Branchy-GNN: A Device-Edge Co-Inference Framework for Efficient Point Cloud Processing

J. Shao, H. Zhang, Y. Mao and J. Zhang, "Branchy-GNN: A Device-Edge Co-Inference Framework for Efficient Point Cloud Processing," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 8488-8492

Advisor: Dr. Chih-Yu Wang

Presenter: Shao-Heng Chen

Date: March 23, 2022

Outline

1. Overview
2. Branchy-GNN framework
3. Cost-Overhead Tradeoff
4. Experiments
5. Future works

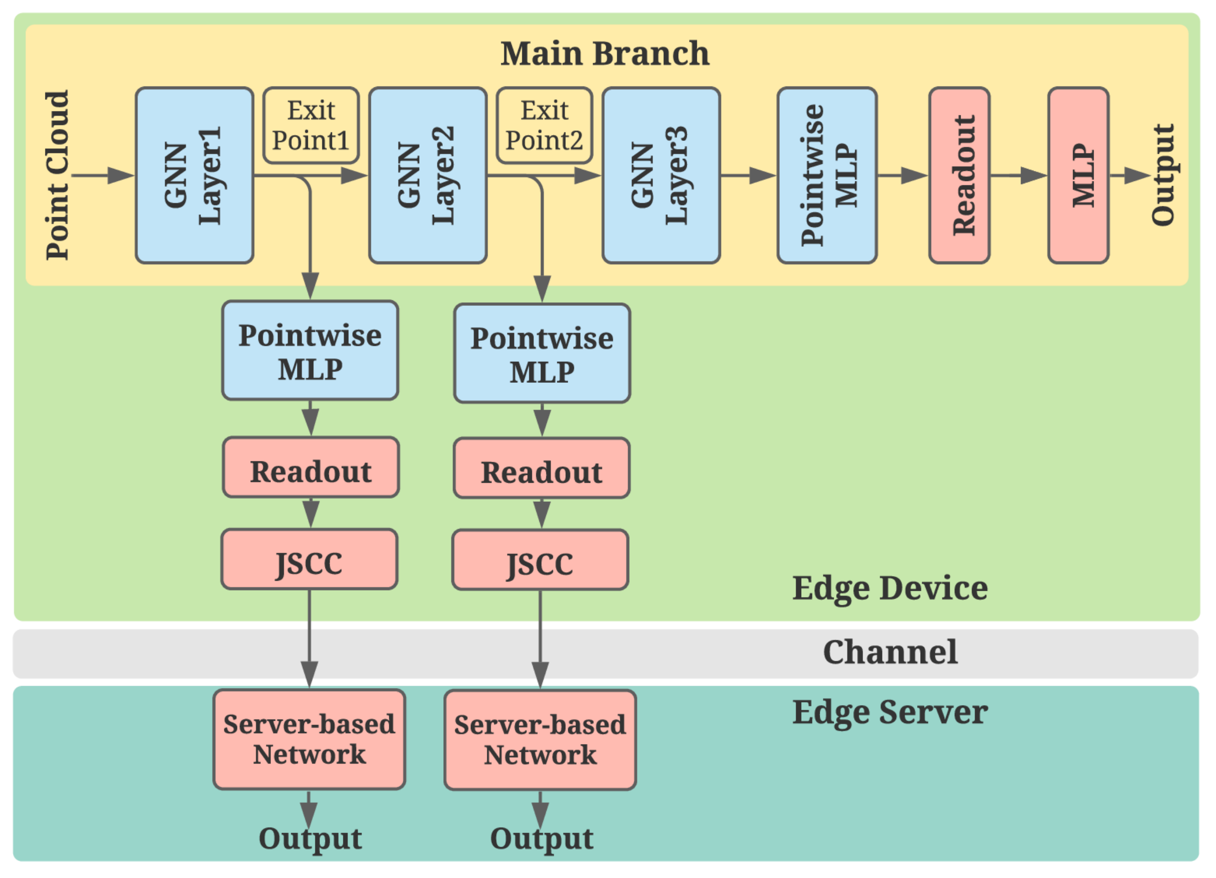
Overview

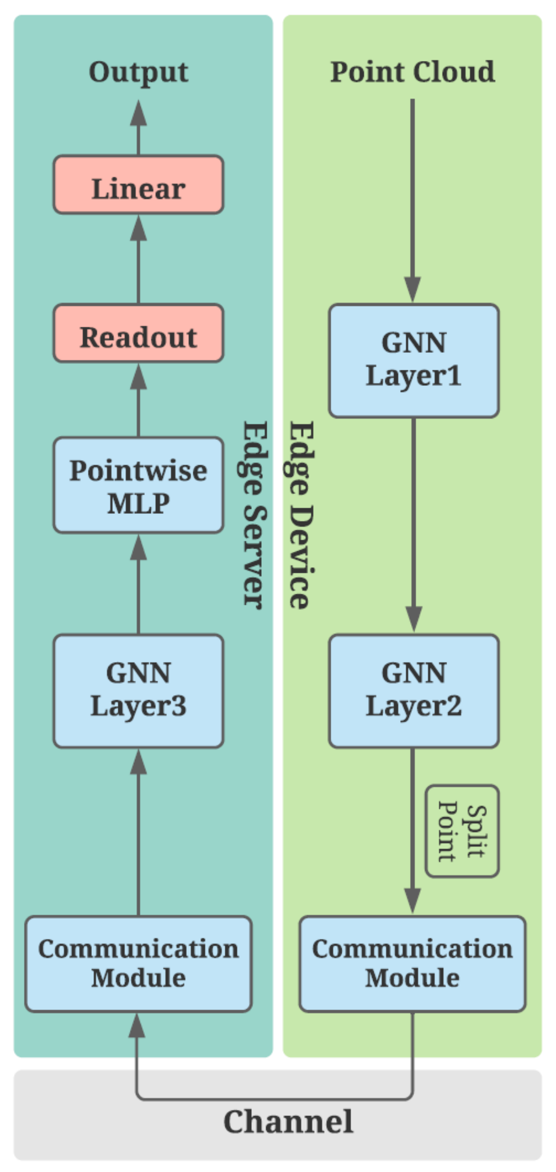
1. GNN vs. 3D-CNN?
   - Input: Point Set vs. Dense Array
   - Operation: PointNet vs. Convolution
   - Neighbor: varying # points vs. fixed # pixel
2. Why not model splitting?
   - more severe data amplification effect
3. What's the possible challenges?
   - data amplification, autoencoder/JSCC design

Branchy-GNN framework

– branch structure reduce computational cost

– JSCC coding scheme reduce comm. overhead





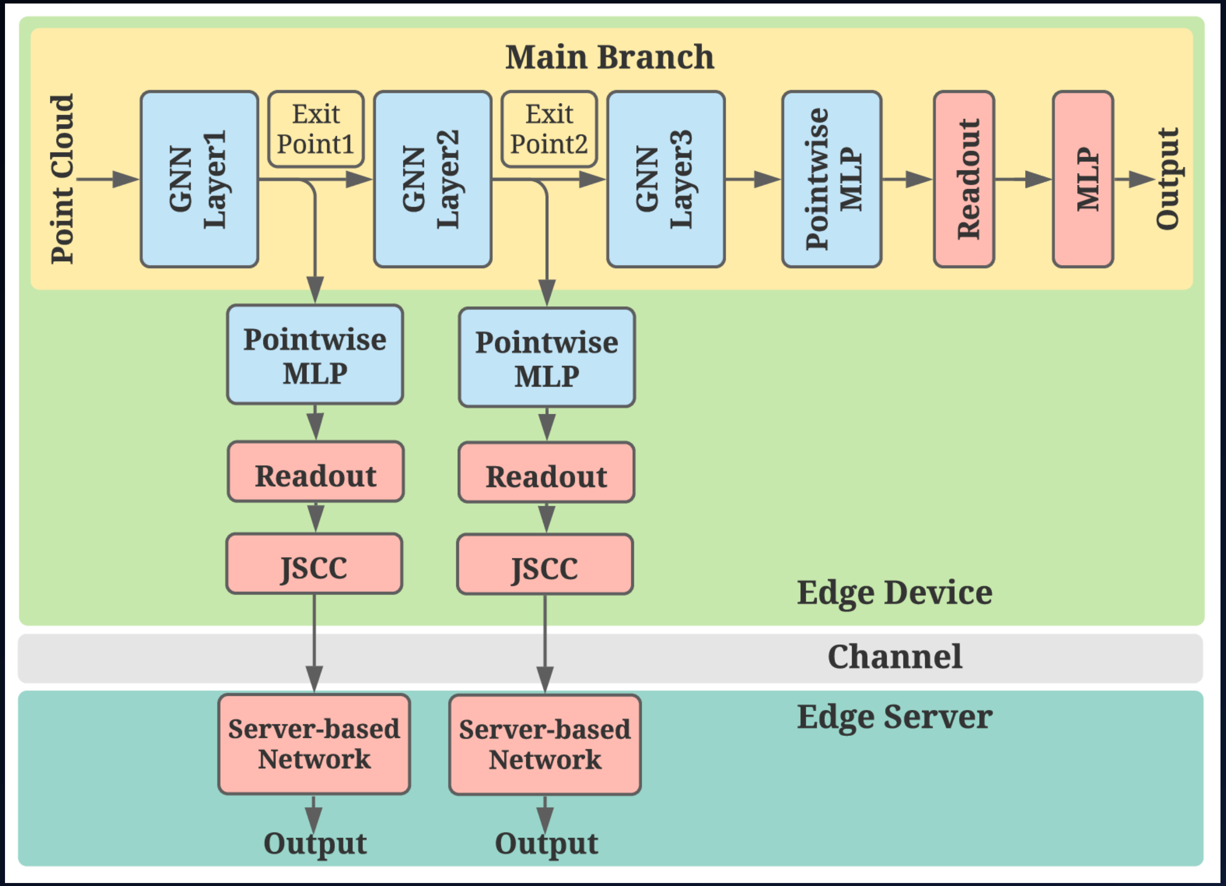
Branchy-GNN framework

(1) A point-wise MLP [1]

(2) A readout layer

(3) A Joint Source-Channel Coding (JSCC) module

(4) A server-based network



[1] C.R. Qi, H. Su, K. Mo, and L.J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 652– 660

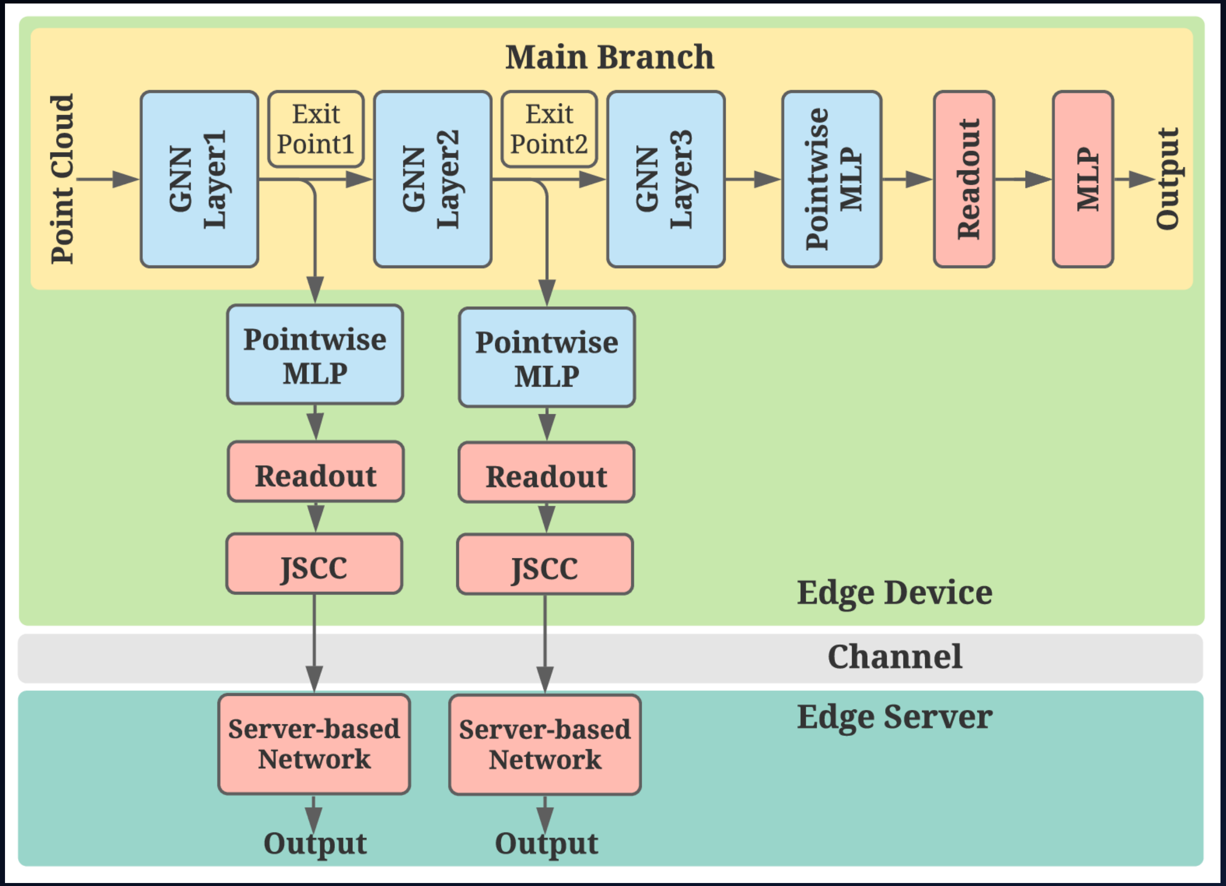
Branchy-GNN framework

(2) A readout layer [2]

readout function

$$s = \frac{1}{N}\sum_{i=1}^{N} x_i \,||\, max_{i=1}^{N} x_i$$

x_i *feature vector of the* i^{th} *point*
N *number of nodes*
$||$ *concatenation*



[2] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, and J.M. Solomon, "Dynamic graph cnn for learning on point clouds," ACM Transactions On Graphics, vol. 38, no. 5, pp. 1–12, 2019

Computation Cost-Overhead Tradeoff

– On-device computation latency vs. communication latency

– Computational cost determined by main branch GNN layers

– Communication overhead determined by JSCC output dimension

– Earlier exit means less on-device computation and higher communication overhead

Training Methodologies

– Complete training may cause slow convergence

(1) Separate training, first train the original GNN / main branch

(2) Then, fix the weights and train the multiple branches

Experimental settings[2]

– Shape recognition on ModelNet40 [3], 40 categories of 12311 images

– Main branch: DGCNN [2]

– AWGN channel with SNR = 20dB

– Edge device: Raspberry Pi3

– Edge Server: PC with RTX 2080Ti

[3] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.
[2] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, and J.M. Solomon, "Dynamic Graph CNN for learning on point clouds," *Acm Transactions On Graphics*, vol. 38, no. 5, pp. 1–12, 2019

Experimental results

– On-device computational cost vs. communication overhead

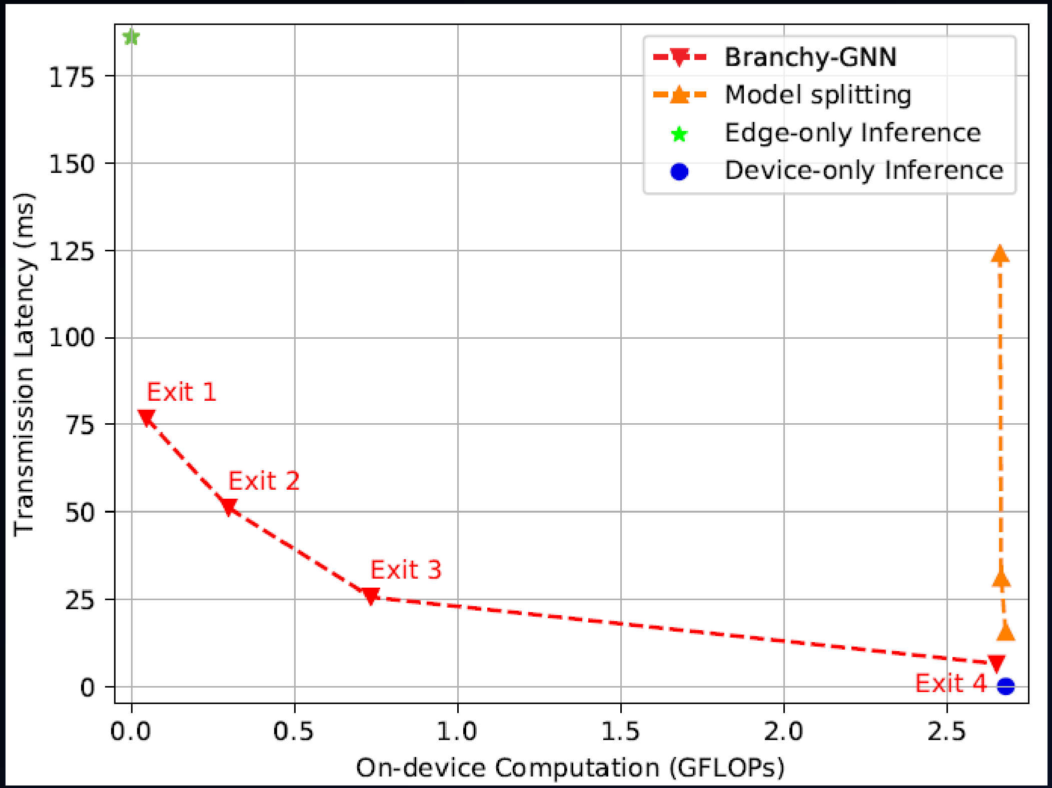


Fig. 2. On – device computation cost and communication latency

– channel BW = 10kHz (optimal)

– baselines: (1) device-only,
(2) edge-only,
(3) model splitting

– input size: 512 (points)

– output size: 1536, 1024, 512, 128

Experimental results

– End-to-end inference latency

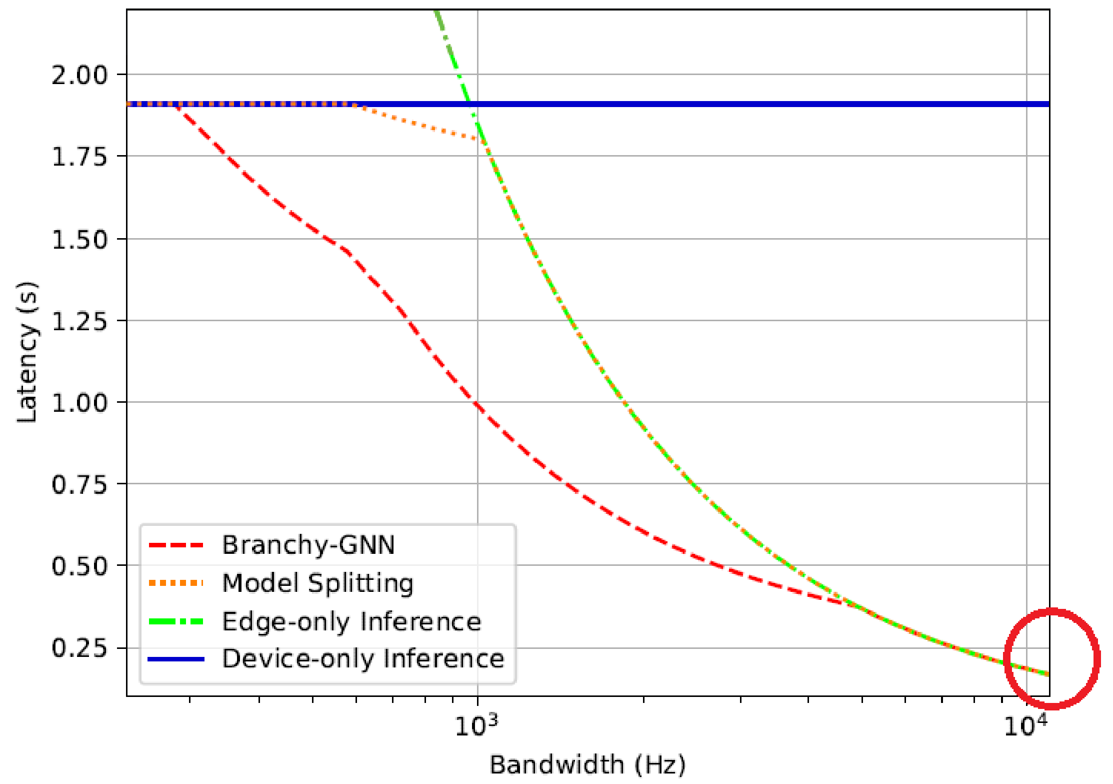


Fig. 3. Edge inference latency in different bandwidth

– meets min. edge inference latency with limited BW (0.3 ~ 5 kHz)

optimal

Experimental results

– Verify robustness: train at SNR = 20 dB, test at SNR = 18~25 dB

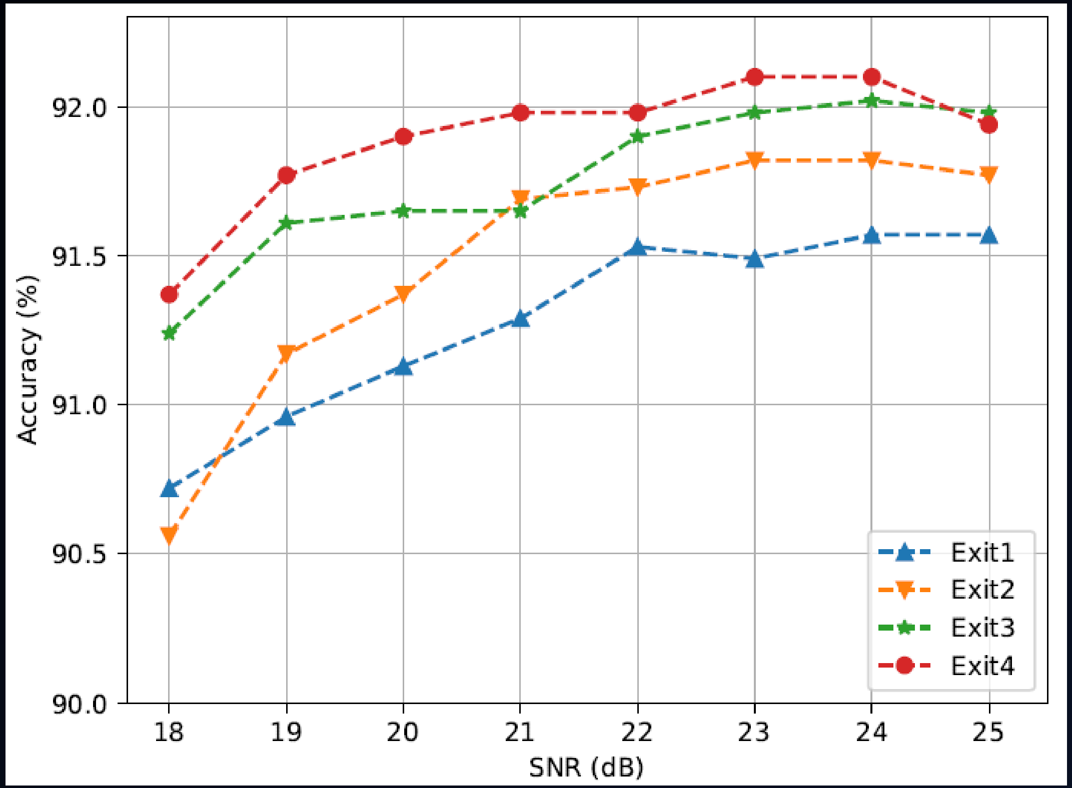


Fig. 4. Classification accuracy under different channel conditions

– input size: 512 (points)

– output size: 1536, 1024, 512, 128

Future Works

– Maybe try some of the ideas on YOLO_CFAR case