MIND THE GAP BETWEEN TRAIN AND TEST GRIDS: RESOLUTION COLLAPSE IN VOXEL-INDEXED POINT NETWORKS IS MOSTLY A UNITS PROBLEM

Anonymous authors
Paper under double-blind review

ABSTRACT

For point cloud semantic segmentation, the leading backbones voxelize the scene and read it along a space-filling curve, which keeps their cost low on large scenes. Evaluated on a grid *coarser* than the one they trained on, they lose around 50 mIoU points on standard indoor and outdoor benchmarks. The usual explanation is that a coarser grid destroys geometry. In this work, we show that most of that collapse reverses without restoring any geometry. Sparse convolution, pooling, serialization and rotary position encoding all read the integer index $\lfloor x/\tau \rfloor$, so changing the voxel size τ changes the unit of every operator, and nothing in the network signals it. We propose *relabelling*: it recomputes that index from the physical coordinate at the training grid, without retraining. On six architecture-dataset pairs, it recovers 70 to $90\,\%$ of the collapse on grids three times coarser than the training one. On dense indoor scenes, it cuts peak memory by 3.4 to $4.4\times$ and inference time by up to $46\,\%$. Most of what a coarser grid appears to cost is a unit mismatch: even on grids *finer* than the training one, which lose no geometry, the network still collapses, and our method recovers all of it. This new training-free relabelling step offers a generic speedup for any voxel-indexed point network.

1 INTRODUCTION

Point cloud semantic segmentation is a core perception task in robotics and autonomous driving. It often runs on embedded hardware with a fixed memory and compute budget. The widely used backbones for this task (Choy et al., 2019; Wu et al., 2024a; Yue et al., 2026) discretize the geometry first. They voxelize the points and index them. Transformers also order them along a space-filling curve (serialization), so that a neighbourhood search becomes a contiguous memory access. The voxel size τ is then the simplest way to set the cost, and the parameter a user changes first to fit another budget. Changing it at inference time, however, is known to make the model collapse.

A network is trained with a voxel size τ_{train}, its *training grid*. We call its initial pipeline *stock*. Evaluated stock at $3\tau_{\text{train}}$, it loses 48.6 to 58.6 points of mean intersection over union (mIoU) on ScanNet and nuScenes (Table 2). The models are LitePT and PTv3, two U-Nets with serialized attention, and SpUNet, a sparse-convolution U-Net without attention. At first sight, a coarser grid seems to destroy the geometric information the model needs to segment the scene. But the stock models also collapse at a third of the training grid, where more points survive (Table 3, Figure 1). Understanding the input pipeline is the way into the problem.

A network that voxelizes a cloud $\mathcal{X}$ of physical coordinates $x_i \in \mathbb{R}^3$ computes

$$I_\tau(x_i) = \left\lfloor \frac{x_i}{\tau} \right\rfloor . \tag{1}$$

The sparse convolution, the pooling, the serialization and the position encoding use these integers, not the coordinates (Figure 2). A model trained at τ_{train} therefore learns in a discrete space whose implicit unit of length is τ_{train}. Evaluated at $3\tau_{\text{train}}$, it receives a scene written in a unit three times longer, and no input tells it that the unit has changed. A kernel of radius one cell now covers three times as many metres. The rotary position encoding maps each index to an angle, and this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

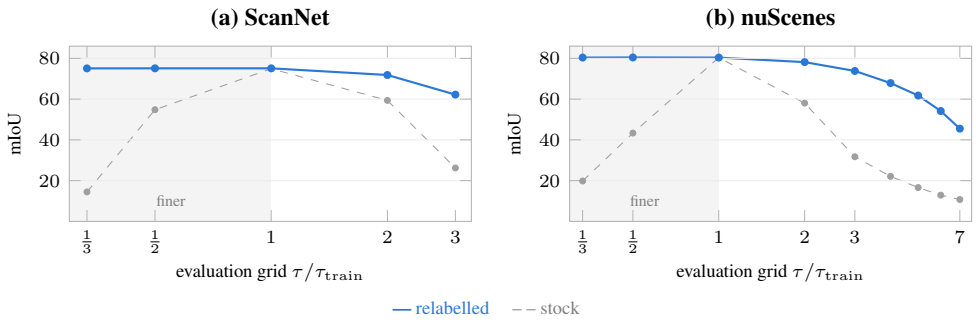


Figure 1: **Evaluated off its training grid, a voxel network collapses on both sides, and relabelling recovers most of the drop on frozen weights.** The two published LitePT-S checkpoints. *Stock* evaluates each one on the grid it is given. Below τ_{train} (shaded) the grid keeps at least as many points, and the stock model still collapses. Relabelling there keeps one token per training cell, as in training (Tables 2, 3 and 8).

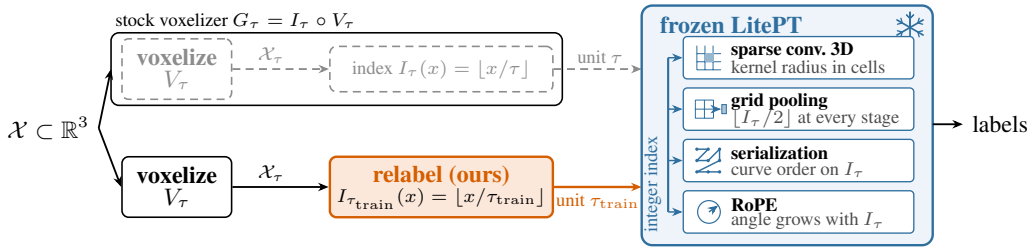


Figure 2: **Relabelling is one step between the voxelizer and a frozen network, here LitePT.** In both rows the coarse grid τ chooses the surviving points $\mathcal{X}_\tau$. The stock voxelizer G_τ (top) indexes them at τ, in cells τ/τ_{train} times longer. Relabelling (bottom) numbers them at τ_{train}, the unit the frozen operators learned.

angle now turns three times more slowly per metre. In these backbones, the voxel size serves two purposes. It decides which points survive, one point per occupied cell, a subset $\mathcal{X}_\tau \subseteq \mathcal{X}$. We call this *voxelization*, V_τ. It also sets the integer coordinates of those points. We call this *indexing*, and we call a network whose operators read that index *voxel-indexed*. The voxelizer performs both in one step, $G_\tau = I_\tau \circ V_\tau$: the same integer picks the points and is passed to the network. Written as two steps, the error is visible: I_τ indexes in cells the network was never trained on. *Relabelling*[1] keeps the points the coarse grid selected and indexes them with $I_{\tau_{\text{train}}}$ instead (Figure 2):

$$G_\tau(\mathcal{X}) = I_\tau \circ V_\tau(\mathcal{X}) \xrightarrow[\text{ours}]{} I_{\tau_{\text{train}}} \circ V_\tau(\mathcal{X}). \tag{2}$$

Relabelling returns no removed point, so nothing it recovers comes from voxelization. It recovers 36 to 52 points at $3\tau_{\text{train}}$ without touching a weight, on six architecture-dataset pairs (Table 2), and the whole drop on finer grids (Table 3). Most of the collapse is therefore a unit mismatch.

Invaria (Chang et al., 2026) reports the collapse for PTv3, on coarser grids and at half the training grid, and answers it by training. Utonia (Zhang et al., 2026) also ties the grid size to the unit, and rescales each domain during pretraining. Both change V_τ and I_τ at once. We separate the two: we measure the contribution of I_τ to the collapse on frozen weights, and recover most of it without retraining. Our contributions are threefold:

- We identify indexing as the main cause of the collapse, for three backbones on two datasets at $3\tau_{\text{train}}$ (Table 2) and at every grid we test on LitePT-S (Figure 5).
- We propose relabelling as an operation that undoes the unit alone and recovers most of the drop on frozen weights, without retraining. On dense indoor scenes, it lets one published checkpoint run on a coarser grid chosen at inference (Section 4.7).

[1]Code and weights: `https://anonymous.4open.science/r/ICLR2027-76F4/`.

- We conducted extensive experiments to demonstrate the effectiveness of relabelling (Section 4). Ablations over the operators show that only relabelling every operator recovers most of the drop (Section 4.5). A comparison with other samplers shows that the remaining drop comes from which points the grid keeps (Section 4.4).

2 Related work

Point cloud backbones and the grid index. A lidar or a depth camera delivers points whose coordinates are in metres. Point-based networks keep the metres. They apply shared per-point functions (Qi et al., 2017a), group points by ball queries around farthest-point samples (Qi et al., 2017b), and convolve (Thomas et al., 2019) or attend (Zhao et al., 2021; Wu et al., 2022) over neighbourhoods measured in metres. Voxel networks instead divide space into cubic cells of side τ and reduce each occupied cell to one point, or *token*. They replace its coordinates by the integer index of the cell, as in Equation 1 (Graham et al., 2018; Choy et al., 2019). Many point transformers keep this grid and restrict attention to a structured subset of it. This subset can be cubic windows (Lai et al., 2022), an octree (Wang, 2023), or windows of a fixed number of tokens (Liu et al., 2021; 2023). The latter are taken along a space-filling curve (Wu et al., 2024a; Yue et al., 2026). Voxelization, like random sampling (Hu et al., 2020) or farthest-point sampling, chooses which points survive, and this is what makes a coarser grid cheaper. Token merging lowers the cost at a fixed grid instead (Bolya et al., 2023; Tran et al., 2025), and Appendix G.2 compares their operation counts. A grid is the only one of these samplers that also sets the unit of the coordinates: after indexing, one step is τ metres.

Diagnosing the collapse. FixRes (Touvron et al., 2019) attributes the drop at a new test resolution in images to the apparent size of objects, and fine-tunes the network at that resolution. In lidar detection, a gap in car size between datasets is closed by rescaling the training boxes and fine-tuning (Wang et al., 2020). Invaria (Chang et al., 2026), a contemporaneous preprint, makes the same diagnosis for point clouds. Its scale factor and density factor correspond to indexing and voxelization. It answers both through training (Section 4.6). We undo the unit alone, on frozen weights.

Three kinds of answers. The first answer learns the robustness. In images, FlexiViT (Beyer et al., 2023) trains one model over random patch sizes. In point clouds, sampling the grid size or the density during training is the usual remedy (Kim et al., 2024), and it is effective. Multi-dataset training (Wu et al., 2024b) and self-supervised pretraining (Wu et al., 2025) mix sensors but voxelize them at one grid size per model. Utonia (Zhang et al., 2026) states that the grid size sets the unit of local neighbourhoods. For that reason, it rescales the coordinates of each domain before one global grid during pretraining. At inference, rescaling the scene cannot replace relabelling, since it returns the stock index (Section 4.5).

The second answer builds the robustness into the architecture. Equivariant layers respect a transformation by construction (Cohen & Welling, 2016), including scale (Sosnovik et al., 2020). OA-CNNs (Peng et al., 2024) make the receptive field adaptive. Rotary encoding (Su et al., 2024), extended to images (Heo et al., 2024), turns a position into a rotation angle. These works study how the angle varies with position, and rarely how many metres one step of position is worth. Remote sensing is the exception, because the ground distance of a pixel is known there. Scale-MAE scales its position encoding by that distance (Reed et al., 2023), and GeoRoPE rescales the offsets of a pretrained model through a trained adapter (Luo et al., 2026). An encoding of the coordinate in metres (Tancik et al., 2020) changes one layer and leaves the other three operators reading the index.

The third answer acts after training, and it is the closest to ours. Test-time adaptation updates the normalization parameters from unlabelled test batches (Wang et al., 2021). Its point-cloud versions update the backbone (Saltori et al., 2022; Mirza et al., 2023; Zou et al., 2024), the input (Shim et al., 2024) or the logits (Wang et al., 2024). None of them recomputes the grid index. Position interpolation (Chen et al., 2023) rescales the index of a pretrained language model and briefly fine-tunes it to read longer contexts. The same rescaling of the voxel index falls short (Section 4.5). Reichardt et al. (2026) tune an attention window at inference without retraining, but its size is a number of tokens, already in the unit of the network. Relabelling also acts after training, but learns nothing from the test data: it recomputes the index at the training grid. This is possible because the

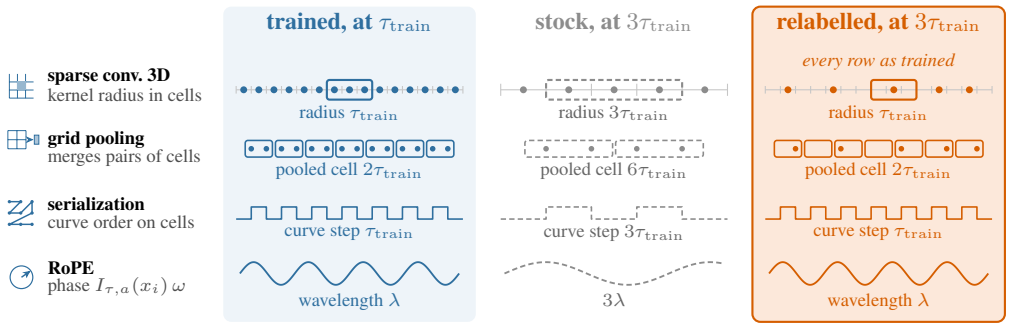


Figure 3: **Relabelling puts every operator back on the scale of the training grid.** Trained at τ_{train} (left), evaluated at $3\tau_{\text{train}}$ stock (middle) and relabelled (right). In the stock pipeline, each operator reads cells three times longer in metres. Relabelled, it reads them at τ_{train}, as in training. The phase turns once every $\lambda = 2\pi\tau_{\text{train}}/\omega$ metres. Rows are schematic.

cell length is a line of the configuration file and is known at deployment. The apparent size of an object in an image is not.

Relabelling as canonicalization. Two methods obtain invariance without changing the layers. Frame averaging averages the predictions of a network over transformed copies of the input (Puny et al., 2022). Canonicalization maps every input to a fixed reference before the network sees it (Kaba et al., 2023). Relabelling is a simple instance of the second idea. It keeps the points that survive the grid and fixes the length of one index step. It fixes neither the pose nor the shape of the cloud, and it needs no auxiliary network. In operator learning, a neural operator trained on a coarse mesh is evaluated on a finer one. This property is called discretization invariance (Li et al., 2021; Kovachki et al., 2023). Voxel-indexed networks lack it because their operators count in cells.

3 Method

3.1 Operators that read the index

Four spatial operators decide neighbourhoods and positional structure. They read the integer index $I_\tau(x_i)$ of Equation 1, and none of them receives τ (Figure 3). Their weights are learned on $I_{\tau_{\text{train}}}(x_i)$. Evaluated at $\tau \neq \tau_{\text{train}}$, they receive $I_\tau(x_i)$, which Section 3.2 replaces with $I_{\tau_{\text{train}}}(x_i)$.

Each of them counts in cells, which costs nothing at a single resolution. For example, the rotary phase $\theta_{i,a} = I_{\tau,a}(x_i)\,\omega$ of axis a turns by ω radians per cell.

3.2 Relabelling on frozen weights

Relabelling recomputes the index from the physical coordinate at the training grid, whatever τ is:

$$\lfloor x_i/\tau \rfloor - \lfloor x_{\min}/\tau \rfloor \quad \longrightarrow \quad \lfloor x_i/\tau_{\text{train}} \rfloor - \lfloor x_{\min}/\tau_{\text{train}} \rfloor, \qquad x_i \in \mathcal{X}_\tau, \tag{3}$$

from the stock index (left) to the relabelled one (right). The origin $x_{\min}$ is the minimum corner of the kept cloud, as in the stock voxelizer. Relabelling changes only the divisor, from τ to τ_{train}. Up to this translation, relabelling computes $I_{\tau_{\text{train}}} \circ V_\tau$ (Equation 2). Section 4.4 measures the effect of the origin. The coarse grid keeps its voxelization role and loses its indexing role. It chooses *which* points survive, and the training grid chooses *how* they are numbered.

Integer ratios. When $\tau = k\,\tau_{\text{train}}$ for an integer k, each cell of side τ is a block of $k \times k \times k$ cells of side τ_{train} (Figure 4). It keeps one point, so no two tokens share a training cell. Relabelling then keeps every token and only renumbers it. Relabelling is grid subsampling at τ followed by inference at the training grid, each token standing for one training cell. Tables 2 and 8 ($k = 2$ to 7) are in this case, so relabelling there changes the index and nothing else. At a non-integer ratio, some training cells receive two tokens. Only one of them is kept, as in training (Figure 4, right). Relabelling then removes a few tokens (Table 7). On a finer grid, most training cells receive several and keep one.

Which point relabelling reads. Each voxelization keeps a real point per cell, drawn at random as in training, and Equation 3 uses its coordinate. The usual voxel filter returns each cell's centroid instead (Zhou et al., 2018), which lowers the relabelled score but not the stock one (Appendix D.1).

Cost of adoption. Relabelling adds no parameter and no layer. It only needs the training grid, which the configuration published with the weights already gives. Relabelling at any other size we tried scores lower (Table 21), so it introduces no hyperparameter to tune.

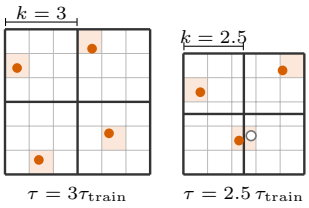


Figure 4: **At an integer ratio, relabelling only renumbers.** Thin lines: cells of τ_{train}. Thick lines: cells of τ. Hollow: a token that shares its fine cell, so it is dropped.

3.3 Measuring the contribution of the unit

We write m_{train}, m_{stock} and m_{relab} for the mIoU of a model trained at τ_{train} when it reads $I_{\tau_{\text{train}}} \circ V_{\tau_{\text{train}}}$, $I_\tau \circ V_\tau$ and $I_{\tau_{\text{train}}} \circ V_\tau$, respectively. The stock drop splits into two terms:

$$\underbrace{m_{\text{train}} - m_{\text{stock}}}_{\text{stock drop}} = \underbrace{m_{\text{train}} - m_{\text{relab}}}_{\text{relabelled drop}} + \underbrace{m_{\text{relab}} - m_{\text{stock}}}_{\text{unit term}}. \tag{4}$$

The unit term is what relabelling recovers. At an integer ratio, its two inputs hold the same tokens (Section 3.2), so it depends on the index alone. The relabelled drop is what relabelling leaves. It holds the *sampling term*, what the points removed by V_τ cost the frozen model, plus anything relabelling does not restore. The main such effect is the sub-cell phase, the position of each token inside its coarse cell, which depends on the grid origin (Section 4.4). The relabelled drop is therefore an upper bound on the sampling term, up to that phase. The three backbones carry nested subsets of the operators (Table 1). An effect that survives on SpUNet therefore comes neither from the serialization nor from the phase. PTv2 (Wu et al., 2022) reads metres, and relabelling leaves it unchanged (Section 4.3).

Table 1: **Each network carries a nested subset of the four operators.** ✓: reads the index. m: reads metres. attn.: where attention takes its neighbours. §: negative control.

	conv.	pool.	serial.	attn.	RoPE
SpUNet	✓	✓	—	—	—
PTv3	✓	✓	✓	✓	—
LitePT	✓	✓	✓	✓	✓
PTv2§	—	m	—	m	—

Rescaling the index or the scene. With $k = \tau/\tau_{\text{train}}$, grid rescaling feeds $k\, I_\tau(x_i)$, which has the right unit but only multiples of k. Shrinking the scene by k, as Utonia does in pretraining, feeds $I_{\tau_{\text{train}}}(x_i/k) = I_\tau(x_i)$, the stock index, so it cannot repair the unit (Section 4.5).

4 Experiments

We ask how much of the collapse is the unit (Section 4.2) and whether relabelling changes anything else (Section 4.3). We then ask what remains (Section 4.4), which operators must be relabelled (Section 4.5), how relabelling combines with training (Section 4.6), and what it saves (Section 4.7).

4.1 Setup

We evaluate semantic segmentation on two datasets. ScanNet (Dai et al., 2017) contains indoor rooms reconstructed from a depth sensor ($\tau_{\text{train}} = 2$ cm, 312 validation scenes). nuScenes (Caesar et al., 2020) contains lidar frames from urban driving ($\tau_{\text{train}} = 5$ cm, 6019 frames). We chose this pair because a room is sampled far above its voxel size everywhere. A lidar frame is sparser than the grid beyond a few metres. We evaluate three backbones, SpUNet, PTv3 and LitePT, which carry nested subsets of the operators (Table 1). Two checkpoints come from their authors, and we trained the rest. All are frozen during the comparisons. Everything runs on a single RTX 5000 Ada with 32 GB, from Pointcept (Pointcept Contributors, 2023), with no architectural change.

Notation. $\Delta_\tau = \text{mIoU}(\tau) - \text{mIoU}(\tau_{\text{train}})$, computed within a row and within one evaluation run, so a drop is a negative Δ_τ. Two drops are comparable only under one test-time augmentation

setting. Unless stated otherwise: mIoU $\times 100$, all validation scenes, each processed whole, every original point scored by the token of its cell, no test-time augmentation, and one seed per trained model. Tables with two seeds say so. Appendix A gives every configuration. We read a difference as real only above the spread between evaluation runs of one checkpoint, at most 0.42 point. Between two trained models, the threshold is the seed spread, 0.83 point indoors and 1.71 outdoors (Table 6).

4.2 Recovery on frozen weights

Table 2: **Relabelling recovers 36 to 52 points at $3\tau_{\text{train}}$ on all six pairs, without touching a weight.** Each shaded row evaluates the *same* weights as the row above it, through Equation 3. In the terms of Equation 4, $\Delta_{3\tau_{\text{train}}}$ is the stock drop on a stock row and the relabelled drop on a shaded row, and *recovered* is the unit term over the stock drop. Bold: relabelled drops, and the highest *recovered* on each dataset. Both rows of a pair are evaluated at the same batch size, since the transformers serialize the batched cloud. • published by its authors, † trained here.

		evaluation grid τ				
dataset	model	τ_{train}	$2\tau_{\text{train}}$	$3\tau_{\text{train}}$	$\Delta_{3\tau_{\text{train}}}$	recovered
ScanNet	LitePT-S (Yue et al., 2026)•	74.92	59.37	26.22	−48.70	
	+ relabelling	75.11	71.84	62.21	**−12.90**	73.9 %
	PTv3 (Wu et al., 2024a)†	75.54	57.78	18.96	−56.58	
	+ relabelling	75.60	72.40	64.25	**−11.35**	**80.0** %
	SpUNet (Choy et al., 2019)†	73.98	44.88	18.06	−55.92	
	+ relabelling	73.98	69.48	57.54	**−16.44**	70.6 %
nuScenes	LitePT-S (Yue et al., 2026)•	80.36	58.03	31.72	−48.64	
	+ relabelling	80.38	78.16	73.82	**−6.56**	86.6 %
	PTv3 (Wu et al., 2024a)†	79.18	52.59	28.64	−50.54	
	+ relabelling	79.21	77.54	74.32	**−4.89**	**90.4** %
	SpUNet (Choy et al., 2019)†	74.28	33.29	15.67	−58.61	
	+ relabelling	74.28	72.34	68.05	**−6.23**	89.4 %

The collapse reproduces on all six pairs (Table 2). Relabelling recovers most of it on published and trained weights alike, so the effect does not depend on our training. At $7\tau_{\text{train}}$, it still recovers 50 to 58 % of the collapse on the three frozen nuScenes checkpoints (Table 8).

Finer grids. Finer grids test whether fewer points cause the collapse. Below the training grid, each training cell splits into finer ones, so at least as many points survive as at τ_{train}. Every stock result collapses anyway. At $\tau_{\text{train}}/3$, the six pairs still lose more than fifty points. Indoors, they even fall below their own score at $3\tau_{\text{train}}$ (Table 3, Figure 1). A change of unit predicts that drop, since every operator of Section 3.1 now covers three times fewer metres. SpUNet has no serialization and no rotary phase (Table 1), and it collapses the same way. So does a model trained at $3\tau_{\text{train}}$ and evaluated at τ_{train} (Section 4.6). Relabelled, each training cell keeps one token, as in training, and every pair comes back to within 0.26 point of its relabelled score at τ_{train} (Table 3), so what this regime adds is the collapse before it.

Second task. Instance segmentation tests whether a network carries a unit outside its backbone. A PointGroup head (Jiang et al., 2020) groups points with a radius and a minimum size counted on the training grid. With a frozen SpUNet, AP_{50} falls to exactly 0.00 at $3\tau_{\text{train}}$. Relabelling the backbone recovers 66 % of that collapse, and 74 % with the head's unit scaled too (Appendix B.2).

4.3 Controls

Identity. At τ_{train}, relabelling returns the same tokens. SpUNet scores the same in Table 2. LitePT-S and PTv3 move by at most 0.2 point, as two identical runs do on the GPU (Table 6).

Negative control. PTv2 (Section 3.3) loses 15.57 points at $3\tau_{\text{train}}$ on ScanNet where the three indexed networks lose 49 to 57, and relabelling leaves it unchanged. Tying its pooling cell to the evaluation grid, and changing nothing else, makes the same weights lose 61.84 instead (Table 10). Its drop stays small because its unit does not change, not because of how it picks neighbours.

Table 3: **On finer grids, where at least as many points survive, every stock result still collapses and relabelling recovers the whole drop.** Layout of Table 2, from a separate run with one scene per forward. Its τ_{train} column differs by at most 0.26 point. $\Delta_{\tau_{\text{train}}/3}$ is computed within each row. Bold: relabelled drops.

		evaluation grid τ			
dataset	model	$\tau_{\text{train}}/3$	$\tau_{\text{train}}/2$	τ_{train}	$\Delta_{\tau_{\text{train}}/3}$
ScanNet	LitePT-S$^{\bullet}$	14.47	54.80	75.18	−60.71
	+ relabelling	75.08	75.11	75.14	**−0.06**
	PTv3†	13.33	44.72	75.62	−62.29
	+ relabelling	75.81	75.34	75.60	**+0.21**
	SpUNet†	13.14	37.59	73.98	−60.84
	+ relabelling	73.94	73.91	73.98	**−0.04**
nuScenes	LitePT-S$^{\bullet}$	19.82	43.34	80.43	−60.61
	+ relabelling	80.39	80.45	80.48	**−0.09**
	PTv3†	27.58	47.15	79.19	−51.61
	+ relabelling	78.99	79.06	79.02	**−0.03**
	SpUNet†	20.15	32.78	74.28	−54.13
	+ relabelling	74.29	74.28	74.28	**+0.01**

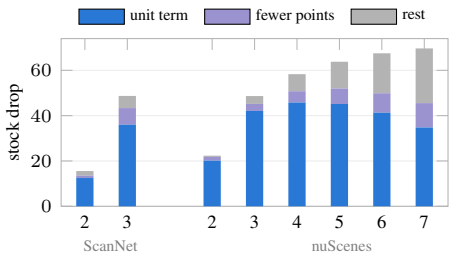


Figure 5: **The unit is the largest part of each stock drop. Outdoors, the rest grows with the grid.** Published LitePT-S, frozen, at $\tau = k\,\tau_{\text{train}}$. *Unit term*: relabelled minus stock score (Equation 4). Fewer points and rest add up to the relabelled drop. *Fewer points*: a random removal of the same fraction at τ_{train}, which keeps more tokens than the grid (Appendix D). *Rest*: the remainder. Indoors it is the draw's protocol advantage, outdoors the grid's choice of tokens (Table 12). Cells: Tables 2, 8 and 14.

4.4 What relabelling does not recover

The relabelled drop is 11 to 16 points at $3\tau_{\text{train}}$ indoors, and 4.9 to 6.6 outdoors (Table 2). The relabelled row is an upper bound on the sampling term of Section 3.3, up to the *sub-cell phase*, the position of each token inside its coarse cell. That phase depends on the grid origin, and moving the origin changes the score by at most 0.57 point (Table 6).

Figure 5 splits each stock drop of the published LitePT-S along Equation 4. The *unit term*, $m_{\text{relab}} - m_{\text{stock}}$, is what relabelling recovers. The relabelled drop, $m_{\text{train}} - m_{\text{relab}}$, splits in two. *Fewer points* is what removing the same fraction of points at random costs at the training grid. The *rest* is the grid choosing which points go, plus any advantage of the draw's protocol. The unit term is the largest part at every grid.

Fewer points and the rest. Indoors at $3\tau_{\text{train}}$, a random removal of the same fraction costs 7.35 points (Appendix D), against 12.90 for the relabelled grid (Table 2). The draw keeps 34 % more tokens and is scored on the points it keeps. With as many tokens as the grid and every point scored, a random draw scores 1.56 points below the grid. Farthest-point and Poisson-disk sampling land within 0.65 of it, and on our PTv3 the grid scores highest of the four (Table 12). All four keep the index at τ_{train} and span at most 3.09 points, against 36 to 45 for the unit. Which points are kept matters far less than the unit they are indexed in. Indoors, the rest of Figure 5 is therefore the draw's extra tokens and scoring. Outdoors, the gap grows with the grid, up to 24 points at $7\tau_{\text{train}}$. There a random draw with as many tokens beats the grid (Table 12), so the rest is the choice of tokens.

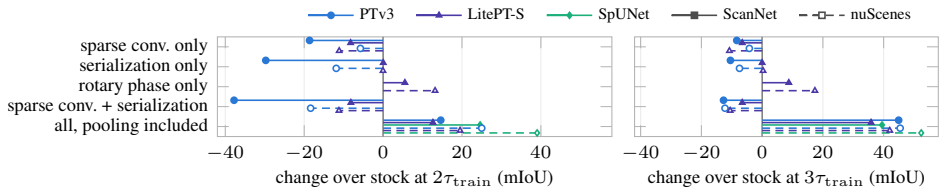


Figure 6: **Only relabelling every operator, pooling included, recovers most of the drop.** Change in mIoU over the stock score of the same frozen weights. Each row relabels only the operators it names, and only the last row relabels the pooling (Table 16). The rotary phase is measured on LitePT-S. SpUNet reads the index once, so it has only the last row.

4.5 Which operators must be relabelled

Relabelling the sparse convolution or the serialization alone recovers nothing on any checkpoint. On PTv3 ScanNet, both fall below the stock score at $3\tau_{\text{train}}$ (Figure 6). The reason is the pooling. It recomputes the index that every later stage inherits. An operator relabelled only where it is first called therefore leaves the later stages on the evaluation grid. The rotary phase escapes this, because we relabel it at every stage. On LitePT-S at $3\tau_{\text{train}}$, it alone recovers 8.8 of the 35.9 points of full relabelling indoors, and 17.4 of 42.0 outdoors (Figure 6). On LitePT-S, relabelling also refills the attention windows of the deep stages. This is not the source of the recovery, since SpUNet has no window and still recovers (Figure 6, Table 17).

Rescaling the index or the scene. On all six pairs, grid rescaling stays 17 to 37 points below relabelling at $3\tau_{\text{train}}$. On PTv3 ScanNet, it loses 18.8 points to stock at $2\tau_{\text{train}}$ (Table 16). Scene shrinking, which returns the stock index, scores within 0.17 point of stock on the three ScanNet backbones (Table 16, Appendix E.1).

4.6 Relabelling and training

We train LitePT-S on ScanNet and LitePT-S*, its mirrored-decoder variant (Appendix A), on nuScenes twice each, with and without multi-resolution augmentation, one seed each (Table 4). This augmentation draws the voxel size from $\{1, 1.5, 2, 3\}\,\tau_{\text{train}}$ at each batch (Table 5). Multi-resolution training shrinks the stock collapse, and relabelling still adds 6.33 points indoors and 5.43 outdoors on top of it at $3\tau_{\text{train}}$ (Table 4). Outdoors, the relabelled multi-resolution models have the smallest drops at $3\tau_{\text{train}}$ in this paper (seed spread in Table 6). Neither approach replaces the other: relabelling needs no retraining, can be combined with it, and applies to published weights.

Table 4: **The two approaches combine, and neither replaces the other.** The same model is trained twice per dataset, one configuration line apart, and evaluated stock and relabelled. The nuScenes pair is LitePT-S*. The last ScanNet row trains the same configuration at $3\tau_{\text{train}}$ (6 cm), evaluated there and at τ_{train}, three times finer than its own grid. One seed per training.

			evaluation grid τ			
dataset	trained model	index	τ_{train}	$2\tau_{\text{train}}$	$3\tau_{\text{train}}$	$\Delta_{3\tau_{\text{train}}}$
ScanNet	no augmentation	stock	74.56	62.42	28.11	−46.45
		relabelled	74.55	71.45	58.84	−15.71
	multi-resolution	stock	74.45	70.74	65.35	−9.10
		relabelled	74.43	73.28	71.68	−2.75
	trained at $3\tau_{\text{train}}$	stock	7.38	—	68.37	—
nuScenes	no augmentation	stock	80.34	54.88	27.69	−52.65
		relabelled	80.39	78.60	74.16	−6.23
	multi-resolution	stock	80.02	78.42	73.36	−6.66
		relabelled	80.20	79.66	78.79	−1.41

Model trained at the coarse grid. Trained at $3\tau_{\text{train}}$ with nothing else changed, LitePT-S loses 6.19 points against the same configuration at τ_{train}, which is the cost of the coarse geometry (Ta-

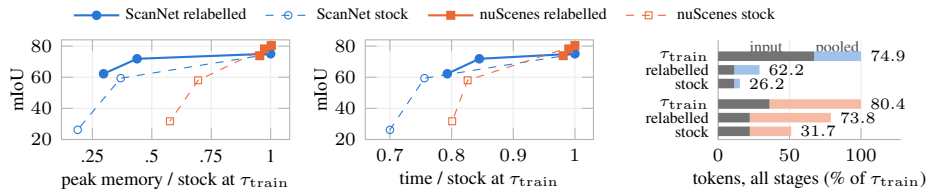


Figure 7: **Indoors, relabelling keeps most of the saving of a coarse grid and little of the drop. Outdoors, there is little to save.** Published LitePT-S, frozen. Left and middle: from τ_{train} (right) to $3\tau_{\text{train}}$ (left). Memory comes from the pass of Table 2, time from a benchmark that includes relabelling. Six pairs: Table 19. Right: tokens at $3\tau_{\text{train}}$ against τ_{train} per stage, no model, mIoU at bar ends.

ble 4). The frozen single-grid model, relabelled, loses 15.71 at $3\tau_{\text{train}}$, 2.5 times as much (Table 4). Its relabelled drop thus exceeds that cost by 9.52 points (Table 4), which training on that grid removes. Multi-resolution training removes most of it. Relabelled, one multi-resolution model scores above the dedicated one at $3\tau_{\text{train}}$. It also keeps its score at τ_{train}, where the dedicated model falls to 7.38 (Table 4).

Invariance through training. Invaria (Chang et al., 2026) targets the same collapse during training. Its own PTv3 baseline falls from 77.6 to 20.5 at $3\tau_{\text{train}}$, the stock collapse diagnosed here. Methods that train for invariance should therefore be compared with a relabelled baseline. Appendix F.1 sets its drops beside ours.

4.7 Cost of relabelling

A coarse grid saves memory and time because fewer points reach the network. At $3\tau_{\text{train}}$ the stock and relabelled networks receive the same points, indoors six times fewer than at τ_{train} (Figure 7, right). After the first pooling, the stock network merges cells three times wider than the trained ones. The relabelled one keeps the trained cells, so its deep stages carry almost as many tokens as at τ_{train}. The saving sits in the first stages. It depends on the fraction of points the coarse grid keeps, which the voxelizer reports before any forward pass. A ScanNet room keeps one point in six at $3\tau_{\text{train}}$. Relabelled, the three networks still divide peak memory by 3.4 to 4.4 and cut inference time by up to 46 % (Table 19). The price is the relabelled drop of Section 4.4. A lidar frame keeps 0.61 of its points, so a coarser grid removes little and the saving does not exist. At $3\tau_{\text{train}}$, relabelled inference is 13 to 38 % slower than stock on the same grid (Appendix G.1).

5 Conclusion

Most of what a voxel-indexed network loses off its training grid is a change of unit in the index it reads, not lost geometry. Relabelling restores the training unit without touching a weight. It gives back 70 to 90 % of the collapse at $3\tau_{\text{train}}$, and all of it below the training grid. The unit must be restored wherever the network reads it: in every operator, since the pooling passes the index to every later stage, and in a head that counts in training cells (Appendix B.2). Relabelling thus separates the two purposes of the voxel size. The index keeps the training unit, and the grid only chooses the points. A coarse grid then saves memory and time wherever it removes points, as in dense indoor rooms. The saving follows the density of the scene, not the network.

One configuration field sets both the cell length and the unit of the index, so a checkpoint should ship with its training grid. A deployment then picks its grid at inference. Robustness methods should be compared with a relabelled baseline, since most of the collapse they start from is a unit mismatch.

Limitations. Relabelling needs the physical coordinate at inference. It only renumbers when the grid is an integer multiple of the training grid, and it merges a few tokens otherwise (Table 7).

Future work. The grid's choice of points costs more as the grid grows (Figure 5, Table 8). A hybrid, with a coarse grid at the input and a token selection inside attention, could keep the saving of the grid and give back part of the sampling term.

REPRODUCIBILITY STATEMENT

The released evaluation tool builds the same network inputs as the pipeline behind our tables, bit for bit; code and weights are at `https://anonymous.4open.science/r/ICLR2027-76F4/`. The paper contains what is needed to reproduce the frozen-weight result. The operation is stated in closed form in Section 3.2, which also names the coordinate it reads, the origin, and the training grid τ_{train}. Section 3.3 defines the two rescalings it is measured against, and Section 4.5 and Table 16 give their scores. Section 4.1 gives the datasets, the evaluation splits, the single GPU and the number of seeds, and Appendix A lists every set of weights with its configuration, the test-time augmentation protocol, the two configuration keys that change a PTv3 score, which readings carry a seed of ours at all, and the noise level of every measurement (Table 6). The limitations in Section 5 state what bounds the result. Two of the checkpoints we read are published by their authors and we did not modify them; the rest derive from public Pointcept configurations with no architectural change.

AI USE STATEMENT

Large language models were used as writing and coding assistants. On the writing side they revised prose and helped make the figures. On the code side they helped write and refactor the code, evaluation scripts included. The authors checked every script, and every number it produced, before it entered the paper.

They did not produce the research ideas, design the experiments, or produce any measurement. They did not choose which results are reported, and they did not decide what any measurement means. Every number here comes from the runs of Section 4. The authors verified each reported value against its source file, and they take full responsibility for the content of this paper, its errors included.

REFERENCES

Maxim Berman, Amal Rannen Triki, and Matthew B. Blaschko. The Lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4413–4421, 2018. doi: 10.1109/CVPR.2018.00464.

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. FlexiViT: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14496–14506, 2023. doi: 10.1109/CVPR52729.2023.01393.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations (ICLR)*, 2023.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Chun-Peng Chang, Shaoxiang Wang, Alain Pagani, Dariu Gavrila, and Holger Caesar. Invaria: Learning scale and density invariance in point clouds via next-resolution prediction. *arXiv preprint arXiv:2605.15923*, 2026.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3075–3084, 2019.

Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2990–2999, 2016.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9224–9232, 2018.

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pp. 289–305, 2024. doi: 10.1007/978-3-031-72684-2_17.

Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11105–11114, 2020. doi: 10.1109/CVPR42600.2020.01112.

Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15546–15566, 2023.

Jaeyeul Kim, Jungwan Woo, Jeonghoon Kim, and Sunghoon Im. Rethinking LiDAR domain generalization: Single source as multiple density domains. In *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pp. 310–327, 2024. doi: 10.1007/978-3-031-72661-3_18.

Lingdong Kong, Youquan Liu, Xin Li, Runnan Chen, Wenwei Zhang, Jiawei Ren, Liang Pan, Kai Chen, and Ziwei Liu. Robo3D: Towards robust and reliable 3D perception against corruptions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19937–19949, 2023. doi: 10.1109/ICCV51070.2023.01830.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *Journal of Machine Learning Research*, 24(89), 2023.

Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3D point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8490–8499, 2022. doi: 10.1109/CVPR52688.2022.00831.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han. FlatFormer: Flattened window attention for efficient point cloud transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Yu Luo, Kun Hu, Mengwei He, Xiaogang Zhu, Shan Zeng, Allen Benter, Wei Xiang, Patrick Filippi, Thomas Francis Bishop, and Zhiyong Wang. GeoRoPE: Ground-aware rotary adaptation for remote sensing foundation models. *arXiv preprint arXiv:2606.14760*, 2026.

M. Jehanzeb Mirza, Inkyu Shin, Wei Lin, Andreas Schriebl, Kunyang Sun, Jaesung Choe, Horst Possegger, Mateusz Koziński, In So Kweon, Kun-Jin Yoon, and Horst Bischof. MATE: Masked autoencoders are online 3D test-time learners. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Bohao Peng, Xiaoyang Wu, Li Jiang, Yukang Chen, Hengshuang Zhao, Zhuotao Tian, and Jiaya Jia. OA-CNNs: Omni-adaptive sparse CNNs for 3D semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Pointcept Contributors. Pointcept: A codebase for point cloud perception research. `https://github.com/Pointcept/Pointcept`, 2023.

Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J. Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *International Conference on Learning Representations (ICLR)*, 2022.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas 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 (CVPR)*, pp. 77–85, 2017a. doi: 10.1109/CVPR.2017.16.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017b.

Colorado J. Reed, Ritwik Gupta, Shufan Li, Sarah Brockman, Christopher Funk, Brian Clipp, Kurt Keutzer, Salvatore Candido, Matt Uyttendaele, and Trevor Darrell. Scale-MAE: A scale-aware masked autoencoder for multiscale geospatial representation learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Laurenz Reichardt, Nikolas Ebert, and Oliver Wasenmüller. PointTransformerX: Portable and efficient 3D point cloud processing without sparse algorithms. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2026. arXiv:2604.24169.

Cristiano Saltori, Evgeny Krivosheev, Stéphane Lathuilière, Nicu Sebe, Fabio Galasso, Giuseppe Fiameni, Elisa Ricci, and Fabio Poiesi. GIPSO: Geometrically informed propagation for online adaptation in 3D LiDAR segmentation. In *European Conference on Computer Vision (ECCV)*, 2022.

Hajin Shim, Changhun Kim, and Eunho Yang. CloudFixer: Test-time adaptation for 3D point clouds via diffusion-guided geometric transformation. In *European Conference on Computer Vision (ECCV)*, 2024.

Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. KPConv: Flexible and deformable convolution for point clouds. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Tuan Anh Tran, Duy M. H. Nguyen, Hoai-Chau Tran, Michael Barz, Khoa D. Doan, Roger Wattenhofer, Ngo Anh Vien, Mathias Niepert, Daniel Sonntag, and Paul Swoboda. How many tokens do 3D point cloud transformer architectures really need? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 13464–13496, 2025. doi: 10.52202/085713-0454.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations (ICLR)*, 2021.

Peng-Shuai Wang. OctFormer: Octree-based transformers for 3D point clouds. *ACM Transactions on Graphics*, 42(4), 2023. doi: 10.1145/3592131.

Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q. Weinberger, and Wei-Lun Chao. Train in Germany, test in the USA: Making 3D object detectors generalize. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Yanshuo Wang, Ali Cheraghian, Zeeshan Hayder, Jie Hong, Sameera Ramasinghe, Shafin Rahman, David Ahmedt-Aristizabal, Xuesong Li, Lars Petersson, and Mehrtash Harandi. Backpropagation-free network for 3D test-time adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.

Xiaoyang Wu, Zhuotao Tian, Xin Wen, Bohao Peng, Xihui Liu, Kaicheng Yu, and Hengshuang Zhao. Towards large-scale 3d representation learning with multi-dataset point prompt training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22193–22204, 2025. Highlight.

Yuanwen Yue, Damien Robert, Jianyuan Wang, Sunghwan Hong, Jan Dirk Wegner, Christian Rupprecht, and Konrad Schindler. LitePT: Lighter yet stronger point transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. Highlight.

Yujia Zhang, Xiaoyang Wu, Yunhan Yang, Xianzhe Fan, Han Li, Yuechen Zhang, Zehao Huang, Naiyan Wang, and Hengshuang Zhao. Utonia: Toward one encoder for all point clouds. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*, 2018.

Tianpei Zou, Sanqing Qu, Zhijun Li, Alois Knoll, Lianghua He, Guang Chen, and Changjun Jiang. HGL: Hierarchical geometry learning for test-time adaptation in 3D point cloud segmentation. In *European Conference on Computer Vision (ECCV)*, 2024.

OVERVIEW

Each appendix follows one part of Section 4. Appendix A gives the configurations and noise levels (Section 4.1). Appendix B covers coarser and non-integer grids and the second task (Section 4.2). Appendix C gives the controls (Section 4.3), and Appendix D what relabelling does not recover (Section 4.4). Appendix E covers the operators (Section 4.5), Appendix F training (Section 4.6), and Appendix G the cost (Section 4.7).

A SETUP AND MEASUREMENT NOISE

Every training derives from a public Pointcept configuration with no architectural change (Table 5). We train on one GPU: where our batch size is smaller than the public file's, the learning rate is left as published. The LitePT-S* nuScenes file is ours, the mirrored decoder on the LitePT-S nuScenes configuration.

Table 5: **The sixteen configurations used in this paper.** `OneCycleLR` throughout. LitePT-S* on ScanNet and both nuScenes LitePT-S rows are trained twice (seed spread, Table 6). *aug.*: voxel size sampled in $\{1, 1.5, 2, 3\}\,\tau_{\text{train}}$. *Lov.*: Lovász-softmax (Berman et al., 2018). LitePT-S*: mirrored convolutional decoder, 16.0 M. $^\bullet$ published. § Appendix C.1. ¶ PointGroup head (Appendix B.2). ‡ trained at $3\tau_{\text{train}}$.

	model	Pointcept configuration	optim.	lr	*wd*	loss	ep. / batch
ScanNet	LitePT-S$^\bullet$	`semseg-litept-v1m1-0-small`	AdamW	0.006	0.05	CE+Lov.	1200 / 6
	LitePT-S	`semseg-litept-v1m1-0-small`	AdamW	0.006	0.05	CE+Lov.	1200 / 6
	+ aug.	`...-v1m1-0b-small-scaleaug`	AdamW	0.006	0.05	CE+Lov.	1200 / 6
	PTv3	`semseg-pt-v3m1-0-base`	AdamW	0.006	0.05	CE+Lov.	800 / 6
	SpUNet	`semseg-spunet-v1m1-0-base`	SGD	0.05	10^{-4}	CE	800 / 12
	PTv2§	`semseg-pt-v2m2-0-base`	AdamW	0.005	0.02	CE+Lov.	900 / 12
	PointGroup¶	`insseg-pointgroup-v1m1-0`	SGD	0.1	10^{-4}	CE	800 / 12
	LitePT-S*	`semseg-litept-v1m2-0-small`	AdamW	0.006	0.05	CE+Lov.	1200 / 12
	LitePT-S‡	`...-v1m1-0-small-train3tau`	AdamW	0.006	0.05	CE+Lov.	1200 / 6
nuScenes	LitePT-S$^\bullet$	`semseg-litept-v1m1-0-small`	AdamW	0.002	0.005	CE+Lov.	50 / 6
	LitePT-S	`semseg-litept-v1m1-0-small`	AdamW	0.002	0.005	CE+Lov.	50 / 6
	+ aug.	`...-v1m1-0b-small-scaleaug`	AdamW	0.002	0.005	CE+Lov.	50 / 6
	LitePT-S*	`semseg-litept-v1m2-0-small`	AdamW	0.002	0.005	CE+Lov.	50 / 6
	+ aug.	`...-v1m2-0b-small-scaleaug`	AdamW	0.002	0.005	CE+Lov.	50 / 6
	PTv3	`semseg-pt-v3m1-0-base`	AdamW	0.002	0.005	CE+Lov.	50 / 6
	SpUNet	`semseg-spunet-v1m1-0-base`	AdamW	0.002	0.005	CE	50 / 12

A.1 TEST-TIME AUGMENTATION

Test-time augmentation is used only in Appendix F.1 and for the two reference scores below. ScanNet uses 13 views: four vertical rotations crossed with scales 0.95, 1, 1.05, plus a mirror. nuScenes uses 10: five scales from 0.9 to 1.1, with and without a mirror. Evaluated this way, our PTv3 gives 77.10 against the 77.6 its authors publish, with `enable_flash=True` and `enable_rpe=False`, two keys that change a PTv3 score. The published LitePT-S gives 76.54 on ScanNet and 82.02 on nuScenes against its reported 76.5 and 82.2. Our starting points therefore match the published models.

A.2 EVALUATION RUNS AND NOISE

An evaluation run is one call of the evaluation command. Tables that use another run say so. Table 6 gives the noise levels every comparison is judged against, and we claim no gap below them.

B RECOVERY ON FROZEN WEIGHTS

B.1 COARSER AND NON-INTEGER GRIDS

Table 7 gives the half-integer ratios of Section 3.2, and Table 8 the grids beyond $3\tau_{\text{train}}$ of Section 4.2.

B.2 SECOND TASK

Instance segmentation gives each point a class and assigns it to an object. In PointGroup, the backbone predicts each point's class and its offset to the object's centre. A head then links the shifted

Table 6: **The noise level of our measurements.** S: LitePT-S. Run-to-run: range over eight evaluation runs of S on ScanNet. Passes: largest gap between our evaluation runs of one checkpoint. Seed: two trainings of S* indoors and of S outdoors, the variants Table 4 does not use. Indoors the cell gives stock / relabelled, and outdoors the range over rows. Origin: largest deviation over the offsets tested.

quantity	τ_{train}	$3\tau_{\text{train}}$
run-to-run, 8 runs	0.34	0.34
passes, PTv3 nuSc.	0.40	—
passes, PTv3 ScanNet	0.42	—
seed, S* ScanNet	—	0.83 / 0.73
seed, S nuScenes	0.02	0.25 to 1.71
origin, PTv3 nuScenes	—	0.10
origin, SpUNet nuScenes	—	0.07
origin, PTv3 ScanNet	—	0.48
origin, S ScanNet	—	0.57

Table 7: **The integer condition is exact, not the boundary of usefulness.** LitePT-S published, ScanNet, a second evaluation run. Merged: stock minus relabelled tokens per scene. No token merges at integer τ/τ_{train}. At 1.0 both columns receive the same tokens and differ within the run-to-run spread (Table 6).

τ/τ_{train}	merged	stock	relab.	gain
1.0	0	75.11	74.91	−0.20
1.5	3 839 (5.9%)	72.01	74.47	+2.46
2.0	0	59.35	72.01	+12.66
2.5	508 (1.9%)	39.73	68.19	+28.46
3.0	0	26.22	62.01	+35.79
3.5	133 (1.0%)	18.91	53.60	+34.69

Table 8: **From $4\tau_{\text{train}}$ to $7\tau_{\text{train}}$ relabelling still recovers more than 34 points.** Frozen nuScenes checkpoints, a second evaluation run. At τ_{train} it lies within 0.06 point of Table 2.

		τ_{train}	$4\tau_{\text{train}}$	$5\tau_{\text{train}}$	$6\tau_{\text{train}}$	$7\tau_{\text{train}}$
LitePT-S	stock	80.39	22.09	16.61	12.88	10.73
	relab.	80.42	67.88	61.76	54.16	45.51
PTv3	stock	79.12	19.47	15.51	12.79	10.81
	relab.	79.15	69.10	63.19	56.38	50.21
SpUNet	stock	74.28	9.36	7.86	7.37	7.18
	relab.	74.28	61.98	55.94	50.91	45.38

points closer than a radius into groups above a minimum size. The backbone reads the index. The head counts its radius in training cells and its group size in training-grid points, which is a second unit tied to the training grid. We use a frozen SpUNet backbone on ScanNet. Its average precision at an overlap of 0.5 (AP_{50}) is exactly 0.00 at $3\tau_{\text{train}}$, because no group reaches the minimum size. Relabelling the backbone's index recovers two thirds of that collapse. Scaling the head's radius and group size to the evaluation grid as well recovers three quarters. Without relabelling, that scaling recovers 4.4 % of it, so the two corrections do not replace each other (Table 9). The instance-segmentation result rests on one backbone, one dataset and one checkpoint.

Table 9: **Instance segmentation (AP_{50}) collapses to nothing, and both units have to be restored.** SpUNet and PointGroup, frozen, 312 ScanNet scenes. *index*: read by the backbone. *head*: linking radius and minimum group size. *recovered*: as in Table 2. Bold: the scores the text names.

index	head	τ_{train}	$3\tau_{\text{train}}$	*recovered*
stock	as released	62.33	**0.00**	—
stock	scaled	61.79	2.72	4.4 %
relabelled	as released	62.08	41.26	66.2 %
relabelled	scaled	62.09	**46.12**	**74.0** %

The instance-segmentation pair of Table 9 uses the configuration of Table 5. The semantic branch of the same forward pass recovers 75.6 % of its own collapse, inside the range of Table 2.

C CONTROLS

C.1 NEGATIVE CONTROL

All three backbones of Table 2 read the integer index. PTv2 (Wu et al., 2022) reads the coordinate in metres and should lose only what the voxelization removed. The prediction is recorded before the run, and has two parts. Relabelling should move it by less than the indoor seed spread. Its stock drop at $3\tau_{\text{train}}$ should match the relabelled drops of the indexed backbones, 11 to 16 points, not their stock 49 to 57. A recovery of tens of points would refute the diagnosis. PTv2 loses 15.57 points, inside that band, and relabelling moves every cell by 0.00 (Table 10). PTv2's configuration feeds the network only the coordinate, in metres, so that zero holds by construction. The drop is the informative number, measured on ScanNet.

PTv2 finds neighbours by nearest-neighbour search, but pools by a cell in metres. Multiplying that cell by the grid ratio moves the unit as an integer stride does and leaves the neighbourhood rule alone. The same weights then fall to 12.07 at $3\tau_{\text{train}}$ (Table 10): PTv2's stock drop stays small because its unit stays in metres.

Table 10: **PTv2, with its unit tied to the grid.** ScanNet, frozen. In metres: the released pooling, stock and relabelled. Tied to grid: pooling cells multiplied by the grid ratio, nothing else changed.

	in metres		
grid	stock	relabelled	tied to grid
τ_{train}	73.91	73.91	73.91
$2\tau_{\text{train}}$	69.74	69.74	47.00
$3\tau_{\text{train}}$	58.34	58.34	12.07
$\Delta_{3\tau_{\text{train}}}$	−15.57	−15.57	−61.84

C.2 SPECIFICITY

Robustness benchmarks simulate a sparser sensor by dropping points at random (Kong et al., 2023). Thinning or shrinking the scene leaves the voxel size unchanged, so Equation 3 returns the stock index up to the origin. Over twelve paired evaluations on two nuScenes checkpoints, relabelling moves the score by at most 0.32 point, and by zero on SpUNet (Table 11). The perturbations themselves cost tens of points.

Table 11: **At a fixed voxel size, relabelling moves the score by at most 0.32 point.** Density: points dropped at random, keeping the fraction shown. Scale: the whole scene shrunk by the factor shown. nuScenes, 6019 frames, two frozen checkpoints, one evaluation run each. On SpUNet the two readings are equal to the hundredth, so each cell gives both.

		PTv3			SpUNet
perturbation	factor	stock	relabelled	gap	stock and relabelled
density	1.00	79.01	79.09	+0.08	74.28
	0.50	74.62	74.77	+0.15	68.14
	0.25	59.08	59.09	+0.01	53.67
scale	1.00	78.97	78.95	−0.02	74.28
	0.50	49.74	49.42	**−0.32**	33.05
	0.33	23.94	23.96	+0.02	17.73

D WHAT RELABELLING DOES NOT RECOVER

Take the fraction of points that $3\tau_{\text{train}}$ leaves indoors. A random draw of that fraction at the training grid costs 7.35 points, and the relabelled grid 12.90, on the same checkpoint (Table 14). The sub-cell phase in the 12.90 is worth at most 0.57 (Table 6). The draw keeps 34 % more tokens (Table 13)

and is scored only on the points it keeps. Table 12 removes both differences. Each sampler keeps as many of the tokens of τ_{train} as the grid does, and the index stays at τ_{train}. Every original point takes the prediction of its nearest kept token. The grid then scores 62.41, 0.20 from Table 2. Scored on every point, the draw of the same fraction of points falls from 67.85 to 66.69. With as many tokens as the grid, it falls to 60.85, below the grid. Outdoors, the draw keeps only 6 % more tokens. There, the random removal and the relabelled grid agree at $2\tau_{\text{train}}$ and differ beyond it. Beyond $2\tau_{\text{train}}$, the grid leaves far fewer occupied neighbours than the draw (Table 13). With as many tokens as the grid, the draw still scores 1.99 points above it at $3\tau_{\text{train}}$ and 18.88 at $7\tau_{\text{train}}$ (Table 12). The fraction of points alone therefore does not account for the relabelled drop (Table 14).

Table 12: **Indoors, the four samplers lie within** 3.09 **points of each other. Outdoors, the random draw and Poisson-disk sampling score above the grid.** Published LitePT-S and our PTv3, frozen, one evaluation run each. Each sampler keeps as many of the tokens of τ_{train} as the scene occupies cells of $k\,\tau_{\text{train}}$. The index stays at τ_{train}, and every original point is scored by its nearest kept token. At $k = 1$ all tokens are kept: grid 74.92, random 75.15, stock 75.16 (PTv3: 75.96, 75.56, 75.67; nuScenes: 80.37, 80.30, 80.36).

	ScanNet				nuScenes	
	LitePT-S			PTv3	LitePT-S	
sampler	$2\tau_{\text{train}}$	$3\tau_{\text{train}}$	peak MB, $3\tau_{\text{train}}$	$3\tau_{\text{train}}$	$3\tau_{\text{train}}$	$7\tau_{\text{train}}$
grid, relabelled	72.02	62.41	276	64.61	73.43	45.25
random	71.77	60.85	269	64.23	75.42	64.13
farthest point	72.17	61.76	287	61.52	71.62	42.44
Poisson disk	72.33	63.06	278	62.03	75.97	54.31

The grid needs no selection step of its own: it is the voxelization that runs in any case.

Tran et al. (2025) report a drop of 7.5 points on PTv3 ScanNet when one point in five is kept at random, against our 7.35 at 0.17. Their grid row averages each cell, so it is a merge. Their backbone and protocol differ from ours, so the two figures are close but not directly comparable. With as many tokens as the grid and every point scored, our relabelled grid ranks ahead of the draw (Table 12), as their grid does.

Table 13: **At a matched fraction of points, the grid leaves fewer occupied first-stage taps than a random draw.** Occupied taps of the 26 of a $3\times3\times3$ kernel at stage 0, and share of tokens with none. No model is run. The draw keeps the grid's fraction, hence more tokens.

	setting	tokens	taps	isolated
ScanNet	τ_{train}, stock	34.2 M	7.78	0.1 %
	$3\tau_{\text{train}}$, relabelled	5.8 M	0.77	44.3 %
	draw, keep 0.17	7.8 M	2.06	17.3 %
nuSc.	τ_{train}, stock	131.3 M	1.60	37.9 %
	$3\tau_{\text{train}}$, relabelled	80.9 M	0.24	78.7 %
	draw, keep 0.61	85.9 M	1.28	43.4 %

Table 14: **Outdoors, the relabelled drop matches a random removal at $2\tau_{\text{train}}$ and exceeds it beyond.** LitePT-S, frozen: relabelled grid against a draw keeping the same fraction at the training grid, scored on the points it keeps. Relabelled column: Table 2 to $3\tau_{\text{train}}$, Table 8 beyond. ScanNet 0.61: the outdoor fraction at $3\tau_{\text{train}}$, which no indoor grid keeps.

grid	kept	relab.	draw	gap
nuScenes				
τ_{train}	1.000	80.38	80.43	—
$2\tau_{\text{train}}$	0.756	78.16	78.78	0.62
$3\tau_{\text{train}}$	0.616	73.82	77.34	3.52
$4\tau_{\text{train}}$	0.522	67.88	75.45	7.57
$5\tau_{\text{train}}$	0.453	61.76	73.62	11.86
$6\tau_{\text{train}}$	0.401	54.16	71.83	17.67
$7\tau_{\text{train}}$	0.359	45.51	69.68	24.17
ScanNet				
τ_{train}	1.00	75.11	75.20	—
—	0.61	—	75.03	—
$2\tau_{\text{train}}$	0.36	71.84	74.15	2.31
$3\tau_{\text{train}}$	0.17	62.21	67.85	5.64

D.1 VOXELIZER

Equation 3 uses a real surviving point. A voxelizer that returns each cell's centroid leaves the stock score unchanged. It lowers the relabelled LitePT-S score at $3\tau_{\text{train}}$ from 62.21 to 56.76 (75.10 at τ_{train}, Table 15). The centroid therefore affects only the relabelled score. The stock index of a centroid is that of its coarse cell, whatever the centroid. Relabelled, a centroid lies closer to the middle of its coarse cell than a random point. Its index therefore drifts towards the rescaled index of Section 3.3. That index scores 38.29 at $3\tau_{\text{train}}$ on this checkpoint (Table 16).

Table 15: **A centroid voxelizer changes only the relabelled score.** Published LitePT-S, ScanNet, frozen, one evaluation run per row. Real point: the voxelizer of every other table (Table 21).

	stock	relabelled	
voxelizer returns	$3\tau_{\text{train}}$	τ_{train}	$3\tau_{\text{train}}$
a real point	26.22	75.11	62.21
the cell's centroid	26.22	75.10	56.76

E OPERATORS AND ATTENTION WINDOW

Each row of Table 16 relabels the named operators only, at their call site. Only the last row includes the pooling (Section 4.5). On LitePT-S, the serialization row gives the stock score. That backbone serializes again after every pooling that feeds attention, and its rotary phase also reads the index.

Table 16: **A factor is not enough, and only relabelling every operator recovers most of the drop.** Frozen weights. Each row relabels the named operators only: the rotary phase at every stage, the other operators at their single call site. Rescaling multiplies the index by k. † scene shrunk by τ_{train}/τ (Utonia), a separate evaluation run. At τ_{train} every row is within 0.04 point of stock, the separate runs within 0.19. This tool recomputes the index in place, so columns compare within a pair. Its first and last rows differ from Table 2 by at most 0.53 point. n/a: the backbone has no such operator, or, for SpUNet, reads the index once.

	PTv3 ScanNet		PTv3 nuScenes		LitePT-S ScanNet		LitePT-S nuScenes		SpUNet ScanNet		SpUNet nuScenes	
relabelled τ/τ_{train}	2	3	2	3	2	3	2	3	2	3	2	3
nothing (stock)	57.61	19.13	52.61	28.78	59.32	26.22	58.13	31.78	44.88	18.06	33.29	15.67
rescaling by k	38.84	26.90	68.72	57.15	64.70	38.29	67.41	54.81	66.65	35.69	50.54	36.39
shrinking†	57.64	19.15	—	—	59.40	26.05	—	—	44.81	18.20	—	—
sparse conv. only	38.97	10.77	46.86	24.57	51.08	19.67	47.01	21.01	n/a	n/a	n/a	n/a
serialization only	27.87	8.69	40.73	21.33	59.37	26.22	58.11	32.09	n/a	n/a	n/a	n/a
rotary phase only	n/a	n/a	n/a	n/a	64.85	35.01	71.30	49.15	n/a	n/a	n/a	n/a
sparse conv. + serialization	19.83	6.42	34.24	16.55	51.15	19.67	46.94	21.24	n/a	n/a	n/a	n/a
all, pooling included	72.23	64.07	77.62	74.22	71.96	62.09	77.63	73.80	69.48	57.54	72.34	68.05

E.1 INDEX RESCALING AND SHRINKING

Section 3.3 defines both fixes, with $k = \tau/\tau_{\text{train}}$ (Figure 8). Shrinking the scene gives exactly the stock index, so it cannot repair the unit.

Index rescaling multiplies the coarse index by k. On PTv3 ScanNet it gains 7.8 points over the stock score at $3\tau_{\text{train}}$ but loses 18.8 at $2\tau_{\text{train}}$, while relabelling gains at both grids. On the five other pairs, it gains at both grids and stays 17 to 32 points below relabelling at $3\tau_{\text{train}}$ (Table 16). It repairs distances, but every token lands on a multiple of k. Tokens lose where the point sits inside its cell, and a kernel of radius one cell finds no neighbour until the first pooling. At $2\tau_{\text{train}}$, the stock score still has much to lose, and emptying adjacency costs more than repairing distances gains. At $3\tau_{\text{train}}$, little is left to lose, and the repair wins. At $k = 2$, the serialization order, and with it the attention windows, also stays as in the stock evaluation. At $k = 3$, it changes. Relabelling recomputes the index from each coordinate, so it repairs distances without emptying adjacency.

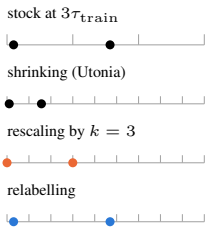


Figure 8: **Four ways to index two points** 4.4 **training cells apart, at** $3\tau_{\text{train}}$**.** Ticks: the cells the index counts. Only relabelling puts the points about 4 cells apart.

E.2 Attention window

On LitePT-S, a ScanNet room fills a quarter of its attention window at $3\tau_{\text{train}}$. Relabelling fills it again while it corrects the unit (Table 17). Capping every window at 239 tokens at the training grid costs 3.63 points, a tenth of the 36.10 the coordinate recovers. Neutralizing the window while keeping the coordinate coarse recovers nothing on any pair. SpUNet, which has no window, still recovers. The recovery therefore does not come from refilling the window.

Table 17: **The attention window is not what relabelling repairs.** Tokens a window receives at $3\tau_{\text{train}}$ of the 1024 it asks for, stock then relabelled (983 at τ_{train} on ScanNet LitePT-S). Then, what each operation recovers alone against stock, in points. *Window alone* restores the window's physical extent and keeps the coarse index. *Coordinate alone* does the reverse. A separate evaluation run from Table 2. At τ_{train}, capping every window of LitePT-S ScanNet at 239 tokens costs 3.63 points.

backbone	dataset	tokens per window	stock collapse	window alone	coordinate alone
LitePT-S	ScanNet	$239 \rightarrow 979$	49.00	-2.45	$+36.10$
PTv3	ScanNet	$1024 \rightarrow 1024$	56.54	-3.47	$+44.95$
SpUNet	ScanNet	no window	55.92	$+0.00$	$+39.48$
LitePT-S	nuScenes	$1017 \rightarrow 1024$	48.71	-4.40	$+42.02$
PTv3	nuScenes	$1024 \rightarrow 1024$	50.59	-5.54	$+45.44$
SpUNet	nuScenes	no window	58.61	$+0.00$	$+52.38$

F Training

Multi-resolution training lowers both the unit term and the relabelled drop (Table 4, Section 4.6). Each training is one seed. The 6.33 points relabelling adds indoors on top of multi-resolution training are more than seven times the seed spread of Table 6. At $3\tau_{\text{train}}$, relabelling closes $76\,\%$ of the gap between our stock score and the model trained at $3\tau_{\text{train}}$. With the published checkpoint, it closes $85\,\%$ (Tables 4 and 2). On nuScenes, Table 4 uses LitePT-S*. The LitePT-S pair behind the outdoor seed spread of Table 6 gives the same result. At $3\tau_{\text{train}}$, relabelling adds 5.24 and 5.39 points on top of multi-resolution training for its two seeds, and the relabelled models then lose only 0.61 and 0.84 point.

F.1 Invaria as context

Invaria (Chang et al., 2026) is a preprint without released code or evaluation protocol. Its architectures differ from ours, so only the drop can be compared (Table 18). Evaluated with test-time augmentation, our relabelled drop moves by at most 1.61 point indoors and 0.94 outdoors, so the comparison is a magnitude, not a ranking. Its reported drops are 8.7 to 8.9 points smaller than our relabelled ones indoors, and 1.22 and 1.32 points smaller outdoors. On the augmentation path relabelling applies only for $\tau > \tau_{\text{train}}$, so at τ_{train} it is the identity by construction.

Table 18: **Only the drop can be compared.** Δ_τ from τ_{train} to $3\tau_{\text{train}}$. Invaria: numbers of Chang et al. (2026). Ours: with test-time augmentation.

model	#params	ScanNet	nuScenes
Invaria	25.5 M	−2.40	−4.30
Invaria-s	16.3 M	−2.60	−4.40
LitePT-S, stock	12.7 M	−49.89	−47.63
LitePT-S, relabelled	12.7 M	−11.29	−5.62

G Cost and relabelling size

Outdoors, relabelling restores a fine index grid over a large scene. A hundred-metre scene at 5 cm has two thousand cells per axis (points kept: Table 19). At $3\tau_{\text{train}}$, the relabelled nuScenes network still processes 79 % of the tokens of τ_{train} (Figure 7). The coarse cloud is already as sparse as a pooled one. Outdoors, the deep levels are therefore limited by the number of cells, and the saving disappears.

Table 19: **A coarse indoor grid keeps its saving once relabelled. Outdoors, it does not.** $3\tau_{\text{train}}$ relabelled against τ_{train} stock. Speed-up: time of τ_{train} stock over time of $3\tau_{\text{train}}$ relabelled, 14 paired rounds of one timing session, 95 % interval. Memory: peak of one forward pass, which batches several scenes, highest over the validation set, in the evaluation run of Table 2. Kept: points left at $3\tau_{\text{train}}$, LitePT-S, timing benchmark.

	kept	model	memory	speed-up
ScanNet	0.17	LitePT-S	3.36×	1.262× [1.258, 1.265]
		PTv3	4.43×	1.616× [1.612, 1.619]
		SpUNet	4.32×	1.852× [1.849, 1.856]
nuScenes	0.61	LitePT-S	1.05×	1.021× [1.019, 1.023]
		PTv3	1.06×	1.044× [1.044, 1.045]
		SpUNet	1.09×	1.045× [1.044, 1.047]

G.1 Time

In a second timing session (21 paired rounds over three grid origins), relabelling makes inference 13 to 38 % slower than the stock evaluation of the same grid at $3\tau_{\text{train}}$. At τ_{train}, it costs nothing.

G.2 Operation count

Tran et al. (2025) merge tokens inside the attention block of PTv3, which attends at every stage. They report 107.5 down to 19.75 GFLOPs for 77.68 down to 76.24 mIoU on ScanNet. LitePT (Yue et al., 2026) attends only in its two deepest encoder stages, where pooling has already reduced the tokens. Merging inside attention has less to act on there. Evaluated stock, a coarse grid divides the operation count by 8.38 indoors and 2.57 outdoors. Relabelled, the count falls by 1.46 and 1.04 (Table 20), so relabelling keeps 36 % of the indoor saving and 6 % of the outdoor one. Kernels are counted dense, so these are bounds. Merging acts at a fixed grid, on a different quantity than the unit, so the two can combine.

G.3 Relabelling at a size other than τ_{train}

Equation 3 divides by τ_{train}, but it accepts any other size in its place. Only τ_{train} itself leaves the score at the training grid unchanged. Every other size costs points there, in both directions, because it then sets the unit at every grid (Table 21). The failure comes from the operation itself, not from these six checkpoints, so a wrong size shows up before deployment. On the timing benchmark, read at $3\tau_{\text{train}}$, relabelling at $1.5\,\tau_{\text{train}}$ lowers the LitePT-S peak by 67.3 MB indoors and 53.9 MB outdoors against relabelling at τ_{train} (Table 22). Relabelling at $1.5\,\tau_{\text{train}}$ costs 0.11 mIoU point

Table 20: **Operation count at each grid.** LitePT-S, GFLOPs of one forward pass on the first validation scene, sparse convolutions counted with dense kernels. In brackets: the count at τ_{train} over the count at the grid.

dataset	pipeline	τ_{train}	$2\tau_{\text{train}}$	$3\tau_{\text{train}}$
ScanNet	stock	187.1	50.1 (3.73×)	22.3 (8.38×)
	relabelled	187.1	148.5 (1.26×)	128.3 (1.46×)
nuScenes	stock	230.2	130.0 (1.77×)	89.4 (2.57×)
	relabelled	230.2	228.7 (1.01×)	221.9 (1.04×)

indoors and 5.41 outdoors at $3\tau_{\text{train}}$ (Table 21), so it is an option a deployment may choose, not the default.

Table 21: **Only relabelling at τ_{train} keeps the training-grid score.** Frozen pairs, evaluated at τ_{train} and $3\tau_{\text{train}}$ in one evaluation run. Bold: the default every other table uses. —: not swept.

				relabelled at			
backbone	dataset	read at	stock	$\frac{1}{2}\,\tau_{\text{train}}$	τ_{train}	$1.5\,\tau_{\text{train}}$	$2\,\tau_{\text{train}}$
PTv3	ScanNet	τ_{train}	75.54	41.33	**75.60**	72.04	57.40
		$3\tau_{\text{train}}$	18.96	6.29	**64.25**	61.46	42.44
	nuScenes	τ_{train}	79.18	46.13	**79.21**	72.43	—
		$3\tau_{\text{train}}$	28.64	33.22	**74.32**	68.52	—
LitePT-S	ScanNet	τ_{train}	74.92	51.17	**75.11**	71.78	59.02
		$3\tau_{\text{train}}$	26.22	8.30	**62.21**	62.10	48.09
	nuScenes	τ_{train}	80.36	39.40	**80.38**	74.17	57.92
		$3\tau_{\text{train}}$	31.72	21.70	**73.82**	68.41	54.03
SpUNet	ScanNet	τ_{train}	73.98	34.91	**73.98**	68.39	45.12
		$3\tau_{\text{train}}$	18.06	9.54	**57.54**	53.55	34.08
	nuScenes	τ_{train}	74.28	31.37	**74.28**	59.81	—
		$3\tau_{\text{train}}$	15.67	23.48	**68.05**	55.54	—

Table 22: **Relabelling at a larger size lowers the peak memory.** LitePT-S, frozen, peak MB on the timing benchmark, eight scenes. The benchmark keeps the inputs of every condition on the GPU, so only differences between rows are meaningful. Read at τ_{train} and relabelled at τ_{train}, the pipeline is the stock one.

		read at		
dataset	relabelled at	τ_{train}	$2\tau_{\text{train}}$	$3\tau_{\text{train}}$
ScanNet	τ_{train}	884.0	659.7	611.7
	$1.5\,\tau_{\text{train}}$	676.7	592.2	544.4
nuScenes	τ_{train}	338.3	329.0	323.7
	$1.5\,\tau_{\text{train}}$	277.2	275.9	269.8