Canonical Capsules: Self-Supervised Capsules in Canonical Pose

Supplementary Material

This appendix provides additional architectural details (Section A), and additional ablation studies (Section B). Note that we also provide code and additional qualitative results as a webpage with videos alongside this appendix; see `README.html`.

A Architectural details

We detail our architecture design for the capsule encoder $\mathcal{E}$, the decoder $\mathcal{D}$ and the regressor $\mathcal{K}$.

A.1 Capsule Encoder – $\mathcal{E}$

The capsule encoder $\mathcal{E}$ takes in input a point cloud $\mathbf{P} \in \mathbb{R}^{P\times D}$ and outputs the K-fold/head attention map $\mathbf{A} \in \mathbb{R}^{P\times K}$ and the feature map $\mathbf{F} \in \mathbb{R}^{P\times C}$. Specifically, the encoder $\mathcal{E}$ is based on ACNe [47] and composed of 3 residual blocks where each residual block consists of two hidden MLP layers with 128 neurons each. Each hidden MLP layer is followed by *Attentive Context Normalization* (ACN) [47], batch normalization [26], and ReLU activation. To be able to attend to multiple capsules, we extend the ACN layer into multi-headed ACN.

Multi-headed ACN. For each MLP layer where we apply ACN, if we denote this layer as i, we first train a fully-connected layer that creates a K-headed attention map $\mathbf{A}^i \in \mathbb{R}^{P\times K}$ given the $\mathbf{F}^i \in \mathbb{R}^{P\times C}$ of this layer. This is similar to ACN, but instead of a single attention map, we now have K. The normalization process is similar to ACN: we utilize the weighted moments of $\mathbf{F}^i$ with $\mathbf{A}^i$, but results in K normalized outcomes instead of one. We them aggregate these K normalization results into one by summation. In more details, given the $\mathbf{F}^i_p \in \mathbb{R}^{P\times C}$, if we denote the weights and biases to be trained for the k^{th} attention head to be $\mathbf{W}^i_k \in \mathbb{R}^{C\times 1}$ and $b^i_k \in \mathbb{R}^1$ we write:

$$\mathbf{A}^i_{p,k} = \frac{\exp(\mathbf{F}^i_p \mathbf{W}^i_k + b^i_k)}{\sum_k \exp(\mathbf{F}^i_p \mathbf{W}^i_k + b^i_k)} \,. \tag{13}$$

We then compute the moments that are used to normalize in ACN, but now for each attention head:

$$\boldsymbol{\mu}_k = \sum_p \frac{\mathbf{A}^i_{p,k}\mathbf{F}^i_p}{\sum_p \mathbf{A}^i_{p,k}} \,,\; \boldsymbol{\sigma}_k = \sum_p \frac{\mathbf{A}^i_{p,k}(\mathbf{F}^i_p - \mu_k)^2}{\sum_p \mathbf{A}^i_{p,k}} \,, \tag{14}$$

which we then use to normalize and aggregate (sum) to get our final normalized feature map:

$$\mathbf{F}^i_p = \sum_k \mathbf{A}^i_{p,k} \frac{(\mathbf{F}^i_p - \boldsymbol{\mu}_k)}{\sqrt{\boldsymbol{\sigma}_k + \epsilon}} \,, \tag{15}$$

where $\epsilon = 0.001$ is to avoid numerical instabilities.

A.2 Capsule Decoder – $\mathcal{D}$

The decoder $\mathcal{D}$ is composed of K per-capsule decoders $\mathcal{D}_k$. Each per-capsule decoder $\mathcal{D}_k$ maps the k^{th} capsule in canonical frame $(\bar{\mathbf{R}}\bar{\boldsymbol{\theta}}_k + \bar{\mathbf{t}}, \boldsymbol{\beta}_k)$ to a group of points $\tilde{\mathbf{P}}_k \in \mathbb{R}^{M\times D}$ that corresponds to a part of the entire object. We then obtain the auto-encoded (reconstructed) point clouds $\tilde{\mathbf{P}}$ by collecting the outputs of K per-capsule decoders and taking their union as in (4). Specifically, each $\mathcal{D}_k$ consists of 3 hidden MLP layers of (1280, 640, 320) neurons, each followed by batch normalization and a ReLU activation. At the output layer, we use a fully connected layer, followed by `tanh` activation to regress the coordinates for each point. Similarly to AtlasNetV2 [12], $\mathcal{D}_k$ additionally receives a set of trainable grids of size $(M \times 10)$[4] and deforms them into $\tilde{\mathbf{P}}_k$, based on the shape code $\boldsymbol{\beta}_k$. Note that the generated point clouds from $\mathcal{D}_k$ lie in the reference frame of the canonicalized pose $\bar{\boldsymbol{\theta}}_k$, so we rotate and translate the point cloud by $\bar{\boldsymbol{\theta}}_k$ in the output layer.

[4] Note the constant 10 is not related to K here.

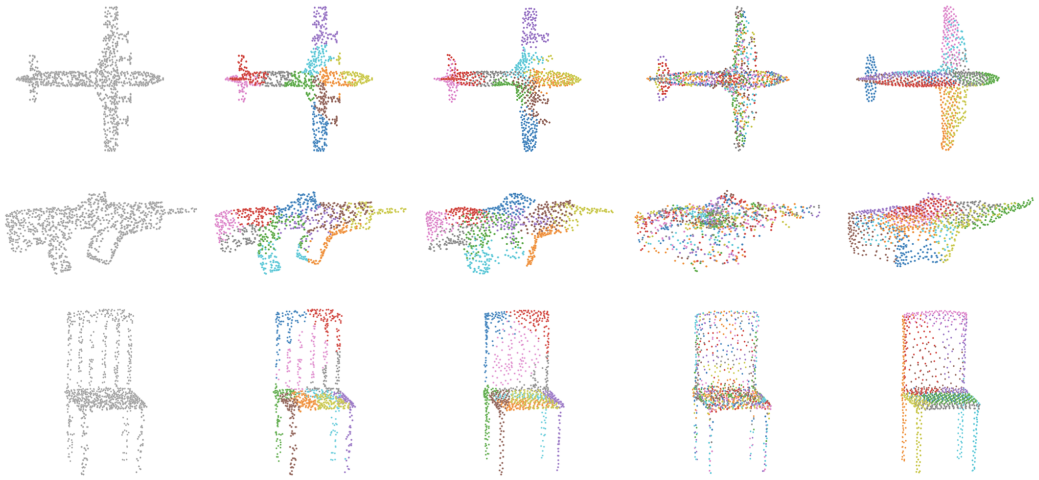


Figure 3: **Auto-encoding / qualitative –** Example decomposition results using Canonical Capsules on the test set, with the *aligned* setup; we color each decomposition (capsule) with a unique color. For 3D-PointCapsNet [64] and AtlasNetV2 [12], rather than capsules, these correspond to "patches" in the reconstruction network. Our method clearly provides the best qualitative results.

A.3 Regressor– $\mathcal{K}$

The regressor $\mathcal{K}$ learns to regress the canonical pose for each capsule $\bar{\boldsymbol{\theta}} \in \mathbb{R}^{K \times D}$ from their descriptors $\boldsymbol{\beta}_k$. In order to do so, we concatenate the (ordered) set of descriptors $\{\boldsymbol{\beta}_k\}$ into a single, global descriptor, which we then feed into a fully connected layer with $128 \times K$ neurons and a ReLU activation, followed by an additional fully connected layer with $D \times K$ neurons, creating K D-dimensional outputs (i.e. canonical pose). We do not apply any activation on the second layer. To make $\bar{\boldsymbol{\theta}}$ zero-centered in the canonical frame, we further subtract the mean of the outputs.

B Additional ablation studies

Auto-encoding with aligned data – Figure 3 and Table 1. For completeness, we further show qualitative results for auto-encoding on an *aligned* dataset, the most common setup of prior works in the literature. As shown in Figure 3, our method provides best reconstruction performance even in this case; for quantitative results, please see Table 1. Interestingly, while our *decoder* architecture is similar to AtlasNetV2 [12], our reconstructions are of much higher quality; our methods provides finer details at the propellers on the airplane, at the handle of the firearm, and at the back of the chair. This further supports the effectiveness of our capsule encoding.

Number of points P – Table 6. To speed-up experiments we have mostly used P=1024, but in the table we show that our findings are consistent regardless of the number of points used. Note that the results of AtlasNetV2 [12] are *very* similar to what is reported in the original paper. The slight differences exist due to random subsets that were used in AtlasNetV2.[5]

Table 6: **Number of points P –** Auto-encoding performance (Chamfer distance $\times 10^3$) as we vary the input point cloud cardinality; *aligned* setup for both training and testing, with *all* object categories.

	1024 pts	2500 pts
3D-PointCapsNet [64]	2.49	1.49
AtlasNetV2 [12]	2.14	1.22
Our method	**1.76**	**0.97**

Effect of number of capsules – Table 7. To verify how the number of capsules affect performance, we test with varying the number of capsules. We keep the representation power constant by *reducing* the dimension of descriptors as more capsules are used; for example, with 10

[5] And to a minor bug in the evaluation code (*i.e.* non deterministic test set creation) that we have already communicated to the authors of [19].

Table 7: **Ablation study on the number of capsules –** We show the reconstruction performance (Chamfer distance $\times 10^3$) with varying number of capsules. While they all perform better than competitors, 10 capsules give best performance. Note the representation power is kept *constant* as we vary the number of capsules.

AtlasNetV2 [12]	5 capsules	10 capsules	20 capsules
2.80	1.25	**1.11**	1.15

Table 8: **Number of capsules w/ fixed descriptor dimension per capsule –** Better reconstruction performance is obtained as number of capsules are increased, since the network capacity expands proportionally.

Number of capsules	5	10	20
CD	1.51	1.11	1.02

Table 9: **One-shot canonicalization –** Reconstruction performance of our method against a naive one-shot alignment, where an arbitrary point cloud is selected as reference.

	Airplane	Chair	All
One shot alignment	**1.10**	2.92	2.37
Our method	1.11	**2.58**	**2.22**

Table 10: **Canonical descriptors –** Auto-encoding performance (Chamfer Distance) of our method with descriptors $\boldsymbol{\beta}_k$ vs. $\bar{\boldsymbol{\beta}}_k$. Note the unaligned setup is the one of primary interest.

	AtlasNetV2 [12]	Ours ($\boldsymbol{\beta}_k$)	Ours ($\bar{\boldsymbol{\beta}}_k$)
Aligned	1.28	**0.96**	0.99
Unaligned	2.80	2.12	**1.08**

capsules we use a 128-dimensional descriptor, with 20 we use 64, and with 5, we use 256. Our experimental results show that representation with 10 capsules achieves the best performance. Note that our method, even with the sub-optimal number of capsules, still outperforms compared methods by a large margin.

Increasing the number of capsules w/ fixed descriptor dimension – Table 8. We also report the performance for the increasing number of capsules while keeping descriptor dimension fixed. Note this is different from Table 7, as in that setting the overall network capacity was kept fixed, but here it grows linearly to the number of capsules. As shown in the Table 8, unsurprisingly, more capsules lead to better reconstruction performance as the network capacity is enhanced.

One-shot canonicalization – Table 9. A naive alternative to our learnt canonicalizer would be to use one point cloud as a reference to align to. Using the canonicalizer provides improved reconstruction performance over this naïve approach, removes the dependency on the choice of the reference point cloud, and allows our method to work effectively when dealing with multi-class canonicalization.

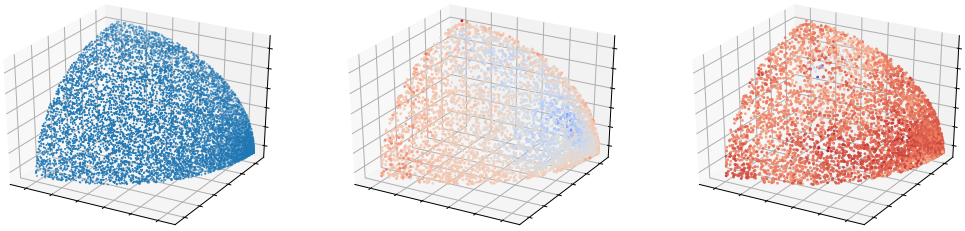
Canonical descriptors – Table 10. We evaluate the effectiveness of the descriptor enhancement strategy described in Section 3.2. We report the reconstruction performance with and without the enhancement. Recomputing the descriptor in canonical frame helps when dealing with the *unaligned* setup. Note that even without this enhancement, our method still outperforms the state-of-the-art.

The impact of $\mathcal{L}_{\text{canonical}}$. Using $\mathcal{L}_{\text{canonical}}$ is required to achieve the best performance; see the inset table. However, the reconstruction loss $\mathcal{L}_{\text{recon}}$ alone is able to guide canonicalization up to some degree. It is worth noting that excluding $\mathcal{L}_{\text{recon}}$, thus training the canonicalization component in a standalone fashion, unsurprisingly provides best mStd performance. Thus, this could be an alternative strategy to train our framework, should one *not* require end-to-end differentiability. Nonetheless, our observations still hold and our method delivers the best performance (including when compared to Compass [45]) with all losses enabled.

Table 11: **The impact of $\mathcal{L}_{\text{canonical}}$ on canonicalization –** Best canonicalization is achieved when $\mathcal{L}_{\text{canonical}}$ is used; see text for details.

	w/o $\mathcal{L}_{\text{recon}}$		w/ $\mathcal{L}_{\text{recon}}$	
	w/o $\mathcal{L}_{\text{canonical}}$	w/ $\mathcal{L}_{\text{canonical}}$	w/o $\mathcal{L}_{\text{canonical}}$	w/ $\mathcal{L}_{\text{canonical}}$
CD	-	-	1.12	**1.11**
mStd	16.0	**7.747**	8.421	**8.278**

Supervising the attention's invariance. Since $\boldsymbol{\theta}$ is inferred by weighted averaging of $\mathbf{P}$ with the attention map $\mathbf{A}$ in (2), we also considered directly adding a loss on $\mathbf{A}$ that enforces invariance instead of a loss on $\boldsymbol{\theta}$. This variant degrades only slightly in terms reconstruction (CD=1.13, where ours provides CD=$\mathbf{1.11}$) but performs very poorly when canonicalizing (mStd=94.906 vs mStd=$\mathbf{8.278}$



(a) (b) (c)

Figure 4: **Random sampling of rotations –** (a) Sampling Euler angles uniformly results in a non-uniform coverage of SO(3); note we visualize only one-eighth of a sphere for ease of visualization on paper. (b) Non uniform sampling results in auto-encoding error to be biased w.r.t rotations (we use a cold-warm colormap to visualize the Chamfer distance error). (c) By properly sampling rotations [41], this bias can be alleviated.

of ours). We hypothesize that this is because $\mathcal{L}_{\text{equivariance}}$ *directly* supervises the end-goal (capsule pose equivariance) whereas supervising $\mathbf{A}$ is an indirect one.

Random sampling of rotations – Figure 4. Lastly, we revisit how rotations are randomly sampled to generate the augmentations used by Siamese training. Uniform sampling of Euler angles (i.e., yaw,pitch,roll) leads to a non-uniform coverage of the SO(3) manifold as shown in Figure 4 (a). Due to this non-uniformity, the reconstruction quality is biased with respect to test-time rotations; see Figure 4 (b). Instead, by properly sampling [41] the reconstruction performance is much more *uniform* across the manifold; see see Figure 4 (c) In our experiments, this leads to a significant difference in auto-encoding performance; CD=**1.11** with proper uniform sampling vs CD=1.19 with the Euclidean random sampling.

Large range of random translations – Table 12. We increase the range of random translations ($[-0.2, 0.2]$ in the original paper). As shown in the Table 12, we observe the negligible changes in reconstruction performance (CD) with larger random translations.

Table 12: **Large range of random translations –** The performance changes are negligible as the magnitude of random translations increases.

Range of translation	$[-0.2, 0.2]$	$[-0.4, 0.4]$	$[-0.8, 0.8]$	$[-1.6, 1.6]$
CD	1.11	1.12	1.1	1.1

C Per-class results for auto-encoding

In addition to the auto-encoding results in Table 1, we provide per-class performance for the models trained with multiple categories. As shown in Table 13, we achieve the best performance for all classes.

Table 13: **Auto-encoding / per-class quantitative –** Performance in terms of Chamfer distance with 1024 points per point cloud – metric is multiplied by 10^3 as in [12].

	Method	Bench	Cabinet	Car	Cellphone	Chair	Couch	Firearm	Lamp	Monitor	Airplane	Speaker	Table	Watercraft	All
Aligned	3D-PointCapsNet [64]	2.06	3.23	2.64	2.25	2.64	2.99	0.93	3.40	2.85	1.36	4.26	2.56	2.05	2.49
	AtlasNetV2 [12]	1.67	2.81	2.42	2.00	2.26	2.63	0.78	2.68	2.52	1.18	3.80	2.16	1.73	2.14
	Our method	**1.44**	**2.37**	**2.10**	**1.77**	**1.90**	**2.26**	**0.59**	**1.61**	**2.09**	**0.99**	**3.04**	**1.80**	**1.31**	**1.76**
Unaligned	3D-PointCapsNet [64]	4.34	5.20	4.57	3.87	5.55	5.33	2.00	4.97	4.30	3.24	6.05	5.17	3.38	4.66
	AtlasNetV2 [12]	2.93	3.65	3.46	2.46	3.53	3.72	1.28	2.91	3.06	2.17	4.42	3.20	2.27	3.08
	AtlasNetV2 [12] w/ STN	2.44	3.30	3.10	2.17	2.93	3.28	0.96	2.56	2.70	1.60	3.96	2.67	1.86	2.60
	Our method	**1.84**	**2.72**	**2.45**	**1.86**	**2.72**	**2.70**	**0.67**	**2.20**	**2.46**	**1.30**	**3.55**	**2.36**	**1.53**	**2.22**