Neural Architecture Search Using Automated Planning

Felipe Roque Tasoniero
School of Technology
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre - Brazil
felipe.tasoniero@edu.pucrs.br

Abstract

Due to recent interest in designing neural architecture without human assistance, there has been a great effort to find methods that generate high-performing CNN architecture with less computational time processing. Traditional techniques as Reinforcement Learning and Evolutionary Algorithms are widely used in Neural Architecture Search (NAS) research, but they are costly to compute. In this paper, we propose to explore a based RL algorithm, known as MetaQNN, to automatically generate CNN architectures and use a penalize configuration to reduce computational cost.

Introduction

It's known that state-of-the-art neural network architectures require a lot of effort of human experts to be discovery. Recently, Neural Architecture Search research field has been widely investigated due to its capacity to automatically find neural network architectures that perform as well as the ones designed manually (Elsken, Metzen, and Hutter 2018).

Two main techniques that are usually used to solve NAS problems: Reinforcement Learning (RL) (Zoph and Le 2016), and Evolutionary Algorithms (EA) (Miikkulainen et al. 2019). In Evolutionary Algorithms, each neural network structure is encoded as a string, and random mutations and recombinations of the strings are performed during the search process; each string is then trained and evaluated on a validation set, and the top-performing models generate "children" (Liu et al. 2018). When using reinforcement learning, the agent performs a sequence of actions, which specifies the structure of the model; this model is then trained, and its validation performance is returned as the reward, which is used to update the controller (Liu et al. 2018).

Although Evolutionary Algorithms has shown significantly progress in designing high-performing neural architectures, RL algorithms are still considered a go-to for NAS. An approach to solve NAS problems is using a Q-learning algorithm, a type of reinforcement learning.

Even though the Q-learning algorithm is less costly than EA algorithms, they are still a problem for stand-alone researchers to train it. A solution to this problem was the implementation of MetaQNN (Baker et al. 2016). It was shown that MetaQNN took 8-10 days to complete the training for each dataset using 10 GPUs.

In this paper, we intend to explore the use of MetaQNN to search a high-performing CNN architecture for the MNIST dataset, aiming the reduction of GPUs/hour cost by penalizing the reward function for large models and slow forward passes.

Technical Approach

The automated process of CNN architecture selection through MetaQNN, summarized in Figure 1, consists of incorporating Q-learning agent, ϵ-greedy strategy, and experience replay.

The Q-learning algorithm is used as an agent to find optimal paths as a Markov Decision Process (MDP) in a finite-horizon environment. The environment here is restricted to a discrete and finite space, as well as action space. So the agent's goal is to maximize the total expected reward over all possible trajectories.

A set of different types of layers is selected by the agent via the ϵ-greedy strategy until it reaches a termination state. The layers that we allow are: Convolution (C), Fully Connected (FC), Pooling (P), Global Average Pooling (GAP), and Softmax (SM). The ϵ-greedy strategy is used in a way that the agent begins in an exploration phase and slowly starts moving to the exploitation phase. Then, the experience replay provides to the agent a memory of its past explored path and rewards.

Related Work

Recently, works related to the NAS research field that makes use of RL algorithms has proven its capability to generate high-performing neural architectures to deal with specifics machine learning and deep learning problems. Due to the known problem of processing time consume related to NAS, many search models were developed to deal with it.

In the search space models, two types of search have showed a high accuracy in neural architectures generated. One of them is the Cell Search Space, where an agent selects cells based on a reward, where these cells are composed of many feature layers, and then these cells are stacked to generate a new neural architecture (Tan et al. 2019). The other one is the Global Search Space, where an agent selects lay-

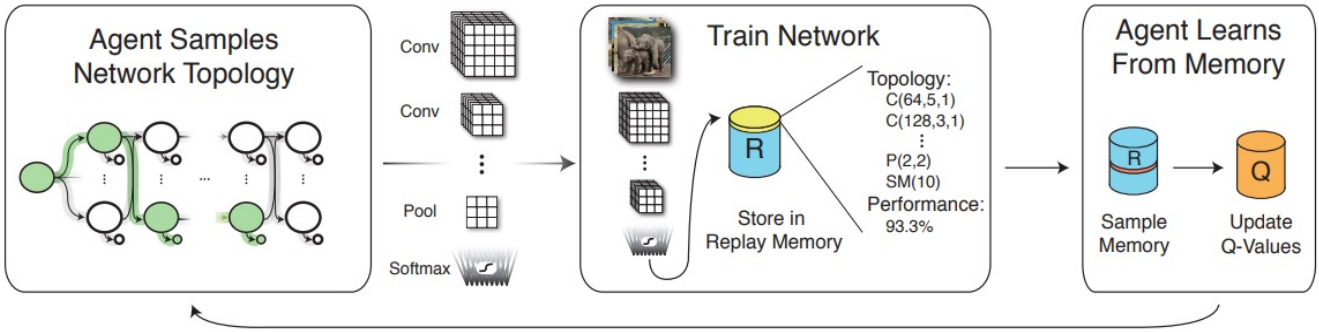

Figure 1: Automated process for Neural Architecture Search using Q-learing.

ers based on a reward to create a new neural architecture (Zoph and Le 2016).

Although the capacity of these models in generating high-performing CNN architectures, they are very similar considering the computational cost. To solve this problem, an RL algorithm called MetaQNN was proposed to significantly reduce the computational time consuming (Baker et al. 2016). Some improvements to this model were proposed too by *Baker et all* (Baker et al. 2017).

Project Management

Considering that we have about five weeks for our deadline, and the very time consuming to train and validate the NAS algorithm, a schedule for accomplishing this work is:

- Week 1: Investigate the MetaQNN algorithm and implementing the penalize configuration.
- Week 2: Testing the algorithm and make modifications if necessary.
- Week 3 and 4: Training and validating the model using penalize configuration.
- Week 5: Analysis of results and writing the final paper.

Conclusion

In this proposal, we aim to explore the use of MetaQNN for NAS due to its less necessity of GPUs/hour usage comparing to traditional RL algorithms for this task. Considering that, we propose to use whey to penalize the agent's reward. We expected that implementing the penalizing configuration could reduce the amount of computing time processing as well as maintaining the high-performing accuracy on generated architectures.

References

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

Baker, B.; Gupta, O.; Raskar, R.; and Naik, N. 2017. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2018. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 19–34.

Miikkulainen, R.; Liang, J.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Shahrzad, H.; Navruzyan, A.; Duffy, N.; et al. 2019. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier. 293–312.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. V. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2820–2828.

Zoph, B., and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.