Learned Point Cloud Compression for Classification

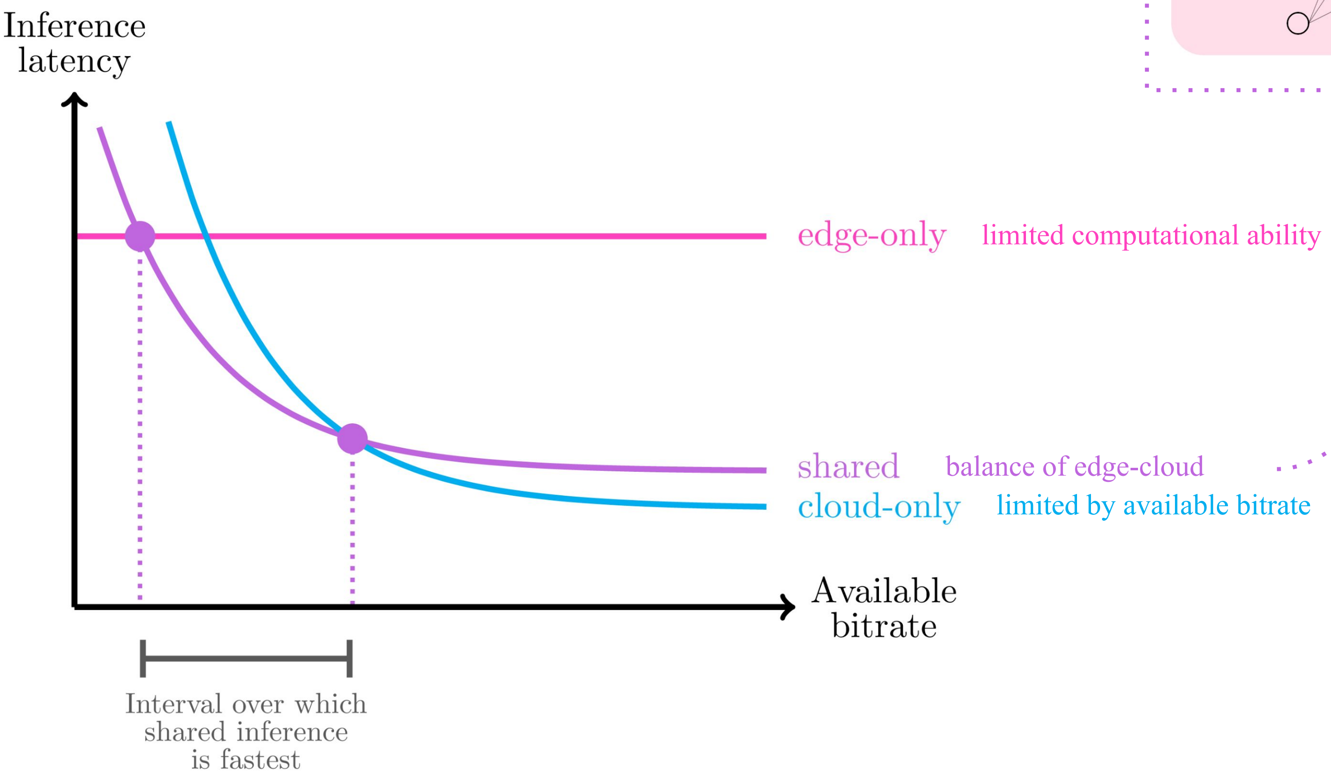
Mateen Ulhaq and Ivan V. Bajić

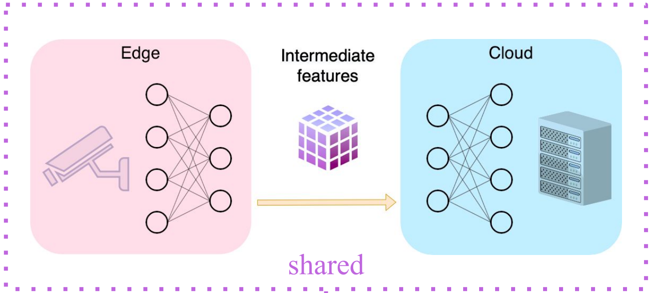
MMSP 2023   Poitiers, France

SIMON FRASER UNIVERSITY

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

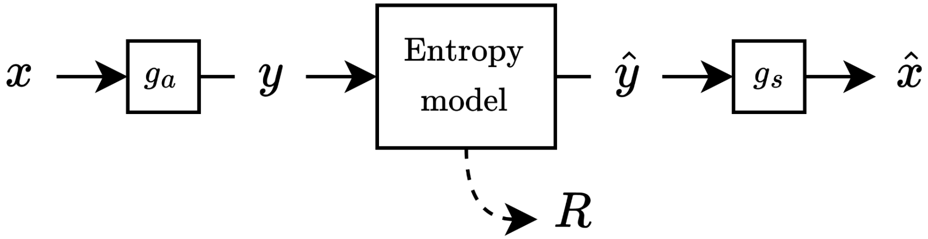
Motivation


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


Edge
Intermediate features
Cloud
shared

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

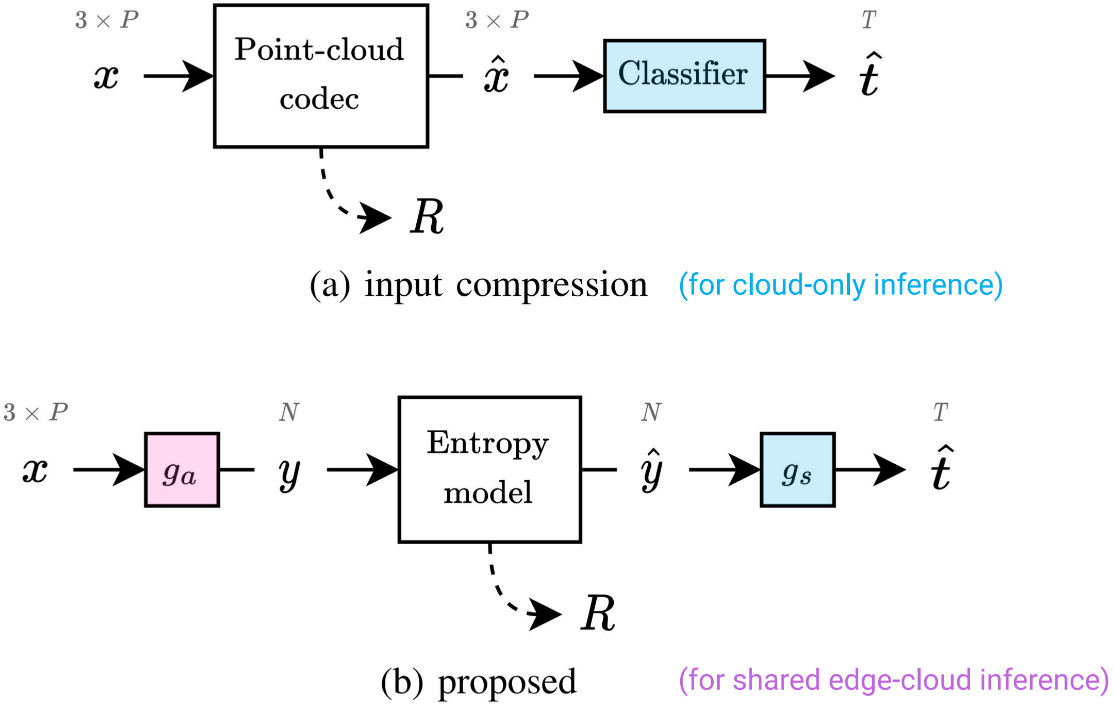


Fig. 1. High-level comparison of codec architectures.

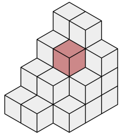
Input data formats

P × 3 matrix

$$[(x_1, y_1, z_1),\ (x_2, y_2, z_2),\ \ldots,\ (x_n, y_n, z_n)]$$

Point list *PointNet, PointNet++*

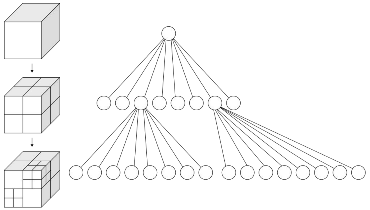
Very light MLP-based architectures.
No worthwhile canonical ordering of points.
Challenges: order-invariance, finding 3d metric structure aware operations.



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

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

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

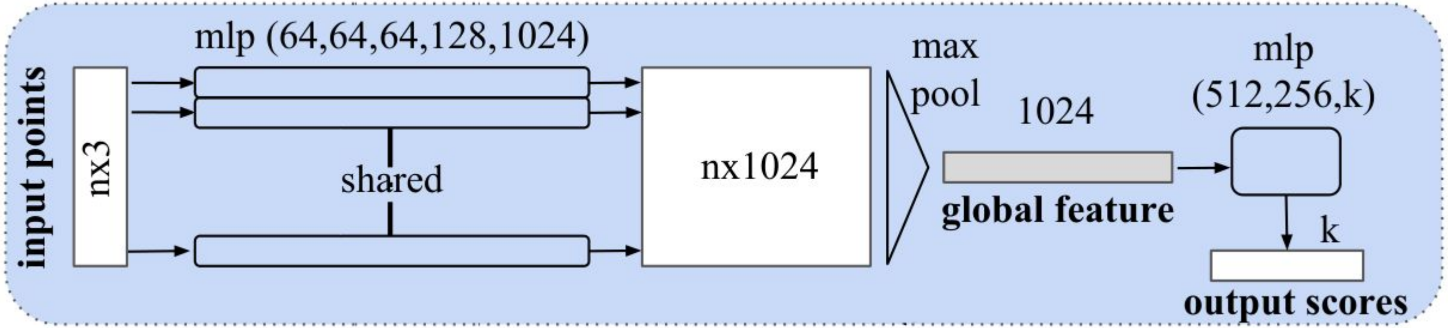


https://commons.wikimedia.org/wiki/File:Octree2.svg

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

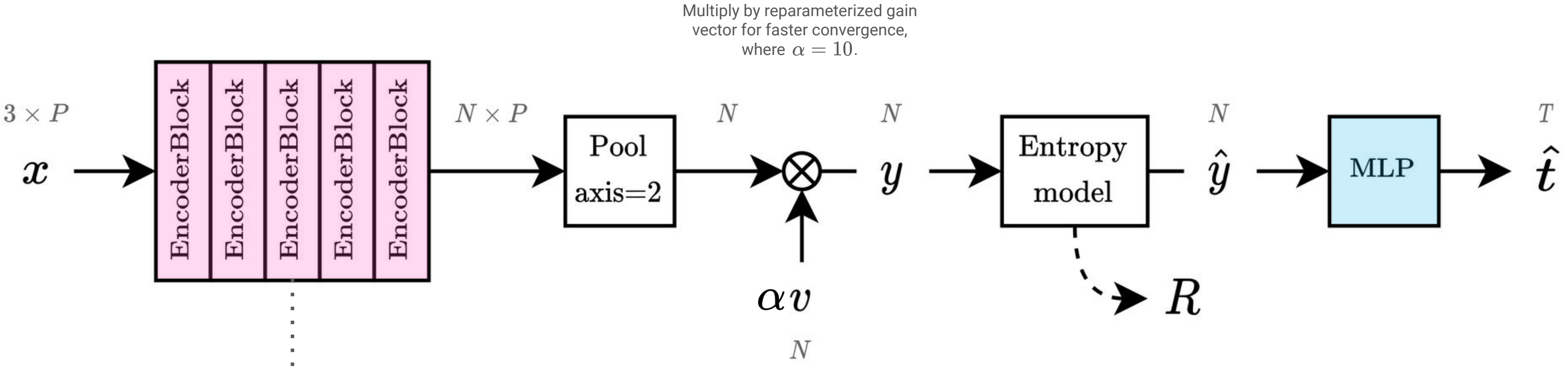
any function γ

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



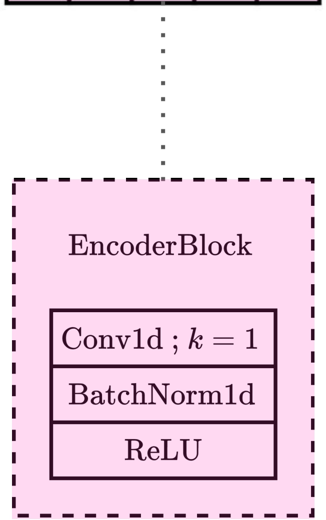


TABLE I
LAYER SIZES AND MAC COUNTS FOR VARIOUS PROPOSED CODEC TYPES

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.)

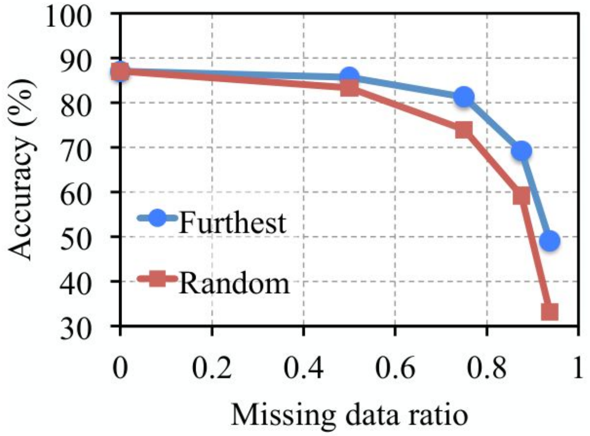
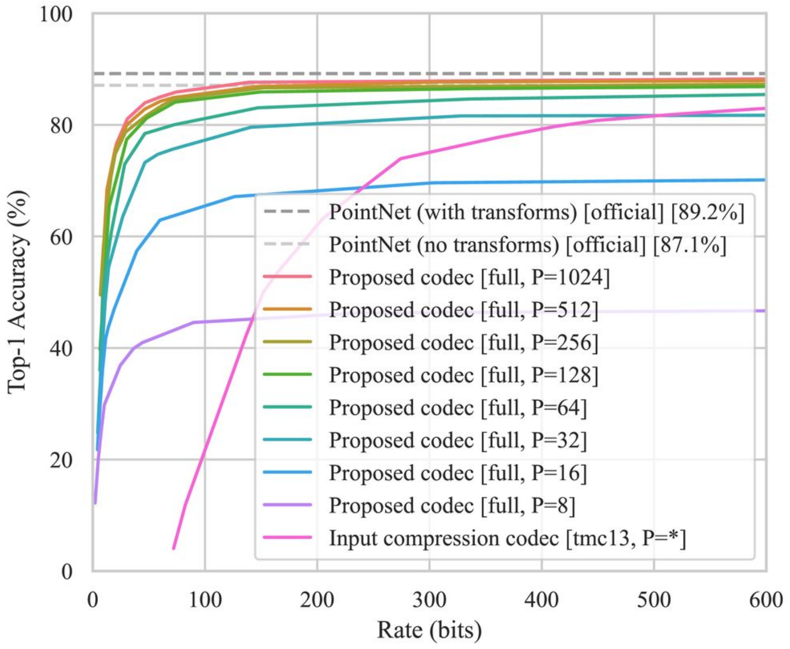
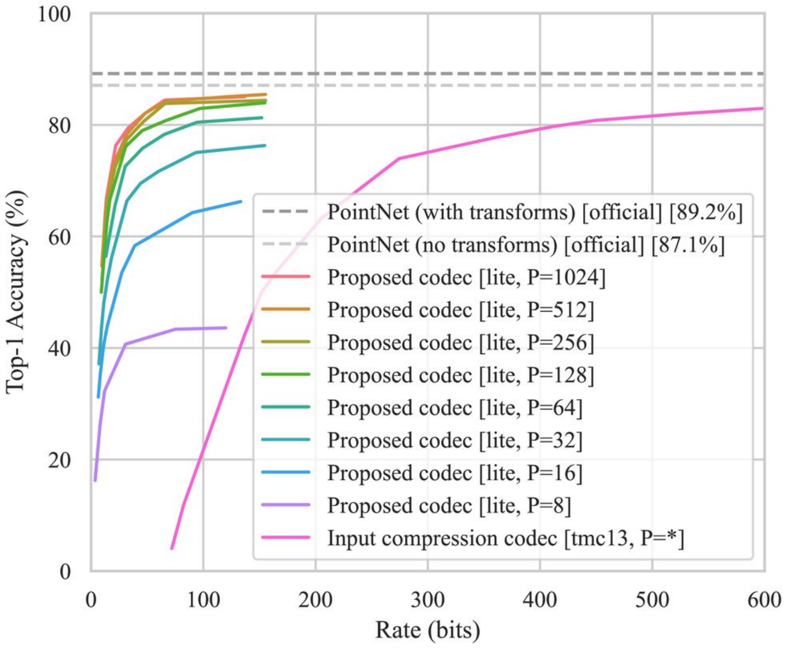


Figure from “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” CVPR, 2017.

Results: rate-accuracy curves

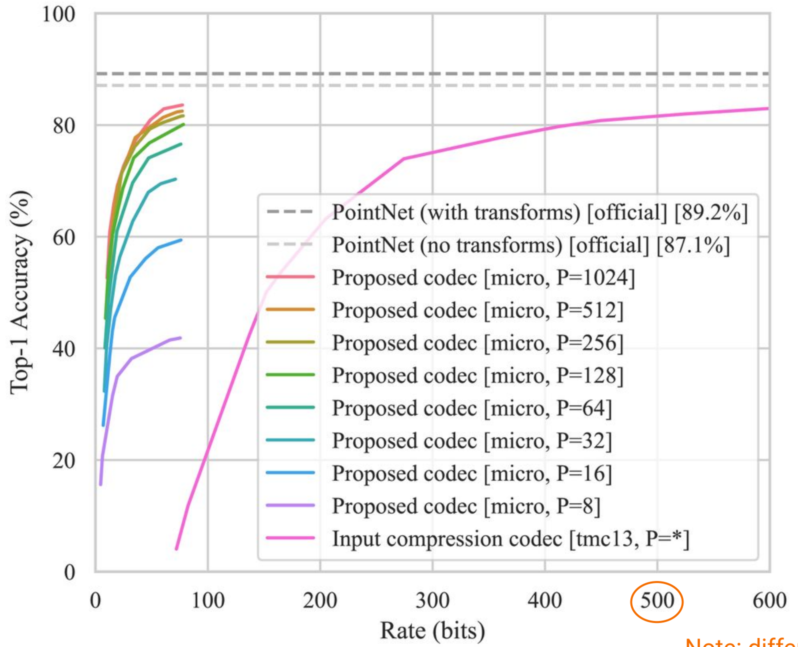


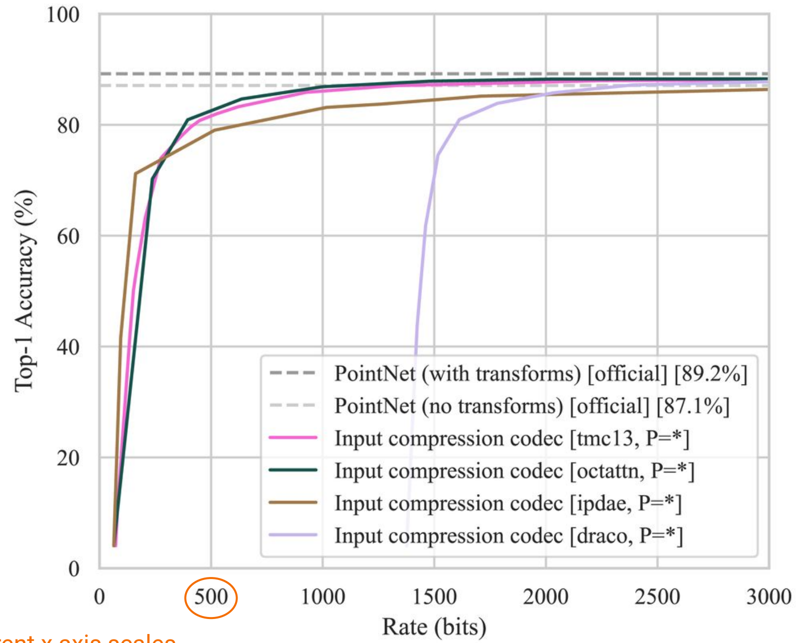
(a) “full” codec



(b) “lite” codec

Results: rate-accuracy curves





Note: different x axis scales.

(c) “micro” codec

(d) input compression codecs

Results

Encoder: 150 kMAC/pt
Decoder: 670 kMAC

Encoder 0.47 kMAC/pt
Decoder: 160 kMAC

Encoder: 0.048 kMAC/pt
Decoder: 150 kMAC

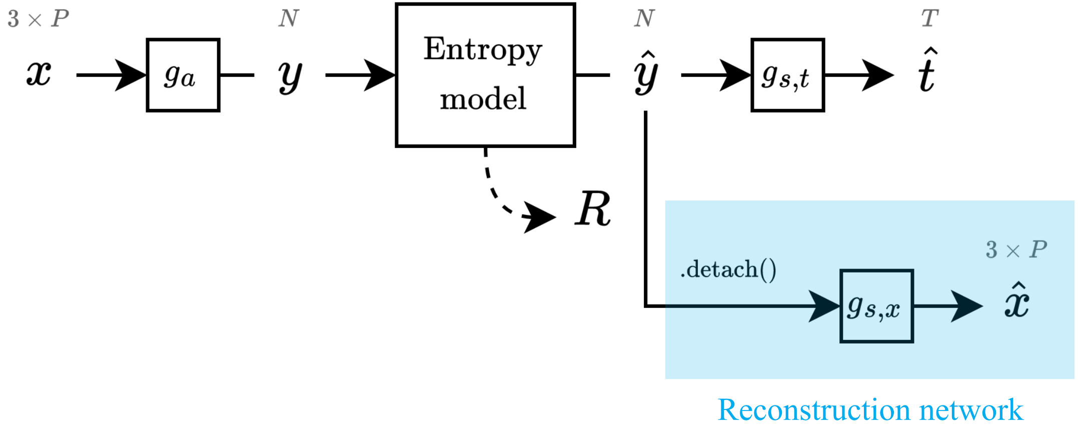
TABLE II
BD METRICS AND MAX ATTAINABLE ACCURACIES PER CODEC

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.

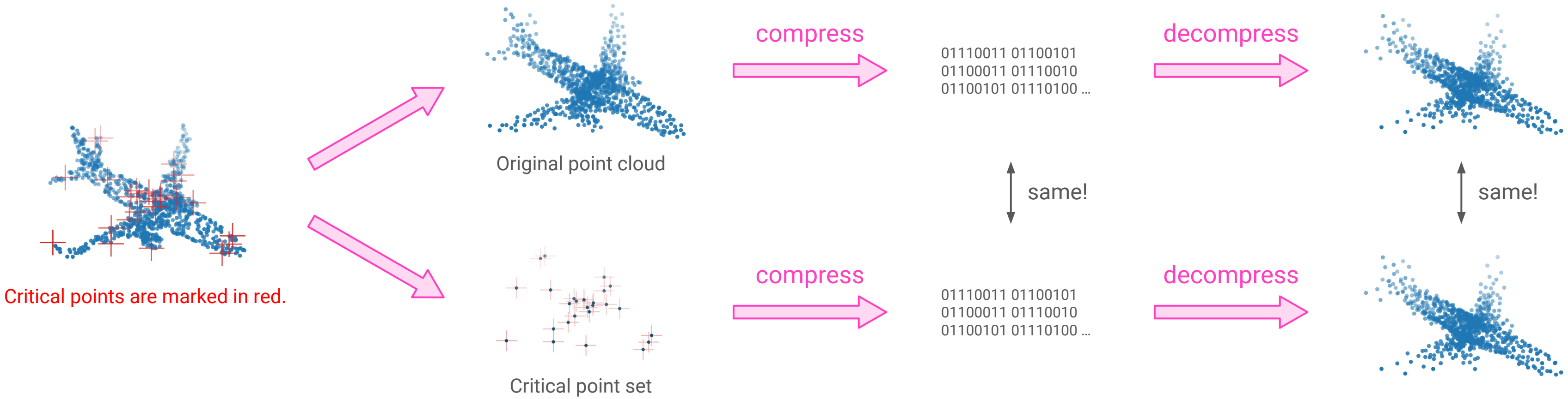
Reconstruction network (for visualization only)

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

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

where $\{h(\boldsymbol{x}_i) : 1 \leq i \leq P\}$ 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.

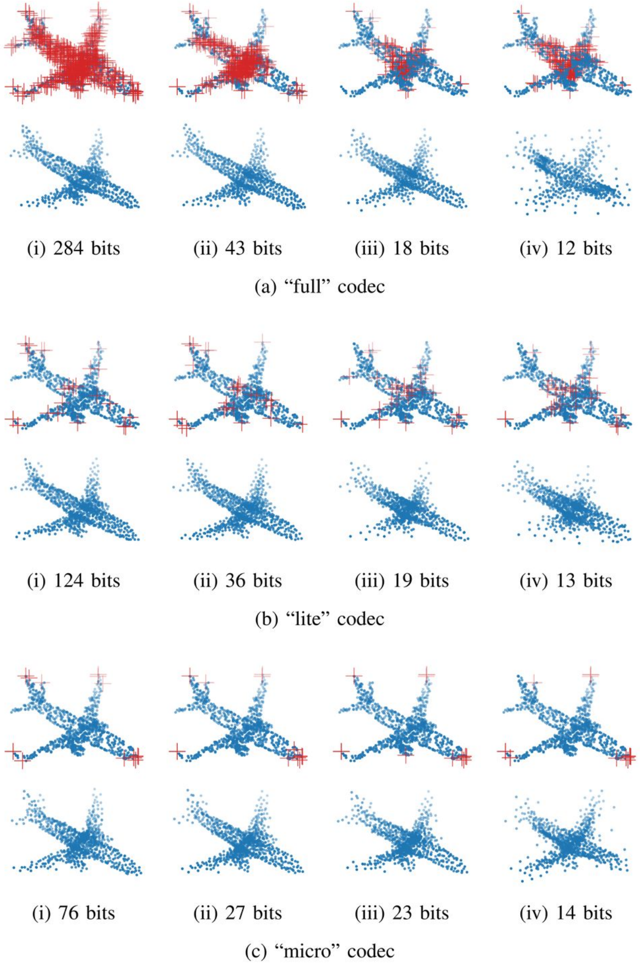


Fig. 4. Reconstructions of a sample airplane 3D model from the ModelNet40 test set for various codecs and bitrates. For each reconstruction, its corresponding reference point cloud is marked with *critical points* in red.

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

Q&A prediction

Q: How was sampling done?
A: Uniformly sampled point cloud from object meshes (surfaces), then randomly subsampled that for smaller clouds. (Or was it *all* uniform sampling? I should check…)

Q: Why not farthest point sampling?
A: Uniform sampling almost models a true Gaussian. And random subsampling = zero computation.

Q: Why the gain vector?
A: Better training stability, and allows network to zero out channels more easily, IIRC. (Even though conv1d is has equivalent representational capabilities.)

Q: Why multiply by 10 before the entropy bottleneck?
A: Much faster convergence since it now operates in range [-10, 10] instead of [-1, 1], which disappears under quantization pretty easily.

Q: Isn't ModelNet40 too easy a dataset?
A: Perhaps, but this was a first work in point cloud compression + classification.

Q: But if N=16… is the theoretical analysis not obvious?! N=16 tensor elements would certainly have small rate…
A: Mathematically, nothing inhibits an element from being uniformly distributed along a domain of $[0, 2^{100000}]$. Well, hardware floating point does, I guess. But a more *mathematically* rigorous upper bound argument is that the number of *critical points* ≤ N, as presented.

Q: …
A: …

Q: …
A: …

Overview

- Motivation
- Preliminaries / Related works
- Architecture
- Experimental setup
- Results
- Discussion
- Conclusion

Discussion

For input point clouds containing $P = 1024$ points, our "full", "lite", and "micro" codec configurations achieve an accuracy of 80\% with as few as 30, 40, and 50 bits.

For comparison, $\log_2(40) \approx 5.3 \text{ bits}$ are required to losslessly encode uniformly distributed class labels of the 40 classes from ModelNet40.

Our codec comes surprisingly close to this theoretical lower bound, despite the fact that our architecture design omits the traditional MLP "classifier" within the encoder.

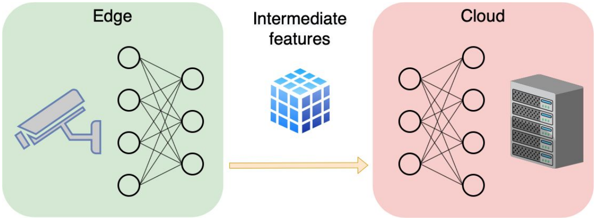
The same pointwise function is applied to all points, and the only operation that "mixes" information between the points is a pooling operation.

ModelNet40 object meshes (before sampling).

$$\min_{p(\hat{\boldsymbol{y}}|\boldsymbol{x})} \quad I(\boldsymbol{x};\hat{\boldsymbol{y}}) - \beta \cdot I(\hat{\boldsymbol{y}};\hat{\boldsymbol{t}})$$

$$\min_{p(\hat{\boldsymbol{y}}|\boldsymbol{x})} \quad H(\hat{\boldsymbol{y}}) + \lambda \cdot D(\boldsymbol{t},\hat{\boldsymbol{t}})$$

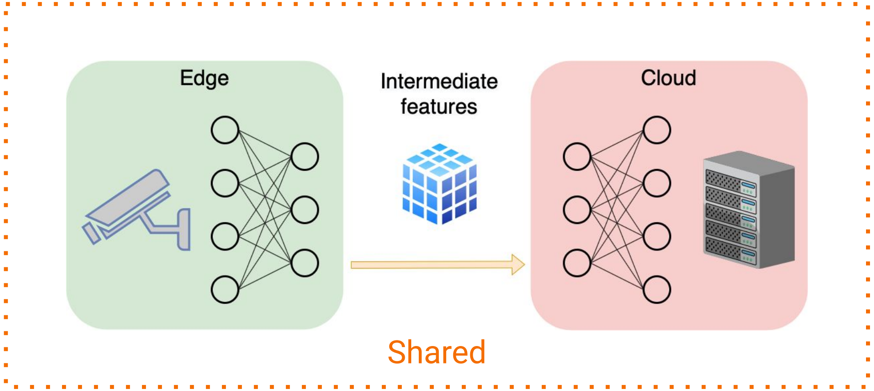
$$\mathcal{L} = R + \lambda \cdot D(\boldsymbol{t},\hat{\boldsymbol{t}})$$

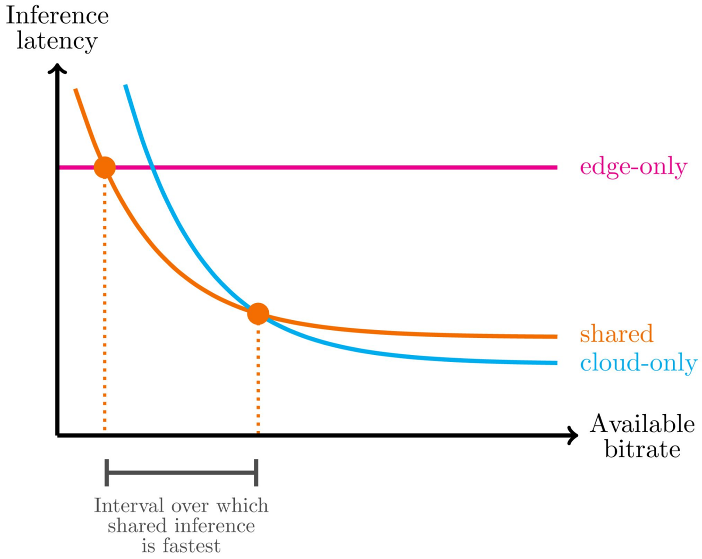


Motivation

Methods:

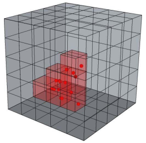
- Edge-only: limited computational ability
- Cloud-only: limited by available bitrate
- Shared: balance of both



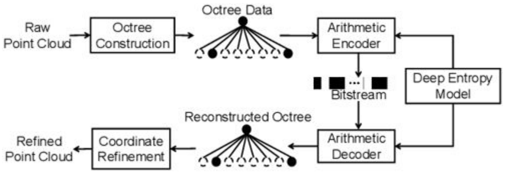


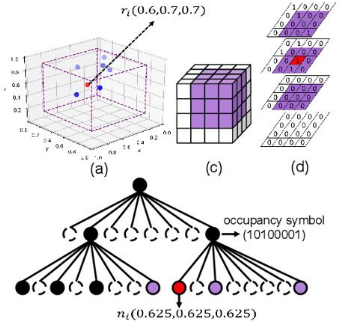
Input data formats

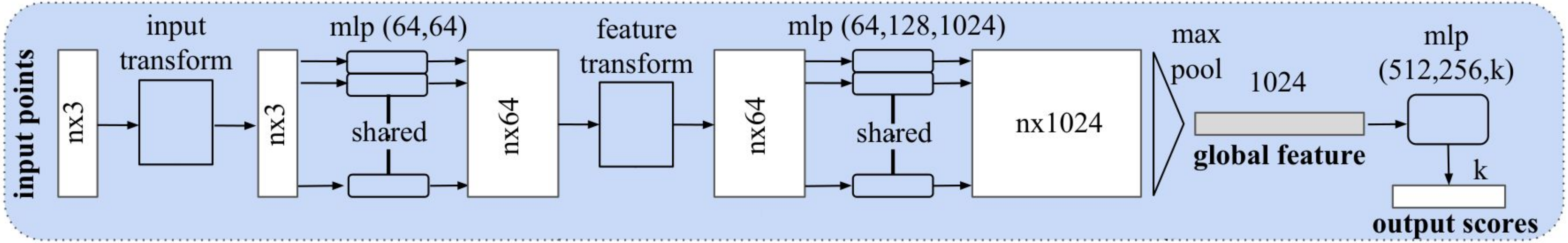
- Raw point lists `[(x1, y1, z1), (x2, y2, z2), …, (xn, yn, zn)]`
  - PointNet, PointNet++, etc
- Voxels
  - 3D ShapeNet, various convolutional models, etc

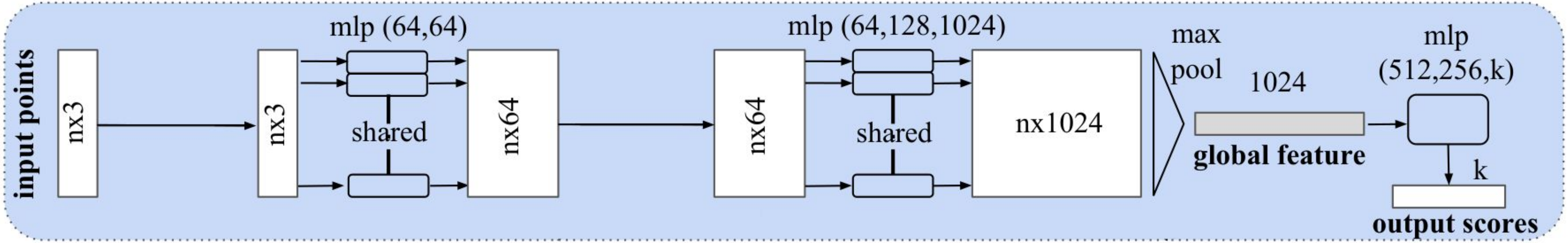


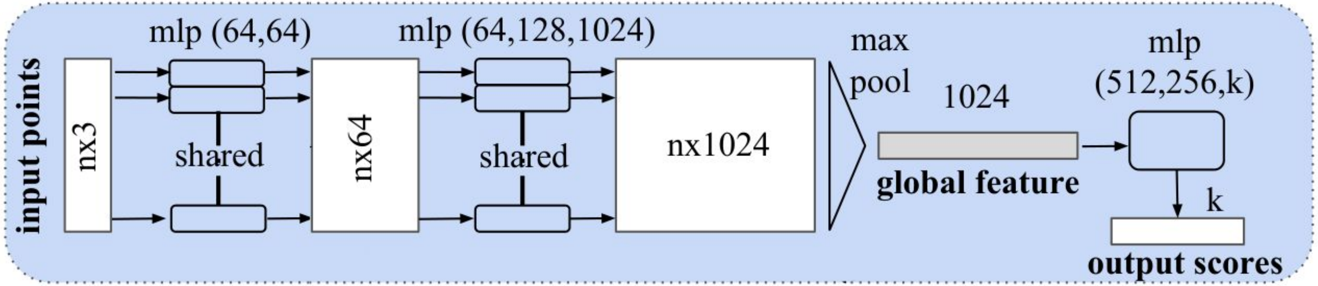
- Octrees
  - VoxelContextNet, OctAttention, octree context modelling approaches, etc



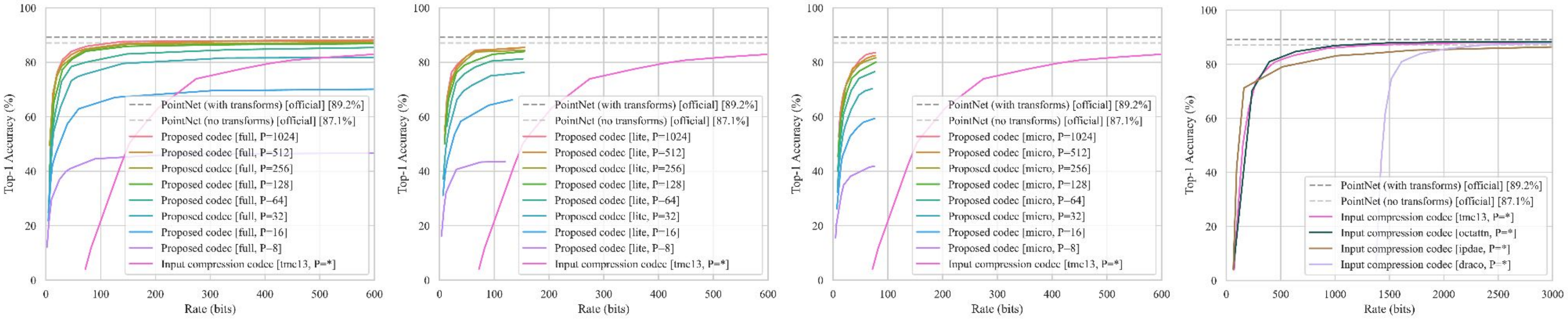



input points
nx3
input
transform
nx3
mlp (64,64)
shared
nx64
feature
transform
nx64
mlp (64,128,1024)
shared
nx1024
max
pool
1024
global feature
mlp
(512,256,k)
k
output scores


input points
nx3
nx3
mlp (64,64)
shared
nx64
nx64
mlp (64,128,1024)
shared
nx1024
max
pool
1024
global feature
mlp
(512,256,k)
k
output scores


input points
nx3
mlp (64,64)
shared
nx64
mlp (64,128,1024)
shared
nx1024
max
pool
1024
global feature
mlp
(512,256,k)
k
output scores

Rate-accuracy curves



(a) "full" codec (b) "lite" codec (c) "micro" codec (d) input compression codecs

Fig. 3. Rate-accuracy curves evaluated on the ModelNet40 test set.

PointNet

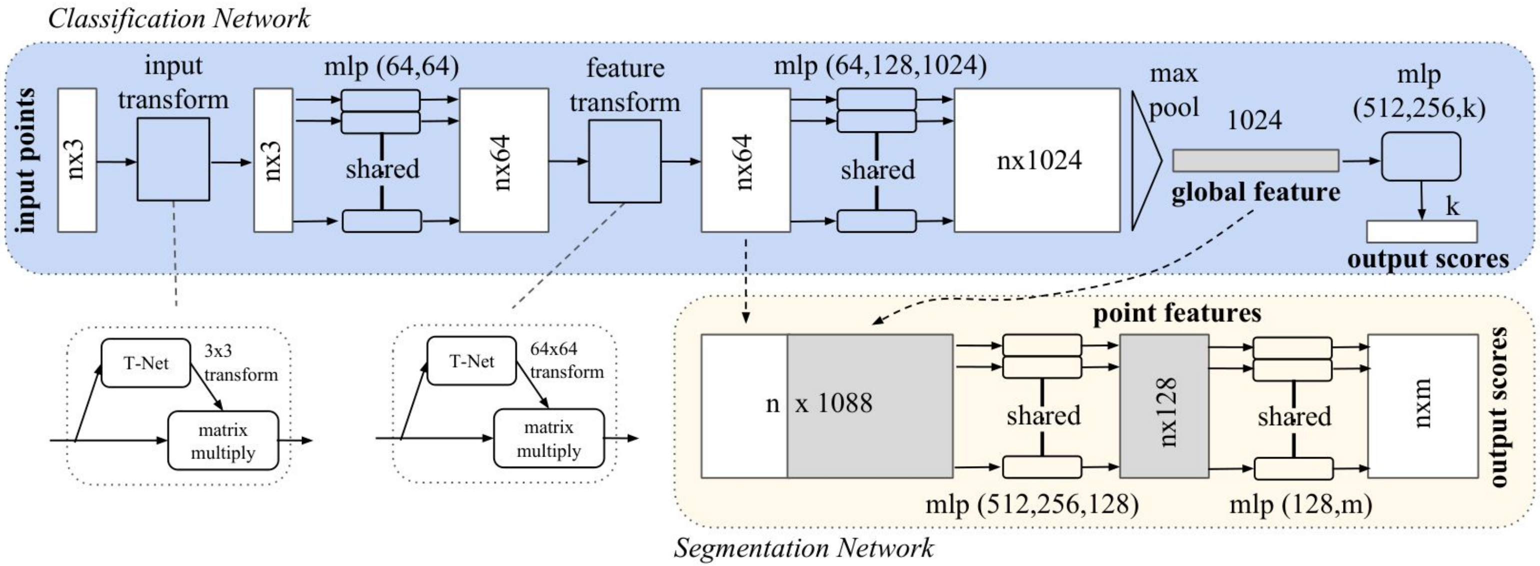


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. “mlp” stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

PointNet

Note: “Shared-MLP” = 1x1 conv
i.e. every point is independently and identically processed
by the same function f(p)

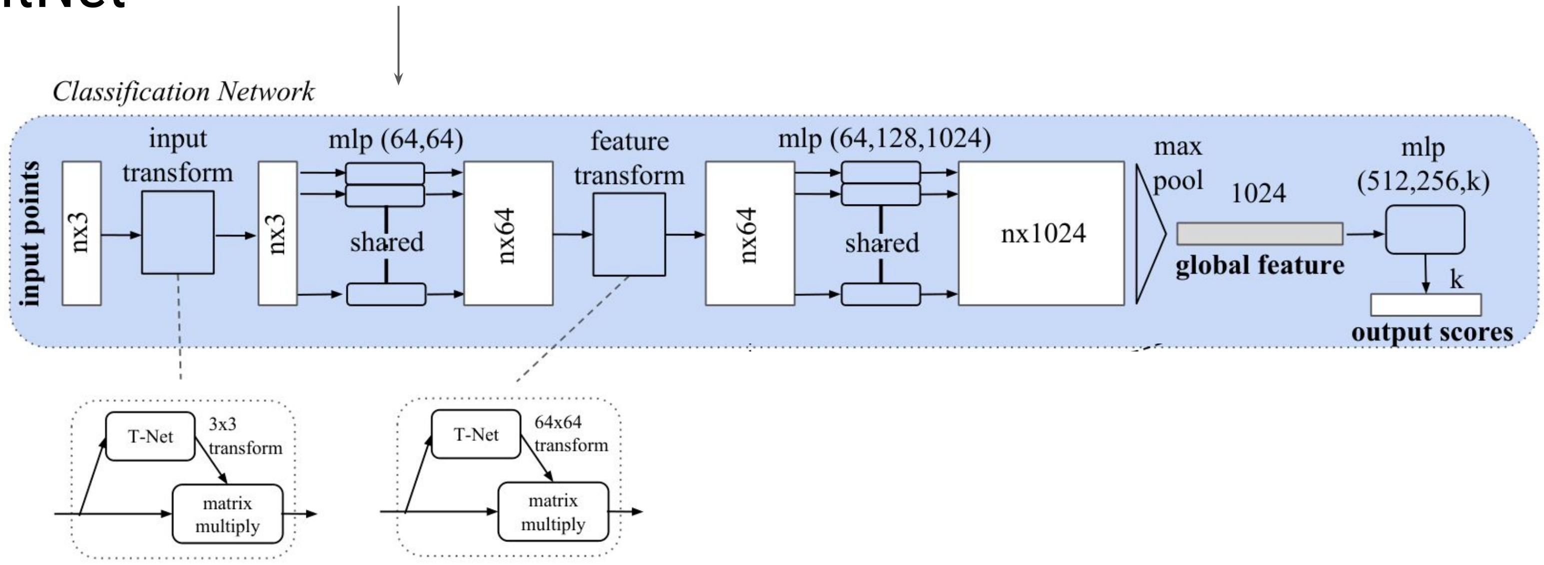


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. “mlp” stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

PointNet

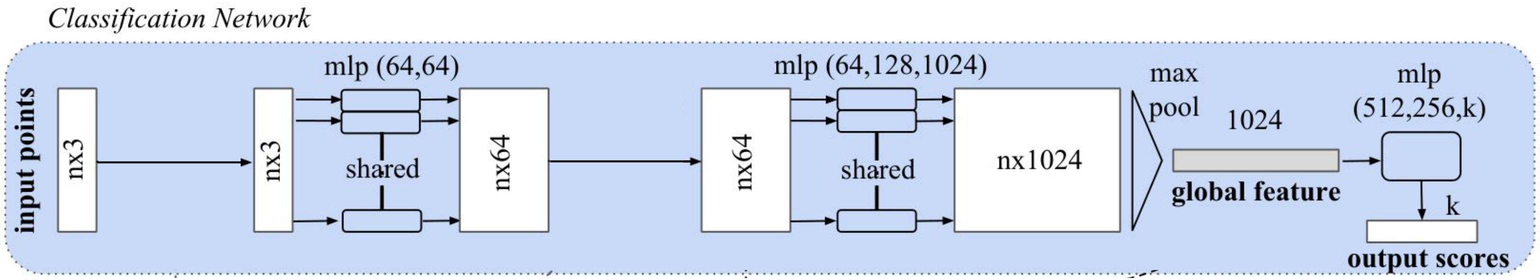


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. “mlp” stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

PointNet

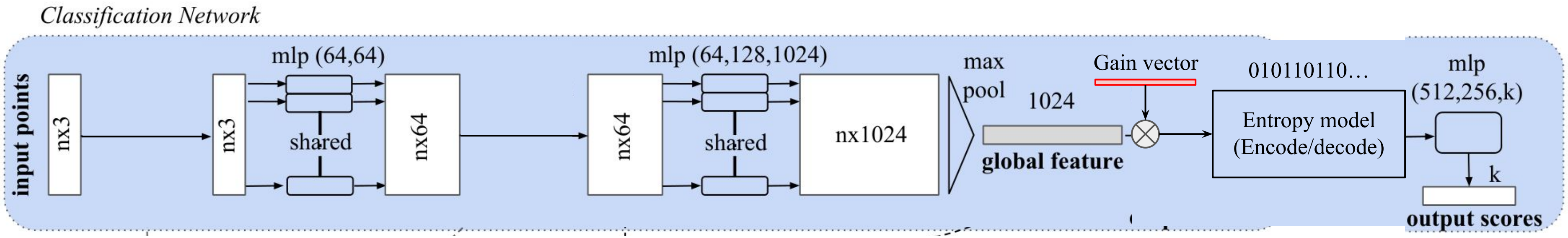


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. “mlp” stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

PointNet with split before the fully-connected classifier

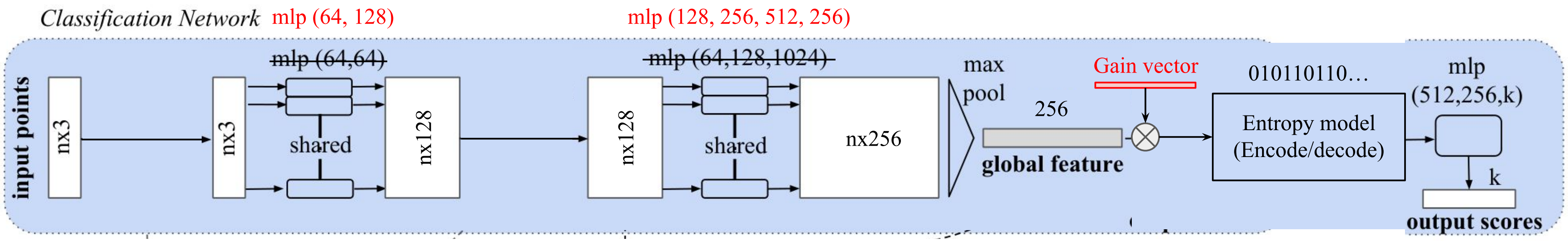


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

PointNet with split before the fully-connected classifier

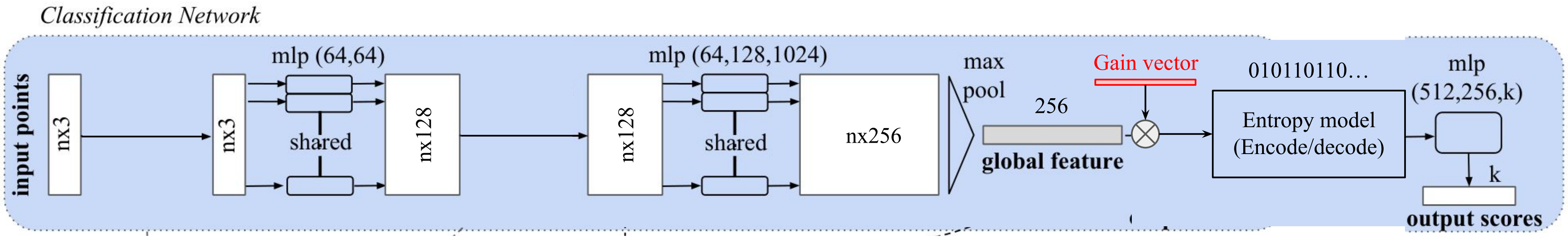
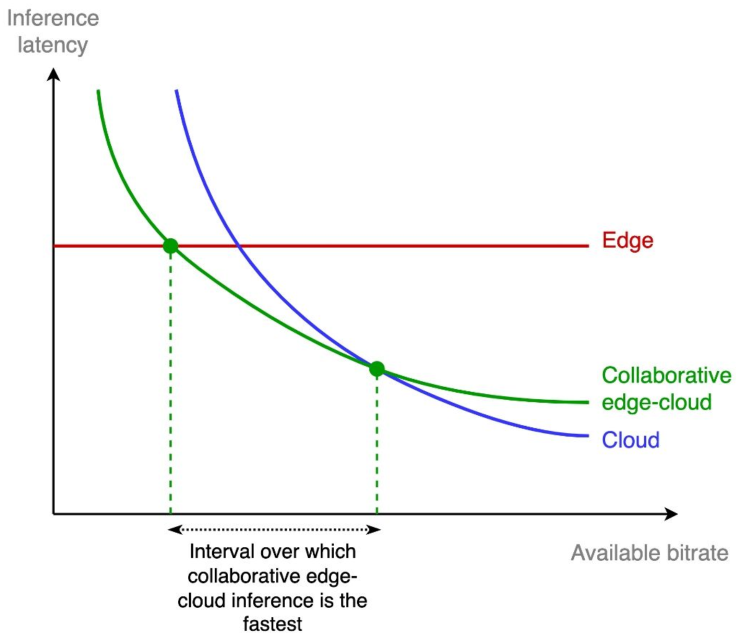


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

Takeaways

- Point-based models are often significantly more computationally lighter than comparable convolutional models.
- PointNet-style models do very well on sparse inputs, unlike “conv” models.
- The amount of points/bits needed for classification on ModelNet is quite low.
- Unlike image data (?), only a small subsample of the original input data is necessary for reasonable classification performance.


Inference
latency
Edge
Collaborative
edge-cloud
Cloud
Available bitrate
Interval over which
collaborative edge-
cloud inference is the
fastest

Auto-LaTeX Equations	Better Math Equations	Equation Editor ++	Hypatia Create	MathType (paid, 30 day free trial)
Semi-editable	Non-editable	Editable	Editable	Editable
Very slow	Medium-fast	Medium	Fast	Medium-slow
No live preview	Live preview	Semi-live preview	Live preview (but no source-only mode :(	Live preview (but no source-only mode :( )
Poor UX	Medium UX	Medium UX	Good UX	Medium UX
Single resolution	Resolution adjustable	Resolution adjustable	Resolution adjustable	Single resolution
Good default resolution	Good default resolution	Poor default resolution	High default resolution	High default resolution
Transparent background	Transparent background	Transparent background	Transparent background	Transparent background
Display mode / inline mode	Inline mode only	Display mode / inline mode	Display mode / inline mode	N/A — not LaTeX based
Single color	Single color	No colors	Colors (paid version)	No colors
				No LaTeX input
				Handwriting OCR input

$$\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-x^2}\,dx = 1$$

$$\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-x^2}\,dx = 1$$

$$\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-x^2}\,dx = 1$$

$$\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-x^2}\,dx = 1$$

$$\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-x^2}\,dx = 1$$

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{\pi^2}{6}$$

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{\pi^2}{6}$$

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{\pi^2}{6}$$

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{\pi^2}{6}$$

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{\pi^2}{6}$$

Autograd topics

- Review backprop 3B1B visual
- Colah and pytorch viz computational graphs https://pytorch.org/blog/overview-of-pytorch-autograd-engine/ ; leaf nodes, requires_grad, detach, etc
- derivatives yaml
- Using .detach() tricks [easy, slower perf] and autograd.function [more work] comparison between different situations (e.g. STE, and my particular situation)
- Jacobian jvp, etc? Ehhh maybe out-of-scope? idk…
- Review: Entropy Bottleneck in-depth, discrete CDFs, etc
- RDOQ, etc