ROLLBACK-FREE STABLE BRICK GENERATION VIA REINFORCEMENT LEARNING FROM PHYSICAL REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

While autoregressive models have advanced 3D generation, creating physically stable brick structures remains a challenge due to the strict requirements of gravity and interconnectivity. Existing approaches rely on external physical simulators during inference to perform rejection sampling and brick-by-brick rollbacks, which severely bottlenecks efficiency. To address this, we propose a reinforcement learning paradigm that shifts physical validity enforcement from test-time correction to training-time policy optimization. By utilizing assembly-level rewards, the model optimizes for collision avoidance, global connectivity, structural interlocking, and shape conformity. This paradigm allows the model to internalize physical priors, enabling the first rollback-free generation of stable brick structures. Experimental results demonstrate that our approach achieves state-of-the-art generation quality while accelerating inference speed by orders of magnitude. Our code and dataset are available at `https://anonymous.4open.science/r/STABLE`.

1 INTRODUCTION

While large autoregressive models have driven profound advancements across various generative tasks (Achiam et al., 2023; Tian et al., 2024), their extension to 3D generation (Nash et al., 2020; Li et al., 2024) often overlooks a crucial practical requirement: physical constructibility. Generating structures from standard, discrete components, such as LEGO® bricks, demands strict adherence to physical constrains during the assembly process. To bridge this gap, pioneer works like BrickGPT (Pun et al., 2025) have explored 3D brick generation (Ge et al., 2024; Chung et al., 2021). A critical challenge in this domain is ensuring the physical stability of the generated structures. This requires the model to be capable of perceiving discrete, constrained, and interactive 3D geometry. Unlike virtual meshes or point clouds, physical assemblies must strictly respect gravity, balance, and precise interconnectivity, where a single misplaced brick can lead to structural collapse.

To address this, existing autoregressive approaches (Pun et al., 2025) achieve stability by tightly coupling the generation process with external physical simulators during inference time. As shown in Fig. 1, whenever an unstable or colliding placement is detected, the system triggers a brick-by-brick rejection sampling and physics-aware rollback mechanism, forcing the model to discard invalid steps and repeatedly re-sample tokens until a stable configuration is found. Unfortunately, this trial-and-error paradigm is highly counter-intuitive to the nature of autoregressive decoding, since it breaks the continuous sequence generation assumption, leading to the degradation of generation quality or collapse of generative model. Meanwhile, the constant need to halt generation, query a simulator, and discard previously generated tokens severely bottlenecks inference efficiency.

The reliance on inference-time simulator-based correction reflects a fundamental limitation of standard autoregressive training for physically constrained generation. Although the demonstrations in the dataset are physically valid, such principle is only implicitly embedded in the completed structures rather than exposed as explicit supervision. Standard autoregressive modeling learns through by-token imitation, encouraging the model to reproduce local sequence patterns and shape-oriented brick placements. As a result, supervised next-token imitation alone provides insufficient pressure for the model to uncover and internalize the hidden construction principles behind stable examples. Motivated by this inherent inefficiency, we raise a fundamental question:

Is structural stability inherently learnable by the autoregressive generative model itself?

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.

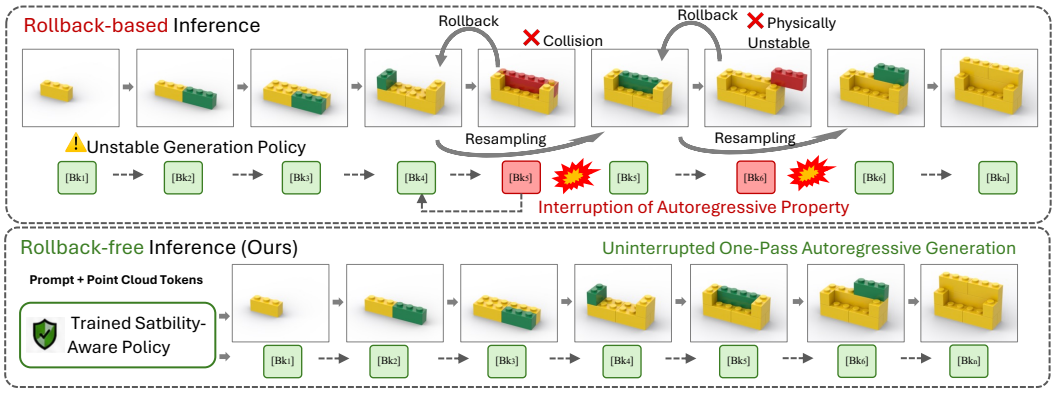


Figure 1: **Comparison between rollback-based and rollback-free inference**. Rollback-based generation requires resampling after unstable steps, interrupting the autoregressive process, while our stability-aware policy enables uninterrupted one-pass generation.

In this work, we demonstrate that structural stability can be learned through **Reinforcement Learning from Physical Rewards (RLPR)**, a novel training paradigm which we propose that optimizes generative models using rewards grounded in physical and construction constraints. Our key insight is to reformulate stable brick generation from by-token imitation into structure-level policy learning. A generated brick sequence can be viewed as an action trajectory and its final assembly can be evaluated by construction-level objectives through reinforcement learning. In this way, global feedback from the completed structure can be propagated back to the token decisions that produced it.

We further decompose assembly-oriented structural stability into several verifiable construction properties. A buildable brick assembly should first satisfy geometric validity, where bricks occupy legal positions without spatial overlap. It should also form a globally coherent structure rather than disconnected or floating fragments. Beyond global coherence, it should exhibit local mechanical robustness, where upper-layer bricks interlock with lower-layer bricks instead of forming weak vertical stacks. These properties together define the stability requirements of brick construction.

To this end, we propose `STABLE`, a reinforcement-learning framework that shifts construction validity enforcement from inference-time correction to training-time policy optimization. We instantiate the above decomposition with deterministic assembly-level rewards, including a collision penalty for geometric validity, a connectivity reward for global coherence, an interlocking reward for local robustness, and a shape reward for aligning the target 3D geometry. By jointly optimizing these objectives, `STABLE` turns latent construction principles into explicit learning signals.

At inference time, `STABLE` directly emits a complete brick sequence in one uninterrupted autoregressive pass, without rejection sampling, brick-by-brick rollback, or external physical simulation. Our experiments show that `STABLE` achieves extremely low levels of collisions and instability while ensuring that the constructed brick assembly closely matches the target. Compared to rollback-based method like BrickGPT (Pun et al., 2025), `STABLE` reduces inference latency by 94%.

Contributions. In summary, our main contributions are three-fold:

- We challenge the conventional view that physically stable discrete 3D generation must rely on external test-time simulators and greedy rollback correction. Instead, we show that stability-aware and collision-free generation can be learned as an intrinsic capability of the generative policy through Reinforcement Learning from Physcial Rewards (RLPR).
- We introduce a comprehensive physics- and geometry-aware reward system (incorporating collision, shape, interlocking, and connectivity constraints) that successfully embeds real-world brick structure construction priors into the autoregressive generative model.
- We demonstrate the first rollback-free framework that achieves structurally stable brick assembly generation without rejection sampling or brick-by-brick rollback, substantially improving inference efficiency while maintaining high geometric and structural quality.

2 Related Works

3D and Brick-Structure Generation. Autoregressive models have been extended from language and visual generation to 3D structures, including mesh generation (Nash et al., 2020), autoregressive point or shape generation (Cheng et al., 2022; Luo et al., 2023), general point model pretraining (Li et al., 2024), and language-compatible 3D tokenization (Yin et al., 2023). While these methods primarily focus on geometric fidelity, brick-structure generation is substantially more constrained because each predicted token corresponds to a discrete physical component whose placement must satisfy geometric and structural requirements. Earlier methods optimize LEGO layouts through iterative refinement and stability analysis (Luo et al., 2015), while learning-based approaches formulate construction as sequential decision making over discrete components (Chung et al., 2021) or study generative pipelines for coherent LEGO structures (Ge et al., 2024). Most closely related to our work, BrickGPT generates physically stable and buildable brick structures with an autoregressive language model (Pun et al., 2025), but relies on test-time validity checking, rejection sampling, and rollback to correct invalid placements. In contrast, we shift physical feasibility from inference-time correction to training-time policy learning, enabling rollback-free autoregressive generation.

Reward Optimization for Physical Constraints. Reinforcement learning provides a general mechanism for optimizing sequence-level objectives that are difficult to capture with supervised likelihood alone. Representative approaches include PPO (Schulman et al., 2017), RLHF for aligning language models with human preferences (Ouyang et al., 2022), and group-relative policy optimization for reasoning-oriented training (Shao et al., 2024). Our method follows this reward-optimization paradigm but targets physical construction constraints. We formulate Reinforcement Learning from Physical Rewards (RLPR), where verifiable assembly-level signals, including collision, connectivity, interlocking, and shape consistency, provide structure-level feedback that is propagated back to autoregressive brick decisions. Unlike preference-oriented or correctness-oriented reward optimization, RLPR directly trains the generative policy to internalize physical feasibility, removing the need for external inference-time correction.

3 Datasets

We study two complementary regimes: PointCloud2Brick uses voxelized geometry and globally positioned bricks, whereas BrickNet Kulits & Schmid (2026) represents diverse assemblies through part connections. Further details are provided in Appendix A.

PointCloud2Brick. PointCloud2Brick is constructed from StableText2Brick Pun et al. (2025) and contains 42,604 training and 4,785 test structures from 21 ShapeNet categories Chang et al. (2015). Each target is rasterized into $V \in \{0,1\}^{20\times20\times20}$, with occupied coordinates serialized as the input point cloud $P = \{(x, y, z) \mid V[x, y, z] = 1\}$. The output is an autoregressive brick sequence

$$B = [b_1, \ldots, b_T], \qquad b_t = (d_t, p_t), \tag{1}$$

where d_t is an oriented brick type and $p_t = (x_t, y_t, z_t)$ its discrete position. The target occupancy V is retained for geometry-aware RL rewards.

BrickNet. BrickNet is derived from large-scale human-designed LDraw objects and contains thousands of part types with stud, hinge, axle, ball, and fixed connections. It represents an assembly as a graph $G = (\mathcal{V}, \mathcal{E})$, whose nodes are parts and edges are valid spatial connections. Sampling a spanning tree yields

$$S = [(v_1), (v_2, e_{p_2,2}), \ldots, (v_T, e_{p_T,T})], \tag{2}$$

where $p_t < t$ is an assembled parent. Each step predicts the next part and its connection, providing a graph-backed test of reward generalization beyond global coordinates.

4 STABLE Framework

The core idea of STABLE is to convert physical feasibility of autoregressive brick generation from a test-time correction problem into a training-time policy learning problem. To accomplish this

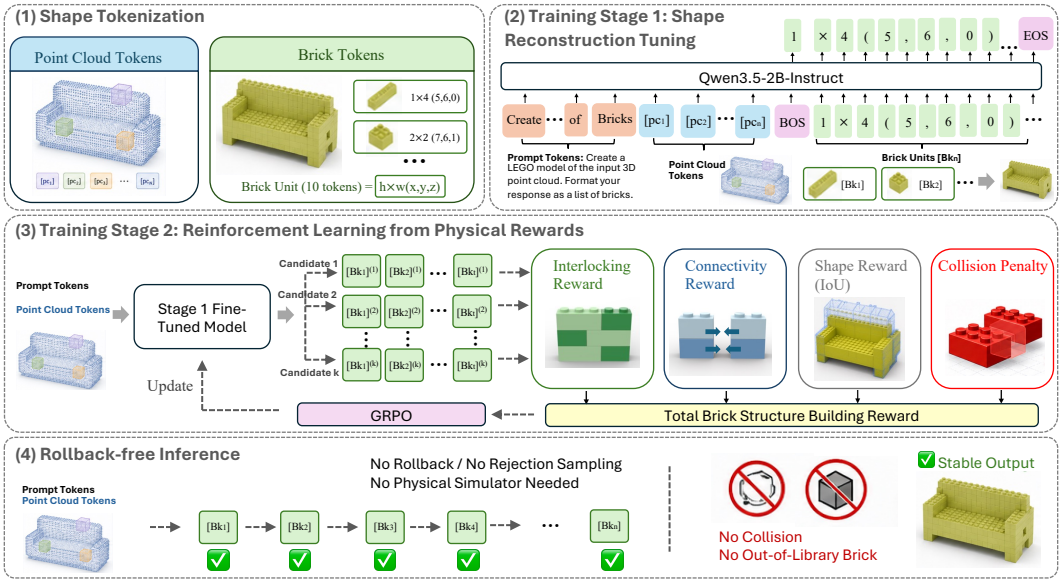


Figure 2: **Overview of STABLE**. SFT learns brick generation, and structure-level reward post-training enables rollback-free inference across coordinate- and graph-backed representations.

goal, we design a two-stage post-training framework as shown in Fig. 2. The first stage provides the model with a reliable brick construction prior, while the second stage focuses on improving the physical stability of the generated brick assembly. This design is motivated by a key limitation of supervised brick generation. A model trained only to imitate ground-truth brick sequences can learn the output format and approximate the target geometry, but it is not explicitly optimized for the consequences of its own free-running generations. Small local errors may accumulate into voxel collisions, disconnected components, or weakly interlocked assemblies. STABLE addresses this gap by combining shape-oriented supervised tuning with stability-oriented reinforcement learning.

4.1 Stage I: Shape Reconstruction with Supervised Fine-Tuning

The first stage aims to scaffold the model with the basic capability of translating a target point cloud into a valid brick sequence. We fine-tune a pretrained instruction-following language model (`Qwen3.5-2B` (Yang et al., 2025) in our experiments) on PointCloud2Brick or Bricknet, where each sample contains (1) a structured instruction prompt, (2) a shape prompt and (3) a target brick layout written in the prescribed textual format. This stage is not intended to solve physical stability by itself.

Instead, it teaches the model the grammar of brick generation, the use of the allowed brick library, and the coarse correspondence between voxelized shapes and brick decompositions. The supervised model provides a stable starting distribution from which the model can generate parseable and shape-relevant candidates, which serves as an initialization for later policy learning. This is crucial for reinforcement learning: if the initial policy cannot produce meaningful brick structures, the reward signals become sparse and noisy. We will show its indispensability later in experiments in Section 6.

4.2 Stage II: Reinforcement Learning from Physical Rewards

Although the brick sequences data used in supervised fine-tuning are derived from physically stable structures, their stability information is only implicitly and indirectly encoded in the autoregressive ordering of bricks. SFT only trains the model to imitate the reference autoregressive sequence. It does not evaluate the model's own free-running generations, nor can it propagate the physical consequences of a completed structure back to the token decisions that produced it. Therefore, even if the training targets are stable, the model may still generate sequences whose final assemblies contain collisions, disconnected components, or weak inter-layer support. The key limitation is not the absence of stability in the data, but the absence of an explicit mechanism that tells the model how its generated bricks affect the physical validity of the whole structure.

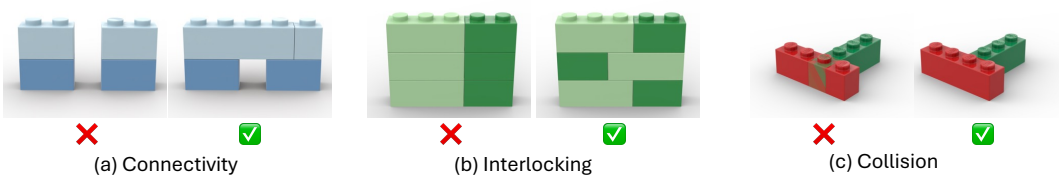


Figure 3: Illustration of rewards related to physical stability

To address this limitation, our STABLE introduces RLPR, a reinforcement-learning-based approach that uses the model's own generated assemblies as the object of optimization. As shown in Fig 2, we first decompose physical stability into several verifiable structural properties, such as spatial feasibility (collision), interlocking, connectivity, and shape consistency. These properties serve as the feedback interface for reinforcement learning, while their concrete reward definitions are detailed in Section 5. For each point-cloud prompt, the policy samples multiple candidate brick sequences; each completed sequence is parsed into a brick assembly and assigned structure-level feedback. We then use Group Relative Policy Optimization (GRPO) (Guo et al., 2025) to propagate this feedback along the sampled autoregressive trajectory, increasing the likelihood of token sequences that lead to more stable structures and suppressing those that lead to invalid ones. In this way, delayed physical consequences are transformed into token-level policy updates that govern the entire generation process, rather than being handled by rollback or rejection sampling at inference time.

4.3 Rollback-Free Decoding

After the two-stage training process, inference follows ordinary autoregressive decoding. Given a point-cloud prompt, the model directly emits a complete brick sequence without simulator calls, rejection sampling, or brick-by-brick rollback. This preserves the causal continuity of autoregressive generation and avoids the efficiency bottleneck of test-time correction. The generated structure is therefore the direct output of a stability-aware policy learned during training.

5 Reward Design

The reward module translates construction requirements into scalar learning signals for policy optimization. Given a generated sequence $\mathbf{y}$ conditioned on an input $\mathbf{x}$, we first parse it into an assembly $\hat{\mathcal{B}}$. The parser and the geometric representation are dataset-dependent: an assembly may be represented by globally positioned bricks on a voxel grid or by parts and their attachment relations. Invalid outputs, including empty responses and malformed part specifications, are treated as fails.

We formulate the reward as a weighted composition of four normalized criteria: $R(\mathbf{y}, \mathbf{x}) = \lambda_{\text{col}} r_{\text{col}}(\mathbf{y}) + \lambda_{\text{shape}} r_{\text{shape}}(\mathbf{y}, \mathbf{x}) + \lambda_{\text{inter}} r_{\text{inter}}(\mathbf{y}) + \lambda_{\text{conn}} r_{\text{conn}}(\mathbf{y})$, where all λ values are non-negative coefficients. For an invalid generation, we instead assign $R(\mathbf{y}, \mathbf{x}) = -\lambda_{\text{invalid}}$. This formulation separates the general construction objectives from their dataset- and representation-specific scales. The concrete hyperparameter values used in our experiments are reported in Section 6.

Collision Penalty. The collision term penalizes spatial overlaps between parts. For a voxelized assembly, let $C_{\hat{\mathcal{B}}}(v)$ denote the number of generated bricks occupying voxel v, and define

$$N_{\text{col}} = \sum_{v} \mathbb{1}[C_{\hat{\mathcal{B}}}(v) > 1]. \tag{3}$$

For part-graph representations, N_{col} is analogously obtained from geometric clashes between positioned parts. We normalize and clip the penalty as

$$r_{\text{col}}(\mathbf{y}) = -\min\left(\frac{N_{\text{col}}}{\tau_{\text{col}}}, 1\right) \in [-1, 0], \tag{4}$$

where $\tau_{\text{col}} > 0$ controls when the penalty saturates. Clipping prevents severe failures from producing excessively large gradients, while λ_{col} independently controls the importance of collision avoidance in the overall objective.

Shape Conformity Reward. The shape term measures agreement between the generated assembly and its target geometry. We denote a representation-appropriate normalized similarity by

$$r_{\text{shape}}(\mathbf{y}, \mathbf{x}) = S_{\text{shape}}(\hat{\mathcal{B}}, \mathbf{x}) \in [0, 1]. \tag{5}$$

For a voxel target V^*, we rasterize the generated assembly as $\hat{V}(v) = \mathbb{1}[C_{\hat{\mathcal{B}}}(v) > 0]$ and instantiate the similarity with volumetric intersection-over-union:

$$S_{\text{shape}}(\hat{\mathcal{B}}, \mathbf{x}) = \text{IoU}(\hat{V}, V^*) = \frac{\sum_v \mathbb{1}[\hat{V}(v) = 1 \wedge V^*(v) = 1]}{\sum_v \mathbb{1}[\hat{V}(v) = 1 \vee V^*(v) = 1]}. \tag{6}$$

The shape reward is not gated by collision-freeness. It therefore supplies a dense learning signal even when the early policy produces imperfect assemblies, while the collision term prevents the policy from improving target coverage through mutually intersecting parts.

Interlocking Reward. Collision-free shape matching does not necessarily imply mechanical robustness. We therefore reward parts that bridge multiple supporting parts rather than forming weak one-to-one stacks. For each generated part b_i, let $\mathcal{S}^-(b_i)$ denote its set of supporting parts. This relation can be inferred from geometric contact in a coordinate-based representation or read from the attachment graph when explicit connection semantics are available.

For the voxelized brick representation, let z_i denote the layer index and $\Phi(b_i) \subset \mathbb{Z}^2$ the horizontal footprint: $\Phi(b_i) = \{(x, y) \mid x_i \le x < x_i + h_i,\ y_i \le y < y_i + w_i\}$, with the support set $\mathcal{S}^-(b_i) = \{b_j \mid z_j = z_i - 1,\ \Phi(b_i) \cap \Phi(b_j) \neq \emptyset\}$. Given an interlocking threshold m_{inter}, a non-grounded part is considered interlocked when it bridges at least m_{inter} distinct supports:

$$I_{\text{inter}}(b_i) = \mathbb{1}\left[b_i \text{ is non-grounded } \wedge\ |\mathcal{S}^-(b_i)| \ge m_{\text{inter}}\right]. \tag{7}$$

The structure-level score is the fraction of non-grounded parts satisfying this condition:

$$S_{\text{inter}}(\hat{\mathcal{B}}) = \frac{\sum_{b_i \in \hat{\mathcal{B}}} I_{\text{inter}}(b_i)}{\max\left(\sum_{b_i \in \hat{\mathcal{B}}} \mathbb{1}[b_i \text{ is non-grounded}], 1\right)}. \tag{8}$$

Let $g(\hat{\mathcal{B}}) \in \{0, 1\}$ be a representation-specific basic-validity gate, which requires a voxel assembly to be collision-free and in bounds and a connection-based assembly to contain only geometrically valid attachments. The gated interlocking reward is

$$r_{\text{inter}}(\mathbf{y}) = g(\hat{\mathcal{B}}) S_{\text{inter}}(\hat{\mathcal{B}}) \in [0, 1]. \tag{9}$$

Connectivity Reward. The connectivity reward encourages a single coherent assembly rather than disconnected or floating components. Let $\mathcal{O}$ denote the assembly elements, occupied voxels for a voxel representation or parts for a connection-graph representation, and let $\mathcal{D} \subseteq \mathcal{O}$ contain the elements disconnected from the main grounded component. We define

$$S_{\text{conn}}(\hat{\mathcal{B}}) = 1 - \frac{|\mathcal{D}|}{\max(|\mathcal{O}|, 1)}. \tag{10}$$

Using the same gate as interlocking, the connectivity reward is $r_{\text{conn}}(\mathbf{y}) = g(\hat{\mathcal{B}}) S_{\text{conn}}(\hat{\mathcal{B}}) \in [0, 1]$.

6 Experiments

Setup. We implement `STABLE` as a two-stage post-training pipeline on top of `Qwen3.5-2B` (Yang et al., 2025). We first conduct supervised fine-tuning and obtain an SFT intermediate model `STABLE-SFT`. Starting from `STABLE-SFT` model, we perform GRPO training and conclude the final `STABLE` model. We use full-parameter fine-tuning for both SFT and GRPO. The model is trained on 4 NVIDIA H100/H200 GPUs with vLLM (Kwon et al., 2023) acceleration for on-policy generation and DeepSpeed (Rasley et al., 2020) ZeRO-2 for memory-efficient optimization. We conduct full- parameter fine-tuning on the model for more stable conver- gence and higher final accuracy. Training details are provided in the Appendix B.

Reward instantiation. Following the parameterized objective in Section 5, we set $\lambda_{\text{invalid}} = 10$, $\lambda_{\text{col}} = 10$, $\tau_{\text{col}} = 5$, $\lambda_{\text{shape}} = 5$, $\lambda_{\text{inter}} = 3$, $\lambda_{\text{conn}} = 2$, and $m_{\text{inter}} = 2$ on PointCloud2Brick. Thus, its collision term is equivalent to $\max(-10, -2N_{\text{col}})$, while the maximum shape, interlocking, and connectivity contributions are 5, 3, and 2, respectively. For BrickNet, we set $\lambda_{\text{inter}} = 0$ because its graph-backed representation does not use the layer-wise interlocking criterion, while retaining the remaining coefficients unless otherwise stated. The underlying collision, connectivity, and target-similarity evaluators follow each dataset's native assembly representation.

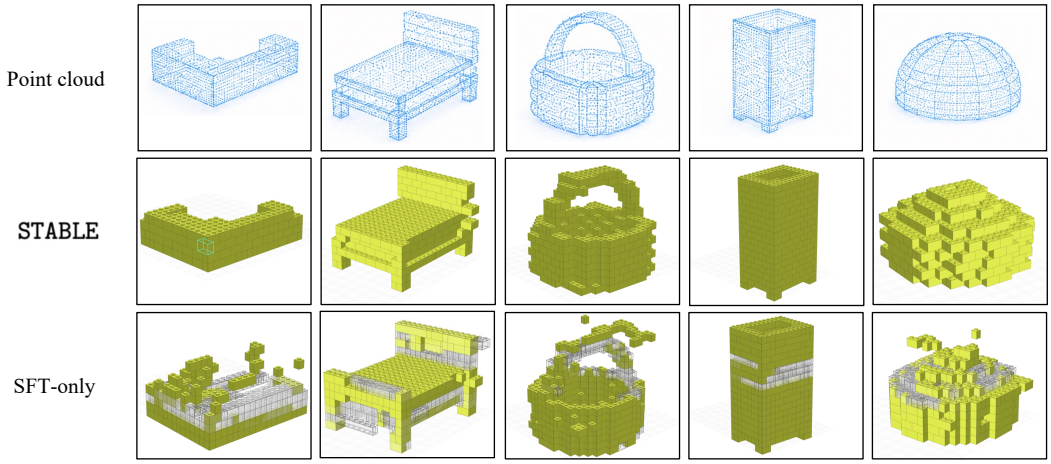


Figure 4: Qualitative results of Point Cloud to Brick generations. Transparency indicates a collision.

Table 1: **Quantitative results.** Main comparison of brick-generation quality. All models are evaluated with greedy decoding and without test-time rollback or rejection sampling except BrickGPT+RB&RS. Mean and Min Stability are from simulator statistics as in Pun et al. (2025).

Method	Coll.-Free Rate	Voxel IoU	Mean Stability	Min Stability	Conn. Ratio	Interlock. Score	Seam Cov.	Mean Bricks	Avg. Time (s/sample)
BrickGPT Pun et al. (2025)	0.10	0.039	0.214	0.031	0.00	0.023	0.056	110.9	32.97
BrickGPT+RB&RS	1.00	0.349	0.983	0.781	0.54	0.148	0.159	168.6	895.41
LegoACE Xu et al. (2025)	0.05	0.080	0.041	0.007	0.12	0.103	0.086	79.3	–
`Gemini-3.1-Pro`	0.02	0.276	0.913	0.643	0.00	0.119	0.074	64.32	–
`Llama-4` Touvron et al. (2023)	0.00	0.000	–	–	0.00	0.000	0.000	0.0	563.12
`Qwen3.5-122B-A10B`	0.00	0.000	–	–	0.00	0.000	0.000	0.0	412.74
`Qwen3.5-2B` (base)	0.00	0.000	–	–	0.00	0.000	0.000	0.0	143.89
`STABLE-ZERO`	0.00	0.000	–	–	0.00	0.000	0.000	0.0	–
`STABLE-SFT`	0.00	0.813	–	–	0.05	0.012	0.010	661.5	183.63
`STABLE`	**0.99**	**0.907**	**1.000**	**1.000**	**0.97**	**0.722**	**0.676**	216.5	57.64

Evaluation. We evaluate all models on samples from the PointCloud2Brick test split. All methods are decoded with greedy decoding and a maximum generation length of 8,192 tokens. We evaluate each generated sequence at two levels. At the geometric level, we report voxel IoU between the generated occupancy and target occupancy grid. At the physical-structure level, we report collision-free rate, connectivity ratio, interlocking score, and seam coverage rate. See details in Apendix C.

6.1 STABLE BRICK CONSTRUCTION RESULTS

Baselines. We use `Gemini-3.1-Pro` (Team et al., 2023) and `Llama-4-Scout-17B-16E-Instruct` (Touvron et al., 2023) as representative closed- and open-source LLM baselines, respectively, and additionally evaluate `Qwen3.5-122B-A10B` (Yang et al., 2025) to study model scale. We also compare with BrickGPT (Pun et al., 2025) and LegoACE (Xu et al., 2025). BrickGPT is evaluated with its original text prompt rather than the PointCloud2Brick point-cloud condition. Consequently, its voxel IoU is reported only for completeness and should not be interpreted as a direct shape-reconstruction comparison. However, the generated brick assemblies are evaluated using the same collision, stability, connectivity, interlocking, and seam-coverage metrics, so these structure-quality results remain informative and comparable. All methods use rollback-free decoding except BrickGPT+RB&RS, which retains its original rollback and rejection-sampling procedure.

Remark 1: General-purpose LLMs do not directly solve point-cloud-conditioned brick construction. As shown in Table 1, general-purpose LLMs such as `Llama-4`, `Qwen3.5-122B-A10B`, and `Qwen3.5-2B` produce almost no valid brick structures under the PointCloud2Brick prompt. This indicates that model scale alone is insufficient to acquire the discrete output grammar and physical constraints required by the task. BrickGPT and LegoACE provide complementary references for brick-structure quality, but they are not native PointCloud2Brick models. In particular, BrickGPT's voxel IoU is not directly comparable since it receives its original text condition; its collision, stability, and connectivity results instead characterize the structural quality of its generated assemblies.

Table 2: Steps to invalidation on BrickNet under graph-based evaluation.

Method	Connectivity (AS / NS)	Collision (AS / NS)
BrickNet	48.6 / 95.7	11.6 / 17.5
BrickNet + Collision Penalty	50.4 / 102.4	61.7 / 90.8
BrickNet + STABLE	90.1 / 170.6	68.2 / 104.2

Table 3: Reward ablation on PointCloud2Brick test samples. All variants are initialized from the same SFT checkpoint and trained with GRPO.

Variant	Coll.-Free Rate	Mean Coll. Voxels	Voxel IoU	Conn. Ratio	Connected Rate	Interlock. Score	Seam Cov.	In-Bounds Rate	Mean Bricks
STABLE	**0.99**	**0.05**	0.907	**0.97**	**0.49**	0.722	**0.676**	**1.00**	216.5
w/o Collision	0.72	14.51	0.897	0.91	0.46	**0.734**	0.638	**1.00**	247.1
w/o Connectivity	0.93	9.50	0.919	0.73	0.45	0.709	0.653	**1.00**	230.0
w/o Structural	0.96	12.46	**0.978**	0.46	0.02	0.069	**0.676**	0.76	387.0

Remark 2: SFT learns shape, but not physical feasibility. As shown in Table 1, STABLE−SFT achieves a high voxel IoU of 0.813, showing that supervised fine-tuning teaches the model to reconstruct target geometry. However, its collision-free rate remains 0.00, and its structural metrics are almost zero. The qualitative examples in Fig. 4 show the same pattern: STABLE−SFT outputs roughly cover the target shape but contain redundant, overlapping, and weakly connected bricks. This confirms that maximum-likelihood imitation alone is insufficient for stable construction. At the same time, STABLE−ZERO, trained directly from the base model, still scores zero on all metrics, meaning SFT remains an essential component since RL alone can not tell the model how to build bricks.

Remark 3: RL is the key to learning physical stability. After GRPO training, STABLE improves the collision-free rate to 0.99 while maintaining strong voxel IoU of 0.907. It also achieves the strongest rollback-free structural metrics, including 1.000 mean stability, 1.000 min stability, 0.97 connectivity ratio, 0.722 interlocking score, and 0.676 seam coverage. These results show that our RL training is the decisive step for stable generation, turning a model that merely reconstructs shapes into one that actively prefers collision-free, connected, and interlocked brick assemblies.

Remark 4: STABLE outperform rollback-based BrickGPT with much lower inference cost. As shown in Table 1, with external rollback and rejection sampling, BrickGPT can reach 1.00 collision-free rate (algorithm enforced) but requires 895.41 seconds per sample due to the calling of external physical simulator. In contrast, STABLE achieves comparable or better stability, while using only 57.64 seconds per sample. This demonstrates that moving physical evaluation from inference-time correction to training-time policy learning yields both stronger generation quality and faster inference.

6.2 Generalization to BrickNet

We further evaluate reward-based post-training on BrickNet using its native graph-backed assembly representation. Because BrickNet represents heterogeneous part connections rather than layer-wise voxel supports, we omit the interlocking reward. Table 2 compares the original model with collision-only and full collision–shape–connectivity training. Following the original protocol, we report the number of generation steps before invalidation, where higher values are better.

Remark 5: Collision-aware RL generalizes beyond voxel-based generation. The collision penalty increases collision-valid generation from 11.6/17.5 to 61.7/90.8 steps under AS/NS, an improvement of more than five times, while slightly improving connectivity from 48.6/95.7 to 50.4/102.4. Thus, collision-aware policy optimization transfers from voxel coordinates to graph-backed generation.

Remark 6: The complete reward provides stronger cross-representation generalization. BrickNet + STABLE performs best across all measurements, reaching 90.1/170.6 connectivity-valid and 68.2/104.2 collision-valid steps. Relative to BrickNet, this nearly doubles connectivity and improves collision by approximately six times, showing that the combined rewards generalize across assembly representations without interlocking supervision.

6.3 Training Dynamics

Fig. 5 visualizes the RL training dynamics of STABLE and ablations. Details in Appendix D.

Remark 7: RL learns stability without sacrificing shape fidelity. As shown in Fig. 5, the total reward increases steadily during GRPO training, while the collision reward quickly approaches

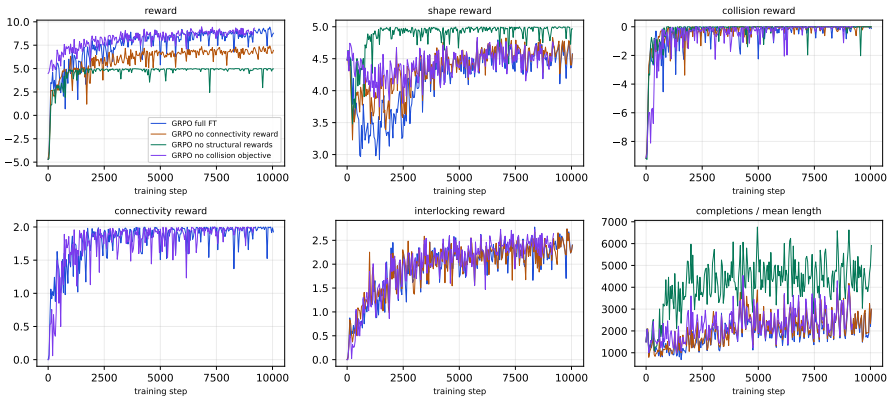


Figure 5: GRPO training dynamics under different reward settings.

zero penalty and the connectivity and interlocking rewards gradually improve. Interestly, the shape reward temporarily drops in the early stage, suggesting that the policy first departs from purely shape-oriented imitation to explore more construction-valid brick decompositions. It then recovers as training progresses, indicating that STABLE learns to satisfy stability constraints while preserving target-shape fidelity. Overall, RL transforms the SFT model from a shape imitator into a policy that generates assemblies that are shape-consistent, collision-free, connected, and interlocked.

7 Ablation Study

We ablate the reward components of STABLE to understand how each physical signal contributes to the final construction quality. All variants start from the same STABLE−SFT checkpoint and are trained with GRPO under identical settings. The reward configurations are summarized in Table 4, and the quantitative results are shown in Table 3.

Table 4: Ablation configurations.

Variant	R_{col}	R_{shape}	R_{inter}	R_{conn}
STABLE	✓	✓	✓	✓
w/o Coll.	×	✓	✓	✓
w/o Conn.	✓	✓	✓	×
w/o Struct.	✓	✓	×	×

Remark 8: Collision reward is necessary for exact spatial feasibility. Removing R_{col} significantly increases the mean number of colliding voxels from 0.05 to 14.51 and reduces the collision-free rate from 0.99 to 0.72. Although the model still obtains reasonable connectivity and interlocking scores, structural rewards alone cannot guarantee voxel-level non-overlap. This confirms that explicit collision supervision is essential for discrete brick placement.

Remark 9: Connectivity reward improves global assembly coherence. Without R_{conn}, the connectivity ratio drops from 0.97 to 0.73. The model can still produce locally interlocked structures, but it becomes less reliable at forming a globally coherent assembly. Thus, connectivity reward provides a complementary global constraint beyond local brick support.

Remark 10: Structural rewards prevent voxel-filling degeneration. When both connectivity and interlocking rewards are removed, the model reaches the highest voxel IoU of 0.978 but suffers from poor structural quality, with connectivity ratio dropping to 0.46 and interlocking score to 0.069. We further observe that the model tends to exploit the shape reward by using many small 1×1 bricks to fill target voxels, leading to high bircks usage. Such outputs can match the target shape but lose the coherent and interlocked structure required for buildable assemblies. Therefore, structural rewards are necessary to guide the model toward physically meaningful brick constructions.

8 Conclusion

We proposed STABLE, a rollback-free framework for stable brick-structure generation across voxelized point-cloud and graph-backed assembly representations. By replacing inference-time correction with training-time structure-level feedback, STABLE teaches collision avoidance, connectivity, interlocking when applicable, and shape conformity. Reward-based post-training improves geometric fidelity and structural validity on PointCloud2Brick and enables substantially longer connectivity- and collision-valid generation on BrickNet, while reducing reliance on expensive test-time correction. These results suggest that discrete 3D construction priors can be internalized by an autoregressive policy across different assembly representations.

LLM USE STATEMENT

LLM is used for polish the grammar and word selection of the paper. The idea, experiments and writing are completely done by the authors.

REPRODUCIBILITY STATEMENT

We provide the source code, dataset, and implementation details necessary to reproduce our experiments at `https://anonymous.4open.science/r/STABLE`. Additional training and evaluation settings are described in the main paper and supplementary material.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Alan Chen and Changliu Liu. AssemblyComplete: 3d combinatorial construction with deep reinforcement learning. *arXiv preprint arXiv:2410.15469*, 2024.

An-Chieh Cheng, Xueting Li, Sifei Liu, Min Sun, and Ming-Hsuan Yang. Autoregressive 3d shape generation via canonical mapping. In *European Conference on Computer Vision*, pp. 89–104. Springer, 2022.

Hyunsoo Chung, Jungtaek Kim, Boris Knyazev, Jinhwi Lee, Graham W Taylor, Jaesik Park, and Minsu Cho. Brick-by-brick: Combinatorial construction with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5745–5757, 2021.

Jiahao Ge, Mingjun Zhou, and Chi-Wing Fu. Learn to create simple lego micro buildings. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Peter Kulits and Cordelia Schmid. Bricknet: Graph-backed generative brick assembly. *arXiv preprint arXiv:2604.22984*, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Zhe Li, Zhangyang Gao, Cheng Tan, Bocheng Ren, Laurence T Yang, and Stan Z Li. General point model pretraining with autoencoding and autoregressive. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20954–20964, 2024.

Ruixuan Liu, Kangle Deng, Ziwei Wang, and Changliu Liu. StableLego: Stability analysis of block stacking assembly. *IEEE Robotics and Automation Letters*, 9(11):9383–9390, 2024.

Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. Legolization: Optimizing lego designs. *ACM Transactions on Graphics*, 34(6): 1–12, 2015. doi: 10.1145/2816795.2818091.

Simian Luo, Xuelin Qian, Yanwei Fu, Yinda Zhang, Ying Tai, Zhenyu Zhang, Chengjie Wang, and Xiangyang Xue. Learning versatile 3d shape generation with improved auto-regressive models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Lin Ma, Jiangtao Gong, Hao Xu, Hao Chen, Hao Zhao, Wenbing Huang, and Guyue Zhou. Planning assembly sequence with graph transformer. In *2023 IEEE International Conference on Robotics and Automation*, pp. 12395–12401, 2023.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter Battaglia. PolyGen: An autoregressive generative model of 3D meshes. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7220–7229. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/nash20a.html.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Maxim Peysakhov and William C. Regli. Using assembly representations to enable evolutionary design of lego structures. *AI EDAM*, 17(2):155–168, 2003.

Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating physically stable and buildable brick structures from text. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14798–14809, 2025.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3505–3506, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Haowei Wen, Ruixuan Liu, Weiyi Piao, Siyu Li, and Changliu Liu. BrickSim: A physics-based simulator for manipulating interlocking brick assemblies. *arXiv preprint arXiv:2603.16853*, 2026.

Hao Xu, Yuqing Zhang, Yiqian Wu, Xinyang Zheng, Yutao Liu, Xiangjun Tang, Yunhan Yang, Ding Liang, Yingtian Liu, Yuanchen Guo, et al. Legoace: Autoregressive construction engine for expressive lego® assemblies. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–11, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Fukun Yin, Xin Chen, Chi Zhang, Biao Jiang, Zibo Zhao, Jianyuan Fan, Gang Yu, Taihao Li, and Tao Chen. Shapegpt: 3d shape generation with a unified multi-modal language model. *arXiv preprint arXiv:2311.17618*, 2023.

A POINTCLOUD2BRICK DATASET

We introduce **PointCloud2Brick**, a point-cloud-conditioned brick-structure corpus generation dataset constructed from StableText2Brick (Pun et al., 2025). PointCloud2Brick reformulates stable toy-brick structures as paired *point-cloud-token inputs* and *brick-token outputs*. Instead of using natural-language captions as the primary condition, each sample provides a voxelized 3D point cloud that specifies the target geometry, and the model is trained to autoregressively generate a sequence of physical brick units. This formulation directly supports geometry-grounded brick generation and provides a unified data interface for both supervised shape reconstruction and reinforcement learning with stability rewards.

A.1 POINTCLOUD2BRICK OVERVIEW

PointCloud2Brick is constructed from StableText2Brick (Pun et al., 2025), which contains 47,389 physically stable brick structures corresponding to 3D objects from 21 ShapeNet (Chang et al., 2015) categories. Each structure is placed in a discrete $20 \times 20 \times 20$ voxel world, where each voxel corresponds to a unit cube. All bricks are 1 unit tall and are selected from a library of 8 brick types, yielding 14 oriented variants.

We use the original train/test split and convert each brick structure into a paired point-cloud and brick-sequence representation of (Pun et al., 2025). The resulting PointCloud2Brick dataset contains 42,604 training samples and 4,785 test samples. PointCloud2Brick is designed to support both supervised fine-tuning and reinforcement learning, where the latter additionally uses voxel targets for rewards.

A.2 POINT CLOUD TOKENIZATION

The input side of PointCloud2Brick is a tokenized point cloud derived from the voxel occupancy of the target structure. Given a ground-truth brick layout, we first rasterize all bricks into a binary occupancy grid $\mathbf{V} \in \{0,1\}^{20\times20\times20}$. We then enumerate all occupied voxel coordinates:

$$\mathcal{P} = \{(x_i, y_i, z_i) \mid \mathbf{V}[x_i, y_i, z_i] = 1\}. \tag{11}$$

Each occupied coordinate is serialized as a point token in the form `(x,y,z)`, and the full point cloud is represented as a sequence of coordinate tokens: $\texttt{(x}_1\texttt{,y}_1\texttt{,z}_1\texttt{)}$, $\texttt{(x}_2\texttt{,y}_2\texttt{,z}_2\texttt{)}$, ..., $\texttt{(x}_n\texttt{,y}_n\texttt{,z}_n\texttt{)}$.

This tokenization preserves the target geometry at voxel resolution while avoiding any assumption about the underlying brick decomposition. The conversion from a brick structure to a point cloud is deterministic, but the inverse problem is not unique: multiple valid brick assemblies may correspond to the same occupied voxel set. Therefore, the model must do more than copy coordinates; it must infer a compact, legal, and physically plausible decomposition into available brick units.

A.3 BRICK TOKENIZATION

Follow (Pun et al., 2025), the output side of PointCloud2Brick is an autoregressive sequence of brick tokens. Each brick is represented by its oriented footprint and its integer anchor position in the voxel world:

$$Bk_t = (d_t, p_t), \qquad d_t \in \mathcal{D}, \quad p_t = (x_t, y_t, z_t), \tag{12}$$

where $\mathcal{D}$ denotes the allowed set of oriented brick dimensions and p_t specifies the brick position as described in A.4. In text form, each brick unit is serialized as: `h×w (x,y,z)`. For example, as shown in Fig. 2, a brick with footprint 1×4 placed at $(5,6,0)$ is written as: `1x4 (5,6,0)`.

A complete brick structure is therefore represented as: $\mathbf{B} = [Bk_1, Bk_2, \ldots, Bk_T]$, where each generation step predicts one brick token conditioned on the point cloud tokens and all previously generated brick tokens. This representation naturally matches the sequential construction process: the model does not output a dense voxel grid, but instead generates an ordered list of brick components.

A.4 BRICK LIBRARY

All bricks are 1 unit tall and are selected from a library of 8 brick types, yielding 14 oriented variants: 1×1, 1×2, 2×1, 1×4, 4×1, 1×6, 6×1, 1×8, 8×1, 2×2, 2×4, 4×2, 2×6, and 6×2.

A.5 DATA INSTANCE

Each point cloud sequence is inserted into a structured instruction prompt:

```
Create a LEGO model of the input 3D point cloud.
Format your response as a list of bricks: <brick
dimensions> <brick position>, where the brick position is
(x,y,z).
Allowed brick dimensions are 2x4, 4x2, 2x6, 6x2, 1x2, 2x1,
1x4, 4x1, 1x6, 6x1, 1x8, 8x1, 1x1, 2x2.
All bricks are 1 unit tall.
### Input Point Cloud:
(x1,y1,z1), (x2,y2,z2), ..., (xn,yn,zn)
```

With estimated output:

```
### Bricks:
h1×w1(x1,y1,z1), h2×w2(x2,y2,z2), ..., hT×wT(xT,yT,zT)
```

A.6 TRAINING FORMATS

SFT format. For supervised fine-tuning, each PointCloud2Brick sample is formatted as a multi-turn chat transcript compatible with the Qwen chat template. The system message is fixed as "`You are a helpful assistant.`"; the user message contains the structured point-cloud prompt described in Section A.2; and the assistant message contains the ground-truth brick sequence, with one brick per line in `h×w (x,y,z)` format. This stage teaches the model to translate voxel-level shape tokens into valid brick-token sequences.

GRPO format. For reinforcement learning, the prompt contains only the system and user messages, while the assistant response is generated by the policy. Each sample is additionally augmented with a `target_voxels` field, which stores the ground-truth binary occupancy grid as a base64-compressed NumPy array. During training, this target voxel grid is passed to the reward functions to compute shape conformity and other voxel-level objectives. Thus, the same point-cloud and brick-token representation supports both supervised shape reconstruction and reward-based physical-stability optimization.

A.7 DATASET STATISTICS

Table 5 summarizes the PointCloud2Brick splits used in our experiments. All splits are converted from StableText2Brick using the point-cloud and brick-tokenization pipeline described above.

Table 5: Statistics of PointCloud2Brick. Each sample contains a point-cloud-token input paired with an autoregressive brick-token output sequence.

Split	**Train**	**Test**
Total unique samples	42,604	4,785
ShapeNet categories	21	
Unique 3D objects	$\sim$28,000	
Voxel world dimension	$20 \times 20 \times 20$	
Oriented brick variants	14	

B TRAINING DETAILS

SFT configuration. Table 6 lists the supervised fine-tuning configuration. SFT is run as full-model fine-tuning with the TRL SFT entry point, using the point-cloud chat-format dataset and the ground-truth brick sequence as the assistant response.

Table 6: Supervised fine-tuning configuration used to train the point-cloud-conditioned brick generator.

Hyperparameter	Value
Base model	`Qwen/Qwen3.5-2B`
Maximum sequence length	8192
Number of processes	4
Per-device train batch size	2
Per-device eval batch size	2
Gradient accumulation steps	4
Learning rate	2×10^{-4}
Learning-rate scheduler	cosine
Warmup steps	100
Training epochs	3
Precision	bfloat16
Dataloader workers	4
Distributed setting	DDP with unused-parameter detection

GRPO configuration. Table 7 lists the reinforcement-learning configuration used after SFT. GRPO starts from the SFT checkpoint, samples multiple brick completions for each point-cloud prompt, and updates the full model using TRL's GRPO trainer with vLLM colocated generation and DeepSpeed ZeRO-2.

Table 7: GRPO configuration used for physical-stability-oriented reinforcement learning after SFT.

Hyperparameter	Value
Initialization	SFT checkpoint
Trainer	TRL `GRPOTrainer`
Generation backend	vLLM colocate mode
vLLM GPU memory utilization	0.4
Number of generations per prompt	8
Maximum completion length	8192
Sampling temperature	0.7
Per-device train batch size	2
Gradient accumulation steps	4
Training epochs	1
Learning rate	5×10^{-6}
Learning-rate scheduler	cosine
Warmup steps	100
Maximum gradient norm	1.0
Precision	bfloat16
Gradient checkpointing	enabled, `use_reentrant=False`
Distributed optimization	DeepSpeed ZeRO-2
Dataloader workers	4
Dataloader pin memory	enabled

C EVALUATION DETAILS

All the models are evaluated on a Nvidia L40S GPU.

Evaluation data and decoding. We evaluate all systems on the held-out PointCloud2Brick test split for the reported experiments. Each prompt contains a serialized point cloud in the fixed $20 \times 20 \times 20$

voxel world and requests a list of bricks in the form `hxw (x,y,z)`. For causal language models, we apply the Qwen chat template and decode up to 8192 new tokens. Evaluation uses deterministic decoding for comparability across models.

Parsing and validity. Generated text is parsed line-by-line into a `BrickStructure`. A sample is counted as successfully parsed only if the output contains at least one valid brick line and the complete set of lines can be converted into legal brick objects. Collision is computed by rasterizing each brick into the voxel grid and counting voxels with occupancy greater than one. Out-of-bounds bricks and floating components are tracked separately so that parse success, geometric validity, and physical plausibility can be distinguished.

Evaluation metric definitions. Let N be the number of evaluated prompts, $\mathcal{P}$ be the subset of samples whose generations are successfully parsed into non-empty brick structures, $\mathcal{C} \subseteq \mathcal{P}$ be the parsed samples with no voxel collisions, and $\mathcal{B} \subseteq \mathcal{P}$ be the parsed samples whose bricks are all inside the 20^3 world. The metrics used in Tables 1 and 2 are:

- **Coll.-Free Rate**: the fraction of generated structures without voxel collisions, reported over all evaluated prompts as $|\mathcal{C}|/N$. A collision occurs when two or more bricks occupy the same voxel.
- **Mean Coll. Voxels**: the average number of colliding voxels per parsed generation, where a colliding voxel has occupancy greater than one. Lower is better.
- **Voxel IoU**: voxel intersection-over-union between the generated occupancy grid V_{gen} and the target occupancy grid V_{tar}, $\text{IoU} = |V_{\text{gen}} \cap V_{\text{tar}}|/|V_{\text{gen}} \cup V_{\text{tar}}|$, averaged over parsed generations.
- **Mean Stability**: the average simulator-based per-brick stability score given by Gurobi. For a valid structure with brick-level scores $\{s_i\}$, we first compute the structure mean $\frac{1}{M}\sum_{i=1}^{M} s_i$, then average across evaluated valid structures. Higher is better.
- **Min Stability**: the weakest-brick stability score $\min_i s_i$ for each valid structure, averaged across valid structures. This measures whether any brick is close to failure.
- **Conn. Ratio**: the fraction of occupied voxels belonging to the main connected, supported component, computed as $1 - |D|/\max(|O|, 1)$ where O is the occupied voxel set and D is the subset disconnected from the grounded component.
- **Connected Rate**: the fraction of parsed generations classified as connected by the connectivity check.
- **Interlock. Score**: the fraction of non-ground-layer bricks that bridge at least two distinct lower-layer support bricks. This measures local mechanical interlocking.
- **Seam Cov.**: the seam-coverage score, measuring how often seams between bricks in a lower layer are covered by bricks in the layer above. Higher values indicate more staggered, buildable layouts.
- **In-Bounds Rate**: the fraction of parsed structures whose bricks all lie inside the fixed $20 \times 20 \times 20$ voxel world, i.e., $|\mathcal{B}|/|\mathcal{P}|$.
- **Mean Bricks**: the average number of generated bricks per parsed structure.
- **Avg. Time**: average wall-clock inference time per sample in seconds, including any decoding-time rollback or rejection-sampling procedure when that method uses one.

D Training Dynamics

Fig. 6 provides the full training dynamics of `STABLE` and its reward-ablation variants. Across all GRPO-trained models, the total reward increases during training and the entropy decreases, indicating that the policy gradually moves from exploratory generation to more stable construction behavior. The collision reward also improves rapidly in the early stage, showing that spatial feasibility is one of the first physical constraints learned by the policy.

The full `STABLE` model, shown separately in Fig. 7, exhibits the most balanced optimization behavior. The shape reward remains high, while the connectivity and interlocking rewards steadily increase.

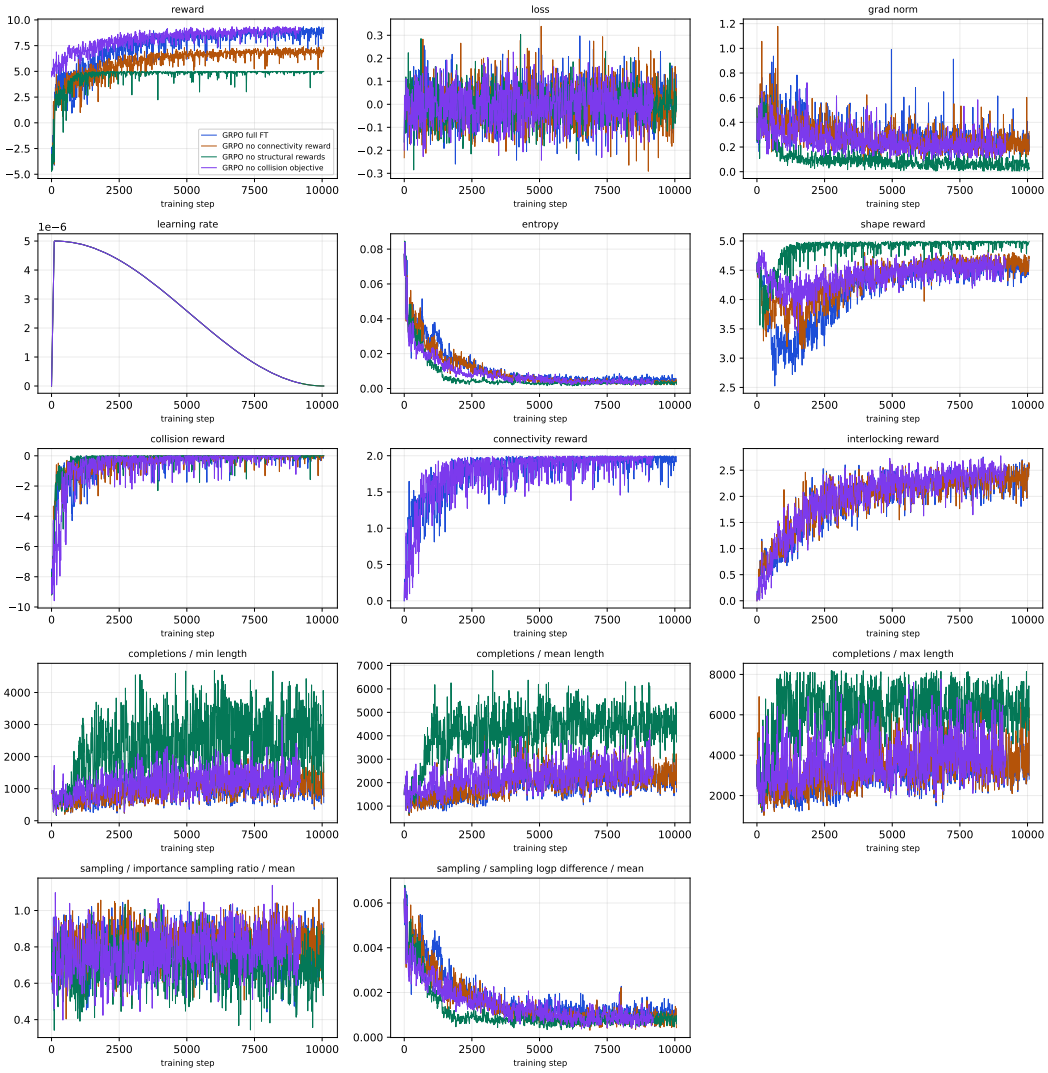


Figure 6: Full GRPO training dynamics of STABLE and reward-ablation variants.

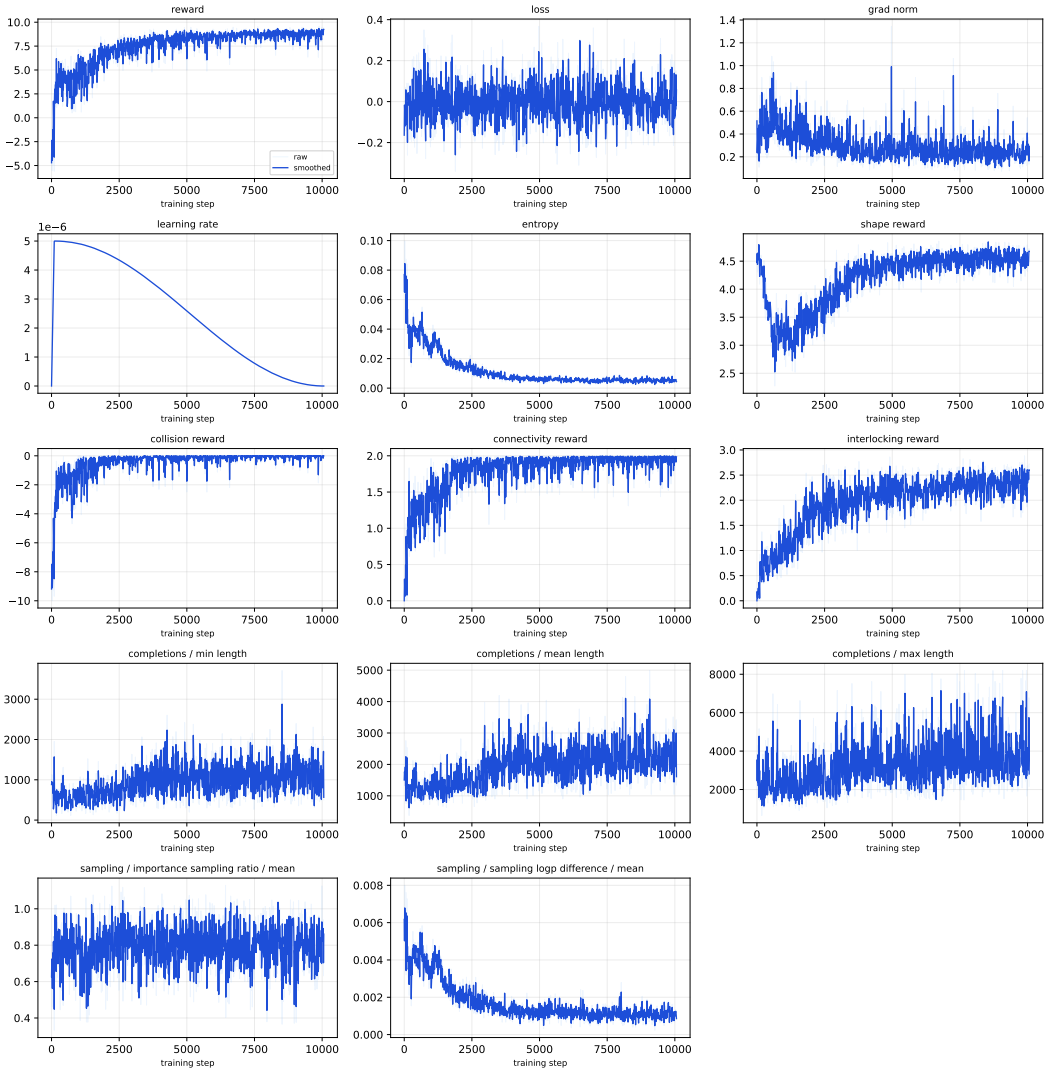


Figure 7: Training dynamics of the full STABLE model.

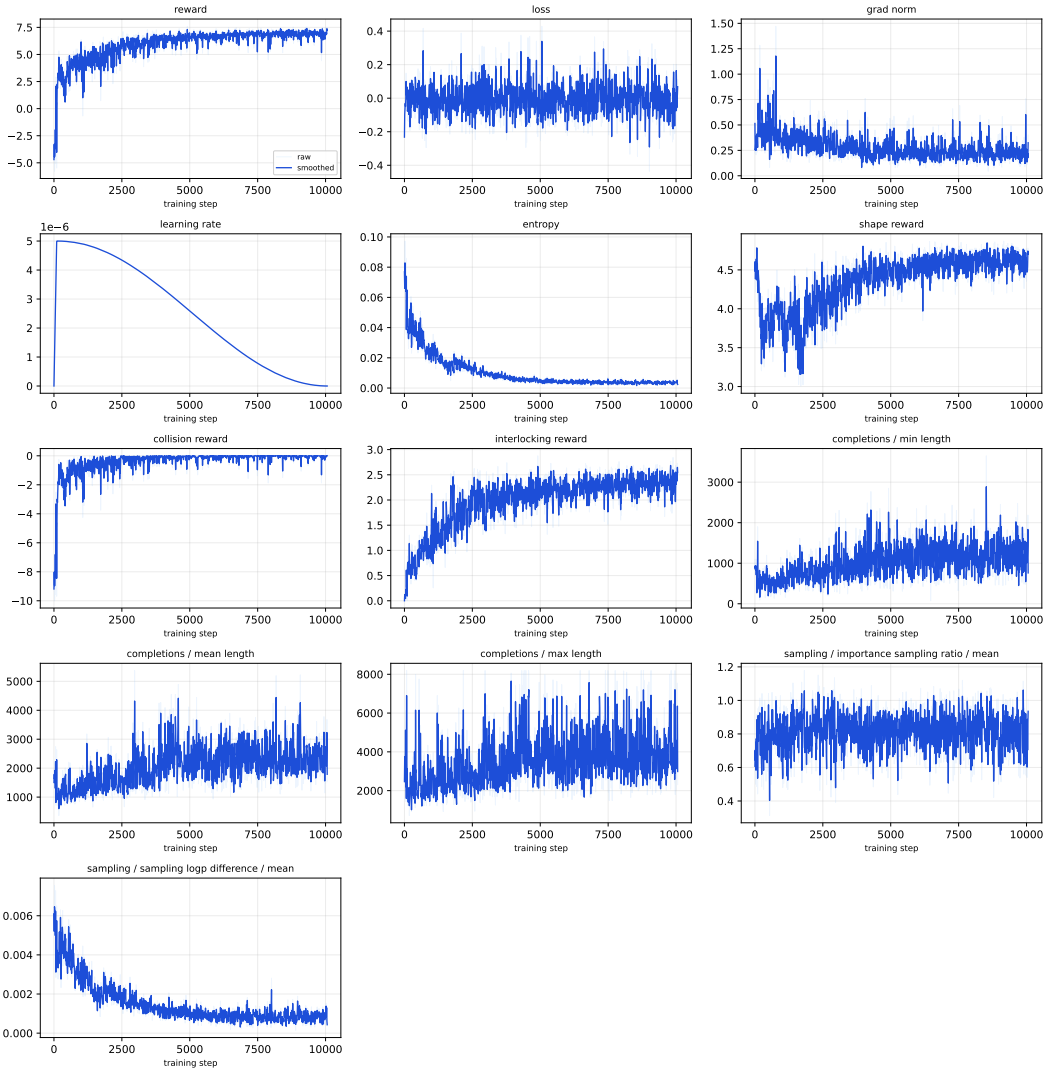


Figure 8: Training dynamics of STABLE without the connectivity reward.

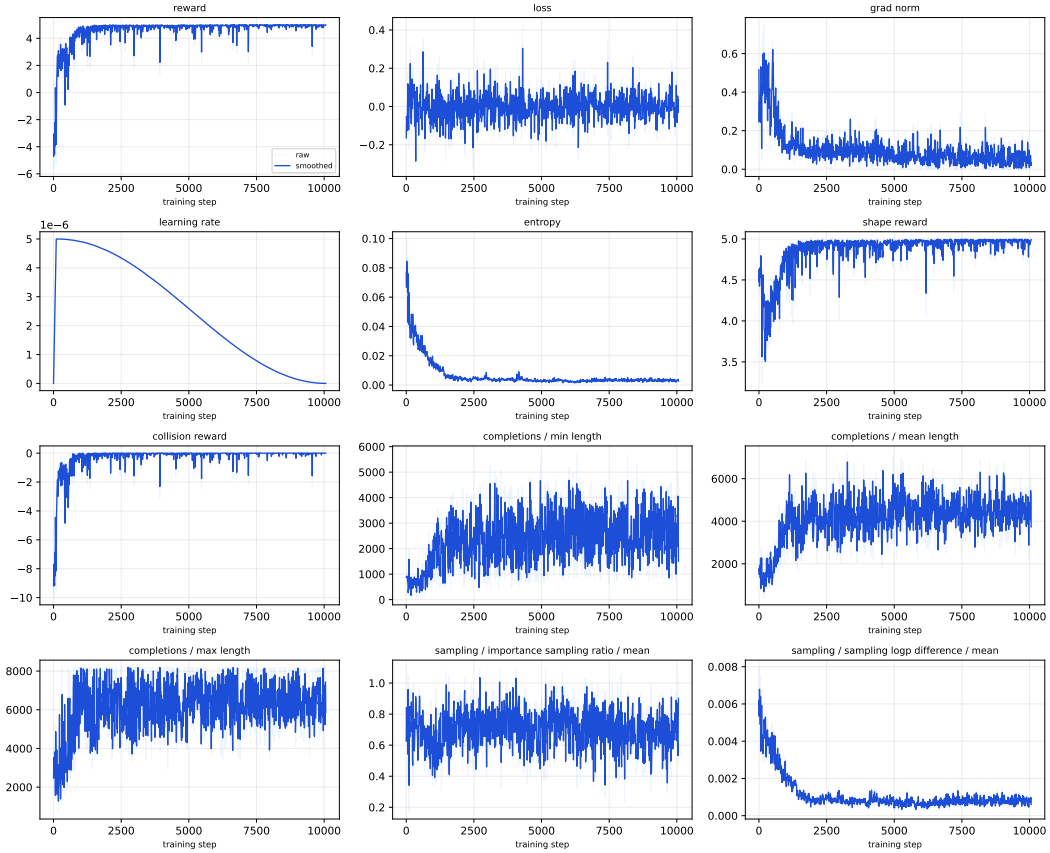


Figure 9: Training dynamics of STABLE without structural rewards.

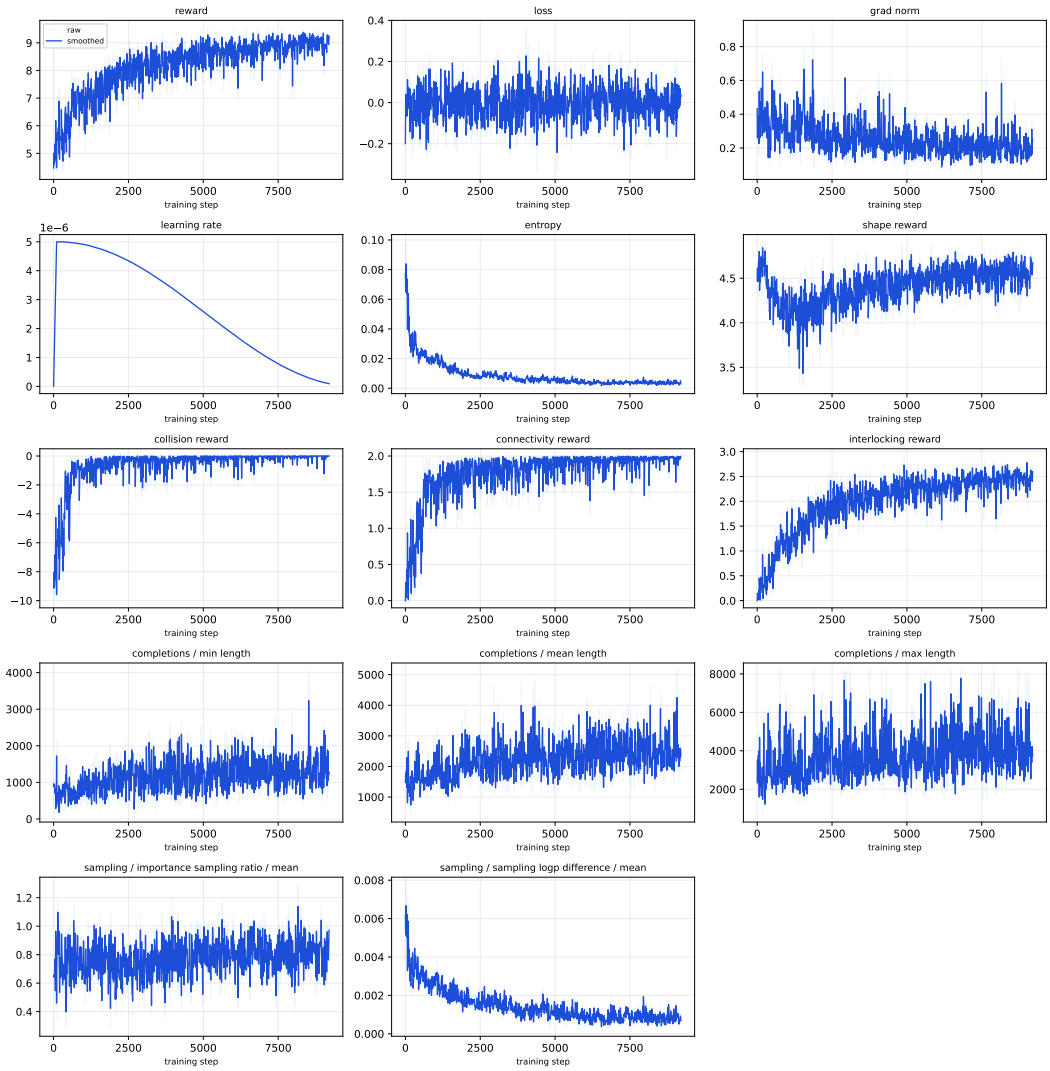


Figure 10: Training dynamics of STABLE without the collision objective.

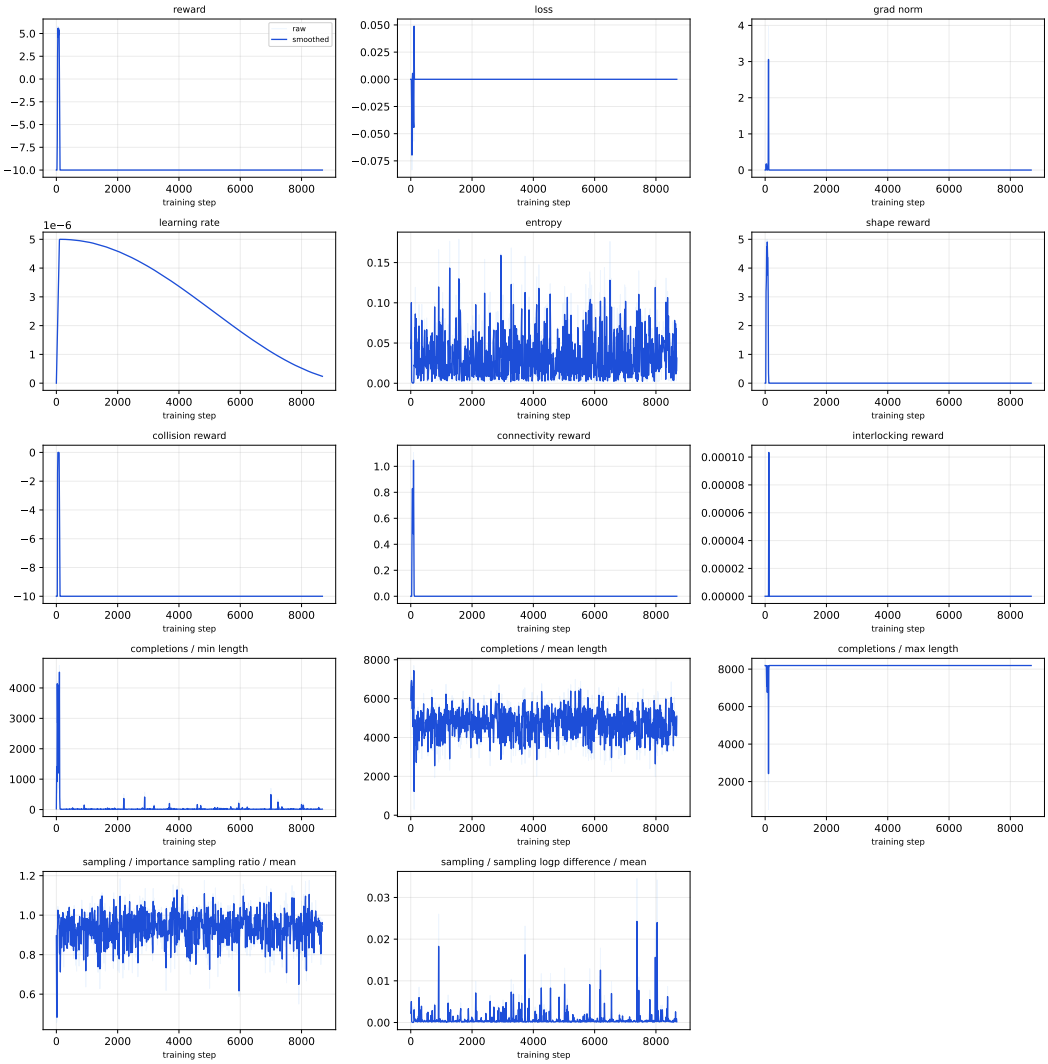


Figure 11: Training dynamics of STABLE−ZERO.

This confirms that STABLE does not obtain stability by sacrificing geometric fidelity; instead, the policy learns brick placements that are simultaneously shape-consistent and structurally coherent.

The ablation dynamics further explain the quantitative results in Table 3. Without the connectivity reward, the model still improves collision and interlocking rewards, but lacks an explicit signal for global assembly coherence. Without both structural rewards, the model mainly optimizes shape and collision, which leads to high IoU but weak connectivity and interlocking. Without the collision objective, the model can still improve shape, connectivity, and interlocking rewards, but cannot reliably eliminate voxel overlaps. These trends support the role of each reward term in constraining a different failure mode of brick generation.

As shown in Fig 11, the training of STABLE-ZERO totally fails because the base model can not produce parseable brick sequence. This means that the RL-only training can not directly induce the brick construction ability due to the sparsity of the reward.

BROADER IMPACT

This work studies how physical constraints can be internalized by generative models instead of being enforced only through test-time correction. The proposed idea may benefit applications in computer-aided design, educational construction tools, robotic assembly planning, and physically grounded 3D content generation. By reducing dependence on expensive inference-time simulation, STABLE may also make stable structure generation more efficient and accessible. At the same time, generated physical designs should not be directly used in safety-critical engineering or real-world load-bearing construction without expert verification. Our method is intended for research and creative assembly scenarios, and responsible deployment should include external validation when physical safety matters.

E EXTENDED RELATED WORK

Physics-aware LEGO stability analysis. A closely related line of work studies how to evaluate the physical stability of LEGO-like block assemblies through explicit mechanics models. These works show that buildability and stability have long been central concerns in computational brick design, but they rely on evolutionary search rather than learning a reusable generative policy. More recently, StableLego (Liu et al., 2024) formulates block-stacking stability analysis as an optimization problem over force-balancing equations and provides a large-scale dataset with stability inference for LEGO layouts. BrickSim (Wen et al., 2026) further develops a real-time physics-based simulator for interlocking brick assemblies, modeling snap-fit mechanics and structural collapse. These methods provide accurate physics-based analysis and could serve as alternative reward or evaluation oracles. In contrast, STABLE does not rely on simulator-based rewards during training. We instead use deterministic construction-aware rewards to expose key stability-related properties to the autoregressive policy, while evaluating final structures with both geometric and structural metrics.

Optimization and evolutionary LEGO design. Classical LEGO construction methods often formulate brick layout generation as a search or optimization problem. Legolization (Luo et al., 2015) converts 3D models into LEGO designs by jointly considering shape approximation, color, workload, and physical stability through iterative refinement. Other works study evolutionary or genetic optimization for LEGO assemblies. Peysakhov and Regli (Peysakhov & Regli, 2003) represent LEGO assemblies as labeled graphs and optimize them with genetic algorithms. These approaches explicitly search over candidate assemblies for each input and use handcrafted objectives to improve buildability. STABLE differs in that it learns a parametric autoregressive policy from data and reward feedback, allowing it to generate complete brick sequences directly at inference time without per-instance optimization, rollback, or evolutionary refinement.

Learning-based combinatorial assembly. Learning-based methods have recently explored LEGO-like construction as an interactive decision-making problem. Brick-by-Brick (Chung et al., 2021) formulates combinatorial construction as sequential placement of unit primitives and uses reinforcement learning with action validity prediction to satisfy geometric constraints. AssemblyComplete (Chen & Liu, 2024) studies 3D combinatorial assembly completion, where an agent completes miss-

ing parts of an incomplete LEGO-like structure under collision, stability, and inventory constraints. These methods are conceptually close to STABLE because they also use reinforcement learning for constrained construction. However, they typically focus on interactive completion or environment-level construction policies, often with explicit action masking. STABLE instead studies conditional autoregressive generation across two assembly regimes: globally positioned bricks conditioned on voxelized point clouds in PointCloud2Brick, and part-connection sequences in the graph-backed BrickNet representation. In both cases, the policy emits a complete sequence without inference-time action masking, rollback, or simulator-guided correction.

Assembly sequence planning. Another related direction is assembly sequence planning, which focuses on finding feasible orders for constructing a given object. Graph Transformer based assembly planning (Ma et al., 2023) represents LEGO models as heterogeneous graphs and learns latent rules for predicting feasible and human-like assembly sequences. PointCloud2Brick and BrickNet allow us to study both sides of this representation choice: the former generates globally positioned brick tokens from voxelized geometry, whereas the latter serializes explicit part connections into an autoregressive build sequence. Our results therefore do not treat graph structure solely as future work; instead, they show that reward-based post-training transfers across coordinate- and graph-backed assembly representations. A remaining direction is to incorporate learned relational graph encoders directly into the policy rather than expressing both representations through serialized tokens.

Relation to STABLE. Together, these works highlight three recurring principles in brick construction. A generated assembly should be geometrically valid, globally coherent, and locally robust under inter-brick support relations. Classical optimization and simulation methods enforce these principles through explicit search or physics solvers, while interactive RL methods learn policies in constrained environments. STABLE combines the advantages of autoregressive generative modeling and structure-level reward optimization. It keeps the inference procedure simple and rollback-free, but uses training-time rewards to make latent construction principles explicit to the model.

LIMITATIONS

STABLE is evaluated in two discrete assembly regimes: PointCloud2Brick uses a fixed voxel world and a compact brick library, while BrickNet contains a substantially larger part vocabulary and graph-structured connection semantics. Although this demonstrates transfer across different representations and component scales, both benchmarks remain LEGO-style discrete construction domains and do not cover continuous geometry or arbitrary mechanical parts. Moreover, the rewards are representation-dependent proxies for collision, connectivity, interlocking, and shape conformity; they do not model all real-world mechanics, such as friction, deformation, material strength, or external forces. The method also reduces inference cost by shifting additional computation to RL training. Extending these rewards to broader physical assembly domains and richer mechanics remains future work.

LLM USAGE

LLM is used for polish the grammar and word selection of the paper. The idea, experiments and writing are completely done by the authors.