SMART: Self-organized Multitask Resilient Learning

Blind Review Version

Abstract

In this work, we propose a self-organized multitask resilient learning paradigm through evolution, named SMART, by constructing and organizing neurons assemblies into modules-level building block. *Exploitation* and *Exploration* are the two executive phases of SMART, the former one, termed *self-organization*, is striving for satisfy required performance by leveraging existed modules, whereas the latter one is able to change the number of modules by evolution to adjust the expressive ability of the network. The network is initialized with only one convolutional and one output module, keep learning different tasks continually, the network architecture is evolved progressively and specific-to-general combinatorial connectivity pattern within the neural network could be achieved. Most importantly, after a certain number of learning tasks, we are able to obtain a satisfying performance when encountering a new task, only by *self-organization*. Experiments demonstrate SMART dramatically increases the specific-to-general learning ability of ANNs. After training our network with 5 randomly selected MNIST binary classification tasks, *self-organization* boosts the transfer ability from 28.9% to 97% accuracy on unforeseen tasks. Furthermore, when transfer from different task sets, e.g. from CIFAR10, SVHN to CIFAR100, SMART increases 11.0%-17.2% transfer ability.

Introduction

Fast, incremental and continuous learning are the essential ingredients that enable intelligent systems to adapt to the changing world (Parisi et al. 2018)(Shin et al. 2017). However, the fundamental architectural abstraction of intelligence that enables human brain to deal with various possible patterns and knowledge is still the biggest mystery. Nevertheless, this challenge does not prevent but encourage people to explore possible solutions to approaching this ultimate goal in artificial neural network research area (Xu, Tao, and Xu 2013)(Kirkpatrick et al. 2016). Recently, multitask learning (MTL) is a promising learning paradigm in machine learning and its aim to leverage useful information contained in multiple related tasks, ideally without the need to memory large specialized parameters for each individual task, to help improve the generalization performance of all tasks (Zhang and Yang 2017).

On the other hand, since deep neural network has scaled up to deal with more complex tasks, with hundreds of hyper parameters and sophisticated architectures, how to design such systems has emerged as a major challenge in front of us (Jaderberg et al. 2017), (Szegedy et al. 2015a), (Szegedy et al. 2015b), (Baker et al. 2016). Manually design network architecture through experimentation will become very difficult, if not infeasible. Lately, some evolutionary optimization schemes of neural networks have been proposed try to tackle this problem (Miikkulainen et al. 2017)(Liang, Meyerson, and Miikkulainen 2018)(Fernando et al. 2017),(Chen et al. 2018), in which researchers are responsible for the high-level design and evolutionary strategies, and leave the optimization systems to leverage the increasing available computational power to figure out the appropriate network architecture. However, how to leverage the current existed components to promote the learning capability for new tasks is overlooked to some extent.

This work touches the field of MTL and evolving neural network. We propose a self-organized multitask resilient learning paradigm through evolution named SMART. The underlying principle of SMART is that if new tasks are partially similar to previously encountered tasks, then leveraging past knowledge can increase the learning rate and capability for the new ones. We construct and organize neurons assemblies into modules-level building block and different executive phases are conducted on different granularities. Specifically, *Exploitation* and *Exploration* are the two executive phases of SMART. *Exploitation* consists of probing a limited region of the search space—current existed modules and updating learning parameters among modules, we termed the *Exploitation phase—"self-organization"*. *Exploration* phase, on the other hand, consists of probing a much larger portion of the search space to meet the required performance by incrementally adjusting the number of learnable modules and updating parameters both inside and among modules through evolution.

When encountering a new task, the *self-organization* (*Exploitation phase*) is launched firstly, try to achieve required performance by utilizing already existed modules. If the network at current evolutionary strategy is not able to reach the expectation, the *Exploration phase* will be executed by adding new learnable modules to enrich its expressive ability through fitness approximation and gradient descent.

The two phases are conducted alternatively, keep learning more tasks, the specific-to-general combinatorial connectivity pattern within the network could be shaped. Experiments on binary classification and multi-class classification tasks demonstrate that, after a certain number of learning tasks, a relatively satisfied performance can be achieved by simply *self-organizing* the currently existing modules.

In a nutshell, the novetly of the SMART is threefold:

- **Enhanced Modules.** Network performance is highly relied on the sufficient expressive ability of the basic building block—modules. Two powerful module named Res-Fire and Dimension Reduction modules are introduced into SMART to acquire noticeable improvement when transfer among different tasks.
- **Self-organization.** The exploitation of SMART is manipulated in the granularity of module-level through learnable scale parameters adjustment, the confined search space is capable to achieve high-efficient specific-to-general learning ability when the network reaching a certain scale.
- **Evolutionary Algorithm.** The proposed evolutionary algorithm is same as the traditional procedure of the evolutionary algorithm, contains *reproduction, mutation, recombination* and *selection*. In order to short the time-consuming process of evolution phrase, we proposed a rational evaluation criteria through experiments to balance the performance accuracy and convergence speed.

SMART is a step towards enabling deep MTL to realize 'specific-to-general' learning ability. We release in TensorFlow version of our design on MNIST, SVHN and CIFAR validation sets. The model and supplement materials are available at https://github.com/Wind-Wing/SMART.

Related work

A selected number of perviously papers touches MTL and neural network design are reviewed before introduce the proposed SMART.

Traditional neural networks have brought great success in many fields(Simonyan and Zisserman 2014), (He et al. 2015)(Szegedy et al. 2016)(Xie et al. 2017). But the system performance is highly relying on engineers to design a proper neural network architecture. This procedure is very time consuming. There are some work demonstrate how evolution can be instrumental in advancing multi-task network design(Fernando et al. 2017)(Meyerson and Miikkulainen 2017)(Liang, Meyerson, and Miikkulainen 2018)(Chen et al. 2018).

In order to transfer knowledge among tasks, Multi-task learning(MTL)(Caruana 1998) has achieved good results in many area like vision(Bilen and Vedaldi 2017), speech task(Huang et al. 2015), natural language processing (NLP)(Dong et al. 2015)(Hashimoto et al. 2016), and reinforcement learning(Devin et al. 2017)(Jaderberg et al. 2016).

Large-Scale Evolution of Image Classification(Real et al. 2017) use evolutionary methods to fine-tune hyper-parameters. In the experiment, the author set a total of 1000 population and evolving it with 250 machines, using tournament selection method to select a good individual, and then mutate it afterwards to achieves good results. However, the computational resources are unaffordable for most researchers. In PathNet(Fernando et al. 2017), Fernando uses evolutionary algorithm to figure out which part of the network to be activated. In order to reduce the searching space of evolutionary algorithm, PathNet uses modules to encapsulate a block of components and parameters, adopts layered structure to arrange modules and uses sum operation to gather all the information from different modules before passing them to the next layer. In this way, the exploring space is sharply reduced when applying evolutionary algorithm to investigate which sub set of the modules to be shared in the transfer phase. The structure of SMART is inspired by PathNet, that arrange resources in the module granularity.

Modularity is an efficient way to achieve transfer learning. In Meyerson' work (Meyerson and Miikkulainen 2017), their experiences revealed that permuted ordering of shared layers performances better than parallel ordering. Further they demonstrated using scale parameters to approximate permuted order is a practical approach. . In order to enhance the composability of the modules and improve the rate of modular reuse in the *self-organization* process, we also applied the same approach in SMART.

SMART

The basic idea of SMART is that, considering there are total of T tasks $\{t_1, t_2, ..., t_T\}$ need to be learned continually, theoretically, there exists a set of different F features $\{f_1, f_2, ..., f_F\}$, and the combination of these features is capable to handle all the T tasks. In this way, if the ANNs keep learning new tasks, new modules can be employed by the evolutionary algorithm to learn new features. The already mastered feature set is marked as $\hat{F}$. After several iterative round of evolution and learning, the elements in set $\hat{F}$ is keeping increasing, if $\exists\ f_i \in \hat{F}$, the ANN is general enough to deal with unseen tasks t_i, perhaps even without the necessary of further training further training. In the following, we first introduce the basic building blocks of SMART and then dig into the detailed implementation of our design.

Building blocks

Modularity is an effective and powerful way to achieve component and parameter reuse, as well as system extensibility in MTL paradigm by encapsulate sub-networks (Fernando et al. 2017). The generalization ability of ANN is highly relied on the sufficient expressive ability of the set of modules. For SMART, there are four basic building blocks, including convolutional module, fully connected modules and output layer and two newly proposed modules, named *Res-Fire (RF)* and *Dimension Reduction (DR)* modules, see Figure 1.

Res-Fire module: *Fire module* is a two-layer structure that is introduced in the SqueezeNet (Iandola et al. 2016). The first layer is composed of one 1×1 convolution kernel and the second layer is composed of two different size of convolution kernels, one is 1×1 and the other is 3×3 ($padding = 1, stride = 1$). The output of two convolu-

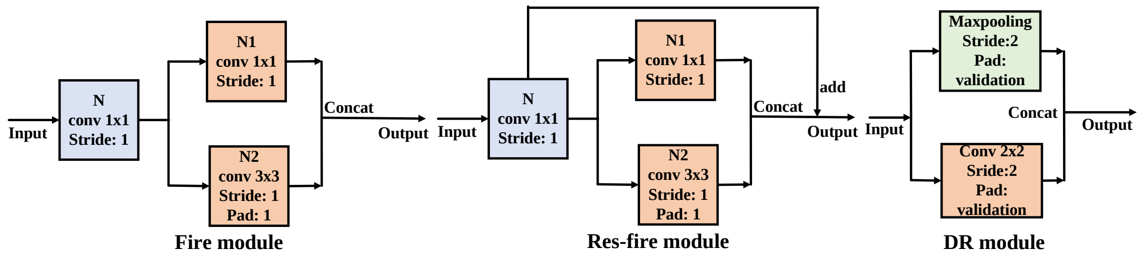

Figure 1: **Enhanced modules:** *RF* module adds a bypass to the *Fire module*, and *DR* module combines the max-pooling with convolutional filters.

tion kernels are concacted together. *RF* module is an enhanced version derived from *Fire module* with residual link that adds input feature to the output. Notice that, the inputs and outputs should have the same channel number.

Dimension reduction module: Max-pooling is often used to reduce network size and prevent overfitting, functionally it can be regarded as a smoothing filter for the previous layer. If we only transmit the features through max-pooling layer without data interaction before and after it, which might arouse information irreversible and feature loss. Therefore, *DR* module combines max-pooling with 2×2 convolutional filters($stride = 2$) to reduce network size while prevent irreversible information loss.

Experiments in Section demonstrate the transfer ability of the ANN is greatly improved after introducing these modules. Notice that, SMART is not limited to the aforementioned modules, other modules can be expanded and enrich the set of basic building blocks of SMART.

Network Architecture

In order to characterize the continuous and multitask learning properties of SMART, two issues need to be addressed, 'what to share' and 'how to share'. 'What to share' asks which components of the network can be shared across tasks. Same as *PathNet* (Fernando et al. 2017), SMART is organized on the module granularity, neurons encapsulated in module-level are shared among different tasks. After determining 'what to share' and 'how to share' specifics concrete ways to share modules among tasks. Intuitively, modules may be shared among different tasks with equal or different effectiveness, and this character should be reflected on the learnable parameters.

The structure of SMART draws inspiration from *PathNet*, see Figure2. *PathNet* applies modular mechanism to achieve efficient components and parameters reuse during transfer learning by freezing a sub-set of networks in the module granularity to avoid catastrophic forgetting. *PathNet* is a relatively redundant network comprises L layers, each layer contains M modules and up to N distinct modules in one layer are permitted to active at the same time for a specific task. There are F_M filters in each module. The outputs of activated modules are summed up before sending to the next layer to reduce the search space of evolutionary algorithm. Notice that L, M, N and F_M are predetermined configuration parameters according to the complexity of different target tasks.

There are three main differences between SMART and *PathNet*: a) *PathNet* only applies convolutional and fully connected modules, we introduce enhanced modules (*RF* and *DR*) to improve the expressiveness of ANN; b) we add learnable scale parameters α and β associated with modules on the concentration and scatter phase between different layers to represent different effectiveness of features for different tasks, see Figure 2(b); c) The scale of SMART is much more resilient, it is task-oriented rather than predefined, so the ANN is capable to expand and shrink according to different task requirements.

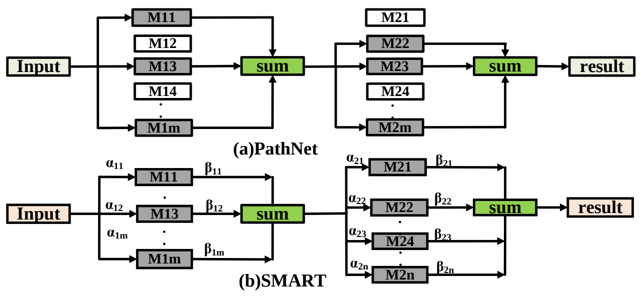

Figure 2: **Network Architecture.** *PathNet* is a redundant network. The number of layers (L), allowed activated and total number of modules (N and M) on each layer are predetermined. In (a), the dark-gray rectangle are the activated modules while the white rectangle are inactivated modules. For SMART, it is initialized contains one convolutional module (Input rectangle) and one output module (result rectangle). The developed architecture is automatically and progressively constructed by the proposed evolution algorithm to satisfy the tasks requirement.

Evolutionary Algorithm

Design neural network architecture for multitask learning is a compelling domain for evolutionary optimization (Miikkulainen et al. 2017)(Liang, Meyerson, and Miikkulainen 2018). Manipulating building blocks in the module granularity highly reduce the exploring space when applying evolutionary algorithm to construct the appropriate architecture.

When encountering a new task, the purpose of *self-organization* is to achieve required performance by exploiting the available modules. The network initially contains one convolutional module and one output module. At the beginning, the network keeps adding new modules to satisfy new

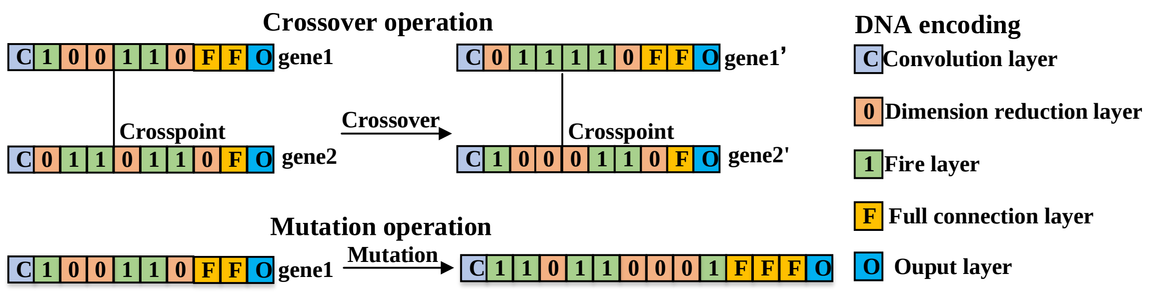
Crossover operation
C 1 0 0 1 1 0 F F O gene1
C 0 1 1 0 1 1 0 F O gene2
Crosspoint
Crossover
C 0 1 1 1 1 0 F F O gene1'
C 1 0 0 0 1 1 0 F O gene2'
Crosspoint
Mutation operation
C 1 0 0 1 1 0 F F O gene1
Mutation
C 1 1 0 1 1 0 0 0 1 F F F O
DNA encoding
C Convolution layer
0 Dimension reduction layer
1 Fire layer
F Full connection layer
O Ouput layer

Figure 3: **Crossover and Mutation.** For all the experiments, the mutation rate is 0.2. The networkis beginning with a convolution layer and ending with an output layer associated with several fully connected layers.

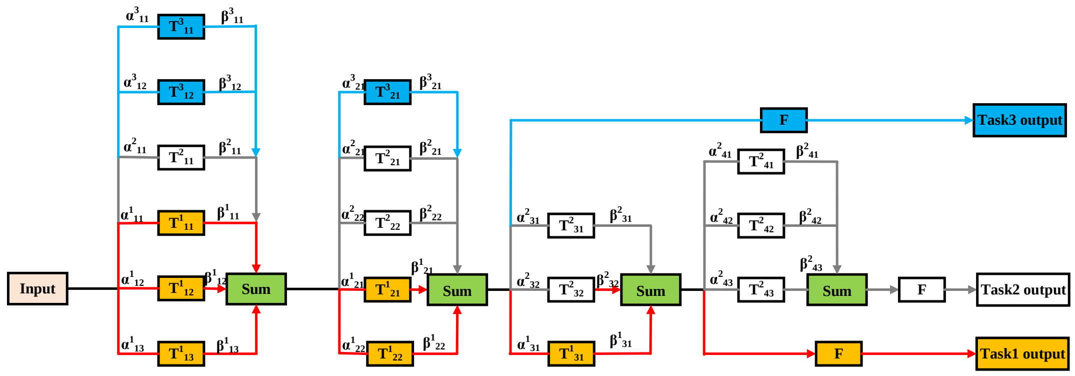

Figure 4: **Self-organization.** All the modules are shared with various tasks. T_{lk}^{t} means it is generated as the k-th module through evolution when training task i at layer l. As can be observed, for different complex task, the affective ANN are different. Self-organization harmonizes these modules for specific task with learnable α and β through gradient descent.

tasks. After that, during transfer learning, we freeze the parameters inside the modules, and manipulate with the scale parameters to examine whether previously studied modules and their combination are capable to handle the new task. If not, evolutionary algorithm will be conducted to add more learnable modules to enrich the representation ability of ANN. Then follow-up tasks are performed and the generalization ability of the network is keeping increased.

Evolution Implementation: Networks are trained for a given number of generations, performance on the target task is considered as the fitness to select the capable candidates in the population.

Candidates are trained for a certain number of steps, afterward, the candidates are evaluated and sorted by their fitness. The best two candidates are selected for further mutation and recombination to produce the offsprings, in order to keep the number of population fixed, the two recombined offsprings will replace the worst two parents. Notice that, evolution is biased towards discovering fast learners instead of top performers due to limited computation power and time constrains. A tradeoff should be required to balance the performance as well as the learning efficiency.

Fitness Criterion: The final convergent accuracy of ANN is the most suitable fitness criterion for evaluating the candidates. However, waiting the network to be converged is very time consuming. In practical, we have to choose a proper fitness criterion to balance the execution time and accuracy.

Generally, there are four indicators: a) *Max accuracy*. Choosing the best candidate with the highest accuracy at S steps, b) *Mean accuracy*. Choosing the candidate with the best average training accuracy over the last S' steps, c) *Slope of the learning curve*. Choosing the one with the biggest slope of accuracy across the last S'' steps, d) *Fitting*. Using curve-fitting method to predict the tendency of the learning curve, and the best one is chosen. According to our experiments, we selected c) as our applied fitness criterion.

Experiments conducted on CIFAR-10 are used to choose the proper criterion among those four indicators, we train the network for 6 generations ($batchsize = 256$), each generation constains six competitors. The final accuracy of evolved networks are 76.5%, 76.4%, 78.5% and 73.0% for *Max accuracy*, *Mean acuracy*, *Slope* and *fitting*. According to the results, the *slope of the learning curve* get the best result so we select this as the fitness criterion in the following experiments. Then, we introduce the *mutation* and *recombination* strategies of SMART.

Mutation strategy:

$$Var' = m_r \times r_n + (1 - m_r) \times Var \tag{1}$$

$$for \quad m_r \times len(Var) : V_M[r_n] = NOT(V_M[r_n]) \tag{2}$$

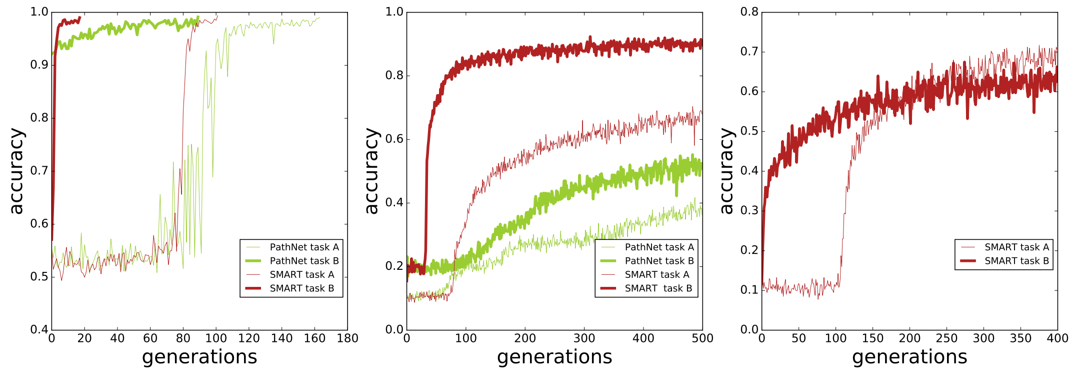

Figure 5: (a) MNIST binary classification tasks, *PathNet* with $L = 4$ layers, $M = 10$ modules and only up to $N = 3$ modules can be selected in each layer, $F_M = 20$ filters in each module. (b) CIFAR-10-SVHN tasks, $L = 4$, $M = 20$, $N = 5$ and $F_M = 20$ for *PathNet*. (c) Mini-ImageNet classification task, $L = 5$, $M = 30$, $N = 7$ and $F_M = 40$ for *PathNet*.

Crossover strategy:

$$r_n = random(0, min(len(p1.V_M), len(p2.V_M))) \quad (3)$$

$$V'_M = p1.V_M[:, r_n] + p2.V_M[r_n, :] \quad (4)$$

$$V''_M = p2.V_M[:, r_n] + p1.V_M[r_n, :] \quad (5)$$

$$Var' = random(min = p1.Var, max = p2.Var) \quad (6)$$

In the formula, V_M is an unsigned binary sequence, V'_M and V''_M are the offsprings after mutation and crossover. The bit-length of it generally represents the number of layers(L). Var represents the scalar quantity including the number of layers(L), module number in layer L (M_L), as well as filter number in each module (F_M). m_r is the mutation rate, which is a relatively small floating point number between 0 and 1. For all the following experiments, we set $m_r = 0.2$.

r_n is a number generated randomly each time with the uniform distribution between the defined minimal and maximal value of Var. $len()$ is a function return the bit-length of the operand. 'NOT' is the bitwise operation, for example $NOT(0101b) = 1010b$. $p1$ and $p2$ are the parents participate in the crossover process, and the mutation conducted at layer, module and filter level. The Var'' is an unsigned binary vector that updated after *crossover*. Notice that, since Var generally represented the L, M_L and F_M, these strategy is well adopted to all the different levels.

In our evolution strategy, SMART is striving to find the appropriate structure instead of a large redundant structure. The slope of candidates after S steps is consider as the fitness. Due to the limited training steps, the final evolution result will be a network with high accuracy and fast convergence speed speed with reasonable scale. Eliminating the redundant modules by L2 similarity measurement and weight-based soft-pruning according to α value can help to prevent the endless expansion of the network.

Self-organization for Multitask Learning

Firstly, we need to answer whether the current evolved neural network is sufficient or not to handle the new tasks by enforcing the utilization of existing modules and their combination as well, through manipulate the α and β values. Classification accuracy is measured as the performance metric. As previously mentioned, if current $\hat{F}$ is not adequate to handle the new task t_{new}, evolution based exploration is launched to learn the new features. The $\hat{F}$ is keeping accumulated and the ultimate goal is to get $\hat{F} = F$. Therefore, achieving specific-to-general learning ability, that is to say, the neural network is capable to deal with other not-encountered tasks through simply *self-organization*, see Figure4.

Experiment

Five different datasets are selected for the multitask experiments, including MNIST, CIFAR-10, CIFAR-100, SVHN and MiniImageNet. For all the experiments, we use *PathNet* as the baseline.

Classification Performance

This experiment shows how much benefit SMART can bring from enriched modules as compared with *PathNet*. For MNIST binary classification, we randomly select two numbers in the dataset to form two binary classification task A and task B. Firstly, we train task A on the ANN to research 99% accuracy. Then test the classification accuracy on task B. For CIFAR-SVHN multi-class classification, we train CIFAR-10 on the ANN for 500 generations, and test the classification accuracy on SVHN dataset. For ImageNet, we randomly choose 10 classes from the dataset to form a 10-classes classification problem, named MiniImageNet.

The comparison results of MNIST is shown in Figure 5(a), the training generation of SMART to reach 99% accuracy only takes 118 generations on task A and 17 generations on task B, whereas *PathNet* takes 163 generations for task A and 89 generations for task B. For CIFAR-SVHN tasks, see Figure 5(b), SMART achieves 69.1% and 91.5% accuracy on CIFAR (task A) and SVHN (task B) at 500 generations, respectively, whereas *PathNet* achieves 36.3% and 51.4% accuracy on CIFAR (task A) and SVHN (task B). SMART shows 1.9x and 1.8x more accuracy and achieves faster convergence than *PathNet* on the respective tasks.

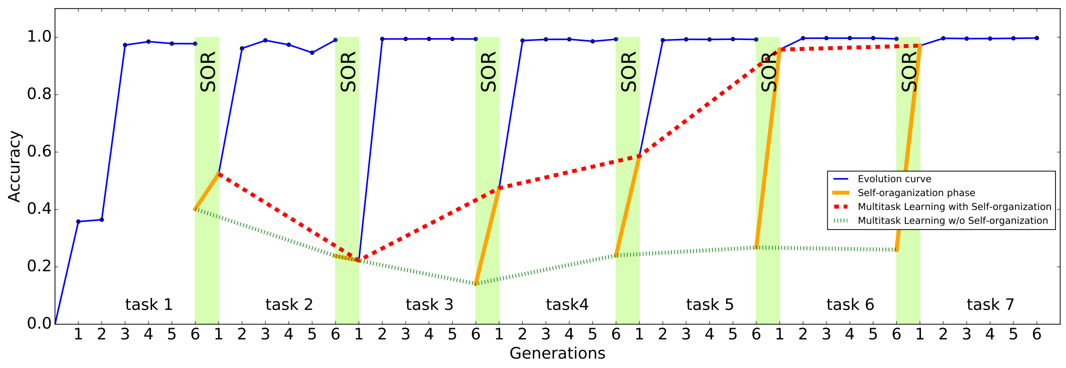

Figure 6: **Self-organization for MNIST dataset.** After learned 5 tasks, we are able to construct a well-performed network only by *self-organization* of the modules learned in previous tasks. By comparing the *self-organization* curve (red dotted line) with the green curve without *self-organization*, we can find that after several task's evolution, *self-organization* demonstrates as an effect way to achieve task transfer only by adjust the scale parameters among modules without adding new modules or changing the parameters inside the modules. The *self-organization* operation helps boosting their performance on not encountered tasks, bring about 12.3%, -1.5%, 33.3%, 34.7%, 68.9% and 71.1% higher accuracy, when transfer from task1 to task7 sequentially.

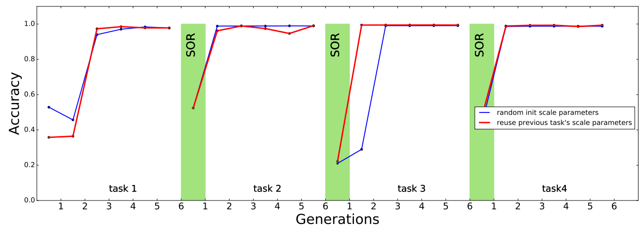

Figure 7: **Comparison two different initialization method for *self-organization* between two continuous tasks on MNIST dataset.** The network is evolved 6 generations and contains 6 candidates in each generation. Randomly initialize the scale parameters achieves a slight better performance, sometime it is difficult to distinguish, versus freeze the scale parameter of previous one.

For MiniImageNet tasks, see Figure 5(c), SMART achieves 70.8% and 66.3% accuracy on task A and task B. However, we fail to get a convergent result if we follow the *PathNet's* method at 400 generations, obviously, SMART exhibits strong learning ability than *PathNet*.

Self-organization for Multitask learning

Binary Classification tasks on MNIST datasets: Binary classification problems derived from the MNIST handwritten dataset to distinguish between two distinct randomly selected digits. In the experiments, we stop training the network at 1000 generations (batch size=256) for each competitors. There are six competitors in each generation, with a total evolution of 6 generations, slope is used as the fitness criterion to evaluate the competitors. Figure 6 shows sequentially learned 7 binary classification tasks, the multitask learning capability is greatly improved by *self-orgnization*.

There is another question comes after that, whether or not should the next task inherit the scale parameters (α and β) of the previous one. According to our experiments, see Figure 7, we observed randomly reinitialize the scale parameters achieves a slight better performance versus freeze these scale parameters between two continuous tasks. Therefore, there is no need to memorize the previous scale parameters.

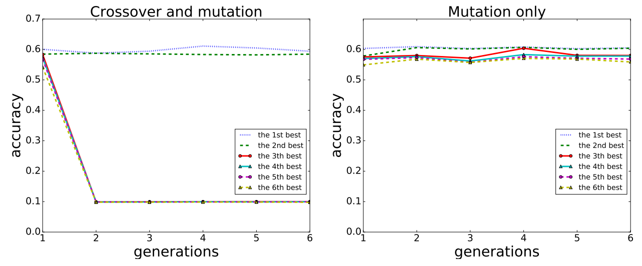

Figure 8: **Crossover and mutation vs Mutation only** When we use crossover and mutation together, many competitors lose their ability to learn.

Crossover and mutation vs Mutation only

We investigate whether the crossover strategy will benefit the network performance. We conduct the following experiment. Randomly generating a seed network to evolve for CIFAR-10, and the training is terminated at 3000 steps (batch size is 256) for each competitors. Six competitors for a generation, and with a total evolution of 6 generations, We ranked each generation of competitors and eventually got Firgure.8. It can be observed that crossover might cause the network performance fluctuation, and mutation only sustains equivalent performance. Therefore, we have omitted the crossover procedure in the evolutionary algorithm.

CIFAR10, SVHN and CIFAR100 SVHN dataset is a real-world digit recognition dataset consisting of photos of house numbers in Google Street View images. The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. For CIFAR-100, the total number of images and image size is same as in CIFAR-10, but it includes 100 classes containing 600 images each. There

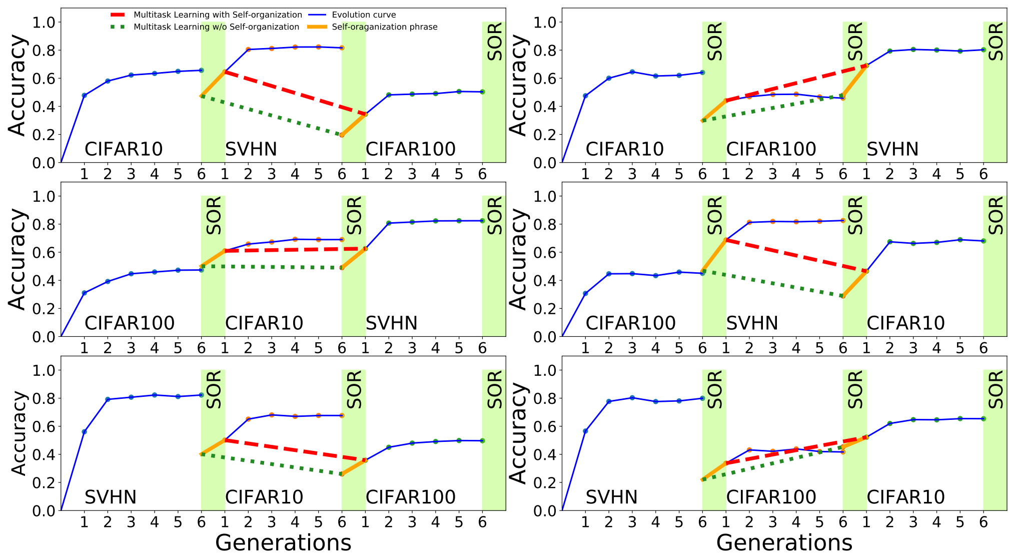

Figure 9: **Self-organization for multitask learning on CIFAR-10, SVHN and CIFAR-100 dataset.**

are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a"coarse" label (the superclass to which it belongs).

For the multi-classification experiments, we perform two 10-classification tasks on both the CIFAR-10 and SVHN, and a 20-classification task on CIFAR-100. Also, we stop training the network at 6 generations ($batchsize = 256$) for each competitors. Six competitors per generation, with a total evolution of 6 generations, slope still as the fitness criterion for network performance evaluation. The *self-organization* helps boosting their transfer capability, brings about 17.2% and 15.8% higher accuracy, when transfers from CIFAR-10 to SVHN, and then from SVHN to CIFAR-100. The convergence accuracy is 79%, 88% and 60% at 6 generations, see the top-left diagram in Figure 9.

Then we tested all the 6 combinations of the order of different tasks. *Self-organization* scheme increases the transfer performance, for example, brings about 11.0% and 13.6% higher accuracy, when transfer from CIFAR-100 to CIFAR-10, and then from CIFAR-10 to SVHN. The convergent accuracy is 59%, 75% and 87% at 6 generations, please see the mid-left diagram in Figure 9.

The total amount of computation can be represent the scale of the network to a certain extent, Figure 10 demonstrates that the SMART is task-oriented rather than ever-expanding, as the computational curve rise and fall according to different task complexity.

Conclusion and future work

It is well known that, manually build dedicated deep neural network structures for multi-tasking learning is infeasible. Our earlier and on-going studies on the multitask learning through soft-organization conducted in relatively small scale applications with a limited variety is introduced. Different

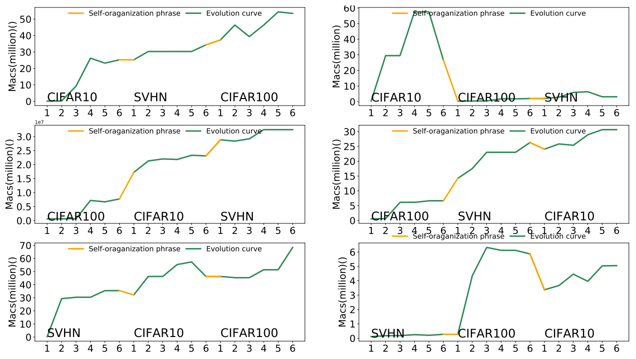

Figure 10: **Changes in the amount of computation for multitask learning on CIFAR-10, SVHN and CIFAR-100 dataset.**

tasks learn to use the same set of modules but different effectiveness can help shed light on how tasks are related.

SMART contains good expressive set of modules. The optimization in SMART is much easier than other deep learning approaches, since it operates in the granularity of module-level, therefore involves significantly reduced parameters. Over generations, modules with respect to new features, are added to the network incrementally, it can potentially achieving specfic-to-general multitask learning capability through evolution. Meanwhile, weight-based soft-pruning and L2 similarity measurement can be applied to eliminate the redundant modules to prevent the endless expansion of the network architecture. SMART helps identify a set of generalizable modules that are assembled in different ways for different tasks. Still, there are several issues to be addressed, follow-up works including extend it to many applications that lend themselves to the multitask approach.

References

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2016. Designing neural network architectures using reinforcement learning.

Bilen, H., and Vedaldi, A. 2017. Universal representations:the missing link between faces, text, planktons, and cat breeds.

Caruana, R. 1998. *Multitask learning*. Springer US.

Chen, B.; Wu, H.; Mo, W.; Chattopadhyay, I.; and Lipson, H. 2018. Autostacker: A compositional evolutionary learning system.

Devin, C.; Gupta, A.; Darrell, T.; Abbeel, P.; and Levine, S. 2017. Learning modular neural network policies for multi-task and multi-robot transfer. In *IEEE International Conference on Robotics and Automation*, 2169–2176.

Dong, D.; Wu, H.; He, W.; Yu, D.; and Wang, H. 2015. Multi-task learning for multiple language translation. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, 1723–1732.

Fernando, C.; Banarse, D.; Blundell, C.; Zwols, Y.; Ha, D.; Rusu, A. A.; Pritzel, A.; and Wierstra, D. 2017. Pathnet: Evolution channels gradient descent in super neural networks.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. 1923–1933.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. 770–778.

Huang, Z.; Li, J.; Siniscalchi, S. M.; Chen, I.; Wu, J.; and Lee, C. H. 2015. Rapid adaptation for deep neural networks through multi-task learning.

Iandola, F. N.; Moskewicz, M. W.; Ashraf, K.; Han, S.; Dally, W. J.; and Keutzer, K. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement learning with unsupervised auxiliary tasks.

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; and Simonyan, K. 2017. Population based training of neural networks.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; and Grabskabarwinska, A. 2016. Overcoming catastrophic forgetting in neural networks. *Proc Natl Acad Sci U S A* 114(13):3521–3526.

Liang, J.; Meyerson, E.; and Miikkulainen, R. 2018. Evolutionary architecture search for deep multitask networks.

Meyerson, E., and Miikkulainen, R. 2017. Beyond shared hierarchies: Deep multitask learning through soft layer ordering.

Miikkulainen, R.; Liang, J.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Shahrzad, H.; Navruzyan, A.; and Duffy, N. 2017. Evolving deep neural networks.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2018. Continual lifelong learning with neural networks: A review.

Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y. L.; Tan, J.; Le, Q.; and Kurakin, A. 2017. Large-scale evolution of image classifiers.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *Computer Science*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015a. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, 1–9.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015b. Rethinking the inception architecture for computer vision. *Computer Science* 2818–2826.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning.

Xie, S.; Girshick, R.; Dollar, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 5987–5995.

Xu, C.; Tao, D.; and Xu, C. 2013. A survey on multi-view learning. *arXiv: Learning*.

Zhang, Y., and Yang, Q. 2017. A survey on multi-task learning.