Supplementary Material

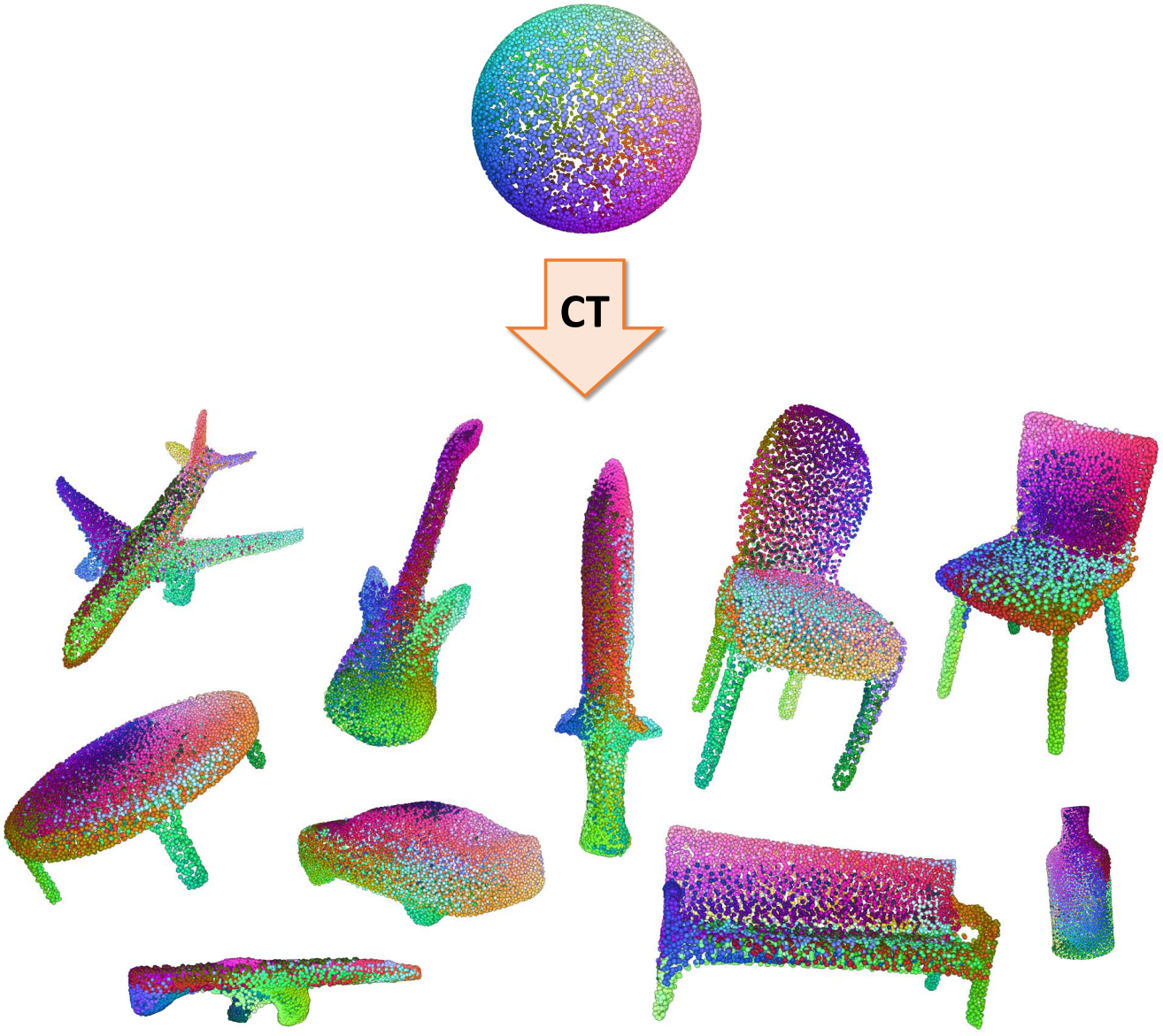

Figure 1: Results of image-based reconstruction. Our network performs the reconstruction by sampling points from the unit sphere S^2 (top) and then "folding" this set into a point cloud corresponding to the input image. Here, we show the point clouds, with points colored according to their initial positions on the sphere (red, green, and blue values are used to color-code each Cartesian coordinate). The color-coding reveals shape correspondences within and across classes.

1. Cloud Transform pseudocode

We provide a pseudo-code in Algorithm 1 as well as the PyTorch code: https://saic-violet.github.io/cloud-transformers/.

Data: Input point cloud positions $P \in \mathbb{R}^{3\times n}$, their high-dimensional input features $X \in \mathbb{R}^{f\times n}$
Result: New high-dimensional output features $Y \in \mathbb{R}^{g\times n}$
Initialize a feature-map $I \in \mathbb{R}^{w\times w\times c_{in}}$ of spatial dimension w with zeros;
for *each point* $\boldsymbol{p}_i \in P$ **do**
  Predict residuals $\boldsymbol{r}_i \leftarrow \texttt{Linear}(\boldsymbol{p}_i)$;
  Compute rasterization positions $\boldsymbol{k}_i \leftarrow T(\boldsymbol{p}_i + \boldsymbol{r}_i)$ (see eq. (1) of the main paper);
  Predict feature to rasterize $\boldsymbol{v}_i \leftarrow \texttt{Linear}(\boldsymbol{x}_i)$;
  Compute bilinear coordinates b_i and rasterization positions $[h_\bullet, w_\bullet]$ of $\boldsymbol{k}_i$ wrt I (see eq. (1) of the main paper);
  // rasterization
  $I[h_0, w_0] \leftarrow \max(I[h_0, w_0], b_i^{00}\boldsymbol{v}_i)$;
  $I[h_0, w_1] \leftarrow \max(I[h_0, w_1], b_i^{01}\boldsymbol{v}_i)$;
  $I[h_1, w_1] \leftarrow \max(I[h_1, w_0], b_i^{10}\boldsymbol{v}_i)$;
  $I[h_1, w_1] \leftarrow \max(I[h_1, w_1], b_i^{11}\boldsymbol{v}_i)$;
end
$I \leftarrow \texttt{Convolve}\,(I)$;
for *each point* $\boldsymbol{p}_i \in P$ **do**
  // de-rasterization
  $\tilde{\boldsymbol{v}}_i \leftarrow b_i^{00} I[h_0, w_0] + b_i^{01} I[h_0, w_1] + b_i^{10} I[h_1, w_1] + I[h_1, w_1] b_i^{11}$;
  Predict output features $\boldsymbol{y}_i \leftarrow \texttt{Linear}(\tilde{\boldsymbol{v}}_i)$;
end
return $Y = \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{n-1}\}$

Algorithm 1: Cloud Transform pseudo-code. For the full discussion please refer to the main manuscript, Section 3.1

2. Additional experimental results

In Figure 1, we show the "foldings" of the input sphere in our Single-View reconstruction experiments. The resulting point colors represent spatial coordinates of their input positions on the sphere. The shape correspondences learned implicitly during training are revealed. Additionally, we provide per-class results for our experiments in Table 2, Table 3, Table 4, Table 5 and Table 6.

3. Experimental details

We train our models using the standard ADAM optimizer [1] with the learning rate $1e{-}4$ for generative tasks (image-based reconstruction and point cloud completion) and with the learning rate $1e{-}3$ for recognition tasks (semantic segmentation and classification). We halve the learning rate every 100k iterations for image-based reconstruction and point cloud inpainting experiments. For classification and semantic segmentation experiments, the learning rate is decayed by 0.7 and every 25k iterations.

For classification and segmentation experiments we use batch size 8 per GPU and 2 GPUs in total. For image-based reconstruction and point cloud completion, the batch size 4 and 2 are used respectively. In both cases, we train the models on eight GPUs.

3D CNN	2D CNN
Res3D(in=32, out=64)	Res2D(in=16, out=32)
MaxPool(2)	MaxPool(2)
Res3D(in=64, out=64)	Res2D(in=32, out=64)
MaxPool(2)	MaxPool(2)
Res3D(in=64, out=64)	Res2D(in=64, out=64)
AvgPool	AvgPool

Table 1: Our mini-CNN architectures, in the 2D and 3D cases. Note that a different ConvNet applied per each head independently.

4. Multi-Headed Cloud Pooling

Here we provide more details on our multi-headed cloud pooling layer used in our classification model.

To solve the classification task, we introduce a multi-headed pooling layer (Figure 2). This is needed, because unlike other tasks, the output of the classification process is a global vector of probabilities. We devise this layer to be as similar to our standard layers as possible.

Thus, similarly to the regular MHCT layer, the new layer performs multiple rasterizations onto 2D and 3D feature maps of spatial size 8 and 16 respectively. The channel dimension of each head is 32 for three-dimensional heads and 16 for two-dimensional heads. Afterward, the resulting feature maps are processed with three standard convolutional residual blocks, each interchanged with a max-pooling. The architectures of these 2D and 3D ConvNets are shown in Table 1. The residual path of Res(in, out) consists of two convolutional layers Conv(in, out) and Conv(out, out) with a 3×3 (or $3 \times 3 \times 3$) spatial filters, each followed by a BatchNormalization and ReLU activation.

The resulting vectors are aggregated across the heads via concatenation and processed with a dense layer to form a final classification vector $\boldsymbol{k}_{class}$.

5. Gradient Balancing

Below we provide a more thorough discussion of our gradient balancing trick introduced in the main paper and

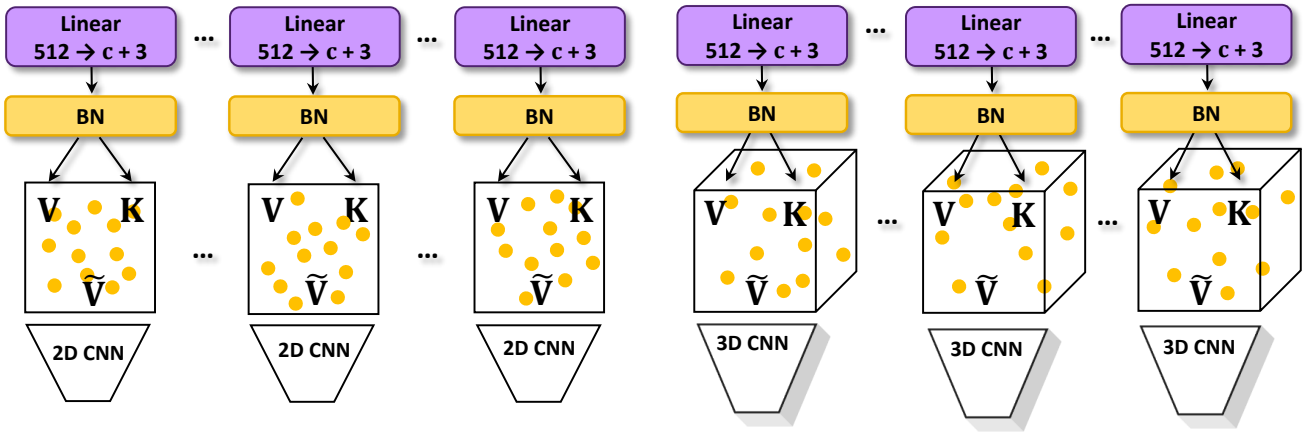

Figure 2: The Multi-Headed Cloud Pooling layer operates as a Multi-Headed Cloud Transform layer except for the de-rasterization step. Instead of de-rasterization, a compact 2D or 3D ConvNet is applied producing a single vector as an output of each head.

its necessity.

Problem discussion Ultimately, the problem can be pinpointed to the fact that as the key $\boldsymbol{k}_i$ moves from the top-left to the bottom-right of a certain grid cell thus traversing only $1/w$-th of its variation range, the assignment weight of $\boldsymbol{v}_i$ to the bottom-right corner changes from 0 to 1 (i.e traverses the full variation range). This means that gradients w.r.t. keys $\boldsymbol{k}_i$ in our architecture will always be roughly w times stronger than those w.r.t. values $\boldsymbol{v}_i$.

We justify this trick by an exact derivation. Further we denote a target loss function as $\mathcal{L}$ and assume the spatial feature dimension to be $w-1$.

Lemma 1. *Let $\boldsymbol{k} = (k_0, k_1) \in [0,1]^2$ be a key, which is typically an output of the network's key prediction branch in our Cloud Transform block. Let $\boldsymbol{b}$ be a vector of bilinear weights of $\boldsymbol{k}$ inside the enclosing cell, as in equation (1) (main paper). Then:*

$$\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{k}} = w \cdot \begin{pmatrix} (w \cdot k_1 - \lceil w \cdot k_1 \rceil) & (w \cdot k_0 - \lceil w \cdot k_0 \rceil) \\ -(w \cdot k_1 - \lfloor w \cdot k_1 \rfloor) & -(w \cdot k_0 - \lceil w \cdot k_0 \rceil) \\ -(w \cdot k_1 - \lceil w \cdot k_1 \rceil) & -(w \cdot k_0 - \lfloor w \cdot k_0 \rfloor) \\ (w \cdot k_1 - \lfloor w \cdot k_1 \rfloor) & (w \cdot k_0 - \lfloor w \cdot k_0 \rfloor) \end{pmatrix} \tag{1}$$

The derivation is straightforward, given the formula for the bilinear weights $\boldsymbol{b} = \left(b^{00}, b^{01}, b^{10}, b^{11}\right)$:

$$\begin{aligned} b^{00} &= \ (w \cdot k_i^0 - h_1)(w \cdot k_i^1 - w_1) \\ b^{01} &= -(w \cdot k_i^0 - h_1)(w \cdot k_i^1 - w_0) \\ b^{10} &= -(w \cdot k_i^0 - h_0)(w \cdot k_i^1 - w_1) \\ b^{11} &= \ (w \cdot k_i^0 - h_0)(w \cdot k_i^1 - w_0) \end{aligned} \tag{2}$$

To ease the notations, we denote the matrix above as D, i.e $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{k}} = w \cdot D$.

Lemma 2. *Let $X = \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_i}$ and $\operatorname{Var}(X)$ be its variance. Then the following inequality on matrix spectral norms holds:*

$$||D \operatorname{Var}(X) D^T||_2 \geq \frac{1}{2} ||\operatorname{Var}(X)||_2 \tag{3}$$

Proof.

$$\begin{aligned} & 2||D \operatorname{Var}(X) D^T||_2 \\ & \geq \operatorname{tr}\left(D \operatorname{Var}(X) D^T\right) && \text{trace cyclic prop.} \\ & \geq \operatorname{tr}\left(DD^T \operatorname{Var}(X)\right) \\ & \geq ||DD^T \operatorname{Var}(X)||_2 \\ & \geq \sigma_n(DD^T) ||\operatorname{Var}(X)||_2 \end{aligned} \tag{4}$$

, where $\sigma_n(DD^T)$ is the smallest non-zero singular value of $D^T D$.

Note that D has a special form, given $-1 \leq a, b \leq 0$:

$$D = \begin{pmatrix} a & b \\ -(1+a) & -b \\ -a & -(1+b) \\ 1+a & 1+b \end{pmatrix} \tag{5}$$

By a straightforward derivation, one can find two non-zero singular values $\sigma_2 = 1$ and $\sigma_1 = (2a+1)^2 + (2b+1)^2 + 1$ of DD^T. And $\sigma_1 > \sigma_1$, therefore:

$$||D \operatorname{Var}(X) D^T||_2 \geq \frac{1}{2} ||\operatorname{Var}(X)||_2 \tag{6}$$

This, informally, means that D does not decrease gradient variance $\operatorname{Var}(X)$ too much. In the case of 3D rasterization operation, the computation is analogous. □

To sum up the results above, the back-propagation from bilinear weights $\boldsymbol{b}_i$ to $\boldsymbol{k}_i$ has a form:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{k}_i} = \left(\frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{k}_i} \right)^T \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_i} = w \cdot D^T \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_i} \tag{7}$$

From the Lemma 2 we know that multiplication by D does not decrease variance's $\mathrm{Var}\,(X)$ spread more than by two times, while w takes typical vales between 16 and 128.

Intuitively, the gradients' variance is scaled up by w at each layer during the backpropagation through the keys. Therefore, given a network with d cloud transform layers, the gradient variance would "explode" as w^d. We have observed such explosions experimentally. The balancing trick discussed above successfully fixes this problem and allows training deep architectures built out of Cloud Transformer blocks.

References

[1] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[2] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proc. ICCV*, 2019.

Method	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book.	sofa	board	clut.
Pointnet	88.8	97.3	69.8	0.1	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
SegCloud*	90.1	96.1	69.9	0.0	18.4	38.4	23.1	75.9	70.4	58.4	40.9	13.0	41.6
Eff. 3D Conv	79.8	93.9	69.0	0.2	28.3	38.5	48.3	71.1	73.6	48.7	59.2	29.3	33.1
TangentConv	90.5	97.7	74.0	0.0	20.7	39.0	31.3	69.4	77.5	38.5	57.3	48.8	39.8
RNN Fusion	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
SPGraph*	89.4	96.9	78.1	0.0	42.8	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
ParamConv	92.3	96.2	75.9	0.3	6.0	69.5	63.5	66.9	65.6	47.3	68.9	59.1	46.2
PointCNN	92.3	98.2	79.4	0.0	17.6	22.7	62.1	74.4	80.6	31.7	66.7	62.1	56.7
CT (ours) std. prot.	94.4	98.2	85.4	0.0	23.6	47.4	71.0	87.3	77.5	66.0	49.3	71.1	57.8
Minkowski32*	91.7	98.7	86.1	0.0	34.0	48.9	62.4	89.8	81.5	74.8	47.2	74.4	58.5
KPConv†	92.8	97.3	82.4	0.0	23.9	58.0	69.0	91.0	81.5	75.3	75.4	66.7	58.9
JSENet†	93.8	97.0	83.0	0.0	23.2	61.3	71.6	89.9	79.8	75.6	72.3	72.7	60.4
CT† (ours)	94.2	97.7	82.7	0.0	34.4	62.8	68.4	89.8	80.4	78.2	61.4	67.7	64.9

Table 2: Semantic segmentation intersection-over-union scores on S3DIS *Area-5* split. The models without any marks use the standard protocol with chunking of the scene into blocks, while the models with † employ KPConv's [2] protocol. The rest protocols are labeled with $*$. Per-class results are provided.

Table 3: Per-class classification results on ScanObjectNN.

	bag	bin	box	cabinet	chair	desk	display	door	shelf	table	bed	pillow	sink	sofa	toilet
3DmFV	39.8	62.8	15.0	65.1	84.4	36.0	62.3	85.2	60.6	66.7	51.8	61.9	46.7	72.4	61.2
PointNet	36.1	69.8	10.5	62.6	89.0	50.0	73.0	93.8	72.6	67.8	61.8	67.6	64.2	76.7	55.3
SpiderCNN	43.4	75.9	12.8	74.2	89.0	65.3	74.5	91.4	78.0	65.9	69.1	80.0	65.8	90.5	70.6
PointNet++	49.4	84.4	31.6	77.4	91.3	74.0	79.4	85.2	72.6	72.6	75.5	81.0	80.8	90.5	85.9
DGCNN	49.4	82.4	33.1	83.9	91.8	63.3	77.0	89.0	79.3	77.4	64.5	77.1	75.0	91.4	69.4
PointCNN	57.8	82.9	33.1	83.6	92.6	65.3	78.4	84.8	84.2	67.4	80.0	80.0	72.5	91.9	71.8
GFNet	59.0	84.4	44.4	78.2	92.1	66	91.2	91.0	86.7	70.4	82.7	78.1	72.5	92.4	77.6
DI-PointCNN	65.1	80.9	62.4	80.4	90.5	78.0	86.8	88.1	84.6	67.0	84.5	82.8	73.3	95.2	74.1
CT (ours)	50.6	86.9	52.6	87.4	96.2	77.3	85.8	93.8	88.4	80.0	76.4	87.6	81.7	94.3	89.4
CT (ours) + scales	57.8	89.9	45.9	87.4	95.6	84.7	84.8	92.9	88.4	77.4	82.7	85.7	85.8	92.9	95.3

Table 4: Per-class point completion results on ShapeNet compared using F-Score@1%. Note that the F-Score@1% is computed on 16,384 points.

Methods	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft
AtlasNet	0.845	0.552	0.630	0.552	0.565	0.500	0.660	0.624
PCN	0.881	0.651	0.725	0.625	0.638	0.581	0.765	0.697
FoldingNet	0.642	0.237	0.382	0.236	0.219	0.197	0.361	0.299
TopNet	0.771	0.404	0.544	0.413	0.408	0.350	0.572	0.560
MSN	0.885	0.644	0.665	0.657	0.699	0.604	0.782	0.708
GRNet	0.843	0.618	0.682	0.673	0.761	0.605	0.751	0.750
CT (ours)	0.921	0.652	0.733	0.710	0.774	0.628	0.811	0.789

Table 5: Per-class point completion results on ShapeNet compared using Chamfer Distance (CD) with L2 norm computed on 16.384 points and multiplied by 10^4.

Methods	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft
AtlasNet	1.753	5.101	3.237	5.226	6.342	5.990	4.359	4.177
PCN	1.400	4.450	2.445	4.838	6.238	5.129	3.569	4.06
FoldingNet	3.151	7.943	4.676	9.225	9.234	8.895	6.691	7.32
TopNet	2.152	5.623	3.513	6.346	7.502	6.949	4.784	4.359
MSN	1.543	7.249	4.711	4.539	6.479	5.894	3.797	3.853
GRNet	1.531	3.620	2.752	2.945	2.649	3.613	2.552	2.122
CT (ours)	1.059	4.592	2.581	4.163	3.294	5.816	3.360	2.274

	AtlasNet	OGN	Matryoshka	Retrieval	Oracle NN	CT(ours)
airplane	0.39	0.26	0.33	0.37	0.45	0.53
ashcan	0.18	0.23	0.26	0.21	0.24	0.33
bag	0.16	0.14	0.18	0.13	0.15	0.20
basket	0.19	0.16	0.21	0.15	0.15	0.22
bathtub	0.25	0.13	0.26	0.22	0.26	0.33
bed	0.19	0.12	0.18	0.15	0.17	0.22
bench	0.34	0.09	0.32	0.3	0.34	0.46
birdhouse	0.17	0.13	0.18	0.15	0.15	0.31
bookshelf	0.24	0.18	0.25	0.2	0.2	0.29
bottle	0.34	0.54	0.45	0.46	0.55	0.59
bowl	0.22	0.18	0.24	0.2	0.25	0.25
bus	0.35	0.38	0.41	0.36	0.44	0.53
cabinet	0.25	0.29	0.33	0.23	0.27	0.44
camera	0.13	0.08	0.12	0.11	0.12	0.18
can	0.23	0.46	0.44	0.36	0.44	0.48
cap	0.18	0.02	0.15	0.19	0.25	0.16
car	0.3	0.37	0.38	0.33	0.39	0.45
cellular	0.34	0.45	0.47	0.41	0.5	0.58
chair	0.25	0.15	0.27	0.2	0.23	0.35
clock	0.24	0.21	0.25	0.22	0.27	0.36
dishwasher	0.2	0.29	0.31	0.22	0.26	0.27
display	0.22	0.15	0.23	0.19	0.24	0.31
earphone	0.14	0.07	0.11	0.11	0.13	0.27
faucet	0.19	0.06	0.13	0.14	0.2	0.30
file	0.22	0.33	0.36	0.24	0.25	0.43
guitar	0.45	0.35	0.36	0.41	0.58	0.60
helmet	0.1	0.06	0.09	0.08	0.12	0.13
jar	0.21	0.22	0.25	0.19	0.22	0.28
keyboard	0.36	0.25	0.37	0.35	0.49	0.32
knife	0.46	0.26	0.21	0.37	0.54	0.61
lamp	0.26	0.13	0.2	0.21	0.27	0.37
laptop	0.29	0.21	0.33	0.26	0.33	0.44
loudspeaker	0.2	0.26	0.27	0.19	0.23	0.33
mailbox	0.21	0.2	0.23	0.2	0.19	0.32
microphone	0.23	0.22	0.19	0.18	0.21	0.21
microwave	0.23	0.36	0.35	0.22	0.25	0.48
motorcycle	0.27	0.12	0.22	0.24	0.28	0.34
mug	0.13	0.11	0.15	0.11	0.17	0.15
piano	0.17	0.11	0.16	0.14	0.17	0.21
pillow	0.19	0.14	0.17	0.18	0.3	0.39
pistol	0.29	0.22	0.23	0.25	0.3	0.35
pot	0.19	0.15	0.19	0.14	0.16	0.25
printer	0.13	0.11	0.13	0.11	0.14	0.19
remote	0.3	0.33	0.31	0.31	0.37	0.44
rifle	0.43	0.28	0.3	0.36	0.48	0.55
rocket	0.34	0.2	0.23	0.26	0.32	0.2
skateboard	0.39	0.11	0.39	0.35	0.47	0.58
sofa	0.24	0.23	0.27	0.21	0.27	0.34
stove	0.2	0.19	0.24	0.18	0.19	0.33
table	0.31	0.24	0.34	0.26	0.34	0.42
telephone	0.33	0.42	0.45	0.4	0.5	0.5
tower	0.24	0.2	0.25	0.25	0.25	0.33
train	0.34	0.29	0.3	0.32	0.38	0.51
vessel	0.28	0.19	0.22	0.23	0.29	0.35
washer	0.2	0.31	0.31	0.21	0.25	0.32

Table 6: F-score evaluation (@1%) in the viewer-centered mode, per-class results.