Mitigating I/O Bottlenecks in LiDAR Pipelines by Directly Merging Neural Decompression and Semantic Segmentation

***Marquez, E., M. Faykus, O. Odetoye, Smith, M. Calhoun, J.**
Holcombe Department of Electrical and Computer Engineering, Clemson University

Introduction

The fast growth of high-resolution LiDAR data creates a significant I/O bottleneck in large-scale analysis and high-performance computing (HPC). This is an issue with multi-step pipelines where LiDAR point clouds are used sequentially and required to be rewritten as files between steps.

This work introduces a novel, unified framework that eliminates this costly redundant storage by directly integrating a RENO-based neural autoencoder with Point Transformer V3 (PTV3) as the segmentation backbone. We designed this end-to-end model to leverage the feature-rich latent space of the RENO decoder as direct input for the segmentation network, bypassing the need to write and read point cloud files.

This approach enables semantic analysis directly on compressed data, significantly reducing I/O overhead and end-to-end processing time while aiming to maintain segmentation accuracy.

Background

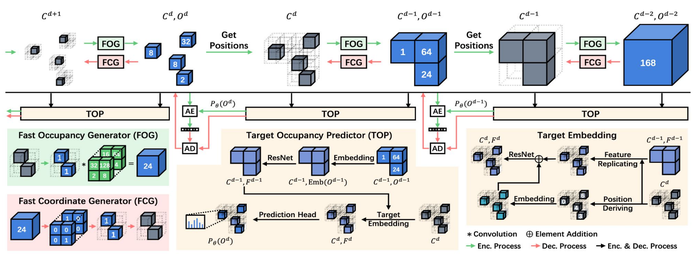

Fig 1: RENO Architecture: A lightweight neural codec for compressing 3D LiDAR point clouds in real time. [1]

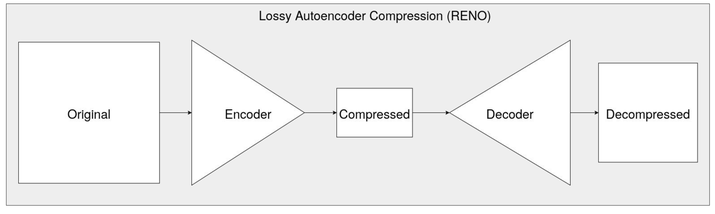

Fig 1.5: RENO Architecture pipeline

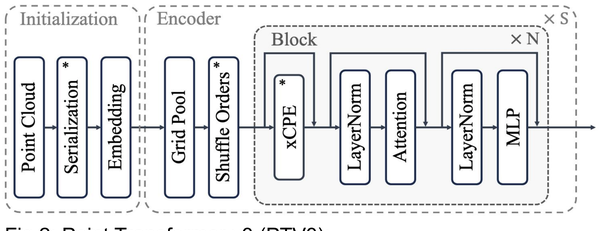

Fig 2: Point Transformer v3 (PTV3):
An architecture for processing 3D point cloud data. The core idea is to prioritize simplicity and efficiency, which allows the model to scale up more effectively. [2]

Background cont.

Fig 4: PTV3D simplified pipeline. The general flow of how data is processed in this work. A point cloud file is inference by a segmentation model, which outputs a class mapping for the point cloud.

Methodology

Stage One:
- Compress and Decompress the GOOSE dataset using RENO at different compression levels
- Train PTV3D on Decompressed LiDAR point clouds and reconstructed label mapping and measure mIoU.

Stage Two:
- Code combined architecture with RENO decoder implemented before PTV3 backbone to directly feed decompressed tensors.
- Train merged RENO-PTV3 architecture and measure mIoU

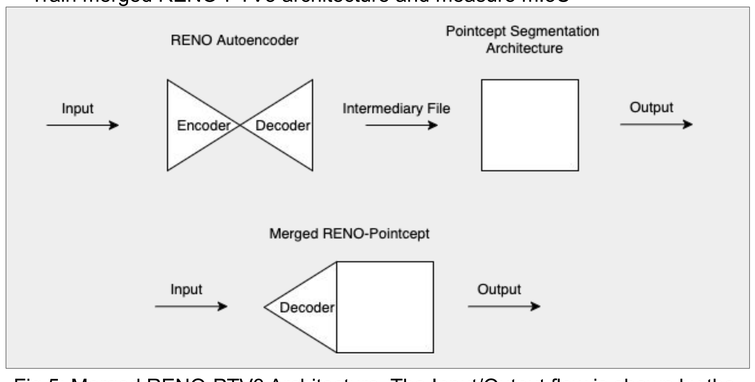

Fig 5: Merged RENO-PTV3 Architecture. The Input/Output flow is shown by the arrows. The top half is the original pipeline where intermediary files is necessary. The bottom half is the merged pipeline, reducing I/O usage.

Challenges

- RENO is a lossy neural compressor. This outputs varrying tensors, which requires reshaping of labels to match the outputs. This is accomplished using a K-Nearest-Neighbors algorithm during dataloading.

- GOOSE uses point cloud representations where each point is (x, y, z, intensity). RENO overwrites intensity, so we must match the original intensity to the new points after decompression.

Results

Fig 6: Left most GOOSE point cloud is uncompressed [1], middle point cloud is Q64 RENO Compression, right most point cloud is Q512 RENO Compression.

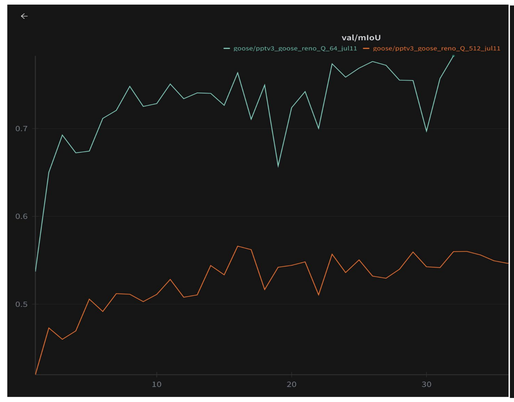

Fig 7: mIoU for Q64 (Green) and Q512 (Orange) Compression

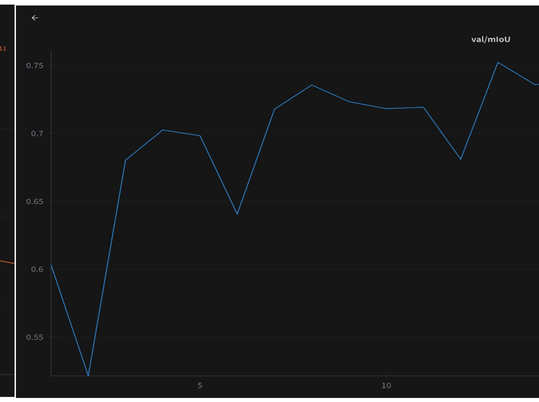

Fig 8: mIoU for merged RENO-PTV3 Q64 (Blue)

$$\mathrm{IoU}_{\mathrm{class}} = \frac{\mathrm{Prediction}_{\mathrm{class}} \cap \mathrm{GroundTruth}_{\mathrm{class}}}{\mathrm{Prediction}_{\mathrm{class}} \cup \mathrm{GroundTruth}_{\mathrm{class}}}$$

Fig 8: IoU accuracy metric used to profile accuracy

Conclusion

- Directly using decompressed feature tensors rather than writing to intermediary files saves 29.9 GB for 13,076 point clouds. This means with a 20 HZ LiDAR system, we save 2.7 GB every minute the LiDAR is running, at no added cost.
- RENO compresses point clouds from 29.9 GB to 1.2 GB.
- Lossy compression of point clouds using RENO efficiently reduces the number of points while keeping relevant features for downstream tasks such as semantic segmentation.
- Lower RENO compression levels relate to a higher mIoU performance on PTV3 segmentation.

References

[1] https://arxiv.org/pdf/2503.12382
[2] https://arxiv.org/pdf/2312.10035
[3] https://goose-dataset.de

Acknowledgements

Thank you to the College of Engineering, Computing and Applied Sciences as well as Dr. Smith for her creative inquiry Machine Learning and Big Data. This work was supported in part by the Clemson University Creative Inquiry + Undergraduate Research Program. This research used in part resources on the Palmetto Cluster at Clemson University under National Science Foundation awards MRI 1228312, II NEW 1405767, MRI 1725573, and MRI 2018069. The views expressed in this article do not necessarily represent the views of NSF or the United States government.