MASKED TOPOLOGY MODELING FOR SELF-SUPERVISED LEARNING ON PARAMETRIC CAD

Anonymous authors
Paper under double-blind review

ABSTRACT

Computer aided design (CAD) is ubiquitous: virtually any modern object was designed using editable CAD tools. However, with the shortage of available CAD datasets in its native editable and parametric format, boundary representation (B-Rep), it is ever more important to develop data-efficient methods for this domain.

We present a new self-supervised pretraining task *Masked Topology Modeling* (MTM) that leverages the face-adjacency graph, a graph whose edge attributes the encoder can be asked to predict. MTM masks a fraction of edges and trains a small head to predict each masked edge's convexity and curve type from the encoder's post-message-passing face features.

We combine MTM with a MoCo-style momentum-queue contrastive learning over B-rep-aware augmentations and pretrain on the ABC dataset and our new procedurally generated dataset to show strong performance on a number of benchmarks. We further demonstrate MTM's effectiveness across B-rep architectures.

1 INTRODUCTION

Computer-Aided Design (CAD) is a foundational technology that has revolutionized manufacturing (Groover & Zimmers, 1983), modern engineering, construction (Heesom & Mahdjoubi, 2004), architecture (Szalapaj, 2013), healthcare (Fujita et al., 2008), and product development (Liu, 2021). By replacing time-consuming and error-prone manual drafting, its ability to precisely build complex 3D models allows designers to visualize, simulate, and test products before being sent for manufacturing or construction.

Boundary Representation (B-rep) is the standard format used by modern CAD systems to define and edit solid 3D models. Unlike mesh formats which approximate a shape using a collection of flat polygons (Bernardini et al., 1999), B-rep defines an object by its exact, mathematical boundaries. This "watertight" model is composed of two core components: *geometry*, which provides the precise mathematical description (such as planes, cylinders, or complex NURBS curves), and *topology*, which is the collection of faces, edges, and vertices and the logical relationships and constraints that connect them. This dual structure creates an unambiguous, data-rich solid that is mathematically precise (Faux & Pratt, 1979).

Despite being so popular and widespread, there is a lack of publicly available B-rep data. In practically every industry, CAD data is considered critical intellectual property necessary for competitive advantage (Stjepandić et al., 2015). Sharing such files can expose companies to significant additional risks, including physical sabotage, supply chain poisoning (Karahan et al., 2025), and data integrity issues in distributed manufacturing (Li et al., 2025). Converting to non-parametric formats such as 3D meshes (e.g. STL) destroys the design intent and feature tree history, making it very difficult to reverse engineer the original design (Lecallard et al., 2021). In practice, CAD designers often send manufacturers only the mesh as a safeguard from theft or modification of the core design, since the mesh only provides an approximation of the objects (Kantaros, 2024).

Recent AI systems have shown profound success across various industries. AI typically depends on large, high-quality datasets, but CAD lacks comparable access to such data. Industrial CAD data is scarce, expensive to label, and often proprietary, making it ever more critical to study data-efficient methods for CAD. In other domains, self-supervised learning (SSL) (Chen et al., 2020; He et al., 2022; Bahri et al., 2022) has substantially improved label efficiency as shown by contrastive

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.

and masked-autoencoding approaches. It turns out that B-reps provide a particularly rich structure for SSL when viewed as topological graphs because they expose intrinsic geometric and topological features arising from the signals—face adjacency, edge topology, curve and surface types, continuity, and geometric constraints that can be learned without manual labels. Therefore, although B-rep data is limited and labeled B-rep data is even more limited, each B-rep nonetheless contains rich internal structure that can generate many high quality self-supervised training signals. Therefore, SSL is a natural way to improve both data and label efficiency.

Our contributions are as follows:

1. **Masked Topology Modeling (MTM).** We introduce a self-supervised objective that removes selected face-adjacency graph edges and predicts their convexity and curve type from the surrounding face representations. MTM turns geometric relationships already available in B-reps into training signals, requiring no manual annotation.
2. **Theoretical analysis.** We characterize the best achievable prediction loss through an identifiability result. Separate separability results for convexity and curve type show that individual face shapes and adjacency alone are not always sufficient, motivating representations that preserve additional geometric information.
3. **Synthetic B-rep pretraining data.** We adapt a procedural sketch-extrude generator (Rukhovich et al., 2025) to construct a large corpus of solids with automatically derived edge labels. We show that using our synthetic dataset in pretraining results in stronger downstream performance across the board.
4. **Strong empirical results and ablations.** We shows strong results on Fusion 360, SolidLetters, MFInstSeg, and CADSynth. We further show that MTM improves representations on not only our architecture but also popular ones including UV-Net (Jayaraman et al., 2021) and AAGNet (Wu et al., 2024), showing that it is versatile across B-rep neural networks.

2 Related Work

B-rep encoders. Research on B-rep has received a lot of recent attention due to its ability to effectively represent 3D models (Guo et al., 2022; Jayaraman et al., 2022; Willis et al., 2022; Cherenkova et al., 2024; Xu et al., 2025). We begin by highlighting the early work in B-rep representation learning. UV-Net (Jayaraman et al., 2021) and BRepNet (Lambourne et al., 2021) are two foundational B-rep encoders. UV-Net represents each face as a UV-grid processed by a CNN and the face-graph by graph convolutions. BRepNet convolves directly over oriented coedges of the data structure. Both are supervised architectures. BRepNet conditions message passing on it, whereas we hide part of it and make it a prediction target. We use UV-grid encoders with GATv2 and Transformers, trained with self-supervision. CADOps-Net (Dupont et al., 2022) learns a representation via joint learning of CAD operation types and steps. Hierarchical CADNet (Colligan et al., 2022) proposes a hierarchical B-Rep graph representation which encodes information about the surface geometry and face topology trained in a supervised way. AAGNet (Wu et al., 2024) is a multi-task graph network over face-adjacency graphs, and BRepMFR (Zhang et al., 2024) uses domain adaptation.

Self-supervised learning for B-reps. The UV-Net paper also proposes a self-supervised contrastive variant, which pretrains the encoder by pulling together augmented views of the same solid. BRepBERT (Lou et al., 2023) adapts masked language modeling by masking and reconstructing per-node attributes. We differ in the masking unit and target: we mask edges and predict relational, kernel-computed properties (convexity and curve type) rather than node features. GC-CAD (Quan et al., 2024) and ContrastCAD (Jung et al., 2024) learn through contrastive alignment of augmented B-rep views. MTM adds local topological reasoning that these objectives do not explicitly supervise. Most recently, BRepMAE (Yao et al., 2026) takes a generative masked-autoencoding route, reconstructing masked geometry. Ours is discriminative: classifying a small set of topological relations at hidden edges rather than regenerating a region. Masking and predicting edge attributes from endpoint embeddings has been studied for GNNs (Hu et al., 2019). HierMAE (Li et al., 2026) reconstructs face and edge geometry and attributes, including convexity and curve type. MTM instead removes selected connections from message passing and predicts these attributes from the two incident faces' contextual features using small classification heads.

3 METHOD

3.1 VIEWING A B-REP AS A LABELED TOPOLOGICAL GRAPH

A B-rep explicitly records how a solid's surfaces connect: which edges bound each face and which vertices terminate each edge. These connections define its *topology*. A B-rep's faces and edges refer to underlying geometric surfaces and curves. MTM operates on a specific induced structure from the B-rep, the *face-adjacency graph*.

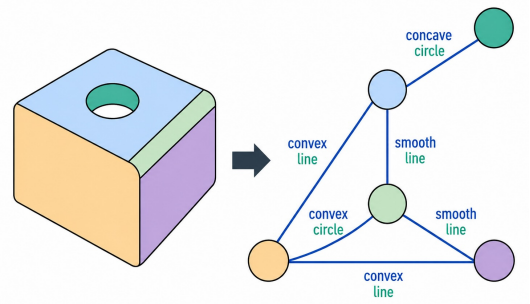

Figure 1: **A B-rep solid as an labeled face-adjacency graph.** Each face of the solid (left) becomes a node (right), and two nodes are joined by a graph edge iff their faces meet along a shared B-rep edge. Every graph edge carries the two labels MTM predicts: a convexity class (*convex*, *concave*, *smooth*, *knife*) and a curve type (*line*, *circle*, *B-spline*, *ellipse*, *other*), both obtained automatically from the B-rep. Because this topological signal lives on the graph's edges, it can be masked and predicted directly. Note that this figure is for illustration purposes only and the graph only uses the visible faces of the solid.

The face-adjacency graph (Ansaldi et al., 1985) represents connections between a solid's faces. Each node represents a face, and graph edges correspond to shared B-rep edges: two faces f_i and f_j are connected if and only if they share a B-rep edge. Following UV-Net (Jayaraman et al., 2021), we represent each face using a UV-grid of sampled points, surface normals, and a trimming mask indicating which samples lie within the face's trimmed boundary. A surface CNN maps this grid to the node's initial feature vector.

We can further automatically label each graph edge with the following two topological attributes: *convexity* $c(e) \in \{\text{concave}, \text{convex}, \text{smooth}, \text{knife}\}$, the dihedral relationship of the two faces across the seam. If the material folds inward, then it's *concave*; and if it's outwards, then it's *convex*. If they meet tangentially, that is at approximately 180 degree angle (the boundary between concave and convex), then it's *smooth*. Finally, the degenerate case where it meets at approximately 0 degree angle, then it's classified as *knife*. The second attribute is *curve type* $t(e) \in \{\text{line}, \text{circle}, \text{B-spline}, \text{ellipse}, \text{other}\}$, the geometry of the seam curve itself. Both are obtained automatically during preprocessing (oriented dihedral geometry for convexity, the parametric curve tag for type), so these labels are *free*; that is, no human annotates them. Details are in Appendix F. This is the reason topology can serve as a self-supervised signal. The graph edges come pre-labeled by construction. Labeling the edges of the face-adjacency graph is an old idea used for process planning (Joshi & Chang, 1988). However, it was a binary concave/convex label. For MTM, we use the convexity classes together with curve type as self-supervised targets.

3.2 CONTRASTIVE LEARNING

Our encoder follows UV-Net (Jayaraman et al., 2021): each face is sampled on a fixed UV parametric grid and embedded by a surface CNN, each edge by a curve CNN, and the resulting per-face features are refined by message passing over the face-adjacency graph to yield embeddings $h_i \in \mathbb{R}^{384}$. We use GATv2 message passing followed by global transformer layers, trained with our self-supervised objective. Full details are in Appendix A.

We adopt Momentum Contrast (MoCo) (He et al., 2020), which casts the objective as a dictionary-lookup task. Each input is rendered into two augmented views. View A is passed through the online (query) encoder to produce a query, while View B is passed through a momentum key encoder to produce a key. The query is trained to match its corresponding key (the positive) against a large set of negatives drawn from a large queue of negatives (here $16{,}384$ negatives). This queue decouples the number of negatives from the batch size and is optimized with an InfoNCE loss (Oord et al., 2018). Full details are in Appendix B.

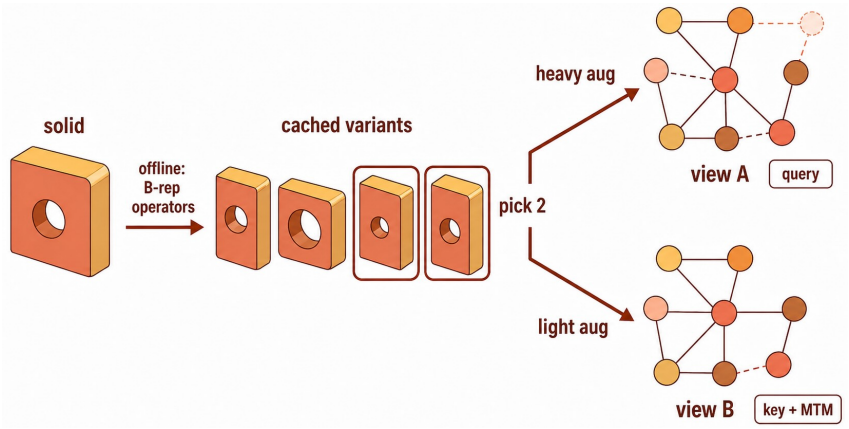

Figure 2: **Two-level positive-pair construction.** *Offline*: each source solid is perturbed by B-rep operators into several cached variants that differ in geometry. *Online*: a training pair draws two of these variants and augments them asymmetrically at the graph level. One receives heavy augmentation to form view A (the contrastive query), the other light augmentation to form view B (the contrastive key, and the graph on which MTM operate). The pair therefore differs in both geometry (distinct perturbations) and graph structure (heavy vs. light corruption).

Positive pairs are formed at two levels. *Offline*: for each source solid we precompute several perturbed variants by applying a randomly chosen B-rep operator through the OpenCASCADE kernel, each validated for a watertight solid. A training pair draws two different variants of the same source. *Online*: each drawn view further undergoes face drop, edge drop, connected-subgraph sampling, and coordinate jitter. Augmentation is asymmetric: view A (the contrastive query) receives the full heavy augmentation, while view B receives only light augmentation. View B serves both as the contrastive key and as the graph on which the MTM objective operate. Keeping B lightly augmented preserves the context for predicting masked edge attributes. MTM uses a separate online-encoder forward on B with gradients; the momentum forward produces the contrastive key without gradients.

Pretraining data. Our method is pretrained entirely out-of-domain, from two sources. The first is real B-rep solids (in STEP format) from ABC (Koch et al., 2019) ($\sim$1M shapes), the largest public collection of CAD models. We filter out sources without a sufficient number of successfully preprocessed variants. The second is a new procedurally generated dataset: we adapt the CAD-generation procedure of CAD-Recode (Rukhovich et al., 2025), which samples random 2D profiles, places them on random planes, extrudes them into 3D primitives, and combines these into a single solid. We apply B-rep operator augmentations to create positive pairs from both sources (see Appendix G). These operations can change geometry and topology.

3.3 Masked Topology Modeling (MTM)

MTM is a *masked prediction* objective on the face-graph's edge set. The key design choice is that the encoder never sees the masked edges. They are deleted from the graph before message passing so the head must reconstruct each missing edge's semantics from contextual features of its two endpoint faces. The endpoint pair is supplied to the head; MTM predicts its attributes, not whether it is adjacent. Given a shape $S{=}(F, E)$, where E denotes physical B-rep edges:

1. **Mask.** On view B, sample a masked edge set $M \subset E$ at rate $p_{\text{mask}}{=}0.7$ and delete all directed graph entries associated with those physical edges, so the encoder receives only $(F, E \setminus M)$. We record each masked edge's two endpoint faces and its ground-truth $(\text{convexity}, \text{curve-type})$ labels.

2. **Forward.** The encoder ϕ runs message passing over the *masked* graph to produce per-face features h_i. Because the masked edges are absent, graph messages use only a face's *remaining* neighbors; the transformer also aggregates global face context.

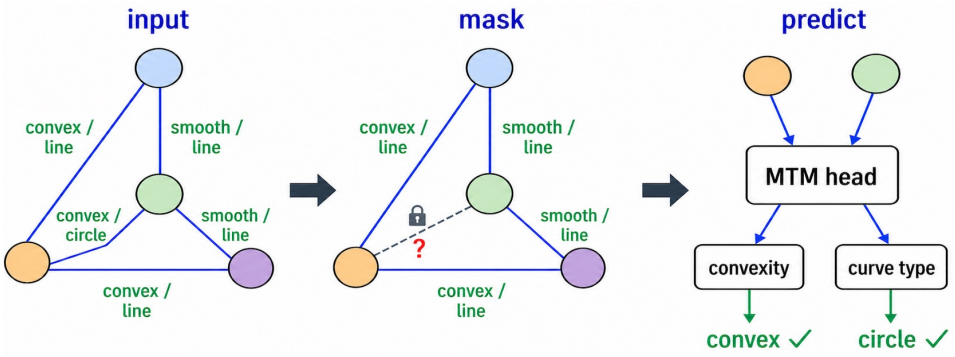

Figure 3: **Masked Topology Modeling (MTM).** Left: each edge of the face-adjacency graph carries a convexity class and a curve type. Middle: we hide one edge, deleting it and its labels from the graph before encoding. Right: from the features of the two endpoint faces, which incorporate the remaining graph and global face context, an MTM head predicts the hidden convexity and curve-type labels.

3. **Predict.** For each masked edge $e{=}(i, j)$, the ordered endpoint feature $[h_i; h_j]$ is passed to two lightweight MLP heads predicting convexity (4 classes) and curve type (5 classes).
4. **Loss.**

$$L_{\text{topo}} = \tfrac{1}{2}(\text{CE}(\text{MLP}_{\text{convexity}}([h_i; h_j]),\, c(e)) + \text{CE}(\text{MLP}_{\text{curve-type}}([h_i; h_j]),\, t(e)))\,, \quad (1)$$

averaged over masked directed entries, weight $\lambda_{\text{topo}}{=}0.5$ and CE denotes cross-entropy loss. Both labels are obtained automatically and no manual annotation is required. Details are in Appendix C.1.

The heads are lightweight ($\approx$0.2M params, discarded after pretraining). They operate on the light-augmentation view's face features. Surviving edges retain their convexity and curve-type embeddings; the masked edges and their labels are withheld from message passing.

3.4 Combined objective

$$L = L_{\text{NCE}} + \lambda_{\text{topo}}\, L_{\text{topo}} \quad (2)$$

The method is trained from random initialization for 100 epochs with $p_{\text{mask}} = 0.7$ and $\lambda_{\text{topo}} = 0.5$.

4 Theory

We present an *identifiability* result and two *separability* results, one for each of the targets (convexity and curve-type). Identifiability bounds the best achievable prediction loss by how much uncertainty about the masked label remains in the encoder's features. The separability results show that individual face shapes and neighbor connections alone are not always sufficient to predict the targets. Thus, the additional geometric features that MTM leverages can play an important role. Proofs are in Appendix D.

Let e be a masked edge shared by faces f_i, f_j. The encoder assigns each face a feature vector, also called an embedding. Write h_i, h_j for the embeddings of these two faces after masking. MTM's prediction head uses (h_i, h_j) to predict the edge's convexity class $c(e)$ and curve type $t(e)$.

Let X_e be the complete input remaining after masking, including the indices i, j identifying the two faces sharing e. Let Y_e be the label to predict, either $c(e)$ or $t(e)$, and let $Z_e = (h_i, h_j)$ be the pair of face embeddings given to the prediction head. The *conditional entropy* $H(Y_e \mid X_e)$ measures how uncertain the label remains when the input is known; $H(Y_e \mid Z_e)$ measures that uncertainty when only the two face embeddings are known.

The following result shows that any prediction head's expected cross-entropy loss is at least the uncertainty about the label remaining in the encoder's features.

Table 1: Fusion 360 Gallery segmentation, full-data and few-shot (8-way support, evaluated on the full test set). Our full-data number is the mean over 3 fine-tuning seeds.

		Full-data		10-shot		20-shot	
Method	Pretrain data	Top-1	mIoU	Top-1	mIoU	Top-1	mIoU
UV-Net	SolidLetters / ABC	92.30	72.40	–	–	–	–
BRepNet	F360 (supervised)	94.30	81.40	–	–	–	–
BRep-BERT	F360	95.14	**82.88**	59.26	19.02	62.30	24.20
From-scratch	None	–	–	61.04	22.23	64.82	27.49
Ours	ABC + synthetic	**95.5**	**82.88**	**82.51**	**56.61**	**86.44**	**61.93**

Theorem 1 (Identifiability)**.** *For either label and a fixed encoder, allow the prediction head to output any probability distribution over labels as a function of the two face embeddings Z_e. The lowest achievable expected cross-entropy loss L^* is*

$$L^* = H(Y_e \mid Z_e) \geq H(Y_e \mid X_e).$$

For separability, we consider encoders given only each face's individual shape and which faces are neighbors. A face's individual shape includes its surface and trimmed boundary, but not its position or orientation within the solid. The neighbor connections, or *face adjacency*, specify which faces meet, but not which parts of their boundaries meet or any edge attributes. The encoder is also not told which side of a face contains material. We assume the encoder is *permutation equivariant*: reordering the faces reorders their embeddings without changing the embedding assigned to each face. We call an encoder satisfying these conditions a *restricted encoder*.

Both separability results concern restricted encoders. MTM receives additional geometric and edge information, so neither result establishes that MTM itself must fail. Rather, it shows that correct prediction in these examples requires MTM to preserve information beyond individual face shapes and neighbor connections in its face embeddings.

The convexity result compares one edge in each of two different solids. Their individual face shapes and neighbor connections match, but the corresponding edges have different convexity labels. We mask the edge separately in each solid and show that the resulting face embeddings are the same.

Theorem 2 (Convexity separability across solids)**.** *There exist two different closed solids with matching individual face shapes and the same face-adjacency graph, but different convexity labels on corresponding edges. After masking the corresponding edge in each solid, any restricted encoder produces identical embeddings for the corresponding pairs of faces. Therefore, a prediction head using these embeddings cannot correctly label both edges.*

For the curve-type result we construct two separate inputs from the same solid, masking a straight edge in one and a circular edge in the other and show that the encoder cannot distinguish the two cases from individual face shapes and neighbor connections alone.

Theorem 3 (Curve-type separability within a solid)**.** *There exists a closed solid containing a straight edge and a circular edge such that, when each is masked in a separate input, any restricted encoder produces identical pairs of face embeddings for the two edges. As such, a prediction head using these embeddings cannot correctly label both curve types.*

5 Experiments

5.1 Results

Fusion 360 Gallery Segmentation. We evaluate on Fusion 360 Gallery Segmentation (Willis et al., 2021) under the fixed public split, in two regimes: full-data fine-tuning and few-shot support. A from-scratch control (identical architecture as ours but fine-tuned from random initialization) measures the contribution of pretraining in the few-shot rows.

On the full-data setting, our method matches BRep-BERT's reported macro mIoU, despite pretraining on ABC and synthetic data and not on F360. In the few-shot regime, pretraining substantially

Table 2: SolidLetters, full-data 26-way accuracy and few-shot 10-way accuracy (10-/20-shot). Our results show the mean ± standard deviation across 5 runs for full data and 20 runs for few-shot (this includes the distinct 10-of-26 letter draws).

Method	Full-data acc	10-shot	20-shot
Point-BERT	93.25	60.29	67.35
Point-MAE	94.16	62.85	68.96
Point-M2AE	96.03	67.34	72.21
UV-Net	97.24	54.31	61.47
BRepNet	97.71	60.24	68.84
BRep-BERT	**98.54**	68.71	75.92
Ours	97.93 ± 0.09	**77.60 ± 2.79**	**85.24 ± 2.55**

Table 3: MFInstSeg per-face segmentation accuracy (%), with mean and standard deviation over 3 runs for the low-label settings and 2 for full-data. Every row is trained and evaluated by us using the official benchmark split. For few-shot, our method and all the baselines used the official validation (9,373 examples) split to choose the checkpoint evaluated at each epoch.

Method	0.1%	0.5%	1%	Full Dataset
GAT	38.68±1.35	54.93±1.48	64.37±2.81	86.84±0.37
GraphSAGE	49.67±0.37	67.19±1.84	84.00±0.38	98.25±0.06
UV-Net	43.15±8.93	77.36±2.84	86.14±1.98	98.89±0.01
AAGNet	58.55±1.03	79.80±2.54	91.50±1.02	99.30±0.01
From-scratch	41.37±2.22	51.18±0.43	72.05±1.15	98.54±0.00
Ours	**84.25±1.03**	**94.12±0.49**	**96.01±0.17**	**99.50±0.01**

Table 4: CADSynth label-ratio accuracy (%). Rows are every method in HierMAE's CADSynth comparison (their Table 1). Our method is over 3 seeds. We use 10,000 additional labeled validation shapes. Baseline numbers are quoted from HierMAE under their reported protocols; our training/validation split differs.

Method	0.1%	0.5%	1%
AAGNet	58.53	87.36	95.58
BRepFormer	59.34	65.60	97.49
Ours	**90.72 ± 0.63**	**97.33 ± 0.20**	**98.38 ± 0.08**
Pretrained on CADSynth (in-distribution)			
BRepMAE	93.63	96.84	98.36
HierMAE	<u>93.82</u>	<u>98.54</u>	<u>98.91</u>

improves over our from-scratch control. Published baseline rows use their reported numbers. Details in Appendix H.

Note that we don't include recent baselines BRepMAE (Yao et al., 2026) or HierMAE (Li et al., 2026) here. BRepMAE does not report downstream F360 segmentation results and HierMAE uses a different data split and defines the few-shot budget as a total number of labeled shapes, instead of shapes per class.

SolidLetters classification. SolidLetters, introduced with UV-Net (Jayaraman et al., 2021), is a synthetic B-rep classification benchmark whose classes are the uppercase English letters. We use it in two regimes: the standard full-data setting and a few-shot setting that samples a subset of letters with limited labeled solids per class. In both settings, we fine-tune the architecture end-to-end.

We compare against the numbers reported in Lou et al. (2023), which includes point cloud methods Point-BERT (Yu et al., 2022), Point-MAE (Pang et al., 2023), and Point-M2AE (Zhang et al., 2022). We see that although our method performs slightly worse on full-data than BRep-BERT, it wins considerably on the few-shot setting. We don't compare HierMAE's results here, as it uses

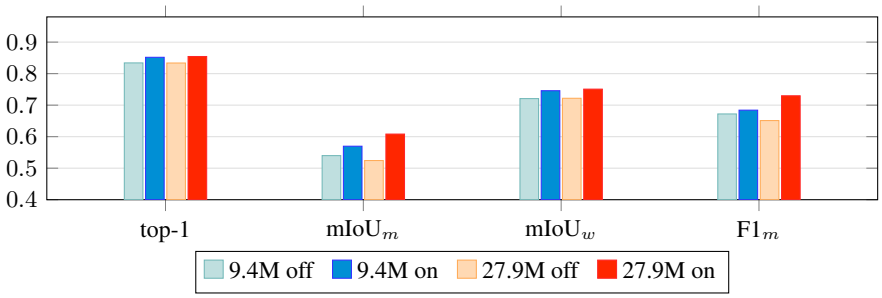

Figure 4: **MTM On vs. Off**: We plot frozen F360 linear probe metrics for both 9.4M and 27.9M models pretrained from scratch with MTM on vs. off. MTM improves all reported metrics.

Table 5: Synthetic-data comparison after fine-tuning (%) at matched pretraining interations. Full data is run on a single seed and few-shot and MFInstSeg are each across 3 seeds.

	F360 full		F360 10-shot		F360 20-shot		MFInstSeg accuracy		
Data	Top-1	mIoU	Top-1	mIoU	Top-1	mIoU	0.1%	0.5%	1%
ABC	93.11	76.17	71.63	43.62	75.20	48.71	76.06	81.08	88.87
ABC + synth	**93.76**	**78.78**	**74.33**	**44.06**	**79.09**	**50.87**	**76.20**	**82.93**	**89.90**

Table 6: Effect of MTM vs masking only for the 9.4M model. Full data F360 is run on a single seed and few-shot F360 and MFInstSeg are each across 3 seeds.

	F360 macro mIoU			MFInstSeg accuracy		
Variant	Full	10-shot	20-shot	0.1%	0.5%	1%
MTM off	77.83	38.75	47.10	75.71±1.21	82.44±0.23	89.35±0.43
Mask only	77.79	37.77	47.30	74.41±1.05	81.58±0.23	88.92±0.51
MTM on	**78.78**	**42.24**	**51.22**	**76.20±1.56**	**82.93±0.23**	**89.90±0.39**

a different split and evaluates few-shot classification over all letters; ours uses a sampled subset of classes. Details in Appendix H.

MFInstSeg. MFInstSeg (Wu et al., 2024) is a B-rep dataset where every face is labeled with a machining-feature class, and the task is per-face segmentation scored by face-level accuracy. Table 3 compares models under the official split, across low-label and full-data settings. The baselines are trained from scratch: GAT (Veličković et al., 2017), GraphSAGE (Hamilton et al., 2017), UV-Net, the task-specific supervised AAGNet (Wu et al., 2024), and our own architecture at random initialization. Our method turns out to be the strongest across the board. Our margin is highest as the label budget shrinks, which suggests that our SSL method is learning useful representations and is label efficient. We don't compare BRepMAE and HierMAE here because their reported results use a larger training split than ours. Our table contains models evaluated under the official split. Details in Appendix H.

CADSynth. CADSynth (Zhang et al., 2024) is a synthetic B-rep benchmark with dense per-face machining-feature annotations. Table 4 compares our model with the published results in Hier-MAE's CADSynth comparison. Our model exceeds the reported AAGNet and BRepFormer results, but falls below BRepMAE and HierMAE at the smallest label budget. At larger budgets, it exceeds BRepMAE but trails HierMAE. However, our method doesn't use CADSynth data in the pretraining set, while HierMAE and BRepMAE do. Details in Appendix H.

5.2 Ablations

Effect of MTM. We evaluate the effect of MTM on our method as well as a smaller version of it, pretraining each from random initialization with MTM off or on (Figure 4). Adding MTM improves

Table 7: Effect of MTM on UV-Net architecture pretrained on ABC. Averaged across 3 seeds.

	F360			MFInstSeg					
	Frozen	10-shot		0.1%		0.5%		1%	
Objective	mIoU	Acc.	mIoU	Acc.	mIoU	Acc.	mIoU	Acc.	mIoU
Contrastive only	21.53	49.58	15.05	54.56	18.55	61.66	22.88	75.38	36.58
Contrastive + MTM	32.05	61.67	22.47	59.36	21.63	68.60	27.11	79.58	41.78
MTM only	**37.66**	**68.59**	**25.81**	**64.32**	**27.36**	**69.58**	**30.50**	**80.10**	**44.31**

Table 8: Effect of MTM on native AAGNet with in-domain pretraining on unlabeled MFInstSeg training shapes. Averaged across 3 seeds.

	MFInstSeg 0.1%		MFInstSeg 0.5%		MFInstSeg 1%	
Objective	Acc.	mIoU	Acc.	mIoU	Acc.	mIoU
Contrastive only	58.79	25.71	80.88	58.91	86.86	71.27
Contrastive + MTM	66.20	36.90	85.26	66.89	92.14	78.76
MTM only	**70.17**	**42.66**	**87.85**	**70.87**	**92.94**	**80.54**

F360 frozen linear probe macro mIoU at both scales. This comparison measures the combination of edge masking and prediction.

Masking and prediction. We test whether MTM's gains come from predicting edge attributes or simply masking edges. Starting from a shared contrastive checkpoint, we compare MTM off, masking alone, and masking with prediction on the smaller architecture (Table 6). The masking without prediction doesn't help and in many cases slightly hurts performance. Adding the prediction loss improves all reported downstream results. We have not compared MTM against predicting the same attributes while retaining the masked edges' connections in the encoder. Details in Appendix I.1.

MTM on UV-Net and AAGNet. We apply MTM to UV-Net (Jayaraman et al., 2021) and native AAGNet (Wu et al., 2024) to test whether the benefit extends to different architectures. For UV-Net, we pretrain from random initialization on ABC using contrastive learning alone, contrastive learning with masking but no prediction loss, contrastive learning with MTM, or MTM alone. Table 7 shows that MTM alone gives the highest scores. Details in Appendix I.2. For AAGNet, we pretrain on unlabeled MFInstSeg training shapes. Table 8 shows that adding MTM to contrastive learning improves performance at every tested label budget, while MTM alone performs best among these variants. Full details in Appendix I.3.

This supports MTM's usefulness beyond our architecture. In both UV-Net and AAGNet, MTM alone outperforms our recipe of combining contrastive loss with MTM. On our architectures, results are mixed, with the combined objective achieving higher scores in most of the reported settings. See Appendix I.4.

Synthetic pretraining data. Here we show that our synthetic dataset for pre-training substantially improves downstream performance. Table 5 compares pretraining with ABC dataset only with ABC plus synthetic data. We start from the same ABC checkpoint and match the number of pretraining iterations so the effect isn't because of more training steps. The ABC plus synthetic model achieves higher performance across the board.

6 Conclusion

We introduced Masked Topology Modeling, an SSL objective that predicts the convexity and curve type of masked B-rep edges from geometry. The method comes with identifiability and separability results. We combined MTM with MoCo momentum-queue contrastive learning and pretrain on ABC and our procedurally generated data, and demonstrated strong empirical performance across four downstream benchmark datasets. We also showed that MTM is versatile as it helps popular architectures such as UV-Net and AAGNet.

AI use statement. Claude Code (Fable 5.1) and Codex (GPT-6 Astra) assisted with implementation of the method, baselines, evaluation code, synthetic CAD data generation and perturbed benchmark variants, experimental design and development of mathematical claims and proofs, and manuscript drafting. Code and mathematical claims underwent iterative review, including adversarial tests and counterexamples, with identified errors corrected and re-tested. The authors reviewed the AI-assisted work and heavily edited the manuscript manually and take full responsibility of the manuscript.

Reproducibility statement. Appendix F describes preprocessing and the construction of MTM targets. Appendix A specifies the model architecture, and Appendix H details downstream data splits, training settings, and checkpoint selection. The ablation protocols are described in Appendix I, including the scope of repeated runs and differences between experimental settings.

REFERENCES

Silvia Ansaldi, Leila De Floriani, and Bianca Falcidieno. Geometric modeling of solid objects by using a face adjacency graph representation. *ACM SIGGRAPH Computer Graphics*, 19(3):131–139, 1985.

Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. *ICLR*, 2022.

Fausto Bernardini, Chandrajit L Bajaj, Jindong Chen, and Daniel R Schikore. Automatic reconstruction of 3d cad models from digital scans. *International Journal of Computational Geometry & Applications*, 9(04n05):327–369, 1999.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

Kseniya Cherenkova, Elona Dupont, Anis Kacem, Gleb Gusev, and Djamila Aouada. Spelsnet: Surface primitive elements segmentation by b-rep graph structure supervision. *Advances in Neural Information Processing Systems*, 37:1251–1269, 2024.

Andrew R Colligan, Trevor T Robinson, Declan C Nolan, Yang Hua, and Weijuan Cao. Hierarchical cadnet: Learning from b-reps for machining feature recognition. *Computer-Aided Design*, 147: 103226, 2022.

Elona Dupont, Kseniya Cherenkova, Anis Kacem, Sk Aziz Ali, Ilya Arzhannikov, Gleb Gusev, and Djamila Aouada. Cadops-net: Jointly learning cad operation types and steps from boundary-representations. In *2022 International Conference on 3D Vision (3DV)*, pp. 114–123. IEEE, 2022.

Ivor D Faux and Michael J Pratt. *Computational geometry for design and manufacture*. Halsted Press, 1979.

Hiroshi Fujita, Yoshikazu Uchiyama, Toshiaki Nakagawa, Daisuke Fukuoka, Yuji Hatanaka, Takeshi Hara, Gobert N Lee, Yoshinori Hayashi, Yuji Ikedo, Xin Gao, et al. Computer-aided diagnosis: The emerging of three cad systems induced by japanese health care needs. *Computer methods and programs in biomedicine*, 92(3):238–248, 2008.

Mikell Groover and EWJR Zimmers. *CAD/CAM: computer-aided design and manufacturing*. Pearson Education, 1983.

Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics (TOG)*, 41(4): 1–18, 2022.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

David Heesom and Lamine Mahdjoubi. Trends of 4d cad applications for construction planning. *Construction management and economics*, 22(2):171–182, 2004.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11703–11712, 2021.

Pradeep Kumar Jayaraman, Joseph G Lambourne, Nishkrit Desai, Karl DD Willis, Aditya Sanghi, and Nigel JW Morris. Solidgen: An autoregressive model for direct b-rep synthesis. *arXiv preprint arXiv:2203.13944*, 2022.

Sanjay Joshi and Tien-Chien Chang. Graph-based heuristics for recognition of machined features from a 3d solid model. *Computer-aided design*, 20(2):58–66, 1988.

Minseop Jung, Minseong Kim, and Jibum Kim. Contrastcad: Contrastive learning-based representation learning for computer-aided design models. *arXiv preprint arXiv:2404.01645*, 2024.

Antreas Kantaros. Intellectual property challenges in the age of 3d printing: Navigating the digital copycat dilemma. *Applied Sciences*, 14(23):11448, 2024.

Çetin Karahan, Hakan Velioğlu, and Yenal Arslan. Unveiling the shadows: Anticipating future cyber risks and their impacts on businesses. In *Futurisks: Risk management in the digital age*, pp. 17–55. Springer, 2025.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big CAD model dataset for geometric deep learning. *CVPR*, 2019.

Joseph G Lambourne, Karl DD Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12773–12782, 2021.

Benoit Lecallard, Trevor T Robinson, and Simao P Marques. Mesh and geometry manipulations for optimization and inverse design. In *AIAA Scitech 2021 Forum*, pp. 1901, 2021.

Chengnan Li, Tong Yu, Wangyan Li, Ying Liu, and Hongjun Yang. Enhancing cad data integrity and security in supply chain networks using blockchain. *International Journal of Information Systems and Supply Chain Management (IJISSCM)*, 18(1):1–22, 2025.

Yifei Li, Kang Wu, Wenming Wu, and Xiao-Ming Fu. Masked brep autoencoder via hierarchical graph transformer. *arXiv preprint arXiv:2603.14927*, 2026.

Fei Liu. Fast industrial product design method and its application based on 3d cad system. *Computer-Aided Design & Applications*, 18, 2021.

Yunzhong Lou, Xueyang Li, Haotian Chen, and Xiangdong Zhou. Brep-bert: Pre-training boundary representation bert with sub-graph node contrastive learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 1657–1666, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Yatian Pang, Eng Hock Francis Tay, Li Yuan, and Zhenghua Chen. Masked autoencoders for 3d point cloud self-supervised learning. *World Scientific Annual Review of Artificial Intelligence*, 1: 2440001, 2023.

Yuhan Quan, Huan Zhao, Jinfeng Yi, and Yuqiang Chen. Self-supervised graph neural network for mechanical cad retrieval. *arXiv preprint arXiv:2406.08863*, 2024.

Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-recode: Reverse engineering cad code from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9801–9811, 2025.

Josip Stjepandić, Harald Liese, and Amy JC Trappey. Intellectual property protection. In *Concurrent engineering in the 21st century: Foundations, developments and challenges*, pp. 521–551. Springer, 2015.

Peter Szalapaj. *CAD principles for architectural design*. Routledge, 2013.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Karl D.D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic CAD construction. In *SIGGRAPH*, 2021.

Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, et al. Joinable: Learning bottom-up assembly of parametric cad joints. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15849–15860, 2022.

Hongjin Wu, Ruoshan Lei, Yibing Peng, and Liang Gao. Aagnet: A graph neural network towards multi-task machining feature recognition. *Robotics and Computer-Integrated Manufacturing*, 86: 102661, 2024.

Xiang Xu, Pradeep Jayaraman, Joseph Lambourne, Yilin Liu, Durvesh Malpure, and Pete Meltzer. Autobrep: Autoregressive b-rep generation with unified topology and geometry. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–12, 2025.

Can Yao, Kang Wu, Zuheng Zheng, Siyuan Xing, and Xiao-Ming Fu. Brepmae: Self-supervised masked brep autoencoders for machining feature recognition. *arXiv preprint arXiv:2602.22701*, 2026.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19313–19322, 2022.

Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in neural information processing systems*, 35:27061–27074, 2022.

Shuming Zhang, Zhidong Guan, Hao Jiang, Xiaodong Wang, and Pingan Tan. Brepmfr: Enhancing machining feature recognition in b-rep models through deep learning and domain adaptation. *Computer Aided Geometric Design*, 111:102318, 2024.

A Architecture Details

A neural network cannot consume the graph directly, so we must turn each face into a fixed-size vector. UV-grid sampling does this. Every parametric surface comes with a built-in two-dimensional coordinate system (u, v) and a map $S(u, v) \in \mathbb{R}^3$ that sends a point of a flat parameter rectangle to a point on the surface in 3D. For a cylinder, for example, u is the angle around the axis and v is the height. Even though the face is curved in space, its parameter domain is flat, so we can lay a regular grid over that domain.

We place a fixed 10×10 grid of (u, v) values over the face's parameter domain and evaluate the surface at each grid point. At each of the 100 samples, we store 7 channels: position (3-dim): the 3D point $S(u, v)$; normal (3-dim): the unit outward normal $n(u, v)$; trimming mask (1-dim): whether the sample lies inside the trimmed face or in the cut-away region, since the parameter rectangle is larger than the actual face. The result is a $10 \times 10 \times 7$ tensor. A surface CNN processes it into a single per-face embedding h_i, which is the *node feature* in our face-adjacency graph.

A B-rep edge is a parametric curve with a single parameter u. We sample it on a length-10 grid and store 6 channels per sample: 3 for position and 3 for the curve tangent, giving a 10×6 tensor, which a 1D curve CNN turns into an edge embedding.

We now explain our architecture choices. Our encoder (27.9M params, hidden width H=384) maps a B-rep face-graph to (i) a shape embedding $z \in \mathbb{R}^{512}$ and (ii) a per-face feature matrix $h \in \mathbb{R}^{N_F \times 384}$. The full data flow, with exact layer specifications (input tensors in Appendix F):

1. **Face encoder (per-face CNN).** Each face's $10 \times 10 \times 7$ UV-grid is permuted to $[7, 10, 10]$ and passed through three 3×3 conv blocks ($\mathrm{Conv2d}(\mathrm{pad}\,1) \rightarrow \mathrm{GroupNorm} \rightarrow \mathrm{GELU}$) with channel widths $128 \rightarrow H \rightarrow H$ (GroupNorm groups $8, 16, 16$), then $\mathrm{AdaptiveAvgPool2d}(1)$, flatten, and $\mathrm{Linear}(H, H)$, giving a face token in $\mathbb{R}^H$.
2. **Edge encoder (per-edge CNN).** Each edge's 10×6 UV-grid is permuted to $[6, 10]$ and passed through two $\mathrm{Conv1d}(3, \mathrm{pad}\,1) \rightarrow \mathrm{GroupNorm} \rightarrow \mathrm{GELU}$ blocks ($128 \rightarrow H$), $\mathrm{AdaptiveAvgPool1d}(1)$, flatten. To this we concatenate two 16-d learned embeddings of the edge's convexity class (4) and curve-type class (5), then $\mathrm{Linear}(H{+}32, H)$. Each directed graph edge reads its parent edge token by index.
3. **Graph message passing.** N_{gat}=4 GATv2 layers (Brody et al., 2021) ($\mathrm{GATv2Conv}(H \rightarrow H, \text{heads}{=}6, \text{concat}{=}\text{False}, \text{edge_dim}{=}H)$), each using the edge token as an additive attention feature, as a pre-residual with LayerNorm: $x \leftarrow \mathrm{LayerNorm}(x + \mathrm{GATv2}(x, \text{edge_index}, \text{edge_attr}))$.
4. **Global self-attention.** Per-graph face tokens are padded to a dense batch (with a key-padding mask) and passed through N_{tf}=8 TransformerEncoder layers (d_{model}=384, nhead=6, dim_feedforward=1536, GELU, norm_first=True). The per-face features h are read out here (post-Transformer, unpadded). These feed the MTM heads.
5. **Attention pooling.** A single learned query attends over the face tokens via $\mathrm{MultiheadAttention}$ (6 heads), followed by LayerNorm and $\mathrm{Linear}(384, 512)$ to produce the shape embedding z.

The 9.4M ablation model is the same stack with H=256, N_{tf}=2, nhead=8, ff=1024, MoCo off. Both scales are instantiated by the same code; their architecture and training settings are in Appendix E.

B Contrastive Learning

We use momentum-queue contrastive learning (He et al., 2020) at the shape-embedding level. Each shape is encoded into two augmented views; view A is encoded by the online encoder f_q (query $q{=}z_a$), view B by a momentum encoder f_k whose weights are an EMA of the online weights, $\theta_k \leftarrow m\,\theta_k + (1{-}m)\theta_q$ with m=0.999. The momentum encoder's outputs (key $k^+{=}z_b$) are enqueued into a FIFO queue of K=16,384 past keys. The query is trained with InfoNCE on normalized

embeddings,

$$L_{\text{NCE}} = -\log \frac{\exp(q \cdot k^+/\tau)}{\sum_{k \in \mathcal{K}_{\text{batch}}} \exp(q \cdot k/\tau) + \sum_{k \in \text{queue}} \exp(q \cdot k/\tau)}, \tag{3}$$

temperature τ=0.07. Here $\mathcal{K}_{\text{batch}}$ contains the current keys gathered across ranks, including the positive. MTM uses a separate gradient-carrying forward through the online encoder on view B.

C Method Details

C.1 MTM Architecture

On view B, E denotes the unique physical edges, grouped by their parent edge index. We sample a masked set $M \subset E$ with $|M| = \max(1, \text{round}(p_{\text{mask}}|E|))$, p_{mask}=0.7, and delete every directed entry associated with those edges before the online-encoder forward. We mask both directions of each selected edge together and only apply masking when at least one edge remains available for message passing. The encoder ϕ runs GATv2 + Transformer layers over the masked graph, producing per-face features $h_i = \phi_i(\text{face grids}; (F, E \setminus M)) \in \mathbb{R}^{384}$. For each masked edge $e=(i,j) \in M$ we form the ordered endpoint feature $[h_i; h_j] \in \mathbb{R}^{768}$ and pass it through two independent heads, each $\text{Linear}(768, 128) \to \text{GELU} \to \text{Dropout}(0.1) \to \text{Linear}(128, \cdot)$, with 4 and 5 output classes respectively. Both heads are discarded after pretraining (198,025 params total). The loss averages over masked directed entries. Masking is applied to view B only, because view B feeds the auxiliary heads while view A carries the heavy contrastive augmentation.

C.2 Augmentation Details

Offline geometric perturbations. Training uses the first $n_{\text{perts}} = 5$ cached perturbation slots per source. Variants are produced by applying a randomly chosen B-rep operator through OpenCASCADE and checking geometric validity; a cache may contain additional unused slots. The operator is sampled from 14 choices with fixed weights (see Appendix G); feature sizes are relative to the shape's bounding-box diagonal. A shape enters pretraining only if $\geq$ min_perts=2 variants preprocessed successfully; a training pair draws two *different* stored variants of the same source.

Online graph augmentations. Per view we apply: *face drop* (fraction p_{face}=0.3 of faces and incident edges, only if N_F>4, capped at 12); *edge drop* (p_{edge}=0.3 of graph edges, capped at 30); *connected-subgraph sampling* (with probability 0.5, BFS from a random seed keeping the first $\lceil 0.7\, N_F \rceil$ faces); *coordinate jitter* (additive Gaussian, std 0.005 of normalized scale, on xyz); and *asymmetric masking*: view A receives the full heavy augmentation, view B only light augmentation (5% face drop, no edge drop or subgraph sampling).

D Proofs for Section 4

Prediction setup. Write $G(S) = (V, E)$ for the face-adjacency graph of a closed B-rep solid S, where V indexes its faces and E its shared physical edges. Each edge joins two incident faces; parallel adjacencies may be retained. The categorical targets are convexity $c(e) \in \{\text{convex}, \text{concave}, \text{smooth}, \text{knife}\}$ and curve type $t(e) \in \{\text{line}, \text{circle}, \text{ellipse}, \text{B-spline}, \text{other}\}$. Convexity is measured through solid material in a cross-section perpendicular to the edge at a regular point. Smooth edges have an interior angle near π; knife edges have opposing face normals at a degenerate sharp limit. Labels are evaluated at edge midpoints using the thresholds in Appendix F.

For a sampled masked set $\mathcal{M} \subseteq E$, remove its connections, features, and labels, including both directed entries per physical edge. For a query $e \in \mathcal{M}$, X_e contains the complete masked input and its ordered endpoint indices. Write $Z_e = (h_i, h_j) = g(X_e)$, where g is the deterministic encoder followed by selection of the two endpoint feature vectors. Let Y_e be either target and let $q(y \mid Z_e)$ be a head assigning label probabilities. Throughout, loss is cross-entropy $-\log q(Y_e \mid Z_e)$, with expectations over sampled solids, masks, and queried edges. For the true conditional distribution p,

$$H(Y \mid X) = \mathbb{E}[-\log p(Y \mid X)], \qquad L^* = \inf_q \mathbb{E}[-\log q(Y_e \mid Z_e)],$$

where the infimum is over unrestricted probability heads for the fixed encoder. Zero conditional entropy means the label is determined with probability one. Thus the identifiability statement concerns expected loss, not merely correct predictions on a finite sample. A fixed finite-capacity head may have a higher optimal loss than L^*. The population conditional entropies are not estimated in this work, and the bound alone does not imply a strictly positive prediction error for MTM.

PROOF OF THEOREM 1 (IDENTIFIABILITY)

For a categorical target Y and representation Z, write $p(\cdot \mid Z)$ for the true conditional distribution and $q(\cdot \mid Z)$ for the head's predicted distribution. The Kullback–Leibler divergence $D_{\mathrm{KL}}(p\|q) = \sum_y p(y)\log(p(y)/q(y))$ is nonnegative. With conditional entropy defined in Section 4,

$$\mathbb{E}[-\log q(Y \mid Z)] = H(Y \mid Z) + \mathbb{E}[D_{\mathrm{KL}}(p(\cdot \mid Z)\|q(\cdot \mid Z))] \geq H(Y \mid Z).$$

Taking the infimum over unrestricted heads gives $H(Y \mid Z)$. Since $Z = g(X)$, conditioning on Z cannot reveal more information about Y than conditioning on X, so $H(Y \mid Z) \geq H(Y \mid X)$. For a finite categorical target, zero conditional entropy means the target is determined almost surely by the conditioning variable. This argument applies separately to convexity and curve type and imposes no ordering between their conditional entropies. □

PROOFS OF THEOREMS 2 AND 3 (SEPARABILITY)

Restricted class and formal statement. Let $\gamma(f)$ denote a face's surface and trimmed boundary up to a rigid motion of that face, omitting its material-side orientation, relative placement, and boundary-to-neighbor correspondence. The restricted class consists of deterministic encoders using only (γ, G) and equivariant to node relabeling. Face-shape features γ remain available, while G supplies adjacency without edge attributes. For each target $y \in \{c, t\}$, we construct two instances (S, e) and (S', e') and a graph isomorphism $\phi : G(S) \to G(S')$ such that

$$\gamma(f) = \gamma(\phi(f)) \quad \text{for every face } f, \qquad \phi(e) = e', \qquad y(e) \neq y(e').$$

The isomorphism includes a matching of physical edges when parallel adjacencies occur. It therefore induces an isomorphism after masking e and e', respectively. The two constructions below establish these conditions; the concluding equivariance argument gives identical ordered endpoint features.

Convexity across two solids (Theorem 2). Let the base solid be the slab $B = [-2, 2] \times [-2, 2] \times [-2, 0]$, with top face at $z = 0$, and let $Q = [-1, 1] \times [-1, 1] \times \{0\}$ be a square inscribed in it. Fix any height $0 < h < 2$ and apex $a = (0, 0, h)$, $a' = (0, 0, -h)$.

- S (*bump*) $= B$ union the right pyramid with base Q and apex a.
- S' (*dimple*) $= \overline{B \setminus P'}$, where P' is the closed right pyramid with base Q and apex a', and the overbar denotes closure.

Both are valid closed solids: $h < 2$ keeps the dimple apex a' strictly above the slab bottom $z = -2$, so S' does not self-intersect.

Face bijection ϕ. Bottom, the four slab sides, and the top face (the square frame surrounding the opening Q at $z = 0$) occur identically in S and S'; map each to itself. The remaining faces are four triangles. In S the east triangle is $T_E = \mathrm{conv}\{(1, 1, 0), (1, -1, 0), a\}$ (and analogously T_S, T_W, T_N); in S' it is $T'_E = \mathrm{conv}\{(1, 1, 0), (1, -1, 0), a'\}$, where conv denotes the convex hull. Set $\phi(T_E) = T'_E$, etc.

ϕ preserves individual face shapes. T'_E is the reflection of T_E through $z = 0$, hence congruent; each triangle is isosceles (base edge length 2, two slant edges $\sqrt{2 + h^2}$), so it is congruent to its mirror image by a proper rotation (about the perpendicular bisector of its base), and $\gamma(f) = \gamma(\phi(f))$ even if γ is taken up to orientation-preserving isometry. All non-triangular faces are literally identical.

ϕ is a graph isomorphism. In both solids the four triangles share slant ("hip/valley") edges pairwise at the apex and share their base edges with the top frame; B's faces meet in the same pattern. The incidence pattern is identical, so ϕ: $G(S) \to G(S')$ is an isomorphism.

A convexity flip. Take e = the slant edge shared by T_E, T_S, from corner $(1, -1, 0)$ to a. In S this is a ridge of the bump: in a perpendicular cross-section, the material occupies an angle smaller than π, so $c(e) =$ convex. Its image $\phi(e)$ runs from $(1, -1, 0)$ to a' along the indentation. There the material occupies an angle larger than π, so $c(\phi(e)) =$ concave. Thus $c(e) \neq c(\phi(e))$ while (γ, G) agree, establishing the hypotheses of the theorem.

Curve type within one solid (Theorem 3). Take the slab $B = [-1, 3] \times [-1, 3] \times [-1, 0]$ and subdivide its top face. Define

$$g(x) = \sqrt{1 - (x - \tfrac{1}{2})^2} - \tfrac{\sqrt{3}}{2}, \qquad 0 \le x \le 1.$$

For $i, j \in \{0, 1\}$, let

$$F_{ij} = \{(x, y, 0) : i \le x \le i + 1,\ j + g(x - i) \le y \le j + 1 + g(x - i)\}.$$

These four patches have disjoint interiors and are translations of one another. Each has two straight sides and two circular-arc sides. The remaining top surface is a single frame face surrounding their union. Together with the slab's side and bottom faces, this is a closed B-rep of B.

The four patch nodes form a cycle, and each shares two boundary edges with the frame. The map $\phi(F_{ij}) = F_{ji}$, fixing every other face, preserves all individual face shapes and the adjacency graph without edge attributes, including parallel adjacencies if retained. Choose e between F_{00} and F_{10}, and e' between F_{00} and F_{01}. Their boundaries are, respectively,

$$e = \{(1, y, 0) : 0 \le y \le 1\}, \qquad e' = \{(x, 1 + g(x), 0) : 0 \le x \le 1\}.$$

Thus $t(e) =$ line and $t(e') =$ circle. The graph isomorphism sends e to e'. Taking $S' = S = B$ with this face partition gives the two prediction instances required for curve type. This witness separates two queries within one solid, rather than two geometrically different solids.

Representations and prediction. In either construction, masking corresponding edges preserves the input correspondence under ϕ, including the queried endpoint indices. An encoder using only (γ, G) therefore receives isomorphic masked inputs. Equivariance to node relabeling gives identical embeddings at corresponding endpoints. A shared head must return the same label distribution for both queries and cannot classify both different labels correctly. Hence correctly predicting both requires endpoint features that distinguish these instances.

The result concerns this restricted representation class, which excludes relative placement, material-side orientation, boundary-to-neighbor correspondence, and surviving edge attributes. MTM receives shared-coordinate face samples, outward normals, and the geometry and labels of surviving edges, so this indistinguishability argument does not apply directly to its encoder. The result does not show that optimization reaches perfect prediction, that MTM improves over another objective using the same inputs, or that different endpoint features imply different pooled shape embeddings. □

E PRETRAINING HYPERPARAMETERS

Combined loss. $L = L_{\text{NCE}} + \lambda_{\text{topo}} L_{\text{topo}}$ with λ_{topo}=0.5 for the main model and 0.3 for the smaller model. L_{NCE} is MoCo InfoNCE (main model) or symmetric in-batch NT-Xent (9.4M, MoCo off). Auxiliary heads are discarded after pretraining; only the original architecture transfers downstream.

The 9.4M continued training checkpoints start from a shared ABC-only checkpoint pretrained contrastively for 100 epochs and continue for 10 epochs, with the settings in the table above. These runs load the architecture and reset the auxiliary heads, optimizer and, where used, MoCo queue. Each scale compares no masking/no MTM loss, masking without the loss, and masking with the loss.

The from-scratch pairs in Figure 4 instead start from random initialization and train for 30 epochs on the mixed corpus. The 9.4M pair uses the smaller-model recipe above. The 27.9M pair uses the main recipe with one rank and accumulation 32, preserving its effective optimization batch but changing its current contrastive batch. The MTM off runs disable both topology masking and its prediction loss. All runs use one pretraining seed; probe and fine-tuning repetitions reuse the resulting weights.

Parameter	Our main 27.9M model	Our 9.4M version
Hidden H	384	256
GATv2 layers / heads	4 / 6	4 / 8
Transformer layers / heads / FFN	8 / 6 / 1536	2 / 8 / 1024
Output (shape-embed) dim	512	512
Parameter count	27.9M	9.43M
MoCo queue K / momentum	16,384 / 0.999	off (0)
InfoNCE temperature τ	0.07	0.07
topo_mask_prob / topo_aux_weight	0.7 / 0.5	0.4 / 0.3
Face drop p_{face} (abs cap)	0.3 (12)	0.3 (12)
Edge drop p_{edge} (abs cap)	0.3 (30)	0.3 (30)
Subgraph trigger / keep ratio	0.5 / 0.7	0.5 / 0.7
Coordinate jitter std (frac)	0.005	0.005
Optimizer	AdamW β=$(0.9, 0.95)$	AdamW β=$(0.9, 0.95)$
LR / weight decay	1e-4 / 0.01	3e-4 / 0.01
Schedule	3-ep warmup $\rightarrow$ const	1-ep warmup $\rightarrow$ const
Epochs	100 (from scratch)	10 (continuation)
Per-GPU batch $\times$ accum $\times$ ranks	$64 \times 16 \times 2$	$96 \times 16 \times 1$
Effective batch	2,048	1,536
Precision	bf16 autocast	bf16 autocast
Max faces / shape	400	400
Perturbations (n_{perts} / min)	5 / 2	5 / 2

F PREPROCESSING AND INPUT REPRESENTATION

Every solid (a STEP file) is converted to a face-graph with UV-grid features using the OpenCASCADE (OCC) kernel. All geometry below is deterministic given the solid; no learned or stochastic component enters preprocessing.

Faces $\rightarrow$ nodes. For each face we build a $10 \times 10 \times 7$ grid by sampling the parametric surface on a uniform 10×10 lattice over $[u_{\min}, u_{\max}] \times [v_{\min}, v_{\max}]$ (degenerate ranges clamped to unit length). The 7 channels are: xyz position (3), unit surface normal (3, flipped when the face orientation is `REVERSED` so it points outward), and a trim mask (1) that is 1 when the sample lies inside/on the face's trimmed region (`BRepClass_FaceClassifier`) and 0 otherwise.

Edges $\rightarrow$ typed graph edges. For each B-rep edge adjacent to ≥ 2 faces we build a 10×6 grid by sampling the edge curve on 10 uniform parameter values: xyz position (3) and unit tangent (3). The edge is instantiated as directed graph edges between every ordered pair of its adjacent faces. Each carries two categorical labels:

- **Convexity** $\in \{\text{concave}=0, \text{convex}=1, \text{smooth}=2, \text{knife}=3\}$, computed at the edge midpoint from adjacent-face normals n_a, n_b and the coedge tangent t_a oriented along face a's boundary: $n_a \cdot n_b > 0.999$ gives smooth, < -0.999 gives knife; otherwise $(n_a \times n_b) \cdot t_a > 0$ gives convex and a non-positive value gives concave. The orientation is essential: swapping the faces reverses both the cross product and the coedge tangent, preserving the physical label.
- **Curve type** $\in \{\text{Line}=0, \text{Circle}=1, \text{B-spline}=2, \text{Ellipse}=3, \text{Other}=4\}$ from the OCC curve class (`GeomAbs_*`); Bézier/hyperbola/parabola/unknown fold into "Other".

These are the labels from fresh extraction; surviving edge labels also condition the edge CNN via 16-d learned embeddings.

Cached convexity labels. For pretraining, the historical cache was corrected in place using stored face and edge grids, preserving its perturbation geometry. This approximation agreed with fresh oriented-coedge extraction on 99.65% of checked F360 entries and 99.92% of checked MFCAD entries. Smooth/knife labels were unchanged; unresolved binary entries retained their old label. All reported pretraining runs use this corrected cache, rather than an exact kernel rebuild.

Normalization. xyz channels of both face and edge grids are centered on the shape's bounding-box center and divided by its bounding-box diagonal, giving a unit-diagonal shape. Grids are stored in fp16.

G SYNTHETIC-DATA GENERATION AND PRETRAINING CORPUS

Corpus. Pretraining uses real ABC solids plus a procedurally generated cadquery synthetic corpus. A source is kept only if ≥ 2 of its 5 perturbed variants preprocessed into valid solids; 2% of sources are held out for validation loss. The main run excludes prefixes `steps_zc`, `f360_step`, and `mfcad`, leaving 3,328,190 training and 67,922 validation sources. The 9.4M full-corpus runs exclude `steps_zc` and `f360_step`; the ABC-only arms use the ABC subset.

Synthetic source generation. We adapt the procedural sketch-and-extrude generator of CAD-Recode (Rukhovich et al., 2025). Each sketch is formed by combining randomly sized and positioned rectangles and circles through planar Boolean unions and differences, producing profiles with straight and curved boundaries and optional holes. Sketches are placed on randomly selected, offset coordinate planes and extruded with sampled depths and directions. The resulting volumes are combined through Boolean union. The generator normalizes and rounds geometric parameters and filters out degenerate or non-contributing operations. Our adaptation constructs the resulting B-reps through OpenCASCADE and converts them into the same face/edge UV-grid representation used for ABC, with automatically derived convexity and curve-type targets. These generated source shapes subsequently undergo the perturbation and preprocessing procedures described below.

Perturbation operators. Each cached variant is produced by sampling *one* operator from the distribution below and applying it with OCC, retrying/validating for a watertight solid (feature sizes are relative to the bounding-box diagonal d).

Operator	Prob.	Key parameters
Rigid + uniform transform	0.15	rotate $\pm\pi$/translate $\pm 0.5d$/mirror/scale $\in [0.7, 1.4]$
Anisotropic scale	0.10	independent x/y/z scale $\in [0.7, 1.4]$
Shear	0.07	one axis by another, angle 5–20°
Drill through-hole	0.08	cylinder cut, radius $\in [0.05, 0.20]$ extent, $\leq 0.10d$
Multiple holes	0.07	2–3 drilled holes
Add boss	0.07	fused cylinder, radius $\leq 0.08d$, height 0.05–$0.15d$
Multiple bosses	0.06	2–3 bosses
Rectangular slot	0.07	through-cut box, len 0.20–0.50, width 0.05–0.15 extent
Rectangular pocket	0.06	blind box cut, depth 0.05–$0.15d$
Rib	0.07	fused thin box, len 0.30–0.60, width 0.03–0.10 extent
Fillet	0.07	1 or 3 edges, radius 0.01–$0.04d$
Variable fillet	0.06	3–5 edges, independent radii
Chamfer	0.04	1 or 3 edges, distance 0.01–$0.04d$
Variable chamfer	0.03	3–5 edges, independent distances

Two views of a training pair are two *distinct* stored variants of the same source, each then passed through the online graph augmentations.

H DOWNSTREAM EVALUATION PROTOCOLS

All downstream tasks load a pretrained checkpoint and read per-face encoder features (segmentation/probe) or the pooled shape embedding (classification). Unless stated, optimization is AdamW (β=$(0.9, 0.95)$), inputs capped at 1,000 faces, features/grids as in Appendix F. Our supervised downstream runs fine-tune the architecture end-to-end and select the checkpoint by validation accuracy, except F360 full-data, which uses validation macro mIoU. Tables report means and, where shown, sample standard deviations.

Frozen linear probe (F360). The checkpoint is frozen. We use saved per-face features from 3,000 sampled shapes, remove all official-test shapes, and fit multinomial logistic regression on

an $80/20$ shape-grouped split of the remaining official-training pool. The retained counts are $2{,}541/2{,}558/2{,}552$ for probe seeds $0/1/2$. We use C=1.0, max_iter=2000. The plotted values average these probe resamplings, with fixed pretrained weights in each arm.

F360 full-data segmentation. End-to-end fine-tune of pretrained checkpoint + a per-face head $\mathrm{Linear}(H, H) \to \mathrm{GELU} \to \mathrm{Dropout}(0.1) \to \mathrm{Linear}(H, 8)$. We reserve 10% of the official training pool for validation. LR 2e-4, weight decay 0.01, 50 epochs, batch 16, 2 warmup epochs, cosine decay, uniform LR. The LR is selected by seed-0 validation mIoU from $\{5\mathrm{e}{-5}, 1\mathrm{e}{-4}, 2\mathrm{e}{-4}\}$, then used for seeds 1 and 2; the table reports the mean across all three runs. The 9.4M downstream control arms retain LR 1e-4.

F360 few-shot segmentation. We use 8-way support. Before drawing support, we reserve 200 official-training shapes for validation using split seed 0. For each class, take shots $\in \{10, 20\}$ shapes whose face-label set contains that class; the support set is their union. Ours and scratch use identical support, validation and test memberships. We average run seeds $0/1/2$, which vary the support draw and fine-tuning randomness. Fine-tune end-to-end with layer-wise LR decay 0.85, base LR 5e-5, weight decay 0.05, cosine, 50 epochs, batch 8; evaluate on the full test split.

SolidLetters classification. We reserve 20 training shapes per selected class for validation using split seed 0, before drawing support. Few-shot uses 10-way support with $10/20$ shapes per class; full-data uses the remaining training pool for all 26 classes. The architecture is fine-tuned with a head $\mathrm{Linear}(512, 512) \to \mathrm{GELU} \to \mathrm{Dropout}(0.1) \to \mathrm{Linear}(512, K)$ on the pooled shape embedding. Few-shot uses base LR 5e-5, layer-wise decay 0.85, weight decay 0.05, cosine, 100 epochs and batch 16. Results average 20 class-draw seeds with run seed 0. Full-data uses LR 3e-5, layer-wise decay 0.95, weight decay 0.05, 30 epochs, batch 64, two warmup epochs and cosine decay, averaged over five fine-tuning seeds. Validation uses 200 additional shapes per few-shot episode and 520 for full-data.

MFInstSeg label-ratio segmentation. All rows use training subsets at ratios $\{0.1, 0.5, 1, 100\}\%$ of the official AAGNet $70/15/15$ train split ($44/219/437/43{,}729$ shapes), evaluated on the same $9{,}375$-shape test split; the metric is per-face micro accuracy over the 25 classes (24 features + stock). We report mean and standard deviation over seeds (3 for each low ratio, 2 for 100%). Our model and scratch use the full $9{,}373$-shape validation split in addition to the labeled training subset.

Our model and the from-scratch control fine-tune the architecture end-to-end with a per-face head $\mathrm{Linear}(H, H) \to \mathrm{GELU} \to \mathrm{Dropout}(0.1) \to \mathrm{Linear}(H, 25)$ (24 features + stock), layer-wise LR decay 0.85 (layer l gets LR $\cdot\ 0.85^{(L-l)}$, L=1+N_{gat}+N_{tf}+1), base LR 2e-4 for the main low-label runs, selected by mean validation accuracy over three seeds at 0.1% from $\{2\mathrm{e}{-5}, 5\mathrm{e}{-5}, 1\mathrm{e}{-4}, 2\mathrm{e}{-4}\}$ and reused at $0.5\%/1\%$. All use weight decay 0.05 and cosine scheduling. The 0.1% cell uses 100 epochs at batch 8, $0.5\%/1\%$ use 50 epochs at batch 32, and 100% uses 30 epochs at batch 64; the smallest budget is gradient-starved at 50 epochs. Under bf16 autocast a single degenerate batch can NaN-poison the weights through gradient clipping; we guard by skipping any step with non-finite loss or gradient norm. GAT, GraphSAGE, UV-Net, and AAGNet are trained from scratch using the 10-dim face attribute and 7-channel face UV-grid that AAGNet consumes, with architecture-specific graph inputs. GAT and GraphSAGE are generic 3-layer message-passing encoders (edges carry no attributes); UV-Net uses its own surface-conv + NNConv graph encoder with the 12-dim edge attribute fed through a small MLP in place of its curve branch (this AAG has no edge UV-grids); AAGNet uses its authors' gAAG-GNN with the full instance with bottom multitask loss and EMA. All four train with AdamW, LR 1e-2, cosine, cross-entropy, 100 epochs at batch 128 for the capped ratios and 30 epochs at batch 256 for 100%.

CADSynth label-ratio segmentation. We use the official $10{,}000$-shape test list. From the remaining $90{,}000$ shapes, we draw $10{,}000$ validation shapes and use the other $80{,}000$ for training. These pools are disjoint. The label ratios select $80/400/800$ training shapes; the validation labels are additional to those budgets. We use the MFInstSeg head, layer-wise LR decay 0.85, weight decay 0.05, and cosine scheduling. The initial budgets are 100 epochs at batch 8 for 0.1%, and 50 epochs at batch 32 for the other ratios. We evaluate LR 5e-5 and 2e-4 over seeds $0/1/2$, selecting the configuration by mean validation accuracy separately for each ratio. For 0.1%, the candidate set was extended to 200 epochs at both learning rates. Each run selects its checkpoint by validation accuracy. The selected configurations are LR 5e-5/200 epochs at 0.1% and LR 2e-4/50 epochs otherwise. The

published comparator rows are quoted from HierMAE rather than reproduced with our constructed train/validation split.

MFCAD few-shot segmentation. MFCAD has no official test-file list. We reserve 200 shapes from the full pool for validation with split seed 0, select 10 classes with class seed 0, and draw $10/20$ support shapes per selected class. The eligible remainder is used for testing. Closed-world evaluation uses the selected classes; open-world evaluation retains all 16 classes. Fine-tuning uses LR 5e-5, layer-wise decay 0.85, weight decay 0.05, cosine, 50 epochs and batch 8, over run seeds $0/1/2$.

Table 9: Frozen F360 probe macro mIoU.

Model	Start	MTM off	Mask only	MTM on
9.4M	Random	0.540 ± 0.024	–	0.570 ± 0.018
27.9M	Random	0.524 ± 0.012	–	0.608 ± 0.057
9.4M	Continuation	0.519 ± 0.022	0.525 ± 0.029	0.531 ± 0.013
27.9M	Continuation	0.585 ± 0.031	0.587 ± 0.027	0.617 ± 0.046

I Ablations

I.1 MTM controls

Table 9 reports the frozen-probe means and sample standard deviations over the three training-pool resamplings. Masking-only controls use the shared-parent continuation design described in Appendix E.

The 9.4M continuation runs in Table 6 are evaluated with matched downstream recipes.

I.2 UV-Net objective comparison

The UV-Net arms in Table 7 train from scratch on ABC only for 30 epochs, using 64-dimensional face features, three graph-convolution layers, a 128-dimensional shape embedding, batch 96, accumulation 16, LR 3e-4, weight decay 0.01, and three warmup epochs followed by a constant learning rate. MoCo is disabled; the contrastive arms use in-batch InfoNCE. The masking control and both MTM arms use masking probability 0.4; the MTM loss weight is 0.5. The contrastive-only runs disables both topology masking and its prediction loss, the masking control removes edges without predicting their attributes, and MTM-only disables InfoNCE.

All arms use pretraining seed 0. The frozen F360 probes use 3,000 official-training shapes and a shape-grouped split, excluding official test shapes before selecting this pool. This is a training-pool probe, separate from the main F360 development-and-refit protocol in Appendix H; its scores are compared only across these UV-Net runs.

For downstream evaluation, we fine-tune on F360 10-shot segmentation and MFInstSeg at 0.1%, 0.5%, and 1% label budgets (Table 10). Validation selects the checkpoint by accuracy; the reported test score is from that checkpoint. MFInstSeg uses the official validation and test pools. F360 reserve validation from the official training pool.

All fine-tuning runs use cosine learning-rate decay, layer-wise LR decay 0.85, and weight decay 0.05. F360 uses LR 5e-5, batch 8, and 50 epochs; SolidLetters uses LR 5e-5, batch 16, and 100 epochs. MFInstSeg uses LR 2e-4, with batch 8 and 100 epochs at 0.1%, and batch 32 and 50 epochs at the larger budgets. Table 7 reports means; Table 10 also gives sample standard deviations over probe or fine-tuning seeds $0/1/2$. Under MTM-only, the shape-level readout receives no pretraining loss and is learned during classification fine-tuning.

I.3 AAGNet objective comparison

We use native AAGNet's face and edge attributes, UV grids, graph encoder, and semantic segmentation head. Pretraining uses only the 43,728 unlabeled MFInstSeg training shapes in the prepared

Table 10: UV-Net frozen-probe and downstream fine-tuning results (%). We report accuracy and macro mIoU for segmentation.

Task / metric	Contrastive only	Contrastive + masking	Contrastive + MTM	MTM only
F360 frozen probe mIoU	21.53±1.22	19.33±1.83	32.05±1.71	**37.66±1.03**
F360 10-shot accuracy	49.58±0.91	51.01±1.60	61.67±2.46	**68.59±1.36**
F360 10-shot mIoU	15.05±2.66	14.42±3.14	22.47±1.38	**25.81±1.83**
MFInstSeg 0.1% accuracy	54.56±2.65	57.63±1.76	59.36±2.12	**64.32±2.05**
MFInstSeg 0.1% mIoU	18.55±3.65	20.54±3.48	21.63±2.79	**27.36±0.87**
MFInstSeg 0.5% accuracy	61.66±2.56	66.93±2.45	68.60±3.10	**69.58±2.39**
MFInstSeg 0.5% mIoU	22.88±2.47	26.46±3.11	27.11±4.14	**30.50±4.05**
MFInstSeg 1% accuracy	75.38±1.83	75.87±1.74	79.58±0.91	**80.10±1.96**
MFInstSeg 1% mIoU	36.58±2.62	35.49±2.20	41.78±1.98	**44.31±2.40**

Table 11: Native AAGNet objective comparison on MFInstSeg (%). Bold indicates the highest mean in each row.

Label budget / metric	Contrastive only	Contrastive + MTM	MTM only
0.1% accuracy	58.79±1.06	66.20±0.30	**70.17±0.94**
0.1% mIoU	25.71±1.27	36.90±2.99	**42.66±3.00**
0.5% accuracy	80.88±0.40	85.26±0.55	**87.85±1.06**
0.5% mIoU	58.91±3.00	66.89±2.57	**70.87±2.07**
1% accuracy	86.86±1.25	92.14±0.13	**92.94±0.17**
1% mIoU	71.27±0.86	78.76±0.53	**80.54±0.43**

Table 12: MTM-only versus contrastive+MTM on our 9.4M architecture at matched pretraining epochs (%). Bold indicates the highest mean in each row.

Task / metric	Epoch	Contrastive + MTM	MTM only
F360 10-shot accuracy	10	**72.05**	71.87
F360 10-shot accuracy	15	**73.61**	72.54
F360 10-shot mIoU	10	42.29	**43.30**
F360 10-shot mIoU	15	43.89	**44.72**
F360 20-shot accuracy	10	**78.48**	76.92
F360 20-shot accuracy	15	**79.48**	77.09
F360 20-shot mIoU	10	**51.21**	50.58
F360 20-shot mIoU	15	**52.32**	51.52
MFInstSeg 0.1% accuracy	10	**75.62±1.57**	74.68±0.55
MFInstSeg 0.1% accuracy	15	74.82±1.71	**75.53±1.40**
MFInstSeg 0.1% mIoU	10	**45.80±2.27**	44.19±2.67
MFInstSeg 0.1% mIoU	15	**45.53±3.71**	44.90±1.43
MFInstSeg 0.5% accuracy	10	**82.01±0.39**	80.88±1.07
MFInstSeg 0.5% accuracy	15	**82.03±0.51**	81.63±1.27
MFInstSeg 0.5% mIoU	10	**51.52±0.90**	48.77±0.97
MFInstSeg 0.5% mIoU	15	**51.66±0.96**	49.75±0.87
MFInstSeg 1% accuracy	10	89.00±0.05	**89.71±0.45**
MFInstSeg 1% accuracy	15	89.33±0.23	**90.48±0.33**
MFInstSeg 1% mIoU	10	**70.83±0.16**	70.69±0.82
MFInstSeg 1% mIoU	15	71.46±0.34	**72.15±0.80**

native cache, with normalization fitted on this training pool. No validation or test shape enters pretraining. This is an in-domain comparison. All objectives share initialization, training data, and a budget of 27,320 updates with batch size 32, LR 3e-4, weight decay 0.01, and 2,732 warmup up-

Table 13: MTM-only versus contrastive with MTM on our 27.9M backbone at matched pretraining epochs (%). Bold indicates the highest mean in each row.

Task / metric	Epoch	Contrastive + MTM	MTM only
F360 10-shot accuracy	10	**73.22**	72.22
F360 10-shot accuracy	15	**76.58**	71.18
F360 10-shot mIoU	10	**44.40**	44.35
F360 10-shot mIoU	15	**47.50**	44.45
F360 20-shot accuracy	10	**77.70**	75.08
F360 20-shot accuracy	15	**80.80**	76.90
F360 20-shot mIoU	10	**51.03**	48.95
F360 20-shot mIoU	15	**54.60**	50.84
MFInstSeg 0.1% accuracy	10	**73.15±1.50**	71.78±1.78
MFInstSeg 0.1% accuracy	15	75.03±0.92	**75.69±1.03**
MFInstSeg 0.1% mIoU	10	**44.01±1.81**	42.15±2.35
MFInstSeg 0.1% mIoU	15	**46.98±3.02**	46.18±2.62
MFInstSeg 0.5% accuracy	10	**86.65±0.67**	85.32±0.53
MFInstSeg 0.5% accuracy	15	**88.04±0.52**	87.33±0.31
MFInstSeg 0.5% mIoU	10	**65.58±1.51**	61.97±0.72
MFInstSeg 0.5% mIoU	15	**69.25±0.76**	66.18±0.61
MFInstSeg 1% accuracy	10	**91.12±0.33**	91.01±0.16
MFInstSeg 1% accuracy	15	92.05±0.62	**92.26±0.18**
MFInstSeg 1% mIoU	10	**75.41±0.58**	74.61±0.60
MFInstSeg 1% mIoU	15	**78.11±0.72**	77.78±0.51

dates. Contrastive learning uses in-batch InfoNCE; the MTM arms use masking probability 0.4 and prediction-loss weight 0.5.

All runs are fine-tuned for 100 epochs with batch size 8, LR 2e-4, weight decay 0.05, and cosine decay, using semantic segmentation supervision without instance or bottom-face losses. Checkpoints are selected by validation accuracy and evaluated on the full official test pool of $9{,}375$ shapes. The 0.1% comparison uses all $9{,}373$ validation shapes. The 0.5% and 1% settings were added after initial 0.1% results were observed and use a fixed $2{,}000$-shape validation subset, shared across objectives, for model selection. The test pool is unchanged.

Table 11 reports means and sample standard deviations over downstream seeds $0/1/2$, each reusing one pretrained checkpoint per objective. Adding MTM to contrastive learning improves both metrics in every paired seed at every reported label budget. MTM-only also exceeds the combined objective in every paired seed.

I.4 MTM-ONLY ON OUR ARCHITECTURE

We compare MTM-only with contrastive+MTM on our 9.4M and 27.9M architectures using checkpoints from pretraining epochs 10 and 15 (Tables 12 and 13). Each arm has one pretraining seed. All test results use validation-selected fine-tuning checkpoints, following Appendix H.