Unveiling CNN Layer Contributions: Application of Feature Visualization in Medical Image Classification Tasks

Master Thesis

Master of Science in Applied Computer Science

Jonida Mukaj

May 12, 2024

Supervisor:

1st: Prof. Dr. Christian Ledig
2nd: M.Sc. Ines Rieger

Chair of Explainable Machine Learning
Faculty of Information Systems and Applied Computer Sciences
Otto-Friedrich-University Bamberg

Abstract

Exploring the application of feature visualization techniques in convolutional neural networks (CNNs), this thesis investigates the enhancement of model interpretability within the medical imaging domain, specifically using the International Skin Imaging Collaboration (ISIC) benchmark datasets from 2019 and 2020.

The objective is to leverage pre-trained CNNs such as AlexNet, VGG16, and ResNet50, for refining model interpretability. The strategy involves fine-tuning these pre-trained models, followed by employing feature visualization techniques to both the pre-trained and fine-tuned models to uncover the contributions of various layers and filters to decision-making processes.

Comparative analysis of the models fine-tuned on the ISIC 2019 and 2020 benchmark datasets highlight the different capabilities of these models, with VGG16 excelling in training accuracy and ResNet50 showing superior generalizability in unseen data. The methodology incorporates data augmentation and hyperparameter tuning to optimize model performance across both binary and multiclass classification tasks.

Feature visualizations applied to the fine-tuned models reveal distinct patterns that align with key diagnostic features of skin cancer. These visual insights not only deepen the understanding of how neural networks interpret and analyze medical images, but also contribute significantly to educational practices in the field of medical imaging. Selective visualization and layer-by-layer analysis provide insights into CNN decision-making processes by focusing on the most activated filters and tracking the evolution of features from simple to complex patterns within the models. Through these techniques, filters that respond robustly to outlier features and class-specific patterns are identified, and these filters are then utilized to apply feature visualizations. Additionally, various regularization techniques explored in feature visualizations lead to cleaner and more interpretable visualizations essential in medical contexts.

However, interpreting feature visualizations in medical contexts poses unique challenges due to the complexity of the medical images, and some quantification methods like correlation analysis and sanity checks are performed to enhance the understanding of the visual patterns identified. The educational implications of this understanding extend to the development of more robust networks, enhancing the training process to achieve peak performance, crucial for improving diagnostic accuracy, and reliability in medical applications.

Furthermore, the findings not only deepen the understanding of how neural networks process and analyze medical images, but they also highlight the importance of integrating additional interpretability methods and exploring further datasets for future research.

The code implemented for this thesis is open-source and available in a Git repository for further examination and utilization[1].

[1]`https://gitlab.studium.uni-bamberg.de/jonida.mukaj/master-thesis`

Acknowledgements

I would like to express my profound gratitude to Prof. Christian Ledig, and Ines Rieger, for their assistance in overcoming the many challenges encountered in my thesis. Finally, I would like to express my sincere gratitude to my husband, my family, friends, and all those who have provided constant support and encouragement throughout my master thesis journey.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ML	Machine Learning
CNNs	Convolutional Neural Network
FV	Feature Visualization
ReLU	Rectified Linear Unit
ISIC	International Skin Imaging Collaboration
ISIC 2019	International Skin Imaging Collaboration of the year 2019
ISIC 2020	International Skin Imaging Collaboration of the year 2020
CAD	Computer-Aided Diagnosis
BKL	seborrheic keratosis,BKL lichenoid keratosis
NV	Melanocytic nevus
MEL	Melanoma
BCC	Basal cell carcinoma
AK	actinic keratosis
SCC	Squamous cell carcinoma
VASC	ascular lesion
DF	ermatofibroma
EDA	Exploratory Data Analysis
CSV	Comma-separated values text file
PIL	Python Imaging Library
AUC	Area under Curve
FC	Fully-Connected Layer
Conv	Convolutional Layer
BCELoss	Binary Cross-Entropy with Logits Loss
AM	Activation Maximization
GAN	Generative Adversarial Network
VAE	Variational Autoencoder
MACO	Magnitude Constrained Optimization
CBR	optimization in the pixel space
OOD	Out-of-Distribution
FID	Fréchet inception distance
SGD	stochastic gradient descent optimizer
ROC	Receiver operating characteristic curve
ResNet	Residual Neural Network
VGG	Visual Geometry Group Network
ILSVRC	ImageNet Large Scale Visual Recognition Challenge

Notation

This section provides a concise reference describing notation as used in the book by Goodfellow et al. (2016). If you are unfamiliar with any of the corresponding mathematical concepts, Goodfellow et al. (2016) describe most of these ideas in chapters 2–4.

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\mathsf{\mathbf{A}}$	A tensor

Sets and Graphs

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$[a, b]$	The real interval including a and b

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
a_{-i}	All elements of vector $\boldsymbol{a}$ except for element i
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
$\mathsf{A}_{i,j,k}$	Element (i, j, k) of a 3-D tensor $\mathsf{\mathbf{A}}$
$\mathsf{\mathbf{A}}_{:,:,i}$	2-D slice of a 3-D tensor
a_i	Element i of the random vector $\mathbf{a}$

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\arg\max$	Argument of maximum value.

Datasets and Distributions

$\mathbb{X}$	A set of training examples
$\boldsymbol{x}^{(i)}$	The i-th example (input) from a dataset
$y^{(i)}$ or $\boldsymbol{y}^{(i)}$	The target associated with $\boldsymbol{x}^{(i)}$ for supervised learning

1 Introduction

The quest to understand the human brain has been a long-standing pursuit throughout human history. The rapid advancement of machine learning technologies, particularly, Convolutional Neural Networks (CNNs), has significantly impacted various fields, including medical imaging. In dermatology, the ability to accurately classify images of skin lesions as benign or malignant is crucial for early diagnosis and treatment of skin cancer. However, despite these achievements, a critical gap remains in the understanding and interpretation of how these models make their decisions. This thesis addresses this gap by focusing on feature visualization techniques to probe the inner workings of CNNs, particularly in the context of distinguishing between benign and malignant skin lesions (Olah et al., 2017).

Feature visualization stands as a relatively less explored area in the domain of medical imaging, despite its potential to enhance educational outcomes and facilitate groundbreaking research (Natekar P. and Neurosci., 2020). By delving into the visualization of what exactly CNNs "see" and "learn" from medical images, the aim was to reveal the decision-making processes of deep learning models (Nguyen et al., 2019). This approach not only aids in understanding the "neural" mechanisms underlying model predictions, but also serves to build trust among medical professionals by making these processes transparent.

The motivation behind employing feature visualization in this study is twofold. Firstly, it allows for a deeper insight into the abstract concepts and high-dimensional data representations that CNNs utilize, which are often beyond intuitive human interpretation and the aim was to provide logical structure to the model's predictions, supporting their validity and reliability. Secondly, from an educational perspective, understanding these visualizations can significantly enhance the training and development of future medical applications and build trust in using AI in medicine.

The following research questions are addressed:

1. What training strategies most effectively enhance the performance of CNN architectures when applied to ISIC benchmark image classification tasks?

2. How can feature visualization techniques be applied to different CNN architectures trained for binary and multiclass classification tasks in the ISIC benchmark dataset?

3. How does the introduction of regularization techniques affect the quality of feature visualizations in CNNs?

4. How can the impact and interpretability of feature visualizations in CNNs be quantified, particularly to enhance model understanding?

5. How can outlier-specific filters be identified through feature activation patterns? How do these patterns differ from class-specific filters when applied to binary and multiclass classification tasks?

This thesis aims to extend the understanding of feature visualizations in CNNs, particularly how these visualizations can be optimized and interpreted to enhance the training and functionality of models used in critical applications such as medical diagnosis.

2 Background

The purpose of this chapter is to provide essential background knowledge to comprehend the domain and methodology used. It begins with an introduction to medical imaging in dermatology. The fundamental principles of CNNs for image classification tasks are further discussed. Finally, an overview of different feature visualization techniques is presented.

2.1 Medical Imaging in Dermatology

Skin cancer is a concerning health issue, with a higher prevalence rate than all other types of cancer combined. In the United States (US) 9500 new cases are being diagnosed every day (SkinCancerFoundation, 2024). The deadliest form of skin cancer, melanoma, is projected to increase by 62% since 2018, with a staggering half million cases since 2040. Tragically, one person loses their life to skin cancer every four minutes, leading many dermatologists to consider this a global epidemic (MelanomaUK, 2020).

Skin cancer is primarily caused by excessive exposure to ultraviolet (UV) radiation (MelanomaSkinCancer, 2023), which can come from natural sunlight exposure (CancerResearchUK, 2023) or other sources. Furthermore, populations living in lower latitudes, where UV radiation levels are high, have been found to have higher rates of non-melanoma skin cancer (Henriksen et al., 1989). Modifiable risk factors such as poor diet (SkinCancerFoundation, 2022), alcohol consumption (WorldCancerResearch, 2022), and smoking (De Hertog et al., 2001) also contribute to the development of skin cancer.

Dermoscopy is a non-invasive imaging technique that allows doctors to view submacroscopic structures that are not visible to the naked eye (Kittler H). Although it was initially limited to the assessment of pigmented melanocytic lesions, its applications have expanded to various fields of dermatology. A melanocytic lesion refers to growths of melanocytic cells in the skin, ranging from benign freckles and moles to malignant melanoma. It is widely held that publicly accessible data regarding skin cancer is not entirely reliable, as non-melanoma instances are not generally documented in cancer registries. Furthermore, incomplete registrations can occur even when treatment is successful, and some less affluent countries may not possess cancer registries at all (WorldCancerResearch, 2022).

The increase in skin cancer cases is posing significant challenges for healthcare services worldwide. As a result, there is a growing demand for remote automated diagnosis options, especially in underprivileged areas where patients may not have access to advanced medical equipment or expert diagnoses. With the increasing use of deep learning techniques in medical image analysis, skin lesion classification has emerged as a promising research field. Medical image analysis is critical in modern medicine because diagnosing from an image alone is challenging. Computer-aided diagnosis approaches are used to gain insights into possible disease mechanisms. Nonetheless, the reliability and consistency of open datasets are fundamental factors that determine the efficacy of state-of-the-art solutions that rely on data for algorithm development.

The challenges in dermatological imaging primarily revolve around the high variability in skin lesion appearances, which can vary widely in color, shape, size, and texture (Cassidy et al., 2022). Additionally, the subtle differences between benign and malignant lesions can make accurate classification difficult even for experienced dermatologists.

Deep learning, particularly convolutional neural networks (CNNs), has emerged as a promising solution to these challenges (Chan et al., 2020). By leveraging large datasets of annotated dermoscopic images, CNNs can learn to identify patterns and features indicative of malignancy with a level of precision that matches or even surpasses human experts. This potential for deep learning to enhance the accuracy, consistency, and efficiency of skin cancer diagnosis represents a significant advancement in dermatology.

2.2 Convolutional Neural Networks for Image Classification

One of the powerful family of neural networks that are designed to learn from image data is Convolutional Neural Networks (CNNs). They are also widespread in the field of computer vision. CNNs tend to be computationally efficient, both because they require fewer parameters than fully connected architectures and because convolutional are easy to parallelize across GPU cores (Chetlur et al., 2014).

With their ability to automatically learn features from raw pixel data, CNNs have achieved remarkable success in a wide range of applications, including object recognition, face detection, and medical image analysis. CNNs consist of multiple layers including input, convolutional, activation, pooling, fully connected, and output layers. The advantages of CNNs include their ability to learn local features through hierarchical representations and focus on local regions of input data, utilize parameter sharing for translation invariance by enabling them to recognize objects regardless of their position in the image, and automatically learn relevant features directly from the data. However, limitations exist, including the requirement for a large amount of labeled training data to generalize well, computational intensity because of powerful GPU requirements, and challenges in model interpretability by making it challenging to interpret the reasoning behind their predictions (O'Shea and Nash, 2015).

For image classification, the workflow involves data collection, preprocessing, designing a suitable CNN architecture, training the model while mitigating overfitting through regularization, and evaluating performance using metrics like accuracy and precision (Sharma et al., 2018). Recent advancements in CNNs such as attention mechanisms and architectural innovations (e.g., Resnet (He et al., 2015), DenseNet (Huang et al., 2018)), aim to improve model efficiency and address issues like vanishing gradients (Pascanu et al., 2013).

Convolutional layers are an essential component of image preprocessing, utilizing filters to perform convolution operations on a matrix of pixel values. For example, when working with a black-and-white 5x5 image where pixel values are only 0 and 1, these operations are executed by sliding the filter matrix over the image and computing the dot product to detect patterns. This convolved feature reduces the image to a smaller matrix. Filters, also known as kernels, are weighted matrices that compose convolutional layers. They slide across the image from left to right, taking input only from a subarea of the image (the respective field). Filters have a width and height. These dimensions determine the size of the respective field of vision. The depth of a filter is equal to the number of filters in the convolutional layer.

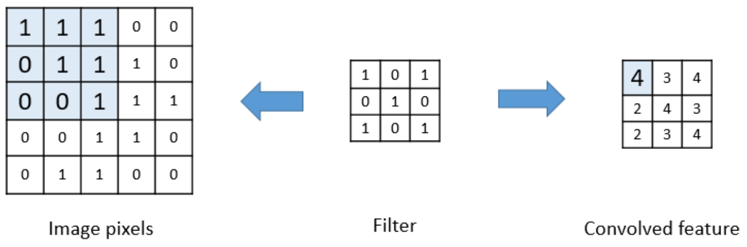


Figure 1: Operation of the Convolutional layer.

The values of the feature map can be calculated using the convolution formula:

$$G[m,n] = (f * h)[m,n] = \sum_{j}\sum_{k} h[j,k] \cdot f[m-j, n-k]$$

Where the input image is denoted by f, the filter by h, and the m and n represent the indexes of rows and columns of the outputted matrix. The computation occurs for every filter within a layer.

Filters are an important feature of image processing. They come with certain parameters that can affect the size of the output for each filter. Stride: the distance the filter moves at a time. A filter with a stride 1 will move over the input image, 1 pixel at a time. Padding: a zero-padding scheme will “pad” the edges of the output volume with zeros to preserve spatial information of the image. While incorporating stride and padding into the process, is ensured that the input and output volumes remain the same size, thus maintaining the spatial arrangement of visual features. As an image passes through more convolutional layers, more precise details activate the layer's filters. The first few convolutional layers often detect edges and geometries in the image, but in later convolutional blocks, the filter activations could target pixel intensity and different splotches of color within the image.

Activation functions determine the relevancy of a given node or pixel value in a neural network. A node relevant to the model prediction will "fire" after passing through an activation function. After a convolution operation, an activation function is applied to introduce non-linearity into the model, enabling it to complex patterns. The Rectified Linear Unit (ReLU) is a common choice due to its simplicity and efficiency (Agarap, 2019). Once the feature maps or the outputs that result from applying a filter to the image during convolution operation, are extracted, the next step is to move them to the ReLU layer. Here an element-wise operation is performed to set negative pixels to zero which will not activate or "fire". The ReLU layer introduces non-linearity to the network for each of the upcoming feature maps. The final output is a rectified feature map.

The ReLU activation function is defined as:

$$f(x) = \max(0, x)$$

The rectified feature map now goes through a pooling layer. Pooling is a down-sampling operation that reduces the dimensionality of the feature map. There are two types of pooling layers: the Max pooling layer and the global average pooling layer. Max pooling, which selects the maximum element from the region of the feature map covered by the filter, is the most common. With global averaging, the feature map's dimensions are reduced drastically by transforming the 3-dimensional feature stack into a 1- 1-dimensional vector. Given a region R, the max pooling operation P is defined as:

$$P(R) = \max_{x \in R} x \tag{1}$$

Towards the end of the network, convolutional and pooling layers are flattened into a vector and fed into fully connected layers. These layers combine features learned by previous layers across the image to identify the higher-level patterns necessary for classification (Basha et al., 2020). Given an input vector x, weights W, and bias b, the output o for a fully connected layer can be computed as:

$$o = Wx + b \tag{2}$$

In the final layer for a classification task, the softmax function is often used to convert the output scores from the fully connected layer into probabilities by taking the exponential of each output and then normalizing these values. Given a vector of raw class scores from the final fully connected layer z, the softmax function $S(z_i)$ for class i is:

$$S(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3}$$

In conclusion, CNNs have revolutionized image classification and other computer vision tasks through their unique capabilities and have seen continuous improvements and broad applications, future research is directed towards overcoming their present limitations. This includes enhancing model interpretability, reducing computational demands, and exploring the new potential in emerging technologies like augmented reality.

2.3 Feature Visualization Techniques

Deep neural networks have multiple layers that can learn hierarchical representations of the input data (Chen et al., 2021). Despite the widespread use of deep neural networks, the specific functions of each layer and their contributions to network performance remain unclear. Currently, researchers are exploring different ways to shed light on the inner workings of these complex systems, often referred to as "black boxes". In 2014, Zeiler and Fergus (2013) introduced a visualization technique that allows us to observe the feature representations learned by the intermediate layers of a convolutional network. Ben-David and Ringel (2019) have developed layer-wise loss functions that can help in understanding the internal representations of deep neural networks. This can help establish a clear role for each layer within the network. Meanwhile, Lakshminarayanan and Singh (2021) have analyzed the ReLU function's "gating" role. This function allows the pre-activation input to either pass or not, which can help in understanding the learning of active sub-networks that may contain valuable information.

Motivated by Zeiler and Fergus (2013) technique, this study is focused on understanding the role of layers in neural network performance using feature visualization by Olah et al. (2017). Several studies have explored the interpretability and understanding of Convolutional Neural Networks (CNNs) through feature visualization techniques, particularly in the context of classification tasks (Molnar, 2022).

A core component of modern CNNs is the convolutional layer (Goodfellow et al., 2016). The convolutional layer takes the output of the previous layer and applies a set of filters or features, which are also known as kernels. After this, a non-linear activation function is applied to the result. The final output of this process is called a feature map or activation map and serves as input for the next layer. CNNs are designed to learn features of varying complexity, starting with simple features such as edges and corners in the first layers and progressing to more complex patterns in subsequent layers (Olah et al., 2017).

One way to understand the features learned by a CNN is by looking at the feature maps or activation maps. Feature map visualization is a useful model interpretation technique that has been used in many medical imaging tasks, such as brain lesion segmentation (Kamnitsas et al., 2017), fetal facial plan recognition (Yang et al., 2018), classification of skin lesions (Pereira et al., 2018), and diagnosis of Alzheimer's disease (Zhang et al., 2019). Visualizing feature maps helps users connect features that humans learn to identify with features that CNNs learn.

Notably, efforts have been directed toward unraveling the complex hierarchical representations learned by CNNs to improve transparency and trust in their decision-making processes Geirhos et al. (2023). Researchers have employed various methods, such as activation maximization to generate insightful visualizations of the learned features at different layers of the network (Olah et al., 2017). Activation maximization is the task of finding an image that maximally activates a certain neuron (aka "unit", "feature", or "feature detector"), which can reveal what each neuron in a Deep Neural Network (DNN) has learned to fire in response to (i.e. which features it detects). This technique can be performed for the output neurons, such as neurons that classify types of images (Simonyan et al., 2014), and can also be performed for each of the hidden neurons in a DNN (Erhan et al., 2009), (Yosinski et al., 2015).

In the medical field, researchers have primarily used feature attribution-based methods (Singh et al., 2020). Feature visualization determines the input pattern or feature that is most exciting to a particular layer or neuron by understanding model structure and function, while attribution-based methods aim to understand model predictions (Huff et al., 2021), (Molnar, 2022).

Within the medical field, there appears to be some ambiguity surrounding the distinction between "feature visualization" and "filter and feature map visualization". While some studies have utilized the latter technique to explore the internal representations of medical image models (which is one of the methods to gain insights into the complex workings of such systems), it is not equivalent to feature visualization. The activation maximization method, as established by Olah et al. (2017), defines feature visualization.

In the medical domain, feature visualization techniques have been applied in various contexts. For example, Yu et al. (2021) utilized these techniques to visualize the progression of Alzheimer's disease across different stages using MR images. Same, Yi et al. (2017) developed a unique method for visualizing the features used by neural networks to classify benign/malignant cases, utilizing the DeepDream algorithm (Mordvintsev et al., 2015b) from the feature visualization blog by Olah et al. (2017). Another study used neuroimaging data to gain insights into the principles underlying model learning (Eitel et al., 2022).

In conclusion, while feature visualization has provided valuable insights into the functioning of deep learning models in the medical domain, the field stands at the cusp of further significant advancements.

3 Data Analysis and Data Preparation

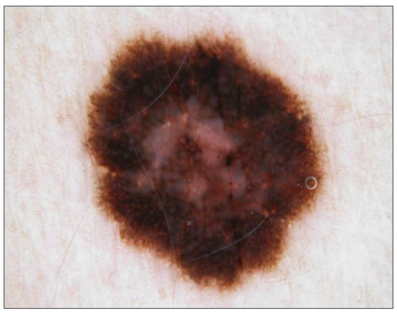
This chapter introduces the dataset information, which includes two datasets, one for binary classification and the other one for multiclass classification tasks. The main dataset for binary classification, with a total of 37648 benign and malignant samples, was used for both training and validating the models. For testing purposes in binary classification, a new testing dataset is created with samples taken from the 2016 version of the ISIC dataset. For multiclass classification, were considered data from both the ISIC 2019 and ISIC 2020 datasets, with a total of 23292 samples.

3.1 Dataset Information

The demand for remote automated diagnosis solutions for skin cancer samples is increasing, given its significant impact on global healthcare services. Skin lesion classification has become very popular in recent years due to the growing use of deep learning techniques in medical image analysis. However, since most state-of-the-art solutions rely on data, the reliability and consistency of open datasets are increasingly crucial for algorithm development. Therefore, this thesis analyzes images from the largest dermoscopic open datasets released in the past five years, the International Skin Imaging Collaboration (ISIC) datasets (Rotemberg et al., 2020).

The dataset that has been used is the medical benchmark dataset ISIC (Rotemberg et al., 2020). This dataset annotation distinguishes between benign and malignant skin lesions, with malignant skin lesions further subdivided into different malignant types. This dataset is used primarily for binary classification.

In 2020, the largest ISIC dataset was released, which includes 33,126 training images and 10,982 test images, making a total of 44,108 images (Rotemberg et al., 2021). The dataset provides ground truth data for the training set only, which shows patient ID, lesion ID, gender, approximate age, anatomical site, diagnosis, and benign or malignant status, same as the previous year. Out of the 33,126 images in the training set, there are 2,056 unique patient IDs and 32,701 unique lesion IDs. This suggests that a significant number of lesion images have been obtained from a relatively small pool of patients at different times. The test set also includes patient metadata such as ID, approximate age, anatomical site, and gender. For evaluation of the test set, the test set of the ISIC 2016 dataset, is used and it has ground truth labels. The total number of samples in this test set is 392, where 28 are malign images and 364 are benign images. This dataset is also checked for any data leakage.


Malign image

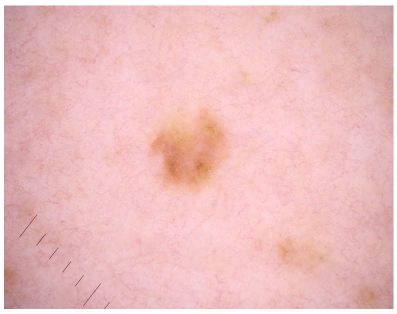

Benign image

Figure 2: Visualization of the classes in the medical benchmark dataset ISIC 2020

The medical benchmark dataset ISIC 2020 has a high level of class imbalance, with only 1.7% malign images and 83% benign images in the training set. This can result in biased models that tend to predict the majority class while ignoring the minority class observations. To overcome this problem, in-depth research was conducted to find a solution. The first winner of the Kaggle Competition (Ha et al., 2020) for

SIIM-ISIC Melanoma Classification Challenge found a solution by using both 2018 and 2019 year's data (including 2018 (Codella et al., 2018a), (Tschandl et al., 2018) and 2019 (Tschandl et al., 2018), (Codella et al., 2017), (Combalia et al., 2019)) and 2020 data. They did 5-fold cross-validation on the combined data to avoid biased models. Another technique used for handling unbalanced data is random oversampling and data augmentation. Additionally, the class weight technique (Elansary et al., 2021) was used to provide a weight for each class.

To address the imbalance issue in this master thesis, an upsampling and class weights technique are employed for the minority class, as explained further in Section 4.3.2. Specifically, only malignant images from the ISIC 2019 dataset are added to the 2020 dataset. After data preprocessing, the training set now contains 5,106 malignant samples and 32,542 benign samples. This resulted in a 15.6% increase in the proportion of malignant class images in the training set. During the training process, the class weights technique is employed by assigning higher weights to the minority classes and lower weights to the majority classes.

Table 1: Class distribution of samples in binary classification.

Diagnosis	Train before	Train	Test
Benign	32,542	32,542	364
Malign	584	5,106	28

Multiclass classification dataset information. In multiclass classification, using data from the ISIC 2019 dataset, only eight primary classes are considered. The objective of the ISIC 2020 competition was to determine whether an image depicted a benign (non-melanoma) or malignant (melanoma) condition. Within the 2019 dataset, the various classes are organized into individual folders without using explicit metadata files. Instead, the image classes are inferred directly from the hierarchical structure of the folders where the images are stored.

The dataset folders are obtained from the Kaggle competition (Prasanna, 2020) where only the training directory is considered and the validation directory is used as a test set, where the number of samples is presented in Table 2 and Fig. 3.

The ISIC challenges are playing a crucial role in the advancement of research on melanoma classification. These challenges offer digital high-resolution skin lesion image datasets that have been confirmed through biopsy, along with expert annotations and metadata from all over the world. The objective of these challenges is to promote research in the field, which will ultimately result in the development of automated Computer-Aided Diagnosis (CAD) solutions for diagnosing melanoma and other types of cancers. Additionally, this community organizes yearly skin lesion challenges to encourage wider participation of researchers for improving the diagnosis of CAD algorithms and to raise awareness about the increasing problem of skin cancer (Codella et al., 2018b).

Table 2: Class distribution of samples in multiclass classification.

Diagnosis	Train	Test	Diagnosis
NV	10,979	931	Melanocytic nevus
MEL	3,812	350	melanoma
BCC	2,820	253	Basal cell carcinoma
BKL	2,215	206	BKL lichenoid keratosis
AK	716	76	Actinic keratosis
SCC	541	45	Squamous cell carcinoma
VASC	202	27	Vascular lesion
DF	206	22	Dermatofibroma

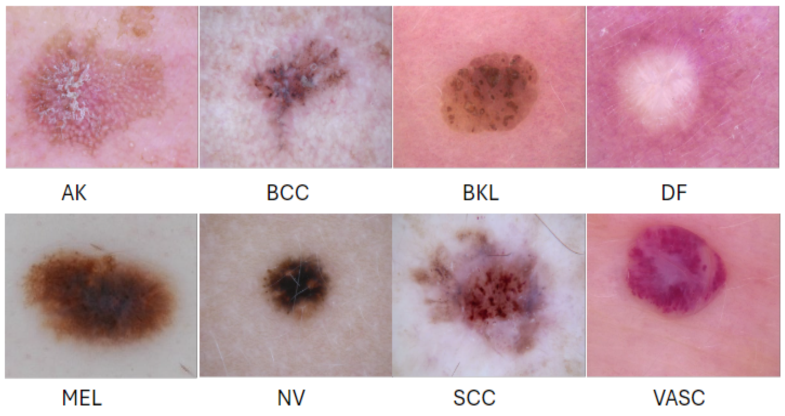


Figure 3: Multiclass dataset samples.

3.2 Outlier Analysis

The training sets of ISIC images have some outlier observations that might hinder the model's performance. These were initially introduced in the paper analyzing ISIC image datasets (Cassidy et al., 2022). The observations include:

1. Outliers
   (a) cropped lesions
   (b) dermoscopy measurement overlay that obscures the lesion
   (c) hair that obfuscates the lesion
   (d) clinical markings
   (e) circular size reference stickers
   (f) physical ruler
   (g) immersion fluid that distorts the lesion
   (h) air pocket in the immersion fluid

For more information see Fig. 4 below.

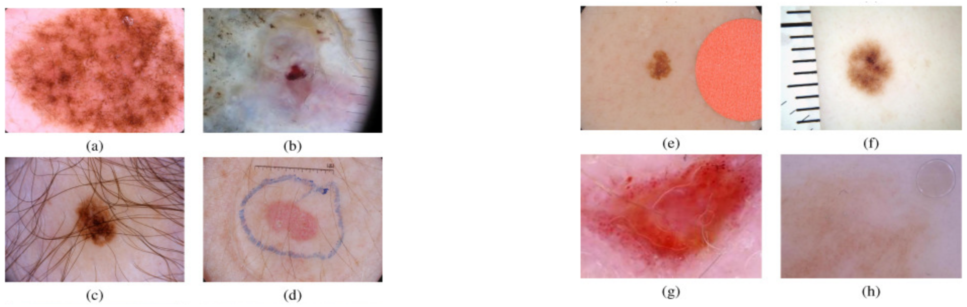


Figure 4: Outliers of the training set

3.3 Data Preprocessing

Data preprocessing is a crucial step in the field of machine learning. It involves cleaning and preparing raw data before it is fed into a model. This process helps improve the quality of the data by removing errors, inconsistencies, and outliers. The accuracy of the model can be negatively affected by several factors, which can make it less effective. Preprocessing the data can help to reduce the amount of noise, such as irrelevant or redundant features, which can improve the efficiency of the learning algorithm. Ultimately, this can result in better predictions and outcomes.

Exploratory Data Analysis (EDA) refers to the initial analysis and findings that are performed on data sets, usually in the early stages of an analytical process (Mahalakshmi and Sujatha, 2023). Some experts describe it as a preliminary investigation aimed at understanding the nature of the data, its characteristics, and how it can be leveraged for statistical analysis. EDA is often a precursor to other kinds of statistical work with data.

The CSV file of the training set for binary classification provides information about each image, including ground truth labels for the training set. These labels include image name, patient ID, sex, age, anatomy, diagnosis, benign/malignant classification, and target. The "target" column is used to distinguish between malign and benign classes for binary classification. To address the issue of class imbalance, external malign samples from the 2018 and 2019 ISIC benchmark datasets were used, as explained in section 3.1. As a result, the metadata file for the training set now includes only the following columns: image name, lesion ID, sex, age, anatomy, and target. Similar to the binary classification dataset, for multiclass classification, the metadata file of the training set provides information about each image, including ground truth labels for all eight classes. The columns are image name, patient ID, sex, age, anatomy, diagnosis, and benign/malignant classification. The column titled "diagnosis" is used to distinguish between various categories in the multiclass classification.

EDA was conducted for the training set, both for binary and multiclass, to analyze the images based on the correlation of their metadata entries. The analysis focused on correlating target classes based on age, gender, and anatomy entries. Also, the age distribution of target types was checked, as well as the gender and anatomy split by target and diagnosis variable.

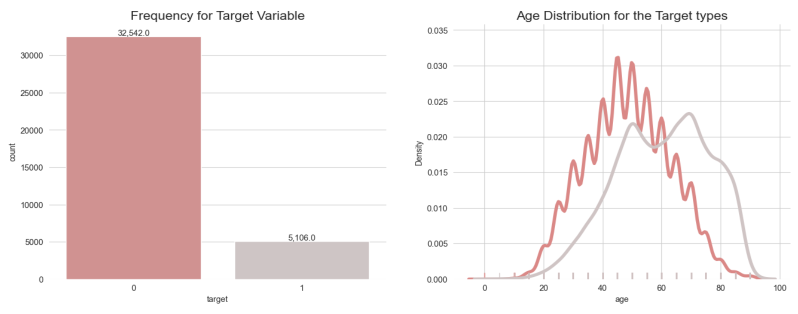


Figure 5: Frequency and age distribution of the "target" column (binary)

According to Fig. 5, there is a clear increase in the number of malignant samples, in comparison to the initial ISIC 2020 benchmark dataset as described in Table 1. The second graph, which shows the age distribution for target values, indicates that the benign samples follow a normal distribution. On the other hand, the malignant samples are slightly skewed to the left, with the peak toward higher age values.

After analyzing the "target" variable, it was found that the dataset has more male cases than female cases. The distribution of benign and malignant cases is similar for all anatomical entries, but the torso has a larger sample size. For more detailed information, please refer to the images shown in Fig. 6 and Fig. 7. The target variable assigned to value "1" is associated with the malignant class and the target variable assigned to value "0" is associated with the benign class.

A similar analysis is conducted with the metadata file of the multiclass dataset, correlating the "diagnosis" column with other metadata variables such as gender and age.

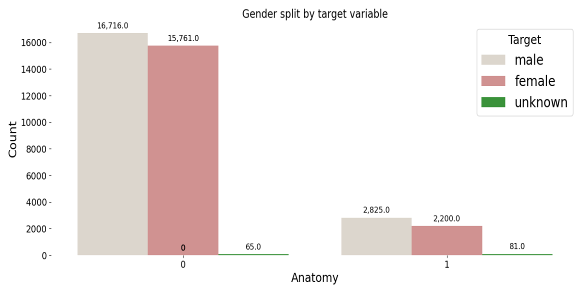


Figure 6: Gender split by “target” column

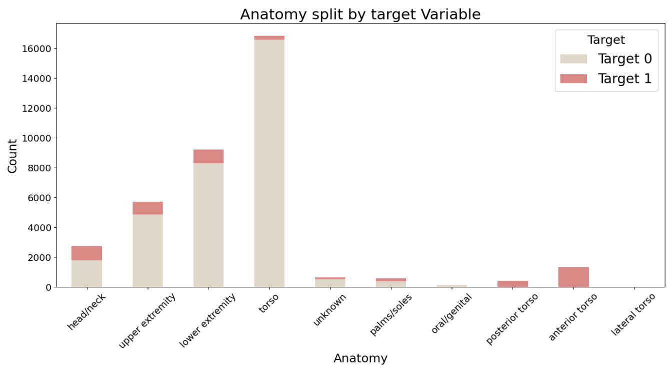


Figure 7: Anatomy split by “target” column

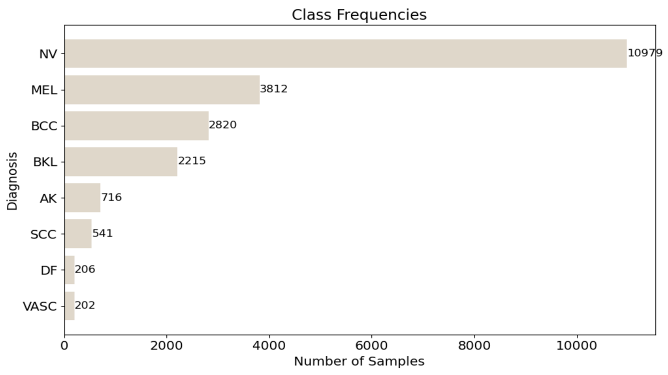


Figure 8: Frequency distribution of the “diagnosis” column (multiclass)

Based on the chart shown in Fig. 8, there is a significant difference in the number of samples across different classes. The majority of the images belong to the class labeled *NV* (nevus), while some classes have only a few images.

3.4 Data Augmentation

This master thesis incorporated a comprehensive preprocessing and data augmentation pipeline to enhance the performance and generalization capabilities of the deep learning model. The preprocessing steps begin with loading the dataset. This function constructs image paths based on a provided metadata DataFrame and project directory. This ensures that the dataset is well-organized and ready for subsequent training.

The core of the preprocessing pipeline lies in the ISIC Dataset class. Here, images are loaded and transformed to prepare them for model input. Initially, images are read as tensors. They are then converted to PIL Images, which is a necessary step for applying various PyTorch transformations. By default, the images are resized to (224, 224) and their pixel values are normalized using predefined mean and standard deviation values for binary classification. This ensures that the input data is standardized, which helps with stable and consistent model training. This preprocessing step was crucial because the image shapes were quite different as described in Fig. 9, where only 100 different images are checked.

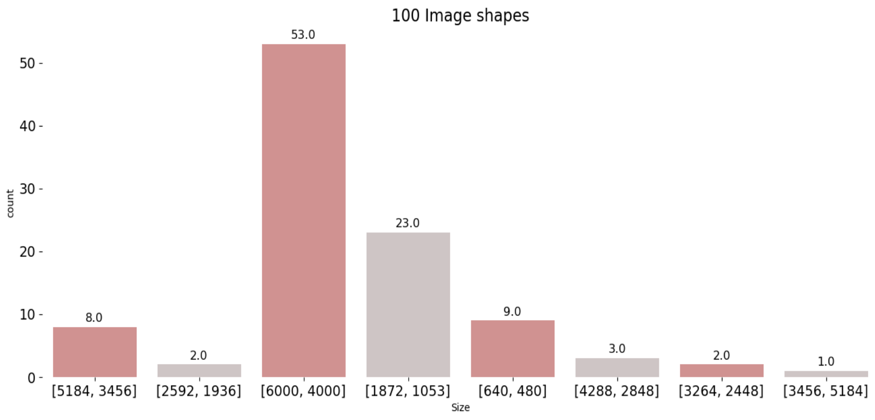


Figure 9: 100 image shapes for the training dataset.

A key element of this preprocessing strategy is the use of data augmentation techniques. A function defines separate transformation pipelines for training and validation datasets. During training, various augmentations are applied to introduce diversity to the dataset. These include random resized crops, affine transformations (rotation, shear, scaling), and random flips (horizontal and vertical). Additionally, color jitter is applied to further enhance the model's ability to generalize to various image conditions.

Based on thorough research on data augmentation techniques suitable for medical images of ISIC datasets, the paper by Perez et al. (2018) recommends the best techniques primarily using ISIC 2017 dataset images. Therefore, the above data augmentations were chosen to use for the ISIC 2020 dataset benchmark. They conclude that the best augmentation scenario, which involves combining geometric and color transformations, outperforms the top-ranked AUC values for the ISIC Challenge 2017 without any additional data. To further enhance performance, one can fine-tune hyperparameters and use model ensembling techniques.

The best augmentation scenario involves the following steps:

1. Random Crops: Randomly crop the original image. The crop has $0.4 - 1.0$ of the original area and 3/4 - 4/3 of the original aspect ratio.
2. Affine: Rotate the image by up to 90 degrees, shear by up to 20 degrees, and scale the area by [0.8, 1.2]. New pixels are filled symmetrically at the edges. This can reproduce camera distortions and create new lesion shapes.
3. Flips: Randomly flip the images horizontally and/or vertically.
4. Saturation, Contrast, Brightness, and Hue: Modify saturation, contrast, and brightness by random factors sampled from a uniform distribution of $[0.7, 1.3]$, simulating changes in color due to camera settings and lesion characteristics. Also, shift the hue by a value sampled from a uniform distribution of $[-0.1, 0.1]$.

For the validation dataset, milder augmentations are employed to simulate real-world scenarios while evaluating the model's performance. This approach balances the need for the model to learn robust features during training with augmented data while ensuring its adaptability to non-augmented validation data.

The use of PyTorch's DataLoader class facilitates efficient batch processing during both training and validation. This preprocessing and data augmentation strategy collectively contributes to the model's ability to learn diverse features, handle variations in input data, and ultimately improve its overall performance on the target task, such as image classification in this master thesis.

Various data augmentation techniques were used during the training of the model using medical skin cancer images. These techniques were necessary to account for variations in skin tones, lighting conditions, and image sizes. One of the critical considerations was addressing significant imbalance ratios. However, a decision was made not to use the hair removal augmentation technique deliberately. Although it could potentially improve the model's performance, the primary objective of this thesis is to evaluate the learned features of the model. It is important to test the model's ability to distinguish between outliers and skin marks.

4 Methods

This chapter provides an insight into the methodology used in this study. First, the software tools and libraries that were employed for the study are presented. Then, the model architectures of the three models are presented. Subsequently, each experimental training methodology used in the study is explained. The chapter concludes with an explanation of the feature visualization technique applied, regularization, and quantification methods.

4.1 Technical Setup

This master thesis is implemented in Python on a local and remote machine with the following hardware:

The remote server was used for training and validation purposes:

- Processor: Intel(R) Xeon(R) Silver 4309Y CPU @ 2.80GHz
- RAM: 16GB
- Operating System: x86-64 GNU/Linux

The local machine was mostly used for testing and applying feature visualizations:

- Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 2592 Mhz, 6 Core(s), 12 Logical Processor(s)
- RAM: 24GB
- Operating System: Windows 11 Enterprise (64-bit operating system, x64-based processor)

The thesis utilized several open-source libraries that are widely accepted. The Torchvision package provided the machine learning models and image transformation capabilities (Marcel and Rodriguez, 2010). Pytorch (Paszke et al., 2019), a widely deep learning framework, was employed for building and training the neural networks. Numpy was used to perform numerical calculations and array operations (Harris, 2020). Pandas was used for efficient analysis and data manipulation (Wes McKinney, 2010). The creation of personalized data visualizations was accomplished using the Matplotlib (Hunter, 2007) and Seaborn (Waskom, 2021) libraries. Furthermore, the Torch Dream library is used to apply feature visualization techniques (Deb, 2021). The original library for research in neural network interpretability and feature visualization is called Lucid (Christopher Olah, 2017). It is a collection of infrastructure and tools built using TensorFlow. The version of Lucid adapted for PyTorch is Lucent by Lim Swee Kiat (2020) and Torch Dream (Deb, 2021).

4.2 Model Architecture

In this master thesis, there are used three different model architectures: AlexNet, VGG16, and Resnet50 (Krizhevsky et al., 2017). The decision to choose those models was made based on the best-performing models on medical images. Based on the ISIC 2020 benchmark dataset leaderboard, the most used model is EfficientNet, but also Resnet50 has given high accuracy (ISICLeaderboard).

4.2.1 ResNet50 Architecture

A Residual Neural Network with 50 layers or ResNet50 is a type of deep CNN architecture developed by Microsoft Research in 2015. It's a variant of the ResNet architecture with 50 layers, which uses residual connections to learn a set of residual functions that map the input to the desired output (He et al., 2015). The architecture is divided into four main parts: convolutional layers, identity block, convolutional block, and fully connected layers. The convolutional layers extract features from the input image, while the identity block and convolutional block process and transform these features. Finally, the fully connected layers are used to make the final classification.

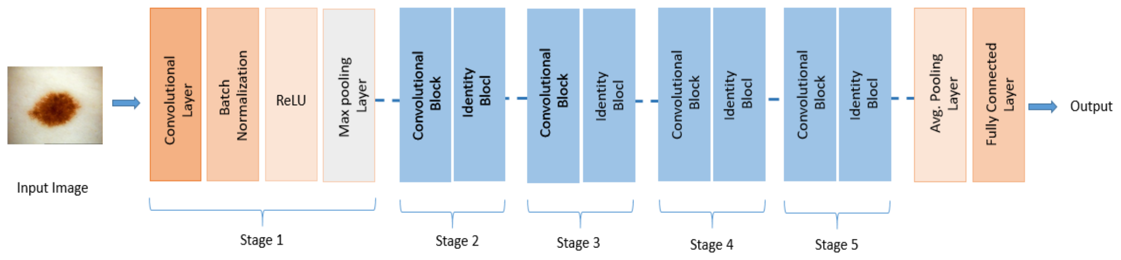


Figure 10: ResNet50 Architecture

ResNet50 has been trained on large datasets and archives state-of-art results on several benchmarks. It has been trained on the ImageNet dataset, which contains over 14 million images and 1000 classes, where ResNet50 achieved an error rate of 22.85% (Deng et al., 2009).

One key feature of the ResNet50 architecture and all residual networks is the skip connection, also known as a residual connection. Skip connections address the problem of vanishing gradients that occurs when training deep neural networks, where the gradient of the parameters in the deeper layers becomes very small, making it difficult for those layers to learn and improve. This problem is solved by the ResNet50 model, by allowing the information to flow directly from the input to the output of the network, bypassing one or more layers. In ResNet50, skip connections are used in the identity block and convolutional block.

The ResNet50 model has been pre-trained on the ImageNet dataset to serve as a feature extractor. It can capture detailed representations within the input images. In this master thesis, a new linear layer with an output size of 1 is used to replace the final fully connected layer of the pre-trained ResNet50 model. This strategic modification is governed by the binary classification nature of the skin cancer classification task, necessitating a singular output unit for discriminative analysis. For the multiclass classification task, the final layer of ResNet50 is replaced by a layer with an output size of eight, because there are eight different classes as described earlier in the Section 3.1.

4.2.2 VGG16 Architecture

VGG16 is an object detection and classification model that can classify 1000 images of 1000 different categories with 92.7% accuracy (Mascarenhas and Agarwal, 2021).

It is one of the popular models for image classification and is easy to use with transfer learning.

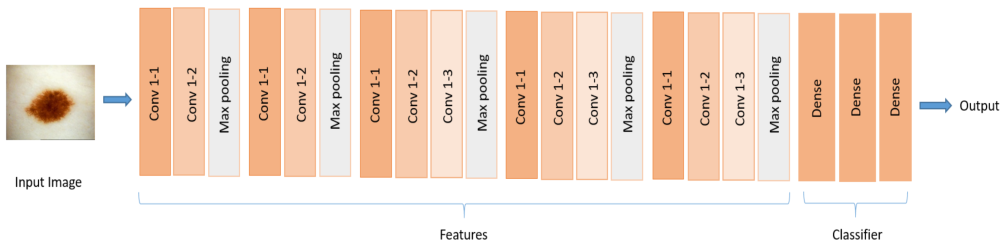


Figure 11: VGG16 Architecture

The term “16” in VGG16 denotes the number of layers in the architecture that have weights. Although there are 21 layers in total, including 13 convolutional layers, five max-pooling layers, and three dense layers, only 16 of these layers have learnable parameters. VGG16 accepts an input tensor size of 224 by 224 with 3 RGB channels. One of the most unique features of VGG16 is that it employs convolutional layers with 3x3 filters and stride 1, as well as always using the same padding and max pool layer with a 2x2 filter and stride 2. The structure of the convolutional and max pool layers remains consistent throughout the entire architecture. The first convolutional layer ($Conv1$) has 64 filters, $Conv2$ has 128 filters, $Conv3$ has 256 filters, and both $Conv4$ and $Conv5$ have 512 filters. After a sequence of convolutional layers, three fully-connected (FC) layers follow. The first two have 4096 channels each, while the third performs 1000-way the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification, and therefore contains 1000 channels (one for each class) (Russakovsky et al., 2015). The final layer is the soft-max layer.

To preserve the features learned by the pre-trained model during initialization, the classifier’s parameters are frozen. Subsequently, the final fully connected layer is modified to accommodate the binary classification task. The original layer is replaced by a new linear layer with a single output, reflecting the binary nature of the classification. For the multiclass classification task in VGG16, the final layer is replaced with an output layer of size *eight*, similar to ResNet50 in Section 4.2.1.

4.2.3 AlexNet architecture

AlexNet, which employed an eight-layer CNN, established a milestone in the field of computer vision by winning the ILSVRC Challenge in 2012 with a substantial lead over its competitors Russakovsky et al. (2013). This breakthrough network demonstrated for the first time that learned features can surpass manually designed features, which was the previous paradigm in computer vision.

The first layer of AlexNet uses an 11x11 convolution window shape to capture larger objects with more visual details in ImageNet data (Deng et al., 2009), which is eight times larger than the MINST dataset Yann. In the second layer, the convolution window shape is reduced to 5x5, and then 3x3. The network adds max-pooling layers after the first, second, and fifth convolutional layers with a window shape of 3x3 and a stride of 2. AlexNet has ten times more convolution channels than LeNet (Lecun et al., 1998). After the final convolutional layer, there are two FC layers with 4096 outputs, which require nearly 1GB of model parameters. AlexNet replaced the sigmoid activation function with a simpler ReLU activation function, which is computationally simpler as it does not have the exponentiation operation found in the sigmoid activation function.

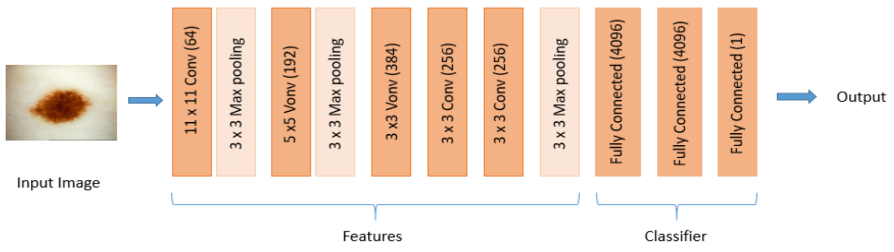


Figure 12: Alexnet Architecture

4.3 Training Methodology

4.3.1 Loss Functions and Model Initialization

The primary objective is to demonstrate the adaptability of pre-trained models for specialized tasks through fine-tuning, which involves adjusting the final layers to match the target number of classes. The core adaption requires modifying the final fully connected layer of the specific model's classifier to accommodate eight output classes, corresponding to the particular classification task available. This adjustment is achieved by replacing the original layer with a new linear layer, where the number of classes is set to eight to match the multiclass classification task and is set to one to match the binary classification task.

For the binary classification task, all three models were initialized by freezing the classifier's parameters to retain the features learned by the pre-trained model. Then, the final fully connected layer is modified to match the binary classification task. The original linear layer is replaced with a new one, which has a single output, reflecting the binary nature of the classification.

A dropout layer, with a rate of 0.1, is applied to improve model generalization and reduce the risk of overfitting to the majority class or any specific patterns in the training data that do not generalize well to the validation dataset.

The selection of an optimizer is crucial, and the Adam optimizer is chosen for its effectiveness in minimizing the loss function (Kingma and Ba, 2017). The learning rate, a significant hyperparameter, is explicitly set to a value that indicates the step size in the parameter space during optimization. In the case of Adam, the default weight decay parameter is left at zero, and its adaptive momentum parameters facilitate faster convergence. Furthermore, a learning rate scheduler - the ReduceOnPlateau (2023) scheduler - is included. This scheduler dynamically adjusts the learning rate based on the model's observed performance. Using a patience of 1, the scheduler intervenes if no improvement is detected for a single epoch, reducing the learning rate by a factor of 0.4. This adaptive strategy enhances convergence and reduces the risk of overshooting optimal parameter values. Lastly, the loss function used is the Binary Cross-Entropy with Logits Loss (BCELoss, 2023). This criterion combines a sigmoid activation function with the binary cross-entropy loss, ensuring numerical stability during training (Ding et al., 2018). Its application aligns with the binary nature of the classification task, where the model output is interpreted as logits to facilitate stable and efficient computation.

For multiclass classification, the learning rate scheduler employed is similar to binary classification, the ReduceLROnPlateau scheduler, which adjusts the learning rate when a metric has stopped improving, specifically reducing the learning rate by a factor of 0.2 after a patience period of two epochs without performance improvement. This strategy helps in fine-tuning the model by gradually decreasing the learning rate to refine the model weights. The loss function used is cross-entropy, which combines a softmax layer and the cross-entropy loss in a single class (Ding et al., 2018). The function also allows for the specification of class weights, enabling the model to pay more attention to samples from underrepresented classes, thus addressing class imbalance.

4.3.2 Handling Class Imbalance

As described earlier in Section 3.1, an upsampling technique is used to handle the class imbalance for binary classification. This technique is employed for the minority class. Specifically, only malignant images from the ISIC 2019 dataset are added to the 2020 dataset. After preprocessing and data augmentation, the proportion of malignant class images in the training set increased by 15.6%.

Despite using the upsampling technique for the minority class, the number of malignant class samples still only accounts for 15.6% of the entire training dataset. During the model evaluation, the confusion matrix for the validation dataset does not classify the minority samples accurately. To address this issue, a class weighting technique was employed.

Class weighting is a technique employed to address the imbalance by adjusting the importance of classes proportional to their representation in the data. This is achieved by assigning higher weights to the minority classes and lower weights to the majority classes during the training process. The underlying goal is to increase the cost of misclassifying the minority class, compelling the model to pay more attention to correctly predicting these classes.

When performing binary classification, class weights are calculated using a function that determines weights inversely proportional to the frequency of each class in the input data. These weights can then be utilized in the loss function of the classifier to give greater importance to the minority class. Meanwhile, in multiclass classification, the weights are assigned manually where the value 0.9 is assigned to the class with the least number of samples and the value 0.2 to the class with the most number of samples. Class weighting directly tackles the issue of imbalance by making the model more sensitive to the minority class, and enhancing its ability to recognize and correctly predict instances of this class.

The strategic combination of class weighting together with the dropout layer mentioned in Section 4.3.1, aims to yield a more balanced and robust model capable of effectively identifying both majority and minority class instances, leading to improved overall performance on imbalanced datasets.

4.4 Feature Visualization

4.4.1 General Concepts

Deep neural networks can learn high-level features in the hidden layers (Molnar, 2022). One of the primary benefits of this approach is its ability to mitigate the need for extensive feature engineering. With convolutional neural networks, the image is fed into the network in its raw form (pixels) and the network transforms the image multiple times. First, the image goes through many convolutional layers where the network learns new and increasingly complex features in its layers. Then, the transformed image information passes through the fully connected layers before being classified or predicted.

Convolutional neural networks can learn abstract features and concepts from raw image pixels. Feature Visualization is a technique used to visualize the learned features by performing activation maximization (Nguyen et al., 2019).

The first few layers of a CNN are responsible for identifying basic image features such as edges and simple textures. Moving deeper into the network, the subsequent convolutional layers start learning more complex textures and patterns. Finally, the last convolutional layers learn to identify objects or parts of objects. The fully connected layers, which come after the convolutional layers, learn to connect the high-level features extracted by the convolutional layers to the specific classes or labels to be predicted.

Feature visualization (FV) is an optimization problem in mathematical terms, where the input that maximizes the activation of a neural network unit is found (Olah et al., 2017).

In the context of neural network interpretability, the term “unit” is associated with various levels of the network’s architecture, ranging from the granular individual neurons to the broader channels, layers, and output probabilities corresponding to classification decisions, as presented in Fig. 14. Neurons represent the most fundamental units, offering the richest detail through feature visualizations. However, a specific layer can contain thousands of neurons, often reaching into the millions, and

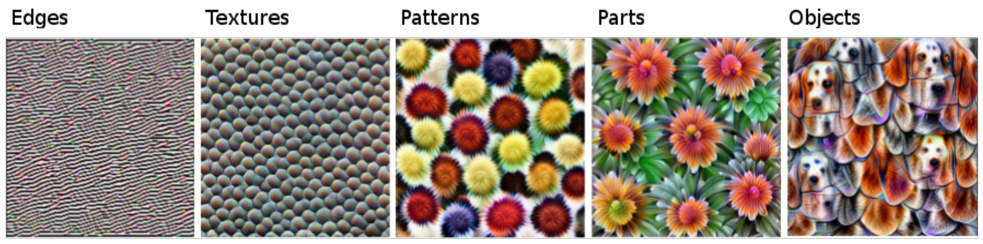


Figure 13: Features learned by a convolutional neural network (Inception V1) trained on the ImageNet data (Molnar, 2022).

this poses a practical challenge to individual analysis due to the extensive computational resources and time required.

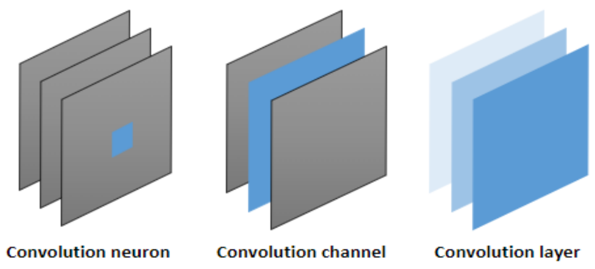


Figure 14: Different units of neural network

Feature visualization (FV), therefore, frequently targets channels - also known as feature maps - which are intermediate representations capturing the presence of specific features within the input. Channels provide a more trackable means of understanding the network's feature extraction process. Moreover, by aggregating feature visualizations at the layer level, one can observe the combination of features that contribute to the network's decision-making process. This layered approach is presented in the DeepDream algorithm developed by Google, which iteratively enhances patterns in an image to produce dream-like visuals, thereby offering a creative visualization of layer activations (Mordvintsev et al., 2015a).

4.4.2 Activation Maximization

Assuming that the weights of the neural network are already fixed, it means that the network has already been trained. The primary goal is to find a new image that maximizes the activation of a particular neuron, which is typically measured by its mean activation.

$$\text{img}^* = \underset{\text{img}}{\arg\max}\ h_{n,x,y,z}(\text{img})$$

The function h is the activation of a neuron, img the input of the network (an image), x and y describe the spatial position of the neuron, n specifies the layer and z is the channel index. For the mean activation of an entire channel z in layer n was maximized:

$$\text{img}^* = \arg\max_{\text{img}} \sum_{x,y} h_{n,x,y,z}(\text{img})$$

In this formula, all neurons in channel z are equally weighted. Alternatively, maximizing random directions involves multiplying neurons with different parameters, including negative ones, to study how they interact within the channel.

This optimization problem can be addressed in different ways. For example, instead of generating new images, was done searching through the training images and selecting those that maximize the activation (Nguyen et al., 2019). Using training data is a valid approach, but it has its drawbacks. The elements on the images can be correlated and cannot be seen as what the neural network is specifically looking for. For instance, if images that yield a high activation of a certain channel show a skin cancer mark and a hair outlier, was not known whether the neural network looks at the skin cancer mark, the hair outlier, or maybe both.

This optimization problem was introduced as Activation Maximization (AM) by Erhan et al. (2009). AM is a non-convex optimization problem for which one can attempt to find a local minimum via gradient-based (Szegedy et al., 2014) or non-gradient methods (Nguyen et al., 2016b). A simple approach is to perform gradient ascent (Erhan et al., 2009) with an update rule such as:

$$x_{t+1} = x_t + \epsilon_1 \frac{\partial a(\theta, x_t)}{\partial x_t}$$

Starting from random initialization x_0 (here, a random image), then iteratively taking steps in the input space following the gradient of $a(\theta, x)$ to find the input x that highly activates a given unit. ϵ_1 is the step size and is chosen empirically.

Using gradient ascent, network input is optimized instead of the network parameters ∂), which are frozen. The optimization was stopped when the neural activation has reached a desired threshold or a certain number of steps has passed.

4.4.3 Regularization Methods

Simply optimizing an image to trigger a neuron and visualize its features is not sufficient. Instead, the final result will be an image full of noise and nonsensical high-frequency patterns, but to which the network responds strongly, as introduced in Fig. 15, where the default number of optimization steps applied is 120.

After optimizing long enough, it was obvious that some of what the neuron detects well, but the image will be dominated by these high-frequency patterns, which are closely related to the phenomenon of adversarial examples (Szegedy et al., 2014).

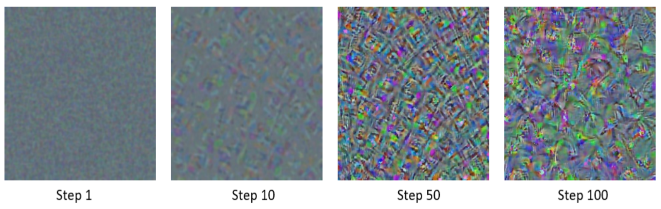


Figure 15: Different optimization steps during feature visualization

Feature visualization research is fundamentally concerned with mitigating the issue of high-frequency noise. To get useful visualizations, a more natural structure needs to be imposed by using some kind of prior, regularizer, or constraint. Therefore, the scope was limited to a specific distribution of images that are comprehensible, such as photo-realistic representations or images that look like those in the training set. This is achievable by the integration of natural image priors into the objective function function, as indicated by references (Mahendran and Vedaldi, 2016) and (Nguyen et al., 2016a). For example, an image prior may promote smoothness (Mahendran and Vedaldi, 2016), or penalize pixels of extreme intensity (Simonyan et al., 2014). These types of constraints are typically embedded in the AM methodology through a regularization term denoted as $R(x)$.

$$\text{img}^* = \underset{\text{img}}{\arg\max}\; h_{n,x,y,z} - R(x))$$

As explained in the main paper of Feature Visualizations by Olah et al. (2017), there are three categories of regularizations:

1. Frequency penalization
2. Transformation robustness
3. Learned priors

Methods such as *Frequency penalization* aim to reduce the undesirable noise characteristic of high-frequency variance in pixel values. Approaches to this issue may involve an explicit penalization mechanism, which reduces the variance among adjacent pixels through *Total variation* methods (Mahendran and Vedaldi, 2015), or they may adopt an implicit strategy by implementing a blurring process at each iterative step of optimization (Nguyen et al., 2015). However, *Frequency penalization* methods, do not differentiate between noise and high-frequency image details, such as edges, which can lead to a loss of important features. An improved approach involves the application of bilateral filters, which are designed to retain edge details while reducing noise (Tyka, 2016).

In the *Transformation robustness* method, the objective is to identify examples that maintain a strong activation of the optimization target despite undergoing minor transformations. To implement this, stochastic manipulations such as jittering, rotation, and scaling are applied to the images before the execution of the optimization algorithm.

The last regularization method *Learned prior*, produces the most realistic visualizations, but it may be unclear what came from the model being visualized and what came from the prior. A viable strategy involves the construction of a generative model such as a Generative Adversarial Network (GAN) or Variational Autoencoder (VAE), which transforms latent space vectors into data instances (Nguyen et al., 2016a).

During the application of feature visualization, there are applied different types of regularization techniques to fine-tune them and visualize a more realistic image, as shown in Table 3. The most used methods are from the two first categories such as *Frequency penalization* and *Transformation robustness*.

Table 3: Overview of different regularization methods used

Name	Description
Jitter	Randomly change the brightness of the image within a given range.
Rotation	Rotate the image by a random angle, 15 degrees.
Translation	Move the image horizontally or vertically, 0.1 value.
Resize	Resize the image to a given size factor of 1.2.
Bilateral Filter	Determined by the blur parameter and sigmaColor set to 75.
L1 Regularization	Regularization strength is controlled by a lambda value of 0.05.

The regularization term $R(x)$ in the formula above calculates the absolute value of the encoded output using L1-regularization. The optimization is initialized with random noise and different regularization techniques were experimented and parameters to obtain meaningful feature visualizations. Specifically, those types of spatial transformations were used: jitter, translation, random rotation, and resize. As well as the two most advanced types of frequency penalization L1-regularization and bilateral filters.

Using L1 regularization encourages sparsity in the model weights and in the generated visualizations, potentially leading to simpler and more interpretable patterns.

The bilateral filter is conditionally applied to the generated visualization to smooth out noise while maintaining sharp edges, often used to make the activation patterns clear.

4.4.4 Quantification Approach

Interpreting feature visualizations, especially in sensitive fields like medical imaging, presents a significant challenge due to the complexity and potential ambiguity of the visualized features. As previously stated in the main feature visualization paper by Olah et al. (2017) feature visualization alone cannot provide a complete understanding. This underscores the need for rigorous methodologies to evaluate and quantify the reliability and interpretability of these visualizations. Two main critical approaches might arise:

1. Quantification of feature visualization involves developing metrics or criteria that can objectively measure the informativeness, relevance, and accuracy of visualized features about the underlying data and the decisions made by the model.

2. Sanity-checking the reliability involves a set of methods or checks to ensure that the visualizations are meaningful representations of what the model has learned, rather than artifacts of the visualization process.

Even looking at hundreds or thousands of feature visualizations, cannot fully understand the neural network (Molnar, 2022). Network Dissection is an approach by Bau et al. (2017) which quantifies the interpretability of a unit of a convolutional neural network. The first difficult, but crucial step is data collection, because Network Dissection requires pixel-wise labeled images, and this was one of the main obstacles faced with the dataset.

Many researchers are convinced that feature visualizations are interpretable (Graetz, 2019) and that "features can be studied and understood" (Olah et al., 2017). One way to move forward is to measure the utility of feature visualizations in terms of their helpfulness for humans. In their studies, Zimmermann et al. (2021) and Borowski et al. (2021) have designed well-controlled psycho-physical experiments that aim to quantify the informativeness of the popular visualization method by Olah et al. (2017) using human judgment. This study is a psycho-physical quantification method that indicates that the widely used visualization by Olah et al. (2017) does not provide a causal understanding of CNN activations beyond simpler baselines.

There are increasing demands for generating interpretable feature visualizations because of their widespread usage. To address these shortcomings, a simple approach was created called Magnitude Constrained Optimization (MACO) (Fel et al., 2023). The main idea was to generate images by optimizing the phase spectrum while keeping the magnitude constant to ensure that generated visualizations lie in the space of natural images. This approach yields significantly better results - both qualitative and quantitative. The most important part is the quantitative method they have used. They introduce three different scores to compare the different feature visualization methods: the first one using AM by Olah et al. (2017), CBR (optimization in the pixel space) (Nguyen et al., 2015), and their method MACO. The metrics they used are:

1. Plausibility score: a feature visualization is considered plausible when it is similar to the distribution of images belonging to the class it presents. The plausibility is quantified through an OOD (Out-of-Distribution) metric which measures how far a feature visualization deviates from the corresponding ImageNet object category images on their representation in the network intermediate layers (Sun et al., 2022).

2. FID score quantifies the similarity between the distribution of the feature visualizations and that of natural images for the same object category. The FID score measures the distance between two distributions, while the plausibility score quantifies the distance from a sample to a distribution.

3. Transferability score measures how consistent the feature visualizations are with other pre-trained classifiers.

The researchers have computed all metrics using 500 feature visualizations and they observe that MACO produces better feature visualizations than those generated by Olah et al. (2017) and CBR (Nguyen et al., 2015). They also emphasize that the three proposed scores provide a more complete and accurate evaluation of the feature visualization methods. These metrics together form a coherent evaluation framework - requiring generated images to robustly maximize logit scores (transferability) while encouraging features that are both diverse concerning the distribution (FID), and representative of the internal paths that are typical for the model (plausibility).

Another approach for quantifying the feature visualization and checking their reliability is introduced by Geirhos et al. (2023). Their paper is named: "Don't trust your eyes: on the (un)reliability of feature visualizations." They provide some meaningful research questions such as:

1. Can feature visualizations be fooled? Checking the adversarial perspective by developing fooling circuits that trick feature visualizations into displaying arbitrary patterns or visualizations of unrelated units.

2. How can be used sanity-check feature visualizations? A simple sanity check demonstrates the use of feature visualizations processed along different paths compared to natural images.

Based on their conclusions, feature visualizations can be arbitrarily fooled or manipulated. Applying a sanity check, it is concluded that feature visualizations are processed very differently from natural images. The recommendation was: (a) to use the feature visualizations for exploratory, but not confirmatory use cases; (b) combining visualizations with additional methods including dataset samples.

5 Experimental Setup

This chapter focuses on the experimental setup starting with the hyperparameter tuning, and model evaluation, and the last experiment is the application of feature visualizations.

5.1 General Concepts

In classification, the model needs to look at features, e.g., the pixel values in an image, and then predict to which category (sometimes called a class) among some discrete set of options, an example belongs (Zhang et al., 2023). The simplest form of classification is when there are only two classes, a problem that is called binary classification. When more than two possible classes are present, the problem is called multiclass classification.

The choice of models such as Alexnet, VGG16, and ResNet50 emanates from their proven efficacy in image classification domains (Sharma et al., 2018). Within the initialization process, all three models undergo weight initialization, essential for instilling learnable parameters conducive to subsequent feature extraction and classification. Furthermore, the models are strategically transferred to the computational resource, a GPU or CPU, through the invocation of $model.to(device)$.

5.2 Hyperparameter Tuning

Upon the completion of a structured series of experiments aimed at optimizing the hyperparameters for the model, using only 10% available training data, was arrived at an optimal configuration that balances learning efficiency with predictive performance. These experiments ensure a systematic approach to hyperparameter tuning in the context of limited data availability. In Table. 4 are shown different hyperparameters selected as the best performing after hyperparameter tuning, for binary classification. For multiclass classification, the number of epochs is set up to 40. The final model configuration, determined to be the most effective across all tested dimensions, is as follows:

1. Batch Size “64” emerged as the optimal choice. Size “32” showed slightly better generalization but at the cost of increased computation time. Larger sizes (“128” and “256”) resulted in faster training, but exhibited signs of poor gradient estimation, as evidenced by inconsistent validation loss trends.

2. Number of Epochs: The model undergoes 25 complete iterations over the entire dataset for binary classification and 45 complete iterations over the entire dataset for multiclass classification, encapsulating the comprehensive scope of the training data. Training for more than 25 epochs in binary classification did not improve validation accuracy significantly, indicating that the model quickly learned patterns in the data.

Table 4: Hyperparameters across three binary classification model architectures

Model	Learning Rate	Batch size	Optimizer	Epochs
AlexNet	0.0001	32	Adam	15
VGG16	0.0001	64	Adam	25
ResNet50	0.0001	64	Adam	25

3. Learning Rate: The learning rate, denoted as 0.0001, dictates the magnitude of weight adjustments during each optimization step, influencing the convergence and stability of the training process. Higher rates (e.g., "0.1" and "0.01") led to volatile loss function behavior, indicating overshooting of the loss minimum. Lower rates (0.00001) significantly slowed down the convergence, requiring more epochs to achieve compatible accuracy.

4. Optimizer Selection: The Adam optimizer is enlisted for weight updates, employing adaptive moment estimation to enhance convergence in the presence of varying gradient magnitudes. While the stochastic gradient descent optimizer (SGD) demonstrated competitive performances, it required more fine-tuning of the learning rates and momentum parameters (Ruder, 2017).

5. Early Stopping Patience: The training regimen integrates an early stopping mechanism with the patience of 10 epochs, ensuring termination if a predefined number of epochs transpire without noticeable improvement in validation performance.

6. Number of Folds for Cross-Validation: The dataset undergoes partitioning into five folds, signifying the application of stratified k-fold cross-validation to ascertain robust model generalization.

5.3 Model Evaluation

5.3.1 Metrics for Model Evaluation

The logging mechanism encapsulated within the code facilitates a comprehensive understanding of the model's progression and performance. Metrics such as training loss, validation loss, balanced accuracy, ROC AUC, precision, recall, and F1 score are carefully documented. The visualization of confusion matrices and the persistently recorded learning rate contribute to a comprehensive analysis of model behavior. Furthermore, the architectural state of the model achieves the optimal validation performance is serialized, affording reproducibility and subsequent deployment in real-world scenarios.

Recall or Sensitivity quantifies the model's ability to correctly identify all relevant instances, calculated as the ratio of true positive predictions to the total actual positives. High recall indicates that the model is effective in detecting positive cases, but does not inform about the number of incorrect positive predictions made (false positives).

Recall, or Sensitivity, is defined as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision evaluates the accuracy of positive predictions from a classification model, calculated as a proportion of true positive predictions to the total predicted positives. High precision indicates that a model generates few false positives.

Precision is defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

F1 score is a statistical measure for assessing the accuracy of a binary classification system. It represents the harmonic mean of precision and recall, balancing them and providing a single metric to access the model's performance in scenarios with uneven class distribution.

The F1 score is defined as the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

A more detailed explanation of what TP, FP, FN, and TN are, can be found in Fig. 16.

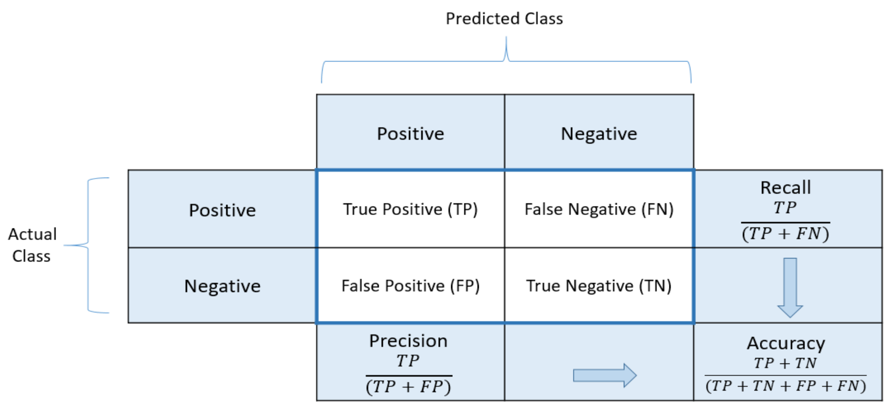


Figure 16: Building blocks of metrics

These metrics are the building blocks of many other metrics used such as accuracy, precision, and recall.

1. TP or True Positive is an instance for which both predicted and actual values are positive.
2. FP or False Positive is an instance for which the predicted value is positive, but the actual value is negative.
3. FN or False Negative is an instance for which the predicted value is negative, but the actual value is positive.
4. TN or True Negative is an instance for which both predicted and actual values are negative.

Balanced accuracy adjusts accuracy for imbalanced datasets because the dataset has a significant imbalance issue, offering a more insightful metric for evaluating classification models. It is computed as the average of the proportion of true positives correctly identified (recall) for each class, thereby accounting for the performance of both minority and majority classes.

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

Validation and Training loss are measures of how well the model's predictions match the actual data labels during and after training. Those metrics are fundamental for monitoring the learning process, identifying when a model is underfitting or overfitting, and tuning hyperparameters.

Finally, the **AUC score** stands for "Area under the ROC Curve" (Bradley, 1997). It measures the entire two-dimensional area underneath the entire ROC curve. The ROC curve stands for "Receiver Operating Characteristic Curve", which is a graph showing the performance of a classification model at all classification thresholds. This curve has two parameters: true positive rate (Recall) and false positive rate. An AUC score of 1.0 represents perfect classification, where the model correctly classifies all positive and negative instances, whereas an AUC score of 0.5 indicates no discriminative ability, akin to random guessing.

5.3.2 K-fold Cross Validation

K-fold cross-validation, as implemented in the Python library scikit-learn Pedregosa et al. (2011) is a statistical method for estimating the efficiency of machine learning models. It involves dividing the original dataset into a training set to train the model and a validation (or test) set for evaluating its performance. This process is repeated k times (folds), with each of the k subsamples used exactly once as the validation data. The outcomes from the k folds are then averaged or combined to produce a single estimate. The benefit of this method over a single split is that it mitigates the dependence of the model's performance on the specific manner in which the data is divided.

During the training process, the data is divided into k folds using "StratifiedFold" from "scikit-learn", ensuring each fold is a good representative of the whole as shown in Table 5. Stratification is crucial for maintaining the proportion of classes across each fold, especially important in multiclass classification tasks to prevent the model's performance bias toward dominant classes. For each fold, the model undergoes training with the training subset and is evaluated on the validation subset.

K-fold cross-validation is crucial for validating the stability and reliability of machine learning models.

Table 5: Resnet50 for binary classification: Training and validation across folds.

Iteration	Train Benign	Train Malign	Validation Benign	Validation Malign
1	26033	4085	6509	1021
2	26032	4086	6509	1021
3	26034	4084	6509	1022
4	26034	4084	6508	1021
5	26032	4086	6509	1021

5.4 Feature Visualizations

5.4.1 Qualitative Methods

Leveraging the PyTorch deep learning framework, as well as auxiliary libraries such as Torch Dream and Matplotlib, introduced in Section 4.1, the next steps endeavor to explain the activated feature maps of a specified layer within a fine-tuned neural network. This methodological pursuit aligns with the broader goal of comprehending the learned representations encoded by the network.

Finding the most activated filters for each layer. Before applying the activation maximization (AM), for each input image d and each convolutional layer m, the activations in the feature maps are collected by attaching a hook to that specific layer. The activation $a(m, i, d, j)$ represents the activation of the $j - th$ element in the feature map calculated from the image d and filter i in the convolutional layer m. For the next steps, once all images (from the validation dataset) of specific classes are processed in batches, the activations that are saved for each batch, are concatenated into a single numpy array for easier processing. This array has dimensions corresponding to (number of images x number of filters x spatial dimensions), where spatial dimensions are the width and height of the layer's output. Then, the maximum activation for each filter across spatial dimensions for each image of each class is found. A threshold is calculated based on the quantile of maximum activations per filter (using an activation threshold like 0.95, meaning the top 5% of activations). Filters that have a maximum above this threshold are considered significantly activated for that particular image in a specific class. To get the most activated filters across images of a specific class, the occurrences of each filter are processed and saved on a CSV file containing column information: class, filter index, and filter occurrence for each class. As a final step, the most activated filter indexes are used to apply feature visualization by AM to each specific model.

This method of finding the most activated filters is introduced later in the class-specific filter Section (referring to 5.5) to identify the class-specific filters mostly activated in each class. This approach effectively isolates the most relevant features (filters) that the network uses to recognize patterns specific to each class. By focusing only on filters that exceed a high activation threshold, is ensured that only the most activated and influential filters per class are considered for further analysis, which could involve more detailed visualization or deeper interpretative studies to understand how these filters contribute to the model's decision-making process.

Application of activation maximization technique. Ultimately, for VGG16 and Resnet50, where the application of feature visualization for each layer and channel is a critical step because of the large number of channels per layer, the goal was to find the top-N feature maps with the highest activation values for each layer. This process embodies a methodological approach to unveil and preserve the most activated feature maps within a designated layer of a pre-trained CNN when presented with a genuine image (Zeiler and Fergus, 2013).

On the other hand, AlexNet has fewer channels per layer compared to other models. In addition, feature visualization does not require the extra step of finding the top feature maps with the highest activation values for each layer. Instead, feature visualization is applied across all channels within the layers.

The next step is to apply the AM technique, as described in Section 4.4.2, with the help of the Torch Dream library listed in Section 4.1. Activation maximization is achieved through an iterative optimization process, where the goal is to find an image that results in the highest activation for a given channel and layer. This is accomplished by defining a loss function that quantifies the activation level of the target channel and then using gradient ascent to modify the image to maximize this loss. The process leverages the backpropagation mechanism, not for the training of the network, but for updating the image itself.

To visualize the activation of specific channels, a custom function is employed. This function computes the loss as the negative mean activation of the target channel, a strategy that encourages features maximally stimulating the neuron to emerge in the optimized image. By focusing on individual channels, the method reveals the diversity of features that the network has learned to recognize, from simple patterns in early layers to complex, abstract representations in deeper ones.

The optimization process is controlled by several hyperparameters and regularization techniques as explained in Section 4.4.3, to ensure the generation of meaningful visualizations. The default number of optimization steps was initially set to 120, but was changed to 200 during feature visualization using a custom function, depending on the model.

5.4.2 Quantitative Methods

The approach introduced in this master's thesis is based on an empirical sanity check for feature visualizations (Geirhos et al., 2023), as mentioned in Section 4.4.4. In the context of saliency methods, sanity checks have proven highly valuable for investigating method reliability (Adebayo et al., 2018). Neural networks use feature visualizations to explain natural input processing. These visualizations should follow similar paths as natural images. Images from the same class are expected to be processed along a similar path because they share certain features. The goal is to compare how similar natural images from a class with feature visualizations of the same class are processed. The comparison is based on different directions:

1. Comparing natural images and feature visualizations of the same class.
2. Comparing natural images and feature visualizations of the different layers.
3. Comparing feature visualizations of the different models.

The most important layers in each model for this type of analysis are the last convolutional layers. Unlike hidden layers, the features in the last convolutional layer have well-known selectivity based on ground truth data.

To measure the similarity between activations of inputs $x(i)$ and $x(j)$ in layer l of the network n, $\Gamma(\mathrm{nl}(x_i), \mathrm{nl}(x_j))$ was calculated, employing Spearman's rank order correlation. There are also alternatives like Cosine Similarity or Pearson correlation which also yield consistent results. Utilizing this metric allows us to assess whether images of a class have similar activations across the network. A high correlation suggests similar processing paths, while zero correlation indicates independent processing. The natural images used for the comparison are the validation indices predicted during each model's training and validation process.

The respective feature visualization of the specific class and layer is then compared with natural images of that class. From each layer, the top five feature visualizations with the highest activations are selected for comparison against natural images belonging to the same class.

5.5 Class-specific Filters

Before applying feature visualizations to specific layers of each model, the most activated filters are highlighted as the most responsive to features within a specific class. The methodology outlined in Section 5.4.1 is introduced in the initial paragraph to identify class-specific filters. This analysis utilizes images from the validation dataset, which were extracted during the training process from the first fold.

Datasets employed for class-specific filter experiments. Three experiments are conducted to identify the most activated filters per class in three different image sets, where only the validation indexes extracted during the training process from the first fold, are considered. The motivation behind using only validation images for analyzing filter activations was to uphold unbiased evaluation standards, and mitigate the risk of overfitting. This approach preserves the model memorizing specific features of the training set and instead focuses on its ability to generalize patterns to unseen data.

The initial dataset contains images from the validation dataset of ISIC 2020 used for binary classification, considering only the first fold indexes, categorized into five distinct classes through manual separation and identified using the z-score method, as later explained. These classes include different numbers of images tailored to specific classes, such as 70 images representing hairy artifacts, 40 images depicting stickers, 70 images featuring clinical markings, 50 images displaying ruler signs, and 60 images showcasing fluid marks, introduced in Section 3.2.

The second dataset consists of all validation images with and without outliers, where only 50% of the validation images for each class were selected, specifically 3254 benign images, and 510 malign images. The primary objective was to identify filters that activate patterns unique to each class of the binary classification task.

Additionally, a comparative approach was employed to determine the most activated filter for the third dataset, containing images of a multiclass task. In this instance, 100 images are randomly selected from the validation indexes for each class.

In the results chapter, Section 6.3 underscores the findings focusing on the most activated filter for particular classes of each model. This approach is particularly useful in contexts where understanding model interpretability and the reasoning behind its decisions is vital.

Application of z-score to identify outlier images. The first step is to identify all the images with outliers of the binary validation dataset and the detection is performed using the concept of Z-scores, which are statistical measures that describe a pixel's relative distance from the mean pixel value of the dataset in terms of standard deviations (Tschuchnig and Gadermayr, 2022). The mean and standard deviation of pixel values are calculated for all images in the dataset. The mean pixel value for each color channel (assuming RGB images, so three channels: Red, Green, and Blue) is determined by dividing the total sum of pixel values by the total number of pixels. Then the squared differences of each image are calculated by subtracting the mean pixel value from each pixel's value, squaring the result, and summing these squared differences. Finally, the standard deviation for each color channel is calculated by taking the square root of the average of the squared differences.

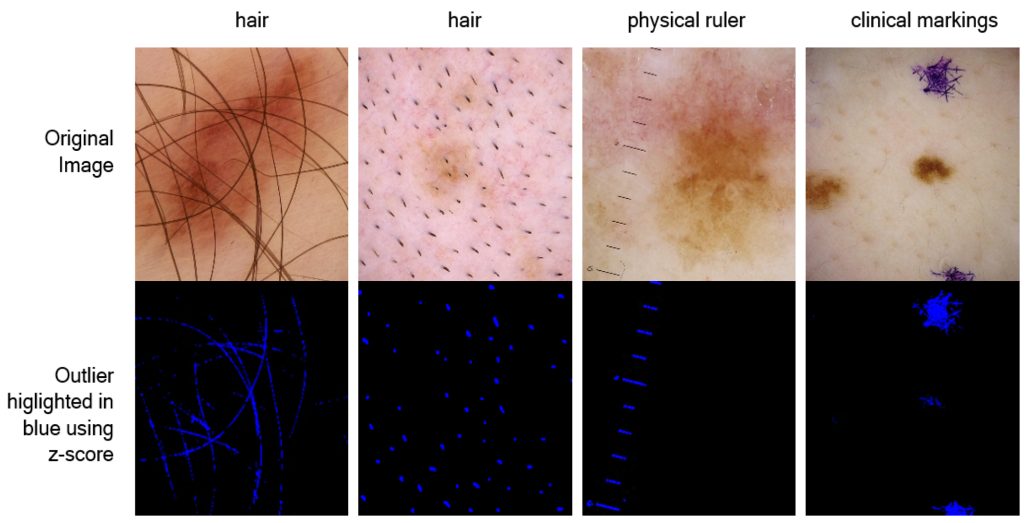


Figure 17: Application of z-score to identify outlier images.

Once the mean and standard deviation are known, the z-scores are calculated for each pixel in each image, by subtracting the mean value of each pixel's respective channel. Then the result is divided by the standard deviation of the respective channel. This gives the z-score for each pixel, which indicates how many standard deviations a pixel's value is from the mean.

The outliers are then detected based on a threshold (set to 2.0, which is typical for many statistical applications). Pixels with z-scores exceeding this threshold in absolute value are considered outliers. For visualization, a mask is created where these outlier pixels are marked in blue. This mask overlays the original image to visually highlight outliers, as shown in Fig. 17.

In the final steps, if any high z-score pixels are found, the original image along with the computed z-score image and the mask highlighting the outliers are saved. This way, the outlier images are saved and used for further processing steps in identifying class-specific filters.

6 Results

This chapter focuses on the presentation and evaluation of the research results. The following sections present the training and evaluation results of the models on both binary and multiclass datasets. The core of this chapter was to unveil the results of the experimental methods of feature visualizations in binary and multiclass classification tasks. Furthermore, qualitative and quantitative methods are introduced. The findings are presented using performance summary tables and graphical representations.

6.1 Binary Classification Task

The performance of each model in the binary classification task is very different because of their complexity. The performance of AlexNet shows progressive improvement over the epochs. Training and validation accuracy curves, as shown in Fig. 18, reveal that the model's accuracy increases steadily with more epochs, although validation accuracy shows a tendency to plateau, indicating that the model might have reached its learning capacity. This can happen due to the simplicity of AlexNet's model trained on a large number of samples. The confusion matrix indicates a higher number of false negatives, suggesting a potential bias toward predicting the negative class. Loss curves for AlexNet, as shown in Fig. 19, display a decreasing trend, but there is a slight uptick in validation loss towards the later epochs, which could imply the onset of overfitting.

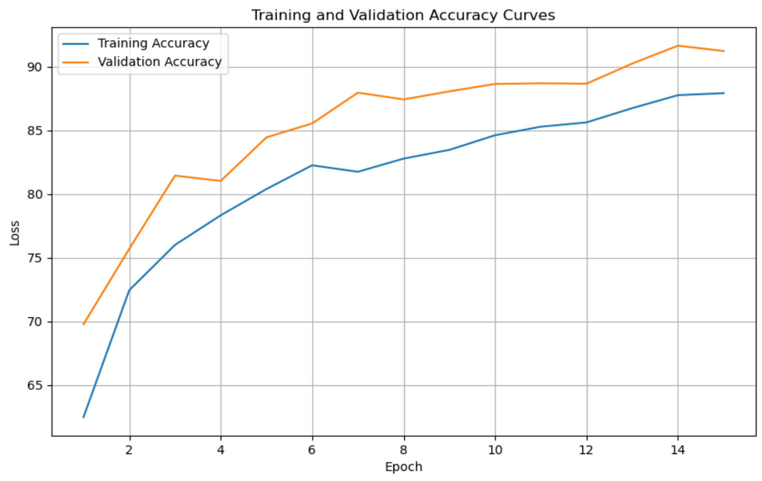


Figure 18: Accuracy curves for AlexNet

VGG16 demonstrates a quick escalation to high accuracy levels, with the training and validation accuracy closely aligned, which suggests good initial learning. However, there is a noticeable fluctuation in the validation accuracy as epochs increase, which may indicate some overfitting or model instability in later stages. The confusion matrix for VGG16 shows a relatively balanced classification with a reduced number of false negatives and positives compared to AlexNet. The loss curves decrease and stabilize early in the training process, with validation loss remaining slightly below the training loss, indicating a good fit.

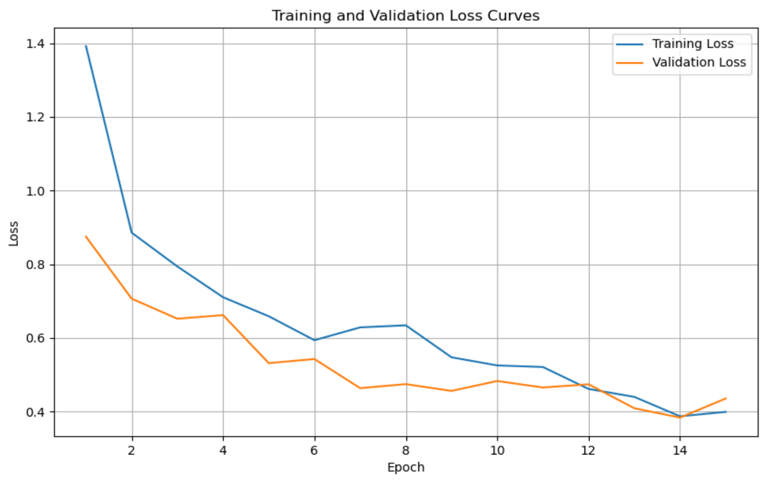


Figure 19: Validation and Training curves for AlexNet

On the other hand, ResNet50 showcases an impressive performance with rapid gains in accuracy in the initial epochs and some fluctuations in later epochs. Both training and validation accuracy curves begin to plateau, which indicates a well-trained model that generalizes well to new data. The confusion matrix for ResNet50 reveals fewer misclassifications both in terms of false positives and false negatives, which speaks to its high level of precision and recall. The loss curves for ResNet50 descend sharply and converge, which suggests a robust fit with neither significant overfitting nor underfitting.

The comprehensive results for both the VGG16 and ResNet50 models, including validation and training curves detailing accuracy and loss, are presented in Appendix A1 (see Appendix A.1).

When comparing the three architectures in Table 6, VGG16 stands out as the most effective model for binary classification in this scenario. It reveals some volatility that might be addressed with further tuning but exhibits remarkable training and validation accuracy. ResNet50 demonstrates superior generalization capabilities, as shown by its stable validation and test accuracy and the balanced confusion matrix. AlexNet, despite its acceptance performance, appears to be the least sophisticated of the three, with its learning curve showing signs of plateauing early and a higher tendency towards false negatives, which might require additional strategies to improve performance. The balanced accuracy and loss results for both the validation and training phases are presented as averages across all folds.

Results of Hyperparameter Tuning. Based on the Section 5.2, the accuracy throughout the hyperparameter optimization process was collected and the main goal was to find the best-performing hyperparameters for each model with the highest validation accuracy on validation data. The results can be distinguished better using the graph in Fig. 20, where each model has been optimized for three hyper-

Table 6: Training, validation, and test results of three model architectures.

	Training		Validation		Test
Model	Accuracy	Loss	Accuracy	Loss	Accuracy
AlexNet	81.1	0.638	85.3	0.534	81.3
VGG16	92.6	0.078	93.2	0.070	70.1
ResNet50	91.3	0.0126	91.1	0.0121	83.2

parameters, and their corresponding accuracies are plotted. Adjusting the learning rate, number of epochs and batch size changes the validation accuracy for each model. It was observed that training the models with a small value of the learning rate set to 0.0001 results in a high value of training and validation accuracy. Using a small number of epochs (only 15) and a batch size of 32 works well for AlexNet. After 15 epochs, the accuracy values plateau. Increasing the number of epochs to 25 and the batch size to 64 leads to higher accuracy for VGG16 and ResNet50. However, high values of batch size caused instability, especially with validation loss.

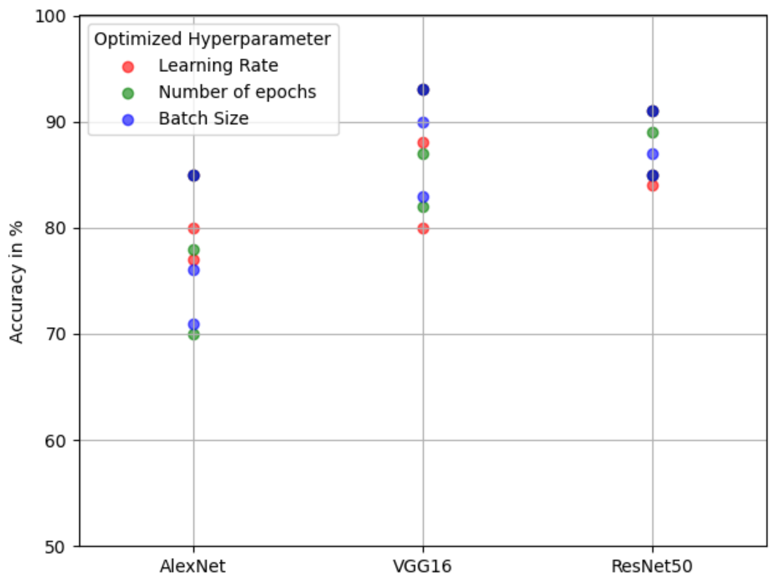


Figure 20: Validation accuracy of the models on validation data during hyperparameter optimization for binary classification. When trying to optimize a model's performance, adjusting hyperparameters like learning rate, number of epochs, and batch size can result in different accuracies on the validation set. Fine-tuning these parameters to find the highest accuracy was necessary.

Evaluation on test dataset Due to the lack of ground truth labels on the test set, the model was evaluated on a limited number of test data. With only 392 data samples, the statistical significance of the results is limited. However, the performance on a small, well-curated dataset can still provide valuable insights, especially in the field of medical image analysis where data can be scarce. Recently, new data were downloaded from the official website of the ISIC archive benchmark dataset (Codella et al., 2018b). Some of these data are dissimilar to the training and validation dataset. For evaluation of the test set, the test set of the 2016 ISIC dataset, is used and it has ground truth labels. The total number of samples in this test set is 392, where 28 are malign images and 364 are benign images. This dataset is also checked for any data leakage.

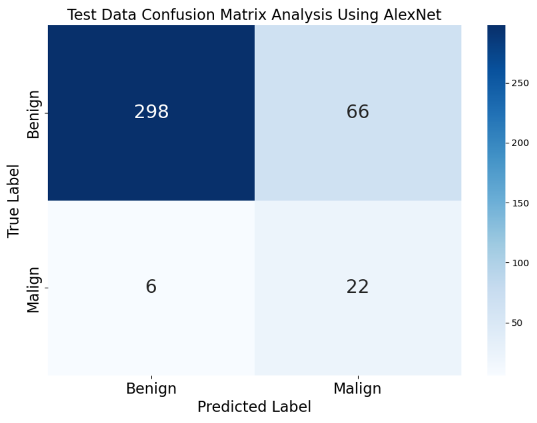
Table 7: Qualitative results on ISIC 2016 test set, binary classification.

Model	AUC	Accuracy %	Ensemble	Source
AlexNet	0.71	82	x	Fine-tuned
VGG16	0.70	69	x	Fine-tuned
ResNet50	0.72	**83.2**	x	Fine-tuned
Yu et al. (2017a)	*0.84*	-	x	Article
Codella et al. (2016)	0.80	-	✓	Published paper
Yu et al. (2017b)	**0.85**	-	✓	Conference Paper

The evaluation results of AlexNet and ResNet50 on the test data are shown in Fig. 21.

The first ranking in terms of AUC and accuracy is highlighted in bold, in Table 7, and the second ranking is indicated in italics. The fine-tuned models (AlexNet, VGG16, and ResNet50) archive better results without using an ensemble of models. Notations: AUC: the area under the ROC curve; Ensemble: ensemble method or not.

As depicted in Table 7, ResNet50 exhibits the most promising results among the fine-tuned models, with a test accuracy of 83.2% and the highest AUC score of 0.72. These metrics indicate its superior performance on this dataset. Specifically, when comparing the AUC score of the fine-tuned models within the same test dataset, the best performance is archived by Yu et al. (2017b) with an AUC score of 0.85. However, it is noteworthy that the AUC score of the fine-tuned models achieved in this master thesis appears comparatively lower when compared with those reported in other publications using the same test dataset.



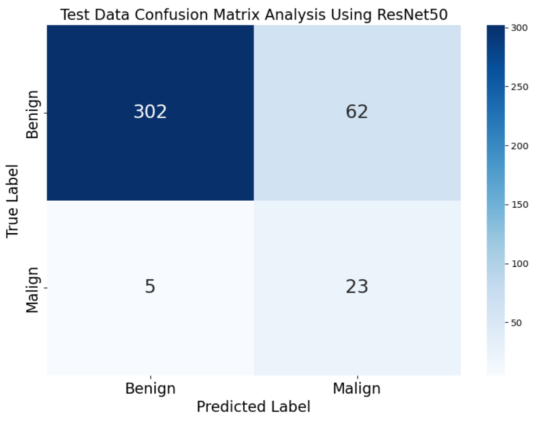


Figure 21: AlexNet vs. Resnet50: Confusion matrix on the test set for binary classification. Two confusion matrices are displayed in the image, one for each model, based on prediction results on the test set. The results show that both models can predict benign and malign samples with high accuracy, achieving a prediction rate of 82% for AlexNet and 83% for ResNet50.

VGG16 and AlexNet exhibit lower performance across both metrics, with VGG16 scoring the lowest. This suggests potential overfitting to the training data, especially if there was insufficient regularization or data augmentation to generalize to new data.

6.2 Multiclass Classification Task

For multiclass classification, only two models were considered, AlexNet and VGG16.

For the VGG16 model, both the training and validation accuracy curves demonstrate a steady increase in accuracy as the number of epochs increases, indicating that the suggesting effective learning. The training accuracy slightly exceeds the validation accuracy, which may suggest that the model fits well with the training data. Similarly, the training and validation loss curves for the VGG16 model show unexpected behavior, where the training loss decreases over time, and the validation loss follows the training loss closely but starts to slightly fluctuate after around 20 epochs. This can be a sign of potential overfitting issues.

The confusion matrix for VGG16 shows that certain classes, like MEL(melanoma) and NV(nevus), have high true positive rates because their number has a significant impact on the dataset. However, there is some confusion between classes such as BKL and NV, as well as between MEL and NV. The color scale indicates the matrix is normalized across all folds, showing consistent performance across different subsets of the data.

For AlexNet, was observed a similar trend in the accuracy curves as with VGG16, with a steady increase over epochs. However, the validation accuracy plateaus earlier and stays more constant compared to VGG16, suggesting that AlexNet may not be

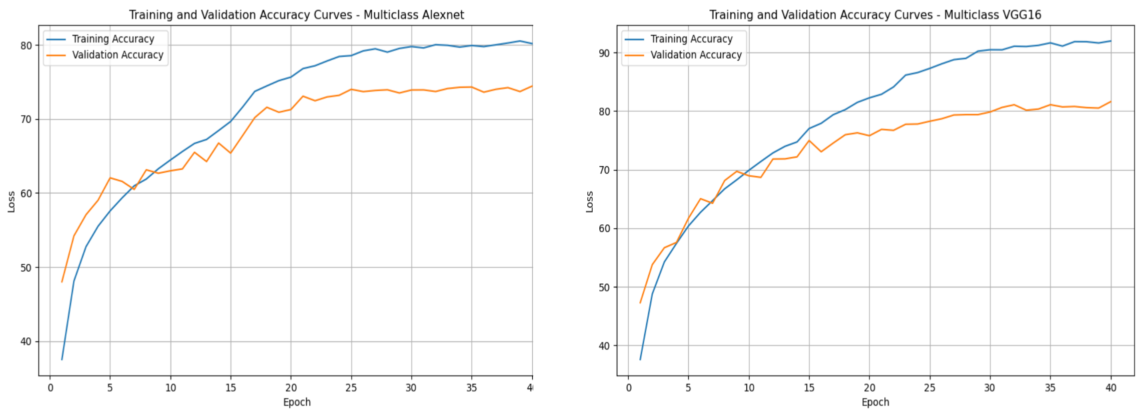


Figure 22: Accuracy curves for AlexNet (left) and VGG16 (right), multiclass classification.

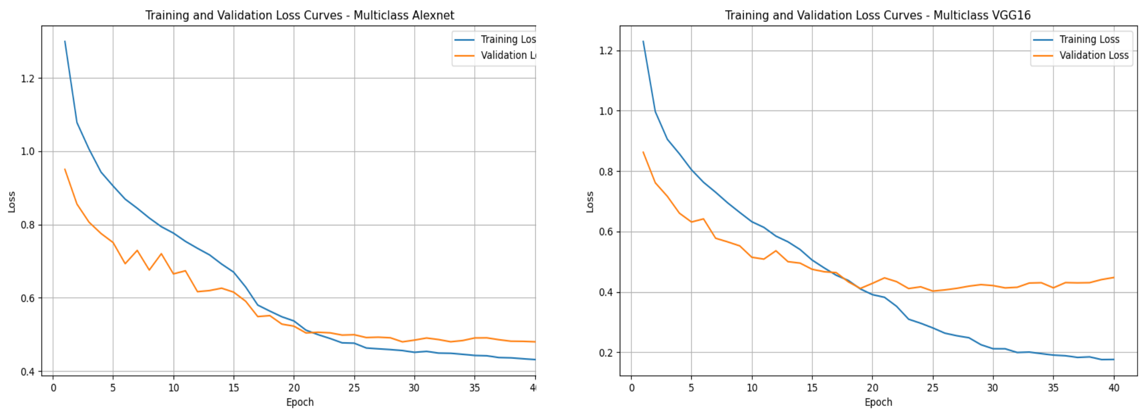
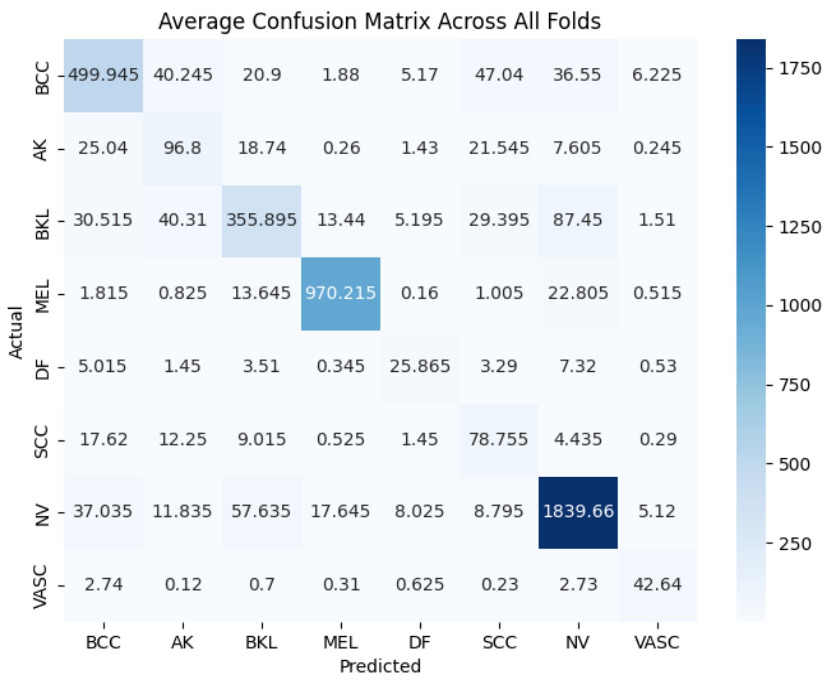


Figure 23: Training and validation loss curves for AlexNet (left) and VGG16 (right), for the multiclass classification task.

learning as effectively toward the later epochs. The loss curves for AlexNet also show a decrease over time, but there is a more noticeable gap between the training and validation loss, which could be indicative of slight overfitting or less effective generalization, shown in Fig. 23. The confusion matrix for AlexNet demonstrates similar patterns to VGG16, with high true positive rates for certain classes, but some confusion between classes such as BKL and NV, MEL and NV. The diagonal values in the confusion matrix are generally high, suggesting good classification performance, but some off-diagonal values indicate misclassifications.

In conclusion, both VGG16 and AlexNet are effective models for classifying skin cancer types on the ISIC dataset, with AlexNet showing a slightly better generalization in the loss curves and VGG16 showing a more pronounced early plateau in validation accuracy, shown in Fig. 22. The confusion matrices for both models indicate that while they perform well in certain classes, there is still room for improvement in distinguishing between similar skin lesion types, shown in Fig. 24 and



Actual \ Predicted	BCC	AK	BKL	MEL	DF	SCC	NV	VASC
BCC	499.945	40.245	20.9	1.88	5.17	47.04	36.55	6.225
AK	25.04	96.8	18.74	0.26	1.43	21.545	7.605	0.245
BKL	30.515	40.31	355.895	13.44	5.195	29.395	87.45	1.51
MEL	1.815	0.825	13.645	970.215	0.16	1.005	22.805	0.515
DF	5.015	1.45	3.51	0.345	25.865	3.29	7.32	0.53
SCC	17.62	12.25	9.015	0.525	1.45	78.755	4.435	0.29
NV	37.035	11.835	57.635	17.645	8.025	8.795	1839.66	5.12
VASC	2.74	0.12	0.7	0.31	0.625	0.23	2.73	42.64

Figure 24: Validation confusion matrix for AlexNet. It is observed that melanoma, nevus, BKL, BCC, and VASC are the best well-predicted classes during multiclass classification. The reason behind this can be the large number of samples that each one of those classes contains, compared with other minority classes such as AK, DF, and SCC.

Fig. 25. The high true positive rates for melanoma are particularly promising, given the importance of accurate detection for this dangerous skin cancer.

Table 8: Results of the training, and validation accuracy for the multiclass task.

Model	Training %	Validation %
AlexNet	71.4	68.6
VGG16	78.5	73.4

As introduced in Table 8, VGG16 has shown better results than AlexNet in both the training and validation processes. Although VGG16 has a higher accuracy value, accuracy alone cannot be used as a sole indicator to determine the best-performing model. The confusion matrices in Fig. 25 and Fig 24 show that AlexNet predicts the values of the majority class more accurately than VGG16.

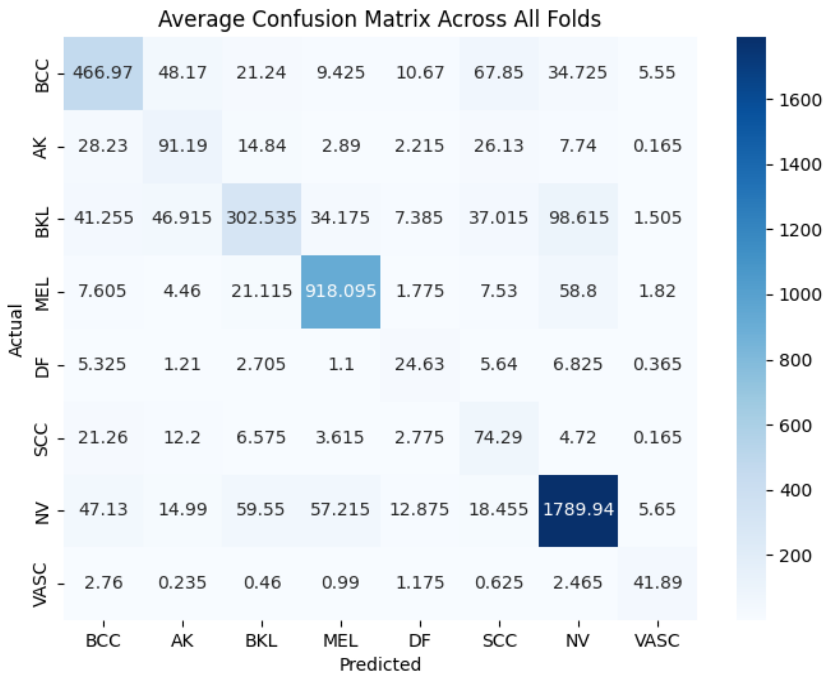


Figure 25: Validation confusion matrix for VGG16. The NV class has the highest true positive value, followed by other majority classes such as melanoma (MEL), BCC, and BKL.

Evaluation of the models in the test set. During the evaluation of the test set, the same procedure is followed as binary classification explained in Section 6.1, due to the lack of ground truth labels on the test set, the model is evaluated on a limited number of test data, with only 1,910 data samples in total, introduced in Table 2.

Table 9: Qualitative results of the test and validation set.

Model	Test AUC	Val. Accuracy	Source
AlexNet	0.56	*68.6*	Fine-tuned
VGG16	0.58	73.4	Fine-tuned
Dohan (2023) ResNet9	-	75.90	GitHub page
Dohan (2023) ResNet34	-	**82.20**	GitHub page

For the assessment of the generalization capabilities of CNNs, the validation dataset results are crucial. These results presented in Table 9, illustrate the validation accuracy, serving as a fundamental metric of each model's ability to correctly classify unseen data. The comparison encompasses both fine-tuned versions of established models and two other model accuracies that were gauged using the identical evaluation set. In the comparison, the results from two variations of the ResNet architecture, as proposed by Prasanna (2020).

The Table 9 provides a clear and comparative snapshot of the performance of each model, enabling us to draw informed conclusions about their relative effectiveness. The results indicate that the ResNet31 model outshines the rest with an accuracy of 82.20%, suggesting that its deeper architecture captures features more effectively. The fine-tuned models have achieved a lower accuracy than other versions, indicating that those models may not be as effective at correctly identifying or classifying patterns in the validation dataset as those with higher accuracies.

6.3 Class-specific Filters.

The use of class-specific filters across various conditions such as: with outliers, and across different classes, is driven by the need to understand and improve the internal representations that a neural network model has learned, introduced in Section 5.5. The outlier images, as detailed in Section 3.2, are identified as those containing artifacts like hair, rulers, fluids, clinical markings, stickers, and other elements.

The AlexNet was employed to compute class-specific filters, for binary and multiclass classification. The motivation behind including outliers in the feature activation can provide insights into the extreme cases where the model might be overfitting to particular examples or capturing noise. Visualizing the features that activate certain filters can give direct insights into what the model "sees" or considers important when classifying data into different classes. If certain filters are found to be frequently activated by noise or irrelevant features, they can be a focus for model improvement. By understanding which features are activated for each class, the training process can be tailored to emphasize or de-emphasize certain filters, leading to better learning and generalization. The most activated filters for a certain class are used to apply feature visualization to show the features learned by that specific filter for that class. The method used is explained in detail in Section 5.4.1.

In Fig. 26, class-specific filters for binary and multiclass classification are presented, showcasing results from a fine-tuned AlexNet model. Only images from the validation dataset are considered. Outlier images are excluded to identify filters activated by specific classes of images with outliers (first row), for binary tasks (second row), and multiclass tasks (third row). During the experiment, only the last convolutional layer of the AlexNet model is considered which can capture the most important features.

The normalization of filter occurrence values into percentages is done by dividing the number of occurrences in each cell, by the total number of images used for

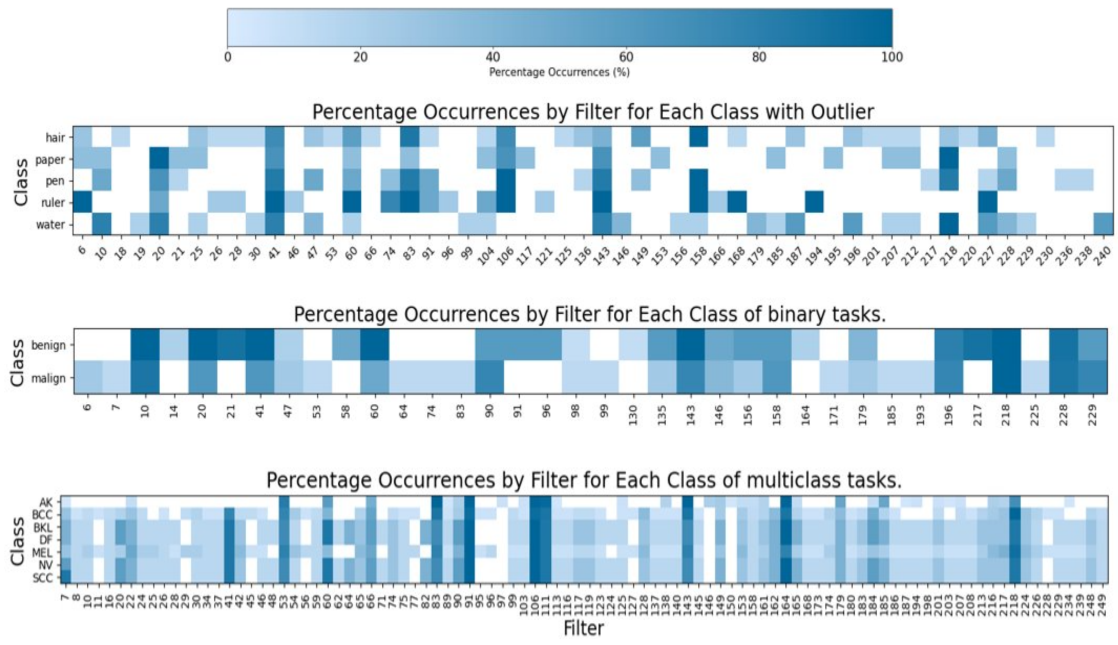


Figure 26: Class-specific filters for binary and multiclass classification, layer 10, last convolutional layer of AlexNet model. Images with and without outliers are taken into consideration, which are part of validation indices.

that particular class and then multiplying by 100. Raw counts are converted into percentages, to compare filters within each class, especially when the total counts for classes differ significantly.

Based on the observations, the filter numbers: 10, 20, 41, 60, 143, 218, and 228 are the most activated filters for the outlier class. It is observed that certain filters activated by the benign and malign classes in binary classification show similarities, yet there are also some differences.

Investigating other heatmaps in Fig. 26 is concluded that some filters are highly activated in all classes, mostly shown in the binary tasks. These filters, which are the same most activated for both outlier images and binary dataset of the AlexNet model, include those with the following indexes: 10, 20, 41, 60, 143, 218, and 228.

In the multiclass classification task using the AlexNet model, the filters that exhibit the highest activation levels are 41, 53, 60, 83, 91, 106, 143, 164, and 218.

The main reason why those certain filters are consistently highly activated across all classes and tasks indicates that these filters are detecting features that are universally present or particularly salient in the input data. Here are some reasons why this might occur based on subjective observations:

1. General features: These filters might capture very general features that are common in all classes, such as edges, colors, textures, or gradients.

2. Considering that the layer that is used is the last layer of the AlexNet model, those filters might have captured high-level abstractions that could be common across classes.
3. Dominant features: Some features might be dominant in the input data and therefore result in a higher activation in the convolutional filters.
4. Transfer learning artifacts: The model is pre-trained on a different dataset and then fine-tuned for a specific task in the ISIC dataset. These filters might have been important for feature detection in the pre-training dataset and retained their significance due to the transfer learning process.

In the context of the CNNs, the number of filters in the last convolutional layer varies across architectures. Specifically, while AlexNet's final layer has 256 filters, VGG16 counts 512, and ResNet50 escalates it to 2048. As introduced in Table 10, the most activated filters exhibit variability across different models; however, these filters remain consistent between the outlier and binary task.

Table 10: Top five most activated filters in the last convolutional layer of each model.

Model	Binary/ Outlier	Multiclass
AlexNet	41, 60, 143, 218, 228	41, 60, 143, 164, 218
VGG16	46, 121, 270, 394, 420	81, 119, 206, 450, 484
ResNet50	152, 388, 1064, 1165, 1836	no data

6.4 Feature Visualization

6.4.1 Qualitative Methods

To begin with, a noisy image is needed and a pre-trained neural network like AlexNet or VGG16 is trained on the ImageNet dataset. Then, feature visualization will be applied to it. The same procedure will be repeated on the fine-tuned models with the skin cancer dataset, both for binary and multiclass classification.

Feature visualization applied to pre-trained VGG16 model, for the binary classification task. The Fig. 27 displays the feature visualization outcomes for the top five feature maps of the selected layer within the pre-trained VGG16 model. The input is a "noisy" image, which refers to an initial image composed of random pixel values. Each feature visualization labeled with the corresponding filter represents the patterns that most strongly activate that particular filter within the neural network. These visualizations can look abstract because the features that neural networks use to identify objects are often not intuitive to human observers.

The feature visualization process applied to the first layer of pre-trained VGG16, as shown in the first row of Fig. 27, highlights the types of features that activate certain filters in this layer. "Filter 0" and "Filter 10" appear sensitive to specific edge orientations, such as vertical or near-vertical patterns. Meanwhile, other filters detect contrasting edges, different textures or noise patterns, and color gradients.

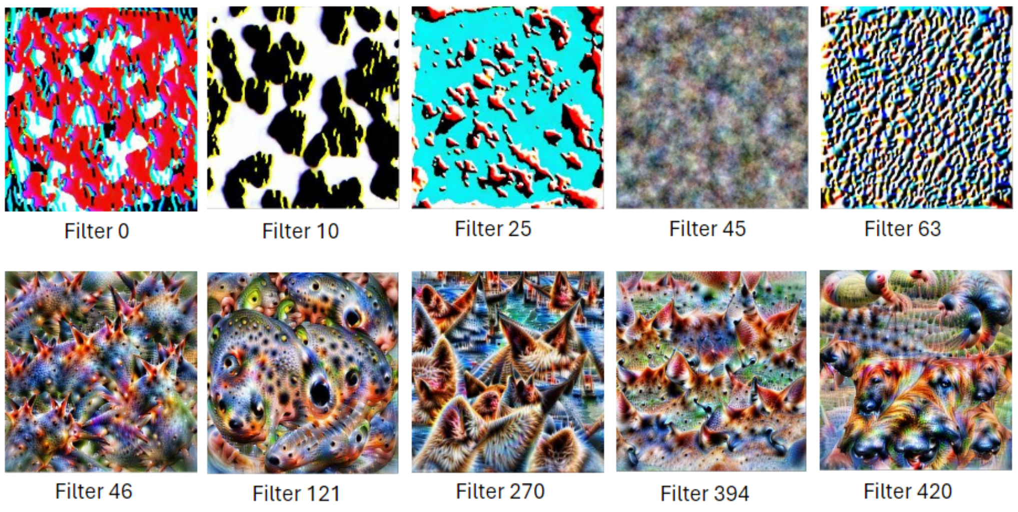


Figure 27: Feature Visualization of the first and last convolutional layer of **pre-trained VGG16**, for the binary classification task. The first layer contains 63 filters, and the last layer (index 28) has 512 filters. The top-5 filters with the highest activation values have been chosen for presentation, calculated as explained in Section 5.4.1. The first row corresponds to the first layer, and the second corresponds to the last layer.

The feature visualization process applied to the last layer of pre-trained VGG16, as shown in the second row of Fig. 27, highlights the types of features that activate certain filters in this layer. Based on the observations, "Filter 46" appears to capture textural details, possibly from natural textures. Other filters detect circular patterns, complex multi-colored patterns, and a mix of features, suggesting sensitivity to diverse visual characteristics in images.

In the next step, a noisy image as input is needed and a fine-tuned neural network like AlexNet or VGG16 is trained on the skin cancer dataset. Then feature visualization will be applied. The feature visualizations below are applied to VGG16 fine-tuned in the skin cancer ISIC benchmark dataset for binary classification tasks.

Feature visualization applied to fine-tuned VGG16 model, for the binary classification task. The feature visualization process applied to the first layer of fine-tuned VGG16, as shown in Fig. 28, highlights the types of features that activate

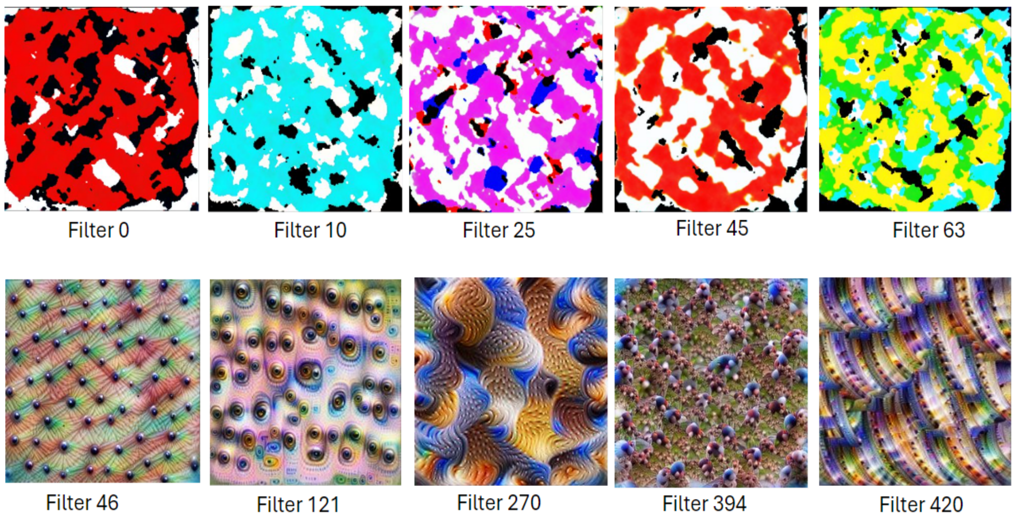


Figure 28: Feature Visualization of the first and last convolutional layer of **fine-tuned VGG16**, for the binary classification task. As with the pre-trained VGG16, the fine-tuned model uses the same filters for each layer. The first row corresponds to the first layer, and the second corresponds to the last layer.

certain filters in this layer. All filters appear to highlight edges and color distribution characteristics of certain skin lesions, although the colors appear excessively exaggerated. There are a few reasons why feature visualization might show colors like red or blue that are not directly present in the dataset:

1. Contrast maximization shows an amplified version to highlight the filter's response.
2. Image preprocessing and regularization techniques can introduce colors to distinguish between features that the filter is sensitive to.

The feature visualization process applied to the last layer of fine-tuned VGG16, as shown in Fig. 28, highlights the types of features that activate certain filters in this layer. Based on the observations, the first most activated filter seems to respond to irregular patterns in lesion growth, indicating its sensitivity to shapes and structures. Notably, the last filter might focus on distinct skin textures relevant to cancer diagnosis, including irregular pigmentation and mole shapes.

Feature visualizations without regularization techniques applied to fine-tuned VGG16 model, for the binary classification task. Applying regularization techniques, explained in the Section 4.4.3 can result in the filters activating on more meaningful patterns in the image rather than fitting to noise and other irrelevant variations, such as ones shown in Fig. 29.

The filters require regularization to better respond to nuanced features associated with particular classes or layers.

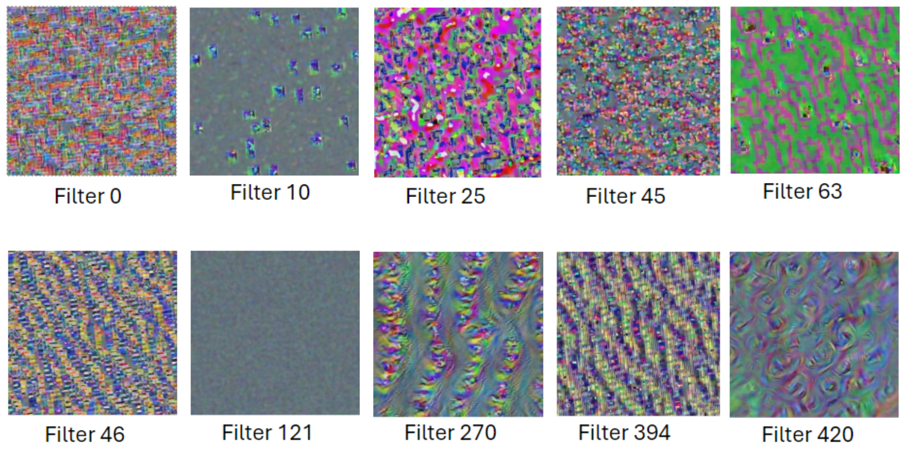


Figure 29: Feature visualizations without regularization techniques applied to the **fine-tuned VGG16 model**, represents the role of using regularization techniques explained in Section 4.4.3. Those visualizations likely represent the raw activations of filters, which can be noisy, overly complex, or too sensitive to the specific details of the dataset. The first row corresponds to the first layer, and the second corresponds to the last layer.

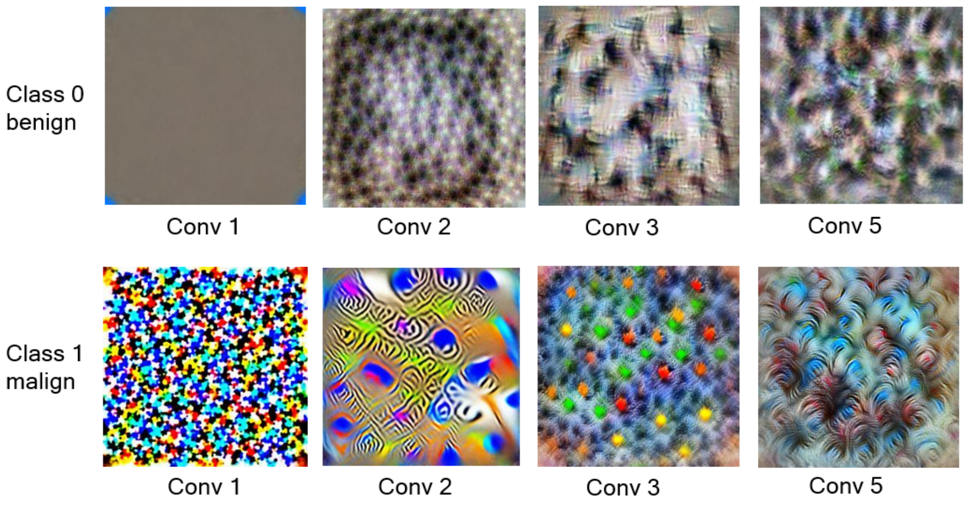


Figure 30: Class-specific feature visualizations of the **fine-tuned AlexNet model** for binary classification. The top row, labeled "Class 0: Benign", showcases the feature visualizations extracted from convolutional layers during the processing of benign images by the model. The bottom row, labeled "Class 1: Malign", illustrates the feature visualizations from the same layers when the model is presented with a malignant sample.

Feature visualizations applied to fine-tuned AlexNet model for each class in binary classification. The Fig. 30 showcases two rows of visual representations of feature visualization outputs from the AlexNet model that has been fine-tuned for the binary classification tasks, to differentiate between benign and malignant classes.

The feature visualizations appear relatively uniform for the benign class, indicating that the features learned for the benign class may not be as complex. Conversely, the feature visualizations of malignant class exhibit greater complexity, with vivid colors and patterns. The contrast between the two classes highlights how the network has learned to activate differently for benign and malign inputs. Earlier layers show more basic feature detectors, while deeper layers exhibit more complex interactions, potentially capturing the higher-order structures that distinguish between the two classes.

The feature visualizations are applied across the entire convolutional layer rather than targeting specific filters, thereby presenting a comprehensive visualization of all features within that layer.

An overview of feature visualizations applied for binary classification. The feature visualizations generated by the ImageNet pre-trained model exhibit a broader array of patterns and colors, which aligns with the diverse range of images present within the ImageNet dataset. The feature visualizations produced by the skin cancer fine-tuned model exhibit a heightened focus on distinct types of features. These features correspond to textures, edges, and colors relevant to identifying different types of skin cancer. The visualizations of the skin cancer dataset may exhibit patterns that are either repetitive or structured, reflecting the regularity or irregularity of skin textures and colors. These patterns are crucial for the model to distinguish between benign and malignant lesions.

In Appendix Section A.2, are presented additional results from the analysis of both pre-trained and fine-tuned AlexNet and ResNet50 models. These results encompass detailed feature visualization results for specific filters found within the first and last convolutional layers of each model. Observations and interpretations of these visualizations provide insights into the changes in feature extraction behavior resulting from the fine-tuning process. Please refer to Section A.2 for a comprehensive view of this analysis.

Feature visualizations applied to the multiclass classification task. In a comparative analysis of feature visualizations from the last convolutional layer of a fine-tuned VGG16 model, as shown in Fig. 31, distinct patterns emerge when the model is adjusted for binary versus multiclass classification. The fine-tuned VGG16 for binary classification demonstrates consistent visual patterns across filters, reflecting the network's focus on distinguishing between cancerous and non-cancerous lesions.

Conversely, the multiclass displays a more diverse array of patterns, reflecting the model's necessity to discriminate between multiple classes of skin conditions. Each filter appears to capture a broader spectrum of features, reflecting the complex nature of identifying various types of skin lesions.

It is crucial to acknowledge that these visualizations do not include those of a pre-trained VGG16 model, as they are generally uniform for binary and multiclass tasks.

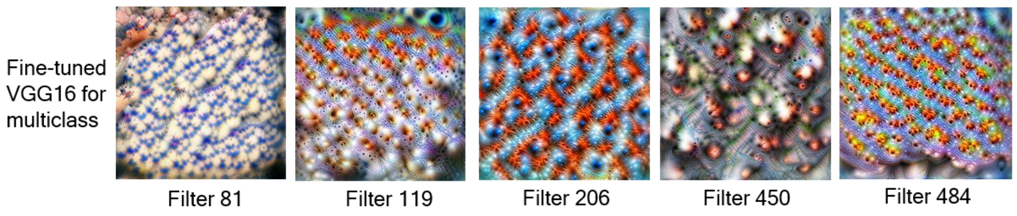


Figure 31: Feature Visualization of the last convolutional layer of **fine-tuned VGG16**, for the multiclass classification task. The top-5 filters with the highest activation values have been chosen for presentation, calculated as explained in Section 5.4.1.

Visualizing the most activated class-specific filters in the AlexNet model. Section 6.3, introduces a method for identifying the most activated filters of each class, using fine-tuned AlexNet model for binary and multiclass classification.

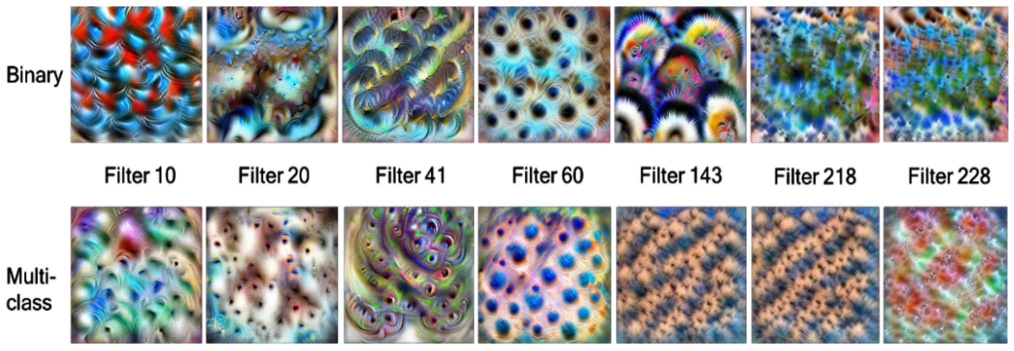


Figure 32: Visualizing the most activated class-specific filters in the AlexNet model.

To better understand the most activated filters across tasks and classes, feature visualizations are applied to each, presented in Fig. 32. These visualizations show that the binary class images reveal symmetrical patterns with vibrant color contrast, signifying the filter's tendencies to respond to specific, high-contrast features within the input data. Similarly, the multiclass filters exhibit greater complexity, indicating a higher level of abstraction in their responsiveness to filters. Additionally, it was challenging to determine whether the feature visualizations included outlier features or not. Identifying outlier patterns in feature visualizations solely through visual inspection can pose a challenge, as some patterns may resemble outliers but require thorough analysis for conclusive determination.

6.4.2 Quantitative Methods

Section 6.4.1 explained the qualitative part of feature visualization in detail. However, these visualizations are hard to interpret, particularly for medical images. As introduced in Section 5.4.2, some metrics can be used to check the reliability of feature visualizations and quantify their correlation between natural samples of the same class or feature visualizations of different layers. The core idea is simple: Feature visualizations are designed to explain how neural networks process natural input (Geirhos et al., 2023). This implies that effective visualizations should undergo processing similar to natural images once generated. This underscores the importance of employing a quantification method as an empirical sanity check for feature visualizations, introduced in Section 5.4.2 of this master's thesis.

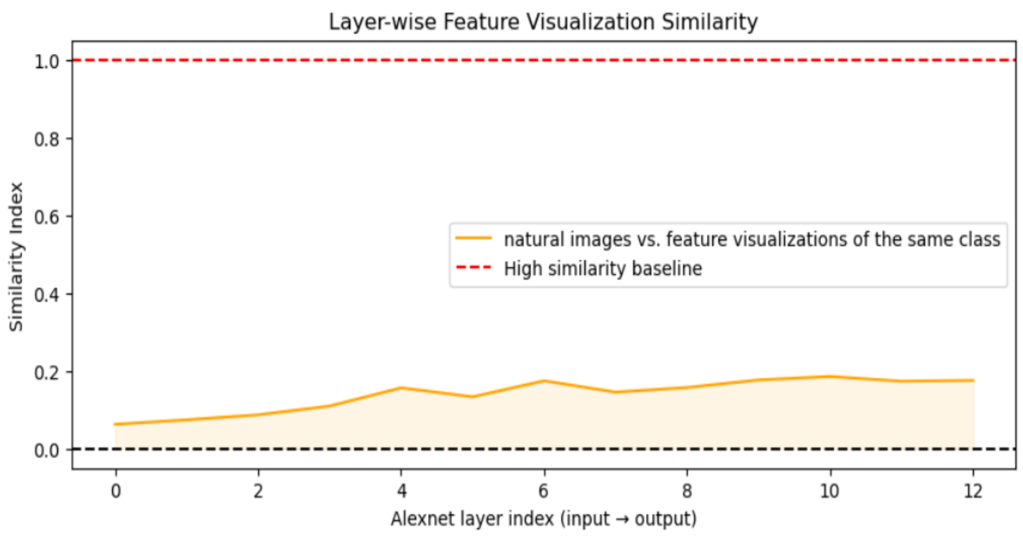


Figure 33: Empirical sanity check for **AlexNet** applied for the **binary classification** tasks. The similarity of a layer's activations caused by natural images and feature visualizations across layers (only the features block) is measured. Similarity increases in the last layers of AlexNet, because of roughly similar patterns with natural images.

As can be seen in Fig. 33, 34, and 35, first-layer feature visualizations are processed differently from natural images throughout most of the network. Although later layers exhibit a stronger correlation, it does not necessarily imply that activations stem from the same path. The low values of the similarity index, and the investigated feature visualization for all three models, do not pass completely the sanity check. The class used to calculate the Spearman correlation for all three models is malignant, due to complex features observed during class-specific feature visualizations.

The $x - axis$ denotes the index of layers, progressing from the input to the output layer, while the $y - axis$ represents the similarity index. The similarity index indicates how similar the feature visualizations are to natural images across the layers of the specific model. In all three models, the similarity index calculated using Spearman's rank correlation, tends to increase as the layers progress deeper. This trend suggests that higher layers capture more complex and abstract features, resulting in visualizations that correlate more closely with those layers.

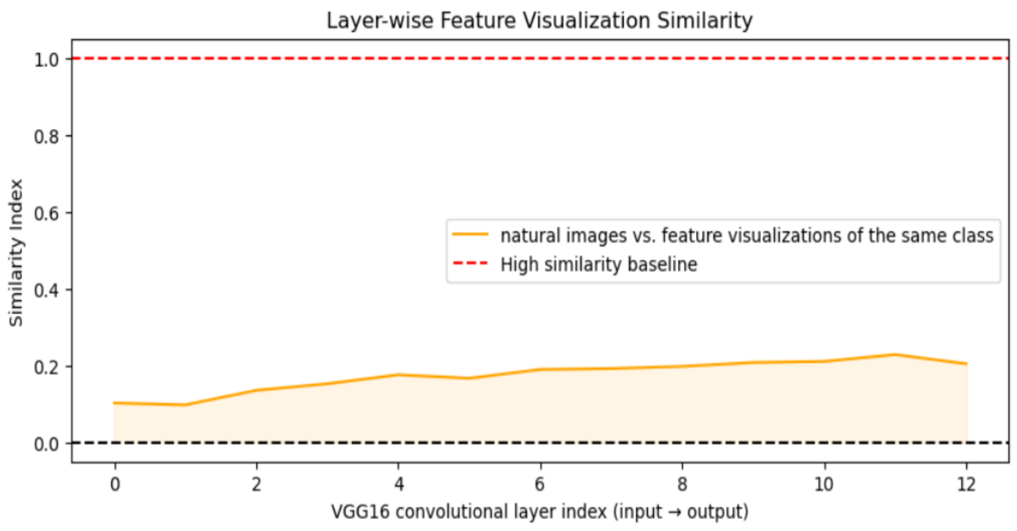


Figure 34: Empirical sanity check for **VGG16**, applied for the **binary classification tasks**. The feature visualizations have a low similarity index with natural images, as the similarity index remains below the middle value.

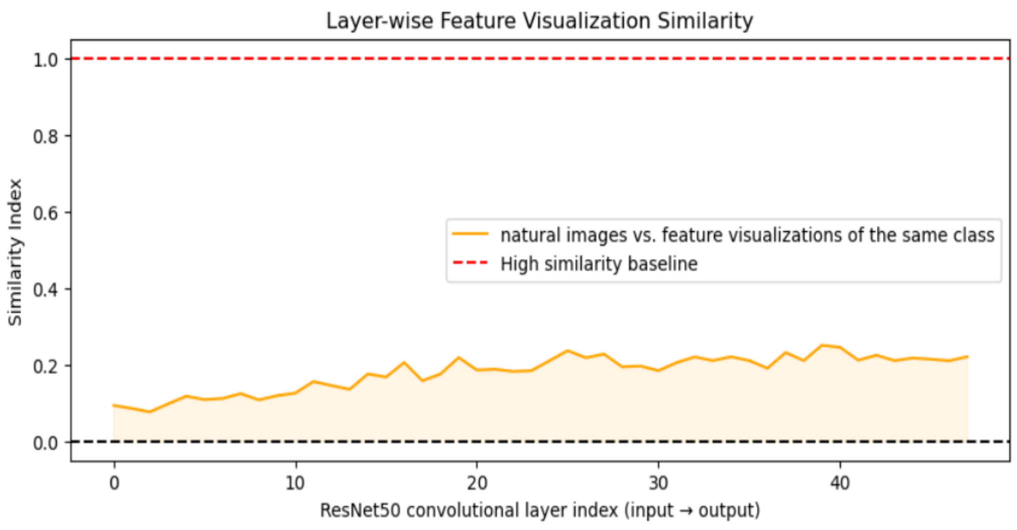


Figure 35: Empirical sanity check for **ResNet50**, applied for **binary classification tasks**.

The core idea was to apply Spearman correlation only to convolutional layers for VGG16 and ResNet50, because of their ability to capture hierarchical features in the data.

Heatmaps displaying correlation of feature visualizations across different layers, for binary classification. The Fig. 36 and Fig. 37 depict the correlation heatmaps of feature visualizations from two different CNNs, especially AlexNet and VGG16.

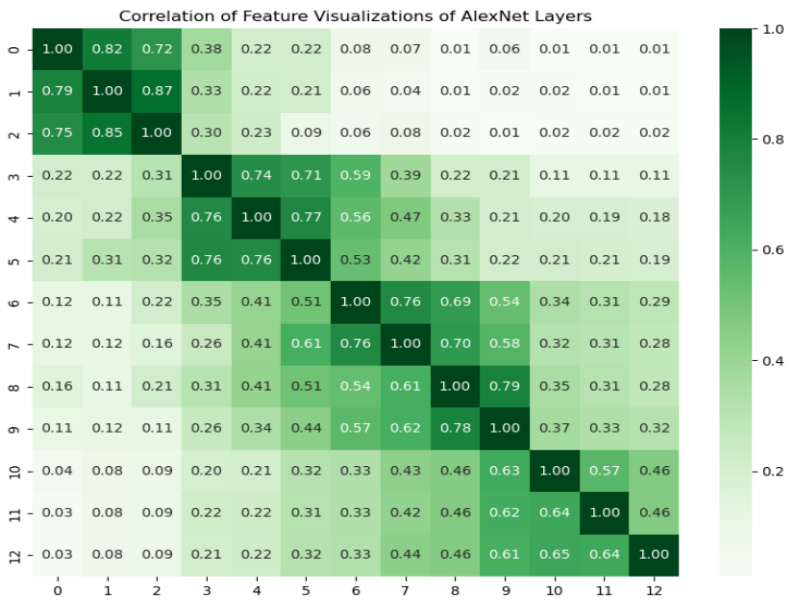


Figure 36: Empirical sanity check for AlexNet: analyzing the correlation across different layers.

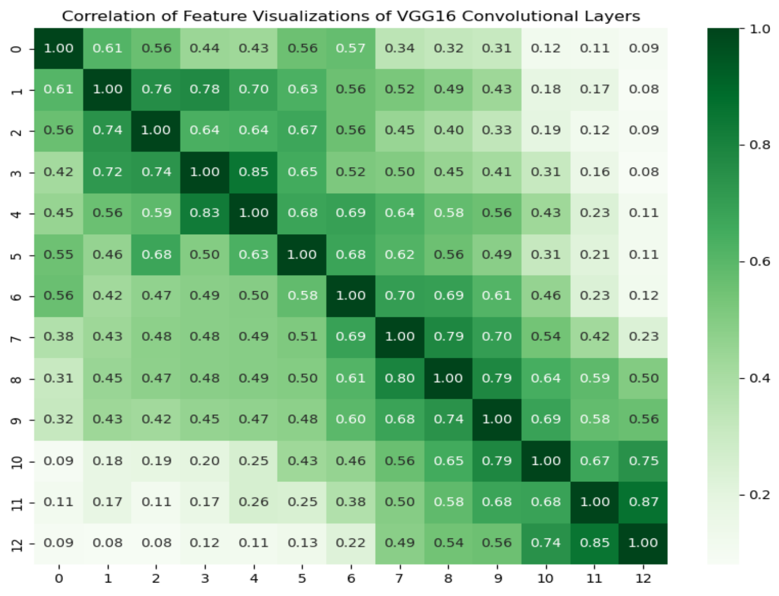


Figure 37: Empirical sanity check for VGG16: analyzing the correlation across different layers.

The only difference in creating the heatmaps is that the diagonal is the correlation of each layer with itself, which is the reason for the highest correlation in the diagonal cells.

The first heatmap in Fig. 36 illustrates the correlation of feature visualizations of various layers in AlexNet. The greener values in the heatmap indicate higher positive correlation values. Higher correlations are observed within a block of layers in the center of the matrix, or layers that are close next to each other, indicating that these layers' features have stronger relationships.

In Fig. 37, the second heatmap illustrates the correlation of feature visualizations of convolutional layers within VGG16. This is a noticeable diagonal pattern of high correlations, which is expected as it represents the correlation of each layer with other layers next to it.

7 Discussion

This chapter presents a discussion of the thesis results, with a particular focus on the application of feature visualizations on the fine-tuned models in the ISIC 2019 benchmark dataset for multiclass classification and 2020 for binary classification. This chapter also highlights the potential for future research and emphasizes the importance of this study in advancing the understanding of the inner workings of deep neural networks in the medical domain. This analysis aims to provide a comprehensive understanding of the study's implications and the possibilities it offers for further investigation.

7.1 Interpretation of Results

The achieved results are positive, despite the significant imbalance issue between positive and negative samples of the binary classification dataset, the imbalance issue between minority and majority classes in multiclass classification, the absence of ground truth labels for both datasets, the large number of layers for VGG16 and especially for ResNet50 during feature visualization application, the lack of studies of feature visualizations in the medical domain, and the absence of quantitative metrics to measure the results of feature visualizations. However, there is significant potential for improvement in evaluating multiclass models on unseen data and quantitatively analyzing feature visualizations.

During binary classification, the comparative analysis of AlexNet, VGG16, and ResNet50, explains the performance of these architectures on the ISIC 2020 benchmark dataset. VGG16 exhibits superior proficiency during the training phase, boasting the highest accuracy (92.6%) and a lower loss (0.078), which may be attributed to its deeper architecture conducive to more complex feature extraction. Nevertheless, this does not translate into commensurate performance in the testing phase, where it falls short with an accuracy of 69%. However, ResNet50, despite exhibiting slightly lower training and validation accuracy than VGG16, demonstrates superior generalizability to the test data, achieving an accuracy of 83.2%. This underscores its ability to effectively balance bias and variance. Notably, the AUC values for all models are relatively similar, suggesting that while there are differences in accuracy, the overall ranking performance of the models remains largely consistent.

During multiclass classification, the results suggest a progressive enhancement in model performance with increased architectural complexity. Utilizing a manually curated test dataset introduces a variable that may not entirely parallel the complexity of the original test data, which is reflected in the modest AUC scores, with the highest value of 0.58. However, these scores remain critical in establishing the model's predictive capabilities. The incremental improvement in validation accuracy observed with the VGG16 model underscores the merit of deeper architectures and their ability to capture more abstract data representations.

After training the models for specific classes, the next step is the representation of the most activated filters in the final layers of the AlexNet, VGG16, and ResNet50 models that hold substantive significance for deep learning interpretability. By focusing on these filters, the feature visualization methods can be deployed, to explain the otherwise opaque decision-making processes of CNNs. This targeted approach avoids the redundancy of analyzing the entire neural network, concentrating instead on the most influential components - those filters that are the most responsive to stimuli across classification tasks. This elective visualization was not only computationally efficient during the application of feature visualization but also explains the patterns and abstractions that these preeminent filters have learned, which may be pivotal in advancing the model's interpretability and trustworthiness in practical applications.

In the presented results, feature visualizations of both pre-trained and fine-tuned models for binary and multiclass classification illuminate the transformative effects of model training on feature extraction, especially with different regularization techniques effectively used for the medical domain. The pre-trained model's visualizations reveal essential patterns that the network is innately sensitive to, which mostly reflect the general visual features of the ImageNet dataset. Conversely, the fine-tuned visualizations display a marked change, with the top-activated filters manifesting distinct patterns characteristic of the specialized dataset they were trained on, namely medical skin cancer images. Their adaption of learned representations to the specific characteristics of the skin cancer dataset is evidenced by the presence of some visible patterns that align with the textual features inherent to skin cancer imagery, such as border irregularity, structural asymmetry, and circular black and brown shapes - critical indicators in melanoma detection. However, quantitatively discerning whether these visualizations represent features indicative of skin cancer poses a considerable challenge, as it is crucial to analyze them accurately.

During the application of feature visualization in fine-tuned models, there is a presence of certain patterns from the pre-trained ImageNet models within the fine-tuned models, which implies that some fundamental visual features are universally applicable across domains, serving as foundational building blocks for the model's learning process.

Comparing the different feature visualizations applied for binary and multiclass tasks in fine-tuned models is seen as a tendency to capture different patterns. The feature visualizations from the binary classification tasks reveal filters that focus on stark, high-contrast features likely corresponding to distinctions between benign and malignant lesions. However, the multiclass task visualizations exhibit a broader spectrum of patterns, suggesting an indication of textural details, potentially to differentiate among various skin lesion types.

Furthermore, while comparing the feature visualizations between different layers of the fine-tuned models, the visualizations from the initial layer exhibit primitive feature detection, capturing basic contrasts and edge detection. As we transition to the final layer, there is an observable shift towards more complex pattern recognition. During the sanity check, it was observed that when correlating feature visualizations with natural samples of the same class across different layers of the models in a binary classification task, there is an increasing similarity between feature visualizations and the natural images when moving across the depth of the network layers for AlexNet, VGG16, and ResNet50. This trend reinforces the approach that the deeper the layer, the more refined the features it captures, but still the correlated values are below the average. When correlating between feature visualizations of different layers in a specific model, the correlation matrices highlight a nuanced intra-layer and inter-layer relationship, with early layers showing higher inter-correlation, reflecting shared primitive features, while deeper layers exhibit more specialized and less correlated feature representations. This pattern indicates a complex transformation from general to particular features.

The comparison of feature visualizations from the last convolutional layers of AlexNet, VGG16, and ResNet50 reveals features that those networks have learned, with VGG16's filters displaying a pattern complexity that might closely resemble the characteristic textures of skin cancer imagery. This suggests that VGG16's architecture may provide a more nuanced extraction of features relevant to the recognition of oncological patterns in dermatological images.

To summarize the findings in response to the research questions posed in the introduction, the results indicate that using deeper architectures like ResNet50 and VGG16 during the training phase can lead to higher accuracy due to their capacity for more complex feature extraction. Utilizing techniques like dropout, L1 regularization, and data augmentation can effectively improve the generalization of the model on unseen data. Another training strategy that highly contributed to the performance of the models was fine-tuning pre-trained models on the ISIC dataset, which helps to leverage learned features from large datasets and adapt them to specific tasks, enhancing efficiency and model performance.

The method of applying feature visualizations is answered with two techniques: selective visualization and layer-by-layer analysis. Focusing on the most activated filters of the last layer of the models can clarify the decision-making process of the CNNs. This avoids the redundancy of visualizing the less significant features. Through a layer-by-layer analysis technique, feature visualization was progressively performed to gain insight into the evolution of features from simple to complex patterns.

The third research question aimed to explore various regularization techniques that can be used in feature visualization. These techniques help the model avoid overfitting the training data, which enables it to generalize better and produce cleaner, more interpretable visualizations. This is especially important in the medical domain, where the visualizations need to be precise and easy to interpret.

The methods used to visualize features required quantification due to their difficulty in being qualitatively interpreted. Those methods include correlation analysis and sanity check. Comparing feature visualizations with natural images of the same class and analyzing correlation matrices between different layers may quantitatively measure how feature representations evolve across the network. However, the low values of the similarity index, indicate that the investigated feature visualizations of all three models, do not pass completely the sanity check. This suggests that feature visualizations only partially capture the essence of natural images belonging to the same class.

For the last research question, analyzing activation patterns and comparing feature responses across binary and multiclass classification tasks can distinguish between filters responding to outlier features and those capturing class-specific details, enhancing the identification of filter specificity. Notably, it is observed that the most activated filters of the outlier dataset are almost similar to the most activated filters of the binary dataset. This occurrence might be attributed to the inclusion of outlier images within the binary dataset, underscoring their influence on filter activations and suggesting a shared representation between the two datasets. Additionally, comparing whether the applied feature visualizations include outlier features can be challenging.

8 Conclusion

In conclusion, this master's thesis underscores the pivotal role of feature visualization in enhancing interpretability within the medical domain, particularly in dermatological oncology. The results demonstrate that VGG16's architecture while excelling in training performance, facilitates detailed feature extraction crucial to recognizing complex textures in skin cancer imagery. Conversely, ResNet50's superior generalization capabilities underscore its robustness across diverse datasets, and AlexNet is the least-performing model.

Through the application of feature visualization, both pre-trained and fine-tuned models reveal distinct patterns by providing visual insights into the neural network's learning process. This enhances understanding of how these complex networks interpret and analyze medical images, contributing significantly to educational practices in medical imaging. Furthermore, the educational application of this understanding aims to develop more robust networks. By grasping how these models learn from medical images, the training process can be refined to achieve peak performance, which is crucial for improving diagnostic accuracy and reliability in medical applications.

However, relying solely on feature visualization may not fully capture what the network is learning, particularly in the medical domain where similar patterns may appear across different conditions. Furthermore, the resulting feature visualizations are challenging to interpret visually. It is crucial to combine feature visualization with other interpretability methods, such as attribution techniques, to obtain a more comprehensive understanding of the learning process.

9 Future Work

Despite the significant advancements in feature visualization and its applications within the medical domain, there remain several gaps and opportunities for further research that can contribute substantially to understanding and utilizing deep learning models in healthcare.

1. Quantification and standardization of feature visualizations: While most of the studies offer qualitative insights into model decisions, there is a lack of standardized methods for quantifying the interpretability and relevance of these visualizations. The use of the Network Dissection method could solve this problem, but it requires pixel-wise labeled datasets, and this was one of the main obstacles faced by the dataset used in this master's thesis (Bau et al., 2017). Furthermore, developing novel quantification methods for medical data could significantly impact the measurement and analysis of complex image features.

2. Advanced attribution methods such as Layer-wise Relevance Propagation (LRP), saliency maps, GradCam and other attribution methods could be integrated with feature visualization in future studies (Molnar, 2022). This integration aims to enhance the interpretability of neural network decisions in medical applications.

3. Future research could compare models trained from scratch with pre-trained models within the medical imaging domain. The presence of certain patterns from the pre-trained ImageNet models within the fine-tuned models, which was observed during the feature visualization application, can be overcome by using trained models from scratch.

A Appendix

A.1 Results of binary and multiclass classification

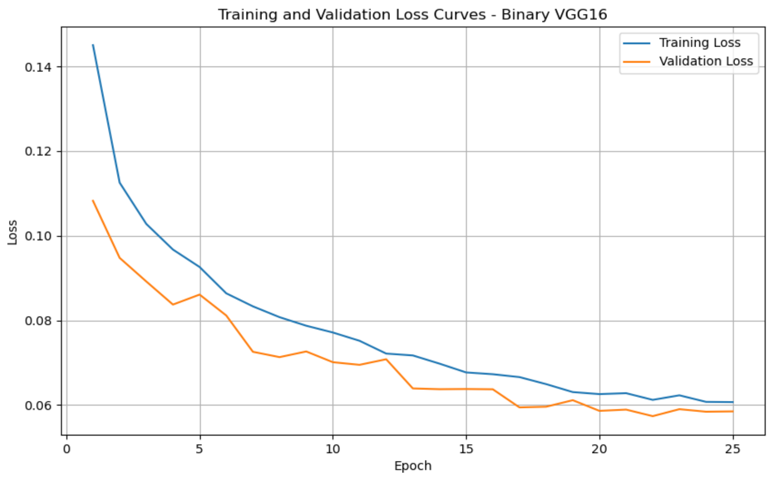


Figure 38: VGG16 Binary classification: Convergence of Training and Validation Loss over epochs. Both losses are decreasing, which suggests the model is learning effectively without overfitting significantly, as the validation loss is close to the training loss.

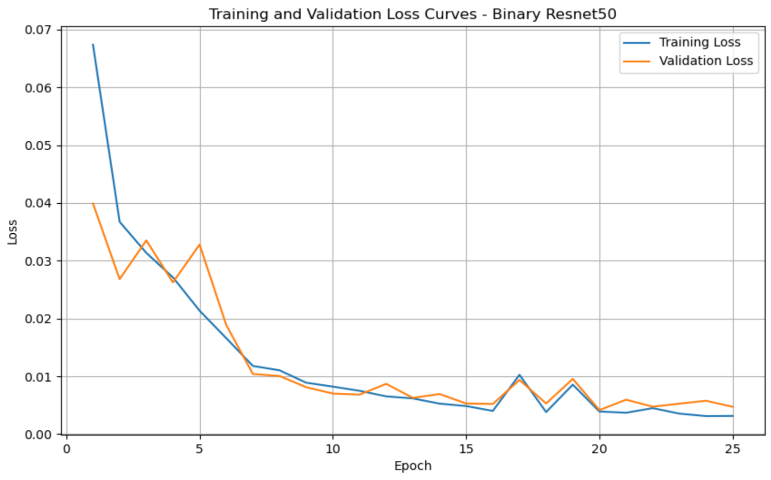


Figure 39: ResNet50 Binary classification: Convergence of Training and Validation Loss over epochs. The training loss decreases sharply and then levels off, while the validation loss decreases and then shows some fluctuation, indicating possible instability in learning as the model may be learning noise from the training data or the hyperparameters need adjustments.

The confusion matrix in Fig. 40, Fig. 41 and Fig. 42 indicate the performance of the three models on validation data for binary classification, highlighting the counts of true positives, true negatives, false positives, and false negatives.

The images in Fig. 43 illustrate the accuracy curves of VGG16 and ResNet50 trained for binary classification tasks over a series of epochs. The VGG16 image in Fig. 43 shows both accuracies improve over time with some fluctuation in validation accuracy, while for ResNet50 both accuracies increase sharply initially and then stabilize, indicating the model's learning effectiveness over epochs. Validation accuracy shows some fluctuations over epochs, indicating any noise and outliers or the model might see slightly different versions of the validation data in each epoch, because of data augmentation used, leading to variability in accuracy.

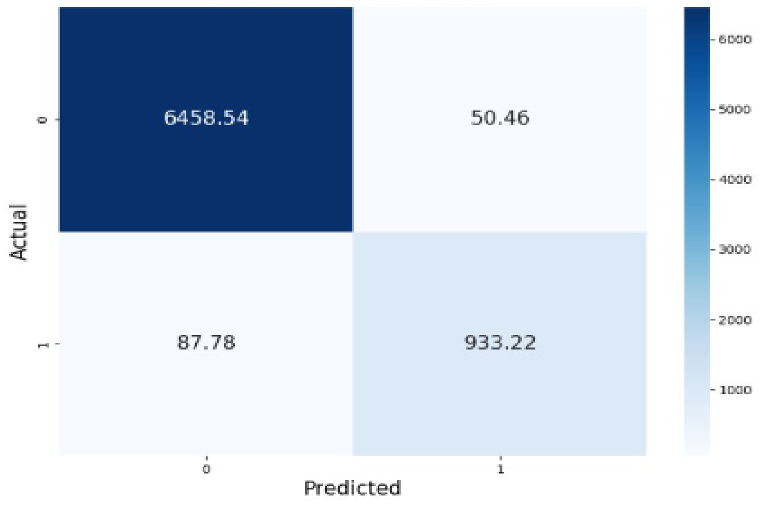


Figure 40: Binary classification confusion matrix of AlexNet on validation data.

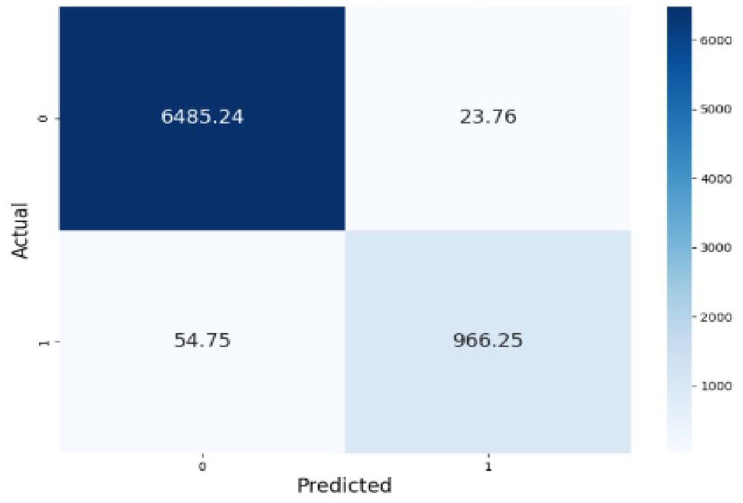


Figure 41: Binary classification confusion matrix of VGG16 on validation data.

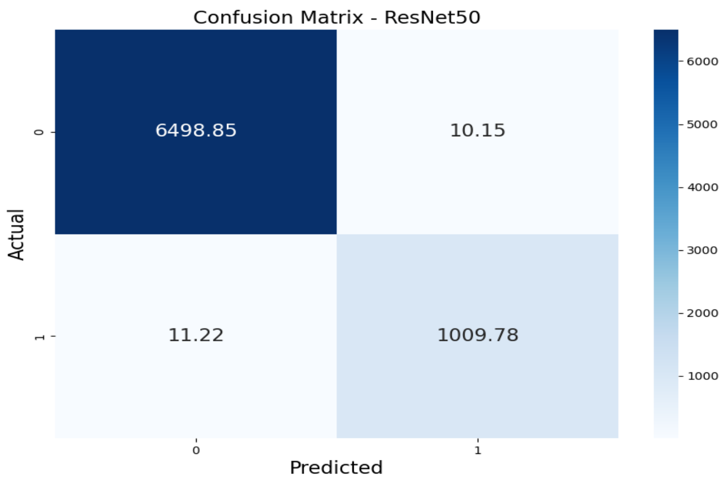


Figure 42: Binary classification confusion matrix of ResNet50 on validation data.

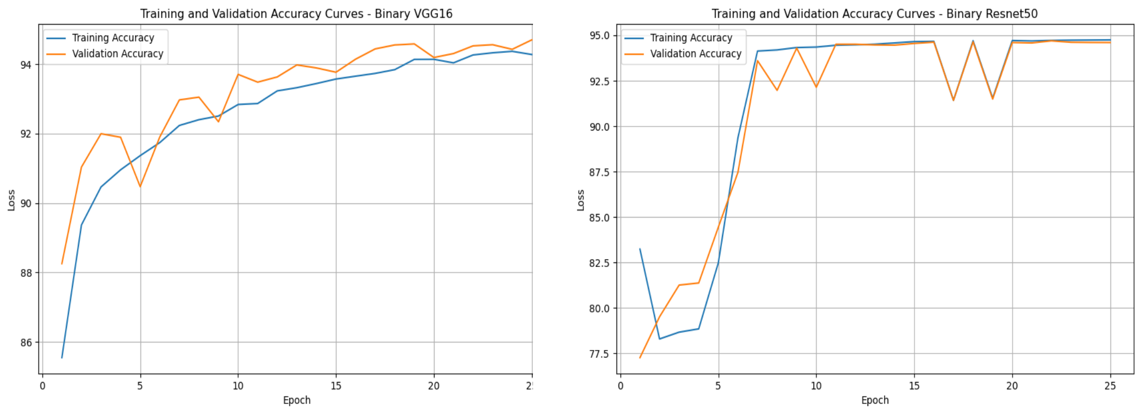


Figure 43: Accuracy curves for VGG16 and ResNet50, for binary classification.

A.2 Qualitative results of feature visualizations

The Fig. 44, Fig. 45, and Fig. 46 show the feature visualizations of pre-trained AlexNet and ResNet50, together with feature visualizations of fine-tuned ResNet50. The filters selected are the most activated filters of ResNet50 introduced in Table 10.

Other feature visualizations shown in Fig.47, and Fig. 48 highlight the features learned by the layers in the specific filters (random filters) for malignant class, both for AlexNet and VGG16. As visually seen the feature visualizations of the last layer, in the second row, for the AlexNet model are almost the same for most filters.

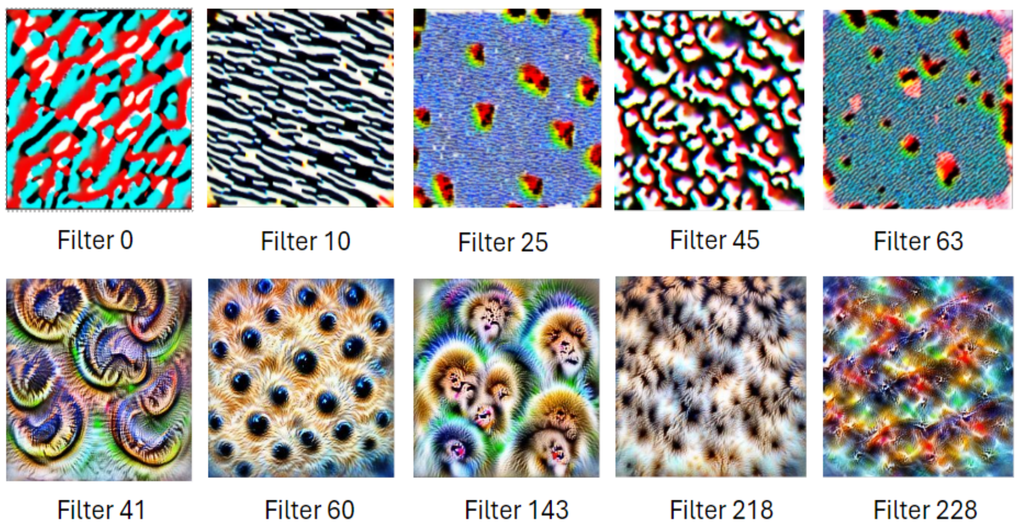


Figure 44: Feature Visualization of AlexNet pre-trained model, binary classification. The first row represents the first convolutional layer, and the second row the last convolutional layer.

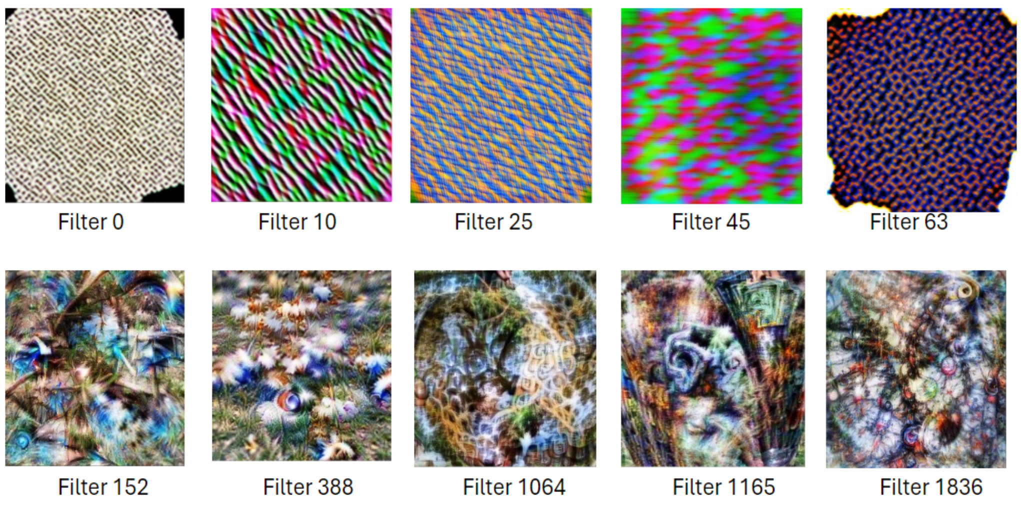


Figure 45: Feature Visualization of ResNet50 pre-trained model, binary classification. The first row represents the first layer, and the second row the last layer.

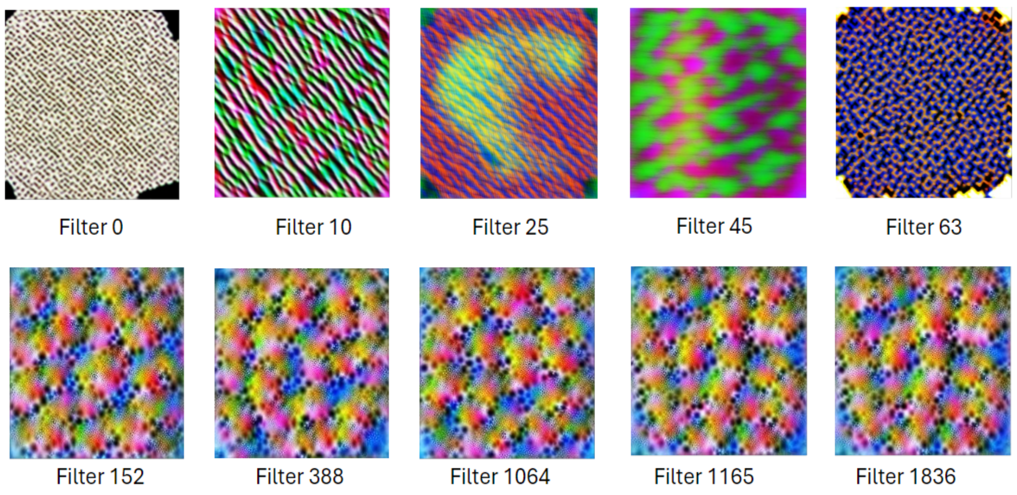


Figure 46: Feature Visualization of ResNet50 fine-tuned model, binary classification. The first row represents the first layer, and the second row the last layer.

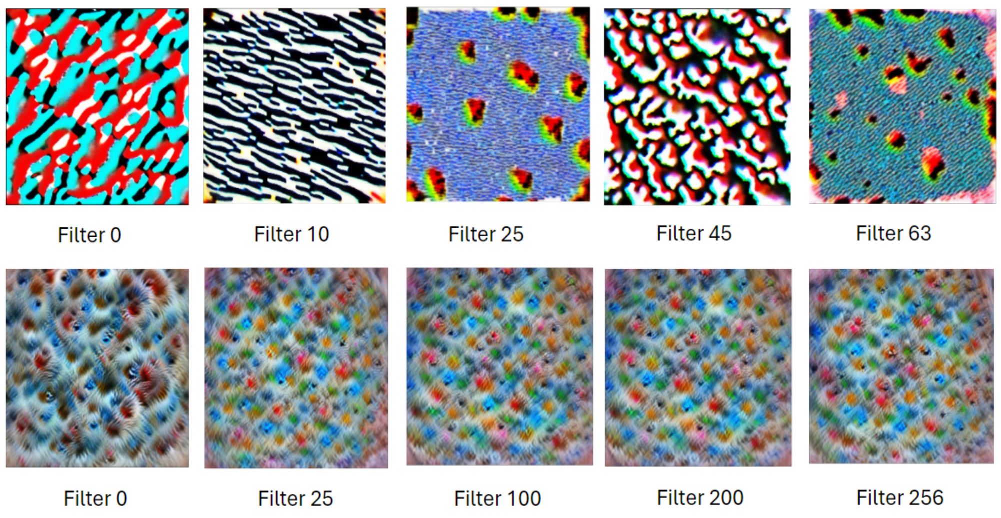


Figure 47: Feature Visualization AlexNet fine-tuned model, binary classification, malign class. The first row represents the first layer, and the second row the last layer.

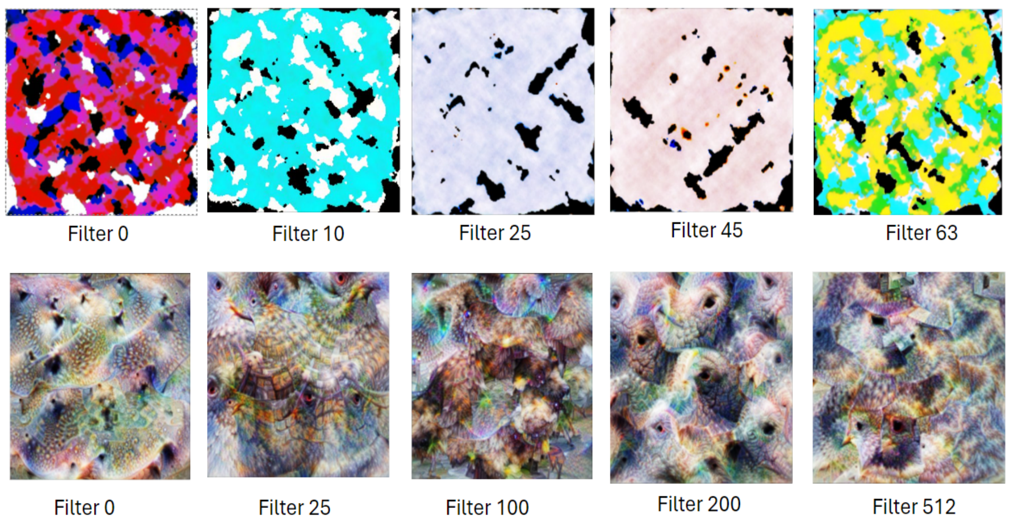


Figure 48: Feature Visualization of VGG16 fine-tuned model, binary classification, malign class. The first row represents the first layer, and the second row the last layer.

Bibliography

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf`.

Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.

S.H. Shabbeer Basha, Shiv Ram Dubey, Viswanath Pulabaigari, and Snehasis Mukherjee. Impact of fully connected layers on the performance of convolutional neural networks for image classification. *Neurocomputing*, 378: 112–119, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.10.008. URL `https://www.sciencedirect.com/science/article/pii/S0925231219313803`.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations, 2017.

Pytorch BCELoss. Bceloss. 2023. URL `https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html`.

Oded Ben-David and Zohar Ringel. The role of a layer in deep neural networks: a gaussian process perspective, 2019.

Judy Borowski, Roland S. Zimmermann, Judith Schepers, Robert Geirhos, Thomas S. A. Wallis, Matthias Bethge, and Wieland Brendel. Exemplary natural images explain cnn activations better than state-of-the-art feature visualization, 2021.

Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. ISSN 0031-3203. doi: https://doi.org/10.1016/S0031-3203(96)00142-2. URL `https://www.sciencedirect.com/science/article/pii/S0031320396001422`.

CancerResearchUK. How does the sun and UV cause cancer?, 2023. [Online; accessed 2023].

Bill Cassidy, Connah Kendrick, Andrzej Brodzicki, Joanna Jaworek-Korjakowska, and Moi Hoon Yap. Analysis of the isic image datasets: Usage, benchmarks and recommendations. *Medical Image Analysis*, 75:102305, 2022. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2021.102305. URL `https://www.sciencedirect.com/science/article/pii/S1361841521003509`.

Heang-Ping Chan, Ravi K. Samala, Lubomir M. Hadjiiski, and Chuan Zhou. *Deep Learning in Medical Image Analysis*. Springer International Publishing, 2020. doi: 10.1007/978-3-030-33128-3_1. URL `https://doi.org/10.1007/978-3-030-33128-3_1`.

Minshuo Chen, Yu Bai, Jason D. Lee, Tuo Zhao, Huan Wang, Caiming Xiong, and Richard Socher. Towards understanding hierarchical learning: Benefits of neural representations, 2021.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning, 2014.

Alexander Mordvintsev Michael Petrov Christopher Olah, Ludwig Schubert. A collection of infrastructure and tools for research in neural network interpretability., 2017. URL `https://github.com/tensorflow/lucid`.

Noel Codella, Quoc-Bao Nguyen, Sharath Pankanti, David Gutman, Brian Helba, Allan Halpern, and John R. Smith. Deep learning ensembles for melanoma recognition in dermoscopy images, 2016.

Noel Codella, Veronica Rotemberg, Philipp Tschandl, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). 2018a.

Noel C. F. Codella, David Gutman, M. Emre Celebi, et al. 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). 2017.

Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, and Allan Halpern. 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). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172, 2018b. doi: 10.1109/ISBI.2018.8363547.

Marc Combalia, Noel C. F. Codella, Veronica Rotemberg, et al. Bcn20000: Dermoscopic lesions in the wild. 2019.

Sofie A.E. De Hertog, Christianne A.H. Wensveen, Maarten T. Bastiaens, Christine J. Kielich, Marjo J.P. Berkhout, Rudi G.J. Westendorp, Bert J. Vermeer, and Jan N. and Bouwes Bavinck. Relation between smoking and skin cancer. *Journal of Clinical Oncology*, 19(1):231–238, 2001. doi: 10.1200/JCO.2001.19.1.231. URL `https://doi.org/10.1200/JCO.2001.19.1.231`. PMID: 11134217.

Mayukh Deb. Feature visualization library for pytorch. `https://github.com/Mayukhdeb/torch-dreams`, 2021.

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

Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841, 2018. doi: 10.1109/CCDC.2018.8407425.

Din Mohammad Dohan. Skin cancer detection - cnn models: Resnet9 resnet34. 2023. doi: https://www.kaggle.com/code/jonidamukaj/skin-cancer-detection-cnn-models-resnet9-resnet34/edit.

Fabian Eitel, Anna Melkonyan, and Kerstin Ritter. Feature visualization for convolutional neural network models trained on neuroimaging data, 2022.

Ismail Elansary, Amr Ismail, and Wael Awad. Efficient classification model for melanoma based on convolutional neural networks. *Studies in Computational Intelligence*, 2021. URL `https://api.semanticscholar.org/CorpusID:245232764`.

Dumitru Erhan, Y. Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*, 01 2009.

Thomas Fel, Thibaut Boissin, Victor Boutin, Agustin Picard, Paul Novello, Julien Colin, Drew Linsley, Tom Rousseau, Rémi Cadène, Laurent Gardes, and Thomas Serre. Unlocking feature visualization for deeper networks with magnitude constrained optimization, 2023.

Robert Geirhos, Roland S. Zimmermann, Blair Bilodeau, Wieland Brendel, and Been Kim. Don't trust your eyes: on the (un)reliability of feature visualizations, 2023.

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

Fabio Graetz. How to visualize convolutional features in 40 lines of code. 2019. URL `https://towardsdatascience.com/how-to-visualize-convolutional-features-in-40-lines-of-code-70b7d87b0030`.

Qishen Ha, Bo Liu, and Fuxu Liu. Identifying melanoma images using efficientnet ensemble: Winning solution to the siim-isic melanoma classification challenge, 2020.

Millman K. Jarrod van der Walt Stéfan J. Gommers Ralf Virtanen Pauli Cournapeau David Wieser Eric Harris, Charles R. Array programming with numpy. 2020. doi: https://doi.org/10.1038/s41586-020-2649-2.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

K Henriksen, K Stamnes, G Volden, and ES Falk. Ultraviolet radiation at high latitudes and the risk of skin cancer. *Photo-dermatology*, 6(3):110—117, June 1989. ISSN 0108-9684. URL `http://europepmc.org/abstract/MED/2762201`.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

Daniel T Huff, Amy J Weisman, and Robert Jeraj. Interpretation and visualization techniques for deep learning models in medical imaging. *Physics in Medicine Biology*, 66(4):04TR01, feb 2021. doi: 10.1088/1361-6560/abcd17. URL `https://dx.doi.org/10.1088/1361-6560/abcd17`.

John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

ISICLeaderboard. Isic live leaderboards. URL `https://challenge.isic-archive.com/leaderboards/live/`.

Konstantinos Kamnitsas, Christian Ledig, Virginia F.J. Newcombe, Joanna P. Simpson, Andrew D. Kane, David K. Menon, Daniel Rueckert, and Ben Glocker. Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78, 2017. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2016.10.004. URL `https://www.sciencedirect.com/science/article/pii/S1361841516301839`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Wolff K Binder M. Kittler H, Pehamberger H. Diagnostic accuracy of dermoscopy. lancet oncol. 2002. URL `https://doi.org/10.1016/s1470-2045(02)00679-4`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL `https://doi.org/10.1145/3065386`.

Chandrashekar Lakshminarayanan and Amit Vikram Singh. Neural path features and neural path kernel : Understanding the role of gates in deep learning, 2021.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Michael Petrov Lim Swee Kiat, Nolan Dey. Lucid library adapted for pytorch, 2020. URL `https://github.com/greentfrapp/lucent`.

K. Mahalakshmi and P. Sujatha. The role of exploratory data analysis and preprocessing in the machine learning predictive model for heart disease. In *2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, pages 1–8, 2023. doi: 10.1109/ACCAI58221.2023.10199714.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196, 2015. doi: 10.1109/CVPR.2015.7299155.

Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, May 2016. ISSN 1573-1405. doi: 10.1007/s11263-016-0911-8. URL `http://dx.doi.org/10.1007/s11263-016-0911-8`.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254. URL `https://doi.org/10.1145/1873951.1874254`.

Sheldon Mascarenhas and Mukul Agarwal. A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification. In *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, volume 1, pages 96–99, 2021. doi: 10.1109/CENTCON52345.2021.9687944.

MelanomaSkinCancer. Melanoma skin cancer, 2023. [Online; accessed 2023].

MelanomaUK. 2020 MELANOMA SKIN CANCER REPORT, 2020. [Online; accessed 2020].

Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022. URL `https://christophm.github.io/interpretable-ml-book`.

A. Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015a. URL `https://api.semanticscholar.org/CorpusID:69951972`.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015b. URL `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`.

Kori A. Natekar P. and Krishnamurthi G. (2020). Front. Comput. Neurosci. Demystifying brain tumor segmentation networks: Interpretability and uncertainty analysis, 2020. URL `https://www.frontiersin.org/articles/10.3389/fncom.2020.00006/full`.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015. doi: 10.1109/CVPR.2015.7298640.

Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks, 2016a.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding neural networks via feature visualization: A survey, 2019.

Anh M Nguyen, Jason Yosinski, and Jeff Clune. Understanding innovation engines: Automated creativity and improved stochastic optimization via deep learning. *Evolutionary Computation*, 24:545–572, 2016b. URL `https://api.semanticscholar.org/CorpusID:27515105`.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2013.

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. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Sérgio Pereira, Raphael Meier, Victor Alves, Mauricio Reyes, and Carlos Alberto Silva. Understanding and interpreting machine learning in medical image computing applications. In *Lecture Notes in Computer Science*, 2018. URL `https://api.semanticscholar.org/CorpusID:52825344`.

Fábio Perez, Cristina Vasconcelos, Sandra Avila, and Eduardo Valle. *Data Augmentation for Skin Lesion Analysis*, page 303–311. Springer International Publishing, 2018. ISBN 9783030012014. doi: 10.1007/978-3-030-01201-4_33. URL `http://dx.doi.org/10.1007/978-3-030-01201-4_33`.

Bhanu Prasanna. Isic 2019. 2020. doi: https://www.kaggle.com/datasets/bhanuprasanna/isic-2019/data.

Pytorch ReduceOnPlateau. Reducelronplateau. 2023. URL `https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html`.

Vered Rotemberg, Natali Kurtansky, Brigid Betz-Stablein, and et al. Siim-isic 2020 challenge dataset. `https://doi.org/10.34970/2020-ds01`, 2020. Creative Commons Attribution-Non Commercial 4.0 International License. The dataset includes images from Hospital Clínic de Barcelona, Medical University of Vienna, Memorial Sloan Kettering Cancer Center, Melanoma Institute Australia, The University of Queensland, and the University of Athens Medical School. License information: `https://creativecommons.org/licenses/by-nc/4.0/legalcode.txt`.

Vered Rotemberg, Natali Kurtansky, Brigid Betz-Stablein, and et al. A patient-centric dataset of images and metadata for identifying melanomas using clinical context. *Scientific Data*, 8:34, 2021. doi: 10.1038/s41597-021-00815-z. URL `https://doi.org/10.1038/s41597-021-00815-z`.

Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

Olga Russakovsky, Jia Deng, Zhiheng Huang, Alexander C. Berg, and Li Fei-Fei. Detecting avocados to zucchinis: What have we done, and where are we going? In *2013 IEEE International Conference on Computer Vision*, pages 2064–2071, 2013. doi: 10.1109/ICCV.2013.258.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia Computer Science*, 132:377–384, 2018. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.05.198. URL `https://www.sciencedirect.com/science/article/pii/S1877050918309335`. International Conference on Computational Intelligence and Data Science.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

Amitojdeep Singh, Sourya Sengupta, and Vasudevan Lakshminarayanan. Explainable deep learning models in medical image analysis. *Journal of Imaging*, 6 (6), 2020. ISSN 2313-433X. doi: 10.3390/jimaging6060052. URL `https://www.mdpi.com/2313-433X/6/6/52`.

SkinCancerFoundation. Can your diet help prevent skin cancer? 2022. URL `https://www.skincancer.org/blog/can-your-diet-help-prevent-skin-cancer/`.

SkinCancerFoundation. Skin Cancer Facts Statistics, 2024. [Online; accessed February 2024].

Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors, 2022.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5:180161, 2018. doi: 10.1038/sdata.2018.161.

Maximilian E. Tschuchnig and Michael Gadermayr. Anomaly detection in medical imaging - a mini review. In Peter Haber, Thomas J. Lampoltshammer, Helmut Leopold, and Manfred Mayr, editors, *Data Science – Analytics and Applications*, pages 33–38, Wiesbaden, 2022. Springer Fachmedien Wiesbaden.

M. Tyka. Class visualization with bilateral filters. 2016. URL `https://mtyka.github.io/deepdream/2016/02/05/bilateral-class-vis.html`.

Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL `https://doi.org/10.21105/joss.03021`.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

WorldCancerResearch. Skin cancer statistics. 2022. URL `https://www.wcrf.org/cancer-trends/skin-cancer-statistics/`.

Xiaopeng Yang, Jae Do Yang, Hong Pil Hwang, Hee Chul Yu, Sungwoo Ahn, Bong-Wan Kim, and Heecheon You. Segmentation of liver and vessels from ct images and classification of liver segments for preoperative liver surgical planning in living donor liver transplantation. *Computer Methods and Programs in Biomedicine*, 158:41–52, 2018. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2017.12.008. URL `https://www.sciencedirect.com/science/article/pii/S0169260717303383`.

Lecun Yann. Mnist dataset. URL `https://www.kaggle.com/datasets/hojjatk/mnist-dataset`.

Darvin Yi, Rebecca Lynn Sawyer, David Cohn III au2, Jared Dunnmon, Carson Lam, Xuerong Xiao, and Daniel Rubin. Optimizing and visualizing deep learning for benign/malignant classification in breast tumors, 2017.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization, 2015.

Lequan Yu, Hao Chen, Qi Dou, Jing Qin, and Pheng-Ann Heng. Automated melanoma recognition in dermoscopy images via very deep residual networks. *IEEE Transactions on Medical Imaging*, 36(4):994–1004, 2017a.

Wen Yu, Baiying Lei, Yanyan Shen, Shuqiang Wang, Yong Liu, Zhiguang Feng, Yong Hu, and Michael K. Ng. Morphological feature visualization of alzheimer's disease via multidirectional perception gan, 2021.

Zhen Yu, Xudong Jiang, Tianfu Wang, and Baiying Lei. Aggregating deep convolutional features for melanoma recognition in dermoscopy images. In Qian Wang, Yinghuan Shi, Heung-Il Suk, and Kenji Suzuki, editors, *Machine Learning in Medical Imaging*, pages 238–246, Cham, 2017b. Springer International Publishing.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning, 2023.

Fan Zhang, Zhenzhen Li, Boyan Zhang, Haishun Du, Binjie Wang, and Xinhong Zhang. Multi-modal deep learning model for auxiliary diagnosis of alzheimer's disease. *Neurocomputing*, 361, 07 2019. doi: 10.1016/j.neucom.2019.04.093.

Roland S. Zimmermann, Judy Borowski, Robert Geirhos, Matthias Bethge, Thomas S. A. Wallis, and Wieland Brendel. How well do feature visualizations support causal understanding of cnn activations?, 2021.

Declaration of Authorship

Ich erkläre hiermit gemäß §9 Abs. 12 APO, dass ich die vorstehende Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass die digitale Fassung der gedruckten Ausfertigung der Abschlussarbeit ausnahmslos in Inhalt und Wortlaut entspricht und zur Kenntnis genommen wurde, dass diese digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Place, Date	Signature