Foundry: Distilling 3D Foundation Models for the Edge

Supplementary Material

Contents

A. Datasets and metrics

This section briefly reports information for each used dataset in this paper.

A.1. Datasets

ShapeNet55 [3] (SN55). The dataset contains synthetic 3D meshes from 55 categories, split into 41,952 shapes for training and 10,518 for validation and testing. Below is the list of shape names: *airplane, trash bin, bag, basket, bathtub, bed, bench, birdhouse, bookshelf, bottle, bowl, bus, cabinet, camera, can, cap, car, cellphone, chair, clock, keyboard, dishwasher, display, earphone, faucet, file cabinet, guitar, helmet, jar, knife, lamp, laptop, loudspeaker, mailbox, microphone, microwaves, motorbike, mug, piano, pillow, pistol, flowerpot, printer, remote, rifle, rocket, skateboard, sofa, stove, table, telephone, tower, train, watercraft, washer*.

ModelNet40 [21] (MN40). The dataset consists of synthetic 3D meshes from 40 distinct classes, with 9,840 samples for training and 2,468 for validation and testing. The shape names are: *airplane, bathtub, bed, bench, bookshelf, bottle, bowl, car, chair, cone, cup, curtain, desk, door, dresser, flower pot, glass box, guitar, keyboard, lamp, laptop, mantel, monitor, night stand, person, piano, plant, radio, range hood, sink, sofa, stairs, stool, table, tent, toilet, tv stand, vase, wardrobe, xbox*.

ScanObjectNN [17] (SONN). ScanObjectNN comprises 2,304 training examples and 581 validation/test samples from 15 semantic classes. The samples are real-scanned objects divided into three splits: (i) `OBJ-BG` includes both object and background, (ii) `OBJ-ONLY` contains only the object, and `PB-T50-RS` objects and backgrounds perturbed samples with challenging random augmentations to increase classification complexity. Below is the list of shape names: *bag, bin, box, cabinet, chair, desk, display, door, shelf, table, bed, pillow, sink, sofa, toilet*.

OmniObject3D [20] (OO3D). The dataset is divided into training and validation/testing sets with a 80/20 split per class. Typically, for a class with 100 elements, the last 20 are used for validation/testing, and the rest for training. If a class has 2 or fewer elements, at least one is used for testing and one for training. The final split counts are 4,641 for training, and 1,270 for validation/testing. The dataset contains 216 categories, including: *anise, antique, apple, asparagus, backpack, ball, bamboo shoots, banana, battery, beauty blender, bed, belt, biscuit, book, bottle, bowl, box, boxed beverage, bread, broad bean, broccoli, broccolini, brush, brussels sprout, bucket noodle, bumbag, bun, burrito, cabinet, cake, calculator, candle, candy, canned beverage, carrot, cauliflower, chair, cheese, cherry, chess, chicken leg, chili, china, chinese chess, chocolate, clock,*

coconut, conch, corn, cucumber, cup, dice, dinosaur, dish, doll, donut, drawing, drum, dumbbell, dumpling, durian, dustbin, earplug, egg, egg tart, eraser, facial cream, fan, fig, fire extinguisher, flash light, flower pot, flute, fork, frisbee, garage kit, garlic, ginger, glasses, glasses case, gloves, green bean cake, guitar, hair dryer, hairpin, hamburger, hami melon, hammer, hand cream, handbag, hat, haw thorn, hazelnut, helmet, hot dog, house, insole, kennel, kettle, keyboard, kite, kiwifruit, knife, laptop, laundry detergent, lemon, light, lipstick, litchi, longan, loquat, lotus root, magnet, mango, mangosteen, medicine bottle, microwaveoven, monitor, mooncake, mouse, mushroom, nipple, onion, orange, ornament, ornaments, oyster, package, pad, pan, pancake, pastry, peach, peanut, pear, pen, picnic basket, pie, pillow, pineapple, pinecone, pistachio, pitaya, pizza, plant, plug, pomegranate, popcorn, potato, potato chips, power strip, projector, puff, pumpkin, razor, red jujube, red wine glass, remote control, rice cake, ricecooker, rubik cube, sandwich, sausage, scissor, shampoo, shaomai, shoe, shrimp, skateboard, soap, sofa, spanner, speaker, squash, starfish, steamed bun, stool, strawberry, suitcase, sushi, sweet potato, sword bean, table, table tennis bat, tape measure, teapot, teddy bear, thermos, thimble, timer, tissue, tomato, tooth brush, tooth paste, toothpick box, toy animals, toy boat, toy bus, toy car, toy motorcycle, toy plane, toy plant, toy train, toy truck, tvstand, umbrella, vase, waffle, wallet, walnut, watch, water chestnut, watermelon, whistle, yam, zongzi.

ShapeNetPart [23] (SNP). This part segmentation dataset contains 16 object types and 50 part categories, comprising 16,881 synthetic 3D shapes. The data is split into a training set of 13,998 shapes and a validation/testing set of 2,874 shapes.

Objaverse [4]. This unannotated dataset contains 661,575 objects in the train set and 3000 in the validation set. PointLLM [22] provides annotations via description on each object but it was not utilized in our framework.

A.2. Metrics

Classification task. We use top-1 overall accuracy which can be sometimes abbreviated as `OA@1`.

Part segmentation task. We use mean intersection over union abbreviated as `mIoU` for classes and instances. We denote them respectively by mIoU_C and mIoU_I. When unspecified, mIoU_C is used as `mIoU`.

B. Implementation details

We give some details about the implementation of each tested method.

B.1. Pre-trainings

Point-JEPA [15]. We pre-train a Transformer [19] encoder using the Point-JEPA method. The size of the encoder has the same hyper-parameters as ViT-S [6]. We use the exact same hyper-parameters and architecture for pre-training except floating-point precision (`float32` vs `float16`), and we apply a truncated normal as weight initialization function [5] with rescaling w.r.t. layer depth (GPT-2 layer-wise weight rescaling). Results can diverge a bit from the initial paper due to these additions. This model is called the *teacher* or *baseline*.

B.2. Distillations

Foundry. Fig. 1 in the main paper illustrates the core idea. We use the architecture described before with our pre-training. As said in the main paper, for the DSO block we use the Gumbel-Softmax [8] with a straight-through [1] estimator for differentiability. In contrast to 3DLST [12], we do not double the output channel of CAU for recovering the teacher token dimension.

Foundry-Gate. The Gate is a 2-layer MLP with a fixed hidden dimension of 128. The final gate activation is a sigmoid function to produce a probability for each token. When distilling or fine-tuning, no gradients from the target task loss are passed to the gate.

Specialists. For the specialists presented in Tab. 2 in main paper, we use fine-tuned models (pre-trained with Point-JEPA) on two datasets, ShapeNet55 (Specialist-CLS) and ShapeNetPart (Specialist-SEG). For each, we have a classical Transformer. We apply distillation only with KL divergence [9] without ground truth attachment to only use the teacher model as we present. We distill on their respective datasets to obtain a set of supertokens for compressing input tokens. We freeze the core encoder, tokenizer, and positional embedder but allow the heads to be updated during the process.

FPS-Student. The idea is to distill an existing backbone with a fixed number of centers for tokenizing the input point cloud to another fixed number of centers with its particular tokenizer. We apply the same distillation strategy as Foundry while changing the architecture so that the new student may be compatible with distillation. Instead of using one single grouping function with $c = 64$ and $k = 32$, we use a second tokenizer with another grouping function with $c_2 = 16$ and $k = 2 \cdot p/c_2$ with p being the number of the initial point cloud (we set 2 for adding overlap between groups). Note that the data augmentation is shared for the two branches as well as the first c sampled centers up to c_2 which are sampled by the Farthest Point Sampling (FPS) [7] algorithm. As the number of output tokens differs between teacher and student, we use a feature upsampler [13] to recover the last tokens of c by interpolating existing c_2 to c.

ToMe [2] and PiToMe [16]. We use our strategy to distill both methods. For obtaining 16 output tokens, we set $r_{\text{tome}} = [32, 16]$ which after the first attention block, removes 32 tokens and at the second, removes another 16. Same for PiToMe but using token ratio instead of a fixed number of tokens to end with. The ten last layers process only 16 tokens. When we have 1 single output token, $r_{\text{tome}} = [32, 16, 8, 4, 2, 1]$. We track the source to be able to place information of merged tokens. For merged tokens, we put information of the corresponding remaining token. We also set proportional attention to `True` to add a bias in the attention matrix based on already fused tokens. We track and add proportional bias during distillation and fine-tunings.

PatchMerger [14]. Once again, we use our strategy to distill the encoder with PatchMerger. For obtaining 16 output tokens, we use the default hyper-parameter by placing the module between the 6th and 7th Transformer layers. We track the source to be able to place information of merged tokens. For merged tokens, we put information of the corresponding remaining token. We track when distilling and fine-tuning.

B.3. Fine-tunings

Classification. We use the architecture and method defined in Point-JEPA for all models except KMeans-Student. For this method, we tokenize all training samples and use them for creating s prototypes with K-Means [10] clustering algorithm. After, we encode these prototypes into a frozen pre-trained backbone. Finally, we retrieve all tokens using the assignation matrix and we pool tokens by concatenating `mean+max` as Point2Vec [24] does for probing and classifying to obtain one single token to use as input for our head.

Part segmentation. We use the architecture and method defined in Point-JEPA.

Few-Shot Classification. We use the architecture and method defined in Point-JEPA.

Linear Probing. We use the architecture and method defined in Point-JEPA.

B.4. Inferences

Random Sampling. We use fine-tuned models and change at inference-time the number of tokens given to the encoder. We tokenize the input point cloud and obtain c centers. We randomly subsample c to c_{rs} which we set beforehand to match the number of supertokens s we want to match.

Larger group size. We use fine-tuned models and change at inference-time the c and k values. We reduce c, and k is obtained via the following calculation $k = 2 \cdot p/c$.

Foundry-Gate. For a fixed r value and dynamic setting with token-budget, we use fine-tuned models and we change r at inference-time without making any additional training. Specifically, when we budget the number of tokens to pass to the encoder, we select a specific r value to have $c - s$ tokens to merge into s supertokens maximum. We provide the code in Listing 1.

```
class BudgetAwareMLPTokenSelector(MLPTokenSelector):
    def __init__(self, token_budget: int, num_supertokens: int, embed_dim: int, hidden_dim:
        int = 128, act: Callable = nn.GELU()) -> None:
        super().__init__(embed_dim, hidden_dim, act)
```

```
        self.token_budget = token_budget
        self.num_supertokens = num_supertokens

    def _gate_forward(self, x: torch.Tensor) -> torch.Tensor:
        B, T, _ = x.shape

        h = self.act(self.fc1(x))
        logits = self.fc2(h)  # (B, T, 1)
        probs = torch.sigmoid(logits).view(B, T, 1)
        return probs

    def _select_ratio(self, probs: torch.Tensor):
        B, T, _ = probs.shape

        if T <= self.token_budget:
            return 1.0

        sorted_vals, _ = probs[:, :, 0].sort(dim=1, descending=True)

        num_to_select = T - self.token_budget + self.num_supertokens
        if num_to_select == T:
            fusion_ratio = probs.new_full((B,), -torch.finfo(probs.dtype).eps)
        else:
            fusion_ratio = sorted_vals[:, num_to_select+1]  # we put +1 because in forward
                we use > (if)

        return fusion_ratio.reshape(B, 1, 1).repeat(1, T, 1)

    def forward(self, x: torch.Tensor, inference: bool = False, **kwargs) ->
        MLPTokenSelectorOutput:
        _, T, _ = x.shape

        probs = self._gate_forward(x)
        fusion_ratio = self._select_ratio(probs)

        selection_hard = (probs > fusion_ratio).float()
        if inference: # non differentiable
            selection = selection_hard
        else:  # straight-through method
            selection = probs + (selection_hard - probs).detach()

        mask = (selection > fusion_ratio).squeeze(-1)  # (B, T), boolean
        seqlens_selected = mask.sum(dim=1)

        final_num_tokens = (T - seqlens_selected + self.num_supertokens).max().item()
        if final_num_tokens > self.token_budget:
            print("more tokens used than token budget, diff:", final_num_tokens - self.
                token_budget)

        outputs = MLPTokenSelectorOutput(
            weighted_tokens=x * selection + x * (1 - selection),
            selected_tokens=x[mask],
            selected_seqlens=seqlens_selected,
            unselected_tokens=x[~mask],
            unselected_seqlens=T - seqlens_selected,
            selected_indices=torch.nonzero(mask.flatten(), as_tuple=False).flatten(),
            unselected_indices=torch.nonzero((~mask).flatten(), as_tuple=False).flatten(),
            probs=probs,
```

```
            selection=selection,
            mask=mask,
        )

        return outputs

    @staticmethod
    def from_mlp_token_selector(gate: MLPTokenSelector, token_budget: int, num_supertokens:
        int) -> "BudgetAwareMLPTokenSelector":
        new_gate = BudgetAwareMLPTokenSelector(
            token_budget,
            num_supertokens,
            embed_dim=gate.embed_dim,
            hidden_dim=gate.hidden_dim,
            act=gate.act,
        )
        new_gate.load_state_dict(gate.state_dict())

        if gate.training:
            new_gate.train()

        return new_gate
```

Listing 1. **Implementation of the token-budgeted Foundry-Gate**. We use PyTorch as the deep learning framework. For quick prototyping, we implement only for one single input sample.

ToMe. We use fine-tuned models and change at inference-time the number of tokens at the encoder output. We choose r_{tome} accordingly to the desired final number of output tokens. We track source tokens only when resolving the part segmentation task.

C. Additional Results

C.1. Detailed Results

Foundry. Table 1 describes detailed results when fine-tuning the distilled backbone using different numbers of supertokens. The maximum number of epochs for classification and part segmentation tasks are 150 and 300 respectively. In addition, we traditionally unfreeze the backbone respectively at 100 and 0 epoch.

# supertokens	Frozen student	FT Unfreeze epoch	Distillation loss	Avg	MN40	OO3D	SONN- OBJ-BG	SONN- OBJ-ONLY	SONN- PB-T50-RS	SN55	SNP mIoU$_C$	SNP mIoU$_I$
baseline ($c = 64$)	-	100	-	-	93.02 $\pm$ 0.14	82.25 $\pm$ 0.37	91.84 $\pm$ 0.47	88.38 $\pm$ 0.41	86.05 $\pm$ 0.42	90.54 $\pm$ 0.14	-	-
baseline ($c = 64$)	-	0	-	-	-	-	-	-	-	-	83.91 $\pm$ 0.25	85.73 $\pm$ 0.15
baseline ($c = 64$)	-	-	-	87.72	93.02 $\pm$ 0.14	82.25 $\pm$ 0.37	91.84 $\pm$ 0.47	88.38 $\pm$ 0.41	86.05 $\pm$ 0.42	90.54 $\pm$ 0.14	83.91 $\pm$ 0.25	85.73 $\pm$ 0.15
Random sampling - inference from baseline												
$c = 16$	-	-	-	75.20	90.57 $\pm$ 0.41	65.19 $\pm$ 0.81	78.95 $\pm$ 1.19	72.70 $\pm$ 1.14	69.74 $\pm$ 0.91	87.80 $\pm$ 0.22	66.96 $\pm$ 0.71	69.71 $\pm$ 0.16
$c = 8$	-	-	-	63.45	83.72 $\pm$ 0.60	41.20 $\pm$ 1.35	67.40 $\pm$ 1.60	62.67 $\pm$ 1.47	55.78 $\pm$ 0.83	80.72 $\pm$ 0.36	56.47 $\pm$ 0.62	59.60 $\pm$ 0.19
$c = 4$	-	-	-	47.46	65.45 $\pm$ 0.80	19.74 $\pm$ 1.08	50.79 $\pm$ 0.69	45.30 $\pm$ 1.41	39.39 $\pm$ 0.72	62.51 $\pm$ 0.38	46.35 $\pm$ 0.46	50.11 $\pm$ 0.23
$c = 2$	-	-	-	32.10	39.32 $\pm$ 0.55	08.03 $\pm$ 0.52	35.51 $\pm$ 1.52	32.19 $\pm$ 1.78	27.07 $\pm$ 0.62	36.00 $\pm$ 0.29	37.11 $\pm$ 0.27	41.56 $\pm$ 0.10
$c = 1$	-	-	-	21.46	18.33 $\pm$ 0.66	03.82 $\pm$ 0.32	23.22 $\pm$ 1.55	21.31 $\pm$ 1.35	19.90 $\pm$ 0.68	15.54 $\pm$ 0.23	33.70 $\pm$ 0.30	35.89 $\pm$ 0.16
Updating group size to reduce number of generated tokens - inference from baseline												
$c = 16$	-	-	-	41.66	63.45 $\pm$ 0.47	13.29 $\pm$ 0.42	11.76 $\pm$ 0.16	22.24 $\pm$ 0.29	12.74 $\pm$ 0.17	59.99 $\pm$ 0.19	72.98 $\pm$ 0.21	76.84 $\pm$ 0.08
$c = 8$	-	-	-	26.36	20.57 $\pm$ 0.19	03.62 $\pm$ 0.19	09.29 $\pm$ 0.00	17.33 $\pm$ 0.36	09.37 $\pm$ 0.00	23.12 $\pm$ 0.18	61.36 $\pm$ 0.35	66.24 $\pm$ 0.13
$c = 4$	-	-	-	20.23	10.32 $\pm$ 0.25	00.66 $\pm$ 0.14	09.29 $\pm$ 0.00	15.44 $\pm$ 0.34	09.37 $\pm$ 0.00	12.65 $\pm$ 0.21	50.31 $\pm$ 0.36	53.80 $\pm$ 0.13
$c = 2$	-	-	-	15.12	04.79 $\pm$ 0.15	00.28 $\pm$ 0.10	09.29 $\pm$ 0.00	12.67 $\pm$ 0.36	09.37 $\pm$ 0.00	05.53 $\pm$ 0.11	37.08 $\pm$ 0.09	41.95 $\pm$ 0.08
$c = 1$	-	-	-	15.78	03.88 $\pm$ 0.19	00.47 $\pm$ 0.13	09.24 $\pm$ 0.63	15.85 $\pm$ 0.82	09.37 $\pm$ 0.00	04.25 $\pm$ 0.09	40.30 $\pm$ 0.44	42.90 $\pm$ 0.34
ToMe [2] - inference from baseline (no additional training needed)												
$s = 16$	-	-	-	73.80	89.10 $\pm$ 0.23	50.51 $\pm$ 0.39	82.19 $\pm$ 0.56	84.22 $\pm$ 0.62	73.87 $\pm$ 0.50	84.89 $\pm$ 0.24	63.09 $\pm$ 0.40	62.51 $\pm$ 0.08
$s = 8$	-	-	-	66.79	84.02 $\pm$ 0.38	33.54 $\pm$ 0.78	76.87 $\pm$ 1.30	79.88 $\pm$ 0.96	67.32 $\pm$ 0.62	79.68 $\pm$ 0.26	56.81 $\pm$ 0.43	56.23 $\pm$ 0.14
$s = 4$	-	-	-	59.88	74.92 $\pm$ 0.65	20.91 $\pm$ 0.59	71.57 $\pm$ 1.05	74.29 $\pm$ 0.54	61.52 $\pm$ 0.49	72.05 $\pm$ 0.30	51.59 $\pm$ 0.52	52.19 $\pm$ 0.12
$s = 2$	-	-	-	53.19	62.76 $\pm$ 0.44	11.21 $\pm$ 0.36	67.76 $\pm$ 1.03	68.49 $\pm$ 0.84	56.30 $\pm$ 0.62	61.66 $\pm$ 0.30	47.47 $\pm$ 0.45	49.88 $\pm$ 0.12
$s = 1$	-	-	-	46.15	48.35 $\pm$ 0.31	07.20 $\pm$ 0.20	61.20 $\pm$ 0.89	62.87 $\pm$ 0.95	51.69 $\pm$ 0.46	42.15 $\pm$ 0.18	45.36 $\pm$ 0.19	50.37 $\pm$ 0.09

Continued on next page

Table 1 – continued from previous page

*ToMe [2] - Trained using **FMD (Ours)** + finetuning*												
$s = 16$	✗	-	0.0658	86.52	93.07	80.95	90.02	87.44	83.66	89.84	82.41	84.76
$s = 1$	✗	-	0.1509	85.87	92.14	80.08	89.85	86.40	83.00	89.84	81.14	84.50
*PiToMe [16] - Trained using **FMD (Ours)** + finetuning*												
$s = 16$	✗	-	0.0643	86.50	92.83	81.97	89.33	87.95	83.66	89.88	81.77	84.65
$s = 1$	✗	-	0.1508	86.14	92.55	80.95	90.19	86.23	82.89	89.76	81.84	84.74
*PatchMerger [14] - Trained using **FMD (Ours)** + finetuning*												
$s = 16$	✗	-	0.0326	87.59	92.83	82.76	91.39	88.99	85.53	90.11	83.59	85.55
$s = 1$	✗	-	0.2811	87.14	92.18	82.13	90.71	88.12	85.05	89.71	83.64	85.54
Foundry (Ours)												
$s = 16$	✓	300	0.1006	-	-	-	-	-	-	-	81.55	84.59
$s = 16$	✓	150	0.1006	-	91.09	76.61	84.68	85.54	80.81	89.68	-	-
$s = 16$	✓	100	0.1006	-	91.73	77.48	85.20	84.17	80.29	89.67	-	-
$s = 16$	✓	0	0.1006	-	91.41	77.80	84.34	85.20	78.70	89.32	82.21	85.05
$s = 16$	✓	-	0.1006	84.33	91.41 $\pm$ 0.32	77.30 $\pm$ 0.61	84.74 $\pm$ 0.43	84.97 $\pm$ 0.72	79.93 $\pm$ 1.10	89.56 $\pm$ 0.21	81.88 $\pm$ 0.47	84.82 $\pm$ 0.32
$s = 16$	✗	300	0.0736	-	-	-	-	-	-	-	81.90	84.87
$s = 16$	✗	150	0.0736	-	91.69	77.72	86.06	86.75	80.43	89.88	-	-
$s = 16$	✗	100	0.0736	-	92.02	78.03	86.75	85.89	81.23	89.95	-	-
$s = 16$	✗	0	0.0736	-	91.53	77.64	85.89	86.23	79.42	89.78	81.85	84.77
$s = 16$	✗	-	0.0736	84.87	91.75 $\pm$ 0.25	77.80 $\pm$ 0.21	86.23 $\pm$ 0.46	86.29 $\pm$ 0.43	80.36 $\pm$ 0.90	89.87 $\pm$ 0.09	81.87 $\pm$ 0.04	84.82 $\pm$ 0.07
$s = 16$ (ViT-T)	✗	300	0.1040	-	-	-	-	-	-	-	81.85	84.49
$s = 16$ (ViT-T)	✗	150	0.1040	-	91.69	75.75	84.34	83.99	79.49	89.04	-	-
$s = 16$ (ViT-T)	✗	100	0.1040	-	91.77	76.22	83.30	85.89	79.46	88.56	-	-
$s = 16$ (ViT-T)	✗	0	0.1040	-	91.00	75.98	82.62	84.51	79.18	88.41	81.51	84.63
$s = 16$ (ViT-T)	✗	-	0.1040	83.75	91.49 $\pm$ 0.42	75.98 $\pm$ 0.24	83.42 $\pm$ 0.87	84.80 $\pm$ 0.98	79.38 $\pm$ 0.17	88.67 $\pm$ 0.33	81.68 $\pm$ 0.24	84.56 $\pm$ 0.10
$s = 8$	✓	300	0.0993	-	-	-	-	-	-	-	82.08	84.53
$s = 8$	✓	150	0.0993	-	91.69	76.69	85.37	84.51	81.05	89.48	-	-
$s = 8$	✓	100	0.0993	-	91.82	76.69	86.23	84.68	80.64	89.56	-	-
$s = 8$	✓	0	0.0993	-	91.61	76.06	85.20	86.23	79.15	89.23	81.60	84.60
$s = 8$	✓	-	0.0993	84.38	91.71 $\pm$ 0.10	76.48 $\pm$ 0.36	85.60 $\pm$ 0.55	85.14 $\pm$ 0.95	80.28 $\pm$ 1.00	89.42 $\pm$ 0.17	81.84 $\pm$ 0.34	84.57 $\pm$ 0.05
$s = 8$	✗	300	0.0749	-	-	-	-	-	-	-	82.09	84.86
$s = 8$	✗	150	0.0749	-	91.49	78.11	86.23	86.57	80.46	89.67	-	-
$s = 8$	✗	100	0.0749	-	91.86	77.17	85.89	84.85	81.19	89.81	-	-
$s = 8$	✗	0	0.0749	-	91.53	76.69	86.57	86.75	79.74	89.34	81.81	84.84
$s = 8$	✗	-	0.0749	84.76	91.63 $\pm$ 0.20	77.32 $\pm$ 0.72	86.23 $\pm$ 0.34	86.06 $\pm$ 1.05	80.46 $\pm$ 0.73	89.61 $\pm$ 0.24	81.95 $\pm$ 0.19	84.85 $\pm$ 0.01
$s = 4$	✓	300	0.0986	-	-	-	-	-	-	-	81.97	84.72
$s = 4$	✓	150	0.0986	-	91.09	77.24	84.51	84.85	79.53	89.41	-	-
$s = 4$	✓	100	0.0986	-	91.33	76.93	85.03	86.92	79.77	89.43	-	-
$s = 4$	✓	0	0.0986	-	91.33	76.69	83.82	86.06	79.53	88.95	81.71	84.61
$s = 4$	✓	-	0.0986	84.25	91.25 $\pm$ 0.14	76.96 $\pm$ 0.28	84.45 $\pm$ 0.60	85.94 $\pm$ 1.04	79.61 $\pm$ 0.14	89.26 $\pm$ 0.27	81.84 $\pm$ 0.18	84.67 $\pm$ 0.08
$s = 4$	✗	300	0.0788	-	-	-	-	-	-	-	81.57	84.76
$s = 4$	✗	150	0.0788	-	91.45	76.14	86.23	85.89	79.67	89.81	-	-
$s = 4$	✗	100	0.0788	-	91.41	77.87	87.09	86.40	80.95	89.74	-	-
$s = 4$	✗	0	0.0788	-	91.57	76.06	84.68	85.37	81.47	89.68	81.99	84.86
$s = 4$	✗	-	0.0788	84.64	91.48 $\pm$ 0.08	76.69 $\pm$ 1.02	86.00 $\pm$ 1.22	85.89 $\pm$ 0.52	80.70 $\pm$ 0.93	89.74 $\pm$ 0.06	81.78 $\pm$ 0.30	84.81 $\pm$ 0.07
$s = 2$	✓	300	0.0979	-	-	-	-	-	-	-	81.58	84.47
$s = 2$	✓	150	0.0979	-	92.06	77.09	83.82	83.30	79.84	89.26	-	-
$s = 2$	✓	100	0.0979	-	91.00	76.77	84.68	85.37	80.12	89.50	-	-
$s = 2$	✓	0	0.0979	-	91.49	77.40	85.03	84.85	79.98	89.27	82.10	84.63
$s = 2$	✓	-	0.0979	84.17	91.52 $\pm$ 0.53	77.09 $\pm$ 0.32	84.51 $\pm$ 0.62	84.51 $\pm$ 1.08	79.98 $\pm$ 0.14	89.34 $\pm$ 0.14	81.84 $\pm$ 0.36	84.55 $\pm$ 0.11
$s = 2$	✗	300	0.0795	-	-	-	-	-	-	-	82.13	84.50
$s = 2$	✗	150	0.0795	-	91.65	78.66	85.37	86.40	80.57	89.68	-	-
$s = 2$	✗	100	0.0795	-	92.02	78.74	85.71	86.40	81.16	89.67	-	-
$s = 2$	✗	0	0.0795	-	91.61	77.56	86.40	85.54	80.60	89.62	81.84	84.44
$s = 2$	✗	-	0.0795	84.86	91.76 $\pm$ 0.22	78.32 $\pm$ 0.66	85.83 $\pm$ 0.53	86.12 $\pm$ 0.50	80.78 $\pm$ 0.33	89.66 $\pm$ 0.04	81.98 $\pm$ 0.21	84.47 $\pm$ 0.04
$s = 1$	✓	300	0.0967	-	-	-	-	-	-	-	81.51	84.61
$s = 1$	✓	150	0.0967	-	91.94	76.93	84.68	85.37	79.77	89.14	-	-
$s = 1$	✓	100	0.0967	-	91.49	76.69	85.37	84.17	79.53	89.37	-	-
$s = 1$	✓	0	0.0967	-	91.37	77.95	84.34	85.54	79.11	89.35	81.40	84.47
$s = 1$	✓	-	0.0967	84.17	91.60 $\pm$ 0.30	77.19 $\pm$ 0.67	84.80 $\pm$ 0.53	85.03 $\pm$ 0.75	79.47 $\pm$ 0.33	89.29 $\pm$ 0.13	81.46 $\pm$ 0.08	84.54 $\pm$ 0.10
$s = 1$	✗	300	0.0792	-	-	-	-	-	-	-	81.66	84.56
$s = 1$	✗	150	0.0792	-	91.86	77.64	86.06	85.20	81.09	89.67	-	-
$s = 1$	✗	100	0.0792	-	91.57	77.72	83.65	87.09	80.40	89.69	-	-
$s = 1$	✗	0	0.0792	-	91.73	77.80	83.99	85.89	80.46	89.58	81.86	84.43
$s = 1$	✗	-	0.0792	84.58	91.72 $\pm$ 0.14	77.72 $\pm$ 0.08	84.57 $\pm$ 1.30	86.06 $\pm$ 0.96	80.65 $\pm$ 0.38	89.65 $\pm$ 0.06	81.76 $\pm$ 0.14	84.50 $\pm$ 0.09

Continued on next page

Table 1 – continued from previous page

Table 1. **Detailed finetuning results of Foundry.** All presented results are finetunings from a frozen or unfrozen backbone during distillation (no distillation stage for the baseline). For the classification task on ModelNet40 (MN40), OmniObject3D (OO3D), the three splits of ScanObjectNN (SONN-{PB-T50-RS, OBJ-BG, OBJ-ONLY}) and ShapeNet55 (SN55), the used metric is the top-1 accuracy on the validation set. For part segmentation with ShapeNetPart (SNP), we show for both category and instance mIoUs. Mean ± Std Dev over 10 runs for the baseline. "FT Unfreeze Epoch" denotes for the epoch when we completely unfreeze the ViT backbone during the finetuning stage ('-' in this column means aggregation over the results above). In columns with results, '-' stands for no running was performed. ToMe [2] and PiToMe [16] halve the number of tokens at each layer up to the specified number of tokens (supertokens, to simplify the table columns). For distilled ToMe, PiToMe and PatchMerger [14], the distilled encoder is unfrozen at 100 epochs for classification datasets and at 0 epochs for part segmentation.

Foundry-Gate. Similarly, we have Tab. 2 which details results when using the gate module. For simplifying the results visualization, Fig. 1 plots the metric values of fine-tuned models for each dataset at each gate regularization value.

λ_{gate}	Frozen student	FT Unfreeze epoch	Distillation loss	Avg	MN40	OO3D	SONN- OBJ-BG	SONN- OBJ-ONLY	SONN- PB-T50-RS	SN55	SNP mIoU_C	SNP mIoU_I
baseline	-	-	-	87.72	93.02 ± 0.14	82.25 ± 0.37	91.84 ± 0.47	88.38 ± 0.41	86.05 ± 0.42	90.54 ± 0.14	83.91 ± 0.25	85.73 ± 0.15
Foundry-Gate (Ours)												
0	✓	-	0.0191	-	-	-	-	-	-	-	-	-
0	✗	-	0.0152	-	-	-	-	-	-	-	-	-
10^{-15}	✓	300	0.0191	-	-	-	-	-	-	-	82.64	85.24
10^{-15}	✓	150	0.0191	-	93.03	79.37	89.67	87.95	83.10	90.30	-	-
10^{-15}	✓	100	0.0191	-	93.48	81.97	92.77	88.12	86.02	90.75	-	-
10^{-15}	✓	0	0.0191	-	92.22	83.23	90.53	88.30	84.84	90.31	83.77	85.63
10^{-15}	✓	-	0.0191	87.16	92.91 ± 0.64	81.52 ± 1.97	90.99 ± 1.60	88.12 ± 0.17	84.65 ± 1.47	90.45 ± 0.26	83.21 ± 0.79	85.44 ± 0.27
10^{-15}	✗	300	0.0142	-	-	-	-	-	-	-	82.76	85.06
10^{-15}	✗	150	0.0142	-	93.56	80.24	88.30	87.95	82.86	90.50	-	-
10^{-15}	✗	100	0.0142	-	93.19	83.23	92.25	89.67	86.12	90.55	-	-
10^{-15}	✗	0	0.0142	-	92.14	82.36	90.88	89.67	85.50	90.38	83.93	85.62
10^{-15}	✗	-	0.0142	87.31	92.96 ± 0.74	81.94 ± 1.54	90.48 ± 2.01	89.10 ± 0.99	84.83 ± 1.73	90.48 ± 0.09	83.35 ± 0.83	85.34 ± 0.39
10^{-14}	✓	300	0.0191	-	-	-	-	-	-	-	82.78	85.11
10^{-14}	✓	150	0.0191	-	93.31	79.61	90.71	88.47	83.07	90.36	-	-
10^{-14}	✓	100	0.0191	-	92.87	81.73	91.22	87.78	86.36	90.58	-	-
10^{-14}	✓	0	0.0191	-	92.59	82.28	90.36	87.95	85.60	90.24	83.66	85.42
10^{-14}	✓	-	0.0191	87.11	92.92 ± 0.37	81.21 ± 1.41	90.76 ± 0.43	88.07 ± 0.36	85.01 ± 1.73	90.39 ± 0.17	83.22 ± 0.62	85.26 ± 0.21
10^{-14}	✗	300	0.0141	-	-	-	-	-	-	-	82.55	85.12
10^{-14}	✗	150	0.0141	-	93.15	80.31	90.02	88.47	83.69	90.22	-	-
10^{-14}	✗	100	0.0141	-	93.15	82.60	90.71	89.16	85.91	90.52	-	-
10^{-14}	✗	0	0.0141	-	92.30	82.68	91.05	87.61	84.39	90.21	84.06	85.60
10^{-14}	✗	-	0.0141	87.17	92.87 ± 0.49	81.86 ± 1.34	90.59 ± 0.53	88.41 ± 0.78	84.66 ± 1.14	90.32 ± 0.18	83.30 ± 1.07	85.36 ± 0.34
10^{-13}	✓	300	0.0193	-	-	-	-	-	-	-	82.68	85.11
10^{-13}	✓	150	0.0193	-	92.95	80.08	89.67	87.09	83.21	90.43	-	-
10^{-13}	✓	100	0.0193	-	92.74	82.36	91.22	87.78	86.09	90.53	-	-
10^{-13}	✓	0	0.0193	-	92.018	81.65	91.05	87.09	85.36	90.08	83.88	85.50
10^{-13}	✓	-	0.0193	86.96	92.57 ± 0.49	81.36 ± 1.17	90.65 ± 0.85	87.32 ± 0.40	84.88 ± 1.50	90.35 ± 0.23	83.28 ± 0.85	85.30 ± 0.28
10^{-13}	✗	300	0.0141	-	-	-	-	-	-	-	82.85	85.07
10^{-13}	✗	150	0.0141	-	93.23	79.84	90.19	89.16	83.03	90.15	-	-
10^{-13}	✗	100	0.0141	-	93.07	82.68	90.02	88.12	85.95	90.47	-	-
10^{-13}	✗	0	0.0141	-	92.50	82.20	90.36	89.50	84.80	90.30	83.79	85.61
10^{-13}	✗	-	0.0141	87.15	92.94 ± 0.38	81.57 ± 1.52	90.19 ± 0.17	88.93 ± 0.72	84.59 ± 1.47	90.31 ± 0.16	83.32 ± 0.66	85.34 ± 0.39
10^{-12}	✓	300	0.0218	-	-	-	-	-	-	-	82.76	85.26
10^{-12}	✓	150	0.0218	-	93.19	79.61	89.33	87.95	82.17	89.71	-	-
10^{-12}	✓	100	0.0218	-	93.15	81.89	91.05	88.47	85.91	89.78	-	-
10^{-12}	✓	0	0.0218	-	91.69	81.89	89.85	88.64	74.98	89.34	83.97	85.76
10^{-12}	✓	-	0.0218	86.47	92.68 ± 0.85	81.13 ± 1.32	90.07 ± 0.88	88.35 ± 0.36	81.02 ± 5.55	89.61 ± 0.24	83.37 ± 0.86	85.51 ± 0.35
10^{-12}	✗	300	0.0157	-	-	-	-	-	-	-	82.64	84.99
10^{-12}	✗	150	0.0157	-	92.99	80.00	89.16	88.47	83.14	88.60	-	-
10^{-12}	✗	100	0.0157	-	93.15	82.36	91.74	88.81	84.84	90.07	-	-
10^{-12}	✗	0	0.0157	-	91.65	81.50	90.02	88.47	74.57	89.74	83.70	85.69
10^{-12}	✗	-	0.0157	86.45	92.60 ± 0.82	81.29 ± 1.20	90.30 ± 1.31	88.58 ± 0.20	80.85 ± 5.50	89.47 ± 0.77	83.17 ± 0.75	85.34 ± 0.49
$10^{-11.75}$	✓	300	0.024	-	-	-	-	-	-	-	82.69	85.23
$10^{-11.75}$	✓	150	0.024	-	92.95	78.90	88.64	87.78	82.41	87.16	-	-
$10^{-11.75}$	✓	100	0.024	-	92.46	81.02	92.25	87.09	85.25	87.96	-	-
$10^{-11.75}$	✓	0	0.024	-	89.06	82.05	90.02	88.64	74.36	87.84	83.82	85.58
$10^{-11.75}$	✓	-	0.024	85.91	91.49 ± 2.12	80.66 ± 1.61	90.30 ± 1.82	87.84 ± 0.78	80.67 ± 5.65	87.66 ± 0.43	83.25 ± 0.80	85.40 ± 0.25
$10^{-11.75}$	✗	300	0.0171	-	-	-	-	-	-	-	82.39	84.97

Continued on next page

Table 2 – continued from previous page

$10^{-11.75}$	✗	150	0.0171	-	92.71	79.92	89.85	88.81	82.03	88.45	-	-
$10^{-11.75}$	✗	100	0.0171	-	92.99	82.68	91.22	88.12	85.53	88.25	-	-
$10^{-11.75}$	✗	0	0.0171	-	88.94	81.26	90.19	88.64	77.65	88.91	83.95	85.78
$10^{-11.75}$	✗	-	0.0171	86.33	91.55 ± 2.26	81.29 ± 1.38	90.42 ± 0.72	88.53 ± 0.36	81.74 ± 3.95	88.54 ± 0.34	83.17 ± 1.11	85.37 ± 0.57
$10^{-11.5}$	✓	300	0.029	-	-	-	-	-	-	-	82.08	84.43
$10^{-11.5}$	✓	150	0.029	-	90.76	79.29	88.30	86.06	82.30	88.48	-	-
$10^{-11.5}$	✓	100	0.029	-	92.71	79.92	90.53	87.78	84.49	87.99	-	-
$10^{-11.5}$	✓	0	0.029	-	90.40	77.87	89.50	87.61	79.98	88.13	83.80	85.66
$10^{-11.50}$	✓	-	0.029	85.67	91.29 ± 1.24	79.03 ± 1.05	89.44 ± 1.12	87.15 ± 0.95	82.26 ± 2.26	88.20 ± 0.25	82.94 ± 1.22	85.05 ± 0.87
$10^{-11.5}$	✗	300	0.0198	-	-	-	-	-	-	-	82.00	84.66
$10^{-11.5}$	✗	150	0.0198	-	91.69	75.98	89.67	88.64	80.22	88.71	-	-
$10^{-11.5}$	✗	100	0.0198	-	92.02	81.50	91.91	89.33	78.14	88.46	-	-
$10^{-11.5}$	✗	0	0.0198	-	89.67	71.42	87.44	86.75	78.90	89.20	83.60	85.63
$10^{-11.50}$	✗	-	0.0198	85.15	91.13 ± 1.27	76.30 ± 5.05	89.67 ± 2.24	88.24 ± 1.34	79.09 ± 1.05	88.79 ± 0.38	82.80 ± 1.13	85.15 ± 0.68
$10^{-11.25}$	✓	300	0.039	-	-	-	-	-	-	-	82.02	84.75
$10^{-11.25}$	✓	150	0.039	-	89.34	78.11	89.16	87.44	81.40	88.36	-	-
$10^{-11.25}$	✓	100	0.039	-	88.29	81.50	88.47	87.78	79.91	88.75	-	-
$10^{-11.25}$	✓	0	0.039	-	90.60	74.02	86.23	87.44	79.91	88.41	82.29	85.00
$10^{-11.25}$	✓	-	0.039	84.84	89.41 ± 1.16	77.87 ± 3.75	87.95 ± 1.53	87.55 ± 0.20	80.41 ± 0.86	88.51 ± 0.21	82.16 ± 0.19	84.87 ± 0.18
$10^{-11.25}$	✗	300	0.0264	-	-	-	-	-	-	-	81.87	84.65
$10^{-11.25}$	✗	150	0.0264	-	89.63	78.11	89.85	88.12	78.24	88.55	-	-
$10^{-11.25}$	✗	100	0.0264	-	89.51	78.35	90.71	86.06	78.38	88.68	-	-
$10^{-11.25}$	✗	0	0.0264	-	90.64	74.49	84.51	85.37	80.64	88.72	82.85	85.11
$10^{-11.25}$	✗	-	0.0264	84.59	89.92 ± 0.62	76.98 ± 2.16	88.35 ± 3.36	86.52 ± 1.43	79.09 ± 1.34	88.65 ± 0.09	82.36 ± 0.69	84.88 ± 0.33
10^{-11}	✓	300	0.0619	-	-	-	-	-	-	-	81.99	84.70
10^{-11}	✓	150	0.0619	-	91.57	77.40	87.95	86.92	77.72	88.61	-	-
10^{-11}	✓	100	0.0619	-	90.11	73.46	88.12	87.95	76.86	88.27	-	-
10^{-11}	✓	0	0.0619	-	91.29	76.77	84.85	87.95	79.18	88.90	82.28	84.79
10^{-11}	✓	-	0.0619	84.36	90.99 ± 0.77	75.88 ± 2.11	86.98 ± 1.84	87.61 ± 0.60	77.92 ± 1.17	88.59 ± 0.31	82.14 ± 0.21	84.75 ± 0.06
10^{-11}	✗	300	0.0419	-	-	-	-	-	-	-	81.63	84.47
10^{-11}	✗	150	0.0419	-	90.60	76.77	87.26	83.65	80.74	88.84	-	-
10^{-11}	✗	100	0.0419	-	90.76	77.40	83.99	86.57	80.01	88.91	-	-
10^{-11}	✗	0	0.0419	-	91.49	76.54	86.40	86.57	80.15	88.90	81.84	84.70
10^{-11}	✗	-	0.0419	84.36	90.95 ± 0.47	76.90 ± 0.45	85.89 ± 1.70	85.60 ± 1.69	80.30 ± 0.39	88.88 ± 0.04	81.74 ± 0.15	84.58 ± 0.16
$10^{-10.75}$	✓	300	0.0955	-	-	-	-	-	-	-	81.65	84.53
$10^{-10.75}$	✓	150	0.0955	-	91.00	75.04	85.54	86.23	78.31	89.02	-	-
$10^{-10.75}$	✓	100	0.0955	-	91.13	78.82	88.64	86.92	77.41	89.09	-	-
$10^{-10.75}$	✓	0	0.0955	-	91.65	77.80	85.89	87.61	80.01	89.03	81.67	84.89
$10^{-10.75}$	✓	-	0.0955	84.51	91.26 ± 0.34	77.22 ± 1.95	86.69 ± 1.70	86.92 ± 0.69	78.58 ± 1.32	89.04 ± 0.04	81.66 ± 0.02	84.71 ± 0.25
$10^{-10.75}$	✗	300	0.0672	-	-	-	-	-	-	-	81.73	84.77
$10^{-10.75}$	✗	150	0.0672	-	91.17	76.69	85.71	86.57	81.02	89.68	-	-
$10^{-10.75}$	✗	100	0.0672	-	92.06	78.74	86.75	88.12	81.26	90.17	-	-
$10^{-10.75}$	✗	0	0.0672	-	91.29	77.64	87.95	86.92	80.08	89.07	81.97	84.95
$10^{-10.75}$	✗	-	0.0672	85.04	91.50 ± 0.48	77.69 ± 1.02	86.80 ± 1.12	87.21 ± 0.81	80.79 ± 0.62	89.64 ± 0.55	81.85 ± 0.17	84.86 ± 0.13
$10^{-10.5}$	✓	300	0.1013	-	-	-	-	-	-	-	81.73	84.52
$10^{-10.5}$	✓	150	0.1013	-	91.57	77.09	84.51	84.34	79.01	88.95	-	-
$10^{-10.5}$	✓	100	0.1013	-	91.17	78.11	84.68	85.37	78.73	89.35	-	-
$10^{-10.5}$	✓	0	0.1013	-	90.92	77.80	85.54	84.51	70.37	89.26	82.13	84.87
$10^{-10.50}$	✓	-	0.1013	83.80	91.22 ± 0.33	77.66 ± 0.52	84.91 ± 0.55	84.74 ± 0.55	76.04 ± 4.91	89.19 ± 0.21	81.93 ± 0.29	84.70 ± 0.25
$10^{-10.5}$	✗	300	0.0753	-	-	-	-	-	-	-	82.27	85.00
$10^{-10.5}$	✗	150	0.0753	-	91.21	76.61	87.95	86.75	80.33	89.88	-	-
$10^{-10.5}$	✗	100	0.0753	-	91.53	77.56	87.09	87.61	80.88	89.87	-	-
$10^{-10.5}$	✗	0	0.0753	-	90.72	77.09	86.40	87.09	78.45	89.16	81.88	84.79
$10^{-10.50}$	✗	-	0.0753	84.88	91.15 ± 0.41	77.09 ± 0.47	87.15 ± 0.78	87.15 ± 0.43	79.89 ± 1.27	89.64 ± 0.41	82.08 ± 0.27	84.89 ± 0.15
$10^{-10.25}$	✓	300	0.0996	-	-	-	-	-	-	-	81.80	84.53
$10^{-10.25}$	✓	150	0.0996	-	91.00	76.30	83.48	84.68	79.35	89.31	-	-
$10^{-10.25}$	✓	100	0.0996	-	90.96	77.80	84.51	84.85	79.15	89.29	-	-
$10^{-10.25}$	✓	0	0.0996	-	91.13	77.40	83.82	84.51	79.04	89.08	82.36	84.85
$10^{-10.25}$	✓	-	0.0996	84.00	91.03 ± 0.08	77.17 ± 0.78	83.94 ± 0.53	84.68 ± 0.17	79.18 ± 0.16	89.22 ± 0.13	82.08 ± 0.39	84.69 ± 0.23
$10^{-10.25}$	✗	300	0.0754	-	-	-	-	-	-	-	82.37	85.01
$10^{-10.25}$	✗	150	0.0754	-	91.90	76.93	85.37	86.75	80.57	89.89	-	-
$10^{-10.25}$	✗	100	0.0754	-	91.53	77.40	85.20	85.54	80.88	89.98	-	-
$10^{-10.25}$	✗	0	0.0754	-	91.37	76.46	86.06	86.06	80.95	89.79	82.49	84.91
$10^{-10.25}$	✗	-	0.0754	84.78	91.60 ± 0.27	76.93 ± 0.47	85.54 ± 0.46	86.12 ± 0.60	80.80 ± 0.20	89.89 ± 0.10	82.43 ± 0.08	84.96 ± 0.08
10^{-10}	✓	300	0.0981	-	-	-	-	-	-	-	82.04	84.68
10^{-10}	✓	150	0.0981	-	91.41	76.69	83.65	85.37	79.49	89.41	-	-
10^{-10}	✓	100	0.0981	-	91.33	76.46	83.99	84.85	79.25	89.58	-	-
10^{-10}	✓	0	0.0981	-	90.56	75.75	83.48	84.85	78.73	88.97	82.22	84.67
10^{-10}	✓	-	0.0981	83.93	91.10 ± 0.47	76.30 ± 0.49	83.71 ± 0.26	85.03 ± 0.30	79.16 ± 0.39	89.32 ± 0.31	82.13 ± 0.13	84.67 ± 0.01
10^{-10}	✗	300	0.0763	-	-	-	-	-	-	-	82.05	84.70
10^{-10}	✗	150	0.0763	-	91.41	77.64	86.75	85.03	80.43	89.68	-	-

Continued on next page

Table 2 – continued from previous page

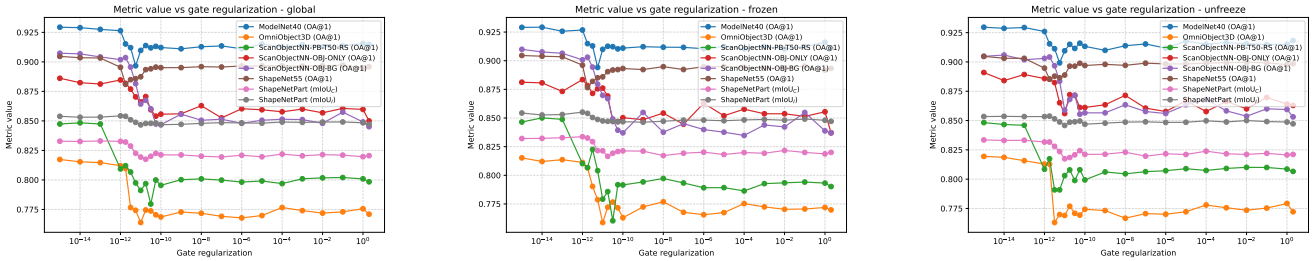
10^{-10}	✗	100	0.0763	-	91.13	77.87	85.20	86.75	80.71	89.95	-	-
10^{-10}	✗	0	0.0763	-	91.45	76.77	85.03	86.57	78.66	89.46	82.18	84.67
10^{-10}	✗	-	0.0763	84.62	91.33 ± 0.18	77.43 ± 0.58	85.66 ± 0.95	86.12 ± 0.95	79.93 ± 1.11	89.70 ± 0.25	82.12 ± 0.09	84.69 ± 0.02
10^{-9}	✓	300	0.0984	-	-	-	-	-	-	-	82.20	84.56
10^{-9}	✓	150	0.0984	-	91.21	77.09	85.54	85.54	79.98	89.26	-	-
10^{-9}	✓	100	0.0984	-	91.17	77.32	85.37	84.85	78.80	89.43	-	-
10^{-9}	✓	0	0.0984	-	91.29	77.32	85.54	84.17	79.49	89.00	82.02	84.71
10^{-9}	✓	-	0.0984	84.27	91.22 ± 0.06	77.24 ± 0.14	85.48 ± 0.10	84.85 ± 0.69	79.42 ± 0.59	89.23 ± 0.22	82.11 ± 0.12	84.64 ± 0.10
10^{-9}	✗	300	0.0774	-	-	-	-	-	-	-	82.23	84.75
10^{-9}	✗	150	0.0774	-	91.29	76.85	85.20	86.57	80.81	89.96	-	-
10^{-9}	✗	100	0.0774	-	91.09	77.32	86.06	86.06	80.53	89.79	-	-
10^{-9}	✗	0	0.0774	-	90.60	77.80	85.71	86.40	80.50	89.64	82.02	84.82
10^{-9}	✗	-	0.0774	84.71	90.99 ± 0.35	77.32 ± 0.47	85.66 ± 0.43	86.35 ± 0.26	80.62 ± 0.17	89.80 ± 0.16	82.13 ± 0.15	84.78 ± 0.05
10^{-8}	✓	300	0.0987	-	-	-	-	-	-	-	81.73	84.62
10^{-8}	✓	150	0.0987	-	91.41	78.43	84.34	85.37	79.94	89.53	-	-
10^{-8}	✓	100	0.0987	-	91.13	77.01	83.48	85.54	80.19	89.65	-	-
10^{-8}	✓	0	0.0987	-	91.00	77.64	83.48	85.37	79.04	89.26	81.71	84.81
10^{-8}	✓	-	0.0987	84.21	91.18 ± 0.21	77.69 ± 0.71	83.76 ± 0.50	85.43 ± 0.10	79.72 ± 0.60	89.48 ± 0.20	81.72 ± 0.01	84.72 ± 0.13
10^{-8}	✗	300	0.0760	-	-	-	-	-	-	-	82.35	84.79
10^{-8}	✗	150	0.0760	-	91.09	76.22	86.75	85.37	80.46	89.72	-	-
10^{-8}	✗	100	0.0760	-	91.61	77.32	86.06	86.92	80.29	89.83	-	-
10^{-8}	✗	0	0.0760	-	91.45	76.46	86.23	89.16	80.60	89.58	82.26	84.94
10^{-8}	✗	-	0.0760	84.86	91.38 ± 0.27	76.67 ± 0.58	86.35 ± 0.36	87.15 ± 1.90	80.45 ± 0.16	89.71 ± 0.12	82.30 ± 0.06	84.87 ± 0.11
10^{-7}	✓	300	0.0999	-	-	-	-	-	-	-	81.95	84.72
10^{-7}	✓	150	0.0999	-	91.33	76.61	85.03	84.34	79.15	89.26	-	-
10^{-7}	✓	100	0.0999	-	91.29	76.77	84.51	84.51	80.15	89.44	-	-
10^{-7}	✓	0	0.0999	-	90.88	76.93	83.99	84.51	78.70	89.00	81.91	84.91
10^{-7}	✓	-	0.0999	84.03	91.17 ± 0.25	76.77 ± 0.16	84.51 ± 0.52	84.45 ± 0.10	79.33 ± 0.75	89.23 ± 0.22	81.93 ± 0.03	84.81 ± 0.13
10^{-7}	✗	300	0.0749	-	-	-	-	-	-	-	82.05	84.89
10^{-7}	✗	150	0.0749	-	91.45	77.17	85.89	85.89	80.40	89.97	-	-
10^{-7}	✗	100	0.0749	-	91.61	77.48	85.89	86.57	80.71	90.00	-	-
10^{-7}	✗	0	0.0749	-	91.53	76.54	85.54	85.71	80.81	89.72	81.88	84.89
10^{-7}	✗	-	0.0749	84.73	91.53 ± 0.08	77.06 ± 0.48	85.77 ± 0.20	86.06 ± 0.46	80.64 ± 0.22	89.90 ± 0.15	81.96 ± 0.12	84.89 ± 0.00
10^{-6}	✓	300	0.0997	-	-	-	-	-	-	-	81.67	84.71
10^{-6}	✓	150	0.0997	-	91.37	76.54	83.99	87.61	78.94	89.68	-	-
10^{-6}	✓	100	0.0997	-	90.84	77.01	84.34	85.71	78.87	89.55	-	-
10^{-6}	✓	0	0.0997	-	90.92	76.14	83.65	85.54	78.94	89.20	82.37	84.93
10^{-6}	✓	-	0.0997	84.14	91.05 ± 0.28	76.56 ± 0.43	83.99 ± 0.34	86.29 ± 1.15	78.92 ± 0.04	89.48 ± 0.25	82.02 ± 0.49	84.82 ± 0.15
10^{-6}	✗	300	0.0765	-	-	-	-	-	-	-	82.28	84.80
10^{-6}	✗	150	0.0765	-	91.21	77.24	85.89	85.20	80.67	89.95	-	-
10^{-6}	✗	100	0.0765	-	91.61	77.01	85.37	86.06	81.05	90.03	-	-
10^{-6}	✗	0	0.0765	-	90.72	76.77	85.54	86.06	80.43	89.51	82.05	84.88
10^{-6}	✗	-	0.0765	84.64	91.18 ± 0.45	77.01 ± 0.24	85.60 ± 0.26	85.77 ± 0.50	80.72 ± 0.31	89.83 ± 0.28	82.16 ± 0.17	84.84 ± 0.05
10^{-5}	✓	300	0.1000	-	-	-	-	-	-	-	81.62	84.64
10^{-5}	✓	150	0.1000	-	91.37	77.32	83.65	85.89	79.39	89.67	-	-
10^{-5}	✓	100	0.1000	-	90.96	76.77	83.65	83.82	79.01	89.68	-	-
10^{-5}	✓	0	0.1000	-	91.37	76.14	83.99	85.29	78.30	89.13	82.02	84.88
10^{-5}	✓	-	0.1000	83.99	91.23 ± 0.23	76.75 ± 0.59	83.76 ± 0.20	85.20 ± 1.19	78.93 ± 0.51	89.49 ± 0.31	81.82 ± 0.28	84.76 ± 0.17
10^{-5}	✗	300	0.0756	-	-	-	-	-	-	-	82.25	84.86
10^{-5}	✗	150	0.0756	-	91.90	77.17	86.92	86.57	80.81	89.92	-	-
10^{-5}	✗	100	0.0756	-	91.73	77.24	86.57	86.50	81.16	89.91	-	-
10^{-5}	✗	0	0.0756	-	91.69	77.24	85.54	86.75	80.71	89.66	81.96	84.85
10^{-5}	✗	-	0.0756	84.96	91.77 ± 0.11	77.22 ± 0.05	86.35 ± 0.72	86.69 ± 0.10	80.89 ± 0.24	89.83 ± 0.15	82.10 ± 0.20	84.85 ± 0.01
10^{-4}	✓	300	0.1000	-	-	-	-	-	-	-	82.03	84.63
10^{-4}	✓	150	0.1000	-	91.21	78.03	82.44	85.89	78.70	89.57	-	-
10^{-4}	✓	100	0.1000	-	90.84	76.69	84.51	85.89	78.73	89.50	-	-
10^{-4}	✓	0	0.1000	-	91.05	77.87	83.48	85.54	78.49	89.20	81.95	85.03
10^{-4}	✓	-	0.1000	84.09	91.03 ± 0.18	77.53 ± 0.73	83.48 ± 1.03	85.77 ± 0.20	78.64 ± 0.13	89.42 ± 0.20	81.99 ± 0.06	84.83 ± 0.28
10^{-4}	✗	300	0.0757	-	-	-	-	-	-	-	82.29	84.91
10^{-4}	✗	150	0.0757	-	91.45	78.35	86.23	86.75	80.26	89.81	-	-
10^{-4}	✗	100	0.0757	-	91.37	77.64	86.75	85.37	81.33	90.06	-	-
10^{-4}	✗	0	0.0757	-	91.73	77.40	87.44	85.20	80.64	89.81	82.49	85.05
10^{-4}	✗	-	0.0757	84.99	91.52 ± 0.19	77.80 ± 0.49	86.80 ± 0.60	85.77 ± 0.85	80.74 ± 0.55	89.89 ± 0.15	82.39 ± 0.14	84.98 ± 0.10
10^{-3}	✓	300	0.1000	-	-	-	-	-	-	-	81.81	84.69
10^{-3}	✓	150	0.1000	-	91.25	77.01	85.37	84.51	79.46	89.40	-	-
10^{-3}	✓	100	0.1000	-	91.41	76.85	84.34	86.06	79.35	89.60	-	-
10^{-3}	✓	0	0.1000	-	91.05	77.87	83.48	85.54	79.01	89.00	82.04	85.06
10^{-3}	✓	-	0.1000	84.20	91.23 ± 0.18	77.24 ± 0.55	84.39 ± 0.95	85.37 ± 0.79	79.27 ± 0.24	89.33 ± 0.30	81.93 ± 0.16	84.87 ± 0.26
10^{-3}	✗	300	0.0761	-	-	-	-	-	-	-	82.42	84.89
10^{-3}	✗	150	0.0761	-	92.02	77.87	86.23	87.09	80.81	90.03	-	-
10^{-3}	✗	100	0.0761	-	91.33	78.19	85.20	87.26	81.92	90.00	-	-

Continued on next page

Table 2 – continued from previous page

10^{-3}	✗	0	0.0761	-	91.61	76.61	86.06	85.54	80.01	89.79	81.90	84.87
10^{-3}	✗	-	0.0761	84.95	91.65 ± 0.35	77.56 ± 0.83	85.83 ± 0.55	86.63 ± 0.95	80.92 ± 0.96	89.94 ± 0.13	82.16 ± 0.37	84.88 ± 0.01
10^{-2}	✓	300	0.1000	-	-	-	-	-	-	-	82.14	84.69
10^{-2}	✓	150	0.1000	-	91.00	77.24	85.20	85.37	78.80	89.51	-	-
10^{-2}	✓	100	0.1000	-	91.09	77.40	83.99	86.23	79.77	89.59	-	-
10^{-2}	✓	0	0.1000	-	90.96	76.46	83.48	84.51	79.46	89.03	82.20	84.92
10^{-2}	✓	-	0.1000	84.17	91.02 ± 0.06	77.03 ± 0.51	84.22 ± 0.88	85.37 ± 0.86	79.34 ± 0.50	89.38 ± 0.30	82.17 ± 0.05	84.81 ± 0.16
10^{-2}	✗	300	0.0759	-	-	-	-	-	-	-	82.31	85.04
10^{-2}	✗	150	0.0759	-	91.77	77.17	85.03	85.37	80.43	89.94	-	-
10^{-2}	✗	100	0.0759	-	91.65	77.95	85.71	85.89	82.10	90.28	-	-
10^{-2}	✗	0	0.0759	-	91.25	76.93	85.37	86.75	80.50	89.69	81.92	84.97
10^{-2}	✗	-	0.0759	84.80	91.56 ± 0.28	77.35 ± 0.54	85.37 ± 0.34	86.00 ± 0.70	81.01 ± 0.94	89.97 ± 0.30	82.11 ± 0.27	85.00 ± 0.05
10^{-1}	✓	300	0.1000	-	-	-	-	-	-	-	81.95	84.78
10^{-1}	✓	150	0.1000	-	91.37	77.01	85.03	84.85	78.87	89.58	-	-
10^{-1}	✓	100	0.1000	-	91.45	77.56	86.06	84.85	79.98	89.62	-	-
10^{-1}	✓	0	0.1000	-	90.84	76.61	85.37	85.71	79.39	89.36	82.04	85.04
10^{-1}	✓	-	0.1000	84.34	91.22 ± 0.33	77.06 ± 0.47	85.48 ± 0.53	85.14 ± 0.50	79.41 ± 0.56	89.52 ± 0.14	82.00 ± 0.06	84.91 ± 0.18
10^{-1}	✗	300	0.0753	-	-	-	-	-	-	-	82.28	85.02
10^{-1}	✗	150	0.0753	-	91.41	78.11	86.23	86.92	80.99	89.76	-	-
10^{-1}	✗	100	0.0753	-	91.37	77.24	85.20	87.09	81.33	90.13	-	-
10^{-1}	✗	0	0.0753	-	91.65	77.24	86.57	86.92	80.67	89.89	82.13	84.80
10^{-1}	✗	-	0.0753	85.00	91.48 ± 0.15	77.53 ± 0.50	86.00 ± 0.72	86.98 ± 0.10	81.00 ± 0.33	89.93 ± 0.19	82.20 ± 0.10	84.91 ± 0.16
1	✓	300	0.0997	-	-	-	-	-	-	-	81.79	84.76
1	✓	150	0.0997	-	92.10	77.48	85.03	85.71	79.04	89.05	-	-
1	✓	100	0.0997	-	91.17	77.09	83.48	84.68	79.98	89.67	-	-
1	✓	0	0.0997	-	91.57	77.01	83.13	86.23	78.94	88.81	81.95	84.85
1	✓	-	0.0997	84.17	91.61 ± 0.47	77.19 ± 0.25	83.88 ± 1.01	85.54 ± 0.79	79.32 ± 0.57	89.17 ± 0.44	81.87 ± 0.11	84.80 ± 0.06
1	✗	300	0.0765	-	-	-	-	-	-	-	82.19	84.89
1	✗	150	0.0765	-	91.65	78.11	86.57	86.57	80.33	90.00	-	-
1	✗	100	0.0765	-	91.57	77.87	85.71	86.23	81.40	89.89	-	-
1	✗	0	0.0765	-	91.29	77.80	85.54	86.40	80.85	89.42	81.95	84.86
1	✗	-	0.0765	84.92	91.50 ± 0.19	77.93 ± 0.16	85.94 ± 0.55	86.40 ± 0.17	80.86 ± 0.54	89.77 ± 0.31	82.07 ± 0.17	84.88 ± 0.02
2	✓	300	0.0992	-	-	-	-	-	-	-	82.07	84.78
2	✓	150	0.0992	-	91.17	76.69	84.51	83.13	79.11	89.41	-	-
2	✓	100	0.0992	-	90.96	77.09	83.13	84.51	79.46	89.51	-	-
2	✓	0	0.0992	-	91.17	77.17	83.48	83.48	78.49	89.07	81.93	84.65
2	✓	-	0.0992	83.82	91.10 ± 0.12	76.98 ± 0.25	83.71 ± 0.72	83.71 ± 0.72	79.02 ± 0.49	89.33 ± 0.23	82.00 ± 0.11	84.71 ± 0.09
2	✗	300	0.0770	-	-	-	-	-	-	-	82.17	84.74
2	✗	150	0.0770	-	92.22	77.01	84.51	86.57	80.19	89.82	-	-
2	✗	100	0.0770	-	91.49	77.01	86.23	85.89	80.74	89.93	-	-
2	✗	0	0.0770	-	91.77	77.64	85.20	86.40	81.05	89.78	82.05	84.70
2	✗	-	0.0770	84.75	91.83 ± 0.37	77.22 ± 0.36	85.31 ± 0.87	86.29 ± 0.36	80.66 ± 0.44	89.84 ± 0.08	82.11 ± 0.08	84.72 ± 0.03

Table 2. **Detailed finetuning results of Foundry-Gate.** All presented results are finetunings from a frozen or unfrozen backbone during distillation using 16 supertokens. During distillation and finetunings, no weight decay is applied on the gate module. For the classification task on ModelNet40 (MN40), OmniObject3D (OO3D), the three splits of ScanObjectNN (SONN-{`PB-T50-RS`, `OBJ-BG`, `OBJ-ONLY`}) and ShapeNet55 (SN55), the used metric is the top-1 accuracy on the validation set. For part segmentation with ShapeNetPart (SNP), we show for both category and instance mIoUs. Mean ± Std Dev over 10 runs for the baseline, 3 for the classification task, and 2 for the part segmentation task.



(a) All distillation type (b) Frozen encoder during distillation (c) Unfrozen encoder during distillation

Figure 1. **Fine-tuning results vs λ_{gate}.** No weight decay used on the gate module during distillation are shown. All metrics have been computed on the validation set for each dataset.

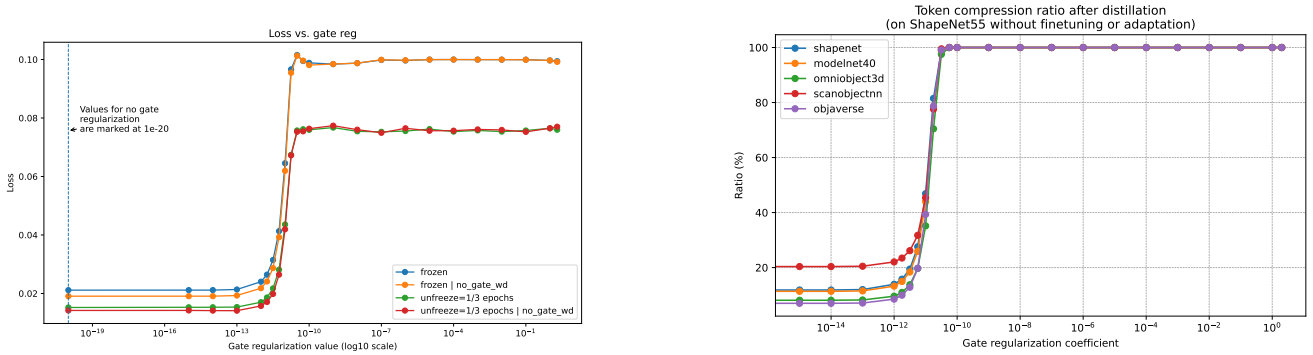
Table 3 shows the Spearman correlation values for each dataset between final metrics and gate regularization as a single global value. We observe an inverse correlation between performances and gate regularization. The more we regularize for

compressing tokens, the less downstream performance is obtained.

Frozen student	MN40 OA@1	OO3D OA@1	SONN-OBJ-BG OA@1	SONN-OBJ-ONLY OA@1	SONN-PB-T50-RS OA@1	SN55 OA@1	SNP mIoU$_C$	SNP mIoU$_I$
✓	-0.4542	-0.6583	-0.5198	-0.8024	-0.6953	-0.0257	-0.6630	-0.4397
✗	-0.1339	-0.2451	-0.0934	-0.7556	-0.6225	0.1948	-0.5702	-0.5099
All	-0.1897	-0.5278	-0.3330	-0.8103	-0.7359	0.1472	-0.5741	-0.4091

Table 3. **Correlation between downstream performances and gate regularization of Foundry-Gate.** All presented results are finetunings from a frozen or unfrozen backbone during distillation using 16 supertokens. We compute the Spearman correlation on ModelNet40 (MN40), OmniObject3D (OO3D), the three splits of ScanObjectNN (SONN-{PB-T50-RS, OBJ-BG, OBJ-ONLY}), ShapeNet55 (SN55) and ShapeNetPart (SNP) between the mean of the considered metrics for each dataset on the validation set and λ_{gate}.

Figure 2 presents the content distillation loss of distilled models for each gate regularization value with specification when we freeze/unfreeze the model during distillation and when we add weight decay on gate. For $\lambda_{\text{gate}} \in [10^{-12}, 10^{-10}]$, we observe an increase in the token compression ratio, leading to 100% of the tokens selected by the gate being merged. We also observe an increase in the distillation loss. By combining these two observations, we can state that the distillation loss has a correlation with the token compression ratio. Additionally, with the help of Tab. 3, by transitivity, the distillation loss is also correlated with downstream task performances.

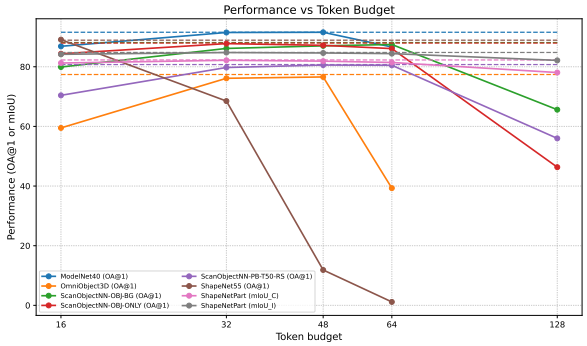


(a) Distillation losses.

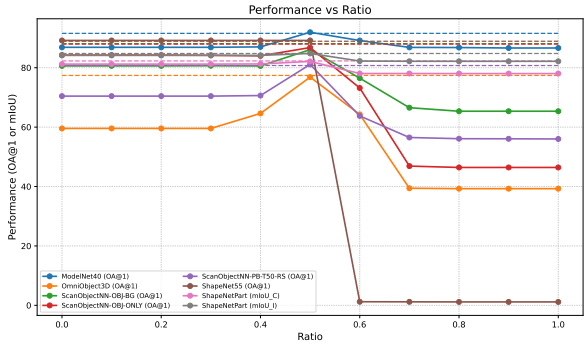
(b) Token compression ratio (frozen backbone).

Figure 2. **Effect of gate regularization λ_{gate} on distillation stability.** Distillation loss and token compression ratio curves for frozen and unfrozen backbones across a wide range of λ_{gate}. This figure shows the relationship between the distillation loss, token compression rate and gate regularization. (Left) Several distillation configurations are compared, frozen/unfrozen, add/remove weight decay on the gate module. We provide also token fusion ratios for the validation set on Objaverse [4] for additional results on an unannotated dataset with more diverse objects. (Right) For $\lambda_{\text{gate}} \in [10^{-12}, 10^{-10}]$, we observe the token compression ratio is increasing to obtain 100% of tokens selected by the gate for being merged.

Budget-Aware Inference with Selective Compression. We budgeted the number of tokens that can be processed by the core encoder by, (i) fixing a maximum number of tokens that the encoder can take, (ii) changing the fusion ratio. These two techniques are used at inference-time and using the output probabilities π_i of the gate. Figure 3a illustrates the first technique. By limiting the number of tokens, we see different behavior. For the ShapeNet55 dataset, it is at its maximum when all tokens are fused. In contrast, for all others the maximum are when we don't fuse between 50/62.5 to 75/87.5% of the input tokens (for 64/128 input tokens). Figure 3b shows the results for the second technique at inference-time. We observe that the best ratio value is for the value used for distilling and fine-tuning the model. We observe a large downfall for ShapeNet55, larger than for any other dataset. We think that it comes from the fact that it was used during pre-training, distillation and fine-tuning and by fixing a r, it does not succeed in inferring the inputs we give. As a hypothesis, we believe that during distillation and fine-tuning, in order to make the ratio more easily changeable at inference time, choosing a random fusion ratio could help models generalize to any value. Table 4 contains all plotted values.



(a) Inference results vs token budget for Foundry-Gate.



(b) Inference results vs fusion ratio for Foundry-Gate.

Figure 3. **Inference results for Foundry-Gate.** We use the best backbone on each dataset with $\lambda_{\text{gate}} = 10^{-11}$ and without weight decay. r represents the fusion ratio threshold. All metrics have been computed on the validation set for each dataset.

Token budget	Foundry-Gate r	Avg	MN40	OO3D	SONN-OBJ-BG	SONN-OBJ-ONLY	SONN-PB-T50-RS	SN55	SNP mIoU$_C$	SNP mIoU$_I$
Foundry-Gate with $\lambda_{gate} = 10^{-11}$ baseline (Ours)										
-	-	84.36	90.99 ± 0.77	75.88 ± 2.11	86.98 ± 1.84	87.61 ± 0.60	77.92 ± 1.17	88.59 ± 0.31	82.14 ± 0.21	84.75 ± 0.06
Foundry-Gate (Ours) - Inference token-budgeted										
16	-	79.44	86.86 ± 0.22	59.49 ± 0.59	79.91 ± 0.59	84.34 ± 0.82	70.42 ± 0.43	89.09 ± 0.12	81.25 ± 0.13	84.17 ± 0.06
32	-	82.09	91.49 ± 0.37	76.13 ± 0.69	86.14 ± 0.57	87.76 ± 0.56	79.75 ± 0.39	68.50 ± 0.19	82.17 ± 0.11	84.77 ± 0.05
48	-	75.16	91.56 ± 0.24	76.58 ± 0.38	87.01 ± 0.74	87.18 ± 0.65	80.57 ± 0.32	11.86 ± 0.13	81.88 ± 0.13	84.62 ± 0.05
64	-	68.37	86.63 ± 0.35	39.30 ± 0.26	87.47 ± 0.61	86.09 ± 0.44	80.50 ± 0.33	01.15 ± 0.00	81.44 ± 0.14	84.41 ± 0.05
80	-	-	-	-	-	-	-	-	80.89 ± 0.13	83.91 ± 0.06
96	-	-	-	-	-	-	-	-	80.07 ± 0.17	83.33 ± 0.05
112	-	-	-	-	-	-	-	-	79.17 ± 0.17	82.67 ± 0.06
128	-	-	-	-	65.61 ± 0.46	46.33 ± 0.53	55.99 ± 0.42	-	78.06 ± 0.11	82.14 ± 0.05
Foundry (Ours) - change fusion ratio r										
-	0.0	79.54	86.87 ± 0.38	59.56 ± 0.72	80.62 ± 0.60	84.25 ± 0.78	70.44 ± 0.41	89.18 ± 0.13	81.21 ± 0.14	84.15 ± 0.05
-	0.1	79.54	86.87 ± 0.38	59.56 ± 0.72	80.62 ± 0.60	84.25 ± 0.78	70.44 ± 0.41	89.18 ± 0.13	81.21 ± 0.14	84.15 ± 0.05
-	0.2	79.54	86.87 ± 0.38	59.56 ± 0.72	80.62 ± 0.60	84.25 ± 0.78	70.44 ± 0.41	89.18 ± 0.13	81.21 ± 0.14	84.15 ± 0.05
-	0.3	79.53	86.87 ± 0.38	59.56 ± 0.74	80.67 ± 0.67	84.18 ± 0.77	70.44 ± 0.41	89.18 ± 0.13	81.21 ± 0.14	84.15 ± 0.05
-	0.4	80.18	87.01 ± 0.34	64.61 ± 0.59	80.60 ± 0.70	83.99 ± 0.65	70.63 ± 0.34	89.18 ± 0.13	81.27 ± 0.13	84.15 ± 0.05
-	0.5	84.86	91.94 ± 0.24	76.83 ± 0.59	86.08 ± 0.94	86.78 ± 0.51	81.10 ± 0.31	89.18 ± 0.13	82.12 ± 0.13	84.81 ± 0.05
-	0.6	66.04	89.15 ± 0.27	64.27 ± 0.71	76.49 ± 0.53	73.17 ± 1.14	63.77 ± 0.35	01.21 ± 0.05	78.05 ± 0.09	82.24 ± 0.06
-	0.7	57.20	86.84 ± 0.21	39.42 ± 0.44	66.54 ± 0.64	46.88 ± 0.38	56.51 ± 0.30	01.18 ± 0.01	78.05 ± 0.09	82.17 ± 0.06
-	0.8	56.91	86.79 ± 0.24	39.28 ± 0.43	65.35 ± 0.61	46.42 ± 0.40	56.10 ± 0.37	01.15 ± 0.00	78.03 ± 0.09	82.16 ± 0.06
-	0.9	56.89	86.64 ± 0.22	39.28 ± 0.43	65.35 ± 0.61	46.42 ± 0.40	56.07 ± 0.37	01.15 ± 0.00	78.03 ± 0.09	82.16 ± 0.06
-	1.0	56.88	86.64 ± 0.22	39.28 ± 0.43	65.35 ± 0.61	46.42 ± 0.40	56.01 ± 0.37	01.15 ± 0.00	78.03 ± 0.09	82.16 ± 0.06

Table 4. **Changing ratio at inference-time and token-budgeted inference for Foundry-Gate.** We use the best backbone on each dataset with $\lambda_{\text{gate}} = 10^{-11}$ and without weight decay. The token-budget represents the maximum number of tokens we accept as input in the core encoder. r represents the fusion ratio threshold.

Empirical computational cost. We provide GPU and CPU costs in Tab. 5. The used GPU is a NVIDIA RTX A3000 6GB Laptop GPU and the CPU is Intel Core i9-11950H @ 2.60GHz. Reported values take into account tokenization and encoding steps.

C.2. Visualizations

Embedding visualization. In Fig. 4, we observe that at the encoder output clear clusters appear, in opposition to outputs of the backbone with supertokens where the clusters are more mixed, which explains the degradation in fine-tuning results. This is due to our chosen architecture where in CAU, to recover tokens we upsample from encoder outputs (processed supertokens) and CAM from DSO and we add the output of the tokenizer to add some original information as done in 3DLST [12]. Nonetheless, it proves that our method succeed to extract basis vectors of the representation space. Between embeddings of different compression levels, we can't say that one is better than the other. With ViT-T, we see more close clusters and the separation is less obvious compared to its left neighbor. We think that comes from both compressing the input and the model embedding size. The capacity of encoding data is less important but helps to speed up inferences.

Method	# supertokens	Token budget	Params (M)	MACs (G)	FLOPs (G)	GPU Mem (MiB)	Avg runtime (s)	Throughput (obj/s)
baseline	-	-	21.84	238.624	478.128	1143.0	0.09/0.93	747.11/68.71
ToMe [2] ($r_{\text{tome}} = 16$)	-	-	21.84	115.412	231.319	1143.0	0.06/0.63	1091.42/100.96
ToMe [2] ($r_{\text{tome}} = 8$)	-	-	21.84	97.8966	196.233	1143.0	0.05/0.60	1174.07/107.56
ToMe [2] ($r_{\text{tome}} = 4$)	-	-	21.84	90.0449	180.505	1143.0	0.05/0.58	1213.10/109.90
ToMe [2] ($r_{\text{tome}} = 2$)	-	-	21.84	86.572	173.548	1143.0	0.05/0.58	1219.81/110.11
ToMe [2] ($r_{\text{tome}} = 1$)	-	-	21.84	85.0621	170.523	1143.0	0.05/0.58	1217.40/109.93
Foundry (Ours)								
Foundry	16	-	22.73	88.9903	178.394	1135.5	0.06/0.61	1160.97/104.32
Foundry (ViT-T)	16	-	6.33	71.6618	143.698	1073.0	0.05/0.58	1335.26/110.52
Foundry	8	-	22.73	78.042	156.464	1135.5	0.05/0.58	1271.58/110.28
Foundry	4	-	22.73	72.5678	145.498	1135.5	0.05/0.59	1309.76/109.08
Foundry	2	-	22.73	69.8307	140.016	1135.5	0.05/0.56	1349.58/113.30
Foundry	1	-	22.73	68.4622	137.274	1135.5	0.05/0.56	1365.92/115.18
Foundry-Gate	16	-	22.78	157.593	315.812	1135.7	0.11/0.97	591.60/65.88
Foundry (Ours) - Inference token-budgeted								
Foundry-Gate	16	16	22.78	87.8332/89.1921	176.076/178.798	1135.7	0.07/0.64	864.02/100.57
Foundry-Gate	16	32	22.78	107.651/107.651	215.774/215.774	1135.7	0.08/0.72	756.92/88.91
Foundry-Gate	16	48	22.78	126.073/127.432	252.674/255.396	1135.7	0.09/0.82	679.59/78.27
Foundry (Ours) - change fusion ratio r								
Foundry-Gate (r=0.1)	16	-	22.78	87.8332/168.012	176.076/336.684	1135.7	0.07/0.66	881.45/97.50
Foundry-Gate (r=0.2)	16	-	22.78	89.1921/163.293	178.798/327.23	1135.7	0.07/0.66	875.99/97.52
Foundry-Gate (r=0.3)	16	-	22.78	159.254/159.141	319.139/318.912	1135.7	0.09/0.72	751.83/88.35
Foundry-Gate (r=0.4)	16	-	22.78	161.028/161.783	322.694/324.205	1135.7	0.11/0.96	588.67/66.36
Foundry-Gate (r=0.5)	16	-	22.78	157.593/156.196	315.812/313.014	1135.7	0.11/0.96	593.70/66.77
Foundry-Gate (r=0.6)	16	-	22.78	152.006/152.006	304.621/304.621	1135.7	0.10/0.94	623.29/67.88
Foundry-Gate (r=0.7)	16	-	22.78	152.006/152.006	304.621/304.621	1135.7	0.10/0.96	623.66/66.88
Foundry-Gate (r=0.8)	16	-	22.78	152.006/153.403	304.621/307.419	1135.7	0.10/0.96	620.73/66.51
Foundry-Gate (r=0.9)	16	-	22.78	152.006/152.006	304.621/304.621	1135.7	0.10/0.95	623.58/67.11

Table 5. **Empirical computational cost at inference-time.** We use random input processed only by the pre-trained considered method. The input contains 64 point clouds of 2^{10} 3D points. We group them into 64 tokens of 32 points each. Runtime and throughput are averaged over 40 runs after 10 warmup runs on an NVIDIA RTX A3000 6GB Laptop GPU/Intel Core i9-11950H @ 2.60GHz (tokenization and encoding are included). All methods take N tokens and output N tokens (regardless of the specific task). ToMe [2] halves the number of tokens at each layer up to the specified r_{tome}. We use unfrozen backbones for Foundry. For Foundry-Gate, we use the model with $\lambda_{\text{gate}} = 10^{-11}$ without weight decay. MACs and FLOPs are for the forward pass only and were computed with the `calflops` library. The Token-budget represents the maximum number of tokens we accept as input in the core encoder.

PCA embedding visualization. In Fig. 5, we observe that the supertokens are more inclined to analyze edges rather than semantics by analyzing PCA embeddings per object. We think that the DSO/CAU blocks try to focus the encoder more on discriminative parts for simplifying analysis. This is why for complex objects, the distilled encoder succeeds in classifying them. Additionally, with Fig. 6, we also share some examples where the semantics seem equal or even better. We also see that the edges are more present compared to the baseline. In Fig. 7, some semantics are degraded by being less precise like flower, plane or glass, or completely lost with the bottle or car when the number of supertokens is too low. This demonstrates the need to use enough supertokens not only to improve performances, but for keeping semantic information in the latent space.

CAM visualization. Figure 8 shows the chosen supertokens for three random examples. Firstly, we observe that the number of chosen supertokens is not constant and it depends on the considered example. Some need more supertokens for representing the object and some only one. Also, we remark that the fewer supertokens we defined at the beginning, the more supertokens are used. For the first column, we observe that for 16 supertokens, one is needed but for 8, we use all. We think that when we let the model with larger number of supertokens, they are more specialized than smaller, which is expected behavior. Finally, reducing the embedding size 384 of ViT-S to 192 of ViT-T degrades the quality of supertokens. This result makes the link with the observation on the global embedding in Fig. 4.

Supertokens visualization. Figure 9 shows selected tokens on few objects. In this figure, we observe that different supertokens are used for each category of objects, even though some are shared between several classes. When divergent forms appear within the same category, other supertokens tend to be used. This is consistent with the fact that merging is performed after tokenization, without access to the context provided by neighboring tokens, which limits the consideration of more global semantic information. Thus, merging relies mainly on low-level cues—such as local patterns or geometric structures (e.g., the orientation or curvature of a shape). This explains why some supertokens are specialized for very local features,

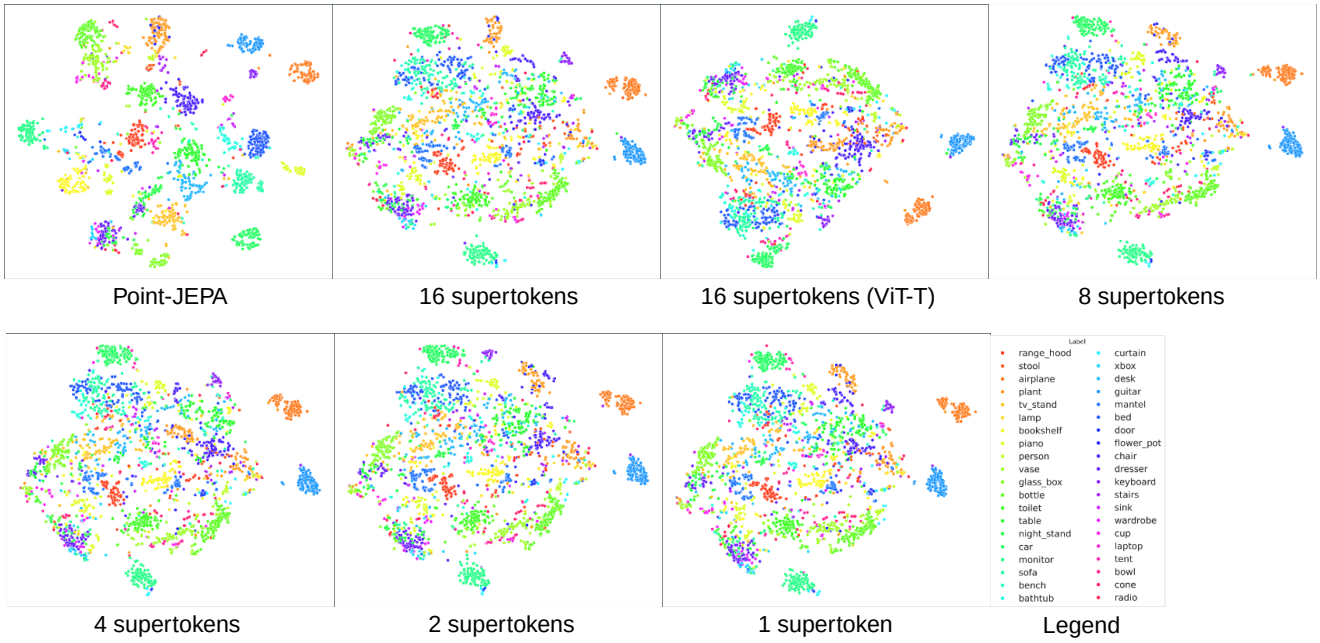


Figure 4. **Embedding visualization for Foundry.** t-SNE [18] projections of the output representation of frozen backbone at distillation ending on ModelNet40 [21] validation set. Point-JEPA outputs are from encoder and CAU for others. If not specified, the Transformer architecture used for student during distillation is ViT-S. We recommend zooming in to see details.

while others, which are more general, appear in several classes. In other words, the distribution of supertokens reflects both the internal geometric diversity within categories and the degree of invariance captured by the merging process.

Figures 10 and 11 respectively displays the embedding space of tokenized inputs and the supertokens set before and after being projected by $\mathbf{W_Q}$ and $\mathbf{W_K}$. In Fig. 10, we observe that the supertokens all appear in the same location in the token and supertoken space before projection. This result is expected, as the similarity used to construct the CAM is based on a linear projection applied identically to tokens and supertokens via the two weight matrices. Figure 11 is therefore more informative: it represents the space in which the fusion is actually performed. In this space, supertokens are generally close to the tokens assigned to them. However, some are not. We believe that this discrepancy stems from the effects of t-SNE projection: the dimensions relevant for distinguishing certain tokens may not be correctly reproduced in 2D, which can mask the actual relationships that exist in high-dimensional space.

D. Hyper-parameters

This section highlights the used hyper-parameters for distillation and fine-tunings. Table 6 shows used values for important hyper-parameters during distillation. For fine-tunings, we use traditional hyper-parameters and for those specific for Foundry and Foundry-Gate, we vary the value of the epoch when we unfreezing the encoder for classification (0, 100 and 150) and part segmentation (0 and 300) tasks and we fine-tuned only the distilled checkpoints which no weight decay is added for gate module. For OmniObject3D, we use ModelNet40 configuration.

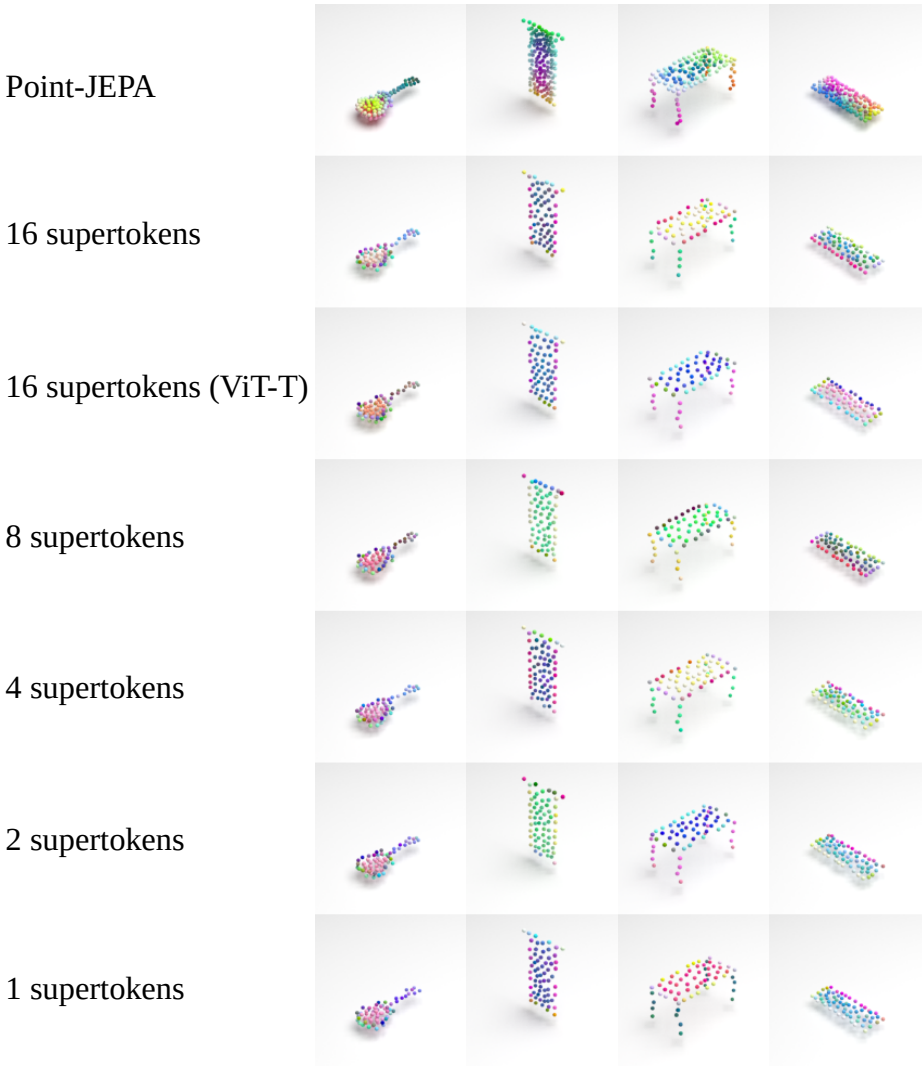


Figure 5. **PCA embedding visualization for Foundry which changes feature quality by focusing on object edges for few examples.** PCA projections of the output representation of frozen backbone at distillation ending on ModelNet40 [21] validation set in RGB space. If not specified, the Transformer architecture used for student during distillation is ViT-S. We recommend zooming in to see details. The associated labels for each object are ordered as follow: guitar, curtain, table and keyboard.

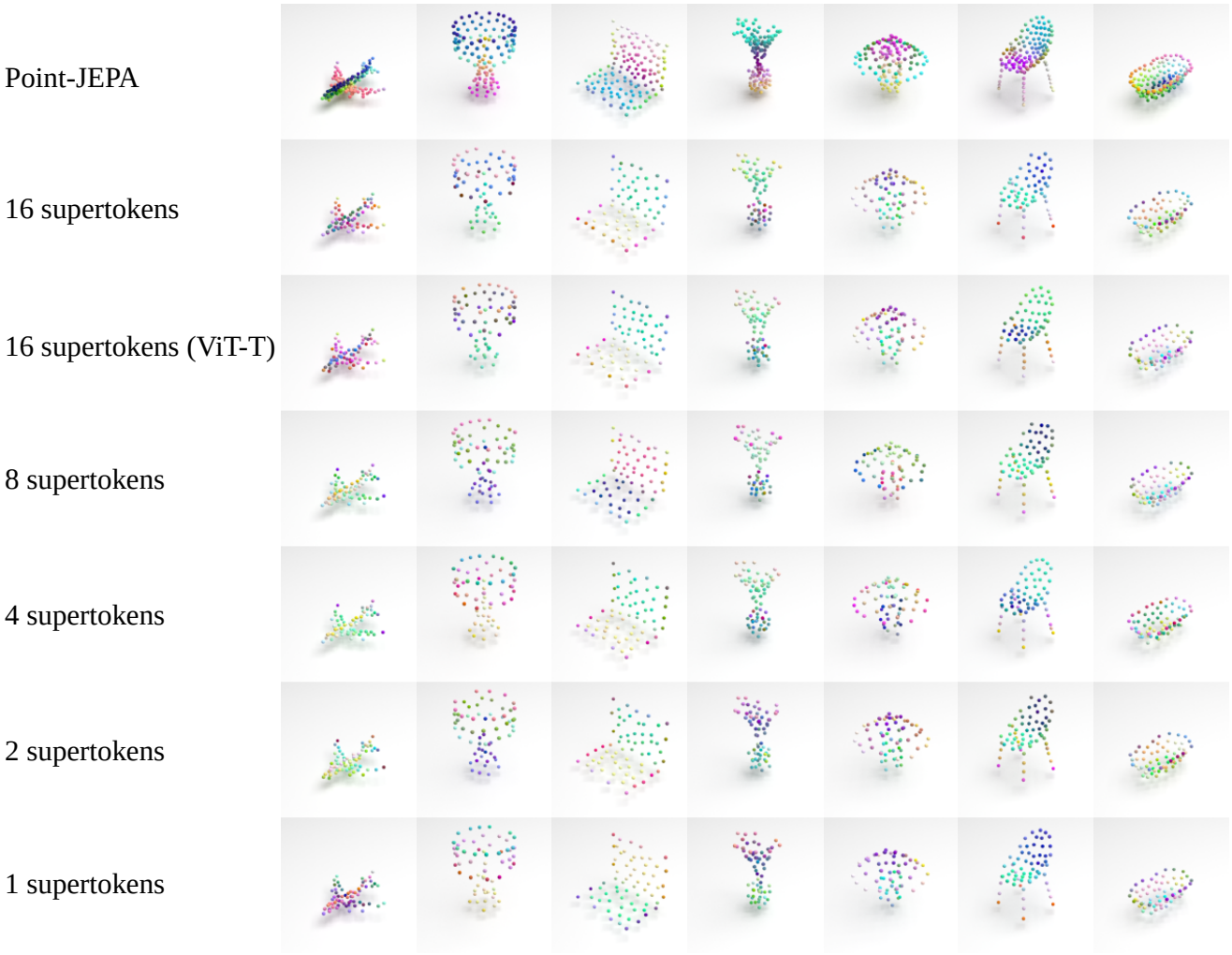


Figure 6. **PCA embedding visualization for Foundry for keeping feature quality on some selected examples.** PCA projections of the output representation of frozen backbone at distillation ending on ModelNet40 [21] validation set in RGB space. If not specified, the Transformer architecture used for student during distillation is ViT-S. We recommend zooming in to see details. The associated labels for each object are ordered as follow: airplane, lamp, laptop, plant, cone, chair and bathtub.

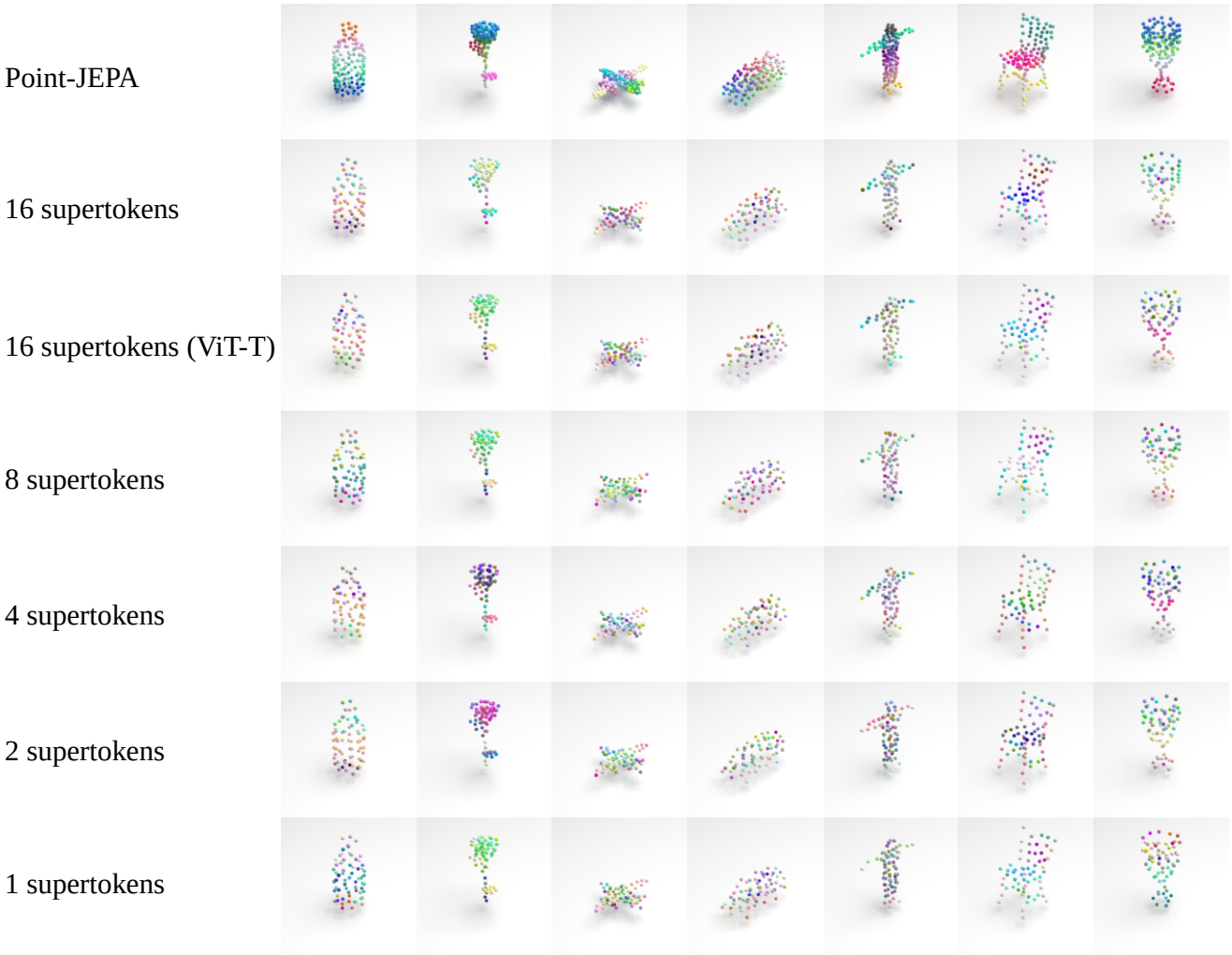


Figure 7. **PCA embedding visualization for Foundry for lower feature quality examples.** PCA projections of the output representation of frozen backbone at distillation ending on ModelNet40 [21] validation set in RGB space. If not specified, the Transformer architecture used for student during distillation is ViT-S. We recommend zooming in to see details. The associated labels for each object are ordered as follow: bottle, plant, airplane, car, person, chair and cup.

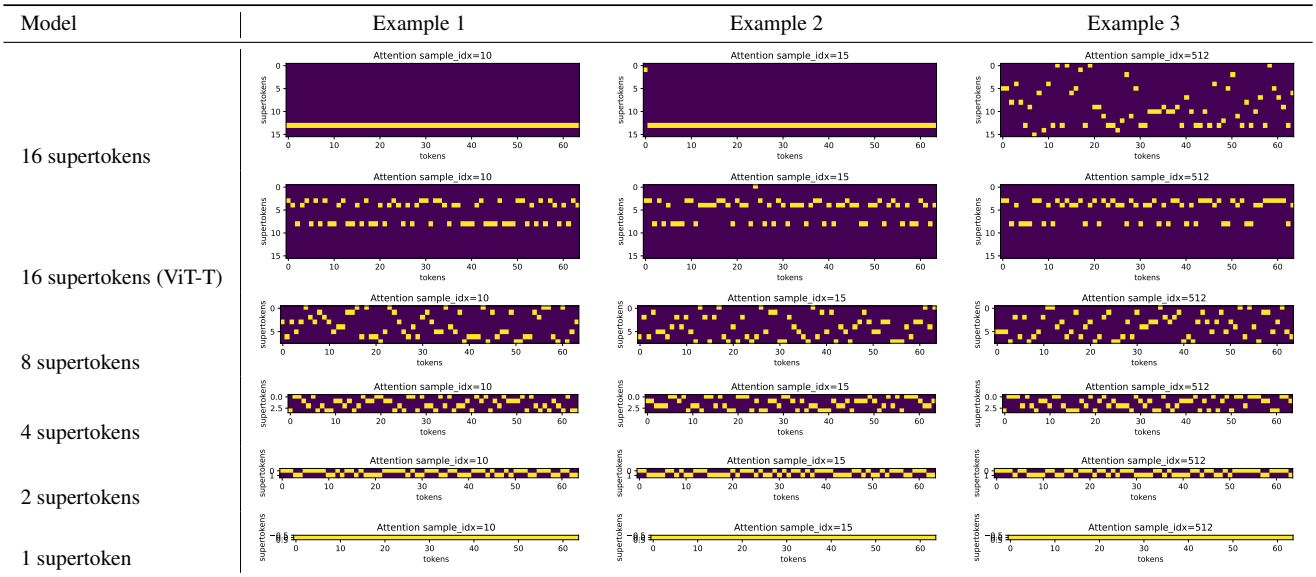


Figure 8. **Cross-Attention Map (CAM) comparison.** Unless specified, ViT-S backbone is used for student branch during distillation. The first example represents a person, second a bottle and third a toilet object. The examples come from validation set of ModelNet40 [21]. The used backbones come from the end of distillation stage.

Figure 9. **Qualitative supertokens selection view in 3D space.** We select examples from following categories: airplane, chair, guitar, keybord, lamp, plant, sofa and, table. Colors are unique and are associated to supertokens throughout all selected examples. The examples come from validation set of ModelNet40 [21]. Classifier is the 16 supertokens best checkpoint.

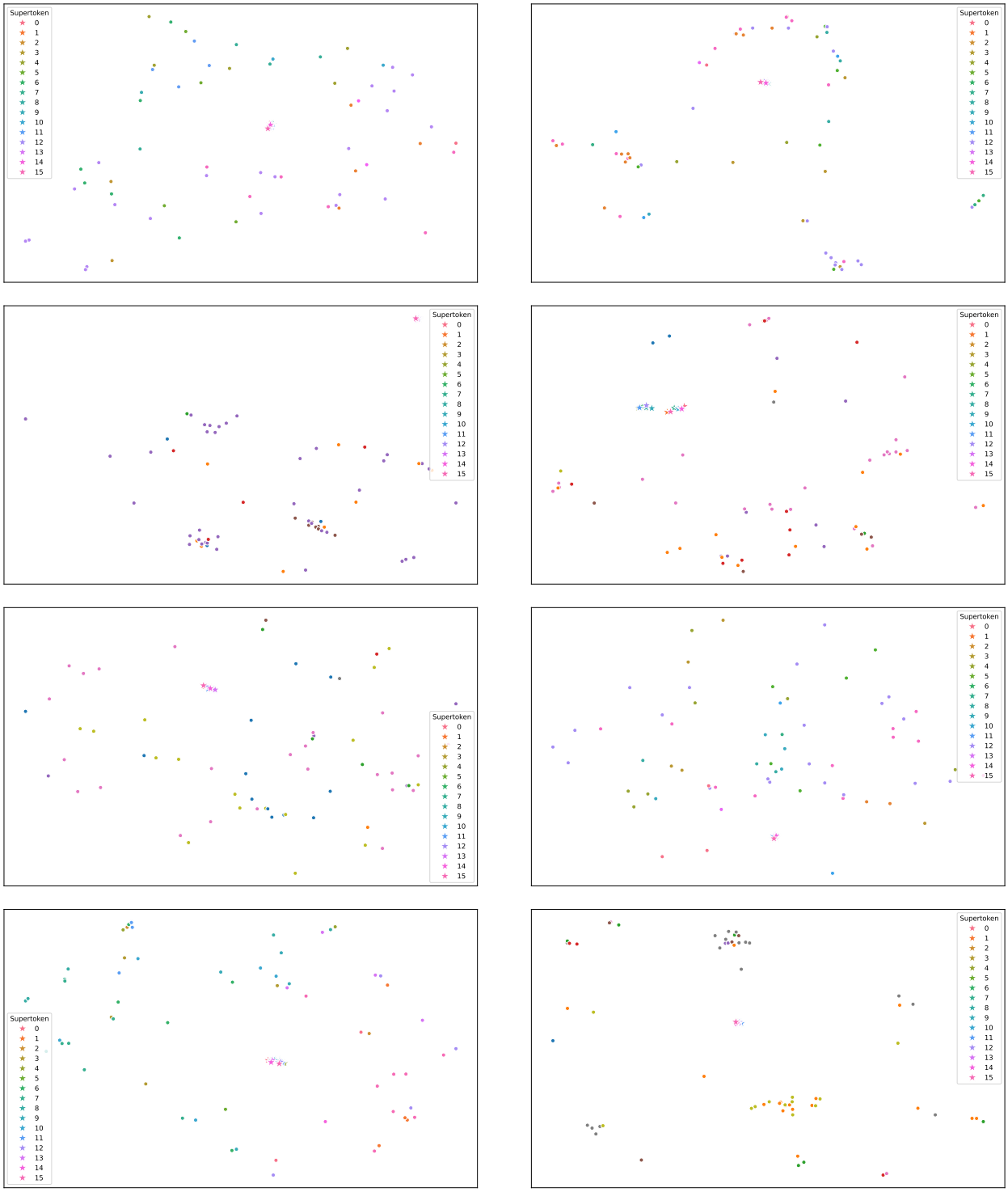


Figure 10. **Qualitative supertokens selection view in token-supertoken space before projecting.** We select examples from following categories: airplane, chair, guitar, keybord, lamp, plant, sofa and, table. Colors are unique and are associated to supertokens throughout all selected examples. The examples come from validation set of ModelNet40 [21] and are the same as Fig. 9. Classifier is the 16 supertokens best checkpoint.

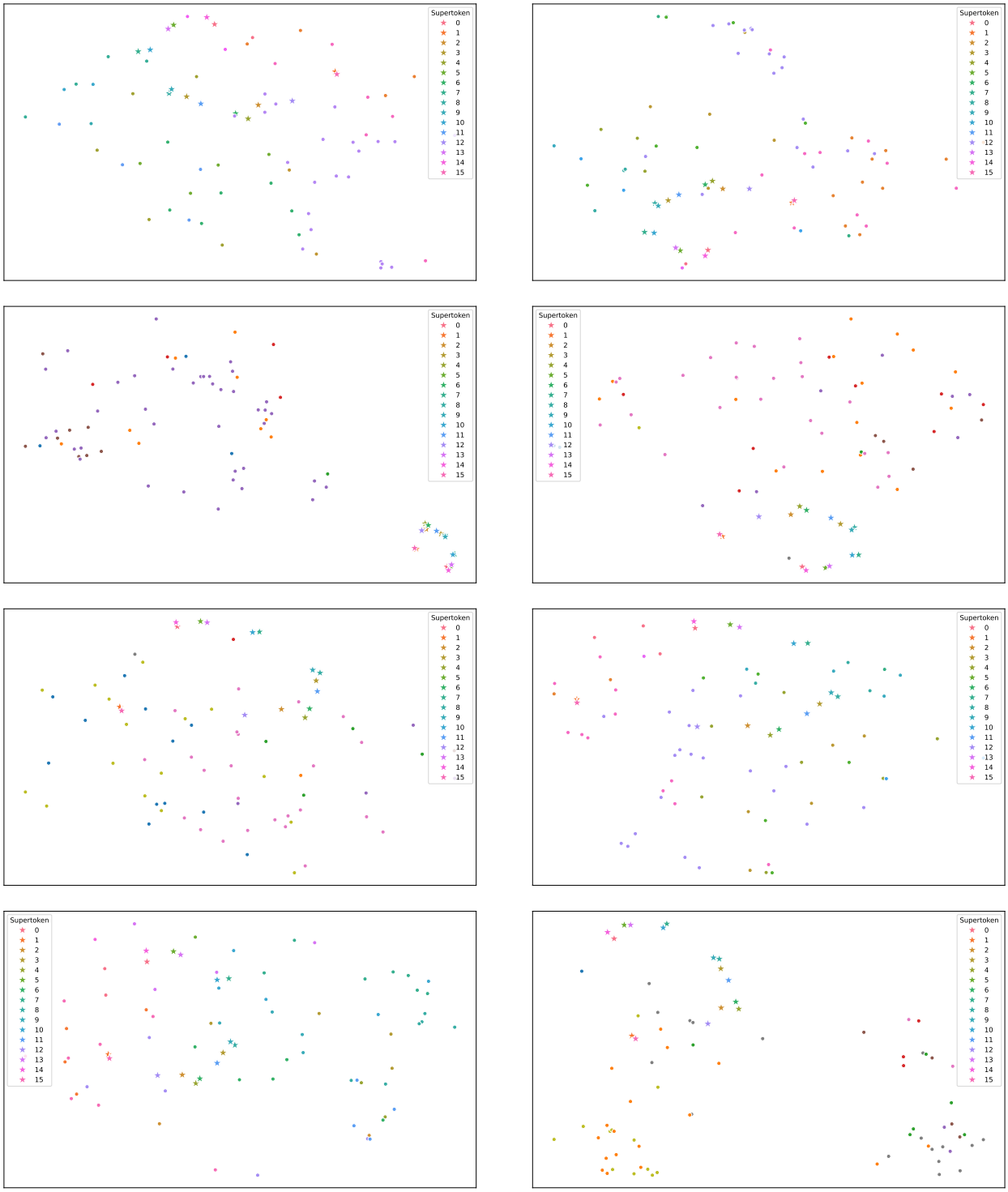


Figure 11. **Qualitative supertokens selection view in token-supertoken space after projecting.** We select examples from following categories: airplane, chair, guitar, keybord, lamp, plant, sofa and, table. Colors are unique and are associated to supertokens throughout all selected examples. The examples come from validation set of ModelNet40 [21] and are the same as Fig. 9. Classifier is the 16 supertokens best checkpoint.

Hyper-parameter	Foundry	Foundry-Gate
data		
datasets	ShapeNet55 [3]	ShapeNet55 [3]
# points	1024	1024
augmentation		
scale	(0.8, 1.2)	(0.8, 1.2)
anisotropic scaling	enabled	enabled
flip	y axis	y axis
rotate Y axis	$[-\pi, \pi]$	$[-\pi, \pi]$
optimization		
batch size	512	512
max epochs	150	150
optimizer	AdamW [11]	AdamW [11]
weight decay	5×10^{-2}	5×10^{-2}
weight decay on gate	-	{false, true}
momentum	0.9	0.9
lr	10^{-3}	10^{-3}
start lr	10^{-6}	10^{-6}
final lr	10^{-6}	10^{-6}
lr scheduler	Cosine annealing	Cosine annealing
warmup type	linear	linear
warmup epochs	15	15
encoder unfreeze	{50, max epochs}	{50, max epochs}
architecture		
teacher encoder	ViT-S	ViT-S
student encoder	ViT-T/S	ViT-S
# groups/tokens	64	64
group size	32	32
max # supertokens	$\{2^i\}_{i=1}^{4}$	16
use supertoken $\mathbf{P}$	false	false
bias on DSO qkv	false	false
$\arg\max$ in DSO	true	true
λ_{gate} values	-	$\{0\} \cup \{10^i\}_{i=-15}^{1} \cup \{10^i \mid i \in A\}$, $A = \{-10.25, -10.5, -10.75, -11.25, -11.5, -11.75\}$
hardware		
dtype	float32	float32
accelerator	1 A100 80G	1 A100 80G

Table 6. **Distillation hyper-parameters for Foundry and Foundry-Gate**.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. 2

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*. OpenReview.net, 2023. 3, 5, 6, 7, 13

[3] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing 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. *CoRR*, abs/1512.03012, 2015. 1, 21

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, pages 13142–13153, 2023. 2, 11

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019. 2

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 2

[7] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE TIP*, 6(9):1305–1315, 1997. 3

[8] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*. OpenReview.net, 2017. 2

[9] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22 (1):79–86, 1951. 2

[10] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982. 3

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*. OpenReview.net, 2019. 21

[12] Dening Lu, Linlin Xu, Jun Zhou, Kyle (yilin) Gao, and Jonathan Li. 3dlst: 3d learnable supertoken transformer for lidar point cloud scene segmentation. *Int. J. Appl. Earth Obs. Geoinformation*, 140:104572, 2025. 2, 12

[13] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 3

[14] Cédric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to merge tokens in vision transformers. *CoRR*, abs/2202.12015, 2022. 3, 6, 7

[15] Ayumu Saito, Prachi Kudeshia, and Jiju Poovvancheri. Point-jepa: A joint embedding predictive architecture for self-supervised learning on point cloud. In *WACV*, pages 7348–7357. IEEE, 2025. 2

[16] Chau Tran, Duy M. H. Nguyen, Manh-Duy Nguyen, TrungTin Nguyen, Ngan Le, Pengtao Xie, Daniel Sonntag, James Y. Zou, Binh Nguyen, and Mathias Niepert. Accelerating transformers with spectrum-preserving token merging. In *NeurIPS*, 2024. 3, 6, 7

[17] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, pages 1588–1597, 2019. 1

[18] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(86):2579–2605, 2008. 14

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017. 2

[20] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *CVPR*, pages 803–814, 2023. 1

[21] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. 1, 14, 15, 16, 17, 18, 19, 20

[22] Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Pointllm: Empowering large language models to understand point clouds. In *ECCV*, pages 131–147. Springer, 2024. 2

[23] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM TOG*, 35(6): 1–12, 2016. 2

[24] Karim Abou Zeid, Jonas Schult, Alexander Hermans, and Bastian Leibe. Point2vec for self-supervised representation learning on point clouds. In *GCPR*, pages 131–146. Springer, 2023. 3