Survey on Differentiable Architecture Search

Abstract

Neural Architecture Search (NAS) has drawn an increasing interest today. Despite the success of recent approaches, most existing methods including reinforcement learning and neuro-evolution cannot be directly applied to large scale problems because of their prohibitive computational complexity. One-shot algorithms provide a fast solution in finding effective network architectures but suffered from large memory and computing overheads in jointly training a super-network and searching for an optimal architecture. While there are admittedly many caveats, this survey will introduce recent attempts in reducing the memory usage for such algorithms, in particular, on Differentiable Architecture Search (DARTS) (Liu, Simonyan, Yang, 2018).

1. Introduction

DARTS converts the operation selection into weighting a fixed set of operations while building a super-net. This introduces continuous architecture parameters making the entire framework differentiable to architecture hyperparameters. However, these results in multiple issues:

1) Suffered from large memory and computing overhead in jointly training a super-network and searching for an optimal architecture
2) Overfits and thus does not work robustly for some new tasks as skip-connect operations will dominate the architecture parameters (Problem of Stability)
3) Operations with more parameters may not have the chance to express desired function since those with less have already done the job (Co-adaption Problem)
4) Punishes underperforming operations by lowering their architecture parameter, and they will get smaller loss gradients (Matthew Effect)
5) No positive correlation between validation and test accuracy resulting in architectures optimized on proxy tasks not guaranteed to be optimal on the target task

The issues engendered with the introduction of DARTS has spurred researches and its variants, which includes ProxylessNAS, SNAS, PDARTS and PC-DARTS. More recently, notably, MergeNAS has been released. While researches aimed to solve the different issues, most converges on the idea of tackling the large memory usage. This is largely due to the correlation between increased memory usage and test accuracy on the target task. Further, we argue that in order for an increase in performance, the memory usage paradox should first be addressed.

2. Addressing Memory Inefficiency

In the searched architecture, N operations and their outputs need to be stored at each node (i.e., each network layer), leading to N × memory used. Due to the large memory usage, to fit into a GPU, batch size must be reduced during search. This slows down the search speed and may deteriorate search stability and accuracy. It is also important to recognise the advantage of one operation over another is not significant unless more training data are involved in a mini batch to reduce the uncertainty in updating the parameters of network weights and architectures. Many approaches have since surfaced and largely lies in 2 categories, pruning and sampling.

2.1 Sampling

PC-DARTS (Xu, Xie, Zhang, Chen, Qi, Tian, Xiong, 2020) suggest usage of partial channel connections by defining a channel sampling mask which assigns 1 to selected channels and 0 to masked ones. Selected channels are sent into mixed computation of operations, while the masked channels bypass these operations and directly copied to the output. The proportion of selected channels is controlled by a hyper-parameter K which determines the trade-off between architecture search accuracy and efficiency. This reduces the memory overhead of computing by K times, allowing for a larger batch size for architecture search. However, since architecture parameters are optimized by randomly sampled channels across iterations, the optimal connectivity could be unstable, causing fluctuation in the searched architecture. PC-DARTS introduced edge normalization to multiply the architecture parameters with normalized coefficient, which is shared throughout the training process, after the architecture search.

ProxylessNAS (Cai, Zhu, Song, 2019) introduce architecture parameters and transforms the path weights to binary gates. This ensures one path of activation is active in memory at run-time and the memory requirement of training the over-parameterized network is reduced to training a compact model. To train the binarized architecture parameters, at every iteration, two paths are sampled and thus reducing the number of candidates to 2. The intuition is that if a path is the best choice at a particular position, it should be the better choice when solely compared to any other path. The architecture parameters of these two sampled paths are then updated using the gradients calculated before rescaling them to keep the path weights of unsampled paths unchanged. As such, in each update step, one of the sampled path weights increases and the other sampled path weight decreases while all others keep unchanged.

PARSEC (Casale, Gordon, Fusi, 2019) introduce a prior $p(\alpha|\pi)$ on the inputs and operations choices where hyper-parameters π are the probabilities corresponding to the different choices.

Child networks are randomly sampled from the specified architecture distribution and thus reducing the problem to be inferring the probability distribution $p(\alpha|\pi)$ of high-performing architectures. The prior hyperparameters π can then be optimized by Bayes Monte Carlo procedure. Since each sample from this distribution is a child architecture from the architecture search space, all computations are done on the child networks sampled hence has the same memory requirements of training a single architecture.

GDAS (Dong, Yi, 2019) similarly suggest learning a hyperparameter A to encode the sampling distribution on the basis that an architecture consists of many neural cells which are sampled from the search space. Specifically, one transformation function from a discrete probability distribution is sampled. However, since sampling hinders back-propagation, Gumbel-Max trick is used to reformulate the objective function and provide an efficient way to draw samples from the discrete probability distribution to optimise the hyperparameter A. Softmax function is used to relax the argmax function so to make the objective function differentiable. Further, the architecture parameter is designed as a one-hot vector which requires only the argmax of the architecture parameters to be calculated. Similarly, during the backward procedure, only that gradient is backpropagated. This saves computation time and reduce the GPU memory cost. Within one training batch, a different cell will produce a different architecture parameter, which is shared for each training example.

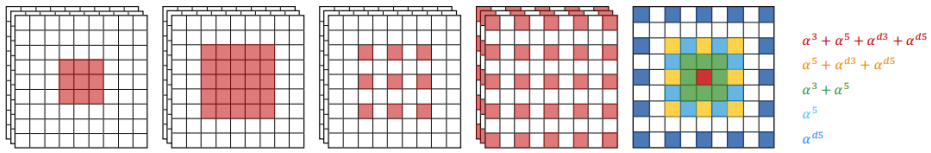
2.2 Pruning

Apart from operation search space, GPU memory usage is proportional to the depth of searched networks. At the initial stage, the search network is relatively shallow, but the operation space is large. PDARTS (Chen, Xie, Wu, Qi, 2019) suggest that after each stage, architecture parameters are learned and the scores of candidate operations on each connection are ranked. While pruning the less important operations, which are assigned with lower weights during the previous stage, the new operation set can be ensured to be smaller if not equivalent in size. This reduces the amount of memory which allows the depth of the searched architecture to be increased by stacking more cells while concurrently approximating the operation space.

2.3 Merging

MergeNAS (Wand, Chao, Yan, Yang, Hu, Sun, 2020) introduce the idea of expanding weight sharing strategy by also sharing weights among the convolutions with different kernel sizes and dilation rates on the same edge. They propose that convolutions can be merged into one operation when they share weights, same for parameter-less operations. This can be done by

constructing a tensor with a large kernel size as the merged weights. All convolutions' weights can then be derived from the single tensor using mask matrices to denote the receptive field of different parametric operations.



(a) 3×3 convolution (b) 5×5 convolution (c) 3×3 dilated conv. (d) 5×5 dilated conv. (e) mask matrix $\tilde{M}$

Figure 1: (a)-(d) is the equivalent mask $\mathbf{M}^o$ of the receptive field for different convolutions (red denotes non-zero weights). (e) is the weighted average mask $\tilde{\mathbf{M}}$. Different color indicate various combinations of architecture parameters α. In the micro search space of DARTS, there are four kinds of convolutions (see Section 4).

By sharing weights among convolutions, MergeNAS can reduce not only the number of parameters but also the amount of computation. To further reduce the memory cost for MergeNAS, skip connect operation and average pooling are merge with normal convolutions into the single convolution based on their propositions.

3. Results

Algorithms	Test Errors (CIFAR-10)	Search Cost (GPU Days)	Test Errors (Imagenet)	Search Cost (GPU Days)
DARTS	2.83	4	26.9	-
PC-DARTS	2.57	0.1	24.2	3.8
ProxylessNAS	2.08	*	25.4	8.3
PARSEC	2.81	1	26.0	-
GDAS	2.93	0.21	26.0	-
PDARTS	2.50	0.3	24.4	-
MergeNAS	2.68	0.6	*	*
DropNAS	2.26	0.6	23.4	-

* represents not published

- represents architecture transferred from CIFAR10

4. Analysis

Evident from the results on CIFAR-10, most variants of DARTS discussed have searched for the architecture in significantly lesser computational cost as compared to the vanilla version. Some variants like PC-DARTS and ProxylessNAS have went on to search for the optimal architecture directly on the Imagenet dataset without the need of a proxy task which was

previously not computationally feasible with DARTS. It is notable that the variants have also achieved better performance on both CIFAR-10 and Imagenet albeit at a lower cost. This could be due to the ability to search on the target test without the need for a proxy task which eliminates the problem aforementioned, which is the lack of correlation between proxy and target tasks.

It is also imperative to note that most sampling methods do act as a regularizer, making it less biased in selecting operations. Specifically, this regularizes the preference of a weight-free operation over a weight-equipped one further addressing the rest of the issue brought up previously by only allowing for a small number of channels to pass through. Previously, weight-free operations often accumulate larger weights at the beginning, making it difficult for the weight-equipped operations to beat them after they have been well trained. Sampling allocate more time for the operations to optimise their weights to produce more consistent outputs.

In particular, DropNAS (Hong, Li, Zhang, Tang, Wang, Li, Yong), which achieves one of the better performances, introduces Grouped Operation Dropout to tackle the issue. Specifically, during the search stage, operations are randomly and independently sampled, outputs are zeroed to make their weights not updated during back-propagation. The operations are first categorised into convolutional operations and one non-parameterized before fixing the probability of each operation to be dropped. During the backpropagation period, the weights of the dropped operations will receive no gradient. This attempts to unifies the sampling strategies by setting the probability of dropping an operation as a hyperparameter and optimizing it.

5. Conclusion

While there are ongoing studies on improving the performance of such one-shot algorithms and joint optimization of the architecture and model's hyperparameters, the huge memory usage should also be addressed as they are more often than not correlated or is a roadblock to other advancements. This also ensures the possibility of extending to larger-scale problems and incorporate into AutoML pipeline to democratise machine learning. Open sourced frameworks while currently support some of the one-shot algorithms often fail to keep up with the upcoming variants of the algorithms or fail to prove its reproducibility.

References

Hanxiao Liu, Karen Simonyan, Yiming Yang, DARTS: Differentiable Architecture Search, ICLR 2019

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, Hongkai Xiong, PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search, ICLR 2020

Han Cai, Ligeng Zhu, Song Han, ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR 2019

Francesco Paolo Casale, Jonathan Gordon, Nicolo Fusi, Probabilistic Neural Architecture Search, 2019

Xuanyi Dong, Yi Yang, Searching for A Robust Neural Architecture in Four GPU Hours, 2019

Xin Chen, Lingxi Xie, Jun Wu, Qi Tian, Progressive DARTS: Bridging the Optimization Gap for NAS in the Wild, 2019

Xiaoxing Wang, Chao Xue, Junchi Yan, Xiaokang Yang, Yonggang Hu and Kewei Sun, MergeNAS: Merge Operations into One for Differentiable Architecture Search, 2020

Weijun Hong, Guilin Li, Weinan Zhang, Ruiming Tang, Yunhe Wang, Zhenguo Li and Yong Yu, 2020