Geon3D: Benchmarking 3D Shape Bias towards Building Robust Machine Vision - Supplementary Materials

Yutaro Yamada, Yuval Kluger, Sahand Negahban, Ilker Yildirim

June 8, 2021

Contents

1 Datasheet

A line of work in psychophysics of human visual cognition have argued that the visual system exploits certain types of shape features in inferring 3D structure and geometry. In Geon3D, by treating these shape features as the dimensions of variation, we model 40 classes of 3D objects, and render them from random viewpoints, resulting in an image set and their corresponding camera matrices.

Data Preparation We construct each Geon using Blender —an open-source 3D computer graphics software Blender [2021].

An advantage of Geons over other geometric primitives such as superquadrics Barr [1981] is that the shape categorization of Geons is qualitative rather than quantitative. Thus, each Geon category affords a high degree of in-class shape deformation, as long as the four defining features of each shape class remains the same. Such flexibility allows us to construct a number of different 3D model instances for each Geon class by expanding or shrinking the object along the x, y, or z-axis. For each axis, we evenly sample the 11 scaling parameters from the interval [0.5, ..., 1.5] with a step size 0.1, resulting in 1331 3D model instances for each Geon category.

Rendering and data splits We randomly sample 50 camera positions from a sphere with the object at the origin. For each model instance, 50 images are rendered using these camera positions with resolution of 224x224. We then split the data into train/validation/test with ratio 8:1:1 using model instance ids, where each instance id corresponds to the scaling parameters described above. We also make sure that all Geon categories are uniformly sampled in each of train/validation/test sets.

Dataset distribution The full Geon3D-40 (black background) is available for download at `https://drive.google.com/uc?id=1v5XwO-QrnB_j9XhJJl4c7d7hMQf-v6gq&export=download`. Geon3D is distributed under the CC BY-SA 4.0 license.[1] The dataset will be maintained by Yutaro Yamada for long-term preservation. The authors bear all responsibility in case of violation of rights and confirmation of the data license. Upon publication, the dataset website will become available, where we will add structured metadata to a dataset's meta-data page, a persistent dereferenceable identifier, and any future updates.

List of 40 Geons In Figure 1, we provide a list of 40 Geons we have constructed. The label for each Geon class represents the four defining shape features, in the order of "axis", "cross section", "sweep function", "termination", as described in the main paper. We put "na" for the termination when the sweep function is constant. We also distinguish the two termination types "c-inc" and "c-dec" when the sweep function is monotonic. For instance, "c-inc" means that the curved surface is at the end of the increasing sweep function, whereas "c-dec" means that the curved surface is at the end of the decreasing sweep function. As a reference, here is the mapping between the name and the code of 10 Geons we used in 10-Geon classification: "Arch": `c_s_c_na`, "Barrel": `s_c_ec_t`, "Cone": `s_c_m_p`, "Cuboid": `s_s_c_na`, "Cylinder": `s_c_c_na`, "Truncated cone": `s_c_m_t`, "Handle": `c_c_c_na`, "Expanded Handle": `c_c_m_t`, "Horn": `c_c_m_p`, "Truncated pyramid": `s_s_m_t`.

---

[1] https://creativecommons.org/licenses/by-sa/4.0/legalcode

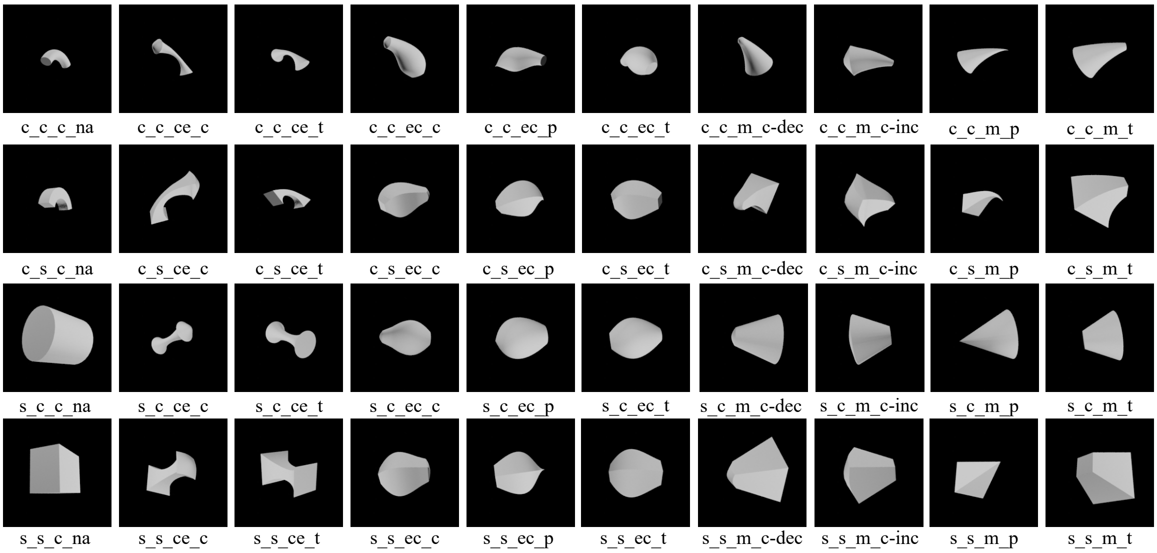


Figure 1: The list of 40 Geons we constructed.

2 Ethics considerations

While not intended to cause any harm, improved shape perception system could be misused to increase worker surveillance, causing detrimental effect on people's autonomy and privacy at work. Furthermore, when combined with face recognition algorithm, improved shape bias of vision models could further advance unethical use such as distinguishing faces of a certain ethic group from those of other ethnicity. We should also be cautious about overly relying on a single metric (e.g. shape bias) to evaluate vision models. For instance, increasing shape bias alone could lead to potential, unknown vulnerability, which could pose new security concerns.

3 Resubmission discussion

In the previous submission, we did not have results for DVR+AT, and only tested DVR-Last and DVR. Therefore, previous reviewers questioned the effectiveness of DVR for increasing robustness in terms of common corruptions and adversarial perturbations. After the retraction, we found that DVR+AT models consistently outperform vanilla AT in terms of common and adversarial robustness. Additionally, previous reviewers were concerned about the black background of Geon3D being too artificial. To address this, we perform experiments where we add texture from real images to the background. We found that 3D shape bias still helps improve robustness in this setting.

4 Additional Results

4.1 Adversarial Robustness

In Figure 2, we provide additional results for adversarial robustness, where we attack AT-L_2 using L_∞-PGD. Similar to the case of AT-L_∞, we see that 3D pretraining improves robustness over the vanilla AT models for all background settings.

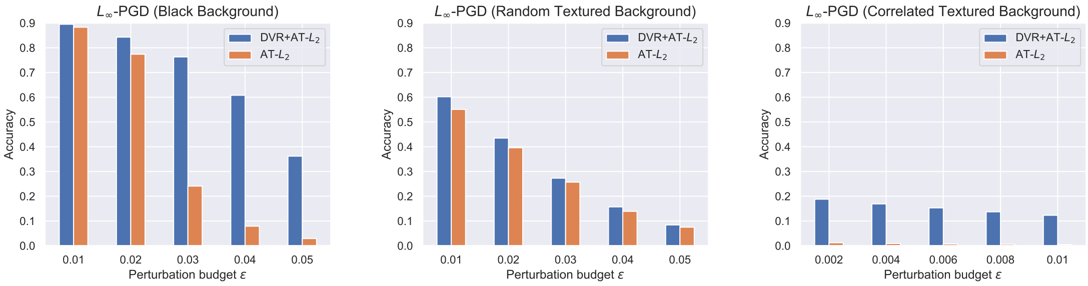


Figure 2: Robustness comparison between AT-L_2 and DVR+AT-L_2 with increasing perturbation budget ϵ on three variations of Geon3D-10. We attack our models using L_∞-PGD with 100 iterations and $\epsilon/10$ to be the stepsize.

4.2 Robustness to Common Corruptions

In this section, we provide additional results for common corruptions. In Table 1, we provide the results for the black background setting. Here again we see that 3D pretraining further improves vanilla AT models. In Table 2, we provide more detailed results of distributional shift in the backgrounds. Even after adding image corruptions, we still see that DVR+AT performs best, confirming that 3D shape bias from 3D pretraining complements the performance of AT to increase model robustness.

5 Details of DVR 3D reconstruction training

We provide details of the problem setup of 3D reconstruction, following [Niemeyer et al., 2020].

During training, we render an image, which is then used to minimize the RGB reconstruction loss. To render a pixel of an image observed by a virtual camera, we need to first find the world coordinate of the intersection of the camera ray with the object surface, and then map the world coordinate into a RGB color.

Let $u = (u_1, u_2)$ be the image coordinate of the pixel we want to render. To find the world coordinates of the intersection, we first parameterize the points along the camera ray $r_{p_0 \to (u_1,u_2)}$ by the distance d to the camera origin p_0 as follows:

$$r_{p_0 \to (u_1,u_2)}(d) = R^T \left(K^{-1} \begin{pmatrix} u_1 \\ u_2 \\ d \end{pmatrix} - T \right)$$

Table 1: Accuracy of shape-biased classifiers against common corruptions under unseen views on Geon3D-10 (black backgrounds).

	REGULAR	INFODROP	STYLIZED	AT-L_2	ATL_∞	DVR+AT-L_2	DVR+AT-L_∞
INTACT	0.866	0.845	0.822	0.908	0.910	0.912	**0.92**
PIXELATE	0.685	0.773	0.781	0.905	0.910	0.911	**0.919**
DEFOCUS BLUR	0.303	0.247	0.755	0.900	**0.909**	0.897	**0.909**
GAUSSIAN NOISE	0.548	0.291	0.803	0.620	0.885	0.914	**0.919**
IMPULSE NOISE	0.140	0.190	0.750	0.542	0.100	0.916	**0.918**
FROST	0.151	0.323	**0.783**	0.140	0.100	0.22	0.3
FOG	0.138	0.163	**0.764**	0.100	0.100	0.119	0.149
ELASTIC	0.612	0.635	0.617	0.628	0.664	0.645	**0.655**
JPEG	0.799	0.821	0.810	0.905	0.911	0.912	**0.92**
CONTRAST	0.510	0.180	**0.772**	0.163	0.258	0.213	0.335
BRIGHTNESS	0.552	0.832	0.818	0.160	0.137	0.385	**0.931**
ZOOM BLUR	0.475	0.462	0.748	0.891	0.917	0.902	**0.92**

Here, $R \in \mathbb{R}^{3\times3}$ is a camera rotation matrix, $T \in \mathbb{R}^3$ is a translation vector, and $K \in \mathbb{R}^{3\times3}$ is a camera intrinsic matrix. In the main paper, we denote $c^{ex} = [R, T]$, and $c^{in} = K$. Here, T is the position of the origin of the world coordinate system with respect to the camera coordinate system. Therefore, the position of the camera origin p_0 (w.r.t. the world coordinate system) is $-R^T T$.

Then we solve the following optimization problem:

$$\text{argmin} \quad d \quad \text{s.t.} \quad r_{p_0 \to (u_1,u_2)}(d) \in \Omega \tag{1}$$

where Ω is the set of points p in $\mathbb{R}^3$ such that $f_\theta(p) = 0.5$.

To solve for d, we start from the camera origin p_0 and step along the ray until object surface is intersected, which we can determine by evaluating the points along the ray via f_θ.

To summarize, we are given a set of object images $\{x_i \in \mathbb{R}^{H\times W\times3}\}_{i=1}^n$, their corresponding binary object masks $\{m_i \in \mathbb{R}^{H\times W}\}_{i=1}^n$, and extrinsic/intrinsic camera matrices $\{c_i = (c_i^{ex} \in \mathbb{R}^{3\times3}\times\mathbb{R}^3, c_i^{in} \in \mathbb{R}^{3\times3})\}_{i=1}^n$. Let $\mathcal{U}_0$ be a set of pixel points which lie inside the ground truth object mask and where the model predicts a depth. $\mathcal{U}_1$ is a set of points outside the object mask where the model falsely predicts depth. Finally $\mathcal{U}_2$ is a set of points inside the object mask where the model does not predict any depth. Then the objective is:

$$\begin{aligned}\underset{\phi,\theta,\theta'}{\text{argmin}}\, \mathbb{E}\Big[&\sum_{u\in\mathcal{U}_0} (||\hat{x}_u - x_u||_1 + \lambda_1 \mathcal{L}_{\text{normal}}(\hat{p}_{u,c}|z)) \\ &+ \lambda_2 \sum_{u\in\mathcal{U}_1} \text{BCE}(f_\theta(\hat{p}_{u,c}|z), 0) + \lambda_3 \sum_{u\in\mathcal{U}_2} \text{BCE}(f_\theta(p_{\text{rand}(u),c}|z), 1)\Big]\end{aligned}$$

Here, BCE stands for Binary Cross Entropy loss, and $\hat{p}_{u,c} = r_{p_0\to u}(\hat{d})$, where $\hat{d}$ is the predicted depth, provided as a solution to the optimization problem 1. The value of $p_{\text{rand}(u),c} = r_{p_0\to u}(d_{\text{rand}(u)})$, where the value of $d_{\text{rand}}(u)$ is chosen uniformly randomly on the ray to encourage occupancy for $u \in \mathcal{U}_2$. $\hat{x}_u = r_{\theta'}(\hat{p}_{u,c}|z)$ for $u \in \mathcal{U}_0$. $z = g_\phi(x_i^{(rand)})$, where we take a random view $x_i^{(rand)}$ from the same object instance as x_i.

Table 2: Accuracy of shape-biased classifiers against common corruptions under unseen views on Geon3D-10 with textured background swap.

	Regular	InfoDrop	Stylized	AT-L_2	AT-L_∞	DVR+AT-L_2	DVR+AT-L_∞
INTACT	0.045	0.121	0.268	0.015	0.311	0.219	**0.439**
PIXELATE	0.044	0.096	0.275	0.017	0.306	0.201	**0.415**
DEFOCUS BLUR	0.044	0.093	0.268	0.024	0.242	0.206	**0.338**
GAUSSIAN NOISE	0.046	0.160	0.269	0.015	0.320	0.209	**0.408**
IMPULSE NOISE	0.058	0.096	**0.228**	0.015	0.078	0.207	0.147
FROST	0.020	0.138	**0.255**	0.070	0.149	0.144	0.227
FOG	0.032	0.114	**0.273**	0.077	0.099	0.149	0.124
ELASTIC	0.044	0.109	0.260	0.100	0.196	0.176	**0.264**
JPEG	0.041	0.089	0.264	0.016	0.306	0.206	**0.419**
CONTRAST	0.055	0.107	**0.274**	0.066	0.090	0.148	0.126
BRIGHTNESS	0.036	0.127	0.268	0.026	0.270	0.189	**0.379**
ZOOM BLUR	0.081	0.082	0.290	0.032	0.269	0.249	**0.375**

$\mathcal{L}_{\text{normal}}(p|z)$ is the normal loss, which is a geometric regularizer to encourage smooth object surface. For a point $p \in \mathbb{R}^3$ and some object encoding z, the unit normal vector can be calculated by:

$$n_\theta(p|z) = \frac{\nabla_p f_\theta(p|z)}{||\nabla_p f_\theta(p|z)||_2}$$

We apply the l_2 loss to minimize the difference between the normal vectors at p and p', where p' is in a small neighbourhood around p. Formally,

$$\mathcal{L}_{\text{normal}}(p|z) = ||n_\theta(p|z) - n_\theta(p'|z)||_2$$

for a point $p \in \mathbb{R}^3$.

6 Reproducibility: Training details

We used GeForce RTX 2080Ti GPUs for all of our experiments. GQN training takes about a week until convergence on a single GPU. DVR 3D reconstruction training takes roughly about 1.5 days on a single GPU. The hyperparameters for 10-Geon classification, described in the main paper, were chosen by monitoring the model convergence on the validation set. All the other results are from a single training run and a single evaluation run.

DVR We used the code [2] open-sourced by Niemeyer et al. [2020]. We followed the default hyperparameters recommended by Niemeyer et al. [2020] for 3D reconstruction training, with the exception of batch size, which we set 32 to fit into a single GPU memory.

Adversarial Training We used the python package [3] to perform adversarial training. For AT(L_2), we use attack steps 7, epsilon 3.0, attack lr 0.5. For AT(L_∞), we use attack steps 7, epsilon 0.05,

---

[2] `https://github.com/autonomousvision/differentiable_volumetric_rendering`

[3] https://github.com/MadryLab/robustness

attack lr 0.01. use best (final) PGD step as example. Both models trained for 70 epochs with batch size 100, which was sufficient for model convergence.

GQN We used the open-source code [4] to implement our GQN. Due to the training instability, we rescale the image size from 224 x 224 to 64 x 64.

InfoDrop We used the original author's implementation [5].

Stylized To stylize Geon3D, we used the code [6] introduced by the original author of Stylized-ImageNet Geirhos et al. [2018].

Dataset For training Geon3D image classifiers, we center and re-scale the color values of Geon3D with $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$, which is estimated from ImageNet. We construct the 40 3D model instances as well as the whole training data in Blender. We then normalize the object bounding box to a unit cube, which is represented as `1.0_1.0_1.0` in the dataset folder.

Background textures We used the following label-to-texture class mapping: {0: 'zigzagged', 1: 'banded', 2: 'wrinkled', 3: 'striped', 4: 'grid', 5: 'polka-dotted', 6: 'chequered', 7: 'blotchy', 8: 'lacelike', 9: 'crystalline' }. For the distributional shift experiment we used the following mapping: { 0: 'crystalline', 1: 'zigzagged', 2: 'banded', 3: 'wrinkled', 4: 'striped', 5: 'grid', 6: 'polka-dotted', 7: 'chequered', 8: 'blotchy', 9: 'lacelike', }. The DTD data is licensed under the Creative Commons Attribution 4.0 License. [7]

Evaluation set For all the evaluation sets in the experiment section, we used the same subset of the test split, where we randomly pick 1000 model instance ids, and randomly sample 1 view out of 50 views for every model instance.

We use the original author's code [8] to generate common corruptions shown in Figure 3.

References

Barr. Superquadrics and Angle-Preserving Transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, Jan. 1981. ISSN 1558-1756. doi: 10.1109/MCG.1981.1673799.

O. C. Blender. Blender - a 3D modelling and rendering package. Blender Foundation, 2021.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, Sept. 2018.

M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations Without 3D Supervision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3512, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5. doi: 10.1109/CVPR42600.2020.00356.

---

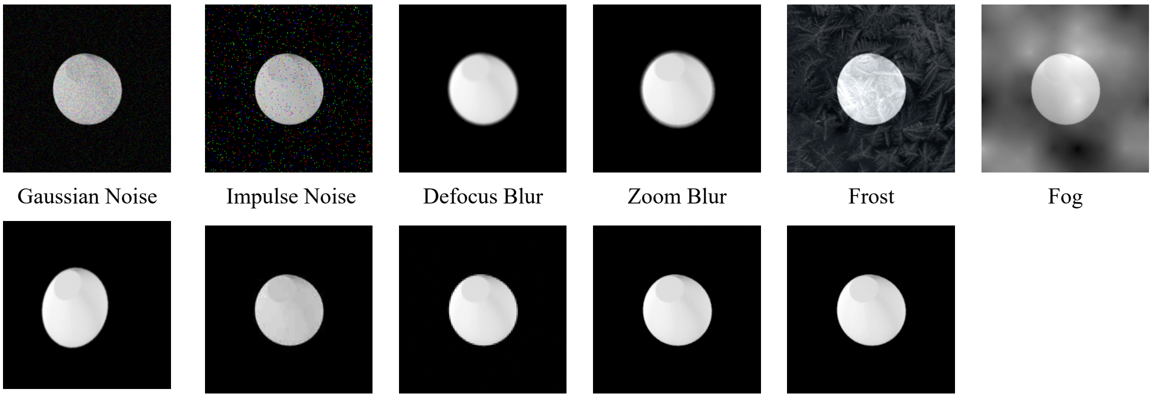
[4] https://github.com/iShohei220/torch-gqn

[5] https://github.com/bfshi/InfoDrop

[6] https://github.com/bethgelab/stylize-datasets

[7] https://creativecommons.org/licenses/by/4.0/, https://www.tensorflow.org/datasets/catalog/dtd

[8] https://github.com/hendrycks/robustness



Gaussian Noise Impulse Noise Defocus Blur Zoom Blur Frost Fog

Elastic Transform JPEG Compression Pixelate Brightness Contrast

Figure 3: Examples of image corruptions.

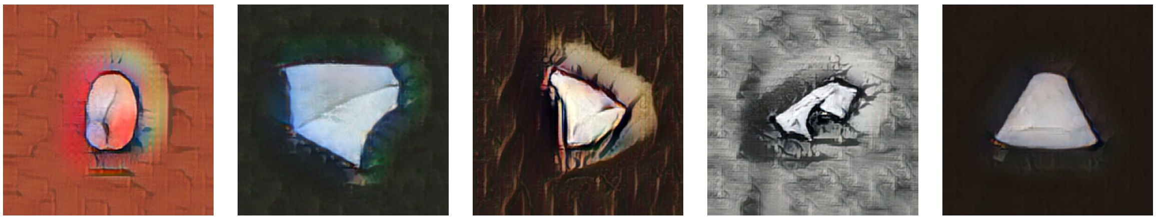


Figure 4: Examples of Stylized Geon