Making Sense of CNNs: Interpreting Deep Representations & Their Invariances with INNs

—

Supplementary Materials

A Implementation Details

A.1 Autoencoder E, D

In Sec. 3.1, we introduced an autoencoder to obtain a representation $\bar{z}$ of x, which includes the invariances abstracted away by a given model representation z. This autoencoder consists of an encoder $E(x)$, and a decoder $D(\bar{z})$.

Because the INNs t and e transform the distribution of $\bar{z}$, we must ensure a strictly positive density for $\bar{z}$ to avoid degenerate solutions. This is readily achieved with a stochastic encoder, *i.e.* we predict mean $E(x)_\mu$ and diagonal $E(x)_{\sigma^2}$ of a Gaussian distribution, and obtain the desired representation as $\bar{z} \sim \mathcal{N}(\bar{z}|E(x)_\mu, \operatorname{diag}(E(x)_{\sigma^2}))$.

Following [10], we train this autoencoder as a Variational Autoencoder using the reparameterization trick [26, 48] to match the encoded distribution to a standard normal distribution, and jointly learn the output variance γ under an image metric $\|x - \bar{x}\|$ to avoid blurry reconstructions. The resulting loss function is thus

$$\mathcal{L}(E, D, \gamma) = \mathbb{E}_{\substack{x \sim p(x) \\ \epsilon \sim \mathcal{N}(\epsilon|0,\mathbb{1})}} \Bigg[\frac{1}{\gamma} \|x - D(E(x)_\mu + \sqrt{\operatorname{diag}(E(x)_{\sigma^2})}\, \epsilon)\| + \log \gamma + \frac{1}{2} \sum_{i=1}^{N_{\bar{z}}} \left\{ (E(x)_\mu)_i^2 + (E(x)_{\sigma^2})_i - 1 - \log(E(x)_{\sigma^2})_i \right\} \Bigg]$$

For experiments on *ColorMNIST*, we use the squared L^2 norm for the image metric, and the encoder and decoder architectures are summarized in Tab. S1.

Table S1. Autoencoder architecture for *ColorMNIST* at resolution 28×28.

Table S2. Encoder.

RGB image $x \in \mathbb{R}^{28\times28\times3}$
Conv, Norm, LReLU $\to \mathbb{R}^{14\times14\times64}$
Conv, Norm, LReLU $\to \mathbb{R}^{7\times7\times128}$
FC $\mapsto (\mu, \sigma^2) \in \mathbb{R}^{64} \times \mathbb{R}^{64}$

Table S3. Decoder.

$z \in \mathbb{R}^{64} \sim \mathcal{N}(\mu, \operatorname{diag}(\sigma^2))$
FC $\to \mathbb{R}^{7\times7\times128}$
Conv transpose, Norm, LReLU $\to \mathbb{R}^{14\times14\times64}$
Conv transpose, Tanh $\to \mathbb{R}^{28\times28\times3}$

Table S4. Architectures used to compute image metrics for *CelebA*, *AnimalFaces* and *Animals* at resolution 128×128.

Table S5. VGG-16 pretrained on ImageNet for feature extraction. Output of bold layers are used to compute feature distances.

RGB image $x \in \mathbb{R}^{128\times128\times3}$
$2\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{128\times128\times64}$
MaxPool $\rightarrow \mathbb{R}^{64\times64\times64}$
$2\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{64\times64\times128}$
MaxPool $\rightarrow \mathbb{R}^{32\times32\times128}$
$3\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{32\times32\times256}$
MaxPool $\rightarrow \mathbb{R}^{16\times16\times256}$
$3\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{16\times16\times512}$
MaxPool $\rightarrow \mathbb{R}^{8\times8\times512}$
$3\times$ **Conv, ReLU** $\rightarrow \mathbb{R}^{8\times8\times512}$

Table S6. Discriminator. All convolutions use kernel size 4. Norm refers to Batch Normalization, Leaky ReLU uses slope parameter 0.2.

RGB image $x \in \mathbb{R}^{128\times128\times3}$
Conv down, LReLU $\rightarrow \mathbb{R}^{64\times64\times64}$
Conv down, Norm, LReLU $\rightarrow \mathbb{R}^{32\times32\times128}$
Conv down, Norm, LReLU $\rightarrow \mathbb{R}^{16\times16\times256}$
Conv down, Norm, LReLU $\rightarrow \mathbb{R}^{8\times8\times512}$
Conv, Norm, LReLU $\rightarrow \mathbb{R}^{8\times8\times512}$
Conv $\rightarrow \mathbb{R}^{8\times8\times1}$

For the experiments on *CelebA*, *AnimalFaces* and *Animals*, we use an improved image metric as in [13], which includes a perceptual loss and a discriminator loss. The perceptual loss consists of feature distances obtained from different layers of a fixed, pretrained network. We used a VGG-16 network pretrained on ImageNet and weighted distances of different layers as in [65]. The discriminator is trained along with the autoencoder to distinguish reconstructed images from real images using a binary classification loss, and the autoencoder maximizes the log-probability that reconstructed images are classified as real images. The architectures of VGG-16 and the discriminator are summarized in Tab. S4. For E we use an architecture based on ResNet-101 and for D we use an architecture based on BigGAN, where we include a small fully connected network to replace the class conditioning used in BigGAN by a conditioning on $\bar{z}$. See Tab. S7 for a summary of this autoencoder architecture.

A.2 Details on the INN for Revealing Semantics of Deep Representations

Previous works have successfully applied INNs for density estimation [12], inverse problems [2], and on top of autoencoder representations [15, 61]. This section provides details on how we embed the approach of [15] to reveal the semantic concepts of autoencoder representations $\bar{z}$, *c.f.* Sec. 3.2.

Table S7. Autoencoder architecture for *CelebA*, *AnimalFaces* and *Animals* at resolution 128×128.

Table S8. *Resnet-101* based Encoder.

RGB image $x \in \mathbb{R}^{128\times128\times3}$
Conv down $\to \mathbb{R}^{64\times64\times64}$
Norm, ReLU, MaxPool $\to \mathbb{R}^{32\times32\times64}$
3× BottleNeck $\to \mathbb{R}^{32\times32\times256}$
4× BottleNeck down $\to \mathbb{R}^{16\times16\times512}$
23× BottleNeck down $\to \mathbb{R}^{8\times8\times1024}$
3× BottleNeck down $\to \mathbb{R}^{4\times4\times2048}$
AvgPool, FC $\mapsto (\mu, \sigma^2) \in \mathbb{R}^{128} \times \mathbb{R}^{128}$

Table S9. Decoder based on *BigGAN*.

$\bar{z} \in \mathbb{R}^{128} \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$ 3× (FC, LReLU) $\to \mathbb{R}^{256}$ FC, Softmax $\to \mathbb{R}^{1000}$ Embed $\mapsto h \in \mathbb{R}^{128}$
FC($\bar{z}$) $\to \mathbb{R}^{4\times4\times16\cdot96}$
ResBlock($\bar{z}$,h) up $\to \mathbb{R}^{8\times8\times16\cdot96}$
ResBlock($\bar{z}$,h) up $\to \mathbb{R}^{16\times16\times8\cdot96}$
ResBlock($\bar{z}$,h) up $\to \mathbb{R}^{32\times32\times4\cdot96}$
ResBlock($\bar{z}$,h) up $\to \mathbb{R}^{64\times64\times2\cdot96}$
Non-Local Block $\to \mathbb{R}^{64\times64\times2\cdot96}$
ResBlock($\bar{z}$,h) up $\to \mathbb{R}^{64\times64\times96}$
Norm, ReLU, Conv up $\to \mathbb{R}^{128\times128\times3}$
Tanh $\mapsto \bar{x} \in \mathbb{R}^{128\times128\times3}$

Since we will never have examples for all relevant semantic concepts, we include a residual concept that captures the remaining variability of $\bar{z}$, which is not explained by the given semantic concepts.

Following [15], we learn a bijective transformation $e(\bar{z})$, which translates the non-interpretable representation $\bar{z}$ invertibly into a factorized representation $(e_i(\bar{z}))_{i=0}^K = e(\bar{z})$, where each factor $e_i \in \mathbb{R}^{N_{e_i}}$ represents one of the given semantic concepts for $i = 1, \ldots, K$, and $e_0 \in \mathbb{R}^{N_{e_0}}$ is the residual concept.

The INN e establishes a one-to-one correspondence between an encoding and different semantic concepts and, conversely, enables semantic modifications to correctly alter the original encoding (see next section). Being an INN, $e(\bar{z})$ and $\bar{z}$ need to have the same dimensionality and we set $N_{e_0} = N_{\bar{z}} - \sum_{i=1}^K N_{e_i}$. We denote the indices of concept i with respect to $e(\bar{z})$ as $\mathcal{I}_i \subset \{1, \ldots, N_{\bar{z}}\}$ such that we can write $e_i = (e(\bar{z})_k)_{k\in\mathcal{I}_i}$.

Deriving a Loss Function for Training the Semantic INN Let e_i be the factor representing some semantic concept, *e.g.* gender, that the contents of two images x^a, x^b share. Then the projection of their encodings $\bar{z}^a, \bar{z}^b$ onto this semantic concept must be similar [15, 30],

$$e_i(\bar{z}^a) \simeq e_i(\bar{z}^b) \quad \text{where } \bar{z}^a = E(x^a), \bar{z}^b = E(x^b). \tag{8}$$

Moreover, to interpret $\bar{z}$ we are interested in the separate contribution of different semantic concepts e_i that explain $\bar{z}$. Hence, we seek a mapping $e(\bullet)$ that strives

to disentangle different concepts,

$$e_i(\bar{z}) \perp e_j(\bar{z}) \quad \forall i \neq j, x \quad \text{where } \bar{z} = E(x). \tag{9}$$

The objectives in Eq. (8), (9) imply a correlation in e_i for pairs $\bar{z}^a$ and $\bar{z}^b$ and no correlation between concepts e_i, e_j for $i \neq j$. This calls for a Gaussian distribution with a covariance matrix that reflects these requirements.

Let $e^a = (e_i^a) = (e_i(E(x^a)))$ and e^b likewise, where x^a, x^b are samples from a training distribution $p(x^a, x^b)$ for the i-th semantic concept. The distribution of pairs e^a and e^b factorizes into a conditional and a marginal,

$$p(e^a, e^b) = p(e^b|e^a)p(e^a) \tag{10}$$

Objective Eq. (9) implies a diagonal covariance for the marginal distribution $p(e^a)$, *i.e.* a standard normal distribution, and Eq. (8) entails a correlation between e_i^a and e_i^b. Therefore, the correlation matrix is $\Sigma^{ab} = \rho\,\mathrm{diag}((\delta_{\mathcal{I}_i}(k))_{k=1}^{N_{\bar{z}}})$. By symmetry, $p(e^b) = p(e^a)$, which gives

$$p(e^b|e^a) = \mathcal{N}(e^b|\Sigma^{ab}e^a, \mathbb{1} - (\Sigma^{ab})^2). \tag{11}$$

Inserting Eq. (11) and a standard normal distribution for $p(e^a)$ into Eq. (10) yields the negative log-likelihood for a pair e^a, e^b. The detailed formulation can be found in the supplementary material.

Given pairs x^a, x^b as training data, another change of variables from $\bar{z}^a = E(x^a)$ to $e^a = e(\bar{z}^a)$ gives the training loss function for e as the negative log-likelihood of $\bar{z}^a, \bar{z}^b$,

$$\begin{aligned}\mathcal{L}(e) = \mathbb{E}_{x^a,x^b}\big[&-\log p(e(E(x^a)), e(E(x^b))) \\ &-\log|\det \nabla e(E(x^a))| - \log|\det \nabla e(E(x^b))|\big]\end{aligned} \tag{12}$$

For simplicity we have derived the loss for a single semantic concept e_i. Simply summing over the losses of different semantic concepts yields their joint loss function and allows us to learn a joint translator e for all of them.

Log-likelihood of Pairs The loss for e in Eq. (12) contains the log-likelihood of pairs e^a, e^b. Inserting Eq. (11) and a standard normal distribution for $p(e^a)$ into Eq. (10) yields

$$-\log p(e^a, e^b) = \frac{1}{2}\left(\sum_{k\in\mathcal{I}_i}\frac{(e_k^b - \rho e_k^a)^2}{1-\rho^2} + \sum_{k\in\mathcal{I}_i^c}(e_k^b)^2 + \sum_{k=1}^{N_{\bar{z}}}(e_k^a)^2\right) + C \tag{13}$$

where $C = C(\rho, N_{\bar{z}})$ is a constant that can be ignored for the optimization process. $\rho \in (0, 1)$ determines the relative importance of loss terms corresponding to the similarity requirement in Eq. (8) and the independence requirement in Eq. (9). We use a fixed value of $\rho = 0.9$ for all experiments.

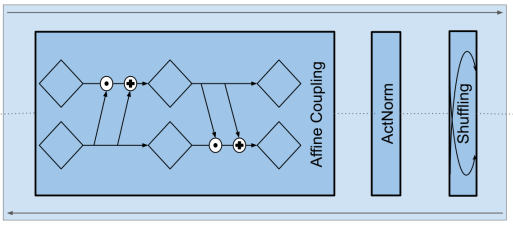

Fig. S1. A single invertible block used to build our invertible neural networks.

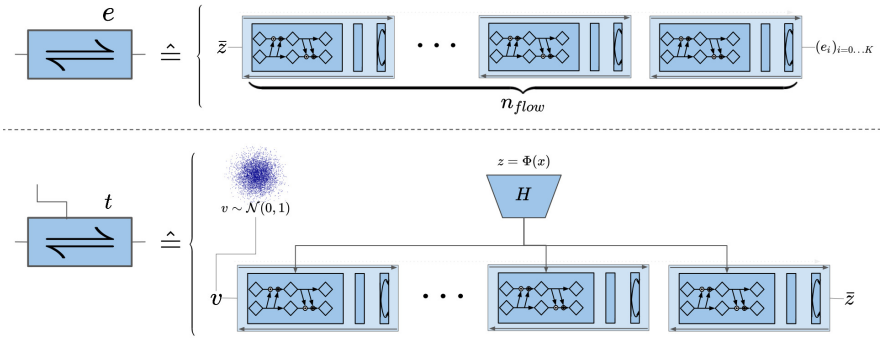

Fig. S2. Architectures of our INN models. *top:* The semantic INN e consists of stacked invertible blocks. *bottom:* The conditional INN t is composed of a embedding module H that downsamples (upsamples if necessary) a given model representation $h = H(z) = H(\Phi(x))$. Subsequently, h is concatenated to the inputs of each block of the invertible model.

Architecture of the Semantic INN In our implementation, e is built by stacking invertible blocks, see Fig. S1, which consist of three invertible layers: coupling blocks [12], actnorm layers [27] and shuffling layers. The final output is split into the factors (e_i), see Fig. S2.

Coupling blocks split their input $x = (x_1, x_2)$ along the channel dimension and use fully connected neural networks s_i and t_i to perform the following computation:

$$\tilde{x}_1 = x_1 \cdot s_1(x_2) + t_1(x_2) \tag{14}$$

$$\tilde{x}_2 = x_2 \cdot s_2(\tilde{x}_1) + t_2(\tilde{x}_1) \tag{15}$$

Actnorm layers consist of learnable shift and scale parameters for each channel, which are initialized to ensure activations with zero mean and unit variance on the first training batch. Shuffling layers use a fixed, randomly initialized permutation to shuffle the channels of its input, which provides a better mixing of channels for subsequent coupling layers.

A.3 Conditional INN for Recovering Invariances of Deep Representations

Architecture of the Conditional INN: We build the conditional invertible neural network t by expanding the semantic model e as follows: Given a model representation z, which is used as the conditioning of the INN, we first calculate its embedding

$$h = H(z) \tag{16}$$

which is subsequently fed into the affine coupling block:

$$\tilde{x}_1 = x_1 \cdot s_1(x_2, h) + t_1(x_2, h) \tag{17}$$

$$\tilde{x}_2 = x_2 \cdot s_2(\tilde{x}_1, h) + t_2(\tilde{x}_1, h) \tag{18}$$

where s_i and t_i are modified from Eq. (15) such that they are capable of processing a concatenated input (x_i, h). The embedding module H is usually a shallow convolutional neural network, used to down-/upsample a given model representation z to a size that the networks s_i and t_i are able to process. This means that t, analogous to e, consists of stacked invertible blocks, where each block is composed of coupling blocks, actnorm layers and shuffling layers, *c.f.* Sec. A.2 and Fig. S1. The complete architectures of both t and e are depicted in Fig. S2. Additionally, Fig. S3 provides a graphical distinction of the training and testing process of t. During training, the autoencoder $D \circ E$ provides a representation of the data that contains both the invariances and the representation of some model w.r.t. the input x. After training of t, the encoder may be discarded and visual decodings and/or semantic interpretations of a model representation z can be obtained by sampling and transforming v as described in Eq. (2).

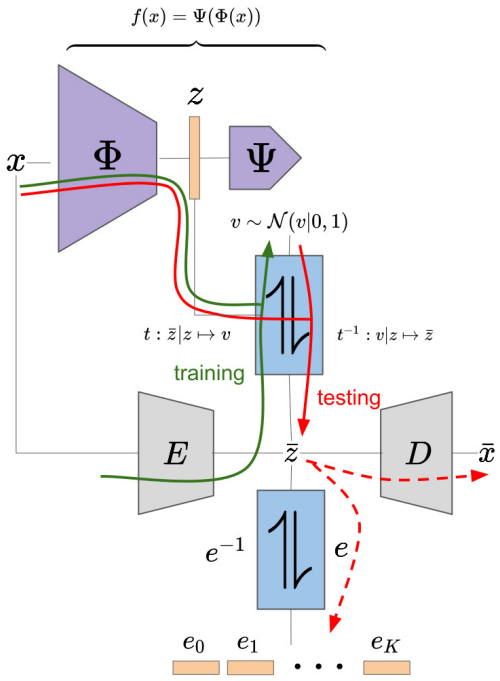

Fig. S3. Graphical distinction of information flow during training and inference. During training of t, the encoder E provides an (approximately complete) data representation, which is used to learn the invariances of a given model's representations z. At inference, the encoder is not neccessarily needed anymore: Given a representation $z = \Phi(x)$, invariances can be sampled from the prior distribution and decoded into data space trough t^{-1}.

B Evaluation Details

An overview of INN hyperparameters for all experiments is provided in Tab. S10.

B.1 Architectures of Interpreted Models

Throughout our experiments, we interpret four different models: *SqueezeNet*, *AlexNet*, *ResNet* and *FaceNet*. Summaries of each of model's architecture are provided in Tab. S11 and Tab. S14. Implementations and pretrained weights of these models are taken from:

- *SqueezeNet (1.1)* `https://pytorch.org/docs/stable/_modules/torchvision/models/squeezenet`
- *ResNet:* `https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html`
- *AlexNet:* `https://pytorch.org/docs/stable/_modules/torchvision/models/alexnet.html`
- *FaceNet:* `https://github.com/timesler/facenet-pytorch`

Table S10. Hyperparameters of INNs for each experiment. n_{flow} denotes the number of invertible blocks within in the model, see Fig. S1. h_w and h_d refer to the width and depth of the fully connected subnetworks s_i and t_i.

Experiment	INN input dim.		n_{flow}	h_w	h_d
Comparison Sec.4.1	t	128	20	1024	2
Understanding Models: FaceNet Sec. 4.2	t	128	20	512	2
Understanding Models: FaceNet Sec. 4.2	e	128	12	512	2
Data Effects: Adversarial Attack Sec. 4.3	t	128	20	1024	2
Data Effects: Texture Bias Sec. 4.3	t	268	20	1024	2
Data Effects: Domain Shift Sec. B.6	t	128	20	1024	2
Modifications: FaceNet & CelebA Sec. 4.4	e	128	12	512	2

B.2 Explained Variance

To quantify the amount of invariances and semantic concepts, we use the fraction of the total variance explained by invariances (Fig. 5) and the fraction of the variance of a semantic concept explained by the model representation (Fig. 4).

Using the INN t, we can consider $\bar{z} = t^{-1}(v|z)$ as a function of v and z. The total variance of $\bar{z}$ is then obtained by sampling v, via its prior which is a standard normal distribution, and z, via $z = \Phi(x)$ with $x \sim p_{\text{valid}}(x)$ sampled from a validation set. We compare this total variance to the average variance obtained when sampling v for a given z to obtain the fraction of the total variance explained by invariances:

$$\mathbb{E}_{x' \sim p_{\text{valid}}(x')} \left[\frac{\text{Var}_{v \sim \mathcal{N}(v|0,\mathbb{1})} \; t^{-1}(v|\Phi(x'))}{\text{Var}_{\substack{x \sim p_{\text{valid}}(x) \\ v \sim \mathcal{N}(v|0,\mathbb{1})}} \; t^{-1}(v|\Phi(x))} \right] \tag{19}$$

In combination with the INN e, which transform $\bar{z}$ to semantically meaningful factors, we can analyze the semantic content of a model representation z. To analyze how much of a semantic concept represented by factor e_i is captured by z, we use e to transform $\bar{z}$ into e_i and measure its variance. To measure how much the semantic concept is explained by z, we simply swap the roles of z and v in Eq. (19), to obtain

$$\mathbb{E}_{v' \sim \mathcal{N}(v'|0,\mathbb{1})} \left[\frac{\text{Var}_{x \sim p_{\text{valid}}(x)} \; e(t^{-1}(v'|\Phi(x)))_i}{\text{Var}_{\substack{v \sim \mathcal{N}(v|0,\mathbb{1}) \\ x \sim p_{\text{valid}}(x)}} \; e(t^{-1}(v|\Phi(x)))_i} \right] \tag{20}$$

Fig. 5 reports Eq. (19) and its standard error when evaluated via 10k samples, and Fig. 4 reports Eq. (20) and its standard error when evaluated via 10k samples.

Table S11. High-level architectures of FaceNet and ResNet, depicted as `pytorch`-modules. Layers investigated in our experiments are marked in bold. Spatial sizes are provided as a visual aid and vary from model to model in our experiments. If not stated otherwise, we always extract from the *last* layer in a series of blocks (*e.g.* in Tab. S13: 23× **BottleNeck down** $\rightarrow \mathbb{R}^{8\times8\times1024}$ refers to the last module in the series of 23 blocks.)

Table S12. *FaceNet:* Implementations of layers `Mixed, Block35, Block17, Block8` can be found at `https://github.com/timesler/facenet-pytorch`. In l.4, the representation from the **2nd** convolutional layer is extraced. Furthermore, BN refers to batch normalization.

RGB image $x \in \mathbb{R}^{128\times128\times3}$
3× Conv, BN, ReLU $\rightarrow \mathbb{R}^{61\times61\times64}$
MaxPool $\rightarrow \mathbb{R}^{30\times30\times64}$
3× **Conv**, BN, ReLU $\rightarrow \mathbb{R}^{13\times13\times256}$
5× Block35 $\rightarrow \mathbb{R}^{13\times13\times256}$
Mixed down $\rightarrow \mathbb{R}^{6\times6\times896}$
10× Block17 $\rightarrow \mathbb{R}^{6\times6\times896}$
Mixed down $\rightarrow \mathbb{R}^{2\times2\times1792}$
5× Block8 $\rightarrow \mathbb{R}^{2\times2\times1792}$
AdaAvgPool $\rightarrow \mathbb{R}^{1\times1\times1792}$
Dropout, Linear, BN $\rightarrow \mathbb{R}^{512}$
identity embedding $\rightarrow \mathbb{R}^{512}$

Table S13. *ResNet-101:* See `https://pytorch.org/docs/stable/torchvision/models.html` for details on other variants of ResNet.

RGB image $x \in \mathbb{R}^{128\times128\times3}$
Conv down $\rightarrow \mathbb{R}^{64\times64\times64}$
Norm, ReLU, **MaxPool** $\rightarrow \mathbb{R}^{32\times32\times64}$
3× BottleNeck $\rightarrow \mathbb{R}^{32\times32\times256}$
4× BottleNeck down $\rightarrow \mathbb{R}^{16\times16\times512}$
23× **BottleNeck down** $\rightarrow \mathbb{R}^{8\times8\times1024}$
3× BottleNeck down $\rightarrow \mathbb{R}^{4\times4\times2048}$
AvgPool, FC
output $\rightarrow \mathbb{R}^{1000}$

B.3 Comparison to Existing Visualization Methods

In Sec. 4.1, we compare to existing layer inversion methods that aim to reconstruct an input x from its representation $z = \Phi(x)$. Both our method and D&B's [13] method were trained on the *Animals* dataset, which consists of a mixture of all carnivorous mammal animal classes from *ImageNet* and all animals from the *Animals with Attributes 2* [60] dataset. Hyperparameters of our autoencoder model can be found in Tab. S7. The decoder in [13] was re-implemented based on our decoder shown in Tab. S9, where we set the latent dimension to 4096 to avoid introduction of an artificial bottleneck and allow for a fair comparison. Both methods were trained by minimizing the image metric described in Sec.A.1 and Tab. S4, where no Kullback-Leibler divergence term was used for

Table S14. High-level architectures of SquuezeNet and AlexNet, depicted as `pytorch`-modules. *C.f.* Tab.S11 for further details.

Table S15. *SqueezeNet.* We extract the penultimate *Fire* block for interpretation in Sec. 4.2.

RGB image $x \in \mathbb{R}^{128\times128\times3}$
Conv, ReLU, MaxPool $\rightarrow \mathbb{R}^{31\times31\times64}$
2× Fire $\rightarrow \mathbb{R}^{31\times31\times128}$
MaxPool $\rightarrow \mathbb{R}^{15\times15\times128}$
2× Fire $\rightarrow \mathbb{R}^{15\times15\times256}$
MaxPool $\rightarrow \mathbb{R}^{7\times7\times256}$
4× **Fire** $\rightarrow \mathbb{R}^{7\times7\times512}$
Dropout, Conv, ReLU $\rightarrow \mathbb{R}^{7\times7\times1000}$
AdaAvgPool $\rightarrow \mathbb{R}^{7\times7\times1000}$
output $\rightarrow \mathbb{R}^{1000}$

Table S16. *AlexNet:* The first convolution uses kernel size 11.

RGB image $x \in \mathbb{R}^{128\times128\times3}$
Conv, ReLU, MaxPool $\rightarrow \mathbb{R}^{15\times15\times64}$
Conv, ReLU, MaxPool $\rightarrow \mathbb{R}^{7\times7\times192}$
Conv, ReLU $\rightarrow \mathbb{R}^{7\times7\times384}$
2× **Conv**, ReLU $\rightarrow \mathbb{R}^{7\times7\times256}$
MaxPool $\rightarrow \mathbb{R}^{3\times3\times256}$
AdaAvgPool, Flatten $\rightarrow \mathbb{R}^{9216}$
Dropout, **Linear**, ReLU $\rightarrow \mathbb{R}^{4096}$
Dropout, **Linear**, ReLU $\rightarrow \mathbb{R}^{4096}$
Linear $\rightarrow \mathbb{R}^{1000}$

D&B's method. Images from [39] are taken from their publication. Additional visual comparisons can be found in Fig. S4, S5, S6.

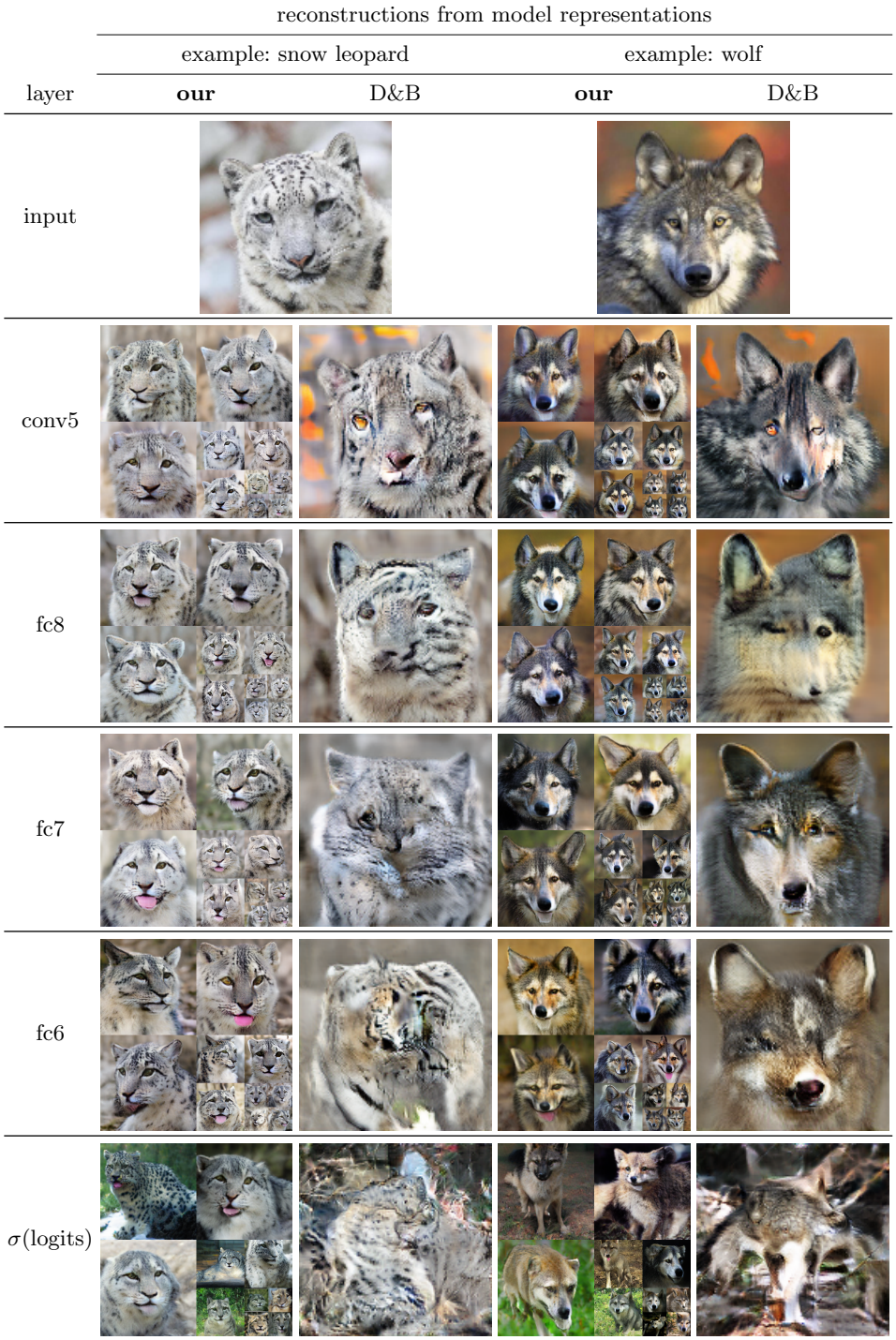

Fig. S4. Additional examples for layerwise reconstructions from model representations $z = \Phi(x)$ with our method and [13] (D&B). We show 10 samples per layer representation obtained with our generative approach. Here, σ denotes the softmax function, *i.e.* reconstructions are obtained from class probabilities provided by the model. We provide a comparison of equally sized images in Fig. S6 and Fig. S5.

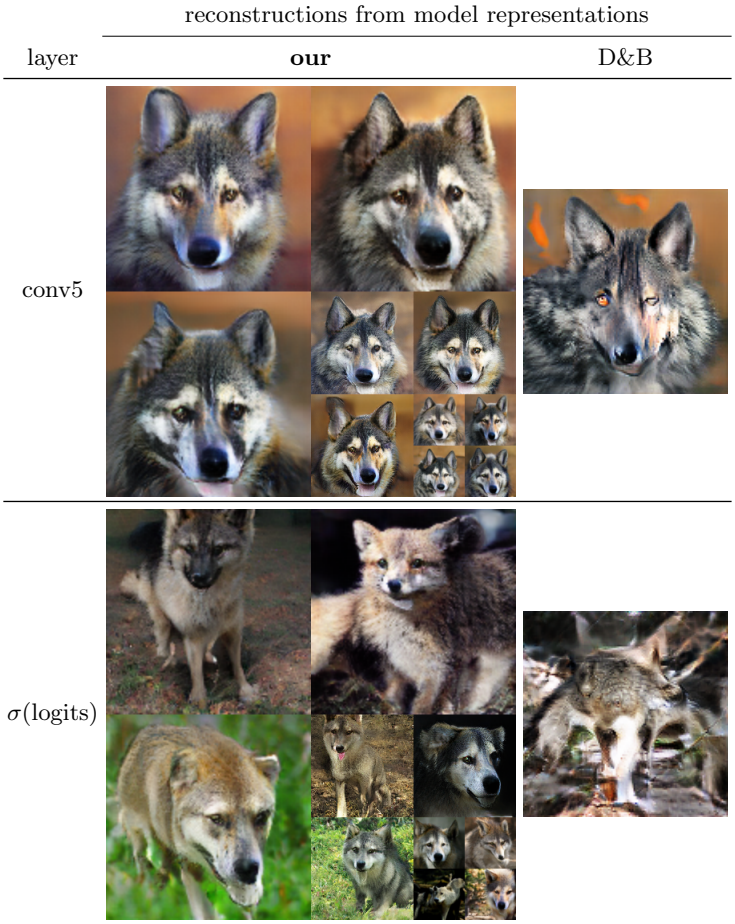

Fig. S5. Zooming into representation conditional samples for example *wolf*. To verify that our samples are outperforming those of [13] in visual quality, we repeat row 2 (conv5) and row 6 (σ(logits)) of Fig. S4 with scaled images. Here, σ denotes the softmax function.

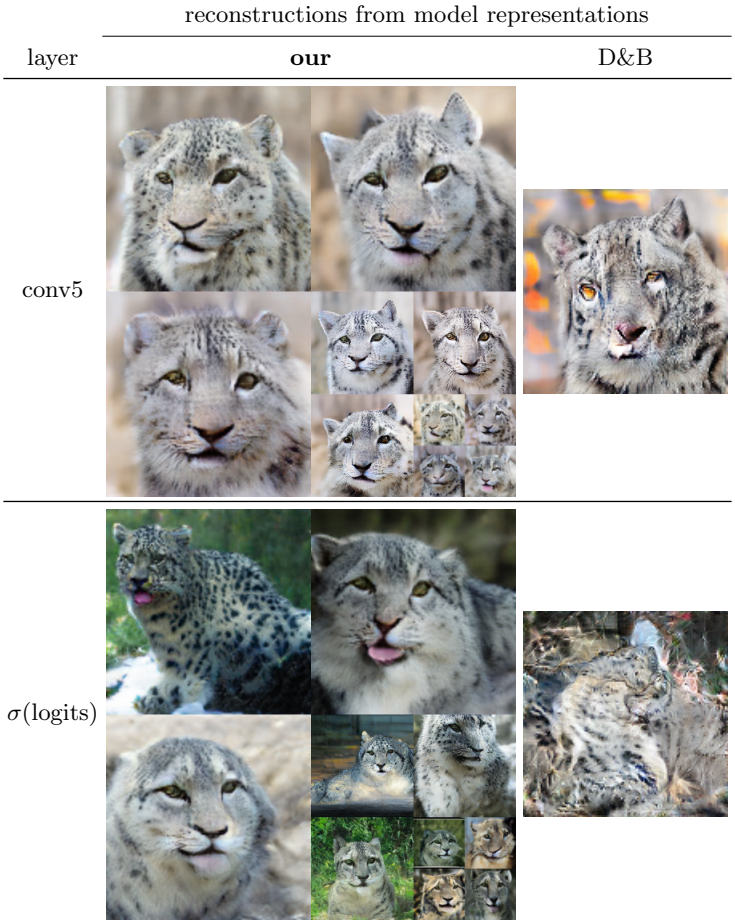

Fig. S6. Zooming into representation conditional samples for example *snow leopard*. To verify that our samples are outperforming those of [13] in visual quality, we repeat row 2 (conv5) and row 6 (σ(logits)) of Fig. S4 with scaled images. Here, σ denotes the softmax function.

B.4 Relevance of Factors

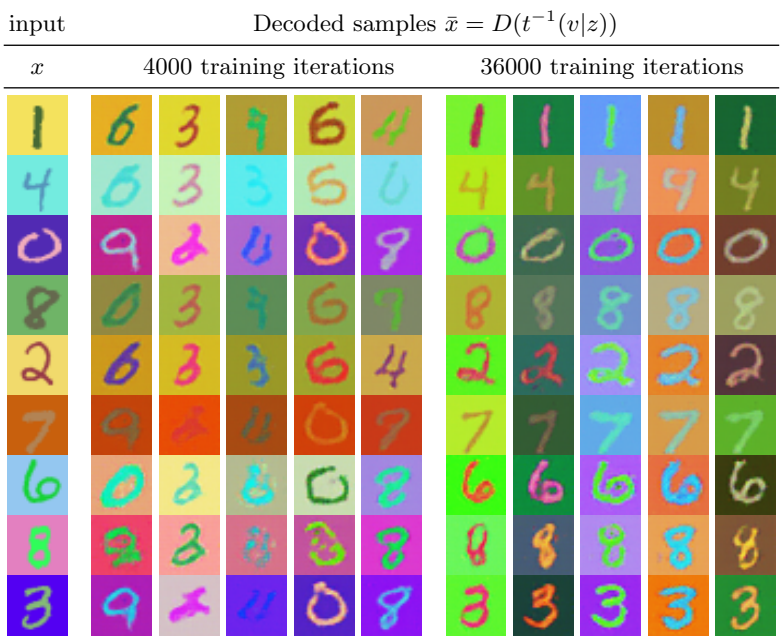

Fig. S7. Additional z conditional samples after 4k and 36k training steps, as in Fig. 4. Each row is conditioned on $z = \Phi(x)$ and each column is conditioned on a $v \sim \mathcal{N}(v|0, \mathbb{1})$. At 4k (resp. 36k) iterations, z explains 2.57% (resp. 36.08%) of the variance in the digit factor. Thus, the digit class of samples obtained at 4k iterations change with the sampled invariances across columns, while it stays the same at 36k iterations. Conversely, at 4k (resp. 36k) iterations, z explains 38.44% (resp. 2.76%) of the variance in the background color factor. Thus, the background color of samples obtained at 4k iterations change with the sampled representation $z = \Phi(x)$ across rows, while it stays the same at 38k iterations.

In Sec. 4.2, we trained *SqueezeNet* for digit classification on *ColorMNIST*, which consists of *MNIST* images with randomly choosen fore- and background colors. In addition, we trained the autoencoder of Tab. S1 on *ColorMNIST* and the INN e to obtain the following factors

- e_1 representing the digit class defined by pairs of images showing the same digit in different styles and colors,
- e_2 representing the foreground color defined by pairs of images showing the same foreground color on different digits and backgrounds,
- e_3 representing the background color defined by pairs of images showing the same background color for differently colored digits.

Finally, we trained the INN t for 20 different checkpoints of *SqueezeNet* obtained between training steps zero and 40k, to obtain the stochastic mapping

from z, the penultimate *Fire* layer of *SqueezeNet*, to the semantic factors (e_i). Fig. 4 plots Eq. (20) against the training step, with shaded areas representing the standard error obtained with 10k samples.

At step zero, *i.e.* for a randomly initialized *SqueezeNet*, we observe that the representation z mostly contains the background color and, to a lesser degree, the foreground color. This observation is consistent with the fact that color information is directly encoded in the pixel representation of the image and that there are more background pixels than foreground pixels. In contrast, information about the digit class is not directly encoded in pixel values and requires learning. As the network starts to learn between steps 10k and 15k, we indeed observe a drastic change in the semantic content of z, which becomes invariant to color information and sensitive to digit class information. Note that the network could also learn to retain color information while seperating digit classes in the last classification layer, but our results demonstrate that the network learns to abstract away task-irrevant information before that.

We show additional z conditional samples, both before and after learning, in Fig. S7.

B.5 Modifying Representations

Training Details: In Sec. 4.4 we trained the autoencoder of Tab. S7 on *CelebA* at resolution 128×128. Using the attribute labels provided for this dataset, we trained an INN e for the semantic factors

- e_1 representing hair color, defined by pairs with the same *Black_Hair* attribute.
- e_2 representing glasses, defined by pairs with the same *Eyeglasses* attribute.
- e_3 representing gender, defined by pairs with the same *Male* attribute.
- e_4 representing beard, defined by pairs with the same *No_Beard* attribute.
- e_5 representing age, defined by pairs with the same *Young* attribute.
- e_6 representing smiling, defined by pairs with the same *Smiling* attribute.

Additional Results and Comparisons We provide a larger version of Fig. 7 with more examples in Fig. S8 and Fig. S9. While our approach aims to provide semantic understanding of representations learned by models, the invertibility of e together with the decoder D enables semantic image editing. To evaluate our approach on this task, we compare it to *StarGAN*[2] [8], a specialized approach for attribute modifications of face images. Our approach consistently outperforms [8] across all semantic attributes in terms of the quality of modified images, which is quantified by FID scores [23] in Fig. S9. Moreover, we observe some particular qualitative differences between our method and [8]: Changing factors with our approach produces more coherent changes, *i.e.* changes in gender cause changes in hair length (for all examples in Fig. S8), changes to an older age cause thin, white hairs (*e.g.* examples 1, 2, 6 in Fig. S8), and changes in the beard factor have no effect on female faces (*e.g.* examples 2, 3, 5, 6 in Fig. S8), suggesting that our model has learned the correct causal structure (as present in the data) where beard is caused by gender and not the other way around. In contrast, [8] produces very localized, pixelwise changes without taking the global structure into account. While such a behavior might be desired for some specialized applications, it generally leads to unnatural results, *e.g.* when changing gender, beard and age in example 2 or gender and beard in example 3 of Fig. S8.

[2] We used the author's official implementation available at `https://github.com/yunjey/stargan`

Fig. S8. Additional examples corresponding to Fig. 7. In each column, we replace a semantic factor $e_i(E(x))$ by e_i^*, which is obtained from another, randomly chosen, image that differs in the corresponding attribute (see Sec. B.5). Subsequently we decode a semantically modified image using the invertibility of e to obtain $\bar{x}^* = D(e^{-1}((e_i^*)))$. The results of *StarGAN* [8] are obtained by negating the binary value for the column's attribute. FID scores in Fig. S9.

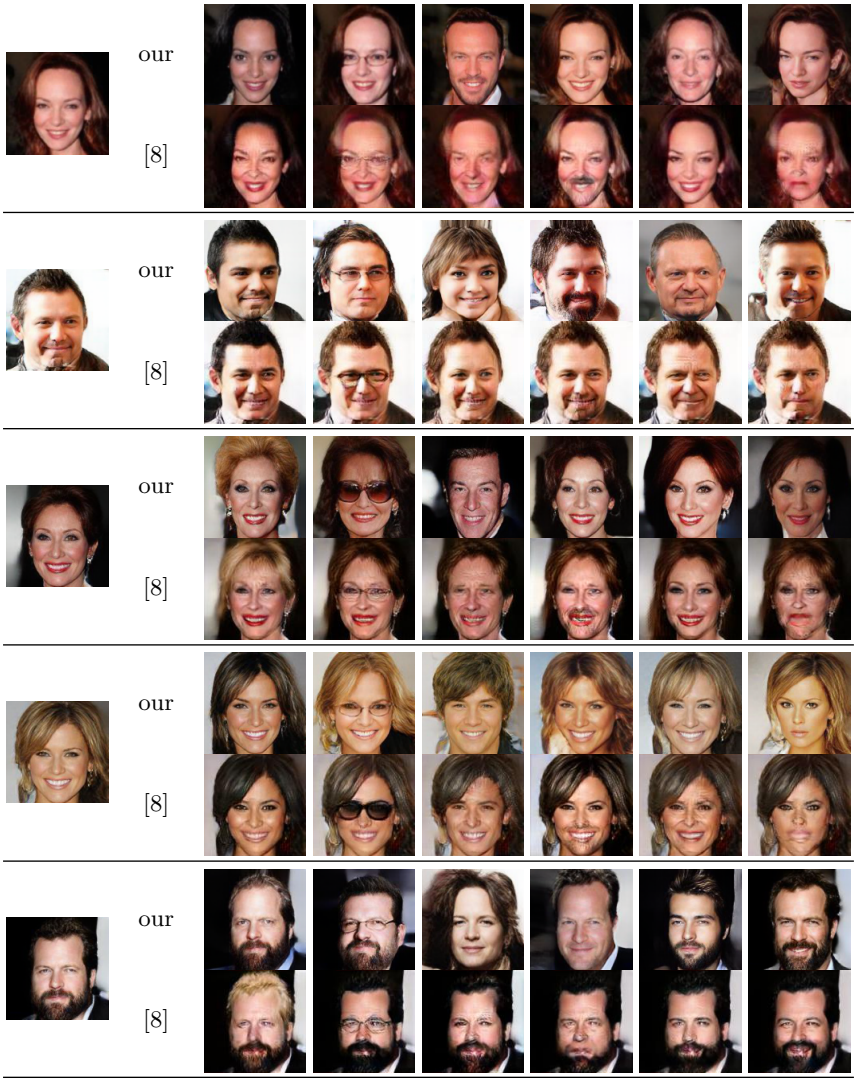

Fig. S9. Additional examples as in Fig. S8. Moreover, the last row contains FID scores [23] of semantically modified images obtained by our approach and [8], which shows that our approach consistently outperforms [8].

B.6 Effects of Data Shifts - Additional Results

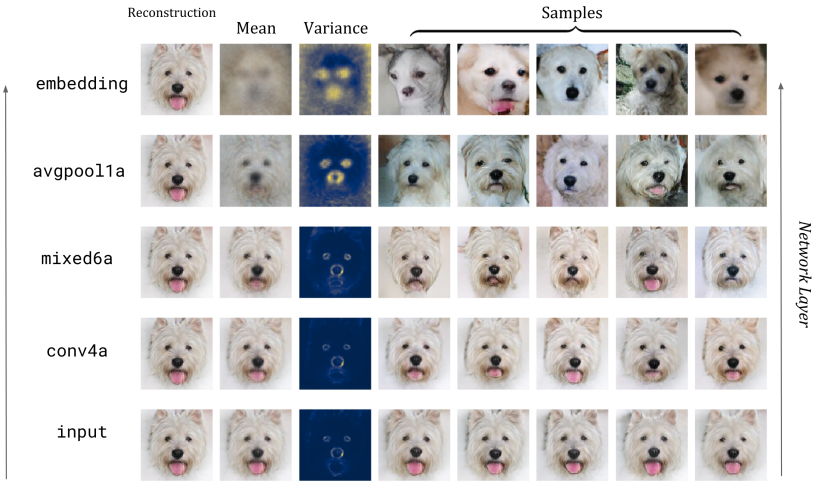

Fig. S10. Shifting domains: Human faces to animal faces evaluated with a fixed *FaceNet*. The evaluation procedure is similar to the method described in Fig. 3. Although never trained on data consisting of something else than human faces, *FaceNet* is able to capture the "identity" of the input to a certain degree. Information about appearance is approximately preserved until the last layer, *i.e.* the final identity embedding.

Data Shift from Humans to Animals As an extension of Sec. 4.3, we run an experiment on FaceNet and condition the invariance recovering model t on five different representations of the model (see Tab. S12) by training t on *AnimalFaces* instead of *CelebA* and an autoencoder which is trained on both *AnimalFaces* and *CelebA*, *c.f.* Tab. S7 for details. Furthermore, note that FaceNet is not re-trained on the new data and fixed during training, *c.f.* Fig. S3.

Fig. S10 depicts the visualized representations and corresponding learned invariances accross several layers of FaceNet. Evidently, even deep representations of the off-domain input image may be visualized, at least as deep as the penultimate layer (**AdaAvgPool**). Another interesting result is that FaceNet seems to conserve class identity of the input to some degree: The appearance of samples generated by conditioning on model representations is similar, to some extend even for the last layer (**identity embedding**). This suggests that the model is able to generalize to a surprisingly large margin of data, given the input images show some kind of symmetry and perceptual similarity to human faces.

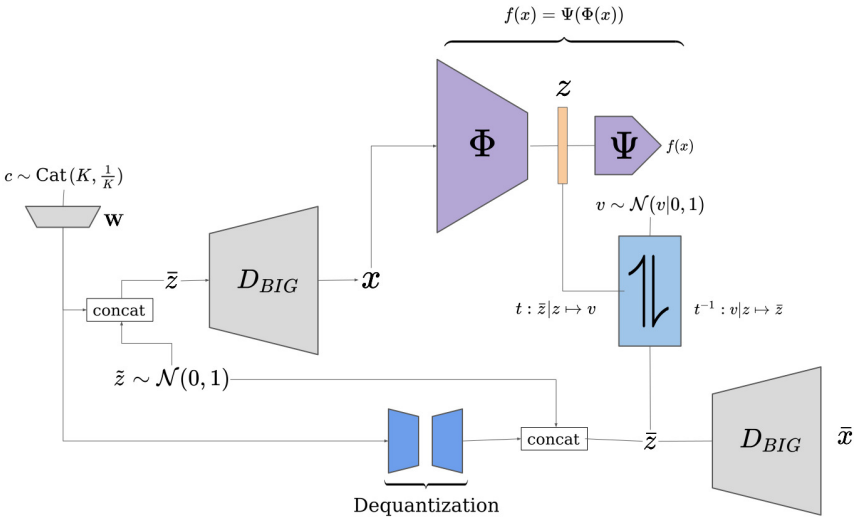

Fig. S11. Applying our approach to *BigGAN* [6]. We directly train t on latent codes of the generator model, utilizing a simple variational autoencoder model for dequantization of discrete classes c. See Sec.B.6 for technical details.

Verifying the Texture-Bias Hypothesis In Section 4.3 we trained the INN t conditioned on representations of ResNet-50 from the penultimate layer (*i.e.* extracted before the final classification layer, *c.f.* Tab. S13) with the goal of validating the *texture-bias hypothesis* from [18]. In their work, [18] showed that typical convolutional neural classification networks are biased towards texture when being trained on ImageNet. They proposed that this bias can be removed by training the CNNs on a *stylized* version of ImageNet instead.

Following [49], we gained access to the dataset and a powerful decoder by relying on a synthetic version of ImageNet, provided through the pre-trained generator of BigGAN [6].[3] Thus, with Eq. (5) in mind, we identify the concatenated vector $(\tilde{z}, \mathbf{W}c))$ as $\bar{z}$. Here, $\mathbb{R}^{140} \ni \tilde{z} \sim \mathcal{N}(0, \mathbb{1})$ is sampled from a multivariate normal distribution and $c \in \{0,1\}^K$ is a one-hot vector representing one of the $K = 1000$ ImageNet classes. $\mathbf{W}$ maps the one-hot class representation c to the space of real numbers, *i.e.* $\mathbf{W}c = h \in \mathbb{R}^{128}$. Note that $\mathbf{W}$ is part of the BigGAN generator D_{BIG} and is thus also pre-trained. See Fig. S11 for a visual summaray of the application of our approach to BigGAN. To avoid overfitting t on a single dimension of $\bar{z}$, the vector h is passed trough a small, fully connected variational autoencoder before being concatenated with $\tilde{z}$ as $\bar{z} = (\tilde{z}, h)$. The architecture of this VAE is depicted in Tab. S17. Utilizing this approach can be interpreted as a variant of deep dequantization. Equipped with a dequantized version of

[3] We used a pretrained generator available at `https://github.com/LoreGoetschalckx/GANalyze`

Table S17. Architecture of the VAE used for dequantization when training solely on synthetic BigGAN data. Here, a slope parameter of $\alpha = 0.01$ is used in Leaky ReLU.

Embedding $h \in \mathbb{R}^{128}$
(FC, LReLU) $\rightarrow \mathbb{R}^{4096}$
$2\times$ (FC, LReLU) $\rightarrow \mathbb{R}^{4096}$
μ, σ^2: for each:
(FC, LReLU) $\rightarrow \mathbb{R}^{128}$
(FC, LReLU) $\rightarrow \mathbb{R}^{4096}$
$2\times$ (FC, LReLU) $\rightarrow \mathbb{R}^{4096}$
(FC, LReLU) $\rightarrow \mathbb{R}^{128}$
$h \in \mathbb{R}^{128} \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$
(FC, LReLU) $\rightarrow \mathbb{R}^{4096}$
$3\times$ (FC, LReLU) $\rightarrow \mathbb{R}^{4096}$
(FC, LReLU) $\rightarrow \mathbb{R}^{128}$

$\bar{z} = (\tilde{z}, h)$ and corresponding images $x = D_{BIG}(\bar{z})$, we trained t as described in Sec. 3.1.

Additional samples conditioned on representations of (i) a ResNet-50 trained on standard ImageNet and (ii) a ResNet-50 trained on the stylized version of ImageNet are provided in Fig. S12. These results further confirm the texture&shape-bias of (i) and the reverse behavior for (ii). Line 7 and 8 explicitly show that a texture-agnostic classifier can be used to create new content based on input sketches or cartoons.

Furthermore, note that both models perform reasonably well on the domain of natural images, *c.f.* line 1-2 of Tab. S12.

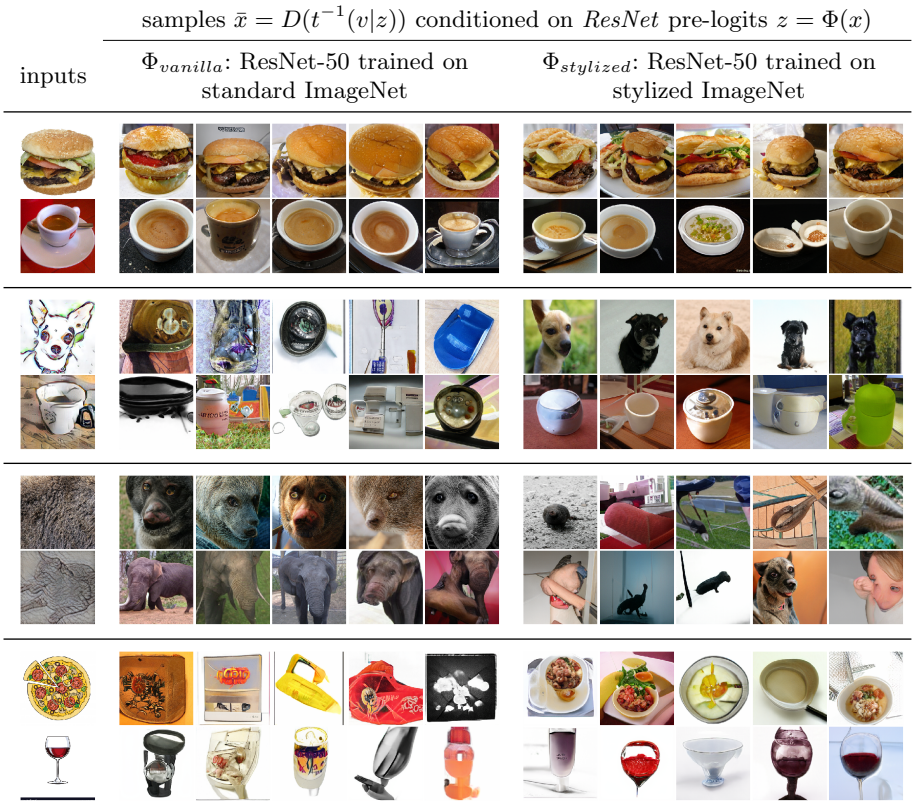

Fig. S12. *Texture bias*: Additional examples for representation-conditional samples of two variants of *ResNet-50*, one trained on standard *ImageNet*, the other on a stylized version of *ImageNet*. See also Tab. 6.

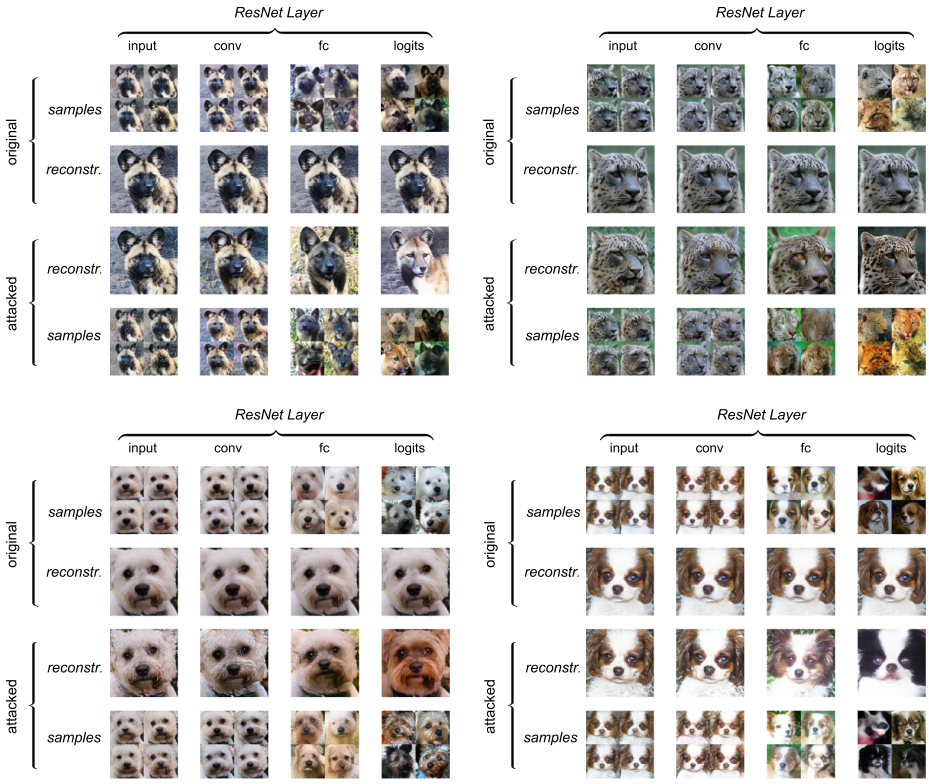

Fig. S13. More visualizations of adversarial attacks as in Fig. 5. Predictions of original *vs*. attacked version of the input image for all depicted examples: *top left:* 'Lycaon pictus' *vs*. 'Cuon alpinus'; *top right:* 'Snow Leopard' *vs*. 'Leopard'; *bottom left:* 'West Highland white Terrier' *vs*. 'Yorkshire Terrier'; *bottom right:* 'Blenheim Spaniel' *vs*. 'Japanese Spaniel'.

References

1. Achille, A., Soatto, S.: Emergence of invariance and disentanglement in deep representations. The Journal of Machine Learning Research **19**(1), 1947–1980 (2018)
2. Ardizzone, L., Kruse, J., Wirkert, S., Rahner, D., Pellegrini, E.W., Klessen, R.S., Maier-Hein, L., Rother, C., Köthe, U.: Analyzing inverse problems with invertible neural networks (2018)
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015)
4. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jul 2017). https://doi.org/10.1109/cvpr.2017.354, http://dx.doi.org/10.1109/CVPR.2017.354
5. Bau, D., Zhu, J.Y., Strobelt, H., Zhou, B., Tenenbaum, J.B., Freeman, W.T., Torralba, A.: Gan dissection: Visualizing and understanding generative adversarial networks (2018)
6. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
7. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: Vggface2: A dataset for recognising faces across pose and age. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). pp. 67–74. IEEE (2018)
8. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018)
9. Commission, E.: On artificial intelligence - a european approach to excellence and trust. Tech. rep. (2020 (accessed February, 2020)), https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2020:65:FIN
10. Dai, B., Wipf, D.: Diagnosing and enhancing vae models (2019)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
12. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp (2016)
13. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks (2016)
14. Esser, P., Haux, J., Ommer, B.: Unsupervised robust disentangling of latent characteristics for image synthesis. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) (Oct 2019). https://doi.org/10.1109/iccv.2019.00279, http://dx.doi.org/10.1109/ICCV.2019.00279
15. Esser, P., Rombach, R., Ommer, B.: A disentangling invertible interpretation network for explaining latent representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9223–9232 (2020)
16. Fong, R., Vedaldi, A.: Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (Jun 2018). https://doi.org/10.1109/cvpr.2018.00910, http://dx.doi.org/10.1109/CVPR.2018.00910

17. Fong, R., Vedaldi, A.: Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8730–8738 (2018)
18. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231 (2018)
19. Goetschalckx, L., Andonian, A., Oliva, A., Isola, P.: Ganalyze: Toward visual definitions of cognitive image properties. arXiv preprint arXiv:1906.10112 (2019)
20. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
21. Goodman, B., Flaxman, S.: European union regulations on algorithmic decision-making and a "right to explanation". AI Magazine **38**(3), 50–57 (Oct 2017). https://doi.org/10.1609/aimag.v38i3.2741, `http://dx.doi.org/10.1609/aimag.v38i3.2741`
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
23. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium (2017)
24. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
25. Jacobsen, J.H., Behrmann, J., Zemel, R., Bethge, M.: Excessive invariance causes adversarial vulnerability (2018)
26. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
27. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: Advances in Neural Information Processing Systems. pp. 10215–10224 (2018)
28. Kotovenko, D., Sanakoyeu, A., Lang, S., Ommer, B.: Content and style disentanglement for artistic style transfer. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 4421–4430 (2019)
29. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
30. Kulkarni, T.D., Whitney, W., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network (2015)
31. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
32. LeCun, Y.: Learning invariant feature hierarchies. In: European conference on computer vision. pp. 496–505. Springer (2012)
33. Li, Y., Singh, K.K., Ojha, U., Lee, Y.J.: Mixnmatch: Multifactor disentanglement and encoding for conditional image generation (2019)
34. Lipton, Z.C.: The mythos of model interpretability (2016)
35. Liu, M.Y., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J., Kautz, J.: Few-shot unsupervised image-to-image translation. arXiv preprint arXiv:1905.01723 (2019)
36. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV) (December 2015)

37. Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., Bachem, O.: Challenging common assumptions in the unsupervised learning of disentangled representations (2018)
38. Lorenz, D., Bereska, L., Milbich, T., Ommer, B.: Unsupervised part-based disentangling of object shape and appearance. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 10947–10956 (2019)
39. Mahendran, A., Vedaldi, A.: Visualizing deep convolutional neural networks using natural pre-images. International Journal of Computer Vision **120**(3), 233–255 (2016)
40. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence **267**, 1–38 (2019)
41. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition **65**, 211–222 (2017)
42. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digital Signal Processing **73**, 1–15 (2018)
43. Mordvintsev, A., Olah, C., Tyka, M.: Inceptionism: Going deeper into neural networks (2015)
44. Nash, C., Kushman, N., Williams, C.K.: Inverting supervised representations with autoregressive neural density models. In: The 22nd International Conference on Artificial Intelligence and Statistics. pp. 1620–1629 (2019)
45. Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J.: Synthesizing the preferred inputs for neurons in neural networks via deep generator networks (2016)
46. Plumb, G., Al-Shedivat, M., Xing, E., Talwalkar, A.: Regularizing black-box models for improved interpretability (2019)
47. Redlich, A.N.: Supervised factorial learning. Neural Computation **5**(5), 750–766 (1993). https://doi.org/10.1162/neco.1993.5.5.750
48. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32. pp. II–1278. JMLR. org (2014)
49. Rombach, R., Esser, P., Ommer, B.: Network fusion for content creation with conditional inns (2020)
50. Samek, W., Wiegand, T., Müller, K.R.: Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296 (2017)
51. Santurkar, S., Tsipras, D., Tran, B., Ilyas, A., Engstrom, L., Madry, A.: Image synthesis with a single (robust) classifier (2019)
52. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 815–823 (2015)
53. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision **128**(2), 336–359 (Oct 2019). https://doi.org/10.1007/s11263-019-01228-7, http://dx.doi.org/10.1007/s11263-019-01228-7
54. Shocher, A., Gandelsman, Y., Mosseri, I., Yarom, M., Irani, M., Freeman, W.T., Dekel, T.: Semantic pyramid for image generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)

55. Simon, M., Rodner, E.: Neural activation constellations: Unsupervised part model discovery with convolutional networks. 2015 IEEE International Conference on Computer Vision (ICCV) (Dec 2015). https://doi.org/10.1109/iccv.2015.136, http://dx.doi.org/10.1109/ICCV.2015.136
56. Simon, M., Rodner, E., Denzler, J.: Part detector discovery in deep convolutional neural networks. ArXiv **abs/1411.3159** (2014)
57. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
58. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2013)
59. Upchurch, P., Gardner, J., Pleiss, G., Pless, R., Snavely, N., Bala, K., Weinberger, K.: Deep feature interpolation for image content changes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7064–7073 (2017)
60. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE transactions on pattern analysis and machine intelligence **41**(9), 2251–2265 (2018)
61. Xiao, Z., Yan, Q., Amit, Y.: Generative latent flow (2019)
62. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization (2015)
63. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. Lecture Notes in Computer Science p. 818–833 (2014)
64. Zhang, Q., Nian Wu, Y., Zhu, S.C.: Interpretable convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8827–8836 (2018)
65. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
66. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Object detectors emerge in deep scene cnns (2014)
67. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2016). https://doi.org/10.1109/cvpr.2016.319, http://dx.doi.org/10.1109/CVPR.2016.319