Exploring the Intersection between Neural Architecture Search and Continual Learning

Mohamed Shahawy*, Elhadj Benkhelifa†, and David White‡
Smarts Systems, AI and Cybersecurity Research Centre (SSAICS), Staffordshire University
Staffordshire, UK
Email: *mohamed.shahawy@research.staffs.ac.uk, †e.benkhelifa@staffs.ac.uk, ‡david.white1@staffs.ac.uk

***Abstract*—Despite the significant advances achieved in Deep Learning, the Deep Neural Networks' (DNNs) design approach remains notoriously tedious, depending primarily on intuition, experience and trial-and-error. This human-dependent process is often time-consuming and prone to errors. Furthermore, the models are generally bound to their training contexts, with no considerations to their surrounding environments. Continual adaptiveness and automation of neural networks is of paramount importance to several domains where model accessibility is limited after deployment (e.g. IoT devices, self-driving vehicles, etc.). Additionally, even accessible models require frequent maintenance post-deployment to overcome issues such as Data/Concept Drift, which can be cumbersome and restrictive. By leveraging and combining approaches from Neural Architecture Search (NAS) and Continual Learning (CL), more robust and adaptive agents can be developed. This study conducts the first extensive review on the intersection between NAS and CL, formalizing the prospective paradigm and outlining research directions for lifelong autonomous DNNs.**

***Index Terms*—Continual Neural Architecture Search, Adaptive Neural Networks, Neural Architecture Search, Continual Learning, Lifelong Learning, Incremental Learning, AutoML**

I. Introduction

Deep Learning has led to notable advances across a number of applications, specifically non-procedural tasks, such as image classification or maneuvering autonomous vehicles. By enabling computers to accomplish these tasks at a near (or sometimes even surpassing) human-level performance [1], tasks that previously relied on human-intervention can now become autonomous.

Despite all the achievements accomplished by Deep Learning, there remains a number of hurdles that keep these AI models from advancing further. Deep Neural Networks (DNNs) are traditionally designed by human experts, where the potentially hundreds of layers are manually specified (as showcased by some of the popular networks [2]–[5]). The architecture design process usually depends on the experts' educated intuition and a tedious trial-and-error process. As the tasks become more complex, the networks are normally expected to be deeper and thus their architectures become particularly challenging to design and optimize.

In addition to the extensive knowledge required to design neural architectures, humans tend to have a fixed thinking paradigm that limits the exploration of the solution space and leads to unvaried designs. These drawbacks of manual DNNs' design were further recognized when an automatically-tuned neural network (Auto-Net 1.0) won against human experts' designs in the ChaLearn AutoML challenge in 2015 [6]–[8]. The field of *Neural Architecture Search* (NAS) emerged to minimize or completely eliminate the limitations posed by the human factor in neural architecture design whilst providing more robust means of finding optimal topologies.

Another struggle modern AI faces lies in certain domains where the input changes over the course of time. Typically, DNNs tend to partially or completely overwrite learned knowledge upon retraining for new tasks, a phenomenon referred to as *Catastrophic Forgetting*, or *Catastrophic Interference* [10], [11]. To address this problem, the field of *Continual Learning* (CL) was introduced (sometimes referred to as *Lifelong Learning* [12]–[14], *(Task-/Domain-/Class-)Incremental Learning* [9], [15]–[17], and *Sequential Learning* [10], [18]).

Continual Learning aims to overcome Catastrophic Forgetting by retaining some form of implicit or explicit memory within the model, or by having continuous access to the tasks' datasets. Although complete prevention of information loss may not be attainable, the trade-off between generalizing the model to adapt to a continuous input stream and consolidating learned knowledge can be optimized. In other words, CL attempts to find a balanced solution on the stability-plasticity Pareto front, maximizing shared learning representations and adaptability to underlying non-stationary data distributions.

Being a subset of Online Learning, Continual Learning also addresses data distribution drifts for both single- and multi-task scenarios. Formally, given a data distribution P that incrementally changes over time t, $(P_{t1}, P_{t2}, ..., P_{t_n})$, the common types of distribution drifts are defined as:

1) *Data Drift*: $P_{t1}(x) \neq P_{t2}(x)$, i.e. the model's input distribution shifts
2) *Concept Drift*: $P_{t1}(y|x) \neq P_{t2}(y|x)$, i.e. the model's posterior probability distribution shifts
3) *Label Drift*: $P_{t1}(y) \neq P_{t2}(y)$, i.e. the model's prior probability distribution (label space) shifts

Where $P(\cdot)$ is the probability distribution, x is the input dataset, and y is the set of labels. Distribution drift poses a formidable challenge to Deep Learning models as it reflects the dynamic nature of the real world. More often than not, DNNs require consistent maintenance to sustain their performance and overcome distribution shifts, which adds potentially staggering expenses when using traditional methods (manual model redesign, retraining, evaluation and deployment for every iteration of data influx). In contrast, automating this process using CL and NAS not only reduces post-deployment

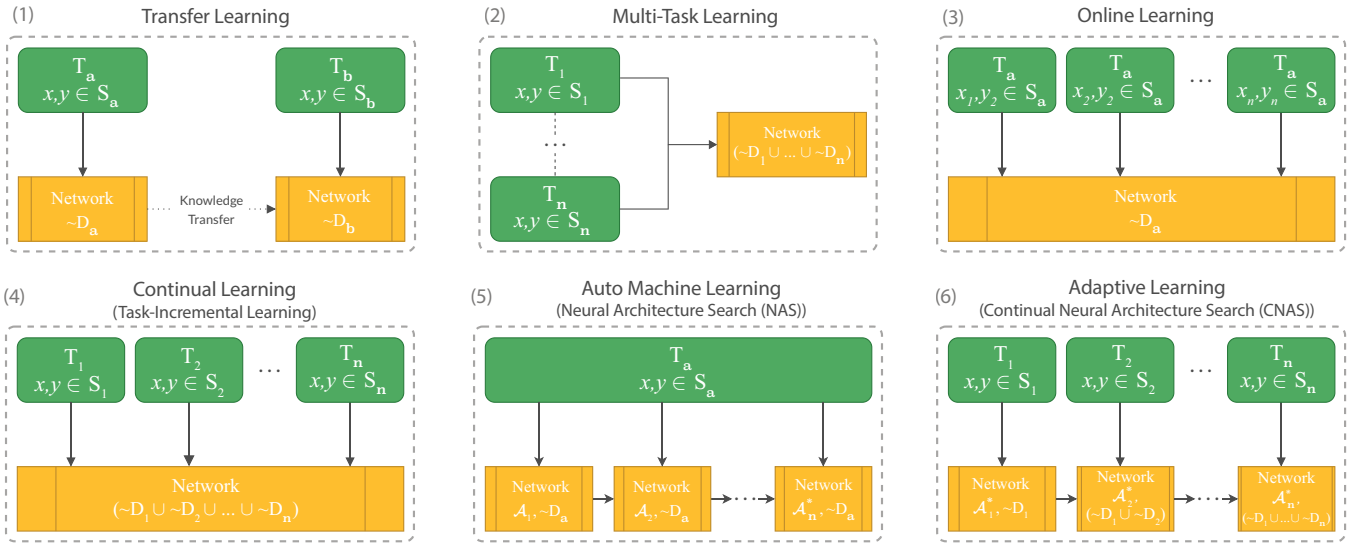

Fig. 1. Comparative illustrations for Adaptive Learning and its neighboring Machine Learning paradigms (inspired by [9]). The temporal component of each paradigm is represented in its diagram's X-axis (time increases left-to-right). The set of tasks, input data, and learned data distributions are given by T, S, and $\sim D$, respectively (sub-scripted by a constant for finite sets, numbered $1 \ldots n$ for variable-length sets). $\mathcal{A}_i^*$ represents a NAS-optimized architecture.

model maintenance costs, but also provides a more robust framework where the optimization process is not limited by human capacity.

A. Problem Description and Motivation

Continually adaptive features in DNNs are becoming increasingly crucial. Limited access to deployed models across several fields results in models that are bound to their fixed pre-deployment states and cannot fit their functionality to their changing surroundings. These fields include autonomous vehicles [19]–[25], robotics [26]–[33], Edge and IoT devices [34]–[37], and numerous others [38]–[44]. In a non-stationary environment, such static models are susceptible to a decline in performance due to distribution shifts.

In addition to the importance of Continual Learning in modern AI applications, the classical process of developing neural networks is heavily-reliant on manual human input; from defining a network architecture to tweaking hyperparameters, several parameters are dependent on experts' knowledge as well as trial-and-error [45]. The AutoML field was introduced to facilitate more robust Machine Learning solutions for non-experts by pipelining automated optimization techniques, such as NAS and Hyperparameter Optimization. This area, however, is still in its infancy and few studies have attempted to achieve an end-to-end self-developing DNN [46], [47].

Intuitively, the more statically-developed a neural network is (e.g. fixed network topology, fixed training learning representation, etc.), the less sensitive it will be to its surrounding environment. By minimizing or completely eliminating manual input from the development of DNNs, a network can become significantly less biased and thus more adaptive to fit the tasks' contexts. This concept of self-managing autonomous agents has been explored in the literature, however it is usually discussed within bounded domains (namely Neuroevolution and Unsupervised Learning) [12], [48], [49].

B. Aim and Contributions to Knowledge

To expand the narrow and dedicated form of intelligence exhibited by the traditional DNN development methodologies, we extend and generalize the inspiring ideas previously proposed beyond the scope of unsupervised learning and Evolutionary Algorithms. *Continual Neural Architecture Search* (CNAS) aims to ultimately eliminate human-intervention from the development and maintenance of DNNs by combining Neural Architecture Search and Continual Learning methodologies. Functional to that aim, this study:

1) Reviews Continual Learning approaches and proposes a concise categorization scheme for memory models
2) Reviews relevant works in the Neural Architecture Search field
3) Explores the intersection between the two domains, formalizes the automated lifelong supervised learning paradigm, and briefly reviews additional data automation techniques (hyperparameter optimization and autonomous data collection/pre-processing/feature engineering)
4) Identifies the gaps and potential future directions for NAS, CL, and their intersection

Although there are several reviews conducted on Continual Learning [9], [13], AutoML [50]–[52], and NAS [45], [53]–[55], the intersection between NAS and Continual Learning has not been formalized or reviewed despite the significant number of existing models in that area. To capture the scope of this study, Fig. 1 visualizes the learning representation of Continual Learning (in a task-incremental setting, which can be seen as a multi-task online learning paradigm), Neural Architecture Search, and their intersection (CNAS).

Paper Outline. To investigate hybrid Continual Learning and Neural Architecture Search models, we first conduct two distinct reviews on the latest developments in each area (Sections II & III). After which, the intersection of the two fields is formalized and discussed, outlining ideal characteristics to

achieve autonomous and adaptive agents. Lastly, Section V summarizes the limitations of the existing methodologies and delineates potential future research directions.

II. Continual Learning Approaches

Continual Learning aims to maximize the capacity for simultaneous solutions by generalizing previous experiences to coincide with new tasks (a feature known as *Forward Transfer* (FWT) of knowledge), whilst also consolidating and maintaining previous knowledge (referred to as *Backward Transfer* (BWT)). Balancing this trade-off between stability and plasticity is essential for a sustainable CL model, and is commonly known as the *Stability-Plasticity dilemma* [56], [57]. For instance, consider two incremental tasks, $\mathcal{T}_i$ and $\mathcal{T}_j$, where $i < j$.

Forward Transfer is the impact of $\mathcal{T}_i$ on the performance of $\mathcal{T}_j$; if a preceding task is advantageous to the learning process of a future task, it is considered positive FWT, which potentially leads to *"zero-shot"/"few-shot" learning* [58]. Negative FWT, on the other hand, occurs when a previous task hinders the learning process of a future task, essentially making a model less pliable to fit new tasks.

Backward Transfer is the effect the learning process of task $\mathcal{T}_j$ has on the performance of task $\mathcal{T}_i$. There exists positive BWT if learning $\mathcal{T}_j$ increases the performance of the preceding task $\mathcal{T}_i$. Conversely, negative BWT exists when learning $\mathcal{T}_j$ decreases the performance on $\mathcal{T}_i$ (i.e. Catastrophic Forgetting) [13], [58].

Since any model is constrained by a finite memory capacity and the number of tasks is potentially unbounded, forgetting can be inevitable. As the number of tasks increases, convergence becomes more compromised, leading to a decline in the model performance. Therefore, the goal behind CL models is not to prevent Catastrophic Forgetting, but to alleviate it and selectively minimize complete loss of knowledge, a feature referred to as *Graceful Forgetting* [9], [59]–[61].

Types of Continual Learning. Given the incremental nature of Continual Learning, there are three fundamental settings that may be encountered [17]. For a set of incremental tasks $\mathcal{T}$, each task t can have $d \geq 1$ incremental datasets $\mathcal{D}_d^t = \{\mathcal{X}_d, \mathcal{Y}_d\} \quad s.t. \quad t \in \mathcal{T}$, where $\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y})$ are the probability distributions of the training data and output labels, respectively.

(i) *Domain-Incremental Learning* (DIL): A single task with dynamic input distribution and a static label space. $|\mathcal{T}| = 1, \mathcal{P}(\mathcal{X}_i) \neq \mathcal{P}(\mathcal{X}_j), \mathcal{P}(\mathcal{Y}_i) = \mathcal{P}(\mathcal{Y}_j), \forall (i \neq j) \in \mathcal{D}$

(ii) *Class-Incremental Learning* (CIL): A single task with dynamic input distribution and an intersecting (but growing) label space. CIL is typically modelled with an expanding single-head output layer. $|\mathcal{T}| = 1, \mathcal{P}(\mathcal{X}_i) \neq \mathcal{P}(\mathcal{X}_j), \mathcal{Y}_i \cap \mathcal{Y}_j, \forall (i \neq j) \in \mathcal{D}$

(iii) *Task-Incremental Learning* (TIL): A set of distinct tasks with disjoint input and output spaces. TIL is typically modelled with a multi-head output layer to accommodate the dissimilar probability distributions of the tasks. $|\mathcal{T}| > 1, \mathcal{P}(\mathcal{X}_i) \neq \mathcal{P}(\mathcal{X}_j), \mathcal{P}(\mathcal{Y}_i) \neq \mathcal{P}(\mathcal{Y}_j), \forall (i \neq j) \in \mathcal{D}$

In addition to the incremental settings, there are several different approaches to retain memory in CL models. While there have already been a few proposed taxonomies [9], [60], [62], we propose a categorization scheme (Fig. 2) that adjusts the view of memory-retention models in the adaptive learning context.

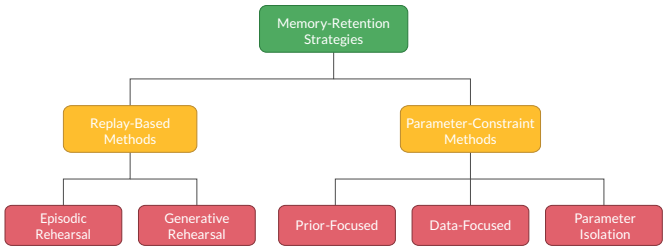

Fig. 2. Continual Learning memory-retention strategies.

Previous works regarded "static/dynamic architectures" as a family of memory models, whereas the adaptiveness of a network topology is tangent to overcoming catastrophic forgetting. For example, some replay-based approaches have fixed topologies [63], while others have dynamic topologies [64]. This showcases that structural plasticity is a catalyst for memory improvement that can be combined with any strategy in Fig. 2, rather than a holistic approach for retaining knowledge. Hence, we propose to classify structural plasticity as an independent quality that, albeit relevant to memory-retention, should not be categorized as a distinct family of CL models.

A. Replay-Based Methods

The first of the two families of methods is *Replay-Based* Continual Learning. Replay-Based models retain an episodic memory by storing representations that best describe the distribution of the data used for each task. These representations could simply be a raw portion of the datasets (*Episodic Rehearsal*), or it could be the model itself that was trained on a previous task used to infer and generate the samples used (essentially reverse-engineering the model) (*Generative Rehearsal*, often called *pseudo-rehearsal*). Each sub-type of the Replay-based models has its own strengths and weaknesses:

- Episodic Rehearsal: store a subset of the datasets used to later "rehearse" them in subsequent training of newer tasks. While this method might be the easiest to implement out of the 2 sub-types, the model can easily overfit to the subsets due to the limited number of samples, and does not scale well in terms of memory cost.
- Generative Rehearsal: store the previous tasks' distributions instead of the raw samples, which can later be used to project and generate auxiliary data for replay purposes. Although this method has not been used frequently in recent studies on CL, the recent emergence of generative networks opens a potential for new Generative Rehearsal models [9].

Both Replay-Based approaches tend to be computationally- and/or memory-expensive, especially as the number of tasks

increases [9], [13], [63], [65]. However, by caching portions of the datasets or the models used for each task, the retraining process can support prioritization for particular tasks, thereby facilitating more control over Graceful Forgetting.

Fundamentally, Replay-Based methods draw inspiration from the biological *Complementary Learning System* framework, where short-term memory is quickly stored to the hippocampus, to be later learned slowly and in a more generalized fashion by the neocortex (likely during sleep). For a more comprehensive review on the association between biological mechanisms and Replay-Based CL models, the recent work by Hayes et al. covers the topic extensively and in-depth [66].

B. Parameter-Constraint Methods

The latter family of approaches is *Parameter-Constraint* Continual Learning. By constraining parameters through isolation, regularization, or any form of transfer learning/fine-tuning, Parameter-Constraint models can efficiently prevent Catastrophic Forgetting whilst not wasting memory on prior data storage. Sub-types of Parameter-Constraint models are split three-fold:

- Data-Focused: constrain parameters through a combination of Fine-Tuning and *Knowledge Distillation* [67], where a temporary (large and highly regularized) model is trained on a single task and is then used to transfer the knowledge to the main (smaller) network.
- Prior-Focused: constrain parameters through regularization. By penalizing changes to the most influential parameters in the network, Catastrophic Forgetting can be minimized. Geometrically, the model attempts to find the area of best fit (where most solutions coincide) in the solution space.
- Parameter-Isolation: constrain parameters through isolation. Unlike Prior-Focused models, this method completely isolates some of the weights ($\lambda = 0$) instead of just applying penalties, hence dedicating portions of the network to individual tasks.

Combining the Prior-Focused and Parameter-Isolation approaches yields a commonly used technique called *Model Decomposition*, where the model is separated into "task-specific" and "task-shared" weights [68]. The task-specific components of the model are isolated subgraphs dedicated to individual tasks, whereas the task-shared components are shared amongst 2 or more tasks. A common structure for multi-head (task-incremental) models uses Model Decomposition by having a shared network stem that branches into $n = |\mathcal{T}|$ task-specific subgraphs.

A minor limitation with penalizing changes to the weights or completely isolating them is that the priority will typically be given to the earlier tasks, rendering Graceful Forgetting less attainable using traditional methods [69], [70]. Although the lack of dedicated external memory in Parameter-Constraint methods is desirable, especially in the broader computationally-intensive CNAS picture, the ability to manage and prioritize task forgetting is also essential to maintain model performance.

By combining Replay-Based and Parameter-Constraint methods, a third, implicit family of approaches can be defined. The first method to include regularization techniques on top of an episodic memory is Gradient Episodic Memory for Continual Learning (GEM) [58]. GEM aims to constrain the parameter changes pertaining to new tasks by placing an upper-bound limit on the losses of each task; the loss of every task can decrease but cannot increase.

C. Evaluation of Continual Learning Performance

Despite the large body of research on Continual Learning, the benchmark tests and evaluation metrics for Continual Learning models are largely heterogeneous and very few works have been devoted to formalize an evaluation framework that captures the models' performance in dynamic environments [13], [27], [69], [78]. The majority of the early works relied on a rather uninformative evaluation approach, measuring only the average accuracy across all tasks. While accuracy is generally considered a reliable measure of the overall model performance, Continual Learning should be ranked based on more robust criteria that are specific to dynamic scenarios.

Lopez-Paz & Ranzato proposed one of the first formal evaluation methods for Continual Learning models [58]. Their methodology revolves around more "human-like" test environments, where the models' evaluation should be based on a large number of tasks, each with a relatively small number of examples, and the reported metrics should include not only accuracy, but also a measure of forward/backward transfer. Given that some models can yield the same accuracies, FWT and BWT can provide a better insight on the model's long-term performance. Following the same direction, Díaz-Rodríguez et al. later proposed a similar evaluation framework, with the addition of 3 metrics: model-size efficiency, samples storage-size efficiency, and computational efficiency [79]. While these metrics are not directly related to the tasks' performance (i.e. accuracy, stability, plasticity, etc.), they are vital indicators to a model's scalability.

While the measure of FWT and BWT can provide extensive information on the behavior of a model, they do not directly estimate how well a model retains memory. Kemker et al. focused on assessing a model's ability to overcome Catastrophic Forgetting particularly [80]. The study establishes 3 (normalized AUC-ROC curve) metrics that jointly measure memory retention, the model's ability to acquire new knowledge, and the model's overall performance: Ω_{base}, Ω_{new}, and Ω_{all}. The base metric evaluates how well the first/base task is retained after subsequent tasks have been learned. Ω_{new} measures the performance of a new task immediately after it is learned, which shows the model's few-shot learning ability. Lastly, Ω_{all} evaluates the model's overall performance in terms of retention and acquisition of knowledge.

Inspired by the aforementioned approach, Hayes et al. proposed a similar evaluation framework that is split into 3 experimental paradigms [81]. The first benchmark test simply measures the model's ability to quickly learn new tasks through continual iid data. While real-world scenarios

TABLE I
BRIEF ANALYSIS OF CONTINUAL LEARNING MODELS FROM AN ADAPTABILITY PERSPECTIVE.

Model	Incremental Setting	Memory Strategy	Limitations
Learning Without Forgetting (LwF) [71]	Task-Incremental	Data-Focused	• Highly dependent on the relevance of the tasks • Training time gradually increases with the number of tasks • Architecture grows linearly with each task-increment
Incremental Classifier and Representation Learning (iCaRL) [63]	Class-Incremental	Episodic Rehearsal	• Requires explicit storage for older tasks • Can easily overfit on the tasks' exemplars subset • The exemplars chosen may not be representative of the entire dataset
Elastic Weight Consolidation (EWC) [72]	Task-Incremental	Prior-Focused	• Strictly requires a fixed network architecture (known in advance) • Numerically unstable for a large number of tasks
Gradient Episodic Memory for Continual Learning (GEM) [58]	Task-Incremental	Episodic Rehearsal & Prior-Focused	• Requires explicit storage for older tasks • Highly dependent on the relevance of the tasks • No memory-management policy
Memory-Aware Synapses (MAS) [59]	Task-Incremental	Prior-Focused	• Strictly requires a fixed network architecture (known in advance)
Progressive Neural Networks (PNN) [69]	Task-Incremental	Parameter-Isolation	• Does not facilitate positive BWT
Experience Replay for Continual Learning (CLEAR) [65]	Task-Incremental	Episodic Rehearsal	• Built on the assumption that input data stream is always iid
Continual Learning through Synaptic Intelligence (SI) [73]	Task-Incremental	Prior-Focused	• Inefficient when applied to a pretrained network • Tasks' order affects performance • The importance of the weights is not properly estimated in some cases (as noted by the authors)
Deep Generative Dual Memory Network for Continual Learning (DGR) [74]	Task-Incremental	Generative Rehearsal	• Strictly requires a fixed network architecture (known in advance) • Model saturates with tasks over time (performance declines as the number of tasks increases)
Incremental Moment Matching (IMM) [75]	Class-Incremental & Domain-Incremental	Prior-Focused	• Requires explicit storage for previous tasks' models
Less-Forgetting Learning (LFL) [76]	Domain-Incremental	Data-Focused	• Highly dependent on the relevance of the tasks • Susceptible to data-distribution (domain) shift [77]
Expert Gate [70]	Task-Incremental	Parameter-Isolation	• Does not facilitate positive BWT due to gating mechanism

tend to comprise of non-iid streams of data, this test is used to establish a baseline metric that can be compared with offline models. The second test evaluates the model's ability to incrementally learn new classes whilst measuring the tasks' test accuracies with each increment, which can also indicate when Catastrophic Forgetting occurs. Finally, the last test consists of a stream of organized non-iid data, resembling a realistic stream of stimuli that could be experienced by a deployed model. This framework also uses the Ω_{all} metric to measure stability, plasticity, and overall performance.

It is important to highlight that Continual Learning environments can vary to great extents. A robust evaluation framework should accommodate these variations and yield consistent metrics regardless of the application's environment. Farquhar & Gal introduced the first desiderata for CL ranking, highlighting the desirable features in an evaluation framework [62]. The proposed desiderata suggests that data from later tasks should resemble earlier tasks to some extent, arguing that some evaluation approaches use permuted datasets (permuted MNIST in particular [80], [81]), which is an unnecessarily excessive test. The study also claims that the output vector used in the training phase should be the same as the one used during testing. Another desirable feature in the evaluation of a CL model is to restrict retraining for older tasks; the model should not be allowed unconstrained access to early tasks' examples. Last but not least, the test should include more than 2 tasks, as success on such a small sample size cannot yield reliable performance metrics that can be applicable to larger scales.

The most recent formal study on Continual Learning evaluation metrics was proposed by New et al. in 2022 as part of the DARPA L2M research program [78]. The primary aim behind this evaluation framework is to robustly rank models in a domain-agnostic fashion, doing so based on 5 metrics: Performance Maintenance (across all tasks), FWT, BWT, Performance Relative to a Single Task Expert, and Sample Efficiency (ability to improve the learning process over time). Together, these metrics capture the effectiveness of memory-retention, knowledge transfer & adaptation, and model sustainability. Not only do such metrics assist researchers in identifying the suitable models for each application domain, they also empirically consolidate all the ideal qualities for CL models proposed in previous studies. Aiming for higher ranks on this domain-agnostic adaptiveness scale further paves the way towards autonomous agents.

The next section reviews the automation of neural architec-

ture optimization to further explore methods of adaptation in DNNs. We refer the reader to the study by Mirzadeh et al., which provides a comprehensive investigation into the impact of neural network architecture on continual learning performance [82]. Their findings reveal that overparameterization of architectures does not uniformly enhance continual learning capabilities, but rather that specific topologies can be more suitable depending on the balance required between learning new tasks (plasticity) and retaining existing knowledge (stability).

III. Neural Architecture Search Approaches

Neural Architecture Search [83], [84], a subfield of AutoML, was introduced to shift from the traditional handcrafted, expert-driven architecture design process to a more robust and automated methodology.

The NAS process comprises of defining 3 main components (as shown in Fig. 3): a *Search Space*, a *Search Algorithm*, and an *Evaluation Strategy* [45], [53], [85].

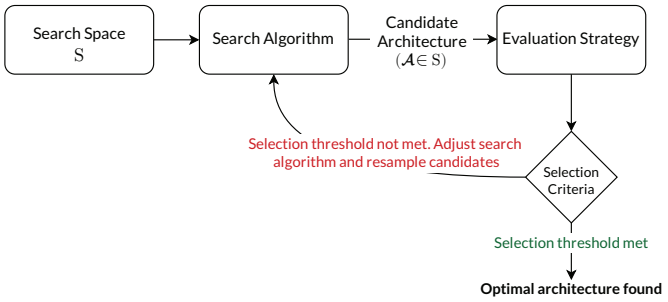

Fig. 3. The Neural Architecture Search framework iterative process.

A. Search Spaces

The first step in NAS is defining the search space, which significantly impacts the search process. An overly simple search space leads to shallow networks, while a high-dimensional space would yield complex solutions but can be resource-intensive to explore. Common search spaces include:

Layer-Wise Space: Often referred to as the *Macro-Architecture* space, Layer-Wise Spaces sample a sequential network from a given pool of layers [45], [114]–[116]. Early works by Zoph & Le and Baker et al. [53], [84], [96] demonstrate this. Some studies introduce Skip-Connections, allowing multi-branch networks like ResNet and DenseNet [4], [117], but these are resource-intensive.

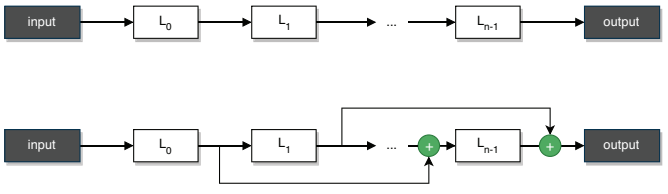

Fig. 4. Layer-Wise (Macro-Architecture) Search Space for NAS; Single-branch architecture (top) and multi-branch (bottom).

Cell-Based Space: The Cell-Based Space (Micro-Architecture) uses repeated sub-structures, reducing resource costs [45], [53], [91], [94], [97], [98], [105], [106], [118]. This modular approach, seen in ResNet and InceptionNet [2], [4], breaks optimization into smaller problems but requires predefined macro-structures.

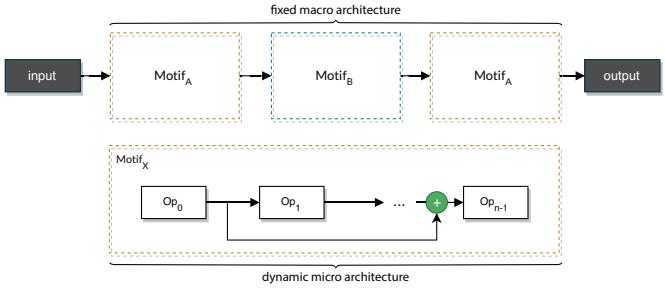

Fig. 5. Generic NAS Cell-based (Micro-Architecture) Search Space. Static macro-structure (top) and optimized motif (bottom).

Hierarchical Space: Hierarchical Space, introduced by Liu et al. [104], constructs architectures using recursive sub-graphs. It uses primitive operations to set up the first level, ℓ_1, and each subsequent level is composed of sub-graphs from the previous level. The composition yields a new operation as a Directed Acyclic Graph (DAG) encoded as adjacency matrices:

$$o_i^{(\ell)} = compose(\mathcal{O}^{(\ell-1)}) \ , \ \forall \ell = 2, ..., L \tag{1}$$

Other Search Spaces: Early neural optimization used Neuroevolution, focusing on simple Perceptrons (Neuronal Search Space) [101], [102], [119]. Recent alternatives include Neural Search-space Evolution (NSE) by Ci et al. [120], and Continuous Search Space by Liu et al. [90], which transforms the search space into a continuous, differentiable space for gradient optimization:

$$\bar{o}_{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{e^{\alpha_o(i,j)}}{\sum_{o' \in \mathcal{O}} e^{\alpha_o(i,j)}} o(x^{(i)}) \tag{2}$$

B. Search Algorithms

After defining a Search Space, the next step in a NAS pipeline is to create and optimize a structure using a Search Algorithm. These algorithms sample candidate networks from a solution space to be evaluated and optimized.

Exhaustive Search: Random Search (RS) selects networks randomly, offering great exploration but lacking optimization feedback [88]. Grid Search (GS) tries combinations of predefined operations, allowing parallelization but often being outperformed by RS [121].

Reinforcement Learning: Reinforcement Learning (RL) uses an Agent, Policy, Environment, and Reward Function. Zoph & Le's model [84] employed an RNN controller with the REINFORCE algorithm [122] to sample CNNs, but required vast computational resources, using 800 GPUs for 4 weeks [84], [97].

TABLE II
PERFORMANCE COMPARISON FOR NAS MODELS BASED ON THEIR SEARCH SPACE, SEARCH ALGORITHM, AND EVALUATION STRATEGY.

Search Algorithm	Reference / Model	Year	Search Space	Evaluation Strategy	GPU Usage (days)[1]	Params (millions)	CIFAR10 Error (%)
RS	Luo et al. (NAONet Random-WS) [86]	2018	Hierarchical	OS	0.25	3.90	3.92
	Brock et al. (SMASHv2) [87]	2018	Memory Bank Representation [87]	OS	16	4.60	4.03
	Li & Talwalkar (RandomNAS) [88]	2020	Cell-Based	OS	2.7	4.30	2.85
	Zhang et al. (Random-NSAS) [89]	2020	Cell-Based	OS	0.7	3.08	2.64
GO	Liu et al. (DARTS (second order)) [90]	2018	Continuous Cell-Based	OS	4	3.30	2.76
	Cai et al. (ProxylessNAS-G) [91]	2018	Continuous Cell-Based	OS	8.33	5.70	2.06
	Dong & Yang (GDAS) [92]	2019	Continuous Cell-Based	OS	0.17	2.50	2.82
	Zhou et al. (BayesNAS) [93]	2019	Continuous Hierarchical	OS	0.1	3.40	2.41
SMBO	Liu et al. (PNAS) [94]	2018	Cell-Based	LFE	-	3.20	3.41
	Kandasmy et al. (NASBOT) [95]	2018	Layer-Wise	LFE	1.6	-	12.09
RL	Baker et al. (MetaQNN) [96]	2016	Layer-Wise	Full Training	90 ± 10	11.20	6.92
	Zoph & Le (NAS) [84]	2017	Layer-Wise	Full Training	22,400	37.40	3.65
	Zoph et al. (NASNet-A) [97]	2018	Cell-Based	Full Training	2,000	3.30	3.41
	Cai et al. (EAS) [98]	2018	Hierarchical	Weight Inheritance	17.5 ± 7.5	23.40	4.23
	Pham et al. (Efficient NAS) [99]	2018	Cell-Based	OS	0.45	4.60	2.89
	Ding et al. (BNAS) [100]	2021	Cell-Based	OS	0.19	4.80	2.88
NE/ENAS	Stanley & Miikkulainen (NEAT) [101]	2002	Neuronal	-	-	-	-
	Stanley et al. (HyperNEAT) [102]	2009	Neuronal	-	-	-	-
	Real et al. (Large-Scale Evolution) [103]	2017	Layer-Wise	Weight Inheritance	2,600 [45]	5.40	5.40
	Liu et al. (Hierarchical EAS-Evolution) [104]	2018	Hierarchical	Full Training	300	15.70	3.75
	Real et al. (AmoebaNET-A) [105]	2019	Cell-Based	Full Training	3,150	3.20	3.34
	Elsken et al. (LEMONADE) [106]	2019	Cell-Based	Weight Inheritance	40 ± 16	3.40	3.60
	Lu et al. (NSGA-Net) [107]	2019	Cell-Based	LFE	4	3.30	2.75
	Guo et al. (SPOS) [108]	2020	Cell-Based	OS	-	-	-
	Yang et al. (CARS) [109]	2020	Cell-Based	OS	0.4	3.60	2.62
MO	Niu et al. (PNAS) [110]	2019	Cell-Based	Weight Inheritance	0.2	2.44	4.62
	Byla & Pang (DeepSwarm) [111]	2019	Layer-Wise	LFE	-	-	11.31
	Mo et al. (SA-NAS-c) [112]	2021	Cell-Based	LFE	0.1	3.20	2.53
	Shahawy & Benkhelifa (HiveNAS) [113]	2022	Layer-Wise	LFE	0.3	1.39	8.90

Neuroevolution/Evolvable-NAS: Neuroevolution (NE) and Evolvable NAS (ENAS) use Evolutionary Algorithms to optimize network structures. NE optimizes both structure and weights, while ENAS focuses on architecture [123]. These methods simulate "survival of the fittest" and show promising results [102]–[106], [124]–[126].

Gradient Optimization: Gradient Optimization (GO) became viable for NAS with the introduction of continuous search spaces [90], [127]. DARTS uses a continuous Cell-Based Space, optimizing both architecture and weights with gradient descent [90]. GDAS reduces search complexity by

[1]Although regularly used as a performance metric in the literature, GPU models vary across the experiments conducted. These numbers serve only as an approximate estimation of computational-demands.

sequentially relaxing search space edges [92].

Other Search Algorithms: Other powerful but less common algorithms include Surrogate Model-Based Optimization (SMBO) [7], [93], [94], [128]–[130], Swarm Intelligence (e.g., Particle Swarm Optimization [110], Ant Colony Optimization [111], [131], [132], and Artificial Bee Colony Optimization [113]), and Simulated Annealing [112].

C. Evaluation Strategies

To guide the Search Algorithm and optimize sampled solutions, an Evaluation Strategy estimates the samples' performance. While it does not heavily impact the search process, it significantly affects computational demands as evaluations are performed at every iteration [133].

Full Training: Full Training (or "Training from Scratch") trains each candidate network until convergence to estimate accuracy and other metrics. This method provides the most accurate feedback but is highly computationally demanding, requiring large numbers of GPUs and extensive training time [84], [97], [105], [134], [135].

Lower Fidelity Estimation: To reduce evaluation time, candidates can be trained on a subset of data [136], [137], for fewer epochs [97], using lower resolution images [138], or using compressed models [97], [105], [139]. This *Lower Fidelity Estimation* (LFE) speeds up evaluations but risks introducing bias. Although relative performance rankings can guide search algorithms, large biases may require a multi-fidelity approach [140]–[143].

Weight Inheritance: Weight Inheritance (WI) from a parent model reduces evaluation time by initializing candidate networks with relevant functions or fine-tuning weights instead of training from scratch. This method, similar to Net2Net's knowledge transfer [144], efficiently evaluates candidates and is commonly used in ENAS models [91], [98], [106], [123], [145].

Performance Predictors: Performance Predictors estimate performance using surrogate models, such as *Learning Curve Extrapolation* (LCE), which partially trains networks and extrapolates their learning curves [146]. Though LCE can be seen as a subset of LFE, it typically stands alone as a distinct methodology [53], [147]. While requiring extensive training for the surrogate model, fixed-hyperparameter frameworks can yield satisfactory performance [147]. For a comprehensive review, see [148].

One-Shot Models: One-Shot (OS) models rely on weight-sharing within a single over-parameterized network (supernet), where sub-graphs represent candidate models [55]. Training one super-network is only slightly more complex than training individual candidates [149]. SMASH by Brock et al. [87] uses a surrogate "HyperNet" to generate pseudo-random weights, allowing fast architecture sampling. However, OS models are constrained by GPU capacity and static supernet structures.

D. Benchmarks

NAS models are predominantly benchmarked on the CIFAR-10 dataset, favoring convolutional networks [150], [151]. Reproducible benchmarks like NAS-Bench [152]–[154] provide pre-evaluated performance metrics, reducing computational demands. NAS has expanded to other domains, including NLP [155], semantic segmentation [156], and speech processing [157].

Notwithstanding the achievements accomplished in the NAS domain, frameworks generally assume a non-incremental experimental setup where all data is present prior to architecture generation, which limits their use to static environments. In the next section, we explore how NAS can be used in lifelong learning settings and formalize autonomous adaptiveness for DNNs.

IV. Continual Neural Architecture Search

Humans have evolved to be capable of general learning and skill-refinement; the essence of our intelligence is captured by our ability to continuously assimilate new knowledge and adapt to the varying stimuli we are exposed to (i.e. Classical Conditioning) [158], [159]. Most modern DNNs are models of what is referred to as *dedicated intelligence* [160], or domain-specific intelligence, which is a product of the particularly restrictive (albeit efficient) single-task methodologies commonly used today [12].

Human-like intelligence has long been the ultimate goal of research in AI [161]. Although the definition of "human intelligence" is largely tacit, one of its most defining features is *adaptive self-modification* [162].

A. Definitions

In the context of Deep Neural Networks, the existing definitions of the term "adaptive" are inconsistent. Most early studies described weight-modification as a type of adaptiveness (implying that all neural networks are fundamentally adaptive) [163], [164], while more recent works generally confined the definition to specific areas (i.e. domain-adaptive [165], mode-adaptive [166], etc.).

Derived from Hebb's principle on Synaptic Plasticity, the term Evolved "Plastic" Artificial Neural Networks (EPANNs) has been used to describe adaptiveness on a cellular/synaptic level [12]. These self-managed plastic neural networks, however, have been exclusively researched in Unsupervised Learning settings using Neuroevolution [12], [48]. To break out of the restricted scope of Neuroevolution and address continual autonomy in Supervised Learning environments, we propose and formalize the intersection of NAS and CL as its own paradigm. Ideally, merging the two domains will results in neural networks that are:

1) **fully-autonomous** throughout their life-cycle,
2) **inherently continual** (i.e. have CL attributes, regardless of their Incremental Learning setting), and
3) are capable of handling and adapting to an **infinite stream of incremental input** (with allowed degradation of performance over time; see Section IV-C-4).

Although there are numerous additional components that could be considered adaptive (e.g. *learning rule plasticity*),

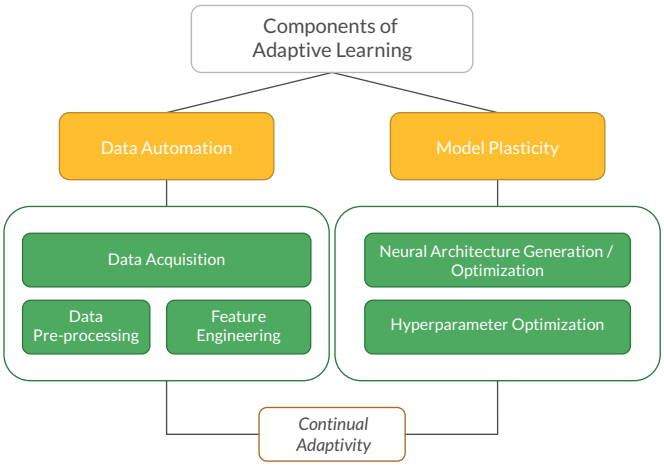

Fig. 6. Proposed taxonomy for the components of lifelong autonomous agents in a standard supervised learning environment.

Fig. 6 illustrates the baseline elements that would ensure end-to-end adaptiveness in a typical supervised learning DNN life-cycle.

Moreover, it is important to highlight that – according to Criterion 2 – CNAS should not simply be an iterative AutoML pipeline. While there exists several approaches that can automate pre-training data preparation and model generation, it is desirable for every component to have Continual Learning attributes; the learning process should not be iterated with statistical independence for every task encountered, but rather enable Forward and Backward Transfer wherever possible. This not only offloads some of the optimization computational requirements, but also generalizes the learned representations across the CNAS pipeline.

B. Continually-Adaptive Components

The first stage in a standard Machine Learning pipeline is Data Preparation, which includes all procedures required before the training stage. These procedures include *Data Acquisition*, *Data Pre-processing*, and *Feature Engineering* (as shown in Fig. 7) [51], [52], [167].

1) Data Acquisition: In particular real world applications, a continuous source of data (e.g. a sensor, live camera, etc.) is attached to the model, rendering the Data Acquisition step unnecessary. However, in other scenarios, the data-source is either insufficient (provides a limited amount of data) or simply does not exist. Considering the importance of the dataset's size and its impact on a model's performance, a number of approaches have emerged to extend limited data sources. The most common techniques to expand small data in an automated setting are *Web Search / Scrape* and *Data Synthesis*.

With the increasing availability of public datasets, web-scraping [168]–[170] and dedicated APIs[2][3] [171] have become viable solutions for automated Data Acquisition. The main concerns with such Data Search approaches are:

[2]https://www.datasearch.elsevier.com
[3]https://datasetsearch.research.google.com/

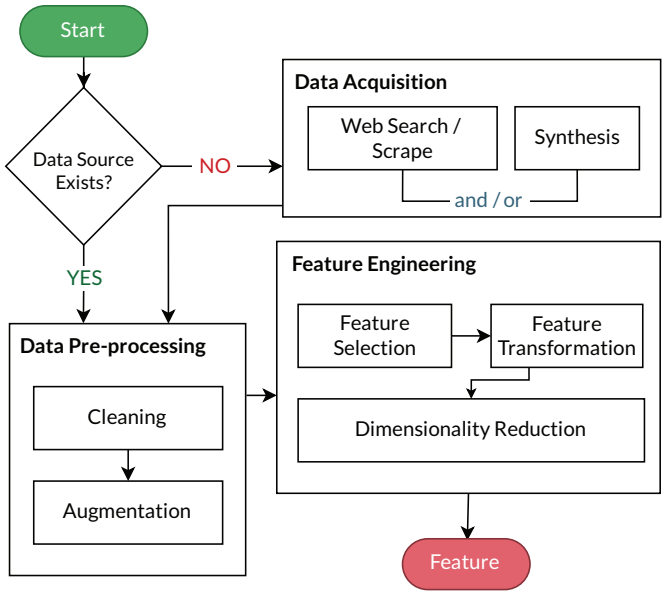

Fig. 7. Continual Data Automation flowchart. In a CNAS framework, the pre-processed features would then be used by the NAS Evaluation Strategy to rank the generated architectures.

- The legality and ethics of web-scraping [172].
- Search results are likely to contain irrelevant data.
- Unpredictable data quality; without a supplementary data-quality verification technique, the acquired data is unreliable.

Data Synthesis, in contrast, offers a more controllable Data Acquisition environment that overcomes all the listed drawbacks posed by Web Search/Scrape. The fast-growing research on generative models and data simulators has shown that synthetic data can be an impressive alternative when real data is unattainable. The mesmerizing realism produced by *Generative Adversarial Networks* (GANs) [173] also proves that synthetic data can be almost indistinguishable from their real counterpart whilst providing granular control of the results [174]. Moreover, tools like OpenAI Gym, Unreal Engine, and even the Grand Theft Auto game have been successfully used to simulate entire scenes for image classification and segmentation [175], [176]. The apparent limitation of Data Synthesis is that the process (in an automated context) requires an initial dataset to establish a baseline distribution for the synthesizer to match.

2) Data Pre-processing: Whether the data is directly fed to the framework through a continuous source or externally acquired, Data Cleaning is inevitable. The cleaning process includes removing noise/outliers, handling missing data (commonly denoted as *NaN*s), and eliminating inconsistencies [177]. Although the Data Pre-processing stage is heavily dependent on the domain of the problem (image, text, audio signals, etc.), some data cleaning algorithms, such as BoostClean [178] and AlphaClean [179], are domain-agnostic and therefore enable generic automation. Moreover, Continuous Data Cleaning frameworks that leverage the user's feedback on cleaning preferences have been briefly explored [180].

Amongst the issues that commonly arise during Data Prepa-

ration are class imbalance (uneven data per label), and lack of variability/over-fitting. By augmenting the data (i.e. using affine transformations on image datasets or pitch-scaling on audio signals; we refer the reader to the survey by He et al. [52] for a Data Augmentation taxonomy), new data points can be generated to help balance the dataset and increase the generalization ability of the model. A byproduct of Data Augmentation is dataset extension and so it can also be regarded as a Data Acquisition approach. However, the consensus is that augmentation is primarily used for its regularization qualities and is thus considered a part of the Data Pre-Processing stage [52], [181].

3) Feature Engineering: More often than not, a dataset contains redundant features that negatively impact the convergence of the model during training. The function of any neural layer is to essentially map an input feature vector $\vec{x} = [x_1, x_2, ..., x_n] \in \mathbb{R}^n$ to an output vector $\vec{y} = [y_1, y_2, ..., y_m] \in \mathbb{R}^m$ with a nonlinearity, which can be modelled as:

$$\hat{y} = \sum_{i=1}^{n} x_i W_i + b_i \ , \tag{3}$$

$$\vec{y} = f(\hat{y}) \tag{4}$$

Where $W = [w_1^\top, w_2^\top, ..., w_n^\top] \in \mathbb{R}^{n \times m}$ is the parameter matrix of the network, $\vec{b} = [b_1, b_2, ..., b_m] \in \mathbb{R}^m$ is the bias vector, and $f(\cdot)$ is an element-wise nonlinearity (e.g. Tanh, Sigmoid, etc.).

A training matrix $X = [\vec{x}, \vec{y}]$ with correlated features (i.e. collinearity or multi-collinearity) is linearly dependent and thus X is not a full rank matrix. Given the high-dimensional nature of deep neural networks, where m and n are often in the order of thousands [117], rank-deficiency significantly impacts the training time of the model [195]. Feature Selection not only speeds up the convergence of the optimization process, it also enables the use of techniques like Low-Rank Approximation that can significantly reduce the memory-footprint of the model by representing the weight tensor as multiple smaller tensors through low-rank factorization [196].

Due its non-trivial and creativity-driven nature, Feature Transformation and Construction is usually considered the most human-dependent Feature Engineering phase [51], [52]. Constructing new features extends the dataset and its generalization by synthesizing new features from existing ones (as opposed to Data Augmentation, which adds new samples/rows rather than columns), whereas Feature Transformation affects the speed of the optimization process. Linear feature transformations, such as normalization and standardization, boost the convergence of the training process by scaling the raw data distribution and ensuring the features have similar magnitudes. Additionally, the traditional use of a single learning rate in DNNs upholds the importance of unifying the features' scale; in a Gradient Descent training scenario, a global learning rate applied to unnormalized data will most likely propel each gradient/dimension at proportionally different magnitudes.

Similar to Feature Selection, Dimensionality Reduction (often called Feature Extraction) aims to separate redundant data from effective information. The primary distinction between the two processes is that Dimensionality Reduction creates new compressed mappings of the original features rather filter them as with Feature Selection. Although neural networks perform Dimensionality Reduction inherently (e.g. pooling or convolution layers extract features from localized regions), methods such as Principal Component Analysis (PCA) can scale a neural network's input and reduce the number of operations performed, thereby minimizing the computational requirements for training [197].

Feature Engineering can be regarded as a stochastic optimization problem and therefore a number of traditional optimization algorithms can be used to automate the process. The limited Automated Feature Engineering frameworks that have been proposed are mainly based on Genetic Algorithms [198], [199] and Reinforcement Learning [200], where both approaches performed impressively well compared to baselines (random- and manual/human-based). With the current lack of a unified benchmark, however, automated Data Preparation frameworks cannot be formally ranked or compared.

4) Neural Architecture Generation and Optimization: The crucial phase of generating a network's topology for a given task has been discussed in depth in Section 3. Nevertheless, the continuous maintenance of a neural network should not be interpreted as a repeated NAS process; while reiterating a NAS algorithm for each encountered task can indeed result in a fully-autonomous continuous pipeline, the computational cost and time required to repeatedly search for architectures from scratch will likely be unfeasible for most applications. To address the expected high resource demand for continual NAS frameworks, a few models have been recently proposed that incorporate CL techniques within the architecture search process (e.g. finding an optimal NAS cell that best fits all encountered tasks and facilitates few-shot learning for future tasks [184]).

Proposed by Gao et al. in 2022, Continual Learning with Efficient Architecture Search (CLEAS) dynamically expands the network architecture whilst preserving previous knowledge through parameter isolation [192]. While this approach experiences no forgetting, it scales poorly; for every encountered task, the network has to expand with no means of compressing superfluous parameters. Similarly, ENAS-S [187] and Learn-to-Grow [182] also experience monotonic expansion without a memory boundary, which makes them unsustainable for long-term real world deployment.

Jiang et al. (2023) proposed Neural Weight Search (NWS), which addresses the issue of scalability in task-incremental learning by optimizing the search process for neural weights rather than architectures [193]. This method employs reinforcement learning to discover weight configurations that can be adapted efficiently as new tasks are introduced. NWS reduces computational overhead by reusing and fine-tuning existing weights, therefore preventing the need for expanding the network with each new task. NWS particularly demonstrates

TABLE III
COMPARISON OF EXISTING CONTINUAL NEURAL ARCHITECTURE SEARCH (CNAS) FRAMEWORKS.

Reference / Model	NAS Optimizer	Search Space	Eval. Strategy	Continual Learning Type		Benchmark	Error (avg. %)
Li et al. 2019 (Learn-to-Grow) [182]	GO	Continuous	OS	Task-Incremental	Prior-Focused & Parameter Isolation	ImageNet + CIFAR100 + + 8 others (avg)	30.16, 20.41, 22.32
Huang et al. 2019 (CNAS) [183]	RL	Layer-Wise	WI	Class-Incremental	Episodic Rehearsal	CIFAR100 (k=10)	36.50
Pasunuru & Bansal 2019 (CAS) [184]	RL	Cell-Based	LFE	Task-Incremental	Prior-Focused	MSR-VTT + MSVD	55.50, 38.30
Zhang et al. 2020 (E^2NAS) [185]	GO	Continuous	OS	Task-Incremental	Prior-Focused	CIFAR10 + CIFAR100 + ImageNet16	6.11, 27.95, 34.33
Zhang et al. 2020 (REC) [186]	RL	Layer-Wise	WI	Task-Incremental	Episodic Rehearsal & Prior-Focused	Permuted MNIST + MNIST Variation + CIFAR100	4.30, 28.50, 40.30
Wu et al. 2020 (Firefly) [119]	MO	Neuronal, Cell-Based	WI	Class-Incremental	Parameter Isolation	CIFAR100 (k=20)	8.97
Du et al. 2021 (ENAS-S) [187]	EA	Cell-Based	LFE	Class-Incremental	Parameter Isolation	CIFAR10, CIFAR100	10.00, 26.50
Niu et al. 2021 (AdaXpert) [188]	RL	Cell-Based	WI	Class-Incremental	Data-Focused	ImageNet100, ImageNet1000	19.26, 18.87
Zhang et al. 2021 (NSAS) [189]	GO	Continuous	OS	Task-Incremental	Parameter-Isolation	CIFAR10 + CIFAR100 + ImageNet16	6.45, 29.31, 57.86
Qin et al. 2021 (BNS) [190]	RL	Layer-Wise	OS	Task-Incremental	Episodic Rehearsal	MNIST, CIFAR10, CIFAR100	0.13, 8.60, 17.61
Mundt et al. 2021 (DP-NAS) [191]	RL	Layer-Wise	LFE	Task-Incremental	Generative Rehearsal	MNIST + FashionMNIST, MNIST + CIFAR10	0.21, 0.63, 23.69, 34.85
Gao et al. 2022 (CLEAS) [192]	RL	Neuronal	WI	Domain-Incremental, Class-Incremental	Parameter Isolation	Permuted MNIST, Rotated MNIST, CIFAR100	3.70, 3.00, 33.10
Jiang et al. 2023 (NWS) [193]	RL	Continuous	OS	Task-Incremental	Parameter Isolation	CIFAR10, CIFAR100	91.03, 79.88
Lu et al. 2024 (ArchCraft) [194]	EA	Continuous	OS	Class-Incremental	Prior-Focused	CIFAR100 (k=5)	76.18

significant improvements in task performance while maintaining a manageable model size, making it a practical solution for real-world applications where computational resources are limited.

Moreover, CNAS [183], REC [186], and BNS [190] all require a dedicated memory to store subsets of training data for replay purposes. While this memory-retention strategy offers great control over the retraining of the models as well as the NAS process, they are not capable of handling an infinite stream of input due to the unrestricted external memory requirement.

ArchCraft, introduced by Lu et al. in 2024, is a framework that integrates Evolutionary Algorithms with CL [194]. This approach focuses on optimizing network architectures continuously while preserving previously acquired knowledge. ArchCraft employs a dynamic evaluation strategy to ensure that the evolved architectures remain efficient and effective as new tasks are encountered. The framework demonstrated state-of-the-art performance on class-incremental (segmented; k=5) CIFAR100, achieving robust Average Incremental Accuracy and Last Accuracy.

An alternative approach proposed by Nïu et al. (dubbed AdaXpert) dynamically adjusts the neural architecture using a secondary model called the Neural Architecture Adjuster (NAA), which aims to search for a new topology only when deemed necessary (i.e. the distance between task boundaries is greater than a given threshold) [188]. Tested incrementally on ImageNet1000, one of the most challenging vision benchmarks, this approach achieved state-of-the-art performance in just 2.5 to 7 GPU days, proving that combining NAS and CL optimization techniques is computationally feasible. However, the NAA module identifies task distribution shift through Wasserstein Distance (commonly referred to as *Earth Mover's Distance*), which does not robustly map task boundaries to neural architectures. In other words, consider 2 datasets $\mathcal{D}_1, \mathcal{D}_2$ with distributions $\mathcal{P}_1, \mathcal{P}_2$, respectively, and an architecture $\mathcal{A}$ that yields a low error on $\mathcal{D}_1$ ($Error_{\mathcal{A}}(D_1) < \epsilon$, where ϵ is the low error threshold). Although $WD(\mathcal{P}_1, \mathcal{P}_2)$ could show that the two datasets are in close proximity, it cannot be guaranteed that $Error_{\mathcal{A}}(D_2) < \epsilon$ due to other

characteristics that are not captured by WD, such as the spatial dimensions of the data (number of channels, input resolution, feature complexity, etc.). Nevertheless, AdaXpert has made strides towards efficient CNAS and knowledge generalization through its novel NAA concept. However, in the experiment proposed for AdaXpert, the NAA uses Wasserstein Distance for every input to compute how far the new task boundary is from the existing trained model's probability distribution, which shows an implicit assumption that the input samples must be grouped prior to training. Given a real world scenario where samples are interleaved, the NAA will be prompted to adjust the network for every increment, rendering its autonomy futile.

An discernible limitation with existing Continual Neural Architecture Search (CNAS) models is their reliance on a priori knowledge, particularly evident in the framework's NAS Search Space design. Typically, the effectiveness of NAS frameworks is heavily influenced by how the search space is constrained [201], with optimal solutions often found within a subset of the search space that has been manually defined based on human experts' intuition and knowledge. This reliance limits the adaptability and generalization of CNAS models, as they may not perform as efficiently when encountering tasks beyond the scope of the predefined search space. Furthermore, this approach undermines the potential for CNAS models to autonomously adapt in response to changes in underlying distributions, restricting their usage in dynamic real-world applications where the data distribution and task requirements can change over time (expanded upon in Section IV-C-1).

Another drawback in the existing CNAS frameworks is their lack of Forward Transfer of Knowledge (as described in Section II. While they excel in autonomously optimizing architectures whilst retaining knowledge, they do not generalize previously learned knowledge to new task, which restricts their ability to harness the full potential of Continual Learning. Incorporating mechanisms for FWT in CNAS frameworks is crucial to ensure that the valuable insights gained from previous tasks can be leveraged to improve performance on subsequent ones, ultimately enhancing the efficiency and scalability of CNAS.

5) Hyperparameter Optimization: Most NAS frameworks use the same set of hyperparameters over the search process [52]. While static hyperparameters might be viable for single-task NAS problems, they cannot be efficiently generalized for unknown future tasks, therefore continuous Hyperparameter Optimization is needed to maintain efficiency across sequentially encountered tasks.

Furthermore, most NAS frameworks consider the architecture and hyperparameter optimization processes as two separate phases [52], [186], which exponentially increases the complexity of the framework. To overcome nesting architecture search and hyperparameter tuning, jointly optimizing both the structure and its hyperparameters can significantly reduce the framework's requirements, especially on a continual basis. A number of studies on evolutionary DNNs suggest that hyperparameters can be endogenous features (i.e. encoded as genes) and therefore get bootstrapped into the optimization process [12].

C. Desiderata for CNAS

In addition to the expected CNAS functional requirements introduced in Section IV-A, there are several desirable qualities that could assist in progressing CNAS towards a more general form of AI. The desiderata curated below stem from a holistic adaptiveness perspective, as well as an individualistic CL/NAS standpoints.

1) *Domain-agnostic*: The model should not rely on a priori task-boundary definition; be capable of adapting to its surrounding environment without exhibiting a preference to a certain domain.
2) *Abstract representation*: Encoding the network should be represented with a high level of abstraction (i.e. in the case of Evolutionary Computation, the genotype-phenotype relation should be indirect). This ensures model-scalability and follows on the biological analogy where the human genome is indirectly mapped to physical body [202].
3) *Dynamic architecture with bounded capacity*: The network topology should grow, shrink, or get remapped to accommodate different input complexities and maintain efficiency. However, the architecture-search process should be bounded by a given memory capacity to avoid overflow and facilitate deployment in real-world applications.
4) *Graceful Forgetting*: With a bounded memory capacity and a potentially unbounded stream of data, forgetting is inevitable. It should, nevertheless, be imposed gracefully through selective means, where the less important tasks' performance gradually decays rather than catastrophically fails.
5) *Few-Shot Learning capability (positive FWT)*: Positive FWT is not often the focus for most Continual Learning models; however, for CNAS, facilitating Few-Shot Learning is a priority in order to reduce the large computational costs over time.
6) *Multi-Modal input support*: To further pave the way for a more generalized form of AI, a CNAS framework should ideally support knowledge-acquisition through different modalities (e.g. image classification, text classification).
7) *Architecture Search Space inference*: The effectiveness of NAS frameworks is almost exclusively influenced by the Search Space design [201], which is typically a human-dependent process. Dynamic Search Space design for NAS is a favorable quality for CNAS to facilitate domain-agnostic adaptiveness.
8) *No explicit data storage for previous tasks*: Storing subsets of examples from previous tasks usually entails linearly increasing memory-consumption. Parameter-Constraint approaches, where the model's "memory" is implicitly stored through the training process, are therefore considered a more favorable memory-retention tech-

nique. Additionally, the globally growing privacy concerns also dictate less data-storage-based methodologies.

9) *Learning rule plasticity*: The training process for DNNs greatly relies on predefined learning rules (or weight-updating equations) that determine the changes to the network parameters. Jointly optimizing the learning rule with the other network parameters (as suggested by [12], [202]) could effectively cut down the training cost.

Although these features do not directly contribute to end-to-end automation and continuity, they aim to generalize CNAS and facilitate their scalability for real-world deployment.

V. Future Directions

A. Continual Learning Gaps & Directions

Notwithstanding the prospect of autonomous agents, there are drastic risks associated with such forms of continual automation. Unmonitored autonomy introduces the possibility of accidental dataset poisoning, where unwanted samples are fed to the model. In certain sensitive domains (UAVs, medical applications, industrial IoT devices, etc.), data-integrity must be verified and potentially have their input restricted to predetermined safe distribution boundaries. Safeguarding CL models against task deviations would further pave the way for safe AI.

One pervasive challenge for CL models is *Knowledge Heterogeneity*. To enable domain-agnostic Continual Learning, models must account for encountered highly diverse tasks and data sources. This potential task diversity can lead to disparities in the underlying data distributions, posing a considerable obstacle to generalizing and consolidating shared knowledge representations. This vulnerability is particularly present in Federated Continual Learning as task diversity becomes more prevalent [203]. Effectively managing (and possibly leveraging) heterogeneous knowledge is an open challenge in CL and directly implicates the broader CNAS goals, and is thus a pertinent research direction.

B. Neural Architecture Search Gaps & Directions

Furthermore, inconsistent with their state-of-the-art performance, Metaheurstic Optimization for NAS is often overlooked in comprehensive surveys and reviews [45], [53]. Scarce coverage of such approaches limits the exploration of the optimization family, which has continued to show great potential in multiple areas and inherently fits the parallel exploration nature of NAS [110], [112], [131]. Some MO approaches, such as Microbial Intelligence, Fish School Search, Glowworm Algorithm, have never been explored in the context of Neural Architecture Search, making them prospective future research directions.

Another gap in the NAS domain is the reliance on predefined search spaces. Yu et al. have concluded that the effectiveness of NAS frameworks is primarily attributed to how the Search Space is constrained [201]; NAS models can achieve state-of-the-art performance with minimal cost if they are explicitly bound to a subset of the Search Space where optimal solutions exist [148]. In the CNAS domain, however, a priori knowledge should not be exploited (Section IV-C-1), and thus domain-agnostic Search Space-inference methods are necessary for real world CNAS applications.

C. Continual Neural Architecture Search Gaps & Directions

A palpable concern regarding CNAS is the potentially staggering computational requirements; pipelining multiple optimization algorithms to produce and evaluate a possibly expanding architecture can be computationally expensive. Nevertheless, several models (Table III) have showcased the feasibility of CNAS in all Incremental Learning settings. Moreover, as per criterion 2 (IV-A-2), Continual Learning should prospectively be incorporated within the NAS process to enable knowledge transfer across learning increments, further lowering the optimization costs. This fundamental inclusion of CL into the learning process has never been explored.

While both NAS and Continual Learning models have implicitly agreed upon benchmarks in the literature, the intersection between both fields does not (as per the disparity in Table III). Some works use the average accuracy over all tasks to rank their model [182], [188]. However, that metric diminishes as the number of tasks approaches infinity. Consider a model with a bounded memory capacity for $\mathbb{C}$ tasks and an unbounded incremental input $x\ ,\ s.t.\ |x| = +\infty$, the running average of accuracies will reach an inflection point at $x_{\mathbb{C}}$ where the inevitable forgetting will set in and the average value is bound to diminish. Furthermore, in the context of a system that enables an infinite stream of input, the model should not be ranked according to the "final" model metrics; by definition, the framework has no final state and intermediate results might not reflect the framework's true performance. Additionally, the disjoint incremental learning settings and benchmarks cannot accurately estimate the models' adaptiveness and performance. Hence, a robust ranking approach for CNAS (potentially based on the rate of change of metrics rather than instantaneous values) should be formalized.

A relevant research area that is adjacent to CNAS is *Social Learning* (sometimes dubbed *collaborative learning*). Human intelligence could not have evolved through one individual in a single life-time; similarly, it is unlikely that a complex set of mechanisms could evolve or be learned in a relatively short period of time without knowledge-sharing across multiple artificial models. Recent works on ML started exploring the concept of Social Learning, where a ML model would share information with other models in the same domain to collectively boost performances [12], [160]. These Social and Federated Learning environments can be complementary to CNAS, especially given their lifelong and distributed learning nature.

Although unsupervised self-managing DNNs have been briefly explored in the literature [12], [48], they are exclusively researched from an evolutionary/synaptic plasticity perspective. The unsupervised counterparts of NAS and CL (dubbed Unsupervised NAS (UnNAS) [204] and Unsupervised Continual Learning (UCL) [205]) have both shown promising

potential individually, but their intersection has not been explored.

One of the primary bottlenecks for full autonomy in CNAS is the need for manual definitions of task boundaries. The biological counterpart of lifelong learning is contingent on spontaneous generation of goals, which are fundamentally driven by intrinsic primal disposition to survival and procreation [206]. Modelling these *goal-seeking neural networks* [207] is an open challenge in AI that should be investigated further to enable real autonomy.

VI. Conclusion

In this extensive review, we have explored the intersection between NAS and CL to formalize the prospective Continual Neural Architecture Search paradigm. Despite the existence of several continually-adaptive models (as per Table III), this intersection has not been reviewed in depth. By combining complementing techniques from CL and NAS, the potential of creating more robust and adaptive artificial neural networks can be realized. This review highlights the challenges of traditional deep learning design, and emphasizes the importance of continual adaptiveness in various domains.

Through self-development and lifelong plasticity, CNAS aims to overcome numerous limitations posed by human factors and result in more optimal models. In doing so, Continual Neural Architecture Search could potentially introduce a new spectrum of applications where the functionality of the model is pliable, even after the deployment phase when remote access is often limited or unavailable.

Additionally, CNAS takes a step towards generalizing intelligence; although the definition of "intelligence" has long eluded psychologist and neuroscientists, it has been repeatedly defined as "goal-directed adaptive behavior" and "adaptive self-modification" rather than mere associative inference [162], [208]. Narrow AI has indeed led to substantial accomplishments, however, there is an undeniable impending bottleneck to the capabilities that can be achieved with traditional methodologies. Continual adaptability and automation, on the other hand, minimize the static nature of Narrow AI and provide an essential basis for General AI.

The proposed categorization schemes and analysis of methodologies aim to propel researches towards a less dedicated form of intelligence and facilitate more robust models that are sensitive to their surrounding environments.

References

[1] A. J. Otoole, P. J. Phillips, F. Jiang, J. Ayyad, N. Penard, and H. Abdi, "Face recognition algorithms surpass humans matching faces over changes in illumination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, p. 1642–1646, 2007.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv:1602.07360*, 2016.

[6] I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. K. Ho, N. Macià, B. Ray, M. Saeed, A. Statnikov *et al.*, "Design of the 2015 chalearn automl challenge," in *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8.

[7] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, "Towards automatically-tuned neural networks," in *Proceedings of the Workshop on Automatic Machine Learning*, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., vol. 64. PMLR, 2016, pp. 58–65.

[8] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, M. Urban, M. Burkart, M. Dippel, M. Lindauer, and F. Hutter, "Towards automatically-tuned deep neural networks," *Automated Machine Learning*, p. 135–149, 2019.

[9] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2021.

[10] M. Mccloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning and Motivation*, p. 109–165, 1989.

[11] J. L. Mcclelland, B. L. Mcnaughton, and R. C. Oreilly, "Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory." *Psychological Review*, vol. 102, no. 3, p. 419–457, 1995.

[12] A. Soltoggio, K. O. Stanley, and S. Risi, "Born to learn: The inspiration, progress, and future of evolved plastic artificial neural networks," *Neural Networks*, vol. 108, p. 48–67, 2018.

[13] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, p. 54–71, 2019.

[14] G. Anthes, "Lifelong learning in artificial neural networks," *Communications of the ACM*, vol. 62, no. 6, p. 13–15, 2019.

[15] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, "Class-incremental learning via deep model consolidation," *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[16] Y. Luo, L. Yin, W. Bai, and K. Mao, "An appraisal of incremental learning methods," *Entropy*, vol. 22, no. 11, p. 1190, 2020.

[17] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, pp. 1–13, 2022.

[18] A. Robins, "Sequential learning in neural networks: A review and a discussion of pseudorehearsal based methods," *Intelligent Data Analysis*, vol. 8, no. 3, p. 301–322, 2004.

[19] J. M. Pierre, "Incremental lifelong deep learning for autonomous vehicles," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3949–3954.

[20] Z. Sun, Y. Wang, and R. Laganière, "Online model adaptation for uav tracking with convolutional neural network," in *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018, pp. 329–336.

[21] B. Wehbe, M. Hildebrandt, and F. Kirchner, "A framework for on-line learning of underwater vehicles dynamic models," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7969–7975, 2019.

[22] Y. Nose, A. Kojima, H. Kawabata, and T. Hironaka, "A study on a lane keeping system using cnn for online learning of steering control from real time images," in *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications*, 2019, pp. 1–4.

[23] W. Si, T. Wei, and C. Liu, "Agen: Adaptable generative prediction networks for autonomous driving," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 281–286.

[24] M. M. Ferdaus, M. Pratama, S. G. Anavatti, and M. A. Garratt, "Online identification of a rotary wing unmanned aerial vehicle from data streams," *Applied Soft Computing*, vol. 76, p. 313–325, 2019.

[25] K. Shaheen, M. A. Hanif, O. Hasan, and M. Shafique, "Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks," *CoRR*, vol. abs/2105.12374, 2021.

[26] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robotics and Autonomous Systems*, vol. 15, no. 1-2, p. 25–46, 1995.

[27] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information Fusion*, vol. 58, p. 52–68, 2020.

[28] S. Chen and J. T. Wen, "Adaptive neural trajectory tracking control for flexible-joint robots with online learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2358–2364.

[29] W. Zheng, H. Liu, and F. Sun, "Lifelong visual-tactile cross-modal learning for robotic material perception," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 3, pp. 1192–1203, 2021.

[30] A. Xie and C. Finn, "Lifelong robotic reinforcement learning by retaining experiences," in *Advances in Neural Information Processing Systems*, 2021.

[31] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021.

[32] Z. Yao, F. Xu, G.-P. Jiang, and J. Yao, "Data-driven control of hydraulic manipulators by reinforcement learning," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2023.

[33] Z. Yao, X. Liang, G.-P. Jiang, and J. Yao, "Model-based reinforcement learning control of electrohydraulic position servo systems," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 3, pp. 1446–1455, 2023.

[34] H.-C. Cheng and W.-W. Liao, "Establishing an lifelong learning environment using iot and learning analytics," in *2012 14th International Conference on Advanced Communication Technology (ICACT)*, 2012, pp. 1178–1183.

[35] M. Song, K. Zhong, J. Zhang, Y. Hu, D. Liu, W. Zhang, J. Wang, and T. Li, "In-situ ai: Towards autonomous and incremental deep learning for iot systems," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 92–103.

[36] Y. Bao and W. Chen, "Il4iot: Incremental learning for internet-of-things devices," *Lecture Notes in Computer Science*, p. 92–107, 2019.

[37] A. K. Chathoth, A. Jagannatha, and S. Lee, "Federated intrusion detection for iot with heterogeneous cohort privacy," 2021. [Online]. Available: https://arxiv.org/abs/2101.09878

[38] M. Rasouli, R. Ghosh, W. W. Lee, N. V. Thakor, and S. Kukreja, "Stable force-myographic control of a prosthetic hand using incremental learning," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015, pp. 4828–4831.

[39] A. Patra and J. A. Noble, "Hierarchical class incremental learning of anatomical structures in fetal echocardiography videos," *IEEE journal of biomedical and health informatics*, vol. 24, pp. 1046–1058, 2020.

[40] A. Elskhawy, A. Lisowska, M. Keicher, J. Henry, P. Thomson, and N. Navab, "Continual class incremental learning for ct thoracic segmentation," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*. Springer, 2020, pp. 106–116.

[41] J. Chauhan, Y. D. Kwon, P. Hui, and C. Mascolo, "Contauth," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 4, p. 1–23, 2020.

[42] S. Srivastava, M. Yaqub, K. Nandakumar, Z. Ge, and D. Mahapatra, "Continual domain incremental learning for chest x-ray classification in low-resource clinical settings," in *DART2021*. Springer, 2021, pp. 226–238.

[43] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Class-incremental learning for wireless device identification in iot," *IEEE Internet of Things Journal*, 2021.

[44] J. Park, M. Kang, and B. Han, "Class-incremental learning for action recognition in videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 698–13 707.

[45] P. Ren, Y. Xiao, X. Chang, P.-y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search," *ACM Computing Surveys*, vol. 54, no. 4, p. 1–34, 2022.

[46] G. Raghavan and M. Thomson, "Neural networks grown and self-organized by noise," 2019. [Online]. Available: https://arxiv.org/abs/1906.01039

[47] A. Kaplunovich and Y. Yesha, "Automatic hyperparameter optimization for arbitrary neural networks in serverless aws cloud," *2021 12th International Conference on Information and Communication Systems (ICICS)*, 2021.

[48] J.-B. Mouret and P. Tonelli, "Artificial evolution of plastic neural networks: a few key concepts," *Growing Adaptive Machines: combining Development and Learning in Artificial Neural Networks*, pp. 251–261, 2014.

[49] T. Miconi, K. Stanley, and J. Clune, "Differentiable plasticity: training plastic neural networks with backpropagation," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3559–3568.

[50] T. Nagarajah and G. Poravi, "A review on automated machine learning (automl) systems," in *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. IEEE, 2019, pp. 1–6.

[51] J. Waring, C. Lindvall, and R. Umeton, "Automated machine learning: Review of the state-of-the-art and opportunities for healthcare," *Artificial Intelligence in Medicine*, vol. 104, p. 101822, 2020.

[52] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[53] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Automated Machine Learning*, p. 63–77, 2019.

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

[55] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," 2019. [Online]. Available: https://arxiv.org/abs/1905.01392

[56] S. Grossberg, "Studies of mind and brain," *Boston Studies in the Philosophy of Science*, 1982.

[57] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in Psychology*, vol. 4, 2013.

[58] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6470–6479.

[59] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[60] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, *Gradient Based Sample Selection for Online Continual Learning*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[61] C. X. Ling and T. Bohn, "A deep learning framework for lifelong machine learning," 2021. [Online]. Available: https://arxiv.org/abs/2105.00157

[62] S. Farquhar and Y. Gal, "Towards robust evaluations of continual learning," *CoRR*, vol. abs/1805.09733, 2018.

[63] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[64] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *International Conference on Learning Representations*, 2018.

[65] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[66] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, "Replay in deep learning: Current approaches and missing biological elements," *Neural Computation*, vol. 33, no. 11, pp. 2908–2950, 2021.

[67] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," 2015. [Online]. Available: https://arxiv.org/abs/1503.02531

[68] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *arXiv preprint arXiv:2302.00487*, 2023.

[69] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1606.04671

[70] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3366–3375.

[71] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, p. 2935–2947, 2018.

[72] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Pro-*

ceedings of the national academy of sciences, vol. 114, no. 13, pp. 3521–3526, 2017.

[73] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3987–3995.

[74] N. Kamra, U. Gupta, and Y. Liu, "Deep generative dual memory network for continual learning," 2017. [Online]. Available: https://arxiv.org/abs/1710.10368

[75] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *Advances in neural information processing systems*, vol. 30, 2017.

[76] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetful learning for domain expansion in deep neural networks," in *AAAI*, 2018, pp. 3358–3365.

[77] R. Aljundi, M. Rohrbach, and T. Tuytelaars, "Selfless sequential learning," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[78] A. New, M. Baker, E. Nguyen, and G. Vallabha, "Lifelong learning metrics," *arXiv preprint arXiv:2201.08278*, 2022.

[79] N. Diaz-Rodriguez, V. Lomonaco, D. Filliat, and D. Maltoni, "Don't forget, there is more than forgetting: new metrics for continual learning," *arXiv preprint arXiv:1810.13166*, 2018.

[80] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[81] T. L. Hayes, R. Kemker, N. D. Cahill, and C. Kanan, "New metrics and experimental paradigms for continual learning," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.

[82] S. I. Mirzadeh, A. Chaudhry, D. Yin, T. Nguyen, R. Pascanu, D. Gorur, and M. Farajtabar, "Architecture matters in continual learning," *arXiv preprint arXiv:2202.00275*, 2022.

[83] M. Tenorio and W.-T. Lee, "Self organizing neural networks for the identification problem," *Advances in Neural Information Processing Systems*, vol. 1, 1988.

[84] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.

[85] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *International Conference on Learning Representations*, 2020.

[86] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[87] A. Brock, T. Lim, J. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *International Conference on Learning Representations*, 2018.

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

[89] M. Zhang, H. Li, S. Pan, X. Chang, and S. Su, "Overcoming multi-model forgetting in one-shot nas with diversity maximization," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7806–7815.

[90] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019.

[91] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019.

[92] X. Dong and 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*, 2019, pp. 1761–1770.

[93] H. Zhou, M. Yang, J. Wang, and W. Pan, "Bayesnas: A bayesian approach for neural architecture search," in *International conference on machine learning*. PMLR, 2019, pp. 7603–7613.

[94] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 19–34.

[95] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," *Advances in neural information processing systems*, vol. 31, 2018.

[96] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *CoRR*, vol. abs/1611.02167, 2016.

[97] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[98] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," in *International Conference on Machine Learning*. PMLR, 2018, pp. 678–687.

[99] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*. PMLR, 2018, pp. 4095–4104.

[100] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. L. P. Chen, "Bnas: Efficient neural architecture search using broad scalable architecture," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[101] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[102] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, no. 2, pp. 185–212, 2009.

[103] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2902–2911.

[104] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *International Conference on Learning Representations*, 2018.

[105] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4780–4789.

[106] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *International Conference on Learning Representations*, 2019.

[107] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: Neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2019.

[108] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *European Conference on Computer Vision*. Springer, 2020, pp. 544–560.

[109] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "Cars: Continuous evolution for efficient neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1829–1838.

[110] R. Niu, H. Li, Y. Zhang, and Y. Kang, "Neural architecture search based on particle swarm optimization," in *2019 3rd International Conference on Data Science and Business Analytics (ICDSBA)*, 2019, pp. 319–324.

[111] E. Byla and W. Pang, "Deepswarm: Optimising convolutional neural networks using swarm intelligence," in *UK Workshop on Computational Intelligence*. Springer, 2019, pp. 119–130.

[112] S. Mo, J. Xia, and P. Ren, "Simulated annealing for neural architecture search," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[113] M. Shahawy and E. Benkhelifa, "Hivenas: Neural architecture search using artificial bee colony optimization," *arXiv preprint arXiv:2211.10250*, 2022.

[114] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 726–10 734.

[115] H. Hu, J. Langford, R. Caruana, S. Mukherjee, E. J. Horvitz, and D. Dey, "Efficient forward architecture search," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

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

[117] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[118] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2423–2432.

[119] L. Wu, B. Liu, P. Stone, and Q. Liu, "Firefly neural architecture descent: a general approach for growing neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 373–22 383, 2020.

[120] Y. Ci, C. Lin, M. Sun, B. Chen, H. Zhang, and W. Ouyang, "Evolving search space for neural architecture search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6659–6669.

[121] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," 2019. [Online]. Available: https://arxiv.org/abs/1912.06059

[122] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[123] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021.

[124] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms." in *ICGA*, vol. 89, 1989, pp. 379–384.

[125] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1379–1388.

[126] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 497–504.

[127] R. Shin, C. Packer, and D. Song, "Differentiable neural network architecture search," in *ICLR*, 2018.

[128] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[129] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[130] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," *arXiv preprint arXiv:1910.11858*, vol. 1, no. 2, p. 4, 2019.

[131] S. Lankford and D. Grimes, "Neural architecture search using particle swarm and ant colony optimization." in *AICS*, 2020, pp. 229–240.

[132] E. Suda and H. Iba, "Neural architecture search and weight adjustment by means of ant colony optimization," in *2020 5th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*. IEEE, 2020, pp. 184–191.

[133] S. Hu, S. Xie, H. Zheng, C. Liu, J. Shi, X. Liu, and D. Lin, "Dsnas: Direct neural architecture search without parameter retraining," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 084–12 092.

[134] G. Kyriakides and K. Margaritis, "An introduction to neural architecture search for convolutional networks," 2020. [Online]. Available: https://arxiv.org/abs/2005.11074

[135] Z. Chen and B. Li, "Efficient evolution for neural architecture search," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[136] A. Klein, E. Christiansen, K. Murphy, and F. Hutter, "Towards reproducible neural architecture and hyperparameter search," 2018. [Online]. Available: https://openreview.net/pdf?id=rJeMCSnml7

[137] P. Liu, M. D. El Basha, Y. Li, Y. Xiao, P. C. Sanelli, and R. Fang, "Deep evolutionary networks with expedited genetic algorithms for medical image denoising," *Medical image analysis*, vol. 54, pp. 306–315, 2019.

[138] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "Autogan: Neural architecture search for generative adversarial networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3224–3234.

[139] Y. Xu, Y. Wang, K. Han, Y. Tang, S. Jui, C. Xu, and C. Xu, "Renas: Relativistic evaluation of neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4411–4420.

[140] A. Zela, A. Klein, S. Falkner, and F. Hutter, "Towards automated deep learning: Efficient joint neural architecture and hyperparameter search," 2018. [Online]. Available: https://arxiv.org/abs/1807.06906

[141] I. Trofimov, N. Klyuchnikov, M. Salnikov, A. Filippov, and E. Burnaev, "Multi-fidelity neural architecture search with knowledge distillation," *arXiv preprint arXiv:2006.08341*, 2020.

[142] S. Yang, Y. Tian, X. Xiang, S. Peng, and X. Zhang, "Accelerating evolutionary neural architecture search via multi-fidelity evaluation," 2021. [Online]. Available: https://arxiv.org/abs/2108.04541

[143] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[144] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," in *ICLR*, 2016.

[145] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "Nas-unet: Neural architecture search for medical image segmentation," *IEEE Access*, vol. 7, pp. 44 247–44 257, 2019.

[146] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," in *ICLR*, 2018.

[147] R. Ru, C. Lyle, L. Schut, M. Fil, M. van der Wilk, and Y. Gal, "Speedy performance estimation for neural architecture search," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[148] C. White, A. Zela, R. Ru, Y. Liu, and F. Hutter, "How powerful are performance predictors in neural architecture search?" *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[149] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *35th ICML*, J. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 550–559.

[150] B. M. Oloulade, J. Gao, J. Chen, T. Lyu, and R. Al-Sabri, "Graph neural architecture search: A survey," *Tsinghua Science and Technology*, vol. 27, no. 4, pp. 692–708, 2021.

[151] H. Abed and B. Gyires-Tóth, "Efficient neural architecture search for long short-term memory networks," in *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2021, pp. 000 287–000 292.

[152] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," in *ICML*. PMLR, 2019, pp. 7105–7114.

[153] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," *arXiv preprint arXiv:2001.00326*, 2020.

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

[155] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, A. Filippov, and E. Burnaev, "Nas-bench-nlp: neural architecture search benchmark for natural language processing," *IEEE Access*, vol. 10, pp. 45 736–45 747, 2022.

[156] X. Zhang, H. Xu, H. Mo, J. Tan, C. Yang, L. Wang, and W. Ren, "Dcnas: Densely connected neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 956–13 967.

[157] J.-H. Lee, J.-H. Chang, J.-M. Yang, and H.-G. Moon, "Nas-tasnet: Neural architecture search for time-domain speech separation," *IEEE Access*, vol. 10, pp. 56 031–56 043, 2022.

[158] R. K. Thomas, "Evolution of intelligence: An approach to its assessment," *Brain, Behavior and Evolution*, vol. 17, no. 6, pp. 454–472, 1980.

[159] E. Stern, "Individual differences in the learning potential of human beings," *npj Science of Learning*, vol. 2, no. 1, pp. 1–7, 2017.

[160] R. Boyd, P. J. Richerson, and J. Henrich, "The cultural niche: Why social learning is essential for human adaptation," *Proceedings of the National Academy of Sciences*, vol. 108, no. Supplement 2, pp. 10 918–10 925, 2011.

[161] E. Baum, M. Hutter, and E. Kitzelmann, "Artificial general intelligence," in *Proceedings of the Third Conference on Artificial General Intelligence*. Citeseer, 2010.

[162] B. Goertzel, "Artificial general intelligence: concept, state of the art, and future prospects," *Journal of Artificial General Intelligence*, vol. 5, no. 1, p. 1, 2014.

[163] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[164] R. Palnitkar and J. Cannady, "A review of adaptive neural networks," in *IEEE SoutheastCon, 2004. Proceedings.*, 2004, pp. 38–47.

[165] M. Ghifary, W. B. Kleijn, and M. Zhang, "Domain adaptive neural networks for object recognition," in *Pacific Rim international conference on artificial intelligence*. Springer, 2014, pp. 898–904.

[166] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–11, 2018.

[167] R. Ashmore, R. Calinescu, and C. Paterson, "Assuring the machine learning lifecycle: Desiderata, methods, and challenges," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–39, 2021.

[168] S. Munzert, C. Rubba, P. Meißner, and D. Nyhuis, *Automated data collection with R: A practical guide to web scraping and text mining*. John Wiley & Sons, 2014.

[169] Y. N. Kunang, S. D. Purnamasari *et al.*, "Web scraping techniques to collect weather data in south sumatera," in *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*. IEEE, 2018, pp. 385–390.

[170] R. Diouf, E. N. Sarr, O. Sall, B. Birregah, M. Bousso, and S. N. Mbaye, "Web scraping: state-of-the-art and areas of application," in *2019 IEEE International Conference on Big Data*. IEEE, 2019, pp. 6040–6042.

[171] M. Dojchinovski and T. Vitvar, "Linked web apis dataset," *Semantic Web*, vol. 9, no. 4, pp. 381–391, 2018.

[172] V. Krotov and L. Silva, "Legality and ethics of web scraping," 2018.

[173] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[174] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. V. Hernández, J. Wardlaw, and D. Rueckert, "Gan augmentation: Augmenting training data using generative adversarial networks," *arXiv preprint arXiv:1810.10863*, 2018.

[175] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1887–1893.

[176] A. Kar, A. Prakash, M.-Y. Liu, E. Cameracci, J. Yuan, M. Rusiniak, D. Acuna, A. Torralba, and S. Fidler, "Meta-sim: Learning to generate synthetic datasets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4551–4560.

[177] P. Li, X. Rao, J. Blase, Y. Zhang, X. Chu, and C. Zhang, "Cleanml: a study for evaluating the impact of data cleaning on ml classification tasks," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 13–24.

[178] S. Krishnan, M. J. Franklin, K. Goldberg, and E. Wu, "Boostclean: Automated error detection and repair for machine learning," *arXiv preprint arXiv:1711.01299*, 2017.

[179] S. Krishnan and E. Wu, "Alphaclean: Automatic generation of data cleaning pipelines," *arXiv preprint arXiv:1904.11827*, 2019.

[180] M. Volkovs, F. Chiang, J. Szlichta, and R. J. Miller, "Continuous data cleaning," *2014 IEEE 30th International Conference on Data Engineering*, pp. 244–255, 2014.

[181] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, pp. 1–48, 2019.

[182] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong, "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3925–3934.

[183] S. Huang, V. François-Lavet, and G. Rabusseau, "Neural architecture search for class-incremental learning," *arXiv preprint arXiv:1909.06686*, 2019.

[184] R. Pasunuru and M. Bansal, "Continual and multi-task architecture search," *arXiv preprint arXiv:1906.05226*, 2019.

[185] M. Zhang, H. Li, S. Pan, X. Chang, Z. Ge, and S. W. Su, "Differentiable neural architecture search in equivalent space with exploration enhancement," in *NeurIPS*, 2020.

[186] J. Zhang, J. Zhang, S. Ghosh, D. Li, J. Zhu, H. Zhang, and Y. Wang, "Regularize, expand and compress: Nonexpansive continual learning," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 843–851.

[187] X. Du, Z. Li, J. Sun, F. Liu, and Y. Cao, "Evolutionary nas in light of model stability for accurate continual learning," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[188] S. Niu, J. Wu, G. Xu, Y. Zhang, Y. Guo, P. Zhao, P. Wang, and M. Tan, "Adaxpert: Adapting neural architecture for growing data," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8184–8194.

[189] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, and S. Su, "One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2921–2935, 2021.

[190] Q. Qin, W. Hu, H. Peng, D. Zhao, and B. Liu, "BNS: Building network structures dynamically for continual learning," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[191] M. Mundt, I. Pliushch, and V. Ramesh, "Neural architecture search of deep priors: Towards continual learning without catastrophic interference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3523–3532.

[192] Q. Gao, Z. Luo, D. Klabjan, and F. Zhang, "Efficient architecture search for continual learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2022.

[193] J. Jiang and O. Celiktutan, "Neural weight search for scalable task incremental learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 1390–1399.

[194] A. Lu, T. Feng, H. Yuan, X. Song, and Y. Sun, "Revisiting neural networks for continual learning: An architectural perspective," *arXiv preprint arXiv:2404.14829*, 2024.

[195] C. Lee, Y.-B. Kim, H. Ji, Y. Lee, Y. Hur, and H. Lim, "On the redundancy in the rank of neural network parameters and its controllability," *Applied Sciences*, vol. 11, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/2/725

[196] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *2013 IEEE ICASSP*, 2013, pp. 6655–6659.

[197] X.-D. Ren, H.-N. Guo, G.-C. He, X. Xu, C. Di, and S.-H. Li, "Convolutional neural network based on principal component analysis initialization for image classification," in *2016 IEEE first international conference on data science in cyberspace (DSC)*. IEEE, 2016, pp. 329–334.

[198] V. V. de Melo and W. Banzhaf, "Automatic feature engineering for regression models with machine learning: An evolutionary computation and statistics hybrid," *Information Sciences*, vol. 430, pp. 287–313, 2018.

[199] N. R. Júnior and J. C. Nievola, "A generalized financial time series forecasting model based on automatic feature engineering using genetic algorithms and support vector machine," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[200] X. Chen, Q. Lin, C. Luo, X. Li, H. Zhang, Y. Xu, Y. Dang, K. Sui, X. Zhang, B. Qiao, W. Zhang, W. Wu, M. Chintalapati, and D. Zhang, "Neural feature search: A neural architecture for automated feature engineering," in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 71–80.

[201] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *International Conference on Learning Representations*, 2020.

[202] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, p. 24–35, 2019.

[203] Z. Wang, E. Yang, L. Shen, and H. Huang, "A comprehensive survey of forgetting in deep learning beyond continual learning," *arXiv preprint arXiv:2307.09218*, 2023.

[204] C. Liu, P. Dollár, K. He, R. Girshick, A. Yuille, and S. Xie, "Are labels necessary for neural architecture search?" in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*. Springer, 2020, pp. 798–813.

[205] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, "Continual unsupervised representation learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[206] A. Gopnik, A. N. Meltzoff, and P. K. Kuhl, *The scientist in the crib: Minds, brains, and how children learn.* William Morrow & Co, 1999.

[207] T. E. Portegys, "Instinct evolution in a goal-seeking neural network." in *Computational Intelligence*. Citeseer, 2006, pp. 210–215.

[208] R. J. Sternberg, *Handbook of human intelligence*. Cambridge University Press, 1982.