Visual Counterfactual Explanations Using Semantic Part Locations

Florence Böttger[1], Tim Cech[2][a], Willy Scheibel[1][b], and Jürgen Döllner[2][c]
[1]*University of Potsdam, Digital Engineering Faculty, Hasso Plattner Institute, Germany*
[2]*University of Potsdam, Digital Engineering Faculty, Germany*
florence.boettger@student.hpi.uni-potsdam.de, tcech@uni-potsdam.de, willy.scheibel@hpi.uni-potsdam.de, doellner@uni-potsdam.de

Keywords: Counterfactuals, Explainable Artificial Intelligence, Convolutional Neural Networks

Abstract: As machine learning models are becoming more widespread and see use in high-stake decisions, the explainability of these decisions is getting more relevant. One approach for explainability are counterfactual explanations, which are defined as changes to a data point such that it appears as a different class. Their close connection to the original dataset aids their explainability. However, existing methods of creating counterfacual explanations often rely on other machine learning models, which adds an additional layer of opacity to the explanations. We propose additions to an established pipeline for creating visual counterfacual explanations by using an inherently explainable algorithm that does not rely on external models. Using annotated semantic part locations, we replace parts of the counterfactual creation process. We evaluate the approach on the CUB-200-2011 dataset. Our approach outperforms the previous results: we improve (1) the average number of edits by 0.1 edits, (2) the keypoint accuracy of editing within any semantic parts of the image by an average of at least 7 percentage points, and (3) the keypoint accuracy of editing the same semantic parts by at least 17 percentage points.

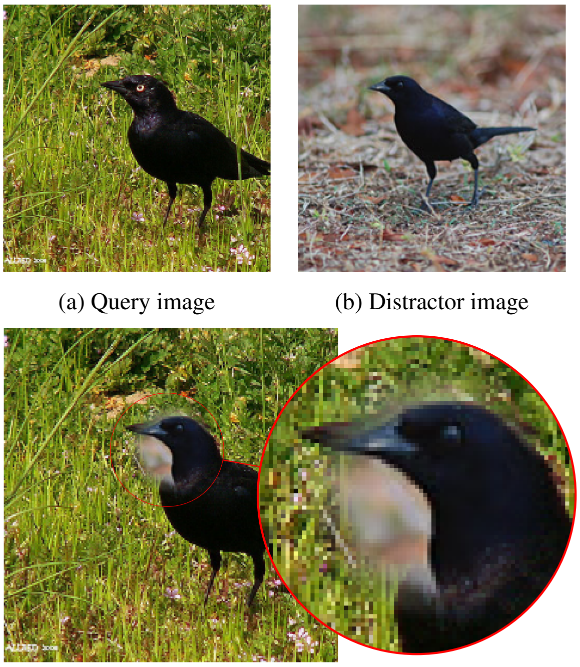
1 INTRODUCTION

In recent years, the impact of machine learning models has increased in many high-stakes areas, e.g., the medical field (Chen et al., 2021) or autonomous driving (Schwarting et al., 2018). The topic of ethics in machine learning for purposes such as bias mitigation (Wachter et al., 2020) is becoming more prevalent too (Zhang et al., 2021). However, most models used in high-stakes decisions are still regarded as black boxes with little to no inherent means of explaining the models' outputs (Castelvecchi, 2016). Given these high stakes, it is essential that machine learning models are explainable. In particular, continued use of machine learning models needs to comply with laws, e.g., the "right to explanation" from the European Union's General Data Protection Regulation (Wachter et al., 2018).

Solving these problems of explainability is researched within the field of *Explainable AI* (XAI). While explanations can be generated on models working with raw data, our focus is on visual explanations. There exist multiple means of generating explanations on visual data: some are external to the data, such as captions (Hendricks et al., 2016) or visual questions answering, which reveals information by answering given questions about the decisions (Goyal et al., 2017; Chen et al., 2020). Others are more closely related to the data, such as saliency maps, which try to show the most pertinent parts of an input image (Simonyan et al., 2014; Petsiuk et al., 2021). We are focusing on visual counterfactual explanations, which are closely related to the input images as we are directly editing them to generate the explanations.

A *Counterfactual Explanation* is defined as an edit of an input image of a given class based on another image of another class. The initial image and its class are referred to as the query image and the query class, while the image that is used for editing and its class are the distractor image and the distractor class. The result of this is a counterfactual image, which is based on the query image while being detected as the distractor class. For example, a decision of a model to identify numbers could be explained by displaying what would need to be changed to instead be detected as a different number, so a query image showcasing a "5" should be modified to instead be detected as

[a] https://orcid.org/0000-0001-8688-2419
[b] https://orcid.org/0000-0002-7885-9857
[c] https://orcid.org/0000-0002-8981-8583



(a) Query image

(b) Distractor image

(c) Merged counterfactual image

Figure 1: Visual representation of a counterfactual created by editing a query image with parts of a distractor image using a vignette overlay. The head of the distractor image is edited over that of the query image. Note that the vignette overlay is used to increase visual clarity and is not an exact representation of the actual counterfactual creation process and has not been tested in studies.

a distractor class of “6”. By creating a specific set of changes from a given data point, the explanation is intrinsically connected to the data, and therefore it should be understandable to anyone who understands the original data. However, this adherence to an original data point also limits the scope of possible explanations, as the explanation should not stray too far from the original, or present impossible or nonsensical changes. If a counterfactual for the number detection example suggested replacing the entire image, for example, that would not be a helpful suggestion, as it would lose all connections to the query image.

A visual representation of a counterfactual can be seen in Figure 1. A counterfactual has been created from two images by editing part of the distrator image over the query image, showing how the query image subject would need to look different in order to be detected as the distractor class. Note that the vignette overlay used here is for presentation purposes and is not an exact representation of the actual process. As opposed to other methods of explanations, which tend to be external to the actual data at hand, counterfactuals tend to have a much closer connection to the data. Counterfactual explanations are strongly correlated to the input; therefore, they are easily understandable. In particular, a counterfactual explanation can support causal reasoning, as it displays a change in state and how it would affect the decision (Pearl, 2000).

One important aspect of counterfactual creation is comprehensibility: if an edit is incomprehensible to an observer, for example because it switched a bird’s head with the legs of another, the explanation would not be understandable, as it does not present a realistic alternative. Therefore, it is important to encourage that the proposed counterfactual explanations obey this sensibility. Additionally, explanations generated using external models add an additional layer of opacity that does not exist when the explanation itself is more inherently explainable.

In this paper, we expand upon existing approaches by Goyal et al. (2019) and Vandenhende et al. (2022) for creating visual counterfactual explanations on the CUB-200-2011 dataset (Wah et al., 2011) by using pre-annotated semantic parts data. We outline how we add an additional loss summand based on annotated date of semantic parts. We also perform a comparison of different edits as well as an extensive optimization of the hyperparameters.

The remainder of this work is structured as follows. We will first go over the related work that we are building upon. Then we outline our approach, briefly going over the dataset before explaining the parts loss, an additional loss term that uses semantic part locations to prefer edits between features containing the same semantic parts. We also add a means to add tolerance to this process, as well as a way to simplify the semantic parts in crowded areas. After this, we explain the experiment setup, going over our implementation before elaborating on the relevant hyperparameters. We also explain the metrics that we optimize for, then briefly outline our optimization algorithm. We perform two experiments, one to determine viability of different hyperparameters and one to determine the best values for the hyperparameters that were deemed promising from the first experiment, then briefly analyze our approach’s runtime. After this, we discuss our results, putting them into perspective and performing comparisons with the previous work, and evaluate possible threats to validity. Lastly, we provide our conclusions and detail possible future work.

2 RELATED WORK

Counterfactual explanations have found widespread use in explaining machine learning models. Mothilal et al. (2020) have developed a counterfactual creation engine for raw information datasets that can create sets of diverse counterfactuals. Gomez et al. (2020) have created a visual analytics tool to visually represent counterfactuals to an information dataset. Boreiko et al. (2022) have generated visual counterfactuals by performing detailed edits to change the subject of the image. As these approaches perform more complex operations, ensuring semantic context becomes a challenge, whereas the approach we are using performs simpler operations that inherently keep more of the context intact.

As opposed to more finely detailed means of counterfactual creation, Goyal et al. (2019) constructed a visual counterfactual explanation by pasting a subset of features from the distractor image onto the query image, resulting in a counterfactual. Because the total number of possible edits arising from all possible combinations of features to edit between was too big to reasonably compute, they proposed a greedy sequential search to determine the individual best edits over multiple edit steps by optimizing a classification loss term.

Vandenhende et al. (2022) expanded upon this framework by adding a semantic consistency loss term to the existing greedy sequential search. For this they used an auxiliary model that determined semantic consistency between spatial cells. They then combined this loss with the classification loss and optimized for this combined loss. Additionally, they also searched over multiple distractor images to find the best edit instead of just one. They optimized for this by only selecting a smaller number of distractor images chosen by semantic consistency with the query image.

Because the semantic consistency approach utilizes an auxiliary model to create its explanation, someone wishing to fully understand their explanations would also need to understand the auxiliary model. By adding this additional layer of opacity, a comprehensive explanation would also need to explain the auxiliary model. Rudin (2019) argues that inherently explainable models should be used instead of black box models when the quality is sufficient. Our explanation is inherently understandable, as it relies on the annotated parts and matches them between query and distractor features. As such, no additional explanation work is needed to fundamentally understand our explanation.

3 APPROACH

Our approach builds upon the previous work done by Goyal et al. (2019) and Vandenhende et al. (2022). Ensuring consistency within edits is an important part of creating understandable edits. For example, an edit that switches the head of a bird with its wing is likely to look nonsensical and is therefore ill-suited as a counterfactual explanation. We can prevent such edits and therefore improve their accuracy by adding an additional loss term based on semantic part locations as well as adding more terms on how said loss is calculated by implementing a tolerance as well as the option to simplify semantic parts.

3.1 The Dataset

We use the CUB-200-2011 dataset (Wah et al., 2011) for our experiments as it has already been used in previous works by Goyal et al. (2019) and Vandenhende et al. (2022) in creating visual counterfactuals. It is also popular among other visual applications of artificial intelligence, such as the image captioning done by Hendricks et al. (2016) or the CutMix approach to training samples proposed by Yun et al. (2019).

The dataset contains 11 788 images of 200 bird species. It is uniquely suitable to our approach, as it includes part locations for 15 bird parts, as seen in Figure 2. We are using these part locations in order to improve the accuracy of the explanation. According to the dataset split, 5 794 of the 11 788 images within the dataset are used for testing, while the others are used for training.

3.2 Parts Loss

When creating a counterfactual, the basis consists of an image I of a query class $c \in \mathcal{C}$ and an image I' of a distractor class $c' \in \mathcal{C}$. The goal is to edit I such that the result is detected as the class c'. In order to detect the classes as seen by a given network, a feature extractor function $f : \mathcal{I} \rightarrow \mathbb{R}^{hw \times d}$ is used, where h and w are the spatial dimensions of the features and d the number of channels. A decision network $g : \mathbb{R}^{hw \times d} \rightarrow [0,1]^{|\mathcal{C}|}$ assigns each feature matrix to a set of probabilities of the matrix belonging to each given class.

We perform edits as first defined by Goyal et al. (2019): we permute the features of I' and then edit only a subset of those features over the features of I (gating). Consider the permutation matrix $P \in \mathcal{P}$, where $\mathcal{P}$ is the set of all $hw \times hw$ permutations over the feature maps, and the gating vector $\mathbf{a} \in \{0,1\}^{hw}$, where $a_i = 1$ means that a given feature will be added to the counterfactual. A counterfactual I' is then exactly defined by P and $\mathbf{a}$:

$$f\left(I^{*}\right) = (\mathbb{1} - \mathbf{a}) \circ f(I) + \mathbf{a} \circ P f\left(I'\right). \qquad (1)$$

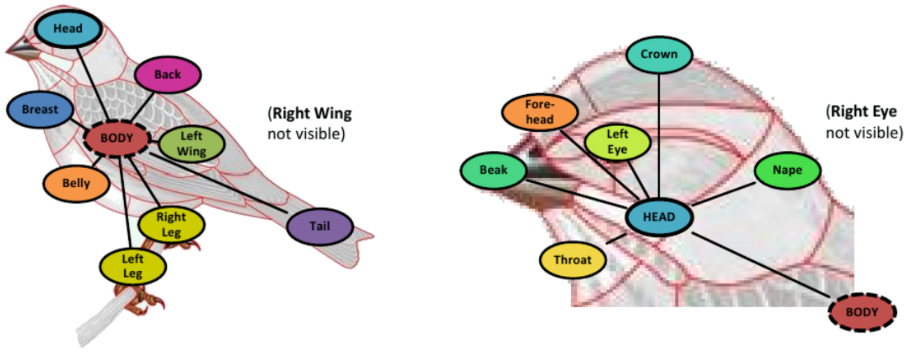


Figure 2: The 15 parts labeled within the CUB-200-2011 dataset, as created by Wah et al. (2011).

Table 1: Different types of sub-losses used to calculate total loss. Table is sorted by source date.

Source	Loss	Formula
Goyal et al. (2019)	$\mathcal{L}_c\left(I,I'\right)$	$g_{c'}\left(\left(\mathbb{1}-\mathbf{a}\right)\circ f(I)+\mathbf{a}\circ Pf\left(I'\right)\right)$
Vandenhende et al. (2022)	$\mathcal{L}_s\left(u\left(I\right)_i,u\left(I'\right)_j\right)$	$\dfrac{\exp\left(u(I)_i\cdot u\left(I'\right)_j/\tau\right)}{\sum_{j'\in u(I')}\exp\left(u(I)_i\cdot u\left(I'\right)_{j'}/\tau\right)}$
Proposed	$\mathcal{L}_p\left(r\left(I\right)_i,r\left(I'\right)_j\right)$	$\max\limits_{0\leq k<p}\left(r\left(I\right)_{i,k}\cdot r\left(I'\right)_{j,k}\right)$

As the dataset we use contains part locations, we locate the feature within the $h \times w$ feature grid that each part's spatial location belongs to. We define

$$r : \mathcal{I} \rightarrow \{0,1\}^{hw\times p}, \qquad (2)$$

where p is the number of distinct parts in the dataset, and $r(I)_{i,k} = 1$ if and only if part k is contained within cell i of the feature representation of I. Our parts loss $\mathcal{L}_p$ is defined in Table 1, where i is the given feature cell of the query image, and j the cell of the distractor image. $\mathcal{L}_p$ is 1 if cell i of the query image and cell j of the distractor image contain a common class between them, otherwise it is 0.

Our loss is combined with the loss terms by Goyal et al. (2019) and Vandenhende et al. (2022) as outlined in Table 1 (with u being the auxiliary model and τ the temperature), leading to the following total loss:

$$\begin{aligned}\mathcal{L}\left(I,I'\right) &= \log\mathcal{L}_c\left(I,I'\right)\\ &+\lambda_1\cdot\log\mathcal{L}_s\left(\mathbf{a}^T u(I),\mathbf{a}^T Pu\left(I'\right)\right)\\ &+\lambda_2\cdot\mathcal{L}_p\left(\mathbf{a}^T r\left(I\right),\mathbf{a}^T Pr\left(I'\right)\right), \qquad (3)\end{aligned}$$

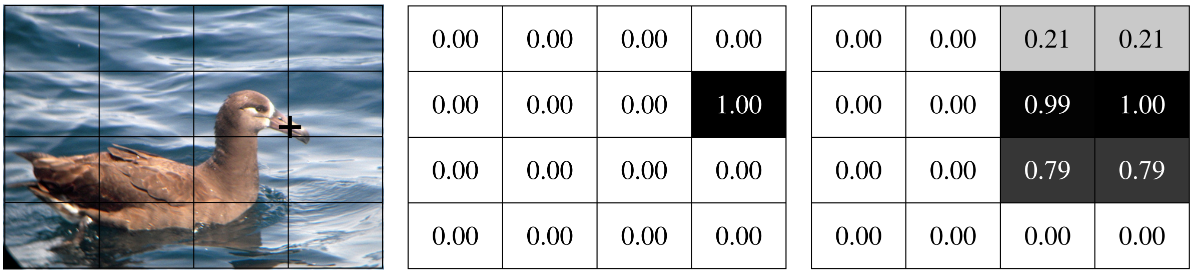
where λ_1 and λ_2 are hyperparameters balancing the individual losses. We keep the use of multiple distractor images. In the case of n different distractor images, the semantic class parts for the distractor images have a combined shape of $\{0,1\}^{nhw\times p}$.

3.3 Tolerance for Parts Loss

By default, the semantic class parts for each image consist of a binary matrix, where a part either exists within a feature or not. However, the specific boundaries between features are arbitrary with relation to the image. Therefore, we implement a tolerance system to make the result less dependent on said layout.

We define $r_D : \mathcal{I} \rightarrow [0,1]^{hw\times p}$ similarly to r in Equation 2. Just as with r, a value of 1 indicates that a part is present within a feature. However, sometimes parts are close to the boundary between features. The previous approach would count them to be wholly within one feature, even though they are very close to another. As such, a binary loss is a poor representation of the reality of the parts.

We take this into account by returning a fractional number within $[0,1]$ if the part is close to the given feature. Given an euclidean distance of d between a class part and a feature cell, the value is $\max\left(0, 1-\frac{d}{D}\right)$, with D being set as the maximum distance that a cell can have from a part. The unit for both values is set as the dimension of a cell, so if e.g. $D = 1$, the gradient takes effect up to 1 cell away from the part location. As such, the previously defined function r is equivalent to r_0. A comparison of the approaches can be seen in Figure 3.



(a) Image with part location

(b) Matrix for $D = 0$

0.00	0.00	0.00	0.00
0.00	0.00	0.00	1.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00

(c) Matrix for $D = 1$

0.00	0.00	0.21	0.21
0.00	0.00	0.99	1.00
0.00	0.00	0.79	0.79
0.00	0.00	0.00	0.00

Figure 3: 4×4 feature matrices for values of r_D for a given part with different tolerances.

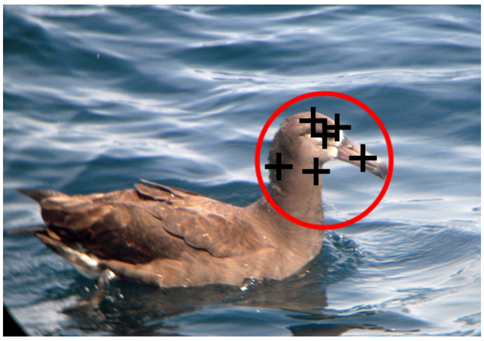


Figure 4: Locations of all head parts in an image of a black-footed albatross from the CUB-200-2011 dataset.

We use r_D in place of r for the calculation of $\mathcal{L}_p$. In that case, $\mathcal{L}_p\left(r_D(I)_i, r_D(I')_j\right)$ is also no longer a binary result in $\{0,1\}$, but can take any value in $[0,1]$.

3.4 Simplifying Semantic Parts

The annotated part locations may be in different degrees of detail based on the specific dataset and the purposes said annotation set out to do. For example, the CUB-200-2011 dataset contains many annotations for parts within a bird's head area, as seen in Figure 4. Such a level of detail is less useful when we are considering the rougher divisions of the feature grid we are using. In fact, it might even detract from the result because the granularity is so much finer than what we are using. Therefore we are grouping all parts that are categorized under "head" within Wah et al. (2011) as the same semantic class.

We use a mapping function $m : [0,1]^{hw\times p} \rightarrow [0,1]^{hw\times p'}$, where p' is the new number of parts. Using a given $p \times p'$ permutation matrix M, we calculate

$$m(R)_{i,j} = \max_{0\leq k<p}\left(R_{i,k}\cdot M_{k,j}\right), \tag{4}$$

where $R \in [0,1]^{hw\times p}$ is a parts matrix. We then use $m(r(I))$ and $m(r(I'))$ in place of $r(I)$ and $r(I')$, respectively, when calculating $\mathcal{L}_p$.

By performing this simplification we are reducing the total number of semantic parts from the original 15 down to 7 by assigning all classes belonging to the head—those being the beak, the crown, the forehead, the eyes, the nape, and the throat—as one "head" class. Vandenhende et al. (2022) only simplified the number of different parts down to 12 by unifying left and right counterparts of body parts.

4 EXPERIMENT SETUP

We are performing a computational experiment where we create a counterfactual for each of the 5 794 samples within the CUB-200-20111 test dataset. The input images within the dataset are of size 224×244 pixels, while the feature map has dimensions of 7×7, meaning that $h = w = 7$. We use the approach outlined in section 3 to generate a series of edits by choosing the best edit via $\mathcal{L}_p$, as seen in Equation 3, until the given network predicts the edited features to belong to the distractor class. For the sake of creating counterfactuals between similar semantic classes, we select the distractor class based on the most similar class to the query class using a confusion matrix included with the dataset. We then randomly select a given number of distractor images to use for counterfactual creation. Then we save the counterfactual and repeat this process for the next one until all of the data has been processed.

4.1 Implementation

To implement the method outlined above, we are building upon the Python implementation created by Vandenhende et al. (2022). Specifically, we are adding our additional loss to the sequential greedy search and expanding upon the program by implementing the necessary hyperparameters as well as editing the parts map to take tolerance and simplified parts into account. To compare our results to

Vandenhende et al. (2022), we are using the same pretrained VGG-16 (Simonyan and Zisserman, 2015) and ResNet-50 (He et al., 2016) networks that they have used.

We perform this experiment using parallelized computing on four servers (512 GiB RAM each), using Python version 3.8.13. The code written for this project as well as additional data are available online[1].

Our results are entirely deterministic, as we are performing a greedy sequential search. If the hyperparameters are identical and the same network is used, the calculations of the losses for each individual edit will also be identical. As such the same edits are chosen, and the evaluation is therefore also identical.

4.2 Hyperparameters

For our initial optimization run, we evaluate several hyperparameters to determine which ones are the most promising. As such, we keep the hyperparameter λ_1 as utilized when weighing $\mathcal{L}_c$ in the calculation of the total loss term as first proposed by Vandenhende et al. (2022). In order to still limit the total number of hyperparameters and yield more substantive results, we are keeping the temperature hyperparameter τ from Table 1 fixed at the best value determined by Vandenhende et al. (2022), being $\tau = 0.1$.

We are also adding three additional hyperparameters: λ_2 from $\mathcal{L}$ in Equation 3 is used to determine the weight of the parts loss $\mathcal{L}_p$. D is be used to balance the tolerance for the semantic class loss using r_D, as outlined in subsection 3.3. Lastly, we use a binary hyperparameter $S \in \{\text{True, False}\}$ to determine whether we further simplify the semantic parts from 12 parts to 7. We set $0 \leq \lambda_1 \leq 2$, $0 \leq \lambda_2 \leq 10$, $0 \leq D \leq 3$ and $S \in \{\text{True, False}\}$ as the optimization ranges for the hyperparameters.

4.3 Optimization Metrics

We use the evaluation metrics initially used by Goyal et al. (2019) and Vandenhende et al. (2022), those being the average number of edits as well as the keypoint (KP) metrics. Let m be the size of the dataset we are editing over, and $E_k \in \mathbb{N}, 0 \leq k < m$ the number of edits on the k-th iteration. Then the average amount of edits E is

$$E = \frac{\sum_{k=0}^{m-1} E_k}{m}. \tag{5}$$

For the keypoint metrics, consider a single given edit $e = (i, j) \in (\mathbb{N}, \mathbb{N}), 0 \leq i, j < hw$ consisting of a pair of a query cell i from I and a distractor cell j from I' to edit. Then the Near-KP metric of that edit $KP_N : \mathbb{N}^{hw \times hw} \to [0, 1]$ is defined as the average number of these cells which contain at least one semantic part within them:

Table 2: Spearman correlation between different metrics over initial testing using VGG-16 and 400 iterations. Note that for clarity, $-E$ is used in place of E as we wish to minimize E while maximizing the KP values.

	$-E$	$KP_{S,N}$	$KP_{S,S}$	$KP_{A,N}$	$KP_{A,S}$
$-E$	1	−0.44	−0.60	0.01	−0.26
$KP_{S,N}$	−0.44	1	0.89	0.86	0.91
$KP_{S,S}$	−0.60	0.89	1	0.67	0.90
$KP_{A,N}$	0.01	0.86	0.67	1	0.89
$KP_{A,S}$	−0.26	0.91	0.90	0.89	1

$$KP_N(i,j) = \frac{1}{2} \cdot \mathbb{1}\left(\bigcup_{k=0}^{p-1} r(I)_{i,k} = 1\right) + \frac{1}{2} \cdot \mathbb{1}\left(\bigcup_{k=0}^{p-1} r\left(I'\right)_{j,k} = 1\right). \tag{6}$$

The Same-KP metric $KP_S : \mathbb{N}^{hw \times hw} \to \{0, 1\}$ is defined as being 1 if the edit cells share a common semantic part between them and 0 otherwise:

$$KP_S(i,j) = \mathbb{1}\left(\bigcup_{k=0}^{p-1} r(I)_{i,k} = r\left(I'\right)_{j,k} = 1\right). \tag{7}$$

For both of these individual edit metrics, we define both a variant defined as the total average of the given metric for all edits across each iteration and a variant defined as the average of only the single first edit across each iteration. This results in five total metrics: the average number of edits E, the Near-KP over a single edit $KP_{S,N}$, the Near-KP over all edits $KP_{A,N}$, the Same-KP over a single edit $KP_{S,S}$ and the Same-KP over all edits $KP_{A,S}$.

We specifically focus on two metrics for the optimization: the average number of edits E and the Same-KP over all edits $KP_{A,S}$. These metrics were chosen as early tests have shown that the different keypoint metrics largely correlate with one another, while edits show little or even opposed correlation with the keypoint metrics, as seen in Table 2. Note that we are using $-E$ instead of E in the table, as we wish to minimize E while maximizing the KP values.

As E is inversely correlated with all of the other metrics that we considered, it was chosen as one of our main metrics. From the KP metrics, we choose the Same-KP over all edits $KP_{A,S}$. This metric is the strictest, as early tests have shown that $KP_{A,S}$ yields

[1] https://doi.org/10.5281/zenodo.8056313

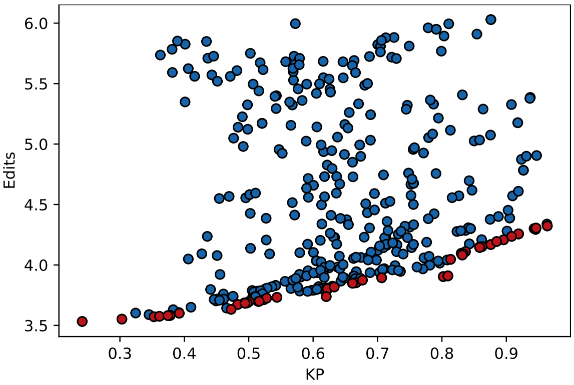


Figure 5: Scatter plot of all 400 optimization iterations from initial testing using VGG-16. The Pareto front is highlighted in red.

the lowest average out of all keypoint metrics. Following, we refer to the $KP_{A,S}$ metric as KP.

4.4 Optimization Algorithm

We use the optimization framework Optuna (Akiba et al., 2019) in order to optimize the given hyperparameters. This framework lets the user have freedom in constructing the search space (define-by-run principle). We slot the existing structure that we expanded from the work by Vandenhende et al. (2022) into an optimization function without requiring drastic changes or forgoing the existing functionality of externally-defined hyperparameters. We then use Optuna to optimize the existing functions, utilizing a tree-structured Parzen estimator (Bergstra et al., 2011). We chose this estimator due to its good performance even for multivariate optimization.

5 EXPERIMENTS

In order to showcase our results, we perform two experiments. Experiment 1 is performed to analyze the quality of our four selected hyperparameters by considering their correlation with our two metrics. From among those we then select the hyperparameters that have a positive correlation with at least one of the metrics. Experiment 2 is performed with just these hyperparameters with the goal of determining the total quality of our results. We are also comparing our results with those set by Goyal et al. (2019) and Vandenhende et al. (2022).

5.1 Experiment 1

We begin by performing an optimization using VGG-16 and all four hyperparameters (λ_1, λ_2, D, and S) over 400 iterations to determine the viability of each of the hyperparameters. A scatter plot of the results from this test can be seen in Figure 5. In order to determine the impact of each hyperparameter, we determine the Spearman correlation (Spearman, 1904) of each hyperparameter on each of the optimization metrics. The Spearman correlation is defined as the correlation between the ranks of a set of values, meaning it measures monotony. The reason we utilize the Spearman correlation is that we mainly wish to determine the hyperparamters' impact on the rank of the values so we can determine their general impact. We also record the corresponding p-values for the null hypothesis that the hyperparameter has no correlation with the metric.

Table 3: Spearman correlation and related p-values between the hyperparameters and the optimization metrics over the first experiment using VGG-16.

	r_{KP}	$p(r_{KP})$	r_E	$p(r_E)$
λ_1	-0.03	56%	0.91	$< 0.1\%$
λ_2	0.47	$< 0.1\%$	0.19	$< 0.1\%$
D	-0.68	$< 0.1\%$	-0.35	$< 0.1\%$
S	-0.23	$< 0.1\%$	0.08	10.1%

As can be seen from Table 3, λ_1 has a very slight negative correlation on KP, while also having a very direct correlation with a higher E. λ_2, meanwhile, has a rather clear correlation with a higher KP, while slightly increasing E. D has a negative correlation on KP, while correlating with fewer edits E. Lastly, S appears to have negative correlation on KP while appearing to have little correlation to E.

Given these correlation values, we can determine that λ_2 and D each positively correlate with one of our metrics: while λ_2 correlates with better KP, D correlates with fewer edits. The other two metrics which we tested in these experiments, λ_1 and S do not display a positive correlation with either of our metrics. Therefore, Experiment 2 will be performed using the hyperparameters λ_2 and D with both the VGG-16 and ResNet-50 models.

5.2 Experiment 2

For optimizing the results using only the hyperparameters λ_2 and D, we begin by queuing up optimization trials for the extreme values of $\lambda_2 = 0$, $D = 0$ and $\lambda_2 = 10$, $D = 0$. Early tests have shown that these values provide extreme values for the results, and choosing them for the initial optimization trials sets a range for the optimization, as well as providing promising initial values. A scatter plot of the 400 optimization iterations for this experiment using VGG can be seen in Figure 6. When visualizing the hyperparameters

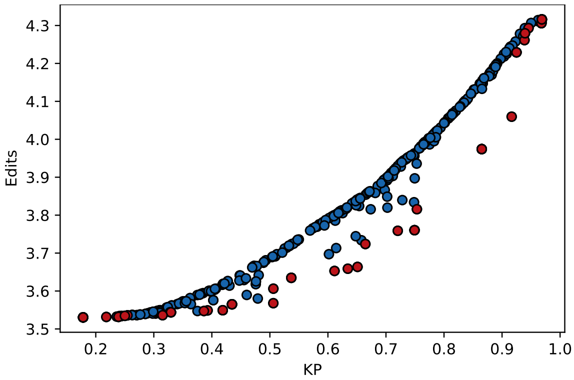


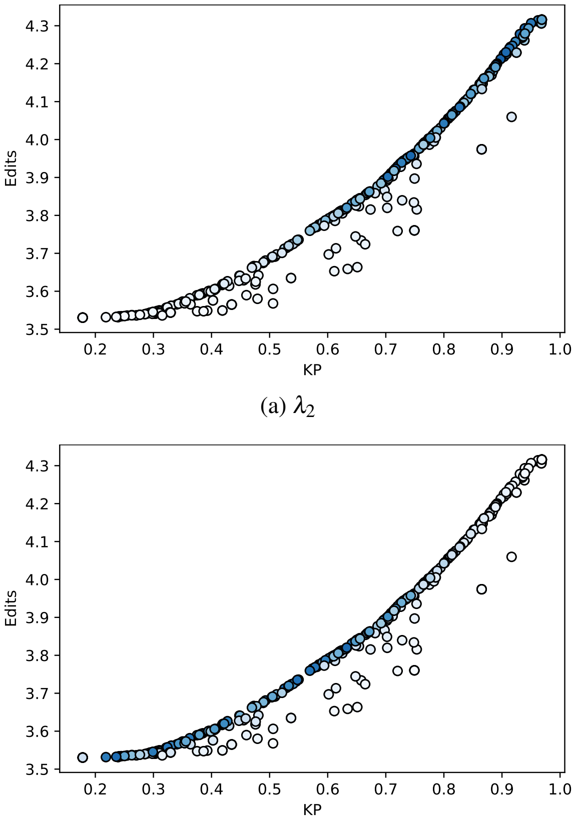
Figure 6: Scatter plot of all 400 iterations from the second experiment using VGG-16. The Pareto front is highlighted in red.

via the saturation of the data points in Figure 7, we can see a tradeoff between λ_2 correlating to good KP but poor E, while D correlates with worse KP but better E. This is also reflected in the Spearman correlation in Table 4, which shows a strong correlation with λ_2 and a good KP but many edits, and the opposite for D.

In order to showcase visual results, a choice of a specific hyperparameter pair is necessary. We opted to choose $\lambda_2 = 0.51,\ D = 0.09$ from the pareto front, which yields 3.65 average edits and an average KP of 61.1%. For these hyperparameters, we showcase the first edit in Figure 8. We can see that the first edit is identified to be between the heads of the birds.

Figure 9 shows a comparison of our work to that of Vandenhende et al. (2022). We note that when they performed a grid search on their hyperparameters (those being λ_1 and τ), it was without some of the other improvements they have added, such as only considering one distractor image instead of choosing out of multiple ones. For the sake of a fair comparison, we repeat their grid search with the addition of their improvements.

Our results consistently outperform those of Vandenhende et al. (2022) along both metrics. The data point with the lowest KP and edits is the data point where the respective weight hyperparameter λ_1 or λ_2 is 0, meaning that only the classification loss $\mathcal{L}_c$ is considered. This point is the equivalent to the initial approach by Goyal et al. (2019), though with the additional features Vandenhende et al. (2022). A full overview of the results for the edits and all KP metrics can be seen in Table 5. In addition to the results from the related work, we have also added results for the Goyal et al. (2019) approach of only using $\mathcal{L}_c$ (i.e. $\lambda_1 = \lambda_2 = 0$) with later additions such as multiple distractor images, as well as our balanced set of hyperparameters with $\lambda_2 = 0.51,\ D = 0.09$ and a set with



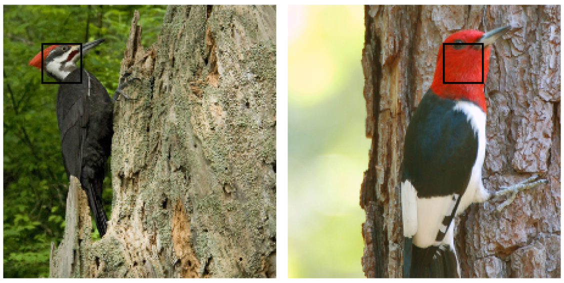
(a) λ_2

(b) D

Figure 7: Scatter plot of all 400 iterations of the second experiment using VGG-16. The saturation of the data points indicates the value of a given hyperparameter, with more saturation corresponding to higher values.

Table 4: Spearman correlation and related p-values between the hyperparameters and the optimization metrics over the second experiment using VGG-16.

	r_{KP}	$p(r_{KP})$	r_E	$p(r_E)$
λ_2	0.77	$< 0.1\%$	0.82	$< 0.1\%$
D	-0.57	$< 0.1\%$	-0.48	$< 0.1\%$



(a) Query image

(b) Distractor image

Figure 8: Features to be edited from a query-distractor pair.

Table 5: Comparison of all metrics between Goyal et al. (2019), Vandenhende et al. (2022), and our implementation on a VGG-16 network. *KP* are given in %. We are listing both a balanced set of hyperparameters with $\lambda_2 = 0.51,\ D = 0.09$ as well as a *KP*-optimizing set of hyperparameters at $\lambda_2 = 10,\ D = 0$. The best results for each metric are bolded.

	E	$KP_{S,N}$	$KP_{S,S}$	$KP_{A,N}$	$KP_{A,S}$
Goyal et al. (2019)	5.5	67.8	17.2	54.6	8.3
Goyal et al. (2019) (Multiple Distractors)	**3.5**	75.3	24.2	70.4	17.8
Vandenhende et al. (2022)	3.8	78.1	41.0	71.9	36.5
Proposed	3.7	85.4	58.7	83.7	61.1
Proposed (*KP* Optimized)	4.3	**100.0**	**100.0**	**97.7**	**96.9**

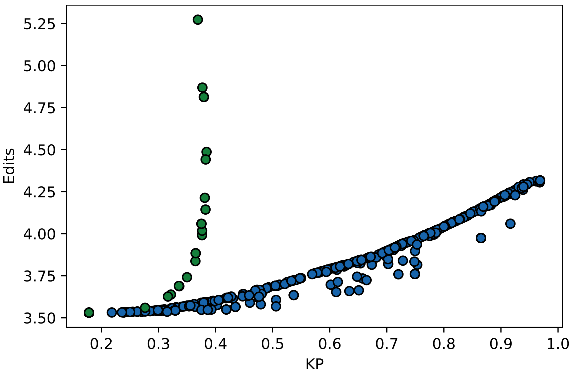


Figure 9: Scatter plot of 400 iterations from the second experiment using VGG-16 (blue) and scatter plot of 18 grid search samples from Vandenhende et al. (2022) (green).

$\lambda_2 = 10,\ D = 0$ which optimizes the KP metrics.

Notably, when optimizing for *KP* metrics, we can see that we reach exactly 100% on the single-edit evaluations. This means that the first edit is always between two features that share a common part. However, this also leads to the highest number of edits out of any of the multi-distractor approaches. On the other hand, using only $\mathcal{L}_c$ in the loss calculation while using multiple distractors yields the fewest edits, but it also features the worst KP metrics out of the multi-distractor approaches. Both the approach by Vandenhende et al. (2022) and our balanced one have results in between these extremes, with our balanced approach consistently outperforming theirs in each metric. While edits are only slightly improved by 0.1 edits, the KP values are improved by at least 7 percentage points at their lowest and up to 24 percentage points at their highest.

5.3 Runtime Performance

We perform a runtime analysis using the conditions of Experiment 1, i.e. 400 optimization iterations using all hyperparameters. Using these conditions, we achieve a maximum runtime of 6449.5 seconds, a minimum runtime of 4239.7 seconds, and an average runtime of 5010.5 seconds. The average time per edit is 0.255 seconds. Note that as this is an optimization, it is to be expected that the results improve during it, and as such, runtime also improves.

This can be seen when comparing the correlation between the number of edits and the total runtime of each optimization iteration. Given that we analyze how accurate the average runtime is, we will perform a Pearson correlation analysis (Pearson, 1896) in addition to a Spearman one. Because Pearson correlation indicates the correlation of the raw values instead of the ranks as Spearman does, a high Pearson correlation indicates a linear relationship between the number of edits and the runtime, meaning that it strongly indicates whether the average time per edit is accurate to most edits. The values have a Pearson correlation of $\rho = 0.995$, a Pearson p-value of $< 0.1\%$, a Spearman correlation of $r = 0.992$, and a Spearman p-value of $< 0.1\%$. This shows that the runtime of a given edit is largely similar among different executions, so the true measure for the runtime is how many total edits are occurring.

6 DISCUSSION

Our approach outperforms existing counterfactual creation approaches within the KP and edits metrics. By adding the parts loss $\mathcal{L}_p$ and therefore specifically encouraging edits that will optimize the KP metrics, we are able to especially achieve higher KP metrics. There exist especially large improvements on the Same-KP metric, indicating that our approach edits between the same semantic parts more frequently than previous work. This becomes clear when considering the ratio of Same-KP to Near-KP, which indicates how many edits that concerned a semantic part also edited between the same semantic parts. In the approach by Vandenhende et al. (2022), when considering only the first edit, this ratio is 52.5%, and over all edits it drops to 50.8%. In our approach, meanwhile, on the first edit, the ratio is 68.8%, and this ratio actually rises to 73.0% over all edits. This shows that the

Same-KP metrics rose comparatively higher than the Near-KP metrics.

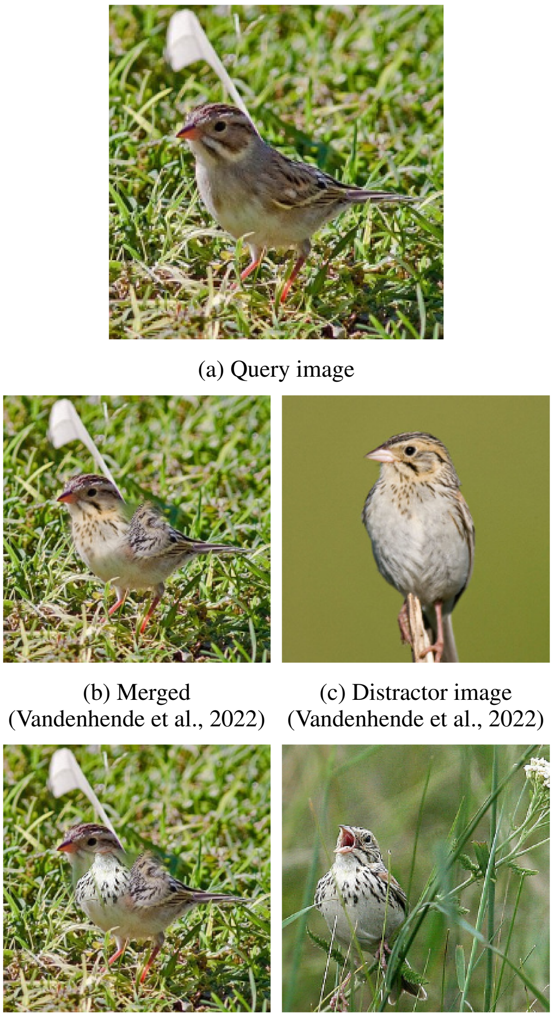
However, we note that these metrics are still only approximations of the actual goal of the field of XAI, that being creating explanations that are understandable to humans. While better performance in these metrics has correlated with better results in the user study performed by Vandenhende et al. (2022), this does not necessarily apply in every circumstance. Choice of metrics in general is not a settled topic (Vilone and Longo, 2021), with Rosenfeld (2021) proposing a general set of metrics that generally lead to simpler and more inherently explainable models being preferred. As such, if the user-based metrics for our approach are poor in comparison to the greater performance of the heuristics-based metrics, a different choice of metrics may better represent the impact on the users.

Goyal et al. (2019) presented some of their counterfactual explanations by merging the edited features over the query image via vignette masks. A display of this approach can be seen in Figure 10. This approach displays a smoother approximation of edits and is likely to be more pleasant viewing than a more discrete pasting of a feature rectangle would be. However, while it does make for a comparatively prettier representation, it is not quite accurate to the process of counterfactual approximation. Within the actual process, the entirety of the feature is edited.

Examples such as Figure 10 show that while our approach performs well within the metrics, the approach by Vandenhende et al. (2022) tends to produce smoother transitions. For instance, while our approach edits the chest area in such a way that produces visible tears, Vandenhende et al. (2022) manage to edit the chest area in such a way that the transition appears seamless. However, merged images such as these have not actually been used within the user studies tracking understandability of the explanations by Goyal et al. (2019) and Vandenhende et al. (2022). Instead, the features to be edited were represented using bounding boxes as seen in Figure 11. While our merged images do not look as seamless as those in the semantic consistency approach, the bounding box representation in this figure still shows an edit between the breasts of the birds.

Internal threats to validity can occur from our heavy reliance on the parts annotation, which is a comparatively less-utilized aspect of the CUB-200-2011 dataset. Errors within said annotations would greatly influence the results.

Regarding external threats, it is impossible to utilize the parts loss on datasets without semantic part locations. The specific hyperparameter values for the



(a) Query image

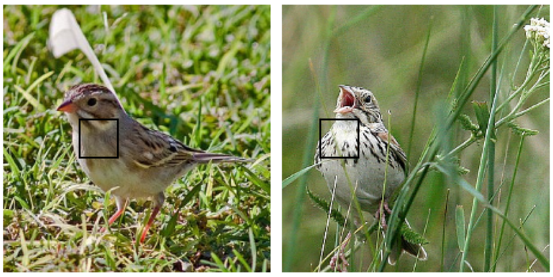
(b) Merged (Vandenhende et al., 2022)

(c) Distractor image (Vandenhende et al., 2022)

(d) Merged (Proposed)

(e) Distractor image (Proposed)

Figure 10: Merged images of the query image and counterfactual features using a vignette mask as well as the distractor images for the second edit, comparing our approach and the one by Vandenhende et al. (2022).



(a) Query image

(b) Distractor image

Figure 11: Features to be edited from a query-distractor pair for our approach on the second edit.

tolerance distance D are also only valid within the context of the specific parts within the CUB-200-2011 dataset that they were optimized for, as well as relying on consistent feature grid dimensions to remain accurate. Changing the dataset or the feature grid dimensions would require optimizing these hyperparameters once more. In comparison, the approach by Vandenhende et al. (2022) is more generically applicable, as semantic consistency is transferable between models and does not require any additional training or optimization once the auxiliary model has been initially trained. As such, our specific results are only applicable to this dataset and only generalizable to a specific subset of datasets. The fact that an improvement over the state of the art exists, even if limited to a specific dataset, may encourage the creation of similar annotations on further datasets.

Additionally, within the first experiment, we only determine correlation, which is not causation. While we perform multiple experiments to confirm our hypotheses and reach very low p-values within them, it is nonetheless not a guaranteed indication of causation. Because we determine the Spearman correlation on data that is gained from optimization, it is possible that the optimization itself acts as a confounder, leading to uncorrelated hyperparameters to be be seen as correlated (Pearl, 2000). This is an especially high risk in the case of the hyperparameter S due to its binary state. Assuming that it is uncorrelated to begin with, random sampling may still create the impression that it is correlated in some way. Therefore the optimization algorithm might prefer that value for S in further tests, leading to a potentially inaccurate view of the actual level of correlation. However, this can only create the impression of a correlation where there is none, and given that S and λ_1 showed a negative correlation, they are either actually negatively correlated or not correlated. Either way, they would not be viable candidates. The positive correlation of λ_2 and D in respect to one of the metrics each has been observed over multiple optimization runs for both the first experiment and additionally for both the VGG-16 and ResNet-50 networks within the second experiment. Therefore, while it is impossible to rule out, it is unlikely that λ_2 or D are not actually correlated to the metrics.

7 CONCLUSIONS & FUTURE WORK

In this work we have presented an approach for creating visual counterfactual explanations based upon a new parts loss term $\mathcal{L}_p$. This loss term takes into account the accuracy of an edit with regards to the semantic parts, controlled by a hyperparameter λ_2. We also created hyperparameters to add to this loss by adding a distance tolerance D to account for the arbitrary nature of the bounds between features, as well as an option S to simplify the semantic parts within the high-density area of the head. Using an in-depth optimization process, we then determined the performance of the new hyperparameters. This is opposed to the semantic loss term $\mathcal{L}_s$ originally introduced by Vandenhende et al. (2022) which used an external network to measure semantic similarity.

Our evaluation determined that the hyperparameter λ_1 controlling semantic loss $\mathcal{L}_s$ has little correlation with the keypoint metrics, and strong negative correlation with the number of edits. We also determined similar results for the simplification option S. As such we focused on experiments using only the hyperparameters λ_2 and D. By optimizing these two hyperparameters we could find configurations that outperform the standard set by Vandenhende et al. (2022). In particular we improve upon their work by an average of 0.1 edits and at least 7 percentage points in the keypoint metrics when considering the keypoints generally being part of the bird and at least 17 percentage points when considering if the edits share a semantic part.

We have created an explanation algorithm that is inherently explainable which outperforms the previous work while not relying on an external model. By using metadata such as semantic part locations, this approach could be expanded to other data sets.

In the future, we plan to perform user studies to determine whether the improved metrics of our approach lead to more understandable explanations. We also invite other researchers to investigate if similar types of metadata from other datasets could be used to improve the performance of this or other explanation approaches on those datasets.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable feedback. This work was partially funded by the German Federal Ministry for Education and Research (BMBF) through grant 01IS22062 ("AI research group FFS-AI").

REFERENCES

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, pages 2623–2631. ACM.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Proc. Neural Information Processing Systems*, NeurIPS '11. Curran Associates, Inc.

Boreiko, V., Augustin, M., Croce, F., Berens, P., and Hein, M. (2022). Sparse visual counterfactual explanations in image space. In *Proc. DAGM German Conference on Pattern Recognition*, GCPR '22, pages 133–148. Springer International Publishing.

Castelvecchi, D. (2016). Can we open the black box of AI? *Nature News*, 538(7623):20–23.

Chen, I. Y., Pierson, E., Rose, S., Joshi, S., Ferryman, K., and Ghassemi, M. (2021). Ethical machine learning in healthcare. *Annual Review of Biomedical Data Science*, 4:123–144.

Chen, L., Yan, X., Xiao, J., Zhang, H., Pu, S., and Zhuang, Y. (2020). Counterfactual samples synthesizing for robust visual question answering. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10797–10806. IEEE.

Gomez, O., Holter, S., Yuan, J., and Bertini, E. (2020). *ViCE*: Visual counterfactual explanations for machine learning models. In *Proc. 25th International Conference on Intelligent User Interfaces*, IUI '20, pages 531–535. ACM.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017). Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPR '17, pages 6325–6334. IEEE.

Goyal, Y., Wu, Z., Ernst, J., Batra, D., Parikh, D., and Lee, S. (2019). Counterfactual visual explanations. In *Proc. 36th International Conference on Machine Learning*, PMLR '19, pages 2376–2384. PMLR.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proc. Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE.

Hendricks, L. A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., and Darrell, T. (2016). Generating visual explanations. In *Proc. European Conference on Computer Vision*, ECCV '16, pages 3–19.

Mothilal, R. K., Sharma, A., and Tan, C. (2020). Explaining machine learning classifiers through diverse counterfactual explanations. In *Proc. Conference on Fairness, Accountability, and Transparency*, FAT* '20, pages 607–617. ACM.

Pearl, J. (2000). *Causality: Models, Reasoning and Inference*. Cambridge University Press.

Pearson, K. (1896). VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 187:253–318.

Petsiuk, V., Jain, R., Manjunatha, V., Morariu, V. I., Mehra, A., Ordonez, V., and Saenko, K. (2021). Black-box explanation of object detectors via saliency maps. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPR '21, pages 11438–11447. IEEE.

Rosenfeld, A. (2021). Better metrics for evaluating explainable artificial intelligence. In *Proc. 20th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '21, pages 45–50. IFAAMAS.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Schwarting, W., Alonso-Mora, J., and Rus, D. (2018). Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. 2nd International Conference on Learning Representations – Workshops Track*, ICLR '14.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proc. 3rd International Conference on Learning Representations*, ICLR '15, pages 1–14. Computational and Biological Learning Society.

Spearman, C. E. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15(1):72–101.

Vandenhende, S., Mahajan, D., Radenovic, F., and Ghadiyaram, D. (2022). Making heads or tails: Towards semantically consistent visual counterfactuals. In *Proc. European Conference on Computer Vision*, ECCV '22, pages 261–279. Springer Nature Switzerland.

Vilone, G. and Longo, L. (2021). Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, 76:89–106.

Wachter, S., Mittelstadt, B., and Russell, C. (2018). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31(2):841–887.

Wachter, S., Mittelstadt, B., and Russell, C. (2020). Bias preservation in machine learning: The legality of fairness metrics under EU non-discrimination law. *West Virginia Law Review*, 123(3):735–790.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. (2019). CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proc. IEEE/CVF International Conference on Computer Vision*, (ICCV '19, pages 6022–6031. IEEE.

Zhang, B., Anderljung, M., Kahn, L., Dreksler, N., Horowitz, M. C., and Dafoe, A. (2021). Ethics and governance of artificial intelligence: Evidence from a survey of machine learning researchers. *Journal of Artificial Intelligence Research*, 71:591–666.