Learn to Bind and Grow Neural Structures

Anonymous Author(s)
Affiliation
Address
email

Abstract

Task-incremental learning involves the challenging problem of learning new tasks continually, without forgetting past knowledge. Many approaches address the problem by expanding the structure of a shared neural network as tasks arrive, but struggle to grow optimally without losing past knowledge. We present a new framework, Learn to Bind and Grow, which learns a neural architecture for a new task incrementally, either by binding with layers of a similar task or by expanding layers which are more likely to conflict between tasks. Central to our approach is a novel, interpretable parameterization of the shared, multi-task architecture space, which then enables computing globally optimal architectures using Bayesian optimization. Experiments on several continual learning benchmarks show that our framework discovers architectures which facilitate positive forward transfer and mitigate negative backward transfer, and outperforms existing network expansion based approaches to continual learning.

1 Introduction

A key challenge for efficient learning systems is to learn new information without forgetting information gained previously. Solving this challenge is crucial for deep neural networks (DNNs), which seek to continue learning new tasks without *catastrophically* forgetting old ones [28, 29, 30, 20]. Because the participating tasks are varied and may have conflicting objectives, we can view task-incremental learning as a complex, multi-objective optimization problem [19]. Broadly, two main approaches have been proposed to address this problem. The first set of methods impose some form of regularization on a fixed capacity network, by restricting changes to important parameters of each task [7, 27, 9] or using constrained experience replay [12, 15]. The first set of methods impose some form of regularization on a fixed capacity network, by restricting changes to important parameters of each task [7, 27, 9], using constrained experience replay [12, 15] or distillation [18]. The second set of methods tackle forgetting by dynamic network expansion, i.e., adding additional task-specific parameters when a new task arrives. The challenge here is to minimize the expansion and avoid exploding the network parameters during learning. In other words, the expansion based approaches deal with the *forget-vs-expand* dilemma, a variant of the classical stability-plasticity dilemma [1].

Existing approaches [17, 24, 22, 10, 23] address the dilemma in different ways. DEN [24] adds new fine-grained neurons selectively per layer, and minimizes expansion by using group sparsity regularization on newly added parameters. PGN [17] *freezes* the learned parameters of the earlier tasks and uses lateral connections to reuse old parameters for the new task. RCL [22] uses reinforcement learning to compute the number of channels that should be added to each network layer, when a new task arrives for training. APD [23] constrains network parameters to be a sum of shared and (sparse-) task specific ones, and further decomposes and clusters task specific parameters to minimize expansion. The Learn-to-grow [10] method, which is closely related, specifies that each network layer may either be reused, adapted or cloned for the new task, and performs differentiable neural architecture search (DNAS) [11] over a combined architecture to ascertain the optimal expansion

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

choice for each layer. Although all existing methods trade-off between growth and interference in different ways, none of them directly exploit *task similarity* or *layer similarity* for selective expansion. They either expand upon the previous network as a whole or add fine-grained neuron units. In contrast, we propose to *identify* an existing coarser-level task *sub-network*, which already computes features relevant to the new task, and reuse it for incremental learning. Moreover, existing methods perform per-task optimization only, without considering global optimization across the task set. Global optimization enables finding networks with better performance-size tradeoffs, which task-local optimization can fail to find.

We present *Learn to Bind and Grow* (L2BG), a new expansion based, task-incremental learning approach, which addresses the above issues. L2BG exploits the observation that in most task groups learned together, the tasks are mutually similar to different degrees. Hence, to learn a new task t, we find a previously learned task b which is similar to t (*bind*), and then build the task network for t upon the learned network structure for b (*grow*), without modifying the rest of the network structure. By considering only a small set of layers from b for expansion, we curtail network growth as well as boost learning efficiency, while retaining the task-specific performance.

L2BG maintains a joint, parameterized network $\mathcal{G}$ for learning multiple tasks, with both task-specific and shared layers. For processing a new task t, we first identify an existing trained task b in $\mathcal{G}$, which is similar to t and then bind t to b. Binding implies that tasks t and b now share the same sub-network $\mathcal{G}_b$ of $\mathcal{G}$. Next, we identify one or more layers l of $\mathcal{G}_b$, on which t and b may *conflict*, i.e., sharing l between t and b may cause catastrophic forgetting for the tasks. We then remove the conflict by *growing* a new corresponding layer l' for $\mathcal{G}_t$, and re-train $\mathcal{G}_t$ for t. Both task binding and layer expansion rely on a novel, hierarchical, *conflict estimation* model proposed in this paper. The model makes the expansion process *interpretable*: we can explain the task bindings and layer expansions choices precisely, using the conflict model scores.

We show that our bind-and-grow strategy induces a new parameterization of the space of shared, multi-task architectures, which is linear in the number of tasks. This, in turn, allows us to employ Bayesian Optimization [14] to search for optimal network architectures, under multiple objectives: maximize the average task performance and keep the joint network size low. Our parameterization enables exploring the space of performance-size tradeoffs by constraining global optimization *flexibly*, to be effective even in low budget scenarios.

In summary, our contributions are:

- A new method to perform task-incremental learning, which reuses previously learned information efficiently by binding to specific trained sub-networks, and dynamically expands conflicting layers to avoid catastrophic forgetting among multiple tasks.

- Unlike existing expansion based methods, our method exploits task similarity and includes an interpretable criteria for dynamic expansions based on an hierarchical conflict model.

- A new parameterization of the multi-task architecture space, which enables efficient multi-objective Bayesian Optimization to find Pareto optimal solutions [16, 19].

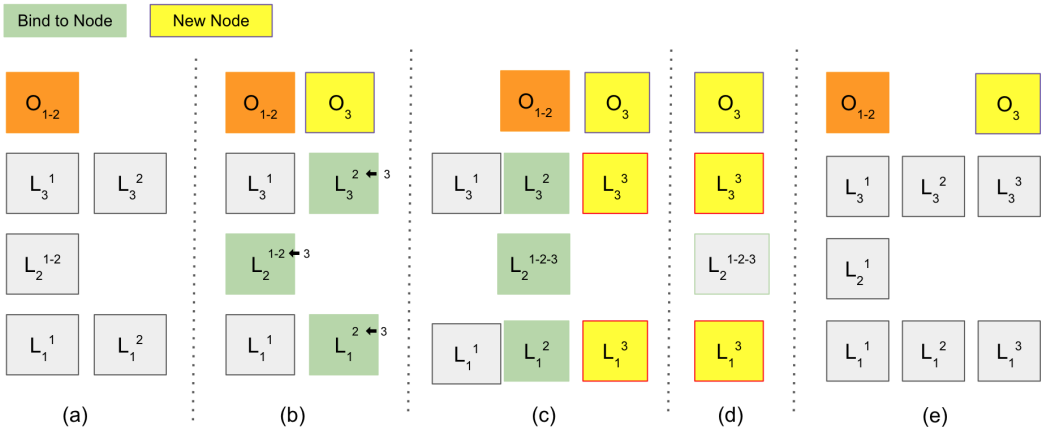

Figure 1: illustration of the proposed learn-to-bind-grow framework. a)Initial state of the Joint Network. The subscript denotes layer index and super script task index, In this example initially there are 2 copies of layer 1 and 3 while layer 2 is shared by tasks 1 and 2. b)New task binds to layers of task2. c)Conflict model is estimated and conflicting layers expanded. d)The new layers and bound layers are trained on new task. e)Joint network is updated to add newly created layers.

2 Preliminaries

Consider a large collection of tasks T to be learned. Each task $t \in T$ has train, validation and test datasets, $\mathcal{D}_t$, $\mathcal{V}_t$, $\mathcal{T}_t$, respectively. Individually, each task has a learnable model $f(\cdot;\theta_t)$, parameterized over θ_t, a loss function $\mathcal{L}_t$. The model is represented as a neural network graph, with parameters distributed among layers, which are drawn from a fixed set L.

In this paper, we consider two scenarios: (a) training data for task t is not available after t is trained (streaming) and (b) training data for all tasks is available throughout, but training is incremental per task. In the first scenario, L2BG performs task-incremental optimization only, whereas in the second one, it performs global optimization across tasks also. Several large-scale multi-task problems across text, vision and RL domains [21, 2, 26, 25] fall into the latter scenario.

During task-incremental learning, we construct a joint neural network graph for tasks in T, called JointNet $\mathcal{G}$, with task-specific as well as shared (between two or more tasks) layers. Formally, the JointNet is a directed acyclic graph (DAG) $\mathcal{G} = (N, E)$, where nodes N correspond to a layer $l \in L$, and edges E connect the layers. The DAG $\mathcal{G}$ has a single source node (input) and multiple sink nodes (outputs), where each sink node corresponds to one or more tasks. For each tasks $t \in T$, we can select a sub-graph $\mathcal{G}_t$ from $\mathcal{G}$; $\mathcal{G}_t$ may share nodes with sub-graphs $\mathcal{G}_{t'}$ of other tasks t'. We call $\mathcal{G}_t$ as the *task-net* for t and that $\mathcal{G}$ *contains* a task-net for each t. In this paper, we assume that task-nets for all tasks $t \in T$ are similar: there exists a *bijective* mapping between layers of each pair of tasks. We use the train dataset $\mathcal{D}_t$ for training each task only once; the validation datasets $\mathcal{V}_t$ are used for evaluation throughout the learning. We also define a *degenerate* JointNet $\mathcal{G}^{\downarrow}$, where task sub-graph $\mathcal{G}_t^{\downarrow}$ for t does not share any nodes with sub-graph $\mathcal{G}_{t'}^{\downarrow}$ for t'. The network $\mathcal{G}_t^{\downarrow}$, called the *independent* task-net for t, is used to train task t in isolation.

3 Learn to Bind and Grow

We now describe the proposed method, Learn to Bind and Grow. The method maintains a shared, multi-task, model JointNet $\mathcal{G}$, for a set of tasks T, containing a task-net for each $t' \in T$, When a new task t arrives, the method *binds* the task-net for t to the task-net of an existing, *similar* task b. Intuitively, this means that the task-net layers of b and t are shared initially, ignoring any difference in the task-specific architectures of the two tasks. In the *grow* step, the method expands the task-net for b by adding new task-specific layers for t, and re-routes the task-net of t to use the new layers.

Algorithm 1 Bind-Grow-Step Algorithm

Input: JointNet $\mathcal{G}$, task t, bind candidate tasks B, grow coefficient δ_t
Output: Expanded JointNet $\mathcal{G}'$, Validation Errors $\mathcal{E}$
for each pair (t, b'), $b' \in B$ **do**
    Compute task conflict scores $\mathbb{C}_{t,b'}$ and the layer-wise conflict distribution $\tilde{\mathcal{C}}_{t,b'}$
end for
Let bind task b := argmin$_{b' \in B}$ $\mathbb{C}_{t,b'}$.
Compute the nucleus layers L' using the cumulative $\tilde{\mathcal{C}}_{t,b}$ distribution.
Expand layers L' in $\mathcal{G}$ to obtain $\mathcal{G}'$.
Train $\mathcal{G}'_t$ using train data $\mathcal{D}_t$.
Compute validation errors $\mathcal{E}_t$ for all existing tasks t using dataset $\mathcal{V}_t$.

There are two key problems here: (a) which task b to bind t with, (b) which layers of b to grow. We propose a novel, hierarchical, *conflict model* to address both problems. The model computes layer-wise conflict scores $\mathcal{C}$ between the task-nets of t and each task b from the set of bind *candidates* B. These scores play a dual role. First, by aggregating pairwise layer scores to pairwise task scores, we determine which task $b \in B$ conflicts the *least* with t; t binds to a task b with the least conflict score. Second, given a layer l in the task net of b, the scores indicate the *likelihood* of b and t to *interfere* on l. The higher the score, more likely the interference and so l cannot be shared between b and t.

In this paper, we assume that the structure of task-nets across tasks is similar, i.e., we have a bijective mapping between layers of individual task-nets for each task pair (t, b). Let l index over the layer pairs L in the mapping. We denote the layer-wise conflict score for a layer $l \in L$ as $\mathcal{C}_{t,b}(l)$.

Binding. Given a set of candidate tasks $b' \in B$ and the layer-wise conflict scores $\mathcal{C}_{t,b'}(l)$, we compute the pair-wise task conflict scores $\mathbb{C}_{t,b'}$ as $\mathbb{C}_{t,b'} = \|\mathcal{C}_{t,b'}(l)\|$. Now, task t binds to a candidate task $b \in B$ with the least $\mathbb{C}_{t,b}$ score. Binding implies that the task-nets for t and b are shared; hence we also say that $\mathcal{G}_t$ binds with $\mathcal{G}_b$.

Grow-Step. Once we bind the task-net $\mathcal{G}_t$ for t to the task-net $\mathcal{G}_b$ of a prior task b, we again use $\mathcal{C}_{t,b}$ to determine which shared layers are more likely to cause interference, and try to *unshare* those layers by expanding them. To expand a layer l, we create a new copy l_t of l and modify incoming and outgoing edges to l_t and update the task-net $\mathcal{G}_t$ for task t, while keeping the rest of the JointNet intact. To compute the layers for expansion, we first transform per layer conflict scores into a probability distribution $\tilde{\mathcal{C}}_{t,b}(l)$ over layers $l \in L$. Now, we may *sample* layers from $\tilde{\mathcal{C}}_{t,b}$ based on their probability mass, and expand them individually. This, however, leads to a combinatorial explosion in the space of explored architectures.

Grow Coefficients. To mitigate this, we introduce a new parameterization based on a *grow coefficient* per task. A grow coefficient $\delta_t \in [0, 1]$ for task t provides an *dynamic* upper bound on the number of new layers created for t, based on the cumulative probability distribution of $\tilde{\mathcal{C}}$.

Given a δ_t value for task t, we first compute a *nucleus* [6] of $\tilde{\mathcal{C}}$: the smallest possible set of layers, whose cumulative probability under $\tilde{\mathcal{C}}$ exceeds the bound δ_t. We then expand all the layers in the nucleus to obtain a new task-net $\mathcal{G}_t$, which is re-trained using the training dataset $\mathcal{D}_t$ for t. Training $\mathcal{G}_t$ inside the joint network $\mathcal{G}$ enables forward information transfer from bound task b (and its transitive bindings), and often improves the task performance on t. Note that with a single coefficient δ_t per task t, we can span the complete expansion space for t: from no expansion or fully shared network with b (δ_t 0) to expanding all the layers (δ_t 1). This capability comes with a caveat that all expansions must abide with the layer ordering given by the conflict distribution $\tilde{\mathcal{C}}$. We summarize our Bind-Grow-Step algorithm in Algorithm 1.

Computing the Conflict Model. Computing a precise, layer-wise conflict model between tasks involves characterizing interference precisely [28, 29, 15], which is intractable [1]. Several approaches to approximate interference are possible but expensive, e.g., we may identify the overlap of important parameters for both tasks using the Fisher Matrix [7] or try different combinations of layer expansions with re-training and evaluation. Instead, we compute a low-cost, approximate conflict model, by correlating the layer-wise features of an independently trained task-net $\mathcal{G}_t^{\downarrow}$ for t (see Sec. 2) and the current task-net $\mathcal{G}_b$ for binding candidate b. We use the current (fully-trained) task-net $\mathcal{G}_b$, instead of

say, a pre-trained $\mathcal{G}_b^{\downarrow}$, to ensure that the conflict model is computed in context of the growing network. When a task t arrives, we first train an individual network $\mathcal{G}_t^{\downarrow}$ for t. Then, we sample data from validation set $\mathcal{V}_t$, feed to both $\mathcal{G}_b$ and $\mathcal{G}_t^{\downarrow}$ and collect post-activations after each layer of the networks. Finally, using the representational similarity analysis (RSA) algorithm [8, 5], we compute correlation scores between pairwise post-activations of mapped layers $l \in L$, for every sample in $\mathcal{V}_t$. The conflict score $\mathcal{C}_{t,b}(l) = \rho$ where ρ denotes the RSA dissimilarity score for l. Our implementation follows the original RSA algorithm [8] and requires space quadratic in the size of layer post-activations.

In the above discussion, we use a *layer* as the basic unit of expansion. More generally, we can partition a layer into sub-components, e.g., a group of channels, and grow one or more groups in each grow step. The notion of a conflict distribution extends and applies to any well-defined, hierarchical, partition of the network structure.

Grow Sequences. By performing a bind-grow step iteratively, for each task in an incoming sequence over tasks T, we obtain a grow sequence $\pi = [(t, b_t, \delta_t)]_{t \in T}$. The sequence characterizes the choices made (bind task t to b_t and expand with grow coefficient δ_t) during a particular task-incremental learning trial. By exploring different grow sequences, we can explore different task orders, task bindings and degrees of network expansion, and optimize over the global architecture space.

Evaluation. When a grow sequence finishes execution with the network $\mathcal{G}$, we compute two measures over $\mathcal{G}$: the average validation error over tasks and the *average multi-task gain* Δ, based on the average per-task drop measure in [13]. For a task t, let the validation error after executing a grow sequence π be $\mathcal{E}_{\pi,t}$ and for the single-task baseline be $\mathcal{E}_{\phi,t}$. The average multi-task gain Δ_π is defined as

$$\Delta_\pi = \sum_{t \in T} (\mathcal{E}_{\pi,t} - \mathcal{E}_{\phi,i}) / \mathcal{E}_{\phi,t}$$

Comparison with Expansion Approaches. Compared to Learn-to-Grow [10], we perform expansion relative to the similar, bound task net, not the full network. We perform global, as opposed to per task, optimization over model structures. We do not consider *adapting* a layer here; our method can be extended by allowing both the task-net $\mathcal{G}_t^{\downarrow}$ and its adapted version to bind with an existing task-net. Compared to RCL [22], our expansion relies on an approximate, conflict model, which allows one or more layers (or layer components) to be expanded simultaneously, as opposed to learning how many nodes to expand per layer. APD [23] assumes a shared and task-specific decomposition for each layer; our method works with the unchanged network structures directly. Finally, we exploit task similarity scores to guide expansion; neither PGN [17] or DEN [24] do so.

Global Optimization. Grow sequences based search optimization is attractive for several reasons. First, they enable global optimization over the *complete* space of shared, multi-task architectures. Next, the optimization is *interpretable*, i.e., we know precisely why a task binds with an older task, as well as, why a particular set of layers are expanded. Further, the optimization is governed by a small (linear in task set size) set of hyper-parameters, namely, the binding task index, and a grow coefficient for each task. Also, we can perform *constrained* optimization under low computational budgets. For example, we may constrain search to a fixed, prior set of bind targets, or alternatively, only explore a small, curated set of grow coefficients, while computing bindings dynamically.

Bayesian Optimization. Exploring all grow sequences to find optimal architectures is intractable because evaluating each grow sequence is expensive. Bayesian Optimization (BO) is a technique to optimize expensive black-box functions efficiently, by using a surrogate probabilistic model, e.g., Gaussian Processes, to model the uncertainty of the unknown function. For details on BO, we refer to Rasmussen and Williams [14]. Additionally, our problem involves *multiple, competing objectives*: average multi-task gain and network parameter count, which we want to maximize and minimize, respectively. Hence, we seek to recover the Pareto optimal solutions [16, 19]: solutions which cannot be improved in one objective without degrading the other objective.

In a grow sequence $\pi = [(t, b_t, \delta_t)]_{t \in T}$, task identifiers t and b_t are (bounded) categorical choices while the coefficient $\delta_t \in [0, 1]$ is sampled from a continuous range. Using BO, we optimize the black-box function $\mathcal{F} : \Pi \rightarrow (\Re \times \Re)$, which maps a growth sequence $\pi \in \Pi$ to a pair of outputs, the average multi-task gain and the parameter count.

4 Experiments

In this section we discuss the effectiveness of using our framework in the existing space of dynamic network expansion methods that tackle the problem of continual learning. We demonstrate the efficacy of our proposed method through experiments on benchmark datasets like **permuted MNIST**, split-cifar100 and VDD. The **permuted MNIST** dataset is derived from the **MNIST** dataset[**?**], each task is a simple classification problem with a unique fixed random permutation generated to shuffle the pixels of each image, while the annotated label is kept fixed. The **Split cifar100** dataset is a variation of the cifar100 dataset with the classes divided into T tasks with each task having 100/T classes. The visual decathlon dataset(VDD) consists of 10 image classification tasks – ImageNet, CIFAR100, Air-craft, DPed, Textures, GTSRB, Omniglot, SVHN, UCF101and VGG-Flowers (add multi-lingual).

For the permuted MNIST experiments, we use a 4-layer fully-connected network with 3 feed-forward layers and the 4th layer is the shared softmax classification layer across all tasks. For split-Cifar100 we use the network used in [10]. The network consists of 3 convolutional layers with maxpooling and relu activations followed by two fully connected layers and finally separate task heads for each incoming task.

How does RSA help? The architectural search space is too vast with a decision to either reuse or expand at each layer required to be made in the shared network for each new task that arrives. Searching the whole space by considering all possible decisions for all the layers in the the shared network is computationally too expensive. RSA helps narrow down search to smaller number of locally optimal decisions. To illustrate the difficulty of finding an optimal shared network we perform a random growth experiment on split cifar100. Whenever a new tasks arrives, We randomly choose to bind to a task that has already been encountered, randomly choose number of layers to expand and randomly choose which layers to expand, as shown in fig the network obtained through random growth is sub-optimal and performs worse when compared to the network formed using RSA.

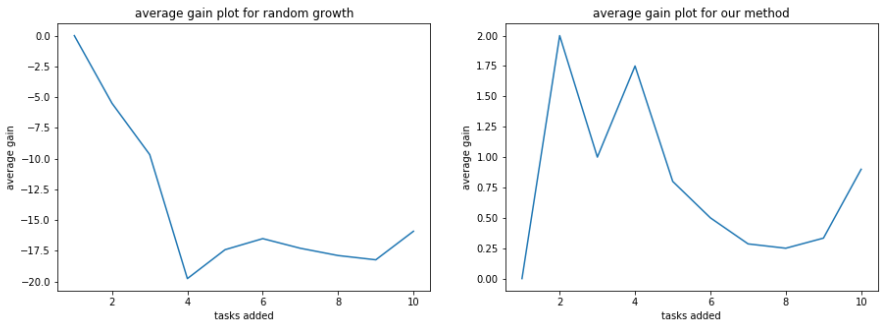

Figure 2: (a)average gain graph for random growth

complexity of model found?

The complexity of the model formed depends on the value of the growth coefficient chosen we perform a grid search over the growth coefficient value in the range of [0.1-0.9] to find the optimal value with the best tradeoff between the model performance and model complexity. if the value of the growth coeffiHigher the value of the growth coefficient more likely is

are meaningful structures found by RSA? ->In the MLT experiment a initial few layers added and seperate batch normalisation layer added. ->Observed incoming tasks more likely to avoid arab due to the difference in script ->interesting observation is the addition of separate batch normalisation layer

Is positive transfer achieved and negative backward transfer mitigated? To measure positive transfer we make use of the average gain metric from the graphs (cifar100 avg gain graph) and (pmnist avg gain graph) we can conclude that the structures sought out facilitate positive forward transfer. To prevent negative backward transfer/catastrophic forgetting freezing weights of previously learned

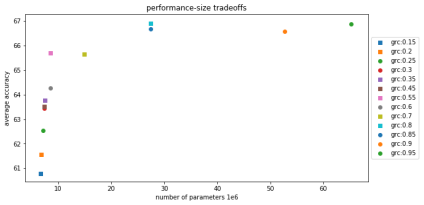

Figure 3: Flower one.

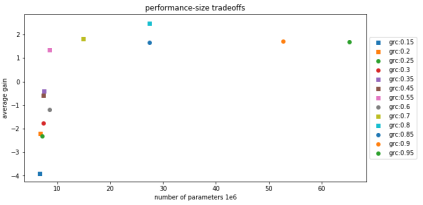

Figure 4: Flower two.

tasks can be done but this blocks any possibility of forwards transfer. Therefore we instead set the learning rate of weights of previous tasks to 1/10 of the LR of the new task this gives us the best of both worlds as can be seen in the figure(insert figure on cifar)

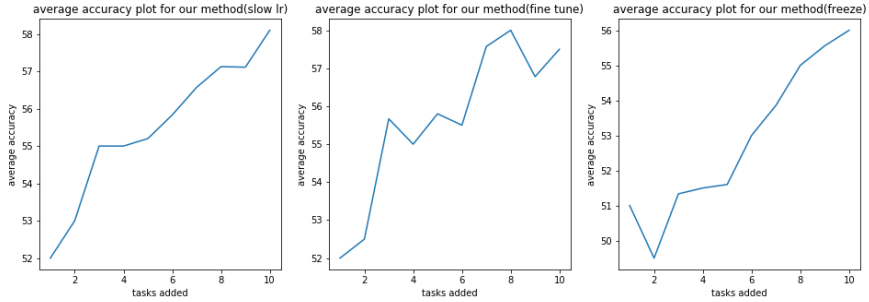

Figure 5: (a)average gain graph for random growth

comparison with state of the art dynamic expansion models. We compare our method with other recent expandable networks continual learning approaches PGN[**?**],DEN[**?**],RCL[**?**], L2G[**?**] We evaluate each compared approach by considering average test accuracy on all the tasks, model complexity and training time. Model complexity is measured via the number of model parameters after training all the tasks.see figure(bar graph) need to use Le net for comparison.

We omit comparisons to regularization based approaches to continual learning [7], as they have been shown earlier to perform worse against the expansion based methods [10, 22].

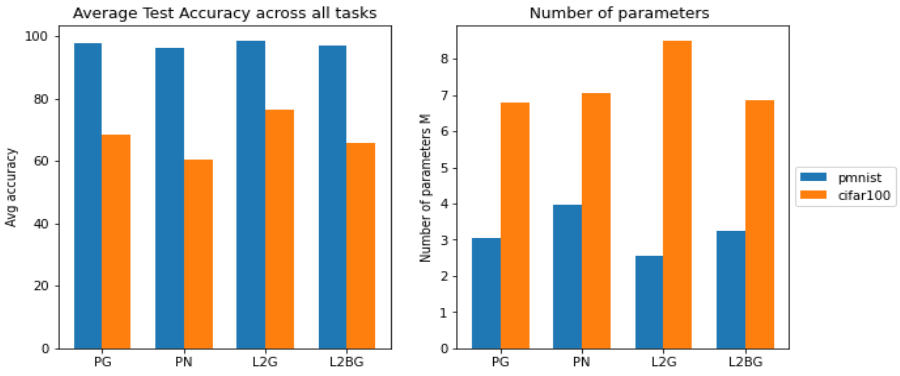

Figure 6: Experiment on the influence of growth coefficient(grc) in model complexity and accuracy on the split-cifar100 dataset

We also investigate the effectiveness of Bayesian Optimization in yielding global, Pareto optimal solutions, even under low budget scenarios. We use the Optuna framework [3], which uses the

NSGA-II algorithm [4] for multi-objective Bayesian Optimization. To restrict the large optimization space of our benchmarks for a low budget BO setting, we assume a fixed task arrival order, keep a share growth coefficient across tasks, and sample coefficient δ from a discrete uniform distribution $\mathcal{U}\{0, 1\}$ with steps of 0.05. We perform 20 grow-sequence trials, each trial taking about 11 minutes. Fig. **??** shows performance-size tradeoffs in obtained solutions for the Split-CIFAR benchmark with varying δ (boxes represent Pareto optimal solutions). In general, large δ values (~ 0.95) yield high average gains, but also yield a large parameter count. Networks with fewer parameters are clustered on the left, with varying average (negative) gains. We find a balanced solution at $\delta = 0.55$ with $< 10m$ parameters and average gain (-1.4). Interestingly, we obtain the highest average accuracy solution at $\delta = 0.8$; the fully expanded network ($\delta \sim 0.95$ is inferior both in size (more than double) and performance. Note that being able to visualize the Pareto optimal solutions enables us to make informed performance-size tradeoffs for downstream deployment scenarios. No other existing expansion based method [24, 17, 10, 22, 23] is able to explore the space of optimal solutions globally.

5 Conclusions

We present a new framework for task-incremental learning, which learns to expand a joint network for multiple tasks dynamically, based on two new ideas: (a) exploit task similarity by binding an incoming task with the network of an existing, similar parent task, and (b) grow the layers relative to the parent task, by estimating a hierarchical conflict distribution between the layers of the parent and new tasks. The method enables multi-objective (performance, size) optimization across the space of shared, multi-task architectures. There are several directions for future work: exploring techniques to scale BO to large multi-task benchmarks [26, 2], scaling RSA computation to large layer activations, enabling expansion for non-symmetrical task networks, and adding layer adaptation to our algorithm.

6 Broader Impact

The current AI systems need to learn a large number of separate micro-skills, often one at a time, to be more useful for the society. A key hurdle towards this goal is that, when trained on multiple sequential tasks, the current systems often forget old skills partially or catastrophically, leading to wastage of tremendous resources. The paper is an effort to mitigate this problem.

Positive effects of this work include being able to build multi-skilled AI systems with fewer resources, smaller model sizes, and higher success ratio. Negative effects of this work include easy creation of multi-skilled AI systems for harmful purposes, as well higher carbon footprint for automatically discovering architectures for such systems.

References

[1] W. C. Abraham and A. Robins. Memory retention – the synaptic stability versus plasticity dilemma. *Trends in Neurosciences*, 28:73–78, 2005.

[2] R. Aharoni, M. Johnson, and O. Firat. Massively multilingual neural machine translation. In *NAACL*, 2019.

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. KDD '19, 2019.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 2002.

[5] K. Dwivedi and G. Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *CVPR*, 2019.

[6] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *ICLR*, 2020.

[7] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. 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. In *PNAS*, 2017.

[8] N. Kriegeskorte, M. Mur, and P. A. Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2008.

[9] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NIPS*. 2017.

[10] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML 2019*.

[11] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[12] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017.

[13] K.-K. Maninis, I. Radosavovic, and I. Kokkinos. Attentive single-tasking of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[15] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.

[16] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *J. Artif. Int. Res.*, 2013.

[17] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[18] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress compress: A scalable framework for continual learning. In *ICML*, 2018.

[19] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. In *NeurIPS*, 2018.

[20] S. Thrun. In *IROS), title=A lifelong learning perspective for mobile robot control, year=1994,*.

[21] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. 2019.

[22] J. Xu and Z. Zhu. Reinforced continual learning. In *NeurIPS*, 2018.

[23] J. Yoon, S. Kim, E. Yang, and S. J. Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *ICLR 2020*.

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

[25] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019.

[26] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.

[27] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Ratcliff, R. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

Robins, A. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995