Zurich University
of Applied Sciences

School of
Engineering

IAMP Institute of Applied
Mathematics and Physics

Bachelor thesis (Computer Science)

Evaluating XAI Algorithms in Skin Cancer Classification: A Path towards Trustworthy AI Systems

Author	Celina Homa
Main supervisor	Monika Reif
Expert	Alan Hochberg
Date	09.06.2023

Declaration Of Originality

By submitting this Bachelor's thesis, the undersigned student confirms that this thesis is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the Bachelor thesis have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date: Name Student:

Abstract

Artificial intelligence (AI) has become a valuable tool in skin cancer classification. However, its widespread adoption is limited by concerns about trustworthiness and its black-box nature. One promising approach to increasing trust is to improve the explainability of AI systems. This thesis explores the potential of using algorithms from the AIX360 library to achieve this goal. The algorithms - ProtoDash, DIP-VAE, LIME, SHAP and CEM - were evaluated for their runtime, simplicity, explainability/interpretability and stability in the context of skin cancer classification. The results suggest that each algorithm offers unique insights, but none can be considered universally superior. In particular, LIME and SHAP showed a promising balance between interpretability and stability, making them strong candidates. Further work is needed to identify the optimal combination of algorithms to increase confidence and to adapt these algorithms to handle complex data sets. This work is a step towards developing AI models that are not only effective, but also transparent and accountable, furthering the quest for trustworthy AI in healthcare.

Keywords: Machine Learning, Artificial Intelligence, Explainable AI, AIX360, Skin Cancer Detection, Image Classification.

Table of Contents

1. Introduction

Artificial intelligence (AI) has entered many critical sectors, including healthcare, demonstrating its potential to transform processes, enhance decision-making capabilities and improve overall operational efficiency. As AI systems evolve, maintaining their trustworthiness, reliability and transparency remains essential [1]. This is particularly important in healthcare, a sector where the accuracy of diagnoses can have a profound impact on the course of a patient's life. [2]

AI systems, when given access to high quality data and built using appropriate models, are proving effective in diagnosing diseases such as skin cancer. In fact, under specific and highly artificial and controlled conditions, these systems have been able to match, and in some cases surpass, the diagnostic expertise of experienced professionals [3]. A notable example is the use of AI in the diagnosis of skin cancer, the most common form of cancer worldwide [4]. Early detection in such cases can greatly improve a patient's chances of recovery, underlining the critical importance of AI in this area.

However, the wider acceptance and application of AI systems is challenged by their black-box nature. The complexity of AI systems can make their decision-making processes unclear and difficult to understand. Improving the explainability of AI systems is therefore crucial to enable users, developers and domain experts to develop trust and use AI systems effectively. For example, in a medical setting, explainable AI (XAI) could clarify diagnoses, help clinicians justify treatment decisions, and promote more effective patient-clinician dialogue. [5]

1.1. State of the Art

XAI has emerged as a vibrant area of research in response to the increasingly complex nature of AI systems and the challenges this complexity poses to their transparency and trustworthiness. Various tools and techniques are being developed to increase the transparency of AI systems, a notable example being IBM's AIX360 library [6]. This library provides a set of algorithms specifically designed to effectively interpret the predictions of machine learning (ML) models.

This progress has been recognised at a regulatory level, with bodies such as the European Commission proposing guidelines for the ethical and responsible use of AI. However, a standardised approach that defines how AI systems should be designed to ensure reliable and trustworthy predictions is currently lacking [7]. This highlights the need for research into practical applications and evaluations of existing XAI tools such as the AIX360 library.

Furthermore, the need for explainability in AI is particularly strong in critical domains such as healthcare. Recent research has explored the interpretability of convolutional neural networks (CNNs), particularly in the field of dermatology. For example, one study aimed to 'open the black box' of CNNs by visualising and examining their learned feature maps in the context of skin conditions. The researchers found that the CNNs focused on features similar to those used by dermatologists in their diagnoses, suggesting a promising way to improve the explainability of these AI systems. [8]

Another study applied deep learning techniques to dermoscopic images for melanoma detection and achieved high accuracy in their model predictions. The authors used local interpretability methods, namely Gradient-weighted Class Activation Mapping (Grad-CAM) and Kernel SHAP, to understand the reasoning behind these predictions. Despite the high accuracy of their models, they found that they occasionally assigned importance to irrelevant features, and different models with similar accuracy produced different explanations. [9]

These findings highlight both the potential and the challenges of explainability in AI. They highlight the ability of AI systems to focus on relevant features and make accurate predictions, but also point to their tendency to occasionally assign impor-

tance to irrelevant aspects.

Moreover, the observed variance in explanations between models underscores the complexity of these systems and the need for standardised methods for interpreting their results. Further research is therefore needed to optimise the use of XAI tools, such as the AIX360 library, to improve the transparency and accountability of AI systems, particularly in critical applications such as skin cancer classification.

1.2. Research Objectives

The primary objective of this thesis is to investigate the potential of the AIX360 library for enhancing the trustworthiness of AI systems, using a skin cancer classification system as a case study.

The thesis proceeds by first providing a detailed exploration of the theoretical background necessary for this thesis. It then discusses the methods used to develop the AI system, along with a comprehensive review of potential approaches to improve explainability. The thesis then turns its attention to evaluating the results obtained, reflecting on the lessons learned, and considering potential future directions in the field.

The overall aim of this thesis is to develop an AI system that is not only capable of accurately classifying skin lesions, but also capable of providing comprehensible explanations for its predictions. This is expected to contribute to making AI more transparent and trustworthy, thereby encouraging its wider acceptance and application.

Specifically, this thesis focuses on improving the explainability of AI systems in the context of skin cancer classification. Using IBM's AIX360 library, it aims to design and implement a classification model that can effectively distinguish between different skin lesions.

This thesis is guided by several key questions: it explores the feasibility of achieving sufficient explainability for the AI system using algorithms from the AIX360 library, seeks the optimal combination of these algorithms to increase trustworthiness, and aims to identify which algorithms are particularly well suited to the context of

skin cancer classification. These questions drive the quest for AI models that are not only functionally effective, but also more transparent and accountable in their predictions.

2. Theoretical Background

The following chapter presents the theoretical background necessary to understand the topics of machine learning, artificial intelligence, and explainable AI.

2.1. Machine Learning Concepts

ML, a subfield of AI, uses algorithms to recognise patterns in data and refine decisions over time, constantly improving the capabilities of computer systems.

Deep learning, an evolution of ML, uses large neural networks to analyse data and make predictions independent of human intervention. ML's widespread applications range from pattern recognition and computer vision to finance and medicine. [10]

The following subsections provide a brief overview of important ML concepts.

2.1.1. Convolutional Neural Networks

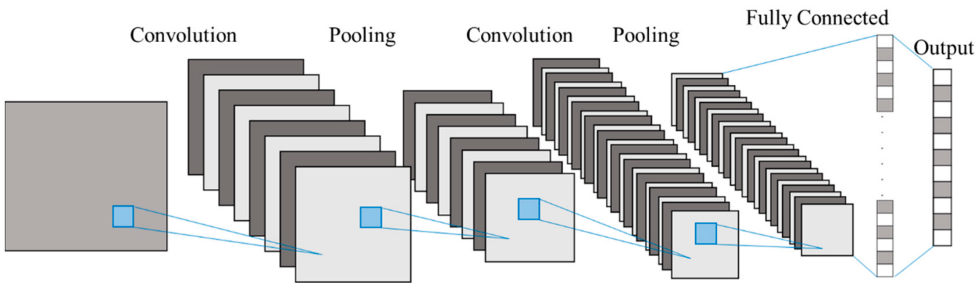


Figure 2.1.: Schematic representation of a CNN architecture with the composition of two convolutional and corresponding pooling layers, followed by a fully connected layer and an output layer, specifically designed for image processing tasks. [11]

CNNs are specialised neural networks designed for image processing. Their ability to learn and recognise patterns and features in images, even when these patterns are not immediately obvious to the human eye, makes them particularly suitable for image and audio identification, recognition and processing tasks. Able to handle large amounts of data, CNNs scale effectively to handle complex tasks. [12]

In CNNs, several types of layers work together, each with different properties, as shown in Figure 2.1, without the flatten layer [13], [14]:

- **Convolutional Layer (Conv2D)**: This layer learns local features of an image through kernels or filters, creating feature maps.
- **Pooling Layer (MaxPooling2D)**: By reducing the spatial size of the feature maps, this layer controls the number of parameters and computational cost, and helps prevent overfitting.
- **Flatten Layer**: This layer transforms the two-dimensional feature maps into a one-dimensional vector to be used as input for the fully connected layers.
- **Fully Connected Layer (Dense)**: This layer connects each neuron in one layer to every neuron in the next, allowing learning from distributed feature data. This layer can also be used as the final output layer when paired with an appropriate activation function, such as softmax for multi-class classification or sigmoid for binary classification tasks.
- **Dropout Layer**: To prevent overfitting and provide a form of model averaging, this layer temporarily removes random neurons during training.

2.1.2. Evaluation Metrics

Evaluation metrics are crucial in determining the performance of a ML model. There is a wide variety of these metrics, each suitable for different application needs. The type of classifier used for the model also plays a critical role in choosing the right metric.

Classifiers can differ in the number of labels or classes into which they can divide the data [15]:

- **Binary classifiers** are used to categorise data into one of two unique classes. For example, an image showing a skin lesion could be classified as either benign or malignant, a binary yes/no case.
- **Multiclass classifiers** are used when data needs to be categorised into more than two classes. For example, an image of a skin lesion can be classified into one of seven possible skin lesion types.
- **Multilabel classifiers** come into play when data is labelled with multiple descriptors. For example, an image of a skin lesion might have multiple labels indicating the location, the type of disease, and the gender and age of the patient.

The metrics described in this section are primarily used in the context of binary classifiers. However, the same metrics can be applied in a multiclass context where each class is evaluated separately.

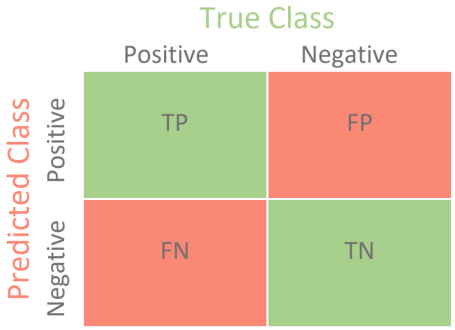


Figure 2.2.: Diagram showing the structure of a confusion matrix for binary classification, showcasing the distribution of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). [16]

First, a confusion matrix is created, similar to the one shown in Figure 2.2. This matrix illustrates the relationship between the predicted and actual classes. True Positives (TP) and True Negatives (TN) represent cases that were correctly predicted, while False Positives (FP) and False Negatives (FN) represent cases that were incorrectly predicted by the model.

From this, several key metrics can be calculated [15]:

- ***Precision*** is a measure of the proportion of positive predictions that are actually correct:
$$\frac{TP}{TP + FP}$$
- ***Recall*** is a measure of the proportion of actual positive cases that were correctly predicted:
$$\frac{TP}{TP + FN}$$
- ***Accuracy*** is a measure of how often the model makes correct predictions:
$$\frac{TP + TN}{TP + FN + TN + FP}$$
- ***F1-score*** is a combination of precision and recall, and is calculated as the harmonic mean of precision and recall:
$$\frac{2TP}{2TP + FP + FN}$$

Choosing the right metric is critical, as it depends heavily on the type of data and the specific problem at hand. Inappropriate metrics can lead to incorrect results. In certain contexts, precision or recall may be more important, and so models should be trained to perform better on one or the other.

For example, in the medical sector, models should prioritise high recall over precision. While false positive predictions may be somewhat tolerable, a high number of false negatives is unacceptable. The primary goal in healthcare is to identify all individuals with a particular condition, so it is vital to optimise the model for high recall. [17]

2.2. Artificial Intelligence Concepts

AI is at the forefront of technological advancement, embodying machine capabilities that mimic human intelligence. It uses data analysis to influence modern technology, including smart devices and voice assistants. Techniques such as natural language processing and computer vision are fundamental to AI, speeding up decision-making and automating tasks. [10]

The following sections provide an overview of some important concepts in AI.

2.2.1. Definitions

The following subsection defines key terms frequently used in the field of XAI, focusing in particular on the concepts of interpretability, explainability and transparency. [18], [19]

2.2.1.1. Interpretability

Interpretability refers to the ability to make the outputs or decisions of an AI system understandable and meaningful to human users. Rather than delving into the technical details of the system, interpretability emphasises the communication of essential and actionable insights derived from the AI's decisions. It involves translating the results of the AI system into concepts that are understandable and relevant to the human user, helping them to make informed decisions.

2.2.1.2. Explainability

Explainability encompasses the methods and techniques used to present and explain the decisions or actions of an AI system. It involves demonstrating the specific processes and factors that led to a decision, typically in a detailed and technical manner. It may involve outlining the steps of an algorithm, illustrating the impact of different parameters, or revealing the significance of different input characteristics. It is often aimed at making the operation of the AI system understandable to technical practitioners, who can use this insight to analyse and improve the system.

2.2.1.3. Transparency

Transparency refers to the degree to which the inner workings of an AI system are open to observation. It involves providing insight into the underlying mechanisms of the AI, including data inputs, algorithms and decision-making processes. Transparency focuses on revealing how the AI system works, without necessarily simplifying or contextualising it for non-expert users. It aims to promote accountability and trust by ensuring that nothing in the AI system is hidden or mysterious.

2.2.2. Explainable AI

XAI represents an important shift in the field of AI. XAI is designed to provide clear and understandable explanations of its decision-making process. This feature is combined with a set of specialised tools and frameworks that help users to interpret and understand the predictions made by their ML models. The importance of XAI is particularly evident in areas such as healthcare, where decisions made by AI can have a huge impact on patient outcomes.

The key difference of XAI is its ability to bring explainability, interpretability and transparency to AI systems. It provides users with a deeper understanding of the decision-making process of these systems, making AI less of a black box and more accessible. It allows users to understand how and why an AI system reaches conclusions or predictions, building a sense of trust among experts, developers and end users.

In addition, the use of XAI supports better decision making. The ability to assess the validity of AI-generated results enables users to make more informed decisions, adding another layer of trust to the AI system's predictions. The transparency and clarity provided by XAI helps identify and correct potential errors or biases in the AI system or its data. This benefit is key to building more reliable and transparent models.

As AI continues to expand across industries, explainability, interpretability and transparency have become not only ethical but also regulatory requirements. XAI meets this need by providing clear insight into how AI systems operate, helping or-

ganisations to remain compliant and reduce the risk of legal or ethical complications. [20]

2.3. XAI Concepts

This section explains the taxonomy required for XAI and also provides a comprehensive overview of the available AIX360 algorithms, both those that are implemented in the course of this thesis and those that are not.

2.3.1. Taxonomy

Figure 2.3 provides a visual representation of the XAI algorithm taxonomy presented in this chapter. This tree diagram serves as a guide to help users navigate through the AIX360 algorithms.

Each decision point in the tree asks a question that corresponds to different explanations, allowing users to find an XAI algorithm that suits their specific needs and applications. Some algorithms are applicable to models working with tabular, image and text data, and some are specified for only one type of data.

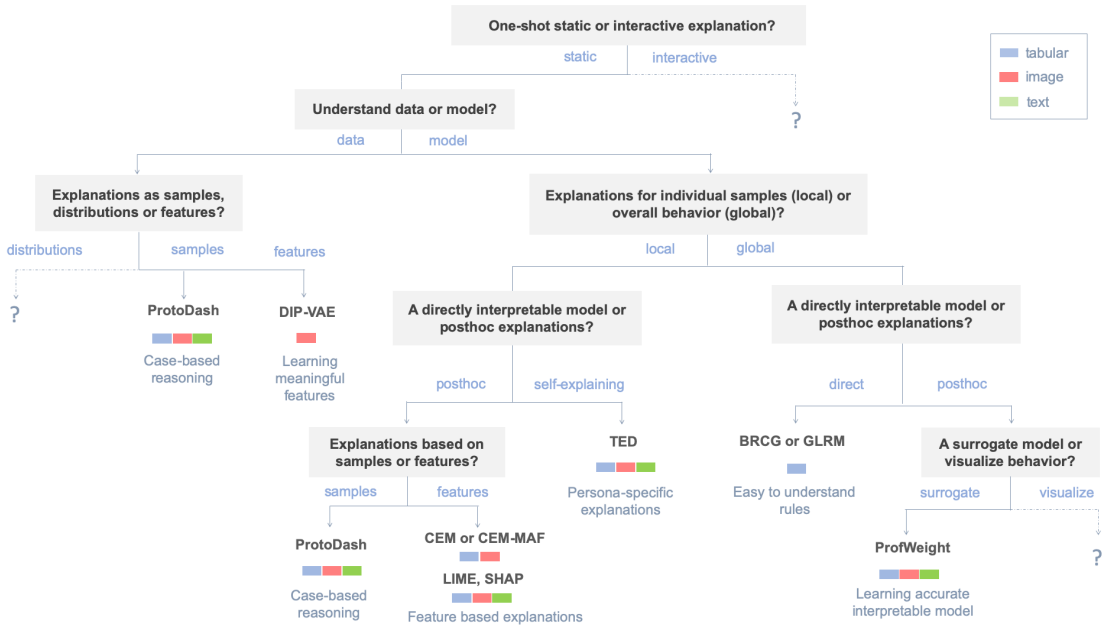


Figure 2.3.: A visual representation of the taxonomy of AI Explainability 360 (AIX 360) algorithms that serves as a user guide to navigate through different explanations and select the most suitable XAI algorithm. [21]

The taxonomy uses several definitions for clarity [6]:

- **Static explanations** remain unchanged after user feedback, while **interactive explanations** adapt to the user's queries.
- **Local explanations** focus on a single prediction, while **global explanations** provide an overview of the model's behaviour.
- **Directly interpretable models** are inherently understandable, whereas **post-hoc explanations** require an helper method after training. **Self-explaining models** generate local explanations but may not be directly interpretable.
- A **surrogate model** is an interpretable model that approximates a more complex model, while **visualisations** are not complete models themselves, but highlight specific aspects.

While the taxonomy is not intended to be perfect or complete, it aims to provide a simple, comprehensive guide to AIX 360 algorithms that is suitable for different users. Algorithms that are not currently represented in the toolkit are marked with a question mark.

2.3.2. XAI Algorithms

This section provides a brief overview of the existing XAI algorithms. However, it's important to note that not all of these algorithms will be used for implementation. The suitability of an algorithm depends very much on the specific characteristics of the data and the model. Careful consideration must therefore be given to selecting the most appropriate algorithm for each particular application. The latest version of the AIX 360 toolkit contains or links to the following algorithms [6], [22]:

ProtoDash ProtoDash [23] is an interpretability technique that selects representative and diverse samples to summarise a data set or explain a particular test instance. This algorithm also learns non-negative importance weights for each of the selected samples, further helping to understand the structure of the data set and the reasoning behind certain predictions.

DIP-VAE The Disentangled Inferred Prior Variational Autoencoder (DIP-VAE) [24] is a model that learns high-level independent features from images that may carry semantic interpretation. The strength of this model lies in its ability to isolate and represent complex features, contributing to a better understanding and explanation of image analysis.

LIME Local Interpretable Model-agnostic Explanations (LIME) [25] is a widely used interpretability algorithm that provides local explanations of model predictions. LIME generates explanations by locally approximating the model's decision boundary and explaining the factors that contribute positively or negatively to a given prediction.

SHAP SHAP (SHapley Additive exPlanations) [26] is an interpretable algorithm that assigns each feature an importance value for a given prediction. It does this by using game theory principles, specifically Shapley values, which provide a unified measure of feature importance.

CEM The Contrastive Explanations Method (CEM) [27] generates local explanations by identifying what is minimally sufficient to maintain the original classification, and what should necessarily be absent. It helps to generate easily interpretable insights, thereby improving the understanding of model predictions.

TED Training data set-based Explanation (TED) [28] is an explainability framework that uses domain-relevant explanations from the training data set to predict both labels and explanations for new instances. Its implementation, known as the Cartesian product, is the most basic version of TED. This approach facilitates a coherent understanding of new predictions based on past data, increasing the transparency and interpretability of the AI system. TED was not used due to a lack of documentation on its implementation.

BRCG Boolean Decision Rules via Column Generation (BRCG) [29] is a algorithm for learning small, interpretable Boolean rules in disjunctive normal form for

binary classification tasks. This approach facilitates comprehensible decision rules and promotes greater transparency in model predictions. This algorithm is not implemented as it is specified for tabular data.

GLRM Generalised Linear Rule Models (GLRM) [30] are powerful regression models that learn a linear combination of constraints through a generalised linear model (GLM) link function. GLRM supports a range of link functions such as identity and logit, making it a versatile tool for modelling and understanding real-valued data. GLRM is not implemented as it is specified for tabular data.

ProfWeight ProfWeight [31] is an algorithm that adjusts the weights of the training set, based on a given interpretable model and a high-performing complex neural network. By retraining the interpretable model on this reweighted training set, ProfWeight can improve the performance of the interpretable model, providing better predictions while maintaining interpretability. This algorithm was not used due to incompatibilities with the implemented model.

2.3.3. XAI Metrics

Recognising the need for effective evaluation tools, the AIX 360 toolkit includes two notable metrics designed to evaluate the performance of selected algorithms. These metrics not only allow users to quantify the efficiency of an algorithm, but also provide valuable insights that can be used to further refine and enhance the underlying models. [32]

Faithfullness The faithfulness metric [33] is designed to assess the relationship between the importance of an attribute, as determined by the interpretability algorithm, and the impact of that attribute on the overall performance of the predictive model. Ideally, an attribute assigned a higher importance would have a proportionately higher impact on model performance and vice versa. This evaluation is done by progressively eliminating attributes that are marked as important and observing the subsequent effect on model performance. The final step is to calculate the

correlation between the importance (weights) assigned to the attributes and their respective impact on model performance.

Monotonicity The monotonicity metric [34] quantifies the impact of individual attributes on the performance of the predictive model by progressively including each attribute according to its order of importance. As each attribute is added, a corresponding increase in model performance should be observed, resulting in a monotonic increase in performance. In essence, this metric monitors the evolution of model performance as more influential attributes are introduced, thereby assessing the effectiveness of the model's feature prioritisation.

2.4. Skin Cancer Use Case

This section aims to provide comprehensive details of the skin cancer use case. The knowledge provided here will form the basis for understanding and further building a skin cancer classifier.

2.4.1. Conventional Skin Cancer Diagnosis

Conventional skin cancer diagnosis revolves around visual inspection techniques. The Skin Cancer Foundation's recommendations emphasise the importance of monthly self-examinations coupled with annual professional examinations to detect potential skin cancers [35].

If a healthcare professional identifies an area of concern during these checks, a more detailed examination will follow. During this assessment, the dermatologist will look at various characteristics of the suspicious spot, such as its size, shape, colour and texture, as well as any signs of bleeding or scaling. The examination may also include nearby lymph nodes, which will be examined for any abnormal enlargement.

A key tool often used by dermatologists is a dermatoscope, a hand-held magnifying device that allows detailed examination of the epidermis and underlying layers. If areas of concern are found, a biopsy may be performed, where a sample of the suspicious skin lesion is taken and sent to a laboratory for further analysis. If a biopsy confirms the presence of skin cancer, the dermatologist will discuss the type of cancer and possible treatment options with the patient.

Overall, the current paradigm in skin cancer diagnosis places a strong emphasis on early detection in order to minimise the cost and extent of treatment and maximise the chances of a successful outcome. While effective, this traditional approach may benefit from improvements or innovations in diagnostic technology such as AI. [36]

2.4.2. Skin Cancer Types

Figure 2.4 shows a visual representation of the diverse types of skin lesions, with each class depicted by a randomly selected image from the data set.

Figure 2.4.: Exemplary illustration of seven classes of skin cancer, each represented by a randomly selected image, showing the variety of appearance and characteristics of different types of skin cancer.

The following provides an overview of the different types of skin lesions relevant to the skin cancer use case and the data set that was later implemented [37]:

- **Actinic Keratoses and Intraepithelial Carcinoma / Bowen's Disease (akiec):** Actinic keratoses are scaly, rough patches on sun-exposed skin that can progress to invasive squamous cell carcinoma. Intraepithelial carcinoma, also known as Bowen's disease, is a non-invasive variant of squamous cell carcinoma characterised by red, scaly patches. Both lesions can be caused by exposure to UV light or human papillomavirus infection, and both have pigmented variants.
- **Basal cell carcinoma (bcc):** BCC is a common type of epithelial skin cancer that rarely metastasises but is destructive if left untreated. It occurs in several morphological variants, including flat, nodular, pigmented and cystic.
- **Benign keratosis-like lesions (bkl):** These lesions include seborrhoeic keratoses, solar lentigo and lichen planus-like keratoses. They are biologically similar and can have a varied dermoscopic appearance. Lichen planus-like keratoses are particularly challenging as they can have morphological features that mimic melanoma.
- **Dermatofibroma (df):** Dermatofibromas are benign skin lesions that occur either as a result of benign proliferation or as an inflammatory response to minimal trauma. They are characterised by peripheral reticulation with a central white patch indicating fibrosis.
- **Melanoma (mel):** Melanoma is a malignant neoplasm derived from melanocytes and can occur in a variety of forms. They are often chaotic in appearance and some melanoma-specific criteria depend on the anatomical site.

- **Melanocytic nevi (nv):** Commonly known as moles, these are benign neoplasms of melanocytes that occur in many variants. They are usually symmetrical in colour and structure.
- **Vascular lesions (vasc):** This category includes a variety of conditions including cherry angiomas, angiokeratomas, pyogenic granulomas and haemorrhages. Angiomas are characterised on dermoscopy by red or purple colour and solid, well-circumscribed structures known as red clots or lacunes.

Understanding the differences between different skin lesions is critical because misclassification by AI models can lead to inaccurate diagnoses and inappropriate treatment plans. By recognising the unique characteristics of each lesion type, model performance can be better evaluated, potential areas of confusion can be identified and algorithms can be refined to improve diagnostic accuracy, ultimately ensuring more effective patient care.

2.4.3. AI Skin Cancer Diagnosis

AI is making progress in the field of skin cancer diagnosis, with several research papers demonstrating its effectiveness and potential for real-world applications.

One study, published in Nature, describes how AI-based algorithms for classifying suspicious skin lesions were implemented in a mobile health (mHealth) app. In 2019, a large Dutch health insurer offered 2.2 million adults free access to this mHealth app for skin cancer detection, and the impact on dermatological healthcare consumption was retrospectively examined. This study reported that users of the mHealth app had more claims for (pre)malignant skin lesions than controls (6.0% vs 4.6%). There was also more than three times the risk of claims for benign skin tumours and nevi for mHealth users compared to controls (5.9% vs 1.7%). [38]

In addition, researchers at MIT have developed an AI system that uses deep convolutional neural networks (DCNNs) to improve early detection of melanoma, a type of skin cancer responsible for over 70% of all skin cancer-related deaths. The system uses wide-field photography, common in most smartphones, to quickly and effectively identify suspicious pigmented lesions (SPLs), an early sign of skin cancer. The AI system was trained on a large data set of wide-field images and achieved over

90.3% sensitivity in distinguishing SPLs from non-suspicious lesions, skin and complex backgrounds. This AI tool demonstrates the potential of computer vision and deep neural networks to achieve accuracy comparable to expert dermatologists, promoting more efficient dermatological screening in primary care and enabling earlier treatment of melanoma. [39], [40]

2.4.4. Imaging Technologies

Effective use of AI in medicine and healthcare, particularly in dermatology, requires accurate handling of medical data. Depending on the specific type of image or dermatological use case, the AI model needs to be developed and fine-tuned accordingly. Some of the commonly used medical images in dermatology and broader medical contexts include the following [41]:

- **Digital images** are taken with digital cameras or scanners and stored digitally. In dermatological imaging, these images often document skin conditions or other external parts of the body. When combined with other imaging techniques such as ultrasound, they can also provide images of internal organs or structures.
- **X-rays** use radiation to create images inside the body. Common applications include visualising bones and fractures and screening for certain types of cancer, such as lung cancer.
- **Magnetic Resonance Imaging (MRI)** uses strong magnetic fields and radio waves to produce detailed images of the body's internal structures. MRI is particularly useful for imaging soft tissues such as the brain, spinal cord and joints.
- **Computed Tomography (CT) scans** use X-rays and computer technology to produce detailed 3D images of the body's internal structures. CT scans are usually used to help diagnose injuries and diseases of the head, chest, abdomen and pelvis.
- **Positron emission tomography (PET) scans** use radioactive tracers to produce images of the body's metabolic activity. They are often used to detect

and monitor the progression of cancer and to diagnose conditions such as Alzheimer's disease.

- **Ultrasound** uses high-frequency sound waves to create images inside the body. Typical applications include monitoring foetuses during pregnancy and diagnosing conditions such as heart disease and blood clots.

A particular feature of medical data, such as images or videos, is the metadata used to identify the patient and the examination in which the data was generated. Metadata in medical images provides important contextual information about the image and the patient, including patient identification, imaging parameters, clinical context, quality assurance, and privacy and security information. This information is essential for proper interpretation of the image, ensuring patient safety and privacy, and maintaining image quality and consistency over time. [42]

3. Methods

The methodology of this thesis is divided into three primary segments: implementing the skin cancer classification model, implementing the XAI algorithm, and evaluating.

Firstly, the implementation of the skin lesion classification model is described, including an analysis of the data used, its selection and pre-processing, the architecture of the model, its training process and the model evaluation.

Secondly, the implementation of the selected XAI algorithms is presented. This includes both data and model explanation algorithms, focusing on specific implementations such as ProtoDash, DIP-VAE for data explanation and LIME, SHAP and CEM for model explanation.

Finally, the evaluation procedures for these implementations are described, focusing on criteria such as runtime, simplicity, interpretability/explainability, and stability.

These distinct but connected segments provide a comprehensive insight into the research process, each contributing to the overall goal of improving the explanatory power of AI in skin cancer classification.

3.1. Implementation of the skin lesion classification model

This chapter discusses the methodology used to develop an AI model for skin cancer classification. The approach includes analysis of available data, selection of appropriate data sets, data pre-processing, and model implementation. Methods to evaluate and validate the performance of the model, such as performance metrics,

are also considered.

3.1.1. Data

A high-quality data set is an important foundation for any ML model. The quality of a data set determines both the effectiveness of model training and the accuracy of its predictions for new cases. Given the sheer volume of freely available data sets, it is crucial to evaluate and test them before committing to a data set.

Data sets can vary greatly in terms of data volume and classifier types. In the context of skin cancer, binary classifiers distinguish between malignant and benign lesions, while multi-class classifiers distinguish between different types of malignant or benign skin lesions.

The two data sets used in the implemented models were selected based on an analysis of the International Skin Imaging Collaboration (ISIC) image data sets [43]. This study evaluates different data sets by comparing factors such as duplicate images, data enhancement and number of images.

Duplicate images can lead to misleading results, as the presence of identical images in both the training and test sets can inflate the accuracy. The primary goal should be to learn patterns from one image and apply them to new images.

In an attempt to create a more comprehensive training set, data sets with a diverse representation of patient images of different skin colours were examined. Unfortunately, the data sets found were lacking in both quality and quantity. A notable example is the Diverse Dermatology Images (DDI) data set curated by Stanford AIMI [44]. However, this data set contains a mere 656 images, making it unsuitable for the purpose. This decision, although difficult, was made in order to mitigate potential bias arising from an inadequate number of images. It's worth acknowledging that this choice could lead to a bias towards lighter skin tones, potentially limiting the universal applicability of the developed solution to all skin types.

3.1.1.1. ISIC 2019 Data Set

The ISIC 2019 data set comprises 25,331 dermoscopic images covering nine different diagnostic categories, namely actinic keratosis, basal cell carcinoma, dermatofibroma, melanoma, melanocytic nevus, benign keratosis, squamous cell carcinoma, vascular lesions and none of the others. [45]

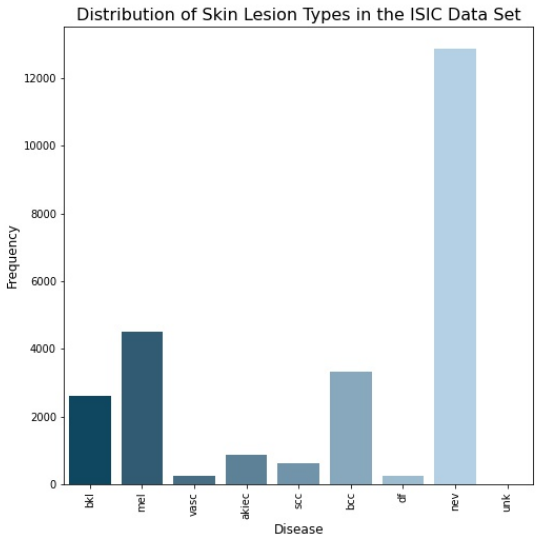


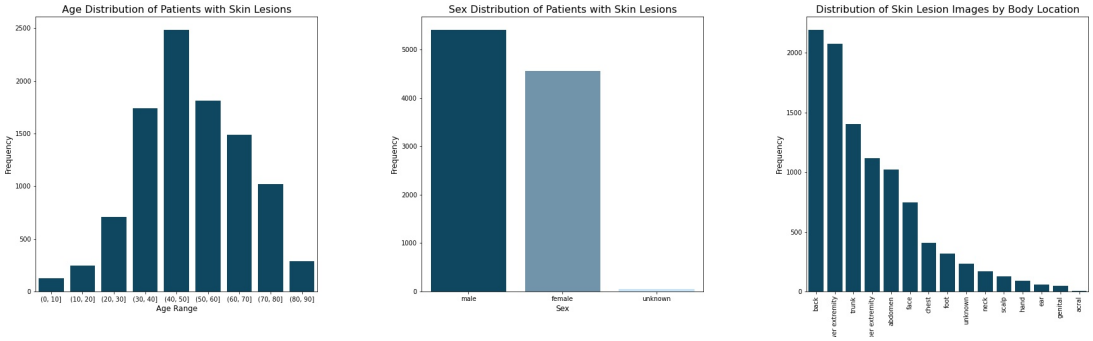
Figure 3.1.: Distribution of disease types in the ISIC 2019 data set. The vertical axis represents the number of cases, while the horizontal axis shows the labels. The graph provides insights into the prevalence of each disease type, with 'nev' being by far the most common, followed by 'mel', 'bcc' and 'bkl'.

Figure 3.1 shows the distribution of these diagnostic categories within the ISIC 2019 data set. The variety of conditions represented in the data allows for a comprehensive analysis of different skin diseases. This will aid in the development and validation of robust diagnostic models that can handle a wide range of skin conditions. [46], [47]

3.1.1.2. HAM10000 Data Set

The HAM10000 (Human Against Machine with 10000 training images) data set was collected over a period of 20 years and includes dermoscopic images from Austria

and Australia. In addition to cell type, the data set includes patient information such as gender, age and location of the pigmented lesion. Although these details do not directly affect the performance of the model, they provide valuable insight into the quality of the data set. [37]



(a) Age distribution of patients, showing a notable prevalence in the 40-50 years bracket.

(b) Gender distribution in the data set, indicating a higher representation of males.

(c) Image acquisition location distribution, revealing a dominance of back and lower extremities.

Figure 3.2.: Visual representations of the demographics of the HAM10000 data set, including distributions of age, gender and image capture location.

The data set contains seven initial classes: actinic keratosis, basal cell carcinoma, benign keratosis, dermatofibroma, melanocytic nevi, melanoma and vascular skin lesions. The age, sex and location distribution of these classes is shown in Figure 3.2 and the original distribution is shown in Figure 3.3a. The characteristics of each class are discussed in the subsection 2.4.2.

It was originally presented in the ISIC 2018 Challenge [43]. However, for the sake of clarity and consistency, it will be referred to exclusively as HAM10000 rather than ISIC 2018 throughout this thesis.

3.1.1.3. Data Set Selection

For the purposes of this thesis, a conscious decision was made to use the HAM10000 data set over the ISIC 2019 data set. Both data sets provide valuable resources for understanding and classifying skin lesions. However, given the specific requirements of this thesis, the HAM10000 data set proved to be a more appropriate choice.

One of the most compelling reasons for this choice is the diversity and quality of the HAM10000 data set. This compilation consists of a wide range of dermatoscopic images of skin lesions obtained from a variety of institutions including hospitals, clinics and private practices. The broad representation in this data set reflects the diversity encountered in real-world scenarios, enhancing the practicality and relevance of the developed model.

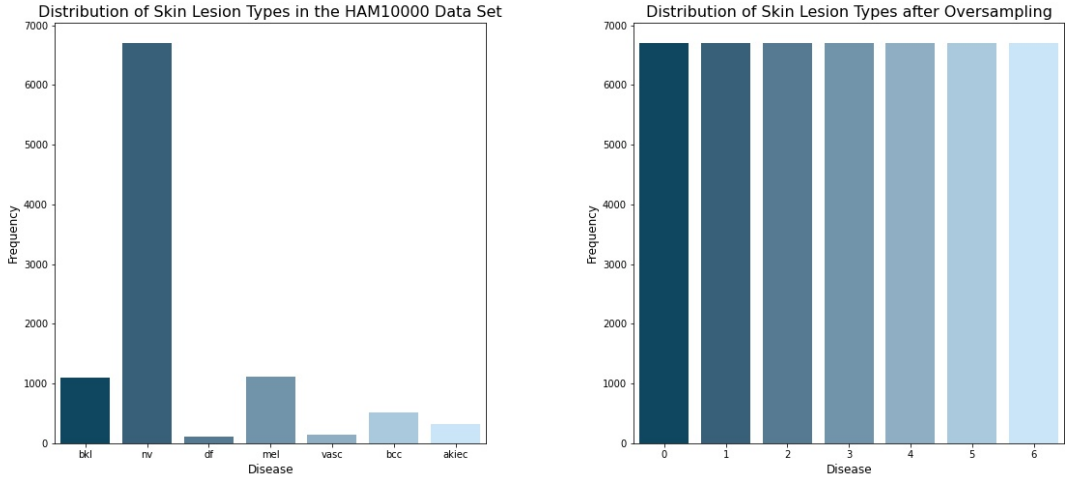
Another key benefit of the HAM10000 data set is the thorough pre-processing and curation it has undergone. This careful preparation has effectively weeded out duplicate and poor quality images. This care ensures that the data set used to train the model is free from potential sources of error, providing a more accurate representation of the data distribution, which in turn increases the reliability of the resulting model.

A notable advantage of the HAM10000 data set that influenced the selection is the availability of the data in a CSV file format containing the pixel values of the images. This format significantly reduces computation time, which is important given the scope of this thesis, which involves the processing and analysis of large amounts of data. The practical benefits of this accelerated computation are numerous, contributing not only to the efficiency of the research process, but also to the timely delivery of our results.

3.1.2. Preprocessing

Imbalanced data is a common problem in many real-world scenarios. Although such distributions often accurately reflect the actual distribution, they can pose a challenge to models that aim to learn patterns across different classes. In particular, for extreme imbalances, classes with fewer samples run the risk of being treated as noise, potentially leading to inaccurate predictions. [48]

This challenge is present in the HAM10000 data set, where there is a significant class imbalance between different disease types, as illustrated in Figure 3.3a. For example, the melanocytic nevus (nv) class contains 6,705 images, representing approximately two-thirds of the entire data set. To mitigate this imbalance, the technique of oversampling was used with the primary aim of preserving the data and maximising



(a) Initial distribution of disease types showing a clear predominance of the 'nv' class.

(b) Disease type distribution after oversampling, showing an equal number of samples across all classes.

Figure 3.3.: Comparison of disease type distributions before and after applying oversampling to address class imbalance.

the use of the limited data set size. As the HAM10000 data set is not particularly large, oversampling allows the model to utilise the full data set, thereby increasing its learning capacity and improving its performance over a wider range of data.

Oversampling essentially involves duplicating the instances of underrepresented classes until they equal the size of the largest class, ensuring a more balanced distribution. It essentially acts as a countermeasure against the model overlooking or misclassifying these underrepresented classes during the learning process. The distribution of disease types after oversampling can be seen in Figure 3.3b.

3.1.3. Model

A CNN was developed to classify skin lesion images into seven different classes. This chapter describes the architecture, training and evaluation of the model. The code implemented to build this model is in the Appendix A.2.1.

3.1.3.1. Model Architecture

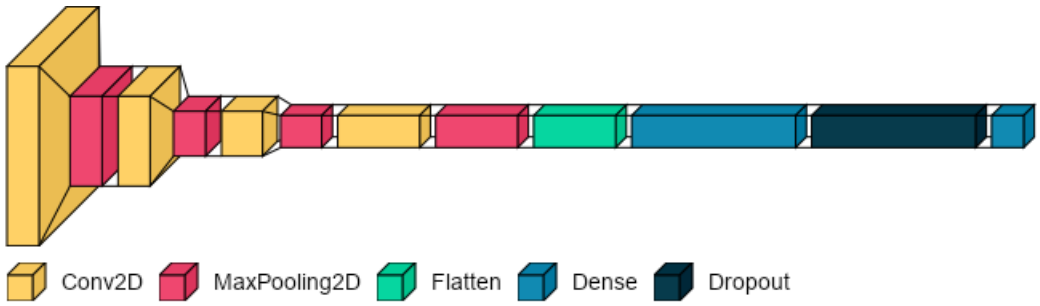


Figure 3.4.: Visual representation of the implemented model architecture. The model is composed of four Conv2D and MaxPooling layers, one Flatten and Dropout layer, and two Dense layers. Visualisation generated with VisualKeras [49].

The CNN used in this thesis is shown in Figure 3.4. This model follows a sequential architecture, i.e. it consists of a linear stack of layers. Each layer receives the transformed input from the previous layer and performs additional transformations.

The architecture of the model in focus can be structured as a sequence of several core layers, including Convolutional layers (Conv2D), MaxPooling layers (MaxPooling2D), a Flattening layer (Flatten), Dense layers, and a Dropout layer. These layers are organised into a cascade that transforms and refines the input image through multiple stages, ultimately leading to the classification output.

Convolutional and MaxPooling Layers The model starts with a Conv2D layer, which acts as a convolutional layer with 64 filters and a kernel size of 3x3. Each filter processes a specific portion of the input image, applying a transformation determined by the weights of the kernel filter. This transformation produces a feature map that captures the filtered aspects of the original image. This layer is followed by a MaxPooling2D layer, which reduces the spatial dimensions of the feature map, thereby reducing the computational complexity of the model. This pooling layer works by sliding a 2x2 filter over the feature map and extracting the maximum element from the area under consideration.

This pair of layers (Conv2D and MaxPooling2D) is repeated, but the Conv2D layer is extended to include 128 filters. This modification allows the model to recognise

more complex patterns within the data. Similar to its predecessor, this Conv2D layer is followed by a MaxPooling2D layer, which further reduces the spatial dimensions of the feature map.

Two further iterations of the Conv2D and MaxPooling2D pair follow, each time increasing the number of filters within the Conv2D layer (first to 256, then to 512). These increases allow the model to recognise even more complex patterns. Each is followed by a MaxPooling2D layer to limit the spatial size of the feature map.

Flattening, Dense, and Dropout Layers Upon completion of the Convolution and Pooling layers, the architecture applies a Flatten layer. This layer restructures the 2D matrix data into a column vector and prepares the processed features for the subsequent Dense layer.

The first Dense layer, a fully connected layer of 1024 units, performs the task of classifying the extracted features. Before the final layer, a Dropout layer is introduced as a form of regularisation. It helps to prevent overfitting by randomly dropping a fraction of the input units during each update in the training phase.

The final layer in the architecture is another Dense layer, this one with seven units. Each unit corresponds to one of the seven output classes.

3.1.3.2. Model Compilation and Training

The data was split into training and test sets using the `train_test_split` function, with an 80:20 split, meaning the training data is 80% and the test data is 20% of the original data. This ratio was chosen to ensure that a significant amount of data was used to learn the underlying patterns, while still leaving enough data for testing. To maintain reproducibility of results over multiple runs, the `random_state` parameter was set to 1.

During training, the model used an exponential decay learning rate strategy. Starting with an initial learning rate of 0.0001, it systematically decreases the learning rate at a decay rate of 0.9 every 1000 steps. This adaptive learning rate strategy promotes an efficient optimisation process, allowing faster convergence at the beginning of training when learning rates are higher, and promoting stability towards the end

of training by lowering the learning rate, thereby improving the overall performance of the model.

The Adam optimiser, known for its efficiency and balance between speed and quality of convergence, was selected for training. It was set with a `clipvalue` of 1.0, a measure introduced to limit the size of the gradient values and prevent the common problem of gradient explosion, which could potentially destabilise the learning process.

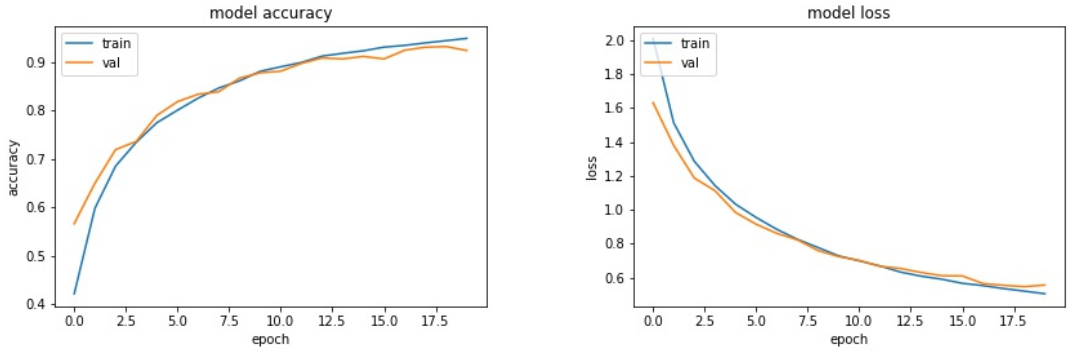
The model incorporated `L2` regularisation, a popular technique used to prevent overfitting by adding a penalty proportional to the square of the size of the weights. This encourages the model to favour smaller and more distributed weight values, allowing the model to learn simpler, more generalisable patterns from the data.

A `ModelCheckpoint` callback was defined to continuously monitor the model's performance on the validation data during training. This mechanism, set to save the model with the highest validation accuracy, ensures that the best model observed during training is retained, providing an additional level of assurance against overfitting.

For the actual training, the `fit` function was used with the input set to the training data and corresponding targets. The model was trained for 20 epochs, each epoch representing a complete pass through the entire training data set. The weights of the model were updated after each batch of 64 samples, a size chosen to strike a balance between computational efficiency and accuracy of gradient estimation.

A crucial aspect of the training strategy was the inclusion of a validation split of 0.2. This means that 20% of the training data was set aside, unseen during training, and used solely to track the model's performance on non-training data, thus aiding in the early detection of overfitting and the tuning of model parameters.

The model has a significant learning capacity, with a total of 2,083,463 parameters. This large parameter space demonstrates the model's ability to learn and capture complex patterns in the data.



(a) The training (blue) and validation (orange) accuracy of the model, showing a consistent increase over 20 epochs.

(b) The training (blue) and validation (orange) loss of the model, showing a steady decrease over 20 epochs.

Figure 3.5.: Evolution of model accuracy and loss over 20 epochs, indicating effective learning without significant overfitting or underfitting.

3.1.3.3. Model Evaluation

The generalisability of the model, i.e. its ability to perform well on unseen data, was assessed on the validation set. This gave a relatively low loss of 0.4520 and an high accuracy of 0.9595. The validation performance metrics are shown as the orange lines in Figures 3.5a and 3.5b. In contrast, the performance on the training set is represented by the blue lines in the same figures.

The model's high accuracy and relatively low loss suggest successful learning and reliable prediction of the different classes. The generalisation ability of the model, as demonstrated by its performance on the validation set, suggests its potential to provide reliable and interpretable explanations when subjected to XAI algorithms. Therefore, this model provides a robust foundation for future efforts aimed at improving the interpretability of AI models in the context of skin cancer classification.

The performance of the model is further highlighted by the confusion matrix shown in Figure 3.6, where the diagonal elements represent correct predictions for each class, demonstrating the efficiency of the model in classifying different types of lesions. It is worth noting that classes 0, 1, 3 and 5 were predicted with no misclassifications.

Classes 2 and 4 have a handful of misclassifications, especially with each other and

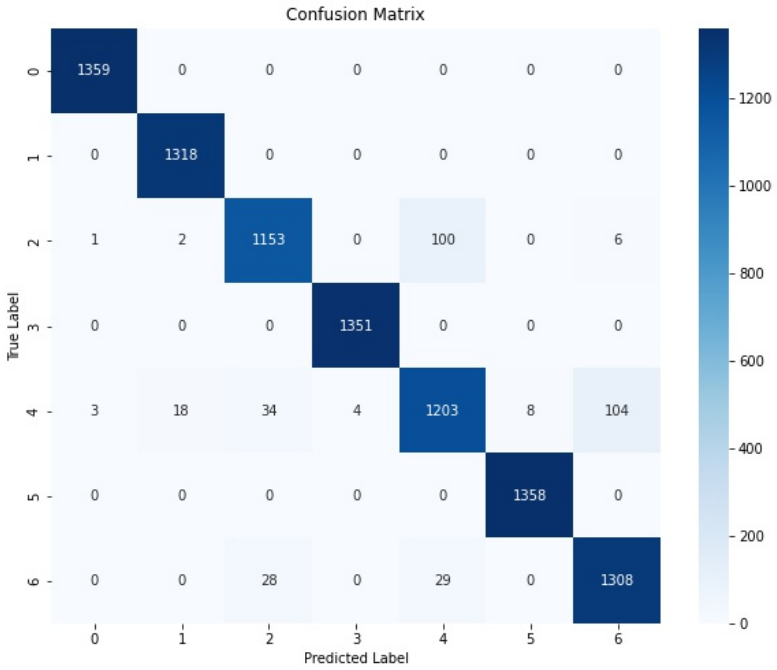


Figure 3.6.: Confusion matrix illustrating the model's predictions. The diagonal cells correspond to correctly predicted cases for each class, indicating a good prediction performance by the model.

with class 6. Although class 6 has the highest number of misclassifications with class 4, the overall performance of the model remains solid.

The classification report shown in Figure 3.7 further confirms the strong performance of the model.

The precision, recall and F1-score metrics are consistently high across all classes. The model achieves near perfect precision and recall for classes 0, 1, 3 and 5, each accompanied by an F1-score of at least 0.99. Classes 2 and 6 also show relatively good results, with F1-scores of 0.93 and 0.94 respectively. Despite the slightly lower F1-score of 0.89 for Class 4, the overall performance is still remarkable.

This consistent and high performance across different classes strengthens the case for the model's effectiveness and confirms its suitability for testing different XAI algorithms, demonstrating reliable predictions across different disease classifications.

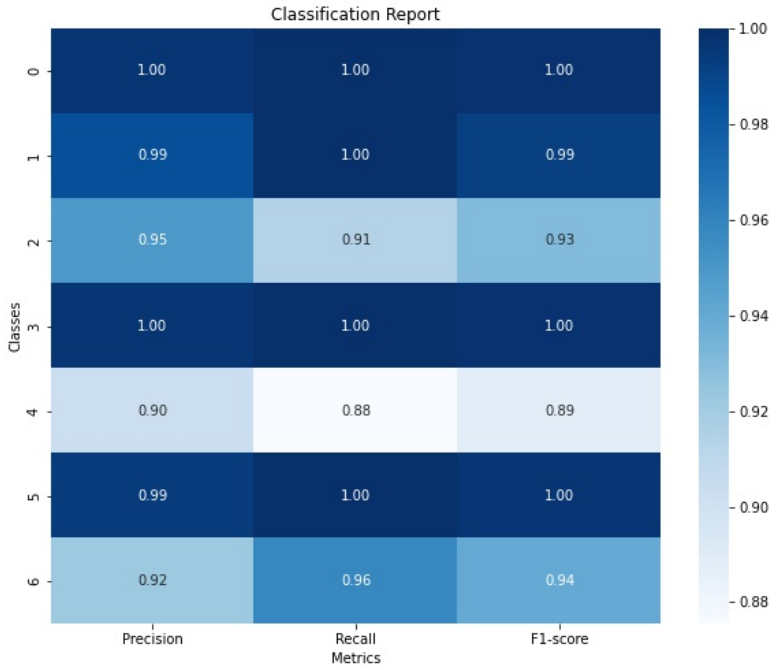


Figure 3.7.: Classification report showing the model's performance metrics - recall, precision, and F1-score for each class. The depicted results reflect the model's effective performance across different classes.

The visualisation in Figure 3.8 shows a randomly selected sample of 25 predictions made by the model, all of which were correctly classified. Each cell in the grid represents a single prediction, with the associated image, predicted and true class. This figure provides a concrete visual understanding of the model's performance, illustrating its accuracy and ability to correctly classify instances.

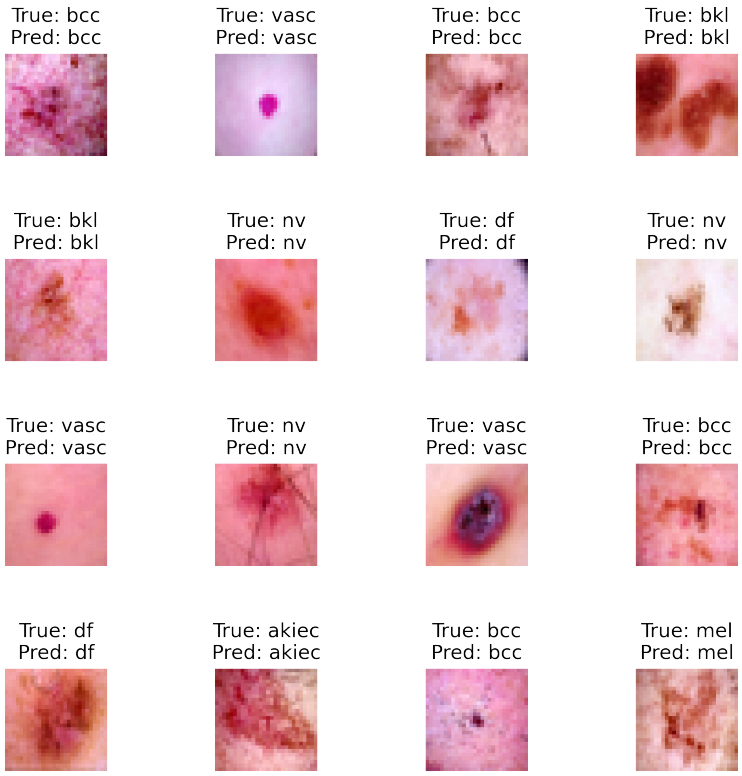


Figure 3.8.: Grid display of the model's predictions for 25 randomly selected images from the test data set. The model accurately predicted the correct class for every single image, demonstrating its strong performance on diverse instances.

3.2. Implementation of existing XAI Algorithms

The following subsections detail the implementation of XAI algorithms provided by the AIX360 toolkit. These algorithms cover both data and model explanation.

3.2.1. Data Explanation

This section shows the implementation of two XAI algorithms, ProtoDash and DIP-VAE, to gain insight into the data set and its images.

3.2.1.1. ProtoDash Implementation

The ProtoDash implementation (Appendix A.2.2) allows the identification of influential prototype instances for each class within the data set. The process starts with data preparation, where the data set is read and split into features and labels. The feature data is then normalised using sklearn's `StandardScaler` for more effective interpretation by the ProtoDash Explainer.

The next step is to select instances. The core of the ProtoDash approach is to iterate through each unique class label in the data set. For each unique label, the corresponding instances are isolated and stored in a variable called `label_data`. A representative instance for the current class label is selected from `label_data`. This instance is transformed and stored in `instance_to_explain`.

After selecting the instance, a ProtoDash explainer is initialised. This is done using the `ProtodashExplainer()` constructor. The explainer's `explain` function is called, which in turn identifies the top ten (`m=10`) prototype instances from `label_data`. These instances have the highest weights in terms of `instance_to_explain`.

The process continues by mapping the indices of the selected prototypes within `label_data` back to their original positions in the whole data set. This mapping results in a variable called `selected_indices`. This is an essential step in maintaining the context and traceability of the prototypes within the original data set.

In the data collection phase, each set of `selected_indices` is collected into a tuple along with their respective weights and current labels. This tuple is then appended to the list of `label_prototype_indices`. This sequence is repeated for each of the seven unique label within the data set.

Finally, the identified influential prototypes for each class are visualised. The images, labels and weights of selected prototypes are retrieved and a grid structure is created to display each prototype, its associated class, ID and weight.

3.2.1.2. DIP-VAE Implementation

The DIP-VAE implementation (Appendix A.2.3) starts with the preparation of the HAM10000 skin cancer dataset. The `ISICDataset` object, which encapsulates the

skin cancer data and its dimensions, is instantiated with the necessary arguments, including `root_images_path`, `file_path_labels` and `batch_size`.

A dictionary is created for setting up the model hyperparameters for both the DIP-VAE and the baseline VAE models. Attributes such as `activation_type`, `num_filters`, `latent_dim`, `num_channels`, `image_size` and `step_size`, among others, are defined within these dictionaries.

In the case of the DIP-VAE, the hyperparameters `lambda_diag_factor` and `lambda_offdiag` are carefully set; these control the regularisation of the DIP-VAE, while for the baseline VAE they are set to zero, resulting in a less constrained latent space.

The next step is to load the pre-trained model parameters. The function `load_trained_model` is used to create the path from which the pre-trained model parameters are loaded, using the previously defined model hyperparameters.

Using the loaded models and the `ISICDataset` object that loads the HAM10000 data set, `DIPVAEExplainer` instances are then instantiated for both the DIP-VAE and VAE models. These explainers perform model reconstructions and latent traversals for individual instances in the data set, visualising the behaviour of the model.

In the next step, the `extract_latens` function is used to derive the latent representations from the DIP-VAE model. Each of these latent representations, together with its corresponding label, forms a class-specific distribution of latent values.

Finally, the process of identifying influential features for each class in the dataset is carried out. This is achieved by examining the distribution of latent values of each class, which is visualised to show the separation or overlap of classes in latent space. These distributions, along with the corresponding labels, are stored in a structured format for further analysis.

It was decided to work with the original high-resolution images from the HAM10000 data set, rather than the pixel values from the CSV file, which gives a better insight into the characteristics of the different types of skin cancer represented in the data set. However, this is only the case for the DIP-VAE algorithm as it is designed to work with image data.

3.2.2. Model Explanation

This section demonstrates the implementation of three XAI algorithms, namely LIME, SHAP and CEM, to gain insight into the model's decision-making process. Each of these algorithms provides a different perspective on the model output, contributing to a comprehensive understanding of how the model works.

3.2.2.1. LIME Implementation

The first step in the implementation (Appendix A.2.4) is to create an instance of `LimeImageExplainer()`. This object is used to explain the predictions made by the model on the test images.

Next, a specific test image is selected for the explanation process. This is done by indexing the `X_test` array which contains all the test images. The index of the image to be explained is set to 0, i.e. the first image in the test set is selected, but this index can be changed to any image in the range of the test data.

Then the actual label of the selected test image is retrieved from `Y_test` and stored in the variable `actual_label`. The `predicted_label` is obtained by applying the `predict_fn` function (which represents the prediction model) to the `test_image`. The output of this function is a probability distribution over all the classes, and `np.argmax` is used to select the index of the maximum value corresponding to the most likely class according to the model's prediction.

Finally, the `explain_instance` function of the explainer object is called to generate the explanation for the `test_image`. The function takes several arguments: the `test_image`, the prediction function (`predict_fn`), the number of top labels to consider in the explanation (`num_classes`), which is seven), a colour to hide in the explanation (`hide_colour`), and the number of samples to use to generate the explanation (`num_samples`). The generated explanation is stored in the explanation variable and will be visualised later.

3.2.2.2. SHAP Implementation

In the implementation of the SHAP algorithm (Appendix A.2.5), a set of background examples is selected from the training data, `X_train`. These examples, stored in `background`, are randomly selected without replacement and serve as the basis for the expectation calculation.

The `DeepExplainer` is then constructed using the trained model and the selected background instances as inputs. This `DeepExplainer`, stored in `e`, is used to explain the model's predictions using SHAP values.

To compute the SHAP values for a particular `test_image`, the `shap_values` function of `e` is called. This function returns the SHAP values for `test_image` that describe the contribution of each feature to the prediction. Finally, the images are visualised.

This implementation of SHAP allows the predictions of the model to be interpreted at an instance level, providing insight into the individual contributions of features.

3.2.2.3. CEM Implementation

The implementation of the CEM algorithm (Appendix A.2.6) begins by selecting a particular `test_image` to explain, in a similar way to the previous explanation algorithms.

The first step in the CEM algorithm is to create an instance of the `CEM` explainer object. This instance, stored in `cem`, is initialised with the trained prediction model (`model`), the desired output shape (`shape`), and the number of possible classes (`num_classes`).

Next, the prediction of the model for `test_image` is calculated using the `predict_fn` function and the result is stored in `predicted_label`. This is the label that will be used as the target for the CEM explanation.

Then the `explain` function of the `cem` object is called, with the test image and the predicted label as inputs. The function also takes a number of optional parameters, such as `num_samples`, which controls the number of counterfactual samples to generate, `beta`, which determines the strength of the L1 regularisation term, and `max_iterations`, which sets a limit on the number of optimisation steps to take. The

generated explanation, which includes both the counterfactual and the perturbation, is stored in the variable `explanation`.

3.3. Evaluation

This section comprehensively covers the evaluation methodology adopted in this thesis, starting with the specific criteria identified for assessing the performance of the implemented XAI algorithm.

3.3.1. Evaluation Procedure

The evaluation procedure for the XAI algorithms implemented in this thesis involves testing and analysis using four different evaluation criteria. These criteria have been carefully selected to provide a comprehensive assessment of the effectiveness and reliability of each XAI algorithm. They cover a wide range of aspects including runtime, simplicity, interpretability/explainability (local vs. global) and stability.

However, not all evaluation criteria are equally applicable or measurable for every XAI algorithm. The suitability of each criterion may depend on several factors, including the nature of the algorithm, the specific design goals it was intended to achieve, and the problem context in which it is used.

It is important to clarify that the faithfulness and monotonicity metrics from the AIX360 toolkit were not used in the evaluation process. Despite being potentially valuable metrics, their application to the skin cancer classifiers proved challenging due to insufficient documentation. Attempts to apply these metrics encountered difficulties, resulting in unreliable results. Therefore, to ensure the reliability and accuracy of the evaluation, emphasis was placed on other metrics that could be more confidently applied and measured.

3.3.2. Evaluation Criteria

This subsection provides a detailed explanation of each of the evaluation criteria used in this thesis. The selection and design of these measures aim to ensure a comprehensive assessment of the XAI algorithms under evaluation.

3.3.2.1. Runtime

Evaluating the runtime of each XAI algorithm is an essential aspect of understanding its performance and efficiency.

This is done using Python's built-in `time` module, specifically its `perf_counter()` function [50]. This function provides a performance counter defined as a clock with the highest resolution available to measure a short duration.

Each algorithms `explain` function is run a hundred times to record a series of runtime measurements. From this series, the minimum, maximum, average and standard deviation of the runtimes are calculated. This provides a thorough understanding of the runtime behaviour of each algorithm, capturing the potential variability in their execution times and ultimately allowing a comparison of their performance efficiency.

3.3.2.2. Simplicity

In real-world scenarios, the simplicity of an XAI algorithm is critical, as complexity can potentially hinder usability and practical deployment. To evaluate simplicity, two key factors are considered.

The first is hyperparameter tuning, as algorithms that require fewer hyperparameters to be adjusted generally embody a greater degree of simplicity. The process of tuning a large set of hyperparameters can be both complex and time consuming.

The second factor is the complexity of the integration. The simplicity of an XAI algorithm can also be measured by the ease with which it can be integrated into existing models. Those algorithms that require minimal changes to the structure or training process of a model are considered simpler.

3.3.2.3. Interpretability/Explainability

Interpretability and explainability, as defined in Section 2.2.1.1 and 2.2.1.2, are critical metrics for measuring the understandability of explanations generated by an XAI algorithm. It is intended for a variety of audiences, including developers, domain

experts, i.e. dermatologists, and end users, and assesses the clarity, comprehensibility and relevance of the explanations. However, while interpretability may be more useful to a user, explainability is also very valuable to domain experts and developers.

The evaluation involves an in-depth analysis of the explanations provided by the XAI algorithm using sample outputs. The interpretability of these explanations is measured by the ability to understand the outputs. Explainability is measured by the way in which it helps to understand the model's decisions.

In addition to general interpretability, the type of interpretability - local versus global - is also assessed. Local interpretability refers to detailed insights for individual cases that shed light on specific data points. Global interpretability aims to provide an overarching understanding of the overall patterns or structures of the data set.

These aspects of interpretability and explainability are measured by analysing the explanations provided by the XAI algorithms output.

3.3.2.4. Stability

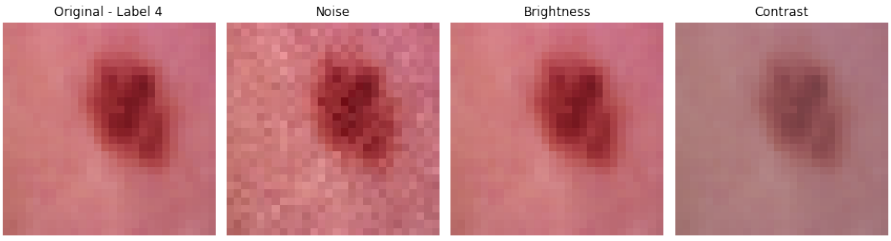


Figure 3.9.: A randomly selected image to illustrate the data augmentation techniques used to generate new images with subtle adjustments to noise, brightness and contrast.

Stability is a key attribute for any XAI algorithm, reflecting its reliability and consistency in generating explanations even when exposed to small variations in the input. To assess stability, subtle changes are made to an image, such as slight changes in brightness and contrast, or the addition of minimal noise. An illustration of such image augmentation, achieved using the `imgaug` library [51], is shown in Figure 3.9.

For each of these altered images, explanations are produced using the XAI algorithm under investigation. These explanations are then compared with those generated for the original unmodified image. A high degree of consistency in the explanations, despite the minor changes to the image, indicates robust stability for the XAI algorithm. This feature is particularly important to ensure the reliability of the explanations, especially in dynamic environments where the input data may undergo small disturbances or changes.

For algorithms that support it, quantitative measures such as percentage agreement, mean absolute difference (MAD) or correlation were calculated over a period of 20 runs. The choice of measures depended on the data provided by each algorithm, allowing a more specific and tailored assessment of their stability.

4. Results

This chapter presents the results obtained from the evaluation of the implemented XAI algorithms on the basis of the defined metrics, and also provides a brief summary.

4.1. Evaluation of the XAI Algorithms

The following sections evaluate the performance of each of the XAI algorithms, namely ProtoDash, DIP-VAE, LIME, SHAP and CEM. The evaluation will focus on the previously outlined metrics to provide a comprehensive assessment of each algorithm's performance.

4.1.1. ProtoDash Evaluation

Runtime As shown in Table 4.1, the runtime results demonstrate the computational efficiency of the ProtoDash algorithm over a number of iterations. The algorithm shows its usefulness with an average runtime of 0.3031 seconds. The recorded maximum and minimum times, 0.9356 seconds and 0.1335 seconds respectively, provide an insight into the variation of the algorithm's performance over different runs. A standard deviation of 0.2649 seconds, although relatively high considering the magnitude of the runtimes, indicates a moderate level of consistency in the algorithm's execution time across iterations. The ProtoDash algorithm thus exhibits a blend of speed and consistency, which enhances its usability in real-world applications.

Measure	**Runtime (seconds)**
Average	0.3031
Maximum	0.9356
Minimum	0.1335
Standard Deviation	0.2649

Table 4.1.: Runtime measurements for the ProtoDash algorithm over 100 iterations.

Simplicity The first factor considered, hyperparameter tuning, assesses the simplicity of an algorithm based on the number and complexity of hyperparameters that need to be tuned. ProtoDash scores remarkably well in this respect, as it only requires the tuning of a single hyperparameter, `m`. This parameter, which determines the number of prototypes to be returned, can be easily adjusted according to the specific needs of the use case, contributing significantly to the overall simplicity of the ProtoDash algorithm.

The second factor, integration complexity, reflects the ease with which an algorithm can be integrated into existing models. In this context, ProtoDash demonstrates an excellent level of simplicity. The implementation of the algorithm does not require extensive modifications to the existing model framework or training pipeline. On the contrary, it can be seamlessly applied to the model post hoc, operating primarily on the input data and predictions. This, combined with the straightforward data preparation process involving normalisation and the simple use of the algorithm itself, underlines the simplicity and ease of integration of ProtoDash.

Interpretability/Explainability The ProtoDash algorithm, with its emphasis on interpretability, is able to generate understandable explanations for a wide range of audiences, from developers to domain experts to end users.

As shown in Figure 4.1, the algorithm identifies ten prototype images for each class, which are distinguished by a spectrum of different attributes such as shape, size and colour. This diverse representation is indicative of ProtoDash's superior local interpretability. In essence, it provides detailed insights for individual cases, improving understanding at a granular level, which is an essential facet of interpretability.

This detailed output is valuable to domain experts such as dermatologists, allowing them to identify patterns and anomalies in a practical context. Similarly, developers

Figure 4.1.: Top ten images determined by ProtoDash weights for each class. The ID and corresponding weight for each image is shown below each image.

can use this detailed interpretability to improve models or algorithms by focusing on the most important features. In addition, end users without extensive technical knowledge can benefit from ProtoDash's visually appealing prototype images, which help them to understand the underlying logic behind certain decisions or classifications.

However, it's important to remember that ProtoDash is primarily designed for local interpretation. Although it is adept at revealing the specifics of individual data

points, it may not be as good at providing insight into global patterns or overarching data structures. Therefore, caution should be taken when extrapolating broad trends from these individual prototypes.

In addition, ProtoDash's weighting system contributes significantly to its interpretability. By comparing images with high and low weights, it is possible to identify the features that the model considers important, thereby clarifying the decision-making process. In addition, these weights provide an insight into class diversity, where a wider range of weights may indicate diverse prototypes within a class, while a narrower range may indicate more homogeneous characteristics.

Stability Figure 4.2 shows the application of the ProtoDash algorithm to each modified image. The algorithm identifies the images with the highest weights and uses them as explanations. Remarkably, ProtoDash selects the same images as the most significant explanations for both the original image and its brightness-adjusted counterpart, suggesting a degree of stability in response to brightness variations.

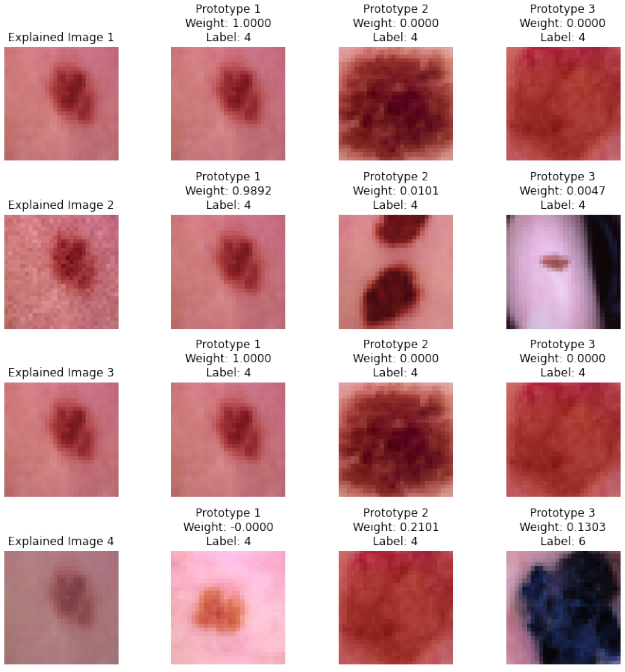


Figure 4.2.: Different iterations of a reference image, including the augmented images and their corresponding ProtoDash explanations, showing the highest similarity between the original and the brightened images.

However, when the image is subjected to other changes, such as random noise or contrast changes, ProtoDash presents a different set of images as explanations. This pattern, seen across several test images, suggests that while ProtoDash has some stability in response to brightness changes, its consistency is somewhat compromised when dealing with other types of image transformations.

In particular, the image with the highest weight in the last row after contrast adjustment is not even the original image, as that was the case for the noise-added and brightness-adjusted images, indicating the different effects these transformations can have on the ProtoDash explainer.

4.1.2. DIP-VAE Evaluation

Runtime The runtime results shown in Table 4.2 encapsulate the efficiency of the DIP-VAE algorithm as measured over multiple iterations. The average runtime of 212.6028 seconds indicates that the algorithm's performance requires a significant amount of time, while the maximum and minimum times of 214.3217 seconds and 210.5328 seconds respectively show the range of variation. The standard deviation of 1.2460 seconds shows the consistency of the running time over several iterations.

Measure	**Runtime (seconds)**
Average	212.6028
Maximum	214.3217
Minimum	210.5328
Standard Deviation	1.2460

Table 4.2.: Runtime measurements for the DIP-VAE algorithm over 100 iterations.

Simplicity The simplicity of the DIP-VAE algorithm was evaluated based on its implementation and configuration.

In terms of hyperparameters, DIP-VAE has several that need to be defined and possibly tuned. These include the number of latent variables, the type of encoder and decoder networks, and the lambda coefficients used to enforce the disentanglement of the latent variables. This list is supplemented by parameters such as `model_args`,

`data set`, `net` and `cuda_available`, which must be initialised when creating a DIP-VAE explainer. When performing latent space edits on images, parameters such as `input_images`, `edit_dim_id`, `edit_dim_value` and `edit_z_sample` must also be provided. Although DIP-VAE offers great flexibility, this multitude of hyperparameters can complicate the tuning process, which can affect the simplicity of the algorithm.

In terms of integration complexity, DIP-VAE is designed with a certain degree of flexibility. Its structure, consisting mainly of an encoder and decoder network, allows it to be integrated with different types of generative models. In addition, DIP-VAE is implemented separately from the model to be explained, keeping the explanation process separate from the model's operations. When using GPUs, the `cuda_available` parameter can be set to true to take advantage of the computational efficiency of GPU acceleration. While these features increase compatibility with various use cases, the need for specific components such as the encoder and decoder networks in the generative model may require certain changes to the model structure, potentially complicating the integration process.

Interpretability/Explainability In terms of global interpretability, DIP-VAE excels at providing detailed insight into individual data points through latent traversals, as shown in Figure 4.3. This process involves varying a single latent at a time while keeping others fixed, which helps to understand the relationship between different generative factors and the output.

The ability of DIP-VAE to uniquely capture important skin lesion characteristics, such as 'diameter' in the fifth dimension, 'border' in the zeroth dimension, and 'asymmetry' in the first dimension, is evident in these latent traversals. This very nuanced understanding allows domain experts, such as dermatologists, to make granular observations and interpretations about skin lesion characteristics.

In addition, this level of global interpretability helps developers refine their models or algorithms to focus on these significant features. By focusing on these critical features, they can improve the predictive accuracy and performance of their models, thereby increasing the utility of DIP-VAE in skin cancer diagnosis.

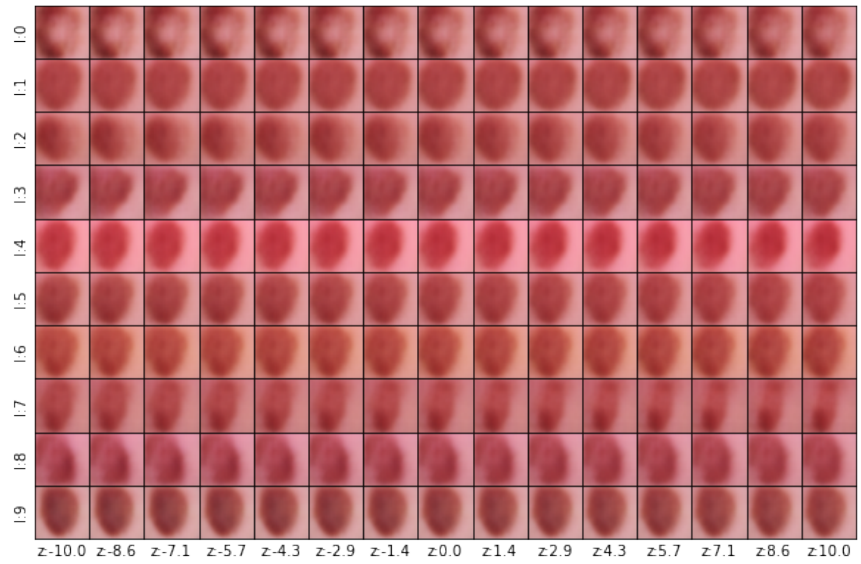


Figure 4.3.: DIP-VAE latent traversal derived from the decoder's output.

Figure 4.4 illustrates the global interpretability of DIP-VAE, using box plots to highlight recognisable patterns of latents across different skin lesion classes. Each vertical line represents a dimension, like the y-axis in Figure 4.3. For example, the first dimension, which seems to capture the 'asymmetry' trait, shows that classes such as 'mel' and 'bcc' show lower activation, whereas classes such as 'akiec' and 'bkl' show higher average activation, suggesting a greater sense of asymmetry.

These visualisations and interpretations can help end users, who may not be immersed in the technical intricacies, to gain a holistic understanding of the different lesion classes. Therefore, these insights enhance the interpretability of DIP-VAE and promote a more comprehensive understanding of the model's decisions in skin cancer diagnosis.

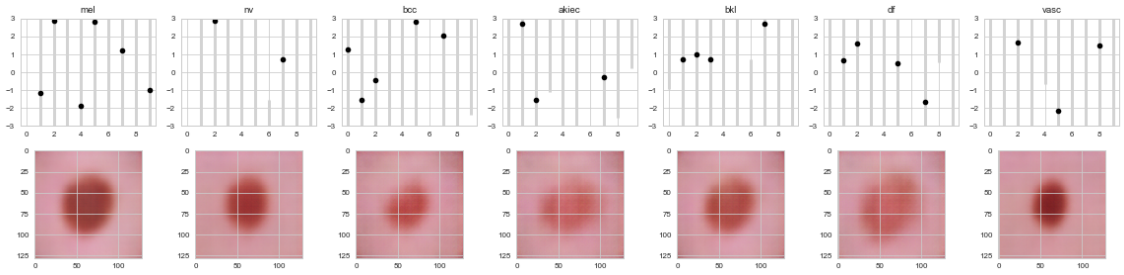


Figure 4.4.: Patterns of latents across classes visualised using DIP-VAE.

When considering metrics of interpretability/explainability, DIP-VAE excels in providing clear, pertinent and meaningful explanations. The relevance of these insights, aligned with the understanding of each stakeholder - developers, domain experts and end users - not only makes the explanations understandable, but also validates the interpretability/explainability of DIP-VAE in the context of skin cancer diagnosis.

Stability In the context of this thesis, a stability assessment for DIP-VAE was not undertaken. As the DIP-VAE model was used primarily for its global interpretability capabilities, a measure of stability is not readily applicable or particularly meaningful. Therefore, an assessment of stability, which typically involves the generation of explanations for subtly modified versions of the original images, was not performed for DIP-VAE within the context of this thesis.

4.1.3. LIME Evaluation

Runtime The Table 4.3 presents the measures of the runtime of the LIME algorithm, which provides an insight into its computational efficiency over different iterations. An average runtime of 0.9887 seconds indicates a commendable speed of operation, demonstrating its effectiveness in delivering timely results. The recorded maximum and minimum runtimes are 1.1476 seconds and 0.9223 seconds respectively, indicating the range of its performance in terms of speed across different executions. With a standard deviation of 0.0505 seconds, the algorithm shows a remarkable consistency in its running time, which reinforces its reliability in delivering timely results over multiple runs.

Measure	**Runtime (seconds)**
Average	0.9887
Maximum	1.1476
Minimum	0.9223
Standard Deviation	0.0505

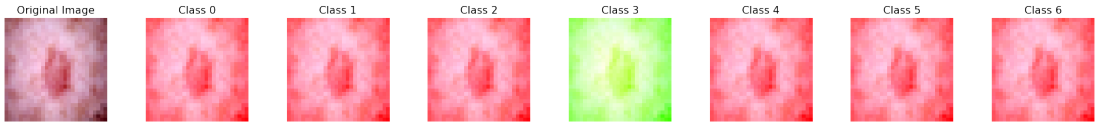
Table 4.3.: Runtime measurements for the LIME algorithm over 100 iterations.

Simplicity The simplicity of the LIME algorithm was evaluated by observing its implementation and configuration.

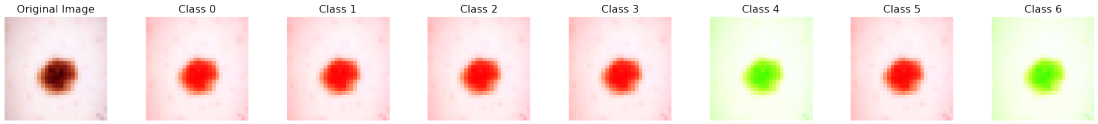
In terms of hyperparameter tuning, LIME presents a simple interface with only two primary hyperparameters to adjust: `num_samples` and `num_features`. The `num_samples` parameter dictates the number of samples to be used to build the local surrogate model, while `num_features` controls the number of interpretable features to be presented in the explanation. Both parameters can be easily adjusted to match the complexity of the model and the level of detail required for the explanation, contributing to the overall simplicity of the algorithm.

In terms of integration complexity, LIME has a high degree of simplicity due to its model agnostic nature. It can be seamlessly integrated into any ML model without requiring changes to the model architecture or training procedures. As shown in the provided implementation, LIME is applied directly to the prediction function of the trained model (`model.predict`), keeping the explanation generation process separate from the model's own operations. This separation facilitates the easy integration of LIME, thereby increasing its overall simplicity. Thus, LIME stands out as an accessible and adaptable tool in the field of XAI.

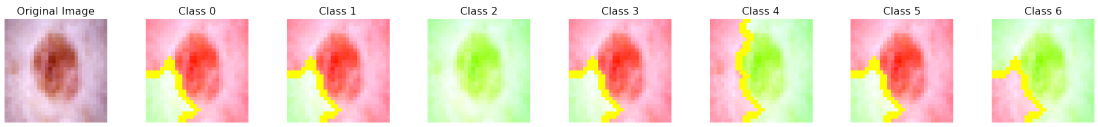
Interpretability/Explainability The utility and effectiveness of the LIME algorithm lies in its ability to provide interpretable and explainable results. For developers, domain experts and end users alike, understanding the reasoning behind the model's decisions is paramount. LIME addresses this need through its visually oriented local interpretability, as shown in Figure 4.5. Each image dissects the model's responses to specific inputs, highlighting influential areas within the image that led to a particular classification.



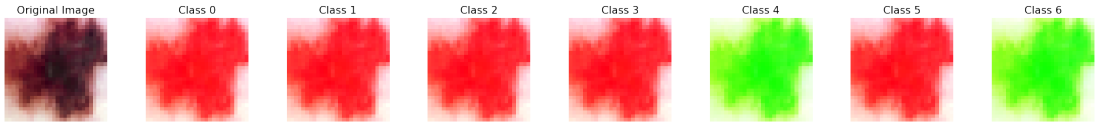
(a) LIME explanation for an instance with congruent true and predicted classes of 3. The highlighted regions indicate class-relevant areas, demonstrating the local interpretability of LIME.



(b) LIME explanation for an instance with both true and predicted classes as 4. The explanation distributes the areas of influence evenly across two classes, demonstrating LIME's ability to handle nuanced scenarios.



(c) LIME explanation for an instance where the true and predicted class are 2. The presentation of both positive and negative class contributions illustrates LIME's ability to distinguish between influential and non-influential regions.



(d) LIME explanation for an instance with a true class of 6 and a predicted class of 4. Equally marked classes suggest a closely contested decision, providing insight into why the wrong class was chosen.

Figure 4.5.: Various examples of LIME explanations showing its diversity. The highlighted red and green areas represent positive and negative contributions respectively, demonstrating how different regions of the input influence the prediction.

This local interpretability provides an valuable perspective for domain experts such as dermatologists, enabling them to examine individual cases, patterns and potential anomalies within skin lesions. The feature-wise explanations provided by LIME can also guide developers in refining their models or algorithms, allowing them to place more emphasis on certain influential regions. At the same time, for non-technical end users, the visual highlighting of significant image regions will enhance their understanding of the rationale behind certain predictions.

The various LIME explanations in Figure 4.5 also highlight the adaptability of the algorithm in dealing with different prediction scenarios. For example, Figure 4.5a shows how LIME skilfully illuminates influential areas when the true and predicted classes match. On the other hand, Figure 4.5b shows a situation where the model considers two different classes equally, demonstrating LIME's ability to capture such nuances. Furthermore, Figure 4.5c illustrates LIME's ability to distinguish between influential and non-influential regions within the same image, regardless of class.

Figure 4.5d shows a LIME explanation for a case where the actual class is 6, but the predicted class is 4. Despite the discrepancy in classification, LIME's explanation shows that the decision for the incorrect class was equivalent to the correct one, as both classes are equally emphasised in the image. This demonstrates LIME's ability to provide insight into cases of incorrect predictions, proving to be a useful tool for debugging models and understanding misclassifications.

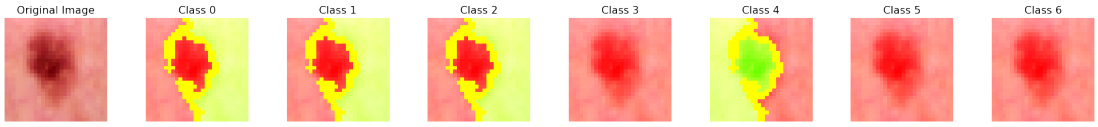
However, while LIME excels at providing detailed local interpretations, it is not explicitly designed to provide global interpretations, i.e. broader trends or patterns across the data set. It is important to interpret these local explanations with caution when attempting to infer global trends or generalised behaviour of the model.

In terms of interpretability and explainability, LIME excels at making the AI's decision-making process more transparent and relatable. Its local interpretability and explainability are critical in scenarios where understanding and trusting the decisions of the AI system are paramount.

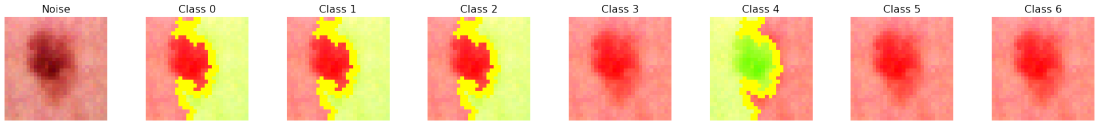
Stability The stability of LIME's explanations is assessed by applying it to various modifications of a single image, as shown in Figure 4.6. This includes the original image and its modified versions subjected to noise, brightness and contrast adjustments. Remarkably, all of these images were accurately classified as Class 4.

The consistency of LIME's explanations across the image transformations was assessed using binary masks. In these masks, each pixel was assigned a binary value of either 1 or -1. Pixels labelled '1' represented significance in the class prediction as determined by LIME, while '-1' represented insignificance.

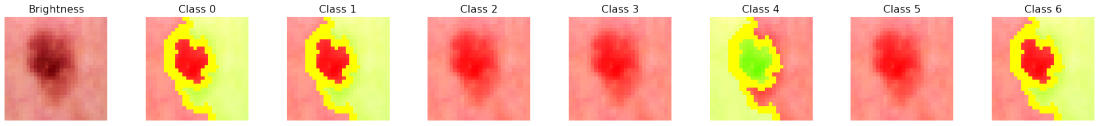
The percentages given represent the degree of agreement between the original im-



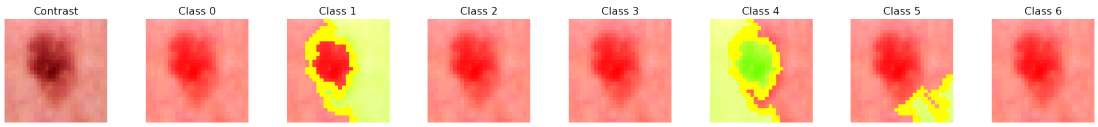
(a) Original image with its LIME explanation.



(b) Noise image and corresponding LIME explanation.



(c) Brightness-adjusted image and corresponding LIME explanation.



(d) Contrast-adjusted image and corresponding LIME explanation.

Figure 4.6.: Different iterations of an image and their respective LIME explanations, showing the highest similarity between the original and the noise corrected images.

age and its modified versions. These were calculated over 20 iterations and provide a snapshot of the stability of the explanation, although more extensive testing could provide a fuller understanding. The percent match for the original and noise-adjusted image was 85.02%, 83.02% for the brightness-adjusted image, and 83.62% for the contrast-adjusted image.

This measure was chosen instead of correlation coefficients because the binary nature of the masks (consisting only of -1 and 1) made the calculation of correlations impractical. Instead, the match percentage serves as an indicator of the degree of agreement between the pixel importance assigned by LIME in the original and modified images.

4.1.4. SHAP Evaluation

Runtime The runtime performance of the SHAP algorithm is shown in Table 4.4. Over multiple iterations, the algorithm produced an average runtime of 3.6854 seconds, reflecting its reasonably efficient computational performance. The observed maximum and minimum runtimes are 8.2580 and 3.3072 seconds respectively, indicating potential variation in execution time across instances. A standard deviation of 0.5357 seconds underscores this variability, highlighting the presence of a degree of inconsistency in execution time across successive iterations. This suggests that the performance of the SHAP algorithm may vary somewhat depending on specific input characteristics or computational circumstances.

Measure	**Runtime (seconds)**
Average	3.6854
Maximum	8.2580
Minimum	3.3072
Standard Deviation	0.5357

Table 4.4.: Runtime measurements for the SHAP algorithm over 100 iterations.

Simplicity When it comes to hyperparameter tuning, SHAP stands out for its simplicity, with only a minimal set of user-defined parameters. The most important configuration step is the selection of a set of background samples from which to compute expectations. This is done by passing a subset of training samples to the `DeepExplainer` constructor. In the given implementation, this subset was randomly selected from the training set. The rest of the SHAP procedure, including the generation of explanations for specific instances, does not require any additional hyperparameter tuning. This feature makes SHAP a user-friendly explanation generation tool, especially for users who may not be highly skilled in hyperparameter optimisation.

In terms of integration complexity, SHAP shines with its inherent simplicity, largely due to its model-agnostic design. SHAP can be seamlessly integrated into any ML model, avoiding the need to adapt the model's architecture or training process. In the demonstrated implementation, the `DeepExplainer` is initialised with the pre-trained model and a set of background examples. An explanation for a given test

image is generated by calling the `shap_values` function on the explainer object, effectively bypassing any interaction with the model's internal operations. This separation between the explanation process and the model's mechanics enhances SHAP's ease of integration and contributes to its overall simplicity.

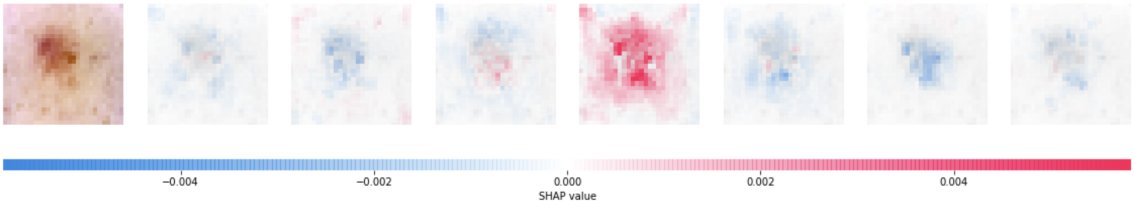
In addition, SHAP includes a user-friendly visualisation function, `shap.image_plot`, which simplifies the interpretation of the generated explanations. This function deftly manages the intricacies of visualising feature contributions, adding another layer of ease to the use of SHAP.

Interpretability/Explainability The interpretability and explainability metrics, which are central to assessing the effectiveness of any XAI algorithm, measure the level of understanding associated with the algorithm's explanations. Aimed at a variety of stakeholders, including developers, domain experts such as dermatologists, and end users, these metrics emphasise the clarity, comprehensibility and relevance of the explanations.

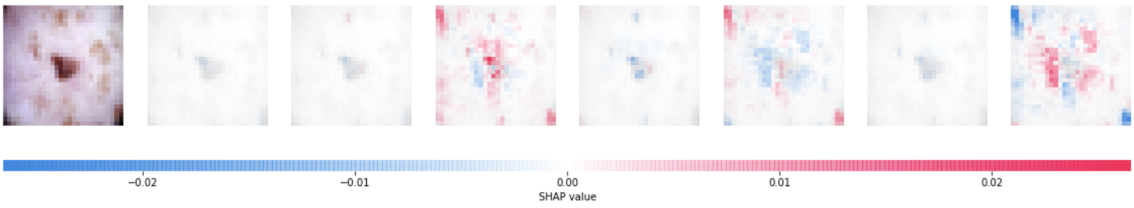
SHAP stands out as an XAI algorithm that provides clear and understandable explanations, as shown in Figure 4.7. Each visual representation explains the model's decisions by highlighting specific regions of the image that influence the predicted class. This algorithm is visible in the colour coding scheme, where red indicates a positive contribution, pushing the model output towards the predicted class, while blue indicates a negative influence.

This detailed, colour-coded mapping provides a balanced blend of local and global interpretability. It provides nuanced insight into individual cases, helping domain experts such as dermatologists to identify patterns or anomalies in skin lesions. At the same time, it provides an overview of the general behaviour of the model, which developers can use to prioritise influential features when optimising models or algorithms. The visual representation also simplifies the logic behind the model's decisions for end users.

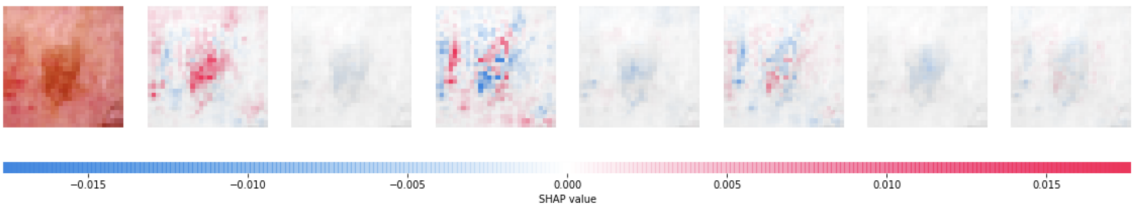
The range of SHAP explanations shown in Figure 4.7 underlines the adaptability of the algorithm to different prediction scenarios. Figure 4.7a shows SHAP's ability to highlight areas of influence when the true and predicted classes match. In contrast, Figure 4.7b demonstrates SHAP's ability to handle complex situations where the



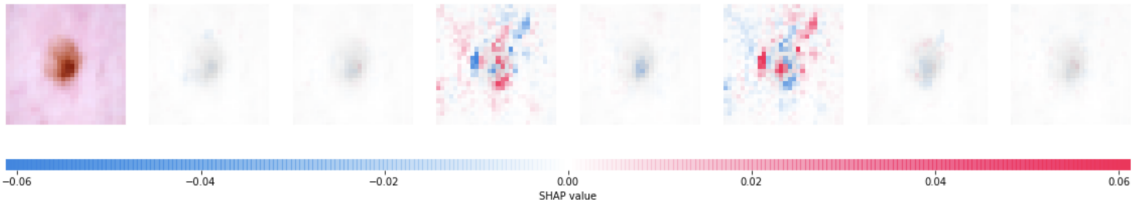
(a) A SHAP explanation showing an instance where the true and predicted classes align as 3. This visualisation conveys a compelling and clear influence from a specific class, highlighting the transparency of SHAP in local interpretability.



(b) A SHAP explanation for an instance where the true and predicted classes are both 2. Despite the congruent class, the explanation unravels comparable influence from different classes, demonstrating SHAP's ability to handle nuanced situations and balance between local and global interpretability.



(c) A SHAP explanation for an instance where both the true and predicted class are 0. This visualisation highlights the positive and negative contributions of each class, signifying SHAP's ability to quantify the impact of features on the model's decision process.



(d) A SHAP explanation for an instance where the true class is 2, but the predicted class is 4. The visualisation outlines the similarities between the two classes, providing insight into why the model incorrectly predicted the class.

Figure 4.7.: Various examples of SHAP explanations showing its diversity. The color-coded regions, with red indicating positive SHAP values and blue representing negative SHAP values, depict how distinct regions of the input contribute differently to the model's prediction.

model considers two different classes equally. In addition, Figure 4.7c highlights SHAP's unique ability to quantify the impact of features, showing both positive (red) and negative (blue) contributions to each class.

Figure 4.7d provides a SHAP explanation for a case where the model misclassifies the true class as 4 instead of 2. Despite this misclassification, SHAP insightfully reveals the closely contested decision between the two classes, providing valuable understanding for troubleshooting such misclassifications.

Unlike LIME, which primarily provides localised interpretability, SHAP excels at providing a more comprehensive view. It can provide aggregate interpretations, giving a broad perspective on how features collectively affect the model across multiple instances. However, it's important to avoid over-generalising these insights.

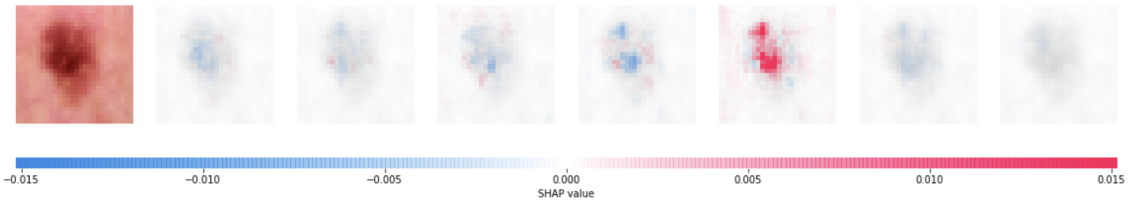
In terms of interpretability and explainability, SHAP makes a significant contribution to making AI decision making more transparent, which is helpful in contexts where understanding and building trust in AI systems is essential.

Stability The stability of the SHAP algorithm's explanations was assessed across a series of modifications applied to a single image, as shown in Figure 4.8. This analysis included the original image and its variants that underwent modifications such as the addition of noise, brightness adjustment and contrast modification.

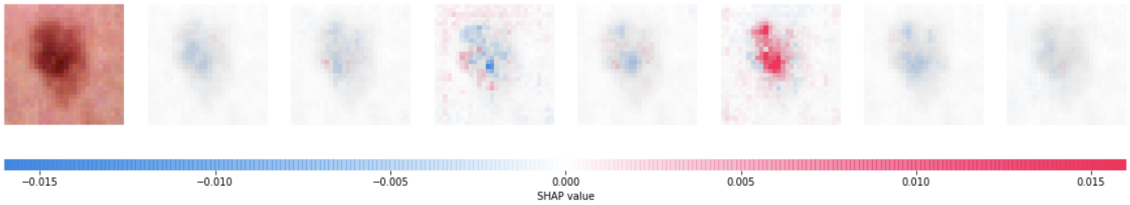
The Mean Absolute Difference (MAD) and correlations were used as key metrics to measure the stability of the SHAP explanations amidst these image transformations. These metrics were calculated over 20 iterations using the SHAP values derived from the SHAP algorithm. While this provides an initial understanding of the stability of the SHAP explanations, a larger number of iterations may provide a more complete perspective.

The choice of MAD as a metric stems from its ability to effectively measure the average size of the differences between two numerical collections, in this case the pixel-level explanation values across the original and transformed images. Correlation, on the other hand, represents the strength and direction of the relationship between these pixel-level explanations.

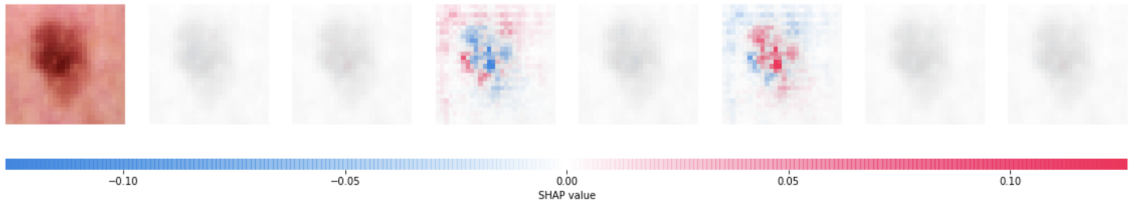
For the noise corrected images, the MAD was 0.000158 with correlations ranging



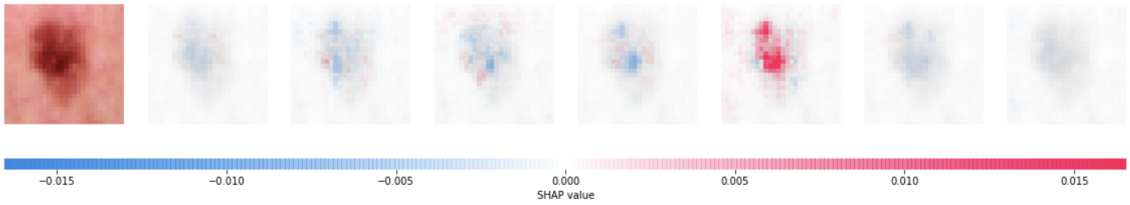
(a) Original image with its SHAP explanation.



(b) Noise image and corresponding SHAP explanation.



(c) Brightness-adjusted image and corresponding SHAP explanation.



(d) Contrast-adjusted image and corresponding SHAP explanation.

Figure 4.8.: Different iterations of an image and their respective SHAP explanations, showing the highest similarity between the original and the contrast-adjusted images.

from 0.836 to 0.947. Brightness-adjusted images had a higher MAD of 0.002088, with correlations ranging from 0.517 to 0.926. Contrast-adjusted images, however, showed the lowest MAD at 0.000119 and correlations ranging from 0.799 to 0.978, indicating a remarkable degree of stability in the explanations of SHAP.

4.1.5. CEM Evaluation

Runtime Table 4.5 shows the runtime results of the CEM algorithm for both Pertinent Positive (PP) and Pertinent Negative (PN) over several iterations.

For the PP function, it shows an average runtime of 361.51 seconds, indicating a high computational cost. The maximum and minimum times recorded are 923.53 seconds and 292.22 seconds respectively, indicating a high degree of variability in the performance of the algorithm over different runs. A standard deviation of 132.50 seconds further illustrates this variability, implying significant inconsistency in running time across iterations.

Measure	**PP Runtime (seconds)**	**PN Runtime (seconds)**
Average	361.5100	356.3295
Maximum	923.5313	869.1436
Minimum	292.2203	289.3256
Standard Deviation	132.4979	126.3662

Table 4.5.: Runtime measurements for the CEM algorithm (PP and PN) over 100 iterations.

The PN function shows similar results with an average run time of 356.33 seconds. The maximum and minimum runtimes for PN are 869.14 seconds and 289.33 seconds respectively. The standard deviation of 126.37 seconds indicates a similar level of variation in run time to the PP function.

These results indicate that the CEM algorithm, both PP and PN, require significant computational resources and that there is considerable variability in run time performance between different instances.

Simplicity The simplicity of the CEM was investigated by evaluating its configuration and implementation process.

In terms of hyperparameter tuning, CEM requires a more complex setup. It involves a considerable number of hyperparameters, including `arg_mode`, `arg_max_iter`, `arg_init_const`, `arg_b`, `arg_kappa`, `arg_beta`, `arg_gamma`, `arg_alpha`, `arg_threshold` and `arg_offset`. Each of these hyperparameters controls a specific aspect of the explanation generation process. While this large set of hyperparameters allows for a high

degree of customisation and flexibility, it also makes the tuning process challenging and potentially time-consuming. Furthermore, fine-tuning these parameters requires an understanding of their respective roles in the CEM algorithm, adding another layer of complexity.

In addition to these hyperparameters, CEM also includes an optional component where an autoencoder can be trained to further refine the explanation process. Although this option increases the adaptability of the algorithm, it may increase the complexity of the preparation phase, as it requires separate training and optimisation.

In terms of integration complexity, CEM shows moderate simplicity due to its design, which operates separately from the original model. It does not require any changes to the structure of the model or the training process. As a result, CEM can be integrated with any pre-existing model without changing the model's architecture. This separation of the explanation generation process from the model's own operations ensures a straightforward and manageable integration process.

Interpretability/Explainability In the case of the CEM algorithm applied to the skin cancer data set, the generation of meaningful explanations was unsuccessful, as shown in Figure 4.9. The PP should have provided the minimum set of pixels present in the image to classify it as class 4, and the PN should have shown the pixels whose presence would change the classification of the original image to class 2. This lack of comprehensible explanations undermined the local interpretability of the algorithm, as critical aspects such as clarity and task relevance could not be properly defined.

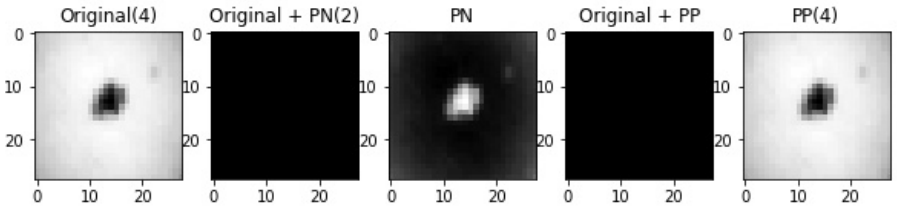


Figure 4.9.: An instance of an unsuccessful CEM explanation.

This inability of CEM to produce individual, localized explanations also adversely

affected its global interpretability. Without sufficient understanding of individual cases, discerning overarching patterns or structures within the data set became challenging. Thus, the overall behavior of the data, from a global perspective, remained unclear.

The unsuccessful generation of explanations by CEM could be attributed to several factors. The complexity of the skin cancer data set and model, the need for fine-tuning the CEM hyperparameters, and possible inherent limitations of the CEM algorithm when applied to this specific context could all contribute to this outcome.

Repeated attempts to adjust the hyperparameters failed to yield valuable explanations from the CEM algorithm, thus limiting its interpretability and explainability in this context. Consequently, with regards to the skin cancer data set, the CEM's performance in terms of interpretability and explainability was limited.

Stability In the case of the CEM algorithm used in this thesis, stability could not be effectively evaluated due to the lack of meaningful explanations for the original skin cancer images. Without a reference point in the form of an understandable explanation for an original image, there was no baseline to compare with possible explanations for slightly modified images. Therefore, the stability of the CEM in response to small variations such as noise, brightness and contrast adjustments could not be adequately assessed.

This limitation highlights the importance of generating meaningful explanations as a prerequisite for assessing other characteristics of an XAI algorithm, such as stability. Without an explanation for the original, unmodified image, assessing how this explanation might change with small variations in the input becomes a challenging task.

4.2. Summary

This thesis evaluated five XAI algorithms, ProtoDash, DIP-VAE, LIME, SHAP, and the CEM algorithm on four key metrics: runtime, simplicity, interpretability/-explainability and stability.

ProtoDash showed high efficiency with an average run time of 0.3031 seconds. It excelled in simplicity due to a single hyperparameter for tuning and easy model integration. Although ProtoDash had excellent interpretability/explainability and stability against brightness changes, it struggled with other image transformations.

DIP-VAE required significantly more computational resources, as reflected in its average run time of 212.6028 seconds. While multiple hyperparameters added complexity, it retained flexibility for integration. The interpretability/explainability of DIP-VAE was robust, providing both local and global insights. However, its stability was not assessed.

LIME demonstrated efficiency with an average runtime of 0.9887 seconds, and simplicity through its minimal hyperparameters and model-agnostic nature. Its interpretability/explainability was strong, producing detailed, visual and localised explanations. LIME also showed good stability over different image transformations.

SHAP offered efficient computation with an average running time of 3.68 seconds. Its simplicity was evidenced by minimal user-defined hyperparameters and easy model integration. SHAP's interpretability/explainability was robust, balancing local and global insights. It also showed high stability, especially for contrast-adjusted images.

CEM, on the other hand, had long runtimes, averaging 361.51 seconds for PP and 356.33 seconds for PN. Despite its complex setup with multiple hyperparameters, it was easy to integrate. However, CEM's interpretability/explainability was limited as it failed to generate meaningful explanations, which further limited the assessment of its stability.

5. Discussion and Outlook

The central aim of this thesis was to investigate the ability of the algorithms in the AIX360 library to provide explainability for AI systems, with a particular focus on the unique and complex use case of skin cancer classification. The aim was to identify the optimal combination of these algorithms for improved explainability and to identify specific techniques that are ideally suited to the intricacies of skin cancer classification tasks.

In line with these objectives, it was found that each algorithm - ProtoDash, DIP-VAE, LIME, SHAP and CEM - has unique strengths and weaknesses that can significantly affect their suitability for skin cancer classification and other specific applications.

ProtoDash proved to be an efficient, user-friendly algorithm with robust local interpretability/explainability, making it a potential candidate for providing individual case insights in skin cancer. However, it failed to capture global trends and its sensitivity to various image transformations highlights the need for further algorithm refinement to improve stability, as the images used were taken in circumstances where small changes can occur.

DIP-VAE, despite its longer runtime and complexity, provided comprehensive interpretability/explanability, highly useful in skin cancer cases where deep and detailed understanding is critical. While its complexity of setup may be a challenge, its compatibility with different generative models extends its applicability in different skin cancer classification models.

LIME stood out for its simplicity, runtime, interpretability/explanability and stability. Its localised interpretability may prove beneficial in medical imaging, particularly in skin cancer detection, where specific regions of the image are of interest.

However, care should be taken not to extrapolate these localised findings to broader conclusions.

SHAP proved to be an efficient, robust and versatile tool. Its balance between global and local interpretability is a unique strength, making it very valuable for skin cancer classification tasks where understanding individual cases and overall patterns is crucial.

CEM faced significant challenges when applied to the complex skin cancer data set. In particular, it fell short in terms of computational efficiency and weak interpretability/explainability. However, its model-independent nature implies potential in various applications, including skin cancer classification, after further refinement.

From a broader perspective, this analysis highlights that the selection of XAI algorithms should consider not only their theoretical properties, but also practical aspects such as data complexity, model structure and computational resources. In terms of finding the optimal combination of algorithms to improve explanatory power, this remains a complex matter and opens opportunities for future research.

It can be concluded that the ProtoDash, DIP-VAE, LIME and SHAP algorithms from the AIX360 library show promising potential to provide sufficient explanatory power for the challenging task of skin cancer classification. However, each algorithm has shown unique strengths and weaknesses, suggesting that there is no single algorithm that can make these systems completely trustworthy, but rather that the combination of these algorithms greatly improves this aspect for all stakeholders.

In conclusion, while this work has shed light on the applicability of different XAI algorithms for skin cancer classification, there is still a way to go to achieve AI models that are not only effective, but also transparent and trustworthy. Future work should focus on refining these XAI algorithms to better handle complex data sets such as skin cancer, balancing simplicity, interpretability/explainability and stability. Exploring a wider range of algorithms and toolkits could open up new opportunities to improve the trustworthiness of AI systems even further, particularly tailored to the unique requirements of each use case, such as skin cancer classification.

Bibliography

[1] Thiebes, Scott et al. (2021) trustworthy artificial intelligence. 31(2):447–464.

[2] Davenport, Thomas and Kalakota, Ravi. (2019) the potential for artificial intelligence in healthcare. 6(2):94–98.

[3] Haggenmüller, Sarah et al. (2021) skin cancer classification via convolutional neural networks: systematic review of studies involving human experts. 156:202–216.

[4] Melanoma UK. (2020) melanoma skin cancer report [online]. Available from: `https://www.melanomauk.org.uk/2020-melanoma-skin-cancer-report` [Accessed 13 March 2023].

[5] Li, Xun et al. (2023) explainable dimensionality reduction (XDR) to unbox AI 'black box' models: A study of AI perspectives on the ethnic styles of village dwellings. 10(1):35.

[6] Arya, Vijay et al. (2019) one explanation does not fit all: A toolkit and taxonomy of AI explainability techniques.

[7] Digital Strategy. (2022) regulatory framework proposal on artificial intelligence [online]. Available from: `https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai` [Accessed 13 March 2023].

[8] Van Molle, Pieter et al. (2018) visualizing convolutional neural networks to improve decision support for skin lesion classification. In *Understanding and Interpreting Machine Learning in Medical Image Computing Applications*, pages 115–123, Cham. Springer International Publishing.

[9] Young, Kyle et al. (2019) deep neural network or dermatologist? In *Interpretability of Machine Intelligence in Medical Image Computing and Multimodal*

Learning for Clinical Decision Support, pages 48–55, Cham. Springer International Publishing.

[10] Columbia University. (2023) artificial intelligence (ai) vs. machine learning [online]. Available from: `https://ai.engineering.columbia.edu/ai-vs-machine-learning/` [Accessed 11 May 2023].

[11] Lo, Chang-Cheng et al. (2020) prognosis of bearing and gear wears using convolutional neural network with hybrid loss function. 20(12):3539.

[12] Mishra, Mayank. (2022) convolutional neural networks, explained [online]. Available from: `https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939` [Accessed 20 April 2023].

[13] Luber, Stefan and Litzel, Nico. (2019) was ist ein convolutional neural network? [online]. Available from: `https://www.bigdata-insider.de/was-ist-ein-convolutional-neural-network-a-801246/` [Accessed 1 March 2023].

[14] Fischer, Julia and Pochwyt, Kevin. (2017) neuronale netze [online]. Available from: `https://user.phil.hhu.de/~petersen/SoSe17_Teamprojekt/AR/neuronalenetze.html`[Accessed 1 March 2023].

[15] Erickson, Bradley J. and Kitamura, Felipe. (2021) magician's corner: 9. performance metrics for machine learning models. 3(3):e200126.

[16] Mohajon, Joydwip. (2022) confusion matrix for your multi-class machine learning model [online]. Available from: `https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826` [Accessed 1 March 2023].

[17] Malato, Gianluca. (2021) precision, recall, accuracy. how to choose? [online]. Available from: `https://www.yourdatateacher.com/2021/06/07/precision-recall-accuracy-how-to-choose/` [Accessed 19 May 2023].

[18] Broniatowski, David A. (2021) psychological foundations of explainability and interpretability in artificial intelligence.

[19] Mittelstadt, Brent. (2022) interpretability and transparency in artificial intelli-

gence. In Carissa Véliz, editor, *The Oxford Handbook of Digital Ethics*. Oxford University Press, 1 edition.

[20] The Royal Society. (2019) explainable ai [online]. Available from: `https://royalsociety.org/topics-policy/projects/explainable-ai/` [Accessed 30 March 2023].

[21] Arya, Vijay et al. (2019) guidance on use of ai explainability 360 algorithms [online]. Available from: `https://github.com/Trusted-AI/AIX360/blob/master/aix360/algorithms/methods-choice-updated.png` [Accessed 04 April 2023].

[22] AIX360. (2019) ai explainability 360 [online]. Available from: `https://aix360.readthedocs.io/en/latest/index.html` [Accessed 30 May 2023].

[23] Gurumoorthy, Karthik S. et al. (2019) efficient data representation by selecting prototypes with importance weights.

[24] Kumar, Abhishek et al. (2017) variational inference of disentangled latent concepts from unlabeled observations. Publisher: arXiv Version Number: 3.

[25] Ribeiro, Marco Tulio et al. (2016) "why should i trust you?": Explaining the predictions of any classifier.

[26] Lundberg, Scott M and Lee, Su-In. (2017) a unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[27] Dhurandhar, Amit et al. (2018) explanations based on the missing: Towards contrastive explanations with pertinent negatives. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[28] Hind, Michael et al. (2019) TED: Teaching AI to explain its decisions. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 123–129. ACM.

[29] Dash, Sanjeeb et al. (2018) boolean decision rules via column generation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and

R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[30] Wei, Dennis et al. (2019) generalized linear rule models. In Chaudhuri, Kamalika and Salakhutdinov, Ruslan, editor, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6687–6696. PMLR, 09–15 Jun.

[31] Dhurandhar, Amit et al. (2018) improving simple models with confidence profiles. volume 31. Curran Associates, Inc.

[32] AIX360. (2019) metrics - aix360 0.1 documentation [online]. Available from: `https://aix360.readthedocs.io/en/latest/metrics.html` [Accessed 19 May 2023].

[33] Alvarez Melis, David and Jaakkola, Tommi. (2018) towards robust interpretability with self-explaining neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[34] Luss, Ronny et al. (2021) leveraging latent features for local explanations.

[35] Skin Cancer Foundation. (2022) annual exams [online]. Available from: `https://www.skincancer.org/early-detection/annual-exams/` [Accessed 20 May 2023].

[36] City of Hope. (2022) skin cancer diagnosis and detection [online]. Available from: `https://www.cancercenter.com/cancer-types/skin-cancer/diagnosis-and-detection` [Accessed 29 May 2023].

[37] Tschandl, Philipp et al. (2018) the HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. 5(1):180161.

[38] Gregoor, Smak et al. (2023) an artificial intelligence based app for skin cancer detection evaluated in a population based setting. 6(1):90.

[39] Beltrami, Eric J. et al. (2022) artificial intelligence in the detection of skin cancer. 87(6):1336–1342.

[40] Lewis, Megan. (2021) an artificial intelligence tool that can help de-

tect melanoma [online]. Available from: https://news.mit.edu/2021/artificial-intelligence-tool-can-help-detect-melanoma-0402 [Accessed 30 May 2023].

[41] Peconic Bay Medical Center. (2019) various types of medical imaging explained [online]. Available from: https://www.pbmchealth.org/news-events/blog/various-types-medical-imaging-explained [Accessed 4 March 2023].

[42] Carestream Health. (2015) metadata: Creating meaningful access to clinical images and data for any user [online]. Available from: https://www.carestream.com/en/us/-/media/publicsite/resources/radiography-and-health-it/published-articles-and-white-papers/ris-pacs/pdf/whitepaper-enterprise-imaging-clinical-collaboration-201507.pdf [Accessed 4 March 2023].

[43] Cassidy, Bill et al. (2022) analysis of the isic image datasets: Usage, benchmarks and recommendations. *Medical Image Analysis*, 75:102305.

[44] Roxana Daneshjou, Kailas Vodrahalli, Roberto A. Novoa, Melissa Jenkins, Weixin Liang, Veronica Rotemberg, Justin Ko, Susan M. Swetter, Elizabeth E. Bailey, Olivier Gevaert, Pritam Mukherjee, Michelle Phung, Kiana Yekrang, Bradley Fong, Rachna Sahasrabudhe, Johan A. C. Allerup, Utako Okata-Karigane, James Zou, and Albert S. Chiou. (2021) disparities in dermatology AI performance on a diverse, curated clinical image set. 8(32):eabq6147.

[45] ISIC. (2019) isic 2019 challenge [closed] [online]. Available from: https://challenge.isic-archive.com/landing/2019/ [Accessed 1 March 2023].

[46] Rotemberg, Veronica et al. (2021) a patient-centric dataset of images and metadata for identifying melanomas using clinical context. 8(1):34.

[47] Codella, Noel et al. (2019) skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (ISIC).

[48] de Vargas, Werner et al. (2023) imbalanced data preprocessing techniques for machine learning: a systematic mapping study. 65(1):31–57.

[49] Gavrikov, Paul. (2020) visualkeras [online]. Available from: https://github.

com/paulgavrikov/visualkeras [Accessed 04 April 2023].

[50] Python. (2023) time — time access and conversions [online]. Available from: https://docs.python.org/3/library/time.html[Accessed 6 June 2023].

[51] Jung, Alexander. (2020) imgaug [online]. Available from: https://imgaug.readthedocs.io/en/latest/ [Accessed 6 June 2023].

List of Figures

List of Tables

A. Appendix

A.1. Project management

A.1.1. Project assignment

- **Type of Work:** Bachelor's Thesis
- **Title:** Evaluating XAI Algorithms in Skin Cancer Classification: A Path towards Trustworthy AI Systems
- **Description:** In the forthcoming year, an EU regulation is expected to be enacted that will require certification from an independent body for AI systems used in safety functions. Existing approaches to developing and testing AI systems have primarily revolved around questionnaires for certifiers. The next step should therefore be to enable certification on a technical basis, to which this work makes a contribution. This work can be carried out as follows:
  1. Familiarisation with the topic of certification of AI systems.
  2. Derivation of the necessary measures for a secure AI system.
  3. Implementation of an AI system (alternatively: adaptation of an existing system) taking into account the given development methods.
  4. Planning of the testing of the system.
  5. Testing of the system using existing tools.

  – From this, the aspects that are not yet covered by the existing tools can be derived, which can be used as an entry point for an optional continuation of the work, in which suitable methods and/or a tool can

then be implemented in order to then close these gaps.

- **Prerequisites:** Basic knowledge in Machine Learning, Interest in the interplay between implementation and related processes, methods, tools
- **Agreement:** Continuation of the PA
- **Thesis Advisor:** Monika Ulrike Reif
- **Primary Supervisor:** Monika Ulrike Reif
- **Primary Subject Area:** Machine Learning
- **Additional Subject Areas:** Data Analysis
- **Degree Program:** Computer Science
- **Institute / Centers:** Institute for Applied Mathematics and Physics (IAMP)
- **Internal Partners:** None
- **English:** Yes

A.1.2. Project plan

The following time schedule illustrates roughly the sequence and duration of each phase of work during the development of this thesis. It provides an overview of the project's progression, from the initial preparation and research stages, through the coding, writing and correction phases, to the final submission.

Schedule for Bachelor Thesis																
Date	**20.02.23**	**27.02.23**	**06.03.23**	**13.03.23**	**20.03.23**	**27.03.23**	**03.04.23**	**10.04.23**	**17.04.23**	**24.04.23**	**01.05.23**	**08.05.23**	**15.05.23**	**22.05.23**	**29.05.23**	**05.06.23**
Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1. Preparatory phase																
Clarify formalities																
Create a schedule																
Create first outline																
2. Research and structuring phase																
Literature																
Literature review																
Literature research																
Data collection																
Evaluation																
Selection																
Sorting of the material																
3. Concept and implementation phase																
Outline implementation																
Data gathering																
Code scaffolding																
Write code																
Training Model																
Evaluate the model																
4. Writing phase																
Structure																
Declaration of Originality																
Introduction																
Theoretical background																
Procedure and methods																
Summary and abstract																
Results																
Discussion and outlook																
Lists and annexes																
5. Correction phase																
Proofreading																
Checking the lists																
Plagiarism check																
Examination layout and formalities																
Check formatting																
6. Submission phase																
Export to PDF																
Submit to Complesis																