Learned Point Cloud Compression for Classification

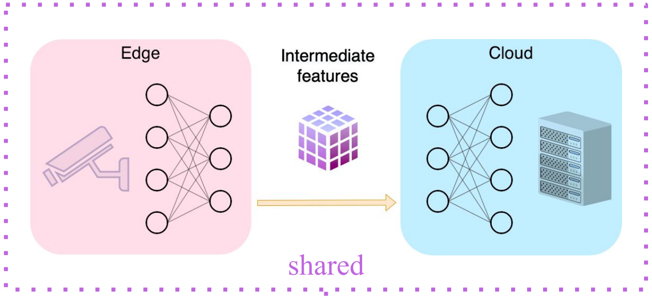
Mateen Ulhaq and Ivan V. Bajić

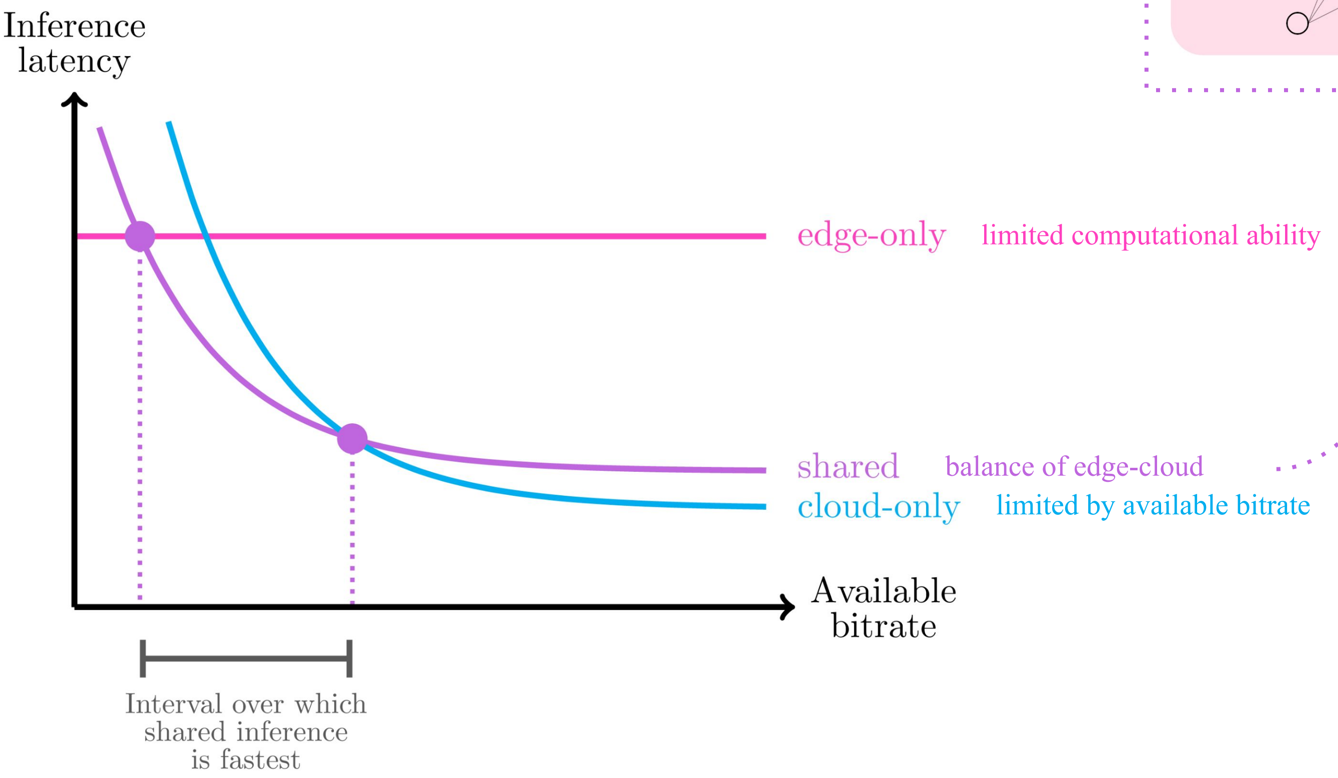
MMSP 2023 Poitiers, France

SIMON FRASER UNIVERSITY

https://github.com/multimedialabsfu/learned-point-cloud-compression-for-classification

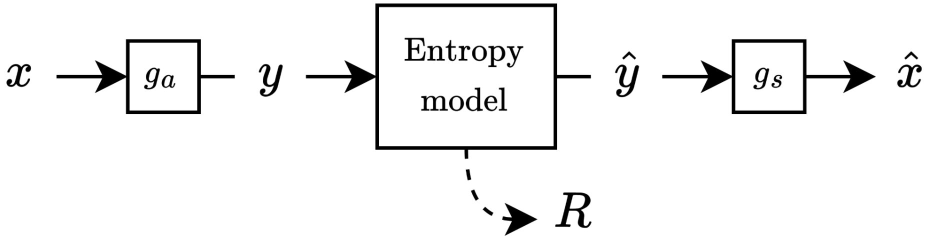
Motivation


Edge
Intermediate features
Cloud
shared


Inference latency
edge-only
limited computational ability
shared
balance of edge-cloud
cloud-only
limited by available bitrate
Available bitrate
Interval over which shared inference is fastest

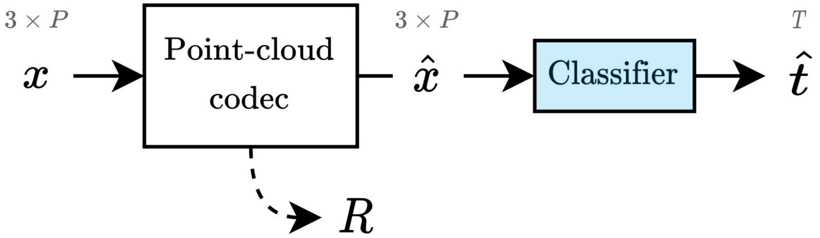
Learned compression

- Ballé *et al.* proposed a "factorized prior" entropy model where each channel of the quantized latent $\hat{\boldsymbol{y}}$ is encoded using a channel-specific distribution.
- Rate is $R = -\log_2 p_{\hat{\boldsymbol{y}}}(\hat{\boldsymbol{y}})$.
- Distortion is $D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ between the input $\boldsymbol{x}$ and reconstructed input $\hat{\boldsymbol{x}}$.
- Loss is $\mathcal{L} = R + \lambda \cdot D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ where λ is a trade-off hyperparameter.

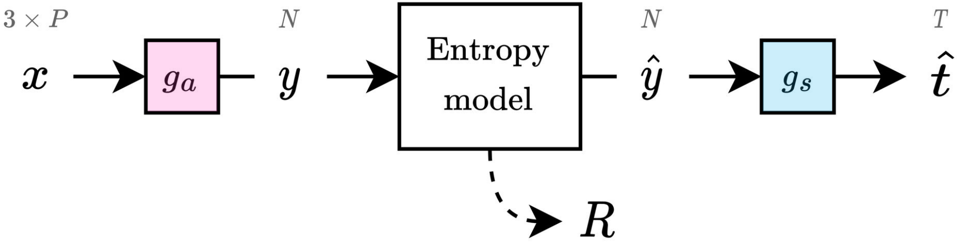


[Ballé *et al.*]: "Variational image compression with a scale hyperprior," *ICLR*, 2018.

Codec architectures



(a) input compression (for cloud-only inference)



(b) proposed (for shared edge-cloud inference)

Fig. 1. High-level comparison of codec architectures.

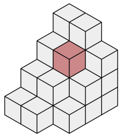
Input data formats

P × 3 matrix

$$[(x_1, y_1, z_1),$$
$$(x_2, y_2, z_2),$$
$$\dots$$
$$(x_P, y_P, z_P)]$$

Point list *PointNet, PointNet++*

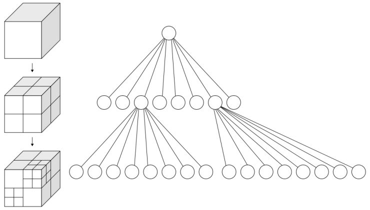
Very light MLP-based architectures.
No worthwhile canonical ordering of points.
Challenges: order-invariance, finding 3d metric structure aware operations.



https://commons.wikimedia.org/wiki/File:Voxels.svg

Voxel grid *VoxNet, 3D ShapeNet, 3D conv-based models*

$O(n^3)$ memory. Limits resolution: 1024 × 1024 × 1024 ⇒ 4 GB per float32 tensor!
3D convs are computationally heavy.
Empty space ⇒ wasted computation.

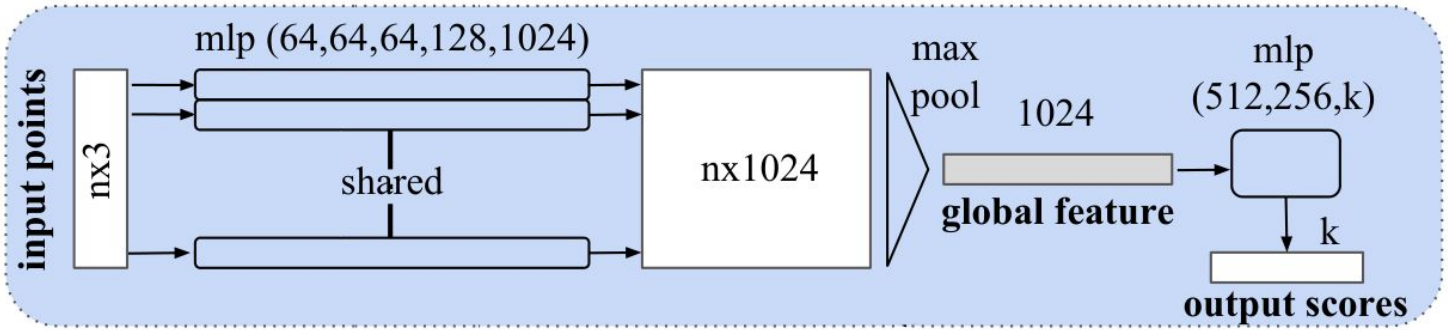


https://commons.wikimedia.org/wiki/File:Octree2.svg

Octree *OctNet, VoxelContextNet, OctAttention, octree context modeling*

More “compact” than voxels.
Large region of empty space represented by a single “0” node.

PointNet



shared single-point-input function h
(shared MLP == 1D conv, kernel_size=1)

input permutation-invariant function π

any function γ

input permutation-invariant function f

$$f(x_1, \ldots, x_n) = (\gamma \circ \pi)(h(x_1), \ldots, h(x_n))$$

Figure adapted from "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *CVPR*, 2017.

Architecture

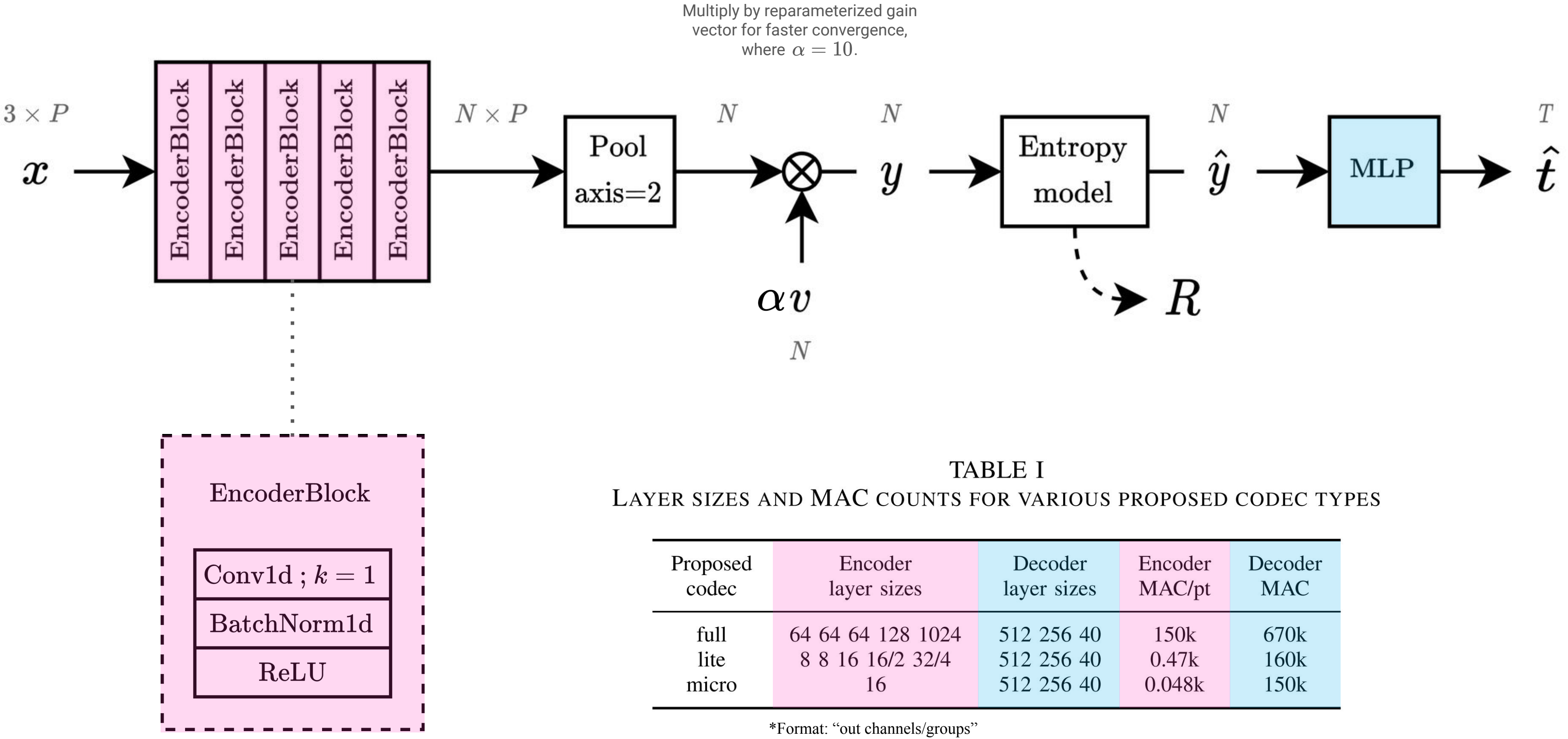


TABLE I
LAYER SIZES AND MAC COUNTS FOR VARIOUS PROPOSED CODEC TYPES

Proposed codec	Encoder layer sizes	Decoder layer sizes	Encoder MAC/pt	Decoder MAC
full	64 64 64 128 1024	512 256 40	150k	670k
lite	8 8 16 16/2 32/4	512 256 40	0.47k	160k
micro	16	512 256 40	0.048k	150k

*Format: "out channels/groups"

Experimental setup

- Dataset: sampled point clouds from ModelNet40 object meshes.
- Loss: $\mathcal{L} = R + \lambda \cdot D(\boldsymbol{t}, \hat{\boldsymbol{t}})$

ModelNet40 object meshes (before sampling).

Trained separate models for various tuples $(\lambda, P, \text{Architecture})$:

- Varying R-D tradeoff $\lambda \in [10, 16000]$
- Number of input points $P \in \mathcal{P} = \{8, 16, 32, 64, 128, 256, 512, 1024\}$
- “full”, “lite”, “micro” architecture sizes

[ModelNet40]: “3D ShapeNets: A Deep Representation for Volumetric Shapes,” *CVPR*, 2015.

PointNet missing data ratio

The PointNet paper indicates that a model trained on $P = 1024$ points degrades in accuracy as the number of points P' in the input point cloud decreases.

To avoid this, we trained models specialized for each P' of randomly-sampled input points, so that $P = P'$. We measured a sizeable improvement by doing so.

(e.g., inputting a P' = 64 point cloud into a P = 1024 trained model results in a 65% reduction in accuracy; whereas, inputting a $P' = 64$ point cloud into a $P = 64$ trained model results in a 3% reduction in accuracy.)

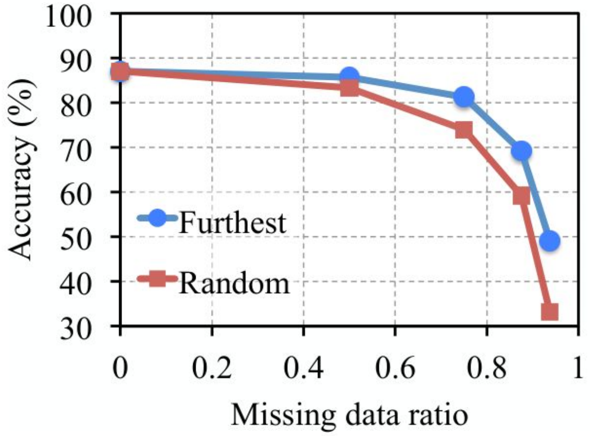
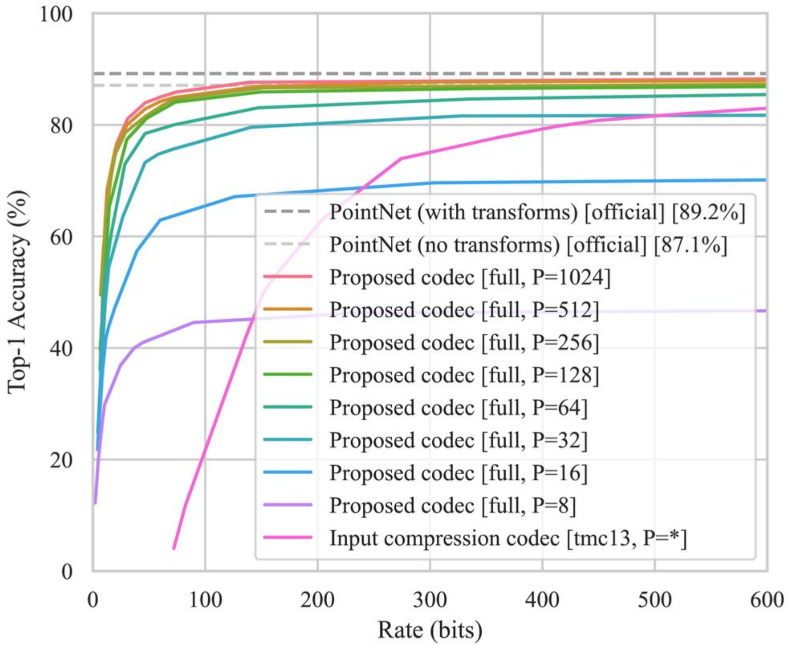
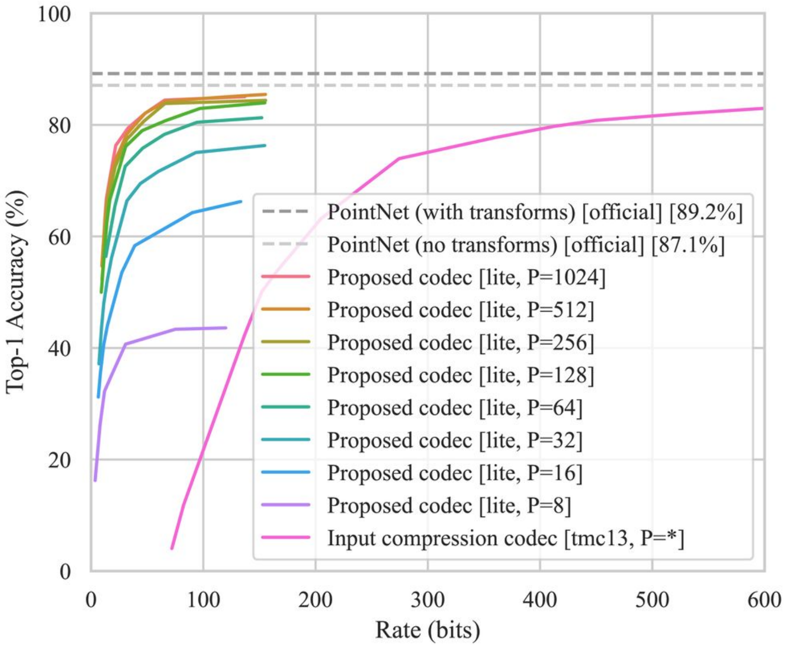


Figure from “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” CVPR, 2017.

Results: rate-accuracy curves

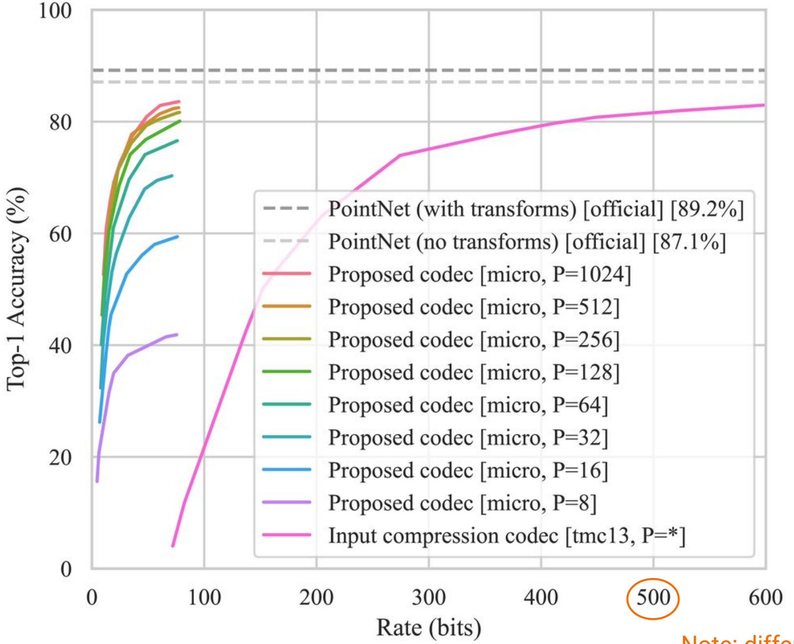


(a) "full" codec

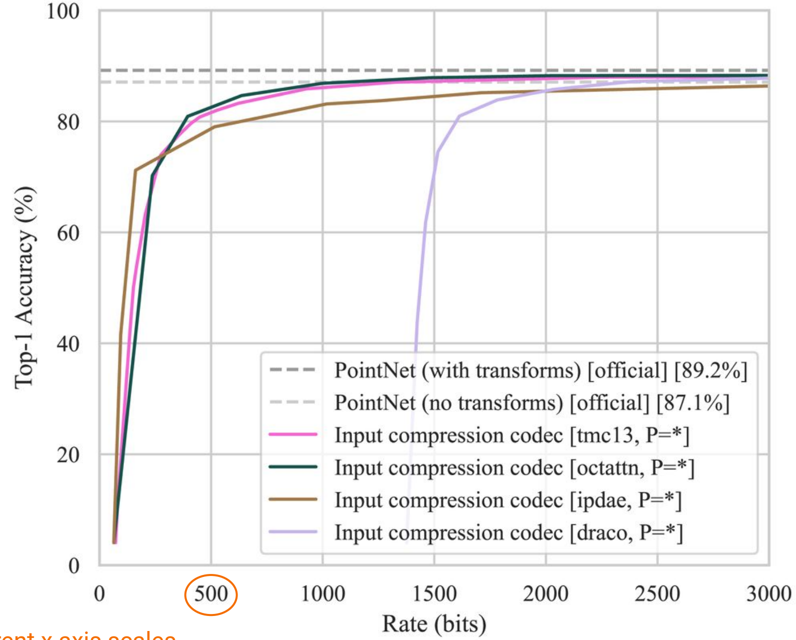


(b) "lite" codec

Results: rate-accuracy curves



(c) “micro” codec



(d) input compression codecs

Note: different x axis scales.

Results

TABLE II
BD METRICS AND MAX ATTAINABLE ACCURACIES PER CODEC

Codec	Max acc (%)	BD rate (rel %)	BD acc (%)
Input compression			
TMC13 [25]	88.5	0.0	0.0
OctAttention [12]	88.4	-13.2	+2.1
IPDAE [13]	87.0	-23.0	+3.6
Draco [26]	88.3	+780.7	-4.2
Proposed (full)			
$P = 1024$	88.5	-93.8	+16.4
$P = 512$	88.0	-93.7	+15.9
$P = 256$	87.6	-93.3	+15.4
$P = 128$	87.1	-92.7	+14.9
$P = 64$	86.1	-91.1	+13.2
$P = 32$	81.8	-90.6	+9.3
$P = 16$	70.4	-86.8	-2.3
$P = 8$	46.8	-88.5	-25.3
Proposed (lite)			
$P = 1024$	85.0	-93.0	+13.5
$P = 512$	85.5	-92.8	+14.2
$P = 256$	84.4	-92.4	+12.8
$P = 128$	84.0	-91.6	+12.5
$P = 64$	81.3	-88.5	+9.8
$P = 32$	76.3	-88.7	+4.9
$P = 16$	66.2	-86.1	-4.1
$P = 8$	43.6	-90.2	-28.0
Proposed (micro)			
$P = 1024$	83.6	-91.8	+12.7
$P = 512$	82.5	-91.6	+11.6
$P = 256$	81.6	-91.1	+11.0
$P = 128$	80.1	-90.9	+9.9
$P = 64$	76.6	-89.9	+6.5
$P = 32$	70.3	-89.0	+0.1
$P = 16$	59.4	-87.6	-10.8
$P = 8$	41.9	-88.3	-28.8

P is the number of points in the input $\boldsymbol{x}$. The BD metrics were computed using the TMC13 input compression codec as the reference anchor.

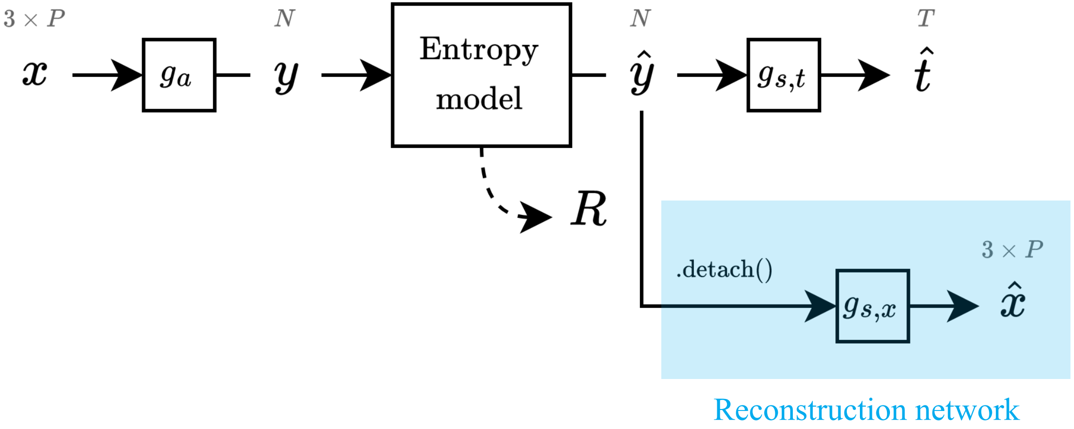
Proposed (full) — Encoder: 150 kMAC/pt; Decoder: 670 kMAC

Proposed (lite) — Encoder 0.47 kMAC/pt; Decoder: 160 kMAC

Proposed (micro) — Encoder: 0.048 kMAC/pt; Decoder: 150 kMAC

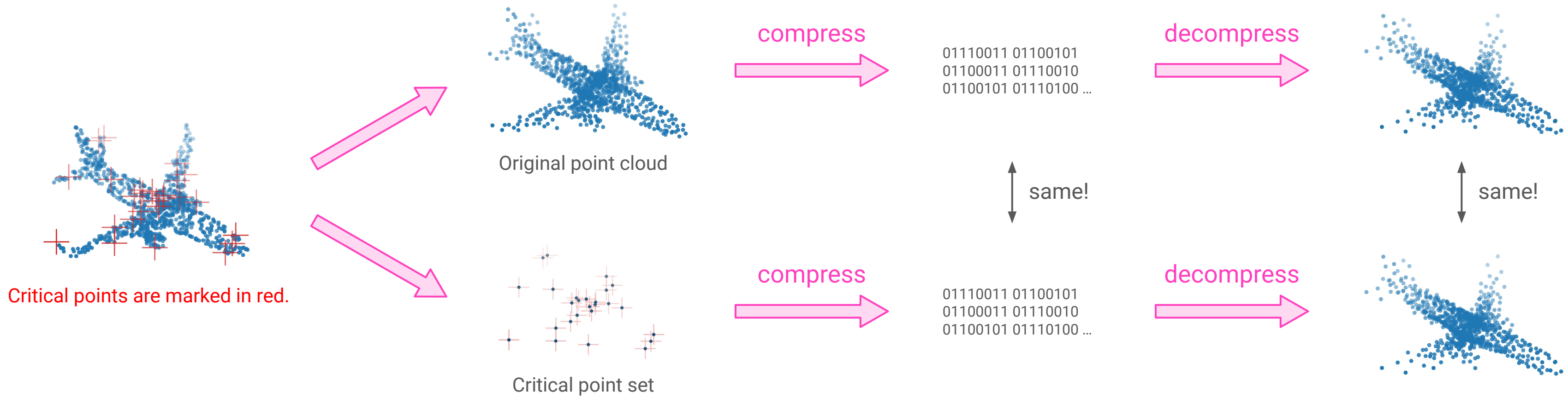
Reconstruction network (for visualization only)

We trained an auxiliary reconstruction network on the loss $\mathcal{L} = D(\boldsymbol{x}, \hat{\boldsymbol{x}})$, where D is Chamfer distance. Note that these gradients are not propagated to $\hat{\boldsymbol{y}}$.



Critical point set

For a specific codec, the **critical point set** is a minimal subset of the input point cloud that generates the exact same compressed bitstream as the input point cloud.



Critical point set (formally)

Definition. For any given point cloud $\boldsymbol{x}$, let $\boldsymbol{x}_C \subseteq \boldsymbol{x}$ denote a *critical point set*. Then, $g_a(\boldsymbol{x}_C) = g_a(\boldsymbol{x}) = \boldsymbol{y}$, and there is uniquely one valid critical point set $(\boldsymbol{x}_C)_C$ for $\boldsymbol{x}_C$, and it is itself.

A critical point set may be computed by

$$\boldsymbol{x}_C = \bigcup_{1 \le j \le N} \arg\max_{\boldsymbol{x}_i \in \boldsymbol{x}} \left(h(\boldsymbol{x}_i)\right)_j,$$

where $\{h(\boldsymbol{x}_i) : 1 \le i \le P\}$ represents the entire set of generated latent vectors immediately preceding max pooling.

Reconstructions

Our codecs achieve 80% accuracy at:

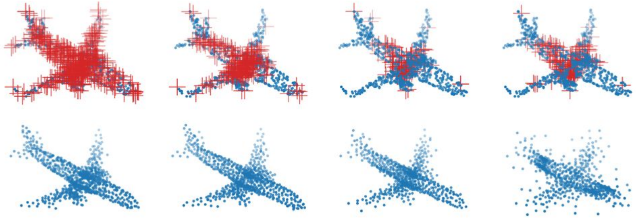
Codec	Rate
full	30 bits
lite	40 bits
micro	50 bits

100% accuracy lower bound on rate for 40 balanced classes:

$$\log_2(40) \approx 5.3 \text{ bits}$$

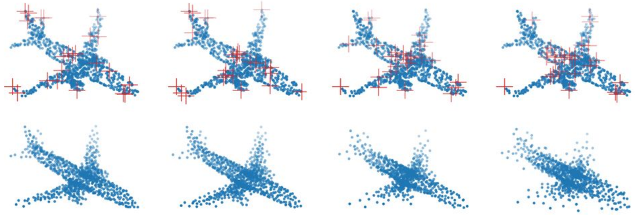
Recall: $h(x)$ is applied to each point independently. No information mixing, except for the max pooling operation!

Contrast with traditional MLP classifier that mixes information to achieve low rate.



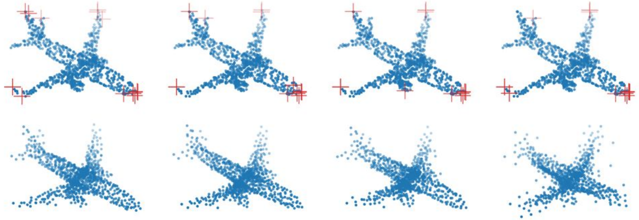
(i) 284 bits (ii) 43 bits (iii) 18 bits (iv) 12 bits

(a) "full" codec



(i) 124 bits (ii) 36 bits (iii) 19 bits (iv) 13 bits

(b) "lite" codec



(i) 76 bits (ii) 27 bits (iii) 23 bits (iv) 14 bits

(c) "micro" codec

Fig. 4. Reconstructions of a sample airplane 3D model from the ModelNet40 test set for various codecs and bitrates. For each reconstruction, its corresponding reference point cloud is marked with *critical points* in red.

Discussion

$$H(\boldsymbol{x}) = I(\boldsymbol{x};\boldsymbol{x}) \geq I(\boldsymbol{x};\hat{\boldsymbol{y}}) = H(\hat{\boldsymbol{y}}) - H(\hat{\boldsymbol{y}} \mid \boldsymbol{x}) = H(\hat{\boldsymbol{y}})$$

Thus, on average, $\hat{\boldsymbol{y}}$ must be at least as compressible as $\boldsymbol{x}$.

In fact, since $\hat{\boldsymbol{y}}$ is the same when generated from the critical point set $\boldsymbol{x}_C \subseteq \boldsymbol{x}$, $H(\boldsymbol{x}) \geq H(\boldsymbol{x}_C) \geq H(\hat{\boldsymbol{y}})$.

Furthermore, $|\boldsymbol{x}_C| \leq N$ = 16 and 32 for the “lite” and “micro” codecs.
Their $H(\hat{\boldsymbol{y}})$ is upper bounded by the entropy of the critical points.

This explains why the rate is so low. (But surprisingly, the accuracy is still good!)

Conclusion

- New codec for point cloud classification.
- Our "full" codec achieves great rate-accuracy performance vs “traditional” methods.
- Our "lite" and "micro" codecs achieve comparable gains in rate-accuracy performance, while consuming minimal edge-side computational resources.
- Helps progress towards achieving more capable end devices.

Future work:

- Other point cloud tasks (e.g. segmentation, object detection).
- Complex tasks involving larger models and point clouds from real-world datasets.
- Scalable and multi-task point cloud compression.

Thank you