Using Attention Techniques for Explainability of Deep Learning Models in Computer Vision

Bachelor Thesis

for the

Bachelor of Science

from the Course of Studies Computer Science

at the Cooperative State University Baden-Württemberg Stuttgart

by

Niklas Ullmann

30th August 2021

Time of Project	12 Weeks
Student ID, Course	4173037, INF18A
Company	IBM Deutschland GmbH, Ehningen
First Reviewer	Marlene Spangenberg
Second Reviewer	Prof. Dr. Bernd Schwinn

Author's declaration

Ich erkläre hiermit ehrenwörtlich:

1. dass ich meine Bachelor Thesis mit dem Thema *Using Attention Techniques for Explainability of Deep Learning Models in Computer Vision* ohne fremde Hilfe angefertigt habe;
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;
3. dass ich meine Bachelor Thesis bei keiner anderen Prüfung vorgelegt habe;
4. dass die eingereichte elektronische Fassung exakt mit der eingereichten schriftlichen Fassung übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Hereby I solemnly declare:

1. that this Bachelor Thesis, titled *Using Attention Techniques for Explainability of Deep Learning Models in Computer Vision* is entirely the product of my own scholarly work, unless otherwise indicated in the text or references, or acknowledged below;
2. I have indicated the thoughts adopted directly or indirectly from other sources at the appropriate places within the document;
3. this Bachelor Thesis has not been submitted either in whole or part, for a degree at this or any other university or institution;
4. I have not published this Bachelor Thesis in the past;
5. the printed version is equivalent to the submitted electronic one.

I am aware that a dishonest declaration will entail legal consequences.

Stuttgart, 30th August 2021

Niklas Ullmann

Abstract

As Artificial Intelligence (AI) models have become more and more complex in recent years, the need for explanations has become greater to increase comprehensibility and trust in the models as trust is becoming increasingly important since AI models are starting to influence decision-making in most aspects of our lives. At the same time, the concept of attention in AI models emerged in the field of Natural Language Processing (NLP), which makes it possible for models to focus on the important things in a complex input and helps us to understand on which things the algorithm has focused. After the good results in NLP, attempts were also made to transfer the attention concepts to the field of Computer Vision (CV). In this bachelor thesis, the newly developed attention-based vision transformer[Dos+21] is applied, and its results are visualised and explained with the help of calculated attention. This novel explainer is compared with an already established explainer called LIME[RSG16a, cp.]. The comparison and the quality determination are carried out by using newly published quantitative metrics.[San+20, cp.] This bachelor thesis is the first paper to-date to independently evaluate these new quantitative evaluation metrics of expainability. A closer look at the metrics revealed that they have a logical basis and, therefore, can measure the quality of explainers. However, the attention-based explainer proposed in this bachelor thesis performed slightly worse than the LIME. This could be partly caused by the functionality of the metrics, the slightly worse general performance of the model, and the explainer itself. With a little fine-tuning, however, most of the problems found can be solved.

In summary, the intrinsic attention-based explainer developed in this bachelor thesis is a good explainer with much potential. It combines attention with the explanation of AI models and thus contributes to more comprehensibility and trust in those models.

Contents

Acronyms

AI	Artificial Intelligence
CNN	Convolutional Neural Network
CV	Computer Vision
DARPA	Defense Advanced Research Projects Agency
FC	Fully Connected
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
GT	Ground Truth
IBM	International Business Machines Corporation
LIME	Local Interpretable Model-agnostic Explanations
LSTM	Long Short Term Memory
MLP	Multi Layer Perceptron
NLP	Natural Language Processing
NN	Neural Network
PDP	Partial Dependency Plot
ReLU	Rectengular Linear Unit
ResNet	Residual Network
RNN	Recurrent Neural Network
SHAP	SHapley Additive exPlanations
TN	True Negative
TP	True Positive
ViT	Vision Transformer
xAI	Explainable Artificial Intelligence

List of Figures

List of Tables

Listings

1 Introduction

Everyone knows the following situation: you are at a party and want to talk to the other person, however, music is playing loudly, other people are also talking to each other, but your brain still manages to tune out all the background noise so that you can concentrate solely on your own conversation. This type of human attention was first defined in 1953 by the psychologist Collin Cherry as the Cocktail Party Effect.[HC05, cp. p.1875-1876]

Attention

Like so many technical achievements and innovations in AI, these originate from observations of the human body. For example, neural networks function like the human brain, in which individual neurons are interconnected and transmit signals to generate specific reactions.

Before the introduction of attention, conventional AI models for text translation suffered from the problem that they often lost context or were exceptionally computationally intensive, especially with long sentences. In 2017, in their paper "Attention is all you need"[Vas+17, cp.], researchers from Google conceptualised the first implementation for an attention-based transformer architecture that put the individual words in a sentence into context and was thus able to translate even longer sentences efficiently and well.

After the success of the attention-based transformer, attention techniques were increasingly utilised in other use cases, like machine translation or text summarization. Attention has been implemented in various use cases, but in the field of Computer Vision (CV), there have been recent approaches to use attention, for example, in image classification.For example, when classifying an elephant, the AI model can concentrate on the head and trunk of an elephant and fade out the "unimportant" background.

Explainable AI

AI models used today are getting better and better, but they are also becoming more and more complex. The problem is that many users enter data but cannot understand or comprehend the decision or prediction of the AI model. Especially in real-world applications, however, comprehensibility is extremely important in order to be able to trust the model as a user even in critical situations. For example, customers should understand

why the AI model rejected the loan they requested from their bank or why a model decided on a certain treatment for a disease. Thus, the keyword here is explainability or Explainable Artificial Intelligence (xAI).

The explainability of AI models helps users understand the decisions and predictions of the model and helps developers improve the models. With explainability, a bias in the model can be better recognised, and, for example, the fairness of the model can be increased.[GSC21a, cp.] Due to the high demand for reasonable explanations, many efficient tools and algorithms were developed in recent years.

Goal of this Thesis

Therefore, this paper aims to combine the two current hot topics of attention and explainability of AI models and test whether this novel explainer produces good explanations.

The complex attention-based AI models also belong to the black box models. Looking at the visualised attention for texts today, it can be deduced where the attention of the model lies. This suggests that the calculated attention weights could also be used to explain attention-based models in CV tasks through visualisation.

For this purpose, a pre-implemented architecture of a Vision Transformer (ViT) is used for the case study presented in this thesis, which was published by a team of Google researchers in 2020[Dos+21, cp.]. When classifying images, a classical task from the field of CV, the explainer should visualise the calculated attention weights and thus show the user where the model's attention lay and which areas were more important than others for the final decision. The aim is not to develop an entirely new method of explaining a ViT, but rather to adapt and use the already known visualisation methods of attention from already known other transformers. In a realistic case study, the ViT and its explainer are compared against a conventional architecture for classifying images and an already known and accepted explainer in the industry. The quality of the explainers is evaluated and compared based on quantitative metrics that have recently been defined in a newly published paper[San+20, cp.].

2 Literature Review

In this chapter, the theoretical foundations for the two topics of attention and explainable AI are established, which form the basis for the later case study. In particular, general definitions and procedures are emphasised, and individual parts are explained using reality-based examples.

2.1 Attention

In recent years, the combination of deeper neural networks and Graphics Processing Units (GPUs) created new parallel computational capabilities which revolutionized machine learning, significantly improving the performance of various tasks, such as language modelling, image classification, machine translation, object recognition, and much more. The conventional models only work moderately well with complex sequential data. The attention mechanism that has emerged in recent years, based on nature concepts, will provide a paradigm shift in this area. This and its various manifestations make it possible for models to pay more attention to certain parts of the input data than others, thus favouring the critical points over the less important ones.[HA21, cp. p.1]

In the following subchapter, the definition of the attention mechanism will be derived from the literature in general, and its various forms will be explained and illustrated employing a realistic example.

2.1.1 Natural Attention

Many techniques and innovations in today's AI field are based on essential functions of our body. For example, a neural network in AI imitates the biological brain, consisting of individual neurons connected to varying degrees. Both structures take an input signal (nerve stimulus or numerical data) and pass it through the neurons to generate an output. In the case of the human brain, this can be an action that the body performs, and in the case of an artificial neural network, it can be data.[Nwa20, cp. p.4-6]

The foundations of artificial attention also lie in nature. Techniques of attention are known from scientific studies, both in humans and in animals. Attention is an essential part of the cognitive process. However, human attention has been the most widely researched. The

human attention system has the overall effect of allowing humans to focus on information relevant to our current tasks while also reacting to unanticipated events. This characteristic was critical for survival in evolution, as humans must be continually attentive to potential hazards while also performing complicated, demanding jobs. Nevertheless, we still benefit today from the concepts of attention.[Hel13, cp. p.43-44]

One of the first and still best-known mechanisms of human attention was observed and defined as early as 1953 by the psychologist Colin Cherry, the so-called "cocktail party effect". This effect refers to the ability of the human brain to concentrate on single audio signals in a mixed-signal environment. For example, at a party, where there are various acoustic signals (music or voices), the human brain can filter out all background noises and focus only on a conversation partner's voice.[HC05, cp. p.1875-1876]

2.1.2 Attention Definition

The definitions of artificial attention, from now on just called attention, vary widely in the literature.

Hassabis et al. define attention in their article on "Neuroscience-Inspired Artificial Intelligence" as a "mechanism for perception."[Has+17] Meanwhile, Galassi et al. characterise attention as an "approach of the relevance of input elements. In that way, neural architectures could automatically weigh the relevance of any region of the input and take such a weight into account while performing the main task."[GLT20] One of the most detailed and general definitions is provided by Terry-Jack in his online article, in which he describes that the "attention mechanism calculates the dynamic (alignment) weights representing the relative importance of the inputs in the sequence [...] for that particular output [...]."[Ter19]

From all these definitions, it can be deduced that similar to natural attention, attention within AI systems is also about weighting a large amount of input data in such a way that the AI model can place particular emphasis on the important elements of the input and accordingly ignore not such essential elements on its own.

2.1.3 Types of Attention Techniques

Attention layers are already integrated into many AI models today and do a good job there. However, attention techniques are distinguished by how they handle the input.

Global vs. Local Attention

Let's stay with the short example sentence from the previous chapter. Four words are no problem for an attention calculation. In contrast, especially in NLP, there are often very long and complicated nested sentences, where it is not so easy to find out which words refer to each other and which open up a different context. In the field of NLP, a distinction is made between global and local attention.

With global attention, the attention is calculated on the whole context. This means that every word of a single long sentence could be included in the attention calculation. Local attention also called sliding window attention, only includes a particular area around the current word. However, local attention is still easy to calculate because it maps the probability distribution with a Gaussian function over all words in the window and is still differentiable.[LPM15, cp. p.3-4]

Soft vs. Hard Attention

At the same time, the field of CV distinguishes between soft and hard attention.

Soft attention, like in the middle, works like global attention from the field of NLP. However, soft attention includes all inputs of the image in the calculation. A categorical distribution is calculated over the whole sequence of elements in soft attention. The probabilities that result represent the relative importance of each element and are utilized as weights to generate a context-aware encoding. As a result, soft attention requires a few parameters and takes less time to compute.[She+18] This structure makes soft attention differentiable. The model can be trained through gradient descent by simply back-propagating the errors.[Wei18, cp.]

In contrast to soft attention, hard attention, like on the right side of the figure, selects a subset of elements from an input sequence. The hard attention mechanism forces a model to focus primarily on the most critical features while completely ignoring the rest. However, hard attention is time inefficient with sequential sampling and non-differentiable due to its combinatorial nature, as all possible combinations of input sequences must be tested. Therefore, it cannot be optimized by back-propagation.[SKS16, cp.] Hard attention is trained via other algorithms like the reinforce algorithm defined by Williams in 1992, which is unfortunately computationally expensive.[Wil92, cp.]

Figure 2.1: Visualisation of Soft and Hard Attention[Aro20]

The figure 2.1 shows very well how soft and hard attention behaves differently with the same input image. The input image is on the far left. The middle picture illustrates soft attention very well. It can be seen that after the attention calculation, all pixels from the original image are still present. The coverage of the individual pixels with darker colours shows the attention. Lighter areas have a higher attention, and darker areas show areas with less attention. In comparison, on the right image with the hard attention, only the calculated pixels with the highest attention are still present. All other pixels (the black area) were discarded. In this case, both attention calculations are circular around the woman's face, but the crucial pixels can also take other shapes.

2.1.4 Self-Attention

Self-attention is defined as the application of attention to a single context rather than several contexts. For example, only the relationship between words in a sentence is calculated and not extended to the whole context of the text.[Ram+19, cp.]

Especially in tasks of NLP, the context of words matters the most. If the word "bank" is a part of a sentence, the context is not immediately apparent, neither for the human nor for the machine. Without looking at the surrounding words, it could stand for the financial institution, the seating opportunity, or land along a river. With the help of this contextualisation through self-attention, the sentence can now be translated much better into other languages, as it is now clear which bank it refers to. For example the attention weight between "river" and "bank" is significantly larger, than the weights between "walk" and bank in the sentence of "walk by river bank".

Thats why self-attention is nowadays mostly used in text classification, sentiment analysis, translation and other NLP tasks.[Cha+20, cp. p.13-14] But also in various applications of CV, such as image classification.[Dos+21, cp. p.2] This suggests that self-attention can be used in many areas of AI and proves the general relevance of self-attention.

Calculating Self-Attention

Calculating self-attention for a single context like a sentence follows a single calculation.

A decent word representation is necessary for a NLP task since mathematical calculations are impossible with words consisting of letters. Therefore they have to be transformed into numerical values. A common way of transforming is word embedding. One way is to represent each word with an index of a vocabulary, but this method fails within larger vocabularies. Vector-based models fit these task much better. Word vectors represent continuous similarities between words in a high-dimensional space as a distance or angle between each. The idea behind that is to map every word to a point in space, where words with a similiar meaning are closer to each other.[Maa+, cp.]

As defined in the Inproceeding of Vaswani et al. the self-attention is calculated as a dot product.[Vas+17, cp.]

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Where Q (Query), K (Keys), V (Values) and the attention output are all vectors, the attention output is a weighted sum of the input values, with the weight allocated to each value determined by the query's compatibility function with the relevant key. Since the attention for the own context is calculated in the self-attention, Q, K, and V are equal.[Vas+17, cp.]

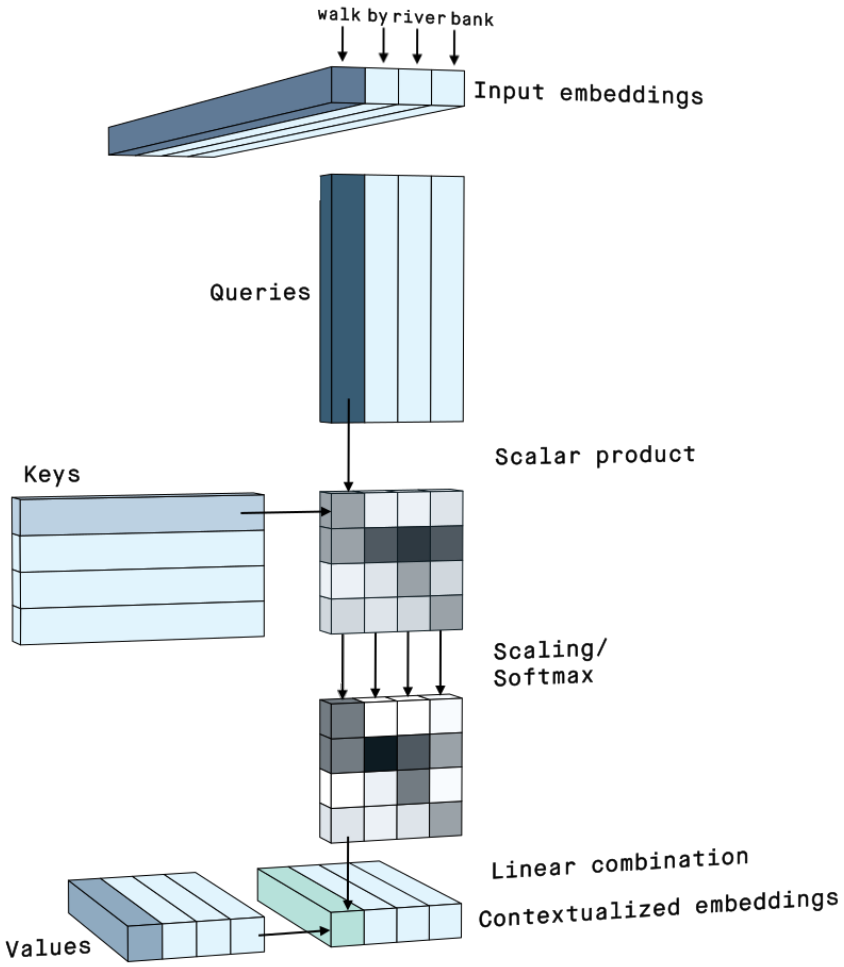


Figure 2.2: Visualisation of the Calculation of Self-Attention[Fut20]

Going from top to bottom on figure 2.2: Each token in the sequence of Q, K and V is represented as a vector. First, each token of Q is scalar multiplied with each token of K, which generates a two-dimensional matrix from both one-dimensional token vectors in each sequence.

When the two token vectors are more correlated or generally more similar, the scalar product is higher, which indicates that they have a strong relationship. So, for example, the scalar product of the two tokens for "walk", both in Q and K, is very high because they are highly correlated. However, the scalar product for "river" and "bank" should also be pretty high because they are semantically connected through word embedding. Conversely, if the tokens have less similar content, the scalar product is lower, suggesting that they are less related to each other.[LG14, cp. p.3]

The scalar product is calculated for every possible pair of tokens from Q and K. The scalar product is then divided again by the root of the size of the token vector to normalise this value and avoid huge values.

The calculated matrix represents the extent to which the individual tokens from Q correlate with the tokens from K.

The next step is to apply the softmax function column by column. The softmax function is usually used as an activation function and calculates the probability over a vector of real numbers. Adding up all numbers in the vector after applying the softmax functions returns 1. In this case, the softmax function exponentially amplifies larger values and brings negative and small values near zero.[Nwa+18, cp.]

Finally, the calculated attention weights have to be projected back on the one-dimensional input sequence. This projection is done with a linear combination of each input token from K and the values from the softmax function. Now the new sequence can be seen as a contextualised embedding of the original input sequence.

In particular, if a token has a strong connection to another one, it's new embedding will be made of a big part of the corresponding other token. If a token does not relate to another, its new contextualised token will be nearly identical to the input token.

Multi-Head Self-Attention

Multi-head attention is a mechanism that increases the representation power of an attention layer, where multiple attention units, as described in the chapter before, operate on different input embeddings, with each attention unit being referred to as an attention head. Therefore, the input sequences can be embedded in different ways, for example, in order to be able to establish other semantic connections. On the other hand, the input dimensions of the embedding can be reduced per head and distributed over several heads to shorten the individual calculations. After each attention head is calculated, they are concatenated to a single head involving weights learned in the training of the AI model. Using multi-head has shown to be more effective than using a single head attention layer.[Bho+, cp. p.3-4]

$$Multihead(Q, K, V) = Concat[head(X)_0, \ldots, head(X)_n] \tag{2.2}$$

$$where\ head(x)_i = Attention(Q * W_{Qi}, K * W_{Ki}, V * W_{Vi}) \tag{2.3}$$

2.1.5 Real Usecases of Attention

As shown earlier, the attention mechanism has been integrated into various application fields and AI models in recent years due to its versatility and some performance improvements.

The first and most famous architecture involving attention is the transformer neural network architecture, a sequence to sequence model, which takes an input sequence with variable length and outputs a sequence with variable length. This architecture was first described in the foundation paper "Attention is all you need", written by various Google researchers and the University of Toronto.[Vas+17, cp.]

This type of model is used for translating sentences from one language into another.

Previous attempts for translating longer sequences were computationally expensive because they could not utilise parallel computing from modern GPUs (Long Short Term Memory (LSTM)) or could not deal with them at all (Recurrent Neural Network (RNN)).[HS97, cp. p.1]

The solution for those problems is the transformer architecture, which employs an encoder-decoder architecture with attention techniques. The overall architecture is defined as follows:

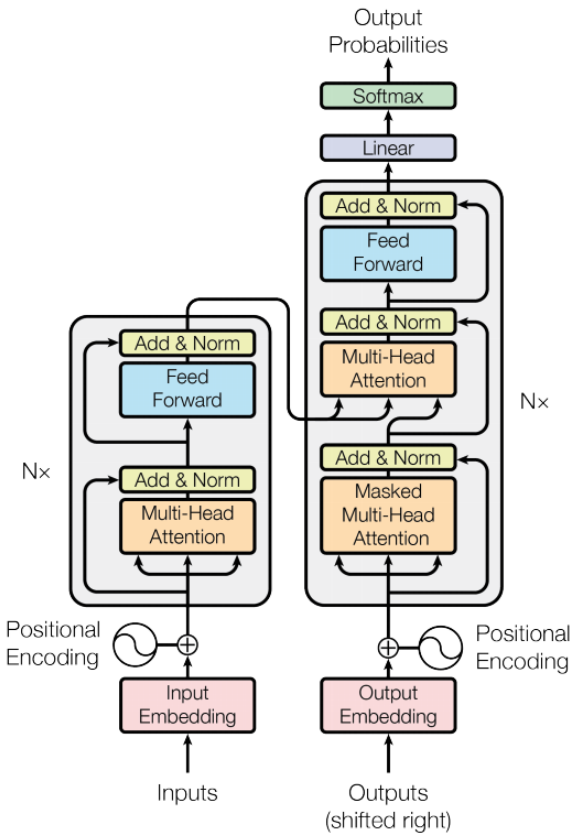


Figure 2.3: Transformer Architecture[Vas+17, p.3]

Generally, the transformer architecture, displayed in figure 2.3, consists of three major components: the encoder, decoder, and output.

It is essential to recognise that the decoder can only determine the next word of the translation. The translation of the individual words is done until the end of the source sentence is reached. As input, it always takes the complete sentence in the source language and the previously translated part of the sentence in the target language.[Vas+17, cp. p.2]

Before the encoder or decoder block, the words are embedded into a vector, like described before. Furthermore, a positional encoding is added to maintain the order of the words in this contextualised embedding for the translation.[Vas+17, cp. p.5-6]

The encoder block consists of a multi-head self-attention layer and a feed-forward layer. The attention layer determines how relevant the i-th word is for each other word in the sentence. For each word, an attention vector is calculated that reflects the contextual dependencies. The subsequent feed-forward layer is a simple neural network applied to each attention vector for each word. These reshape the attention vectors into a shape that is compatible with the next block. Since the calculations for each word are the same and not dependent on each other, they can each be calculated in parallel.[Vas+17, cp. p.3]

During the training of the transformer, the corresponding sentence in the target language is fed into the decoder block. In production, the already translated partial sentence is then transferred. As with the encoder block, these are also embedded first and provided with a positional encoding. The first layer in the decoder block is the same as in the encoder block. A multi-head self-attention is calculated on the input vector. The only difference is that this is a masked attention block. It can take all words of the source sentence into account but only to the words in the target sentence that were already translated. The rest is masked out in the training process. Among other things, this prevents it from simply outputting the next word from the target sentence. The second block combines the two now all calculated vectors from the encoder block and the first layer of the decoder block. This attention layer calculates how dependent individual words from the input sentence are on the words from the target sentence. This is where the main translation work happens. The output of this layer is accordingly attention vectors for each word in the input and output. The next layer is again a feed-forward network, which serves the same purpose as in the decoder. In addition, it makes the output more compatible for the next step.[Vas+17, cp. p.3]

The output block is now responsible for converting the calculated numbers and vectors back into words. This consists of a linear layer, which has the dimensions of the target language. The following softmax function transforms the layer's output into a probability distribution, which humans can now interpret. The final word of the translation is the word with the highest probability.

2.2 Explainable Artificial Intelligence

AI algorithms are increasingly being utilised in critical sectors such as healthcare or banking to make decisions and execute autonomous activities. Explainable Artificial Intelligence has sparked a lot of academic and public interest due to the necessity to comprehend AI to improve, contest, create appropriate trust, and better engage with AI systems.[Lia+21, cp.] However, in recent years models have become better and better at their tasks and, in some cases, better than their human counterparts. In 2011, IBMs AI, called Watson, beat the best human contestants on the American quiz show Jeopardy. The aim was to understand questions in natural language and answer them in one natural sentence.[Fer12, cp.] Nevertheless, the models behind those tasks have also become more complex to achieve this performance.

Typically, the term black-box model refers to models with a complicated structure that humans find difficult to comprehend. This term is frequently used to describe a model with many coefficients or complicated mathematical transformations. In contrast to a black-box model, a glass-box or white-box model is a model that is simple to comprehend. It has a straightforward structure with only a few coefficients.[BB21, cp.]

In order to be able to explain these complex black-box models, a whole field of research is concerned with the explainability of AI. In the following, explainability will be defined, its different types will be shown, and two well-known libraries in the industry will be illustrated with examples.

2.2.1 Explainable Artificial Intelligence Definition

Explainability is defined in different ways in the overall literature.

The Defense Advanced Research Projects Agency (DARPA) defines xAI as "AI systems that can explain their rationale to a human user, characterize their strengths and weaknesses, and convey an understanding of how they will behave in the future. [...] DARPAs objective [is] to create more human-understandable AI systems through the use of effective explanations."[GA19]

"Explainable Artificial Intelligence (xAI) [...] focuses research on machine learning interpretability and aims to make a shift towards a more transparent AI. The main goal is to create a suite of interpretable models and methods that produce more explainable models whilst preserving high predictive performance levels."[CPC19]

"Machine learning techniques that make it possible for human users to understand, appropriately trust, and effectively manage AI."[Hol12, p.3]

All definitions of xAI are united by the idea of clarifying the reason for a decision of an AI model and making it understandable for a human being. This explanation forms the basis for interpreting the results for a human being, regardless of whether it is a technical person or a user using the system.

Interpetability

In addition, the word interpretability is also used in the literature on several occasions. Interpretability is often defined as follows: "the ability to explain or to provide the meaning in understandable terms to a human."[Bar+20]

Interpretability is a "desirable quality or feature of an algorithm which provides enough expressive data to understand how the algorithm works. Here, interpretable domains could include images or text which are comprehensible by humans."[DR20]

The two terms explainability and interpretability are often used interchangeably in the literature. However, they can be clearly distinguished with the following definition:

An explanation clarifies anything that is not clear, whereas interpretation tells the meaning of something. In other words, this means that the trunk of an elephant in a picture was marked for the picture's classification, is considered an explanation. What conclusions the human draws from this and whether the marked area is important or unimportant for the AI model is then a matter of interpretation.

2.2.2 Need for Explainability

Some AI models are more explainable than others by their fundamental nature. For example, the decisions of a rule-based AI model can be understood very well by humans because the boundaries for individual decisions are very clearly defined. If, for example, the rule is set that only people with a monthly income of more than 5,000 euros get a loan from a bank, both the bank employee and the customer can easily understand why the customer with a monthly income of 2,500 euros did not get a loan. Even simple regressions with a few parameters may still allow technical users to understand by hand why and how which value was calculated. However, such models with few parameters or simple rules often have the problem of not being accurate enough for an actual use case. Nevertheless more accurate models are much more complex and often difficult for a human to understand or even comprehend. This phenomenon is called the accuracy-explainability tradeoff because an AI model can either be accurate or easy to explain.[Dut21, cp. p.4-5]

AI models are increasingly being used in real-world scenarios to improve or even automate processes. This development makes it necessary to examine the models and their development process as closely as possible. "If we [the developers or users] cannot explain the algorithms, we cannot argue against them, verify them, improve them, or learn from them."[GSC21b] This problem could be solved with a good explanation of the model.

A good explainability of AI models brings many advantages for users, developers and may improves the model itself.

xAI for Users

Good explainability increases users' trust, as they no longer throw just data into a black box AI model and are then faced with a final decision without knowing what the model has done. The explanation can help for more understanding and comprehensibility for the decision. Transparency is also increased by good explainability, as everyone can understand how a model decides.[DR20, cp. p.4-5] If, in a fictitious example, an AI model is to verify the authenticity of documents for authority, it is also essential, in addition to the assessment of authenticity or forgery, at which points the model has determined the forgery.

xAI for Developers

In addition, good explainability also helps the developers, to understand their models better and validate them.

There are usually not huge datasets on which the models can be trained in the real world. Therefore, it is even more critical to determine whether a model has learned the generally correct things or only focuses on unimportant things that fit the small dataset. This can also prevent overfitting, for example, and increase the general performance of the model on actual data.[GSC21a, cp. p.1287]

The best example of this is a project in which pictures of wolves and huskies were classified. After training, the test set worked very well, but the results were worse when used with authentic images. The model concentrated mainly on the white snow in the background of the huskies and not on their muzzles or other features, as shown in figure 2.4. Such misbehaviour is often not apparent when only the data is taken into account, and a good explanation remedied this.[RSG16a, cp. p.8-9]

(a) Husky classified as wolf (b) Explanation

Figure 2.4: Husky Explanation Example[RSG16b]

In addition, good explanations can also prevent a biased model and thus increase the overall fairness of the model.[GSC21a, cp. p.1287] For example, it can be detected early in the development process if a model has been trained in a disadvantageous way for specific user groups, and corrections can be made immediately. If sticking to the example of credit granting, it can not happen that people from certain countries of origin are categorically excluded from credit. Such unfair behaviour of the model could hardly be recognised without an explanation.

AI models are nowadays also used for automation, where the model must deliver good results stably and reliably.[Yal20, cp.] Here, too, a good explanation can help to check the reliability and highlight possible edge cases.

2.2.3 Types of Explainability

AI models and their decisions can be explained in many different ways. A distinction is made about whether an explanation only applies to specific data input or whether the entire model gets described by the explanation. Furthermore, a distinction is made as to which mathematical methods are used to create an explanation. An explanation can also be characterised as to whether the explanation is built directly into the model or other tools have to be used to generate an explanation. Finally, a distinction can also be made about whether models are explainable by themselves or require an explicit explanation. Many classifications can also be made simultaneously and are usually closely linked.[DR20, cp. p.2]

Distinction by Scope (Local vs. Global Explainability)

Local explanations are used to understand the decisions of a model on particular data input. The methods for a local explanation approximate the complex properties of the original model with the help of a simpler model. This method is possible because the model is not entirely approximated, but only the model's properties for the input's data point is modelled. The rest of the model is neglected by the approximation. For example, for applications from the DeepLearning field, which are based on neural networks, a gradient method is often used for explanation. In this case, the data is passed through the neural net, and the local gradients are observed, for example, to be able to explain image classification. Then, those gradient values are visualised with the help of maps that overlay those values with the image.[AB18, cp. p.52148]

Global xAI, unlike local explanation, explains the entire model and its decisions, not just the behaviour of single data input.

An example of an algorithm for globally explaining an AI model are Partial Dependency Plots (PDPs), proposed by Friedman in 2001. PDPs are a visualisation technique that shows the impact of certain features or subsets of features on the algorithm's predictions.[Fri01, cp. p.26-27]

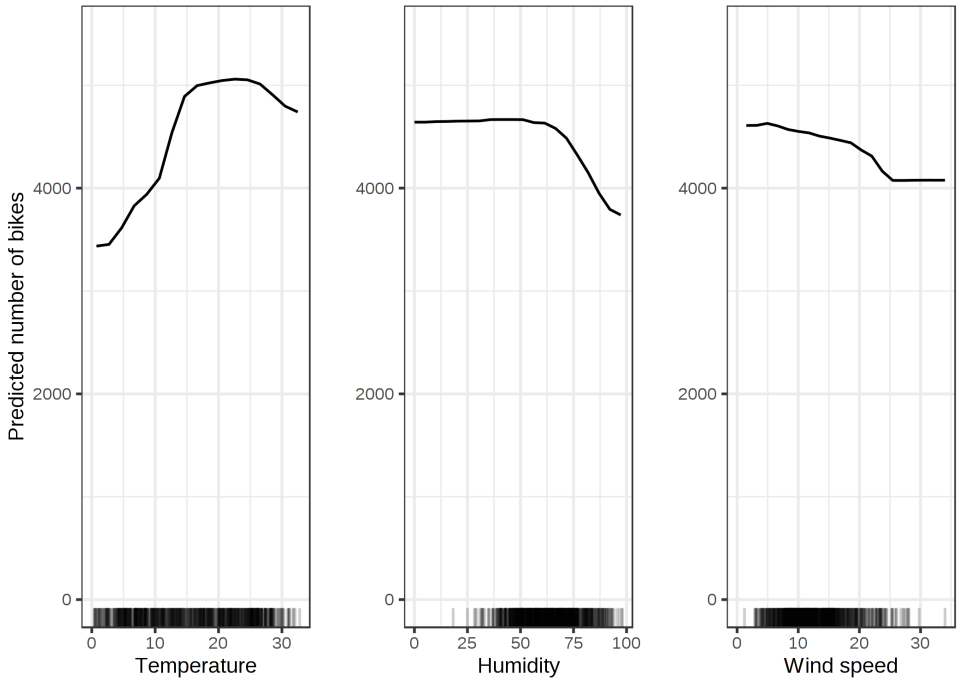


Figure 2.5: Partial Dependence Plot - Bicycle Count Prediction[Mol19]

The figure 2.5 shows the dependencies of individual features in relation to the number of bicycles rented in a day according to the model prediction. For example, in the plot on the

left, which shows the temperature in degrees Celsius and the number of bicycles expected to be rented, the higher the temperature, the more bicycles will be rented. However, as soon as the temperature rises above 25°C, the model predicts a decreasing number of rented bicycles. From the other two PDPs, it can be seen that the model behaves similarly with increasing humidity and wind speeds, predicting fewer borrowed bikes.

Distinction by Mathemathical Method (Backpropagation-based vs. Perturbation-based)

The explainable model's core algorithmic principle can be broadly classified based on the implementation methods. For example, backpropagation-based or perturbation-based methods can be used to create an explanation of a model.

In backpropagation-based approaches, the explainable algorithm does a forward pass through the neural network, then creates attributions using partial derivatives of the activations during the backpropagation step.[DR20, cp. p.3] With that, saliency maps or similar can be generated and visualised for the user, like in the following figure 2.6.

Figure 2.6: Saliency Map Example[MCF20, cp. p.6]

Within the figure 2.6, the original photos are displayed alongside their associated saliency maps. Each saliency map's hue corresponds to the layer of activation. Early in network processing, blue or violet patches, for example, are engaged. Later in the network, the yellow and red parts are active. White spots can be found in all network layers. According to Mundhenk et al., those areas correspond to the essential features in the image. In the case of the tractor image, this would mean that the area around the bonnet was considered important in all layers of the model. Again, like in the dog's picture, especially the muzzle's contours and the eyes were marked as critical.[MCF20, cp. p.6]

Algorithms based on perturbation use occlusion, partially substituting features via filling operations or generative algorithms, masking, conditional sampling, and other techniques to change the feature set of a given input instance. It is observed how the output of a ki model changes when individual parameters of the input are changed. In most cases,

a single forward pass is sufficient to build attribution representations, with no need for backpropagating gradients.[DR20, cp. p.3] For example, how does a client's predicted credit score change when their monthly income doubles? Such questions can be answered or explained with the help of these methods.

Distinction by Origin of the Explanation (Intrinsic vs. Post-Hoc Explanation)

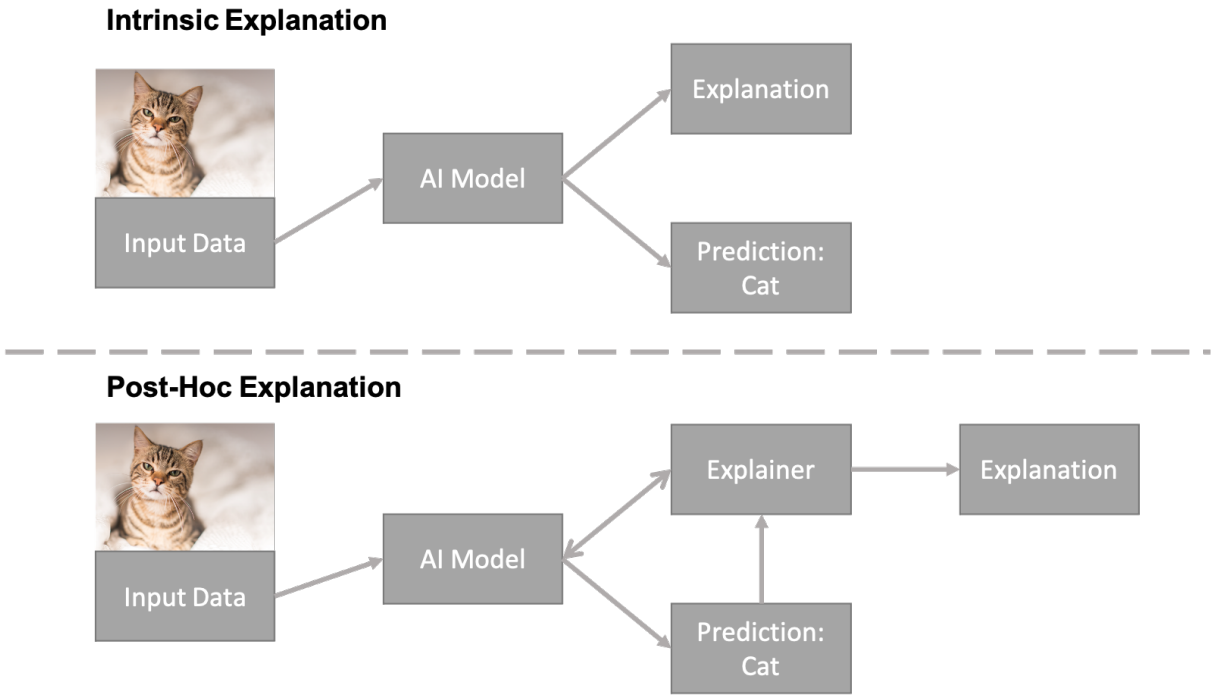


Figure 2.7: Intrinsic and post-hoc Explainability

The difference regarging the origin is drawn by the literature by the point in time when the explanation is generated. When the explanation is built-in the models, it is called an intrinsic explanation. The explanation often ends up being a by-product alongside the actual task, as shown in figure 2.7. However, these models are usually straightforward in structure, which makes the explanation easy, but they do not generally give such good results.[PV20, cp. p.6]

On the other hand, post-hoc explainability is achieved by analysing the model and its outcome, as shown in the figure 2.7 above. This explainability is often achieved by generating a simple model from the original complex model, which can then be better explained. Thus, intrinsic explanation models can explain post-hoc models, but not always the other way around. Moreover, like the models themselves, these explained models suffer from the transparency-accuracy trade-off; intrinsic models typically provide accurate explanations, but their prediction performance suffers due to their simplicity. In contrast,

post-hoc models frequently maintain the correctness of the original model but are more challenging to obtain a qualitative explanation.[PV20, cp. p.6]

Distinction by Model Dependency (Model-specific vs. Model-agnostic Methods)

Model-specific methods include techniques that can be used for a single specific architecture, like explanations algorithms for decision trees. It requires training the model using a dataset. Therefore, intrinsic explainability methods are, by definition, model specific. Model-specific techniques might compromise or enhance the performance of a model. Those are usually preferable when having the opportunity to design an own model.[WL20, cp. p.63]

Model-agnostic methods, on the other hand, contain techniques that can be used across many models. These will not affect the model's performance since the explanation only happens after the training and processing. These methods also require no (extra) training of the model. That is why post-hoc methods are usually model-agnostic.[WL20, cp. p.63] The explainers described in the following section are also model-agnostic.

2.2.4 Explainable AI in Real World Scenarios

In addition to a wide variety of algorithms and methods for creating an explanation for AI models, two libraries have become established in the industry in recent years that are mainly used: Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) .[Sag19, cp.]

Local Interpretable Model-agnostic Explanations (LIME)

LIME focuses on training a local model around the input data, which should be explained. LIME's approach is relatively intuitive and straightforward.

The initial situation is the original black box model, where data is input and predictions are output. The goal of LIME is to understand why the model has made a particular decision. In the process, the original data point is changed repeatedly so that a new dataset is created, which lies roughly in the input region. Using the predictions of the original model and the dataset, a new but simpler model is now generated. This new model only has the properties of the original model at the data point, as the rest of the model is unimportant for the explanation. A linear model is utilised for this approximation. LIME can be used for AI models with tabular data, image and text input.[Mol19, cp.]

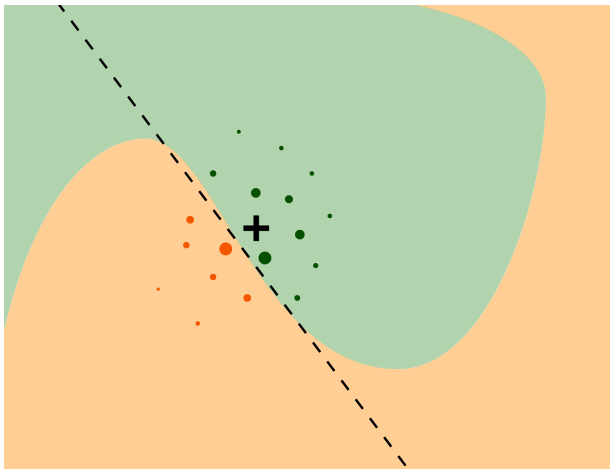


Figure 2.8: LIME Method Explanation[BB21]

Shown in this figure 2.8 is the visualisation of a complex black-box model and the simplified model created by LIME.

The original model represents a binary classification between two classes, with class 1 represented by the green area and class 2 by the orange area. The complexity of the model is clear to see, as a concatenation of several polynomial functions can only represent the boundary between the two classes. The cross in the middle is the data point, which classification is to be explained by LIME. The data point in this example has the two features of the X and Y-axis. However, this could also be higher dimensional. LIME creates further data points based on the original data point by modifying it slightly. All these points lie in the vicinity of the original data point and are represented here by the green and orange dots, respectively. Their colouring corresponds to the prediction that the model has made for this point. A simplified model is created with the help of these points. The dashed line represents this. The size of the points in this visualisation indicates the weighting for the new model. The further away the points are from the original point, the less meaningful they are for the new model. It is easy to see that the new model corresponds to a linear function, which is easy to explain. This new model is very accurate at the original data point but loses its accuracy with increasing distance from it, which is unimportant for explaining the individual point.[BB21, cp.]

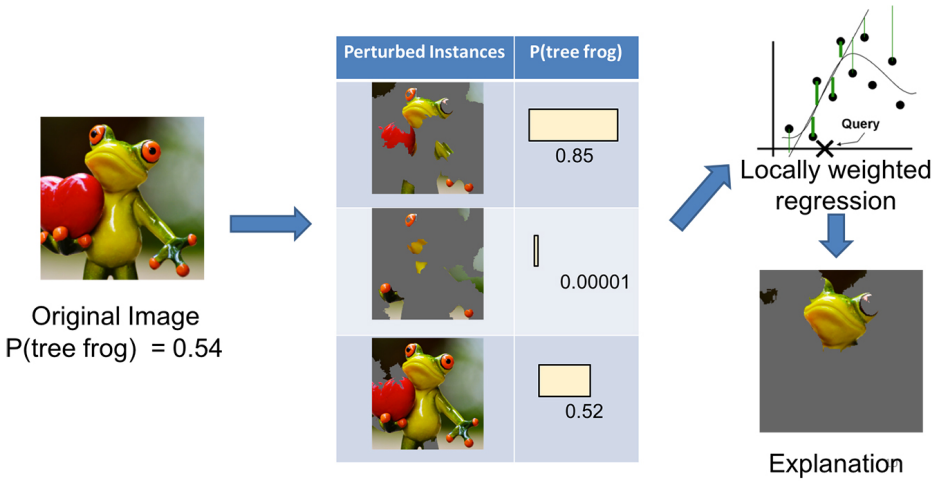


Figure 2.9: LIME Explanation on Image Data[Tul16]

As shown in the figure 2.9 LIME works similarly with image data. In this example, the question is about which part of the image was decisive for the model to classify it as a frog. For this purpose, the original image was changed several times and classified by the model. With the help of the probabilities with which the model believes to have recognised a frog, a simplified model could be created and thus explains that the frog's head was the decisive part.

With these properties, LIME is a library for local explanations, which is model agnostic and can be used for various AI models, such as for models that work on text, tables or image data, but also for classification and regression models.

SHapley Additive exPlanations (SHAP)

Compared to LIME, SHAP follows a different approach, which is based on coalitional game theory. SHAP's purpose is to compute the contribution of each feature to the prediction of a data point to explain it.[LL17, cp. p.4]

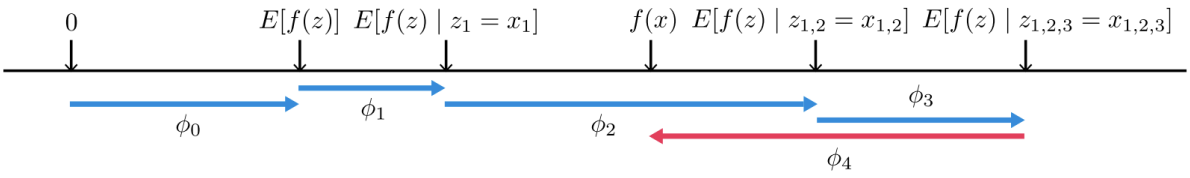


Figure 2.10: SHAP Value Attribution[LL17]

Imagine a customer who wants to apply for a loan at a bank, where the bank employee is supported by an AI model that predicts how high the probability is that the customer will not pay back his loan. However, the model is a complex black box model without any explanation or rationale for the prediction.

SHAP works as follows: First, SHAP assumes that nothing is known about the customer and uses this empty feature set to calculate the base risk. In this case the base risk is: $E[f(z)]$ in figure 2.10. Then, step by step, the customer's data is added to the feature set, and it is observed how the model's output changes. For example, the customer's risk is unlikely to repay increases because the customer already has a current loan. In this example the second blue line could visualise the increas of the risk, to the point of $E[f(z)|z_1 = x_1]$. This concept is carried out until all features are added to this observation. However, this does not only go in one direction, but a feature can, of course, also have a positive effect on a prediction, as the red line shows, in this step, for example, the feature of the client's income could have been added. As a result, the client has a steady and high income, making it less likely for him not to repay the loan. Here represented by the red line.[Mol19, cp.]

In a linear model with independent features, the features can be inserted one after the other. However, all possible combinations of features must be calculated for more complex models, taking the average of all impacts on the output.[LL17, cp.]

SHAP, like LIME, is a local model that is model agnostic. However, compared to LIME, SHAP ensures that all features are equally distributed in the calculation. On the other hand, this requires much calculation, especially if the model is very complex or the feature space is higher-dimensional.[Mol19, cp.]

3 Case Study

In this chapter, a case study is conducted in which a novel algorithm for the explanation of attention-based AI models is implemented. This explainer is compared to the already established algorithm LIME for the explanation with the help of several quantitative metrics.

3.1 Current Status in Scientific Research

Attention techniques are increasingly becoming established in concrete applications of NLP, as already shown in chapter 2.1. Thus, transformer architectures are not only used for machine translations but also for text summarization or similar tasks. Nevertheless, attention is also slowly arriving in the field of computer vision. For example, in their paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale"[Dos+21], researchers from Google described how they successfully managed to classify images with the help of an attention-based transformer architecture. Moreover, the so called ViT showed a similar or even better performance in the tests than conventional models for image classification[Dos+21, cp. p9]. Thus, just as attention can already be used to explain NLP models today[GLT20, cp. p.2], it seems reasonable to assume that the calculated attention weights can also be used for computer vision tasks to better understand the model's outcome and explain it to a user.

3.2 Goal of this Case Study

This case study aims to answer the extent to which an explainer based on the internal attention weights provides a good explanation. Furthermore, the quality of the explainer is evaluated based on quantitative metrics. In order to explain the attention-based AI model, a novel algorithm is proposed, which is oriented towards the handling of attention weights from NLP and self-attention. With the help of these attention weights, an explanation in the form of visualisation is created. To put the calculated metrics in perspective, the quality of an industry-established explainer, LIME, for a conventional convolutional neural network is measured and compared simultaneously. The aim of this case study is not to compare different algorithms for explaining the attention-based model, but to test the

quality of the proposed attention-based explainer. Furthermore, the aim is not to compare different reference architectures and their explainers with each other. Instead, the choice of the reference architecture is based on how well this architecture is established in industry problems. The same applies to the choice of the explainer for the chosen architecture.

3.3 Fundamentals and Methods of this Case Study

The following chapter describes the fundamentals, such as the dataset used, the types of AI models and the different approaches to explainability, and the methods used to measure and compare the two AI models and their explanations of this case study.

3.3.1 Implementation and used Libraries

The implementation of the case study is done in Python 3.8, as Python offers all kinds of tools and libraries for working with images and machine learning. The latest version of PyTorch, 1.8.1, is used as the library for the machine learning algorithms.

3.3.2 Dataset

ImageNette is used as the dataset for the image classification task in this case study. This subset of the ImageNet dataset is smaller and more suitable for this case study. The ImageNette dataset is provided by Fast AI and consists of 10 different classes[Doc21a, cp.]. In total, the dataset consists of 13.416 pre-labelled images, about equally distributed among all classes

In the figure 3.1 below, an example image is given for each class of the dataset. In addition, the class names and the corresponding class number are mentioned, which are used synonymously in the following chapters.

Figure 3.1: Classes of ImageNette Dataset

Data Augmentation

Sometimes AI models in CV tend to fit perfectly on the training data but struggle with generalising on data apart from the trainingsset, which is called overfitting. Therefore, one of the most important topics is to improve these models' generalization capacity. Data augmentation is a established way to generate more training data by slightly modifying the initial data. Data augmentation is a therefore a way to improve the generalization capacity of a model. New data points are automatically generated from the known data points for the AI model, which have the image's core content but are slightly different from the original image. Data augmentation can be carried out in a variety of ways. However, the most common and widely used methods for data augmentation in CV are geometric transformations, like rotating, flipping or cropping the image. These are often very easy to implement, not particularly computationally intensive, and, above all, considered safe in most cases.[SK19, cp. p.1, 7–9] Data augmentation safety reffers to the ability to preserve the label after the transformation. For example, rotating a 6 by 180 degrees int save, because it can be seen as a 9. [SK19, cp. p.7]

In the case study instance, all images from the ImageNette dataset are flipped once horizontally and once mirrored vertically, as shown in figure 3.2 and saved as new images so that the dataset increased size to around 40,000 images.

Figure 3.2: Image Augmentation

Data Prepocessing

The next step before the AI models can work with the images is data preprocessing.[Min+21, cp. p.1] In the case of images from the ImageNette dataset, they are all different sizes. For this purpose, all images are loaded from the hard disk and first transformed into a uniform size of 320 by 320 pixels, which fits the requirements of the later used AI models. Then the images are all normalised and converted into a tensor, which the two AI models can later work with. The procedure for this is shown in figure 3.3.

Figure 3.3: Image Preprocessing

The normalisation of the images in the data preprocessing is essential for the later AI model since all input pixels have a uniform data distribution. This speeds up convergence in the training process.[B17, cp.]

A tensor is a data type, that can be seen as a generalised form of multidimensional arrays that stores all pixel and color information of an image.[Ji+19, cp. p.2] Each PyTorch model expects the data in a tensor to be able to work with it effectively.

Dataset Split

Before training the model, a train, validation and test split is conducted. The train set is used to train the actual model and therefore makes up the biggest part. After each training epoch, the model is evaluated on the validation set, to see how well the model performs on unseen data and whether it has improved. Test, or hold-out, is withheld from the model throughout the training process. Therefore, this dataset is entirely unknown to the model and is used to evaluate the model after the training is completed. How exactly the dataset is divided is not specified and depends on the size of the dataset.[RS15, cp. p.1-2]

For the dataset of 40,000 images used here, the following split (figure 3.4) was applied, which at the same time resembles the percentages commonly used in the industry.[RBH21, cp. p.5] This means training with 32,000 images for the dataset, validating this with 4,000 images and testing with another 4,000 images at the end.

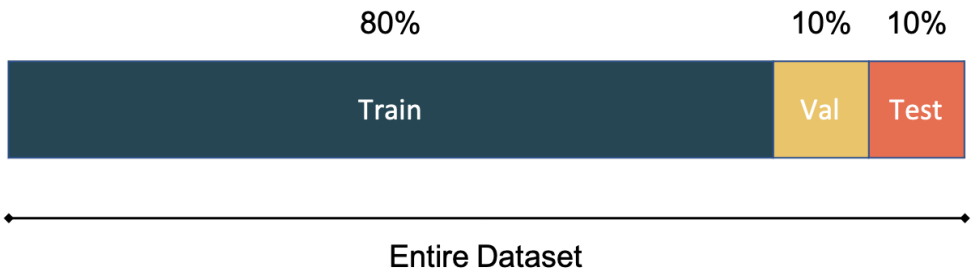


Figure 3.4: Dataset Splitting - Percentages

To ensure that all classes are equally represented in all datasets, the dataset distribution was plotted across the individual classes with the three datasets Train, Val and Test in figure 3.5. It is easy to see that all ten classes have approximately equal shares in all datasets. However, class 3 (chainsaw) has considerably fewer shares in the entire dataset. This is since images of chainsaws were already slightly underrepresented in the ImageNette

dataset. However, the difference to the other classes is not so big overall that an influence on the performance of the models is to be expected.

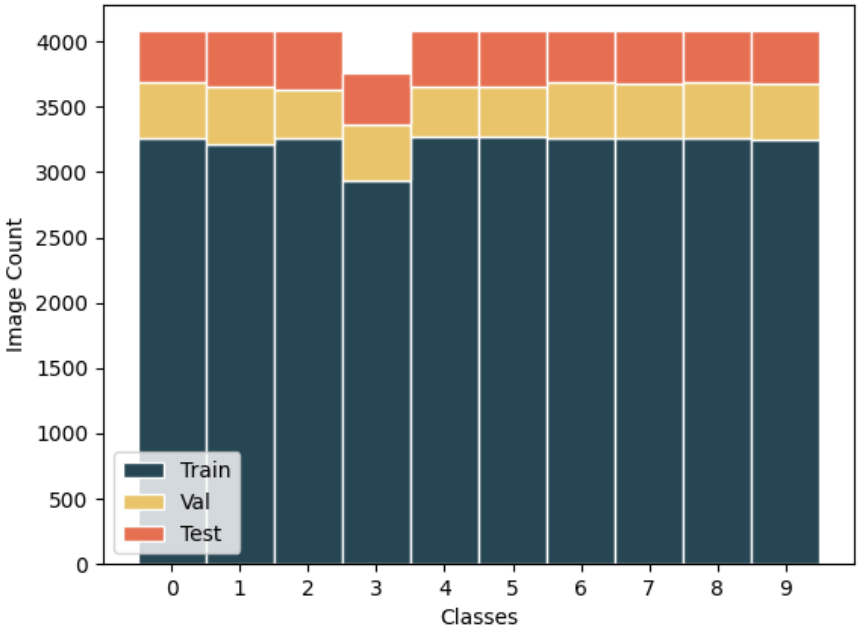


Figure 3.5: Dataset Splitting - Data Distribution

Seeding

The individual images in the dataset are randomly shuffled once before splitting. However, to ensure reproducibility of the results, a so-called seeding is used. This seeding ensures that the same results are continuously generated, regardless of the environment or random number generators.[Doc21b, cp.]

3.3.3 Model Architecture

The compared models show a differing architecture. The model with integrated attention follows the architecture outlined in Google's paper.[Dos+21, cp.] The comparative model for consists of a classic Convolutional Neural Network (CNN) architecture. In the following, both are briefly taken up again and explained in their mode of operation. Both architectures are structured differently, but in order to ensure comparability between the two models and the resulting explanations, the following section also takes actions that put the two models back into the same perspective and thus make them more comparable.

Convolutional Neural Network

CNN is a deep learning model for processing data with a grid pattern, such as pictures, meant to learn spatial hierarchies of features, from low- to high-level patterns, automatically and adaptively. CNN consists typically of three different parts, called layers: convolutional, pooling and fully connected layer.[Yam+18, cp. p.2]

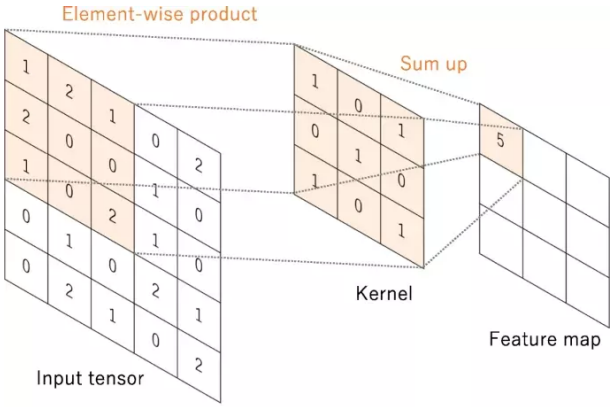


Figure 3.6: Convolutional Layer[Yam+18]

The convolutional Layer is there to find individual features on the image but independent of the location. For example, the face of a dog is never in the same place on the image. For this purpose, a matrix with weights, called kernel, is created in training, which wanders over the image and brings together the individual features, which is shown in figure 3.6.[Ind+18, cp. p.3] The goal of pooling layers is to gradually reduce the dimensionality of the representation, lowering the number of parameters and the model's computational complexity, to improve the overall performance.[ON15, cp. p.8] For example, with Max Pooling, an H x W window goes over the image and thus only takes the maximum value into the new representation, as shown in figure 3.7.

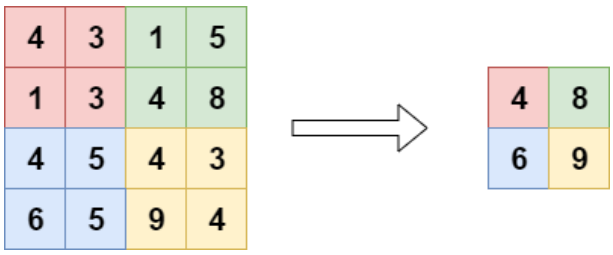


Figure 3.7: Max Pooling Layer[Ver20]

The Fully Connected (FC) Layer is similar to a conventional neural network, where each node is connected to every node from the next layer. Thus, the number of nodes in the

output of the Fully Connected Layer corresponds to the number of classes.[AMA17, cp. p.5] In addition, dropout layers can be built in, which are set to 0.5 as a regulation technique in training random activations to prevent overfitting.[Yam+18, cp. p.9] Furthermore, activation functions as Rectengular Linear Unit (ReLU) or sigmoid can be added to the model. The activation function is a decision-making function that aids in the learning of complex patterns.[Kha+20, cp. p.10]

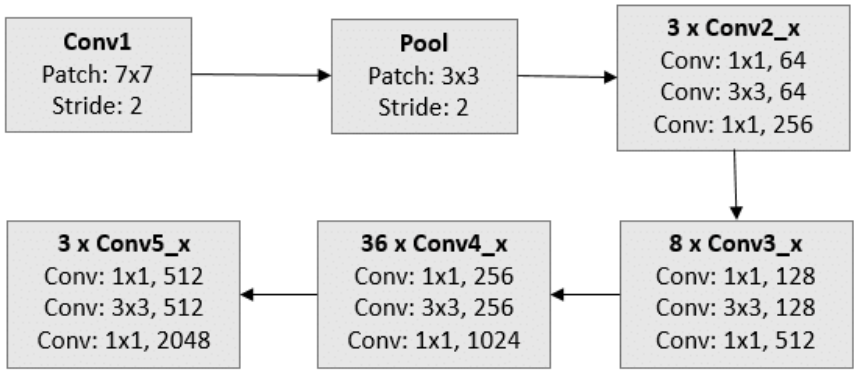


Figure 3.8: RetNet152 Architecture[Ngu+18]

As the reference architecture for this case study, an already pre-trained CNN model is chosen. It has already shown the desired performance in classification tasks for the huge ImagenNet dataset (not to be confused with ImageNette dataset) and thus made it no longer necessary to develop and train an individual model. The AI model behind this architecture corresponds to Residual Network (ResNet)152, which is a deep CNN.[Tea21, cp.] This ResNet152 is made of 152 primarily convolutional layers, as shown in figure 3.8. In addition, the problem of exploding gradients on conventional CNNs is prevented by the ability to skip individual layers.[He+15, cp. p.2, 4] ResNet152 has been trained with 224x244 images, but it is possible to use the larger images from this case study. In addition, a so-called adaptive average pool layer at the end ensures that no matter what dimensions the input had, it is made suitable for the FC layer. Only the FC layer of the pretrained ResNet was adapted to the ten classes of the dataset. And training with the data from the dataset was applied to the adjusted but pre-trained model.

Vision Transformer

The architecture for the ViT, which Google proposed[Dos+21, cp.], is based on the traditional transformer architecture, which was first proposed in the paper "Attention is all you need" by Vaswani et al. in 2017[Vas+17, cp.]. The transformer encoder-decoder was described in detail in chapter 2.1.5.

However, working with images is different from working with text. To make transformers work for computer vision tasks, the images are divided into smaller square patches. These are then placed one after the other to form a sequence. Each image patch functions like a word in the NLP transformer. The sequence corresponds to a sentence in this analogy.

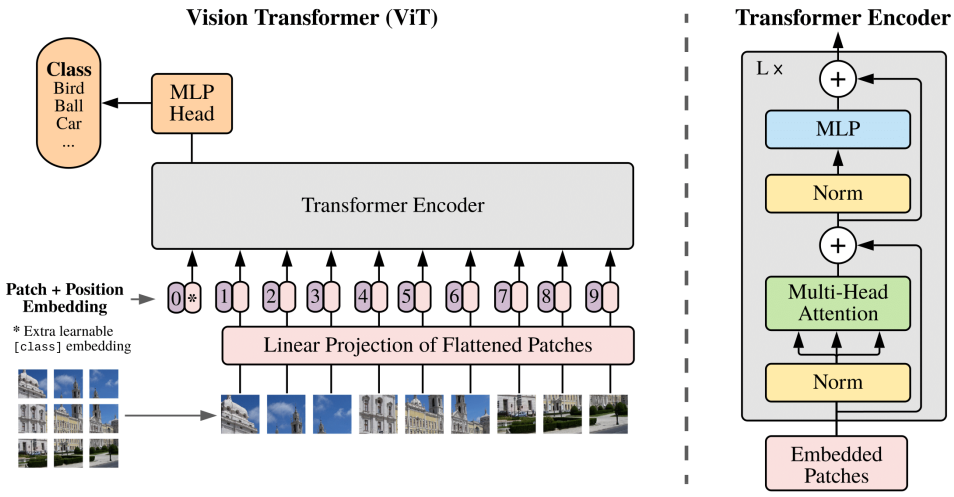


Figure 3.9: Vision Transformer Architecture Overview[Dos+21]

The patches, encoded with a position, are then put back into individual transformer encoder blocks, which are again similar to their models from the transformer encoder-decoder architecture. As in the NLP transformer, the data is first normalised, and a multi-head attention is calculated based on this. The resulting attention vectors are normalised again and fed into a neural network, which takes over the processing and adaptation of the input for the next step, titled Multi Layer Perceptron (MLP), and corresponds to a conventional Neural Network (NN). The outputs from all transformer encoder blocks are again fed into a comprehensive MLP, which outputs the probabilities for the individual classes with a softmax function.

The ViT architecture, as shown in figure 3.9, was not completely rewritten in this case study but uses an existing module that implements the architecture proposed by Google and makes it available as a trainable model. The implementation originates from the GitHub user lucidrains.[luc21, cp.]

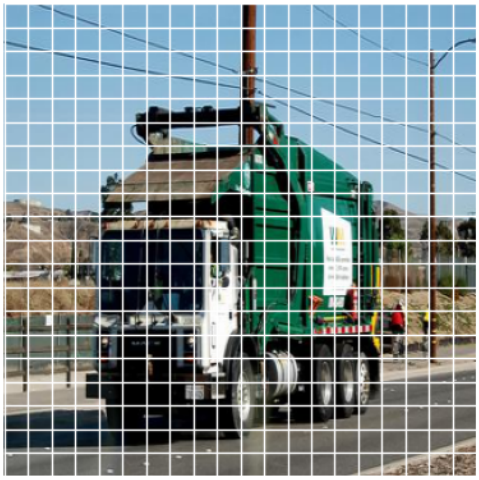


Figure 3.10: Patches on Image

For this case study, the images of size 320x320 pixels were divided into 20 patches on the vertical and horizontal axis, corresponding to the size 16x16 pixels, as illustrated in figure 3.10. These were then fed into the transformer model encoded with a position. Thus, each transformer encoder block calculates a multi-head attention with 16 different heads.

The model is trained together with the images and the corresponding labels.

3.3.4 Model Perfomance

In order to be able to compare the two models, their fundamental performance values are to be measured and compared. This also prevents a model from providing a good explanation and performing poorly in its actual task. Nevertheless, both points are essential in a real scenario.

To measure the general performance of the two models, several generally known metrics for classification tasks are considered.

Accuracy

Accuracy is the most commonly used metric for classification models. Accuracy is a metric that indicates how well the model predicts the complete collection of data correctly. The single individual in the dataset is the metric's basic element: each unit has the same weight and contributes equally to the accuracy value.[GBV20, cp. p.3]

The following formula calculates accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Where in a binary case of i.e. possibly sick patients:

- True Positive (TP) - Prediction is: ill and patient is: ill
- True Negative (TN) - Prediction is: healthy and patient is: healthy
- False Positive (FP) - Prediction is: ill and patient is: healthy (Type I Error)
- False Negative (FN) - Prediction is: healthy and patient is: ill (Type II Error)

In the case of multiple classes, the accuracy is calculated from the confusion matrix by dividing all correctly classified data by the total number of all data.

The calculated accuracy ranges between 0 and 1, where 1 is the best value, which means that all examples from the dataset were classified correctly.

F1 Score

The F1 score also indicates the performance of a classification model. This score is a weighted average between precision and recall and ranges between 0 and 1, whereby 1 is the best value, which describes that the model behaves optimally, and 0 is the worst value.[GBV20, cp. p.4]

The precision is calculated by dividing the number of True Positive elements by the total number of positively predicted units. Thus, precision refers to the percentage of units that the model predicts will be positive and are positive. In other words, precision indicates how certain users can be in the model's prediction of a Positive individual.[GBV20, cp. p.2]

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

The fraction of True Positive elements divided by the total number of positively classified units is known as the recall. The recall is a metric that assesses a model's predictive accuracy for the positive class: it measures the model's ability to discover all Positive units in the dataset.[GBV20, cp. p.3]

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

Finally, the F1 score is calculated with the two values precision and recall.[GBV20, cp. p.4]

$$F1Score = 2 * \left(\frac{Precision * Recall}{Precision + Recall} \right) \tag{3.4}$$

Again, the F1 score is intended for binary classification. In order to make it usable for several classes, the values for precision and recall of the individual classes are combined either in a macro or micro F1 score. In the micro F1 score, all TP, TN and FP of all classes are added up, and the F1 score is calculated. The macro F1 score is the average of all F1 scores of the individual classes.[GBV20, cp. p.7]

$$MacroF1Score = 2 * \left(\frac{AvgPrecision * AvgRecall}{AvgPrecision + AvgRecall} \right) \tag{3.5}$$

The macro F1 score is used to evaluate the models from the case study, as the classes are all equally distributed.

3.3.5 Explainability Methods

In the following subchapter, the methods used to explain the CNN and the ViT model are presented. The method used for the CNN corresponds to the reference method in this case study. This method is also used in many AI projects for explainability. On the other hand, the method for the explanation of the ViT model is new and was developed in the course of this bachelor thesis, this method thereby adapts already known attention concepts from NLP.

CNN

For the explainability of the CNN, the LIME library is used in this case study. Its basic functionality has already been explained in chapter 2.2.4. The ImageExplainer by LIME is used precisely, and it takes the original image and 1000 permutations of it and finds out by the classification results of the CNN which area was most important for the correct classification of the original image. Based on its findings, the explainer returns a bitmap of the size of the original image. In this map, the "important" places are marked with a 1, the unimportant ones with a 0.

The bitmap is placed on the original image for visualisation, and the critical areas are marked with a yellow border.

Figure 3.11: Lime BitMap and Visualisation

In this figure 3.11, which belongs to the class of churches, the area around the church tower has been identified as important. Nevertheless, parts of the roof and the background also influenced the image classification result.

For the following metrics on the quality of the explainer, the bitmap itself is used.

ViT

As noted initially, the use of attention to explain classification results is new in the field of computer vision. For this reason, this bachelor thesis proposes a novel way to use attention weights for an explanation, which is adapted from the way of dealing with attention weights from the NLP field. The used architecture of the ViT model can return the classification result for an image and all attention weights that were calculated during the classification.

$$attention\_tensor.shape\ [(1, 6, 16, 401, 401)] \tag{3.6}$$

These attention weights are available in a multidimensional tensor. However, this is not immediately comprehensible. The tensor consists of the dimensions of the batch size, the layers, the attention heads, and a self-attention matrix. This tensor is transformed in several steps into a human-readable structure. The batch size is first removed, as this always equals 1 in the architecture. The second step is to average overall six layers and all 16 attention heads of the multi-head attention. After that, a 401 x 401 matrix with the average attention weights remains.

The matrix in figure 3.12 corresponds to the matrix from calculating the self-attention in NLP, except that this time it is not the embedded words of the sentence at the edges but

the embedded and linearly strung patches of the image. The image is divided into 20 x 20 patches, as it is also defined in the basics of the case study, in 3.3.3. This results in 400 linear patches plus one patch, which, according to the description, only takes care of positional encoding. However, this is removed for further consideration.

x	Patch 1	Patch 2	Patch 3	Patch 4	Patch 5	Patch 6	Patch 7	...	Patch 400
$Patch_1$								...	
$Patch_2$								...	
$Patch_3$								...	
$Patch_4$								...	
$Patch_5$								...	
$Patch_6$								...	
$Patch_7$								...	
...	...	...	...	...	...	...	...	...	
$Patch_{400}$									

Sum up

x	Patch 1	Patch 2	Patch 3	Patch 4	Patch 5	Patch 6	Patch 7	...	Patch 400
Atten-tion								...	

Figure 3.12: Self-attention Matrix

To calculate the attention for each patch, the attention weights are added up column by column, which creates a one-dimensional 400 element array, with a single attention weight for every image patch.

Using the calculated attention, the weights are converted back into a 2-dimensional array in the shape and size of the image. As a final step, all weights are normalised between 0 and 1. An implementation of this algorithm has been documented under: Appendix: 1.

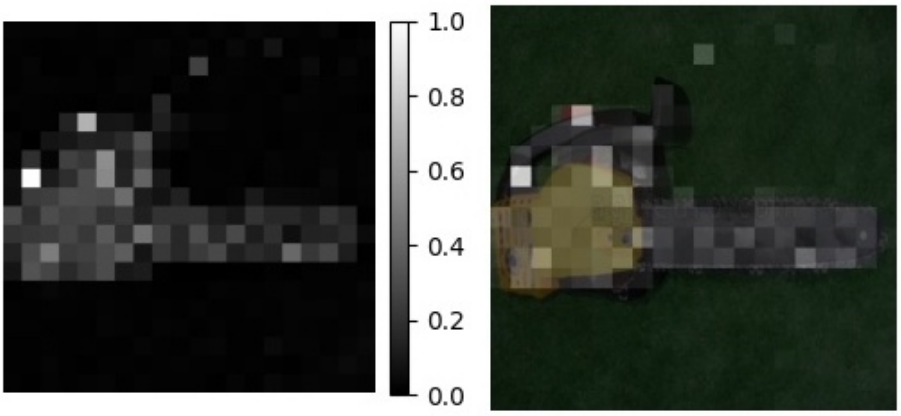
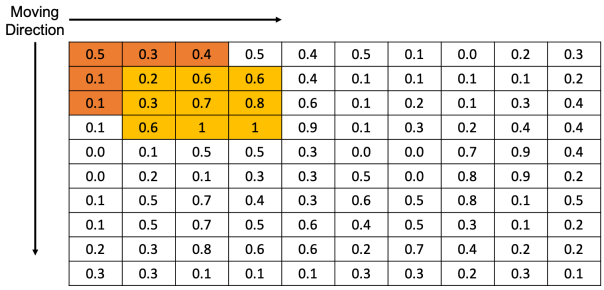


Figure 3.13: Attention Matrix and Visualisation

With the help of the representation in figure 3.13, it is easy for a human to visualise which parts of the image the attention of the model lay during the classification. However, the calculated decimal attention weights in the attention matrix and visualisation for each patch are not so easy to work with, like the bitmap of the LIME explanation.

Therefore, in a second step, a bitmap is created from this 2-dimensional array with floating-point values between 0 and 1. For this, however, the challenge wad to be solved as to which parts of the image were ultimately most important for the classification and may have contributed only partially.

The idea of defining a threshold value to distinguish whether a patch with certain attention was essential or not was quickly discarded. Since it is not possible to logically justify how the threshold value should be set. A second approach is image segmentation, using the Felzsenszwalb method to separate the areas with high attention from the areas with low attention[FH04, cp.], based on contrast. However, this method does not yield very satisfactory results to visualise on which parts the most attention lies, because the marked areas did not correspond to those that have a high attention. Similarly, an algorithm was tested to find the contours of the crucial parts of the array, which is based on a 2d version of the marching cubes algorithm.[LC87, cp. p.164]



0.5	0.3	0.4	0.5	0.4	0.5	0.1	0.0	0.2	0.3
0.1	0.2	0.6	0.6	0.4	0.1	0.1	0.1	0.1	0.2
0.1	0.3	0.7	0.8	0.6	0.1	0.2	0.1	0.3	0.4
0.1	0.6	1	1	0.9	0.1	0.3	0.2	0.4	0.4
0.0	0.1	0.5	0.5	0.3	0.0	0.0	0.7	0.9	0.4
0.0	0.2	0.1	0.3	0.3	0.5	0.0	0.8	0.9	0.2
0.1	0.5	0.7	0.4	0.3	0.6	0.5	0.8	0.1	0.5
0.1	0.5	0.7	0.5	0.6	0.4	0.5	0.3	0.1	0.2
0.2	0.3	0.8	0.6	0.6	0.2	0.7	0.4	0.2	0.2
0.3	0.3	0.1	0.1	0.1	0.3	0.3	0.2	0.3	0.1

Figure 3.14: Sliding Window Algorithm

In the end, a sliding window algorithm was chosen, which deliveres the desired results of a bitmap, showing important and unimportant areas of an image. A certain size window is gradually slid over the entire attention array in this algorithm, and the average attention in the window is calculated, as shown in figure 3.14. After a complete run, the area with the highest average attention is marked on the output bitmap and set to zero in the attention array. This means that the area with the highest attention is not found again in a further run but with the second-highest. If all areas on the bitmap are marked, the algorithm excludes them and returns them. This bitmap can then be used for the calculations of the metrics, which are used in the evaluation of the quality of the explainer. A corresponding bitmap for the chainsaw example could look like the bitmap in figure 3.15.

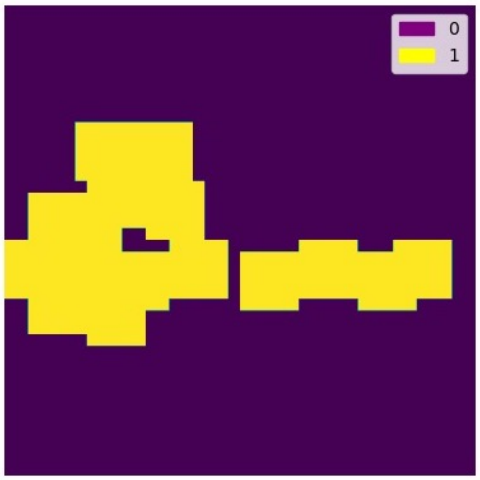


Figure 3.15: Corresponding BitMap Chainsaw Example

The size of the window and the number of areas to be included in the bitmap can vary and were tested in a test phase. In the test phase, the combination of a 40x40 window and the top 15 windows produced the best looking results, which marks the areas with the highest attention.

3.3.6 Benchmarks of Explainability

The quality of explainers can be measured in a variety of ways. Since an explainer is also always closely linked to the users, human studies are often conducted for this purpose. For example, in a crowdsourcing approach by Hutton et al., where explanations created by humans or by an algorithm are compared and evaluated by humans.[HLM12, cp. p.22] However, these tests always depend on the subjectivity of humans. Nevertheless, explainers can also be compared quantitatively using various metrics. The metrics of the quantitative comparison on the quality of the two explainers are based on the paper of Santhanam et al.[San+20]. Consistency, correctness and confidence of the two explainers are measured.

It should be noted that the contents of the paper are the first in the field to propose computationally inexpensive metrics that can measure the quality of an explainer. The paper has been under review by other researchers since its publication in 2020. The explanations and backgrounds to the individual metrics are logical and comprehensible so that nothing hinders their use.

Key assumptions

The paper states the following as prerequisites, which are essential as a basis for the following metrics. First, the classifier must have an overall good general performance on a test set of images. Furthermore, the classifier should perform similarly well on semantically invariant transformed images, such as flipped or mirrored images. Those prerequisites ensure that the quality of the explainer is not affected by a reduced functioning classifier.[San+20, cp. p.2-3]

The quality of the two classifiers is covered by the metrics mentioned above. In addition, the ability to classify transformed images well should be covered by the training with the augmented images and the abstraction ability of the two models.

Consistency

In the paper from Santhanam et al., consistency is defined "as the ability of the explainer to capture the same relevant components under various transformations to the input."[San+20, cp. p.5] Thus, the consistency metric compares whether the generated explanation for the altered input (following an inverse transform) is similar to the original input. In the case of this case study, the bitmaps generated by the explainers are compared. For this purpose, one image and its horizontally mirrored counterpart are explained by the explainer.

In the case study, the consistency is calculated based on two bitmaps according to the following algorithm, shown in figure 3.16:

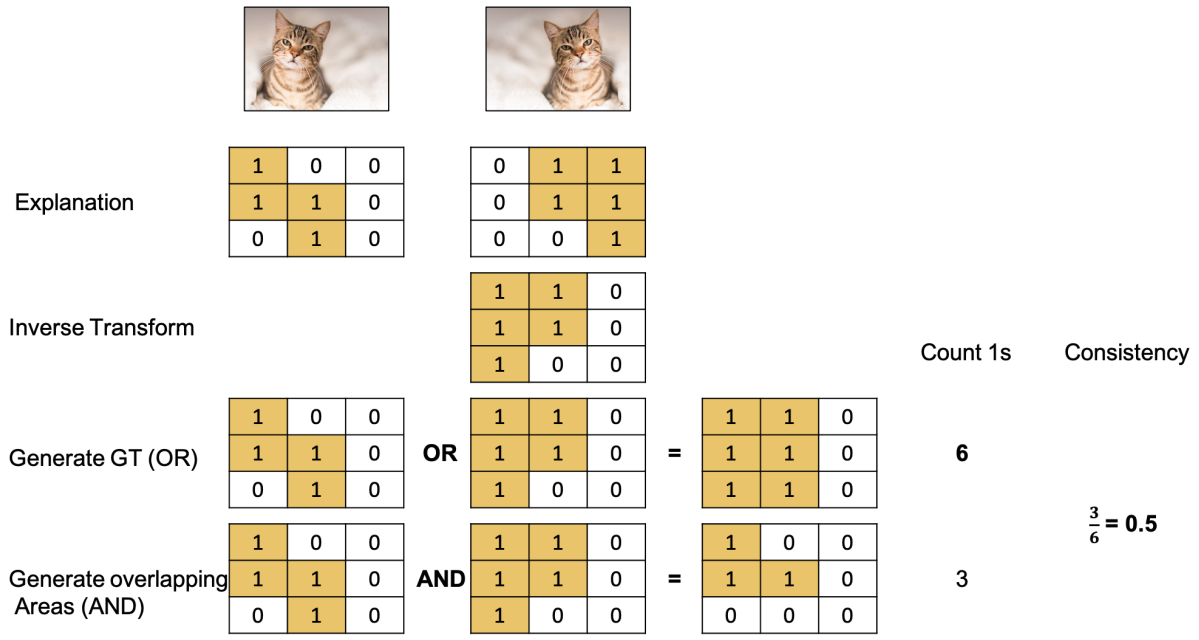


Figure 3.16: Consistency Calculation Algorithm Example

First, the original and the mirrored image are explained by the instructor. Then, the explanation of the mirrored image is mirrored back. Next, Ground Truth (GT) is calculated from both bitmaps. The GT corresponds to the whole two areas marked by the explainer and is mapped via a logical OR operation. The next step is to calculate the area that matches both explanations. This area is mapped with a logical AND. The value for consistency is the area where both explanations match, divided by the total marked area of the explanations.

If the areas of both explanations are the same, the consistency is 1. If completely different areas are marked as important in both explanations, the consistency is zero.

Finally, an average is calculated over all consistency values from the dataset.

In this case study, the consistency is calculated both for the LIME explanationon of the CNN model and the proposed attention-based explainer using ten images from each class, which equates to 100 images.

Correctness

The paper defines correctness or also called fidelity or sensitivity, as an explainer's ability to accurately identify input features that contribute the most to the classifier's prediction.[San+20, p.4] This metric further describes that the explainer identifies important parts of the image and dismisses unimportant parts.

This calculation is done by masking two images according to the confidence of the model. According to the theory, the important pixels from an image with high confidence should be used on an image with low confidence, and a new classification by the model should lead to higher confidence. Overall, the accuracy of the low confidence set is then compared to the set with the masked images. The masked accuracy should be higher for a good explainer.

For this purpose, a pool of 1000 random images per class is created. The model in the following step classifies every image, and the corresponding class's confidence is saved. The images of each class are sorted according to the confidence so that the pairs of an image with a very high and an image with very low confidence can be selected. The fact that the pool consists of a huge number of images per class prevents the phenomenon from the paper that images with very high (e.g. 99%) and high (e.g. 70%) confidence are paired.[San+20, cp.] This happens quickly if the pool is too small.

An explanation and a corresponding bitmap are generated for the image with high confidence. Then, the "important" pixels from the high confidence image are placed on the low confidence image. Finally, the masked image is classified again by the model.

In the case of the correctness metric, the accuracy of the images with low confidence is compared with the accuracy of the masked images. An visual exaple of the masking algorithm can be found within figure .1 in the appendix.

For this purpose, five pairs are formed from each class, so the correctness is calculated from 100 images.

Confidence

Confidence is about whether individual explanations achieve higher confidence within a masked image. Explainers that generate explanations with high confidence predictions are preferred over those that do not, according to the paper.[San+20, p.6] Again, the important pixels are extracted from images where the model was very confident in the prediction. These important pixels are again masked on images where the model was not quite sure. The same algorithm is used for masking as for correctness. However, this time, not the overall accuracy is measured, but the individual increases in confidence that the model outputs.

Also, when calculating the confidence, five pairs are formed from each of the ten classes.

4 Results and Evaluation

In the following chapter, the measured performance and the measured values of the individual benchmarks are presented and evaluated. In addition, further points of the attention-based explanation are addressed in a discussion.

4.1 Results

The models defined in the previous sub-chapter and the metrics for the performance of the models and the quality of their explanations are measured in the sections below.

4.1.1 Model Perfomance

The general performance of the two models is evaluated with the 4000 images from the test set from the dataset.

AI Model	**Accuracy**
CNN	96.42%
ViT	87.63%

Table 4.1: Accuracy Results

AI Model	**Macro F1**
CNN	96.41%
ViT	87.61%

Table 4.2: Macro F1 Score Results

4.1.2 Quality of the Explanations

The calculation of the three metrics for the quality of the explainers with the defined number of images produced the following results.

Consistency

AI Model	Consistency
CNN	51.28%
ViT	46.45%

Table 4.3: Consistency Results

Correctness

AI Model	Acc. Low Images	Acc. Masked Images	Δ Acc.
CNN	30%	62%	+32%
ViT	4%	16%	+12%

Table 4.4: Correctness Results

Confidence

AI Model	∅ Prob. Low Images	∅ Prob. Masked Images	∅Δ Prob.
CNN	9.34%	48.18%	+38.84%
ViT	0.01%	14.17%	+14,16%

Table 4.5: Confidence Results

4.1.3 Further Investigations

In addition to the calculated metrics, some experiments were conducted to better understand the overall calculated attention weights.

Initially, it was investigated which image areas were most frequently the subject of high attention. For this purpose, the average of all attention maps of all images from the dataset was formed and visualised using the usual methods.

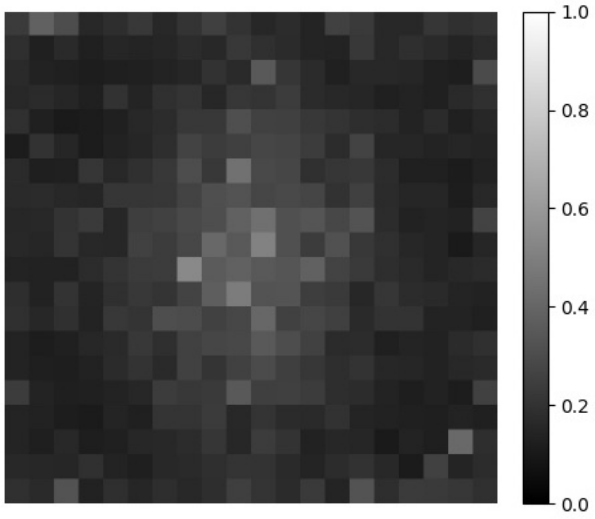


Figure 4.1: Average Attention Weight over all Images

In addition, tests were performed to see how the ViT model and the calculated attention weights behave with only black or white images. This means that they only consist of zeros (for black) and ones (for white). In order to avoid a lucky hit, the average of ten black and ten white images was calculated.

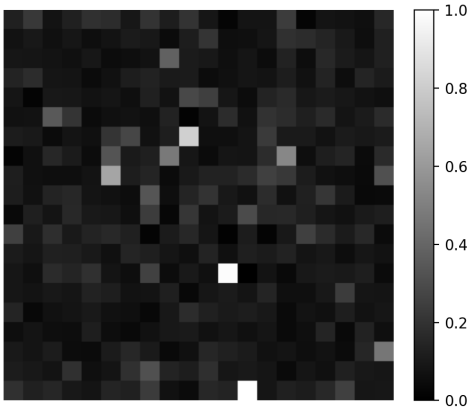


Figure 4.2: Attention for a white Image

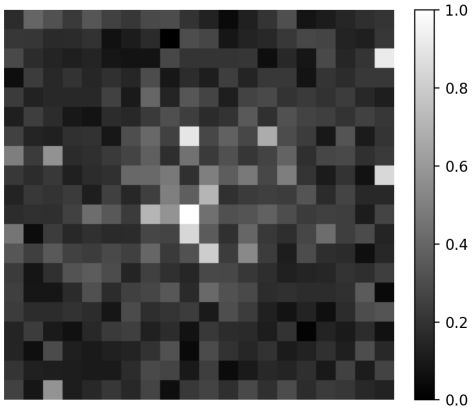


Figure 4.3: Attention for a black Image

4.2 Evaluation of the Results

Calculated numbers only give an overview of how good one explainer is compared to another, but these still need to be evaluated to make an accurate statement about the quality of an explainer.

Requirements and Performance

For this purpose, it must first be determined whether the two models and their explainers meet the basic requirements of the defined metrics. In order to be able to apply the metrics at all, the tested models must generally have good performance and be able to classify transformed images well. The first point is tested in the following. The second point was ensured by training with the augmented images. However, these tests always depend on the subjectivity of humans. Nevertheless, explainers can also be compared quantitatively using various metrics. Looking at the general performance of the two models in table 4.1 and 4.2, both perform very well on the unseen test set of images. The pre-trained CNN stands out with around 96% accuracy, although the around 87% accuracy of the ViT is also very impressive. This is especially true because no pre-trained model was used here, and the current model was trained with only a few epochs. The confusion matrices also show that the two models work about equally well on all ten classes. So the first requirement is met. The second requirement that the models work just as well on transformed images is given by the training with the augmented images.

The general performance values for the two models are also similar or so close that the two models can be compared with the calculated metrics. No model is significantly worse than the other.

However, what is remarkable is the strong similarity of accuracy and F1 score in the two models, which only differ minimally. This speciality is caused by the equal distribution of the classes as well as the calculation of the macro F1 score through the averages. If the dataset were uneven, the accuracy would differ more from the F1 score.

Consistency

Looking at the consistency results in table 4.3, the explanation for the image and the mirrored image overlap by about 50% for both explainers. The explanation of LIME overlaps on average 6% more than the explanation method of the ViT proposed in this bachelor thesis. This difference could be caused by the slightly worse general performance of the ViT or by the parameters in the sliding window algorithm. For example, a fine-tuning of the number and size of the windows could improve the results. However, this slight difference is rather marginal in the overall picture.

Correctness

However, it gets more interesting with the two metrics that use masked images to check the extent to which explainers select important pixels.

In terms of correctness, it can be seen in table 4.4 that, as expected, the poorly recognised images are also recognised with very low accuracy. Here, the generally better CNN model is still represented with an accuracy of a good 30%. The ViT model recognises the worst images in comparison with an accuracy of only 4%. Together with the important pixels from the well-recognised images, the images were masked and classified again by the model. The paper about the metrics, describes the expectation that this would increase the accuracy of these images, which generally occurred.[San+20, cp.] The accuracy of the CNN increased by 32%, that of the ViT by only 12%. At first glance, it could be concluded that the explanation and selection of the important pixels of LIME is better because the increase in accuracy was much higher. However, the proportional increase should also be taken into account too. This shows that the accuracy in the explanation of the ViT improved by a factor of 4 and that of CNN and LIME only by 2. However, it is certainly more important to have generally better accuracy than a good proportional increase in real applications. An increase by a factor of 10 could also be achieved very easily from accuracy of 0.1% to 1%. An increase in accuracy in this case means that the explainer generally has marked the important areas in the images with the high confidence.

Confidence

A similar picture emerges when looking at the confidence metric in table 4.5. Here too, the probabilities that the models assign to the images are very low. As the pairs of masked images increase, so does the average probability assigned. Here, too, it can be observed that the average increase for the CNN with the explanation by LIME is higher than that of the ViT explanation. Again, the absolute increase is more important than the proportional increase. An individual increase in confidence means that the explainer can mark the important areas on the individual pictures.

In summary, the three calculated metrics show that the already established explanation of the CNN model by LIME is slightly better than the method proposed here to use the attention weights of a ViT for explainability. However, the values of the explanation for the ViT are not completely outside and still behave as described in the paper, by Santhanam et. al, for a good explainer.[San+20, cp.]

4.2.1 Critical Review of Results and Metrics

Since the metrics for measuring the quality of explainers are very fresh and still being reviewed by the public, they are explicitly emphasised here.

In general, the metrics applied have shown that they are based on logical principles and are useful in the quantitative measurement of the quality of explainers. The measured values for both the already established explainer LIME and the attention-based explainer show that they can be applied, even when applied to a new dataset that has not been tested before. However, in the following, the meaningfulness of the metrics will be examined more closely, and an attempt will be made to justify the more unsatisfactory performance of the attention-based explainer.

One reason for the more unsatisfactory performance of the attention-based explainer for the ViT model in the two metrics that use masking could be how the essential areas are selected.

In LIME, the 1000 iterations with different parts of the image are used to determine which part contributes the most to a correct classification and thus find out the important ones. So it is no problem to transfer the important pixels to another image and expect a better classification result.

With attention-based ViT, it works a little bit different. The marked pixels by the explanation are those on which the highest attention is located, making them the most important. However, they are only the most important parts in the context of the whole image and all patches. The calculated self-attention, therefore, depends on all parts of the image. Now it is possible to mask the important pixels from image A to image B without any problem, but the full context for the important pixels from image A is lost in this process. Therefore, a new self-attention is computed on the masked image B, which includes the important pixels from A but in a different context than on the original image A. The new context can cause the previously important pixels to lose their importance.

This is like choosing the person from France who speaks the national language French the best and then bringing this person to Germany and expecting that this person is still the best person there who speaks the (new) national language. However, the context, the national language, is lost on the way, so the person will not be the person who speaks the best german in Germany.

Nevertheless, this is not so much a problem of the explainer because it still finds out and marks the important pixels for the classification on the image. It is more a problem in the metric because it does not include the context, which is unfavourable for a context-dependent explainer.

The issue raised in the paper that images can be matched even though they may both have higher confidence is based on the size of the pool from which the images can be selected and the model's performance. For smaller randomly selected image pools, the likelihood is high that there will be fewer low confidence images than pairs to be formed. In this case study, 100 times more images were put into the pool than were needed for the pairs. Consequently, only images with very high and very low confidence were matched. Should the model be perfect, i.e. recognise all images correctly and with high confidence, then the metrics cannot be applied, as this means that no pairs can be formed. However, this case is rather to be classified as unrealistic since there are hardly any perfect models. Even with the very good CNN from this case study, it was still possible to find good pairs.

Finally, there is the assumption that the lower accuracy of the ViT also contributes to the poorer performance of the attention-based explainer. In the end, the poorer performance of the attention-based explainer cannot be attributed to the slightly poorer accuracy, but it also cannot be proven that this was not the reason. This is definitely a point that needs to be looked at more deeply, but it is a separate consideration that could certainly fill a separate bachelor thesis.

In summary, the metrics used are appropriate to measure the quality of explainers on a quantitative level. It is good that the metrics cover all sides of a good explainer, like finding the most important areas in a picture and that the explanations are consistent even if the picture changes. When implementing, a sufficiently large pool of images should be considered, as shown in the previous paragraph. However, these metrics are more for post-hoc explainers like LIME or SHAP, which are not contextual like the proposed attention-based explainer.

4.3 Further Discussion of Attention

A concentration of the high attention weights in the centre of the attention map is visible, in figure 4.1. The ViT model places a higher average value on the parts of the picture in the centre than on the parts at the edge.

This bias can probably be explained by the images from the dataset. On these, the essential parts are usually shown in the centre, so that the ViT has apparently learned that these are the areas to which it must pay more attention. The disadvantage of this behaviour is that images in which the object to be recognised is not in the centre but instead at the edge are probably recognised more unsuccessfully. In a production environment with authentic images, however, a central alignment cannot always be ensured.

To prevent this phenomenon, another step could be added in data preprocessing. PyTorch offers a stock function that crops the original image into five images with different centring, as shown in figure 4.4. This cropping moves the object of interest to other corners, perhaps leading to a better overall distribution of attention weights.

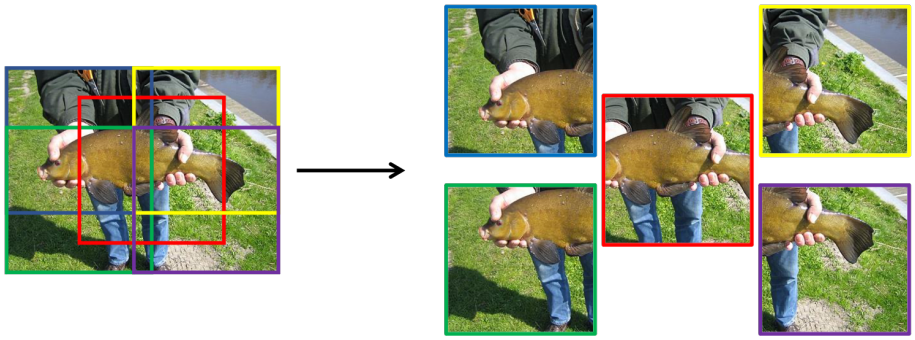


Figure 4.4: Five Crop Example

When investigating results of figure 4.2 and 4.3, one would expect an equal distribution of all attention weights over the entire image or, as shown in the previous example, a concentration of the higher weights in the centre of the image. However, very different attention weights were generated for the white and the black image. With the white image, no pattern or similarity is immediately recognisable. There are isolated areas in which the attention is very high, while the remaining parts of the image have very low attention or almost none at all. The attention weights for the black image behave similarly to the previously calculated overall average of all images. The high attention weights are in the centre of the image. The reason for this could be a predominantly dark dataset in training, which would indicate that the black images tend to behave like the general average of the calculated attention. The white images and their calculated attention are out of line because there are fewer bright images in the dataset.

Discussion of the Explainer

Returning to the explainer developed in this paper: As an advantage, this one has a kind of intrinsic explanation that otherwise only exists in very simple models such as decision trees. With LIME or SHAP, different combinations of features have to be classified repeatedly by the model to decide which were the decisive features or areas in the image. Instead, the already calculated attention weights can be used directly to generate an explanation. In this way, the image only passes through the model once. Therefore, it is reasonable to assume that if the model improves, the explainer's quality will automatically improve since the explainer is based directly on the attention weights also used for the model's classification and explanation. On the other hand, one disadvantage of this explainer is

evident on closer inspection. This explainer only looks at the generated attention weights, which make up a biggest part of the model architecture, but for example, the final MLP layer is not considered at all by the explainer.

Additionally, it would still be worth considering trying out to modify some of the hyperparameters and how they affect the performance of the ViT. For example, the ViT used in this case study was first trained with a patch size of 32x32 pixels, which provided good results but was still worse than the ViT used now with patches of 16x16. This patch size was already suggested in the paper from Google [Dos+21, cp. p.1] and showed promising results there and in this case study. However, smaller patches, such as 8x8 or even 4x4, should be investigated since the assumption is that the model can pay attention to even more minor details. Unfortunately, it was not possible to test this due to hardware limitations, as the calculations become more and more computationally intensive with the increase in patches. Other parameters, such as a higher dropout than the default one, could prevent overfitting when training with many epochs and make the ViT model more generalised.

Conclusion of Discussion

In summary, it can be said that the lower performance of the attention-based explainer is probably due to several reasons. It has been shown that the metrics, which are still new, are helpful in the quantitative measurement of the quality of explainers. However, especially the metrics based on masking have a problem with attention, as it is 100% based on context, which is lost when masking. In addition, it can be assumed that the slightly worse performance of the attention-based explainers is also due to the slightly lower accuracy of the ViT. However, this can neither be confirmed nor refuted in this work. On the other hand, the ViT model and the attention-based explainer still have much potential with a little fine-tuning. Thus, the all-important patch size should be tested even more in further investigations; a clear recommendation would be to reduce the size of the patches so that the model can pay attention to even more subtleties. Nevertheless, the explainer still has some potential for improvement, especially in converting the attention matrix into the bitmap using the sliding window algorithm.

5 Conclusion and Future Outlook

In recent years, the many innovations and successes of attention-based AI models in all possible application areas show enormous potential for these technologies in the future. Initially observed in nature by a psychologist, the concept of attention was also adopted for AI models. The basic idea is that the model includes more important things in the decision or prediction and can fade out less important things. Attention was first successfully implemented for translators in 2017 by researchers of Google.[Vas+17, cp.] After this success, attention was implemented in many other NLP tasks and CV tasks more recently.

Explainable AI

At the same time, the general trend is towards the explainability of AI models. This becomes more and more relevant, especially in real application cases with very complex models. Not only users can benefit from good explainability by understanding the decision path of a model and and thereby increase trust in models. But also developers can use deeper insights to prevent incorrect training of a model, detect biases and thus also improve fairness.[GSC21a, cp.] Accordingly, many algorithms and tools have evolved to provide explanations for various models. The most common tools, such as LIME and SHAP, are based on post-hoc methods that interact with the model after the actual decision or prediction has been made and find out which parts of the input have positively or negatively affected the result by trying different feature combinations.

Attention-based Explainer

Using the calculated attention weights for texts from NLP for a visualization suggests that these can also be used in CV for visualisation and a subsequent explanation. This bachelor thesis has proposed a simple way to use the calculated attention weights for a visualization that highlights the areas on which the most attention of the model lay. In a sense, this visualization resembles an explanation of the model and its decisions.

Case Study

The goal of this work was to test whether this newly proposed attention-based explainer is a good explainer. In a real-world setting, an already established explainer LIME on a conventional CNN was contrasted with the novel ViT and its explanation. By using three quantitative metrics calculated on a data set of sample images, the quality was determined. It was observed to what extent the two explainers mark the most important areas for the classification and whether these are the same, even if the images are mirrored, for example.

In general, both models performed very well in the classification of ten different classes of images. After training, the pre-trained CNN performed with about 96% accuracy and the ViT without much finetuning or hyperparametertuning with about 87% accuracy. Compared to the already established LIME explainer, the explainer proposed in this bachelor thesis performed slightly more unsatisfactory overall. For example, LIME marked approximately 50% of the same relevant areas on mirrored images. On the other hand, the explanations of the ViT only matched to about 46%.A similar pattern was observed for the two metrics, which use masking to find out whether the explainer selected the important parts of the image. The metrics for the two explainers behave as described in the paper.[San+20, cp.] The accuracy and the confidence on the masked images increased, which means that the important pixels were detected. However, a bigger increase can be observed for the explanation of LIME on the CNN.

The reasons for the lower performance of the attention-based explainer can be multifaceted. For this reason, the attention-based explainer was examined more closely and the suitability of the selected metrics for calculating the quality of an explainer, as these are still new in this field.

It was concluded that the metrics used have a logical basis and can quantitatively measure the quality of an explainer from various points of view. For example, the extent to which the explainer selects the important areas in the image and whether this selection is also consistent on transformed images is examined. However, a point of criticism is the metrics based on the masking of certain sections of the image to determine whether the explainer has marked the most important areas. With conventional explainers, such as the LIME used here, this is a logical step to check. However, the highly contextual nature of the attention-based explainer may have caused it to perform worse in the metrics. This is because image areas were always torn out of one context and inserted into another. At the same time, context is super critical for the attention-based explainer. Another reason for the lower score could be the slightly worse general performance of the model used. However, this could neither be disproved nor proven within the scope of this

Bachelor's thesis. Nevertheless, further points emerged during the investigation as to how the attention-based ViT model and the associated explainer could still be improved. On the one hand, one recommendation would be to train the model with additional augmented images from the training set that do not have the object to be classified centred to expand the model's generalisation capability further. On the other hand, the size of the patches,into which the image is divided by the ViT, should be varied in future experiments. Especially with smaller patches, it seems likely that the model can pay attention to even more minor subtleties in the image. However, reducing the size of the patches increases the training time, as more calculations have to be made. In addition, a little fine-tuning of the actual attention-based explainer could also increase the quality.

In summary, the intrinsic attention-based explainer developed in this bachelor thesis is a good explainer with much potential. It combines attention with the explanation of AI models and thus contributes to more comprehensibility and trust in the models.

Bibliography

Books

[BB21] Przemyslaw Biecek and Tomasz Burzykowski. *Explanatory Model Analysis*. Chapman and Hall/CRC, New York, 2021. ISBN: 9780367135591. URL: https://pbiecek.github.io/ema/.

[Mol19] Christoph Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. ebook. Feb. 2019. ISBN: 9780244768522.

[Wei18] Jianing Wei Wei Di Anurag Bhardwaj. *Deep Learning Essentials*. ebook. Packt Publishing, Jan. 2018. ISBN: 9781785880360.

Articles

[AB18] Amina Adadi and Mohammed Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052.

[Bar+20] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2019.12.012. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[Cha+20] Sneha Chaudhari et al. "An Attentive Survey of Attention Models". In: *Journal of the ACM* 37 (Dec. 2020). arXiv: 1904.02874 [cs.LG].

[CPC19] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. "Machine Learning Interpretability: A Survey on Methods and Metrics". In: *Electronics* 8.8 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8080832. URL: https://www.mdpi.com/2079-9292/8/8/832.

[Fer12] D. A. Ferrucci. "Introduction to "This is Watson"". In: *IBM Journal of Research and Development* 56.3.4 (2012), 1:1–1:15. DOI: 10.1147/JRD.2012.2184356.

[FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision* 59.2 (2004), pp. 167–181. DOI: 10.1023/B:VISI.0000022288.19776.77. URL: https://doi.org/10.1023/B:VISI.0000022288.19776.77.

[GA19] David Gunning and David Aha. "DARPA's Explainable Artificial Intelligence (XAI) Program". In: *AI Magazine* 40.2 (June 2019), pp. 44–58. DOI: 10.1609/aimag.v40i2.2850. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/2850.

[GLT20] Andrea Galassi, Marco Lippi, and Paolo Torroni. "Attention in Natural Language Processing". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020), pp. 1–18. ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.3019893. URL: http://dx.doi.org/10.1109/TNNLS.2020.3019893.

[HA21] Adrián Hernández and José M. Amigó. "Attention Mechanisms and Their Applications to Complex Systems". In: *Entropy* 23.3 (2021). ISSN: 1099-4300. DOI: 10.3390/e23030283. URL: https://www.mdpi.com/1099-4300/23/3/283.

[Has+17] Demis Hassabis et al. "Neuroscience-Inspired Artificial Intelligence". In: *Neuron* 95.2 (July 2017), pp. 245–258. DOI: 10.1016/j.neuron.2017.06.011. URL: https://www.cell.com/neuron/fulltext/S0896-6273(17)30509-3.

[HC05] Simon Haykin and Zhe Chen. "The Cocktail Party Problem". In: *Neural computation* 17 (Oct. 2005), pp. 1875–902. DOI: 10.1162/0899766054322964.

[Hol12] Will Holman. "Lessons from the Front Lines of Social Design". In: *Places Journal* 2012 (Jan. 2012). DOI: 10.22269/120116.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "LONG SHORT-TERM MEMORY". In: *Neural Computation* 9.1 (Jan. 1997), pp. 1–42. DOI: 10.1162/neco.1997.9.1.1. URL: https://www.bioinf.jku.at/publications/older/2604.pdf.

[Ind+18] Sakshi Indolia et al. "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach". In: *Procedia Computer Science* 132 (2018). International Conference on Computational Intelligence and Data Science, pp. 679–688. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.05.069. URL: https://www.sciencedirect.com/science/article/pii/S1877050918308019.

[Ji+19] Yuwang Ji et al. "A Survey on Tensor Techniques and Applications in Machine Learning". In: *IEEE Access* PP (Oct. 2019), pp. 1–1. DOI: 10.1109/ACCESS.2019.2949814.

[Kha+20] Asifullah Khan et al. “A survey of the recent architectures of deep convolutional neural networks”. In: *Artificial Intelligence Review* 53.8 (Apr. 2020), pp. 5455–5516. ISSN: 1573-7462. DOI: 10.1007/s10462-020-09825-6. URL: http://dx.doi.org/10.1007/s10462-020-09825-6.

[LC87] William E. Lorensen and Harvey E. Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *ACM SIGGRAPH Computer Graphics* 21.4 (Aug. 1987), pp. 163–169. DOI: 10.1145/37402.37422.

[ON15] Keiron O’Shea and Ryan Nash. “An Introduction to Convolutional Neural Networks”. In: *ArXiv e-prints* (Nov. 2015).

[RBH21] Anita Rácz, Dávid Bajusz, and Károly Héberger. “Effect of Dataset Size and Train/Test Split Ratios in QSAR/QSPR Multiclass Classification”. In: *Molecules* 26.4 (2021). ISSN: 1420-3049. DOI: 10.3390/molecules26041111. URL: https://www.mdpi.com/1420-3049/26/4/1111.

[SK19] Connor Shorten and Taghi M. Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning”. In: *Journal of Big Data* 6.1 (2019), p. 60. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

[Wil92] Ronald J. Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine Learning* 8.3 (1992), pp. 229–256. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696.

[Yam+18] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9.4 (2018), pp. 611–629. DOI: 10.1007/s13244-018-0639-9. URL: https://doi.org/10.1007/s13244-018-0639-9.

Papers

[Bho+] Srinadh Bhojanapalli et al. *Low-Rank Bottleneck in Multi-head Attention Models*. URL: http://proceedings.mlr.press/v119/bhojanapalli20a/bhojanapalli20a.pdf.

[Dos+21] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[DR20] Arun Das and Paul Rad. *Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey*. 2020. arXiv: 2006.11371 [cs.CV].

[Dut21] Arup Duttaroy. *3 X's of Explainable AI*. Techreport. 2021. URL: https://www.lntinfotech.com/wp-content/uploads/2021/01/3xExplainable-AI.pdf.

[Fri01] Jerome H. Friedmann. *Greedy Function Approximation: A Gradient Boosting Machine*. Feb. 2001. URL: https://statweb.stanford.edu/~jhf/ftp/trebst.pdf.

[GBV20] Margherita Grandini, Enrico Bagli, and Giorgio Visani. *Metrics for Multi-Class Classification: an Overview*. 2020. arXiv: 2008.05756 [stat.ML].

[GSC21a] Julie Gerlings, Arisa Shollo, and Ioanna Constantiou. *Reviewing the Need for Explainable Artificial Intelligence (xAI)*. 2021. arXiv: 2012.01007 [cs.HC].

[GSC21b] Julie Gerlings, Arisa Shollo, and Ioanna Constantiou. *Reviewing the Need for Explainable Artificial Intelligence (xAI)*. 2021. arXiv: 2012.01007 [cs.HC].

[He+15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[HLM12] Amanda Hutton, Alexander Liu, and Cheryl Martin. *Crowdsourcing Evaluations of Classifier Interpretability*. 2012. URL: https://www.aaai.org/ocs/index.php/SSS/SSS12/paper/view/4267.

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. 2015. arXiv: 1508.04025 [cs.CL].

[Maa+] Andrew Maas et al. *Learning Word Vectors for Sentiment Analysis*. URL: https://ai.stanford.edu/~ang/papers/acl11-WordVectorsSentimentAnalysis.pdf.

[MCF20] T. Nathan Mundhenk, Barry Y. Chen, and Gerald Friedland. *Efficient Saliency Maps for Explainable AI*. 2020. arXiv: 1911.11293 [cs.CV].

[Min+21] Tran Ngoc Minh et al. *Automated Image Data Preprocessing with Deep Reinforcement Learning*. 2021. arXiv: 1806.05886 [cs.CV].

[Nwa+18] Chigozie Nwankpa et al. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 2018. arXiv: 1811.03378 [cs.LG].

[Nwa20] Martin C. Nwadiugwu. *Neural Networks, Artificial Intelligence and the Computational Brain*. 2020. arXiv: 2101.08635 [q-bio.NC].

[PV20] Erika Puiutta and Eric MSP Veith. *Explainable Reinforcement Learning: A Survey*. 2020. arXiv: 2005.06247 [cs.LG].

[Ram+19] Prajit Ramachandran et al. *Stand-Alone Self-Attention in Vision Models*. 2019. arXiv: 1906.05909 [cs.CV].

[RSG16a] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG].

[RSG16b] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG].

[San+20] Gokula Krishnan Santhanam et al. *On Evaluating Explainability Algorithms*. 2020. URL: https://openreview.net/forum?id=B1xBAA4FwH.

[She+18] Tao Shen et al. *Reinforced Self-Attention Network: a Hybrid of Hard and Soft Attention for Sequence Modeling*. 2018. arXiv: 1801.10296 [cs.CL].

[SKS16] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. *Action Recognition using Visual Attention*. 2016. arXiv: 1511.04119 [cs.LG].

Inproceedings

[AMA17] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[LG14] Omer Levy and Yoav Goldberg. "Neural Word Embedding as Implicit Matrix Factorization". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014, p. 3.

[Lia+21] Q. Vera Liao et al. "Introduction to Explainable AI". In: *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI EA '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380959. DOI: 10.1145/3411763.3445016. URL: https://doi.org/10.1145/3411763.3445016.

[LL17] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 4–5. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[Ngu+18] Long Nguyen et al. "Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation". In: May 2018, pp. 1–5. DOI: 10.1109/ISCAS.2018.8351550.

[RS15] Vikas C. Raykar and Amrita Saha. “Data Split Strategiesfor Evolving Predictive Models”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Annalisa Appice et al. Cham: Springer International Publishing, 2015, pp. 3–19. ISBN: 978-3-319-23528-8.

[Vas+17] Ashish Vaswani et al. “Attention is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

Online Resources

[Aro20] Himanshu Arora. *Attention Mechanisms in Vision Models*. retrieved 16.06.2021. Dec. 2020. URL: https://medium.com/jumio/self-attention-in-computer-vision-b929cca5caf8.

[B17] Nikhil B. *Image Data Pre-Processing for Neural Networks*. retrieved 05.07.2021. Sept. 2017. URL: https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258.

[Doc21a] Fast AI Docs. *Tutorial - Training a model on Imagenette*. retrieved 05.07.2021. 2021. URL: https://docs.fast.ai/tutorial.imagenette.html.

[Doc21b] PyTorch Docs. *REPRODUCIBILITY*. retrieved 10.07.2021. 2021. URL: https://pytorch.org/docs/stable/notes/randomness.html.

[Fut20] Romain Futrzynski. *Self-attention: step-by-step video | Peltarion*. retrieved 15.06.2021. 2020. URL: https://peltarion.com/blog/data-science/self-attention-video.

[luc21] lucidrains. *vit-pytorch*. retrieved 10.07.2021. 2021. URL: https://github.com/lucidrains/vit-pytorch.

[Sag19] Ram Sagar. *8 Explainable AI Frameworks Driving A New Paradigm For Transparency In AI*. retrieved 28.06.2021. Oct. 2019. URL: https://analyticsindiamag.com/8-explainable-ai-frameworks-driving-a-new-paradigm-for-transparency-in-ai/.

[Tea21] Pytorch Team. *ResNet - Deep residual networks pre-trained on ImageNet*. retrieved 16.08.2021. 2021. URL: https://pytorch.org/hub/pytorch_vision_resnet/.

[Ter19] Mohammed Terry-Jack. *Deep Learning: The Transformer*. retrieved 15.06.2021. June 2019. URL: https://medium.com/b.terryjack/deep-learning-the-transformer-9ae5e9c5a190.

[Tul16] Marco Tulio. *Local Interpretable Model-Agnostic Explanations (LIME): An Introduction*. retrieved 28.06.2021. Aug. 2016. URL: https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/.

[Ver20] Christian Versloot. *What are Max Pooling, Average Pooling, Global Max Pooling and Global Average Pooling?* retrieved 06.07.2021. Jan. 2020. URL: https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/.

[Yal20] Orhan G. Yalç06n. *5 Significant Reasons Why Explainable AI is an Existential Need for Humanity*. retrieved 16.08.2021. Dec. 2020. URL: https://towardsdatascience.com/5-significant-reasons-why-explainable-ai-is-an-existential-need-for-humanity-abe57ced4541.

Others

[Hel13] Helgi Helgason. "General Attention Mechanism for Artificial Intelligence Systems". PhD thesis. May 2013, pp. 44–43.

[WL20] Dr. Wei Wei and Prof. James Landay. *ML Interpretability and Intrinsic Models*. Tech. rep. Stanford University, 2020.

Appendix

.1 Attention Map Algorithm

```
1
2   def generate_attention_map(att_tensor, patches_per_row, patch_size):
3
4       attn = torch.squeeze(input=att_tensor, dim=0)
5       attn = torch.mean(attn, dim=[0, 1])
6       attn = torch.sum(attn, dim=0)
7       attn = attn.detach().cpu().numpy()
8       attn = attn[1:].copy()
9       attn = np.interp(attn, (attn.min(), attn.max()), (0, +1))
10
11      b = np.reshape(attn, (patches_per_row, patches_per_row))
12      b = np.kron(b, np.ones((patch_size, patch_size), dtype=b.dtype))
13
14      return b
```

Listing 1: Convert Attention Tensor to corresponding Attention Matrix

.2 Masking Algorithms

```
1
2   def cnn_mask_image(model, high, low, class_num):
3
4     x_high, y_high = load_single_image(high)
5     temp, mask = model.lime_and_explain(y_high, class_num)
6     mask = np.where(mask == 1, 255, mask)
7
8     masked_image = Image.composite(just_load_resize_pil(high),
          just_load_resize_pil(low),
9          Image.fromarray(np.uint8(mask)).convert('L'))
10
11    return masked_image
```

Listing 2: Masking with CNN and Lime Explanation

```
1
2     def vit_mask_image(model, high, low):
3         x_high, y_high = load_single_image(high)
4         preds, attns = model.predict_and_attents(x_high)
5
6         a_map = generate_attention_map(attns, 20, 16)
7         mask = sliding_window_method(a_map)
8         mask = np.where(mask == 1, 255, mask)
9
10        masked_image = Image.composite(just_load_resize_pil(high),
              just_load_resize_pil(low),
11            Image.fromarray(np.uint8(mask)).convert('L'))
12
13        return masked_image
```

Listing 3: Masking with ViT and Attention Map Explanation

Figure .1: Masking Algorithms Examples

.3 Complete Codebase

The complete codebase for the case study used in this bachelor thesis, including the augmented dataset, all AI models, as well as the implementation for the attention-based explainer and the metrics, can be found in the following public GitHub repository:

`https://github.com/NiklasUllmann/DHBW_BachelorThesis_Code`