Res2Net Accelerator

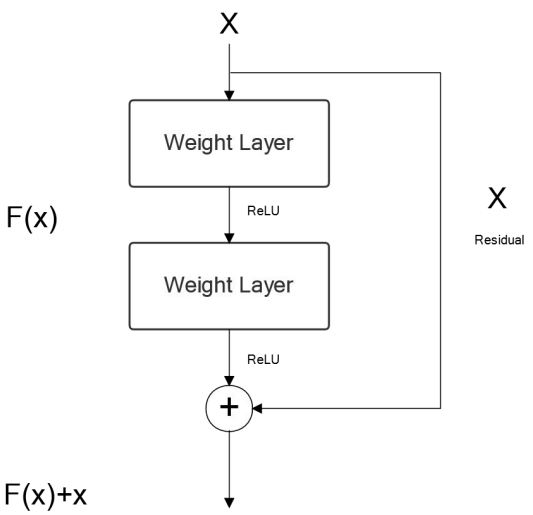
Introduction

Res2Net is a novel residual structure. The main difference between Res2Net and ResNet lies in their network structures. ResNet uses residual blocks to address the vanishing gradient problem in deep neural networks, while Res2Net constructs hierarchical residual connections within a single residual bottleneck to represent multi-scale features and increase the receptive field range of each network layer. Res2Net's hierarchical residual connections can better extract multi-scale features, thereby enhancing the network's representational capacity. Moreover, Res2Net is compatible with existing ResNet models, so it can be easily applied to current deep learning tasks.

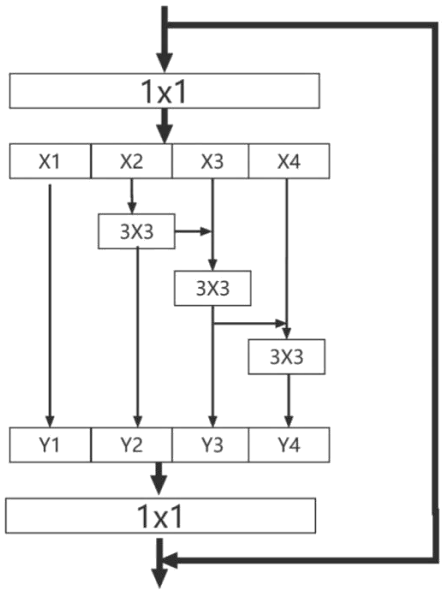
The Res2Net Bottleneck includes a 1x1 convolution to upscale the input data, several 3x3 convolutions to hierarchically extract input features, and a final 1x1 convolution to adjust the input data to the required number of channels for the output.

Res2Net has shown good results in convolutional networks, achieving better effects while reducing computational load.

However, unlike ResNet, Res2Net uses group convolution and 1x1 convolution layers to reduce the computational load, which may result in the matrices that are used for residual connections being of different sizes. This requires additional down sampling and resizing through a convolutional network.



ResNet Diagram



Res2Net Diagram

Description of Implementation

The specific heterogeneous software implementation will be on the platform:

Firstly, we will base our implementation on Res2Net's official Pytorch. I implemented the parallel Group Convolution, but did not implement the parallel residual connections of Res2Net.

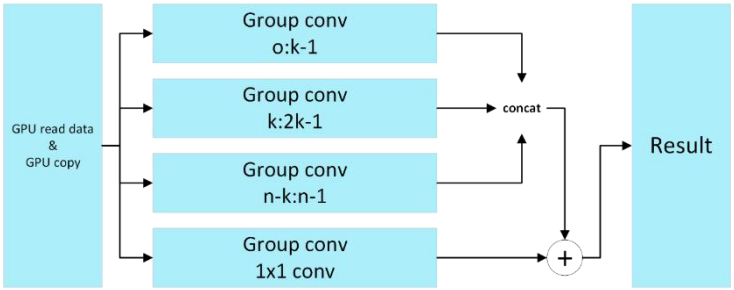
Pytorch has implemented the CPU and GPU parallelization of Group Convolution, but it cannot parallelize the residual connection addition part (including the downsampling operation).

In Res2Net, the following two parallel implementations are required:

1. Group Convolution: In PyTorch, Group Convolution is a convolution operation that divides the input channels into several groups, where the channels within each group share weights, but the channels between different groups have independent weights. This can be achieved by setting the groups parameter, which is often used to reduce the number of parameters and computational complexity. In the implementation, the input data is first divided into groups, then an independent convolution kernel is applied to each group, and finally, the outputs of each group are concatenated to form the final output. Group Convolution is widely used in convolutional neural networks, especially when balancing computational efficiency and representational capability, significantly reducing the model's parameter quantity and improving training and inference efficiency.

2. Res2Net's Residual Connection: In PyTorch, the Res2Net residual block is an extended residual connection structure designed to enhance the neural network's feature expression capability. Unlike traditional ResNet, Res2Net introduces multiple sub-paths in the residual connection, each with its independent convolution branch, which are combined by shared weights. This design increases the network's width and improves feature extraction capability, thereby better capturing features of different scales and complexities. Res2Net residual blocks divide the input data into multiple branches, then fuse these branches with appropriate weights to achieve more powerful feature extraction and deeper network training.

We will implement group convolution, convolution, and residual connections.



Implementation illustration

Evaluation

We will take Res2Net's official Pytorch implementation as our baseline.

After analyzing PyTorch, we found that PyTorch has implemented parallel Group Convolution, but not parallel residual connections of Res2Net.

Therefore, the speedup compared to the serial version should be

$$\frac{S + T_1 + T_2}{S + \frac{T_1}{k_1} + \frac{T_2}{k_2}}$$

The speedup compared to Pytorch's parallel implementation version should be

$$\frac{S + \frac{T_1}{k_1} + T_2}{S + \frac{T_1}{k_1} + \frac{T_2}{k_2}}$$

T_1 is the time for GroupConv, T_2 is the time for GroupConv.

Reference

[1]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[2]Gao, Shang-Hua, et al. "Res2net: A new multi-scale backbone architecture." IEEE transactions on pattern analysis and machine intelligence 43.2 (2019): 652-662.