Quantifying the Limits of Segment Anything Model: Analyzing Challenges in Segmenting Tree-Like and Low-Contrast Structures

Yixin Zhang[1*]   Nicholas Konz[1*]   Kevin Kramer[5]   Maciej A. Mazurowski[1,2,3,4]

[1] Department of Electrical and Computer Engineering,   [2] Department of Computer Science,
[3] Department of Radiology,   [4] Department of Biostatistics & Bioinformatics,
Duke University, NC, USA
[5] Minnesota Health Solutions

{yixin.zhang7, nicholas.konz, maciej.mazurowski}@duke.edu, kevin@minnhealth.com
Code: https://github.com/mazurowski-lab/SAM-TexturalConfusion-Metrics

Abstract

Segment Anything Model (SAM) has shown impressive performance in interactive and zero-shot segmentation across diverse domains, suggesting that they have learned a general concept of "objects" from their large-scale training. However, we observed that SAM struggles with certain types of objects, particularly those featuring dense, tree-like structures and low textural contrast from their surroundings. These failure modes are critical for understanding its limitations in real-world use. In order to systematically examine this issue, we propose metrics to quantify two key object characteristics: tree-likeness and textural separability. Through extensive controlled synthetic experiments and testing on real datasets, we demonstrate that SAM's performance is noticeably correlated with these factors. We link these behaviors under the concept of "textural confusion", where SAM misinterprets local structure as global texture, leading to over-segmentation, or struggles to differentiate objects from similarly textured backgrounds. These findings offer the first quantitative framework to model SAM's challenges, providing valuable insights into its limitations and guiding future improvements for vision foundation models.

Introduction

Segment Anything Model (SAM) [27], the most widely-used foundation model for image segmentation, has demonstrated promising zero-shot and fine-tuned segmentation ability for objects not seen in training, within a wide range of domains since its release, such as biomedical images [10, 22, 30], remote sensing [6, 35], etc. However, cer-

Figure 1. SAM's segmentation performance tends to drop noticeably when the object has high tree-likeness (left, on DIS) or low textural separability (right, on iShape)–even with heavy prompting–which we investigate in this work. From left to right: input images, object annotations, and their SAM segmentations.

tain *failure modes* for SAM have been found empirically, where SAM produces surprisingly underwhelming performance on datasets with specific uncommon objects or atypical contexts, such as those with dense, branching structures (*e.g.*, retinal blood vessels) [11, 36], concealed/low-contrast objects [7, 24, 39], small or irregular objects, and others [24][1]. A solution to this problem could simply be fine-tuning the model on such challenging examples using various strategies [17, 29, 32], but this would still lack an understanding of the underlying causes of the issues, and so may not result in the best solution. As such, it is important to develop a better theory for why these failures happen, in order to better understand (and potentially mitigate) the limitations of SAM for new applications.

In this paper, we propose that several of these failure modes can be explained as SAM misinterpreting textural cues for how it disambiguates an object from its surround-

---

*Equal contribution.

[1]We focus on "unintentional" failure modes of SAM, as opposed to intentional adversarial attacks/examples [45, 47].

ings: it typically confuses objects with irregular, dense tree-like local structure as large-scale textures (resulting in over-segmentation), and similarly is generally challenged by objects that have similar textures to their surroundings. We propose metrics developed from first principles for quantifying the characteristics of (1) object tree-likeness and (2) textual separability. Objects tree-likeness (Sec. 2.2) is based on concepts such as spatial contour frequency and the difference in typical object variability between global and local scales. Textural separability (Sec. 3.1) is based on the detectability of differences in textural features of an object compared to its surroundings. We designed these metrics to be simple and interpretable, such that they can be applied to any dataset or object type. We also present them in quickly-computable PyTorch-like algorithmic form that can be applied to any masks.

Next, we begin to probe the effect that these characteristics have on SAM's segmentation ability through experiments on synthetic data which are designed to cover a range of object tree-likeness (Sec. 2.3.1) and textural separability (Sec. 3.2.1). These experiments are carefully controlled to disentangle the effects of object shape and texture on performance, as well as mitigate any other potential confounding factors. We then continue with similar experiments on real datasets (Secs. 2.3.2 and 3.2.2) that represent a wide range of objects to segment. In all experiments, we find both the exact measured tree-likeness and the textural separability of objects are noticeably correlated with SAM's performance (IoU) in segmenting it, over a wide range of object and image types. Our findings are the first to quantitatively model and experimentally verify how the performance of segmentation foundation models is affected by certain measurable object characteristics, providing an explanation for prior findings of SAM's unexpected failures on new datasets.

Our overall contributions are as follows:

1. Based on first principles, we propose quantitative metrics for object tree-likeness of Contour Pixel Rate (**CPR**) and Difference of Gini Impurity Deviation (**DoGD**), as well as a metric for object **textural separability**.
2. We present carefully-designed experiments for the segmentation of synthetic images which cover a range of tree-likeness and textural separability, where important factors such as object shape and texture are controlled.
3. We show that SAM's segmentation performance tends to negatively correlate with an object's tree-likeness, and positively correlate with it's textural separability, on a variety of experiments on both synthetic and real images.

We hope that this inspires future work in understanding the limitations of vision foundation models. We release easy-to-use code for our proposed metrics at https://github.com/mazurowski-lab/SAM-TexturalConfusion-Metrics.

Related Works

In addition to the various "failure modes" and biases of *segmentation foundation models* discussed in the introduction which motivated this work, various works have also studied the same for *general vision models*. Central to these is the goal of understanding the visual features that networks use to make predictions from images, and whether object *shape* or *texture* is more important. This is important to study because any unexpected failure of such models will depend on the features used for inference.

Zhang et al. [46] posited that objects within images can be described by three characteristics: shape, texture, and the composition of textures within shapes, and analyzed how these characteristics may affect the challenge of segmenting such objects. It is well known that convolutional neural networks' (CNNs') predictions are typically biased towards the texture of objects rather than their shape, for both classification [3, 16, 21, 38] and segmentation [46], resulting in curious examples where network predictions are based on texture even when the shape clearly defines a different object. For example, [28] found that when a CNN is presented with images of objects with only their shape visible, not texture (silhouettes), performance was greatly worsened.

It has also been found that the shape-learning behavior of neural networks can differ noticeably depending on the particular training dataset and objective [20, 21], motivating us to study the specific behavior of SAM (given its' unique training dataset and pipeline) in detail. Interestingly, there has been some evidence that vision transformers (like SAM's image encoder) may instead be biased towards shape, rather than texture for classification tasks [40]. Here, we study segmentation rather than classification, where it is unclear if those results extend, as our results generally point to SAM focusing on texture over shape (to the point of even confusing complex, locally-varying shape itself as texture).

1. Methods: SAM Usage and Prompting

We experiment with both the ViT-H (default) and ViT-B standard pretrained SAM models [27]. As is default for SAM, all input images are resized to a resolution of 1024×1024, and normalized to $[0, 255]$. All images that possess instance segmentations for multiple objects (*e.g.*, iShape and Plittersdorf) will be evaluated by SAM segmenting each of the images' objects one at a time (guided by clear prompting), with all other mask pixels set as background. In all experiments, we use the oracle-chosen mask out of SAM's three outputted predictions.

For all experiments on the synthetic images of Sec. 2.3.1, we provide a tight bounding box about the object of interest as the prompt. For all experiments on real images as well as the style-transferred images of Sec. 3.2.1, we provide the same bounding box as well as a certain number of posi-

tive and/or negative point prompts randomly sampled from the object foreground and background *within the bounding box*, respectively, depending on the dataset (full details in Appendix C.1). We use this relatively heavy prompting strategy in order to minimize the ambiguity of instructions provided to SAM, which will also help to minimize any differences of the oracle prediction from the other two.

2. The Challenge of Tree-Like Structures

2.1. Motivation

We first wished to understand SAM's challenges in segmenting certain objects with dense, tree-like structures, following our observation of a trend of SAM having difficulty with such objects including retinal blood vessels [29, 34] or satellite road images [14, 42] (see *e.g.*, Fig. 2). Interestingly, while both of these object types have branching features, a characteristic which naively could relate to SAM's failure, SAM's performance on retinal vessel images is noticeably worse (avg. IoU $\simeq 0.05$ [34]) compared to on satellite road structures (avg. IoU $\simeq 0.2$ [14]), which was reproduced in our own experiments (Fig. 4).

Figure 2. Example retinal blood vessel (top) and satellite road (bottom) images and accompanying object segmentation masks.

This motivated us to quantify the features of these types of objects that relate to SAM's failure. We hypothesize that such objects become more challenging to segment when their branching structures are **dense** and **irregularly-spaced**, which we refer to as "tree-like". We will show that SAM considers such structures as textures rather than shapes, resulting in significant over-segmentation. In the following section we propose how to characterize the tree-likeness of objects using two new quantitative metrics: CPR (Contour Pixel Rate) and DoGD (Difference of Gini Impurity Deviation).

2.2. Quantifying Tree-Like Structures

2.2.1. Contour Pixel Rate

Consider some image $x \in \mathbb{R}^{C \times H \times W}$ with a "ground truth" binary segmentation mask for some object within it, $m \in \{0, 1\}^{H \times W}$ (in general, this could also be just one class of some multi-class segmentation mask). We first propose to measure the degree of tree-like structure of the object m

according to the percentage of the object's pixels which lie on it's *contour*, which are defined as follows.

Definition 1 (Contour Pixels). *Given some mask m, a pixel m_{ij} is a contour pixel if $m_{ij} = 1$ and there exists a different pixel m_{kl} such that $m_{kl} \neq m_{ij}$ and $||(k, l) - (i, j)||_1 < R$, given some small contour width threshold $R > 0$.*

In other words, contour pixels have at least one pixel of a different class within a small neighborhood. Taking F to be the set of foreground pixels $F := \{(i, j) : m_{ij} = 1\}$, the set of contour pixels of m is then $C = \{(i, j) \in F : \exists (k, l) \in F^c$ such that $||(k, l) - (i, j)||_1 < R\}$, which we use to define the *Contour Pixel Rate* (CPR) of the object as

$$\text{CPR}(m) := \frac{|C|}{|F|}. \quad (1)$$

Intuitively, tree-like objects will have a higher percentage of contour pixels, resulting in higher CPRs. We demonstrate fast computation of CPR in full detail in Algorithm 1, via vectorized PyTorch-like pseudocode.

Algorithm 1 Contour Pixel Rate (CPR) of an object (PyTorch-like pseudocode).

Input: Object `mask` ($H \times W$ `tensor`), contour width threshold **R** (`int`).

```python
from skimage.morphology import diamond

def CPR(mask, R):
    neighb_kernel = diamond(R)
    neighb_counts = conv2d(mask,
        neighb_kernel, padding=R)
    contour_pix = logical_and(mask>0,
        neighb_counts <= (neighb_kernel.sum()-1))
    cpr = contour_pix.sum()/mask.sum()
    return cpr.item()
```

2.2.2. Difference of Gini-impurity Deviation (DoGD)

We alternatively propose to measure the tree-likeness of some object according to how the variability of object presence across different locations in the image differs between global and local scales. Intuitively, irregularly-spaced tree-like structures have high variability at small scales due to alternating frequently between areas of mixed and uniform pixel classes, but low variability at large scales due to the repetitive nature of the structure becoming more homogeneous, which we propose to quantify as follows.

First, we quantify the object presence within some $k \times k$ square window of the mask anchored at some coordinates $h_0 < H, w_0 < W$ via the *Gini impurity*:

$$\text{Gini}(m; k, h_0, w_0) := 1 - \sum_j [p_j(\mathcal{W}_{h_0, w_0}^k(m))]^2 \quad (2)$$

where we denote the $k \times k$ square window of the mask anchored at h_0, w_0 as $\mathcal{W}_{h_0, w_0}^k(m) := m[h_0 : h_0 + k, w_0 :$

$w_0 + k]$. Here, $p_j(\mathcal{W}^k_{h_0,w_0}(m))$ denotes the probability of the object of class j being in some pixel within the window, which is computed simply as n_j/k^2, where n_j is the number of pixels in the window of class j. In our binary segmentation case, the Gini impurity simplifies to

$$\mathrm{Gini}(m; k, h_0, w_0) := \qquad\qquad (3)$$
$$1 - [p(\mathcal{W}^k_{h_0,w_0}(m))]^2 - [1 - p(\mathcal{W}^k_{h_0,w_0}(m))]^2,$$

where we write $p := p_1$. The Gini impurity measures the degree of uncertainty (ranging from 0 to 1) of whether an object is present in the given window. For example, mask windows containing pixels of mostly one class will have $\mathrm{Gini} \simeq 1$, while having similar pixel amounts of both classes will result in $\mathrm{Gini} \simeq 0$[2].

Next, we compute the variability of object presence at a given scale/window size across the entire mask by sampling all possible $k \times k$ windows with anchors (h_0, w_0), and computing the standard deviation of the Gini impurity across these windows, as

$$\sigma^{\mathrm{Gini}}_k(m) := \sqrt{\mathrm{Var}_{h_0,w_0}[\mathrm{Gini}(m; k, h_0, w_0)]}. \quad (4)$$

Finally, we define the Difference of Gini Impurity Deviation (**DoGD**) between global and local scales as

$$\mathrm{DoGD}(m) := \sigma^{\mathrm{Gini}}_a(m) - \sigma^{\mathrm{Gini}}_b(m), \qquad (5)$$

where the global and local window sizes $k = a$ and $k = b$ are chosen such that $a \gg b$. We present DoGD in optimized PyTorch-like form in Algorithm 2.

Algorithm 2 Difference of Gini Impurity Deviation (DoGD) of an object (PyTorch-like pseudocode).

Input: Object **mask** ($H \times W$ tensor), global and local window sizes **a, b** (ints).

```
def DoGD(mask, a, b):
    gini_std = {}
    for k in [a,b]:
        avg_kernel = ones(k,k)
        p = conv2d(mask, avg_kernel)
        gini = 1 - p**2. - (1-p)**2.
        gini_std[k] = gini.std().item()
    return gini_std[a] - gini_std[b]
```

Intuitively, objects with significant tree-like or fractal-like structure will exhibit relatively large values of $\sigma^{\mathrm{Gini}}_b(m)$ due to high variability in pixel composition at small scales (frequently alternating between areas of mixed classes and areas of a single class), yet small $\sigma^{\mathrm{Gini}}_a(m)$ due

to structures with high uniformity and/or repetitions at large scales, altogether increasing the DoGD.

We perform all experiments with the hyperparameters for CPR and DoGD set to $R = 5$, $a = 127$, and $b = 3$, which we found via grid search for the values which resulted in the Kendall's τ between IoU and DoGD with the lowest p-value on the held-out DIS training set using ViT-H SAM. We show results using a wide range of other values for these hyperparameters in Appendix B.1, where we found our findings to be consistent for most other settings. Moreover, we note that CPR and DoGD are correlated with each other (Pearson $|r| = 0.83$ on average; Appendix D.1), showing their consistency in how they quantify different aspects of tree-likeness.

2.3. The Relationship between Object Tree-likeness and Segmentation Performance

2.3.1. Experiments on Synthetic Data

In this section, we will first carefully probe the effect of object tree-likeness on SAM's segmentation performance by testing it on synthetic images which solely possess objects of varying tree-likeness, with different independently chosen, uniform foreground and background textures. These objects are contiguous components samples from retinal blood vessel and satellite road masks, with the full procedure of generating these images detailed in Appendix A.2. Example generated images, masks and SAM segmentation predictions for them are shown in Fig. 3. As shown (as well in Fig. 4), the objects cover a wide range of tree-likeness as measured by these quantities.

In order to mitigate any confounding on SAM performance due to an object's textural contrast from its surroundings (which we study in Sec. 3), for each mask m_c generated by our procedure, we apply SAM to $K = 7$ images created by applying K different randomly-sampled pairs of textures to the object's foreground ($m_c = 1$) and background ($m_c = 0$). We then obtain SAM's final prediction for this object via pixel-wise majority voting over its predictions on these K images.

In Fig. 4 we show the relationship between an object's tree-likeness (as measured by CPR or DoGD) with SAM's performance (IoU) in segmenting the object, for all generated synthetic objects. We quantify the strength of this relation via rank/non-linear correlation, measured by Kendall's tau (τ) [25] and Spearman's rho (ρ) [37], shown in Table 1[3]. Intriguingly, we see that object tree-likeness is quite predictive of SAM's performance, with average absolute correlations of $|\tau| = 0.76$ and $|\rho| = 0.93$ for CPR and $|\tau| = 0.64$ and $|\rho| = 0.82$ for DoGD; *i.e.*, more prevalent dense tree-like structures corresponding to worse performance.

While this relationship is certainly strong, it is still on synthetic, controlled data. In the following section, we will

---

[2]The Gini impurity is closely related to the *entropy* $1 - \sum_j [p_j(\mathcal{W}^k_{h_0,w_0}(m)) \log p_j(\mathcal{W}^k_{h_0,w_0}(m))]$ (multiplied by $1/2$), which we use instead of entropy due to it being symmetric and faster to compute, following practices in decision tree learning [4].

[3]We evaluate τ in addition to ρ due to it being more robust to outliers.

4

Figure 3. **Left:** Example synthetic tree-like images and object masks. **Right:** Trend of increasing tree-likeness (increasing CPR/decreasing DoGD) of these objects resulting in worse SAM segmentation predictions (to the right of each object mask).

	CPR		DoGD	
SAM Encoder	τ	ρ	τ	ρ
ViT-H	-0.77	-0.93	0.61	0.80
ViT-B	-0.75	-0.93	0.66	0.84

Table 1. Rank correlation between SAM segmentation IoU and object tree-likeness (CPR and DoGD), on the synthetic dataset.

Figure 4. SAM segmentation IoU vs. object tree-likeness (CPR, left; DoGD, right) on the synthetic dataset.

show that this finding is also present for real data which is subject to noise from a variety of uncontrollable factors.

2.3.2. Experiments on Real Data

We will now perform the same analysis on two real datasets which contain objects which cover a spectrum of tree-likeness, DIS5k and iShape. DIS5k [33] (or "DIS" for short) is a dataset that contains extremely detailed segmentation masks of objects with varying degrees of hollowness, and both regular and irregular tree-like structures. All DIS experiments will be reported on its validation set unless otherwise stated. We show example images with objects with varying degrees of tree-likeness (by CPR) in Fig. 1 left.

iShape [43] consists of six sub-datasets of real and realistic-appearing synthetic images for instance segmentation of different objects: antenna, branches, fences, logs, hangers, and wires (see *e.g.* Fig. 9). We analyze these classes individually to mitigate potential confounding/noise factors due to inter-class variations; we do not do this for

DIS due to the larger number of classes which are much more fine-grained in their differences, additionally because there are few images per class for DIS.

Results. In Fig. 5 we show how SAM's performance (IoU) on these images relates to the tree-likeness of the objects which it is segmenting, with accompanying quantitative correlation results shown in Table 2. In order to reduce the noise incurred by the large variety of segmented objects in the dataset and nuisance confounding factors in the images, we analyze results with *aggregated objects*: we cluster groups of 5 objects/images with similar IoU and tree-likeness (CPR or DoGD) into single datapoints of the average value of these metrics, and similar for the IoU vs. textural separability experiments of Sec. 3.

Figure 5. SAM segmentation IoU vs. object tree-likeness (CPR, **top**; DoGD, **bottom**) on DIS (**left**) and iShape (**right**).

We see that despite the many potential confounding factors in real data, there is still a clear correlation between object tree-likeness as measured by the proposed metrics and SAM segmentation performance. In particular, we see an average absolute correlations of $|\tau| = 0.56$ and $|\rho| = 0.74$ for CPR and $|\tau| = 0.44$ and $|\rho| = 0.62$ for DoGD, excluding the antenna and wire objects of iShape which had outlying correlations, likely because those two object classes cover only small ranges of tree-likeness as opposed to the other iShape classes (Fig. 5), such that noise from other confounding factors obscures any dependence of performance on tree-likeness.

3. The Challenge of Textural Separability

In the previous section, we demonstrated that SAM struggles with segmenting non-conventional shapes, in particular, dense tree-like structures, which we hypothesize is due

		DIS		antenna		branch		fence		hanger		log		wire	
	SAM Enc.	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ
CPR:	ViT-H	-0.59	-0.76	-0.13	-0.21	-0.61	-0.81	-0.61	-0.81	-0.25	-0.37	-0.60	-0.77	-0.06	-0.10
	ViT-B	-0.63	-0.81	-0.18	-0.27	-0.63	-0.82	-0.63	-0.83	-0.44	-0.62	-0.62	-0.80	-0.12	-0.17
DoGD:	ViT-H	0.58	0.77	0.23	0.30	0.52	0.70	0.46	0.65	0.52	0.72	0.30	0.46	0.08	0.11
	ViT-B	0.45	0.64	0.12	0.14	0.51	0.69	0.45	0.64	0.50	0.70	0.13	0.22	0.06	0.08

Table 2. Rank correlation between SAM prediction IoU and object tree-likeness (CPR, **upper table**) and DoGD, **lower table**), on DIS and iShape.

to the model confusing the dense structure as the *texture* of a non-treelike, more regular shape, rather than a shape itself (see *e.g.* Fig. 1 left). Similar to this behavior is that even for objects with simpler shapes, SAM can still be confused if the object's texture is even somewhat similar to its surroundings, which we will study in this section.

3.1. Measuring Textural Separability

We will define the textural contrast or *separability* between some object mask and its surroundings by *how easily their textures can be distinguished from one another*. Motivated by findings that early layers of classification-pretrained CNNs primarily capture low-level features involving edges and textures [44], we will characterize an image's textures using the first convolutional layer of a ResNet-18 [19] pretrained on ImageNet [9]. Denoted by $f_1 : \mathbb{R}^{C \times H \times W} \to \mathbb{R}^{C' \times H' \times W'}$, this outputs a textural feature map corresponding to all 7×7 windows in the image.

We then measure the textural separability of an object according to if a simple classifier g can be trained to discriminate between (a) the activations for the pixels of the object foreground and (b) the activations right outside of the object boundary[4]. We define this explicitly in Algorithm 3, where Dilate and Disk refer to `scikit-image.morphology` functions [41]. We use simple logistic regression for g with inverse regularization parameter $C = 2$ [31]. We also evaluate using other hyperparameter settings, as well as a random forest classifier for g instead, in Appendix B.2, where we found similar results.

3.2. The Effect of Textural Separability on Segmentation Performance

3.2.1. Style Transfer for Controlled Textural Contrast

Similar to Sec. 2.3, in studying the effects of textural separability on SAM's performance, we also wish to perform experiments on well-controlled synthetic images in order to more carefully disentangle the effects of textural separability shape on performance from the effects due to object

---

Algorithm 3 Textural separability of an object.

Require: Image $x \in \mathbb{R}^{C \times H \times W}$ with mask $m \in \{0, 1\}^{H \times W}$, first convolutional layer of pretrained CNN f_1, simple classifier g.
1: $h = f_1(x) \in \mathbb{R}^{C' \times H' \times W'}$ *(Get textural features.)*
2: $m = \text{Resize}(m)$ to $H' \times W'$ *via NN interpolation.*
3: $m' := \text{Dilate}(m, \text{Disk}(5))$ *(Expand boundary.)*
4: $m' = m' - m$ *(Extract boundary only.)*
5: $h_{\text{obj}} := h[:, m \neq 0]$
6: $h_{\text{bdry}} := h[:, m' \neq 0]$
7: *Train g to classify between activations in h_{obj} vs. h_{bdry}.*
8: **return** *Training classification accuracy of g.*

---

shape, before evaluating on real data (in the following section). Here, we will do so using neural style transfer [15] to precisely modify the textural contrast of real objects/images. This will involve (1) modifying objects' shape without altering their background using an inpainting model, followed by (2) applying neural style transfer (NST) to the composite image to adjust the textural contrast of the object with the surrounding image.

We begin with images sampled from the VOC2012 dataset [13] accompanied by objects (instance masks), according to certain criteria. Namely, we pick objects which take up between 5% and 25% of the entire image's area (to have large enough objects while still maintaining sufficient background), sampling 484 objects total (with accompanying background) for study. For each object, we use a Stable Diffusion-based inpainting model (details in Appendix C.2) to remove the object from its corresponding image and fill in it's mask area with the background. This creates a pair of a background image and a separate object (mask), allowing us to have careful control of modifying the object (and its mask accordingly) before placing it back into the inpainted background. For a given object, we then create three types of composite images with varying degrees of changes to the object's shape (without modifying the background):

1. **Controlled:** the object is not modified.
2. **Altered:** the object undergoes major non-affine geometric transformations (see App. C.2), resulting in shape modification and its texture being squeezed or expanded.

---

[4]This procedure is somewhat similar to the probing of hidden activation concepts [2, 26], although these works detected concepts (such as textures) at the image level, not at the single activation level as we do here.

3. **Mixed:** this variant uses pixels from the altered image where the original and altered masks overlap, and pixels from the original object where they do not overlap.

After one of these three composite image types is created, we use neural style transfer (NST) to apply a texture to the image (similar to experiments in [16]) randomly sampled from Colored Brodatz [1], via an implementation of the NST model of Gatys et al. [15] (details in Appendix C.2). We illustrate an example of this entire procedure in Fig. 6.

Figure 6. Example images from each component of the synthetic textural separability dataset creation pipeline.

We adjust textural separability by performing the style transfer at eight different degrees of severity using different settings for the weighting hyperparameters that control the balance between content preservation and style transfer (details in Appendix C.2), with example styled images shown in Fig. 7 (more in Appendix. A.3). In Fig. 8 left, we validate this scheme by showing that indeed, steadily increasing the NST intensity results in decreasing textural separability.

Figure 7. Using style transfer with different intensities to adjust the textural separability of an object (highlighted in red), on *controlled* (upper row) and *altered* (lower row) versions of the object.

In Fig. 8 right, we show how SAM's segmentation performance changes with respect to style transfer intensity on the test set, for each of the three types of object transformations. We first see that SAM's performance decreases with

Figure 8. **Left:** Increasing NST intensity results in decreased textural separability. **Right:** SAM segmentation IoU vs. NST intensity, on the three composite image types.

lower textural separability/higher NST intensity. Second, there is an interesting pattern of clear gaps in segmentation performance between the three object transformation types. SAM performs best on the **altered** objects/foregrounds, which have both distorted shape and texture, compared to the un-altered/**controlled** objects, with the **mixed** objects in-between. This further demonstrates the effect of textural contrast on segmentation performance, as the altered objects have additional textural contrast to their backgrounds due to the transformations which were applied to their shape, boundary and texture.

3.2.2. Experiments on Real Data

We will now evaluate datasets of real images; iShape (Sec. 2.3.2) and Plittersdorf [18], both of which were used in the original SAM paper [27], and possess objects with a wide range of textural separability from their surroundings. Plittersdorf consists of camera trap video frames of wild deer recorded in a wildlife park. These frames often have low contrast objects due to frequent low-light conditions, making it a useful dataset for this analysis. Example images and objects from both datasets are shown in Fig. 9.

Figure 9. Example images and object masks from Plittersdorf (left group) and iShape (right group).

In Fig. 10, we show how the textural separability (Algorithm 3) of these objects relates to SAM's segmentation performance on them, with correlation results shown in Table

7

3. Overall, across all datasets we see a fairly strong correlation between textural separability and segmentation performance (average correlation of $\tau = 0.49$ and $\rho = 0.66$), especially considering the variety of objects and backgrounds, with objects with low separability resulting in especially poor segmentation (IoU ~ 0.3). In these cases, SAM was confused by objects close to the object of interest which *also* have similar textures (see *e.g.*, Fig. 10, right).

Figure 10. SAM prediction IoU vs. object textural separability (Algorithm 3), for iShape (left) and Plittersdorf (right).

Discussion

Understanding and Interpreting SAM's Failure Modes. First, we see that SAM's performance issues on cases of high tree-likeness (Sec. 2.3) were primarily due to over-segmentation, rather than under-segmentation; the model would frequently generate large false-positive regions in the empty space of densely packed, thin tree-like structures (*e.g.*, Figs. 3, right and 1, left). This could not simply be due to image resolution/downsampling blurring the thin objects, as for structures which are also thin but more sparsely packed, the over-segmentation is reduced (see *e.g.*, Fig. 3, center rightmost). We therefore hypothesize that the root of SAM's failure on these objects may lie in the highly repetitive yet irregular, *dense* patterns inherent to tree-like structures in particular: SAM confuses these patterns of shape as the texture of a more regularly-shaped object.

In the cases of both tree-like and low-textural contrast objects, SAM either has trouble with correctly delineating an object's shape when faced with a "confusing" textural cue, be it either a dense, irregular shape which appears to instead be a texture, or an object texture that is similar to its surroundings. One explanation for this is that SAM's training set SA-1B [27] possessed few objects with these qualities. This seems likely for tree-like objects, given the typically low concavity of objects in SA-1B (Fig. 6, right in [27]), which were captured in photographic contexts where such objects are uncommon. Low-textural contrast objects are similarly uncommon in most photographs, and furthermore, segmenting objects is generally more challenging if they are harder to pick out from their surroundings.

Applications of Our Findings and Metrics. A simple application of our findings and proposed metrics for the tree-likeness and textural separability of objects would be to predict if SAM is expected to perform underwhelmingly on new data, according to if the objects' measured tree-likeness is high or the textural separability is low. Similarly, the general diversity of objects in a segmentation foundation model's training/fine-tuning dataset could be measured according to the dataset's distribution of these metrics, which could inform whether additional training data is needed to be acquired/annotated to develop a better generalist model.

Limitations and Future Work. For our findings of the correlations between the object characteristics measured by our metrics and SAM's performance in segmenting such objects, we attempted to minimize the influence of other confounding factors which could effect segmentation performance by first establishing a baseline with our carefully-controlled synthetic data experiments. We then chose real evaluation datasets which covered a wide range of object types according to these metrics in order to gain a better "signal-to-noise ratio" for the studied trends. However, it is impossible to mitigate all potential confounding factors which could affect performance in real data, which is why our correlation findings are still noticeable for the real datasets, yet not quite as tight as on the synthetic data.

We do not evaluate the 2D version of the recently released SAM 2 model due to it likely resulting in similar behavior as SAM 1, along with evidence that oracle predictions of SAM and SAM 2 are typically similar in performance to within $\lesssim 0.1$ IoU [12].

Another future direction would be to explore the specific component(s) of SAM which can be most attributed to these failures (in a mechanistic sense), which could guide further model development. We deem it likely that the susceptible component of SAM is the image encoder, simply because it is the main backbone of the model which extracts features to be used by the lightweight mask decoder, but obtaining a more fine-grained answer would require a careful treatment.

Conclusion

In this paper, we quantitatively modeled how the segmentation performance of SAM relates to certain measurable object characteristics: tree-likeness and textural separability. We find SAM's performance to typically be noticeably correlated with these factors, showing the need for further work in understanding and potentially mitigating such "failure modes" of vision foundation models.

Acknowledgements

	iShape												Plittersdorf	
	antenna		branch		fence		hanger		log		wire			
SAM Enc.	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ
ViT-H	0.44	0.59	0.50	0.68	0.65	0.85	0.63	0.83	0.33	0.48	0.49	0.67	0.26	0.38
ViT-B	0.46	0.61	0.56	0.75	0.67	0.86	0.65	0.84	0.36	0.52	0.55	0.73	0.36	0.50

Table 3. Rank correlation between SAM prediction IoU and object textural separability, on iShape and Plittersdorf.

References

[1] Safia Abdelmounaime and He Dong-Chen. New brodatz-based image databases for grayscale color and multiband texture analysis. *International Scholarly Research Notices*, 2013(1):876386, 2013. 7, 1

[2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016. 6

[3] Nicholas Baker, Hongjing Lu, Gennady Erlikhman, and Philip J Kellman. Deep convolutional networks do not classify based on global object shape. *PLoS computational biology*, 14(12):e1006613, 2018. 2

[4] Leo Breiman. *Classification and regression trees*. Routledge, 1984. 4

[5] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. 3

[6] Keyan Chen, Chenyang Liu, Hao Chen, Haotian Zhang, Wenyuan Li, Zhengxia Zou, and Zhenwei Shi. Rsprompter: Learning to prompt for remote sensing instance segmentation based on visual foundation model. *IEEE Transactions on Geoscience and Remote Sensing*, 2024. 1

[7] Tianrun Chen, Lanyun Zhu, Chaotao Deng, Runlong Cao, Yan Wang, Shangzhan Zhang, Zejian Li, Lingyun Sun, Ying Zang, and Papa Mao. Sam-adapter: Adapting segment anything in underperformed scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 3367–3375, 2023. 1

[8] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. Deepglobe 2018: A challenge to parse the earth through satellite images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018. 1

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[10] Ruining Deng, Can Cui, Quan Liu, Tianyuan Yao, Lucas Walker Remedios, Shunxing Bao, Bennett A Landman, Yucheng Tang, Lee E Wheless, Lori A Coburn, et al. Segment anything model (sam) for digital pathology: Assess zero-shot segmentation on whole slide imaging. In *Medical Imaging with Deep Learning, short paper track*, 2023. 1

[11] Guanliang Dong, Zhangquan Wang, Yourong Chen, Yuliang Sun, Hongbo Song, Liyuan Liu, and Haidong Cui. An efficient segment anything model for the segmentation of medical images. *Scientific Reports*, 14(1):19425, 2024. 1

[12] Haoyu Dong, Hanxue Gu, Yaqian Chen, Jichen Yang, and Maciej A Mazurowski. Segment anything model 2: an application to 2d and 3d medical images. *arXiv preprint arXiv:2408.00756*, 2024. 8

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://host.robots.ox.ac.uk/pascal/VOC/voc2012/, 2012. 6

[14] Wenqing Feng, Fangli Guan, Chenhao Sun, and Wei Xu. Road-sam: Adapting the segment anything model to road extraction from large very-high-resolution optical remote sensing images. *IEEE Geoscience and Remote Sensing Letters*, 21:1–5, 2024. 3

[15] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 6, 7, 2

[16] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. 2, 7

[17] Hanxue Gu, Haoyu Dong, Jichen Yang, and Maciej A Mazurowski. How to build the best medical image segmentation algorithm using foundation models: a comprehensive empirical study with segment anything model. *arXiv preprint arXiv:2404.09957*, 2024. 1

[18] Timm Haucke, Hjalmar S. Kühl, and Volker Steinhage. Socrates: Introducing depth in visual wildlife monitoring using stereo vision. *Sensors*, 22(23), 2022. 7

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[20] Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. *Advances in Neural Information Processing Systems*, 33:9995–10006, 2020. 2

[21] Katherine Hermann, Ting Chen, and Simon Kornblith. The origins and prevalence of texture bias in convolutional neural networks. *Advances in Neural Information Processing Systems*, 33:19000–19015, 2020. 2

[22] Yuhao Huang, Xin Yang, Lian Liu, Han Zhou, Ao Chang, Xinrui Zhou, Rusi Chen, Junxuan Yu, Jiongquan Chen, Chaoyu Chen, et al. Segment anything model for medical images? *Medical Image Analysis*, 92:103061, 2024. 1

[23] Abdallah Wagih Ibrahim. Retina Blood Vessel, 2023. 1

[24] Wei Ji, Jingjing Li, Qi Bi, Tingwei Liu, Wenbo Li, and Li Cheng. Segment Anything Is Not Always Perfect: An Investigation of SAM on Different Real-world Applications. *Machine Intelligence Research*, 21(4):617–630, 2024. 1

[25] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938. 4

[26] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018. 6

[27] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 1, 2, 7, 8

[28] Jonas Kubilius, Stefania Bracci, and Hans P Op de Beeck. Deep neural networks as a computational model for human shape sensitivity. *PLoS computational biology*, 12(4): e1004896, 2016. 2

[29] Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, 2024. 1, 3

[30] Maciej A. Mazurowski, Haoyu Dong, Hanxue Gu, Jichen Yang, Nicholas Konz, and Yixin Zhang. Segment anything model for medical image analysis: An experimental study. *Medical Image Analysis*, 89:102918, 2023. 1

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 6

[32] Zelin Peng, Zhengqin Xu, Zhilin Zeng, Xiaokang Yang, and Wei Shen. Sam-parser: Fine-tuning sam efficiently by parameter space reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4515–4523, 2024. 1

[33] Xuebin Qin, Hang Dai, Xiaobin Hu, Deng-Ping Fan, Ling Shao, and Luc Van Gool. Highly accurate dichotomous image segmentation. In *European Conference on Computer Vision*, pages 38–56. Springer, 2022. 5

[34] Zhongxi Qiu, Yan Hu, Heng Li, and Jiang Liu. Learnable ophthalmology sam. *arXiv preprint arXiv:2304.13425*, 2023. 3

[35] Simiao Ren, Francesco Luzi, Saad Lahrichi, Kaleb Kassaw, Leslie M Collins, Kyle Bradbury, and Jordan M Malof. Segment anything, from space? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 8355–8365, 2024. 1

[36] Peilun Shi, Jianing Qiu, Sai Mu Dalike Abaxi, Hao Wei, Frank P-W Lo, and Wu Yuan. Generalist vision foundation models for medical imaging: A case study of segment anything model on zero-shot medical segmentation. *Diagnostics*, 13(11):1947, 2023. 1

[37] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. Place: US Publisher: Univ of Illinois Press. 4

[38] Ajay Subramanian, Elena Sizikova, Najib Majaj, and Denis Pelli. Spatial-frequency channels, shape bias, and adversarial robustness. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[39] Lv Tang, Haoke Xiao, and Bo Li. Can sam segment anything? when sam meets camouflaged object detection. *arXiv preprint arXiv:2304.04709*, 2023. 1

[40] Shikhar Tuli, Ishita Dasgupta, Erin Grant, and Thomas L Griffiths. Are convolutional neural networks or transformers more like human vision? *arXiv preprint arXiv:2105.07197*, 2021. 2

[41] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 2014. 6

[42] Nan Xu, Kerry Nice, Sachith Seneviratne, and Mark Stevenson. Leveraging segment-anything model for automated zero-shot road width extraction from aerial imagery. In *2023 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 176–183. IEEE, 2023. 3

[43] Lei Yang, Yan Zi Wei, Yisheng He, Wei Sun, Zhenhang Huang, Haibin Huang, and Haoqiang Fan. ishape: A first step towards irregular shape instance segmentation. *arXiv preprint arXiv:2109.15068*, 2021. 5

[44] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision– ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014. 6

[45] Chenshuang Zhang, Chaoning Zhang, Taegoo Kang, Donghun Kim, Sung-Ho Bae, and In So Kweon. Attack-sam: Towards attacking segment anything model with adversarial examples. *arXiv preprint arXiv:2305.00866*, 2023. 1

[46] Yixin Zhang and Maciej A Mazurowski. Convolutional neural networks rarely learn shape for semantic segmentation. *Pattern Recognition*, 146:110018, 2024. 2

[47] Sheng Zheng, Chaoning Zhang, and Xinhong Hao. Black-box targeted adversarial attack on segment anything (sam). *arXiv preprint arXiv:2310.10010*, 2023. 1

Quantifying the Limits of Segment Anything Model: Analyzing Challenges in Segmenting Tree-Like and Low-Contrast Structures

Supplementary Material

A. Additional Dataset Details

A.1. Retinal Blood Vessel and Road Satellite Images

The retinal blood vessel and road satellite image object masks which we use to generate the synthetic images for the experiments in Sec. 2.3.1, are from the Retina Blood Vessel [23] and the Road Extraction Challenge data of Deep-Globe18 [8]. For our retinal vessel mask set, we use the vessel masks from the training set of 80 image/mask pairs, and for our road mask set, we randomly sample the same number of masks from the split-merged full dataset, in order to ensure that these two object types appear equally in the synthetic dataset of Sec. 2.3.1.

A.2. Synthetic Tree-like Object Images

Our algorithm for creating synthetic images of tree-like objects (used in Sec. 2.3.1) is shown as follows.

1. Sample object mask m from either retinal blood vessel or satellite road image datasets (details in Appendix A.1).
2. Randomly select one of the contiguous components of m, denoted m_c, and enclose it with a tight bounding box.
3. Resize m_c such that its bounding box is 512×512, and randomly place it in a blank 1024×1024 image.
4. Apply two different textures randomly sampled from Colored Brodatz [1] to the foreground ($m_c = 1$) and background ($m_c = 0$), resulting in the final image.

In Fig. 12 we show example images generated with different foreground and background texture combinations for various objects.

A.3. Neural Style Transfer (NST) Generated Images

In Fig. 11 we provide additional example images generated via inpainting and neural style transfer for the textural separability experiments of Sec. 3.2.1.

B. Hyperparameter/Ablation Studies

B.1. Object Tree-likeness Experiments

B.1.1. Synthetic Data Experiments

Tables 4 and 5 show results on the synthetic tree-like dataset using a wide range of CPR and DoGD hyperparameters (R and a, b, respectively).

B.1.2. Real Data Experiments

Tables 6, 7, 8, and 9 show results on DIS and iShape using a wide range of CPR and DoGD hyperparameters (R and a, b, respectively).

	ViT-H		ViT-B	
R	τ	ρ	τ	ρ
1	-0.82	-0.96	-0.79	-0.94
3	-0.80	-0.95	-0.76	-0.94
5	-0.77	-0.93	-0.75	-0.93
7	-0.74	-0.92	-0.73	-0.91
9	-0.72	-0.91	-0.70	-0.90
11	-0.71	-0.90	-0.70	-0.89

Table 4. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **CPR** with different hyperparameter values R on the **synthetic dataset**, to supplement Table 1.

		ViT-H		ViT-B	
a	b	τ	ρ	τ	ρ
63	3	0.42	0.60	0.48	0.66
63	7	0.45	0.63	0.49	0.68
63	15	0.44	0.64	0.48	0.68
63	31	0.45	0.66	0.49	0.71
127	3	0.61	0.80	0.66	0.84
127	7	0.63	0.81	0.67	0.84
127	15	0.62	0.81	0.66	0.84
127	31	0.60	0.80	0.64	0.83
255	3	0.66	0.84	0.68	0.85
255	7	0.66	0.84	0.68	0.86
255	15	0.65	0.84	0.67	0.85
255	31	0.59	0.79	0.60	0.80

Table 5. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **DoGD** with different hyperparameter values a, b on the **synthetic dataset**, to supplement Table 1.

B.2. Object Textural Separability Experiments

Table 10 show the textural separability experiment results on real data (iShape and Plittersdorf) using various hyperparameters and models for the classifier g.

C. Additional Experimental Details

C.1. SAM Prompting Strategies

In addition to the SAM prompting details presented in Sec. 1, all real and style-transferred (NST) images are prompted with (1) a tight bounding-box and (2) several positive and/or

Figure 11. Additional examples of inpainting+neural style transfer-generated images to supplement Fig. 6.

negative prompts randomly sampled from either the foreground or background of the object mask, respectively, with respective counts n_{pos} and n_{neg}. For iShape, we use $n_{pos} = n_{neg} = 5$; for DIS, we use $n_{pos} = 5$ and $n_{neg} = 10$ due to the complexity of the objects. For Plittersdorf and NST images, we simply use $n_{pos} = 0$ and $n_{neg} = 2$, due to the objects typically possessing simple shapes.

C.2. Neural Style Transfer Experiments

Models. For the NST experiments (Sec. 3.2.1), the inpainting model used is Runway's stable diffusion in-

painting pipeline, fp16 variant. The NST model it-self is an implementation of [15] based on `https://github.com/pytorch/tutorials/blob/main/advanced_source/neural_style_tutorial.py`, using typical settings of a standard VGG19 CNN with content layer of conv_4, style layers of conv_i for i = 1, 2, 3, 4, 5, and ImageNet normalization.

Style transfer intensity. As mentioned in Sec. 3.2.1, we perform style transfer experiments for a monotonically-

Figure 12. Examples of synthetic tree-like object images. Object masks shown in the right-most column, with images created using randomly chosen different foreground and background texture combinations in the left columns (before the majority-voting procedure described in Sec. 2.3.1).

R	ViT-H		ViT-B	
	τ	ρ	τ	ρ
1	-0.56	-0.73	-0.61	-0.79
3	-0.55	-0.72	-0.61	-0.79
5	-0.59	-0.76	-0.63	-0.81
7	-0.53	-0.7	-0.59	-0.77
9	-0.53	-0.71	-0.57	-0.75
11	-0.51	-0.69	-0.56	-0.74

Table 6. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **CPR** with different hyperparameter values R on **DIS**, to supplement Table 2.

increasing range of eight degrees of style transfer intensity. This is created by defining content and style weights λ_c and λ_s, respectively for the NST algorithm according to

$$\lambda_c = \frac{1}{1+\alpha} \quad \text{and} \quad \lambda_s = \frac{\alpha}{1+\alpha}, \qquad (6)$$

where α ranges linearly from $\alpha = 1$ to $\alpha = 4,000$ with eight equally-spaced values, representing the degree of style transfer intensity.

Object/image altering procedure. The procedure that creates the "altered" version of objects via non-affine trans-

formations is detailed as follows.

1. Apply a non-affine transformation created using the `albumentations` Python library [5] to both the image and the object/mask, defined shortly.
2. Clean up unexpected obsolete regions.
3. Properly align the location of the distorted object.
4. Remove small isolated regions of the object.

In Python code, this is implemented as follows, given an input image NumPy array `raw_img_arr` and corresponding binary object mask `raw_msk_arr`:

```python
import torch
import numpy as np
from PIL import Image
import albumentations as A
import torchvision.transforms as transforms
from skimage.morphology import dilation,label, \
disk, remove_small_holes, remove_small_objects, \
opening, closing

shape_transformation = A.Compose([
    A.ElasticTransform(
        alpha=2500, sigma=24, alpha_affine=0,
        interpolation=1, border_mode=0, value=0,
        mask_value=0, always_apply=True,
        approximate=False, same_dxdy=False, p=1.0),
    A.GridDistortion(
        num_steps=20, distort_limit=.99,
        interpolation=1, border_mode=0, value=0,
        mask_value=0, normalized=False,
        always_apply=False, p=1.0)]
)

imsize = 512
loader = transforms.Compose([
    transforms.Resize(imsize),   # scale imported image
    transforms.CenterCrop(imsize),
    transforms.ToTensor()])

unloader = transforms.ToPILImage()

def shift_array(array, shift_spec):
    assert len(shift_spec) == 2 and \
        isinstance(array,np.ndarray)

    shift_corrected_arr = array.copy()
    if shift_spec[0] != 0:
        shift = shift_spec[0]
        shift_corrected_arr = np.concatenate(
            [shift_corrected_arr[shift:],
            shift_corrected_arr[:shift]],
            axis=0)

    if shift_spec[1] != 0:
        shift = shift_spec[1]
        shift_corrected_arr = np.concatenate(
            [shift_corrected_arr[:,shift:],
            shift_corrected_arr[:,:shift]],
            axis=1)

    return shift_corrected_arr

def center_correction(raw_msk, distorted_msk):
    assert raw_msk.ndim == 4 and \
        distorted_msk.ndim == 3
    pos_idx = torch.argwhere(raw_msk[0,0])
    _raw_msk_center = (pos_idx.min(dim=0).values + \
        pos_idx.max(dim=0).values)//2
    pos_idx = torch.argwhere(distorted_msk[0])
    _distorted_msk_center = (pos_idx.min(dim=0).values + \
        pos_idx.max(dim=0).values)//2

    position_correction = _distorted_msk_center -_raw_msk_center
    corrected_distorted_msk = distorted_msk.squeeze()
    print(corrected_distorted_msk.shape)
    assert position_correction.ndim == 1 and len(position_correction) == 2

    if position_correction[0] !=0:
        shift = position_correction[0]

        corrected_distorted_msk = torch.cat(
            [corrected_distorted_msk[shift:],
            corrected_distorted_msk[:shift]],
            dim=0)
    if position_correction[1] !=0:
        shift = position_correction[1]
        corrected_distorted_msk = torch.cat(
            [corrected_distorted_msk[:,shift:],
            corrected_distorted_msk[:,:shift]],
            dim=1)
    return corrected_distorted_msk

def get_distorted_ImageAndMask(trans, img, *msks):

    # format check
    assert all([msk.ndim == 2 for msk in msks])
```

3

R	SAM Enc.	antenna		branch		fence		hanger		log		wire	
		τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ
1	ViT-H	−0.18	−0.25	−0.63	−0.83	−0.68	−0.88	−0.30	−0.43	−0.60	−0.78	−0.08	−0.12
	ViT-B	−0.19	−0.29	−0.64	−0.83	−0.68	−0.88	−0.47	−0.66	−0.61	−0.79	−0.12	−0.18
3	ViT-H	−0.20	−0.30	−0.63	−0.83	−0.65	−0.85	−0.28	−0.40	−0.61	−0.79	−0.07	−0.10
	ViT-B	−0.17	−0.26	−0.64	−0.83	−0.65	−0.85	−0.48	−0.66	−0.61	−0.79	−0.13	−0.19
5	ViT-H	−0.13	−0.21	−0.61	−0.81	−0.61	−0.81	−0.25	−0.37	−0.60	−0.79	−0.06	−0.10
	ViT-B	−0.18	−0.27	−0.63	−0.82	−0.63	−0.83	−0.44	−0.62	−0.62	−0.80	−0.12	−0.17
7	ViT-H	−0.16	−0.23	−0.57	−0.77	−0.54	−0.74	−0.20	−0.29	−0.60	−0.79	−0.06	−0.10
	ViT-B	−0.15	−0.24	−0.59	−0.77	−0.55	−0.75	−0.39	−0.55	−0.62	−0.81	−0.10	−0.15
9	ViT-H	−0.16	−0.23	−0.52	−0.72	−0.47	−0.65	−0.13	−0.19	−0.59	−0.78	−0.05	−0.08
	ViT-B	−0.16	−0.22	−0.57	−0.76	−0.49	−0.67	−0.29	−0.42	−0.62	−0.81	−0.10	−0.15
11	ViT-H	−0.08	−0.13	−0.49	−0.67	−0.51	−0.70	−0.02	−0.03	−0.60	−0.78	−0.03	−0.04
	ViT-B	−0.12	−0.16	−0.54	−0.73	−0.55	−0.75	−0.21	−0.28	−0.64	−0.82	−0.07	−0.11

Table 7. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **CPR** with different hyperparameter values R on **iShape**, to supplement Table 2.

		ViT-H		ViT-B	
a	b	τ	ρ	τ	ρ
63	3	0.63	0.83	0.57	0.78
63	7	0.59	0.79	0.57	0.77
63	15	0.54	0.72	0.50	0.67
63	31	0.45	0.62	0.39	0.56
127	3	0.58	0.77	0.45	0.64
127	7	0.58	0.77	0.49	0.66
127	15	0.51	0.67	0.42	0.57
127	31	0.36	0.50	0.26	0.39
255	3	0.53	0.71	0.42	0.59
255	7	0.53	0.71	0.42	0.59
255	15	0.47	0.66	0.36	0.52
255	31	0.29	0.43	0.19	0.29

Table 8. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **DoGD** with different hyperparameter values a, b on **DIS**, to supplement Table 2.

```
#shift the arrays so that the obj location aligns
distorted_img = shift_array(distorted_img_shifted,shift_spec)
distorted_msk = shift_array(distorted_msk_shifted,shift_spec)


# remove relatively small regions
distorted_msk_labeled = label(distorted_msk)
unique_ids, id_cnts = np.unique(
    distorted_msk_labeled, return_counts = True)
area_distribution = id_cnts[1:].astype('float')/id_cnts[1:].sum()

for area_id, area_rate in zip(unique_ids[1:], area_distribution):
    if area_rate < 0.125:
        distorted_msk_labeled[distorted_msk_labeled==area_id] = 0

distorted_msk_arr = distorted_msk_labeled>0

return distorted_img, distorted_msk_arr

distorted_img_arr, distorted_msk_arr = get_distorted_ImageAndMask(
    shape_transformation,raw_img_arr,raw_msk_arr,raw_msk_arr.copy())
```

D. Additional Results

D.1. Correlation Between Metrics

In Fig. D.1, we show the relationship between our proposed tree-likeness metrics CPR and DoGD on all three datasets (with default hyperparameters and ViT-H SAM), quantified by correlations shown in Table 11.

```
# apply transformation to both image and msk
transformed = trans(image=img, masks = msks)
distorted_img_shifted = transformed['image']
distorted_msks_shifted = transformed['masks']

#clean up the unexpected obsolete regions
## get area excluding border
positive_region_map = label(distorted_msks_shifted[1])
label_ids, label_cnts = np.unique(positive_region_map,return_counts=True)
max_id = label_ids[1:][np.argmax(label_cnts[1:])]
eligible_region_map = np.where(positive_region_map==max_id, 1, 0)

distorted_msk_shifted = distorted_msks_shifted[0]*eligible_region_map
distorted_msk_shifted = remove_small_objects(remove_small_holes(
    closing(opening(distorted_msk_shifted>0, disk(2)),disk(2))
    ))

#find center shift as a results of transformation
pos_idx = np.argwhere(msks[0]).squeeze()
_raw_msk_center = (pos_idx.min(0)+pos_idx.max(0))//2
pos_idx = np.argwhere(distorted_msk_shifted).squeeze()
_distorted_msk_center = (pos_idx.min(0)+pos_idx.max(0))//2
shift_spec = _distorted_msk_center -_raw_msk_center
```

a	b	SAM Enc.	antenna τ	antenna ρ	branch τ	branch ρ	fence τ	fence ρ	hanger τ	hanger ρ	log τ	log ρ	wire τ	wire ρ
63	3	ViT-H	0.39	0.55	0.62	0.82	0.65	0.85	-0.15	-0.23	0.23	0.35	0.35	0.50
		ViT-B	0.25	0.35	0.62	0.81	0.60	0.81	-0.07	-0.11	0.17	0.25	0.36	0.51
63	7	ViT-H	0.28	0.38	0.61	0.81	0.62	0.83	-0.13	-0.21	0.22	0.33	0.34	0.49
		ViT-B	0.13	0.18	0.61	0.81	0.60	0.80	-0.07	-0.11	0.14	0.22	0.33	0.48
63	15	ViT-H	0.10	0.12	0.59	0.78	0.59	0.80	-0.05	-0.07	0.19	0.30	0.29	0.43
		ViT-B	0.04	0.03	0.59	0.77	0.56	0.77	-0.01	-0.01	0.09	0.16	0.25	0.36
63	31	ViT-H	0.06	0.04	0.55	0.74	0.46	0.65	0.07	0.10	0.14	0.26	0.09	0.13
		ViT-B	-0.11	-0.17	0.56	0.74	0.45	0.64	0.07	0.11	0.02	0.08	0.02	0.03
127	3	ViT-H	0.23	0.30	0.52	0.70	0.46	0.65	0.52	0.72	0.30	0.46	0.08	0.11
		ViT-B	0.12	0.14	0.51	0.69	0.45	0.64	0.50	0.70	0.13	0.22	0.06	0.08
127	7	ViT-H	0.23	0.26	0.48	0.65	0.41	0.59	0.51	0.70	0.29	0.46	-0.01	-0.01
		ViT-B	0.10	0.09	0.51	0.68	0.41	0.59	0.49	0.68	0.13	0.22	-0.03	-0.05
127	15	ViT-H	0.14	0.17	0.43	0.59	0.38	0.55	0.50	0.69	0.28	0.43	-0.09	-0.12
		ViT-B	0.07	0.03	0.45	0.62	0.36	0.52	0.48	0.67	0.12	0.20	-0.14	-0.20
127	31	ViT-H	0.05	0.05	0.36	0.50	0.08	0.12	0.48	0.67	0.30	0.45	-0.15	-0.22
		ViT-B	0.04	0.00	0.38	0.52	0.12	0.18	0.46	0.65	0.12	0.20	-0.21	-0.31
255	3	ViT-H	0.04	0.06	0.33	0.46	-0.20	-0.30	0.44	0.61	0.52	0.72	-0.17	-0.25
		ViT-B	0.21	0.31	0.30	0.42	-0.13	-0.20	0.42	0.59	0.42	0.60	-0.23	-0.34
255	7	ViT-H	0.02	0.02	0.29	0.41	-0.25	-0.35	0.43	0.60	0.51	0.71	-0.20	-0.29
		ViT-B	0.19	0.27	0.26	0.36	-0.18	-0.26	0.42	0.58	0.41	0.59	-0.25	-0.36
255	15	ViT-H	-0.04	-0.06	0.24	0.34	-0.31	-0.45	0.43	0.61	0.50	0.70	-0.20	-0.30
		ViT-B	0.21	0.29	0.20	0.28	-0.26	-0.39	0.40	0.56	0.40	0.58	-0.28	-0.40
255	31	ViT-H	-0.11	-0.14	0.11	0.16	-0.47	-0.65	0.43	0.59	0.46	0.66	-0.23	-0.33
		ViT-B	0.13	0.20	0.07	0.09	-0.42	-0.60	0.39	0.55	0.38	0.55	-0.30	-0.44

Table 9. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **DoGD** with different hyperparameter values R on **iShape**, to supplement Table 2.

Weak Classifier Model	SAM Enc.	iShape antenna τ	antenna ρ	branch τ	branch ρ	fence τ	fence ρ	hanger τ	hanger ρ	log τ	log ρ	wire τ	wire ρ	Plittersdorf τ	Plittersdorf ρ
Logistic, $C = 2$	ViT-H	0.44	0.59	0.50	0.68	0.65	0.85	0.63	0.83	0.33	0.48	0.49	0.67	0.26	0.38
	ViT-B	0.46	0.61	0.56	0.75	0.67	0.86	0.65	0.84	0.36	0.52	0.55	0.73	0.36	0.50
Logistic, $C = 1$	ViT-H	0.42	0.58	0.49	0.67	0.64	0.84	0.62	0.81	0.31	0.46	0.49	0.67	0.34	0.49
	ViT-B	0.44	0.60	0.55	0.75	0.66	0.85	0.63	0.82	0.35	0.50	0.55	0.72	0.39	0.55
Logistic, $C = 0.5$	ViT-H	0.39	0.53	0.49	0.67	0.65	0.85	0.60	0.80	0.30	0.43	0.48	0.65	0.33	0.47
	ViT-B	0.48	0.65	0.57	0.76	0.66	0.86	0.62	0.81	0.33	0.48	0.52	0.69	0.41	0.56
Random Forest, $n_{\text{estimators}} = 2$	ViT-H	0.48	0.66	0.45	0.63	0.46	0.64	0.45	0.64	0.19	0.27	0.25	0.36	0.35	0.50
	ViT-B	0.50	0.67	0.49	0.68	0.47	0.65	0.50	0.69	0.24	0.34	0.31	0.45	0.41	0.56
Random Forest, $n_{\text{estimators}} = 4$	ViT-H	0.51	0.69	0.42	0.60	0.54	0.73	0.48	0.67	0.21	0.30	0.30	0.43	0.22	0.31
	ViT-B	0.51	0.70	0.48	0.66	0.53	0.72	0.51	0.70	0.22	0.32	0.37	0.52	0.32	0.45
Random Forest, $n_{\text{estimators}} = 8$	ViT-H	0.48	0.65	0.46	0.63	0.56	0.75	0.50	0.69	0.19	0.28	0.35	0.50	0.22	0.31
	ViT-B	0.49	0.67	0.49	0.68	0.55	0.74	0.54	0.73	0.21	0.31	0.40	0.57	0.25	0.35
Random Forest, $n_{\text{estimators}} = 16$	ViT-H	0.45	0.62	0.48	0.66	0.57	0.77	0.52	0.72	0.20	0.29	0.38	0.53	0.08	0.12
	ViT-B	0.58	0.77	0.52	0.70	0.54	0.74	0.54	0.74	0.23	0.33	0.41	0.57	0.15	0.23

Table 10. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object textural separability (Algorithm 3), on real datasets (iShape and Plittersdorf), using different weak classifier models g and hyperparameters to measure textural separability (to supplement Table 3).

Figure 13. Relationship between tree-likeness metrics CPR and DoGD. From left to right, on synthetic data (Fig. 4), DIS, and iShape (Fig. 5).

	Synth.	**DIS**	iShape							**Avg.**
			antenna	branch	fence	hanger	log	wire	*all*	
Pearson r	-0.94	-0.67	-0.92	-0.93	-0.82	-0.81	-0.90	-0.66	-0.79	-0.83
Spearman ρ	-0.82	-0.70	-0.91	-0.93	-0.86	-0.76	-0.94	-0.79	-0.76	-0.83
Kendall τ	-0.62	-0.51	-0.75	-0.78	-0.68	-0.58	-0.79	-0.61	-0.57	-0.65

Table 11. Linear (Pearson r) and nonlinear (Spearman ρ, Kendall τ) correlations between CPR and DoGD on all evaluated datasets.