Multi-scale Local Explanation Approach for Image Analysis Using Model-agnostic Explainable Artificial Intelligence (XAI)

by

Hooria Hajiyan

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Science (MSc) in Computer Science

Faculty of Science
University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

August 2022

THESIS EXAMINATION INFORMATION

Submitted by: **Hooria Hajiyan**

Master of Science in Computer Science

Thesis Title: Multi-scale Local Explanation Approach for Image Analysis Using Model-agnostic Explainable Artificial Intelligence (XAI)

An oral defense of this thesis took place on July 25, 2022 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Patrick Hung
Research Supervisor	Dr. Mehran Ebrahimi
Examining Committee Member	Dr. Heidar Davoudi
Thesis Examiner	Dr. Min Dong

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

The recent success of deep neural networks has generated a remarkable growth in Artificial Intelligence (AI) research, and it received much interest over the past few years. However, one of the main challenges for the broad adoption of deep learning based models such as Convolutional Neural Networks (CNN) is the lack of understanding of their decisions.

Local Interpretable Model-agnostic Explanations (LIME) is an explanation method which produces a coarse heatmap as a visual explanation highlighting the most important superpixels affecting the CNN's decision.

This thesis aims to explore and develop a multi-scale scheme of LIME to explain decisions made by CNN models through heatmaps of coarse to finer scales. More precisely, when LIME highlights large superpixels from the coarse scale, there are some tiny regions in the corresponding superpixel that influenced the model's prediction at the finer scale. Therefore, we propose a multi-scale scheme of LIME and two weighting approaches based on Gaussian distribution and a parameter-free framework to produce visual explanations observed from different scales. We investigated the proposed multi-scale scheme on Flower Dataset from TensorFlow and a biological dataset, Camelyon 16. The results prove that the explanations are faithful to the underlying model, and our visualizations are reasonably interpretable.

Keywords: Explainable Artificial Intelligence (XAI); LIME; Multi-scale Explanations; Parameter-free Weighting Framework.

Author's Declaration

I hereby declare that this submission is entirely my own work, in my own words, and that all sources used in researching it are fully acknowledged. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Hooria Hajiyan

Statement of Contributions

Part of this work described in Chapters 4, and 5 are being prepared for publication.

Acknowledgements

I would like to express my sincere appreciation to my supervisor Dr. Mehran Ebrahimi for his patience, help, and support that allowed me to successfully accomplish my research. He trusted me and gave me the opportunity to be a member of his team. Additionally, I would like to thank Dr. Kourosh Davoudi for his valuable suggestions on my research.

I would like to express my gratitude and appreciation to the faculty and the Computer Science program members.

Last but not least, I would love to give my special regards to my amazing family and express my deepest gratitude to my beloved brother, Dr. Mohammadhossein Hajiyan, for his endless support. This journey would not have been possible without their patience, love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Automatic image analysis is termed computer vision, which is an interdisciplinary field that deals with how computers can gain an understanding of digital images or videos [64]. Computer vision seeks to automate common image analysis tasks such as classification, segmentation, object detection, and registration through Artificial Intelligence (AI) and Deep Learning (DL). DL is a subset of machine learning and AI in which the processes of feature selection and extraction from images and classification happen sequentially which eliminate the need for human intervention during the process [137]. In other words, DL is a type of representation learning in which no feature selection is used. Instead, the algorithm learns on its own which features are best for classifying the data [64]. The success of Deep Neural Networks (DNNs), specifically Convolutional Neural Networks (CNNs) and CNNs with fully connected layer, has led several research in image denoising [43, 76, 117], auto segmentation [63, 73], image reconstruction [33, 89, 129], and image classification [13, 26, 66, 133].

Despite the high accuracy and satisfactory performance of DNNs in AI, it remains unclear how a particular neural network arrives at a specific decision. Moreover, the architecture of DNNs makes these models look like a black-box which means that we cannot easily interpret based on what information in the input the model comes to a decision, when it can be trusted or even corrected. In other words, although DNNs exhibit high performance, they are opaque in terms of explainability. There may be an inherent conflict between the predictive accuracy and explainability. Often, the highest performing methods are the least explainable, and the most explainable models are the least accurate [48].

Generally, humans tend to adopt interpretable, tractable, and trustworthy techniques, given the increasing demand for explanation [9]. Explanations supporting the output of a model are crucial for high-risk use cases, for example in medical domains, where experts require more information from the model than a simple prediction [54, 120]. Therefore, just because DNNs are not interpretable, it does not mean that they are not explainable [111]. To address this issue, Explainable Artificial Intelligence has been proposed to shift toward more transparent AI, resulting in the development of techniques to explain decisions by AI models. This area of research is often called Explainable-AI (XAI), which emphasizes understanding cause and effect within the AI system by examining the sensitivities of the output to changes in the parameter inputs without needing to understand the complex computation of the model [2, 9, 74]

1.2 Problem Definition and Challenges

Explaining a prediction means presenting textual or visual artifacts that provide a qualitative understanding of the relationship between the instance's components (e.g. words in text, patches in images) and the model's prediction. We argue that explaining predictions is an important aspect of getting humans to trust and use machine learning effectively if the explanations are faithful and intelligible [102]. For example, in image classification, the cause of a decision comes from the important regions/features of an input image. Several XAI methods aim to interpret the decision of a black-box model in classification problems by representing the important features of an image at region/superpixel revel or pixel level. In order to choose the best XAI method in image analysis, there are two main remarks in the literature: Does it need access to the inner structure of the black-box model? How much the XAI tool is accurate to represent tiny regions/pixels of an image affecting the prediction, which is called localization?

Generally, if XAI methods need access to the network's structure, they cannot be easily applied, and altering the structure of the black-box model is a real barrier. These approaches explain the decision of a black-box model by taking the gradient of each feature with respect to (w.r.t.) the target class. On the other hand, XAI methods that do not need to change the inner structure of black-box models are easy to apply and implement, such as model agnostics methods. This thesis presents a new version of the Local Interpretable Model-agnostic Explanations (LIME) for multi-scale visual explanations of a black-box decision process in image classification problems. It proposes accurate visual explanations while representing the most influential regions of an image from coarse to finest scales.

1.3 Research Question

This thesis studies how multi-scale visual explanations can be inferred from CNNs with LIME as a model-agnostic explanation instead of gradient-based approaches and how to visualize the results to have more variations obtained from different scales. In detail, although gradient-based approaches can localize multiple objects in an individual image, but it still needs access to the inner structure of the black-box model. In this thesis, we try to answer the following research question: How can we use a model-agnostic perturbation-based approach to explain the contribution of each superpixel from multi-scales in an arbitrary CNN?

1.4 Contribution

The main contributions of this thesis is to explain the decision of a CNN model in image classification problems. We summarize the main contributions in the following.

- This research aims to propose a multi-scale version of LIME, which can represent the most important regions of an input image through heatmaps where the important superpixels/regions are highlighted at coarse to fine scales.
- To summarize the explanation heatmaps produced by the proposed multi-scale scheme, we proposed two weighting strategies, based on the Gaussian function and a parameter-free framework based on the number of superpixels at each scale.
- The Gaussian function will assign higher weights to mid-range scale experiments with the idea of not focusing on too coarse or too fine scales.
- The parameter-free automated weighting approaches will take the number of superpixels at each level, then assign weights to the heatmaps according to how much-varied information each scale provides.

More precisely, the main idea is to use LIME in a multi-scale scheme to explain the decision of a CNN model w.r.t. to the features of an input image and provide explanations for a range of scales. Finally, we expect to have visual explanations of coarse to finest scales that represent the most important regions affecting the model's prediction instead of a coarse heatmap that LIME provides. This proposed multi-scale explanation of LIME could also be more precise and class discriminative. To our knowledge, it is the first attempt to infer multi-scale visual explanations from LIME.

1.5 Thesis Outline

This thesis is organized in six Chapters and structured as follows:

- Chapter 2 introduces the necessary definitions and concepts related to XAI and DNNs. This Chapter includes the expectation of an XAI system, along with different modes of explanation, and various current approaches in the literature.

- Chapter 3 presents current approaches for visualization and understanding the decision making process of DNNs for images, followed by the quantifying explainability of visual explanations.

- Chapter 4 presents our proposed multi-scale framework for LIME to produce visual explanations by representing the important patches of an image from coarse to finest scales. In this Chapter, we also propose different ways to calculate the weighted heatmaps that LIME produces through the proposed multi-scale scheme.

- Chapter 5 reports the results of the proposed framework on Flower classification dataset from TensorFlow Datasets (TFDS), and a Biological dataset, Camelyon 16. In this Chapter, we compare the effectiveness of the proposed multi-scale scheme compared to the existing versions of LIME, quantitatively and qualitatively.

- In Chapter 6, we highlight the key contributions and some of the limitations of the thesis research. The Chapter also presents some interesting future directions along which the thesis research can be extended.

1.6 Software & Source Code

Software

The implementation of the proposed models are in Python programming language. Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is meant to be an easily readable language. Majority of the training and experiments in this research, were performed on graphical processing units (GPUs), while central processing units (CPUs) were used for data loading and pre-processing purposes. Following packages are examples of Python packages used in this thesis.

- NumPy is a Python package created for the purpose of scientific computing in Python. It supports large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions and operations that can be applied to these arrays. https://numpy.org/
- TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. https://www.tensorflow.org/
- Scikit-image, or skimage, is an open source Python package designed for image preprocessing , and computer vision. https://scikit-image.org/

Source Code

You can access the Python implementations of our models, evaluation metrics, and visualization of the explanations at the following link.
https://github.com/hajiyanh/Multi-scale-Visual-Explanation-of-LIME

Chapter 2

Background

Accuracy concerns the ability of a model to make correct predictions, while interpretability concerns to what degree the model allows for human understanding [58]. Explanations for machine decisions and predictions are needed to justify their reliability. This requires greater interpretability, which often means we need to understand the mechanism underlying the algorithms [86]. One of the main challenges of using DL models, such as CNNs, is the lack of understanding of their decisions. A simpler and less capable model is favourable in many applications as they can be easily understood. However, Deep Learning models, called black-box models, have superior performance, but they do not end up being highly interpretable. Understanding the trade-off between the model's accuracy and interpretability in real-world problems is important. Having a model that gives the best accuracy leads to a low level of interpretability.

2.1 Deep Learning

When programmable computers were first conceived, people wondered whether such machines might become intelligent. AI has been a thriving field with many practical applications and active research topics that have led to intelligent software to automate routine labor, understand speech or images, make diagnoses in medicine and support basic scientific research. In the early days of AI, this field rapidly tackled and solved problems that are intellectually difficult for human beings but relatively straightforward for computers, such as problems that can be described by a list of formal, mathematical rules, like recognizing spoken words or faces in images [44].

The difficulties faced by systems relying on hard-coded knowledge suggest that AI systems need the ability to acquire their knowledge by extracting patterns from raw data. This capability is known as Machine Learning (ML). The introduction of machine learning-enabled computers to tackle problems involving knowledge of the real world and make decisions that appear subjective. A simple ML algorithm called logistic regression [44]. The performance of these simple ML algorithms depends heavily on the representation of the data they are given. For example, when logistic regression recommends cesarean delivery, the AI system does not examine the patient directly. Instead, the doctor tells the system several pieces of relevant information, such as the presence or absence of a uterine scar. Each piece of information included in the representation of the patient is known as a feature. Logistic regression learns how each of these patient features correlates with various outcomes. However, it cannot influence how features are defined in any way. For example, if logistic regression were given an MRI scan of the patient rather than the doctor's formalized report, it would not be able to make useful predictions. This is because individual pixels in an MRI scan have a negligible correlation with any complications that might occur during delivery.

Many AI tasks can be solved by designing the right set of features to extract. However, it is difficult to know what features should be extracted. For example, suppose we would like to write a program to detect cars in photographs. We know that cars have wheels, so we might want to use the presence of a wheel as a feature. Unfortunately, it is difficult to describe precisely what a wheel looks like in terms of pixel values. A wheel has a simple geometric shape, but its image may be complicated by shadows falling on the wheel, the sun glaring off the wheel's metal parts, the car's fender or an object in the foreground obscuring part of the wheel, and so on. One solution to this problem is to use ML to discover the mapping from representation to output and the representation itself. This approach is known as representation learning. Learned representations often result in much better performance than can be obtained with hand-designed representations. They also enable AI systems to adapt to new tasks with minimal human intervention rapidly. For example, a representation learning algorithm can discover a good set of features for a simple task in minutes or for a complex task in hours to months. This is done by allowing computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts. By gathering knowledge from experience, this approach avoids the need for human operators to specify all the knowledge that the computer needs formally. The hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers [44]. The quintessential example of a DL model is the deep feedforward network, or multilayer perceptron. A multilayer perceptron is just a mathematical function mapping some set of input values to output values. The function is formed by composing many simpler functions. We can think of each application of a different mathematical function as providing a new representation of the input. The idea of learning the right representation for the data provides one perspective on deep learning. Each layer of the representation can be thought of as the state of the computer's memory after executing another set of instructions in parallel. Networks with greater depth can execute more instructions in sequence. Sequential instructions offer great power because later instructions can refer back to the results of earlier instructions.

2.1.1 Convolutional Neural Networks

Deep Neural Networks (DNNs) refer to Artificial Neural Networks (ANN) with multilayers, which have been considered one of the most powerful tools and have become very popular over the last few decades as they can handle a vast amount of data. The interest in having deeper hidden layers has recently begun to surpass classical methods in different fields [3].

Convolutional Neural Networks (CNNs) are currently one of the most popular deep learning architectures used in a wide range of applications, especially for image-based visual recognition tasks. More specifically, a CNN is a particular type of deep learning-based feed-forward neural network, which can take an input and assign prediction for a specific task for objects present in the input. CNNs are primarily used to solve complex image-driven tasks, including classification, object detection, registration and segmentation [25, 69, 88, 92, 115]. The information process has inspired the idea of architectural design for CNNs in the human brain's visual cortex. In the visual cortex, each neuron responds to stimuli only for a certain region of the visual field known as a receptive field.

CNNs are designed for data with spatial structure, e.g. sequence of characters in the text, sound signals, images, videos, and 3D voxel data. In each case, input is a high-dimensional tensor with highly correlated features. For example, in the case of a color image, the input is a $h \times w \times 3$ array where h and w are image height and width, respectively, and each index represents a pixel color in a spatial structure. These pixels are highly correlated, which means their values and locations in a spatial neighbourhood form structural information.

As the name suggests, a CNN employs a mathematical operation known as convolution. Convolutional networks are neural networks that use convolution in place of general matrix multiplication in at least one of their layers [44] The network consists of different layers, and each layer consists of small convolutional kernels. At each layer, convolution operators are being performed on the output from the previous layer to form a new input,

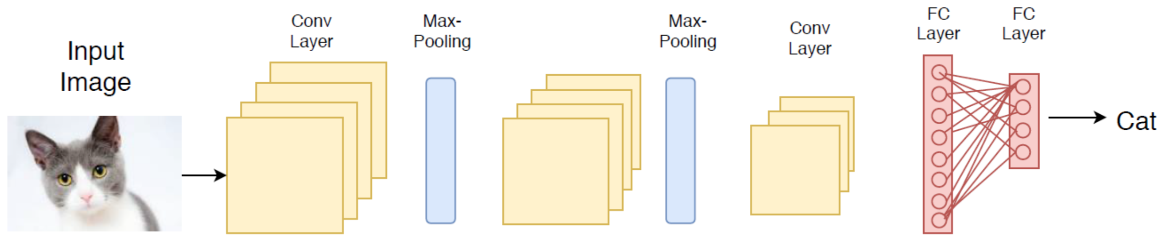


Figure 2.1: Example of CNN architecture design: consists of 3 convolutional layers, 2 max pooling layers, and two fully connected layers. At each layer, the input is convolved with the convolutional kernels to create activation maps for the next layer. The output of the network is a categorical probability distribution and a loss function is being used during training to find the values of the convolutional kernels.

called a feature map, for the next layer. These kernels can be seen as local feature extractors encode the input in hierarchical order passing through the network. The values of the convolutional kernels are the network parameters, called weights, that need to be determined during the training process. Fig 2.1 is a schematic view of a CNN network for an image classification task. An image is passed to a network, and at each layer, convolutional filters convolve the input to create activation maps for the next layer. The output of the network is categorical class labels representing the probability of the input being one of the categories. A loss function is being used during training to update the values of the convolutional kernels using backpropagation.

While CNN architectures have many parameters that make them unique from each other, CNNs are typically composed of few essential constituent layers and operations. The main constituents of the typical CNN architecture design are the convolutional layer, pooling layer, activation function, batch-normalization layer, dropout and fully connected layer. These individual components are explained below in detail [68, 93].

Convolutional Layer

In mathematics, the convolution of a 1D signal f with another signal g is defined as

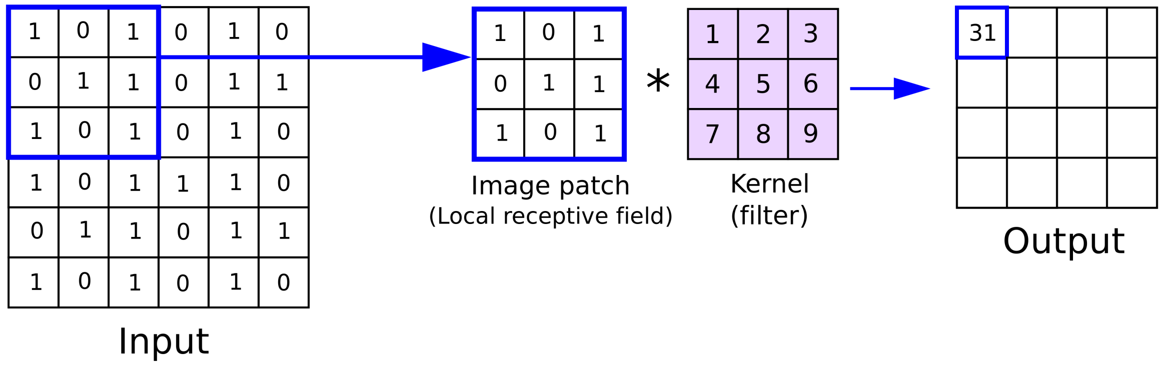


Figure 2.2: Convolutional layer with a local receptive field extracts local features in a hierarchical order.

$$o[n] = f[n] * g[n] = \sum_{u=-\infty}^{+\infty} f[u]g[n-u]. \tag{2.1}$$

Here, n and u are discrete variables. This definition can be extended for 2D convolution, as required for pixel based convolutions as

$$o[m,n] = f[m,n] * g[m,n] = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} f[u,v]g[m-u,n-v], \tag{2.2}$$

where m and n are the two dimensions of the original signal, and u, v are indexes of the second signal. However, when considering convolutions applied on images, the input is bound to finite numbers, i.e., size of the input images and associated filters.

The most critical building block of a CNN is a convolutional layer. Neurons in the convolutional network are only connected to every pixel in their receptive field [82]. For example, neurons in the first layer of a CNN only see a small region of the input image with the size of the first convolutional filter, Fig 2.2. Consecutively, each neuron in the second layer is connected only to a small region from the output of the first layer. This architecture allows the network to extract local low-level features in the early layers and then assemble them into higher-level features in the consecutive layers.

Activation Function

Activation functions are used for inducing non-linear properties in neural networks. These functions are responsible for deciding whether a neuron should be fired or not. In other words, the activation function will check to see if the information from the previous neuron is relevant or not. Usually, an activation function is applied to the linear output of every convolution to prevent the network from collapsing into a single layer. As shown in equation 2.3, we multiply the input I with the weight W of the neuron and add the bias b. Then apply a non-linear activation function σ. The transformed output O is then sent to the next layer.

$$O = \sigma(W \times I + b) \tag{2.3}$$

Some of the popular activation functions are sigmoid, hyperbolic tangent function (tanh), and Rectified Linear Unit (ReLu).

Pooling Layer

The pooling layer is generally applied after convolution and non-linear activation layers or functions. The pooling layer in a CNN architecture is responsible for reducing the spatial size of the feature maps at a given layer. The previous layers' local region and window size are replaced with statistics that summarize the neighbouring outputs in the spatial reduction process. As a result, the size of every input region and, eventually, the input feature map is spatially reduced. The spatial reduction is usually done using max or average operation. For example, in the max-pooling process, pooling is done over a window, as shown in Fig 2.3, and the maximum activation value of the window size is selected. Mathematically, a pooling layer with a square input matrix of size M_{in}, outputs a square matrix of size M_{out}. Assuming choosing a stride that leads to no overlapping of filters, the input is divided into pooling regions $p_{i,j}$ of a stride of size $k \times l$.

$$M_{out} = max_{(k,l) \in p_{i,j}} M_{in} \tag{2.4}$$

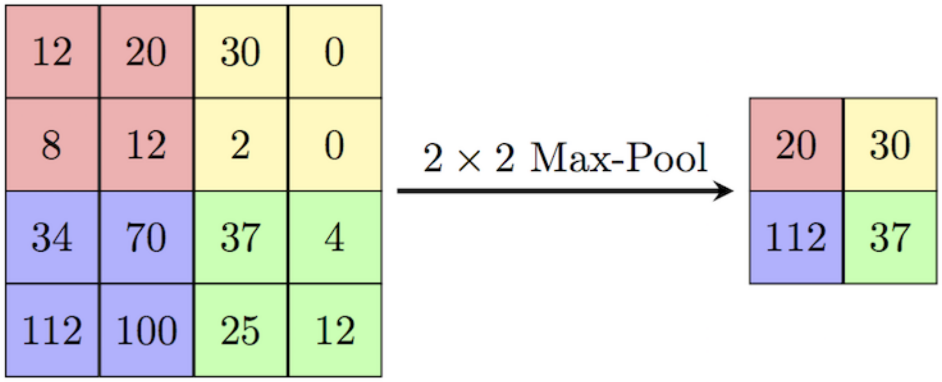


Figure 2.3: Example of max-pooling using a 2×2 stride. The pooling operation is done to decrease the computational power required to process the data through dimensionality reduction and propagate only the most dominant and relevant information further down the network.

Max pooling becomes a max operation applied element-wise to a given region and tiled across the feature map with the given stride. In equation 2.4, k and l is the stride, integer values, where usually k has the same value as l. In average pooling, the average of the activation values in the window is used. Most commonly, max pooling is used over average pooling in various modern CNN architectures.

Batch Normalization Layer

The batch normalization (BN) layer normalizes the output of the previous layer x by subtracting the mean μ from the output and dividing it with the standard deviation of the output matrix σ^2 for a given batch b [56].

Dropout Layer

The idea of dropout layer is based on probability. The method temporarily drops out certain neurons during training of the neural network. Mathematically, we use a probability p, to select whether to drop a neuron temporarily. Usually the value range for p varies from 0.25 - 0.5.

Fully Connected (FC) Layer

Fully Connected (FC) layers are generally used closer to the output layer to capture global context and model high-level concepts. As its name suggests, each neuron in the previous layer is connected to every neuron in the next layer in a fully connected layer. This layer can be realized as matrix multiplication and adding of a bias term. Consider a neural network with L hidden layers expressed in matrix form in equation 2.5,

$$O_l = \sigma_l(W_l I_{l-1} + b_l), \tag{2.5}$$

where l is a particular layer in the given network, O_l is the output of the layer l, σ is the activation function, W_l is the weight matrix of the layer, I_{l-1} is the output of the previous layer, and b_l is the bias.

2.1.2 CNN Architecture

Combining the different components mentioned in the sections above, we can obtain the architecture as described in Fig 2.1. The architecture of the CNN consists of three convolutional layers, two max-pooling layers and two fully connected layers in the end. Furthermore, activation functions are assumed to be implicitly applied after each convolutional operation, which are not shown explicitly in the figure [68].

First, the input image is fed into the first convolutional layer for information processing. Then, the convolutional filters are applied to the different channels and summed up individually to form feature maps equal to the number of convolutional filters in the given layer. Next, each map is passed through the activation function and the max-pooling layers. This process is repeated two more times. After the last convolutional, the feature map outputs are flattened and passed along to the fully connected layers. Finally, we pass the obtained activations through the classifier, such as softmax activation, for multi-classification problems to obtain the prediction probability of an input image belonging to each class.

2.2 Explainable Artificial Intelligence (XAI)

XAI has experienced significant growth over the last few years. This is due to the widespread application of ML, particularly DNNs, that has led to the development of highly accurate models but lack of explainability and interpretability [125]. This opacity of such black-box models has created the need for XAI architectures motivated mainly by three reasons. 1) The demand to produce more transparent models. 2) The need for techniques that enable humans to interact with them. 3) The requirement of trustworthiness of their inferences [28].

An XAI system aims to make the model's behavior more intelligible to humans by providing explanations. Every explanation is set within a context that depends on the task, abilities, and expectations of the user of the AI system [48]. The upturn in the XAI research outputs of the last decade is prominently due to the fast increase in using ML and DNNs in several business areas like e-commerce [130], games [4], criminal justice [104], computer vision [105], and healthcare [34].

Looking at the first definition of XAI models, explanations can be full or partial [48]. Fully interpretable models give full and completely transparent explanations. These models are recognized as transparent models, like regression or decision trees. Transparent models obey "interpretability constraints" that are defined according to the domain when the model obeys particular rules and constraints to certain variables. In contrast, black-box models do not necessarily follow these constraints. The non-transparent structure of black-box models leads to partial explanations, which reveal important pieces of the reasoning process. Partial explanations may include variable importance w.r.t. the output.

The conceptual framework at the basis of the proposed XAI system is represented in Fig 2.4. Most of the XAI methods focus on interpreting and making the entire process of building an AI system transparent, from the inputs to the outputs via the application

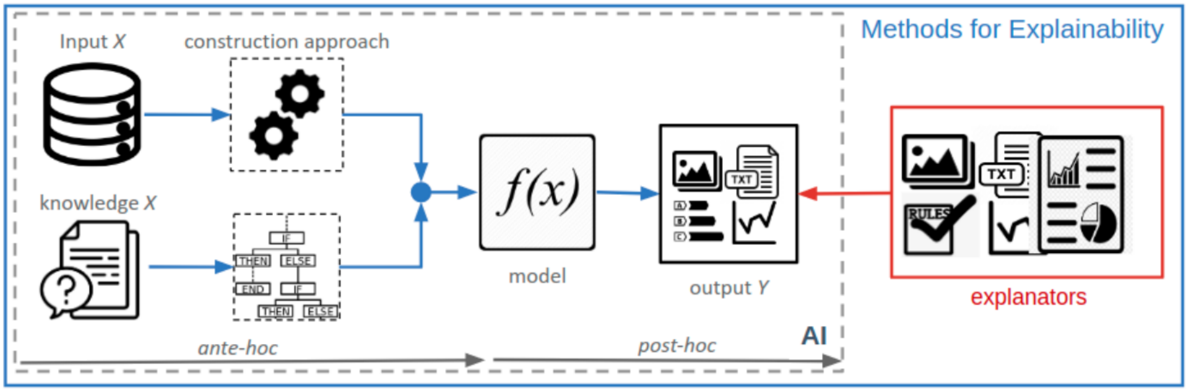


Figure 2.4: Diagrammatic view of how an XAI solution is typically constructed.

of a learning approach to generate a model [126]. The outcome of these methods are explanations of different formats, such as rules, numerical, textual or visual information or a combination of the former.

2.2.1 Discriminative Aspects of XAI Methods

There are five main criteria for discriminating XAI methods in the literature [47, 126], summarized below.

- *Scope*: First, the scope of an explanation can be either global or local. In the former case, the goal is to make the entire inferential process of a model transparent and understandable as a whole [47]. In the latter case, the objective is to explain each inference of a model [22, 75].

- *Stage*: The second dimension refers to the stage at which a method generates explanations. Ante-hoc methods aim to consider the explainability of a model from the beginning and during training to make it naturally understandable while still trying to achieve optimal accuracy [28,79,80]. Post-hoc methods keep a trained model unchanged and mimic or explain its behaviour by using an external explainer at testing time [28, 75, 90, 94].

- *Problem type*: The third dimension refers to the problem type. XAI methods can vary according to the underlying problem, either classification or regression.

- *Input data*: It says that the mechanisms followed by a model to classify images can substantially differ from those used to classify textual documents, thus, the input data of a model can play an important role in constructing a method for explainability.

- *Output*: Finally, the output format, similarly to input data, can demand different formats of explanations to be considered by a method for explainability: numerical, rules, textual, visual or mixed [125].

The different formats of explanations are the natural consequence of the widespread application of AI-powered technologies that are utilized by different users in various fields to solve distinct problems [24, 50, 127]. System designers, developers and AI practitioners find useful explanations that accurately reflect the logic implemented within a model [131]. In this case, rule-based explanations represent a structured, compact yet understandable way to represent a set of logical instructions. On the other hand, end-users belonging to the lay public prefer reconstructive explanations that build a story, exposing which input features contribute the most to the model's prediction. For example, visual and textual explanations can tell why the image of an animal was assigned to a certain class in an intuitive way, such as "this image represents a penguin because it is white and black and it has a beak" [127].

Visual explanations are probably the most natural way of communicating things and a very appealing manner to explain them [125]. Scholars have analysed various visualization tools to determine which ones are the most suitable for certain applications or meet the favour of scholars and practitioners. An example of these tools are heatmaps which highlight the specific areas of an image or specific words of a text that mainly influence the inferential process of a model by using different colors [102, 114]. Another intuitive form of explanation for humans is textual explanations, natural language statements which can either be written or orally uttered. An example is a phrase “This is a Brewer Blackbird because this is a blackbird with a white eye and long pointy black beak” shown by an explainer of an image classification model [52] Rules are a schematic, logical format, more structured than visual and textual explanations but still intuitive for humans. Rules can be in the form of “IF...THEN” statements with AND/OR operators, and they are very useful for expressing combinations of input features and their activation values [38].

2.2.2 XAI Taxonomy

In the literature, various terms exist to indicate the opposite of the black-box nature of some of the AI and ML, especially DL, models. In this section, we distinguish the following terms [7]:

- *Interpretability*: Interpretability is defined as explaining a model or providing the meaning in understandable terms to a human [75]. For example, a ML model is interpretable if its internal structure and decision-making process are clear and understandable for a human. As it also referred to transparency, a model is considered to be transparent if by itself it is understandable for a human, like regression, and decision tree. [75].

- *Explainability*: It is related with the notion of explanation as an interface between humans and an AI system [47]. In other words, explainability refers to the extent to which a system's internal mechanism can be explained in human terms. It comprises AI systems that are accurate and comprehensible to humans [47].

Although these terms are similar in their semantic meanings, they provide different levels of AI to be accepted by humans. Typical transparent models [2] include K-nearest neighbors (KNN), decision trees, rule-based learning, bayesian network, and etc,. The decisions from these models are often transparent. However, typical opaque models refer to any model that is not transparent by itself [96] including ML models, DNNs, Support Vector Machines (SVMs), and etc,. Although these models often achieve high accuracy, they are not transparent.

For more details, the ontology and taxonomy of XAI at a high level can be detailed as below:

- *Model agnostic*: Model-agnostic explanation methods [27] is designed to be generally applicable. As a result, they have to be flexible enough so that they do not depend on the intrinsic architecture of the model. Therefore, these models are based on relating the input of a model to its outputs.

- *Model specific*: Model-specific XAI approaches often take advantage of knowing a specific model and aim to bring transparency to a particular type of one or several models [11].

- *Explanation by simplification*: By simplifying a model via approximation [121], we can find alternatives to the original models to explain the prediction we are interested in. For example, we can build a linear model or a decision tree around the predictions of a model, using the resulting model as a surrogate to explain the more complex one.

- *Explanation by feature relevance*: This idea is similar to simplification. Roughly, this type of XAI approaches attempts to evaluate a feature based on its average expected marginal contribution to the model's decision after all possible combinations have been considered [18, 97].

2.3 Transparency of a black-box Model

Most of the DNNs used in image classification, image processing and computer vision tasks work like black-box models. The non-transparent structure of these models led to several types of research in XAI. [23, 77, 132].

In the medical field, clearly human lives are on the line. A comprehensive literature review on the applications of XAI on medical images is provided in [31]. Detection of a disease at its early phase is often critical to the recovery of patients or to prevent the disease from advancing to more severe stages [120]. Therefore, research community always keep track of how algorithms are used and how their usage can be trusted or improved. As a result, interpretability and explainability of ML algorithms, specifically DNNs, have thus become pressing issues and raising critical questions: If things go wrong, can we explain why? If things are working well, do we know why and how to leverage them further? There is an interchangeable misuse of interpretability and explainability in the literature. The notable difference among these concepts is interpretability refers to a passive characteristic of a model referring to the level at which a given model makes sense for a human, which is also expressed as transparency. By contract, explainability refers to an active characteristic of a model with its internal functions [9]. In other words, A model can be explained, but the interpretability of the model is something that comes from the design of the model itself [9].

2.3.1 Interpretability vs Explainability

The ML literature predominantly uses the term "interpretability" instead of "explainability". Still, according to [14], interpretability itself is insufficient as it does not cover all possible problems associated with understanding black-box models. Therefore, to gain users' trust and acquire meaningful insights about the causes, reasons, and decisions of black-box approaches, explainability is required rather than simple interpretability. Although explainable models are interpretable by default, the opposite is not always true.

As ML penetrates critical areas such as medicine, health care support systems, and financial markets, the inability of humans to understand these models seems problematic [75]. The main difference between interpretability and explainability of a complex model relies on how much access to the model's internal structure and mathematical concepts is needed. In other words, interpreting a neural network means understanding the mathematical principles and the structure of that model even by going through each layer of a DNN model. However, explainability refers to such techniques that explain the output of a complex model without needing to understand the black-box model's inside. Even if we understand the underlying mathematical principles, explainability is important. Providing explanations of predictions can be beneficial in teaching, learning, and research [62].

2.3.2 Faithfulness vs Interpretability

The faithfulness of an XAI method to a model is its ability to explain the function learned by the model accurately. So naturally, there is a trade-off between an XAI system's interpretability and faithfulness. It is often impossible for an explanation to be completely faithful unless it is a complete description of the model itself. For an explanation to be meaningful, it must at least be locally faithful, i.e., it must correspond to how the model behaves in the vicinity of the instance being predicted [102]. A more faithful explanation is typically less interpretable and vice versa. One could argue that a fully faithful explanation is the entire description of the model, which is not interpretable/easy to explain in the case of deep models [110].

To check the faithfulness of an explanation, we can measure the difference in the model's prediction while removing the essential feature detected by the explanation. For example, suppose the input is an image, and the explanation is a localized heatmap. In that case, one obvious choice for such a visualization is image occlusion [134], where we measure the difference in the model's predictions when patches of the input image are masked. In other words, patches that change the model's prediction score should be the patches that the visual explanation assigned high intensity.

2.3.3 Expectations of an XAI System

XAI assumes that an explanation is provided to an end-user who depends on the decisions, recommendations, or actions produced by an AI system. There might be many different users, such as intelligence analysts, judges, or operators. However, other users who demand an explanation of the system might be a developer or test operator who needs to understand where there might be areas for improvement. Another user might be policy-makers trying to assess the system's fairness. Each user group may have a preferred explanation type to communicate information most effectively. A practical explanation will take the target user group into account, who might vary in their background knowledge and needs for what should be explained [48]. One of the principal reasons to produce an explanation is to gain the trust of users [30]. Trust is the main way to increase users' confidence with a system [119] and to make them feel comfortable while controlling and using it [75]. There are also other positive effects brought by explainability. According to [49], it is part of human nature to assign causal attribution of events. A system that provides a causal explanation of its inferential process is perceived as more human-like by end-users. Thus, causality is considered a fundamental attribute of explainability [15, 75, 87, 91]. Explanations must make the causal relationships between the inputs and the model's predictions explicit, especially when these relationships are not evident to end-users in black-box models. Data-driven models are designed to discover and exploit associations in the data, but they cannot guarantee a causal relationship in these associations. There are four main reasons supporting the necessity to explain the logic of an inferential system, or a learning algorithm suggested in [2]:

- *Explain to justify*: the decisions made by utilizing an underlying model should be explained to increase their justifiability.

- *Explain to control*: explanations should enhance the transparency of a model and its functioning, allowing its debugging and the identification of potential flaws.

- *Explain to improve*: explanations should help scholars improve the accuracy and efficiency of their models.

- *Explain to discover*: explanations should support the extraction of novel knowledge and the learning of relationships and patterns.

2.4 Modes of Explanations

According to the literature, explanations are often categorized into two main aspects [2, 47]. The first distinguishes whether the explanation is for an individual prediction, called "local explanation", or the model's prediction process as a whole, called "global explanation". The second differentiates between the explanation emerging directly from the prediction process, which means Ante-hoc, versus requiring post-processing or Post-hoc. We describe both of these aspects in detail in the following,.

2.4.1 Local vs Global

A "local explanation" provides information or justification for the model's prediction on a specific input. A "global explanation" provides a similar justification by revealing how the model's predictive process works. In other words, global explanation describes the whole decision process as a human term independent of any particular input [23].

2.4.2 Ante-hoc vs Post-hoc

Whether the explanation is local or global, explanations differ on whether they arise as part of the prediction process or whether their generation requires post-processing following the model making a prediction [23].

Ante-hoc

An Ante-hoc approach, which may also be referred to as directly interpretable [10], generates the explanation at the same time as the prediction is made, decision trees and rule-based models are examples of Ante-hoc explanation models.

Post-hoc

In contrast, a Post-hoc approach requires an additional operation after the predictions are made. Local Interpretable Model-agnostic Explanations (LIME) [102] is an example of producing a local explanation using a surrogate model applied following the predictor's operation.

2.5 Types of Explanations

Just because models such as deep neural networks are not interpretable does not mean they are not explainable. While complex models can be difficult for humans to understand, we can use techniques to generate explanations that show which parameters are used for decision-making and their relative importance to one another. Furthermore, we can infer which parameters are being used by considering what we know about the training data and the models' structures. Through these explanations, we can regain some of the transparency of non-interpretable models without sacrificing performance [111].

2.5.1 Decision Tree Proxy Models One

This is one of the earlier methods developed for explaining neural networks to present them as decision trees. Initially, the Continuous/discrete Rule Extractor via Decision tree induction (CRED) method was used to translate shallow neural networks [107]. Deep Rule Extractor extended this method via Decision tree induction (DeepRED) to handle arbitrarily deep networks [136]. DeepRED uses a number of techniques to prune unnecessary branches from the resulting tree [111].

2.5.2 Additive Feature Importance

The SHapley Additive exPlanation (SHAP) framework presented in [81] is the first presented additive feature importance framework. SHAP generates explanations by calculating Shapley values, which are the additive importance that each feature of the input has on the output of the model. Shapley values are a concept that originated in game theory. In that context, they serve as a measure of how important each player's actions are to the outcome of the game. In the context of machine learning, the players are the input features, and the outcome is the result generated by the model.

2.5.3 Salience Mapping

Salience mapping was first conceptualized in [65], which is based on combining visual features that contribute to attentive selection such as color, intensity, orientation, and motion queues as attentive selectors into a single map [57]. The salience at a given position in the image is determined primarily by how different that position is from its surroundings regarding the attentive selectors being considered. Techniques such as Randomize Input Sampling for Explanation of black-boxes (RISE) [98]. RISE has been developed to use salience maps to explain model's behavior. As salience maps can be generated independent of a classifier, one can view them as an explanation of how a model should treat a given image.

2.5.4 First Derivative Saliency

This approach is also called the attribution explanation method, which estimates the input contribution towards the output by computing the partial derivative of the predicted output w.r.t the input. This approach is closely related to older concepts such as sensitivity [106]. First derivative saliency is particularly convenient for ANN models as it can be computed for any layer using [116]. As suggested by its name and definition, first derivative saliency can be used to enable feature importance explainability. [23]. Gradient-weighted Class Activation Mapping (Grad-CAM) is an attribution method to calculate each input's contribution to the output. Grad-CAM explains the output layer decision by using gradients flowing into the last layer of the an ANN [45], and it can be applied on various CNNs with fully connected layers. It produces a localization map highlighting the important regions in the image and uses the gradient of the target class, flowing into the final convolutional layer. In other words, Grad-CAM takes the gradient of the target class w.r.t feature maps of the last convolutional layer, while the gradients of all other predicted classes are zero. Finally, these gradients capture the importance of each feature map for the target class. Although a saliency map computed by taking a gradient provides a good explanation of the important pixels of a given image, this approach requires access to the internal of the base model, such as the gradient of the output to the input, intermediate feature maps, or the network's weights. Many types of research focused on Grad-CAM and its extensions like Grad-CAM++. [35, 70, 101]. We describe these methods in detail in Chapter 3.

2.5.5 Layer-wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) is a technique that explains the individual decisions of a model by propagating the prediction from the output backwards to the input, which informs us to what extent the input features affected the final decision [11]. In other words, LRP has been used to enable feature importance explainability in different layers of an ANN models [21, 99]. As it only takes two passes through the model, one forward and one backward, we can apply it to a range of connectionist models. LRP is flexible and computationally efficient. This approach reveals attribute relevance to features computed in an intermediate layer of an ANN. It is the most common approach in ANN layers, including fully connected layers, convolution layers and recurrent layers.

2.5.6 Perturbation-based Approach

One of the approaches to faithfully explain the predictions of any black-box model in an interpretable manner is called Local Interpretable Model-agnostic Explanations (LIME). LIME takes an instance and creates some neighbours around that, called perturbations, then builds a locally linear model around the predictions of an opaque model to explain it. LIME pioneered the perturbation-based approach, and it is generating random perturbations of the input x and training an explainable model, which is usually a linear model, on the perturbations, and the predicted labels [102]. Perturbation-based approaches are mainly used to enable surrogate models [6]. As this is the explanation framework that we use in this research, we will discuss this method in greater detail in Chapter 4.

2.5.7 Model-Agnostic Explanation

The core of model-agnostic explanations is to evaluate the importance of features with respect to the prediction. In other words, model-agnostic explanation probes the black-box model by observing the probability change on the predicted class when erasing a specific region of an input, like an image [17]. The most used model-agnostic explanation approach is LIME which estimates individual feature contribution locally [86, 102].

Chapter 3

Literature Review

3.1 Visual Explanations from CNNs

CNNs have enabled unprecedented bwreakthroughs in a variety of computer vision tasks, from image classification [51,67] to object detection [42], semantic segmentation [78], image captioning [19,32,59,128], and visual question answering [8,40,83,100]. Although these DNNs enable superior performance, their lack of decomposability into intuitive and understandable components makes them hard to interpret [75]. Visual explanation techniques are good explanation tools to achieve model-agnostic explanations. Representative works in this area can be found in [20], which presents a portfolio of visualization techniques to help in the explanation of a black-box model built upon the set of extended techniques mentioned earlier. Visual explanations are less common in model-agnostic techniques for post-hoc explainability. Since the design of these methods must ensure that they can be seamlessly applied to any model disregarding its inner structure, creating visualizations from just inputs and outputs of an opaque model is a complex task. This is why almost all visualization methods in this category work along with feature relevance techniques, which provide the information that is eventually displayed to the end-user. This Chapter presents state-of-the-art XAI methods that produce visual explanations quite well for image classification problems.

3.1.1 Occlusion Maps

One obvious choice for such a visualization of a CNN's decision is image occlusion maps [134]. This method measures the difference in CNN scores when patches of the input image are masked. More precisely, occluding patches and classifying the occluded image, typically resulting in lower classification scores for relevant objects when those objects are occluded.

3.1.2 Guided Backpropagation

Several previous works [39,112,113,134] have visualized CNN predictions by highlighting important pixels, i.e., change in intensities of these pixels have the most impact on the prediction's score. The one that visualize partial derivatives of predicted class scores w.r.t. the pixel intensities is [112], which proposed guided backpropagation. This approach adds an additional guidance signal from the higher layers to usual backpropagation. This prevents backward flow of negative gradients, corresponding to the neurons which decrease the activation of the higher layer unit we aim to visualize. Although this method produces fine-grained visualizations, it is not class discriminative [110]. Visualizations w.r.t. different classes are nearly identical.

3.1.3 Class Activation Mapping

Class Activation Mapping (CAM) has been proposed by [135] for identifying discriminative regions used by a restricted class of image classification using CNNs that do not contain any fully connected layers. A class activation map for a particular category indicates the discriminative image regions used by the CNN model to identify that category. CAM essentially trades off model complexity and performance for more transparency.

However, as mentioned, CAM cannot explain CNNs with fully connected layers, and it is the main limitation of CAM, resulting in proposing new extensions. Another limitation of CAM is that it requires feature maps to precede softmax layers directly, so it is only applicable to a particular kind of CNN architecture performing global average pooling over convolutional maps immediately before prediction.

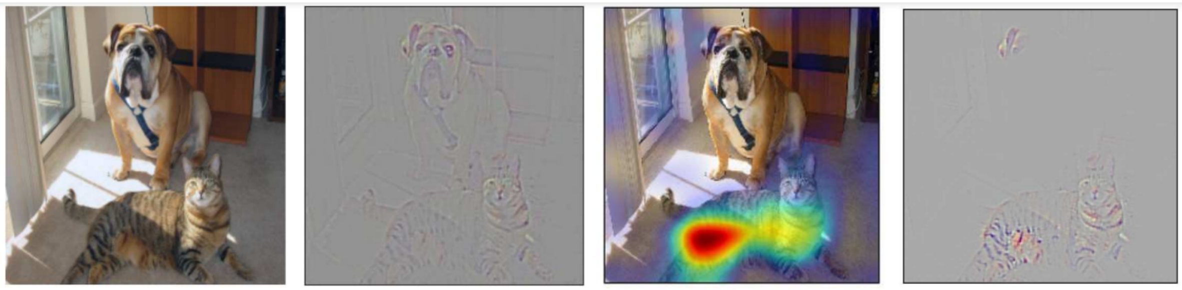
3.1.4 Gradient-weighted Class Activation Mapping

To make existing DNNs more interpretable without altering their architecture, Gradient-weighted Class Activation Mapping (Grad-CAM) has been proposed [110]. Grad-CAM is a visual explanation approach that uses the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image toward the specific concept.

Grad-CAM locates certain areas of an input image so that CNN can identify the objects using feature maps. This algorithm requires a linear combination of feature maps, with each feature map's weight determined by the mean of its gradients. This concept is based on CAM, and Grad-CAM is an extension of that method. Unlike previous approaches, Grad-CAM is applicable to a wide variety of CNNs, like CNNs with fully connected layers, CNNs used for structured outputs, e.g., captioning, and CNNs used in tasks with multi-modal inputs, e.g. visual question answering, or reinforcement learning without architectural changes or retraining [110]. To summarize, Grad-CAM uses the gradient information flowing into the last convolutional layer of the CNN to understand the importance of each neuron for a decision of interest and produce a coarse localization map highlighting the important regions in an input image.

3.1.5 Guided Grad-CAM

While Grad-CAM visualizations are class-discriminative and localize relevant image regions well, they lack the ability to show fine-grained importance like pixel-space gradient



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d) Guided Grad-CAM 'Cat'

Figure 3.1: : (a) Original image with a cat and a dog. (b) Guided Backpropagation: highlights all contributing features. (c) Localizes class-discriminative regions. (d) Guided Grad-CAM, which gives high-resolution class-discriminative visualization.

visualization methods. To better understand the visualizations provided by each of these approaches, Fig 3.1 has been taken from [110]. In this experiment, the author provided three localization maps for a sample image of a cat and dog, and the network for investigation is ResNet50. As (b) and (c) in Fig 3.1 show, guided backpropagation is not class discriminative, and Grad-CAM provides a class-discriminative coarse localization map, but it does not show the fine-grained details of important pixels/regions. Grad-CAM can easily localize the cat region and provide class-discriminative by a coarse heatmap localization. However, it is unclear from the low-resolutions of the heatmap why the network predicts this particular instance as *tiger cat*. In order to combine the best aspects of both, Guided Grad-CAM uses guided backpropagation and Grad-CAM visualizations via point-wise multiplication, see image (c) in Fig 3.1. This visualization is both high-resolution and class-discriminative, when the class of interest is *tiger cat*, and it also identifies important *tiger cat* features like stripes, pointy ears and eyes.

3.1.6 Grad-CAM++

While the visualizations generated by gradient-based methods such as Grad-CAM explain the prediction made by the CNN model with fine-grained details of the predicted class, these methods have limitations. For example, their performance drops when localizing multiple occurrences of the same class [16]. In addition, for single object images, Grad-CAM heatmaps often do not capture the entire object, which is required for better performance on the associated recognition task. To address these limitations, Grad-CAM++ has been proposed as a generalized visualization technique for explaining CNN decisions, which improves the flaws mentioned above and provides a more general approach. Grad-CAM++ is known as a further upgraded version of Grad-CAM and it is a very good algorithm among most well-known visual interpretation methods. For sanity check-based tasks, Grad-CAM++ yields perfect results among the recently studied state-of-the-art methods [55]. More precisely, this approach is a pixel-wise weighting of the gradients of the output concerning a particular spatial position in the final convolutional feature map of the CNN. This approach gives the importance of each pixel in a feature map toward the overall decision of the CNN. Importantly, Grad-CAM++ derives closed-form solutions for the pixel-wise weights and obtains exact expressions for higher-order derivatives. Furthermore, Grad-CAM++ requires a single backward pass on the computational graph, making it computationally equivalent to prior gradient-based methods while giving better visualizations. Grad-CAM++ resolved the main limitation of Grad-CAM as it can localize multiple objects in an individual image, but it still needs access to the inner structure of the black-box model.

3.1.7 LIME

LIME is an XAI approach relying on segmenting images into superpixels based on the Quick-Shift algorithm [1]. LIME is an explanatory framework for the decision of any ML classifier. The original implementation can process classifiers with text, images, or tabular data as input. For example, in the case of explaining the decision of a CNN in image classification with LIME, the output of LIME is a set of connected pixel patches and a weighting for each patch. These weights indicate how strong a patch is correlated with the classifier decision [108]. For images, this is achieved by randomly removing patches from the image and replacing them with the mean color of the patch or with some chosen color, default is grey. Then every instance, as a perturbed version of the original image will be measured by the proximity measure that indicates how different the perturbed image is from the original instance. This proximity measure, called distance measure, is used to enforce locality for the linear model [102]. Finally, each of these perturbed images will be classified by the CNN model, and then the linear model will be fitted to these experiments. The linear model approximates the weights/importance for each patches by selecting k features with Lasso, called K-Lasso [102]. The weights (w) are ultimately found through K-Lasso, a procedure that is based on the regression method [118]. The input is the number of features limit k which is the number of patches the user wants in the explanation. LIME relies on segmentation of the image into superpixels, that is on similarity based grouping of pixels into larger structures based on local features [37]. The segmentation of an image into superpixels is crucial for the generation of the explanation in LIME since perturbation of superpixels is used to identify which of the image areas has been relevant for a specific class decision [108].

An explorative study of how different superpixel methods impact the generated visual explanations of LIME has been done in [108]. This study compared superpixel segmentation algorithms, namely Felzenszwalb, Simple Linear Iterative Clustering Segmentation (SLIC) and Compact-Watershed, and their specific effects on LIME. This comparative study reveals that SLIC makes it possible to influence the actual size of the superpixels through a parameter. Additionally, a lower variance and standard deviation was achieved. These results show that SLIC has advantages over Quick-Shift due to showing a better correspondence between superpixels and relevant areas. We explain the SLIC segmentation algorithm in detail in Chapter 4.

3.2 Quantifying Explainability of Saliency Methods

Visual explanations of a CNN model for justifying any target category can be checked quantitatively through each of the following metrics.

- Structural SIMilarity index (SSIM): The main question is whether different XAI methods point to the same input regions to explain a given prediction. It measures the similarity of different XAI methods or even LIME with different segmentation algorithms [46]. SSIM has also been used to obtain the best values of parameters. It measures the similarity of the visual explanations obtained from LIME with different sets of parameters, e.g., whether SSIM changed when the number of superpixels and number of samples differ by a shift of n [46].

- Explanation accuracy: Instead of prediction accuracy (pa), when the entire sample image is passed to the black-box model, the obtained feature subset, containing the most important regions, is passed to the same black-box model. Then, the prediction label is observed and noted as an explanation accuracy (ea). If ea increases than the pa, we can state the explanation is correct, and it contains the most important regions of the input image. In other words, the explanation method captures the extent to which the explanations are truly influential toward the prediction [103].

- Faithfulness: Saliency methods generates relevance scores w.r.t. model predictions assigned to the features, pixels or superpixels for images. Described in [5], the faithfulness of an explanation refers to whether the relevance scores reflect the true importance. A typical approach for quantifying the property of XAI is through strategical modification on the input according to the explainer indication and monitor the model behaviors. There are several different metrics proposed based on this approach. For example, measuring Area Under Curve (AUC), which is motivated by deletion or insertion. The intuition behind the deletion metric is that removing the cause, important pixels, will force the model to change its decision. A sharp drop, thus a low area under the probability curve. [98].

- Sensitivity check: Sensitivity check means localizing the specific region of the target in the image. If an explanation is faithful, it should give different explanations for different decisions. The similarity measure of the highest prediction score and the lowest one should be close to zero, e.g., Pearson correlation [72].

- Localization: Basically, localization means capturing fine-grained details, and highlight important pixels in the image. It is also called *high-resolution*, which states that a good saliency map should highlight a few discriminative superpixels rather than the entire object in an input image [46].

To summarize, a good visual explanation should localize important regions of an image that correspond to any decision of interest in high-resolution detail, even if the image contains evidence for multiple possible objects, which is important for predicting a particular object.

Chapter 4

Methodology

4.1 Introduction

According to the literature, the most leading and common XAI tools in medical domain is the local explanation methods [29,60,61,101,109]. The gradient-based methods got more attention due to their accuracy, but there is no specific research on LIME to improve its performance in localizing multiple objects.

With the idea of producing fine-grained visual explanations of medical images, the results of LIME have been investigated for classification of lymph node metastases on Camelyon16 dataset in [95] using three well-known segmentation algorithms, including Felzenszwalb's, SLIC, and Quickshift. All three algorithms have a parameter "*sigma*" that defines the width of a Gaussian preprocessing step. Higher sigmas typically result in a smaller number of segments. This is the only parameter shared in common by these algorithm, and each of these algorithms has several parameter affecting the explanation by LIME. The sensitivity of these algorithms to variations in texture and color and their own variables raises the question of how the best set of parameters can be determined. As an alternative approach, [95] proposed a new method to segment each studied image into grids of 9, 16, 36, 64, 144, 256, and 576 equally-sized squares. On the one hand, these segments did not hold the contextual meaning of a typical superpixel, but scheme guaranteed that every image was divided into exactly the same number of segments on the same positions. These square segments were then passed to the LIME algorithm like the usual superpixels would be. The resulting weighted heatmaps gave a rough idea of what sub-regions of the image were most relevant for a given classification. The finer grids provided a more fine-grained view with a larger number of small square segments.

Due to the importance of XAI to explain decisions of DL approaches in different areas, specifically medical diagnosis, and the increasing demand for such explainable models in health-related applications, we proposed a multi-scale scheme of LIME as a model agnostic explanation tool.

4.1.1 Motivation: Multi-scale Version of LIME

Although in the square-grid segmentation approach, the image will be divided into the same number of segments with the same probability to be masked, but this approach does not hold the main contextual meaning of a segmentation task called boundary recall. A good segmentation approach with higher boundary recall will stick to image boundaries. According to the literature [1], Simple Linear Iterative Clustering Segmentation (SLIC) is a good superpixel algorithm with high boundary recall and low under-segmentation error, as we explained in Section 4.5. With this idea, we were motivated to propose a multi-scale version of LIME using the SLIC segmentation algorithm and providing visual explanations at a coarse, finer, and the finest scales of an image. In this Chapter, we first explain what LIME sees in an image, and how it works from the mathematical point of view in Section 4.2. Then, we described sampling in LIME by turning on/off some superpixels, creating perturbations, and predicting the probability of the target class for each perturbation in Section 4.3. Finally, the surrogate model is explained in Section 4.4, besides the important remarks that helped us define the main contribution of this work.

4.2 Mathematical Function of LIME

As we mentioned in Chapter 3, there are some methods for visual explanations that are gradient-based, and the explanations generated in these cases are heatmaps where each pixel in the image receives a value according to its relevance for a given classification. Due to the computation complexity of gradient-based XAI methods and sometimes modifying the last layers of several CNN models, model-agnostic explanation tools are much easier to implement, specifically for a wide range of black-box models. In this thesis, we focus on LIME as a model-agnostic XAI method as it only requires the classifier's outputs for different images. In this case, a given image is segmented into superpixels, and the relevance of each superpixel for a given classification is determined using a linear model. LIME can be used for any image classifying system, not just neural networks, as it does not employ specific steps to any individual model type.

In this Section, we describe the mathematical function of LIME and explain what does LIME see in a sample image from a mathematical point of view [41, 102].

Formally, we define an explanation as a model $g \in G$, where G is a class of potentially interpretable models, such as linear models, and decision trees. i.e., a model $g \in G$ can be readily presented to the user with visual or textual artifacts. The domain of g is $\{0, 1\}^{d'}$. In other words, g acts over absence/presence of the interpretable components in binary form. As not every $g \in G$ may be simple enough to be interpretable, thus $\Omega(g)$ is a measure of complexity, which is opposed to interpretability of the explanation $g \in G$. For example, for decision trees $\Omega(g)$ may be the depth of the tree, while for linear models, $\Omega(g)$ may be the number of non-zero weights.

- Let the model being explained be denoted $f : \mathbb{R}^d \to \mathbb{R}$.

  In classification, $f(x)$ is the probability that x belongs to a certain class.

- i is an index of the number of samples, $1 \leq i \leq n$, and $\pi_i(x)$ is a proximity measure between an instance i to x, so as to define locality around x.

- Finally, let $L(f, g, \pi_i)$ be a measure of how unfaithful g is in approximating f in the locality defined by π_i.

In order to ensure both interpretability and local fidelity, LIME minimizes $L(f, g, \pi_i)$ while having $\Omega(g)$ be low enough to be interpretable by humans [41]. The explanation produced by LIME is obtained by

$$argmin\ \ L(f, g, \pi_i) + \Omega(g). \tag{4.1}$$

This formulation can be used with different explanation families G, fidelity functions L, and complexity measures Ω. LIME uses sparse linear regression, Lasso, but in this thesis we used Ridge regression as the explanation model and performed the search using perturbations.

4.3 Sampling for Local Exploration

LIME wants to minimize the locality-aware loss $L(f, g, \pi_i)$ without making any assumptions about f, since the explainer is model-agnostic. Thus, in order to learn the local behavior of f as the interpretable inputs vary, we approximate $L(f, g, \pi_i)$ by drawing i samples, weighted by π_i.

In order to explain what does LIME do on an image, we consider a model f for sample image ξ. LIME

- decomposes ξ in d superpixels, that is, small homogeneous image patches.

- creates i number of new images $x_1, ..., x_i$ by randomly turning on and off some of these superpixels. In other words, sampling instances around x by drawing nonzero elements of x uniformly at random.

- queries the model, and gets predictions $y_i = f(x_i)$. Basically, LIME recovers the sample image from binary domain $\mathbb{R}^{d\prime}$ to the original representation domain $z \in \mathbb{R}^d$, and obtain $f(z)$, which is used as a label for the explanation model. Given this dataset Z of perturbed samples with associated labels, LIME can optimize equation 4.1.

- builds a local weighted surrogate model $\hat{\beta}$, fitting the y_is to the presence or absence of superpixels.

The primary intuition behind LIME is presented in Fig 4.1, where sample instances in the vicinity of x have a high weight due to π_i and far away from x have low weight, π_i [102]. Even though the original model may be too complex to explain globally, LIME presents a locally faithful explanation, where the locality is captured by π_i. LIME is robust to sampling noise since the samples are weighted by π_i in equation 4.1. We now present a concrete instance of this general framework.

According to LIME, each coefficient in $\hat{\beta}$ is associated to a superpixel of the original image ξ. The more positive the more important the superpixel is for the prediction at ξ. Generally, LIME visualizes $\hat{\beta}$ by highlighting the superpixels associated to the top positive coefficients. This top positive coefficients is a parameter k in LIME. From now on, we consider a model $f : [0,1]^D \to \mathbb{R}$ as well as an image example to explain $\xi \in [0,1]^D$. D denotes the number of pixels of the sample image on which model f operates. Generally, the inputs of f are always 2- or 3- dimensional arrays. In particular, grayscale images are usually encoded as $h \times w$ arrays, and RGB images are represented by $h \times w \times 3$, when each channel represent to a primary color.

4.3.1 Superpixels

At the first step, LIME splits the image ξ into d number of superpixels. According to the definition of superpixels, these are contiguous patches of the image that share color

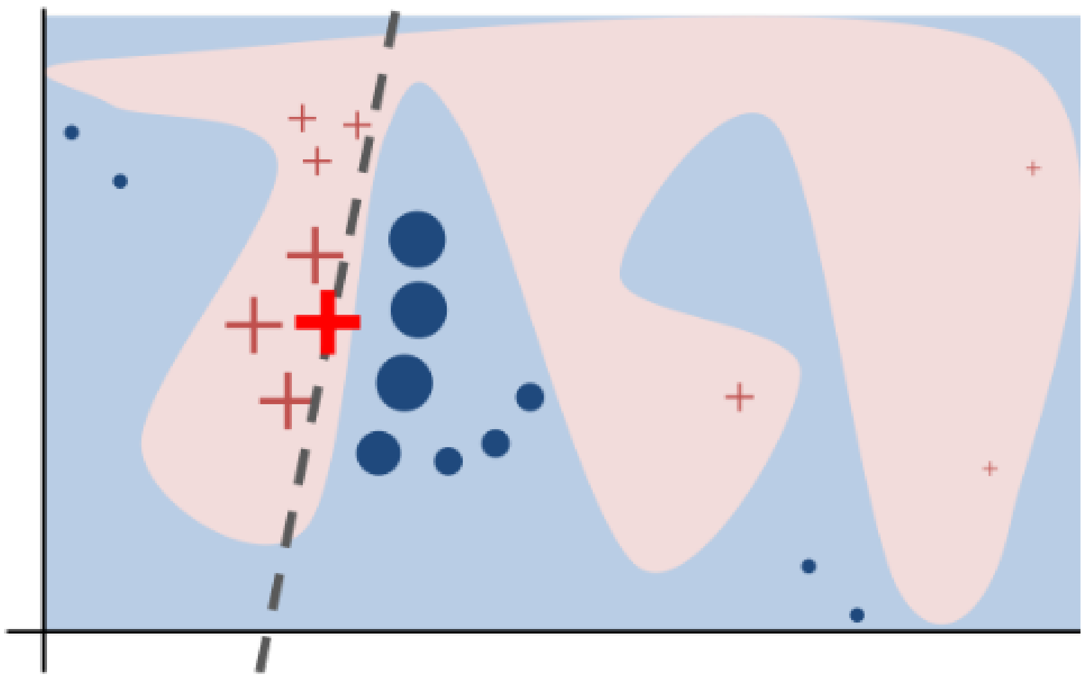


Figure 4.1: The black-box model's complex decision function f which is unknown to LIME is represented by the blue/pink background. It cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained, which are represented here by size. The dashed line is the learned explanation that is locally faithful.

and/or brightness similarities. For any $1 \leq k \leq d$, kth superpixel associated to the input image ξ by J_k. Therefore, the d subsets $J_1, ..., J_d$ form a partition of the pixels. LIME masks some of these superpixels by randomly setting them to zero to create perturbations.

4.3.2 Perturbations

One of the LIME's key ideas is to create n new samples from the input ξ by randomly replacing/masking some superpixels of the image. To be more precise, we assume that ξ is fixed, and $J_1, ..., J_d$ are given. Then, the first step of taking samples is to compute the replacement image. For each sample $1 \leq i \leq n$, LIME samples a random vector $z_i \in \{0, 1\}^d$ where each coordinate of z_i is independent and identically distributed (i.i.d.) Bernoulli with parameter 1/2. Each $z_{i,j}$ corresponds to the activation or inactivation of superpixel j in sample i.

Fig 4.2 shows superpixel segmentation with Quickshift on a sample image of *tulips*. Let's assume this is a classification problem, and the model predicts the label *tulips* for this specific image. To see how we can create perturbations and the corresponding labels, we used Quickshift to segment out the image, then randomly turned on/off some of these superpixels Fig 4.2 (b), (c), and (d). These samples are passed to the model, and the corresponding labels are given. As it is obvious, active superpixels in images (b) and (c) show more regions of *tulips* compared to the image (d). The given probabilities by the model also prove this fact.

4.3.3 Weights

The new sample images x_i can be quite different from the original image. For instance, if most of the superpixels are activated, which means most of the $z_{i,j}$ are 1, then the sample x_i is close to the original image ξ. Therefore, the new instance will be given a positive weight π_i that takes the proximity into account based on equation 4.2. The weights are defined as

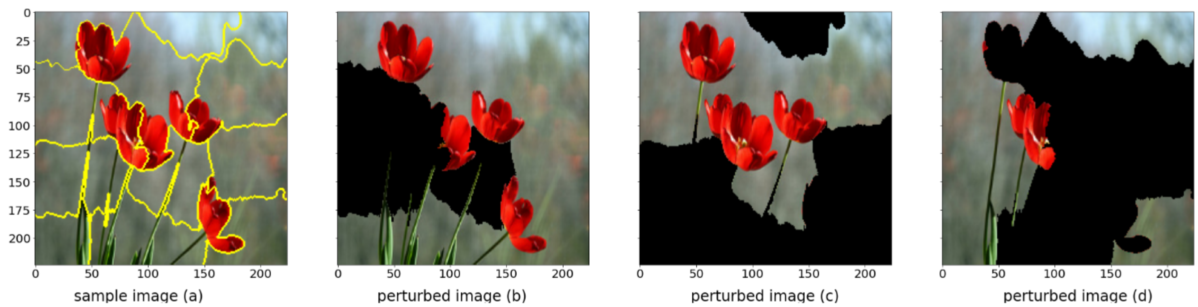


Figure 4.2: Segmentation of a sample image (a) with Quickshift, and the perturbation samples by randomly masking some superpixels. LIME queries the model to predict the label for each of these samples. The prediction score of the label "tulips" for perturbed images (b), and (c) are 0.97 and 0.99, respectively. But in the perturbed image (d), it is visually clear that most of the regions representing tulips have been masked, so the predicted label of this sample should be significantly low, which is 0.1.

$$\forall 1 \leq i \leq n, \quad \pi_i := exp(\frac{-d_{cos}(1, z_i)^2}{2\upsilon^2}), \tag{4.2}$$

where υ is a positive bandwith parameter, called "kernel-width", equal to 0.25 by default, and d_{cos} is the cosine distance. We see that,

- $d_{cos}(1, z_i)$ takes near 0 values if most of the superpixels are activated.
- and values near 1 in the opposite scenario, as expected.

4.4 Surrogate Model

The last stage of LIME is to build a surrogate model around the samples taken from the input. LIME builds a linear model with the interpretable features z_i as input and the model predictions $y_i := f(x_i)$ as responses. This linear model, in the default implementation, is obtained by (weighted) Ridge regression [53]. Finally, in equation 4.3, the output of LIME for model f and image ξ are given by

$$\hat{\beta^{\lambda}} \in argmin\{\sum_{i=1}^{n} \pi_i (y_i - \beta^T z_i)^2 + \lambda ||\beta||^2\}, \tag{4.3}$$

where $\hat{\beta^{\lambda}}$ are the interpretable coefficients, the $0th$ coordinate of $\hat{\beta^{\lambda}}$ is the intercept of the model, and $\lambda > 0$ is a regularization parameter.

4.5 Segmentation Algorithm

4.5.1 Simple Linear Iterative Clustering Segmentation

Simple Linear Iterative Clustering (SLIC) performs a local clustering of pixels in the 5-D space defined by the L, a, b values of the CIELAB color space and the x, y pixel coordinates. The distance measure of SLIC enforces compactness and regularity in the superpixel shapes and seamlessly accommodates grayscale as well as color images. SLIC is simple to implement and easily applied in practice. The only parameter specifies the desired number of superpixels. SLIC is significantly more efficient than competing methods while producing segmentations of similar or better quality as measured by standard boundary recall and under-segmentation error measures [84]. Compact and highly uniform superpixels that respect image boundaries, such as those generated by SLIC in Fig 4.3, are desirable for many vision tasks. SLIC superpixels outperforms competing methods for two vision tasks: object class recognition and medical image segmentation [1]. In both cases, SLIC results in similar or greater performance at a lower computational cost than existing methods [1]. Superpixel algorithms need to be easy to use. Difficult-to-set parameters can result in lost time or poor performance. A good superpixel segmentation algorithm should have low under-segmentation error and high boundary recall. To be useful as a pre-processing algorithm, such a segmentation should result in equally sized compact superpixels with control over their number [95]. For the same reason, the algorithm should preferably have a low computational cost and require few input

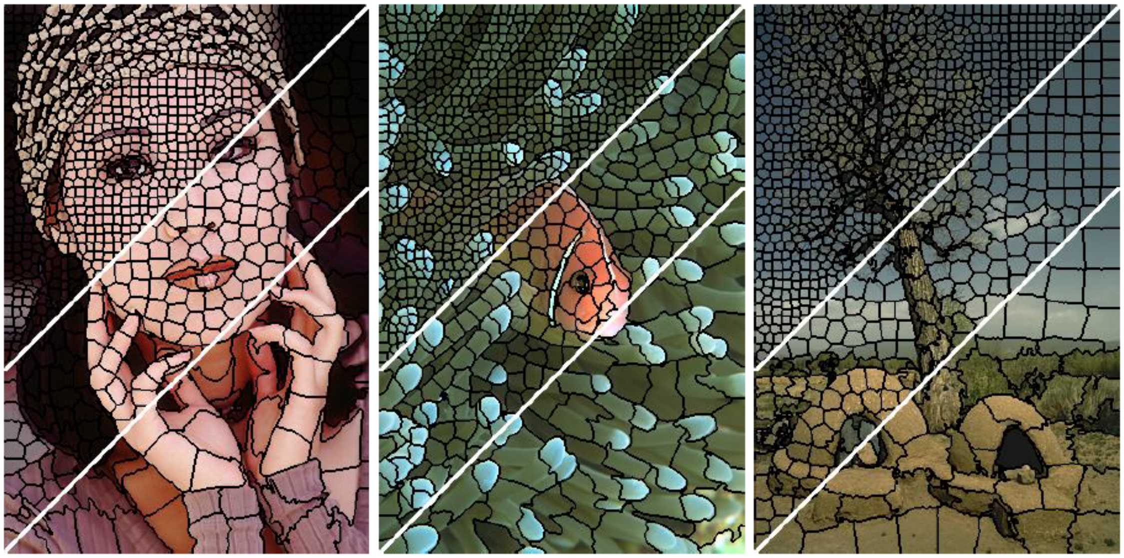


Figure 4.3: Image segmented using SLIC algorithm into superpixels of size 64, 256, and 1024 pixels approximately. These are the desired number of superpixels defined by the user, and the model outputs a possible number of superpixels which is closest to the desired one. The superpixels are compact, uniform in size, and adhere well to region boundaries.

parameters. SLIC generates superpixels by clustering pixels based on their color similarity and proximity in the image plane. SLIC algorithm is defined in more detail in the following [1].

Distance Measure

SLIC algorithm takes the parameter k as a desired number of approximately equally-sized superpixels. For an image with N pixels, the approximate size of each superpixel is therefore N/K pixels. For roughly equally-sized superpixels there would be a superpixel center at every grid interval $S = \sqrt{N/K}$.

At the onset of this algorithm, it choose K superpixel cluster centers,

$$C_k = [l_k, a_k, b_k, x_k, y_k]^T, \tag{4.4}$$

with $k = [1, K]$ at regular grid intervals S. Since the spatial extent of any superpixel is approximately S^2, we can safely assume that pixels that are associated with this cluster center lie within a $2S \times 2S$ area around the superpixel center on the xy plane. This becomes the search area for the pixels nearest to each cluster center.

Euclidean distances in CIELAB color space are perceptually meaningful for small distances. Suppose spatial pixel distances exceed this perceptual color distance limit. In that case, they begin to outweigh pixel color similarities (resulting in superpixels that do not respect region boundaries, only proximity in the image plane). Therefore, instead of using a simple Euclidean norm in the 5D space, SLIC uses a distance measure D_s defined as

$$
\begin{aligned}
d_{lab} &= \sqrt{(l_k - li)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \\
d_{x,y} &= \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \\
D_s &= d_{lab} + \frac{m}{S} d_{xy},
\end{aligned}
\tag{4.5}
$$

where D_s is the sum of the lab distance and the xy plane distance normalized by the grid interval S. A variable m is introduced in D_s allowing us to control the compactness of a superpixel. The greater the value of m, the more spatial proximity is emphasized and the more compact the cluster. According to the literature, this value can be in the range [1, 20] [1]. Fig 4.4 shows SLIC segmentation with different values of compactness m for the sample image "*tulips*". As we want to explore small regions of an image in a multi-scale scheme, the superpixels should have more spatial proximity and proper compactness. A higher compactness value results in square-grid shape segments, which do not necessarily stick to the object's boundaries.

In this thesis we aim to segment a sample image into several steps, going from coarse to finest scales. Therefore, we hypothesize that the segmentation approach should stick to the object's boundaries so that small regions of a finer scale segmentation would still be meaningful.

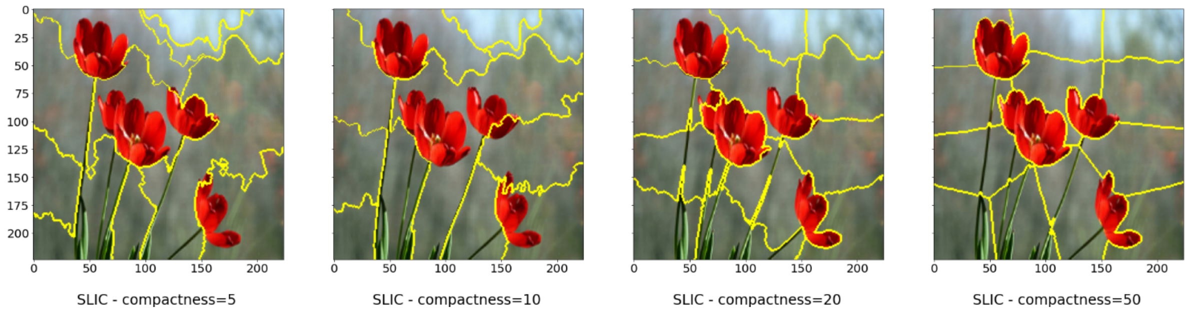


Figure 4.4: SLIC segmentation of sample image with varying compactness. The higher value of this parameter results in the square-grid shape of segments.

Under-segmentation Error

Under-segmentation error was first introduced in [71] and quantifies the leakage of superpixels across ground truth segments. Under-segmentation error essentially measures the error an algorithm makes in segmenting an image w.r.t. a known ground truth, human segmented images in this case. This measure thus penalizes superpixels that do not tightly fit a ground truth segment boundary. Given ground truth segments $g_1, g_2, ..., g_m$ and a superpixel output $s_1, s_2, ..., s_L$, the under-segmentation error for a ground truth segment g_i is

$$U = \frac{1}{N}(\sum_{i=1}^{M}(\sum_{[s_j|s_j \cap g_i > B]} |s_j|) - N); \tag{4.6}$$

where s gives the size of the segment in pixels, N is the size of the image in pixels, and B is the minimum number of pixels that need to be overlapping. The expression $s_j \cap g_i$ is the intersection or overlap error of a superpixels s_j with respect to a ground truth segment g_i. B is set to be 5 percent of $|s_j|$ to account for small errors in ground truth segmentation data. The value of U is computed for each ground truth image and then averaged. The lower under-segmentation error means better segmentation algorithm.

Boundary Recall

Boundary Recall is part of the precision-recall framework introduced in [85] and quantifies the fraction of boundary pixels correctly captured by a superpixel segmentation. Higher boundary recall describes better adherence to image boundaries. In other words, the standard boundary recall measure computes what fraction of ground truth edges fall within one pixel of a least one superpixel boundary.

4.6 Multi-scale Segmentation Scheme

Choosing an appropriate superpixel segmentation algorithm is so important in preprocessing step for computer vision applications like object class recognition and medical image segmentation. Such segmentation algorithms should output high-quality superpixels that are compact and roughly equally sized for low computational overhead. SLIC is a simple superpixel segmentation algorithm to implement and output better quality superpixels. It needs only the number of desired superpixels as the input parameter. As a result, it scales up linearly in computational cost and memory usage. The efficacy of SLIC superpixels in object recognition and medical image segmentation has been proved in [1]. SLIC obtains better quality and higher computational efficiency than other state-of-the-art algorithms. In this work, we have decided to use the SLIC segmentation algorithm and specify two main parameters: compactness and the desired number of superpixels. We aimed to segment an image into a multi-scale scheme to look at the superpixels of an image from coarse to finer scales by keeping the boundaries and sticking to the contextual meaning of superpixels. The pseudo-code of the proposed multi-scale scheme is shown in Algorithm 1, and the list of notations is presented in Table 4.1.

Due to some important remarks in the literature [41]:

- When we experiment on a large n number of samples, basically when the number of samples is 10 times larger than the number of superpixels that an image is split up in, we can consider that $\lambda = 0$, the regularization parameter in LIME, and still get meaningful results.

- Part of what makes LIME attractive is to visualize the results of LIME and look at the highlighted part of the image. The final step of LIME for images is to display the k superpixels associated to the top positive coefficients of $\hat{\beta}_n^{\lambda}$, usually it is five.

Hence, we define the *number of samples* to be $10 \times$ *number of superpixels* at each level, we can use LIME without regularization and get meaningful results. Since we aimed to present the visual explanations of the proposed scheme on heatmaps that highlight all superpixels along with the corresponding coefficients that show the importance of each region, we do not need to specify the parameter k. Furthermore, we focus on the importance of each superpixel toward the target class, which could be the top one or any of the predicted classes. We are normalizing the coefficients before showing them on heatmaps. In other words, we set the negative values of the output of LIME to zero and are not interested in evaluating superpixels that negatively affect the target class.

As the list of parameters are shown in Table 4.1, the proposed multi-scale scheme takes the desired number of segments/superpixels, $s_1, \cdots, s_L$, where L is the number of levels. Since we want to keep the objects' boundaries through SLIC segmentation algorithm, we should set c to be small enough. Hence, we set $c = 5$ in all experiments in the proposed multi-scale scheme of LIME. Then an iterative process executes inside the for loop in Algorithm 1. At each level l, $1 \leq l \leq L$, SLIC takes s_l as the desired number of superpixels and returns s_l' as the number of superpixels after the segmentation process. Then, the algorithm defines total number of samples/perturbations, n, and creates these neighbors by randomly masking some superpixels. A separate linear model is fitted at each level, then we take the coefficients, normalize the value of the coefficients in a range of (0,1) and represent the result on heatmaps.

Hence, the output is L number of heatmaps which represent superpixels which mostly

List of Notations / Parameters	
x	Input image of size $(h, w, 3)$.
L	Number of levels for multi-scale segmentation.
l	Index of segmentation level.
s	Desired number of superpixels.
c	Compactness.
s′	Final number of superpixels that SLIC provides at each level.
n	*Num-of-samples* = 10 * s′.
i	index of the number of samples.
p	Perturbation or masked image which is a matrix of size $(m, n, 3)$.
X	Binary matrix of size (n, s′).
Y	Vector of size (1, n): predicted label for each perturbation.
W	Vector of size (1, n): distance of perturbations from x.
coefficients	Vector of size (1, s′).
E	Explanation matrix of size (h, w) represents the coefficients.

Table 4.1: Parameters description

affected the prediction from coarse to fine scale. The results can be interpreted by an expert or final user by looking at the produced heatmaps. In order to look at interes objects in an image from different scales, like too coarse or too fine scales, the number of levels/scales in the proposed method might be large enough. In this case, the interpretation of the produced heatmaps would be difficult. For this reason, we were motivated to classify the heatmaps using a weighting approach. To classify the heatmaps and produce explanations at different scales, we proposed two weighting strategies in Section 4.7. The general framework of this method is also provided in Fig 4.5.

Algorithm 1 LIME with Multi-scale Scheme

Require: $s_1 \dots s_L$, and c

for l $\leftarrow$ 1 to L **do**

Apply SLIC segmentation algorithm with s_l, and c;

Take *'num of superpixels'* s_l';

Define *num-of-samples* n;

Create samples/perturbations $(p_1, ..., p_n)$;

Compute distance/weight of perturbations from input image $(w_1, .., w_n)$;

Predict label/class for each perturbation $(y_1, ..., y_n)$;

Fit linear model on set of perturbations X and predicted labels Y;

Take the coefficients of the linear model;

Do normalization of the coefficients;

Represent coefficients on explanation heatmaps $(E_1, ..., E_l)$;

Return the heatmap;

end for

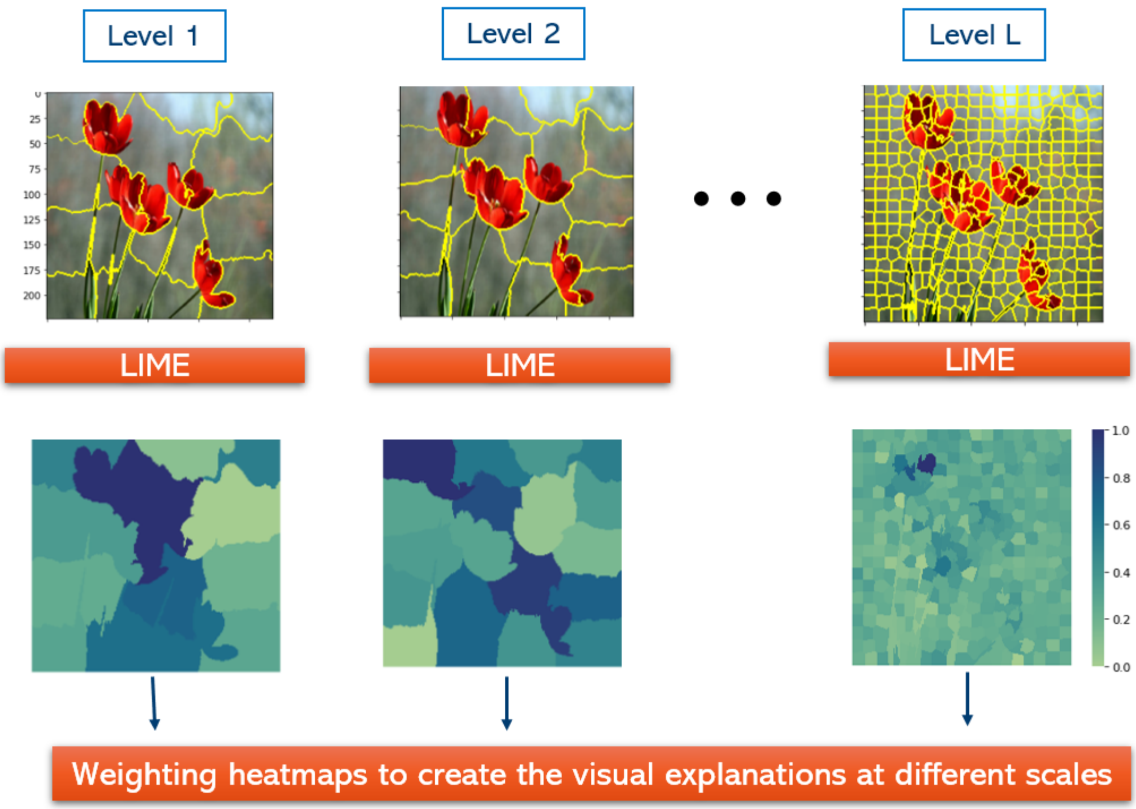


Figure 4.5: General framework of the proposed multi-scale scheme of LIME

4.7 Multi-scale Visual Explanation

To combine the visual explanations of the proposed multi-scale scheme in an appropriate way, we proposed two weighting approaches, using discrete Gaussian function, and a parameter-free approach. The main point of these two weighting approaches is to summarize the explanation heatmaps of different levels and highlight the most important superpixels/regions from coarse to finest scales.

4.7.1 Weighting Heatmaps with Discrete Gaussian Function

Gaussian functions are often used to represent the probability density function of a normally distributed random variable with expected value μ and variance σ^2. In this case, the Gaussian is of the form in equation 4.7

$$C = [1/(\sigma\sqrt{2\pi}](\exp(-(x-\mu)/2\sigma)^2), \forall c \in [-n/2, (n/2)+1]. \tag{4.7}$$

Normalized Gaussian curves with value μ and variance σ^2 are shown in Fig 4.6. Gaussian functions are widely used in statistics to describe the normal distributions, in signal processing to define Gaussian filters, in image processing where two-dimensional Gaussians are used for Gaussian blurs, and in mathematics to solve heat equations and diffusion equations.

This work aims to combine the heatmaps provided by the multi-scale version of LIME when we assign weights by discrete Gaussian function. We expect that mid-range scale experiments keep objects' boundaries and be more localized compared to the very first or last experiments. Hence, we set μ=0, and σ=5, as we want to assign higher weights to mid-range scale experiments for an arbitrary number of scales that we want to combine. In order to provide more variation in the explanations, we combine the heatmaps in three different levels described below. At each level, the mid-range scale experiments will get higher weights and the weights are becoming smaller from the sides. Assume L is the number of levels, therefore

- Coarse scale visual explanation: Weighted average of experiments 1 - int($L/3$)

- Finer scale visual explanation: Weighted average of experiments 1 - int($2L/3$)

- Finest scale visual explanation: Weighted average of experiments int($2L/3$) - l.

In this way, we use Gaussian distribution when σ is constant and shift the mean, μ, to the right to combine heatmaps of different scales.

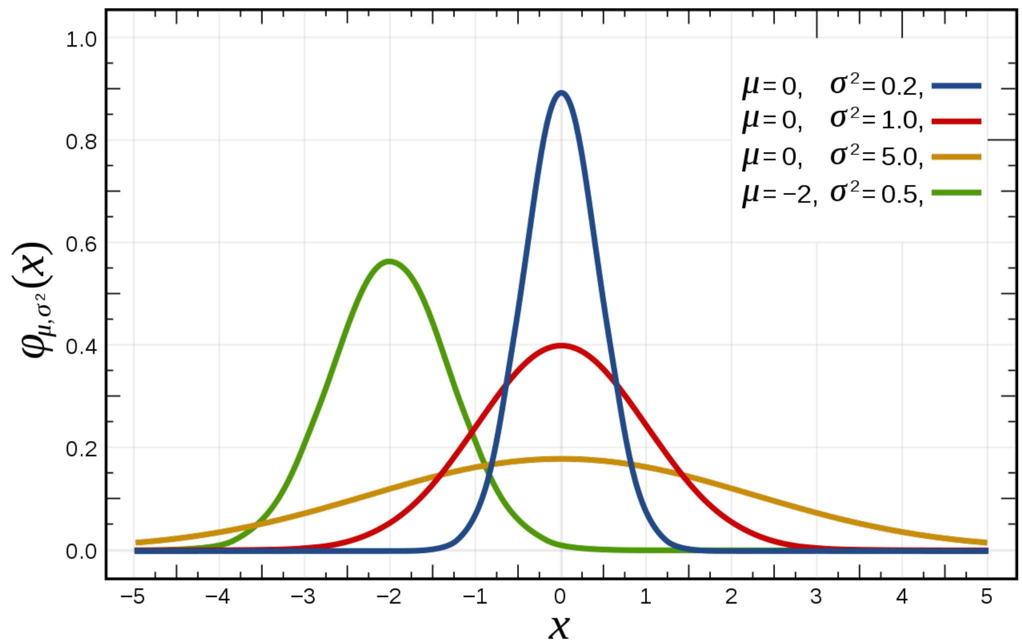


Figure 4.6: Gaussian curves with different values for mean and variance

4.7.2 A Parameter-free Automated Weighting Approach

Trapezoidal Rule

The trapezoidal rule is mostly used for approximating area under the curves. This is possible if we divide the total area into smaller trapezoids instead of using rectangles. The trapezoidal rule integration actually calculates the area by approximating the area under the graph of a function as a trapezoid.

In calculus, the trapezoidal rule is a technique for approximating the definite integral as its shown in Fig 4.7 and equation 4.8.

$$\int_a^b f(x)\,dx \approx (b-a)\,(f(a)+f(b)) \tag{4.8}$$

The integral can be even better approximated by partitioning the integration interval, applying the trapezoidal rule to each sub-interval, and summing the results. In practice, this chained/composite trapezoidal rule is usually what is meant by "integrating with the trapezoidal rule". Let $\{x_k\}$ be a partition of [a,b] such that $a = x_0 < x_1 < \cdots < x_{N-1} <$

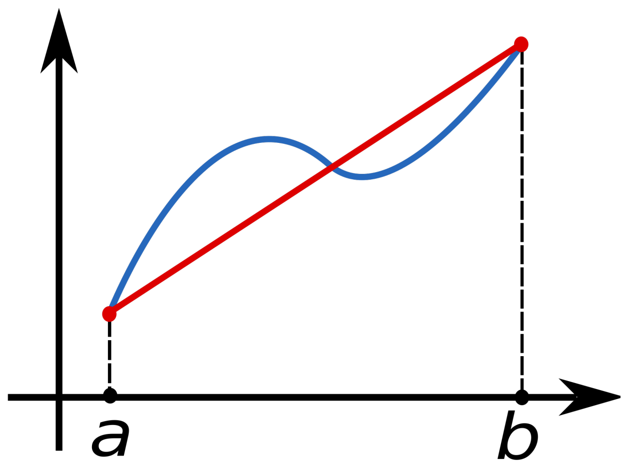


Figure 4.7: The function $f(x)$ in blue is approximated by a linear function in red.

$x_N = b$ and Δx_k be the length of the $k-th$ sub-interval. That is $\Delta x_k = x_k - x_{k-1}$, then

$$\int_a^b f(x)\,dx \approx \sum_{k=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2} \Delta x_k. \tag{4.9}$$

When the partition has a regular spacing, that means all the Δx_k have the same value Δx, the formula can be simplified for calculation efficiency by factoring Δx out as

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{2} \left(f(x_0) + 2f(x_1) + 2f(x_3) + \cdots + 2f(x_{N-1}) + f(x_N) \right). \tag{4.10}$$

When the grid spacing is non-uniform, one can use the formula of equation 4.9. Fig 4.8 shows approximating the definite integral with the irregularly-spaced partition of [a,b].

With the idea of approximating definite integral with the irregularly-spaced partition, we decided to assign weights to each scale. In other words, in segmentation of an image at different scales, the number of segments might vary according to the user-specified set of parameters. Therefore, the number of segments in this multi-scale segmentation is not specifically equal, and the spacing is non-uniform. Hence, we propose two different schemes to assign weights to the heatmaps given from different scales. Then, we compute the weighted heatmap to represent the results. We aim to assign weight to each heatmap

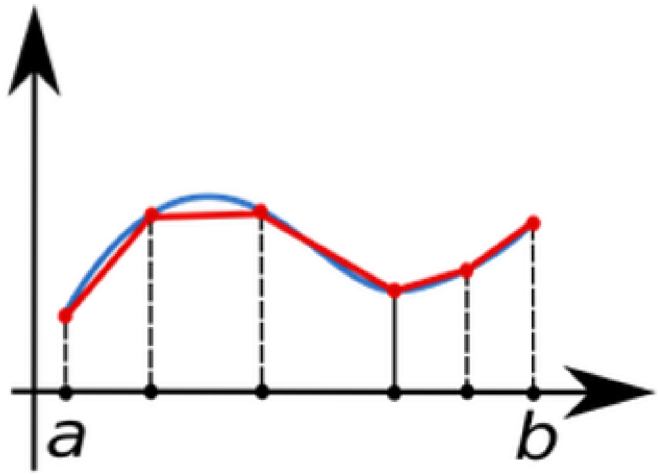


Figure 4.8: Illustration of "chained trapezoidal rule" used on an irregularly-spaced partition.

based on how far/close the number of segments is from the next level. Let's assume $1, \cdots, L$ is the number of scales that we want to segment out an image from coarse to fine, where $s'_1, \cdots, s'_L$ is the number of segments that SLIC provides at each scale. If s'_l is too smaller than s'_{l+2}, it means the input image is segmented out into tiny superpixels compared to the previous level, and the space between level l and $l+2$ is large. We assign more weight to the heatmap given from scale $l+1$ to enhance different results of different scales and provide more variations in visual explanations. On the other hand, if the number of segments at scale $l+1$ and $l+3$ are too close, i.e., $s'_{l+1} \approx s'_{l+3}$, these segmentations are more likely to provide similar results on the explanation heatmaps. Therefore, the automatic weighting approach based on the trapezoidal rule will assign a lower weight to the explanation heatmap taken from level $l+2$. The results of this parameter-free automatic weighting approach are provided in Chapter 5.

First Automated Weighting Approach

In this work, we assume the partitions/number of segments are not equally space. Hence, we expanded the equation 4.9. With the idea of chained trapezoidal rule for approximating definite integral and the irregularly-spaced partition, we define the first automated weighting approach as

$$\sum_{l=1}^{l+1} \frac{(E_{l-1} + E_l)}{2} \times (s'_l - s'_{l-1}) = E_0 \frac{d_0}{2} + E_1(\frac{d_0}{2} + \frac{d_1}{2}) + \cdots + E_L(\frac{d_{l-1}}{2}), \tag{4.11}$$

Where $E_1, \cdots, E_L$ is a set of explanation heatmaps, $s'_1, \cdots, s'_L$ is the final number of segments/superpixels and $d_0, \cdots, d_{l-1}$ is a set of intervals between number of segments. Therefore, each heatmap will be multiplied by the corresponding weight. These weights are calculated based on the distances/intervals of the number of segments in the proposed multi-scale scheme.

Second Automated Weighting Approach

In the second automated weighting approach, we consider the inverse of the final number of segments to calculate the distances/weights, $\frac{1}{s'_0}, \cdots, \frac{1}{s'_n}$.We define the second weighting approach as

$$\sum_{l=1}^{l+1} \frac{(E_{l-1} + E_l)}{2} \times (\frac{1}{s'_i} - \frac{1}{s'_{i-1}}) = E_0 \frac{d'_0}{2} + E_1(\frac{d'_0}{2} + \frac{d'_1}{2}) + \cdots + E_L(\frac{d'_{l-1}}{2}). \tag{4.12}$$

Chapter 5

Experiments and Results

In this Chapter, we report and discuss the experimental results of the proposed multi-scale scheme of LIME. First, we will cover the experiment design, multi-scale SLIC segmentation, which consists of 17 levels, the set of parameters, and datasets, besides a summary of the classification models, ResNet50 and VGG19, that are trained and proposed previously for the specific datasets used in this thesis. Furthermore, we present the explanation results of the proposed multi-scale scheme, qualitatively and quantitatively. Finally, we discuss and compare the overall performance of the proposed multi-scale scheme with the previously proposed square-grids segmentation [95].

5.1 Experiment Design

In this thesis, we investigated the result of the proposed multi-scale scheme of LIME with SLIC segmentation algorithm on a flower classification dataset from TFDS and a biological dataset. The multi-scale segmentation scheme is shown in Table 5.1. Basically, SLIC has two parameters which affects the segmentation results, *compactness*, and *desired number of segments*. Compactness, c, balances color proximity and space proximity. Higher values give more weight to space proximity, making superpixel shapes more square/cubic. This parameter depends strongly on image contrast and the shapes of objects in the image. It is recommended exploring possible values on a log scale, e.g., 0.01, 0.1, 1, 10, 100. From here to the end of this thesis, we set $c = 20$ for all experiments in a multi-scale scheme. We found that SLIC segmentation with this parameter provides meaningful superpixels while keeping the boundaries in an image. To compare the results of the proposed multi-scale scheme of LIME with the equally sized square-grids in the literature, we set $c = 50$ for the square-grids segmentations of LIME. The only parameter that is changed during the process is the *desired number of segments*. We define $\sqrt{s}$ as the square root of the *desired number of segments*, starting from 4 to 20. Then s will be passed to the SLIC segmentation algorithm, returning the final number of segments/superpixels, s'. Fig 5.1 shows the SLIC segmentation algorithm on a sample image from TensorFlow flower dataset with desired number of segments specified in Table 5.1. The sample image and the desired number of segments are passed to the SLIC segmentation algorithm. Finally, the segmented image with a final number of segments is presented in Fig 5.1. The desired number of segments and the number of segments after SLIC segmentation algorithm are displayed on top of each image.

5.2 Flower Classification from TensorFlow Dataset

TensorFlow Datasets (TFDS) provide a collection of ready-to-use datasets with TensorFlow, Jax, and other Machine Learning frameworks. To explore the results of the proposed multi-scale explanation framework with LIME in a classification problem, we used the flower dataset from TFDS. This dataset was split into a training set, and a validation set shown in Table 5.2. The dataset consists of color images of size 224×224 with 5 possible class labels, see Fig 5.2, *'daisy', 'tulip', 'sunflower', 'dandelion', 'rose'*.

Table 5.1: Multi-scale segmentation scheme with desired number of segments.

Multi-scale Scheme			
Level	Square root of s	Desired number of segments (s)	Number of segments s'
1	4	16	12
2	5	25	21
3	6	36	34
4	7	49	45
5	8	64	60
6	9	81	77
7	10	100	93
8	11	121	111
9	12	144	135
10	13	169	160
11	14	196	184
12	15	225	210
13	16	256	242
14	17	289	279
15	18	324	347
16	19	361	347
17	20	400	383

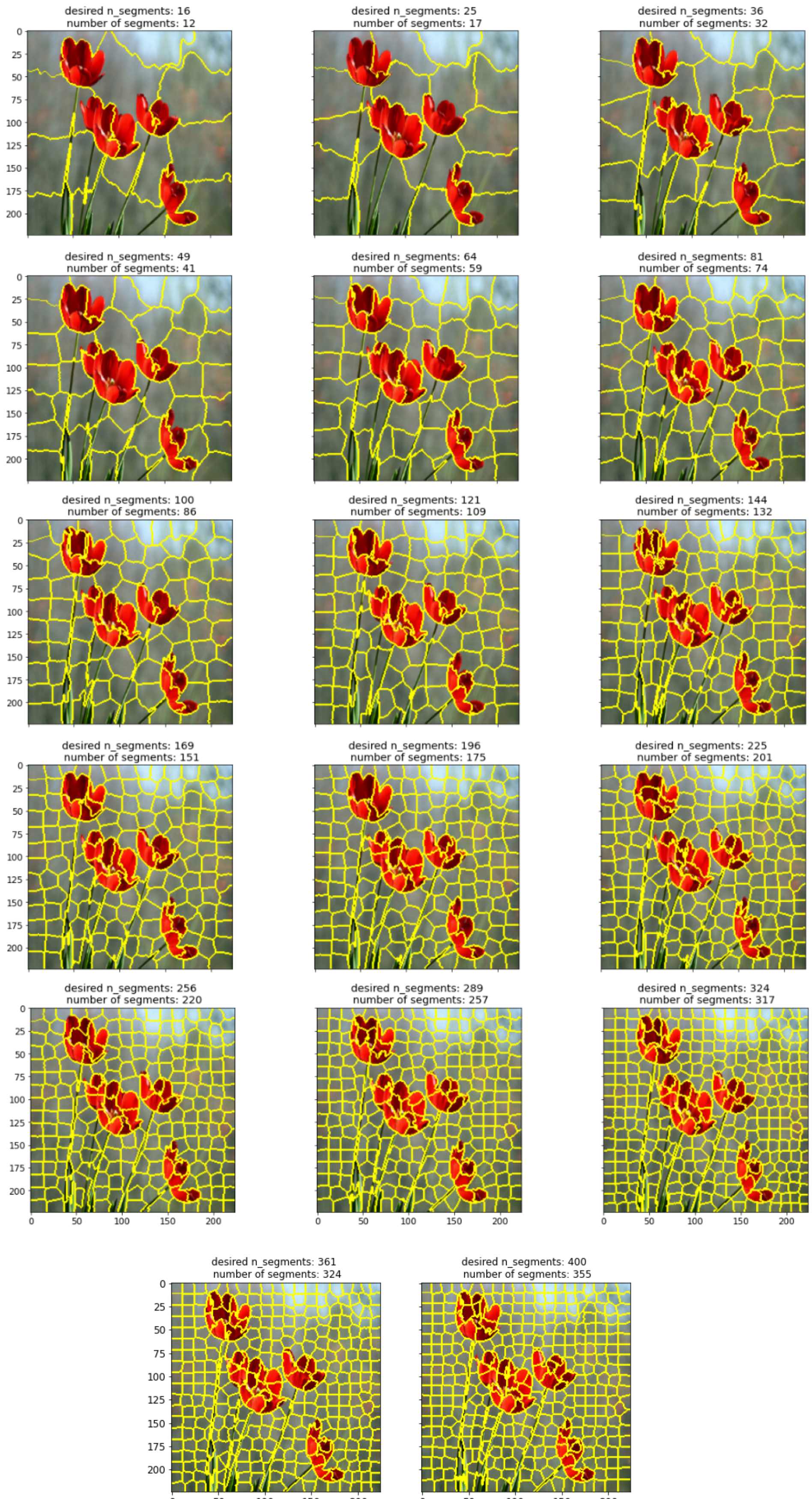


Figure 5.1: Multi-scale segmentation scheme with 17 levels.

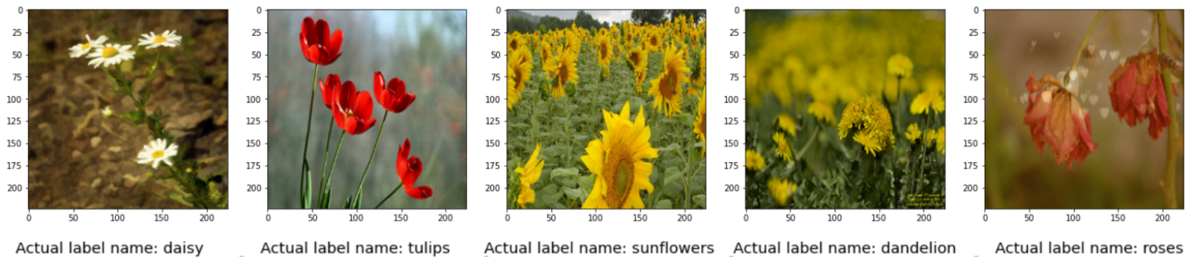


Figure 5.2: Flower classification of TFDS. Sample images of 5 classes with actual labels.

TensorFlow Flower Dataset					
	daisy	tulip	sunflower	dandelion	rose
Total	769	984	734	1052	784
Training	615	787	587	841	627
Validation	154	197	147	211	157

Table 5.2: Flower Dataset: number of samples per class in training and validation set.

5.2.1 Classification model: Fine-tuning ResNet50

Several pre-trained CNN has been evaluated on this dataset, like VGG16, VGG19, and ResNet50. Accorsing to Kaggle Flower Classification Keras, ResNet50 provided the best performance among state-of-the-art pre-trained classification networks. Hence, we used pre-trained ResNet50 model and fine-tuned that on this dataset.

The ResNet50 model consists of 5 stages each with a convolution and identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet50 has over 25 million trainable parameters. The architecture of this network is shown in Fig 5.3.

To fine-tuning ResNet50 on this dataset, we set the parameters of the model to be non-trainable and then added six more layers, including Batch-Normalization layer and Dense layer, to the network as below:

- Batch-Normalization layer

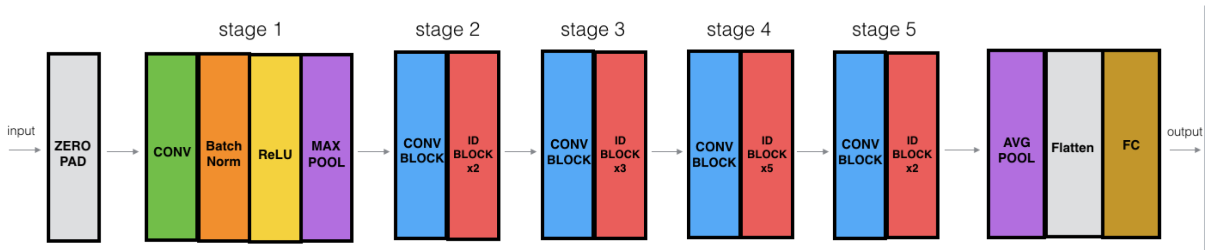


Figure 5.3: Architecture of ResNet50 Convolutional Neural Network

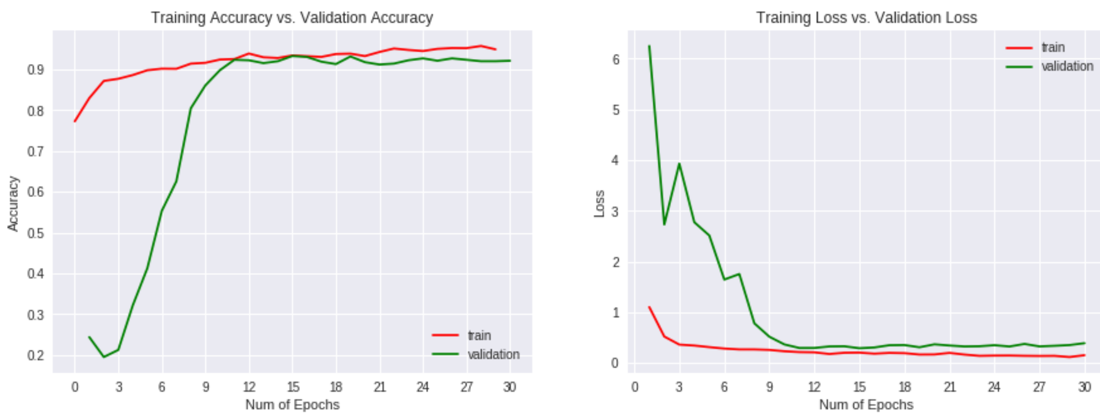


Figure 5.4: Accuracy and Loss during epochs: Training and validation set

- Dense layer (2048, ReLU activation)
- Batch-Normalization layer
- Dense layer (1024, ReLU activation)
- Batch-Normalization layer
- Dense layer (num-classes=5, softmax activation)

Then, we trained the model on training data and evaluated how well the model performed on training and validation sets.

The accuracy of fine-tuned ResNet50 on this dataset is 0.92 on training set and 0.90 on validation set. Fig 5.4 shows the accuracy and loss during 30 epochs.

In the following Sections, we evaluated the results of the proposed multi-scale version of LIME on three different samples taken from the Flower dataset, qualitatively and

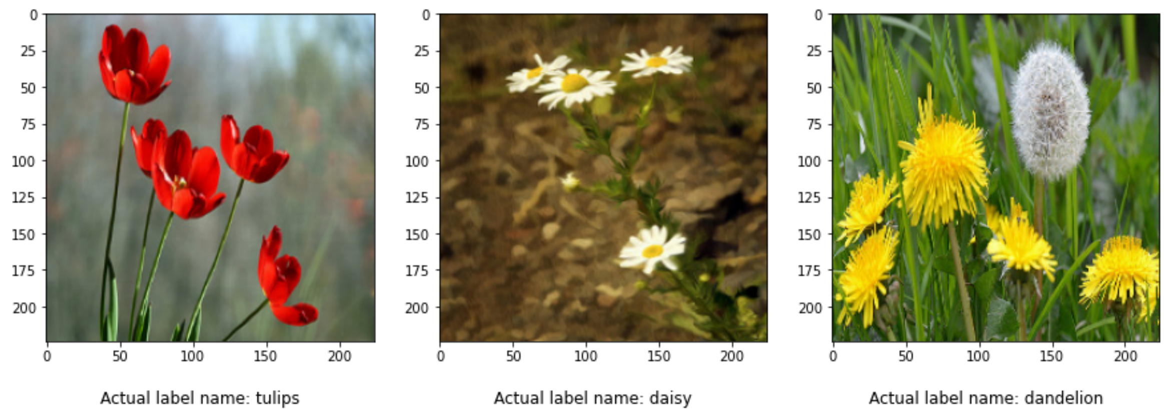


Figure 5.5: Sample images with the actual labels from Flower dataset

quantitatively. First, in Section 5.3, we investigated how LIME detects the most important superpixels at different levels qualitatively. This section evaluated the localization and class discriminative aspects of the proposed multi-scale scheme. Then, Section 5.4 checked the faithfulness of the proposed multi-scale scheme using Area Under Curve (AUC) with the idea of deletion the highest superpixels/regions. We also calculated the explanation accuracy of the multi-scale scheme at coarse, finer, and finest scales.

5.3 Qualitative Results

In order to examine how the proposed multi-scale scheme of LIME works, we chose three different samples from the Flower dataset shown in Fig 5.5. The reason we chose these sample images is that there are multiple objects of the same class in each image and these objects are represented on different scales.

The first sample image is '*tulips*' from Fig 5.5. When we passed this image to the ResNet50, the predicted labels and probabilities corresponding to each class are shown in Table 5.3. As the predicted probabilities show, the first predicted class is '*tulips*' with a probability of 0.98, and the second predicted class is '*roses*' with the probability of 0.18. We passed this sample image into the multi-scale scheme with 17 levels, and the

Sample image 1 -Actual class name: tulips	Class name	Predicted probability
First predicted class	tulips	0.98
Second predicted class	roses	0.18
Third predicted class	daisy	0.00
Forth predicted class	sunflowers	0.00
Fifth predicted class	dandelion	0.00

Table 5.3: Predicted probabilities per class. Sample image: tulips

final heatmaps corresponding to each level are presented in Fig 5.6. In other words, we implemented LIME at each scale and fitted individual linear models. The heatmaps obtained from each level represent the top superpixels toward the target class. As Fig 5.6 shows, we explored the regions of interest from coarse to fine scales.

The execution time of the very first levels, coarse scales, is less than a minute, and the process takes more time to produce the samples at finer scales, about 3-4 minutes. Totally the computation time of the proposed multi-scale scheme with 17 levels is around an hour for each sample image. Clearly, when the number of samples is constant at each level, the computation time is highly dependent on the the number of levels.

Finally, we presented the weighted combination of as visual explanations the first and second classes visually, to evaluate the localization and class discriminative aspects of the proposed multi-scale scheme.

The second sample image is '*daisy*' from Fig 5.5. First, we passed this image to the black-box model, ResNet50, and the predicted probabilities per class are shown in Table 5.4. The first predicted class is '*daisy*' with the prediction probability of 0.868. Also, the black-box model predicted '*dandelion*' as the second class that we were interested in investigating and explaining the results visually. In other words, we wanted to see which part of the image the black-box model was looking at to predict the first and the second classes.

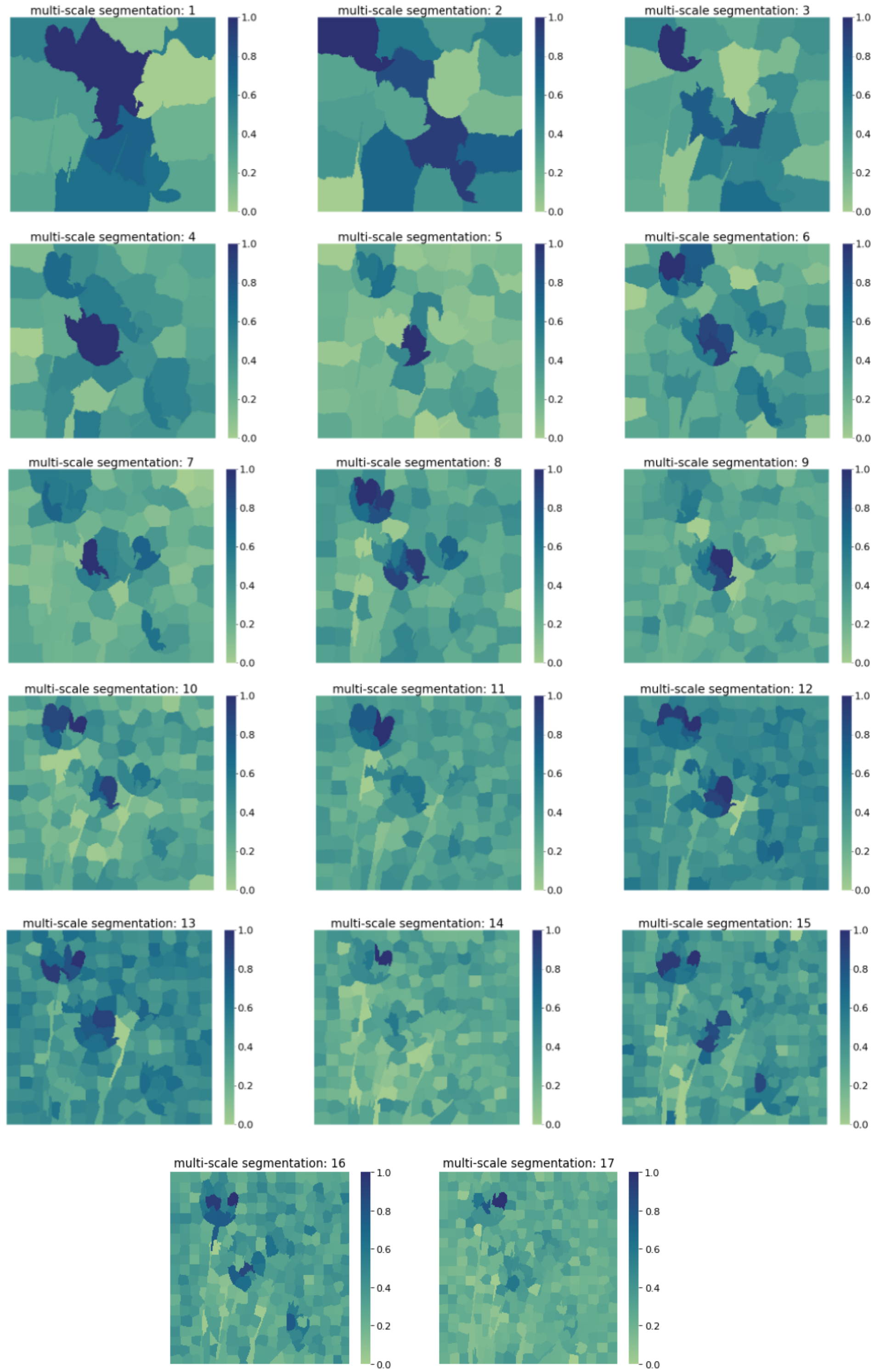


Figure 5.6: Heatmaps of multi-scale segmentation scheme with 17 levels. Sample image: tulips

Sample image 2 -Actual class name: daisy	Class name	Predicted probability
First predicted class	daisy	0.86
Second predicted class	dandelion	0.07
Third predicted class	sunflowers	0.05
Forth predicted class	roses	0.00
Fifth predicted class	tulips	0.00

Table 5.4: Predicted probabilities per class. Sample image: daisy

Sample image 3 -Actual class name: dandelion	Class name	Predicted probability
First predicted class	dandelion	0.83
Second predicted class	daisy	0.16
Third predicted class	sunflowers	0.00
Forth predicted class	tulips	0.00
Fifth predicted class	roses	0.00

Table 5.5: Predicted probabilities per class. Sample image: dandelion

The third sample image is '*dandelion*' from 5.5. When we passed this image to the fine-tuned ResNet50, the model predicted the correct label, '*dandelion*', as the first prediction class with probability 0.83. The predictions are shown in Table 5.5. The second predicted class for this image is '*daisy*', and we also explored the class discriminative aspect of the proposed multi-scale scheme through visual explanations for this sample image as well. For better comparison and consistency in the results, all of the heatmaps provided in this section are the weighted combinations with Gaussian function.

5.3.1 Visual Explanation Results with Gaussian Function

Evaluating Localization

This section evaluates two main aspects of the proposed multi-scale scheme through the weighted heatmaps from coarse, finer, and finest scales. As mentioned in the literature, a good visual explanation method should be able to localize the important pixels/superpixels instead of highlighting a whole region of a target object towards the interest label/class.

To compare the results of the proposed multi-scale scheme with the original LIME, we represented the results of each scale with the original LIME on heatmaps, besides the superimposed heatmaps on top of the sample image. Also, we compared the results obtained from multi-scale with the square-grids segmentation approach proposed in [95]. All the results presented in this section are obtained from the first predicted class for each sample image. For the first sample "*tulips*", the localization results presented in Figs 5.7, and 5.10, respectively. As Fig 5.7 shows, the interest regions are more localized in the multi-scale scheme compared to the original LIME, even at the finest scale. In Fig 5.10, we aimed to evaluate the results obtained from the weighted combination at each scale with the corresponding results of square-grids segmentation. As it is shown, the proposed scheme can significantly represent the important superpixels while keeping the object boundaries.

The same results are provided for second and third sample images as well, see Figs 5.8, and 5.11 for sample "*daisy*", Figs 5.9, and 5.12 for sample "*dandelion*". The results in Fig 5.8, is a good example of visual explanations where the image contains multi objects and different parts of each object have been detected at each scale perfectly. Looking at the original results of LIME in Fig 5.8, shows that LIME might not be able to detect all top superpixels/regions with highlighted boundaries. Although the heatmap of the original LIME highlighted some parts of the interest object, the small objects are

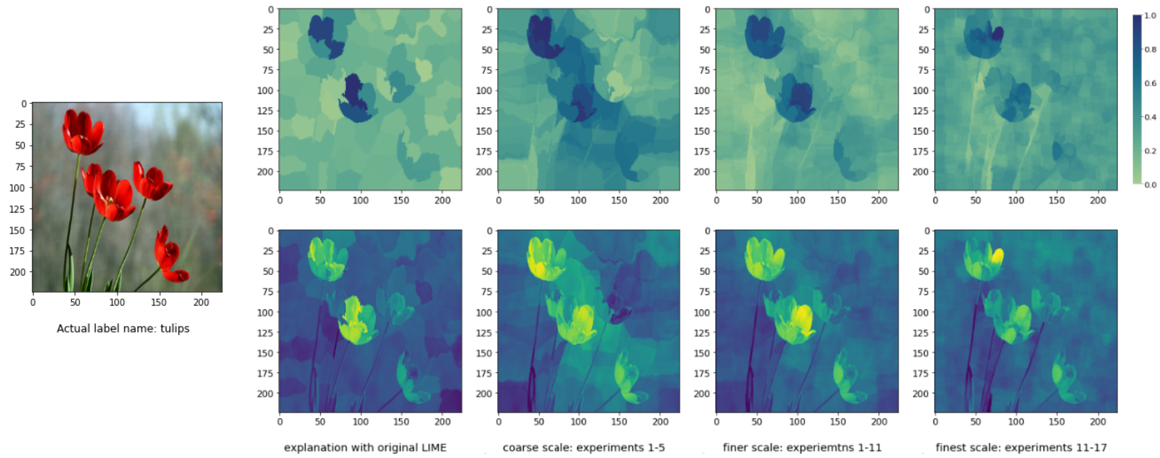


Figure 5.7: First predicted class“tulips”: comparison of original LIME vs multi-scale scheme from coarse to finest scale. The heatmaps are superimposed on top of the original image shown in the second row for better visualization.

somehow ignored. On the other hand, the explanation results of the proposed multi-scale scheme for this sample show how the objects detected at coarse and finer scales, even the smaller objects. Also, when the multi-scale scheme looks at tiny regions of the same objects from the very last level, tiny superpixels appeared in the results that the model was looking at during the decision-making process. Fig 5.11 shows the explanation results of the same sample image “*daisy*” with the square-grids segmentation, corresponding to each scale. These heatmaps show a lot of highlighted regions in the background, and the boundaries were not detected correctly. The results are somehow the same for third sample “*dandelion*” in Figs 5.9, and 5.12. The results obtained from multi-scale scheme are significantly better than the original LIME and square-grids segmentation for this sample image.

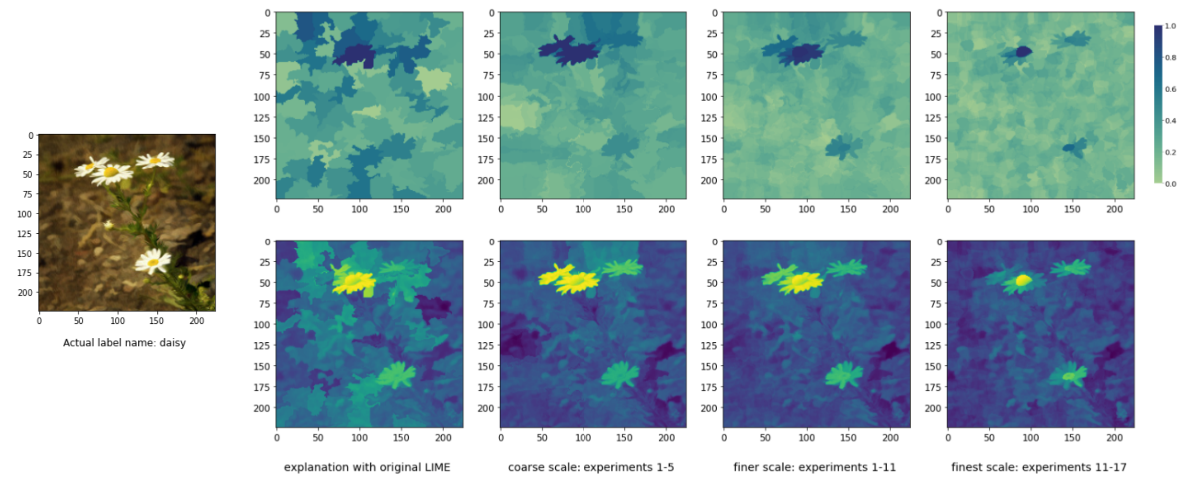


Figure 5.8: First predicted class "daisy": comparison of original LIME vs multi-scale scheme from coarse to finest scale.

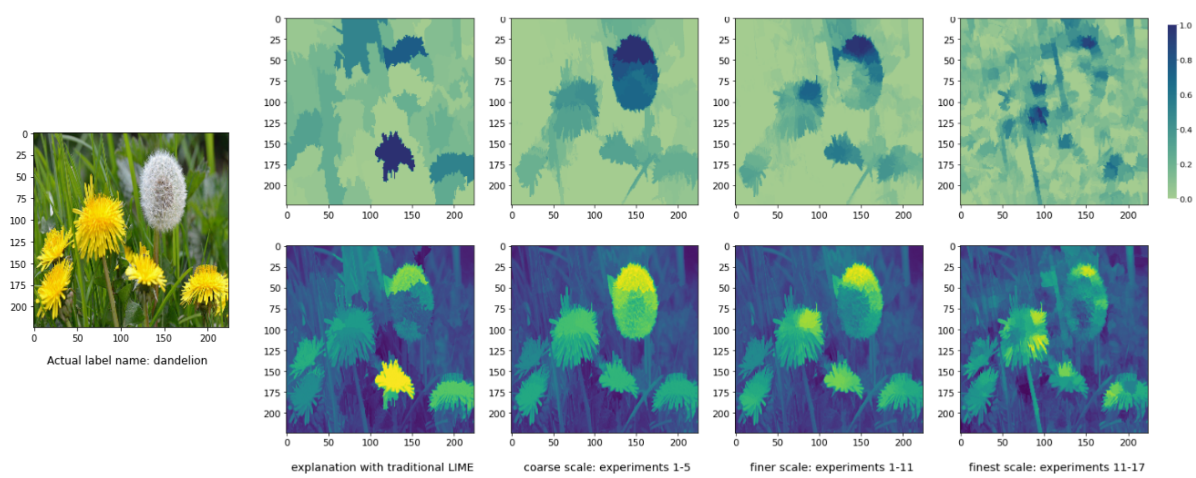


Figure 5.9: First predicted class "dandelion": comparison of original LIME vs multi-scale scheme from coarse to finest scale.

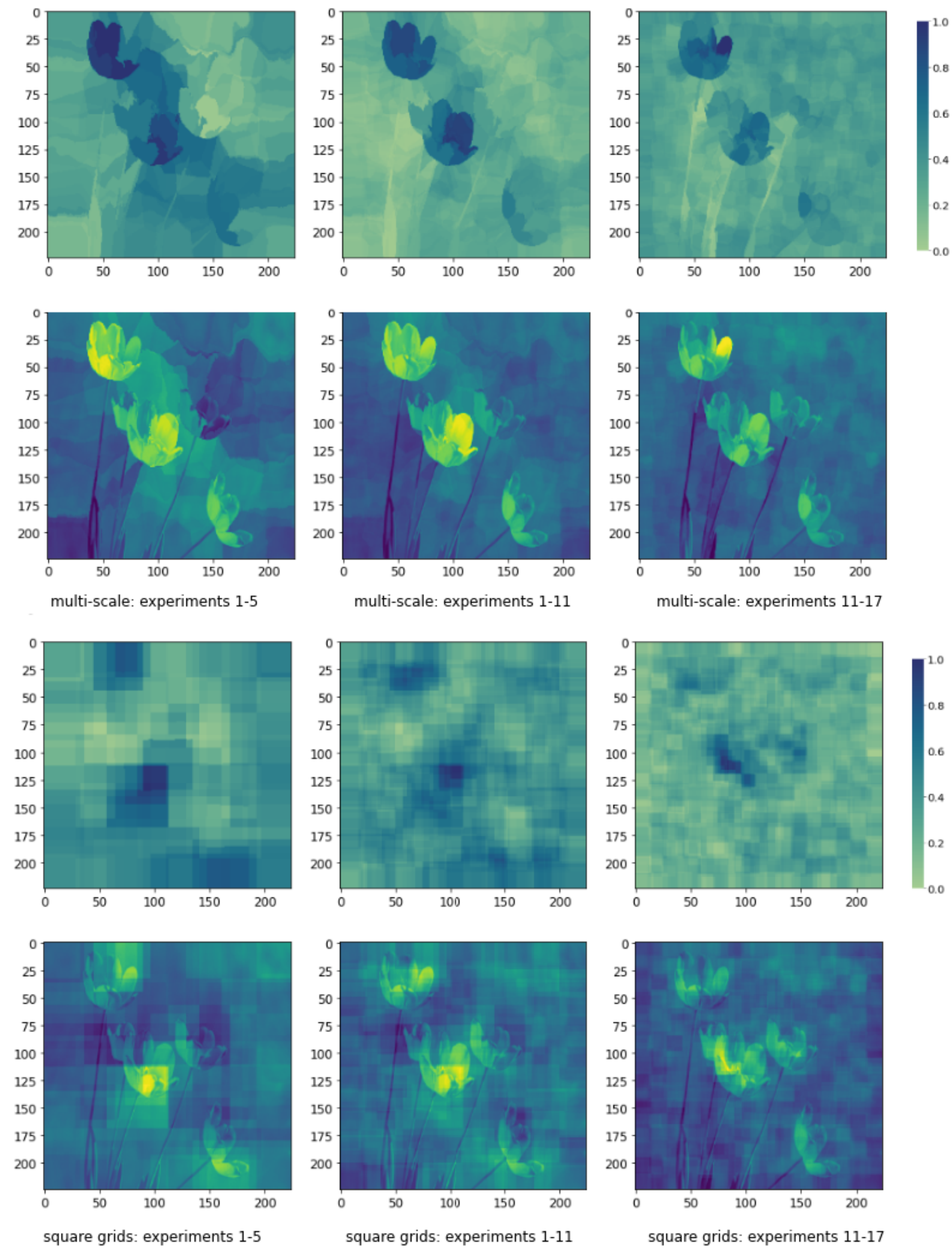


Figure 5.10: First predicted class "tulips": Localization of multi-scale scheme from coarse to finest scale vs square-grids segmentations

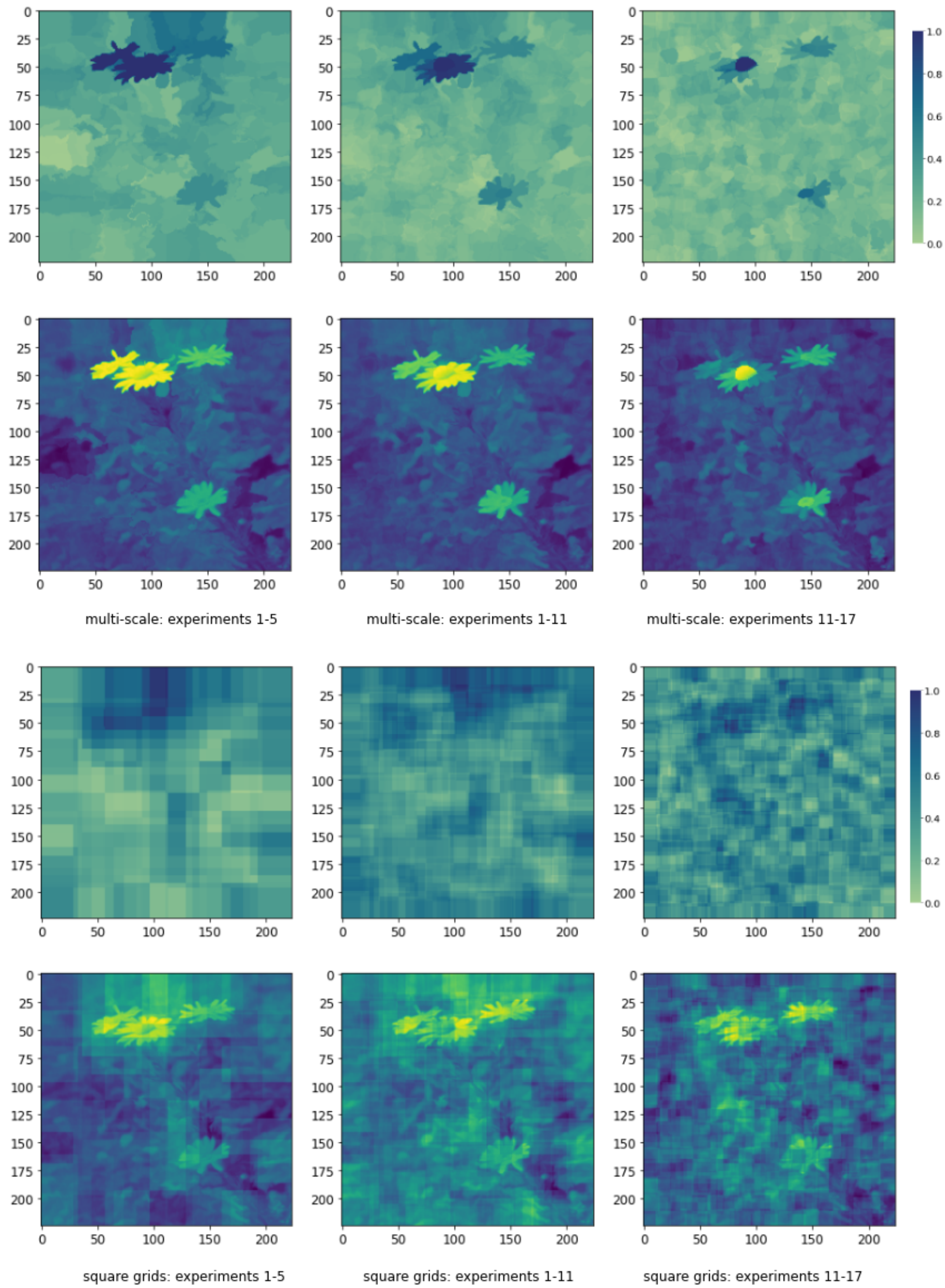


Figure 5.11: First predicted class "daisy": Localization of multi-scale scheme from coarse to finest scale vs square-grids segmentations

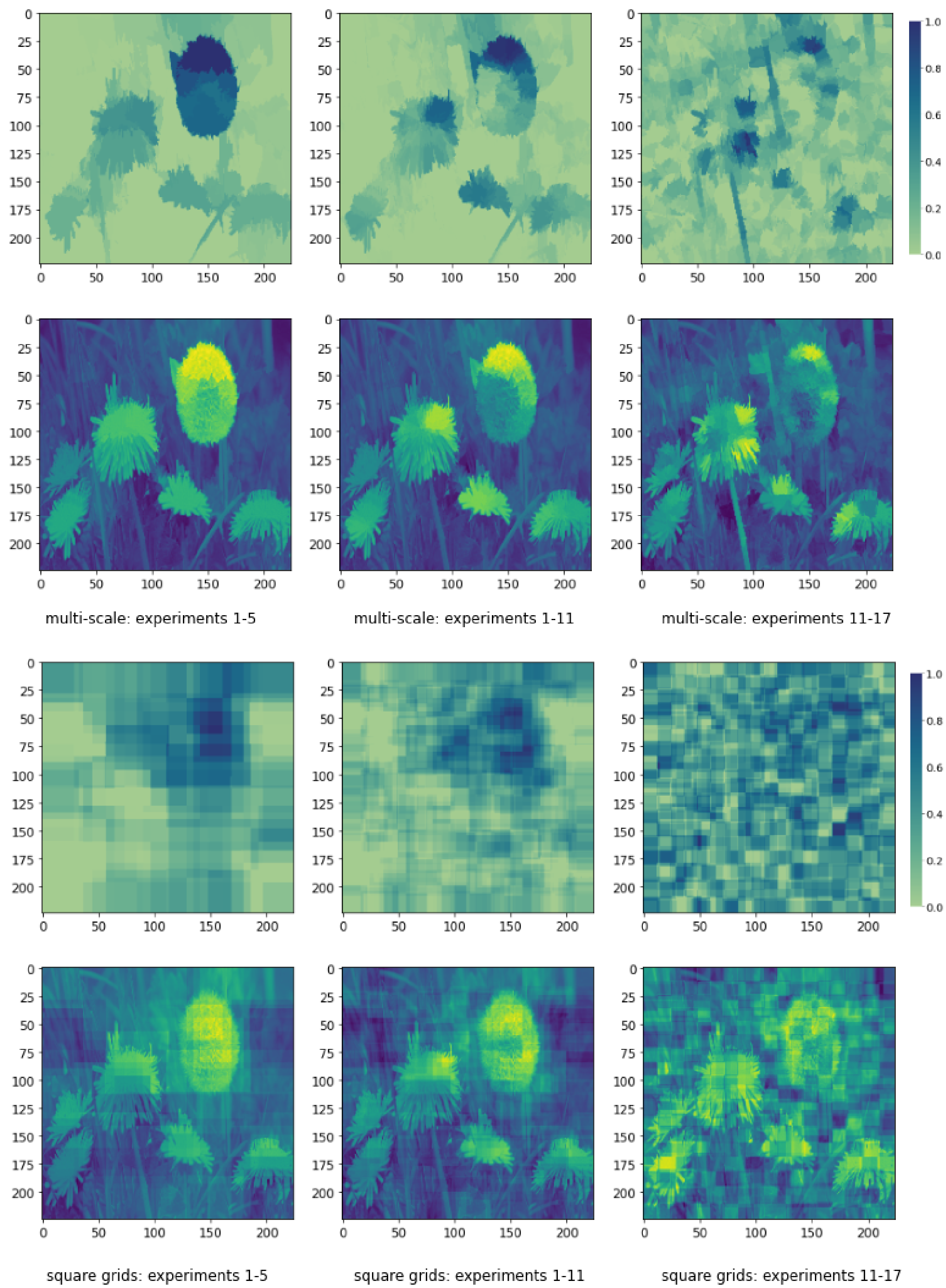


Figure 5.12: First predicted class "dandelion": Localization of multi-scale scheme from coarse to finest scale vs square-grids segmentations

Evaluating Class Discriminative

Besides localization, a good visual explanation should be class-discriminative, which means in a multi-classification problem, the heatmaps of different classes should not be the same. There are several approaches to examining the similarity of two heatmaps/images. This section provided the results of the first and second predicted class for each sample image corresponding to each scale. The results prove the multi-scale scheme refers to different superpixels/regions w.r.t. the target class. In Figs 5.13, 5.14, and 5.15 we investigated how the heatmaps are changed corresponding to the first and second predicted class for each sample. In Fig 5.13, the second predicted class for sample image "*tulips*" is "*roses*" with accuracy of 0.18, see Table 5.3, and the heatmaps refer to the irrelevant regions highlighted in the background and the tulips are ignored by the black-box. Although the heatmaps of the first and second predicted class obtained with original LIME are not the same for this sample, it is not clear where the black-box model focused on predicting "*roses*". Through the multi-scale scheme, some regions highlighted at coarse scale are a little bit look like roses, and the model might be fooled in this case. Another interesting result is provided in Fig 5.14 for sample image "*daisy*". In this case, the second predicted class is "*dandelion*" , see Table 5.4, and the results obtained from the original LIME is look-alike, with some minor difference in some regions. But the most important superpixels with the highest coefficient are the same in the first and second predicted classes. In other words, the original LIME is not class discriminative in this sample since it does not explain which superpixels the model has paid attention to predict the second class. We applied the proposed multi-scale scheme of LIME to this sample, and the result of the weighted heatmap at the finest scale represents the tiny superpixel that the model was looking at in the second prediction class. The same experiments have been done to the third sample image "*dandelion*", and the second predicted class is "*daisy*", see Table 5.5. The same as the previous instance image, the original LIME is not specifically class discriminative, Fig 5.15. The regions that the original LIME

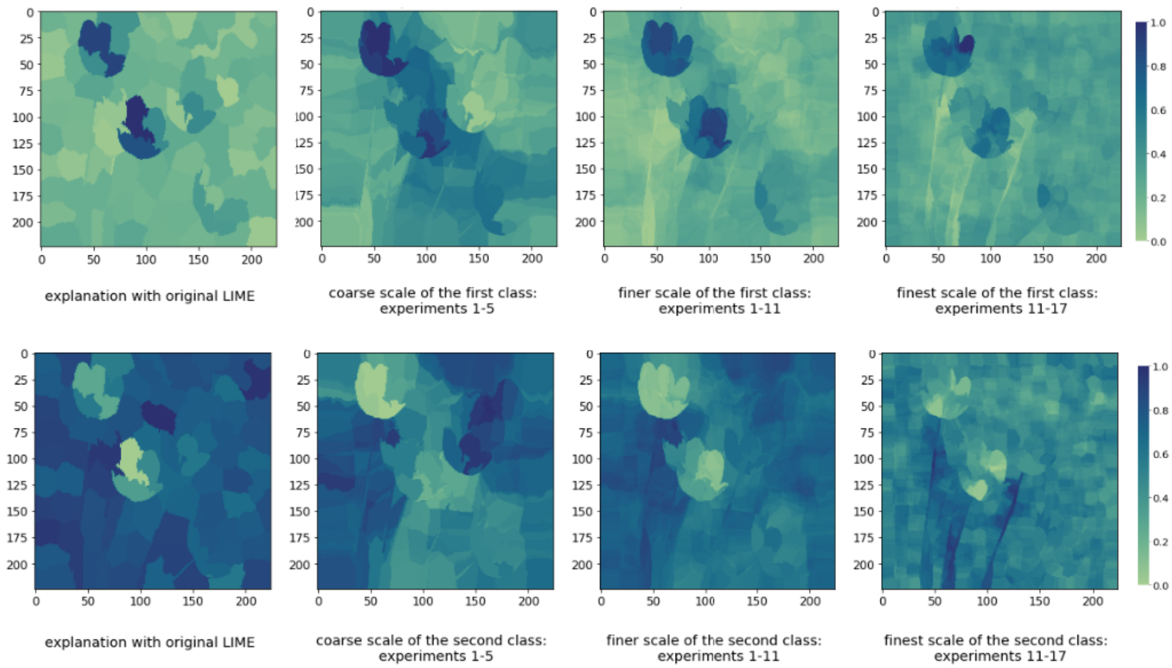


Figure 5.13: Evaluation of class discriminative aspect: multi-scale scheme. Sample tulips

represented in the first and second classes are somehow pointing to the same object in the image. While looking at the explanations of the multi-scale scheme, the top super-pixels/regions that appeared in the heatmaps of coarse and finer scales w.r.t. the second class are different than the first predicted class.

5.3.2 Visual Explanation Results with Automated Approaches

Evaluating Localization

Based on the number of segments specified in Table 5.1 for 17 levels in the proposed multi-scale scheme, the weights are calculated for the first and second automated weighting approaches that are shown in Fig 5.16. For example, according to the first automated weighting approach presented in Section 4.7.2, experiments of finer scale segmentations will get higher weights. On the other hand, if we calculate the distance between two levels by taking the inverse of the number of segments, see Section 4.7.2, some expla-

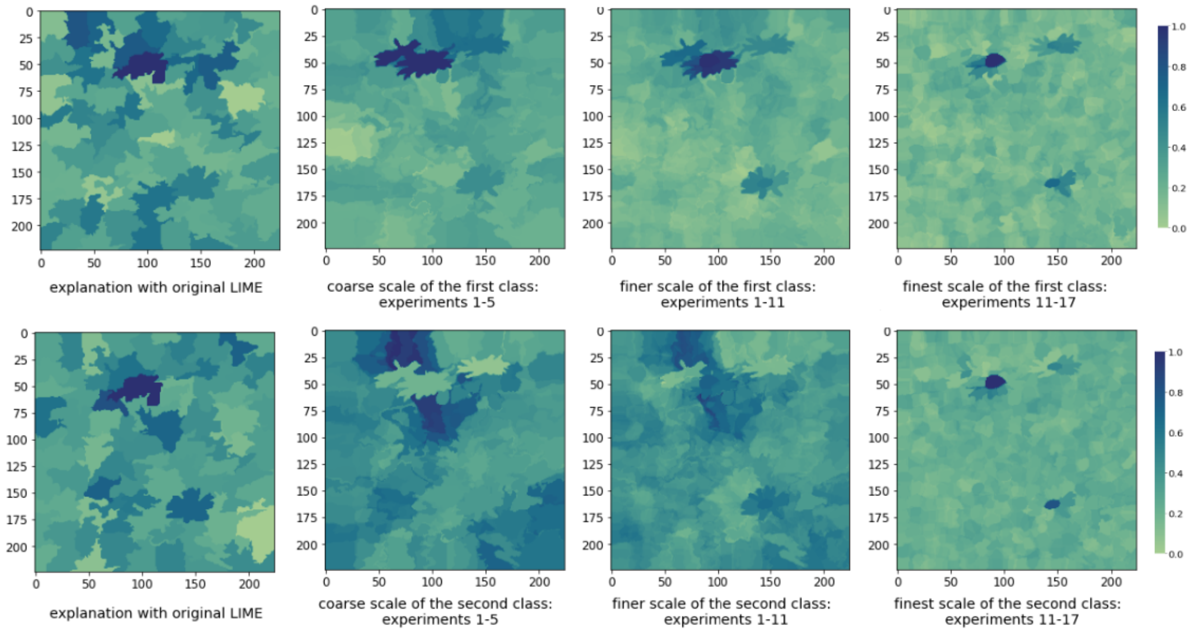


Figure 5.14: Evaluation of class discriminative aspect: multi-scale scheme. Sample daisy

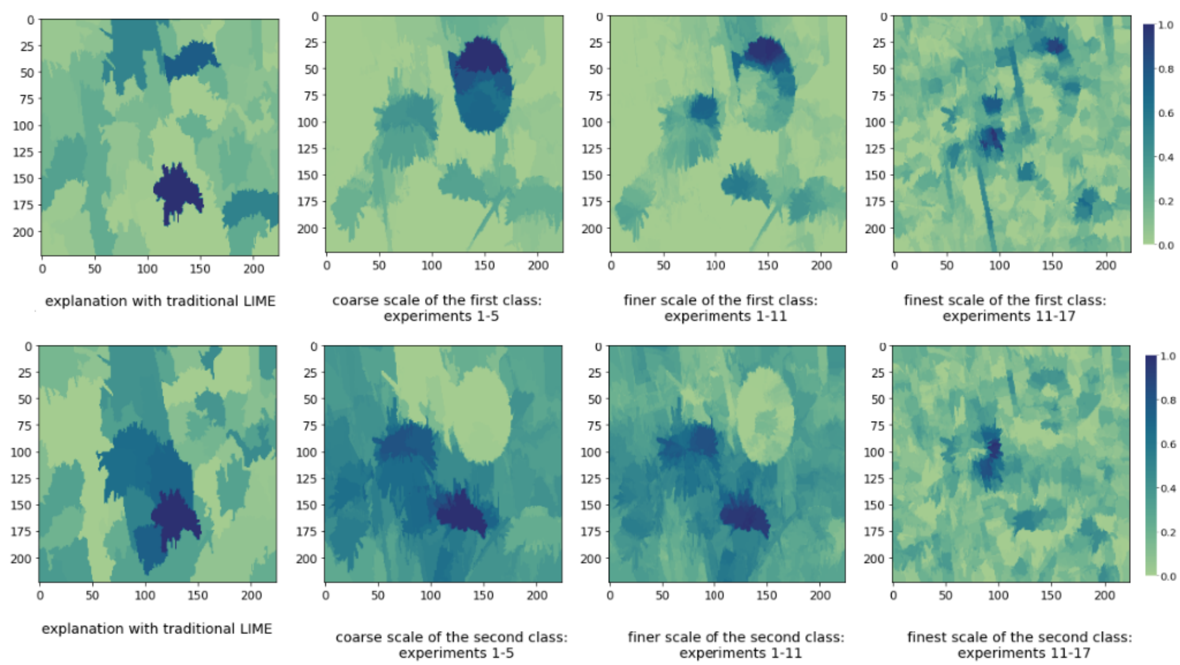


Figure 5.15: Evaluation of class discriminative aspect: multi-scale scheme. Sample dandelion

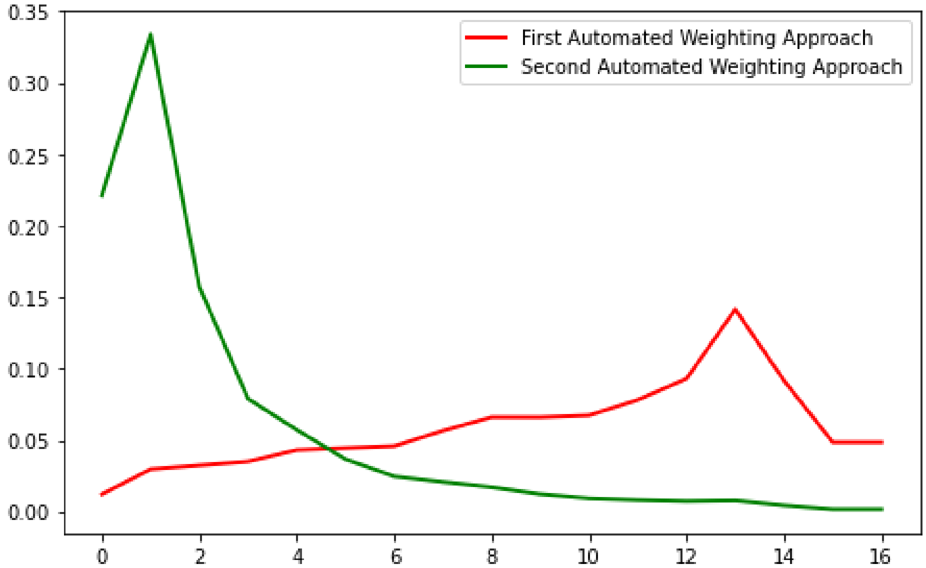


Figure 5.16: Automated weighting approaches

nations obtained from the coarse scale will get higher weights and explanations of finer scales will be equally weighted. The explanation results of these two automated weighting approaches are shown in Figs 5.17, 5.18, 5.19, for "*tulips*", "*daisy*", and "*dandelion*", respectively. The explanation results of heatmaps obtained from these automated weighting approaches are considerably consistent with the results of coarse and finest scales weighted with Gaussian function.

Evaluating Class Discriminative

In this section, we evaluated the class discriminative aspect of the multi-scale scheme with automated weighting approaches. Figs 5.20, 5.21, and 5.22 represent the results of the first and second predicted class on heatmaps obtained from automated weighting approaches for sample images "*tulips*", "*daisy*", and "*dandelion*", respectively. In each figure, the first and second rows represent the weighted heatmaps obtained from the automated weighting approaches for the first and second predicted classes, respectively. The results are visually consistent with the Gaussian function weighting approach at the

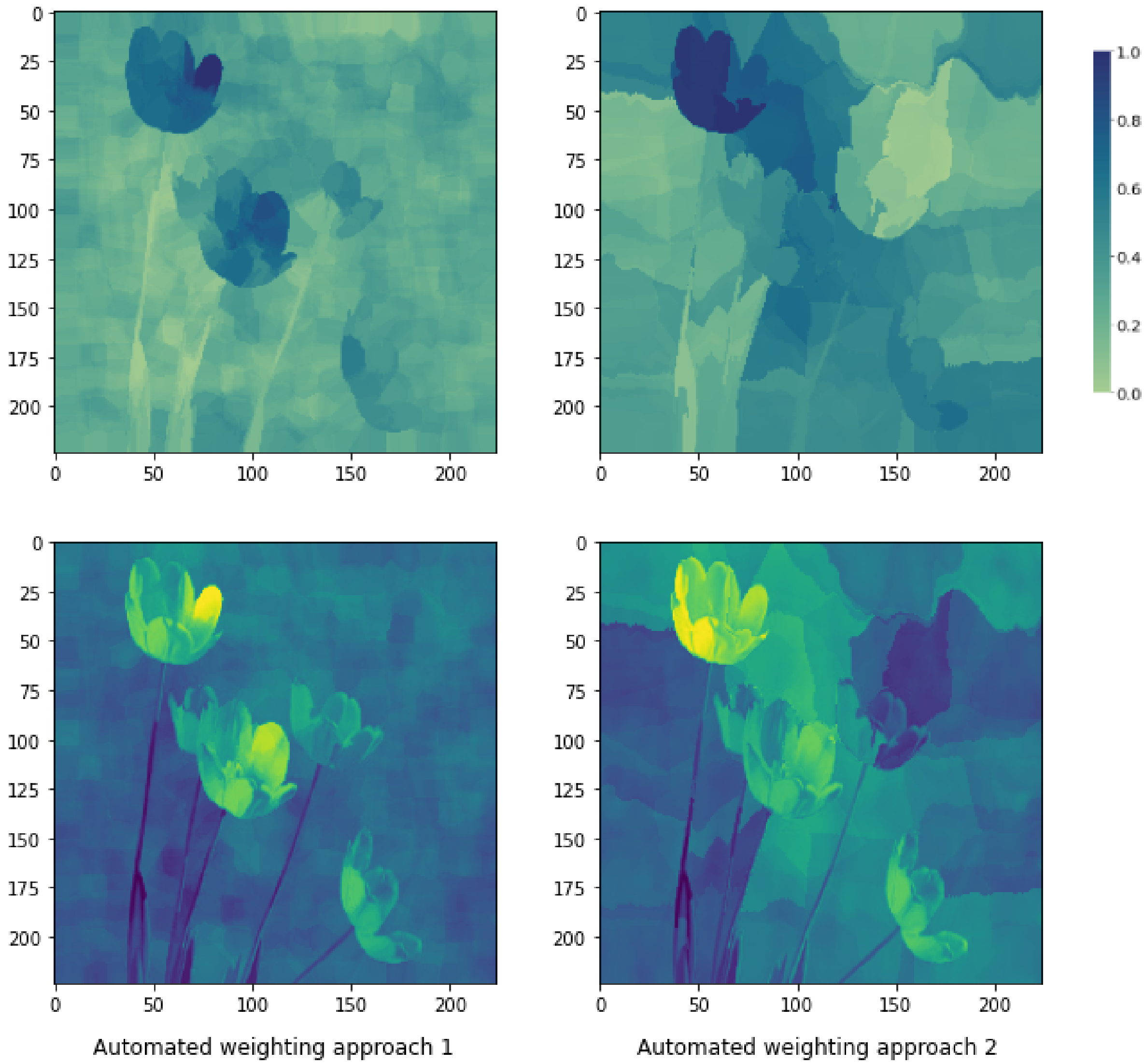


Figure 5.17: Explanation result of the automated weighting approaches for sample image tulips

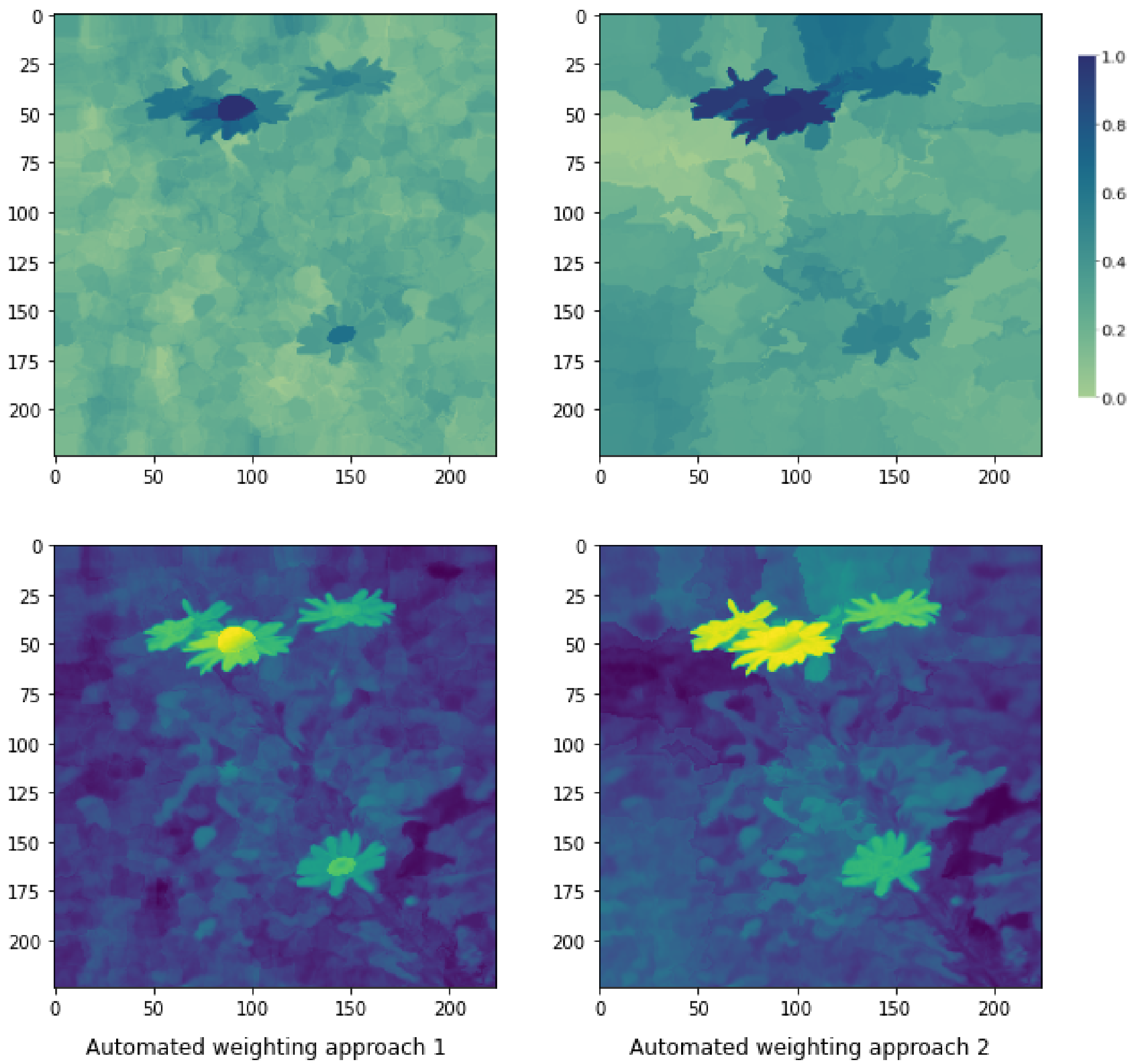


Figure 5.18: Explanation result of the automated weighting approaches for sample image daisy

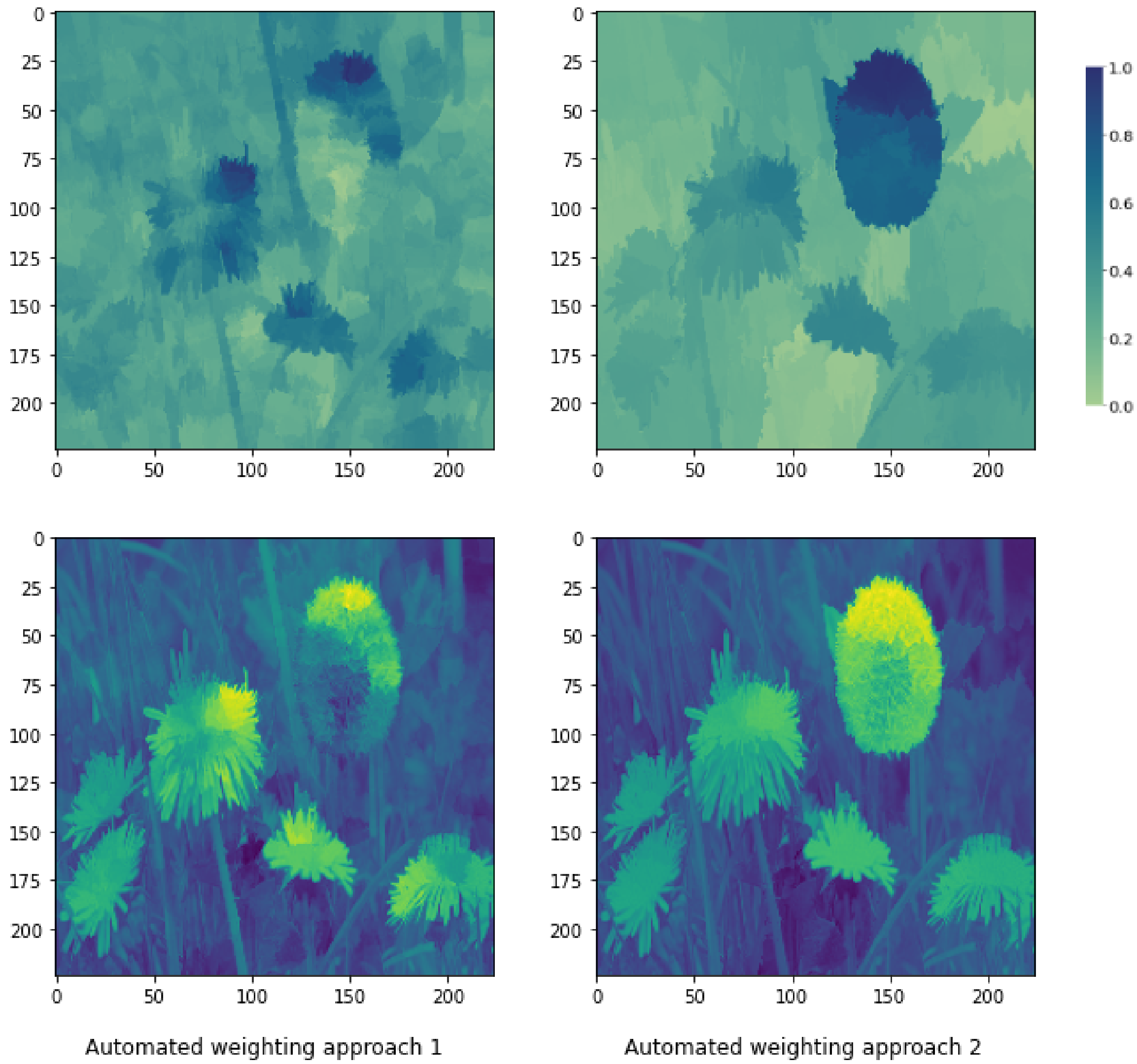


Figure 5.19: Explanation result of the automated weighting approaches for sample image dandelion

coarse and finest scale.

These are two general approaches to classifying the heatmaps. We believe that the weighting approaches are subjective and the weights might be assigned based on an expert's opinion. In other words, the measure of goodness is subjective when the scales are increasing or decreasing. In these weighting approaches, we aim to provide various information at different scales. For example, we assign weights based on Gaussian function to focus on mid-range scale experiments and produce explanations that are more localized to the objects' boundaries. Also in the proposed parameter-free weighting approaches, when it considers the difference in the number of superpixels at each level, the purpose is that experiments which provide various results get higher weights. Therefore, increasing the number of scales does not matter in this case. Another way is to assign weights to the heatmaps based on an expert's opinion in high-risk cases, like medical images.

Since the automated weighting approaches provided consistent results with varying scales, we are motivated to use these approaches on biological datasets in Section 5.5.

5.4 Quantitative Results

This section evaluates the faithfulness of the proposed multi-scale scheme through AUC and the explanation accuracy at each level.

5.4.1 Area Under Curve

Estimation of the importance of each superpixel in state-of-the-art methods has been investigated in [98]. To assess the performance of LIME with the proposed multi-scale explanation scheme, we used the automatic evaluation metric called "deletion", motivated by [36, 98]. The intuition behind the deletion metric is that the removal of the *cause* will force the base model to change its decision. Specifically, this metric measures a decrease in prediction probabilities as more and more important pixels are removed,

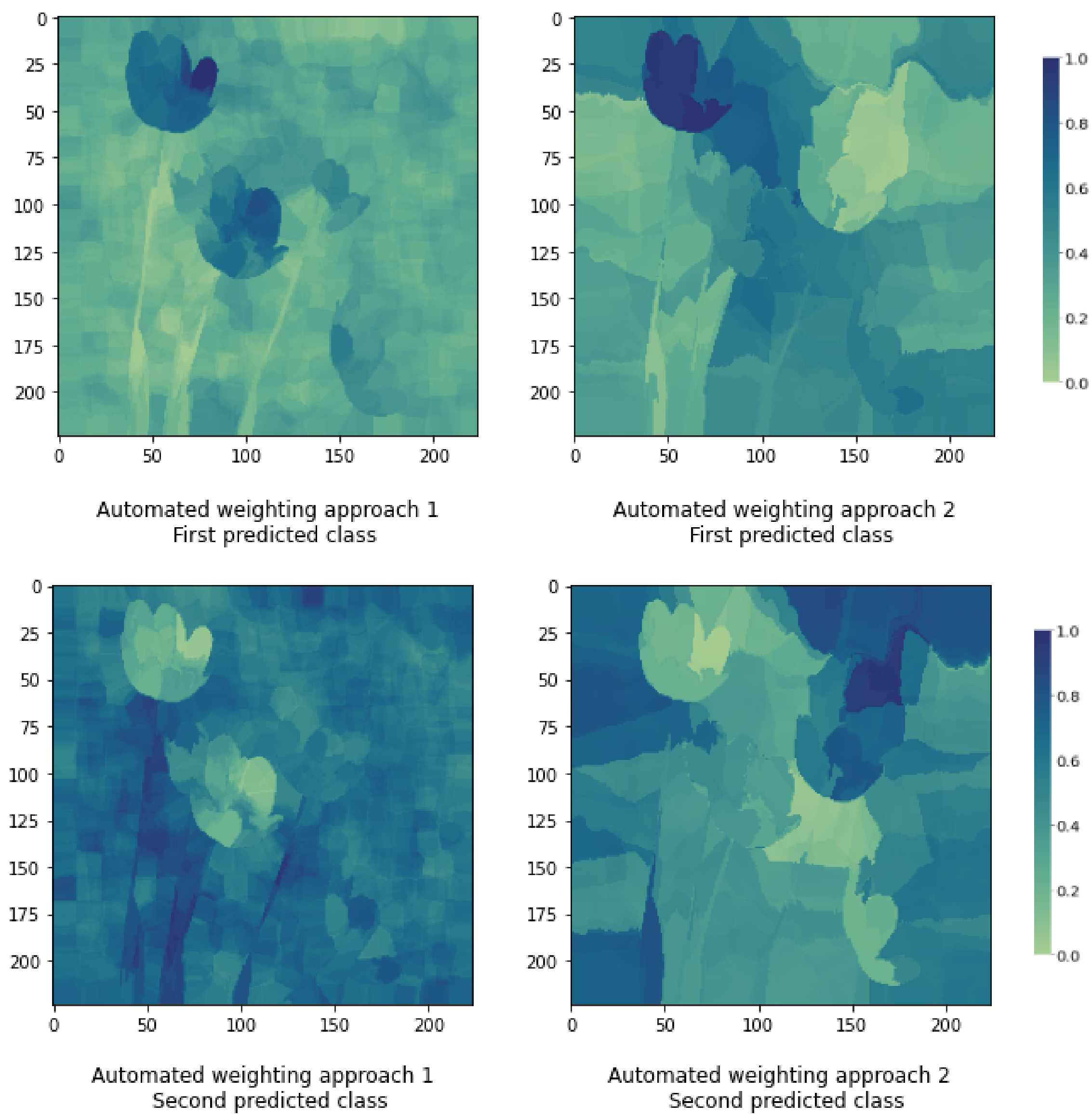


Figure 5.20: Evaluation of class discriminative aspect: automated weighting approaches for sample image tulips

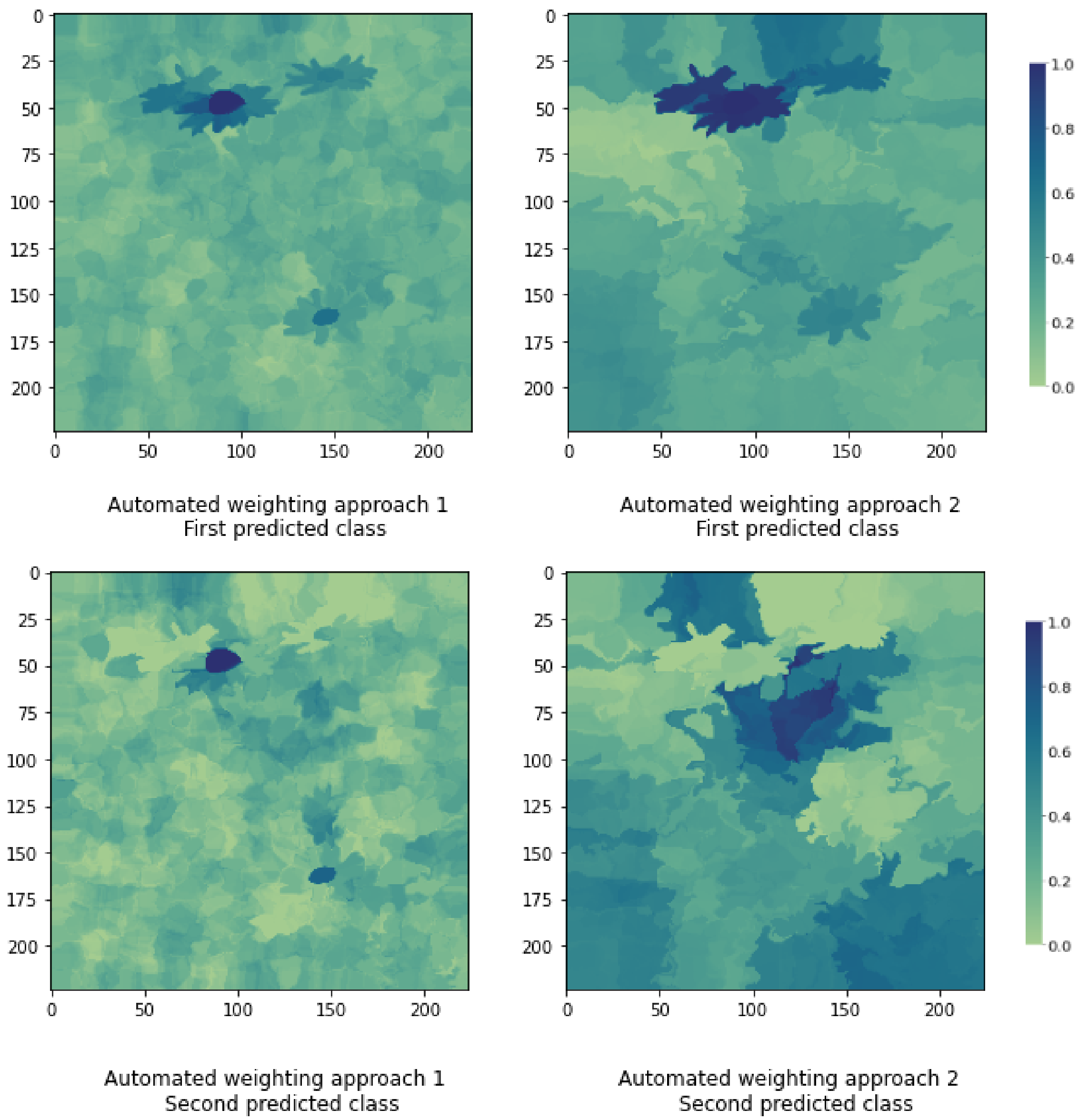


Figure 5.21: Evaluation of class discriminative aspect: automated weighting approaches for sample image daisy

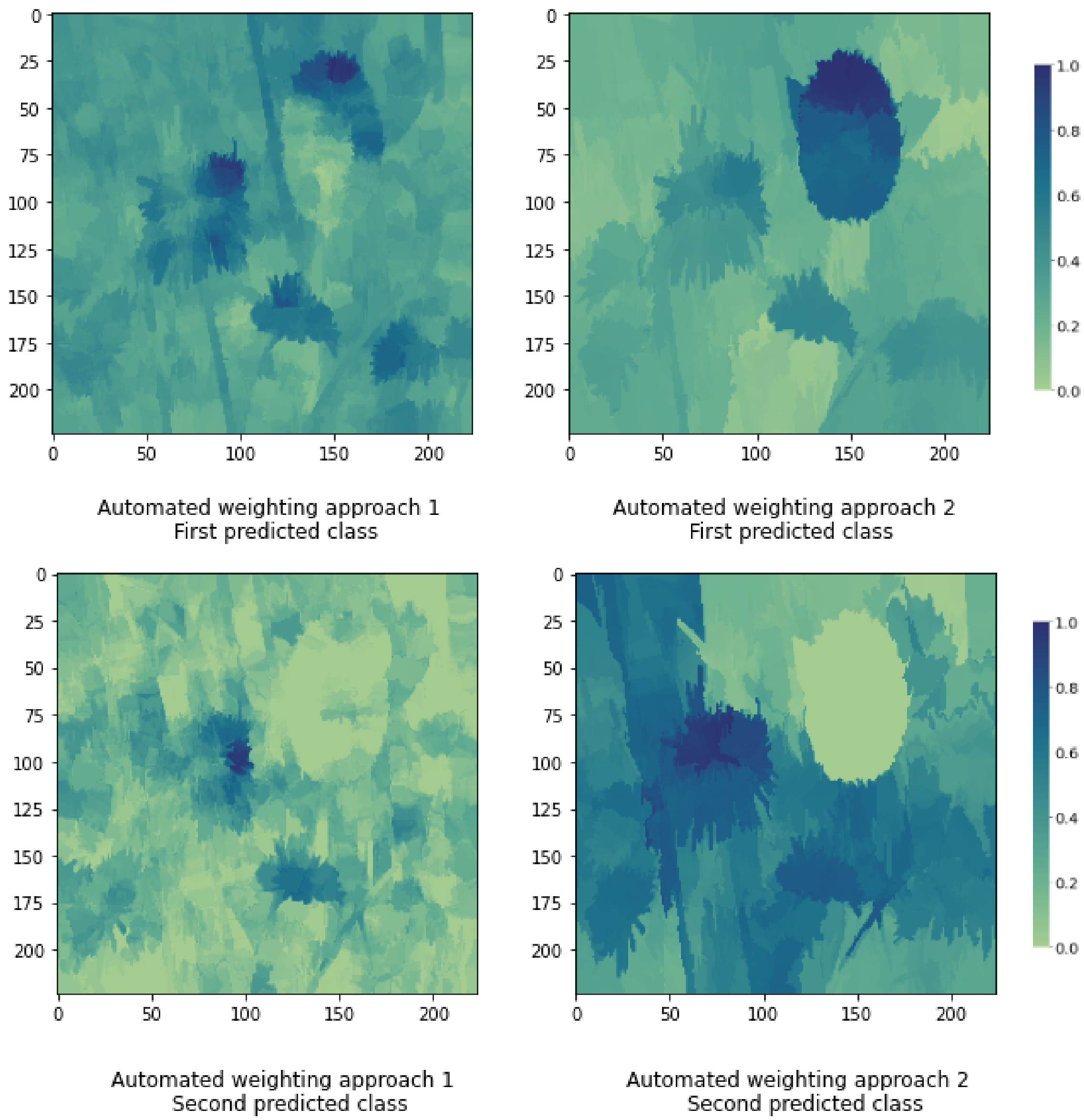


Figure 5.22: Evaluation of class discriminative aspect: automated weighting approaches for sample image dandelion

where the importance is obtained from the importance map. A sharp drop and thus a low area under the probability curve means a good explanation. In this section, we calculated the predicted probability after removing top superpixels. In other words, salient superpixels/regions are gradually masked from the input image by setting pixels to zero, then the masked image will be passed to the black-box model.

The probability of the sample image "*tulips*" is 0.98. We gradually removed the top superpixels with an increment of 1 and passed the new sample image to the model. Fig 5.23 shows deletion of top superpixels one by one in segmentations 1 to 5. In Fig 5.23, the first row represents the removal of the top two superpixels led to the prediction accuracy near zero. The second row corresponds to the deletion of superpixels with an increment of 1 in segmentation 2. This process continued, and the six sample images corresponding to the deletion of top superpixels in segmentation 3 to 5 are shown. To see how AUC changed and the probabilities varied to become zero/near zero in higher levels with a large number of superpixels/regions, we decided to remove top regions with an increment of 5, then check the probabilities in Fig 5.24, where x-axis shows the number of masked images, and y-axis shows the prediction accuracy corresponding to each masked image. Each plot shows a drop in the prediction accuracy after deletion of top superpixels with an increment of 5. In other words, we removed the top 5 superpixels at each level/segmentation and then passed the created masked image to the black-box. A sharp drop and thus a low area under the probability curve shows that LIME can perform as expected for a range of scales. Since there are multiple objects of the same class in the masked images, some regions of other objects still got a chance to be detected by the black-box and represented by the multi-scale scheme, specifically at lower levels where the superpixels are too fine. That is why we can see a few significant increases in predicted probabilities at some steps. Finally, all the probabilities will get near zero, proving that the explanation is faithful and the model being explained could find the most relevant regions even by looking at finer scales through the proposed multi-scale scheme.

5.4.2 Explanation Accuracy

In recent research, a methodology has been proposed in [103] toward quantifying the reliability of the explanations procured as an outcome from XAI algorithms, and proceeds in the direction of building user trust in the black-box models. The ideology behind the proposed quantifiable system is that salient regions detected by XAI algorithms should be sufficient to obtain approximately similar prediction accuracy, if not higher than the accuracy obtained when the entire attribute set is provided to the model. In other words, the proposed quantifying system feeds the explanations of the input features obtained from the XAI algorithm to the black-box model as input and notes its accuracy and prediction label. This information is then compared to the accuracy and prediction label computed when the original input feature subset is passed as input to the same black-box model. The whole idea is that since the algorithm already captures the most influential features during the training phase, sending the explanations would aid in quantifying the reliability of the built model. Initially, Prediction Accuracy "*pa*" is computed when the entire feature subset or input image is passed to a black-box model and the corresponding prediction class is noted. The prediction is then utilized in computing the model's explanation and a feature subset. More precisely, by sending the initial input image with its corresponding prediction to an XAI algorithm, salient features of the original input are obtained. This obtained feature subset containing the most influential features in computing the prediction is then passed to the aforementioned black-box model, and the prediction label with its accuracy is observed and noted as explanation accuracy "*ea*".

Since the weighted combination of the heatmaps with Gaussian function provided enough variation from coarse to fine scales, we evaluated the explanation accuracy at these three scales. Therefore, each of these weighted heatmaps is multiplied by the input image, then the output, which is a masked image where some part of the superpixels are

Figure 5.23: Masked images after removing top superpixels with an increment of 1.

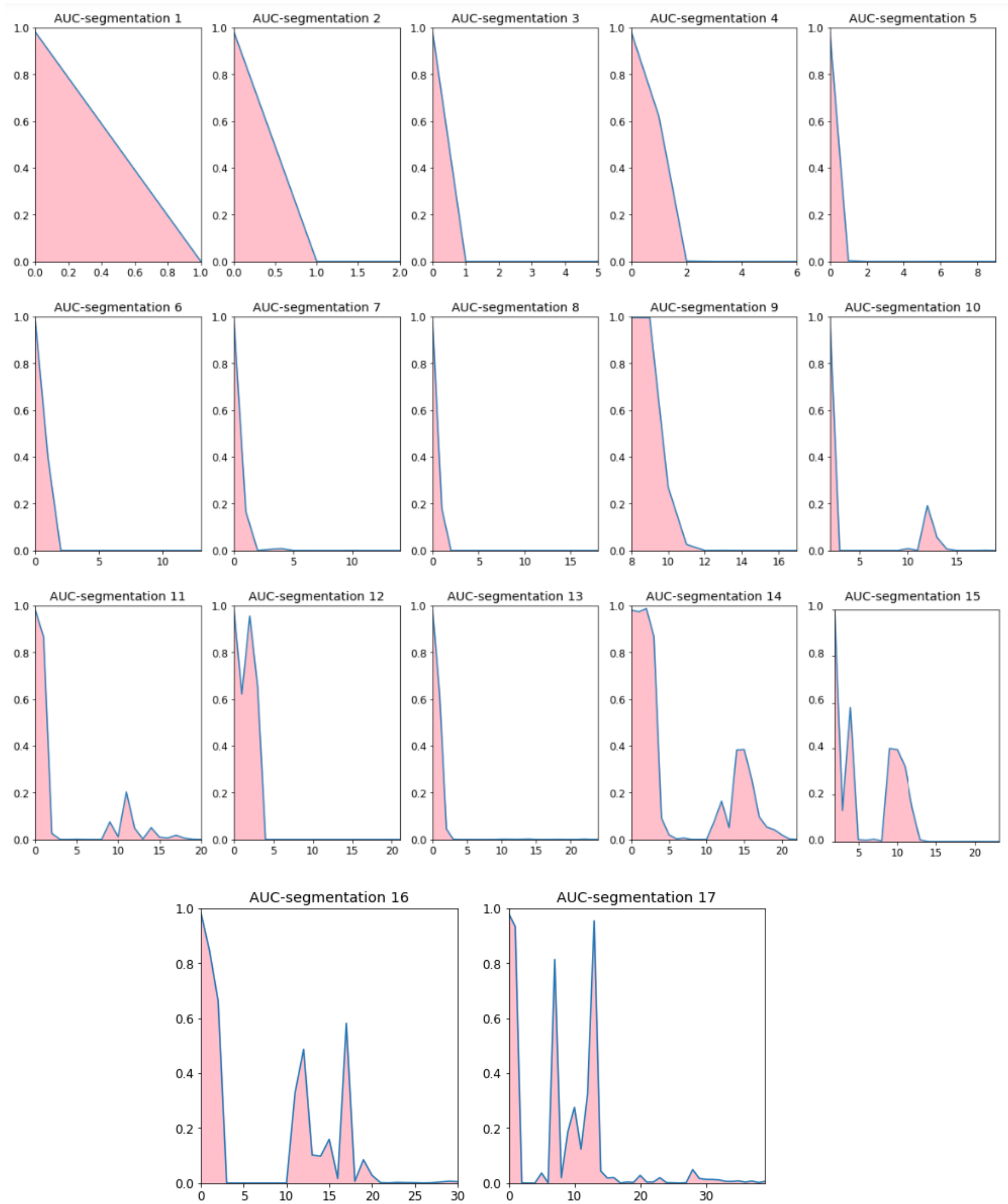


Figure 5.24: AUC for sample image "tulips". The predicted probabilities after deletion of top 5 superpixels in each sample image.

	explanation accuracy (ea)	ea of square grids	es of multi-scale scheme
Original LIME	0.98	-	-
Coarse-scale	-	0.96	1.00
Finer-scale	-	0.94	0.99
Finest-scale	-	0.90	0.99

Table 5.6: Quantitative results of multi-scale scheme w.r.t. the original LIME, and square-grids segmentation. **Sample image: tulips**. The prediction accuracy of the input image is 0.98. The explanation accuracies with square-grid segmentations are decreased, which is against what we expect from an XAI method.

gray, will be passed to the same black-box, and explanation accuracy will be observed. Figs 5.25, 5.26, and 5.27 show the masked images after multiplying the input image by the corresponding heatmaps for "*tulips*", "*daisy*", and "*dandelion*", respectively. We compared the *ea* of the multi-scale scheme with the original LIME for each sample image in Tables 5.6, 5.7, and 5.8.

5.5 Histopathology Cancer Detection

In [95] the square-grids segmentation has been tested on Camelyon16 dataset with VGG19 network as the black-box model. Since the results of VGG19 network has been proved on this dataset, we chose the same black-box model with the same dataset to assess the results of proposed multi-scale scheme. This model consisted of a CNN with 0.968 prediction accuracy on the test set [95]. The only change we made was in the last layer of the network to get the prediction probabilities through softmax activation function, then trained the model on the test set. This section aims to evaluate the visual explanations of proposed multi-scale scheme with the automated weighting approaches on Camelyon 16. Therefore, we compared the results with the square-grids segmentations qualitatively,

	explanation accuracy (ea)	ea of square grids	ea of multi-scale scheme
Original LIME: predicted dandelion	0.65	-	-
Coarse-scale	-	0.89	0.97
Finer-scale	-	0.87	0.95
Finest-scale	-	0.87	0.94

Table 5.7: Quantitative results of multi-scale scheme w.r.t. the original LIME, and square-grids segmentation. **Sample image: daisy**. The *pa* of this sample is 0.86 and LIME led to a wrong prediction in this sample, *dandelion*. The explanation accuracies are not significantly increased by the explanations of square-grids segmentation, but the multi-scale scheme led to 10 percent increase in accuracies obtained from each scale.

	explanation accuracy (ea)	ea of square grids	ea of multi-scale scheme
Original LIME:	0.73	-	-
Coarse-scale	-	0.76	1.00
Finer-scale	-	0.83	0.99
Finest-scale	-	0.62	0.97

Table 5.8: Quantitative results of multi-scale scheme w.r.t. the original LIME, and square-grids segmentation. **Sample image: dandelion**. The *pa* of this sample is 0.83, and the explanation accuracies of the square-grids segmentation were decreased. Generally, the multi-scale scheme provides significantly better results than LIME and its extension with square-grids segmentation.

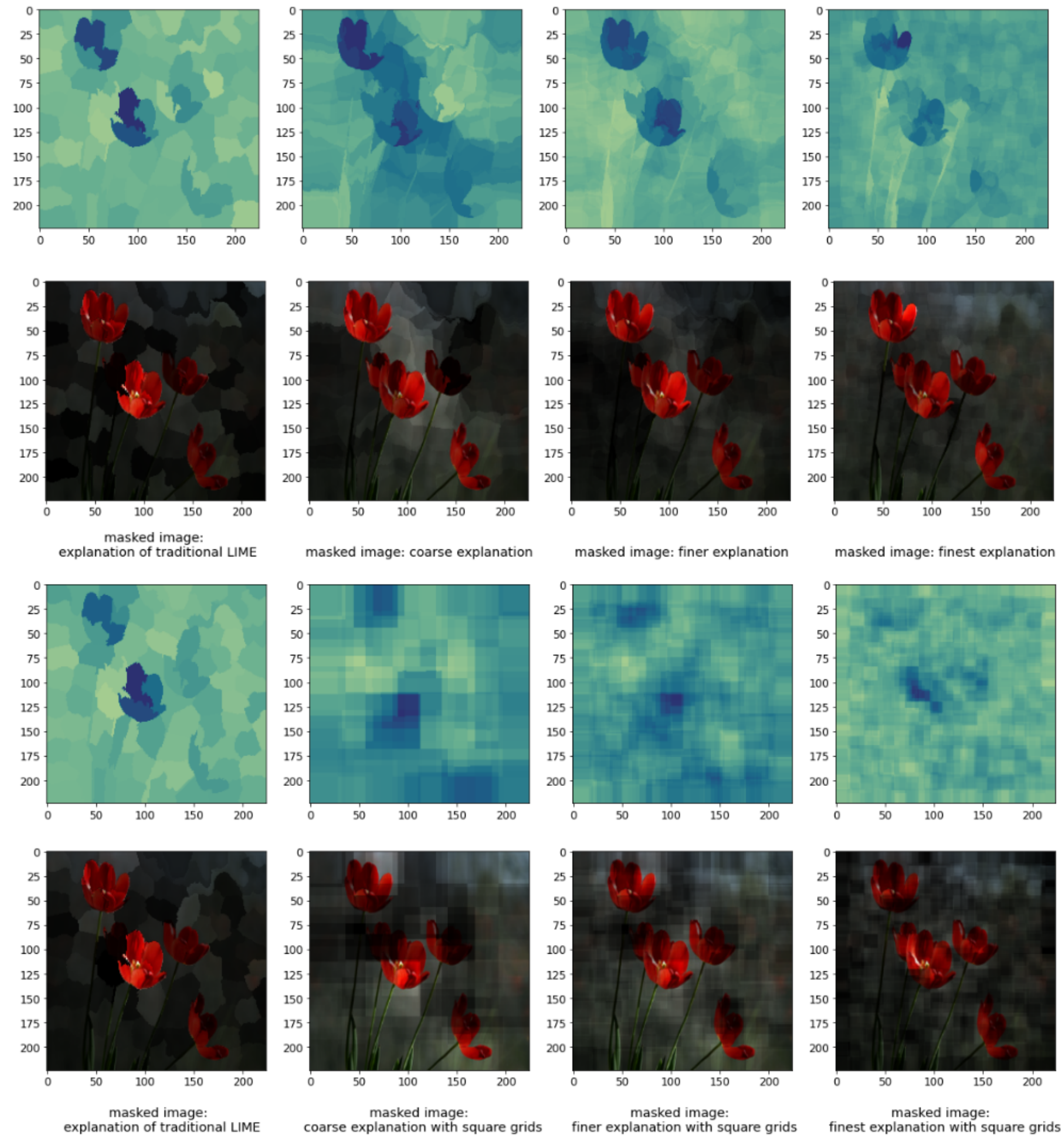


Figure 5.25: Sample image: tulips. Masked images of multiplying input image by the explanation heatmap obtained from each scale. We passed each of these images to the same black-box and the *ea* is obtained. As Table 5.6 shows, the explanation accuracies are decreased by square-grids segmentation. On the other hand, explanation accuracies are slightly increased through the multi-scale scheme, which became around 1.

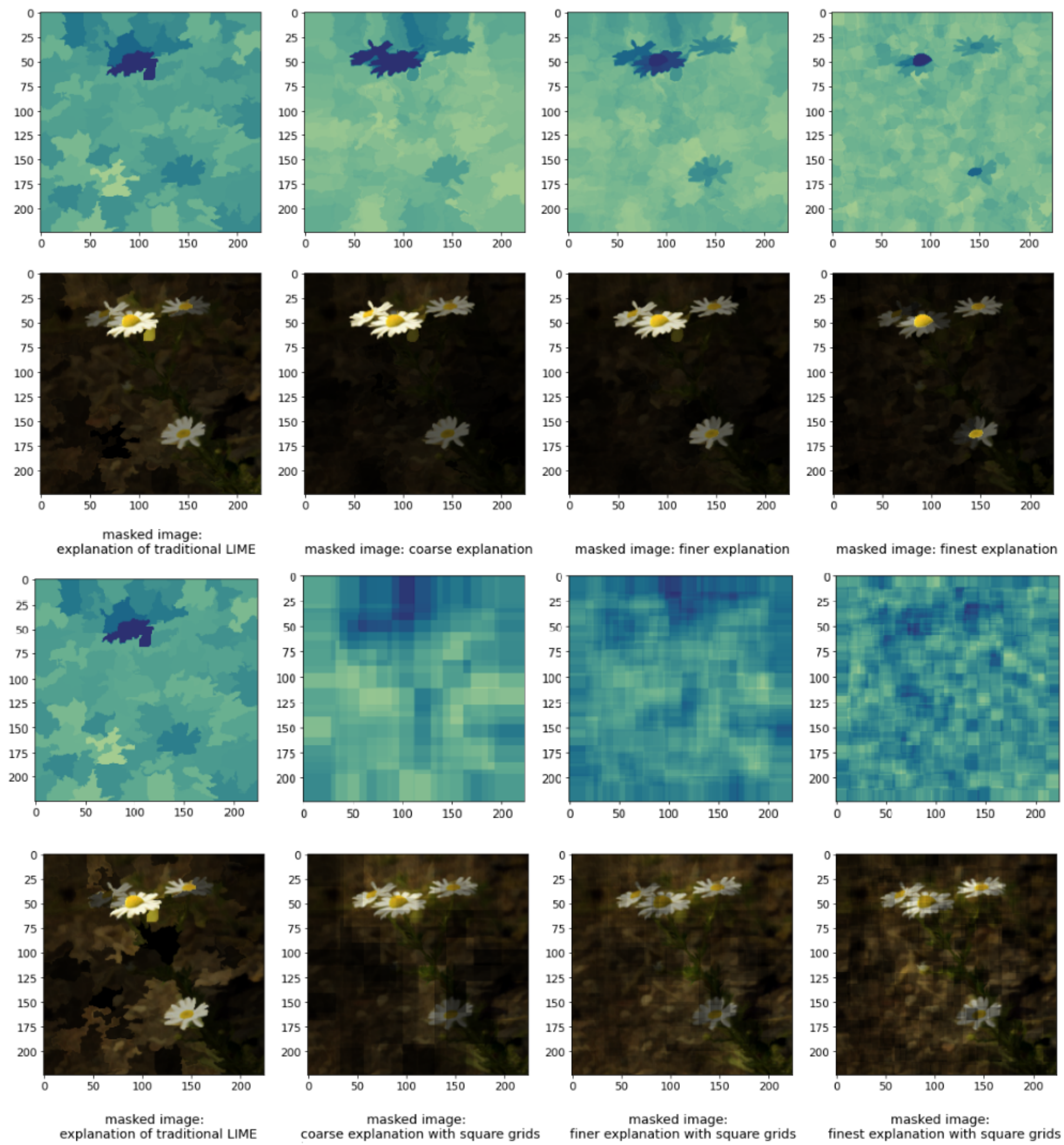


Figure 5.26: Sample image: daisy. Masked images of multiplying input image by the explanation heatmap obtained from each scale. In this sample, the explanation obtained from traditional LIME led to a wrong prediction, *dandelion*, see Table 5.7.

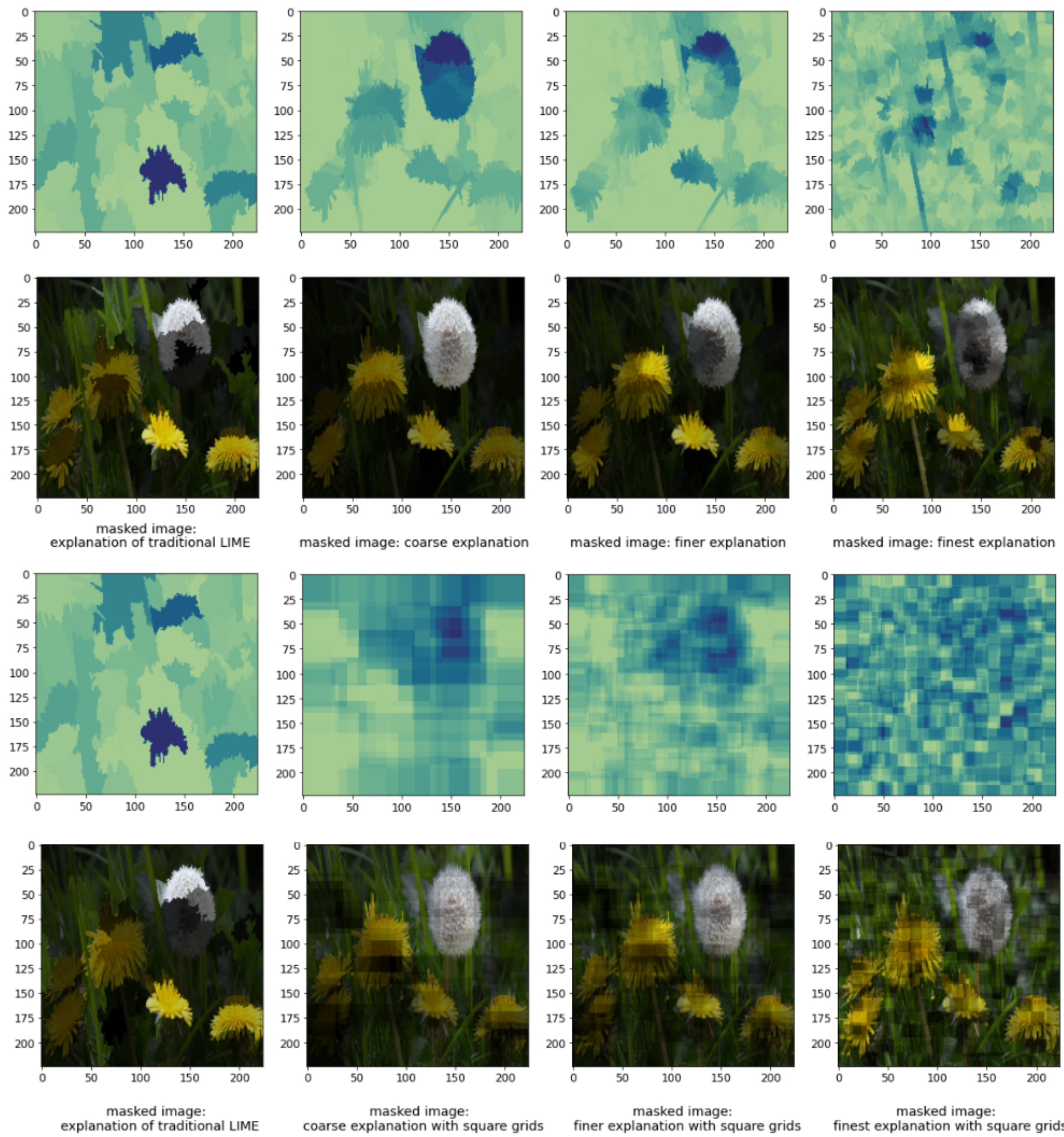


Figure 5.27: Sample image: dandelion. Masked images of multiplying input image by the explanation heatmap obtained from each scale. The prediction accuracy of the input image is 0.83, and Table 5.8 shows the explanation accuracies of traditional LIME and the square-grids segmentations are decreased. However, the multi-scale scheme caused the accuracies got around 0.99, and 1.

and quantitatively.

5.5.1 Dataset

Lymph Node Metastases on Camelyon16

Patch Camelyon (P-CAM) is a dataset developed by Veeling et al. [124]. It was derived from the Camelyon16 hematoxylin and eosin-stained WSIs. The original whole slide images were acquired and digitized with a 40× objective (corresponding to a pixel resolution of 0.243 microns) [12], and undersampled at 10× to increase the field of view for P-CAM. The 96 by 96 pixel patches were extracted by [124] from the gigapixel WSIs by converting the slides to hue-saturation-value format (HSV), followed by blurring and filtering out patches that had saturation lines below 0.07, which was shown to exclude irrelevant background patches [95]. The binary labels of each image corresponded to the presence or absence of at least one pixel of tumor tissue in the central 32 by 32 pixel square of each patch. 0 meaning absence, and 1 meaning the presence of tumor tissue. Tumor tissue outside this square did not influence the label. However, this outer region was maintained in the image, both to it possibly giving relevant context for classification algorithms and to enable the usage of certain types of fully-convolutional models that do not use zero-padding [122]. The dataset was presented in two versions. The first one was a GitHub repository [122] containing the full dataset of 327,680 patches divided into training validation and test sets following the same division of the original Camelyon16. The second version was the same as the first, but removed duplicate images present in the original P-CAM due to the probabilistic sampling of patches. This version was presented as a dataset for a Kaggle competition [123]. Besides the removal of duplicates, there is no differences between the splits or other aspects of the data. There are 220,026 labelled images in the second version. For the remainder of this thesis, we work on the second version of this dataset downloaded from Kaggle. Two sample image from Camelyon16

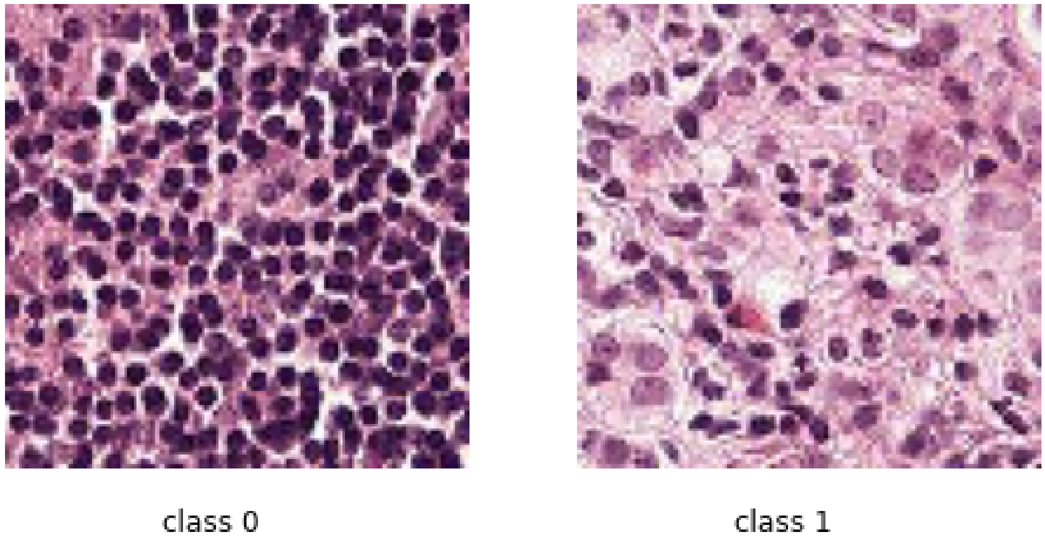


Figure 5.28: Sample images from Camelyon 16.

dataset are shown in Fig 5.28.

5.5.2 Explanation Results of Multi-scale Scheme on Biological Dataset

We chose two sample images from Camelyon 16 dataset with the actual label 1. These samples are correctly classified as true positive with the VGG19 network and the prediction accuracy is 0.741, and 0.834, respectively, see Figs 5.29, and 5.32. For the first sample image, the results of multi-scale scheme and the combined heatmaps by the automated weighting approaches are presented in Fig 5.29. To compare the results quantitatively, we multiplied the sample image by each of these heatmaps, and the masked images are shown in Fig 5.31, along with the *ea* corresponding to each image. As the results show, the *ea* obtained from traditional LIME is 0.548, which caused a significant decrease in the prediction accuracy of the original input image. Although the square-grids segmentation can detect some superpixels/regions by weighting approach 1, the results of the multi-scale scheme are more convincing even at finer scales. The explanation accuracies

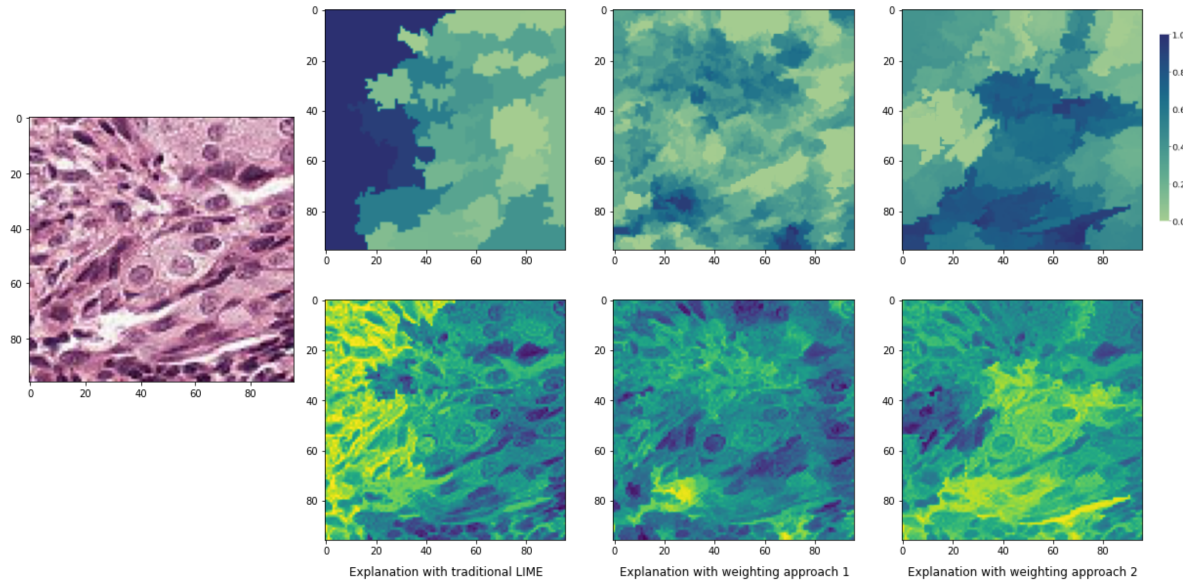


Figure 5.29: True positive sample image from Camelyon16. Results of the multi-scale scheme with automated weighting approaches. Prediction accuracy of this sample image with VGG19 was 0.741, and reached to 0.82 by multi-scale scheme

of multi-scale scheme at coarse and fine scales have been increased to 0.82, and 0.79, respectively. Visually speaking, the localization of multi-scale scheme compare to the square-grids is much more convincing, specifically at finer scale.

The same results were observed for the second sample image as well. The localization of the multi-scale scheme compared to the traditional LIME and square-grids segmentation is provided in Fig 5.33. The heatmaps of the multi-scale scheme referred to different regions/superpixels of an input image. Hence, we were motivated to check the explanation accuracies and compare the results with traditional LIME and square-grids segmentation in Fig 5.34. As the results prove, the *ea* obtained from traditional LIME reached 0.691, which means the explanation obtained from traditional LIME does not contain the most important regions of the input image. On the other hand, the values of *ea* increased through the proposed multi-scale scheme compared to traditional LIME and square-grids segmentation, Fig 5.34.

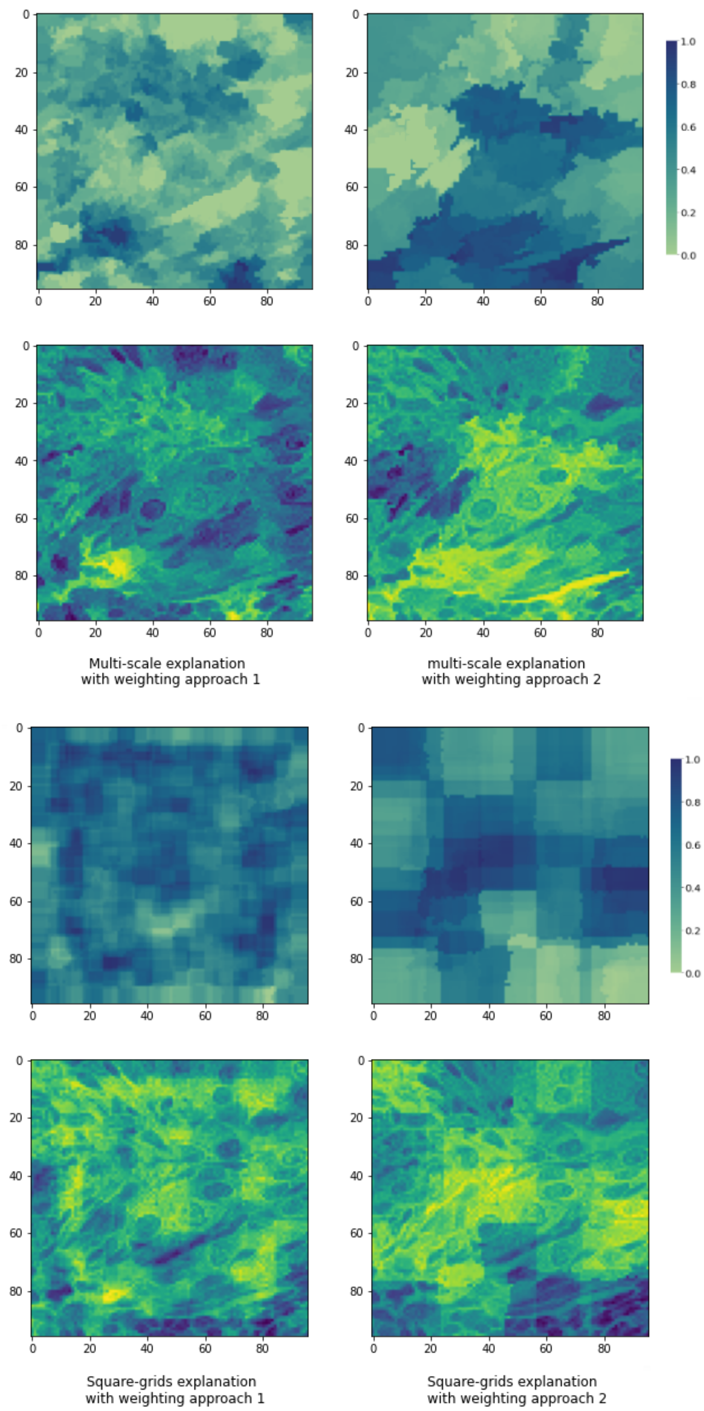


Figure 5.30: Localization of multi-scale scheme vs square grids explanation. True positive sample image from Camelyon16 dataset.

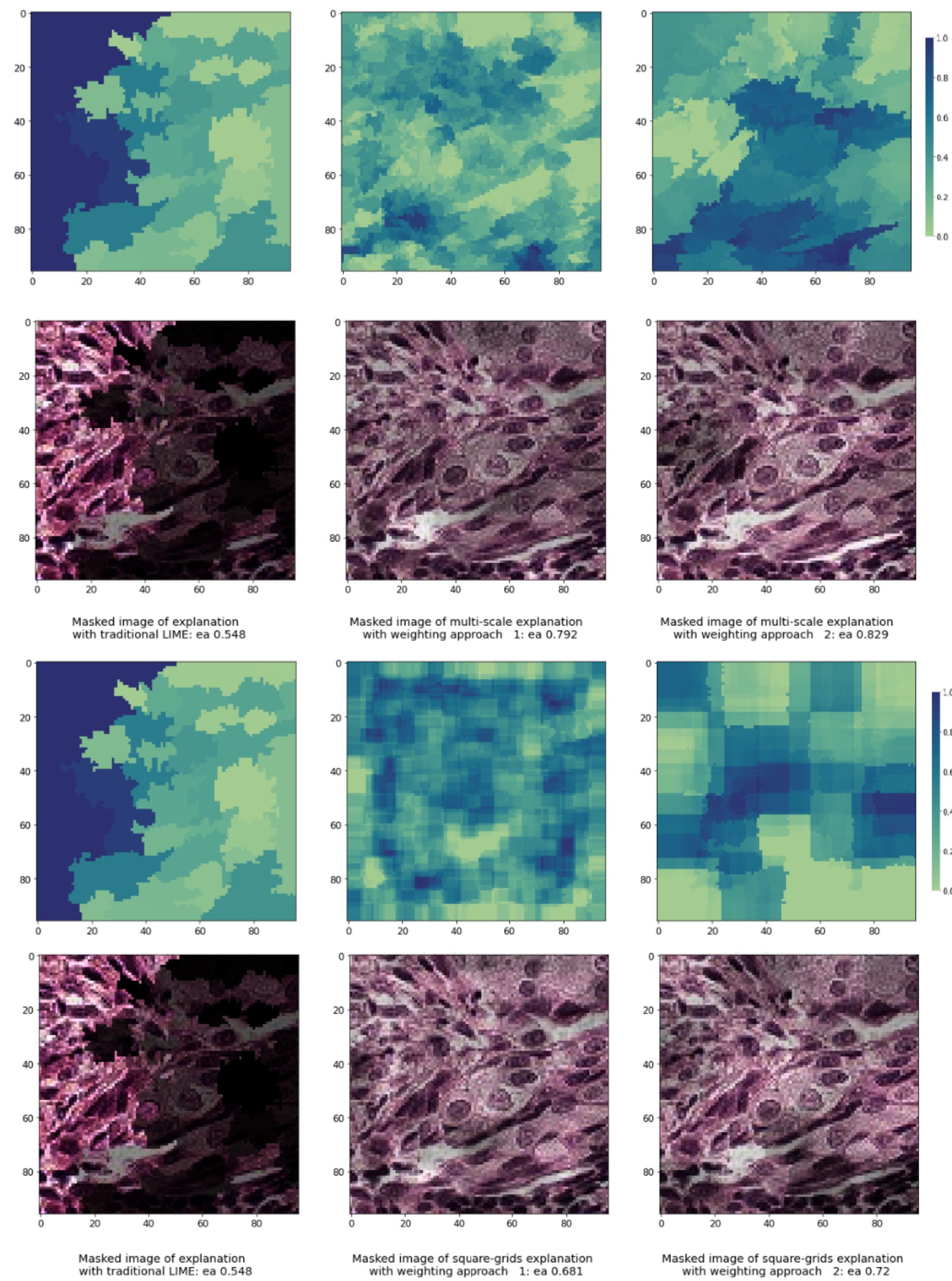


Figure 5.31: Quantitative results: Explanation accuracies of multi-scale scheme vs square-grids and traditional LIME. The *pa* for this sample is 0.741.

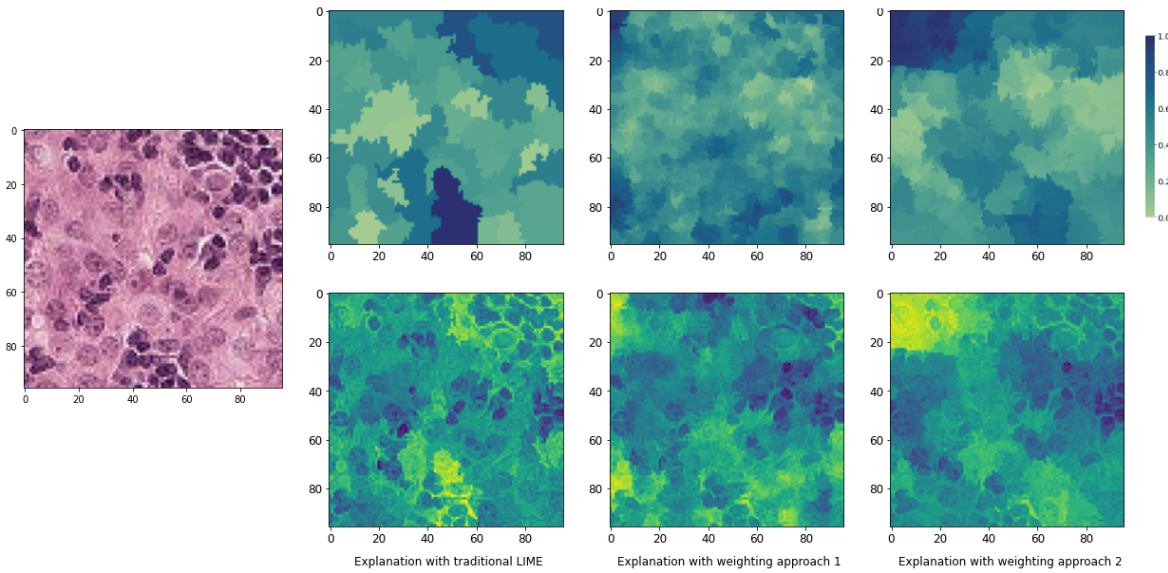


Figure 5.32: True positive second sample image from Camelyon16. Results of the multi-scale scheme with automated weighting approaches. Prediction accuracy of this sample image with VGG19 was 0.83, and reached to 0.89 by multi-scale scheme

5.6 Summary

In this chapter, we evaluated the results of the proposed multi-scale scheme of LIME on Flower Dataset from TFDS and a Biological dataset, Camelyon 16, along with the fine-tuned networks trained explicitly for these two classification problems. Finally, we compared the explanation results obtained from the multi-scale version of LIME with the traditional LIME and the square-grids segmentation proposed in the literature. This study proves that the explanations provided as heatmaps of the proposed multi-scale scheme are significantly better than the previous extensions of LIME, qualitatively and quantitatively. We also proposed two ways to calculate the weighted heatmaps and focus on the influential regions from coarse to fine scale using Gaussian function and the parameter-free automated weighting approaches.

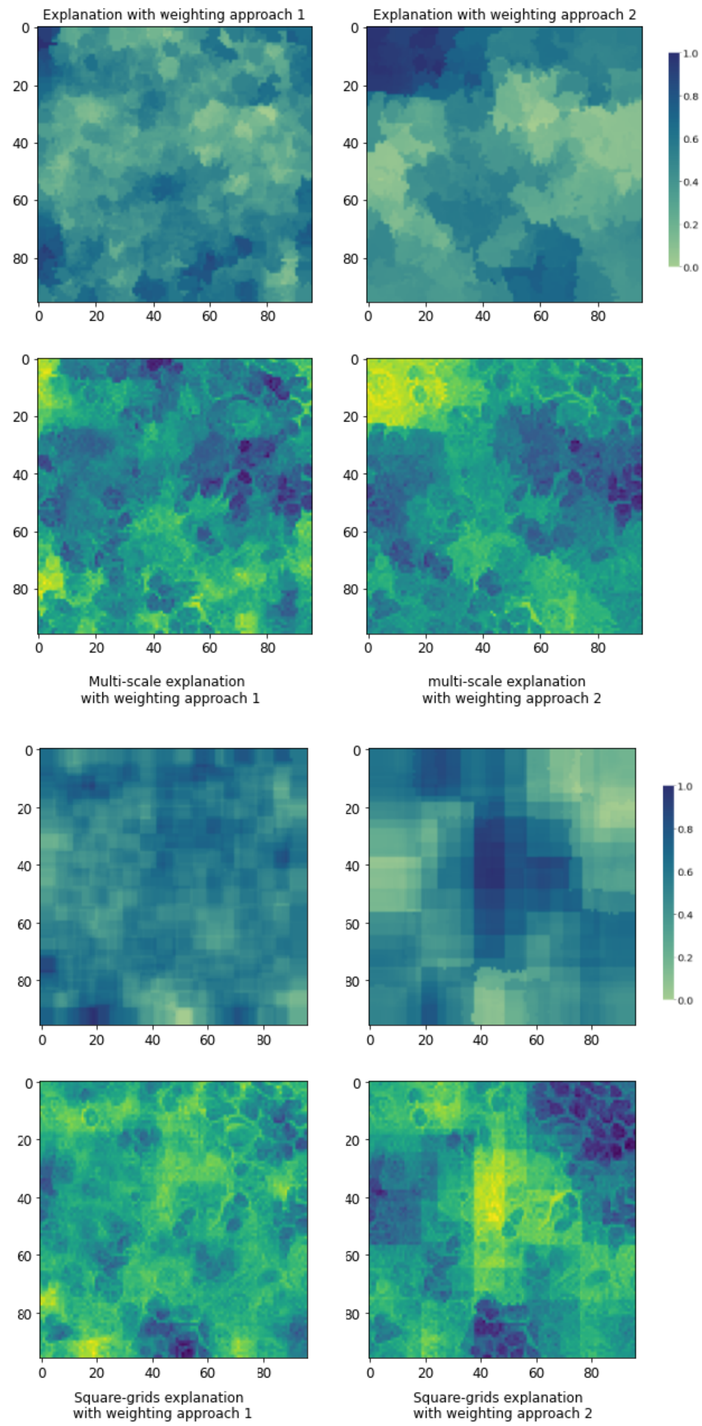


Figure 5.33: Localization of multi-scale scheme vs square-grids explanation. True positive second sample image from Camelyon16 dataset.

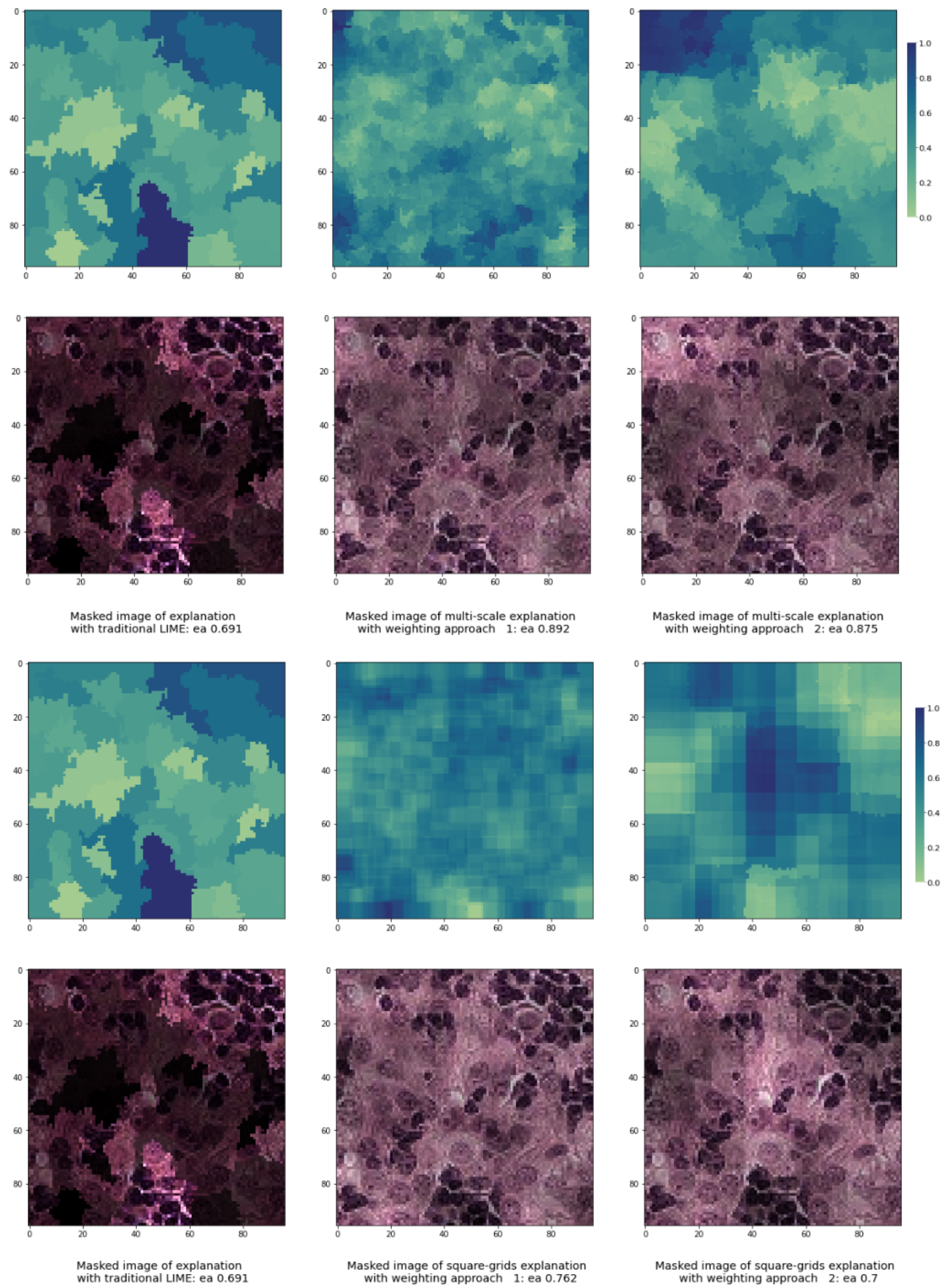


Figure 5.34: Quantitative results: Explanation accuracies of multi-scale scheme vs square-grids and traditional LIME. The *pa* for this sample is 0.83.

Chapter 6

Conclusion and Future work

This research aimed to study the effectiveness of LIME as a model-agnostic explanation approach for interpreting the decision of black-box models and address some of its shortcomings corresponding to images.

We proposed a multi-scale scheme to enhance visual explanations of LIME at different scales through the results from attentive heatmaps that can produce more comprehensive explanations from coarse to fine scale. In other words, this thesis contributed to the extension of LIME with local explanations obtained from different scales. This thesis also introduced weighting approaches with Gaussian distribution and a parameter-free framework for producing heatmaps of different levels as visual explanations. Therefore, this Chapter summarizes the main contributions presented in this thesis and offers promising future research directions.

6.1 Thesis Contribution Highlights

The main contribution of this thesis in Chapter 4 can be summarized as follows:

- **Multi-scale scheme for visual explanations of LIME**

  Since LIME is too sensitive to superpixel segmentation, we segment an image into the desired number of superpixels at each level, then apply LIME and fit a linear model to achieve a visual explanation on heatmaps that represent how strong each patch is correlated with the classifier decision. In this multi-scale framework, first, an input image is segmented into large patches, coarse scale segmentation, at the first level, and the patches will get smaller at the last level. Hence, through this multi-scale scheme, LIME will focus on the same patch at coarse to finer scales to interpret how important each tiny piece of an image is toward the classifier decision.

- **Calculating weighted heatmaps obtained from multi-scale scheme**

  This thesis presents two ways to calculate the weighted heatmaps and produce visual explanations from coarse, finer and finest scales. Firstly, we use Gaussian distribution when the mean is shifted to the left and right to produce various explanations of different scales. Secondly, we proposed parameter-free automated weighting approaches to assign weights to the heatmaps based on the difference in the number of superpixels. To be more precise, if the number of superpixels is close at two sequential levels, they might probably produce similar results. Therefore, the explanations obtained from these levels will get low weights. On the opposite, a large gap between the number of superpixels of two levels will achieve various explanation results. The results of automated weighting approaches are aligned with the explanation results obtained from Gaussian distribution weighting at coarse and finest scales.

The results show this multi-scale scheme of LIME achieves highly accurate results compared to the square-grids segmentation, presented in the literature, as the objects' boundaries remain in this approach and the explanations are more localized. Furthermore, the quantitative results prove that LIME can perform as expected for a range of scales, and the explanations would be better or at least equal to the original LIME, qualitatively and quantitatively.

6.2 Limitations

The qualitative and quantitative experimental results demonstrated that the proposed methods produce quite interpretable results. However, there are some limitations accompanied with this method which should be considered.

- Since the number of superpixels are different at each level, the implementation of LIME and fitting of the linear model are done separately, which is time-consuming in large-scale images.

- To check the results quantitatively, we calculated AUC to make sure that LIME can detect relevant superpixels, even at the very fine scale segmentation level, besides the explanation accuracies, which produced convincing results. Since the datasets used in this thesis do not have ground truth, we could not evaluate the localization aspect of the proposed method quantitatively. There is a metric called "Pointing Game" which could assess the explanation method and count how many pixels with the highest score are located inside the ground truth area.

6.3 Future Work

The proposed methods in this thesis open several new directions for future work. Since the number of superpixels/coefficients is varied in each level of the proposed multi-scale scheme, LIME should be implemented separately. We suggest creating a projection matrix of joining segmentation obtained from coarse to fine scales, then using the projection matrix to fit the linear model. We can overlap tiny superpixels from finer scale segmentation to the corresponding region at coarse scale segmentation. Therefore, the masked images of each scale can be mapped to the joint segmentation in the projection matrix, where the number of superpixels/coefficients are constant. So, a single linear model will be fitted to the perturbations. Furthermore, instead of taking average from heatmaps, the sampling can be modified in LIME instead of taking an equal number of samples/perturbations at each level. Sampling in the proposed multi-scale scheme could also be modified. For example, the proposed weighting approaches, either Gaussian distribution or parameter-free automated approaches, could be used to define the number of samples at each level. The proposed scheme could also be implemented along with various segmentation algorithms in the literature, such as Quickshift, and Felzenszwalb, to compare the effectiveness of the segmentation algorithm in LIME.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels. Technical report, 2010.

[2] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6:52138–52160, 2018.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[4] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. innvestigate neural networks! *J. Mach. Learn. Res.*, 20(93):1–8, 2019.

[5] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31, 2018.

[6] David Alvarez-Melis and Tommi S Jaakkola. A causal framework for explaining the predictions of black-box sequence-to-sequence models. *arXiv preprint arXiv:1707.01943*, 2017.

[7] Plamen P Angelov, Eduardo A Soares, Richard Jiang, Nicholas I Arnold, and Peter M Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1424, 2021.

[8] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[9] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[10] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019.

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

[12] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes Van Diest, Bram Van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen AWM Van Der Laak, Meyke Hermsen, Quirine F Manson, Maschenka Balkenhol, et al. Diagnostic assessment

of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *Jama*, 318(22):2199–2210, 2017.

[13] Jyostna Devi Bodapati and Naralasetti Veeranjaneyulu. Feature extraction and classification using deep convolutional neural networks. *Journal of Cyber Security and Mobility*, pages 261–276, 2019.

[14] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[15] Urszula Chajewska and Joseph Y Halpern. Defining explanation in probabilistic systems. *arXiv preprint arXiv:1302.1526*, 2013.

[16] A Chattopadhyay, A Sarkar, P Howlader, and VN Balasubramanian. Grad-CAM++: Improved visual explanations for deep convolutional networks. arxiv 2017. *arXiv preprint arXiv:1710.11063*.

[17] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection. *arXiv preprint arXiv:2004.02015*, 2020.

[18] Hugh Chen, Scott Lundberg, and Su-In Lee. Explaining models by propagating shapley values of local components. In *Explainable AI in Healthcare and Medicine*, pages 261–270. Springer, 2021.

[19] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[20] Paulo Cortez and Mark J Embrechts. Opening black box data mining models using sensitivity analysis. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 341–348. IEEE, 2011.

[21] Danilo Croce, Daniele Rossini, and Roberto Basili. Explaining non-linear classifier decisions within kernel-based deep architectures. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 16–24, 2018.

[22] Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. Explainable software analytics. In *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, pages 53–56, 2018.

[23] Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. A survey of the state of explainable AI for natural language processing. *arXiv preprint arXiv:2010.00711*, 2020.

[24] Maartje MA De Graaf and Bertram F Malle. How people explain action (and autonomous intelligent systems should too). In *2017 AAAI Fall Symposium Series*, 2017.

[25] Anamika Dhillon and Gyanendra K Verma. Convolutional neural network: a review of models, methodologies and applications to object detection. *Progress in Artificial Intelligence*, 9(2):85–112, 2020.

[26] Wang Di. A comparative research on clothing images classification based on neural network models. In *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT*, pages 495–499. IEEE, 2020.

[27] Jürgen Dieber and Sabrina Kirrane. Why model why? assessing the strengths and limitations of lime. *arXiv preprint arXiv:2012.00093*, 2020.

[28] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

[29] Rachel Lea Draelos and Lawrence Carin. Hirescam: Faithful location representation in visual attention for explainable 3D medical image classification. *arXiv preprint arXiv:2011.08891*, 2020.

[30] Mary T Dzindolet, Scott A Peterson, Regina A Pomranky, Linda G Pierce, and Hall P Beck. The role of trust in automation reliance. *International journal of human-computer studies*, 58(6):697–718, 2003.

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

[32] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1473–1482, 2015.

[33] Parastoo Farnia, Mohammad Mohammadi, Ebrahim Najafzadeh, Maysam Alimohamadi, Bahador MakkiAbadi, and Alireza Ahmadian. High-quality photoacoustic image reconstruction based on deep convolutional neural network: towards intra-operative photoacoustic imaging. *Biomedical Physics & Engineering Express*, 6(4):045019, 2020.

[34] Jean-Marc Fellous, Guillermo Sapiro, Andrew Rossi, Helen Mayberg, and Michele Ferrante. Explainable artificial intelligence for neuroscience: behavioral neurostimulation. *Frontiers in neuroscience*, page 1346, 2019.

[35] Tomas Folke, Scott Cheng-Hsin Yang, Sean Anderson, and Patrick Shafto. Explainable AI for medical imaging: explaining pneumothorax diagnoses with bayesian teaching. *arXiv preprint arXiv:2106.04684*, 2021.

[36] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pages 3429–3437, 2017.

[37] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *2009 IEEE 12th international conference on computer vision*, pages 670–677. IEEE, 2009.

[38] Glenn Fung, Sathyakama Sandilya, and R Bharat Rao. Rule extraction from linear support vector machines. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 32–40, 2005.

[39] Chuang Gan, Naiyan Wang, Yi Yang, Dit-Yan Yeung, and Alex G Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2568–2577, 2015.

[40] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question. *Advances in neural information processing systems*, 28, 2015.

[41] Damien Garreau and Dina Mardaoui. What does lime really see in images? In *International Conference on Machine Learning*, pages 3620–3629. PMLR, 2021.

[42] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[43] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 241–246. IEEE, 2016.

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

[45] Lukasz Gorski, Shashishekar Ramakrishna, and Jedrzej M Nowosielski. Towards Grad-CAM based explainability in a legal text processing pipeline. *arXiv preprint arXiv:2012.09603*, 2020.

[46] Mara Graziani, Thomas Lompech, Henning Müller, and Vincent Andrearczyk. Evaluation and comparison of cnn visual explanations for histopathology. *Explainable Agency in Artificial Intelligence at AAAI21*, pages 195–201, 2020.

[47] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[48] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. XAI—explainable artificial intelligence. *Science Robotics*, 4(37):eaay7120, 2019.

[49] Taehyun Ha, Sangwon Lee, and Sangyeon Kim. Designing explainability of an artificial intelligence system. In *Proceedings of the Technology, Mind, and Society*, pages 1–1. 2018.

[50] Maaike Harbers, Karel van den Bosch, and John-Jules Ch Meyer. A study into preferred explanations of virtual agent behavior. In *International Workshop on Intelligent Virtual Agents*, pages 132–145. Springer, 2009.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[52] Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Grounding visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 264–279, 2018.

[53] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[54] Andreas Holzinger, Chris Biemann, Constantinos S Pattichis, and Douglas B Kell. What do we need to build explainable AI systems for the medical domain? *arXiv preprint arXiv:1712.09923*, 2017.

[55] Xavier Alphonse Inbaraj, Charlyn Villavicencio, Julio Jerison Macrohon, Jyh-Horng Jeng, and Jer-Guang Hsieh. Object identification and localization using Grad-CAM++ with mask regional convolution neural network. *Electronics*, 10(13):1541, 2021.

[56] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[57] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[58] Ulf Johansson, Cecilia Sönströd, Ulf Norinder, and Henrik Boström. Trade-off between accuracy and interpretability for predictive in silico modeling. *Future medicinal chemistry*, 3(6):647–663, 2011.

[59] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4565–4574, 2016.

[60] Md Rezaul Karim, Till Döhmen, Michael Cochez, Oya Beyan, Dietrich Rebholz-Schuhmann, and Stefan Decker. Deepcovidexplainer: explainable COVID-19 diagnosis from chest X-ray images. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1034–1037. IEEE, 2020.

[61] Md Rezaul Karim, Jiao Jiao, Till Doehmen, Michael Cochez, Oya Beyan, Dietrich Rebholz-Schuhmann, and Stefan Decker. Deepkneeexplainer: Explainable knee osteoarthritis diagnosis from radiographs and magnetic resonance imaging. *IEEE Access*, 9:39757–39780, 2021.

[62] Sujata Khedkar, Vignesh Subramanian, Gayatri Shinde, and Priyanka Gandhi. Explainable AI in healthcare. In *Healthcare (April 8, 2019). 2nd International Conference on Advances in Science & Technology (ICAST)*, 2019.

[63] Hojin Kim, Jinhong Jung, Jieun Kim, Byungchul Cho, Jungwon Kwak, Jeong Yun Jang, Sang-wook Lee, June-Goo Lee, and Sang Min Yoon. Abdominal multi-organ auto-segmentation using 3D-patch-based deep convolutional neural network. *Scientific reports*, 10(1):1–9, 2020.

[64] Eyal Klang. Deep learning and medical imaging. *Journal of thoracic disease*, 10(3):1325, 2018.

[65] Christof Koch and Shimon Ullman. Selecting one among the many: A simple network implementing shifts in selective visual attention. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1984.

[66] Sajja Tulasi Krishna and Hemantha Kumar Kalluri. Deep learning and transfer learning approaches for image classification. *International Journal of Recent Technology and Engineering (IJRTE)*, 7(5S4):427–432, 2019.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[68] Devinder Kumar. Class based strategies for understanding neural networks. 2020.

[69] Wooju Lee, Donggyu Sim, and Seoung-Jun Oh. A CNN-based high-accuracy registration for remote sensing images. *Remote Sensing*, 13(8):1482, 2021.

[70] Piyawat Lertvittayakumjorn and Francesca Toni. Human-grounded evaluations of explanation methods for text classification. *arXiv preprint arXiv:1908.11355*, 2019.

[71] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2290–2297, 2009.

[72] Xiao-Hui Li, Yuhan Shi, Haoyang Li, Wei Bai, Yuanwei Song, Caleb Chen Cao, and Lei Chen. Quantitative evaluations on saliency methods: An experimental study. *arXiv preprint arXiv:2012.15616*, 2020.

[73] Ying Liang, Diane Schott, Ying Zhang, Zhiwu Wang, Haidy Nasief, Eric Paulson, William Hall, Paul Knechtges, Beth Erickson, and X Allen Li. Auto-segmentation of pancreatic tumor in multi-parametric MRI using deep convolutional neural networks. *Radiotherapy and Oncology*, 145:193–200, 2020.

[74] Igor Linkov, Stephanie Galaitsi, Benjamin D Trump, Jeffrey M Keisler, and Alexander Kott. Cybertrust: From explainable to actionable and interpretable artificial intelligence. *Computer*, 53(9):91–96, 2020.

[75] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[76] Baozhong Liu and Jianbin Liu. Overview of image denoising based on deep learning. In *Journal of Physics: Conference Series*, volume 1176, page 022010. IOP Publishing, 2019.

[77] Hui Liu, Qingyu Yin, and William Yang Wang. Towards explainable NLP: A generative explanation framework for text classification. *arXiv preprint arXiv:1811.00196*, 2018.

[78] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[79] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

[80] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013.

[81] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[82] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[83] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE international conference on computer vision*, pages 1–9, 2015.

[84] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.

[85] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

[86] Christian Meske and Enrico Bunde. Transparency and trust in Human-AI-Interaction: The role of model-agnostic explanations in computer vision-based decision support. In *International Conference on Human-Computer Interaction*, pages 54–69. Springer, 2020.

[87] Tim Miller, Piers Howe, and Liz Sonenberg. Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences. *arXiv preprint arXiv:1712.00547*, 2017.

[88] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[89] Reza Mohammadi, Iman Shokatian, Mohammad Salehi, Hossein Arabi, Isaac Shiri, and Habib Zaidi. Deep learning-based auto-segmentation of organs at risk in high-dose rate brachytherapy of cervical cancer. *Radiotherapy and Oncology*, 159:231–240, 2021.

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

[91] Morten Mørup. Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):24–40, 2011.

[92] Zohaib Mushtaq, Shun-Feng Su, and Quoc-Viet Tran. Spectral images based environmental sound classification using cnn with meaningful data augmentation. *Applied Acoustics*, 172:107581, 2021.

[93] Kamyar Nazeri Naeini. *Structure guided image restoration: a deep learning approach*. PhD thesis, 2019.

[94] Andrés Páez. The pragmatic turn in explainable artificial intelligence (XAI). *Minds and Machines*, 29(3):441–459, 2019.

[95] Iam Palatnik de Sousa, Marley Maria Bernardes Rebuzzi Vellasco, and Eduardo Costa da Silva. Local interpretable model-agnostic explanations for classification of lymph node metastases. *Sensors*, 19(13):2969, 2019.

[96] Frank Pasquale. *The black box society*. Harvard University Press, 2015.

[97] Dino Pedreschi, Fosca Giannotti, Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, and Franco Turini. Meaningful explanations of black box AI decision systems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9780–9784, 2019.

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

[99] Nina Poerner, Benjamin Roth, and Hinrich Schütze. Evaluating neural network explanation methods using hybrid documents and morphological agreement. *arXiv preprint arXiv:1801.06422*, 2018.

[100] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. *Advances in neural information processing systems*, 28, 2015.

[101] Mauricio Reyes, Raphael Meier, Sérgio Pereira, Carlos A Silva, Fried-Michael Dahlweid, Hendrik von Tengg-Kobligk, Ronald M Summers, and Roland Wiest. On the interpretability of artificial intelligence in radiology: challenges and opportunities. *Radiology: Artificial Intelligence*, 2(3):e190043, 2020.

[102] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should I trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[103] Pratyush Rokade and Kumar Raju Alluri BKSP. Building quantifiable system for XAI models. *Available at SSRN 4038039*.

[104] Cynthia Rudin. Algorithms for interpretable machine learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1519–1519, 2014.

[105] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[106] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[107] Makoto Sato and Hiroshi Tsukimoto. Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3, pages 1870–1875. IEEE, 2001.

[108] Ludwig Schallner, Johannes Rabold, Oliver Scholz, and Ute Schmid. Effect of superpixel aggregation on explanations in lime–a case study with biological data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 147–158. Springer, 2019.

[109] Kathryn Schutte, Olivier Moindrot, Paul Hérent, Jean-Baptiste Schiratti, and Simon Jégou. Using stylegan for visual interpretability of deep learning models on medical images. *arXiv preprint arXiv:2101.07563*, 2021.

[110] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[111] Sumeet S Shah and John W Sheppard. Evaluating explanations of convolutional neural network image classifications. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

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

[113] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[114] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):667–676, 2017.

[115] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. Evolving deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*, 24(2):394–407, 2019.

[116] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[117] Chunwei Tian, Yong Xu, Lunke Fei, Junqian Wang, Jie Wen, and Nan Luo. Enhanced CNN for image denoising. *CAAI Transactions on Intelligence Technology*, 4(1):17–23, 2019.

[118] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[119] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *2007 IEEE 23rd international conference on data engineering workshop*, pages 801–810. IEEE, 2007.

[120] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE transactions on neural networks and learning systems*, 32(11):4793–4813, 2020.

[121] Julian Tritscher, Markus Ring, Daniel Schlr, Lena Hettinger, and Andreas Hotho. Evaluation of post-hoc XAI approaches through synthetic tabular data. In *International symposium on methodologies for intelligent systems*, pages 422–430. Springer, 2020.

[122] B Veeling. The patchcamelyon (PCam) deep learning classification benchmark, 2019.

[123] B Veeling, BE Bejnordi, G Litjens, and JVD Laak. Histopathologic cancer detection, 2019.

[124] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pages 210–218. Springer, 2018.

[125] Giulia Vilone and Luca Longo. Explainable artificial intelligence: a systematic review. *arXiv preprint arXiv:2006.00093*, 2020.

[126] Giulia Vilone and Luca Longo. Classification of explainable artificial intelligence methods through their output formats. *Machine Learning and Knowledge Extraction*, 3(3):615–661, 2021.

[127] Giulia Vilone and Luca Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, 76:89–106, 2021.

[128] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[129] Ge Wang, Jong Chul Ye, and Bruno De Man. Deep learning for tomographic image reconstruction. *Nature Machine Intelligence*, 2(12):737–748, 2020.

[130] Weiquan Wang and Izak Benbasat. Recommendation agents for electronic commerce: Effects of explanation facilities on trusting beliefs. *Journal of Management Information Systems*, 23(4):217–246, 2007.

[131] Michael R Wick and William B Thompson. Reconstructive explanation: Explanation as complex problem solving. In *IJCAI*, pages 135–140, 1989.

[132] Sarah Wiegreffe and Ana Marasović. Teach me to explain: A review of datasets for explainable nlp. *arXiv preprint arXiv:2102.12060*, 2021.

[133] Chia-Hung Yeh, Min-Hui Lin, Po-Chao Chang, and Li-Wei Kang. Enhanced visual attention-guided deep neural networks for image classification. *IEEE Access*, 8:163447–163457, 2020.

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

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

[136] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, pages 457–473. Springer, 2016.

[137] Avraham Zlochower, Daniel S Chow, Peter Chang, Deepak Khatri, John A Boockvar, and Christopher G Filippi. Deep learning AI applications in the imaging of glioma. *Topics in Magnetic Resonance Imaging*, 29(2):115–00, 2020.