Computationally efficient dynamic neural networks

Abstract for the general public

Driven by the amazing and constantly improving capabilities of the state-of-the-art deep neural networks, the field of deep learning has been experiencing a massive growth in interests from scientists and industry in the last decade. This progress was made possible by the algorithmical and model design enhancements as well as the increasing computational power of modern General Purpose Graphics Processing Units (GPGPU). A straightforward scaling-up of the architecture frequently results in improved performance, thus the average size and computational cost of state-of-the-art models are constantly increasing. The energy and equipment cost of training or evaluating such models is often prohibitive for a majority of research institutions and small companies, limiting their ability to compete.

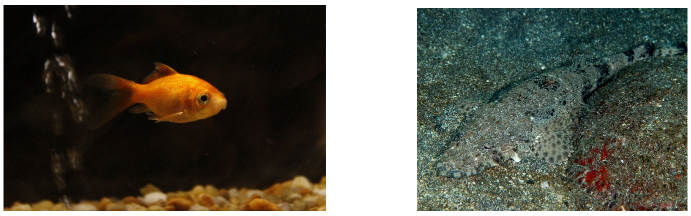

Figure 1: Two samples of the same class with significantly different classification difficulty.

The aim of this project is to study, develop and enhance current deep learning methods for obtaining models with computationally efficient inference and training. Particularly, we will focus on dynamic neural networks, a group of methods in which the computational graph is conditioned on the current input. Fig. 1 shows an exemplary pair of image inputs. The first image is clearly easier to classify than the second, and thus it possibly requires less computation than the second image to get a correct answer from the model. This also corresponds to how the biological inspiration for deep neural networks – the human brain – works.

Existing dynamic network models usually focus only on a single aspect of conditional computation. Early-exiting models attach additional classification heads to intermediate layers of the network and then use them to return an early answer and save computation. Mixture of Experts (MoE) models select the appropriate expert modules to execute for the current input. Other models adjust the computation for each part of the input separately. While some of these were shown to considerably reduce the amount of compute without performance degradation, the relationship between these methods was never fully explored. Futhermore, MoE models were usually used to preserve the computation cost while simultaneously scaling up the number of parameters, a related but different goal than reducing that cost.

We intend to bridge those gaps by proposing a series of analyses, experiments, and novel method designs. First, we will propose a novel method inspired by boosting that adjusts the compute allocated to each token. This will allow us to reduce the amount of computation, and, crucially, to rank the examples by the amount of compute needed for every one of them. Intuitively, this ranking would tell us which examples are considered easy or hard for that particular method. After calculating similar example rankings for other types of conditional computation methods, we can ask the following question: do different types of dynamic networks rank the examples similarly? Assuming a we have a static model with Rectified Linear Units (ReLU) we could also ask: how many activations in the intermediate layers of those models are zeroed-out and what is their distribution? Answering those questions could lead to a more computationally efficient method and thus more accessible research environment, prolonged battery lives and reduced greenhouse gas emissions.