1. *Related Work*

**Sparse CNNs Accelerators on FPGA.** Works on FPGA-based acceleration of sparse CNNs can be categorized by different pruning granularity levels: (i) specific on structured pruned models, (ii) specific on unstructured pruned models, (iii) specific on mixed-grained pruned models. Zhu et al improves the ASIC-based SCNN and implement the hardware design on FPGA. This work presents a zero-skipping dataflow for feature, whose zero elements are generated by coarse-grained channel-wise pruning and filter-wise pruning. Although such method raises computing efficiency, zero elements in temporal result still occupy storage resource. Lu et al. propose a weight-oriented dataflow with tile look-up table on FPGA. By using element-matrix multiplication as the core operation, Lu et al. accelerates fine-grained pruned CNNs with little decoding cost. However, our 2s-AGCN model differs from above basic convolutional workloads in that each element in feature is generated by graph matrix multiplication. Such weight-oriented design ignores useless convolutional computation but cannot skip corresponding graph computation. Li et al. work on PCONV, a mixed-grained pruning method where structure filter-dropping and unstructured pruning inside kernels are combined. With weight-stationary dataflow designed on FPGA, Li et al. improve the computing efficiency by 14.7%~44%. However, this work still occupies storage space for huge scale of zero data like Lu et al, and its simple hardware structure cannot tackle four different workloads in our task.

**GCN Accelerators on FPGA**. Many works on accelerating large graph’s GCNs based on FPGA are presented in recent time. AWB-GCN combines offline software averaging and runtime hardware workloads balancing on several large graph datasets. Zhang (ASAP GCN) et al. partition input data into smaller segments, then perform graph sparsification and node re-ordering for computation reduction and data locality. Hy-GCN splits GCNs workloads into *Aggregation* and *Combination* phases. Different hardware structures and dataflows are designed for two phases respectively. To sum up, above works focus on: (i) leveraging and expanding graph adjacency matrix’s sparsity, (ii) avoiding irregularity and randomness of data distribution in graph computation, (iii) keeping balanced workloads between PEs or computing phases, via offline and online ways. Unfortunately, graph in skeleton-based GCNs for action recognition models is dense and unchangeable. The data sparsity is embedded in temporal feature and pruned weights, not the graph. Moreover, action recognition GCNs behave not only like CNNs, but also like graph processing, leading to graph-specific hardware design requirements. Therefore, current specialized architectures on CNNs and GCNs cannot efficiently perform target algorithms since they just address one of the two sides.

While there exist many GCNs accelerators on large graph in social media and graph analytics, few works have been proposed to accelerate skeleton-based GCNs for action recognition. ST-GCN, a smaller GCNs model for action recognition, is accelerated by Ding et al. on FPGA. However, their work falls short on more complex action recognition GCNs for: (i) they only apply quantization on model, does not prune or optimize ST-GCN from the view of software-hardware co-design; (ii) Ding et al. compress human skeleton graph into CSC format, while skeleton relationship matrix in some models is learnable and dense. (iii) their hardware design is established on sparse matrix-vector multiplication (SpMV) units, but only skeleton adjacent matrix is compressed. Data sparsity is not thoroughly utilized in their work; (iv) although the proposed single PE design improves DSP efficiency, its throughput performance does not meet the requirement of expected application scenario.