Supplementary Material: Linking in Style: Understanding learned features in deep learning models

Maren H. Wehrheim[1,2], Pamela Osuna-Vargas[1,2], and Matthias Kaschube[1,2]

[1] Frankfurt Institute for Advanced Studies (FIAS), Frankfurt, Germany
[2] Department of Computer Science and Mathematics, Goethe University Frankfurt, Frankfurt, Germany
{wehrheim, osuna, kaschube}@fias.uni-frankfurt.de

S1 Code availability

The code can be accessed at `https://github.com/kaschube-lab/LinkingInStyle.git` including code and example images to visualize individual unit representations.

S2 Linking networks

Alternatively to the linear regression described in Sec. 3.1, we also train two fully connected neural networks (FC) to find a mapping between the R-space and the W-space. The FC networks contain one hidden layer with a ReLu non-linearity. We train two models with different loss functions. The first FC optimizes the mean squared error between the original w and the prediction w^*:

$$\mathcal{L}_1 = ||w^* - w||_2^2 \,. \tag{1}$$

The second nonlinear linking model incorporates information about the main variance components of activations in both spaces R and W. Previous work [2] has shown that the principal components of the W-space capture most relevant information about the variance in the features of the generated images. To make explicit use of this information, we train a second FC model that in addition to $\mathcal{L}_1$ also optimizes the similarity of the projections onto principal components U in W with their predictions via the linking network:

$$\mathcal{L}_2 = \mathcal{L}_1 + \lambda||(w^T U)^* - w^T U||_2^2 \,. \tag{2}$$

To this end, we first compute the PCA in W from the complete training set. Each sample is then projected onto the first 10 PCs U and mapped through the GAN, classifier, and linking network. We refer to the linear regression linking network as LR, to the FC network trained with $\mathcal{L}_1$ as $\mathrm{FC}_{\mathcal{L}_1}$, and to the FC network trained with $\mathcal{L}_2$ as $\mathrm{FC}_{\mathcal{L}_2}$. We observe a similar performance across all of these models (Fig. S1), indicating that even a simple linear model is sufficient to establish an accurate mapping between R and W.

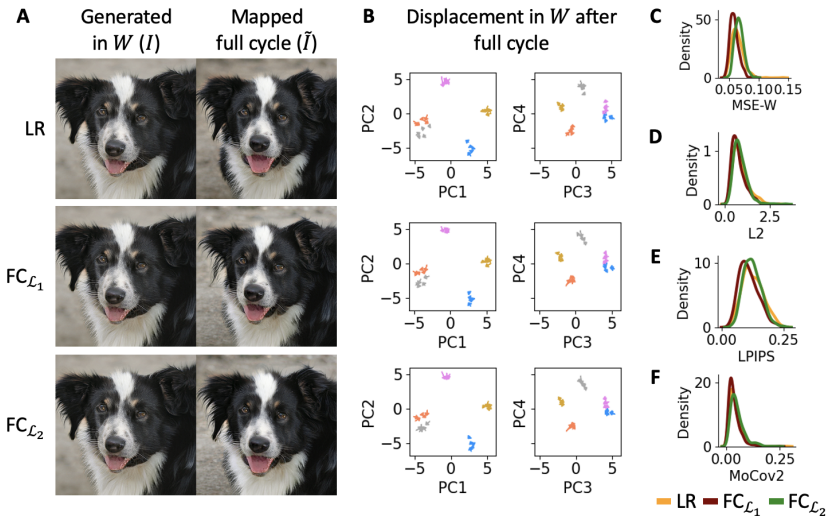

Fig. S1: Linear regression and more complex linking networks achieve similar mapping results. A) High similarity between generated images (I) and full-cycle mapped images ($\tilde{I}$) across the three types of linking networks studied for ResNet-50. **B**) All three linking networks map the images into the W space with only small offsets in the first PCs, indicating high similarity between the largest axes of variability. Dog classes are color-coded. **C -F**) We compute the performance based on 1,000 different initializations (i.e., randomly chosen ws and their generated images). We observe similar performance across all linking networks. The simple linear regression model (yellow) performs similarly to the non-linear models (red and green) based on the mean squared error (MSE) in W (C), as well as on several metrics in the image domain (D: L2; E: LPIPS; F: MoCov2).

S3 Extensions to other classifiers

Additionally to ResNet-50 reported in Sec. 4.2, we also analyze the performance of the linking network for three other classifiers (AlexNet [3], DenseNet-201 [1], IP-IRM [6]). Apart from the two commonly used CNNs (AlexNet and DenseNet-201) we also include IP-IRM, as it was designed to contain highly disentangled latent representations, which might have an effect on the mapping performance to the W-space. Note that both AlexNet and DenseNet-201 have a higher-dimensional latent space compared to ResNet-50, while IP-IRM was built on the ResNet-50 architecture. We observe similar performances for all three types of linking networks (see Fig. S2 for example images and Fig. S3 for loss functions), with the exception of DenseNet, which shows a better performance for the non-linear methods than linear regression. Note that DenseNet is also the model with the largest representational space. Overall, these results indicate that our method can be flexibly transferred to different classifiers.

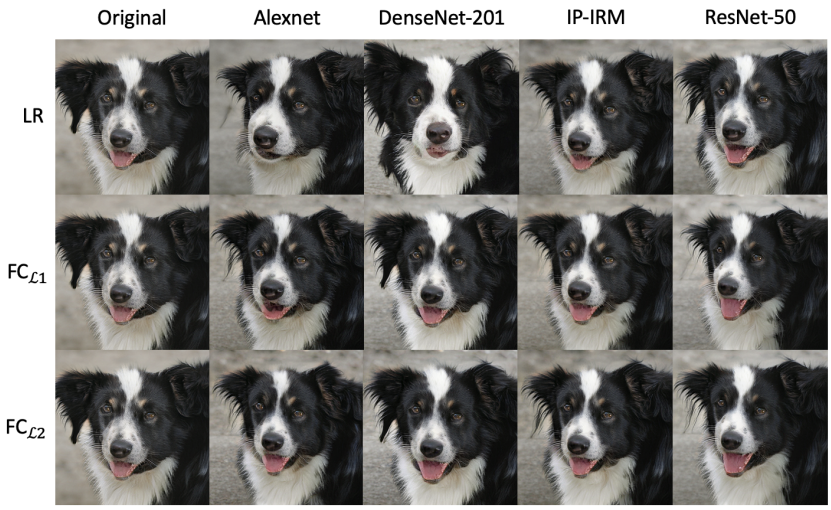

Fig. S2: Examples of reconstruction performance for different classifiers. We traine three different linking networks for four different classifiers (see Sec. S2). The classifiers include AlexNet (second column), DenseNet-201 (third column), IP-IRM (forth column), and ResNet-50 (last column). Here we depict one example of the reconstruction performance after an image (left column) was mapped a full cycle, i.e., through the classifier, the respective linking network, and the GAN (as in Fig. S1A). For comparison, we also include the ResNet-50 results from Fig. S1A.

Additionally, we demonstrate broader applicability of our approach by studying ResNet-50 classifiers that were trained to differentiate gender or ethnicity from human faces. Note that the data sets used for training these classifiers are different from the data set used for training the GAN (FFHQ). Fig S4 shows

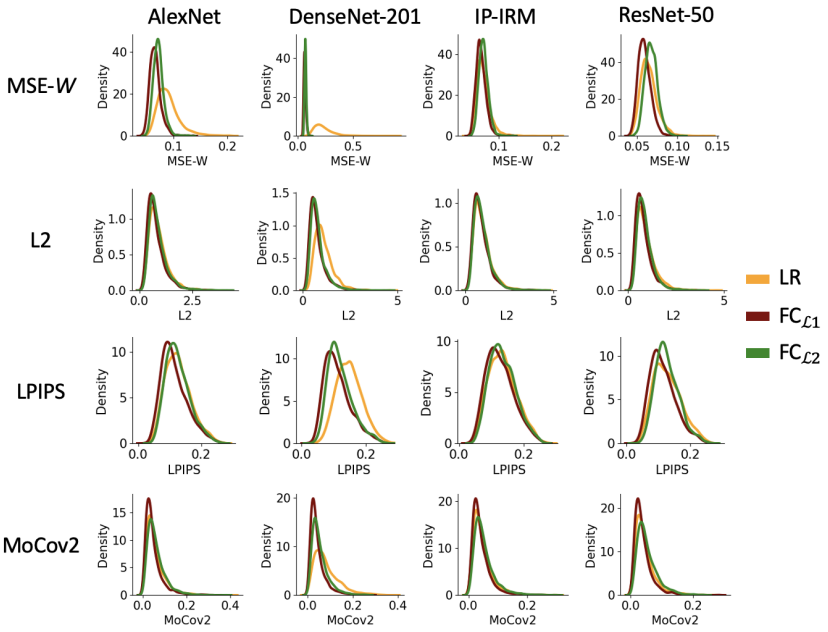

Fig. S3: Similar reconstruction performance across different linking networks. We compare the reconstruction performance of the different linking networks for four classifiers. The different classifiers include AlexNet (first column), DenseNet-201 (second column), IP-IRM (third column), and ResNet-50 (last column). For each classifier, we compute the performance of the linking network as the mean squared error (MSE) in the W-space (top row) for 1,000 different initializations (i.e., randomly chosen ws and their generated images). Additionally, in the image domain, we compute the pixel-wise L2 distance (second row), LPIPS (third row), and MoCov2 (last row) between the original image (I) and the full-cycle mapped image ($\tilde{I}$). For comparison we include the results for ResNet-50 (reproduced from Fig. S1).

single-unit representations using the label-free face-trained StyleGAN-XL and classifiers trained on gender (top) or ethnicity (bottom), linked with linear regression. Our method reveals that individual units in the penultimate layer encode complex, human-interpretable features related to gender or ethnicity.

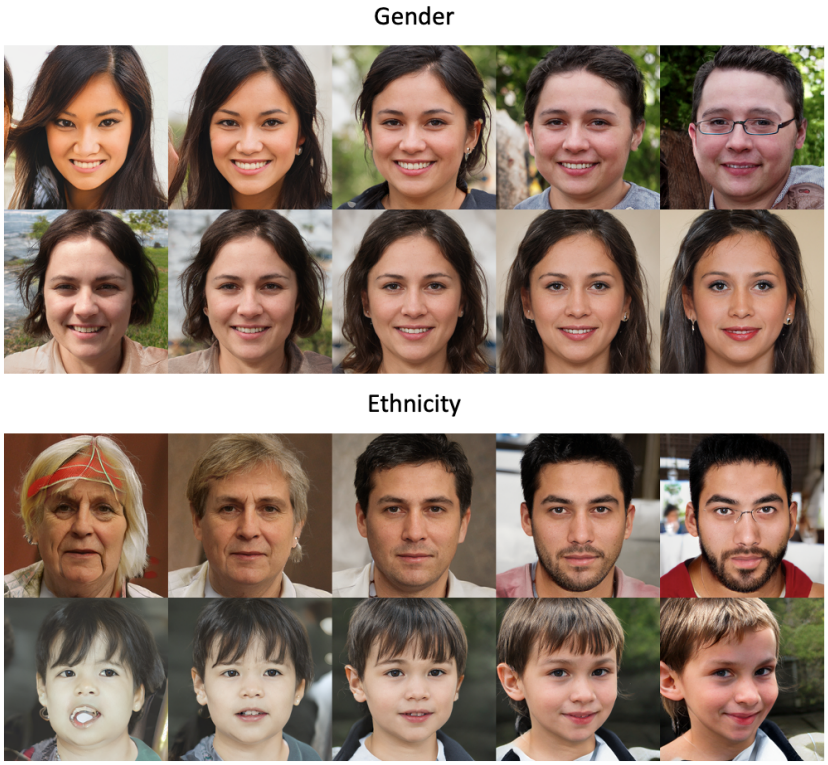

Fig. S4: Application to face-trained classifiers. For different classifiers trained either to distinguish between genders (top) or ethnicity (bottom) our pipeline reveals learned representations in several units in the penultimate layer.

Table S1: Image segmentation labels for different types of subclasses

Subclass	Labels
dogs	nose, snout, eyes, ears, tongue, head, body, legs, tail
birds	beak, eye, head, leg, wing, body tail
fungi	cap, stem

S4 Image segmentation model

As we study to what extend CNN classifiers learn fine-grained, abstract features such as different shapes of snouts, we use image segmentation to reveal changes in such concepts given changes in the representation space. Specifically, we use the method introduced by [5] to predict segmentation masks from the activations in the GAN using a few-shot learning approach (Fig. S5).

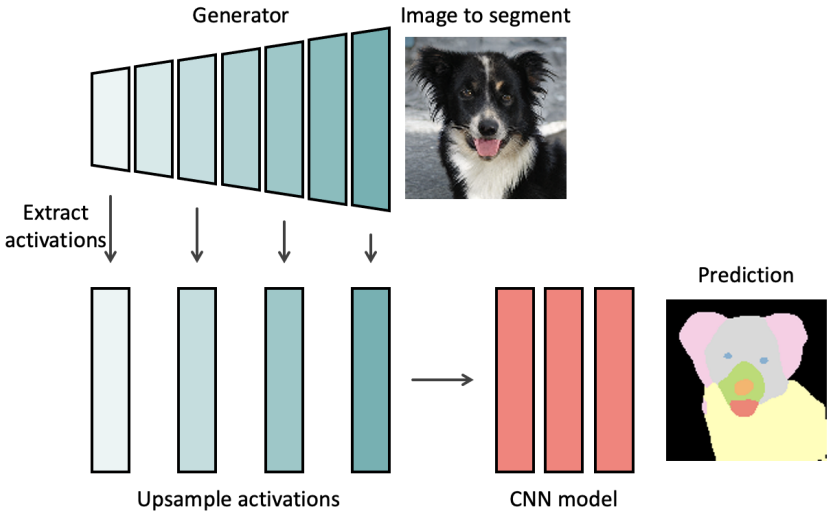

Fig. S5: Image segmentation method to extract fine-grained features. We use the method introduced by [5] to generate fine-grained segmentation masks across several concepts using a few-shot learning approach.

We label five instances per class with varying numbers of labels (see Table S1). Note that, to the best of our knowledge, these labels show higher detail than any open-source database for image segmentation.

The image segmentation model trained with five classes generalizes to similar classes (e.g, other dog breeds; Fig. S6), drastically reducing the labeled data needed.

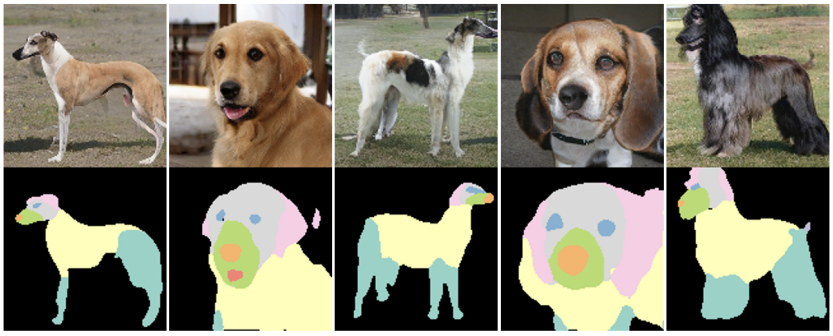

Fig. S6: Image segmentation model generalizes to other (similar) classes. We train the image segmentation model for five dog classes and find high generalization to other dog classes.

Additionally, we label a test set of another five images per class to validate the performance of the image segmentation approach using the intersection over union (IoU) between the labeled and predicted regions (Fig. S7). We observe particularly high performance for fungi. For the other subclasses the performance appears to be correlated with size and frequency of occurrence of the label.

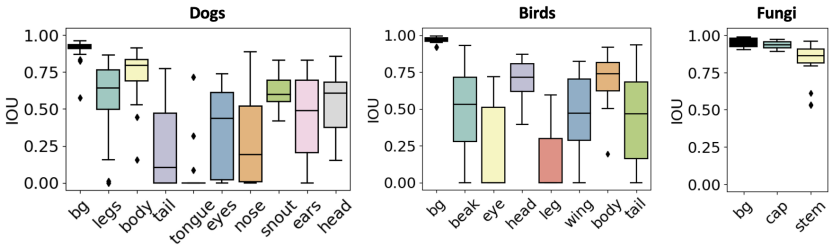

Fig. S7: Image segmentation validation. We compute the intersection over union (IOU) between labels of a manually labeled test set and the prediction using the few-shot learning approach.

Depicting the labeled results indicates that errors in the segmentation masks are most often confined to the border of the object (Fig. S8). Note that our method monitors changes in the segmentation mask, hence such errors do not affect much the performance of our approach, as long as these are consistent across all images of the trajectory.

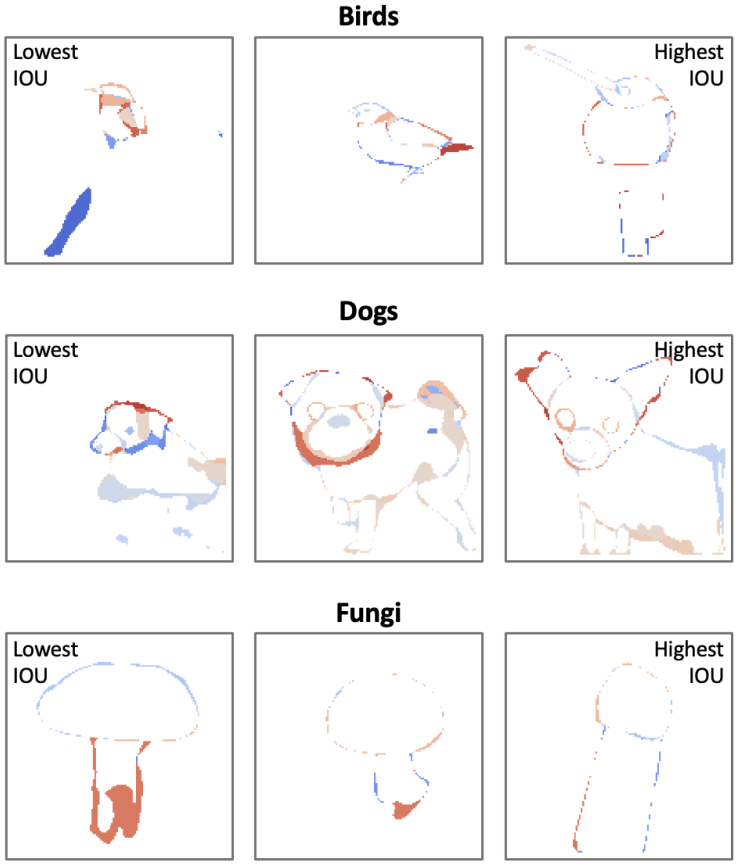

Fig. S8: Image segmentation validation – individual images. We depict the difference between the original and predicted labels for images with the lowest IoU (left), average IoU (center), and highest IoU (left).

S5 Interpretation of single unit representations

S5.1 Supervised evaluation method

For each quantification metric (area, luminance, entropy, eccentricity, and angle) and for each label (leg, body, tail, tongue, eye, nose, snout, ear, head), we can find the units that encode the largest change. In the main text, we mostly report changes in area, luminance, or entropy. In Fig. S9 we additionally show two examples with a large change in the eccentricity of the ears (left) and a large change in the angle of the nose and snout (right).

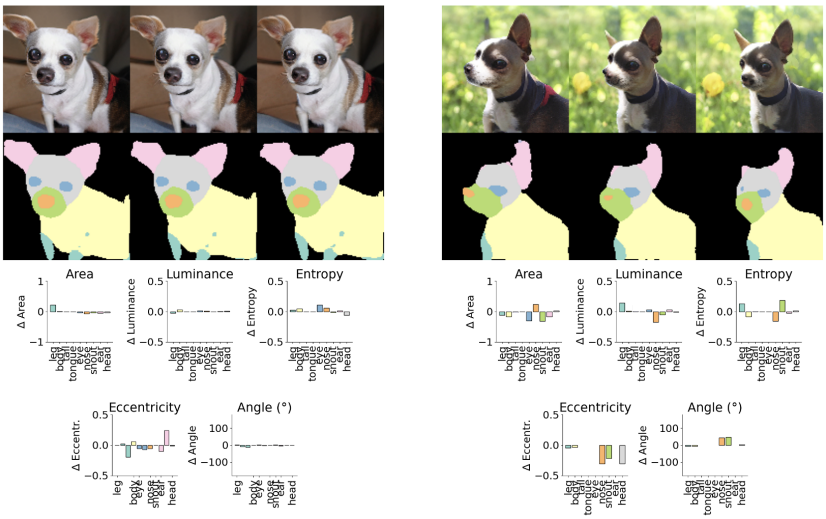

Fig. S9: Examples of units encoding a high change in eccentricity and angle. We display the change along the image trajectory for tuning a single unit, as well as the change in quantification metrics. Left) An example of a large change in ear eccentricity. Right) An example of a large change in nose and snout angle.

We find these units through an exhaustive search across all units for the largest change in ear eccentricity (Fig. S9 left) and snout angle (Fig. S9 right). Specifically, we rank units by the amount of change for specific quantification metrics. Although we observe some overlap between the ranking in units, each metric allows us to discover a specific set of units. Figs. S10 and S11 show examples of quantification metrics for specific labels. In Fig. S11 we observe similar units to be relevant for luminance and entropy (e.g., row two corresponds to the same unit in both metrics, and row five of luminance corresponds to row three in entropy). However, we can also discover non-overlapping units such as the unit with the highest entropy change , where only small changes in luminance can be detected, while the color of the head of the dog becomes much smoother (see Fig. S11, upper right row).

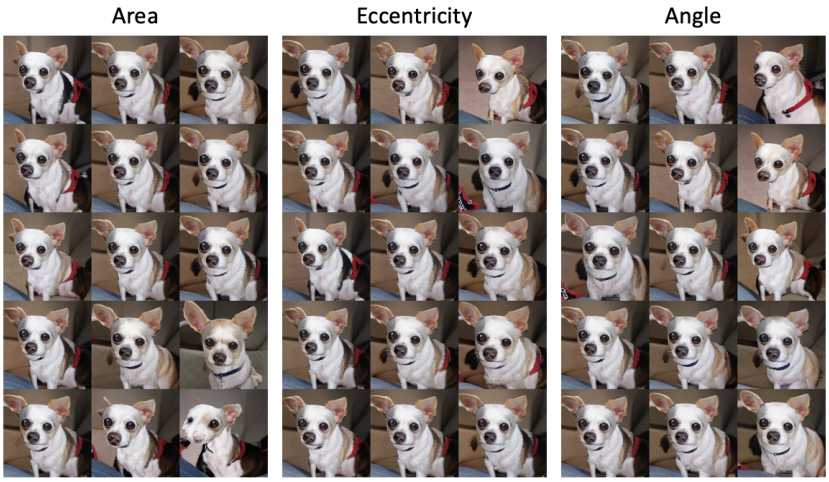

Fig. S10: Illustration of effects on the label ear caused by single unit activation changes, across different quantification metrics. For one example image, we depict the changes associated with the change in the activation of a single unit, focusing on those units which are identified to represent the largest changes in area (left), eccentricity (middle), or angle of the ear (right). Different rows depict the effect of different units.

Across different quantification metrics, we observe a continuous distribution of feature changes that resembles a normal distribution (Fig. S12). Hence, R may not be disentangled into single units that strongly encode a specific feature, but rather several units represent one feature and in combination account for the entire representation encoded in the classifier.

In the main text (see Fig. 6A and Sec. 4.4), we show how our method can be used to discover disentangled representations in single units and illustrate examples. Here, we now quantify the sparsity (disentanglement) of R for individual labels. Therefore, we first tune every single unit in R and generate the respective image. We then compute the change in each quantification metric caused by the change in unit activation. This results in a vector with the same dimensions as R for each metric and label. We then scale this vector to unit length and compute the sparsity. A sparsity approaching one would indicate that few units represent that label-metric combination, whereas a sparsity approaching zero would indicate that this label-metric combination is represented across many units (distributed). We repeat this process for several seeds. We observe that some features (e.g., tongue, red, Fig. S13) are sparsely encoded, while others are more distributed (e.g., head, gray, Fig. S13). Our method can hence reveal insights into how R is structured in terms of disentangled representations.

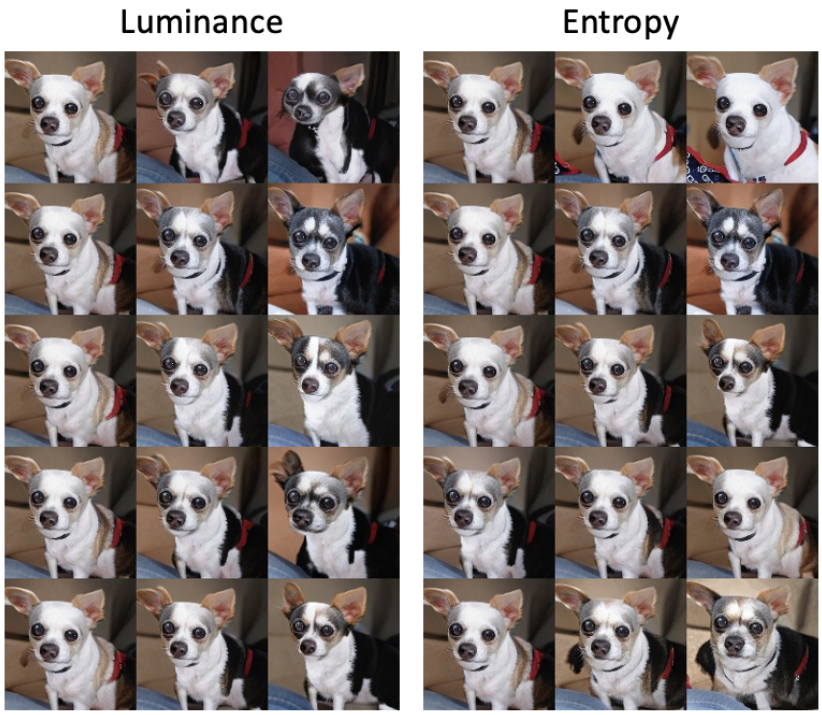

Fig. S11: Highest changes in the label head in luminance and entropy. For one example image, we depict the units that were identified to represent large changes in luminance and entropy of the head.

S5.2 Unsupervised evaluation method

Our unsupervised evaluation pipeline uses PUMP [4] to compare images. PUMP is a feature matching method based on finding corresponding representations between two images in an unsupervised fashion. Specifically, PUMP builds a correlation volume between two images first using the local consistency of found matches by iteratively comparing representations in a parametric module. Additionally, PUMP specifies that each data point in the one image can at most match to one point in the second image.

We use our unsupervised evaluation pipeline (see Sec. 3.2) to visualize feature changes associated with individual units. In Fig. S14 we demonstrate how our method highlights individual concepts that are tuned with activation changes in a single unit. We produce such trajectories by continuously, linearly increasing the activity of a single unit within a fixed range and generating the corresponding sequence of images. We then use PUMP [4] to align this image sequence via affine transformations using corresponding features between the images. Given N images, we first align image I_N to I_{N-1} using PUMP and the affine transformation to produce I_N^a. Then, we compute I_{N-1}^a by aligning I_{N-1} to I_{N-2}. I_N^a is

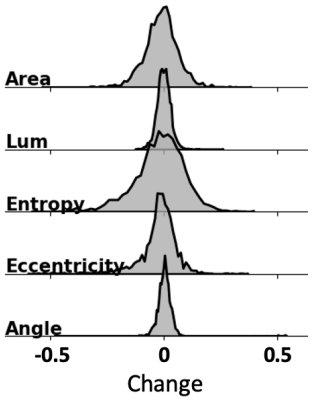

Fig. S12: Changes in individual metrics are similar to a Normal distribution. For each quantification metric (area, luminance, entropy, eccentricity, angle) we display the distribution of changes across all units. Note that we scale all metrics to be between -1 and 1.

then again aligned to I^a_{N-1}. We repeat this procedure until image I_0 is reached. We then compute PUMP again on this aligned sequence, to extract and visualize the local feature changes that are depicted in Fig. S14.

Additionally, we can compute the strength of global feature changes such as translation or rotation from the affine transformation. Extracting these features across several input instances reveals that some units are tuned for such global representations. Specifically, unit 278 in Fig. S15 (orange) shows a clear preference towards a translation to the lower right whereas units 300 (blue) and 1312 (yellow) are overall not encoding any global features.

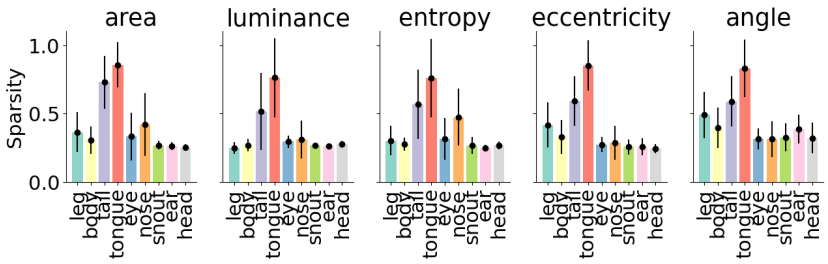

Fig. S13: Discovering disentangled concepts in R. We compute the average sparsity of each label and each quantification metric across all units in R to discover concepts that are represented sparsely in R. Values close to one indicate high sparsity, whereas values close to zero indicate distributed representations. The error bar is computed across 100 different seeds.

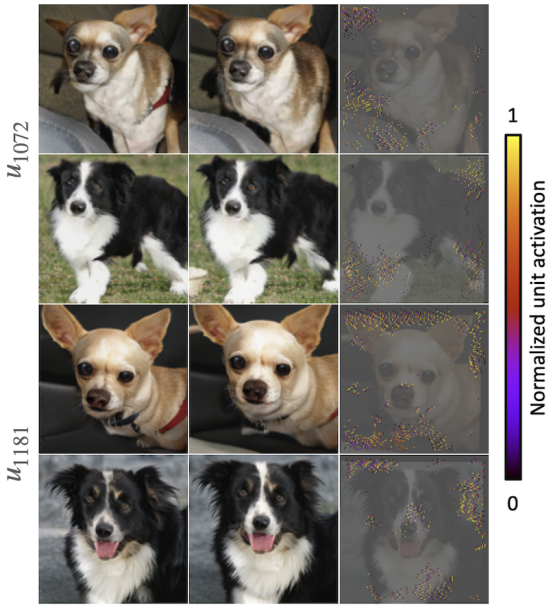

Fig. S14: Unsupervised discovery of single unit concepts. Given an initial image, we tune the activation of single units and use our unsupervised evaluation pipeline to visualize the relevant concepts encoded by these units. The trajectories (right column) indicate the direction of the localized feature changes and the color describes the activation strength of the tuned unit.

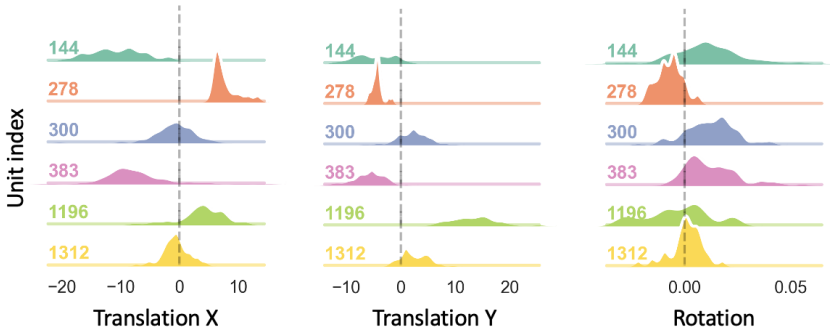

Fig. S15: Extracting global feature changes reveals location-tuned units. We extract global features (translation, rotation) from the affine transformation computed using our unsupervised evaluation method (Sec. 3.2).

S6 Class relevance of single units

In the main text, we describe how single units can be relevant for a single class and also show that some units are relevant for several classes (Sec. 4.4). In Fig. S16 we additionally show the distribution of a single unit's class relevance for the five different dog classes, based on 100 different seed images. This analysis allows to draw several conclusions about the structure of R. Firstly, it reveals differences in a single unit's relevance for different classes. For example, unit 1113 is highly relevant for the Chihuahua class but not for the Weimaraner (Fig. S16, left). Hence, analyzing the units across classes can help to discover concepts learned by the classifier to distinguish between classes. Secondly, since class relevance is described as a distribution, sometimes with few outliers, our method can be used to analyze the robustness of the classifier with respect to single-unit changes in specific concepts for a given class.

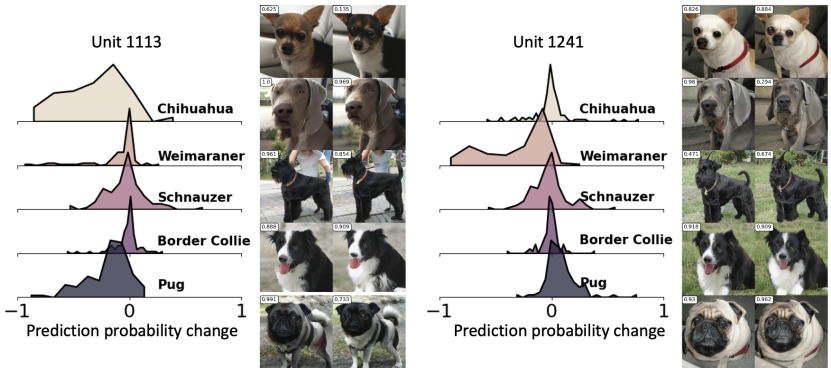

Fig. S16: Distribution of class relevance of single units shows different distributions across classes. We compute the change in prediction probability for the original input class when tuning a single unit in R. We observe that for different units, some classes show a higher change in prediction probability than others. Also, across 100 examples per class, we observe that some examples are more robust to a change in a unit's activation than others.

S7 Counterfactual directions in R

In Sec. 3.3, we describe the application of our method to the generation of counterfactual examples, and the analysis of the decision boundary. We compute the counterfactual example of a given representation r with the aim of shifting its predicted class to a target class c_{target} while keeping the change in the image minimal. To find a counterfactual direction, we randomly initialize the shift vector and then optimize it to shift a representation $r \in R$ in the direction of a target class. The optimization results are deterministic for a given vector r and a target class c_{target}. However, we find a variety of counterfactual directions using different initializations. In Fig. S17, we show additional examples of sequences sampled along counterfactual trajectories for different starting conditions.

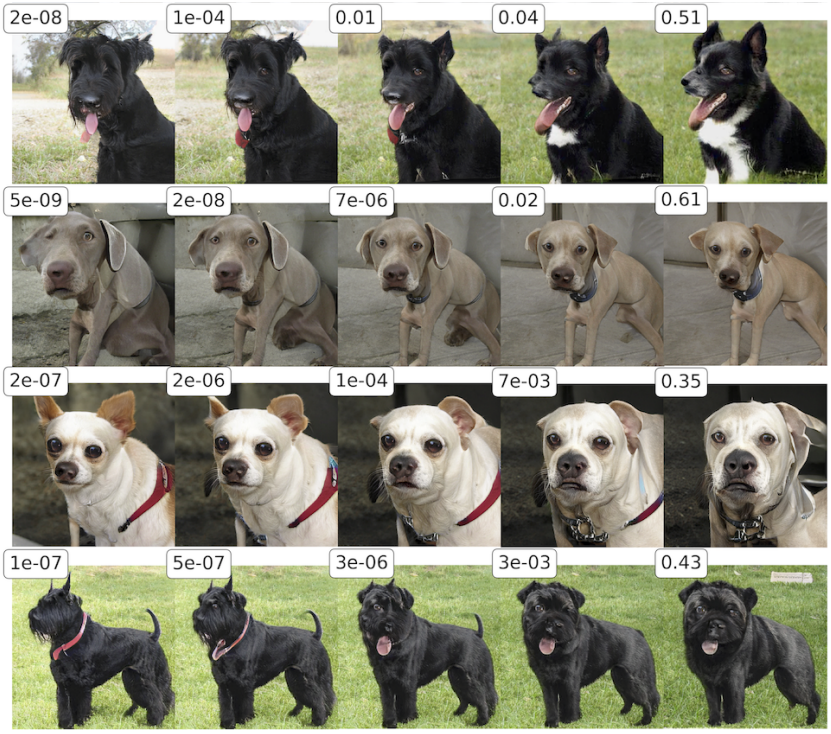

Fig. S17: Sampling along the counterfactual direction. Similar to Fig. 7 in the main text, we generate image sequences by linearly sampling along a counterfactual direction between an original and target class. The classifier's prediction probability for the target class is shown in the top left corner. Note the large increase in prediction probability around the decision boundary (between the second to last and last image).

Additionally, in Fig. S18 we show more examples of the continuous feature change along the counterfactual direction, similar to Fig. **??** in the main text. We

observe mostly smooth changes in single features across the decision boundary, but sometimes individual features change drastically, or even show kinks close to the decision boundary.

Next, we show that the counterfactual direction is more sparse between some pairs of classes than between others. We compute the sparsity of the counterfactual direction across different initializations and show the distributions for all combinations of the five dog classes (Fig. S19). We observe some classes to allow for a less sparse counterfactual direction (e.g. original class 151 (Chihuahua) and target class 254 (Pug)). Note that we here cycle the counterfactual representation through the linking network, GAN, and back into the classifier and only report counterfactual directions that are actually *reachable* by the model. Computing the sparsity of counterfactual directions is, in principle, also possible only in R, however, R does not have to be continuous. Hence, our method identifies directions that can be represented in R.

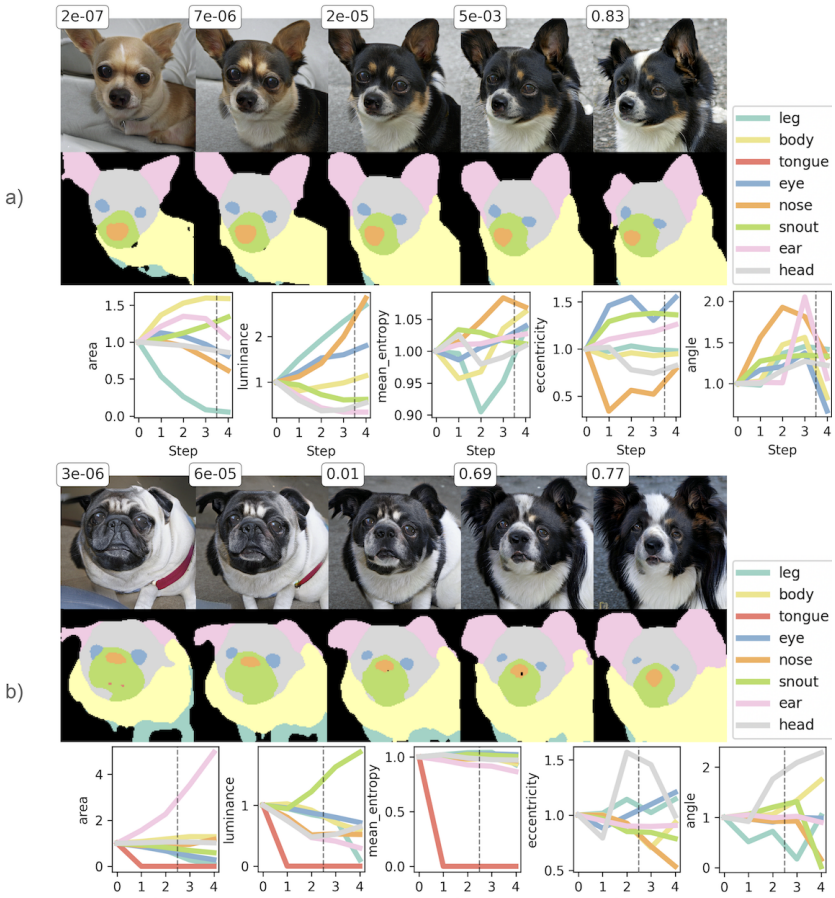

Fig. S18: Evolution of quantification metrics along the counterfactual trajectory. For two counterfactual examples, we depict the changes in individual labels and metrics. The generated images and respective segmentation masks along the counterfactual trajectory are shown on the top. The prediction probability for the target class is indicated in the upper left corner of each generated image. The evolution of the quantification metrics for each label is shown below the segmentation masks (bottom row). The decision boundary is marked by a gray dotted line. Note that, for simplicity, we set the decision boundary in the middle between the last image that was predicted as the original class and the first image predicted to be in the target class. Each color represents one label type.

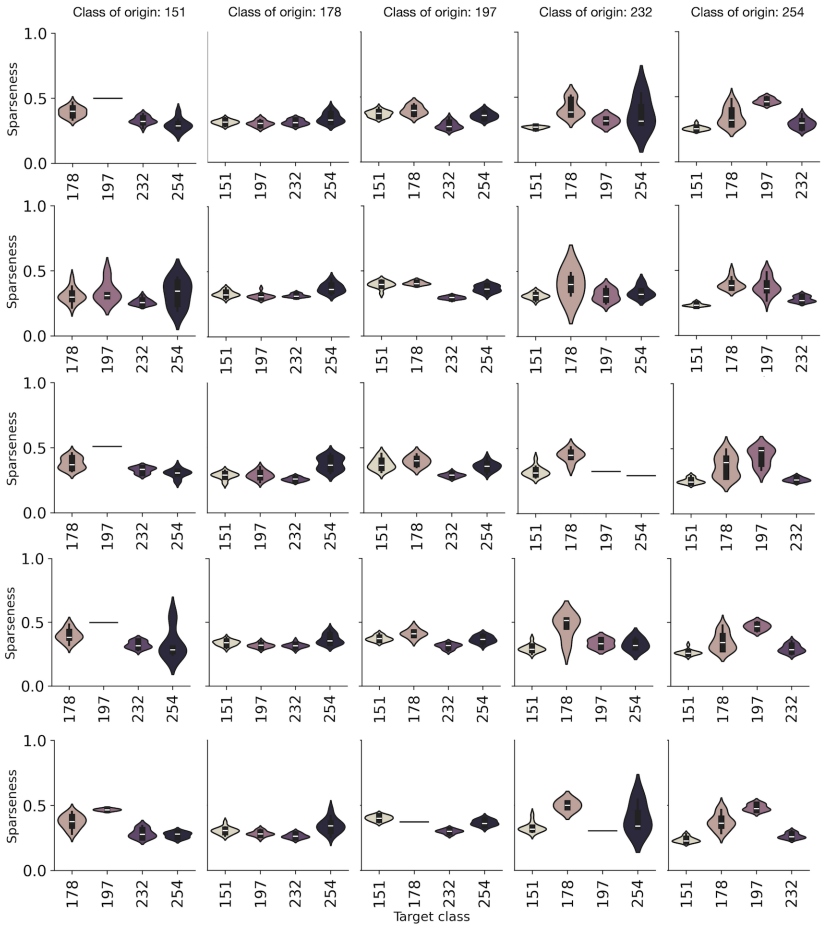

Fig. S19: Different levels of sparsity between the counterfactual directions of different class pairs. Given a target class, and an image generated from an original class, we consider the shift vectors resulting from the counterfactual optimization process described in Sec. **??**. Across 10 different initializations, we measure the sparsity s as in Eq. **??** for each counterfactual direction and show the distribution. Each row depicts a different input image of the original class. The colors correspond to the target class. The original class is indicated as the title of a column.

S8 Counterfactual directions in the GAN

As the StyleGAN-XL was trained with all ImageNet classes, we can also find counterfactual directions in the GAN instead of the classifier. This allows for an analysis of the change in R related to a given change in the class. In StyleGAN-XL, a latent vector z and a class embedding c are mapped in the W-space using a mapping network (G_m; see Sec. **??**). Here we exploit the GAN's property to produce similar images (viewpoints, features) across classes when sampled from the same z. To create a counterfactual example of one image of a given class, we produce an example of a target class that utilizes the same z but a different class embedding. We can then linearly interpolate in W between these two examples and generate the respective images (Fig. S20 top left). Using our pipeline, we can first extract the most relevant features that differ between classes (Fig. S20 right). We can then map all the resulting images in the classifier and analyze, for example, the units in R that exhibit the largest change. We observe that these units encode human-interpretable features such as a change in color or a change in the shape of the ear (Fig. S20 bottom). Note that along these counterfactual directions more features change, since we do not regularize on minimal shifts in W as for the counterfactual directions in R. However, using the GAN allows us to keep many global features similar while only changing the features that the GAN learned to associate with specific classes. These analyses demonstrate that the GAN not only acts as a generation tool but also offers a valuable source for interpreting learned representations in the classifier.

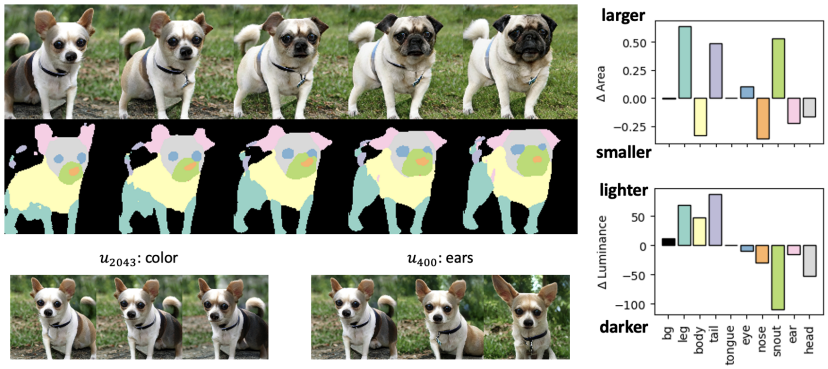

Fig. S20: Counterfactual examples created in the GAN reveal relevant units. We used the StyleGAN-XL to generate an image of a different class with the same viewpoint (top left). We then use our proposed pipeline to analyze the relevant features that change between the classes (right). Encoding the image trajectory into the classifier, allows us to extract the units with the largest change. Using our analysis pipeline, we observe that these units encode human-interpretable features such as a change in the color or ear shape.

S9 Replication to other classes

In this section, we show that our method also generalizes to other sets of classes.

S9.1 Fungi

First, we train our pipeline with four different classes of fungi (Agaric, Gyromitra, Stinkhorn, and Bolete). In Fig. S21 we identify units that represent individual features of fungi and show that they are transferable between similar classes (Fig. S22). Our method also reveals a high robustness of the different classes of fungi to changes in single units. Only one unit (1131) has a strong effect on the prediction probability (see Fig. S23).

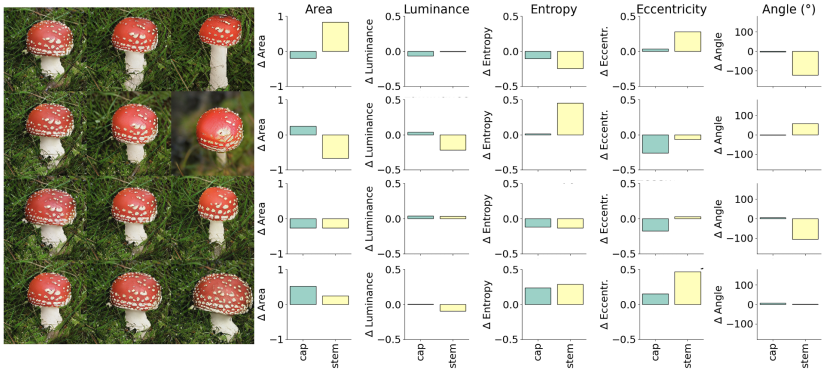

Fig. S21: Single-unit representations in Agaric. Our method reveals units that represent relevant features of fungi. We identify units that, for example, represent the shape (eccentricity) and size (area) of the stem or cap. The images on the left show the generated images along the trajectory of a single unit change (from left to right, each row one unit). The right graphs show the corresponding changes in the quantification metrics.

Our method can also be used to analyze the decision boundary between the different classes of fungi. We show several counterfactual examples in Fig. S24. Note that here we also observe a large increase in prediction probability once the decision boundary is reached; however, the change appears to be less rapid than for the dog breads, and the prediction probability increases further after crossing the decision boundary. As shown in Fig. S25 the classifier seems to be highly robust to general changes in area for both counterfactual examples, whereas small changes in the entropy already change the predicted class.

S9.2 Birds

Additionally, we show the generalizability of our framework to six different classes of birds (Brambling, House Finch, Indigo Bunting, Bee Eater, Jacamar

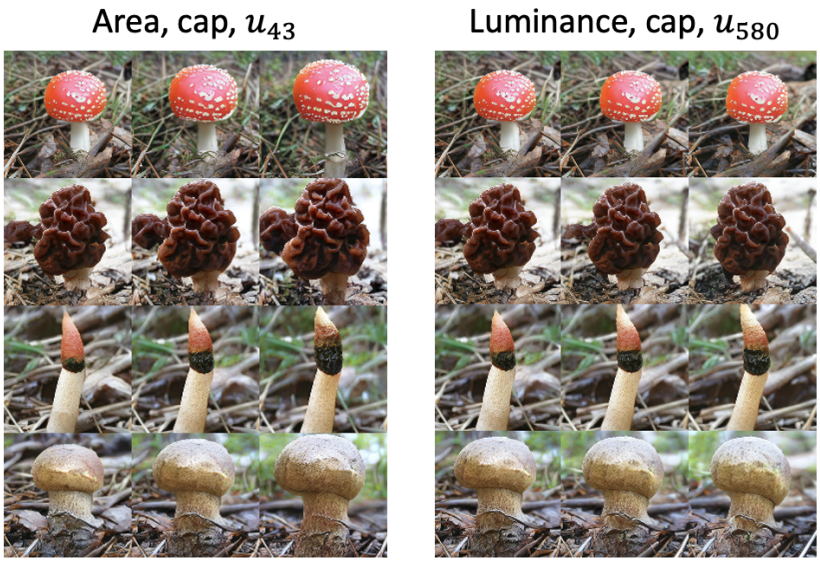

Fig. S22: Single-unit representations across different classes of fungi. We show examples of representations in single units that are transferable across other classes of fungi. However, the Stinkhorn (third row) seems to be encoded differently from the other three fungi. The unit shown on the left that is selective for the size of the cap in the three other fungi is rather selective for the color of the cap in the Stinkhorn than for the size.

and Toucan). We train a linear linking model to map the different examples of birds from the ResNet-50 representation layer to the W-space. Again, we find that single units encode specific features and the features can be transferred between classes (Fig. S26). We further show that our method can again be utilized to study counterfactual examples for different classes of birds (Fig. S27).

The two further examples (fungi and birds) studied in this section suggest that our method can be transferred also to many other classes, hence opening up the possibility of a comprehensive analysis of the representations and decision boundaries of a classifier.

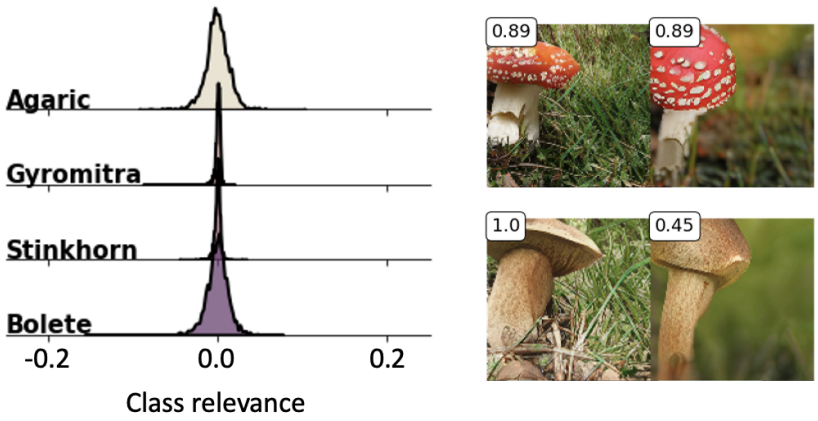

Fig. S23: Different classes of fungi show high robustness to changes in individual units. For 100 different seeds, we compute the change in prediction probability when altering the activation of a single unit. Left) We observe high robustness for some classes (Gyromitra or Stinkhorn), whereas others (Agaric and Bolete) appear to be less robust. Right) Only unit 1131 has on average a strong effect on the prediction probability of the Bolete (≥ 0.15). We depict an example image of an Agaric and a Bolete for this unit. The prediction probability for the Agaric does not change, whereas the prediction of the Bolete changes to Mushroom.

Fig. S24: Counterfactual examples for several classes of fungi. We depict the images along the trajectory of generated counterfactual examples. The example image of the original class, indicated on the left, is changed using our optimization procedure, such that the target class, indicated on the right, is predicted, while minimizing the change in W.

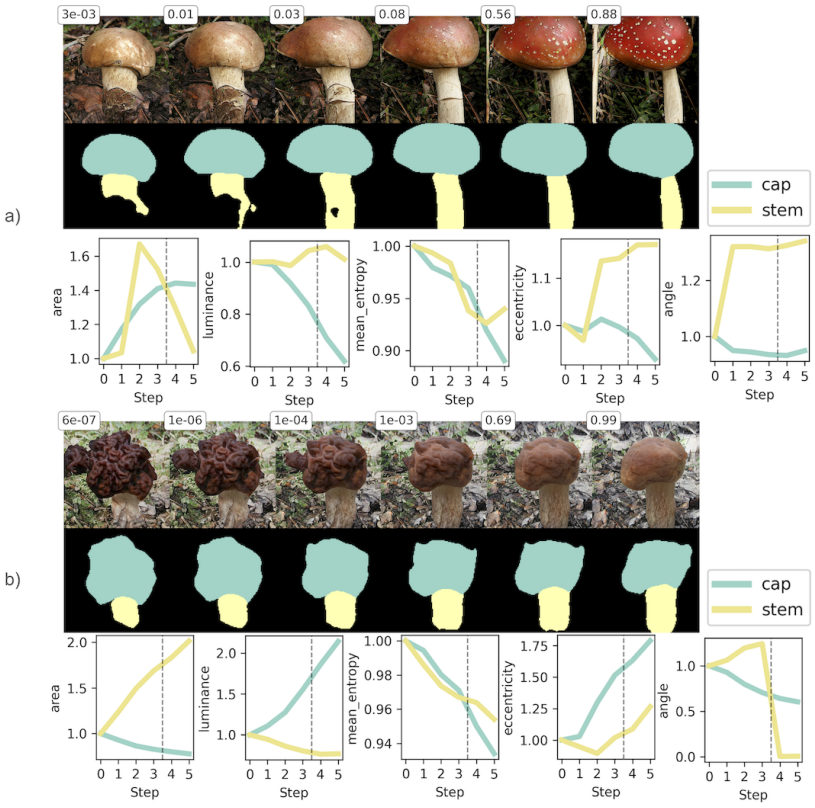

Fig. S25: Examples of the trajectories of feature changes along counterfactual direction. For two counterfactual examples, we depict the changes in individual labels and for all quantification metrics. The generated images and respective segmentation masks along the counterfactual trajectory are shown on the top left. The prediction probability for the target class is indicated in the upper left corner of each generated image. The evolution of the quantification metrics for each label is shown below the segmentation masks. The decision boundary is marked by a gray dotted line. Note that for simplicity, we set the decision boundary in the middle between the last image that was predicted as the original class and the first image predicted to be in the target class. Each color represents one label type.

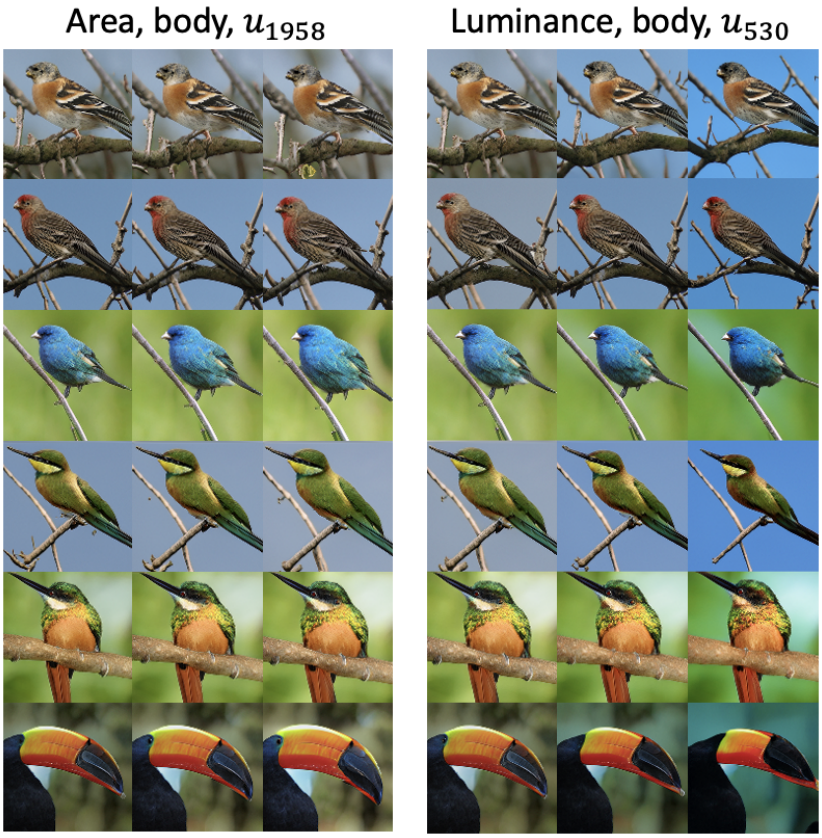

Fig. S26: Single-unit representations across different classes of birds. We show that representations in single units are transferable also across classes of birds. However, the changes are much more subtle than in dogs and fungi.

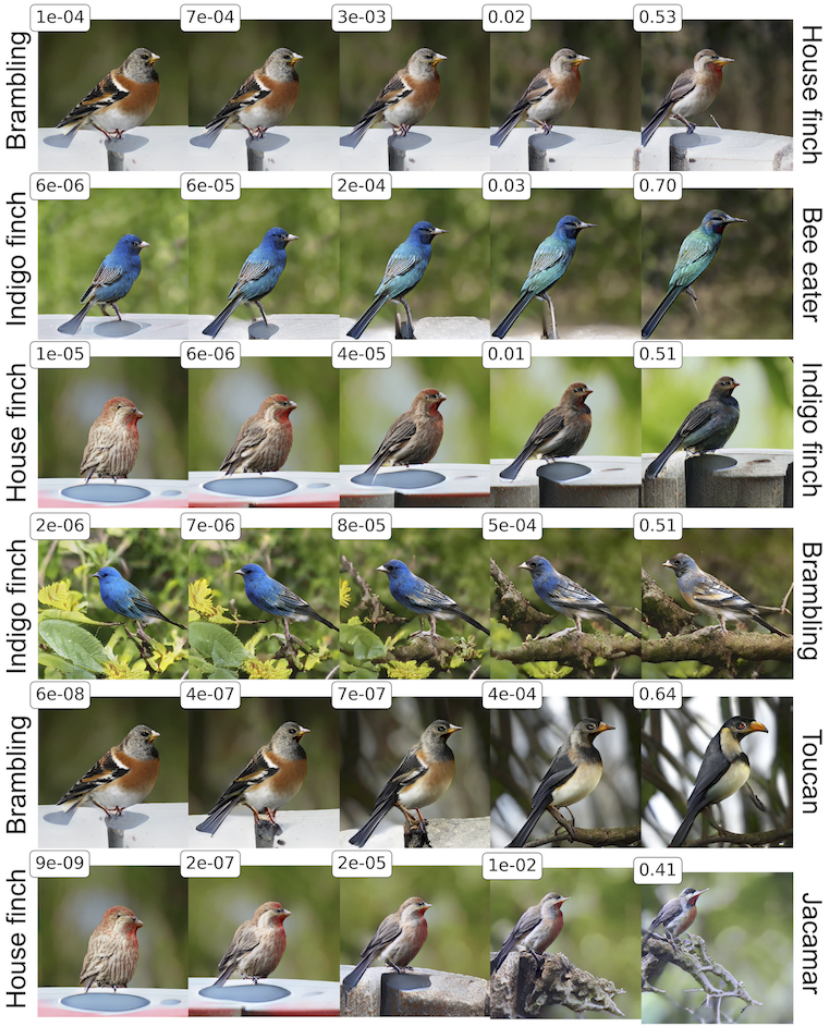

Fig. S27: Counterfactual examples for several classes of birds. We depict the images along the trajectory of generated counterfactual examples. The example image of the original class, indicated on the left, is changed using our optimization procedure, such that the target class, indicated on the right, is predicted, while minimizing the change in W.

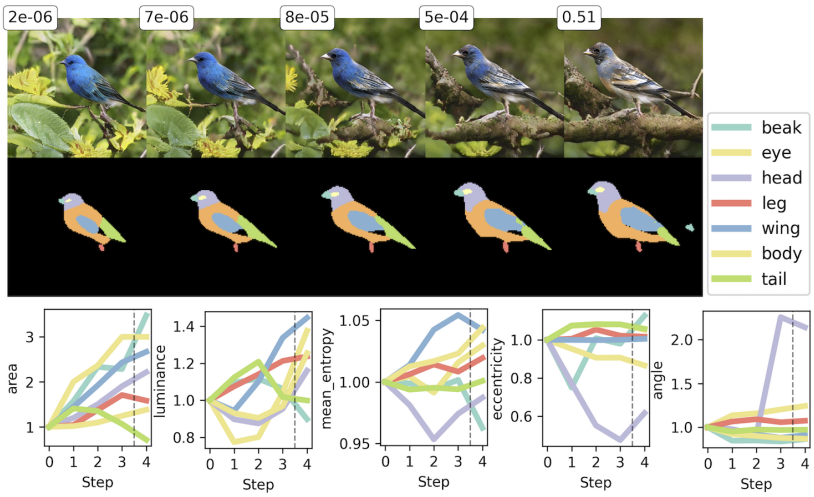

Fig. S28: Examples of the trajectories of feature changes along counterfactual directions. We depict the changes in individual labels for all quantification metrics (bottom row). The generated images and respective segmentation masks along the counterfactual trajectory are shown on the top. The prediction probability for the target class is indicated in the upper left corner of each generated image. The evolution of the quantification metrics for each label is shown below the segmentation masks. The decision boundary is marked by a gray dotted line. Note that for simplicity, we set the decision boundary in the middle between the last image that was predicted as the original class and the first image predicted to be in the target class. Each color represents one label type. Note that, for instance, for 'leg' several metrics indicate a kink near the decision boundary.

References

1. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely Connected Convolutional Networks (Jan 2018). https://doi.org/10.48550/arXiv.1608.06993
2. Härkönen, E., Hertzmann, A., Lehtinen, J., Paris, S.: GANSpace: Discovering Interpretable GAN Controls (Dec 2020). https://doi.org/10.48550/arXiv.2004.02546
3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012)
4. Revaud, J., Leroy, V., Weinzaepfel, P., Chidlovskii, B.: PUMP: Pyramidal and Uniqueness Matching Priors for Unsupervised Learning of Local Descriptors. pp. 3926–3936 (2022)
5. Tritrong, N., Rewatbowornwong, P., Suwajanakorn, S.: Repurposing GANs for One-shot Semantic Part Segmentation (Jul 2021). https://doi.org/10.48550/arXiv.2103.04379, http://arxiv.org/abs/2103.04379, arXiv:2103.04379 [cs]
6. Wang, T., Yue, Z., Huang, J., Sun, Q., Zhang, H.: Self-Supervised Learning Disentangled Group Representation as Feature (Oct 2021). https://doi.org/10.48550/arXiv.2110.15255