CYBERNETICS AND PHYSICS, VOL. 12, NO. 3, 2023, 201–206

DEEP LEARNING MUSCLE SEGMENTATION MODEL FOR CT IMAGES IN DICOM FORMAT

Ian Schmidt
Saint Petersburg State University
Russia
yanschmidt@mail.ru

Elena Kotina
Saint Petersburg State University
Russia
e.kotina@spbu.ru

Pavel Buev
Saint Petersburg State University
Russia
pavel.d3v@gmail.com

Article history:
Received 09.11.2023, Accepted 26.11.2023

Abstract

This work solves the problem of automatic segmentation of medical images in DICOM format using machine learning methods. A new developed tool is used in the form of a separate module for labeling medical data in the DICOM format. The trained model, proposed in the paper, can be useful in the tasks of muscle segmentation. One can apply it in different ways, but some of the most common include assessment of diseases related to muscles, and sarcopenia is one of them.

The further applications of the muscle segmentation model may include examining various medical cases with patients, that tend to have muscle-related diseases. For instance, detecting cachexia may be one of the extensions of the model's application field.

Key words

Medical imaging, computer tomography (CT), machine learning, DICOM, muscle segmentation

1 Introduction

Currently, a large number of various modality images are used in medical diagnostics, which require processing and analysis: segmentation [Shen et al., 2023], detection and classification [Salemi et al., 2022], calculation of diagnostically significant parameters [Ploskikh and Kotina, 2021]. Deep learning approaches are widely used for these purposes [Liu et al., 2021].

DICOM (Digital Imaging and Communications in Medicine) is a de facto standard that establishes rules for exchanging medical images (X-Ray, MRI, CT, SPECT, PET) and associated information between equipment from different suppliers, computers and hospitals.

DICOM files have a `.dcm` extension and provide means of storing information under individual tags. DICOM consists of a set of header and image data packaged into a single file. The header information is organized into a consistent and standardized series of tags. By extracting data from these tags, one can access important information regarding patient demographics, study parameters, and so on.

There are dosens of medical imaging softwares available to work with medical data, however some of them lack on pithiness and are overloaded with numerous functions, sometime tending to be redundant in special cases, when only few of them are truly needed. What is more, to create masks of ROI, that are to be used in machine learning tasks, one usually needs to write additional scripts to enable convertation of the coordinates array, forming a contour, into `.json` format. Our labeling module solves this problem, simultaneously yielding both `.dcm` and `.json` files of the current study.

Sarcopenia is a muscle disease, sometimes reffered as a confition, associated with loss of muscle mass, strength and functionality. It is believed that 5–15% of the population aged 60 and over 80 years have sarcopenia. Timely detection, correct diagnosis and treatment can affect the patient's subsequent condition, because the presence of sarcopenia increases the risk of falls and fractures, loss of the ability to cope with everyday activities, and also complicates the rehabilitation process after surgery.

It's important to have a model for segmenting muscle tissues, as then one will have the ability to calculate various indicators which will allow to construct different prognostic models, helping medical workers ease their everyday tasks and make diagnosing processes more accurate.

Constructing contours of medical images requires a lot of time, as well as suitable software. This work presents a manual contouring module that works with `.dcm` format files as input and, after constructing the contours, yields a corresponding `.dcm` file and a constructed mask, which is stored in `.json` format as an

array of points $(x, y) \in \mathbb{R}^2$ forming a polygon delimiting the ROI of the image.

Baseline data includes three-dimensional CT scans taken from internet sources [Roth et al., 2015], [Simpson et al., 2023], [Kirk et al., 2016], [Tong and Li, 2022]. After labeling the images using the proposed DICOM contouring module, a total of 288 `.dcm` images with corresponding muscle masks were available during training.

2 Problem Statement

The goal of this work is to build a neural network for processing and segmenting medical images in the DICOM format. As contouring is often done manually by radiologists, the process can be very time-consuming. That is why automation of the actions mentioned above will significantly reduce the time spent by specialists.

The modern trend towards creating systems and algorithms using artificial intelligence (AI) [Fradkov A., 2022] generates a large number of tasks, related to DICOM processing. There is currently a shortage of tools that would provide tools for PACS systems and DICOM contouring images, with the ability to export data and subsequent editing contours saved in the system. There are various tools that in either event perform data labeling. For example, in the CoreSlicer service described in article [Mullie and Afilalo, 2019], one can outline areas of the image and export circled areas, but they are saved in `.png` raster format, which makes it difficult to work with the original image. Another tool image viewer medDream [Med,] from Softneta allows create contours in video of geometric shapes and free contours, handdrawn, but does not allow you to export them. OHIF Viewer Project [OHI,] from the Open Health Imaging Foundation has a large variety of instruments, incl. marking the image with contours and allows you to export these contours in `.csv` format, but has limitations: contours can only be in video of geometric shapes - ellipse and rectangle. Thus, one of the tasks of this paper is to develop a separate module for la'beling images in `.dcm` format.

Once there is a muscle segmentation model it's possible to use it in assessing sarcopenia, which may be done by calculating different metrics related to muscle tissues density distribution and measuring various coefficients, that should be presupplied and recommended to be calculated by specialists and radiologists.

As a great amount of papers suggest, it's desirable to choose the axial scans at the level of the third *lumbar vertebra*, usually signed as L3 [Kim et al., 2020]. It is believed that the examination of only one scan at the L3 level can replace the study of the whole patient's body, reducing both dimentionality as well as time and cost of the study.

3 Contouring module

To label medical images, a module was developed that implements the functionality of the PACS system and provides tools for manual DICOM image contouring. This module is an information system that includes a PostgreSQL database server, a server application written in the Go language, a client application created using the React.js framework, and two nginx web servers for accessing the server and client applications. All of the above components are containerized in the Docker virtualization system for easy deployment of the system on the user's machine. User access to the system is through a graphical interface in a web browser.

The developed system has the following modules:

Objects list module. This module is implemented as a table that displays the fields of an object and has pagination settings. Above the table there is a navigation chain that allows the user to return to the previous page of the object. Navigation chain levels differ according to the level of the DICOM object hierarchy: "Study" $\rightarrow$ "Series" $\rightarrow$ "Instance".

Object view module. When a row in the table is clicked in the objects list module, the page for viewing the object located in the selected row opens. If the selected study or series contains children (for example, "Study" contains "Series" objects and "Series" contains "Instance" objects), a similar table of child objects will appear on the object view page.

DICOM image viewer module. The Cornerstone.js library was used to develop this module. The module can receive DICOM images in accordance with the WADO standard, display them and provide some basic image manipulation tools (zooming, panning, window width and center adjustment, viewing a series of images). Also in this module are tools for working with contours - creating, deleting, saving and exporting.

DICOM file upload module. The module loads studies into the system using the STOW specification of the DICOMweb standard. For each selected file, the module sends a corresponding request to the server.

In order to get a file with a markup, you need to select the required study instance in the system, select the contours using the appropriate tool in the DICOM image viewer, and use the export button. The system will send two files to the user, a DICOM image file with a `.dcm` extension and a contours file with a `.json` extension. Using these files, you can create a DICOM image mask without converting to a bitmap format, which can cause data loss.

An example of working with contouring module is shown below on Figure 1.

4 Data preprocessing

During the neural network training process, the supplied data from the training set is preprocessed to achieve the best training results. The image processing scheme

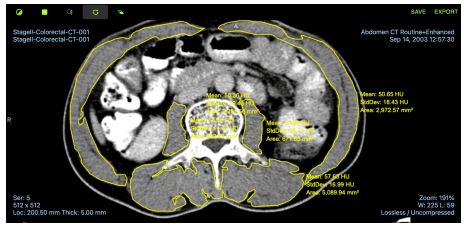

Figure 1. An interface of contouring module

can be represented as a sequence of the following actions:

1. extracting an array of image pixels from a `.dcm` file,
2. scaling the image using RescaleIntersept and RescaleSlope, presupplied by the `.dcm` file,
3. applying windowing to a scan,
4. augmentation of the image and the corresponding muscle mask with subsequent conversion of the `numpy` array into tensor,
5. normalizing the image and its mask.

Only after the actions have been completed, the data is fed to the input of the neural network during the learning process. It is important to note that during model training, work is done with arrays of `.dcm` images. It would be possible to use the corresponding `.png` images of the pictures instead, but in this case a lot of priceless for the neural network information would be lost, because DICOM files are encoded with 16 bits, while `.png` images are compressed to 8 bits encoding. As a result, the 2^{16} shades of gray in the original image are reduced to 2^8 in the `.png` image, which may result in the loss of valuable information for the neural network, which it can take into account during the training process.

Scaling. To correctly interpret information on medical images, it is necessary to first make additional transformations that will make the image more "readable" for a specialist. Pixel values extracted from a DICOM file often do not correspond to tissue density. Thus, in order to solve this problem, you need to apply a linear transformation on the pixel values, taking into account RescaleIntercept and RescaleSlope tags, stored in a `.dcm` file. These tags are determined by the hardware manufacturer.

Windowing. DICOM images have a wide HU range, and a windowing process is used to allow people to see the corresponding structures in the image. Windowing functions can be understood as some means of manipulating HU values in order to change the appearance of the image in order to ensure the selection of certain structures and tissues of a patien. The window has 2 values: W — window width, and L — window level. A wider window will allow one to observe a larger range of pixel values, which will result in greater contrast in the image. The window level specifies the midpoint of the window width and corresponds to the pixel value that will be located in the middle of the entire considered HU interval. The window level determines the brightness of the image. The values of W, L for windows allowing visualization of muscle tissue vary in the depending on the specific task and data. It was experimentally established that preprocessing of images with the following training of the neural network should be carried out with window values of W = 200, L = 80. A window with exactly these parameters will allow muscle tissue to be best highlighted, while intestines and bones will be clearly separated, and fatty and soft tissues will not be visualized at all.

Normalization. Having normalized data as input is very important to obtain accurate results from the neural network during the training phase, as it affects the procces of optimization (RMSProp, NAdam, etc.) ensuring that gradients will fluctuate substantianally. Usually the goal is to ensure that the values are enclosed in interval $[0, 1]$. During the training process, the normalize function of the `pytorch` library was used to normalize the images.

Data augmentation. To increase the number of training examples, data is augmented during the training process. Augmentation means the application of random transformations on data, such as changes in brightness or contrast, adding noise, random mirror shifts, reflections and rotations.

The use of augmentation makes it possible to make the model more robust to natural noise and interference in the data.

5 Dataset overview

The dataset which was used while training the model contains 288 `.dcm` files and corresponding masks in `.png` format. Volume CT studies were taken from different sources [Roth et al., 2015], [Simpson et al., 2023], [Kirk et al., 2016], [Tong and Li, 2022]. In each study the scan at the level of L3 was manually chosen and labeled in the proposed contouring module, which yieled `.dcm` and `.json` of a study. Additional code was written to extract an array of points from `.json` file and perform further creation of a `.png` binary mask, consisting of zeros and ones, where ones indicate pixels of a mask.

Exploratory Data Analysis (EDA). Before training the model it's important to know various data characteristics in order to obtain stable learning process. An EDA was carried out including calculating distribution of muscle densities on scans as well as distribution of means and standart deviations of muscle densities throughout the whole dataset. A metadata `.csv` file was created in order to store valuable metainformation that

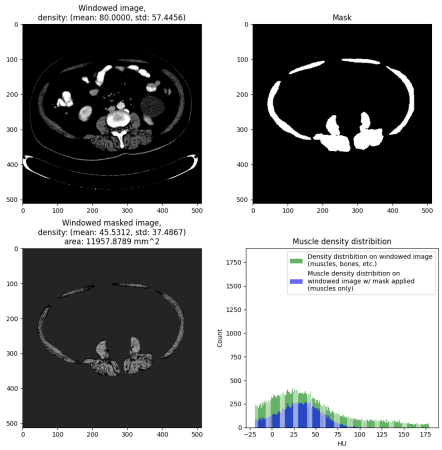

Figure 3. An example of a patient with low amount of dense muscles. Taking into consideration the area and mean/std values, one may conclude, that this patient is prone to sarcopenia

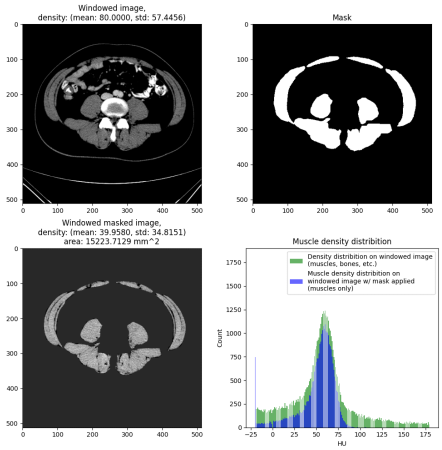

Figure 4. A diametrically opposed to Figure 3 example of a patient with well-developed dense muscles with high muscle area. There's much more pixels with high HU values that on Figure 3

may shad light on the structure of the data. The metadata file is a dataframe with the following columns: PatientAge, PatientSex, MuscleDensityMean, MuscleDensityStd and MuscleArea where MuscleDensityMean and MuscleDensityStd correspond to the mean and standard deviation values of muscle densities on a given scan respectively, while MuscleArea is the are of the muscles measured in mm^2. Other values were extracted from the `.dcm` file of the study.

The majority of studies included patients 50 years of age and older, where the amount of men prevailed over the number of women. There wasn't found any significant correlation between the age and the mean and standart deviation of muscle densities values of patients, but the EDA showed that men tend to have greater area of muscles compared to women. One should notice, that there is some positive correlation between muscle density statistics and the muscle area (see Figure 2). It indicates, that in many cases when the muscle area is relatively low, mean and std values are not big eather. This realization plays the key role in furher applications of muscle segmentation model in medical tasks, as there is a possibility to predict the probability of sarcopenia via numerous muscle characteristics, such as muscle area, mean and standard deviation values of muscle density. On Figure 3 one can see, that there isn't much pixels with high HU values, meaning that there is a little percentage of dense muscles on this image, and a patient is prone to have a muscle-related diseases, while on Figure 4 there is a great amount of pixels with large HU values, indicating high density of muscle groups.

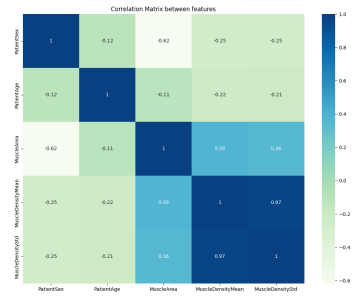

Figure 2. Correlation matrix between features in the metadata

6 Model overview

Architecture. As for the model, the U-Net architecture was chosen. Medical scans in `.dcm` format of size $(512, 512)$ were used from a labeled dataset containing 288 scans. Masks were build via proposed contouring module.

Each step contraction and expansive paths have two $3\times$ 3 convolutional layers, followed by BatchNorm2d and ReLU. In the original paper U-Net [Ronneberger et al., 2015] was used with 0 padding, but we use 1 so that final feature map is not cropped.

In the contracting path the feature map is down-sampled with a 2×2 max pooling layer. Then, at the up-sampling, the feature map is up-sampled with a 2×2 up-convolution. After Conv2d, BatchNorm2d and ReLU Dropout is applied with $p = 0.4$.

At every step in the expansive path the corresponding feature map from the contracting path is concatenated with the current feature map.

The training was carried out with `batch size` $=$ 10, learning rate equals 0.00075 in the beginning and changes throughout the learning process. The model has 53 layers and 117,489 trainable parameters. The model was built from scratch.

Metrics. The *Dice coefficient* (Sørensen–Dice coefficient) was chosen as the main metric for training the

Table 1. Metrics on test set

accuracy	precision	recall	IoU	specificity
0.91	0.89	0.83	0.72	0.76

Table 2. Main segmentation metrics. $\mathrm{TP}, \mathrm{TN}, \mathrm{FP}, \mathrm{FN}$ — True Positive, True Negative, False Positive and False Negative.

Metrics	Values
TP	$\sum_{i=1}^{\ell}[y_i = 1][a(x_i; w, w_0) = 1]$
TN	$\sum_{i=1}^{\ell}[y_i = 0][a(x_i; w, w_0) = 0]$
FP	$\sum_{i=1}^{\ell}[y_i = 0][a(x_i; w, w_0) = 1]$
FN	$\sum_{i=1}^{\ell}[y_i = 1][a(x_i; w, w_0) = 0]$
Dice	$\frac{2\lvert Y_{pred]} \cap Y_{GT}\rvert}{\lvert Y_{pred}\rvert + \lvert Y_{GT}\rvert} = \frac{2TP}{2TP + FP + FN}$
accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
precision	$\frac{TP}{TP + FP}$
recall	$\frac{TP}{TP + FN}$
IoU	$\frac{TP}{TP + FP + FN} = \frac{\lvert Y_{pred} \cap Y_{GT}\rvert}{\lvert Y_{pred} \cup Y_{GT}\rvert}$
specificity	$\frac{TN}{TN + FP}$

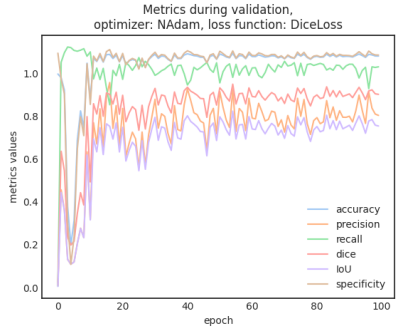

Figure 5. Metrics during validation

model and assessing the quality of segmentation:

$$\mathrm{Dice}(Y_{GT}, Y_{pred}) = \frac{2|Y_{GT} \cap Y_{pred}|}{|Y_{GT}| + |Y_{pred}|}, \qquad (1)$$

where Y_{GT} is the correctly marked mask on the object in question (GT is ground truth), and Y_{pred} is the prediction of the U-net. Other metrics such as accuracy, precision, sensitivity (recall), IoU and specificity were also calculated.

Accuracy shows the proportion of correctly predicted mask pixels to all predictions; precision refers to the proportion of correct positive predictions, or what fraction of pixels assigned to a mask class actually belongs to it, while recall (sensitivity) shows what part of the mask pixels were detected by the algorithm.

In medicine, the metrics specificity, sensitivity are accepted as standards for assessing segmentation. In mask pixel prediction problems, sensitivity, which is the same as recall and TPR (True Positive Rate), mainly focuses on the ability to make TP predictions, while specificity, also known as TNR (True Negative Rate) evaluates the ability to correctly identify TN cases. The sensitivity metric is often used in segmentation problems, but is less sensitive than F-measure-based metrics such as Dice and IoU. However, specificity can lead to misinterpretation of segmentation results if the content of this metric is misunderstood [Müller et al., 2022]. Specificity reflects the ability of the model to recognize the minor class of the image.

The list of metrics and ways to calculate them are listed in Table 2. In this table, $a(x_i, w, w_0)$ denotes an answer of the model and $[\cdot]$ stands for the binary indicator.

Loss function. DiceLoss was used as a loss fucntion,

$$\mathrm{DiceLoss}(Y_{pred}, Y_{GT}) = 1 - \mathrm{Dice}(Y_{pred}, Y_{GT}) = \\ = 1 - \frac{2|Y_{pred} \cap Y_{GT}|}{|Y_{pred}| + |Y_{GT}|}. \qquad (2)$$

Optimizer. RMSProp (running mean square) was chosen to carry out optimization process. Its implementation can be found in `pytorch`. RMSProp is defined as

$$w := w - \eta \mathcal{L}_i'(x) \oslash (\sqrt{G} + \varepsilon), \qquad (3)$$

where $G := \alpha G + (1 - \alpha)\mathcal{L}_i'(x) \odot \mathcal{L}_i'(x)$ is a vector of squared partial derivatives ($\oslash, \odot$ — coordinate-wise division and multiplication of vectors respectively).

7 Results

Learning for 100 epochs on GPU the U-net model managed to achieve results showing over 0.80 in Dice coefficient, which was chosen to be the main metric. Other metric values calculated on test set are listed in Table 1.

An example of U-net muscle tissue prediction is shown below in Figure 6. It worths mentioning, that the resulting mask one can see in Figure 6 in the lower right is the result of applying a sigmoid function to round predicted mask's values to 0 and 1 just in purpose of visualisation. Those pixels, in which U-net is not much confident to mark them as pixels truly belonging to the mask may be rounded and interpreted in the incorrect way when the specific threshold is not chosen. In the paper the threshold was figured out to be $0,5$ however it is possible to find a better value in the future explorations.

One should keep in mind this fact when building a classifier above muscle segmentation model. Thus, it is recommended to use non-rounded predictions of a model,

as classificator can take an advantage of those pixels that are not classified with sufficient confidence.

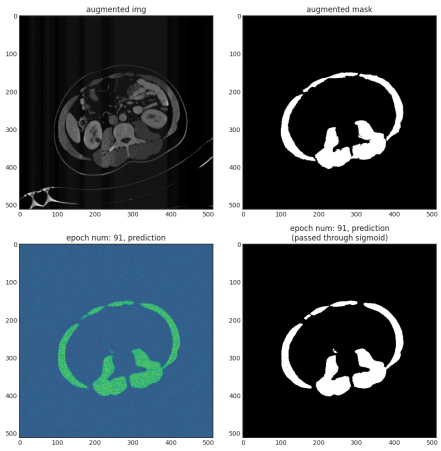

Figure 6. An example of U-net prediction

The process of model training is shown on Figure 7 indicating the change of loss during training and validation sections. It's possible to train a better model on richer dataset with less epochs, which is the subject of further experiments.

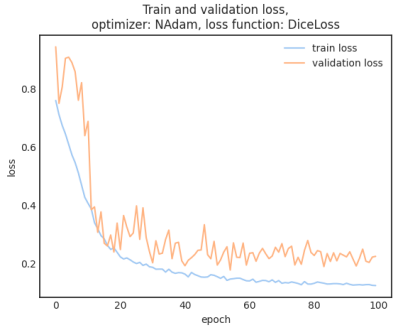

Figure 7. Training loss

8 Conclusion

To sum up everything said above, the results of this work include the trained U-net model for muscle segmentation and the DICOM-contouring module. Furthermore, they can be used to label `.dcm` files of different modalities, building masks of ROI and exporting them as `.dcm` and corresponing `.json` of a mask, not only for CT, but, for instance, MRI, as the only condition of usage is the `.dcm` extention of the study.

It's important to note that the model and contouring module can be used in medical tasks, such as sarcopenia, to carry out tasks of extracting muscle tissues from the raw scans. Further tasks may include calculating metrics, suggested by specialists, which might have a potential of helping in diagnosing.

References

Meddream html5 dicom viewer — softneta. https://www.softneta.com/products/meddream-dicom-viewer/. Accessed: 2023-11-05.

Open health imaging foundation. https://ohif.org/. Accessed: 2023-11-05.

Fradkov A., S. A. (2022). The history of cybernetics and artificial intelligence: a view from saint petersburg. *Cybernetics and Physics*, **11** (4), pp. 253–263.

Kim, S., Kim, T.-H., Jeong, C.-W., Lee, C., Noh, S., Kim, J. E., and Yoon, K.-H. (2020). Development of quantification software for evaluating body composition contents and its clinical application in sarcopenic obesity. *Scientific Reports*, **10** (1).

Kirk, S., Lee, Y., Sadow, C. A., Levine, S., Roche, C., Bonaccio, E., and Filiippini, J. (2016). The cancer genome atlas colon adenocarcinoma collection (tcga-coad).

Liu, X., Song, L., Liu, S., and Zhang, Y. (2021). A review of deep-learning-based medical image segmentation methods. *Sustainability*, **13** (3), pp. 1224.

Müller, D., Soto-Rey, I., and Kramer, F. (2022). Towards a guideline for evaluation metrics in medical image segmentation. *BMC Research Notes*, **15** (1).

Mullie, L. and Afilalo, J. (2019). CoreSlicer: a web toolkit for analytic morphomics. *BMC Medical Imaging*, **19** (1).

Ploskikh, V. and Kotina, E. (2021). Challenges of gated myocardial perfusion SPECT processing. *Cybernetics and Physics*, **10** (3), pp. 171–177.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. **9351**, pp. 234–241.

Roth, H., Lu, L., Seff, A., Cherry, K. M., Hoffman, J., Wang, S., Liu, J., Turkbey, E., and Summers, R. M. (2015). A new 2.5 d representation for lymph node detection in ct (ct lymph nodes).

Salemi, S., Behzadi-Khormouji, H., Rostami, H., Keshavarz, A., Keshavarz, Y., and Tabesh, Y. (2022). Incremental deep learning training approach for lesion detection and classification in mammograms. *Cybernetics and Physics*, **11** (4), pp. 234–245.

Shen, T., Huang, F., and Zhang, X. (2023). CT medical image segmentation algorithm based on deep learning technology. *Mathematical Biosciences and Engineering*, **20** (6), pp. 10954–10976.

Simpson, A. L., Peoples, J., and Creasy (2023). Preoperative ct and survival data for patients undergoing resection of colorectal liver metastases (colorectal-liver-metastases).

Tong, T. and Li, M. (2022). Abdominal or pelvic enhanced ct images within 10 days before surgery of 230 patients with stage ii colorectal cancer.