Main Contribution:

1. A united new pruned method for skeleton-based GCN action recognition models.
2. A Block RAM based run-time sparse vector compress storage design is proposed, which reduces the utility of BRAM in order to map all layers on FPGA.
3. A layer-by-layer pipeline architecture for 2s-AGCN model which achieves xx compared with GPU.

Introduction

Action recognition models has great potential to apply in hospital, school and nurse house.

Skeleton-Based GCN action recognition models achieve SOTA performance in the field.

Skeleton-extracting models are mature and can work on video stream, GCN model can work with such algorithm in an end-to-end way.

Problems:

1. Huge size of parameter and computation: graph computation, spatial convolution, temporal convolution and shortcut structure;
2. Conventional pruning method is not suitable for GCN models;
3. Computing speed gap between fronted models and GCN models, needs higher throughput;
4. Low-power application scenario demands customized hardware platforms.

Software solution:

1. Modifying graph-spatial convolution’s data-flow into a more pruned-friendly way.
2. Using graph-spatial convolution pruned method to apply structured pruning on temporal convolution.
3. Temporal convolution is further pruned in a fine-grained method via delicate sampling scheme.
4. Skip some input frames for input human skeleton information existing some repetition.

Hardware solution:

1. Layer-by-layer pipeline architecture to accelerate 2s-AGCN model.
2. Using low-power FPGA as hardware platform.
3. Layer-by-layer design’s demanding storage resource is beyond FPGA, previous zero-aware works is useful for sparse convolutions via skipping useless computation, however it cannot reduce the on-chip storage needs. Based on such dilemma proposing BRAM-based sparse vector storage design.
4. Dynamic data dispatch to decrease the DSP used.

Background

1. The introduction to 2s-AGCN model.
2. The dataflow of the GCN.
3. Clipping unimportant computation and validate its influence on performance.

GCN Prune Method

1. Reorganize the data-flow of graph computation, where lies a united pruning chance. One zero in filter can reduce a vector-matrix operation in current group, when extended into the same in-channel on all filters, all graph computation in current channel can be skipped.
2. Temporal convolution’s result is the input feature for graph-spatial convolution. The united prune method actually drops all data in some channels of the feature, which means the corresponding filters in temporal convolution can be pruned for their computation make no sense to the united-pruned model.
3. Temporal convolution’s kernel size is 9x1, working on temporal dimension with 9 input frames. This convolution operation can be seen as sampling: if the weight in filter is zero, then the data in this frame will not be sampled and computed. Based on this observation, a fine-grained pruned method is proposed. Several sampling schemes are dedicated designed and settled in the temporal convolutional filters left, these schemes produce zero in filters in repetition, which in further decrease the cost of mapping them on hardware.

GCN Hardware Design

The whole architecture is consisted of graph-spatial convolution module, temporal convolution module and sparse storage module between every convolution function part.

Graph-spatial convolution module:

1. Only non-zero weight is stored in ROM.
2. PEs are settled along output channel, which can cooperate with pruned method.
3. Feature pool for shortcut path.

Temporal convolution module:

1. PEs are settled along input channel, which can achieve pipeline-balance with graph-spatial convolution module.
2. Only non-zero weight is stored in ROM, and their pruning scheme masks are also stored to choose valid input feature.
3. Dynamic dispatch works between DSPs in one PE to reduce the resource occupation.

Sparse storage:

Idea: multi-bank BRAM storage, average sparse degree of vectors in each layer.

1. Encoding part.
2. Multi-bank sparse storage based on BRAM.
3. Decoding part.

Experiment

1. Our Pruning method’s performance compared with structured prune, unstructured on (1) accuracy, (2) compress ratio, (3) hardware-friendly feature
2. Dynamic scheduling’s effect on (1) delay, (2) DSP used, (3) utilization ratio of DSPs, (4) compared with other sparse accelerator’s DSP utilization ratio.
3. Run-time sparse storage’s compare between dense storage and CSC format on (1) size, (2) encoding-decoding cost and delay, (3) memory access operation’s data-flow.
4. Architecture performance compared with GPU, ST-GCN accelerator, balanced layer-by-layer pipeline.