Incremental Learning with Differentiable Architecture and Forgetting Search

James Seale Smith  
Georgia Institute of Technology  
Atlanta, Georgia, USA  
jamessealesmith@gatech.edu

Zachary Seymour  
SRI International  
Princeton, NJ, USA  
yzachary.seymour@sri.com

Han-Pang Chiu  
SRI International  
Princeton, NJ, USA  
han-pang.chiu@sri.com

Abstract—As progress is made on training machine learning models on incrementally expanding classification tasks (i.e., incremental learning), a next step is to translate this progress to industry expectations. One technique missing from incremental learning is automatic architecture design via Neural Architecture Search (NAS). In this paper, we show that leveraging NAS for incremental learning results in strong performance gains for classification tasks. Specifically, we contribute the following: first, we create a strong baseline approach for incremental learning based on Differentiable Architecture Search (DARTS) and state-of-the-art incremental learning strategies, outperforming many existing strategies trained with similar-sized popular architectures; second, we extend the idea of architecture search to regularize architecture forgetting, boosting performance past our proposed baseline. We evaluate our method on both RF signal and image classification tasks, and demonstrate we can achieve up to a 10% performance increase over state-of-the-art methods. Most importantly, our contribution enables learning from continuous distributions on real-world application data for which the complexity of the data distribution is unknown, or the modality less explored (such as RF signal classification).

Index Terms—continual learning, incremental learning, neural architecture search

I. INTRODUCTION

Machine learning applications built on deep neural networks are typically trained offline on a large dataset. The model is frozen and deployed, and it assumes the distribution of data it will see is of the same distribution of the training dataset. Unfortunately, this is not the case for many real-world applications. The distribution of seen data may have a domain shift, leading to prediction errors from the model.

This domain shift may come in the form of the pixel space (e.g., different weather conditions affecting background and lighting) or the semantic space (e.g., we encounter novel objects which were not seen in the training data). These shifts are typically not a single instance; often, the distribution of data continuously shifts over the lifetime of deployed application, meaning we must continuously update the model in response to these shifts. In this paper, we focus on continuous domain shifts in the semantic space, a paradigm commonly referred to as class-incremental continual learning, or incremental learning for short.

Models designed for the incremental learning setting should be able to incorporate new information about novel classes seen in a continuously shifting datastream, while simultaneously avoiding catastrophic forgetting of previously seen classes which may disappear from the stream for extended periods of time [1]. Specifically, the broad literature in typical incremental learning asks us: “how can we acquire new semantic classes without forgetting previously seen classes?”

Another concurrent challenge in machine learning applications is selecting an appropriate neural architecture for data of unknown complexities or less-explored modality. For example, while the commonly used ResNet architecture [2] achieves high performance on image datasets such as ImageNet [3], we can often achieve higher performance at a lower cost in inference time and model parameters by using a neural architecture selected or learned in an automated fashion. Additionally, architectures have been less explored for modalities such as RF signal classification [4], giving us fewer pre-existing options to explore. We have motivated and described the paradigm of Neural Architecture Search (NAS), a broad research area which pushes to automatically find neural architectures to boost model performance for real-world deep learning applications.

Visualized in Figure 1, we unify the two crucial concepts of incremental learning and neural architecture search to design a flexible incremental learning approach which can be applied to semantically-shifting data of unknown underlying complexity. This could be important for applications such as on-device user-specific learning (where the complexity of user actions may be unknown), exploratory agents relying on non-pixel image perception (where novel objects are expected), or animal motion detection with (where it may be cheap to rely on dated, pre-existing sensors for which finding a neural architecture may be difficult).

Our high level approach is to first integrate state-of-the-art incremental learning strategies (including replay, knowledge distillation regularization, class-balanced fine-tuning, etc.) [5] into a popular and efficient NAS method, Differentiable Architecture Search (DARTS). This required a combination of high-level design and numerous empirical analyses, and resulted in up to a 10% increase in final accuracy compared to the state-of-the-art. We took our work one step forward, using analysis of the inferred architecture at different tasks to motivate a regularization penalty on the architecture decisions themselves, resulting in additional performance gains. In summary, we make the following contributions:
1) We unify the concepts of *incremental learning* and *neural architecture search* to propose learning from continuous distributions of data from less explored modalities or unknown underlying complexity.

2) We propose Incremental-DARTS (I-DARTS), which combines SOTA incremental learning strategies with the DARTS method to increase incremental classification accuracy by as much as 10%.

3) We use analysis of the I-DARTS architecture over tasks to motivate an architecture decision regularization, resulting in additional performance gains.

II. BACKGROUND AND RELATED WORK

**Catastrophic Forgetting:** Methods to mitigate catastrophic forgetting can broadly be separated into either (i) architecture expansion or (ii) regularization. Architecture expansion [6]–[10] is useful for applications in which the architecture can grow in parameters with the number of tasks. However, these are typically associated with *task-incremental learning* rather than *class-incremental learning*, an important nuance (discussed in the next section) which cannot be compared to our setting. Regularization approaches mitigate forgetting by regularizing the model with respect to past task knowledge while training on a new task. This can be done by either regularizing the model in the weight space (i.e., penalize changes to model parameters) [11]–[15] or the prediction space (i.e., penalize changes to model predictions) [5], [16]–[18].

**Neural Architecture Search:** Orthogonal to the catastrophic forgetting problem is concept of Neural Architecture Search (NAS). This type of work automates the process of architecture selection, replacing the expensive (with respect to time and costly errors) engineering of neural architectures with an optimization problem. Because incremental learning (the main focus of this paper) typically associates a cost to training time, our work is only concerned with efficient, one-shot NAS approaches. For more details on other types of NAS, we refer the reader to recent literature surveys [19]–[21].

The high level idea of one-shot NAS is to model the space of candidate architectures with a single, large model, and then search for an optimal sub-model within the large “super-model”. While several works [22]–[25] fit under this constraint, we selected the DARTS method [26] because (i) its two-step differentiable algorithm fits nicely with existing incremental learning concepts, and (ii) there exists a user-friendly implementation [27], allowing us to focus our work on the incremental learning contributions.

**Continuous Neural Architecture Search:** Our work is not the first to consider concepts of neural architecture search in a continuous learning setting. Firefly neural architecture search [28] continuously grows a model for continual learning by searching for the optimal growing method in each task (split existing neurons, grow new neurons, or grow new layers). Learn-to-Grow [29] and NSAS [30] also formulates continuous neural architecture search from a parameter reuse perspective. However, these works focus on the *task-incremental learning*, which cannot be compared with our work on *class-incremental learning*. Continually Neural Architecture Search (CNAS) [31] is the first work to propose neural architecture search for class-incremental learning. However, this work assumes access to past task data, with its contributions focusing on the efficient expansion of an architecture rather than considering important continual learning constraints (and we therefore cannot compare to this method). Finally, recent
work on neural architecture search of deep priors [32] has invoking findings on learning a deep prior on which to train a linear classifier without forgetting, but the problem setting assumes training on classes from a similar distribution from the target distribution (which we strongly do not assume, given our work is designed specifically for the case of unknown data complexities). Importantly, none of the discussed works unify the typical incremental learning setting with NAS, a key contribution of our work.

III. PRELIMINARIES

In incremental learning, we show a model labeled data corresponding to $M$ semantic object classes $c_1, c_2, \ldots, c_M$ over a series of $N$ tasks corresponding to non-overlapping subsets of classes. We use the notation $T_n$ to denote the set of classes introduced in task $n$, with $|T_n|$ denoting the number of object classes in task $n$. To describe our inference model, we denote $\theta_{i,n}$ as the model $\theta$ at time $k$ that has been trained with the classes from task $n$. For example, $\theta_{1,1}$ refers to the model trained during task 1 and its logits associated with all tasks up to and including class 1. We drop the second index when describing the model trained during task $n$ with all logits (for example, $\theta_n$).

In this setting, each class appears in only a single task, and the goal is to incrementally learn to classify new object classes as they are introduced while retaining performance on previously learned classes. The incremental learning setting is challenging because no task indexes are provided to the learner during inference and the learner must support classification across all classes seen up to task $n$ [33]. Incremental learning, sometimes referred to as class-incremental learning, is more difficult than task-incremental learning, where the task indexes are given during both training and inference (which is an entirely different learning setting).

A key component of the incremental learning setting discussed in this work is that competitive methods must sample a small coreset of training data from past tasks [34]–[46]. We denote this coreset as $X_{core}$, and it will be replayed in future tasks $T_n$ along with the corresponding training data, denoted as $X_n$.

IV. A STRONG BASELINE FOR INCREMENTAL LEARNING

In this section, we first review the pertinent details of the neural architecture approach, DARTS. As discussed in the previous section, we chose to build our method using the DARTS approach due to its differentiability and ease of implementation. We then describe how we integrate incremental learning strategies into DARTS to create our Incremental-DARTS baseline approach.

A. Differentiable Architecture Search (DARTS)

The goal of DARTS [26] is to find an optimal neural architecture given a “super-model” of candidate operations. Specifically, we consider a set of candidate operations $O$ (e.g., convolution, pooling, identity, zero) as our super-model. These operations can be considered as nodes in a directed acyclic graph. At each node, we refer to the intermediate representation as $x^{(i)}$, with each directed edge $(i, j)$ being associated with some operation $o^{(i,j)}$ from $O$. The goal of DARTS is select which operations $o^{(i,j)}$ at each $(i, j)$ should be used for optimal model performance.

In DARTS, we model the operation at each node as a mixture of the candidate operations at that node. That is, $x^{(i)}$ is transformed in the super-model at each $(i, j)$ by a weighted sum of each $o^{(i,j)}$ transformation. The mixture weights at node $(i, j)$ are parameterized with a vector $\alpha^{(i,j)}$ of dimension $|O|$. With this formulation, we can now solve a bilevel optimization which iterates optimizing architecture weights $w$ (which parameterize the candidate operations) with respect to training data and the mixture weights $\alpha$ (which parameterize the weighting of the candidate operations) with respect to holdout data. Formally, we have:

$$
\min_{\alpha} \mathcal{L}_{val}(w^{*}(\alpha), \alpha) \\
\text{s.t.} \quad w^{*}(\alpha) = \arg\min_{w} \mathcal{L}_{train}(w, \alpha) \tag{1}
$$

At the end of this training, we can then infer a discrete architecture using the argmax of the $\alpha$ (i.e., only the $o^{(i,j)}$ at each $(i, j)$ with the highest corresponding $\alpha^{(i,j)}$ is retained). Finally, we retrain the inferred architecture using the combined training and holdout data.

To be consistent with our notation described in the previous section, we will simply refer to model optimization in the rest of this paper as optimizing with respect to $\theta_n$. If we optimize with respect to $\theta_n$ during the bilevel optimization stage of our approach, we will specifically update both architecture weights $w$ and mixture weights $\alpha$. Otherwise, we will only be updating architecture weights $w$ of the inferred neural architecture.

B. Incremental DARTS (I-DARTS)

An intuitive approach for an incremental variant of DARTS might be to simply run the DARTS algorithm including a coreset of exemplar data for replay. However, this would not take advantage of the numerous advancements in incremental learning which far exceed simple replay data. Here, we propose a strong incremental learning variant of DARTS, referred to as I-DARTS.

We first address catastrophic forgetting with a prediction space regularization, which encourages the model to learn the classes of task $T_n$ without unlearning the representation of classes of tasks $T_1 \cdots T_{n-1}$. We observe here that we incorporate prediction space regularization over model space regularization because the former has been found to perform better for class-incremental learning [47], [48].

Specifically, in task $T_n$ we include a knowledge distillation (KD) loss [18] over all data $X_{core} \cup X_n$, which forces $\theta_n$ to learn $T_n$ with minimal degradation to $T_1 \cdots T_{n-1}$ knowledge. Denoting $p_{\theta_n}(x)$ as the predicted class distribution produced by model $\theta_n$ for some input $x$, we replace the loss $\mathcal{L}$ in Eq.(1) to instead minimize:

$$
\min_{\theta_n} \mathcal{L}_{CE}(p_{\theta_{n,1:n}}(x), y) + \mu \mathcal{L}_{KD}(x, \theta_n, \theta_{n-1}) \tag{2}
$$
between (a) and (b) in Figure 3 is alarming.

Because we can consider the model after task 1 to be a typical offline DARTS solution, the sharp distinction after task 1 is of a very distinctly different distribution compared to the CIFAR-100 benchmark, the distribution of \( \alpha \) after training task 2 is much closer to the distribution after task 1. We are now more confident that our super-model is learning stable \( \alpha \) over incremental tasks, which is reflected in our final results. We refer to I-DARTS with this regularization as I-DARTS*, with the following formal objective:

\[
\min_{\theta_n} L_{CE}(p_{y_{n-1:n-1}}(x), y) + \mu L_{KD}(x, \theta_n, \theta_n) + \lambda L_{reg,\alpha}(\theta_n)
\]

where \( \lambda \) is a scalar hyperparameter which weights the contribution of \( L_{reg,\alpha} \).

V. Alpha Analysis and Regularization

When analyzing the distribution of the \( \alpha_{1:n-1:n-1} \) in our I-DARTS method, we made an interesting discovery. We found the \( \alpha_{1:n-1:n-1} \) tend to increase after each task, which prevents useful learning in future tasks if the \( \alpha_{1:n-1:n-1} \) are already converged to high values. We visualize this in Figure 3, showing that when training I-DARTS on two tasks of the ten task incremental CIFAR-100 benchmark, the distribution of \( \alpha_{1:n-1:n-1} \) after training task 2 is of a very distinctly different distribution compared to after task 1. Because we can consider the model after task 1 to be a typical offline DARTS solution, the sharp distinction between (a) and (b) in Figure 3 is alarming.

Using this finding, we propose to add an additional regularization to our learning objective to reduce this behavior in the \( \alpha_{1:n-1:n-1} \) distribution. Our intuition is that \( \alpha_{1:n-1:n-1} \) should be able to update in the future incremental learning tasks, and we therefore need to prevent over-fitting in the current task. We realize this intuition by adding an L2 distance regularization penalty on the \( \alpha_{1:n-1:n-1} \). Specifically, we add the following regularization penalty:

\[
L_{reg,\alpha}(\theta_n) = \sum_{i,j \in \theta_n} (\alpha_{i:n-1:n-1} - \mu)^2
\]

With this added loss function, we see in (c) and (d) of Figure 3 that the distribution of \( \alpha_{1:n-1:n-1} \) after task 2 is much closer to the distribution after task 1. We are now more confident that our super-model is learning stable \( \alpha_{1:n-1:n-1} \) over incremental tasks, which is reflected in our final results. We refer to I-DARTS with this regularization as I-DARTS*, with the following formal objective:

\[
\min_{\theta_n} L_{CE}(p_{y_{n-1:n-1}}(x), y) + \mu L_{KD}(x, \theta_n, \theta_n) + \lambda L_{reg,\alpha}(\theta_n)
\]

where \( \lambda \) is a scalar hyperparameter which weights the contribution of \( L_{reg,\alpha} \).

VI. Experiments

We evaluate I-DARTS and I-DARTS* for an RF signal incremental classification benchmark and two incremental image classification benchmarks. We compare to several recent state-of-the-art approaches: elastic-weight consolidation (EWC) [13], learning without forgetting (LwF) [18], and end-to-end incremental learning (E2E) [5]. We also compare to a
model trained with no incremental learning strategy (denoted as “naive”) and simple data replay (denoted as “replay”).

We emphasize that the impact of this work is not to incrementally push the state-of-the-art performance for a well-studied problem (such as incremental image classification), but rather to demonstrate that we can incrementally learn in applications where we must also simultaneously learn our architecture. Thus, we benchmark with the RF classification dataset to demonstrate the large performance gains that can be achieved on understudied modalities or datasets. We then benchmark with common image classification datasets to demonstrate that our method is comparable to state-of-the-art methods which use a highly developed architecture.

Candidate Operations: Following the original DARTS work, we include the following candidate operations in $O$: $3 \times 3$ and $5 \times 5$ separable convolutions, $3 \times 3$ and $5 \times 5$ dilated separable convolutions, $3 \times 3$ max pooling, $3 \times 3$ average pooling, identity, and zero. For the RF signal classification dataset, we replace the 2d convolutions in the DARTS operations with 1d convolution operations.

Evaluation Metrics: Following prior works, we evaluate methods in the class-incremental learning setting using: (I) final performance, or the performance with respect to all past classes after having seen all $N$ tasks (referred to as $A_{N,1:N}$); (II) maximum model parameters, or the maximum model parameters at any given time (important for DARTS); (III) final model parameters, or the model parameters of the final inferred architecture; and (IV) the training time given in GPU days. We use index $k$ to index tasks through time and index $n$ to index tasks with respect to test/validation data (for example, $A_{k,n}$ describes the accuracy of our model after task $k$ on task $n$ data). Specifically:

$$A_{k,n} = \frac{1}{|D_{test}|} \sum_{(x,y) \in D_{test}^{k,n}} 1(\hat{y}(x, \theta_{k,n}) = y \mid \hat{y} \in T_n) \quad (6)$$

For the final task accuracy in our results, we will denote $A_{N,1:N}$ as simply $A_N$.

Additional Experiment Details: We augment image training data using standard augmentations such as random horizontal flips and crops. Our results were generated using 2080 Ti GPUs. In implementing our experiments, we leveraged both the Microsoft nni [27] package for DARTS [26] and the Avalanche [50] continual learning library for incremental learning benchmarks.

We tuned hyperparameters using a grid search. The hyperparameters were tuned using k-fold cross validation with three folds of the training data on only half of the tasks. We do not tune hyperparameters on the full task set because tuning hyperparameters with hold out data from all tasks may violate the principle of continual learning that states each task in visited only once [51]. The results reported are on testing splits (defined in the dataset).

A. I-DARTS is SOTA for RF signal classification

Our first benchmark is four-task incremental learning on the RADIOML 2018.01a DEEPSIG dataset [4] for RF signal classification. The DEEPSIG dataset includes 2 million examples, each 1024 samples long, which have been modulated by 24 digital and analog modulation types, and the classification task is to predict which of the 24 modulation types was used for each signal. This benchmark exemplifies the key contribution of our work because there is no clear neural architecture to apply on this dataset.

We borrowed the RF ResNet [2] network introduced in the original DEEPSIG work [4]. This model contains 6 Residual stacks followed by three fully connected layers. We also tested the second architecture listed in the work, but do not report results given they far under-perform the ResNet variant. For the RADIOML 2018.01a DEEPSIG dataset [4], we created
| Method          | Replay | $A_N$ (%,$\uparrow$) | Max Model Params (↓) | Final Model Params (↓) | GPU Days (↓) |
|-----------------|--------|-----------------------|-----------------------|------------------------|--------------|
| I-DARTS         | 1e3    | 56.3 ± 0.0            | 2.5e5                 | 5.7e5 ± 4.6e3          | 3.0          |
| I-DARTS*        | 1e3    | 57.3 ± 0.2            | 2.5e5                 | 5.8e4 ± 4.7e3          | 3.0          |
| Remove Class Balancing | 1e3 | 53.2 ± 0.8            | 2.5e5                 | 5.3e4 ± 2.7e2          | 3.0          |
| Remove All (DARTS) | 1e3 | 55.0 ± 0.0            | 2.5e5                 | 5.6e4 ± 2.0e3          | 2.5          |

TABLE II: Results for class-incremental learning on four-task Deepsig. Results are reported as an average of 2 runs. A coset of 1000 samples is leveraged for each method.

| Method          | Replay | $A_N$ (%,$\uparrow$) | Max Model Params (↓) | Final Model Params (↓) | GPU Days (↓) |
|-----------------|--------|-----------------------|-----------------------|------------------------|--------------|
| Naive           | 0      | 18.5 ± 0.5            | 1.7e5                 | 1.7e5                  | 4.6e – 1     |
| EWC             | 0      | 18.3 ± 0.0            | 1.7e5                 | 1.7e5                  | 4.9e – 1     |
| LWF             | 0      | 24.4 ± 1.6            | 1.7e5                 | 1.7e5                  | 4.9e – 1     |
| Replay          | 1e3    | 44.1 ± 0.7            | 1.7e5                 | 1.7e5                  | 4.7e – 1     |
| E2E             | 1e3    | 47.3 ± 0.6            | 1.7e5                 | 1.7e5                  | 5.4e – 1     |
| DARTS           | 1e3    | 55.0 ± 0.0            | 5.6e4 ± 2.0e3         | 2.5                    |
| I-DARTS         | 1e3    | 56.3 ± 0.0            | 5.7e4 ± 4.6e3         | 3.0                    |
| I-DARTS*        | 1e3    | 57.3 ± 0.2            | 5.8e4 ± 4.7e3         | 3.0                    |

TABLE III: Ablation results for class-incremental learning on ten-task CIFAR-100. Results are reported as an average of 2 runs. A coset of 2000 images is leveraged for each method.

| Method          | Replay | $A_N$ (%,$\uparrow$) | Max Model Params (↓) | Final Model Params (↓) | GPU Days (↓) |
|-----------------|--------|-----------------------|-----------------------|------------------------|--------------|
| I-DARTS         | 2e3    | 37.2 ± 0.1            | 2.0e6                 | 5.3e5 ± 1.4e4          | 6.4          |
| I-DARTS*        | 2e3    | 37.7 ± 0.1            | 2.0e6                 | 4.8e5 ± 1.3e4          | 6.6          |
| Remove Class Balancing | 2e3 | 36.7 ± 0.4            | 2.0e6                 | 5.0e5 ± 5.7e4          | 5.1          |
| Remove KD       | 2e3    | 29.8 ± 6.1            | 2.0e6                 | 4.5e5 ± 9.0e3          | 5.1          |
| Remove All (DARTS) | 2e3 | 33.4 ± 0.9            | 2.0e6                 | 4.5e5 ± 1.3e4          | 3.9          |

TABLE IV: Results for class-incremental learning on five-task CIFAR-10. Results are reported as an average of 2 runs. A coset of 1000 images is leveraged for each replay-based method.

| Method          | Replay | $A_N$ (%,$\uparrow$) | Max Model Params (↓) | Final Model Params (↓) | GPU Days (↓) |
|-----------------|--------|-----------------------|-----------------------|------------------------|--------------|
| Naive           | 0      | 19.7 ± 0.0            | 2.7e5                 | 2.7e5                  | 3.6e – 1     |
| EWC             | 0      | 19.6 ± 0.0            | 2.7e5                 | 2.7e5                  | 4.5e – 1     |
| LWF             | 0      | 22.0 ± 0.5            | 2.7e5                 | 2.7e5                  | 4.2e – 1     |
| Replay          | 1e3    | 60.8 ± 0.8            | 2.7e5                 | 2.7e5                  | 3.9e – 1     |
| E2E             | 1e3    | 73.5 ± 0.8            | 2.7e5                 | 2.7e5                  | 5.0e – 1     |
| DARTS           | 1e3    | 64.6 ± 0.1            | 1.95e6                | 3.1e5 ± 3.1e3          | 3.9          |
| I-DARTS         | 1e3    | 71.3 ± 0.1            | 1.95e6                | 3.8e5 ± 3.2e3          | 4.9          |
| I-DARTS*        | 1e3    | 70.6 ± 5.0            | 1.95e6                | 4.0e5 ± 3.5e3          | 4.3          |

TABLE V: Results for class-incremental learning on ten-task CIFAR-100. Results are reported as an average of 2 runs. A coset of 2000 images is leveraged for each replay-based method.
incremental learning tasks by allocating modulation types such that similar modulation types appeared in the same task. The motivation behind this is to make the tasks more “realistic”. We reemphasize that this benchmark exemplifies the key contribution of our work because there is no clear neural architecture to apply on this dataset.

We train our super-model using the DARTS bi-level optimization for 50 epochs with the Adam optimizer and a learning rate of 0.05 for the architecture weights and 5e-3 for the \( \alpha \) weights. For the second training stage, we train for 125 epochs. The learning rate is set to 0.05 and is reduced by 10 after 50, 75, and 100 epochs. We use a weight decay of 0.0002 and batch size of 128. Using a simple grid search to find hyperparameters for Eq (5), we found \( \mu \) to be 0.5 and \( \lambda \) to be 1e-3.

We first compare I-DARTS and I-DARTS* to regular DARTS with stored replay data, as well as two additional ablations on our method, in Table I. We see that each of our contributions incrementally improves the DARTS performance without sacrificing the number of model parameters or training time in GPU days.

In Table II, we compare out approach to several popular incremental learning methods (listed in the beginning of this section). We implemented all of these methods with the two different neural architectures from the original paper [4], and only report the higher performing architecture, which is build on Residual blocks [2]. We show here that our approach makes substantial improvements in accuracy over the existing methods, while having both a similar maximum number of architecture parameters and a smaller number of final model parameters. We observe that this performance comes at a cost in GPU days, but we argue this is less important given that the significant engineering time likely went into designing the baseline model, which is not accounted for in our metrics.

### B. I-DARTS is near SOTA for image classification

Next, we benchmark our method using the CIFAR-10 and CIFAR-100 datasets [52] class-incremental learning benchmarks. These datasets contains 10/100 classes of 32x32x3 images, respectively. We only benchmark image classification on the two CIFAR benchmarks (and not the ImageNet benchmarks) because DARTS on Imagenet [3] is beyond our computational limits. Following prior work [53], we train with a 32-layer ResNet [2]. We use the same training protocol from the RF classification experiments.

Similar to the RF classification experiments, we first compare I-DARTS and I-DARTS* to regular DARTS with stored replay data, as well as two additional ablations on our method (given in Table III). We again see that each of our contributions incrementally improves the DARTS performance without sacrificing the number of model parameters or training time in GPU days.

We see a different story in the results from Tables IV/V, which compares performance to the same incremental learning methods. We actually find that E2E [5] achieves the highest performance, with I-DARTS* closely behind. Why does our method not improve CIFAR image classification? We conjecture that there is actually a simple and intuitive explanation here. The ResNet [2] architecture has been profoundly studied for this dataset, with the optimal architecture settings evolving over years of research. In fact, most of the incremental learning we list here were developed specifically for this architecture. Keeping this in mind, it makes sense that there is little to none performance gap to be closed with our incremental DARTS methods. Rather, the story here is that our contributions improve DARTS in the incremental setting for this image classification dataset.

### VII. Conclusions

We unify two concepts crucial for machine learning applications: incremental learning and neural architecture search. Our motivation is to build reliable models for applications which undergo continuous domain shifts in training data after deployment, while being flexible to handle data of less explored modalities or unknown underlying complexity. We start by proposing a baseline method, I-DARTS, which unifies concepts from (i) the one-shot NAS approach DARTS and (ii) state-of-the-art incremental learning methods. We analyze the performance of I-DARTS and use this to motivate regularization on the DARTS mixture weights, further boosting our performance. We demonstrate that I-DARTS achieves high performance gains on an RF classification task in which the existing neural architecture has not been highly engineered, while additionally demonstrating that we are highly competitive on state-of-the-art image classification tasks. The main takeaway from our work is that our method can successfully learn both a neural architecture and architecture weights in simultaneous fashion, enabling new real-world application for machine learning models.

### Acknowledgements

This material is based upon work supported by DARPA under Contract No. N6523620C8020. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US government/Department of Defense.

### References

[1] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, “An empirical investigation of catastrophic forgetting in gradient-based neural networks,” arXiv preprint arXiv:1312.6211, 2013.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.

[4] T. J. O’Shea, T. Roy, and T. C. Clancy, “Over-the-air deep learning based radio signal classification,” IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 168–179, 2018.

[5] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, “End-to-end incremental learning,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 233–248.

[6] S. Ebrahimi, F. Meier, R. Calandra, T. Darrell, and M. Rohrbach, “Adversarial continual learning,” arXiv preprint arXiv:2003.06553, 2020.
