Adaptive Filters and Aggregator Fusion for Efficient Graph Convolutions

Shyam A. Tailor [1] Felix L. Opolka [1] Pietro Liò [1] Nicholas D. Lane [1 2]

The development of hardware-efficient techniques is key to the deployment of deep learning. We have seen the deployment of convolutional neural networks (CNNs) to enable previously unthinkable applications at the edge, due to innovation at the hardware and algorithmic level. Recently, we have seen research efforts aimed at tackling the deployment challenges associated with language models in both the data center and at the edge.

Research into efficiency often focuses on hardware-software co-design: the development of techniques to enable the software to take better advantage of the hardware, and vice-versa. There are several facets to this field: usage of low precision arithmetic (quantization) is one example. Another common approach is pruning, where we remove weights from the model. A key area, however, is *designing neural network architectures with efficiency as a design goal*. This requires domain-specific knowledge, and approaches may not be directly transferable from one domain to another. As neural network architecture designers, we cannot simply rely on improvements in hardware to make our proposals viable for real-world deployment.

Graph Neural Networks (GNNs) have emerged as an effective way to build models over arbitrarily structured data, with successes in many different application domains. For example, recent work has shown that they can be applied to physical simulations. There has also been success on computer vision tasks: GNNs can deliver high performance on point cloud data and for feature matching across images. Code analysis is another application domain where GNNs have found success. Our work aims to enable these applications, and many more, by investigating approaches to design GNN architectures that enable us to obtain high accuracy without requiring large increases in memory consumption or latency.

We propose a new GNN architecture, Efficient Graph Convolution (*EGC*), which does not require trading accuracy for runtime memory or latency reductions. Our proposal's memory consumption is linear in the number of vertices in the graph ($\mathcal{O}(V)$)—not the number of edges ($\mathcal{O}(E)$), which is the case for many popular approaches that achieve high accuracy. We achieve our results through a novel adaptive filtering approach, which has no correspondence to attention-based mechanisms that have become popular in other areas of deep learning, but result in $\mathcal{O}(E)$ memory consumption. Our architecture is a *drop-in replacement* on a wide variety of tasks.

Our approach has an intuitive interpretation of giving each node in the graph its own weight matrix. In contrast, existing state-of-the-art architectures rely on dynamically modulating the information each node receives from each of its neighbours on a per-edge basis, which is the cause of the growth in memory consumption.

Hardware considerations carefully inform our architecture design. EGC is well suited to existing accelerator designs, while offering substantially better accuracy than existing approaches. Further to this, we propose a novel technique, *aggregator fusion*, to further accelerate our architecture at training and inference time. This technique is motivated by our observation that the sparse operations required by GNNs are memory-bound, hence we can cheaply perform additional computation in times when we would otherwise be waiting for data to arrive from memory. Aggregator fusion is a technique that can be widely adopted by the community, and is not exclusive to EGC.

We provide a rigorous evaluation of our architecture across 5 large graph datasets covering both transductive and inductive use-cases. Application domains ranging from citation graphs through to code analysis and molecular property prediction are covered, and it is demonstrated that our approach consistently achieves better results than strong baselines which represent the state-of-the-art in the field.

To ensure our work is reproducible, we release all code, hyperparameters, and pre-trained models associated with this work at: `https://github.com/shyam196/egc`.

EGC has already been upstreamed to PyTorch Geometric, the most popular GNN library; our aggregator fusion technique is expected to be adopted in the future.

[1]Department of Computer Science & Technology, University of Cambridge, United Kingdom [2]Samsung AI Center, Cambridge, United Kingdom. Correspondence to: Shyam A. Tailor <sat62@cam.ac.uk>.