THESIS

Kumar Dheeraj

Debrecen
2023

University of Deberecen

Faculty of Informatics

Identifying Cancer using MRI and CT scans

Supervisor:
Dr. Tomán Henrietta
Associate Professor

Candidate:
Kumar Dheeraj
Computer Science BSc

Debrecen

2023

Contents

Abstract

Medical imaging plays a pivotal role in the early detection and diagnosis of cancer, including Renal Cell Carcinoma (RCC) or Kidney Cancer, which is among the top ten most common cancers worldwide, with over 430,000 new cases diagnosed and 180,000 deaths each year (Science Direct, 2021 and Our World in Data, 2019). Often, kidney cancer remains undetected until advanced stages due to its subtle symptoms, making early diagnosis crucial for improving survival rates. The 5-year survival rate for patients with localized kidney cancer is 93%, but it drops to 15% when cancer has metastasized (American Cancer Society, 2021).

With recent advancements in Machine Learning (ML) and image processing techniques, significant progress has been made in the domain of cancer detection, potentially saving lives through early diagnosis and timely treatment. This thesis focuses on the creation and assessment of machine learning models for detecting cancer in Kidney Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) scan images, placing a strong emphasis on meticulous preprocessing to ensure high image quality and accuracy.

The study began with the selection of an appropriate public dataset, prioritizing minimal noise, high image quality, and favorable image statistics. An extensive preprocessing pipeline was implemented using OpenCV, a preprocessing tool, to further refine the images, encompassing noise removal, denoising, edge detection and blending, histogram equalization, data augmentation, and normalization. The processed images were detailed and well-suited for training machine learning models with good accuracy.

Various machine learning models were trained and evaluated based on accuracy and resource utilization. An Artificial Neural Network (ANN) was initially applied, yielding unsatisfactory accuracy. Subsequently, a Support Vector Machine (SVM) was utilized, achieving a 96.79% accuracy with relatively low resource consumption. VGG16, a type of Convolutional Neural Network (CNN), resulted in a 93.17% accuracy but necessitated an extensive training period. Although the focus of this research was the analysis of the three aforementioned models, the Swin Transformer, a recent and accurate model, was examined and achieved a 96.58% accuracy, albeit with a prolonged training duration.

The findings of this study indicate that the **Support Vector Machine (SVM)** is the most efficient and accurate model for detecting cancer in Kidney Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) scan images among the assessed models, underscoring the significance of image preprocessing in boosting the performance of machine learning models. The development of such models is crucial for enhancing early cancer detection and reducing cancer-related deaths. Future research should consider evaluating the performance of additional machine learning models, delving into the potential of alternative preprocessing techniques, and broadening the scope to encompass other types of medical images and cancer detection tasks.

Chapter 1

Introduction

1.1 Background and Significance

Cancer claims the lives of millions of people every year, making it one of the primary causes of death globally[1]. The disease's deadly nature comes from its ability to proliferate unchecked and infiltrate nearby tissues, often metastasizing to remote organs before being discovered. Early diagnosis is crucial to enhance patient outcomes and survival rates, which emphasizes the necessity for precise and efficient diagnostic techniques. Kidney cancer is especially fatal compared to other cancers due to its rapid progression and lack of symptoms during the early stages[2][3].

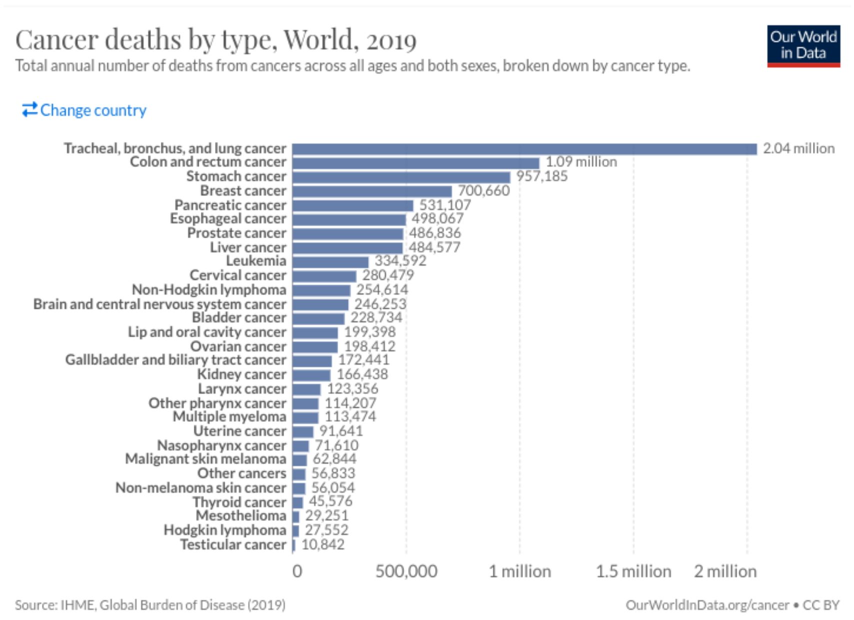


Figure 1.1: Number of deaths recorded by cancer types in 2019[1].

As illustrated in **Fig: 1.1**, kidney cancer accounted for about 200,000 deaths in 2019, making it one of the ten most lethal diseases[1]. Although its mortality rate is not the highest, kidney cancer is viewed as the most deadly due to its proclivity to go unnoticed until advanced stages. By the time it's discovered, it is often too late for effective treatment[2].

Statistics: Kidney cancer poses a considerable health risk for individuals of all genders. Men have approximately a 1 in 46 (2.02%) chance of encountering kidney cancer in their lives, while for women, this figure is about 1 in 80 (1.03%)[4][2]. The World Health Organization predicts that 2023 will see 81,800 new kidney cancer diagnoses in men and 29,440 in women, with an estimated 9,920 men and 4,970 women succumbing to the illness in the same year[4][5].

1.2 Motivation

Detecting kidney cancer at an early stage is difficult, as patients typically experience vague or no symptoms until the tumor has grown significantly[3]. Consequently, the five-year survival rate for patients diagnosed with advanced kidney cancer is considerably lower[3]. This highlights the need for improved diagnostic methods to detect kidney cancer earlier when treatments have a better chance of success[2].

Machine Learning (ML)[6] importance in medical image analysis and cancer diagnosis cannot be overstated[7]. ML algorithms can process and interpret vast amounts of intricate medical imaging data[8]. These algorithms can be designed to uncover hidden features and patterns in data without human intervention[7]. By using Machine Learning (ML)[6] to develop more accurate, efficient, and reliable diagnostic tools, researchers and medical professionals can enhance patient outcomes and overall healthcare quality[7].

The application of Machine Learning (ML)[6], specifically Deep Learning (DL)[9], to medical image analysis and the diagnosis of cancer and other diseases holds tremendous potential[7]. It can improve the speed, accuracy, and efficiency of disease detection, leading to better patient outcomes. When trained properly, ML algorithms[8] can accurately identify abnormalities in medical images, such as tumors[7]. In addition to processing and analyzing large volumes of medical images much faster than human experts, Artificial Intelligence (AI)[10] systems can also minimize false positives and negatives, complementing human expertise. This can aid doctors in making more accurate and timely diagnoses[7].

1.3 Research Objectives

The motivation to explore Machine Learning (ML) strategies for early kidney cancer diagnosis stems from personal experiences and observations in the field. Witnessing the devastating impact of cancer on patients and their families fuels a dedication to contribute to ongoing efforts to enhance cancer diagnostics. Kidney cancer's increasing prevalence and significant treatment costs[11] continue to pose a considerable public health challenge[3]. The rapid advancements in ML[6] and its successful application in various domains show immense promise for revolutionizing medical image analysis and early cancer detection.

By developing and evaluating kidney cancer-specific Machine Learning (ML) models[8] and utilizing Open Source Computer Vision Library (OpenCV)[12], this research aims to improve patient outcomes and reduce the societal burden of this life-threatening disease.

1.4 Contributions of this Thesis

This thesis makes several critical contributions to the field of kidney cancer detection using Machine Learning (ML) models[8] on Magnetic Resonance Imaging (MRI)[13] and Computed Tomography (CT)[14]. This research delves into the study of image preprocessing methods[15] and evaluates the relative effectiveness of various models:

1. **Literature Review**: The literature review in this thesis outlines the prevalence of kidney cancer and the primary challenges encountered when detecting the disease[3, 16]. This section also discusses various models that can be utilized for training the dataset, comparing their advantages and disadvantages[17, 18, 19, 20]. Following this comparison, the issues with previous models are examined, which paves the way for establishing the significance of this thesis[21]. The primary objective is to identify renal cancer in its early stages using Open Source Computer Vision Library (OpenCV), necessitating a highly accurate model.

2. **Advanced Image Preprocessing Techniques**: This investigation explores and implements various data cleansing and preprocessing strategies[15], such as removing outliers[22], diminishing noise[23], enhancing images[24], and applying histogram equalization[25]. These techniques are employed to improve the quality of MRI[13] and CT[14] scans. Subsequently, data augmentation[26] and normalization[27] are performed to produce new features and ensure uniformity in ranges or distributions across the dataset. This comprehensive preprocessing pipeline facilitates effective learning for ML models and may enhance their ability to detect kidney cancer.

3. **Model Comparison and Evaluation**: Multiple ML models[8] are trained[28] on the preprocessed dataset, and their efficacy and precision are compared. This work provides valuable insights into the performance of various models for kidney cancer detection, identifying the most promising approaches for future research and clinical application.

4. **Practical Implications and Future Research Directions**: The findings of this thesis have significant practical implications for the early detection and treatment of kidney cancer using MRI[13] and CT[14] scans. By focusing on preprocessing techniques[15] and model comparison, this research lays the groundwork for the creation of more effective diagnostic tools that can save lives and lessen the societal burden of this fatal disease. It also discusses the limitations of the work and suggests several promising future research directions in the area of image preprocessing and ML-based[6] detection of kidney cancer.

This thesis aims to significantly advance the field of kidney cancer detection using Magnetic Resonance Imaging (MRI)[13] and Computed Tomography (CT)[14] scans with the assistance of *Open Source Computer Vision Library (OpenCV)*[12] and *Machine Learning (ML)*[6], thereby improving diagnostic accuracy and patient outcomes.

1.5 Thesis Structure

This thesis is organized into six chapters, including this introductory section.

- **Chapter # 02:** This section presents a comprehensive **literature review**, offering an overview of existing models[8] for detecting kidney cancer. Additionally, a comparative analysis of these models[8] will be conducted to further elucidate their strengths and weaknesses.

- **Chapter # 03:** In this section, the focus is on the **data collection** process, which entails exploring public datasets for kidney cancer imaging, evaluating image quality, and ultimately finalizing the dataset to be employed in the study.

- **Chapter # 04:** This section addresses **data cleansing and preprocessing**[15], incorporating techniques such as outlier exclusion[22], image noise mitigation[23], and image improvement[24]. Furthermore, it delves into histogram equalization[25], data augmentation[26], and data normalization[27].

- **Chapter # 05:** In this section, the selected dataset is **trained**[28] using multiple models[8]. To determine the best approach for early kidney cancer detection, the **performance and efficiency** of each model[8] will be compared.
- **Chapter # 06:** This chapter provides a **summary** of the study's findings, emphasizing the importance of using ML techniques[6] for early kidney cancer detection and suggesting future research directions and potential improvements.

The block diagram below visually represents the structure of this thesis as shown in **Fig: 1.2**:

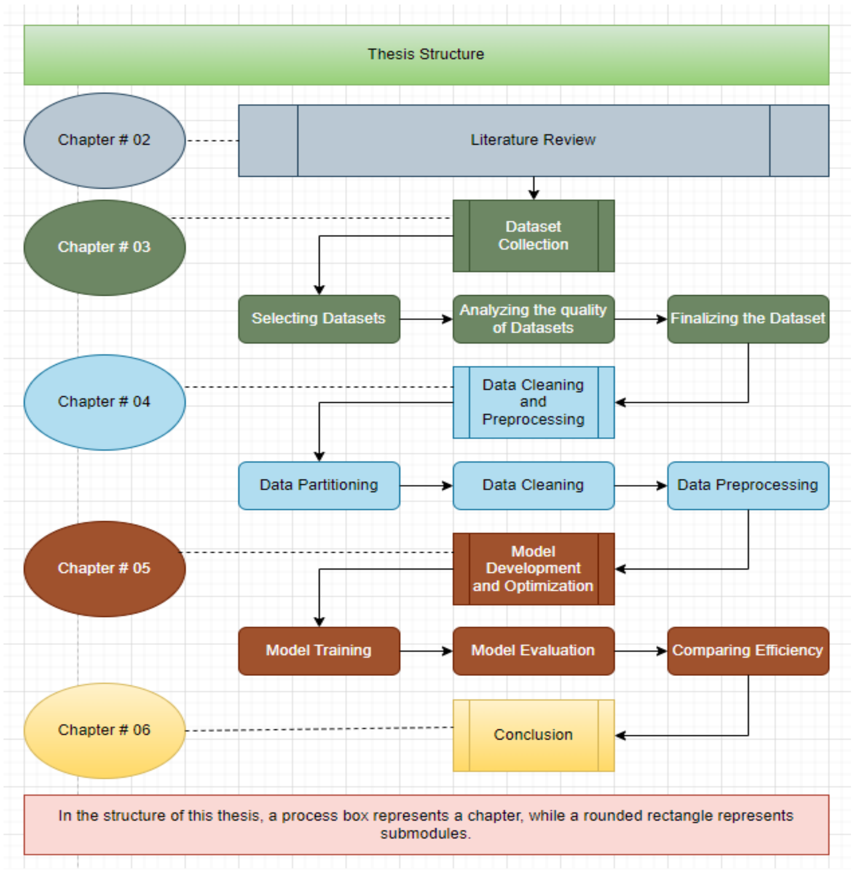


Figure 1.2: Thesis Structure

Chapter 2

Literature Review

2.1 Introduction

2.1.1 Medical Image Processing for Cancer Detection

Rapid advancements in medical imaging technology have revolutionized the diagnosis and treatment of various illnesses, including cancer[29]. Medical image processing plays a crucial role in assisting healthcare professionals in detecting, diagnosing, and monitoring cancer, as it provides accurate and comprehensive visuals of the body's internal structures[30]. Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) scans are particularly important tools for early cancer detection and characterization[30].

However, accurately identifying cancer through MRI[13] and CT scans[14] is a challenging task due to complex images, varying tumor appearances, and the need for meticulous examination by radiologists[29]. Recent research on medical errors estimates that errors may contribute to approximately 251,000 deaths per year in the U.S., making them the third leading cause of death[31].

2.1.2 Machine Learning Techniques in Medical Image Analysis

In light of the challenges discussed above, ML algorithms[8] have gained significant attention in medical image processing[29]. ML techniques such as Convolutional Neural Network (CNN)[32], Artificial Neural Network (ANN)[33], and Support Vector Machine (SVM)[34] have demonstrated promising results in various medical image analysis tasks, including cancer detection. Combining ML algorithms with OpenCV[12], a widely-used library for preprocessing, has the potential to create reliable and efficient solutions for cancer detection from medical images[35][29].

This literature review aims to examine the current state of research in medical image processing for cancer detection, with a particular focus on kidney cancer[29][30]. The review will explore existing techniques and methodologies employed in MRI[13] and CT scan[14] analysis for kidney cancer detection, as well as the challenges and limitations faced by these approaches. The emphasis will be on the application of ML algorithms[8] and OpenCV[12] to improve the accuracy and reliability of cancer detection in medical images[35]. Through this review, the aim is to identify gaps in the existing literature and establish a foundation for the research, which seeks to develop an innovative approach for kidney cancer detection using ML and OpenCV.

2.2 Search Strategy

To conduct a wide-ranging investigation of the current research landscape on medical image processing methods[29] for identifying kidney cancer[3] through MRI[13] and CT[14] scans, a methodical approach to literature exploration was employed. This strategy hinged on utilizing a multitude of electronic databases, featuring PubMed[36], IEEE Xplore[37], NIH[38], and Science Direct[39]. These databases were chosen due to their capacity to offer access to an eclectic mix of studies from various fields, such as medical imaging, computer science, and oncology, thereby guaranteeing an all-encompassing examination of pertinent literature.

The search strategy was devised by integrating an assortment of keywords and search terms relevant to the research subject in focus. The main keywords selected for this endeavor encompassed **Machine Learning, Medical Image Processing, Medical Image Classification, Kidney Cancer, Renal Cell Carcinoma (RCC), MRI, CT Scan, CNN, ANN, SVM,** and **OpenCV.** To sharpen the search outcomes and boost the significance of the retrieved studies, these keywords were deployed in diverse combinations with the assistance of Boolean operators like **AND** and **OR**. By leveraging this tactical approach, the search was customized to generate a sturdy compilation of research articles, laying the groundwork for an extensive analysis of the subject matter.

To identify potentially relevant studies, the search results were first vetted by analyzing the titles and abstracts. After a thorough evaluation and elimination of irrelevant or duplicate materials, the remaining studies provided a comprehensive overview of the current state of research in the field, ensuring a solid foundation for this discussion. In total, approximately 30 publications were found, which proved helpful in understanding the machine learning models and identifying the problems in those models, ultimately contributing to the completion of the research objectives.

2.3 Thematic Classification

This review presents a multifaceted and interconnected examination of the existing literature, arranged into five primary themes, which encapsulate the fundamental aspects of the research topic. The themes are described below:

2.3.1 Medical Imaging Techniques for Cancer Detection

Medical imaging methods[29], such as Magnetic Resonance Imaging (MRI)[13] and Computed Tomography (CT) scans[14], are integral in detecting and diagnosing various forms of cancer, including renal malignancies[40]. These imaging techniques offer non-invasive, detailed visual representations of internal bodily structures, which are crucial for pinpointing and assessing cancerous growths[40].

Magnetic Resonance Imaging (MRI) represents a non-invasive imaging approach that combines a potent magnetic field, radio frequency pulses, and computer processing to produce intricate images of the body's inner structures[13]. MRI is especially beneficial for soft tissue imaging, offering high-resolution visuals of the kidneys and adjacent structures[13]. However, MRI can be lengthy, expensive, and contraindicated for patients with specific implants or medical conditions.

Computed Tomography (CT) scans employ a fusion of X-rays and computer algorithms to generate sectional images of the body. CT scans present various benefits for kidney cancer detection, such as swift acquisition times, widespread accessibility, and the capability to visualize calcification and vascular involvement[14]. Nevertheless, CT scans expose patients to ionizing radiation, which can be risky, particularly for younger patients or those requiring multiple scans[14].

Medical professionals adhere to precise guidelines and best practices for interpreting MRI[13] and CT scans[14] to guarantee accurate diagnosis and staging of renal cancer[3]. These guidelines take into account factors like lesion size, shape, enhancement patterns, and the existence of additional findings to differentiate benign from malignant growths and establish the disease's extent[14]. Cutting-edge developments in imaging methods, such as multi-parametric MRI and dual-energy CT, have further enhanced the diagnostic precision of these techniques in detecting kidney cancer[13].

Each imaging modality presents distinct benefits and drawbacks, necessitating a thorough evaluation of patient-specific factors and clinical context when choosing the most suitable method. By adhering to well-established guidelines and best practices, healthcare professionals can maximize the diagnostic accuracy of MRI[13] and CT scans[14], ultimately enhancing patient outcomes and alleviating the impact of renal cancer[3].

2.3.2 Kidney Cancer: Prevalence, Diagnosis, and Challenges

Kidney cancer poses a significant global health issue, with increasing incidence rates. In this section, the epidemiology, clinical aspects, and challenges surrounding kidney cancer are examined, touching on its prevalence, risk factors, subtypes, and diagnostic procedures[16][41]. The difficulties in early detection and accurate diagnosis, especially in medical image interpretation, are also highlighted[41].

- **Kidney Cancer Epidemiology and Prevalence:** Ranked among the top fifteen most common cancers, kidney cancer accounts for around 2.4% of new cases of cancer[16]. With over 400,000 cases and 180,000 deaths in 2020, the incidence varies globally, with North America and Europe experiencing the highest rates[16]. These disparities necessitate further research to understand and address the unique challenges faced by different populations[41].
- **Kidney Cancer Risk Factors:** Several risk factors have been identified for kidney cancer, including age, gender, race, obesity, smoking, hypertension, and exposure to certain chemicals or substances[41]. Men are at higher risk, and kidney cancer incidence rises with age[16].
- **Kidney Cancer Subtypes:** Kidney cancer can be classified into several histological subtypes, with the most common being Renal Cell Carcinoma (RCC)[5], accounting for approximately 90% of kidney cancer cases[41]. Within RCC, clear cell RCC is the most prevalent subtype, followed by papillary and chromophobe RCC. A comprehensive understanding of these subtypes is essential for the development of targeted therapies and personalized treatment plans for kidney cancer patients[16].
- **Kidney Cancer Diagnosis:** The process of diagnosing kidney cancer typically starts with a comprehensive review of the patient's medical history and a physical examination[3]. This helps the healthcare provider to identify any risk factors or symptoms that may be associated with kidney cancer[41]. Following the initial assessment, imaging studies, such as Computed Tomography (CT) scans[14], or Magnetic Resonance Imaging (MRI)[13], are conducted to detect renal masses and differentiate between benign and malignant lesions.
- **Challenges in Early Detection and Accurate Diagnosis:** Despite progress in diagnostic tools and imaging methods[29], numerous obstacles persist in the quest for early and accurate kidney cancer diagnosis[3]. These hurdles encompass:

- **Distinguishing Benign from Malignant Kidney Lesions:** The process of distinguishing between benign kidney lesions, such as simple cysts or angiomyolipomas, and malignant kidney tumors can be a complex task, particularly when dealing with small renal masses[41][16]. Accurate differentiation between these lesions is crucial, as it helps healthcare professionals determine the most appropriate treatment strategies for patients.
- **Variability of Tumor Subtypes:** Kidney cancer embodies diverse histological and molecular subtypes, each with unique traits that can add complexity to the diagnostic procedure, especially when relying solely on imaging methods[41][16]. Pinpointing the specific tumor subtype is critical, as it influences prognosis and treatment options, significantly impacting patient care[3].
- **Limitations of Imaging Modalities:** While Magnetic Resonance Imaging (MRI)[13] and Computed Tomography (CT)[14] scans are instrumental in detecting kidney cancer, they possess inherent limitations[41]. Challenges such as image misinterpretation, exposure to ionizing radiation (in the case of CT scans), and patient contraindications can impede their efficacy[41].
- **Accessibility and Resource Limitations:** The accessibility and caliber of diagnostic imaging services can differ substantially between regions and healthcare environments, potentially affecting the prompt and precise diagnosis of kidney cancer[5][41][16]. Addressing these disparities requires a united effort to enhance access to top-quality imaging services, particularly in resource-limited settings. This approach will help guarantee that all patients receive the best possible care, irrespective of their geographic location or socioeconomic status.

In conclusion, early detection and precise diagnosis of kidney cancer remain challenging due to the intricate interplay of epidemiological, clinical, and technological aspects[5][41][16]. By acknowledging and tackling these challenges, healthcare professionals and researchers can strive for more efficient diagnostic strategies, ultimately enhancing patient outcomes and reducing kidney cancer's overall burden[3].

2.3.3 Machine Learning Approaches in Medical Image Processing

Machine learning techniques[8] have demonstrated remarkable potential in medical image analysis[29], particularly in cancer detection through MRI[13] and CT scans[14]. In this section, machine learning techniques and algorithms will be explored[8]. There are two primary techniques in machine learning:

- **Supervised Learning:** Supervised learning is a form of machine learning where the algorithm learns from labeled data[42]. In this approach, input data is tagged with the correct output or target value, allowing the algorithm to learn how to predict the output based on the input data[42].
- **Unsupervised Learning:** Unsupervised learning is a form of machine learning where the algorithm learns from unlabeled data[42]. In this approach, input data is not tagged with any target value, enabling the algorithm to learn how to identify patterns or structures within the data[42].

With a deeper understanding of supervised and unsupervised learning[42], it is now possible to explore specific machine learning models[8]. Generally, ANN[33], SVM[34], and CNN[32] can be used for comparative analysis after training the dataset. Although ANN, CNN, and SVM are state-of-the-art methods capable of achieving high results, they are comparatively older approaches. This prompted the exploration of newer, innovative models, such as transformers[43]. As a result, an additional method will be employed to further strengthen the investigation using an advanced approach.

- **Artificial Neural Network (ANN)**: ANN[33] are computational models that mimic the structure and function of biological neural networks, like the human brain[44]. They process and analyze complex data, learn from it, and make predictions or decisions based on identified patterns[44]. In medical imaging, ANNs play a crucial role in image analysis, segmentation, and classification[33]. The architecture of an ANN is shown below in **Fig: 2.1**:

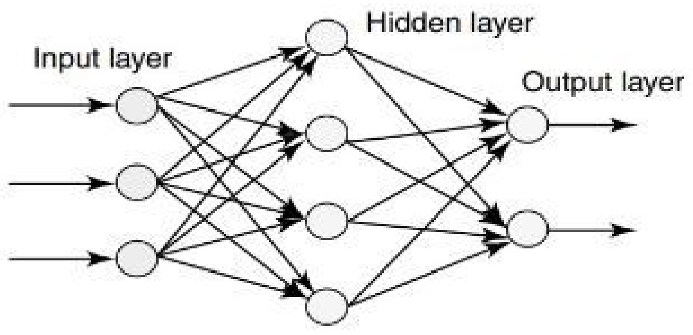


Figure 2.1: ANN Architecture[44]

- **Support Vector Machine (SVM)**: SVM[34] are a type of supervised learning algorithm used for classification or regression tasks. The goal of an SVM is to find the optimal decision boundary, or hyperplane, that separates different classes in the input space with maximum margin[45]. In medical image analysis, SVMs have been applied for tasks like image classification and lesion detection[18]. The **Fig: 2.2** below illustrates an example of the SVM architecture:

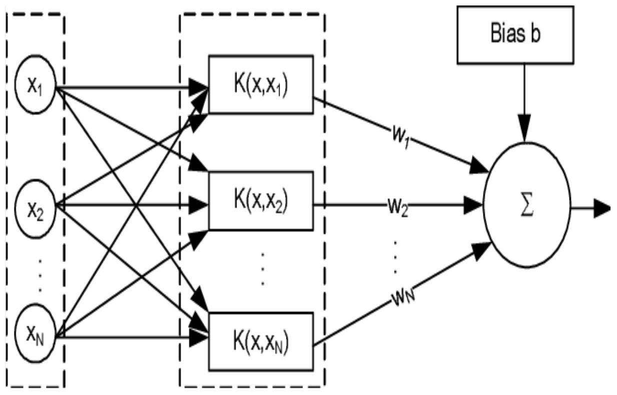


Figure 2.2: SVM Architecture[46]

- **Convolutional Neural Network (CNN)**: CNNs[32] are deep learning architectures tailored for processing grid-like data, such as images[47]. They draw inspiration from the human brain's visual processing mechanism and consist of multiple layers, including convolutional, pooling, and fully connected layers[47]. In medical image analysis, CNNs excel in tasks like image segmentation, classification, and object detection, such as identifying tumors or distinguishing between normal and abnormal tissues[19]. The **Fig: 2.3** below depicts an example of the CNN architecture:

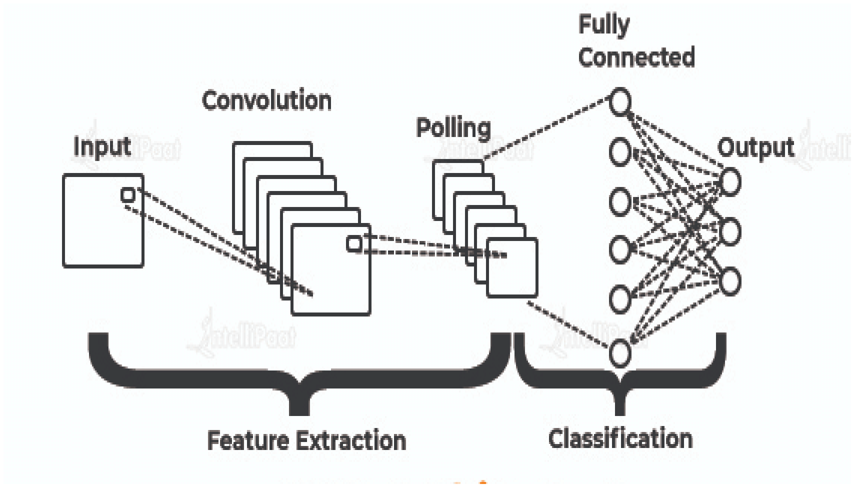


Figure 2.3: CNN Architecture[47]

- **Transformers**: Originally designed for natural language processing tasks, transformers[43] have gained attention in medical image analysis due to their self-attention mechanism and parallel data processing capabilities[48]. Researchers are exploring their applications, such as cancer detection in MRI[13] and CT scans[14][20]. The **Fig: 2.4** below shows the transformer's architecture:

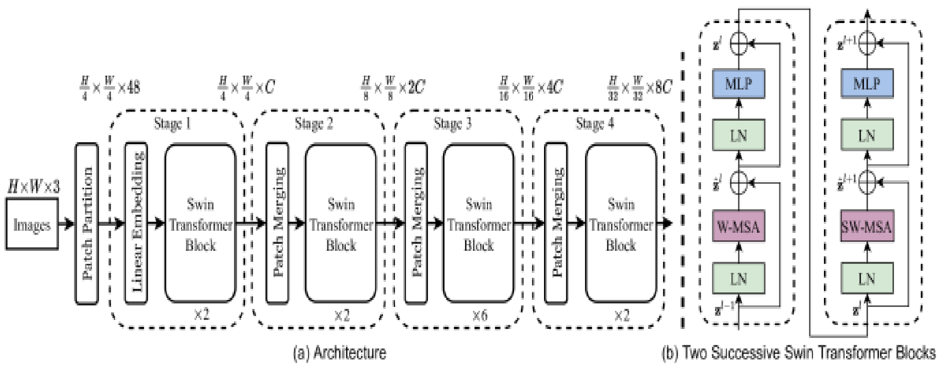


Figure 2.4: Transformers Architecture[48]

2.4 Analysis of Key Findings

2.4.1 Artificial Neural Network (ANN)

- **Key Findings**: ANNs have been successful in many fields, and they have shown moderate success in kidney cancer detection. However, their performance generally falls short compared to CNNs in tasks involving image classification[33].

- **Methodology**: In some studies, researchers used hand-crafted features extracted from MRI and CT scans as inputs to train an ANN for cancer classification[17]. Hand-crafted features are specific characteristics or patterns in the images that are manually selected by experts, rather than being automatically learned by the model.

- **Comparison**: ANNs are not as effective as CNNs at capturing spatial information in images. Spatial information refers to the relationships between neighboring pixels or regions in an image, which is crucial for understanding the structure and patterns in the data[19, 17]. Typically, the accuracy of ANNs varies, ranging from high results of around 90% to lower results close to 50% when training a model for medical images[49][50].

- **Advantages**: ANNs provide numerous benefits in medical imaging. They can model complex relationships between inputs and outputs, enabling diverse applications[33]. Additionally, they are adaptable, allowing updates with new data, making them well-suited for dynamic medical imaging environments. Overall, ANNs offer flexibility, adaptability, and parallelism in medical imaging, which are crucial for improved diagnosis and treatment[33].

- **Disadvantages**: ANNs in medical imaging have limitations, such as the need for manual feature extraction, which can be error-prone and time-consuming[44]. The training process is computationally intensive, and finding optimal network parameters is challenging. Addressing these challenges often requires expert knowledge and careful planning[17].

2.4.2 Support Vector Machine (SVM)

- **Key Findings**: SVMs can achieve reasonable accuracy in kidney cancer detection tasks when combined with effective feature extraction techniques[46]. This means that SVMs can perform well in classifying cancer presence when they are provided with relevant features derived from the medical images.

- **Methodology**: Researchers first extracted features from medical images, such as texture, shape, and intensity information. Then, they used these features as input for the SVM classifier, which was trained to perform binary classification based on the provided features[18].

- **Comparison**: In general, SVMs are not as effective as CNNs in image-based tasks because they rely on manual feature extraction, which can be time-consuming and may not capture all relevant information[19, 18]. However, SVMs are known to be very accurate and can achieve accuracy rates ranging from 85% to 95%[45][34]

- **Advantages**: SVMs offer multiple benefits compared to other machine learning models. They are robust to noise and outliers in training data due to their focus on maximizing the margin between classes[34]. The decision function depends only on a subset of training data, making the model compact and efficient. Additionally, SVMs can handle nonlinear relationships between inputs and outputs using kernel functions, which map input data into higher-dimensional spaces[34]. These advantages make SVMs popular in applications such as image and text classification, bioinformatics, and financial forecasting.

- **Disadvantages**: SVMs also have limitations, such as scalability, feature engineering, and interpretability. They can be computationally expensive to train, particularly with large datasets and high-dimensional input spaces. Similar to ANNs, SVMs often require manual feature extraction from input images, which can be time-consuming and error-prone[44].

2.4.3 Convolutional Neural Network (CNN)

- **Key Findings**: A variety of intriguing CNN architectures exist, with unique structures and varying degrees of performance[51]. Identifying the ideal architecture for a given task requires exploring each method and observing its performance with different datasets. In many cases, Visual Geometry Group (VGG16) demonstrate impressive results, although it can be time-consuming to train[51, 19].

- **Methodology**: Researchers trained a CNN using a dataset containing labeled MRI and CT scans. The images in the dataset were annotated as either cancerous or non-cancerous. During training, the CNN learned to recognize features and patterns that are indicative of kidney cancer[32].

- **Comparison**: CNNs generally outperform traditional machine learning models like Support Vector Machine (SVM) and Artificial Neural Network (ANN) in image classification tasks. The primary advantage of CNNs is their ability to learn complex patterns in images by automatically extracting relevant features, whereas traditional models rely on manual feature extraction. CNN's VGG16 can achieve remarkable accuracy which usually lies between 85% - 90%[51, 19].

- **Advantages**: CNNs offer notable benefits in image analysis. They can automatically learn features, removing the need for manual feature engineering and leading to efficient, accurate feature extraction. CNNs are robust to minor translations and rotations in input data, making them suitable for analyzing images with various orientations and positions.

- **Disadvantages**: CNNs also have drawbacks. They demand substantial computational resources and extensive labeled data for training. Deep CNNs may overfit, especially with limited training data. Regularization techniques can mitigate overfitting but add model complexity.

2.4.4 Transformers

- **Key Findings**: In comparison to CNN, the use of transformers in detecting kidney cancer from MRI and CT scans is still limited. Exploring and improving transformers for cancer detection in medical images could be a promising area of research[43]. Transformers also have a variety of architectures, like CNNs. The Swin Transformer was chosen due to its better accuracy and efficiency compared to previous models, such as Vision Transformers (ViT), which is considered one of the famous transformer models[48, 52].
- **Methodology**: Researchers adapted the transformer-based architectures, originally designed for sequence-to-sequence tasks, to handle image classification tasks such as cancer detection. This involved converting image data into a suitable format, like sequences of image patches or pixel embeddings, and feeding it into the transformer model for training and classification[48].
- **Comparison**: Transformers have demonstrated success in various domains, particularly in natural language processing, but their performance in cancer detection from medical images is not yet as well established as CNNs[19]. While transformers have potential in this area, more research is needed to understand their strengths and weaknesses compared to other ML models[20].
- **Advantages**: Transformers' self-attention mechanism enables them to model long-range dependencies and complex relationships, which may help identify intricate patterns in medical images that other models might miss, potentially enhancing cancer detection accuracy[48, 52].
- **Disadvantages**: Limited research on transformers in medical image analysis makes it challenging to assess their effectiveness compared to established models. Moreover, transformers can be computationally intensive, necessitating powerful hardware and possibly longer training times[48, 20].

2.4.5 Integration of ML Models with OpenCV

- **Key Findings**: The integration of Machine Learning (ML) models with OpenCV has demonstrated potential for improved performance in cancer detection using medical imaging[35]. Studies that have combined ML models, such as ANN, CNN, and SVM, with OpenCV have achieved enhanced feature extraction, real-time processing, and increased automation in cancer detection tasks[35, 12].

- **Methodology**: Researchers in these studies utilized OpenCV for various image pre-processing tasks, such as noise reduction, normalization, and segmentation, before applying ML models for cancer classification or detection[35]. The integration involved using OpenCV for feature extraction and ML models for learning patterns and making predictions.
- **Comparison**: The integration of ML models with OpenCV has shown improvements in performance compared to standalone ML models or traditional image processing techniques[35]. The combination of ML models' learning capabilities and OpenCV's efficient image processing functions allow for more accurate and faster cancer detection in medical imaging[12].
- **Advantages**: Integrating ML models with OpenCV offers several benefits in medical imaging, such as improved accuracy, faster processing, and greater adaptability. The synergy between ML models and OpenCV enables the development of more robust and efficient cancer detection systems that can adapt to varying imaging conditions and data types[12].
- **Disadvantages**: Despite the advantages, the integration of ML models with OpenCV also presents some challenges. Manual fine-tuning and optimization of the combined system can be difficult and time-consuming[12, 44]. Additionally, the lack of standardized evaluation metrics makes it challenging to consistently compare the performance of different integrated approaches[35].

In **Table 2.1**, a summary of the accuracy and efficiency of various machine learning models commonly used in cancer detection tasks, specifically in the context of medical image processing, is presented. The table compares the performance of models using the information obtained after a literature review.

Model	Accuracy	Efficiency
ANN	Average	Average
SVM	Good	Depends
CNN	High	Good
Transformers	Good	Below Average

Table 2.1: Accuracy of Machine Learning Models

2.5 Identifying Research Gaps and Inconsistencies

- **Lack of Integration of Models with OpenCV:** In recent times, there has been an increasing interest in using machine learning models for kidney cancer detection[5, 8]. Studies have used different models like CNN[32], ANN[33], SVM[34], and Transformers[43] for kidney cancer detection. However, a comprehensive comparison of these models is lacking in the literature, making it challenging to determine the most suitable model for the task. Additionally, integrating machine learning models with OpenCV, a valuable resource in various image processing tasks, for kidney cancer detection has not been extensively researched[3, 12].

  By conducting in-depth research and comparing different machine-learning models in the context of kidney cancer detection, researchers can gain valuable insights into their performance, accuracy, and suitability for the task[3, 8]. Such comparisons can inform the choice of model for this application

- **Low accuracy of Existing Models:** The accuracy of some existing machine learning models in detecting kidney cancer is low due to the complexity of the disease, which is characterized by the heterogeneous appearance and variable tumor locations[3, 8]. Although there are models that can identify other types of cancer from medical images such as breast cancer and lung cancer, their accuracy in kidney cancer detection is often unsatisfactory[53, 54]. The challenges in detecting kidney cancer arise from the diverse ways in which the disease presents itself, with tumors sometimes hidden or obscured by other anatomical structures or appearing in unusual locations. As a result, models that achieve high accuracy rates for other cancers may not perform as well in kidney cancer detection[3].

- **Insufficient Training Data or Unbalanced Datasets:** Insufficient training data or unbalanced datasets can result in less accurate and inherently biased machine learning models[30]. Some studies have faced issues due to insufficient or unbalanced datasets, negatively affecting their model performance [21, 55]. To address this issue, diverse and representative datasets must be collected, incorporating images of kidney cancer at various stages of development and from different imaging techniques, such as CT, and MRI scans[14, 13]. By exposing the model to a wide array of examples, a more comprehensive understanding of the underlying patterns and features associated with kidney cancer can be developed, which can significantly enhance the performance of the model.

- **Limited Generalizability to Different Cancer Stages:** Machine learning models for kidney cancer diagnosis may have limited generalizability to different types of medical images or cancer stages[5, 29]. Some models may have been developed and tested using specific imaging modalities, such as Computed Tomography (CT) or Magnetic Resonance Imaging (MRI)[13, 14], or for particular cancer stages, which can result in reduced effectiveness when applied to other contexts [56, 57]. Therefore, it is crucial to develop adaptable models that can accurately detect kidney cancer across all stages and a wide range of imaging techniques, including ultrasound, to improve the diagnostic process and patient outcomes.

2.6 Significance of the Thesis

This research holds great value in addressing several notable gaps within the domain of kidney cancer detection using medical images[21, 58]. Through the examination and comparison of various Machine Learning (ML) models[8], such as Convolutional Neural Network (CNN)[32], Artificial Neural Network (ANN)[33], Support Vector Machine (SVM)[34], and Transformers[43], this study aims to discover the most efficient and precise technique for diagnosing kidney cancer from Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) scans[13, 14]. This in-depth evaluation will not only yield valuable insights into the pros and cons of each model but also expand the general understanding of how machine learning can be applied in medical image processing.

The proposed approach has the potential to substantially improve the accuracy of kidney cancer detection in medical images[3]. By optimizing the selected ML model, the goal is to minimize diagnostic errors and contribute to more effective patient care and outcomes. A highly accurate model can also serve as a valuable aid to radiologists and healthcare professionals, enabling them to make better-informed decisions.

An essential aspect of this research is the integration of ML models with OpenCV, a popular open-source computer vision library[6, 12]. OpenCV offers a comprehensive array of tools for image processing and analysis that, when combined with cutting-edge ML methods, can lead to considerable enhancements in overall performance[15]. By harnessing the capabilities of OpenCV for tasks such as image preprocessing, feature extraction, and post-processing, this study seeks to establish a more accurate and effective cancer detection system.

2.7 Summary

In this literature review, the critical role of medical image processing techniques in the early and accurate detection of cancer, particularly kidney cancer, is emphasized[3]. The analysis focuses on the identification of cancer through Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) scans[14, 13]. It synthesizes knowledge on various machine learning frameworks, such as Convolutional Neural Network (CNN), Artificial Neural Network (ANN), Support Vector Machine (SVM), and Transformers, highlighting their application in medical image processing for cancer detection[32, 33, 34, 43]. Key gaps found in the literature include limited accuracy and applicability of certain models, insufficient training data, skewed datasets, and a lack of comprehensive comparison between Machine Learning (ML) models and their integration with OpenCV[21, 54].

The present research addresses these gaps by examining and contrasting diverse machine learning frameworks and assessing their ability to enhance kidney cancer detection accuracy[41, 8]. By integrating ML models with OpenCV, the study aims to improve the overall effectiveness of cancer identification[12]. The findings are expected to contribute to the field of medical image processing for cancer detection by providing valuable insights into the most accurate and efficient methodologies for kidney cancer diagnosis[29, 3].

It is important to consider the limitations of this literature review, as the rapidly evolving landscape of machine learning and medical image processing may introduce new models or methods not covered herein. Therefore, future investigations may reveal further insights into the challenges and opportunities in this field.

Chapter 3

Data Collection

3.1 Introduction

In contemporary times, where data is the driving force, the success of Machine Learning (ML)[6] and Artificial Intelligence (AI)[10] models largely depends on the quality[59] of the datasets used for their training and evaluation. As the adage *garbage in, garbage out* suggests that a model trained on low-quality data will produce inadequate or incorrect results. The accuracy of predictions holds paramount importance in critical fields like healthcare, and autonomous systems, as the consequences of inaccurate forecasts can be extremely consequential. Thus, it is imperative to allocate sufficient time and resources towards acquiring a dataset that exhibits high quality standards. The **Fig: 3.1** depicts the structure of this chapter through a block diagram.

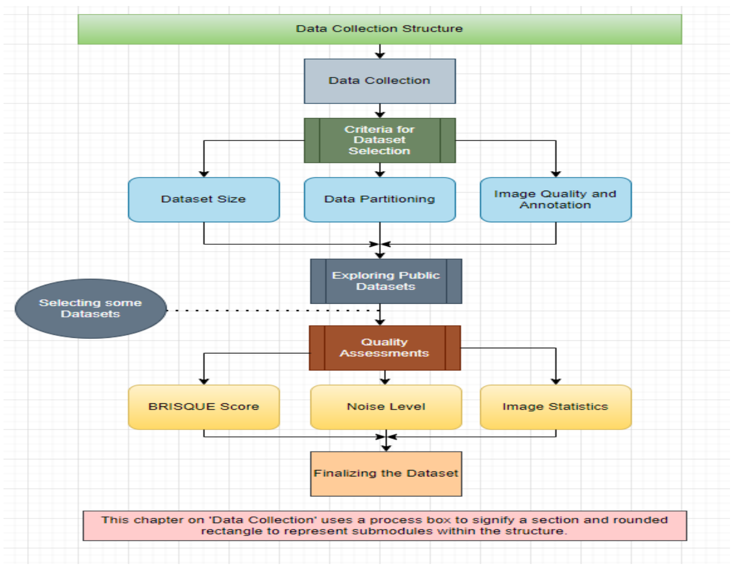

Data Collection Structure
Data Collection
Criteria for Dataset Selection
Dataset Size
Data Partitioning
Image Quality and Annotation
Exploring Public Datasets
Selecting some Datasets
Quality Assessments
BRISQUE Score
Noise Level
Image Statistics
Finalizing the Dataset
This chapter on 'Data Collection' uses a process box to signify a section and rounded rectangle to represent submodules within the structure.

Figure 3.1: Data Collection Structure

3.1.1 Technical Requirements

This section outlines the technologies and programming languages used throughout the thesis. For practical implementation in real-world scenarios, a thorough comprehension of the required hardware and software is necessary.

The hardware used for this thesis includes a laptop with the following specifications: 20N3S88305, Intel(R) Core(TM) i5-8365U CPU @ 1.60 GHz, 1896 Mhz, 4 Core(s) 8 Logical Processor(s) and 8 GB RAM. These specifications are adequate for running the software and libraries mentioned, as well as for training and testing the machine learning models.

Python has been chosen as the principal programming language for this dissertation. Python, a flexible, open-source, high-level programming language introduced in 1991, is renowned for its readability, simplicity, and extensive standard library. 3.11.3 is the specific version of Python being used. In addition, essential libraries such as **os, opencv-python, matplotlib**, and **numpy** have been incorporated. Jupyter Notebooks serve as the primary platform for the project, and the thesis contains numerous visual examples of implementation.

The TensorFlow library is utilized for training ANN and CNN. Google Brain's TensorFlow is an open-source machine learning framework that simplifies the implementation and deployment of a variety of machine learning models[60]. For SVM, the widely-used Python library scikit-learn for machine learning and data mining tasks is implemented[61]. Finally, the PyTorch library is used for transformers[62]. This library is an additional open-source machine learning library developed by Facebook's AI Research division. Following this discussion, the optimal dataset selection criteria will be considered.

3.2 Criteria for Optimal Dataset Selection

The optimal performance of a machine learning algorithm[8] is dependent on the availability of a high-quality dataset[63]. A set of criteria was established to assist the selection process for a dataset that best suits the needs of the specific assignment after extensive research and review. These criteria are influenced by the issue domain's particular problems and demands, as well as the features and constraints of the machine learning algorithms[8] and resources available. By carefully evaluating these variables, the goal was to find a dataset that would not only fulfill the requirements, but also contribute to the effective application of machine learning algorithms[63] and the development of dependable, accurate results. The following are the criteria used to choose a dataset:

3.2.1 Dataset Size

The size of the dataset is one of the most important factors that determines how well a machine learning model[8] works. A large collection provides the model with more examples to learn from, helping it find patterns and make accurate predictions. On the other hand, a small dataset can lead to over-fitting[64], which happens when the model depends too much on the training data[65] and can't adapt to new situations. To avoid overfitting, there must be a large enough set of images for training[65], testing[66], and validating[67] the model. Taking into account the technical limitations of the machine learning system, a dataset between 5,000 and 25,000 images was chosen.

3.2.2 Image Quality and Annotation

For an accurate diagnosis of cancer, detailed and high-quality medical images are necessary. To avoid overlooking subtle signs of cancerous cells, these images must be clear and of high quality[59]. In order for machine learning models[8] to learn and recognize the correct patterns, it is also necessary to annotate images with accurate ground-truth labels. Poorly annotated or low-quality images can hinder the performance of the model and even lead to erroneous conclusions[59].

3.2.3 Data Partitioning: Training, Testing, and Validation Sets

To obtain a fair evaluation of a model's performance in the field of machine learning[6], it is essential to divide data into training[65], testing[66], and validation sets[67]. An equal distribution of data across groups is essential for reliable model assessment. It is generally agreed that the ideal split should be 60% training, 20% testing, and 20% validation. The training set[65] is the most important part because it will be used to teach the model how to spot and understand particular trends in the data. The testing set[66] is used to evaluate the trained[28] model's ability to generalize to new data and produce predictions. To prevent over-fitting, which can lead to poor performance on new data, the model's parameters are fine-tuned using the validation set[67].

3.3 Exploring and Identifying Potential Datasets

When it comes to finding the right dataset, it's important to start the search by looking for publicly available datasets that are relevant to the specific topic. The search for datasets began by exploring various online resources such as:

- Google Datasets[68]
- Kaggle Datasets[69]
- The Cancer Imaging Archive (TCIA)[70]
- National Biomedical Imaging Archive (NBIA)[71]

Through these public databases mentioned in **List: 3.3**, several potential datasets were identified. After carefully evaluating the options, five datasets [1] [2] [3] [4] [5] were ultimately selected that met all of the criteria outlined in **Section: 3.2**. The next step is to critically assess these datasets, weighing their advantages and disadvantages to find the one that best fits the requirements. In the following sections, the evaluation procedure will be discussed in detail, looking at each dataset from multiple angles[72].

3.4 Datasets Quality Assessment

After conducting thorough research, it has become evident that selecting the most appropriate dataset is crucial to achieving optimal results in machine learning[6][59]. To make an informed decision, a variety of quality evaluation techniques[72], such as Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) score[73], can be employed to analyze and compare different datasets, ultimately choosing the one that best meets the needs. In this case, a random sample of 500 images from each dataset has been chosen for quality assessment[72], providing a representative snapshot without overburdening system resources. The evaluation process begins by setting up the paths to the datasets and reading the images.

3.4.1 Initialize paths and Read Images

First, the variables with paths of datasets are initialized, and later the existence of the file paths is verified using Python's **os** library. After setting the paths, the next step is to import all images from the datasets and then conduct Image Quality Assessment (IQA)[72]. Two libraries, **os** and **OpenCV**[12], are incorporated. os is used to list all the images, and OpenCV is used to read all of the images, which are later stored as arrays.

[1]`https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=52757270`
[2]`https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=11829555`
[3]`https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=16056331`
[4]`https://www.kaggle.com/datasets/junyussh/tcga-kirp`
[5]`https://www.kaggle.com/datasets/obulisainaren/multi-cancer`

By storing the images as arrays, quality assessments[72] can easily be implemented on the images. As we are only evaluating the quality of the datasets, we will refrain from making any modifications to preserve the integrity of the images. The code defined in **Code: 3.4.1** is for reading images from the dataset:

```
for filename in os.listdir(path):

    # Read the image in grayscale and add to dict
    image_path = os.path.join(path, filename)
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    images[filename] = image
```

Code 3.1: Code to read all the Images of Sample Datasets

3.4.2 Quality Assessment Techniques

Following extensive research, a selection of quality assessment techniques[72] has been identified as particularly effective for evaluating and comparing image datasets[74][75][76][77]. These methods include the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) score[73], noise detection in images using the laplacian filter[78], and image statistics, such as mean, median, and standard deviation[76][77]. By employing these techniques, a comprehensive understanding of each dataset's quality can be gained, and an informed decision regarding the most suitable dataset for specific needs can be made. The assessments will commence by applying the first technique, which is the BRISQUE score.

3.4.2.1 BRISQUE Score

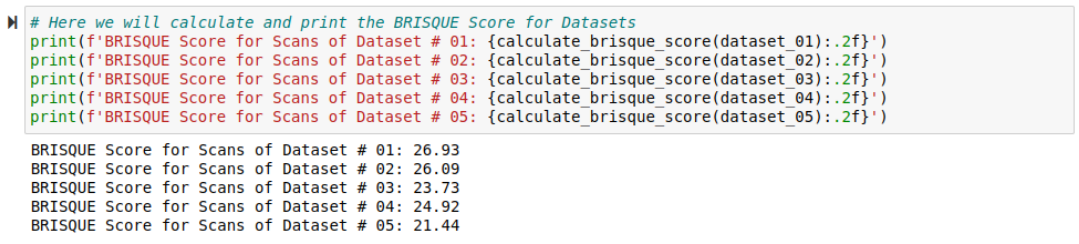
The BRISQUE[73] assesses the observed quality of a picture without the need for a reference or *perfect* image for comparison. Due to the fact that the algorithm does not require any extra data or comparisons to assess picture clarity, it is referred to as *blind* or *referenceless*. The BRISQUE algorithm analyzes a picture's spatially-domain statistical characteristics, collecting the Natural Scene Statistics (NSS) of the image. To find anomalies like blur or compression artifacts, it specifically assesses the local mean, local standard deviation, and spatial relationships between the two[74]. By quantifying these distortions, the algorithm creates a single scalar value that serves as an indicator of overall image quality. It is important to note that the BRISQUE score[73] is inversely related to image quality, meaning that higher scores indicate lower image quality.

To compute the BRISQUE score[74], the **brisque** class from the BRISQUE library will be imported and an instance of the BRISQUE() object will be created, which can be utilized to calculate the score. Additionally, **numpy** will be used to calculate the mean BRISQUE score[74] of the scans. The **Code: 3.4.2.1** and **Fig: 3.2** show the code and implementation of BRISQUE on the datasets:

```
brisque_evaluator = BRISQUE()

for scan_id, scan in scans.items():
    # Calculate BRISQUE Score and store it
    scan = cv2.cvtColor(scan, cv2.COLOR_GRAY2BGR)
    brisque_score = brisque_evaluator.score(scan)
    brisque_scores.append(brisque_score)
```

Code 3.2: Code to Calculate BRISQUE score for Images of Sample Datasets



```
# Here we will calculate and print the BRISQUE Score for Datasets
print(f'BRISQUE Score for Scans of Dataset # 01: {calculate_brisque_score(dataset_01):.2f}')
print(f'BRISQUE Score for Scans of Dataset # 02: {calculate_brisque_score(dataset_02):.2f}')
print(f'BRISQUE Score for Scans of Dataset # 03: {calculate_brisque_score(dataset_03):.2f}')
print(f'BRISQUE Score for Scans of Dataset # 04: {calculate_brisque_score(dataset_04):.2f}')
print(f'BRISQUE Score for Scans of Dataset # 05: {calculate_brisque_score(dataset_05):.2f}')
```

```
BRISQUE Score for Scans of Dataset # 01: 26.93
BRISQUE Score for Scans of Dataset # 02: 26.09
BRISQUE Score for Scans of Dataset # 03: 23.73
BRISQUE Score for Scans of Dataset # 04: 24.92
BRISQUE Score for Scans of Dataset # 05: 21.44
```

Figure 3.2: BRISQUE Score of all Sample Datasets

After analyzing the BRISQUE scores[73], as seen in **Fig: 3.2**, it has been found that *Datasets # 03* **(23.73)** and *Datasets # 05* **(21.44)** performed exceptionally well because their BRISQUE scores were considerably lower than those of the other datasets, which also indicates that these datasets have higher quality images[59] compared to the other datasets.

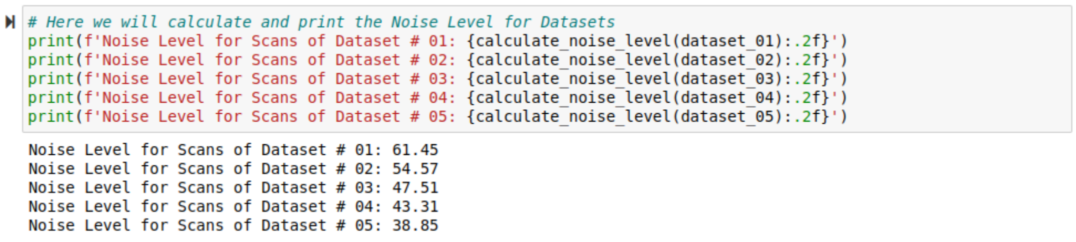
3.4.2.2 Noise Level

The second Image Quality Assessment (IQA)[72] technique employed is detecting noise[79] in the images. Laplacian noise detection[75] is a method for detecting noise in digital images that can be introduced into pictures during the capture or editing phases. It has a smooth, low-frequency texture that can hide features and decrease image clarity. The Laplacian operator[78] computes an image's second derivative, which emphasizes the image's borders and sharp changes. This is because image borders correspond to rapid changes in pixel intensity, which can be captured by the image's second derivative[78].

By applying the Laplacian operator[78] to the image and computing the Laplacian answer's variance, we can detect Laplacian noise in images. The **OpenCV**[12] library is used to apply the **Laplacian**[75] technique, and the **numpy** library is needed to get the mean and standard deviation[77]. The standard deviation of the Laplacian filter[78] is the measure of spread noise in the image. The **Code: 3.4.2.2** and **Fig: 3.3** demonstrate checking noise levels in the datasets:

```python
for scan_id, scan in scans.items():
    # Calculate noise level and then store it
    laplacian = cv2.Laplacian(scan, cv2.CV_64F)
    noise_level = np.std(laplacian)
    noise_levels.append(noise_level)
```

Code 3.3: Code to calculate the Noise Level in Images of Sample Datasets



```python
# Here we will calculate and print the Noise Level for Datasets
print(f'Noise Level for Scans of Dataset # 01: {calculate_noise_level(dataset_01):.2f}')
print(f'Noise Level for Scans of Dataset # 02: {calculate_noise_level(dataset_02):.2f}')
print(f'Noise Level for Scans of Dataset # 03: {calculate_noise_level(dataset_03):.2f}')
print(f'Noise Level for Scans of Dataset # 04: {calculate_noise_level(dataset_04):.2f}')
print(f'Noise Level for Scans of Dataset # 05: {calculate_noise_level(dataset_05):.2f}')
```

```
Noise Level for Scans of Dataset # 01: 61.45
Noise Level for Scans of Dataset # 02: 54.57
Noise Level for Scans of Dataset # 03: 47.51
Noise Level for Scans of Dataset # 04: 43.31
Noise Level for Scans of Dataset # 05: 38.85
```

Figure 3.3: Noise Level in Sample Datasets

By using the Laplacian method[78] to assess the images' noise levels, as seen in **Fig: 3.3**, it has been found that *Dataset # 04* **(43.31)** and *Dataset # 05* **(38.85)** have lower noise[79] compared to other datasets, which means images of these datasets are less distorted and are cleaner than the rest of the datasets.

3.4.2.3 Image Statistics

The mean, median, and standard deviation[76][77] of the images will be determined to gain insight into how the dataset tends to cluster and the degree of variation it contains[77]. This analysis will aid in quality control and data filtering operations.

The mean of an image is the average of its pixels, calculated by summing all pixel values and dividing by the total number of pixels[76]. For example, a high mean value may indicate that the scans are typically bright, while a low mean value may suggest that the scans are generally dark[77]. The median, in contrast, represents the middle value in the sorted data. Particularly when extreme values or anomalies are present in the data, the median value can provide a more reliable measure of the central trend[76]. The standard deviation measures the dispersion of the dataset by quantifying the distance between the data points and the mean[77]. A high standard deviation implies that the data points are widely spread, whereas a low standard deviation indicates that the data points are tightly clustered around the mean[77].

Numpy will be used exclusively to compute the *Mean*, *Median*, and *Standard Deviation*[76][77] for the scans in the dataset. The **Code: 3.4.2.3** and **Fig: 3.4** demonstrate image statistics of datasets:

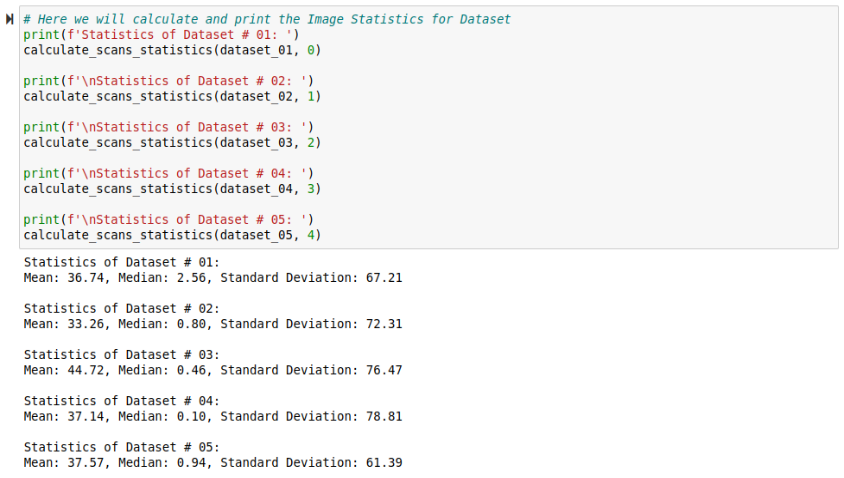
```python
for scan_id, scan in scans.items():
    # Get the Mean, Median, STD of the scans
    mean = np.mean(scan)
    median = np.median(scan)
    std = np.std(scan)

    # Then store the Mean, Median and STD of the scans
    means.append(mean)
    medians.append(median)
    stds.append(std)

# Then we will get the average of all values
avg_mean = np.mean(means)
avg_median = np.mean(medians)
avg_std = np.mean(stds)

print(f"Mean: {avg_mean:.2f}, Median: {avg_median:.2f}, Standard
   Deviation: {avg_std:.2f}")
```

Code 3.4: Code to Print Statistics of Sample Datasets



```python
# Here we will calculate and print the Image Statistics for Dataset
print(f'Statistics of Dataset # 01: ')
calculate_scans_statistics(dataset_01, 0)

print(f'\nStatistics of Dataset # 02: ')
calculate_scans_statistics(dataset_02, 1)

print(f'\nStatistics of Dataset # 03: ')
calculate_scans_statistics(dataset_03, 2)

print(f'\nStatistics of Dataset # 04: ')
calculate_scans_statistics(dataset_04, 3)

print(f'\nStatistics of Dataset # 05: ')
calculate_scans_statistics(dataset_05, 4)
```

```
Statistics of Dataset # 01:
Mean: 36.74, Median: 2.56, Standard Deviation: 67.21

Statistics of Dataset # 02:
Mean: 33.26, Median: 0.80, Standard Deviation: 72.31

Statistics of Dataset # 03:
Mean: 44.72, Median: 0.46, Standard Deviation: 76.47

Statistics of Dataset # 04:
Mean: 37.14, Median: 0.10, Standard Deviation: 78.81

Statistics of Dataset # 05:
Mean: 37.57, Median: 0.94, Standard Deviation: 61.39
```

Figure 3.4: Image Statistics of all Sample Datasets

The image statistics analysis results reveal general consistency across most datasets, as shown in **Fig: 3.4**. *Dataset # 05* appears to be the most appropriate choice. Exhibiting the lowest standard deviation **(61.39)**, it indicates fewer outlier images and a more consistent level of image brightness[77]. In addition, its mean value **(37.57)** is relatively moderate when compared to other datasets. Consequently, *Dataset # 05* emerges as the most consistent option among all the evaluated datasets.

3.4.2.4 Evaluation

In this section, Image Quality Assessment (IQA)[72] methods such as BRISQUE scores[73], noise levels[75], and image statistics[76][77] were employed to determine the most appropriate dataset. Due to its lower BRISQUE score, reduced noise, and overall higher image quality compared to the other datasets, *Dataset # 05* [5] emerged as the best option after a thorough evaluation. Following this selection, the dataset will be preprocessed[15] in preparation for the construction of a machine learning model[8]. Before moving on to that step, visualization of some of the images from the finalized dataset will be implemented.

3.4.3 Visualization

Visualization plays an important role in machine learning projects[8], helping to understand the data, guide model development, evaluate performance and interpret results[28]. It can help identify patterns or specific characteristics that the model is struggling with, and guide adjustments to the model architecture, training data[65], or hyper-parameters.

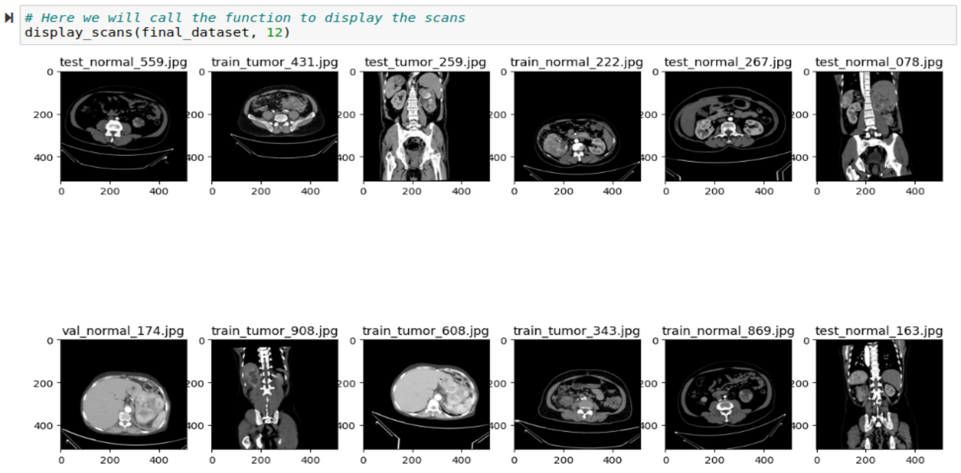
The **matplotlib** library will be used to display images. To add variety to the selection of images to display, the **random** library will be utilized to randomly select some images specified by the user. The final results and the code used to display the images are shown in **Fig: 3.5** and **Code: 3.4.3**:

```python
random_scans = random.sample(list(scans.items()), num_of_images)
plt.figure(figsize = (15, 15))

for i, (scan_id, scan) in enumerate(random_scans):
    plt.subplot(3, 6, i + 1)
    plt.imshow(scan, cmap='gray')
    plt.title(scan_id)
plt.show()
```

Code 3.5: Code to Visualize the Finalized Dataset

[5]https://www.kaggle.com/datasets/obulisainaren/multi-cancer



```
# Here we will call the function to display the scans
display_scans(final_dataset, 12)
```

Figure 3.5: Visualization of Finalized Dataset

3.5 Summary

In this module, different datasets were compared to find the best one for the machine learning project[8]. The BRISQUE[73] score was first used to assess the quality of images. Then, noise levels[75] in the datasets were examined using the laplacian noise detection[78] method. The images' mean, median, and standard deviation were also calculated to evaluate their quality and consistency[76][77]. After considering all these factors, *Dataset # 05*[5] was chosen as the best option because it had a lower BRISQUE score, less noise, and better overall image quality than the other datasets. Data visualization was then utilized to display the finalized dataset, as seen in **Fig: 3.5**.

In the next module, the focus will be on preprocessing and cleaning[15] the data to ensure the dataset is ready for the machine learning model[8]. This chapter will cover denoising the dataset[23], data augmentation[26], histogram equalization[25], and normalization[27]. The intention is to optimize the dataset for the subsequent phases of model development and training[28] by taking these steps.

[5]`https://www.kaggle.com/datasets/obulisainaren/multi-cancer`

Chapter 4

Data Cleaning and Preprocessing

4.1 Introduction

After finalizing the dataset, the exploration phase[80] can begin, which is essential for ensuring the precision and reliability of the results. The exploration procedure[80] consists of multiple critical phases, with preprocessing[15] taking precedence. A block diagram is utilized to display the structure of this chapter in the **Fig: 4.1**.

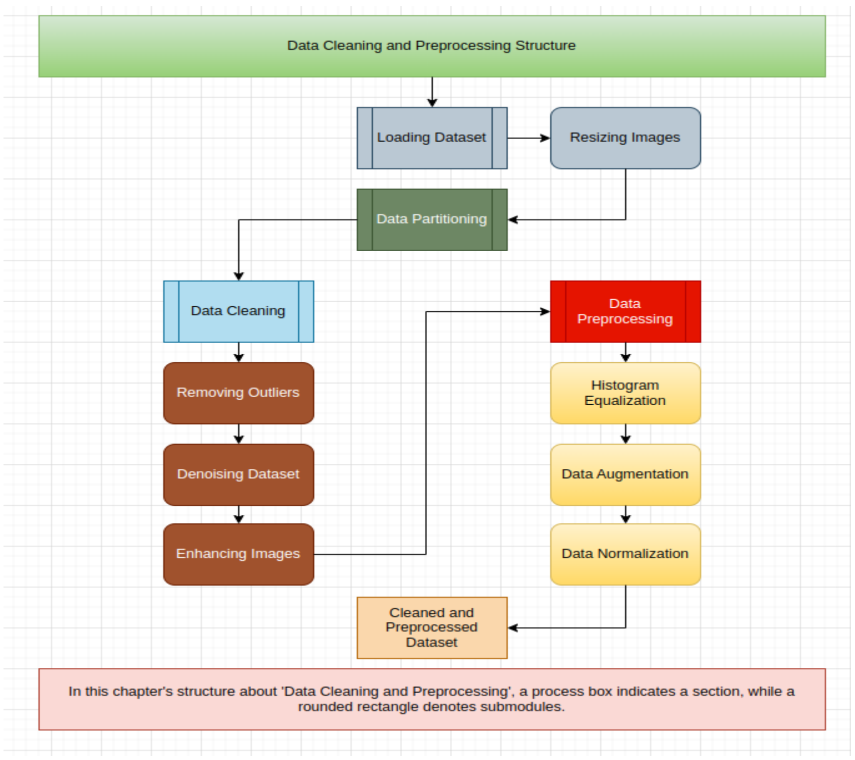

Data Cleaning and Preprocessing Structure
Loading Dataset
Resizing Images
Data Partitioning
Data Cleaning
Data Preprocessing
Removing Outliers
Histogram Equalization
Denoising Dataset
Data Augmentation
Enhancing Images
Data Normalization
Cleaned and Preprocessed Dataset
In this chapter's structure about 'Data Cleaning and Preprocessing', a process box indicates a section, while a rounded rectangle denotes submodules.

Figure 4.1: Data Cleaning and Preprocessing Structure

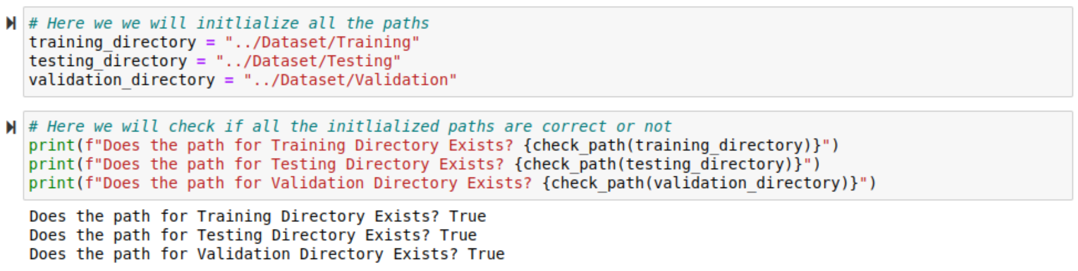
Preprocessing[15] is an integral part of any data analysis or machine learning endeavor[8], as it prepares the raw data for more efficient and effective processing. Preprocessing is significant due to its capacity to resolve data-related issues, such as noise[79], inconsistencies, missing values, and outliers[22]. It also improves the overall quality of the dataset, making it more appropriate for analysis and model training[28].

It is critical to load and inspect the dataset before going into data cleaning[81] and preprocessing[15]. This first stage allows one to become acquainted with the data and gain a better knowledge of its structure before moving on to the necessary preprocessing activities.

4.2 Load and Resize Dataset

4.2.1 Initialize Paths

The first task involves locating the dataset and reading all the images. To accomplish this, the paths must be initialized and validated using Python's **os** library. Below in the **Fig: 4.2** the paths are initialized and checked.



```
# Here we we will initlialize all the paths
training_directory = "../Dataset/Training"
testing_directory = "../Dataset/Testing"
validation_directory = "../Dataset/Validation"
```

```
# Here we will check if all the initlialized paths are correct or not
print(f"Does the path for Training Directory Exists? {check_path(training_directory)}")
print(f"Does the path for Testing Directory Exists? {check_path(testing_directory)}")
print(f"Does the path for Validation Directory Exists? {check_path(validation_directory)}")
```

```
Does the path for Training Directory Exists? True
Does the path for Testing Directory Exists? True
Does the path for Validation Directory Exists? True
```

Figure 4.2: Initializing the paths for our Dataset and checking them

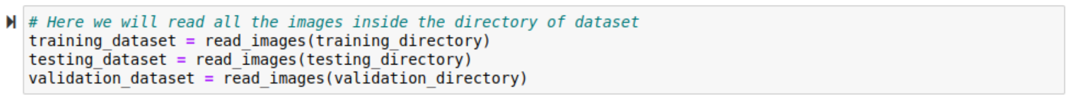
4.2.2 Load Images

After validating the location of the dataset as seen in **Fig: 4.2**, all the images can be loaded. To accomplish this, the code defined in **Code: 3.4.1** will be used. However, some modifications are required to accommodate the structure of the dataset, which consists of three directories: *Training*[65], *Testing*[66], and *Validation*[67]. In each of these directories, there are images divided as *Normal* and *Tumor* images.

4.2.2.1 Resize Images

To ensure consistency across the datasets, all datasets, including training[65], testing[66], and validation[67], will be resized. Once the images are read, they can be resized and then immediately stored in the dictionary. In addition, the images will be resized to **224x224**, which maintains a reasonable level of quality while drastically reducing the amount of computational capacity required for training[28] and preprocessing[15].

Code will be modified by adding stacked dictionaries to save pictures from the *Normal* and *Tumor* directories. The image below demonstrates how the dataset is loaded, which has already been separated into three distinct subsets: *Training*[65], *Testing*[66], and *Validation*[67].



```
# Here we will read all the images inside the directory of dataset
training_dataset = read_images(training_directory)
testing_dataset = read_images(testing_directory)
validation_dataset = read_images(validation_directory)
```

Figure 4.3: Reading all the Images of our Dataset

Now that all the images have been loaded and the dataset's structures are set up as seen in **Fig: 4.2** and **Fig: 4.3**, the essential phase of cleaning[81] and preprocessing[15] can begin. To ensure that the training, testing, and validation sets are divided appropriately, it is necessary to evaluate the data partitioning[82] before beginning the image preprocessing journey.

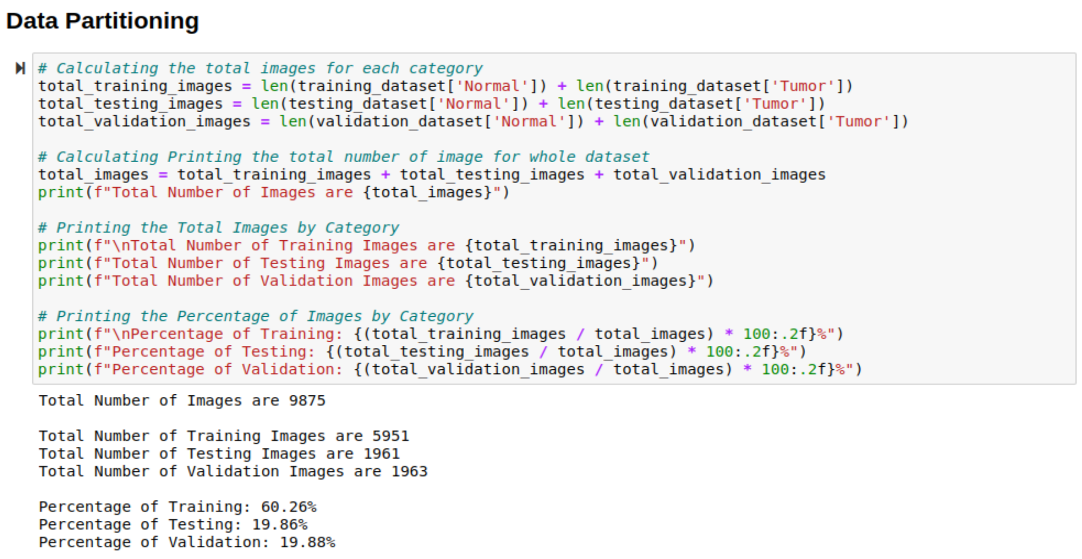
4.3 Data Partitioning

It is vital to ascertain that the dataset is proportionally partitioned[82] before delving into the preprocessing[15] part. Since the preprocessing operations will only be applied to the training dataset[65], it is first required to split the dataset and only clean[81] and preprocess the training dataset.

The ideal split ratio for a training[65], testing[66], and validation dataset[67] should be between 60:20:20 and 70:15:15. If the training set accounts for less than 60% of the data, the model may not have enough information to learn effectively. On the other hand, if the training set represents more than 70% of the data, the testing and validation sets could be too limited to provide an accurate assessment of the model's performance. If the dataset's current partitioning[82] deviates from these preferred ratios, it is advisable to adjust it to approximately 60:20:20 before moving forward with any additional preprocessing steps[15].

Although previously discussed, it is important to emphasize the significance of being vigilant during data cleaning[81] to ensure that the *Testing*[66] and *Validation*[67] datasets, which reflect real-life situations, stay unchanged. Modifying the images in these datasets could have a negative impact on the efficacy of the model[8].

As shown in **Fig: 4.4**, the data have already been partitioned[82] with an ideal ratio of approximately 60:20:20. Consequently, there is no need for further partitioning, and the process can proceed seamlessly with data cleaning[81] and preprocessing[15] operations.



Data Partitioning

```
# Calculating the total images for each category
total_training_images = len(training_dataset['Normal']) + len(training_dataset['Tumor'])
total_testing_images = len(testing_dataset['Normal']) + len(testing_dataset['Tumor'])
total_validation_images = len(validation_dataset['Normal']) + len(validation_dataset['Tumor'])

# Calculating Printing the total number of image for whole dataset
total_images = total_training_images + total_testing_images + total_validation_images
print(f"Total Number of Images are {total_images}")

# Printing the Total Images by Category
print(f"\nTotal Number of Training Images are {total_training_images}")
print(f"Total Number of Testing Images are {total_testing_images}")
print(f"Total Number of Validation Images are {total_validation_images}")

# Printing the Percentage of Images by Category
print(f"\nPercentage of Training: {(total_training_images / total_images) * 100:.2f}%")
print(f"Percentage of Testing: {(total_testing_images / total_images) * 100:.2f}%")
print(f"Percentage of Validation: {(total_validation_images / total_images) * 100:.2f}%")
```

```
Total Number of Images are 9875

Total Number of Training Images are 5951
Total Number of Testing Images are 1961
Total Number of Validation Images are 1963

Percentage of Training: 60.26%
Percentage of Testing: 19.86%
Percentage of Validation: 19.88%
```

Figure 4.4: Data Partitioning of our Dataset

Data cleaning[81] and data preprocessing[15] are two key components of getting a dataset ready for machine learning[6], and they will be the main topics of this module. Therefore, this chapter will be split into two separate sections so that each of these tasks can be addressed effectively. These are the activities that will be performed in each section:

- **Data Cleaning:**
  - Remove Outliers
  - Denoise Images
  - Enhance Images
- **Data Preprocessing:**
  - Histogram Equalization
  - Data Augmentation
  - Data Normalization

4.4 Data Cleaning

4.4.1 Importance of Data Cleaning

Cleaning images[81] is an essential step in preparing a dataset for machine learning[6] or computer vision applications. This process involves removing or correcting any errors or inconsistencies in the image data that can negatively impact the performance of the machine learning algorithm[8]. Ensuring a clean dataset not only improves the overall quality of the data but also aids in the accurate identification and extraction of meaningful patterns and relationships during the analysis and modeling stages.

4.4.2 Selecting and Applying Data Cleaning Techniques

In the pursuit of identifying suitable data cleaning techniques for handling diverse image datasets, thorough research was conducted to understand the range of methods available[81][83]. Given the varying nature of datasets and their unique properties, it became clear that no single technique would be a perfect fit for all situations. However, based on the research, a selection of data cleaning techniques[81] was narrowed down that have demonstrated effectiveness in addressing common image data quality issues[84].

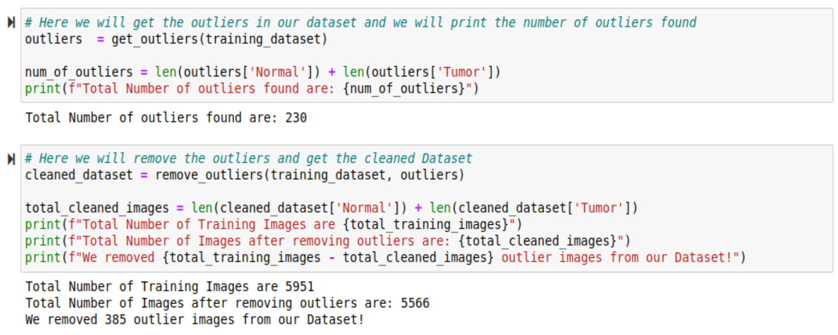
The data cleaning techniques[81] chosen include the z-score threshold method[85] to identify and remove outliers[22], denoising images to minimize noise and improve image clarity, and image enhancement techniques[24] to improve overall image quality[81][83]. By employing these methods, the goal is to create a clean and consistent dataset that allows for the successful application of machine learning algorithms[8] and the generation of reliable, accurate results.

4.4.2.1 Remove Outliers

Outliers are data points that significantly deviate from the norm and can lead to inaccurate results. Common causes of outliers[22] in images include noise[79], corruption, and errors in data entry. However, since a model with a low level of standard deviation was chosen, not many outliers are expected to be encountered. It is important to note that removing outliers from the dataset may impact the overall mean value[76], potentially leading to the identification of new outliers.

Z-score threshold[85] is a statistical metric used to find anomalies in a collection. It measures the deviation of a data point from the mean[76] in terms of standard deviations[77]. It is used to find data values that deviate considerably from the mean and may be classified as outliers[22].

The **numpy** library will be used to compute the *Mean* and *Standard Deviation* of the images. Based on these values, the z-score[85] for each image will be calculated. If the z-score of an image surpasses a predetermined threshold, which may be set by default, the image will be removed from the dataset. This method eliminates images with intensity values that considerably deviate from the average, possibly due to outliers or other anomalies[22]. The result and code for applying the z-score threshold to the dataset can be seen below in the **Fig: 4.5** and **Code: 4.4.2.1**.



```
# Here we will get the outliers in our dataset and we will print the number of outliers found
outliers  = get_outliers(training_dataset)

num_of_outliers = len(outliers['Normal']) + len(outliers['Tumor'])
print(f"Total Number of outliers found are: {num_of_outliers}")
```

```
Total Number of outliers found are: 230
```

```
# Here we will remove the outliers and get the cleaned Dataset
cleaned_dataset = remove_outliers(training_dataset, outliers)

total_cleaned_images = len(cleaned_dataset['Normal']) + len(cleaned_dataset['Tumor'])
print(f"Total Number of Training Images are {total_training_images}")
print(f"Total Number of Images after removing outliers are: {total_cleaned_images}")
print(f"We removed {total_training_images - total_cleaned_images} outlier images from our Dataset!")
```

```
Total Number of Training Images are 5951
Total Number of Images after removing outliers are: 5566
We removed 385 outlier images from our Dataset!
```

Figure 4.5: Removing Outliers from our Dataset

It may be observed that the function for *get_outliers()* has not been defined. This code is appropriately defined in **Code: A**

```
if outlier_scans is None:
    outlier_scans = get_outliers(dataset)

for category in ["Normal", "Tumor"]:
    for filename, image in dataset[category].items():
        # Store image which is not in outlier scans
        if filename not in outlier_scans[category]:
            cleaned_dataset[category][filename] = image

# Check the number of remaining outliers
rem_outliers = get_outliers(cleaned_dataset)
outliers_num = len(rem_outliers['Normal']) + len(rem_outliers['Tumor'])

# If num of outliers is <= 5, return the cleaned dataset
if outliers_num <= 5:
    return cleaned_dataset

# Call the function again with if there are more outliers
return remove_outliers(cleaned_dataset, rem_outliers)
```

Code 4.1: Function to remove all Outliers from the Dataset

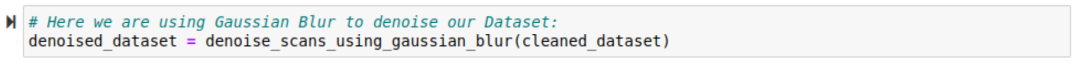
It is evident from **Fig: 4.5** that approximately *250* of the *6,000* images were initially detected as outliers. It is important to note that the removal of outliers[22] affected the overall mean value[76], potentially resulting in additional outliers. That's why an effort was made to eliminate as many outliers as possible, leaving only a handful. In total, around *400* outlier images were removed. With this step accomplished, the next step will be to denoise[23] the images.

4.4.2.2 Denoising Images

The next stage in the cleaning process[81] is to eliminate noise[79] from the images after removing the outliers[22] from the training dataset[65]. Denoising images is necessary because it improves the quality of input data[59], which in turn improves the performance of subsequent tasks, such as image classification, and object detection.

It is crucial to avoid denoising methods that excessively smooth images, leading to loss of vital information[86][87]. A literature review revealed Gaussian blur as a highly effective technique[88], but care must be taken to ensure it doesn't blur significant portions of images or cause information loss[87][89].

Gaussian blur denoising[88] is a commonly used image processing technique[15] that reduces noise[79] in digital images by applying a Gaussian filter[89]. This filter smooths the image by averaging pixel values with their surrounding neighbors using a matrix of values generated from the Gaussian function. Gaussian blur denoising is a simple and computationally efficient technique compared to others, but there is a trade-off between noise reduction and preserving image details. The *GaussianBlur()* function from the **OpenCV**[12] library was used to perform Gaussian blur denoising[88]. Below is the image **Fig: 4.6** and **Code: 4.4.2.2** for implementing Gaussian blur denoising on the dataset.



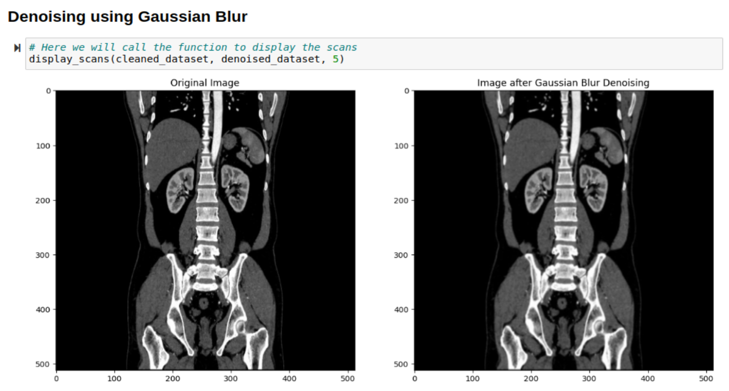
```
# Here we are using Gaussian Blur to denoise our Dataset:
denoised_dataset = denoise_scans_using_gaussian_blur(cleaned_dataset)
```

Figure 4.6: Noise filtering our dataset using Gaussian Blur

```
for category in ["Normal", "Tumor"]:
    for filename, image in dataset[category].items():
        # Apply Gaussian blur denoising and store image
        denoised = cv2.GaussianBlur(image, ksize=ksize, sigmaX=sigmaX)
        denoised_dataset[category][filename] = denoised
```

Code 4.2: Function to denoise our dataset using Gaussian Blur

In **Fig: 4.6**, Gaussian blur denoising[88] was applied using the code defined in **Code: 4.4.2.2** on the dataset, but it is necessary to make sure that it has not removed any information or smoothed the images too much. Therefore, the images were visualized. **Fig: 4.7** presents a visual comparison of the original images and their denoised versions[89]. The **Matplotlib** library was used, and the code defined in **Code: 3.4.3** was manipulated to display both scans side by side, making it easier to compare.



Denoising using Gaussian Blur

```
# Here we will call the function to display the scans
display_scans(cleaned_dataset, denoised_dataset, 5)
```

Figure 4.7: Visualization of Denoised Images using Gaussian Blur

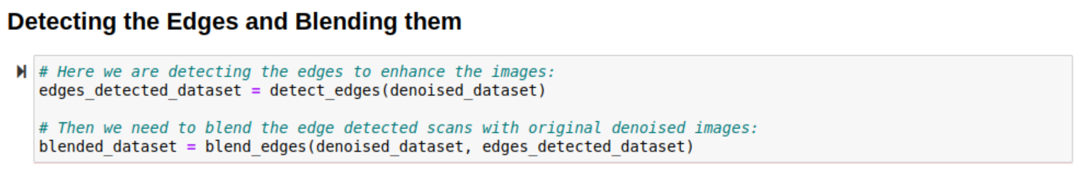
In **Fig: 4.7**, it can be observed that Gaussian blur[88] is quite effective in reducing noise[79] while maintaining image quality. Therefore, it can be concluded that Gaussian blur does not have any side effects while reducing noise in the images. This leads to improved image quality, ultimately benefiting patients and resulting in fewer errors in their examinations.

4.4.2.3 Enhance Images

After denoising the dataset[23] and removing outliers[22], the next crucial step is to enhance the quality of the images. This is essential because revealing fine details in the images can greatly facilitate the training of the model[28]. To select the most appropriate image enhancement method, extensive literature review and research on various techniques[90][91][92] were conducted. While many methods showed potential, edge detection and blending[93][94] emerged as the most suitable for the dataset due to their effectiveness in revealing subtle details and improving the overall quality of medical images[91].

By enhancing the boundaries between different anatomical structures and regions of interest, edge detection enables medical professionals to identify and differentiate various tissues, organs, or pathological features more easily[95][94]. Blending can be used to combine images, resulting in a composite image with complementary information from different modalities.

A combination of **OpenCV**, **TensorFlow**, and **NumPy** libraries was utilized to implement edge detection and blending[95][91]. OpenCV's *Canny* function was employed for edge detection, TensorFlow's *adjust_ contrast* function was used to adjust the contrast of the images, and *numpy* was utilized for numerical computing. Below is the image, **Fig: 4.8**, in which edge detection is implemented and edges are subsequently blended onto the dataset. To check the code, refer to **Code: A** for the *detect_ edges* and *blend_ edges* functions.



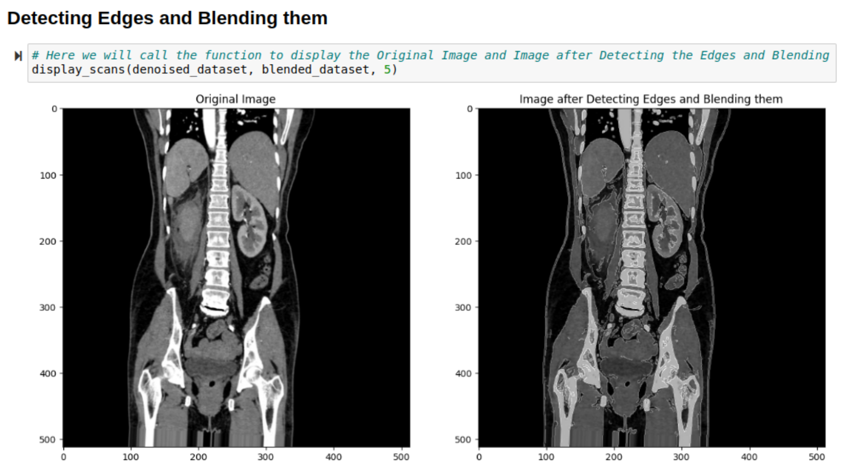
Detecting the Edges and Blending them

```
# Here we are detecting the edges to enhance the images:
edges_detected_dataset = detect_edges(denoised_dataset)

# Then we need to blend the edge detected scans with original denoised images:
blended_dataset = blend_edges(denoised_dataset, edges_detected_dataset)
```

Figure 4.8: Detected Edges and Blended on our Dataset

As depicted in **Figure 4.8**, edge detection and blending techniques[95][94] have been implemented on the dataset. However, it is crucial to conduct a thorough analysis to determine the effectiveness of these methods for the dataset. **Fig: 4.9** presents a visual comparison of the original images and their corresponding edge-detected and blended versions.



Detecting Edges and Blending them

```
# Here we will call the function to display the Original Image and Image after Detecting the Edges and Blending
display_scans(denoised_dataset, blended_dataset, 5)
```

Figure 4.9: Analysis of Original Image and Image after Detecting and Blending Edges

Upon examining the images, it can be observed that the edge detection and blending techniques[93][94] have effectively enhanced the boundaries between different anatomical structures and regions of interest. This enhancement makes it easier to differentiate between various tissues, organs, and pathological features, ultimately improving the quality of the images for subsequent analysis and machine learning tasks[95][94].

This marks the conclusion of this subsection, which addressed outlier removal[22], denoising of images[88], and implementation of edge detection and blending[95] to enhance the medical images in the dataset. In the upcoming section, the focus will be on data preprocessing, particularly histogram equalization[25], data augmentation[26], and data normalization[27], to further prepare the dataset for machine learning tasks.

4.5 Data Preprocessing

4.5.1 Importance of Data Preprocessing

This is the second part of the data cleaning process[81]. The initial segment focused on eliminating outliers[22], filtering images to reduce noise, and then detecting and merging edges. In this section, the focus is on data preprocessing. Data preprocessing is an essential step in the machine learning[6] and data analysis pipeline, as it prepares unprocessed data for more efficient and precise processing by machine learning algorithms[8]. The primary objective of data preprocessing is to transform the data into a more suitable format for analysis and modeling, ensuring that the underlying patterns and relationships can be identified and exploited accurately[83].

4.5.2 Selecting and Applying Preprocessing Techniques

In the quest to find effective preprocessing methods[15] for handling various datasets, extensive research was conducted, and three key techniques—Histogram Equalization[25], Data Augmentation[26], and Data Normalization[27]—stood out as particularly valuable and widely applicable in data preprocessing[96][97][98]. These methods enhance data quality and usability across various domains and improve the performance of subsequent machine learning models[8].

4.5.2.1 Histogram Equalization

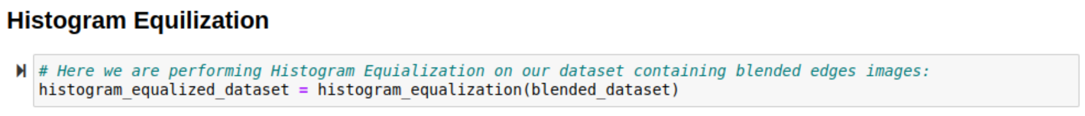
Histogram Equalization is a method used in image processing and computer vision to improve visual contrast by dispersing the intensity values of pixels [99]. Intensity values refer to the brightness or darkness of a pixel. Histogram equalization can be used for a variety of purposes, such as enhancing picture contrast, making various features, objects, or textures easier to detect and differentiate, preprocessing for machine learning, and improving an image's overall visual quality [25].

Histogram equalization can improve the performance of machine learning models[8] by providing cleaner and more representative data for training [25]. Additionally, histogram equalization can aid in data cleaning, which involves identifying and correcting errors or inconsistencies in data. The **OpenCV**[12] library will be used to perform Histogram equalization[25]. Here is the code and implementation of histogram equalization on the dataset:

```python
for category in ["Normal", "Tumor"]:
    for filename, image in dataset[category].items():
        # Convert the image and perform Histogram Equalization
        image_8bit = cv2.normalize(image, None, 0, 255, cv2.NORM_MINMAX,
            cv2.CV_8U)
        hq_image = cv2.equalizeHist(image_8bit)

        # Convert the equalized image to the original type and store it
        hq_image = hq_image.astype(image.dtype)
        hq_dataset[category][filename] = hq_image
```

Code 4.3: Function to apply Histogram Equalization on our Dataset



Histogram Equilization

```python
# Here we are performing Histogram Equialization on our dataset containing blended edges images:
histogram_equalized_dataset = histogram_equalization(blended_dataset)
```

Figure 4.10: Histogram Equalization Applied to Our Dataset

In **Fig: 4.10**, Histogram equalization[25] has been applied to the dataset using the code defined in **Code: 4.5.2.1**. The next step is to augment[26] the dataset.

4.5.2.2 Data Augmentation

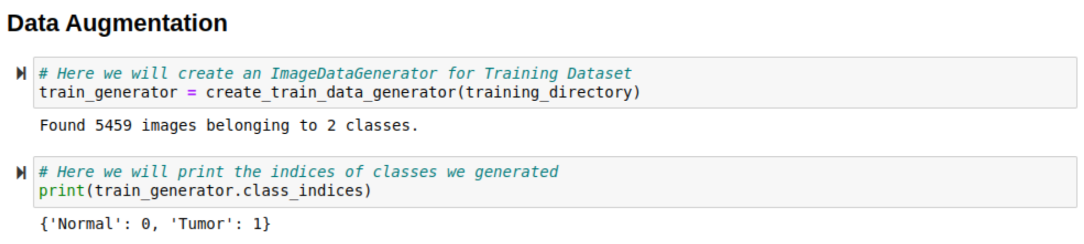
Data augmentation is a popular approach in machine learning[6] and computer vision that includes generating new training examples by transforming the existing dataset[26]. The main purpose is to broaden the training set by adding new variants, which helps the model generalize more effectively and decreases the risk of overfitting[64]. When dealing with restricted or unbalanced datasets, this strategy is very useful since it effectively improves the amount and variety of training samples. Data augmentation is a crucial technique for enhancing the efficacy of machine learning models[8], particularly in medical imaging applications such as cancer detection. Data augmentation is essential to the development of accurate and dependable cancer detection models, as it serves to surmount the obstacles posed by limited and imbalanced medical image datasets[26].

First, store the histogram-equalized images[25], as they will be utilized with data generators for data augmentation[26]. Then, import **ImageDataGenerator** from *tensorflow.keras.preprocessing.image* and create generators and then apply data augmentation[26] to the training dataset[65]. A batch size of **64** is chosen for the ImageDataGenerator and a target size of **224x224** pixels is implemented, which is a widely accepted standard. Various augmentation[26] techniques are employed on the training dataset[65], such as rotation, horizontal and vertical shifts, zooming, horizontal or vertical flipping, and nearest-neighbor filling when data is missing. Below is the code and implementation of data augmentation on the dataset:

```python
# To create ImageDataGenerator with Augmentation and Normalization
train_datagen = ImageDataGenerator(
    rescale = 1.0 / 255.0,      # To Normalize our Dataset
    rotation_range = 10,        # To Rotate the Image
    width_shift_range = 0.1,    # To Shift the Image Horizontally
    height_shift_range = 0.1,   # To Shift the Image Vertically
    zoom_range = 0.1,           # To Zoom in or out
    horizontal_flip = True,     # To Horizontally flip
    vertical_flip = True,       # To Vertically flip
    fill_mode = 'nearest'       # To fill missing Pixels
)

# Then create Generator for Training dataset
train_generator = train_datagen.flow_from_directory(
    training_directory,
    target_size = target_size,
    batch_size = batch_size,
    color_mode = 'grayscale',
    class_mode = 'binary'
)
```

Code 4.4: Function to Augment and Normalize Training Dataset


Data Augmentation

```python
# Here we will create an ImageDataGenerator for Training Dataset
train_generator = create_train_data_generator(training_directory)
```

```
Found 5459 images belonging to 2 classes.
```

```python
# Here we will print the indices of classes we generated
print(train_generator.class_indices)
```

```
{'Normal': 0, 'Tumor': 1}
```

Figure 4.11: Data Augmentation Applied to Our Dataset

Data Augmentation[26] has been implemented on the dataset as shown in **Fig: 4.11** using the code defined in **Code: 4.5.2.2**, and the subsequent step involves normalization[27].

4.5.2.3 Data Normalization

Data normalization is a preprocessing technique[15] used to standardize and transform data into a consistent format, allowing for easier comparison and analysis[27]. It scales the features or variables of a dataset to a common range, typically [0, 1] or [-1, 1]. This process mitigates the impact of varying scales and units, enabling algorithms to converge more efficiently and produce more accurate results[27].

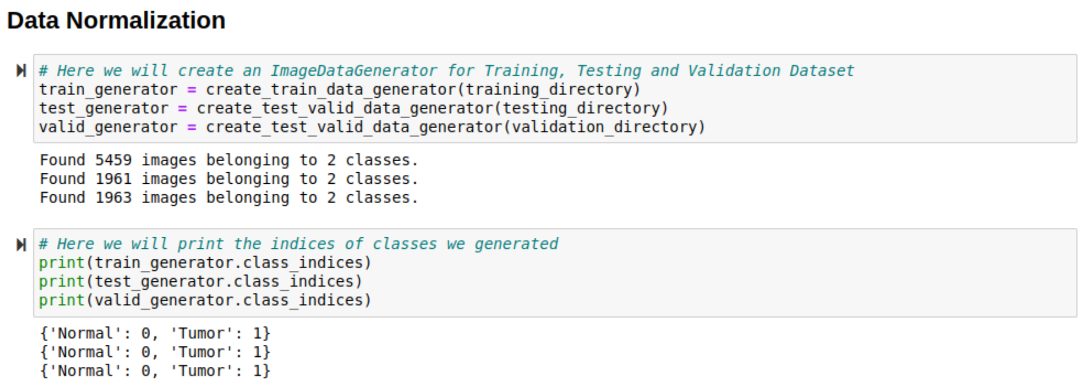
Data normalization is essential for medical imaging in cancer detection, as it ensures that variations in pixel intensities, illumination conditions, and other factors do not negatively impact the performance of machine learning models. It is important to note that data normalization[27] should be performed on the whole dataset, not just the training dataset[65]. The training dataset[65] was already normalized in **Section: 4.5.2.2** and now the test and validation datasets[27] will be normalized. First, import **ImageDataGenerator** from *tensorflow.keras.preprocessing.image* and create generators and apply normalization. Here is the code and implementation of data normalization on the train and validation dataset:

```python
# First, set the parameters for the Data Generator
batch_size = 64
target_size = (224, 224)

# To create ImageDataGenerator with Normalization
test_and_val_datagen = ImageDataGenerator(
    rescale = 1.0 / 255.0    # To Normalize
)

# Then create Generator for Testing or Validation Dataset
generator = test_and_val_datagen.flow_from_directory(
    directory,
    target_size = target_size,
    batch_size = batch_size,
    color_mode = 'grayscale',
    class_mode = 'binary'
)
```

Code 4.5: Function to Normalize Test and Validation Dataset



Data Normalization

```
# Here we will create an ImageDataGenerator for Training, Testing and Validation Dataset
train_generator = create_train_data_generator(training_directory)
test_generator = create_test_valid_data_generator(testing_directory)
valid_generator = create_test_valid_data_generator(validation_directory)
```

```
Found 5459 images belonging to 2 classes.
Found 1961 images belonging to 2 classes.
Found 1963 images belonging to 2 classes.
```

```
# Here we will print the indices of classes we generated
print(train_generator.class_indices)
print(test_generator.class_indices)
print(valid_generator.class_indices)
```

```
{'Normal': 0, 'Tumor': 1}
{'Normal': 0, 'Tumor': 1}
{'Normal': 0, 'Tumor': 1}
```

Figure 4.12: Data Normalization Applied to Our Dataset

Data normalization[27] has been implemented on the dataset, as shown in **Fig: 4.12** using the code defined in **Code: 4.5.2.3**, marking the end of this module. The pipeline of preprocessing data is now complete, and the next step is training the model.

4.6 Summary

In this section, a comprehensive preprocessing pipeline[15] was discussed to ensure that a medical imaging dataset is of high quality and suitable for training machine learning models[8] for cancer detection. The preprocessing operations applied to the dataset include denoising[23], outlier removal[22], image enhancement, data normalization[27], data augmentation[26], and histogram equalization[25].

Denoising techniques were used to eliminate noise from the images, while outlier removal helped to eliminate extreme values. Image enhancement techniques were applied to improve the visual quality of the images. Data normalization ensured that the dataset was transformed into a consistent format. Data augmentation increased the size and diversity of the dataset, enabling the models to generalize better and avoid overfitting. Histogram equalization was employed to improve the contrast of the images and ensure that the pixel intensities were distributed uniformly. By applying these preprocessing operations[15], the dataset is well-prepared for effective analysis and accurate cancer detection using machine learning algorithms[8]. With this preprocessing pipeline complete, the next stages of the project can be approached with confidence, including model selection, training[28], and evaluation.

Chapter 5

Model Training and Evaluation

5.1 Introduction

In the previous chapter, the comprehensive pipeline of Data Cleaning[81] and Preprocessing[83] was discussed. As a result, there is now a clean and preprocessed dataset ready for model training and evaluation. The **ImageDataGenerator** from the previous chapter will be used for training, testing, and validation purposes. Based on the literature review, as decided in **Section: 2.3.3**, the dataset will be trained using Artificial Neural Network (ANN)[33], Support Vector Machine (SVM)[34], Convolutional Neural Network (CNN)[32], and Transformers[48]. Before diving into the training process, it is crucial to understand the activation functions[100] and evaluation metrics[101] used during model creation and training.

5.2 Activation Functions and Evaluation Metrics

5.2.1 Activation Functions

In neural networks, an activation function is a mathematical operation applied to each neuron's output within a layer, influencing the network's final output[100]. Some prevalent types of activation functions include:

- **Sigmoid Function:** The sigmoid function maps any input value to a value ranging between 0 and 1[100]. It is frequently employed in binary classification tasks and was widely used in the past. The sigmoid function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5.1}$$

- **Rectified Linear Activation Unit (ReLU):** ReLU is a favored activation function that maps all negative input values to 0 while keeping positive input values unaltered[100]. It is computationally efficient and has proven effective in numerous neural network architectures. The ReLU function is defined as:

$$f(x) = \max(0, x) \tag{5.2}$$

- **Softmax Function:** The softmax function is typically used in multi-class classification problems to generate a probability distribution across the output classes[100]. It accepts an input vector and maps it to a vector of probabilities that sum to 1. The softmax function is defined as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \qquad j = 1, \ldots, K \tag{5.3}$$

After learning about activation functions[100], the discussion will be focused on evaluation metrics[101].

5.2.2 Evaluation Metrics

Evaluation metrics are used to assess the performance of the trained models[101]. In this section, various metrics used to evaluate the models are discussed.

- **Accuracy:** The accuracy of a model refers to its ability to correctly predict or classify the data it is trained on[101]. It is usually measured as the percentage of correct predictions or classifications made by the model on a test dataset.
- **Precision:** Precision is a metric used to evaluate the performance of a model in making accurate positive predictions[101]. It is the proportion of true positives (TP) among the total number of positive predictions (TP + false positives, FP).
- **Loss:** In machine learning, loss refers to the error or discrepancy between the predicted output of a model and the true output[101]. The goal of training a model is to minimize the loss function, which is a mathematical function that quantifies the difference between the predicted output and the true output.
- **F1 Score:** The F1 score is a commonly used metric to evaluate the performance of a model on a binary classification problem[101]. It is the harmonic mean of precision and recall, which are two important metrics used in evaluating a model's performance.

- **Area under the Curve (AUC):** The Area under the Curve (AUC) score is a commonly used metric for evaluating the performance of a binary classification model, particularly when the dataset is imbalanced[101]. The AUC score measures the model's ability to distinguish between positive and negative classes across all possible threshold values.

- **Confusion Matrix:** A confusion matrix is a table that is used to define the performance of a classification algorithm[101]. It presents the true class labels against the predicted class labels, allowing for an easy assessment of the classifier's accuracy and identification of misclassified instances.

After learning about evaluation metrics[101] and activation functions[100] now steps involved in training the dataset will be discussed:

5.3 Process of Model Development

5.3.1 Model Creation

The first step is to establish the model, either by constructing it from scratch or using transfer learning[102]. If transfer learning is employed, pre-trained models are imported and incorporated into the model. Activation functions, such as **ReLU** and **Sigmoid**, are used to introduce non-linearity and help the model learn complex patterns[103][102].

5.3.2 Model Compilation

The next step is to compile the model, setting specific evaluation metrics and choosing an optimizer for updating the model's weights to minimize the loss function[103][102][101]. Metrics were discussed in **Section: 5.2.2**. For more information, check that section. The learning rate is also set, determining the step size the optimizer takes to reach minimum loss[103].

5.3.3 Model Training

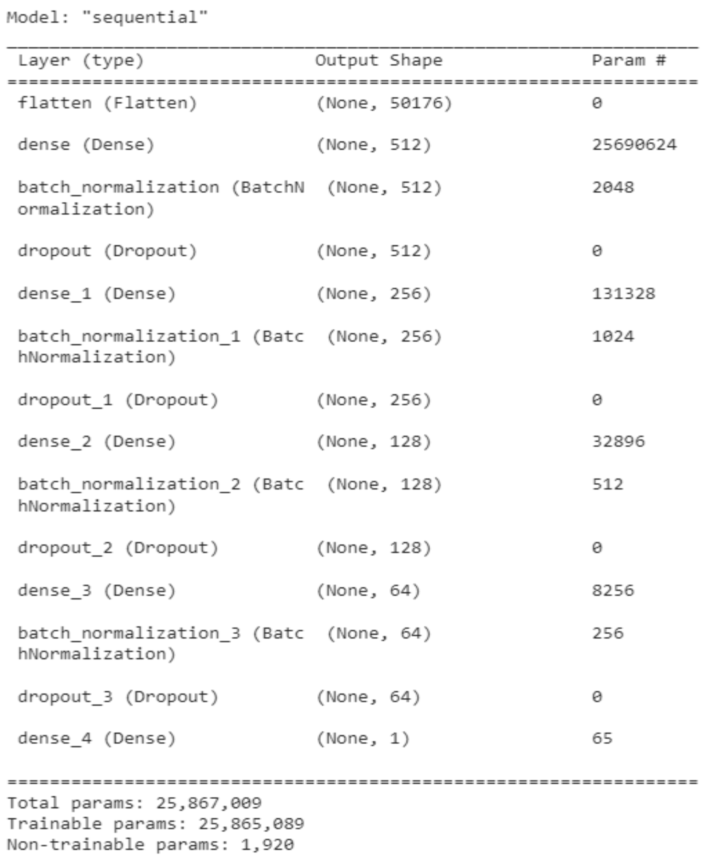
The final step is training the model using the training generator, validation generator, and epochs[102]. Early stopping is implemented to save the best weights of the model, which can be retrieved later for improved performance if the model starts overfitting after a certain number of epochs[102].

With a strong foundation in model training basics, the models will be trained using the ANN[33], SVM[34], CNN[32], and Transformers architectures[48]. Their performance will be evaluated using various evaluation metrics, including Accuracy, Precision, Recall, F1 Score, Area under the Curve (AUC), and Loss, to identify the most suitable model for the specific task, taking into account both accuracy and computational efficiency[102][101]. Additionally, a confusion matrix will be used to further strengthen the justifications.

5.4 Model Training

5.4.1 Artificial Neural Network (ANN)

In this section, the performance of the Artificial Neural Network (ANN)[33] model will be trained and evaluated[101]. The ANN architecture will be constructed, the model will be compiled with suitable metrics, and the dataset will be used for training the model[17]. The ANN will be implemented using the **TensorFlow Keras API**[60].



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 50176)             0

 dense (Dense)               (None, 512)               25690624

 batch_normalization (BatchN  (None, 512)              2048
 ormalization)

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 batch_normalization_1 (Batc  (None, 256)              1024
 hNormalization)

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 batch_normalization_2 (Batc  (None, 128)              512
 hNormalization)

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 batch_normalization_3 (Batc  (None, 64)               256
 hNormalization)

 dropout_3 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 1)                 65

=================================================================
Total params: 25,867,009
Trainable params: 25,865,089
Non-trainable params: 1,920
```

Figure 5.1: Artificial Neural Network (ANN)'s Architecture

To create an ANN model, an empty Sequential model was first created and a Flatten() layer was added to convert the input tensor into a 1D tensor. Next, Dense() layers were stacked with decreasing numbers of neurons (512, 256, 128, and 64)[17]. Each Dense() layer is followed by a BatchNormalization() layer to improve training and generalization, and a Dropout() layer to prevent overfitting. All Dense() layers, except for the last one, use the **ReLU** activation function and have L1 and L2 regularization applied to their kernels to reduce overfitting[17]. The final Dense() layer was added with one neuron and a **Sigmoid** activation function for binary classification. In the image above **Fig: 5.1** you can see the architecture of ANN model.

With the model created, it was compiled using **binary_crossentropy** as the loss function for the binary classification task, the **Adam** optimizer for its efficiency, and a learning rate of **1e-4**[17][102]. The model was trained for **50** epochs. To check the code for training the model using ANN, refer to **Code: B**. The following are the results obtained after training the dataset using ANN model:

Metric	Result for Validation	Results for Training
Accuracy	64.54%	63.54%
Precision	64.73%	63.53%
F1 Score	67.50%	67.54%
Loss	59.90%	59.97%
AUC	68.98%	67.97%
Recall	66.80%	66.90%

Table 5.1: Results of ANN Model

The **Table 5.1** presents the performance metrics[101] of the ANN model on both validation and training sets. The model demonstrates relatively similar performance across the datasets, with accuracy, precision, and recall at 64.54%, 64.73%, and 66.80% for validation, and 63.54%, 63.53%, and 66.90% for training, respectively[101]. The F1 score and Area under the Curve (AUC) further show a balanced performance in terms of precision and recall[101]. Despite the model's ability to generalize well to unseen data, there is room for improvement in its performance. Enhancements could be achieved by fine-tuning the model architecture, optimizing hyper-parameters, or employing advanced techniques such as transfer learning[17].

The time taken for this model was relatively average, but the accuracy is not satisfactory. As mentioned in the literature review in **Section: 2.4**, an ANN model is not as effective as a CNN and cannot outperform it. Typically, the accuracy of ANN varies, and it can achieve high results as high as 90% and as low as 50% when training a model for medical images[49][50]. Compared to these results, the model is underperforming. There is room for development, and the model has not overfit[64] after 50 epochs, indicating that it could perform better if trained further; however, due to time constraints, it was not possible to train it further. The confusion matrix[101] of this dataset was also plotted, as shown below in the **Fig: 5.2**::

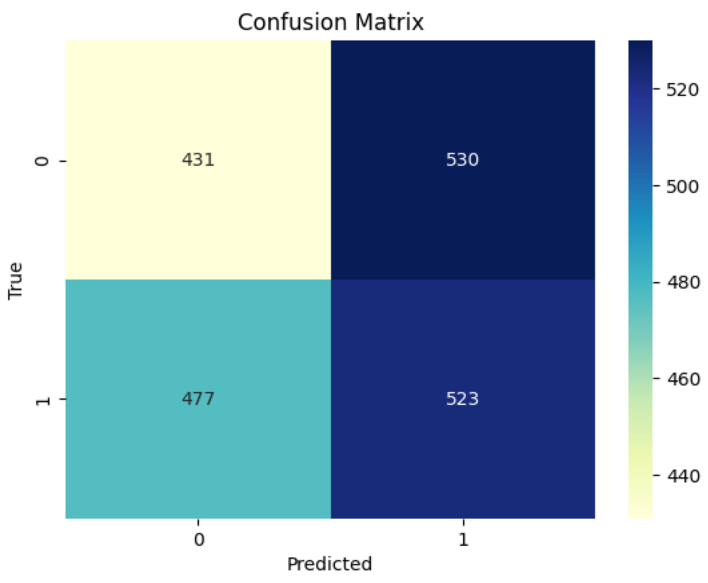


Figure 5.2: ANN's Confusion Matrix

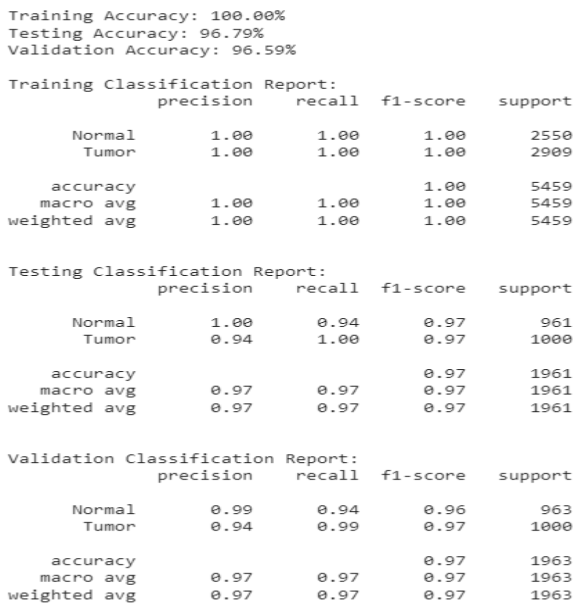
As you can see in **Fig: 5.2** the confusion matrix also suggests that the results are not quite satisfactory, as the number of false positives and false negatives is relatively high, though the true positives and true negatives are still significant[101]. In conclusion, the performance was not exceptional. This marks the completion of the ANN model training[17], and the next step is to train the SVM model[45].

5.4.2 Support Vector Machine (SVM)

In this section, the Support Vector Machine (SVM) model[18] is trained and evaluated using **scikit-learn**[61], which simplifies the process by automating model creation, compilation, and training[45]. However, manual feature extraction is required, which can be time-consuming as discussed in **Section: 2.4**. After training, the model's performance is assessed with various evaluation metrics[101]. As SVM can take considerable time to train on large datasets, the process is optimized through data preprocessing[83], feature selection, and parameter tuning to achieve the best performance and minimize training time.

To train the model, a decision must be made regarding the use of a linear or nonlinear kernel. A linear kernel is chosen for linearly separable data or when there are many features, while a nonlinear kernel is used when data is not linearly separable[45]. The C value, a regularization parameter, and the probability value must be set to enable probability estimates for classification tasks[45]. In this case, a linear kernel is chosen, C is set as 1, and the probability is set as true.

There's no direct way to visualize an SVM architecture, and the number of epochs doesn't need to be set. The model learns from the training data until it is fully learned, avoiding overfitting[64][18]. Therefore, the training of the model proceeds. It is worth mentioning that SVM can generate a classification report, but the classification report generated by SVM doesn't include AUC and loss scores[45]. To check the code for training the model using SVM, refer to **Code: B**. The following classification report presents the results of training the SVM in **Fig: 5.3**:


```
Training Accuracy: 100.00%
Testing Accuracy: 96.79%
Validation Accuracy: 96.59%

Training Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00      2550
       Tumor       1.00      1.00      1.00      2909

    accuracy                           1.00      5459
   macro avg       1.00      1.00      1.00      5459
weighted avg       1.00      1.00      1.00      5459

Testing Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      0.94      0.97       961
       Tumor       0.94      1.00      0.97      1000

    accuracy                           0.97      1961
   macro avg       0.97      0.97      0.97      1961
weighted avg       0.97      0.97      0.97      1961

Validation Classification Report:
              precision    recall  f1-score   support

      Normal       0.99      0.94      0.96       963
       Tumor       0.94      0.99      0.97      1000

    accuracy                           0.97      1963
   macro avg       0.97      0.97      0.97      1963
weighted avg       0.97      0.97      0.97      1963
```

Figure 5.3: SVM's Classification Report

The results presented in **Table 5.2** show that the SVM model[18] demonstrates excellent performance for both validation[67] and training sets[65]. For the validation set, the model achieves an accuracy of 96.59%, a precision of 96.50%, and a recall rate of 96.50%. Similarly, for the training set, the model exhibits an accuracy of 96.79%, a precision of 97.00%, and a recall rate of 97.00%. The F1 scores for both the validation and training sets are also impressive, at 96.50% and 97.00% respectively, indicating a good balance between precision and recall.

After training the SVM model, following results were obtained:

Metric	Result for Validation	Results for Training
Accuracy	96.59%	96.79%
Precision	96.50%	97.00%
F1 Score	96.50%	97.00%
Recall	96.50%	97.00%

Table 5.2: Results of SVM Model

The results of the SVM model[18] indicate excellent performance across key classification metrics, such as accuracy, precision, recall, and F1-score, for both training and validation sets[101]. The model demonstrates a strong ability to predict class labels for new samples and maintains a good balance between precision and recall, as evidenced by its high F1 score. It is important to note that the time taken for training and evaluation of this SVM model was not as high as expected from the literature review it was minimal. Although SVM accuracy typically ranges from 85% to 95%[45][18], the model outperformed this range with an accuracy of approximately **96.79%** and the preprocessing pipeline's effectiveness can be observed[15]. Overall, the results are impressive. The confusion matrix[101] of this dataset has also been plotted, which is shown in the **Fig: 5.4**.

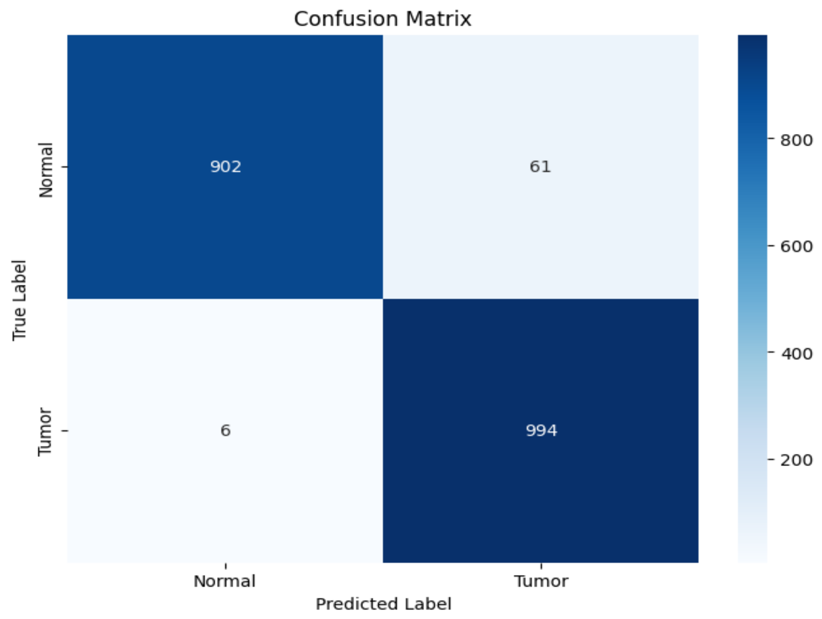
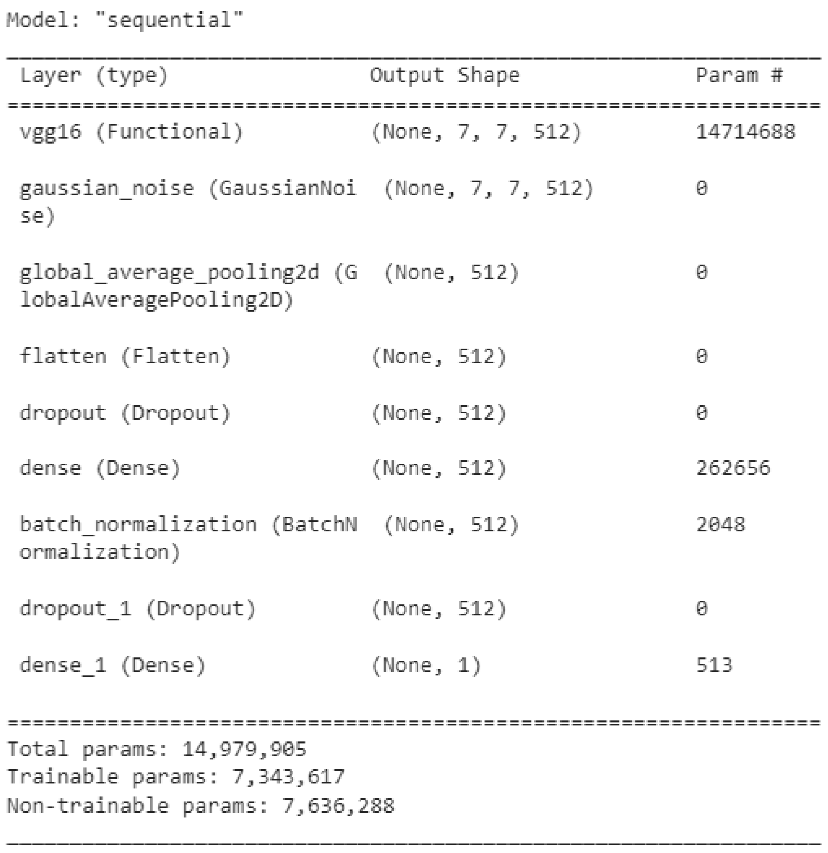


Figure 5.4: SVM's Confusion Matrix

As seen in the **Fig: 5.4**, the confusion matrix confirms that the results are exceptional, with the model accurately distinguishing between true positives and negatives as well as false positives and negatives[101]. Comparing the results of the SVM[45] and ANN models[17], there is no contest: the SVM model excels, while the ANN model is only average in its performance. This marks the completion of SVM model[18] training, and the next step is to move on to CNN[19].

5.4.3 Convolutional Neural Network (CNN)

In this section, the performance of the Convolutional Neural Network (CNN) model[51][19] is trained and evaluated. The CNN architecture for image classification is built using **TensorFlow's Keras API**[60], the model is compiled with appropriate metrics, and then trained on the dataset[19]. Once trained, its performance is assessed using various evaluation metrics to determine its effectiveness[101]. As discussed in **Section: 2.4**, the **Visual Geometry Group (VGG16)** architecture is used for training the model[19]. Although not the most resource-efficient, it is highly accurate.


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 gaussian_noise (GaussianNoi  (None, 7, 7, 512)        0
 se)

 global_average_pooling2d (G  (None, 512)              0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 512)               0

 dropout (Dropout)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 batch_normalization (BatchN  (None, 512)              2048
 ormalization)

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 14,979,905
Trainable params: 7,343,617
Non-trainable params: 7,636,288
_________________________________________________________________
```

Figure 5.5: CNN VGG16's Architecture

To construct the model, the base model is used and layers are added on top[51]. GaussianNoise() is added to reduce overfitting, GlobalAveragePooling2D() to decrease spatial dimensions and a Flatten() layer to transform the output tensor to a 1D tensor[19]. Dropout() layers and Dense() layers with L2 regularization are incorporated to prevent overfitting, and BatchNormalization() to improve training and generalization[19]. Lastly, another Dropout() layer and a Dense() layer with a single unit and **Sigmoid** activation are added for binary classification. The summary of the VGG16 model can be seen in **Fig: 5.5**.

With the model ready, it is compiled using metrics like Accuracy, Precision, Loss, F1 Score, AUC, and Recall[101][51]. **binary_crossentropy** is employed as the loss function, suitable for binary classification tasks, and the **Adam** optimizer for efficiency[51]. The learning rate is set at **1e-4** and the VGG16 architecture is trained for **50** epochs[19]. The code for training a model using a CNN model is similar to that of an ANN model. You can refer to **Code: B** to review the code. The **Table 5.3**, presents the results obtained after training:

Metric	Result for Validation	Results for Training
Accuracy	94.40%	93.17%
Precision	97.14%	96.36%
F1 Score	67.61%	67.75%
Loss	19.91%	22.44%
AUC	98.33%	98.27%
Recall	91.70%	90.00%

Table 5.3: Results of CNN Model

As seen in **Table 5.3**, the CNN model[19] performs well in key metrics for both validation[67] and training sets[65], achieving an accuracy of 94.40% for validation and 93.17% for training. However, the F1 score is relatively lower at 67.61% for validation and 67.75% for training, suggesting a need for improvement in balancing precision and recall. The loss percentage is 19.91% for validation and 22.44% for training, indicating another potential area for enhancement. The AUC score is high at 98.33% for validation and 98.27% for training, signifying the model's strong ability to discriminate between positive and negative samples in both datasets[101].

It is important to note that the training and evaluation time for this CNN model[19] is relatively high, which could be a concern in terms of computational efficiency. However, exceptional results are achieved, with the VGG16 model reaching an accuracy of **93.17%**

while the average accuracy for VGG16 usually lies between 85% - 90%[51, 19], which is remarkable considering its complexity. Although there is still room for improvement, the model has not over-fitted[64] after 50 epochs and could perform even better if trained further. The confusion matrix[101] of this dataset has also been plotted as it can be seen in **Fig: 5.6**.

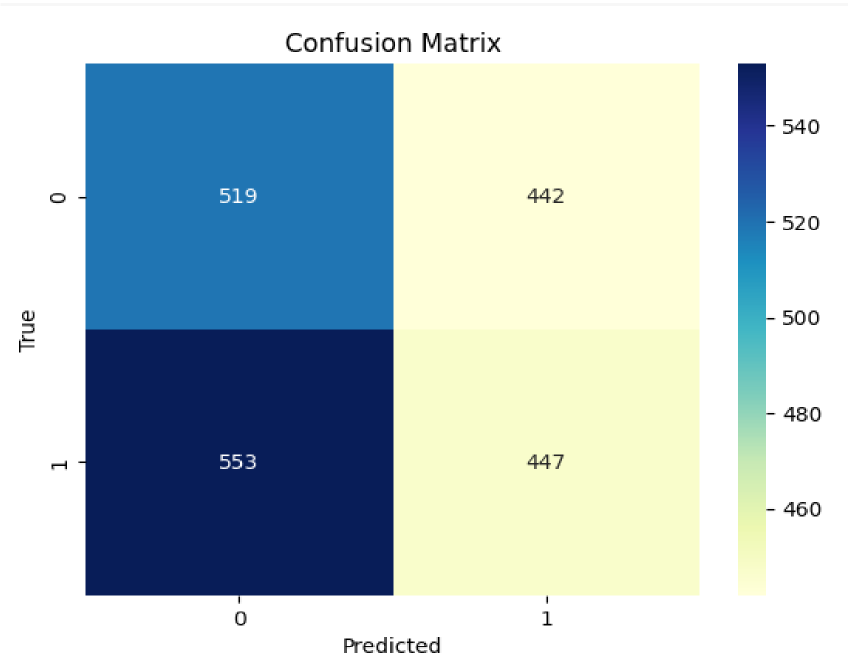


Figure 5.6: CNN's Confusion Matrix

The confusion matrix shows satisfactory results, but there is room for improvement, as the number of false negatives is relatively high[101]. Although the overall accuracy is high, the model is time-consuming, making it less efficient compared to the SVM model[18]. Therefore, the SVM model may be considered superior compared to both CNN's VGG16 model[19] and ANN model[17]. This completes the training of the CNN model[19], and the evaluation of transformers[52] can now proceed.

5.4.4 Transformers

In this section, the performance of the Transformers model[52] will be trained and evaluated. The Transformers architecture for image classification will be built using **PyTorch**[62], compiled with appropriate metrics[101], and trained on the dataset. Once trained, its performance will be assessed using various evaluation metrics to determine its effectiveness[101]. As discussed in **Section: 2.4**, the **Swin Transformer** architecture will be used for training the model, as it is more efficient and accurate[43].

To construct the model, the base Swin Transformer's pre-trained model will be created with *timm.create_ model*[52]. The loss function used is **BCEWithLogitsLoss()**, as there are only two classes, and the optimizer is **Adam**[48]. There is no direct way to print the structure in PyTorch[62], unlike in TensorFlow[60], so the structure of the created model cannot be printed. The training is set to **20** epochs, as one epoch takes almost 2 hours to complete, and it would be impractical to spend 100 hours training the model with 50 epochs. To check the code for training the model using the Transformers model, refer to **Code: B**. The **Table 5.4**, presents the results obtained after training:

Metric	Result for Validation	Results for Training
Accuracy	96.89%	96.58%
Precision	94.25%	93.72%
F1 Score	97.04%	96.76%
Loss	21.50%	23.49%
AUC	96.83%	96.51%
Recall	100.00%	100.00%

Table 5.4: Results of Transformers Model

As seen in **Table 5.4**, the Transformers model's[20] results show high performance, with 96.89% accuracy and 97.04% F1 score on the validation set, and 96.58% accuracy and 96.76% F1 score on the training set. Precision is 94.25% for validation and 93.72% for training. Loss values are 21.50% and 23.49% for validation and training, respectively. The AUC scores are 96.83% and 96.51% for validation and training, and the model achieves 100% recall on both sets, which is highly impressive as it means there are no false negatives[101].

It is important to note that the training and evaluation time for this Transformers model[48] is very high, which could be a concern in terms of computational efficiency. However, exceptional results are achieved, with the Transformers model[20] reaching an accuracy of **96.58%**. Since the Swin Transformer[48] is a relatively new method and hasn't been used extensively in the medical field, there are no credible resources available to compare the recorded accuracy. However, it is still the second best after the SVM model[45], with a very small difference between both accuracies. The confusion matrix[101] of this dataset has also been plotted as shown in the **Fig: 5.7**.

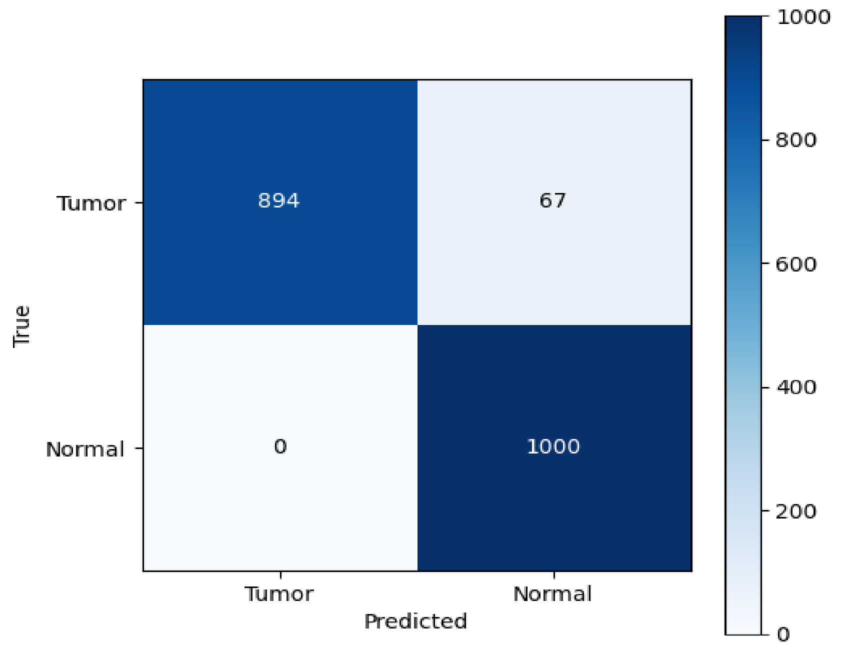


Figure 5.7: Transformers' Confusion Matrix

Fig: 5.7 shows the confusion matrix, which presents excellent results, with zero false negatives and only 64 false positives[101]. The accuracy and overall prediction of the model to identify false positives are remarkable, except for the high computational power required. Therefore, the SVM model[18] may still be considered superior in this case due to its lower computational demands, but this model has achieved accuracy higher than even the CNN. This completes the training of the Swin Transformer model[52], and all models have now been trained and evaluated. In the next section, the accuracy and efficiency of each trained model will be compared.

5.5 Evaluation of Models

In this section, four models were trained, including ANN[17], SVM[18], CNN[19], and Transformers[48]. Almost all methods demonstrated exceptional accuracy, except for ANN. The table below compares the accuracy and efficiency of each model[101].

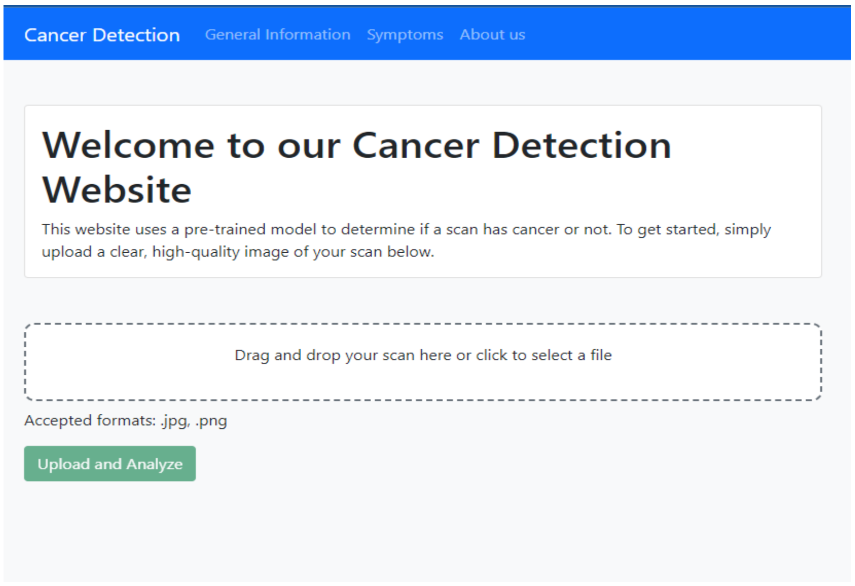
In **Table: 5.5**, it is observed that the highest accuracy was achieved by the SVM model[18], with 96.79%, followed by the Transformers model[20] at 96.58%. The precision of the SVM model is quite high at 97.00%, but the CNN model[19] also has a higher precision of 96.36% compared to the Transformers model at 93.72%. The Transformers model has the highest recall at 100.00%, which means there are no false positives, while the recall is lower for the other models[101]. Resource consumption for the SVM model is lower than both the CNN model and the Transformers model. Therefore, it can be concluded that the SVM model is one of the best in terms of accuracy, precision, and computational requirements. However, it is important to consider that the requirements and accuracy will vary for each dataset.

Model	Accuracy	Precision	Recall	Computational Requirement
ANN	63.54%	63.53%	66.90%	Moderate
SVM	96.79%	97.00%	97.00%	Low
CNN	93.17%	96.36%	90.00%	High
Transformers	96.58%	93.72%	100.00%	Very High

Table 5.5: Comparison of Models

5.6 Model Deployment

In previous sections, various models were trained and evaluated on the dataset, with the **Support Vector Machine (SVM)** model emerging as the most accurate and efficient choice, boasting the highest precision. This section focuses on deploying the final SVM model and creating a user-friendly PHP website that allows users to upload their scans for analysis. A Python script runs the model in the background, and the results are displayed to the user.



Welcome to our Cancer Detection Website

This website uses a pre-trained model to determine if a scan has cancer or not. To get started, simply upload a clear, high-quality image of your scan below.

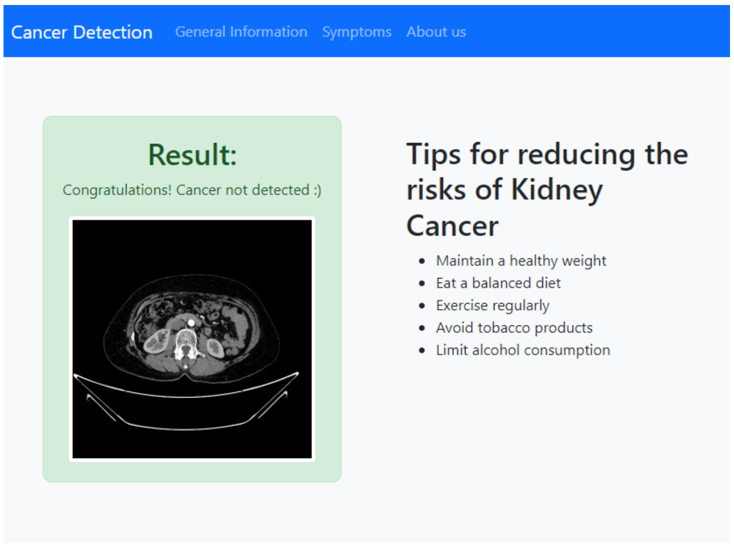
Drag and drop your scan here or click to select a file

Accepted formats: .jpg, .png

Figure 5.8: Front Page of the Website

The initial page of the website, shown in **Fig: 5.8**, allows users to drag and drop or select a scan for analysis.

After uploading an image, the user can click on the **Upload and Analyze** button, which triggers the analysis process. The website then displays whether the uploaded image is cancerous or non-cancerous. **Fig: 5.9** and **Fig: 5.10** show the responses for normal and cancerous images, respectively. For non-cancerous images, the website offers users tips on maintaining their health. For cancerous images, it provides a list of recommended doctors, with the first visit being free or discounted.



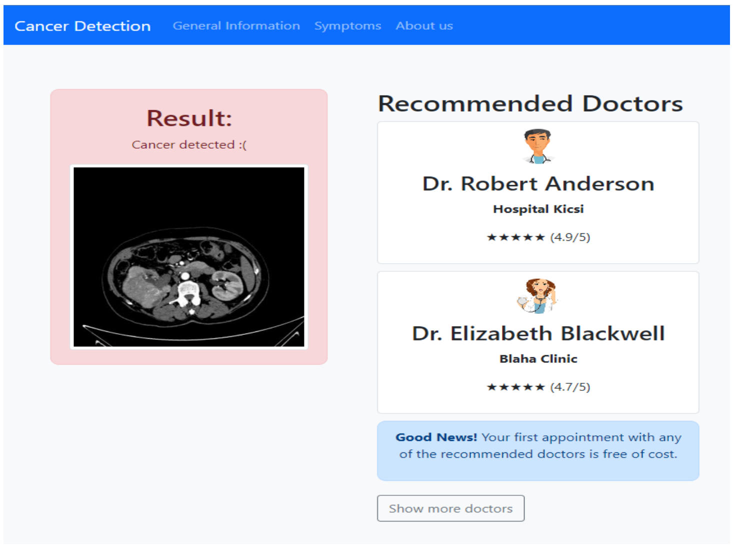
Result:

Congratulations! Cancer not detected :)

Tips for reducing the risks of Kidney Cancer

- Maintain a healthy weight
- Eat a balanced diet
- Exercise regularly
- Avoid tobacco products
- Limit alcohol consumption

Figure 5.9: Response for a Normal Image



Result:

Cancer detected :(

Recommended Doctors

Dr. Robert Anderson
Hospital Kicsi
★★★★★ (4.9/5)

Dr. Elizabeth Blackwell
Blaha Clinic
★★★★★ (4.7/5)

Good News! Your first appointment with any of the recommended doctors is free of cost.

Figure 5.10: Response for a Cancerous Image

The deployment of the SVM model in a user-friendly website demonstrates the practical application of the developed machine learning classifier. This application has the potential to facilitate early detection and diagnosis of kidney cancer, ultimately contributing to improved patient outcomes and reduced healthcare costs.

5.7 Summary

This chapter provided a detailed account of the training and evaluation processes for each model[17][18][19][20], including the architecture, metrics, and loss functions used[101]. Results were presented in the form of tables and figures, showcasing the performance metrics and confusion matrices[101]. Some challenges were encountered during the training process, such as long training times for CNN and Transformers models, and model complexity.

The performances of the models were compared, with the **Support Vector Machine (SVM)** model[18] being considered superior in terms of accuracy computational efficiency, despite the impressive results achieved by the CNN and Transformers models[19][20]. The focus was also on how the preprocessing pipeline[15] contributed to achieving higher accuracy than average models found on the internet. Furthermore, in the final stages of this research, the SVM model was deployed, showcasing a practical application of the trained model with the potential to enhance patient outcomes. The chapter concludes with a comprehensive understanding of the strengths and limitations of each model in the context of image classification.

Chapter 6

Conclusion

6.1 Summary

This thesis presented a thorough investigation into developing a machine learning model for the classification of Renal Cell Carcinoma (RCC) in medical images. The primary aim was to create an accurate and efficient classifier capable of differentiating between malignant and non-cancerous images, thus contributing to earlier cancer detection and improved patient outcomes.

The research commenced with an extensive literature review, followed by the careful selection of five renal cancer-related datasets. Multiple image quality metrics and statistical measures were employed to evaluate the datasets and choose the most suitable one for further analysis. The chosen dataset was then preprocessed using a series of techniques, such as outlier removal, denoising, edge detection, histogram equalization, and data augmentation, resulting in enhanced model performance.

Several machine learning models, including Artificial Neural Network (ANN), Support Vector Machine (SVM), Convolutional Neural Network (CNN) (VGG16), and Transformer (Swin Transformer), were assessed. The findings showed that the SVM model achieved the highest level of accuracy and precision, making it the ideal choice for implementation in a practical application. The Transformer model, however, exhibited the best recall, which could be essential for applications where reducing false negatives is a priority.

To demonstrate the practical implications of the research, a website was developed, enabling users to upload scans for analysis by the deployed SVM model. The website provided instant feedback on whether the uploaded image was malignant or noncancerous, as well as health advice or physician recommendations based on the results.

This thesis showcased the potential of machine learning, specifically the **Support Vector Machine (SVM)** model, in detecting and diagnosing renal cancer at an early stage. Through the development of a comprehensive preprocessing pipeline using Open Source Computer Vision Library (OpenCV) and the selection of an effective classifier, the research has made a significant contribution to the field, with the potential to positively impact patient outcomes and reduce healthcare costs. The user-friendly website serves as an example of how the classifier can be applied in real-world scenarios, offering a valuable tool for early kidney cancer detection.

6.2 Limitations

Even though this research has made significant contributions to the field of Renal Cell Carcinoma (RCC) detection using machine learning, several limitations must be acknowledged.

- **Binary Classification:** Differentiating between malignant and non-cancerous images were the focus of the current study, which focused on a binary classification problem. The model may not be directly pertinent to classification problems involving multiple classes, such as identifying different stages or types of cancer.
- **Dataset Scope:** The selected dataset was limited to Renal Cell Carcinoma (RCC), limiting the applicability of the findings to other organs or varieties of cancer. In addition, the scale of the dataset may not be sufficient to convey the complexity and variation of kidney cancer cases in the actual world.
- **Comparing Models:** Although several machine learning models were evaluated in this study, it is possible that other models or techniques could produce superior results. Moreover, model hyperparameters and architectural decisions may also influence the results.

6.3 Future Work

Several avenues for future research can be proposed to build on the findings of this thesis and resolve its identified limitations:

- **Multi-class Classification:** Future research could investigate the extension of the developed model to multi-class classification problems, enabling the model to identify various stages or types of cancer, thereby enhancing its clinical utility.

- **Broader Dataset Scope:** Investigating the performance of the developed model on datasets about other organs or cancer types could aid in determining its generalizability and applicability to a broader array of diagnostic scenarios.
- **Model Improvements:** Continuous refinement of the machine learning model, including the investigation of alternative architectures, hyperparameter optimization, and the incorporation of new techniques, could aid in enhancing its accuracy and precision in detecting Renal Cell Carcinoma (RCC).
- **Integration with Healthcare Systems:** Research into the practicality and potential impact on patient outcomes and healthcare costs of integrating the developed classifier into existing healthcare systems could yield valuable insights.
- **Validation in the Real World:** Evaluating the performance of the model using real-world data and in clinical settings would provide a more accurate evaluation of its efficacy and potential for adoption in routine diagnostic procedures.

By pursuing these prospective research directions, the work presented in this thesis can be refined and expanded, ultimately leading to the development of more accurate and efficient diagnostic instruments for the detection of Renal Cell Carcinoma (RCC) and the improvement of patient care.

Appendix A

Additional Source Code for Data Cleaning and Preprocessing

```python
# Get the mean and standard deviation of pixel intensities
mean_intensity, std_intensity  = calculate_mean_intensities(dataset)

outliers = {"Normal": [], "Tumor": []}

for category in ["Normal", "Tumor"]:
    for filename, image in dataset[category].items():

        # Calculate the image's mean intensity and z-score
        image_mean_intensity = np.mean(image)
        z_score = (image_mean_intensity - mean_intensity) /
            std_intensity

        # If z-score is greater than threshold then it is an outlier
        if abs(z_score) > threshold:
            outliers[category].append(filename)
```

Code A.1: Code to get all the outliers found in our Dataset

```python
edges_dataset = {"Normal": {}, "Tumor": {}}

for category in ["Normal", "Tumor"]:
    for file_path, scan in dataset[category].items():

        # We will add extra dimension and adjust the contrast
        scan_extended = tf.expand_dims(scan, -1)
        scan_ac = tf.image.adjust_contrast(scan_extended, cf)
        scan_ac = tf.squeeze(scan_ac, -1)
        # We will use Canny to detect edges and store it
        scan_ac_np = scan_ac.numpy()
        edges_scan = cv2.Canny(scan_ac_np.astype(np.uint8), low_th,
           high_th)
        edges_dataset[category][file_path] = edges_scan
```

Code A.2: Code to detect edges in images

```
for category in ["Normal", "Tumor"]:
    for file_path, scan in dataset[category].items():
        # Get the edge-detected scan
        edges_scan = edges_dataset[category][file_path]

        # Normalize the original scan and edge scan
        og_normalize_scan = cv2.normalize(scan, None, alpha=0, beta=255,
            norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
        edges_normalize_scan = cv2.normalize(edges_scan, None, alpha=0,
           beta=255, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)

        # Blend the original scan with the edges scan and store
        blended_scan = cv2.addWeighted(og_normalize_scan, blend_r,
           edges_normalize_scan, 1 - blend_r, 0)
        blended_dataset[category][file_path] = blended_scan
```

Code A.3: Code to blend edges with original images

Appendix B

Additional Source Code for Model Training and Evaluation

```python
# Here we will train the ANN Model
history = model.fit(
    train_generator,                        # Provide Training Generator
    steps_per_epoch = steps_per_epoch,      # Number of Steps per Epoch
    epochs = epochs,                        # Total Epochs for Training
    validation_data = valid_generator,      # Provide Validation Generator
    validation_steps = validation_steps,    # Number of Validation Steps
    callbacks = callbacks                   # Previously defined Callbacks
)
```

Code B.1: Code for Training of ANN Model

```python
from sklearn.svm import SVC

# Here we will train the SVM Model
svm = SVC(kernel='linear', C=1, probability=True)
svm.fit(train_features_scaled, train_labels)
```

Code B.2: Code for Training of SVM Model

```python
for epoch in range(num_epochs):
    # Set model to training mode and Initialize the loss
    model.train()
    running_loss = 0.0

    for inputs, labels in train_loader:
        # Move input and label tensors and zero gradient optimizer
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        # Forward pass, compute loss and backward pass
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()

        # Update model parameters and accumulate the loss
        optimizer.step()
        running_loss += loss.item()

    # Compute the average loss for this epoch and print it
    avg_loss = running_loss / len(train_loader)
    print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {avg_loss}')
```

Code B.3: Code for Training of Transformers Model

Bibliography

[1] Max Roser and Hannah Ritchie. *Deaths From Cancer*. Accessed: April 5, 2023. n.d. URL: `https://ourworldindata.org/cancer`.

[2] American Cancer Societ. *Survival Rates For Kidney Cancer*. Accessed: April 5, 2023. 2023. URL: `https://www.cancer.org/cancer/kidney-cancer/detection-diagnosis-staging/survival-rates.html`.

[3] Wikipedia. *Renal Cell Carcinoma — Wikipedia*. Accessed: April 5, 2023. 2021. URL: `https://en.wikipedia.org/wiki/Renal_cell_carcinoma`.

[4] American Cancer Society. *Key Statistics About Kidney Cancer*. Accessed: April 5, 2023. 2023. URL: `https://www.cancer.org/cancer/kidney-cancer/about/key-statistics.html`.

[5] Ping Jiang et al. "A Renal Cell Carcinoma Tumorgraft Platform To Advance Precision Medicine". In: (2021). Accessed: April 5, 2023. DOI: `10.1016/j.celrep.2021.110055`. URL: `https://www.sciencedirect.com/science/article/pii/S2211124721015412`.

[6] Wikipedia. *Machine Learning — Wikipedia*. Accessed: April 5, 2023. 2023. URL: `https://en.wikipedia.org/wiki/Machine_learning`.

[7] Hassan Habehh and Sweta Gohel. "Machine Learning In Healthcare". In: (2021). Accessed: April 5, 2023. DOI: `10.2174/1389202922666210705124359`. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8822225/`.

[8] Microsoft Learn. *What Is A Machine Learning Model?* Accessed: April 6, 2023. 2021. URL: `https://learn.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model`.

[9] Wikipedia. *Deep Learning — Wikipedia*. Accessed: April 5, 2023. 2021. URL: `https://en.wikipedia.org/wiki/Deep_learning`.

[10] Wikipedia. *Artificial Intelligence — Wikipedia*. Accessed: April 5, 2023. 2023. URL: `https://en.wikipedia.org/wiki/Artificial_intelligence`.

[11] National Cancer Institute. *Financial Burden Of Cancer Care | Cancer Trends Progress Report.* Accessed: April 5, 2023. 2022. URL: `https://progressreport.cancer.gov/after/economic_burden`.

[12] Wikipedia. *OpenCV — Wikipedia.* Accessed: April 6, 2023. 2012. URL: `https://en.wikipedia.org/wiki/OpenCV`.

[13] Wikipedia. *Magnetic Resonance Imaging — Wikipedia.* Accessed: April 5, 2023. 2022. URL: `https://en.wikipedia.org/wiki/Magnetic_resonance_imaging`.

[14] Wikipedia. *CT Scan — Wikipedia.* Accessed: April 5, 2023. 2021. URL: `https://en.wikipedia.org/wiki/CT_scan`.

[15] C. Fan et al. "A Review On Data Preprocessing Techniques Toward Efficient And Reliable Knowledge Discovery From Building Operational Data". In: (2021). Accessed: April 5, 2023. DOI: `10.3389/fenrg.2021.652801`. URL: `https://www.frontiersin.org/articles/10.3389/fenrg.2021.652801/full`.

[16] Tomoyuki Makino et al. "Epidemiology And Prevention Of Renal Cell Carcinoma". In: (2022). Accessed: April 6, 2023. DOI: `10.3390/cancers14164059`. URL: `https://www.mdpi.com/2072-6694/14/16/4059`.

[17] Dora Radecic. "TensorFlow For Computer Vision - How To Train Image Classifier With Artificial Neural Networks". In: (2021). Accessed April 6, 2023. URL: `https://towardsdatascience.com/tensorflow-for-computer-vision-how-to-train-image-classifier-with-artificial-neural-networks-304bc82f3dd`.

[18] Jiang Yun et al. "An Improved SVM Classifier For Medical Image Classification". In: Accessed April 6, 2023. 2007. ISBN: 978-3-540-73450-5. DOI: `10.1007/978-3-540-73451-2_80`. URL: `https://www.researchgate.net/publication/221052858_An_Improved_SVM_Classifier_for_Medical_Image_Classification`.

[19] Zhi-Peng Jiang et al. "An Improved VGG16 Model For Pneumonia Image Classification". In: (2021). Accessed April 6, 2023. DOI: `10.3390/app112311185`. URL: `https://doi.org/10.3390/app112311185`.

[20] Ali Hatamizadeh et al. "Swin UNETR: Swin Transformers For Semantic Segmentation Of Brain Tumors In MRI Images". In: (2022). Accessed April 6, 2023. URL: `https://arxiv.org/abs/2201.01266`.

[21] n.d. *Hidden Stratification Causes Clinically Meaningful Failures In Machine Learning For Medical Imaging.* Accessed: April 6, 2023. n.d. URL: `https://dl.acm.org/doi/abs/10.1145/3368555.3384468`.

[22] Md Sohel Mahmood. *Outlier Detection.* Accessed: April 5, 2023. 2022. URL: `https://towardsdatascience.com/outlier-detection-part1-821d714524c`.

[23] T. A. Team. *Image De-noising Using Deep Learning.* Accessed: April 5, 2023. 2023. URL: `https://towardsai.net/p/deep-learning/image-de-noising-using-deep-learning`.

[24] Chithra P.L and Bhavani P. *A Study On Various Image Processing Techniques.* Accessed: April 5, 2023. 2019. URL: `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3388008`.

[25] Kyaw Saw Htoon. *A Tutorial To Histogram Equalization.* Accessed: April 5, 2023. 2020. URL: `https://medium.com/@kyawsawhtoon/a-tutorial-to-histogram-equalization-497600f270e2`.

[26] Caiming Shorten and Taghi M. Khoshgoftaar. “A Survey On Image Data Augmentation For Deep Learning”. In: (2019). Accessed: April 5, 2023. DOI: `10.1186/s40537-019-0197-0`. URL: `https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0`.

[27] S. Madeleine. *Normalization, Zero Centering And Standardization Of CT Images.* Accessed: April 5, 2023. n.d./. URL: `https://imaios.com/en/resources/blog/ct-images-normalization-zero-centering-and-standardization`.

[28] Datagen. *AI/ML Model Training: Data, Models, And Metrics - Datagen.* Accessed: April 6, 2023. n.d. URL: `https://datagen.tech/guides/data-training/model-training/`.

[29] n.d. “Artificial Intelligence In Medical Imaging: Switching From Radiographic pathological data to clinically meaningful endpoints”. In: (2020). Accessed: April 6, 2023. DOI: `10.1016/S2589-7500(20)30160-6`. URL: `https://doi.org/10.1016/S2589-7500(20)30160-6`.

[30] Ana Barragán-Montero et al. “Artificial Intelligence And Machine Learning For Medical Imaging: A Technology Review”. In: (2021). Accessed: April 6, 2023. DOI: `10.1016/j.ejmp.2021.04.016`. URL: `https://www.sciencedirect.com/science/article/pii/S1120179721001733`.

[31] Martin A Makary and Michael Daniel. “Medical error-the Third Leading Cause Of Death In The US”. In: (2016). Accessed: April 6, 2023. DOI: `10.1136/bmj.i2139`. URL: `https://doi.org/10.1136/bmj.i2139`.

[32] Wikipedia. *Convolutional Neural Network — Wikipedia.* Accessed: April 5, 2023. 2019. URL: `https://en.wikipedia.org/wiki/Convolutional_neural_network`.

[33] Wikipedia. *Artificial Neural Network — Wikipedia.* Accessed: April 5, 2023. 2019. URL: https://en.wikipedia.org/wiki/Artificial_neural_network.

[34] Wikipedia. *Support Vector Machine — Wikipedia.* Accessed: April 5, 2023. 2017. URL: https://en.wikipedia.org/wiki/Support_vector_machine.

[35] Kai Xie, Jianhua Zhang, and Hong Zhang. "Deep Learning In Computer Vision: A Critical Review Of Emerging Techniques And Application Scenarios". In: (2021). Accessed: April 6, 2023. DOI: https://doi.org/10.1016/j.mlwa.2021.100134. URL: http://www.sciencedirect.com/science/article/pii/S2666827021000246.

[36] n.d. *PubMed.* Accessed: April 6, 2023. n.d. URL: https://pubmed.ncbi.nlm.nih.gov/.

[37] n.d. *IEEE Xplore.* Accessed: April 6, 2023. n.d. URL: https://ieeexplore.ieee.org/Xplore/home.jsp.

[38] n.d. *National Institutes Of Health (NIH).* Accessed: April 6, 2023. n.d. URL: https://www.nih.gov/.

[39] n.d. *ScienceDirect.com | Science, Health And Medical Journals, Full Text Articles and Books.* Accessed: April 6, 2023. n.d. URL: https://www.sciencedirect.com/.

[40] Raghda Hajjo et al. "Identification Of Tumor-Specific MRI Biomarkers Using Machine Learning (ML)". In: (2021). Accessed: April 6, 2023. DOI: 10.3390/diagnostics11050742. URL: https://pubmed.ncbi.nlm.nih.gov/33919342/.

[41] Maedeh Barahman. "Renal Cell Carcinoma: An Overview Of The Epidemiology, Diagnosis, And Treatment - GIN". In: (2022). Accessed: April 6, 2023. URL: https://giornaleitalianodinefrologia.it/2022/05/39-03-2022-03/.

[42] IBM. *Supervised Vs. Unsupervised Learning: What's the Difference?* Accessed: April 6, 2023. 2021. URL: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.

[43] M.A. Elshikh et al. *Transformers In Medical Image Analysis.* Accessed April 6, 2023. 2022. DOI: 10.1016/j.imed.2022.07.002. URL: https://doi.org/10.1016/j.imed.2022.07.002.

[44] Zilliz Vector Database Blog. *All You Need To Know About Artificial Neural Network In Machine Learning.* Accessed: April 6, 2023. n.d. URL: https://zilliz.com/blog/ANN-machine-learning.

[45] Venu Shanmukh. “Image Classification Using Machine Learning-Support Vector Machine(SVM)”. In: (2021). Accessed April 6, 2023. URL: https://medium.com/analytics-vidhya/image-classification-using-machine-learning-support-vector-machine-svm-dc7a0ec92e01.

[46] Jun Yan, Yuan Wang, and Changming Liu. *Downscaling GCMs Using The Smooth Support Vector Machine Method*. Accessed: April 6, 2023. 2013. URL: https://www.researchgate.net/figure/Architecture-of-the-Support-Vector-Machine-The-coefficients-w-and-b-are-the-adjustable_fig2_225366566.

[47] Intellipaat. *What Is Convolutional Neural Network - CNN Tutorial*. Accessed: April 6, 2023. n.d. URL: https://intellipaat.com/blog/tutorial/artificial-intelligence-tutorial/convolution-neural-network/.

[48] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows*. Accessed: April 6, 2023. 2021. URL: https://www.arxiv-vanity.com/papers/2103.14030/.

[49] Osamu Manabe et al. “Diagnostic Accuracy Of An artificial Neural Network Compared With Statistical Quantitation Of Myocardial Perfusion Images: A Japanese Multicenter Study”. In: (2018). Accessed April 6, 2023. DOI: 10.1007/s00259-017-3834-x. URL: https://doi.org/10.1007/s00259-017-3834-x.

[50] J. Jiang, P. Trundle, and J. Ren. *Medical Image Analysis With Artificial Neural Networks*. Accessed April 6, 2023. 2022. URL: https://www.studypool.com/documents/22736429/medical-image-analysis-with-artificial-neural-networks.

[51] Nitesh Raj Pathak. “Lungs Disease Prediction Using Medical Imaging with Implementation Of VGG, ResNet”. In: (2020). Accessed April 6, 2023. URL: https://medium.com/analytics-vidhya/lungs-disease-prediction-using-medical-imaging-with-implementation-of-vgg-resnet-and-183e73b85df9.

[52] Yucheng Tang et al. “Self-supervised Pre-training Of Swin Transformers For 3D Medical Image Analysis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Accessed April 6, 2023. 2022. URL: https://arxiv.org/abs/2111.14791.

[53] M. A. Sabbagh et al. “Deep Convolutional Neural Network-based Software Improves Radiologist Detection Of Malignant Lung Nodules On Chest Radiographs”. In: (2020). Accessed: April 6, 2023. DOI: 10.1148/radiol.2019182465. URL: https://doi.org/10.1148/radiol.2019182465.

[54] K. Jiang et al. "Current Evidence And Future Perspective Of Accuracy Of Artificial Intelligence Application For Early Gastric Cancer Diagnosis With Endoscopy: A Systematic And Meta-Analysis". In: (2021). Accessed: April 6, 2023. DOI: `10.3389/fmed.2021.629080`. URL: `https://doi.org/10.3389/fmed.2021.629080`.

[55] H. C. Shin et al. "Convolutional Neural Networks For Medical Image Analysis: Full Training Or Fine Tuning?" In: (2016). Accessed: April 6, 2023. DOI: `10.1109/TMI.2016.2535302`. URL: `https://doi.org/10.1109/TMI.2016.2535302`.

[56] T. Magadza and S. Viriri. "Deep Learning For Brain Tumor Segmentation: A Survey Of State-of-the-Art". In: (2021). Accessed: April 6, 2023. DOI: `10.3390/jimaging7020019`. URL: `https://doi.org/10.3390/jimaging7020019`.

[57] P. D. Chang et al. "Deep-Learning Convolutional Neural Networks Accurately Classify Genetic Mutations In Gliomas". In: (2018). Accessed: April 6, 2023. DOI: `10.3174/ajnr.A5667`. URL: `https://doi.org/10.3174/ajnr.A5667`.

[58] Xiangbin Liu et al. "A Review Of Deep-Learning-Based Medical Image Segmentation Methods". In: (2021). Accessed: April 6, 2023. DOI: `10.3390/su13031224`. URL: `https://www.mdpi.com/2071-1050/13/3/1224`.

[59] Vajira Thambawita et al. "Impact Of Image Resolution On Deep Learning Performance In Endoscopy Image Classification: An Experimental Study Using A Large Dataset Of Endoscopic Images". In: (2021). Accessed: April 6, 2023. DOI: `10.3390/diagnostics11122183`. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8700246/`.

[60] TensorFlow. *Module: tf.keras | TensorFlow v2.12.0.* Accessed April 6, 2023. n.d. URL: `https://www.tensorflow.org/api_docs/python/tf/keras`.

[61] Scikit learn contributors. *Scikit-learn: Machine Learning In Python.* Accessed April 6, 2023. n.d. URL: `https://scikit-learn.org/stable/`.

[62] PyTorch. *PyTorch.* Accessed April 6, 2023. n.d. URL: `https://www.pytorch.org`.

[63] Anjali. *Selecting a Dataset 101.* Accessed April 6, 2023. 2021. URL: `https://medium.com/codex/selecting-a-dataset-101-a3275ca08628`.

[64] Wikipedia. *Overfitting — Wikipedia.* Accessed: April 6, 2023. 2017. URL: `https://en.wikipedia.org/wiki/Overfitting`.

[65] MonkeyLearn. *What Is Training Data In Machine Learning?* Accessed: April 5, 2023. 2020. URL: `https://monkeylearn.com/blog/training-data/`.

[66] Deepchecks. *What is Test Set in Machine Learning.* Accessed: April 6, 2023. n.d. URL: https://deepchecks.com/glossary/test-set-in-machine-learning/.

[67] Deepchecks. *What is Validation Set in Machine Learning.* Accessed: April 6, 2023. n.d. URL: https://deepchecks.com/glossary/validation-set-in-machine-learning/.

[68] n.d. *Google Dataset Search.* Accessed April 6, 2023. n.d. URL: https://datasetsearch.research.google.com/.

[69] n.d. *Kaggle: Your Machine Learning And Data Science Community.* Accessed April 6, 2023. n.d. URL: https://www.kaggle.com/.

[70] The Cancer Imaging Archive. *The Cancer Imaging Archive (TCIA).* Accessed April 6, 2023. n.d. URL: https://www.cancerimagingarchive.net/.

[71] n.d. *NBIA Image Archive.* Accessed: April 6, 2023. n.d. URL: https://nbia.cancerimagingarchive.net/nbia-search/.

[72] Guangtao Zhai and Xiongkuo Min. "Perceptual Image Quality Assessment: A Survey". In: (2020). Accessed April 6, 2023, pp. 1–22. URL: https://doi.org/10.1007/s11432-019-2757-1.

[73] LearnOpenCV. *Image Quality Assessment: BRISQUE.* Accessed April 6, 2023. 2018. URL: https://learnopencv.com/image-quality-assessment-brisque/.

[74] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. "Making A Completely Blind Image Quality Analyzer". In: (2012). Accessed April 6, 2023, pp. 209–212. URL: https://ieeexplore.ieee.org/document/6353522.

[75] Wikipedia. *Laplace Distribution — Wikipedia.* Accessed April 6, 2023. 2023. URL: https://en.wikipedia.org/wiki/Laplace_distribution.

[76] n.d. *Mean And Median Values – Image Processing And Computer Vision.* Accessed April 6, 2023. n.d. URL: https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20172018/LectureNotes/MATH/meanandmedian.html.

[77] Martha. *What Is Mean And Standard Deviation In Image Processing?* Accessed April 6, 2023. 2022. URL: https://www.icsid.org/uncategorized/what-is-mean-and-standard-deviation-in-image-processing/.

[78] Shen-Chuan Tai and Shih-Ming Yang. "A Fast Method For Image Noise Estimation Using Laplacian Operator And Adaptive Edge Detection". In: (2008). Accessed April 6, 2023. DOI: 10.1109/ISCCSP.2008.4537384. URL: https://www.researchgate.net/publication/224317132_A_fast_method_for_image_noise_estimation_using_Laplacian_operator_and_adaptive_edge_detection.

[79] Abinash Swain. "Noise In Digital Image Processing". In: (2020). Accessed April 6, 2023. URL: https://medium.com/image-vision/noise-in-digital-image-processing-55357c9fab71.

[80] Jakub Cieślik. *How To Do Data Exploration For Image Segmentation And Object Detection.* Accessed April 6, 2023. 2023. URL: https://neptune.ai/blog/data-exploration-for-image-segmentation-and-object-detection.

[81] V7 Labs. *Data Cleaning In Machine Learning: Steps & Process.* Accessed April 6, 2023. n.d. URL: https://www.v7labs.com/blog/data-cleaning-guide.

[82] S. Agrawal. "How To Split Data Into Three Sets (Train, Validation, And Test) And Why?" In: (2021). Accessed April 6, 2023. URL: https://towardsdatascience.com/how-to-split-data-into-three-sets-train-validation-and-test-and-why-e50d22d3e54c.

[83] isahit. *What Is The Purpose Of Image Preprocessing In Deep Learning?* Accessed April 6, 2023. 2022. URL: https://www.isahit.com/blog/what-is-the-purpose-of-image-preprocessing-in-deep-learning.

[84] Yun Zhang et al. "ImageDC: Image Data Cleaning Framework Based On Deep Learning". In: (2020). Accessed April 6, 2023. DOI: 10.1109/ICAIIS49377.2020.9194803. URL: https://www.researchgate.net/publication/344859637_ImageDC_Image_Data_Cleaning_Framework_Based_on_Deep_Learning.

[85] A. P. Gulati. "Dealing With Outliers Using The Z-Score Method". In: (2022). Accessed April 6, 2023. URL: https://www.analyticsvidhya.com/blog/2022/08/dealing-with-outliers-using-the-z-score-method/.

[86] Google Books. *Image Processing.* Accessed April 6, 2023. n.d. URL: https://books.google.com/books/about/Image_Processing.html?id=smBw4-xvfrIC.

[87] L. Fan et al. "Brief Review Of Image Denoising Techniques". In: (2019). Accessed April 6, 2023. DOI: 10.1186/s42492-019-0016-7. URL: https://doi.org/10.1186/s42492-019-0016-7.

[88] Wikipedia. "Gaussian Blur — Wikipedia". In: (2009). Accessed April 6, 2023. URL: https://en.wikipedia.org/wiki/Gaussian_blur.

[89] StackPath. *Filtering Techniques Eliminate Gaussian Image Noise*. Accessed April 6, 2023. n.d. URL: https://www.vision-systems.com/home/article/14174546/filtering-techniques-eliminate-gaussian-image-noise.

[90] J. D. Seo. "Computer Vision Feature Extraction 101 On Medical Images — Part 1: Edge Detection / Sharpening / Blurring / Emboss / Super Pixel". In: (2018). Accessed April 6, 2023. URL: https://towardsdatascience.com/computer-vision-feature-extraction-101-on-medical-images-part-1-edge-detection-sharpening-42ab8ef0a7cd.

[91] S. Sobhana and R. Sukeshini. "A Survey On Various Edge Detector Techniques". In: (2012). Accessed April 6, 2023. DOI: doi.org/10.1016/j.protcy.2012.05.033. URL: https://www.sciencedirect.com/science/article/pii/S2212017312001013.

[92] Vaishnavi M. *Comprehensive Guide To Edge Detection Algorithms*. Accessed April 6, 2023. 2022. URL: https://www.analyticsvidhya.com/blog/2022/08/comprehensive-guide-to-edge-detection-algorithms/.

[93] Wunderman Thompson Technology. *Edge Detection And Processing Using Canny Edge Detector And Hough Transform*. Accessed April 6, 2023. 2022. URL: https://wttech.blog/blog/2022/edge-detection-and-processing-using-canny-edge-detector-and-hough-transform/.

[94] Ziqi Xu et al. "Edge Detection Algorithm Of Medical Image Based On Canny Operator". In: (2021). Accessed April 6, 2023. DOI: 10.1088/1742-6596/1955/1/012080. URL: https://dx.doi.org/10.1088/1742-6596/1955/1/012080.

[95] B.Ramamurthy. *Content Based Image Retrieval For Medical Images Using Canny Edge Detection Algorithm*. Accessed April 6, 2023. 2011. URL: shorturl.at/ginNO.

[96] Animesh Karnewar. *Back-to-basics Part 1: Histogram Equalization In Image Processing*. Accessed April 6, 2023. 2018. URL: https://medium.com/@animeshsk3/back-to-basics-part-1-histogram-equalization-in-image-processing-f607f33c5d55.

[97] Robert Allred. *Image Augmentation For Deep Learning Using Keras And Histogram Equalization*. Accessed April 6, 2023. 2018. URL: https://towardsdatascience.com/image-augmentation-for-deep-learning-using-keras-and-histogram-equalization-9329f6ae5085.

[98] Coresignal. *Data Normalization: Definition, Importance, And Advantages.* Accessed April 6, 2023. n.d. URL: https://coresignal.com/blog/data-normalization/.

[99] S. Sudhakar. “Histogram Equalization”. In: (2021). Accessed April 6, 2023. URL: https://towardsdatascience.com/histogram-equalization-5d1013626e64.

[100] V7labs. *Activation Functions In Neural Networks [12 Types & Use Cases].* Accessed April 6, 2023. n.d. URL: https://www.v7labs.com/blog/neural-networks-activation-functions.

[101] Ankita Mishra. “Metrics To Evaluate Your Machine Learning Algorithm”. In: (2020). Accessed: April 6, 2023. URL: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

[102] Nikolaos Kafritsas. “Create Image Classification Models With Tensorflow In 10 Minutes”. In: *Medium* (2022). Accessed April 6, 2023. URL: https://towardsdatascience.com/create-image-classification-models-with-tensorflow-in-10-minutes-d0caef7ca011.

[103] Benjamin Neo. “Building An Image Classification Model With PyTorch From Scratch”. In: (2022). Accessed April 6, 2023. URL: https://medium.com/bitgrit-data-science-publication/building-an-image-classification-model-with-pytorch-from-scratch-f10452073212.

List of Figures

List of Tables

List of Codes

List of Acronyms

AI Artificial Intelligence. 7

ANN Artificial Neural Network. 11, 12, 16, 18–21, 23, 27, 53–55, 58–60, 62, 63, 83, 85

AUC Area under the Curve. 54, 56, 59, 61

BRISQUE Blind/Referenceless Image Spatial Quality Evaluator. 29–31, 34, 35, 83

CNN Convolutional Neural Network. 11, 12, 16–18, 20, 21, 23, 27, 53, 54, 58–60, 62, 63, 65, 83–85

CT Computed Tomography. 8, 9, 11–13, 15, 18, 20, 21, 23

IQA Image Quality Assessment. 31

ML Machine Learning. 7–12, 21, 22, 24, 25

MRI Magnetic Resonance Imaging. 8, 9, 11–13, 15, 18, 20, 21, 23

NSS Natural Scene Statistics. 30

OpenCV Open Source Computer Vision Library. 4, 9, 11, 12, 21–25, 29, 32, 42, 44, 46

RCC Renal Cell Carcinoma. 12, 14

ReLU Rectified Linear Activation Unit. 51, 52

SVM Support Vector Machine. 11, 12, 16, 17, 19–21, 23, 27, 53, 55–58, 60–63, 65, 66, 83, 85

VGG16 Visual Geometry Group. 4, 20, 58–60, 66, 84