Recurrent Point Transformer via Test-Time Training

Wentao Zhou[1,2] Xuweiyi Chen[1] Charles Saunders[3] Aruni RoyChowdhury[3] Zezhou Cheng[1]
[1]University of Virginia, Charlottesville [2]Georgia Institute of Technology [3]MathWorks

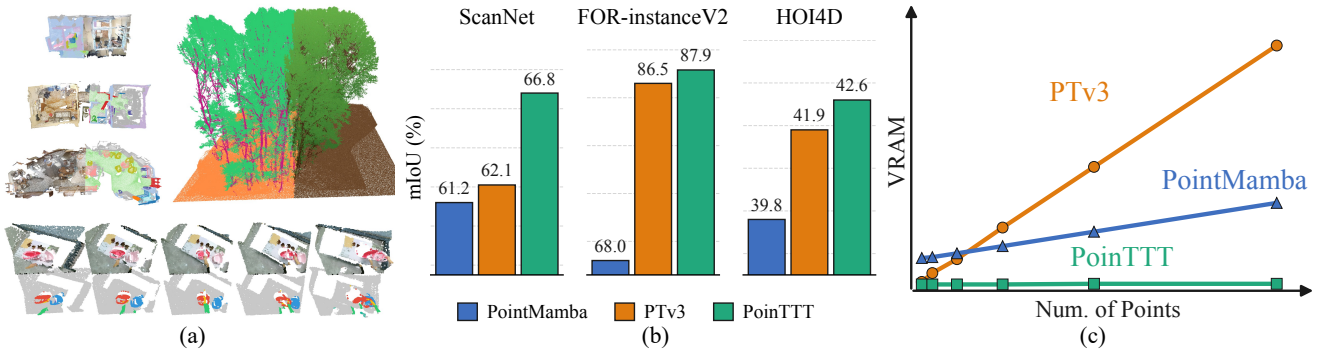


Figure 1. **PoinTTT scales to arbitrary-size point clouds in space and time.** (a) Real-world point clouds vary widely in scale, from room-scale 3D scans to large forest LiDAR surveys and long 4D sequences. (b) PoinTTT processes these inputs recurrently as fixed-size chunks, using test-time training to aggregate long-context information. It outperforms PTv3 [50] and Mamba-based baselines [26, 30] on static and dynamic segmentation benchmarks, including ScanNet [10], FOR-instanceV2 [53], and HOI4D [28]. The ScanNet bars use full-scene training (Tab. 1); the HOI4D blue bar denotes P4Mamba (39.8 mIoU). (c) PoinTTT maintains nearly constant memory as input size grows, unlike standard PTv3 [50]. Although trained with a fixed point budget per chunk, PoinTTT can be applied recurrently to scenes of arbitrary spatial or temporal extent.

Abstract

Real-world point clouds are not naturally bounded: 3D scans grow as sensors move through rooms, LiDAR surveys extend across forest plots, and 4D captures can span thousands of frames. Yet most point-cloud encoders are designed for fixed-size inputs, forcing inference-time cropping or tiling that loses long-range context and requires ad-hoc output stitching. We formulate arbitrary-extent point-cloud segmentation as recurrent sequence-of-chunks processing and introduce PoinTTT, a recurrent point-cloud encoder built on Point Transformer V3 (PTv3). PoinTTT augments PTv3 with residual Large Chunk Test-Time Training (LaCT) layers in both the encoder and decoder. Each LaCT layer maintains compact fast weights as recurrent state: tokens from each chunk update the fast weights, which are then applied to produce chunk features and passed to the next chunk. This design allows PoinTTT to train on fixed-size chunks while processing scenes of arbitrary spatial or temporal extent, with activation memory bounded by the chunk size. A pilot study on ScanNet and nuScenes shows that recurrent TTT improves segmentation over PTv3 baselines. On FOR-instanceV2, PoinTTT achieves state-of-the-art forest semantic segmentation of 87.9 *mIoU with nearly constant memory. On HOI4D, online full-sequence recurrent inference carries state across clips rather than resetting at benchmark boundaries, reaching* 42.6 *mIoU; a matched full-sequence Mamba control reaches approximately* 39.9*. These results suggest that test-time training is a practical primitive for scalable spatial and temporal point-cloud understanding.*

1. Introduction

Modern 3D perception increasingly operates on observations that are unbounded in space, time, or both. As shown in Fig. 1a, indoor scans may span multiple rooms [1, 6, 10, 57], 4D captures grow as frames arrive [28], and dense LiDAR surveys can cover broad outdoor plots [3, 4, 17, 40, 44, 53]. Point clouds naturally represent such expanding observations, yet most point-cloud backbones assume bounded inputs [5, 9, 33, 34, 43, 46, 49–51, 58, 59, 61]. This creates a mismatch for tasks like semantic segmentation: every point requires a label, but the needed evidence may lie outside the current crop, tile, or clip. Partitioning large scenes therefore limits context and forces the model to reset at artificial boundaries, rather than accumulating information across the

full scene or sequence.

We study a different interface for point cloud understanding: rather than treating cropping or tiling as an external inference procedure, we make chunking part of the model. An arbitrary-extent scene is represented as a sequence of fixed-size chunks processed by a recurrent encoder. The model sees one bounded chunk at a time, keeping training and activation memory controlled, while carrying a compact state across chunks so inference can scale with the spatial or temporal extent of the observation. For point clouds, this interface is practical: the same model can consume indoor scans, forest plots, and 4D frame streams without a memory footprint that grows with the number of points.

The key design question is where recurrence should be introduced and what form the recurrent state should take. A strong local backbone such as PTv3 [50] already provides effective geometric ordering and neighborhood reasoning within each chunk, but it has no explicit mechanism for carrying information across chunks. Recent state-based sequence models, including linear attention [20, 42, 55], Mamba [11, 16, 22], and Test-time Training [41, 60], show that compact recurrent states can replace explicit all-pair context and improve efficiency. Our goal is complementary: rather than replacing the point-cloud backbone, we add a cross-chunk memory module to it. Test-time training provides a natural mechanism for this memory by storing context in fast weights that are updated by each chunk and reused by subsequent chunks.

We instantiate this idea as *PoinTTT*, a recurrent point-cloud encoder built upon PTv3 [50]. PoinTTT augments both encoder and decoder of PTv3 [50] with residual LaCT [60] layers. Each LaCT layer maintains a small fast-weight network as recurrent state. Given a chunk, its tokens update the fast weights; the updated weights are then applied to produce chunk features and carried forward to the next chunk. The residual is gated from zero, so PoinTTT retains PTv3 at initialization and learns to add cross-chunk computation during training. The same recurrent encoder supports both offline and online settings: when the whole sequence is available, PoinTTT uses a bidirectional update-and-apply pass for full-scene spatial inference; for online 4D input, it propagates the same fast-weight state causally through the temporal stream, predicting each arriving chunk without future chunks (Sec. 4.3).

We first conduct a pilot study on indoor ScanNet [10] and outdoor nuScenes [4]. On ScanNet, we find that feeding chunks bottlenecks the network, regardless of whether the network is trained on entire ScanNet scenes or trained chunk-wise. This supports our hypothesis that partial observation is suboptimal for point-cloud understanding and highlights the necessity of using context from the entire scene. We further compare PointMamba [26] and PoinTTT, and find that PoinTTT outperforms PointMamba by a large margin.

Table 1. Pilot results on ScanNet with 10,240-point chunked inference and on nuScenes with single-pass evaluation without TTA. The training-input setting applies to both datasets. PointMamba [26] is a point-cloud state-space baseline; recurrent PoinTTT improves under the same bounded-memory protocol.

Backbone	Training input	ScanNet mIoU	nuScenes mIoU
No recurrent state			
PTv3 [50]	full scene	62.1	78.1
PTv3 [50]	10,240 chunks	63.8	77.8
State-space / recurrent state			
PointMamba [26]	full scene	61.2	73.9
PoinTTT	full scene	**66.8**	**78.7**

This indicates that test-time training provides stronger performance in this comparison (see Sec. 2). On nuScenes, temporal adaptation also improves PTv3 under single-pass evaluation without test-time augmentation. Furthermore, we demonstrate PoinTTT as a unified point-cloud understanding backbone for both temporal and spatial settings. We stress-test PoinTTT on dense tree segmentation using FOR-instanceV2 against methods tailored for this task, such as ForestFormer3D [53], and PoinTTT achieves state-of-the-art performance (see Sec. 5.2). We then compare online 4D segmentation methods on HOI4D [28] (Sec. 5.3), where PoinTTT carries state across complete videos and outperforms a matched recurrent Mamba baseline.

Our contributions are threefold. (1) We formulate arbitrary-extent point-cloud segmentation as recurrent chunk processing across spatial and temporal regimes (Sec. 2, Sec. 4, Fig. 1) (2) We introduce *PoinTTT*, augmenting PTv3 [50] with residual LaCT [60] layers that carry compact fast-weight states for offline full-scene and online temporal inference (Sec. 4). (3) We demonstrate strong performance on ScanNet [10], FOR-instanceV2 [53], HOI4D [28], and nuScenes [4], with activation memory bounded by chunk size (Sec. 2, Sec. 5.2, Sec. 5.3). Across all settings, PoinTTT processes arbitrary-extent inputs with strong performance and activation memory bounded by chunk size.

2. Pilot Study

We examine segmentation with bounded inputs on indoor ScanNet [10] and outdoor nuScenes [4]. On ScanNet, all methods in Tab. 1 use 10,240-point inference chunks, with predictions reassembled before computing scene-level mIoU. On nuScenes, evaluation is single-pass without test-time augmentation (TTA). The table's *Training input* setting applies to both datasets. PointMamba implementation details are provided in Sec. 5.1.

On ScanNet, the full-scene-trained PTv3 [50] checkpoint drops from 76.6 mIoU under full-scene evaluation to 62.1 with chunked inputs. The full-scene score is a reference

obtained with a larger per-forward point budget. Training directly on chunks recovers only part of this loss. PoinTTT carries a compact recurrent state across chunks and outperforms both PTv3 training variants and PointMamba [26] under the same chunked evaluation. On nuScenes, temporal adaptation likewise improves over PTv3 under the single-pass protocol (Tab. 1).

Fig. 1c shows that the memory usage of PTv3 and PointMamba grows with input size, while PoinTTT keeps activation memory bounded with fixed-size chunks. These pilot results motivate combining local point-cloud encoding with persistent scene memory.

3. Related Work

3.1. Point-Cloud Segmentation Backbones

Point-cloud segmentation has progressed from permutation-invariant set encoders to architectures with increasingly rich local geometry and context. PointNet [33] and PointNet++ [34] introduced direct neural processing of point sets, while DGCNN [46] and KPConv [43] improved local interaction modeling. MinkowskiNet [9] made large voxelized scenes practical, and Point Transformer [61], PTv2 [49], Swin3D [56] and PTv3 [50] introduced attention-based architectures for 3D segmentation. Recent systems further scale this line through point prompt training in PPT [51], multi-dataset pretraining in Point-MoE [7], lighter transformer design in LitePT [59], and vanilla transformer variants in VOLT [58]. PointMamba [26] explores state-space sequence modeling for point-cloud analysis. These models provide strong backbones for bounded point sets, voxel sets, windows, or scenes. PoinTTT complements this progress by retaining the backbone as a local point-cloud encoder while introducing a compact recurrent state that carries information across chunk boundaries. This allows computation to extend beyond individual chunks and supports spatially or temporally extended inputs.

3.2. Large-Scale Point-Cloud Understanding

Large-scale point-cloud benchmarks make the bounded-input assumption visible. Indoor datasets such as S3DIS [1], Matterport3D [6], ScanNet [10], and ScanNet++ [57], together with outdoor LiDAR datasets such as Semantic3D [17], SemanticKITTI [3], nuScenes [4], Waymo [40], DALES [44], and FOR-instanceV2 [53], span diverse environments. Yet mainstream backbones, including point-based methods [33, 34], voxel and sparse convolutional methods [8, 15, 23, 39], and transformer-based methods [49, 50, 56, 61], are in practice trained and evaluated on sampled blocks, sliding windows or tiles.

Dynamic point cloud datasets such as MSR-Action3D [24], NTU RGB+D [37], Synthia 4D [9], and HOI4D [28], together with the temporal extensions of outdoor LiDAR benchmarks above, add a temporal version of the same scaling issue. A line of work models point-cloud sequences via spatiotemporal neighborhoods, 4D convolutions, transformer attention, or primitive-level hierarchies [9, 12–14, 25, 27, 28, 30, 47, 48], but the common protocols still operate on short clips or reset state at clip boundaries. PoinTTT uses a shared recurrent chunk interface for large spatial scenes and full temporal sequences, carrying state across chunk boundaries.

3.3. State Recurrence and Test-Time Training

State recurrence has re-emerged as a scalable alternative to explicit all-pair attention. Linear attention [20, 35, 36, 42, 54, 55] can be viewed as a recurrent accumulation of key-value statistics, state-space models such as Mamba [11, 16, 22] use compact causal state, and TTT layers [41] make the state an inner-loop-updated network. LaCT [60] improves the efficiency and capacity of this view by updating large chunks with expressive fast-weight MLPs. In 3D, closely related reconstruction systems use TTT or LaCT-like states to compress long image sequences into compact scene representations, including tttLRM [45], ZipMap [19], and Fast Spatial Memory [31]. A separate line of work brings test-time learning to point clouds for distribution shift rather than long-range memory: MATE [32] uses masked autoencoding as an online test-time objective for point cloud classification, SMART-PC [2] adapts robust skeletal representations under point-cloud corruptions, and PointMAC [18] uses meta-learned auxiliary objectives for test-time point-cloud completion. These methods adapt a model to a sample or target distribution, whereas PoinTTT uses fast weights that store context across chunks of the same large scene or temporal sequence, and its update-and-apply recipe applies for cross-chunk recurrence while respecting the unordered nature of points within each chunk.

4. Method

PoinTTT processes arbitrary-extent point clouds as sequences of bounded chunks. Its core mechanism is a recurrent fast-weight state, implemented with residual LaCT [60] layers, that is updated by each chunk and reused by subsequent chunks. The underlying point-cloud backbone provides local feature extraction within each chunk, while LaCT carries information across chunks. This separates local geometry processing from global context accumulation, allowing one model to handle both large static scenes and streaming 4D inputs while keeping activation memory bounded by the size of each chunk.

4.1. Chunked Point-Cloud Segmentation

PoinTTT is designed around a chunked inference interface: the input may be arbitrarily large, but the model only receives a bounded point-cloud chunk at each step. Let a semantic

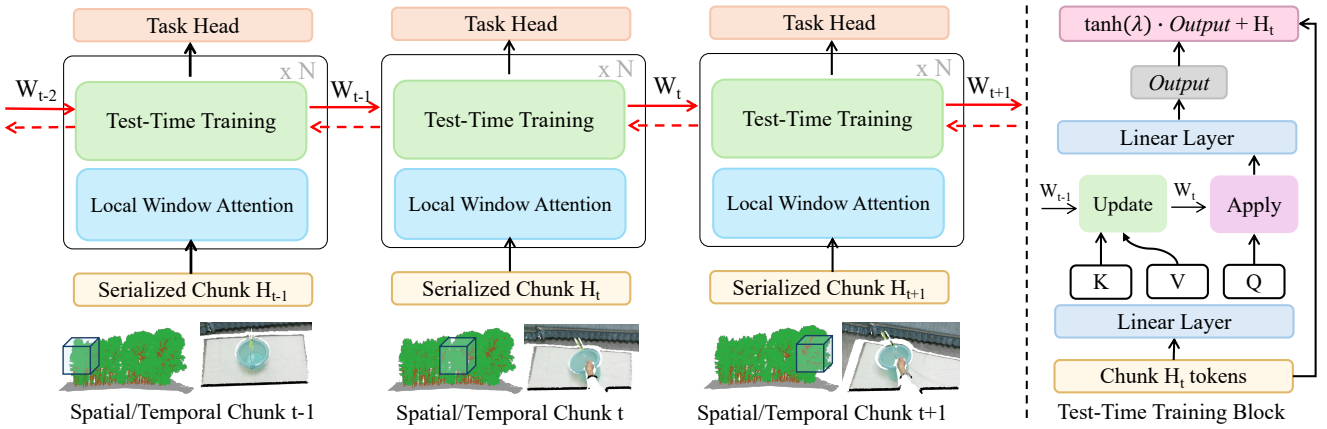


Figure 2. **PoinTTT uses recurrent fast weights for unified spatial and temporal point-cloud inference.** Each step processes one spatial or temporal chunk with a shared PTv3-based [50] backbone augmented by a residual LaCT [60] block. LaCT maps chunk features H_t to Q, K, V, updates fast weights from W_{t-1} to W_t using K, V, and applies the updated weights to Q. The readout is normalized and linearly projected before being added to H_t through the gate $\tanh(\lambda)$. Solid arrows denote causal propagation for temporal streams, while dashed reverse arrows indicate bidirectional propagation over spatial chunks. The resulting compact recurrent state enables large-scene and streaming 4D inference without storing full-scene activations.

segmentation scene be $\mathcal{S} = \{(p_i, a_i, y_i)\}_{i=1}^{n}$, where n is the number of points, $p_i \in \mathbb{R}^3$ is a point coordinate, a_i denotes optional point attributes, and y_i is the semantic label. At inference time, the scene is represented as an ordered chunk sequence $\Pi(\mathcal{S}) = (\mathcal{P}_1, \ldots, \mathcal{P}_T)$, where each chunk is $\mathcal{P}_t = \{(p_{t,j}, a_{t,j})\}_{j=1}^{M_t}$ with $M_t \le M$. Here M is the fixed chunk budget and T grows with the spatial or temporal extent of the observation. The goal is to predict one semantic label for every point while keeping activation memory bounded by M, rather than by the full scene size.

Chunk ordering depends on the acquisition setting. For static point clouds, we voxelize the scene on a fixed grid, encode each voxel's integer coordinate with a 3D Hilbert curve, sort the resulting indices, and partition the 1D ordering into consecutive fixed-size blocks. Hilbert locality keeps each chunk spatially compact. For 4D point clouds, each short temporal segment of frames is voxelized and forms one chunk. In both cases, the fast-weight state evolves along the chunk axis. This shared recurrent formulation supports offline full-scene segmentation and online segmentation of successive point-cloud frames.

4.2. LaCT as Recurrent Scene State

A LaCT [60] layer stores scene context in the *fast weights* of a small neural network. Each chunk updates these weights through a key–value association task, then queries the updated network to retrieve context. We use a SwiGLU MLP [38] with parameters $W = \{W^{(0)}, W^{(1)}, W^{(2)}\}$ and mapping $f_W(x) = W^{(1)}(\mathrm{SiLU}(W^{(0)}x) \odot W^{(2)}x)$. Superscripts distinguish the three matrices, while the subscript in W_t identifies the current recurrent step.

Given backbone features $H_t \in \mathbb{R}^{M_t \times d}$, we first apply layer normalization, multiply by the projection matrix W^{qkv}, and apply SiLU. Splitting the result yields queries $\widetilde{\mathbf{Q}}_t$, keys $\widetilde{\mathbf{K}}_t$, and values $\mathbf{V}_t$. We then ℓ_2-normalize each query and key vector using $z/(\|z\|_2 + \epsilon)$ to obtain $\mathbf{Q}_t$ and $\mathbf{K}_t$, respectively. Keys and values supply the update, while queries read from the updated state.

The update uses a chunk-local associative TTT objective,

$$\mathcal{L}_{\mathrm{TTT}}(W; H_t) = -\sum_{j=1}^{M_t} \langle f_W(\mathbf{K}_{t,j}), \mathbf{V}_{t,j} \rangle ,$$

which encourages the fast-weight network to associate each key with its value. Starting from the incoming state W_{t-1}, we take a gradient step for each matrix $W^{(r)}$, $r \in \{0, 1, 2\}$. The step uses Newton–Schulz orthogonalization, denoted by NS, followed by Norm to restore the row-wise weight norm within each head. These operations define $W_t = \mathrm{Update}(W_{t-1}, H_t)$:

$$\begin{aligned} \Delta W_t^{(r)} &= -\mathrm{NS}(\nabla_{W^{(r)}} \mathcal{L}_{\mathrm{TTT}}(W_{t-1}; H_t)) , \\ W_t^{(r)} &= \mathrm{Norm}\left(W_{t-1}^{(r)} + \Delta W_t^{(r)}\right) . \end{aligned} \tag{1}$$

We query the updated state and project its output:

$$O_t = \mathrm{Apply}(W_t, H_t) = P_{\mathrm{out}}(f_{W_t}(\mathbf{Q}_t)) . \tag{2}$$

Here P_{out} applies per-head layer normalization, concatenates the heads, and performs the learned output projection shown as the Linear Layer after Apply in Fig. 2. Thus, W_t serves two roles: it conditions the current chunk output through the query features, and it is carried forward as a compact recurrent scene memory that accumulates context as new chunks arrive.

4.3. PoinTTT Architecture

PoinTTT augments a point-cloud backbone with residual LaCT [60] layers. The backbone extracts local geometric features inside each chunk, while LaCT carries informa-

tion across chunks through the recurrent fast-weight state in Eq. (1). We insert LaCT layers in both the encoder and decoder, allowing both geometric and semantic features to accumulate cross-chunk context.

Each LaCT layer is inserted as a gated residual branch,

$$H_t^{\text{out}} = H_t + \tanh(\lambda)\, O_t.$$

Here O_t is the projected LaCT output from Eq. (2), H_t denotes the backbone features for chunk $\mathcal{P}_t$, and λ is initialized to zero. Thus, PoinTTT starts from the original backbone behavior and learns to add cross-chunk computation during training. Spatial and temporal inference use the same residual LaCT block, with fast weights propagated according to the chosen inference mode.

The recurrent interface has two modes, depending on whether all chunks are available before prediction.

Offline spatial inference. For static scenes, all chunks are available before prediction. PoinTTT first scans the chunks in forward order under `no_grad`, using Eq. (1):

$$W_t = \text{Update}(W_{t-1}, H_t), \qquad W_0 = W_{\text{init}}.$$

This pass initializes the fast weights with full-scene context without storing full-scene activations.

PoinTTT then runs an *apply* pass over the sequence in reverse order, starting from the final state W_T, and produces the chunk features $\widetilde{H}_t$ and logits $\hat{Y}_t$,

$$\widetilde{H}_t = H_t + \tanh(\lambda)\, \text{Apply}(W_T, H_t),$$
$$\hat{Y}_t = \text{Decode}(\widetilde{H}_t).$$

The reverse apply pass is differentiable with respect to both the LaCT layers and the surrounding backbone parameters during training. Gradients are accumulated across the apply sequence and backpropagated after the full pass completes. At inference time, this update-and-apply procedure predicts each chunk using state informed by the full spatial sequence, while keeping activation memory bounded by chunk size.

Online temporal inference. For dynamic scenes, future observations are unavailable when the current chunk must be predicted. PoinTTT therefore uses causal recurrence:

$$W_t = \text{Update}(W_{t-1}, H_t),$$
$$\widetilde{H}_t = H_t + \tanh(\lambda)\, \text{Apply}(W_t, H_t),$$
$$\hat{Y}_t = \text{Decode}(\widetilde{H}_t).$$

The prediction at step t depends only on chunks observed up to time t, making the model online in the standard streaming sense. The fast weights act as recurrent memory over the 4D sequence. During training, gradients are accumulated across the sequence and backpropagated after the full sequence has been processed.

Table 2. Semantic segmentation on the FOR-instanceV2 test split. Best results are in **bold**, second-best are underlined.

Method	Ground	Wood	Leaf	mIoU
PointMamba [26]	95.9	16.1	91.9	68.0
ForAINetV2_R8 [52]	98.3	68.1	94.3	86.9
ForAINetV2_R16 [52]	98.8	69.2	94.5	87.5
OneFormer3D [21]	97.8	63.1	93.4	84.8
ForestFormer3D [53]	96.9	67.7	94.1	86.2
PTv3 [50]	99.0	65.9	94.7	86.5
PoinTTT (ours)	**99.3**	**69.3**	**95.1**	**87.9**

5. Experiments

We evaluate whether PoinTTT scales beyond the ScanNet and nuScenes pilot study (Sec. 2) to arbitrary-extent spatial and temporal inputs. We study two settings: *forest segmentation*, where the number of spatial chunks grows with scan size, and *online 4D segmentation*, where chunks arrive over time. Unlike standard scan- or clip-based inference, PoinTTT initializes its fast-weight state once per test scan or video and processes the entire input recurrently before computing the final metric.

5.1. Setup

Datasets and metrics. FOR-instanceV2 [53] contains large forest LiDAR plots with ground, wood, and leaf labels; individual scenes range from millions of points to roughly $4{\times}10^8$ points. We report class IoU and mIoU. HOI4D [28] contains dynamic hand–object point-cloud sequences; we report semantic segmentation mIoU over the temporal stream.

Training. For FOR-instanceV2 and HOI4D, we first pretrain the PTv3 [50] backbone on fixed-size point-cloud chunks using the standard point-wise cross-entropy segmentation loss. This stage learns local geometric and semantic features for the chunk encoder. We then insert residual LaCT [60] layers and finetune PoinTTT end-to-end, updating both the backbone and recurrent layers with the update-and-apply recurrence active. Forest pretraining and finetuning use 100 and 200 epochs, respectively; HOI4D uses 50 epochs for each stage. Both stages use AdamW with OneCycleLR schedules. Forest training uses a single A100 GPU in both stages. HOI4D pretraining runs on one A100 GPU, while finetuning uses eight A100 GPUs. Inference for both benchmarks is performed on a single A100 GPU. See the appendix for batch sizes, parameter-group learning rates, and scheduler settings for pretraining and finetuning.

Mamba baselines. For spatial modeling, we adapt PointMamba [26] to scene-level semantic segmentation using the same PTv3-style scene processing protocol as our baselines, while preserving its original tokenization and serialization. For the PTv3+Mamba control in Sec. 5.3, we use the temporal Mamba implementation from PointNet4D [30].

Evaluation pipeline. We follow the chunked point-cloud for-

Table 3. Semantic segmentation on HOI4D. † marks published offline supervised results from PointNet4D [30], Table 9; P4Mamba is its online result from Table 7. PTv3 uses fixed windows; PTv3+Mamba and PoinTTT share the PTv3 backbone and full-video streaming protocol.

Method	Input window	Context horizon	mIoU
P4Mamba [30]	3	3	39.8
P4Transformer† [13]	3	3	40.1
PPTr† [48]	3	3	41.0
PointNet4D† [30]	3	3	41.6
PointNet4D++† [30]	3	3	41.9
PTv3 (our baseline)	3	3	41.9
PTv3+Mamba	3	full video (300)	39.9
PoinTTT (ours)	3	full video (300)	**42.6**

mulation in Sec. 4.1. For forest segmentation, non-recurrent baselines process local crops independently and merge their point-wise predictions back into the full plot. This gives them access to the same input points and evaluation labels, but not to persistent information across chunks. PoinTTT instead runs offline spatial recurrence over all chunks in the plot: an update pass traverses the chunk sequence to build a fast-weight state, and an apply pass decodes per-chunk predictions using that state; each chunk has at most 102,400 points. Predictions are then reassembled in the original plot coordinates before computing IoU over the full point set.

For HOI4D, published semantic segmentation protocols use local temporal windows. PoinTTT runs online recurrence over the full sequence: the state is initialized once per video, updated by each three-frame chunk, and carried forward without observing future chunks. All methods use the same semantic labels and evaluation metrics. PoinTTT carries recurrent context across the full video instead of treating short clips as independent inputs.

5.2. Forest Spatial Segmentation

Tab. 2 compares PoinTTT with prior forest segmentation methods on FOR-instanceV2 [53]. PoinTTT is trained only on fixed-size chunks, but at test time it is applied to each forest plot as a full-scene recurrent sequence over all chunks. Thus, the model keeps the same bounded-memory interface used during training while allowing information to be carried across the entire plot. PoinTTT reaches 87.9 mIoU, improving over PTv3 [50] by 1.4 points and over ForestFormer3D [53] by 1.7 points.

The largest gain appears on the *wood* class, where stems and branches often extend across local partitions. This suggests that full-plot context is useful for point-cloud understanding, and that fast-weight recurrence provides an effective memory mechanism for aggregating such cross-chunk information. Qualitative results in Fig. 3(b) further show that PoinTTT better preserves connected tree structures during offline spatial inference.

5.3. Online 4D Temporal Segmentation

Streaming inference beyond the local window. PoinTTT processes each HOI4D video as a stream of three-frame chunks, each serving as the *input window* jointly encoded at one step. The fast-weight state is initialized once per video and carried across chunks; predictions depend only on observations received so far. The *context horizon* grows with the observed video prefix, reaching approximately 300 frames at the end of a video. The recurrence has no preset video-length limit: processing can continue with the same chunk budget and fixed-size state. The reported experiment tests this capability over complete HOI4D videos.

Published protocols and matched memory mechanisms. Prior HOI4D baselines use short-clip protocols, typically three frames, with ten-frame variants also reported [30]. Our PTv3 baseline follows this local-window setup and is trained and evaluated on three-frame clips. Tab. 3 reports the published three-frame results, distinguishing the offline supervised results in PointNet4D [30], Table 9, from its online P4Mamba result in Table 7.

To compare recurrent memory mechanisms on the same long sequences, we construct a PTv3+Mamba baseline by replacing LaCT's fast-weight memory with Mamba selective state-space layers. We retain the same PTv3 backbone, three-frame input windows, training setup, parameter budget, and full-video streaming protocol. Both models carry their respective recurrent states across the complete video. Tab. 3 reports 39.9 mIoU for PTv3+Mamba and 42.6 for PoinTTT.

Complementary HOI4D approaches include LeaF [29] and 4DMAP, the frame-wise masked autoregressive pretraining scheme introduced with PointNet4D [30]. Our comparison here focuses on 4D point-cloud encoder backbones under supervised semantic segmentation training.

5.4. Qualitative Comparison

On ScanNet (Fig. 3), PTv3 exhibits block-shaped and diagonal artifacts introduced by Hilbert-order chunk partitioning; PoinTTT reduces these artifacts and produces smoother semantic boundaries. On FOR-instanceV2, PTv3 struggles with tilted trunks and irregular branches that deviate from dominant vertical structures, while PoinTTT better preserves their geometric continuity. On HOI4D, PTv3 is more prone to frame-local mistakes, such as confusing moving objects with nearby background regions. Temporal context across frames helps PoinTTT make more stable predictions and distinguish moving objects from the background.

5.5. Ablations

Memory and capacity. Tab. 4a retains the LaCT modules while disabling cross-chunk aggregation during both training and evaluation. The added transformations remain available, but their information is confined to individual chunks. This model learns under the same restricted context that it en-

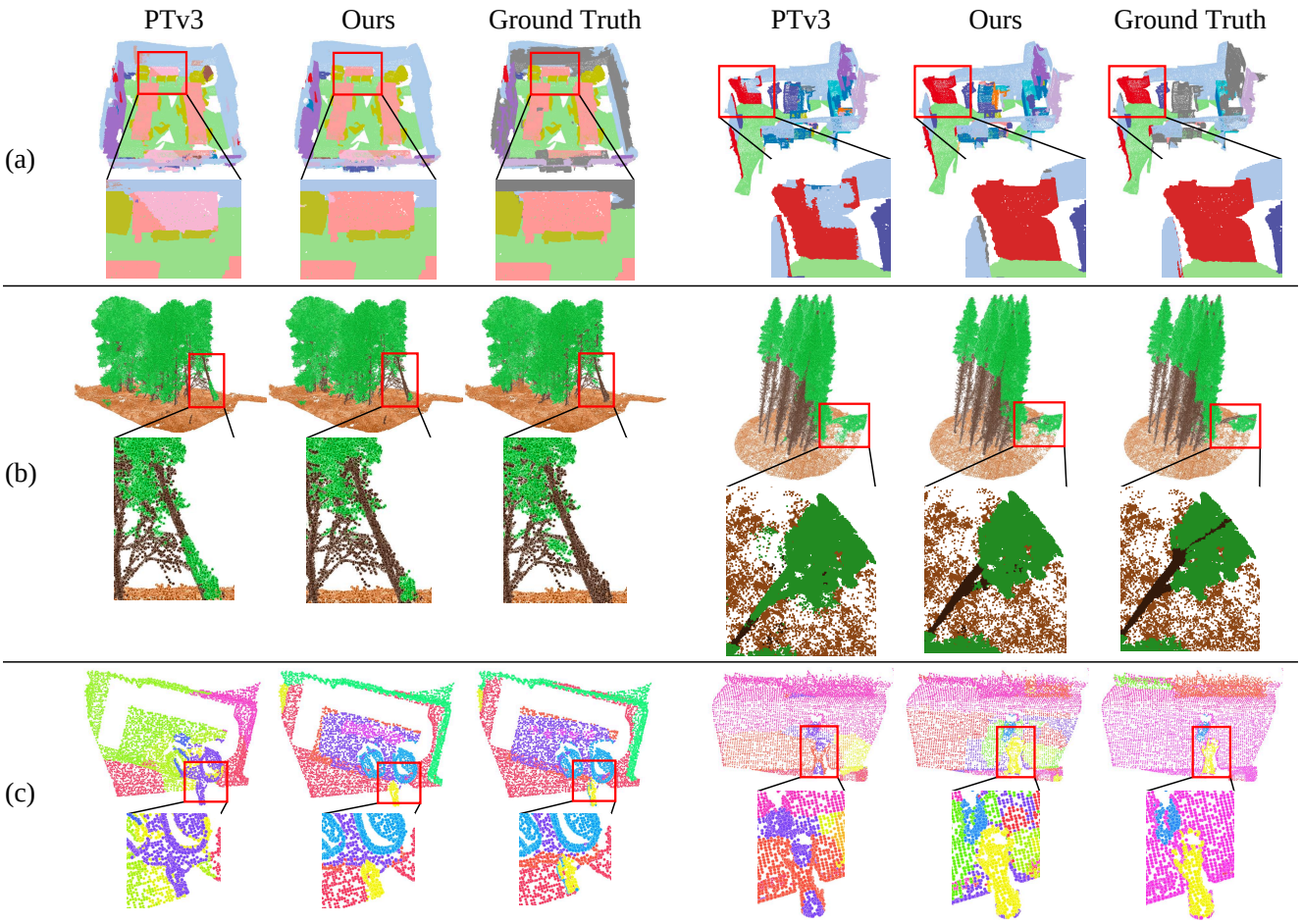


Figure 3. **Qualitative comparison between PoinTTT and PTv3.** Two examples per dataset show semantic predictions and ground truth; red boxes mark enlarged regions. PoinTTT (Ours) reduces chunk-boundary artifacts on (a) ScanNet [10], better preserves tilted trunks and irregular branches on (b) FOR-instanceV2 [53], and more clearly separates moving objects from the background on (c) HOI4D [28].

counters at inference. Its lower accuracy supports using the fast weights to communicate across chunks, beyond their role in transforming local features. Tab. 4b examines the training schedule: extending PTv3 training to the same total number of epochs still leaves an accuracy gap. Together, these comparisons motivate making scene context available to local predictions through a persistent state.

Architecture design. Joint encoder–decoder insertion outperforms either placement alone (Tab. 4c), suggesting that scene context benefits both feature extraction and point-wise reconstruction. Each inserted layer maintains its own fast weights and forms associations from the features at that stage, allowing different resolutions to maintain different representations of the scene. Extending recurrence to more encoder and decoder stages improves accuracy (Tab. 4f), supporting our choice to distribute scene memory across different levels of the feature hierarchy.

Chunk size. Chunk size controls both the geometry jointly available to the backbone and the granularity of recurrent updates. For a fixed scene, larger chunks produce fewer state updates, each incorporating more points; the fast-weight state capacity remains unchanged. The forest sweep in Tab. 4d is non-monotonic: the intermediate budget performs best, and increasing it further does not improve accuracy. One possible explanation is a balance between local geometric support and the frequency of context updates across the scene. We use 102,400 points per chunk as the best-performing tested setting, processing fewer points at once than the largest tested budget.

Chunk construction. Hilbert and Z-order both outperform random chunk membership (Tab. 4e). Both curves keep spatially related points together, providing coherent neighborhoods for the local point encoder. Random membership disperses these observations across chunks, changing the geometry available within each forward pass. Recurrent context can connect the resulting features across chunks, but it does not give the encoder explicit access to the other chunks' point neighborhoods. The gap between Hilbert and Z-order is smaller than either curve's gap to random membership, consistent with the value of retaining spatial coherence

Table 4. Ablations of recurrent aggregation and model design. Metrics are percentages except (i). Panels (a)–(g) use FOR-instanceV2; (h)–(i) use ScanNet. Panel (a) disables aggregation in training and evaluation; (b) matches total training epochs. In (d)–(g), † marks the default. Random membership removes within-chunk spatial coherence. Panel (i) compares full-scene PTv3 with chunked PoinTTT.

(a) Aggregation

PoinTTT variant	mIoU
No aggregation	86.8
Full model	87.9

(b) Training budget

Method	mIoU	mAcc	allAcc
PTv3	86.8	90.7	95.7
PoinTTT	**87.9**	**91.9**	**96.0**

(c) LaCT placement

Placement	mIoU	mAcc	allAcc
encoder	87.3	**91.9**	95.8
decoder	87.1	90.6	95.8
all	**87.9**	**91.9**	**96.0**

(d) Chunk size

Points/chunk	mIoU
51,200	87.44
102,400†	87.89
204,800	87.68

(e) Chunk construction

Construction	mIoU
Hilbert†	87.89
Z-order	87.56
Random membership	87.08

(f) Recurrent depth

Encoder	Decoder	Layers	mIoU
4	0	2	87.45
3,4	0,1	4	87.62
1–4†	0–3	8	87.89

(g) Update steps

Steps/chunk	mIoU
1†	87.89
2	87.64
4	87.56

(h) Point budgets (ScanNet)

Points/chunk	PTv3	PoinTTT	Gain
4,096	58.3	59.2	+0.9
20,480	65.9	70.2	+4.3
40,960	67.1	73.1	+6.0
102,400	68.4	74.4	+6.0

(i) Memory footprint (ScanNet)

Memory	PTv3	PoinTTT
Peak VRAM	19 GB	**0.3 GB**
Fast-weight state	–	1 MB

alongside recurrent context. We use Hilbert serialization for the remaining experiments.

Update steps. Each inner update writes key–value associations from the current chunk into the fast weights. Repeating the update revisits the same observations; it does not introduce additional points or extend the observed scene. More updates per chunk do not improve segmentation accuracy (Tab. 4g). The associative update objective and the segmentation objective serve different purposes, so more inner optimization need not yield better predictions. This also differs from adding recurrent layers in Tab. 4f, where each layer writes and reads features from a different stage of the backbone. We use one update per chunk, which gives the best result in this sweep without the additional inner iterations.

Input budgets and memory scaling. Tab. 4h extends the pilot study to test whether the advantage of recurrence persists across different local input budgets. On ScanNet, both models benefit from larger evaluation chunks, while PoinTTT outperforms independently chunked PTv3 at every matched point budget. Both methods eventually process all points in a scene. However, merging independently predicted chunks does not let an individual prediction use information from outside its own chunk. PoinTTT can condition that prediction on information accumulated in the recurrent state. Its advantage at larger budgets suggests that local geometric coverage and cross-chunk context offer complementary benefits: a larger chunk strengthens the local input, while recurrence supplies information beyond it.

Tab. 4i compares full-scene PTv3 with chunked PoinTTT on the same scene, using different per-forward point budgets. Full-scene processing keeps activations for the entire input, whereas PoinTTT processes a bounded chunk and retains scene information in its fast weights. These weights have a fixed size per layer, independent of the total point count, and are not a growing cache of point-wise features. Compressing context into this state allows information to persist across chunks without keeping full-scene activations, accounting for the lower peak VRAM of chunked execution. The appendix provides the per-forward point budgets and the memory required to store the fast weights.

6. Conclusion

We argue that arbitrary-extent point-cloud segmentation should be treated as a recurrent inference problem, rather than as independent crop or clip prediction. Chunking provides the necessary bounded-memory interface, but our ScanNet [10] pilot shows that chunking alone creates a context bottleneck: predictions made from local chunks cannot fully recover scene-level reasoning.

Thus, we introduce PoinTTT to address this bottleneck by carrying compact LaCT [60] fast weights across the chunk sequence. These fast weights act as recurrent scene memory, allowing the model to accumulate context while keeping activation memory tied to the chunk size. The same mechanism supports both offline spatial inference over large static scenes and online temporal inference over 4D streams. Across ScanNet, FOR-instanceV2 [53], HOI4D [28], and nuScenes [4], PoinTTT improves segmentation under fixed-memory in-

ference and demonstrates that fast-weight TTT recurrence is a practical way to connect local point-cloud chunks into coherent full-scene or full-sequence predictions.

Acknowledgments

We thank Adobe for its research gift. We acknowledge computational support from the Research Computing and Data Analytics Center at the University of Virginia, the AMD AI and HPC Cluster Program, ACCESS, and the NAIRR Pilot. These resources include Purdue University's Anvil supercomputer (NSF award OAC-2005632) and the Delta and DeltaAI systems (NSF award OAC-2005572).

References

[1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[2] Ali Bahri, Moslem Yazdanpanah, Sahar Dastani, Mehrdad Noori, Gustavo Adolfo Vargas Hakim, David Osowiechi, Farzad Beizaee, Ismail Ben Ayed, and Christian Desrosiers. Smart-pc: Skeletal model adaptation for robust test-time training in point clouds. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 2442–2457, 2025.

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019.

[4] 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*, pages 11621–11631, 2020.

[5] Jinming Cao, Hanchao Leng, Dani Lischinski, Daniel Cohen-Or, Changhe Tu, and Yangyan Li. Shapeconv: Shape-aware convolutional layer for indoor rgb-d semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7088–7097, 2021.

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *Proceedings of the International Conference on 3D Vision*, pages 667–676, 2017.

[7] Xuweiyi Chen, Wentao Zhou, Aruni RoyChowdhury, and Zezhou Cheng. Point-moe: Large-scale multi-dataset training with mixture-of-experts for 3d semantic segmentation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[8] Yukang Chen, Yanwei Li, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Focal sparse convolutional networks for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5428–5437, 2022.

[9] 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*, pages 3075–3084, 2019.

[10] 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*, pages 2432–2443, 2017.

[11] Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, pages 10041–10071, 2024.

[12] Hehe Fan and Yi Yang. Pointrnn: Point recurrent neural network for moving point cloud processing. *arXiv preprint arXiv:1910.08287*, 2019.

[13] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14204–14213, 2021.

[14] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. Pstnet: Point spatio-temporal convolution on point cloud sequences. In *International Conference on Learning Representations*, 2021.

[15] 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*, pages 9224–9232, 2018.

[16] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[17] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D. Wegner, Konrad Schindler, and Marc Pollefeys. Semantic3d.net: A new large-scale point cloud classification benchmark. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1/W1:91–98, 2017.

[18] Linlian Jiang, Rui Ma, Li Gu, Ziqiang Wang, Xinxin Zuo, and Yang Wang. Pointmac: Meta-learned adaptation for robust test-time point cloud completion. In *Advances in Neural Information Processing Systems*, 2025.

[19] Haian Jin, Rundi Wu, Tianyuan Zhang, Ruiqi Gao, Jonathan T Barron, Noah Snavely, and Aleksander Holynski. Zipmap: Linear-time stateful 3d reconstruction via test-time training. *arXiv preprint arXiv:2603.04385*, 2026.

[20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5156–5165, 2020.

[21] Maxim Kolodiazhnyi, Anna Vorontsova, Anton Konushin, and Danila Rukhovich. Oneformer3d: One transformer for unified point cloud segmentation. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20943–20953, 2024.

[22] Aakash Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, J Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *The Fourteenth International Conference on Learning Representations*, 2026.

[23] Lihan Li, Haofeng Zhong, Rui Bu, Mingchao Sun, Wenzheng Chen, Baoquan Chen, and Yangyan Li. Pointcnn++: Performant convolution on native points. *arXiv preprint arXiv:2511.23227*, 2025.

[24] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 9–14, 2010.

[25] Wenhao Li, Xueying Jiang, Gongjie Zhang, Xiaoqin Zhang, Ling Shao, and Shijian Lu. Sts-mixer: Spatio-temporal-spectral mixer for 4d point cloud video understanding. *arXiv preprint arXiv:2604.11637*, 2026.

[26] Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. Pointmamba: A simple state space model for point cloud analysis. In *Advances in Neural Information Processing Systems*, 2024.

[27] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9246–9255, 2019.

[28] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21013–21022, 2022.

[29] Yunze Liu, Junyu Chen, Zekai Zhang, Jingwei Huang, and Li Yi. Leaf: Learning frames for 4d point cloud sequence understanding. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 604–613. IEEE, 2023.

[30] Yunze Liu, Zifan Wang, Peiran Wu, and Jiayang Ao. Pointnet4d: A lightweight 4d point cloud video backbone for online and offline perception in robotic applications. *arXiv preprint arXiv:2512.01383*, 2025.

[31] Ziqiao Ma, Xueyang Yu, Haoyu Zhen, Yuncong Yang, Joyce Chai, and Chuang Gan. Fast spatial memory with elastic test-time training. *arXiv preprint arXiv:2604.07350*, 2026.

[32] M. Jehanzeb Mirza, Inkyu Shin, Wei Lin, Andreas Schriebl, Kunyang Sun, Jaesung Choe, Mateusz Kozinski, Horst Possegger, In So Kweon, Kuk-Jin Yoon, and Horst Bischof. Mate: Masked autoencoders are online 3d test-time learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16709–16718, 2023.

[33] Charles Ruizhongtai 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*, pages 652–660, 2017.

[34] Charles Ruizhongtai 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*, 2017.

[35] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.

[36] Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. Learning associative inference using fast weight memory. In *International Conference on Learning Representations*, 2021.

[37] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016.

[38] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[39] Guangsheng Shi, Ruifeng Li, and Chao Ma. Pillarnet: Real-time and high-performance pillar-based 3d object detection. In *European conference on computer vision*, pages 35–52. Springer, 2022.

[40] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[41] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 57503–57522, 2025.

[42] Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

[43] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[44] Nina Varney, Vijayan K. Asari, and Quinn Graehling. Dales: A large-scale aerial lidar data set for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[45] Chen Wang, Hao Tan, Yifan Wang, Zhiqin Chen, Yuheng Liu, Kalyan Sunkavalli, Sai Bi, Lingjie Liu, and Yiwei Hu. tttlrm: Test-time training for long context and autoregressive 3d reconstruction. *arXiv preprint arXiv:2602.20160*, 2026.

[46] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):146:1–146:12, 2019.

[47] Yimin Wei, Hao Liu, Tingting Xie, Qiuhong Ke, and Yulan Guo. Spatial-temporal transformer for 3d point cloud sequences. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1171–1180, 2022.

[48] Hao Wen, Yunze Liu, Jingwei Huang, Bo Duan, and Li Yi. Point primitive transformer for long-term 4d point cloud video understanding. In *Computer Vision – ECCV 2022*, pages 19–35, 2022.

[49] 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*, 2022.

[50] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[51] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[52] Binbin Xiang, Maciej Wielgosz, Theodora Kontogianni, Torben Peters, Stefano Puliti, Rasmus Astrup, and Konrad Schindler. Automated forest inventory: Analysis of high-density airborne lidar point clouds with 3d deep learning. *Remote Sensing of Environment*, 305:114078, 2024.

[53] Binbin Xiang, Maciej Wielgosz, Stefano Puliti, Kamil Král, Martin Krůček, Azim Missarov, and Rasmus Astrup. Forestformer3d: A unified framework for end-to-end segmentation of forest lidar 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24717–24727, 2025.

[54] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, 2024.

[55] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

[56] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *Computational Visual Media*, 11(1):83–101, 2025.

[57] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023.

[58] Kadir Yilmaz, Adrian Kruse, Tristan Höfer, Daan de Geus, and Bastian Leibe. Volume transformer: Revisiting vanilla transformers for 3d scene understanding. *arXiv preprint arXiv:2604.19609*, 2026.

[59] Yuanwen Yue, Damien Robert, Jianyuan Wang, Sunghwan Hong, Jan Dirk Wegner, Christian Rupprecht, and Konrad Schindler. Litept: Lighter yet stronger point transformer. *arXiv preprint arXiv:2512.13689*, 2025.

[60] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025.

[61] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

A. Supplementary Material

A.1. Training Details

The training procedure and baseline adaptations are described in Sec. 5.1; Tab. 5 lists the detailed configurations.

The ScanNet experiments also use a single A100 GPU for PTv3 pretraining, PoinTTT finetuning, and inference. We will release the code with the exact configuration files, checkpoint schedules, and hardware settings.

Table 5. Training configurations for PTv3 pretraining and PoinTTT finetuning. All runs use AdamW and OneCycleLR with `div_factor`=10.

Setting	HOI4D	Forest	ScanNet
PTv3 Pretraining			
Epochs	50	100	800
Batch size	64	1	12
GPUs (A100)	1	1	1
Base LR	6e−3	6e−4	6e−3
Block LR	6e−4	6e−5	6e−4
`pct_start`	5%	5%	5%
`final_div_factor`	1000	1000	1000
PoinTTT Finetuning			
Epochs	50	200	200
Batch size	32	1	12
GPUs (A100)	8	1	1
LR (single group)	1e−4	1e−4	1e−4
`pct_start`	5%	10%	10%
`final_div_factor`	1000	100	100

A.2. Evaluation Pipeline Details

For static forest plots, PoinTTT resets the fast-weight state at the beginning of each test plot. The plot is partitioned into a sequence of fixed-size chunks, the offline update pass processes all chunks to build a plot-level state, and the apply pass decodes chunk predictions using that state. Per-chunk predictions are then mapped back to the original points, and class IoU and mIoU are computed over the full plot.

For 4D sequences, PoinTTT resets the fast-weight state only at the beginning of each video. At step t, the model receives the current three-frame chunk, updates the state, emits point-wise predictions, and carries the updated state to $t{+}1$. Future chunks are not used. This differs from window-based HOI4D evaluation, where methods typically process short windows and reset the backbone state at clip boundaries.

A.3. Additional Design Ablations

The main-paper ablation summary in Tab. 4 combines matched memory and capacity controls with the FOR-instanceV2 design sweeps. The no-aggregation model disables aggregation during both training and evaluation. Random membership changes which points belong to each chunk, while Hilbert and Z-order preserve spatial coherence.

ScanNet evaluation budgets. Tab. 4(h) reports additional chunk-size evaluations for ScanNet. The evaluation chunk budget is matched between PTv3 and PoinTTT.

Memory measurements. At approximately 5.2 million input points, full-scene PTv3 requires about 19 GB of peak inference memory, while PoinTTT uses about 0.3 GB with 10,240-point chunks. The recurrent fast-weight state has approximately 0.28 million parameters (about 1 MB); this state size is distinct from total inference memory. These measurements use different point budgets per forward pass.

nuScenes evaluation protocol. Both methods use the same PTv3 backbone and single-pass evaluation pipeline without TTA. PoinTTT adds temporal adaptation, giving 78.7 mIoU versus 78.1 for PTv3. The published PTv3 score of 80.4 uses TTA and is not directly comparable.

A.4. Limitations and Future Work

PoinTTT provides a memory-bounded interface for arbitrary-extent point-cloud segmentation, and several directions remain for future work. First, because the fast-weight state is propagated recurrently, inference is naturally sequential along the chunk axis. This keeps activation memory bounded by chunk size, but the total runtime still grows with the number of chunks. Future work could improve inference efficiency with parallel, hierarchical, or cached state propagation while preserving the recurrent context mechanism.

Second, our experiments cover four complementary settings: ScanNet for indoor segmentation, FOR-instanceV2 for large forest plots, and HOI4D for online 4D segmentation, together with nuScenes for outdoor driving LiDAR. Due to compute constraints, we do not scale the evaluation to a larger collection of datasets or longer training schedules. Extending PoinTTT to more domains, sensors, and scene types would further test the generality of fast-weight recurrence for point-cloud understanding.

Finally, PoinTTT is trained in a supervised setting and does not yet use large-scale point-cloud pretraining. We expect pretraining on diverse 3D and 4D data to improve robustness, cross-domain transfer, and long-range context aggregation. This is a natural next step, especially for settings with domain shift such as forests captured under different sensors, acquisition conditions, or vegetation distributions. We leave this as future work.