

Appendix

Table of Contents

A Background on diffusion models	13
B Muse as a classifier	14
C Limitations	14
D Image attribution	15
E Details on ResNet-50 training with diffusion noise	15
F Additional plots for model-vs-human benchmark	16
G Quantitative benchmark scores and rankings	16

A BACKGROUND ON DIFFUSION MODELS

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020; Song & Ermon, 2020) are latent variable generative models defined by a forward and reverse Markov chain. Given an unknown data distribution, $q(\mathbf{x}_0)$, over observations, $\mathbf{x}_0 \in \mathbb{R}^d$, the forward process corrupts the data into a sequence of noisy latent variables, $\mathbf{x}_{1:T} := \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$, by gradually adding Gaussian noise with a fixed schedule defined as:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad (3)$$

where $q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \text{Normal}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$. The reverse Markov process gradually denoises the latent variables to the data distribution with learned Gaussian transitions starting from $\text{Normal}(\mathbf{x}_T; 0, \mathbf{I})$ i.e.

$$p_{\theta}(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \cdot \prod_{t=0}^{T-1} p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) := \text{Normal}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\theta}(\mathbf{x}_t, t))$. The aim of training is for the forward process distribution $\{\mathbf{x}_t\}_{t=0}^T$ to match that of the reverse process $\{\tilde{\mathbf{x}}_t\}_{t=0}^T$ i.e., the generative model $p_{\theta}(\mathbf{x}_0)$ closely matches the data distribution $q(\mathbf{x}_0)$. Specifically, these models can be trained by optimizing the variational lower bound of the marginal likelihood (Ho et al., 2020; Kingma et al., 2021):

$$-\log p_{\theta}(\mathbf{x}_0) \leq -\text{VLB}(\mathbf{x}_0) := \mathcal{L}_{\text{Prior}} + \mathcal{L}_{\text{Recon}} + \mathcal{L}_{\text{Diffusion}}$$

$\mathcal{L}_{\text{Prior}}$ and $\mathcal{L}_{\text{Recon}}$ are the prior and reconstruction loss that can be estimated using standard techniques in the literature (Kingma & Welling, 2014). The (re-weighted) diffusion loss can be written as:

$$\mathcal{L}_{\text{Diffusion}} = \mathbb{E}_{\mathbf{x}_0, \varepsilon, t} \left[\|\mathbf{w}_t \|\mathbf{x}_0 - \tilde{\mathbf{x}}_{\theta}(\mathbf{x}_t, t)\|_2^2 \right]$$

with $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, $\varepsilon \sim \text{Normal}(0, \mathbf{I})$, and $t \sim \mathcal{U}([0, T])$. Here, \mathbf{w}_t is a weight assigned to the timestep, and $\tilde{\mathbf{x}}_{\theta}(\mathbf{x}_t, t)$ is the model’s prediction of the observation \mathbf{x}_0 from the noised observation \mathbf{x}_t . Diffusion models can be conditioned on additional inputs like class labels, text prompts, segmentation masks or low-resolution images, in which case $\tilde{\mathbf{x}}_{\theta}$ also takes a conditioning signal \mathbf{y} as input.

B MUSE AS A CLASSIFIER

In Figure 8, we visualize why we were not able to include Muse as a (successful) classifier in our experiments. Even on clean, undistorted images Muse achieved only approximately chance-level accuracy. This may, however, just be a limitation on how we attempted to extract classification decisions out of the model; it is very well possible that other approaches might work better.



Figure 8: **Why Muse is a poor classifier.** This figure illustrates why Muse reconstructions cannot be used for classification. Given a starting image of a bottle with bear texture, we here plot 16 reconstructions each prompted with a different label. The \mathcal{L}_2 distance to the original image is shown in red for all categories except the one with the lowest distance (here: top left), for which the distance is plotted in green. The images with the lowest distance (row 1 column 1: airplane; row 2 column 3: car, row 4 column 4: truck) appear to be categories for which the model is unable to generate a realistic reconstruction, and thus simply sticks close to the original image. The image generated by prompting with the correct shape category (bottle) is shown in row 2 column 2. Since Muse returns images fairly close to the original one if it is not able to generate a realistic reconstruction, the approach of measuring distance between original and reconstruction is not a feasible classification approach for Muse.

C LIMITATIONS

As mentioned in the introduction, using a generative model comes with advantages and disadvantages: potentially better generalization currently comes at the cost of being slower computationally compared to standard discriminative models. While this doesn’t matter much for the purpose of analyses, it is a big drawback in practical applications and any approaches that improve speed would be most welcome—in particular, generating at least one prediction (i.e., image generation) per class as we currently do is both expensive and slow.

Furthermore, the models we investigate all differ from another in more than one ways. For instance, their training data, architecture, and training procedure is not identical, thus any differences between

the models cannot currently be attributed to a single factor. That said, through the inclusion of a set of diverse models covering pixel-based diffusion, latent space diffusion, and autoregressive models we seek to at least cover a variety of generative classifiers in order to ensure that the conclusions we draw are not limited to a narrow set of generative models.

D IMAGE ATTRIBUTION

Rabbit-duck image:

Attribution: Unknown source, Public domain, via Wikimedia Commons.

Link: <https://upload.wikimedia.org/wikipedia/commons/9/96/Duck-Rabbit.png>

Rock image:

Attribution: Mirabeau, CC BY-SA 3.0, via Wikimedia Commons.

Link: https://commons.wikimedia.org/wiki/File:Visage_dans_un_rocher.jpg

Vegetable portrait image:

Attribution: Giuseppe Arcimboldo, Public domain, via Wikimedia Commons.

Link: https://upload.wikimedia.org/wikipedia/commons/4/49/Arcimboldo_Vegetables.jpg

Woman image:

Attribution: W. E. Hill, Public domain, via Wikimedia Commons.

Link: https://upload.wikimedia.org/wikipedia/commons/5/5f/My_Wife_and_My_Mother-In-Law_%28Hill%29.svg

E DETAILS ON RESNET-50 TRAINING WITH DIFFUSION NOISE

We trained a ResNet-50 in exactly the same way as for standard, 90 epoch ImageNet training with the key difference that we added diffusion noise as described by the code below. Since this makes the training task substantially more challenging, we trained the model for 300 instead of 90 epochs. The learning rate was 0.1 with a cosine learning rate schedule, SGD momentum of 0.9, weight decay of 0.0001, and a per device batch size of 64. For diffusion style denoising we used a flag named “sqrt_alphas” which ensures that the noise applied doesn’t completely destroy the image information in most cases. We did not optimize any of those settings with respect to any of the observed findings (e.g., shape bias) since we were interested in generally applicable results.

Listing 1: Python example of how diffusion noise was added as a data augmentation technique.

Copyright 2023 XXXXX REDACTED FOR ANONYMITY XXXXX
#SPDX-License-Identifier: Apache-2.0

```
import dataclasses
from typing import Sequence, Tuple
import grain.tensorflow as tf_grain
import jax
import tensorflow as tf

@dataclasses.dataclass(frozen=True)
class AddNoise(tf_grain.MapTransform):
    """Adds diffusion-style noise to the image."""
    sqrt_alphas: bool = True

    def map(self, features: FlatFeatures) -> FlatFeatures:
        image = features["image"]
        alpha = tf.random.uniform([])
        if self.sqrt_alphas:
            alpha = tf.sqrt(alpha)
        std = tf.sqrt(1 - alpha * alpha)
```

```

image = image * alpha + std * tf.random.normal(image.shape)
features["image"] = image
features["noise_level"] = std
return features

```

F ADDITIONAL PLOTS FOR MODEL-VS-HUMAN BENCHMARK

We here plot detailed performance for all models with respect to a few different properties of interest / metrics:

- Aggregated performance across 17 datasets: Figure 9
- Out-of-distribution accuracy: Figure 10 for parametric datasets and Figure 11 for nonparametric datasets
- Model-to-human error consistency: Figure 10
- Human-to-human, model-to-model error consistency (for nonparametric datasets): Figures 6 and 14 to 17
- Shape bias: Figure 12 and Figure 13

All metrics are based on the model-vs-human toolbox and explained in more detail in (Geirhos et al., 2021).

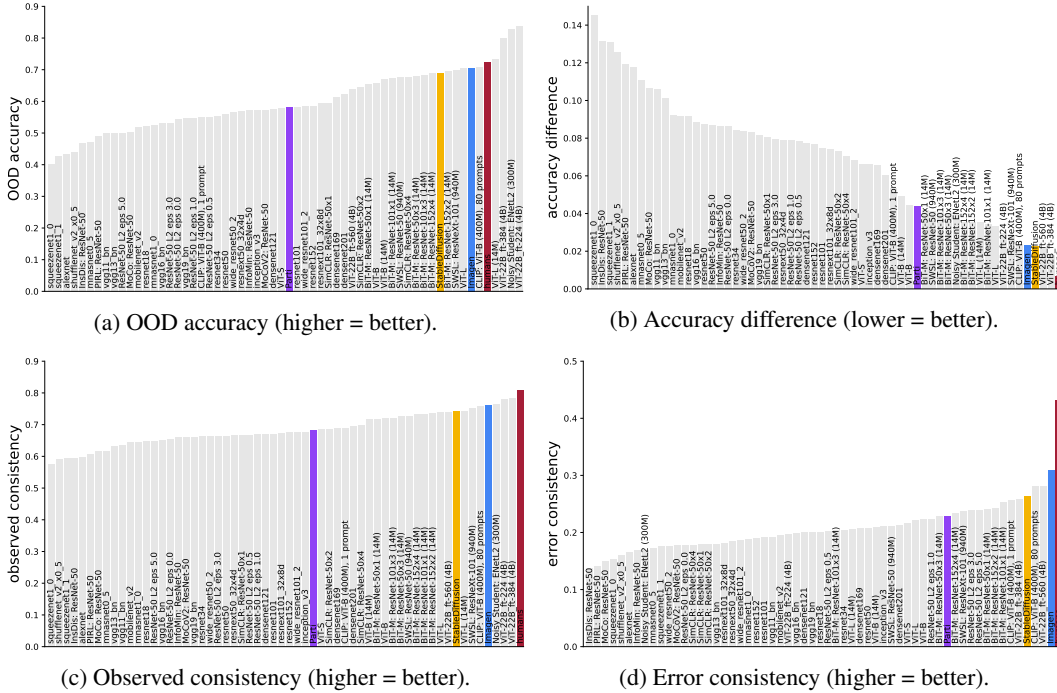


Figure 9: Benchmark results for different models, aggregated over datasets.

G QUANTITATIVE BENCHMARK SCORES AND RANKINGS

Table 2 and Table 3 list the detailed performance aggregated across 17 datasets for each model, with the former focusing on metrics related to “most human-like object recognition behavior” and the latter focusing on out-of-distribution accuracy.

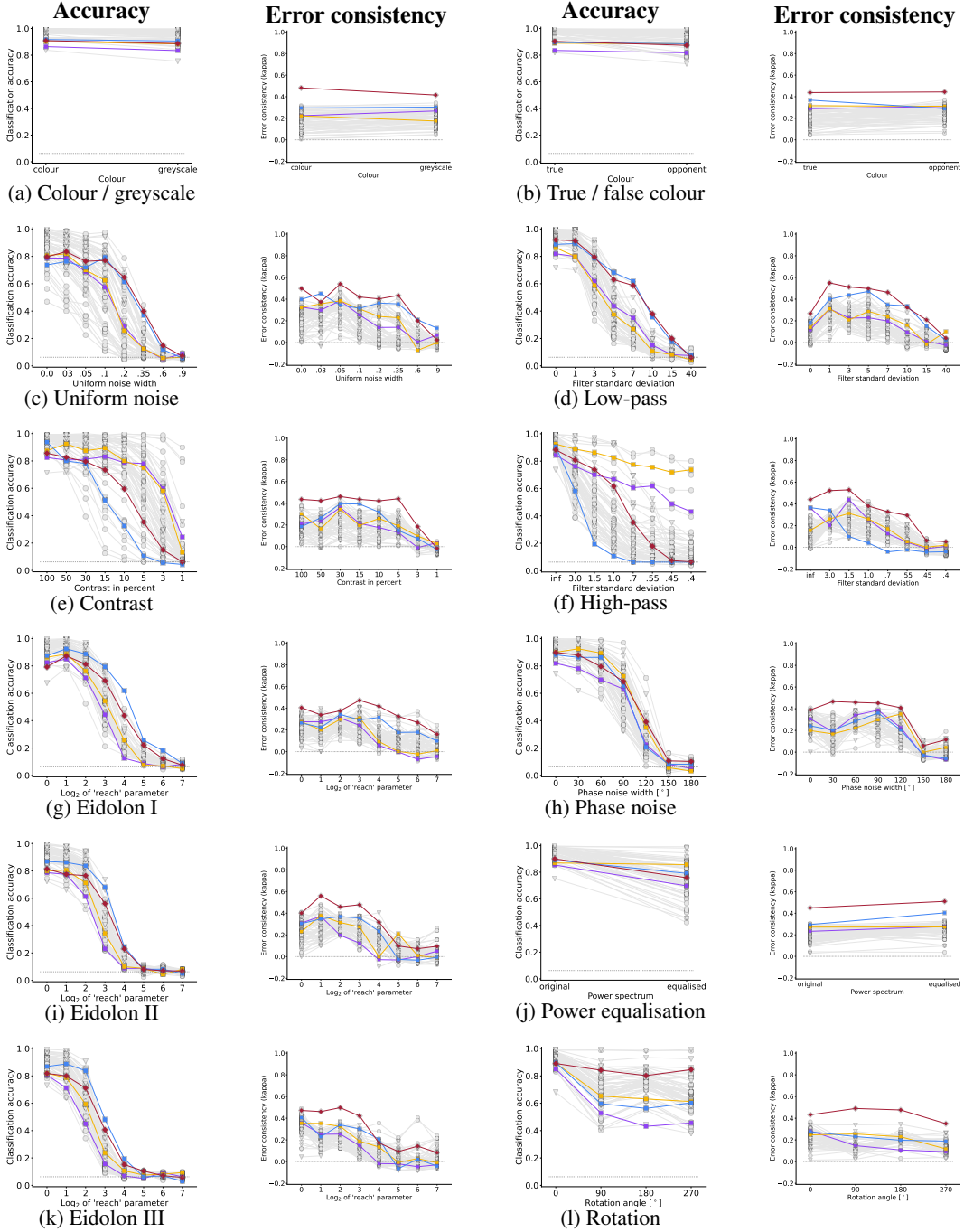


Figure 10: OOD accuracy and error consistency across all twelve parametric datasets from Geirhos et al. (2021). Error consistency results for nonparametric datasets are plotted in Figures 6 and 14 to 17.

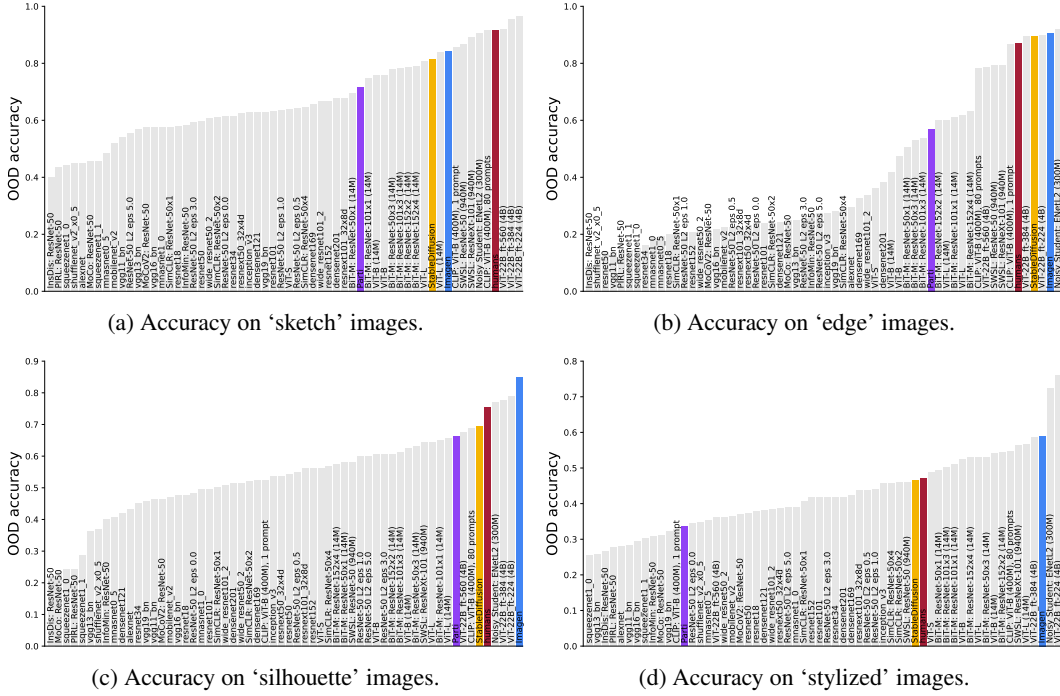


Figure 11: OOD accuracy on all four nonparametric datasets (i.e., datasets with only a single corruption type and strength).

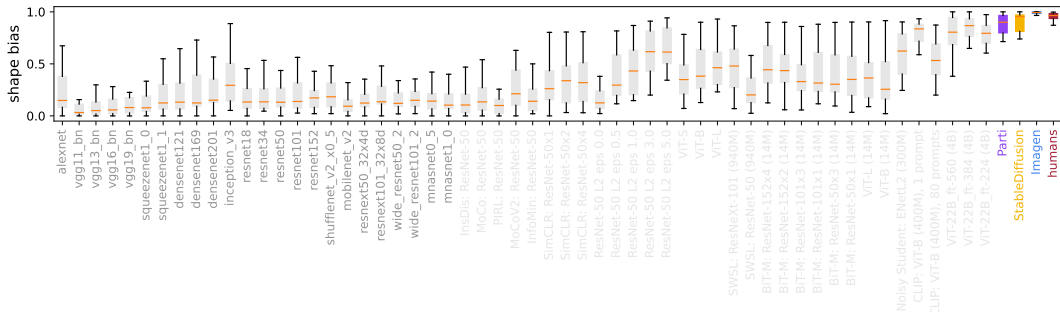


Figure 12: Zero-shot generative classifiers achieve a **human-level shape bias**: 99% for **Imagen**, 93% for **Stable Diffusion**, 92% for **Parti** and 92–99% for individual **human observers** (96% on average). This figure shows boxplots highlighting the spread across 16 categories for each model as a different way of visualizing the data from Figure 1.

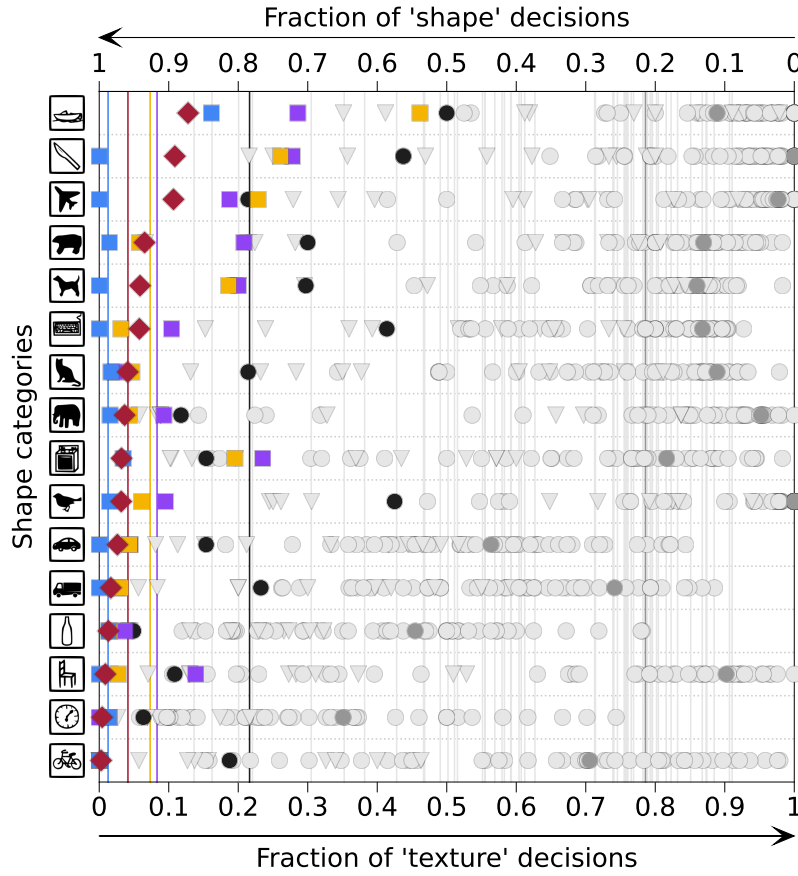


Figure 13: Shape bias before and after training with diffusion noise: This figure shows how ResNet-50 shape bias increases from 21% for a vanilla model (dark grey circles) to 78% when trained and evaluated with diffusion noise (black circles). Horizontal lines indicate the average shape bias across categories. Other models as in Figure 12.

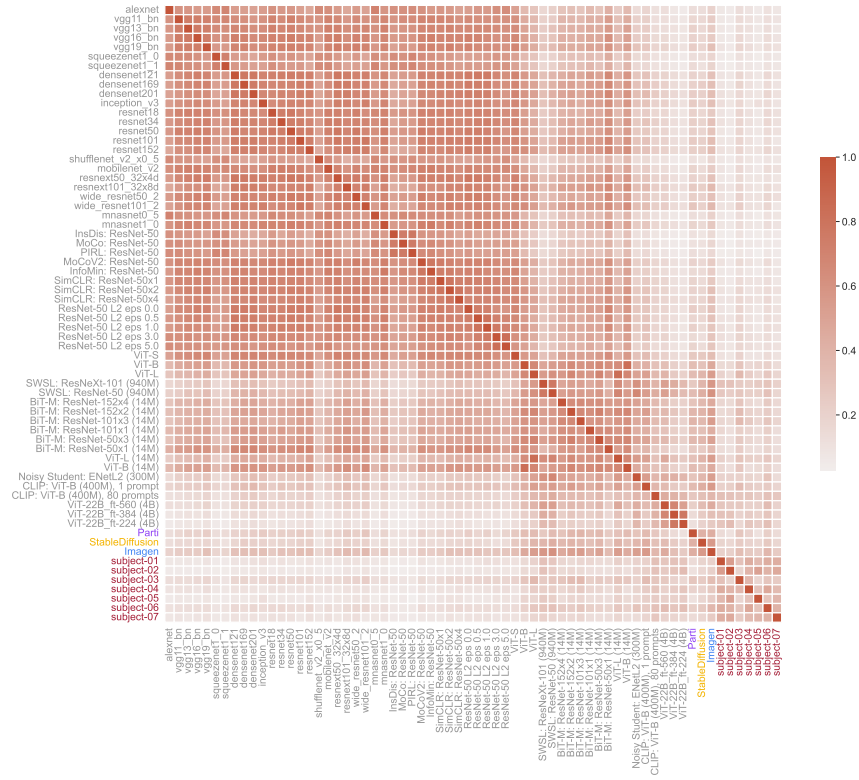


Figure 14: Error consistency for 'sketch' images.

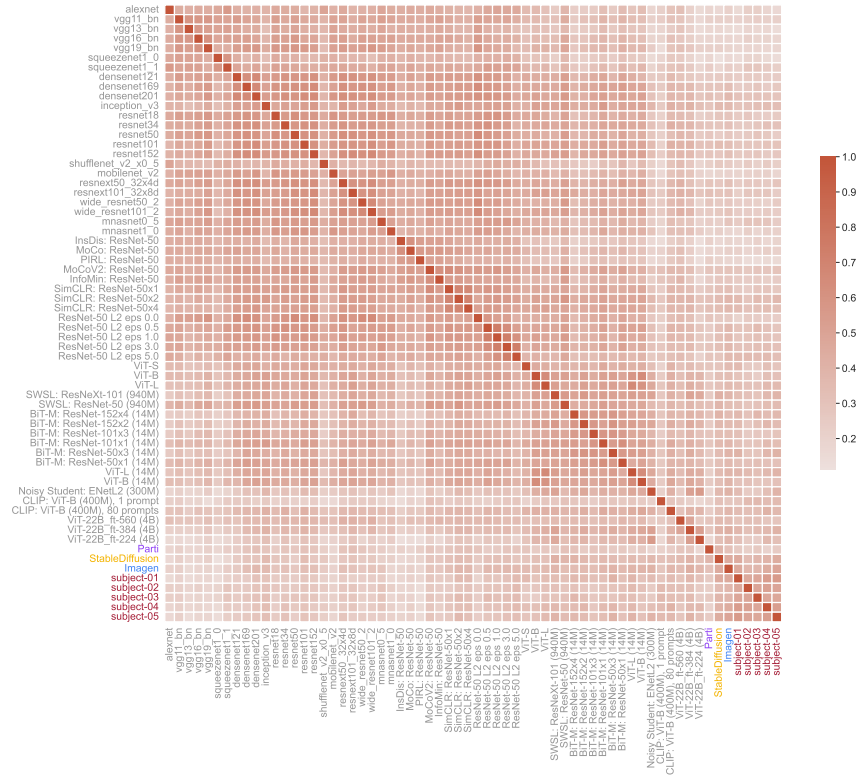


Figure 15: Error consistency for 'stylized' images.

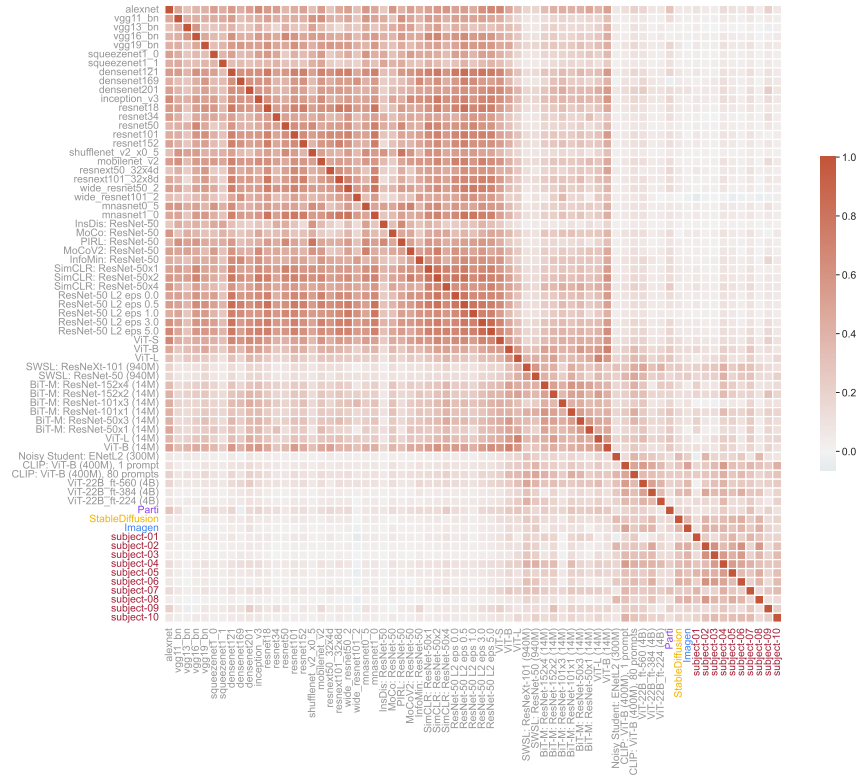


Figure 16: Error consistency for ‘edge’ images.

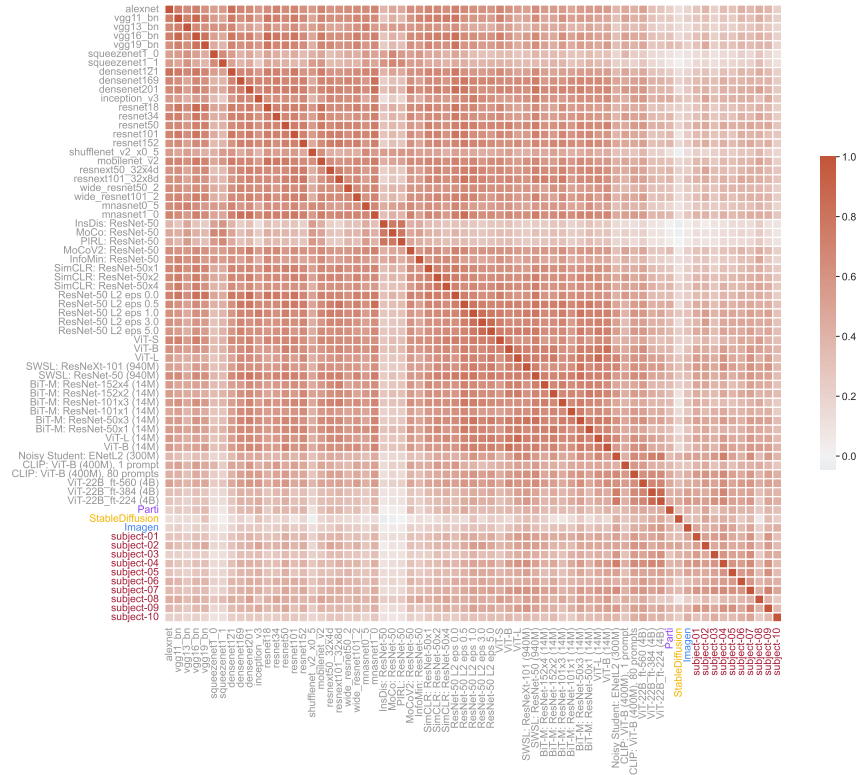


Figure 17: Error consistency for ‘silhouette’ images.

Table 2: Benchmark table of model results for most human-like behaviour, aggregated over all 17 datasets from Geirhos et al. (2021). The three metrics “accuracy difference” “observed consistency” and “error consistency” each produce a different model ranking. The mean rank of a model across those three metrics is used to rank the models on our benchmark.

model	accuracy diff. ↓	obs. consistency ↑	error consistency ↑	mean rank ↓
ViT-22B_ft-384 (4B)	0.018	0.783	0.258	2.333
Imagen (860M)	0.023	0.761	0.309	3.000
ViT-22B_ft-560 (4B)	0.022	0.739	0.281	4.333
CLIP: ViT-B (400M), 80 prompts	0.023	0.758	0.281	4.333
StableDiffusion	0.023	0.743	0.264	5.000
SWSL: ResNeXt-101 (940M)	0.028	0.752	0.237	8.000
BiT-M: ResNet-101x1 (14M)	0.034	0.733	0.252	9.333
BiT-M: ResNet-152x2 (14M)	0.035	0.737	0.243	10.000
ViT-L	0.033	0.738	0.222	11.667
BiT-M: ResNet-152x4 (14M)	0.035	0.732	0.233	12.667
ViT-L (14M)	0.035	0.744	0.206	14.000
ViT-22B_ft-224 (4B)	0.030	0.781	0.197	14.000
BiT-M: ResNet-50x3 (14M)	0.040	0.726	0.228	14.333
BiT-M: ResNet-50x1 (14M)	0.042	0.718	0.240	14.667
CLIP: ViT-B (400M), 1 prompt	0.054	0.688	0.257	16.000
SWSL: ResNet-50 (940M)	0.041	0.727	0.211	16.667
ViT-B	0.044	0.719	0.223	17.000
BiT-M: ResNet-101x3 (14M)	0.040	0.720	0.204	19.333
ViT-B (14M)	0.049	0.717	0.209	20.000
densenet201	0.060	0.695	0.212	20.333
Noisy Student: ENetL2 (300M)	0.040	0.764	0.169	22.333
ViT-S	0.066	0.684	0.216	22.333
densenet169	0.065	0.688	0.207	23.000
inception_v3	0.066	0.677	0.211	23.333
ResNet-50 L2 eps 1.0	0.079	0.669	0.224	26.667
ResNet-50 L2 eps 3.0	0.079	0.663	0.239	27.667
SimCLR: ResNet-50x4	0.071	0.698	0.179	30.333
wide_resnet101_2	0.068	0.676	0.187	30.333
ResNet-50 L2 eps 0.5	0.078	0.668	0.203	31.000
densenet121	0.077	0.671	0.200	31.000
SimCLR: ResNet-50x2	0.073	0.686	0.180	31.333
resnet152	0.077	0.675	0.190	31.667
resnet101	0.074	0.671	0.192	31.667
resnext101_32x8d	0.074	0.674	0.182	32.667
ResNet-50 L2 eps 5.0	0.087	0.649	0.240	32.667
resnet50	0.087	0.665	0.208	34.333
resnet34	0.084	0.662	0.205	35.000
vgg19_bn	0.081	0.660	0.200	35.667
resnext50_32x4d	0.079	0.666	0.184	36.333
SimCLR: ResNet-50x1	0.080	0.667	0.179	38.000
resnet18	0.091	0.648	0.201	40.333
vgg16_bn	0.088	0.651	0.198	40.333
wide_resnet50_2	0.084	0.663	0.176	41.667
MoCoV2: ResNet-50	0.083	0.660	0.177	42.000
mobilenet_v2	0.092	0.645	0.196	43.000
ResNet-50 L2 eps 0.0	0.086	0.654	0.178	43.333
mnasnet1_0	0.092	0.646	0.189	44.333
vgg11_bn	0.106	0.635	0.193	44.667
InfoMin: ResNet-50	0.086	0.659	0.168	45.333
vgg13_bn	0.101	0.631	0.180	47.000
mnasnet0_5	0.110	0.617	0.173	51.000
MoCo: ResNet-50	0.107	0.617	0.149	53.000
alexnet	0.118	0.597	0.165	53.333
squeezenet1_1	0.131	0.593	0.175	53.667
PIRL: ResNet-50	0.119	0.607	0.141	54.667
shufflenet_v2_x0.5	0.126	0.592	0.160	55.333
InsDis: ResNet-50	0.131	0.593	0.138	56.667
squeezenet1_0	0.145	0.574	0.153	57.000

Table 3: Benchmark table of model results for highest out-of-distribution robustness, aggregated over all 17 datasets from Geirhos et al. (2021).

model	OOD accuracy \uparrow	rank \downarrow
ViT-22B_ft-224 (4B)	0.837	1.000
Noisy Student: ENetL2 (300M)	0.829	2.000
ViT-22B_ft-384 (4B)	0.798	3.000
ViT-L (14M)	0.733	4.000
CLIP: ViT-B (400M), 80 prompts	0.708	5.000
Imagen (860M)	0.706	6.000
ViT-L	0.706	7.000
SWSL: ResNeXt-101 (940M)	0.698	8.000
BiT-M: ResNet-152x2 (14M)	0.694	9.000
StableDiffusion	0.689	10.000
BiT-M: ResNet-152x4 (14M)	0.688	11.000
BiT-M: ResNet-101x3 (14M)	0.682	12.000
BiT-M: ResNet-50x3 (14M)	0.679	13.000
SimCLR: ResNet-50x4	0.677	14.000
SWSL: ResNet-50 (940M)	0.677	15.000
BiT-M: ResNet-101x1 (14M)	0.672	16.000
ViT-B (14M)	0.669	17.000
ViT-B	0.658	18.000
BiT-M: ResNet-50x1 (14M)	0.654	19.000
SimCLR: ResNet-50x2	0.644	20.000
ViT-22B_ft-560 (4B)	0.639	21.000
densenet201	0.621	22.000
densenet169	0.613	23.000
SimCLR: ResNet-50x1	0.596	24.000
resnext101_32x8d	0.594	25.000
resnet152	0.584	26.000
wide_resnet101_2	0.583	27.000
resnet101	0.583	28.000
ViT-S	0.579	29.000
densenet121	0.576	30.000
MoCoV2: ResNet-50	0.571	31.000
inception_v3	0.571	32.000
InfoMin: ResNet-50	0.571	33.000
resnext50_32x4d	0.569	34.000
wide_resnet50_2	0.566	35.000
resnet50	0.559	36.000
resnet34	0.553	37.000
ResNet-50 L2 eps 0.5	0.551	38.000
CLIP: ViT-B (400M), 1 prompt	0.550	39.000
ResNet-50 L2 eps 1.0	0.547	40.000
vgg19_bn	0.546	41.000
ResNet-50 L2 eps 0.0	0.545	42.000
ResNet-50 L2 eps 3.0	0.530	43.000
vgg16_bn	0.530	44.000
mnasnet1_0	0.524	45.000
resnet18	0.521	46.000
mobilenet_v2	0.520	47.000
MoCo: ResNet-50	0.502	48.000
ResNet-50 L2 eps 5.0	0.501	49.000
vgg13_bn	0.499	50.000
vgg11_bn	0.498	51.000
PIRL: ResNet-50	0.489	52.000
mnasnet0.5	0.472	53.000
InsDis: ResNet-50	0.468	54.000
shufflenet_v2_x0.5	0.440	55.000
alexnet	0.434	56.000
squeezenet1_1	0.425	57.000
squeezenet1_0	0.401	58.000