

---

SAARLAND UNIVERSITY

Faculty of Mathematics and Computer Science  
Department of Computer Science  
MASTER THESIS

---



# Improve CNN Performance Using User Feedback

submitted by  
Md Abdul Kadir  
Saarbrücken  
January 2022

---



**Advisor: Dr. Fabrizio Nunnari**

Cognitive Assistants  
DFKI GmbH  
Stuhlsatzenhausweg 3  
Campus D3.2  
66123 Saarbrücken  
Germany

**Reviewer 1: Prof. Dr. Antonio Krüger**

DFKI GmbH  
Stuhlsatzenhausweg 3  
Campus D3.2  
66123 Saarbrücken  
Germany

**Reviewer 2: Prof. Dr. Daniel Sonntag**

DFKI GmbH  
Stuhlsatzenhausweg 3  
Campus D3.2  
66123 Saarbrücken  
Germany

Applied Artificial Intelligence  
University of Oldenburg  
Marie-Curie Str. 1 D-26129 Oldenburg  
Germany

**Submitted**

28<sup>th</sup> January 2022

Saarland University  
Faculty MI – Mathematics and Computer Science  
Department of Computer Science  
Campus - Building E1.1  
66123 Saarbrücken  
Germany

# Declarations

## Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

## Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

## Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

## Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, \_\_\_\_\_  
(Datum/Date)

\_\_\_\_\_  
(Unterschrift/Signature)



### **Erklärung**

Ich erkläre hiermit, dass die vorliegende Arbeit mit der elektronischen Version übereinstimmt.

### **Statement**

I hereby confirm the congruence of the contents of the printed data and the electronic version of the thesis.

Saarbrücken, 28.01.2022  
(Datum/Date)

  
\_\_\_\_\_  
(Unterschrift / Signature)



## Acknowledgments

Foremost, I would like to convey my heartfelt appreciation to my advisor Dr. Fabrizio Nunnari for his ongoing support of my Master's research for his patience, encouragement, and spirit. His advice assisted me in the research and writing of the thesis. I could not have envisioned having a better advisor and mentor for my Master thesis.

Besides my advisor, I would like to thank Prof. Dr. Daniel Sonntag for giving me guidelines and inspiration on the human-in-the-loop AI paradigm.

Furthermore, I would like to thank Prof. Antonio Krüger for his precious feedback on the Master's thesis seminar.

I would also like to thank my friend Omair Shahzad Bhatti and Chirag Bhuvaneshwara for continuously motivating me.

Last but not least, I would like to thank my wife, Fatema Tuj Johura Tonny, and my parents, Md Abdul Hasim and Ruhena Akter, for their continuous support.

## Abstract

A convolutional neural network (CNN) is a special kind of neural network that got massive attention in image classification and computer vision tasks. Nowadays, CNN is deployed in real-life applications, for example, autonomous driving and disease classification, although it is a black-box algorithm. One can train a CNN on a large amount of skin cancer images and deploy it in real-life applications. Dermatologists can use the classifier as a decision support system (DSS). However, its acceptance is constantly challenged by critique when it is involved in making sensitive and risky decisions. Due to the black-box nature of CNN, it is difficult to identify the reasoning behind CNN's decision. Similarly, the neural decision mainly depends on training on a large amount of data and does not allow user interaction in the post-deployment phase. CNN models are limited to making decisions only; they are not built to receive feedback from users. One popular application of CNN is skin cancer classification. To overcome this limitation, we propose user feedback-based fine-tuning. Our proposed method will let users give feedback on CNN's output. This thesis investigates the effect of providing feedback on CNN's visual explanation and classification in the skin lesion classification task and observes how the algorithm reacts to the user intervention. More specifically, the objective of the thesis is to visualize the reason for skin lesion classification, and if the reasons are not correct, use user feedback to tune the CNN. We propose a novel CNN architecture that integrates the Grad-CAM technique for explaining the model's decision in the training phase. We observe that fine-tuning our model using simulated user feedback on explanation and classification improves the model's performance in providing visual explanation while retaining classification accuracy.

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Machine Learning and Neural networks . . . . .	2
1.3	Evolution of ML methods . . . . .	3
1.4	Convolutional neural network . . . . .	5
1.5	Online machine learning . . . . .	7
1.6	Interactive learning . . . . .	8
1.7	Research goals and outline . . . . .	8
<b>2</b>	<b>Related work</b>	<b>10</b>
2.1	Explanation methods . . . . .	10
2.1.1	Input augmentation . . . . .	10
2.1.2	Class activation mapping . . . . .	11
2.2	Self-explaining neural networks . . . . .	13
2.3	Interactive and explainable AI . . . . .	13
2.4	Domain specific related work: skin cancer detection . . . . .	17
<b>3</b>	<b>Technical background</b>	<b>21</b>
3.1	Transfer learning . . . . .	21
3.2	Base Architecture . . . . .	22
3.2.1	Training data . . . . .	22
3.3	Saliency map . . . . .	25
3.4	Thresholding saliency map . . . . .	26
3.5	Loss function . . . . .	27
3.6	Online and incremental machine learning . . . . .	28
3.7	Optimization technique . . . . .	28
<b>4</b>	<b>Method</b>	<b>30</b>
4.1	Self explainable model . . . . .	30
4.2	Integrating the explanation with the original model . . . . .	32
4.3	User feedback . . . . .	34
4.4	Implementation . . . . .	34

<b>5 Experiments and Results</b>	<b>36</b>
5.1 Data description . . . . .	36
5.2 Data preparation . . . . .	36
5.3 Simulating feedback on the full simulation set . . . . .	38
5.4 Simulation on slices of data . . . . .	39
5.5 Training on different loss functions . . . . .	40
5.6 Experiment using unbalanced data . . . . .	40
5.7 Experiment using reduced balanced data . . . . .	41
5.8 Experiment using upsampled balanced data . . . . .	44
5.9 Result of sliced simulation . . . . .	46
5.10 Training performance . . . . .	48
 <b>6 Conclusion</b>	 <b>49</b>
6.1 Summary . . . . .	49
6.2 Evaluation . . . . .	50
6.3 Future work . . . . .	50
 <b>Bibliography</b>	 <b>52</b>

---

# Chapter 1

## Introduction

### 1.1 Motivation

A Convolutional Neural Network (CNN) [68] can detect malignant skin lesions; however, it cannot produce the correct explanation behind a prediction all the time. As a result, the acceptance of such algorithms in the medical domain is quite rare. In image classification, there are several ways to explain a prediction. Nonetheless, in some cases, the explanations can be misleading, and the network does not provide the flexibility of learning from a given correction on the misleading explanation. On the one hand, if we put human decision-makers in that situation, they can learn from feedback. For example, let us compare the decision-making approach of humans and neural networks. We see that if humans make a wrong decision and explanation, they can perceive the reason for the mistake and can take necessary action to avoid the same kind of mistake in the future, given that there is an availability of correct feedback. On the other hand, CNN can only give predictions and explanations but can not learn from feedback if the prediction or the explanation is incorrect.

These limitations of neural networks fall under the categories of lack of interactivity. As the domain of the experiment is image classification, here we focus only on the deep convolutional neural network because it performs better than a fully connected neural network in image classification [45]. The deep CNNs are deep in terms of layers. Each of the layers contains many convolution kernels. They identify different features of an image, and based on the detection of features; the neural networks classify an object on the image. Nonetheless, the high performance of CNN, the explanations of classifications are also required in the application domain. Researchers introduced visual explanation techniques to introduce explainability in deep learning-based image classification. They do nothing but visualization of the discriminatory regions of an image based on specific class identity [90, 73, 114]. Highlighting class discriminative regions in an image is an example of explainability. These approaches show a clear path to explainability in image classification. However, what if the visual explanation wrongly identifies the location of interest regardless of the classification result. How can we fix this problem? Imagine

that in the field of dermatology, if we show a wrong visual explanation of a skin lesion classifier to a dermatologist, s/he could probably identify it and provide feedback. We can leverage the feedback to update the classifier.

## 1.2 Machine Learning and Neural networks

CNN has an enormous impact on computer vision and image processing. Much research in computer vision and image processing has reached a milestone due to the advancement of CNN architectures. Many CNN models can classify objects as accurately as a human [48]. In the past, image and video processing were critical tasks. They required a lot of mathematical techniques to identify features in frames. Due to the advancement of CNN, one can now easily apply a pre-trained CNN model in a variety of image and video data. From home to large industrial, everywhere there are potentials for CNN. Home security cameras now can use CNN to detect the known person in a house. Cars use a set of cameras to detect objects nearby for safety. Self-driving cars use CNN to detect and map roads. Detecting the number plate of cars is also an application. Smartphones use CNN to detect the owners' faces to unlock. Additionally, we can use CNN in industrial devices. Anomaly and fault detection in products is one example. We can also use it to measure the stress levels of industrial workers. It will increase productivity. Besides image data, we can apply it to text and time-series data. CNN can improve disease diagnosis. As we know, a large part of medical devices uses screening techniques, and doctors identify diseases based on the screening image. Doctors can feed MRI, X-ray, skin lesion data to CNN to find helpful information. It will reduce the medical cost by minimizing the required time for diagnosing patients. Also, if a CNN can diagnose an acute disease, patients with less acute diseases will not have to go to specialists or go for expensive tests in the first place. As a result, it will save time and reduce the cost of high-priced tests. A CNN can detect skin cancer by looking at images of skin lesions. Detecting malignancy of skin lesions using a neural network is a cost-effective and handy alternative.

Machine learning (ML) is a subset of data-driven computer science, statistics, and artificial intelligence. It deals with the algorithms that utilize data to infer decisions and predictions. The general purpose of ML algorithms is to infer information from data. The task of an ML algorithm can be classification, object detection, or cluster finding. We can divide ML algorithms into three main categories based on the task and data. Supervised, unsupervised, and semi-supervised are these three categories. The supervised algorithm deals with a task where ground truth is known. For example, we need a supervised ML algorithm for object classification tasks. The requirement for training supervised ML algorithms is ground truth data. By iteratively looking at ground truth, supervised algorithms learn to do classification. This iterative approach is also known as training. For example, in plant diseases prediction using a picture of a plant leaf, the disease name is the ground truth, and the leaf's picture is the feature. The feature is the information from which an ML algorithm infers a decision.

Differently, an unsupervised algorithm deals with data where ground truth information is unknown. Localizing clusters in a dataset is an example of an unsupervised algorithm. The unsupervised algorithm looks at the features in the data and identifies boundaries between different clusters based on the pattern of the data. Moreover, a semi-supervised algorithm falls between supervised and unsupervised algorithms. In reality, there is a limited amount of ground truth data. Many datasets have missing ground truth. In such a scenario, we can utilize a semi-supervised algorithm. It learns from a small amount of ground truth data and a vast amount of feature information. This thesis falls under the



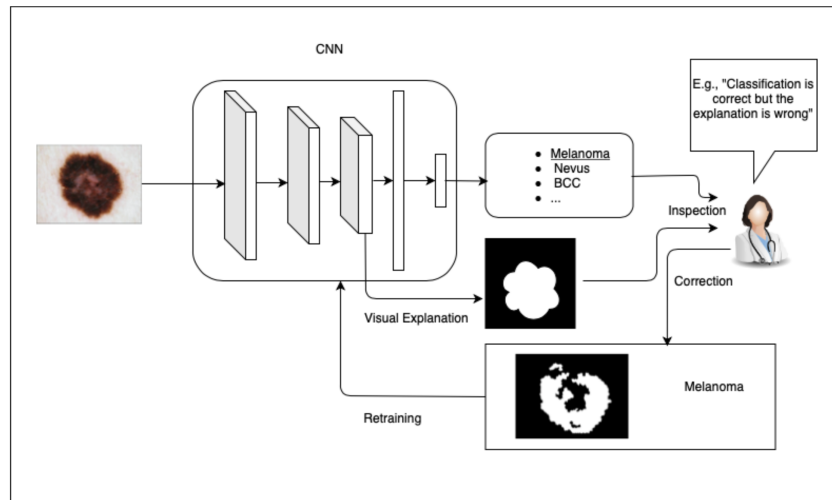


Figure 1.1: This figure gives an overview of a hypothetical human-in-the-loop skin cancer classification and explanation system.

hood of supervised machine learning.

Nowadays, the IT industry uses machine learning algorithms for business purposes. The health sector is also looking forward to employing machine learning algorithms. Initially, ML algorithms were rudimentary and explainable. However, over time researchers are introducing more complicated algorithms. As a result, ML algorithms are becoming less interpretable. For example, when an ML algorithm infers a decision, it comes from mathematical reasoning. The ability to present this mathematical reasoning to a user is explainability. Less complex ML algorithms are mathematically explainable. On the other hand, more complex algorithms are less explainable because it is hard to identify how they model input and output relations in specific situations. However, explainability is the far-reaching attribute of an algorithmic model.

Besides this, models must ensure one more attribute; interactivity, for real-life applications. The interactivity of a model is the ability to provide and accept information from the outside world. Generally, the ML algorithms lack interactivity because they are not designed to get information from the outside world. Introducing interactivity is one of the goals of this thesis. Figure 1.1 shows an interactive system where an image CNN classifier classifies skin cancer images and explains them in front of a dermatologist. The dermatologist uses the system as an assistive tool. However, if the result is not convincing, s/he can provide feedback to the system.

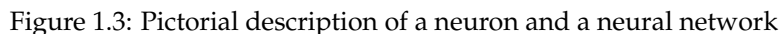
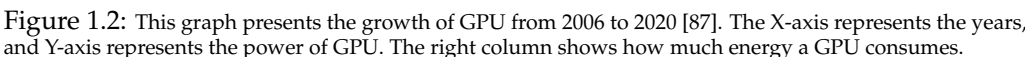
### 1.3 Evolution of ML methods

The history shows that the least square linear regression was the first machine learning algorithm that finds a rough linear fit to a set of point. Legendre and Gauss used it for predicting planetary movement in 1805 and 1808. Later, in the 1830s and 1840s, Verhulst invented the logistic function under the guidance of Quetelet. It is the core of logistic regression. It is predominant for classification. Similarly, Fisher proposed LDA (Linear Discriminant Analysis) for classification for categorical variables in 1936. In 1951 Fix

and Hodges introduced a non-parametric classifier, k-NN (K-nearest Neighborhood). For linear classification, Rosenblatt proposed perceptron in 1958. It is the elementary artificial neural network (NN) model. Initially, these algorithms were explainable, but later more complex machine learning algorithms came. In 1992 Boser and colleagues proposed SVM (Support Vector Machine) for classification and regression. A few years later, the research community adapted the algorithm. It showed very high accuracy in low-resolution image classification. It finds the separation line between two classes in a dataset. It uses kernels to transform features to a linear plane from nonlinear data. Linear SVM models are interpretable like other linear model. However, when we use nonlinear kernels, they lose interpretability.

Although the idea of artificial neural networks is old, the application has become popular at the end of the twentieth century. A neural network, also known as an artificial neural network, is a set of artificial neurons connected in various fashions. An artificial neuron mimics the neuron of the brain. The mathematical implementation does nothing but calculate the inner product between inputs and weights and passes through nonlinear activation to generate output. Weights are just numbers that give priority to each input. They are learned through training. Nonlinear activation is just a transformation function for the inner product. Figure 1.3 illustrates the functionality of a neuron and a neural network. Figure 1.3a is a neuron with two inputs,  $a_1$  and  $a_2$ . These two inputs get multiplied with weights and then pass through the activation to generate output. Activation is essential for neural networks to introduce nonlinearity. Neural networks are unable to find nonlinear separation lines between classes without nonlinearity. In reality, features of different classes of objects are only nonlinearly separable. So neural networks must ensure nonlinear separation capability. ReLU, Sigmoid, Tanh, and other activation functions are only a few examples. ReLU is one of the commonly used activation functions. Figure 1.3b is a small neural network with four inputs and one output. Depending on the task, the number of inputs and output can be arbitrary. The layers between the input and the output layers are called hidden layers. In this picture, we see two hidden layers. The input and the hidden layers aim to change the feature from one hyperspace to another. The output layer categorizes or forecasts the results.

A convolution neural network (CNN) utilizes convolution operation instead of an inner product in the layers. The convolution operation has been a familiar term among the signal processing community for a long time. However, it created an enormous impact in deep learning after 2010. A convolution is simply moving the inner product of the signal with a kernel. A kernel can be a vector or a matrix. A convolution operation helps to reduce the number of weights in a neural network. Images have a large number of features where each pixel is a feature. Because of the large number of pixels in an image, a NN requires a vast number of weights. As a result, learning the value of these weights takes a very long training time. Contrary, a CNN make it simple by replacing all the weights with the kernel. For example, a CNN kernel consists of fewer weights. It takes less time than NN during training. Figure 1.4 shows a simple convolution operation. A cascade of a convolutional neural network and a fully connected neural network shows good performance. We also interpret it a CNN. Next section, we will discuss more on CNN. Moreover, recurrent neural network (RNN) is different from the other two networks. It passes temporal information in the network and uses the temporal connection between nodes. As it is out of the scope of this research, we do not go into discussing details. In the previous paragraphs, we talked about CNN, and we learned that deeper CNN is necessary for better performance. However, a deeper model means there are millions of learnable parameters. Training such a model requires a high computation power. Also, classifying high-resolution images requires a



## 1.4 Convolutional neural network

A convolutional neural network is a special kind of neural network that uses convolution operations to extract features from data and, predominantly, deals with image data. Convolution kernel is the main ingredient to extract features from images. Kernels are nothing but weights of the CNN. We already discussed that neural network models learn weight through training on a large amount of data. For CNN, this statement also holds. One can train a CNN model on a large amount of image data. It iteratively updates the kernel values during training to find an optimal value. Precisely, training a CNN is to find kernels that can detect specific features from images. Traditionally, researchers used

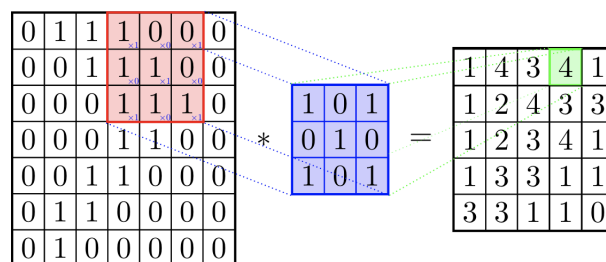


Figure 1.4: A convolution operation of 2D image with a  $3 \times 3$  kernel

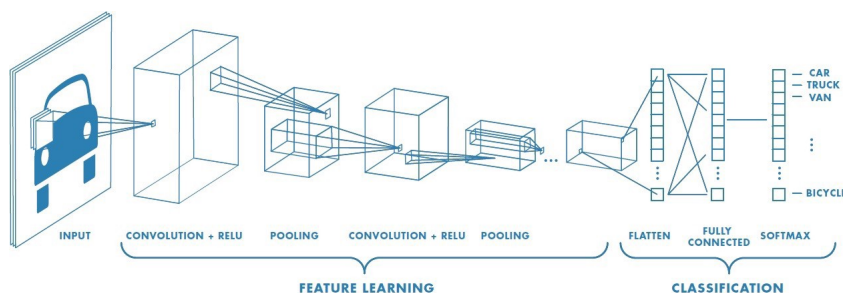


Figure 1.5: A convolutional neural network (CNN) block diagram by Saha [86]

predetermined kernels for feature detection from images. These predetermined kernels are also known as filters. There are several kinds of filters for detecting features from images. The Sobel operator [107] is one example of such a filter. In the past, researchers employ filters for attribute detection from an image. Then, they passed these features into an ML model (i.e., SVM) to detect objects on the image. It has a limitation; for example, it can only identify known features. A CNN can overcome this limitation by learning unknown attribute detection kernels. There are several layers in a CNN. The purpose of each layer is to detect different kinds of features from images. A CNN can have multiple fully connected neural network layers before the output layer. For example, figure 1.5 shows the architecture of CNN. There are three main parts in figure 1.5, convolution layer, fully connected layer, and output layer. The purpose of convolution layers is to detect task-specific features. Contrary, fully connected layers transform the features into different hyperspace, and the output layers classify objects in the image using that transformed feature. Besides the layers mentioned above, it has pooling and dropout layers. The purpose of the pooling layer is to squeeze the information, and the dropout layer randomly drops some information to reduce the training time and overfitting.

We have already discussed the convolutional neural network. Most of these architectures deal with classification tasks. In this section, we will discuss "Very Deep Convolutional Networks for Large-scale Image Recognition (VGG16)" [94] as it is the core architecture used in this thesis. Before it, many architectures were published [45, 113, 91, 94]. The general differences between VGG16 and the former architectures are that in VGG16, kernel dimension decreased, and the depth of the architecture increased. However, we must thank the original AlexNet [45] for showing the right direction in the very beginning. After VGG16, many architectures came with higher accuracy [94, 98, 80]. They are more profound in the number of layers than the VGG16. We could choose one of this contemporary architecture, but according to Nunnari et al. [67], VGG16 shows great success in generating saliency maps.

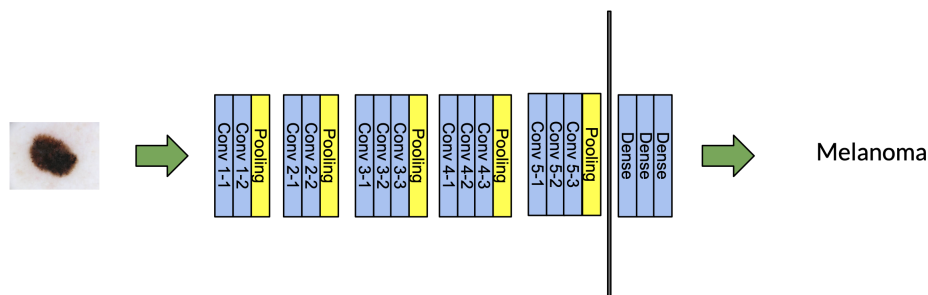


Figure 1.6: Layout of VGG16 detecting skin cancer from dermoscopic image.

Let us get introduced to VGG16. It is initially designed for image size  $224 \times 224$ . The images pass through a stack of convolution layers of size  $3 \times 3$ . The stride of convolution is one pixel. Five max-pooling layers are applied after each convolution segment, performing spatial pooling (not all the convolution layers are followed by max-pooling). With stride 2, max-pooling is done across a  $2 \times 2$  pixel frame. Three Fully-Connected (FC) layers follow a stack of convolutional layers (which have varying depths in different topologies). The first two fully connected layers have 4096 neurons, and the output softmax contains 1000 channels for 1000 classes. Figure 1.6 shows the arrangement of convolution, pooling, and fully connected layers. A fully connected layer is also known as a dense layer. This thesis employs a modified version of this architecture. In the implementation part, we discuss more on the modification.

The idea of CNN is not very old. There is plenty of research on the application of CNN in different aspects of life. In the motivation section, we will look at some of the applications and limitations of current CNN models.

## 1.5 Online machine learning

Another aspect of machine learning is online learning. Traditional machine learning learns from a big chunk of data, but online learning algorithms learn from a stream of data and update knowledge continuously. On the other hand, an offline learning algorithm requires retraining the model on new data for knowledge updates. Naturally, data is not abundant, so it is not convenient to collect a plethora of data at one time. As a result, the offline learning algorithms are not robust. Online learning algorithms overcome this limitation by training sequentially. When new training data enters, an online learner can promptly and efficiently update the model, which solves the shortcomings of traditional batch learning. For supervised learning tasks, online learning algorithms can be developed. One of the most prevalent tasks is classification, which aims to predict the categories to which a new data instance belongs previous training data examples with labeled categories. In online supervised learning, a learner needs complete feedback information. Many online learning problems can be expressed as an online convex optimization problem that can be solved using the online gradient descent OGD algorithm [37].

## 1.6 Interactive learning

Human-in-the-loop (HITL) machine learning is a potential subfield of machine learning. A typical ML algorithm requires a large amount of data to train a model. However, we know that the supervised data is limited. It is the biggest challenge for the ML algorithm to advance. Similar to transfer learning, researchers explore HITL machine learning to overcome data limitations [108]. Human-in-the-loop machine learning is also known as interactive learning. According to Monarch [61], HITL machine learning is a strategy that combines human and machine knowledge to ensure one or more of these criteria.

1. Improve a machine learning model's accuracy.
2. Faster attainment of a machine learning model's target accuracy.
3. To improve accuracy, combine human and machine intelligence.
4. To improve productivity, use machine learning to assist human jobs.

Other than overcoming data limitations, human-in-the-loop (HITL) machine learning benefits the users in several ways. Most ML models present results to users, which is not a very expressive way of helping humans. Humans seek the explanation and reasoning behind every effect. Interactive machine learning overcomes this limitation. Similarly, human likes to see how a system learns after giving feedback. Updating initial decision based on feedback convinces humans how the system is learning. This kind of interaction is possible in HITL machine learning. Using the HITL approach, one can develop a user-friendly algorithm.

## 1.7 Research goals and outline

CNN does not learn from feedback, as discussed in the last part. This section explains how user feedback can be incorporated into CNN models (Figure 1.1). This thesis aims to investigate a way for incorporating user feedback into CNN training. We will look at how user feedback keep the model up to date with new information. We are primarily concerned with post-deployment training. A CNN model is trained on a vast amount of data initially. It is deployed to conduct various activities following the evaluation process, such as performance testing. A CNN model can be retrained on new data regularly once deployed. However, after deployment, a model can make a wrong decision, and the user can understand it by looking at the explanation. Figure 1.7 shows classifications and explanations of a dog and husky classifier. Although classification performance is good, the model classifies based on unrelated features. For example, if we look at the explanation of the classifier, we see that model is looking at snow and classifies it as a wolf. It concludes that the feature in the image model supposed to consider is not considered.

There are four possible situations in the post-deployment phase of a model.

1. Classification and explanation both are correct.
2. Classification is correct, but the explanation is incorrect.
3. Classification and explanation both are incorrect.

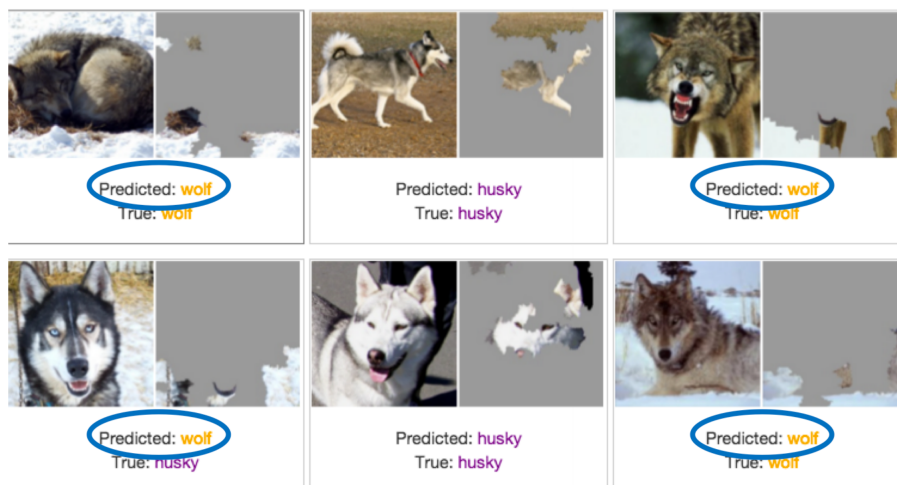


Figure 1.7: An example of right classification for wrong reason. [81]

#### 4. Classification is incorrect, but the explanation is correct.

For the first situation, we do not have to deal with anything. We keep the model as it is. In the second possible situation, we must modify the model knowledge. The model is doing the right classification, but generating the wrong explanation. Roughly speaking, this kind of classification does not hold a value because the model is looking at the wrong perspective. To solve this problem, we need to retrain the model by giving correct feedback. The third scenario, model is classifying something that is not the task of the model. The model learned to recognize features that were not at all similar to the task's features. For this case, we need to give information on the correct feature for updating its knowledge. When the explanation is correct but the classification is wrong, the model is not understanding the feature. For this case, we need more data to classify features correctly. One way to solve these three problems is to retrain the model using user feedback. However, CNN only retrains based on the only class label. They are not originally built to learn from explanation because explanation data is very rare.

Here, we are exploring a way to update the model's knowledge based on user feedback. User feedback can be of different types. We considered only classification and explanation feedback. We modify the loss function of the original model and integrate both of the feedbacks as ground truth when the model makes a wrong decision. For the simplicity of the simulation, we feed the user feedback to the model regardless of the model's decision because the model only learns when it makes a wrong decision. In a practical application, users can draw the contour around the critical region of skin lesion using interactive tools to generate feedback. Firstly, this thesis presents a way to modify a pre-trained model to introduce explainability (e.g., self-explainable model). Secondly, we integrate explanation loss with the original classification loss. In this modification, we ensured that the pre-trained model performance sustains. Thirdly, we evaluate the performance on classification, explanation. This evaluation shows us the effect of integrating explanation in the loss function.

---

## Chapter 2

### Related work

This section presents the researches related to explainability and interactivity in machine learning and the deep learning domain. At the beginning phase of the development, machine learning and deep learning algorithms were black-box models except for linear and tree-based models. Black-box models only predict, but they do not explain the prediction. It is the main reason behind the slow growth in the deployment of this kind of algorithms. Many researchers tried and are still trying to find out ways to explain the decision of ML/DL algorithms. There are two ways to explain them, explaining the model itself, or explaining the reason for a specific prediction. They are known as global and local methods. In this thesis, we will only focus on the local methods because the thesis aims to improve the explanation of a sample-based prediction. First, we will cover the different explanation generation techniques of CNN-based and typical ML-based image classifiers. After that, we will explain the existing interactive method where users provide feedback to NNs/AI algorithms to improve accuracy. We will also present the limitation of these methods.

#### 2.1 Explanation methods

There are generally two ways to locally explain a neural network's decision. The first way includes augmenting attributes, and the second way involves calculating the class activation map (CAM) for a prediction. Here, we will explain the research related to the augmentation-based explanation technique and the CAM technique.

##### 2.1.1 Input augmentation

Ribeiro et al. [81] propose LIME, which can explain the prediction of any black-box classifier. In elaborate form, LIME is Local Interpretable Model-agnostic Explanations. They use textual or visual representations of reasons of classifiers' decision by explaining a prediction. To explain a prediction, the LIME first creates an augmented dataset from the original data and predicts what is in the original data. Then, it calculates the distance between the original data and the augmented data. To explain a black-box model, it feeds



the augmented dataset to the model and observes the output. It only selects  $m$  number of augmented features based on the likelihood of the most probable output. Here, the most probable output is the prediction for the original feature. The authors say that the distance between the top  $m$  augmented features and the original feature represents the importance of the top  $m$  features for the prediction. To see how accurate an explanation is, they use two explainable models (Sparse logistic regression and decision tree) and train on a toy data set with a maximum of 10 essential features (gold features). During the test, they calculate the fraction of the ten golden features present in the explanation of LIME. The more significant the fraction, the higher the explainability of LIME. It only focuses on generating an explanation but does not give a solution if it is wrong.

Similarly, RISE [73] is another technique that utilizes the feature augmentation method to explain black-box classifiers. It identifies the class-specific essential regions of an input image by predicting a masked version of the input. The mask is random, not the choice. They calculate the confidence score of the target class for each masked version and then do a linear combination of the confidence scores with the masks. This linear combination defines the importance of specific image pixels during the classification. The value of pixels determines the crucial pixels in the explanation (saliency map). RISE can be used with any classification algorithm for generating explanations. The authors of RISE also propose an automatic evaluation metric to check the quality of an explanation. By removing the pixel from the image based on an explanation, they can determine which image areas explain the image better. This metric requires the insertion and deletion of essential pixels in an image for a specific classification task. They assume the AUC (Area under the curve) for classifying an image is a function of the importance of a pixel in the image. Removing necessary pixels reduces the AUC; meanwhile, introducing important pixels increases AUC.

Lundberg and Lee [57] propose SHAP (SHapley Additive exPlanation) for explaining prediction. SHAP assigns an 'importance' parameter to each input feature for a specific prediction. SHAP unifies six existing explanation methods, LIME [81], DeepLIFT [93], Layer-Wise Relevance Propagation [6], Shapley regression values [53], Shapley sampling values, and Quantitative input influence. The reason for unification is that all the methods mentioned above use a linear representation of a complex model for generating explanation. According to the authors, the methods mentioned above do not always comply with three important properties of estimating SHapley [106] values in game theory: local accuracy, missingness, and consistency. The three properties are very desirable for assigning the importance of features during classification. So, they propose one unique solution that complies with the three properties. Although finding a unique solution is an NP-hard problem, the authors propose some surrogate functions for finding the unique solution. This unique solution is called SHAP. SHAP can explain any machine learning model without knowing the internal structure. However, It can be a little slower for some models than others.

### 2.1.2 Class activation mapping

Class activation technique generally works in CNNs. In this technique, the discriminatory classification features are extracted from the activation of any convolutional layer.

Generating class activation map (CAM) is a technique for localizing class-specific significant features used in explaining convolutional neural networks. CAM has a remarkable localization capability. Zhou et al. [114] describe a procedure for generating CAM using global average pooling (GAP) on convolution layers. GAP is the weighted sum of the

convolutional feature map. In some CNN, a neural decision is made from the weighted sum of the GAP outputs. According to the authors, we can spot important image regions by linearly combining the weights of the output layer with the activation of the last convolution layer. CAM can be used to localize objects during classification. They evaluate the localization performance of CAM on weakly-supervised object localization on the ILSVRC [84] benchmark. They found that CAM can do accurate object localization. Although these methods perform well, there are some limitations. CAM only works on fully connected CNN. It does not work if there are fully connected layers before the output softmax layer. Most of the high-performance CNN model has more than one fully connected layer [66, 94, 35].

Selvaraju et al. [90] propose an explanation technique known as Grad-CAM, which is an extended version of CAM. It utilizes any target class flow gradients through the final convolution layer. The gradients of the target class in the last convolution layer represent and localize the important regions or features for identifying the target class. They extract the gradient in the last convolution layer because the spatial gradient information vanishes in the fully connected layer. The authors assume that the last convolution layer extracts semantic class-specific information necessary for a specific class localization. The Grad-CAM method splits into two steps. The first step calculates the activation of the last convolution layer. This activation has information about all classes. The second step removes information about other classes and only keeps the information about the class with maximum probability from the activation output. To do so, it calculates the weight of each activation based on the gradient of the most probable class. Using a linear combination of the activation and the weights identifies necessary pixels of an image in the same resolution of the last layer for a classification. Later, the resolution increases to the original input image size of the network for better presentation. The resultant output is known as a saliency map. The authors also test the method using weakly-supervised localization. In this method, they binarize the saliency map by thresholding at 15% of the maximum value of each pixel and compare the most significant connected segment with the bounding box ground truth of the object by calculating the intersection over the union. We see Grad-CAM generate a local explanation, but it does not provide a way to fix a CNN when the explanation is wrong.

Grad-CAM++ [12] is an extended version of Grad-CAM. In "Improved Visual Explanations for Deep Convolutional Networks", Chattopadhyay et al. [12] propose this generalized method. It can produce improved visualization behavior of CNNs' predictions and performs better in visualizing multiple instances of an object during a classification. It produces an explanation similar to how Grad-CAM produces an explanation, but the only difference is that it only considers the positive gradient of the output class. We described the two steps for producing a saliency map in Grad-CAM in the previous section. In Grad-CAM++, the first step is the same to calculate convolution activation. However, in the second step, it calculates the gradient of the output class concerning the last convolution layer, passes it to a ReLU [1] function, and multiplies it with a weight-coefficient to calculate the weights. Weight-coefficient calculated from the inverse of the sum of the output of the ReLU. Finally, multiplying the activation with the weight produces an explanation (a saliency map).

There are other works related to the CAM, SmoothGrad [95] and Smooth Grad-CAM++ [69]. These methods are similar to previously mentioned two mentioned, but the only difference is that these methods add Gaussian noise with the input image. Introducing noise in these methods reduces the noise in the saliency map. Barata et al. [5] proposed a hierarchical CNN-LSTM attention model that uses hierarchical information about classes and then produces attention mapping and hierarchical classification results. The

attention map hints at how the classification algorithm looks at the objects in an image.

## 2.2 Self-explaining neural networks

Alvarez-Melis and Jaakkola [2] suggest that there are three prerequisites for ensuring the correct explanations from neural networks. The requirements are explicitness, faithfulness, and stability. The authors proclaim that the existing interpretable models do not assure the specifications. They advocate a self-explaining neural network (NN) framework and scrutinize its performance to validate the prerequisites to overcome these limitations. According to the authors, a linear model is self-explainable. If someone can generalize the coefficients of a complex model to a linear model, the complex model becomes a self-explainable one. This generalization is possible because the parameters of a complex model can be a function of input features. They propose a learnable encoding function that encodes the raw input feature to human-understandable atomic feature units. This encoding helps the network to produce explanations. Furthermore, they introduce a gradient-regularized penalty with the objective function to ensure its correctness. Moreover, to establish the robustness of explanations, they add a regularizer. They also evaluate the performance of the framework in four datasets, MNIST digit recognition [46], benchmark UCI [25], Propublica’s COMPAS Recidivism Risk Score [77], and CIFER10 [44]. The classification test gives 99.11% test accuracy in MNIST, 82.02% in COMPAS, and 78.56% in CIFER10.

The second test compares the visual explanation with other frameworks, e.g., LIME [81], SHAP [57], OCCLUSION [113], and GRAD×INPUT [93], and so on to check the explicitness. They qualitatively prove their model performs better than others. The third experiment is indirect due to the deficiency of ground-truth explanations. It analyzes the effect of removing input features on the prediction. In comparison with SHAP and LIME, their model outperforms. In the robustness test, they observe the explanation change while adding noise with the features. Using a local stability function, they calculate the distance between the original and the noisy version and find that model’s performance exceeds the other state-of-art models. This framework fulfills the prerequisites. However, the additional encoder requires extra memory space to store, and training the encoder is an additional load in a GPU. The network is not interactive. So, we might not achieve the user’s faith on the model’s decision.

## 2.3 Interactive and explainable AI

Teso and Kersting [100] argue that interactive learning places the user into the loop, but the learner stays as a black-box for the user. Here the user is an agent (e.g., human) who can interact with the learner (e.g., a machine learning model). They also suggest a novel explanatory interactive learning (XIL) framework that can overcome the limitation of interactive learning. Moreover, it can help the user gain trust in the learner by introducing completeness, directability, and understandability. In XIL, a user gives feedback to the learner’s output in an active learning manner when required. The proposed framework utilize LIME [78] as a local explainer and an additional component. They call the framework Caipirinhas (CAIPI). They use three functionalities, labeling the unlabeled data using user input, fitting the model on labeled and unlabeled features, and explaining a prediction using the local explainer. By including an additional component known as **ToCounterExamples**, authors allow CAIPI to learn from the user’s feedback on the

label and the explanation. **ToCounterExamples** generate counterexamples from explanation correction to teach the network to avoid irrelevant features. Counterexamples are nothing but original input images with randomized irrelevant regions. There are three scenarios during the interaction between the learner and the user: the prediction and the explanation are correct, both are wrong, or only the label is correct. CAIPI focuses mainly on the last. It trains itself from the user's feedback. It converts it into counterexamples and then retrains the LIME again.

The authors carry out four experiments. The first one is to check how the user's trust increases/decreases in the model due to an explanation, and the other three evaluate model performance. They check the impact of three different kinds of interactions between the CAIPI and the user in the first experiment. In the first scenario (S1), they only allow the test subject to see the label from CAIPI. In the second (S2), the test subject can also see the explanation. The third (S3) is the same as S2, but the only difference is that S2 converges to the correct rule, but S3 does not. In all of these scenarios, the user can only give feedback on the label of CAIPI's output. In each case, the test subject answers three-question whether CAIPI eventually learned to classify images correctly, the correct classification rule and whether test subjects prefer to see further performance of CAIPI. After the evaluation, the authors see that the trust/distrust of the user increase based on the interactions. In the second experiment, the authors find that CAIPI is robust to the input noise when it learns from explanation feedback. The base accuracy for noisy input is 48% when there is no correction. It increases to 82% when there is at least one counterexample. In the case of more of that, it outperforms the input-gradient (IG) [83]. The last two experiments are in the textual domain. Here the framework also performs steadily better in terms of explanation quality. CAIPI's performance increases due to the feedback explanation. However, the counterexample requires more disk space to store and GPU performance to retrain the model.

The main difference between CAIPI and this thesis is how we train the network on feedback. CAIPI augments the original training data based on user feedback and then fine-tune or train a new model. There is no change in the objective function of the model. Also, storing original training data is necessary for retraining. However, the approach we follow in this thesis only requires post-deployment test images and feedback. Initial training data is unnecessary as the model already learned from the original training data.

In "The Skincare project, an interactive deep learning system for differential diagnosis of malignant skin lesions." Sonntag et al. [96] describe the functionality and interface of an interactive decision support system for differential diagnosis of malignant skin lesions. They used the baseline deep learning system that trained on ISIC [14] data. They implement a web interface for introducing interactivity in understanding the explanation of neural classification. It uses U-Net [82] to extract the location, shape, and features of the lesions in an input image and overlap them with the input image to show the significant part of skin lesions. They show that based on the shape of features of a lesion, an expert can easily compare them with ABCD [33] rules and find a conclusion. The methods in the report give some generic ideas for interactive machine learning; however, there are some limitations too. They use two different architectures for classification and feature extraction. The classification architecture (VGG16) is only involved in classification results, and the feature extraction architecture (U-Net) is only involved in feature extraction. As a result, it is hard to conclude that all the features extracted by the U-Net influences the classification result. A combined network that only classifies the relevant features and uses Grad-CAM as a feature extractor is worth trying. According to Teso [99], the Explanatory Active Learning (EAL) [100] algorithm depends

on a post-hoc explainer, and it can generate a fragile and unfaithful explanation. He says the self-explainable active learning model is a solution to that. It is a combination of active learning and self-explainable neural networks (SENNs) [2]. He names it Calimoch or CALI. More specifically, during active learning, the CALI proposes a decision and an explanation to a user. The user gives feedback (correction) if that explanation is wrong, and based on the feedback, the model learns. This model contains two different loss functions, classification and explanation loss. Also, a regularizer introduces stability and robustness when there is a pixel-wise local change in the input. The classification loss is the classical neural network loss function. However, the explanation loss is a ranking loss that minimizes the square Euclidean distance between the model's current explanation and user feedback while maximizing that distance between the former and wrongly predicted explanation. This ranking loss only contributes when the user gives feedback. The author runs two main experiments on CALI. The first experiment is to check if it learns from user feedback. It shows that classification loss and explanation loss decrease significantly. The author also shows that the classification loss converges fast due to the contribution of explanation loss. Moreover, the author experimented on the number of SENNs' explanation module layers. He sees that if it increases, the classification loss decays faster.

Although this method works in synthetic data, the author does not perform any experiment on a real dataset. This model uses SENNs' architecture which has two combined interdependent models. As a result, it requires more parameters to learn.

Ghai et al. [29] introduce Explainable Active Learning (XAL) in An Empirical Study of How Local Explanations Impacts Annotator Experience in 2020. XAL is an Active Learning method that also includes the explanation of the learning algorithm's prediction for the training sample it wants to learn. Here Active Learning (AL) is a machine learning algorithm with human-in-the-loop for training from un-labeled data by querying a label from a human. It means XAL is nothing but a paradigm of explaining the evolving model during active learning. A similar paradigm, coactive learning (CL), in which a user provides annotation on the prediction, and the model learns from that. In the mentioned paper, the authors also conduct user studies in AL, CL, and XAL to see which methods users prefer when it comes to the application. They studied 37 participants to find the empirical answer of mainly three pre-defined questions. Firstly, how does an explanation impacts the learning algorithm and its outcome? Secondly, how does an explanation affect users' trust, satisfaction, engagement, and cognitive workload? Thirdly, is there any difference in the conclusion of the above two questions during the early and the later learning stages? As a result of the experiment, it is evident that the three models learn relatively on the same scale, but at the early stage, learning is faster than that of a later stage. The accuracies of the three models are the same after learning during the process. However, user trust in the model increases in the early stage of learning. Moreover, during the later stage of learning, XAL increases the user's engagement while AL decreases that. Similarly, the workload on users decreases in the later stage of XAL, but it remains the same on the user in the later stage of AL. It concludes that introducing explanations in process active learning increases user experience. However, the authors do not provide feedback on explanation if the model gives a wrong explanation.

Ross et al. [83] propose an idea of training differentiable machine learning models by constraining its explanation. They show that their approach produces faithful explanations of classification. The input gradient is not the perfect way of explaining neural networks' decisions in all use-case scenarios. So, in their approach, they regularized the loss function using the sum of the square of pixel-wise multiplication of annotation and input gradient. Here, annotation is a binary mask representing each pixel's importance in

the input image. The latter is the gradient of the model's output probability concerning its input. The annotation matrix is possible to get from experts. However, in the absence of that, the algorithm can generate multiple models based on an iterative annotation update. There will be a correct model and corresponding explanation among them. A domain expert can choose the best model suitable for the right reason.

We see "Clever Hans" behavior in machine learning [89]. This behavior happens due to the less significant features in a dataset. Due to this reason, one model can show high performance. This high performance is not valuable at all. It is also known as "Right for the wrong scientific reasons" [89]. Schramowski et al. [89] propose a XIL approach to overcome "Clever Hans" behavior in a plant phenotyping research task. They introduce a human-in-loop framework to involve scientists revising a model using a feedback loop. Their result shows that embracing the user in the learning loop can overcome the "Clever Hans" problem. To include users' feedback, the authors propose a multi-purpose loss function. The loss function has two main parts: Right answer and Right reason. The "Right answer" loss is there to train the model to improve the classification accuracy. Similarly, The "Right reason" loss is to train the model to increase explanation performance. It calculates the difference between the explanation and user feedback. Here, the explanation is nothing but a gradient-weighted class activation map, and the ground truth explanation is a mask localizing the vital region in the input image.

There is a high potential of combining local explanation with active learning to supervise black-box algorithms. These algorithms can misjudge the quality of the model's knowledge. The online learning algorithm manifests its belief by classifying and explaining the label of the query. If the machine performs artificially well on query instances that it chooses, then the model is not misrepresenting its performance. This behavior can bias machines when they present narratives to the user. Popordanoska et al. [76] propose an approach to overcome these narrative biases. Their proposed method allows users to choose query instances. Moreover, it shows the global explanation so that users can choose challenging query instances.

Stuntebeck et al. [97] propose a human-in-loop machine learning framework. This framework collects data from the patient using health sensors and trains a machine learning model on that data. Sometimes, due to the inefficiency of the sensors, the model prediction becomes wrong. To overcome this problem, they involve the user in the learning loop. Occasionally, the model gives the prediction, and the user gives feedback on the prediction by comparing what they are experiencing. Based on the feedback, the model tune itself. This framework is similar to this research involving humans in the learning loop. However, the feedback in this framework is only a yes or no decision. Holzinger et al. [38] argues that automatic ML suffers in performance because of insufficient training data; on the other hand, interactive ML has the flexibility to allow a user to select suitable features heuristically from a vast search space. As a result, it can reduce the complexity of NP-hard using outside knowledge (Human intervention). The authors demonstrated the effectiveness of interactive machine learning and showed how to open a black-box technique to a glass-box one, enabling humans to interact with an algorithm. The authors demonstrate the interactive machine learning method in Ant Colony Optimization (ACO) [38]. It is a solution to the shortest pathfinding problem. According to the author, a human can interact with the algorithm using Human Interaction-Matrix. A matrix allows the user to control ACO's decisions while finding the shortest path. It is an example of human interaction in classical AI problems, but in the same manner, we can involve user interaction in modern AI problems.

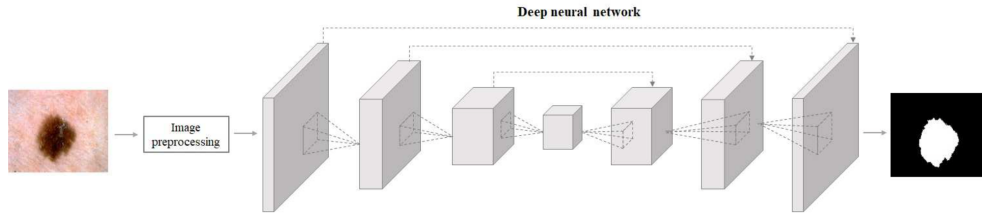


Figure 2.1: This figure [48] gives an overview of the segmentation network. A lesion image is passed through the network on the left side, and on the right side, it generates a segmentation mask.

## 2.4 Domain specific related work: skin cancer detection

Skin cancer analysis using deep learning is an emerging research field. There is a limited amount of research published in this field. There are only 24 research papers published on skin cancer segmentation and 36 articles published on skin cancer classification using deep learning from 2015 to 2019 [47].

Segmentation is the method of dividing a picture into discrete sections containing pixels with similar properties. Skin disease diagnosis relies on segmentation since it allows physicians to see the boundaries of lesions. Figure 2.1 shows an example of skin lesion segmentation using a neural network. Several factors make lesion image segmentation difficult such as skin tones, air bubbles, non-uniform lighting, the physical location of lesions, and lesion variations in respect to texture, shape, size, and location in the image [9]. The physical variances complicate skin disease segmentation in the appearance of skin lesions [48]. Image processing techniques can remove unwanted features from skin lesions [51]. Then deep learning algorithms can be applied to perform segmentation.

There are plenty of deep learning methods used in skin lesion segmentation, and they showed good performance [40, 111, 7, 31, 4]. These researches used modified versions of fully connected CNNs (FCNs), U-Nets, GANs, or encoder-decoder architectures. Several pieces of research were published skin segmentation where the core architecture FCN. For benign nevi, melanoma, and seborrheic keratoses pictures, Goyal et al. [31] suggested a multi-class segmentation approach based on FCN. The authors found dice coefficient indices of 55.7 percent, 65.3 percent, and 78.5 percent for the three classes, respectively, on the ISIC dataset. The authors used several architectures, FCN-AlexNet, FCN-32s, FCN-16s, and FCN-8s architecture, for assessing their performance. The first one is the variant of AlexNet, and the remainings are variants of VGG16. They also used the transfer learning approach to overcome the data limitation. The initial data to train the models were ImageNet [19], and Pascal-VOC [28]. Among the architecture mentioned above, FCN-Alexnet and FCN-8s have superior performance. Phillips et al. [74] propose a new multi-stride FCN architecture for segmenting prognostic tissue features in cutaneous melanoma images. Multiple networks were pre-trained on the PascalVOC segmentation dataset and fine-tuned on the prognostic tissue image. The results demonstrated that the proposed method could attain a high degree of accuracy. Yuan et al. [112] proposed a fully automated solution for skin lesion segmentation that uses 19-layer deep convolutional neural networks that are trained end-to-end and do not require previous data knowledge. Their strategy ensured effective and efficient learning with limited training data. Additionally, they used Jaccard index-based loss function instead of the cross-entropy loss function. They trained and evaluated their model on ISBI 2016 and PH2 datasets, and they found their proposed method outperformed other

state-of-the-art algorithms on these two databases. Li et al. [47] suggested a dense deconvolutional neural network based on residual learning. Dense deconvolutional layers (DDLs), chained residual pooling (CRP), and hierarchical supervision are all parts of the proposed network. It was trained from the beginning to end, with no prior knowledge or sophisticated postprocessing methods required. Extensive trials using the public ISBI 2016 and 2017 skin lesion challenge datasets show that their suggested strategy outperforms state-of-the-art methods in terms of segmentation.

U-Net is a famous architecture for semantic segmentation [82]. The network was built using FCN, and its design has been tweaked and expanded in several studies that have shown superior segmentation results [22, 115, 39]. Moreover, there have been many studies that have used U-Net to segment skin lesions [65, 55, 64]. They also showed an increase in the performance. Chang [11] proposed skin Inceptions V3 NN, a newly constructed transfer learning-based deep neural network that aids in high prediction accuracy. For skin lesion categorization, the author inputs both the segmented pictures and the original dermoscopic images to a deep network made up of two Inception V3 networks. Lin et al. [52] compare U-Net segmentation performance with the C-Means cluster algorithm. C-Means is a clustering algorithm works fast in a CPU. On the other hand, U-Net requires enormous computational power (i.e., computers with GPUs). Compared to the clustering strategy, U-Nets produced a much higher Jaccard Index. U-Net gives a 0.62 average Jaccard index, and C-Means give 0.44 average Jaccard index. To tackle the segmentation of lesions in skin pictures, Ji et al. [41] suggested a binary segmentation approach called HCDS based on salient objects. This approach follows U-Net. The original U-Net model has various issues in its application to skin image segmentation: non-pathological area information remains in the segmentation results. Ji and colleagues extended U-Net to overcome the limitation by adding a hybrid convolution module to the direct connection between the down-sampling and up-sampling. The scientists improve segmentation by deepening the abstract comprehension of input characteristics in shallow layers and removing information interference from non-pathological areas. Test on a portion of ISIC 2018 segmentation data show HCDS method gives an IoU of 0.62 and U-Net gives IoU of 0.56. Codella et al. [15] offer a system that integrates current advances in deep learning with well-established machine learning methodologies to create ensembles of methods capable of segmenting skin lesions and assessing the detected region and surrounding tissue for the identification of melanoma. They used an ensemble of 10 U-Nets with joined RGB and HSV channel inputs for skin lesion segmentation to get the best result. Their method gives 0.841 average Jaccard index comparable with the state-of-art result, 0.843. Canalini et al. [8] propose an architecture that combines multiple pertained feature extractor CNN models and encoder-decoder models. Ensembling multiple models' decision gives an IoU of 0.845. According to Tschandl and colleagues, transferring encoder weights from a network trained on a classification job on pictures from the same domain might provide helpful information for segmentation [101]. They train a fully convolutional network using ResNet34 layers as encoding layers in a U-Net architecture. The model with the transferred information was then trained using the official ISIC 2017 challenge dataset for a binary segmentation. According to the results, the model with fine-tuned weights attained a better Jaccard index than the network with random initializations. People adopted the notion of residual block or dense block into existing image segmentation architectures to construct effective deep networks for skin lesion segmentation, owing to CNN classifiers' exceptional performance in image classification tasks [48]. According to Yu et al. [111], deep networks (more than 50 layers) can acquire richer and more discriminative characteristics for more accurate recognition. They started by building a



fully convolutional residual network (FCRN) for skin lesion segmentation that included multi-scale feature representations. The learned FCRN was then used to extract patches from skin scans containing lesion locations, then used to train an intense residual network for melanoma classification. Finally, they construct a two-stage framework by smoothly integrating the proposed FCRN (for segmentation) with other extremely deep residual networks (for classification). Experiments on the ISBI 2016 Skin Lesion Analysis Towards Melanoma Detection Challenge dataset show that the proposed framework outperforms the competition, placing first in classification and second in segmentation among 25 and 28 teams, respectively.

Besides the method mentioned above, GANs are becoming very popular in medical image segmentation [3, 49, 109, 23]. However, limited research occurred using GAN in skin lesion segmentation [75, 70, 21]. GANs showed potential in lesion segmentation. Udrea and Mitra [102] propose a GAN-based skin lesion segmentation method that can detect lesions with 92% accuracy in pigmented and non-pigmented images. For low-power computers, Sarker et al. [88] propose a lightweight GAN model known as SLSNet. SLSNet employs a GAN model to integrate 1-D kernel factorized networks, position and channel attention, and multiscale aggregation techniques. The 1-D kernel reduces the 2-D filtering cost. The model can discriminate between the lesion and no-lesion features with the help of the position and channel attention module. It showed a Jaccard index coefficient of 90.63% in testing on ISIC2018 data. Peng et al. [72] proposed an adversarial U-Net for semantic segmentation of skin lesions. They connect a discriminator network with U-Net by a particular convolutional layer and train both networks alternately. They found that this architecture improved the segmentation accuracy to 0.94 (dice score). Besides FCN and U-Net, and GAN, there are other approaches used. For example, Ünver et al. [103] propose an innovative and successful pipeline for dermoscopic skin lesion segmentation that combines a deep convolutional neural network called 'You Only Look Once (YOLO)' and the 'GrabCut' algorithm. They follow four steps for segmentation, post-processing, detection of the lesion, lesion segmentation, and post-processing of segmentation mask. The model gives a close result to other contemporary approaches.

Besides segmentation, several pieces of research are published in skin lesion classification [18, 56, 110, 58, 60, 50]. Some of them used shallow architecture, but some utilized deep architecture. This section will go through some impactful research on skin cancer classification. According to Esteva and colleagues, classifying cancer in human skin lesions is a challenging task due to fine-grained features in lesions [27]. They show how a single CNN can classify skin lesions using only pixels and disease labels as input and can be trained end-to-end using lesion images. The number of images was 129,450 and contained 2,032 disease labels. They compare the performance of CNN with 21 certified dermatologists and result shows that CNN's performance is comparable to the dermatologists. The CNN model used in this research is Inception v3, trained on 1.28 million images with 1000 classes. Dorj et al. [24] propose a CNN and SVM based model for skin lesion classification. Before training, they crop and pre-process the training images to remove unnecessary information. Then they use pre-trained AlexNet for feature extraction. After that, they pass the feature to an SVM for classification. The algorithm shows comparable performance with the state-of-art. Hekler et al. [36] experiment combined performance of skin lesion classifier and dermatologists. They measure the classifier's performance, dermatologists' performance, and dermatologists' and classifier's combined performance. They found that combined performance is better than individual performance. Furthermore, they found a classifier defeat dermatologists in detecting skin cancer. Adjubo et al. [10] integrate Gabor filter with CNN to improve the accuracy. They found that integrating the Gabor filter improves the accuracy by 2.37%.

Gabor filtering can effectively extract spatial information such as edges and textures, thereby alleviating the strain of feature extraction on CNN. Saba et al. [85] propose a cascaded system that uses contrast enhancement, boundary extraction, and transfer learning together for image classification. Firstly, they transform images to HSV color space and enhance the contrast using the first local Laplacian filtering. Secondly, they use color CNN and XOR operation to extract the lesion from images. In the third stage, they use fine-tuning in the Inception V-3 pre-trained (ImageNet) network for feature extraction from the fully connected and average pooling layer. They combine the features using hamming distance in the final stage and pass it to an MLP network (multi-layer perceptron module) for classification. This approach gives an accuracy of 98.4%, 95.1%, and 94.8% in PH2, ISIB, and ISIC 2017 datasets, respectively. Kawahara and colleagues use CNN feature extractor and linear classifier for lesion classification [43]. They extract features from the pre-trained AlexNet and use these features to train a linear classifier. This approach had superior results compared to the other methods of that time.

We see that there are several approaches for skin cancer classification. Many of the methods have human-level accuracy. However, the application of these methods is constantly challenged by critique for legitimate reasons; for example, the proposed classifiers are black-box models, need training updates on new data, and lack interactivity with humans and the environment. So, in this thesis, we explore the interactive side of skin cancer classification.

---

## Chapter 3

### Technical background

In this chapter, we will look at the basic building blocks of this research. There have been several research works followed to build the experiment setup. As we know, transfer learning is prevalent in deep learning research; this research also utilizes transfer learning. For explaining convolutional neural networks, the saliency map has become very popular. In the related work chapter, it is already mentioned that there are several methods for explaining neural decisions—many of these research leverage the saliency map for explainability. For example, CAM, Grad-CAM, Grad-CAM++ are few among them. For this research, the Grad-CAM algorithm is followed. Here we will discuss how we integrate the Grad-CAM algorithm for generating the explanation in the interactive loop. The saliency map has a continuous pixel value, and it gives us how effective a pixel in an image is for belonging to a class. We need the ground truth saliency map to integrate the saliency map with the loss function. However, there is no available ground truth saliency map for experimental purposes. There are only binary masks available as ground truth. We need to convert the saliency map to a binary mask to adapt to the ground truth. In the thresholding section, we will discuss more on that. There are two kinds of loss functions, classification loss and explanation loss, used for training the model. We will discuss the background of this loss function. For optimizing the network, an online gradient descent algorithm is used.

#### 3.1 Transfer learning

Training CNN from scratch requires a lot of data. It is the biggest challenge in deep learning. In machine learning, it is necessary to be consistent in data during training and testing. For example, if training data of an ML model whether data, the test data must be weather data. However, in deep learning, this statement does not hold. Deep learning necessitates many labeled data, making it difficult to train a model from scratch.

Transfer learning (TL) is the solution to this problem. Unlike machine learning, in deep learning one can transfer knowledge from different domains. Data scarcity is extreme in the medical domain. Medical data is sensitive. Due to these reasons, hospitals do not publish data in public databases. As a result, transfer learning has many potentials in

the medical domain. According to Morit et al. [62], the information gathered from vast amounts of non-medical data can be transferred to solve a specific medical problem using TL. For example, To tackle a medical imaging problem, parameters from well-trained CNN models on non-medical ImageNet data containing natural images (e.g., AlexNet [45], VGG [94], and ResNet [35]) can be transferred to a CNN model. It is a recent trend that researchers use non-medical data for medical image analysis. According to a survey by Litjens et al. [54], there are around 300 papers on medical image analysis using deep neural networks. In the deep learning community, transfer learning is a popular approach to utilize knowledge from one task to another.

In the previous paragraph, we discuss feature extraction using a convolution kernel. Here we discuss how one can use this learned convolution kernel in different tasks. The use of learned convolution kernels in different tasks is known as transfer learning. For example, one can train a model in a general object classification task<sup>1</sup> and use the learned CNN kernel in the skin disease classification task. Although the domains of the tasks are different, the transfer-learning still works very well. Kernels learn how to detect coarse features, for example, edge and texture. Coarse features generally exist in every kind of image. As a result, the models do not have to learn kernels for detecting coarse features. As a result, training time and data demand are reduced. We do not train the transferred kernels, but we only train additional kernels. If we train the transferred kernels, we call it fine-tuning. In fine-tuning, the transferred kernels initialize the model, and we train the model and update the kernels for better performance.

## 3.2 Base Architecture

A pre-trained VGG 16 model is the basis of this research. The VGG16 was trained initially on ImageNet. For this experiment, we have a minor modification of VGG16. First, we remove all the fully connected layers and the output layer. Then we added two fully connected layers followed by two dropout layers consecutively of 2048 neurons each. Finally, the output layer contains eight-way softmax. Moreover, we use half of the neurons in each fully connected layer and two dropout layers to reduce the training time. We get the idea of using dropout layers from the original AlexNet. Listing 3.1 is an overview of the modified model. After the model modification, we train on ISIC2019 classification data. In the table 3.1 we describe the detailed description of fine-tuning of modified VGG16.

### 3.2.1 Training data

In may 2019, The International Skin Imaging Collaborator (ISIC) published skin lesion classification data. We use this data for training and testing the initial model. This data contains eight classes: melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma, vascular lesion, and squamous cell carcinoma. We name them MEL, NV, BCC, AK, BKL, DF, VASC, and SCC in the same order for ease of reading. We used 20273 images to train the model, 2529 images for validation, and 2529 for testing. Table 3.1 gives a short description of training and testing. We use this fine-tuned model for the main task of the thesis. Let's call this Skin Care VGG16 Model, and it is developed using the Keras framework.

<sup>1</sup><https://image-net.org/challenges/LSVRC/>

Listing 3.1: An excerpt of the VGG16 architecture used for the multi-label classification task.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 227, 227, 3)	0
block1_conv1 (Conv2D)	(None, 227, 227, 64)	1792
block1_conv2 (Conv2D)	(None, 227, 227, 64)	36928
block1_pool (MaxPooling2D)	(None, 113, 113, 64)	0
block2_conv1 (Conv2D)	(None, 113, 113, 128)	73856
block2_conv2 (Conv2D)	(None, 113, 113, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 2048)	51382272
dropout_1 (Dropout)	(None, 2048)	0
fc2 (Dense)	(None, 2048)	4196352
dropout_2 (Dropout)	(None, 2048)	0
predictions (Dense)	(None, 8)	16392
Total params: 70,309,704		
Trainable params: 70,309,704		
Non-trainable params: 0		

Skin Care VGG16 model		
Training properties	Value	Details
Task	Fine tuning	Fine tuning On ISIC2019 classification data
Training samples	20273	A large portion of ISIC2019 data
Validation samples	2530	A small portion of ISIC2019 data
Test samples	2529	A small portion of ISIC2019 data
Number of classes	8	-
Classes	MEL, NV, BCC, AK, BKL, DF, VASC, SCC	melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma, vascular lesion, and squamous cell carcinoma
Class weights	0.178, 0.508, 0.131, 0.034, 0.104, 0.010, 0.010, 0.025	-
Color space	RGB	Red green blue; no color augmentation
Image size	227 x 227	-
Image resizing filter	Nearest	-
Augmentation	Horizontal flip, 24 rotations	24 rotations over 360 degree
Epoch	20	-
Train accuracy	0.475	Global
Test accuracy	0.850, 0.819, 0.926, 0.962, 0.894, 0.992, 0.996, 0.976	Order: MEL, NV, BCC, AK, BKL, DF, VASC, SCC
Test sensitivity	0.608, 0.743, 0.765, 0.651, 0.649, 0.739, 0.920, 0.613	Order: MEL, NV, BCC, AK, BKL, DF, VASC, SCC

Table 3.1: Fine tuning description of modified VGG16 ImageNet model

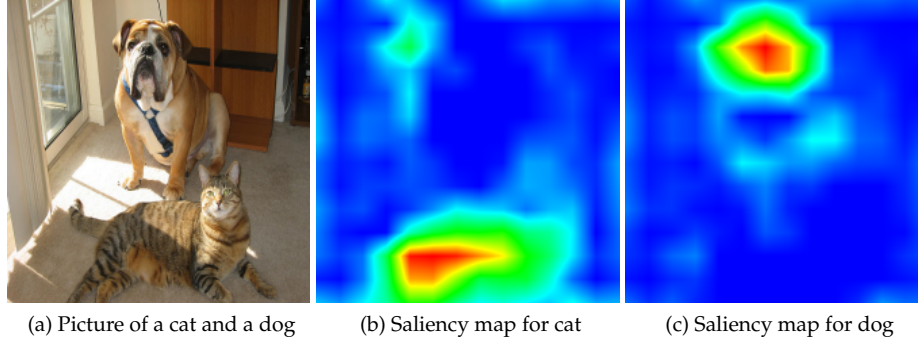


Figure 3.1: Saliency map generated using Grad-CAM [17]. The original picture is on the left, and the two pictures on the right show the saliency map of the cat and the dog.

### 3.3 Saliency map

In the introduction, we told that saliency maps could explain CNN's, and we also found that there are multiple researches on explaining neural networks using saliency maps. In this section, we focus on the saliency map in depth. Generally speaking, in computer vision, a saliency map is a picture that emphasizes the region on which people's eyes focus initially [105]. The purpose of a saliency map is to highlight important characteristics of an image. Each pixel in a saliency defines how vital that specific pixel is for the image's subject. Saliency maps can be used in numerous subfields of computer vision, such as image and video compression, video and image quality assessment, and object detection. Moreover, explaining CNN's decision saliency map is becoming very popular nowadays [114, 90, 12, 95, 69, 5]. Generating saliency maps from CNN's classification highlights discriminatory regions for different classes. In the figure 3.1, we show how the Grad-CAM saliency maps look like. Grad-CAM generates the two saliencies on the right (3.1b and 3.1c). The CNN model used in this task is VGG16. VGG16 predicts both cats and dogs (Figure 3.1a) with two probabilities. The Grad-CAM algorithm marks the cat's location using the first saliency map when we consider the cat picture. On the other hand, Grad-CAM locates the dog's face when it is the dog. In this example, the saliency map explains the prediction visually. This kind of saliency map is also called a class activation map.

Grad-CAM saliency map can be generated from any convolution layer of a CNN. In the picture mentioned above, they consider the last convolution layer. Let us consider the output (Receptive field) of the last convolution layer as  $A_k$ , and  $k$  is the number of kernels in the last convolution layer. For each kernel, we get one output. That means we have  $k$  number of outputs. These  $k$  outputs detect general features from the input image. We can multiply the  $k$  number of coefficients ( $w_k$ ) with the outputs to generate class specific saliency map. These  $k$  coefficients tell us which output has more influence on the classification. After that, we sum up all of the output to get the saliency map. The mathematical definition of  $w_k$  and saliency map is equations 3.1 and 3.2. We can also visually understand how to generate a saliency map by looking at figure 3.2 [90].

$$w_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\delta y_c}{\delta A_{i,j}^c} \quad (3.1)$$

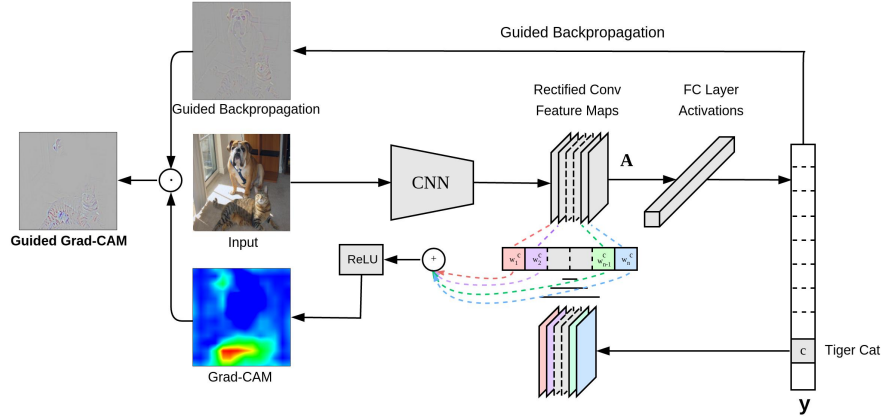


Figure 3.2: An overview of the Grad-CAM algorithm [17].



Figure 3.3: Saliency map from Figure 3.2 is converted to binary masks (Left to right) using threshold value from 0% to 100% of max pixel value of the saliency map with an increment of 10%. [67]

$w_k^c$  is the class specific gradient ( $c^{th}$ ) of the input image in the  $k^{th}$  convolution layer.

$$Saliency\ map = ReLU \left( \sum_k w_k^c A^k \right) \quad (3.2)$$

### 3.4 Thresholding saliency map

We use simulated user feedback where the feedback comes from previously-stored dermatologist-annotated attribute maps. The attribute maps are binary masks. We already know that we can generate saliency explanations using the Grad-CAM. However, we cannot compare them with the feedback. We must adapt the explanation to binary mask pattern. To adapt the explanation, we can make a thresholded saliency map. However, we can not use a random threshold value for that. For using a random threshold value, we could lose some information. Nunnari et al. [67] show that choosing a threshold value for converting a saliency map to a binary mask is data-dependent. We could cut a saliency map at 40% of the maximum pixel value for some data; for others, the value could differ. They propose that thresholding at 50% of the max pixel value is a good approximation for the ISIC2018 task to attribute detection data. As our data is the same, we also use 50% of the max pixel value for the threshold. Figure 3.2 shows a melanoma image with its corresponding grey and saliency maps, and figure 3.3 shows how different threshold values affect the saliency map. The saliency map is scaled to  $[0.0 - 1.0]$ , so 0.5 is the threshold value.



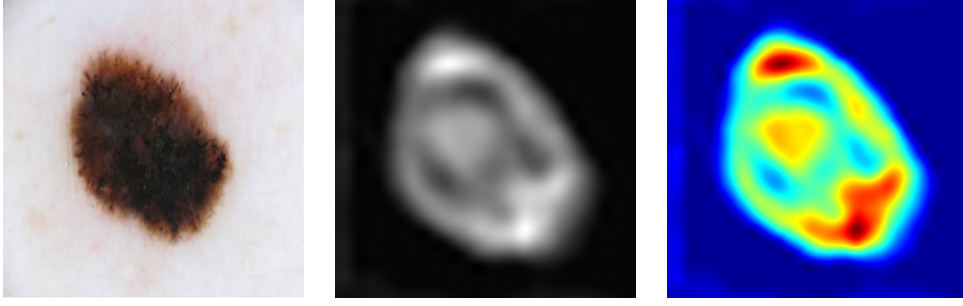


Table 3.2: A melanoma sample image with gray map and saliency map [67]

### 3.5 Loss function

The loss function is also known as cost function or objective function that maps the difference between the prediction and ground truth in deep learning. By reducing the loss value, we optimize the neural network. In this thesis, we have two kinds of loss functions. The first kind is the *classification loss function*, and the second is the *explanation loss function*.

The classification loss function is used to optimize the model on classification data. It compares the difference between classification ground truth and calculates the gradient of the loss to update the model weights. In the thesis, we use the same classification loss of the Skin Care VGG 16 model: cross-entropy loss (Equation 3.3) [63].

$$\text{Classifications Loss: } L_{cls} = - \sum_{c=0}^M y_{o,c} \log(p_{o,c}) \quad (3.3)$$

Where  $M$  is the number of classes,  $y$  is the binary indicator (0 or 1), the class label  $c$  and  $p$  is the predicted probability.

Similarly, explanation loss is used to optimize the model on explanation data. One can use an inverse similarity function for calculating the distance between the predicted and the ground truth explanations. This thesis will use the Jaccard index to formulate loss function. Several papers used the Jaccard index to develop loss function [26, 79]. Jaccard loss comes from the idea of "intersection over union (IoU), which is also known as Jaccard index". [79]. The IoU score is a typical performance metric for the object category segmentation issue. The equation 3.4 defines the IoU measure. For an object detection task, it gives the similarity between the predicted region and the ground-truth region.

$$\text{Jaccard index: } J = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A| \times |B|} \quad (3.4)$$

Jaccard index is zero if the two predicted region and ground truth region does not match. The maximum value of the Jaccard index is one. IoU is generally used for boolean variables. To utilize it for a continuous variable, we can modify it to a continuous version [79]. Equation 3.5 gives the mathematical definition of IoU for continuous variables.

$$J(y, \hat{y}) = \frac{y \times \hat{y}}{y + \hat{y} - y \times \hat{y}} \quad (3.5)$$

We can modify the Jaccard index to Jaccard loss using equation 3.6, which is a quadratic equation.

$$\text{Explanation loss: } J_L(y, \hat{y}) = \left(1 - \frac{y \times \hat{y}}{y + \hat{y} - y \times \hat{y}}\right)^2 \quad (3.6)$$

### 3.6 Online and incremental machine learning

Many machine learning and deep learning methods face challenges due to the lack of available data at training time, and some data changes sequentially (e.g., stock market data and environment data); as a result become uninformed on new data. Moreover, day-by-day new sensors and devices are being developed. They have better performance than the former versions in their respective task. For example, current camera technology is more accurate than in the past. Similarly, humans also gather more experience and knowledge over time; as a result, human feedback also can change due to the underlying experience. So, when our training data is limited or comes in sequential order, or has the possibility of periodic updates, we must follow incrementally or online machine learning methods to keep the model up to date on new information. According to Hazan et al. [34] in many practical application, modeling and utilizing classical theory and mathematical optimization is infeasible; however, considering optimization as a process can model systems that are related to our daily lives. Learning from continuous observation is the same as optimization as a process. The framework deals with it is known as online convex optimization (OCO). OCO is the generic idea behind online and incremental machine learning. The term "online machine learning" refers to a machine learning approach in which data is made accessible sequentially and is used to update the best predictor for future data at each step [104]. On the other hand, offline learning algorithms learn from data collection. Online learning methods can update the model's knowledge iteratively on new data. Also, online machine learning algorithms have less computational power demand due to the limited amount of data. Contrary, offline algorithms require more computational power for optimization. So, online learning is commonly used when the training on a large amount of data is infeasible. A very similar approach is called incremental learning. The incremental learning method continuously feeds the data to the existing model for knowledge update. Incremental learning is included in many standard machine learning methods such as decision trees, artificial neural networks, and SVM [20].

Formally we can define incremental learning as follows. A hyperspace,  $H$ , contains all possible models; we search for a function  $f$  where  $f \in H$ .  $f$  creates the relation  $f : X \rightarrow Y$ ; where  $X$  is the feature space, and  $Y$  is the output space.  $f$  predicts the joint probability  $p(x, y)$  where  $x \in X$  and  $y \in Y$  are training instances. However, the learner does not know the instances' distribution  $p(x, y)$ . A loss function  $L : X \rightarrow Y \in R$  measures the distance between the  $f(x)$  and  $y$ . The ideal goal is to choose an  $f$  that minimizes the overall distance of all instances. This thesis assumes that the data comes from users in a continuous manner. So we use online gradient descent to optimize the model.

### 3.7 Optimization technique

Previously we talked about loss function. The absolute value of loss tells us how poorly a model is performing. The optimization target is to learn the model's weight that

generates a minimum loss. A neural network loss function is multidimensional and difficult to visualize. There are plenty of local minima and one global minimum. The optimization technique tries to find out the global minimum in the hope of increasing the model's performance. But it is not guaranteed whether we can reach the global minima. It is impossible to tell the exact number of local minima in a loss function. The gradient descent technique can search for the global minimum in the loss function hyperplane. After an extensive search, we choose the weights of a model based on the minimum loss. We can visualize the hyperplane for quadratic function and see only one minimum. Gradient descent guarantees that reaching minima is possible. We also need to consider a fine weight initialization to optimize neural network loss. Without a good weight initialization, optimizing a neural network might take a long time.

"Training deep models is a sufficiently difficult task that most algorithms are strongly affected by the choice of initialization. The initial point can determine whether the algorithm converges at all, with some initial points being so unstable that the algorithm encounters numerical difficulties and fails altogether."- Goodfellow et al. [30]

Transfer learning reduces the amount of work for choosing weights for a model. There are several gradient-based algorithms for minimizing loss and improving the model performance. We follow the online gradient descent algorithm, which is stochastic gradient descent with batch size one.

Stochastic gradient descent is an iterative approach to updating the weight of the neural network. In this iterative approach, one can update the weights of a network based on small batches of samples. It is called mini-batch gradient descent. The size of a batch influences the computation requirement neural network. A larger batch of samples requires more computational power and memory than a smaller batch. The batch size can not be large for image data due to the huge amount of information single image. For the initial training of the classifier, we used a mini-batch algorithm. But for training the final model, we use SGD with batch size one, which reduces memory demand. The pseudo code for the stochastic algorithm is presented below.

---

**Algorithm 1** Stochastic gradient descent

---

**Require:**  $x, y \in X \times Y$

**Require:**  $\lambda$

▷ Learning rate

**Require:**  $e$

▷ Number of epochs

Initialize  $w$

**while** loss is not optimal **do**

**while**  $n < N$  **do**

▷  $N$  is the number of total samples

$L(w) \leftarrow f(x, y)$

▷  $w$  is the weight

$w \leftarrow w - \lambda \nabla L(w)$

▷  $\nabla L(w)$  is the gradient of the loss

**end while**

**end while**

---

---

## Chapter 4

### Method

This thesis aims to build an architecture that explains decisions to users and gives them the possibility to perform corrections that improve the model’s performance. We know that convolutional neural network architectures are available, but they have limitations. They are trained on large amounts of data, but they do not have a built-in structure for generating explanations. In most cases, we need to use an explanation algorithm to explain the decision of these classifiers. This research proposes a self-explainable model that can be trained individually using explanation loss and classification loss. We can also train this model by combining both losses. Moreover, the model will interact with a simulated user, which will give feedback on classification and explanation, and the model will learn from the feedback. Figure 4.1 provides an overview of the system. From the left side, we see a skin lesion is fed into a convolution neural network. The convolution neural network detects the class of the lesion and explains the classification to a dermatologist (on the right side). The dermatologist is not convinced with the explanation, so s/he gives it a feedback. Based on the feedback, we re-train the model. In this chapter, we will discuss how to generate the explanation from a simple VGG16 convolutional neural network and how to integrate the explanation with the loss function of the neural network. Moreover, we will also discuss how to simulate a dermatologist.

#### 4.1 Self explainable model

The Skin Care VGG16 model is originally trained using the Keras [13] framework. The Human-in-the-loop model was not in the plan at the first time. So for modifying the network and adapting the human-in-the-loop interface, we converted the model from Keras framework to PyTorch [71] framework. The PyTorch framework is more flexible to architectural modification than the Keras framework [59]. We use the MMDnn<sup>1</sup> library for model conversion. It is a verified library for converting deep neural network models to convert from one framework to another. We compared the performance of both versions on skincare test data. Without small numerical changes, both versions showed the same characteristics in classification. Tables 4.1 and 4.2 show the performance of

---

<sup>1</sup><https://github.com/microsoft/MMDnn>

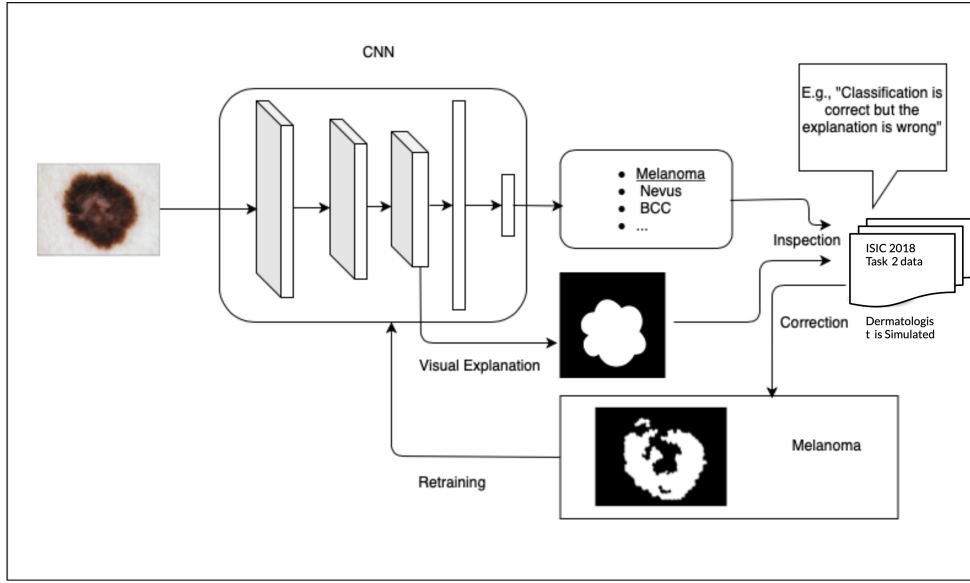


Figure 4.1: Method overview: interactive learning is a way of training the model using user feedback. In our case, the model produces two kinds of output: explanation and classification. So, the idea of interactive training is to retrain the model if any of these outputs is wrong.

the Keras version and the PyTorch version. Looking at the tables, we see no significant differences between the values. So we proceed to the next step, adapting the model for generating a saliency map.

Class	Accuracy	Specificity	Sensitivity	F1	AUC
MEL	0.85	0.90	0.60	0.59	0.87
NV	0.82	0.89	0.74	0.81	0.92
BCC	0.93	0.95	0.76	0.73	0.96
AK	0.96	0.97	0.65	0.54	0.96
BKL	0.89	0.92	0.65	0.56	0.91
DF	0.99	0.99	0.74	0.62	0.98
VASC	1.0	0.99	0.92	0.81	0.99
SCC	0.98	0.98	0.61	0.55	0.96

Table 4.1: Performance of the Original Model (Keras)

Requirement: We have to extract saliency maps while training, and the implementation must be differentiable to allow the backpropagation.

Hence, we can not use an independent GradCAM algorithm because we need a gradient of the saliency map for our next task. So we have to integrate the grad cam functionality with the architecture. We already know that we have to generate CAM from the last convolution layer, so we modify the architecture to get the receptive field of the final convolution layer. Let us name the architecture SkinCare VGG16. The SkinCare VGG16 architecture has two outputs: intermediate output from the last convolution layer and the last layer (softmax output), and we name the intermediate output as activation map (AM). We can generate CAM from AM during training and testing. We know from

Class	Accuracy	Specificity	Sensitivity	F1	AUC
MEL	0.84	0.89	0.60	0.57	0.86
NV	0.81	0.89	0.73	0.80	0.92
BCC	0.93	0.95	0.79	0.74	0.96
AK	0.96	0.97	0.62	0.52	0.95
BKL	0.89	0.92	0.64	0.55	0.90
DF	0.99	0.99	0.90	0.61	0.97
VASC	1.0	1.0	0.92	0.87	0.99
SCC	0.98	0.99	0.58	0.55	0.96

Table 4.2: Performance of the Original Model (PyTorch)

equation 3.2 that we can multiply  $A^k$  with the  $w_k$  and pass with the ReLU function to generate the CAM. In this case,  $A_k$  is the AM. Our next task is to get  $w_k$  while training. We will utilize a simple approach to get  $w_k$  during the training. We know that  $w_k$  is related to the class-specific gradient. Therefore, we need to calculate the gradient for a given image in the last convolution layer. But there is another challenge: during training, we also need to calculate the gradient of each layer and update the model using online gradient descent (OGD).

Technically, calculating the gradient two times destroys the model knowledge. So we created copy of the model: gradient instance. Gradient instance calculates the gradient of the model in the last convolution layer. We calculate the gradient instance's gradient ( $w_k$ ) with respect to the ground truth class and only store it in the memory. After generating  $w_k$ , we delete the gradient instance and keep the primary model for training. We get two outputs during the forward pass, AM and sigmoid output. We multiply the AM ( $A^k$ ) and  $w_k$  and pass it to a ReLU to generate CAM (saliency map).

After generating CAM, our next task is to create an explanation from the CAM. The explanation is a binary mask. We use a threshold function that converts the CAM to an explanation map. As a result, our model has two outputs, explanation and classification, each time passing an image through the network. Figure 4.2 shows the graphical overview of generating explanations and classification output.

During training, before each forward and backward pass, we update the gradient instance of the model from the updated primary model. It allows keeping track of weight update in each instance.

## 4.2 Integrating the explanation with the original model

In a real-life situation, we get feedback from users as ground truth. Therefore, we can create two loss functions for the model: explanation loss and classification loss. As we said before, there will be no user input. We will use the previously annotated attribute map to mimic the user. We will discuss the interactive loss function in the method chapter. But before that, we talk about generating a saliency map in the interactive training. For interactive training, we consider a single sample in each training loop. A bigger batch size increases the workload on the GPU. So batch size one is consistent in this study. Practically, we get data from the data loader: input image, ground truth attribute map, and ground truth class. Our target is to increase the similarity between predicted and ground truth explanations. We use the Jaccard index to find the similarity. And we also

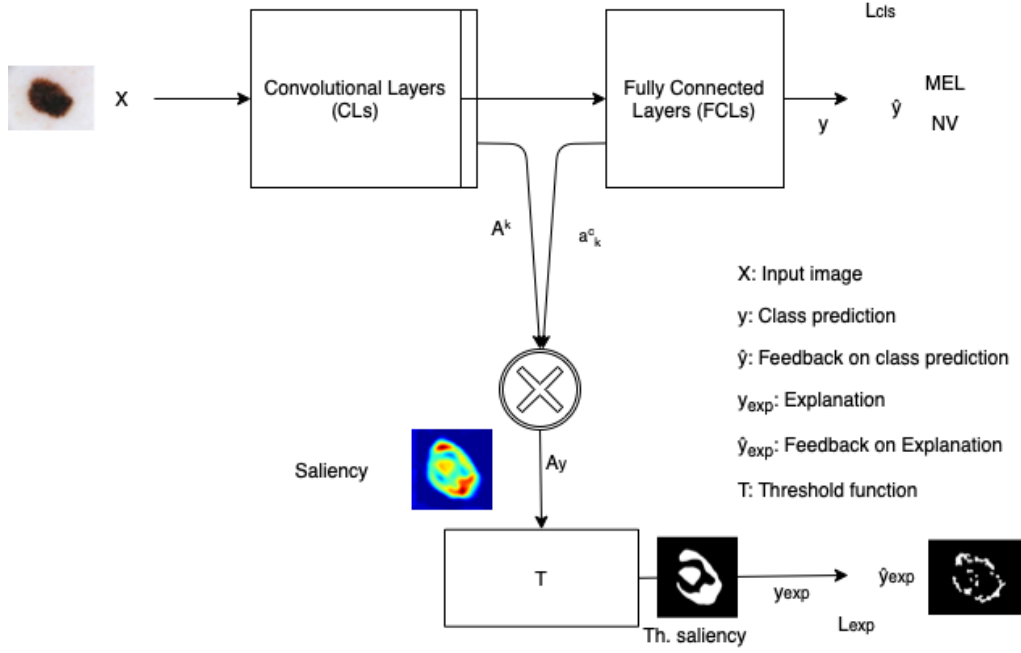


Figure 4.2: Flow of input image through the network.

predict the class and using the cross-entropy loss between predicted class and ground truth we reduce the classification error.

In this experiment, we are using VGG16. VGG16 can be split into two individual networks: convolution and fully connected layers. In general, any CNN architecture with fully connected layers at the end can be considered a base architecture for this experiment.

We can partition a CNN into two functional blocks, convolutional layers (CL) block and fully connected layers (FCL) block. In the figure 4.2, we see that the input images ( $X$ ) feed into the CL at first, and the outputs of CL pass through the FCL to generate the output ( $y$ ). In general, the CNN model is non-explainable, but using the idea of Grad-CAM, we can produce a class-specific saliency map at the end of CL. The block  $T$  can modify the saliency map into an explanation map or a feature map ( $y_{exp}$ ) according to the designer's choice. Here we use the pre-trained CNN's weights, but for introducing interactivity during online learning, we modify the loss function of the whole network. The loss function has two components, classification loss and explanation loss. The classification loss is the loss of a pre-trained network. However, the explanation loss is a newly introduced function, and it punishes the overall cost based on the difference between generated explanation and user feedback on explanation.

The loss function can be written as

$$L = (1 - \lambda)L_{cls}(y, \hat{y}) + \lambda L_{exp}(y_{exp}, \hat{y}_{exp}) \quad (4.1)$$

Here  $L_{cls}$  and  $L_{exp}$  are the classification and explanation loss respectively.  $\lambda$  is the hyper-parameter that can be search during the parameter tuning.

$$L_{cls}(y, \hat{y}) = - \sum_i \hat{y}_i \log(y_i) \quad (4.2)$$

$$L_{exp}(y_{exp}, \hat{y}_{exp}) = \sum_i J(y_{exp_i}, \hat{y}_{exp_i}) \quad (4.3)$$

$$y_{exp} = T(A_y) \quad (4.4)$$

$$A_y = ReLU(\sum a_k^c A^k) \quad (4.5)$$

Where  $A^k$  is the activation in a convolution layer,  $A_y$  is the output of Grad-CAM and  $a_k^c$  is defined as

$$a_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\Delta y^c}{\Delta A_{ij}^k} \quad (4.6)$$

### 4.3 User feedback

In general, user feedback is the information users provide to an agent for a future update of agent knowledge. Getting user feedback for an experiment is very costly and time-consuming especially in the medical domain. However, if we have user annotated data, we can consider it as feedback. ISIC 2018 [16] published data on skin lesion attributes. This data contains images of skin lesions and masks of five different attributes, pigment network, negative network, streaks, milia like cyst, and globules, of the lesion. These attribute maps are binary masks locating the different attributes. Union of all of these attribute maps can provide generalized information on skin lesions. So, we consider the union of the attribute map as feedback from the user. We call this explanation feedback and consider the ground truth class label as classification feedback. We keep 10% of the data untouched to check the model performance and to see if the model learns from feedback. More details on the data will be presented in chapter 5.

### 4.4 Implementation

The architecture and the proposed explainable model are built using the PyTorch framework. PyTorch is a powerful python deep learning framework. The self explainable VGG16 model uses GPU for computation. We need to classify the input image, generate the explanation, and compare it with user feedback during the simulation. The loss function makes the comparison. In modeling the loss function, we must consider that generating an explanation requires calculating the gradient. So we have to calculate the gradient of the model with respect to the model's prediction. In PyTorch, we can use the backward() function to calculate the gradient in each layer. We only store the gradient of the last convolution layer. To keep the main model safe from unexpected weight updates, we create a clone and use that cloned model to get class-specific weight. After that, we forward the image to the main model and get the activation of the last convolution layer. We get the Grad-CAM saliency map by multiplying the activation and class-specific weight. Next, we convert the saliency map to a binary mask. Finally,



compare the feedback image with the binary mask using Jaccard loss. Algorithm 2 shows the training steps for the self-explainable model.

---

**Algorithm 2** Training the self-explainable model

---

**Require:**  $e$   $\triangleright$   $e$ : number of epoch  
**Require:**  $m = F(\theta)$   $\triangleright$   $m$ : pre-trained model  
**Require:**  $X, Y, Z$   $\triangleright$   $X$ : input sample,  $Y$ : label,  $Z$ : binary mask (i.e. explanation ground truth)  
**Require:**  $N$   $\triangleright$   $N$ : total samples  
**Require:**  $L_{cls}$   $\triangleright$   $L_{cls}$ : classification loss function  
**Require:**  $L_{exp}$   $\triangleright$   $L_{exp}$ : explanation loss function  
**Require:**  $\gamma$   $\triangleright$   $\gamma$ : learning rate  
 $i = 1$   
**while**  $i \leq e$  **do**  
     $n = 0$   
    **while**  $n \leq N$  **do**  
         $m_g \leftarrow \text{deepcopy}(m)$   
         $a_k^c \leftarrow \frac{1}{R} \sum_p \sum_q \frac{\delta m_g(x)[c]}{\delta A_{ij}^k}$   $\triangleright \frac{\delta m_g(x)[c]}{\delta A_{ij}^k}$ : class-specific gradient on layer  $k$  for  
        image  $x$   
         $A^k, \hat{y} \leftarrow m(x)$   $\triangleright A^k$ : receptive field of layer  $k$ ,  $\hat{y}$ : predicted class  
         $S \leftarrow \text{ReLU}(A^k \times a_k^c)$   $\triangleright S$ : saliency map  
         $y_{exp} \leftarrow \text{th}(S, t)$   $\triangleright t$ : threshold value  
         $L(\theta) \leftarrow (1 - \lambda)L_{cls}(y, \hat{y}) + \lambda L_{exp}(z, y_{exp})$   $\triangleright \lambda$ : hyper-parameter for loss balance  
         $\theta \leftarrow \theta - \gamma \nabla L(\theta)$   $\triangleright \nabla L(\theta)$ : gradient with respect to loss  
    **end while**  
**end while**

---

---

# Chapter 5

## Experiments and Results

### 5.1 Data description

In this thesis, we experiment with the data from ISIC 2018 and 2019: Skin Lesion Analysis Towards Melanoma Detection [32]. ISIC is a skin cancer detection challenge on dermoscopic images. ISIC 2019 data is used for training and ISIC 2018 task 2 data is used for simulation. Every year they publish dermoscopic data and arrange a challenge. In 2018 ISIC published three kinds of data: segmentation, attribute detection, and classification. In this thesis, we only experiment on attribute detection data (ISIC 2018 task 2 data), because the attribute ground truth is human annotated. It contains images of lesions. The lesion images were collected with different types of dermoscopy from many parts of the human body. Each image includes a primary lesion and other marks, e.g., secondary lesion, fiducial marker, or other pigmented regions. On the other hand, response data (ground truth data) contains five binary masks locating five kinds of attributes in each lesion image. The binary masks are pigment network, negative network, streaks, milia like cyst, and globules. The mask dimension is the same as the decimation of the lesion image. They are 8-bit binary images in PNG format. The pixel value 0 indicates the dermoscopic attribute is absent, pixel value 255 means the dermoscopic attribute is present. Also, the image has a ground truth label for classification. There are only three available classes in this dataset: nevus, melanoma, and benign keratosis. Figure 5.1 describes the three samples from the data. We combine this ground truth attribute to generate explanation feedback using union (Figure 5.2) operation.

### 5.2 Data preparation

ISIC2019 data was used to train a baseline classification model, and ISIC2018 attribute detection data was used for simulation. Before the experiment, we checked the data overlap between ISIC2019 and ISIC2018 attribute detection data. One can raise the question of why is 2018 Task 2 data (attribute detection data) simulation data. ISIC

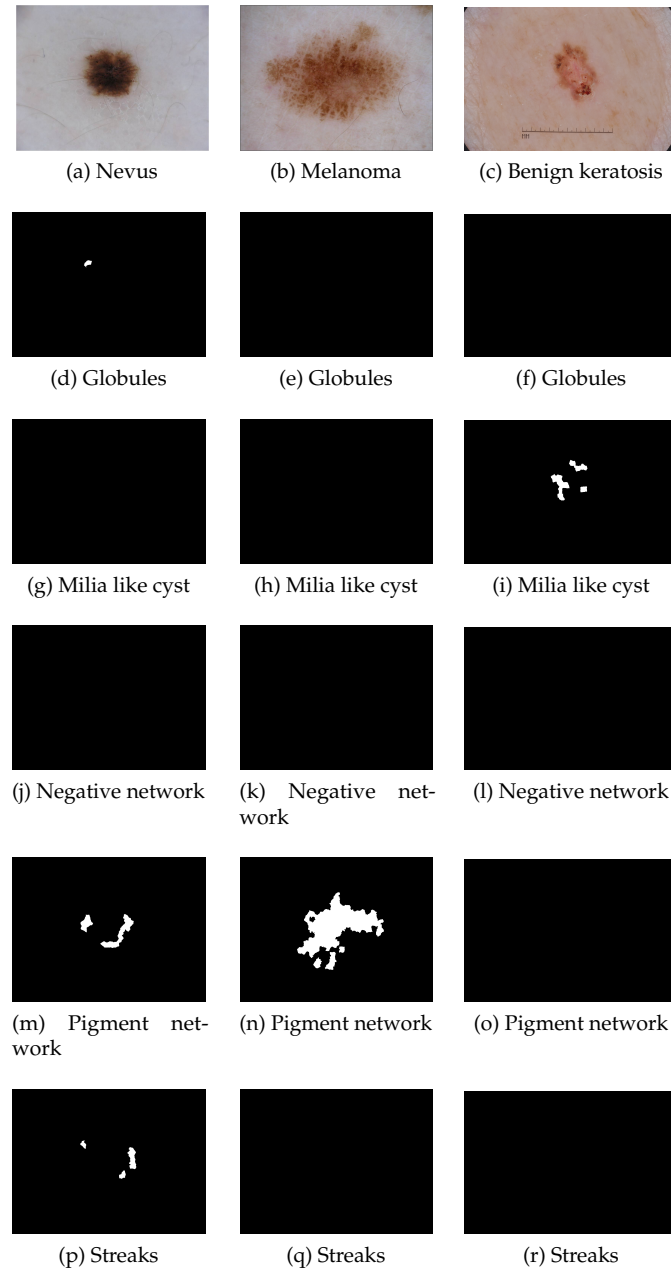


Figure 5.1: Sample images and ground truth attribute of nevus, melanoma, and benign keratosis. They are the only available classes in the ISIC2018 task 2 dataset

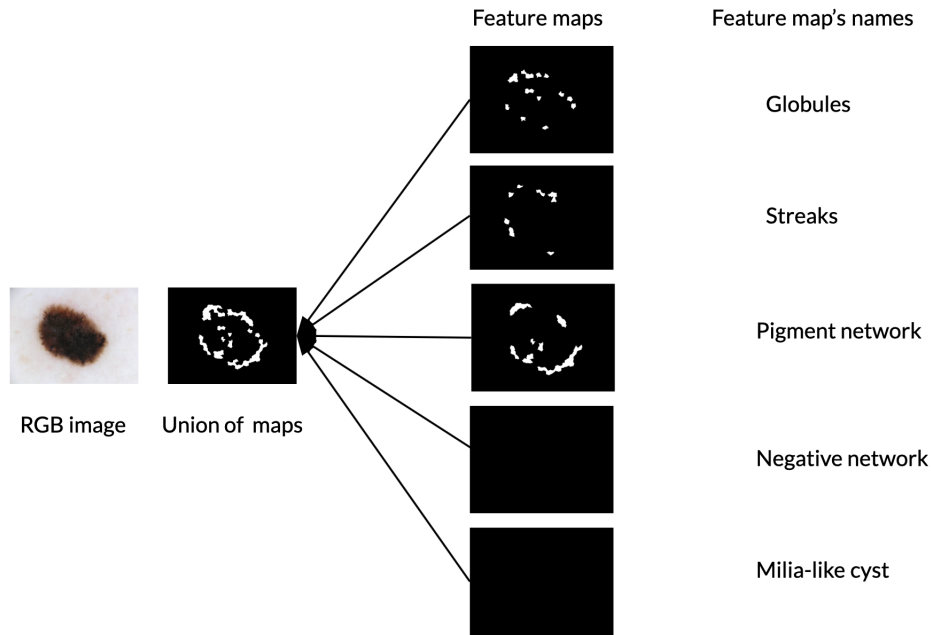


Figure 5.2: The union of the feature maps.

2019 has a sufficient number of samples for classification data, but doesn't have lesion attribute maps. So we used ISIC2018 ground truth lesion attribute maps as simulation data. ISIC data archive has a unique ID for each of the images. Due to that, the overlap checking was effortless. There is an overlap of 627 samples between the 25k+ images of ISIC2019 and the 2595 images in ISIC 2018 task 2 training data. Although the overlap will not affect the simulation, simulation test data must be overlap-free. So we kept those 627 samples out of the simulation test and the simulation validation dataset. We keep them in the simulation training data. Figure shows the distribution of the data. The images and attribute maps' original size were the same and are high-resolution images. To adapt to the network, we reduce the images and attribute maps to  $227 \times 227$ . The nearest neighbour filter is used for resizing them.

### 5.3 Simulating feedback on the full simulation set

The ISIC 2018 task 2 dataset has a total 2595 samples. Let's call this dataset SimSet. The goal is to measure if the model improves at on providing a "big" quantity of corrective data. We use a 10% SimSet for testing and a 10% SimSet for validation, and the remaining 80% samples are used for fine-tuning the model as if doctors are giving feedback to the model. We use training ground truth samples as feedback information to the model. The model learns from the feedback and update the knowledge. We had to tune the model for a maximum of 40 epochs during the simulation. From the first epoch model starts optimizing. However, between 10 to 14 epochs model gives the best performance. We use the validation data for choosing the best model. First of all, we feed one sample to the model. The task of the model is to classify and explain each instance. We also get feedback on classification and explanation output. We follow the same approach for all

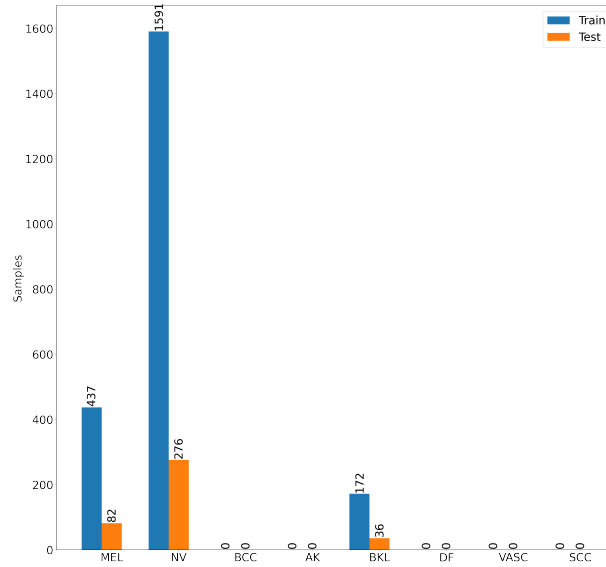


Figure 5.3: Figure shows the distribution of ISIC2018 attribute detection data.

training samples for several epochs. After each epoch, we use validation data to check the model performance by calculating the accuracy and average Jaccard index. We use classification accuracy for observing how well the model classifies. Similarly, we use the average Jaccard index to see how well the model explains its decisions. After 40 epochs, we retrieve the best model based on the highest average Jaccard index. We evaluate the model's classification accuracy and average Jaccard index on the test data.

## 5.4 Simulation on slices of data

In this approach, we mimic that dermatologists give feedback in smaller chunks, and the model learns from the small chunk of feedback. In this experiment, we divide the data into three parts. 80% of the data is for training the model, 10% data for validation, and 10% data for test. Additionally, we divide 80% of the data into 20 slices. Each of the slices contains 4% data. We train the model on all the slices iteratively. However, we select the best model from one slice using validation data and train it again on the next one. We follow the approach iteratively on all the slices. For example, the model learns from one slice at a time. During learning, the model gets one image at a time, similar to the simulation on total data. It optimizes based on combined loss for 20 epochs. We select the best model using the validation data and test explanation performance on test data. Then we take the next slice and train, validate and perform the test on the model trained on the previous slice.

## 5.5 Training on different loss functions

We use three different loss functions in the experiment: classification loss, explanation loss, and combined loss. The purpose of training the model on classification loss is to see how models perform when feedback is given only to classification data. During the training on classification, the model gets feedback only on classification output. So, the loss function only contains the classification part. We observe training and validation loss and average Jaccard index during the tuning process. On the other hand, the objective of using explanation loss is to see how the model performs when there is only explanation feedback. The explanation loss is the Jaccard loss which compares the predicted and feedback explanations. Similarly, the combined loss combines both losses by a regularization parameter  $\lambda$ , which balances the two losses. The purpose is to see how classification and explanation feedback improve model performance. The target is to compare how the model behaves when adding explanation feedback and compare the performance with only explanation feedback.

## 5.6 Experiment using unbalanced data

We know that our data is unbalanced. In real-life scenarios, data is also unbalanced. If we look at figure 5.3, we see that there are many nevus samples and a moderate number of melanoma samples. The amount of images from benign keratosis is less than the amount of images in other two classes. We perform the first simulation and test to see how our model performs in this unbalanced data. Table 5.1 shows the result from our first experiment. We compare the result with the baseline performance. Figure 5.4 shows the loss plot for three loss functions. We see that loss is decreasing slightly while the epoch is increasing. However, when we simulate the model on the explanation loss, the loss suddenly increases. Most probably, this happens due to gradient overflow [30]. Gradient overflow occurs due to unexpected high value of gradient and diverges from current minima in the loss surface. Similarly, in figure 5.5 and 5.6 we see there is not much improvement in accuracy. In the last plot, we see that the model lost the knowledge it learned previously due to the divergence from the local minima. On the other hand, we see a sudden increase in the average Jaccard score in figure 5.6 when we train the model on explanation loss. However, the average Jaccard loss dropped suddenly due to the gradient overflow. For performance measures, we look at three criteria, accuracy, average Jaccard score, and class-specific sensitivity of each class. The first column in the table 5.1 represents the baseline performance of the model. The second column described the model's performance when simulated training was only on classification loss. Similarly, the third column shows the test performance of the model trained on explanation loss. The fourth column presents the test result corresponding to the combined loss training. The classification test result tells us how good the model is in classifying images in this experiment. The average Jaccard index shows how good the model is explaining its decision. From the table, we see that the test accuracy of the model does not change in three different tests from the baseline. We see that accuracy slightly decreases while signing only explanation loss or combined loss. On the other hand, there is a huge increase in the average Jaccard index while using explanation loss or combined loss. However, if we notice the class-specific sensitivity of the model, we see that during the training the model become biased to classifying melanoma only. Mainly, the unbalanced data is biasing the model. One of the reason being that due to GPU RAM limits, we have to train with batch size 1. To check if it's the main reason, we train the model on only

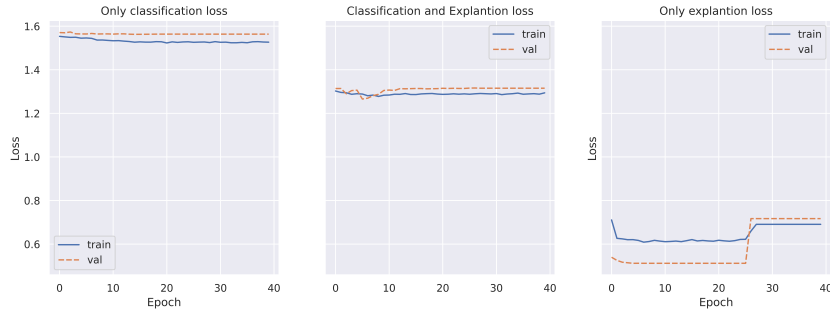


Figure 5.4: An overview of loss vs epoch plot for unbalanced simulation.

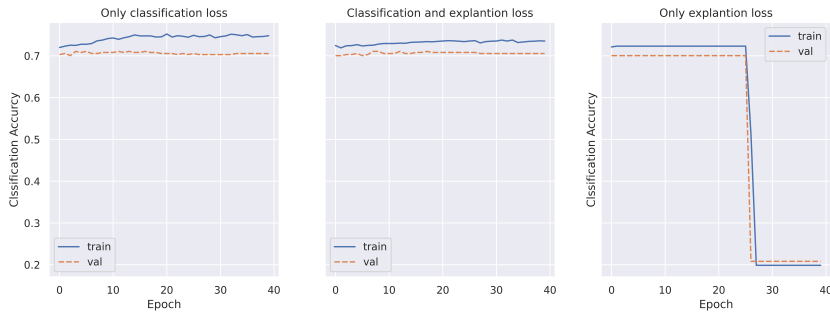


Figure 5.5: An overview of accuracy vs epoch plot for unbalanced simulation.

a balanced portion of the training data. We found that the data imbalance is biasing the model. We balance the training data and rerun the experiment to overcome this challenge.

## 5.7 Experiment using reduced balanced data

To overcome the data unbalance bias, we reduced the number of nevus samples and removed the BKL class from the simulation training data. BKL class has a few number of samples. Figure 5.7 shows the modified distribution of the data. We have only 437 melanoma. So we randomly retained 437 samples of nevus. And the red cross mark means that we removed the BKL data from the experiment. However, we keep the validation and test data unchanged for NV and MEL classes. We run the previously mentioned experiment again for only 20 epochs. The result of the investigation is presented in table 5.2. In this experiment, we only use NV and MEL classes for testing. We see that per-class sensitivity after simulation does not become zero. This means that our model does not become biased due to the class unbalance. We also see that our simulation improves the performance of the model in comparison with the baseline model. Simulation-based on explanation loss increases the accuracy from 0.71 to 0.72. The average Jaccard index increased from 0.06 to 0.274.

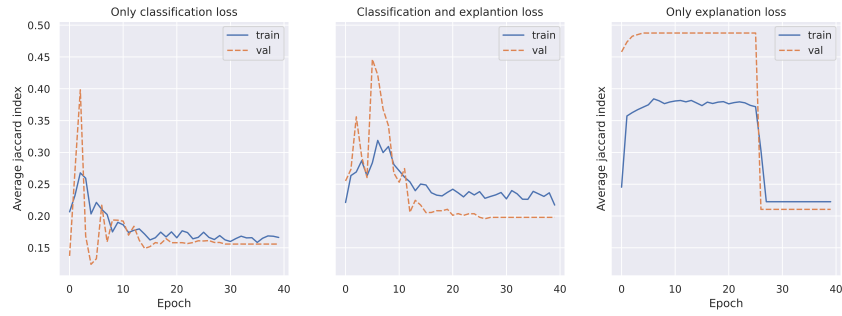


Figure 5.6: An overview of average Jaccard score vs epoch plot for unbalanced simulation.

	Baseline	Classification loss	Combined loss	Explanation loss
Accuracy	0.71	0.71	0.70	0.70
Average Jaccard index	0.10	0.165	0.473	0.447
Per-class sensitivity [NV, MEL, BKL]	[0.534, 0.772, 0.885]	[0.05, 1.0, 0.0]	[0.0, 1.0, 0.0]	[0.0, 1.0, 0.0]

Table 5.1: Test performance of model in unbalanced data. A strong bias can be noticed by observing the per-class sensitivities.

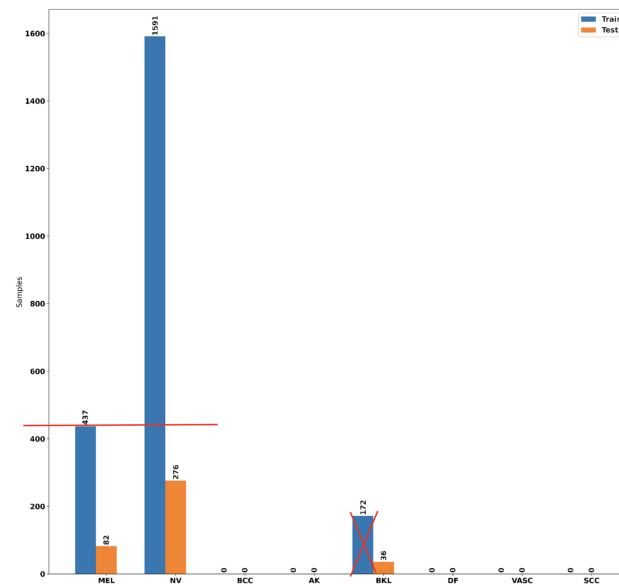


Figure 5.7: An overview of reducing samples from the original data.



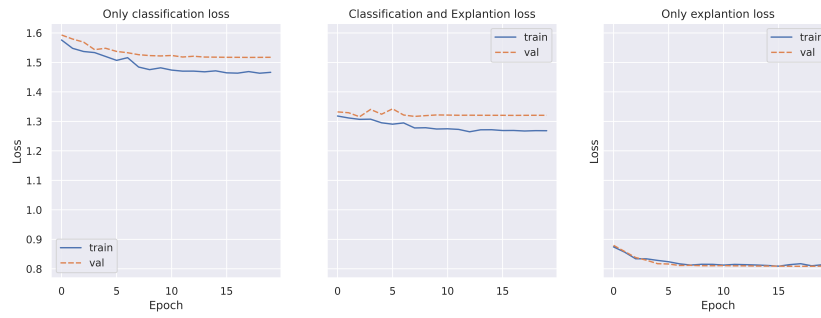


Figure 5.8: An overview of loss vs epoch plot for reduced balanced simulation.

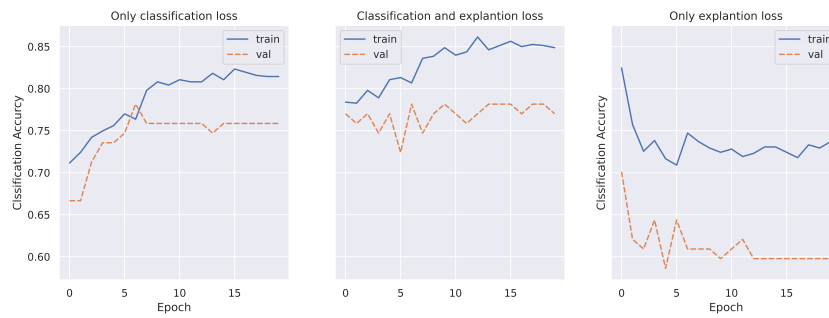


Figure 5.9: An overview of accuracy vs epoch plot for reduced balanced simulation.

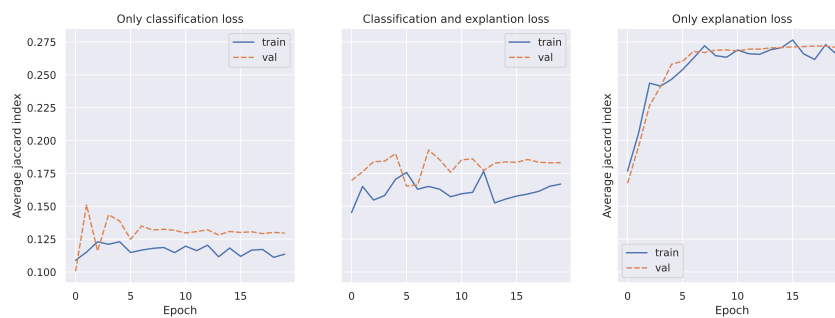


Figure 5.10: An overview of average Jaccard score vs epoch plot for reduced balanced simulation.

	Baseline	Classification loss	Explanation loss	Combined loss
Accuracy	0.71	0.69	0.66	0.72
Average Jaccard index	0.06	0.134	0.192	0.274
Per-class sensitivity [NV, MEL]	[0.521, 0.757]	[0.720, 0.764]	[0.878, 0.678]	[0.463, 0.838]
Average sensitivity	0.639	0.742	0.778	0.650

Table 5.2: Test performance of model in reduced balanced data is presented in this table.

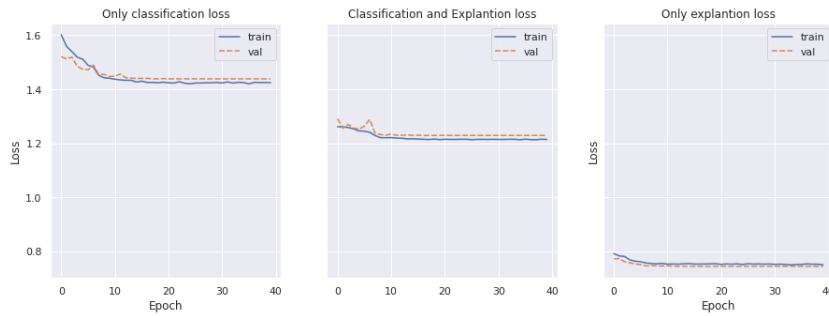


Figure 5.11: An overview of loss vs epoch plot for upsampled balanced simulation.

## 5.8 Experiment using upsampled balanced data

We equalize the data per class by upsampling in the simulation set. We know that the maximum sample belongs to the nevus class. There are 1951 samples. On the other hand, we have 437 examples that belong to MEL class, and only 172 pieces belong to BKL class. We increase the number of the MEL and BKL samples to 1951 samples by coping them randomly. As a result, we get 1951 samples from each of the classes. We simulated this augmented dataset for 40 epochs. In figure 5.11 we see the loss update for three different loss functions. With three loss functions, we see that the model learns gradually over increasing epochs. Indeed, in figure 5.12, the classification accuracy increases while we simulate the model trained with classification loss and combined loss. However, training with only explanation loss decreases the classification accuracy at earlier epochs and becomes stable later. On the other hand, if we look at the average Jaccard index plot, we see that the average Jaccard index does not improve significantly with classification loss. But with the combined loss, there is a noticeable improvement in the average Jaccard index. Similarly, there is a massive improvement in the average Jaccard index with the explanation loss. It concludes that we can improve the explainability of a CNN model by providing correct explanation feedback. Table 5.3 presents the final result of the simulation.

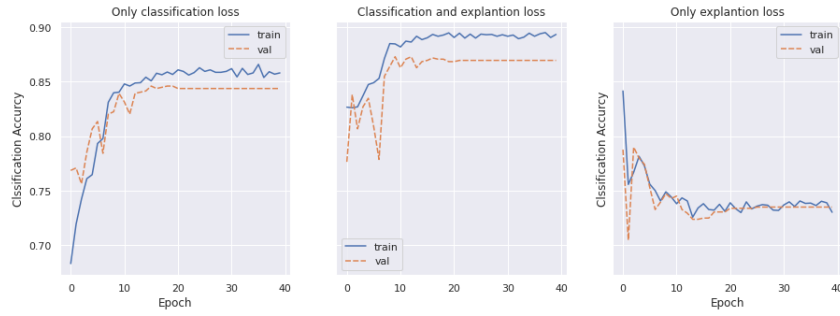


Figure 5.12: An overview of accuracy vs epoch plot for upsampled balanced simulation.

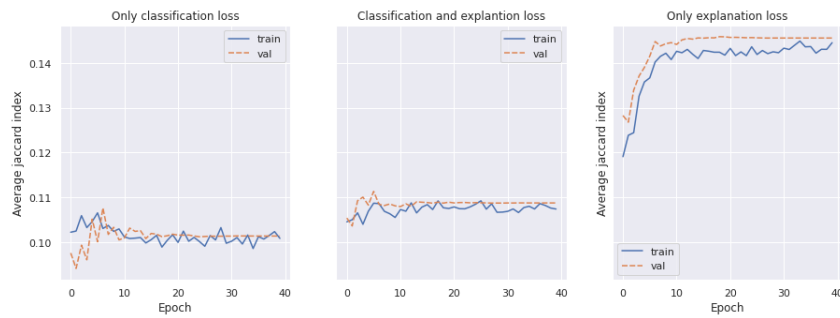


Figure 5.13: An overview of average Jaccard score vs epoch plot for upsampled balanced simulation.

Model	Train data	Loss	Test set	$\lambda$	Test acc	Sensitivity			Avg sensitivity	Avg J-mean
						MEL	NV	BKL		
Baseline	ISIC 2019	$L_{cls}$	ISIC2019 2.5K test	n/a	0.71	0.61	0.74	0.65	0.71	n/a
Baseline	ISIC 2019	$L_{cls}$	100% Sim Set	n/a	0.73	0.53	0.77	0.88	0.73	0.10
Sim-model	80% of Sim Set	$L_{cls}$	10% of Sim Set	0	0.74	0.74	0.75	0.61	0.70	0.106 (0.08)
Sim-model	80% of Sim Set	$L_{exp}$	10% of Sim Set	1	0.76	0.49	0.85	0.64	0.66	0.18 (0.13)
Sim-model	80% of Sim Set	$L_{cls}$ & $L_{exp}$	10% of Sim Set	0.3	0.64	0.78	0.59	0.72	0.70	0.127 (0.10)

Table 5.3: Complete result of simulation on upsampled samples. We see that there is improvement of average Jaccard index when the explanation loss is included in the loss computation.

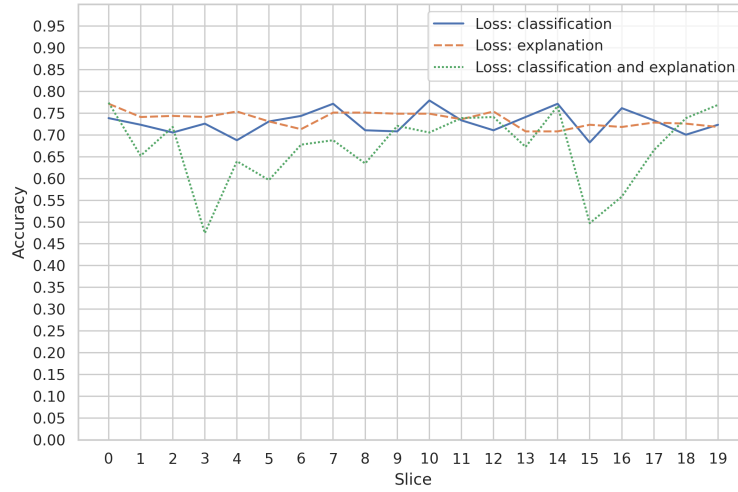


Figure 5.14: An overview of accuracy vs slices plot.

## 5.9 Result of sliced simulation

This section presents the result of sliced simulation on upsampled balanced samples for classification, explanation, and combined loss functions. Specifically, we will see how a gradual data increase improves our explainable model's performance. Figure 5.14 shows how model accuracy increases over an increasing amount of data. In the x-axis, a slice represents a chunk of data. Each chunk has 238 samples. We have a total of 20 slices, and the model is iteratively fine-tuned on all the slices. We have a total of three experiments for three kinds of loss functions. First, the classification loss function's accuracy plot shows no improvement. Secondly, the explanation loss function's accuracy looks more stable than that of the former loss function. However, we see more accuracy fluctuation when combining classification and explanation loss functions.

Similarly, figure 5.15 presents the change in the average Jaccard index over the number of slices. Looking at the classification loss function's graph, we see that the average Jaccard index is not increasing while the model sees more data (Increments of slices). There is a slight improvement when we use combined loss. However, we see moderate improvement of the average Jaccard index while using the explanation loss function.

Table 5.4 shows the summary of the sliced simulation. After tuning the model on the first slice, we see that the average Jaccard index is 0.15. However, after the 20th slice, we see that the Jaccard index increases to 0.19. On the other hand, the accuracy reduces to 0.72 from 0.77 by keeping the average sensitivity the same when using the explanation loss function. These results conclude that model explanation performance increases without reducing classification accuracy when the model gets correction feedback in smaller chunks.

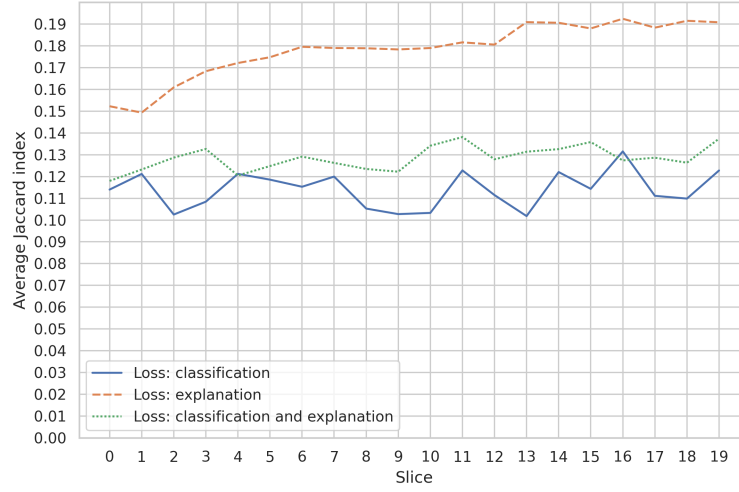


Figure 5.15: An overview of average Jaccard score vs slices plot.

Loss func- tion	Slice no	Test set	Test acc	Test avg. sensitivity	Avg. Jaccad index(sd)
$L_{cls}$	0	20% of sim- Set	0.74	0.70	0.11(0.09)
$L_{cls}$	19	20% of sim- Set	0.72	0.69	0.12(0.10)
$L_{exp}$	0	20% of sim- Set	0.77	0.66	0.15(0.12)
$L_{exp}$	19	20% of sim- Set	0.72	0.67	0.19(0.15)
$0.70 \times L_{cls} + 0.3 \times L_{exp}$	0	20% of sim- Set	0.77	0.66	0.12(0.10)
$0.70 \times L_{cls} + 0.3 \times L_{exp}$	19	20% of sim- Set	0.76	0.70	0.14(0.11)

Table 5.4: Complete result of sliced simulation. We see that there is improvement of average Jaccard index and a limited drop of accuracy when model gets more explanation feedback.

## 5.10 Training performance

Our computing machine has 64GB ram, 12 GB GeForce GTX Titan GPU, and Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz processor with 8 cores. Training Skin Care VGG16 model took around 37 hours to train 25k images. For simulation, we used the same computing machine. It took 262 minutes to 554 minutes per epoch for simulation on complete data. During the simulation using classification loss training took max 320 minutes per epoch. However, training took around 500 to 600 minutes for explanation loss and combined loss. Due to the upsampling of training data, we had about 4600 simulation samples. While we were doing the sliced simulation using the classification loss function, it took around 13 to 42 minutes per 230 samples. Contrary, simulation using the explanation and combined loss function took 25 to 55 minutes per slice. We noticed that it took longer during sliced simulation than during the initial training because simulation requires two gradient calculations. Also, the loss function is relatively complex than the initial loss function.

---

## Chapter 6

# Conclusion

We started with the evolution of machine learning and deep learning research. We specifically looked at recent findings in explainable AI and contemporary machine learning and deep learning research in skin cancer detection and analysis. We also went through the technical detail for developing an interactive and explainable model. Moreover, we saw how to build an explainable model that can interact with the simulated user and update knowledge based on user feedback. Finally, we presented the detailed results of an experiment on an interactive model in a simulated environment. This chapter presents the overall summary and future work of this thesis.

### 6.1 Summary

Machine learning is a century-old field of computer science and statistics. Many machine learning algorithms were invented even before the invention of the digital computer. However, due to the advancement of powerful computers, the research in this field accelerated. Moreover, the complexity of the algorithm also increased. Similarly, deep learning and artificial intelligence also get more focus due to the invention of powerful computing machines. Deep learning algorithms are currently considered black-box algorithms. It is challenging to decode why deep learning models make particular decisions. Due to its black-box nature, the real-life application of the deep learning model is not yet reached the application milestone. Researchers are putting hard work into developing explainable deep learning approaches. This paper specifically explained how we could decode convolutional neural networks' decisions. Several algorithms are currently available for explaining CNNs' decisions, for example, Grad-CAM, LIME, SHAP. Although they are decisive for the local explanation, they are not inbuilt with the convolution architecture. However, to ensure interactivity and explainability, we need some architecture that can explain models' decisions and also allow us to train them using user feedback. These algorithms belong to the self-explainable neural networks (SENNs) category. SENN can classify and explain at the same time.

This thesis proposed a self explainable convolutional model that does not require any encoder or decoder network. We showed that with the help of the concept of the Grad-

CAM algorithm, one could convert a black box VGG16 model to a self-explainable VGG16 algorithm. We can make any CNN architecture self-explainable using this method. This approach also ensures no decay in models' previous knowledge. We can also tune the proposed self-explainable model for a balance between classification and explanation data.

Convolution neural network shows a promising result in skin cancer classification and analysis. Researchers use CNN for the classification and segmentation of skin cancer images. We also see that some researchers explain CNNs' decision using Grad-CAM. In this researches, we see that the use of the Grad-CAM algorithm is limited to explaining models' decisions. However, this thesis brings an entirely new approach to retrain a CNN model using classification and explanation data. We can leverage this approach to retrain a model from user feedback in an interactive way. Furthermore, this thesis presented how to retrain a VGG16 skin cancer classifier in different settings using simulated user feedback. We evaluate the performance of the model in unbalanced data and balanced data. We noticed that unbalanced data is problematic to simulation. Balancing using repeated samples can overcome the challenge. Moreover, we saw that explanation, and combined feedback increases the model's explanation capability. In the following, we discuss the overall view of this research. We also evaluate the goal and discuss how we can make further improvements.

## 6.2 Evaluation

This thesis aimed to develop an interactive feedback-based tuning method for CNN that can allow dermatologists to give feedback on classification and explanation. Firstly, a new self-explainable CNN architecture paired with a training algorithm is developed by utilizing the Grad-CAM explanation technique. This algorithm does classification as well as generates the explanation for that. We can also tune the network based on classification and explanation feedback. There are three objective functions: classification, explanation, and combined while tuning the network. The combined objective function fuses the classification and explanation loss using balancing hyper-parameter. We found that simulated user feedback increased the average Jaccard index from 0.10 to 0.18. There is a slight decrease of the accuracy while an improvement in explainability. Finally, we can also summarize by stating that the experimental results agree that integrating user feedback increases model explainability.

## 6.3 Future work

In this study, we found that the explanation capability of a CNN model improves when the model is tuned using user feedback. Even though the model produces better explanations after the tuning, there is space for further improvements. There are several possible areas to work on for further improvement. For example, using an improved technique for generating saliency maps, accepting robust user feedback, and updating the loss function are possible areas for further improvement. Shinde et al. [92], propose an improved technique for generating a saliency map of a CNN model. They show that their way performs better than the Grad-CAM technique. Firstly, they generate class activation maps out of each convolution layer from the network. Then, they fuse all the class activation maps for a more meaningful saliency map. It is a robust way of generating a class activation map since it considers the importance of the output of every



convolution layer. Adding this technique will improve the performance because it will consider each of the convolution layers during the tuning of the model.

In this thesis, we had to modify the saliency map to binary maps to match the saliency maps with the ground truths. This conversion is not lossless. We reduce information on pixel-wise importance features. One probable solution is adapting saliency maps as feedback. Adapting non-binary saliency maps with the loss function will allow the model to get more accurate and piecewise information. Binary user feedback gives information on the significant region of an image. However, nonbinary user feedback can present precise knowledge about each pixel of an image. As a result, tuning on nonbinary feedback will optimize the model based on the importance of the value of each pixel. To adapt to this, we need to update the loss function. In this scenario, we can use the pixel-wise distance between the output saliency map and user feedback. Moreover, it is also essential to know which modality users prefer to provide feedback. Providing non-binary feedback can be cumbersome for humans. So, a good interface should also be designed. If we can define the user preference, we can make an encoder system that encodes user feedback to the saliency space. Moreover, it is important to see how this model performs in a real-life test environment. In this thesis, we only look at the skin cancer domain. However, we can apply user feedback-based CNN tuning to any other field. One must ensure that the task is image classification.

Moreover, in the simulation data, we found an overlap with the model's training data. However, there is no overlap between the test data of simulation and initial training data. We could use the non-overlapping simulation data, but it would reduce 25% of the samples. In real-life applications, we will not see overlapping simulation data because dermatologists will always give new data to the model instead of the past training data. So in the future, we will do the simulation again with non-overlapping data.

I would also speculate that simultaneous training for classification and explanation could become a common practice when tuning data is available. Now the problem is that the training procedure is "hacked" and requires a copy of the model at each iteration. Optimization is possible by calculating gradient for each layer of the model multiple times in each iteration, if the underlying framework (Keras, PyTorch) supports it.

---

## Bibliography

- [1] Abien Fred Agarap. 2018. Deep Learning using Rectified Linear Units (ReLU). (2018).
- [2] David Alvarez-Melis and Tommi S. Jaakkola. 2018. Towards Robust Interpretability with Self-Explaining Neural Networks. *CoRR* abs/1806.07538 (2018). <http://arxiv.org/abs/1806.07538>
- [3] Paolo Andreini, Simone Bonechi, Monica Bianchini, Alessandro Mecocci, and Franco Scarselli. 2020. Image generation by GAN and style transfer for agar plate image segmentation. *Computer methods and programs in biomedicine* 184 (2020), 105268.
- [4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 12 (2017), 2481–2495. DOI: <http://dx.doi.org/10.1109/TPAMI.2016.2644615>
- [5] Catarina Barata, Jorge S. Marques, and M. Emre Celebi. 2019. Deep Attention Model for the Hierarchical Diagnosis of Skin Lesions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- [6] Homanga Bharadhwaj. 2018. Layer-wise Relevance Propagation for Explainable Recommendations. (2018).
- [7] Lei Bi, Jinman Kim, Euijoon Ahn, Ashnil Kumar, Michael Fulham, and Dagan Feng. 2017. Dermoscopic Image Segmentation via Multistage Fully Convolutional Networks. *IEEE Transactions on Biomedical Engineering* 64, 9 (2017), 2065–2074. DOI: <http://dx.doi.org/10.1109/TBME.2017.2712771>
- [8] Laura Canalini, Federico Pollastri, Federico Bolelli, Michele Cancilla, Stefano Allegretti, and Costantino Grana. 2019. Skin Lesion Segmentation Ensemble with Diverse Training Strategies. (06 2019).
- [9] M. Emre Celebi, Quan Wen, Hitoshi Iyatomi, Kouhei Shimizu, Huiyu Zhou, and Gerald Schaefer. 2015. A State-of-the-Art Survey on Lesion Border Detection in Dermoscopy Images. 97–129. DOI: <http://dx.doi.org/10.1201/b19107-5>
- [10] Esther Chabi Adjobo, Amadou Tidjani Sanda Mahama, Pierre Gouton, and Joël Tossa. 2019. Proposition of Convolutional Neural Network Based System for Skin Cancer Detection. In *2019 15th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*. 35–39. DOI: <http://dx.doi.org/10.1109/SITIS.2019.00018>
- [11] Hao Chang. 2017. Skin cancer reorganization and classification with deep neural network. *CoRR* abs/1703.00534 (2017). <http://arxiv.org/abs/1703.00534>

- [12] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. 2018. Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (Mar 2018). DOI:<http://dx.doi.org/10.1109/wacv.2018.00097>
- [13] François Chollet and others. 2015. Keras. <https://github.com/fchollet/keras>. (2015).
- [14] Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin K. Mishra, Harald Kittler, and Allan Halpern. 2017. Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC). *CoRR abs/1710.05006* (2017). <http://arxiv.org/abs/1710.05006>
- [15] Noel C. F. Codella, Quoc-Bao Nguyen, Sharath Pankanti, David A. Gutman, Brian Helba, Allan Halpern, and John R. Smith. 2016. Deep Learning Ensembles for Melanoma Recognition in Dermoscopy Images. *CoRR abs/1610.04662* (2016). <http://arxiv.org/abs/1610.04662>
- [16] Noel C. F. Codella, Veronica Rotemberg, Philipp Tschandl, M. Emre Celebi, Stephen W. Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael A. Marchetti, Harald Kittler, and Allan Halpern. 2019. Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC). *CoRR abs/1902.03368* (2019). <http://arxiv.org/abs/1902.03368>
- [17] Abhishek Das, Ram Prasaath, and Varun Agrawal. 2017. Grad-CAM: Gradient-weighted Class Activation Mapping. (2017). <https://github.com/ramprs/grad-cam>
- [18] Sergey Demyanov, Rajib Chakravorty, Mani Abedini, Alan Halpern, and Rahil Garnavi. 2016. Classification of dermoscopy patterns using deep convolutional neural networks. 364–368. DOI:<http://dx.doi.org/10.1109/ISBI.2016.7493284>
- [19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- [20] C.P. Diehl and G. Cauwenberghs. 2003. Svm incremental learning, adaptation and optimization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, Vol. 4. IEEE, Portland, Oregon USA, 2685–2690. DOI:<http://dx.doi.org/10.1109/IJCNN.2003.1223991>
- [21] Mehwish Dildar, Shumaila Akram, Muhammad Irfan, Hikmat Ullah Khan, Muhammad Ramzan, Abdur Rehman Mahmood, Soliman Ayed Alsaieri, Abdul Hakeem M Saeed, Mohammed Olaythah Alraddadi, and Mater Hussien Mahnashi. 2021. Skin Cancer Detection: A Review Using Deep Learning Techniques. *International journal of environmental research and public health* 18, 10 (2021), 5479.
- [22] Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, and Yike Guo. 2017. Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks. In *Medical Image Understanding and Analysis*, María Valdés Hernández and Víctor González-Castro (Eds.). Springer International Publishing, Cham, 506–517.

- [23] Xue Dong, Yang Lei, Tonghe Wang, Matthew Thomas, Leonardo Tang, Walter J Curran, Tian Liu, and Xiaofeng Yang. 2019. Automatic multiorgan segmentation in thorax CT images using U-net-GAN. *Medical physics* 46, 5 (2019), 2157–2168.
- [24] Ulzii-Orshikh Dorj, Keun-Kwang Lee, Jae-Young Choi, and Malrey Lee. 2018. The skin cancer classification using deep convolutional neural network. *Multimedia Tools and Applications* 77, 8 (2018), 9909–9924.
- [25] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. (2017). <http://archive.ics.uci.edu/ml>
- [26] David Duque-Arias, Santiago Velasco-Forero, Jean-Emmanuel Deschaud, François Goulette, Andres Serna, Etienne Decencière, and Beatriz Marcotegui. 2021. On Power Jaccard Losses for Semantic Segmentation:. In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, Online Streaming, — Select a Country —, 561–568. DOI:<http://dx.doi.org/10.5220/0010304005610568>
- [27] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. 2017. Dermatologist-level classification of skin cancer with deep neural networks. *nature* 542, 7639 (2017), 115–118.
- [28] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision* 88, 2 (jun 2010), 303–338. DOI:<http://dx.doi.org/10.1007/s11263-009-0275-4>
- [29] Bhavya Ghai, Q. Vera Liao, Yunfeng Zhang, Rachel Bellamy, and Klaus Mueller. 2020. Explainable Active Learning (XAL): An Empirical Study of How Local Explanations Impact Annotator Experience. *arXiv e-prints* (Jan. 2020), arXiv:2001.09219.
- [30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [31] Manu Goyal and Moi Hoon Yap. 2017. Multi-class Semantic Segmentation of Skin Lesions via Fully Convolutional Networks. *CoRR* abs/1711.10449 (2017). <http://arxiv.org/abs/1711.10449>
- [32] David Gutman, Noel C. F. Codella, Emre Celebi, Brian Helba, Michael Marchetti, Nabin Mishra, and Allan Halpern. 2016. Skin Lesion Analysis toward Melanoma Detection: A Challenge at the International Symposium on Biomedical Imaging (ISBI) 2016, hosted by the International Skin Imaging Collaboration (ISIC). (2016).
- [33] Emma Harrington, Barbara Clyne, Nienke Wesseling, Harkiran Sandhu, Laura Armstrong, Holly Bennett, and Tom Fahey. 2017. Diagnosing malignant melanoma in ambulatory care: a systematic review of clinical prediction rules. *BMJ open* 7, 3 (2017), e014096. DOI:<http://dx.doi.org/10.1136/bmjopen-2016-014096>
- [34] Elad Hazan. 2019. Introduction to Online Convex Optimization. *CoRR* abs/1909.05207 (2019). <http://arxiv.org/abs/1909.05207>
- [35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2016). DOI:<http://dx.doi.org/10.1109/cvpr.2016.90>

- [36] Achim Hekler, Jochen S. Utikal, Alexander H. Enk, Axel Hauschild, Michael Weichenthal, Roman C. Maron, Carola Berking, Sebastian Haferkamp, Joachim Klode, Dirk Schadendorf, Bastian Schilling, Tim Holland-Letz, Benjamin Izar, Christof von Kalle, Stefan Fröhling, Titus J. Brinker, Laurenz Schmitt, Wiebke K. Peitsch, Friederike Hoffmann, Jürgen C. Becker, Christina Drusio, Philipp Jansen, Joachim Klode, Georg Lodde, Stefanie Sammet, Dirk Schadendorf, Wiebke Sondermann, Selma Ugurel, Jeannine Zader, Alexander Enk, Martin Salzmann, Sarah Schäfer, Knut Schäkel, Julia Winkler, Priscilla Wölbing, Hiba Asper, Ann-Sophie Bohne, Victoria Brown, Bianca Burba, Sophia Deffaa, Cecilia Dietrich, Matthias Dietrich, Katharina Antonia Drerup, Friederike Egberts, Anna-Sophie Erkens, Salim Greven, Viola Harde, Marion Jost, Merit Kaeding, Katharina Kosova, Stephan Lischner, Maria Maagk, Anna Laetitia Messinger, Malte Metzner, Rogina Motamedi, Ann-Christine Rosenthal, Ulrich Seidl, Jana Stemmermann, Kaspar Torz, Juliana Giraldo Velez, Jennifer Haiduk, Mareike Alter, Claudia Bär, Paul Bergenthal, Anne Gerlach, Christian Holtorf, Ante Karoglan, Sophie Kindermann, Luise Kraas, Moritz Felcht, Maria R. Gaiser, Claus-Detlev Klemke, Hjalmar Kurzen, Thomas Leibing, Verena Müller, Raphael R. Reinhard, Jochen Utikal, Franziska Winter, Carola Berking, Laurie Eicher, Daniela Hartmann, Markus Hepp, Katharina Kilian, Sebastian Kramer, Diana Lill, Anne-Charlotte Niesert, Eva Oppel, Elke Sattler, Sonja Senner, Jens Wallmichrath, Hans Wolff, Anja Gesierich, Tina Giner, Valerie Glutsch, Andreas Kerstan, Dagmar Presser, Philipp Schrüfer, Patrick Schummer, Ina Stolze, Judith Weber, Konstantin Drexler, Sebastian Haferkamp, Marion Mickler, Camila Toledo Stauner, and Alexander Thiem. 2019. Superior skin cancer classification by the combination of human and artificial intelligence. *European Journal of Cancer* 120 (2019), 114–121. DOI:<http://dx.doi.org/https://doi.org/10.1016/j.ejca.2019.07.019>
- [37] Steven C.H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. 2021. Online learning: A comprehensive survey. *Neurocomputing* 459 (Oct. 2021), 249–289. DOI:<http://dx.doi.org/10.1016/j.neucom.2021.04.112>
- [38] Andreas Holzinger, Markus Plass, Katharina Holzinger, Gloria Cerasela Crisan, Camelia-M. Pinte, and Vasile Palade. 2017. A glass-box interactive machine learning approach for solving NP-hard problems with the human-in-the-loop. (2017).
- [39] Vladimir Iglovikov and Alexey Shvets. 2018. TeraNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation. *CoRR* abs/1801.05746 (2018). <http://arxiv.org/abs/1801.05746>
- [40] M.H. Jafari, N. Karimi, E. Nasr-Esfahani, S. Samavi, S.M.R. Soroushmehr, K. Ward, and K. Najarian. 2016. Skin lesion segmentation in clinical images using deep learning. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. 337–342. DOI:<http://dx.doi.org/10.1109/ICPR.2016.7899656>
- [41] Weijia Ji, Lizhi Cai, Wei Chen, Mingang Chen, and Gang Chai. 2018. Segmentation of Lesions in Skin Image Based on Salient Object Detection with Deeply Supervised Learning. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. 1567–1573. DOI:<http://dx.doi.org/10.1109/CompComm.2018.8780745>
- [42] Yu Kai, Jia Lei, Chen Yuqiang, and Xu Wei. 2013. Deep Learning: Yesterday, Today, and Tomorrow. *Journal of Computer Research and Development* 50, 9 (2013), 1799. [https://crad.ict.ac.cn/EN/abstract/article\\_1340.shtml](https://crad.ict.ac.cn/EN/abstract/article_1340.shtml)

- [43] Jeremy Kawahara, Aicha BenTaieb, and Ghassan Hamarneh. 2016. Deep features to classify skin lesions. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*. 1397–1400. DOI:<http://dx.doi.org/10.1109/ISBI.2016.7493528>
- [44] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.
- [45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>
- [46] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist/>, (2010). <http://yann.lecun.com/exdb/mnist/>
- [47] Hang Li, Xinzi He, Feng Zhou, Zhen Yu, Dong Ni, Siping Chen, Tianfu Wang, and Baiying Lei. 2019. Dense Deconvolutional Network for Skin Lesion Segmentation. *IEEE Journal of Biomedical and Health Informatics* 23, 2 (2019), 527–537. DOI:<http://dx.doi.org/10.1109/JBHI.2018.2859898>
- [48] Hongfeng Li, Yini Pan, Jie Zhao, and Li Zhang. 2021. Skin disease diagnosis with deep learning: A review. *Neurocomputing* 464 (Nov. 2021), 364–393. DOI:<http://dx.doi.org/10.1016/j.neucom.2021.08.096>
- [49] Yuexiang Li and Linlin Shen. 2018a. cC-GAN: A robust transfer-learning framework for HEp-2 specimen image segmentation. *IEEE Access* 6 (2018), 14048–14058.
- [50] Yuexiang Li and Linlin Shen. 2018b. Skin lesion analysis towards melanoma detection using deep learning network. *Sensors* 18, 2 (2018), 556.
- [51] Haofu Liao. 2015. A Deep Learning Approach to Universal Skin Disease Classification.
- [52] Bill S. Lin, Kevin Michael, Shivam Kalra, and H.R. Tizhoosh. 2017. Skin lesion segmentation: U-Nets versus clustering. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1–7. DOI:<http://dx.doi.org/10.1109/SSCI.2017.8280804>
- [53] S. Lipovetsky and M. Conklin. 2001. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry* 17 (2001), 319–330.
- [54] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. 2017. A survey on deep learning in medical image analysis. *Medical Image Analysis* 42 (Dec. 2017), 60–88. DOI:<http://dx.doi.org/10.1016/j.media.2017.07.005>
- [55] Lina Liu, Lichao Mou, Xiao Xiang Zhu, and Mrinal Mandal. 2019. Skin Lesion Segmentation Based on Improved U-net. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*. 1–4. DOI:<http://dx.doi.org/10.1109/CCECE.2019.8861848>

- [56] Adria Romero Lopez, Xavier Giro-i Nieto, Jack Burdick, and Oge Marques. 2017. Skin lesion classification from dermoscopic images using deep learning techniques. In *2017 13th IASTED international conference on biomedical engineering (BioMed)*. IEEE, 49–54.
- [57] Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *CoRR* abs/1705.07874 (2017). <http://arxiv.org/abs/1705.07874>
- [58] Tomas Majtner, Sule Yildirim-Yayilgan, and Jon Yngve Hardeberg. 2016. Combining deep learning and hand-crafted features for skin lesion classification. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE, 1–6.
- [59] Piotr Migdal and Rafat Jakubanis. 2018. Keras vs PyTorch. <https://deepsense.ai/keras-or-pytorch/>. (2018). Accessed: 2022-01-12.
- [60] Palaniappan Mirunalini, Aravindan Chandrabose, Vignesh Gokul, and SM Jaisakthi. 2017. Deep learning for skin lesion classification. *arXiv preprint arXiv:1703.04364* (2017).
- [61] Robert (Munro) Monarch. 2021. Human-in-the-Loop Machine Learning. (2021). <https://www.manning.com/books/human-in-the-loop-machine-learning>
- [62] Mohammad Amin Morid, Alireza Borjali, and Guilherme Del Fiol. 2021. A scoping review of transfer learning research on medical image analysis using ImageNet. *Computers in Biology and Medicine* 128 (Jan. 2021), 104115. DOI:<http://dx.doi.org/10.1016/j.compbiomed.2020.104115>
- [63] Kevin P. Murphy. 2012. Machine Learning: A Probabilistic Perspective. (2012).
- [64] Zabir Al Nazi and Tasnim Azad Abir. 2020. Automatic Skin Lesion Segmentation and Melanoma Detection: Transfer Learning Approach with U-Net and DCNN-SVM. In *Proceedings of International Joint Conference on Computational Intelligence*, Mohammad Shorif Uddin and Jagdish Chand Bansal (Eds.). Springer Singapore, Singapore, 371–381.
- [65] Ho Minh Duy Nguyen, Abraham Ezema, Fabrizio Nunnari, and Daniel Sonntag. 2020. A Visually Explainable Learning System for Skin Lesion Detection Using Multiscale Input with Attention U-Net. In *KI 2020: Advances in Artificial Intelligence* (2020-09-01), Vol. 12325. Springer, 313–319. [https://www.dfki.de/fileadmin/user\\_upload/import/11178\\_KI\\_2020.pdf](https://www.dfki.de/fileadmin/user_upload/import/11178_KI_2020.pdf)[https://link.springer.com/chapter/10.1007/978-3-030-58285-2\\_28](https://link.springer.com/chapter/10.1007/978-3-030-58285-2_28)
- [66] Fabrizio Nunnari, Chirag Bhuvaneshwara, Abraham Obinwanne Ezema, and Daniel Sonntag. 2020. A Study on the Fusion of Pixels and Patient Metadata in CNN-Based Classification of Skin Lesion Images. In *Machine Learning and Knowledge Extraction (Lecture Notes in Computer Science)*, Andreas Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar Weippl (Eds.). Springer International Publishing, Cham, 191–208. DOI: [http://dx.doi.org/10.1007/978-3-030-57321-8\\_11](http://dx.doi.org/10.1007/978-3-030-57321-8_11)
- [67] Fabrizio Nunnari, Md Abdul Kadir, and Daniel Sonntag. 2021. On the Overlap Between Grad-CAM Saliency Maps and Explainable Visual Features in Skin Cancer Images. In *Machine Learning and Knowledge Extraction*, Andreas Holzinger, Peter Kieseberg, A. Min Tjoa, and Edgar Weippl (Eds.). Springer International Publishing, Cham, 241–253.

- [68] Fabrizio Nunnari and Daniel Sonntag. 2019. A CNN toolbox for skin cancer classification. (2019). <https://arxiv.org/abs/1908.08187>
- [69] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldermariam. 2019. Smooth Grad-CAM++: An Enhanced Inference Level Visualization Technique for Deep Convolutional Neural Network Models. (2019).
- [70] Şaban Öztürk and Umut Özkaya. 2020. Skin lesion segmentation with improved convolutional neural network. *Journal of digital imaging* 33, 4 (2020), 958–970.
- [71] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems* 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [72] Yanjun Peng, Ning Wang, Yuanhong Wang, and Meiling Wang. 2019. Segmentation of Dermoscopy Image Using Adversarial Networks. *Multimedia Tools Appl.* 78, 8 (apr 2019), 10965–10981. DOI:<http://dx.doi.org/10.1007/s11042-018-6523-2>
- [73] Vitali Petsiuk, Abir Das, and Kate Saenko. 2018. RISE: Randomized Input Sampling for Explanation of Black-box Models. *CoRR* abs/1806.07421 (2018). <http://arxiv.org/abs/1806.07421>
- [74] Adon Phillips, Iris Teo, and Jochen Lang. 2019. Segmentation of Prognostic Tissue Structures in Cutaneous Melanoma Using Whole Slide Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- [75] Federico Pollastri, Federico Bolelli, Roberto Paredes Palacios, and Costantino Grana. 2018. Improving skin lesion segmentation with generative adversarial networks. In *2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)*. IEEE, 442–443.
- [76] Teodora Popordanoska, Mohit Kumar, and Stefano Teso. 2020. Toward Machine-Guided, Human-Initiated Explanatory Interactive Learning. *arXiv:2007.10018 [cs]* (July 2020). <http://arxiv.org/abs/2007.10018> arXiv: 2007.10018.
- [77] Propublica. 2017. propublica/compas-analysis. (Jun 2017). <https://github.com/propublica/compas-analysis/>
- [78] Hema Raghavan, Omid Madani, and Rosie Jones. 2006. Active Learning with Feedback on Features and Instances. *J. Mach. Learn. Res.* 7 (Dec. 2006), 1655–1686.
- [79] Md Atiqur Rahman and Yang Wang. 2016. Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation. In *Advances in Visual Computing*, George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Fatih Porikli, Sandra Skaff, Alireza Entezari, Jianyuan Min, Daisuke Iwai, Amela Sadagic, Carlos Scheidegger, and Tobias Isenbergh (Eds.). Vol. 10072. Springer International Publishing, Cham, 234–244. DOI:[http://dx.doi.org/10.1007/978-3-319-50835-1\\_22](http://dx.doi.org/10.1007/978-3-319-50835-1_22) Series Title: Lecture Notes in Computer Science.



- [80] Joseph Redmon and Ali Farhadi. 2016. YOLO9000: Better, Faster, Stronger. *CoRR* abs/1612.08242 (2016). <http://arxiv.org/abs/1612.08242>
- [81] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *CoRR* abs/1602.04938 (2016). <http://arxiv.org/abs/1602.04938>
- [82] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]* (May 2015). <http://arxiv.org/abs/1505.04597> arXiv: 1505.04597.
- [83] Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. 2017. Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. *CoRR* abs/1703.03717 (2017). <http://arxiv.org/abs/1703.03717>
- [84] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and et al. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* 115, 3 (Apr 2015), 211–252. DOI:<http://dx.doi.org/10.1007/s11263-015-0816-y>
- [85] Tanzila Saba, Muhammad Attique Khan, Amjad Rehman, and Souad Larabi Marie-Sainte. 2019. Region extraction and classification of skin cancer: A heterogeneous framework of deep CNN features fusion and reduction. *Journal of medical systems* 43, 9 (2019), 1–19.
- [86] Sumit Saha. 2018. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. (2018). Accessed: 2021-09-30.
- [87] Grigory Sapunov. 2019. Hardware for Deep Learning. Part 1: Introduction. <https://blog.inten.to/hardware-for-deep-learning-current-state-and-trends-51c01ebbb6dc>. (2019). Accessed: 2022-01-02.
- [88] Md. Mostafa Kamal Sarker, Hatem A. Rashwan, Farhan Akram, Vivek Kumar Singh, Syeda Furruka Banu, Forhad U H Chowdhury, Kabir Ahmed Choudhury, Sylvie Chambon, Petia Radeva, Domenec Puig, and Mohamed Abdel-Nasser. 2021. SLSNet: Skin lesion segmentation using a lightweight generative adversarial network. (2021).
- [89] Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. 2020. Right for the Wrong Scientific Reasons: Revising Deep Networks by Interacting with their Explanations. *arXiv:2001.05371 [cs, stat]* (Jan. 2020). <http://arxiv.org/abs/2001.05371> arXiv: 2001.05371.
- [90] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR* abs/1610.02391 (2016). <http://arxiv.org/abs/1610.02391>
- [91] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. 2014. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv:1312.6229 [cs]* (Feb. 2014). <http://arxiv.org/abs/1312.6229> arXiv: 1312.6229.

- [92] Sumeet Shinde, Priyanka Tupe-Waghmare, Tanay Chougule, Jitender Saini, and Madhura Ingalkar. 2021. Predictive and discriminative localization of pathology using high resolution class activation maps with CNNs. *PeerJ. Computer Science* 7 (2021), e622. DOI:<http://dx.doi.org/10.7717/peerj-cs.622>
- [93] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. *CoRR* abs/1704.02685 (2017). <http://arxiv.org/abs/1704.02685>
- [94] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]* (April 2015). <http://arxiv.org/abs/1409.1556> arXiv: 1409.1556.
- [95] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. SmoothGrad: removing noise by adding noise. (2017).
- [96] Daniel Sonntag, Fabrizio Nunnari, and Hans-Jürgen Profitlich. 2020. The Skincare project, an interactive deep learning system for differential diagnosis of malignant skin lesions. Technical Report. *arXiv:2005.09448 [cs, eess]* (May 2020). <http://arxiv.org/abs/2005.09448> arXiv: 2005.09448.
- [97] Erich P. Stuntebeck, John S. Davis, Gregory D. Abowd, and Marion Blount. 2008. HealthSense: Classification of Health-Related Sensor Data through User-Assisted Machine Learning. In *Proceedings of the 9th Workshop on Mobile Computing Systems and Applications (HotMobile '08)*. Association for Computing Machinery, New York, NY, USA, 1–5. DOI:<http://dx.doi.org/10.1145/1411759.1411761>
- [98] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *CoRR* abs/1512.00567 (2015). <http://arxiv.org/abs/1512.00567>
- [99] Stefano Teso. 2019. Toward Faithful Explanatory Active Learning with Self-explainable Neural Nets. *Interactive Adaptive Learning* 2444 (2019), 13.
- [100] Stefano Teso and Kristian Kersting. 2019. Explanatory Interactive Machine Learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (AIES '19)*. Association for Computing Machinery, New York, NY, USA, 239–245. DOI: <http://dx.doi.org/10.1145/3306618.3314293>
- [101] Philipp Tschandl, Christoph Sinz, and Harald Kittler. 2019. Domain-specific classification-pretrained fully convolutional network encoders for skin lesion segmentation. *Computers in Biology and Medicine* 104 (2019), 111–116. DOI:<http://dx.doi.org/https://doi.org/10.1016/j.compbiomed.2018.11.010>
- [102] Andreea Udrea and G.D. Mitra. 2017. Generative Adversarial Neural Networks for Pigmented and Non-Pigmented Skin Lesions Detection in Clinical Images. *2017 21st International Conference on Control Systems and Computer Science (CSCS)* (2017), 364–368.
- [103] Halil Murat Ünver and Enes Ayan. 2019. Skin Lesion Segmentation in Dermoscopic Images with Combination of YOLO and GrabCut Algorithm. *Diagnostics* 9 (2019).
- [104] Wikipedia contributors. 2021a. Online machine learning — Wikipedia, The Free Encyclopedia. (2021). [https://en.wikipedia.org/w/index.php?title=Online\\_machine\\_learning&oldid=1039010301](https://en.wikipedia.org/w/index.php?title=Online_machine_learning&oldid=1039010301) [Online; accessed 24-January-2022].

- [105] Wikipedia contributors. 2021b. Saliency map — Wikipedia, The Free Encyclopedia. [https://en.wikipedia.org/w/index.php?title=Saliency\\_map&oldid=1049691575](https://en.wikipedia.org/w/index.php?title=Saliency_map&oldid=1049691575). (2021). [Online; accessed 24-January-2022].
- [106] Wikipedia contributors. 2022a. Shapley value — Wikipedia, The Free Encyclopedia. (2022). [https://en.wikipedia.org/w/index.php?title=Shapley\\_value&oldid=1064989172](https://en.wikipedia.org/w/index.php?title=Shapley_value&oldid=1064989172) [Online; accessed 24-January-2022].
- [107] Wikipedia contributors. 2022b. Sobel operator — Wikipedia, The Free Encyclopedia. (2022). [https://en.wikipedia.org/w/index.php?title=Sobel\\_operator&oldid=1067623546](https://en.wikipedia.org/w/index.php?title=Sobel_operator&oldid=1067623546) [Online; accessed 24-January-2022].
- [108] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2021. A Survey of Human-in-the-loop for Machine Learning. *arXiv:2108.00941 [cs]* (Nov. 2021). <http://arxiv.org/abs/2108.00941> arXiv: 2108.00941.
- [109] Wenjun Yan, Yuanyuan Wang, Shengjia Gu, Lu Huang, Fuhua Yan, Liming Xia, and Qian Tao. 2019. The domain shift problem of medical image segmentation and vendor-adaptation by Unet-GAN. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 623–631.
- [110] Jordan Yap, William Yolland, and Philipp Tschandl. 2018. Multimodal skin lesion classification using deep learning. *Experimental dermatology* 27, 11 (2018), 1261–1267.
- [111] Lequan Yu, Hao Chen, Qi Dou, Jing Qin, and Pheng-Ann Heng. 2017. Automated Melanoma Recognition in Dermoscopy Images via Very Deep Residual Networks. *IEEE Transactions on Medical Imaging* 36, 4 (2017), 994–1004. DOI: <http://dx.doi.org/10.1109/TMI.2016.2642839>
- [112] Yading Yuan, Ming Chao, and Yeh-Chi Lo. 2017. Automatic Skin Lesion Segmentation Using Deep Fully Convolutional Networks With Jaccard Distance. *IEEE transactions on medical imaging* 36, 9 (Sept. 2017), 1876–1886. DOI: <http://dx.doi.org/10.1109/TMI.2017.2695227>
- [113] Matthew D. Zeiler and Rob Fergus. 2013. Visualizing and Understanding Convolutional Networks. *CoRR* abs/1311.2901 (2013). <http://arxiv.org/abs/1311.2901>
- [114] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. 2016. Learning Deep Features for Discriminative Localization. *CVPR* (2016).
- [115] Juntang Zhuang. 2018. LadderNet: Multi-path networks based on U-Net for medical image segmentation. *ArXiv* abs/1810.07810 (2018).