Universität Augsburg

Fakultät für Angewandte Informatik

Lehrstuhl für Multimodale Mensch-Technik Interaktion

Bachelorarbeit:

Implementation and Comparison of Occlusion-based Explainable Artificial Intelligence Methods

von

Benedikt Limmer

Sommersemester 2020
Sachsenkamstraße 35, 81369 München
Bachelor Informatik, 8-tes Fachsemester
Matrikelnummer: 1490379
Betreut von Prof. Dr. Elisabeth André

Contents

1 Introduction

In recent years, the performance of deep learning algorithms has steadily improved. They are already outperforming humans in various areas like image classification [HZRS15], lipreading [ASWDF16] or localizing specific proteins in a human cell [RHR^{+}18]. However, there is still a big roadblock in the way of AI becoming even more relevant. With the growing complexity of these algorithms, it is hardly if at all possible to comprehend the decisions of neural networks [SB18]. They are therefore often referred to as "black box" algorithms. The neural network takes an input and then uses it to produce an output. From the human perspective, it is not apparent what happens in between and therefore how the output was computed.
Obviously, the main goal of a neural network should be to predict the correct result with a very high probability, but it is also highly desirable to know why a certain decision was made. This could be particularly interesting for applications where the network provides a proposal, but a human makes the decision in the end (e.g. the network proposes a medical diagnosis). The result of the algorithm is checked for plausibility and the confidence in the algorithm could be increased.
This is the problem that the research area of Explainable AI tries to deal with. The target is to shed light on the decision-making of neural networks and to generate comprehensible explanations for the output of these algorithms. In the case of Convolutional Neural Networks, these can be saliency maps, which highlight pixels that lead to the corresponding prediction.
The goal of this thesis is to evaluate and compare four different occlusion based explanation approaches, which generate such saliency maps. The functionality of these algorithms will be explained in Chapter 4, but the general idea is similar between all of them. After creating occluded versions of the original image, by covering certain regions, it is fed to the model. The results are then compared to the original ones. By doing so an explanation in the form of a saliency map can be created. This procedure stays the same for all the approaches, but they do differ in the way the images are occluded and the saliency map is ultimately computed.
To test the four approaches they will be implemented and applied for two different models. A reinforcement learning network which learned to play 2600 MsPacman, described in section 5.1 and a emotional recognition model described in section 5.2. After generating results for these models the explanations of the different approaches will be compared and tested in chapter 6.

2 Motivation for XAI

With the increasing importance of machine learning, the desire to understand the decisions of neural networks is growing. Especially in sensitive areas, where decisions have a direct impact on people, this is not only interesting but could become legally mandatory in the future [GF17]. If a bank denies a loan to a customer, based on the result of a neural network, the customer may have a claim to the reasons for this decision.
Besides the already mentioned legal reasons for using Explainable AI, there are other benefits to it:

Creating Trust If a user does not trust the result of a neural network, he will probably not use it in the long run. XAI can help to strengthen this trust by explaining the reasons for a decision and thus making it more comprehensible [RSG16b].

Validate Results The explanation of results can lead to the early detection of overfitting and biases. An example of this, are the results of a network, which predicted that patients with asthma have a particularly good chance of survival with pneumonia [TG19]. The prediction is obviously wrong and was caused by a systematic bias in the training data, as pneumonia patients with asthma were transferred directly to the hospital for intensive treatment. Without the recognition that asthma has a major influence on the prediction of the model, it would probably have been difficult to detect this bias.

Improving Models Besides detecting overfitting and biases, explainable AI can also help find problems in the architecture of the network. When visualizing the outputs of the model, the reasons for bad performance can become apparent. This can help developers to debug the network and thus potentially improve the performance. [BXS+20].

Learning from the Model Neural networks are very good at finding patterns and relationships in data or learning new profitable strategies. An example of this is AlphaGo, a neural network, which managed to beat the best human player in Go [LWY+16]. The model made some moves, that were not used by any professional player before. By understanding how a model comes to certain decisions, new insights can be gained. In the case of Go, this could lead to the adoption of new optimal strategies, by human players.

3 Convolutional Neural Networks

3.1 Basics

Convolutional Neural Networks [GBC16], [ON15] are Deep Learning algorithms, most commonly used for image recognition tasks. They are able to classify and detect objects. During the training, the Neural Network learns to extract features and structures, e.g. edges and other shapes, from the image and uses this information to make predictions about it.

3.2 Design

3.2.1 Input

Depending on the task of the network, the input can vary. Most commonly they are images. In the case of an RGB image, these are defined by height, width, and the three color channels. For a CNN to be able to work with an input, it must be preprocessed in advance. Firstly, convolutional neural networks expect a uniform input shape. The images which are supposed to be used as input can have different shapes and therefore might need to be resized or padded. Secondly, it can be useful to normalize the color channels to a value range, for example to values between [-1, 1]. The goal is now, to feed the input to the neural network and use mathematical operations, to extract the sought information from the pixels.

3.2.2 Convolution Layer

In a convolution layer, the extraction of the pixel information happens. For this purpose, a filter matrix is defined, which has a fixed height and width. The filter matrix now moves with a predefined stride across the input image and calculates a feature map. During the training, the values of each filter matrix, called weights, are adjusted, in order to be able to extract certain features from the input. However, not only one filter is used per convolution layer, but several, each one computing the probability of a different feature. This results in a higher dimensional feature map. After the convolution, the output is now determined by means of an activation function. In the case of CNN's, the ReLu function ($Relu(x) = max(0, x)$) is most commonly used, which ensures positive results.

3.2.3 Pooling Layer

Convolutional layers are oftentimes followed by a pooling layer. These are used to lower the complexity and thus reduce the calculation effort. There are different pooling approaches. Two well-known ones are Max- and average pooling. Both methods define a filter, which moves across the feature map. In the case of Max Pooling, the maximum value in the area of the filter is adopted, while Average Pooling calculates an average value. Depending on the filter size, the amount of data is significantly reduced, whereby ideally as little information as possible is lost.

3.2.4 Fully Connected Layer

The Fully Connected Layers are normally located after several repetitions of convolutional and pooling layers and calculate the output of the network. The previously calculated elements of the feature maps are multiplied with weights and combined to obtain the classification probabilities of the network. Usually, the dimensions of the feature map are reduced down to the number of possible classifications. This results in an output vector, in which each entry represents the probability for a certain class. To ensure a probability distribution between 0 and 1, a soft-max function is usually applied. To calculate these probabilities, the multidimensional feature map must first be converted into a one-dimensional feature vector. This is commonly referred to, as flattening the feature map.

3.2.5 Dropout Layer

Dropout layers are a possibility to regularize a CNN. During the training, after each convolutional and fully connected layer computed the corresponding feature map, there is a set probability for each entry to be dropped and thus replaced by 0. This procedure has the goal of preventing the network of overfitting to the training data. Overfitting happens when the model learns to recognize specific training data, rather than learning general features. This usually occurs when the training dataset is too small. Because in each training step a different subset of features is available to the neural network, the chance to recognize previously seen training data decreases. This forces the model to learn to detect general features, thereby reducing the risk of overfitting.

3.3 Training

During the training of a neural network, the actual learning happens. Training data is required to train a network. In the case of CNN's, these are often labeled images. During the training, the data is used as input of the neural network. The output of the network is then compared with the label and a loss function is used to calculate an error. The intuition behind neural networks is to adjust the weights (in the case of CNN's the values of the filters), so that future results are closer to the truth and thus the error is minimized. In order to know how to change the weights, an optimization algorithm is used. One of those algorithms is gradient descent, which calculates the partial derivatives of the loss functions with respect to the weights to minimize it. After a fixed number of training steps, the network is evaluated. An independent data set is used to check the accuracy of the network. During the training, the error of the model should decrease, while the accuracy increases.

4 Occlusion Analysis

When talking about Explainable AI a couple of characteristics have to be defined:

Ante-Hoc: Ante-Hoc approaches are used for algorithms, which are intrinsically explainable (e.g. decision trees). This means that the model is designed to be explainable. [Hol18]

Post-Hoc: Post-Hoc approaches generate explanations for "black box" models after the training is already completed. The model itself is not inherently explainable. [Hol18]

Global explainability: Global explanations try to clarify how certain input features of the whole data set affect the model's result. [CPC19]

Local explainability: Local explanations focus on explaining single predictions, the model made. [CPC19]

In this work, I will implement and compare different occlusion based explanation approaches. These approaches can be used to generate local post-hoc explanations, specifically in the context of convolutional neural networks. Given a model $f : \mathcal{I} \rightarrow \mathbb{R}^n$, which determines the classification probability for n classes, an image I taken from a set Images $\mathcal{I} = \{I | I : \Lambda_I \rightarrow \mathbb{R}^c\}$ is used as the input. The set of images $\mathcal{I}$ is defined by a height and width $H \times W$ and the number of color channels c. Each of the images $I \in \mathcal{I}$, which can be used as an input for the model, are mappings of coordinates $\Lambda_I = \{1, ..., H\} \times \{1, ..., W\}$ to c color channels. This means that an image is defined as a set of pixels restricted by height and width. Each pixel is defined by a set number of channels and a value range. An RGB image for example is defined by a fixed height and width, three color channels (each of them representing the share of red, green, and blue), and a value range of $[0, 255]$. An RGB image, therefore, consists of $H \cdot W$ pixels, each having three values. Each of those values corresponds to one color channel and lies in a value range of $[0, 255]$.
The input image I is fed to the model resulting in a prediction with n different classification probabilities. Each of these probabilities correlates with one of n possible classes and indicates the confidence that it was detected. The goal of the occlusion analysis approaches is to explain those results by analyzing pixels or pixel regions and their impact on the classification. This is done by modifying the original image, occluding, or blurring parts of the image. To create an explanation, the newly modified image is fed to the neural network, resulting in a second output. The occluding and feeding to the network is done multiple times, each iteration perturbing different parts of the image.

The intuition behind this approach is that the classification should drastically change when an important pixel is occluded. Similarly, the occlusion of pixels that have little impact on the classification of an image and therefore are less important, should not drastically change the result. After occluding different parts of the image and analyzing the differences in results, a saliency map is created. This map gathers all the previously acquired information and portrays them in the form of a heat map, highlighting the most important pixels and thus showing, which pixels impacted the decision of the network the most. A distinct advantage of occlusion based approaches is that they are model agnostic. Meaning no direct adaptions to the model have to be made in order to generate an explanation. Only the inputs and outputs are needed to make the algorithms work. The explanations in the form of saliency maps give a good first insight into the most important regions of the image, which were used to make a certain decision.
After explaining the core functionality of occlusion based approaches, it is now time to explore different variations of this technique. The basic procedure stays the same between all these approaches, but they vastly differ in how the images are occluded and thus saliency maps are generated.

4.1 Occlusion Sensitivity

Probably the most basic and straight forward occlusion analysis approach is done in the Occlusion Sensitivity approach by Zeiler et al. [ZF14]. To create an explanation for the original image I, it is occluded by a grey patch P, which is a square defined by a height and width $ps \times ps$. The patch is shifted across the input image with a horizontal and vertical stride of ps, each time setting the values beneath the square to grey. Every time the patch is shifted, this results in the patch being at a new pixel $\lambda_I \in \Lambda_I$ (The pixel λ_I corresponds to the top left corner of the grey patch). A patch located at a pixel λ_I is defined as $P(\lambda_I)$. Because the stride and the size of the patch are equal, every pixel is covered eventually. This operation is expressed with $\odot$ and highly depends on the value range of the original image. Given an RGB Image with the value range [0,255], all color channels are set to 127. In the case of a greyscale image with the value range [0,1], the one channel is set to 0.5. The result of the operation being $\lceil H/ps \rceil \cdot \lceil W/ps \rceil$ occluded variations of the original image, each having a different region covered with the grey square $I \odot P(\lambda_I)$. In this case, $\lceil x \rceil$ is defined as the ceiling operation, which takes a real number x and outputs the next highest integer.
These occluded versions are then fed to the network resulting in a new prediction for the different classifications $f(I \odot P(\lambda_I))$. Since one classification is explained at a time, only the probability of this classification is considered in the following calculations. This classification probability is referred to as $confidence$. The $confidence$ of all occluded variations of the image are then used to compute a saliency map $S : \Lambda_{I'} \to \mathbb{R}$, where $\Lambda_{I'} = \{1, ..., \lceil H/ps \rceil\} \times \{1, ..., \lceil W/ps \rceil\}$. Since the grey square is shifted across the image with a set stride, each entry of S corresponds to one occluded version of the original image where a specific region was covered. The $confidence$ of that occluded image is used to calculate the importance of that covered region.

$$S(\lambda_{I'}) = 1 - confidence(\lambda_{I'})$$

In the formula above the saliency map is computed, with each entry of $S(\lambda_{I'})$ representing the importance of a region covered by the grey square. This value is computed for all $\lambda_{I'} \in \Lambda_{I'}$. The intuition behind this formula is that if an important pixel is covered by the grey square the confidence for the corresponding classification should notably drop and therefore the saliency score is higher. Because the saliency score is computed for every covered variation of the original image, the saliency map has a size of $\lceil H/ps \rceil \times \lceil W/ps \rceil$. The map is upsampled to the size of the original image and can then be used as a colormap to highlight the most important regions of the image, which the neural network used to make a prediction.

4.2 RISE: Randomized Input Sampling for Explanation of Black-box Models

Randomized Input Sampling for Explanation (RISE) [PDS18] works similarly to the Occlusion Sensitivity approach, but the occlusion of pixels is more complex. Instead of using a gray square to cover up pixels, masks with the same size as the image are generated. These masks are acquired by producing smaller masks, where each element is set to 1 with a certain probability p and 0 otherwise. Those binary masks are then upsampled to the size of the image using bilinear interpolation. This results in masks with the same size as the image, where all the elements have a value between 0 and 1. The bilinear interpolation is done to smooth the edges of the mask.
To generate an explanation for a given classification, the image I has to be occluded first. This is done by computing an element-wise multiplication $\odot$, with a previously generated mask $M : \Lambda_I \rightarrow [0, 1]$. The modified image is then used to generate a new prediction $f(I \odot M)$. This step is repeated N times, each time using a different Mask M_n, taken from a set of masks $\{M_1, ..., M_N\}$ and thus occluding different regions of the image. The results are then combined, computing the importance $S_{I,f}(\lambda_I)$ of each pixel $\lambda_I \in \Lambda_I$, to the resulting classification:

$$S_{I,f}(\lambda_I) = \frac{1}{p \cdot N} \sum_{i=1}^{N} f(I \odot M_i) \cdot M_i(\lambda_I)$$

This formula calculates the importance of a pixel λ_I to a prediction of model f. To do this, all predictions where the specific pixel was visible, are added together. This is the case if the value of the mask $M_i(\lambda_I)$ at the position of the pixel is not 0. The summed up predictions are then divided by the number of masks N and the probability p that a value of the mask was originally set to 1, to calculate an average across all occluded variations. After computing the importance of every pixel, the results can be visualized

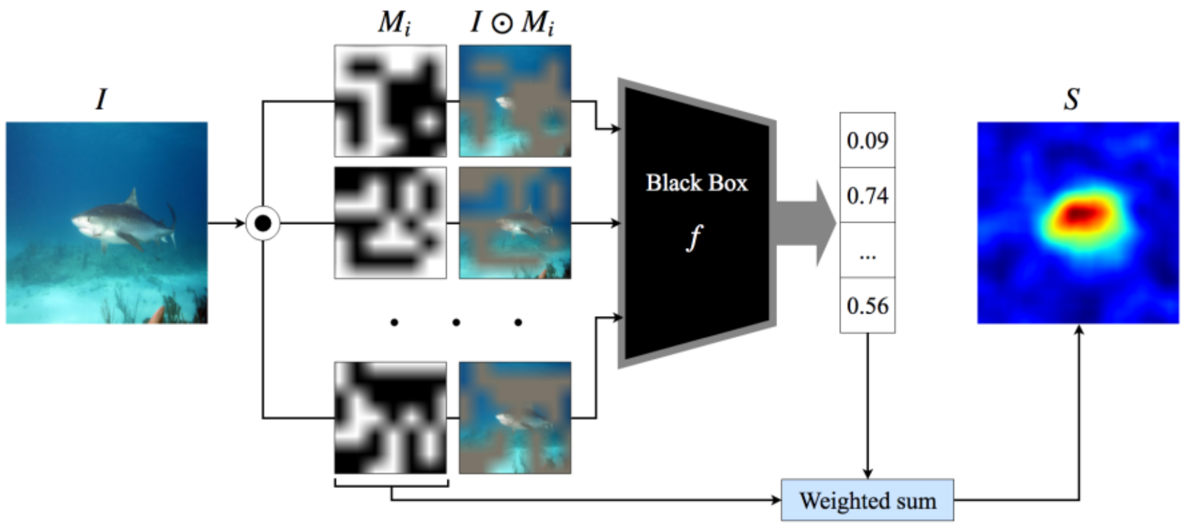


Figure 4.1: RISE occludes different regions of the image with masks. The images are then fed to the model resulting in new classification probabilities. Those probabilities are then used to compute a weighted sum of the masks, which can be used as a saliency map. (The figure was taken from [PDS18])

in the form of a saliency map, highlighting the key pixels, which lead the model to make a certain prediction. The intuition behind this approach is, that if a mask covers important pixels, $f(I \odot M_N)$ will produce a low probability. Likewise $f(I \odot M_N)$ will produce higher probabilities if the mask does not cover important pixels. Following this intuition, pixels that lead to a low classification probability will be valued higher in the saliency map than those who produced a higher score when covered by a mask. A summary of the whole procedure can be seen in figure 4.1

4.3 Noise Sensitivity

The Noise Sensitivity approach was first introduced by Greydanus et al. [GKDF18]. When compared to other occlusion approaches it mainly differs in the way the images are occluded. Instead of covering up image regions, they get blurred. This means that instead of taking away information, uncertainty is added to it. Apart from the occlusion, the procedure stays the same. After blurring the image, it is fed to the network. The results are then compared to the original ones, identifying the most important image regions. To generate an explanation for an Image I it has to be occluded $\Phi(I, \lambda_I)$ first:

$$\Phi(I, \lambda_I) = I \odot (1 - M(\lambda_I)) + A(I, \sigma_A) \odot M(\lambda_I)$$

The formula above is composed of different parts, which are now defined. Given an Image I, a Gaussian blur $A(I, \sigma_A)$ is added around an area, specified by a mask $M(\lambda_I)$.

The blur corresponds to a 2D Gaussian, centered at $\mu = \lambda_I$. The 2D Gaussian is defined as the mathematical convolution [Wik20] with the Gaussian function, which takes the pixels in the neighborhood of μ and computes a weighted average, thereby blurring the image in that area. This creates a perturbed version of the original frame, adding noise to a certain area. The image is then fed to the model, resulting in a new prediction. To determine the saliency scores $S(\lambda_I)$ of a certain region around λ_I, the outputs of the logit units of the perturbed input $\pi(\Phi(I, \lambda_I))$ are compared with the outputs for the original image $\pi(I)$. The logit units $\pi(x)$ in this context are defined as the output of the layer before the final softmax activation of the model. They are therefore similar to the classification prediction score of the other approaches.

$$S(\lambda_I) = \frac{1}{2}||\pi(I) - \pi(\Phi(I, \lambda_I)||^2$$

The above-described procedure calculates the importance S of the area around λ_I. Ideally, this is done to every pixel, but to safe computational cost it seems sufficient to only determine the importance score for pixels in a set interval. The intuition behind this approach stays the same, as in the previously discussed ones. If an area is perturbed and the prediction drastically changes, the occluded pixels had a big impact on the decision of the original input. Based on the results of $S(\lambda_I)$ a saliency map can be created to visualize the importance of the different image regions.

4.4 LIME

Local Interpretable Modelagnostic Explanations (LIME) introduced by Ribero et al. [RSG16b] is probably one of the most popular explainability approaches. It is mostly used to generate explanations in the context of computer vision and natural language processing. For both types of tasks, the procedure is similar. Just like in the other occlusion based approaches, some part of the input is changed, in order to observe differences in the output. In the case of NLP, some of the input words are removed and in the case of image classification, some pixels are covered.
To create an explanation for an input image I, it is divided into several superpixels. The superpixels are found by using image segmentation algorithms and contain interesting pixel regions of the image I. In the context of image segmentation, interesting regions are parts of the image with the same color or texture and regions that are divided by edges. This results in a binary representation $B \in \{0, 1\}^d$ of the image. The binary representation indicates whether a superpixel is “present” or “absent”, d being the number of superpixels. Initially, all superpixels are ”present” and the values of B are therefore all 1. After the segmentation, occluded versions of the original image are created, by setting certain values of B to 0, resulting in $B' \in \{0, 1\}^d$. This means that specific superpixels are being covered by setting the corresponding pixel values to 0 and thus creating occluded versions of the original image $\mathcal{I}' = \{I' | I' : \Lambda_I \to \mathbb{R}^c\}$, each occluded image I' having different permutations of “absent” superpixels. After that the occluded images

are fed to the original network, resulting in new predictions $f(I')$. The occluded images and corresponding predictions are then used to train a local linear regression model g, which estimates how much each superpixel contributes to the original prediction. The goal is to generate an explanation $\xi(I)$ by optimizing the following equation:

$$\xi(I) = \text{argmin}_g \mathcal{L}(f, g, \Pi_I) + \Omega(g)$$

Where:

$$\begin{aligned}
&\mathcal{L}(f, g, \Pi_I) = \sum_{B', I' \in \mathcal{I}'} \Pi_I(I')(f(I') - g(B'))^2 \\
&\Pi_I(I') = exp(-D(I, I')^2/\sigma^2) \\
&g(B') = w_g \cdot B'
\end{aligned}$$

The optimization is done by minimizing $\mathcal{L}(f, g, \Pi_I)$, which calculates the imprecision of g at approximating f, while keeping $\Omega(g)$, the complexity of the model, at a human-comprehensible level. The complexity $\Omega(g)$ is defined as the measurement of how understandable the model is. In the context of a neural network, this can be the number of weights or how many hidden layers it has. $\mathcal{L}(f, g, \Pi_I)$ is computed by summing up the distances $\Pi_I(I')$ of the original images I to the perturbed version I' multiplied with the corresponding error between the prediction of the original model f and the linear model g. The distance between the images is calculated to give more weight to the occluded images with high similarity to the original image. To calculate the distance $\Pi_I(I')$ between the images, a distance function D and width σ is used. The width σ determines how close the occluded image has to be, in order to impact the local model. For image classification tasks the L2 distance is most commonly chosen. During the training, the weights w_g of the interpretable linear model are adjusted in order to get high local fidelity, while ensuring interpretability. The weights of this model can now be visualized to highlight those superpixels, which had the biggest impact on the decision of the network and thus explaining its behavior locally. Because every weight represents one superpixel, those parts of the image with the highest weights can be highlighted, giving an insight into the decision-making process of the model. An example of the whole process can be seen in figure 4.2.

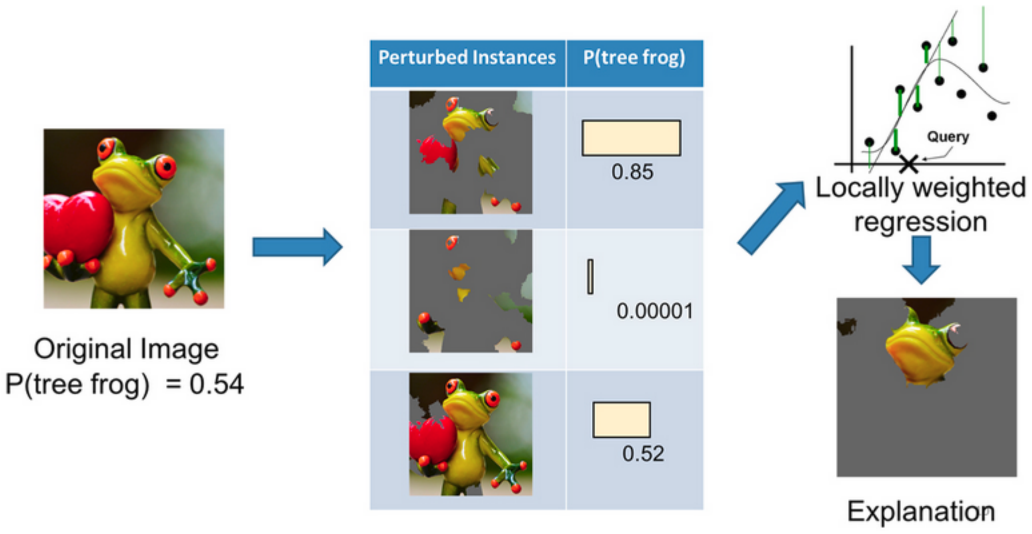


Figure 4.2: In LIME parts of the image are segmented and then partly occluded. The linear model is then trained to evaluate the importance of each super pixel. (The figure was taken from [RSG16a])

5 Implementations

To compare the differences and advantages of the approaches described in chapter 4, explanations for two different models are generated. The two resulting sets of explanations can then be used to evaluate the different algorithms. Explaining more than one model also makes it easier to investigate the property of model agnosticism, which is listed as a major advantage of occlusion based approaches and implies that the model itself does not have to be touched when generating explanations for it.

5.1 Reinforcement Learning Network on Atari Games

In the following experiments, a deep reinforcement learning algorithm (DQN) was used, which learned to play Atari 2600 MsPacman. The model was trained with the DQN implementation of the OpenAI Baselines Framework [DHK$^+$17].
In Pac-Man, the player navigates the character through a two-dimensional maze and tries to maximize points by "eating" dots and fruits, while simultaneously being chased by ghosts. The player loses a life when touched by a ghost unless Pac-Man consumes a "power pellet", which grants a temporary ability to eat ghosts. After clearing all the dots, the player reaches the next level, which resets the playing field and increases the difficulty. The game ends when the player loses all three of his lives.
The model makes predictions by observing the current and past frames of the game and then choosing from a pool of 9 possible actions. The frames are RGB Images with a size of 160×210, displaying a state of the game. Based on the newest frame and three older ones a prediction for the next, most optimal action, is made. To do this the frames are converted to an 84×84 greyscale version, with a value range of $[0, 1]$. The frames are then stacked together, resulting in a $4 \times 84 \times 84$ input to the network. After passing the stacked frames through the convolutional network a vector of 9 values is put out, each entry representing one possible action. The action with the highest value is then chosen and executed in the game, resulting in a new frame. This frame is then preprocessed and added to the stacked frame, replacing the oldest of the 4 frames in the stack. The modified stack is then used to generate a new action. This procedure continues until a finished game state is reached.
To compare the different explanation methods, different game states are chosen and explained by each approach. The results can then be analyzed and evaluated. The game states which are to be explained were chosen by the HIGHLIGHTS algorithm, as described in [HWAA20]. The goal of the algorithm is to find the most important states in the simulated game. An important state is defined as a state that has a major impact on future rewards. This means that choosing a wrong action at the given state will lead

to worse rewards for the remaining run.
The intuition behind explaining the HIGHLIGHT game states is that those are the most impactful decisions of the model. Therefore understanding these actions will offer the most insight when trying to understand the decisions of the model.

5.2 Emotion Recognition Network (Affectnet)

To generate the second set of explanations a convolutional neural network, detecting emotions on images of human faces was chosen. The model was created using VGG-Face [PVZ15], a popular Convolutional Neural Network Architecture. The network was trained on the AffectNet dataset [MHM17], which is a database of over 1 million annotated facial images.
Given an input image that is fed to the model, an output vector with 8 values is produced by the algorithm. Each of these values corresponds to one of 8 possible emotions, with the value indicating the probability it was detected. During the training, the model learned to differentiate between an angry, contempt, disgusted, fearful, happy, sad, surprised, and a neutral facial expression. To generate such an output the image is preprocessed first. Given an RGB image with a variable size of $X \times Y \times 3$ it is resized to a fixed input size of $224 \times 224 \times 3$. After that the pixel values are converted from a value range of $[0, 255]$ to $[-1, 1]$. The preprocessed image can now be used as input to the network.
Just like in the previously described reinforcement network, the different occlusion based approaches can be used to generate explanations for different input images. This provides a second basis on which the approaches can be compared and evaluated.

5.3 Adjustments

After implementing the Noise Sensitivity and Occlusion Sensitivity approach and generating explanations for the Pac-Man reinforcement learning network, it seemed like they produce really poor results when compared to the other algorithms. These problems only occurred for the Pac-Man model and not for the emotional recognition network. After investigating potential problems with both approaches and testing different adjustments it appears like the problem originates from the way the frames are occluded.
After visualizing occluded versions of the input frame (as seen in figure 5.1) a potential problem became apparent. Both the Lime and RISE approach set the occluded pixels to 0 and thus convert them to black. Meanwhile, the Noise Sensitivity and Occlusion Sensitivity approach use grey shapes, to occlude the input frames. On the one hand, the Occlusion Sensitivity approach uses a grey square, and Noise Sensitivity on the other hand uses a gaussian blur. Because the input frames are converted to greyscale images and the walls of the Pac-Man playing field are grey, the gaussian blur is also predominantly grey. This potentially causes the problem that the occlusion is not drastic enough and the inserted shapes are recognized as walls and therefore do not influence the output enough.

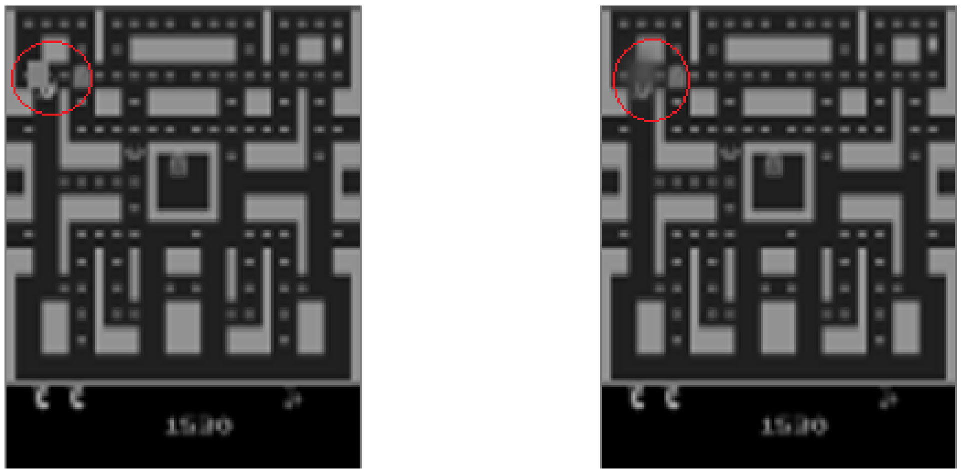


Figure 5.1: Occlusion Sensitivity (left) and Noise Sensitivity (right) approach before implementing adjustments. Encircled in red is the area where the occlusion happens.

The obvious fix for this problem is to try a different color when occluding the input frames. Because black seems to work for the other two approaches this color is chosen. In the case of the Occlusion Sensitivity approach, the adjustment is rather small. Instead of using a grey square, a black square is used to occlude the image.
For the Noise Sensitivity approach, the functionality has to be changed more. As described in section 4.3, the algorithm uses a gaussian blur to obscure a region of the image, based on the colors of the surrounding pixels. Since the walls of the playing field and the points are grey the blur tends to be also grey. As a first approach to fix the issue the “blurring”-strategy was abandoned in favor of just straight up covering the frame. Inside the area, which used to be blurred, the pixels are set to 0 and therefore converted to black.
The resulting occluded images after the adjustments can be seen in figure 5.2. The resulting explanations will be discussed in chapter 6, but it appears like there is a major improvement after applying the changes.
A problem that arises after adjusting the algorithms is the similarity between the Occlusion Sensitivity and the Noise Sensitivity approach. Before the changes, they differed mainly in the way the occlusion was done. One covered the frame, the other blurred it. After the changes, both algorithms cover the image with a black shape. Now, the only difference is the shape of the occlusion (as seen in figure 5.2), the way the shape is shifted across the frame, and how the results are ultimately computed. After reviewing the explanations it appears that both algorithms produce similar results with small differences. It does not seem possible to adjust the Noise Sensitivity algorithm in a way, where the blurring is the main source of occlusion and the results are usable when explaining the Pac-Man model.
An interesting note on this problem is that the original paper of the Noise Sensitivity approach [GKDF18] also noted that the explanations for the Pac-Man example were not

particularly informative, mostly highlighting the walls of the maze. This poor performance could potentially stem from the same issue.

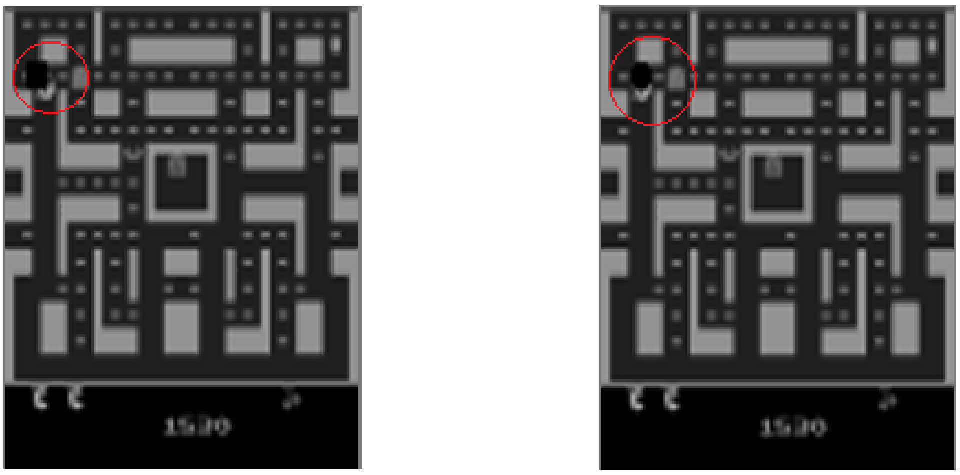


Figure 5.2: Occlusion Sensitivity (left) and Noise Sensitivity (right) after implementing the adjustments. Instead of using a grey shape, a black one is used to occlude the frame.

6 Presentation of Results

In this chapter, the explanations of the four occlusion based explanation approaches, presented in chapter 4 will be shown. Furthermore, two ways of evaluating the quality of these explanations and the corresponding results of the two tests will be presented. Both the explanations and results of the tests will be discussed in the next chapter.

6.1 Visual Assessment of Reinforcement Learning Explanations

The first set of evaluations were generated for the reinforcement learning algorithm which learned to play Pac-Man. To find suitable game situations the HIGHLIGHT algorithm, as described in section 5.1 was applied to capture four important game states. Given those game states, the different explanation approaches were used to generate an explanation for the decision the algorithm made. The goal is to highlight those areas that had the greatest influence on the network when choosing the next action. The resulting explanations can be seen in Figure 6.1.

Occlusion Sensitivity For all four game states, the occlusion sensitivity approach clearly highlights Pac-Man and its surroundings. In the first game state, it appears like both the Pac-Man and the ghost, which Pac-Man tries to eat, are highlighted. In the third game state, the ghost above Pac-Man is highlighted. There seems to be a clear focus on Pac-Man and the ghosts if they are close enough. One other observation is that in state 1 and 4 the area which Pac-Man is moving towards is also highlighted.

RISE The RISE approach also clearly highlights Pac-Man in all 4 explanations. Besides that, there are some other slight highlights across the image, most notably around the life count and the score.

Noise Sensitivity Like in the previous two approaches the explanations produced by the noise sensitivity approach are also highlighting Pac-Man and the regions around him. In the first state, there seems to be a second highlight spot beneath the middle area of the playing field. In the fourth game state, there is a light highlighting on the ghost in the right bottom corner. Overall the explanations are very similar to the ones produced by the occlusion sensitivity approach. The main focus is always on Pac-Man and ghosts if they are close enough.

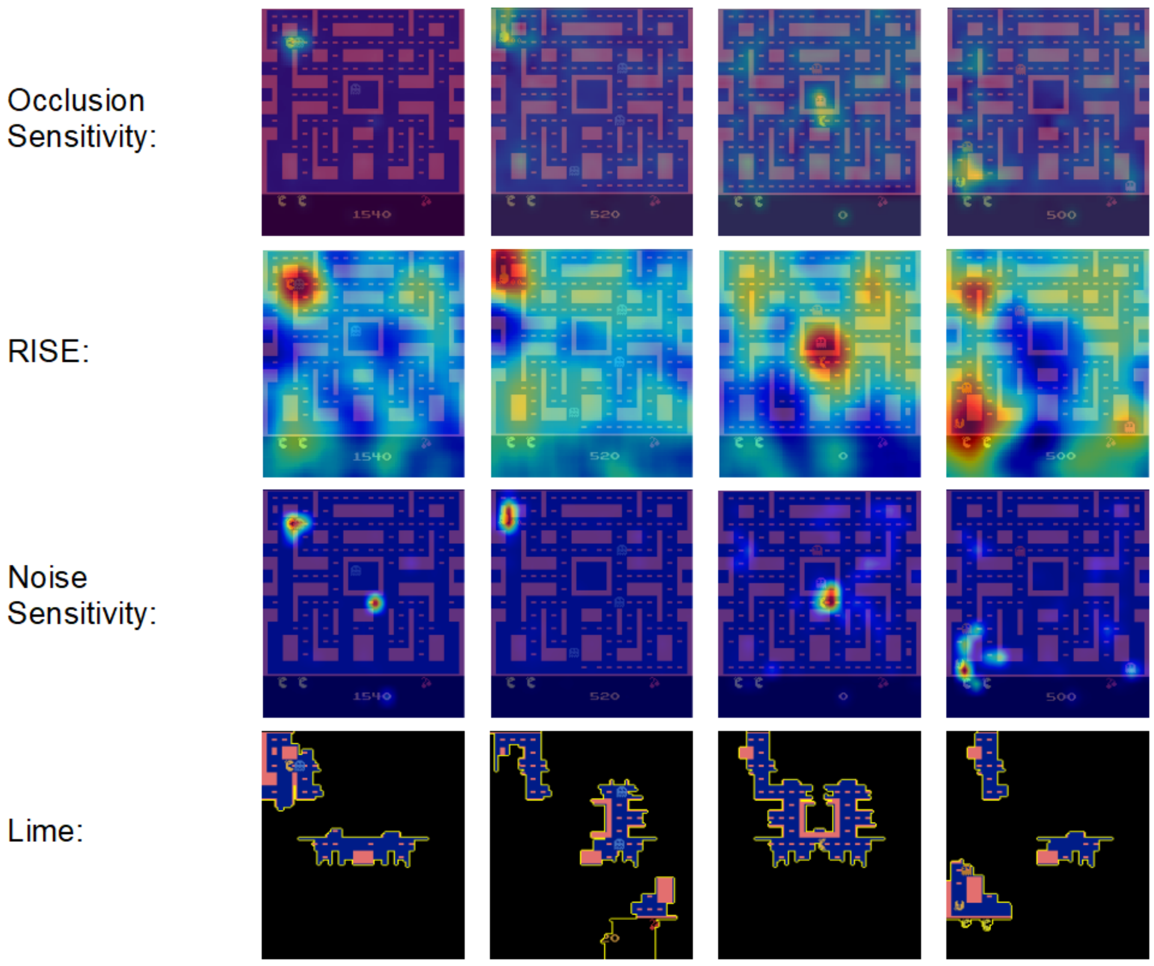


Figure 6.1: Explanations for different Pac-Man game states, based on the explanation approaches described in chapter 4.

LIME The explanations produced by the LIME approach look a bit different from the other results. That is because LIME produces a vector as explanations. This vector ranks superpixels by importance. This vector can then be used to produce an explanation, by only showing the most important superpixels. Just like in the previous explanations Pac-Man is in those regions. Only in the second game state, Pac-Man is not directly visible, but instead, the region above him where he is moving towards. It is also notable that in 3 of the 4 explanations one or more ghosts are visible. When compared to the other approaches the superpixels shown by LIME are similar to the areas highlighted by the saliency maps. The only bigger difference appears in state 3, where Pac-Man is not visible, but rather the area where he is moving towards. It is also noteworthy that LIME is the only approach highlighting both ghosts in the second explanation.

6.2 Results of the Reinforcement Learning Explanations before adjustments

As discussed in section 5.3, the resulting explanations for the Noise Sensitivity and the Occlusion Sensitivity approaches were initially both very poor. Examples for both explanations produced by the two approaches before adjusting them can be seen in figure 6.2. When observing the explanations the shortcomings become apparent rather quickly. The saliency maps never focus on Pac-Man or the ghosts. Instead, seemingly random areas of the playing field are highlighted. The reasons and fixes for the problems were already discussed in section 5.3. The results of the explanations after the adjustments were already discussed in the previous section and can be seen in figure 6.1.

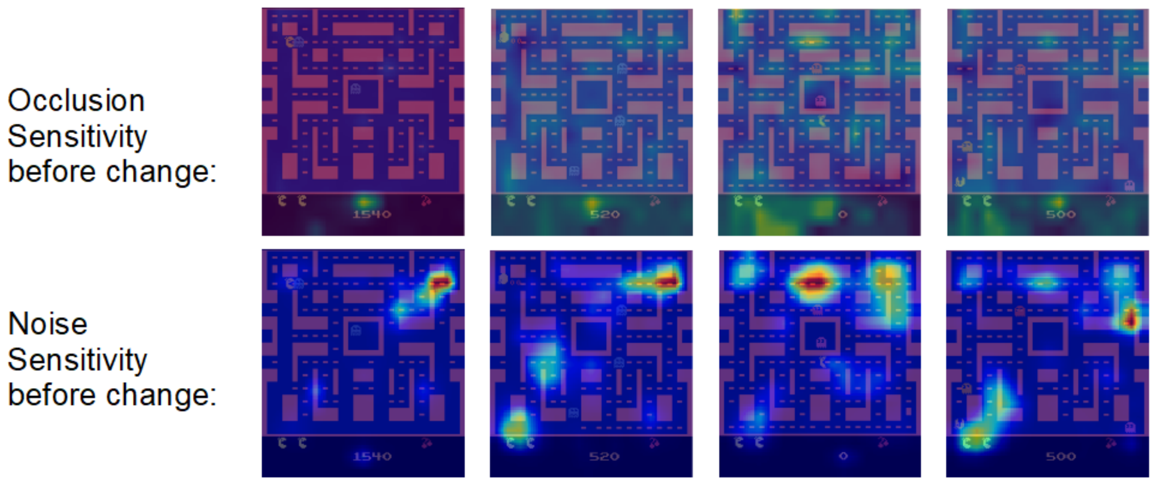

Occlusion Sensitivity before change:
Noise Sensitivity before change:
1540
520
0
500
1540
520
0
500

Figure 6.2: Explanations for different Pac-Man game states, produced by the Occlusion Sensitivity and Noise Sensitivity approach before implementing the adjustments.

6.3 Visual Assessment of Emotion Recognition Explanations

For the second set of explanations, a convolutional neural network detecting emotions on human faces, as described in section 5.2 was chosen. Given four different facial expressions (the images are available under the creative commons license), explanations were generated with each of the presented occlusion based explanation approaches. A notable fact is that the third facial expression was falsely classified as happy, with the correct label being disgust. Therefore the explanation highlights the regions, which led to the wrong classification.

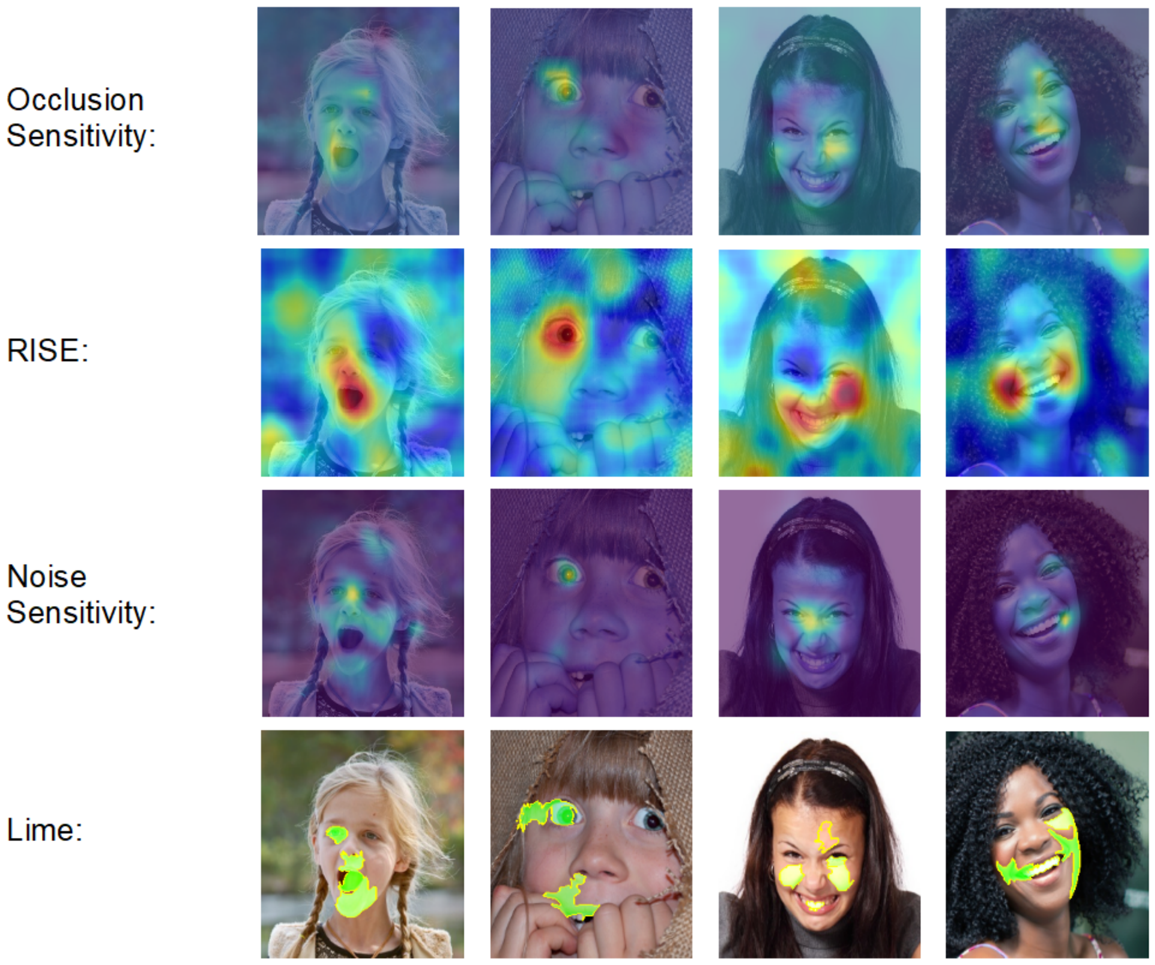


Figure 6.3: Explanations for different facial expressions, based on the previously described explanation approaches. The emotions from left to right being: surprise, fear, disgust, and happiness. As already mentioned, the third image was incorrectly classified as happy.

Occlusion Sensitivity On the first image, which was correctly classified as surprise, the wide-opened mouth is clearly highlighted. Apart from that, there is a highlighted spot above the left eyebrow. On the second image which was correctly classified as fear, the most notable highlight is in the wide-opened right eye of the child. The third explanation for the incorrectly classified image highlights the area beneath the left eye and on the right cheek. The fourth emotion was correctly classified as happy. On the resulting explanation, a highlight across the left half of the face right next to the nose can be seen.

RISE Like the previous approach, the RISE approach also highlights the mouth and the eye for the first two images. On the last two explanations, there is a clear focus on the mouth area of the face. On the third explanation, the whole mouth is highlighted, while on the last explanation only the corners of the mouth are highlighted.

Noise Sensitivity On the first explanation, the noise sensitivity approach diverges from the other approaches. Instead of the wide-opened mouth, the area between the eyes is highlighted the most. Besides that, other parts around the mouth are lightly highlighted. The second explanation is similar to the other approaches and clearly highlights the wide-opened right eye. Just like in the first explanation the third explanation also highlights the area between the eyes. The last explanation mostly highlights the left corner of the mouth.

LIME The results of the LIME approach are very similar to the ones of the rise approach. The first explanation highlights the opened mouth. The second explanation has a focus on the right eye and the area on the right side between the mouth and the mouth. On the last explanation, both corners of the mouth are highlighted. The only real difference between the result of both approaches is the third explanation. Here the lime approach focuses on the cheeks and frontal teeth, while RISE highlights the whole area around the mouth.

6.4 Sanity Checks

After evaluating the results of the different explanation approaches it appears like they oftentimes highlight the same areas, but not always. This leads to the potential problem of having to choose the best approach for a given task. If for a given model two explanations show two slightly different results there is no way of knowing which explanation is better. One way, to critically evaluate the performance of explanation methods are sanity checks, as described in [AGM+18]. The paper proposes a parameter randomization test, which evaluates how much an explanation approach relies on the model parameters. Because saliency maps aim to visualize the decisions of a neural network, ideally there should be a strong correlation between the parameters of the network and the map. To check for this correlation the already trained network is taken and the parameters are consecutively randomized. In each iteration the parameters of a new layer are changed, starting from the output layer. Every time a new layer is randomized, explanations are generated for this version of the model. These results are then compared to original explanations. To evaluate the correlation of two saliency maps three different metrics are used:

Pearson: The Pearson correlation coefficient can be used to measure linear correlations between two datasets. When comparing two images, each pixel is compared with its equivalent in the other image. This results in a metric that indicates the similarity of both images. In the case of comparing two saliency maps, this is not directly computed on the normal representation of the maps, but rather on their gradient representation. To do this the Histogram of oriented gradients (HOGs) is created for both maps.

Spearman: The Spearman rank-order correlation coefficient also measures the correlation between two datasets (saliency maps). The Spearman metric is very similar to the Pearson metric, but instead of using the pixel values to determine the correlation coefficient, their ranking is used. To do this the data is ordered and ranked first. The rankings are then compared pairwise and used to compute the correlation between the datasets.

SSIM: The Structural SIMilarity index (SSIM) [WBSS04] is a metric used to compare the structural similarity between two images. Besides structural similarity, two perceptual phenomena, luminance masking, and structural masking are also factored in when computing the index. Luminance masking is a phenomenon which suggests that differences in brighter image regions are less noticeable. Structural masking on the other hand states that differences in regions with high activity are also less noticeable.

All of the three metrics return a result between 0 and 1, were 0 represents no correlation between two maps and 1 represents completely identical maps.
The intuition behind this approach is that when a strong correlation between model parameters and explanation exists, randomizing parameters will cause a big change in the explanation. When compared to the original explanation the metrics should show a very low similarity between the two saliency maps. Accordingly, a strong similarity between the explanation of the randomized model and the original explanation indicates independence between the model parameters and the explanation approach.
This sanity check can be used to produce a metric that evaluates a desired property for explanation approaches and therefore can be used to compare the methods. The lower the similarity scores, the better.

6.4.1 Applying sanity checks on occlusion based explanation approaches

To check the correlation between the results of the four presented occlusion based explanation approaches and the model parameters the parameter randomization test is applied. To generate the different correlation metrics the model has to be progressively randomized first. The reinforcement network described in section 5.1 is chosen for this task. Because the model has five layers (three convolutional and two fully connected layers), five randomized variations of the model are created. Beginning from the output layer an additional layer is randomized for each new variation. This means that in the first variation of the model only the last fully connected layer (`fc_2`) is randomized, in the second variation the two last layers are randomized (`fc_2` and `fc_1`) and so on. After the randomization step is completed, explanations are generated for the five variations of the model. A sample of these explanations can be seen in figure 6.4.

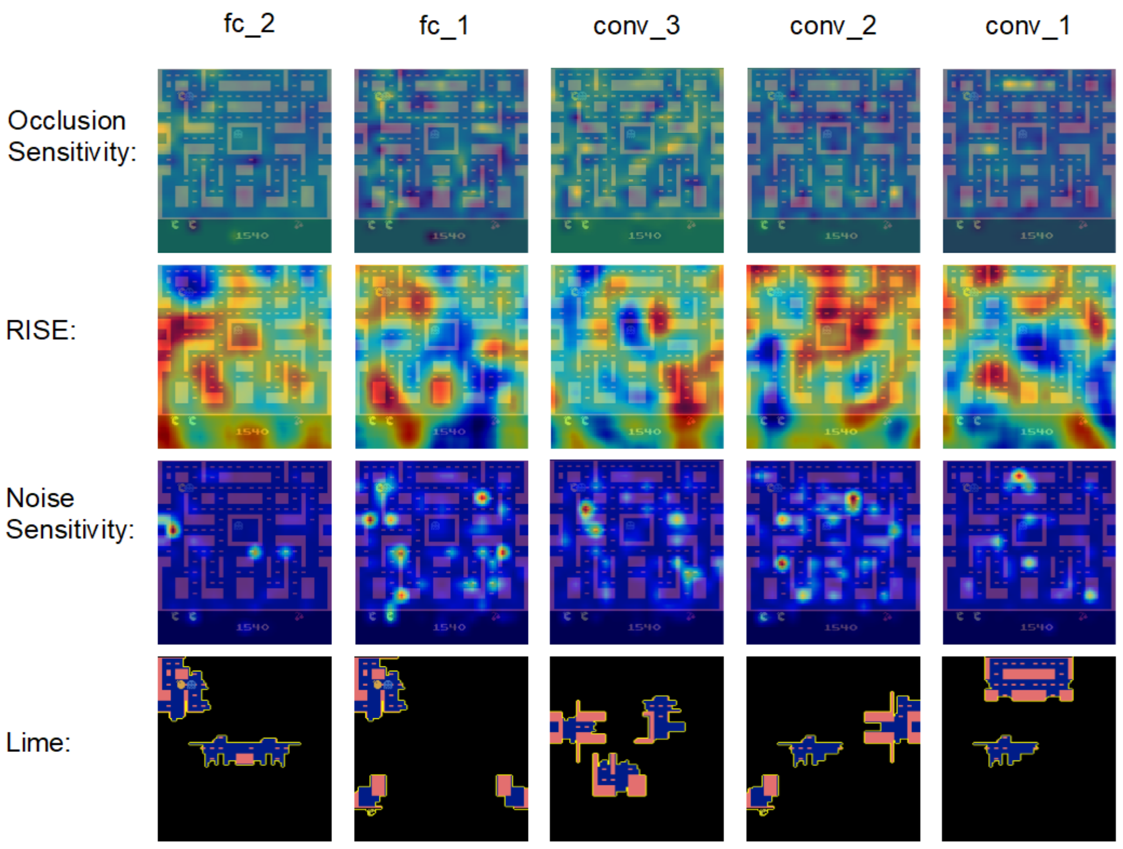


Figure 6.4: Explanations, produced by the different explanation approaches for the progressively randomized model. The expressions on top can be understood as the layer up until which the layers have been randomized, beginning from the back. For example `conv_2` means that starting from the back of the model all layers up to this point have been randomized (`fc_2`,`fc_1`,`conv_3`,`conv_2`).

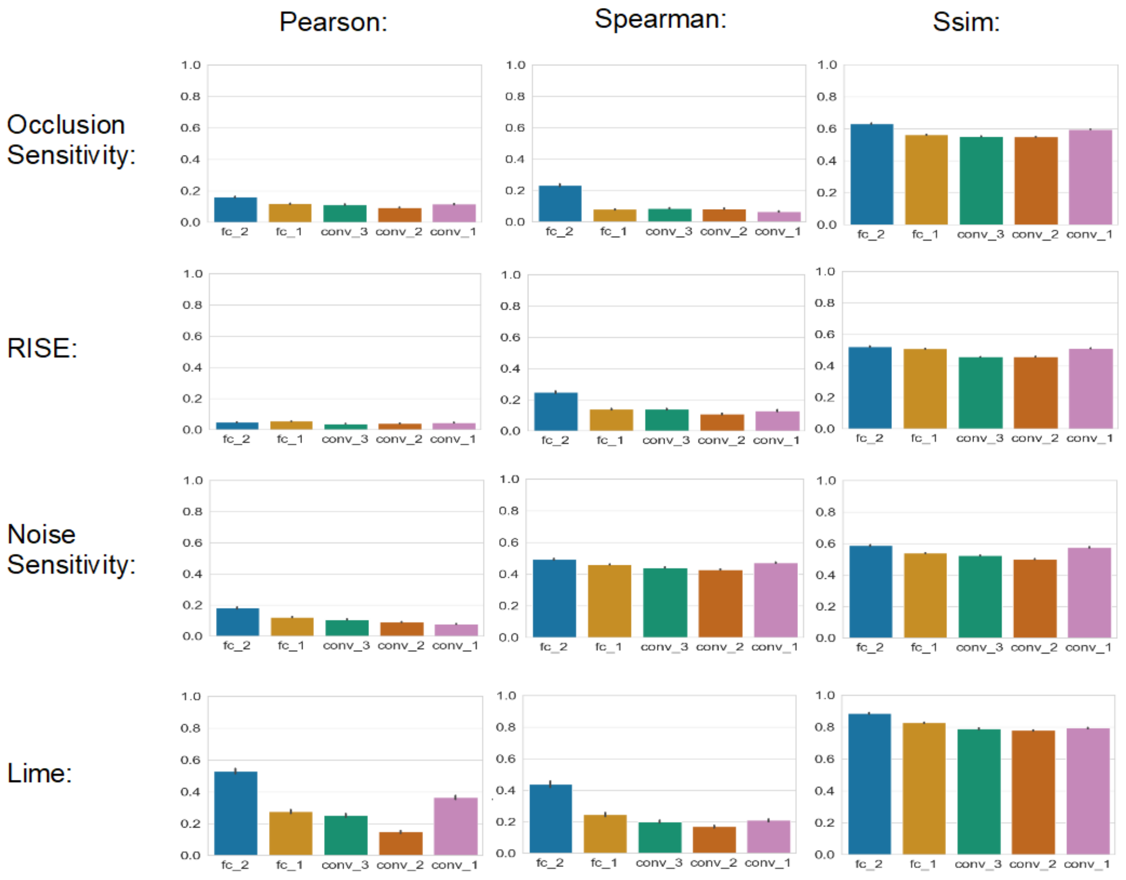


Figure 6.5: Results of the parameter randomization tests for the different explanation approaches.

6.5 Evaluating Correctness of Explanations

The problem of having to decide which algorithm and which explanation to choose was already mentioned in section 6.4. The presented parameter randomization test offers a way to check for the correlation between the model parameters and the explanations. While this test measures a desirable property of an explanation technique and therefore can be used to compare different explanation algorithms, it does not measure the correctness of the produced explanations. What this means is that the parameter randomization test does indicate whether an explanation approach relies on the model parameters when creating an explanation, but it does not measure if the highlighted pixels of the explanations are the correct ones. Correctness [PDS18] in this context means that pixels with the biggest impact on the decision of the model are the ones being highlighted by the explanation. One straight forward way to check the correctness of an explanation is the human eye test. In the cases of the Pac-Man and the emotional recognition model, a human can rationalize the explanation and argue whether they make sense or not. Besides the fact that this is obviously not the most accurate way of measuring the correctness of an explanation, this approach might become really hard if not outright impossible in more abstract domains, like cancer detection. Because of these problems, it would be great to have other ways of measuring the correctness of an explanation. Two methods that try to tackle this issue are the insertion and deletion game, proposed in [PDS18]. Given the original image and the corresponding explanation in the form of a saliency map, both approaches try to measure the correctness.

Deletion Game: Starting with the original image, in each step a fixed percentage of pixels are deleted (turned black). To choose which pixels to delete, they are sorted from high to low importance. The importance is given by the saliency map. After each step, the modified image is fed to the model and the resulting predictions are being tracked. The intuition behind this approach is that the probability of the original prediction should rapidly drop if the saliency map actually highlights the most important pixels.

Insertion Game: Starting with a blurred version of the original image, in each step, a fixed number of pixels are revealed. Just like in the deletion game the pixels are revealed from high to low importance. In this case, the intuition is that the original prediction should rapidly rise if the saliency map highlights the correct pixels.

A noticeable fact about the deletion game is that it works very similarly to the way the explanations are created. In both cases, the image is occluded, to analyze the resulting predictions. To avoid conflicts that could originate from the similarity, this thesis will focus on the results of the insertion game, which can be seen in figure 6.6.

Occlusion Sensitivity: The graph rises quickly for all explanations except the third one, where it steadily climbs.

RISE: The graph rises quickly for all four explanations. The first and third graphs slightly fall in the end.

Noise Sensitivity: The graph quickly rises for the second and fourth explanation. The graph for the first explanation has a jump in the beginning after which it goes slightly down and then increases for the remaining time. The third graph steadily increases to the middle point, where it stagnates and even goes slightly down.

LIME: The first graph goes up fast and then steadily down until the end where it rises once more. The second graph increases very fast and then keeps a high level. The third graph stays very low for half the time and increases at the midway point. After reaching a high point the graph steeply falls before the end. The last graph has a steep increase in the beginning and stays at that height for the rest of the time.

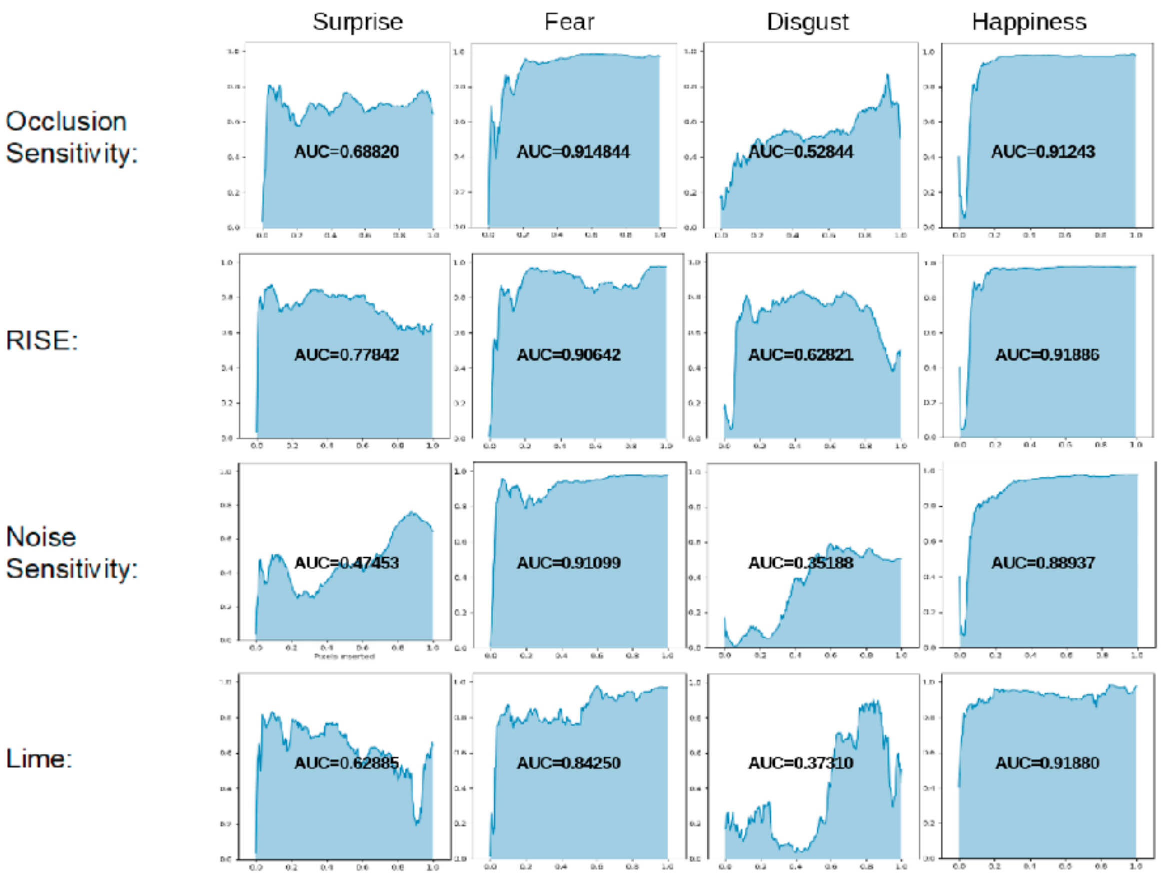
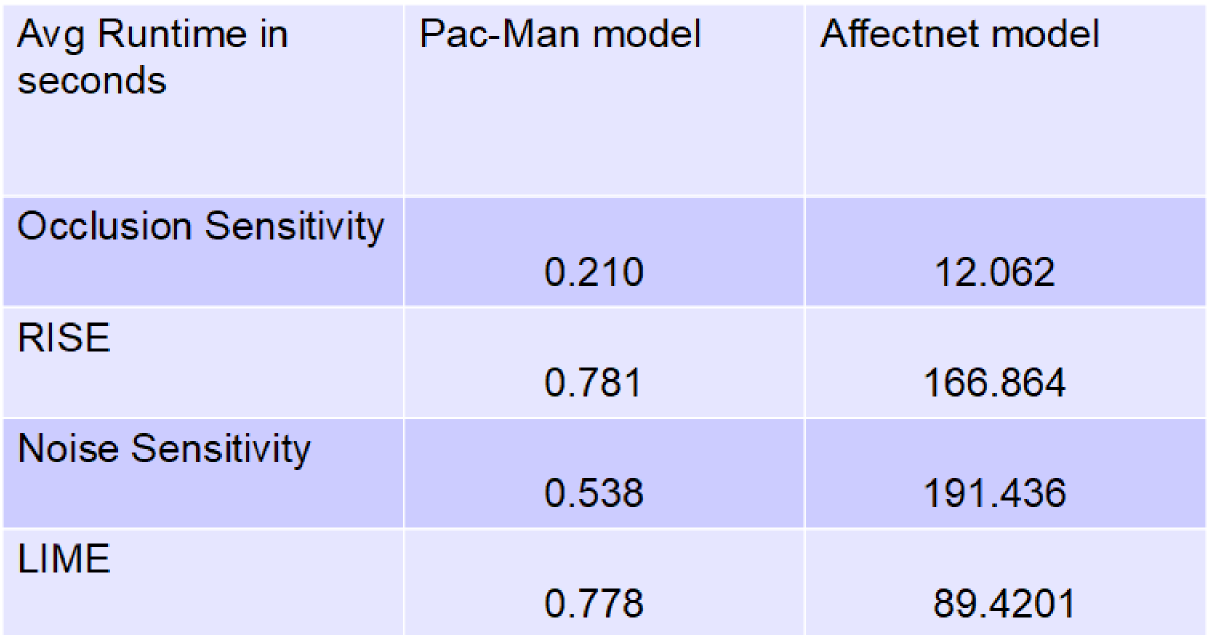


Figure 6.6: Results of the insertion game for the different explanation approaches. The graphs are in the same order as the explanations. Emotions from left to right: surprise, fear, disgust, happiness. The x-axis depicts the percentage of deleted pixels and the y-axis represents the prediction of the original class.

6.6 Runtime Analysis

Another criterion to factor in when comparing the different approaches is their runtime. Depending on time constraints and the size of the dataset which is to be explained, algorithms that take too long might not be feasible. The runtimes of all four approaches were computed for both models using the timeit python library, by measuring the average time it took to create an explanation. The average was computed by explaining different inputs, measuring the time each explanation took, and calculating an average for each approach. This was separately done for both models. In the case of the Pac-Man model, multiple rounds of the game were simulated, explaining 1000 frames in total. In the case of the affectnet model, 9 different images with facial expressions were explained. The resulting times can be seen in figure 6.7. The Occlusion Sensitivity approach is the fastest for both models. In both cases, it massively outperforms the other algorithms, which makes sense because it is the most simplistic approach in terms of occlusion and calculation of the explanations. RISE and Noise Sensitivity both perform way worse then occlusion sensitivity. LIME is somewhere in the middle. The three approaches that perform worse then the Occlusion Sensitivity approach all have more sophisticated methods when it comes to occluding the image and calculating the final explanations, which explains the drop in runtime.



Avg Runtime in seconds	Pac-Man model	Affectnet model
Occlusion Sensitivity	0.210	12.062
RISE	0.781	166.864
Noise Sensitivity	0.538	191.436
LIME	0.778	89.4201

Figure 6.7: The average number of seconds it took an occlusion based approach to generate one explanation. The average was computed across 1000 frames for the Pac-Man model and 9 facial images for the affectnet model.

7 Discussion

The goal of this chapter is to discuss the explanations produced by the different occlusion based explanation methods. To do this, both the explanations and results of the quality metrics presented in section 6.4 and 6.5 are going to be used.

7.1 Discussion of the Explanations

7.1.1 Reinforcement Learning Network on Atari Games

The first set of explanations were generated for the reinforcement learning algorithm which learned to play Pac-Man, as described in section 5.1. Upon visual assessment of the different explanations, which can be seen in figure 6.1, it seems like all approaches produce similar results. Generally, all clearly focus foremost on Pac-Man. This seems rather logical because he is the most important aspect of the game. Besides that, there seems to be a focus on ghosts which are close to the player, as can be seen in state 1 and 4. This also makes perfect sense, since ghosts are the only way Pac-Man can die and therefore are dangerous especially if they are close. Overall, all approaches appear to produce reasonable results, highlighting areas that make sense from a human perspective in the context of Pac-Man. One noteworthy downside of LIME is that it only shows the most important superpixels, but does not produce a saliency map. This makes it impossible to exactly pinpoint what regions inside the superpixels are the most relevant and how different superpixels compare to each other.
One notable finding during the implementation of the algorithms for the Pac-Man model was the property of domain agnosticism. Domain agnosticism refers to whether an explanation approach has to be adjusted depending on the characteristics of different kinds of inputs. As discussed in section 5.3 this property is not given for occlusion based approaches. In fact, the way an algorithm occludes an image has a major impact on the outcome and might need adjustment, in order to generate useful results.

7.1.2 Emotion Recognition Network (Affectnet)

When assessing the different explanations it is noticeable that the results are similar in some cases, but differ in others. For instance, all four approaches clearly highlight the wide-opened eye in the second image. In all other results, there is usually at least one explanation that differs from the rest. In the first image, the Noise Sensitivity approach highlights the nose and the space around the mouth, while the others clearly highlight the wide-opened mouth. On the fourth image, the Occlusion Sensitivity approach highlights the area above the mouth and around the left eye, while all other approaches focus on the corners of the mouth. On the third image, the Noise Sensitivity approach also diverges from the rest, highlighting the area between the eyes. Overall it seems like the results produced by RISE are the best. The highlight spots always seem to be on the correct spots. The only downside of this approach is that there is a lot of noise across the explanation. LIME generally also produces decent results, but the same problem as in the case of the Pac-Man explanations remains. Since the explanations only highlight the superpixels which are the most important but do not show their order, it is hard to compare them to each other. The other two approaches seem to agree with the highlighted spots of RISE and LIME in some cases but differ in others. Especially the Noise Sensitivity approach clearly highlights different areas in the first and third explanation.

7.2 Discussion of the Sanity Checks

When visually assessing the randomized saliency maps, which can be seen in 6.4, it appears like in most explanations random areas are highlighted, which is the expected behavior. The only real outlier seems to be LIME, were the first explanation is identical to the original one. These observations are confirmed in the resulting metrics, which can be seen in figure 6.5. They were obtained by computing the correlations between the explanations of the randomized models and the original ones, as described in section 6.4. When comparing the results of the different approaches there seems to be little to no correlation for the RISE and Occlusion Sensitivity approach. When observing the results for the Noise Sensitivity approach it seems like the Spearman rank is a bit higher than usual, but both the Pearson and SSIM metric indicate a very low correlation. The only real outlier seems to be LIME. All metrics appear to be a bit higher than the average, especially SSIM. This indicates that there is at least a bit of parameter independence in the LIME approach. This result has to be taken with a grain of salt though. As already mentioned in section 6.1 LIME is the only of the four approaches, which does not produce a saliency map as the explanation. Instead, a vector is generated, which indicates whether a superpixel was important to decision or not and ranks them in order of importance. The sanity check was therefore calculated on input images where only the superpixels with the highest importance were visible. While the randomization of the network should also randomize the importance of the superpixels and thus which superpixels are shown, this still means that a majority of every explanation will be black. This might be a reason why the SSIM is as high as it is.
Overall, RISE seems to perform the best in the context of the parameter randomization test, followed by the Occlusion Sensitivity and Noise Sensitivity approach. As already mentioned LIME seems to perform the worst, especially the SSIM metric indicates that there are some problems when it comes to parameter dependence.

7.3 Discussion of the Insertion Game

As described in section 6.5 the Insertion Game offers a way to evaluate the correctness of an explanation. The results of the insertion game for the affectnet explanations can be seen in figure 6.6. These results seem to very much correlate with the insights of the visual assessment. RISE seems to perform the best. All graphs rise really fast after just a small percentage of revealed pixels. The Noise Sensitivity graphs perform well for the explanations which are similar to the explanations of the other approaches (Image 2 and 4) and bad for the explanations which are different (Image 1 and 3). Similarly, the first and second graph of the Occlusion Sensitivity approach rises quickly and the third one not so much. The only surprise is that while the explanation for the fourth image seems to differ from the ones of the other approaches, it seems to perform well in the insertion game. Just like in the sanity checks, LIME does not seem to perform that well. Especially when compared to the Occlusion Sensitivity approach and RISE, the graphs seem to indicate worse performance. As with the sanity checks, the results have to be taken with a grain of salt again. Because the LIME approach does not produce a saliency map there is no clear order in which to insert the pixels. Instead of sorting the saliency map from important to unimportant, the ranking of the superpixels is used to decide which pixels to insert first. After inserting pixels from the most important superpixels, random ones are chosen.

7.4 Concluding the Discussion

In summary, RISE seems to perform the best across visual assessment, sanity checks, and the insertion game. The highlights generally seem to be on the right spot. The only real issue with this approach is that the explanations do not seem to be very selective. This means that there is a lot of noise around the saliency map, which should be ignored. For an inexperienced user, this might be a bit confusing.
The Occlusion Sensitivity approach also performs well across all quality measurements. Especially for the Pac-Man model, it produced very clear explanations. The parameter randomization test did imply parameter dependence and the explanations also performed well in the insertion game. Another big upside is the quickness of the algorithm. When compared to the other approaches it had by far the lowest runtime. The only downside of the approach is that it had to be adjusted for the Pac-Man domain and that the explanations sometimes deviate a little bit from the results of the other approaches.
The Noise Sensitivity approach performed a bit worse than the previous two approaches. For the Pac-Man model, it had to be adjusted first, which meant major changes to the functionality. After the adjustments the results were quite good, producing similarly clear results as the Occlusion Sensitivity approach. The explanations for the affectnet model, using the original approach of occluding the image by blurring regions, were a bit more hit or miss. Some explanations appeared to be correct while others were not. This can also be seen in the results of the insertion game, were some explanations performed better than others. This means that the approach has the potential to produce very

clear and good results but sometimes does not. It also might need some fine-tuning and adjustments to produce valuable results.
LIME seemed fine upon visual assessment but performed the worst in both the sanity check and the insertion game. Because LIME does not produce saliency maps as output it was hard to apply the quality measures to them. This could mean that the quality measurements indicate a worse performance then it is actually the case. The shortcomings of the actual visual representation of the explanation and the problems in terms of testing the quality of them, make LIME the weakest of the four approaches. While it still produces usable results that generally highlighted the correct areas, one has to be cautious of these problems. One upside of LIME is its ability to not only visualize which pixels affected the decision of the model positively but also the ones which had a negative impact on the classification.

8 Conclusion

This thesis discussed four different occlusion based explanation approaches, which all try to offer insights into the decisions of neural networks. Because the network itself does not have to be modified when implementing the algorithms they offer a straight forward way of better understanding the model they are used for and offer a useful first insight into what information is factored in, when the model makes a decision. They therefore can be useful to developers and end-users alike, by helping to shed light in the black boxes that neural networks tend to be. Besides making neural networks more trustworthy, these approaches can also be useful in finding problems like overfitting.
After implementing and deploying the explanation approaches, explanations were generated for two models. A reinforcement model described in section 5.1 and a emotional recognition network described in section 5.2. The resulting explanations can be seen in figure 6.1 and 6.3. Upon first inspection, both the Occlusion Sensitivity and Noise Sensitivity approach seemed to perform very poorly. The problems and fixes were discussed in section 5.3. As it turns out the way the images are occluded has a major impact on the quality of the results. This leads to the first finding of this thesis. While the occlusion based approaches are model agnostic, they are not domain agnostic. What this means is that the model itself does not have to be changed to make the explanation approaches work, but some explanation approaches might need adjustments depending on the input of the model. While a certain algorithm might work for a color image right away, this might not be the case for a grayscale image, as in the case of the Noise Sensitivity and Occlusion Sensitivity approach. This also implies that the way an algorithm occludes the image has a major impact on the quality of the result.
After adjusting the algorithms, all approaches seem to perform reasonably well, especially in the Pac-Man case. When looking at the explanations for the emotional recognition model it seems like they also produce useful results, but the explanations occasionally differ between the approaches. This introduces a potential problem, in the sense that it is not clear, which explanation to choose when they slightly differ.
One idea to tackle this problem are sanity checks, in particular, the parameter randomization test, discussed in section 6.4. The idea behind this test is to produce metrics measuring the correlation between explanations and the model parameters. Because the explanations try to explain the decision-making process of the model a high correlation would be desirable. The results of this test can be seen in figure 6.5. All approaches perform reasonably well, with RISE performing the best and LIME the worst. These results confirm the first impression of the visual assessment of the explanations.
While the model randomization test does measure an important aspect of the quality of an explanation, it does not make a statement about the correctness of it. It implies whether the model parameters were used to create the explanation, but not if the actual

correct pixels were highlighted. To assess this problem the insertion game was introduced in section 6.5. The results of the insertion game can be seen in figure 6.6 and substantiate the findings of the visual assessment. Again RISE seems to perform the best, followed by the Occlusion Sensitivity. The Noise Sensitivity approach and LIME both perform worse than the previous two.
Overall, all occlusion based approaches do a decent job explaining the decisions of neural networks. Especially their property of model agnosticism makes them extremely valuable. RISE seems to perform the best across the board. The only downside of the approach is that the explanations tend to have a lot of noise and that runtime tends to be relatively high. The Occlusion Sensitivity approach is the simplest of the four approaches and usually performs reasonably well. Especially in the case of the Pac-Man model the explanations were particularly clear. The simplicity of the approach also leads to a particularly low runtime. The Noise Sensitivity approach produced good and clear results for the Pac-Man model, but a bit more hit or miss explanations for the affectnet model. The LIME approach produced decent results upon visual assessment, but the format of the results made it hard to evaluate them with the quality metrics. This also might be the reason for the poor results of the sanity checks.
In conclusion, RISE seems to be for experienced users who look for the most correct results. The noise in the explanations is something the user has to ignore. Filtering out this noise would be a major improvement to make the algorithm even better. The Occlusion Sensitivity approach is advised for users who look for a simple and clear explanation approach. The user has to be aware though that the occasional explanation might not be a hundred percent correct. The Noise Sensitivity can be useful but tends to perform equally good or worse than the Occlusion Sensitivity approach while having a more complex approach and a longer runtime. Because the LIME approach performed poorly on the parameter randomization test and the problems with the representation of the results it is hard to advise it to anyone. It can produce useful results, but the user has to be aware of the shortcomings.

Bibliography

[AGM$^+$18] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515, 2018.

[ASWDF16] Yannis M Assael, Brendan Shillingford, Shimon Whiteson, and Nando De Freitas. Lipnet: End-to-end sentence-level lipreading. *arXiv preprint arXiv:1611.01599*, 2016.

[BXS$^+$20] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 648–657, 2020.

[CPC19] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[DHK$^+$17] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GF17] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a “right to explanation”. *AI magazine*, 38(3):50–57, 2017.

[GKDF18] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801, 2018.

[Hol18] Andreas Holzinger. Explainable ai (ex-ai). *Informatik-Spektrum*, 41(2):138–143, 2018.

[HWAA20] Tobias Huber, Katharina Weitz, Elisabeth André, and Ofra Amir. Local and global explanations of agent behavior: integrating strategy summaries with saliency maps. *arXiv preprint arXiv:2005.08874*, 2020.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[LWY+16] Chang-Shing Lee, Mei-Hui Wang, Shi-Jim Yen, Ting-Han Wei, I-Chen Wu, Ping-Chiang Chou, Chun-Hsun Chou, Ming-Wan Wang, and Tai-Hsiung Yan. Human vs. computer go: Review and prospect [discussion forum]. *IEEE Computational intelligence magazine*, 11(3):67–72, 2016.

[MHM17] Ali Mollahosseini, Behzad Hasani, and Mohammad H Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 10(1):18–31, 2017.

[ON15] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[PDS18] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[PVZ15] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In Xianghua Xie, Mark W. Jones, and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, pages 41.1–41.12. BMVA Press, 2015.

[RHR+18] Elisabeth Rumetshofer, Markus Hofmarcher, Clemens Röhrl, Sepp Hochreiter, and Günter Klambauer. Human-level protein localization with convolutional neural networks. In *International Conference on Learning Representations*, 2018.

[RSG16a] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Local interpretable model-agnostic explanations (lime): An introduction. `https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/`, 2016. [Online; accessed 2-September-2020].

[RSG16b] 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*, pages 1135–1144, 2016.

[SB18] Andrew D Selbst and Solon Barocas. The intuitive appeal of explainable machines. *Fordham L. Rev.*, 87:1085, 2018.

[TG19] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Towards medical xai. *arXiv preprint arXiv:1907.07374*, 2019.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[Wik20] Wikipedia contributors. Convolution — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Convolution&oldid=970626979`, 2020. [Online; accessed 4-August-2020].

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.