Exercise 2: Detection of liver tumors

Gal Almog
Agathe Benichou

June 20, 2023

1 Background

In this assignment, we explored detection of liver tumors in medical images. The dataset we worked with consists of contrast-enhanced CT images from patients with primary cancers and metastatic liver disease, as a result of colorectal, breast and lung primary cancers. The corresponding target regions of interest (RoIs) were the segmentation of the liver and tumors inside the liver. We were tasked with using the RetinaNet model to perform object detection on the liver tumors in the dataset. In order to work with this data, we had to convert the segmentation mask labels into corresponding bounding boxes, and use them to compare to our predicted bounding boxes in order to evaluate our performance.

2 Data Exploration

The dataset, called Liver Tumor Segmentation (LiTS) Benchmark, is a dataset created as part of a project that aims to evaluate and compare automatic methods for liver and liver tumor segmentation from CT scans. The dataset has significant label unbalance between larger (liver) and small (tumor) target ROIs. The scans have been annotated by radiologists with the liver and liver tumors, where the annotations are provided as binary masks with three label values: 0 for background, 1 for liver and 2 for tumors.

The first goal of the assignment was to perform data exploration in order to evaluate and visualize the data. Medical imaging techniques capture the 3D structure of the image by generating volumetric data. These images can be visualized as a series of 2D slices along the axial, sagittal and coronal planes:

We also generated animations of the CT scans in the dataset by viewing each slice in a scan sequentially. This view allows us to really examine the contents of the scan and identify objects within it. The animations are available in our code notebook.

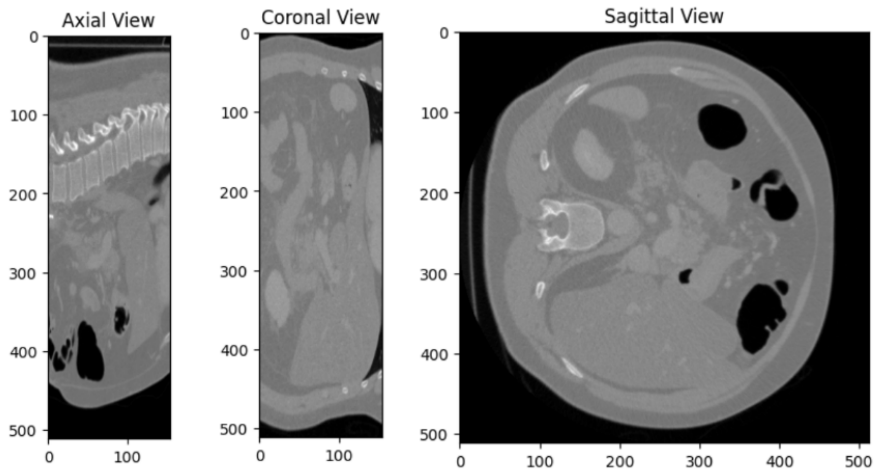

Figure 1: 2D slices of sample subject in dataset along the axial, sagittal and coronal planes.

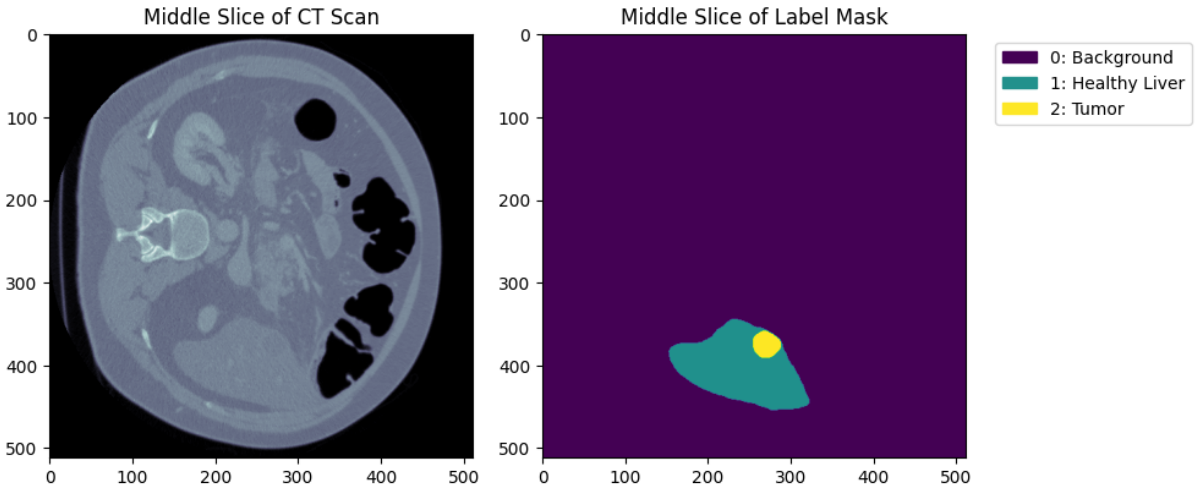

Figure 2: Inspecting a sample slice for the scan and the mask at given slice index

3 Data Preprocessing and Augmentation

The next goal of this assignment was to perform data preprocessing and data augmentations. The LiTS paper mentions that HU-value clipping, normalization and standardization were the most frequent techniques for data preprocessing, and that data augmentation was widely used and mainly focused on standard geometric transformation such as flipping, shifting, scaling or rotation. We followed in this direction to augmented the training data to increase its diversity and variability, allowing the model to learn more robust and generalizable patterns. We oriented the images to the standard orientation, ensured that all images are of the same size by cropping larger images and padding smaller ones, rescaled the pixel intensity values to better condition the optimization problem, and use an affine transformation to ensure our model will be robust to variations in the data.

To split the dataset into training and testing sets, we took the total number of 131 subjects and used 105 for training and 13 for each of the validation and test datasets.

4 Object Detection: RetinaNet

To perform object detection, we use RetinaNet to find and identify objects in an image. While traditional object detection methods involve a two step process of proposing RoIs (where objects might be found) and classifying those regions, RetinaNet was designed to eliminate the need for RoI proposal (thus making the detection process more efficient). RetinaNet is known for its ability to detect objects across a wide range of scales and it deals with class imbalance. In its backbone, it is a deep CNN (convolutional neural network) whose role is to extract features from the input images. It contains a Feature Pyramid Network (FPN) that takes the feature maps produced by the backbone and creates a pyramid of enhanced feature maps. Each level of the pyramid corresponds to a different scale, which allows for the detection of objects of various sizes. Each level of the FPN is linked to two small subnetworks, one for classifying objects and one for regressing the bounding boxes. RetinaNet uses a focal loss that is designed to address the problem of class imbalance in object detection by reducing the weight of easy to classify examples and increasing the weight of hard to classify examples.

In our case, we train RetinaNet to detect tumors in these 2D slices of the CT Scan. Each tumor can be considered an object to be detected (with the corresponding bounding box and class label). The ground truth labels can be obtained from the segmentation masks. In order to do so, we needed a function that takes a batch of 3D masks as input and returns a list of 2D bounding boxes and the corresponding labels for each one. We created such a function (called mask_to_bbox) that returns a list of dictionaries, where each dictionary represents a 2D slice and contains two keys: one for the bounding boxes and one for the labels. Each bounding box corresponds to an object and its label indicates the class of the object.

For the loss functions, we use the default RetinaNet Focal Loss function, which combines classification loss (to predict the correct class of the object in the bounding box) and regression loss (to predict the coordinates of the bounding box). For object detection, the combination of classification and regression losses is combined. We defined an IoU threshold of 0.5 to consider a positive detection, meaning we consider a detected tumor correct if it overlaps at least 50% with the ground truth bounding box. We defined a stochastic gradient descent optimizer with a learning rate of 0.01 and momentum of 0.9. This optimizer was be used to update the model's weights during training.

The learning rate controls the step size at each iteration while moving toward a minimum of a loss function, we chose 0.01 as a fair starting point such that it is not too small (small updates to the parameters) or too large (large updates to the parameters). The momentum helps the navigate along the relevant directions and soften the oscillation. We choose 0.9 as a common starting point, before experimenting with the value.

The training loop in the code trains the model to improve its performance by minimizing the defined loss function. We iterate over a number of ten epochs (where each epoch is one complete pass through the entire training dataset). In our training, we iterate over batches of training data and perform the following:

- Fetch the input images and convert them from segmentation masks to bounding boxes using our defined mask_to_bbox() function
- Make the forward pass with the input to the model

- Compute the loss using the outputs from the forward pass
- Calculates the gradient of the loss with respect to the models parameters through the backward pass
- Optimize the models parameters
- Accumulate the total loss by summing over the loss for each batch

After all batches have been processed, the total training loss is averaged by the number of samples in the dataset to calculate the average training loss for that epoch. The validation phase is attained once the model has been trained on all batches for an epoch. It calculates the average validation loss. This entire process is repeated for each epoch so that the model learns the distribution of our training data.

5 Evaluation

Individual lesions are defined as 3D connected components within an image. A lesion is considered detected if the predicted lesion has sufficient overlap with its corresponding reference lesion, measured as the intersection over the union of their respective segmentation masks. It allows for a count of true positive, false positive, and false-negative detection, from which we compute the precision and recall of lesion detection. We evaluated the results using precision, which is the number of true positions over the number of true positives plus the number of false positives. Precision evaluates the accuracy of our detection system by calculating the proportion of detected bounding boxes that are relevant to the total number of boxes that are detected. In our case, true positives are the correctly identified objects and the false positives are the incorrectly identified objects.

6 Conclusion

7 Note on 3D, 2.5D

Due to the 3D nature of CT scans, this assignment required us to decide how to process the data and train our model. In our first attempt, we wanted to keep the data in 3D to take full advantage of the increased information. Training an object detection model for medical CT scans on 3D data offers several advantages and some challenges compared to using 2D data. One significant advantage is the ability to capture volumetric information, allowing the model to analyze the relationships and spatial context between objects in different slices of the scan. This can improve the accuracy of object localization and enable the detection of complex structures that span multiple slices. Furthermore, 3D models can provide a more comprehensive understanding of anatomical structures, aiding in accurate diagnosis and treatment planning. However, training on 3D data also introduces challenges. One major concern is the increased computational complexity and memory requirements due to the larger input volume. For our 3D model, we had to process the data in volumetric patches in order to reduce the GPU loads for training. The benefit of this, on the other hand, was that in order to help our model handle the class imbalance better, we used the LabelSampler from TorchIO that samples patches in

each batch with different probabilities according to the labels. We defined it so that patches were samples with p = 0.2 for background, p = 0.3 for the liver and p = 0.5 for the liver tumors. This means that patches are more likely to be sampled from regions with label values = 2 (tumor). Unfortunately, we were not able to get our 3D model to run so were forced to adapt our model to 2D data.

Another approach we tried to perhaps mitigate some of the challenges of 3D but still allow us to use the volumetric information is 2.5D. In 2.5D, the 3D data is processed by considering 2D slices or patches extracted from the volumetric scans. While the slices are processed entirely in 2D, the model also considers one slice before and one slice after the given slice, this way, it is still able to use some of the context of the slices to make more informed object detection decisions. This approach combines the advantages of 3D analysis with the computational efficiency of 2D methods. By processing 2D slices or patches individually, it becomes feasible to use existing 2D object detection models and benefit from pre-trained models, transfer learning, and available annotated 2D datasets. Because CT scan data is in greyscale, we were able to train with 3 slices (1 slice, 2 context slices), and still use the pre-trained RetinaNet model as it expects a 3-channel (RGB) 2D image. However, it is important to note that the 2.5D approach may lose some of the spatial information present in the full 3D data, which can impact the accuracy of detection and localization for objects that span multiple slices.

Overall, we were not successful in implementing either the 3D or the 2.5D methods for RetinaNet on this dataset. We provide our code for these models as well for reference.