Supplementary Material

Understanding Invariance via Feedforward Inversion
of Discriminatively Trained Classifiers

1 Noise resampling

In Figures 1 and 2, we provide additional noise resampling results for both robust and non-robust models. For each block of 9 images, the upper left is the input to the classifier. The next 8 are the same logits with different noise vectors fed into the generator.

2 Interpolations between logits

In Figures 3 and 4 and we interpolate between two logits of two vectors linearly. We find robust models to give a smoother sequence.

3 Interpolations between noise vectors

In Figures 5 and 6 and we interpolate between two noise vectors linearly. We find that the robust model has a smaller semantic difference between the two endpoints.

4 Reconstructing Incorrectly Classified Images

In figures 7 and 8 we show additional examples for incorrectly classified samples. We find that even for incorrectly classified samples, reconstructions are of high quality.

5 Iterative Stability

We explore iterative stability of the model; if we pass the reconstructed sample back through the classifier, how often does it predict the same class as the ground truth image? The results are much different for the robust and non-robust models. We take 500 samples from 50 classes of ImageNet for this experiment. For the robust model; if the ground truth sample was correctly classified, the reconstruction is classified identically as the ground truth 54% of the time. If the ground truth was incorrectly classified, the reconstruction is classified identically

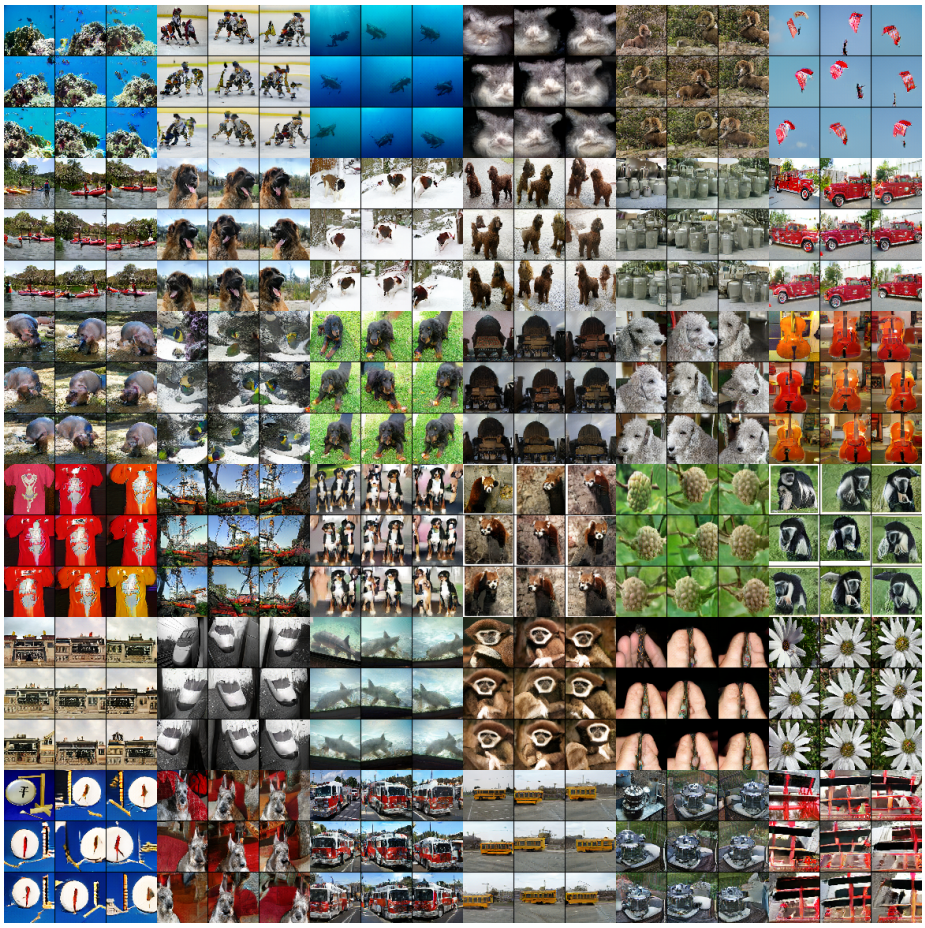

Figure 1: Additional noise resamplings for the robust ResNet-152 on randomly chosen images from the ImageNet validation set.

as the ground truth 35% of the time. There is a substantial difference between correctly classified samples, despite qualitative observations that reconstructions are similarly good for both. The results for the non-robust model are 49.3% and 29.9% respectively. The robust model is more stable, implying that the inversion model is better at preserving features used for classification with the robust model.

6 Analysis of scale and shift effects

In the main paper, we show that manipulating logits before they are input to inversion model resulted in systematic changes; both scale and shift seemed to encode brightness for the robust ResNet-152 model, while it didn't have

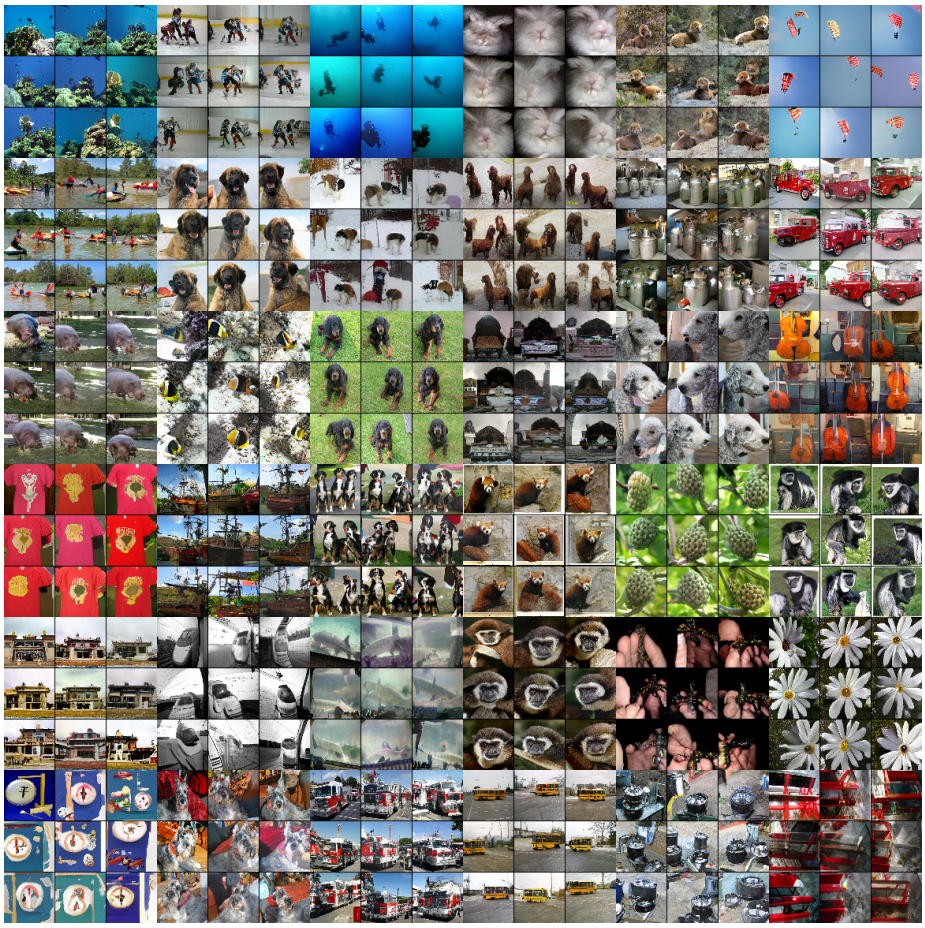

Figure 2: Additional noise resmaplings for the non-robust ResNet-152.

as pronounced an effect on the non-robust model. Here, we show exactly this experimentally for shifts. We start with a batch of images, and then multiply its pixel values by a fixed value to adjust the brightness. Then for each brightness-adjusted image, we compute robust logits. We then find a best-fit shift to get back to the non-brightness adjsuted logits. In other words, we fit the shift parameter in the mode.: $logits_{original} = logits_{brightness-adjusted} + shift$. We do this independently for each image, and report the mean across brightness factors in Figure 9 We find that as the brightness increases , the best-shift does as well. This verifies the finding in the main paper. However, for the non-robust model, the expected shift peaks and then decreases, shown in 10.

For scale, we perform a similar experiment, except we fit the scale in the model $logits_{original} = logits_{brightness-adjusted} * scale$. We find that the relationship for scale is weaker than for shift. However, locally around the identity

transform, there is still a positive relationship between scale and brightness for the robust ResNet-152 (Figure 11); the best-fit scale peaks at a greater brightness than the identity transform. On the other hand, for the non-robust ResNet-152 (Figure 12), the identity transform represents the peak expected scale of 1.

We repeat the experiment for image sharpness manipulations. Image sharpness of 0 means a smoothed image, while image sharpness of 2 means a sharpened image. We show results for the Robust ResNet-152 in Figures 13 and 15, and non-robust ResNet-152 in Figures 14 and 16.

We find that in both models, best-fit scale and shift positively correlate with sharpness. However, for the Robust model, the magnitudes of expected scales and shifts is much much larger.

6.1 Effect of image-space rotations on reconstructions

In the main paper, we explored manipulations of the logits, i.e., the input to the decoder. With logit manipulations, we can infer what aspects of the image will cause the classifier to produce corresponding variation in the logits. In this section, we directly manipulate images, in order to determine whether the robust classifier preserves information about the manipulation and whether the decoder can recover the original image. The particular manipulation we study is rotation of the image by 90 degrees. Figure 17 shows an original image, the reconstruction of this image from the robust logits, and the reconstruction of this image rotated by 90 degrees. To make it easier to compare images, we counter-rotate the latter reconstruction so that it will be identical to the original image if it is reconstructed veridically. We observe some signal in the rotated reconstruction, but clearly there is degradation. For example, animals—which are nearly always upright in ImageNet—are horribly distorted. The classifier and/or decoder do not appear to generalize well outside of the data distribution they were trained on (i.e., upright data). Interestingly, textures are fairly well reconstructed, such as the background brush in the third row, left triplet, which makes sense given that textures tend not to have a reliable orientation in the natural world.

7 Training Details

7.1 Network Architectures

We use a BigGAN generator configured for 64x64 images. The layer configuration is listed below. Each residual block contains 3 conditional-batch norms and a single 2x upsampling operation, in addition to convolutions and nonlinearities. We use hierarchical noise input, detailed in [1]. For more details, please see [1] and the compare_gan codebase.

Similarly, we use the BigGAN discriminator. There are 4 resblock with downsampling, and one without. The proection discriminator, described in the

Layer	Out Resolution	Out Channels
Linear(noise)	4	1536
Residual Block Upsampling	8	1536
Residual Block Upsampling	16	768
Residual Block Upsampling	32	384
Residual Block Upsampling	64	192
Non-local block (self-attention)	64	192
BN, ReLU, and Conv	64	3
Tanh	64	3

Table 1: Generator architecture

main paper, comes at the end. Again, for more details, please see [1] and the compare_gan codebase.

Layer	Out Resolution	Out Channels
ResBlock Down	32	192
Non-local block (self-attention)	32	192
ResBlock Down	16	384
ResBlock Down	8	768
ResBlock Down	4	1536
ResBlock	4	1536
ReLU, Global Sum pooling	1	1536
Conditional Projection	1	1

Table 2: Discriminator architecture

We train with the Adam optimizer with β_1 set to 0 and β_2 set to 0.999, and a learning rate of 0.0001 for the generator and 0.0005 for the discriminator. We take 2 discriminator steps for every generator step. We initialize weights orthognally, and use spectral norm in both the generator and discriminator. For more details, see below.

7.2 Training Configuration

Below, the exact configuration used for the inversion models in the paper. It can be used directly with the compare_gan code.

```
# Parameters for AdamOptimizer:
# ==============================================
AdamOptimizer.beta1 = 0.0
AdamOptimizer.beta2 = 0.999

# Parameters for D:
# ==============================================
D.spectral_norm = True
```

```
# Parameters for dataset:
# ==============================================
dataset.name = 'imagenet_64'

# Parameters for resnet_biggan.Discriminator:
# ==============================================
resnet_biggan.Discriminator.project_labels = True

# Parameters for eval_z:
# ==============================================
eval_z.distribution_fn = @tf.random.stateless_normal

# Parameters for G:
# ==============================================
G.batch_norm_fn = @conditional_batch_norm
G.spectral_norm = True

# Parameters for resnet_biggan.Generator:
# ==============================================
resnet_biggan.Generator.embed_labels = True
resnet_biggan.Generator.hierarchical_z = True

# Parameters for loss:
# ==============================================
loss.fn = @hinge

# Parameters for ModularGAN:
# ==============================================
ModularGAN.conditional = True
ModularGAN.d_inputs = ['label']
ModularGAN.d_lr = 0.0005
ModularGAN.d_optimizer_fn = @tf.train.AdamOptimizer
ModularGAN.g_inputs = ['label']
ModularGAN.g_lr = 0.0001
ModularGAN.g_optimizer_fn = @tf.train.AdamOptimizer
ModularGAN.g_use_ema = True

# Parameters for options:
# ==============================================
options.architecture = 'resnet_biggan_arch'
options.batch_size = 2048
options.disc_iters = 2
options.gan_class = @ModularGAN
options.lamba = 1
options.training_steps = 250000
options.z_dim = 120
```

```
# Parameters for penalty:
# ==============================================
penalty.fn = @no_penalty

# Parameters for run_config:
# ==============================================
run_config.iterations_per_loop = 500
run_config.save_checkpoints_steps = 2500
run_config.tf_random_seed = 8

# Parameters for spectral_norm:
# ==============================================
spectral_norm.singular_value = 'auto'

# Parameters for standardize_batch:
# ==============================================
standardize_batch.decay = 0.9
standardize_batch.epsilon = 1e-05
standardize_batch.use_moving_averages = False

# Parameters for weights:
# ==============================================
weights.initializer = 'orthogonal'

# Parameters for z:
# ==============================================
z.distribution_fn = @tf.random.normal
```

References

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

Figure 3: Linear interpolations between two logit vectors for the robust ResNet-152. Upper-left and lower-right are ground truth images.

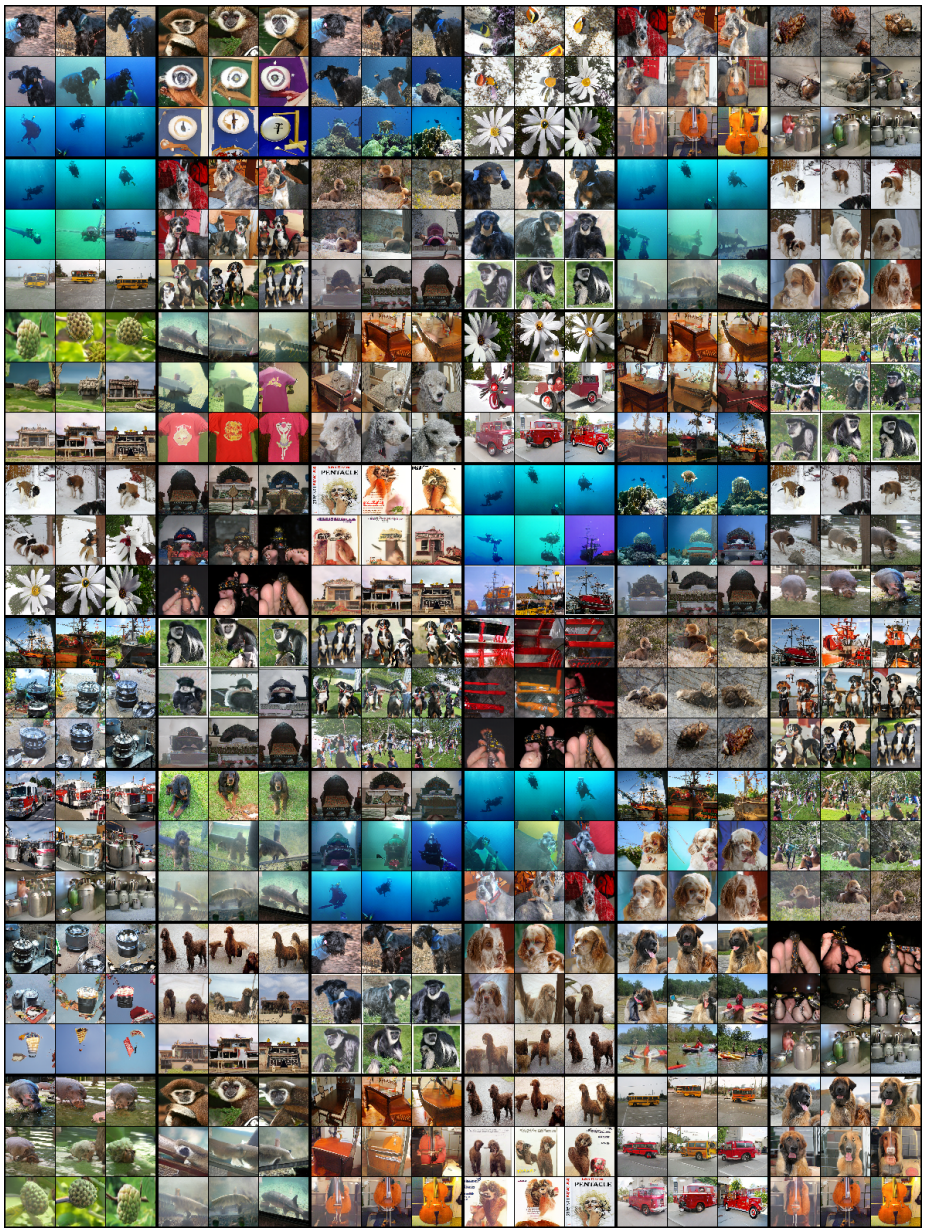

Figure 4: Linear interpolations between two logit vectors for the non-robust ResNet-152. Upper-left and lower-right are ground truth images.

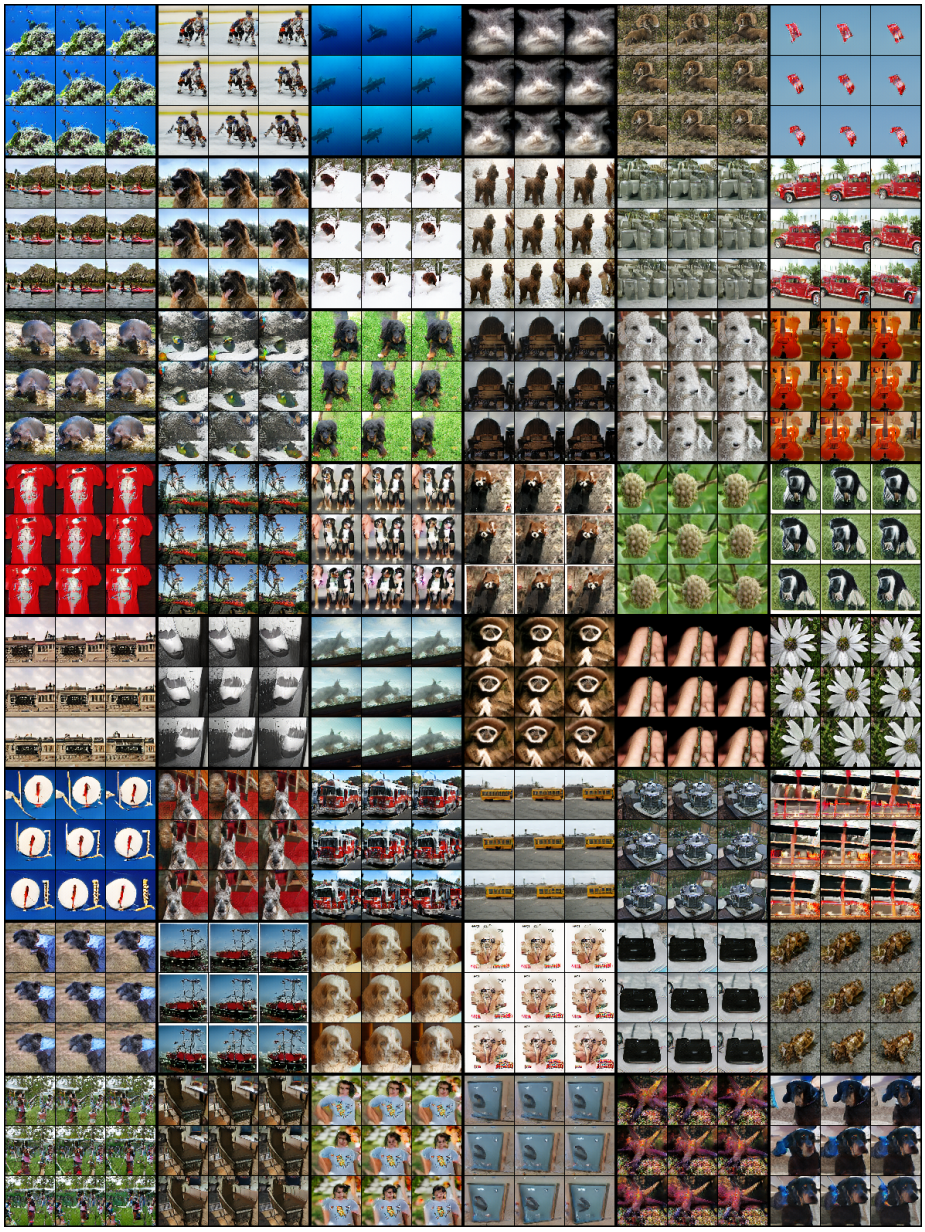

Figure 5: Linear interpolations between two noise inputs for the robust ResNet-152

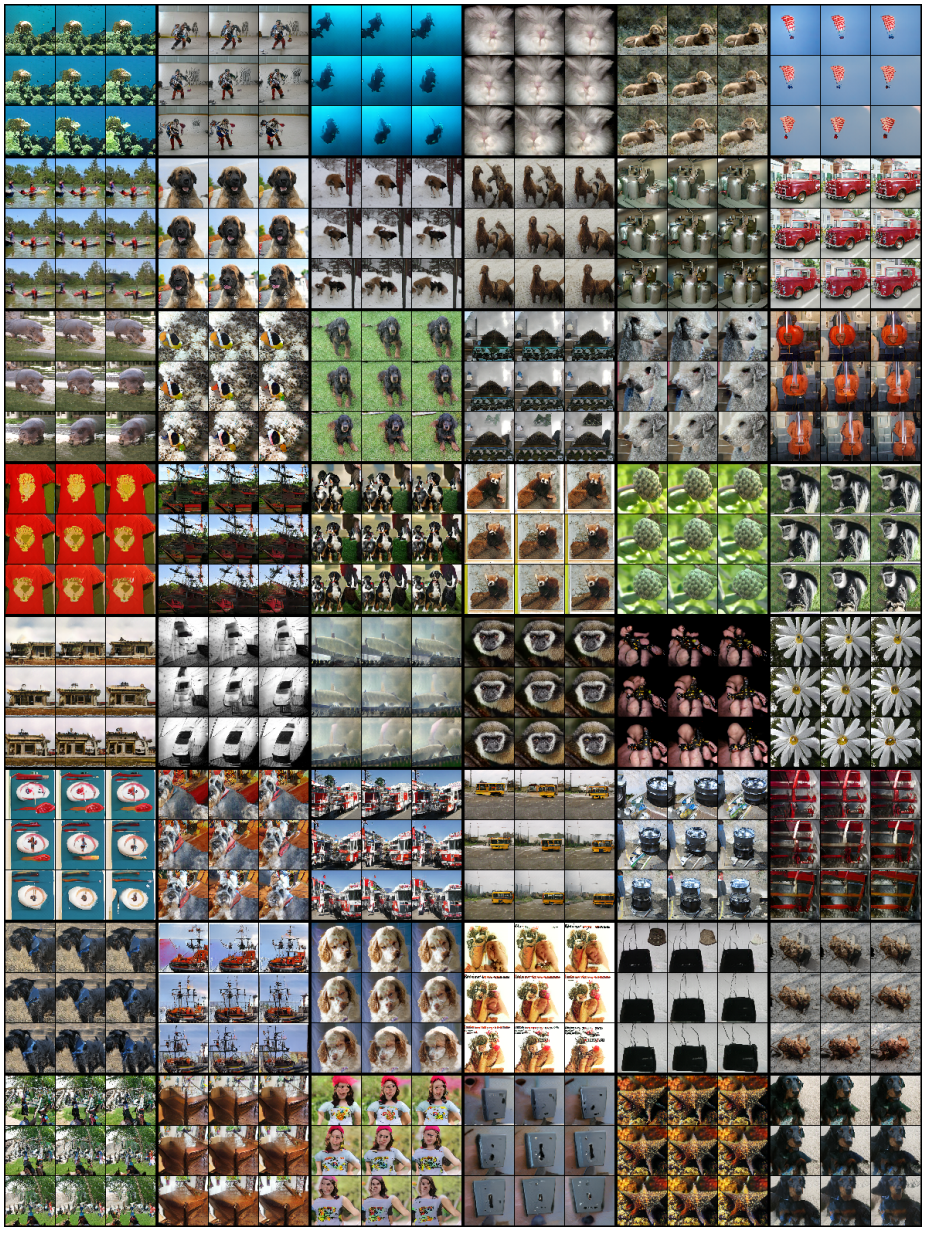

Figure 6: Linear interpolations between two noise inputs for the non-robust ResNet-152

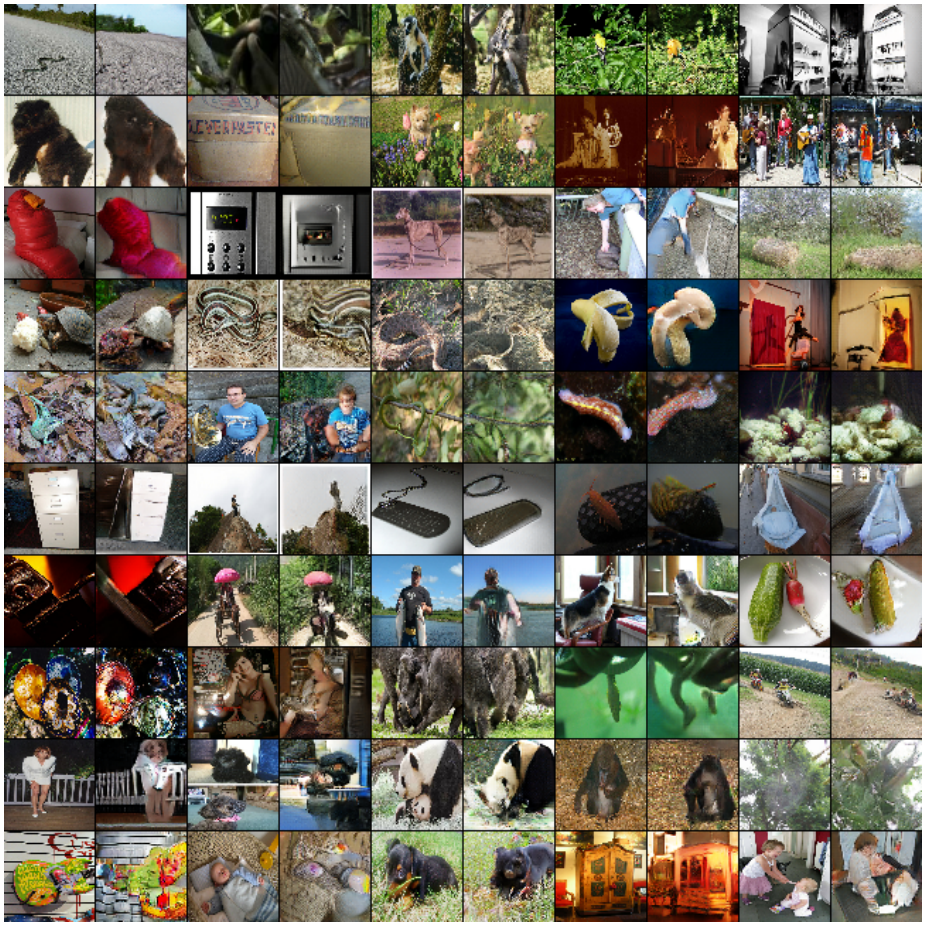

Figure 7: Reconstructions of incorrectly classified images from the robust ResNet-152.

Figure 8: Reconstructions of incorrectly classified images from the non-robust ResNet-152.

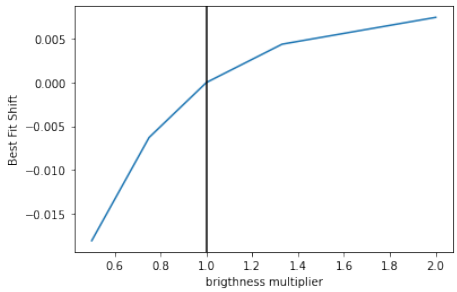

Figure 9: Plotting best-fit shifts vs. brightness factor for Robust ResNet-152. As brightness increases, so does expected shift. Brightness factor of 1 corresponds to the identity function, so expected shift is 0, this is represented by the vertical line.

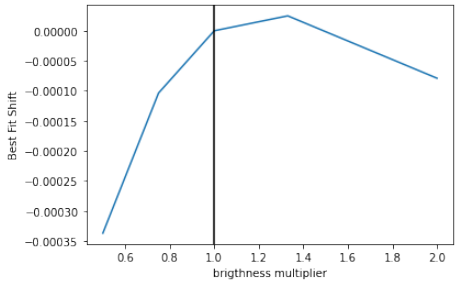

Figure 10: Plotting best-fit shifts vs. brightness factor, for non-robust ResNet-152. There is a much weaker relationship than for the robust ResNet-152 model. Brightness factor of 1 corresponds to the identity function, so expected shift is 0, this is represented by the vertical line.

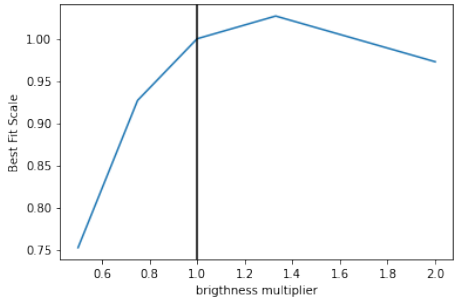

Figure 11: Plotting best-fit scales vs. brightness factor, for robust ResNet-152. Brightness factor of 1 corresponds to the identity function, this is represented by the vertical line. Locally, around this line, there is a positive relationship between scale and brightness.

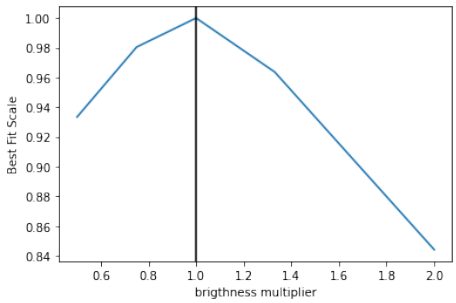

Figure 12: Plotting best-fit scales vs. brightness factor, for non-robust ResNet-152. Brightness factor of 1 corresponds to the identity function, this is represented by the vertical line.Locally, around this line, there is no positive relationship between scale and brightness.

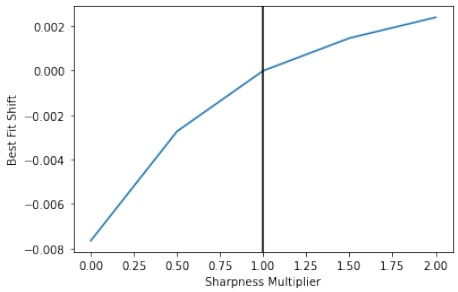

Figure 13: Plotting best-fit shifts vs. sharpness factor, for Robust ResNet-152. Sharpness factor of 1 corresponds to the identity function, so expected shift is 0, this is represented by the vertical line.

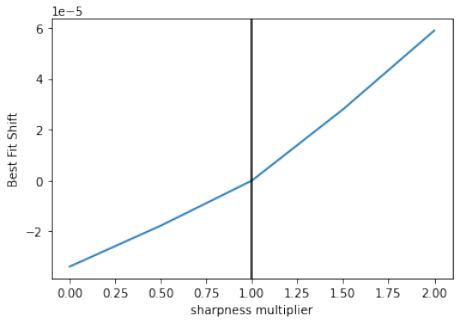

Figure 14: Plotting best-fit shifts vs. sharpness factor, for non-robust ResNet-152. Sharpness factor of 1 corresponds to the identity function, so expected shift is 0, this is represented by the vertical line. There is a positive relationship.

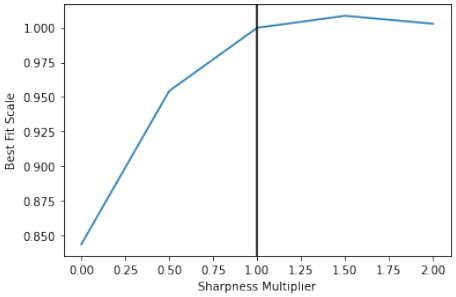

Figure 15: Plotting best-fit scales vs. sharpness factor, for robust ResNet-152. Brightness factor of 1 corresponds to the identity function, this is represented by the vertical line. Although there is a positive relationship, the best fit shifts are much smaller magnitude.

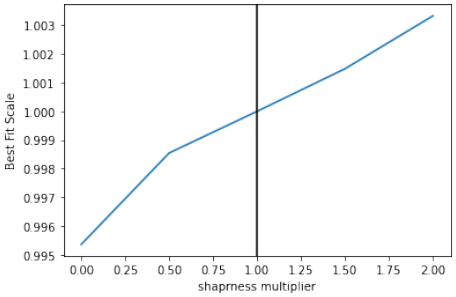

Figure 16: Plotting best-fit scales vs. sharpness factor, for non-robust ResNet-152. Sharpness factor of 1 corresponds to the identity function, this is represented by the vertical line. There is a strong relationship. Again, although there is a positive relationship, the expected scales magnitudes are much smaller than for the robust model.

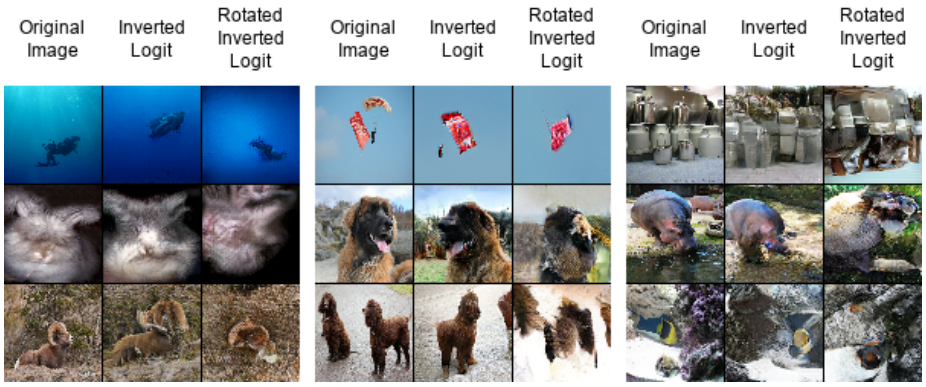

Figure 17: Reconstruction of rotations. We compare images encoded with Robust ResNet-152 (left column) to the their reconstructions (middle), and reconstructions of the same image rotated by 90 degrees (right). We counter-rotate the reconstruction for ease of comparison. The rotated samples reconstruct textures reasonably well but objects are horribly disfigured.