NEURAL ARCHITECTURE SEARCH FOR EFFICIENT DEEP LEARNING HARDWARE

Swarnima Joshi, Jaishyam Reddy Reddivari, Aditya Ghadge, Anusha Jeevan Prakash

***Abstract*— Automating the process of designing neural network architectures has revolutionized the field of deep learning and the reason behind it, Neural Architecture Search (NAS). Through this paper we explore the evolution of NAS from pure software-based architectures to hardware-aware and co-design approaches. We dive deep into how NAS works by exploring its building blocks such as search space, search strategy, and estimation strategy. Moreover we investigate how NAS is used for distinct hardware platforms, emphasizing its critical role in tailoring neural networks to suit diverse devices. Not only do we discuss the benefits and its role in accelerating innovation by improving performance, and overcoming computational complexity, we also tackle challenges associated with it. Real-life use cases of NAS across different domains, such as speech recognition on edge devices, autonomous driving, mobile image super-resolution, medical imaging and Google's AutoML are perfect examples to demonstrate its transformative potential in solving real-world problems.**

I. INTRODUCTION

Neural Architecture Search (NAS) has emerged as a groundbreaking technique in the field of deep learning, offering automated solutions for crafting neural network architectures. Manually crafting neural network architectures involves domain expertise, intuition, and iterative experimentation to design architectures suitable for specific tasks. This process includes problem understanding, architecture selection, design, hyperparameter tuning, training, evaluation, and refinement. However, it is labor-intensive, time-consuming, and limited by human biases. NAS automates this process by employing computational algorithms to explore a vast search space of architectures. The algorithms systematically search for optimal architectures tailored to the task and constraints, utilizing techniques such as reinforcement learning, evolutionary algorithms, and gradient-based optimization. This is what we will be discussing in this paper, how it evolved from a basic software-based approach to being hardware efficient. Challenges and benefits along with real-life use cases spanning diverse domains will be explored in detail and we will showcase its ability to bring about significant change in enhancing the efficiency and abilities of deep learning technologies.

II. METHODOLOGY

In this section we will first explore the evolution of NAS, then we will study about its building blocks and finally a framework analysis to understand its theoretical underpinnings.

A. NAS Evolution: From Pure-Software NAS to Co-Design NAS via Hardware-Aware NAS

We will briefly discuss about how NAS has evolved from a pure software based architecture to involving hardware compatibility within it.

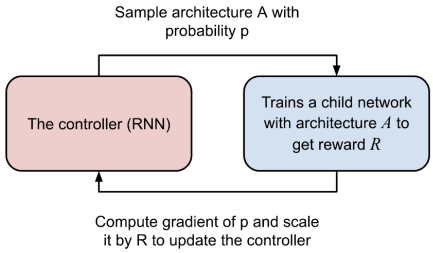

Fig. 1. Pure-Software NAS. Credit: Lu-Jiang

1) Pure-Software NAS: This phase focuses solely on exploring the structure of neural networks. A controller, usually an RNN (Recurrent Neural Network) [1], generates a child network and trains it to obtain its accuracy. This accuracy is then used as feedback for the controller to improve in the next iteration. The search ends when the controller reaches the maximum accuracy or meets a termination condition. So, this method is majorly based on trial and error untill a good enough architecture is found and hence is labor-intensive, time-consuming, and limited by human biases.

2) Hardware-Aware NAS: This phase takes into account the efficiency of the neural network on a fixed hardware platform, such as mobile phones or desktops. The framework tests each child network for hardware efficiency like latency and energy consumption and combines this information with accuracy to update the controller. The goal is to find a neural network architecture that not only performs well in terms of accuracy but also meets hardware specifications to upgrade every performance metrics.

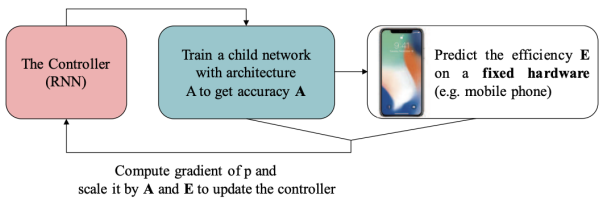

Fig. 2. Hardware aware NAS. Credit: lilianweng

Hardware-aware Neural Architecture Search (HW-NAS)

has emerged as one of the most promising techniques to automatically generate efficient CNN models accomplishing acceptable accuracy-performance tradeoffs. HW-NAS algorithms optimizes the accuracy and hardware execution metrics such as latency, energy, size, and so on while exploring the search space of a CNN. It is noticed through various works [2][3] that a model searched or optimized for a given hardware may or may not be efficient on different hardware. For example, FBNet [4] showed that the when a CNN model was searched with respect to iPhoneX, it took 19.84 ms to execute, but the latency increased to 23.33 ms on Samsung S8. However, the CNN architecture searched for Samsung S8 executes with a latency of 22.12 ms, but consumes 27.53 ms on iPhoneX.

3) Co-Design NAS: This current phase involves optimizing both the neural network architecture and the hardware design simultaneously. The framework conducts a two-level exploration, consisting of both rapid and gradual iterations, to quickly identify and remove architectures with inferior hardware efficiency, followed by training and refinement of the more promising candidates.

To summarize this up, NAS has grown from just focusing on the design of neural networks to considering how well these designs work on specific hardware and now to optimizing both the network and hardware together. Hence, the most recent development is about finding the best combination of network design and hardware setup, so that the final system is both accurate and efficient according to specific requirements like cost, energy use, or reliability.

B. Building blocks of NAS

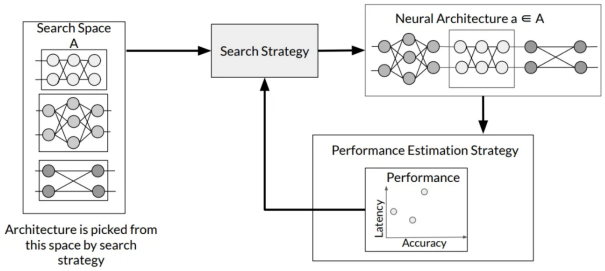

Fig. 3. NAS building blocks. Credit: deeplearning.ai

NAS components are divided into 3 dimensions. We will look into each one of them in detail:

1) Search Space: This defines the pool of all possible network architectures that can be considered for a given task.

For example, we have different types of layers and each layer type serves a specific function and is appropriate for specific kinds of tasks within a given network.

Convolutional Layers : Primarily used in convolutional neural networks (CNNs) [5], often applied to image processing tasks, these layer apply a convolution operation to the input, effectively sliding a filter over the input data to create a feature map that summarizes the presence of detected features in the input.

Recurrent Layers: Recurrent layers are used in recurrent neural networks (RNNs), designed to handle sequential data, such as time series or text which have connections that feed the output of a neuron back into itself, allowing the network to maintain a memory of previous inputs, influencing the current output.

Each of these layer types has specific hyperparameters [6] associated with them. The search space for a neural network architecture includes all the possible configurations of these layers, connection between them and their hyperparameters, allowing for a wide variety of network designs to be explored.

There are two main types of search spaces, macro and micro.

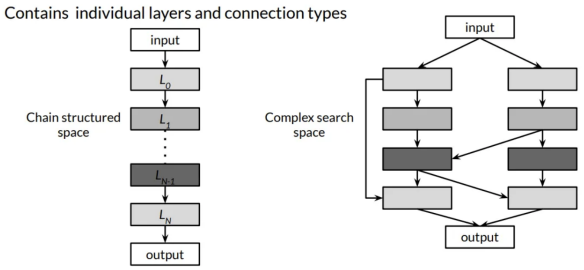

Fig. 4. Macro search space. Credit: deeplearning.ai

Macro: Macro search space is where you find all the different layers and connection types in a neural network. Neural architecture search explores this space to find the best model, building it up layer by layer. You can make networks by stacking layers one after another (called a chain structured space) or by using multiple branches and skip connections for a more complex setup.

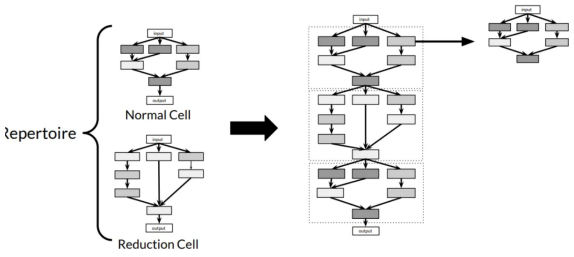

Fig. 5. Micro search space. Credit: deeplearning.ai

Micro: In a micro search space, neural architecture search constructs a neural network using cells, with each cell representing a smaller network. There are two main types of cells, a normal cell and a reduction cell. These cells are stacked together to create the final network. We can see the notable performance benefits with this method compared to the macro approach.

2) Search strategy: This entails the algorithm used to explore the search space and select the appropriate neural architecture. And how does NAS do it? The algorithm optimizes the performance metrics of child models, drawn from a

sample of the population of network candidates, as rewards to generate high-performance architecture candidates. There are various methods used to optimize search strategies in order to make the process deliver efficient results with consistency. We will discuss some of them below:

Grid Search: In this method we search everything, meaning we cover everything we have in search base.

Random Search: In random search, as the name suggests we select the next option randomly within the given search space.

While both of these work quite well given a search base is smaller, we cannot see the same results when the search space is larger. This is when other strategies like Bayesian Optimization, Reinforcement Learning, Evolutionary Algorithms were introduced.

Reinforcement Learning: Reinforcement Learning is a search technique where machine learning models learn by making a series of decisions. The goal is to find the best solution to a problem by maximizing rewards and minimizing penalties. Through trial and error, the model learns to navigate complex environments and select configurations that result in better neural networks for NAS.

In 2018, Chi-hung Hsu introduced an important advancement in NAS called Multi-Objective Neural Architecture Search (MONAS) [9] with Reinforcement Learning. This approach addresses key challenges in NAS, such as computational and memory constraints, by optimizing for scalability, accuracy, and power consumption.

Bayesian Optimization: When working with model architectures, we often use a common probability distribution, usually a Gaussian one, to understand how well different architectures perform. We gather data from tested architectures to shape this distribution, which helps us decide on the next option to try. This allows us to build up an architecture based on the test results and the constrained distribution.

Evolutionary Algorithms: First, we create an initial pool of n different model architectures randomly. Each architecture's performance is then evaluated using a performance estimation strategy. Next, the top X performers are chosen as parents for the next generation. These new architectures can either be slightly modified versions of their parents, through random alterations or mutations, or they can be combinations of the parents.

The performance of the offspring is assessed using the same performance estimation strategy. Possible mutations can include actions like adding or removing a layer, connection, or changing the size of a layer or other hyperparameters. Afterward, Y architectures are selected to be removed from the population. This selection can be based on factors like being the worst performers, the oldest individuals, or a combination of both. The offspring then replace the removed architectures, and the process starts again with this new population.

3) Estimation strategy: In order to generate better performing architecture, NAS requires search strategies to estimate the performance of generated architectures. Below we discuss some of those performance estimation strategies:

Lower Fidelity estimates: The objective is to reduce the training time by simplifying the problem. This can involve training on a smaller subset of data, employing fewer filters and cells in each layer or using lower resolution images. This as it can be observed often leads to underestimating performance even though it significantly cuts down on computational resources.

Learning Curve Extrapolation: This relies on the assumption that you possess reliable mechanisms to predict the learning curve. Therefore, extrapolation becomes a sensitive and valid choice in this context.

Weight Inheritance(Network morphism): Weight Inheritance or Network Morphism is like passing on knowledge from one trained network to another. It's similar to how we reuse what we've learned in one task to help us in another. With Network Morphism, we change the architecture of the network while keeping its main function the same. This helps the new network learn faster because it starts with some useful knowledge from the old one. This method saves time by needing only a few days on a computer to design and test the new network. And it allows us to make the network bigger over time without needing to start training from the beginning each time. Also, networks can be designed without any fixed limit on their size, giving us more freedom in designing and improving them.

C. NAS FOR SPECIFIC HARDWARE

Now that we've delved into the foundational components and workings of NAS, let's examine its integration with hardware.

1) CPU-AWARE NEURAL ARCHITECTURE SEARCH:

a) Mobile/Edge CPUs: Modern CNN models are great but can be quite demanding in terms of computational power and memory. This is a problem for mobile devices which have limited resources. To solve this, researchers have come up with ways to create efficient CNN models specifically for mobile CPUs using different search methods:

- RL-based Methods: MnasNet [10] was a pioneer in mobile-aware NAS. It used reinforcement learning to consider both the network's accuracy and how fast it runs on mobile devices. MobileNetV3 [11] improved upon this by adding smart connections within the network and tweaking the number of filters for better performance.
- Gradient-based Methods: ProxylessNAS-mobile [12] used the latency of each layer in the loss function, which is a measure of the network's performance. It used a model to predict how long each operation would take, helping it find a good network that's both accurate and fast on mobile.
- Evolutionary Search Methods: Once-For-All (OFA) [13] trained a big, flexible network and then created smaller versions optimized for different devices like mobile phones and GPUs. It's efficient because you train the big network once and then customize it for wherever you need to deploy it.

b) Desktop/High-end CPUs: CPUs are also crucial for running DNNs on laptops, desktops, and servers. Here are a few methods developed for these kinds of CPUs:

- ProxylessNAS-CPU: This method [12] was designed for a specific Intel CPU and prefers smaller, simpler network structures because desktop CPUs can't handle as many operations at once as GPUs can.
- EfficientNet: This approach [**?**] scales up a network's size uniformly until it meets a certain speed requirement, rather than searching for a large model from the very start. It balances three factors: the number of filters, the number of layers, and the input image size.
- LA-DNAS and HardCoRe-NAS: These methods [15] involve predicting how long a network will take to run and then incorporating that prediction into the search process to find efficient networks for desktop CPUs.

2) GPU-Aware Neural Architecture Search: GPUs are super powerful and can handle thousands of operations at once, making them great for running DNNs. Here are some methods specifically for GPUs:

- High-performance GPUs: Models like ProxylessNAS-GPU and EfficientNetV2 prefer wider networks with more operations in the early layers because GPUs can handle that parallelism. They aim to be both accurate and fast on high-performance GPUs.
- Energy-efficient Models on GPU: Some methods, like MONAS, focus on designing CNNs that are not only accurate but also consume less energy.
- Multi-GPU Platform: When using multiple GPUs, it's important to design networks that can efficiently run in parallel. ColocNAS [16], for example, restructures traditional cell-based networks to reduce communication overhead and speed up performance across multiple GPUs.
- Edge/Mobile GPUs: EdgeNAS and MoGA NAS [**?**] aim to create efficient models for edge GPU devices, which are less powerful than high-end GPUs but still need to be fast and accurate for tasks like image recognition on mobile phones.

All these methods are about finding the right architecture for neural networks that balances speed, accuracy, and sometimes energy efficiency, depending on the hardware they're intended to run on. It's like tailoring a suit, you want the perfect fit for the occasion, a device in this case!

III. Benefits and Challenges

Comprehending the advantages and drawbacks of Neural Architecture Search (NAS) for effective deep learning is crucial to appreciating its importance in the field's advancement. Numerous benefits of NAS, including enhanced performance, automation, and hardware-aware design, simplify the process of creating neural network topologies. To fully realize the potential of NAS, however, obstacles including processing complexity, assessment bottlenecks, and limited interpretability must be overcome. To fully understand the opportunities and complications that NAS presents for effective deep learning, let's take a closer look at these areas.

A. Benefits

1) Automation: Neural Architecture Search (NAS) automation offers numerous important benefits for effective deep learning. First of all, it cuts down on the amount of time needed for architectural design, enabling researchers to quickly investigate a variety of combinations. Because of the time and resource savings resulting from this efficiency, researchers are able to concentrate on higher-level tasks like issue formulation and result interpretation. Automation also guarantees that architectural design is consistent and repeatable throughout several tests, which improves the validity of study results. Scalability is made easier since automated search techniques can change to accommodate datasets and structures that become more complicated. Accessibility is enhanced because advanced model designs are now more widely available to a wider audience thanks to user-friendly NAS frameworks. Automation also makes it possible to develop architectural design iteratively, progressively arriving at ideal solutions. In the end, automated NAS speeds up the transfer from lab to practical implementation, propelling advances in deep learning applications and research.

2) Hardware-Aware Design: Neural network designs can be customized to specific hardware restrictions and optimizations via NAS algorithms by incorporating knowledge of underlying hardware architectures during the search process. With this method, the identified architectures are guaranteed to be optimized for target hardware platforms—CPUs, GPUs, or specialized accelerators like TPUs or FPGAs—for deployment. By considering variables like memory bandwidth, processing power, and energy efficiency, hardware-aware NAS approaches allow for the development of models that optimize performance while consuming the fewest resources possible. This enhancement makes deep learning models more useful and efficient in actual applications, especially in resource-constrained settings such as mobile devices, edge computing devices, and Internet of Things devices. NAS enhances the effectiveness, scalability, and suitability of deep learning systems across a variety of hardware platforms and deployment circumstances by creating architectures that are cognizant of the hardware.

3) Improved Performance : Neural network architectures that outperform those that are manually developed can be found using NAS algorithms through automation of the architectural design process. Depending on the demands of the particular activity, these architectures are adjusted to reach higher levels of accuracy, speed, or resource utilization. By thoroughly investigating the architectural space, NAS finds new configurations that minimize overfitting and efficiently capture significant patterns and features in the data. Because of this, deep learning models produced using NAS perform better than models built with standard architectures, which increases their usefulness in a range of applications like computer vision, natural language processing, and reinforcement learning. Better accuracy, quicker inference times, and an improved overall user experience are just a few of the concrete advantages that result from this increased

performance for end users.

4) Automated Model Design: By automating the process of finding optimal or nearly ideal configurations, NAS simplifies the complex process of developing neural network topologies. Through the use of search algorithms, NAS looks at a wide range of possible architectures and chooses ones that meet predetermined standards like scalability, accuracy, and efficiency. Due to this automation, researchers and practitioners are spared the tedious manual experimentation, which greatly lessens their workload. Alternatively, they could concentrate on more advanced assignments like formulating problems and analyzing outcomes. Furthermore, automated model construction guarantees that results are consistent and repeatable from experiment to experiment, which increases the validity of study conclusions. In the end, NAS spurs deep learning innovation by facilitating quick iterations and architectural configuration exploration, which results in the identification of innovative and effective models suited to certain objectives and limitations.

5) Resource Efficiency: The goal of NAS algorithms is to identify neural network topologies that optimize resource usage while maintaining performance. NAS finds architectures that optimize energy usage, reduce memory footprint, and increase compute efficiency by automating the search process. When implementing deep learning models on resource-constrained devices like mobile phones, edge devices, and Internet of Things sensors, this emphasis on resource efficiency is especially important. Through the customization of designs to certain hardware platforms and limitations, NAS guarantees that models can function well in practical situations without sacrificing performance. Additionally, by lowering the requirement for pricey computing resources and energy consumption, resource-efficient architectures contribute to cost savings and environmental sustainability. All things considered, the focus on resource efficiency in NAS makes it possible to create realistic and long-lasting deep learning solutions that can be used at scale in a variety of contexts and applications.

6) Generalization: The goal of NAS algorithms is to identify neural network topologies that exhibit strong performance on training data and good generalization to unknown data. NAS investigates a wide range of configurations that capture underlying patterns and features in the data without overfitting by automating the search for architectures. By placing a strong emphasis on generalization, the resulting models are made to be resilient and flexible across a variety of datasets, tasks, and contexts. Consequently, NAS aids in the creation of deep learning models that demonstrate exceptional performance and dependability in practical applications. The process of generalization allows models produced by NAS to continue to exhibit high levels of accuracy and efficacy over a range of scenarios, hence augmenting their practical usability and worth across a multitude of disciplines, including computer vision, natural language processing, and reinforcement learning.

7) Accelerating Innovation: NAS speeds up the investigation and testing of new thoughts and ideas in deep learning by automating the architectural design process. Researchers can investigate novel concepts, methods, and paradigms more effectively because they can quickly iterate across different architectural setups. Because of the rapid speed of invention, researchers are able to push the limits of deep learning, which results in the creation of unique architectures that are more successful at solving complicated issues. Additionally, NAS promotes cooperation and knowledge sharing by facilitating the exchange of creative ideas among researchers. NAS enables academics to concentrate on high-impact research topics and pursue ambitious research goals by decreasing the time and effort needed for architectural design. This, in turn, propels developments in deep learning and advances technological progress across multiple domains.

8) Reduced Human Bias: NAS reduces the impact of human biases and preferences that could unintentionally affect manual design decisions by automating the architectural design process. The intuition and expertise of researchers are frequently relied upon in traditional approaches to architecture design, which might create subjective biases and restrictions. On the other hand, NAS algorithms adhere to predetermined evaluation criteria and search protocols, which results in more unbiased and objective architecture selection conclusions. Fairness, openness, and reproducibility in deep learning research are enhanced by this decrease in human bias. NAS makes it easier to explore novel and potentially ground-breaking architectural arrangements that could otherwise go unnoticed by reducing the influence of human prejudices. This encourages creativity and diversity in the field of deep learning research, which eventually results in the creation of neural network architectures that are more reliable, effective, and efficient for a range of uses.

9) Accessible to Non-Experts: As deep learning expertise varies, NAS approaches are becoming increasingly accessible and easy to utilize for researchers and practitioners. The process of conducting architectural search experiments is made simpler by user-friendly NAS frameworks and tools, which abstract away the difficulties of algorithm implementation and hyperparameter adjustment. Modern model designs and methodologies can be utilized by people without deep learning expertise because of the democratization of architecture design. Without requiring deep technical knowledge, non-experts can effectively explore architectural configurations suited to their particular requirements and restrictions by using NAS algorithms. By reducing the entry barrier, NAS encourages cooperation, creativity, and information exchange among members of the deep learning community by enabling a wider audience to participate in architectural design and experimentation.

B. Challenges

1) Computational Cost: To search through the enormous search space of possible topologies, NAS algorithms frequently need a significant amount of processing power and time. The magnitude of the dataset and the complexity of the neural network architectures under consideration both raise the search process' complexity. The scalability and usability

of NAS can be hampered by high computational costs, which may be unaffordable for academics and practitioners with limited access to powerful hardware or cloud computing resources. Furthermore, as researchers repeat trials to refine search parameters and assess candidate structures, the computational cost might increase quickly. In order to overcome the computational cost challenge in NAS, more effective search algorithms, parallelization methods, and optimization strategies must be developed. These will speed up the search process and use less resources, which will ultimately make NAS more scalable and available to a wider range of users.

2) Evaluation Bottleneck: An important part of the NAS process is assessing the candidate designs' performance, which directs the search for architectures that satisfy predetermined standards like accuracy, efficiency, and generalization. But evaluating structures can be costly and time-consuming computationally, particularly when working with large-scale information and intricate neural network architectures. The entire search process may be considerably slowed down by this evaluation bottleneck, which would restrict the NAS algorithms' capacity to scale and perform well. Another problem is creating efficient evaluation criteria that effectively capture different facets of model performance. The creation of effective evaluation techniques, such as surrogate models, early stopping criteria, and hardware-accelerated evaluation techniques, is necessary to address the evaluation bottleneck in NAS in order to speed up the search and lower computational overhead. Researchers can accelerate the NAS procedure and find the best architectures for a variety of deep learning applications by reducing the assessment bottleneck.

3) Limited Interpretability: Although NAS algorithms streamline the architectural design process, the resulting designs could not be easily interpreted, which makes it difficult for academics to comprehend the underlying choices and workings that underpin their effectiveness. Researchers' capacity to learn about architectural trends, spot possible areas for development, and transfer information between tasks or domains is hampered by this limited interpretability. Furthermore, in real-world applications like healthcare and finance where interpretability and explainability are crucial, the absence of interpretability may make it more difficult for NAS approaches to be adopted. The difficulty of limited interpretability in NAS calls for the creation of methods to improve the search processes and the resulting architectures' transparency and explainability. This comprises techniques for understanding learnt representations, finding important architectural elements that affect model performance, and visualizing and assessing architectural configurations. Enhancing interpretability can help researchers better comprehend the architectures found by NAS, which will help them make better decisions and make it easier to implement effective deep learning models in real-world applications.

4) Computational Complexity: It takes a lot of time and computer power to explore the huge space of possible structures. The computational needs of NAS methods rise in tandem with the complexity and size of deep learning models and datasets. In particular, for practitioners and researchers with restricted access to high-performance computer equipment, this computational complexity may become prohibitive. Furthermore, processing limitations may limit the scalability of NAS methods, making it more difficult for them to effectively search enormous search spaces and find ideal structures. The creation of more effective search algorithms, parallelization methods, and optimization tactics is necessary to overcome the computational complexity problem in NAS and speed up the search process. Through the reduction of computational complexity, scientists can increase the accessibility and scalability of network area storage (NAS) and facilitate the development of effective deep learning architectures for various applications.

IV. Real Life Use Cases of NAS

Exploring the practical uses of Neural Architecture Search (NAS) reveals its revolutionary potential across multiple fields. Through the automation of neural network architecture design, NAS provides creative answers to challenging issues in a variety of sectors. From improving medical imaging to transforming autonomous driving, NAS shows how adaptable and effective technology is at solving problems in the real world. Let's examine a few prominent use cases where NAS has advanced significantly, demonstrating its influence on real-world uses and scientific discoveries.

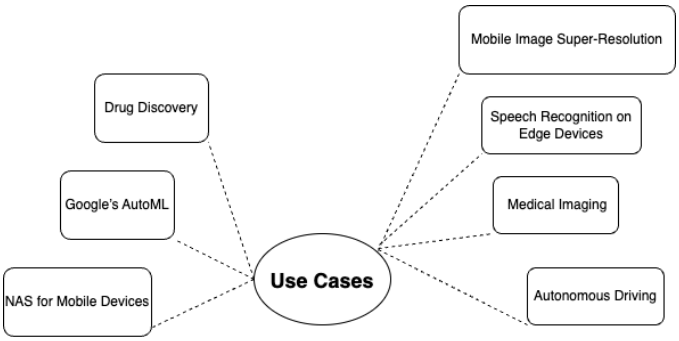

Fig. 6. NAS use cases

A. *Mobile Image Super-Resolution*

The need for mobile applications that improve the quality of photos taken on mobile devices is rising as smartphone photography becomes more and more common. The goal of mobile picture super-resolution is to enhance the visual quality and resolution of low-resolution photographs so that they are more suited for printing or high-resolution display viewing. Neural network topologies tailored for super-resolution tasks can be designed automatically using NAS algorithms, which strike a compromise between computing economy and accuracy to guarantee real-time performance on mobile devices. By using deep learning approaches, these architectures are able to learn intricate mappings between high-resolution and low-resolution picture spaces, which can effectively improve image clarity and restore fine features.

Through the utilization of Network Address Space, developers can produce mobile picture super-resolution applications that minimize processing resources and yield high-quality results, allowing users to edit images while on-the-go with little latency and battery usage.

B. Speech Recognition on Edge Devices

One important real-world application where NAS can have a big influence is speech recognition on edge devices. An increasing number of wearables, IoT devices, smartphones, and other devices with voice-based interfaces are being integrated, which has led to a desire for effective speech recognition systems that can run directly on edge devices without requiring cloud-based processing. Neural network architectures optimized for speech recognition tasks may be designed automatically by NAS, making them ideal for deployment on edge devices with limited resources. By balancing computational efficiency and accuracy, these designs allow for real-time speech recognition with low latency and energy usage. Through the use of NAS, developers can produce speech recognition systems that perform accurately and responsively on edge devices, improving user experience and opening the door for a variety of voice-activated applications, such as hands-free navigation systems, virtual assistants, and smart home appliances.

C. Medical Imaging

With the development of deep learning algorithms, medical imaging has made significant strides in areas including image-guided interventions, illness diagnosis, and treatment planning. Neural network topologies customized for certain medical imaging tasks, like organ segmentation, disease categorization, and lesion detection, can be automatically designed using NAS. These designs can be tuned to run well on medical imaging devices with constrained computational resources, while achieving good accuracy and robustness. Researchers and medical practitioners can create sophisticated medical imaging systems that enhance diagnosis precision, enable early disease identification, and customize treatment plans by utilizing neural information systems. Furthermore, NAS can facilitate the incorporation of artificial intelligence into clinical processes and hasten the creation of novel imaging procedures, both of which will eventually improve patient outcomes and advance the field of medical imaging.

D. Autonomous Driving

Autonomous vehicles are becoming a reality thanks to the quick developments in artificial intelligence and sensor technology, offering safer and more effective transportation options. Neural network architectures specifically designed for autonomous driving systems' perceptual tasks—like object identification, lane tracking, and traffic sign recognition—can be automatically designed by NAS. These systems provide real-time processing of sensor data from cameras, radar, and LiDAR by balancing precision, dependability, and computational efficiency. Developers can improve the safety and dependability of autonomous vehicles by utilizing NAS to build strong and flexible perception systems that let them successfully traverse challenging situations and communicate with other drivers. Furthermore, by supporting the creation of sophisticated perception algorithms and aiding the quick deployment of autonomous cars on a greater scale, NAS can spur innovation in autonomous driving technology and ultimately change the face of transportation.

E. Drug Discovery

There is a growing need for effective ways to speed up the drug discovery process due to the rising need for novel treatments to treat complex ailments. Neural network topologies suited for tasks like chemical screening, drug-target interaction prediction, and molecular property prediction can be designed automatically using NAS. By customizing these designs, one can accurately predict pharmacological characteristics and biological activities by extracting significant aspects from molecular data. Utilizing NAS, scientists can find new drug candidates more quickly, rank compounds for experimental validation, and tailor drug designs to target specific molecules. Furthermore, NAS can make it easier for artificial intelligence to be included into drug discovery pipelines, which will speed up the creation of individualized medications and make it possible to find new drug candidates. All things considered, NAS has the power to completely transform the drug discovery process by facilitating the development of safer, more efficient, and more precisely focused treatments for a variety of illnesses.

F. Google's AutoML

Google created the AutoML [22] set of machine learning tools, which automates the model training, hyperparameter tuning, and architecture search processes. One well-known aspect of AutoML is AutoML Vision, which lets users build unique picture categorization models without needing a lot of experience with machine learning. In the background, AutoML uses NAS techniques to find the best neural network topologies on its own for different picture categorization tasks. Because this automated method does not require human model design and tuning, a wider range of users—including non-experts and domain specialists—can utilize it. AutoML can effectively explore architectural configurations and find high-performing models that satisfy the user's needs for precision, speed, and resource efficiency by utilizing NAS. Therefore, AutoML eliminates the need for specialist understanding in data science or deep learning and enables users to make use of cutting-edge machine learning techniques for a variety of applications, such as object detection, picture recognition, and visual inspection.

G. EfficientNet

It stands for a class of convolutional neural network architectures with accuracy and computational efficiency as its primary goals. EfficientNet, created by Google researchers, uses NAS techniques to automatically find the best model topologies at various depths and scales. Using less

computing power than conventional architectures, EfficientNet's automated method allows it to achieve state-of-the-art performance on a variety of computer vision applications, such as semantic segmentation, object detection, and image classification. EfficientNet balances model complexity and computational cost by effectively exploring the architectural space, which makes it ideal for deployment on resource-constrained devices like edge servers, mobile phones, and Internet of Things devices. Consequently, EfficientNet has gained widespread traction in both academia and business, showcasing the usefulness of NAS in propelling deep learning breakthroughs and facilitating the creation of effective and scalable artificial intelligence systems for a range of applications.

H. NAS for Mobile Devices

Mobile devices' Neural Architecture Search demonstrates how automated architecture design may be used to optimize deep learning models for deployment on devices with limited resources. The increasing number of smartphones and other mobile devices has led to a need for effective artificial intelligence solutions that can function well with constrained processing resources. NAS for mobile devices streamlines the process of finding neural network topologies that balance performance and accuracy so that models may function flawlessly on mobile hardware and produce high-quality output. Developers may create unique architectures for mobile applications like augmented reality, natural language processing, and picture recognition by utilizing NAS. By enabling mobile apps to handle sophisticated AI tasks locally instead of relying on cloud-based processing, these optimized architectures help reduce latency, preserve battery life, and protect user privacy. With the help of NAS for mobile devices, developers may produce creative and resource-efficient AI-powered apps that improve user experience and broaden the scope of mobile technology.

V. CONCLUSIONS

The literature discussion on Neural Architecture Search highlights the technology's deep theoretical understanding and signals its potential to revolutionize hardware design. These academic publications we researched emphasize the importance of NAS-generated architectures and their ability to navigate diverse hardware constraints, envisioning a future with flexible neural architectures suitable for different computing scenarios.However, the study does not hold back when discussing the theoretical dead ends and puzzles surrounding NAS. As it works through these nuances, the discussion highlights the difficult problems caused by scalability and the intricate complexities of search spaces.Despite theoretical challenges, the benefits of NAS for hardware optimization justify further research and development efforts.In conclusion, our findings underscore the pivotal role of NAS in propelling the advancement of deep learning technology. The automation of neural network architecture discovery through NAS could significantly enhance the efficiency and success of deep learning methods. Looking ahead, NAS is poised to play a crucial role in facilitating the continued development of artificial intelligence by unlocking new thresholds of performance and accessibility.

ACKNOWLEDGMENT

We extend our deepest gratitude to Professor Saman Kumarawadu for his invaluable guidance and support, not only through this research development but throughout this academic journey. His mentorship has been instrumental in shaping our conceptual understanding about the field and are truly grateful for his encouragement and inspiration in enriching our learning experience immeasurably.

REFERENCES

[1] GeeksforGeeks. Dec 2023. Introduction to Recurrent Neural Networks. Retrieved from https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[2] Bichen Wu et al. 2019. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[3] Han Cai et al. 2018. ProxylessNAS: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018).

[4] Bichen Wu et al. 2019. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In IEEE/CVF Conference on Computer Vision and Pattern Recognition. 10734–10742.

[5] DataCamp. Nov 2023. Introduction to Convolutional Neural Networks. Retrieved from https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

[6] Amazon Web Services. 2024. What is Hyperparameter Tuning? Retrieved from Hyperparameter

[7] Deci AI. 2024. Neural Architecture Search. Retrieved from https://deci.ai/neural-architecture-search/

[8] Ashwin Kumar. Apr 2, 2022. Neural Architecture Search. Retrieved from https://medium.com/@ashwinkumarjs/neural-architecture-search

[9] Chi-Hung Hsu et al. 2018. MONAS: Multi-objective neural architecture search using reinforcement learning.

[10] Mingxing Tan, Bo Chen, et al. 2019. MnasNet: Platform-aware neural architecture search for mobile. IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[11] Andrew Howard et al. 2019. Searching for MobileNetV3. In IEEE/CVF International Conference on Computer Vision.

[12] Han Cai et al. 2018. ProxylessNAS: Direct neural architecture search on target task and hardware.

[13] Han Cai et al. 2019. Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791 (2019).

[14] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks.International Conference on Machine Learning. PMLR, 6105–6114.

[15] Yuhui Xu et al. 2020. Latency-aware differentiable neural architecture search. arXiv preprint arXiv:2001.06392 (2020).

[16] Cheng Fu et al. 2020. Enhancing model parallelism in neural architecture search for multidevice system.

[17] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. 2020. MoGA: Searching beyond MobileNetV3. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

[18] Krishna Teja Chitty-Venkata and Arun K. Somani I. Iowa State University, USA. November 2022. Neural Architecture Search Survey: A Hardware Perspective.

[19] Qing Lu, Weiwen Jiang, Xiaowei Xu, Yiyu Shi, Jingtong Hu. November 4-7, 2019, Westminster, CO. On Neural Architecture Search for Resource-Constrained Hardware Platforms.

[20] Prasanna Balaprakash, Romain Egele, Misha Salim, Stefan Wild, Venkatram Vishwanath, Fangfang Xia, Tom Brettin, Rick Stevens. SC '19, November 17–22, 2019, Denver, CO, USA. Scalable Reinforcement-Learning-Based Neural Architecture Search for Cancer Deep Learning Research.

[21] Benmeziane et al. 22 jan 2021. A comprehensive survey on hardware-aware neural architecture search

[22] AutoMl. 2024. Retrieved from: https://www.automl.org/automl/