5-13-2022

Improving deep neural network training with batch size and learning rate optimization for head and neck tumor segmentation on 2D and 3D medical images

Zachariah Douglas
zd187@msstate.edu

Improving deep neural network training with batch size and learning rate optimization for head and neck tumor segmentation on 2D and 3D medical images

By

Zachariah Douglas

Approved by:

Haifeng Wang (Major Professor)
William Neil Duggar (Committee Member)
Linkan Bian (Graduate Coordinator)
Jason M. Kieth (Dean, Bagley College of Engineering)

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Industrial and Systems Engineering
in the Department of Industrial and Systems Engineering

Mississippi State, Mississippi

May 2022

Name: Zachariah Douglas

Date of Degree: May 13, 2022

Institution: Mississippi State University

Major Field: Industrial and Systems Engineering

Major Professor: Haifeng Wang

Title of Study: Improving deep neural network training with batch size and learning rate optimization for head and neck tumor segmentation on 2D and 3D medical images

Pages of Study: 96

Candidate for Degree of Master of Science

Medical imaging is a key tool used in healthcare to diagnose and prognose patients by aiding the detection of a variety of diseases and conditions. In practice, medical image screening must be performed by clinical practitioners who rely primarily on their expertise and experience for disease diagnosis. The ability of convolutional neural networks (CNNs) to extract hierarchical features and determine classifications directly from raw image data makes CNNs a potentially useful adjunct to the medical image analysis process. A common challenge in successfully implementing CNNs is optimizing hyperparameters for training. In this study, we propose a method which utilizes scheduled hyperparameters and Bayesian optimization to classify cancerous and noncancerous tissues (i.e., segmentation) from head and neck computed tomography (CT) and positron emission tomography (PET) scans. The results of this method are compared using CT imaging with and without PET imaging for 2D and 3D image segmentation models.

Key words: head and neck cancer, convolutional neural networks, deep learning, batch size, learning rate, 3D segmentation, CT images, PET images

DEDICATION

This thesis is dedicated to my loving grandma, Renate Shuck. From this moment on...

ACKNOWLEDGEMENTS

I would like to graciously acknowledge several individuals for their support and contribution toward this thesis. Dr. Haifeng Wang, my advisor, thank you for your guidance and encouragement in this process. Your compassion and generous accessibility has been unwavering and much appreciated. Committee members, Dr. William Neil Duggar and Dr. Linkan Bian, thank you for your feedback and direction. Your consideration has helped shape and define my efforts. As a distance student, I express appreciation for my fellow student, Yibin Wang. You resolved on-site technical issues with such urgency that my work was never interrupted. Lastly, thank you Miki Diossy for being there for me every step of the way, in every way imaginable. I could not have asked for a better family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

AI Artificial Intelligence

ANN Artificial Neural Network

BOS Bayesian Optimization-derived Scheduling

CNN Convolutional Neural Network

CSV Comma-Separated Values

CT Computed Tomography

DSC Dice Score Coefficient

EHA Exploratory Hyperparameter Analysis

GPU Graphics Processing Unit

hyperparameter Any quantitative variable that affects model training behavior.

LOR-RAMLA Line of Response-Row Action Maximum Likelihood Algorithm

MRF Markov Random Field

MICCAI Medical Image Computing and Computer Assisted Intervention

MLP Multilayer Perceptron

OSEM Ordered Subset Expectation Maximization

PSCC Primary Squamous Cell Carcinoma

PET Positron Emission Tomography

ROI Region of Interest

SGD Stochastic Gradient Descent

SUV Standard Uptake Value

SVM Support Vector Machine

TCT Training Computation Time

CHAPTER I

INTRODUCTION

Medical imaging is commonly used by healthcare practitioners to detect and quantify malignant tumors. Traditionally, this is achieved by visually differentiating (segmenting) cancerous tissue from healthy tissue for a given image modality. This chapter introduces the task of using convolutional neural networks (CNNs) to facilitate image segmentation of head and neck tumors in computed tomography (CT) and positron emission tomography (PET) images.

1.1 Motivation

Approximately 600,000 new global cases of head and neck cancer are diagnosed annually, making it the fifth most commonly diagnosed cancer worldwide [5, 38]. Among these new cases, 40-50% result in death [38]. The prevalence and incidence of head and neck cancer have lead to a significant investment in how this disease is diagnosed and treated. A critical tool utilized in the diagnosis, treatment, and prognosis of head and neck cancer (as well as other cancer types) is medical imaging. Different medical imaging modalities allow healthcare practitioners to visualize specific physiological features. Among these features is malignant cancer tumors. The limitations of the analysis of these medical images reside in the practitioners experience and ability to detect cancerous tissue from noncancerous tissue. The practitioner is tasked with correctly classifying regions of a given image. This is known as segmentation (Figure 1.1).

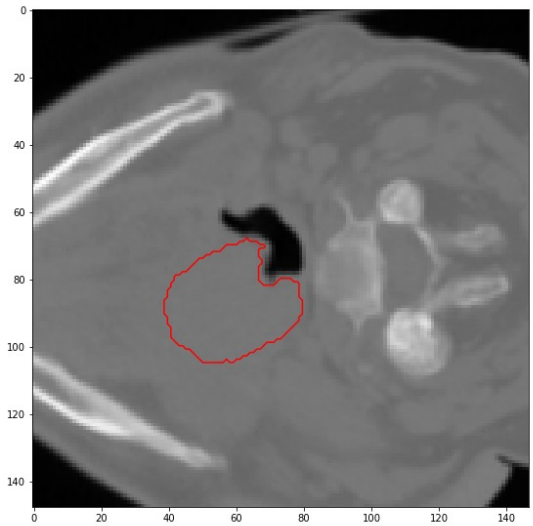

Figure 1.1

Cancer Tumor Segmentation of a CT Image

Deep neural networks, specifically CNNs, have widely been used for image classification tasks. The architecture of CNNs allow them to extract hierarchical features and learn models which can classify images pixel by pixel [33]. Image classification at this level of detail provides class detection as well as spatial information. The end result is a model which is able to predict "what" is present in a given image as well as "where" it is located. This capability makes CNNs a powerful tool which can be implemented to aid practitioners in the medical image analysis process.

1.2 Problem Statement

While medical practitioners can pose limitations due to experience and ability, CNNs possess limitations as well. The ability of a CNN to correctly segment a medical image relies heavily in how the CNN is trained. Several studies investigate varying aspects of training which lead to increased model performance for medical image segmentation [1, 18, 21]. Hyperparameter tuning

is among one of the most critical tasks when training a CNN model. Hyperparameters determine the specifications for how a model is trained and can affect the model's ability to generalize. Hyperparameters can be related to the network architecture itself, such as the number of layers, or can relate to the algorithms used to train the model (learning rate and batch size).

The learning rate is one of the most important hyperparameters to consider. With all other design choices held constant, studies show that an improperly tuned learning rate can make the difference between exceptional or poor model performance [23]. Batch size is another hyperparameter which can be tuned, with or without consideration of the learning rate [51].

A common approach of refining CNN training is scheduling hyperparameters during training (e.g., decaying the learning rate) [51]. The logic behind decaying the learning rate is that at the start of training, a larger learning rate is more beneficial to successfully navigate the total relevant space of the loss function (avoiding local minima) [59]. Conversely, assuming that the optimizer algorithm is converging to a minimum, smaller learning rates should be implemented later in training to ensure continued convergence and avoid overshooting.

This explanation, however, is by no means definitive. You et al. suggest that the benefits from large initial learning rates are seen instead because large learning rates avoid memorization of noise [59]. Smith et al. found that decreasing the learning rate throughout training was quantitatively equivalent to increasing the batch size [51]. Additionally, several studies support that the learning rate and batch size are in fact intimately connected [20, 25, 51, 52]. This could help explain why there is some debate about the mechanism behind the benefits of decaying the learning rate. Supporting the interconnection between batch size and learning rate, He et al. stress the importance of considering the batch size to learning rate ratio in order to generalize well [20, 25].

Although previous literature supports the idea that an optimal batch size to learning rate ratio exists, no such ratio has been empirically identified and used in training [20, 25, 51, 52]. Further, it is unclear if an optimal batch size to learning rate ratio exists for optimizers other than stochastic gradient descent (SGD). If batch size to learning rate ratio is a more accurate lens in which to view hyperparameter scheduling, then what is actually relevant is that training is scheduled from a low ratio to a higher one.

The problem being addressed in this study is that some de facto methods tune the learning rate and batch size independently of each other. Failure to recognize the interdependence of the learning rate and batch size may result in suboptimal decision making when training deep neural networks.

1.3 Assumptions

Several assumptions are made for this study:

- One of the most significant assumptions is that the labels used to train the model are in fact correct. The data used for this study is sourced by the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). The images originate from Vallières et al., were used by Andrearczyk et al., and were then re-annotated by an expert for the MICCAI 2020 Head and Neck Tumor Segmentation Challenge [5].

- The medical images used in training are limited to CT and PET images of the head and neck region in Gnu zipped file format.

- Each image used in training is the resultant of an axis-aligned bounding box used to crop the original image. Not to be confused with a minimum bounding box, the bounding boxes are

the same dimensions for each image. The cropped images contain the centered tumor along with a sufficient amount of proximal tissue to ensure proper training.

- The TensorFlow Python API version 2.6.0 is used to facilitate development and training of the CNN models used for segmentation.
- Processes are executed on a NVIDIA GeForce RTX 3090 GPU.
- Hyperparameters are optimized using a package titled Bayesian Optimization, which is a pure Python implementation for constrained global optimization using Bayesian inference.
- The Hyperparameter optimization used in this study is exclusively used for medical image segmentation tasks.

1.4 Research Objectives

The batch size to learning rate ratio is a derived hyperparameter which provides a new context for interpreting CNN model performance. Within the context of batch size to learning rate ratios, decaying the learning rate and increasing the batch size are effectively doing the same thing - increasing the batch size to learning rate ratio.

The apparent negative correlation between generalization ability and batch size to learning rate ratio suggests that smaller batch size to learning rate ratios are preferred [20]. However, intuition reveals that there is a lower limit to how small this ratio should be. Decreasing the batch size to learning rate ratio excessively will select for large learning rates that prohibit convergence.

The primary objective of this research is to improve deep neural network performance by optimizing both the learning rate and the batch size (optimizing the batch size to learning rate

ratio). Secondary objectives are to validate the presence of optimal batch size to learning rate ratios when using the Adam optimizer and to empirically test whether scheduling benefits can be realized with ratio scheduling.

1.5 Contribution

There are several challenges which affect successful implementation of CNN models for medical image segmentation. One of the most well known challenges is the fine tuning of hyperparameters [10, 59]. Several hyperparameter optimization methods exist to streamline the process of hyperparameter tuning. Among these methods is Bayesian optimization. Bayesian optimization updates a probabilistic model of the loss function over multiple iterations to determine hyperparameters which will minimize the loss function.

Previous studies have also investigated whether altering hyperparameters during training could yield increased model performance. Although a significant amount of literature exists for scheduling learning rates, less research has been conducted with respect to scheduling other hyperparameters such as batch size. At the time of this study, there are no known studies that explicitly optimize the batch size to learning rate ratio. Moreover, the scheduling of this ratio has not been investigated.

In this study, we implement a new method which combines Bayesian hyperparameter optimization with scheduled training. Two different sets of hyperparameters are used for each schedule component. For each schedule component, Bayesian optimization is used to determine the batch size and the learning rate. This Bayesian optimization-derived scheduling (BOS) is compared

against unscheduled models using SGD and Adam for both CT and CT/PET images. Methods are further compared with respect to 2D U-Net and 3D U-Net implementation.

1.6 Thesis Overview

The structure of this thesis is as follows: Chapter 2 covers the fundamentals of supervised machine learning, then reviews the literature for machine learning-based segmentation, CNNs in medical image segmentation, the relationship between learning rate and batch size, scheduled learning, and Bayesian hyperparameter optimization. Chapter 3 details the methodology used in this study, including the basics of deep learning, the neural network architecture used, the Dice score coefficient and Dice loss, the mechanism of Bayesian optimization, and Bayesian optimization-derived scheduling. Chapter 4 describes the Experimental Results and provides an analysis and discussion of results. Chapter 5 provides Conclusions and directions for future work.

CHAPTER II

LITERATURE REVIEW

This chapter presents the information required to understand the basic fundamentals of supervised machine learning. Then, a literature review of machine learning-based image segmentation is provided. The literature review covers non-deep learning approaches and compares their advantages and disadvantages. Then, an overview of deep learning segmentation methods using CNNs is given. The relationship between learning rate and batch size and the implementation of Bayesian hyperparameter optimization is discussed as well.

2.1 Supervised Machine Learning Fundamentals

Artificial Intelligence (AI) efforts date as far back as the 1950's when Alan Turing developed the Turing Test to evaluate how well a machine could imitate a human [8]. Since then, further developments in the field of AI resulted in the sub-field titled machine learning [8]. The overall idea of machine learning is for a machine to take data and automatically make decisions based off of this data. With every decision, the machine "learns" and will make better subsequent decisions [8].

2.1.1 How Training Works

Machine learning starts with a model [8]. This model receives an input and will return an output or prediction [13]. In the case of supervised machine learning, there is correct output data for all input data. It is then the machine's job to iteratively alter the model so that the model can best predict future input data [42]. The assumption is that there is a pattern that can be learned from the amount of training data used to create the model [13]. In order for reliable patterns to be learned, there needs to be a sufficient amount of data and there needs to be a way to quantify how well a model performed throughout its training process (i.e., the correctness of its predictions) [8, 42]. This performance is measured by a loss function which returns a specific amount of error when given the model's prediction and the correct prediction [42].

2.1.2 Loss Function Optimization

Machines alter the model towards better predictions by use of the loss function. What the machine is altering during this process is the parameters of the model [8, 42]. Since the loss function is defined in terms of these same parameters, the minimum value of the loss function will correspond to the model parameter values which yield the smallest error [13]. As a result, optimizing the loss function will provide the "best" model.

2.2 Machine Learning-based Image Segmentation

Several traditional machine learning algorithms have been used for image segmentation. Each algorithm presents a different approach with its own unique considerations. The literature regarding clustering, random forests, support vector machines, and Markov random fields is reviewed to summarize machine learning-based image segmentation.

2.2.1 Clustering

Clustering is known as a type of unsupervised machine learning. In unsupervised machine learning, input image data is not associated with any correct labeling (the correct output of the image is unknown) [41]. In supervised machine learning, this labeling provides the machine with information needed in order to learn patterns. In the case of clustering, patterns need to be learned a different way [41].

K-means clustering is a common algorithm used to group or *cluster* data without the use of labels [11]. The K-means algorithm achieves this by use of quantitative data features [11]. Since the data features are quantitative, the "closeness" of a feature against another feature can be measured by the Euclidean distance between these two features [11, 41]. Regions of an image are then segmented according their respective cluster. Theoretically, each cluster will contain pixels that are more similar to pixels of their same cluster while being more different than pixels of each alternative cluster [11, 41].

2.2.2 Random Forests

A random forest is an ensemble method which uses multiple decision trees to achieve a prediction [27]. Each tree contains different criteria for making a particular set of binary decisions with respect to the data [27, 48]. After these decisions, a final decision (the prediction) is made. Since each tree contains a different set of decisions, each one may arrive to a different conclusion [27, 48]. Random forests are used to address this variation [27]. In a random forest, many trees are constructed, and the final prediction is made based on what the majority of decision trees predicted, given the same input data [27].

2.2.3 Support Vector Machines

A significant challenge in classification tasks is that separability may not exist for original input data [48, 60]. This lack of separability makes classification difficult as there is no possible hyperplane which can separate the data in its given dimensional space [60]. A key concept in Support Vector Machines (SVMs) is that the input data can demonstrate separability if the data is transformed into a higher dimensional representation (Figure 2.1) [32, 60]. Kernel functions perform this type of transformation [60].

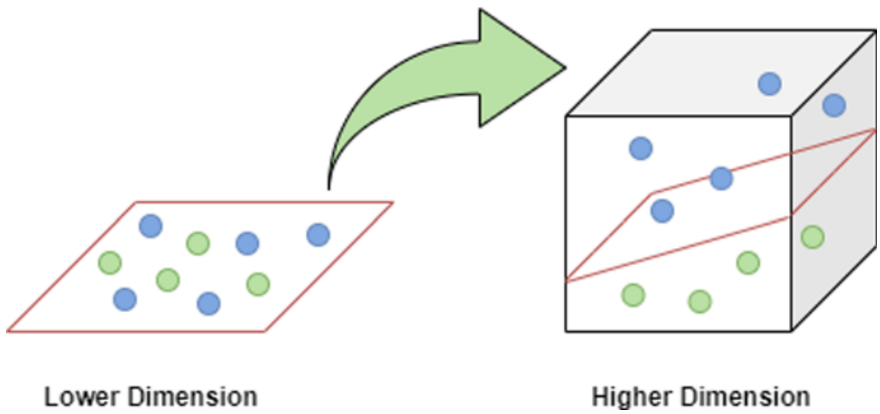

Figure 2.1

How SVMs transform data to higher dimensions, leading to separability [32]

2.2.4 Markov Random Fields

Markov random fields (MRFs) use conditional probability to produce image segmentations [48]. Since adjacent pixels are likely to be similar, the probability of a given pixel classification is affected by neighboring pixels [48]. In MRFs, pixels are grouped together as objects [24, 48]. The

label for an individual pixel is determined based on its similarity between a given object and its neighboring objects [24].

2.2.5 Image Segmentation Model Comparisons

Each machine learning approach has its own advantages and disadvantages. The breadth of continuing literature is often a direct result of attempting to improve upon a currently established segmentation methodology [40]. Table 2.1 summarizes these different approaches and their respective advantages and disadvantages.

A key parameter that the K-means algorithm requires is the number of clusters, k [11, 41]. For the image predicted, the number of classes which are to be identified must be known in advance [41]. Successful implementing K-means clustering for image segmentation also requires appropriate initialization of the cluster, a task which can be considerably challenging [11]. As a result, K-means clustering has been useful for clustering similar features together but has been less successful for image segmentation [60].

Another criticism of K-means clustering has been its all-or-nothing nature of classification [60]. In K-means clustering, each pixel belongs to a singular cluster (full membership to one cluster) [11, 41]. Fuzzy C-means clustering allows for partial membership which has made it more suitable for real-world tasks [56]. The significant disadvantage of this partial membership application is that membership functions are particularly difficult to determine [56]. Additionally, since both K-means and fuzzy C-means clustering are unsupervised, they rely on intraclass variation being low and interclass variation being high to segment images properly, which is not always the case

[41, 60]. Figure 2.2 shows how fuzzy C-means can segment a brain image when regions of the image are relatively homogeneous [58].

SVMs have been used for image segmentation as well but notably lack robustness, often times failing to classify pixels when significant noise is present [46, 60]. Although reasonable segmentation results have been achieved on select datasets using SVMs, these models are often not pure SVM implementations [46]. Instead, additional algorithms or models are used in conjunction with an SVM to support the segmentation process [46].

Hartmann et al. found that a random forest model performed well for medical image segmentation, demonstrating a Dice score coefficient (DSC) value of 0.85 [19]. In other cases, random forests can be limited depending on how similar regions of interest are to their surroundings [35]. One advantage that was noted for random forests was that they could be implemented when Graphics Processing Units (GPUs) are not available [19]. In this case, random forests could be used in smaller IT infrastructures given that there is sufficient hardware memory to produce optimal image segmentation [19].

MRFs have been implemented to segment images with success on specific image types (e.g., brain MR images and mammograms) [22, 55]. In MRFs, filters are used to extract features from image data, but unlike other methods, these filters are not learned and must be carefully designed and/or selected [48]. This inability to learn filters has the potential to be fairly limiting. Held et al. highlighted three filters which were used for image segmentation, non-parametric distributions of tissue intensities, neighborhood correlations, and signal inhomogeneities [22].

Table 2.1

Advantages and Disadvantages of Image Segmentation

Approach	Authors (year)	Advantages	Disadvantages
K-means	Zhang et al. (2016)	• Suitable for grouping/clustering	• Inaccurate segmentation
Fuzzy C-means	Jasim and Mohammed (2021)	• Can be useful for real-world problems due to partial membership	• Membership function is difficult to determine
SVM	Zhang et al. (2016)	• Inherently a classification method	• Not sensitive to noise • Inexact segmentation
SVM	Rajan et al. (2017)	• Good results on Berkeley segmentation database • Competitive performance on color images	• Lacks robustness to noise • Required implementation of fuzzy C-means
RF	Balsiger et al. (2015)	• Duration suitable for clinical use • Reports "feasible" segmentation	• Cannot compete with state-of-the-art methods • Only segments distal femur
RF	Liu et al. (2019)	• High accuracy segmenting lung nodules • Fully automatic	• Difficulty with lung nodules stuck together
RF	Hartmann et al. (2021)	• Competitive F1 score performance • Does not require GPU	• Requires large amount of hardware memory
MRF	Held et al. (1997)	• Performed well on 3D brain MR images	• Only considers 3 features • Not tested for classes with similar signal intensity
MRF	Venmathi et al. (2019)	• Accurately segmented breast calcifications • Can segment breast from pectoral muscle	• Advanced segmentation difficult without classifiers

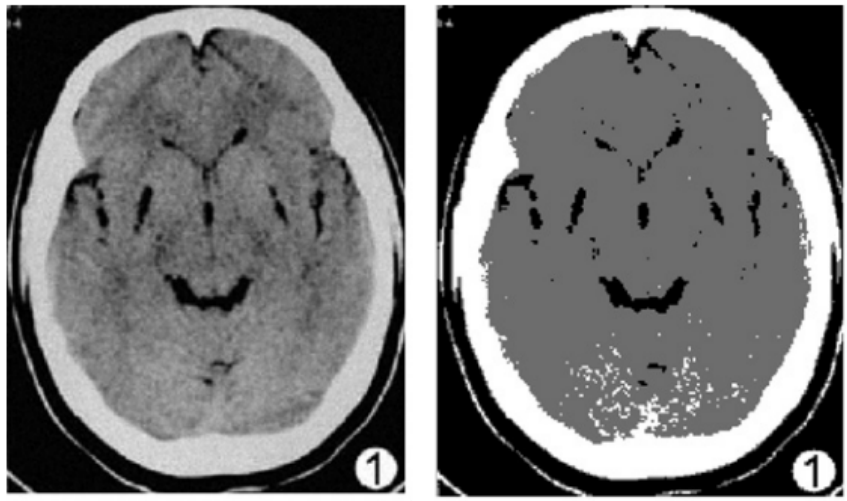

Figure 2.2

Original Brain Image and Fuzzy C-Means Segmentation [58]

Alternatively, it is not uncommon for CNNs to have hundreds of filters which are learned and therefore presumed to actively contribute to segmentation processes [40, 48].

2.3 CNNs in Medical Image Segmentation

CNNs are among the most common and successful architectures for computer vision problems, making them particularly suitable for medical image segmentation [40]. Several studies have implemented various CNNs to perform image segmentation on both normal and pathological anatomical structures (Table 2.2).

Implementation of 3D V-Net CNN models has been used to segment different organs such as bladders, rectums, femurs, prostates, and various head organs [12, 39, 47]. Disease regions of interest have been segmented as well, such as cancer and liver and kidney disease [1, 5, 15, 49].

Table 2.2

Medical Image Segmentation Literature Using CNNs

Authors (year)	Modality	Region of Interest	Model	DSC[1]
Abdelhafiz et al. (2020)	X-Ray	Cancer (breast)	2D U-Net	0.951
Andrearczyk et al. (2020)	PET-CT	Cancer (head/neck)	2D V-Net	0.606
Duanmu et al. (2020)	CT	Various Organs (head)	3D V-Net	0.82
Golla et al. (2020)	CT	Arteries (abdomen)	2D U-Net	0.822
Gonella et al. (2019)	MRI	Cancer (brain)	3D V-Net	0.641
Milletari et al. (2016)	MRI	Prostate	3D V-Net	0.869
Savenije et al. (2020)	MRI	Bladder, Rectum, Femur	3D V-Net	0.97
Shin et al. (2020)	CT	Liver/Kidney Disease	3D V-Net	0.961
Tan et al. (2019)	MRI	Prostate	3D V-Net	0.646

[1] Highest DSC is shown for results with multiple regions of interest

Table 2.2 shows that the differences in performance can vary significantly, with DSC values as low as 0.606 for head and neck tumor segmentation, to 0.97 for femurs [5, 47].

The extent to which a model can segment a medical image can be inferred somewhat qualitatively. Visually, MR images in Savenije et al. show that femurs are clearly defined with distinct boundaries rather than blending within the background (Figure 2.3) [47]. This observation can be seen in Abdelhafiz et al. as well [1]. Mammography images use X-rays to produce images with high contrast, helping to define clear lesion boundaries [1]. Lower DSC values are seen in cases where regions of interest are visually more difficult to distinguish from their surroundings (Figure 2.4) [5, 15].

Andrearczyck et al. performed automatic head and neck tumor segmentation using 2D and 3D V-Net CNNs [5]. Both CT and PET images were used to demonstrate the benefits of utilizing both modalities together [5]. Andrearczyk et al. implemented an early fusion approach where CT and

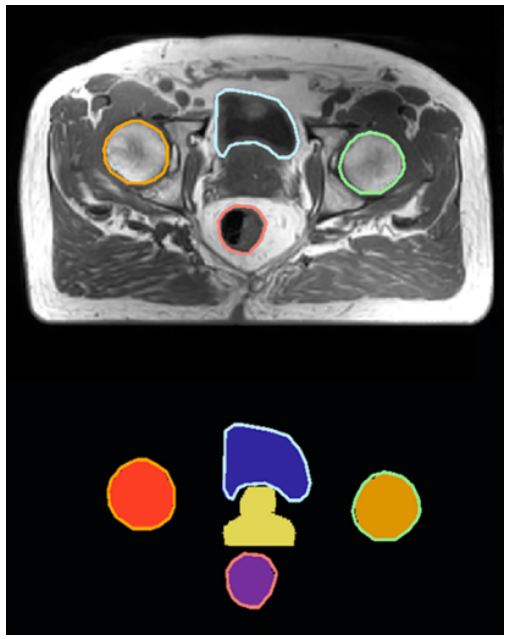

Figure 2.3

Femur, Rectum, and Prostate Segmentation [47]

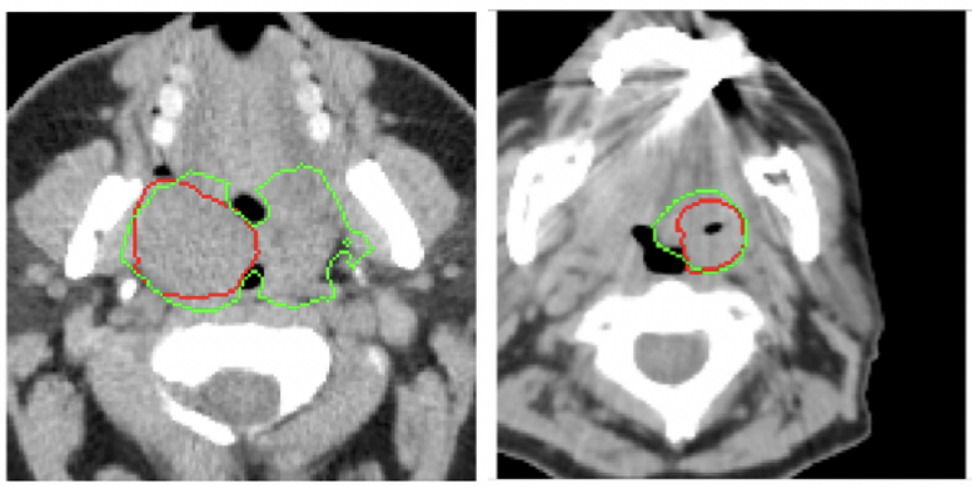

Figure 2.4

Axial CT Image (head), Predicted Segmentation (green), Correct Segmentation (red) [5]

Table 2.3

Results of Andrearczyk et al. [5]

Model	Modality	DSC	Precision	Recall
2D/3D	CT	48.7%/49.2%	52.7%/48.6%	54.1%/65.0%
2D/3D	PET	58.2%/58.6%	59.7%/59.1%	66.7%/70.2%
2D/3D	early fusion	58.5%/58.9%	58.1%/59.0%	70.2%/70.8%
2D/3D	late fusion	60.6%/59.7%	69.4%/62.8%	62.1%/69.1%

PET represented multiple input channels for the model and a late fusion approach by averaging probabilities for individual CT and PET models [5]. Their results can be seen in Table 2.3.

2.4 Learning Rate and Batch Size Relationship

Studies have demonstrated an important relationship between learning rate and batch size [20, 25, 51]. This relationship has primarily been evaluated for SGD [20, 25]. Jastrzębski et al. concluded that scheduling the batch size is a suitable alternative to scheduling the learning rate [25]. Smith et al. investigated scheduling batch size further, showing that scheduling the batch size resulted in the same test accuracy with equal epochs and less parameter updates compared to scheduling the learning rate [51]. They showed that fewer parameter updates lead to greater parallelism and less training time when training with SGD, Momentum, and Adam optimizers [51].

Goyal et al. found that equal generalization could be achieved with different batch sizes as long as the learning rate was scaled linearly [16]. To further explain this finding, Jastrzębski et al. discretized SGD as a stochastic differential equation to determine what factors influence the final minima learned from SGD [25]. Their work theoretically explains and experimentally supports that a higher learning rate to batch size ratio correlates with a wider minima and greater generalization

[25]. This supports the work of He et al., where 1,600 models were trained using CIFAR-10 and CIFAR-100 datasets [20]. The primary finding for this study was that the batch size to learning rate ratio was negatively correlated with generalization ability [20]. Figure 2.5 shows the results of this negative correlation [20]. Both theoretical and empirical evidence was provided to support this finding [20].

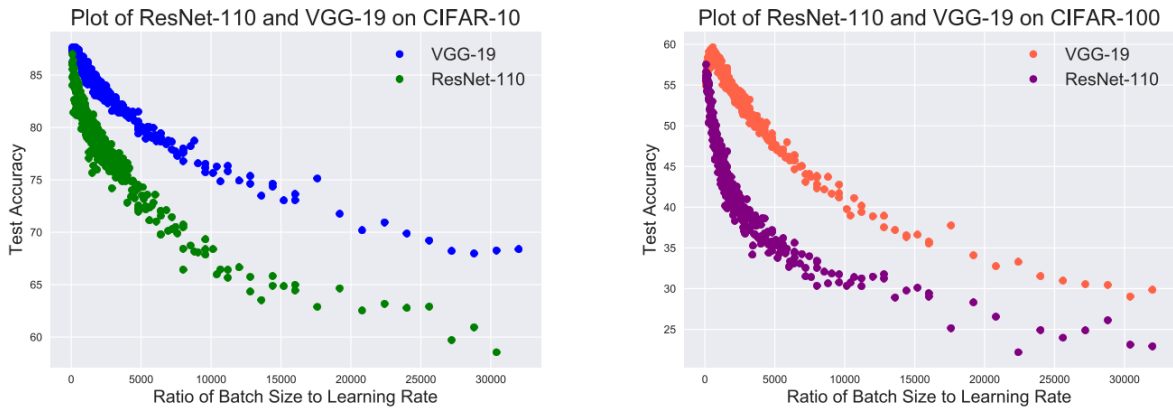

Figure 2.5

Negative Correlation of Batch Size to Learning Rate Ratio and Test Accuracy [20]

2.5 Scheduled Learning

Hyperparameters do not have to remain static during training. Several neural network training strategies involve scheduled learning, where different hyperparameters are used at different stages in the training process [2, 28, 51]. Several implementations have focused on scheduling the learning rate [28, 51, 59]. Although studies have suggested that scheduling the batch size could yield

identical, and in some respect, better results than scheduling the learning rate, this implementation remains scare within the literature [51].

Scheduling of optimizers has also been of interest [2, 28]. Since different optimizers are believed to accomplish different benefits, particularly with respect to early training versus late training, starting with a particular optimizer then switching to another has been explored [28]. Keskar and Socher found that they were able to switch from Adam to SGD and obtain the generalization performance of SGD while also retaining rapid initial progress properties from Adam [28]. Akiba et al. switched optimizers in a similar way, opting to use root mean square propagation as a "warm-up" to overcome difficulties associated with the beginning of training [2].

Several different methods such as grid search, random search, Bayesian optimization, and genetic algorithms have been developed to optimize hyperparameters for deep learning [3, 14, 26, 31]. Gao and Ding concluded that Bayesian hyperparameter optimization was a more stable method than grid search and random search [14]. Kunang et al. were able to increase the test accuracy of their intrusion detection model from 0.8233 to 0.99991 by using Bayesian optimization to fine tune hyperparameters [31]. In order to predict real estate prices, Kalliola et al. were able to increase the relative mean error of their deep artificial neural network (ANN) model by 2.5% by using Bayesian hyperparameter optimization. [26]. Other studies show different results. Alibrahim and Ludwig compared grid search, a Bayesian algorithm and a genetic algorithm and found that each method performed similarly, but the genetic algorithm produced the best results [3]. The Bayesian algorithm resulted in a DSC value of 0.81 while grid search and the genetic algorithm values were 0.85 and 0.87 respectively [3]. Differences in accuracy were less, with the genetic

algorithm leading with an accuracy of 0.9059, grid search resulting in 0.8976 and the Bayesian algorithm at 0.8959 [3].

CHAPTER III

METHODOLOGY

This chapter first covers the basics of deep learning. Once basics have been established, the methods used to improve CNN training are described, including the network architecture, the loss function used for this task, and how Bayesian hyperparameter optimization is combined to implement scheduled training.

3.1 Basics of Deep Learning

Deep learning is a subset of machine learning which has made significant advancements in nearly every application domain [4, 13]. Deep learning primarily utilizes neural networks as computational models to achieve state of the art results in areas such as image classification, autonomous vehicles, and speech recognition [4]. This section describes an overview of the basics of deep learning, covering elements of deep learning and mechanisms explaining how neural networks are able to learn.

3.1.1 Perceptrons

Perceptrons are the functional units which make up neural networks [13]. Figure 3.1 shows a diagram of a perceptron and the operations within it that transform data inputs into a single output value. Data inputs are multiplied by their respective parameters and then summed. The activation

function will then receive this summation and return the final output. A bias term (denoted in Figure 3.1 as w_0) is often added in the summation to provide greater output flexibility via horizontal shifting of the activation function [4]. Similar to more traditional machine learning methods, a perceptron is a model which receives input data and will output a prediction. In the context of classification tasks, this prediction is the predicted class of the input data [4, 8].

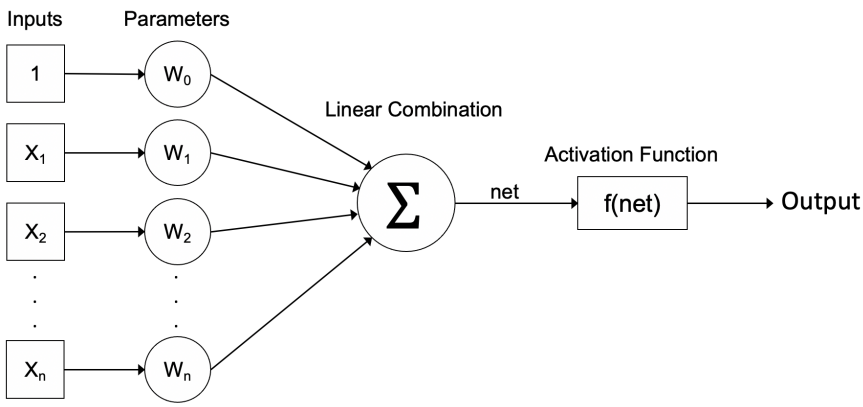

Figure 3.1

Perceptron Diagram

3.1.2 Neural Networks

Single perceptrons can be used to classify linearly separated data quite well [8]. In most cases, however, the data is not linearly separable, and a nonlinear model is required [13]. Connecting perceptrons so that the output of one perceptron is given to the next perceptron as input can model this nonlinearity [8, 13]. Connecting perceptrons in this way is how more general ANNs are formed, specifically, feedforward multilayer perceptron (MLP) neural networks [8]. Multiple perceptrons that receive the same input are referred to as layers (also *hidden* layers) [4]. These

layers will then send their respective outputs to subsequent hidden layers until the last layer yields the prediction [4, 13]. Deep neural networks simply obtain their name by consisting of enough layers to be considered "deep" [8].

3.1.3 Convolutional Neural Networks

CNNs are a type of deep neural network that has been particularly successful in image classification applications [1, 4, 13, 37]. From a high level perspective, the concepts related to perceptrons and MLPs hold true for CNNs - groups of functional units receive input data, then pass their outputs to other functional units via a network until a final output is achieved [8]. CNNs, however, contain several structural differences which help explain their performance success [13].

3.1.3.1 Kernels

For perceptrons, the first operation is that input data are multiplied by parameters. In the case of MLPs, this process will occur for every perceptron in the first hidden layer [8]. This process also occurs within CNNs, albeit in a slightly different yet equally systematic way. This difference can largely be explained by understanding kernels and their role within the CNN structure [4, 8, 13].

In Figure 3.1, the input of the activation function is simply the dot product of the input data vector and the vector of corresponding parameters. While Figure 3.1 depicts this dot product linearly, if the number of data inputs is a perfect square, we can just as easily depict this same dot product in the form of a square grid [4, 13].

The size of this square grid is typically 3x3 or 5x5 and is referred to as a kernel. The kernel itself is a grid organization for a specific set of parameters [13]. In 2D images, data is represented as a grid of pixels and each pixel contains a numerical value which will determine the exact color

that will be emitted [10]. The kernel size is intentionally smaller than a typical image size so that one kernel (with the same parameters) can be applied over each portion of the image systematically [4].

3.1.3.2 Convolutional Layers

When kernels are applied over respective regions of an input image, dot products between the input data and kernel parameters are fed as input to activation functions, not unlike what is seen for perceptrons (Figure 3.1) [4, 13]. The output of the activation function produces values which make up an activation map [13]. An individual activation map is a different representation of the input image which is why kernels are also known as filters [13].

Consider input data X and kernel K of dimensions (wxw). Applying the kernel will output the activation map, M, where

$$M_{i,j} = \sigma(\sum_{i'=1}^{w}\sum_{j'=1}^{w} X_{i+i'-1,j+j'-1} \cdot K_{i',j'}) \tag{3.1}$$

$M_{i,j}$ is the portion of the activation map which is a result of applying the kernel K to a section of the input data $X_{i,j}$, where (i, j) is the starting point and σ is typically a nonlinear activation function. A stride parameter s is set for how the kernel will slide across the image. Sliding for $s = 1$ will result in the kernel being applied to the image with every pixel being a starting point (i, j), $s = 2$ will perform operations so that (i, j) will be every other pixel, and so forth. For whichever s, the kernel is slid across the entire image resulting in a complete activation map, M.

Applying different kernels or filters to an image will produce different activation maps [13]. When stacked, these 2D activation maps create a volume known as a convolutional layer [13]. The term convolution refers to the operation of sliding the kernel over the entire input image to

collect each dot product [13]. Although this sliding dot product operation is technically a cross correlation, for the purposes of training a CNN, calculating the cross correlation and calculating the convolution will not produce any practical differences in the model's ability to learn or perform [13].

3.1.3.3 Pooling Layers

Kernels and convolutional layers explain how a model can receive image data and apply different parameters to obtain alternative image representations [4, 8]. Different image representations will emphasize different image features [8]. Identification and utilization of these different features are what enable the model to make successful predictions [4, 13].

The ability to make predictions is increased by a CNN's ability to identify different levels of features [4, 13]. Low level features such as lines and edges can be identified with the earlier convolutional layers, but in order to detect higher level features, pooling is introduced [4]. A common approach is known as max pooling, which considers a square grid of the activation map (similar to a kernel), and returns the maximum value of the grid [6, 13]. If the square grid is (2x2), then the result would be X_{max}, where

$$X_{max} = max\{X_{i,j}, X_{i+1,j}, X_{i,j+1}, X_{i+1,j+1}\} \tag{3.2}$$

X_{max} is passed to a separate layer known as the pooling layer [13]. This type of pooling downsamples the original input data, reducing the image size, while also allowing for extraction of higher level features and is illustrated in Figure 3.2 [4, 13].

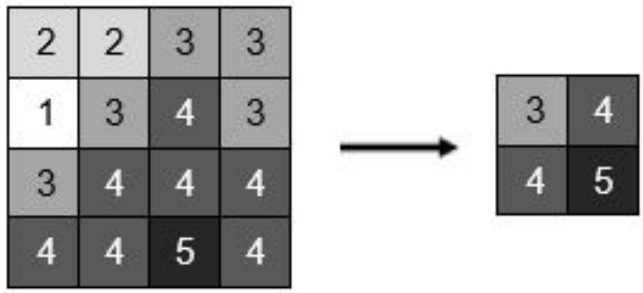

Figure 3.2

Maxpooling Operation with a Stride of $s = 2$

3.1.4 Backpropagation and Optimizers

With any deep neural network, the fundamental goal remains the same. Given a model of parameters, optimize parameter values so that the model will result in the smallest error (or loss) between the model's output and the given training data [8]. Chapter 2.1.2 introduced the optimization of a loss function. More sophisticated models require specific methods to achieve this optimization [4, 8].

In contrast to more rudimentary functions, it is not feasible to differentiate loss functions of deep neural networks and simply compute parameters which correspond to the global minimum [48]. These loss functions are complex and often times non-convex [13]. Instead, once a set of parameters is defined, an algorithm is used to determine a subsequent set of parameters which result in a decreased value of the loss function [8]. Determining subsequent sets of parameters utilizes the backpropagation algorithm and optimizers [4, 13, 48].

In backpropagation, the chain rule is used to determine how a specific change in one parameter will effect the end value of the loss function [4, 13]. Figure 3.3 shows a simple schematic of a neural network to help track the backpropagation process in the following equation,

$$\frac{\partial L(\hat{y}, y)}{\partial w} = \frac{\partial L(\hat{y}, y)}{\partial y} \frac{\partial y(h)}{\partial h} \frac{\partial h(s)}{\partial s} \frac{\partial s(w)}{\partial w} \tag{3.3}$$

In order to determine how a change in w will affect the change in the loss function $L(\hat{y}, y)$, the change in the loss function is computed with respect to the predicted output, $\hat{y}$ (recall in Chapter 2.1.1 that the loss function accepts both the model prediction and correct output as variables). After this, the chain rule is applied "backwards" through the network [13, 48].

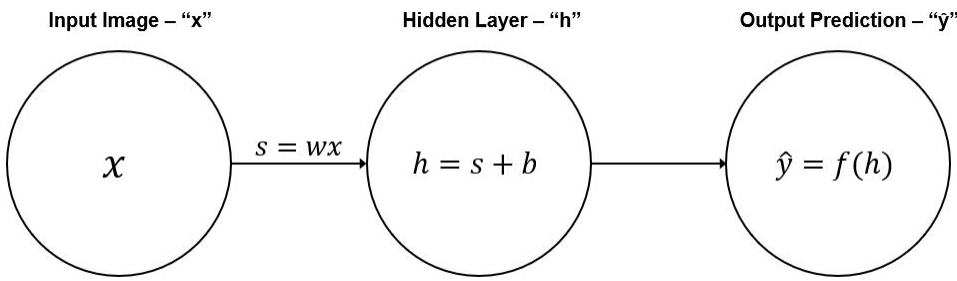

Figure 3.3

Backpropagation: Parameter (weight) w influences output $\hat{y}$

Backpropagation helps determine how *each* parameter affects the loss function. However, how exactly the parameter is updated in each iteration can vary [4, 48]. The algorithms which determine these parameter updates are known as optimizers. Different optimizers often have much in common, with newer optimizers building off of previous ones to help increase overall performance [4, 13, 48]. One of the most basic optimizers is gradient descent (or batch gradient

descent), which has since been divided into SGD and minibatch gradient descent [25]. SGD and the relatively recent Adam optimizer has lead to promising results in various deep learning applications [1, 5, 49, 54].

3.1.4.1 Gradient Descent

Gradient descent addresses two important considerations when updating model parameters, should parameters increase or decrease, and if so, what should be the magnitude of this increase or decrease [25, 50]. Since the gradient of the loss function will represent the direction for which the loss function will increase the most, the negative of the gradient is used to provide the direction of steepest descent [4]. In gradient descent, the magnitude of parameter change with respect to this direction is manually selected and is termed as the learning rate [59]. In its simplest form,

$$w = w - \eta \nabla L(\hat{y}, y) \tag{3.4}$$

where w is a parameter or weight, η is the learning rate, and $\nabla L(\hat{y}, y)$ is the gradient of the loss function.

3.1.4.2 Gradient Descent Variants

To appreciate gradient descent variants, Equation 3.4 needs to be expanded upon. A set of parameters and multiple samples of input data must be considered. Here we will consider w as a vector of parameters used in the model F to make a single prediction $\hat{y}^{(i)}$, for a single sample $x^{(i)}$ (Equation 3.5).

$$\hat{y}^{(i)} = F(w, x^{(i)}) \tag{3.5}$$

$$w_{t+1} = w_t - \eta \cdot \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(\hat{y}^{(i)}, y^{(i)})}{\partial w} \tag{3.6}$$

Equation 3.6 shows the parameter update process when considering *every* training sample before performing an update. This is known as batch gradient descent [13]. The resultant change in loss with respect to current parameters is averaged for all N samples. Conceptually, we see that no one sample will dictate the direction of the update [4, 13].

One update in batch gradient descent requires computing the gradients for all samples which can be impractical for larger datasets [13, 48]. In direct contrast, SGD performs parameter updates for each training sample i,

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L(\hat{y}^{(i)}, y^{(i)})}{\partial w} \tag{3.7}$$

Changing the direction of parameter updates for each sample can result in a more indirect path to minimum convergence, aptly naming this variant of gradient descent as stochastic [25, 50].

Another gradient descent variant is minibatch gradient descent. In minibatch gradient descent, a preselected number of samples (batch) is chosen [48]. This minibatch is what is averaged for each parameter update [13]. For $1 < n < N$,

$$w_{t+1} = w_t - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \frac{\partial L(\hat{y}^{(i)}, y^{(i)})}{\partial w} \tag{3.8}$$

Gradient descent algorithms will typically contain an *algorithm* parameter, batch size, which allows practitioners to choose how many training samples will be considered for a single *model* parameter update [4]. Since this *algorithm* parameter is used to influence a different type of parameter, it is considered a *hyperparameter* [8]. In this study, SGD is technically referring to minibatch gradient descent. The term SGD is used instead as this is the algorithm's official name within the TensorFlow API documentation.

3.1.4.3 Adam

Adam is an optimizer which differs from gradient descent in two primary ways. Adam utilizes momentum and adaptive learning rates to perform parameter updates [28, 30]. Momentum in terms of a deep learning optimizers refers to an algorithm's ability to keep track of previous parameter update directions [30]. The addition of this temporal element will help the algorithm compute an exponential moving average of prior directions (negative gradients) [7, 30]. This average will help dampen movements in directions that are not toward the minimum and instead accelerate (build momentum) toward the overall direction of minimization [30]. Momentum, m_t, can be described in the following equation,

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t \tag{3.9}$$

where g_t is the current gradient and β_1 is a hyperparameter to specify the weightage of the exponential moving average [30].

Adaptive learning rates function similarly. However, instead of computing a moving average of prior negative gradients, an exponential moving average v_t of the prior squared gradients is tracked,

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{3.10}$$

where β_2 is a separate hyperparameter to be distinguished from β_1 [7, 30].

For Adam, the parameter update utilizes m_t and v_t in he following equation,

$$w_{t+1} = w_t - \eta \cdot \frac{m_t}{\sqrt{v_t + \epsilon}} \cdot g_t \tag{3.11}$$

where ϵ is a small scalar quantity to prevent division of zero [30]. We see that m_t provides momentum for the direction of greatest descent while the square root of v_t has the effect of

normalizing the learning rate [7, 30]. Parameters which cause the path of convergence to oscillate are penalized by v_t while parameters that do the opposite are encouraged by m_t [30].

3.2 Deep Segmentation Network Architecture

Different CNNs can be constructed differently based on the number of convolutional layers and pooling layers, how these convolutional layers and pooling layers are connected, and if deconvolution layers are present, how they are connected as well [44]. This study implements an architecture called U-Net, first introduced by Ronneberger et al. in 2015 [44].

U-Net architecture is divided into two pathways, a contracting pathway and an expansive pathway. The contracting pathway works identically to what is described in Chapter 3.1.3, multiple filters are applied to an image to produce convolutional layers which undergo pooling to downsample the data. It is at this point where U-Net architectures differ from a standard vanilla CNN. From here, deconvolutions take place which upsample the data until the original image resolution is met. This upsampling creates the expansive pathway.

Upsampling by itself, however, results in a loss of spatial information. In reference to Figure 3.2, if a deconvolution operation was performed to reverse the image resolution from 2x2 to 4x4, how pixel values of the 2x2 image should be mapped onto the 4x4 image would be unknown. Specifically, if we considered the "3" in the 2x2 image, we only know that it represents the maximum of the upper left 2x2 quadrant of the 4x4 image. It is unknown how many "3's" were present in this upper left quadrant and exactly *where* they were located.

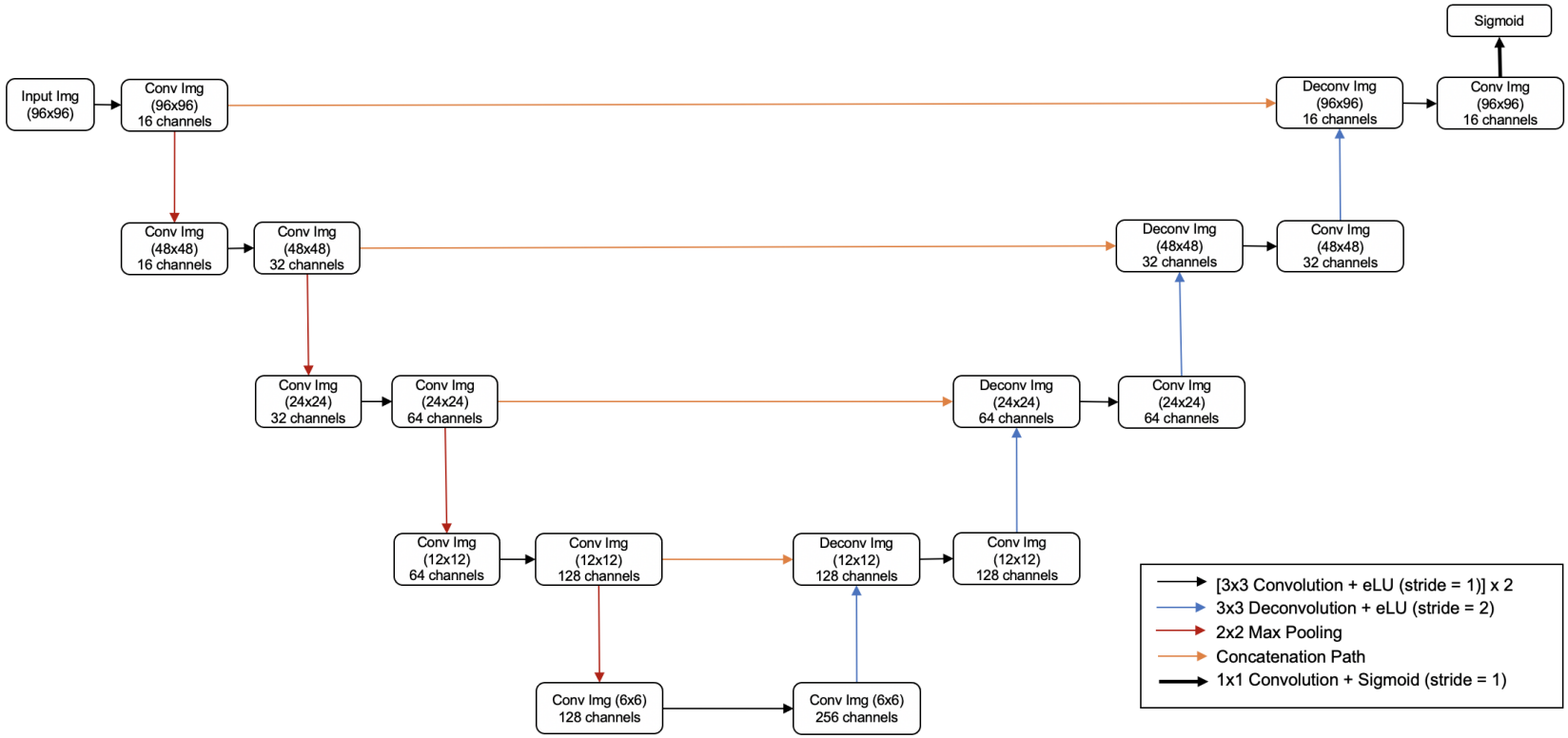

Figure 3.4

U-Net CNN Architecture

To preserve this spatial information, skip connections are formed between layers of the contracting pathway and layers of the expansive pathway. Convolutional layers of the contracting pathway are concatenated with their deconvolutional counterpart of the same resolution. Details such as the depth of the network and number of convolutions can be seen in Figure 3.4. These same architecture specifications are implemented in this study for 3D images.

3.3 Dice Score Coefficient and Dice Loss

The primary metric which is used to evaluate the success of segmentation prediction in this study is the Dice score coefficient (DSC). For binary classification,

$$DSC = \frac{2\sum_{i}^{N} \hat{y}_i y_i}{\sum_{i}^{N} \hat{y}_i + \sum_{i}^{N} y_i} \tag{3.12}$$

where the measurement for correct predictions is the intersection between the prediction $\hat{y}$ and correct output label y with respect to the sum of correct and predicted labels. Formulating the correctness of predictions in this way is an appropriate metric of evaluation when data contains class imbalances. In the case of head and neck tumors, the actual tumor itself may only represent a small percentage of the overall medical image. DSC values will elucidate how well a model segmented an image when a class is not well represented while accuracy as a metric may be inflated due to the model's correct negative predictions. To incorporate the DSC as the loss function of the CNN, the Dice loss is used,

$$Loss = 1 - DSC \tag{3.13}$$

3.4 Bayesian Hyperparameter Optimization

Hyperparameter tuning has the potential to be extremely tedious if it requires excessive evaluation of the loss function for each hyperparameter combination [45]. Further, the relationship between the loss function and the hyperparameter combinations cannot be directly measured [45, 17, 29]. Bayesian optimization allows for tuning of hyperparameters without knowing the loss function while also requiring less evaluations [45, 9].

For a given loss function, $\varphi(x)$, where $x \in \mathbb{X}$, x is a vector of hyperparameters for a given combination. Via Bayesian optimization, the optimal hyperparameter combination, x^*, is searched, where $x^* = \arg\min_{x\in\mathbb{X}} \varphi(x)$ [45]. In order to achieve this, an a-priori probability distribution, $P(\varphi)$, and an acquisition function, $a_{P(\varphi)}$, are used. D_n is a set of n hyperparameter combinations where $D_n = (x_i, y_i)_{i=1,\dots,n}$ and $y_i = \varphi(x_i) + \varepsilon_i$ [45]. Hyperparameter combinations are determined by the combination that maximizes the acquisition function,

$$x_{n+1} = \arg\max_{x\in\mathbb{X}} a_{P(\varphi|D_n)}(x). \tag{3.14}$$

Once x_{n+1} is obtained, the loss is calculated, $y_{n+1} = \varphi(x_{n+1}) + \varepsilon_{n+1}$ [45]. (x_{n+1}, y_{n+1}) is then added to D_n [45]. From here, both the probability model, $P(\varphi|D_{n+1})$, and the acquisition function, $a_{P(\varphi|D_{n+1})}$ are updated [45]. This update is achieved by use of Bayes' Theorem which describes that the posteriori probability distribution for a set with additional points is proportional to the a-priori probability distribution multiplied by the similarity of this new set,

$$P(\varphi|D_{n+1}) \propto P(D_{n+1}|\varphi)P(\varphi). \tag{3.15}$$

This process is repeated for several iterations, theoretically converging to an optimal set of hyperparameters, x^* [45, 29].

A Gaussian process was used for the a-priori model, which utilizes a mean function, $m(x)$, and a covariance function, $k(x, x')$, where

$$\varphi(x) \sim GP(m(x), k(x, x')). \tag{3.16}$$

Expected improvement as an acquisition function has resulted in promising results for previous Bayesian Optimization implementations [43, 45]. For this study, expected improvement is used to select batch size and learning rate with currently observed data, D_n,

$$a^{EI}_{P(\varphi|D_n)}(x) = (\varphi(x^*) - m(x))\Phi(z) + \sigma(x)\phi(z) \tag{3.17}$$

where $z = \frac{\varphi(x^*) - m(x) - \xi}{\sigma(x)}$, Φ is the cumulative distribution function of the normal distribution, $N(m(x), k(x, x')$, and ϕ is the density function [45]. ξ is a parameter which can control how hyperparameters are searched in terms of exploration and exploitation [45, 17].

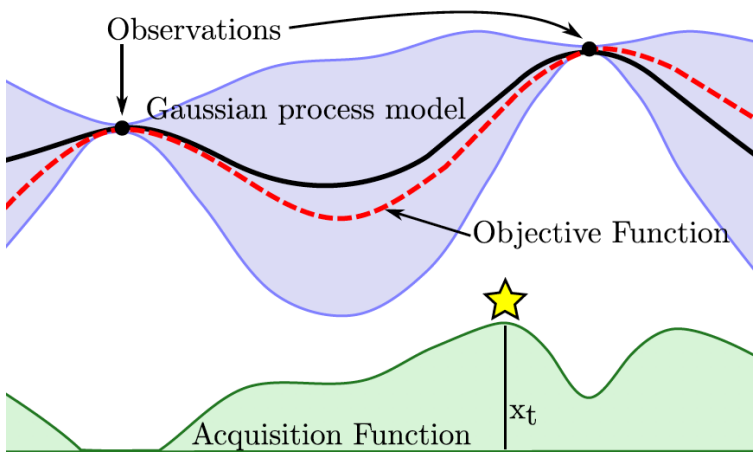

Figure 3.5

Gaussian process model and acquisition function used to determine x_t [17]

3.5 Bayesian Optimization-derived Scheduling (BOS)

The method used to improve deep neural network training is titled Bayesian optimization-derived scheduling (BOS). BOS uses Bayesian optimization to determine both the learning rate and the batch size for two separate phases of scheduled training. Optimizing the learning rate and batch size simultaneously ensures that the optimal batch size to learning rate ratio is used for each training phase. Exploratory data analysis revealed that certain hyperparameter combinations could result in a fast, reliable increase in validation DSC values while other hyperparameter combinations resulted in a stable, more gradual progression without overfitting. This result motivated a two phase schedule.

Algorithm 1 Bayesian Optimization-derived Scheduling (BOS)

1: X: Original images; y: Label images
2: **procedure** 1: Preprocess Images
3: Preprocess original images: $X \leftarrow \text{preprocess}(X)$
4: Preprocess label images: $y \leftarrow \text{preprocess}(y)$
5: Randomize and split original images: $X_{opt}, X_{val}, X_{test} \leftarrow \text{split}(X)$
6: Randomize and split label images: $y_{opt}, y_{val}, y_{test} \leftarrow \text{split}(y)$
7: **end procedure**
8: **procedure** 2: Bayesian Hyperparameter Optimization
9: **for** $i = 1, \ldots, N$ **do**
10: Train phase 1 using optimization set: model.fit(X_{opt}, y_{opt})
11: Train phase 2 using optimization set: model.fit(X_{opt}, y_{opt})
12: Evaluate model using validation set: model.evaluate(X_{val}, y_{val})
13: Return DSC, learning rate, and batch size
14: **end for**
15: **return** Learning rate and batch size of highest DSC
16: **end procedure**
17: **procedure** 3: K-Fold Cross Validation
18: **for** $j = 1, \ldots, K$ **do**
19: Split X_{test} and y_{test} with respect to K folds
20: Train phase 1 using test set: model.fit(X_{test}, y_{test})
21: Train phase 2 using test set: model.fit(X_{test}, y_{test})
22: Evaluate model
23: Return DSC, Dice loss, precision, recall, and duration
24: **end for**
25: **return** Results of all K folds in data frame
26: **end procedure**

Algorithm 1 describes the general process of BOS in pseudocode. The exact original images X can vary from CT images, PET images, or CT/PET fused images (2D or 3D). The training/validation/test split is divided into a 21/9/70 split, where the training set is designated as an optimization set to be used in procedure 2. The validation set is used to evaluate the generalization ability gained from steps 10 and 11. Lastly, for each fold in procedure 3, the test set is divided according to the K value specified.

In procedure 2, N represents the sum of exploratory and exploitative hyperparameter combinations tested. Training phase 1 of procedure 2 is identical to training phase 1 of procedure 3. Similarly, training phase 2 of procedure 2 is identical to training phase 2 of procedure 3. In training phase 1, a stopping criteria is defined using the EarlyStopping callback. Model training in training phase 1 will cease when the Dice loss has not improved for two epoch iterations. When this occurs, training phase 2 is initiated with a different learning rate and batch size. The stopping criteria for phase 2 is defined for when the Dice loss has not improved for ten epoch iterations. In phase 2, improvement is defined for when the the current Dice loss has minimized with a difference greater than 0.001 of the previous smallest Dice loss. For example, if the previous smallest Dice lost was 0.45, a Dice loss of 0.449 or lower will constitute improvement while any value within $(0.449, 1]$ would constitute as failure to minimize. A checkpoint is set within training phase 2 to so that the best performing model is evaluated in step 12 and step 22.

Once the surrogate model in procedure 2 has evaluated all N hyperparameter combinations, the combination which resulted in the highest DSC value is passed to procedure 3 for K-fold cross validation. Procedure 3 is used for $K = 10$, with the test set separated into a 90/10 split for each iteration. This ensures that each 10^{th} of the test set is evaluated. The results of all 10 folds are stored in a data frame which is then exported as a comma-separated values (CSV) file and/or Excel file.

CHAPTER IV

EXPERIMENTAL RESULTS

This chapter provides a discussion of the results. The data is described along with the steps taken to preprocess the data. The results of an exploratory hyperparameter analysis (EHA) are shared which motivated the use of a two phase training schedule which utilizes Bayesian optimization for hyperparameter selection - Bayesian optimization-derived scheduling (BOS). The results of each strategy are discussed. Lastly, segmentation predictions are visualized and discussed as well.

4.1 Data Description

The data was sourced from the 2021 Medical Image Computing and Computer Assisted Intervention (MICCAI) Head and Neck Tumor Segmentation Challenge. It is comprised of 224 CT and PET images in NIfTI format. Ground truth contours were originally annotated by radiation oncologists and were then curated (re-annotated) by experts who were both a radiologist and nuclear medicine physician for the purpose of the MICCAI Challenge. Fused CT/PET image 3D contours were edited using the Siemens Syngo.Via RT Image suite [5]. Figure 4.1 shows a CT image, the ground truth mask, and the ground truth mask contour transposed onto its corresponding CT image.

All patients were diagnosed with primary squamous cell carcinoma (PSCC) of the head and neck (TNM-Stage I-IV) and were being treated either with radiation therapy only or chemotherapy

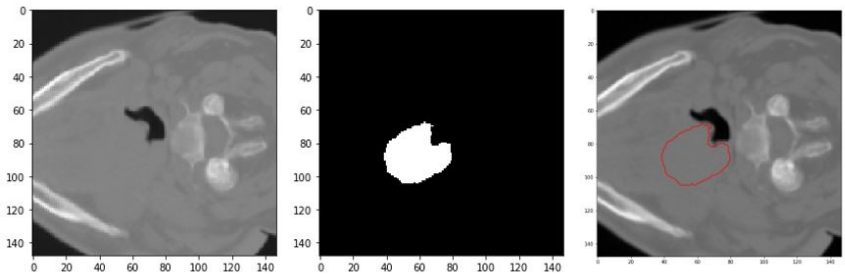

Figure 4.1

MICCAI Head and Neck images (CT, ground truth, and CT + ground truth)

and radiation therapy. TNM-Staging classifies the size of the tumor progression (T), number of nodes that have cancer (N), and level of metastasis (M). Tumors were located in the oropharynx region. Patient ages ranged from 18 to 84, with a mean of 61 years old. The majority of patients were male (74.6% male and 25.4% female). Table 4.1 details these patient demographics.

CT and PET images are sourced from 4 different Canadian hospital centers, Hôpital Général Juif, Montréal (CHGJ); Centre Hospitalier Universitaire de Sherbrooke, Sherbrooke (CHUS); Hôpital Maisonneuve-Rosemont, Montréal (CHMR); and Centre Hospitalier de l'Université de Montréal, Montréal (CHUM). All centers used a hybrid PET/CT scanner where the bed was placed in multiple positions for image acquisition. Each position is acquired for a specific duration of time (position acquisition). The median position acquisition and range for each machine in each center is listed in Table 4.2. For PET, the radiotracer activity is recorded for the medium that is intravenously injected. These median activity values and ranges are given in Table 4.2 as well. For centers CHGJ, CHMR, and CHUM, reconstruction of attenuation corrected images used an ordered subset expectation maximization (OSEM) iterative algorithm with a median span of

5 (axial smash). For CHUS, attenuation corrected images were reconstructed using a Line of Response-Row Action Maximum Likelihood Algorithm (LOR-RAMLA) iterative algorithm.

Table 4.1

Characteristics of MICCAI Patient Data

Characteristic	Type	No. of Patients
Gender	Male	167 (74.6%)
	Female	57 (25.4%)
Age	Range	34-90
	Mean ± STD	63 ± 9
TNM-Stage	Stage I	4 (1.8%)
	Stage II	48 (21.4)
	Stage IV	172 (76.8%)
T-Stage	T1	26 (11.6%)
	T2	94 (42.0%)
	T3	58 (25.9%)
	T4	45 (20.1%)
	Tx	1 (0.4%)
N-Stage	N0	33 (14.7%)
	N1	26 (11.6%)
	N2	150 (67.0%)
	N3	15 (6.7%)
M-Stage	M0	219 (97.8%)
	M1	4 (1.8%)
	Mx	1 (0.4%)
HPV Status	Positive	84 (37.5%)
	Negative	30 (13.4%)
	N/A	110 (49.1%)
Treatment	Radiation only	27 (12.1%)
	Chemo-radiation	197 (87.9%)

Table 4.2

PET/CT Scan Specifications

Specification	CHGJ	CHUS	CHMR	CHUM
Machine Type	Discovery ST, GE Healthcare	GeminiGXL 16, Philips	Discovery ST, GE Healthcare	Discovery ST, GE Healthcare
Radiotracer Activity	584 MBq (368-715)	325 MBq (165-517)	475 MBq (227-859)	315 MBq (199-3182)
Position Acquisition	300 s (180-420)	150 s (120-151)	360 s (120-360)	300 s (120-420)
PET Slice Thickness	3.27 mm	4 mm	3.27 mm	4 mm
PET In-Plane Resolution	3.52 x 3.52 mm^2	4 x 4 mm^2	3.52 x 3.52 mm^2	4 x 4 mm^2
CT Energy	140 kVp	140 kVP	140 kVP	120 kVp
CT Exposure	12 mAs	210 mAs	11 mAs	350 mAs
CT Slice Thickness	3.75 mm	3 mm	3.75 mm	1.5 mm
CT In-Plane Resolution	0.98 x 0.98 mm^2	1.17 x 1.17 mm^2	0.98 x 0.98 mm^2	0.98 x 0.98 mm^2

4.2 Data Preprocessing

In each each original 3D CT or PET image, the patient occupies a fraction of the entire image. Images are cropped according to bounding boxes provided by the MICCAI Challenge (Figure 4.2). This ensures that the model will be trained for regions of the image that solely contain the patient rather than excessive background area. Cropped images are then resized to 96×96×96. The intensity values of the CT images are clipped in the range of [-1024, 1024] Hounsfield Units. After clipping, values are mapped to [-1, 1]. Each PET image was normalized individually using Z-score normalization.

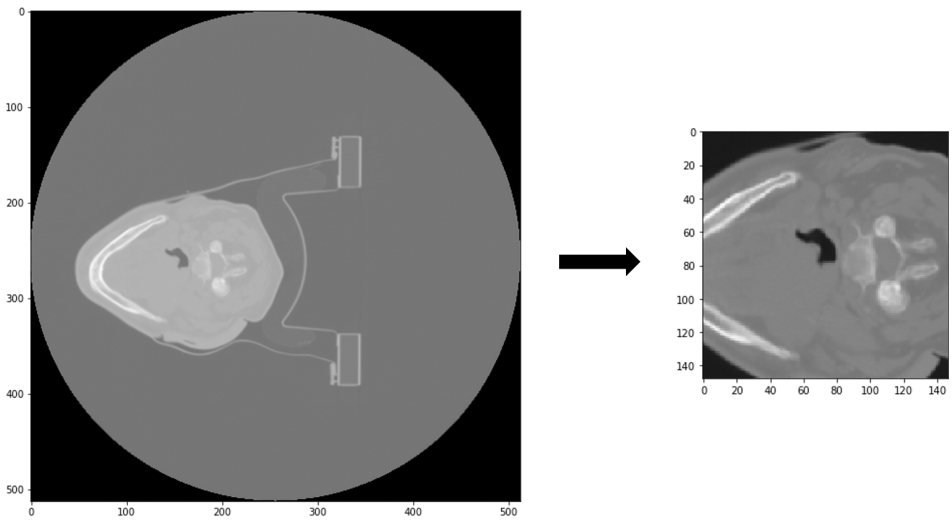

Figure 4.2

Cropping of original CT image

A 2D and 3D version of a U-Net model is used in this study. In order for the original 3D images to satisfy the 2D requirement of the 2D U-Net model, each 3D image is converted into a series of

2D slices in the axial plane, resulting in 96 slices per patient. Three different image modalities are considered, CT, PET, and CT/PET. While both 3D and 2D formats are of interest, 2D slices are also trained where all slices contain at least some portion of a tumor (Figure 4.3). Image sets that exclusively use these slices are denoted with the prefix "ROI" to designate slices with specific regions of interest (tumors).

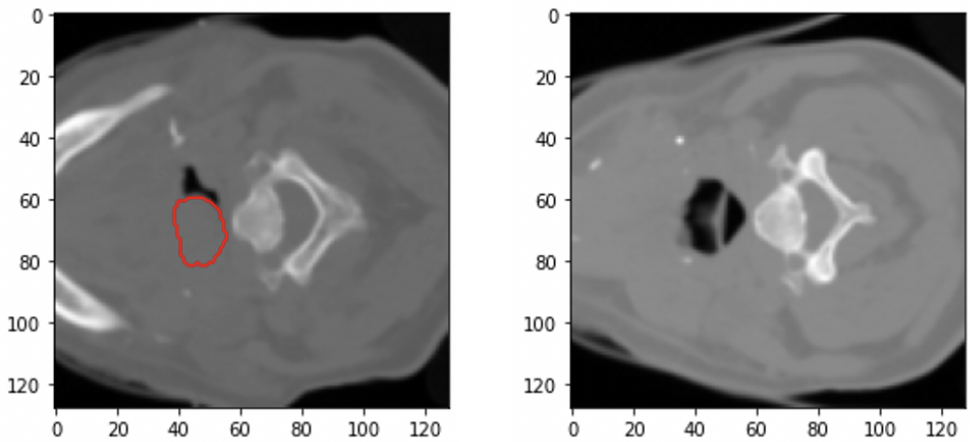

Figure 4.3

Image sets labeled with the "ROI" prefix only contain slices with a tumor present (left). Image sets simply labeled "CT", "PET", or "CT/PET" contain these same slices in addition to slices for which there are no tumors present (right).

Restricting 3D images to a bounding box volume containing each tumor would result in imbalances where the tumor would occupy a significant proportion of each image. Instead, PET modality variations are used in addition to CT, PET, and CT/PET to evaluate different cases for BOS. Gaussian derivatives are used to obtain gradient magnitudes for PET-grad filters (Figure 4.4). Each transformation uses a standard deviation of 0.5 in all three axes. The PET-grad transformation

emphasizes the edges of the original PET images. Additional PET filters are created using the maximum standard uptake value (SUV) of each image. Thresholds of 50%, 40%, 30%, and 20% of the maximum SUV are used to eliminate potential noise in the original PET images (Figure 4.4). Sridhar et al. used gradient, 30%, 40%, and 50% thresholds as tumor segmentation methods and found that gradient segmentation resulted in the highest correlation with pathologic tumor volume [53]. Their results motivate the use of PET variants to hopefully increase model performance.

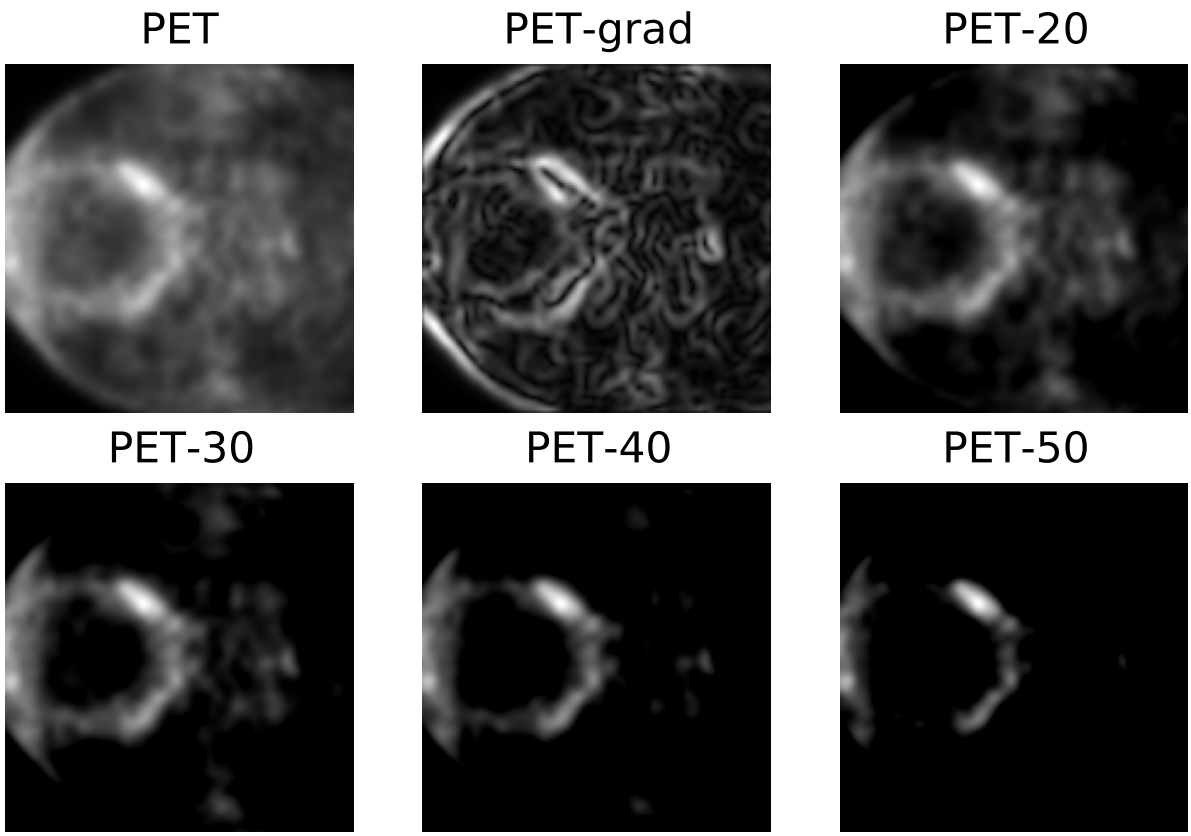

Figure 4.4

Original PET image and PET variations used in BOS

Table 4.3

Hyperparameters used in Exploratory Analysis

Optimizer	Learning Rate	Batch Size	Initializer
Adam	1e-2	2	He Normal
Nadam	1e-3	10	Glorot Normal
SGD	1e-4	20	
	1e-5		
	1e-6		

4.3 Exploratory Hyperparameter Analysis (EHA)

After successfully preprocessing images and constructing the U-Net model, an analysis of model behavior was performed with respect to different hyperparameter design choices. Through various trials, generalization performance appeared to be highest with ROI-2D images. ROI-2D CT images were used in the analysis to determine how the model would behave with different combinations of hyperparameters. Table 4.3 shows the different hyperparameters that are explored in the analysis.

DSC, precision, and recall history were recorded for 100 epochs for both training and validation datasets. Generalization was assessed visually using the DSC values of the validation dataset for different hyperparameter combinations. The majority of hyperparameter combinations showed either excessive overfitting or underfitting. Each figure from the analysis shows trends for a single optimizer and specific batch sizes and learning rates.

4.3.1 Adam and Nadam

For Adam and Nadam, a learning rate of 1e-6 resulted in reliable model training without overfitting and underfitting. At a larger learning rate of 1e-5, overfitting is observed while a general

trend towards increasing DSC values is observed. For learning rates larger than 1e-5, we see that the model is completely unable to generalize. When considering how batch sizes affect the model, we see that for a learning rate of 1e-6, a smaller batch size is preferred. However, when moving from a batch size of 20 to 10 to 2, slightly greater instability is observed as the model's generalization improves. Neither Adam nor Nadam achieved a validation DSC value greater than 0.40 when considering hyperparameter combinations that did not result in overfitting. These observations can be seen in Figure 4.5. The results of the training loss, DSC, precision, recall and validation loss, precision, and recall can be seen in Figures 4.6-4.12.

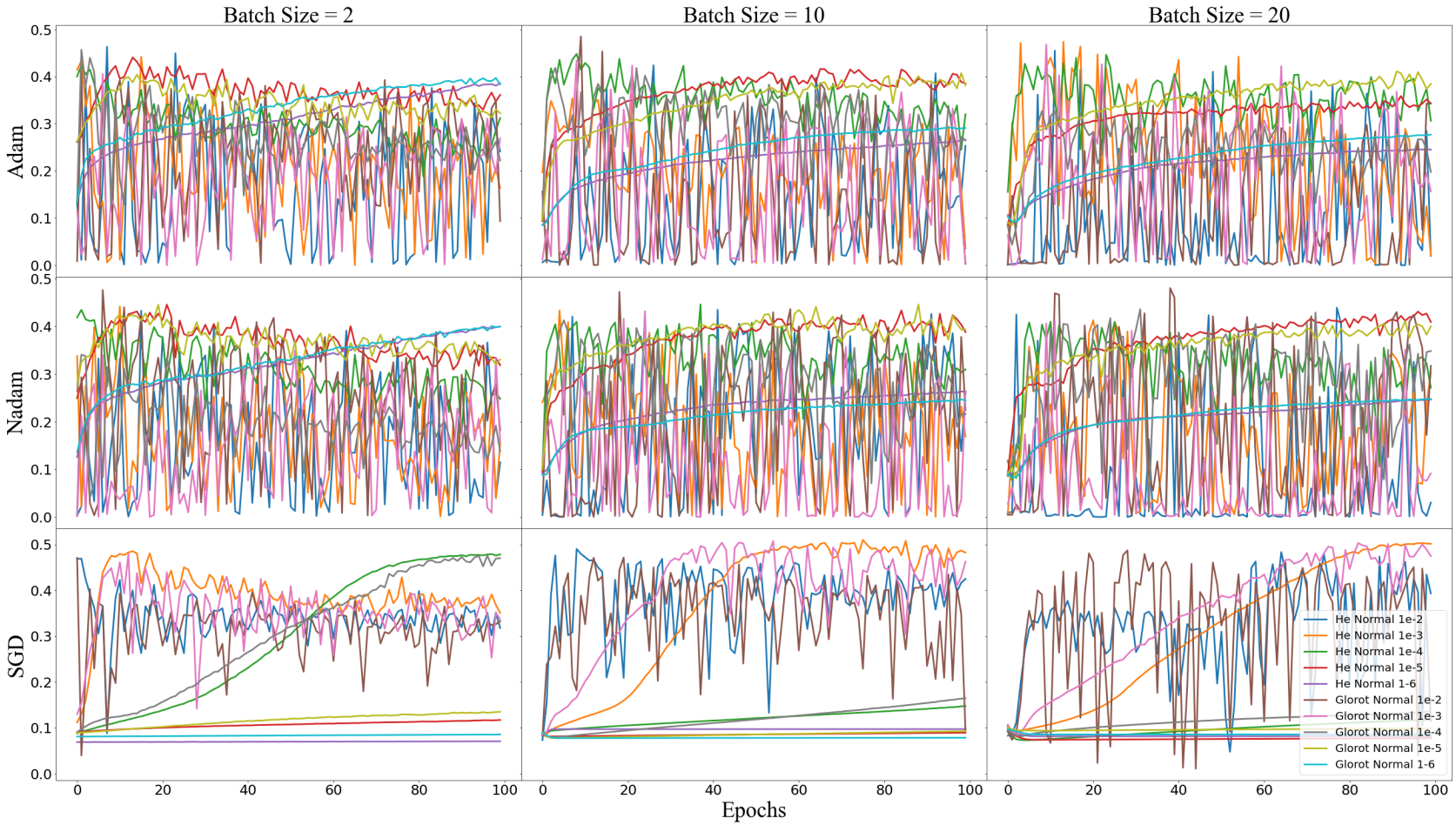

Figure 4.5

Validation Dice Score Coefficients

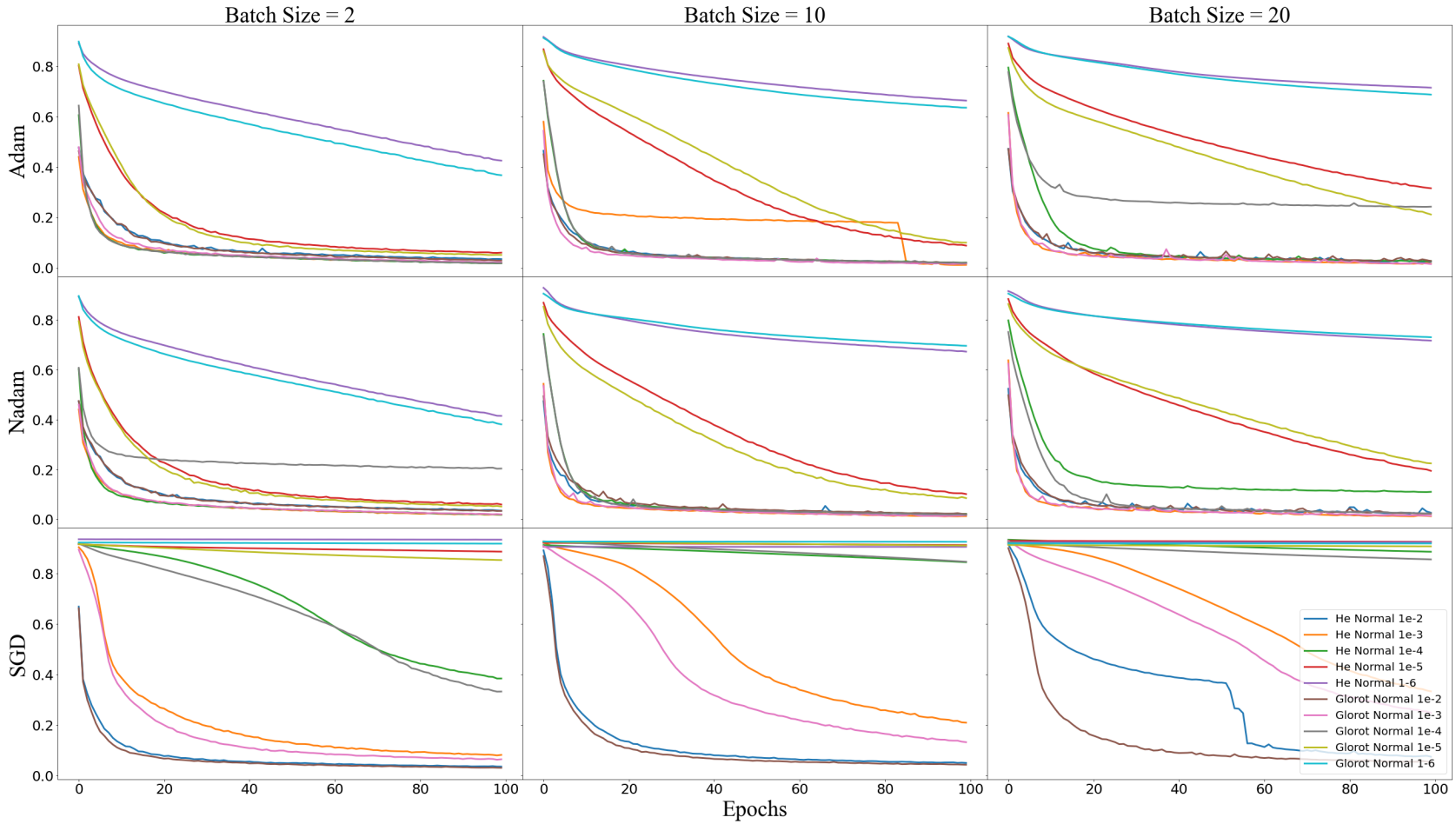

Figure 4.6

Training Dice Loss

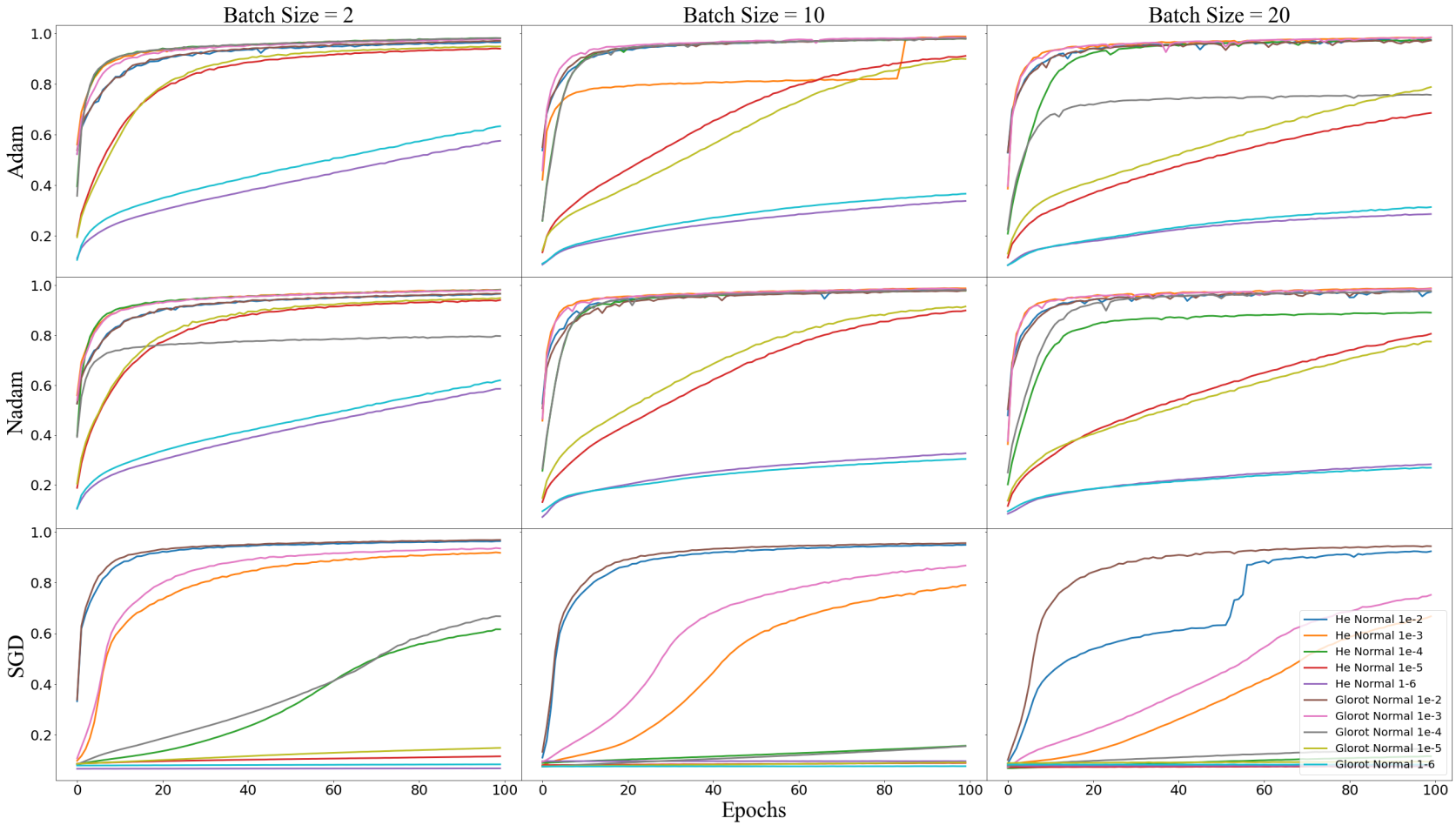

Figure 4.7

Training Dice Score Coefficients

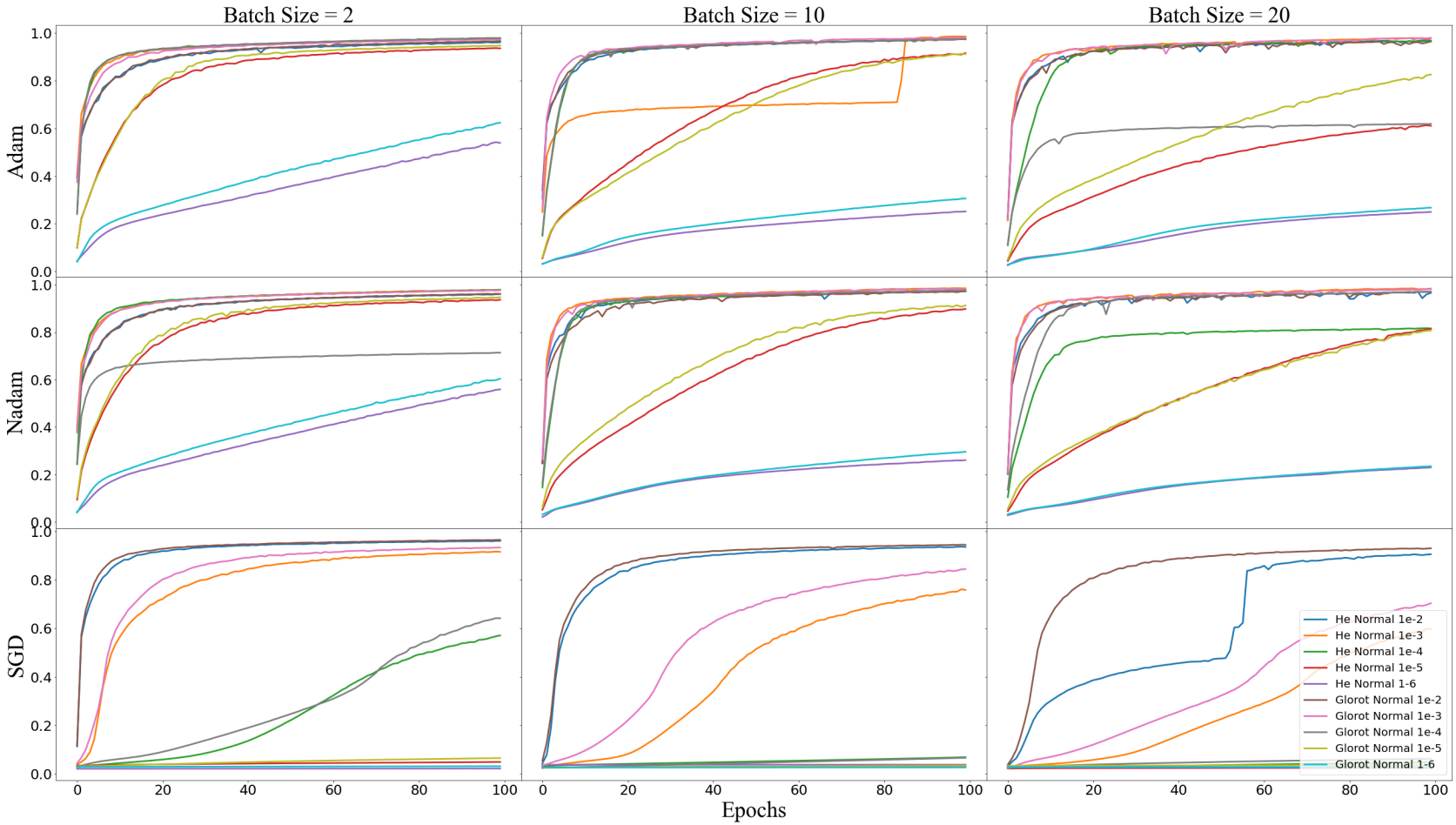

Figure 4.8

Training Precision

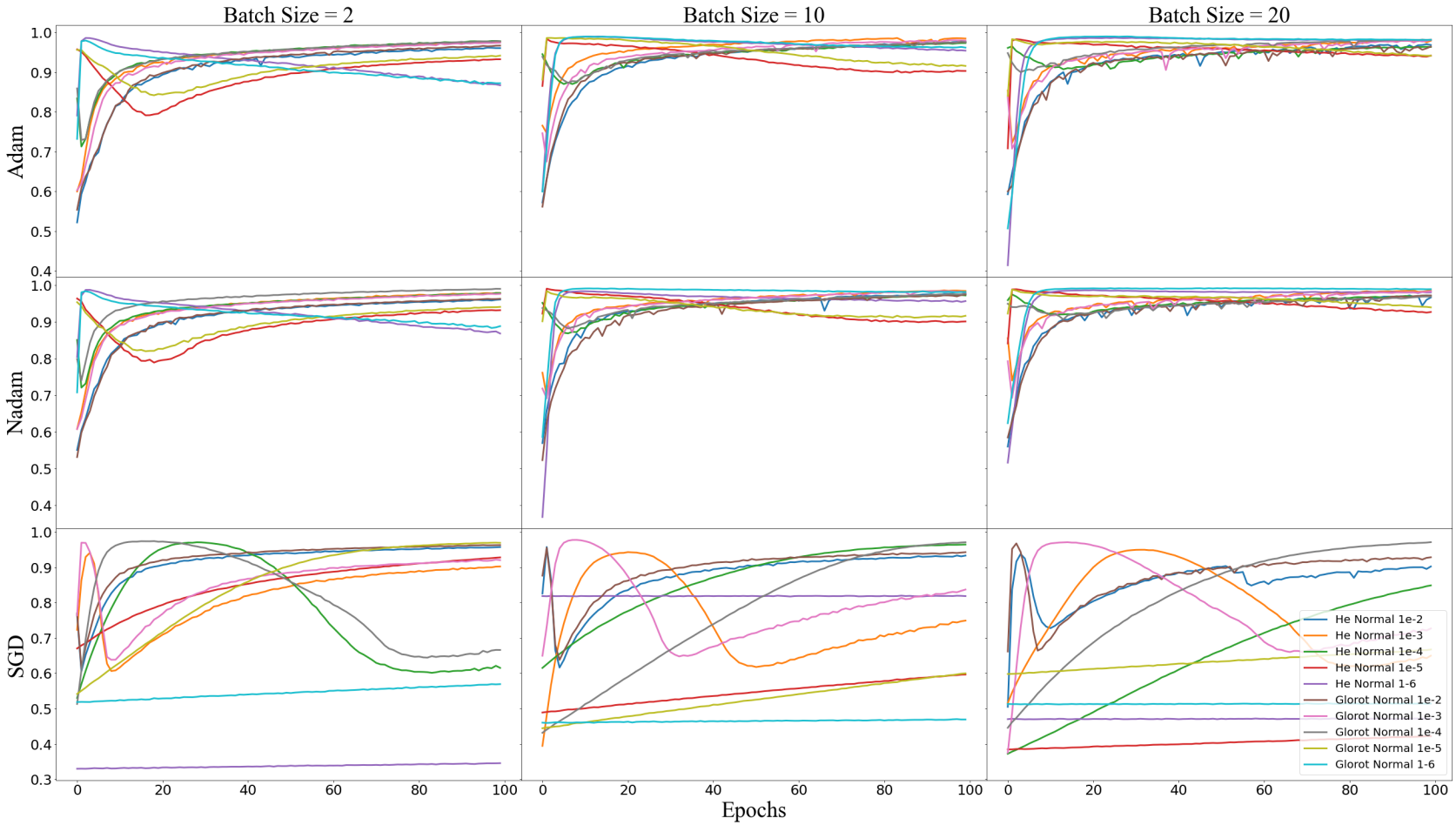

Figure 4.9

Training Recall

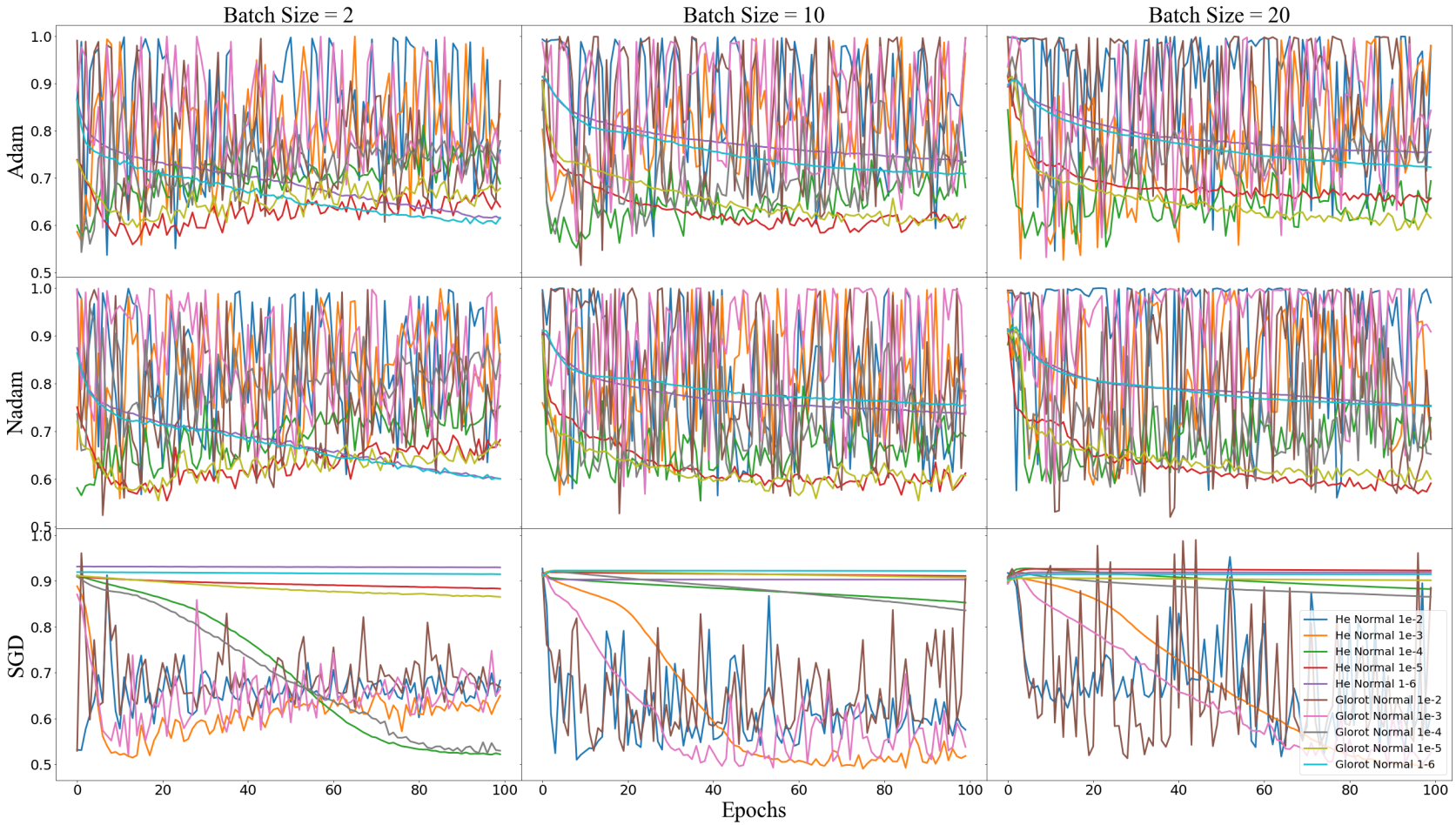

Figure 4.10

Validation Dice Loss

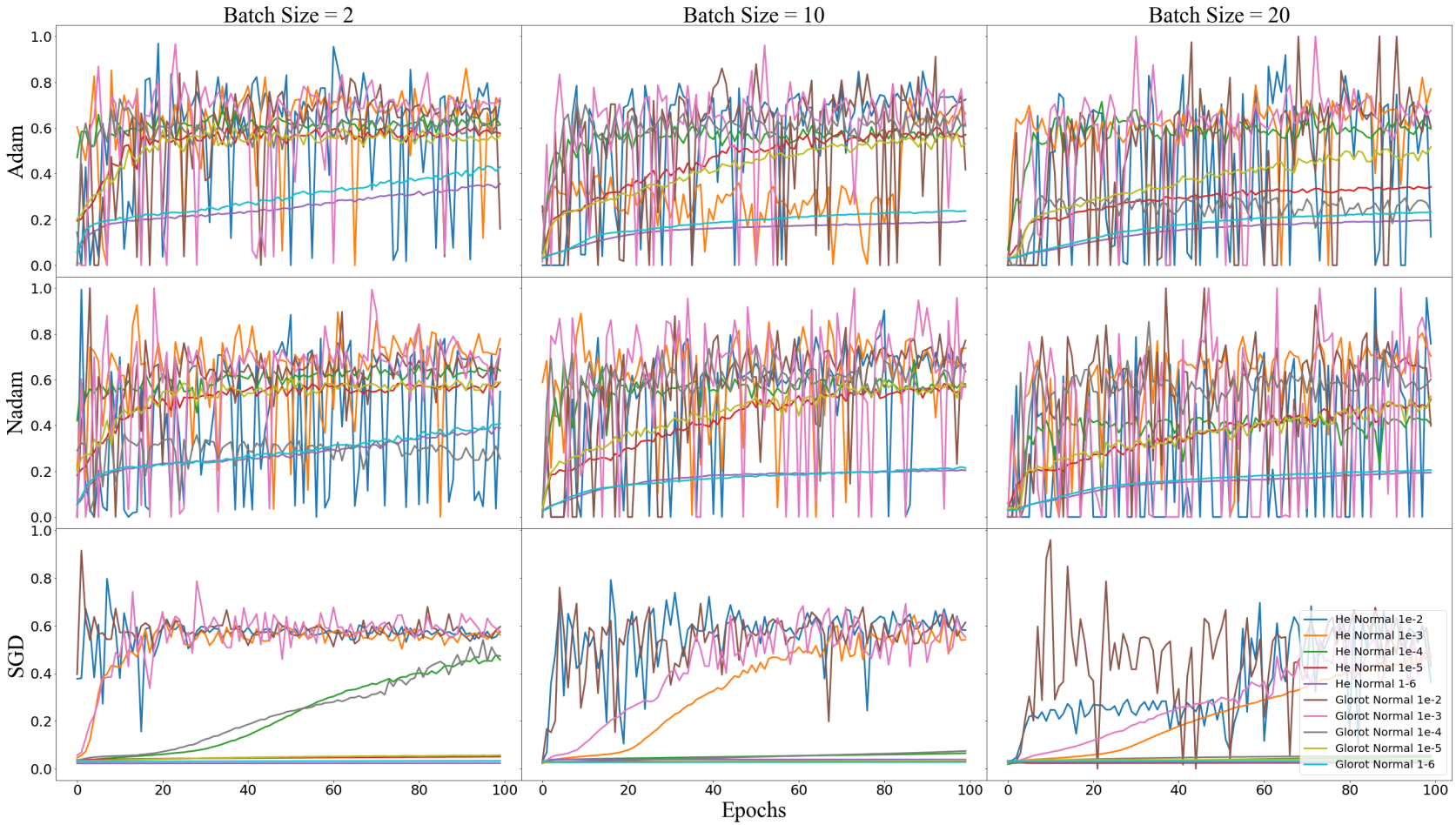

Figure 4.11

Validation Precision

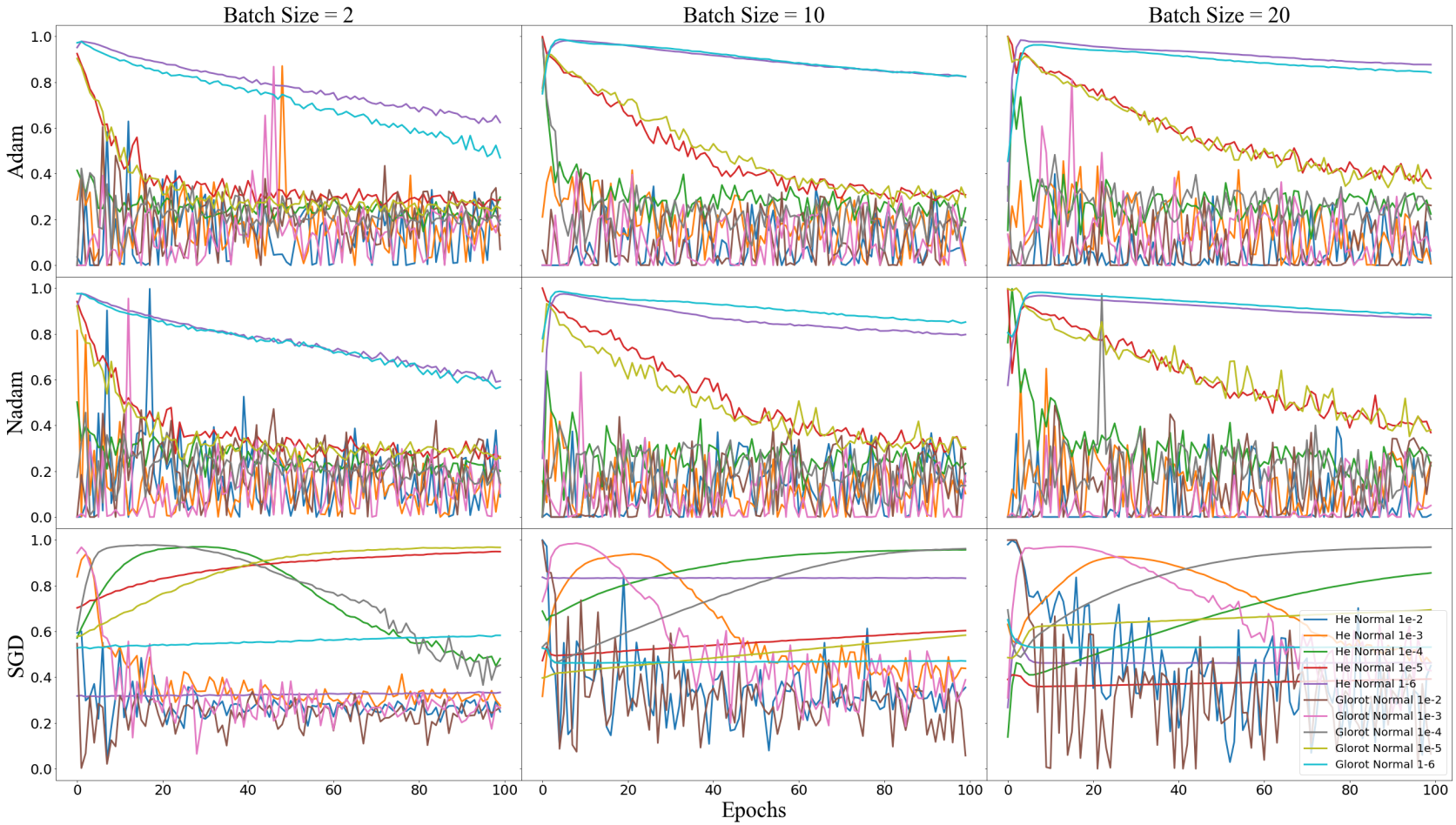

Figure 4.12

Validation Recall

4.3.2 SGD

SGD demonstrated behavior slightly different from Adam and Nadam. For learning rates 1e-5 and 1e-6, the model showed severe underfitting, with validation DSC values never reaching 0.20. With a batch size of 10 and 20, a learning rate of 1e-4 showed similar underfitting. In contrast, overfitting occurred when learning rates were too large. The effect of batch size can be seen with He Normal initialization and a learning rate of 1e-4 (Figure 4.5). As mentioned before, this combination with batch sizes of 10 and 20 demonstrated significant underfitting. At a batch size of 2, however, the model performs significantly better, reaching a validation DSC value of 0.47 with no signs of overfitting. It appears that for this combination (SGD/He Normal/1e-4), decreasing the batch size helped the model transition from underfitting to suitable fitting. Interestingly, He Normal initialization with a learning rate of 1e-4 and batch size of 2 resembles He Normal initialization with a learning rate of 1e-3 and a batch size of 20. Unsurprisingly, these two combinations have identical batch size to learning rate ratios (20,000). In the latter combination, a batch size of 20 results in suitable fitting. Decreasing the batch size then pushes the model from suitable fitting to overfitting (Figure 4.5).

It becomes apparent that for a given hyperparameter combination, a specific batch size exists which places the model somewhere between overfitting and underfitting. The same pattern can be seen with differing learning rates. For any of the batch size combinations, learning rates of 1e-2, 1e-3, and 1e-4 collectively show the effect of a learning rate overfitting when too large and underfitting when too small (Figure 4.5). Overfitting appears to occur when either the batch size is too small and/or the learning rate is too large. Underfitting appears to occur when either the batch size is too large and/or the learning rate is too small. In terms of batch size to learning rate

ratios, if the batch size to learning rate ratio is too large, underfitting may occur. If the batch size to learning rate ratio is too small, overfitting may occur. Figure 4.13 illustrates this optimal batch size to learning rate ratio. Ratios were calculated with batch sizes ranging from 2 to 32 and learning rates from 1e-4 to 1.

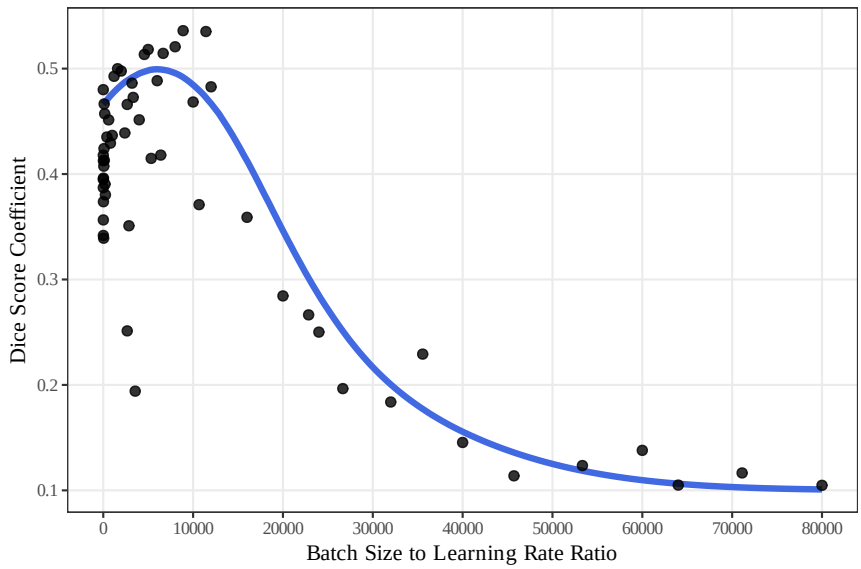

Figure 4.13

Observation of Optimal Ratio with SGD

Another observation unique to SGD is that in some hyperparameter combinations, once overfitting is observed, the validation DSC values fluctuate, but not in such a manner that they oscillate and approach zero. For instance, He Normal initialization with a batch size of 2 and learning rate of 1e-3 shows overfitting, but never drops below 0.35. With Adam or Nadam, it is not uncommon to see much more drastic fluctuations, with validation DSC values regularly fluctuating between 0.35 and 0.01. An additional note of interest is that when a hyperparameter combination involving

SGD overfits, it usually also demonstrates a sharp, reliable increase in validation DSC values in the beginning of training, which is much faster than more stable hyperparameter combinations that are void of such overfitting.

The set of hyperparameters which yielded the best validation results was SGD, He Normal initialization, with a batch size of 20 and a learning rate of 1e-3. This model demonstrated a validation DSC value of 0.50. There was only one other combination which produced comparable results, SGD, He Normalization, with a batch size of 2 and learning rate of 1e-4 which demonstrated a validation DSC value of 0.48. Both of these hyperparameter combinations had a batch size to learning rate ratio of 20,000.

Two findings motivated the use of EHA scheduling, the consistent sharp increase in the beginning of training that was observed in a combination such as SGD/1e-3/batch size 2 and the stability observed in the best performing combination, SGD/1e-3/batch size 20. The resultant DSC validation behavior of scheduling these two combinations is shown in Figure 4.14.

This specific schedule is derived purely from our EHA. It was evaluated using 10-fold cross validation CT, PET, and CT/PET images (2D and ROI-2D). The results for mean DSC value, precision, recall, and training computation time (TCT) in minutes, along with 95% confidence intervals are shown in Table 4.4. These results serve as a basis of which to compare Bayesian optimization-derived scheduling (BOS) against.

4.4 Bayesian Optimization-derived Scheduling (BOS) Results

Bayesian optimization was used to identify four optimal hyperparameters for a two phase training schedule, the learning rate and batch size for phase 1, and the learning rate and batch size

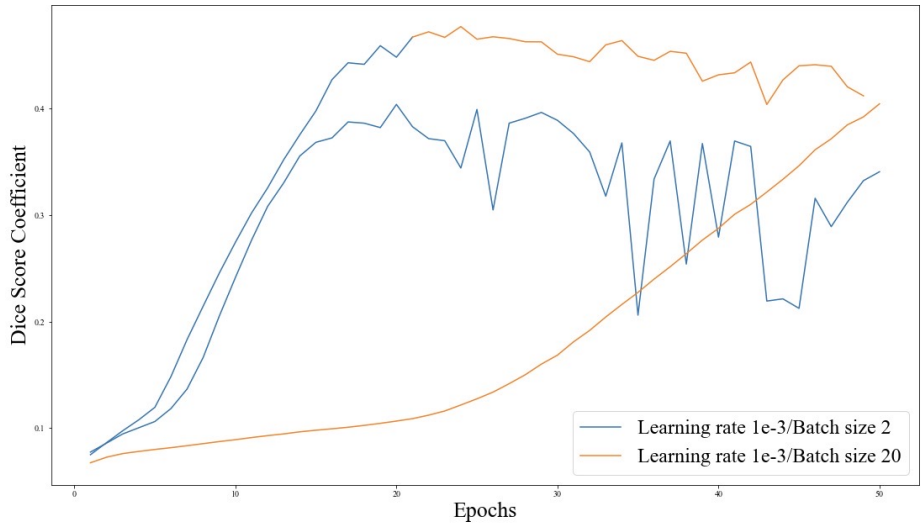

Figure 4.14

EHA Scheduling

for phase 2. Bayesian hyperparameter optimization was performed using SGD and He Normal initialization. A total of 30 iterations were performed, 5 of which were exploratory. The batch size parameter bounds were defined from 1 to 32 for 2D images and 1 to 16 for 3D images. The batch size to learning rate ratio parameter bounds were defined from 1 to 10,000 for both 2D and 3D images. The resultant optimal hyperparameters for each phase are shown in Table 4.5 for each image modality.

For 2D images, there is variation among learning rates and batch sizes for both the first and second phase of training. However, one consistent finding is that the hyperparameters selected result in a relatively lower batch size to learning rate ratio for the first phase, then switch to a higher batch size to learning rate ratio for the second phase. This further emphasizes not only

Table 4.4

2D 10-Fold Cross Validation Results

Image modality	Strategy	DSC (95% CI)	Precision (95% CI)	Recall (95% CI)	TCT (95% CI)
CT	EHA	0.24 (0.22-0.26)	0.43 (0.33-0.54)	0.49 (0.40-0.58)	19.8 (17.9-21.6)
PET	EHA	0.41 (0.37-0.45)	0.76 (0.72-0.80)	0.82 (0.75-0.89)	19.4 (15.6-22.7)
CT/PET	EHA	0.41 (0.37-0.45)	0.76 (0.73-0.80)	0.80 (0.74-0.85)	19.2 (17.2-21.2)
ROI-CT	EHA	0.51 (0.44-0.57)	0.49 (0.39-0.58)	0.47 (0.41-0.54)	5.5 (4.7-6.3)
ROI-PET	EHA	0.77 (0.72-0.81)	0.75 (0.71-0.79)	0.82 (0.78-0.88)	7.7 (6.2-9.2)
ROI-CT/PET	EHA	0.79 (0.75-0.83)	0.76 (0.74-0.78)	0.84 (0.80-0.89)	6.4 (5.7-7.1)
CT	BOS	0.25 (0.22-0.28)	0.41 (0.33-0.48)	0.57 (0.47-0.67)	23.3 (16.5-30.1)
PET	BOS	0.41 (0.37-0.45)	0.84 (0.81-0.88)	0.77 (0.71-0.82)	5.8 (3.9-7.8)
CT/PET	BOS	0.42 (0.38-0.46)	0.80 (0.78-0.82)	0.80 (0.75-0.85)	5.2 (4.0-6.5)
ROI-CT	BOS	0.51 (0.46-0.56)	0.60 (0.50-0.69)	0.43 (0.34-0.52)	1.4 (1.1-1.6)
ROI-PET	BOS	0.81 (0.78-0.85)	0.79 (0.76-0.82)	0.84 (0.80-0.87)	6.7 (5.7-7.7)
ROI-CT/PET	BOS	0.81 (0.78-0.85)	0.79 (0.76-0.82)	0.84 (0.80-0.87)	4.1 (3.1-5.1)

the existence of an optimal batch size to learning rate ratio, but that this optimal ratio is uniquely dependent upon the model's progression during training.

The results for 3D images show the same trend of progressing from smaller ratios to higher ones. One difference is that in 2D models, the change from the first batch size to second batch size is inconsistent. Half of the models show a switch from a smaller batch size to a larger batch size while the other half show a switch from a larger batch size to a smaller batch size. With the exception of CT/PET-50, all 3D models start at a smaller batch size and switch to a larger batch size during the second phase of training.

Once optimal hyperparameters were identified, a 10-fold cross validation was performed for each image modality. The 2D results for mean DSC value, precision, recall, and TCT in minutes, along with 95% confidence intervals are shown in Table 4.4. Figure 4.15 shows box plots that compare each strategy (EHA and BOS) for each performance evaluation metric. Results show that model performance is primarily dependent on image modality and type (original or ROI).

Table 4.5

Optimal Learning Rate and Batch Size Identified by BOS

Image modality	2D/3D	1st Learning Rate	1st Batch Size	1st Ratio	2nd Learning Rate	2nd Batch Size	2nd Ratio
CT	2D	9.819e-1	3	3	6.328e-4	2	3058
PET	2D	5.196e-1	21	41	3.347e-3	22	6700
CT/PET	2D	2.670e-2	7	275	9.332e-3	28	3024
ROI-CT	2D	1.430e-1	11	78	1.057e-2	32	3005
ROI-PET	2D	2.978e-3	6	1863	1.118e-3	4	3456
ROI-CT/PET	2D	2.634e-2	29	1107	1.450e-3	5	3710
CT	3D	4.698e-3	7	1490	4.367e-3	9	2061
PET	3D	1.563e-2	2	128	4.532e-3	15	3310
CT/PET	3D	2.521e-2	3	119	5.629e-3	14	2487
CT/PET-grad	3D	3.670e-2	4	109	4.801e-3	14	2916
CT/PET-20	3D	6.931e-2	7	101	3.880e-3	12	3093
CT/PET-30	3D	1.911e-2	3	157	5.165e-3	15	2904
CT/PET-40	3D	1.078e-2	4	371	2.078e-3	14	6738
CT/PET-50	3D	3.731e-3	6	1608	1.029e-4	1	9718

DSC values are significantly higher in ROI image sets for both EHA and BOS strategies. This is likely due to the ROI images containing greater representation of the tumor in each slice, avoiding complications related to class imbalances.

The highest mean DSC value was found in the ROI-PET and ROI-CT/PET models using BOS at 0.81. The models using EHA were comparable, demonstrating 0.77 and 0.79 mean DSC values for ROI-PET and ROI-CT/PET respectively. The main difference between the EHA and BOS strategies is in the TCT. The hyperparameters selected via BOS resulted in shorter TCT's compared to models implementing EHA, with the most noticeable difference seen in the CT/PET model, with a mean TCT observed at 19.2 minutes for EHA and 5.2 minutes for BOS.

Although greater DSC values were achieved in models trained with ROI images, it is worth noting that deriving all ROI image sets (including the images used for performance evaluation) require the ground truth mask. This means that practical implementation of an ROI image-based model has inherent limitations. Unseen images cannot be preprocessed in the same manner as evaluation images without available ground truth masks.

The results of the 3D models address these limitations. Table 4.6 and Figure 4.16 show the results of 3D 10-fold cross validation. In Table 4.6, EHA shows exceptionally low DSC values, with the highest mean DSC value achieved being 0.046 in the PET model. In contrast, the BOS models performed better with the highest mean DSC value of 0.76, which was achieved using CT/PET-grad images. This is on par with model performances seen in current literature [5, 36]. Lo Faso et al. performed a literature review of 14 studies that used deep learning methods to segment head and neck tumors [36]. Results showed a mean DSC value of 0.725 for bimodal CT/PET models.

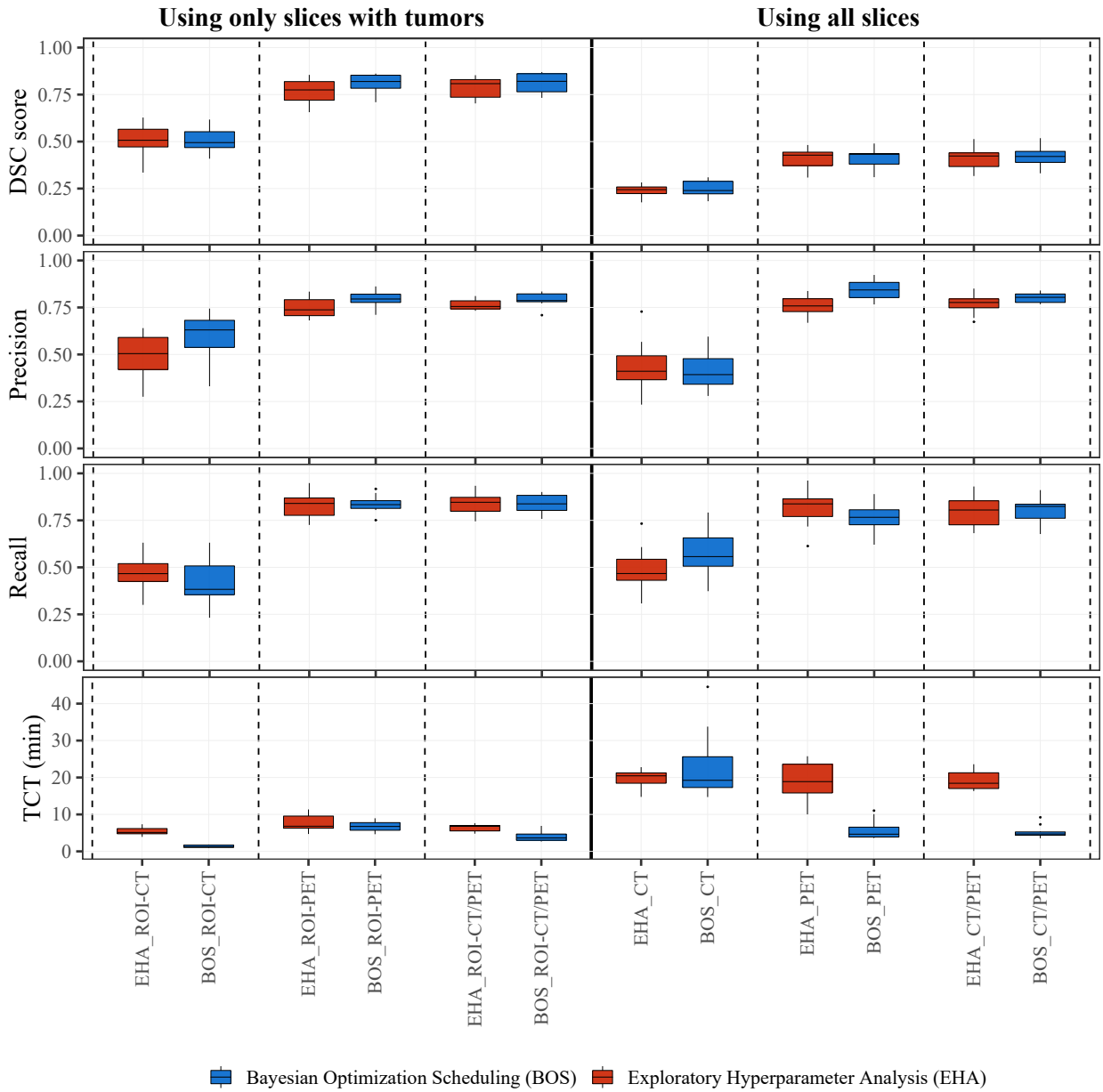

Figure 4.15

2D 10-fold Cross Validation Results

To provide better context as to whether or not this performance is comparable to manual segmentations, we can consider the DSC value among individual practitioners. Leibfarth et al.

found an average DSC value of 0.72 between three radiation oncologists who manually segmented head and neck tumors using PET/MR imaging [34]. Results showed a mean DSC value of 0.75 between automatic and manual segmentations, suggesting that the variation between practitioners is similar to the variation between the model and practitioner [34]. Further, similar DSC values can be observed between practitioners when manually segmenting anatomical head and neck organs [57]. Wong et al. found DSC values between radiation oncologists to be as low as 0.70, 0.69, and 0.72 for the spinal cord, parotid gland, and submandibular gland, respectively [57]. CT, CT/PET-40, and CT/PET-50 image modalities demonstrated relatively poor performance. These findings indicate that BOS is adaptable, selecting for appropriate hyperparameters when image modalities change as well as when working with 2D versus 3D images.

Further, 3D results highlight the importance of information found within PET images. CT images alone were unsuccessful in training models with acceptable performances. The DSC values of Figure 4.16 support the importance of PET imaging. PET, CT/PET, CT/PET-grad, and CT/PET-20 all show DSC values around 0.7 for BOS. The transition from CT/PT-20 to CT/PT-50 increases the threshold of what PET values are allowed to be displayed. PET-20 eliminates all voxel values which are below 20% of the maximum SUV. PET-50 eliminates even more voxels, voxels that are below 50% of the maximum SUV are not represented in the image. Figure 4.16 shows that DSC values for BOS CT/PET-30 contain a considerably high variation within its 95% confidence interval. This shows that for this image set, the model's ability to learn was extremely variable. It appears that excluding voxels which are below 30% of the maximum SUV can potentially provide the model with enough information to learn, but not always. CT/PET-40 and CT/PET-50 show that restricting SUV values to this extent results in significantly poorer performance.

Table 4.6

3D 10-Fold Cross Validation Results

Image modality	Strategy	DSC (95% CI)	Precision (95% CI)	Recall (95% CI)	TCT (95% CI)
CT	EHA	0.021 (0.013-0.029)	0.008 (0.005-0.011)	0.77 (0.65-0.88)	5.2 (2.3-8.2)
PET	EHA	0.046 (0.031-0.061)	0.021 (0.015-0.027)	0.98 (0.97-1.0)	9.2 (7.4-11.0)
CT/PET	EHA	0.045 (0.031-0.059)	0.019 (0.013-0.025)	0.97 (0.94-1.0)	7.9 (5.3-10.5)
CT/PET-grad	EHA	0.043 (0.029-0.057)	0.018 (0.012-0.024)	0.98 (0.95-1.0)	9.4 (7.7-11.0)
CT/PET-20	EHA	0.031 (0.026-0.036)	0.012 (0.010-0.014)	0.92 (0.88-0.97)	5.3 (3.5-7.1)
CT/PET-30	EHA	0.026 (0.018-0.034)	0.009 (0.006-0.012)	0.72 (0.57-0.87)	4.4 (2.0-6.8)
CT/PET-40	EHA	0.024 (0.018-0.030)	0.009 (0.006-0.012)	0.72 (0.62-0.83)	5.6 (4.0-7.2)
CT/PET-50	EHA	0.021 (0.015-0.027)	0.008 (0.005-0.011)	0.65 (0.50-0.80)	4.4 (2.5-6.3)
CT	BOS	0.023 (0.016-0.030)	0.008 (0.005-0.011)	0.74 (0.67-0.82	3.7 (2.0-5.4)
PET	BOS	0.71 (0.65-0.76)	0.68 (0.61-0.75)	0.69 (0.60-0.77)	9.2 (7.6-10.8)
CT/PET	BOS	0.73 (0.66-0.80)	0.72 (0.67-0.76)	0.70 (0.60-0.79)	7.8 (6.3-9.3)
CT/PET-grad	BOS	0.76 (0.71-0.82)	0.75 (0.70-0.80)	0.73 (0.64-0.83)	12.0 (10.1-14.0)
CT/PET-20	BOS	0.67 (0.61-0.74)	0.65 (0.58-0.72)	0.65 (0.57-0.72)	9.1 (8.8-9.5)
CT/PET-30	BOS	0.41 (0.16-0.66)	0.39 (0.13-0.65)	0.78 (0.68-0.88)	8.2 (5.8-10.6)
CT/PET-40	BOS	0.057 (0.026-0.088)	0.054 (0.0-0.12)	0.81 (0.74-0.88)	7.4 (4.7-10.1)
CT/PET-50	BOS	0.18 (0.015-0.35)	0.18 (0.0-0.38)	0.71 (0.60-0.82)	7.9 (6.1-9.7)

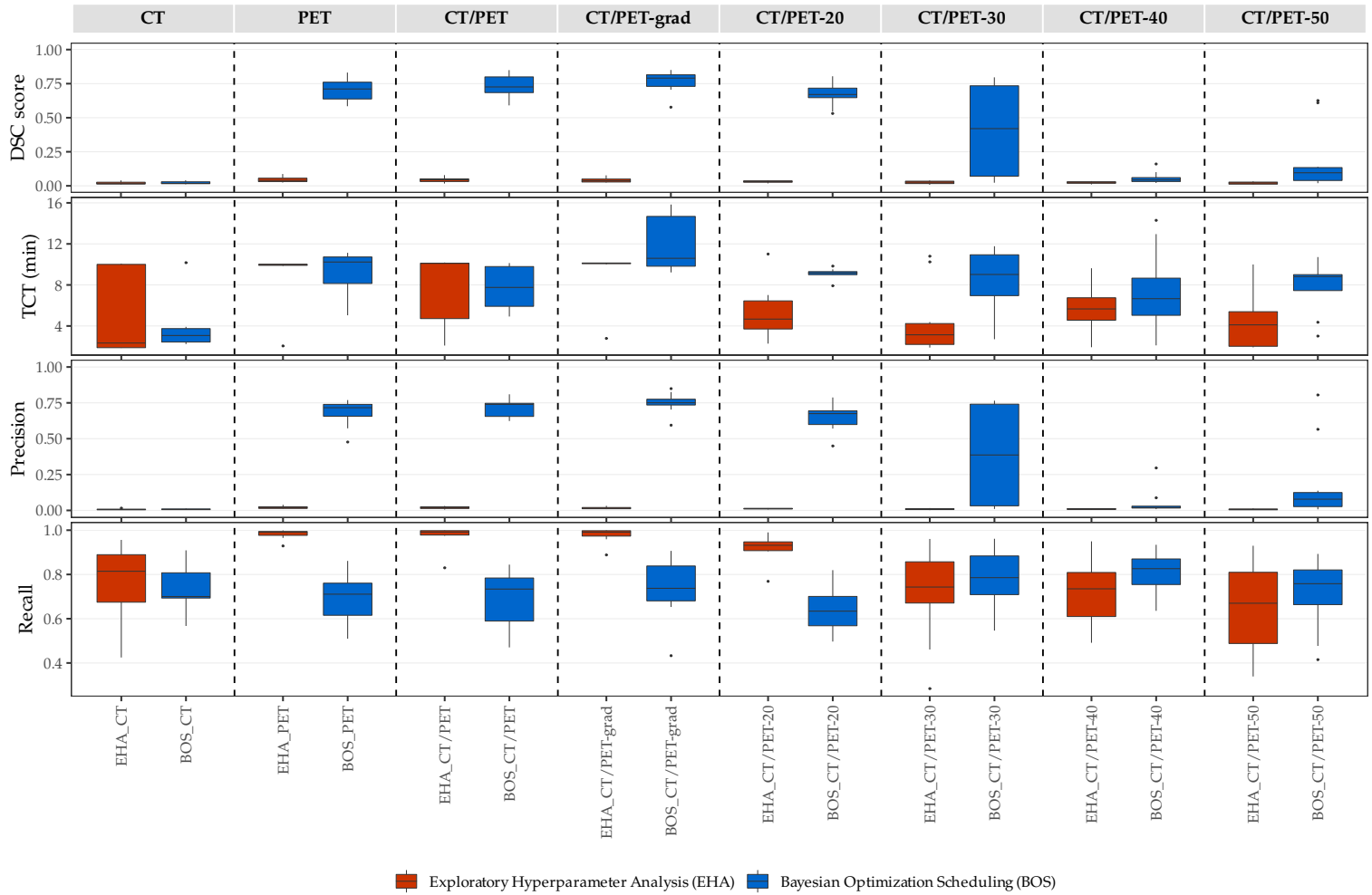

Figure 4.16

3D 10-fold Cross Validation Results

Table 4.7

Information for Patients Selected for Segmentation Predictions

Hospital Center	Patient ID	Tumor Slice Range	Age	Gender	T-Stage	N-Stage	M-Stage
CHGJ	041	37-65	58	Male	T3	N2b	M0
CHUS	095	27-53	65	Female	T4	N2	M0
CHMR	002	43-49	75	Male	T1	N1	M0
CHUM	055	57-71	56	Female	T4	N2	M0

4.5 3D BOS Segmentation Predictions

The following section illustrates the predicted segmentations by each model implementing the BOS strategy. The furthest left image is the original grayscale CT image overlayed with its respective original colored PET image. The dimensions of this image are 96×96×96. The ground truth for the tumor is segmented in red in this image as well as in the three subsequent images. The three subsequent images show a zoomed portion of the original image. Three images are shown to avoid overlapping and increase clarity. Different colors represent different models as denoted by the accompanying legend. Four patients are shown, one from each of the four hospital centers in the data set. Due to the tumor sizes of the patients selected from centers CHGJ and CHUS, every second slice is shown for each 3D image. All tumor slices are shown for patients selected from CHMR and CHUM. Table 4.7 summarizes information from the selected patients.

4.5.1 CHGJ - Patient 041

Figure 4.17 shows Patient 041 sampled from center CHGJ. Patient 041 is a 58 year old male with a tumor staging of T3, N2b, M0. The tumor is posterior to the patient's mandible and anterior to the cervical vertebrae. Each row represents a sagittal slice of the 96×96×96 image. Tumor slices are shown from the patient's right to left, starting with 37 and ending at 65. Due to the size of the tumor, every second slice is shown. Models PET, CT/PET, CT/PET-grad, CT/PET-20 and CT/PET-30 perform comparably for each slice. Models CT, CT/PET-40, and CT/PET-50 fail to predict tumors for several of the slices. These models only make predictions toward the center of the 3D image, and even then they lack a coherent boundary that could approximate the ground truth mask.

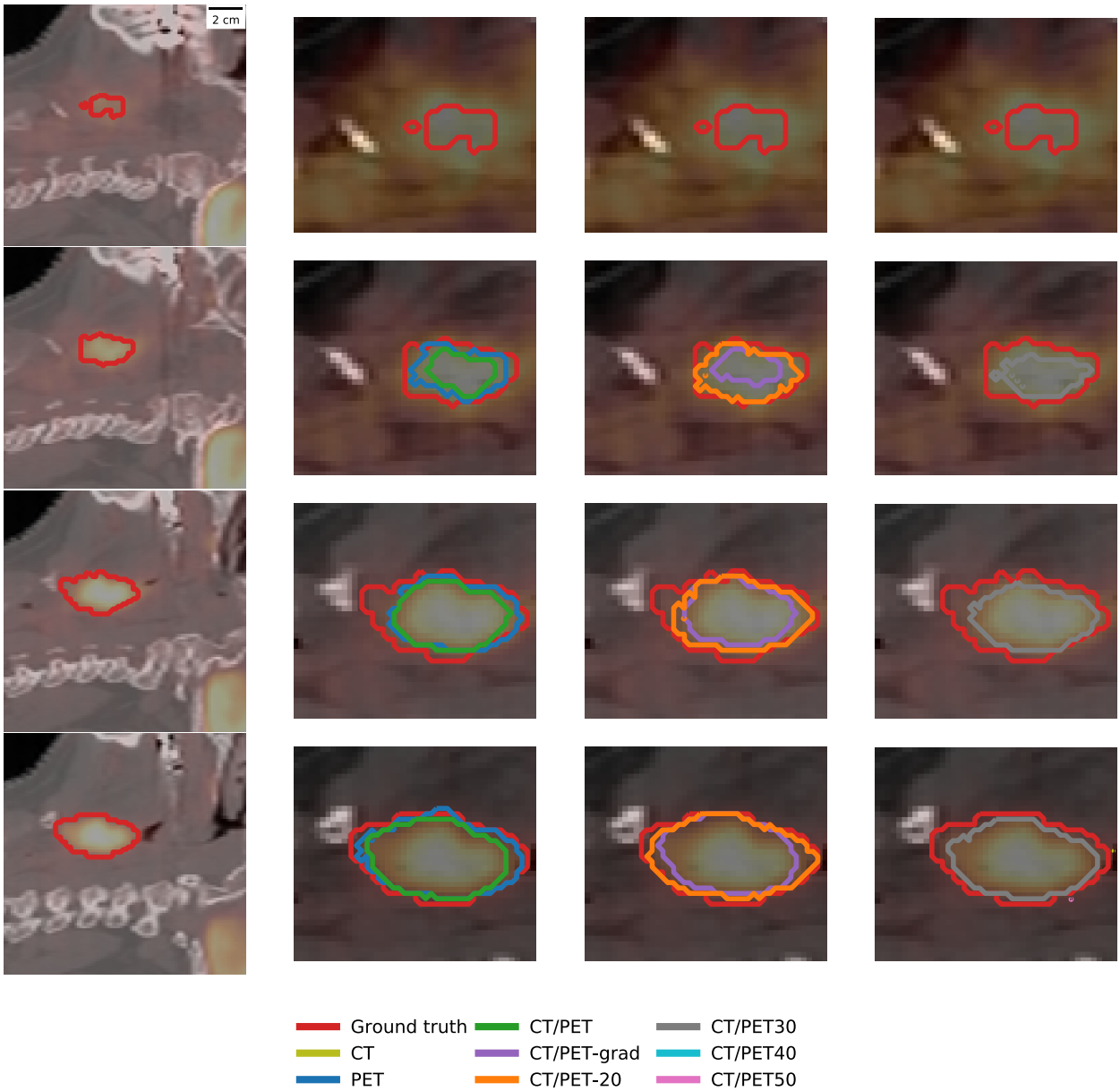

Figure 4.17

Comparison of segmentation predictions based on different inputs for Patient 041 of center CHGJ, a 58 year old male with T3, N2b, M0 tumor staging. Each row represents every second slice cut in the sagittal plane, from slice 37 to slice 65. The left most column shows ground truth tumor segmentations of the full image where as the three adjacent columns show a close-up view of prediction segmentations for each input.

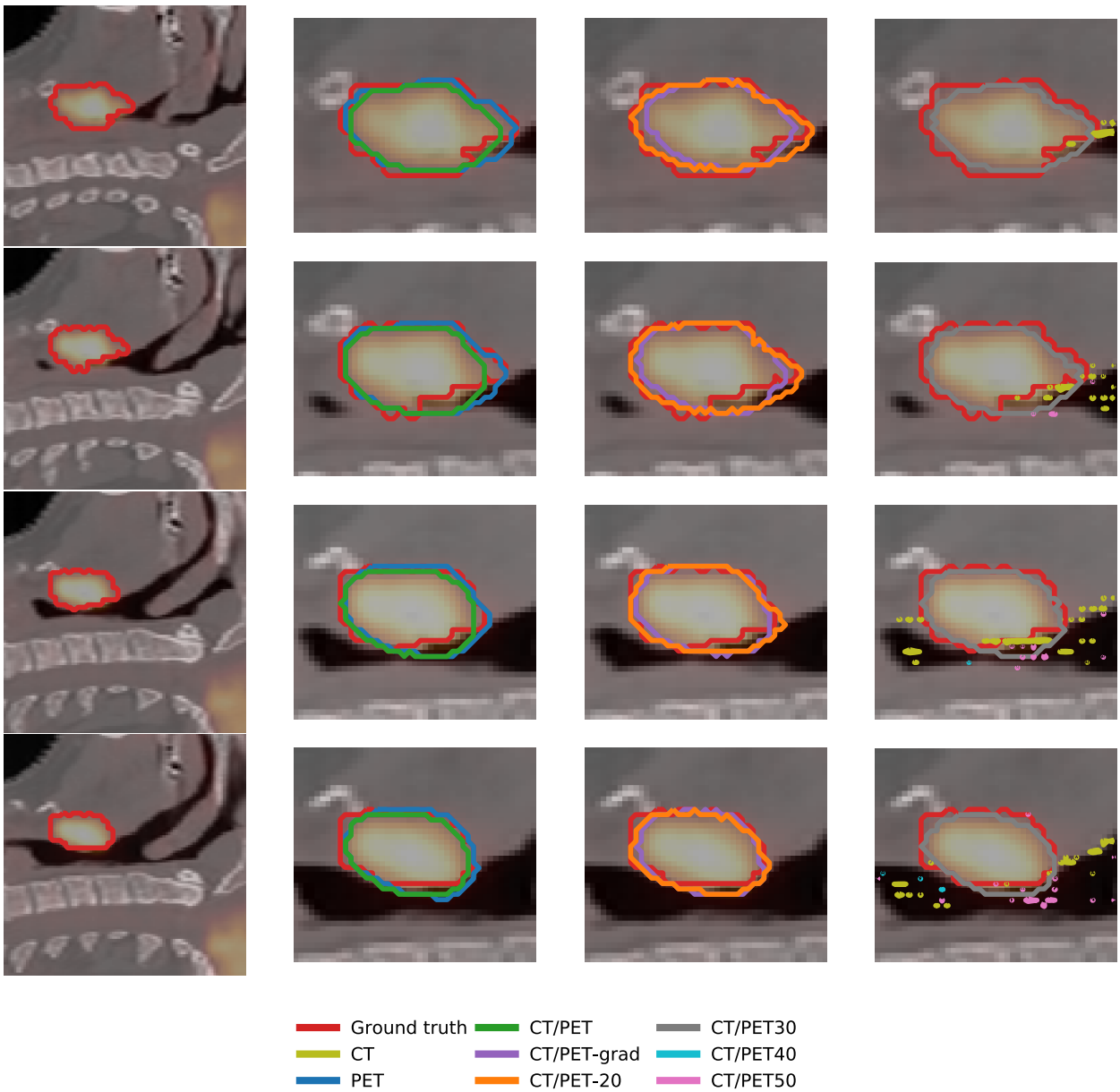

Figure 4.17 (continued)

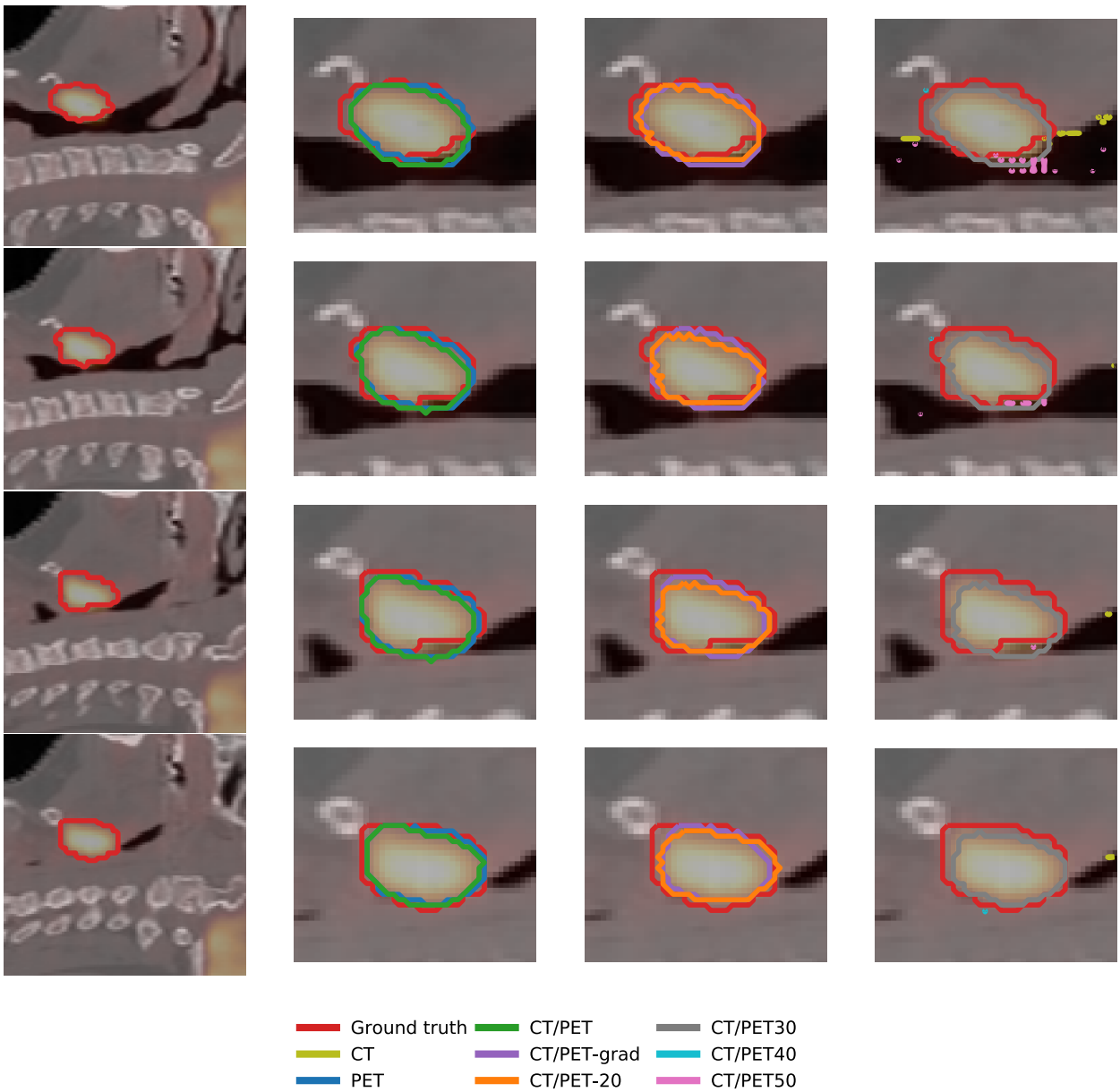

Figure 4.17 (continued)

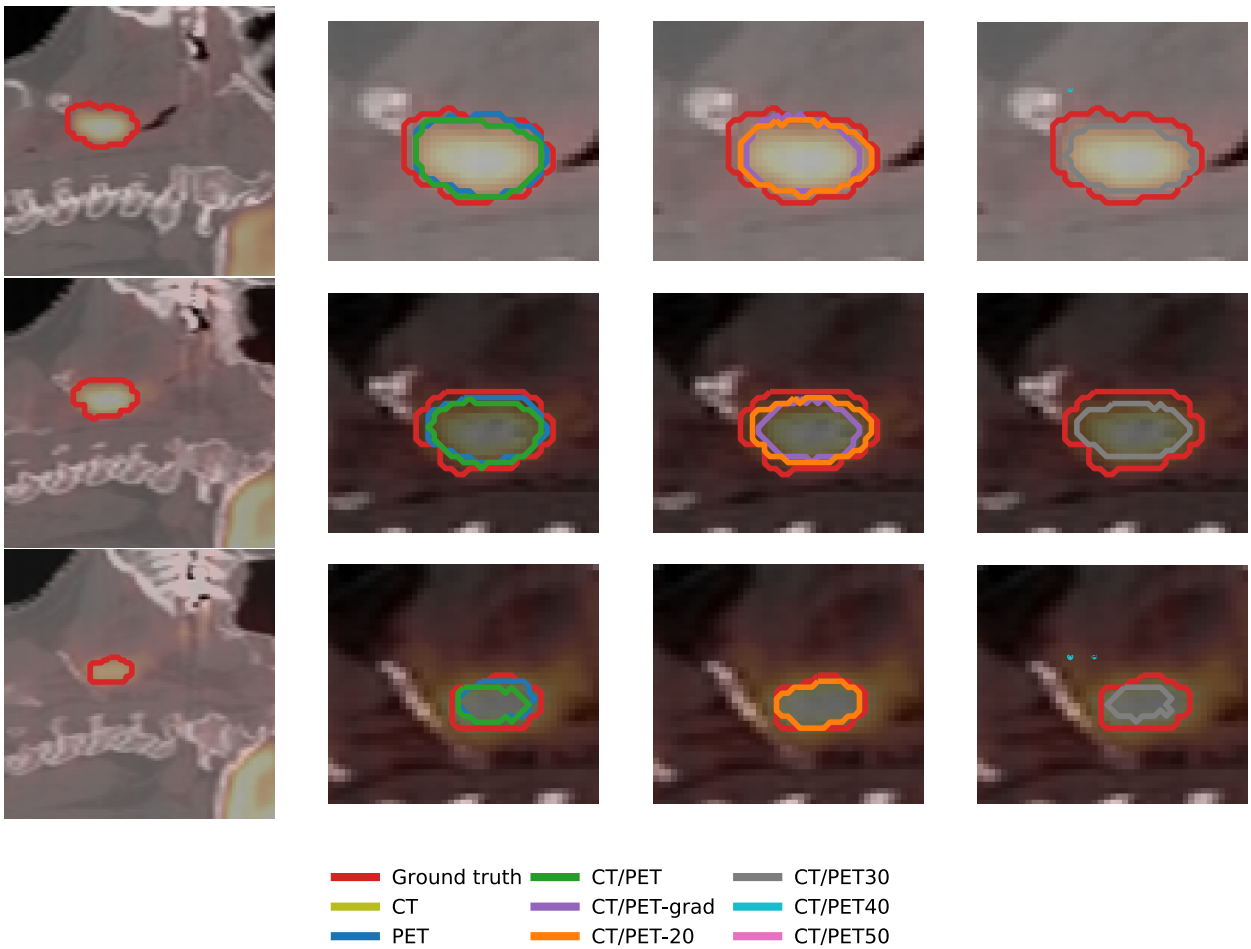

Figure 4.17 (continued)

4.5.2 CHUS - Patient 095

Figure 4.18 shows a similar trend in model performance when illustrating sagittal slices of Patient 095 from center CHUS. Like Patient 041 from CHGJ, PET, CT/PET, CT/PET-grad, CT/PET-20, and CT/PET-30 perform reasonably well, however there are alternative observations that can be noted. The tumor for this patient is larger than the tumor shown in Figure 4.17. The slices observed toward the center of the tumor, where the cross section is larger, show a different model behavior for CT/PET-grad. The segmentation for CT/PET-grad appears as a doughnut-shape, where the model fails to make tumor predictions for the center of the tumor. This is likely due to the PET-grad transformation being used to train this model since this transformation emphasizes edges. With a large enough tumor, image information associated with the tumor's center is lost with this particular type of PET transformation.

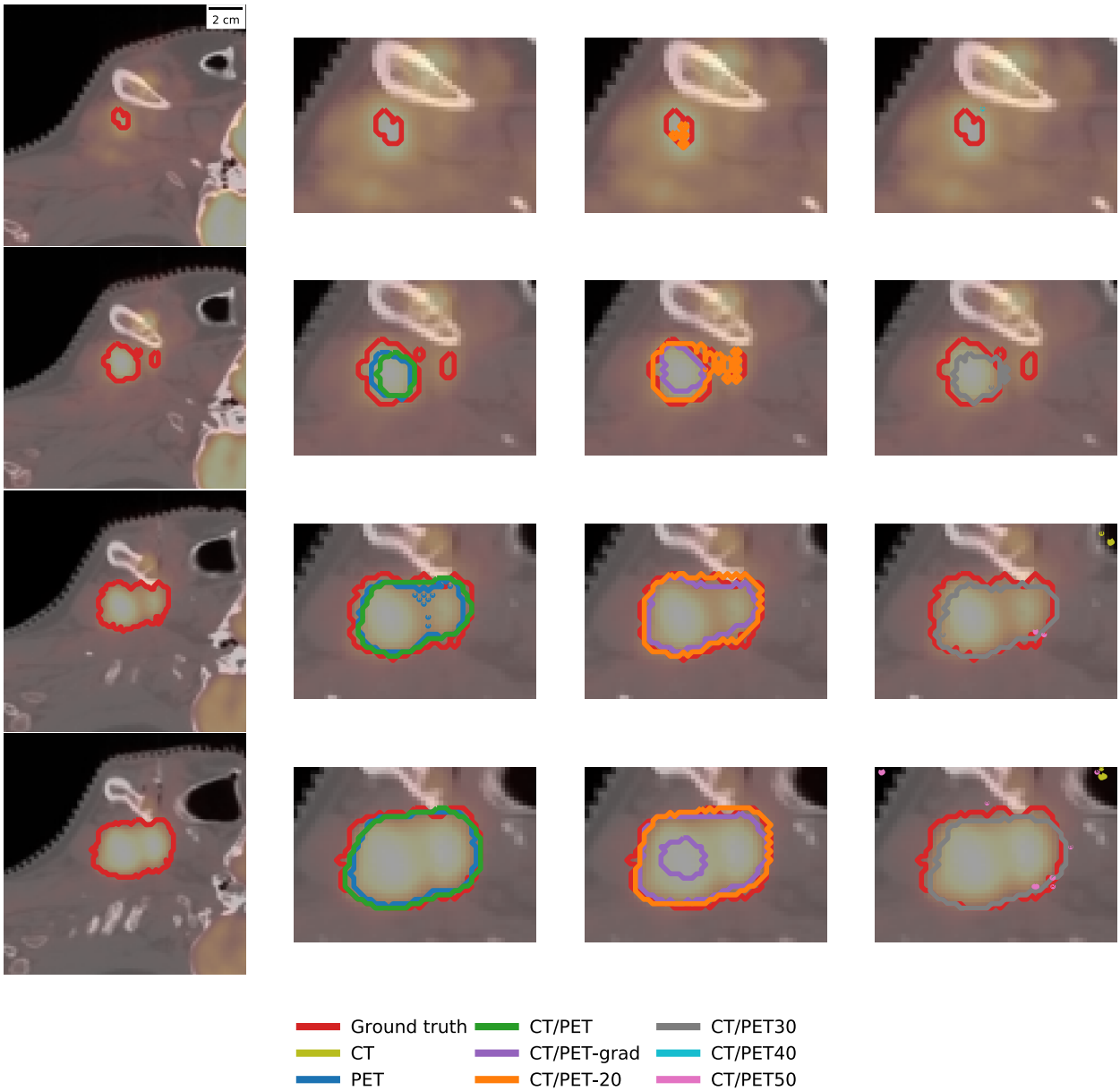

Figure 4.18

Comparison of segmentation predictions based on different inputs for Patient 095 of center CHUS, a 65 year old female with T4, N2, M0 tumor staging. Each row represents every second slice cut in the sagittal plane, from slice 27 to slice 53. The left most column shows ground truth tumor segmentations of the full image where as the three adjacent columns show a close-up view of prediction segmentations for each input.

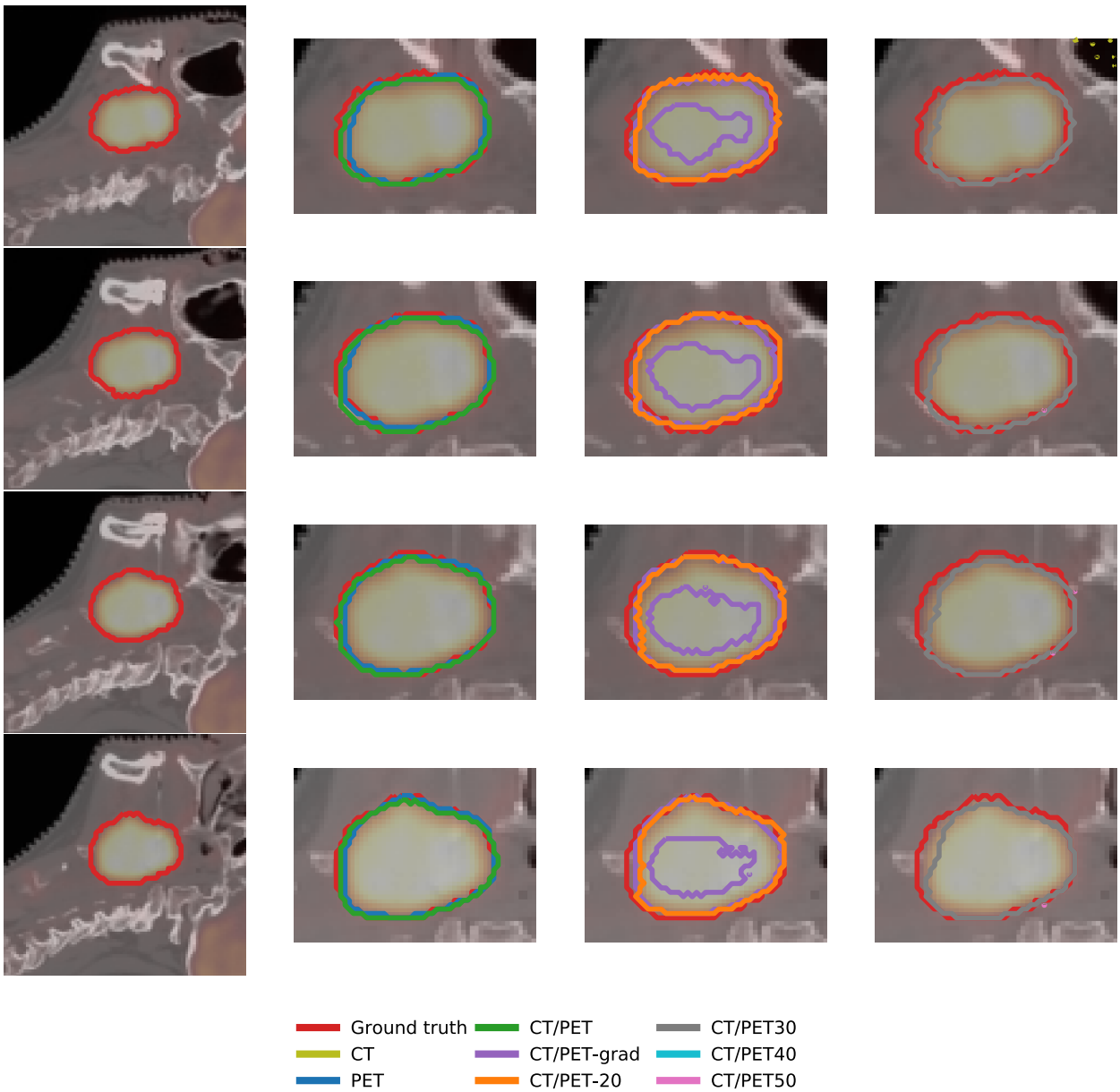

Figure 4.18 (continued)

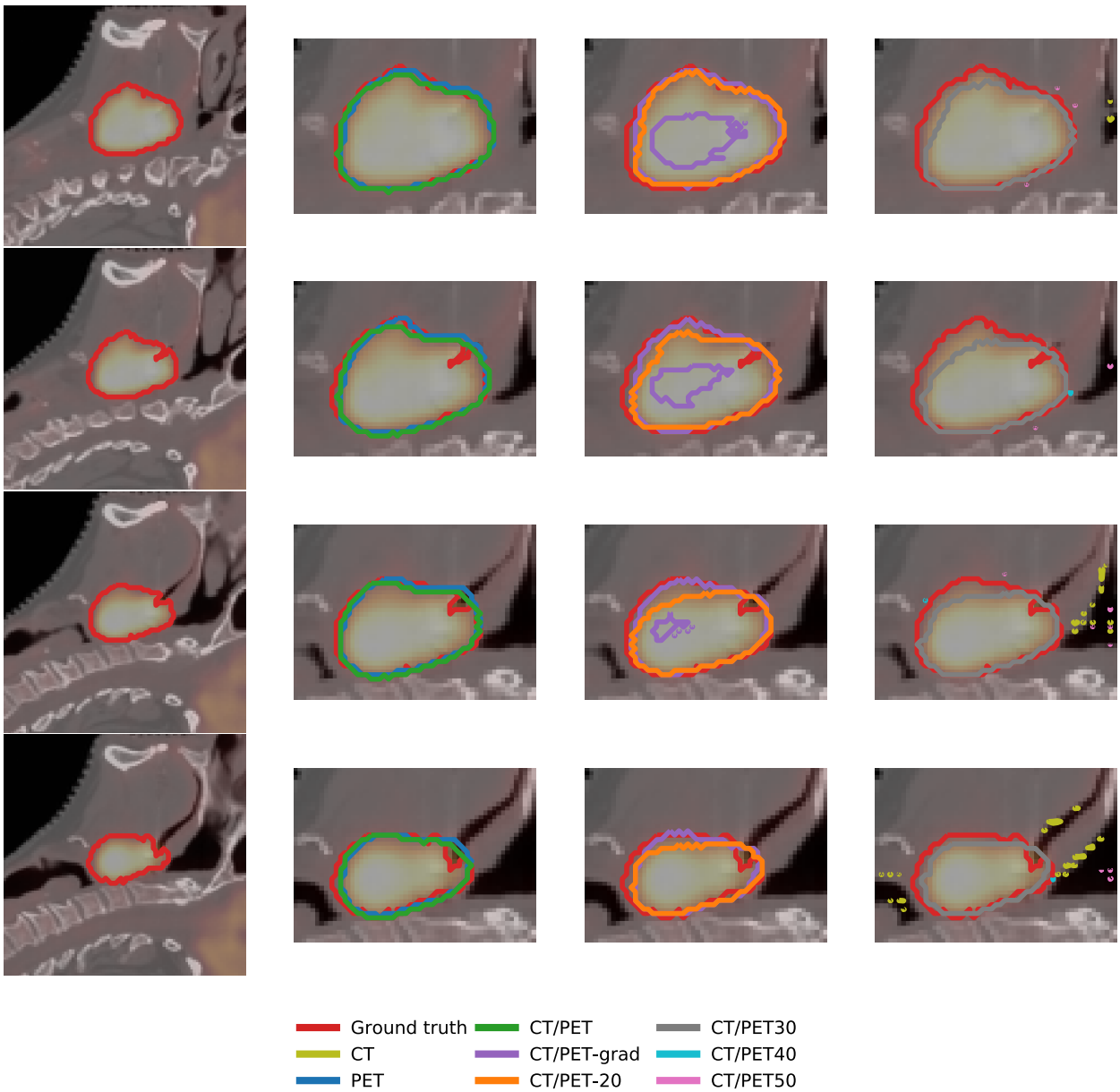

Figure 4.18 (continued)

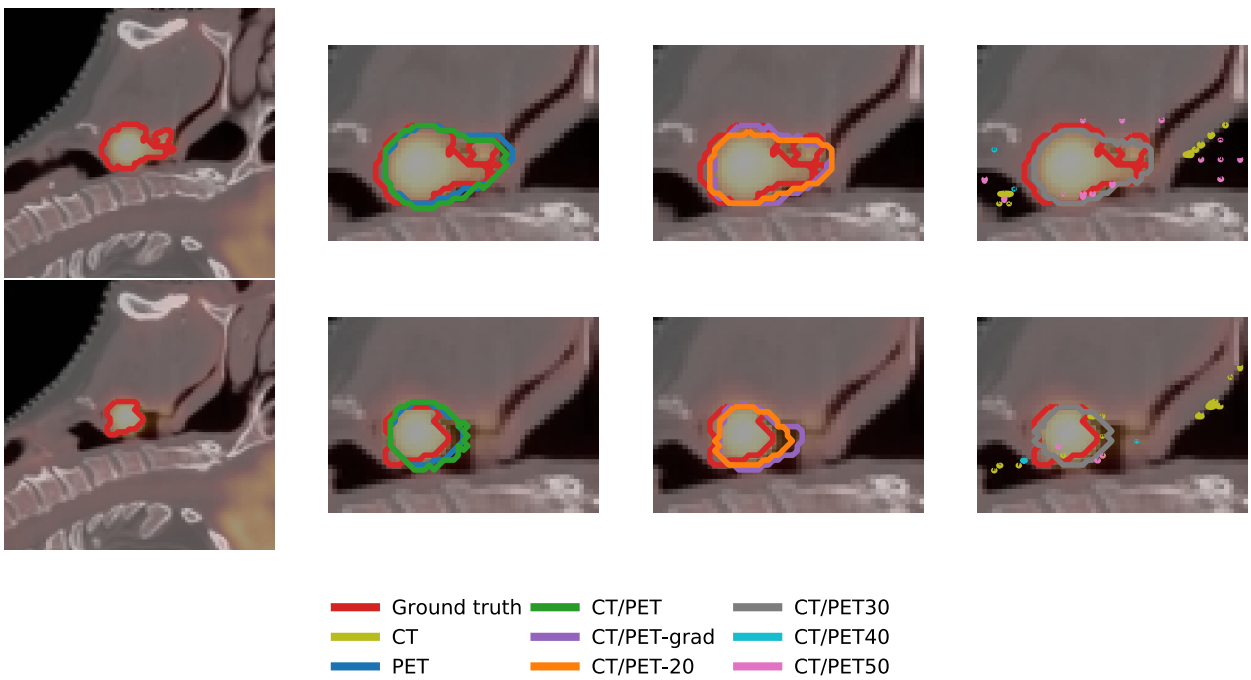

Figure 4.18 (continued)

4.5.3 CHMR - Patient 002

In Figure 4.19, segmentations are shown for a relatively smaller tumor for Patient 002 from center CHMR. It is speculated that PET-30 represents the point at which PET threshold models begin to degrade in performance. Table 4.16 shows the relatively smaller confidence interval of CT/PET-20 followed by the larger confidence interval of CT/PET-30. Then, from CT/PET-40 and CT/PET-50 we see a decline in DSC values along with smaller confidence interval ranges. In Figure 4.19 we can see this visually. At least for this example, a small enough tumor prevents CT/PET-30 from making any tumor predictions.

Up until this point, it was assumed that the PET model performed rather similarly to the CT/PET model. In Figure 4.19, we see the PET model visibly outperforming the CT/PET model. Previous results supported the idea that PET imaging information is more useful than CT imaging in order for the model to learn. The decreased performance of the CT/PET model in Figure 4.19 might indicate that CT information may not just be less useful, it may be disadvantageous and introduce unnecessary noise.

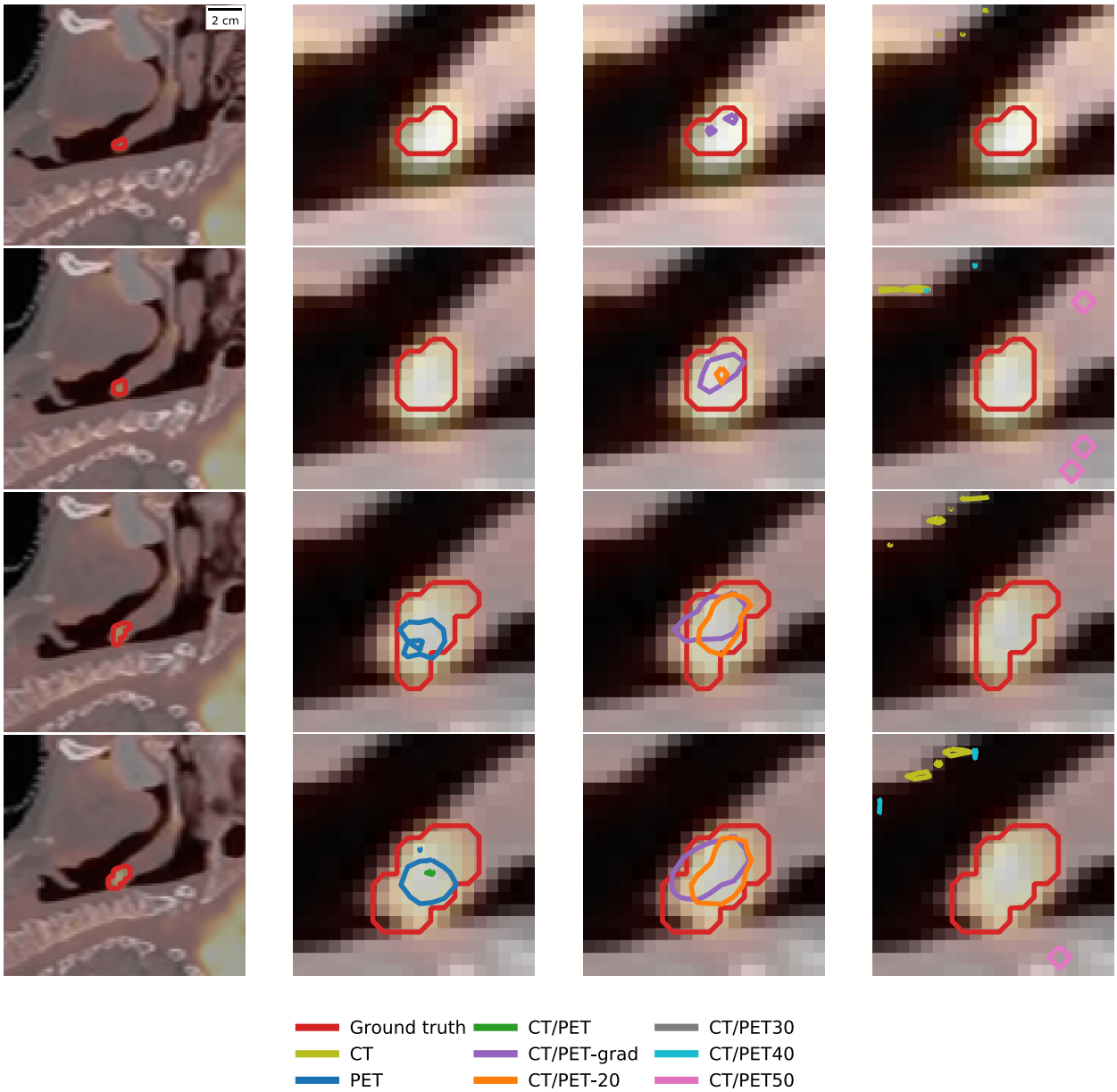

Figure 4.19

Comparison of segmentation predictions based on different inputs for Patient 002 of center CHMR, a 75 year old male with T1, N1, M0 tumor staging. Each row represents every slice cut in the sagittal plane, from slice 43 to slice 49. The left most column shows ground truth tumor segmentations of the full image where as the three adjacent columns show a close-up view of prediction segmentations for each input.

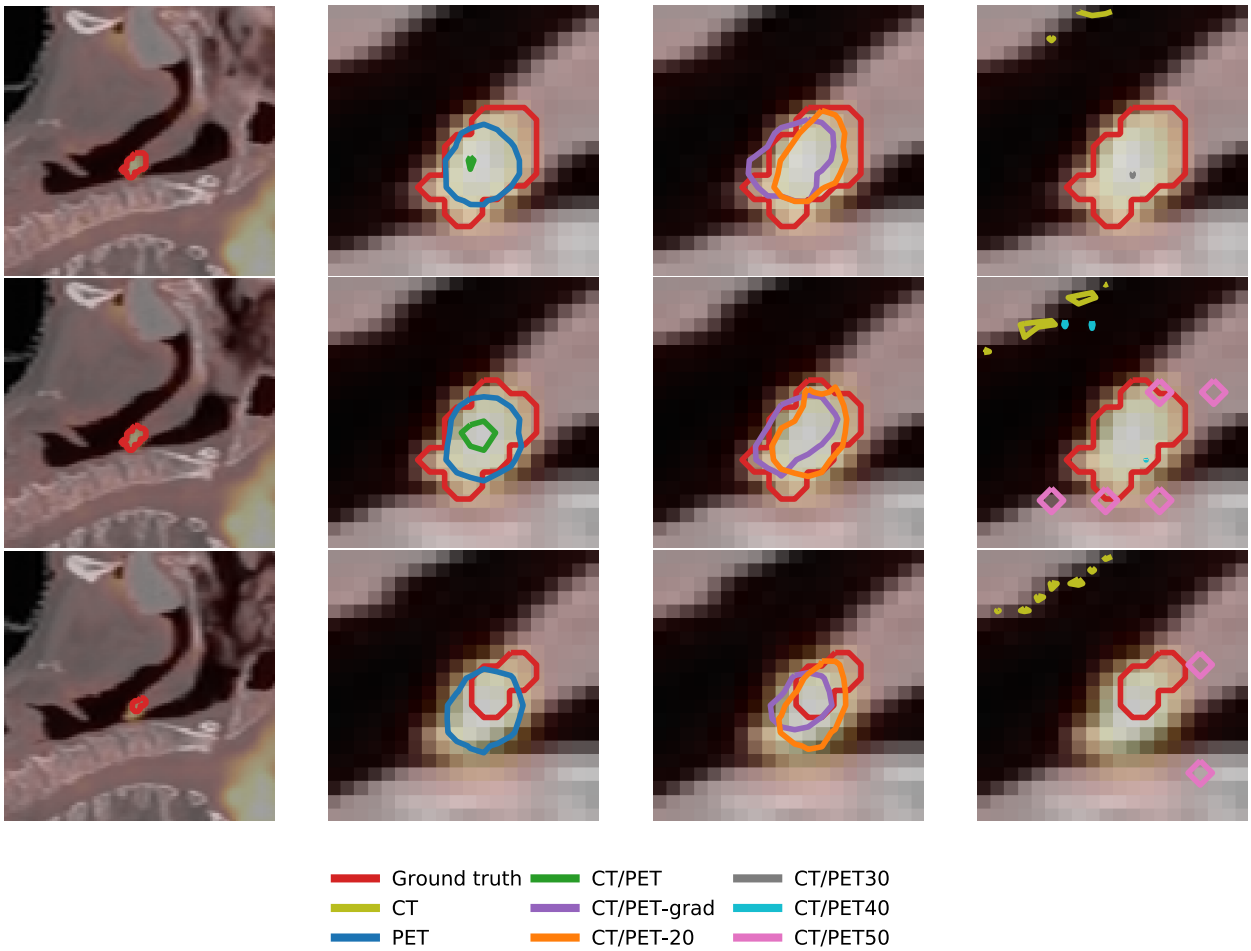

Figure 4.19 (continued)

4.5.4 CHUM - Patient 055

Patient 055 from center CHUM shows a unique ground truth tumor delineation in that the right-most side of the tumor (earlier rows) shows two separate tumors. Whether this was an annotation error remains to be unknown. If it were, then PET, CT/PET, CT/PET-grad, CT/PET-20, and CT/PET-30 models successfully labeled this region as one, intact tumor.

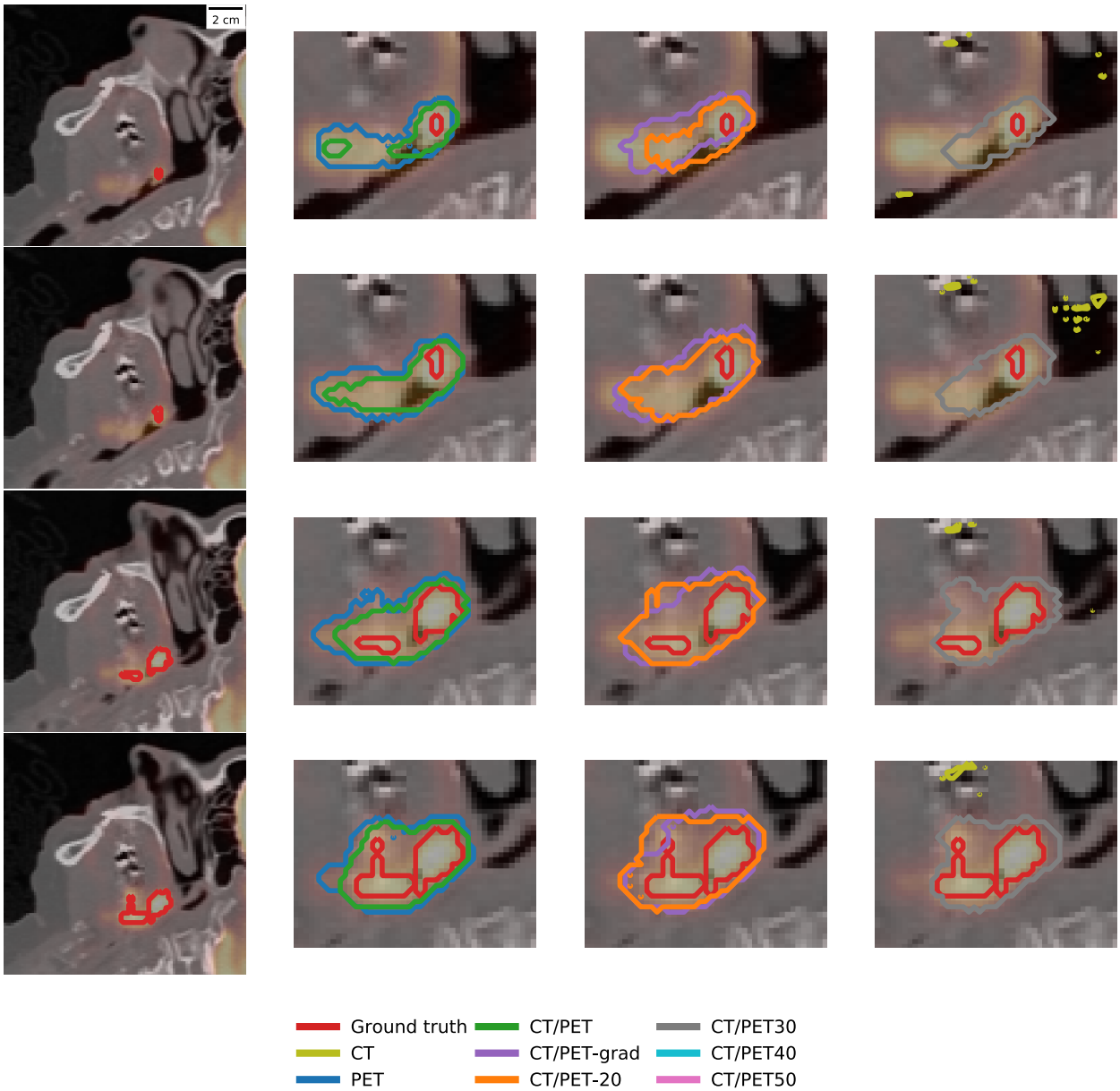

Figure 4.20

Comparison of segmentation predictions based on different inputs for Patient 055 of center CHUM, a 56 year old female with T4, N2, M0 tumor staging. Each row represents every slice cut in the sagittal plane, from slice 57 to slice 71. The left most column shows ground truth tumor segmentations of the full image where as the three adjacent columns show a close-up view of prediction segmentations for each input.

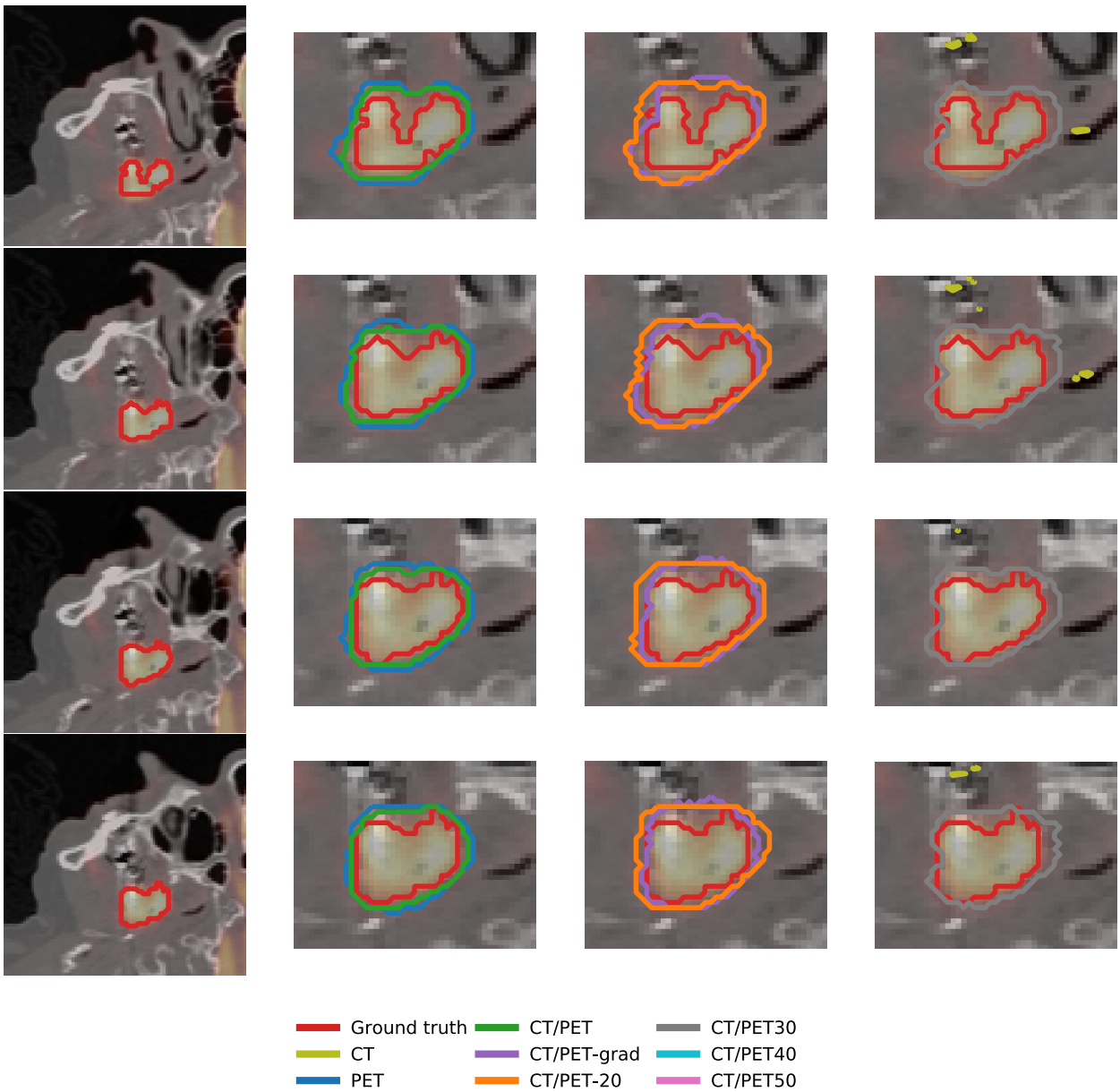

Figure 4.20 (continued)

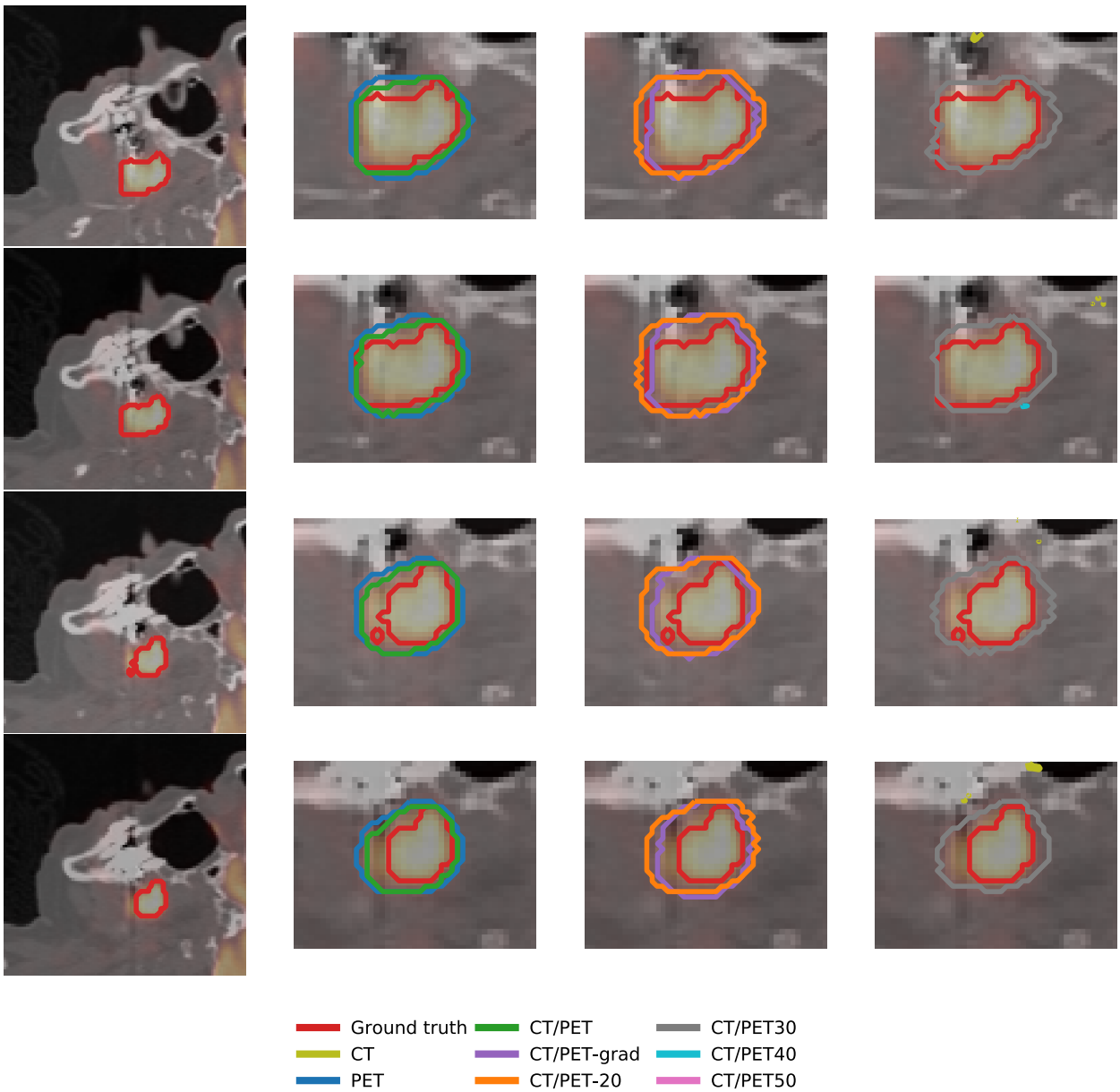

Figure 4.20 (continued)

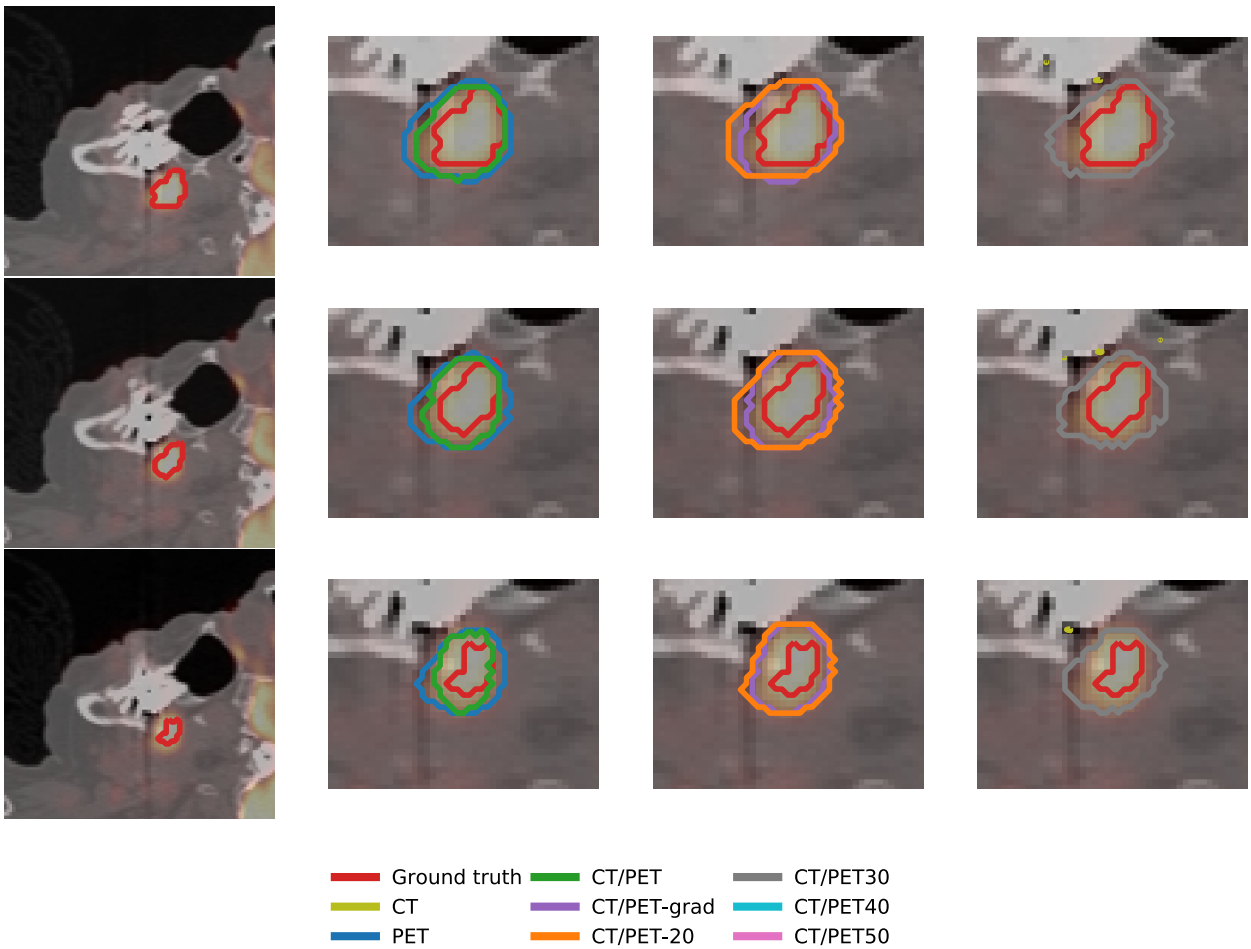

Figure 4.20 (continued)

CHAPTER V

CONCLUSION

With the prevalence of head and neck cancer and its associated complications, this disease remains an important challenge to both diagnose and treat [38]. Medical imaging provides practitioners with a critical tool to help aid in the detection and localization of tumors noninvasively [5]. Deep learning segmentation models hold enormous potential to contribute to this process [1, 5]. Using previously annotated CT and PET images, convolutional neural networks can be used to train and predict new images both in 2D and 3D format [5].

Multiple design choices can be considered when formulating deep learning models. The practicality and usefulness of these models is often a result of the culmination of these choices [23]. In this study, specific design choices are considered and describe two strategies, Exploratory Hyperparameter Analsysis (EHA) and Bayesian Optimization-derived Scheduling (BOS). EHA utilizes a grid search method to explore different hyperparameters such as batch size, learning rate, initialization, and optimizer. A given set of options for each hyperparameter is considered, and each combination is trained. Depending on the number of hyperparameters being explored, this task can be time intensive even when implementing a single image modality. Visual inspection of model behavior after training each combination can also be laborious.

BOS differs in that the strategy is more automatic. Using Bayesian optimization, optimal hyperparameters are found instead using conditional probability [45]. Hyperparameters are explored similar to EHA, but not in an exhaustive hyperparameter space. Instead, iterations are explored so that hyperparameter spaces which tend to yield greater model performance are favored [45]. BOS possesses limitations as well. The number of combinations needed for exploration increases significantly if the specific number of hyperparameters is not reasonably restricted.

This study restricts hyperparameters searches in BOS to batch size and learning rate, two hyperparameters which are incredibly vital to model performance [18]. To increase training flexibility, two distinct sets of hyperparameter combinations are allowed during training, one for phase one and another for phase two. This allows model training to better discriminate between hyperparameter values, ideally utilizing values that support loss function convergence at different periods in training progression.

Results show that for 2D-based models, EHA and BOS are comparable. There are slight advantages of BOS with respect to lesser training computation times (TCT). In order to achieve favorable performance for 2D images, both EHA and BOS strategies required preprocessing of CT and PET image data so that image slices always contained the presence of a tumor. Although this increase in model performance resulted in mean DSC values ranging from 0.77 to 0.81 when using PET images, using CT images alone kept mean DSC values at 0.51. An additional drawback to using slices that only contain tumors is that it prevents the use of future images, those for which the presence or absence of a tumor is completely unknown.

Best results were observed using 3D-based models. Fused images were used with CT and PET or CT and a PET variation obtained using different preprocessing transformations. PET variations

included PET-grad, a transformation resulting in gradient magnitudes, and different PET thresholds. PET thresholds were obtained by taking a percentage of each PET image's maximum SUV. The transformed image consequently retains whatever values are above this percentage threshold. For example, PET-20 contains voxel values which are 20% or more than the maximum SUV for each respective image.

PET, CT/PET, CT/PET-grad, and CT/PET-20 performed within the range of 0.67 to 0.71 for mean DSC values. Performance began to degrade at CT/PET-30, indicating that the information in PET imaging is likely responsible for adequate model performance. The low performances of CT (0.023), CT/PET-40 (0.057), and CT/PET-50 (0.18) support this conclusion. The adaptability of BOS is also noted as no EHA strategy achieved higher than a mean DSC value of 0.046 for 3D modeling.

5.1 Contributions

This study provides support for the existence of an optimal batch size to learning rate ratio. Previous studies have suggested the interdependence between batch size and learning rate, citing that decreasing the learning rate provides the same result as increasing the batch size [20, 25, 51]. Although He et al. emphasized reducing the batch size to learning rate ratio to improve generalization, they failed to caution that there is a limit to which this ratio should be reduced [20]. The optimal hyperparameter values determined via BOS show a consistent trend among all 14 models where a smaller batch size to learning rate ratio is desired for the start of training, while a larger batch size to learning rate ratio is preferential toward the end of training.

5.2 For Further Research

It was assumed in this study that a two phase training schedule would provide enough flexibility for the model to fully benefit from optimal batch size to learning rate ratios. Future research could investigate this assumption by implementing training schedules with more than two phases. Additionally, the batch size to learning rate ratio was only verified using the SGD optimizer. It is unclear whether an optimizer with an adaptive learning rate such as Adam would demonstrate similar behavior with respect to this ratio.

Practically, results show that there are significant limitations using CT imaging in isolation. More dedicated investigation for CT imaged-based models would be required if any implementation is to be considered in the case that CT imaging is available and PET imaging is not.

REFERENCES

[1] D. Abdelhafiz, J. Bi, R. Ammar, C. Yang, and S. Nabavi, "Convolutional neural network for automated mass segmentation in mammography," *BMC bioinformatics*, vol. 21, no. 1, 2020, pp. 1–19.

[2] T. Akiba, S. Suzuki, and K. Fukuda, "Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes," *arXiv preprint arXiv:1711.04325*, 2017.

[3] H. Alibrahim and S. A. Ludwig, "Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization," *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 1551–1559.

[4] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[5] V. Andrearczyk, V. Oreiller, M. Vallières, J. Castelli, H. Elhalawani, M. Jreige, S. Boughdad, J. O. Prior, and A. Depeursinge, "Automatic segmentation of head and neck tumors and nodal metastases in PET-CT scans," *Medical Imaging with Deep Learning*. PMLR, 2020, pp. 33–43.

[6] A. Bhandari, J. Koppen, and M. Agzarian, "Convolutional neural networks for brain tumour segmentation," *Insights into Imaging*, vol. 11, 2020, pp. 1–9.

[7] S. Bock and M. Weiß, "A proof of local convergence for the Adam optimizer," *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[8] Ö. Çelik, "A research on machine learning methods and its applications," *Journal of Educational Technology and Online Learning*, vol. 1, no. 3, 2018, pp. 25–40.

[9] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks," *IEEE Access*, vol. 8, 2020, pp. 52588–52608.

[10] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," *arXiv preprint arXiv:1910.05446*, 2019.

[11] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image segmentation using K-means clustering algorithm and subtractive clustering algorithm," *Procedia Computer Science*, vol. 54, 2015, pp. 764–771.

[12] H. Duanmu, J. Kim, P. Kanakaraj, A. Wang, J. Joshua, J. Kong, and F. Wang, "Automatic Brain Organ Segmentation with 3D Fully Convolutional Neural Network for Radiation Therapy Treatment Planning," *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2020, pp. 758–762.

[13] J. Fan, C. Ma, and Y. Zhong, "A selective overview of deep learning," *arXiv preprint arXiv:1904.05526*, 2019.

[14] L. Gao and Y. Ding, "Disease prediction via Bayesian hyperparameter optimization and ensemble learning," *BMC research notes*, vol. 13, no. 1, 2020, pp. 1–6.

[15] G. Gonella, E. Binaghi, P. Nocera, and C. Mordacchini, "Investigating the behaviour of machine learning techniques to segment brain metastases in radiation therapy planning," *Applied Sciences*, vol. 9, no. 16, 2019, p. 3335.

[16] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[17] S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh, "Bayesian optimization for adaptive experimental design: A review," *IEEE access*, vol. 8, 2020, pp. 13937–13948.

[18] I. R. I. Haque and J. Neubert, "Deep learning approaches to biomedical image segmentation," *Informatics in Medicine Unlocked*, vol. 18, 2020, p. 100297.

[19] D. Hartmann, D. Müller, I. Soto-Rey, and F. Kramer, "Assessing the Role of Random Forests in Medical Image Segmentation," *arXiv preprint arXiv:2103.16492*, 2021.

[20] F. He, T. Liu, and D. Tao, "Control batch size and learning rate to generalize well: Theoretical and empirical evidence," *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 1143–1152.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[22] K. Held, E. R. Kops, B. J. Krause, W. M. Wells, R. Kikinis, and H.-W. Muller-Gartner, "Markov random field segmentation of brain MR images," *IEEE transactions on medical imaging*, vol. 16, no. 6, 1997, pp. 878–886.

[23] H.-L. Hsieh and M. M. Shanechi, "Optimizing the learning rate for adaptive estimation of neural encoding models," *PLoS computational biology*, vol. 14, no. 5, 2018, p. e1006168.

[24] B. Ibragimov and L. Xing, "Segmentation of organs-at-risks in head and neck CT images using convolutional neural networks," *Medical physics*, vol. 44, no. 2, 2017, pp. 547–557.

[25] S. Jastrzębski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey, "Three factors influencing minima in sgd," *arXiv preprint arXiv:1711.04623*, 2017.

[26] J. Kalliola, J. Kapočiūtė-Dzikienė, and R. Damaševičius, "Neural network hyperparameter optimization for prediction of real estate prices in Helsinki," *PeerJ computer science*, vol. 7, 2021, p. e444.

[27] B. Kang and T. Q. Nguyen, "Random forest with learned representations for semantic segmentation," *IEEE Transactions on Image Processing*, vol. 28, no. 7, 2019, pp. 3542–3555.

[28] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," *arXiv preprint arXiv:1712.07628*, 2017.

[29] K.-r. Kim, Y. Kim, and S. Park, "A probabilistic machine learning approach to scheduling parallel loops with bayesian optimization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, 2020, pp. 1815–1827.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Improving Classification Attacks in IOT Intrusion Detection System using Bayesian Hyperparameter Optimization," *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE, 2020, pp. 146–151.

[32] A. Kurani, P. Doshi, A. Vakharia, and M. Shah, "A Comprehensive Comparative Study of Artificial Neural Network (ANN) and Support Vector Machines (SVM) on Stock Forecasting," *Annals of Data Science*, 2021, pp. 1–26.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[34] S. Leibfarth, F. Eckert, S. Welz, C. Siegel, H. Schmidt, N. Schwenzer, D. Zips, and D. Thorwarth, "Automatic delineation of tumor volumes by co-segmentation of combined PET/MR data," *Physics in Medicine & Biology*, vol. 60, no. 14, 2015, p. 5399.

[35] H. Li, M. Krček, and G. Perin, "A comparison of weight initializers in deep learning-based side-channel analysis," *International Conference on Applied Cryptography and Network Security*. Springer, 2020, pp. 126–143.

[36] E. A. Lo Faso, O. Gambino, and R. Pirrone, "Head–Neck Cancer Delineation," *Applied Sciences*, vol. 11, no. 6, 2021, p. 2721.

[37] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[38] W. Lv, S. Ashrafinia, J. Ma, L. Lu, and A. Rahmim, "Multi-level multi-modality fusion radiomics: application to PET and CT imaging for prognostication of head and neck cancer," *IEEE journal of biomedical and health informatics*, vol. 24, no. 8, 2019, pp. 2268–2277.

[39] F. Milletari, N. Navab, and S. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation. Proc-2016 4th Int Conf 3D Vision, 3DV 2016; 2016: 565–571,", 2016.

[40] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[41] D. L. Pham, C. Xu, and J. L. Prince, "Current methods in medical image segmentation," *Annual review of biomedical engineering*, vol. 2, no. 1, 2000, pp. 315–337.

[42] T. Qin, "Machine Learning Basics," *Dual Learning*, Springer, 2020, pp. 11–23.

[43] I. Roman, J. Ceberio, A. Mendiburu, and J. A. Lozano, "Bayesian optimization for parameter tuning in evolutionary algorithms," *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 4839–4845.

[44] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[45] C. Ruther and J. Rieck, "A Bayesian Optimization Approach for Tuning a Genetic Algorithm Solving Practical-Oriented Pickup and Delivery Problems," *IEEE Transactions on Automation Science and Engineering*, 2021, pp. 1–12.

[46] K. Sakthivel, R. Nallusamy, and C. Kavitha, "Color image segmentation using SVM pixel classification image," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 8, no. 10, 2015, pp. 1919–1925.

[47] M. H. Savenije, M. Maspero, G. G. Sikkes, J. R. van der Voort van Zyp, A. N. TJ Kotte, G. H. Bol, and C. A. T. van den Berg, "Clinical implementation of MRI-based organs-at-risk auto-segmentation with convolutional networks for prostate radiotherapy," *Radiation Oncology*, vol. 15, 2020, pp. 1–12.

[48] H. Seo, M. Badiei Khuzani, V. Vasudevan, C. Huang, H. Ren, R. Xiao, X. Jia, and L. Xing, "Machine learning techniques for biomedical image segmentation: An overview of technical aspects and introduction to state-of-art applications," *Medical physics*, vol. 47, no. 5, 2020, pp. e148–e167.

[49] T. Y. Shin, H. Kim, J.-H. Lee, J.-S. Choi, H.-S. Min, H. Cho, K. Kim, G. Kang, J. Kim, S. Yoon, et al., "Expert-level segmentation using deep learning for volumetry of polycystic kidney and liver," *Investigative and clinical urology*, vol. 61, no. 6, 2020, p. 555.

[50] S. Smith, E. Elsen, and S. De, "On the Generalization Benefit of Noise in Stochastic Gradient Descent," *International Conference on Machine Learning*. PMLR, 2020, pp. 9058–9067.

[51] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv preprint arXiv:1711.00489*, 2017.

[52] S. L. Smith and Q. V. Le, "A bayesian perspective on generalization and stochastic gradient descent," *arXiv preprint arXiv:1710.06451*, 2017.

[53] P. Sridhar, G. Mercier, J. Tan, M. T. Truong, B. Daly, and R. M. Subramaniam, "FDG PET metabolic tumor volume segmentation and pathologic volume of primary human solid tumors," *American Journal of Roentgenology*, vol. 202, no. 5, 2014, pp. 1114–1119.

[54] L. Tan, A. Liang, L. Li, W. Liu, H. Kang, and C. Chen, "Automatic prostate segmentation based on fusion between deep network and variational methods," *Journal of X-ray Science and Technology*, vol. 27, no. 5, 2019, pp. 821–837.

[55] A. Venmathi, E. Ganesh, and N. Kumaratharan, "Image segmentation based on Markov random field probabilistic approach," *2019 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2019, pp. 0490–0495.

[56] N. J. Wala'a and R. J. Mohammed, "A Survey on Segmentation Techniques for Image Processing," 2021.

[57] J. Wong, A. Fong, N. McVicar, S. Smith, J. Giambattista, D. Wells, C. Kolbeck, J. Giambattista, L. Gondara, and A. Alexander, "Comparing deep learning-based auto-segmentation of organs at risk and clinical target volumes to expert inter-observer variability in radiotherapy planning," *Radiotherapy and Oncology*, vol. 144, 2020, pp. 152–158.

[58] Z. Yang, F.-L. Chung, and W. Shitong, "Robust fuzzy clustering-based image segmentation," *Applied soft computing*, vol. 9, no. 1, 2009, pp. 80–84.

[59] K. You, M. Long, J. Wang, and M. I. Jordan, "How does learning rate decay help modern neural networks?," *arXiv preprint arXiv:1908.01878*, 2019.

[60] T. C. Zhang, J. Yang, J. P. Zhang, and J. Zhang, "SVM Methods in Image Segmentation," *Advanced Collaborative Networks, Systems and Applications*, 2016, pp. 62–65.