Lung Segmentation using Deep Learning

Article · December 2023

CITATIONS
0

READS
15

1 author:

Shilpa Kuppili
Yeshiva University
3 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Lung Segmentation using Deep Learning

Shilpa Kuppili
Yeshiva University
skuppili@mail.yu.edu

Abstract

Lung cancer has become a serious illness that poses a threat to people's lives and health. Accurate tumor localization requires precise segmentation of lung regions, which can yield valuable information for lung image analysis. First, we present a lung image segmentation model in this work.In order to recover the segmentation map, the suggested architecture can expand the feature map and retrieve high-level information. We suggest adding a post-processing layer to eliminate the segmentation map's unnecessary parts in order to further enhance the segmentation results.

1. Introduction

One of the malignant tumors that is most dangerous to people's health and lives and has the quickest rate of increase in morbidity and death is lung cancer. Numerous nations have reported a notable rise in lung cancer incidence and mortality during the last five decades. In the United States, lung-related diseases continue to rank among the top ten causes of death. Lung cancer prevention is crucial as a result. According to the definition, lung cancer is defined as a malignant lung tumor with uncontrollably growing lung tissue. Physicians discover that early-stage lung cancer has a 90 percent cure rate. As a result, early detection of lung cancer is crucial for disease control. A chest x-ray, or CXR, is frequently used as a lung cancer diagnostic imaging procedure with a diagnosis of pneumonia. But there are a variety of variables, including the patient's location and the depth of in1This work was composed prior to our earlier publication.Perspiration can alter the CXR's appearance, which further complicates interpretation. Furthermore, physicians deal with every shift, reading copious amounts of photographs. It is challenging enabling the medical professional to get a precise diagnosis without the assistance from a different tool. Development of segmentation technologies is therefore required to increase the efficacy of the therapy. Nonetheless, creating a successful lung segmentation technique is a difficult issue because the ROIs are frequently unclear.One of the most popular and basic segmentation techniques for lung segmentation is the threshold method.This method, known as area segmentation, splits the gray value into two or more grayscale regions, selecting one or more of the suitable benchmarks to determine whether the area satisfies the threshold necessary based on the variation between the target from the backdrop, and distinguishes between the goal is generate a binary picture. Global threshold and adaptive threshold are the two types of threshold processing.Whereas the adaptive threshold establishes several thresholds, the global threshold simply sets one. Setting the threshold allows for the segmentation of the target and background zones.

2. Related Work

In the segmentation process, deep neural networks have also been used, and they have supplanted numerous conventional methods. Garcia attempted to provide an overview of deep learning-based segmentation in a review of deep learning techniques that was published. To solve the segmentation task, there exist multiple models.One of the first methods for solving the end-to-end segmentation problem in deep learning-based image processing is the fully convolutional network. The dense prediction uses a convolution network, which eliminates the need to go through the entire connection layer.Using this technique, it becomes possible to segment photos of.It works well with any size and is a lot quicker than the patch classification approach. Nearly every other more sophisticated This architecture is followed by methods. Nevertheless, the FCN model has a number of drawbacks, including the fact that its intrinsic spatial invariance means that it is unable to account for valuable global context data, and its performance suffers in high-resolution scenarios and is not real-time division.

3. Methods

A custom dataset class called SegmentationDataset is defined. This class is designed to be used with PyTorch's torch.utils.data.Dataset module, which provides an interface

for custom datasets in PyTorch.The root directory where the dataset is located. This directory is expected to contain subdirectories for images and masks.transform: An optional argument representing a data transformation to be applied to the images and masks. This is typically a composition of image processing operations defined using the torchvision.transforms module.transforms.Compose: This is a container for composing multiple image transformations. It allows you to chain several transformations together. transforms.Resize(256): This transformation resizes the input image to have a height and width of 256 pixels. It maintains the aspect ratio of the original image.transforms.ToTensor(): This transformation converts the image to a PyTorch tensor. It also scales the pixel values from the range [0, 255] to [0.0, 1.0].Overall, this class is intended for use in semantic segmentation tasks, where images and corresponding masks are needed for training a model.PyTorch's torchvision.transforms module is used to define a composition of image transformations, and then it creates an instance of the SegmentationDataset class, specifying the root directory for the dataset and the transformation to be applied to each image and mask.We set up a PyTorch data loader for the training dataset, allowing you to load data in batches during the training of a neural network.Further, we define a function that returns a sequential composition of three neural network layers: a convolutional layer (nn.Conv2d), a batch normalization layer (nn.BatchNorm2d), and a rectified linear unit (ReLU) activation layer (nn.ReLU).in channels: The number of input channels for the convolutional layer.out channels: The number of output channels (i.e., the number of filters or kernels) for the convolutional layer.kernel: The size of the convolutional kernel.stride: The stride of the convolution operation. padding: The zero-padding added to both sides of the input.Thelayer performs a 2D convolution operation. It takes as input the number of input channels (in channels), the number of output channels (out channels), the kernel size (kernel), the stride of the convolution (stride), and the padding (padding). The output is the result of applying the convolution operation to the input.Batch normalization is applied to normalize the activations of the convolutional layer. It helps stabilize and accelerate the training process by normalizing the input to each layer.The nn.Sequential container is used to define a sequence of layers that will be applied in order.The forward method processes input through the ResNet50 layers.It then passes the features through the decoder layers and performs upsampling.The decoder features are concatenated and upsampled. The final layers are applied, and the output is upsampled again before being returned.An architecture that follows a U-Net-like structure is defined in semantic segmentation tasks, with a ResNet50 backbone for feature extraction and decoder layers for upsampling and refining features. The final output is an upsampled prediction map.During training, we would typically have a loop where we forward pass the input through the model, compute the loss using the specified criterion (CrossEntropyLoss), backward pass to compute gradients, and then update the model's parameters using the optimizer (SGD in this case).nn.CrossEntropyLoss(): This is a loss function commonly used for multi-class classification problems. It combines the softmax activation and the negative log likelihood loss. In the context of semantic segmentation, where each pixel is assigned to a class, this loss function is suitable for training the model to predict class labels for each pixel.torch.optim.SGD: Stochastic Gradient Descent (SGD) is an optimization algorithm commonly used for training neural networks.model.parameters(): This argument specifies the parameters (weights and biases) that will be updated during the optimization process. It passes all the parameters of the model to the optimizer.weight decay=1e-4: This is a regularization term applied to the weights during optimization. It helps prevent overfitting by penalizing large weights. The value 1e-4 is the strength of the regularization.lr=0.001: Learning rate is a hyperparameter that determines the step size at each iteration while moving toward a minimum of the loss function. 0.001 is a common starting learning rate.momentum=0.9: Momentum is a parameter that accelerates the optimization process by adding a fraction of the previous update to the current update. A momentum value of 0.9 is a typical choice.If a GPU is available, the model will be trained on the GPU to leverage its computational power, and if not, it will fall back to CPU execution.Further, we compute the Intersection over Union (IoU) for binary segmentation masks. IoU is a metric commonly used to evaluate the performance of segmentation models by measuring the overlap between predicted and ground truth masks.Further, iterate over epochs and processes batches of data through the network, calculating the loss, updating the model's parameters, and printing metrics such as loss, accuracy, and IoU at the end of each epoch.Further, we evaluate the model on the test dataset, calculates the average IoU and loss, and stores the predictions and true masks for further analysis or visualization.

4. Results

A trained segmentation model is evaluated on a test dataset.For each batch in the test dataset the model's predictions outputs are computed.The predictions are compared to the ground truth masks to calculate the loss and IoU.The loss and IoU are accumulated for later calculation of averages.The model is evaluated on the test dataset, calculates the average IoU and loss, and stores the predictions and true masks for further analysis or visualization.After iterating through the entire test dataset, the average IoU and average loss are computed by dividing the accumulated values by the number of batches The IoU is calculated as 0.87.

5. Discussion

Our model attains a higher IoU score, which is one of its clear advantages. The segmentation net that has been designed is appropriate for segmenting images, and the segmented results are enhanced by the post-processing layer that eliminates superfluous parts of the image. Our model scores 0.87, and adding more similar images to the training set is an easy way to improve it.

6. Conclusion

We present the first lung segmentation using the segmentation architecture in this paper, and we obtain a 0.87 IoU score accurate segmentation. Utilizing a postprocessing layer, the superfluous portion of the prediction map is eliminated. Additionally, the suggested model can be used for a variety of different medical image segmentation tasks. In the upcoming phase, we aim to create a more resilient encoder and decoder model that can be utilized in various scenarios.

References

Jinsa Kuruvilla and K Gunavathi, "Lung cancer classification using neural networks for ct images," Computer methods and programs in biomedicine, vol. 113, no. 1, pp. 202–209, 2014.

Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," arXiv preprint arXiv:1704.06857, 2017.

Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 12, pp. 2481–2495, 2017.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in International Conference on Medical image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," arXiv preprint arXiv:1412.7062, 2014.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le, "Learning transferable architectures for scalable image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.