A Implementation details

This section discusses additional implementation details of our experiments in Sec. 4 and Sec. 5.

Data. We use the following publicly available datasets for fine-grained image recognition: CUB [58], iNaturalist-2021 [55], and StanfordDogs [28]. The image classifier implementation follows the typical implementation in PyTorch and uses standard image augmentations (i.e., random resized cropping and random horizontal flipping). We provide additional details for each dataset below:

- **CUB**: The CUB dataset consists of images of 200 bird species annotated with keypoint locations of 15 bird parts, e.g., crown, beak, etc. Some of the keypoint annotations distinguish between the left-right instances of parts: 'left wing' / 'right wing', 'left leg' / 'right leg', and 'left eye' / 'right eye'. We treat these as a single part during the evaluation of the Near-KP and Same-KP metrics, i.e., 'left wing' and 'right wing' as 'wing'.
- **iNaturalist-2021**: The iNaturalist-2021 dataset consists of various super-categories (e.g., plants, insects, birds, etc.), covering 10,000 species in total. The dataset contains a larger number of classes and more complex scenes compared to other fine-grained image recognition datasets. Therefore, the iNaturalist-2021 dataset can be considered as a more challenging testbed for our approach. However, this dataset lacks keypoint annotations. We used the bird supercategory for our quantitative evaluation in Sec. 4, and requested human annotators to provide part keypoint information for 2,060 validation images. The keypoint definitions from the CUB dataset are used. We did not perform a quantitative evaluation of other supercategories, as these are considerably more challenging to annotate. Specifically, we identified the following challenges: (i) some supercategories do not have identifiable parts (e.g., fungi), and (ii) some supercategories are too diverse, and do not support common keypoint definitions across all sub-categories (e.g., plants, mammals, insects, etc.). We do provide qualitative results on several of these supercategories in Sec. C.
- **StanfordDogs**: The StanfordDogs dataset contains images of 120 dog breeds taken from the ImageNet [19] dataset. The keypoint annotations are provided by [10]. Again, we treat left-right instances of parts as the same part, i.e., left ear and right ear as just ear.

Classifier. The training follows the typical VGG-16 and ResNet-50 implementation in PyTorch [37] with 100 epochs. All models use pre-trained ImageNet [19] weights. The training uses stochastic gradient descent with momentum 0.9 and weight decay of 0.0001. We use batches of size 32 for the CUB and Stanford Dogs datasets, and batches of size 256 for the iNaturalist-2021 dataset. The initial learning rate is selected via grid search and decreased by 10 at the 70-th and 90-th percentile of training.

Self-supervised models. We used the publicly available weights that were provided by the authors of the respective works [13,23]. For DeepCluster and SWAV, we adopt the models trained via the multi-crop augmentation from [13]. We pre-trained all models on ImageNet. Different pre-training schemes could be used when considering more specialized domains like medical images.

Parts detector. We trained a parts detector on CUB. The parts detector consists of a ResNet-50 backbone followed by a 1×1 convolutional layer. The input images are 224×224 pixels and the output has spatial dimensions 7×7. We project the ground-truth bird keypoint annotations onto a grid of shape 7×7 and train the parts detector to predict keypoint presence via a multi-class cross-entropy loss. The loss is only applied to cells that contain at least one keypoint. Training follows the classifier implementation but we decrease the number of epochs to 50. The initial learning rate is 0.001. We evaluate the predictions via the mean AP metric (excluding cells that contain no keypoints). The best model obtains a mean AP of 92.3 on the validation set.

B Additional results

We report additional results that complement the ablation studies in the main paper.

Selection of query-distractor classes. The experiments in Sec. 4 examine counterfactual examples for query-distractor classes obtained via the confusion matrix - for a query class c, we select the distractor class c' as the class with which images from c are most often confused. This procedure differs from the approach in [22] which uses attribute annotations to select the most confusing classes. In particular, the authors select c' as the nearest neighbor class of c in terms of the average attribute annotations provided with the dataset. We argue that our setup is more generic as it does not use additional annotations. For completeness, we provide results with both selection procedures in Table S1. We observe that there are no significant differences in the results when adjusting the selection procedure. In practice, the two selection procedures often generate the same query-distractor class pairs. In conclusion, our selection procedure provides a viable and more generic alternative to the method from [22].

Background as a metric. The CUB dataset contains mask annotations that segment the foreground object. Prior work [22] used the object segmentation to measure how often the counterfactuals select cells belonging to the foreground object. Like the Same-KP metric, this *foreground metric* is a proxy for how often the counterfactuals select discriminative cells, i.e., cells that explain the class differences. For completeness, we report the results of the foreground metric in Table S2. Our method outperforms the baseline [22] in terms of the foreground metric, which indicates that we select more discriminative cells in the image. This

Table S1: **Comparison of different methods to select the query-distractor classes.** We select the distractor class as the most confusing class in the confusion matrix, or as the nearest neighbor class in terms of the average attribute annotations. The results are reported for a VGG-16 classifier on CUB.

Method	Selection Procedure	Near KP	Same KP	#Edits
Goyal *et al.* [22]	Confusion Matrix	54.6	8.3	5.5
	Attributes	55.0	8.6	5.4
Ours	Confusion Matrix	68.5	35.3	3.9
	Attributes	68.6	35.6	3.9

Table S2: **Ablation study of the foreground metric.** The results are with VGG-16 on CUB. We compare the baseline [22] against our method.

	Foreground	Near KP	Same KP	#Edits
Goyal *et al.* [22]	94.2	54.6	8.3	5.5
Ours	99.1 (+4.9)	68.5 (+13.9)	35.3 (+23.0)	3.9

observation aligns with our conclusions in the paper based upon the Near-KP metric. Note that the other metrics were discussed in Section 4.3.

Soft versus hard semantic constraint. We have modeled the semantic consistency constraint in a *soft* way (see Eq. 4). That is, we select replacements that balance the increase of $g_{c'}(\cdot)$ with the semantic similarity of the image regions. Alternatively, the constraint could be implemented in a *hard* way. That is, we cluster the auxiliary spatial features first, e.g., K=50, and only replace query cells with distractor cells from the same cluster. Table S3 compares the two mechanisms. The hard constraint selects considerably less discriminative cells (lower Near-KP and more edits) as it's more restrictive of the cells that can be replaced. In contrast, the soft constraint achieves better results, as it balances the $\mathcal{L}_s$ and $\mathcal{L}_c$ losses in Eq. 4.

Table S3: **Comparison of the hard and soft constraint mechanism**. Results are obtained with a VGG-16 classifier on CUB. We use a single distractor image.

Constraint	Near KP	Same KP	#Edits
Hard	40.1	13.3	9.3
Soft	**52.1**	**22.0**	**6.8**

Clustering. We studied the part clustering accuracy via different auxiliary models in Table 4. In this way, we verified whether the spatial feature represen-

tations of the auxiliary models are capable of disentangling parts. In this section, we provide qualitative results of this experiment. Figure S1 visualizes clusters found via a CUB classifier and DeepCluster model. We select several clusters and highlight cells assigned to the same cluster. The DeepCluster features better disentangle parts, i.e., cells assigned to the same cluster refer to the same part.

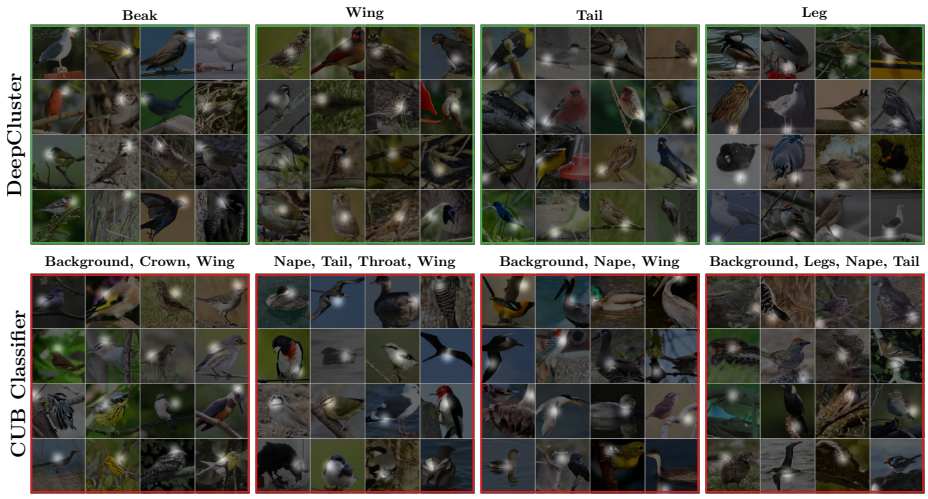

Fig. S1: **Clustering visualizations.** We study part disentanglement when clustering the spatial features obtained with different auxiliary models. We highlight image regions assigned to the same cluster (best viewed in color digitally). We indicate the parts found in each cluster for the purpose of visualization.

Ablation of pre-filtering operation. We study the influence of the pre-filtering step from Sec. 3.3 in Table S4. Namely, we vary k in the selected top-k% permutations to be used in the multi-distractor setup. As it can be observed, our method is very robust to the choice of the value k, so we selected k in order to achieve around x10 speedup over the vanilla multi-distractor approach.

Effect of receptive field. We discuss how the receptive field size of the classification network effects the quality of the counterfactuals. Large receptive fields can lead to poorer localization and reduce the counterfactual's quality. For example, the ResNet-50 models with receptive field size 299 yield lower numbers in Table 2 compared to their VGG-16 counterparts with receptive field size 212. We tried to mitigate this behavior by using features from the earlier `conv5_1` layer instead of `conv5_3` for ResNet and found it improves results. For example, on CUB, this change led to a reduction in the number of edits from 8.0 to 3.2.

Table S4: **Ablation of pre-filtering operation** that selects $k\%$ of permutations via the semantic similarity loss. We study the influence of varying $k\%$.

k%	Near KP	Same KP	#Edits	Time (s)
0.01	61.2	34.1	4.8	0.18
0.05	66.8	34.9	4.2	0.64
0.10	68.5	35.3	3.9	1.15
0.15	68.9	35.5	3.9	1.71
0.20	69.1	35.9	3.8	2.20
1.00 (no-prefiltering)	69.2	36.0	3.8	10.82

In the future, we could further address this issue by using features from earlier layers with better localization. Alternatively, we could explore techniques which control the receptive field size of the network.

Interpretation of Same-KP. Visualization of our counterfactual explanations shows that we consistently identify class-specific and semantically matched parts (see Figure 3). However, the absolute values of the Same-KP metric in Table 2 might still seem low ($< 40\%$). There are two main reasons for this. First, we project the keypoint annotations from the query and distractor images onto the discrete spatial cells during evaluation, associating each cell with a set of keypoints. Now, keypoints lying near the cell boundaries are assigned to only one cell. At the time of evaluation, for such a keypoint, even if a neighboring cell is chosen for replacement, Same-KP metric is penalized. Secondly, a keypoint represents just a near center point of a semantic part, rather than the whole part. Hence, a wing of the bird may actually belong to two adjoining spatial cells, but the keypoint is only in one cell. Thus we also report Near-KP metric which does not suffer form these issues. In conclusion, our counterfactuals are faithful, which is also reflected in our qualitative results.

C Qualitative results

Additional qualitative examples. Figures S2-S3 show counterfactual examples generated with different methods on the CUB [58], iNaturalist-2021 Birds [55] and StanfordDogs [28] datasets. The model is ResNet-50. In each case, we highlight the single best edit. We observe that our counterfactual explanations consistently identify class-specific and semantically consistent image regions. This is opposed to other counterfactual explanations [22,59] which often replace regions of different parts.

Qualitative examples for other iNaturalist-2021 supercategories. The paper only studied the birds supercategory on iNaturalist-2021, as other supercategories are considerably harder to annotate with keypoint information (see discussion in Sec. A). In this subsection, we demonstrate that our method can

be applied to other supercategories too via qualitative results. We train separate image classifiers on the supercategories of 'Mammals', 'Insects', and 'Ray-finned Fishes', and generate counterfactual explanations using our approach. The image classifiers use a ResNet-50 model, and training follows the iNaturalist-2021 implementation detailed in Sec. A. Figure S5 shows the results for the single best edit. Again, we find that our counterfactual explanations highlight class-specific and semantically consistent image regions in the query and distractor images. In conclusion, our method applies to a broad variety of fine-grained image classification tasks.

Visualization of multiple edits. Recall that our method iteratively replaces cells between the query image and distractor image(s) until the model's decision changes. So far, we have only showed visualizations of the first cell edit. Figure S6 shows counterfactual explanations where we visualize all edits until the model's decision changes. For the purpose of visualization, we randomly selected counterfactual explanations that require three cell replacements. In each case, we observe that all edits select class-specific and semantically consistent image regions. In conclusion, our method achieves the desired result, not only for the first edit, but across all edits.

Visualization of failure cases. Figure S7 shows three failure cases, where our counterfactual explanations replace regions referring to different bird parts. The examples were generated for a ResNet-50 classifier trained on the CUB dataset. We make the following observations. First, the failure cases seem to occur for (i) odd looking birds; e.g., the query bird in Example 1 looks quite different from other birds in the CUB dataset, and (ii) for query images where certain bird parts fall outside the image; e.g., the discriminative parts of the query bird in Example 2 and Example 3 fall outside the image. Second, we observe that other methods make similar mistakes. In conclusion, the failure cases seem caused by the challenging nature of the examples, rather than being a pitfall in our method.

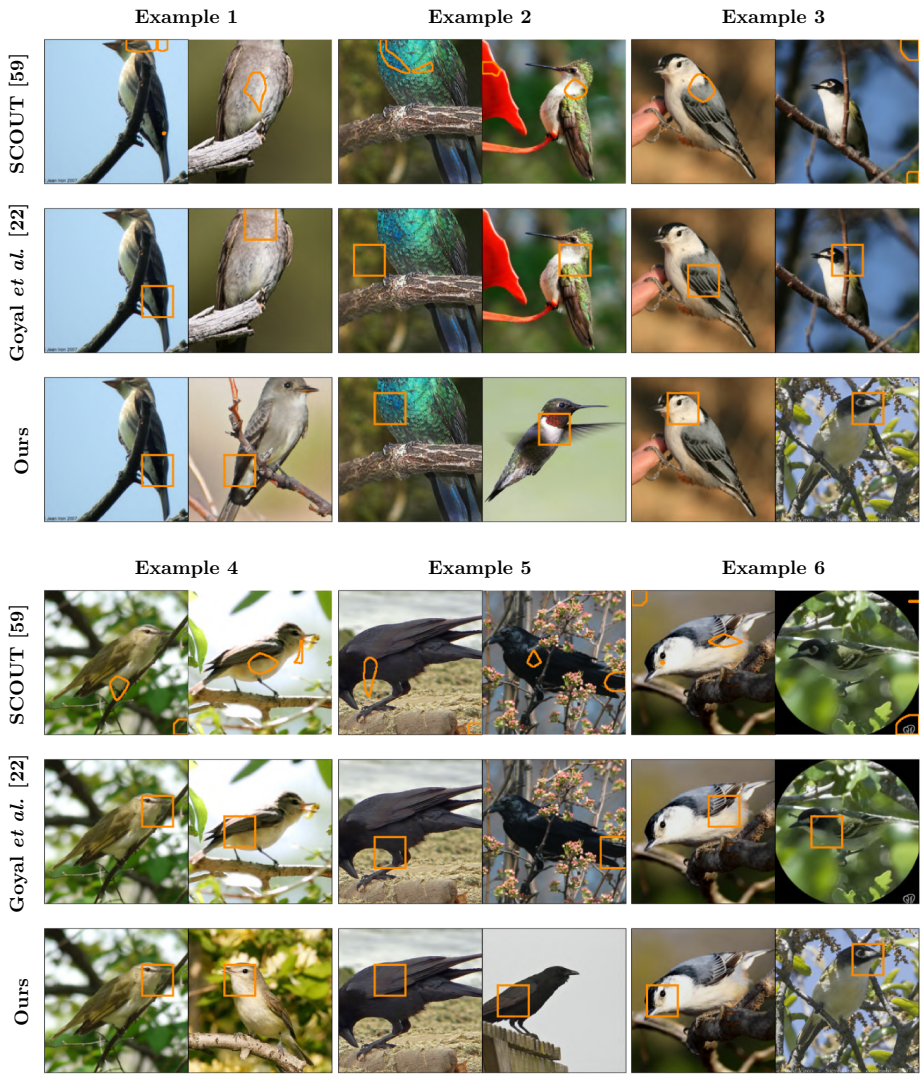

Fig. S2: **Additional qualitative results on CUB [58].** We highlight the best edit in the query image (left) and distractor image (right).

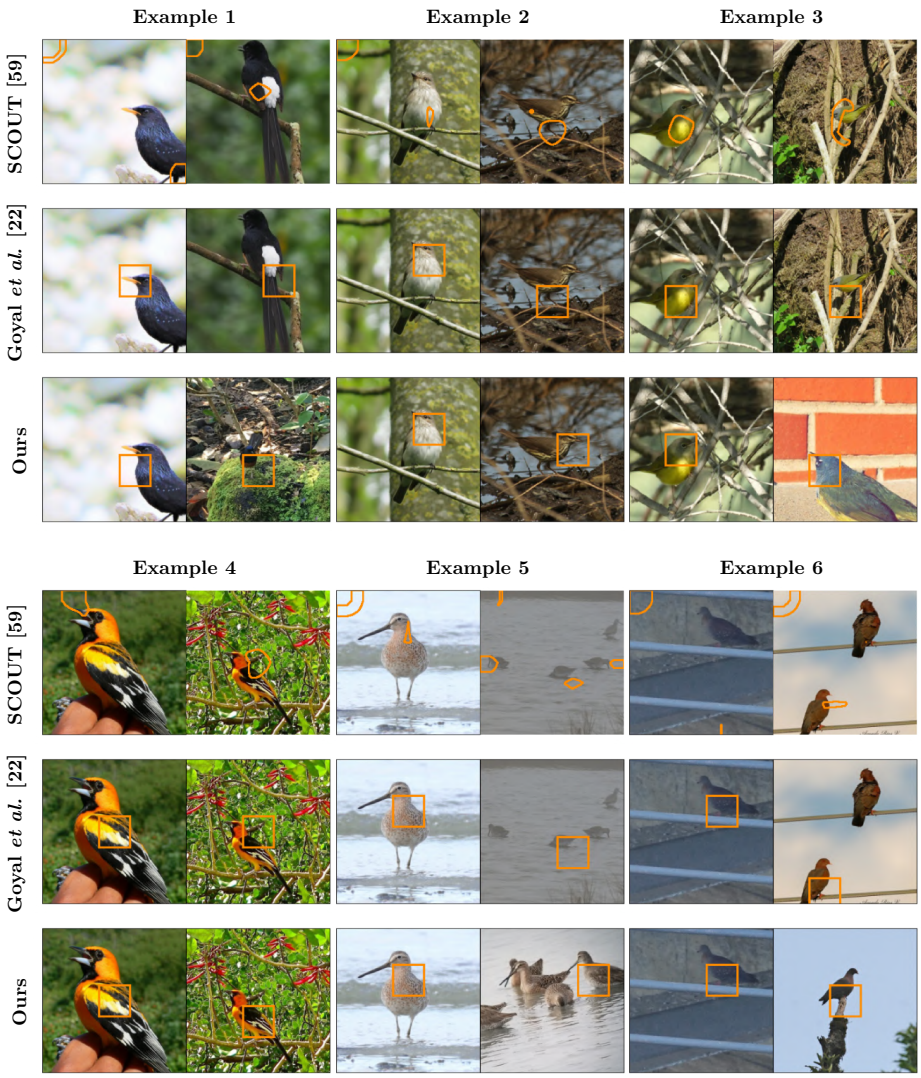

Fig. S3: **Additional qualitative results on iNaturalist-2021 Birds [55].** We highlight the best edit in the query image (left) and distractor image (right).

Fig. S4: **Additional qualitative results on StanfordDogs [28].** We highlight the best edit in the query image (left) and distractor image (right).

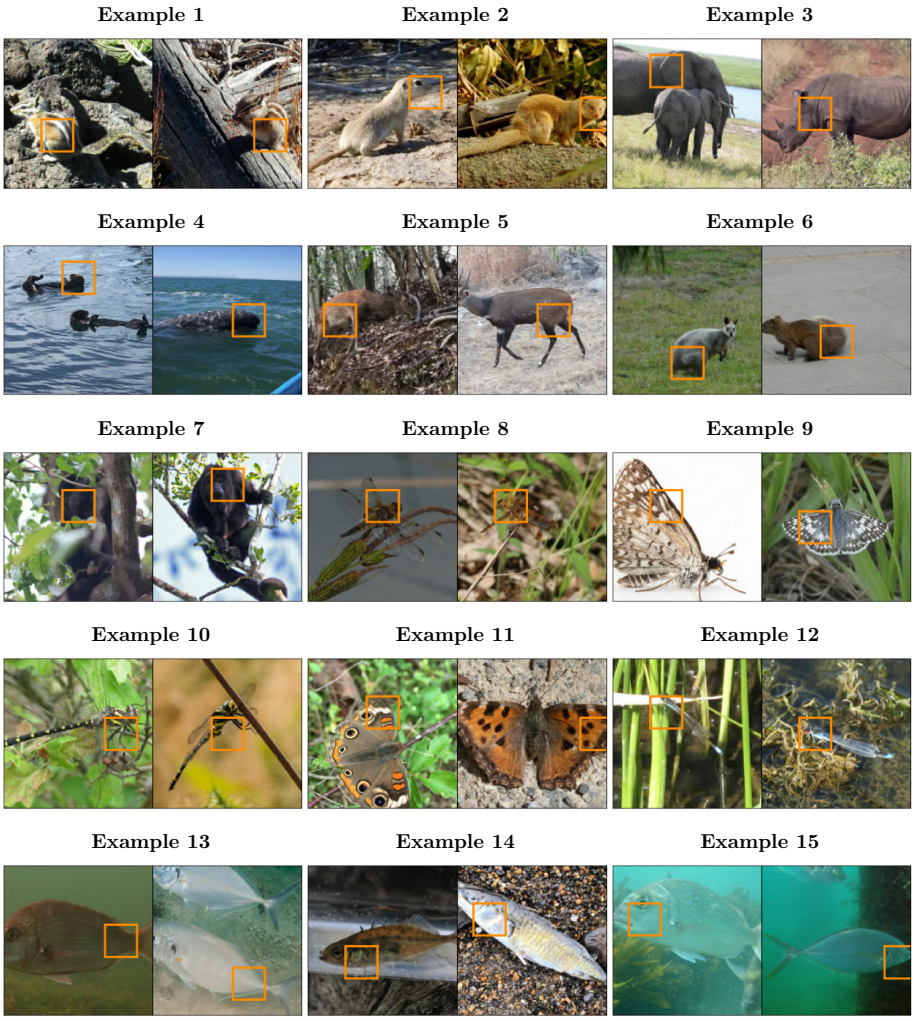

Fig. S5: **Qualitative results on other iNaturalist-2021 supercategories**. We show counterfactual explanations from our method on the following iNaturalist-2021 supercategories: 'Mammals', 'Ray-finned Fishes' and 'Insects'. We highlight the best edit in the query image (left) and distractor image (right).

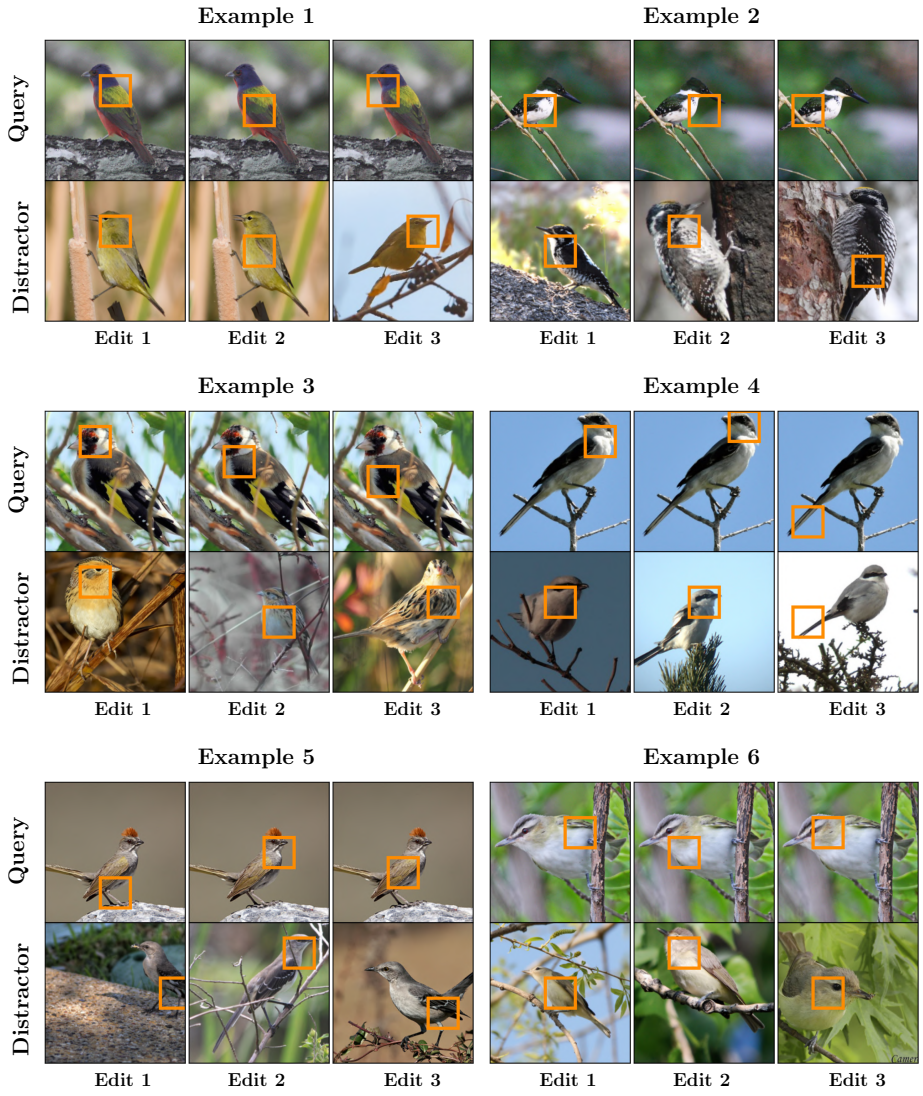

Fig. S6: **Visualization of consecutive edits on CUB.** Our counterfactual explanations iteratively replace single cells until the model's decision changes. The figure highlights these consecutive edits in the query image and distractor image(s). To generate the figure, we select counterfactual explanation that use three edits.

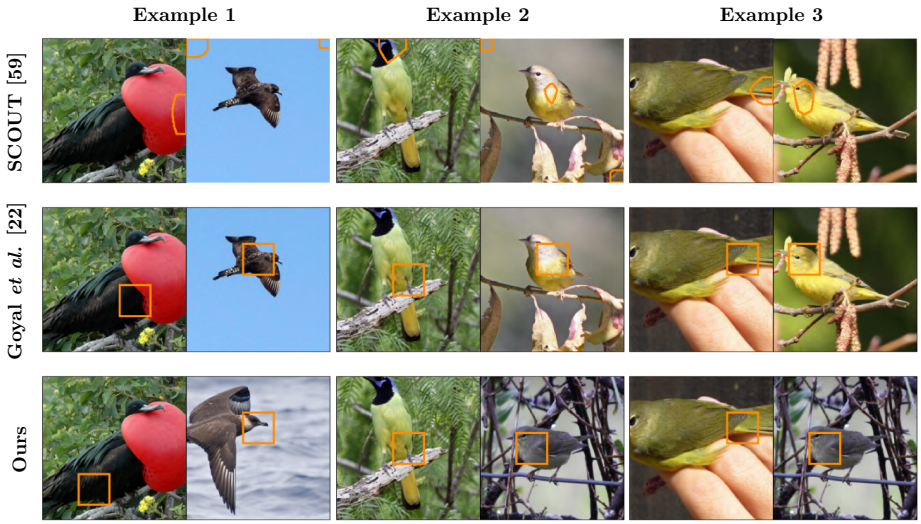

Fig. S7: **Failure cases**. We show some failure cases, where our counterfactual explanations replace regions referring to different parts.

D Computational cost analysis

In this section, we perform a complexity analysis of our approach. Additionally, we include a compute time analysis under the multi-distractor setup.

Computational complexity. We perform a back-of-the-envelope calculation of the number of multiply-add computations (MACs) in our framework. To simplify the analysis, we consider the computational cost of performing a single edit. Recall from Sec. 3 that the computational complexity of the classification loss ($\mathcal{L}_c$ in Eq. 2) and semantic loss ($\mathcal{L}_s$ in Eq. 3) can be summarized as:

$$C_{\mathcal{L}_c} = 2 \cdot C_f + h^2 w^2 \cdot C_g \tag{5}$$

$$C_{\mathcal{L}_s} = 2 \cdot C_u + h^2 w^2 \cdot C_{\text{dot}}. \tag{6}$$

We compute counterfactual explanations using the $7 \times 7 \times 512$ spatial features of the `max_pooling2d_5` layer in VGG-16 [47]. The evaluation of the classification loss $\mathcal{L}_c$ uses 318.8×10^9 MACs ($\approx 2 \cdot 15.4 \times 10^9$ for f + $h^2 w^2 \cdot 1.2 \times 10^8$ for g), while the semantic loss $\mathcal{L}_s$ computation uses 8.2×10^9 MACs ($\approx 2 \cdot 4.1 \times 10^9$ for u + $h^2 w^2 \cdot 2.0 \times 10^3$ for the dot-product in the softmax s). We conclude that the classification loss is expensive to compute due to it's quadratic dependence on the number of cells hw and the relatively high cost of evaluating the decision network $g(\cdot)$. In contrast, the semantic loss does not suffer from it's quadratic term because the dot-product operation is inexpensive to compute. In conclusion, $\mathcal{L}_s$ can be computed more efficiently compared to $\mathcal{L}_c$.

The pre-filtering operation from Sec. 3.3 relies on the fast computation of the semantic similarity loss to realize a speed-up. For example, we select the top-10% most similar cells ($k = 0.1$) according to the semantic loss, and then only consider the classification loss for this subset of cells. This reduces the complexity of $\mathcal{L}_c$ to 59.6×10^9 MACs ($\approx 2 \cdot 15.4 \times 10^9$ for f + $\underline{k} h^2 w^2 \cdot 1.2 \times 10^8$ for g). As a result, the overall computational cost is reduced from 327×10^9 to 67×10^9 MACs, meaning our framework holds a significant speed advantage over methods that compute $\mathcal{L}_c$ exhaustively [22].

This analysis does not consider the memory aspect of computing $\mathcal{L}_c$ and $\mathcal{L}_s$. It's worth noting that the similarity loss also holds an advantage in terms of memory usage compared to the classification loss. Specifically, in order to compute the classification loss for all $h^2 w^2$ permutations, we need to construct all permutations in memory. This involves the allocation of $h^2 w^2$ spatial cell matrices by replacing cells in $f(I)$ with cells from $f(I')$. In contrast, computing the similarity loss does not require to allocate $O(h^2 w^2)$ extra memory as it does not involve replacing cells. Instead, the semantic loss operates directly on the spatial feature matrices of the auxiliary model, i.e., $u(I)$ and $u(I')$.

Compute time analysis in a multi-distractor setup: Figure S8 reports the average computation time per edit (on a single V-100 GPU) as a function of the number of distractor images. We note that our method which includes a pre-filtering operation (Sec. 3.3) scales linearly with the number of distractor images and is about an order of magnitude faster compared to [22].

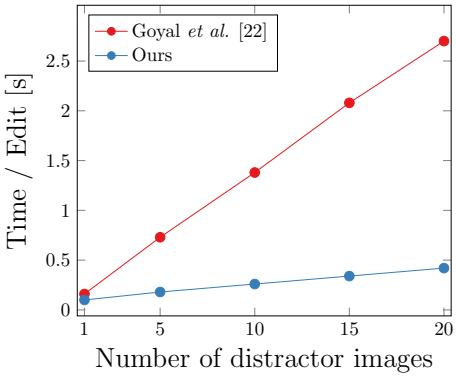

Fig. S8: Time analysis of multi-distractor setup.

E Attributes

E.1 Implementation details

We provide additional implementation details of how we add natural language attribute information to the visual counterfactual explanations in Sec. 5. Recall, the classifier is a ResNet-50 model trained to identify bird species on CUB. The spatial feature extractor f computes the $h \times w \times d$ spatial feature output of the last convolutional layer, and g performs a global average pooling operation followed by a linear classifier.

Parts detector. We reuse the CUB parts detector from Sec. A. The parts predictor is used to select the top-3 parts for the spatial cells that are being replaced in the counterfactual.

Attribute classifiers. We train linear classifiers to predict part-attributes on top of the average-pooled features from $f(\cdot)$. The part-attributes are derived from the attribute annotations used by [30]. Specifically, we only use attributes that refer to parts for which we have keypoint locations. This results in 77 attributes in total. We train linear classifiers to predict the part-attributes via a multi-class cross-entropy loss. The training uses SGD with momentum 0.9 and initial learning rate 0.04. We use batches of size 64 and train for 100 epochs. The learning rate is decayed by 10 after 70 and 90 epochs. We use weight decay 1e-6.

Interpretable basis decomposition. We perform the interpretable basis decomposition as follows. Consider a counterfactual that replaces a cell i in $f(I)$ with a cell i' from $f(I')$. First, we determine the attributes that should be used for the decomposition. To this end, we take the union of detected parts in cell i and i' first, and then select the attributes that are associated with the detected parts, e.g., if one of the parts is 'wing' we select attributes like 'has_wing_color::blue'. We then apply the algorithm from [64] to decompose the weights of the linear

layer in g in terms of the selected attribute classifiers. The decomposition is performed for the query $f(I)$ and counterfactual $f(I^*)$.

Additional examples. Figure 7 shows additional examples, where our method succeeds in adding attribute information to our visual counterfactual explanations. In each case, the returned attribute belongs to class c but not to class c', or vice-versa. Thus, the returned attributes are discriminative of the classes c and c'.

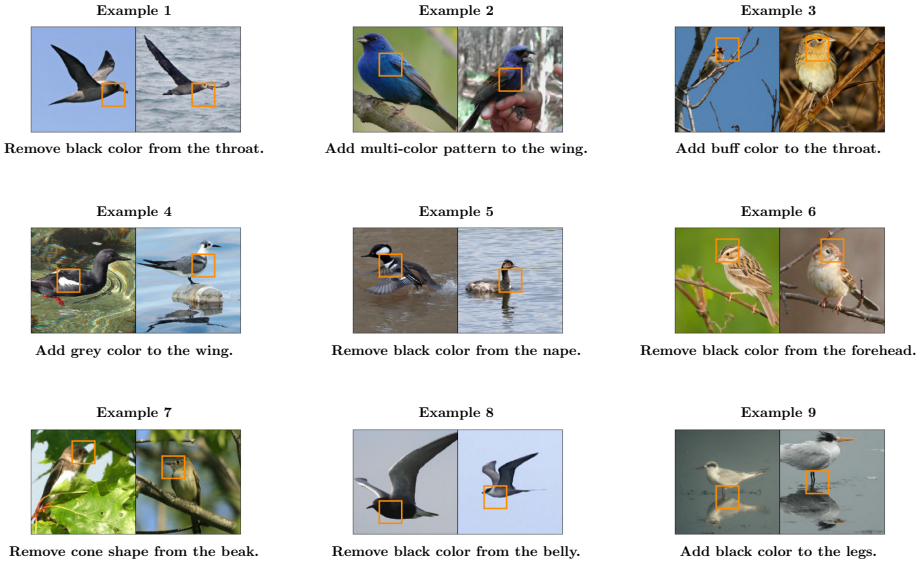

Fig. S9: **Language-based counterfactuals.** We identify the attribute that is most important for chaenging the model's decision.

F Licenses

We include the licenses for images from iNaturalist used in our visualizations.

Table S5: Authors and Creative Commons Copyright notice for images in Figure S3. lauriekoepke: CC BY-NC 4.0, walterflocke: CC BY-NC 4.0, leo_v: CC BY-NC 4.0, Herbert Herbinia: CC BY-NC 4.0, leptim: CC BY-NC 4.0, leptim: CC BY-NC 4.0, nanorca13: CC BY-NC 4.0, toucan55: CC BY-NC 4.0, Jim Brighton: CC BY-NC 4.0, wildmouse3: CC BY-NC 4.0, Will Richardson: CC BY-NC 4.0, Amado: CC BY-NC 4.0, Esteban Munguia: CC BY-NC 4.0, jamesbeat: CC BY-NC 4.0, jamesbeat: CC BY-NC 4.0, N. Mahathi: CC BY-NC 4.0, ddun: CC BY-NC 4.0, John G. Phillips: CC BY-NC 4.0.

Table S6: Author and Creative Commons Copyright notice for images in Figure S5. Daniel George: CC BY-NC 4.0, thehaplesshiker: CC BY-NC 4.0, tiyumq: CC BY-NC 4.0, Rohit Chakravarty: CC BY-NC 4.0, ungerlord: CC BY-NC 4.0, dushenkov: CC BY-NC 4.0, jenhenlo: CC BY-NC 4.0, amniotasmarinos6: CC BY-NC 4.0, Lawrence Troup: CC BY-NC 4.0, Andrew Deacon: CC BY-NC 4.0, Torbjorn von Strokirch: CC BY-NC 4.0, 金翼白眉: CC BY-NC 4.0, Andrés Matos: CC BY-NC 4.0, Christoph Moning: CC BY-NC 4.0, asandlermd: CC BY-NC 4.0, Eric Giles: CC BY-NC 4.0, Paul Cools: CC BY-NC 4.0, Laura Kimberly: CC BY-NC 4.0, sterling: CC BY 4.0, elisabraz: CC BY-NC 4.0, Tom Warnert: CC BY-NC 4.0, Don Loarie: CC BY 4.0, Marlo Perdicas: CC BY 4.0, luca tringali: CC BY-NC 4.0, pmmullins: CC BY-NC 4.0, Erik Schlogl: CC BY-NC 4.0, sea-kangaroo: CC BY-NC-ND 4.0, Jason Grant: CC BY-NC 4.0, Donald Hobern: CC BY 4.0, Richard Ling: CC BY-NC-ND 4.0, BeachBumAgg: CC BY-NC 4.0, Tony Strazzari: CC BY-NC 4.0.