UNIVERSITÄT KOBLENZ · LANDAU
Faculty 4: Computer Science

WeST People and Knowledge Networks
Institute for Web Science and Technologies

Generating Counterfactual Images for Visual Question Answering by Editing Question-Critical Objects

Master's Thesis

in partial fulfillment of the requirements for
the degree of Master of Science (M.Sc.)
in M.Sc. Web and Data Science

submitted by
Timo Hartmann

First supervisor: Prof. Dr. Matthias Thimm
Institute for Web Science and Technologies

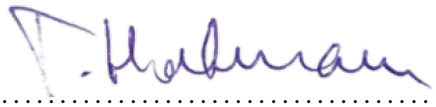
Second supervisor: Dr. Zeyd Boukhers
Institute for Web Science and Technologies

Koblenz, August 2021

Statement

I hereby certify that this thesis has been composed by me and is based on my own work, that I did not use any further resources than specified – in particular no references unmentioned in the reference section – and that I did not submit this thesis to another examination before. The paper submission is identical to the submitted electronic version.

	Yes	No
I agree to have this thesis published in the library.	☒	☐
I agree to have this thesis published on the Web.	☒	☐
The thesis text is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐
The source code is available under a GNU General Public License (GPLv3).	☒	☐
The collected data is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐

Köln, 13/08/2021 

(Place, Date) (Signature)

Note

- If you would like us to contact you for the graduation ceremony, please provide your personal E-mail address: timo.hartmann@posteo.de
- If you would like us to send you an invite to join the WeST Alumni and Members group on LinkedIn, please provide your LinkedIn ID : .....

Zusammenfassung

Wenngleich Visual Question Answering (VQA) Systeme sich in den letzten Jahren erheblich verbessert haben, neigen sie noch immer dazu, Fehler zu produzieren, die für menschliche Benutzer:innen schwer zu deuten sind. Dies zeigt die Notwendigkeit auf, ihre Entscheidungsfindungsprozesse besser interpretierbar zu machen. In der vorliegenden Masterarbeit wird *CountEx-VQA - Counterfactual Explanations for Visual Question Answering* vorgestellt. Dabei handelt es sich um eine Methode, die kontrafaktische Bilder generiert, um die Interpretierbarkeit eines VQA-Modells zu erhöhen. Konkret nimmt der Bildgenerator die minimal erforderlichen Änderungen an einem Bild vor, sodass das VQA-Modell seine Antwort auf eine gegebene Frage verändert. Dabei ist das Modell darauf trainiert, dass die generierten Bilder semantisch sinvolle Änderungen im Vergleich zum Original enthalten, visuell realistisch sind und sich nur in solchen Bereichen von ihren Originalen unterscheiden, die zur Beantwortung einer gegebenen Frage relevant sind. Die Ergebnisse umfangreicher Experimente mit einem anspruchsvollen Datensatz zeigen, dass die Methode teilweise erfolgreich kontrafaktische Bilder generiert. Sie deuten insbesondere darauf hin, dass *CountEx-VQA* im Kontext farbbasierter Fragen weitestgehend geeignet ist, während die Methode Schwierigkeiten damit hat, größere semantische Veränderungen, wie z.B. Formveränderungen, vorzunehmen. Darüber hinaus neigt der Bildgenerator in einigen Fällen dazu, unrealistische Artefakte zu erzeugen.

Abstract

While Visual Question Answering (VQA) systems improved significantly in recent years, they still tend to produce errors that are hard to construe for human users. The lack of interpretability in black-box VQA models raises the necessity for discriminative explanations alongside the models' outputs. This thesis introduces *CountEx-VQA - Counterfactual Explanations for Visual Question Answering*, a method that generates counterfactual images to increase a VQA model's interpretability. Specifically, given a question-image pair, the proposed method predicts the minimal number of edits to the image in order to change the VQA model's predicted answer. Thereby, the model is trained such that the generated images contain semantically meaningful changes, are visually realistic, and differ from their originals only in question-critical areas. The results of extensive experiments on a challenging dataset show the method to be partly successful in generating meaningful counterfactual images. Particularly, they suggest that *CountEx-VQA* is suitable concerning many color-based questions, while it has difficulties regarding larger semantic modifications (e.g., altering shapes). In addition, it tends to produce unrealistic artifacts in some cases.

Contents

1. Introduction **1**

2. Background **5**
2.1. Introduction to Deep Learning 5
2.2. Convolutional Neural Networks 10
2.3. Autoencoders 11
2.4. Generative Adversarial Networks 12

3. Related Work **15**
3.1. Interpretable Machine Learning 15
3.2. Visual Question Answering (VQA) 19
3.2.1. Methods for VQA 19
3.2.2. Interpretable VQA 21

4. CountEx-VQA **23**
4.1. Overview 23
4.2. Implementation Details 26
4.2.1. Attention Mechanism 27
4.2.2. Spectral Normalization 28
4.2.3. Generator 29
4.2.4. Discriminator 31
4.2.5. Training procedure 32
4.3. Experimental Setup 33
4.3.1. Data 33
4.3.2. VQA Model 35
4.3.3. Evaluation 36

5. Results **39**
5.1. Attention Mechanism 39
5.2. Evaluation of the Counterfactuals 44

6. Discussion **49**

7. Conclusion and Future Work **51**

List of Abbreviations **53**

List of Figures **56**

A. Appendix **59**

References **61**

1. Introduction

Throughout the past years, many scholars from both the computer vision and the Natural Language Processing (NLP) communities investigated the task of VQA [23, 35, 60, 5, 4, 11, 85]. Typically, VQA models aim at answering a natural language question about an image [85]. While VQA systems improved significantly in recent years, they still tend to be erroneous. Partly, these errors result from the fact that VQA models tend to focus primarily on the question while neglecting most of the images' visual content [11]. Specifically, Niu et al. [53] suggest that VQA models often suffer from a *language prior*, which describes a strong correlation between questions and answers in training datasets. The VQAv1 dataset [5] is heavily unbalanced regarding certain types of questions. For instance, simply answering *tennis* to any sports-related question without considering the image yields an accuracy of approximately 40%. Furthermore, the authors point out that human questioners tend to pose questions about objects they see in an image. Therefore, VQA training datasets usually suffer from what they call a *visual priming bias*. For example, blindly answering *yes* to all questions starting with *Do you see a...?* without considering the image or the rest of the question yields approximately 90% accuracy in the VQAv1 dataset [53, 23].

The errors produced by a VQA model are often hard to construe for human users [53]. What caused the model to output a given answer based on an image-question pair? The lack of interpretability regarding the decisions made by VQA models raises the necessity to provide discriminative explanations alongside their outputs. In the past decades, the Machine Learning (ML) community has widely researched and acknowledged the importance of increasing the interpretability of black-box ML models [10, 71, 49, 88, 79, 46, 89, 29, 19, 16]. Doshi-Velez and Kim [16] suggest that the motivations for making ML models interpretable are manifold, including the need to prevent certain groups from being discriminated against or to protect sensitive information in data to foster privacy. Moreover, to ensure the reliability and robustness of ML systems, it can be crucial to understand how an algorithm performs when its parameters or inputs vary. Ultimately, interpretability increases the users' trust in ML systems. For instance, if clinicians use such a system to help them in deciding whether a magnetic resonance tomography (MRT) depicts a malignant tumor, they might want an explanation as to why the model made a certain prediction before taking any action [2].

In the VQA domain, only a limited number of papers exist that investigate the black-box processes inherent in VQA models. Existing attempts include (i) the identification of visual attributes that are relevant to the question [35, 90] and (ii) the generation of counterfactuals [58, 53, 78]. For instance, Zhang et al. [90] implemented a method that generates a heatmap over the input image to highlight the images' regions that lead to the VQA model's answer. Fernández-Loría et al. [17] point out that while highlighting feature importance explains a model's *predictions*, this method is not sufficient to explain the *decisions* that led to the model's output. The

authors suggest that counterfactual explanations offer a more sophisticated way to increase the interpretability of a ML model because they reveal the causal relationship between features in the input and the model's decision. For instance, suppose that a model classifies an MRT to show a malignant tumor and a heatmap highlights a particular spatial region in the image that was key to the prediction. While this approach explains how and why a model reached a certain decision, it does not provide insight into how it would behave under alternative conditions. Would altering a few pixels in a particular area of the highlighted region be enough to make the model classify it as benign, or would the whole region have to look entirely different? A counterfactual image that is minimally different from the original but causes the classifier to change its prediction would provide an answer to this question. To my knowledge, only three existing methods aim at making VQA models interpretable by providing counterfactual images. Chen et al. [11] introduce a method that generates counterfactual samples by applying masks to critical objects in images or words in questions. Similarly, Teney et al. [78] suggest a method that masks features in images whose bounding boxes overlap with human-annotated attention maps. Finally, in their ongoing research, Pan et al. [58] propose a framework to generate counterfactual images by editing the original image such that the VQA model returns an alternative answer for a given question. Currently, their approach is mainly restricted to color-based questions.

Contributions The present thesis introduces *CountEx-VQA - Counterfactual Explanations for VQA*. *CountEx-VQA* is a method which aims to enhance Pan et al.'s [58] counterfactual generator such that it (i) can produce larger semantic edits (e.g., apply changes also for shape-based questions) and (ii) focuses more on those spatial image regions that are key to answering a posed question (i.e., the question-critical objects) rather than the entire image. Specifically, this thesis provides the following contributions:

- An implementation of the proposed counterfactual generator, called *CountEx-VQA (Counterfactual Explanations for Visual Question Answering)*, is publically available and can be accessed on GitHub[1].
- It introduces an attention mechanism that identifies question-critical objects in an image and guides the counterfactual generator regarding where to apply modifications.
- It introduces a weighted reconstruction loss that allows the counterfactual generator to make more significant changes to question-critical spatial regions than the rest of the image.
- Extensive experiments on the challenging VQAv1 dataset [5] were conducted to validate the approach.

[1] `https://github.com/tihartmann/CountEx-VQA`

- Training Generative Adversarial Networks (GANs) can suffer from instability that can cause non-convergence [45]. To tackle this problem, Pan et al. [58] use a method called *gradient clipping*, which requires extensive empirical fine-tuning of the training algorithm. To bypass this extensive procedure, *CountEx-VQA* uses spectral normalization as introduced by Miyato et al. [45] for increased training stability.

Thesis Outline

This thesis is organized as follows:

- Section 2 provides the technical foundations necessary for the conducted research. It introduces the principles of Deep Learning (DL) and Convolutional Neural Networks (CNNs) as well as the fundamentals of GANs.
- Section 3 discusses the related work in the field of interpretable ML and different methods for VQA, including existing attempts to make them interpretable.
- Section 4 presents the details of the proposed method and the experimental setup. It provides a description of the attention mechanism as well as the architecture of the counterfactual image generator. Furthermore, it discusses the model's training procedure and the used methods of evaluation.
- Section 5 presents qualitative and quantitative results of the conducted experiment, including several visual examples of the attention mechanism and the counterfactual generator.
- Section 6 discusses the main findings' implications based on the conducted evaluations.
- Section 7 concludes this thesis by summarizing its key contributions and providing an outlook for potential future work.

2. Background

CountEx-VQA uses a GAN-based DL architecture to generate counterfactual images for VQA. The original GANs as introduced by Goodfellow et al. [22] build on the concept of Multi-Layer Perceptrons (MLPs) [59]. More recent implementations of GANs adopt CNNs, which significantly improves their ability to extract image features [59]. The following subsections provide the theoretical foundations required for the method proposed in this thesis: DL, CNNs, and GANs.

2.1. Introduction to Deep Learning

In recent years, DL has made major advances and shapes the state-of-the-art in many ML domains, such as NLP and image recognition [68]. ML aims at extracting patterns from raw data to make predictions without relying on hard-coded inference rules (e.g., using a knowledge base) [21]. While ML algorithms require human engineers to transform raw data, such as images, into suitable internal representations (e.g., feature vectors) first, DL allows to process it in its raw form [32]. To this end, DL uses representation learning to automatically discover the representations necessary to detect patterns in the input [21]. Ultimately, these representations can be used to learn complex functions, e.g., for image generation tasks [32]. DL methods can be roughly classified into two major categories, namely supervised learning and unsupervised learning [26]. In supervised learning, the data used to train a DL system includes the desired output, called *labels*. For instance, a supervised DL algorithm could learn to distinguish real images from fake images generated by a GAN based on example images along with their *class* (i.e., real or fake). Unsupervised DL models, on the other hand, aim to find patterns in unlabeled data, for instance, to cluster them into groups of similar instances [26]. GANs use a combination of supervised and unsupervised learning: First, a *generator* model aims to produce new images that follow a similar distribution as the training data, which is an unsupervised learning problem [63]. Simultaneously, the *discriminator*, which is used to penalize unrealistic generated images, is trained on images from the training data (labeled as *real*) and the set of generated images (labeled as *fake*) to learn to distinguish them [63].

Neural Networks

Neural Networks (NNs) are the building block of DL [26]. One of the simplest forms of a NN is the MLP [57], which is also adopted in the original GAN architecture [22]. An MLP consists of multiple layers of Linear Threshold Units (LTUs), which are two-state, sigmoid elements or *neurons* that are linked via weighted connections [57, 26]. Specifically, accdording to Pal and Mitra [57], an *input layer* is followed by an arbitrary number of intermediate *hidden layers* and a final *output layer* at the end. Thereby, the authors suggest that each neuron is fully-connected to all neurons in the adjacent layers via weights, which determine the degree of correlation between

two neurons. In addition to a fully-connected LTU, all layers of an MLP except the output layer usually also contain an extra *bias neuron*, which always outputs 1 ($b = 1$) [26]. Finally, each layer except for the input layer uses a monotonic, nonlinear *activation function* to compute its output [57].

Activation Functions

Since the training procedure of MLPs includes Gradient Descent (GD), which is explained in the subsequent section, it is necessary to apply an activation function that has a well-defined non-zero derivative at each point [26]. The following activation functions are used in this thesis [54]:

- The *Logistic function*, $\sigma(z) = \frac{1}{1+exp(-z)}$, is a continuous and differentiable *sigmoid* function (i.e., shaped like the letter S) that outputs values between $[0, 1]$.
- The *Hyperbolic Tangent Function (Tanh)*, $\tanh(z) = 2\sigma(2z) - 1$, is also a continuous sigmoid function that is differentiable. In contrast to the logistic function, its output values are in the range $[-1, 1]$, which results in the outputs being more or less normalized (i.e., centered around zero) in the early training stages. This characteristic often leads to faster convergence of an MLP.
- The Rectified Linear Units (ReLU) function, $\text{ReLU}(z) = \max(0, z)$, is a continuous function, but it is not differentiable if $z \leq 0$. In this case, its slope changes abruptly, which can be problematic when using GD. On the other hand, it is fast to compute and works well in most scenarios, so it is usually well suited in practice.

Backpropagation Training Algorithm

To train an MLP, Rumelhart et al. [66] introduced the *backpropagation* algorithm. For each training instance, the MLP computes the output of every neuron in the network. This procedure is called the *forward pass*. By measuring the difference between the desired ground-truth and the network's prediction, the MLP computes the *output error*. It then propagates the error gradient backward layer by layer to measure how much each neuron contributed to the error in the subsequent layer. Finally, using these error gradients, the backpropagation algorithm performs a *GD* step on all the connection weights [66]. According to Géron [26], GD tweaks a model's parameters to minimize the computed error rate by measuring the local gradient of the error function for the model's parameter vector θ. Thereby, a hyperparameter called *learning rate* specifies the size of the steps. The author suggests using a small learning rate requires the algorithm to perform many iterations until it converges. In contrast, if the learning rate is too high, the algorithm may fail to converge as it jumps across the error rate's minimum, resulting in even larger values [26]. Unless the cost function is *convex*, GD may only find a *local minimum* rather than the *global minimum* [66].

Vanishing and Exploding Gradients

The backpropagation algorithm with GD can be problematic when training larger NNs, and GANs in particular [26, 61, 84]. Since the algorithm starts at the output layer and progresses through the network until it reaches the input layer, Deep Neural Networks (DNNs) may suffer from different layers learning at different speeds, which leads to unstable gradients [26]. According to Géron [26], the *vanishing gradients* problem occurs when gradients decrease significantly the further the backpropagation algorithm progresses. As a result, the GD step leaves the weights of the lower layers virtually unchanged. The author suggests that one root cause of vanishing gradients can be the choice of the used activation function. For instance, the sigmoid function saturates at 0 or 1 when its inputs are very large negative or positive values, respectively. In turn, the derivative shifts extremely close to 0, which results in almost no gradients being backpropagated through the network. Similarly, the author points out that while it cannot saturate for positive values, the ReLU function can yield the same problem as the sigmoid function for negative values. This property is referred to as *dying ReLU* [26]. To tackle this problem, a modified version of ReLU is used in this thesis, namely *leaky ReLU* [54]:

$$\text{LeakyReLU}_{\alpha}(z) = \begin{cases} z, & \text{if } z > 0 \\ \alpha \times z, & \text{if } z \leq 0, \end{cases}$$

where the α-parameter determines the slope of the function for $z < 0$. Usually, α is set to a small value (e.g., 0.01), ensuring that the gradients will not saturate during training.

In contrast to vanishing gradients, gradients growing bigger and bigger as the backpropagation algorithm progresses to the lower layers is known as the *exploding gradients* problem [26]. In their approach to generate counterfactual images for VQA, Pan et al. [58] applied a technique called *gradient clipping* to tackle this problem. This method rescales a gradient if its norm exceeds a certain threshold c[26]:

$$\mathbf{g} = \begin{cases} c \times \frac{\mathbf{g}}{||\mathbf{g}||}, & \text{if} ||\mathbf{g}|| \geq c \\ \mathbf{g}, & \text{otherwise} \end{cases}$$

where $\mathbf{g}$ is the gradient and $||\mathbf{g}||$ is the norm of $\mathbf{g}$. Another popular method to address the vanishing and exploding gradients problem is to apply *instance normalization* [81] right before each layer's activation function to zero-center each input across its height and width [26]:

$$
\begin{aligned}
\mu_I &= \frac{1}{uv}\mathbf{x}, \\
\sigma_I^2 &= \frac{1}{uv}(\mathbf{x} - \mu_I)^2, , \\
\hat{\mathbf{x}}^{(\mathbf{i})} &= \frac{\mathbf{x}^{(i)} - \mu_I}{\sqrt{\sigma_I^2 + \epsilon}}, \\
\mathbf{z}^{(i)} &= \gamma \hat{\mathbf{x}}^{(i)} + \beta,
\end{aligned}
\tag{1}
$$

where

- μ_I is the empirical mean evaluated over the current instance.
- σ_I is the empirical standard deviation evaluated over the current instance.
- $\hat{\mathbf{x}}^{(i)}$ is the input after being shifted (i.e., zero-centering) and normalized.
- γ is the scaling parameter learned for each instance-normalized layer.
- β is the shifting parameter (offset) learned for each instance-normalized layer.
- ϵ represents a *smoothing term* that prevents division by zero.
- $\mathbf{z}^{(i)}$ is the instance normalization's output, i.e., the scaled and zero-centered input.

Miyato et al. [45] introduce a method called *spectral normalization*, which normalizes the weights of a NN. The authors suggest that this method, which can also be used in conjunction with instance normalization, achieves better results regarding exploding and vanishing gradients than gradient clipping. Therefore, *CountEx-VQA* adopts spectral normalization, which is discussed in detail in section 4.

Optimization Algorithms

Training a GAN using the regular GD algorithm can be very inefficient while applying a faster optimizer can speed up the training procedure significantly [26]. One of the most popular optimization algorithms is the *adam* optimizer [30], which is also used for the experiments conducted in this thesis. In contrast to traditional stochastic GD, which uses a single learning rate, the adam optimizer maintains a separate learning rate for each of the network's parameters. These individual learning rates are adapted as the training procedure progresses by updating estimates of the gradients' mean and the squared gradients' uncentered variance (called *exponential moving averages*). These are referred to as the gradients' first (denoted $\mathbf{m}_t$) and second ($\mathbf{v}_t$) *momentum* vectors, which are initialized at 0. The parameters θ_t of an NN at step t are computed as follows [30]:

$$\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\Delta_\theta J(\theta)$$
$$\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \times (\Delta_\theta J(\theta))^2$$
$$\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1 - \beta_1^t}$$
$$\hat{\mathbf{v}}_t \leftarrow \frac{\mathbf{v}_t}{1 - \beta_2^t}$$
$$\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon},$$

where

- $J(\theta)$ is the cost function with respect to the weights θ.
- $\Delta_\theta J(\theta)$ is the gradient vector.
- $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ denote the bias-corrected versions of the vectors vectors $\mathbf{m}_t$ and $\mathbf{v}_t$.
- α is the stepsize of the algorithm.
- $\beta_1, \beta_2 \in [0, 1]$ are hyper-parameters that control the exponential decay rates of the moving averages. β_1 is usually initialized to 0.9, while β_2 is typically initialized to 0.999).

Regularization

DNNs such as GANs can consist of millions of parameters, which makes them both extremely flexible to diverse datasets and prone to overfitting the training data [26]. Therefore, as Géron [26] points out, it can be useful to apply a regularization technique. According to the author, a straightforward solution to prevent a DNN from overfitting is to stop the training procedure when the model's performance starts dropping for the validation set. Another solution is to use the *dropout* algorithm introduced by Srivastava et al. [74], which temporarily excludes each neuron (except for the output neurons) with a probability of p at every training step. According to the authors, dropout ensures that a DNN does not learn to focus only on a small set of neurons to make a prediction. Since every neuron has a probability p to be ignored, they (i) cannot co-adapt with their neighboring neurons but must work on their own without relying on them, and (ii) must learn to work with a randomly chosen sample of neurons. As a result, the DNN generalizes better to new inputs. The authors state that the *dropout rate* p is typically set to 0.5, and dropout is only applied during training. For testing, each input connection weight must be multiplied by p to ensure that the neurons do not receive input signals twice as large as during training [74].

2.2. Convolutional Neural Networks

CNNs [33] are a particular kind of DL networks that is widely applied in computer vision. Computer vision algorithms seek to identify patterns, such as illumination, color distributions, edges, or shapes in images [76]. Since Radford et al. [63] introduced deep convolutional generative adversarial networks (DCGANs), most GAN adopt a CNN-based architecture [59]. Therefore, the proposed *CountEx-VQA* method also uses CNNs, whose functionality is explained in the subsequent subsections. The basic architecture of CNNs is based on three types of layers: (i) convolution layers, (ii) pooling layers, and (iii) fully-connected layers [56].

Convolution Layer

The eponymous convolution layer is the central building block of a CNN [26]. Géron [26] points out that, in contrast to perceptrons, a convolution layer's neurons are not connected to every neuron of the previous layer, but only to a small rectangle of neurons. The author states that, in order to obtain features from an input image (or a feature map at layers past the first one), the convolution layer performs a dot product between two matrices: The first matrix represents a set of learnable parameters, the so-called *kernel*. The second matrix is a segment of the input of the size of the convolution layer's receptive field [26]. Usually, the kernel is spatially smaller than the input image but has greater depth [21]. Digital images are represented by matrices of size $h \times w \times d$, where h is the image's height, w is its width and d corresponds to its color channel depth ($d = 3$ in case of RGB images). During the forward pass, the *kernel* K slides across the input image to produce a 2D feature map f [26]. Thereby, the size of the kernel's slides is determined by the *stride* parameter. According to Géron [26], it is common practice to add zeros around the inputs of a convolution layer, which is known as *zero-padding* and ensures that a layer has the same size as the previous layer. The output of a neuron in a convolutional layer is computed using the following equation [26]:

$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_d} x_{i',j',k'} \times w_{u,v,k',k},$$

where

- $z_{i,j,k}$ represents the output of the neuron located in the ith row, jth column in the kth features map of the lth convolution layer.
- f_h and f_w are the receptive field's height and width, respectively.
- f_d represents the number of feature maps in the previous layer (or channels if the previous layer is the input layer).

- $x_{i',j',k'}$ corresponds with the output of the neuron located in the i'th row, j'th column, k'th feature map (or channel if the previous layer is the input layer) of layer $l-1$.
- b_k is the bias term for feature map k in layer l.
- $w_{u,v,k',k}$ is the weight connecting any neuron in the kth feature map of the lth layer and its input located at the uth row, vth column of the neuron's receptive field, and the k'th feature map.

Pooling

According to Goodfellow et al. [21], each layer of a CNN usually comprises three stages. First, CNN stacks a cascade of convolution layers that produce a set of linear activations. Next, a nonlinear activation function (e.g., using a ReLU layer) is applied to each linear activation. Finally, the output is further processed using a pooling layer [21]. Géron [26] suggests that the pooling layer's main purpose is to *subsample* (i.e., to shrink) the input image. The author states that the effect of shrinking the input is three-fold: First, it decreases the computational load and the memory usage during backpropagation in a CNN. Second, subsampling reduces the number of parameters that need to be trained, making a neural network less prone to overfitting. Finally, shrinking the input allows for a small degree of *location invariance*. As a result, small amounts of changes in the input would leave most pooled outputs untouched [26]. This effect can be beneficial, for instance, if a CNN should learn whether a particular feature is present in the input rather than exactly where it is [21].

Géron [26] points out that, similar to the way convolution layers work, each neuron in a pooling layer connects to a subset of neurons in the previous layer using a rectangular receptive field, whose size, stride, and padding mode are hyperparameters. The author suggests that, in contrast to the convolution layer, the pooling layer does not include weights but applies one of two types of aggregation functions to its inputs. When using *max pooling*, the pooling layer passes only the maximum value inside each receptive field to the next layer while dropping all other values. For example, a pooling layer with a 2×2 kernel and a stride of 2 would subsample a 2D input by the factor of 2 on both dimensions. Hence, the pooling operation would drop $\frac{3}{4}$ of the values, resulting in the output feature map having an area four times smaller than the input. The other pooling mode is *average pooling*, which follows the same principle but outputs the mean of a given receptive field rather than its maximum value [26].

2.3. Autoencoders

Autoencoders are NNs that aim to randomly generate new data that is similar to its inputs [26]. To this end, these networks learn to model an efficient representation

of the inputs and map them to outputs that follow a similar distribution [26]. These networks typically consist of two parts [21]: Given an input $\mathbf{x}$, an *encoder* learns a function $\mathbf{h} = f(\mathbf{x})$, where $\mathbf{h}$ denotes a hidden layer that describes a representation of the input (called *coding*). Next, the *decoder* part takes the coding as input and learns to produce a reconstruction $\mathbf{r} = g(\mathbf{h})$ of the input. In order to prevent the autoencoder to simply learn the identity function $g(f(\mathbf{x})) = \mathbf{x}$, it is constrained to copy the input only approximately [21]. For instance, Géron [26] suggests that limiting the dimensionality of the coding described by $\mathbf{h}$ will allow the network only to approximate the input data, rather than trivially copying it, which is referred to as *undercompleteness*. The author points out that to enable the autoencoder to reconstruct the input, the cost function applied during training typically contains a *reconstruction loss*, which increases the more the output deviates from the input. Common choices of the reconstruction loss are the $\mathcal{L}_{L1}$ and the $\mathcal{L}_{L2}$ regularization terms [26]:

$$\mathcal{L}_{L1} = \mathbb{E}_{\mathbf{x},\mathbf{r}}[||\mathbf{x} - \mathbf{r}||_1],$$
$$\mathcal{L}_{L2} = \mathbb{E}_{\mathbf{x},\mathbf{r}}[||\mathbf{x} - \mathbf{r}||_2^2].$$

Regarding its architecture, an autoencoder typically resembles an MLP, except that the number of neurons in the output layer must match the number of inputs [26]. Géron [26] suggests that if an autoencoder has more than one hidden layer, it is called *deep autoencoder*. In this case, the author states that the encoder and the decoder are typically symmetrical with respect to $\mathbf{h}$: Each layer i has the same number of neurons as layer $n - i$, where n denotes the total number of layers. According to the author, a special type of autoencoder is the *adversarial autoencoder*, in which an encoder-decoder network learns to reproduce the input. Concurrently, a second NN is trained adversarially to judge whether the reproductions are real or fake. These models are also referred to as GANs [26].

2.4. Generative Adversarial Networks

In 2014, Goodfellow et al. [22] first introduced the notion of GANs. According to the authors, GANs aim at artificially generating new data that follows a similar distribution as the training data. For example, if the training data comprises images of hand-written digits, the goal is to generate new images that depict digits similar to the original ones. To this end, an adversarial process is used to train two models in parallel [9]:

- A **Generator** $\mathcal{G}$ learns to produce new examples based on the training data.
- A **Discriminator** $\mathcal{D}$ is trained to distinguish between real examples from the original data distribution and fake ones produced by the Generator.

Brownlee [9] suggests that to generate new examples that follow a similar distribution as the training data, generator and discriminator compete against each other

in a zero-sum game. According to the author, the discriminator receives both real and fake data to learn to distinguish them. Concurrently, the generator is trained to fool the discriminator by generating realistic synthetic examples. The training data comprises a set of points denoted $\chi \subset \mathbb{R}^n$ [83]. For instance, given a set of images with size 4×4, each image corresponds to a point $x \in \chi \subset \mathbb{R}^n$ with $n = 16$. According to Wang [83], enabling the generator to create realistic synthetic data points requires the training data to follow a certain probability distribution $\mu \to \mathbb{R}^n$. Thereby, in most applications of GANs, it is assumed that μ has a continuous density function $p(x)$. The author suggests that the similarity between the original dataset χ and a set of generated data points ψ can then be defined in terms of the probability distribution: the goal is to find a probability distribution ν with density $q(x)$ that approximates μ as closely as possible. An optimal generator would create new examples such that $p(x) = q(x)$. However, since the probability distribution μ of the training data is unknown, GANs aim to find ν by learning μ from a finite set of training samples χ. The more complex the distribution of the training data, the more training samples $x \in \chi$ are necessary for a GAN to generate a set of new examples ψ with a probability distribution that closely approximates μ [83]. To train a GAN, two objective functions $J_G(\theta G; \theta_D)$ and $J_D(\theta_D; \theta_G)$ are simultaneously updated in an iterative fashion [13]. The cost of the GAN's training is monitored using a value function $v(\mathcal{G}, \mathcal{D})$ that depends on both networks and aims at solving the following equation [13]:

$$\max_{\mathcal{D}} \min_{\mathcal{G}} v(\mathcal{G}, \mathcal{D}).$$

The default choice for v in GAN training reads as follows [13]:

$$v(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{P(x)} \log \mathcal{D}(x) + \mathbb{E}_{q(x)} \log(1 - \mathcal{D}(x))$$

Keeping the generator unchanged, there exists a unique optimal discriminator $\mathcal{D}^*(x) = \frac{p(x)}{p(x)+q(x)}$ [22]. Furthermore, Goodfellow et al. [22] point out that when $v(\mathcal{G}, \mathcal{D})$ converges, the discriminator is no longer required, as its output will always be 0.5. According to the authors, convergence is guaranteed in case $\max_{\mathcal{D}}$ is convex in θ_G. However, they argue that in practice $\max_{\mathcal{D}}$ is often not convex, which makes training GANs a challenging task in which $\mathcal{G}$ and $\mathcal{D}$ may never reach an equilibrium.

According to Goodfellow et al. [21], GANs usually have a CNN-based architecture when applied in the context of synthetic image generation. However, the authors point out that they use a different convolution operation as in image recognition tasks, in which the CNN learns a more high-level representation of the input image at each layer, continuously dropping information using pooling operations. Contrarily, according to the authors, GANs aim at adding rich details as the input progresses through the network before generating the final image representation at the last layer. To this end, convolutional GANs use the *transpose* of the convolution operator, which is often referred to as *upsampling* or *deconvolution*. To upsample a

feature map, the generator applies the inverse of the max-pooling operation under the following constraints [21]:

1. The max-pooling operation's stride must be equal to the pooling region's width.
2. Each pooling region's maximum input should be the upper-left corner.
3. Each pooling region's non-maximum inputs must be zero.

Using an integer value k to specify the size of the pooling region, the inverse pooling operation copies each value from the input's spatial coordinate i to the output tensor's spatial coordinate $i \times k$ (initialized at 0) [21].

3. Related Work

3.1. Interpretable Machine Learning

Throughout the past decades, the notion of interpretability increasingly gained attention in the ML domain [10, 71, 49, 88, 79, 46, 89, 29, 19, 16]. According to Kim et al. [29], interpretability is particularly important with respect to systems whose decisions have significant impact. Similarly, Chakraborty et al. [10] suggest that the increasingly important role ML methods play in decision-making processes in healthcare, criminal justice systems, finance, or policy-making point to the necessity of interpretability. Doshi-Velez and Kim [16] propose that interpretability in ML serves several purposes:

1. It allows to better protect certain groups from being (explicitly or implicitly) discriminated against.
2. Understanding how input or parameter variations affect an algorithm's performance fosters a system's reliability and robustness.
3. Interpretability helps increase users' trust in a system by allowing them to assess its provided assistance suitably.

While scholars seem to agree upon the necessity of interpretability in ML, Lipton [39] argues that a flurry of distinct definitions of the concept exists. Similarly, Murdoch et al. [50] suggest that the notion of interpretability is broad. The subsequent paragraph provides an overview of how researchers in ML define the term interpretability.

Defining Interpretability

Kim et al. [29] suggest that an ML method is interpretable if it allows a human to consistently and correctly classify its outputs. Similarly, Chakraborty et al. [10] define interpretability as a human's ability to understand and reason about a model's output. According to Gilpin et al. [19], a system is interpretable if it produces meaningful descriptions so that a human can understand its internals. Comparably, Doshi-Velez and Kim [16] claim that a method is interpretable if it comes with a human-understandable explanation. According to Murdoch et al. [50], interpretable ML refers to 'the use of machine-learning models for the extraction of *relevant* knowledge about domain relationships contained in data.' Lipton [39] distinguishes between two distinct categories of model interpretability: transparency and post-hoc explanations. First, transparency refers to a model whose underlying mechanisms are human-understandable. A system's transparency can be defined at the entire model's level, only a part of it (e.g., its parameters), and the algorithm used to train it. Second, post-hoc interpretability refers to models that provide human-understandable explanations of why they produced a certain output rather

than elucidating precisely how they work internally. 'Some common approaches to post-hoc interpretations include natural language explanations, visualizations of learned representations or models, and explanations by example (e.g., *this tumor is classified as malignant because to the model it looks a lot like these other tumors*' [39]. Similarly, Moraffah et al. [48] also distinguish between (i) inherently interpretable models that provide explanations during training or the decision-making process and (ii) generating post-hoc explanations for an already existing model. Furthermore, they propose causal interpretable models, which allow understanding the real causes of decisions made by an ML algorithm rather than only capturing statistical correlations. Moreover, the authors claim that causal interpretability can improve ML models' performance, especially when used in real-world environments. Pearl [62] introduces a three-layer hierarchy of causal interpretability: (i) associational, (ii) interventional, and (iii) counterfactual interpretability. First, associational interpretability helps to answer *What is?* questions (e.g., *What does a symptom tell me about a disease?*. Second, interventional interpretability refers to *What if?* questions (e.g., *What if I take aspirin, will my headache be cured?*. Finally, the author suggests that counterfactuals are the most powerful type of causal interpretability as they allow to answer questions such as *Why did X happen?, Was it X that caused Y?*, or *What if I had acted differently?*. Counterfactuals allow assessing what changes to a given situation would be needed to arrive at an alternative outcome [24].

Throughout the past decades, research in interpretable ML has made significant advances. In the early 2000s, Breiman [8] introduced the built-in feature importance measure of random forests. Around fifteen years later, when Deep Learning (DL) methods had already become prevalent in artificial intelligence, many researchers began to develop methods for both model-agnostic and model-specific explanations [46]. The following section provides an overview of state-of-the-art research in ML interpretability. In line with Moraffah et al. [48], I categorize interpretable models into three main categories: inherently interpretable models, post-hoc-interpretability, and causal interpretable models.

Inherently Interpretable Models

ML models are inherently interpretable if they provide explanations during the training procedure or while they generate an output. One type of inherently interpretable models follows a collection of *if...then...* rules to make predictions. For instance, decision trees use *if...then...* rules by checking whether a condition of a feature holds or not at each internal node. Decision trees make inferences by tracing a path from the root to a leaf node representing the class label [48]. However, Liu et al. [41] argue that decision trees are often complex and cumbersome, which lowers their interpretability for humans. Similar to decision trees, rule-based models build on a collection of rules and often serve classification purposes [40]. In contrast to decision trees, these models' rules do not need to be mutually exclusive and exhaustive and may have a priority-based hierarchy. As a result, a sample might

trigger multiple rules at the same time, or even no rule at all [48].

In addition to rule-based methods, both linear and logistic regression are interpretable models. Linear regression models learn a linear relationship between the target and a weighted sum of the feature inputs. The linearity inherent in these models allows for a straightforward interpretation. Contrarily, logistic regression learns a logistic function to compute probabilities between 0 and 1 for classification problems and is non-linear. However, computing the so-called log-odds (i.e., the logarithm of the probability of an event relative to the probability of no event) results in a linear function that is interpretable [46]. Other examples of inherently interpretable models are Attention Networks and Disentangled Representation Learning. Since they report a weighting over inputs or internal features, Attention Networks highlight their importance for making a specific inference. These networks are used for various tasks such as machine translation, graph embedding, image classification, and VQA. Disentangled Representations separate latent factors that highly correlate with meaningful patterns from those of lower importance. Principal Component Analysis, Independent Component Analysis, and Nonnegative Matrix Factorization are examples of techniques that learn to discover disentangled features of data [48, 46].

The list of inherently interpretable ML models is growing consistently. Other examples include Generalized Linear Models (GLMs), Generalized Additive Models (GAMs), Naive Bayes Classification, or K-Nearest Neighbors [46]. An exhaustive discussion of these methods is beyond the scope of this thesis. The recently published surveys by Molnar [46] and Gilpin et al. [19] provide a detailed overview of current methods.

Post-Hoc Interpretability

While inherently interpretable models include explanations by design, post-hoc methods aim to generate explanations for trained black-box ML models [48]. One way to create such explanations is to use local surrogate models. These models represent an approach to explain individual predictions of a trained ML model to understand its decision-making process [46]. Ribeiro et al.'s [64] Local Interpretable Model-agnostic Explanations (LIME) is a model-agnostic implementation of a local surrogate approximating a black-box model's outputs by examining how variations in the training data affect its predictions. Particularly, LIME permutes a trained black-box model's training samples to generate a new dataset. Based on the black-box model's predictions on the permuted dataset, LIME ' trains an interpretable model, which is weighted by the proximity of the sampled instances to the instance of interest' [46].

Another method to increase the interpretability of black-box models is to use feature visualization. This method aims at visualizing those features that maximize the activation of a NN's unit [46]. Conceptually, feature visualization methods iteratively compute the derivatives of a NN's layers to figure out which of the input's

features impact the model's prediction the most [55]. In image classification, using saliency maps is one method to highlight the pixels that have the most significant impact on a model's prediction [48]. For example, Montavon et al. [47] propose a method that uses deep taylor decomposition to backpropagate the explanations of a DNN from the output to the input layer by computing the derivative of the weight vector. Similarly, Simonyan et al.'s [72] *image-specific class saliency maps* build on a single backpropagation pass to extract the derivative of a DNN's weight vector. 'The magnitude of the derivative shows the importance of each pixel for the class score' [48].

Research in psychology suggests that providing examples is an effective strategy for human learning and decision-making [1]. Hence, another method commonly used to increase interpretability in ML is to generate example-based explanations for complex data distributions. This approach aims to find prototypes in a training dataset to summarize it [29]. To this end, an algorithm (e.g., k-nearest neighbors) selects an instance from the dataset representing the ML model's prediction or the data distribution [48]. However, Kim et al. [29] suggest that examples alone are not enough when it comes to making ML methods interpretable since real-world data are heavily complex and seldom contain representative prototypes. The authors thus propose a method called Maximum Mean Discrepancy (MMD) to additionally identify some *critisism* samples that deliver insights regarding those prototypes the black-box model does not capture [29].

Finally, another common approach is to use inherently interpretable models to estimate a black-box model's predictions. To this end, these so-called global surrogates aim at approximating a 'prediction function f as closely as possible with the surrogate model prediction function g' [46]. Thereby, the surrogate model's function must be interpretable, such as a decision tree or a linear regression model [46, 48].

Causal Interpretability

While inherently interpretable models and post-hoc interpretability methods foster the human understanding of how and why an ML algorithm reached a certain decision, they cannot answer questions as to how the algorithm would behave under alternative conditions (e.g., being trained with different data). Causal interpretable models address questions such as *Why did the model make this decision instead of another one?* or *How would the prediction change if we had a different input to it?*. Finding answers to these questions is necessary for a variety of applications. For instance, understanding whether certain socio-demographic features such as a person's gender or age caused a system to deny a credit application might be relevant to ensure fairness [48]. The underlying statistical concept of causal interpretability is causal inference, which aims at extracting causal relationships from data (i.e., investigating how and whether changes in one variable cause an effect in another variable) [50]. According to Moraffah et al. [48], causal interpretability for model-based interpretations refers to methods that explain to what extent a black-box model's compo-

nents caused its predictions. One way to achieve this goal is to compute the average causal effect (ACE) of an NN's neurons on the output. Another method to provide causal interpretability is to generate counterfactual examples. Counterfactual explanations aim at demonstrating how changes in a feature would have affected the model's prediction. These explanations emerge by making minimal modifications to an original instance so that the model's prediction changes. For instance, a counterfactual could be an image that is minimally different from the original image but labeled differently by the black-box model [48]. Pearl [62] suggests that generating counterfactuals allows for the highest degree of interpretability among all methods to explain black-box models. Consequently, many scholars proposed methods to generate counterfactuals throughout the past years [24, 15, 20, 73, 82, 27]. For instance, in their paper *Counterfactual Visual Explanations*, Goyal et al. [24] propose a method that identifies how a given image I could change so that the image classifier outputs a different class by replacing the key discriminative regions in I with pixels from an identified "distractor" image I' that has a different class label. Similarly, Gomez et al. [20] present an interactive visual analytics tool, *ViCE*, that identifies the minimal changes to a given sample that are needed to make the model predict a different output.

3.2. Visual Question Answering (VQA)

Throughout recent years, a series of papers from the ML community addressed the task of VQA [75, 5, 23, 87, 4, 11, 18, 42, 25, 75]. According to Antol et al. [5], VQA methods aim at answering natural language questions about an input image. The combination of image and textual data makes VQA a challenging multi-modal task that involves both image understanding and natural language processing [75]. Particularly, image understanding requires techniques from computer vision, while NLP methods are needed to understand the question [25]. On the one hand, computer vision algorithms aim at recovering the properties of objects in images, such as shapes, illumination and color distributions [77]. On the other hand, NLP refers to computational techniques for analyzing and representing natural language texts with the goal of achieving human-like language processing to make computer systems understand and manipulate them [36, 12]. Finally, generating an answer for a given question about an input image requires a combination of computer vision and NLP techniques [25]. The answer's format can be of several types: a word, a phrase, a binary answer, a multiple choice answer, or a fill in the blank answer [25, 75].

The subsequent section provides an overview of the state-of-the-art in VQA, followed by a section that discusses existing attempts to increase VQA systems' interpretability.

3.2.1. Methods for VQA

The first attempts at solving the VQA task share the characteristic of being restricted to answering questions of predefined forms [85, 42, 80, 18]. Malinowski and Fritz's

[42] method requires human-produced question-answer pairs for training. The authors propose a system that models the probability of an answer given a question and an image by sampling from the nearest neighbors in the training set. In [80], Tu et al. focus on graph-based joint parsing from videos and narrative text descriptions to answer queries of the type *who*, *what*, *when*, *where*, and *why* (e.g., "Who has a cap?") using a query parser. Geman et al. [18] introduced a system that automatically generates a series of binary questions for a given image.

In contrast to earlier contributions, more recent VQA approaches aim at generating answers to free-form open-ended questions [85]. First, Agrawal et al. [5] propose a system that mirrors the human perception of real-world environments. To this end, the model classifies an answer to a given question about an image by combining a CNN to extract features from the image and Long Short-Term Memory (LSTM) or recurrent networks for language processing. Their model, referred to as *Vanilla VQA*, can be considered as a benchmark for DL-based VQA methods [75]. In [86], Yang et al. introduce *Stacked Attention Networks* (SANs) that uses CNNs and LSTMs to compute an images' regions related to the answer based on the semantic representation of a natural language question. Similarly, Anderson et al. [4] introduce a combined bottom-up and top-down visual attention mechanism to narrow down the features in images. Specifically, their system uses top-down signals based on a natural language question to determine what to look for. Their method combines these signals with bottom-up signals stemming from a purely visual feed-forward attention mechanism.

Despite the continuous advancements in VQA, several papers suggest that VQA models tend to suffer from the language prior problem: models tend to achieve good superficial performances but do not truly understand the visual context [23, 87, 11, 92]. Specifically, Goyal et al. [23] suggest that in the VQAv1 dataset [5] blindly answering "yes" to any question starting with "Do you see a..." without taking into account the rest of the question or the image results in an accuracy of 87%. Therefore, the authors propose a novel balanced VQA dataset to counter language biases. Specifically, for a given (image,question,answer) triplet (I, Q, A) from the VQAv1 dataset [5], they asked humans to identify a similar image I' for which the answer to question Q is different from A. For the same reason, Zhang et al. [87] propose a balanced VQA dataset for binary questions. For each question, they collected pairs of images showing abstract scenes so that the answer to the question is "yes" for one image and "no" for the other. Likewise, Chen et al. [11] criticize that existing VQA models tend to capture superficial linguistic correlations between questions and answers in the training data and, hence, only offer limited generalizability to unseen question-answer sets. Therefore, in their recent paper, they propose a model-agnostic Counterfactual Samples Synthesizing (CSS) training scheme that aims at improving VQA models' visual-explainable and question-sensitive abilities. The CSS algorithm masks (i) objects relevant to answering a question in the original image to generate a counterfactual image and (ii) critical words to synthesize a counterfactual question.

In the same context, Zhu et al. [92] propose a self-supervised learning framework that balances the training data. The system first identifies whether a given question-image pair is relevant (i.e., the image contains critical information for answering the question) or irrelevant. This information is then fed to the VQA model to overcome language priors.

3.2.2. Interpretable VQA

Although several existing approaches attempt to overcome the language prior problem to create less superficial VQA models, they rarely provide human-understandable explanations regarding the underlying processes that led to an answer. In most real-world scenarios, human users want to get an explanation along with a VQA system's output, especially if it fails to answer a question correctly [35]. To this end, various scholars proposed methods to make VQA systems more interpretable [35, 58, 53, 14, 90]. In [35], Li et al. introduce a method that mimics the human question-answering process. First, the authors apply pre-trained attribute detectors and image captioning to extract attributes and generate descriptions for a given image. Second, the generated explanations are used instead of the image data to infer an answer to a question. Providing critical attributes and captions to the end-user allows them to understand better what the system extracts from the image. While Li et al. [35] use textual descriptions to provide an explanation, Zhang et al. [90] introduce a heat map-based system to showcase a user the image's regions relevant to the question. To this end, the authors train a model using region descriptions and object annotations provided in the Visual Genome dataset [31]. To increase human interpretability, Pan et al. [58] introduce a method that provides counterfactual images along with a VQA model's output. Concretely, for a given question-image pair, the authors' system generates a counterfactual image that is minimally different from the original image and visually realistic but leads the VQA model to output a different answer for the given question. In its current form, their method is restricted to the context of color questions. Furthermore, since their model makes edits on a pixel-by-pixel level, the counterfactual images contain changes also in areas irrelevant to a given question.

4. CountEx-VQA

4.1. Overview

This thesis introduces *CountEx-VQA - Counterfactual Explanations for Visual Questions Answering*, a method to provide human-interpretable discriminatory explanations to increase a VQA system's interpretability. It aims at providing human users a visual explanation for the following intuitive question: *How would the image look like if the model's answer was different?* Concretely, given an image-question pair (I, Q) and a VQA model $f : (I, Q) \rightarrow A$, where A is its predicted answer, the goal is to train a model $\mathcal{G}$ to generate a new image I':

$$\mathcal{G} : (I, Q, A) \rightarrow I'$$

Thereby, I' is a counterfactual image so that $f : (I', Q) \rightarrow \hat{A}$ and $A \neq \hat{A}$. Along the lines of Pan et al. [58], the new image I' should be (i) minimally different from I, (ii) visually realistic, and (iii) contain semantically meaningful edits (iv) applied only in question-relevant image regions. Intuitively, the model observes the original image I along with question Q and answer A and generates a new image I' for which the VQA model outputs a different answer than for the original image. To ensure that the system's modifications in the original image I are applied only to semantically relevant regions, the method should only edit question-critical objects. To this end, the method includes an attention mechanism that first identifies each pixel's contribution to the VQA model's output and guides g regarding where to apply modifications in I. Finally, the generator should output a semantically meaningful counterfactual image I' for which the VQA model returns a different answer than for the original image, keeping the question constant.

The proposed model is an extension of the counterfactual GAN introduced by Pan et al. [58]. Specifically, I applied the following additions to the original model aiming to (i) allow for semantic edits beyond color modifications (e.g., editing shapes) and (ii) ensure that only the question-critical regions in an image are altered while retaining the background:

- An attention mechanism guides the generator regarding where to apply edits.
- I introduce a weighted reconstruction loss that allows for larger semantic edits in those spatial image regions which the VQA model paid most attention to when answering the question (i.e., the question-critical image regions) than in the rest of the image.
- Since providing GANs with additional information can foster their performance [28], both the generator and the discriminator are conditioned on an embedding of the VQA model's final logits weight vector. This conditioning functions as a control mode to direct the data generation process with respect to the VQA model's prediction to create a counterfactual image that contains semantically meaningful changes relative to the original image.

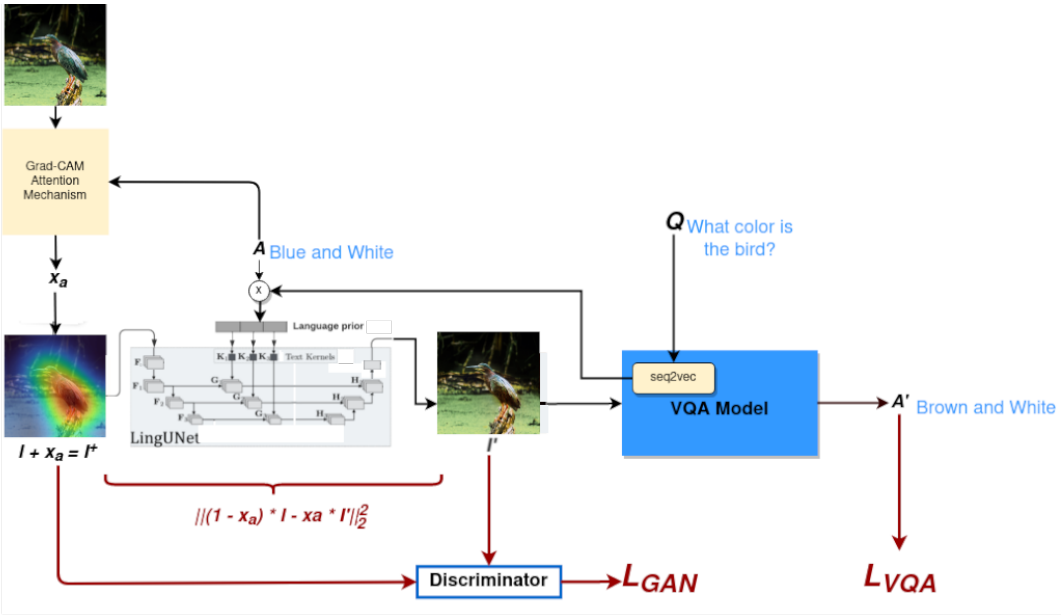


Figure 1: Overview of the general architecture of *CountEx-VQA*'s architecture inspired by Pan et al. [58].

- Spectral normalization [45] is added to both the discriminator and the generator to stabilize the model's training procedure.[2]

Extending the constraints applied by Pan et al. [58], $\mathcal{G}$ must fulfill the following prerequisites:

- **Prerequisite I:** Given the same question, the VQA model's output should be different for the counterfactual image I' than for the original image I.
- **Prerequisite II:** The differences between I and I' should be minimal.
- **Prerequisite III:** I' should be visually realistic and contain semantically meaningful modifications.
- **Prerequisite IV:** The model $\mathcal{G}$ should primarily edit pixels that fall in those regions in I that are relevant to Q, while retaining other pixels unchanged.

Figure 1 displays the basic architecture of *CountEx-VQA*, which is described in detail in the subsequent sections. In the depicted example, to explain why a model predicted the bird to be *Blue and White*, the generator should output a new image I' that is similar to the original image I but results in a different answer (i.e., *Brown and White* in this case). To this end, the model conducts a series of steps:

[2] An existing implementation of spectral normalization was used: `https://github.com/christiancosgrove/pytorch-spectral-normalization-gan/blob/12dcf945a6359301d63d1e0da3708cd0f0590b19/spectral_normalization.py`

1. **Computing an attention map that guides the generator:** To ensure that the model meets the desired prerequisite to predominantly edit regions in I that are relevant to the question Q, the proposed method aims to identify the question-critical objects in I. To this end, an attention mechanism is used to determine each pixel's importance regarding the VQA model's decision. Specifically, the Gradient-weighted Class Activation Mapping (Grad-CAM) algorithm [69] is applied to the VQA model's final convolution layer, which results in a continuous attention map $x_a \in \mathbb{R}^{1\times h\times w}$ between [0,1], where h and w correspond to the input image's height and width, respectively. The concatenation $[I, x_a]$ of image I and attention map x_a serves as an input to the generator $\mathcal{G}$.

2. **Language-Conditioned Counterfactual Image Generation:** Following Pan et al. [58], the generator $\mathcal{G}$ uses an architecture based on LingUNet [44]. LingUNet is an encoder-decoder NN similar to the popular pixel-to-pixel UNet model [65] and maps conditioning language to key intermediate filter weights based on an embedding of natural language text. Similar to Pan et al. [58], I used a concatenation of the VQA model's question encoding and its final logits weight vector regarding its prediction for image I as the language embedding. This embedding serves as an additional input to $\mathcal{G}$. Furthermore, the VQA model's final logits weight vector given image I and question Q is also passed to $\mathcal{G}$ as a conditional class embedding to direct its generation process towards negating the prediction. LingUNet performs a series of convolution and deconvolution operations to generate a new image $\hat{I}$. The final counterfactual I' is retrieved as follows:

$$I' = x_a \odot \hat{I} + (\mathbf{1} - x_a) \odot I,$$

   where $\odot$ denotes the element-wise multiplication and $\mathbf{1}$ is an all-ones vector that has the same size as x_a. Intuitively, I' is created by adding the *foreground* of $\hat{I}$ (i.e., pixels with large attention values have a higher intensity than those with low attention values) to the original image's *background*.

3. **Constraining Image Generation**: The image generation process is subject to several constraints that ensure that the generator meets the desired prerequisites. Similar to Pan et al. [58], the generator is constrained under three losses:

   a) I' should contain semantically meaningful differences compared to I, such that the VQA model outputs two different answers for I' and I given the same question Q. Using the VQA model's negated cross-entropy with A as the target trains the generator for any answer other than A.

   b) Although I and I' should have distinct semantics with respect to a given question Q, the differences between the two images should be minimal. This constraint is achieved by incorporating a reconstruction loss, which

penalizes the generator for creating outputs different from the input. To ensure that question-critical objects can change their semantic meaning, the generator should be allowed to make significantly more changes in the corresponding image regions (i.e., the *foreground*) than in the rest of the image (i.e., the *background*). A modified ℓ_2-loss adapted to this purpose, which incorporates the attention map computed in step (1) as a relative weighting term, acts as the reconstruction loss:

$$\ell_2 = [||(1 - x_a) \odot I - (1 - x_a) \odot I'||_2^2].$$

Applying a weighted reconstruction loss aims at contributing to the desired traits that (i) the model predominantly edits critical objects and (ii) a relatively loose ℓ_2 constraint is applied to question-critical regions, allowing for more significant semantic edits. Contrarily, the stricter ℓ_2-constraint for question-irrelevant regions ensures that the generator retains them nearly unchanged.

c) The counterfactual images generated by $\mathcal{G}$ should be visually realistic. To this end, a PatchGAN discriminator as proposed by Isola et al. [28] is used. It learns to distinguish between real and fake images and penalizes unrealistic generated counterfactuals. The generator and the discriminator are trained in an adversarial manner as in GAN training [22].

4. **Applying Spectral Normalization to stabilize training**: Training GANs can suffer from instability, and they are particularly vulnerable to the problems of exploding and vanishing gradients [38]. In their approach, Pan et al. [58] applied gradient clipping to counter this problem, which requires extensive empirical fine-tuning of the training regime. To bypass this extensive procedure, the proposed method uses *spectral normalization* [45, 38] instead to counteract training instability. Moreover, Miyato et al. [45] suggest that using spectral normalization in GANs can lead to the generated images having a higher quality relative to other training stabilization techniques, such as gradient clipping.

4.2. Implementation Details

The counterfactual generator aims to translate images from a source domain $\chi \subset \mathbb{R}^n$ to the target domain $\psi \subset \mathbb{R}^n$. To this end, it estimates a probability distribution ν with density $q(x)$ that approximates the distribution μ with density $p(x)$ of the source domain χ as closely as possible. The LingUNet [44] based approach to generate counterfactual images introduced by Pan et al. [58] functions as a starting point for *CountEx-VQA*. Since their implementation is not publicly available, I reimplemented it from scratch and applied several changes. The following subsections describe each of the method's components: (i) the attention-mechanism, (ii) spectral normalization, (iii) the architectures of generator $\mathcal{G}$ and discriminator $\mathcal{D}$,

and (iv) the training procedure. The proposed method is implemented using Python version 3.8.10[3] and PyTorch 1.8.0[4].

4.2.1. Attention Mechanism

Qustion: What color are the man's shorts?

Figure 2: Example question-image pair from the VQAv1 dataset [5]. The red bounding box indicates the question-critical object.

Complex images may contain various objects, of which usually only a few are relevant when answering a given question. In other words, an object can be considered to be question-critical if it is key to finding an answer to a given question. For example, given the question "What color are the man's shorts?", the question critical object in Figure 2 is the man's shorts. Hence, the first component of the proposed architecture consists of determining question-critical image regions. Similar to Mejjati et al. [3], the input image $I \in \chi$ is passed to an attention network A_χ that outputs an attention map $x_a = A_\chi(I)$. x_a is a continuous attention map between [0,1] which determines each pixel's intensity. The intuition behind this approach is to identify those spatial regions in an image that are most relevant to answer a given question. To generate the attention map, I incorporated the Grad-CAM algorithm [69]. Grad-CAM is suitable for various CNN-based models and, hence, can be applied to most VQA models. Convolutional layers retain spatial information that is not kept by fully connected layers. Hence, Selvaraju et al. [69] expect a CNN's last convolutional layer to have the finest balance between high-level semantics and fine-grained spatial information. Grad-CAM exploits this property by finding the gradient of the most dominant logit (i.e., in the case of a VQA model, this corresponds to the answer with the highest probability) that flows into the model's final activation map. Intuitively, the algorithm computes the importance of each neuron activated in the

[3] https://docs.python.org/3.8/reference/
[4] https://pytorch.org/docs/1.8.0/

CNN's final convolutional layer with respect to its prediction. Computing the gradient y^a of the logit corresponding to the VQA model's predicted answer a with respect to the kth feature map's activations A^k of a convolutional layer, i.e., $\frac{\delta y^a}{\delta A^k}$, reveals the localization map $L^c_{Grad-CAM} \in \mathbb{R}^{u \times v}$ of width u and height v. Next, channel-wise pooling with respect to the width and height dimensions is applied to the gradients. The pooled gradients are then used to weigh the activation channels. Finally, the weighted activations α^a_k reveal each channel's importance with respect to the VQA model's prediction [69]:

$$\alpha^c_k = \frac{1}{Z} \sum_u \sum_v \frac{\delta y^a}{\delta A^k_{ij}}$$

Performing a weighted combination of forward activation maps followed by a ReLU finally yields a coarse saliency map of the same size as the convolutional feature maps [69]:

$$L^c_{Grad-CAM} = ReLU(\sum_k \alpha^a_k A^k)$$

Finally, to obtain the attention map x_a, the feature maps $L^c_{Grad-CAM}$ are interpolated to match the size of the input image I. Furthermore, I applied a gaussian filter [7] with a mean $\mu = 0$ and a population standard deviation $\sigma = 2$ for improved preservation of the selected image regions' edges.

4.2.2. Spectral Normalization

Along the lines of Miyato et al. [45], I incorporated spectral normalization into *CountEx-VQA* to stabilize its training procedure. According to the authors, this method normalizes a NN's weights by applying the Lipschitz constraint. In general, a real function $g : \mathbb{R} \to \mathbb{R}$ is follows the Lipschitz constraint if

$$\frac{|g(x_1) - g(x_2)|}{|x_1 - x_2|} \leq k,$$

where k is the *Lipschitz constanz* (e.g., 1) [45]. Given a CNN $\mathcal{CNN}_\theta$ with L layers and weights $\theta = \{w_1, w_2, \ldots, w_L\}$, its output for an input x can be computed as [38]:

$$\mathcal{CNN}_\theta = a_L \odot l_{w_L} \odot a_{L-1} \odot l_{w_{L-1}} \odot \ldots \odot a_1 \odot l_{w_1}(x),$$

where a_i $(i = 1, \ldots, L-1)$ denotes the activation function in the i-th layer and a_L describes the final layer's activation function. Spectral normalization regularizes the convolutional kernels $w_i \in \mathbb{R}^{c_{out} \times c_{in} \times k_w \times k_h}$ with kernel width k_w and height k_h of the fully connected layers $l_{w_i}$with c_{in} input and c_{out} output channels. To this end, w_i is first reshaped into a matrix $\hat{w}_i \in \mathbb{R}^{c_{out}} \times (c_{in} \times k_w \times k_h)$, which is then normalized such that the spectral norm $||\hat{w}_i||_{sp} = 1 \, \forall i \in [1, L]$. Thereby, the spectral norm is computed as follows [38]:

$$||\hat{w}_i||_{sp} = \frac{\hat{w}_i}{u_i^T \times \hat{w}_i \times v_i},$$

where u_i and v_i denote the left and right singular vectors of $\hat{w}_i$ with respect to its largest singular values

4.2.3. Generator

For the generator $\mathcal{G}$, I used Pan et al.'s [58] approach as a starting point and incorporated the required modifications. Specifically, $\mathcal{G}$ uses a modified LingUNet [43] architecture. LingUNet builds on Ronneberger et al.'s [65] U-Net image generation method by incorporating language conditioning to the reconstruction phase. The generator receives the following inputs:

- A concatenation $[I, x_a]$ of the original image I and the corresponding attention map x_a based on the VQA model's prediction for the image-question pair (I,Q).
- A question embedding $\bar{q}$, which stems from the VQA model's language encoding of question Q.
- An answer embedding $\bar{a}$ represented by the VQA model's final logits weight vector with respect to its prediction for the image-question pair (I,Q).

All inputs are normalized to be within the range [-1,1] before they are passed to the generator. Furthermore, images are resized to 256×256 to decrease computational load.

Figure 3 depicts the general architecture of $\mathcal{G}$. First, as in [43], $\mathcal{G}$ applies a two-layer CNN with a kernel size of 4×4, stride 2, padding 1, and leaky ReLU to the input $[I, x_a]$. This operation yields a feature map $F_0 = \text{CNN}_0(x')$ of size $64 \times 128 \times 128$. Subsequently, $\mathcal{G}$ generates $m = 3$ feature maps $F_j = CNN_j(F_{j-1})$, $j = 1...m$ by applying a cascade of m convolutional blocks with leaky ReLU, dropout, and instance normalization to F_0. The layers use 4×4 kernel sizes, strides of 2 and paddings of 1.

Furthermore, $\mathcal{G}$ applies a series of operations to condition the image generation process on language. First, the question embedding $\bar{q}$ and the answer embedding $\bar{a}$ are concatenated to create a language representation $\bar{x}$. Second, $\mathcal{G}$ applies a 2D 1×1 convolutional filter with weights K_k to each F_j. Each K_k is computed by splitting $\bar{x}$ into m equally sized vectors $\{\bar{x}_j\}_{j=1}^m$ and applying a 1×1 linear transformation to each of them. Applying the filter weights to each feature map F_j yields the language-conditioned feature maps G_j [43].

Next, $\mathcal{G}$ applies a one-layer CNN with kernel size 4×4, stride 2, and a padding of 1 to G_m to produce the internal *coding* described by $\mathbf{h}$. Furthermore, since $\mathcal{G}$ should change the input x such that the VQA model's predicted answer changes, the generator is conditioned on the VQA model's prediction. To this end, $\mathcal{G}$ applies a $1 \times$

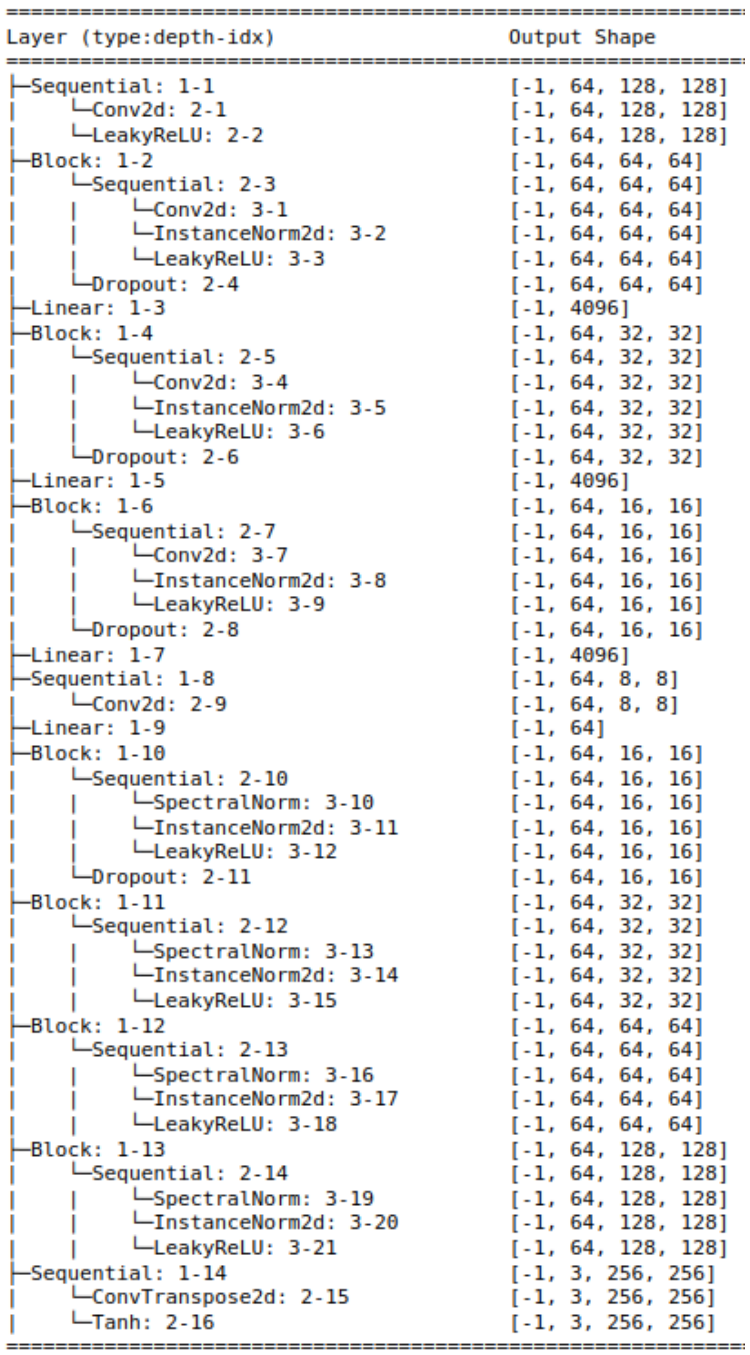


Figure 3: Architecture of the generator

1 linear transformation to the VQA model's final logits weight vector $\bar{a}$. The output is then concatenated with the output of $\mathbf{h}$ to generate the final convolution feature

map G_{m+1}. This concatenation is intended to direct the generator's upsampling procedure with respect to the VQA model's prediction by learning class-specific patterns. Providing this additional information aims at assisting the generator to apply modifications such that the VQA model perceives semantically meaningful changes. Finally, $\mathcal{G}$ applies a series of $m+1$ upscale and convolution operations to generate a sequence of $m+1$ feature maps of increasing size $H_{m+1}, ..., H_1]$ [43]:

$$H_{m+1} = \text{DECONV}_{m+1}((G_{m+1})$$
$$H_j = \text{DECONV}_j([H_{j+1}; G_j]).$$

Each DECONV_j is a deconvolution block including spectral normalization, instance normalization and a leaky ReLU activation function. Additionally, the generator uses dropout in the first two deconvolution operations to prevent the model from overfitting. Furthermore, in image translation, the input and output usually share a vast amount of low-level information (e.g., the location of prominent edges) [**?**]. Therefore, it is desirable to roughly align the structure in the input with the structure in the output. To this end, $\mathcal{G}$ uses skip connections between the equally-sized convolution and deconvolution layers. Each skip connection concatenates all channels of feature map G_j with all channels of feature map H_{j+1}. Next, $\mathcal{G}$ applies a 2D transposed convolution operation with Tanh [54] to the final deconvolution layer's output. This operation yields an image y' of size 256×256. Ultimately, the final output of $\mathcal{G}$ is computed as follows:

$$\mathcal{G}(x) = y = x_a \odot y' + (1 - x_a) \odot I$$

Intuitively, this operation adds the foreground (i.e., the question-critical regions) of y' to the background of the original image I. As a result, most of the original image's background is retained unchanged in y, while the foreground is replaced.

4.2.4. Discriminator

To penalize unrealistic generated images, *CountEx-VQA* adopts a modified version of the *PatchGAN* discriminator [28], which distinguishes real and fake images. The PatchGAN discriminator uses an $N \times N$ receptive field to map each $N \times N$ square of the output to the corresponding square in the input. Effectively, the discriminator $\mathcal{D}$ classifies each patch in an image as real or fake. The final output of $\mathcal{D}$ is then computed by averaging all patch-classifications. For the experiment at hand, I modified an existing PyTorch implementation of PatchGAN[5]. It uses a 70×70 receptive field to restrict the discriminator's attention to the structure in local image patches. Figure 4 depicts the discriminator's architecture. Similar to the generator, I incorporated a class embedding to condition the discriminator on the VQA

[5] `https://github.com/aladdinpersson/Machine-Learning-Collection/blob/master/ML/Pytorch/GANs/Pix2Pix/discriminator_model.py`

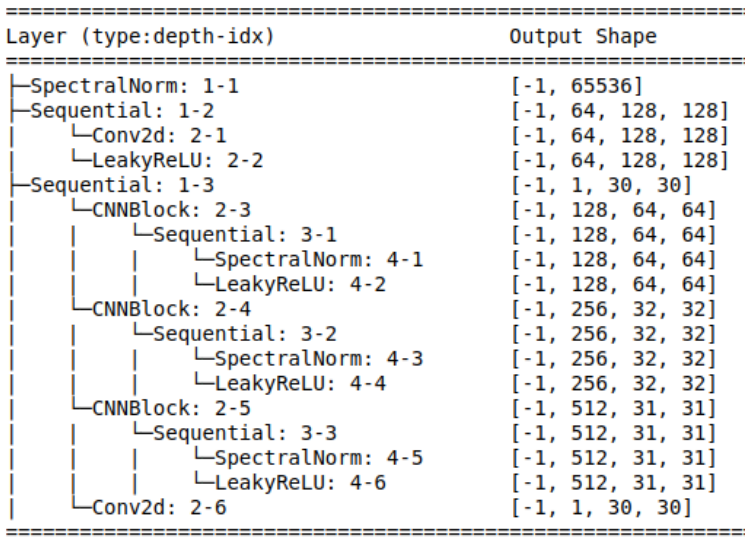


Figure 4: Architecture of the discriminator

model's prediction. Specifically, in addition to the input image I, the discriminator receives the normalized final logits weight vector $\bar{a}$ of the VQA model, which is passed through a 1×1 linear layer and concatenated with I. The concatenated inputs then flow through two three-layer CNNs. Each of them has a convolutional kernel size of 4×4, stride 1, and a padding of 1, and uses spectral normalization and a leaky ReLU activation function. The resulting feature map is then passed through an equally designed CNN, except for its stride, which is 2 instead of 1. Finally, an additional convolution layer with kernel size 4×4, stride 2, and a padding of one outputs the discriminator's final prediction.

4.2.5. Training procedure

As a starting point, I used the original adversarial GAN loss [22] to govern the training procedure of the proposed model:

$$\mathcal{L}(\mathcal{G}, \mathcal{D}) = \mathbb{E}_I[\log \mathcal{D}(I)] + \mathbb{E}_I[\log(1 - \mathcal{D}(\mathcal{G}(I)))].$$

Intuitively, $\mathcal{G}$ and $\mathcal{D}$ play an adversarial minimax-game, where the generator tries to minimize $\mathcal{L}(\mathcal{G}, \mathcal{D})$, while the discriminator aims at maximizing it, i.e. $\mathcal{G}^* = \arg\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{G}, \mathcal{D})$ **[?]**. In addition to the GAN [22] loss, I incorporated the following losses for $\mathcal{G}$ and $\mathcal{D}$ in order for the model to meet the specified prerequisites.

First, the generator should apply only the minimal amount of changes to the input image. To this end, I trained $\mathcal{G}$ using an adapted version of the ℓ_2 reconstruction loss **[?]**. Specifically, to allow for more extensive semantic edits in question-critical image regions, I incorporated a relative weighting of the ℓ_2 loss with respect to the

attention map x_a:

$$\ell_2 = [||(1 - x_a) \odot I - (1 - x_a) \odot I'||_2^2].$$

By enforcing the ℓ_2 constraint on the *backgrounds* of I and I', respectively, it allows for more edits in the *foreground* than in the rest of the image.

Second, to train the generator to produce images such that the VQA model's output changes, I incorporated the negated cross-entropy with the VQA model's original answer $\hat{A}$ as the target [21]:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{n} -a_i \log(p_i),$$

for n classes, where a_i is the truth label and p_i is the Softmax probability for answer a_i. As a result, the generator learns to aim for any answer $\hat{A}' \neq \hat{A}$. In line with Pan et al. [58], I warmed up the generator using only the adapted ℓ_2-loss for one epoch before adding the other two losses. As a result, the final loss function for the generator reads as follows, where $e \in [1, ..., n]$ with n representing the number of epochs:

$$\mathcal{L}_G = \begin{cases} \ell_2 & \text{if } e = 1 \\ \mathcal{L}_{GAN}(\mathcal{G}, \mathcal{D}) + \lambda_{\ell_2}\ell_2 + \lambda_{CE}\mathcal{L}_{CE} & \text{otherwise,} \end{cases}$$

with $\lambda_{\ell_2} = 0.05$ and $\lambda_{CE} = 5$.

The discriminator was trained using $\mathcal{L}_{GAN}(\mathcal{G}, \mathcal{D})$ only. Both the discriminator and the generator were trained for six epochs using the Adam optimizer described in section 2 with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ following [28]. To train the discriminator, a learning rate of $2e-4$ was used, while the generator was trained with a learning rate of $1e-4$. To prevent the discriminator from getting too strong too quickly, one-sided label smoothing was added to the discriminator as proposed by Salimans et al. [67]. Specifically, all the true labels (i.e., 1) were replaced with a smoothed value of 0.95.

4.3. Experimental Setup

4.3.1. Data

The approach was applied to a subset of the VQAv1 dataset's *Real Images* portion introduced by Agrawal et al. [5]. The dataset covers (i) images of everyday scenes, (ii) a wide variety of questions about the images, and (iii) the corresponding ground truth answers. The images stem from the Microsoft Common Objects in Context (MS COCO) [37] dataset.

Specifically, it comprises 204,721 images of real-life everyday scenes depicting common objects in their natural environment. Second, for each image in the MS COCO dataset, Agrawal et al. [5] gathered three questions about the scene it shows. To this end, the authors asked human subjects to formulate image-related questions that they believed a *smart robot* able to understand images would have trouble answering. The subjects were presented with previously asked questions about a given image to ensure question diversity . Overall, the VQAv1 dataset [5] includes 614,163

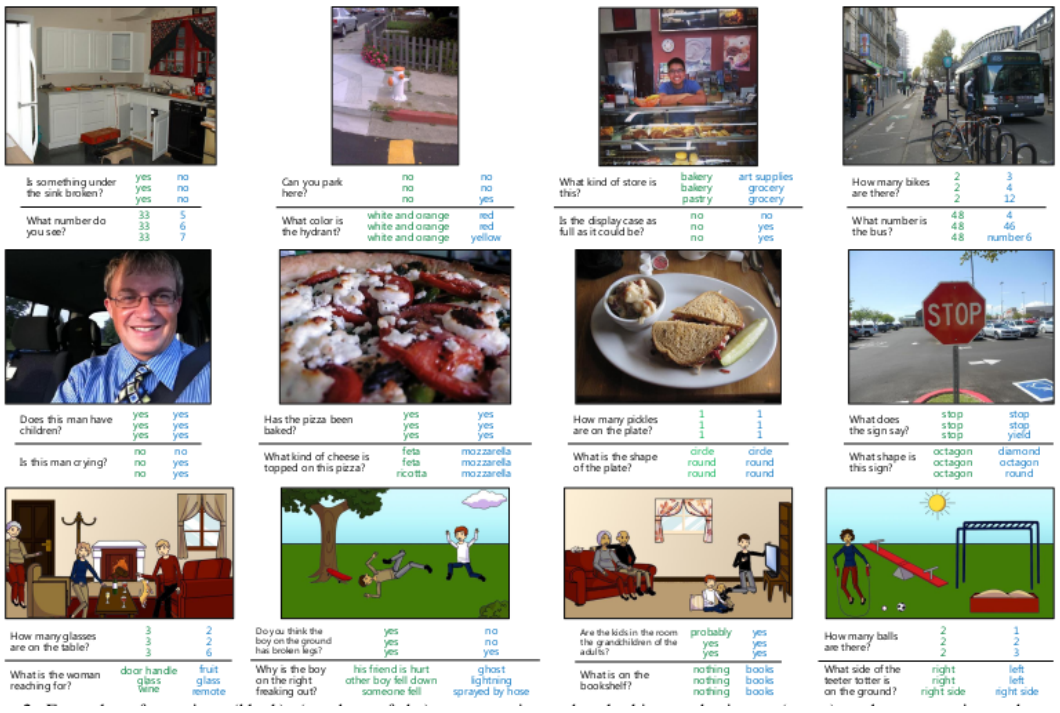


Figure 5: 'Examples of questions (black), (a subset of the) answers given when looking at the image (green), and answers given when not looking at the image (blue) for numerous representative examples of the dataset' [5].

questions. Finally, the dataset comprises corresponding answers for each of the questions. First, for open-ended questions, Agrawal et al. [5] asked ten unique human subjects to answer the question with a short phrase (i.e., not a full sentence). The authors deemed an answer to be correct if at least three workers provided it. More than one answer can be included in the dataset if multiple answers apply to a given question. Regarding *multiple-choice* questions, the authors gathered 18 candidate answers from human subjects. Again, the authors considered an answer to be correct if at least three subjects provided it. For each multiple-choice question, four types of answers were used to generate a candidate set. First, the most common answer was considered the *correct* answer. Second, *plausible* answers were included by collecting answers from three subjects without showing them the image. While these answers may be incorrect, the authors consider them plausible since they are related to the question. Third, Agrawal et al. [5] included the ten most popular answers across the entire dataset: *yes*, *no*, *2*, *1*, *white*, *3*, *red*, *blue*, *4*, and *green*. Finally, the authors filled up the union of *correct*, *plausible*, and *random* answers with correct answers from random questions in the dataset to create a set of 18 candidate answers. Overall, the VQAv1 dataset [5] covers 7,984,119 answers. With 89.32%, single-word answers represent the majority across the dataset, followed by two-word (6.91%)

and three-word (2.74%) answers.

For reasons of feasibility, the present experiment focused on color and shape-based questions only. To this end, I applied a string-based filter to keep only questions that start with the substring *What color* or *What shape*, respectively. As a result, the dataset used in this study covers 23,469 question-image pairs with 414 unique answers. Figure 5 depicts a number of examples from our subset of the VQAv1 dataset [5].

4.3.2. VQA Model

The VQA model utilized in the experiment at hand is a pre-trained MUTAN model [6]. Given a question $Q \in \mathcal{Q}$, an image $I \in \mathcal{I}$ and the set of MUTAN's parameters θ, the VQA model is trained to predict an answer $\hat{A} \in \mathcal{A}$ that matches the ground-truth answer $A^* \in \mathcal{A}$:

$$\hat{A} = \arg\max_{A \in \mathcal{A}} p_\theta(A|I, Q)$$

For image encoding, a CNN architecture embeds I into a vector representation $\mathbf{v} \in \mathbb{R}^{2048}$ [6]. Similarly, the VQA model creates a vector embedding $\mathbf{q} \in \mathbb{R}^{2400}$ based on an encoder-decoder network. MUTAN then performs a bilinear operation $\mathcal{T}$ to fuse the representations of v and q into a single vector $\mathbf{y}$ [6]. Finally, applying a softmax function to $\mathbf{y}$ yields the final answer $\hat{A}$.

Question: What color is the woman wearing

Predicted answer: red.

Question: What shape are the tires on the truck?

Predicted answer: round.

Figure 6: Example predictions of the pre-trained MUTAN on images from the VQAv1 validation set [5].

In an experiment, Ben-Younes et al. [6] trained MUTAN using the VQAv1 dataset (i.e., the same dataset used in this thesis' experimental setup). According to the authors, MUTAN achieved an overall accuracy of approximately 67 percent on the VQAv1 test set. The model performed particularly well regarding questions with binary Yes/No answers (85.14%). For questions including numbers (e.g., "How many ...?"), it achieved 39.81% accuracy. For all other question types (including color and shape-based questions), it achieved an accuracy of 58.52%. Figure 6 depicts an example output of MUTAN for a color and a shape-based question, respectively.

4.3.3. Evaluation

Automatically and objectively assessing the quality of synthetically generated images is a challenging task [91, 51, 67]. Salimans et al. [67] suggest that there exists no objective function to assess a GAN's performance. To evaluate the proposed method, I applied both automatic and manually conducted assessments. Specifically, I evaluated the counterfactual generator under five criteria:

1. **Suitability of the attention mechanism:** I qualitatively assessed the outputs of the applied attention mechanism by evaluating its ability to focus the question-critical objects in an image.

2. **Change of Semantics:** I quantitatively assessed the share to which the VQA model outputs two different answers to the same question when making inference on the original image and the counterfactual image, respectively. Furthermore, I manually assessed whether the generator produced meaningful counterfactuals with changed semantics.

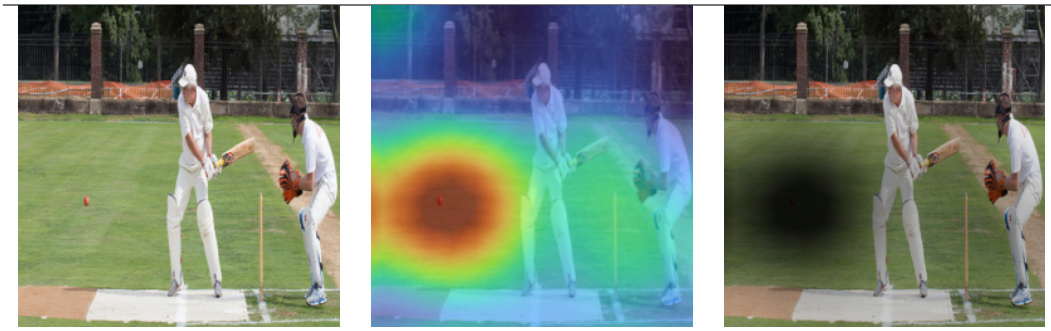
3. **Sensitivity to language-conditioning:** In line with Pan et al. [58], I qualitatively assessed the counterfactual generator's sensitivity to language-conditioning. To this end, I evaluated whether the generator produced different image edits for different input question-answer pairs.

4. **Realism:** In a qualitative evaluation, I assessed whether or not the generated images are realistic.

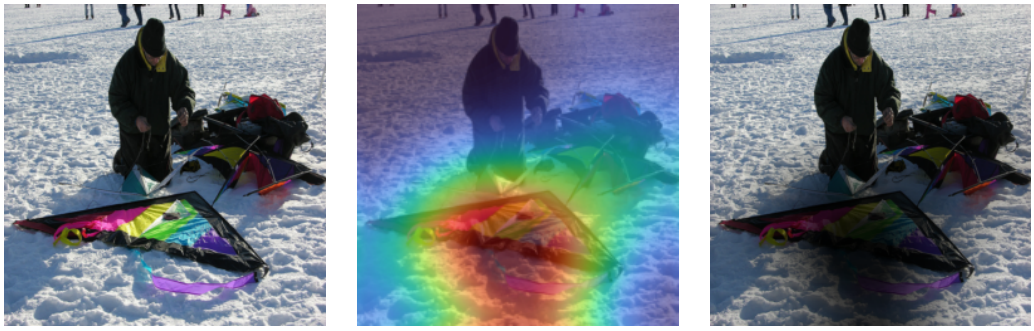
5. **Minimality of image edits:** I computed the L1 norm to measure the number and magnitude of perturbations present in the counterfactual images relative to the original images [52].

5. Results

5.1. Attention Mechanism

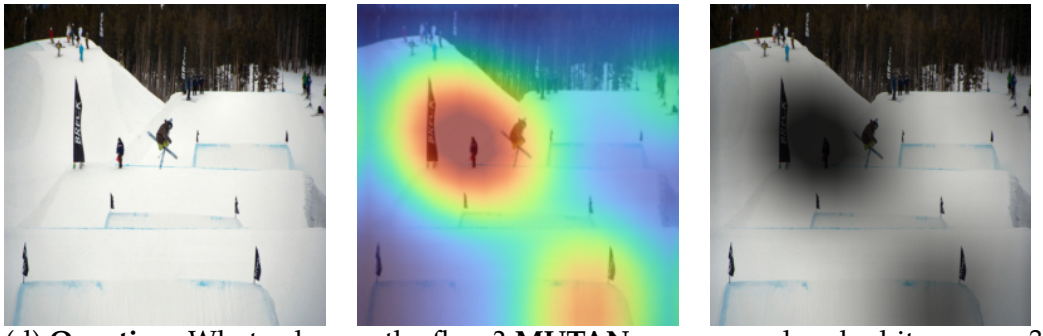


(a) **Question:** What color is the ball? **MUTAN answer:** orange.



(b) **Question:** What colors are the kite? **MUTAN answer:** green and red.

(c) **Question:** What color is the bus? **MUTAN answer:** red



(d) **Question:** What color are the flags? **MUTAN answer:** red and white.

Figure 7: Example outputs of the Grad-CAM algorithm applied to MUTAN for color-based questions. Left: original image. Center: Interpolated attention map projected on the original image. Right: The *background* image.

The first part of the proposed method consists in applying the Grad-CAM algorithm to the VQA model. Since the counterfactual generator is allowed to make changes predominantly in those image regions on which the attention mechanism focuses, its performance heavily relies on the attention maps. This section summarizes the results of a qualitative evaluation of the attention mechanism's ability to focus question-critical objects in images.

Figure 7 depicts, for four example images, the outputs of the Grad-CAM algorithm applied to MUTAN as well as the latter's answer to the question. On the left is the original image I that serves as input to the VQA model. The center image shows MUTAN's attention x_a in the form of a heatmap. Finally, the rightmost image depicts the *background* obtained by computing $(\mathbf{1} - a) \odot I$, where $\mathbf{1}$ is the all-ones vector and $\odot$ denotes the element-wise multiplication. The generator $\mathcal{G}$ is allowed to make more significant modifications to pixels with large intensity values in x_a than to those with low values. For image (a) and the question *What color is the ball?*, the projection indicates that MUTAN mostly focuses on the ball and a relatively small area around it. Consequently, the ball disappears almost entirely when obtaining the corresponding background image. Furthermore, the lower part of the field, as well as the players' lower bodies, appear slightly less bright than in the original image. Similarly, in image (b), given the question *What colors are the kite?*, the projection focuses on the kite's inner portion. Furthermore, a segment of the ground in the image's lower part is highlighted as well. As a result, the kite's colored area and the center part of the image's bottom are almost entirely hidden in the background image. The projection for the question *What color is the bus?* in image (c) indicates that the VQA model mostly focuses on a relatively small section at the bus' bottom and top, while the attention is weaker regarding its rear. Finally, for image (d), MUTAN outputs an incorrect answer: While the flags are actually black, the VQA model outputs *red and white* for an answer. The projections indicate how the model might have come up with its prediction: It focuses on a person holding a red object in the upper left area and a small area of the snow at the image's bottom. These observations are reflected in the background image as well.

(a) **Question:** What shape are the tires on the truck? **MUTAN's answer:** round.

(b) **Question:** What shape is the white pillow? **MUTAN's answer:** oval.

Figure 8: Example outputs of the Grad-CAM algorithm applied to MUTAN for shape-based questions. Left: original image. Center: Interpolated attention map projected on the original image. Right: The *background* image.

Figure 8 shows two example outputs of Grad-CAM for shape-based questions. The order of the images is the same as in Figure 7. The projection in image (a) indicates that given the question *What shape are the tires on the truck?*, the VQA model focuses on the vehicle's entire lower part rather than merely the tires. Furthermore, its attention is greater at the truck's rear than its front. In the *background* image, the back wheel is almost entirely missing, while the front wheel is visible. Furthermore, a relatively large portion of pixels corresponding with the truck's load area and the car on top of it have very small intensity values. Similarly, in image (b), the projection also focuses on an area of the image that is larger than the question-critical object. Instead of mainly focusing on the pillow, the attention is mostly on the bed. This becomes even more apparent when considering the background image: While most of the bed is nearly unrecognizable, the pillow is still visible. However, it is blurred to some extent.

These examples discussed in this section are vicarious for four main patterns regarding the attention mechanism:

- If the object relevant to answering a question is relatively small compared to the rest of the image, the attention mechanism focuses on it completely in most cases. In other words, the computed intensities are higher for pixels belonging to the object than for the rest of the pixels. Under these circumstances, the generator can make larger changes to the entire object than to the rest of the image.
- Contrarily, if the object is very large or MUTAN pays attention to the background, the projection usually focuses only on a part of it. Consequently, the information the generator receives allows it to apply more significant changes to a segment of the object or the background than to the rest of it.
- If MUTAN makes an incorrect prediction, this is often reflected by the projection not focusing on the question-critical object, but another element of the image, such as in example (d) of Figure 7.
- For most shape-based questions, the projections focus on larger image regions compared to those for color-based questions.

(a) **Question:** What color are the peppers in the bottom left corner?

MUTAN's answer: yellow **MUTAN's answer:** red

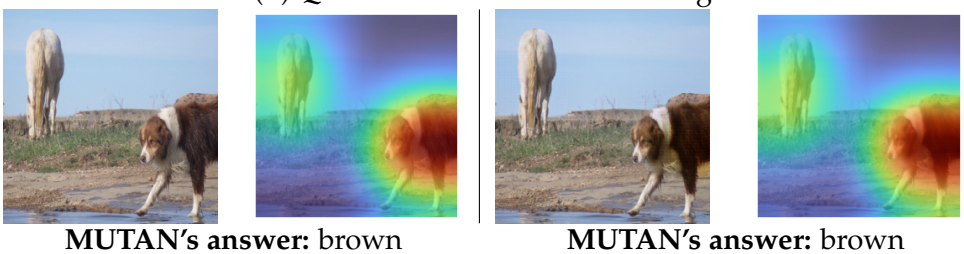
(b) **Question:** What color is the large central flower?

MUTAN's answer: yellow **MUTAN's answer:** orange

(c) **Question:** What color is the bird?

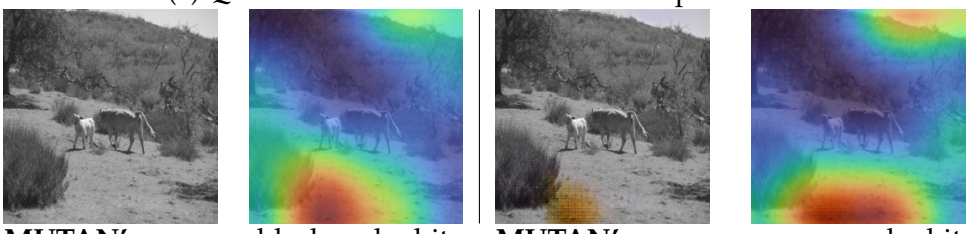
MUTAN's answer: blue and white **MUTAN's answer:** brown and white

(d) **Question:** What color is the dog?



MUTAN's answer: brown **MUTAN's answer:** brown

(e) **Question:** What color scheme is the picture in?



MUTAN's answer: black and white **MUTAN's answer:** orange and white

Figure 9: Example outputs of the counterfactual generator for color-based questions from the VQAv1 [5] validation set. Left: original image and corresponding attention map. Right: Generated counterfactual image and corresponding attention map.

5.2. Evaluation of the Counterfactuals

Change of Semantics

For the VQAv1 [5] validation set, the VQA model's output changes for 37.78% of the counterfactuals. In particular, the VQA model outputs a different answer for the counterfactual than for the original image for 38.05% of the color-based questions and 25.45% of the shape-based questions. For the rest of the generated counterfactuals, the VQA model's prediction mostly does not change for either of the following two reasons. First, in some cases, the difference between the original and counterfactual images is too small for the model or a human observer to notice. Second, suppose the question-critical object accounts for a large portion of the image, or the question is about the image's background. In that case, the generator often only edits those parts on which the attention mechanism focuses. As a result, the relevant image region is not modified in its entirety so that the VQA model or a human observer cannot perceive a semantically meaningful change.

Figure 9 depicts qualitative results of the counterfactual generator for color-based questions that are exemplary for these findings. The figure shows the original image and the heatmap denoting the VQA model's attention during inference on the left. On the right, the counterfactual image and the corresponding heatmap are displayed. Rows (a) - (c) show example counterfactuals that successfully change the original image's semantics and cause the VQA system to make a different prediction. In example (a), the color of the peppers in the bottom left corner changes from yellow to red, which the VQA system successfully observes. As the heatmaps illustrate, while it correctly captures the semantic change and still focuses the question-critical objects, the VQA model seems to slightly shift its attention to an additional region in the upper left corner when making an inference on the counterfactual image. Contrarily, it solely focuses on the lower-left corner in the original image. In example (b), the color of the large flower in the image's center changes from yellow to orange. Similarly, the bird's wing color successfully changes from blue to brown in example (c). These counterfactuals show the generator to be successful in changing the semantics of an image such that they are perceived both by the VQA model and a human observer. However, for several images, the model fails to alter their semantic meaning. Row (d) in Figure 9 depicts such an example: the VQA model's answer does not change when being provided with the counterfactual. While the dog appears slightly darker in the counterfactual than in the original image, the color-change is marginal and not perceived as such by the VQA model.

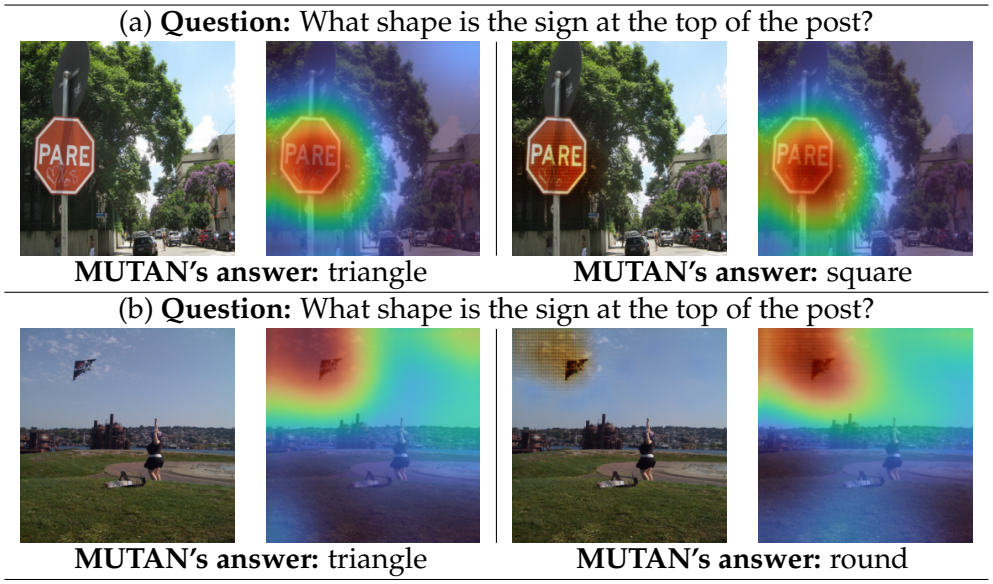


Figure 10: Example outputs of the counterfactual generator for shape-based questions from the VQAv1 [5] validation set. Left: Original image and corresponding attention map. Right: Generated counterfactual image and corresponding attention map.

While the model can generate semantically meaningful counterfactuals for many color-based questions, it fails mostly fails in the context of shapes. Figure 10 depicts two examples. While the VQA model predicts a different answer for the counterfactual than for the original image in both cases, the changes are not semantically meaningful from a human observer's perspective. In example (a), the object's shape remains roughly unchanged, while the counterfactual generator slightly edits the sign's color. Furthermore, the VQA model predicts an incorrect answer regarding both the original and the counterfactual image. As the heatmap indicates, the modifications to the original image appear to be significant enough for the VQA model to shift its focus slightly to the lower right portion of the sign when making an inference on the counterfactual. This shift seems to cause the model to change its prediction. In contrast to example (a), the changes to the image in example (b) are more dominant. The counterfactual generator produces an artifact covering the kite and a segment of the sky surrounding it. As the heatmaps show, the VQA model focuses on the same area in both the original and the counterfactual image, but the artifact seems to cause it to change its answer. These two instances are exemplary for most of the counterfactual for shape-based questions: the generator (i) applies only a few edits that are barely noticeable or (ii) produces artifacts that are not semantically meaningful for a human observer.

Sensitivity to language-conditioning

Both the examples in Figure 9 and Figure 10 show that the counterfactual generator's edits vary depending on the questions and answers. Since the attention maps pose a strong constraint for the generator, its edits heavily depend on them. If the attention map focuses on the question-critical object, such as the bird in row (c) of Figure 9, the generator successfully modifies it. In this case, the heatmap predominantly focuses on the bird's body, while it mostly neglects the rest of the image. Contrarily, if the attention map focuses only on a small portion of the object in question (such as in example (e) of Figure 9), language conditioning does not have the desired effect. In these cases, the generator fails to modify the areas relevant to the question-answer pair suffieciently.

Realism

The model successfully generates visually realistic counterfactual images in several cases. For instance, in the counterfactuals in rows (a) - (c) of Figure 9, the changes to the question-critical objects appear realistic to a human observer. Furthermore, since the generator mainly retains the backgrounds unchanged and does not seem to produce obvious artifacts, the modified objects are still located in their original environments. This makes the counterfactual images appear visually realistic to a human observer. In example (b), however, the attention map focuses on a small section of the background in addition to the question-critical object (i.e., the flower). As a result, the counterfactual generator also slightly alters the background by adding an orange checkerboard pattern in the corresponding area. This addition also seems to affect the VQA model, which pays less attention to the bottom left corner in the counterfactual than in the original image. Furthermore, although they do not cause the example counterfactuals to be visually unrealistic, in some cases, the altered regions contain gentle artifacts. This trend can, for instance, be observed in example (a) of Figure 9: the peppers are covered by a subtle checkerboard pattern.

In other cases, the generator produces noticeable artifacts that cause the counterfactual images to be visually unrealistic. For example, in row (e) of Figure 9, a relatively small yellowish-brownish pattern seems to suffice for the VQA model to change its prediction relative to the original image. This modification results in a counterfactual image that looks *fake* to a human observer. Similarly, the counterfactual in example (b) of Figure 10 contains a large checkerboard pattern that makes it appear unrealistic to a human observer.

Minimality of Image edits

The ℓ_1-norm was computed across both the training and the validation set to evaluate the counterfactual generator's performance regarding the minimality-of-edits constraint. It measures the magnitude of changes in the counterfactual relative to the original image. Lower values indicate fewer changes than greater values.

		All	Colors	Shapes	Same VQA Answers			Different VQA Answers		
					All	Colors	Shapes	All	Colors	Shapes
Training set	μ	0.0175	**0.0174**	0.0207	0.0177	**0.0176**	0.0212	**0.0172**	**0.0172**	0.0195
	σ	**0.0039**	**0.0039**	0.0048	0.0040	**0.0039**	0.0049	0.0038	**0.0038**	0.0046
Validation set	μ	0.0175	**0.0174**	0.0208	0.0177	**0.0176**	0.0212	**0.0173**	**0.0173**	0.0198
	σ	0.0040	**0.004**	0.0047	**0.0041**	**0.0041**	0.0048	**0.0038**	**0.0038**	0.0042

Table 1: Mean (μ) and standard deviation (σ) of the ℓ_1-norm computed across the training and validation set and split across categories.

Table 1 summarizes the results for this experiment. The mean (denoted μ) and standard deviation (denoted σ) values are calculated for different splits of both the training and the validation set. The first three columns represent the values computed across the entire datasets and for color-based and shape-based questions only. The remaining six columns contain the same computations for the portion of pairs of original and counterfactual images for which the VQA model predicted distinct or equal answers, respectively. The results indicate that the counterfactual generator applied fewer changes regarding color-based questions than concerning shapes. This observation applies to both the training and the validation set and across all splits. Moreover, overall, the generator applied fewer changes in those cases where the VQA model's predictions regarding the original and counterfactual image were distinct than if they were equal. Similar to the mean values, the standard deviations are lower for color-based questions than for shape-based questions. Moreover, the values are slightly larger for pairs of images and counterfactuals for which the VQA model outputs the same answer to the corresponding question than for those pairs for which the answers are different.

6. Discussion

The experiments conducted in this thesis show the proposed method to be partly effective for editing question-critical objects to generate counterfactual images for VQA. Specifically, it succeeds in applying semantically meaningful, language-conditioned, and visually realistic edits concerning several color-based questions, while it fails to do so for shape-based questions. Furthermore, the counterfactual generator performs well when the attention mechanism focuses on the question-critical object as a whole rather than only a part of it. This finding indicates that using an attention mechanism suits the desired trait of training the generator to alter question-critical objects while retaining the rest of the image. Using a weighted reconstruction loss also promotes the desired trait of training the model to apply more edits to question-critical image regions than to the rest of the image. Therefore, for a human observer, the *background* of the counterfactual image looks identical to the original image's *background* in many of the qualitative results. However, in some cases, the attention map focuses only partly on the question-critical object. This problem mainly occurs when the question-answer pair is about the image's background or an object covering a large area. In these cases, it seems that the VQA model takes into account only a part of the object to predict an answer to a given question. The reason for the generator to fail regarding these instances might be two-fold: First, the reconstruction loss appears to be too rigid for those parts of the object to which the attention map pays only weak attention, forcing the generator to reproduce the original pixels predominantly. Second, since the generated image is merged with the original image's *background*, the generator cannot apply significant edits to the pixels with large intensity values in the *background* without causing an unrealistic counterfactual. This could be solved by relaxing the reconstruction loss across the entire image. With the current approach, the continuous attention map can contain zeros or values close to zero. As a result, the generator cannot (significantly) alter the corresponding pixels in the image. By increasing all values in the attention map that are below a certain threshold (e.g., below 0.2), the generator would be allowed to alter all pixels at least to some extent. However, a drawback of this solution could be that image regions not related to the question-answer pair might not be fully retained. Using an attention mechanism that more reliably focuses the entire question-critical object could solve this problem.

In some instances, the counterfactual generator modifies the question-critical objects in a way that the VQA model perceives a semantically meaningful change, while a human observer does not notice it. Enforcing the constraint of negating the VQA model's answer might solve this problem. This could be achieved by applying a higher relative weighting to the negated cross-entropy loss or by using it for training both the generator and the discriminator, rather than only the former. In this case, the discriminator would not only penalize the generator for unrealistic counterfactuals but also when they do not lead to an alternative answer prediction of the VQA model.

While the method introduced in this thesis achieves encouraging results for color-based questions, it failed to modify objects regarding their shapes. Along the lines of Pan et al. [58], whose counterfactual generator also fails to make semantic edits that go beyond colors, this might be explained by the following two reasons. First, the reconstruction loss might still be too rigid despite the attention mechanism allowing for more modifications to question-critical regions than to the rest of the image. The results of the evaluation regarding the minimality of edits support this conclusion. Specifically, the generator seems to require more changes to successfully alter an object's shape than regarding its colors. The reconstruction loss seems to force the generator not to go beyond a certain amount of modifications. As a result, it cannot alter the objects sufficiently so that their shape does not change, or at least not enough. Furthermore, the attention mechanism often does not focus on the entire object. In these cases, the reconstruction loss is too rigid regarding an object's edges, which are crucial to changing its shape. This could be solved by applying an attention mechanism that more accurately focuses the entire question-critical object. Second, the training data contains significantly less shape-based than color-based questions. Hence, the dataset might not be diverse enough for the generator to learn how to alter shapes sufficiently. This problem could be solved by using a more balanced training dataset.

7. Conclusion and Future Work

This thesis aimed at providing discriminative explanations for a VQA system's predictions to increase the interpretability of its decision-making processes. To this end, it introduced *CountEx-VQA*, a GAN that edits question-critical objects in images to generate counterfactuals. Specifically, the implementation consists of a modified LingUNet [43] as the counterfactual generator and a PatchGAN discriminator [28] that penalizes unrealistic generated images. Given a VQA model $f : (I, Q) \rightarrow \hat{A}$ that predicts an answer $\hat{A}$ based on an image-question tuple (I, Q), the GAN is trained to generate an image I' that is minimally different from I such that the VQA model's answer changes (i.e., $f : (I', Q) \rightarrow \hat{A}' \neq \hat{A}$). By incorporating a Grad-CAM-based attention mechanism that determines each pixel's importance regarding the VQA model's decision making process, the counterfactual generator learns to apply modifications in an image predominantly to question-critical objects, while retaining the rest of the image.

Extensive experiments on the challenging VQAv1 dataset [5] have shown the proposed method to achieve encouraging results for color questions. In many cases, the counterfactual generator successfully changed an image's semantics by modifying question-critical objects. Furthermore, the results indicate that using an attention mechanism is an appropriate means to guide the modification process. The quality of the counterfactual images depended to a large extent on the attention maps. The generator edited the appropriate locations in those cases where it successfully focused on the question-critical object(s). The rest of the image, on the other hand, remained intact and did not contain any modifications detectable by a human observer. Contrarily, if the map did not focus on the entire object but only a part of it, the generator had difficulties modifying the image. Either it did not modify the image sufficiently for the VQA model to perceive a change in its semantic meaning, or it inserted visually unrealistic artifacts.

Introducing a reconstruction loss weighted on the attention map sought to allow the counterfactual generator to make more modifications in areas focused by the attention map than in the rest of the image. This was (i) to allow for more extensive semantic modifications beyond color changes and (ii) to ensure that question-irrelevant areas of the image remained nearly unchanged. While the weighted reconstruction loss seems to facilitate the latter, the results for shape-based questions show that the generator lacks the potential for the former. Similar to the approach developed by Pan et al. [58], *CountEx-VQA* failed to generate realistic counterfactuals that contain semantically meaningful differences compared to the original image in these cases. The following reasons could explain this problem. First, the reconstruction loss might overall be too strict to allow for major semantic interventions. Second, the training dataset used in the experiment contains significantly fewer shape-based questions than color-based questions. Therefore, it could be that the generator did not receive enough information to learn appropriate patterns regarding the shape of objects.

For future work, the model could be trained on a larger, more diverse dataset. For instance, Goyal et al. [23] proposed the VQAv2 dataset, which contains multiple images per question rather than only a single one as in the VQAv1 dataset [5]. Moreover, using an attention mechanism that focuses on the question-critical objects more accurately could also significantly improve the model's performance. For example, Li et al. [34] propose an approach that employs super-pixel segmentation to extract *concepts* and uses the Shapley Value [70] algorithm to determine each concept's contribution to a DNN's decision. In images, a concept can be a color, texture, or a group of similar segments. Training the generator to alter the most important concept(s) in an image rather than providing it an attention map determining each pixel's intensity could alleviate the limitations encountered regarding the method proposed in this research. Moreover, replacing an entire instance of a concept rather than editing an image on a pixel-by-pixel level could pave the way for semantic changes even larger than altering shapes.

Acknowledgements

I want to thank my first supervisor, Prof. Dr. Matthias Thimm, who entrusted me with the subject of this thesis and provided essential feedback throughout the process. Special thanks go to my second supervisor, Dr. Zeyd Boukhers, who consistently supported me in conducting this thesis, provided crucial input and suggestions during our weekly meetings and was always available to discuss progress and issues along the way. I would also like to thank Prof. Kimiaki Shirahama from the Department of Informatics / Cyber Informatics Research Institute at Kindai University for the discussions, his expertise and feedback, and his suggestions regarding my Master's thesis.

List of Abbreviations

VQA Visual Question Answering

LIME Local Interpretable Model-agnostic Explanations

DNN Deep Neural Network

NN Neural Network

NLP Natural Language Processing

DL Deep Learning

CNN Convolutional Neural Network

LSTM Long Short-Term Memory

CSS Counterfactual Samples Synthesizing

ML Machine Learning

GAN Generative Adversarial Network

MS COCO Microsoft Common Objects in Context

Grad-CAM Gradient-weighted Class Activation Mapping

ReLU Rectified Linear Units

Tanh Hyperbolic Tangent Function

LTU Linear Threshold Unit

MLP Multi-Layer Perceptron

GD Gradient Descent

List of Figures

1. Overview of the general architecture of *CountEx-VQA*'s architecture inspired by Pan et al. [58]. . . . 24
2. Example question-image pair from the VQAv1 dataset [5]. The red bounding box indicates the question-critical object. . . . 27
3. Architecture of the generator . . . 30
4. Architecture of the discriminator . . . 32
5. 'Examples of questions (black), (a subset of the) answers given when looking at the image (green), and answers given when not looking at the image (blue) for numerous representative examples of the dataset' [5]. . . . 34
6. Example predictions of the pre-trained MUTAN on images from the VQAv1 validation set [5]. . . . 36
7. Example outputs of the Grad-CAM algorithm applied to MUTAN for color-based questions. Left: original image. Center: Interpolated attention map projected on the original image. Right: The *background* image. . . . 39
8. Example outputs of the Grad-CAM algorithm applied to MUTAN for shape-based questions. Left: original image. Center: Interpolated attention map projected on the original image. Right: The *background* image. . . . 41
9. Example outputs of the counterfactual generator for color-based questions from the VQAv1 [5] validation set. Left: original image and corresponding attention map. Right: Generated counterfactual image and corresponding attention map. . . . 43
10. Example outputs of the counterfactual generator for shape-based questions from the VQAv1 [5] validation set. Left: Original image and corresponding attention map. Right: Generated counterfactual image and corresponding attention map. . . . 45
11. Additional qualitative results (Part I) of *CountEx-VQA* for question-image pairs from the VQAv1 [5] validation set based on a pre-trained MUTAN VQA model [6]. . . . 59
12. Additional qualitative results (Part II) of *CountEx-VQA* for question-image pairs from the VQAv1 [5] validation set based on a pre-trained MUTAN VQA model [6]. . . . 60

List of Tables

1. Mean (μ) and standard deviation (σ) of the ℓ_1-norm computed across the training and validation set and split across categories. 47

A. Appendix

Figure 11: Additional qualitative results (Part I) of *CountEx-VQA* for question-image pairs from the VQAv1 [5] validation set based on a pre-trained MUTAN VQA model [6].

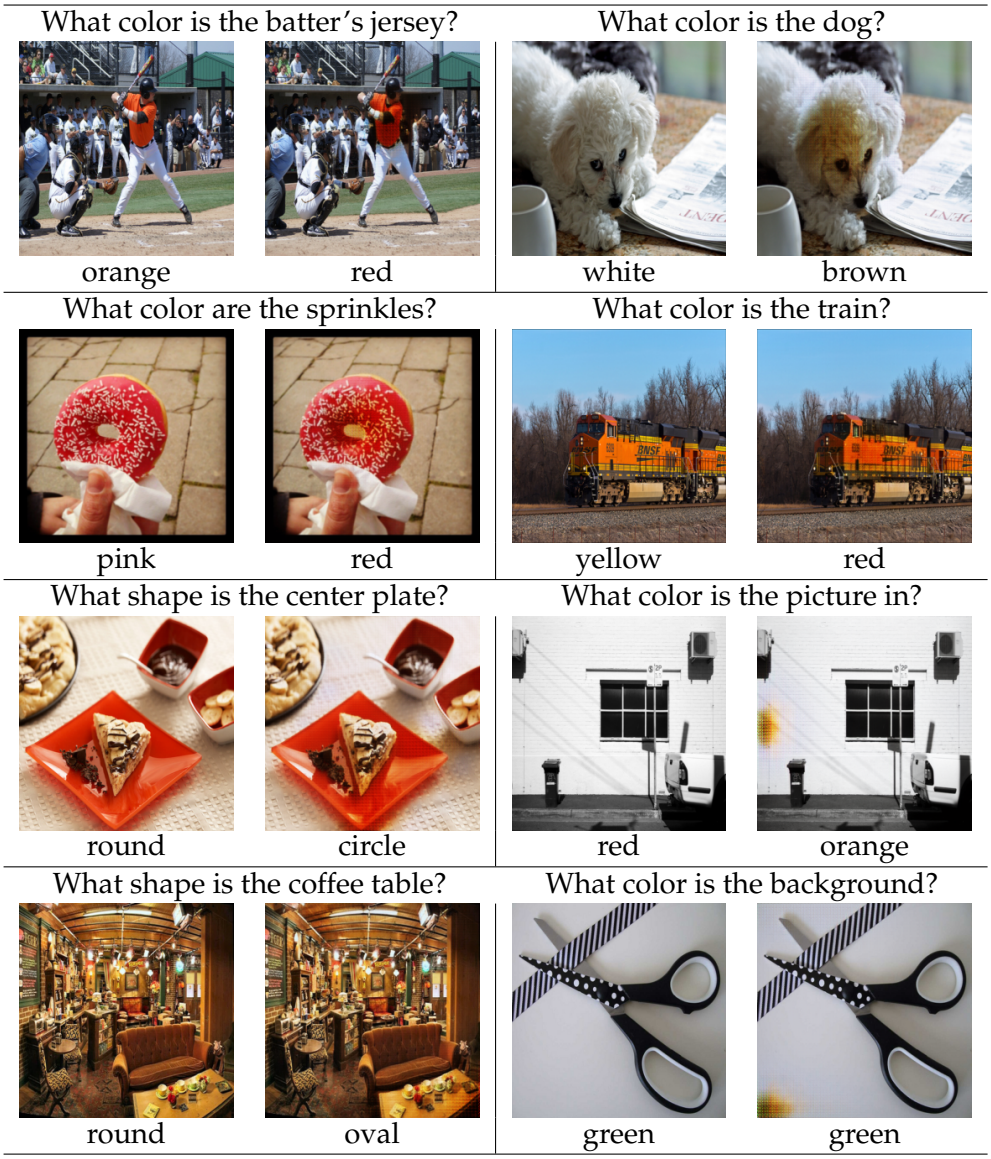


Figure 12: Additional qualitative results (Part II) of *CountEx-VQA* for question-image pairs from the VQAv1 [5] validation set based on a pre-trained MUTAN VQA model [6].

References

[1] Agnar Aamodt and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7:39–59, August 2001.

[2] Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. Interpretable Machine Learning in Healthcare. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, pages 559–560, New York, NY, USA, August 2018. Association for Computing Machinery.

[3] Youssef Alami Mejjati, Christian Richardt, James Tompkin, Darren Cosker, and Kwang In Kim. Unsupervised Attention-guided Image-to-Image Translation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *arXiv:1707.07998 [cs]*, March 2018. arXiv: 1707.07998.

[5] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. *International Journal of Computer Vision*, 123:4–31, 2017.

[6] Hedi Ben-younes, Remi Cadene, Matthieu Cord, and Nicolas Thome. MUTAN: Multimodal Tucker Fusion for Visual Question Answering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2631–2639, 2017.

[7] Fredrik Bergholm. Edge Focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):726–741, 1987.

[8] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[9] J. Brownlee. *Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image Translation*. Machine Learning Mastery, 2019.

[10] S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao, T. D. Kelley, D. Braines, M. Sensoy, C. J. Willis, and P. Gurram. Interpretability of deep learning models: A survey of results. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–6, August 2017.

[11] Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. Counterfactual Samples Synthesizing for Robust Visual Question Answering. *arXiv:2003.06576 [cs]*, March 2020. arXiv: 2003.06576.

[12] K. R. Chowdhary. Natural Language Processing. In *Fundamentals of Artificial Intelligence*, pages 603–649. Springer India, New Delhi, 2020.

[13] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, 35(1):53–65, January 2018.

[14] Abhishek Das, Harsh Agrawal, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions? *arXiv:1606.03556 [cs]*, June 2016. arXiv: 1606.03556.

[15] Emily Denton, Ben Hutchinson, Margaret Mitchell, Timnit Gebru, and Andrew Zaldivar. Image Counterfactual Sensitivity Analysis for Detecting Unintended Bias. *arXiv:1906.06439 [cs, stat]*, October 2020. arXiv: 1906.06439.

[16] Finale Doshi-Velez and Been Kim. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv:1702.08608 [cs, stat]*, March 2017. arXiv: 1702.08608.

[17] Carlos Fernández-Loría, Foster Provost, and Xintian Han. Explaining Data-Driven Decisions made by AI Systems: The Counterfactual Approach. *arXiv:2001.07417 [cs, stat]*, June 2021. arXiv: 2001.07417.

[18] Donald Geman, Stuart Geman, Neil Hallonquist, and Laurent Younes. Visual Turing test for computer vision systems. *Proceedings of the National Academy of Sciences*, 112(12):3618–3623, March 2015.

[19] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining Explanations: An Overview of Interpretability of Machine Learning. *arXiv:1806.00069 [cs, stat]*, February 2019. arXiv: 1806.00069.

[20] Oscar Gomez, Steffen Holter, Jun Yuan, and Enrico Bertini. ViCE: visual counterfactual explanations for machine learning models. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, pages 531–535, New York, NY, USA, March 2020. Association for Computing Machinery.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[22] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.

[23] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. *arXiv:1612.00837 [cs]*, May 2017. arXiv: 1612.00837.

[24] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual Visual Explanations. *arXiv:1904.07451 [cs, stat]*, June 2019. arXiv: 1904.07451.

[25] Akshay Kumar Gupta. Survey of Visual Question Answering: Datasets and Techniques. *arXiv:1705.03865 [cs]*, May 2017. arXiv: 1705.03865.

[26] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc., 1 edition, 2017.

[27] Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Generating Counterfactual Explanations with Natural Language. *arXiv:1806.09809 [cs]*, June 2018. arXiv: 1806.09809.

[28] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv:1611.07004 [cs]*, November 2018. arXiv: 1611.07004.

[29] Been Kim, Rajiv Khanna, and Oluwasanmi O. Koyejo. Examples are not enough, learn to criticize! Criticism for Interpretability. *Advances in Neural Information Processing Systems*, 29:2280–2288, 2016.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. arXiv: 1412.6980.

[31] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *International Journal of Computer Vision*, 123(1):32–73, May 2017.

[32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[33] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object Recognition with Gradient-Based Learning. In *Shape, Contour and Grouping in Computer Vision*, Lecture Notes in Computer Science, pages 319–345. Springer, Berlin, Heidelberg, 1999.

[34] Jiahui Li, Kun Kuang, Lin Li, Long Chen, Songyang Zhang, Jian Shao, and Jun Xiao. Instance-wise or Class-wise? A Tale of Neighbor Shapley for Concept-based Explanation. In *ACM International Conference on Multimedia (MM)*, 2021.

[35] Qing Li, Jianlong Fu, Dongfei Yu, Tao Mei, and Jiebo Luo. Tell-and-Answer: Towards Explainable Visual Question Answering using Attributes and Captions. *arXiv:1801.09041 [cs]*, January 2018. arXiv: 1801.09041.

[36] Elizabeth Liddy. Natural Language Processing. *School of Information Studies - Faculty Scholarship*, January 2001.

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[38] Zinan Lin, Vyas Sekar, and Giulia Fanti. Why Spectral Normalization Stabilizes GANs: Analysis and Improvements. *arXiv:2009.02773 [cs, stat]*, April 2021. arXiv: 2009.02773.

[39] Zachary C. Lipton. The Mythos of Model Interpretability. *arXiv:1606.03490 [cs, stat]*, March 2017. arXiv: 1606.03490.

[40] Han Liu, Alexander Gegov, and Mihaela Cocea. Rule-based systems: a granular computing perspective. *Granular Computing*, 1(4):259–274, December 2016.

[41] Han Liu, Alexander Gegov, and Mihaela Cocea. Rule Based Networks: An Efficient and Interpretable Representation of Computational Models. *Journal of Artificial Intelligence and Soft Computing Research*, 7(2):111–123, April 2017. Publisher: Sciendo Section: Journal of Artificial Intelligence and Soft Computing Research.

[42] Mateusz Malinowski and Mario Fritz. A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input. *arXiv:1410.0210 [cs]*, May 2015. arXiv: 1410.0210.

[43] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[44] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction. *arXiv:1809.00786 [cs]*, March 2019. arXiv: 1809.00786.

[45] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *arXiv:1802.05957 [cs, stat]*, February 2018. arXiv: 1802.05957.

[46] Christoph Molnar. Interpretable Machine Learning, 2020.

[47] Grégoire Montavon, Sebastian Bach, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining NonLinear Classification Decisions with Deep Taylor Decomposition. *Pattern Recognition*, 65:211–222, May 2017. arXiv: 1512.02479.

[48] Raha Moraffah, Mansooreh Karami, Ruocheng Guo, Adrienne Raglin, and Huan Liu. Causal Interpretability for Machine Learning - Problems, Methods and Evaluation. *ACM SIGKDD Explorations Newsletter*, 22(1):18–33, May 2020.

[49] Edoardo Mosca. Explainability of Hate Speech Detection Models. Master's thesis, Technische Universität München - Department of Mathematics, 2020.

[50] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, October 2019. arXiv: 1901.04592.

[51] Seonghyeon Nam, Yunji Kim, and Seon Joo Kim. Text-Adaptive Generative Adversarial Networks: Manipulating Images with Natural Language. *Advances in Neural Information Processing Systems*, 31:42–51, 2018.

[52] Daniel Nemirovsky, Nicolas Thiebaut, Ye Xu, and Abhishek Gupta. CounteRGAN: Generating Realistic Counterfactuals with Residual Generative Adversarial Nets. *arXiv:2009.05199 [cs, stat]*, September 2020. arXiv: 2009.05199.

[53] Yulei Niu, Kaihua Tang, Hanwang Zhang, Zhiwu Lu, Xian-Sheng Hua, and Ji-Rong Wen. Counterfactual VQA: A Cause-Effect Look at Language Bias. *arXiv:2006.04315 [cs]*, June 2020. arXiv: 2006.04315.

[54] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *CoRR*, abs/1811.03378, 2018.

[55] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature Visualization. *Distill*, 2(11):e7, November 2017.

[56] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks. *CoRR*, abs/1511.08458, 2015.

[57] S.K. Pal and S. Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks*, 3(5):683–697, September 1992. Conference Name: IEEE Transactions on Neural Networks.

[58] Jingjing Pan, Yash Goyal, and Stefan Lee. Question-Conditioned Counterfactual Image Generation for VQA. *arXiv:1911.06352 [cs]*, November 2019. arXiv: 1911.06352.

[59] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent Progress on Generative Adversarial Networks (GANs): A Survey. *IEEE Access*, 7:36322–36333, 2019. Conference Name: IEEE Access.

[60] Supriya Pandhre and Shagun Sodhani. Survey of Recent Advances in Visual Question Answering. *arXiv:1709.08203 [cs]*, September 2017. arXiv: 1709.08203.

[61] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. volume 28(3) of *Proceedings of Machine Learning Research*, pages 1310–1318. PMLR, 2013.

[62] Judea Pearl. Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution. *arXiv:1801.04016 [cs, stat]*, January 2018. arXiv: 1801.04016.

[63] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]*, January 2016. arXiv: 1511.06434.

[64] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, August 2016. Association for Computing Machinery.

[65] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351, pages 234–241. Springer, 2015.

[66] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Internal Representations by Error Propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[67] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. *arXiv:1606.03498 [cs]*, June 2016. arXiv: 1606.03498.

[68] Hannes Schulz and Sven Behnke. Deep Learning. *KI - Künstliche Intelligenz*, 26(4):357–363, November 2012.

[69] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *arXiv:1610.02391 [cs]*, December 2019. arXiv: 1610.02391.

[70] Lloyd S. Shapley. A value for n-person games. In *Contributions to the Theory of Games*. 1953.

[71] Ravid Shwartz-Ziv and Naftali Tishby. Opening the Black Box of Deep Neural Networks via Information. *arXiv:1703.00810 [cs]*, April 2017. arXiv: 1703.00810.

[72] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv:1312.6034 [cs]*, April 2014. arXiv: 1312.6034.

[73] Kacper Sokol and Peter Flach. Glass-Box: Explaining AI Decisions With Counterfactual Statements Through Conversation With a Voice-enabled Virtual Assistant. In *Twenty-Seventh International Joint Conference on Artificial Intelligence Demos*, pages 5868–5870, 2018.

[74] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[75] Yash Srivastava, Vaishnav Murali, Shiv Ram Dubey, and Snehasis Mukherjee. Visual Question Answering using Deep Learning: A Survey and Performance Analysis. *arXiv:1909.01860 [cs]*, December 2020. arXiv: 1909.01860.

[76] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[77] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer-Verlag, London, 2011.

[78] Damien Teney, Ehsan Abbasnedjad, and Anton van den Hengel. Learning What Makes a Difference from Counterfactual Examples and Gradient Supervision. *arXiv:2004.09034 [cs]*, April 2020. arXiv: 2004.09034.

[79] Michael Tsang, Hanpeng Liu, Sanjay Purushotham, Pavankumar Murali, and Yan Liu. Neural Interaction Transparency (NIT): Disentangling Learned Interactions for Improved Interpretability. *Advances in Neural Information Processing Systems*, 31:5804–5813, 2018.

[80] Kewei Tu, Meng Meng, Mun Wai Lee, Tae Eun Choe, and Song-Chun Zhu. Joint Video and Text Parsing for Understanding Events and Answering Queries. *arXiv:1308.6628 [cs]*, February 2014. arXiv: 1308.6628.

[81] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR*, abs/1607.08022, 2016. _eprint: 1607.08022.

[82] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual Explanations for Machine Learning: A Review. *arXiv:2010.10596 [cs, stat]*, October 2020. arXiv: 2010.10596.

[83] Yang Wang. A Mathematical Introduction to Generative Adversarial Nets (GAN). *arXiv:2009.00169 [cs, math, stat]*, August 2020. arXiv: 2009.00169.

[84] Maciej Wiatrak, Stefano V. Albrecht, and Andrew Nystrom. Stabilizing Generative Adversarial Networks: A Survey. *arXiv:1910.00927 [cs]*, March 2020. arXiv: 1910.00927.

[85] Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40, October 2017.

[86] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked Attention Networks for Image Question Answering. *arXiv:1511.02274 [cs]*, January 2016. arXiv: 1511.02274.

[87] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and Yang: Balancing and Answering Binary Visual Questions. *arXiv:1511.05099 [cs]*, April 2016. arXiv: 1511.05099.

[88] Quan-shi Zhang and Song-chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, January 2018.

[89] Quanshi Zhang, Xin Wang, Ying Nian Wu, Huilin Zhou, and Song-Chun Zhu. Interpretable CNNs for Object Classification. *arXiv:1901.02413 [cs, stat]*, March 2020. arXiv: 1901.02413.

[90] Yundong Zhang, Juan Carlos Niebles, and Alvaro Soto. Interpretable Visual Question Answering by Visual Grounding from Attention Supervision Mining. *arXiv:1808.00265 [cs]*, August 2018. arXiv: 1808.00265.

[91] Dawei Zhu, Aditya Mogadala, and Dietrich Klakow. Image manipulation with natural language using Two-sided Attentive Conditional Generative Adversarial Network. *Neural Networks*, September 2020.

[92] Xi Zhu, Zhendong Mao, Chunxiao Liu, Peng Zhang, Bin Wang, and Yongdong Zhang. Overcoming Language Priors with Self-supervised Learning for Visual Question Answering. *arXiv:2012.11528 [cs]*, December 2020. arXiv: 2012.11528.