INF FACULTY OF COMPUTER SCIENCE

Otto-von-Guericke-University
Magdeburg

Master's Thesis

Towards Introspective Artificial Intelligence

Self-Assessment of Visual Recognition Systems based on Attribution

Konstantin Kirchheim

Supervisor
Prof. Frank Ortmeier
(OvGU Magdeburg)

Supervisor
Prof. Sebastian Zug
(TU Bergakademie Freiberg)

December 9, 2019

Abstract

Convolutional Neural Networks achieve state of the art results in various visual recognition tasks like object classification and object detection. While CNNs perform surprisingly well, it is difficult to retrace why they arrive at a certain prediction. Additionally, they have been shown to be prone to certain errors. As CNN are increasingly deployed into physical systems - for example in self driving vehicles - undetected errors could result in catastrophic consequences. Approaches to prevent this include the usage of attribution based explanation methods to facilitate an understanding in the systems decision in hindsight, as well as the detection of recognition errors at runtime, called self-assessment. Some state-of-the-art self-assessment approaches aim to detect anomalies in the activation patterns of neurons in a CNN.

This work explores the usage of attribution based explanations for self-assessment of CNNs. We build multiple self-assessment models and evaluate their performance in various settings. In our experiments, we find that, while self-assessment based on attribution does not outperform self-assessment based on neural activity on its own, it always surpasses random guessing. Furthermore, we find that self-assessment models using neural activation patterns as well as neural attribution can in some cases outperform models which do not consider attribution patterns. Thus, we conclude that it might be possible to improve self-assessment models by including the explanation of the model into the assessment-process.

Contents

List of Figures

List of Tables

Notation

$\hat{y}$	Label assigned to an instance
y	True label of an instance
$\Phi(x)$	Embedding function, maps input x to a vector
$c(x)$	Decision function of a classifier
$h(x)$	Hypothesis of a classifier
$R^c(x)$	Relevance of x for output neuron c
$S^c(x)$	Output of neuron c for input x
A	tensor containing the activation of the neurons of a certain layer
A^k	the k^{th} activation-map in tensor A
A^k_{ij}	the value at position (i, j) in the k^{th} activation map in tensor A

Chapter 1

Introduction

Fuelled by the exponential increase of available computational power (coupled with falling costs of hardware) as anticipated by Moore's law, and the collection of vast amounts of data, the use of machine learning algorithms has proliferated in recent years. It is expected to become ubiquitous in the future. Possible applications span every aspect of human life, including health care, manufacturing, education, financial modeling, policing, marketing, and warfare [34]. Especially when coupled with physical components in so-called Cyber-Physical Systems, these digital systems have the potential to precipitate catastrophic events in the physical world that could inflict massive damage to the economy, the environment, and society in general.

An essential technology in the area of artificial intelligence is Neural Networks, a machine learning approach loosely inspired by the functioning of biological neurons [23]. Neural network-based machine learning models constitute the state-of-the-art for tasks like natural language processing [15] and speech recognition [25]. In recent years, a specific neural network architecture, called Convolutional Neural Networks, has excelled the state-of-the-art in many applications, especially in the field of computer vision, which is concerned with tasks like recognizing or localizing objects in images [39].

1.1 Motivation

CNN-powered applications are progressively deployed in safety-critical environments in the physical world, for example, in autonomous vehicles [22]. However, even though CNNs outperform traditional computer vision approaches, they are prone to certain errors. Recent examples of recognition failures with catastrophic consequences include crashes of self-driving cars that received broad coverage by the media [66, 68].

There are several possible reasons for the failures of visual recognition systems.

CNNs are usually trained to recognize a certain set of object-types. In the real world, however, they are constantly confronted with images of objects of previously unseen classes or changing perception conditions [6]. This problem is termed the Open-World problem. Furthermore, Szegedy et al. [64] first demonstrated that it is possible to inject specially crafted perturbations into a formerly correctly classified image. While these perturbations are almost imperceptible to the human eye, they trick CNNs into making a wrong prediction with high confidence. The methods to craft such malicious input are called adversarial attacks. It has been shown that they can be employed to deceive systems in the physical world [38]. Thus, CNN-based visual recognition systems work surprisingly well but are prone to fail unexpectedly and apparently for no reason. Some people argue that these pathologies of CNNs are inherent to their architecture [21].

1.2 Existing Approaches

Since 2010, there have been attempts to formally verify properties of neural networks [48], for example, to prove their robustness to adversarial attacks [36]. Despite remarkable advancements in this field, it remains unfeasible to apply its methods to complex visual recognition tasks at present. First of all, this is due to the vast number of parameters in contemporary CNNs. State-of-the-art methods demonstrated the verification of some formal constraints - like invariance to certain distortions - on models with about 1 million parameters [36]. However, the amount of parameters in modern CNNs tends to be one to two orders of magnitude larger [58]. Secondly, there is usually no formal description of the classes of objects that should be recognized - for example, pedestrians - which a formal proof would require. This implies that the possibility of failures of these systems can not be eliminated in practice.

In general, there are two approaches to deal with these shortcomings of neural-network-based recognition systems: The “optimistic” approach aims to improve or harden the systems in order to decrease the likelihood of errors [24]. The idea is that if systems become sufficiently good at their task - for example, if they surpass human-level performance - the remaining risk of failure can be tolerated. For instance, if autonomous vehicles would cause significantly fewer accidents than human drivers, one could argue that these cars drive sufficiently good, and the risk of failure should not impede the deployment in real-world applications. The more “pessimistic” approach, on the other hand, acknowledges that there is always a substantial risk of failure, and aims to enable the systems to fail gracefully, i.e., to detect their errors and to either handle them or to fall back to a safe state [22, 26, 70, 13]. For instance, an autonomous vehicle that detects that it is unable to recognize an obstacle on the road could prompt a human operator to take control. This approach is called *self-assessment*, *introspection*

[13] or *meta-recognition* [53]. Both of the approaches mentioned above are not necessarily disjointed and subject to contemporary research.

Additionally, methods have been developed that aim to explain the decisions of CNNs in hindsight. They can be used to understand how a recognition system makes its decisions and thus facilitates the understanding of its biases. Such explanation methods can help to detect if the model is, for example, racially-biased before it is s deployed into the physical world [59].

1.3 Proposed Approach

Explanation methods are, to the best of our knowledge, not used to prevent or detect system failures at runtime. We suspect that it is possible to harness CNN explanations to detect failures of visual recognition systems at runtime. The goal of this thesis is the development of a framework that allows creating self-assessment models that can be used to test this proposition.

1.4 Scope of this Thesis

In this section, we will outline the limitations on the scope of this thesis. Since this work focuses on visual recognition systems for image classification, we will not consider other tasks - like natural language processing - where this approach could be applied as well. Furthermore, even though attribution-based explanations could potentially be used to support the estimation of arbitrary error metrics, this possibility will not be examined. We will only consider the prediction of classification errors, and not, for example, errors regarding the localization of an object in an image. We limit the examination of the idea to use explanations for the detection of failures to gradient-based attribution methods. One should keep in mind that these explanation methods only constitute a fraction of existing explanation approaches for CNNs. The goal of this thesis is explicitly not to design a failure prediction model that excels state-of-the-art but to design a model that meets the demand to gather evidence regarding the proposition. Contemplating the imposed resource limitations in terms of available time and compute, this work can merely serve as a proof-of-concept for the proposed approach.

1.5 Structure of this Thesis

In the following chapter, we will first review the literature on background information and existing approaches. Afterward, in Chapter 3, we will explain the concept in detail and bring forward arguments to support the proposition. Based on the concept, we propose a framework that allows developing self-assessment models based on attribution-based explanations. We then utilize the framework

to design a self-assessment model. In Chapter 4, we will provide an overview of the considered recognition systems and the implementation of the framework and the self-assessment model. Finally, we will conduct and evaluate experiments regarding our hypothesis and draw conclusions (Chapter 5). The thesis closes with a discussion of the flaws and limitations of the presented approach and an outlook on future research (Chapter 6).

Chapter 2

State of the Art

In this chapter, the literature relevant to the concept proposed in Chapter 3 will be presented. The survey will focus on the basic building blocks of CNNs, transfer learning, ensemble classifiers, explanation methods, reasons for failures in visual recognition systems, and approaches for the prediction of such failures. It will be briefly explained how CNNs work in general, and how they can be combined to increase their predictive performance. Afterward, the black-box character of such systems will be emphasized, and attribution based methods that aim to explain particular classifications results will be presented. Then, common reasons for classification system failures will be introduced, and eventually, methods to detect such misclassifications will be presented.

The most influence on this thesis had the work of W. Scheirer [53, 51, 52] and A. Bendale [7, 6] on failure detection and the open-set risk, D. Hendrycks [31, 30, 29] on reasons for images failures and the evaluation of robustness for CNNs and Selvaraju et al. on the GradCAM explanation method [55].

2.1 Convolutional Neural Networks

Convolutional Neural Networks are a particular neural network architecture invented in 1989 by Y. LeCun, that (later) attracted attention in the scientific community by significantly outperforming the state-of-the-art in image classification tasks [23]. As of today, this architecture is widely used and still superior to *"classic"* approaches in the field of computer vision.

Common CNN architectures are built from multiple layers, with alternating convolutional and subsampling layers[1] in front (close to the input) followed by FC layers in the back (close to the output). The convolutional and subsampling layers serve as dimensionality reducing feature extractors, while the fully connected layers perform the classification based on the extracted features. The

[1] Subsampling: e.g., Average Pooling or Max Pooling

basic building blocks of a common CNN can be found in Figure 2.1. Each of them will be explained in the following.

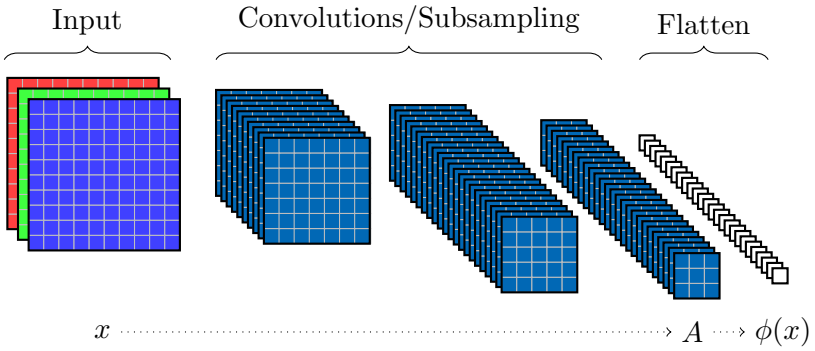


Figure 2.1: Sketch of the architecture of a typical convolutional feature embedding by convolutions and subsampling.

2.1.1 Convolutional Layers

RGB images are usually represented as three $h \times w$-dimensional matrices, where h is the image's height and w the image's width, and each matrix represents a different color channel - red, green and blue respectively. Informally, such a multidimensional array is called a tensor. Each convolutional (or subsampling) layer takes a number of matrices (i.e., a tensor) as input, performs some operation, and outputs a (possibly higher) number of smaller matrices. This operation is usually the calculation of the weighted sum of all input neurons (a linear mapping which can be expressed as matrix multiplication), followed by a non-linear activation function, which calculates the final activation of the neuron. A common non-linear activation function is the so-called rectified linear unit ($ReLU$), which is defined as

$$ReLU(x) = max(0, x) \tag{2.1}$$

Each of the resulting matrices - also called *feature-maps* or *activation-maps*[2] - can be seen as a maps that indicates the presence of a certain kind of feature. When stacking multiple convolutional layers, these detected features become more and more abstract with increasing depth of the network. While the first layers detect edges or color blobs, the deeper layers detect higher-level concepts [69]. The idea of learning hierarchical representations of data with increasing degrees of abstractions is a recurring theme in Machine Learning and the essence of Deep Learning [54].

[2]Later on, we will extract features from these maps. To prevent confusion, we will refer to these maps as *activation-maps*.

The sequential convolutions and subsampling of a CNN map the input image to a tensor A, which consists of k activation maps. This tensor contains the activation of each neuron in the last convolutional layer. When referring to the k^{th} activation map, we write A^k.

2.1.2 Flattening

Before the A can be fed into the dense layer, it is usually flattened, as the dense layers handle vectors. This flattened version of the activation-maps is called the *embedding* vector. The embeddings vector of an image x is denoted $\phi(x)$, where ϕ is called embedding function of a CNN. Contemporary architectures, e.g., [11, 63] incorporate a global average pooling layer, which reduces each of the k activation-maps to a scalar value, resulting in a k dimensional vector. The global average pooling is calculated as

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} = \frac{1}{Z} \sum_i \sum_j A_{ij}^k \tag{2.2}$$

where Z is the number of pixels in the activation-map.

2.1.3 Fully Connected Layers

The classification is then usually carried out by one or more dense layers, which calculate the final prediction with a function c of the embedding vector. The prediction $h(x)$ of the CNN is thus

$$h(x) = c(\phi(x)) \tag{2.3}$$

The final layer, when trained for N classes, usually contains N neurons and uses the softmax activation [3] which is defined as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{n=1}^{N} e^{z_n}} \quad \text{for} \quad j \in 1, ..., N \tag{2.4}$$

This activation function scales the net-inputs (i.e. the weighted sum of the inputs of a neuron) z of the N output neurons so that they sum up to one. Each output can then be interpreted as a probability for exactly one class.

For instance, consider the Xception architecture trained on the large ImageNet image dataset as provided by the deep learning library Keras, and a 299×299 RGB image which is represented as 268,203 values $\in [0, 1]$ [11, 14, 12]. The convolutional layers extract 2048 activation-maps of size 10×10, from which the final 2048 dimensional embedding vector is calculated applying global average pooling. A single FC output layer of 1000 neurons with softmax activation performs the classification. Some example activation-maps for the Vgg19 CNN are depicted in Figure 2.2.

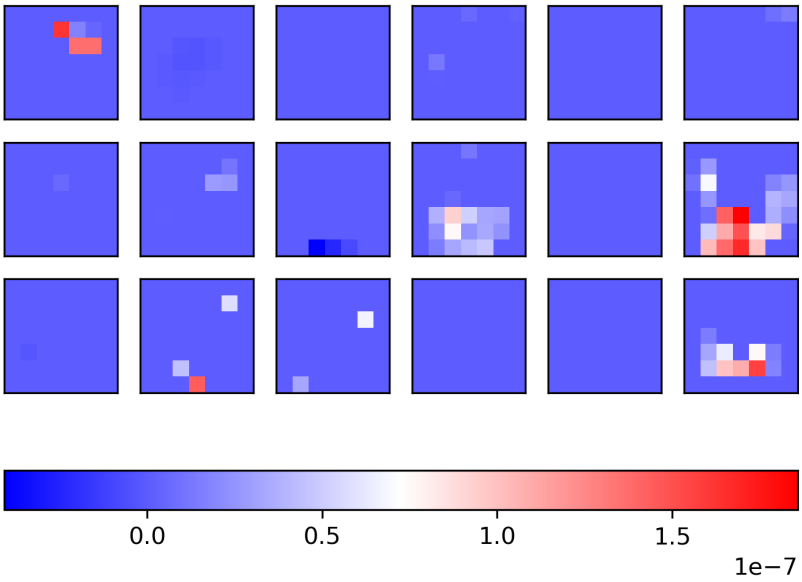


Figure 2.2: The first 7×7 pixel activation-maps $A^k \in \mathbb{R}^{7\times7}$ of the Vgg19 architecture for an example image. It is not trivial, if not even unfeasible to transfer the semantic of these activation-maps into human comprehensible concepts.

2.2 Transfer Learning

Training deep neural networks with many parameters is time and energy-consuming and can thus be expensive [61]. Additionally, DNNs require vast amounts of training data [23]. Transfer learning, in general, describes the idea to store knowledge learned in one domain and to transfer it to another [46]. When applied to DNNs, transfer learning bears the potential to reduce the required amount of time, energy, and training data that has to be invested in the training. In the following, the most common approach will be described. As elaborated in the previous chapter, a CNN can usually be divided into two major components: a convolutional part (plus flattening) which calculates the embedding vector $\Phi(x)$, and a classifier c that calculates the systems output $h(x) = c(\Phi(x))$. The concept is depicted in Figure 2.3. The basic idea for transfer learning is to use the convolutional layers as generic mid-level feature extractors that can be trained on one dataset, and only (re-)train the classifier c for other datasets. Transfer learning can also be used to adapt an existing classifier to changes in the underlying data distribution, for example, lighting conditions or viewpoint variations by fine-tuning only the higher layers [46].

Other approaches, for example, in the field of medical image analysis for automated diagnosis, replace the FC layers with other classifiers that are not based on neural networks, for example Support Vector Machines [1]. Apart from conventional supervised machine learning approaches, clustering methods can be

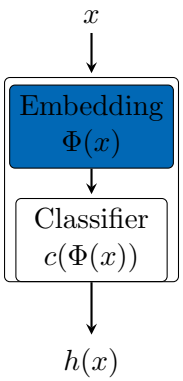


Figure 2.3: $\phi(x)$ embeds the input x into a lower dimensional vector space. This embedding vector is then fed to a classifier c, which outputs the hypothesis $h(x)$. Transfer learning can be performed by retraining only the weights of c, while keeping the weights of ϕ constant.

used to classify images based on the extracted features [42, 41], most notably k-Nearest-Neighbor or Nearest Class Mean. An advantage of clustering-based methods is that adding or removing cluster centers, or adjusting the existing ones is often computationally cheaper than to retrain the entire classifier. This facilitates adding new classes to the classifier.

2.3 Ensembles Methods

A simple yet effective approach to increase the performance of a classification system is to combine multiple classifiers into a single, more powerful recognition system [27]. In the case of neural networks, ensemble methods are also reffed to as Committee Machines [28]. There exist a plethora of ensemble methods that can loosely be grouped into static and dynamic approaches. This section aims to provide a brief overview of different ensemble techniques. However, we will only introduce the mechanisms that are relevant in the context of this thesis. Other popular ensemble methods - for example, *Boosting* - will not be considered.

In general, many ensemble techniques calculate the final prediction as a weighted linear combination of the original predictions. According to [23], some of them can also be interpreted as a regularization technique. Assuming that

1. the base classifiers make independent errors and

2. the base classifiers predictions are correct at least half of the time

it can be mathematically proven that the overall error of an ensemble will de-

crease monotonically with the number of incorporated classifiers, and theoretically becomes zero if an infinite amount of classifiers is used [27].

2.3.1 Static

Static approaches combine the output of the individual classifiers and do not include the input into their calculation.

2.3.1.1 Averaging

The averaging ensemble approach calculates a weighted linear combination of the response of all classifiers c as follows:

$$h(x) = \sum_{i=1}^{N} \alpha_i c_i(x) \tag{2.5}$$

where α is the weight of a classifier. For instance, the mean averaging ensemble approach assigns the same weight $\alpha = \frac{1}{N}$ to each classifier, where N is the number of classifiers:

$$h(x) \quad = \quad \sum_{i=1}^{N} \frac{1}{N} c_i(x) \quad = \quad \frac{1}{N} \sum_{i=1}^{N} c_i(x) \tag{2.6}$$

2.3.1.2 Majority Voting

Voting based ensemble methods calculate the final classification by determining which class was predicted by the majority of the base models. There are several types of majorities, for example, unanimous voting (100 %), simple majority voting (more than 50 %) or plurality voting (relative majority). The behavior of such ensembles in the case that no consensus could be achieved may differ. An option is, for instance, to reject the input.

From a safety perspective, majority voting has the advantage that this type of ensemble method can, by design, not introduce new errors that have not been made a majority of the base models. The averaging ensemble, on the other hand, does not ensure this [67]. However, the voting-based ensemble is also unable to correct errors that have been made by the majority of the base models, while the majority voting is.

2.3.1.3 Bootstrap Aggregating

Bootstrap Aggregation, also referred to as *Bagging*, generates random subsets of the training data (uniform with replacement) and trains a different classifier of the same type on each. For classification tasks, the final prediction is then

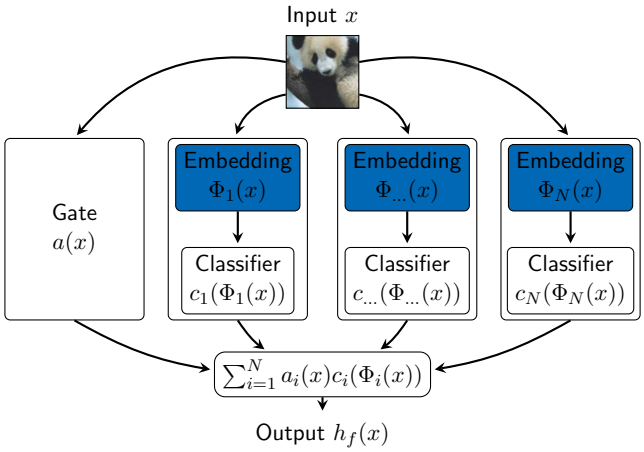


Figure 2.4: Architecture of a Mixture of Experts Model using CNNs. The output of the classifiers $c_1, ...c_N$ is aggregated in a weighted linear combination to obtain the final classification result.

obtained by a voting mechanism, as presented in 2.3.1.2. The Dropout regularization [60] for neural networks can be seen as a form of bagging with shared parameters [23].

2.3.2 Dynamic

Dynamic ensembles also often calculate a weighted linear combination of the original predictions, but contrary to static ensembles, the assigned weights depend on the input [28]. A typical dynamic ensemble approach is the so-called Mixture of Experts (MoE). The basic idea is to divide a complex problem into multiple easier subproblems and to train a classifier for each subproblem. A so-called gating-function then decides which classifier should be used, or how the responses of the individual classifiers should be combined to obtain a more accurate prediction. Thereby they follow the divide-and-conquer problem-solving paradigm [20].

An example MoE approach, called *Adaptive Fusion*, was proposed in [1]. They train multiple classifiers for image recognition, and an additional gating network a learns to combine the individual predictions based on a thumbnail of the image. The resulting calculation is as follows:

$$h(x) = \sum_{i=1}^{K} a_i(x)c_i(x) \tag{2.7}$$

The general architecture of MoE approaches is depicted in Figure 2.4.

2.3.3 Convolutional Ensembles

In 2018, a novel approach for the improvement of the performance of CNNs for image recognition tasks was developed independently by [3] and [67]. The approaches have an ensemble like character but do not fully meet the traditional definition of ensembles. These approaches employ pre-trained convolutional embedding functions and (can), therefore, be seen as a form of transfer learning. As they incorporate multiple such convolutional embeddings, the concepts are also related to ensemble strategies. However, these convolutional ensembles feed the extracted features into a single classifier and are thereby, strictly speaking, not ensembles. Thus, we will refer to them as "ConSembles", a neologism coined by Wehmeier [67]. In the following, an in-depth explanation of the ConSemble approach will be provided.

[67] suspects that most of the learning happens in the convolutional layers of a CNN. Therefore, the inferior performance of some classifiers is not caused by the FC layer but rather a result of the preceding feature extraction process. It could be possible that the convolutional layers of some of the base models learned to extract features that are more representative for some classes than for others, which would, in turn, allow a classifier to identify instances of these classes with higher accuracy.

For instance, contemplate the following simplified example: Consider two CNNs, A and B, with their respective embedding function $\Phi_A(x)$ and $\Phi_B(x)$. Let us assume that A and B were trained to decide whether an image depicts an *apple* or an *banana*. During training, model A learned to make this classification (primarily) based on the shape, so the embedding function Φ_A is essentially encoding some properties of the shapes detected in x. Model B, in contrast, (primarily) relies on the color, so its embedding function Φ_B essentially encodes properties of the colors detected in x. For the simple example of oranges and pears, each characteristic (shape or color) would probably be sufficient to achieve comparably high accuracy. However, we can anticipate that a ConSemble C that bases its prediction on both embeddings and is thus provided with more features will most likely exceed the individual classifiers A and B. A delineation of the general architecture is depicted in Figure 2.5.

2.4 Attribution based Explenations

State-of-the-art CNNs are highly complex, non-linear systems with a vast amount of parameters. An overview of the number of parameters for some example CNNs is given in Table 2.1. Due to this complexity, it is difficult to explain why a particular input resulted in the obtained response, and CNNs are often regarded as black-box systems [59]. In regards to this property, CNNs can be compared to the human brain: even though the behavior of individual

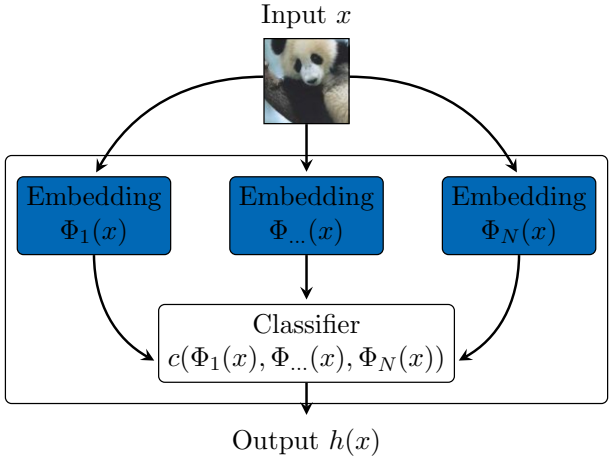


Figure 2.5: Architecture of a convolutional ensemble. The input x is processed by multiple (convolutional) embedding functions $\Phi_1(x), ..., \Phi_N(x)$. The extracted features are then fed into a single classifier c to produce the output $h(x)$.

neurons is a well-researched topic, the emergence of high-level cognitive abilities from the vast amount of interconnected neurons remains unclear.

To tackle this problem, several methods that aim to explain the output of CNNs have been developed. In the following, will focus on attribution-based explanation methods (also referred to as *sensitivity-based*, *relevance-based* or *contribution-based* methods) [4]. In general, these kinds of methods aim to ex-

Table 2.1: Instances of different neural network architectures as provided by Keras [12]. Most of the parameters of the Vgg19 are contained in the FC layers.

Architecture	**Parameters**	**Depth**	**Year**
Vgg19	143,667,240	26	2014
Inception V3	23,851,784	159	2016
Xception	22,910,480	126	2017

plain the decision of a system by estimating the contribution that certain parts of the input had on the system's output. Formally, they can be described as follows [4]: Consider a neural network with input $x = [x_1, ..., x_N] \in \mathbb{R}^N$ and output $S(x) = [S_1(x), ..., S_C(x)] \in \mathbb{R}^C$ where C is the number of output neurons. Given a specific neuron c, attribution-based explanation methods try to determine the contribution $R^c = [R^c_1, ..., R^c_N] \in \mathbb{R}^N$ of each x_i on the neurons output S_c. When R^c has the same shape as x, R^c is called attribution-map(s) [4].

In the case of a CNNs for image classification with one output neuron per class, S_c denotes the score of class c, which is usually the softmax score (see Equation 2.4), and R^c the contribution of each input - on a pixel or feature level - for the classifiers score for class c. In some cases, R^c is not calculated directly, but only estimated. Several methods for estimating relevance exist. Many of them are based on the assumption that if the classification result is sensitive to a specific input - i.e., small changes in the input cause significant changes in the system's output - this input is relevant for the classification. Highlighting relevant input can, therefore, help to understand the reasons for the classifier's decision, and can be seen as an explanation for its prediction [44]. In the following, some example methods for relevance estimation will be presented. An extensive survey of the effectiveness of different gradient-based attribution methods is presented in [4].

2.4.1 Occlusion Method

The Occlusion Method is the least complex approach to calculate the importance of a specified input for the prediction. For that reason, they are introduced here [46]. The idea is simple: to estimate the relevance of some input image, successively occlude different portions of it (e.g., with gray patches) and observe the deviations in the classifier's output. The more the output changes, the more relevant the occluded input region is for the system's prediction. A drawback of this approach is that its computational costs are high compared to other methods, as it requires several forward passes through the CNN. On the other hand, it does not require the calculation of gradients and thus no backward passes for back-propagation. This property makes the method applicable to systems that are not differentiable.

2.4.2 Saliency Maps

The calculation of saliency maps, as proposed by [57], is a more sophisticated approach to calculate an attribution-map for the input image. Saliency maps are, to the best of our knowledge, the first explanation method for CNNs that harnesses the derivative of the score with respect to the input image. As explained, CNNs calculate the function $h(x)$. The partial derivative of a function (i.e., a class score S^c) with respect to an argument of that function (i.e., an input pixel x_{ij}) measures how the function value would change if the argument did (while all other arguments remain constant). Thus, it can be interpreted as sensitivity (i.e., R^c) of the output with respect to that input. To express the sensitivity of the class score with respect to some input pixel, we can write the following:

$$R^c = \frac{\partial S_c}{\partial x_{ij}} \tag{2.8}$$

Compared to the Occlusion method, the calculation of a saliency map is computationally cheap, as it required only a single back-propagation pass [57].

2.4.3 Class Activation Maps

The term Class Activation Map refers to an image that highlights the relevant input regions with respect to a certain class, i.e., an attribution-map R^c. There is a set of algorithms that can be used to calculate CAMs [71, 55, 10]. They are based on the following assumption: The activation-maps A of the last convolutional layer of the CNN have the highest level of abstraction that still contains information about the location of the detected features in x because for example Flattening or Global Pooling Layers would destroy that information. CAM algorithms propose different methods to estimate the relevance α^c of each value in A. To calculate the CAM $L^c(x)$ for some class c, the activation-maps are weighted by this relevance, summed up, and passed through the $ReLU$ (see Equation 2.1) function, so that only pixels that have a positive impact are included. The formal notation of this calculation is given in Equation 2.9.

$$L^c_{ij}(x) = ReLU(\sum_k \alpha^{ck}_{ij} A^k_{ij}) \tag{2.9}$$

For example, consider the Vgg19 model with 512 activation-maps of size 7×7. CAM algorithms will generally calculate a relevance α^c for each of these activation-maps. Each map is weighted by the corresponding α^c, then all 512 activation-maps are summed, which results in a 7×7 image. Each value in that image that is < 0 will then be set to 0. This process results in L^c. In order to obtain the attribution-map R^c for the input image, L^c is scaled up to match the size of the original image. Each pixel in the resulting CAM is assumed to estimate the relevance of the corresponding pixel in the input image.

2.4.3.1 Vanilla CAM

The first method to calculate the relevances α^c is referred to as *vanilla CAM* [71]. The relevances are estimated by training a linear classifier[3]. The weights of this linear classifier are then used as relevance. As this requires retraining and is not applicable to every network type, the method is severely limited.

2.4.3.2 GradCAM

GradCAM [55] aims to overcome the limitations of vanilla CAM by improving and generalizing the estimation of the weights. The relevance α of a single pixel

[3] A linear classifier calculates the result as a linear-combination of the input, i.e. $f(x) = w^\top x$.

(i, j) in an activation-map A^k for the score S_c of class c is calculated as:

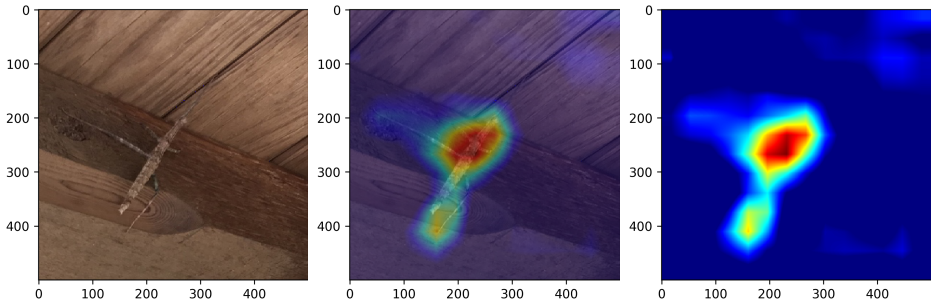
$$\alpha_{ij}^{ck} = \frac{\partial S_c}{\partial A_{ij}^k} \tag{2.10}$$

The relevance α^{ck} of an activation-map k with respect to class c is then calculated as a mean over gradients of each activation-map:

$$\alpha^{ck} = \frac{1}{Z} \sum_i \sum_j \frac{\partial S_c}{\partial A_{ij}^k} \tag{2.11}$$

where Z is the (constant) number of pixels in an activation-map A^k. This means that the weight α of activation-map k with respect to class c is the mean of the gradients of activation-map A^k with respect to the score S^c for class c. Note that unlike for vanilla CAM, GradCAM assigns the same weight to all pixels in an activation-map. This allows calculating a GradCAM for each classifier whose partial derivatives can be calculated. Like saliency maps, this approach is computationally cheaper compared to vanilla CAM, as calculating the gradients can be done via backpropagation and does not require retraining. An example of a GradCAM is given in Figure 2.6.

Several modifications of GradCAM exist, such as *Guided GradCAM* [55] or *GradCAM++* [10]. As they do not contribute to the concept of this thesis, they are considered to be out of scope.



(a) Original (b) Superimposed (c) GradCAM

Figure 2.6: Example GradCAM image from the ImageNet-A dataset [31] and the Vgg19 [58] architecture.

2.5 Reasons for Image Recognition Failures

The following section will elaborate on reasons for the failure of CNN-based visual recognition systems. A prominent interpretation of machine learning as a branch of statistics assumes that the input data is sampled from an unknown

Table 2.2: Types of misclassified Images. Images of known classes that are not malicious are *Ordinary* images. Images of unknown classes (which are always misclassified) are called Open-Set images. Images of known classes that are intentionally modified to be misclassified are adversarial images. Images of an unknown class that are malicious (e.g., synthesized for the purpose of being misclassified) belong into the *Fooling* category.

Image Type	**Unknown Class**	**Malicious**
Ordinary [31, 29]	×	×
Open-Set [7]	✓	×
Adversarial [24]	×	✓
Fooling [45]	✓	✓

probability distribution. The purpose of a machine learning algorithm is to create a model that can make accurate predictions for previously unseen samples that stem from the same distribution [9, 23]. When such systems are deployed in real-world applications, this assumption is sometimes violated.

For instance, consider a company that trains a CNN on a dataset and then deploys it into a web application that allows users to upload images for classification. Let us assume that the training dataset contains high-resolution images without blur that depict a single object of interest in the center of the image. The images uploaded by the users may stem from smart-phones with lower resolution, blur, and possibly multiple uncentered objects. Under such conditions, severe deterioration in performance can be expected. Thus, a general reason for failures in image recognition systems are drifts in the underlying data distribution.

The following section approaches the problem from a practical perspective. It aims to identify types of images that may be encountered in a production environment and cause a recognition system to fail. We conducted a literature survey regarding these failures, and loosely grouped them into four classes. An overview of the categories is provided in Table 2.2. Each category will be introduced in the following.

2.5.1 Ordinary Images

The term *ordinary images* denotes a class of images for which the model could have theoretically made a correct prediction, but was unable to do so. We identified two subtypes that will be introduced in the following.

2.5.1.1 Natural Adversarial Images

Natural advserarial exmaples are natural images that are difficult to recognize for CNNs and cause many classifiers to degrade [31]. These images exploit flaws inherent to contemporary CNN architectures, namely their over-reliance on color, texture and background cues instead of, for example, the shape of objects. If we consider the example images given in Figure 2.7 it is apparent which properties of the image cause the classifier to make its particular prediction.

2.5.1.2 Corrupted Images

Corrupted images are often misclassified [29]. Typical corruptions include different types of noise (gaussian, impulse), blur (motion, defocus) and for example weather related corruptions like snow, frost, rain or fog.

(a) Stingray (b) Obelisk

Figure 2.7: Examples of natural adversarial images: (a) Stingray, classified as *albatross* (87.2 %), and (b) Obelisk, confused with a *flagpole* (87.7 %) by the Vgg19 architecture.

2.5.2 Open-Set Images

Image recognition systems are usually designed to recognize objects from a finite set of classes. In real-world applications, however, systems may be exposed to objects of classes that have not been part of this set. CNNs with a softmax output will still predict the object as belonging to some known class, as the architecture does not allow them to make a prediction outside of their known class set.

For instance, consider a CNN that was trained to classify images as showing either a *cat* or a *dog*. This classifier will usually have two output neurons whose values will sum up to one because of the softmax activation function (see Equation 2.4). If this classifier is presented with an image of a horse, it might output, for example, that the image depicts a cat or a dog with 50 % probability each. This prediction is not accurate, as the image depicts neither a cat nor a dog,

so both neurons should ideally output zero percent probability, which is impossible. Even if the net-inputs of the output neurons are small, the softmax will scale these values up. This example illustrates the inability of softmax-based recognition systems to make a correct prediction in such cases, a problem that is referred to as the *Open-World Problem* or the *Open-Set Risk* [6, 52]

2.5.3 Malicious Images

Apart from natural images, there are two other categories of images that are designed to be misclassified with high confidence. We refer to them as malicious input.

2.5.3.1 Adversarial Images

Visual perception systems, like CNNs, must be invariant to irrelevant variations of the input in order to work in real-life scenarios [39]. In 2013, however, Szegedy et al. demonstrated that CNNs could be tricked into misclassifying an image by injecting specially crafted perturbations that were invisible to the human eye [64]. They defined adversarial inputs as:

> *"inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake."* [64]

The findings of their publication gave birth to a research area concerned with the development of attacks to exploit machine learning models, as well as means to defend them against adversaries. An example is provided in Figure 2.8.

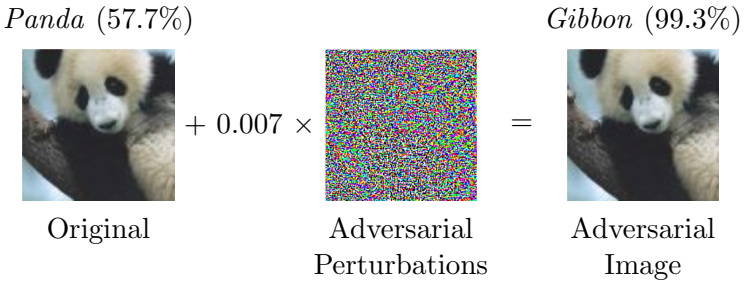
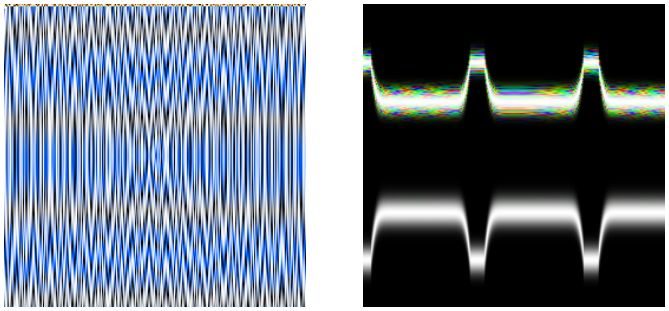


Figure 2.8: Adversarial Image, as presented by [24]

2.5.3.2 Fooling Images

So-called fooling images are entirely synthetic images created for the purpose of being misclassified by CNNs. Some example images are provided in Figure 2.9. They were first presented by Nguyen er al. [45], who used an evolutionary algorithm or gradient ascend to generate images that would maximize the activation of a certain output neuron.

We differentiate between fooling images and adversarial images, because a classifier could theoretically make a correct prediction for an adversarial image depicting a known object, but is by design unable to correctly classify a fooling image as it does not depict any real object. Therefore, in the context of this thesis, fooling images are more related to the Open-Set problem, even though they meet the definition of adversarial images given in the previous section.



(a) Broom (96.3 %) (b) Digital Clock (71.4 %)

Figure 2.9: Examples of fooling images classified as (a) *Broom* or (b) *Digital Clock* with high confidence by the Vgg19.

2.6 Self-Assessment

The previously described pathologies that CNNs suffer from - and that might be inherent to their architecture - present a challenge in safety-critical environments. A way to mitigate the uncertainty that emerges from the inability to prevent failure is to detect situations in which the recognition system is likely to make a false prediction. These methods can be interpreted as confidence estimation for the prediction - the higher the likelihood of an error, the lower the confidence.

As described in the previous section, the reason for recognition failures can be seen as a situation in which the CNN is presented with input that does not stem from the distribution it was trained on. In fact, the concepts of out-of-distribution detection and failure detection are closely related [30]. In the context of visual recognition systems, the detection of errors or the estimation of the likelihood of errors is also referred to as *Meta- Recognition* [53] or *Introspection* [13]. For the remainder of this thesis, however, we will refer to the concept as *self-assessment*. In this section, the state-of-the-art for self-assessment of CNN-based computer vision systems will be presented. The various approaches are grouped by the data that is involved in the calculation: *Output*, *Input*, *Intermediate Data* or *Context*. An overview is presented in Figure 2.10. Furthermore, it is possible to train neural networks to reject unknown input by training with

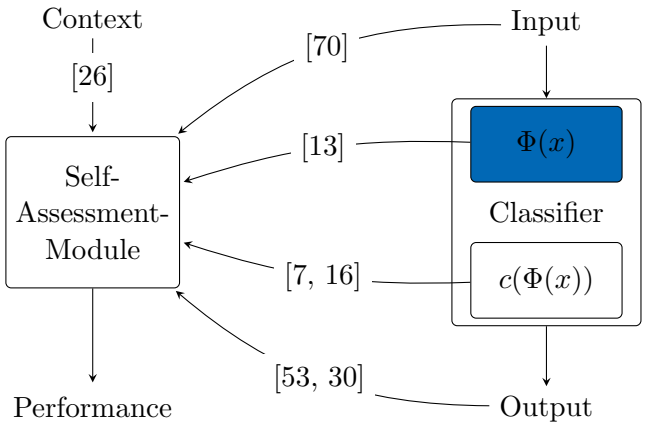


Figure 2.10: Overview of different self-assessment approaches based on input, output, intermediate features and context.

an *"Other"* class.

2.6.1 Output Based

Several publications are concerned with performance prediction-based on the recognition systems output. Generally, these approaches are called post recognition score analysis.

2.6.1.1 Tresholding Softmax

A straight forward way to reject predictions is to threshold the softmax output of a classifier. Hendrycks et al. [30] proposed this as a baseline for the detection of misclassifications, as misclassified instances tend to have lower maximum softmax scores. However, studies have shown that this method is insufficient to reliably predict failures, mainly due to properties inherent to the softmax function, some of which were mentioned earlier [7].

2.6.1.2 Meta-Recognition

In 2011, Scheirer et al. [53] introduced the term *"meta-recognition"* to describe the capability of a system to assess its own predictive performance. They defined such a system as follows:

> *"Let X be a recognition system. We define Y to be a meta-recognition system when recognition state information flows from X to Y , control information flows from Y to X, and Y analyzes the recognition performance of X, adjusting the control information based upon the observations."* [53]

The idea is to estimation the confidence of a prediction-based on an analysis of the systems output and extreme value theory, which was later adopted by [7]. Extreme value theory is a branch of statistics that deals with rare events, i.e., extreme deviations from the median of probability distributions. Scheirer et al. analyzed the softmax scores and found that they followed a Weibull distribution. This allowed them to model the output-score-distribution and to estimate the probability that a score is an outlier of this distribution.

2.6.2 Input Based

Some self-assessment systems evaluate the input (i.e., the image) in order to predict if the recognition system will fail. An advantage of input-based self-assessment systems is that they can be used on any recognition system without the need for adjustments [70].

An example of such an input-based self-assessment approach proposed by Zhang et al. is ALERT [70]. The idea is to extract a number of generic appearance features from the image, using classical feature extraction methods like SIFT, HOG, GIST, line histograms, local binary patterns, and self-similarity. Afterward, a Support Vector Machine is trained to predict the performance of the system for a variety of computer vision tasks such as semantic segmentation, vanishing point estimation, and camera parameter estimation.

2.6.3 Intermediate Based

Self-assessment methods for CNNs based on intermediate features use neither the output nor the input of thy system, but usually the activation of some hidden layer, for example, the penultimate layer.

2.6.3.1 Confidence Estimation Branch

Devries et al. proposed to augment a CNN with an additional confidence estimation branch with one or more dense layers that outputs a scalar confidence value [16]. This branch is fed with the output of the penultimate layer and trained in parallel with the prediction branch by modifying the loss function. The approach applies a threshold to the confidence score to label instances with low confidence as out-of-distribution.

2.6.3.2 Introspective Perception

The approach called *Introspective Perception* proposed by Daftry et al. uses the feature embedding of the input, and trains a Support Vector Machine to estimate error metrics for various computer vision applications [13].

2.6.3.3 OpenMax

The concept of the *OpenMax Layer* as proposed by Bendale et al. [7] is derived from the meta-recognition model proposed by [53]. Instead of using the softmax scores for the assessment, it uses the neural activations in the penultimate layer. Bendale et al. found that for images of a single class, these activations would be similar to each other. Thus, they assumed that the instance of one class would form clusters in the vector space of activation. Using the Nearest Class Mean Centroid clustering algorithm, they determined a cluster center for each class using NCM clustering (called Mean Activation Vector) and calculated the distance to this cluster center for each correctly classified instance in the training set. They found that these distances follow a Weibull distribution. Hence, they estimate the parameters of this distribution to build an extreme value theory-based model of how far the activation are usually distributed around the cluster center of each class. In order to estimate the confidence of a prediction, the distance of the activation in the penultimate layer to the cluster center of the predicted class is calculated. Afterward, the probability of encountering a correctly classified instance with that particular distance is calculated using the parameters of the Weibull distribution and extreme value theory.

2.6.4 Context Based

Another approach to establishing confidence in the model's prediction is also to include additional information which the perception system did not receive, e.g., metadata of the image (location, time, weather conditions) [26].

Generally, the context of the input can serve as evidence for the distribution that the input stems from. If, for instance, the lane detection of an autonomous vehicle was trained on data gathered under sunny weather conditions, but a weather service indicates that the current input was captures during a blizzard, one could suspect that input image might be corrupted (see Section 2.5.1.2) and thus stems from another distribution which can lead to system failures.

2.6.5 Training with an "Other" Class

[51] argues that labeling something as *new*, *unknown* or *other* should always be considered a valid option in an open environment. An approach to implement this requirement is to design a CNN with an explicit *"unknown"* class, i.e. presenting it with diverse images that do not belong to any other class, and let the classifier learn to assign an *"unknown"*, *"background"* or *"garbage"*-class to them. This can be seen as a baseline-approach for the Open-Set problem, as the classifier learns to identify images that do not belong to any other class. The goal is that the classifier learns to detect images that do not stem from the distributions of any other classes, and assigns a default class to them. However, [17] states that this is in practice difficult to achieve because it requires that all

those instances are mapped into the same region in the feature space. To enforce this constraint, they propose the Objectosphere-loss-function, which maximizes the L_2 distance for the clusters of the unknown class and all other classes in the feature space and thus aims to improve the deep feature representation.

Nguyen et al. report that training with a rejection class fails to provide a reliable mechanism for defense against fooling images. They were able to create a new batch of fooling images that would be miss-classified even if the network has been trained to reject fooling images [45].

Chapter 3

Attribution based Self-Assessment

As described in the introduction, state-of-the-art visual perception systems are sometimes multiple orders of magnitudes too complex - in terms of the number of their parameters - to be formally verifiable by contemporary technology and available compute. There are, in general, two approaches to deal with the uncertainty that arises from the pathologies which some experts assume to be inherent to contemporary Deep Neural Networks [21]. The first approach aims to improve the performance of CNNs in order to diminish the likelihood of errors, e.g., by using ensemble models as described in Section 2.3. The second approach tries to detect errors when - not if - they occur in order to be able to handle them and retain the system in a safe state, as described in Section 2.6. We refer to the latter method as *self-assessment*.

In this chapter, the self-assessment approach for visual recognition systems exploiting attribution based explanations, that was proposed in the introductory chapter, will be explained in detail. Section 3.1 will introduce the general concept and provide arguments to substantiate it. In Section 3.2, a framework that facilitates the development of explanation based self-assessment models, will be presented. Ultimately, a statistical self-assessment model derived from that framework will be introduced. This model, and thus the approach, will be evaluated in Chapter 5.

3.1 Concept

As described in section 2.4, the purpose of attribution-based explanation methods is to provide a means to create a human-comprehendible explanation for the prediction made by a classifier by identifying parts of the input that were most relevant for the prediction. Thus, they allow a human in hindsight to investigate if the decision of the classifier was reasonable. In GradCAM, as pro-

posed by Selvaraju et al., this is achieved by using the activation-maps and an estimate of the sensitivity of the prediction with respect to each map. For their evaluation, Selvaraju et. al. conducted user studies that suggested that human non-experts[1] were able to identify the more reliable of two classifiers when provided with the GradCAMs of both for several input-images. This experiment can be understood as the assignment of confidence to an image recognition system based on patterns in the activation of neurons (i.e., the features the CNN extracted from the images) and the relevance of these neurons for the prediction. We suspect that it is not only possible to assign explanation-based confidence on a system level, but also to use explanations to assign confidences to individual predictions. On an abstract level, we argue that an explanation that is similar to the explanation observed for correct predictions can be regarded as evidence for a correct prediction. On the other hand, an "unusual" explanation that is dissimilar to explanations given for correct predictions is likely to be evidence for a misclassification. Furthermore, we presume that the process of evaluating explanations to detect recognition failures can be automated, which enables its employment for self-assessment at runtime.

Throughout our literature study in the field of recognition failure detection approaches, as presented in Section 2.6, we were unable to find any publications considering the harnessing of explanation methods for self-assessment. As there are numerous different explanation methods, an exhaustive evaluation is considered to be out of scope. Instead, we aim to provide a proof-of-concept by focusing on the specific method for attribution based explanation that inspired the concept: GradCAM. The straight forward approach would be to generate a GradCAM for each image in a dataset and to train a CNN to predict if the original image was misclassified based on that CAM. We suspect that such an approach would yield suboptimal results for the following reasons: In order to make the explanation human understandable, CAM algorithms condense the information from activation- and attribution-maps into a single image by calculating a weighted mean. This is necessary as humans are unable to inspect several hundred images at the same time. As a side effect, this destroys information about which specific activation-maps were most active and most relevant. We argue that this information is potentially valuable for the self-assessment and should be kept. Therefore, we will not base the self-assessment models directly on the GradCAM. Instead, we will use the information that the CAMs are calculated from - activation- and attribution-maps - as a proxy for attribution-based explanation and hypothesize the following:

Proposition: In Convolutional Neural Networks for image recognition, patterns of attribution observed for an image that are dissimilar to patterns ob-

[1]The participants where hired on the platform Amazon Mechanical Turk. See `https://www.mturk.com/`

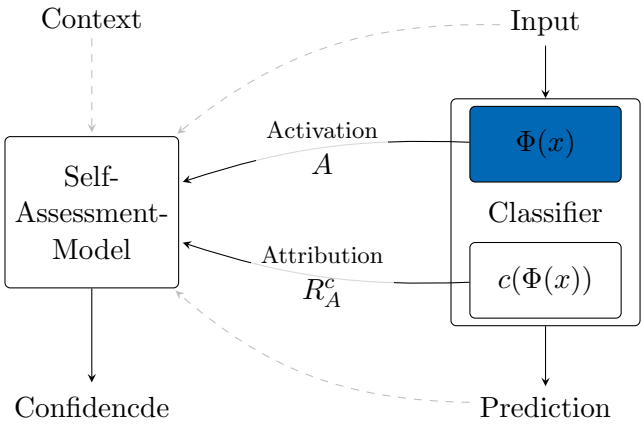


Figure 3.1: Proposed self-assessment approach based on activation A of neurons for a certain layer and the relevance R_A^c of these neurons for the predictions.

served for correctly classified images during training indicate a higher likelihood of a misclassification of the CNN for that image.

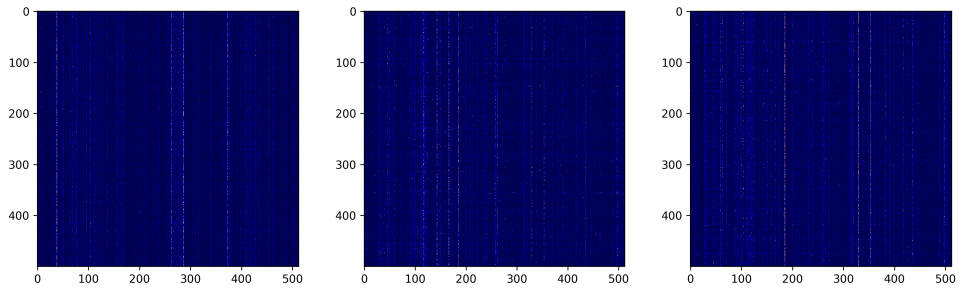
If the proposition holds true, we can predict from it that it is possible to harness the attribution-based explanation methods for self-assessment. This concept is delineated in Figure 3.1. The main contribution of this thesis is the evaluation of this proposition by gathering evidence to support (or contradict) it. The remainder of this chapter will provide arguments and some empirical evidence to substantiate this hypothesis.

3.1.1 Activation

The idea to use the activation values of neurons for the purpose of self-assessment has already been discussed in the scientific community [13, 7]. Nevertheless, in this section, arguments for the use of feature embeddings for self-assessment will be gathered, as the reasoning is similar to the reasoning for the use of attribution values for neurons. First, a more intuitive line of thought will be introduced in order to provide the reader with an intuition of the general idea. Afterward, formal arguments will be brought forward.

3.1.1.1 Intuition

Consider a CNN trained to classify images as showing either an object of class *Tiger*, *White Shark* or *Tiger Shark*. The convolutional layers will extract features from the given images that allow the classifier to distinguish between different classes. These features can stem from the object itself, or its surroundings. For instance, one would expect that *White Shark* and *Tiger Shark* will often be depicted in an underwater environment, while for images of



(a) Tiger (b) White Shark (c) Tiger Shark

Figure 3.2: Global Average Pooled activation-maps of 500 instances for 3 example classes in the ImageNet 2017 training set for the Vgg19. The vertical lines indicate that instances of the same class tend to have similar values in the activation maps.

*Tiger*s, this will be much less likely the case. Additionally, one would expect that the features detected from the *White Shark* and *Tiger Shark* themselves will be more similar, compared to that of a tiger, because these classes are apparently more similar in terms of their visual cues.

For instance, consider Figure 3.2, which visualizes the embeddings of instances from different classes. The visible vertical lines indicate that instances of the same class tend to have similar embeddings. Now, let us assume that the classifier predicts that some image belongs to the class *Tiger*. If this prediction is correct, we would expect that the features are similar to the features of instances of class *Tiger* in the training set. If, on the contrary, the classifier predicts *Tiger*, while only a few neurons usually associated with *Tiger*s are active, or features usually associated with underwater environments are being detected, we would be suspicious about the prediction.

3.1.1.2 Arguments

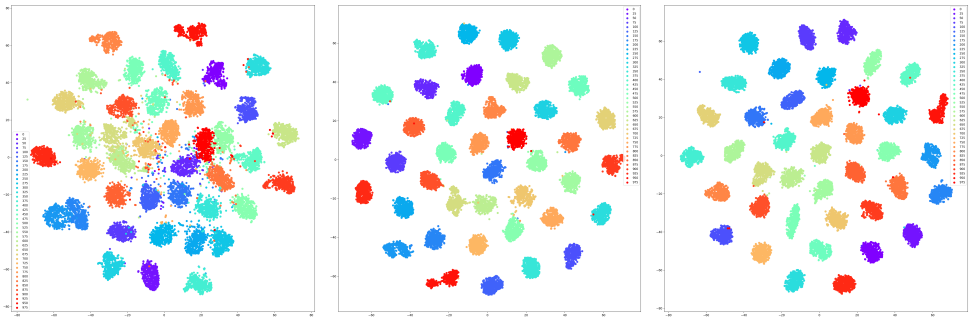
As mentioned in Section 2.2, the front layers of a CNN map the input into a lower-dimensional space, called feature space. The process of learning this embedding function Φ is some times referred to as representation learning because the input data is transferred into another lower dimensional representation that should preserve as much information about the original input as possible [23]. It is widely assumed that the embedding function preserves similarities in the input. This means that if two input images x_1 and x_2 are similar, then $\Phi(x_1)$ is expected be similar to $\Phi(x_2)$ according to some distance metric [32]. As similar images are mapped to the same regions of the feature space, images of the same

class can be expected to form clusters. In the simplest case, we can think of the instances of a class as residing inside of a hypersphere around some class center(s).

This property of the neural activation has been visualized in Figure 3.3. We applied a similarity preserving dimensionality reduction method called t-SNE to transform the embeddings of instances of different classes into the 2D plane [40]. We notice that different classes form distinct clusters.

Therefore, when the classifier predicts that an instance x belongs to a certain class, while $\Phi(x)$ significantly diverges from the embeddings usually observed for this class (i.e., $\Phi(x)$ is an outlier of the class-cluster(s)), we can anticipate that x also significantly diverged from the instances observed during training. We should, therefore, be skeptical about the classifier's prediction.

This idea is related to the concept of the open-set problem, which states that instances that are dissimilar from the training instances of a class are more likely to belong to another *unknown* class [6].

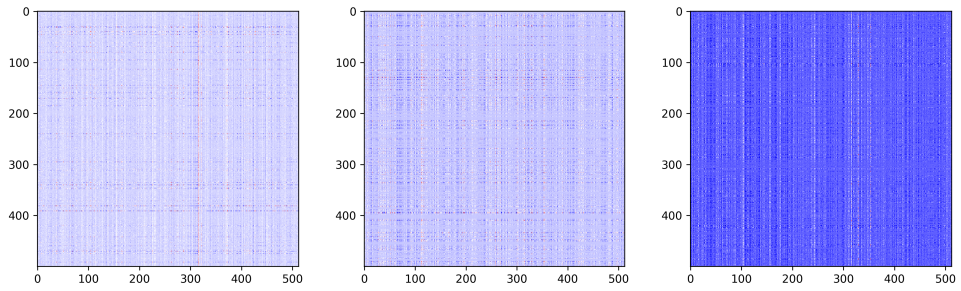


(a) Vgg19 (b) Xception (c) Inception v3

Figure 3.3: 2D-activations (t-SNE embeddings [40]) for instances of 40 classes from the ImageNet 2017 training set. Apparently, the Vgg19 architecture does not produce equally distinctive clusters.

3.1.2 Attribution

In this section, arguments for the use of neural attribution values for self-assessment will be presented. To the best of our knowledge, this approach has never been discussed in scientific literature before. However, the line of reasoning is analog to the one for the neural activation. At first, a more intuitive line of thought will be presented in order to provide the reader with an intuition of the general concept. Afterward, formal arguments will be brought forward.



(a) Tiger (b) White Shark (c) Tiger Shark

Figure 3.4: Global Average Pooled attribution-maps of 500 instances for 3 example classes in the ImageNet 2017 training set for the Vgg19. The vertical lines indicate that instances of the same class tend to entail similar values in the attribution-maps.

3.1.2.1 Intuition

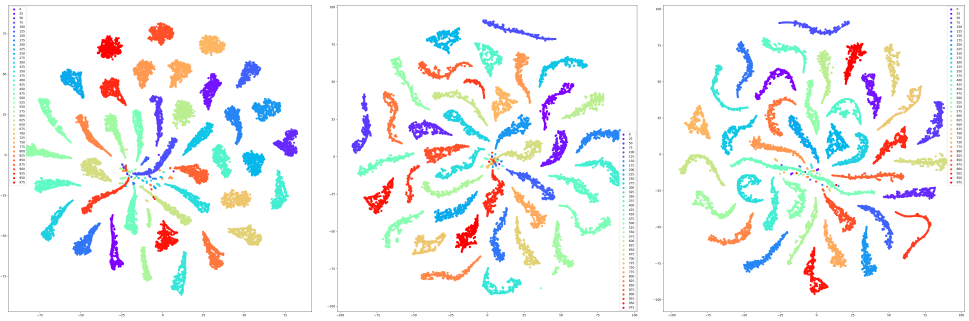
Again, consider a CNN trained to classify images as showing either an object of class *Tiger*, *White Shark* or *Tiger Shark*. The convolutional layers extract features from these images. The relevance of these features for the example classes is depicted in Figure 3.4. Let us assume that for some image, the classifier predicts the class *Tiger*. We would now expect the classifier to base this prediction on features that are usually associated with the class *Tiger*. Therefore, the attribution values obtained from the explanation method should be similar to the ones observed for correctly classified images of class *Tiger*. If the classifier, on the contrary, based its decision on features that are in general not associated with tigers, we should be skeptical about the prediction. If, for example, we observed unusually high relevance scores assigned to features for underwater environments, while the classifier predicts *Tiger*, we should be suspicious about the classifier's decision.

3.1.2.2 Arguments

In line with the similarity of neural activations, we argue that if two instances x_1 and x_2 are similar, the relevance assigned to the neurons should be similar as well. If similar inputs result in similar extracted features - which, as described in the previous section, is generally assumed to be true for well-trained networks throughout the scientific literature - then, the relevance assigned to these features should be similar as well, because otherwise, the classifier's output could be different.

This property of attribution values has been visualized in Figure 3.5. Again, we used t-SNE to transform the attribution values of instances of different classes

into the 2D plane. We notice that different classes form distinct clusters.



(a) Vgg19 (b) Xception (c) Inception v3

Figure 3.5: 2D attribution (t-SNE embedding [40]) for instances of 40 classes from the ImageNet 2017 training set.

3.2 Framework

In this section, we will present a framework that facilitates the development of models for self-assessment based on activation- and attribution-maps. The goal is to modularize the process in a way that enables the transparent exchange of individual algorithms. For each of the steps, we will provide some examples of algorithms that may be used. The general idea is to model how "good" or "bad" explanations (in the form of activation- and attribution-maps) look. The model will then assign confidence to new predictions based on how similar their explanation looks like to the ones previously observed. We follow the typical machine learning framework of gathering data, extracting meaningful features from that data, and then building a model to make predictions for unseen data. An overview of the framework is provided in Figure 3.6.

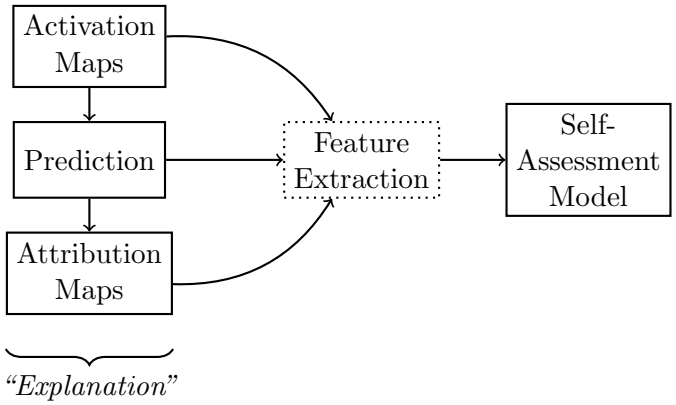


Figure 3.6: Scheme of the proposed framework. First, gather the required data and extract features if necessary. Finally, build a model to make predictions for unseen data.

3.2.1 Data Acquisition

Initially, we require data to build a self-assessment model. This data will usually consist of a number of explanations for predictions of the CNN of interest. Since we focus on attribution based explanations, we will gather activation-maps, attribution-maps, and predictions, as they contain all information that is used to calculate a CAM.

3.2.1.1 Calculation of Activation-Maps

First, we have to decide on a layer whose activation we will use. In general, the activation A of any layer of the CNN could be used, as long as it is possible to calculate the relevance of these activations for the prediction. In the most extreme case, even the input layer could be chosen (where $A = x$). In practice,

however, high-level features are preferable, as they are more distinctive for different classes [7]. For that reason, Bendale et al. [7] used the activation of the penultimate layer, which resulted in ten channel 1×1 "activation-maps" that contain the outputs of the penultimate layer.

If we intend to preserve information about the location or the distribution of detected features in the input data, we have to pick a layer that is lower than any layer that destroys this information (i.e., flattening or global pooling).

3.2.1.2 Predictions

Next, we have to calculate the predictions of the CNN for the images, as they are required to calculate the attribution-maps in the following step. Calculating the predictions is straight forward. To the best of our knowledge, there are no variations that can be applied.

3.2.1.3 Calculation of Attribution-Maps

Now that we obtained the activation-maps and the predictions, we can calculate the relevance of each neuron for the classifier's prediction. Examples of applicable algorithms include the occlusion method, or gradient based methods like saliency maps, GradCAM, Layer-wise Relevance Propagation [5], Deep Taylor Decomposition [43] or DeepLIFT [56].

3.2.2 Feature Extraction

The calculation of neural activations and the estimation of relevance scores for these activations will result in several activation- and attribution-maps for each image. If a layer was chosen that does not destroy spacial information, the self-assessment could be understood as an image classification task involving two images (activation- and attribution-maps) of the same size with k channels each (where k is the number of activation/attribution-maps)[2]. As for other image classification tasks, the data can be high dimensional, so it might be desirable to extract features from these maps. In general, every feature extraction process that can be applied to images can be used, as long as it can be applied to images of any size with an arbitrary number of channels. Examples for algorithms include calculating the mean over each activation-map, which is equivalent to global average pooling. More sophisticated methods include HOG, SIFT, GIST, or pixel histograms, which are used in state-of-the-art computer vision tasks that are not learning to extract features themselves. Moreover, it is possible to apply feature learning (or representation learning) methods. However, this might impair the explainability of the model. Other options to reduce the

[2]It could also be interpreted as image classification task for a single image with $2k$ channels.

dimensionality of the data include linear- or nonlinear dimensionality reduction algorithms, like Principle Component Analysis or manifold learning.

3.2.3 Self-Assessment Model

The final step is to build a model that estimates the confidence in the prediction based on the features extracted from the neural activation- and attribution-maps in the previous step (or the maps themselves if no feature extraction was performed).

In the context of multi-class image recognition, a fundamental decision is whether to build one model for all classes or create an individual model for each class[3]. While the former is probably less complex with regard to the implementation, we anticipate that the self-assessment model itself has to be more complex to make reliable predictions for all classes. On the other hand, the model might be able to represent some properties of the overall system that can not be captured by smaller models that only predict failures for one class. If, for instance, the activity of one particular neuron indicates a failure regardless of the prediction, a single model for all classes could be able to leverage this correlation more effectively.

In the following chapters, several models will be suggested. In general, these models can be loosely grouped into statistical models and machine learning models[4].

3.2.3.1 Statistical Model

As discussed earlier, it is generally assumed that the embeddings of instances of the same class are similar to one another, and we argue that the same holds for the relevance scores of these embeddings. Mathematically, *similarity* is usually described as a distance as calculated by some metric in a metric space (e.g., a normed vector space). Therefore, instances of the same class will tend to reside in the same region (according to this distance metric) of the vector space, while instances of other classes will tend to reside in different regions. Therefore, if an instance resides in a different region than instances of the same class observed during training, this instance is more likely to be misclassified, as it is dissimilar to the training data and might stem from a different distribution. Such models fall into the category of Proximity-based Outlier Detection. One variation of this approach is to determine cluster centers of the instances in the feature space. The confidence in a prediction is then calculated based on the distance from

[3]Building one self-assessment model per class effectively turn the model into a mixture-of-experts ensemble (see Section 2.3.2), as it divides the problem space into smaller subproblems that are easier to solve. The CNNs output can be seen as a gating function, as it determines which self-assessment submodel will be invoked.

[4]For an in-depth explanation of the differentiation of these concepts, see [9].

these cluster centers. This approach was, for example, used by [7] and could be applied to the attributions as well.

There are multiple distance metrics that one might use, e.g., the Euclidean, Manhattan on the Cosine metric. Bendale et al. [7] used a linear combination of different metrics. It is also possible to learning metrics. This approach was used for image classification before [41, 42]. In order to determine the cluster centers, various clustering algorithms may be used. A common approach in the literature is NCM [42].

3.2.3.2 Machine Learning Model

Another approach is to train a machine learning model to learn to distinguish between correct and incorrect predictions based on the features extracted from the attribution- and activation-maps.

This method was used, for example, by[70] and [16]; however, they harnessed the activation of the last convolutional layer or the penultimate layer and did not employ the relevance values. Daftry et al. [13] trained a Support Vector Machine to estimate the confidence based on the feature embeddings. Another option would be the usage of Siamese networks, a deep neural network architecture that is used, for example, in face recognition [65]. In this approach, two identical networks explicitly learn to extract features of images in a way that minimizes the distance of the feature embedding for images of similar classes, while maximizing this distance for images of different classes. As indicated in the previous section, when using convolutional architectures, the feature extraction step could be omitted, as the convolutions extract characteristic features by themselves.

A problem that may arise is in a machine learning setting is the class balance: with recognition systems becoming increasingly more powerful, the number of errors in the training set decreases, and thus the number of possible negative training examples for a machine learning algorithm. To remedy this, one can assign higher sample weights to the minority class. Models that learn only from correctly classified instances do not suffer from this problem. These models do not learn to discriminate between samples from multiple distributions, but only estimate whether a sample is out-of-distribution. To some extent, this mitigates the problem that all instances from “unknown” classes have to be mapped to the same region of the feature space (see Section 2.6.5). One such approach is the (stacked convolutional) autoencoder architecture as used for anomaly detection [23]. This network architecture learns what “normal” input looks like by mapping the input images into a lower-dimensional representation, and to reconstruct the original input from this compressed representation afterward.

The reconstruction process usually introduces reconstruction errors. After successful training, these errors are expected to be small for inputs similar to the ones observed during training, but significantly larger for unusual input.

3.3 Modeling

This chapter introduces the self-assessment models that were used throughout the experiments. For each step in the afore-described generic framework, we will explain which algorithms were used and why.

As stated earlier, the purpose of this thesis is to evaluate whether attribution scores can be harnessed to predict accurate confidences, or if they can be used to improve the confidence estimation based on neural activation. To test this, we extracted three different types of features from the activation- and attribution-maps of the last convolutional layer, and build three self-assessment model (per CNN) based on one feature each. While two of these features incorporate the attribution-maps, the other one does not. An overview is provided in Figure 3.7. In Chapter 5, the performance of these models will be compared, which allows us to make statements regarding the hypothesis. If, for example, the results show that self-assessment the models using the attribution score do not perform better than random guessing, the hypothesis becomes less tenable.

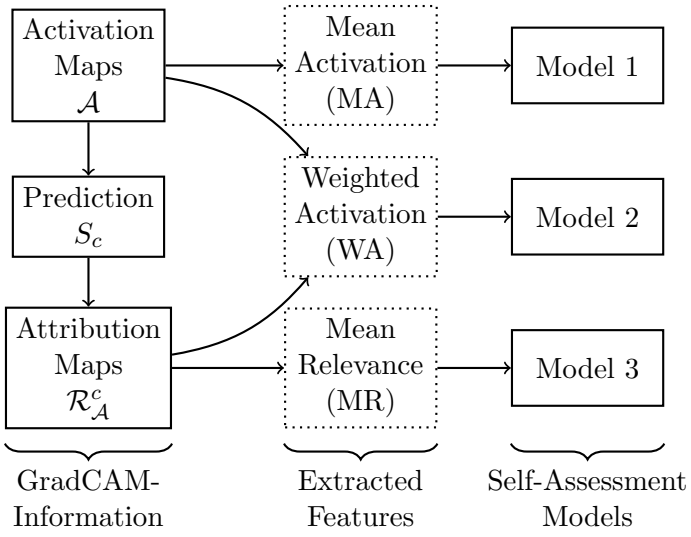


Figure 3.7: Overview of the self-assessment models derived from the framework. We extracted three different features from the information that GradCAMs are calculated from and built a separate model for each.

3.3.1 Data Acquisition

As described in Section 3.2.1, we have to acquire the relevant data from the CNN first.

3.3.1.1 Activation-Map Calculation

We decided to use the activation of the last convolutional layer for the reasons mentioned earlier. The calculation of the neural activation A from the input image is straight forward. The input x is passed forwards through the layers down to the layer of interest. Note that in the following description of the self-assessment model, A could also refer to the activation of other layers.

3.3.1.2 Predictions

From the extracted features, we can calculate the prediction of the network by propagating A forwards through the remaining layers and obtain a predicted class c.

3.3.1.3 Attribution-Map Calculation

Next, we calculate the relevance R_A^c of each neural activation in A for the class score S_c. We decided to use the gradient of the class score with respect to that activation value as relevance, which is equivalent to the GradCAM approach (or the Saliency-Map approach, if you see A as the input of the standalone neural network that consists of all remaining layers). The formula for GradCAM is given in Equation 2.10. Transferred to our approach, we get:

$$R^c_{A^k_{ij}} = \frac{\partial S_c}{\partial A^k_{ij}} \tag{3.1}$$

The resulting attribution-maps have the same dimensionality as the extracted activation-maps.

3.3.2 Feature Extraction

In order to reduce the dimensionality of the data, we extracted three different features from the activation- and attribution-maps. The calculation of these features will be described in the subsequent sections. In the following, the extracted features for some image x are referred to as $f_\lambda(x)$, where λ denotes the explicit type of feature. When $f(x)$ is used without a subscript (e.g. in Figure 3.9), this indicates that the statement applies to all features.

3.3.2.1 Mean Activation

The first feature f_A, called *Mean Activation*, will be derived from the activation-maps. As described in Section 2.1.1, in modern CNN architectures like Xception and Inception v3, the activation-maps A^k are flattened using global average pooling before being fed into the FC layers (see Equation 2.2). Inspired by this, the global average pooled activation-maps, denoted $f_A(x)$, are used as feature vector for our model:

$$f_A(x) = \begin{pmatrix} f_A(x)_1 \\ \vdots \\ f_A(x)_k \end{pmatrix} = \frac{1}{Z} \sum_i \sum_j A^k_{ij} \tag{3.2}$$

where Z is equal to the number of pixels in each activation-map A^k. For the Inception v3 and the Xception architecture, this breaks down to using the vector embedding $\phi(x)$ as feature. The Vgg19 on the other hand flattens its activation-maps without global pooling, thus $f_A(x)$ is unequal to $\phi(x)$ in that case.

3.3.2.2 Mean Relevance

The second feature $f_{R^c_A}$, called *Mean Relevance*[5], will be based on the attribution-maps. Inspired by GradCAM, we calculated the mean over each attribution-map, which can be seen as global average pooling of the attribution-maps. In GradCAM, this value is denoted as α^k and is used as an estimate of the relevance for each activation-map A^k, as stated in Equation 2.11. We calculate $f_{R^c_A}$ as follows:

$$f_{R^c_A}(x) = \begin{pmatrix} f_{R^c_A}(x)_1 \\ \vdots \\ f_{R^c_A}(x)_k \end{pmatrix} = \frac{1}{Z} \sum_i \sum_j R^c_{A_{ij}} \tag{3.3}$$

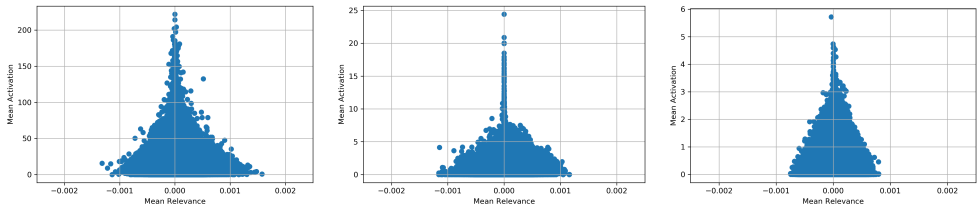
where Z is equal to the number of pixels in each activation-map A^k.

3.3.2.3 Weighted Mean Activation

To examine if there is a correlation between the Mean Activation and the Mean Relevance feature, we plotted both values against each other. The results are depicted in Figure 3.8. The Mean Activation (y-axis) is positive as all CNNs use the ReLU activation function, which can only return positive values (see Equation 2.1).

Apparently, both features are not independent, as we notice that the variance in the Mean Relevance decreases with the magnitude of the Mean Activation.

[5] We chose the name *Mean Relevance* instead of *Mean Attribution*, so the abbreviation would not collide with the *Mean Activation* (MA).



(a) Vgg19 (b) Inception v3 (c) Xception

Figure 3.8: Scatterplot of Mean Relevance (x-axis) plotted against the Mean Activation (y-axis) for all activation-maps of images in the Imagenet 2017 Validation set. While the range for the Mean Relevance seems to be similar for each architecture, the magnitude of the Mean Activation feature varies between the networks. Apparently, the magnitude of the Mean Relevance feature seems to decrease with higher Mean Activation values.

We interpret this is the following way: neurons with a high activation will not change the resulting prediction much when being changed marginally, as the activation will remain on a high level[6]. Therefore, to quantify the impact of a certain neuron, the activation will be weighted with the relevance, as done in the GradCAM approach.

Hence, the third feature, $f_{AR_A^c}$, aims to combine information from the activation-maps and the attribution-maps in a meaningful way. Inspired by GradCAM, we weighted each of the Mean Activations f_A with the Mean Relevance $f_{R_A^c}$ (compare to Equation 2.9) elementwise (Hadamard product). As $f_{R_A^c}(x)$ and $f_A(x)$ are both vectors at the same size, the result of this operation will again be a vector of the same size, The idea is that this vector estimates the impact of each activation-map on the classification result. Unlike GradCAM, however, we do not apply the $ReLU$ function to this calculation, as we do not wish to discard features that have a negative impact on the class score (i.e., all values where $f_{R_A^c} < 0$). This information might be valuable to the self-assessment model. Thus, the feature is calculated as follows:

$$f_{AR_A^c}(x) = f_{R_A^c}(x) \circ f_A(x) = \begin{pmatrix} f_{R_A^c}(x)_1 f_A(x)_1 \\ \vdots \\ f_{R_A^c}(x)_k f_A(x)_k \end{pmatrix} \tag{3.4}$$

[6]This interpretation is not necessarily accurate. For the sake of argument, let us assume that the Mean Relevance follows a normal distribution with a variance that is independent of the magnitude of the activation. There are many instances with a small activation, and a few instances with a high activation. Statistically, this results in smaller observed extreme values for high activation values, as there are fewer samples.

3.3.3 Self-Assessment Modell

The main difference between what we referred to as *statistical models* and *machine-learning models* in Section 3.2.3 is their explainability. Complex machine learning models like Deep Neural Networks or Random Forrest are derived from simple statistical models (e.g., perceptrons and decision trees), but their complexity hinders the understanding of their decisions.

Furthermore, machine-learning-based models could suffer from the same pathologies that motivated us to build a self-assessment model in the first place, i.e., unexpected failures. We, therefore, decide to use a proximity-based statistical model as described in Section 3.2.3.1 because the predictions of such models are easier to explain compared to machine-learning-based approaches. The employment of such a model will allow us to make statistical statements regarding the clustering properties of activation and attribution values.

As mentioned in Section 3.1.1, the feature embeddings of instances of the same class gather in the same regions of the feature space, and, as elaborated in Section 3.1.2 we argue that the same holds for the relevance scores of their features. For simplicity, we make the following assumption: In the high dimensional vector space of the features extracted from the activation- and attribution-maps (i.e., $f_A, f_{R_A^c}$ and $f_{AR_A^c}$), all instances of a class reside in a hypersphere around some cluster center(s). Instances whose true class is unknown that are closer to the centers of these hyperspheres have a higher probability of being classified correctly, while instances in greater distance from the cluster centers have a higher probability of actually belonging to another (possibly unknown) class. An example feature space with some correctly (blue) and misclassified (red) instances is depicted in Figure 3.9.

It should be mentioned that we can not eliminate the possibility of unexpected failures for proximity-based models either, as these models are based on the assumptions mentioned above. If these assumptions are violated (e.g., instances that are dissimilar to instances encountered during training are mapped to the same region), the model will not work as expected. Therefore, the models depend on the quality of the convolutional feature extraction. However, should the self-assessment fail unexpectedly, we can deduce that the activations or the relevances violate the similarity-assumption.

The goal of the "training" of the model is to estimate the clusters centers of each class. As we assume that each class has its own clusters, we will create a separate model for each class, as described in Section 3.2.3. After determining the cluster centers, we will create a statistical model that describes how dense the (correctly classified) instances are distributed around these centers. For some unseen instance classified as c, we can then calculate its distance from the

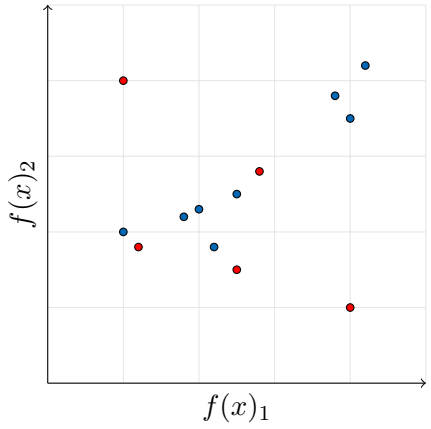


Figure 3.9: Features of correctly classified instances (blue) of the same class c gather in the same region of the high dimensional feature space defined by f.

nearest center of class c and use the model to estimate the probability that a correctly classified instance of class c will be located in that distance.

3.3.3.1 Clustering

In order to estimate the class-specific cluster centers, we used K-Means clustering. This algorithm minimizes the sum of the squared euclidean distances to the cluster center for all instances in that cluster. The clustering results in a set of K cluster centers $\{\mu_1, ..., \mu_K\}$. When applied to the example with $K = 2$, the result would look as depicted in Figure 3.10.

3.3.3.2 Distance Calculation

To describe how the correctly classified instances are distributed around these cluster centers, we calculate the distance d_{min} of each of these instances to the nearest cluster center as follows:

$$d_{min}(x) = \min_{\forall k \in \{1,...,K\}} d(f(x), \mu_k) \tag{3.5}$$

The idea is related to the k-Nearest-Neighbor algorithm. For our implementation, we used the Euclidean distance metric, as we found that the used distance metric did not have a significant impact on the results. This process creates a set of distances. An example histogram of such a distance-distribution is depicted in Figure 3.11a.

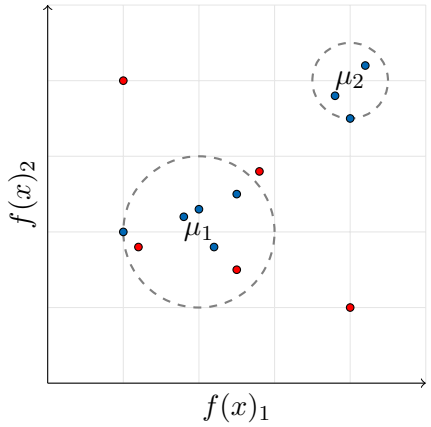
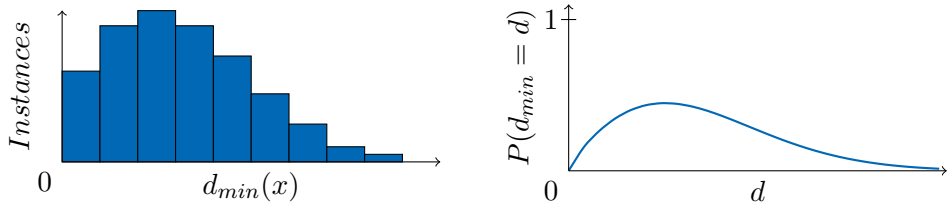


Figure 3.10: Features of correctly classified instances (blue) of the same class c clustering around some cluster centers μ_k in the high dimensional feature space. Misclassified instances (red) are assumed to have a greater distance from these cluster centers than correctly classified instances.



(a) Histogram (b) Weibull PDF

Figure 3.11: Example of a Histogram of minimal distances for all correctly classified instances x of a class c and a corresponding Weibull PDF that fits the distance distribution.

3.3.3.3 Distribution Fitting

[7] found that the distance of image embeddings to their nearest class mean center follows a Translated Weibull Distribution[7]. We were able to confirm this for the layers we used and our clustering algorithms during our experiments. The probability density function of a Translated Weibull Distribution is defined as:

$$w(x; k, \lambda, \tau) = \begin{cases} \frac{k}{\lambda}\left(\frac{x-\tau}{\lambda}\right)^{k-1} e^{-\left(\frac{x-\tau}{\lambda}\right)^{k}} & x \geq \tau, \\ 0 & x < \tau \end{cases} \tag{3.6}$$

where

$k > 0$ is the shape parameter,

[7]Note that they used a different layer and a different CNN architecture.

$\lambda > 0$ is the scale parameter and

τ is the location parameter

We estimate the Weibull parameters k, λ, and τ of the distance distributions. For our example, the resulting Weibull PDF could look as depicted in Figure 3.11b.

3.3.3.4 Confidence Calculation

Using the cumulative distribution function F of the Weibull curve, we can calculate the probability that for some feature extraction function f and some instance x, we observe a correctly classified instance with $d_{min}(x) \geq d$.

$$F(x) = P(d_{min}(x) \leq d) = 1 - P(d_{min}(x) \geq d) \tag{3.7}$$

This corresponds to the probability that the feature embedding of x has a distance $\geq d_{min}(x)$ from the nearest cluster center. For a Translated Weibull Distribution, the CDF is defined as:

$$F(x; k, \lambda, \tau) = \begin{cases} 1 - e^{-\left(\frac{x-\tau}{\lambda}\right)^k} & x \geq \tau \\ 0 & x < \tau \end{cases} \tag{3.8}$$

Given the Weibull parameters, the confidence of some instance x is calculated as:

$$\text{confidence}(x) = \begin{cases} e^{-\left(\frac{d_{min}(x)-\tau}{\lambda}\right)^k} & x \geq \tau \\ 1 & x < \tau \end{cases} \tag{3.9}$$

The resulting probability for our example distance distribution is plotted in Figure 3.12. The resulting probability model is delineated in Figure 3.13. Since we do not create a separate distance distribution model for each cluster (i.e., we do not consider which specific cluster is closest), the radii of the hyperspheres surrounding each cluster center are equal.

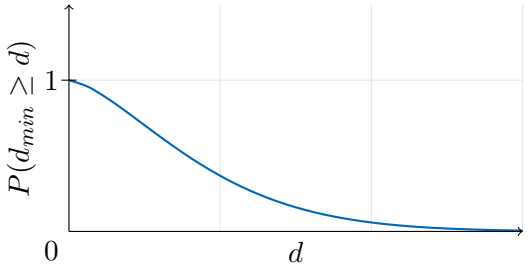


Figure 3.12: Example of the function 1− Commulative Density Function for a Weibull distribution.

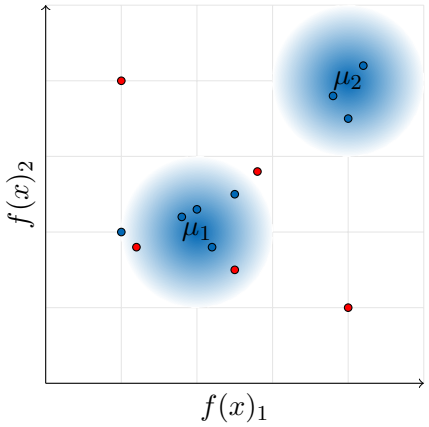


Figure 3.13: The resulting probability model. Note that both clusters centers use the same probability density function.

Chapter 4

Implementation

In the following chapter, the implementation and training of the considered convolutional models and their self-assessment models will be described in detail. All code was written in the Python programming language. Throughout the implementation, we used Keras [12] with Tensorflow [2] as backend, and many functions from the scikit-learn framework [47]. The main goals for the implementation were ...

1. to be able to conduct experiments to gather evidence regarding the proposition (see Section 3.1),
2. to allow others to reproduce the experiments,
3. to able to test a variety of datasets, algorithms and models (as described in Section 3.2) and
4. to be able to evaluate and compare our results to other approaches

In Section 4.1, it will be explained which CNNs were used to test the self-assessmen, why we chose these networks, and on which data they were trained. Afterward, in Section 4.2, it will be described how the framework and the self-assessment model for the selected CNNs was implemented. Ultimately, we will provide some notes on the development process in general.

4.1 Convolutional Neural Networks

This section will introduce the CNNs that were used throughout the experiments.

4.1.1 Training Dataset

We decided to train the base models on the ImageNet 2017 dataset, which is a large scale image dataset and foundation of the ImageNet Large Scale Visual

Recognition Challenge (ILSVRC) [50]. The subset of the ImageNet used in the challenge incorporates 1000 ImageNet categories and about 1.3 million images.

4.1.2 Convolutional Base Models

We decided to use three CNN based on different architectures: Vgg19 [58], Xception [11], and Inception v3 [63]. These model architectures have been chosen because they are well known, and pretrained models for the ImageNet are publicly available for each architecture. This will enable others to replicate the experiments.

4.1.3 Convolutional Ensembles

In addition to the other models, we decided to train a ConSemble as described in Section 2.3.3 based on the three convolutional base models.

4.1.3.1 Preprocessing

Since we did not intend to train the convolutional layers of the network, we first extracted the attribution-maps of the last convolutional layer of each image in the dataset for each of our base models. Training the ConSemble on the activation-maps allows us to increase the batch size, as it is possible to fit more instances into the memory at the same time. This speeds up the training process. Additionally, these activation-maps are a product of the data collection step of the self-assessment (see Section 4.2.1.2). Table 4.1 provides an overview of the layers whose activations we used, as well as the size of the obtained activation-maps.

Table 4.1: Size of Activation- and Attribution-Maps for different network architectures

CNN	Layer Name	Maps	Map Shape
Vgg19	block5_pool	512	7×7
Inception V3	mixed10	2048	8×8
Xception	block14_sepconv2_act	2048	10×10

4.1.3.2 Training

The ConSemble has been trained on an NVIDIA DX-1-Cluster with 8 NVIDIA Tesla V100 GPUs and 512 GB Memory. The hyperparameter optimization was performed using a guided grid search. More sophisticated strategies for neural architecture search include evolutionary algorithms, Bayesian optimization, and reinforcement learning [18]. Due to time limitations, we were unable to perform a more extensive HPO and therefore suspect that the results could be further

improved by investing additional compute. For the same reason, we did not perform dataset augmentation, which also has the potential to create more robust networks and may thus lead to performance gains [23].

Our hyperparameter set included an optional hidden dense layer with various number of neurons, the batch size, the number of epochs, the learning rate, L_1, L_2 and L_1L_2 regularization, dropout regularization [60] and batch-normalization [33]. To limit the size of the search space, we considered only the *ReLU* activation function[1] (see Section 2.1.1) for the hidden dense layer, and the prevalent categorical cross-entropy as loss function. To prevent over-fitting and reduce training time, we implemented early stopping. For weight optimization, the Adam optimization algorithm was used [35].

We chose the largest batch size that would occupy the majority of the 32 GB memory of the Tesla V100, as this enables us to utilize the performance of the GPU. For unknown reasons, we did not experience the close-to-linear speed up that is usually expected when training on multiple GPUs.

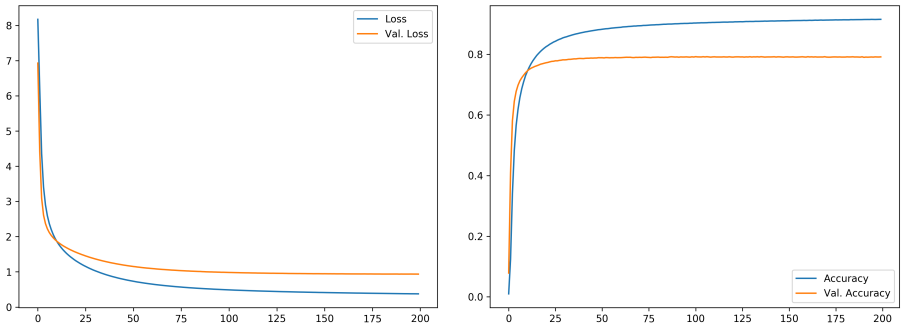
We used random uniform weight initialization with $[-0.05, 0.05]$; Goodfellow et al. claim that when using the rectifier activation function, as we did, the outcomes could be improved by initializing all weights with a small positive value [23]. This ensures that all ReLUs are active at the beginning, which is important considering that the weights of inactive neurons are not adjusted during gradient descent (since the gradient for inactive units is zero).

The best performing network in terms of the validation loss has approximately 5.6 M trainable parameters. The architecture is depicted in Figure 4.2. All hyperparameters not included in the sketch are listed in Table 4.2. An overview of the accuracy and the loss over the course of the training can be found in Figure 4.1.

Table 4.2: Hyperparameters of the ConSemble with the best performance.

Hyperparameter	**Value**
Optimizer	Adam ($\beta_1 = 0.9$ and $\beta_2 = 0.999$)
Batch Size	10.000
Learning Rate	0.000025
Epochs	300
Loss	Categorical Crossentropy
Weight Initialization	Random Uniform $\in [-0.05, 0.05]$
Trainable Parameters	5,610,000

[1] According to [23], the ReLU activation function is an "excellent default choice".



(a) Loss (b) Accuracy

Figure 4.1: Accuracy and categorical crossentropy (loss) during the ConSemble training. The minimum validation loss was reached after 100 Epochs.

4.2 Self-Assessment

4.2.1 Framework

The implementation of the framework, as described in Chapter 3.2 strives toward multiple objectives to achieve the three goals stated in the opening of this chapter. The most important objectives are modularity and performance. The former allows for the rapid implementation of self-assessment methods using different algorithms or features, without having to alter other components. Performance is of the essence as we intend to process huge amounts of data in a strictly limited amount of time.

To met these requirements, we decided to embed the framework into a pipeline-based approach. The steps in the pipeline are connected via interfaces. Each step may be altered as long as this change does not break the compliance with any interface. This approach facilitates modularity and speed, as each pipeline step may be modified or reconfigured without invalidating the results of the preceding steps. Furthermore, when using a pipeline approach, bottlenecks can be identified easily, which allows us to optimize specific components of the system for performance. The pipeline, along with input and output-data (or configurable options) for each individual step, is delineated in Figure 4.3. As you can see, we added a pre- and a post-processing-step. Each step will be described in the following.

Note that the workflow using the framework is not necessarily strictly sequential. Throughout the self-assessment modeling, certain consecutive steps have been reiterated with different parameters; for example, the model training and the evaluation.

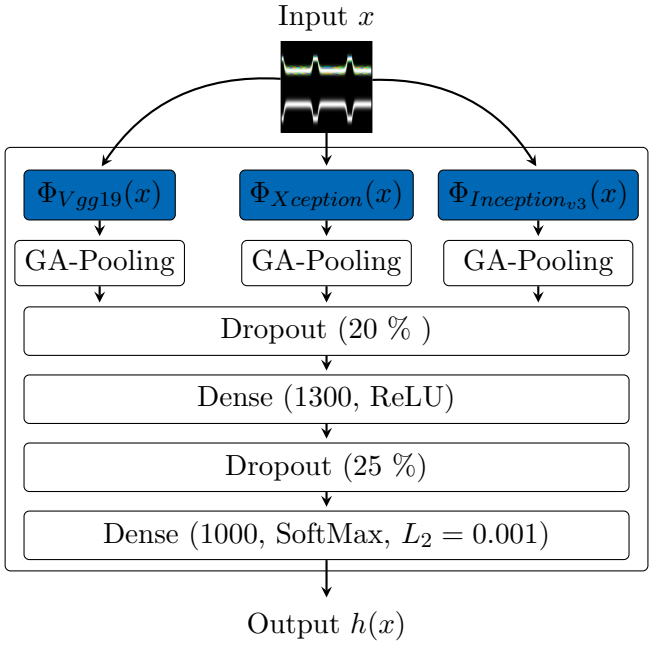


Figure 4.2: Architecture of the ConSemble with the best performance.

4.2.1.1 Dataset Preparation

In the first step, we preprocess the image datasets so that different datasets can be handled in a uniform manner by the framework. This step has to be implemented for each dataset individually.

4.2.1.2 Raw Data Extraction

Once the datasets are in the appropriate format, we can start to gather the data that is required for the self-assessment. We loop through all images in the prepared dataset. For each image, we calculate the activation-maps, the attribution-maps, and the prediction for some CNN and some layer.

As large quantities of data are processed (the ImageNet 2017 training set contains more than 100 GB of compressed images), we split the images into work packages that fit into the memory and may be processed in parallel by multiple workers using dedicated GPUs. The results for individual work packages are aggregated and stored in an HDF5 file.

4.2.1.3 Feature Extraction

In the following step, the different features, as explained in Section 3.3.2, are extracted from the activation- and attribution-maps.

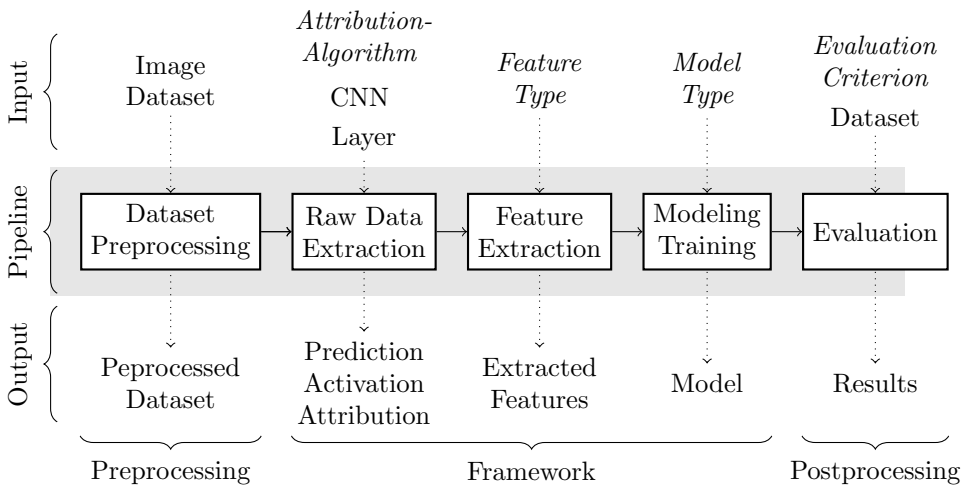


Figure 4.3: Scheme of the pipeline-based implementation of the proposed Framework. The output(s) of each step serve as input to the next step. Each step may require additional configuration and input data and must be finished entirety before the next step can commence. Configurable parameters are emphasized.

For the Xception CNN, this reduces the amount of raw data for the ImageNet Training set from 2.1 TB to about 31 GB for all three feature types[2].

4.2.1.4 Modeling

The next step constructs a model based on the extracted features. The model has to implement a particular interface and is then "trained" by the framework. For the proximity-based statistical model proposed in Chapter 3.3, we estimated the parameters of the Weibull distributions using the libMR provided by [53]. The process creates a self-assessment model for the CNN passed into the pipeline at step two.

4.2.1.5 Evaluation

The evaluation is implemented in a generic way that allows evaluating arbitrary models created by the framework[3], given some evaluation criteria and an HDF5 dataset. The implementation of different evaluation criteria allows us to compare the results to other state-of-the-art approaches that use different evaluation methodologies. The chosen evaluation methodology, as well as the results, will be provided in the following chapter.

[2]The Xception has feature maps of size 10×10. Global Average Pooling reduces these 100 pixels to a scalar value, effectively compressing by the factor 100.

[3]I.e., models that implement the mentioned interface.

4.2.2 Demo

To provide the reader with a demonstration of the proposed self-assessment approach, we deployed a CNN and a self-assessment model into a web application[4]. It allows the user to upload images which will then be processed by the framework. The web application provides softmax classification results as well as confidence scores for the top prediction for comparison. Figure 4.4 provides a screen-shot of the user interface of the demo application. In order to ease the deployment, the application server is containerized in a docker image.

4.3 Development

The source code[5] provides two Python packages. The first one facilitates the training of convolutional ensembles, while the second one provides the core components of the framework. For version control, git has been used. Gitlab allows to create pipelines for continuous integration. We configured it to run unit tests, and to build the documentation and the Python packages after each change in the code.

[4] Available at `https://ma.kondas.de`

[5] Available at `https://git.kondas.de/kkirchheim/ma-code-consemble/`

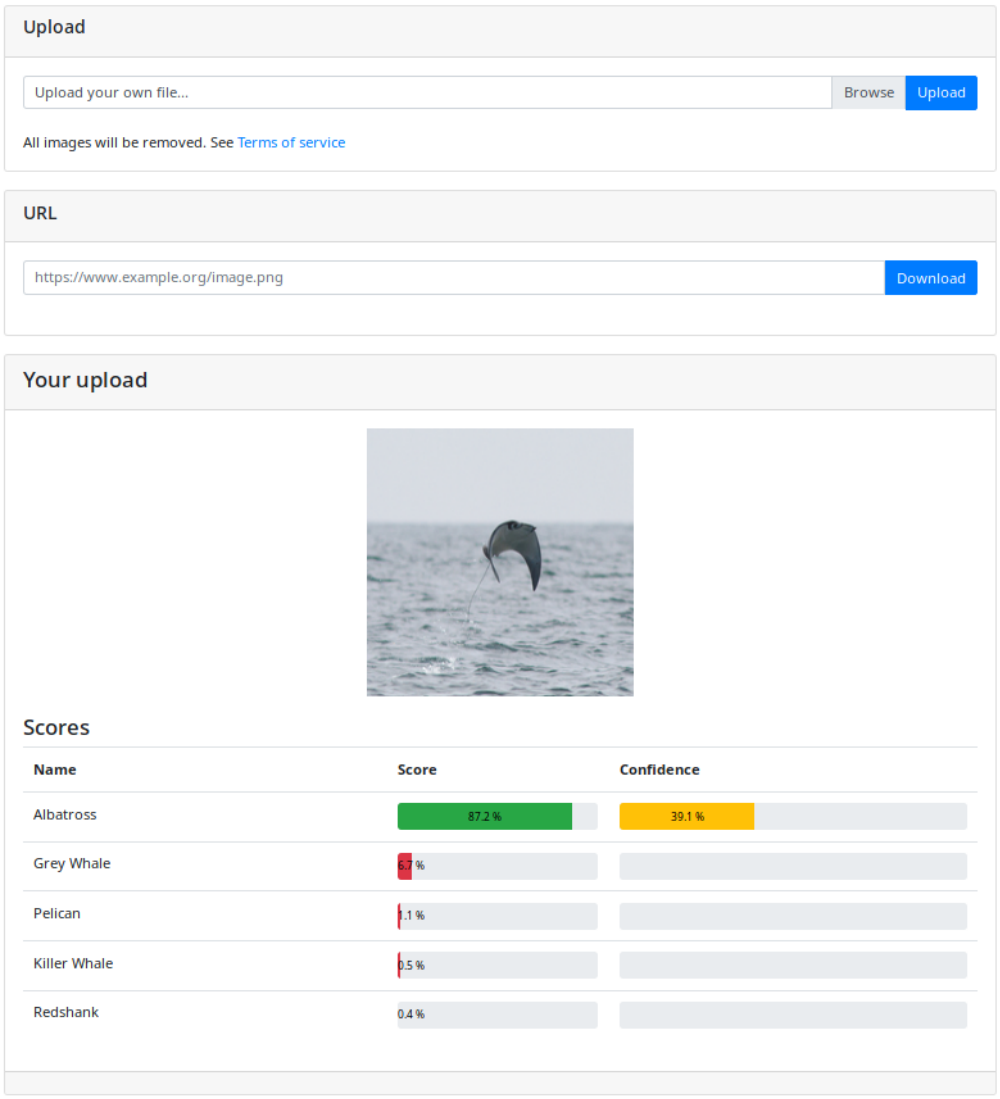


Figure 4.4: User Interface of the Developed Demo Web Application

Chapter 5

Evaluation

In this chapter, the self-assessment approach, based on different features as proposed in Chapter 3, will be evaluated. Recall that the superordinate idea of this thesis is to use attribution-based explanation methods to check if the prediction of a CNN is plausible. We use the activation-maps and attribution-maps as a proxy for the explanation, as they contain all the information that the considered example explanation method - GradCAM - is calculated from. The attribution- and activation-maps provide more information than a GradCAM, since GradCAM only visualized the averages of the weighted activation-maps, thereby discarding information about the specific features that have been detected. The goal of the conducted experiments is to gather evidence with regards to whether it is possible to detect misclassified images by evaluating patterns in the attribution-maps, which are created by the explanation method. The design of our model should allow us to conclude whether the attribution-maps form clusters like the vector embedding do. Additionally, by including models that are based on a combination of activation- and attribution-maps (i.e., self-assessment models based on the Weighted Activation feature), we can evaluate whether it is possible to improve self-assessment beyond self-assessment based exclusively on activation patterns.

In the first section of this chapter, the datasets that were used throughout the experiments will be introduced. Afterward, the performance of the considered CNNs on these datasets will be evaluated, following the evaluation protocol Wehmeier [67] that emphasizes safety aspects. The goal of this pre-evaluation is to put the performance of the recognition systems into perspective before evaluating the self-assessment models for these CNNs. In Section 5.3, the results for the self-assessment models will be presented. The chapter closes with a discussion of the findings.

Table 5.1: Datasets used for the conducted experiments and the images types they contain, as described in Section 2.5

Dataset	Ordinary	Open Set	Adversarial	Fooling
ImageNet-A [31]	✓	×	×	×
ImageNet 2017 [14]	✓	×	×	×
ImageNet 2010 OS [14]	×	✓	×	×
Fooling Images [45]	×	×	×	✓
Mix	✓	✓	×	✓

5.1 Datasets

In this section, the datasets employed in the following evaluation procedures will be described. The datasets have been selected with the intent to cover as many of the image types that may cause errors that were identified during the literature survey (see Section 2.5) as possible. On overview of the datasets is provided in Table 5.1. Due to time limitations, we have not been able to include corrupted and adversarial images.

5.1.1 ImageNet 2017

Our CNNs were trained on the ImageNet 2017 training set [14] (see Section 4.1.1). The validation set of ImageNet 2017 comprises 50,000 images, i.e., 50 images per class. As the ImageNet is the foundation of a challenge for computer vision experts, the test sets are not publicly available. Therefore, we will use the validation set throughout the evaluation as done previously by other [7, 37, 45, 58].

5.1.2 ImageNet-A

ImageNet-A is a small dataset of Natural Adversarial Examples published by Hendrycks et al. [31]. It comprises 7500 Images from 200 ImageNet classes. The classes have been selected such that misclassifications between them would be considered egregious. For instance, confusion among dog races is, in general, considered less critical than confusion among totally unrelated classes, for example, dogs and cars [14].

5.1.3 Fooling Images

We use a dataset of 5000 fooling images provided by [45]. They have been generated for the AlexNet [37] using an evolutionary algorithm or gradient ascend.

Due to time limitations, we were unable to create images crafted explicitly for

the classifiers we tested. However, [45] claims that fooling images generalize well to other neural network architectures, which we could confirm during our experiments, as fooling images would be classified with relatively high confidence in general. Thus, it is assumed that these images will be sufficient for a proof of concept.

5.1.4 ImageNet 2010 OpenSet

As reported by [50], ImageNet 2010 contains 360 classes that were not included in later releases of the dataset. As suggested by [7], the images of these classes can be used as images of unknown classes for classifiers that have been trained on newer versions of the ImageNet. They are suited for this task because they share the general properties of the ImageNet images, e.g., there is usually one object in the center of the image that occupies a large part of it. However, the images depict objects that the classifier can not know.

5.1.5 Mix Dataset

Additionally, we create a composed set of images, from now on referred to as *Mix Dataset*, which includes images from each of the datasets mentioned above. As there are 5000 publicly available fooling images, we drew 5000 images at random from each other dataset and added them to the composition. The entire Mix Dataset contains 20,000 images. Details are listed in Table 5.2.

Table 5.2: Composition of the Mix Dataset. All Images were drawn at random without replacement from the entire datasets.

Dataset	**Images**	**Ratio**	**Used**
Fooling Images	5000	25 %	100 %
ImageNet-A	5000	25 %	66.66 %
ImageNet 2017 Validation	5000	25 %	10 %
ImageNet 2010 OpenSet	5000	25 %	1.6 %
Mix Dataset	20000	100 %	-

5.2 Convolutional Neural Networks

In this section, the CNNs for which self-assessment models were created will be evaluated, emphasizing the ConSemble and safety aspects.

5.2.1 Evaluation Methodology

We aim to follow the evaluation protocol proposed in the ConSemble technical report from Wehmeier [67]. However, the following modifications have been

applied: Wehmeier reported the error rate for all models in order to emphasize the safety perspective. However, accuracy is prevalent in the literature, so we decided to report this metric as a performance measure instead. Consequently, to evaluate the performance of the ConSemble, we compare its Top-1 and Top-5-accuracy to that of its base models. Additionally, the scores for a Model-Averaging (calculated as described in Equation 2.6) ensemble and a Majority-Voting ensemble as baseline ensemble approaches are provided. Those models will not be included in the evaluation of the self-assessment.

5.2.2 Results

In the following, we will report and discuss the results we obtained following the evaluation protocol described above.

5.2.2.1 Accuracy

The Top-1-Accuracy for each network can be found in Table 5.3. The Top-5 accuracy is documented in Table 5.4. We omitted all datasets that contain images of unknown classes (i.e., the Fooling Image, the ImageNet 2010 OpenSet, and the Mix Dataset), as the CNNs are by design unable to predict the correct class or reject an image as unknown.

As we can see, the ConSemble approach achieves a better performance than any of its base models, improving the absolute Top-1-accuracy[1] of the best performing base model by approximately 1.4 %. However, the ConSemble seems to be more likely to misclassify natural adversarial images than the Inception v3 or the Xception architecture, which could be an indication of overfitting or inferior generalization capabilities. This observation will be discussed further in Section 5.4.3.3. As we can see, the Xception model has lower accuracy on the training set, and a higher accuracy on the Validation set compared to the Inception v3. This indicates that the Xception network generalizes better than the Inception v3. Additionally, this Xception has the highest accuracy on the natural adversarial examples. The ConSemble performs better than Model-Averaging or Majority-Voting, both of which are unable to increase the overall performance. The fact that the baseline ensembles are unable to improve the predictions could have multiple causes. We suspect the comparably low performance of the Vgg19 and a lack of diversity to be the most probable reasons (i.e., too few models were included which do not make independent errors, see Section 2.3). It should be mentioned that the implementation of majority voting that we chose assigns confidence 1.0 to the instance that has the most votes, which entails that the Top-1 and the Top-5 accuracy are identical in that case[2].

[1] Sometimes the performance gain is reported in terms of a *relative* improvement over some reference. In contrast, *absolute* accuracy denotes the actual performance gain on the dataset.

[2] If all three models disagree on the prediction, we take the prediction with the highest softmax output

Table 5.3: Top-1-Accuracy

Model	ImageNet Train	ImageNet Val	ImageNet-A
Vgg19	71.62 %	64.67 %	1.21 %
Inception v3	88.96 %	76.20 %	4.24 %
Xception	88.39 %	77.51 %	**4.26 %**
Model-Averaging	88.59 %	76.49 %	3.68 %
Majority-Voting	88.96 %	76.26 %	4.24 %
ConSemble	**91.82 %**	**79.10 %**	3.78 %

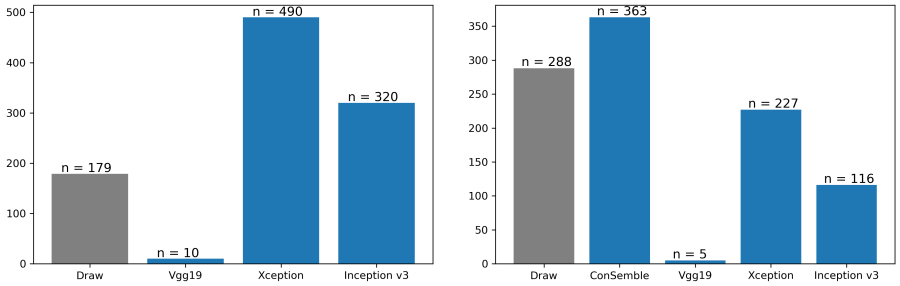
Table 5.4: Top-5-Accuracy

Model	ImageNet Train	ImageNet Val	ImageNet-A
Vgg19	90.54 %	85.77 %	5.57 %
Inception v3	98.61 %	92.93 %	13.81 %
Xception	98.31 %	93.74 %	**14.34 %**
Model-Averaging	98.17 %	92.26 %	11.40 %
Majority-Voting	88.96 %	76.24 %	4.24 %
ConSemble	**99.16 %**	**94.76 %**	14.20 %

Furthermore, we evaluated which model obtained the highest top-1-accuracy on a per-class basis. The results are displayed in the bar plots in Figure 5.1. The ConSemble surpasses the performance of its base models for 36.3 % of classes and performs as good as the best base model for 10.9 % of classes.

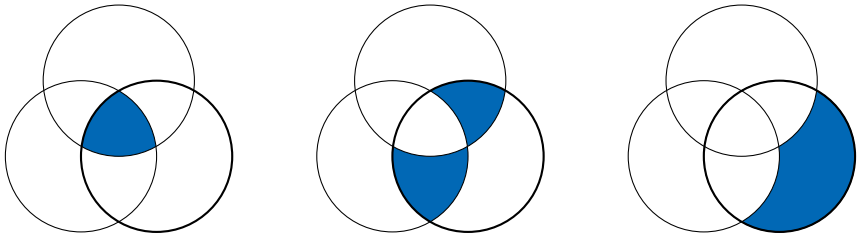
5.2.2.2 Base Model Errors

Following the evaluation protocol of [67], the *common*, *shared* and *unique* errors of the model have been analyzed for the ImageNet 2017 validation set. *Common* errors are errors that occur in all base models. *Shared* errors occur in at least



(a) Base-Models (b) Base-Models and Consemble

Figure 5.1: Number of classes for which a CNN dominated all other networks.



(a) Common (b) Shared (c) Unique

Figure 5.2: Intersection of errors for different Models

two of the base models, but are not common. *Unique* errors are errors that are being made by exactly one model. For ease of understanding, the concept of intersecting errors is visualized in Figure 5.2. We did not consider the concrete prediction into this evaluation but solely evaluated whether the classification was correct or incorrect. For instance, if an image of a dog is misclassified two times, once as a car and once as a cat, this still counts as a shared error, even though the predictions differ. The results for the base models are listed in Table 5.5. The Vgg19 Model is responsible for the majority (74.61 %) of the unique errors within the base models, which is to be expected considering its comparably weak performance.

Table 5.5: Intersection errors of base models for the ImageNet 2017 Validation Dataset

Model	**Errors**			
	Common	**Shared**	**Unique**	**Total**
Vgg19	7762	3783	6103	17647
Xception	7762	2619	862	11243
Inception v3	7762	2890	1214	11866
Aggregated	7762	4646	8179	

5.2.2.3 Ensemble Errors

As described in Section 2.3, ensemble methods are used to compose classifiers that surpass the performance of each base model. In order to do so, ensembles have to correct common, shared, or unique errors of their base models. On the other hand, they should introduce as few new errors - i.e., errors that have not been made by any of its base models - as possible. Table 5.6 lists the newly introduced errors and the corrected errors for each considered ensemble technique.

By design, the voting-based ensemble is unable to correct common errors, but

can also not introduce new errors. The Ensemble-Averaging model did not introduce new errors. However, it corrected 20 of the errors made by all of its base models. The ConSemble approach corrected 434 common errors but also introduced 98 new errors.

These results again highlight an obstruction for the deployment of ConSembles (and non-voting based ensemble methods in general) in safety-critical applications. While the ensemble technique increases the performance beyond the performance of the base models, and thereby reduces the number of overall errors, it also carries the potential to introduce new errors. This behavior implies that, even if tests verify that some convolutional neural networks meet certain safety requirements, these properties can not necessarily be transferred to a convolutional or mean-ensemble of these networks.

Table 5.6: Introduced and corrected errors for the ensemble models on the ImageNet 2017 validation set.

Model	**New Errors**	**Corrected Errors**		
		Common	**Shared**	**Unique**
Majority-Voting	0	0	1256	6965
Ensemble-Averaging	0	20	1650	7166
ConSemble	98	434	2331	7474

5.2.2.4 Model Relevance

In [67], Wehmeier investigated which base-model was favoured by the ConSemble by analysing the weights in the FC layer. The method can be compared to the method vanilla CAM (as described in Section 2.4.3.1) uses in order to determine the relevance of features. This straight forward approach is applicable because their ConSemble architecture only uses a single dense layer. In our case, this method can not be employed since we use multiple FC layers. Therefore, in the following, we propose a method to estimate the relevance of base models of a convolutional ensemble.

We can approximate the relevance of specific inputs across multiple dense layers by utilizing the approach GradCAM uses to calculates the relevance of inputs. Consider a convultional ensemble with a set of base models M, and a set of images X. Every base model $m \in M$ feeds some embedding vector $\phi(x) = y = [y_1, ..., y_n]$ where $y \in \mathbb{R}^n$ into the convolutional ensemble. For an image $x \in X$, we can estimate the relevance $R^c_{y_n}$ of each element in y_n in the embedding vector y using the Equation 3.1. Afterwards, we weight each activation with its relevance and sum up the absolute values for each model m (Equation 5.1). Weighting the activations with their sensitivity

Table 5.7: Estimated relevance of base models for the decision of the ConSemble. We approximated the relevance of the base models for each instance in the ImageNet sets and calculated the mean and the standard deviation σ — all values in percent.

Model	**Average Model Relevance**	σ
Imagenet 2017 Val		
Vgg19	23.26	7.87
Xception	26.65	3.92
Inception v3	50.08	6.76
Imagenet 2017 Train		
Vgg19	23.31	7.81
Xception	26.84	3.89
Inception v3	49.84	6.68

values is motivated by the explanation given in Section 3.3.2.3. This resulting relevance R^c_m approximates the impact m had on the ConSemble's decision.

We then scale this value, by dividing it by the sum of all R^c_m (Equation 5.2). This scaling ensures that the model relevance values sum up to one. Finally, we calculate the mean of the model relevance scores over all images in the given set to obtain the approximated relevance of the entire model over the dataset.

$$R^c_m = \sum_n |y_n R^c_{y_n}| \quad \text{for each } m \in M \tag{5.1}$$

$$\hat{R}^c_m = \frac{R^c_m}{\sum_{i \in M} R^c_i} \tag{5.2}$$

The results are listed in Table 5.7[3]. Apparently, the ConSemble favors the Inception v3 and assigns an approximately similar weight to the output of the other convolutional base models. The tendency to assign higher relevance to base models with better performance has already been observed by Wehmeier.

5.3 Self Assessment

In this section, the results of the experiments conducted to evaluate the self-assessment methodology proposed in Chapter 3 will be presented. All results apply to a self-assessment model based on K-Means clustering with five cluster centers since we found that it provided the best performance tradeoff between

[3]The results should be taken with a grain of salt, as the design of methods to estimate the relevance of base models for a convolutional ensemble is not in the scope of this thesis. As of now, we have not verified the results on other models.

different datasets. Additional results for other self-assessment models can be found in appendix A.

The following section will introduce the evaluation methodology, while the subsequent sections will present the results we obtained by applying this methodology for each dataset individually.

5.3.1 Evaluation Methodology

Failure prediction can be seen as binary classification problem. The self-assessment model predicts whether a prediction is either correct or incorrect (i.e. *one* or *zero*, *true* or *false*, *positive* or *negative*). Usually, the classifier outputs a continuous value between zero and one that estimates the probability a failure (or the confidence) and then applies a threshold to reduce the continuous to a binary value. [19]. As the choice of this threshold depends on the desired properties of the systems, scores are usually evaluated for varying thresholds. In our setup, the *positive* class denotes correctly classified images. Therefore, a high confidence value corresponds to a high probability for the classifier's prediction being correct.

Since there is, to the best of our knowledge, no prevalent evaluation protocol for self-assessment models, we will evaluate our models regarding multiple evaluation criteria and report the results for all of them. In the following, the evaluation criteria that we use will be introduced.

5.3.1.1 Receiver Operator Characteristic

A means to visualize and evaluate the performance of binary classifiers with continuous output is the Receiver Operating Characteristic [19]. This method has been used in several publications concerning self-assessment [30]. The graph characterizes the dependency between the hit rate (True Positive Rate) and the false alarm rate (False Positive Rate). This is, in general, more meaningful than the plain accuracy. The ROC is constructed by gradually increasing the threshold at which a prediction is treated as belonging to the positive class, while plotting the resulting FPR on the x-axis against the TPR on the y-axis. The choice of TPR and FPR (or the threshold) is usually a tradeoff. Which threshold is to be preferred depends on the requirements of the specific use case. For instance, in a high-security context where false-negatives can not be tolerated (consider, for example, the search for weapons at airports), one should choose a low threshold, so instances have to obtain low confidence (i.e., high probability of not carrying weapons) to be classified as negatives. While this will lead to a higher hit rate (more people that actually carry weapons are detected), it will also increase the number of false alarms (falsely accusing people of carrying weapons). In order to compare the characteristics of two systems,

they are usually rated by the area under the ROC or AUROC for short. An AUROC of 0.5 corresponds to random guessing.

For each of the self-assessment models (one model per feature and per CNN), the ROC curve and AUROC scores will be provided and compared to softmax-thresholding as proposed by [30].

5.3.1.2 F_1-Score

Several publications plot the F_1-score against the threshold [7, 51] for the evaluation. The F_1-score is calculated as

$$F_1 = \frac{\text{true-prositive}}{\text{true-positive} + \text{false-negative} + \text{false-positive}} \tag{5.3}$$

A caveat of the F_1-score is that is assigns equal weight to false-positives and false-negatives, which may be inappropriate, depending on the use case.

5.3.1.3 Accuracy

As mentioned earlier, images from the fooling image dataset or the ImageNet 2010 OpenSet subset can, by design, never be classified correctly by the CNNs. This entails that there can be no true-positives (i.e., correctly classified images), which makes it infeasible to determine the ROCs or F_1-scores. Therefore, we will plot and compare the accuracy against a threshold for these datasets, as others did [7]. The conclusions drawn from these diagrams have to be taken with a grain of salt because the accuracy is inflated by the high proportion of negative examples [7]. Contemplating the fooling image dataset, a model that outputs 0.0 confidence, regardless of the input, would achieve 100 % accuracy at any threshold. Thus, high accuracy for fooling images or open images alone can not serve as an indicator of reliable self-assessment capabilities, which is one of the reasons that motivated us to create the Mix dataset.

5.3.2 ImageNet 2017

In this section, the results for the ImageNet 2017 validation set will be presented. We will evaluate the performance of the self-assessment models utilizing the ROC and the F_1-scores.

5.3.2.1 Vgg19

Figure 5.3 depicts the ROCs of the self-assessment models for the Vgg19. None of the self-assessment models exceeds the softmax-score, but all models surpass random guessing. The performance of the models using the relevance scores is higher than the performance of the model based on the activations only.

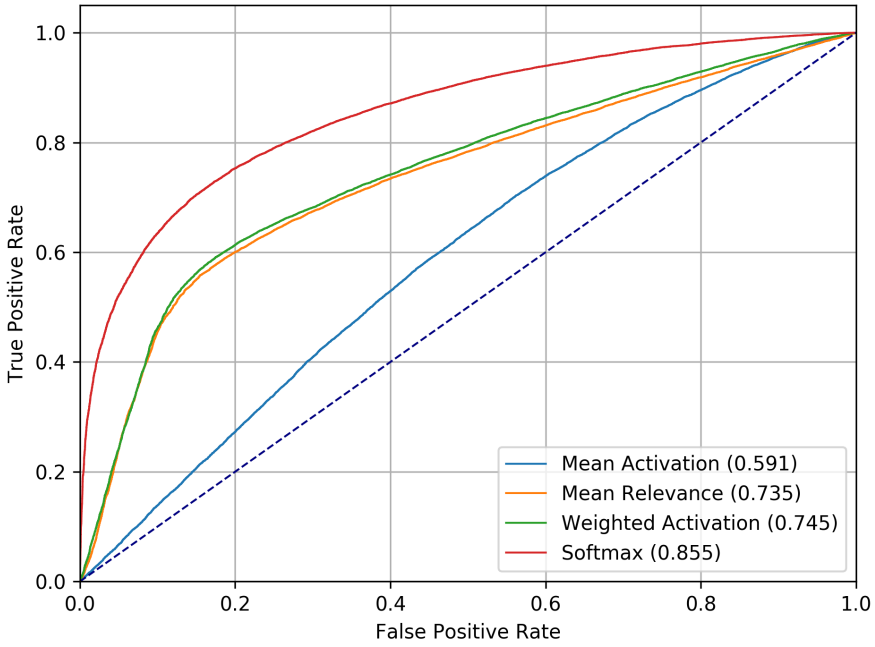


Figure 5.3: Vgg19: ROC for the ImageNet validation set

The F_1-scores for varying thresholds are depicted in Figure 5.4. The results resemble the results of the ROCs. The graphs for the Mean Relevance and the Weighted Activation are similar. The self-assessment models F_1-scores degrade earlier than the performance of the baseline; however, the performance of the Mean Activation-based model falls at a slower rate compared to the other models and even surpasses the other models at a threshold of 0.75.

5.3.2.2 Inception v3

Figure 5.5 depicts the ROCs of the self-assessment models for the Inception v3. Again, none of the self-assessment models surpasses the baseline, but all models exceed random guessing by a wide margin. The model based on the Weighted Activation yields the best performance, beating the models based on activation or relevance alone. Compared to the Vgg19, the Mean Activation yields better results in terms of the AUROC. However, the Mean Activation-based models ROC slightly surpasses the ROC of the Weighted Activation-based model for higher thresholds.

Figure 5.6 displays the F_1-scores. Again we observe that the performance of the model based on the Mean Activation degrades slower for rising thresholds.

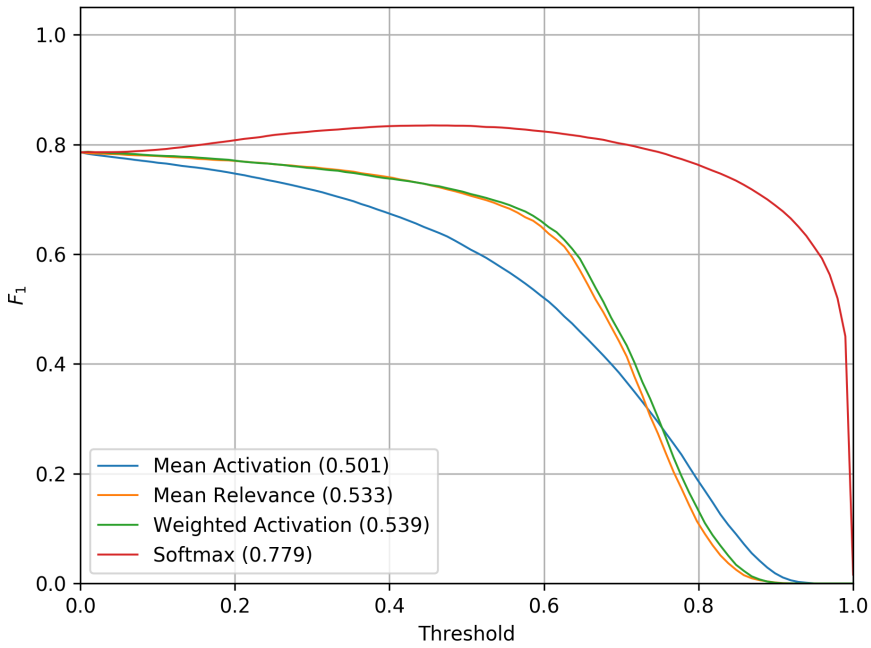


Figure 5.4: Vgg19: F_1 curves for the ImageNet validation set

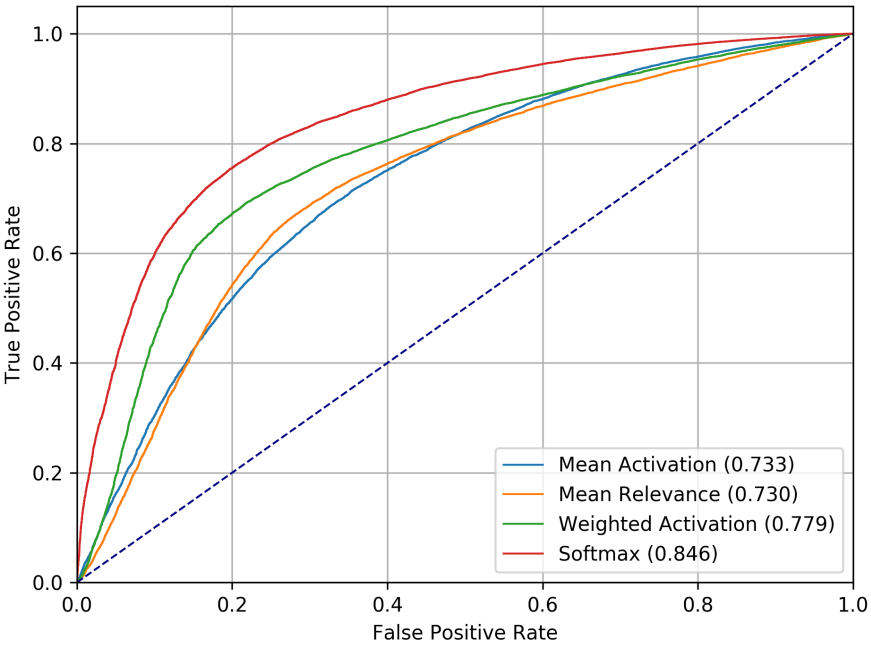


Figure 5.5: Inception v3: ROC for the ImageNet validation set

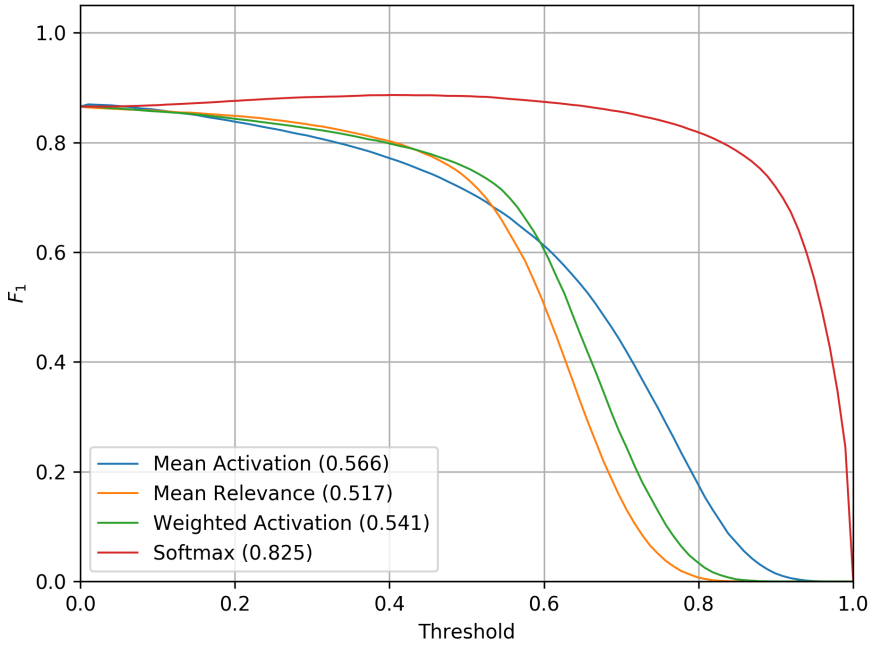


Figure 5.6: Inception v3: F_1 curves for the ImageNet validation set

5.3.2.3 Xception

Figure 5.7 depicts the ROCs of the self-assessment models for the Xception CNN. The results are similar to the ones for the Inception v3. All models surpass random guessing, while none exceeds the baseline. The model combining the relevance and the activation yields the best AUROC score of the self-assessment models, but the ROC of the model based on the activation alone is slightly better for FPRs > 0.45.

The corresponding F_1-scores are provided in Figure 5.8. In general, they support the findings of the ROC-based evaluation. Similar to the Inception v3 and the Vgg19, the scores of the Mean Activation-based model degrade at a slower rate compared to models based on other features. However, the model based on the Mean Relevance slightly surpasses the other models for lower thresholds, which does not quite suit the results of the ROC.

5.3.2.4 ConSemble

The ROCs of the self-assessment models regarding the ConSemble are presented in Figure 5.9. Again, all models surpass random guessing but do not exceed the softmax-thresholding-baseline. Similar to Vgg19, Inception v3, and Xception, the self-assessment models that include the relevance perform better than the

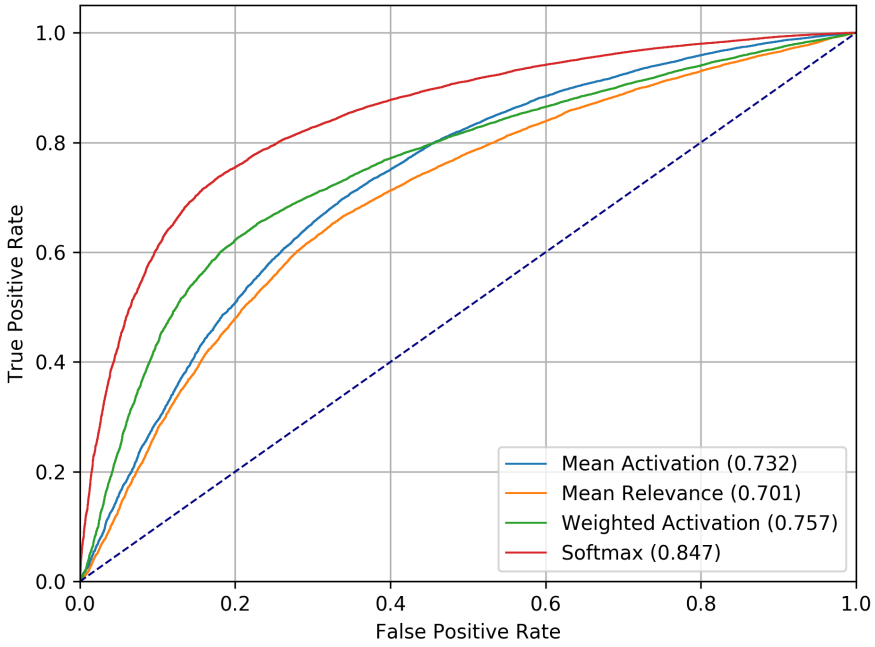


Figure 5.7: Xception: ROC for the ImageNet validation set

model based on the activations only. Compared to the results obtained for other CNNs, we notice the following: while the AUROC of the self-assessment models based on the relevance is higher than for the other CNNs, the AUROC of the Mean Activation is lower than for the Xception and the Inception v3 architecture. Again, the ROC of the Mean Activation-based model slightly surpasses the ROC of the relevance-derived models for high FPRs.

The corresponding F_1-scores are provided in Figure 5.10. While the relevance-based models initially dominate the model based on the Mean Activation feature, the later excels for thresholds > 0.6, because its performance degrades at a lower rate. The graphs of the relevance-based models approach zero for thresholds > 0.8. This indicates that very few of the predictions in the ImageNet validation set receive a confidence score larger than this value.

5.3.2.5 Evaluation

The results show that the self-assessment, while inferior to the softmax-thresholding, continually surpasses random guessing. We observe that for the ImageNet validation set, the Weighted Activation-based model performs better than the Mean Activation- or the Mean Relevance-based ones.

We conclude that self-assessment models that combine the activation and the

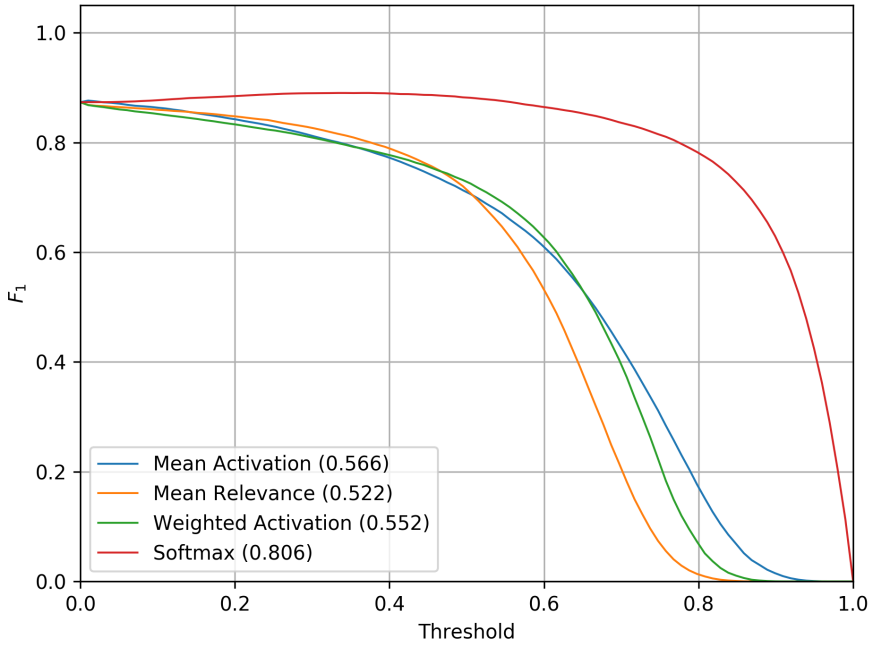


Figure 5.8: Xception: F_1 curves for the ImageNet validation set

relevance of neurons can surpass the performance of self-assessment models based on patterns in the activation alone. Furthermore, we observed that the relevance-based models are surpassed by the Mean Activation-based model for larger thresholds or higher FPR. This is caused by the fact that the relevance based models assign lower confidence scores in general. We deduce from this behavior that the attribution values for unseen instances tend to be further apart from their nearest cluster center than the activation values. This suggests that, for (unseen) instances of a particular class, the relevance the classifier assigns to neurons varies to a higher degree than the activation of those neurons. The 2D-Projection of the activation and attribution-clusters provided in Figure 3.3 and 3.5 substantiate this conclusion, since the attribution-clusters appear to be more intertwined, indicating that the attributions are more scattered.

5.3.3 ImageNet-A

In this section, the results for the ImageNet-A, that contains natural adversarial images will be presented. We omit the F_1-scores for this dataset, since the scores are low in general and do not contribute to the general findings of this section.

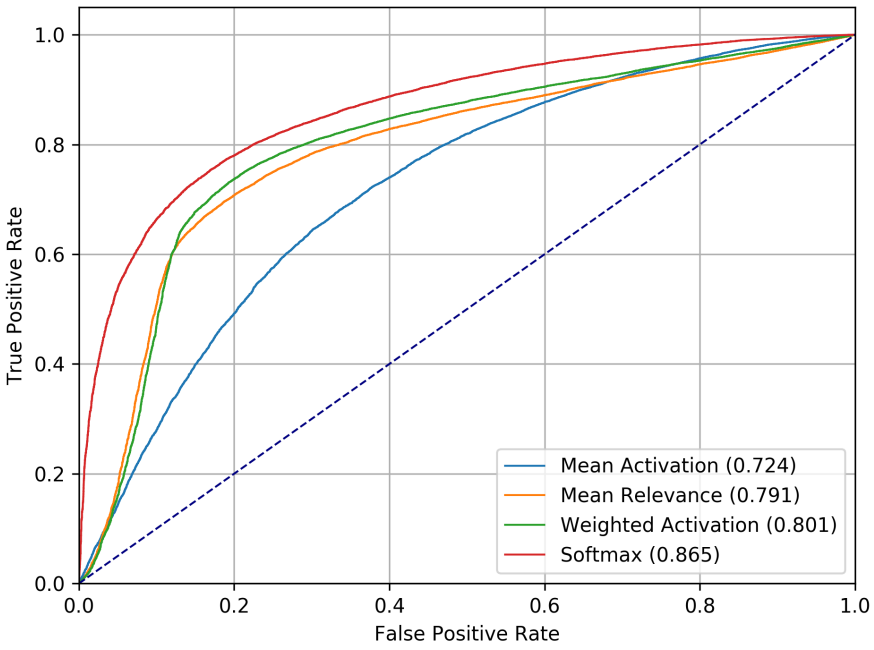


Figure 5.9: ConSemble: ROC for the ImageNet validation set

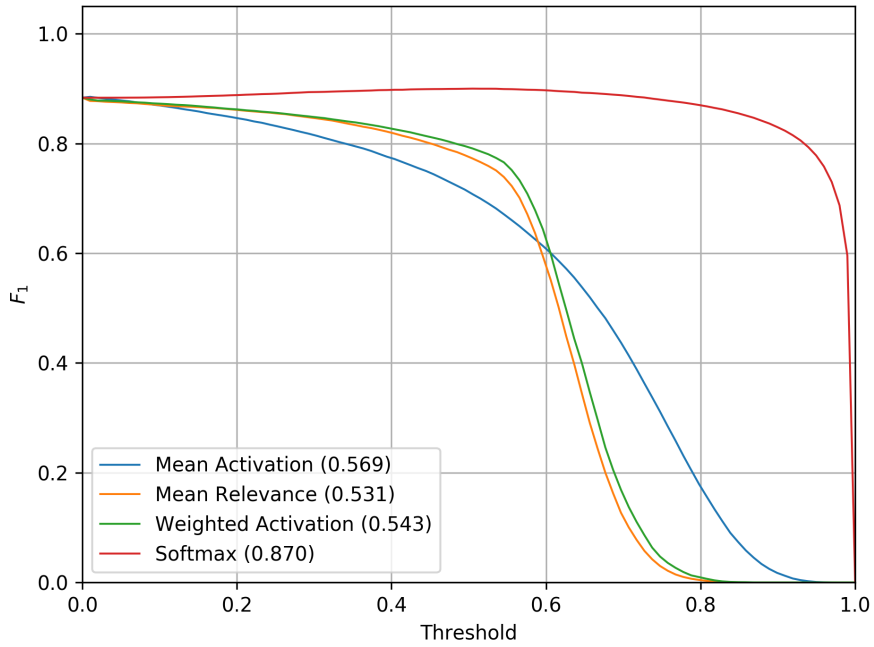


Figure 5.10: ConSemble: F_1 curves for the ImageNet validation set

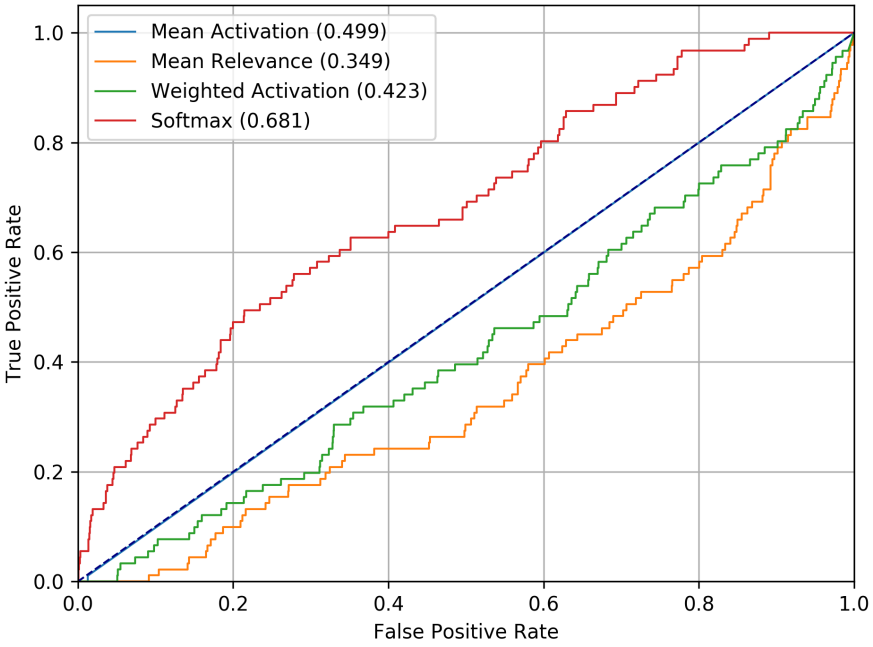


Figure 5.11: Vgg19: ROC for the ImageNet-A

5.3.3.1 Vgg19

Figure 5.11 depicts the ROC of the self-assessment models for the Vgg19. The performances of the self-assessment models deteriorate below random guessing. The softmax-thresholding baseline, on the other hand, surpasses random guessing by a wide margin.

5.3.3.2 Inception v3

Figure 5.12 depicts the ROC of the self-assessment models for the Inception v3. Again, the performance of the self-assessment models is surpassed by random guessing. Compared to the Vgg19, the softmax-thresholding presents a lower AUROC but still exceeds random guessing. While the Mean Activation-based score is close to random guessing, the self-assessment models that employ the relevance exhibit an even weaker performance.

5.3.3.3 Xception

Figure 5.13 depicts the ROC of the self-assessment models for the Xception CNN. The results are similar to the ones obtained for the Inception v3 recognition system, except that the Mean Activation based model performs slightly better than random guessing. Again the models employing the relevance perform significantly worse compared to random guessing.

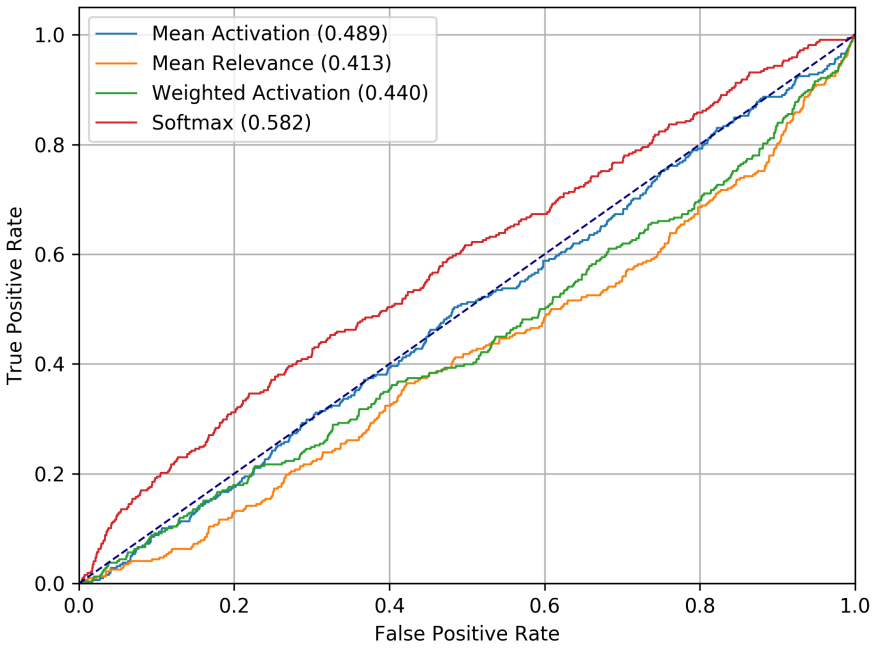


Figure 5.12: Inception v3: ROC for the ImageNet-A

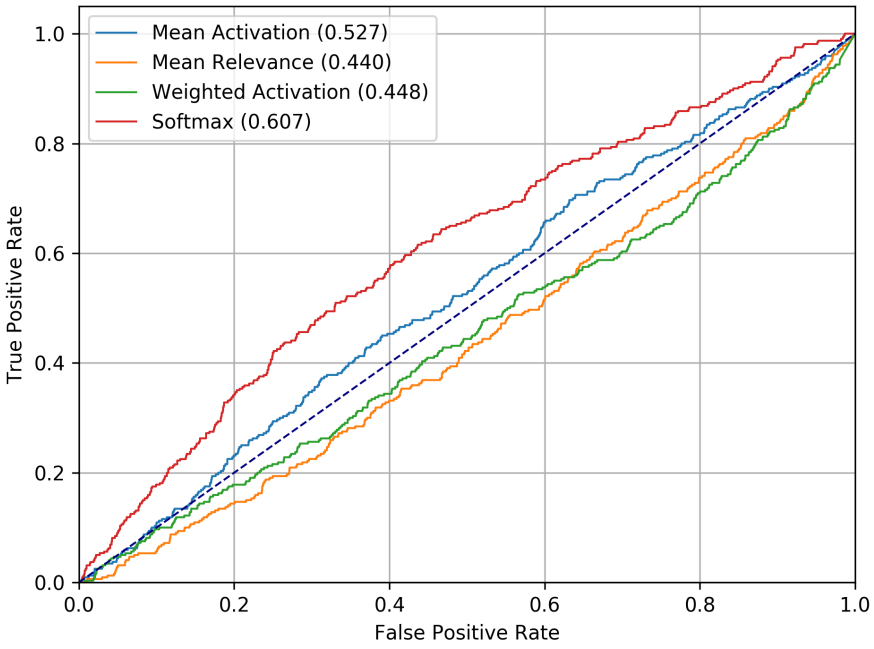


Figure 5.13: Xception: ROC for the ImageNet-A

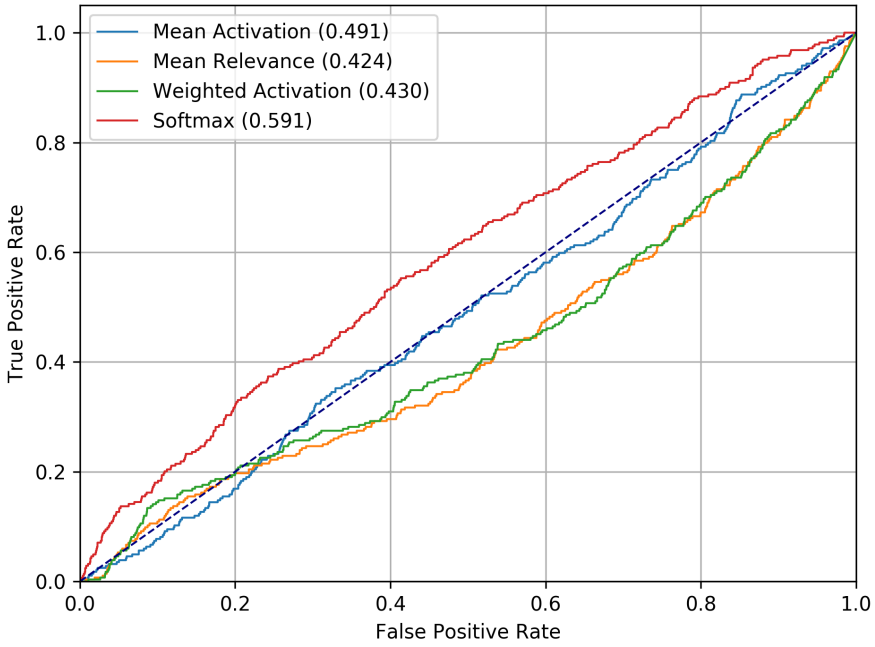


Figure 5.14: ConSemble: ROC for the ImageNet-A

5.3.3.4 ConSemble

Figure 5.14 depicts the ROC of the self-assessment models for the convolutional ensemble. The results are similar to the ones obtained for the Inception v3 and the Xception CNN. While the model based on the Mean Activation is close to random guessing, the performance of the models based on Mean Relevance and the Weighted Activation is significantly lower.

5.3.3.5 Evaluation

In summary, the self-assessment models fail to detect natural adversarial examples, especially models that rely on the relevance scores. However, the design of our model allows us to draw conclusions from this behavior. For the ImageNet-A, the activation of neurons, as well as their relevances, are similar to the activations and relevances of correctly classified images observed during training. This implies that the convolutional layers tend to map natural adversarial images close to the cluster centers. Moreover, the relevance scores do not seem to exhibit any anomalies. This leads us to the conclusion that the feature extraction in the convolutional layers of a CNN is responsible for the weak performance of recognition systems for natural adversarial images, and not the dense layers. In the feature space, the natural adversarial images "look" similar to the correctly classified images observed during the training. This ap-

plies even more to the relevances since the performance for models based on the Mean Relevance, and the Weighted Activation is significantly lower than random guessing.

Nguyen et al. [45] stated that the reason for the apparent inability of CNNs to make reliable predictions for natural adversarial images is their over-reliance on color and texture, and their disregard of other cues like, for example, the shape of the detected object. Therefore, we conclude that the features extracted by the convolutional layers are, in general, those features that Nguyen deems responsible for the failing of CNNs for natural adversarial images.

These findings support the premiss of [67], who assumed that the majority of the learning process takes place in the convolutional layers. Consequently, when trying to remedy the weak performance of CNNs regarding natural adversarial images, it seems reasonable to focus on the features extraction process.

5.3.4 Fooling Images

In this section, the results for the Fooling Image Dataset will be presented. As mentioned earlier, it is not feasible to calculate the ROC or F_1-scores, because the dataset only contains images that can by design not be classified correctly by the recognition systems. Therefore, we will use the Accuracy-against-Threshold plots to evaluate the models for this dataset.

5.3.4.1 Vgg19

The accuracy of the self-assessment models for the Vgg19 CNN is depicted in Figure 5.15. All models surpass random guessing, as well as the softmax-thresholding-baseline. Mean Activation and Weighted Activation achieve similar performance.

5.3.4.2 Inception v3

The accuracy of the self-assessment models for the Inception v3 CNN is depicted in Figure 5.16. Unlike for the Vgg19, the Mean Activation exceeds all other models by a wide margin. In the case of the Mean Activation, about 90 % of the fooling images receive confidence < 0.2. The self-assessment based on the relevance only, however, achieves lower accuracies than the other self-assessment models, but always outperforms the baseline.

5.3.4.3 Xception

Figure 5.17 depicts the accuracy of the self-assessment models for the Xception CNN for varying thresholds. Compared to the Inception v3, the softmax-thresholding performs slightly better, while the performance of the model based

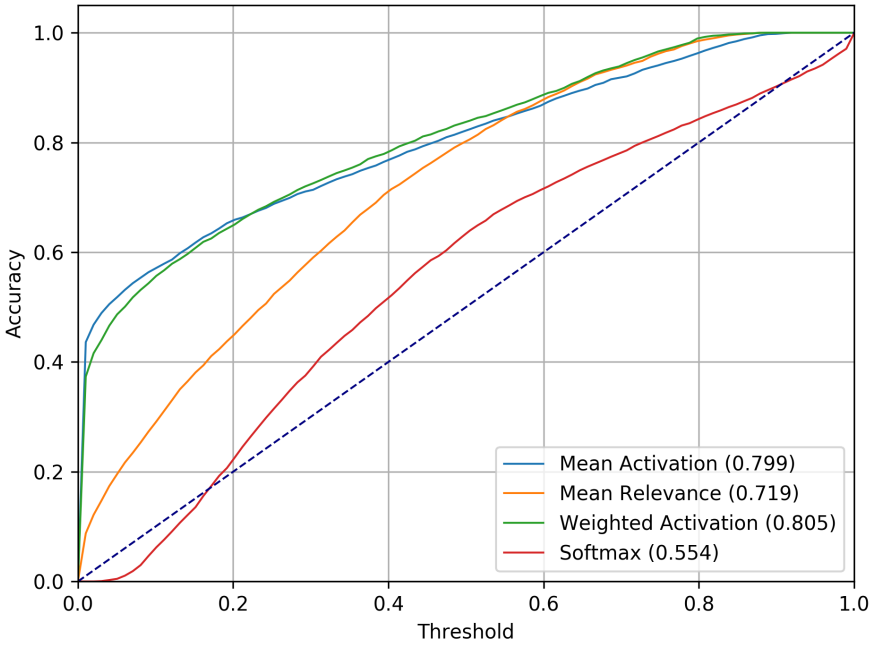


Figure 5.15: Vgg19: Accuracy curves for the Fooling Images

on the Mean Relevance feature drops. As a result, the Mean Relevance-based model is surpassed by the baseline for lower thresholds. The accuracies of the models employing the neural activation resemble the performance of the corresponding models for the Inception v3.

5.3.4.4 ConSemble

Figure 5.18 depicts the accuracy characteristic of the self-assessment models for the convolutional ensemble. The performance of the self-assessment exceeds the performance of the other CNNs. The Mean Activation rejects fooling images exceptionally well, approaching 1 for thresholds > 0.1, indicating that none of the fooling images receive higher confidence. The self-assessment based on the relevance only is again surpassed by the baseline for lower thresholds, while the performance of the model combining activation and relevance resides between the other two classifiers.

5.3.4.5 Evaluation

In summary, we find that the self-assessment models detect fooling images with higher accuracy than the softmax-thresholding. Models that include the neural activation (i.e., the Mean Activation and the Weighed Activation) achieve significantly higher scores than the baseline. Especially the self-assessment model

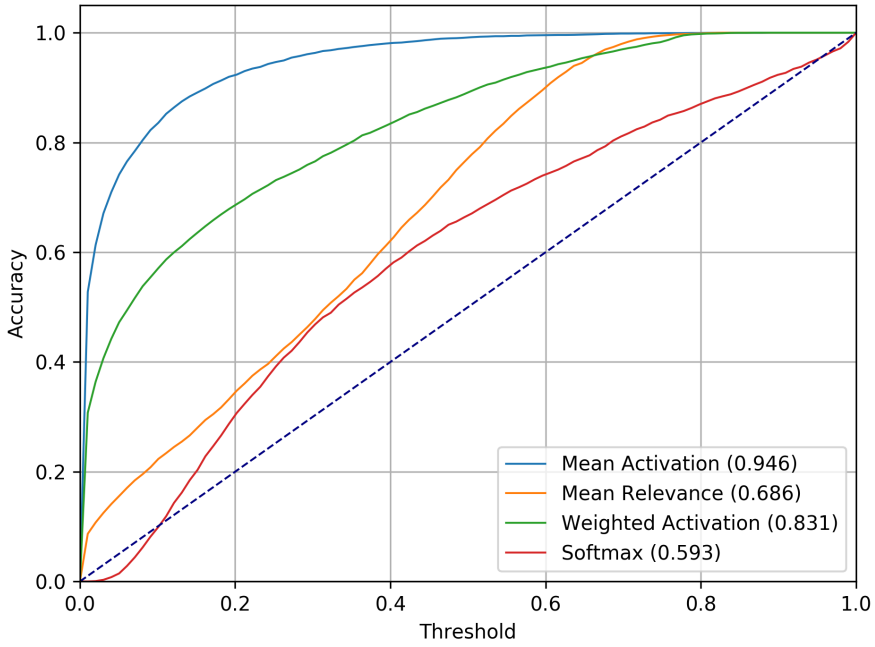


Figure 5.16: Inception v3: Accuracy curves for the Fooling Images

of the ConSemble seems to be resistant to this type of image.

We conclude that the activation-maps extracted from fooling images significantly differ from the activation-maps of correctly classified images observed during the training. We suspect that this is caused by missing contextual cues, which can typically be found in the background of an image of a particular class. The fooling images resemble patterns that are similar to the textures of the mimicked object, but they do not include other textures usually present in images of that class. This holds for the attribution-maps as well, but to a lesser degree. Still, the overall performance of attribution-based self-assessment continually surpasses the performance of random guessing and in general, also the performance of softmax-thresholding.

5.3.5 ImageNet 2010 OpenSet

In this section, the results for the ImageNet 2010 OpenSet dataset will be presented. Similar to the fooling images, we will evaluate the accuracy regarding varying thresholds.

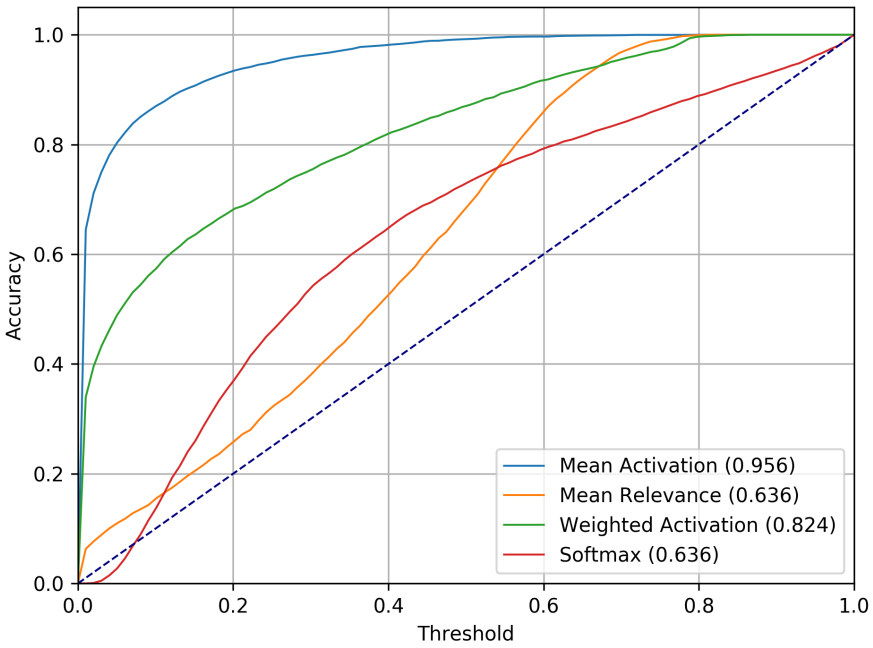


Figure 5.17: Xception: Accuracy curves for the Fooling Images

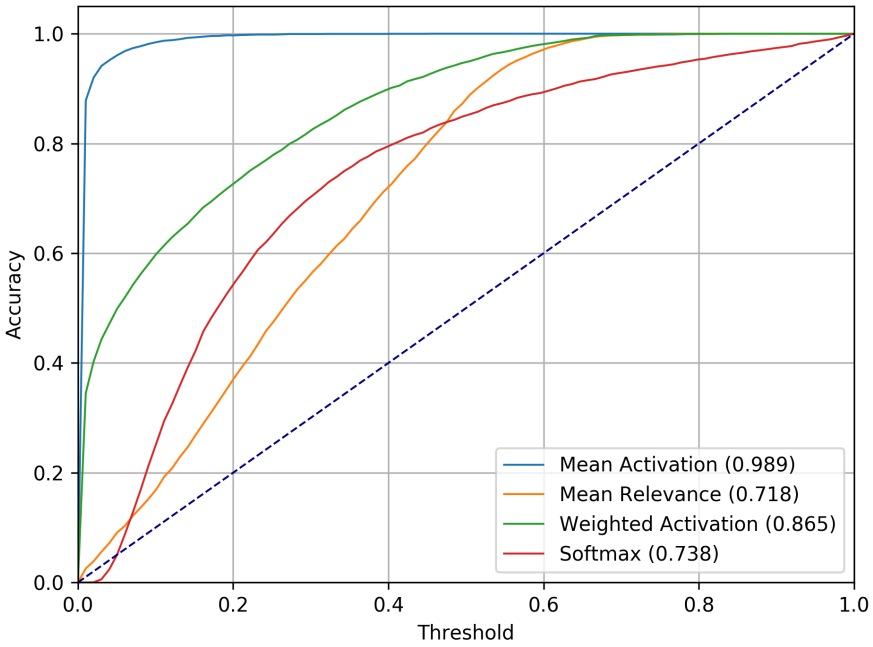


Figure 5.18: ConSemble: Accuracy curves for the Fooling Images

5.3.5.1 Vgg19

The accuracy of the self-assessment models for the Vgg19 CNN is depicted in Figure 5.19. All self-assessment models surpass random guessing. Models incorporating the relevance also outperform the baseline.

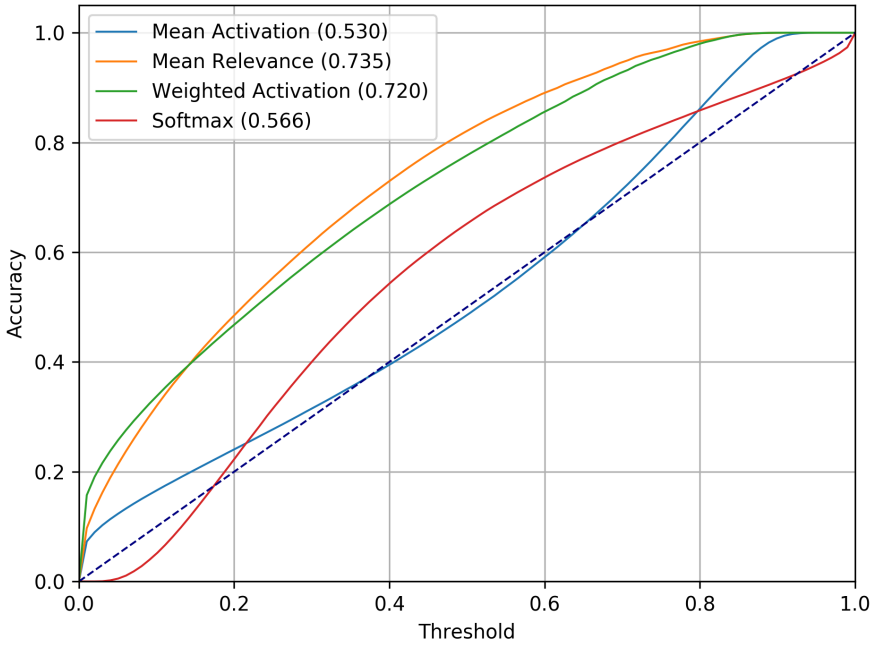


Figure 5.19: Vgg19: Accuracy curves for the ImageNet 2010 OpenSet

5.3.5.2 Inception v3

The accuracy of the self-assessment models for the Inception v3 is depicted in Figure 5.20. All self-assessment models exceed random guessing, as well as the softmax-thresholding by a wide margin. Contrary to the results for the Vgg19, models using features from the activation-maps yield the better performance.

5.3.5.3 Xception

Figure 5.21 presents the results of the self-assessment models for the CNN based on the Xception architecture. Analog to the fooling images presented in the previous chapter, the softmax-thresholding achieves slightly higher performance compared to the Inception v3 CNN. The accuracy of the self-assessment models is similar to the results observed for the Inception v3.

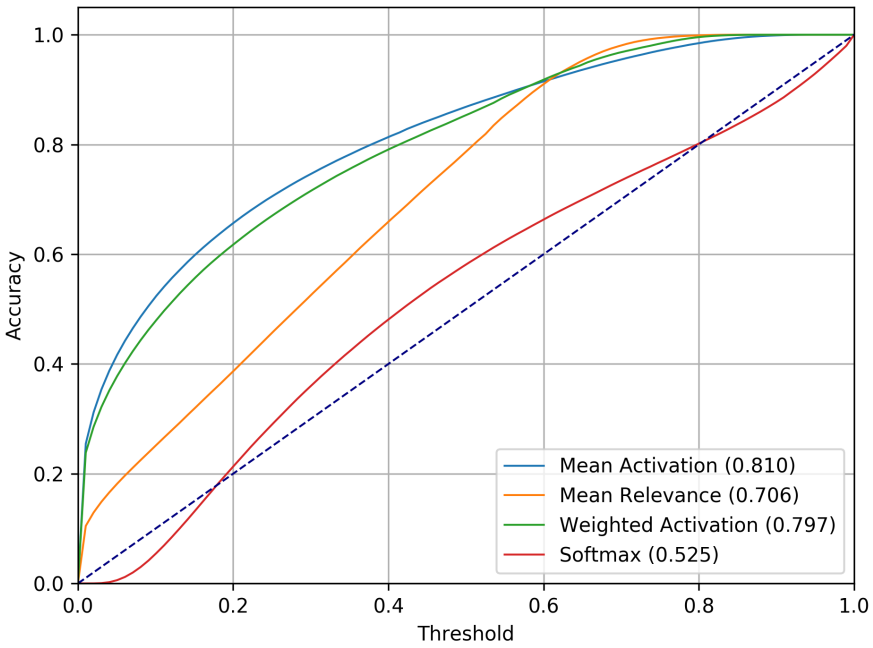


Figure 5.20: Inception v3: Accuracy curves for the ImageNet 2010 OpenSet

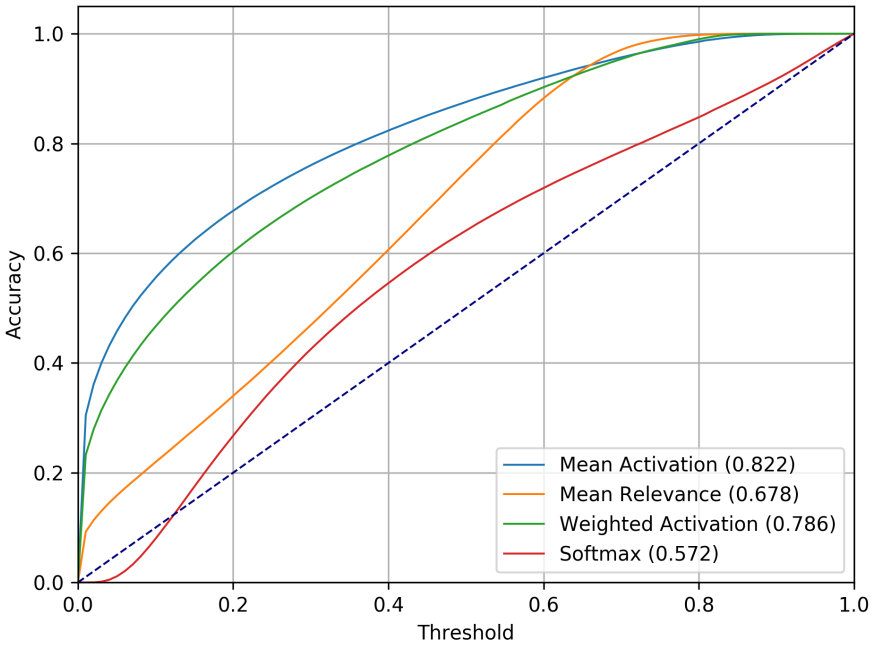


Figure 5.21: Xception: Accuracy curves for the ImageNet 2010 OpenSet

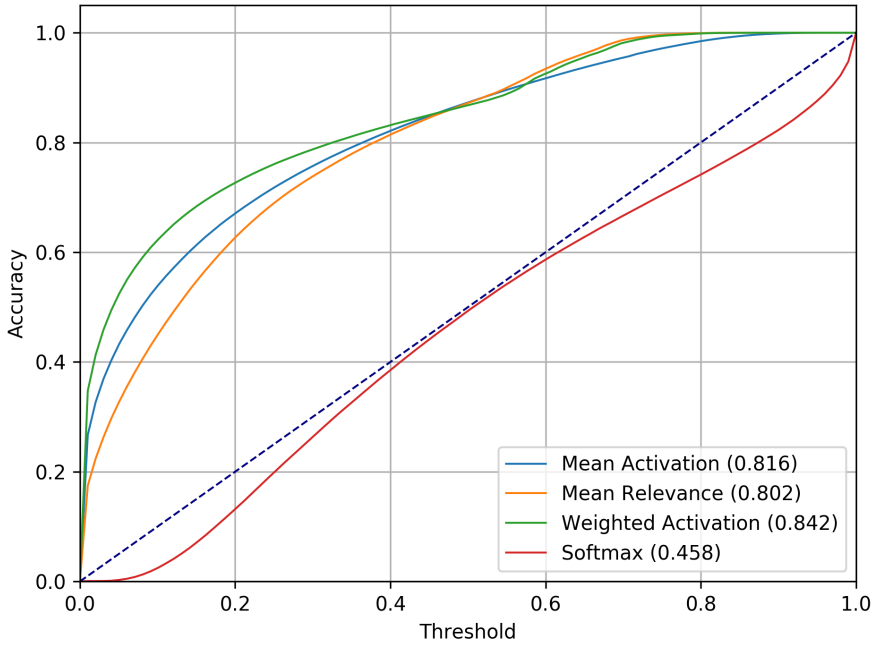


Figure 5.22: ConSemble: Accuracy curves for the ImageNet 2010 OpenSet

5.3.5.4 ConSemble

The accuracy of the convolutional ensemble is presented in Figure 5.22. The softmax-thresholding deteriorates below random guessing and is outperformed by all self-assessment models by a wide margin. The model based on the Weighted Activation exceeds the accuracy of all other models.

5.3.5.5 Evaluation

We conclude that the self-assessment models can detect images of unknown classes from the ImagenNet 2010 OpenSet with higher accuracy than the softmax-thresholding. However, the results are not as precise as for the fooling images. We suspect that this is because, unlike fooling images, images of this dataset contain background-cues and are therefore mapped closer to the region of the feature space where the images of the predicted class usually reside. The attribution based models, on the other hand, yield almost similar performance as for the fooling images.

5.3.6 Mix Dataset

In this Section, the results for the Mix Dataset will be presented. We will report the ROCs, as well the F_1-scores.

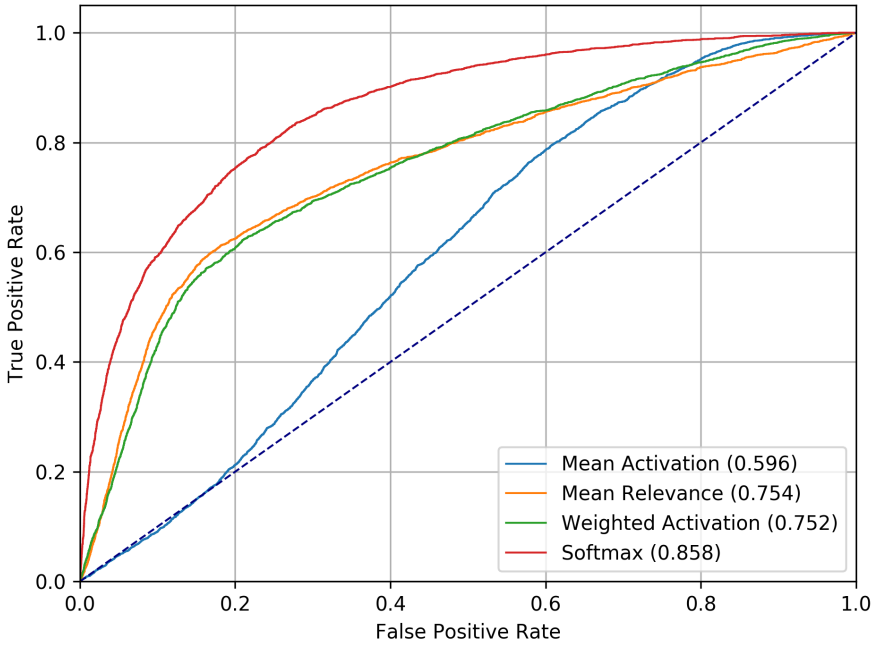


Figure 5.23: Vgg19: ROC for the Mix Dataset

5.3.6.1 Vgg19

The graphs in Figure 5.23 depict the ROC of the self-assessment models for the Vgg19. All self-assessment models surpass random guessing but achieve lower AUROCs than the softmax-thresholding. The self-assessment models that employ the relevance (i.e., Mean Relevance and Weighted Activation) exhibit similar performance, while the model based on the Mean Activation performs significantly worse.

Figure 5.24 depicts the F_1-score for varying thresholds of the self-assessment models for the Vgg19. The F_1-scores are in line with the ROCs. While the Mean Activation based self-assessment model achieves the lowest scores, the models incorporating the relevance perform similarly. None of the self-assessment models continually surpasses the baseline. However, the self-assessment seems to work better than the baseline for a threshold lower than 0.3. The performance of the Mean Relevance- and the Weighted Activation-based model degrade for thresholds > 0.6 and approach zero at around 0.9. This indicates that almost none of the correct predictions receives confidence scores above 0.9, leading to a small number of true-positive and a large number of false-negatives.

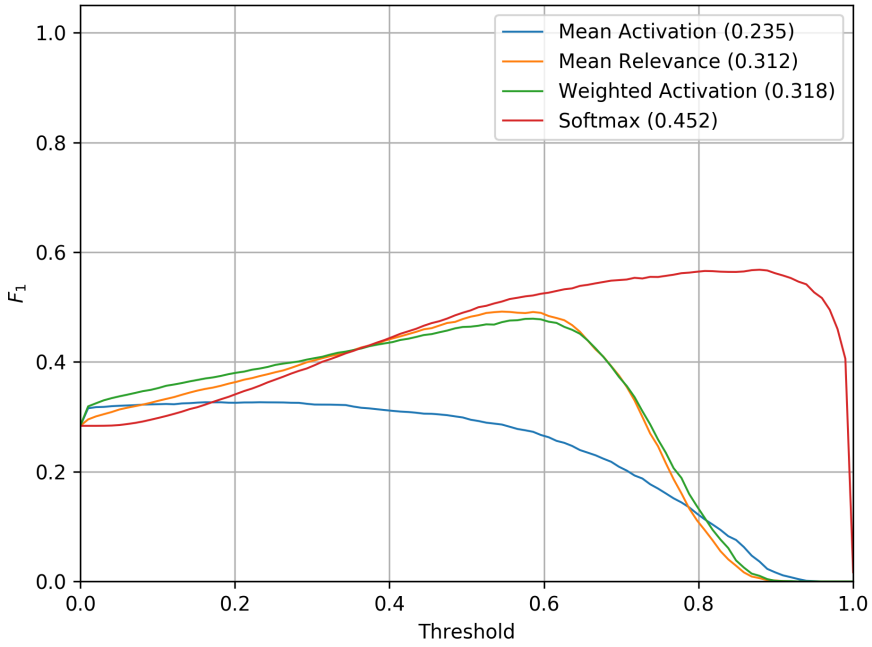


Figure 5.24: Vgg19: F_1 curves for the Mix Datase

5.3.6.2 Inception v3

The plots in Figure 5.25 depict the ROC of the self-assessment models for the Inception v3 CNN. All self-assessment models surpass random guessing. The model based on the relevance alone yields the lowest performance, while the model based on the Mean Activation surpasses the softmax-thresholding baseline, particularly for lower FPRs. The performance of the model based on the Weighted Activation lies between the other two models.

Figure 5.26 depicts the F_1-score for varying thresholds of the self-assessment models for the Inception v3. The F_1-scores differ slightly from the results of the ROCs. The graphs reveal that the self-assessment models perform better for lower thresholds. However, at thresholds > 0.6, all self-assessment models are surpassed by the baseline. Nevertheless, the F_1-scores confirm that the models employing the activation perform better than the model based on the relevance only.

5.3.6.3 Xception

The graphs in Figure 5.27 depicts the ROC of the self-assessment models for the CNN based on the Xception architecture. The ROCs are similar to the ones for the Inception v3. Only the Mean Activation-based model surpasses

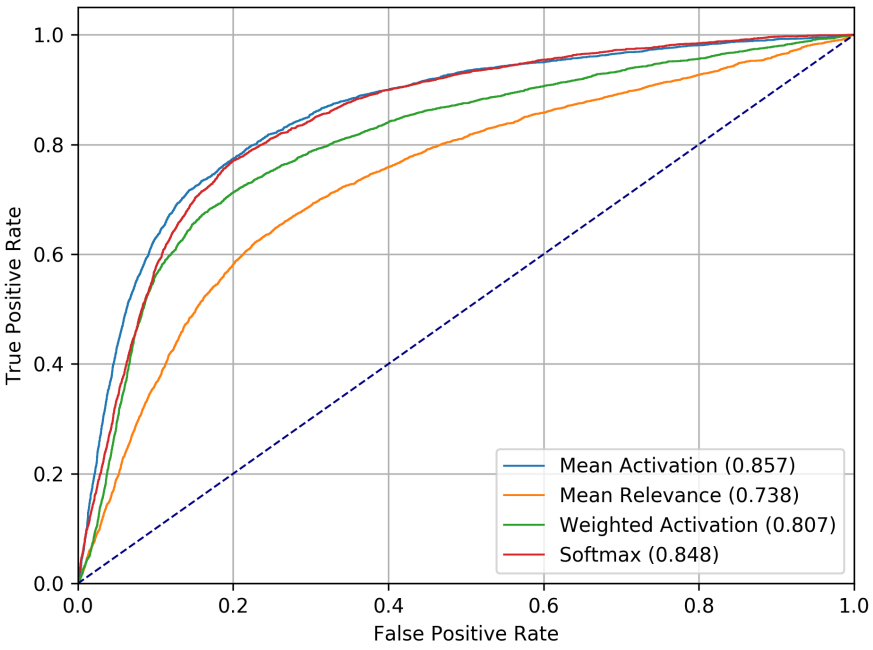


Figure 5.25: Inception v3: ROC for the Mix Dataset

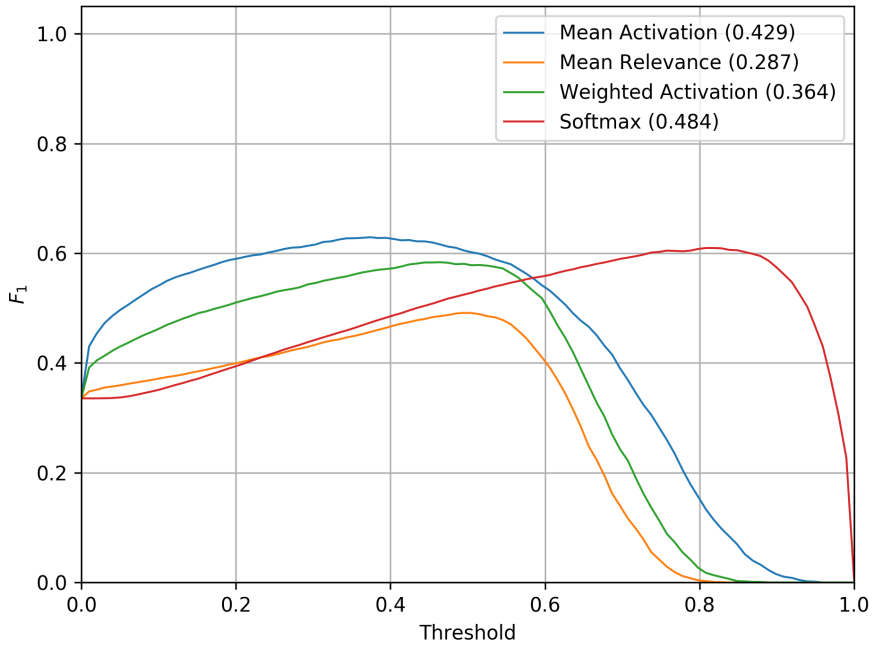


Figure 5.26: Inception v3: F_1 curves for the Mix Datase

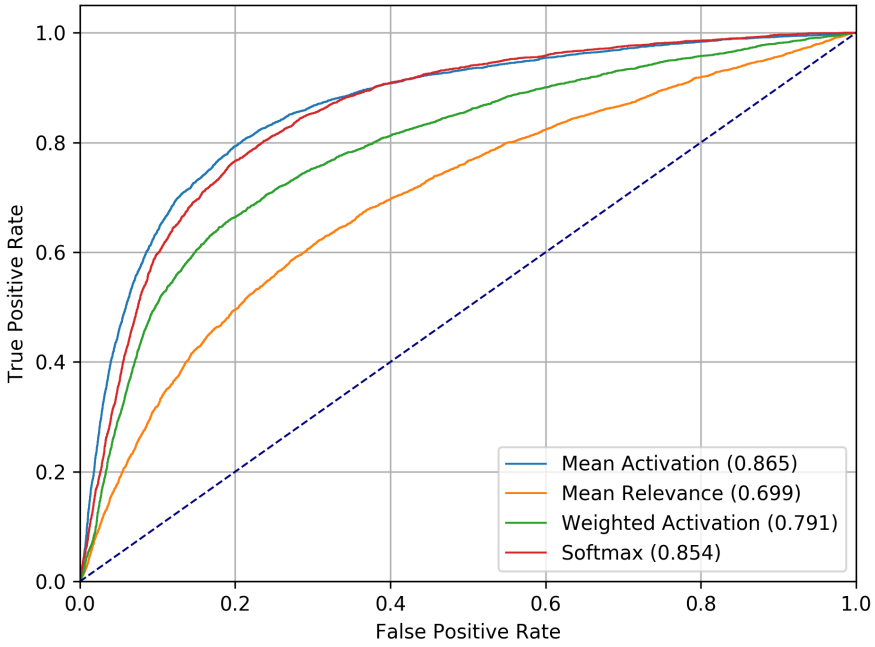


Figure 5.27: Xception: ROC for the Mix Dataset

the softmax-thresholding. The Mean Relevance-based model exhibits the lowest performance.

Figure 5.28 depicts the F_1-score. It confirms the results of the ROC, but again reveals that the self-assessment models yield optimal performance for lower thresholds and are surpassed by the baseline for thresholds > 0.5.

5.3.6.4 ConSemble

The graphs in Figure 5.29 depicts the ROC for the self-assessment models of the convolutional ensemble. None of them exceed the baseline, but all of them surpass random guessing. Compared to the ROCs of the other CNNs, all self-assessment models achieve significantly better performance, and the differences between the models are smaller.

Figure 5.30 depicts the F_1-score for the self-assessment models of the ConSemble. The results resemble the findings for the other CNNs. While the self-assessment models exhibit optimal performance for lower thresholds, the scores degrade drastically for thresholds > 0.6. At that point, the models are surpassed by the baseline. Similarly to other experiments, we observe that the performance of the model based on the Mean Activation degrades more slowly than the performance of the other models.

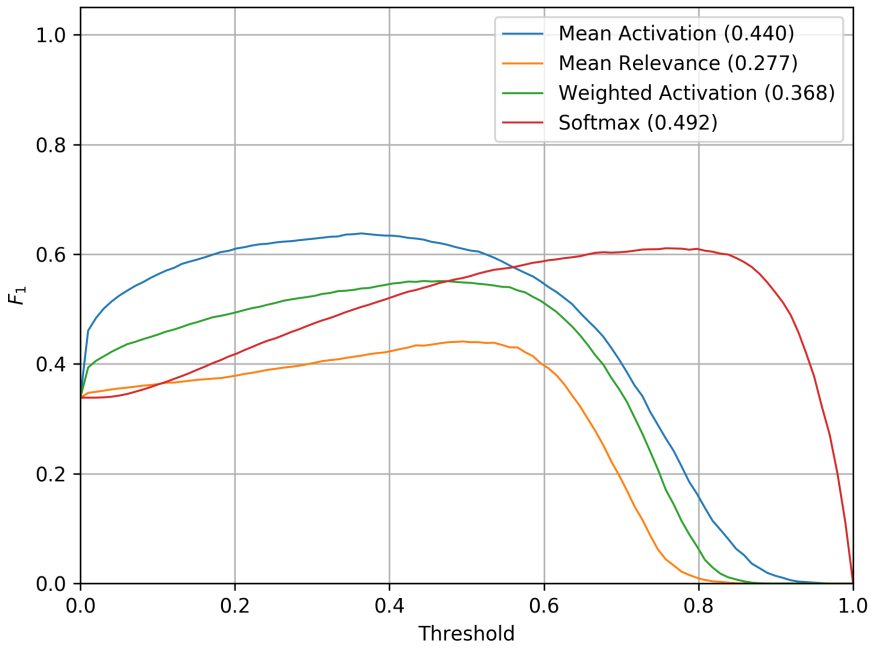


Figure 5.28: Xception: F_1 curves for the Mix Datase

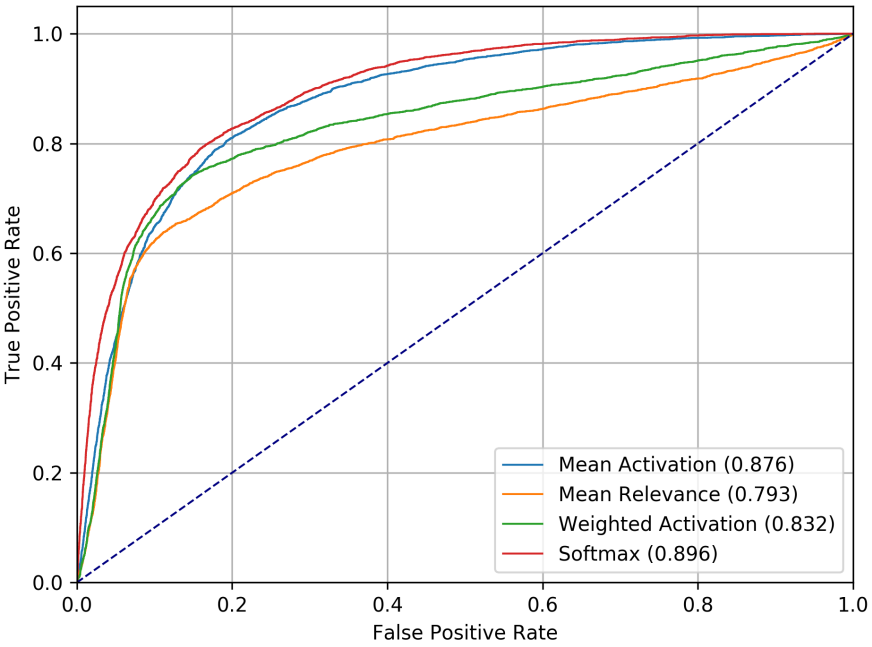


Figure 5.29: ConSemble: ROC for the Mix Dataset

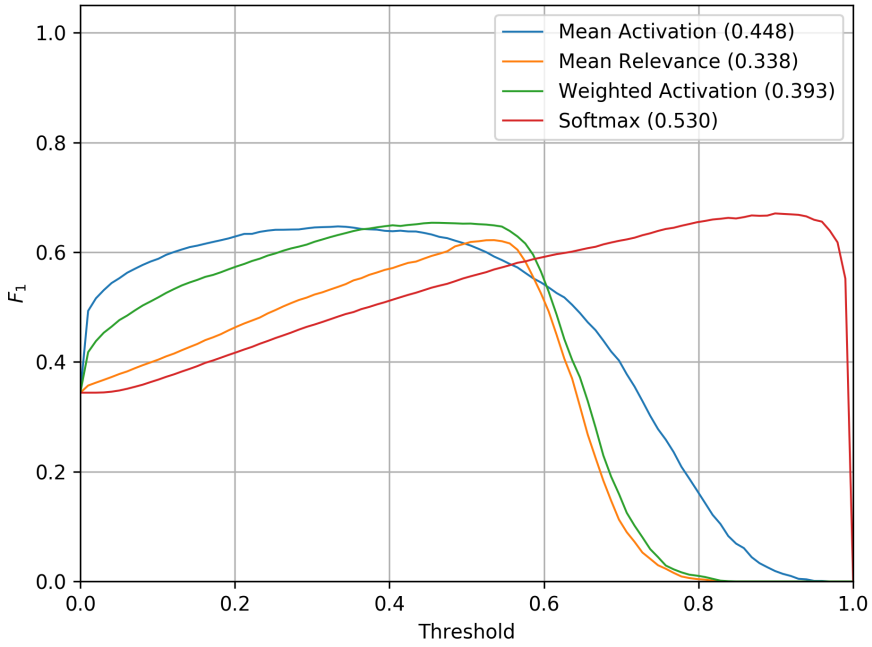


Figure 5.30: ConSemble: F_1 curves for the Mix Dataset

5.3.6.5 Evaluation

We conclude that all models outperform random guessing for the Mix Dataset. The self-assessment models based on the Mean Activation feature match the performance of the softmax-thresholding and sometimes surpass it. For the ConSemble, the Xception, and the Inception v3, the F_1-scores surpass the baseline for lower thresholds. We suspect that this is because the self-assessment models assign lower confidence values to images than the softmax in general. As observed during the evaluation of previous datasets, correct predictions rarely receive confidence > 0.8. We presume that this behavior is inherent to the design of the proximity-based model.

Consider, for instance, a prediction with a confidence of 0.5. This score indicates that 50 % of the correctly classified instances observed during training resided in a greater distance to the nearest cluster center. We argue that such a prediction should be accepted since rejecting predictions with confidence < 0.5 would entail that half of the correct predictions in the training set would be rejected as well.

5.4 Discussion

In this section, the outcomes of the conducted experiments will be discussed further.

5.4.1 Comparison to other Approaches

At first, the performance of the proximity-based statistical model will be compared to other approaches. However, we notice that it is difficult to compare the performance of our self-assessment model to other state-of-the-art methods. The primary reason is that for many publications, neither the dataset(s) nor the code is publicly available. Furthermore, as mentioned in Section 5.3.1, there seems to be no unified evaluation protocol.

Table 5.8: AUROC values for the self-assessment models based on the Mean Activation (MA), the Mean Relevance (MR) and the Weighted Activation (WA). The best score for the self-assessment models is highlighted. *Baseline* refers to softmax-thresholding.

Network	MA	MR	WA	Baseline
Mix Dataset				
Vgg19	0.596	**0.754**	0.752	0.858
Inception v3	**0.856**	0.738	0.807	0.848
Xception	**0.857**	0.699	0.791	0.845
ConSemble	**0.876**	0.793	0.832	0.896
ImageNet 2017 Validation				
Vgg19	0.591	0.735	**0.751**	0.855
Inception v3	0.733	0.730	**0.779**	0.846
Xception	0.732	0.701	**0.751**	0.847
ConSemble	0.724	0.791	**0.801**	0.865
ImageNet-A				
Vgg19	0.397	**0.459**	0.417	0.681
Inception v3	**0.489**	0.413	0.440	0.582
Xception	**0.527**	0.440	0.448	0.607
ConSemble	**0.491**	0.424	0.430	0.591

5.4.1.1 Comparison to OpenMax

The OpenMax-Layer proposed by Bendale et al. [7] has been described in Section 2.6.3.3. Bendale et al. explain how their test-dataset was created, but as far as we know, the images are not publicly available. Moreover, their study was concerned with the AlexNet architecture, which has a much lower base performance than the models we used (57.1 % on the ImageNet

2012 validation set) [37]. Furthermore, their experiments were based on the ImageNet 2012. We noticed that Bendale et al. reported the F_1-score of their approach for thresholds between 0 and 0.45, which is approximately the threshold-range where our self-assessment models perform best (see, e.g., Figure 5.30). For this threshold-range, they report a relative improvement over the peak of the softmax-thresholding of 4.3 %. If we consider the ConSemble self-assessment based on the Mean Activation feature and ignore all F_1-scores for thresholds > 0.45, we would achieve a relative performance gain over the softmax-thresholding-peak of 17.9 %. However, if we also include the thresholds above 0.45, the highest self-assessment score is 3.6 % lower than the peak of the softmax-thresholding. Therefore, we can not directly compare our self-assessment model to the OpenMax.

5.4.1.2 Comparison to Confidence Estimation Branch

The Confidence Estimation Branch approach proposed by Devries et al. (described in Section 2.6.3.1) reported AUROCs > 0.99, which exceeds the self-assessment capabilities of our models. However, their experiments were based on smaller datasets, for example, the TinyImageNet, which features 200 ImageNet classes with 500 images per class with a size of 64×64 pixels. The dataset we used contains five times more classes, 13 times more images, and between 12 and 21 times more pixels per image - depending on the respective CNN.

Additionally, the Confidence Estimation Branch employs a neural network for the self-assessment. Consequently, even though the results of our self-assessment models do not match the results of Devries et al., our models provide a higher degree of explainability.

5.4.2 Attribution based Self-Assessment

In this section, the findings for the experiments will be summarized. Table 5.8 provides an aggregated overview of the AUROC for different datasets.

In Section 3.1, we argued that attribution-maps could be harnessed for self-assessment. We expected them to exhibit clustering properties similar to the activation-maps, which are already used in state-of-the-art approaches. The results of the experiments support this hypothesis. As we can see, the relevance scores can indeed be harnessed for self-assessment, since the approach based on the Mean Relevance feature performed significantly better than random guessing in the majority of scenarios we tested. Because of the way the self-assessment models are designed, we can deduce that the attribution-maps for correct predictions are, in general, more similar to the ones observed during training than the attribution-maps for incorrect predictions.

For the ImageNet-A dataset, the performance of the attribution-based model

did, in fact, not surpass random guessing. However, neither did activation-based models. This indicates that the properties of the natural adversarial images are responsible for this behavior and not the approach itself. From the weakness of the self-assessment models, we can infer that for natural adversarial images, the assumption that the convolutional layers will map instances of the same class to similar regions of the feature space is violated. We find that in the case of the Xception, the Inception v3, and the ConSemble, the self-assessment based on the Mean Activation produces more reliable confidence values. This suggests that the clustering properties of the attribution-maps are not as distinctive as for the activation-maps (see Section 5.3.2.5). However, the combination of attribution- and neural activation-patterns (i.e., the Weighted Activation) improved the self-assessment results for the ImageNet 2017 validation set for all tested recognition systems. We conclude that attribution-maps can help to identify recognition failures for "simple" images.

5.4.3 Comparison of Recognition Systems

In this section, the differences between the results of the CNNs will be addressed.

5.4.3.1 Xception and Inception v3

The results show that the Inception v3 and the Xception architecture achieve comparable performance. We presume that this is because the Xception is a derivative of the Inception v3 architecture. While the Xception (the name is an abbreviation for *Extreme Inception*) achieves slightly higher accuracy, both models are based on the same fundamental ideas. Consequently, their similar performance is unsurprising.

5.4.3.2 Behavior of the Vgg19

The behavior of the Vgg19, on the other hand, diverges significantly from the other two base CNNs. Most notable is the weak performance of self-assessment models based on the Mean Activation. We suspect that there are multiple underlying causes.

While Xception and Inception extract 2048 activation-maps, the Vgg19 only extracts 512. Therefore, the overall weaker performance of the Vgg19 could be caused by the smaller number of extracted features. The Vgg19 has to map the same number of classes into separate regions of a vector space of lower dimensionality than the other models, which is more complicated. Furthermore, unlike the Vgg19, the Xception and Inception v3 flatten the extracted activation-maps using global average pooling. Therefore, the Mean Activation of the Xception and the Inception v3 is equal to their vector embedding. For the Vgg19, however, it is not. We suspect that the Mean Activation feature might not be as representative for the Vgg19 as it is for the other models, because the Vgg19

is not required to learn good global average pooled features during training. Comparing the scatterplots of the 2D vector embeddings in Figure 3.3, we notice that the Vgg19's feature embedding does not produce equally distinctive clusters, which supports the considerations above. Nevertheless, the behavior of the Vgg19 CNN requires further investigation.

5.4.3.3 Convolutional Ensemble

The ConSemble, in general, exceeds the performance of the base models in terms of self-assessment and softmax-thresholding. As explained in Section 2.3.3, access to a larger amount of extracted features (in our case 4608 pooled features) allows the ConSemble to make more accurate predictions. While the existing literature on convolutional ensembles emphasized the accuracy gains, we find that the ConSemble also surpasses its base models in terms of its self-assessment-capabilities. The ConSemble did not only make more accurate predictions but, for example, for fooling images, it made wrong predictions with lower confidence compared to its base models[4].

However, this does not hold for the ImageNet-A and the ImageNet 2010 OpenSet. For natural adversarial images, the ConSemble achieved a lower Top-1-Accuracy, as well as a lower self-assessment score (AUROC) for both the softmax and the self-assessment models. We suspect the following: The ConSemble has access to the same features as the base models. During the training, the ConSemble learns which of these features allow it to make the best predictions for the ImageNet training set, and favors these features for predictions. In order to be able to make better predictions than the base models, the ConSemble has to select and combine exactly those features from the model that work best on the training set. We suspect that the features that allow the model to make the most accurate predictions for the ImageNet tend to be the features that are, according to Nguyen et al., responsible for the weakness of CNNs for natural adversarial images, namely features based on color or texture. Thus, the improved performance of the ConSemble on the "simple" images comes at the price of weaker performance for "hard" images. For the ImageNet 2010 OpenSet, the ConSembles softmax-thresholding performance deteriorated below random guessing. We suspect that this behavior is related to the causes that led to decreased performance on the ImageNet-A - presumably overfitting to the ImageNet.

[4]See, for example, Figure 5.18. The superior accuracy of the ConSemble for lower thresholds indicates that fooling images, in general, receive lower confidences.

Chapter 6

Conclusion

This chapter provides the conclusion of the present thesis. We will first summarize the content and afterward discuss the findings critically. From this discussion, we will derive perspectives for future work that aims to overcome the limitations of this work, and present research questions that emerge from the findings of this thesis.

6.1 Summary

In the first chapter of this thesis, we described the underlying pathologies contemporary CNNs suffer from: They are highly complex non-linear models that can not be formally verified by contemporary means due to their vast number of parameters. Therefore, these systems can not be proven to meet safety requirements. Secondly, the decisions made by artificial neural networks are difficult to explain, and these recognition systems are, therefore, often described as black-boxes that may fail unexpectedly [59]. Additionally, CNN-based visual recognition systems are prone to certain errors. We pointed out that these pathologies constitute a problem in safety-critical environments. Based on these considerations, we motivated approaches that aim to mitigate the potential safety-threat posed by artificial neural networks deployed in real-world applications, as well as methods to facilitate an understanding of the system's decisions. We introduced two conventional strategies: while the first one aims to improve the performance of the systems such that errors are sufficiently improbable, the second approach aims to detect recognition errors at runtime and thus provides the system with the capability to fail gracefully in order to retain in a state that is considered safe. Additionally, gradient-based attribution methods have been developed to explain the decisions made by neural networks in hindsight by estimating which parts of the input where most relevant for the decision. In that process, attribution scores are assigned to neurons, that measure the importance of that particular neuron for the prediction of the recognition system. We pointed out that many of the state-of-the-art self-assessment approaches are based on as-

sessing patterns in the activation of neurons, but - to the best of our knowledge - no work seems to consider attribution methods for the detection of recognition failures at runtime. Therefore, we proposed the usage of attribution values for self-assessment.

In the second chapter, we provided an overview of work related to the proposed approach. At first, we explained fundamental building blocks of CNNs, emphasizing the distinction between the convolutional layers that produce activation-maps, the flattening that creates the vector embedding from the activation-maps, and the fully connected layers that make predictions based on the embedding vector. Afterward, the concept of transfer learning has been motivated and explained. We provided an overview of ensemble methods, which aim to improve the performance of neural networks. We proceed by introducing some example attribution methods with the focus on CAMs. In the following, we present the results of a literature survey that identified different types of images that may cause recognition systems to fail. The chapter closed with a survey of literature regarding existing self-assessment methods, which we loosely group into categories based on the type of information they use for the assessment.

In the third chapter, we explained the explanation-based self-assessment approach proposed in the first chapter in detail. We provided arguments as well as some empirical evidence to substantiate the idea. Subsequently, we presented a framework that would allow us to create explanation-based self-assessment models. Since an exhaustive evaluation of existing attribution-based explanation methods would have exceeded the scope of this thesis, we focused on a prototypical explanation approach, namely GradCAM. For GradCAM, the explanation is calculated from the attribution- and activation-maps. We argued that explanations that are similar to the explanations observed for correctly classified examples during training could be regarded as evidence for correct classifications, while dissimilar explanations indicate the opposite. Mathematically, the concept of similarity is usually expressed as a distance measured by a particular metric. Therefore, we anticipated that the "similar" explanations (i.e., the activation- and attribution-maps as used by GradCAM) form clusters in a vector space. For simplicity, we assume that the shape of these clusters is spherical. Consequently, we used the framework to developed a proximity-based self-assessment model for the evaluation of the proposed self-assessment approach. This model learns from the correctly classified training examples what a "usual" explanation looks like by estimating the centers of the hyper-spherical clusters of the activation- and attribution-maps. The fundamental assumption utilized for the self-assessment is that "explanations" that are further apart from these cluster centers are more"unusual" and therefore indicate a higher likelihood of recognition failure. To inquire if the attribution-maps carry the potential to improve the self-assessment, we created multiple models of the same type but based on different features — one based on activation, one based

on attribution, and one based on a combination of both.

In the *Implementation* chapter, we introduced the CNNs we used to evaluate the self-assessment capabilities of our model. We explain why we chose these CNNs and how they were trained. We proceeded by describing the implementation of the proposed framework and the created models in particular. Ultimately, we presented a demo application that allows users to try the self-assessment models.

The fifth chapter explained our evaluation methodology, reported the results, and discussed them. Initially, we described the datasets that were used during the evaluation and also presented a new composed *Mix Dataset* that contains the majority of image types we identified as responsible for recognition failures in the literature survey. We presented results for the CNNs on these datasets, following the evaluation protocol proposed by Wehmeier [67]. Afterward, we reported the results for the self-assessment models on all datasets and compared them to the softmax-thresholding. We found that the models based on the activation - as used in state-of-the-art self-assessment methods - achieved the best performance. Especially fooling images could be detected with high accuracy. As we anticipated, the self-assessment models using the attribution values significantly outperformed random guessing for almost every dataset. Therefore, we concluded that it is possible to predict recognition errors by evaluating patterns in the relevance of neurons. Natural adversarial images constitute an exception, which led us to conclude that for these images, the relevance of neurons, as well as the activations of neurons, are similar to the relevances and activations observed for correct predictions. Furthermore, we found that the combination of attribution and activation was able to outperform the model based on solely the activation for the ImageNet 2017 validation set. This substantiates the proposition that self-assessment methods harnessing attribution-based explanations can improve the self-assessment beyond the approaches based on the neural activation alone.

6.2 Limitations

This chapter will provide a critical reception of the findings of this thesis and point out the flaws and limitations of the presented approach and the evaluation methodology.

6.2.1 Reproducibility Considerations

The reproducibility of results is a crucial requirement imposed on scientific work. In a paper published recently, Bouthillier et al. stated that many findings in the field of deep learning are - according to their definition - not reproducible [8]. They assert that the term *reproducability* can refer to three different notions of

reproducibility, each of which will be discussed in the following in regard to this thesis.

6.2.1.1 Method Reproducability

Method Reproducability refers to reproducability in terms of deterministic output. Running the same code with the same input twice should produce identical results. In our experiments, we used a fixed seed for random number generators. However, due to the stochastic nature of deep learning and k-Means clustering, the results vary within certain bounds if we use other seeds[1]. In order to show that the results meet this definition of reproducibility, Bouthillier et al. recommend to calculate mean scores over multiple trainings, and to report the interval bounds. In our case, the usage of publicly available, pre-trained networks mitigates this problem to some extent, as the predictions of these networks are deterministic.

6.2.1.2 Results Reproducibility

Results Reproducibility refers tho the idea that a reimplementation of the same method should yield statistically similar outcomes. A reimplementation of the proposed framework is out of scope of this thesis. Thus, we are as of now unable to make reliable statements concerning the results reproducibility of this thesis. By publishing the proposed approach, we could facilitate the result reproducability and encourage others to replicate our experiments.

6.2.1.3 Inferential Reproducibility

Inferential Reproducibility refers to the idea that results should be statistically stable for other experimental setups, for example other datasets, or, in our case, other CNNs. Since we provided evaluation results for multiple datasets, different clustering algorithms and different self-assessment models (see Appendix A and B) we ensured a certain level of inferential reproducibility. However, the results between the different image types diverged significantly, which was broadly discussed in Chapter 5.

6.2.2 Self-Assessment Model

In this section, the limitations of the self-assessment model itself will be discussed.

[1]Even though we used a fixed seed for Tensorflow, we noticed deviations in the performance of the trained ConSembles.

6.2.2.1 Model Performance

The key finding of this thesis is that, in our experiments, the self-assessment based on attribution outperformed random guessing and could in some cases improve the self-assessment beyond the performance achieved using the activation alone. In line with the description of the scope of this work in Section 1.4, we favored a self-assessment model that is interpretable and allows us to draw conclusions regarding the proposed approach, over a model with a better performance that can not be explained. Since the created models did not outperform the softmax-thresholding by a wide margin, we should remain skeptical about the generalizability of the findings to self-assessment models with better performance.

6.2.2.2 Model Design

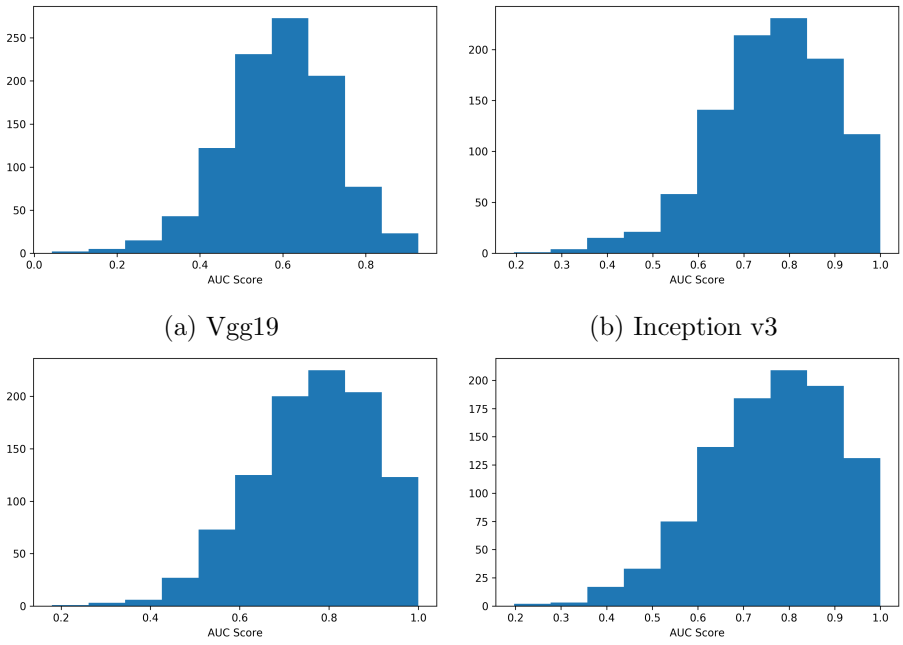
Furthermore, we could refine the present self-assessment model itself. At the moment, we employ one sub-model per class. However, all class-models use the same hyperparameters, i.e., the same clustering algorithm and the same distribution-function for the distances. Figure 6.1 depicts histogram of AUROC scores for individual classes. As we can see, there are classes for which the self-assessment does not work as well as for others. We suspect that the performance of the model could be enhanced by conducting a HPO for each class individually.

Moreover, the proposed proximity-based model considers the distance of instances to their nearest cluster, but is agnostic on which specific cluster center has been closest. This could lead to the following problem: let us assume we have two cluster centers for one class, A and B. Training instances distribute densely around A but are much more scattered around B. We now encountered an instance that is closest to A but resides in an unusually large distance from it. The same distance, however, is usual for instances around B. We should now be skeptical about the CNNs prediction since the instance is unusually far away from A. However, the model, as described in this thesis, does not differentiate between the two cluster centers and could thus be prone to accept the instance. Therefore we suspect that the results could be improved by modeling the density of instances around each cluster center individually.

6.2.3 Security Considerations

This work emphasized a safety perspective. The security of the proposed approach in environments that may be targeted by an omniscient[2] adversary remains questionable. We demonstrated that the self-assessment models are more

[2]By *omniscient* we mean that the adversary has access to the model.



(a) Vgg19 (b) Inception v3

(c) Xception (d) ConSemble

Figure 6.1: Histogram of AUROC scores of the self-assessment model based on the Mean Activation feature for individual classes in the ImageNet 2017 validation set. These histograms show that the lower performance of the Vgg19 precipitated by weaker performance in general, and not by a weaker performance for particular classes only.

robust to Fooling Images than the softmax. These Fooling Images were generated by maximizing the activation of a specific output neuron. However, there are adversarial attacks that target deep feature embeddings [49]. Such attacks undermine the underlying assumption of the self-assessment models - that images whose activation/attribution are similar to the ones observed during training are more likely to be classified correctly - by modifying the images in a way that places their feature embeddings in the same region of the feature space as the correctly classified training images. By leveraging this method, the authors were able to defeat OpenMax (see Section 2.6.3.3), which is based on a similar assumption. This suggests that the presented self-assessment model could also be vulnerable to such an attack. Furthermore, a recent publication demonstrated that it is possible to fool GradCAM using adversarial patches [62]. These patches are able to change the prediction of the CNN arbitrarily, but pixels in the patches are not highlighted by GradCAM, even though they are responsible for the prediction. The authors conclude that GradCAM is not a reliable method to highlight the relevant input. This suggests that, while the self-assessment model may provide some security, it is probably not able to resist attacks directly targeting the method itself.

To put these caveats into perspective, consider the following. If the proposition of [21] proves true and the pathologies of CNNs are inherent to their architecture, it seems reasonable to assume that an adversary with access to the model might always be able to find means to defeat mechanisms implemented to defend against adversarial attacks.

6.3 Future Work

In this section perspectives for future work will be presented.

6.3.1 Future Work on Convolutional Ensembles

We found that the Convolutional Ensemble was able to exceed the performance of its base models in our experiments. However, as stated in Chapter 4, we did not perform an exhaustive HPO for the ConSemble and suspect that the performance could be improved further by extending the set of considered hyperparameters. We could, for example, include other weight initialization methods or different optimizers, like Stochastic Gradient Descent[3].

Apart from the point mentioned above, it seems promising to evaluate the performance of the ConSemble when using a larger number of convolutional base models with a variety of different architectures. To the best of our knowledge,

[3]Stochastic Gradient Descent is generally assumed to achieve better results than the Adam optimizer if the parameters are carefully chosen. However, finding an optimal configuration for these parameters usually requires additional compute.

existing literature only reports experiments with up to three base models. We expect an additional gain in performance when using a larger number of convolutional base models, as this would increase the variability in the models and thereby increase the chance that at least one of the models can extract distinctive features for a given image. There is, to the best of our knowledge, no publicized work concerned with the comparison of the convolutional ensemble approach to sophisticated dynamic ensembles or MoE approaches, like the *Adaptive Fusion* method described in Section 2.3.2.

6.3.2 Future Work on Self-Assessment

The proposed approach is promising, as we demonstrated that it is possible to increase the performance of a self-assessment model by using information created by explanation methods.

Considering the limitations presented in Section 6.2, additional effort should be directed towards ensuring the reproducibility of this finding. Future work should assess the stability of the created approach, for example, by "training" multiple self-assessment models with the same configuration and examining variations in the results. Furthermore, the self-assessment method should be applied to additional CNNs and datasets. Using smaller datasets, for example, TinyImageNet, MNIST, or Caltech 265, will facilitate the comparison to other state-of-the-art techniques. It should also be tested if the self-assessment model provides equally strong protection against fooling images crafted explicitly for the assessed CNN. Furthermore, the performance for corrupted and adversarial images should be investigated.

The flexibility of the proposed framework facilitates the development of other self-assessment models based on the activation- and attribution-maps, as described in Section 3.2. This should be leveraged to examine other explanation methods, like Layer-Wise Relevance Propagation. Besides, we suspect that self-assessment models based on neural networks, for example, CNNs, Siamese networks, or Auto Encoder, exceed the performance of the statistical model. Concerning the limitations that addressed the performance of the self-assessment model, we recommend testing the transferability of the proposed approach to methods that trade explainability for performance. Furthermore, it might be promising to extract additional features from the activation- and attribution-maps that are more representative for the specific instance. For example, we suspect that the results could be improved by incorporating the distribution of the values in the activation-maps and not only mean values. For instance, during some experiments at the inception of this thesis, we noticed that a CNN we trained to discriminate between synthetic images of rectangles and circles of different colors would achieve high accuracy in general, but had difficulties classifying objects located close to the border of images. This is merely anecdotal

evidence; however, it illustrates the idea that the location of the detected object in the image could be useful for self-assessment as well.

As stated in Section 2.4, gradient-based attribution methods like GradCAM are not limited to convolutional neural networks, but can be applied to arbitrary differentiable classifier [55, 4]. In natural language processing, for instance, the approach of training a neural network to map the input to an embedding vector for further processing is also common [15]. Attribution-based methods can be used to estimate the relevance of a specific value in the embedding, or even the relevance for specific words in the input [4]. Thus, the presented framework could be applied to such tasks as well and is not necessarily limited to visual recognition.

Within the scope of our literature survey, we have been unable to find publications concerned with multimodal self-assessment[4]. Combining information from the Input, Output, Context, and Intermediate features (like neural activation or attribution of intermediate layers) could potentially increase the performance of self-assessment approaches.

[4]The work on context-based self-assessment did include the input of the classifier (i.e., an image) into the assessment process [26]. However, the combination of multiple modalities was not the primary focus of this work.

Bibliography

[1] *All Together Now! The Benefits of Adaptively Fusing Pre-trained Deep Representations*, February 2019.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[3] Thangarajah Akilan, Qingming Jonathan Wu, and Hui Zhang. Effect of fusing features from multiple dcnn architectures in image classification. *IET Image Processing*, 12(7):1102–1110, 2018.

[4] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[5] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[6] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1893–1902, 2015.

[7] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[8] Xavier Bouthillier, César Laurent, and Pascal Vincent. Unreproducible research is reproducible. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 725–734, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[9] Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

[10] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[11] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[12] François Chollet et al. Keras, 2015.

[13] Shreyansh Daftry, Sam Zeng, J Andrew Bagnell, and Martial Hebert. Introspective perception: Learning to predict failures in vision systems. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1743–1750. IEEE, 2016.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[16] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[17] Akshay Raj Dhamija, Manuel Günther, and Terrance Boult. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems*, pages 9157–9168, 2018.

[18] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

[19] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[20] Dimitrios Frosyniotis, Andreas Stafylopatis, and Aristidis Likas. A divide-and-conquer method for multi-net classifiers. *Pattern Analysis & Applications*, 6(1):32–40, 2003.

[21] Ben Goertzel. Are there deep reasons underlying the pathologies of today's deep learning algorithms? In *International Conference on Artificial General Intelligence*, pages 70–79. Springer, 2015.

[22] Tim Gonschorek, Marco Filax, and Frank Ortmeier. A very first glance on the safety analysis of self-learning algorithms for autonomous cars. In *37th International Conference on Computer Safety, Reliability, & Security SAFECOMP2018SAFECOMP2018*, 2018.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[24] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[25] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.

[26] Corina Gurău, Dushyant Rao, Chi Hay Tong, and Ingmar Posner. Learn from experience: probabilistic prediction of perception performance to avoid failure. *The International Journal of Robotics Research*, 37(9):981–995, 2018.

[27] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):993–1001, 1990.

[28] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[29] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *CoRR*, abs/1903.12261, 2019.

[30] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[31] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CoRR*, abs/1907.07174, 2019.

[32] Chen Huang, Chen Change Loy, and Xiaoou Tang. Local similarity-aware deep feature embedding. In *Advances in neural information processing systems*, pages 1262–1270, 2016.

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[34] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[36] Panagiotis Kouvaros and Alessio Lomuscio. Formal verification of cnn-based perception systems. *CoRR*, abs/1811.11373, 2018.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[38] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[39] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE, 2010.

[40] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[41] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *European Conference on Computer Vision*, pages 488–501. Springer, 2012.

[42] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.

[43] Grégoire Montavon, Sebastian Bach, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *CoRR*, abs/1512.02479, 2015.

[44] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[45] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[46] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[47] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[48] Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In *International Conference on Computer Aided Verification*, pages 243–257. Springer, 2010.

[49] Andras Rozsa, Manuel Günther, and Terrance E Boult. Adversarial robustness: Softmax versus openmax. *arXiv preprint arXiv:1708.01697*, 2017.

[50] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[51] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2012.

[52] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2317–2324, 2014.

[53] Walter J Scheirer, Anderson Rocha, Ross J Micheals, and Terrance E Boult. Meta-recognition: The theory and practice of recognition score analysis. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1689–1695, 2011.

[54] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[55] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[56] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org, 2017.

[57] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[58] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[59] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. How can we fool lime and shap? adversarial attacks on post hoc explanation methods. *arXiv preprint arXiv:1911.02508*, 2019.

[60] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[61] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[62] Akshayvarun Subramanya, Vipin Pillai, and Hamed Pirsiavash. Fooling network interpretation in image classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2020–2029, 2019.

[63] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[64] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[65] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[66] Bill Vlasic and Neal E Boudette. Self-driving tesla was involved in fatal crash, us says. `https://www.nytimes.com/2016/07/01/business/self-driving-tesla-fatal-crash-investigation.html`, 2016. [Online; accessed 9-December-2019].

[67] Leon Wehmeier. Consemble - variants of n-version programming for deep-learning. Technical report, 2019.

[68] Danny Yadron and Dan Tynan. Tesla driver dies in first fatal crash while using autopilot mode. `https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk`, 2016. [Online; accessed 9-December-2019].

[69] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[70] Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. Predicting failures of vision systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, 2014.

[71] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

Appendix A

Alternative Clustering Algorithms

This chapter provides supplementary results for the proximity-based self-assessment model proposed in Section 3.

A.1 K-Means

The self-assessment model presented in this thesis employs the k-Means clustering algorithm. The parameter k of this algorithm specifies the number of assumed clusters. We evaluated the performance of our model for different values for k. Some example results are depicted in the Figures A.1 and A.2.

We noticed that, while different values for k impact the results on individual datasets, the effects seem to even out for the composed Mix Dataset. As we can see, while an increased amount of cluster centers leads to improved accuracy on the ImageNet 2010 OpenSet, it slightly impairs the AUROC for the Mix dataset. We decided to report the results for 5 cluster centers, as this would provide the best tradeoff between different datasets.

A.2 Nearest Class Mean

This section contains results for the self-assessment model based on the Nearest Class Mean clustering algorithm. The aggregated AUROCs are listed in Table A.1.

The results confirm the findings of this thesis. The self-assessment models do generally not outperform the softmax-thresholding. The Mean Relevance-based model for the Vgg19 achieves a better performance than models based on other features. Models using attribution-maps do not perform well on natural adversarial images but can improve the results for the ImageNet validation set.

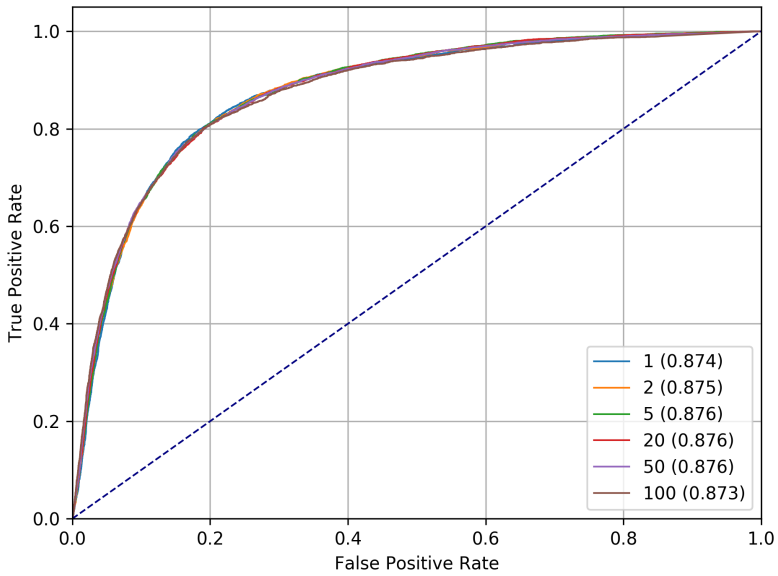


Figure A.1: ROC of the ConSemble using k-Means clustering with different numbers of cluster centers on the Mix dataset.

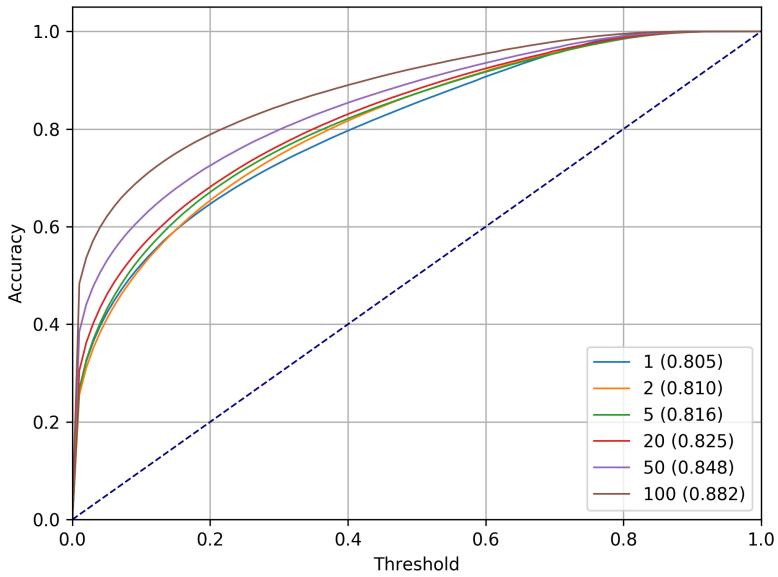


Figure A.2: Accuracy of the ConSemble using k-Means clustering with different numbers of cluster centers for the ImageNet 2010 OpenSet dataset.

Table A.1: AUROC for the self-assessment models based on the Mean Activation (MA), the Mean Relevance (MR) and the Weighted Activation (WA) using NCM. The best scores for the self-assessment models are highlighted.

Network	MA	MR	WA	Baseline
Mix Dataset				
Vgg19	0.607	0.698	**0.739**	0.858
Inception v3	**0.850**	0.661	0.777	0.848
Xception	**0.861**	0.588	0.772	0.845
ConSemble	**0.873**	0.739	0.815	0.896
ImageNet 2017 Validation				
Vgg19	0.584	0.684	**0.734**	0.855
Inception v3	0.731	0.671	**0.747**	0.846
Xception	**0.537**	0.410	0.446	0.847
ConSemble	0.723	0.747	**0.780**	0.865
ImageNet-A				
Vgg19	**0.496**	0.445	0.427	0.681
Inception v3	**0.498**	0.421	0.445	0.582
Xception	**0.537**	0.410	0.446	0.607
ConSemble	**0.501**	0.429	0.435	0.591

A.3 Affinity Propagation

In addition to the NCM, we also tested the affinity propagation clustering algorithm. This clustering approach selects a number of prototypical instances from the data. The algorithm determines the exact number of prototypes automatically.

The aggregated results are depicted in Table A.2. In general, they confirm the findings presented in the evaluation chapter: for the Mix dataset, the Mean Activation feature works best, and the AUROCs match the softmax-thresholding, except for the Vgg19 CNN, for which relevance based models perform significantly better. On the validation set, models combining activation and attribution achieve the best performance, which substantiates the hypothesis that attribution methods can improve the self-assessment beyond methods based on neural activations only. The performance on the ImageNet-A does not surpass random guessing for all models, which is in line with the results for all proximity-based models.

Table A.2: AUROC for the self-assessment models based on the Mean Activation (MA), the Mean Relevance (MR) and the Weighted Activation (WA) using Affinity Propagation. The best scores for the self-assessment models are highlighted.

Network	**MA**	**MR**	**WA**	**Baseline**
Mix Dataset				
Vgg19	0.587	0.739	**0.740**	0.858
Inception v3	**0.829**	0.769	0.787	0.848
Xception	**0.845**	0.744	0.782	0.845
ConSemble	**0.850**	0.798	0.807	0.896
ImageNet 2017 Validation				
Vgg19	0.591	0.709	**0.716**	0.855
Inception v3	0.713	0.734	**0.741**	0.846
Xception	0.718	0.718	**0.731**	0.847
ConSemble	0.697	0.740	**0.745**	0.865
ImageNet-A				
Vgg19	0.401	**0.411**	0.401	0.681
Inception v3	**0.426**	0.390	0.389	0.582
Xception	**0.481**	0.435	0.417	0.607
ConSemble	**0.424**	0.379	0.411	0.591

Appendix B

Random Forrest

In order to demonstrate the flexibility of the proposed framework, we used it to develop another self-assessment model. It is based on the same features that we utilized for the proximity-based model but employs a machine learning approach - namely Random Forrest. This classifier is an ensemble of decision trees, which is created by *bagging*, as described in Section 2.3.1.3.

B.1 Training

We trained one Random Forest classifier per class on both correctly and incorrectly classified instances of the ImageNet 2017 training set. In order to address the class-inbalance that arises because the classifiers make correct predictions for the majority of images, we assigned a higher weight to misclassified instances. To prevent overfitting, we limited the depth of the trees to three. Each random forest incorporates 1000 decision trees.

B.2 Results

The evaluation follows the process described in Section 5.3.1. For the sake of space, the results for the ImageNet 2010 OpenSet subset and the Fooling Images dataset will be omitted. The aggregated results are listed in Table B.1.

The AUROCs diverge slightly from the scores obtained from the proximity-based models. Again, the models do not surpass the softmax-thresholding-baseline but exceed random guessing. However, the self-assessment based on attribution alone yields the largest AUROC for eight out of twelve cases. Contrary to the proximity-based models, random forest slightly surpasses random guessing for the ImageNet-A.

Even though the outcomes differ from the results obtained with the proximity-based models, they support the conclusion of this thesis.

Table B.1: AUROC for the self-assessment models based on the Mean Activation (MA), the Mean Relevance (MR) and the Weighted Activation (WA) using a Random Forrest classifier. The best scores for the self-assessment models are highlighted.

Network	**MA**	**MR**	**WA**	**Baseline**
Mix Dataset				
Vgg19	0.445	**0.717**	0.637	0.858
Inception v3	0.526	**0.697**	0.448	0.848
Xception	0.656	**0.688**	0.459	0.845
ConSemble	0.616	**0.670**	0.482	0.896
ImageNet 2017 Validation				
Vgg19	0.587	**0.702**	0.671	0.855
Inception v3	0.608	**0.648**	0.500	0.846
Xception	**0.731**	0.627	0.548	0.847
ConSemble	**0.702**	0.670	0.603	0.865
ImageNet-A				
Vgg19	0.576	**0.607**	0.553	0.681
Inception v3	**0.568**	0.500	0.536	0.582
Xception	0.567	**0.592**	0.589	0.607
ConSemble	**0.613**	0.458	0.537	0.591

Statement of Authorship

Thesis: Self-Assessment of Visual Recognition Systems based on Attribution

Name: Kirchheim **Surname:** Konstantin

Date of birth: 12.11.1994 **Matriculation no.:** 205415

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning. I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the lawenforcement agency.

Magdeburg, October 9, 2024

Acronyms

AUROC Area under the ROC. 62, 63, 65, 66, 69, 79, 85, 86, 88, 93, 94, 107, 109–112

CAM Class Activation Map. 15, 26, 32, 90

CDF Commulative Density Function. 43, 44

CNN Convolutional Neural Network. iii, xi, 1–3, 5–8, 11–20, 22, 23, 25–28, 30, 32–34, 36–38, 42, 45, 46, 49–51, 53–57, 62, 65, 66, 69, 71–73, 76, 80, 82, 86–93, 95, 96, 109

DNN Deep Neural Network. 8

FC Fully Connected. 5, 7, 8, 12, 13, 38, 59

FPR False Positive Rate. 61, 65–67, 80

HPO Hyper Parameter Optimization. 46, 93, 95

MoE Mixture of Experts. 11, 96

NCM Nearest Class Mean. 9, 23, 35, 109

PDF Probability Density Function. 42, 43

ROC Receiver Operating Characteristic. ix, x, 61–66, 68–72, 78–83, 108

SVM Support Vector Machine. 8, 22

TPR True Positive Rate. 61