MGU - hyper-parameter self-learning using Multi Gated Units in deep neural graphs

XCS229ii

Authors: Andrei DAMIAN, Laurentiu PICIU

Problem definition

Training deep neural networks is not as straightforward as just simply adding computational elements within the network. Finding the optimal architecture, i.e. hyperparameter optimization, is an active research topic and is also a major time consumer in practice. The architectures are generally designed by adding specific feature processing methods such as feature normalization, dropout, activations, skip or residual connections, etc. usually following linear transformations no matter their complexity (fully connected, convolutions, etc). Thus, due to the fact that the actual challenge consists in finding the optimal architecture, one should iterate through multiple combinations (i.e. grid search, AutoML, etc) of the various possible processing methods. This either heuristic or exhaustive search approach has the objective of finding the optimal “formula” between this feature processors (e.g. normalization after the activations or before, adding or not additional skip/residual connections, etc), thus generating the optimal computational module (a computational module is the computational layer with all the added processors).

This half-automated architectural design approach has two major drawbacks. The first obvious issue is that the search space grows exponentially with the number of feature processors used within a module as well as with the number of options for each processor. Although there are proposed methods for avoiding exhaustive search and even advanced ones such as AutoML this drawback is far from solved from the perspective of computational efficiency and “green” Deep Learning. The second issue is related to the “copycat” approach within each individual section of a neural graph due to the fact that the same module is repeated a certain number of times within the section. This architectural constraint implies a clear limitation to the class of hypotheses. Considering the possibility that the optimal hypothesis might need unique structures for each individual module throughout the whole deep neural graph, this ideal hypothesis will never be found.

Solution proposal

In this work, we propose a novel architecture (called Multi Gated Unit - MGU) that encapsulates any kind of neural computational layer (linear, convolutional, etc.) that is able to self-learn the optimal modules configuration for the encapsulated layer through self-gating mechanisms. We argue that the proposed architecture drastically eliminates the need for grid search approaches required to find the optimal architecture of a deep neural graph.

Using just one end-to-end optimization process, the deep neural graph using MGUs will learn *a unique individual module structure* for all modules within the graph. Beside finding a potentially ideal architecture for the proposed task we dramatically reduce the carbon footprint of the whole model generation process by eliminating the need for multiple end-to-end optimizations.

Datasets, input-outputs and applicability

Finally, we argue that the proposed architecture has applicability in any area or problem that can be modelled with deep neural graphs. In this work, we will quantify the gain of using MGUs in performance (accuracy-metrics) and needed time to find the optimal architecture on specific tasks such as image classification (MNIST, CIFAR-10) and recommendations generation. For each task, we will benchmark the single *self-hyper-parametrized* neural network with multiple neural networks resulting from a grid-search process.

Challenges and further proposed research

We will also explore the idea of *prune-convert-optimize* the MGUs following the end-to-end training in order to optimize the inference time and energy consumption. The challenge is to find a heuristic approach of determining the gating thresholds that allow the decision of eliminating gating mechanisms such as for the obvious case when a gate is always opened or closed no matter the input. This heuristic will allow us to “copy” the structure and weights from the MGU module to the new “pruned” modules and thus eliminate the need for the multiple self-gating operations.