BigDatasetGAN: Synthesizing ImageNet with Pixel-wise Annotations

Supplementary Material

A. BigDatasetGAN Implementation Details

A.1. BigGAN

Training. When training the segmentation branch of BigGAN, we first load the pre-recorded latent codes of the annotated samples. We then pass the latent codes as well as the class information into the pre-trained BigGAN model (with resolution 512×512) to extract the features. With BigGAN features and class information (input to BigGAN), we can train the segmentation branch with the binary cross-entropy loss on the human-annotated segmentation masks. Note that in this case we exploit the fact that only foreground object and background scene are annotated in each image and the class information is known. We use the Adam [9] optimizer with a learning rate 0.001 and batch size of 8. We train the segmentation branch with 5k iterations.

Uncertainty Filtering. As mentioned in Sec 3.2 in the main paper, we apply uncertainty filtering on top of the truncation trick [2] and rejection sampling [12]. We train 16 segmentation heads, and follow [10] and [16] to use the Jensen-Shannon (JS) divergence as the uncertainty measure. Concretely, the JS divergence is defined as:

$$JS(P_1, P_2, ..., P_N) = H\left(\frac{1}{N}\sum_i^N P_i\right) - \frac{1}{N}\sum_i^N H(P_i),$$

where N is the number of models, in our case is 16. H is the entropy function and P_i is the output distribution of model i. We record the uncertainty score for each sample and then follow [16] to filter out the 10% most uncertain image samples.

Ablation Study on Truncation Threshold. Here, we study the effect of the truncation threshold. We sample datasets with different truncation thresholds ranging from 0.6 to 2.0. Higher truncation thresholds increase the diversity of the dataset but lower image fidelity. In some cases, if the truncation threshold is too high, noisy images are generated. We show the downstream task performance on our ImageNet segmentation task for the dog class. We report segmentation mIoU as well as FID (5k samples) to measure sampled image quality for different truncation thresholds in Table 1. We find a sweet-spot for both downstream task performance and image sample quality at the truncation threshold 0.9 and we use this truncation threshold for all experiments.

Truncation Threshold	*ImageNet-Dog segmentation mIoU*	*FID-5k*
0.6	84.2	22.35
0.8	86.4	18.42
0.9	87.0	14.95
1.0	86.0	15.54
1.5	87.5	19.57
2.0	86.6	39.52

Table 1. **Ablation study on truncation threshold**. Here, we measure downstream task performance in mIoU and sampled image quality in FID-5k. Higher truncation rates increase the image diversity, which is beneficial for downstream task performance, but can hurt image quality as measured in FID-5k.

A.2. VQGAN

Architecture Details. We use VQGAN [6] trained on ImageNet at 256×256 resolution. VQGAN's class-conditional transformer consists of 48 self-attention [13] layers, each with 1536 dimensions, operating on the 16×16 codebook with the vocabulary size of 16384. We gather features from every fourth transformer layer for each spatial location (16×16) of the encoder output. Additionally, we gather features from the decoder layers at 16×16 to 256×256 resolutions. In total, F_{VQGAN} consists of 12 transformer features and 7 decoder features. Similarly to BigGAN, we group features according to their spatial resolutions into a high-level group (16×16 to 32×32), a mid-level group (64×64 to 128×128), and a low-level group (256×256). We reduce the dimensionality of each feature to 128 using a `1x1conv` operation and then `upsample` features within the same group into the highest resolution within the group. Features from two separate levels are fused by upsampling the features of the previous, higher level group to match the current layer's spatial resolution and then concatenating the two levels, followed by a `mix-conv` operation which contains a `3x3conv` layer. Finally, three `1x1conv` layers are used to output the segmentation logits. We use layer normalization [1] between the `1x1conv` layers.

Training. To train the segmentation branch for VQGAN, we leverage its encoder to encode the images that were previously obtained via BigGAN sampling (with resolution 256×256). We can then extract the features as described above and obtain the segmentation logits. Additionally, the class-conditional transformer takes in the class label as input. We use the same binary cross-entropy loss for the foreground and background segmentation as in BigGAN training. We use the Adam [9] optimizer with a learning rate of 0.001 and batch size of 8.

B. ImageNet Segmentation Benchmark Details

B.1. Dataset Statistics

Table 2 provides the dataset split details for the 7 tasks in the benchmark. Please refer to Sec 5.1 in the main paper for the dataset details and Sec 5.2 for details about benchmark

	Dog	*Bird*	*FG/BG*	*MC-16*	*MC-100*	*MC-128*	*MC-992*
train	657	366	5294	1268	540	5294	5294
test	1040	512	8316	1967	798	8316	8316

Table 2. **ImageNet segmentation benchmark splits.** The training set is based on *Synthetic-annotated* (Images sampled from BigGAN), while the testing set consists of images from *Real-annotated*.

	Imagenet-MC-16 (*train*)	MS-COCO-MC-16 (*test*)
Sup.IN	58.7	25.8
BigGAN-off	64.6	29.2

Table 3. **Off-the-shelf usage of synthetic labels.** Here we compare semantic segmentation performance when training on the ImageNet-MC-16 task and testing on MS-COCO images with the same classes. Comparing to the baseline method Sup.IN, which does not use any synthetic datasets generated by our method, BigGAN-off shows higher in-domain performance as well as out-of-domain test performance. Numbers are measured in mIoU.

tasks. We plan to provide a detailed class mapping when launching the public benchmark.

B.2. Training Setup

For all the baselines and our methods, we use DeepLab-v3 [3] with Resnet-50 [8] as the image backbone model. We use the SGD optimizer with learning rate 0.01, momentum 0.9 and weight-decay 0.0001. We use polynomial learning rate decay with power 0.9. We use batch size 64 for all tasks and train for 200 epochs. For augmentation, we use *random resize* with scales from $0.5 - 2.0$, *random crop* and *random horizontal filp*. We use resolution 224 for both training and testing. When training with our synthetic dataset, we use the same training schedule and augmentation policy as described here.

As objective, we use the cross-entropy loss function for all tasks except for *MC-992*. The task *MC-992* needs to segment over 992 classes, where the object pixel distribution is varying greatly between different classes. This corresponds to an imbalanced training setup, which makes learning the model difficult. We found that training did not converge well using the standard cross-entropy loss. In order to mitigate this issue, we use the focal loss [11] for all the methods in task *MC-992*.

B.3. Off-the-shelf Usage of Synthesized Labels

In our experiments, we primarily use the synthesized labeled datasets for pre-training. However, we can also directly use them "off-the-shelf" for downstream task training. In order to show the off-the-shelf usage of our synthesized labels from ImageNet, we prepared a filtered dataset from MS-COCO with the same 16 classes as in our ImageNet segmentation benchmark task MC-16. We compare semantic segmentation performance when training on the ImageNet-MC-16 task and directly test on the MC-COCO data (denoted as MS-COCO-MC-16) without any fine-tuning. Our method BigGAN-off shows better performance than standard supervised training in both the in-domain (ImageNet training data) and out-of-domain (MC-COCO) test settings (Table 3).

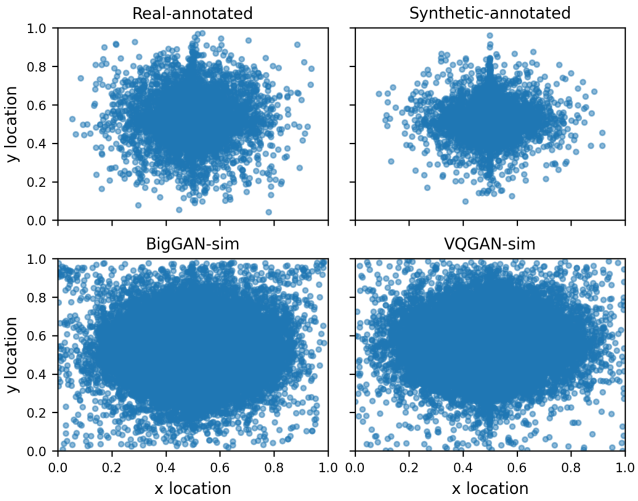

Figure 1. **Object center scatter plots.** We visualize center positions of the tight bounding box of pixel-wise labels for the *Real-annotated*, *Synthetic-annotated*, *BigGAN-sim* and *VQGAN-sim* datasets. The box center is normalized with respect to the original image size.

C. Dataset Analysis Details

C.1. Center Distributions

We visualize scatter plots of object center locations for the *Real-annotated*, *Synthetic-annotated*, *BigGAN-sim* and *VQGAN-sim* datasets in Figure 1. We fit a tight bounding box over the segmentation mask and use the center of the bounding box as the center location. Note that the location is normalized with respect to the original image size. We see that most of the object centers are biased towards the image center for all datasets. Compared to the human-annotated datasets, *BigGAN-sim* and *VQGAN-sim* show noisier center distributions, in particular towards the image corners.

C.2. Geometry Metrics

In Table 1 in the main paper, we compute geometry metrics to measure shape complexity (SC) and shape diversity (SD) of the human-annotated dataset as well as the synthetic dataset. Here, we provide the corresponding implementation details.

Implementation.. We first filter out masks with a total number of pixels below 100 to avoid noisy labels. We then calculate the connected components in the mask and only use the largest connected component in our measurement.

We further use OpenCV's `findContours` function with the `CHAIN_APPROX_SIMPLE` flag which compresses horizontal, vertical, and diagonal segments and leaves only their end points to extract a simplified polygon. Note that the simplification happens in pixel space and the extracted polygons do not have the same scales. In order to overcome such scaling issues, we normalize the extracted polygon to a unit square by $p_i = (p_i - p_{min})/(p_{max} - p_{min})$ (this operation is applied separately for both horizontal and vertical directions in an image), where p_i is the point in the extracted polygon, and p_{min} and p_{max} are the minimum and maximum point coordinates (for either horizontal or vertical direction) over the set of points for a given polygon. We further apply the Douglas-Peucker algorithm [5] with a threshold of 0.01 to remove redundant points.

Shape Complexity. After applying the pre-processing steps mentioned above, the shape complexity (SC) is calculated as the number of points of the normalized simplified polygon. We also measure polygon length (PL) in Table 1 from the main paper.

Shape Diversity. We calculate the shape diversity by mean pair-wise Chamfer distance [7] per class and average across classes. Specifically, Chamfer distance is defined as

$$d_{\text{CD}}(S_1, S_2) = \sum_{\mathbf{x} \in S_1} \min_{\mathbf{y} \in S_2} ||\mathbf{x} - \mathbf{y}||_2^2 + \sum_{\mathbf{y} \in S_s} \min_{\mathbf{x} \in S_1} ||\mathbf{x} - \mathbf{y}||_2^2$$

where S_1 and S_2 are sets of points corresponding to different polygons. In our case, we calculate average pair-wise d_{CD} between all polygons within each class. Then, we compute the shape diversity (SD) metric as the average distance over all classes.

D. Transfer Learning Experiments

D.1. Pre-training Implementation

We closely follow the pre-training setup of DenseCL [14], and use SGD as the optimizer with weight decay and momentum set to 0.0001 and 0.9, respectively. The dictionary size is set to 65536 and the momentum update rate of the encoder is set to 0.999. We also adopt its augmentation policy with 224×224 random resized cropping, random color jittering, random gray-scale conversion, Gaussian blurring and random horizontal flips for contrastive learning. Note that since texture and color are important hints for semantic segmentation, we only use random resized cropping with resolution 224×224 and random horizontal flip when training the segmentation branch with our synthetic dataset. The batch size is 256, and we train on 8 GPUs with a cosine learning rate decay schedule. We pre-train the backbone on ImageNet using the original contrastive losses for 150 epochs and then jointly train the segmentation branch using the cross-entropy loss for another 50 epochs.

D.2. Downstream Task Details

Benchmark Implementation. For object detection and instance segmentation tasks on Pascal VOC, MS-COCO and Cityscapes, we use standard benchmark configurations from OpenSelfSup[1]. The model is trained using Detectron2 [15]. For semantic segmentation tasks on Pascal VOC and Cityscapes, we use configurations from DenseCL[2] implemented in MMSegmentation [4].

E. Qualitative Results

We show random samples from the human-annotated and synthetic datasets. See Figure 2 for dataset samples from the *Real-annotated* dataset where annotation is done on real images. Figure 3 shows dataset samples from the *Synthetic-annotated* dataset where human annotation is on BigGAN generated images. For our synthetic dataset *BigGAN-sim* generated by BigGAN, see Figure 4, and for the *VQGAN-sim* dataset generated by VQGAN, see Figure 5. Compared to the synthetic dataset, *Real-annotated* images have more complex backgrounds and structure. However, we also see that the synthetic datasets generated by BigGAN and VQGAN include photo-realistic and diverse images as well as high quality labels.

We also show per-class samples where images in the same row are from the same class. For *BigGAN-sim* per-class samples, please see Figure 6. For *VQGAN-sim* per-class samples, see Figure 7. Note that we select the same classes for both *BigGAN-sim* and *VQGAN-sim* for easy comparison. Comparing to *BigGAN-sim*, the *VQGAN-sim* dataset samples are more diverse in terms of object scale, pose as well as background. However, we see *BigGAN-sim* has better label quality than *VQGAN-sim* where in some cases the labels have holes and are noisy.

We also include mean shape visualizations for different classes from the *BigGAN-sim* dataset (see Figure 8). We first use a tight bounding box to fit the segmentation, and then crop and resize the segmentation into resolution 32×32. We use k-means clustering with $k = 5$ for each class to calculate the major modes of the resized segmentation mask. In Figure 8, we randomly select 500 classes and for each class we randomly select one cluster out of the 5 clusters and visualize the mean shape. We see diverse shapes with different poses especially for classes related to animals. Some classes do not have clear shapes. This might be because the randomly selected mode is potentially not the most meaningful mode.

[1] https://github.com/open-mmlab/OpenSelfSup
[2] https://github.com/WXinlong/DenseCL

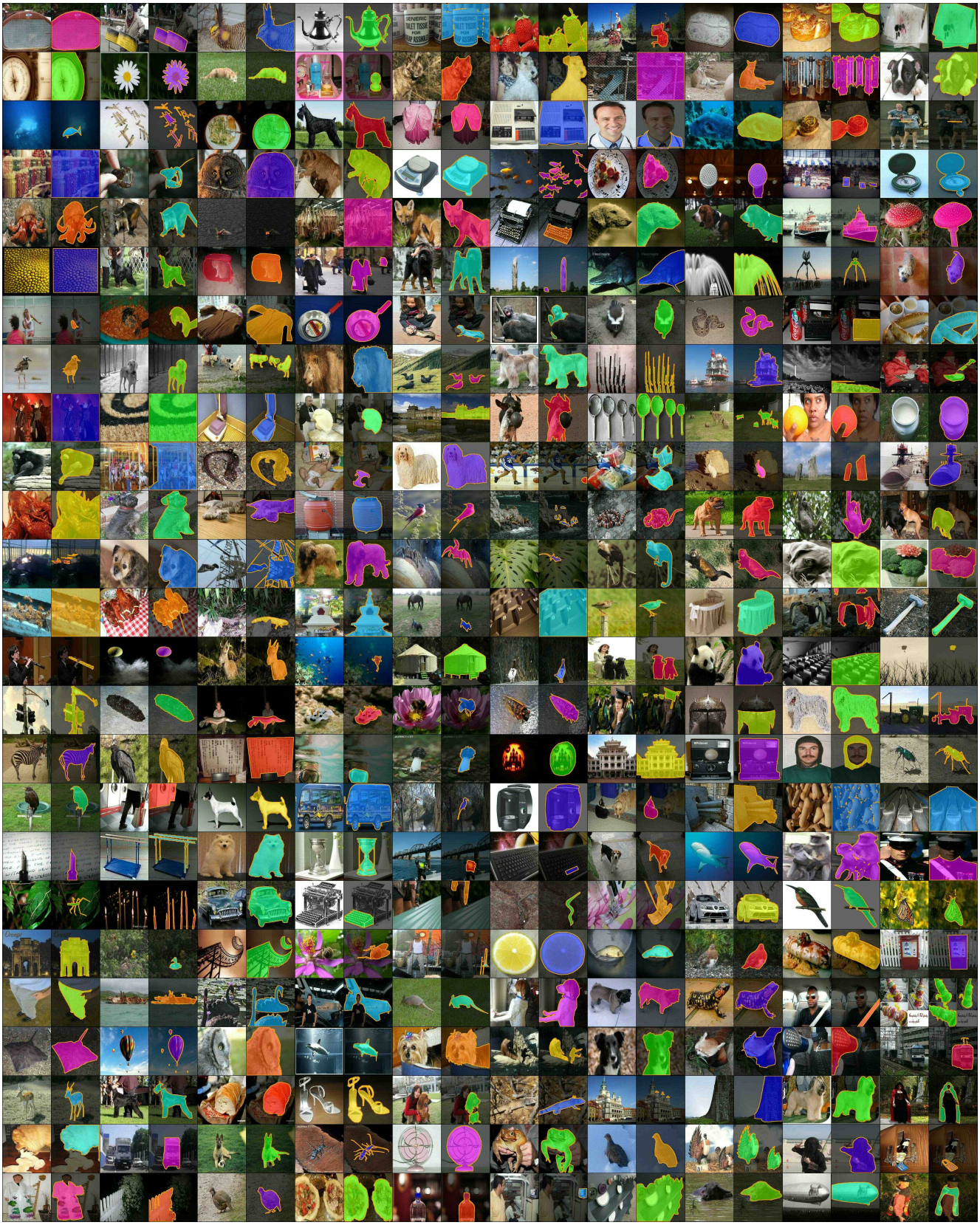

Figure 2. **Examples from the Real-annotated dataset.** We visualize both the segmentation masks as well as the boundary polygons.

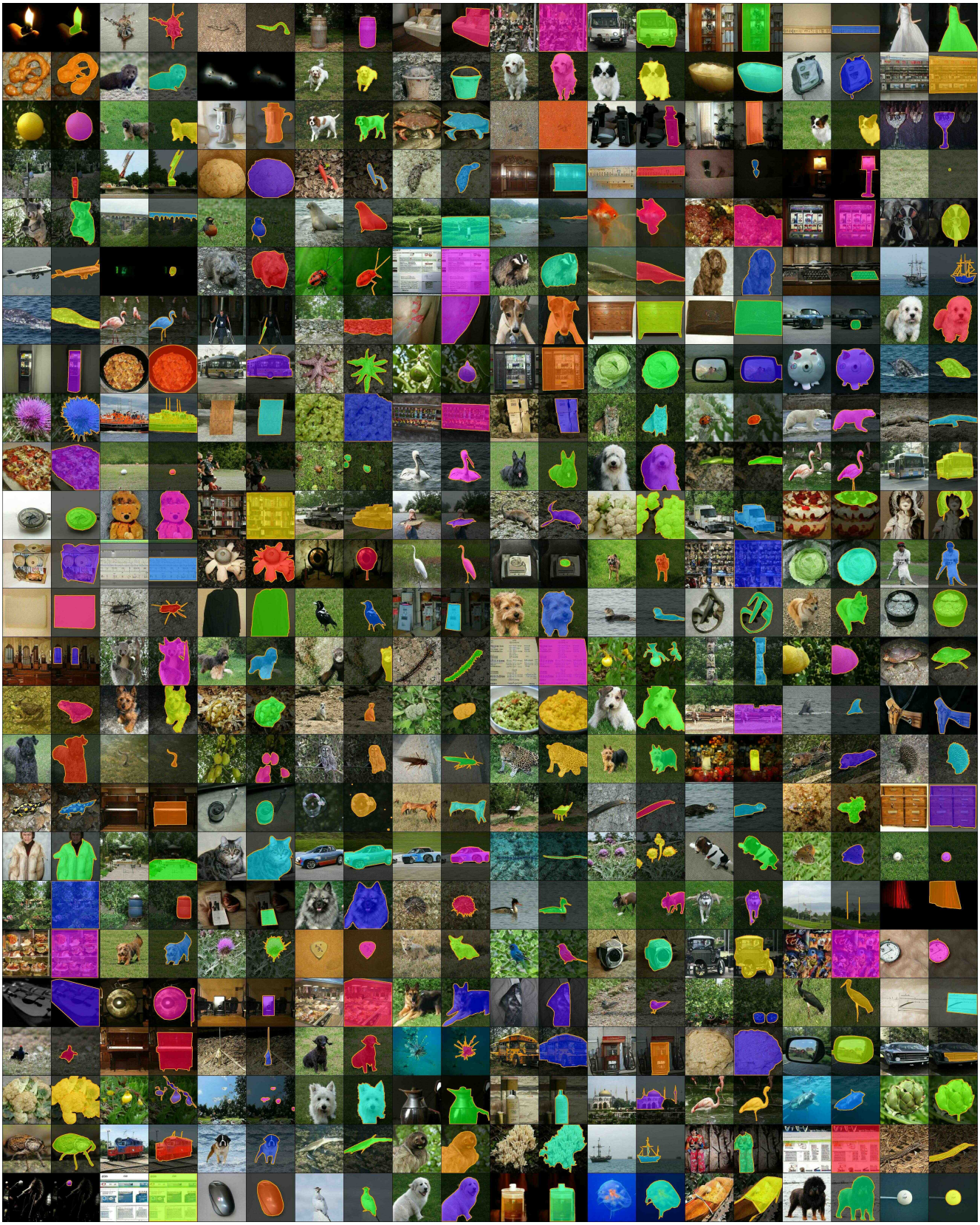

Figure 3. **Examples from the Synthetic-annotated dataset.** We visualize both the segmentation masks as well as the boundary polygons.

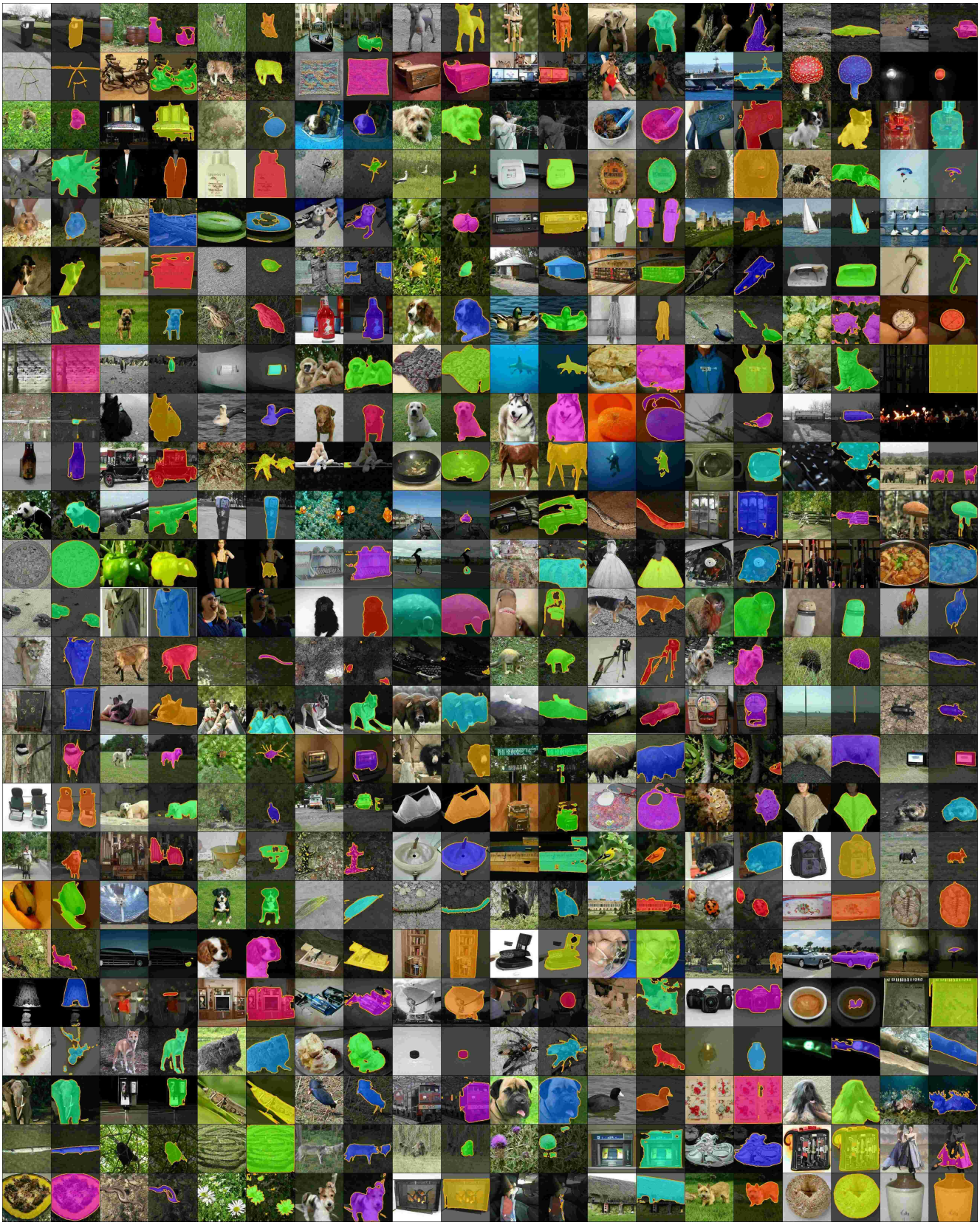

Figure 4. **BigGAN-sim random samples.** We visualize both the segmentation masks as well as the boundary polygons.

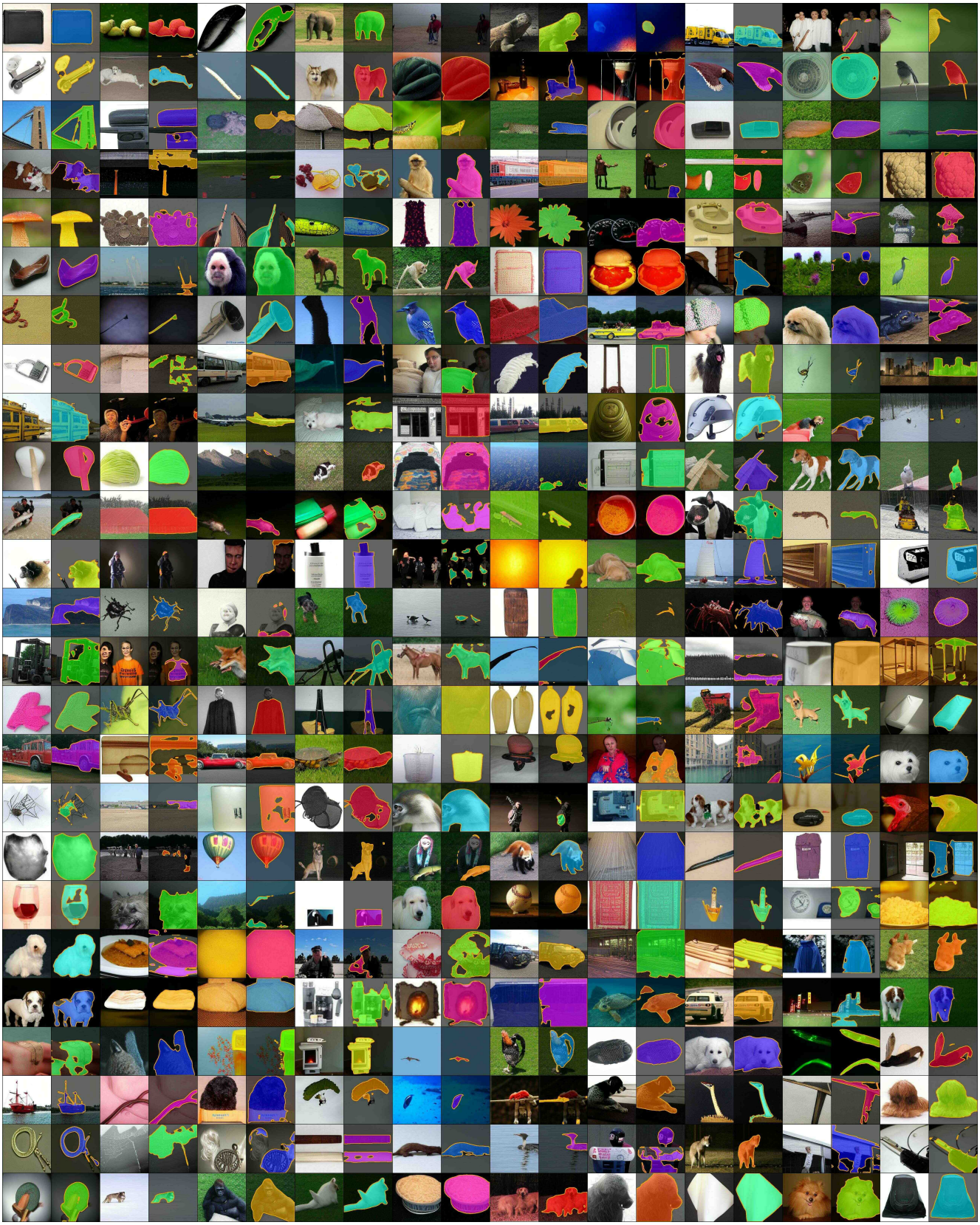

Figure 5. **VQGAN-sim random samples.** We visualize both the segmentation masks as well as the boundary polygons.

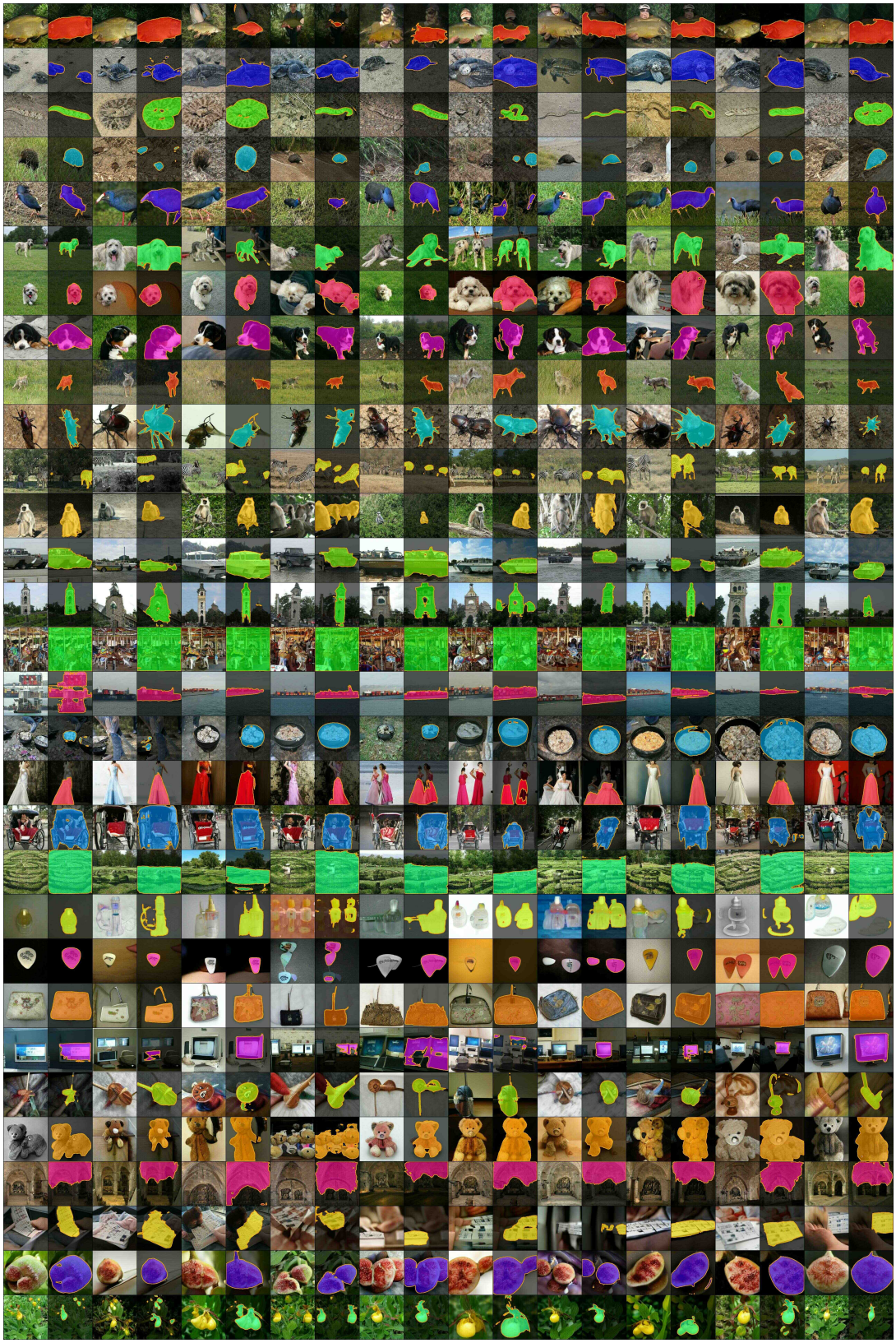

Figure 6. **BigGAN-sim per-class samples.** We visualize both the segmentation masks as well as the boundary polygons.

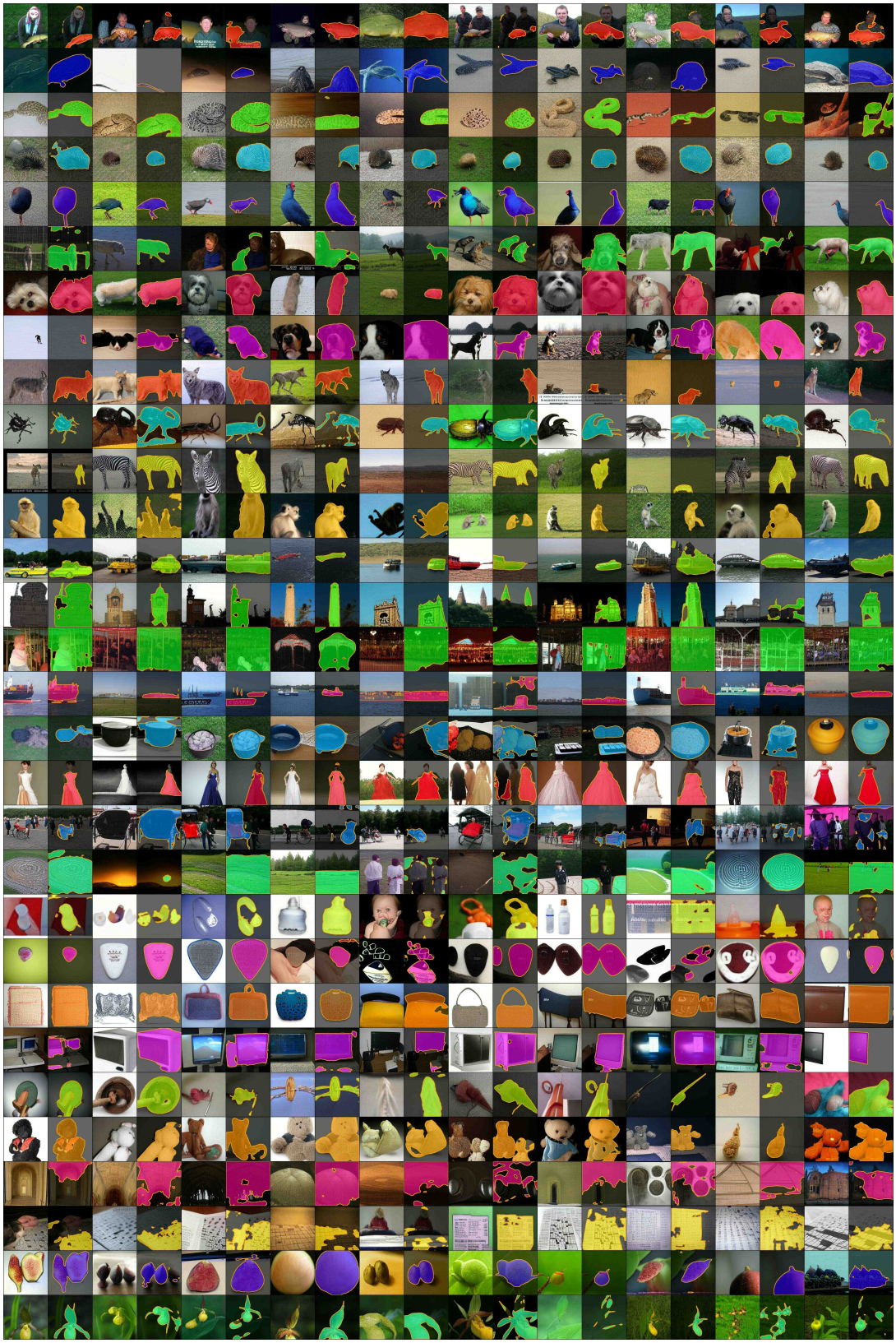

Figure 7. **VQGAN-sim per-class samples.** We visualize both the segmentation masks as well as the boundary polygons.

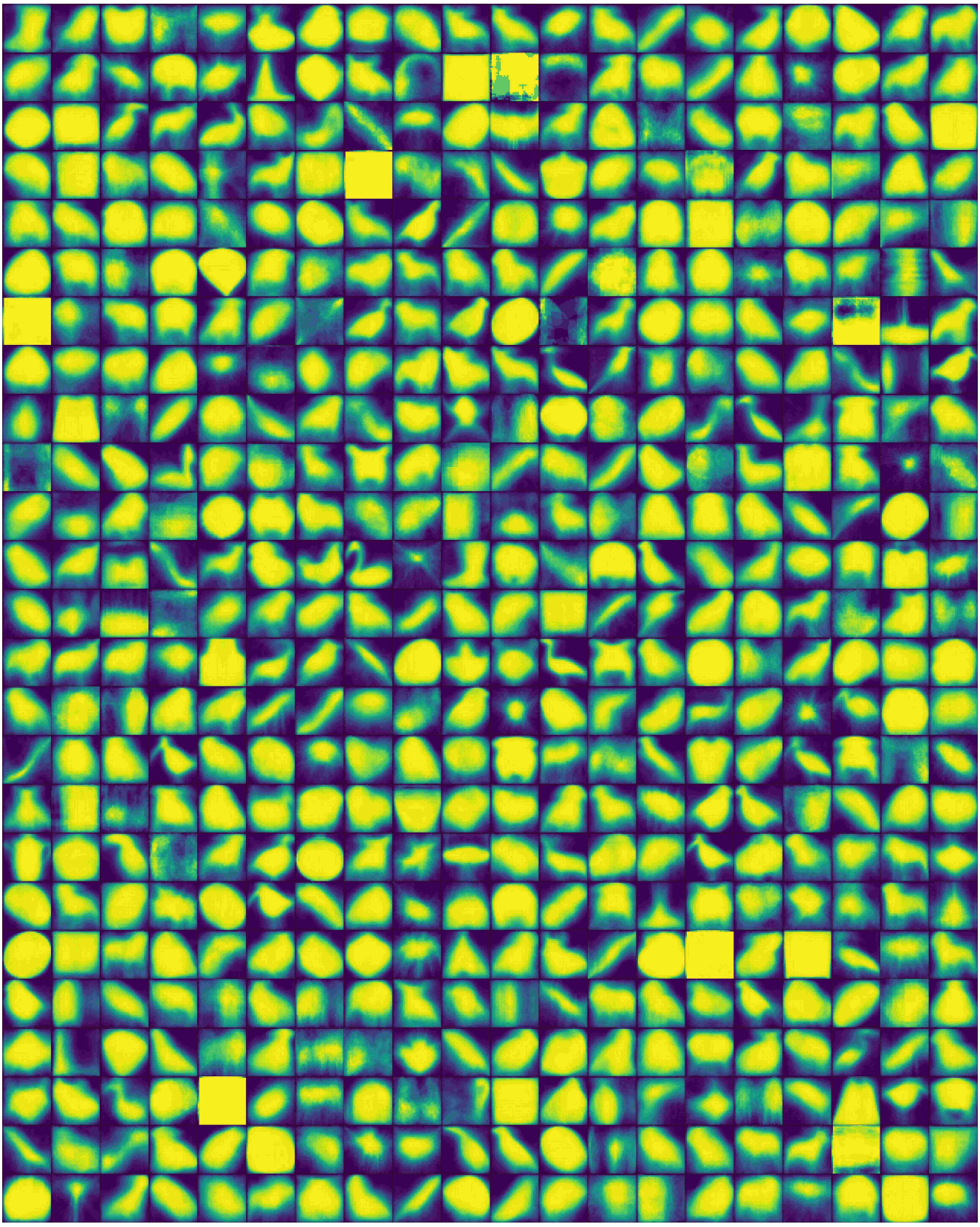

Figure 8. **BigGAN-sim mean shapes.** Mean shapes are calculated using k-means clustering over normalized segmentation masks.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2019. 1

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017. 2

[4] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 3

[5] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973. 3

[6] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020. 1

[7] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. *arXiv preprint arXiv:1612.00603*, 2016. 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 2

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017. 1

[10] Weicheng Kuo, Christian Häne, Esther Yuh, Pratik Mukherjee, and Jitendra Malik. Cost-sensitive active learning for intracranial hemorrhage detection. In *MICCAI*, 2018. 1

[11] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2018. 2

[12] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019. 1

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1

[14] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. *arXiv preprint arXiv:2011.09157*, 2021. 3

[15] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 3

[16] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *CVPR*, 2021. 1