Albstadt Sigmaringen University

Evaluating and Establishing Adversarial Attacks on Neural Networks based on Explainable AI Algorithms

Robin Schmitt

(Mat.-Nr. 88353)

Examiner: Prof. Dr. Andreas Knoblauch

Co-Examiner: Prof. Dr. Haeberlein

03/2022

Statement of Authorship

I, Robin Schmitt, hereby declare that this thesis "Evaluating and Establishing Adversarial Attacks on Neural Networks based on Explainable AI Algorithms" represents my own written work and is based on my own research. Furthermore I declare, not to have used any sources nor aids other than those indicated. All passages quoted or paraphrased from publications are properly attributed and cited. This work was not submitted to another examination board, neither in a substantially similar version, nor partially, and was not published elsewhere. I affirm, that the electronic version of this thesis is identical to the printed ones.

Robin Schmitt

Contents

List of Figures

List of Tables

Abstract

Machine learning (ML) has been successfully adopted into products used in everyday life, raising the question how reliable these models are in the face of adversaries. Explainable artificial intelligence (explainable AI) methods seek to provide insights into the decision-making process of non-linear classifiers like neural networks. We hypothesize that applying explainable AI also unveils vulnerabilities of ML models. To this end, we propose two novel white-box adversarial attacks by perturbing input data. Our novel attacks LRP-grad and LRP-mean are based on a custom implementation of Layerwise Relevance Propagation (LRP). We qualitatively compare our attacks to the state-of-the-art Iterative Fast Gradient Sign Method (IFGSM) and LRP-flip assuming worst-case perturbations. Next, we quantify the performance benefits of our attacks compared to IFGSM and LRP-flip using the MNIST, CIFAR10, CALTECH256 and dog vs. cat datasets with convolutional neural networks trained for each dataset. Our analysis shows that our LRP-grad attack reaches the same attack success rate (ASR) as IFGSM at up to 60.4% lower L_1-norm perturbations compared to IFGSM on CALTECH256. Furthermore, the novel LRP-mean attack overcomes vanishing gradients, an issue that renders gradient-based methods such as IFGSM ineffective. Our findings link the fields of explainable AI and adversarial attacks, contributing to more robust image recognition models and highlighting the interplay of explanations and attack surfaces.

1 Introduction

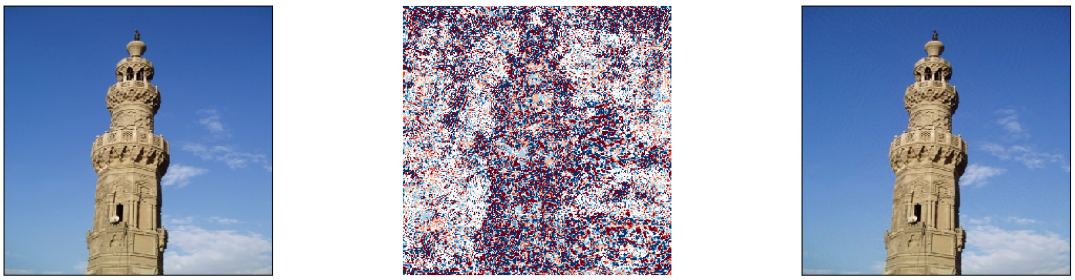
1.1 Motivation

A survey on information security by Ernst & Young Global in 2020 showed that 65% of companies only take cyber security into account after an incident has already happened [49]. Based on this insight they propose the method security-by-design to ensure that preventive measures against attacks are implemented during development of products.

Due to the increased usage of artificial intelligence methods in software applications, regulators are increasingly paying attention to the implications of artificial intelligence. The Centre for European Policy Studies (CEPS) for example launched a task force in 2019 to analyze the interplay between cybersecurity and artificial intelligence [48]. In their report, they identify that machine learning-based systems share the vulnerabilities of any software product based on bugs introduced during development but also have additional attack surfaces for which novel attack and defence methods are employed. One famous attack method specific to neural networks is the Fast Gradient Sign Method (FGSM) that is capable of manipulating an image to a degree that no human can discern, but enough so that the neural network misclassifies it, as is demonstrated in Figure 1.1. To effectively implement a security-by-design approach, developers of systems based on artificial intelligence need knowledge about the specific attack surfaces as well as how increase the robustness of their products against such attacks.

Increasing attention is also drawn to explainable artificial intelligence (explainable AI), a field which aims to provide insights into individual decisions made by artificial intelligence systems [54]. Research in this field has risen significantly within recent years [13] and regulators are increasingly exploring explainable AI with regards to policies, most notably for example in the context of the General Data Protection Regulation (GDPR). In GDPR article 5, transparency is mentioned as one of the key principles under which data has to be handled by companies [73]. Even though this does currently not encompass details regarding classifications in diverse systems such as credit fraud detection, current reports on explainable AI by the European Union explore this [5]. Article 22 GDPR states the right of a data subject to contest a decision made solely by automated decision making [73]. Should such a decision be contested, a reconstruction of the reasoning behind the automated decision making might be necessary and could be enabled by explainable AI methods.

As both of these fields are gaining importance for researchers and developers in artificial intelligence, attacks on neural networks utilizing explainable AI are surfacing [34]. Considering that neural networks achieved above-human performance on bench-



(a) Sample x of class 142 Confidence 0.9996

(b) Perturbation generated by FGSM

(c) Perturbed $\tilde{x}$ of class 142 Confidence $4 \cdot 10^{-9}$

Figure 1.1: The commonly used adversarial attack Fast Gradient Sign Method was applied to an image of the CALTECH256 dataset. After the perturbation, the image is misclassified by the classifier.

marks like image recognition [28], it is fascinating that they are susceptible to seemingly small perturbations. This is why combining adversarial attacks and explainable AI is so intriguing, because building adversarial attacks on explainable AI allows an intuitive understanding of the attacks.

1.2 Objectives

We hypothesize that a method that aims to explain single decisions by neural networks in the input dimensions can be utilized to create enhanced adversarial attacks. Our goal is to utilize the relevance score per input dimension that is provided by such methods to choose the perturbation targets. Based on this objective, the method layerwise relevance propagation (LRP), which was proposed as an explainable AI algorithm by Lapuschkin et al. [42], is identified due to its deterministic property and concise mathematical foundation.

With the goal to propose an adversarial attack method that outperforms state-of-the-art methods, we investigate utilizing LRP to compute adversarial attacks on image recognition neural networks. Within this scope, different approaches to generate perturbations are analysed and defined to conduct benchmarks and identify the best performing variant. Since full control of designing the adversarial attacks is necessary, a custom implementation of LRP besides the LRP-toolbox by Lapuschkin et al. [42] is built.

Once new methods for adversarial attacks have been described, a quantitative analysis of those attacks is conducted, first assuming worst case conditions in a theoretical setting and second experimental using four datasets. Within this analysis, a state-of-the-art method for adversarial attacks on neural networks, the Fast Gradient Sign Method [23] will be used as benchmark. To benchmark the novel attack methods, for each image recognition dataset a convolutional neural network is to be trained. Also, a benchmarking setup has to be defined to enable a quantitative comparison of the attack's

effectiveness with the perturbation that is introduced into the original data. Finally, the robustness of a novel adversarial attack method against adversarial transfer learning should be evaluated.

While a pure mathematical quantitative analysis of the novel attack methods is conducted, the detectability of the adversarial attacks by humans is not evaluated. We aim to describe the proposed attack methods in a way that allows further work to apply other explainable AI methods to our attack algorithms by just adapting an adversarial pattern, making our attack methods reusable. By concentrating on the field of image recognition and analysing the performance of the attack methods on four datasets, our goal is to understand the effectiveness of our attack methods in one application area to an extensive degree, allowing further work to investigate the application of these attack methods in other application areas like natural language processing.

1.3 Contribution

1. **Custom implementation of layerwise relevance propagation and adversarial attacks.** We built a custom implementation of layerwise relevance propagation that is compatible with TensorFlow and which can be extended to be machine learning-framework agnostic. The explainer object in this implementation enables calculation of the relevance matrices as well as creating adversarial attacks. (cf. Chapter 2)
2. **Proposal of two novel adversarial attack methods.** Two novel attack methods, LRP-grad and LRP-mean, that utilize layerwise relevance propagation to identify a subset of input dimensions to perturb are proposed, quantitatively analysed and compared to the state-of-the-art method iterative FGSM as well as LRP-flip. (cf. Chapters 3, 4)
3. **Quantitative benchmark of IFGSM, LRP-flip, LRP-grad and LRP-mean.** To evaluate the quantitative performance of IFGSM, LRP-flip, LRP-grad and LRP-mean, a benchmark is conducted on the datasets MNIST, CIFAR10, CALTECH256 and dog vs. cat, for which the L_1, L_2 and L_∞-norm over attack success rate is evaluated. In this benchmark, LRP-grad achieves the same ASR as IFGSM at an up to 60.5% percent lower L_1-norm perturbation on CALTECH256. Additionally, we found that the LRP-mean method can circumvent vanishing gradients during adversarial attacks, allowing it to achieve success where gradient-based attack methods fail. (cf. Chapter 4)

2 Background

2.1 Neural Networks

This chapter aims to lay the theoretical foundation and establish the state of the art regarding neural networks as classifiers, attack and defence methods, as well as how robustness of neural networks against attacks is measured.

2.1.1 Application Areas

Artificial neural networks can be described as models to detect statistical patterns in data [10]. Research on artificial neural networks originated from attempting to mathematically represent how biological systems interpret information with publications dating as early as 1943 [10]. Widely regarded as groundlaying work on neural networks is the publication on perceptrons by Rosenblatt in 1958 [63], in which Rosenblatt utilized single-layer perceptrons and made the prediction that this type of program will at some point be able to perform tasks such as language translation [63]. Increasing computing power and breakthroughs such as the invention of the backpropagation algorithm [15] led to a point where today neural networks are utilized widely across all industrial domains as well as in research and medical applications [1].

What makes neural networks so universally applicable is their ability to approximate any function given enough training iterations and training data [10]. In recent years the collection of data has exponentially increased and widened across many fields. One example for this trend is the collection of user data by tech companies such as Google [65]. An analysis on the collection of personal data by Google in 2018 estimates that Google is able to collect about 350 MB of user data from a connected android smartphone per month [65]. While it seems trivial that a smartphone that is uniquely tied to a Google account enables companies to collect user data, projects such as the website amiunique.org [41] aim to raise awareness that just accessing websites via a browser leaves an almost unique fingerprint which can be tracked by companies and subsequently used for ad-targeting or other purposes. With all this data, machine learning applications can be used to derive patterns concerning single individuals, groups as well as metainformation. An overview of some areas and the according application of neural networks is given below:

- Finance: Credit card fraud detection [22], customer churn prediction [37]
- Meteorology: Weather forecasting, air temperature forecasting [20]
- Medicine: Pandemic modelling [72], cancer detection in mammograms [27]

- Energy: Energy demand forecast [59]
- Sales and marketing: Sales forecast [80], customer behaviour prediction [81]

While this list is by far not complete, it aims to give an impression of the diverse scenarios in which neural networks achieve state of the art performance and often perform better than alternative methods such as support vector machines or linear models.

2.1.2 Mathematical Foundations

In general, neural networks model a function that maps the input x to an output $y = f(x)$. More precisely, a two-layer neural network performs a series of linear combinations with non-linear activations in order to calculate $f(x)$. The following notations and equations are based on Bishop et al. [10]. Based on Figure 2.1, the following definitions are made:

- x Neural network input
- x_j Input regarding dimension j
- l Layer of the neural network
- $w_{ji}^{(l)}$ Weight connecting neuron j in layer $l+1$ with neuron i in layer l
- $a_j^{(l)}$ Activation of neuron j in layer l
- y Neural network output
- y_k Output regarding dimension k
- $h()$ A non-linear activation function (such as ReLU or sigmoid)

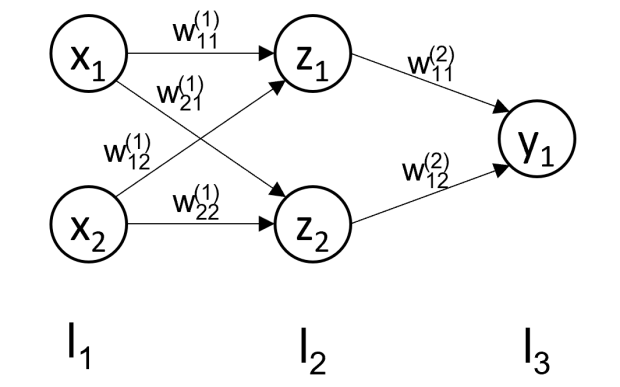


Figure 2.1: A feed-forward network with the input layer l_1, a hidden layer l_2 and an output layer l_3, based on [10].

Based on these definitions, the activation $a_1^{(2)}$ is calculated as shown in the following equation:

$$a_1^{(2)} = x_1 \cdot w_{11}^{(1)} + x_2 \cdot w_{12}^{(1)} \tag{2.1}$$

Generalized and taking the bias unit w_{j0} into account, $a_j^{(l+1)}$ is calculated accordingly over the input dimensions $x_1 \dots x_D$:

$$a_j^{(l+1)} = \sum_{i=1}^{D} (x_i^{(l)} \cdot w_{ji}^{(l)}) + w_{j0}^{(l)} \tag{2.2}$$

The final step is applying a non-linear activation function in order to calculate z:

$$z_j^{(l)} = h(a_j^{(l)}) \tag{2.3}$$

Combining these two steps, a generalized formulation to calculate a_j is:

$$a_j = \sum_i w_{ji} z_i \tag{2.4}$$

The output of the example in Figure 2.1 is calculated as follows. In case of a binary classifier, the sigmoid function $\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$ is applied to the output y_1 afterwards:

$$y_1 = h(x_1 \cdot w_{11}^{(1)} + x_2 \cdot w_{12}^{(1)}) \cdot w_{11}^{(2)} + h(x_1 \cdot w_{21}^{(1)} + x_2 \cdot w_{22}^{(1)}) \cdot w_{12}^{(2)} \tag{2.5}$$

$$f(\mathbf{x}) = \text{sigmoid}(y_1) \tag{2.6}$$

The definitions in equations 2.1 to 2.6 are applicable for dense layers such as the ones depicted in Figure 2.1. Together, the function depending on the input $\mathbf{x}$ and the weights and biases $\mathbf{w}$ can be defined as follows:

$$y_k(\mathbf{x}, \mathbf{w}) = h(\sum_{j=0}^{M} w_{kj}^2 h(\sum_{i=0}^{D} w_{ij}^1 x_i)) \tag{2.7}$$

While equation 2.7 describes a fully connected neural network layer, as the research in neural networks has grown, other layer types have been introduced with special properties in mind. One of these layer types are convolutional layers, which are widely used in image or video recognition tasks in Convolutional Neural Networks (CNNs) [25]. For a good performance in image recognition tasks, invariance such as translation or scaling should not have an impact on the classifier's performance in order to achieve a good generalization. To also increase robustness against invariance in inputs, convolutional layers analyze only a subsection of the input image at a time and calculate their output based on the according feature map [10]. Thus the locality of information that is especially important in image recognition is taken into account. Additionally, convolutional layers do not increase the parameter count of neural networks considerably due to weight sharing [10]. The number of added parameters is dependent on the kernel size and the number of channels / convolutions that are applied.

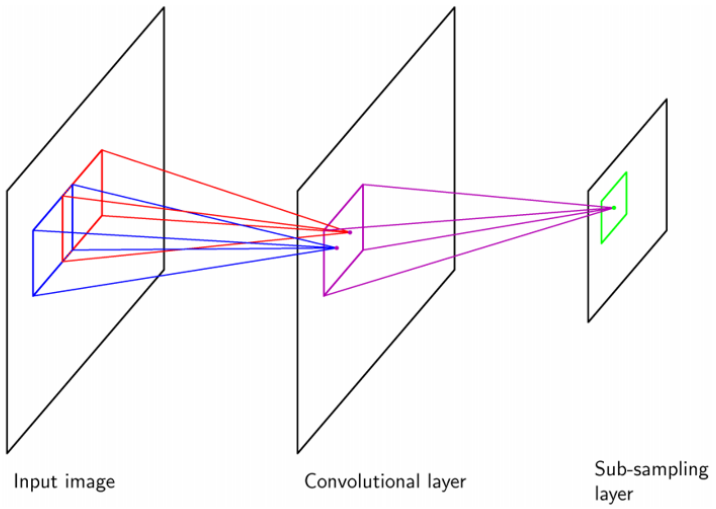


Figure 2.2: Illustration of a convolution layer with successive subsampling, taken from [10].

As depicted in Figure 2.2, convolutional layers are often combined with sub-sampling layers. Sub-sampling layers such as max-pool or average-pool layers subsample an area by applying a function such as computing the maximum of all values in that area and outputting only this result. Thus they add another level of robustness against small shifts in images and additionally reduce the feature dimensions [10]. This means that in max pooling, the maximum value in the kernel is propagated to the next layer, while the other values are discarded. In average pooling, the average value of all values in the kernel is propagated.

2.1.3 Learning Model Parameters using Backpropagation

As described above, neural networks can consist of multiple layers (e.g., convolutional layers followed by fully connected layers) of varying architecture that are accompanied by weight and bias matrices $\mathbf{w}$. These matrices are the parameters Θ of the model that are optimized in such a way that the error or cost function is minimized. Calculating the error E for a concrete set of inputs x_n with $n = 1, \ldots, N$ and the according target vectors t_n, a sum of squares error function is calculated as follows [10]:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 \tag{2.8}$$

In order to minimize $E(w)$, the weight vector w must be adapted by introducing a change in the vector $w = w + \delta w$ [10], for which the information about the gradient $\nabla E(w)$ is required. However for neural networks the change in the error function $E(w)$ is too complex to be derived analytically due to the nonlinear properties of neural networks

[10]. Thus the numerical algorithm backpropagation was introduced [15] with which incremental changes $\triangle w^{\tau}$ per iteration τ of the weight vector can be calculated. Utilizing the gradient information, an update of w per step τ with the learning rate η is defined as [10]:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E\left(w^{(\tau)}\right) \tag{2.9}$$

Deriving the error function described in equation 2.8 for one input n with regard to a weight w_{ji} in a simple linear model $y_k = \sum w_{ki} x_i$ leads to the error function and derivative [10]:

$$\frac{\delta E_n}{\delta w_{ji}} = (y_{nj} - t_{nj})\, x_{ni} \tag{2.10}$$

Applying this to neural networks, based on equations 2.4 and 2.3, equation 2.10 for any activation a can be written as [10]:

$$\frac{\delta E_n}{\delta w_{ji}} = \frac{\delta E_n}{\delta a_{ji}} \frac{\delta a_{ji}}{\delta w_{ji}} \tag{2.11}$$

With the definition that $\delta_j = \frac{\delta E_n}{\delta a_j}$ and following from 2.4 that $\frac{\delta a_j}{\delta w_{ji}} = z_i$, equation 2.11 can be written as [10]:

$$\frac{\delta E_n}{\delta w_{ji}} = \delta_j z_i \tag{2.12}$$

It can be concluded that in order to derive $E(w)$, the term δ_j has to be calculated for every hidden- and output-unit in the neural network. Considering h' as the derivative of the activation function, the backpropagation through the neural network is then calculated by [10]:

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{2.13}$$

2.2 Explainable Artificial Intelligence

Due to the complexity of neural networks and their increasing depth [18], the decisions of neural networks that are for example used as classifiers are not easily comprehensible for humans. The decisions of machine learning models that are seemingly simpler, such as decision trees, on the other hand are retraceable due to their easily comprehensible nature. Explaining the decision of a simple decision tree is as easy as following the tree along all nodes and understanding the input dimension that was evaluated at that node. While these models are also widely used and generally a powerful alternative to neural networks [4], they are limited and recent progress on tasks such as image recognition were achieved by applying neural networks [16]. However, when deploying an intelligent system that utilizes neural networks in the field, decision makers and those who approve the use of intelligent systems have to be able to decide whether they are convinced of the performance and ability of the intelligent system. In this context, neural networks were often called black box classifiers, symbolizing the inability to understand the internal workings and only being able to observe the output of the classifier [42].

There are numerous benefits of being able to understand decisions made by complex neural networks. Firstly during development, when explanations help debugging the classifier and releasing it [78], and secondly during production and increasing customers' trust by explaining recommendations [36]. According to [36], the main benefits of explanations for end-users are:

- Justification of a recommendation
- Involvement of the user
- Education on the systems reasoning and limits
- Acceptance

Additionally, incidents such as accidents during testing of autonomous vehicles are already strongly analysed by the respective regulators. The report of such an incident HWY18MH010 in the United States of America dedicated a whole paragraph describing the classifier's predictive behaviour [11]. The interest in explainable artificial intelligence has recently gained momentum, with governmental organisations such as DARPA [26] and the European Union [5] building programs to fund research in this field. Also, an increasing number of companies such as Bosch and Mercedes-Benz also name explainability as a core principle when deploying artificial intelligence [51, 19].

2.2.1 Defining Explanations and their Target Group

The field of explainable artificial intelligence covers a multitude of possible target groups as well as multiple approaches on explanations [64].

Regarding target groups, Samek et al. differentiate between customers or users of intelligent systems and developers of such systems [64]. This distinction is made mostly about the depth of the given explanation, since the target groups possess differing prior knowledge about the intelligent system. While a developer or quality manager of the

intelligent system has in-depth knowledge about the algorithms that are used and thus can interpret complex internal diagnostics, a regular user only requires coarse information about the decision, preferably with regard to the input domain. In the case of the input domain being a sentence and each word being an input dimension, the most influential words should be highlighted. Similarly, in image recognition the most important pixels should be highlighted. Contrary to that, a developer might be interested in the word embeddings in natural language processing or the activation maps of different CNN filters.

Regarding the explanation itself, four different approaches are defined in [64]:

- Learned Representations
- Individual Predictions
- Model Behaviour
- Representative Examples

The research on learned representations in [79] stems from the urge to understand and improve the performance of convolutional neural networks in image recognition. Their concept of explanation is based on the idea that by observing feature map activity in hidden layers of the CNN and maximizing their activation, the learned concepts behind these feature maps can be obtained [79, 82]. This type of explanations produces constructed inputs that maximize a specific set of feature maps in the classifier. Thus, it allows to visualize disentagled representations of the patterns or objects the classifier is capable to identify [82]. These disentangled representations are shown to emerge in convolutional neural networks without it being specifically incentivized to learn them and can thus be used as an argument towards their ability to generalize concepts out of training data [79].

In order to understand an individual prediction on one specific input, the target is to attribute to every input dimension its contribution to the output $f(x)$. In the case of image recognition, this can have the form of heatmaps attributing to every pixel a relevance score [42] or in case of natural language processing highlighting the relevant words in a sentence or the most relevant documents in the corpus [60].

Explanations in the context of model behaviour aim to identify the general strategy of a model in achieving its task. Such strategies can be identified when analysing the individual predictions over a subset of the data and identifying e.g., regions or patterns that are associated with high relevance [44]. Lapuschkin et al. identify so called Clever-Hans predictors in [44] by showing that some classifiers base their decisions on watermarks in pictures. These Clever-Hans predictors generally learn to recognize classes based on patterns that are not subject related which might impede their performance in real-world applications. Model behaviour explanations thus aim to deliver a much broader understanding on how a model behaves and whether a model is actually a Clever-Hans predictor [64].

A different approach to explanations is searching for representative examples of data that fulfil different criteria [64]. In a scenario where an input is misclassified, it can be

of interest to compare that input with high confidence correct classifications. Similarly, comparing low and high confidence classifications might help understand the limits of a model. Koh and Liang define influence functions in [39] to identify the training data that impacts the classification result most. With this approach, an explanation of a model's decision is created through the data that the model was trained on. Khanna et al. use such an approach in [38] to identify mislabelled training samples, clean training data and to distil the training data to only the most influential samples in order to reduce training time while still maintaining high performance.

This work focuses on individual predictions, since the goal of the newly proposed attack method is to attack neural networks by manipulating an input x. Explanations of individual predictions are presented in the same dimensions as the input and thus enable a manipulation of the input features with a high relevance for the output.

2.2.2 Definition of Explainability in the Context of Linear Classifiers

Research about explanations incorporated with human machine interfaces and an underlying theory of explanations was already conducted in 1993 by Johnson and Johnson, addressing how an end-user wants to understand the reason for an action taken by an intelligent system [35]. Further research by Sommer and Paxson has coined the term semantic gap [70]. The paper discusses the output of an anomaly detection system and how it corresponds to intrusion detection. One point raised in this research is that from the perspective of an operator, the interpretation of the anomaly detection defines whether it can be classified as an attack or just not yet observed values [70]. In the context of machine learning models such as neural networks, this semantic gap can be bridged by attributing to every input data point its importance (contribution) to the output [42].

To further discuss the explainability of models, a simple linear classifier will be defined below. Linear classifiers are inherently comprehensible to humans, because their weights directly explain the component-wise contribution to the classification. The exemplary linear classifier is defined as a binary classifier, dividing two classes. The decision threshold between the classes is set as 0. For this linear classifier the definitions are:

- $\mathbf{w} \in R^D$ weight vector
- b bias term
- $\mathbf{x} \in R^D$ input sample vector

Based on these definitions, the linear classifier is described by the following [42]:

$$f(x) = \mathbf{w}^T\mathbf{x} + b = y \tag{2.14}$$

Formula 2.14 can alternatively be written as a sum [42]:

$$f(x) = \sum_{d} w_d \cdot x_d + b \tag{2.15}$$

Considering that any $f(x)$ below 0 corresponds to class 1 and any $f(x)$ above 0 corresponds to class 2, equation 2.15 leads to the conclusion that the contribution (and thus importance to the output) of each x_d is defined by w_d. An input agnostic explanation of this model (and thus the sensitivity of the model to each input dimension) is given by w [42]. The explanation of why a specific sample was classified to a certain class can be split into every input dimension d by computing $w_d \cdot x_d$. In other words, $w_d \cdot x_d$ calculates the contribution of input dimension d to the classification result $f(x)$ [42]. This contribution is positive if the value of the input dimension x_d is an indicator for class 2 and negative, when the value of the input dimension x_d is an indicator for class 1. Since w describes the change of the output with regard to the input, it is the gradient of the output with regard to the input.

2.2.3 Explaining Non-Linear or Blackbox Classifiers

Blackbox classifiers, such as support vector machines or convolutional neural networks, are not inherently interpretable due to their complexity. In order to get an intuitive understanding of these classifiers, multiple approaches to explain predictions or the behaviour of non-linear classifiers have been proposed:

- Surrogate models [21]
- Local approximations based on perturbations [60]
- Gradient-based methods [56]
- Relevance decompositions [42]

The idea behind using surrogate models to explain decisions of non-linear classifiers evolves from the inherent explainability of models such as decision trees or linear classifiers. This concludes that a surrogate model that performs on a similar level has to be trained [9], while under the restriction that this surrogate model must be interpretable and thus explainable. While this process can produce models that are explainable and maintain a satisfying performance on the initial problem [46], benchmarks in for example image classification are dominated by neural networks.

An example for local approximations is the method Local Interpretable Model-agnostic Explanations (LIME) [60]. In this method, for image recognition a matrix of binary superpixels is generated that encodes whether an RGB-pixel is relevant for the classification or not. This is achieved by generating perturbed input data, calculating a similarity score to the original data and calculating a prediction on the perturbed data. Based on the difference in the new prediction result and knowing what parts of the input were perturbed, the importance of the input data to the prediction result is determined. Due to the random generation of perturbations, LIME is of stochastic nature. The

results are additionally based on hyperparameters such as the sample count and perturbation rate. This results in non deterministic results when calculating an explanation for the same data multiple times, which might be an issue depending on the domain in which this method is deployed [47].

Utilizing gradients of the output with regard to the input, Simonyan et al. [67] propose a method to generate saliency maps that show the effect of a change in the input on the prediction output. Similarly, deconvolutions are proposed in [79] to visualize the strongest activations of each feature map in the convolution layers or generate a saliency map based on occluded inputs to identify which regions of the input correspond to which output.

In order to attribute a relevance score to each input feature, and thus understand the contribution of each input feature to the prediction, Bach et al. [42] propose a component-wise relevance decomposition called layerwise relevance propagation (LRP). Based on a set of mathematical rules, a relevance score R is computed for each input feature so that the resulting relevance matrix has the same dimensions as the input data. Compared to local approximations based on perturbations, LRP is deterministic and thus produces repeatable results which is why LRP was chosen as explainable AI method for this work.

2.2.4 Layerwise Relevance Propagation

This section relies on the published work of Lapuschkin et al. who proposed layerwise relevance propagation as an explainable AI algorithm in multiple works [42]. When defining explanations of individual predictions, a relevance or contribution score R is calculated for each input feature x_i [8, 42]. In [42] a mathematical ruleset is defined in order to ensure a correct implementation of LRP. This ruleset is layed out and translated into matrix-based operations in order to efficiently implement it in Python.

With R being the relevance score, the relevance score of component j in layer l is described by R_j^l. Since the relevance scores are a decomposition of the prediction, the first condition is that the sum of all relevances can not be greater than $f(x)$. This applies to all layers in the CNN with the intuition behind it that the model's decision is finally distributed across all input pixels and colour channels which can be derived from equation 2.16. Additionally, this condition describes the conservation of relevance throughout the propagation and is used in the implementation in the form of a check sum to confirm that the relevance is fully distributed. [42]

$$f(x) = \sum_{j \in l+1} R_j^{(l+1)} = \sum_{j \in l} R_j^{(l)} = \cdots = \sum_j R_j^{(1)} \tag{2.16}$$

As second condition, the propagation of relevance scores throughout the layers of the CNN is calculated according to the connections and weights between neurons as described in equation 2.17. This is similar to backpropagation during the training of neural networks and can be understood as messages $R_{i \leftarrow j}^{(l,l+1)}$ that are propagated from the output to the input layer. [42]

$$R_i^{(l)} = \sum_{i:\, i \text{ is input for neuron } j} R_{i \leftarrow j}^{(l,l+1)} \tag{2.17}$$

With equations 2.16 and 2.17, the calculation of a relevance message in a simple multilayer neural network in which neuron i sends the message $x_i w_{ij}$ to neuron j is done as follows [42]:

$$R_{i \leftarrow j}^{(l,l+1)} = R_j^{(l+1)} \frac{x_i w_{ij}}{\sum_i x_i w_{ij}} \tag{2.18}$$

Equation 2.18 assumes the relevance of neuron j in layer $l+1$ was already calculated. The local contribution of neuron i is then weighted by the global contributions of all neurons i in layer l [42]. Combining equations 2.18 and 2.17, R_i^l is calculated [42]:

$$R_i^l = \sum_j R_j^{(l+1)} \frac{x_i w_{ij}}{\sum_i x_i w_{ij}} \tag{2.19}$$

In 2.3 the visualization of a simple network during feed-forward and relevance propagation is depicted and the principle of propagating messages from the output layer all the way to the input layer can be seen. During prediction in the feed-forward the sum of the inputs to neuron 5 is $f(x)$. Thus in the relevance propagation, the initial relevance value is $f(x) = R_5^{(1)}$. This holds true also for multiclass classification, where the relevance score of the output layer in relation to a specific class is set to the label's one-hot encoding.

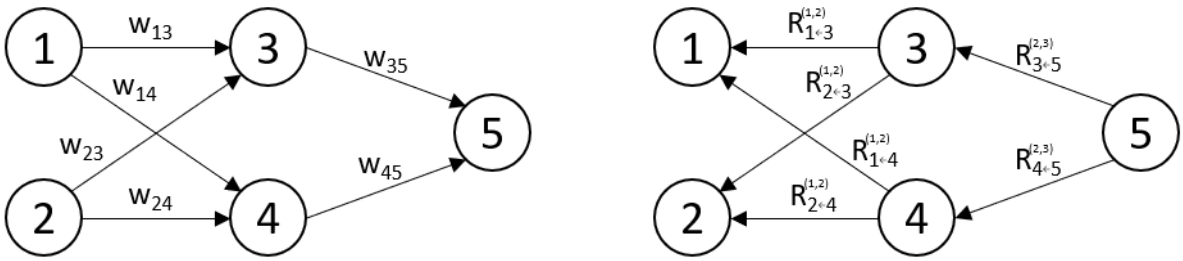


Figure 2.3: A feed-forward network and the back-propagation of Relevance.

As layed above, the implementation must follow equations 2.16 and 2.17 and ideally implement these in an efficient way. Furthermore, different rules for LRP were proposed that satisfy the equations [52]. An excerpt of the rules which were used in this work are:

- LRP-0
- LRP-ϵ
- LRP-ab

Starting with the LRP-0 rule, which is described in equation 2.18, scaling it to a full layer is done utilizing matrix operations. Considering $d1$ as the dimensionality of layer

l, $d2$ as the dimensionality of layer $l+1$ and n as the number of input samples, first the denominator of equation 2.18 is calculated:

$$\mathbf{Z}_{(n,d2)} = \mathbf{input}_{(n,d1)}\mathbf{weights}_{(d1,d2)} \tag{2.20}$$

Secondly, the relevance matrix of layer $l+1$ is normalized by element-wise division by Z.

$$\mathbf{S}_{(n,d2)} = \frac{\mathbf{R}^{l+1}_{(n,d2)}}{\mathbf{Z}_{(n,d2)}} \tag{2.21}$$

Lastly, the relevance matrix of layer l is calculated by multiplying S with the transposed weight matrix and finally by performing an element-wise multiplication with the layer input.

$$\mathbf{R}^{l}_{(n,d1)} = \mathbf{S}_{(n,d2)}\mathbf{weights}^{T}_{(d1,d2)} \cdot \mathbf{input}_{(n,d1)} \tag{2.22}$$

In equation 2.21, it is possible to encounter divisions by zero or close to zero that might lead to numerically unstable results. Thus, in LRP-ϵ a normalization factor ϵ is added when calculating Z, leading to the following equations [42]:

$$R^{l}_{i} = \sum_{j} R^{(l+1)}_{j} \frac{x_i w_{ij}}{\epsilon + \sum_h x_h w_{hj}} \tag{2.23}$$

$$\mathbf{Z}_{(n,d2)} = \mathbf{input}_{(n,d1)}\mathbf{weights}_{(d1,d2)} + \epsilon_{(n,d2)} \tag{2.24}$$

However, the addition of the stabilization factor ϵ leads to relevance leaking, and thus compromises the conservation rule described in equation 2.16. A different approach to increase numerical stability and stabilize the relevance propagation is calculating positive and negative activations separately. With the notation that positive activations are defined as $z^{+}_{j} = \sum_i z^{+}_{ij}$ and negative activations are defined as $z^{-}_{j} = \sum_i z^{-}_{ij}$, a relevance message according to the LRP-ab rule is defined as [42]:

$$R^{(l,l+1)}_{i\leftarrow j} = R^{(l+1)}_{j}\left(a\frac{z^{+}_{ij}}{z^{+}_{j}} + b\frac{z^{-}_{ij}}{z^{-}_{j}}\right) \tag{2.25}$$

The parameters a and b are constrained to $a+b=1$ and determine the weighting of positive and negative activations.

Finally, the propagation through all layers of a neural network can be described as an algorithm that propagates the relevance across layers, choosing between the LRP-rules that are defined beforehand [42].

input : $R^L = f(x)$
for $l \in (L, \ldots, 0)$ **do**
  Calculate $R_{i \leftarrow j}^{(l,l+1)}$
  $R_i^{(l)} = \sum_j R_{i \leftarrow j}^{(l,l+1)}$
end
output: $\forall d : \ R_d^1$

Algorithm 1: Propagating relevance through the layers of a neural network [29].

2.3 Adversarial Attacks against Neural Networks

With the increased usage of neural networks in real-world environments, recent research has been done regarding integrity and security concerns of those systems [14]. Studies showed that even state-of-the-art neural networks are vulnerable to perturbed inputs, so called adversarial examples, that are only slightly off the data distribution of training data [23]. In image recognition applications this vulnerability is especially dangerous since the manipulation of the image may be imperceptible to humans while big enough to deceive neural network based systems [14].

2.3.1 Types of Adversarial Attacks

In the context of image classification, an adversarial attack can have one of the following goals according to [14]:

- Reducing classification confidence: In applications with high requirements for confidence, like medical applications or in the automotive area, a reduction of classifier confidence can already lead to misbehaviour of the system
- Misclassification: In this case the goal is to ensure the classification result does not match the correct class, however there is no targeted class that the adversarial example should be classified as.
- Targeted misclassification: Here, the goal is to create adversarial examples so that the classification result is always a specific class, no matter the original input class.
- Source/Target misclassification: A specific adversarial mapping between original classes and predicted classes is created, so that all input images of the original class a are classified as class b and so on.

In order to fulfil these targets, an attacker can modify either input data, training data or the learning algorithm. Besides the goal of an adversarial attack, it can be differentiated between four categories that describe the level of access that the attacker has to the target system. In [14] these levels are described as:

- White-box attack: The attacker has complete access to the model and training data, including full knowledge about architecture, weights, biases and all hyperparameters.
- Non-adaptive black-box attack: The attacker has access to the training data and

can thus train an approximate model on which attacks are created.

- Adaptive black-box attack: The attacker can query the target for an output y on any input x and leverage this to create an approximate model.
- Strict black-box attack: The attacker only has access to input-output pairs but can not query the target with new inputs.

In this work, the focus is on white-box attacks with the goal of misclassification by modifying input data.

2.3.2 Input Manipulation and IFGSM

Adversarial attacks on machine learning models that use data manipulation aim to produce an adversarial input $\tilde{x}$ based on the original input x with a perturbation η that is small enough to not be detectable by humans that might monitor the system or automated defenses. The amount of manipulation that was applied to an image can be described by applying norms:

$$L_1\text{-norm} : ||x||_1 = |x_1| + |x_2| + \cdots + |x_i| \tag{2.26}$$

$$L_2\text{-norm} : ||x||_2 = \sqrt{|x1|^2 + |x2|^2 + \cdots + |x_i|} \tag{2.27}$$

$$L_\infty\text{-norm} : ||x||_\infty = \max\left(|x1|, |x2|, \ldots, |x_i|\right) \tag{2.28}$$

With these definitions, an adversarial input is created by adding the perturbation η to the input, so that $\tilde{x} = x + \eta$ [23]. The step size ϵ of the perturbation is defined by the input data's signal resolution. In this work, all datasets that are used have 8 bit image encoding and are scaled to $[-1, \ldots, 1]$, which translates to a $2/256$ perturbation.

Generally, the goal is to find the minimum perturbation η that can be described by any of the L_p-norms so that the models output is not the correct class y_t [57]:

$$\arg\min_{\eta} ||\eta||_p \rightarrow f(x + \eta) \neq y_t \tag{2.29}$$

To illustrate how a perturbation can be calculated, the established attack method FGSM [23] is introduced in the following paragraphs.

During network training, model parameters Θ together with x and y define the loss as $J(\Theta, x, y)$ [23]. In order to calculate η in FGSM, the gradient of the loss with regard to the input is calculated and its sign is multiplied with ϵ [23]:

$$\eta = \epsilon \cdot \text{sign}(\nabla_x J(\Theta, x, y)) \tag{2.30}$$

Because the gradient can be obtained by backpropagation, FGSM is cheap from a com-

(a) Sample x of class 9
Confidence 0.998

(b) Perturbed $\tilde{x}$ of class 9
Confidence 0.004

(c) Adversarial pattern
$\text{sign}(\nabla_x J(\Theta, x, y))$

Figure 2.4: The original input image belonging to class 9 of the MNIST dataset and the perturbed image after 50 iterations of FGSM together with the adversarial pattern of the first FGSM iteration.

putation standpoint. Much like some explainable AI algorithms, e.g., LIME, FGSM abuses the fact that neural networks behave linear on a local scale even though they are non-linear models [23]. Additionally, adversarial attacks in general benefit from the fact that training data only describes a fraction of the possible input space. An example for an adversarial attack using FGSM for multiple iterations (which is then abbreviated IFGSM) on a CNN that was trained on the MNIST [45] dataset can be found in Figure 2.4.

It should be noted that while visual artifacts of the perturbation can easily be seen in Figure 2.4, 50 iterations were calculated for demonstration purposes and to achieve a saturation in confidence loss on the original class.

Last, a general description of an attack algorithm can be found below. This algorithm takes into account attack methods that rely on multiple iterative attacks in order to sufficiently manipulate the input image. Additionally, the adversarial pattern $\mathbf{A}$, which can take the values $[-1, 0, 1]$ and has the dimensions of $\mathbf{x}$, is introduced. In order to calculate the perturbation η, the adversarial pattern $\mathbf{A}$ is multiplied element-wise with ϵ.

$$\eta = \mathbf{A} \cdot \epsilon \tag{2.31}$$

This way, not only the sign of the perturbation can be defined but also which input data shall be perturbed and which not. This distinction is necessary for the LRP-based adversarial attacks.

input : Θ, x, f, original class, $n_{iterations}$
for $i \in (1, \ldots, n_{iterations})$ **do**
  Calculate $\mathbf{A}$ (in the case of FGSM by $\text{sign}(\nabla_x J(\Theta, x, y))$)
  Multiply $\epsilon \cdot \mathbf{A}$ to calculate η
  Add perturbation η to $\mathbf{x}$
end
output: $\tilde{x}$

Algorithm 2: Generic algorithm for the creation of adversarial inputs.

2.4 Defending against Adversarial Attacks

As increasingly performant attack strategies on neural networks have been researched, defensive strategies were also explored [76, 50, 14]. These defensive strategies can be applied at different stages in the development and deployment of CNNs:

- Network training: adversarial training [23], defensive distillation [76], null-class [62]
- Production/Deployment: adversarial input detection, gradient masking [14]

2.4.1 Defensive Strategies during Training

Applying defensive strategies against adversarial attacks during classifier training has the advantage that the additional computational effort is only applied during training. When the CNN is deployed, no additional latency due to additional computation is necessary.

During adversarial training, the loss function is adapted to include adversarial manipulation on the input sample as a regularization method [23, 76]. However, this additional regularization has two drawbacks: it can worsen the classifiers performance on the initial problem and the regularization term hast to be directed towards one specific attack method. An example of the adapted loss function when adding a regularization term based on FGSM is given below [23]:

$$\tilde{J}(\Theta, x, y) = aJ(\Theta, x, y) + (1 - a)J(\Theta, x + \epsilon \,\text{sign}(\nabla_x J(\Theta, x, y)), y) \tag{2.32}$$

The hyperparameter a in equation 2.32 defines the weight of the regularization term during training. FGSM is computationally efficient since it is using the already available gradient information to create the perturbation. Thus the impact on computational effort during training is not obstructive. However, when more computationally expensive attack algorithms are used for the regularization term, the benefit of the adversarial training must be weighted against the additional computational effort.

Applying defensive distillation during network training is a two-step training method in which a second smaller model is not trained on the original training labels, but the output of the first model. Traditionally, the understanding is that the information that a classifier gained on the training data is stored in the model parameters Θ [2]. Defensive

distillation however utilizes the information that is encoded in a fully trained neural networks output, which is the class probability [30]. The steps of defensive distillation are [14]:

- Train a large / deep neural network on the training data with the training labels
- Store the class probability of the first model for the training data set
- Train a second smaller neural network on the training data with the class probabilities of the first model as labels

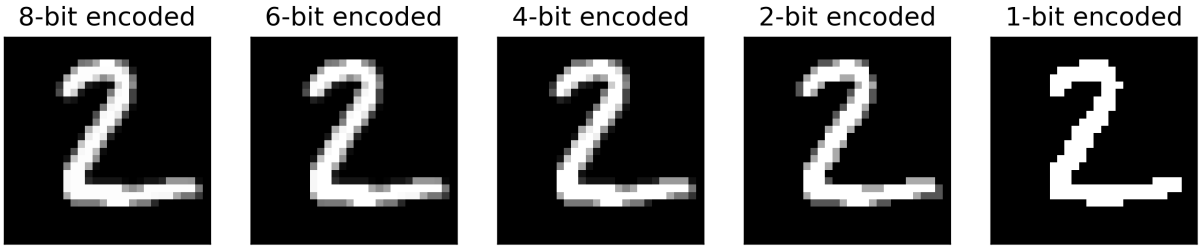
Another approach to reduce the effectiveness of adversarial attacks on neural networks is training with null-labels [62]. When training a neural network on a dataset like MNIST [45] that is a multiclass classification with $k = 10$ classes, Roos proposes to introduce a $k + 1$ class, the null-class. This null-class is supposed to represent the input space that does not belong to the original k classes. In order to use this null-class for training, samples of this class have to be created using methods like random noise or overlaying samples of other classes [62]. The resulting decision boundaries for the original k classes and the null-class thus increase the neural networks robustness against adversarial attacks, which rely on creating samples that are out of the data distribution that was used for training [62].

2.4.2 Defensive Strategies in Production / Deployment

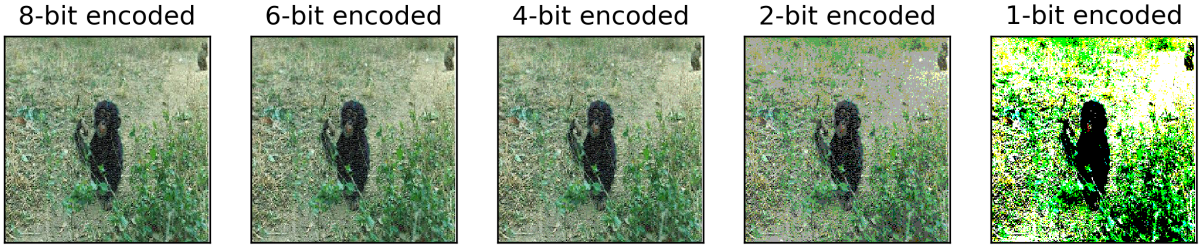
When deploying neural networks into a production environment, defensive strategies can rely on either detecting adversarial inputs and discarding them or on masking the adversaries access to information about the neural network by gradient obfuscation.

Detecting adversarial examples can be achieved by applying specific stepwise preprocessing to images and calculating the output of the neural network for each step. The hypothesis is that for non-adversarial inputs the result of the classification remains consistent, since image preprocessing is already a standard component in image recognition pipelines during training and application. For adversarial inputs however, the classification result will change with each iteration, since the data was already manipulated [77].

During training, image preprocessing is used to increase a neural networks generalization ability by increasing the distribution of input samples. At the same time, preprocessing can be used to smoothen highlights in pictures, to increase their sharpness or to scale the input into pre-defined ranges like $[-1, \ldots, 1]$. One preprocessing technique used for adversarial input detection is feature squeezing [77]. When the input data is encoded in 8-bit images, feature squeezing reduces the bit-depth stepwise until 1-bit encoding is reached. An example on how feature squeezing effects an input image is given in Figure 2.5 .



(a) The effect of bit-depth reduction on a sample image of the MNIST-dataset.



(b) The effect of bit-depth reduction on a sample image of the CALTECH256-dataset.

Figure 2.5: In these examples, bit-depth reduction was applied in multiple steps. As can be seen in a) and b), the image retains the vital information about the content depicted in it.

3 Adversarial Attacks Utilizing Layerwise Relevance Propagation

Based on the introduced explainable AI technique LRP and principles of adversarial attacks, in this chapter a new adversarial attack method is proposed and described in this chapter.

When revisiting equation 2.30 and the definition of the FGSM attack, the intuition behind the attack is to utilize the gradient information of the output with regards to the input. This gradient describes how much the output changes when the input datapoints are manipulated, which can be seen as a relevance matrix for an adversarial attack, since high gradient values imply a high change in $f(x)$ when manipulated. Following this thought, the question is whether explainable AI methods and the information that is gained about which input data influences the output the most, can be used to also derive attacks on the classifier. A Python implementation of layerwise relevance propagation utilizing the TensorFlow framework is described and finally used to implement a adversarial attacks based on LRP.

3.1 LRP Implementation and Concept

Even though Lapuschkin et al. published a Python module LRP-toolbox [43] in which LRP is implemented based on the caffe framework [33], a custom tensorflow based implementation of LRP was built to allow more flexibility for implementing the adversarial attack.

3.1.1 Python Implementation and Design Choices

Up to this point, all definitions and equations are programming language and framework agnostic. The actual implementations are done in Python using the TensorFlow framework. The following versions are used:

- Python: 3.6.7
- Numpy: 1.18.1
- TensorFlow: 2.1.0
- Keras: 2.3.1

White-box adversarial attacks and also layerwise relevance propagation are based on full

access to the model, including model parameters as well as the network architecture. To efficiently work with layerwise relevance propagation to create adversarial attacks, all functions to calculate relevance or the perturbation are implemented in a class called LrpExplainer.

The LrpExplainer class stores the Keras model, hyperparameters a and b as described in equation 2.25, as well as the defined LRP-process. Within the LrpExplainer class, functions are implemented to:

- Retrieve model parameters Θ
- Retrieve model inputs and outputs per layer
- Retrieve the neural network architecture and layer types
- Calculate the relevance matrix R based on algorithm 1
- Calculate adversarial inputs based on algorithm 2

As an example, the helper function to get each layer's weights is lined out in listing 3.1. The function is implemented in such a way that a list of weight matrices is returned. In the case of layers without defined weights (e.g., pooling-layers), *None* is appended to the list.

Listing 3.1: Function to extract weights from the model's layers

```
def get_weights(self, model):
  weights = []
  for i in range(0,len(model.layers)):
    try:
      weights.append(model.layers[i].get_weights()[0])
    except:
      weights.append(None)
  return weights
```

3.1.2 Datasets

In order to evaluate the relevance calculation with LRP and also the adversarial attack based on LRP, four datasets are used.

The MNIST [45] dataset is a well established and widely used dataset that contains 28x28 greyscale images with 8-bit depth of all 10 arabic numerals as can be seen in Figure 3.1. It contains 60.000 training samples and 10.000 test samples. Working with MNIST is computationally cheap, since the image size is only 28x28 pixels and there is only one colour channel. Due to its common usage in research, the comparability with other works is high.

In order to evaluate the LRP-based adversarial attack on RGB images, the CIFAR10 [40] dataset is additionally used. It is a 32x32 RGB 8-bit depth 10-class dataset with 50.000 training samples and 10.000 test samples. Due to the small resolution, this dataset also is computationally cheap but provides information about the LRP-based adversarial attack's performance on RGB images.

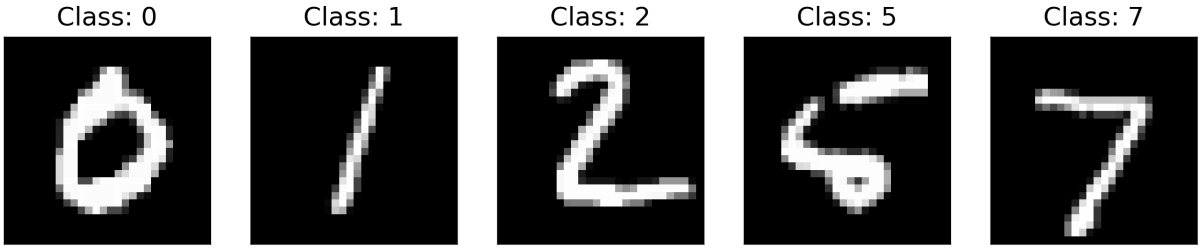


Figure 3.1: Examples taken from the MNIST test-set.

Figure 3.2: Examples taken from the CIFAR10 test-set.

Another dataset to evaluate the adversarial attacks on is the CALTECH-256 dataset [24] that is comprised of RGB images of 256 classes with 8-bit colour depth. This dataset contains images of various sizes and a various number of examples per class, to a total of 30607 images. Also, the aspect ratio is not restricted. When calculating the mean of $\sqrt{width \times height}$ over all images, the mean picture size is 351 pixels [24]. For this work, a Keras datagenerator was trained to rescale all pictures to a $256x256$ picture size. This dataset was chosen since it provides a high variety of classes and because the larger picture size is closer to real world applications than the smaller images of MNIST or CIFAR10.

Figure 3.3: Examples taken from the CALTECH-256 test-set.

The final dataset is the dogs vs. cats [75] dataset. It is a two-class dataset containing in total 25.000 images. It is also an RGB 8-bit colour depth dataset for which a datagenerator is trained to rescale the pictures to a size of $200x200$ pixels.

Each dataset is preprocessed so that the value range per pixel is transformed into $[-1, \ldots, 1]$.

Figure 3.4: Examples taken from the dog vs cat test-set.

3.1.3 Applied Neural Networks

Since this work relies on actual trained neural networks, for each dataset a CNN was trained and evaluated to serve as attack target. The architecture of the CNNs per dataset can be seen in Figure 3.5.

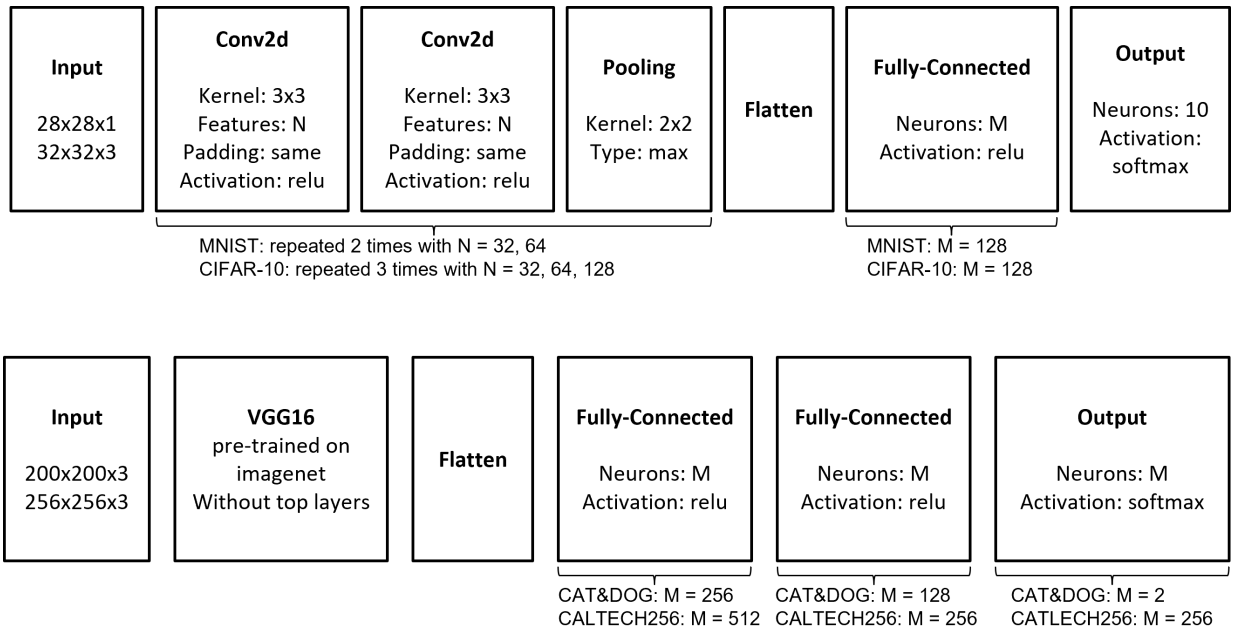


Figure 3.5: CNN architectures for the respective datasets.

While the CNNs for the MNIST and CIFAR10 dataset were fully trained specifically for this work, the CNN for CALTECH256 and the dogs vs. cats dataset utilizes the pretrained convolution layers of a VGG16 [68] CNN with appended flatten and fully-connected layers that were trained specifically for this work. Due to the differing sizes of the CNNs, the computational complexity of calculating the layerwise relevance propagation and the following adversarial attack differs.

A measure that shows this difference between the CNNs is the number of parameters per model, with a higher parameter count indicating higher computational cost:

- MNIST: 467,818 parameters

- CIFAR10: 550,570 parameters
- CALTECH256: 31,689,536 parameters
- dogs vs. cats: 19,466,690 parameters

For training, the datasets were split into training data, validation data and test data. Training and validation data were used to optimize hyperparameters and CNN architecture, while test data was only used to calculate the final f1-scores for each dataset. The final f1-scores of the CNNs for their respective dataset were:

- MNIST: 0.991
- CIFAR10: 0.772
- CALTECH256: 0.695
- dogs vs. cats: 0.953

3.1.4 Applying LRP: Hyperparameters and Process

As described in Section 2.2.4, there are multiple formulas for relevance propagation from layer $l+1$ to layer l that all fulfil the ruleset of LRP as defined in [42]. Here, the importance of utilizing the optimal LRP-formulas based on layer type and position of the layer in the neural network is shown. Additionally, the hyperparameters that were introduced in equation 2.25 will be discussed and a choice for the rest of this work will be made.

First, to further build upon the understanding of the relevance score, a visualization for the relevance matrix is introduced. As introduced in Section 2.2.4, the final relevance matrix has the same dimensions as the original image to which the relevance matrix is calculated. The calculation of the relevance matrix is a function of the model parameters Θ, the input image, the model output $f(x)$ (or a relevance target) and hyperparameters (a,b, process). In this work, the relevance matrix is visualized as a heatmap that is scaled from the minimum of R with negative values depicted red to the maximum of R with positive values depicted blue. This scaling is done per heatmap, meaning all heatmaps have differing scaling. Additionally, for each heatmap, the original class (c_a), the relevance target class (c_t) as well as hyperparameters a and b if not complying to default values are noted in the heatmap's title.

Taking an example image of the MNIST dataset, using an optimized set of hyperparameters, the relevance heatmap of an image of class 7 with the relevance target class 7 can be seen in Figure 3.6.

This relevance heatmap shows positive relevances of up to 0.019 and negative relevances down to -0.001. Calculating the sum over all dimensions of the relevance matrix returns a value of 1, which is the desired outcome, since the class target 7 written as a one-hot encoded vector also has a sum 1. This can be understood as a check-sum that ensures that no numerical instability lead to relevance leaking during the relevance propagation.

As noted in previous paragraphs, c_t can differ from c_a, which leads to a relevance matrix

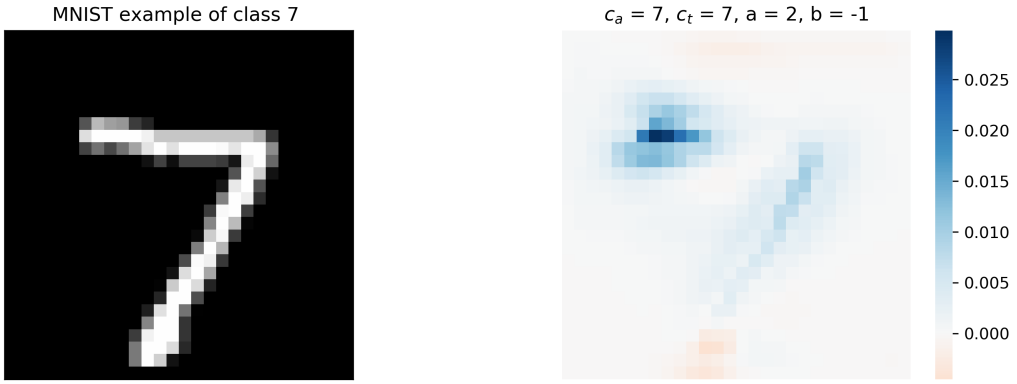


Figure 3.6: For the given input image of the MNIST dataset (left side), the relevance matrix (right side) was computed with optimized hyperparameters. The highest positive relevance is attributed to the upper horizontal line in the 7, while the lower part of the 7 ist attributed negative relevance.

that shows for each input feature positive or negative relevance regarding that class. To give an example, the relevance matrices of the same input as in Figure 3.6 for each possible class of the MNIST dataset were calculated and depicted in Figure A.1.

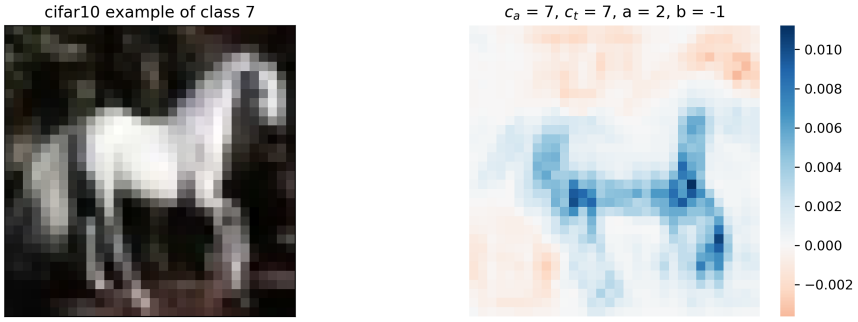


Figure 3.7: A sample input image of the CIFAR10 dataset with its respective relevance heatmap. The relevance is summed up over the colour channels for each pixel.

When displaying heatmaps based on RGB-images, the sum of the relevance of the colour channels for each pixel is calculated and displayed. To showcase this, in Figure 3.7 the relevance matrix for an input image of $c_a = 7$ is calculated and the respective heatmap depicted.

Additionally, the relevance heatmaps for each separate colour channel of the same sample image as in Figure 3.7 are displayed in Figure 3.8. By summing up the relevance per channel, a 2-dimensional depiction can be achieved.

After this introduction of depicting relevance matrices and how c_a and c_t can be different to gain insights on what parts of an image show positive and negative relevances even for a different class,the choice of hyperparameters is discussed next. Revisiting equation

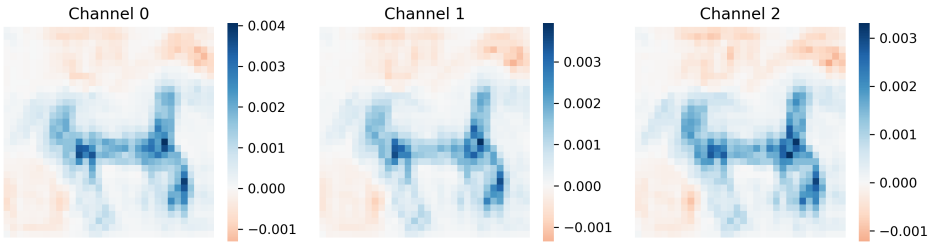


Figure 3.8: The relevance matrices for each separate colour channel of a CIFAR10 example.

2.25, hyperparameters a and b can be used to weigh positive and negative relevances during relevance propagation. They are constrained by the condition that $a+b=1$ [64]. Choosing values of $a=1, b=0$, only positive contributions are propagated, while values of $a=0, b=1$ lead to only negative contributions being propagated. In this work, as proposed in [42], the values $a=2, b=-1$ are used unless otherwise stated.

Lastly, the LRP process that refers to the chosen LRP rule to apply per layer is described. There are multiple possible ways to implement LRP that all comply with the LRP ruleset and these different rules can be concatenated throughout the layers to create an LRP process.

Each variation of LRP has its advantages and disadvantages regarding multiple criteria:

- Computational effort
- Added bias
- Conserving both positive and negative relevances
- Numerical stability

First, the propagation rules for pooling layers are compared. In CNNs, there are two prominently used pooling layers: maxpooling and averagepooling. When applying relevance propagation, these two rules can be separately addressed with corresponding propagation rules: for max pooling layers, only the dimension in layer l with the highest value is attributed the full respective relevance of the respective dimension in layer $l+1$. Similarly, for average pooling layers, the relevance in layer $l+1$ is evenly distributed across all corresponding dimensions in layer l.

When comparing the results of a relevance propagation, no significant differences can be seen in Figure 3.9.

However, extracting the intermediate relevance matrices directly after the upper pooling layer shows how the max pooling leads to sparser relevance matrices. This can be observed in Figure 3.10, where the intermediate relevance matrices with dimensions $R_{14,14}$ are depicted.

In the chosen MNIST example, this does not lead to immediate changes in the final relevance matrix. The full effect of the different pooling propagation methods is shown in the upcoming Figure 3.11, where also different propagation rules for the convolutional

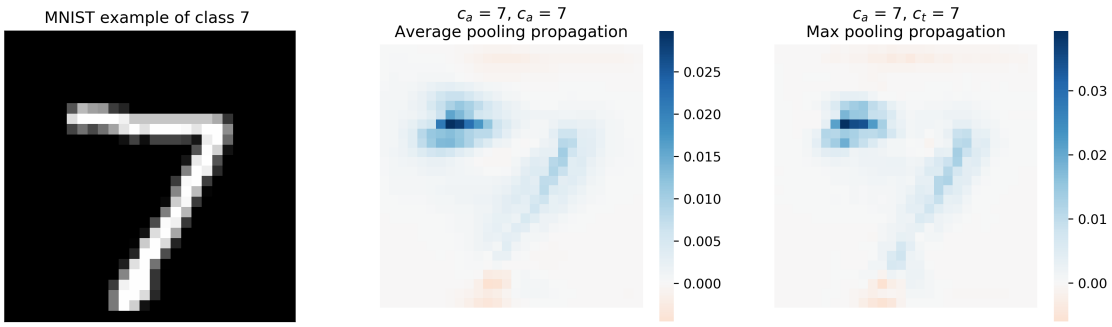


Figure 3.9: An MNIST example of class 7, for which two different LRP processes where applied. The left relevance matrix was created using average pooling rules, the right was created using max pooling rules.

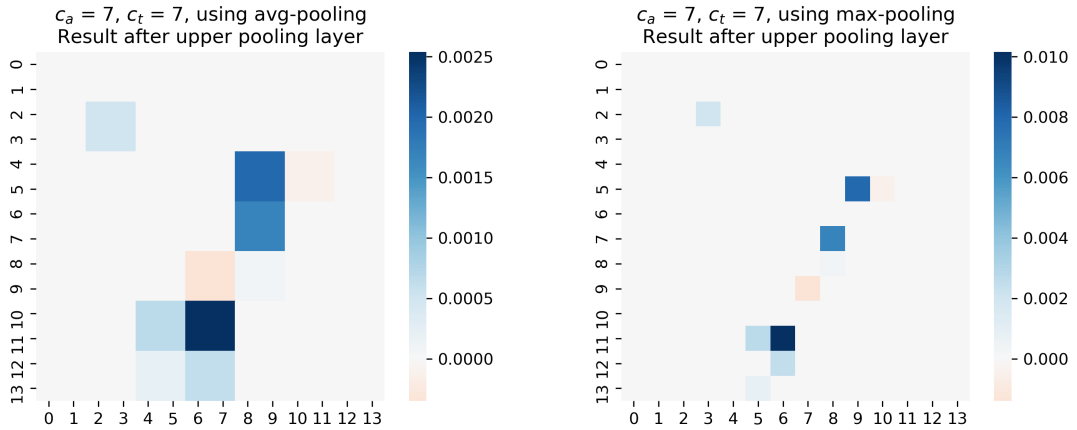


Figure 3.10: Relevance matrices directly after the upper pooling layer were extracted, showing the intermediate relevance matrices.

layers were applied.

For fully connected layers and convolution layers, there are various different propagation rules that adhere to the LRP constraints. In [52], the argumentation to use different LRP rules for different layers, thus creating a composite process, is that upper, middle and lower layers have different levels of entanglement of the concept the model learned. In upper layers, due to their comparatively low neuron count, concepts are likely entangled and a propagation rule that is close to the gradient should be chosen (e.g., the epsilon rule) [52]. However for middle and lower layers, due to the increasing neuron count and thus less entanglement of learned concepts, more sophisticated propagation rules like the $\alpha\beta$-rule can be used to reduce the noise in the final heatmap [52]. The following Figure 3.11 depicts two resulting heatmaps of a composite optimized process and a process that only utilized maxpooling and eps-rule based propagation.

Based on these insights, specific composite LRP processes that were optimized for numerical stability and noise reduction are used per model and dataset.

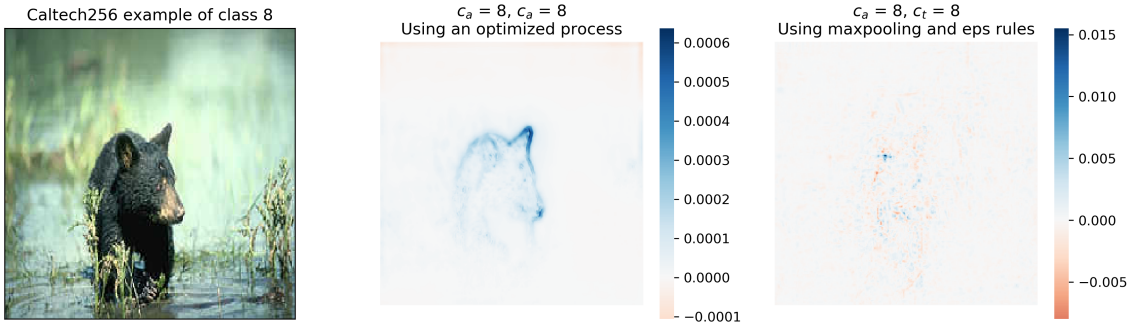


Figure 3.11: The effect of two difference LRP processes on the resulting heatmap of a CALTECH256 example. The heatmap created with a composite process is less noisy and better highlights the subject of the image than the heatmap created with only maxpooling and epsilon rules.

The final point to be made about the LRP process is that in the process of optimizing the resulting heatmaps for noise reduction and numerical stability, a bias is introduced. To a human eye, the left relevance heatmap of Figure 3.11 seems like a more understandable and plausible explanation, than the noisy right heatmap. However, this is just a question of preference. The goal of explainable AI is to provide explanations that are not biased but rather allow us to assess the actual decision making of a classifier.

In the end, as long as the chosen LRP process utilizes only LRP rules that adhere to the LRP framework, the mathematical foundation holds and thus the creation of the heatmaps justified. The research question, what influence the LRP process has on the effectiveness of the adversarial attacks utilizing LRP is not further analysed in this work, however intriguing to pursue.

3.2 LRP-based Adversarial attacks

While explainable AI and adversarial attacks may seem to be two separate research fields that aim to achieve opposite goals, on a second thought, they are based on similar prepositions. Adversarial attacks aim to find the minimum necessary perturbation η to fool the classifier, as described in equation 2.29. In the case of FGSM this leads to a gradient that if multiplied with ϵ will lead to the highest change in $f(\tilde{x})$. In explainable AI, more specifically the explanation of single classifier decisions as described in previous chapters, a metric to determine the contribution of each input dimension to the decision is calculated.

Thus in both fields, for every input dimension, a numeric value is attributed that describes either the effect of manipulating that dimension or the contribution of that dimension to the classifier decision. This common ground is used in this work to propose a new adversarial attack utilizing an explainable AI algorithm, more specifically LRP, to determine η in adversarial attacks.

3.2.1 LRP-Based Attacks Utilizing Pixel Flips

Having established how a relevance matrix is calculated using LRP, multiple approaches to generating perturbations based on the relevance matrix are proposed.

The first method is taken over from the paper [6], in which Lapuschkin et al. use pixel flipping as a way to verify that the relevance matrices highlight important pixels for the classification. As established in Section 3.1.2, the value range per pixel is $[-1, \dots, 1]$, allowing the multiplication with -1 to flip the pixel's sign. In this method, the adversarial pattern A is defined by the following equation 3.1.

$$A_{i,j,k} = \begin{cases} 1, & \text{if } R_{i,j,k} = max(R) \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Calculating η per pixel leads to a pixelwise perturbation of $\eta_{i,j,k} = 2 \cdot x_{i,j,k}$ due to the symmetry of the value range $[-1, \dots, 1]$. The first perturbation approach can then be described by the following algorithm 3 for greyscale images.

input : Θ, x, f, c_a, c_t, $n_{iterations}$
for $i \in (1, \dots, n_{iterations})$ **do**
  Calculate R with $c_t = c_a$
  Find A
  Set $\tilde{x}_{j,k} = x_{j,k} \cdot -1$ if $A_{j,k} > 0$
end
output: $\tilde{x}$

Algorithm 3: Algorithm to generate an adversarial input based on greyscale images using the LRP-flip method.

Since the focus of this work is to create white-box attacks with the goal of misclassifi-

cation, the calculation of R in algorithm 3 is done with $c_a = c_t$. However, by changing c_t to any other class, a targeted misclassification can be created. Applying algorithm 3 to a sample image of the MNIST dataset leads to the result depicted in Figure 3.12 Calculating the measures for image manipulation, i.e., L-norms, for this example leads to the following results:

- L_1-norm: 102.84
- L_2-norm: 14.08
- L_∞-norm: 2.0

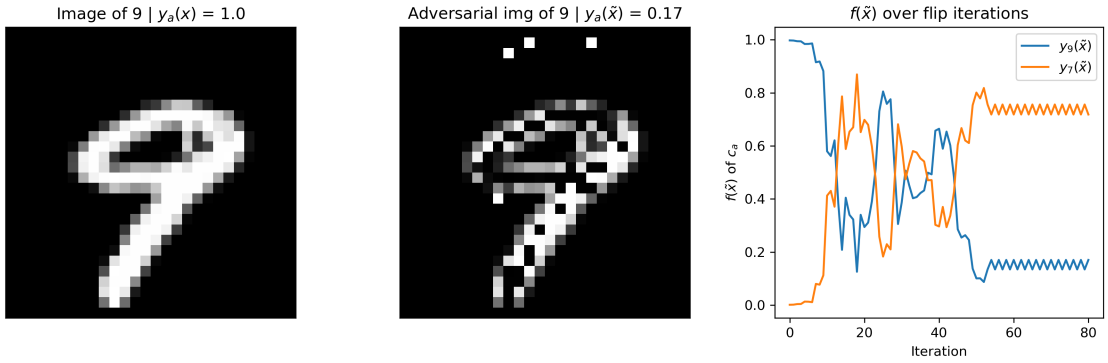


Figure 3.12: The original image and the manipulated adversarial image of an MNIST example with the according output of the classifier.

In the example in Figure 3.12, the image manipulation was not stopped after achieving a misclassification, but rather after a set number of iterations. After 13 iterations, the classifier predicted the class to be 7 instead of 9. However, this behaviour changes four times until a stagnation or rather oscillation of the predicted class can be observed. In that case, flipping the most relevant pixel leads to looping behaviour.

Another observation is that pixels that are not part of the initial 9 were flipped, showing how the CNN uses context information beyond the subject of the image to classify the image. For greyscale images, the flipping of pixels is intuitive, since it changes black pixels to white pixels and white pixels to black pixels. For RGB images, the algorithm has to be extended to include the colour channel as parameter, as can be seen in algorithm 4.

input : Θ, x, f, c_a, c_t, $n_{iterations}$
for $i \in (1, \ldots, n_{iterations})$ **do**
  Calculate R with $c_t = c_a$
  Calculate A according to equation 3.1
  Set $\tilde{x}_{j,k,l} = x_{j,k,l} \cdot -1$ if $A_{j,k,l} > 0$
end
output: $\tilde{x}$

Algorithm 4: Algorithm to generate an adversarial input based on RGB images using the LRP-flip method.

The resulting adversarial image can be seen in Figure 3.13, here the pixels change colour, per manipulated pixel only the colour channel with the highest relevance score is flipped. For all upcoming attack methods, only the version applicable to RGB images will be explicitly shown.

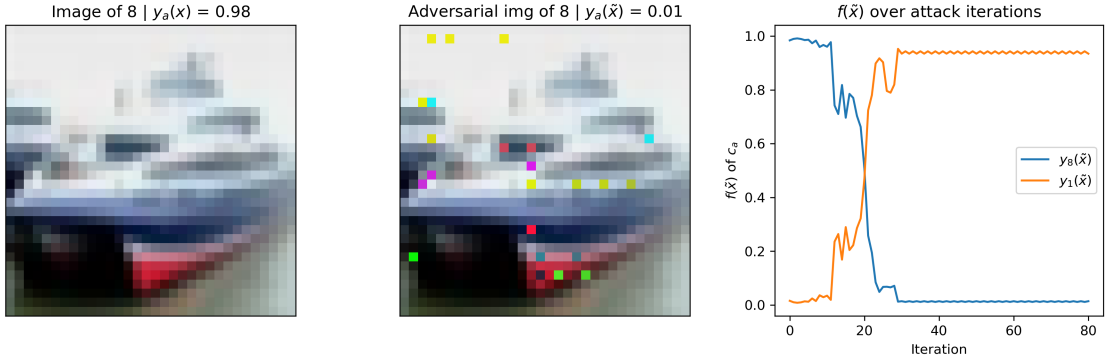


Figure 3.13: The original image and the manipulated adversarial image of a CIFAR10 example when the flip attack is applied. Additionally, the according output of the classifier $f(\tilde{x})$ is depicted over the attack iterations.

Similarly to Figure 3.12, at one point the algorithm shows looping behaviour, where $f(\tilde{x})$ oscillates between two values, because the same pixels are manipulated repeatedly. Building on this attack approach, a computationally cheaper version of algorithm 4 manipulates multiple pixels per loop, reducing the number of relevance propagations that have to be done.

Applying this optimized batch flip attack to the same input image as in Figure 3.13 leads to the result shown in Figure 3.14.

Here, similar areas of the original image are manipulated when compared to Figure 3.13. However, the $f(\tilde{x})$ over the iterations seems more monotonous and does not show oscillating behaviour. Whether the batch flip attack is generally more stable than the flip attack will be discussed in a later section. For all further attack methods, this optimization to modify multiple pixels per relevance propagation will not be explicitly shown but applied where necessary to reduce computational effort.

input : Θ, x, f, c_a, c_t, $n_{iterations}$, n_{batch}
for $i \in (1, \ldots, n_{iterations})$ **do**
  Calculate R with $c_t = c_a$
  for $m \in (1, \ldots, n_{batch})$ **do**
    Find j, k, l so that $R_{j,k,l} = max(R)$
    Set $\tilde{x}_{j,k,l} = x_{j,k,l} \cdot -1$
  end
end
output: $\tilde{x}$

Algorithm 5: Algorithm to generate an adversarial input based on RGB images using the LRP-batchflip method. Per calculation of R, multiple pixels are manipulated to reduce the computational effort.

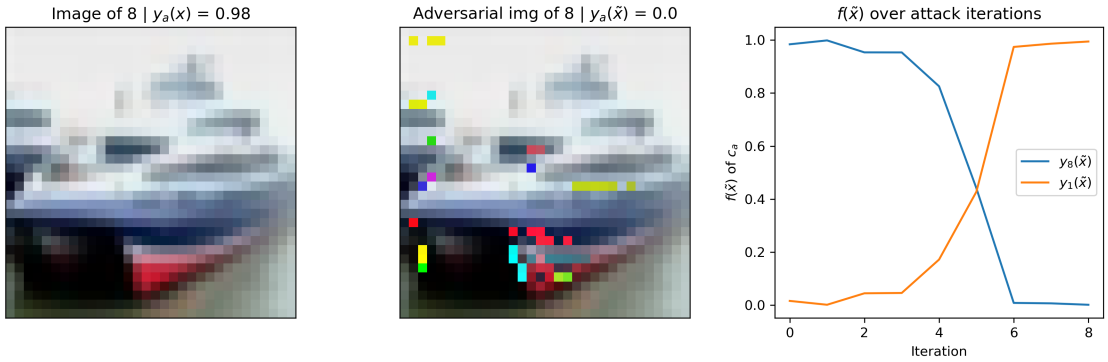


Figure 3.14: The original image and the manipulated adversarial image of a CIFAR10 example when the batch flip attack is applied. Additionally, the according output of the classifier is depicted over the attack iterations.

3.2.2 LRP-Based Attack Utilizing Pixel Averaging

The next proposed attack method is the LRP-mean attack. Here, the input dimension with the highest relevance is manipulated towards the mean of that colour channel over all pixels. First, the adversarial pattern is defined like in equation 3.1. Then for each colour channel, the mean pixel value m_k is calculated and η is calculated by the following equation 3.2.

$$\eta_{i,j,k} = \text{sign}(m_k - x_{i,j,k}) \cdot \epsilon \tag{3.2}$$

This method highlights the problem that LRP-based adversarial attacks have. While relevance propagation provides information about whether a pixel contributes positively or negatively to one specific classification, there is no information about the influence of increasing or decreasing the value of this pixel and colour channel on the classification outcome. The LRP-mean attack assumes that averaging out all pixels eventually leads to a strong enough information reduction such that a misclassification will occur. In the following algorithm, the LRP-mean method is described.

input : Θ, x, f, c_a, c_t, $n_{iterations}$
Calculate m_k for each colour channel
for $i \in (1, \ldots, n_{iterations})$ **do**
  Calculate R with $c_t = c_a$
  Calculate A according to equation 3.1
  Calculate η according to equation 3.2
  Set for each pixel and colour channel $\tilde{x}_{j,k,l} = x_{j,k,l} + \eta_{j,k,l}$ if $A_{j,k,l} > 0$
end
output: $\tilde{x}$

Algorithm 6: Algorithm to generate an adversarial input based on RGB images using the LRP-mean method.

Applying this attack method to the same example from the cifar10 dataset again leads to the result shown in Figure 3.15.

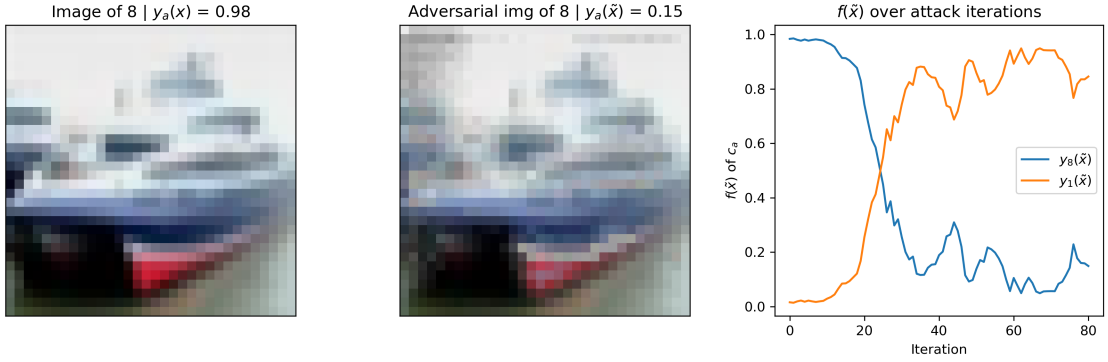


Figure 3.15: The original image and the manipulated adversarial image of a CIFAR10 example when the LRP-mean attack is applied. Additionally, the according output of the classifier is depicted over the attack iterations.

From the graph of $f(\tilde{x})$ it can be seen that this method for this specific example seems to lead to slower changes in the classifiers confidence. The final $y_a(\tilde{x})$ did not fall as low compared to the other methods.

3.2.3 LRP-Based Attack Utilizing Gradient Information

Finally, the last proposed attack method is a composite method combining LRP and gradients, the LRP-grad method. The idea is to utilize the information about the relevance of individual pixels and colour channels to prioritize which input dimension to modify. The perturbation itself then is based on the adversarial pattern that is defined in equation 2.30. First the adversarial pattern A is calculated according to the following equations 3.3 and 3.4.

$$A = \text{sign}(\nabla_x J(\Theta, x, y)) \tag{3.3}$$

$$A_{i,j,k} = \begin{cases} A_{i,j,k}, & \text{if } R_{i,j,k} = max(R) \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

These equations show how the LRP-grad approach combines the gradient information that is utilized in I-FGSM together with the prioritization information that is gained by applying LRP. Based on these definitions, η is then accordingly calculated by equation 3.5.

$$\eta = A \cdot \epsilon \tag{3.5}$$

Finally, the algorithm for the LRP-grad method is formulated in algorithm 7.

input : Θ, x, f, c_a, c_t, $n_{iterations}$
for $i \in (1, \ldots, n_{iterations})$ **do**
  Calculate R with $c_t = c_a$
  Calculate A according to equations 3.3 and 3.4
  $\eta = A \cdot \epsilon$
  $\tilde{x} = x + \eta$
end
output: $\tilde{x}$

Algorithm 7: Algorithm to generate an adversarial input based on RGB images using the LRP-grad method.

In Figure 3.16, the result of applying this attack method to the cifar10 example image is depicted.

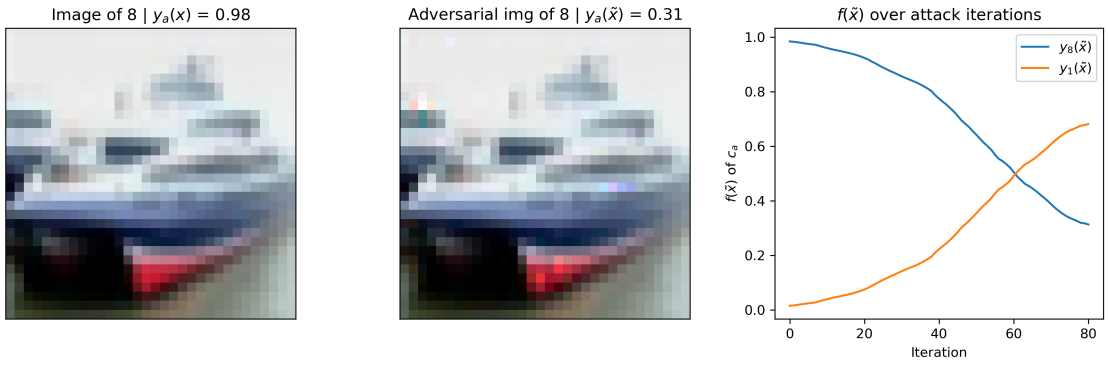


Figure 3.16: The original image and the manipulated adversarial image of a CIFAR10 example when the LRP-grad attack is applied. Additionally, the according output of the classifier is depicted over the attack iterations.

One observation to be made of the graph showing $f(\tilde{x})$ over the attack iterations in Figure 3.16 is that the decrease in classifier confidence is much smoother compared to the other methods.

4 Evaluating LRP-Based Attacks against IFGSM

Two novel LRP-based adversarial attacks were proposed and described. Additionally, the IFGSM and LRP-flip attack was analysed and implemented. Here, the following attack methods will be quantitatively evaluated:

- LRP-flip
- LRP-mean
- LRP-grad
- IFGSM

Each attack method will first be analysed regarding their numerical image manipulation from a theoretical perspective. After this they will be applied to the four image datasets and their CNNs, and their performance will be evaluated.

4.1 Analytical Comparison of the Proposed Attack Methods

The metrics to evaluate how much an image has been perturbed were described in equations 2.26, 2.27 and 2.28. For all L-norms, the upper and lower bounds per dataset will be analysed and a generic description of the worst case scenario for the L-norms derived. Because this theoretical approach assumes maximum perturbation effects, it is limited in its full validity and will be compared to the experimental results. The limitation is due to the fact that during the attack iterations, the sign of the gradient for one single pixel and colour channel can change multiple times. For the maximum perturbation evaluation though, we assume that each pixel and colour channel will be perturbed into the same direction for all iterations.

The metric D describes the total number of input values of an image, which is calculated by $D = height \cdot width \cdot n_{channels}$. This metric D takes the following values for the datasets:

- MNIST: $D = 28 \cdot 28 \cdot 1 = 784$
- CIFAR10: $D = 32 \cdot 32 \cdot 3 = 3,072$
- CALTECH256: $D = 256 \cdot 256 \cdot 3 = 196,608$
- dog vs. cat: $D = 200 \cdot 200 \cdot 3 = 120,000$

4.1.1 Upper and Lower Bounds of the L-Norms

Firstly, since all L-norms evaluate the absolute values of perturbation, as can be seen in equations 2.26, 2.27 and 2.28, the lower bound for all L-norms is 0.

For the L1-norm, the definition of equation 2.26 can be rewritten using η to the following definition.

$$||L_1|| = \sum_{i,j,k} |\eta_{i,j,k}| \tag{4.1}$$

Thus the maximum value the L_1-norm can take would be a maximum perturbation of each pixel. Due to the value range of the pixel values of $[-1, \ldots, 1]$, and since the metric D was introduced, the L_1-norm upper bound can be calculated as following:

$$||L_1|| \leq 2 \cdot D \tag{4.2}$$

Thus, the upper and lower bound of the L_1-norm for each dataset is:

- MNIST: $||L_1|| \in [0, \ldots, 1568]$
- CIFAR10: $||L_1|| \in [0, \ldots, 6144]$
- CALTECH256: $||L_1|| \in [0, \ldots, 393216]$
- dog vs. cat: $||L_1|| \in [0, \ldots, 240000]$

Coming to the L_2-norm, rewriting it using η leads to equation 4.3.

$$||L_2|| = \sqrt{\sum_{i,j,k} \eta_{i,j,k}^2} \tag{4.3}$$

Similarly to the L_1-norm, η can be substituted by 2 and the sum function by multiplying with D.

$$||L_2|| = \sqrt{2^2 \cdot D} \tag{4.4}$$

It follows that the upper and lower bound of the L_2-norm for each dataset are:

- MNIST: $||L_2|| \in [0, \ldots, 56]$
- CIFAR10: $||L_2|| \in [0, \ldots, \sim 110.85]$
- CALTECH256: $||L_2|| \in [0, \ldots, \sim 886.81]$
- dog vs. cat: $||L_2|| \in [0, \ldots, \sim 692.82]$

Lastly, the L_∞-norm's upper and lower bounds are independent of the datasets and only depending on the value range that was defined per pixel, since the max function is applied to the perturbation η. Thus the value range is $||L_\infty|| \in [0, \ldots, 2]$ for all datasets.

4.1.2 Perturbation of the Attack Methods per Iteration

For the IFGSM method, calculating the L-norms is straightforward. Based on the definitions in equations 2.30 and 2.31, it can be deducted that per iteration, each pixel and colour channel is perturbed by ϵ. Thus the following formulas are derived for IFGSM after replacing η with ϵ in equation 4.1 and replacing the sum term with D:

$$||L_1(n_{iter}, D, \epsilon)|| = \epsilon \cdot D \cdot n_{iter} \tag{4.5}$$

$$||L_2(n_{iter}, D, \epsilon)|| = \sqrt{(\epsilon \cdot n_{iter})^2 \cdot D} \tag{4.6}$$

$$||L_\infty(n_{iter}, \epsilon)|| = \epsilon \cdot n_{iter} \tag{4.7}$$

The definitions for the LRP-grad method are similar, with the difference that only a specific number of pixels is targeted. Thus, in equations 4.5, 4.6 and 4.7 the term D is substituted by n_{batch} with $n_{batch} \in [1, \ldots, D]$, leading to the following equations:

$$||L_1(n_{iter}, n_{batch}, \epsilon)|| = \epsilon \cdot n_{batch} \cdot n_{iter} \tag{4.8}$$

$$||L_2(n_{iter}, n_{batch}, \epsilon)|| = \sqrt{(\epsilon \cdot n_{iter})^2 \cdot n_{batch}} \tag{4.9}$$

$$||L_\infty(n_{iter}, \epsilon)|| = \epsilon \cdot n_{iter} \tag{4.10}$$

For the LRP-mean method, the equations 4.8, 4.9 and 4.10 also apply. This can be derived from equation 3.2, since the sign() function only returns a value in $[-1, 0, 1]$, similarly to equation 2.30. While the adversarial pattern A is different between LRP-grad and LRP-mean, they both share the property that only a subset of all pixels is perturbed by ϵ per iteration. The difference between the theoretical maximum perturbation of these methods is not related to the perturbation per iteration, but the maximum achievable perturbation.

It follows that for LRP-grad it is possible, even if highly unlikely, for an image to consist of only black pixels and all those pixels being changed to white pixels. The LRP-mean method on the other hand will have the highest total perturbation when half of all pixels are black, the other half white and all of them will be converted to pixels with the value 0. Essentially, in equations 4.2 and 4.4, the multiplier 2 has to be halved for the LRP-mean method.

Lastly, analysing the LRP-flip attack, the first observation to be made is that for each pixel, only two states will exist. The original state and the flipped state. Thus, the perturbation will reach the maximum, once every pixel has been flipped exactly once. Assuming that no pixel will be flipped an even number of times, even though the first experiments already showed different behaviour in Figure 3.13, the perturbation η per

iteration are dependent on the value the flipped pixel and colour channel has. Thus, to derive a way to estimate the perturbation per iteration, for each dataset an average absolute pixel value must be calculated according to equation 4.11.

$$\overline{x} = \frac{\sum_{i,j,k} |x_{i,j,k}|}{D} \tag{4.11}$$

This average pixel value $\overline{x}$ can then be used to derive the L-norms.

$$||L_1(n_{iter}, n_{batch}, \epsilon)|| = \overline{x} \cdot n_{batch} \cdot n_{iter} \tag{4.12}$$

$$||L_2(n_{iter}, n_{batch}, \epsilon)|| = \sqrt{(\overline{x} \cdot n_{iter})^2 \cdot n_{batch}} \tag{4.13}$$

The L_∞-norm can be derived from the maximum absolute pixel value of the input data x.

$$||L_\infty(n_{iter}, \epsilon)|| = 2max(|x|) \tag{4.14}$$

4.1.3 Comparing the Attack Methods in a Theoretical Setting

In order to compare the four attack methods, first the metric $\overline{x}$ has to be calculated for the datasets. This leads to the following values:

- MNIST: $\overline{x} = 0.946$
- CIFAR10: $\overline{x} = 0.423$
- Caltech-256: $\overline{x} = 0.565$
- dog vs cat: $\overline{x} = 0.441$

With these values for $\overline{x}$ and setting $\epsilon = 2/256$, the total perturbation η per iteration and per batch size can be calculated. In this comparison, the focus is on the L_1 and L_2 norms. For this, the L-norms are plotted for different n_{iter} and n_{batch} values. Because the LRP-grad and LRP-mean methods share the theoretic formulas, they are combined under the LRP-grad label.

Because IFGSM does not use n_{batch} as parameter, the L-norms were only calculated as a function of n_{iter}. For the LRP-flip algorithm, the maximum of $n_{iter} \cdot n_{batch}$ is equal to D, which would lead to a narrow surface. Thus, the LRP-flip L-norms were plotted as scatterplots for visibility reasons. Since the plot will show similar geometric and analytical properties for all datasets, only the MNIST figure is depicted.

In Figure 4.1 it can be seen that the IFGSM, LRP-grad and LRP-mean methods share the same values for the L_1- and L_2-norm, when $n_{batch} = D$. This can also be derived from equations 4.5 and 4.8. An insight gained from the plots in Figure 4.1 is that n_batch influences the resulting L_1-norm linearly but the L_2-norm quadratic for LRP-grad. A hypothesis from that is that the LRP-grad method will exhibit a larger advantage

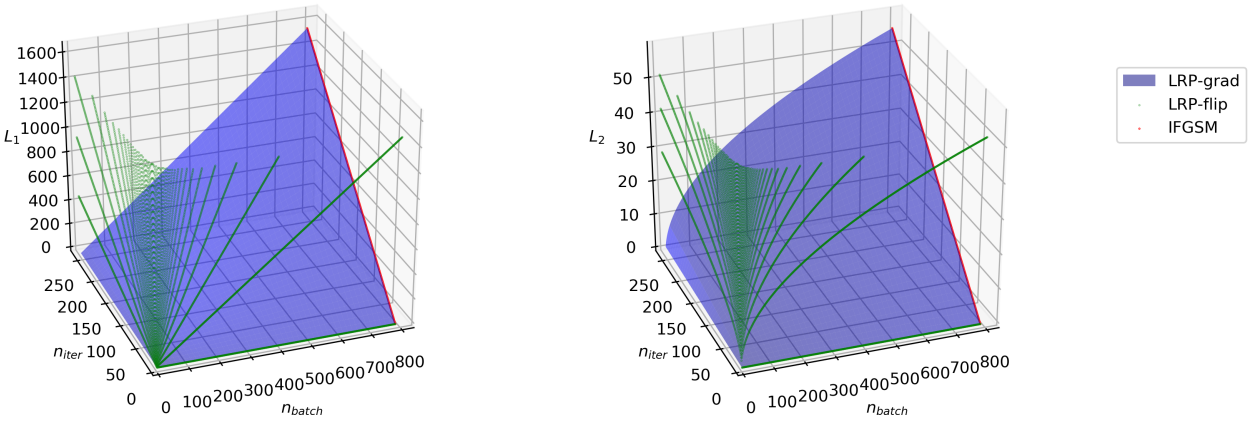


Figure 4.1: The 3D surfaces of the L-norms for all evaluated attack methods using the MNIST dataset parameters. The LRP-flip L-norms are plotted as scatterplots.

compared to IFGSM in the experiments regarding the L_1-norm, than the L_2-norm.

Furthermore, it can be seen that for the LRP-flip method, L_1- and L_2-norm values do not reach the theoretical upper bound for the dataset. This can be explained by the fact that $2 \cdot \overline{x} < 2$ and thus equations 4.12 and 4.13 exhibiting smaller upper limits than equations 4.8 and 4.9.

However, this theoretical comparison can only show the expected perturbation values. Since this does not show, how effective the perturbation was in leading to a misclassification, an experimental benchmarking will be conducted.

4.2 Benchmarking the Proposed Attack Methods against IFGSM

After the attack methods have been proposed and the theoretical analysis has been concluded, they will be applied to the four datasets and evaluated. For this, first the setup for the benchmark will be introduced, then the results per method presented and lastly they will be compared.

4.2.1 Benchmarking Setup

In accordance to benchmarking literature [66, 53], only images on which the classifier was not trained on will be used. Thus, a sample of the test set of each dataset will be used to conduct the benchmark. Additionally, for each method the exact same images will be used to ensure comparability between the results.

To evaluate the performance of each method, the L-norms were already introduced for

measuring the perturbation. In order to evaluate the methods, metrics to evaluate their success have to be introduced as well. One metric used in literature [74] is the attack success rate (ASR). Here, the ASR is defined as the fraction of the true positive classifications $n_{(TP)}$ at a certain time $t = x$ compared to the true positive classifications before the adversarial attack $t = 0$:

$$ASR = \frac{n_{(TP,t=0)} - n_{(TP,t=x)}}{n_{(TP,t=0)}} \tag{4.15}$$

This ensures that the initial performance of the classifiers of each dataset does not influence the evaluation metric.

Furthermore, for each classifier $f(\tilde{x})$ will be documented per iteration to gather more detailed information about the distribution of the numeric impact.

The benchmark will be conducted using the following hardware:

- CPU: Intel i7-7700
- GPU: NVIDIA GeForce GTX1080
- RAM: 32 GB

GPU-acceleration utilizing the CUDA framework is applied whenever possible. However the LRP-based attacks were not optimized regarding computational effort.

4.2.2 Optimizing Attack Parameters

The proposed LRP-based attack methods rely on two parameters n_{iter} and n_{batch}. To ensure that each attack method performs optimally, the parameters are determined experimentally. This process will be shown based on the CIFAR10 dataset, however it was conducted separately for all datasets and their respective classifiers.

To ensure that the attack methods can be compared on the dataset in graphs, the parameter n_{iter} will be set to 100. With this, the graph of ASR and the output of the classifier for the original class c_a for the perturbed image $\tilde{x}$, $f_{c_a}(\tilde{x})$, can be plotted. For the $f_{c_a}(\tilde{x})$ graph, boxplots with the mean highlighted as orange line, the box ranging from the lower to the upper quartile and the 5^{th} and 95^{th} percentile represented as whisker are created per iteration.

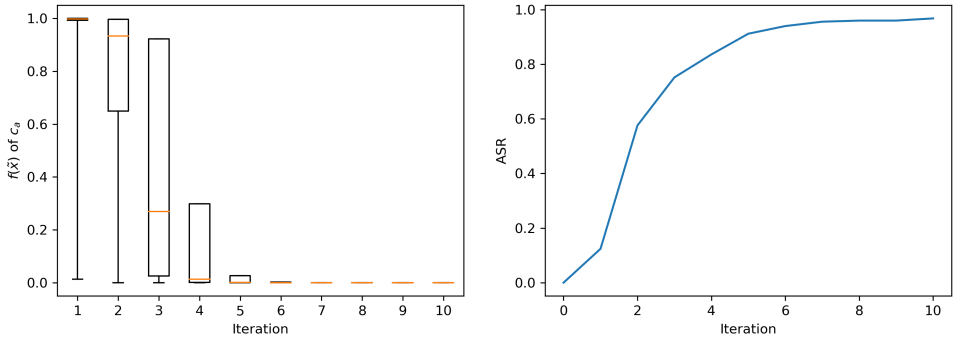


Figure 4.2: $f_{c_a}(\tilde{x})$ and ASR of IFGSM applied to the CIFAR10 dataset. The boxplot shows the distribution of the classifier's confidence on the adversarial images over the iterations. In the right graph the respective ASR over iterations is displayed.

As can be seen in Figure 4.2, the IFGSM method already reaches a 90% ASR after five iterations. Additionally, the $f_{c_a}(\tilde{x})$ distribution gets narrower with higher iteration count. Thus it can be concluded that for IFGSM 100 iterations are enough to reach a saturation in the attack success.

Testing the LRP-flip method on the CIFAR10 dataset with $n_{iter} = 100$ and $n_{batch} = 100$ lead to the following results.

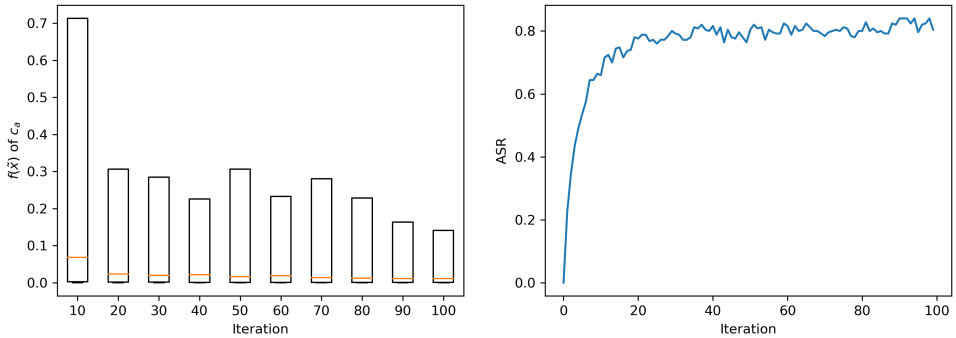


Figure 4.3: $f_{c_a}(\tilde{x})$ and ASR of LRP-flip applied to the CIFAR10 dataset.

The boxplots of $f_{c_a}(\tilde{x})$ show a steep decline in mean classifier confidence, with the first quartile stagnating between iterations 20 and 70, until it finally decreases for the last 30 iterations. In the ASR plot, the success rate strongly increases for the first 30 iterations, after which the ASR plot does not show further significant improvements, with the ASR stagnating at about 80%.

The next method LRP-mean was applied to the CIFAR10 dataset using $n_{iter} = 100$ and $n_{batch} = 500$.

A decline in mean classifier confidence can be observed in Figure 4.4, however the 5^{th}

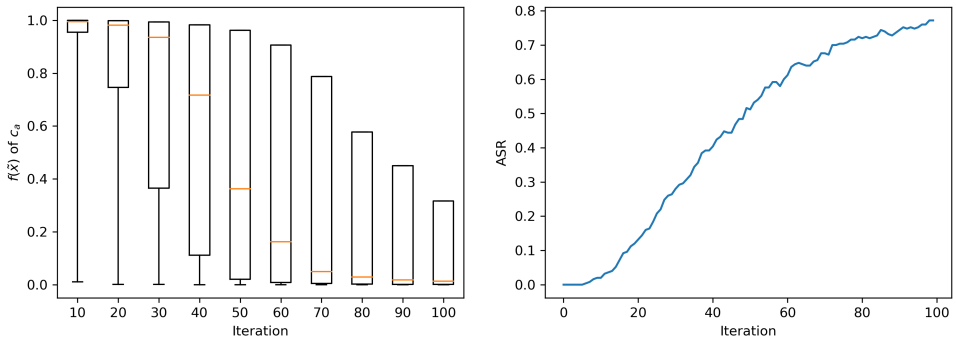


Figure 4.4: $f_{c_a}(\tilde{x})$ and ASR of LRP-mean applied to the CIFAR10 dataset. The boxplot shows the distribution of the classifier's confidence on the adversarial images over the iterations. In the right graph the respective ASR over iterations is displayed.

percentile of $f_{c_a}(\tilde{x})$ in the boxplot does not yet approach a limit. This can be also seen in the ASR that reaches a value of 80% by 100 iterations. For the final experiment, n_{batch} will be increased to ensure that ASR reaches its maximum.

Finally, LRP-grad was applied using $n_{iter} = 100$ and $n_{batch} = 200$.

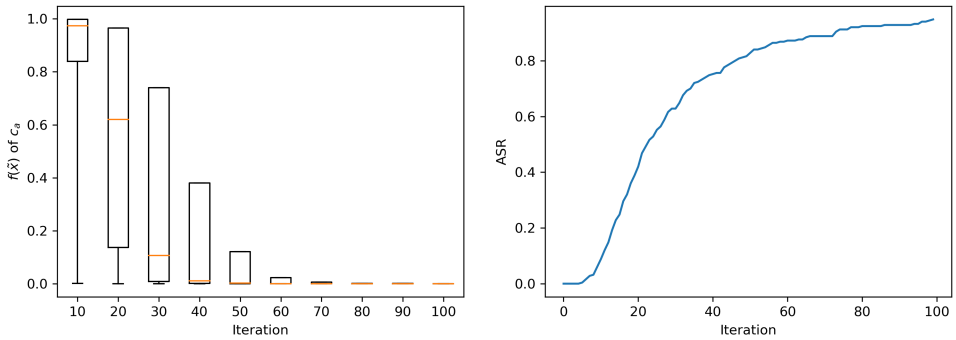


Figure 4.5: $f_{c_a}(\tilde{x})$ and ASR of LRP-grad applied to the CIFAR10 dataset. The boxplot shows the distribution of the classifier's confidence on the adversarial images over the iterations. In the right graph the respective ASR over iterations is displayed.

Similar to the graph that was observed in 4.2, the boxplots of $f_{c_a}(\tilde{x})$ in Figure 4.5 show that the attack method reached saturation. The ASR reaches a maximum of 94.8% by 100 iterations, with a steady increase over all iterations.

The final choice of attack parameters for all datasets and methods is shown in the following Table 4.1. Additionally the number of images n_{images} that were used for the experiment per dataset are listed in that table.

Table 4.1: Attack parameters n_{iter}, n_{batch} and ϵ for all datasets and attack methods.

Dataset	Method	n_{iter}	n_{batch}	ϵ	n_{images}
MNIST	IFGSM	100	D	2/256	250
	LRP-flip		5	-	
	LRP-mean		100	2/256	
	LRP-grad		300	2/256	
CIFAR10	IFGSM	100	D	2/256	250
	LRP-flip		100	-	
	LRP-mean		500	2/256	
	LRP-grad		250	2/256	
CALTECH256	IFGSM	100	D	2/256	128
	LRP-flip		500	-	
	LRP-mean		1000	2/256	
	LRP-grad		2000	2/256	
dog vs. cat	IFGSM	100	D	2/256	128
	LRP-flip		500	-	
	LRP-mean		10000	2/256	
	LRP-grad		200	2/256	

4.2.3 Benchmarking Results

To compare the effectiveness of the attack methods on the MNIST dataset, the L-norms and ASR of each method were plotted over the iteration count in Figure 4.6.

In the first plot of Figure 4.6, the L_1-norm of LRP-flip initially increases the steepest with each iteration. However, as expected by the upper bound analysis in Figure 4.1, the LRP-flip method reaches an upper bound and is finally surpassed by the IFGSM method. The LRP-mean method shows the slowest rise in L_1 and stays at the lowest L_1 over all iterations.

For the L_2-norm, a similar observation can be made, however the LRP-flip method is not surpassed by IFGSM at any point.

In the L_∞-norm, the LRP-flip method quickly reaches the upper limit due to the behaviour described in equation 4.14. The IFGSM, LRP-mean and LRP-grad method on the other hand show a steady increase in L_∞, with the LRP-mean method exhibiting the lowest increase and final L_∞. The hypothesis for this behaviour is that in the LRP-mean method, in the beginning there are pixels that are perturbed in nearly all iterations due to their high relevance across multiple iterations. However, if they at some point reach the mean pixel value but still show high relevances, they are no longer perturbed and the L_∞-norm rises slower.

Finally, comparing the ASR across iterations, the LRP-flip method exhibits the steepest increase, however again reaching a limit at about 75%, from where no further improve-

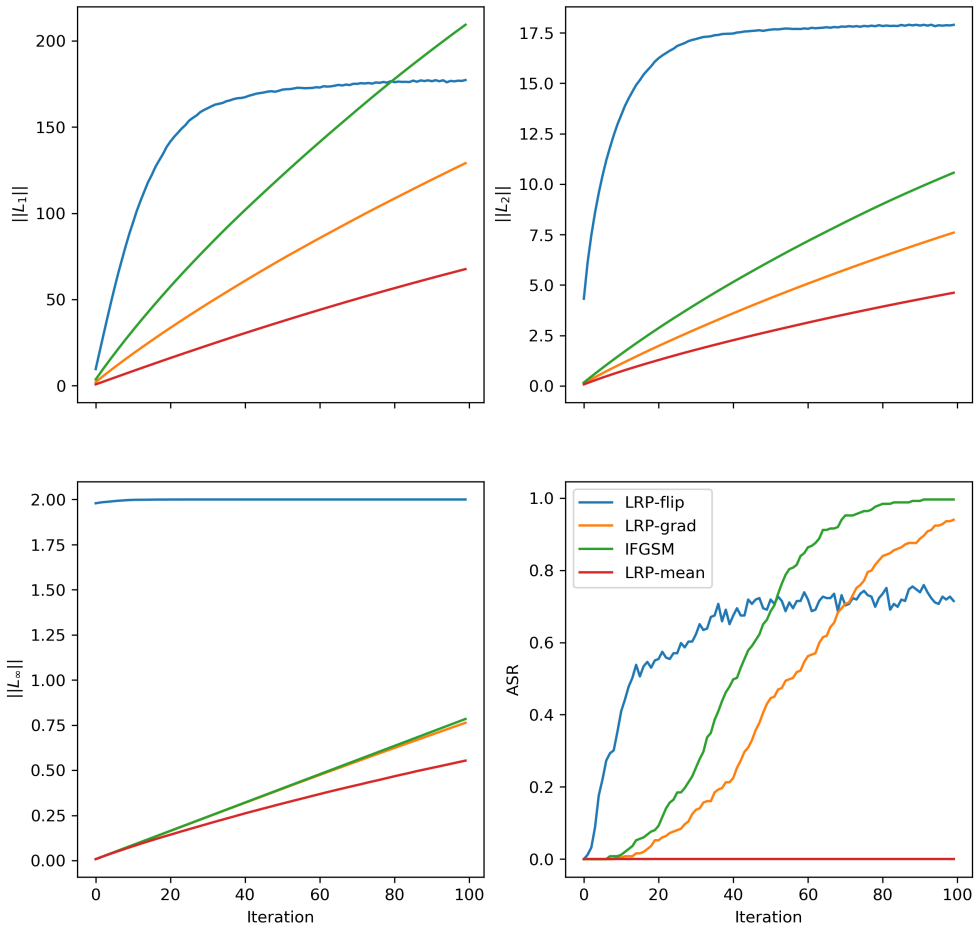


Figure 4.6: L-norms and ASR of all attack methods over the iteration count on the MNIST dataset.

ment seems to occur. At about 40 iterations, the IFGSM method outperforms LRP-flip, reaching a nearly 100% ASR after 100 iterations. The LRP-grad method exhibits the second slowest increase, however overtaking LRP-flip after about 60 iterations and not reaching an upper limit after 100 iterations. Lastly, the LRP-mean method does not show any success on the MNIST dataset. A hypothesis is that the MNIST images mostly contain fully black pixels with a value of -1 or white pixels with a value of 1. Since pixel values that are closer to 0 are not common in the dataset and thus not part of the training distribution, the CNN might not be sensitive to perturbations that are only directed towards a pixel value of 0.

To evaluate the effectiveness, i.e., ASR over L-norms, of each method, the results of Figure 4.6 were postprocessed and plotted with the L-norms as a function of ASR. For all other datasets, this representation of the experiment results will be used.

From the first graph in Figure 4.7, it can be seen that for the L_1-norm of 0 to about 50, the LRP-flip method exhibits the highest ASR. However, for all higher L_1-norm values,

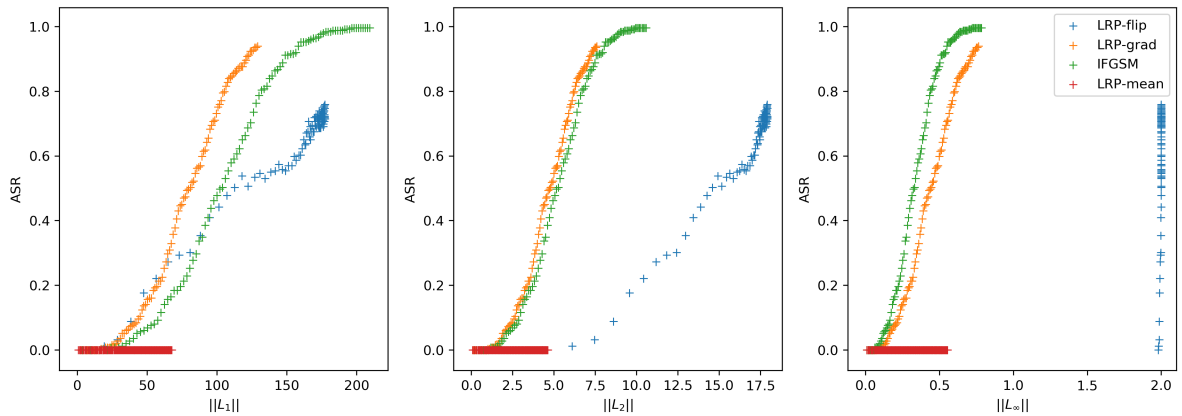


Figure 4.7: L-norms of all attack methods over ASR on the MNIST dataset.

LRP-grad has the highest ASR compared to all other methods. From an L_1 value of about 95 and higher, IFGSM also outperforms LRP-flip.

The second graph depicting the L_2-norms shows both the LRP-grad and IFGSM method outperforming LRP-flip. The LRP-grad method shows slightly better performance than IFGSM, however both performing similar.

In the third graph of Figure 4.7, IFGSM performs best in the L_∞ comparison, with LRP-grad second and the LRP-flip method exhibiting the worst results.

The overall highest ASR on the MNIST dataset was achieved by the IFGSM method with an ASR of 99.6%.

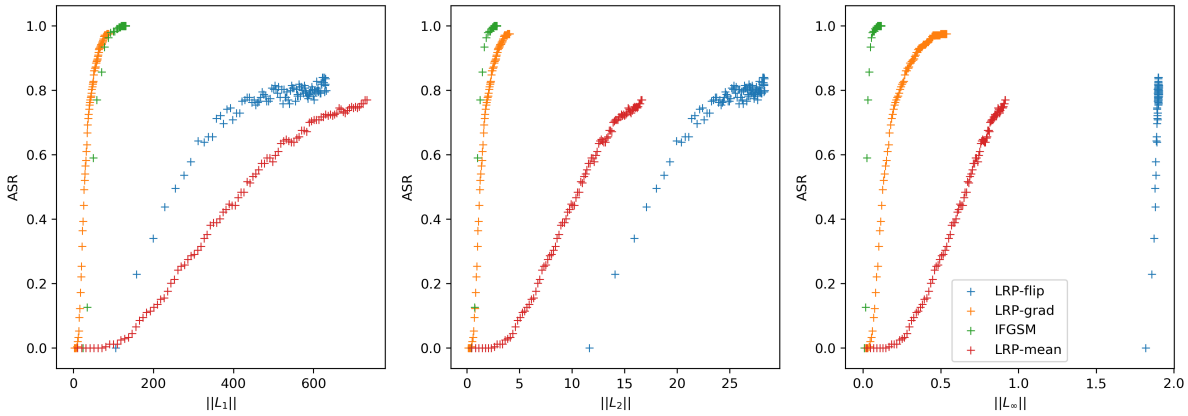


Figure 4.8: L-norms of all attack methods over ASR on the CIFAR10 dataset.

The results of applying the attack methods on the CIFAR10 dataset are depicted in Figure 4.8. In the first graph showing the L_1-norm and ASR, the best performing attack method is LRP-grad, closely followed by IFGSM. Both methods reach an ASR greater than 95% while staying at an L_1-norm below 200. The LRP-flip and LRP-mean methods perform considerably worse, with LRP-flip performing better than LRP-mean. These

methods reach an ASR of about 80%, however the L_1-norms reach values higher than 600.

In the second graph showcasing the L_2-norm and ASR, IFGSM and LRP-grad again exhibit the best performance and stay at an L_2-norm below 5, with IFGSM slightly outperforming LRP-grad at a higher ASR. The LRP-mean method exhibits the third best performance, reaching an 80% ASR at an L_2-norm of 15. Lastly, the LRP-flip method reaches its highest ASR of 80% at L_2-norm values above 25.

The third graph showcasing the L_∞-norm shows the IFGSM method performing best, followed by LRP-grad and then LRP-mean. Again, the LRP-flip method exhibits the worst performance in L_∞, due to the pixel flipping leading to high single pixel perturbations.

The overall highest ASR on the CIFAR10 dataset was achieved by the IFGSM method with an ASR of 100%.

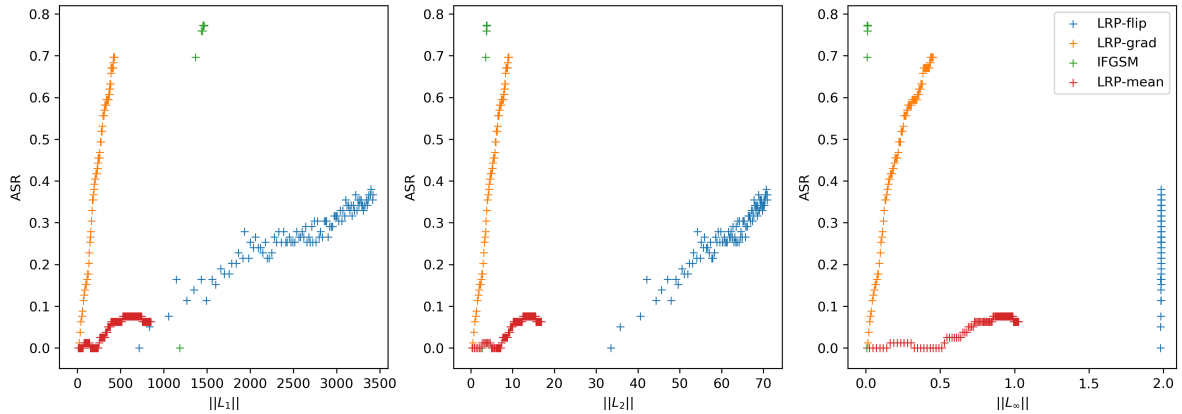


Figure 4.9: L-norms of all attack methods over ASR on the CALTECH256 dataset.

The first observation to be made on the graphs in Figure 4.9 is that none of the attack methods reaches an ASR greater than 80% on the CALTECH256 dataset. Nonetheless, in the first graph showing the L_1-norm, LRP-grad shows the best performance, reaching an ASR of 70% at an L_1-norm of about 500. The second best results were achieved by IFGSM with an ASR of 77% at an L_1-norm of 1500. LRP-flip achieves an ASR of 40%, however reaching the highest L_1-norm of 3500. Last, the LRP-mean method does not achieve a higher ASR than 10% at an L_1-norm of 1000.

The second graph depicting the L_2-norm shows a very similar picture, however IFGSM performs better than LRP-grad and reaches the ASR threshold at lower L_2-norm values. Finally the third graph shows the L_∞-norm, in which IFGSM performs best, followed by LRP-grad. The L_∞-norm of LRP-mean reaches its maximum of 1 at an ASR of 10%, the LRP-flip method instantly exhibits L_∞-norm values of 2 after the first iteration.

The overall highest ASR on the CALTECH256 dataset was achieved by the IFGSM method with an ASR of 77.2%.

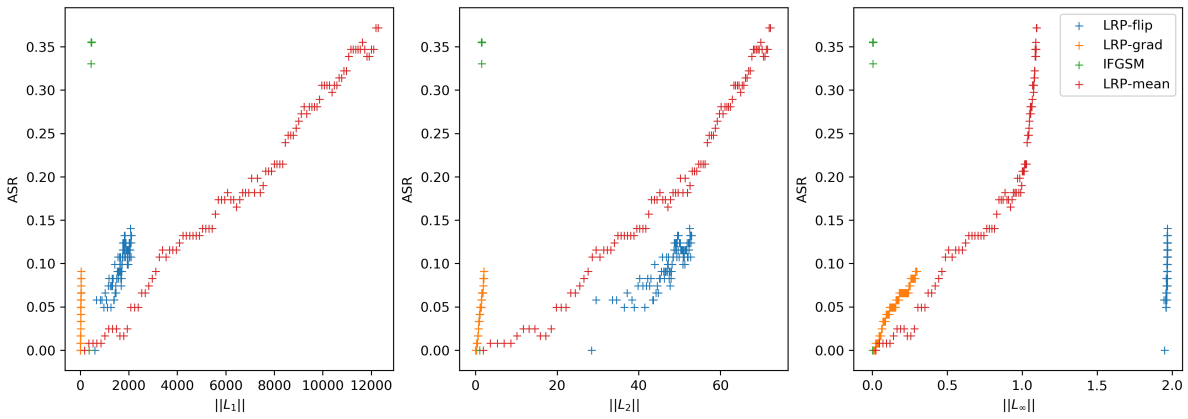


Figure 4.10: L-norms of all attack methods over ASR on the dog vs. cat dataset.

In the first graph of Figure 4.10, showcasing the dog vs. cat results, no method reaches an ASR higher than 37%, however IFGSM reaches its ASR maximum of 35.5% at the lowest L_1-norm value, with a maximum L_1-norm of 450. Also, the LRP-mean method reaches an ASR of 37.2%, it does however reach very high L_1-norm values of 12000. LRP-mean and LRP-flip do not reach an ASR higher than 15% while staying at an L_1-norm lower than 2000 and at all points showing a higher ASR than LRP-mean at the same L_1-norm values.

The second graph shows similar results, however LRP-flip does reach lower ASR values than LRP-mean at the same L_2-norm.

In the third graph showing the L_∞-norm, IFGSM again shows the best results, with LRP-mean exhibiting better ASR values at low L_∞-norms than LRP-mean. LRP-mean however reaches its threshold at an L_∞ of 1, at which it also reaches its ASR maximum. The LRP-flip method exhibits the same results as in Figure 4.9, however reaching lower ASR.

The overall highest ASR on the dog vs. cat dataset was achieved by the LRP-mean method with an ASR of 37.2%.

4.2.4 Analysing the Missing Gradient in the CALTECH256 and dog vs. cat Models

When analysing the ASR over L-norm graphs in Figures 4.9 and 4.10, two observations can be made: the IFGSM method reaches its highest ASR after a small number of iterations and the LRP-mean method does not seem to reach an upper limit within the experiment parameters. This leads to the hypothesis that the ASR ceiling might be related to gradients.

To investigate this, the initial gradient of the loss with regards to the input according to equation 2.30 of each image per dataset is evaluated. The percentage of gradients containing only zeros is calculated, leading to the following Table 4.2.

Table 4.2: Overview of the fraction of initial images with a zero gradient according to equation 2.30 as well as the highest achieved ASR and the respective method.

Dataset	Fraction of zero gradients	max(ASR)	According Method
MNIST	0%	99.6%	IFGSM
CIFAR10	0%	100%	IFGSM
CALTECH256	19.5%	77.2%	IFGSM
dog vs. cat	60.9%	37.2%	LRP-mean

As can be seen in Table 4.2, the datasets MNIST and CIFAR10, with their respective models for which an ASR of 99% or higher was achieved by IFGSM, no missing gradients were detectable. In the CALTECH256 and dog vs. cat dataset and models however 19.5% and 60.9% of images had missing gradients. While vanishing [31, 71] or exploding [58] gradients are a well studied topic when training neural networks, encountering these issues during adversarial attacks is not as thoroughly researched.

However, two methods to circumvent the vanishing gradients were explored:

- Returning random noise as gradient, when vanishing gradients are encountered
- Separating the final dense layer from the softmax activation for the output

A method to deal with vanishing gradients during model training is injecting random noise during backpropagation [7, 61, 55]. Since adversarial attacks are applied to already trained models, the noise must be injected during calculation of the perturbation or the adversarial pattern. In this work, the function creating the adversarial patterns was modified to apply a random noise pattern if a vanishing gradient is detected by evaluating, whether the sum of all elements of the gradient matrix is equal to zero. The random noise is created using a Gaussian noise generator with a mean of 0 and a succeeding sign operation, as can be seen in algorithm 8.

input : Θ, x, f, original class, $n_{iterations}$
for $i \in (1, \ldots, n_{iterations})$ **do**
  Calculate A by $\text{sign}(\nabla_x J(\Theta, x, y))$
  if $sum(A) = 0$ **then**
    $A = \text{sign}(\text{noise}(\sigma = 0))$
  end
  Multiply $\epsilon \cdot A$ to calculate η
  Add perturbation η to x
end
output: $\tilde{x}$

Algorithm 8: Adapted IFGSM with random noise injection when vanishing gradients are identified.

While applying this method to the CALTECH256 and dog vs. cat dataset did reveal gradients after repeated iterations, it did not lead to overall improved results and further work should explore this approach.

As a second approach, the softmax activation was separated from the final dense layer in the neural network architecture. It is known that the choice of activation function influences the vanishing gradient [32]. When vanishing gradients are encountered due to numerical instability, the softmax and sigmoid functions are known to intensify these issues. Similarly to the random noise injection however, this approach did not lead to immediate improved results and should be considered for further analysis on this topic.

4.3 Evaluating the Robustness of LRP-Based Attack Methods against Defence Techniques

With the increasing research and improvement of adversarial attacks, defensive measures are also developed and applied against adversarial attacks [3]. Here, adversarial transfer training will be applied to the classifier for the CIFAR10 dataset in order to evaluate the robustness of the LRP-based attacks against defensive measures.

4.3.1 Setup for Adversarial Training

The evaluation of the robustness against adversarial training is done for the best-performing LRP-based attack based on the benchmarking results. Thus, the LRP-grad attack will be evaluated on the CIFAR10 dataset.

Training data images are sampled from each class and then adversarial attacks on those samples are created using $n_{batch} = 250$ and $n_{iter} = 100$. After 1000 adversarial images have been created, the original model is transfer learned on a training set containing the 1000 adversarial samples and 9000 original samples for 50 epochs with a reduced learning rate. Examples for these adversarial images are depicted in Figure 4.11. Then, the same 250 test images that were used for the initial adversarial attacks are again perturbed using $n_{batch} = 250$ and $n_{iter} = 100$ and the LRP-grad method. Finally, two metrics are calculated:

- The ASR of the transfer learned classifier on the adversarial images based on the original classifier
- The ASR of the transfer learned classifier on the newly created adversarial images

While the first metric shows if the classifier can withstand the previously created adversarial images better, the second metric will show whether the LRP-grad method can circumvent adversarial training or if adversarial training is a valid defence mechanism against this attack as well.

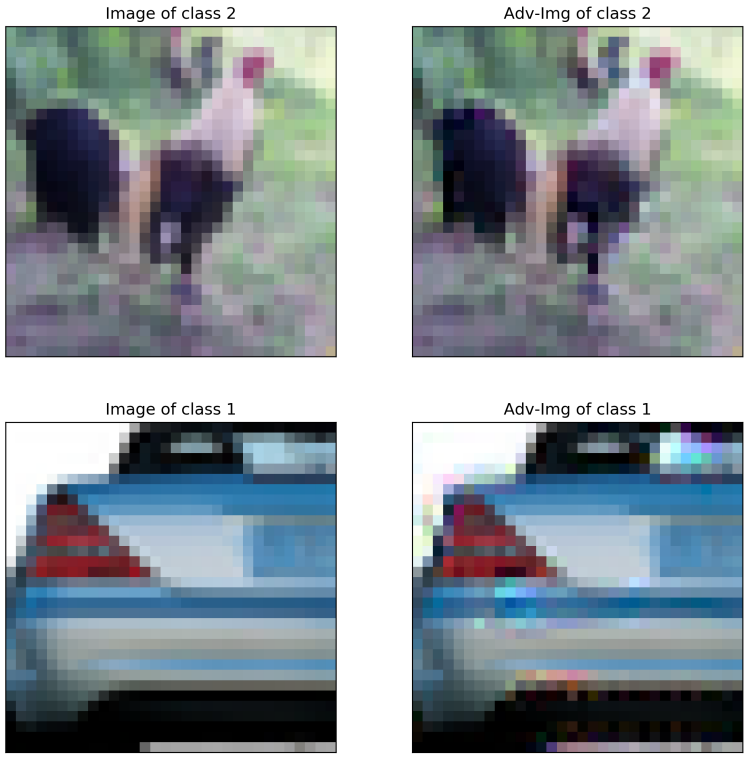


Figure 4.11: Examples from the generated adversarial images utilizing LRP-grad.

4.3.2 Results of Repeated Adversarial Attacks after Adversarial Transfer Learning

After conducting the adversarial attacks and transfer learning, the first analysis can be made on the difference in accuracy between the original classifier C_o and the transfer learned classifier C_t. This can be computed on both the images used for transfer learning in their original state X_t and after perturbation $\tilde{X}_t$.

Table 4.3: Accuracy of C_o and C_t on X_t and $\tilde{X}_t$.

	C_o	C_t
X_t	97.1%	95%
$\tilde{X}_t$	0.2%	53.1%

The first observation to make is that the original classifier maintains the higher accuracy of 97.1% on the original images while the transfer learned classifier achieves an accuracy of only 95%. Secondly, while C_o has an accuracy of 0.2% on the adversarial images, the transfer learned C_t exhibits an accuracy of 53.1% on $\tilde{X}$. This however is to be expected, since C_t was specifically transfer learned on those images.

In a second step, the same 250 images X_1 that were perturbed ($\tilde{X}_1$) for Figure 4.8 were

evaluated against C_t, which led to the following results:

Table 4.4: Accuracy of C_o and C_t on X_1 and $\tilde{X}_1$.

	C_o	C_t
X_1	97.6%	87.2%
$\tilde{X}_1$	0.2%	19.2%
ASR($\tilde{X}_1$)	97.5%	78%

While the performance of C_t on the original images X_1 is reduced by more than 10% to an accuracy of 87.2%, the performance on the initially perturbed images $\tilde{X}_1$ is increased to 19.2%. This shows that the transfer learning seems to have generalized enough so that one fifth of the initial LRP-grad generated adversarial images are classified correctly.

5 Discussion and conclusion

We discuss the performance and applicability of LRP-based adversarial attacks in comparison to state-of-the-art methods. To this end, we compare LRP-grad and LRP-mean with IFGSM to outline the differences and critique our contribution. Next, we outline three strands of future work that build on our results. Last, we conclude with an outlook of the implications of explainable AI-based attacks and mitigation approaches on the future development in AI.

5.1 Utilizing LRP for Adversarial Attacks

While Bach et al. showed in [6] with the LRP-flip method that layerwise relevance propagation does succeed in highlighting the most relevant pixels for a classification, their work did not aim towards proposing a new adversarial attack method. Novel attack methods introduced in this work are the LRP-mean and LRP-grad attack method. The proposed algorithms in Section 3.2 provide a clear definition for these adversarial attacks based on LRP and how adversarial patterns can be designed using a relevance matrix.

5.1.1 Adversarial Patterns or Global Perturbation

From a theoretical perturbation perspective, the evaluation in Section 4.1.3 showed that the proposed methods LRP-grad and LRP-mean are guaranteed to introduce a lower perturbation per attack iteration than IFGSM due to them utilizing adversarial patterns that only impact a subset of the image. This more selective nature of the LRP-based attacks is advantageous in two perspectives: only those parts of the image are manipulated that have a high contribution to the classifier output, resulting in a more precise perturbation vector. Also the more frequent calculation of the adversarial pattern due to higher iteration count combined with a smaller L-norm perturbation per iteration can enable more effective early stopping of adversarial attacks.

To showcase the difference in resulting adversarial images, the image used for Figures 3.13, 3.15 and 3.16 was again perturbed using IFGSM and the resulting adversarial images for IFGSM, LRP-mean and LRP-grad are shown in Figure 5.1.

Comparing the adversarial images in Figure 5.1 regarding their pattern of perturbation, a subjective opinion could be that the global perturbation of IFGSM creates an obvious distortion in the adversarial image. The perturbation in the adversarial image of the LRP-mean method seems more organically disturbed due to the averaging characteristic towards a dark grey and the perturbance focussing on one area in the image rather than

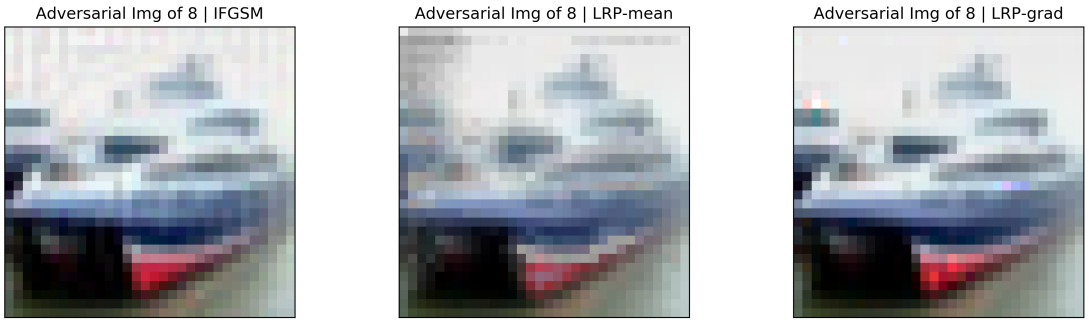


Figure 5.1: An image of class 8 of the CIFAR10 dataset is perturbed using IFGSM (left), LRP-mean (middle) and LRP-grad (right).

a global perturbation. In the LRP-grad based adversarial image, large parts of the image are unchanged (e.g., the sky in the background), making the perturbation possibly less obvious. However, these observations are of pure subjective nature.

5.1.2 Results of the Benchmark against IFGSM

Based on the experiments in Section 4.2.3, it can be concluded that the LRP-based attack methods exhibit distinct advantages compared to IFGSM.

The first observation is that LRP-grad exhibited lower L_1-norms than IFGSM on MNIST, CIFAR10 and CALTECH256. For the dog vs cat dataset, no valid comparison can be made, since the vanishing gradients prevented both methods from producing stable results. The following Table 5.1 compares the L-norms of IFGSM and LRP-grad at the highest comparable ASR datapoint for these datasets:

Table 5.1: Comparing L-norms of IFGSM and LRP-grad on MNIST, CIFAR10 and CALTECH256.

Dataset	$\|\|L_1\|\|$		$\|\|L_2\|\|$		$\|\|L_\infty\|\|$	
	IFGSM	LRP-grad	IFGSM	LRP-grad	IFGSM	LRP-grad
MNIST (ASR 94%)	158.16	129.06	8.03	7.60	0.55	0.76
CIFAR10 (ASR 98%)	92.74	83.49	1.99	3.85	0.06	0.50
CALTECH256 (ASR 70%)	1365.25	417.63	3.55	8.90	0.01	0.43

The first column shows that LRP-grad achieves the ASR at a lower L_1-norm than IFGSM for all three datasets and classifiers. In MNIST, the L_1-norm perturbation of LRP-grad is 18.4% , in CIFAR10 10% lower and in CALTECH256 60.4% lower. Thus, LRP-grad performs better than IFGSM in an L_1-norm constrained attack task, making the novel attack method a better choice in such cases.

However, when analysing the L_2-norms, it can be seen that LRP-grad outperforms IFGSM only on MNIST. This can be explained by the way, the L_1- and L_2-norms

are defined. While the L_1-norm weighs all perturbed dimensions the same, the L_2-norm emphasizes single high perturbation due to its quadratic function, which can be seen in equation 2.27. The more selective process of LRP-grad purposely modifies only the most relevant pixels, which leads to successive manipulations of the same pixel. Additionally, these observations are aligned with the hypothesis based on the theoretical analysis based on Figure 4.1. This analysis showed that the n_{batch} and n_{iter} parameters increase the perturbation steeper in an L_2-norm evaluation than in a L_1-norm evaluation.

Consequently, IFGSM outperforms LRP-grad in the L_∞-norm on all datasets. For the L_∞-norm, since only the highest single perturbation is taken into account, LRP-grad is in a way designed to perform worse in this metric than IFGSM.

One further advantage of the LRP-based adversarial attacks was found when analysing the missing gradients in Section 4.2.4. Gradient based adversarial attacks like IFGSM can be defended against using gradient masking or fail when no gradient can be computed due to high saturation, like it happened for the dog vs cat dataset and parts of the CALTECH256 dataset. The LRP-mean and LRP-flip adversarial attacks however can work if no gradient is available, as was shown on the dog vs cat dataset, which can be seen in Figure 4.10. This property can allow the LRP-mean and LRP-flip attacks to be successful, when gradient-based methods fail. However, in the experiment conducted in this work, the resulting perturbations especially from the LRP-mean method was considerably higher compared to the other methods.

5.2 Discussion and Future Work

To support in guiding further research initiatives, the most relevant open topics encountered in this work that could not be followed through are presented.

5.2.1 Improving the Proposed Attack Methods

First, further possible improvements on the LRP-based attacks themselves will be proposed. One method to improve adversarial attacks is the introduction of momentum [17]. This momentum can be introduced in the form of a projected gradient descent [12], which is a method stemming from convex optimization. Introducing momentum can improve the behaviour of an adversarial attack in local minima [17], which in turn could be one method to prevent the vanishing gradients that were encountered during the experiments in this work.

Another approach to further improve the LRP-based attacks could be building a composite attack. In this composite attack, the advantageous characteristics of the LRP-grad and LRP-mean attacks could be combined. This would mainly result in an adaptation of the algorithms 6 and 7 to incorporate an if-statement that checks for vanishing gradients and deploys the LRP-mean attack, if one is detected. Thus the advantage of LRP-mean, i.e., not relying on gradients, would be combined with the better performance of LRP-grad when a gradient is existing.

A further area of improvement could be improving the performance of the adversarial attacks. Ideally, all calculations for machine learning applications should be created utilizing hardware acceleration like CUDA. However, for the LRP-based adversarial attacks in this work, enabling hardware acceleration was not fully considered. While no comprehensive comparison of the computation time was conducted, for one randomly chosen CALTECH256 example, the unoptimized LRP-grad attack took 219.8 seconds, while the IFGSM attack utilizing tensorflow based gradient functions took 2.04 seconds.

Finally, the impact of LRP hyperparameters on the effectiveness of the adversarial attack should be evaluated. In Section 3.1.4, the influence of the LRP process on the resulting heatmap was presented. With this information, a benchmarking of different LRP processes should be conducted to confirm that the suggested composite LRP process in [52] should also be used for conducting adversarial attacks. Additionally, this benchmark could provide an objective measure to define an optimal LRP process under the hypothesis that a comparison of L-norm over ASR performance between LRP processes will identify the objectively best LRP process also for strict explainable AI applications. Also different choices for parameters a and b should be evaluated regarding their impact on the adversarial attack's performance.

5.2.2 Further Evaluation of Attack Performance Using Surveys

Even though the LRP-based attack methods were analysed regarding their L-norm over ASR performance, this pure mathematical comparison does not take into account whether or how easily the different attack methods are detectable by humans. To follow up on this, a survey should be conducted, consisting of the following steps:

1. Perturb a set of images by IFGSM and LRP-based adversarial attacks
2. Build a randomized set of original and perturbed images
3. Apply automated detection methods like feature squeezing
4. Have survey participants evaluate if an image was perturbed
5. Analyse the fraction of detected images per method

This survey can lean on previous work like presented by Zhou et al. in [83].

5.2.3 Extending the Support of LRP-Based Attacks on further Neural Network Types and Applications

While this work focussed on adversarial attacks on CNNs, more specifically on CNNs used for image recognition tasks, as introduced in Section 2.1.1 there are numerous fields in which neural networks are utilized. To transfer the adversarial attacks to these fields, some additional steps need to be done.

First, the implementation of LRP should be extended to include further layer types of neural networks that are used in other fields than image recognition. Additionally, further LRP-rules should be implement to allow more flexibility in defining the LRP

process for each explainer. It was shown by Xu et al. in [34] that LRP-based attacks on text classification models can be successful by introducing perturbations in the form of misspellings to generate adversarial text. Since text manipulation is different to image manipulation, as small perturbations like introducing misspellings can be easily recognized by humans, it is an interesting and difficult task to create adversarial text.

5.3 Conclusion

The initial assumption that explainable AI methods not only provide an explanation for decisions of neural networks, but also reveal vulnerabilities, was confirmed. Not only can the relevance matrices created by LRP be used for adversarial attacks, but they also enable novel attack methods like LRP-grad that outperform IFGSM in an ASR over L_1-norm evaluation. This highlights the impact that explainable AI has and will have on the development of artificial intelligence. Since the adversarial attacks that we analysed are white-box attacks that imply full access to the neural network however, these results do not imply that any software utilizing machine learning can easily be manipulated.

The takeaway is that researchers and developers utilizing artificial intelligence should be aware of explainable AI methods and the specific attack surfaces of neural networks. Furthermore, with this knowledge these vulnerabilities can be mitigated, as was shown when we applied adversarial transfer learning. Due to the fundamental changes to our lives that can be enabled by systems utilizing artificial intelligence, like for example autonomous driving or intelligent assistants, the community working on artificial intelligence has the responsibility to be diligent about the security of these systems. We aim to contribute a piece in the puzzle of making widespread adoption of artificial intelligence safer and more resilient in the presence of adversaries.

A Appendix

A.1 Relevance Matrices

A.1.1 Altering the Relevance Target in LRP

Because the adversarial attacks in this work are created with the goal of untargeted misclassification, all heatmaps in chapters 1 to 5 were created using $c_t = c_a$. However, when using different c_t for the LRP calculation, a relevance heatmap is created showing the relevance for classes other than the original label. When analysing the heatmaps in figure A.1, it can be seen that for each different c_t, different parts of the image show positive or negative relevances. An interpretation of the heatmap with $c_t = 1$ could be, that the vertical part of the seven is also an indicator for a one, however the horizontal upper part of a seven is an indicator that the image does not depict a one. Thus the relevance values in these areas are positive (vertical part) and negative (horizontal part).

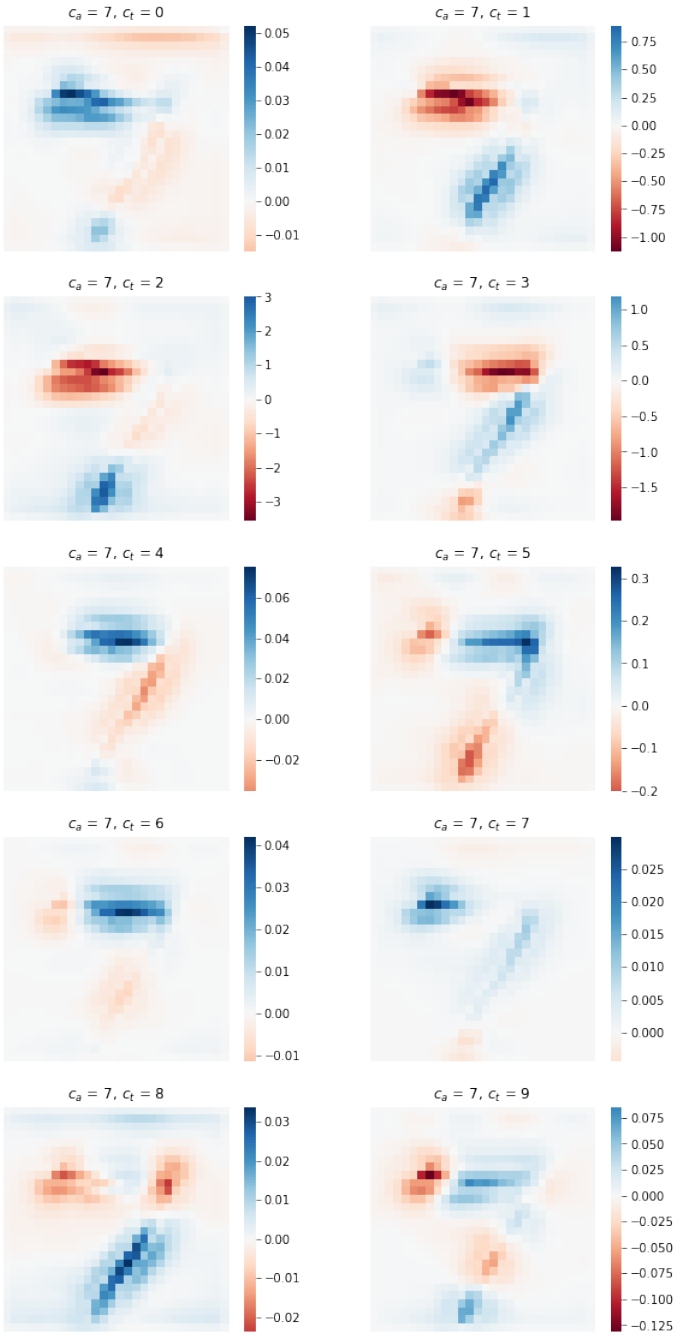


Figure A.1: Relevance matrices for the input image as shown in figure 3.6 for all 10 MNIST classes as c_t.

A.1.2 Changes in Relevance Heatmaps after Adversarial Training

One topic in explainable AI is the stability of explanations on repeated executions [69]. This stability is ensured with LRP due to its deterministic nature. An interesting question that arose in this work was how stable an explanation matrix is when transfer learning on adversarial inputs. The classifier changes with the transfer learning, however the assumption would be, that the same general concepts in images are recognized for each class. To gain insight into this question, we calculated relevance matrices for both the original classifier C_o and the transfer learned classifier C_t on the same inputs.

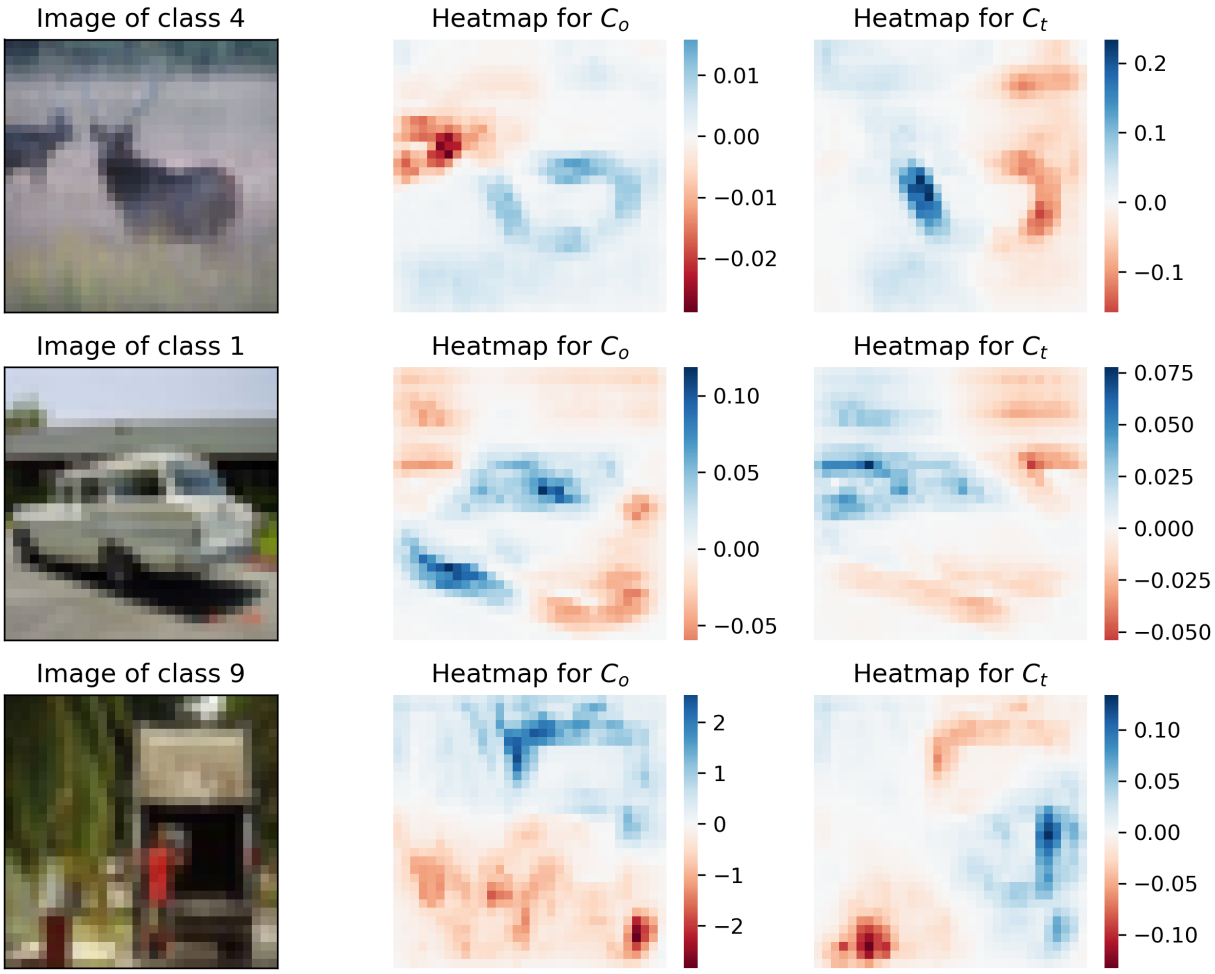


Figure A.2: Random images of the CIFAR10 dataset were sampled and the relevance matrices for the original classifier C_o and the adversarially trained classifier C_t were calculated.

In figure A.2, it can be seen that for the first two examples, similar regions of the image show positive relevance. In the third example however, the relevance heatmap is almost flipped. However, the heatmap of C_o in this case exhibits high absolute relevances, which

is an indication for numerical instability.

A.2 Adversarial Attack Examples

Here, we provide additional adversarial images of the MNIST, CIFAR10 and CALTECH256 datasets created with IFGSM and LRP-grad.

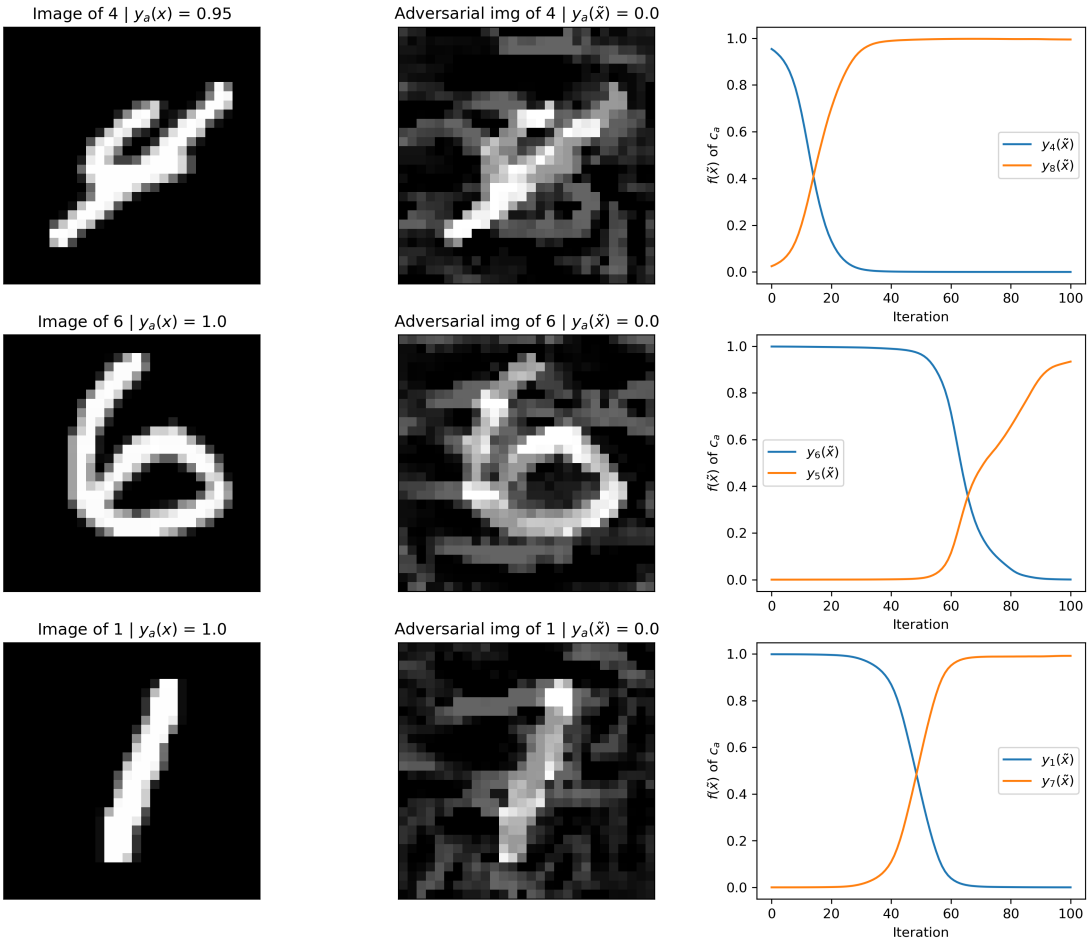


Figure A.3: Examples of adversarial images created with IFGSM on the MNIST dataset.

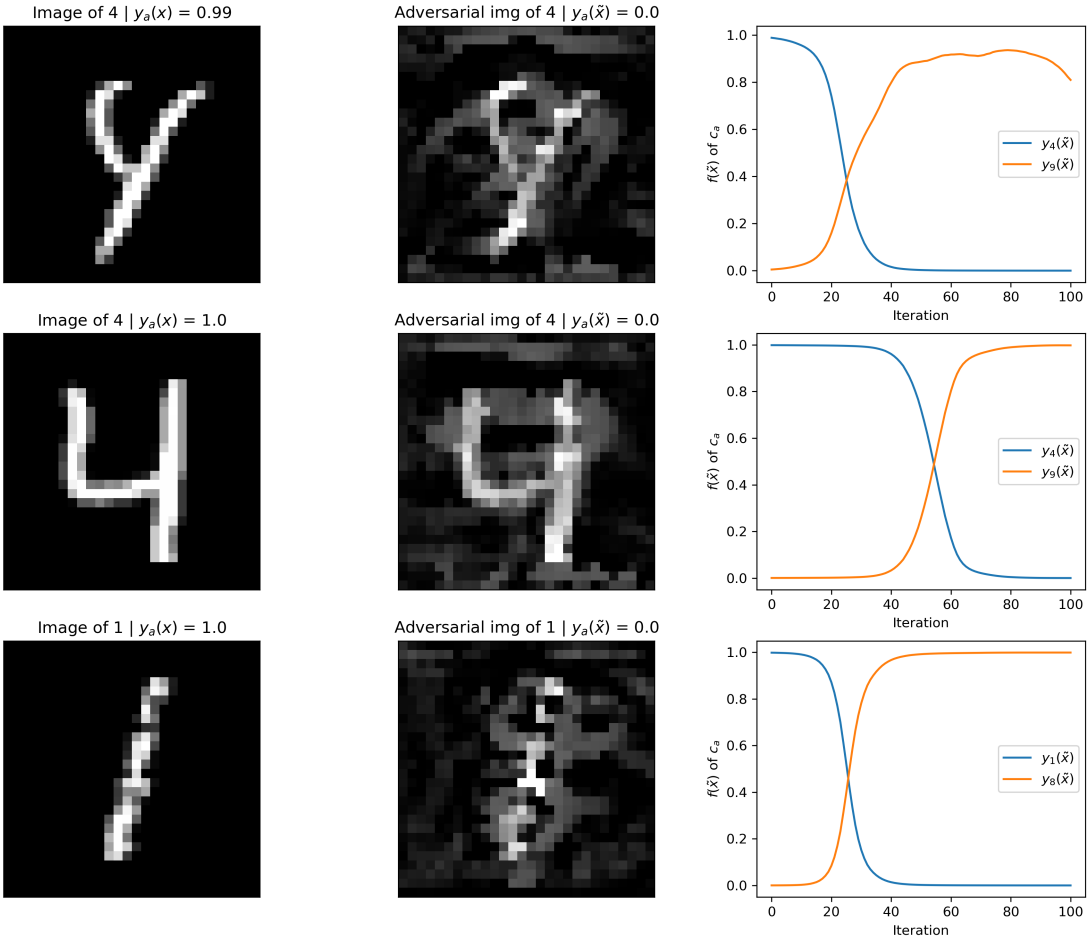


Figure A.4: Examples of adversarial images created with LRP-grad on the MNIST dataset.

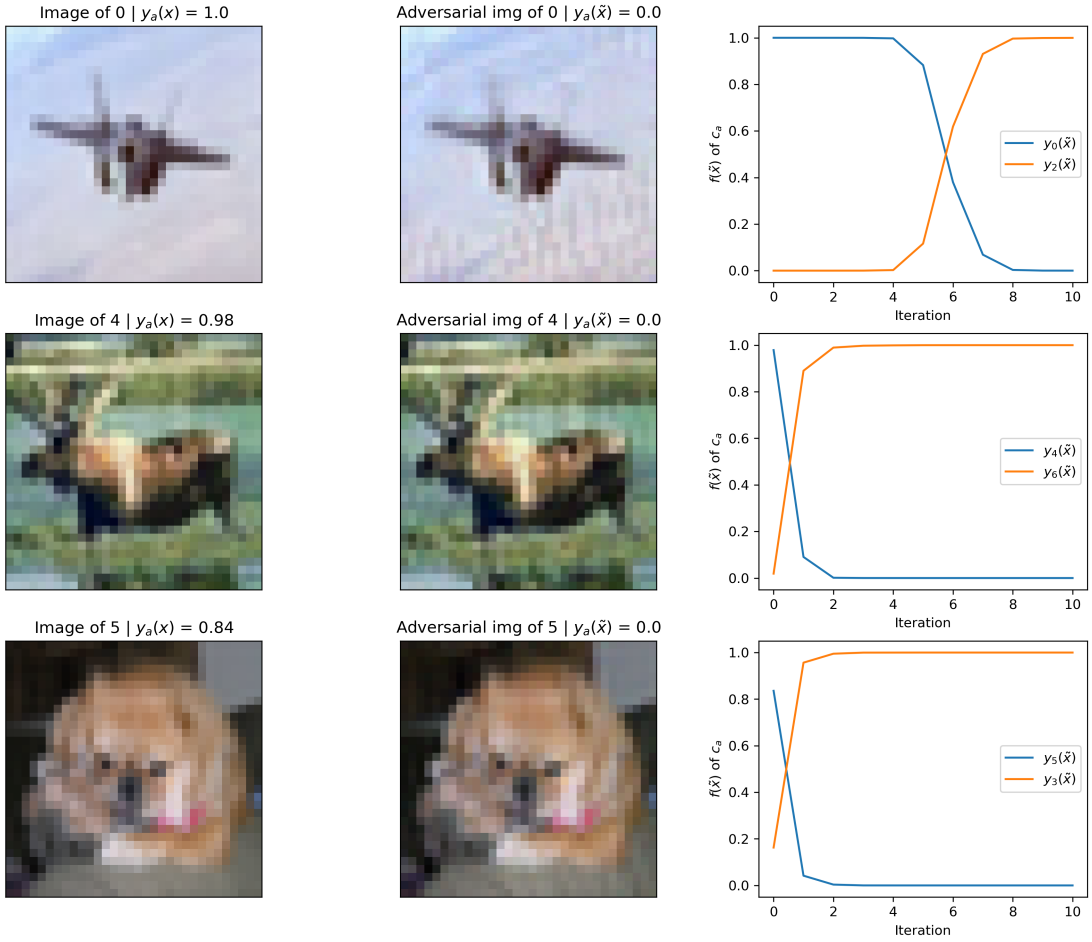


Figure A.5: Examples of adversarial images created with IFGSM on the CIFAR10 dataset.

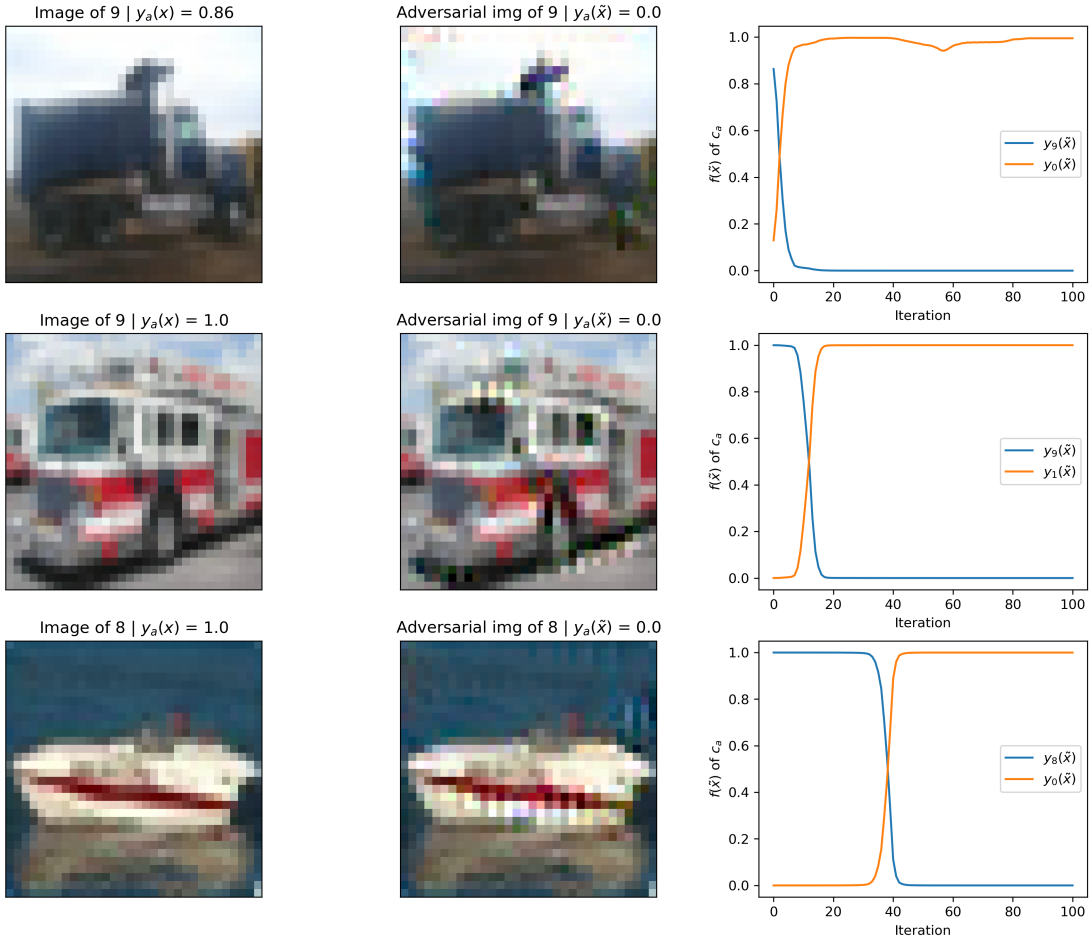


Figure A.6: Examples of adversarial images created with LRP-grad on the CIFAR10 dataset.

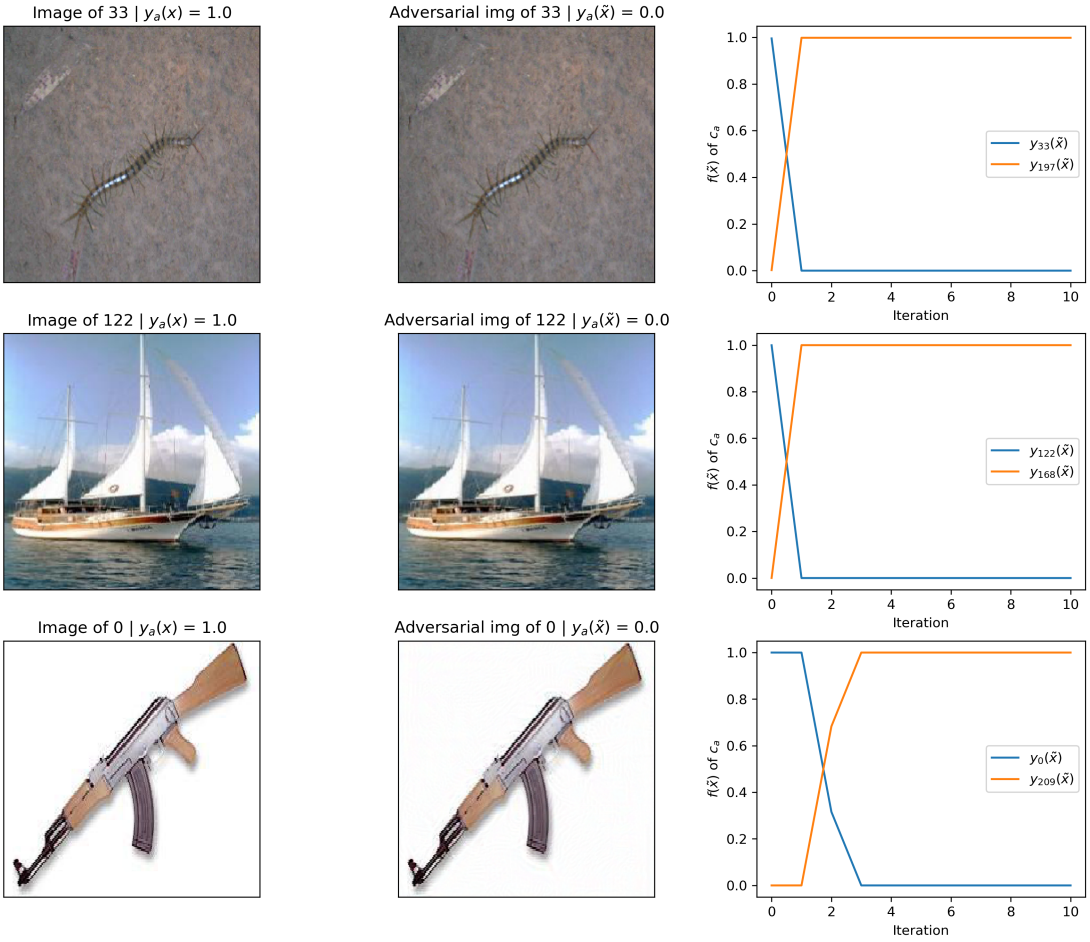


Figure A.7: Examples of adversarial images created with IFGSM on the CALTECH256 dataset.

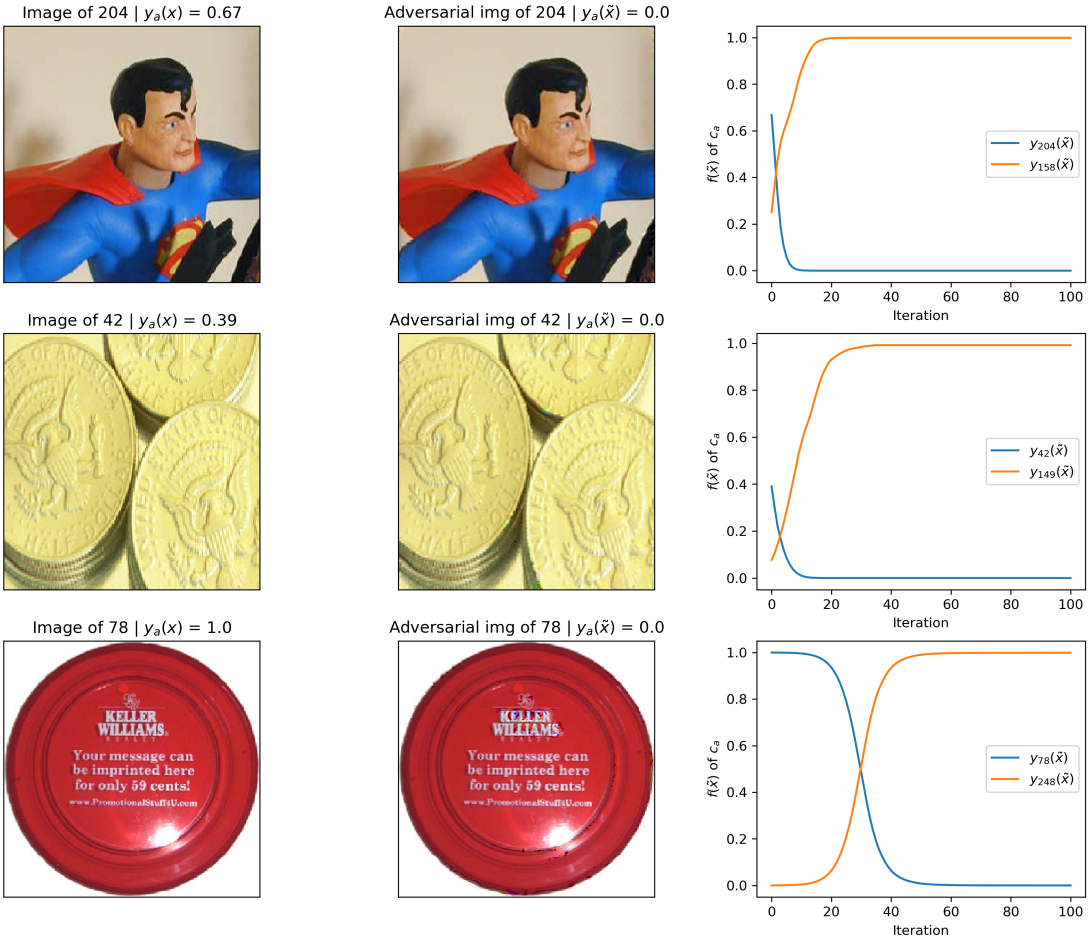


Figure A.8: Examples of adversarial images created with LRP-grad on the CALTECH256 dataset.

Bibliography

[1] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. "State-of-the-art in artificial neural network applications: A survey". In: *Heliyon* 4.11 (2018), e00938.

[2] Alessandro Achille, Giovanni Paolini, and Stefano Soatto. *Where is the Information in a Deep Neural Network?* 2020.

[3] Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. *Advances in adversarial attacks and defenses in computer vision: A survey*. 2021.

[4] Manuel Fernandez Delgado et al. *Do we need Hundreds of Classifiers to Solve Real World Classification Problems*. 2014.

[5] Ronan Hamon et al. *Robustness and Explainability of Artificial Intelligence*. 2017.

[6] Lapuschkin S et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLoS ONE* (2015).

[7] Guozhong An. "The Effects of Adding Noise During Backpropagation Training on a Generalization Performance". In: *Neural Computation* 8.3 (Apr. 1996), pp. 643–674.

[8] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. *Towards better understanding of gradient-based attribution methods for Deep Neural Networks*. 2018.

[9] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. "Interpreting Blackbox Models via Model Extraction". In: *CoRR* abs/1705.08504 (2017).

[10] Christopher M. Bishop. *Pattern Revignition and Machine Learning*. Springer Science, 2009.

[11] National Transport Safety Board. *Preliminary report HWY18MH010*. 2018.

[12] Sébastien Bubeck. *Convex Optimization: Algorithms and Complexity*. 2015.

[13] Nicholas Carlini. *A Complete List of All (arXiv) Adversarial Example Papers*. `https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html`. Accessed: 2022-02-13.

[14] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. *Adversarial Attacks and Defences: A Survey*. 2018.

[15] Ronald J. Williams David E. Rumelhart Geoffrey E. Hinton. "Learning representations by back-propagating errors". In: *Nature* 323 (1986).

[16] Ali Diba, Vivek Sharma, Ali Pazandeh, Hamed Pirsiavash, and Luc Van Gool. "Weakly Supervised Cascaded Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5131–5139.

[17] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. *Boosting Adversarial Attacks with Momentum*. 2018.

[18] Ronen Eldan and Ohad Shamir. *The Power of Depth for Feedforward Neural Networks*. 2016.

[19] *Ethical Guidelines for Artificial Intelligence*. `https://www.bosch.com/stories/ethical-guidelines-for-artificial-intelligence/`. Accessed: 2022-02-11.

[20] I. El-Feghi, Zakaria Zubi, and S. Abozgaya. "Efficient Weather Forecasting using Artificial Neural Network as Function Approximator". In: *INTERNATIONAL JOURNAL of NEURAL NETWORKS and ADVANCED APPLICATIONS* 1 (Jan. 2014), pp. 49–55.

[21] Nicholas Frosst and Geoffrey Hinton. *Distilling a Neural Network Into a Soft Decision Tree*. 2017.

[22] Sevdalina Georgieva, Maya Markova, and Velisar Pavlov. "Using neural network for credit card fraud detection". In: vol. 2159. Oct. 2019, p. 030013.

[23] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015.

[24] Alex Holub Greg Griffin and Pietro Perona. *Caltech-256 Object Category Dataset*. 2007.

[25] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. "Recent Advances in Convolutional Neural Networks". In: *CoRR* abs/1512.07108 (2015).

[26] David Gunning. *Explainable Artificial Intelligence*. 2017.

[27] Simon Hadush, Yaecob Girmay, Abiot Sinamo, and Gebrekirstos Hagos. *Breast Cancer Detection Using Convolutional Neural Networks*. 2020.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015.

[29] heatmapping.org/tutorial/. *Tutorial: Implementing Deep Taylor Decomposition / LRP*. 2017.

[30] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015.

[31] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116.

[32] Zheng Hu, Jiaojiao ZHANG, and Yun Ge. "Handling Vanishing Gradient Problem Using Artificial Derivative". In: *IEEE Access* PP (Jan. 2021), pp. 1–1.

[33] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[34] Qingfeng Du Jincheng Xu. *Adversarial attacks on text classification models using layer-wise relevance propagation*. 2020.

[35] Hilary Johnson and Peter Johnson. "Explanation facilities and interactive systems". In: Jan. 1993, pp. 159–166.

[36] John Riedl Jonathan L. Herlocker Joseph A. Konstan. *Explaining Collaborative Filtering Recommendations*. 2000.

[37] Adesesan B. Adeyemo Kamorudeen A. Amuda. *Customers Churn Prediction in Financial Institution Using Artificial Neural Network*. 2019.

[38] Rajiv Khanna, Been Kim, Joydeep Ghosh, and Oluwasanmi Koyejo. *Interpreting Black Box Predictions using Fisher Kernels*. 2018.

[39] Pang Wei Koh and Percy Liang. *Understanding Black-box Predictions via Influence Functions*. 2017.

[40] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *CIFAR-10 (Canadian Institute for Advanced Research)*.

[41] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. *Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints*. 2016.

[42] Sebastian Lapuschkin. *Opening the Machine Learning Black Box with Layer-wise Relevance Propagation*. 2018.

[43] Sebastian Lapuschkin, Alexander Binder, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. "The LRP Toolbox for Artificial Neural Networks". In: *Journal of Machine Learning Research* 17.114 (2016), pp. 1–5.

[44] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. *Unmasking Clever Hans Predictors and Assessing What Machines Really Learn*. 2019.

[45] Yann LeCun and Corinna Cortes. *MNIST handwritten digit database*. `http://yann.lecun.com/exdb/mnist`. 2010.

[46] Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. "Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model". In: *The Annals of Applied Statistics* 9.3 (Sept. 2015).

[47] Kotsiantis S Linardatos P Papastefanopoulos V. *Explainable AI: A Review of Machine Learning Interpretability Methods*. 2021.

[48] Afonso Ferreira Lorenzo Pupillo Stefano Fantin and Carolina Polito. *Artificial Intelligence and Cybersecurity*. 2021.

[49] Ernst & Young Global Ltd. *How to manage cyber risk with a Security by Design approach*. `https://www.ey.com/en_gl/consulting/how-to-manage-cyber-risk-with-a-security-by-design-approach`. Accessed: 2022-02-11.

[50] Dongyu Meng and Hao Chen. *MagNet: a Two-Pronged Defense against Adversarial Examples*. 2017.

[51] *Mercedes-Benz Principles for Artificial Intelligence*. `https://group.mercedes-benz.com/sustainability/data/ki-guidelines.html?r=dai`. Accessed: 2022-02-11.

[52] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. "Layer-Wise Relevance Propagation: An Overview". In: (2019). Ed. by Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, pp. 193–209.

[53] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. *DeepFool: a simple and accurate method to fool deep neural networks*. 2016.

[54] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. *Definitions, methods, and applications in interpretable machine learning*. Oct. 2019.

[55] Arvind Neelakantan, Luke Vilnis, Quoc Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. "Adding Gradient Noise Improves Learning for Very Deep Networks". In: (Nov. 2015).

[56] Ian E. Nielsen, Dimah Dera, Ghulam Rasool, Nidhal Bouaynaya, and Ravi P. Ramachandran. *Robust Explainability: A Tutorial on Gradient-Based Attribution Methods for Deep Neural Networks*. 2022.

[57] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. *The Limitations of Deep Learning in Adversarial Settings*. 2015.

[58] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. "Understanding the exploding gradient problem". In: *CoRR* abs/1211.5063 (2012).

[59] Alejandro Real, Fernando Dorado, and Jaime Durán. *Energy Demand Forecasting Using Deep Learning: Application to the French Grid*. Mar. 2020.

[60] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016.

[61] Matías Roodschild, Jorge Gotay Sardiñas, and Adrian Will. "A new approach for the vanishing gradient problem on sigmoid activation". In: *Prog. Artif. Intell.* 9 (2020), pp. 351–360.

[62] Matthew J. Roos. *Utilizing a null class to restrict decision spaces and defend against neural network adversarial attacks*. 2020.

[63] F. Rosenblatt. "The Perceptron: a probabilistic model for information storage and organization in the brain". In: *Psychological Review* 65 (1958).

[64] Wojciech Samek and Klaus-Robert Müller. "Towards Explainable Artificial Intelligence". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Ed. by Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. Cham: Springer International Publishing, 2019, pp. 5–22.

[65] Douglas C. Schmidt. *Google Data Collection*. 2018.

[66] Shoaib Siddiqui, Andreas Dengel, and Sheraz Ahmed. *Benchmarking adversarial attacks and defenses for time-series data*. Aug. 2020.

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

[68] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015.

[69] Kacper Sokol and Peter Flach. *Explainability fact sheets*. Jan. 2020.

[70] R. Sommer and V. Paxson. "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection". In: *2010 IEEE Symposium on Security and Privacy*. 2010, pp. 305–316.

[71] Hong Hui Tan and King Hann Lim. "Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization". In: *2019 7th International Conference on Smart Computing Communications (ICSCC)*. 2019.

[72] Steffen Uhlig, Kapil Nichani, Carsten Uhlig, and Kirsten Simon. "Modeling projections for COVID-19 pandemic by combining epidemiological, statistical, and neural network approaches". In: *medRxiv* (2020).

[73] European Union. *General Data Protection Regulation*. 2018.

[74] Jing Wu, Mingyi Zhou, Ce Zhu, Yipeng Liu, Mehrtash Harandi, and Li Li. *Performance Evaluation of Adversarial Attacks: Discrepancies and Solutions*. 2021.

[75] www.kaggle.com. *Dogs vs. Cats Dataset*. 2013.

[76] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. *Adversarial Attacks and Defenses in Images, Graphs and Text: A Review*. 2019.

[77] Weilin Xu, David Evans, and Yanjun Qi. "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks". In: *Proceedings 2018 Network and Distributed System Security Symposium* (2018).

[78] Roozbeh Yousefzadeh and Dianne P. O'Leary. *Auditing and Debugging Deep Learning Models via Decision Boundaries: Individual-level and Group-level Analysis*. 2020.

[79] Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013.

[80] Kui Zhao and Can Wang. "Sales Forecast in E-commerce using Convolutional Neural Network". In: *CoRR* abs/1708.07946 (2017).

[81] Bichen Zheng, Keith Paul Thompson, Sarah S. Lam, Won Jae Yoon, and Nathan Gnanasambandam. "Customers' Behavior Prediction Using Artificial Neural Network". In: 2013.

[82] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. *Object Detectors Emerge in Deep Scene CNNs*. 2014.

[83] Zhenglong Zhou and Chaz Firestone. "Humans can decipher adversarial images". In: *Nature Communications* 10.1 (Mar. 2019).