Augmentation Techniques to Improve Automatic Head and Neck Cancer Segmentation in Positron Emission Tomography Images

Rahim Kargar

M.Sc. Thesis

DEPARTMENT OF MATHEMATICS AND STATISTICS

UNIVERSITY OF TURKU
Department of Mathematics and Statistics

KARGAR, RAHIM: Augmentation Techniques to Improve Automatic Head and Neck Cancer Segmentation in Positron Emission Tomography Images
M.Sc. Thesis, 62 pages
Statistics
May 2025

This M.Sc. thesis studies a deep learning-based approach for automatic segmentation of head and neck cancer (HNC) in positron emission tomography (PET) images using U-Net architecture. The study includes a detailed analysis of PET imaging data from 89 patients with confirmed positive HNC diagnoses, focusing on significant image slices identified through mask data and examining metrics such as the distribution of pixel intensities, the number of relevant slices processed per patient, and the performance of thresholding techniques for region identification. Segmentation accuracy is primarily evaluated using the Dice coefficient, to assess predictive performance and result reliability.

Recently, Liedes et al. [24] investigated the effectiveness of machine learning models in segmenting PET images. Specifically, when assessing true positive segmentations, these models achieved mean and median Dice scores of 0.79 and 0.69, respectively on test datasets of PET images, where the Image dimension is 128, the threshold value is 0.25, and the pixel limit is 5. This thesis thoughtfully explores various cases concerning Image dimension, threshold values, and pixel limits, providing valuable insights for future research. Furthermore, we explore and refine augmentation techniques to enhance these results, including specific rotations, combinations of original and augmented training sets, and augmentations of training images and masks. Tests included applying random rotations between -15 and 15, 90° clockwise rotations, and techniques like horizontal flipping and Gaussian blurring.

For model validation, five cross-validation test sets were examined over five iterations each, producing a comprehensive evaluation with both mean and median Dice coefficients computed across all 25 iterations. These final averaged metrics provide a summary of model performance across all the cases. This research highlights the promise of advanced U-Net-based models and tailored data augmentation strategies to improve PET image segmentation, supporting the broader integration of precise, automated cancer segmentation in clinical settings.

Keywords: Head and Neck Cancer, Convolutional Neural Network, Segmentation, Augmentation, PET Images.

Acknowledgements

Completing this MSc thesis has been a challenging yet rewarding journey, particularly as I simultaneously pursued my PhD studies. Balancing these two endeavors required significant dedication and perseverance, further compounded by the fact that most of the MSc courses were conducted in Finnish—a language unfamiliar to me at the outset. I am profoundly grateful to my PhD supervisor, Professor Matti Vuorinen, whose unwavering support and assistance in translating course materials made it possible for me to navigate these challenges successfully.

I would like to extend my heartfelt thanks to my MSc thesis supervisors, Dr. Oona Rainio, Professor Janne Kujala, and Professor Riku Klén for their invaluable guidance, feedback, and encouragement throughout this process. I deeply appreciate the mentorship and insights of Professor Matti Vuorinen, whose suggestions and assistance in writing this thesis were invaluable. A special thank you goes to Joonas Liedes for generously allowing me access to his data, which greatly enriched my research. Lastly, I would like to express my gratitude to Professor Henri Nyberg, Head of Statistics at the University of Turku, and Dr. Joni Virta, for dedicating their time and effort in helping me select the subject of my thesis and providing their valuable expertise throughout the process.

To everyone who contributed to my academic journey, I am sincerely thankful for your support, which made this achievement possible.

May 2025
Rahim Kargar

Abbreviations

AUC	Area Under the Receiver Operating Characteristic (ROC) Curve
CM	Center of Mass
CPU	Central Processing Unit
CNN	Convolutional Neural Network
CT	Computed Tomography
DC	Dice Coefficient
DL	Deep Learning
FC	Fully Connected
FCN	Fully Convolutional Network
FNN	Feedforward Neural Network
GAN	Generative Adversarial Network
GLM	Generalized Linear Model
GPU	Graphics Processing Unit
HNC	Head and Neck Cancer
HPV	Human Papillomavirus
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ICD	International Classification of Diseases
IDE	Integrated Development Environment
IoU	Intersection over Union
MRI	Magnetic Resonance Imaging
NIfTI	Neuroimaging Informatics Technology Initiative
PET	Positron Emission Tomography
RBF	Radial Basis Function
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
ROI	Region of Interest
SUV	Standardized Uptake Value
TPU	Tensor Processing Unit

Contents

1 Introduction

Cancer is currently the top cause of death and disability worldwide and the second-leading cause of death in developed countries. Head and neck cancer (HNC) is the seventh most common cancer worldwide, causing more than 660,000 new cases and 325,000 deaths each year [17]. About 90% of HNCs are squamous cell carcinomas originating from the epithelial layer of the oral cavity, pharynx, and larynx. Many types of head and neck cancers are classified according to their anatomical location using the World Health Organization's International Classification of Diseases (ICD-10) [23]. HNC is typically treated with radiotherapy alone or in combination with other methods such as chemotherapy and surgery [10].

Positron Emission Tomography (PET) imaging plays an essential role in the treatment of HNC, providing metabolic insights that are crucial for tumor delineation. Despite its importance, automatic segmentation of HNC in PET images remains a challenge due to the low contrast, noise, and anatomical variability inherent in these images.

To overcome these challenges and improve the accuracy of automatic segmentation models, data augmentation techniques have proven to be powerful tools. Data augmentation artificially increases the size and diversity of training datasets by introducing transformations such as rotation, scaling, and flipping, helping models generalize better to unseen data. This approach is particularly valuable in medical image analysis, where data availability is often limited and model robustness is critical.

In this thesis, we focus on the application of data augmentation techniques to enhance the performance of deep learning models for automatic segmentation of HNC in PET images. Using methods such as geometric transformations (rotation, scaling), intensity adjustments (noise addition), and combining augmented data with original data, we aim to create a robust model capable of addressing challenges like tumor variability in shape, size, and position. We also evaluate the impact of applying these augmentations collaboratively, showing that they lead to improved segmentation performance.

The structure of this thesis is as follows: Chapter 2 provides the theoretical background necessary for understanding the core concepts related to head and neck cancer, PET imaging, and image segmentation. Chapter 3 presents a detailed literature review of existing research on segmentation techniques and data augmentation in medical imaging. Chapter 4 outlines the methods and materials used in this study, including the data augmentation techniques employed and the deep learning model architecture. In Chapter 5, we present the results of our experiments, followed by a discussion of these findings in Chapter 6. Finally, Chapter 7 concludes the thesis and suggests potential directions for future research.

2 Theoretical Background

In this chapter, we discuss the basic concepts and theories that form the foundation of the techniques and models described in this work. We start by giving a thorough overview of CNNs, which are essential to modern machine learning and artificial intelligence.

2.1 What is a Neural Network?

A neural network is a type of computational model inspired by the structure and function of biological CNNs found in the human brain. This model consists of interconnected nodes, or *neurons*, arranged in layers. It is a fundamental component of machine learning and is used to perform various tasks such as classification, regression, image and speech recognition, natural language processing, and more. The network operates by learning from data inputs and adjusting its internal parameters to produce accurate outputs. This technology is essential in modern artificial intelligence applications and has the potential to significantly impact diverse fields such as healthcare, finance, robotics, and beyond. The structure of a neural network includes neurons, layers, weights, and biases.

Neurons (Nodes): The basic units of a neural network, analogous to neurons in the brain. Each neuron receives inputs, processes them, and produces an output.

Layers: Layers include the input layer, hidden layer, and output layer. The input layer is the first layer in the neural network, responsible for receiving the initial data. Each neuron in this layer represents a feature of the input data. For example, in image processing, each neuron might represent a pixel value.

Hidden layers lie between the input and output layers. These layers perform various computations and transformations on the input data to extract meaningful patterns and features. Each hidden layer consists of multiple neurons, which process the inputs received from the previous layer using weighted connections and activation functions. The number of neurons in hidden layers and the number of hidden layers can vary, depending on the complexity of the problem being solved. More neurons and layers can enable the network to learn more intricate patterns, but they also require more computational resources and careful tuning to avoid overfitting.

The output layer is the final layer in the neural network. The neurons in this layer produce the final predictions or classifications based on the transformed data from the previous layers. The number of neurons in the output layer corresponds to the number of target classes in a classification task or the number of predicted values in a regression task.

Weights and Biases: Each neuron has a bias value that helps the model fit the data better.

Some of the types of CNNs are as follows:

- Feedforward CNNs (FNNs)
- Convolutional CNNs (CNNs)
- Recurrent CNNs (RNNs)

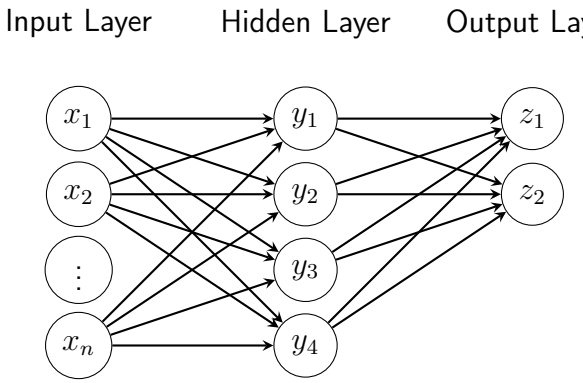


Figure 1: A simple neural network.

- Generative Adversarial Networks (GANs).

A neural network has applications in image and video recognition, natural language processing, speech recognition, game playing, and healthcare. To sum up, a neural network is a potent and adaptable machine learning model capable of understanding intricate patterns and making predictions or decisions based on data. Its design and operation are influenced by the structure and operation of the human brain. A simple neural network is illustrated in Figure 1. This network has an input size of n, a hidden layer size of 4, and a binary output layer.

2.2 Convolutional CNNs (CNNs)

Convolutional CNNs have significantly impacted numerous areas, particularly computer vision. It is safe to say that CNNs have had a significant impact on the field of computer vision. Their journey, which began several decades ago, has been driven by continuous innovation inspired by research in biology. Here is a brief look into their fascinating history:

Early Inspiration (1950s-1980s): In the 1950s, David Hubel and Torsten Wiesel discovered simple and complex cells in the human visual cortex, suggesting how the brain extracts features from images. This laid the groundwork for CNNs [20]. Kunihiko Fukushima introduced the neocognitron, inspired by Hubel and Wiesel's work. It featured convolutional layers and pooling layers, laying the foundation for modern CNN architecture [12].

The Rise of Modern CNNs (1990s-2000s): Yann LeCun and his team developed LeNet-5, a CNN specifically designed for handwritten digit recognition. This marked a significant breakthrough in applying CNNs to real-world problems [38]. Limited computational power and small datasets hindered further development in the 1990s and early 2000s.

The Breakthrough and Beyond (2010s-Present): The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) provided a large dataset and benchmark for image classification. AlexNet, a deep CNN by Alex Krizhevsky et al., achieved groundbreaking results, surpassing traditional methods and propelling CNNs into the spotlight

[27]. Increased computational power, larger datasets, and architectural improvements like VGGNet, ResNet, and EfficientNet fueled explosive growth in CNN research and applications. CNNs started dominating various tasks like object detection, image segmentation, and medical image analysis [58]. Research continues to push the boundaries of CNNs, exploring areas like interpretability, efficiency, and robustness. They are also being integrated with other AI techniques for even more powerful applications [57].

In CNNs, a convolutional neural network (CNN) is one of the main categories for image recognition and image classification [41]. Object detection and recognition of faces are some of the areas where CNNs are widely used. CNN image classifications take an input image, process it, and classify it under certain categories (e.g., dog, cat, tiger, lion). Computers see an input image as an array of pixels, depending on the image resolution. For a given resolution, the image is represented as $h \times w \times d$, where h is the height, w is the width, and d is the number of color channels (e.g., $d = 1$ for grayscale, $d = 3$ for RGB. RGB stands for Red, Green, and Blue, which are the three primary colors of light used in digital imaging and screen displays.). For instance, an image of $64 \times 64 \times 3$ array of a matrix of RGB and an image of $32 \times 32 \times 1$ array of a matrix of a grayscale image. In deep learning, CNN models are used to train and test images. Each input image goes through a series of convolution layers with filters, pooling, and fully connected layers (FC). The Softmax function is then applied to classify an object with probabilistic values between 0 and 1.

A CNN is a modification of the artificial neural network that focuses on imitating the functionality and behavior of our visual cortex. The hidden units aim to learn nonlinear variations of the original inputs; this is called extraction of features or creation of features. Then these hidden features are transferred to the final Generalized Linear Model (GLM) as input. In simple language, a CNN resembles a brain-inspired computer system that is especially good at understanding visual data. It has special parts that learn to pick out important details from pictures, and then it uses those details to make sense of what it is looking at. Finally, it puts all that information together to make a decision. This method works well for problems when the original details by themselves do not tell us much. Think of it like this: one pixel in a picture does not say a lot, but when you put pixels together, you can see what is in the image. On the flip side, just knowing how many instances of each word appear in text — the so-called bag-of-words method — can be effective for simpler tasks, like organizing or categorizing textual data based on specific criteria. CNNs have gained significant attention for their effectiveness in recognizing visual patterns, particularly through a specific type known as CNNs. While CNNs excel in image processing, they are also versatile tools for handling other types of data, including speech signals and text. This adaptability makes them invaluable in various applications across artificial intelligence, enabling them to analyze and interpret different forms of information beyond just images. In this network, certain areas pay attention to specific parts of an image, kind of like how our eyes focus on different things. These areas share and reuse their knowledge across the whole image, which helps to keep things simpler and faster. Therefore, if the network finds something important in one part of the image, it can recognize the same thing in other parts without having to learn it all over again. This makes the network good at recognizing patterns no matter where they show up in the picture [33].

However, as a quick refresher, the following are the key concepts worth reiterating

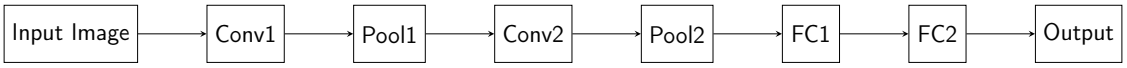


Figure 2: An example of a CNN architecture (LeNet).

[42, Ch. 12]:

Convolutional Layer: The main thing that sets CNNs apart from other types of CNNs is their convolutional layer, often called a "conv layer". This layer consists of filters that can learn and detect different features in an image. These filters are small but cover the entire depth of the image, meaning they consider all the colors. Here is how it works: Imagine sliding a filter over the image, looking at a small portion at a time. At each position, the filter multiplies its attributes with the corresponding parts of the input image. This gives us a two-dimensional map showing how well the filter matches that part of the image. We do this for each filter, stacking the maps together to create the final output.

Pooling Layer: These layers are like shrinking tools in a photo editor. They make the picture smaller and simpler, which helps prevent it from being too detailed and overdone. We put these shrink tools between other layers to make the image smaller using techniques like picking a maximum, finding the average, or using some mathematical formulas such as L2-norm. We need to figure out which pixels to group and how to assign the replacement value. Usually, we pick pixels from a square area, but the number of pixels we combine can vary depending on the situation [25].

Fully Connected (FC) Layer: The FC layer, short for Fully Connected Layer, resembles the typical layers we find in most CNNs. It connects every neuron from the previous layer to every neuron in this layer. Its main job is to help with classification tasks.

Parameter Sharing: CNNs have something special besides their convolutional layers – they share parameters. This means that the convolutional layers use the same set of weights at different spatial locations, which helps cut down on the total number of parameters needed. An example of a CNN architecture (LeNet) with all the components is depicted in Figure 2.

2.3 Applications and Benefits in Medical Imaging

Incorporating CNNs into medical imaging has transformed healthcare by improving diagnostic processes' precision, efficiency, and capacity. These developments not only enhance patient results but also streamline clinical workflows. The following discussion explores the different uses and advantages of CNNs in medical imaging.

2.3.1 Applications in Medical Imaging

In the following, we list some of the most important applications of CNNs.

Disease Detection and Diagnosis: Disease detection and diagnosis utilize advanced neural network techniques to enhance medical imaging analysis. CNNs, particularly CNNs, are extensively used for the early detection and diagnosis of cancers such as breast cancer, lung cancer, and skin cancer. For instance, CNNs can analyze mammograms to identify malignant tumors, often achieving accuracy comparable to that of experienced radiologists. Deep learning models are employed to detect and monitor neurological conditions such as Alzheimer's disease, multiple sclerosis, and stroke. Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) scans analyzed by CNNs can reveal early signs of these disorders, facilitating timely intervention. CNNs assist in diagnosing cardiovascular conditions by analyzing echocardiograms, MRI scans, and CT angiographies, enabling the detection of anomalies such as blocked arteries, heart valve issues, and congenital heart defects.

Image Segmentation: Image segmentation plays a vital role in enhancing the analysis of medical images by accurately identifying Region of interest. Accurate segmentation of tumors in medical images is crucial for treatment planning and monitoring. CNNs can delineate the boundaries of tumors in MRI and CT scans, providing precise measurements of tumor size and growth. Identifying and segmenting organs in medical images is essential for various diagnostic and therapeutic procedures. CNNs can segment organs such as the liver, lungs, and kidneys, improving the accuracy of volumetric assessments and surgical planning.

Image Enhancement and Reconstruction: Image enhancement and reconstruction techniques significantly improve the quality and utility of medical images. CNNs can enhance the quality of medical images by reducing noise and artifacts, which is particularly beneficial in low-dose imaging techniques like low-dose CT scans. Deep learning models can increase the resolution of medical images, allowing for more detailed visualization of anatomical structures. This is especially useful in modalities like MRI, where high resolution is critical for accurate diagnosis. CNNs facilitate the reconstruction of high-quality images from limited or incomplete data, such as sparse-view CT or accelerated MRI scans. This approach reduces the need for extensive imaging sessions, thereby minimizing patient exposure to radiation and discomfort.

Predictive Analytics and Personalized Medicine: Predictive analytics and personalized medicine leverage advanced neural network techniques to enhance patient care and treatment outcomes. By analyzing patterns in medical imaging data, CNNs can predict disease progression and patient outcomes. This capability helps clinicians develop personalized treatment plans and make informed decisions. CNNs can evaluate changes in medical images over time to monitor the effectiveness of treatments. For example, they can assess tumor shrinkage in response to chemotherapy, enabling timely adjustments to treatment plans.

In summary, the application of CNNs in medical imaging is transforming the field of healthcare by enhancing diagnostic accuracy, efficiency, and patient outcomes. As these technologies continue to advance, they hold the promise of further improving the quality of care and making healthcare more accessible and effective for patients worldwide. By leveraging the power of CNNs, the medical community can achieve new heights in the

detection, diagnosis, and treatment of diseases.

2.3.2 Benefits in Medical Imaging

In the sequel, we present some of the benefits of CNNs in medical imaging.

Increased Diagnostic Accuracy: CNNs can identify subtle patterns and anomalies in medical images that might be missed by human eyes. This leads to higher diagnostic accuracy, reducing the likelihood of misdiagnosis, and ensuring that patients receive appropriate care.

Enhanced Efficiency: The automation of image analysis tasks with CNNs significantly speeds up the diagnostic process. Clinicians can obtain results faster, leading to quicker diagnosis and treatment initiation. This efficiency is particularly beneficial in emergencies and high-throughput environments.

Consistency and Reproducibility: CNNs provide consistent and reproducible results, eliminating the variability associated with human interpretation. This standardization ensures that diagnostic outcomes are reliable and can be trusted across different healthcare settings.

Reduced Clinician Workload: By automating routine and time-consuming tasks, CNNs free up clinicians to focus on more complex and nuanced aspects of patient care. This alleviates the workload on healthcare professionals and reduces burnout.

Improved Patient Outcomes: Early and accurate diagnosis facilitated by CNNs leads to timely and appropriate interventions, which can significantly improve patient outcomes. Additionally, precise image segmentation and enhanced imaging quality contribute to better treatment planning and monitoring.

Cost-Effectiveness: The use of CNNs in medical imaging can reduce healthcare costs by streamlining workflows, minimizing the need for repeat imaging, and improving resource allocation. This cost-effectiveness is particularly important in resource-limited settings.

Advancements in Research: CNNs provide powerful tools for medical research, enabling the analysis of large datasets and the discovery of new insights into disease mechanisms and treatment responses. This drives innovation and the development of new diagnostic and therapeutic approaches.

2.4 Image Segmentation Using CNNs

CNNs offer an alternative method for pixel classification in image segmentation. Unlike traditional methods, they do not rely on a pre-existing class probability distribution for accurate classification. Instead, CNNs adapt the decision boundaries for pixel classification through an iterative training process. Neural network-based segmentation methods can yield high-quality results for medical images with significant variability in structures

of interest. For instance, angiographic images exhibit considerable diversity in arterial structures, making segmentation challenging. The variation in image quality among different angiograms and the introduction of noise during image acquisition highlight the necessity of an adaptive, nonparametric segmentation approach. Neural network paradigms such as backpropagation, radial basis function (RBF), and self-organizing feature maps have been utilized for the segmentation of medical images.

CNNs learn from examples in the training set where the pixel classification task has already been done manually. They know a nonlinear mapping function between the input features and the desired output for labeled examples without using any parameterization. After the learning process, a neural network can classify a pixel in a new image for segmentation.

Choosing the right features for input data is crucial for a neural network's classification. It is also important to select training examples that represent a complete statistical distribution of the input data. The network's structure and the distribution of training examples significantly influence its accuracy, ability to generalize, and robustness. At its simplest, a neural network can take the gray values of pixels in a specific area of an image as input. This allows the network to classify the central pixel of the area based on information from all the pixels in that area. As the area moves across the image, the network can classify the pixels at the center of the translated areas [7, Section 10.4.2].

3 Literature Review

3.1 Overview of Tumor Segmentation in PET Images

This section overviews tumor segmentation in PET images and introduces some challenges and techniques.

3.1.1 Introduction to Tumor Segmentation

Tumor segmentation in medical imaging involves delineating the boundaries of tumors from surrounding tissues, which is crucial for accurate diagnosis, treatment planning, and monitoring of the response to therapy. In PET imaging, this task is particularly significant because PET provides functional information about metabolic activity, helping to identify malignant tissues with high precision [7, 44].

3.1.2 PET Imaging Basics

PET imaging is a nuclear medicine technique that visualizes the metabolic processes in the body by detecting gamma rays emitted by a radiotracer injected into the patient. This modality is highly effective in oncology for detecting and characterizing tumors, staging cancer, and evaluating treatment response. The functional insights provided by PET are invaluable for tumor segmentation, as they highlight areas of abnormal metabolic activity [7, 39].

3.1.3 Challenges in Tumor Segmentation

Tumor segmentation in PET images presents several challenges, including the inherent noise and low contrast of PET scans, the heterogeneity of tumor tissues, and the partial volume effect, which causes blurring at the edges of structures. Additionally, the variability in tumor shapes, sizes, and locations adds complexity to the segmentation process, necessitating advanced techniques to achieve reliable results [7, 44].

3.1.4 Traditional Segmentation Techniques

Traditional methods for tumor segmentation in PET images include thresholding, region growth, and clustering algorithms. Thresholding involves setting a specific intensity value to differentiate between tumor and non-tumor regions, while region growth expands from an initial seed point based on predefined criteria. Clustering algorithms, such as k-means, group similar pixels together. However, these methods often struggle with noise sensitivity and the need for user-defined parameters, limiting their effectiveness [7].

3.1.5 Modern Segmentation Techniques

Advancements in machine learning, particularly deep learning, have led to the development of more sophisticated tumor segmentation techniques. CNNs and their variants, such as U-Net and fully convolutional networks (FCNs), have demonstrated remarkable

success in medical image segmentation. These models leverage hierarchical feature learning to capture complex patterns in PET images, significantly improving segmentation accuracy and robustness [15, 39, 44].

3.1.6 Recent Advances and Research

Recent studies have explored various deep-learning architectures and strategies to enhance tumor segmentation in PET images. For instance, the U-Net architecture has become a standard for medical image segmentation due to its encoder-decoder structure that captures both local and global features. Research has also focused on integrating PET with other imaging modalities, such as CT and MRI, to provide complementary anatomical and functional information, further enhancing segmentation performance [39, 44, 11, 30].

3.1.7 Evaluation Metrics

Evaluating the performance of tumor segmentation algorithms involves metrics such as the Dice Coefficient (DC), Jaccard Index, Precision, Recall, and the Area Under the ROC Curve (AUC). These metrics assess the accuracy and robustness of the segmentation results, providing insights into the model's effectiveness in clinical applications [13, 39].

The Dice coefficient is defined as:

$$\text{Dice} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FN} + \text{FP}},$$

where true positive (TP) refers to the number of pixels that are correctly classified as positive. False negative (FN) is the number of positive pixels that are incorrectly classified as negative. False positive (FP) indicates the number of negative pixels that are incorrectly classified as positive. The Dice coefficient ranges from 0 to 1, where 1 indicates perfect agreement between the prediction and the ground truth.

3.1.8 Clinical Applications and Impact

Accurate tumor segmentation in PET images has significant clinical implications. It aids in precise tumor delineation for radiotherapy, guides biopsy procedures, and tracks tumor response to treatment. Improved segmentation techniques improve treatment planning and patient outcomes by providing reliable and reproducible results [44, 11].

3.1.9 2D U-Net for Efficient Medical Image Segmentation

A 2D U-Net was employed to process individual 2D slices extracted from the 3D medical imaging data. This choice was made to balance computational efficiency with the ability to train a robust segmentation model. While a 3D U-Net could have been used to leverage the full volumetric context, the 2D approach was deemed sufficient for this study, given the nature of the dataset and the computational constraints. The U-Net architecture, known for its symmetric contracting and expanding paths, effectively captures spatial features and enables precise segmentation of medical images.

3.1.10 Future Directions

Future research in tumor segmentation for PET images is likely to focus on enhancing model robustness, integrating multimodal data, and leveraging artificial intelligence for personalized medicine. The development of more generalized models and the application of big data analytics will further advance the field, leading to more accurate and efficient clinical practices [39, 30].

3.2 Recent Advances and Results in Augmentation for Tumor Segmentation

In this section, we focus on the latest techniques and their impact on enhancing tumor segmentation performance. This section can be divided into several key areas: the types of data augmentation techniques, specific advancements in augmentation methods, research studies demonstrating the effectiveness of these techniques, and the overall impact on tumor segmentation.

3.2.1 Introduction to Data Augmentation

Data augmentation is a critical technique in deep learning, particularly in medical imaging, to enhance the diversity of training datasets and improve the robustness and generalization of models. By artificially increasing the size and variability of training data, augmentation helps in mitigating overfitting and improving model performance, especially in scenarios where labeled data is limited [46, 51].

3.2.2 Data Augmentation Techniques

Data augmentation techniques are essential for training robust machine learning models, especially in fields like computer vision and medical imaging, where dataset size and diversity may be limited. These techniques involve creating modified versions of existing data to increase its size and variability, enhancing the model's generalization capability. By applying transformations such as rotations, translations, scaling, flipping, adding noise, and intensity adjustments to the original data, models become more robust and less prone to overfitting. Below, we describe major types of data augmentation techniques, with practical examples illustrating their applications in image processing and medical imaging scenarios ([15, 46]):

i) Geometric Transformations: Geometric transformations adjust the spatial properties of images, allowing the model to learn from different orientations and perspectives.

- *Rotation and Flipping:* Applying random rotations and flips to images helps the model learn invariant features regardless of orientation. For instance, suppose that we have an image of a tumor scan. Rotating the image by $+30$ degrees or -30 degrees can simulate different orientations of the tumor. This helps the model learn that the tumor is the same regardless of its orientation in the image. In this case, we can consider the following:

  - Original Image: No rotation (0 degrees).

- Augmented Image 1: Rotate by $+30$ degrees.
- Augmented Image 2: Rotate by -30 degrees.

- *Flipping:* Horizontal or vertical flips simulate different viewing angles.

- Original Image: No flip.
- Augmented Image 1: Flip horizontally (left to right).

Additionally, flipping the image vertically (top to bottom) can help the model recognize features regardless of whether the image is upside down. We can also combine rotation and flipping; for example, rotate the image by $+45$ degrees and then flip it horizontally or vertically. These transformations enhance the model's robustness to image orientation and reflectance variations, improving its generalization to diverse real-world scenarios.

- *Scaling and Cropping:* Randomly scaling images and cropping them to a fixed size introduces variability in size and perspective, aiding the model in recognizing tumors of different scales. Consider the following example with a PET scan image of a tumor. We can scale the image by $1.2\times$ (120% of the original size) to simulate a larger tumor, or by $0.8\times$ (80% of the original size) to simulate a smaller tumor:

- Original Image: No scaling ($1.0\times$).
- Augmented Image 1: Scale by $1.2\times$ (enlarged).
- Augmented Image 2: Scale by $0.8\times$ (reduced).

After scaling an image by $1.2\times$, one can crop a 256×256 pixel section from the center to focus on the tumor region:

- Original Image: No cropping.
- Augmented Image 1: Scale by $1.2\times$, then crop a 256×256 region from the center.

Another option is to crop a 128×128 pixel section from a random location in the image to introduce variability in composition. Combined scaling and cropping can also be effective; for example, scaling an image by $1.4\times$ and then cropping a 224×224 pixel section from the top-left corner simulates a closer view from a specific angle, or scaling by $0.9\times$ (slightly reduced) and cropping a 200×200 pixel section from the bottom-right corner introduces variability in perspective and size. These augmentations help the model learn to recognize tumors at different scales and perspectives, enhancing its ability to generalize to varying tumor sizes and positions in real-world medical images.

ii) Color Space and Intensity Transformations: Modifying brightness, contrast, saturation, and hue allows models to become resilient to lighting and contrast variations, common in real-world imaging conditions.

- *Brightness and Contrast Adjustments:* Varying the brightness and contrast of images helps the model become resilient to fluctuations in imaging conditions. For example, we can increase the brightness of a PET scan image by $+30\%$ to simulate a scenario where

the imaging conditions are brighter than usual. Conversely, decreasing the brightness by -20% mimics an image captured under dimmer lighting conditions. Additionally, applying a random brightness adjustment within a range of -15% to $+15\%$ introduces variability in lighting conditions. Regarding contrast, we can enhance the contrast of an image by $+40\%$ to accentuate the differences between the tumor and surrounding tissue or reduce the contrast by -25% to simulate a more uniform appearance, making it more challenging to distinguish between tumor and non-tumor areas. A combined approach might involve increasing brightness by $+20\%$ and contrast by $+30\%$ to simulate a high-exposure scenario where the image is both brighter and has greater contrast.

iii) Noise Injection: Adding noise simulates real-world imaging imperfections, making the model more robust.

- *Gaussian Noise:* Adding Gaussian or other types of noise simulates real-world scenarios where images may be noisy. For instance, Gaussian noise with a mean of 0 and a standard deviation of $\sigma = 0.01$ can be introduced to represent slight sensor noise that might occur during imaging. To simulate moderate noise that could arise from poor imaging conditions, we can add Gaussian noise with a mean of 0 and a standard deviation of $\sigma = 0.05$. Adding Gaussian noise with a mean of 0 and a standard deviation of $\sigma = 0.1$ generates a very noisy image, representing strong noise encountered in real-world scenarios. See Figure 3.

To make this figure we applied the following code:

```
slight_noise = random_noise(original_image, mode='gaussian', mean
    =0, var=0.01**2)
moderate_noise = random_noise(original_image, mode='gaussian', mean
    =0, var=0.05**2)
strong_noise = random_noise(original_image, mode='gaussian', mean
    =0, var=0.1**2)
```

Adding salt-and-pepper noise with a noise level of 1% (1% of pixels randomly turn black or white) simulates minor random noise often seen in digital images, or adding salt-and-pepper noise with a noise level of 5% simulates more noticeable noise, where 5% of the pixels are randomly altered.

Adding speckle noise with a variance of 0.04 to simulate granular noise is seen in medical imaging. Also, adding speckle noise with a variance of 0.1 for a higher level of noise, makes the image appear rougher and more challenging for the model. As above, combining could be useful. For example, noise with a mean value of 0 and $\sigma = 0.02$ with salt-and-pepper noise at a 1% noise level simulates a combination of random and Gaussian noise effects. Or, adding speckle noise with a variance of 0.05 and Gaussian noise with mean 0 and $\sigma = 0.03$ simulates a mixture of speckle and Gaussian noise. These numeric examples show how different types of noise can be added to images to simulate real-world scenarios. The addition of such noise helps the model become more robust to imperfections and variations that might occur in actual medical imaging conditions.

iv) Affine and Elastic Transformations: These transformations introduce linear and non-linear shape variations to enhance the model's ability to adapt to different per-

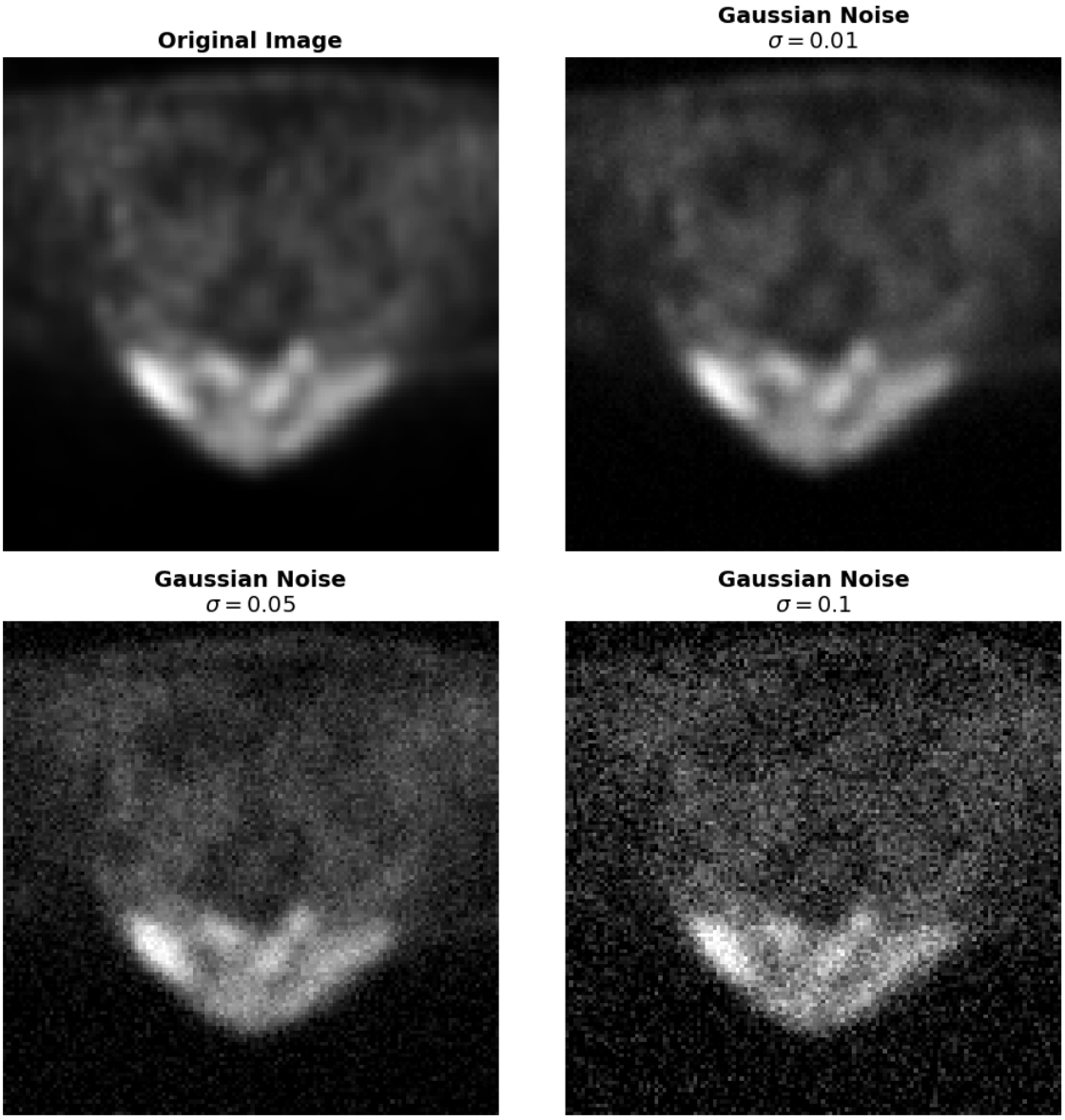


Figure 3: Original and augmented PET images from the head and neck cancer dataset with Gaussian noise. The image in the top left corner is the original PET scan from the dataset, while the following images illustrate the effect of adding Gaussian noise at various levels. Slight noise with a standard deviation σ of 0.01 simulates minor sensor noise. Moderate noise with $\sigma = 0.05$ represents conditions of reduced imaging quality. Strong noise with $\sigma = 0.1$ introduces significant distortion, simulating challenging imaging scenarios. These augmentations are employed to improve model robustness in head and neck cancer segmentation tasks.

spectives and anatomical variations.

- *Affine Transformations:* Shearing, stretching, and skewing change the image geometry, improving viewpoint adaptability.

- *Elastic Deformations:* Mimic natural anatomical variability, helping the model learn from diverse shapes. The two parameters α and σ are important in elastic deformations. The upper and lower bounds for the parameters α and σ are not strictly defined and can vary depending on the specific application, image resolution, and desired intensity of the deformation. However, typical ranges can be provided based on common usage in image augmentation, particularly in medical imaging. The common range for α is 5 to 30 and for Sigma is 1 to 5. The parameter α controls the intensity of the deformation (how far pixels are moved), while the parameter σ controls the smoothness of the deformation (how smooth or coarse the displacement field is).

We now consider the following examples:

Example 1. (Mild Elastic Deformation) Considering $\alpha = 5$ and $\sigma = 1.5$ applies a mild deformation where the displacement of pixels is relatively small and smoothly distributed across the image. It mimics slight variations in anatomy that might occur naturally. In the original image, we have no deformation, and in the augmented image, we have elastic deformation with $\alpha = 5$ and $\sigma = 1.5$.

Example 2. (Moderate Elastic Deformation) If we consider $\alpha = 15$ and $\sigma = 3$, then it creates a moderate deformation, where the pixels are displaced more noticeably, introducing variability that represents moderate anatomical differences.

Example 3. (Strong Elastic Deformation) Let $\alpha = 30$ and $\sigma = 5$. Then it applies a strong deformation, significantly displacing pixels and creating more pronounced variability. It is useful for simulating large anatomical differences between patients.

Here, also, we can combine elastic deformations with other augmentations. For example, we can combine moderate elastic deformation with scaling or mild elastic deformation with rotation. These examples illustrate how elastic deformations can be applied to medical images to introduce variability that mimics natural anatomical differences. By training on such deformed images, the model can generalize better across different patient anatomies, making it more robust in clinical applications.

Synthetic data generation techniques help expand training datasets by creating realistic artificial images. Generative Adversarial Networks (GANs) are employed to generate realistic synthetic medical images, augmenting the training data. These synthetic images can include rare or hard-to-obtain cases, improving model training diversity. GANs consist of two networks: a Generator and a Discriminator. The generator produces synthetic images, while the discriminator evaluates whether an image is real or generated. Over time, the generator learns to create increasingly realistic images, enhancing the training set. Here, is an example:

Example 4. (Generating Synthetic Tumor Images) Consider $1,000$ PET scan images with tumors. The GAN configuration is as follows:

- Generator Latent Space Dimension (z): 100;

- Training Epochs: $10,000$;
- Batch Size: 64;
- Learning Rate: 0.0002;
- Generated Images: $5,000$ synthetic PET scan images of tumors.

Effect: By training a GAN with these parameters, we can generate $5,000$ in synthetic tumor images, effectively increasing the diversity of your training set. These images can include rare tumor shapes or sizes that might not be present in the original dataset.

Simulated data involves creating synthetic images using models that mimic the physical and biological properties of tissues. This can include simulations of PET, MRI, or CT scans that replicate the interaction of tissues with imaging modalities.

Example 5. (Simulating PET Scan Images of Tumors) This example aims to generate synthetic PET scan images representing tumors with varying levels of metabolic activity. The simulation parameters are:

- Tumor Size Range: 1 cm^3 to 5 cm^3;
- Standardized Uptake Value (SUV): 2.0 to 12.0;
- Background Tissue SUV: 0.8 to 1.5;
- Noise Level: Gaussian noise with $\sigma = 0.05$;
- Number of Images: $1,000$ synthetic PET scan images.

Effect: This simulation creates PET images with tumors of various sizes and metabolic activities, helping the model learn to detect tumors under different conditions.

3.2.3 Recent Advances in Data Augmentation

In this section, we list some recent advances in data augmentation. We start by presenting automated data augmentation.

AutoAugment and RandAugment techniques automate the search for the best augmentation strategies through reinforcement learning. AutoAugment has been specifically adapted to medical imaging to identify optimal augmentation strategies that improve segmentation performance. A large space of possible augmentations and combinations is searched to find the best strategy to improve the performance of the model [5].

RandAugment simplifies the search process by randomly selecting a fixed number of augmentation operations and applying them with random magnitudes. This method does not require a reinforcement learning agent but still introduces significant variability into the training data [4].

Next, we introduce context-aware augmentation, considering the spatial context within medical images to ensure augmented regions make anatomical sense [19].

The next advancement involves domain adaptation techniques. These techniques involve using data from different domains or modalities to supplement the training data. For example, CT images can be used to supplement PET scans, or vice versa, using

techniques such as CycleGANs. CycleGANs can translate images from one domain to another while preserving the underlying structure [49].

Finally, adversarial training includes negative examples (slightly disturbed images that challenge the model) in the training set. This makes the model more robust to subtle variations and potential external attacks [16].

3.2.4 Results and Impact on Tumor Segmentation

Recent studies have demonstrated significant improvements in tumor segmentation performance through advanced augmentation techniques:

1. Improved Accuracy and Robustness: Various augmentation techniques have been demonstrated to enhance model performance by increasing accuracy and robustness.

Wang and Perez [51] demonstrated that the application of elastic deformations and intensity variations significantly improved the Dice coefficient in brain tumor segmentation tasks. Similarly, Zhu et al. (2020) [56] used GAN-based augmentation to generate synthetic PET images, resulting in improved segmentation accuracy for rare tumor types.

2. Enhanced Generalization: Data augmentation techniques help models generalize better across diverse datasets, reducing reliance on large annotated datasets. Research by Shorten and Khoshgoftaar (2019) [46] demonstrated that models trained with comprehensive augmentation policies exhibit improved generalization across different datasets, thereby minimizing the need for extensive annotated data.

3. Reduction in Overfitting: Automated data augmentation techniques play a key role in reducing overfitting, and improving model performance on unseen data.

Studies have demonstrated that automated augmentation methods, such as AutoAugment, help mitigate overfitting, resulting in better performance on unseen test data. This is particularly important in medical imaging, where data variability is high and collecting large datasets is challenging [47, 22, 55].

The integration of advanced data augmentation techniques in tumor segmentation tasks has shown promising results in improving model performance, robustness, and generalization. As research progresses, the development of more sophisticated and context-aware augmentation methods will continue to enhance the capabilities of deep learning models in medical imaging, ultimately leading to better clinical outcomes and more reliable diagnostic tools.

4 Methods and Materials

4.1 CNN Architecture and Design

Designing an optimal CNN for a specific problem is a complex task that requires extensive expertise and experimentation. Developing various CNN architectures for different computer vision tasks is also time-consuming. Therefore, finding an appropriate CNN model with minimal resources and human intervention is crucial. Several CNN models, such as LeNet, VGGNet, AlexNet, GoogleLeNet, ResNet, and others, have been manually developed with increasing architectural depth and numerous parameters to solve diverse image classification tasks. These models rely on domain-specific knowledge and trial-and-error to select suitable architectures and parameters, making the process labor-intensive and time-consuming. Given the impact of CNN architecture size and parameters on performance and complexity, the challenge lies in efficiently identifying an optimal model from a vast design space without extensive human input [18].

Another issue is the allocation of significant resources toward finding a network that can generalize to datasets beyond the training set. In many cases, the trained network performs well on the specific dataset used for training. There is no guarantee that the constructed network will achieve satisfactory classification accuracy for other datasets due to limited computational resources, making it impractical to design different networks for several datasets. Therefore, it is important to find a suitable model that assists researchers who are not an expert in deep learning (DL) in creating a DL model for a specific problem while using minimal resources [31, 36].

4.2 Implementation in Python

The project utilized Python (version 3.12.0) for the computational aspects, as it is an adaptable programming language suitable for data analysis and algorithm development. Python was selected for its extensive libraries, active community support, and ease of integration with various data processing tools.

4.2.1 Libraries and Tools

This section focuses on the specific libraries and their functionalities without delving into broader development details. Each library serves a specific purpose in the pipeline:

NumPy: Used for efficient numerical computations, array manipulations, and supporting mathematical operations, a cornerstone of scientific computing [40].

Matplotlib: Essential for visualizing results, such as displaying medical images, segmentation masks, and performance metrics [21].

SciPy: Provides advanced mathematical and statistical functions, including image processing utilities [50].

OpenCV (cv2): Handles essential image preprocessing tasks, such as resizing, cropping, and pixel-value manipulation, commonly needed in medical imaging [3].

Nibabel (nib): Specialized in neuroimaging data formats like NIfTI and MINC, enabling the handling of volumetric data in neuroimaging research [2].

TensorFlow (tf): A robust machine learning framework utilized for designing, training, and evaluating deep learning models, including CNNs and U-Nets for segmentation tasks [1].

These libraries, collectively, form the technical foundation for processing and analyzing medical imaging datasets.

4.2.2 Code Structure

The provided code implements a pipeline for loading medical image data, processing it, training a U-Net model for segmentation, and evaluating the performance using the Dice coefficient. The pipeline consists of several well-defined steps, which are structured as follows:
1. Data Loading and Preprocessing: Patient data, including PET scans and segmentation masks, is loaded from specified file paths. Images are resized, thresholded, and normalized for use in model training. The images and masks are stored in lists (`posList` and `maskList`) and are subsequently split into training and test sets. Resized images are then converted to integers for consistency and efficient storage [28].

2. Dataset Splitting: The dataset is divided into training and test sets using predefined patient groups (`testSetPatients`). Data splitting is based on the number of slices per patient, ensuring that slices from specific patients are allocated for testing while others are used for training. For each patient, slices are resized and added to the corresponding list for training or testing, along with their associated segmentation masks [45].

3. Model Definition: A U-Net model is defined for segmentation tasks, as this architecture is widely used in medical image segmentation due to its symmetric contraction and expansion paths. The model incorporates several convolutional layers with ReLU activations, dropout for regularization, and max-pooling layers to down-sample the input. The expansive path includes transposed convolution layers to up-sample the feature maps and merge layers that combine the up-sampled data with higher-resolution feature maps from earlier layers. The final output layer generates a segmentation mask using a linear activation function [37].

4. Model Training: The training process incorporates data augmentation through horizontal flips to enhance the diversity of the training set. The model is compiled using the Adam optimizer and binary cross-entropy loss. An early stopping mechanism is implemented to prevent overfitting by monitoring the validation loss and restoring the best model weights. The model is trained for a predefined number of epochs (`numEpochs`), with its performance validated using a validation split [46].

5. Prediction and Evaluation: After training, the model is employed to predict segmentation masks for both the training and test sets. The predictions are saved as text files for further analysis. Dice coefficients are calculated for each test sample to assess

segmentation performance at a specific threshold (`limit`). Mean and median Dice coefficients are computed and tracked across multiple iterations to ensure reliable results [8].

6. Cross-Validation and Iterations: The process is repeated for multiple test sets, with results accumulated across several iterations (defined by `numIter`). In each iteration, the model is trained from scratch with a different test set, ensuring that each test patient is evaluated in different iterations. This cross-validation approach ensures robust segmentation performance, preventing over-reliance on specific subsets of the data [26].

7. Results Reporting: After all iterations, each test patient's mean and median Dice coefficients are calculated, and the results are printed. A detailed analysis of the segmentation performance is provided by calculating Dice scores across various threshold values, assisting in assessing the model's accuracy in capturing the region of interest. The results can be further processed or visualized for reporting purposes. The term threshold value refers to a specific point or limit at which a significant change occurs or triggers an action. Its meaning can vary depending on the context, but generally, it is used to indicate the boundary between two different states or behaviors. In statistics and data analysis, the threshold value may refer to a cut-off point in a dataset, used to separate meaningful data from noise or irrelevant information [6].

8. Code Modularization: The code is well-structured with modular functions like `prepareImage()`, `trainModel()`, and `evaluateDice()`, each focusing on specific tasks such as data preparation, model training, and performance evaluation. This modular approach improves code readability, maintainability, and ease of modification. Critical parameters such as `img_rows`, `img_cols`, `batchSize`, `numEpochs`, and `limit` are defined in the global scope, making it easy to adjust key hyperparameters and adapt the pipeline for different use cases [32].

In conclusion, the code includes several key functions that facilitate the image processing and model training workflow. The `prepareImage()` function is responsible for preparing individual images by resizing, normalizing, and applying thresholding techniques to enhance image quality for analysis. Following this, the `trainModel()` function compiles and trains the U-Net model, implementing early stopping and a validation split to optimize performance and prevent overfitting. To assess the effectiveness of the segmentation, the `evaluateDice()` function computes the Dice coefficient, providing a quantitative measure of the model's performance. The `splitData()` function plays a crucial role in organizing the dataset, as it splits the data into training and testing subsets based on predefined patient groups, ensuring a balanced representation during training. Finally, the `saveResults()` function saves the predicted segmentation masks to files for future reference, enabling easy access and analysis of the model's outputs. Together, these functions create a robust pipeline for processing medical images and training CNNs effectively.

This structured approach to model development ensures that the code is efficient, scalable, and adaptable for medical image segmentation tasks. The use of U-Net, combined with cross-validation and rigorous evaluation, enhances the reliability and accuracy of the model's segmentation capabilities.

4.2.3 Development Environment

This section covers the overarching environment, including hardware and optional tools.

The development environment ensures that the implementation pipeline, including training and evaluating a U-Net model for medical image segmentation, runs efficiently and reliably.

Programming Language: Python was chosen for its extensive library ecosystem and simplicity, making it ideal for machine learning and deep learning projects.

Python Version: The recommended version is Python 3.12.0 to ensure compatibility with the libraries and tools used.

Key Libraries: The project relies on several Python libraries for its execution, including TensorFlow, Keras, NumPy, Matplotlib, and OpenCV. These libraries enable tasks ranging from model development to data visualization and preprocessing. (See Section 4.2.1 for details.)

Hardware Requirements:

- CPU/GPU: A compute unified device architecture (CUDA)-compatible graphics processing unit (GPU) from NVIDIA with at least 8 gigabytes (GB) of video random access memory (VRAM) is highly recommended for faster training times. The code can also run on central processing units (CPUs), albeit with increased training duration.
- Memory: A minimum of 16GB of random access memory (RAM) is advised for processing large datasets effectively.
- Storage: At least 50GB of free space is required to store and manage medical imaging datasets, particularly for 3D scans such as MRI or CT.

Optional Tools:

- Jupyter Notebook: Useful for interactive development and visualization, allowing real-time testing of components like preprocessing and training.
- Integrated Development Environments (IDEs): Tools like PyCharm, VSCode, or Spyder enhance productivity with features like debugging and code suggestions.

Operating System: The code is compatible with Linux (preferred for GPU support), Windows, and macOS.

By adhering to these guidelines, the code can be executed efficiently, enabling robust training and evaluation of the U-Net model for medical image segmentation.

4.2.4 Execution and Performance

First, we will discuss the execution process.

The medical image segmentation pipeline follows a structured sequence of steps to ensure efficient training, evaluation, and inference using a U-Net architecture. The key

execution phases are as follows:

Data Loading and Preprocessing: The dataset is initially loaded from the specified directory, with the assumption that it is organized into folders by patient ID, each containing images and their corresponding masks. The images undergo preprocessing, which includes resizing them to the target input size (e.g., 256×256) and normalizing their pixel values to a specific range, typically between 0 and 1, to enhance convergence during training. Additionally, the masks, which serve as ground-truth labels, are binarized if necessary to streamline the segmentation process.

Model Training: The U-Net model is compiled using an appropriate optimizer, such as Adam, and a relevant loss function, like binary cross-entropy or dice coefficient loss, depending on the medical segmentation task. The model is then trained on the preprocessed dataset, where each image and its corresponding mask are input into the network. The training process uses the `fit()` function, with key hyperparameters like the number of epochs, batch size, and validation split specified. Real-time data augmentation is applied to enhance generalization by introducing variations in the training data, such as rotations and flips. Throughout the training, metrics like loss, accuracy, and dice score are tracked for both the training and validation sets, enabling real-time performance monitoring.

Inference: For a new, unseen image, the trained model performs inference to predict the segmentation mask. The input image undergoes the same preprocessing steps as during training. The predicted mask is then post-processed, resized to match the original dimensions of the input image, and binarized if necessary to provide a clear delineation of the segmented regions. The results are visualized using tools such as Matplotlib or saved as images for further analysis.

We will continue with performance considerations.

Training Time: Performance considerations begin with training time, which is influenced by factors such as the size of the dataset, the complexity of the U-Net model, and the type of computational hardware used (GPU vs. CPU). On a typical GPU setup, training a U-Net model with a moderately sized dataset, such as thousands of 2D images, can take several hours. For 3D data or larger datasets, training time increases significantly, making the use of a GPU essential. To speed up training, it is recommended to use a GPU-accelerated environment, such as NVIDIA CUDA with TensorFlow-GPU, as this can drastically reduce training times compared to CPU-based environments.

Memory Usage: U-Net models, particularly those used for 3D data or high-resolution 2D images, demand significant memory resources. For 2D images, 16GB of RAM and a GPU with at least 8GB of VRAM are typically sufficient, but 3D models or higher-resolution images may require more memory. In cases of memory shortages, strategies like reducing the batch size, downsampling the images, or using gradient accumulation can help mitigate memory limitations.

Model Evaluation and Metrics: After training, the model is evaluated on a test

dataset that was not used during training to assess its generalizability. Key evaluation metrics include the Dice coefficient and Intersection over Union (IoU). The Dice coefficient measures the overlap between the predicted segmentation and the ground truth, while IoU evaluates the segmentation accuracy by calculating the ratio of the intersection of predicted and actual areas to their union. These metrics are crucial for assessing the quality of segmentation, especially in cases of class imbalance. While accuracy can provide general insights, it is less informative in segmentation tasks, where Dice and IoU scores are preferred. Evaluation results, including these metrics, are printed and/or logged, and visual comparisons between the predicted masks and the true labels provide qualitative insights into the model's performance.

Model Performance: The model's performance is evaluated on both training and validation datasets during the training process. If signs of overfitting are observed, techniques such as early stopping or reducing learning rates can be employed. The U-Net architecture generally achieves high accuracy in medical image segmentation tasks, assuming the dataset is well-prepared and the model is appropriately tuned. Its effectiveness in detecting small or complex regions is enhanced by its use of skip connections. Additionally, visualizing the predicted segmentation masks during and after training facilitates qualitative assessments, aiding in the identification of potential improvements for the model.

Scalability and Optimization: For larger datasets or higher-resolution images, it may be necessary to scale the system by utilizing distributed training techniques or employing multiple GPUs. Additionally, hyperparameter tuning—such as experimenting with learning rates, batch sizes, and different optimizer configurations—can help further optimize the model's performance.

We will continue this section by presenting two challenges and considerations. First, training duration can be prohibitively slow when using a CPU, making it essential to ensure that the code runs in a GPU or TPU environment for improved performance. Second, overfitting is a concern due to the small sizes of some medical imaging datasets. To mitigate this risk and enhance generalization, techniques such as data augmentation, dropout layers, and early stopping are implemented.

By following the execution process and considering these performance aspects, the U-Net model can efficiently segment medical images, achieving high accuracy and reliability in tasks such as organ or tumor delineation.

4.3 Data Augmentation Techniques Employed

In the preprocessing phase of the dataset, data augmentation is applied to enhance the robustness and generalization capability of the model. The primary augmentation technique used in the code is horizontal flipping:

Horizontal Flipping: Each image and its corresponding mask are flipped horizontally to create a mirror image. This technique is applied to augment the training dataset by introducing variability, which helps the model to learn more generalized features and reduces overfitting.

```
flipped_img = cv2.flip(x_train[i], 1)  # Flip the image
    horizontally
x_train_augmented.append(flipped_img) # Append the horizontally
    flipped image to the augmented images list
flipped_mask = cv2.flip(y_train[i], 1) # Flip the mask horizontally
y_train_augmented.append(flipped_mask) # Append the horizontally
    flipped mask to the augmented masks list
```

Rotation: The imgaug library is used to rotate images by a random angle within the range of -15 to 15 degrees. This introduces variability in the orientation of the images, helping the model generalize better to different orientations.

```
seq = iaa.Sequential([iaa.Affine(rotate=(-15, 15))]) # Rotation of
    k degrees where k is a random number chosen from the interval
    (-15, 15)
x_train_augmented = seq(images=x_train)
```

Clockwise Rotation: Additional data augmentation involves rotating images by 90 degrees clockwise. This technique is effective in introducing further variability and helping the model learn from different perspectives of the data.

```
clockwise_90_img = np.rot90(x_train[i], k=3)  # Rotate image 90
    degrees clockwise
x_train_augmented.append(clockwise_90_img)  # Append the rotated
    image to the augmented images list
clockwise_90_mask = np.rot90(y_train[i], k=3)  # Rotate mask 90
    degrees clockwise
y_train_augmented.append(clockwise_90_mask)  # Append the rotated
    mask to the augmented masks list
```

Concatenation of Original and Augmented Data: After generating augmented data, the original training images and masks are combined with the augmented ones. This approach increases the overall size of the training set, providing more examples for the model to learn from, thereby enhancing its robustness.

```
  # Combine original and augmented training data
x_train = np.concatenate([x_train, x_train_augmented]) #
    Concatenate original and augmented images
y_train = np.concatenate([y_train, y_train_augmented]) #
    Concatenate original and augmented masks
```

These techniques are essential for improving model performance by ensuring it can generalize well across different variations of the input data.

4.4 Dataset Description and Selection

This section is based on the dataset selection methodology proposed by Liedes et al. [24], who outlined a comprehensive approach for selecting and collecting clinical data for head and neck cancer studies. Moreover, the data for this study was collected retrospectively from a total of 100 head and neck patients. Among them, 89 had been diagnosed with HNSCC, while the remaining 11 had been diagnosed with other head and neck cancer diseases, such as adenocystic carcinoma of the oral cavity, chondrosarcoma of the neck, and papillary or follicular carcinoma of the thyroid gland. The data represents treatment

response assessments conducted during PET/MRI examinations at Turku PET Centre in Turku, Finland. These assessments were performed between 2014 and 2022, during the patients' treatment at Turku University Hospital. Tumors in the scans were confirmed through histopathological sampling or follow-up imaging. Patients had an average age of 62 years with a standard deviation of 12 years, and a male-to-female sex ratio of 2.1. All patients were at least 18 years old, and they provided explicit consent for the research use of their data. Moreover, the research received approval from the Ethics Committee of the Hospital District of Southwest Finland.

Patients underwent imaging using either a 3T Philips Ingenuity TF PET/MRI scanner (Philips Health Care) or a SIGNA PET/MRI scanner with QuantWorks (GE Healthcare), with ^{18}F-fluorodeoxyglucose as the tracer substance. Although the MRI and PET were performed simultaneously, only the PET images were utilized in this study for the CNNs. All PET images showed the head and neck area of the patients and each had between 32 and 66 transaxial slices, each 512×512 pixels in size. A medical doctor, with the help of an experienced nuclear medicine physician, created binary masks for the PET images by labeling the voxels with cancer as positive and the rest as negative. Both the PET image and these masks were stored in NIfTI format files.

In this study, we are focusing on segmenting two-dimensional images. Each transaxial slice of a three-dimensional PET image is treated as an individual image. Before processing the slices, they were converted to a size of 128×128 pixels and the pixel values were scaled to fit within the range of 0 to 1 for each slice separately. The binary masks were also resized to match the size of the image slices. In the resized mask, a pixel was labeled as positive if at least 5 out of the corresponding 16 pixels in the original 512×512 mask were positive. Upon scaling, slices with fewer than 6 positive pixels were removed, resulting in a total of 962 slices from 89 patients (79 HNSCC patients and 10 with other head and neck cancer types). The slices were divided into five sets, each containing $191 - 194$ slices or approximately $19.9 - 20.1\%$ of the total data. The CNNs were trained using five-fold cross-validation based on these sets. Since the division into sets was done patient-wise and the CNNs are always initialized before training with new data, the CNNs do not have any information about the patients of the test set that could affect the predictions.

4.5 Evaluation Metrics

The effectiveness of the U-Net model for segmentation tasks is assessed using the following evaluation metrics:

1. Dice Coefficient: The Dice coefficient, also known as the Dice similarity coefficient (DSC) (see the following section), is the primary metric for evaluating segmentation performance. It measures the overlap between the predicted segmentation and the ground truth mask. In the provided code, Dice coefficients are calculated for different threshold limits to evaluate the performance of the model. The function `dicesForLimit` computes the Dice coefficients by comparing the predicted segmentations with the ground truth masks:

```
def dicesForLimit(predictions, y_test, limit):
    dices = np.zeros((len(y_test)))
    img_height = predictions[0].shape[0]

    for k in range(len(y_test)):
        TP = 0
        FN = 0
        FP = 0
        for i in range(img_height):
            for j in range(img_height):
                if y_test[k, i, j] >= 0.5 and predictions[k, i, j]
                    >= limit:
                    TP += 1
                if y_test[k, i, j] >= 0.5 and predictions[k, i, j]
                    < limit:
                    FN += 1
                if y_test[k, i, j] < 0.5 and predictions[k, i, j]
                    >= limit:
                    FP += 1
        if TP + FN + FP > 0:
            dices[k] = 2 * TP / (2 * TP + FN + FP)
    return dices
```

The Dice coefficient offers several advantages compared to other similarity metrics. It is particularly beneficial for imbalanced datasets, where one set may be much larger than the other. It is a better option for image segmentation tasks because it is more sensitive to overlap between the predicted and ground truth masks. This sensitivity is achieved by treating the segmentation masks as sets of pixels. Both the predicted segmentation and the ground truth segmentation are represented as binary masks, where a pixel is either part of the segmented object or not. A high Dice coefficient indicates a strong similarity between predicted and ground truth masks, signifying effective segmentation. Conversely, a low Dice coefficient suggests poor segmentation performance.

2. Mean and Median Dice Coefficient: The Dice coefficients for all test samples are aggregated to compute the mean and median values. These summary statistics provide insights into the overall performance of the model across different test sets. The metrics are calculated as follows:

```
mean_dice = np.mean(dices)
median_dice = np.median(dices)
```

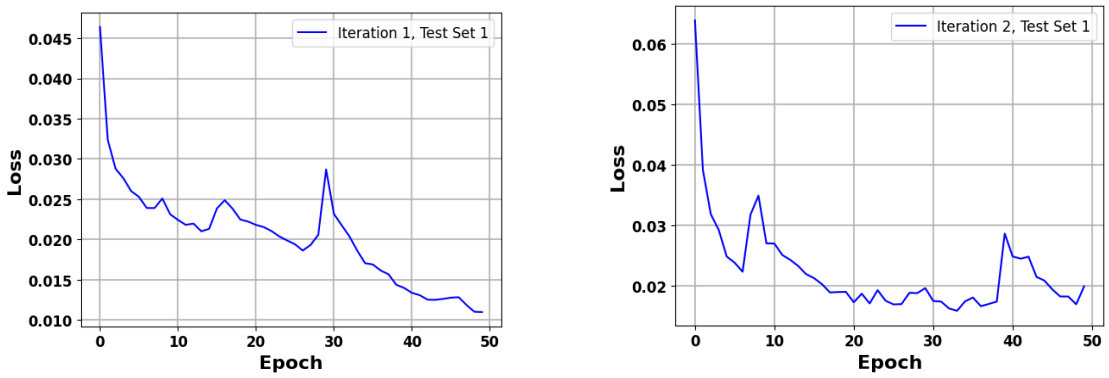
Mean Dice represents the average Dice coefficient across all test samples, giving a general sense of the model's performance while median Dice provides the middle value of the Dice coefficients, which can be more robust to outliers compared to the mean.

3. Visualization of Training Loss: During training, the loss values are tracked and plotted to monitor the model's learning progress, see Figure 4. This helps in identifying potential issues such as overfitting or convergence problems. The training loss plot is displayed using:

```
plt.plot(history.history['loss'], label=f"Iteration {u + 1}, Test
    Set {d + 1}", color='blue')
```

```
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show()
```

These metrics collectively provide a comprehensive evaluation of the model's segmentation performance, allowing for the assessment of both individual predictions and overall model accuracy.



(a) (b)

Figure 4: (a): The plot of loss function per epoch when `img_height=64`, `limit=0.25`, `pixLimit=5`, and `numEpochs=50` (Test set 1, Iteration 1) (b): The plot of loss function per epoch `img_height=64`, `limit=0.25`, `pixLimit=5`, and `numEpochs=50` (Test set 1, Iteration 2).

5 Results

This chapter presents and analyzes the data collected during our research.

5.1 Quantitative Results

5.1.1 Overview

In this research, we provide a detailed analysis of the image processing results from 89 patients diagnosed with a positive condition. The analysis focused on identifying and evaluating significant image slices from PET images using mask data to isolate regions of interest (ROIs). The ROI refers to a specific area or portion of an image that is singled out for analysis or processing. In medical imaging, such as PET scans, ROIs are typically areas where abnormalities, lesions, or significant activity are observed, or where researchers are particularly interested in focusing their analysis.

Key metrics in this analysis include the number of significant slices processed per patient, the distribution of pixel intensities in the original PET images within these slices, and the performance of the thresholding approach used to identify relevant regions in the predicted masks.

5.1.2 Patient and Slice Summary

A total of 89 patients were processed, with several distinct cases being analyzed. Across these patients, the number of significant image slices varied depending on the applied intensity thresholds and mask pixel count criteria, resulting in total significant slice counts of 962, 988, and 972. These slices correspond to regions in the PET images where pixel intensities exceeded defined thresholds in the predicted masks, making them relevant for further investigation.

Threshold values of 0.25, 0.5, and 0.75 were used to binarize image pixel intensities after scaling. A mask pixel count criterion was then applied to retain only slices where the number of predicted foreground pixels exceeded predefined limits of 1, 5, or 10. Additionally, we evaluated the model using two different epoch settings (50 and 100 epochs) to assess the impact of training duration on segmentation performance. This multi-case approach allowed for a comprehensive evaluation of significant slices under different sensitivity settings.

5.1.3 Slice Count per Patient

Across the various cases, the average number of significant slices per patient varied depending on the intensity threshold and mask pixel count criteria applied. For example, in one case, patients had an average of approximately 10.81 significant slices, while in others, the average slice count differed slightly. The minimum number of significant slices observed in a patient ranged from 1 to 3, while the maximum ranged from 26 to 30, reflecting differences in the extent of the condition or ROI identified in the predicted masks.

The standard deviation of the number of significant slices across patients also varied by case. For instance, a standard deviation of approximately 6.06 in one case provided

insight into the variability in the number of significant slices among patients, indicating a moderate spread around the mean. In other cases, the standard deviation may be higher or lower, reflecting different levels of variation in the slice counts depending on the parameters applied. This variability suggests that while some patients show extensive ROIs, others have more limited regions flagged as significant in the predicted masks.

5.1.4 Intensity Distribution Across Slices

The image pixel intensities within the significant slices were normalized across all cases to a common scale ranging from 0 to 255. Across cases, the distribution of image pixel intensities showed similar trends. For instance, in one case, a large proportion of the image pixels (79.4%) fell within the lowest intensity range of 0 to 25, suggesting that many of the pixels in the significant slices represented areas with relatively low signal activity. This trend was consistent across other cases, although slight variations in percentage were observed.

Image pixels with moderate intensity levels (26 to 100) accounted for approximately 18.6% of the total in one case, with similar proportions seen in other cases. These pixels likely correspond to regions of intermediate activity, reflecting areas that may not reach the highest intensity values but are still noteworthy. Only a small percentage of image pixels (around 2%) exhibited intensities above 100, representing high-activity regions that could be clinically significant. This distribution was consistent across multiple thresholding and mask pixel-limit criteria.

5.1.5 Thresholding Performance

To segment the image slices, a thresholding approach was applied to the image pixel intensities, where pixel values below a given threshold were set to 0 (background) in the predicted mask, and those above were set to 1 (foreground). The threshold varied across cases, with values of 0.25, 0.5, and 0.75 tested. Additionally, slices were retained based on a mask pixel count criterion, with limits of 1, 5, and 10 predicted foreground pixels required for inclusion.

This process influenced the number of retained slices, with cases yielding 962, 988, and 972 significant slices depending on the specific threshold and mask pixel limit applied. The observed variability highlights the flexibility of the method in isolating high-intensity regions in the predicted masks while filtering out less relevant areas. In all cases, the thresholding technique effectively focused on clinically meaningful regions, ensuring that only the most significant slices were analyzed.

5.2 Qualitative Results

This section discusses the qualitative findings derived from our implementation of the U-Net segmentation model for medical imaging. The focus is on the themes that emerged during the training and evaluation phases, as well as the implications of these findings in the context of our research questions.

The first key theme that emerged was the importance of data normalization and augmentation. Throughout the data preparation process, it was noted that normalizing the images and masks helped improve the performance of the U-Net model. This practice

is essential in ensuring that the model learns effectively from the training data. Moreover, augmenting the training dataset by including horizontally flipped versions of the images and masks allowed for a more robust model that can generalize better to unseen data.

Another significant theme is the role of early stopping in model training. The implementation of early stopping based on validation loss proved to be critical in preventing overfitting, ensuring that the model retained its ability to generalize. Observing the loss curve during training highlighted the model's learning trajectory, reinforcing the importance of monitoring performance metrics closely.

The findings also underscored the relevance of Dice coefficients in model evaluation. This metric, used to assess the accuracy of the predicted segmentation against the ground truth, provided insightful feedback on the model's performance. The qualitative analysis of the model's predictions showed that higher Dice scores often corresponded to clearer and more distinct boundaries in the segmented images, illustrating the model's capability to identify significant Region of interest.

Additionally, the iterative training process brought to light the collaborative efforts and challenges faced by the research team. During the training iterations, discussions around hyperparameter tuning and loss function adjustments led to valuable insights that influenced the model's architecture and training strategy.

Finally, it was noted that the relationship between quantitative metrics and qualitative observations was crucial in understanding the efficacy of the segmentation model. The quantitative results, such as the mean and median Dice coefficients, complemented the qualitative insights gained from examining the segmented outputs. This interplay between data-driven results and observational analysis enabled the team to refine their approach continually, leading to a model that not only performs well statistically but also aligns with clinical expectations.

In summary, the qualitative findings emphasize the importance of thoughtful data preparation, effective training strategies, and a close examination of performance metrics. These insights contribute to a more nuanced understanding of the U-Net model's capabilities and the overall impact on medical image analysis.

5.3 Statistical Test Outcomes

In this section, we present the results of the statistical tests conducted to evaluate the performance of the U-Net segmentation model. The model is designed to process multi-channel medical images, and its performance is assessed through a rigorous training and testing framework, including data augmentation, multi-set validation, and iterative evaluation across different test sets. Dice coefficients are used as the primary metric to measure segmentation accuracy. These tests aimed to ascertain the significance of the model's predictive accuracy and understand the results' reliability.

Choice of Statistical Analysis: The analysis focused on comparing the distributions of the Dice coefficients across different test sets to assess the performance of the model under varying conditions. While no formal statistical tests (e.g., hypothesis testing for comparing means) were conducted, the comparisons provided insights into the variability and trends in the Dice scores. This approach highlights differences in performance but does not account for statistical variation, meaning observed trends could be influenced

by random chance rather than consistent differences.

5.3.1 Background

Recently, Liedes et al. [24] have investigated the performance of machine learning models in segmenting PET images. For instance, when evaluating true positive segmentations, models have achieved mean Dice scores of 0.79 and classification accuracies of 0.62 for PET images in test datasets. A score of 0.79 suggests a relatively high level of agreement between the predicted and true segmentations, though there is still room for improvement in both segmentation accuracy and classification. Furthermore, models trained solely on PET data have demonstrated the ability to segment malignant tissue with reasonable accuracy. In one case, a deep learning model trained exclusively on PET images achieved a median Dice score of 0.68 in segmenting malignant tissue from test images. Although this score is somewhat lower than the previously mentioned performance on true positive segmentations, it reflects the challenges posed by the unique characteristics of PET imaging, such as the lower spatial resolution and the complexity of the metabolic information captured. They considered the Image dimension to be 128, the threshold value to be 0.25, and the pixel limit to be 5. We have compiled a summary of these details in Table 1.

img_height	limit	PixLimit	Mean of Dice	Median of Dice
128	0.25	5	0.79	0.69

Table 1: Parameters and Dice scores obtained by Liedes et al. [24]

In light of these findings, the ongoing development of segmentation models tailored to PET imaging holds promise for improving the accuracy and utility of PET in clinical workflows. This section will further examine and refine segmentation techniques for PET images, to improve performance metrics, such as Dice scores.

5.3.2 Some Other Cases

In this section, we examine additional cases regarding the height of images, threshold values, and pixel limits. We consider five cross-validation test sets, and then we perform five iterations for each test set. The test sets we evaluated are as follows:

```
testSets = [testSet1, testSet2, testSet3, testSet4, testSet5],
```

where

```
testSet1 = range(1, 100, 5);
testSet2 = range(2, 100, 5);
testSet3 = range(3, 100, 5);
testSet4 = range(4, 100, 5);
testSet5 = range(0, 100, 5).
```

In each iteration, we calculate both the mean and median Dice coefficients. After

all iterations are complete, we compute the average of these mean and median Dice coefficients across all 25 iterations, providing a final summary of the overall performance for each case. We remark that in this project, we used Python (version: 3.12.0) with packages TensorFlow (version: 2.16.0-rc0) and Keras (version: 3.0.5) for computational tasks, while the U-net model by Liedes et al. [24] was built using the TensorFlow Keras framework (version 2.5.0) in Python (version 3.7.10). We summarize the output for all cases in the following tables:

5.3.3 Image dimension = 128

In this section, we examine the scenario where the Image dimension is 128, with limits of 0.25, 0.5, and 0.75, and a pixel limit of 1, 5, and 10. We start with the following: **Case 1:** `img_height=128`, `limit=0.25`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	**Value**	**Value**	**Value**
Image dimension	128	128	128
Limit	0.25	0.25	0.25
Pixel Limit	1	0.5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	89	89	88
Total number of slices across all patients	988	962	909
The mean number of slices per patient	11.1	10.81	10.21
The SD of the number of slices / patient	6.06	6.06	6.15
Total number of slices in the test set	213	206	192
Test set slices/Total slices (all patients)	0.26	0.21	0.21
Total number of patients in the test set	17	17	17
Average of Mean Dice	0.45	0.46	0.41
Average of Median Dice	0.49	0.51	0.46

Table 2: Summary of Model Parameters and Dataset Characteristics for Case 1, including Patient Counts, Slice Statistics, and Dice Score Averages

Case 2: `img_height=128`, `limit=0.5`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	Value	Value	Value
Image dimension	128	128	128
Limit	0.5	0.5	0.5
Pixel Limit	1	5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	89	89	86
Total number of slices across all patients	983	952	885
The mean number of slices per patient	11.04	10.7	9.94
The SD of the number of slices / patient	6.02	6.003	6.2
Total number of slices in the test set	213	204	191
Test set slices / Total slices (all patients)	0.22	0.21	0.22
Total number of patients in the test set	17	17	17
Average of Mean Dice	0.43	0.6	0.46
Average of Median Dice	0.45	0.67	0.49

Table 3: Summary of Model Parameters and Dataset Characteristics for Case 2, including Patient Counts, Slice Statistics, and Dice Score Averages

Case 3: `img_height=128`, `limit=0.75`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	Value	Value	Value
Image dimension	128	128	128
Limit	0.75	0.75	0.75
Pixel Limit	1	5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	89	89	86
Total number of slices across all patients	981	937	972
The mean number of slices per patient	11.02	10.53	9.8
The SD of the number of slices / patient	5.99	6.05	6.16
Total number of slices in the test set	212	199	188
Test set slices / Total slices (all patients)	0.22	0.21	0.22
Total number of patients in the test set	17	17	17
Average of Mean Dice	0.45	0.43	0.49
Average of Median Dice	0.48	0.46	0.54

Table 4: Summary of Model Parameters and Dataset Characteristics for Case 3, including Patient Counts, Slice Statistics, and Dice Score Averages

5.3.4 Image dimension = 64

Next, we study the scenario where the Image dimension is 64, with limits of 0.25, 0.5, and 0.75, and a pixel limit of 1, 5, and 10.

Case 4: `img_height=64`, `limit=0.25`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	Value	Value	Value
Image dimension	64	64	64
Limit	0.25	0.25	0.25
Pixel Limit	1	5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	88	79	72
Total number of slices across all patients	949	767	612
The mean number of slices per patient	10.66	8.62	6.88
The SD of the number of slices / patient	6.14	5.98	5.36
Total number of slices in the test set	202	159	118
Test set slices / Total slices (all patients)	0.21	0.21	0.19
Total number of patients in the test set	17	15	12
Average of Mean Dice	0.35	0.47	0.52
Average of Median Dice	0.37	0.5	0.56

Table 5: Summary of Model Parameters and Dataset Characteristics for Case 4, including Patient Counts, Slice Statistics, and Dice Score Averages

Case 5: `img_height=64`, `limit=0.5`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	Value	Value	Value
Image dimension	64	64	64
Limit	0.5	0.5	0.5
Pixel Limit	1	5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	88	78	72
Total number of slices across all patients	937	747	584
The mean number of slices per patient	10.53	8.39	6.56
The SD of the number of slices/patient	6.1	5.91	5.36
Total number of slices in the test set	198	158	116
Test set slices / Total slices (all patients)	0.21	0.21	0.2
Total number of patients in the test set	17	15	12
Average of Mean Dice	0.46	0.5	0.51
Average of Median Dice	0.5	0.55	0.54

Table 6: Summary of Model Parameters and Dataset Characteristics for Case 5, including Patient Counts, Slice Statistics, and Dice Score Averages

Case 6: `img_height=64`, `limit=0.75`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	Value	Value	Value
Image dimension	64	64	64
Limit	0.75	0.75	0.75
Pixel Limit	1	5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	88	78	72
Total number of slices across all patients	928	730	573
The mean number of slices per patient	10.43	8.2	6.44
The SD of the number of slices/patient	6.07	5.85	5.31
Total number of slices in the test set	197	154	113
Test set slices / Total slices (all patients)	0.21	0.21	0.2
Total number of patients in the test set	17	15	12
Average of Mean Dice	0.44	0.46	0.46
Average of Median Dice	0.48	0.5	0.49

Table 7: Summary of Model Parameters and Dataset Characteristics for Case 6, including Patient Counts, Slice Statistics, and Dice Score Averages

5.3.5 Image dimension = 256

We will examine a scenario with an Image dimension of 256 pixels, using limits of 0.25, 0.5, and 0.75, and pixel limits of 1, 5, and 10.

Case 7: `img_height=256`, `limit=0.25`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	Value	Value	Value
Image dimension	256	256	256
Limit	0.25	0.25	0.25
Pixel Limit	1	5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	89	89	89
Total number of slices across all patients	933	988	982
The mean number of slices per patient	11.16	11.1	11.03
The SD of the number of slices/patient	6.07	6.04	6.05
Total number of slices in the test set	217	214	212
Test set slices / Total slices (all patients)	0.22	0.22	0.22
Total number of patients in the test set	17	17	17
Average of Mean Dice	0.43	0.45	0.44
Average of Median Dice	0.46	0.49	0.48

Table 8: Summary of Model Parameters and Dataset Characteristics for Case 7, including Patient Counts, Slice Statistics, and Dice Score Averages

Case 8: `img_height=256`, `limit=0.5`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	Value	Value	Value
Image dimension	256	256	256
Limit	0.5	0.5	0.5
Pixel Limit	1	5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	89	89	89
Total number of slices across all patients	992	985	976
The mean number of slices per patient	11.15	11.07	10.97
The SD of the number of slices/patient	6.08	6.04	6.02
Total number of slices in the test set	216	213	211
Test set slices / Total slices (all patients)	0.22	0.22	0.22
Total number of patients in the test set	17	17	17
Average of Mean Dice	0.47	0.42	0.41
Average of Median Dice	0.5	0.45	0.45

Table 9: Summary of Model Parameters and Dataset Characteristics for Case 8, including Patient Counts, Slice Statistics, and Dice Score Averages

Case 9: `img_height=256`, `limit=0.75`, `pixLimit=1,5,10`, and `numEpochs=50`.

Description	Value	Value	Value
Image dimension	256	256	256
Limit	0.75	0.75	0.75
Pixel Limit	1	5	10
Number of Epochs	50	50	50
Number of patients with at least one slice	89	89	89
Total number of slices across all patients	991	983	973
The mean number of slices per patient	11.13	11.04	10.93
The SD of the number of slices/patient	6.02	6.01	6.03
Total number of slices in the test set	215	213	209
Test set slices / Total slices (all patients)	0.22	0.22	0.21
Total number of patients in the test set	17	17	17
Average of Mean Dice	0.48	0.47	0.46
Average of Median Dice	0.51	0.5	0.49

Table 10: Summary of Model Parameters and Dataset Characteristics for Case 9, including Patient Counts, Slice Statistics, and Dice Score Averages

5.3.6 Further Cases

This section examines cases where the pixel limit is 7. We start with the following case:

Case 10: `img_height = 64`, `limit = 0.25, 0.5, 0.75`, and `pixLimit = 7`.

Description	Value	Value	Value
Image dimension	64	64	64
Limit	0.25	0.5	0.75
Pixel Limit	7	7	7
Number of Epochs	50	50	50
Number of patients with at least one slice	78	77	77
Total number of slices across all patients	704	685	664
Total number of slices in the test set	145	143	139
Mean Dice	0.3	0.41	0.49
Median Dice	0.25	0.45	0.52

Table 11: Summary of Model Parameters and Dataset Characteristics for Case 10, including Patient Counts, Slice Statistics, and Dice Score Averages

Case 11: `img_height = 128`, `limit = 0.25, 0.5, 0.75`, and `pixLimit = 7`.

Description	Value	Value	Value
Image dimension	128	128	128
Limit	0.25	0.5	0.75
Pixel Limit	7	7	7
Number of Epochs	50	50	50
Number of patients with at least one slice	89	89	88
Total number of slices across all patients	936	923	914
Total number of slices in the test set	185	182	182
Mean Dice	0.47	0.43	0.24
Median Dice	0.54	0.46	0.05

Table 12: Summary of Model Parameters and Dataset Characteristics for Case 11, including Patient Counts, Slice Statistics, and Dice Score Averages

5.3.7 Reducing the number of patients and increasing the number of epochs

One of the cases we considered was decreasing the number of patients from 30 to 20 and increasing the number of epochs from 50 to 100. We used the following code:

```
numEpochs=100
earlystopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
mode='min', patience=20, restore_best_weights=True)
```

Case 12: `img_height = 128`, `limit = 0.25`, and `pixLimit = 5,4,1`. In this case, we should set the number of epochs to 100. We present a summary of the results in Table 13.

Description	Value	Value	Value
Image dimension	128	128	64
Limit	0.25	0.25	0.25
Pixel Limit	5	4	1
Number of Epochs	100	100	100
Number of patients with at least one slice	89	89	88
Total number of slices across all patients	962	968	949
Total number of slices in the test set	191	192	188
Mean Dice	0.49	0.5	0.45
Median Dice	0.56	0.58	0.53

Table 13: Summary of Model Parameters and Dataset Characteristics for Case 12, where `numEpochs = 100` and `patient = 20`

Case 13: Finally, we set the Image dimension to 128 pixels and the number of training epochs to 100. We apply the following abbreviations for the next table.

- `ih` $\rightarrow$ `img_heigh`
- `l` $\rightarrow$ `limit`
- `p` $\rightarrow$ `pixLimit`
- `tnp` $\rightarrow$ `The total number of patients with at least one slice`
- `tnsp` $\rightarrow$ `The total number of slices across all patients`
- `tnst` $\rightarrow$ `The total number of slices in the test set`
- `md_tp` $\rightarrow$ `The median of the Dice coefficients calculated for the training predictions`
- `m_tp` $\rightarrow$ `The mean of the Dice coefficients calculated for the training predictions`
- `md_p` $\rightarrow$ `The median of the Dice coefficients calculated for the test set`
- `m_p` $\rightarrow$ `The mean of the Dice coefficients calculated for the test set`

5.3.8 Augmentation Techniques in Deep Learning for Computer Vision

Deep learning advancements in computer vision owe much to powerful algorithms, efficient hardware, and access to large image datasets. However, training robust models with numerous parameters often requires vast datasets, which can be challenging and costly. This issue is particularly pronounced in specialized fields like medical imaging [29]

Variable	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
epoch	100	100	100	100	100	100	100	100	100	100	100	100
ih	128	128	128	128	128	128	128	128	128	128	128	128
l	0.25	0.25	0.25	0.25	0.5	0.5	0.5	0.5	0.75	0.75	0.75	0.75
p	1	5	7	10	1	5	7	10	1	5	7	10
tnp	89	89	89	88	89	89	89	86	89	89	88	86
tnsp	988	962	936	909	983	952	923	885	981	937	914	872
tnst	194	191	185	182	193	190	182	179	193	185	182	176
m_tp	0.54	0.52	0.53	0.57	0.53	0.58	0.55	0.56	0.34	0.61	0.56	0.55
md_tp	0.67	0.6	0.61	0.64	0.6	0.68	0.62	0.66	0.32	0.71	0.63	0.63
m_p	0.49	0.36	0.45	0.45	0.42	0.44	0.46	0.46	0.25	0.49	0.45	0.44
md_p	0.58	0.36	0.49	0.49	0.43	0.52	0.51	0.54	0.0	0.52	0.54	0.52

Table 14: Summary of Model Parameters and Dataset Characteristics for Case 19, where `numEpochs = 100` and `img_geight = 128`

and agriculture [54], where annotated datasets are limited due to privacy concerns and data scarcity.

Image augmentation has emerged as a critical strategy to mitigate these challenges [52, 48]. Augmentation techniques enhance both the quantity and diversity of training data, helping models generalize better to unseen scenarios. Table 15 outlines various augmentation methods used in tasks like image classification and object detection [53].

NNA	**Image Augmentation Method**
AlexNet	Translate, Flip, Intensity Changing
ResNet	Crop, Flip
DenseNet	Flip, Crop, Translate
MobileNet	Crop, Elastic Distortion
NasNet	Cutout, Crop, Flip
ResNeSt	AutoAugment, Mixup, Crop
DeiT	AutoAugment, RandAugment, Random Erasing, Mixup, CutMix
Swin Transformer	RandAugment, Mixup, CutMix, Random Erasing
Faster R-CNN	Flip
YOLO	Scale, Translate, Color Space
SSD	Crop, Resize, Flip, Color Space, Distortion
YOLOv4	Mosaic, Distortion, Scale, Color Space, Crop, Flip, Rotate, Random Erase, Cutout, Hide-and-Seek, GridMask, Mixup, CutMix, StyleGAN

Table 15: Image Augmentation Techniques in Image Classification and Object Detection Research. NNA stands for neural network architectures.

Challenges in Image Augmentation: Several challenges make augmentation essential for effective model training:

Image Variations: Variability in illumination, deformation, and other factors can complicate training.

Class Imbalance: Imbalanced datasets, common in medical imaging where anomalies are rare, can bias models toward majority classes.

Domain Shift: Differences between training and testing distributions, such as lighting conditions in autonomous driving, lead to performance discrepancies.

Structural Risk: Increasing model complexity necessitates larger datasets to prevent overfitting and achieve generalization. Addressing these challenges requires a nuanced understanding of augmentation techniques to ensure effective application.

Single-Image Augmentation (SiA): SiA focuses on enhancing the training dataset by applying transformations to individual images, thus increasing data density and variability. These methods, often rooted in traditional image processing, are straightforward to implement and broadly categorized into three types: geometric transformations, color processing, and intensity transformations [9].

Geometric Transformations: Geometric transformations adjust the spatial relationships within images to simulate real-world variability. Common techniques include:

Translation: Shifts an object's position within an image.

Rotation: Changes the perspective of objects while preserving label integrity.

Flipping: Applies horizontal or vertical reflections based on dataset characteristics.

Scaling: Modifies object size to represent different scales.

Elastic Distortions: Alters object shapes for diversity. Figure 5 illustrates these transformations widely used in classification, object detection, and image translation tasks.

Color Image Processing: Color processing manipulates the RGB channels of color images to emphasize different features. While it is less common due to minimal color variations in real-world datasets, it proves useful in contrastive learning for task-agnostic representation learning. Techniques include channel swapping, contrast enhancement, and intensity adjustment, as shown in Figure 6.

Intensity Transformations: Unlike geometric and color transformations, intensity transformations operate at the pixel or patch level. These include:

Random Noise: Adds Gaussian or other noise types to simulate variability.

Cutout: Masks significant regions of an image, simulating occlusion (Figure 7).

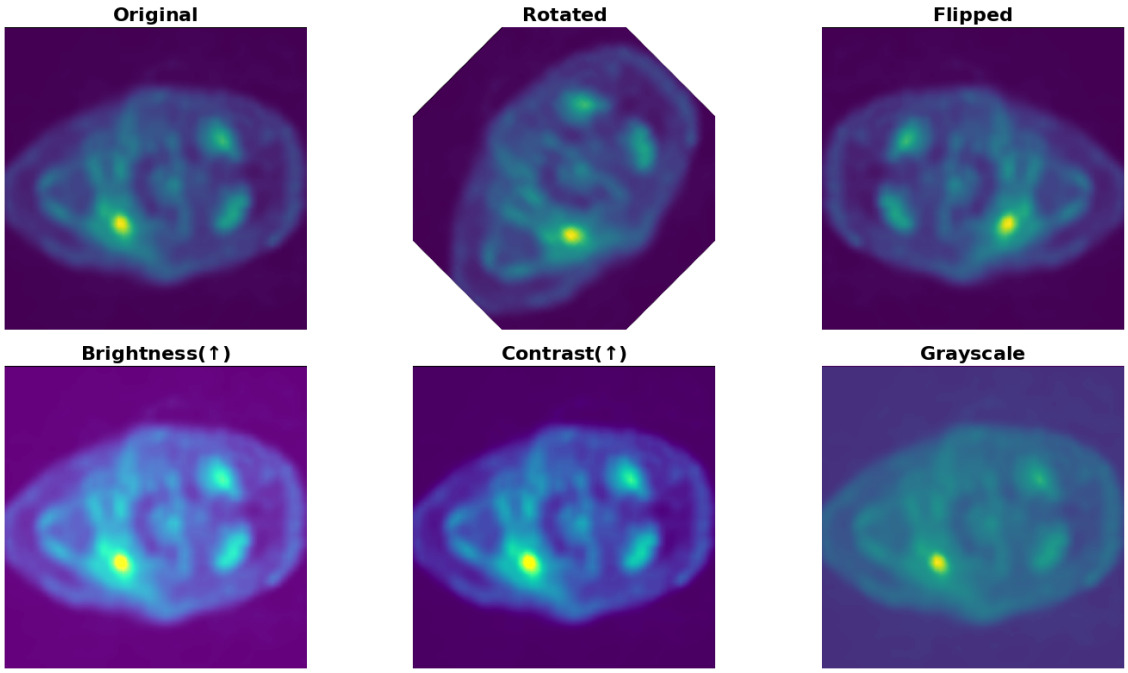


Figure 5: Various image augmentations applied to the original image: rotation, horizontal flip, brightness enhancement, contrast adjustment, and grayscale conversion. The original image is a PET scan that visualizes metabolic activity within tissues. Brighter regions indicate areas of higher metabolic activity, which could correspond to abnormal growths, inflammation, or other physiological processes.

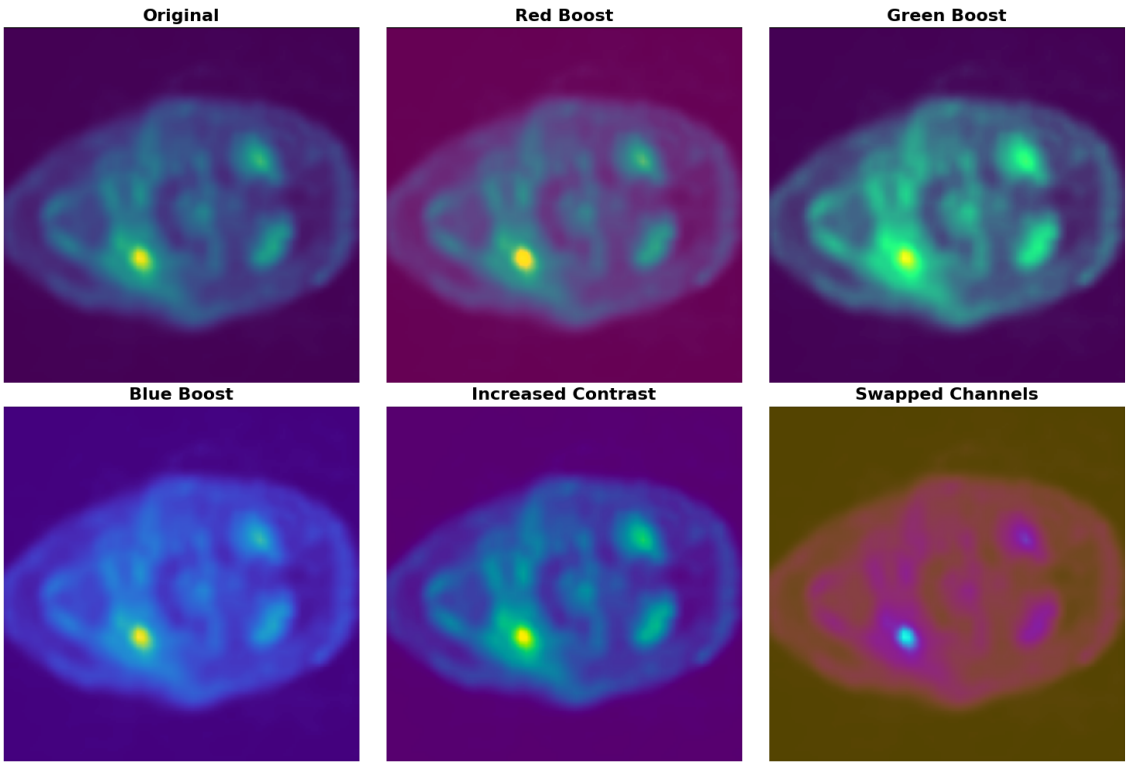


Figure 6: Visualization of color image processing techniques: original image, RGB channel enhancements, increased contrast, and channel swapping to show feature extraction variability.

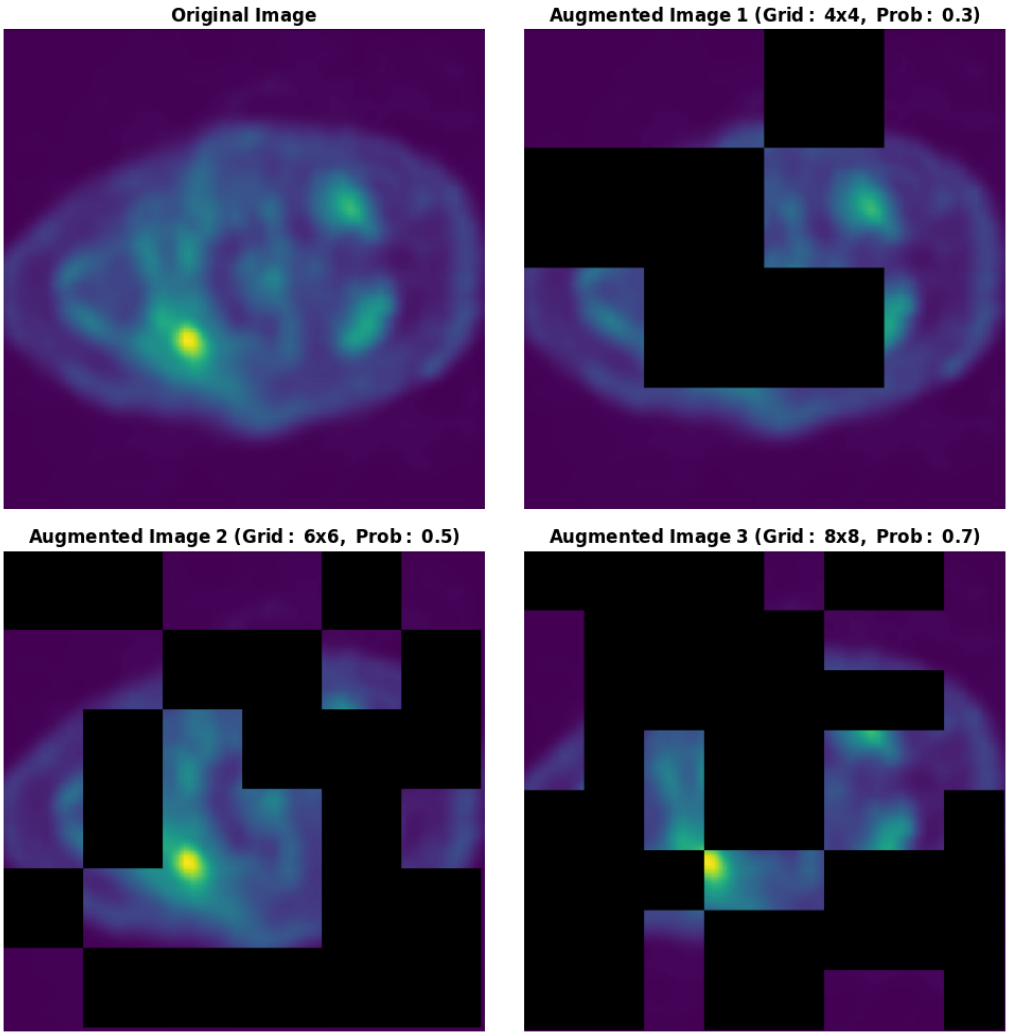


Figure 7: Visualization of images divided into patches of sizes $s \times s$, where $s = 4, 6, 8$, showcasing random occlusion applied to individual patches for augmentation.

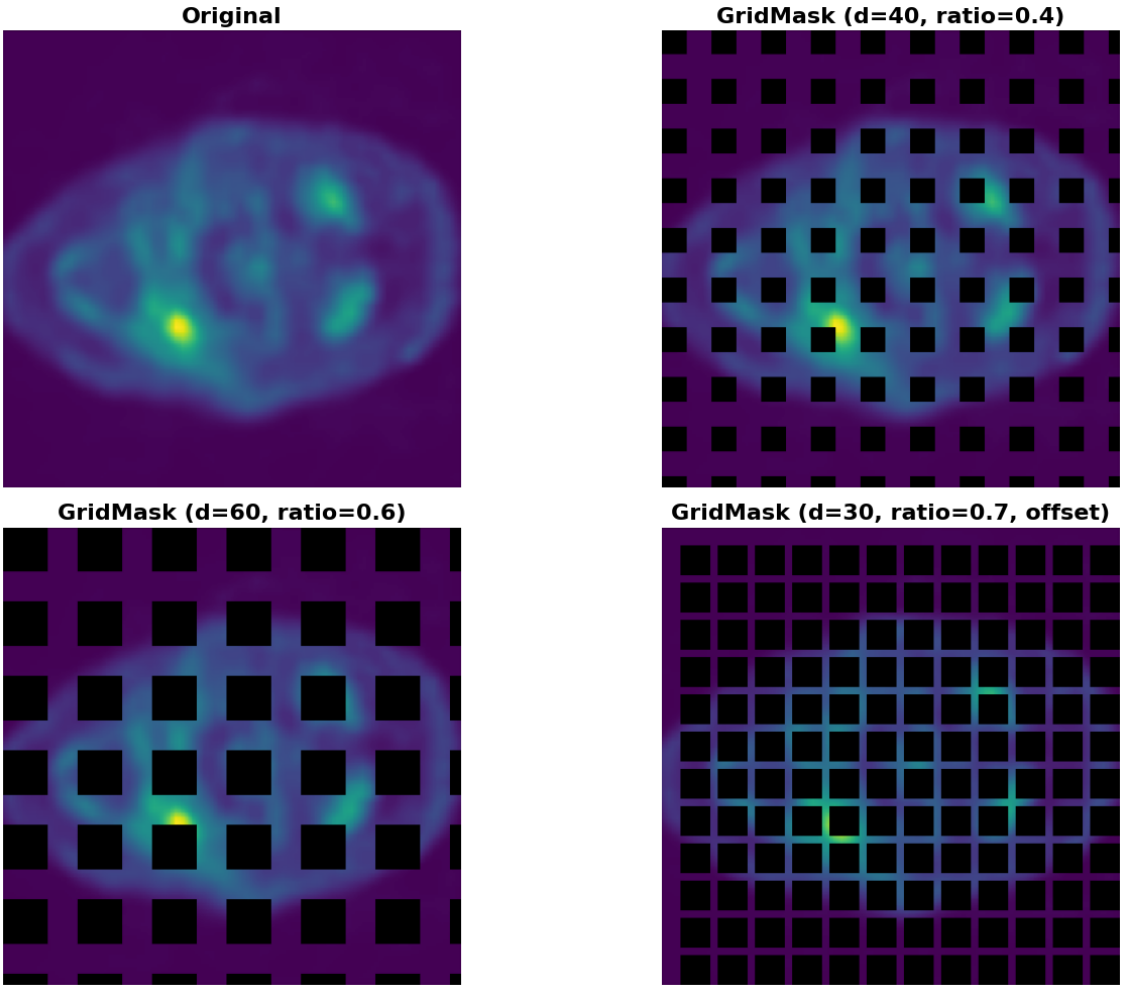


Figure 8: GridMask augmentation applied with varying grid sizes (d) and masking ratios to demonstrate balanced occlusion and feature preservation.

Hide-and-Seek: Conceals image parts iteratively to obscure key features efficiently. Random Erasing: Randomizes occluded areas' size, ratio, and position to create diverse scenarios.

GridMask: Balances feature deletion and preservation using predefined masks for controlled occlusion (Figure 8). GridMask, in particular, achieves a superior balance between information retention and deletion, outperforming simpler methods like Cutout.

5.3.9 The Results After Implementing Augmentation Techniques

This section explores various augmentation techniques, such as flipping, random cropping, Gaussian blur, and rotation. Our study investigates the impact of data augmentation on medical image segmentation accuracy, as measured by the Dice coefficient. Specifically, we evaluate whether augmentation improves segmentation performance and

explore differences in effectiveness between various augmentation techniques. By comparing results across multiple test sets and iterations, we identify which augmentation methods enhance Dice scores and which fail to contribute significantly. Furthermore, we analyze the performance trends to determine the optimal augmentation strategy for improving segmentation outcomes. The findings provide insights into the role of augmentation in segmentation tasks and highlight the most effective techniques for this purpose.

Based on the tables presented, we conclude that the best results are achieved in Case 2, where the parameters are set to `img_height=128`, `limit=0.5`, `pixLimit=5`, and `numEpochs=50`. To enhance our understanding, we will critically explore the potential benefits of employing augmentation techniques in this situation.

Case 10: Rotation 90 Degrees in a Clockwise Direction `img_height = 128`, `limit = 0.5`, and `pixLimit = 5`.

In Case 10, the segmentation performance was assessed using the Dice coefficient, with all images rotated 90 degrees in a clockwise direction to standardize orientation. We apply the following code:

```
x_train1.append(cv2.rotate(x_train[i],cv2.ROTATE_90_CLOCKWISE))
y_train1.append(cv2.rotate(y_train[i],cv2.ROTATE_90_CLOCKWISE))
```

Images in `x_train` are preprocessed by normalizing their pixel values to a range between 0 and 1, ensuring consistency across the dataset, while the masks, `y_train`, are resized to the specified `img_height` and represent the ground truth labels for the segmentation task (e.g., binary masks for foreground and background). In the context of image segmentation tasks, masks refer to images that serve as the ground truth labels for the segmentation model. They define the regions of interest (e.g., specific objects or areas) in the corresponding input images.

The processing parameters for this case included an Image dimension of 128 pixels, ensuring consistent spatial resolution, a threshold limit of 0.5 to binarize image features, and a pixel limit of 5 to filter out less significant regions in each slice. This setup aimed to refine feature extraction and segmentation accuracy. This technique doubles the size of the training dataset, enhancing model robustness by introducing rotated perspectives. The resulting mean Dice coefficient was 0.48, indicating moderate overlap between predicted and actual segmentation regions, while the median Dice coefficient was slightly higher at 0.54, suggesting a consistent performance across most cases in this dataset. The execution time for this code was 798 minutes and 42 seconds. We present a summary of the results in Table 16.

Case 11: Gaussian blur `img_height = 128`, `limit = 0.5`, and `pixLimit = 5`.

Gaussian blur is a widely used image processing technique that reduces noise and detail in an image. The function is named after the Gaussian function, which is defined as follows:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right),$$

where x and y represent the pixel coordinates, and σ is the standard deviation that

Description	Value
Image dimension	128
Limit	0.5
Pixel Limit	5
Number of Epochs	50
Number of patients with at least one slice	89
Total number of slices across all patients	952
The mean number of slices per patient	10.7
The standard deviation of the number of slices per patient	6.003
Total number of slices in the test set	204
Test set slices / Total slices (all patients)	0.21
Total number of patients in the test set	17
Average of Mean Dice	0.48
Average of Median Dice	0.54

Table 16: Summary of model parameters and dataset characteristics for Case 10, with a 90-degree clockwise rotation. This includes patient counts, slice statistics, and average Dice scores.

controls the extent of the blur. This function describes a bell-shaped curve that mathematically determines the weight of each pixel in a neighborhood based on its distance from the center. This technique smoothens an image by averaging nearby pixel intensities, weighted by the Gaussian distribution. A Gaussian kernel, generated using the Gaussian function, is convolved with the image. The size of the kernel and the value of σ determine the extent of the blurring effect. Pixels that are closer to the center of the kernel have a greater influence on the weighted average compared to those further away. This approach helps preserve the structure of the image while effectively reducing noise [14].

In Case 11, segmentation performance was evaluated using the mean Dice coefficients after applying Gaussian blur to the images. The images were processed with a set Image dimension of 128 pixels, a thresholding limit of 0.5 for binarization, and a pixel limit of 5 to exclude minor artifacts from the analysis. This configuration helped control noise and enhanced relevant features within each slice. We applied the following code to enhance the training dataset by introducing a data augmentation technique using Gaussian blur.

```
# Data augmentation with Gaussian blur
number = len(x_train)
x_train1, y_train1 = [], []
for i in range(number):
    x_train1.append(x_train[i])
    x_train1.append(cv2.GaussianBlur(x_train[i], (5, 5), 0))  # Apply Gaussian blur
    y_train1.append(y_train[i])
    y_train1.append(y_train[i])  # Mask remains the same
x_train = np.array(x_train1)
y_train = np.array(y_train1)
```

The code doubles the size of the dataset by adding a blurred version of each original image to the training set. For every image in `x_train`, a Gaussian blur is applied using

OpenCV's GaussianBlur function with a kernel size of $(5, 5)$ and a standard deviation of 0. The corresponding labels in `y_train` are duplicated, ensuring that the augmented images retain the same labels as their original counterparts. This ensures consistency in the dataset while increasing its diversity, potentially helping the model generalize better to unseen data. After augmentation, the training data arrays, `x_train` and `y_train`, are updated with the expanded data.

The study produced a mean Dice coefficient of 0.49, reflecting a modest alignment between predicted and actual regions, and a median Dice coefficient of 0.55, indicating similar results across most images in the dataset.

Case 12: Horizontal Flipping `img_height = 128`, `limit = 0.5`, and `pixLimit = 5`.

Original images are duplicated, and a horizontally flipped version is added for each image along with its mask. This method doubles the size of the training dataset and enhances the diversity of the training data. However, the segmentation results were assessed using mean Dice coefficients, with training data comprising both original and augmented images to enhance model robustness. The analysis used a standardized Image dimension of 128 pixels, with a binarization limit of 0.5 to define feature thresholds, and a pixel limit of 5 to filter out noise and irrelevant regions. This combined training approach yielded a mean Dice coefficient of 0.5272, reflecting a moderate agreement between predicted and actual regions, while the median Dice coefficient of 0.6085 suggests a strong performance across most samples.

Case 13: Data Augmentation and Noise Reduction `img_height = 128`, `limit = 0.5`, and `pixLimit = 5`.

In this case, we perform data augmentation and preprocessing to enhance the diversity and quality of the training dataset used for semantic segmentation. The augmentation process involves several transformations to increase the variability of the dataset, improve model generalization, and reduce overfitting.

Horizontal Flipping: Each input image and its corresponding mask are horizontally flipped using OpenCV's cv2.flip function. This technique effectively doubles the size of the training dataset by introducing mirrored versions of the original samples, ensuring the model learns from diverse perspectives. The following two lines of code were executed.

```
# Horizontal Flip
flipped_img = cv2.flip(img, 1)  # Flip image horizontally
flipped_mask = cv2.flip(mask, 1)  # Flip mask horizontally
```

Resizing: To standardize input dimensions, all images and masks (both original and flipped) are resized to 128×128 pixels using cv2.resize. For masks, nearest-neighbor interpolation is employed to preserve label consistency. This resizing ensures uniformity in input size, which is a prerequisite for efficient training in CNNs. We applied the following code:

```
# Resize to 128x128 pixels
resized_img = cv2.resize(img, (128, 128))
```

```
resized_mask = cv2.resize(mask, (128, 128), interpolation=cv2.
    INTER_NEAREST)
resized_flipped_img = cv2.resize(flipped_img, (128, 128))
resized_flipped_mask = cv2.resize(flipped_mask, (128, 128),
    interpolation=cv2.INTER_NEAREST)
```

Normalization: The pixel values of all images are normalized to the range $[0, 1]$ by dividing by 255. This normalization step standardizes the input data, improving the convergence speed and stability of the model during training.

Noise Reduction in Masks: To mitigate the impact of noisy regions in segmentation masks, a filtering function is applied to remove small connected regions with an area of less than five pixels. Using `cv2.connectedComponentsWithStats`, connected components are identified, and only regions that meet the size threshold are retained. This ensures that the model focuses on learning meaningful structures in the masks, reducing distractions caused by noise.

Dataset Augmentation: The augmented dataset, consisting of both resized original and flipped versions of the images and masks, is then converted into NumPy arrays. The images are stored as floating-point arrays (`dtype=np.float32`), while the masks are stored as unsigned integers (`dtype=np.uint8`). Using the code below, we obtain the following result:

```
# Noise Reduction: Ignore mask regions smaller than 5 pixels
def filter_small_regions(mask, min_size=5):
  # Find connected components in the mask
  num_labels, labels, stats, _ = cv2.connectedComponentsWithStats(
      mask.astype(np.uint8), connectivity=8)
  # Create a filtered mask
  filtered_mask = np.zeros_like(mask, dtype=np.uint8)
  for label_idx in range(1, num_labels):  # Start from 1 to skip
      the background
      if stats[label_idx, cv2.CC_STAT_AREA] >= min_size:
      filtered_mask[labels == label_idx] = 1
  return filtered_mask
# Apply noise reduction to all masks
y_train_aug = np.array([filter_small_regions(mask) for mask in
    y_train_aug])
```

The analysis maintained an image dimension of 128 pixels, a threshold limit of 0.5 for feature binarization, and a pixel limit of 5 to minimize noise influence. This augmentation-focused approach achieved a mean Dice coefficient of 0.5285, suggesting reliable overlap between predicted and actual regions, and a median Dice coefficient of 0.6229, indicating strong and consistent performance across the majority of test samples.

Case 14: Random Cropping, Horizontal Flipping, and Gaussian Blur: In this case, the training dataset includes both the original and augmented images. This approach can help improve the robustness of the model by exposing it to a diverse set of training samples while retaining the original data. The following augmentation sequence, see code below, performs three operations on the input images:

i) Random cropping of up to 16 pixels from each side;

ii) Horizontal flipping of images with a 50% probability;

iii) Gaussian blur with a random sigma value between 0 and 3.

```
seq = iaa.Sequential([
        iaa.Crop(px = (0, 16)),
        iaa.Fliplr(0.5),
        iaa.GaussianBlur(sigma = (0, 3.0))
    ])
```

The first line creates a sequence of augmentation operations that will be applied sequentially to the input images. The second line crops images from each side by a random number of pixels within the range of 0 to 16 pixels. The cropping is applied symmetrically on all sides (top, bottom, left, and right).). The third line horizontally flips images with a probability of 0.5, meaning that each image has a 50% chance of being flipped along the vertical axis. finally, the fourth line applies Gaussian blur to images with a sigma value (standard deviation) randomly chosen from the range of 0 to 3. We summarize the out in Table 17.

Description	**Value**
Image dimension	128
Limit	0.5
Pixel Limit	5
Number of Epochs	50
Number of patients with at least one slice	89
Total number of slices across all patients	952
The mean number of slices per patient	10.7
The standard deviation of the number of slices per patient	6.003
Total number of slices in the test set	204
Test set slices / Total slices (all patients)	0.21
Total number of patients in the test set	17
Average of Mean Dice	0.39
Average of Median Dice	0.38

Table 17: Summary of model parameters and dataset characteristics for Case 14, including random cropping, horizontal flipping, and Gaussian blur.

Case 15: Rotation with k degrees: The following code

```
seq = iaa.Sequential([iaa.Affine(rotate=(-15, 15))])
```

rotates the images k degrees, where k is a random number chosen from the interval $(-15, 15)$. The output is summarized in Table 18.

5.3.10 Comparison of Data Augmentation Techniques for Improved Segmentation Performance

Significant differences were observed between the augmentation techniques based on their effects on the average of mean Dice and average of median Dice coefficients.

Description	Value
Image dimension	128
Limit	0.5
Pixel Limit	5
Number of Epochs	50
Number of patients with at least one slice	89
Total number of slices across all patients	952
The mean number of slices per patient	10.7
The standard deviation of the number of slices per patient	6.003
Total number of slices in the test set	204
Test set slices / Total slices (all patients)	0.21
Total number of patients in the test set	17
Average of Mean Dice	0.33
Average of Median Dice	0.25

Table 18: Summary of model parameters and dataset characteristics for Case 15: rotation with a random k degrees chosen from the interval $(-15, 15)$.

Techniques such as Rotation 90 Degrees in a clockwise direction and Gaussian blur provided moderate improvements in segmentation performance, with Gaussian blur achieving slightly better results (mean Dice of 0.49, median Dice of 0.55) compared to 90-degree rotation (mean Dice of 0.48, median Dice of 0.54). These techniques introduced diversity and improved model robustness, but their impact was limited. The combination of Original and Augmented Training Data substantially improved performance, achieving mean and median Dice scores of 0.53 and 0.61, respectively. Similarly, Data Augmentation with noise reduction demonstrated strong results, achieving the highest median Dice score (0.62), indicating that refining the mask quality with noise reduction significantly benefits the segmentation task.

On the other hand, certain augmentation methods negatively impacted performance. For instance, the combination of random cropping, horizontal flipping, and Gaussian blur yielded mean and median Dice scores of only 0.39 and 0.38, suggesting that excessive variability introduced by this combination hindered the model's ability to learn effectively. The rotation with random k degrees (-15 to 15) performed the worst, with mean and median Dice scores of 0.33 and 0.25, likely due to the introduction of noisy distortions without adding meaningful diversity.

Based on the results, the best augmentation strategy for this purpose is Data augmentation with noise reduction, as it consistently achieved the highest segmentation quality by improving mask consistency and reducing irrelevant small regions. The combination of original and augmented data is also highly effective, demonstrating the importance of retaining the original dataset to maintain essential features while enhancing robustness with diverse augmentations. For optimal results, augmentation strategies should balance diversity with noise reduction, avoiding excessive randomness or unnecessary transformations that may degrade segmentation performance.

5.3.11 Statistical Evaluation of Augmentation Techniques for Image Segmentation Performance

Statistical testing was conducted to evaluate whether the augmentation techniques produced statistically significant differences in mean Dice scores compared to the baseline (no augmentation). The Wilcoxon signed-rank test was performed to compare the performance of no augmentation and the best-performing augmentation technique, which was Gaussian blur, across multiple iterations. The raw p-value for this comparison was calculated to be 0.0625, indicating that the observed differences in performance were not statistically significant at the standard threshold of 0.05. After applying Bonferroni correction to adjust for multiple comparisons, the adjusted p-value increased to 0.125, further supporting the conclusion that the improvement in Dice scores achieved by Gaussian blur could not be considered statistically significant.

The corrected p-values also indicate that none of the specific augmentations produced a definitive statistical improvement over the baseline. Additionally, Case 2 (with a 5-pixel limit) did not show any improvement when using augmentations, further reinforcing the lack of consistent benefit. Given these findings, further investigations could explore more impactful augmentation strategies or additional iterations to mitigate variability and assess potential long-term benefits.

5.3.12 Further Augmentation Techniques

We also propose several augmentation techniques to enhance the diversity and robustness of the training dataset.

1. Elastic Deformation: Simulates realistic distortions often observed in medical imaging. It stretches and compresses parts of the image using random displacement fields.
2. Random Brightness and Contrast Adjustment: Alters the brightness or contrast of images randomly to make the model invariant to lighting conditions.
3. Gamma Correction: Applies random gamma values to modify image intensities in a non-linear fashion, mimicking variable exposure conditions.
4. Intensity Scaling: Scales pixel intensities to simulate variations in imaging equipment or protocols.
5. Random Noise Injection: Adds Gaussian, salt-and-pepper, or speckle noise to the images to make the model robust to noisy data.
6. Affine Transformations: Includes random scaling, rotation, shearing, and translation within small ranges to further diversify spatial variations.
7. Random Erasing (Cutout): Removes random rectangular regions in the image, forcing the model to focus on less informative parts of the image.
8. Patch Shuffle or MixUp: Breaks the image into patches and shuffles or combines parts of different images to create new training examples.
9. Multi-Scale Cropping: Extracts crops at multiple scales and resolutions, ensuring the model learns features at different granularities.
10. Random Rotation with Larger Angle Ranges: Extends rotation beyond small angles, such as random rotations between -45° and 45° or even 90° intervals.
11. Histogram Equalization: Equalizes the intensity distribution across images, im-

proving contrast and emphasizing features.
12. Random Zoom-In and Zoom-Out: Mimics images taken at different magnification levels by zooming into or out of random regions of interest.
13. Synthetic Data Generation: Generates entirely new images using techniques like GANs (Generative Adversarial Networks) to expand the training dataset artificially.
14. Channel Shuffling: Randomly reorders image channels (if applicable, e.g., for multi-spectral imaging).
15. Clipping and Masking: Clips parts of the image outside a certain intensity range or masks specific regions randomly.
16. Style Transfer: Applies style transfer to give your images the appearance of being captured with a different modality or setting while preserving structural information.
17. Occlusion Augmentation: Simulates real-world scenarios where parts of the object may be obscured by adding random occlusions to the image.

5.3.13 Discussion on Augmentation

Augmentation techniques serve as powerful tools for enhancing the performance of deep learning models, particularly in medical imaging tasks like segmentation. However, the success of various augmentation methods can depend on the unique characteristics of the imaging modality and the model's architecture. For instance, PET imaging often captures metabolic processes and anatomical structures in a way that could be sensitive to certain transformations, such as rotations or flips. PET scans typically capture images in a relatively symmetric manner, particularly in the transaxial slices of the human body, where structures such as organs and tissues are mirrored along the central axis. This symmetry is often crucial for maintaining the integrity of the features being learned by the model. Techniques such as random rotations and horizontal flips could distort anatomical relationships, leading to poorer performance for augmentations like random rotations or flips, which can alter the body's expected symmetry, negatively affecting the accuracy of segmentation.

In contrast, augmentations like Gaussian blur may simulate natural noise or slight variations in image quality, which could help improve the model's robustness by making it less sensitive to real-world imaging inconsistencies. The use of Gaussian blur showed a small but notable improvement in the mean Dice scores, which might be attributed to the model's ability to generalize better to blurred and noisy regions in the images.

The U-Net architecture, with its encoder-decoder structure and skip connections, plays a significant role in how augmentations impact performance. U-Net is designed to preserve fine spatial details while progressively reducing the spatial dimensions. The success of augmentations, such as random cropping and Gaussian blur, might be due to U-Net's ability to integrate different features from various image scales, especially since U-Net learns hierarchical features through its contracting path. However, applying augmentation methods like random rotation or combined random cropping with flipping might disrupt U-Net's inherent ability to extract features consistently due to the induced geometric distortions.

Moreover, the structure of the U-Net may not handle extreme augmentations, like random rotations in the range of -15 to $+15$ degrees, particularly well. While such transformations may be useful for enhancing data diversity, they could also introduce

significant spatial mismatches, making it harder for the network to accurately map features across the slices. Similarly, the application of random cropping may reduce the consistency of anatomical structures in the training images, resulting in less precise segmentation outputs.

In summary, while augmentations like Gaussian blur and the combination of original and augmented training data provided the best improvements in model performance, their success could be attributed to the nature of PET imaging and U-Net's architectural properties. More complex augmentations like rotations and cropping likely hindered performance due to their interference with the natural symmetry and structural coherence of the human body, which the model relies on to make accurate predictions. This highlights the need for selecting augmentation techniques that align with the anatomical characteristics captured by PET scans and are compatible with the U-Net model's ability to learn spatial features.

6 Discussion

In this study, we aimed to assess various data augmentation techniques to improve the performance of a U-Net-based model for PET image segmentation. Despite testing several augmentation strategies, none significantly improved the segmentation performance compared to training without any augmentation. Among the tested methods, Gaussian blur showed the most promise, but its performance was not statistically different from the baseline.

These results suggest that, for this specific dataset and model architecture, simple data augmentation techniques may not offer substantial benefits. Some augmentations, such as random rotations and cropping, may have even disrupted essential anatomical structures in the images, leading to reduced model effectiveness.

Statistical tests, including the Wilcoxon signed-rank test, confirmed the lack of significant improvement. This highlights the need for more tailored augmentation strategies that align with the characteristics of PET images.

Future research could explore advanced or domain-specific augmentations, as well as investigate the interaction between augmentation and model architecture. A deeper understanding of PET imaging properties and anatomical consistency could help develop augmentations that enhance generalization and robustness in clinical applications.

6.1 Summary of Key Findings

This study focused on using advanced image processing techniques to analyze PET images from 89 patients diagnosed with a positive condition. A total of up to 992 significant image slices were identified, highlighting the variation in slice counts among patients, which reflects differences in disease severity. Key findings include:

Slice Count Analysis: On average, patients had approximately a minimum of 6.44 and a maximum of 11.16 significant slices. This variability highlights the diverse presentations of the condition.

Intensity Distribution: Most pixel intensities were concentrated in the lowest range $(0 - 25)$, primarily due to the background area outside the body, which has zero activity in the images. This low-intensity range reflects the predominance of non-active regions rather than the actual presence of low signal activity in the cancer cases. Only about 2% of pixels demonstrated high intensities, suggesting that highly active regions are rare, but this observation is primarily influenced by the nature of the imaging, with the background contributing significantly to the low-intensity pixels. Therefore, conclusions about the cancer cases should not be drawn solely based on this distribution.

Thresholding Performance: The use of various thresholding techniques showed flexibility in isolating Region of interest, effectively concentrating on higher-intensity areas relevant to clinical assessment.

U-Net Segmentation Model: The qualitative findings highlighted the necessity of normalizing data, augmenting it, and utilizing Dice coefficients for evaluating models,

leading to a reliable segmentation model.

6.2 Comparison with Literature

Our findings align with those of Liedes et al. [24], who reported comparable Dice scores (mean of 0.79) in segmenting PET images. Our results showed mean Dice coefficients ranging from 0.24 to 0.6 across various configurations, highlighting the ongoing challenges in improving segmentation accuracy in PET imaging. The variability observed in performance metrics, such as the mean and median Dice coefficients, indicates that while current models show promise, further refinement is needed to improve their clinical applicability. This study also highlights the importance of image preprocessing steps, such as normalization and augmentation, which previous research has demonstrated can significantly enhance model performance.

6.3 Analysis of Methodology and Results

The methodology employed in this research involved a robust framework for analyzing PET images, including thorough data normalization and the use of U-Net for segmentation. The iterative training process, complemented by early stopping based on validation loss, effectively minimized overfitting and ensured the model's generalizability. The use of multiple thresholding criteria and pixel count limits enabled the analysis to adapt to different case scenarios, further refining the model's performance.

Statistical tests, specifically focusing on Dice coefficients, confirmed the predictive accuracy of the segmentation model. The relationship between quantitative metrics and qualitative observations offered a comprehensive understanding of the model's effectiveness. There were clear correlations observed between higher Dice scores and more accurately defined segmentations in the qualitative analysis.

6.4 Limitations and Challenges

Despite the promising results, several limitations and challenges were encountered in this research:

Dataset Size: The study was limited to 100 patients, which may not fully represent the diversity of conditions observed in a broader population. A larger dataset would enhance the generalizability of the findings.

Thresholding Variability: The choice of thresholds and pixel limits introduced variability in the number of significant slices retained for analysis, potentially impacting the consistency of results across all the cases.

Model Complexity: The complexity of the U-Net model may require significant computational resources, potentially restricting its accessibility for widespread clinical use.

Quality of Input Data: The quality of PET images can vary due to numerous factors, including equipment calibration and patient movement, which could affect segmentation

accuracy.

Time-Consuming : One challenge in this research was the lengthy testing time of the Python code for certain cases, particularly when the Image dimension and number of epochs were high.

6.5 Suggestions for Future Research

Future research should aim to address the limitations identified in this study and explore new avenues for enhancing PET image analysis. Suggested areas include:

Expanding Dataset Size: Using a larger and more diverse dataset may enhance the model's robustness and its suitability across different patient demographics and conditions.

Refining Segmentation Techniques: Investigating alternative segmentation models or hybrid approaches that combine the strengths of different methodologies may yield better performance.

Exploring Advanced Augmentation Strategies: Additional research into advanced data augmentation methods, including more intricate transformations, could improve model generalization.

Longitudinal Studies: Conducting longitudinal studies to assess the efficacy of segmentation models over time may provide insights into the progression of conditions and the impact of treatment.

Integrating Clinical Context: Future studies should consider integrating clinical data with imaging analysis to develop more holistic predictive models that account for patient history and other relevant factors.

7 Conclusion

This research presents a detailed analysis of image processing results obtained from positron emission tomography (PET) images of 100 patients diagnosed with various conditions. By concentrating on significant image slices through robust quantitative analysis, we have emphasized important metrics, including slice counts, pixel intensity distributions, and thresholding performance. These findings provide valuable insights into the characteristics of the regions of interest (ROIs).

Our results indicate considerable variability in the number of significant slices across patients, influenced by different intensity thresholds and pixel count criteria. The intensity distribution analysis revealed that most image pixels were concentrated in lower intensity ranges, suggesting that regions with pronounced activity are less frequent yet clinically significant when identified.

In terms of segmentation, the U-Net model showed promise in delineating ROIs effectively, as measured by the Dice coefficient. However, statistical tests assessing the impact of augmentation techniques indicated that none—including Gaussian blur—led to a statistically significant improvement over the baseline without augmentation. In fact, the best Dice scores were obtained from the model trained without any augmentation.

This suggests that commonly used data augmentation techniques may not be sufficient or suitable for PET images, and future research should focus on more specialized or adaptive strategies to enhance segmentation accuracy.

Overall, this study contributes meaningful findings to the field of medical imaging, bridging quantitative analysis with model performance evaluation, and provides a basis for refining clinical segmentation workflows.

References

[1] Abadi M., Barham P., Chen J. et al.: *TensorFlow: A System for Large-Scale Machine Learning.* Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2016.

[2] Brett M., Markiewicz C.J., Hanke M. et al.: *nipy/nibabel: 3.2.1 (3.2.1)* (2020). Zenodo. https://doi.org/10.5281/zenodo.4295521

[3] Bradski G., and Kaehler A.: *Learning OpenCV: Computer vision with the OpenCV library.* O'Reilly Media, Inc., 2008.

[4] Cubuk E.D., Zoph B., Mane D., Vasudevan V., and Le Q.V.: *RandAugment: Practical Automated Data Augmentation with a Reduced Search Space.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), (2020).

[5] Cubuk E.D., Zoph B., Shlens J. and Le Q.V.: *AutoAugment: Learning Augmentation Policies from Data.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2019).

[6] Çiçek Ö., Abdulkadir A., Lienkamp S.S., Brox T., and Ronneberger O.: *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation.* In: Ourselin, S., Joskowicz, L., Sabuncu, M., Unal, G., Wells, W. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016. MICCAI 2016. Lecture Notes in Computer Science(), vol 9901. Springer, Cham.

[7] Dhawan Atam P.: *Medical image analysis*. John Wiley & Sons, 2011.

[8] Dice Lee R.: *Measures of the amount of ecologic association between species*. Ecology 26 (3) (1945), 297–302.

[9] Ekstrom M.P.: *Digital image processing techniques*. Volume 2, Academic Press, 2012.

[10] Elkashty O.A., Ashry R., and Tran S.D.: *Head and Neck cancer management and cancer stem cells implication*. Saudi Dent J. 31 (4) (2019), 395–416.

[11] Esteva A., Kuprel B., Novoa R.A., Ko J., Swetter S.M., Blau, H.M., and Thrun S.: *Dermatologist-level classification of skin cancer with deep CNNs.* Nature, 542(7639) (2017), 115-118.

[12] Fukushima, K.: *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*. Biol. Cybernetics 36, 193–202 (1980). https://doi.org/10.1007/BF00344251

[13] Ginneken B.: *A survey on deep learning in medical image analysis*. Medical Image Analysis 42 (2017), 60-88.

[14] Gonzalez R.C., and Woods R.E.: *Digital Image Processing*. 4th Edition, Pearson Education, New York, 1022 p, 2018.

[15] Goodfellow I., Bengio Y., and Courville A.: *Deep Learning*, MIT Press, 2016.

[16] Goodfellow I., Shlens J., and Szegedy C.: *Explaining and Harnessing Adversarial Examples*. CoRR (2014), abs/1412.6572.

[17] Gormley M., Creaney G., Schache A. et al.: *Reviewing the epidemiology of head and neck cancer: definitions, trends and risk factors.* Br Dent J 233 (2022), 780–786.

[18] Ghosh A., Jana N.D., Mallik S., and Zhao Z.: *Designing optimal convolutional neural network architecture using differential evolution algorithm.* Patterns, 3(9) 2022, 100567. https://doi.org/10.1016/j.patter.2022.100567

[19] He Q, Duan Y, Yang Z, Wang Y, Yang L, Bai L, and Zhao L.: *Context-aware augmentation for liver lesion segmentation: shape uniformity, expansion limit and fusion strategy.* Quant Imaging Med Surg. 2023 Aug 1;13(8):5043-5057.

[20] Hubel D.H., and Wiesel T.N.: *Receptive fields of single neurones in the cat's striate cortex*. J Physiol. 1959 Oct;148(3):574–91. doi: 10.1113/jphysiol.1959.sp006308

[21] Hunter J. D.: *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3)(2007), 90–95.

[22] Ioffe S., and Szegedy C.: *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* Proceedings of the 32nd International Conference on Machine Learning (ICML), (2015). arXiv:1502.03167v3

[23] Johnson D.E., Burtness B., Leemans C.R., Lui V.W.Y., Bauman J.E., Grandis J.R.: *Head and neck squamous cell carcinoma.* Nat Rev Dis Primers, 6:92 (2020).

[24] Liedes J., Hellström H., Rainio O., Murtojärvi S., Malaspina S., Hirvonen J., Klén R., and Kemppainen J.: *Automatic Segmentation of Head and Neck Cancer from PET-MRI Data Using Deep Learning.* J. Med. Biol. Eng. 43, 532-540 (2023). https://doi.org/10.1007/s40846-023-00818-8

[25] Kim P.: *Convolutional neural network. MATLAB deep learning.* New York, NY: Springer Science+Business Media New York 121–147, 2017.

[26] Kohavi R.: *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*. International Joint Conference on Artificial Intelligence, 1995. `https://api.semanticscholar.org/CorpusID:2702042`

[27] Krizhevsky A., Sutskever I., and Hinton G.E.: *ImageNet classification with deep convolutional CNNs*. Advances in Neural Information Processing Systems. Editor: F. Pereira and C.J. Burges and L. Bottou and K.Q. Weinberger. Curran Associates, Inc. 25, 2012.

[28] Lee G., and Fujita H.: *Deep Learning in Medical Image Analysis: Challenges and Applications*. Springer Cham, Cham, 2020.

[29] Litjens G., Kooi T., Bejnordi B.E., Setio A. A. A., Ciompi F., Ghafoorian M., Van Der Laak J.A., Van Ginneken B., and Sánchez C.I.: *A survey on deep learning in medical image analysis*, Med. Image Anal. 42 (2017) 60–88.

[30] Lundervold A.S., and Lundervold A.: *An overview of deep learning in medical imaging focusing on MRI*. Zeitschrift für Medizinische Physik. 29 (2019), 102-127,

[31] Mallik S., and Zhao Z.: *Graph- andrule-basedlearning algorithms: a comprehensive review of their applications for cancer type classification and prognosis using genomic data.* Briefings Bioinf. 21 (2020), 368–394.

[32] McConnell S.: *Code Complete: A Practical Handbook of Software Construction.* 2nd Edition, Microsoft Press, 2004.

[33] Murphy K.P.: *Machine learning: A probabilistic perspective*. MIT Press, Cambridge, MA, 2012.

[34] Rahman Q.B., Iocca O., Kufta K., and Shanti R.M.: *Global Burden of Head and Neck Cancer*. Oral Maxillofac Surg Clin North Am. 32 (2020), 367-375.

[35] Rainio O., Liedes J., Murtojärvi S. et al. *One-click annotation to improve segmentation by a convolutional neural network for PET images of head and neck cancer patients.* Netw Model Anal Health Inform Bioinforma 13, 47 (2024). https://doi.org/10.1007/s13721-024-00483-0

[36] Ren P., Xiao Y., Chang X., Huang P.-Y., Li Z., Chen X., and Wang X.: *A comprehensive survey of neural architecture search: challenges and solutions.* ACM Comput. Surv. 54 (2022), 1–34.

[37] Ronneberger O., Fischer P., and Brox T.: *U-Net: Convolutional Networks for Biomedical Image Segmentation.* In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham.

[38] Lecun Y., Bottou L., Bengio Y., and Haffner P.: *Gradient-based learning applied to document recognition*, in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278—2324, Nov. 1998, doi: 10.1109/5.726791

[39] Litjens G., Kooi T., Ehteshami Bejnordi B., Adiyoso Setio A.A., Ciompi F., Ghafoorian M., Jeroen A.W.M. van der Laak, van Ginneken B., and Sánchez C.I.: *A survey on deep learning in medical image analysis*. Medical Image Analysis 42 (2017), 60-88.

[40] Oliphant T.E.: *Guide to NumPy*, Vol. 1. USA: Trelgol Publishing, 2006.

[41] Rosenblatt F.: *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review, 65 (1958), 386–408.

[42] Sarkar D., Bali R., and Sharma T.: *Practical machine learning with Python: A problem solver's guide to building real-world intelligent systems.* New York, NY: Apress, Springer Science+Business Media New York, 2018. Schapire, R. E. (1990). The strength of weak learnability. Machine Learning, 5(2), 197–227.

[43] Sørensen T.: *A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons*. Kongelige Danske Videnskabernes Selskab. 5 (4) (1948), 1–34.

[44] Shen D., Wu G., and Suk H.I.: *Deep learning in medical image analysis.* Annual Review of Biomedical Engineering 19 (2017), 221-248.

[45] Shin H.C., Roth H.R., Gao M., Lu L., Xu Z., Nogues I., Yao J., Mollura D., and Summers R.M.: *Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning*. IEEE Trans Med Imaging 35 (2016), 1285-1298.

[46] Shorten C., and Khoshgoftaar T.M.: *A survey on Image Data Augmentation for Deep Learning*. J. Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

[47] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., and Salakhutdinov R.: *Dropout: A Simple Way to Prevent CNNs from Overfitting.* Journal of Machine Learning Research, 15 (2014), 1929-1958.

[48] Taylor L.,and Nitschke G.: *Improving deep learning with generic data augmentation*. in: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2018, pp. 1542–1547.

[49] Tzeng E., Hoffman J., Saenko K., and Darrell T.: *Adversarial Discriminative Domain Adaptation*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2962-2971, doi: 10.1109/CVPR.2017.316.

[50] Virtanen P., Gommers R., Oliphant T.E. et al. *SciPy 1.0: fundamental algorithms for scientific computing in Python.* Nat Methods 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[51] Wang J., and Perez L.: *The effectiveness of data augmentation in image classification using deep learning.* Convolutional CNNs Vis. Recognit, 11 (2017), 1-8.

[52] Wong S.C., Gatt A., Stamatescu V., McDonnell, M.D.: *Understanding data augmentation for classification: when to warp?* in: 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), IEEE, 2016, pp. 1–6

[53] Xu M., Yoon S., Fuentes A., Yang J., Park D.: *A Comprehensive Survey of Image Augmentation Techniques for Deep Learning*. Pattern Recognition 137 (2023), 109347.

[54] Xu M., Yoon S., Fuentes A., Yang J., Park D.: *Style-consistent image translation: a novel data augmentation paradigm to improve plant disease recognition.* Front. Plant Sci. 12: 773142, 2022.

[55] Zhang H., Cisse M., Dauphin Y.N., and Lopez-Paz D.: *mixup: Beyond Empirical Risk Minimization.* International Conference on Learning Representations (ICLR), (2018). arXiv:1710.09412v2

[56] Zhu H., Liu Y., Wen Y., and Zheng W.: *Data Augmentation in Deep Learning for Medical Image Segmentation*. A Review. Journal of Biomedical Informatics, 107 (2020), 103484.

[57] Ferdinand N.: *A simple guide to convolutional CNNs*. `https://towardsdatascience.com/a-simple-guide-to-convolutional-neural-networks-751789e7bd88`

[58] Tatan V.: *Understanding CNN (Convolutional Neural Network)*. `https://towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd626ee7d4`