Exploring Randomly Wired Neural Networks for Image Recognition

Purvanshi Mehta

October 1, 2019

1 Review

Four possible paper extensions have been provided to the paper.

- Nearly all of the popular NAS papers manually design the architecture framework. They deploy either bayesian optimization, evolutionary algorithms, reinforcement learning, or differentiable architecture search to find the blocks to be integrated into the framework. In this paper the authors discuss how other approaches are limited due to the biases in the network construction. Here also the relaxation on constraints is minimal. Each node still does relu-convolution-BN, not changing the input size. The processing is manually divided into stages with progressively downsampled representations. In the end, the only thing that it is free to change is **the arrangement of residual connections**.

  A possible solution to this problem could be to have dynamic edges. Although the network is randomly generating network space using graph algorithms, the network wirings are still fixed at training. The **edge connections could be dynamically chosen**. This can be done by **assigning a weight to every edge** after fixing the initial number of edges. The weights of the edges are relevant to our current network according to a certain threshold T. **The edge($a \rightarrow b$) weight updation** would be a **function of the weight of a -** w_a **and the partial derivative of** $\frac{\partial w_b}{\partial L}$**.** Which intuitively can be seen as the importance of transitioning from a to b with respect to loss. In such a way the *unrestricted search* may discover more efficient approaches.

- Another thing to note in the present NAS structures is the fixation of depth. One possible naive solution would be to find the loss at temporary output layers within the network. Another approach is to observe if just the skip connections are useful and the weights inside the blocks are sparse. One more sophisticated way to think about it is to **form this as an optimization problem. The optimization would be on the number of layers and number of nodes in each layer.** Our aim

is to find the minimum number of layers and nodes which could solve the problem in hand. Obviously more time is required to formulate the mathematical equations.

- One of the applications of such randomly generated network architecture could be **continual learning**. In one line - making one path learn certain tasks and another path learning N different tasks. Obviously during formalizing this approach we will have to look to the problems of continual learning and how to handle those using network search - like catastrophic forgetting. We can do so by putting constraints on our architecture search space.

- The NAS technique proposed in the paper is very **inefficient in terms of computation**. The graph generated in the paper could be divided into sub graphs for parallel training. There are algorithms in graph theory which divided the models used in the paper - ER model, BA model and WS model into sub graphs which could lead to parallel training.