algorithm and a starting structure *Fo*. Ensure: NAS-designed ZenNet *F**.

Algorithm 1: Algorithm Zen-NAS

```
Initialize population $P = Fo$.
for $t = 1, 2, \cdots,$ T do
    Pick something at will $F_t \in P$.
    Mutate $F_e = MUTATE(F_e, D)$.
    if in the event that $f_e$ is larger than the inference budget or contains more than J layers, then
        Do nothing.
    else
        Get Zen-Score z = Zen($F_e$).
        Append $F_e$ to $P$.
    end if
    If $P > H$, then discard the network with the lowest Zen-Score.
end for
The highest Zen-Score network in $P$, $F^*$, should be returned.
```

This approach ranks candidate network designs without training to replace the expensive NAS training process with computation-efficient proxies [42,167,168]. Several studies estimate the expressivity of deep neural networks by quantifying the number of linear regions. It is also possible to find the best neural network architectures without training by utilizing zero-shot NAS and AutoML. Specify the various components of the structure, such as its layers, interconnections, activation functions, etc. Construct a predictor of future performance by leveraging the current architecture and historical data.

Train a subset of the architectural representations in the search space to establish a relationship between architecture and performance and assess the outcomes. Design system architectures for automatic machine learning searches. Viable choices for such searches include Bayesian optimization, reinforcement learning, genetic algorithms, and evolutionary algorithms. AutoML navigates the search space using model predictions as substitutes [169].

8.3. *Resource-aware NAS*

The earliest topologies of neural networks were developed by graduate students. Thereafter, dataset-specific structures have been algorithmically created by means of neural architecture search. To facilitate rapid searching, gradient-based NAS techniques have been created. Optimizing the search phase can happen quicker with gradient-based NAS solutions because the search is structured. The performance of neural architecture in the real world goes beyond only accuracy. This study incorporates resource costs into a gradient-based NAS framework [170]. Multi-objective neural architecture search automatically designs networks with high precision and efficacy for a variety of applications. Existing methods must sample multiple networks to instruct the search behavior of the controller. Provide a NAS architecture for dynamically sampling networks in accordance with the latency needs of connected devices and programs. To match the latency of the sampled network, layer-wise network sampling modifies layer sampling probability based on the remaining inference delay space. Thus, unnecessary network sampling is avoided and search efficacy is improved [171].

A convolutional neural network is trained to predict labels based on data from a target distribution [170]. A neural model is fitted to a distribution by locating the optimal network parameters that minimize the predicted prediction error on a dataset.

$$\theta^* = \arg\min[S(\theta) = W_{((z,t)\sim p\text{-}data}N_1(f(z;\theta),t)]. \tag{11}$$

In equation (11), S is the objective function, z are observations from the dataset, t is labeled from the dataset, *p-data* is the empirical distribution, N_1 is a prediction error loss function and f is the neural network parameterized by θ.

8.4. *DARTS*

As experts strive to discover cutting-edge neural network topologies, the popularity of algorithms automating architecture design is growing. Rather than searching a discrete set of candidate designs, it is preferable to render the search space continuous. This allows gradient descent to optimize the validation set performance of the architecture. In contrast to black-box search methods, gradient-based optimization is more data-efficient, allowing DARTS to compete with state-of-the-art approaches using significantly fewer computational resources [114]. The remarkable performance improvement achieved by these methods, reducing the cost of architecture discovery to just a few GPU days, can be attributed to their utilization of gradient-based optimization instead of non-differentiable search strategies.

Algorithm: DARTS – differentiable architecture search make a hybrid business out of it $p^{(k,l)}$ characterized by a set of $\beta^{(k,l)}$ along every border (k,l) while not converged do.

(1) Update architecture α by descending $\nabla_\beta L_{val}(x - \xi\nabla_w L_{train}(x,\beta),\beta$ ($\xi = 0$ if one was to employ a first-order estimate).
(2) Recalculate the weights w in descending order $\nabla_w L_{train}(x, \beta)$ learned information should be used to derive the final architecture.

8.5. *NAS performance comparison*

Several studies on NAS have put forth different forms of neural architectures, each tailored to specific circumstances. For example, some architecture variants exhibit superior performance but are more resource-intensive, while others are compact enough for deployment on mobile devices but have lower performance. Consequently, we will present results that are representative of each study. Furthermore, to ensure accurate comparisons, we employ precision and efficiency as the indices for evaluating algorithm performance.

The hyphen (−) indicates that the corresponding information is missing from the original document. RLA, EBA, GD and RS stand for the reinforcement learning algorithm, evolution-based algorithm, gradient descent and random search, respectively. SMBO stands for surrogate model-based optimization. Slightly different studies use distinct types of GPUs; therefore, we as a nation approximate the efficiency using GPU Days, which can be characterized as in equation (12):

$$\text{GPU}\quad\text{Days} = M \times Z\,, \tag{12}$$

where M reflects the total amount of GPUs and Z denotes the genuine search duration in days. Table 2 contrasts the CIFAR-10 and ImageNet performance of various NAS techniques. In addition, because the majority of NAS approaches begin with the search for the neural architecture based on a small dataset (CIFAR-10) and then transition the architecture to a larger dataset (ImageNet), the time spent searching is comparable for both datasets. According to the tables, early research on EBA- and RLA-based NAS approaches placed a greater emphasis on high performance than on the required quantity of resources. Despite 3150 GPU days and 450 GPUs, AmoebaNet performed well on CIFAR-10 and ImageNet. NAS innovation improved search efficiency while maintaining model performance. EENA painstakingly builds mutation and crossover techniques to utilize obtained knowledge to direct evolution and improve EBA-based NAS systems. ENAS uses parameter-sharing to reduce GPU budgets to 1 and search in less than a day. Gradient descent architecture optimization solutions minimize searching computing resources and deliver SoTA results. Multiple additional studies have optimized this. RS-based approaches yield comparable results. Weight sharing outperformed ENAS and DARTS.

Additionally, it has been observed that architecture optimization approaches based on gradient descent can significantly decrease

Table 2
Comparison of NAS operating capabilities.

Model	First appeared in	Params	Top-1/5 Accuracy (%)	GPUs	GPU Days	Optimization method
CARS [172]	CVPR20	5.1	75.2/92.5	–	0.4	EBA+GD
RENASNet [173]	CVPR19	5.36 M	75.7/92.6	–	–	EBA+RLA
OFA-random [174]	CVPR20	7.7 M	73.8/-	–	–	RS
Hierarchical-random [122]	ICLR18	–	79.6/94.7	200	8.3	RS
PNAS-5($N = 4$, $F = 216$) [120]	ECCV18	86.1 M	82.9/96.2	–	225	SMBO
GHN [175]	GHN [214]	6.1 M	73.0/91.3	–	0.84	SMBO
PNAS-5($N = 3$, $F = 54$) [120]	ECCV18	5.1 M	74.2/91.9	–	255	SMBO
PC-DARTS [176]	CVPR20	5.3 M	75.8/92.7	8V100	3.8	GD
DSNAS [177]	CVPR20	–	74.4/91.54	4 Titan X	17.5	
P-DARTS [178]	ICCV19	4.9 M	75.6/92.6	–	0.3	GD
GDAS [179]	CVPR19	1.4 M	72.5/90.9	1	0.175	GD
SGAS [180]	CVPR20	5.4 M	75.9/92.7	1 1080Ti	0.25	GD
BayesNAS [176]	ICML19	3.9 M	73.5/91.1	1	0.2	GD
DARTS (searched on CIFAR-10) [181]	ICLR19	4.7 M	73.3/81.3	–	4	GD
DenseNAS [182]	CVPR20	–	75.3/-	–	2.7	GD
OFA [174]	ICLR20	7.7 M	77.3/-	–	–	GD
FBNetV2-L1 [183]	CVPR20	–	77.2/-	8V100	2	GD
EfficientNet-B0 [184]	ICML19	5.3 M	77.3/93.5	–	–	RLA
Hierarchical-EAS [122]	ICLR18	–	79.7/94.8	200	300	EBA
CGP-ResSet [185]	IJCAI18	6.4 M	94.02	2	27.4	EBA
BayesNAS [44]	ICML19	3.4 M	97.59	1	0.1	GD
Lemonade [186]	ICLR19	3.4 M	97.6	8 Titan	56	EBA
ProxylessNAS-RL (mobile) [121]	ICLR19	–	74.6/92.2	–	8.3	RLA
NASNet-A (4@1056) [199]	ICLR17	5.3 M	74.0/91.6	500 P100	2000	RLA
ProxylessNAS(GPU) [121]	ICLR19	–	75.1/92.5	–	8.3	RLA
GreedyNAS [163]	CVPR20	6.5 M	77.1/93.3	–	1	EBA
AmoebaNet-C($N = 4$, $F = 50$) [117]	AAAI19	6.4 M	75.7/92.4	450 K40	3150	EBA
AmoebaNet-C($N = 6$, $F = 228$) [44]	AAAI19	155.3 M	83.1/96.3	450 K40	3150	EBA
MobileNetV2 [187]	CVPR18	6.9 M	74.7	–	–	Manually designed
SENet-154 [188]	CVPR17	–	71.32/95.53	–	–	Manually designed
ResNet-152 [189]	CVPR16	230 M	70.62/95.51	–	–	Manually designed
PyramidNet [190]	CVPR17	26 M	96.69	–	–	Manually designed
ResNet-110 [49]	ECCV16	1.7 M	93.57	–	–	Manually designed
RENASNet+c/o [173]	CVPR19	3.5 M	91.12	4	1.5	EBA+RLA
CARS [172]	CVPR20	3.6 M	97.38	–	0.4	EBA+GD
GHN [175]	ICLR19	5.7 M	97.16	–	0.84	SMBO
PNAS [120]	ECCV18	3.2 M	96.59	–	225	SMBO
EENA (more channels [191]	ICCV19	54.14 M	97.79	1 Titan Xp	0.65	EBA
Large-scale ensemble [118]	ICML17	40.4 M	95.6	250	2500	EBA
RandomNAS-NSAS [192]	CVPR20	3.08 M	97.36	–	0.7	RS
SMASH [193]	ICLR18	16 M	95.97	–	1.5	RS
DARTS - random+c/o [181]	ICLR19	3.2 M	96.71	1	4	RS
GDAS-NSAS [192]	CVPR20	3.54 M	97.27	–	0.4	GD
SETN [194]	ICCV19	4.6 M	97.31	–	18	GD
MiLeNAS [156]	CVPR20	3.87 M	97.66	–	0.3	GD
SharpDARTS [195]	ArXiv19	3.6 M	98.07	1 2080Ti	0.8	GD
PC-DARTS+c/o [176]	CVPR20	3.6 M	97.43	1 1080Ti	0.1	GD
SGAS [180]	CVPR20	3.8 M	97.61	1 1080Ti	0.25	GD
DARTS (second order)+c/o [181]	ICLR19	3.3 M	97.23	4 1080Ti	4	GD
FPNAS [196]	ICCV19	5.76 M	96.99	–	–	RLA
ENAS+macro [114]	ICML18	38.0 M	96.13	1	0.32	RLA
Path-level EAS [197]	ICML18	5.7 M	97.01	–	200	RLA
MetaQNN [198]	ICLR17	–	93.08	10	100	
NASNet-A (6@768)+c/o [199]	CVPR18	3.3 M	97.35	500 P100	2000	RLA
NASv3 [12]	ICLR17	7.1 M	95.53	800 K40	22,400	RLA

the computational resources required for searching and getting state-of-the-art (SoTA) outcomes. Numerous further investigations have been undertaken to attain additional enhancement and optimization in this particular area. It is worth noting that approaches based on Reed-Solomon (RS) codes can also provide results that are comparable in nature. In this section, the author demonstrates that the utilization of weight sharing in the RS approach yields superior performance compared to several other very effective methods, including ENAS and DARTS. Given that RS is comparable to more sophisticated techniques such as DARTS and ENAS, it is pertinent to investigate the merits and significance of alternative algorithms for architecture optimization in comparison to RS. Instead of merely evaluating the model's ultimate accuracy, scholars have attempted to address this question using alternative metrics. The majority of NAS approaches consist of two distinct stages: First, the search for the most optimal architecture based on training set performance, and second, the expansion of this architecture to a deeper structure and its subsequent estimation on the validation set. Nonetheless, there is typically a considerable disparity between these two phases. The architectural [200]design that generates the best performance on the training dataset may not be the best option for the validation dataset. Several studies on neural architecture search have therefore employed the Kendall Tau (τ) [201]measure to analyze the correlation between the model's performance in the search and evaluation phases [177].

Here is the equation that defines the Tau (τ) parameter in equation (13):

$$\tau = \frac{N_C - N_D}{N_C + N_D}. \tag{13}$$

The variables N_C and N_D represent the quantities of concordant and discordant pairs, respectively. Variable τ represents a numerical value within the interval of [−1,1].

- The value of τ is equal to 1, indicating that two ranks are equivalent.
- When $\tau = -1$, it indicates that two rankings are absolutely opposed to one other.
- At $\tau = 0$, there exists no discernible association between the two rankings.

The neural architecture search (NAS) that is conducted in two stages is divided into two unique phases, which are referred to as the searching stage and the assessment stage, respectively. During the evaluation phase, more guidance is imparted to the model that shows the superior level of performance in the preceding searching stage, as depicted in Fig. 14.

The one-stage neural architecture search, referred to as NAS, as depicted in Fig. 15, enables the direct deployment of a high-performing model without the need for additional retraining or fine-tuning.

The existence of a directing arrow in both directions indicates that the architectural optimization and parameter training processes are being carried out simultaneously.

8.6. Difference between AutoML and traditional methods of machine learning

As machine learning (ML) continues to gain prominence, organizations are increasingly embracing automation to expedite the model selection process. AutoML, or automated machine learning, is a technology that automates the entire ML process, from data pre-processing to model selection and hyperparameter tuning. In traditional ML all the approaches are done by human manually which is time-consuming and also need the assistance of data scientists who have expertise in machine learning. AutoML automatically identifies and employs the most optimal machine learning algorithm for a given task. It does this with two concepts.

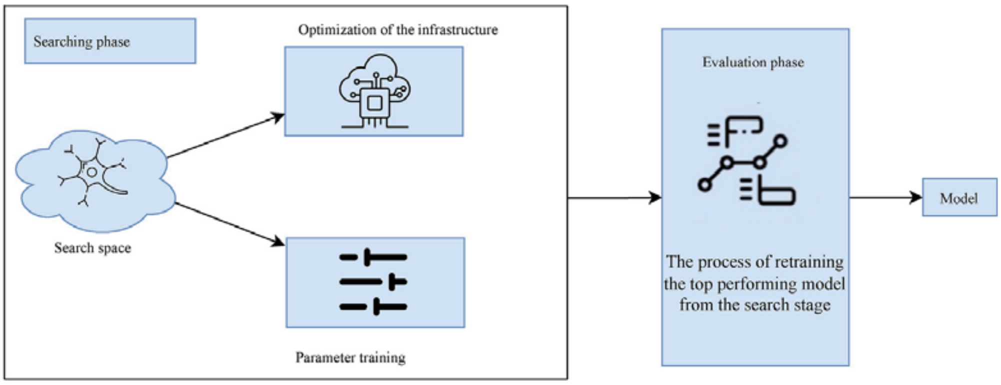
Searching phase
Optimization of the infrastructure
Search space
Parameter training
Evaluation phase
The process of retraining the top performing model from the search stage
Model

Fig. 14. Illustration of two-stage neural architecture search flow.

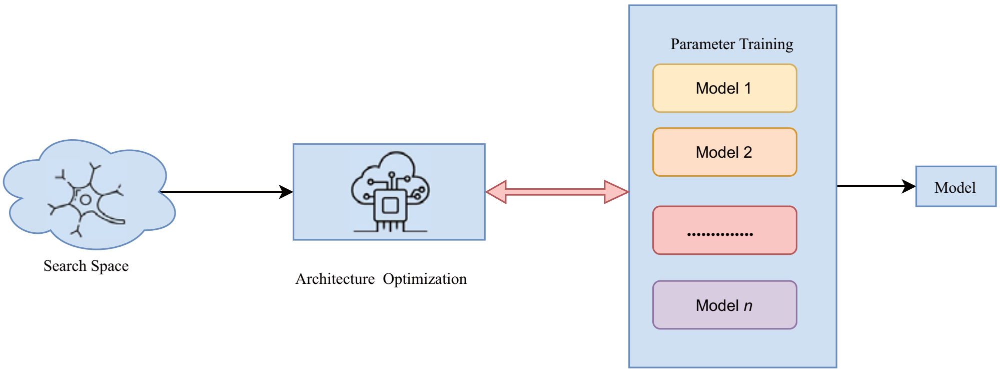

Fig. 15. Illustration of one-stage neural architecture search flow.

- **Neural architecture search:** This process automates the design of neural networks, enabling AutoML models to explore and discover new architectures tailored to specific problem requirements.
- **Transfer learning:** Pre-trained models leverage their acquired knowledge to adapt to new datasets. Transfer learning is the technique that facilitates AutoML in applying existing architectures to tackle new problems that demand such adaptation.

Below is a comparison highlighting the distinctions between AutoML and traditional machine learning methods.

- **Automation:** AutoML is designed to automate various aspects of the machine learning pipeline, including data preprocessing, feature selection, model selection, hyperparameter tuning, and model evaluation. It aims to minimize the need for manual intervention at each stage.In traditional machine learning, many tasks, such as feature engineering, hyperparameter tuning, and model selection, often require manual effort and domain expertise. These tasks are typically performed by data scientists or machine learning engineers.
- **Accessibility:** AutoML tools and platforms are designed to make machine learning more accessible to users with limited machine learning expertise.They are intended for non-experts who want to leverage machine learning for their specific applications. On the other hand, traditional machine learning methods often require a deep understanding of algorithms, data preprocessing techniques, and domain knowledge. They are typically used by data scientists and machine learning practitioners with specialized expertise.
- **Model Selection:** AutoML automates model selection by exploring and evaluating various models. It selects the best model based on performance, reducing the need for experimentation. In traditional ML, model selection typically necessitates human intervention and involves a time-consuming process of extensive experimentation and testing.
- **Hyperparameter tuning:** AutoML platforms often use techniques like Bayesian optimization or genetic algorithms to automatically search for optimal hyperparameters, reducing the need for manual tuning. Hyperparameter tuning in traditional machine learning often involves manual trial-and-error or grid search methods, which can be labor-intensive.
- **Scalability:** AutoML tools are designed to be scalable and can handle large datasets and complex machine-learning tasks with minimal manual effort.The scalability of traditional machine learning methods can be limited by the manual processes involved, making them less efficient for large-scale tasks.

Both automated and traditional methods have their respective advantages and disadvantages. AutoML excels in providing automation and simplicity, but it may come with limitations in terms of control and customization, particularly for specialized or domain-specific tasks. Traditional methods, on the other hand, offer greater control and flexibility, empowering experts to finely customize algorithms and preprocessing steps to meet specific requirements. The choice between AutoML and traditional methods depends on the specific requirements, resources, and expertise available for a given machine learning project.

9. AutoML with robotics

The application of automated machine learning in robotics can be a potent strategy to speed up the creation of machine learning models for a variety of robotic activities. Robotic systems may more easily incorporate machine learning since autoML platforms and libraries offer resources and techniques for automating the processes of feature engineering, hyperparameter tweaking, and model selection. An AutoML at the network's edge can assist in making online forecasts about potential outages of collaborating robots. Robotics metrics were gathered and put into an AutoML model in order to determine the most important dependability criteria and forecast when safe robot pauses would be necessary [202]. In order to deploy artificial intelligence and deep learning-based solutions on the robot without sending any sensitive data to the cloud, it is necessary to pair strong computing units, which ensures compliance with privacy regulations [203]. Pipelines for automatic data preparation are frequently offered by autoML technologies. For human-robot interaction, it can involve object detection, obstacle avoidance, route planning, or even natural language comprehension. Robot capabilities enabled by machine learning may be developed much more quickly by using AutoML in robotics [204]. It makes it possible for robots to adapt to their surroundings, carry out jobs on their own, and communicate with people more successfully.

10. Security and privacy concerns for AutoML

With the rapid advancement of AutoML and neural networks, ensuring ethical practices has become an even more important and pressing issue. Guidelines and frameworks need to be enforced to regulate the development and implementation of AutoML technologies. Machine learning technologies are speculated to heavily impact the privacy of people in recent years. It is a sensitive topic that should be deliberated with care while designing applications and services. There have been concerns raised about how Intelligent Personal Assistants (IPA) for example, Siri and Alexa, could potentially be always on and collecting data [205]. These technologies need a large amount of data but the subjects or the sources of the data have limited rights regarding the usage of their data. Long-term history of anyone who generates storable data can be found. Algorithms with the characteristic of pattern identification can be deployed to find digital history. This implies that the assumption of inconspicuousness is lost. Any person can be identified with the use of facial recognition applications or the data of social media patterns. Our online habits and activity patterns can not only reveal our identity but might also indicate our political views. Social media patterns can also predict a person's personality category and even life events. The choice of words of a person or the pressure of handwriting on a digital stylus can determine a person's emotional state and if the person is telling lies. There are endless possibilities of AutoML and neural networks technologies. There are also countless possible consequences and misuses of them. This raises the question of who is liable for the use and misuse of these technologies. The european parliament resolution (2017) states on the topic of AI that, the owner, developer, or operator of an AI will be held legally responsible for an AI's action or lack of action. These systems must be designed in such a way that promotes human and civil rights. It is believed by the scientific community that trust in these technologies might be earned only by rightness, clarity, answerability and regulations.

11. Open issue discussion

In detail, in our last open issue discussion section, we will base our discussion on these four items specifically focusing on performance and accuracy. Furthermore, by examining the comparison presented in the table, we have identified the limitations of the existing neural architecture search approaches in terms of meeting the requirements. These shortcomings are considered open issues for future research and improvements.

Improving the sample efficiency of NAS algorithms is an ongoing challenge. Currently, NAS methods often require a large number of trials to find optimal architectures. Developing techniques that can effectively search for high-performing architectures with fewer samples is an important area for improvement. As datasets and computational requirements continue to grow, there is a need for scalable NAS methods that can handle large-scale datasets and complex models. Developing NAS algorithms that can efficiently search for architectures in a scalable manner is crucial for practical applications.

NAS methods need to produce architectures that generalize well to unseen data. Ensuring that the discovered architectures are not only high-performing on the training set but also demonstrate good generalization on unseen data is a key challenge in NAS. Current NAS methods rely heavily on automated search algorithms without explicitly incorporating domain knowledge. Finding ways to integrate prior knowledge or domain-specific constraints into the NAS process can lead to more interpretable and effective architectures. NAS algorithms often suffer from a lack of transferability across different tasks or domains. Developing NAS techniques that can transfer knowledge and architectures learned from one task or dataset to another, thereby reducing the search space and computational costs is an important area of research. NAS methods typically generate complex architectures that are challenging to interpret and understand. Developing techniques to enhance the interpretability of the discovered architectures can aid in better understanding the underlying principles and reasoning behind the automated design process. Addressing these open issues in AutoML based on NAS performance will contribute to the advancement and practicality of automated machine learning systems enabling more efficient and effective model design and optimization.

12. Conclusion

In the process of building an AutoML system like this practitioners have many concerns to build an effective system. The ability to think about and solve data selection, modeling selection and optimization methods, hyperparameter adjustment methods, etc. has now become an essential competency for machine learning professionals. In this review, we have mentioned the different aspects of using AutoML with all possible methods and provided explanations for each. This explanation can be used to develop by directly building an AutoML future system that has recently attracted attention. The design of neural network architectures has been a manual and labor-intensive process, requiring domain expertise and extensive experimentation. NAS aims to automate this process by using algorithms and computational resources to discover effective neural network architectures. Neural architecture search is a powerful idea that leverages automation and computational resources to discover neural network architectures tailored to specific tasks. It has contributed to significant advancements in the field of deep learning and continues to be an active area of research. This review will help the reader to understand NAS and AutoML clearly with some proper strong references and experiment result compression. AutoML is poised to have a significant impact on the advanced artificial intelligence domain and the era of AI in the coming years.

CRediT authorship contribution statement

Imrus Salehin: Conceptualization, S.I. and S. P.; Data curation, S.I.; Formal analysis, S.I. and S. P.; Investigation, S.I., B. M.A, and H.M; Methodology, I. S.M and N. S.M; Project administration, T.A; Software, S.I.; Supervision, S.I; Validation, S.I; Visualization, S.I.; Writing–original draft, S.I., I.S.M, S.P, N.S.M; Writing–review and editing, H.M and B.M.A. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

The authors express their gratitude to the members of the Dongseo University Machine Learning/Deep Learning Research Lab, the Daffodil International University Innovation Lab, and the anonymous referees for their valuable feedback on earlier drafts of this paper. Special thanks are extended to Professor Moon for their insightful review and experimental assistance. The review of this article was coordinated by Dr. Jungong Han.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Wortsman, G. Ilharco, S.Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A.S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, L. Schmidt, Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, in: Proceedings of the 39th International Conference on Machine Learning, PMLR, New York, 2022, pp. 23965–23998.

[2] X. Zhai, A. Kolesnikov, N. Houlsby, L. Beyer, Scaling vision transformers, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2022, pp. 12104–12113.

[3] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L.M. Ni, H.-Y. Shum, DINO: DETR with improved denoising anchor boxes for end-to-end object detection, 2022, https://arxiv.org/pdf/2203.03605.pdf.

[4] L. Martin, B. Muller, P.J.O. Suárez, Y. Dupont, L. Romary, Éric Villemonte de La Clergerie, D. Seddah, B. Sagot, Camembert: A tasty french language model, 2019, https://arxiv.org/pdf/1911.03894.pdf.

[5] H. Le, L. Vial, J. Frej, V. Segonne, M. Coavoux, B. Lecouteux, A. Allauzen, B. Crabbé, L. Besacier, D. Schwab, FlauBERT: Unsupervised language model pre-training for French, 2019, https://arxiv.org/pdf/1912.05372.pdf.

[6] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90.

[7] D. Erhan, C. Szegedy, A. Toshev, D. Anguelov, Scalable object detection using deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2014, pp. 2147–2154.

[8] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2016, pp. 770–778.

[9] X. He, K. Zhao, X. Chu, AutoML: A survey of the state-of-the-art, Knowledge-Based Systems 212 (2021) 106622.

[10] A.D. Morozov, D.O. Popkov, V.M. Duplyakov, R.F. Mutalova, A.A. Osiptsov, A.L. Vainshtein, E.V. Burnaev, E.V. Shel, G.V. Paderin, Data-driven model for hydraulic fracturing design optimization: Focus on building digital database and production forecast, Journal of Petroleum Science and Engineering 194 (2020) 107504.

[11] R. Luo, F. Tian, T. Qin, E. Chen, T.-Y. Liu, Neural architecture optimization, Advances in Neural Information Processing Systems 31 (2018).

[12] B. Zoph, Q.V. Le, Neural architecture search with reinforcement learning, 2016, https://arxiv.org/pdf/1611.01578.pdf.

[13] J. Liang, E. Meyerson, B. Hodjat, D. Fink, K. Mutch, R. Miikkulainen, Evolutionary neural AutoML for deep learning, in: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, New York, 2019, pp. 401–409.

[14] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, J. Vanschoren, An open source AutoML benchmark, 2019, https://arxiv.org/pdf/1907.00909.pdf.

[15] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. Bayan Bruss, R. Farivar, Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools, in: Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2019, pp. 1471–1479.

[16] J. Yang, R. Shi, B. Ni, MedMNIST classification decathlon: A lightweight AutoML benchmark for medical image analysis, in: Proceedings of the 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), IEEE, 2021, pp. 191–195.

[17] Y. Li, Y. Shen, W. Zhang, C. Zhang, B. Cui, Volcanoml: Speeding up end-to-end AutoML via scalable search space decomposition, The International Journal on Very Large Data Bases 32 (2) (2023) 389–413.

[18] S.K. Karmaker, M.M. Hassan, M.J. Smith, L. Xu, C. Zhai, K. Veeramachaneni, AutoML to date and beyond: Challenges and opportunities, ACM Computing Surveys (CSUR) 54 (8) (2021) 1–36.

[19] M. Wever, A. Tornede, F. Mohr, E. Hüllermeier, AutoML for multi-label classification: Overview and empirical evaluation, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (9) (2021) 3037–3054.

[20] L. Ferreira, A. Pilastri, C.M. Martins, P.M. Pires, P. Cortez, A comparison of AutoML tools for machine learning, deep learning and XGBoost, in: Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, 2021, pp. 1–8.

[21] M. Bahri, F. Salutari, A. Putina, M. Sozio, AutoML: State of the art with a focus on anomaly detection, challenges, and research directions, International Journal of Data Science and Analytics 14 (2) (2022) 113–126.

[22] H. Rakotoarison, L. Milijaona, A. Rasoanaivo, M. Sebag, M. Schoenauer, Learning meta-features for AutoML, in: Proceedings of the International Conference on Learning Representations, OpenReview.net, 2022, pp. 113–126.

[23] C. Wang, Q. Wu, M. Weimer, E. Zhu, FLMAL: A fast and lightweight AutoML library, Proceedings of Machine Learning and Systems 3 (2021) 434–447.

[24] C. Wong, N. Houlsby, Y. Lu, A. Gesmundo, Transfer learning with neural AutoML, Advances in Neural Information Processing Systems 31 (2018).

[25] A. Yakovlev, H.F. Moghadam, A. Moharrer, J. Cai, N. Chavoshi, V. Varadarajan, S.R. Agrawal, S. Idicula, T. Karnagel, S. Jinturkar, N. Agarwal, AutoML. Oracle, A fast and predictive AutoML pipeline, Proceedings of the VLDB Endowment 13 (12) (2020) 3166–3180.

[26] E. Real, C. Liang, D. So, Q. Le, AutoML-Zero: Evolving machine learning algorithms from scratch, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2020, pp. 8007–8019.

[27] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, S. Han, AMC: AutoML for model compression and acceleration on mobile devices, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2018, pp. 784–800.

[28] M. Lindauer, F. Hutter, Best practices for scientific research on neural architecture search, The Journal of Machine Learning Research 21 (1) (2020) 9820–9837.

[29] J. Kim, S. Lee, S. Kim, M. Cha, J.K. Lee, Y. Choi, Y. Choi, D.-Y. Cho, J. Kim, Auto-meta: Automated gradient based meta learner search, 2018, https://arxiv.org/pdf/1806.06927.pdf.

[30] Z. Zhang, X. Wang, W. Zhu, Automated machine learning on graphs: A survey, 2021, https://arxiv.org/pdf/2103.00742.pdf.

[31] V.-K. Vo-Ho, K. Yamazaki, H. Hoang, M.-T. Tran, N. Le, Meta-learning of NAS for few-shot learning in medical image applications, 2022, https://arxiv.org/pdf/2203.08951.pdf.

[32] J. Artin, A. Valizadeh, M. Ahmadi, S.A.P. Kumar, A. Sharifi, Presentation of a novel method for prediction of traffic with climate condition based on ensemble learning of neural architecture search (NAS) and linear regression, Complexity 2021 (2021) 1–13.

[33] Y. Akimoto, S. Shirakawa, N. Yoshinari, K. Uchida, S. Saito, K. Nishida, Adaptive stochastic natural gradient method for one-shot neural architecture search, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2019, pp. 171–180.

[34] X. Chen, C.-J. Hsieh, Stabilizing differentiable architecture search via perturbation-based regularization, in: Proceedings of the International conference on machine learning, PMLR, New York, 2020, pp. 1554–1565.

[35] H.-P. Cheng, F. Liang, M. Li, B. Cheng, F. Yan, H. Li, V. Chandra, Y. Chen, ScaleNAS: Multi-path one-shot NAS for scale-aware high-resolution representation, in: Proceedings of the International Conference on Automated Machine Learning, PMLR, New York, 2022, pp. 1–15.

[36] S. Santra, J.-W. Hsieh, C.-F. Lin, Gradient descent effects on differential neural architecture search: A survey, IEEE Access 9 (2021) 89602–89618.

[37] Y. Hirose, N. Yoshinari, S. Shirakawa, NAS-HPO-Bench-II: A benchmark dataset on joint optimization of convolutional neural network architecture and training hyperparameters, in: Proceedings of the Asian Conference on Machine Learning, PMLR, New York, 2021, pp. 1349–1364.

[38] N. Nayman, Y. Aflalo, A. Noy, L. Zelnik, N.A.S. Hardcore-, Hard constrained differentiable neural architecture search, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2021, pp. 7979–7990.

[39] J.C.O. Koh, G. Spangenberg, S. Kant, Automated machine learning for high-throughput image-based plant phenotyping, Remote Sensing 13 (5) (2021) 858.

[40] W. Jia, W. Xia, Y. Zhao, H. Min, Y.-X. Chen, 2D and 3D palmprint and palm vein recognition based on neural architecture search, International Journal of Automation and Computing 18 (2021) 377–409.

[41] F.P. Such, A. Rawal, J. Lehman, K. Stanley, J. Clune, Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2020, pp. 9206–9216.

[42] J. Xu, L. Zhao, J. Lin, R. Gao, X. Sun, H. Yang, KNAS: Green neural architecture search, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2021, pp. 11613–11625.

[43] Y. Zhao, L. Wang, Y. Tian, R. Fonseca, T. Guo, Few-shot neural architecture search, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2021, pp. 12707–12718.

[44] H. Zhou, M. Yang, J. Wang, W. Pan, BayesNAS: A Bayesian approach for neural architecture search, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2019, pp. 7603–7613.
[45] Z. Lu, R. Cheng, Y. Jin, K.C. Tan, K. Deb, Neural architecture search as multiobjective optimization benchmarks: Problem formulation and performance assessment, IEEE Transactions on Evolutionary Computation, 2023.
[46] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, A. Filippov, E. Burnaev, N.L.P. NAS-Bench-NLP: Neural architecture search benchmark for natural language processing, IEEE Access 10 (2022) 45736–45747.
[47] T. Chau, Ł. Dudziak, H. Wen, N. Lane, M. Abdelfattah, BLOX: Macro neural architecture search benchmark and algorithms, Advances in Neural Information Processing Systems 35 (2022) 30851–30864.
[48] B. Lyu, S. Wen, Y. Yang, X. Chang, J. Sun, Y. Chen, T. Huang, Designing efficient bit-level sparsity-tolerant memristive networks, IEEE Transactions on Neural Networks and Learning Systems 2 (2023) 1–10.
[49] S. Tuli, N.K. Jha, EdgeTran: Co-designing transformers for efficient inference on mobile edge platforms, 2023, https://arxiv.org/pdf/2303.13745.pdf.
[50] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, S. Han, HAT: Hardware-aware transformers for efficient natural language processing, 2020, https://arxiv.org/pdf/2005.14187.pdf.
[51] S. Tuli, B. Dedhia, S. Tuli, N.K. Jha, FlexiBERT, Are current transformer architectures too homogeneous and rigid? Journal of Artificial Intelligence Research 77 (2023) 39–70.
[52] M.A. Khan, N. Iqbal, Imran, H. Jamil, D.-H. Kim, An optimized ensemble prediction model using AutoML based on soft voting classifier for network intrusion detection, Journal of Network and Computer Applications 212 (2023) 103560.
[53] A. Karras, C. Karras, N. Schizas, M. Avlonitis, S. Sioutas, AutoML with bayesian optimizations for big data management, Information 14 (4) (2023) 223.
[54] H.A. Madni, M. Umer, A. Ishaq, N. Abuzinadah, O. Saidani, S. Alsubai, M. Hamdi, I. Ashraf, Water-quality prediction based on H_2O AutoML and explainable AI techniques, Water 15 (3) (2023) 475.
[55] E.K. Sahin, S. Demir. Greedy-AutoML, A novel greedy-based stacking ensemble learning framework for assessing soil liquefaction potential, Engineering Applications of Artificial Intelligence 119 (2023) 105732.
[56] L. Deng, The MNIST database of handwritten digit images for machine learning research [Best of the Web], IEEE Signal Processing Magazine 29 (6) (2012) 141–142.
[57] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Handbook of Systemic Autoimmune Diseases 1 (4) (2009).
[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[59] X. Li, C. Xu, X. Wang, W. Lan, Z. Jia, G. Yang, J. Xu, COCO-CN for cross-lingual image tagging, captioning and retrieval, IEEE Transactions on Multimedia 21 (9) (2019) 2347–2360.
[60] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, L. Fei-Fei, The unreasonable effectiveness of noisy data for fine-grained recognition, in: Proceedings of the 2016 European Conference on Computer Vision, Springer, Cham, 2016, pp. 301–320.
[61] Y. Roh, G. Heo, S.E. Whang, A survey on data collection for machine learning: A big data-ai integration perspective, IEEE Transactions on Knowledge and Data Engineering 33 (4) (2019) 1328–1347.
[62] D. Yarowsky, Unsupervised word sense disambiguation rivaling supervised methods, in: Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Cambridge, 1995, p. 189196.
[63] M. Farouk Abdel Hady, F. Schwenker, Combining committee-based semi-supervised learning and active learning, Journal of Computer Science and Technology 25 (4) (2010) 681–698.
[64] Y. Zhou, S. Goldman, Democratic co-learning, in: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, IEEE, 2004, pp. 594–602.
[65] J. Yang, X. Sun, Y.K. Lai, L. Zheng, M.M. Cheng, Recognition from web data: A progressive filtering approach, IEEE Transactions on Image Processing 27 (11) (2018) 5303–5315.
[66] F.R. Adi Pratama, S.I. Oktora, Synthetic minority over-sampling technique (smote) for handling imbalanced data in poverty classification, Statistical Journal of the IAOS 39 (1) (2023) 233–239.
[67] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI GYM, 2016, https://arxiv.org/pdf/1606.01540.pdf.
[68] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, F.-Y. Wang, Generative adversarial networks: Introduction and outlook, IEEE/CAA Journal of Automatica Sinica 4 (4) (2017) 588–598.
[69] Y. Li, M. Min, D. Shen, D. Carlson, L. Carin, Video generation from text, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, Menlo Park, 2018, pp. 588–598.
[70] G. Tevet, G. Habib, V. Shwartz, J. Berant, Evaluating text GANs as language models, 2019, https://arxiv.org/pdf/1810.12686.pdf.
[71] F. Ridzuan, W.M.N.W. Zainon, A review on data cleansing methods for big data, Procedia Computer Science 161 (2019) 731–738.
[72] V. Raman, J. Hellerstein, Potter's wheel: An interactive framework for data transformation and cleaning, 2001, https://api.semanticscholar.org/CorpusID:13918226.
[73] M.L. Lee, T.W. Ling, W.L. Low, IntelliClean, A knowledge-based intelligent data cleaner, in: Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, 2000, pp. 290–294.
[74] X. Chu, J. Morcos, I.F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, Y. Ye, Katara: A data cleaning system powered by knowledge bases and crowdsourcing, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, New York, 2015, pp. 1247–1261.
[75] S. Krishnan, D. Haas, M.J. Franklin, E. Wu, Towards reliable interactive data cleaning: A user survey and recommendations, in: Proceedings of the Workshop on Human-In-the-Loop Data Analytics, Association for Computing Machinery, New York, 2016, pp. 1–5.
[76] S. Krishnan, M.J. Franklin, K. Goldberg, E. Wu, BoostClean: Automated error detection and repair for machine learning, 2017, https://arxiv.org/pdf/1711.01299.pdf.
[77] S. Krishnan, E. Wu, AlphaClean: Automatic generation of data cleaning pipelines, 2019, https://arxiv.org/pdf/1904.11827.pdf.
[78] I.F. Ilyas, Effective data cleaning with continuous evaluation, IEEE Data Engineering Bulletin 39 (2016) 38–46.
[79] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning, Journal of Big Data 6 (1) (2019) 1–48.
[80] T. Devries, G.W. Taylor, Improved regularization of convolutional neural networks with cutout, 2017, https://arxiv.org/pdf/1708.04552.pdf.
[81] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, Mixup: Beyond empirical risk minimization, 2018, https://arxiv.org/pdf/1710.09412.pdf.
[82] S. Yun, D. Han, S. Chun, S.J. Oh, Y. Yoo, J. Choe, CutMix: Regularization strategy to train strong classifiers with localizable features, in: Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, 2019, pp. 6023–6032.
[83] A. Buslaev, V.I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, A.A. Kalinin, Albumentations: Fast and flexible image augmentations, Information 11 (2) (2020) 125.
[84] Q. Ma, X. Huang, Research on recognizing required items based on openCV and machine learning, in: Proceedings of the SHS Web of Conferences, EDP Sciences, 2022, p. 01016.
[85] A. Mikołajczyk, M. Grochowski, Data augmentation for improving deep learning in image classification problem, in: Proceedings of the 2018 International Interdisciplinary PhD Workshop (IIPhDW), IEEE, 2018, pp. 117–122.
[86] A. Mikołajczyk, M. Grochowski, Style transfer-based image synthesis as an efficient regularization technique in deep learning, 2019, https://arxiv.org/pdf/1905.10974.pdf.
[87] A. Antoniou, A. Storkey, H. Edwards, Data augmentation generative adversarial networks, 2018, https://arxiv.org/pdf/1711.04340.pdf.
[88] S.C. Wong, A. Gatt, V. Stamatescu, M.D. McDonnell, Understanding data augmentation for classification: when to warp?, 2016, https://arxiv.org/pdf/1609.08764.pdf.
[89] J. Chen, Z. Yang, D. Yang, MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification, 2020, https://arxiv.org/pdf/2004.12239.pdf.

[90] A.W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, Q.V. Le, QANet: Combining local convolution with global self-attention for reading comprehension, 2018, https://arxiv.org/pdf/1804.09541.pdf.

[91] E.D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q.V. Le, AutoAugment: Learning augmentation strategies from data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 113–123.

[92] Y. Li, G. Hu, Y. Wang, T. Hospedales, N.M. Robertson, Y. Yang, Dada: Differentiable automatic data augmentation, 2020, https://arxiv.org/pdf/2003.03780.pdf.

[93] R. Hataya, J. Zdenek, K. Yoshizoe, H. Nakayama, Faster AutoAugment: Learning augmentation strategies using backpropagation, 2019, https://arxiv.org/pdf/1911.06987.pdf.

[94] S. Lim, I. Kim, T. Kim, C. Kim, S. Kim, Fast AutoAugment, Advances in Neural Information Processing Systems 32 (2019).

[95] C. Lin, M. Guo, C. Li, X. Yuan, W. Wu, J. Yan, D. Lin, W. Ouyang, Online hyper-parameter learning for auto-augmentation strategy, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, IEEE, 2019, pp. 6579–6588.

[96] A. Naghizadeh, M. Abavisani, D.N. Metaxas, Greedy AutoAugment, 2019, https://arxiv.org/pdf/1908.00704.pdf.

[97] M. Geng, K. Xu, B. Ding, H. Wang, L. Zhang, Learning data augmentation policies using augmented random search, 2018, https://arxiv.org/pdf/1811.04768.pdf.

[98] T.C. LingChen, A. Khonsari, A. Lashkari, M.R. Nazari, J.S. Sambee, M.A. Nascimento, UniformAugment: A search-free probabilistic data augmentation approach, 2020, https://arxiv.org/pdf/2003.14348.pdf.

[99] M. Gada, Z. Haria, A. Mankad, K. Damania, S. Sankhe, Automated feature engineering and hyperparameter optimization for machine learning, in: Proceedings of the 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2021, pp. 981–986.

[100] Y. Xu, J. Pei, L. Lai, Deep learning based regression and multiclass models for acute oral toxicity prediction with automatic chemical feature extraction, Journal of Chemical Information and Modeling 57 (11) (2017) 2672–2685.

[101] Q. Tang, Y. Liu, H. Liu, Medical image classification via multiscale representation learning, Artificial Intelligence in Medicine 79 (2017) 71–78.

[102] D. Xin, E.Y. Wu, D.J.-L. Lee, N. Salehi, A. Parameswaran, Whither AutoML? Understanding the role of automation in machine learning workflows, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, 2021, pp. 1–16.

[103] M. Goodarzi, B. Dejaegher, Y.V. Heyden, Feature selection methods in QSAR studies, Journal of AOAC International 95 (3) (2012) 636–651.

[104] M. Cerrada, L. Trujillo, D.E. Hernández, H.A. Correa Zevallos, J.C. Macancela, D. Cabrera, R.V. Sánchez, AutoML for feature selection and model tuning applied to fault severity diagnosis in spur gearboxes, Mathematical and Computational Applications 27 (1) (2022) 6.

[105] S.C. Yusta, Different metaheuristic strategies to solve the feature selection problem, Pattern Recognition Letters 30 (5) (2009) 525–534.

[106] Z.M. Hira, D.F. Gillies, A review of feature selection and feature extraction methods applied on microarray data, Advances in Bioinformatics 2015 (2015).

[107] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: Explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on International Conference on Machine Learning, Madison, 2011, pp. 833–840.

[108] S.K. D'mello, S.D. Craig, A. Witherspoon, B. McDaniel, A. Graesser, Automatic detection of learner's affect from conversational cues, User Modeling and User-Adapted Interaction 18 (2008) 45–80.

[109] R. Vilalta, Y. Drissi, A perspective view and survey of meta-learning, Artificial Intelligence Review 18 (2002) 77–95.

[110] T. Mu, H. Wang, C. Wang, Z. Liang, X. Shao, Auto-cash: A meta-learning embedding approach for autonomous classification algorithm selection, Information Sciences 591 (2022) 344–364.

[111] Y. Yang, Y. Zhu, H. Cui, X. Kan, L. He, Y. Guo, C. Yang, Data-efficient brain connectome analysis via multi-task meta-learning, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, 2022, pp. 4743–4751.

[112] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 14323–14332.

[113] Z. Zhong, J. Yan, W. Wu, J. Shao, C.-L. Liu, Practical block-wise neural network architecture generation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 14323–14332.

[114] H. Pham, M. Guan, B. Zoph, Q. Le, J. Dean, Efficient neural architecture search via parameters sharing, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2018, pp. 4095–4104.

[115] B. Zoph, Q. Le, Neural architecture search with reinforcement learning, 2017, https://arxiv.org/pdf/1611.01578.pdf.

[116] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 770–778.

[117] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, Menlo Park, 2019, pp. 4780–4789.

[118] E. Real, S. Moore, A. Selle, S. Saxena, Y.L. Suematsu, J. Tan, Q.V. Le, A. Kurakin, Large-scale evolution of image classifiers, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2017, pp. 2902–2911.

[119] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q.V. Le, MnasNet: Platform-aware neural architecture search for mobile, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 2820–2828.

[120] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, K. Murphy, Progressive neural architecture search, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2018, pp. 19–34.

[121] H. Cai, L. Zhu, S. Han, ProxylessNAS, Direct neural architecture search on target task and hardware, 2018, https://arxiv.org/pdf/1812.00332.pdf.

[122] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, K. Kavukcuoglu, Hierarchical representations for efficient architecture search, 2017, https://arxiv.org/pdf/1711.00436.pdf.

[123] T. Chen, I. Goodfellow, J. Shlens, Net2net: Accelerating learning via knowledge transfer, 2015, https://arxiv.org/pdf/1511.05641.pdf.

[124] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015, https://arxiv.org/pdf/1503.02531.pdf.

[125] T. Wei, C. Wang, Y. Rui, C.W. Chen, Network morphism, in: Proceedings of The 33rd International Conference on Machine Learning, PMLR, New York, 2016, pp. 564–572.

[126] Y. Guo, Y. Luo, Z. He, J. Huang, J. Chen, Hierarchical neural architecture search for single image super-resolution, IEEE Signal Processing Letters 27 (2020) 1255–1259.

[127] B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, W. Ouyang, GLiT: Neural architecture search for global and local image transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, IEEE, 2021, pp. 12–21.

[128] M. Wistuba, Practical deep learning architecture optimization, in: Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2018, pp. 263–272.

[129] P. Shanmugavadivu, P. Chitra, S. Lakshmanan, P. Nagaraja, U. Vignesh, Bio-optimization of deep learning network architectures, Security and Communication Networks 2022, 2022.

[130] P.A. Vikhar, Evolutionary algorithms: A critical review and its future prospects, in: Proceedings of the 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), IEEE, 2016, pp. 261–265.

[131] A. Slowik, H. Kwasnicka, Evolutionary algorithms and their applications to engineering problems, Neural Computing and Applications 32 (16) (2020) 12363–12379.

[132] M. Anton, Automated machine learning using evolutionary algorithms, in: Proceedings of the 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), IEEE, 2020, pp. 101–107.

[133] M. Naeem, S.T.H. Rizvi, A. Coronato, A gentle introduction to reinforcement learning and its application in different fields, IEEE Access 8 (2020) 209320–209344.

[134] W. Qiang, Z. Zhongli, Reinforcement learning model, algorithms and its application, in: Proceedings of the 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), IEEE, 2011, pp. 1143–1146.

[135] J. Parker-Holder, R. Rajan, X. Song, A. Biedenkapp, Y. Miao, T. Eimer, B. Zhang, V. Nguyen, R. Calandra, A. Faust, et al., Automated Reinforcement Learning (AutoRL): A survey and open problems, Journal of Artificial Intelligence Research 74 (2022) 517–568.

[136] S. Greenhill, S. Rana, S. Gupta, P. Vellanki, S. Venkatesh, Bayesian optimization for adaptive experimental design: A review, IEEE Access 8 (2020) 13937–13948.
[137] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, BOA: The Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), Morgan Kaufmann Publishers Inc., San Francisco, 1999, pp. 525–532.
[138] L.C.J. Pérez, E.C. Garrido Merchán, Towards automatic Bayesian optimization: A first step involving acquisition functions, in: Proceedings of the Advances in Artificial Intelligence: 19th Conference of the Spanish Association for Artificial Intelligence, Springer, Cham, 2021, pp. 160–169.
[139] E.M. Dogo, O.J. Afolabi, N.I. Nwulu, B. Twala, C.O. Aigbavboa, A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks, in: Proceedings of the 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), IEEE, 2018, pp. 92–99.
[140] C. Zhang, M. Yao, W. Chen, S. Zhang, D. Chen, Y. Wu, Gradient descent optimization in deep learning model training based on multistage and method combination strategy, Security and Communication Networks 2021 (2021) 1–15.
[141] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, Neural Networks: Tricks of the Trade: Second Edition 7700 (2012) 437–478.
[142] H. Muhsen, I. Tanninah, Analysis and simulation of maximum power point tracking based on gradient ascent method, in: Proceedings of the 2021 12th International Renewable Engineering Conference (IREC), IEEE, 2021, pp. 1–5.
[143] C. Daskalakis, I. Panageas, The limit points of (optimistic) gradient descent in min-max optimization, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, Association for Computing Machinery, New York, 2018, pp. 9256–9266.
[144] S. Lu, R. Singh, X. Chen, Y. Chen, M. Hong, Alternating gradient descent ascent for nonconvex min-max problems in robust learning and gans, in: Proceedings of the 2019 53rd Asilomar Conference on Signals, Systems, and Computers, IEEE, 2019, pp. 680–684.
[145] A. Karras, C. Karras, N. Schizas, M. Avlonitis, S. Sioutas, AutoML with Bayesian optimizations for big data management, Information 14 (4) (2023) 223.
[146] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, Z. Li, DARTS+: Improved differentiable architecture search with early stopping, 2019, https://arxiv.org/pdf/1909.06035.pdf.
[147] L. Li, A. Talwalkar, Random search and reproducibility for neural architecture search, in: Proceedings of the Uncertainty in Artificial Intelligence, PMLR, New York, 2020, pp. 367–377.
[148] X. Chu, B. Zhang, R. Xu, FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, IEEE, 2021, pp. 12239–12248.
[149] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, D. Marculescu, Single-path NAS: Designing hardware-efficient convNets in less than 4 hours, in: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, Cham, 2020, pp. 481–497.
[150] K. Zhou, Q. Song, X. Huang, X. Hu, Auto-GNN: Neural architecture search of graph neural networks, 2019, https://arxiv.org/pdf/1909.03184.pdf.
[151] X. Dong, M. Tan, A.W. Yu, D. Peng, B. Gabrys, Q.V. Le, AutoHAS: Differentiable hyper-parameter and architecture search, 2020, https://arxiv.org/pdf/2006.03656v1.pdf.
[152] G. Bender, H. Liu, B. Chen, G. Chu, S. Cheng, P.-J. Kindermans, Q.V. Le, Can weight sharing outperform random architecture search? An investigation with tuNAS, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 14323–14332.
[153] A. Boulch, ShaResNet: Reducing residual network parameter number by sharing weights, 2017, https://arxiv.org/pdf/1702.08782.pdf.
[154] P. Savarese, M. Maire, Learning implicitly recurrent CNNs through parameter sharing, 2019, https://arxiv.org/pdf/1902.09701.pdf.
[155] Z. Zhong, J. Yan, W. Wu, J. Shao, C.-L. Liu, Practical block-wise neural network architecture generation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2018, pp. 2423–2432.
[156] C. He, H. Ye, L. Shen, T. Zhang, MileNAS: Efficient neural architecture search via mixed-level reformulation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 11993–12002.
[157] Y. Kim, W.J. Yun, Y.K. Lee, J. Kim, Two-stage architectural fine-tuning with neural architecture search using early-stopping in image classification, 2022, https://arxiv.org/pdf/2202.08604.pdf.
[158] H. Zhang, Y. Li, H. Chen, C. Shen, Memory-efficient hierarchical neural architecture search for image denoising, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 3657–3666.
[159] R. Pang, Z. Xi, S. Ji, X. Luo, T. Wang, On the security risks of AutoML, in: Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 3953–3970.
[160] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, F. Hutter, NAS-Bench-101: Towards reproducible neural architecture search, in: Proceedings of the International Conference on Machine Learning, PMLR, New York, 2019, pp. 7105–7114.
[161] X. Dong, Y. Yang, NAS-Bench-201: Extending the scope of reproducible neural architecture search, 2020, https://arxiv.org/pdf/2001.00326.pdf.
[162] P. Chrabaszcz, I. Loshchilov, F. Hutter, A downsampled variant of imageNet as an alternative to the CIFAR datasets, 2017, https://arxiv.org/pdf/1707.08819.pdf.
[163] S. You, T. Huang, M. Yang, F. Wang, C. Qian, C. Zhang, GreedyNAS: Towards fast one-shot NAS with greedy supernet, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 1999–2008.
[164] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, S. Su, One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (9) (2020) 2921–2935.
[165] Y. Xiao, Y. Qiu, X. Li, A survey on one-shot neural architecture search, in: Proceedings of the IOP Conference Series: Materials Science and Engineering, IOP, 2020, p. 012223.
[166] G. Li, Y. Yang, K. Bhardwaj, R. Marculescu, ZiCo: Zero-shot NAS via inverse coefficient of variation on gradients, 2023, https://arxiv.org/pdf/2301.11300.pdf.
[167] M. Javaheripi, S. Shah, S. Mukherjee, T.L. Religa, C.C.T. Mendes, G.H. de Rosa, S. Bubeck, F. Koushanfar, D. Dey, LiteTransformerSearch: Training-free on-device search for efficient autoregressive language models, 2022, https://arxiv.org/pdf/2203.02094.pdf.
[168] G. Li, S.K. Mandal, U.Y. Ogras, R. Marculescu, Flash: Fast neural architecture search with hardware optimization, ACM Transactions on Embedded Computing Systems (TECS) 20 (5s) (2021) 1–26.
[169] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, R. Jin, N.A.S. Zen-NAS, A zero-shot NAS for highperformance image recognition, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, IEEE, 2021, pp. 347–356.
[170] S. Green, C.M. Vineyard, R. Helinski, C.K. Koç, RAPDARTS: Resource-aware progressive differentiable architecture search, in: Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–7.
[171] Z. Yang, Q. Sun, Efficient resource-aware neural architecture search with dynamic adaptive network sampling, in: Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, 2021, pp. 1–5.
[172] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, C. Xu, CARS: Continuous evolution for efficient neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 1829–1838.
[173] Y. Chen, G. Meng, Q. Zhang, S. Xiang, C. Huang, L. Mu, X. Wang, Reinforced evolutionary neural architecture search, 2018, https://arxiv.org/pdf/1808.00193.pdf.
[174] H. Cai, C. Gan, T. Wang, Z. Zhang, S. Han, Once-for-all: Train one network and specialize it for efficient deployment, 2019, https://arxiv.org/pdf/1908.09791.pdf.
[175] C. Zhang, M. Ren, R. Urtasun, Graph hypernetworks for neural architecture search, 2018, https://arxiv.org/pdf/1810.05749.pdf.
[176] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, H. Xiong, PC-DARTS: Partial channel connections for memory-efficient architecture search, 2019, https://arxiv.org/pdf/1907.05737.pdf.
[177] S. Hu, S. Xie, H. Zheng, C. Liu, J. Shi, X. Liu, D. Lin, DSNAS: Direct neural architecture search without parameter retraining, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 12084–12092.
[178] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive differentiable architecture search: Bridging the depth gap between search and evaluation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, IEEE, 2019, pp. 1294–1303.
[179] X. Dong, Y. Yang, Searching for a robust neural architecture in four GPU hours, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 1761–1770.
[180] G. Li, G. Qian, I.C. Delgadillo, M. Muller, A. Thabet, B. Ghanem, SGAS: Sequential greedy architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 1620–1630.

[181] H. Liu, K. Simonyan, Y. Yang, DARTS: Differentiable architecture search, 2018, https://arxiv.org/pdf/1806.09055.pdf.
[182] J. Fang, Y. Sun, Q. Zhang, Y. Li, W. Liu, X. Wang, Densely connected search space for more flexible neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 10628–10637.
[183] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, P. Vajda, J.E. Gonzalez, FBNetV2: Differentiable neural architecture search for spatial and channel dimensions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 12965–12974.
[184] M. Tan, Q. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, New York, 2019, pp. 6105–6114.
[185] M. Suganuma, S. Shirakawa, T. Nagao, A genetic programming approach to designing convolutional neural network architectures, in: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, New York, 2017, pp. 497–504.
[186] T. Elsken, J.H. Metzen, F. Hutter, Efficient multi-objective neural architecture search via Lamarckian evolution, 2018, https://arxiv.org/pdf/1804.09081.pdf.
[187] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 4510-4520.
[188] H. Jie, S. Li, S. Gang, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 7132–7141.
[189] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, A. Torralba, Eye tracking for everyone, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2016, pp. 2176–2184.
[190] D. Han, J. Kim, J. Kim, Deep pyramidal residual networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2017, pp. 5927–5935.
[191] H. Zhu, Z. An, C. Yang, K. Xu, E. Zhao, Y. Xu, EENA: Efficient evolution of neural architecture, in: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, IEEE, 2019, pp. 1891-1899.
[192] M. Zhang, H. Li, S. Pan, X. Chang, S. Su, Overcoming multi-model forgetting in one-shot NAS with diversity maximization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2020, pp. 7809–7818.
[193] A. Brock, T. Lim, J.M. Ritchie, N. Weston, Smash: One-shot model architecture search through hypernetworks, 2017, https://arxiv.org/pdf/1708.05344.pdf.
[194] X. Dong, Y. Yang, One-shot neural architecture search via self-evaluated template network, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, IEEE, 2019, pp. 3681–3690.
[195] A. Hundt, V. Jain, G.D. Hager, SharpDARTS: Faster and more accurate differentiable architecture search, 2019, https://arxiv.org/pdf/1903.09900.pdf.
[196] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A.L. Yuille, L. Fei-Fei, Auto-Deeplab: Hierarchical neural architecture search for semantic image segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 82–92.
[197] H. Cai, J. Yang, W. Zhang, S. Han, Y. Yu, Path-level network transformation for efficient architecture search, in: International Conference on Machine Learning, PMLR, New York, 2018, pp. 678–687.
[198] B. Baker, O. Gupta, N. Naik, R. Raskar, Designing neural network architectures using reinforcement learning, 2016, https://arxiv.org/pdf/1611.02167.pdf.
[199] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2018, pp. 8697–8710.
[200] R. Istrate, F. Scheidegger, G. Mariani, D. Nikolopoulos, C. Bekas, A.C.I. Malossi, TAPAS: Train-less accuracy predictor for architecture search, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2019, pp. 3927–3934.
[201] M.G. Kendall, A new measure of rank correlation, Biometrika 30 (1/2) (1938) 81–93.
[202] M. Da Silva, R. Regnier, M. Makarov, G. Avrin, D. Dumur, Evaluation of intelligent collaborative robots: A review, in: Proceedings of the 2023 IEEE/SICE International Symposium on System Integration (SII), IEEE, 2023, pp. 1–7.
[203] J. Mišeikis, P. Caroni, P. Duchamp, A. Gasser, R. Marko, N. Mišeikiene, F. Zwilling, C. de Castelbajac, L. Eicher, M. Früh, et al., Lio-A personal robot assistant for human-robot interaction and care applications, IEEE Robotics and Automation Letters 5 (4) (2020) 5339–5346.
[204] M.J. Tamasi, R.A. Patel, C.H. Borca, S. Kosuri, H. Mugnier, R. Upadhya, N.S. Murthy, M.A. Webb, A.J. Gormley, Machine learning on a robotic platform for the design of polymer-protein hybrids, Advanced Materials 34 (30) (2022) 2201809.
[205] C. Roche, P.J. Wall, D. Lewis, Ethics and diversity in artificial intelligence policies, strategies and initiatives, AI and Ethics (2022) 1–21.

Imrus Salehin earned his B.S. degree in Computer Science and Engineering from Daffodil International University, Dhaka, Bangladesh, in 2021. Currently, he is pursuing an M.S. degree in Computer Engineering at Dongseo University, Busan, South Korea. Imrus serves as a Graduate Research Assistant (RA) at the Machine Learning/Deep Learning Research Lab in Busan, South Korea. His contributions to research are notable, with publications in esteemed national and international journals and conferences. He also actively participates as a reviewer for several reputable conferences and journals. His primary research interests encompass Deep Learning, Machine Learning, AutoML, Data Science, Transfer Learning, and IoT, with additional focus areas in AI-based Energy and Computer Vision. To date, he has authored more than 07 journal articles and 13 conference papers. In recognition of his outstanding research contributions, he was honored with the title of Researcher by Daffodil International University in 2021 & 2022. Email:deeplab43@gmail.com.

Md. Shamiul Islam received a B.S. degree in Computer Science and Engineering from Bangladesh University of Business & Technology, Dhaka, Bangladesh in 2023. His main research interest fields are Deep Learning, Machine Learning, AutoML, Cyber Security, Transfer Learning and IoT.

Pritom Saha received his Bachelor of Science degree from the Department of Computer Science and Engineering at Daffodil International University, Dhaka, Bangladesh in 2022. His research interest lies in machine learning, deep learning, data analysis, and IoT. Currently, he is collaborating with many researchers in the field of computer science.Email: pritom15-1807@diu.edu.bd.

S. M. Noman obtained his Bachelor of Science in Engineering degree from the Department of Electrical and Electronic Engineering at Daffodil International University. Currently, he is studying MEng in Renewable Energy at the Frankfurt University of Applied Science, Germany. He has completed a PMP course and more than three years of practical experience in energy and sustainable development projects. He was a Student Member of IEEE, IEEE Power & Energy Society and worked in different social volunteer activities. He has a deep passion for AI and energy research technology. He has published a reputed journal article and international conference papers. His research specialty and interests include Renewable Energy, Grid Integration and Smart Grids, Energy Efficiency, Energy Policy and Economics, Deep Learning, Machine Learning and IoT systems. He received recognition from the research division of Daffodil International University in 2021 for a fabulous research publication.

Azra Tuni obtained her Bachelor of Science in Computer Science and Engineering, from the University of Asia Pacific, Dhaka, Bangladesh. Her research interest lies in Machine learning, Deep learning, and data analysis.

Md. Mehedi Hasan is a Computer Science graduate from Ahsanullah University of Science and Technology, Dhaka, Bangladesh. He is now working as a software engineer in a renowned IT company. Passionately drawn to machine learning, he is now immersing himself in research focused on machine learning and deep learning. He aspires to contribute not just to software development but also to cutting-edge innovations in technology, making a mark in the field of Machine learning.

Md. Abu Baten is a final semester undergraduate student at Northern University Bangladesh, Dhaka, Bangladesh, majoring in Computer Science and Engineering. With a profound interest in machine learning and deep learning, he has actively engaged in coursework and projects, demonstrating a strong commitment to advancing artificial intelligence. His dedication to staying updated with the latest developments in these fields, coupled with hands-on experience, positions him as an aspiring researcher ready to contribute to the ever-evolving landscape of Machine learning and Deep learning.