Neural Architecture Search with Reinforcement Learning

Pavel Khakimov

1 Question 1

-Outline the main idea of the paper.
The modern tendency is do not manually extract features from data but build a more complex NN for doing that. For doing that authors use two NNs(Neural Network). The first NN is a RNN(Recurrent Neural Network) which generates the special sequence. This sequence could be interpreted as a layers scheme of a new NN. According to this scheme autors build a new NN and measure its performance. The objective is to find a special sequence, such that the NN based on this special sequence will outperform the other NNs. Error rate of each NN is interpreted as a negative reward for an RL(Reinforcement Learning) agent. This interpretation allows to use RL notations and searching solution algorithms. Authors use a policy gradient method.

-What are the requirements to apply the same approach on a problem of practical interest to you?
This approach performs well on a special types of problems, which could be solved using convolution architectures. I think there are 2 main requirements:

- convolution architecture of a solving problem
- many hidden layers

If you work only with fully-connected NNs I think it is still possible to apply this approach. But you need to make some modification in RNN model. Instead of 5 numbers for representation of each hidden layer of a new NN, now there need only 1 number (size of hidden layer) and maybe one more extra parameter like a type of activation. We should simplify a generating sequence by eliminating: Filter Height, Filter Width, Stride Height, Stride Width, Number of Filters could be interpreted as Size of Hidden Layer.

2 Question 2

-Is it possible to extend the model to also perform search over hyperparameters like learning rate, batch size, etc?
Hyperparameters like a learning rate or batch size are parameters of a current model of NN. This paper describes a searching method in a multidimensional space of *child networks*. Each point of this space is a sequence, which corresponds to a particular NN model with fixed layers scheme. Authors call this RNN as **the controller**, it generates *child networks*, but there is another module, which estimate performance of a given *child network*, I'll call it **the estimator**. Searching among hyperparameters like learning rate or batch size is an objective of **the estimator**, not an objective of **the controller**.

-If so, argue what's the limitation, otherwise propose an extension.
We need somehow compare different *child networks*. Each *child network* has its own potential or limit - how successful chosen architecture is. *child network* is tuned well if its best hyperparameters like a learning rate and batch size are found. For bad architectures - tuning doesn't affect much, but for good architectures tuning could be crucial. This paper's approach uses the same number of epochs and the same set of hyperparameters for each *child network*. It means that we could miss some successful networks, because our set of hyperparameters could not be optimum for those successful networks.

But this is still possible to implement additional searching for each *child network* in its own hyperparameters space. In this case **the estimator** becomes more complex and computational cost of a whole model will significantly raise up.

3 Question 3

-The controller is stochastic. Is this a blessing or a curse? Explain your reasoning.
Definitely blessing. Because we are searching in a multidimensional space. It is still infinite even if sequences we are working with consist of natural numbers. It is not possible to check out every available point of this space.

-Can it be made deterministic and how, if not — why?
No. It can't be made deterministic. Each *child network* achieves an accuracy R. This R uses as a reward signal in RL notation. Since this reward signal R is non-differentiable, there is only policy gradient method for iteratively updating parameters of **the controller**.