SEAL: Searching Expandable Architectures for Incremental Learning

Matteo Gambella, Vicente Javier Castro Solar, and Manuel Roveri

Politecnico di Milano, Milano, Italy
Email: vicentejavier.castro@mail.polimi.it, {matteo.gambella, manuel.roveri}@polimi.it

Abstract. Incremental learning is a machine learning paradigm where a model learns from a sequential stream of tasks. This setting poses a key challenge: balancing plasticity (learning new tasks) and stability (preserving past knowledge). Neural Architecture Search (NAS), a branch of AutoML, automates the design of the architecture of Deep Neural Networks and has shown success in static settings. However, existing NAS-based approaches to incremental learning often rely on expanding the model at every task, making them impractical in resource-constrained environments. In this work, we introduce SEAL, a NAS-based framework tailored for data-incremental learning, a scenario where disjoint data samples arrive sequentially and are not stored for future access. SEAL adapts the model structure dynamically by expanding it only when necessary, based on a capacity estimation metric. Stability is preserved through cross-distillation training after each expansion step. The NAS component jointly searches for both the architecture and the optimal expansion policy. Experiments across multiple benchmarks demonstrate that SEAL effectively reduces forgetting and enhances accuracy while maintaining a lower model size compared to prior methods. These results highlight the promise of combining NAS and selective expansion for efficient, adaptive learning in incremental scenarios.

1 Introduction

Incremental Learning (IL), or Continual Learning, is a special set of ML where the models are set to learn a sequential stream of tasks [33]. Under this setting, each task defines a set of observations from which the model needs to learn to perform an action. As the training on new tasks progresses, Deep Neural Networks (DNNs) usually present two unwanted properties: catastrophic forgetting [32] that is the phenomenon of forgetting knowledge from previously seen tasks, and plasticity loss or knowledge saturation [7] that is the inability to keep learning additional tasks. For this reason, many IL solutions focus on mitigating these behaviors. From the stability perspective, some works penalize changes on critical parameters on prior tasks [18, 37], or using knowledge distillation [21]. From the plasticity perspective, promising methods are dynamically expanding networks, that are able to add new weights to the network in order to increase the capacity towards new incoming tasks [36, 27]. IL is particularly useful in many relevant scenarios such as Internet-of-Things and TinyML where the focus is on ensuring efficient adaptation in dynamic and resource-constrained environments [28].

Furthermore, additional work has been developed around the loss landscape geometry, guiding the optimization algorithm towards flatter loss regions to increase robustness of the models, being more resistant to perturbations [9, 29, 15]. These flat solutions have been shown to largely improve the performance in continual learning systems since new incoming tasks can be seen as perturbations of the models [25].

Another direction in ML research is Automated Machine Learning (AutoML), the development of automatic approaches to build more efficient ML pipelines. Within this field, Neural Architecture Search (NAS) aims at automating the design of NN architectures, optimizing them for a specific problem [35]. Many works about NAS have highlighted the importance of flatness to boost the generalization of the searched models [11, 34, 14, 26]. NAS has successfully been applied to IL problems, specifically to automatically search for the optimal model expansion on new tasks arrivals [31, 12, 20]. However, the current approaches perform the search on every task arrival, requiring high computational resources.

This paper proposes an alternative usage for NAS in IL, being the first work jointly searching for the optimal expansion policy and NN architecture. This allows our method to be highly efficient in production, reducing the resources and time needed to train on new task arrivals. Moreover, the method is framed under a specific type of incremental learning problem: a data-incremental scenario [32]. In this context, tasks are defined as disjoint sets of samples, maintaining the same number of classes. We refer to this sets as *sample sets*, to differentiate the term from traditional IL. The proposed framework, called Searching Expandable Architectures for Incremental Learning (SEAL), utilizes a similar design to FlatNAS [11], using an OFA supernet for faster evaluation of the models [3] and adaptative switching for surrogate prediction [24]. Additionally, in order to accelerate the evaluation and improve the evaluation performance of the expanded models, we define the expansion operator to allow all expanded models to be valid OFA sub-network and perform their evaluation by fine-tuning after initializing the new parts of the expanded models with the pre-trained weights of the OFA supernet.

We tested SEAL on three different datasets and showed that the models retrieved by the framework are competitive with traditional IL methods, while requiring fewer memory resources. Furthermore, the inclusion of a flatness metric showed promising improvements in the IL metrics.

The work is structured as follows. Section 2 introduces the related works regarding NAS solutions for incremental learning. Section 3 provides the background. Section 4 introduces the SEAL framework

developed in this study, while in Section 5 the experimental results aiming at evaluating the effectiveness of SEAL are shown. Conclusions are finally drawn in Section 6. To facilitate comparisons and reproducibility, the source code of SEAL is released to the scientific community as a public repository. [1]

2 Related Literature

In this section, we first present techniques for IL and then discuss NAS methods addressing flatness and the incremental problem.

Regarding model expansion, Dynamic Expandable Networks (DEN) [36] were among the first methods to effectively address the plasticity-stability trade-off. DEN enforces sparse weight representations during training to isolate the most relevant parameters for each task. The decision to expand is determined by applying a hard threshold on the pre-computed loss over incoming samples. AdaXpert [27] performs dynamic expansions based on task similarity. The decision to expand is determined using the Wasserstein distance between the distributions of new and previously learned data. The extent of the required expansion is computed from the performance gap. Elastic Weight Consolidation (EWC) [18] mitigates catastrophic forgetting by applying a regularization term to penalize changes to parameters critical for prior tasks, using the Fisher Information Matrix. Synaptic Intelligence (SI) [37] employs a similar regularization strategy but dynamically tracks each weight's contribution to the loss during training. Learning without Forgetting (LwF) [21] preserves knowledge from previous tasks without requiring old data, using knowledge distillation to retain prior knowledge while learning new tasks. Starting from a static tiny model, Tybox [28] incrementally designs on-device versions of TinyML models while adhering to RAM constraints of target devices. Tybox's approach seamlessly balances expansion and resource efficiency in memory-constrained environments.

FlatNAS [11], built on a constrained NAS framework [10], introduces a robustness metric grounded in a rescaling-invariant notion of flatness [15, 29] to design architectures resilient to out-of-distribution (OOD) samples while maintaining a constraint on the number of parameters. By incorporating local energy flatness into its objective, which balances accuracy and robustness, FlatNAS effectively guides models toward robust solutions. Other NAS methods leveraging flatness, though not explicitly targeting OOD tasks, include ADVRUSH [26], GeNAS [14], and NA-DARTS [34].

ENAS-S [8] is the first NAS-based solution explicitly tackling the incremental problem, focusing on architectures suitable for edge devices. Their genetic search algorithm optimizes for accuracy and stability, where stability is defined as the accuracy degradation caused by one-shot noise injection in each model layer. ENAS-S demonstrates superior performance in class-incremental tasks compared to traditional CNN models. CLEAS [12] is a NAS using a reinforcement learning controller to have neuron-level control on the expansion search and it is especially effective for catastrophic forgetting as it explicitly isolates previous weights before each expansion. Learn-to-Grow [20] uses DARTS during the search, achieving excellent results. However, the current approaches perform the search on every task arrival, requiring high computational resources. Differently, in our work, we consider all the arrivals in a single search for computation saving and focus specifically on data-incremental tasks.

[1] https://github.com/AI-Tech-Research-Lab/SEAL

3 Technical background

Neural Architecture Search (NAS) solutions can be categorized across three key dimensions [35].

The first dimension is the Search Space, which defines the architecture representation. The simplest approach is the entire-structured Search Space, where the architecture is depicted layer-wise, with each node representing a single layer. Inspired by handcrafted architectures with repetitive motifs [13], the cell-based Search Space defines each node as a cell (or block), which represents a group of layers. Variations of this include hierarchical cells or morphism-based identity transformations between layers.

The second dimension is the Search Strategy, which governs how the search space is explored. Common strategies include reinforcement learning [16], gradient optimization [22], and evolutionary algorithms [2, 1]. Among these, genetic algorithms—particularly NSGA-II [24]—are widely used. NSGA-II generates offspring via specific crossover and mutation operations, selecting the next generation based on fitness through non-dominated sorting and crowding distance comparisons.

The third dimension is the Performance Estimation Strategy, which aims to estimate the performance of candidate architectures without fully training each one, a process that would be computationally expensive. A popular strategy is weight sharing, with Once-For-All (OFA) [3] being a leading example. OFA involves training a supernet that encompasses multiple network configurations once before the search, and candidate networks are evaluated by fine-tuning from the pre-trained supernet weights during every NAS search process. Another approach utilizes surrogate models, such as Gaussian Processes and Radial Basis Functions. For example, MSuNAS [24] introduces an adaptive-switching technique that selects the most accurate predictor among four surrogate models based on a correlation metric, Kendall's Tau [17], at each NAS iteration.

3.1 Incremental Learning

IL represents a special kind of ML problem, where models are trained on a sequential stream of *tasks*, instead of single static dataset. Under this setting, each task consist of a set of observations, following the same or different data distributions, from which the model needs to learn to perform an action. The specific combination of observation and output depends on the type of incremental learning we are solving. IL is considered an open problem for deep learning methods as, experimentally, it has been noted that neural networks are prone to loose previously learned knowledge when training on new tasks [32], an effect known as *catastrophic forgetting*. Furthermore, their ability to keep learning new tasks is also diminished with the increase of training sessions [7].

Consider a task-agnostic framework, with $\mathcal{D}_t$ being a dataset associated to a task T_t from a time step t. The main objective of an incremental model is to learn the optimal weights w^* from a stream of tasks $\{T_0, ..., T_t\}$, such that the joint loss function over their datasets is minimized at any given time t.

$$w_t^* = \min_w \sum_{i=0}^{t} \mathcal{L}(D_i^*|w)$$

In this work, we address a problem of data-incremental learning where disjointed dataset samples with same labels arrive sequentially and are not retained in memory for the learner to revisit in the future. Just to mention, other relevant types of settings are

domain-incremental, class-incremental, and task-incremental learning [32, 33].

Commonly, there are three relevant aspects when evaluating the performance of continual learning models: overall performance on all seen tasks; knowledge-stability or retaining previously learned knowledge and plasticity for future tasks [33]. Normally, stability and plasticity are opposites in formulation, and a trade-off is required to balance what is more relevant for the model.

Accuracy is the most commonly used metric to measure general performance. Incrementally, we evaluate it with the *average accuracy* and the *average incremental accuracy* [4], [33]. The first metric computes how good the model is at solving the tasks for a specific timestamp t, while the second adds an average by time, informing the overall progress in time. Consider a_{ki} the *accuracy* on the task i after training on task k, notice $k > i$:

$$\text{Average Accuracy } = AA_k = \frac{1}{k}\sum_{i=0}^{k} a_{ki}$$

$$\text{Average Incremental Accuracy} = AIA_k = \frac{1}{k}\sum_{i=0}^{k} AA_k$$

Forgetting is the tendency of the model to replace previously learned tasks with new information. There are two commonly used metrics related to this property: the *average forgetting* [4] and *backward-transfer* (BWT) [23], the second measures the influence that learning a new task has for all previously learned tasks.

$$\text{Forgetting} = f_i^k = (\max_{j\in\{0\dots k-1\}} a_{ji}) - a_{ki}$$

$$\text{Average Forgetting} = F_k = \frac{1}{k}\sum_{i=0}^{k} f_i^k$$

$$\text{Backward Transfer} = BWT_k = \frac{1}{k}\sum_{i=0}^{k}(a_{ki} - a_{ii})$$

Then, f_i^k is the forgetting of the task i after training on k tasks.

For a data-incremental setup, the tasks are intrinsically related, so the $\max$ accuracy for a task will not be necessarily immediately after its training session, making *average forgetting* a more reliable measurement.

Finally, the plasticity of a network relates to its ability to learn new tasks. Although there is no standard measurement for this metric, a common proxy is the *forward transfer (FWT)* [23] or the influence that learning a task has in future tasks.

$$\text{Forward Transfer} = FWT_k = \frac{1}{k}\sum_{i=1}^{k}(a_{(i-1)i} - \bar{b}_i)$$

Where $\bar{b}_i$ represent the accuracy achieved by a randomly initialized model trained on the task i. In other words, this metric measures the added performance improvement by the incremental training before the task.

4 The proposed SEAL

This section, introducing and detailing SEAL, is organized as follows: Section 4.1 provides the problem formulation, Section 4.2 provides an overall description of SEAL, Section 4.3 describes the incremental setting addressed by our framework, Section 4.4 explains the expansion operator used to expand the candidate NNs, Section 4.5 provides the details about the training of the candidate NNs, Section 4.6 presents the figure of merits accounting for accuracy, the number of parameters and flatness of the incremental model.

4.1 Problem formulation

SEAL addresses the problem of selecting a NN architecture and its expansion in a data-incremental learning setting by simultaneously optimizing incremental accuracy and robustness to model expansion. SEAL can be defined as the following optimization problem:

$$\begin{aligned} &\text{minimize } \mathcal{G}\left(\mathcal{F}(\bar{x},\vec{d},w),\mathcal{H}(\bar{x},\vec{d},\sigma)\right) \\ &\text{s. t. } \sum_{i=0}\vec{d}_i > 0, \\ &\bar{x}\in\Omega_{\bar{x}} \end{aligned} \tag{1}$$

where $\mathcal{G}$ is a multi-objective optimization function, $\vec{d}$ is the expansion operator, $\bar{x}$ and $\Omega_{\bar{x}}$ represent a candidate NN architecture and the search space of the NN exploration, respectively; the novel metrics $\mathcal{F}(\bar{x},\vec{d},w)$ and $\mathcal{H}(\bar{x},\vec{d},\sigma)$ calculate the accuracy in the incremental setting and the robustness to model expansion respectively, and will be introduced in Section 4.6. The constraint on the expansion vector ensures that at least one expansion is applied. The optimization problem outlined in Eq. 1 is addressed by the SEAL framework, which is detailed in what follows.

4.2 The overall view of SEAL

The SEAL framework, illustrated in Fig. 1, operates as follows. It receives a dataset $\mathcal{DS}$, which includes a training set for training candidate networks and a validation set for their validation. The framework utilizes a type of OFA supernet to select a set of candidate networks Ω_x, defining the *Search Space* of the NAS. It receives a K parameter defining the number of splits of the dataset for the data-incremental setting. The inputs w and σ are parameters used by the first and second objectives of the NAS, respectively.

The *Search Space* module identifies a constrained search space $\Omega_{\bar{x}}$ inside the whole search space to sample only the candidate networks used as base models for the expansion. Additionally, it determines $\Omega_{\vec{d}}$, which is the search space for the possible expansions, ensuring at least one expansion. The pretrained weights of the expanded networks are sampled from the original search space Ω_x.

The Evaluator module receives a NN architecture from the set $\Omega_{\bar{x}}$ and trains it by using the $\mathcal{DS}$ dataset following the data-incremental setting with K splits. Subsequently, it assesses incremental accuracy and robustness. The resulting output is an *archive*, i.e., a collection of tuples $\langle\bar{x},\vec{d},\mathcal{F}(\bar{x},\vec{d},w),\mathcal{H}(\bar{x},\vec{d},\sigma)\rangle$, representing the NN architecture, the expansion vector, and their corresponding novel figure of merits, i.e., $\mathcal{F}(\bar{x},\vec{d},w)$ for the incremental accuracy, and $\mathcal{H}(\bar{x},\vec{d},\sigma)$ for the robustness to model expansion. Initially, a representative subset of the $\Omega_{\bar{x}}$ is sampled before starting the search. The number N_{start} of selected NN architectures is determined by the user and it is typically set to 100. Therefore, the initial size of the archive is equal to N_{start}.

In SEAL, the optimization problem outlined in Eq. 1 is reformulated as follows:

$$\begin{aligned} &\text{minimize } \mathcal{G}\left(\mathcal{S}_{\mathcal{F}}(\bar{x},\vec{d}),\mathcal{S}_{\mathcal{H}}(\bar{x},\vec{d})\right) \\ &\text{s. t. } \sum_{i=0}\vec{d}_i > 0, \\ &\bar{x}\in\Omega_{\bar{x}} \end{aligned} \tag{2}$$

where $\mathcal{S}_{\mathcal{F}}(\bar{x},\vec{d})$ is the predicted value of $\mathcal{F}(\bar{x},\vec{d})$ as estimated by using the incremental accuracy surrogate model, and $\mathcal{S}_{\mathcal{H}}(\bar{x},\vec{d})$ is the predicted value of $\mathcal{H}(\bar{x},\vec{d})$ by using the robustness surrogate model.

The core of SEAL is the *Search Strategy* module. It adopts the NSGA-II genetic algorithm to solve the bi-objective problem introduced in Eq. 2, by optimizing the objectives $\mathcal{S}_{\mathcal{F}}(\bar{x},\vec{d})$ and $\mathcal{S}_{\mathcal{H}}(\bar{x},\vec{d})$. The search process is iterative: at each iteration, the two surrogate models are chosen by employing a mechanism called *adaptive-switching*, the same method used by MSuNAS, which selects the best surrogate model according to a correlation metric (i.e., Kendall's Tau). Then, a ranking of the candidate NNs, based on $\mathcal{S}_{\mathcal{F}}(\bar{x},\vec{d})$ and $\mathcal{S}_{\mathcal{H}}(\bar{x},\vec{d})$, is computed, and a new set of candidates is obtained by the genetic algorithm and forwarded to the *Evaluator*. The *Evaluator* updates the *archive*, which becomes available for evaluation in the next iteration. At the end of the search, SEAL returns the set of the M NN architectures and expansions $X^{\circ} = \{(\bar{x}_1,\vec{d}_1)\ldots,(\bar{x}_M,\vec{d}_M)\}$ characterized by the best trade-off among the objectives, where M is a user-specified value.

4.3 Incremental Setting

The measure of the capacity of the current weights to learn the incoming samples is crucial to know when to expand. As in previous works, we measure the cross-entropy loss of the model weights with respect to the incoming samples [36], but instead of applying a threshold on this value, the loss is compared with the previous training iterations. This metric is more robust than setting a threshold directly on the loss value, as the comparison with the previous losses grounds the estimator to a problem the model has already seen and trained on. Formally, given W_{t-1}, the weights of the models trained until the last observed sample set and $\mathcal{L}_{t-1}$, the losses stored when receiving the previous sample set. The model is expanded if

$$1 - \frac{\mathcal{L}(W_{t-1},\mathcal{D}_t)}{\mathcal{L}_{t-1}} < \tau$$

with the hyperparameter τ defined as the capacity threshold.

4.4 Expansion of a candidate NN

In an OFA setup, we aim at utilizing as much as possible the weights and architecture already trained on the OFA set. To this end, the constrained OFA set $\Omega_{\bar{x}}$ ensures that each model expansion always produces another model inside the OFA set. This approach reuses OFA weights for expanded sections, reducing fine-tuning time. The weights of the original sections are inherited from the original model. We can specify the expansion of a NN architecture $\bar{x}$ by sampling a 3-elements vector $\vec{d}$ from the IL set $\Omega_{\vec{d}}$. These three bits define three directions on which a model may be expanded, being the depth, width, and kernel size of the candidate NN. Each expansion is applied to the first available OFA-compliant block (e.g., depth expansion can be applied only if the depth of the block is less than the maximum allowed in the OFA set) starting from the last block of layers. The Fig. 2 depicts all the possible expansions in the case of a convolutional layer. In the experimental section, we discuss why the procedure start from the last block instead of the first ones as shown in Fig. 5.

4.5 Training with cross-distillation

Expanding the model and so increasing model plasticity to learn new samples, nevertheless, may introduce interference in previously learned representations. To avoid this issue, after each expansion, the model is trained with a *cross-distillation* loss. This forces the added weights to keep a functional representation close to the one before the expansion, reducing catastrophic forgetting [21]. The distillation loss uses the *Kullback-Leiber Divergence Loss (KL)* on soft targets created by passing the new samples through the old model. The final loss for the expansion follows:

$$\mathcal{L}_{CD} = (1-\alpha)\mathcal{L}_{CE}(W_t,\mathcal{D}_t) + \alpha\mathcal{L}_{KL}(W_{t-1},D_t)$$

As we are already using the distillation loss to avoid forgetting, the proposed method allows re-training of all model weights after each expansion, thus avoiding the possible problems of keeping frozen the parameters computed on the previous task.

4.6 The novel figure of merits of SEAL

The framework casts a multi-objective optimization problem, with the first objective being a relation between the *average accuracy* at the last incremental sample set and the *number of parameters* of the model. We define the first objective as follows:

$$\mathcal{F}(\bar{x},\vec{d},w) = (1-\mathrm{AA_K}(\bar{x},\vec{d}))\cdot \mathtt{size}(\bar{x},\vec{d})^{w},\ \ w=0.07$$

where $\mathtt{size}(\bar{x},\vec{d})$ is the number of parameters of the expanded model.

For the second objective, we use a rescaling-invariant flatness metric, a similar measurement used in [11]. In this definition, the flatness is measured as the change in performance when perturbing the weights under multiplicative noise. Given σ the noise intensity, the final flatness is evaluated using a neighborhood of N (i.e., sampling N perturbations):

$$\Delta\mathrm{ACC}_K = \|\mathrm{ACC}_{\mathcal{D}_K}(W_K) - \mathrm{ACC}_{\mathcal{D}_K}(W_K+\sigma\cdot z_i\odot W_K)\|$$

$$\mathcal{H}(\bar{x},\vec{d},\sigma) = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathrm{ACC}_{\mathcal{D}_K}(W_K)-\Delta\mathrm{ACC}_K(W_K,\sigma)}{\mathrm{ACC}_{\mathcal{D}_K}(W_K)}$$

where $\odot$ denotes the element-wise product, z_i is a perturbation sampled by a Gaussian distribution, W_K are the weights of the NN at the last task.

5 Experimental Results

5.1 Datasets

We evaluate our methods on three benchmark datasets: CIFAR-10 [19], CIFAR-100 [19], and ImageNet16-120[5]. CIFAR-10 and CIFAR-100 consist of 32×32 color images, representing 10 and 100 classes, respectively. Each dataset contains 60000 images split into 50000 training images and 10000 testing images. ImageNet16-120 is a downsized version of the ImageNet dataset [6] tailored for NAS evaluation, comprising 16×16 color images across 120 object classes.

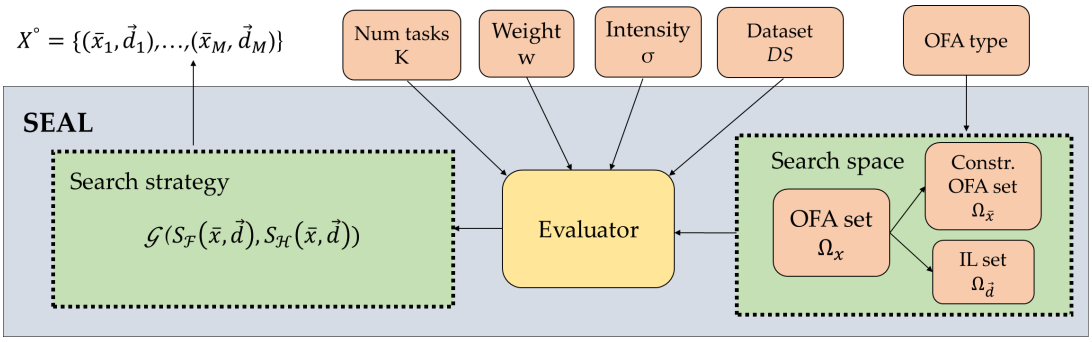

Figure 1. SEAL diagram of the main components. The output of the NAS is a candidate set of tuples, each containing a NN architecture and its corresponding expansion vector.

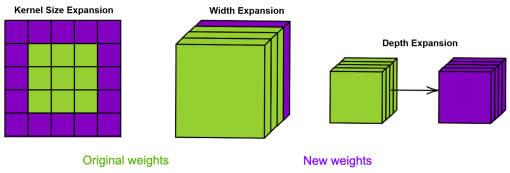

Figure 2. The model expansions performed by SEAL.

5.2 *Analysis of the results*

For all the experiments, the *MobileNetV3*-based [13] supernet is used and the hyperparameters of the NN architecture set $\Omega_{\bar{x}}$ consist of resolution, depth, width, and kernel size. The value of the threshold τ for expansion is 0.2 for CIFAR-10, and 0.13 fo the other datasets. Although the most relevant metric is the AA computed after the last sample set training, all the metrics are tracked across the training. Tables 1 and 2 report the average accuracy, average forgetting, and forward transfer values. The average incremental accuracy and the backward transfer are not presented as their values directly relate to the average accuracy and the average forgetting, respectively. Finally, to evaluate the models post-search, we aim at simulating a real-case scenario. As in incremental learning we are interested in the change in performance for the training samples. Instead of evaluating the model in a completely new dataset we augment the original training set with more samples, coming from data splits not available during the NAS. In this way, we are able to accurately measure how the model and expansion policy will perform in the future. These experiments were performed under a regime of K=5 sequential splits of disjoint samples, where the model was trained with 3 epochs per task. We emphasize that the model is expanded only if triggered by the condition presented in 4.3, checked on every new task instead of growing at any task to retain the model complexity.

Regarding the baselines, we followed the benchmarking done in [32] and selected models that suited our constraints and our data-incremental setting. Based on these criteria, we selected EWC [18], LWF [21], and SI [37]— being straightforward yet state-of-the-art methods across a range of continual learning settings. To strengthen the baselines, we included the *naive* and the *joint-training* benchmark cases, common baselines in IL. The *naive* method is a model trained in an incremental way without any additional constraint, it shows the performance degradation of a model just by being trained in a continual setting. On the other hand, the *joint-training* refers to a model trained on the whole dataset at once, as if it were a static dataset. SEAL has been executed with 5 different seeds on CIFAR-10 and CIFAR-100 and once on Imagenet16-120. All the benchmarks have been trained with 5 different seeds. In Table 1, we show our main result, i.e., that our SEAL models are competitive with the other baselines in every metric on each dataset. Notably, the *naive* approach sometimes achieves high accuracy—especially on CIFAR-10—when tasks are similar and do not require strong regularization. However, while SEAL outperforms *naive* in overall performance, it also shows increased forgetting and lower forward transfer on CIFAR-10, potentially indicating overfitting on the latest tasks. Finally, *joint-training* does not always yield the highest accuracy, likely due to task interference in heterogeneous datasets.

Table 1. Comparison between our searched NN architecture and expansion policy against standard usage of baseline methods

CIFAR-10@5			
Method	ACC	Forgetting	FWT
Naive	93.18 (±0.05)	1.41 (±0.50)	0.68 (±0.27)
Joint	94.85 (±0.15)	-	-
EWC	93.87 (±0.61)	1.13 (±0.30)	1.33 (±0.45)
SI	94.31 (±0.37)	0.71 (±0.43)	1.44 (±0.02)
LWF	92.49 (±2.16)	2.18 (±2.27)	1.22 (±0.84)
Ours	95.35 (±0.18)	1.76 (±0.41)	0.18 (±0.19)

CIFAR-100@5			
Method	ACC	Forgetting	FWT
Naive	81.32 (±0.85)	7.65 (±0.66)	-0.45 (±1.21)
Joint	84.71 (±0.65)	-	-
EWC	82.38 (±0.37)	6.94 (±0.40)	0.58 (±0.60)
SI	82.55 (±0.39)	5.89 (±0.48)	-0.53 (±0.75)
LWF	83.21 (±0.25)	5.50 (±0.14)	2.65 (±0.27)
Ours	83.94 (±0.33)	4.81 (±0.37)	1.59 (±0.25)

ImageNet-16@5			
Method	ACC	Forgetting	FWT
Naive	51.71 (±0.15)	5.54 (±0.35)	1.00 (±0.37)
Joint	51.84 (±1.30)	-	-
EWC	51.75 (±0.03)	5.50 (±0.65)	2.00 (±1.14)
SI	51.39 (±0.12)	5.73 (±0.44)	0.83 (±0.41)
LWF	53.02 (±0.10)	4.10 (±0.15)	3.27 (±0.24)
Ours	52.30 (±0)	3.35 (±0)	6.91 (±0)

An initial experiment compares performance between single-objective optimization using the mono-objective genetic algorithm Differential Evolution [30], and multi-objective optimization incorporating the flatness objective. Table 2 shows metrics for four models, each row aggregating the searched model from five NAS runs: the searched single-objective model (accuracy) and the searched multi-objective models by accuracy, trade-off, and flatness.

Table 2. Multi-objective vs single-objective NAS search results for a 5-splits data-incremental problem. Note: * denotes that the selected models coincide.

CIFAR-10@5					
Selection	Params	Flatness	ACC	Forgetting	FWT
SingleObj	3.98M ($\pm$0.63)	73.61 ($\pm$1.82)	94.82 ($\pm$0.57)	2.09 ($\pm$1.00)	0.55 ($\pm$0.74)
MultiObj-Acc	3.92M ($\pm$0.50)	75.05 ($\pm$1.14)	95.35 ($\pm$0.18)	1.76 ($\pm$0.41)	0.18 ($\pm$0.19)
MultiObj-Flat	3.79M ($\pm$0.40)	82.09 ($\pm$2.13)	94.53 ($\pm$0.51)	1.37 ($\pm$0.54)	0.75 ($\pm$0.34)
MultiObj-Trade	3.65M ($\pm$0.15)	78.19 ($\pm$3.33)	94.89 ($\pm$0.68)	1.89 ($\pm$0.56)	0.25 ($\pm$0.25)

CIFAR-100@5					
Selection	Params	Flatness	ACC	Forgetting	FWT
SingleObj	4.37M ($\pm$0.06)	52.57 ($\pm$1.84)	83.01 ($\pm$1.20)	5.72 ($\pm$1.28)	0.83 ($\pm$1.02)
MultiObj-Acc*	4.09M ($\pm$0.21)	57.88 ($\pm$3.48)	83.94 ($\pm$0.33)	4.81 ($\pm$0.37)	1.59 ($\pm$0.25)
MultiObj-Flat	3.85M ($\pm$0.02)	58.49 ($\pm$2.87)	83.61 ($\pm$0.01)	5.14 ($\pm$0.04)	1.12 ($\pm$0.22)
MultiObj-Trade*	4.09M ($\pm$0.21)	57.88 ($\pm$3.48)	83.94 ($\pm$0.33)	4.81 ($\pm$0.37)	1.59 ($\pm$0.25)

ImageNet-16@5					
Selection	Params	Flatness	ACC	Forgetting	FWT
SingleObj*	4.24M ($\pm$0.51)	45.42 ($\pm$1.82)	52.31 ($\pm$0.01)	3.35 ($\pm$0.28)	6.91 ($\pm$0.17)
MultiObj-Acc*	4.24M ($\pm$0.51)	45.42 ($\pm$1.82)	52.31 ($\pm$0.01)	3.35 ($\pm$0.28)	6.91 ($\pm$0.17)
MultiObj-Flat	3.55M ($\pm$0.06)	56.16($\pm$1.89)	50.40 ($\pm$0.20)	2.29($\pm$0.22)	4.21 ($\pm$0.34)
MultiObj-Trade	4.06M ($\pm$0.36)	49.98 ($\pm$2.59)	51.51 ($\pm$0.29)	2.24 ($\pm$0.26)	4.33($\pm$0.25)

The results validate the superiority of the multi-objective problem and the use of flatness for incremental learning. An additional insight is how increasing the flatness seems to correlate with smaller models. This may suggest that flatten-minima models may require fewer parameters to fully learn the problem, an important insight for constrained-resources environments were NAS are commonly used.

Additionally, we show in Fig. 3 the performance differences between initializing the new sections of the model from random weights, without using the OFA set, and initializing the weights from the OFA. It is clear that initializing from the OFA set has a beneficial effect over the model accuracy and forgetting, although it is relevant to notice that this effects seems to vanish with increases in data complexity.

Furthermore, we validate the usage of the cross-distillation loss for training the expanded models. From Fig. 4, there is a clear performance increase by using the distillation approach. Not only the accuracies achieve higher values, but the forgetting and knowledge transfer metrics also improve.

The last experiment evaluates whether to expand the early or final layers. Since our data-incremental approach does not drastically alter dataset distributions, expanding the last layers better preserves fine-grained features, as confirmed in Fig. 5. Differently, early expansion yields a lower accuracy and higher forgetting, and, intriguingly, the forward transfer suggests that early layers excel at compressing similar samples into joint representations.

6 Conclusions

The aim of this paper was to propose SEAL, the first NAS framework able to optimize jointly the architecture and the expansion policy for a DNN. Our results show that the learned policies effectively reduce forgetting, improve the model performance, and reduce the memory complexity compared to previous methods. SEAL has been designed for a data-incremental setting but could be extended to many other relevant IL settings. Another promising direction lies in enhancing the role of NAS within the model growing policy. In this work, certain decisions were treated as fixed hyperparameters based on the assumption that architectures would behave similarly across the splits of our data-incremental setup. However, this assumption may not hold in domain-incremental or class-incremental scenarios. A possible improvement would be to incorporate a capacity-related control signal (e.g., a threshold-bit) directly into the NAS process, allowing the search to determine when and how the model should expand dynamically.

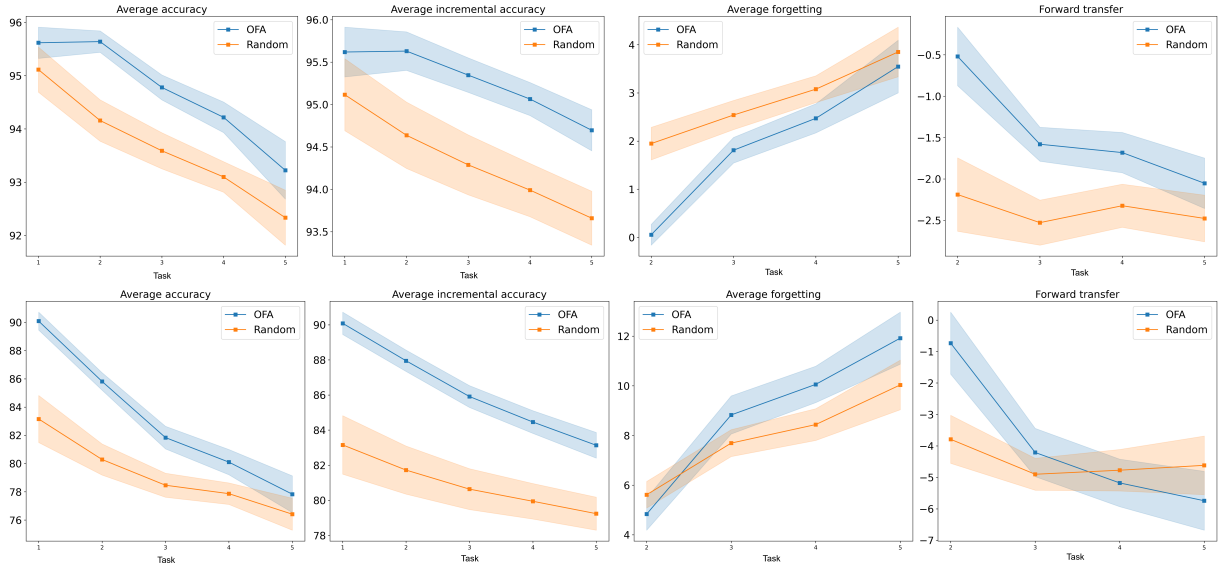

Figure 3. Performance distribution for our IL problem. Expansion starts from random weights vs OFA-initialized weights on the CIFAR-10 and CIFAR-100 datasets.

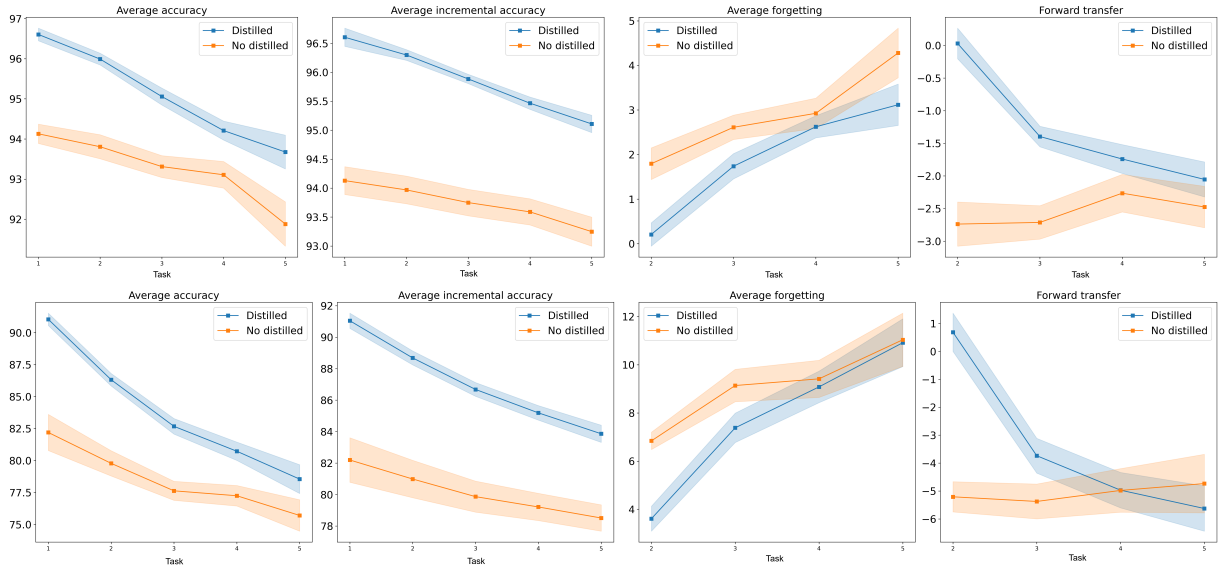

Figure 4. Performance distribution when models are trained with the distillation loss on CIFAR-10 and CIFAR-100 datasets.

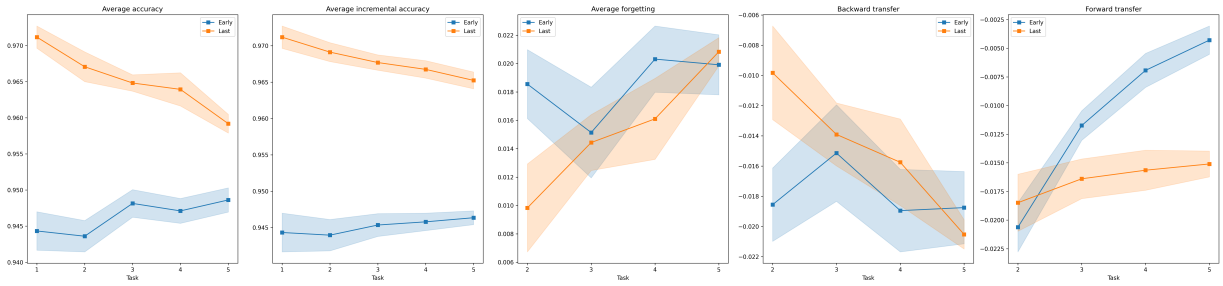

Figure 5. Performance change when expanding the early layers vs the last layers of the model.

References

[1] D. Ashlock. *Evolutionary Computation for Modeling and Optimization*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 1441919694.

[2] T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., GBR, 1st edition, 1997. ISBN 0750303921.

[3] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once-for-All: Train One Network and Specialize it for Efficient Deployment, Apr. 2020. arXiv:1908.09791 [cs, stat].

[4] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[5] P. Chrabaszcz, I. Loshchilov, and F. Hutter. A downsampled variant of imagenet as an alternative to the CIFAR datasets. *CoRR*, abs/1707.08819, 2017. URL http://arxiv.org/abs/1707.08819.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[7] S. Dohare, J. Hernandez-Garcia, Q. Lan, P. Rahman, A. Mahmood, and R. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632: 768–774, 08 2024. doi: 10.1038/s41586-024-07711-7.

[8] 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)*, pages 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9534079.

[9] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization, 2021. URL https://arxiv.org/abs/2010.01412.

[10] M. Gambella, A. Falcetta, and M. Roveri. Cnas: Constrained neural architecture search. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2918–2923, 2022. doi: 10.1109/SMC53654.2022.9945080.

[11] M. Gambella, F. Pittorino, and M. Roveri. Flatnas: optimizing flatness in neural architecture search for out-of-distribution robustness. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2024. doi: 10.1109/IJCNN60899.2024.10650433.

[12] Q. Gao, Z. Luo, D. Klabjan, and F. Zhang. Efficient architecture search for continual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8555–8565, 2023. doi: 10.1109/TNNLS.2022.3151511.

[13] A. G. Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[14] J. Jeong, J. Yu, G. Park, D. Han, and Y. Yoo. Genas: Neural architecture search with better generalization. In E. Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 911–919. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/101. URL https://doi.org/10.24963/ijcai.2023/101. Main Track.

[15] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio. Fantastic generalization measures and where to find them. 2019. URL https://arxiv.org/abs/1912.02178.

[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey, 1996.

[17] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30 (1/2):81–93, 1938. ISSN 00063444. URL http://www.jstor.org/stable/2332226.

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114, 12 2016. doi: 10.1073/pnas.1611835114.

[19] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, April 2009.

[20] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3925–3934. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/li19m.html.

[21] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[22] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable Architecture Search, Apr. 2019. arXiv:1806.09055 [cs, stat].

[23] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6470–6479, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[24] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti. NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision (ECCV)*, 2020.

[25] C. Lyle, Z. Zheng, E. Nikishin, B. A. Pires, R. Pascanu, and W. Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pages 23190–23211. PMLR, 2023.

[26] J. Mok, B. Na, H. Choe, and S. Yoon. AdvRush: Searching for Adversarially Robust Neural Architectures, Aug. 2021. arXiv:2108.01289 [cs].

[27] 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 M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8184–8194. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/niu21a.html.

[28] M. Pavone, A. Lacava, and D. Brunelli. Tybox: An automatic design and code generation toolbox for incremental on-device tinyml models. In *Proceedings of the 2024 ACM SIGPLAN International Conference on Generative Programming: Concepts & Experiences (GPCE '24)*. ACM, 2024. doi: 10.1145/3604566. URL https://doi.org/10.1145/3604566.

[29] F. Pittorino, C. Lucibello, C. Feinauer, G. Perugini, C. Baldassi, E. Demyanenko, and R. Zecchina. Entropic gradient descent algorithms and wide flat minima. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xjXg0bnoDmS.

[30] K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 3540209506.

[31] M. Shahawy, E. Benkhelifa, and D. White. Exploring the intersection between neural architecture search and continual learning. *IEEE transactions on neural networks and learning systems*, PP, 09 2024. doi: 10.1109/TNNLS.2024.3453973.

[32] G. van de Ven, T. Tuytelaars, and A. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4:1–13, 12 2022. doi: 10.1038/s42256-022-00568-3.

[33] L. Wang, X. Zhang, H. Su, and J. Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024. doi: 10.1109/TPAMI.2024.3367329.

[34] X. Wang, S. Cao, M. Li, and K. M. Kitani. Neighborhood-Aware Neural Architecture Search, Oct. 2021. arXiv:2105.06369 [cs].

[35] M. Wistuba, A. Rawat, and T. Pedapati. A Survey on Neural Architecture Search, June 2019. arXiv:1905.01392 [cs, stat].

[36] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[37] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.