Exercise 2 report: Detection of Liver Tumors

Linoy Elimelech (ID. 319122610), Ron Azuelos (ID. 207059114)

AI for Healthcare course with Ayelet Akselrod-Ballin, Reichman University, 2023

1 Introduction

The goal of our project is to provide an overview of the second part of the course material, in which we study deep learning healthcare algorithms such as classification, detection, segmentation, and natural language processing (NLP). In this project, we investigated the identification of liver cancers in medical images. You may view our code in the following Google Colab notebook – https://colab.research.google.com/drive/1xMdZ7qyvXx-ubQRGNp5OYOGj-J5ouCRd?usp=sharing

1.1 The Problem

A critical but challenging area of medical imaging and healthcare is the detection of liver cancer. The early diagnosis of liver tumors is vital for effective treatment and improved outcomes for patients because they can have serious effects on their health. Radiologists' manual assessment of medical pictures, which can be time-consuming, subjective, and sensitive to human error, is frequently used in traditional methods of detecting liver cancer.

Deep learning algorithms, particularly those based on convolutional neural networks (CNNs), have shown great promise in automating and improving the detection of liver tumors. These algorithms leverage the power of deep learning to analyze large volumes of medical images and identify subtle patterns and features associated with tumors. By training these algorithms on annotated datasets, they can learn to distinguish between normal liver tissue and tumor regions, enabling accurate and efficient detection.

Through this project, we have developed real-world expertise using medical picture datasets, put cutting-edge deep learning algorithms into practice, and assessed the effectiveness of our models. By gaining knowledge and skills in this project, we will be better able to comprehend deep learning (DL) Healthcare algorithms and how they are used in medical imaging, which will pave the way for future developments in the field of liver tumor detection and healthcare in general.

1.2 Tools

The code was written in Python using the Google Colab Pro platform. We used an existing YOLOv7 model that is available in the following Git repository https://github.com/WongKinYiu/yolov7.

2 The Data

We were requested to use the data supplied by Medical Segmentation Decathlon [1, 2], which contains 201 contrast-enhanced CT images from patients with primary cancers and metastatic liver disease. Each CT image is a 3D scan of a NIfTI file, which is a compressed binary file that can be processed into 2D images, each representing the slices of the CT, and the number of slices changes between scans. Figure 1 describes the number of slices in each 3D scan.

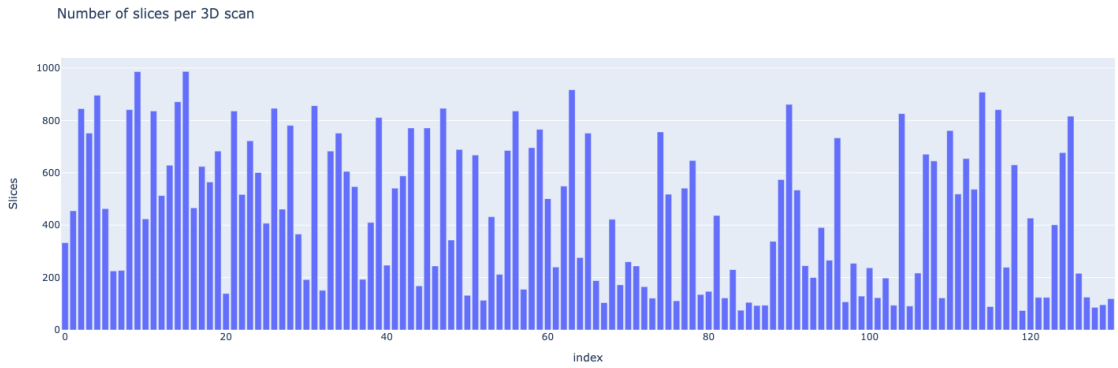

Figure 1: Number of slices per 3D scan

Each 3D scan is represented by a 3D matrix of size 512x512xS, where S is the number of slices. The data is split into train data in the size of 131 scans, and test data in the size of 70 scans. In addition, the data include label files, that have the same format as the CT images. Each 3D scan has a matching label file that contains the markings of the regions of interest (ROIs) in each 2D file, which are the background, liver, and tumors. A label file is represented as a 3D matrix of the same size as the scan, that contains the values 0, 1, or 2, with respect to the types of the ROI above. An example of a slice and its matching label file can be seen in Figure 2.

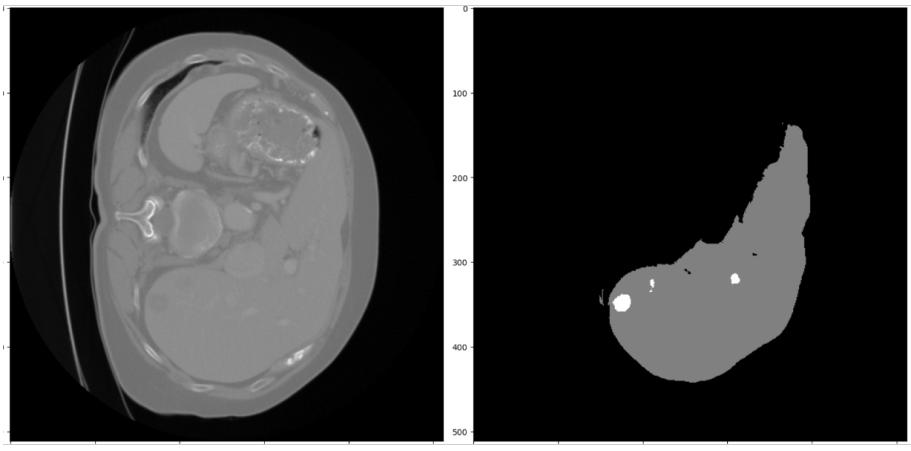

Figure 2: 2D slice of a scan and its matching label file

The label files are only available for the train data, and it is impossible to know whether a prediction for the test set is right or wrong. Therefore, we rearranged the data and splitted the training set into 3 sections – train, validation, and test.

2.1 Data Preprocessing

2.1.1 Images

Data preprocessing involves transforming raw data into a format that is suitable for model training. The process aims to clean, normalize, and enhance the quality of the data to improve the accuracy and effectiveness of modeling.

As part of data preprocessing for medical imaging analysis, We needed to convert a NIfTI file to an image format. Creating 2D images from the NIfTI files massively increases the volume of the data and provides multiple examples. In this instance, we transform each 2D slice from the NIfTI into a unique image in JPG format. Later, we normalize each 2D image with min-max normalization to rescale the pixel intensity values of an image to a specific range, typically between 0 and 1. We chose to normalize each image according to its own pixels and not the entire data.

In the case of min-max normalization, the minimum and maximum pixel intensity values in the image are identified, and the entire range of pixel values is linearly scaled to span the desired range (0 to 1). In medical imaging, lighting conditions can impact pixel intensities. Normalizing the pixel intensities to a consistent range can help mitigate these variations, and improve the convergence of ML models during training.

2.1.2 Labels

As we were requested to use the YOLO algorithm, we needed to convert the current label files into the format that the network expects to receive. As a segmentation algorithm, YOLO requires input in the shape of pair of an image and an annotation text file. Annotations refer to the labeled data that specifies the objects' locations and classes in an image. Annotations are provided as bounding box coordinates and class labels for the objects present in the image. Each bounding box is defined by the coordinates (x, y, width, and height), where (x, y) denotes the center of the box, and width and height represent its dimensions. The class label indicates the category or type of the object within the bounding box, in our case it can be either a liver or a tumor. We generate such a file for each slice, namely a 2D image. To do so, we iterated over the slices of a scan and calculated the connected components in each slice using *measure.label* function by *skimage* library. That function gets a 2D array and returns another 2D array, where each connected component is represented by a different value, so we can distinguish them while documenting them in the annotation file. The function also returns the number of connected components found. We then iterated over the found components, got the original label of the component from the original slice, and calculated the center point, width, and height of each component. We documented the label and location in the annotation file for all non-background components. An example of generated file can be seen in Figure 3.

```
1.0 0.4970703125 0.2734375 0.021484375 0.04296875
1.0 0.5146484375 0.677734375 0.388671875 0.32421875
2.0 0.5537109375 0.52734375 0.001953125 0.01171875
2.0 0.5498046875 0.626953125 0.087890625 0.1015625
2.0 0.3837890625 0.65234375 0.009765625 0.01171875
2.0 0.68359375 0.6875 0.046875 0.04296875
```

Figure 3: An example of a generated label file for a specific 3D scan slice

2.1.3 Data Division

As mentioned, the original test data does not contain the label files so we had no way to estimate our prediction for the supplied test set. To overcome that challenge, and due to the fact that we can increase the volume of the supplied training set, we decided to split it into 3 segments – training, validation, and testing.

The original training set included 131 NIfTI 3D scans, which was later increased by converting them into 2D images. We splitted the original training set into 3 new sets of train, validation and test. First, we randomly splitted the data to 80% for the train set (104 scans) and 20% for the test set (27 scans). Then, we used 15% of the train set for a validation set (16 scans), to be used for measuring the training process. To summarize, we divided the scan indices into 3 segments of the above-mentioned sizes, converted the relevant 3D scan (according to their index) into 2D images, and saved them directly into dedicated folders.

Using the same method, we converted the label files into the needed text format. Figure 4 presets the final amount of 2D images in each set of the data, at the end of the preprocessing. Figure 5 presents the distribution of image types in each dataset-images with only liver, only tumor(s), liver with tumor(s), or no liver or tumor at all.

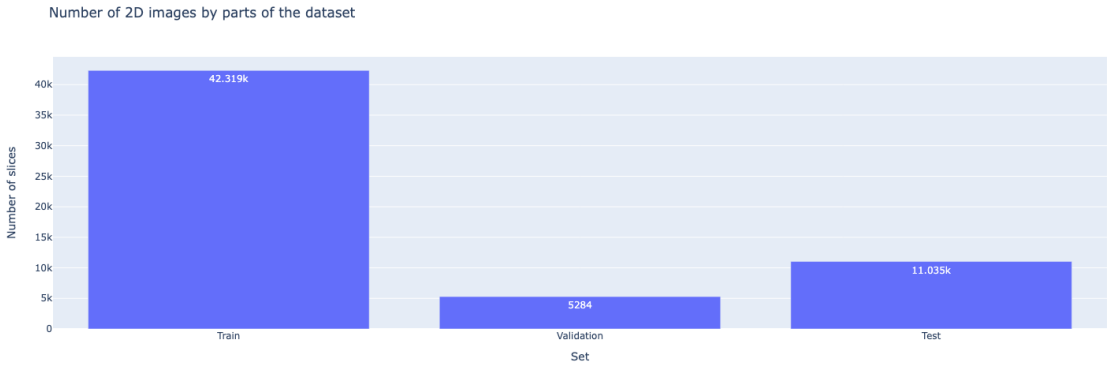

Figure 4: Total amount of images per dataset

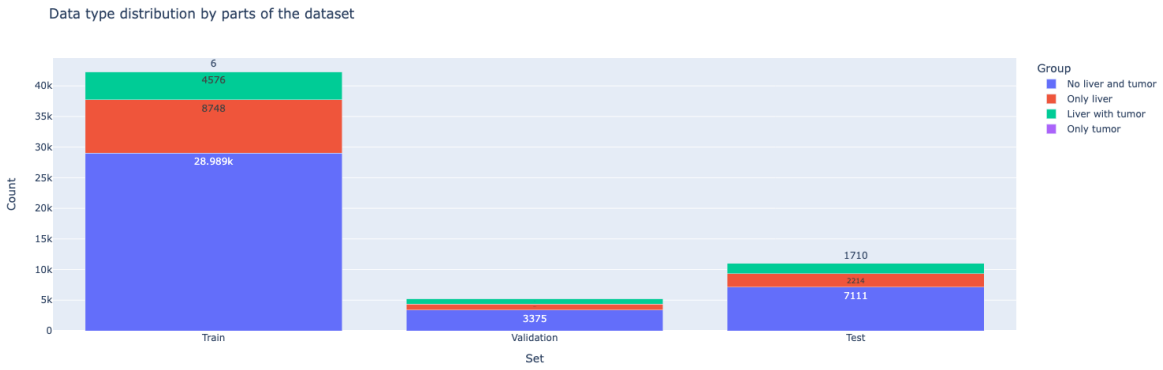

Figure 5: Image type distribution per dataset

3 Solution Design

3.1 YOLOv7 Model

To implement the learning model, we were instructed to use the YOLO algorithm. YOLO is a popular object detection algorithm that is used in computer vision tasks. YOLO models are designed to detect and localize objects in images or video frames by dividing the input image into a grid and predicting bounding boxes and class probabilities for objects within each grid cell.

The main idea behind the YOLO algorithm is that the input image is divided into a fixed-size grid, and the cell predicts a fixed number of bounding boxes. A bounding box is defined by five values – two coordinates of the box's center relative to the cell, two values of width and height of the box relative to the entire image, and a confidence score representing the probability of the box containing an object. For each bounding box, the grid cell predicts class probabilities for different object categories. The process ends with eliminating redundant and overlapping bounding boxes, keeping only the ones with the highest confidence scores.

We chose a late version of YOLO, YOLOv7 for it is known for its effectiveness and precision in tasks involving object detection. It is ideal for medical imaging applications since it uses a single deep neural network to directly predict bounding boxes and class probabilities from an input image. The main advantage of the YOLOv7 model is its quick image processing, an important skill considering the fact that we had to process a high number of images.

3.2 Experimental Settings

As mentioned in 1.2, we used a publicly available YOLOv7 model developed by Wong Kin-Yiu. The model was already trained on different sets of medical images, so we started the training process with its pre-trained weights, therefore the model performed transfer learning.

The model is implemented in Python but its function can be run as scripts using CLI, receiving various parameters to control the model functionality. The model expects to a *.yml* file that contains the paths for the train, validation and test data, and also the number and names of the classes. After each epoch, the training process saves the last weights to a file system, a feature that enabled us to resume the training process once it is crushed without starting from scratch.

The model was trained using mini-batches of 16 slices, and a decaying learning rate with Adam optimizer. We used to default loss function of YOLOv7, *BCEBlurWithLogitsLoss*, which combines the use of smooth binary cross-entropy (BCE) loss and focal loss. We initially intended to train the model for 55 epochs, however, after 18 epochs we noticed that the performance is not improving anymore and are satisfying.

3.3 Experimental Results

After training the model for 18 epochs, we evaluated it and the results were similar to those that are presented in the Git repository of the YOLOv7. The model achieved a mAP@.5 of 51.7% for livers, 50.4% for tumors, and 51.1% for all images. Moreover,

the predictor reached a precision rate of 88.8% for livers, 70.2% for tumors, and 79.5% for all data. Regarding the recall, the predictor achieved 50.4% for livers, 46.4% for tumors, and 48.4% for all data. The predicting process generates images with marked bounding boxes and saves them into the filesystem. In Figure 6 we can see few examples of successful predictions of livers with tumors, done by the model.

Figure 6: Examples of successful predictions of livers and tumors

References

[1] Antonelli, M., Reinke, A., Bakas, S., Farahani, K., Kopp-Schneider, A., Landman, B. A., Litjens, G., Menze, B., Ronneberger, O., Summers, R. M., et al. The medical segmentation decathlon. *Nature communications 13*, 1 (2022), 4128.

[2] Simpson, A. L., Antonelli, M., Bakas, S., Bilello, M., Farahani, K., Van Ginneken, B., Kopp-Schneider, A., Landman, B. A., Litjens, G., Menze, B., et al. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv preprint arXiv:1902.09063* (2019).