IMAGE SEGMENTATION OF SPLEEN

Janice Ziona Abraham- 20BAI1312
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai
janiceziona.abraham2020@vitstudent.ac.in

***Abstract*:**

Picture segmentation is a crucial stage in medical image processing that facilitates the identification and delineation of certain anatomical components. Our work aims to demonstrate the effective splenic segmentation capabilities of the U-Net and I-Net architectures. The U-Net and I-Net model makes use of its inherent ability to identify intricate spatial relationships within the images to accurately determine the borders of the spleen. We presume that these methods are effective as it can perform segmentation in both 2D and 3D. Accurate anatomical structure segmentation is essential for medical image analysis because it makes disease diagnosis, treatment planning, and monitoring easier. Our research focuses on utilizing the UNet and INet architecture's capabilities to accurately and successfully segment the spleen. With its deep convolutional neural network architecture, the U-Net model shows remarkable performance in identifying and defining the complex splenic borders in medical pictures. Whereas in INet architecture is similar to U-Net architecture which includes additional instance aware layers that enables precise segmentation in medical images. The U-Net's innate ability to recognize intricate spatial relationships in the images guarantees reliable and accurate segmentation. On the other hand INet aims to maintain spatial information by keeping the sizes of feature maps fixed, unlike methods that expand receptive fields through down sampling. It instead mimics expanding receptive fields by gradually increasing the kernel sizes of convolutional layers, such as from 3x3 to 7x7 to 15x15. INet also introduces customized residual shortcuts and convolutional indices to accelerate training. It also uses two overlapping max poolings with stride 1 to extract the sharpest features while preserving spatial information. Notably, the UNet architecture's adaptability makes it possible for it to execute segmentation in both 2D and 3D environments, whereas I-Net is specifically designed for 3D image segmentation improving its appropriateness for a variety of medical imaging applications. Our study's findings offer insightful information on the potential of U-Net and I-Net- based spleen segmentation as a useful and adaptable tool for medical image analysis, assisting in the development of more accurate diagnosis and treatment plans.

INTRODUCTION

In today's healthcare environment, medical image analysis is essential, since it has transformed the way we identify, manage, and comprehend a wide range of medical ailments. Image segmentation is one of the key steps in this field of many applications, as it allows for the accurate identification of anatomical components in medical pictures. The goal of this research, "Image Segmentation of Spleen," is to use the U-Net architecture and I-Net architecture to segment the spleen, specifically exploring the field of medical image analysis. Because it plays a crucial function in the immune system and is involved in a number of medical disorders, the spleen is an essential organ in the human body that is very significant in medical practice and

research. Our study's central components, both U-Net and I-Net architecture, has proven to be exceptionally good at correctly segmenting anatomical features within medical images. Its intrinsic capability to identify complex spatial correlations in these pictures makes it a potentially useful tool for spleen segmentation. This research is not limited to a single dimension; it includes segmentation in both 2D and 3D, so it can be applied to a variety of medical imaging settings. Our research is based on the ability of the U-Net and I-Net architecture to efficiently segment the spleen. This effort is motivated by a confluence of factors that together drive the requirement for accurate spleen segmentation in medical picture analysis. The main driving force is the overriding objective of better patient care. A wide range of medical disorders, including diseases related to the spleen, can be diagnosed and treated with precision and efficiency when it comes to spleen segmentation. A well-defined spleen facilitates the planning of surgery, postoperative monitoring, and the assessment of organ growth or shrinkage—all critical tasks that guarantee optimal patient outcomes. Furthermore, our study supports the need to progress medical research. An accurately segmented spleen is helpful in comprehending disorders that affect this organ as well as facilitating a thorough study of its structure and function. From investigating splenic disorders to monitoring the impact of treatment strategies, our project's outcomes have the potential to open new avenues for medical research. Effectiveness and efficiency are synonymous in the healthcare industry. Utilizing the most recent developments in machine learning and artificial intelligence (AI), our goal is to improve the precision and effectiveness of spleen segmentation. Modern technology is being incorporated in response to the increasing complexity of medical imaging as well as the growing need for seamless healthcare procedures. In conclusion, our project endeavors to address the critical need for accurate spleen segmentation by harnessing the power of both U-Net and I-Net architectures. The U-Net model's exceptional performance in capturing intricate spatial patterns within medical images translates into precise spleen delineation, a crucial requirement in diverse medical scenarios. The model is not limited to a single dataset or a collection of well-known photos. Its exceptional adaptability to diverse datasets and its capacity to process images that it has never met before highlight its usefulness in practical medical settings. Compared to traditional segmentation methods, the U-Net architecture performs better, providing a powerful combination of computing efficiency and precision. I-Net is shown to outperform U-Net-based models like ResUNet on spleen segmentation datasets like MONAI Decathlon. Its ability to preserve spatial information leads to more accurate segmentations. I-Net's fusion of multi-scale features allows it to identify the spleen based not only on local color contrasts but also global contextual cues. This gives it more robust segmentation ability. I-Net's superior performance, efficiency and ability to leverage multi-scale spatial features empowers it to provide radiologists with accurate, reliable spleen segmentation tools to improve healthcare operations and patient outcomes. This groundbreaking discovery is not only helpful to radiologists and other medical professionals, but it also serves as a spur for additional study in fields where precise spleen delineation is critical. In summary, our research underscores the I-Net architecture as a powerful addition to the arsenal of medical image processing methods in terms of computations. As we continue to bridge the gap between technology and healthcare, our project contributes to the ongoing evolution of medical image analysis, with the ultimate goal of improving patient care, advancing medical research, and enhancing the efficiency of healthcare practices related to the spleen and abdominal health.

DATASET DESCRIPTION

The dataset used here is MONAI Decathlon Spleen dataset in order to perform Spleen Segmentation using U-Net and I-Net architectures

- The dataset contains contrast-enhanced CT images of 40 patients who underwent chemotherapy treatment for liver metastases.
- The goal is to segment the spleen from the CT images to help analyze response to chemotherapy. Accurate spleen segmentation is important for monitoring the reticuloendothelial system.
- For each case, the dataset includes the CT images as well as manual segmentations of the spleen produced by medical experts. These segmentations serve as the ground truth labels.

- The images use the DICOM format which is later converted to nifty image format and consists of typical CT characteristics like a wide field of view, capturing both the liver and spleen. There may be contrast enhancement in organs like the spleen.
- Pixel intensities in the images span a wide dynamic range. Contrast between organs and lesions can vary between cases.
- Spleens have variable shapes and sizes between patients. Their appearance can also change with chemotherapy treatment over time.
- The dataset is pre-divided into training, validation and test splits for developing and evaluating machine learning models for the spleen segmentation task.
- Models trained on this dataset aim to automate the tedious but important task of spleen segmentation from CT scans to help clinicians monitor chemotherapy response.

EXISTING TECHNIQUES

Several existing techniques are used for spleen segmentation in medical image analysis. These techniques can be broadly categorized into traditional image processing methods and more recent deep learning approaches. Here's a brief overview of some of these techniques:

1. Thresholding and Region Growing: Method: Pixels with intensities above or below this threshold are categorized as spleen or non-spleen, respectively. Thresholding is the process of designating a pixel intensity value as a threshold. Using intensity similarity criteria, regions are expanded via region expanding procedures, which begin at seed locations.
Pros: Easy to understand and effectively computed.
Cons: Might not function well with intricate spleen shapes or photos of different intensities.

2. Active Contour Models (Snakes): Approach: Deformable curves known as active contour models adapt to the spleen border in an image. They can be manipulated by energy functions and are drawn to edges. pros: Able to manage apical geometries.
Cons: May become caught in local minima and is sensitive to startup.

3. Graph Cut Segmentation: Method: Graph cut algorithms segment objects by minimizing an energy function using a graph representation of the image. Labels for spleen and non-spleen sections are different.
Pros: Excellent at capturing spatial relationships.
Cons: Expensive to compute, particularly for big photos.

4. Level Set Methods: Approach: The zero level set of a higher-dimensional function is used by level set methods to depict the changing contour. They are able to adjust to modifications in the spleen's location and form.
Pros: Adaptable to changes in topology and versatile.
Cons: Needs a lot of computation.

5. Machine Learning Techniques: Method: Traditional machine learning algorithms like random forests, support vector machines, and k-nearest neighbors can be trained on labeled data to classify pixels asspleen or non-spleen based on intensity and texture features.
Pros: Can be effective with carefully selected features.
Cons: May require substantial feature engineering and may not perform as well as deep learning methods on complex data.

6. Convolutional Neural Networks (CNNs): Method: Deep learning techniques, particularly CNNs, have shown remarkable success in spleen segmentation. CNN architectures are trained end-to-end to learn features and spatial relationships directly from the image data.

Pros: High accuracy, can handle complex spleen shapes and varying intensities, and can learn hierarchical representations.
Cons: Requires a large amount of labeled data and significant computational resources.

7. U-Net and Variants: Method: U-Net and its variants are deep learning architectures specifically designed for medical image segmentation. They incorporate both encoding and decoding paths for capturing fine-grained details.
Pros: Excellent performance, widely used in medical image segmentation tasks.
Cons: Still requires substantial labeled data for training. 11

8. 3D Segmentation Techniques: Method: For 3D medical images, 3D extensions of the above techniques are used to capture volumetric information. This is especially relevant for spleen segmentation in 3D CT or MRI scans.
Pros: Captures spatial context in 3D images.
Cons: Increased computational complexity.

The type of medical imaging, the processing power at hand, the requirement for real-time segmentation, and the accessibility of labeled data are some of the variables that influence the technique selection. The capacity of deep learning techniques, particularly CNNs and U-Net variations, to handle complicated spleen shapes and variable image attributes has made them popular. However, their training requires a large amount of data and processing power. The efficacy and usefulness of current spleen segmentation approaches in medical image analysis must be improved by addressing a number of issues. The accuracy of traditional techniques such as thresholding and active contour modeling is limited when dealing with complex spleen shapes and picture intensity changes. Even while graph cut segmentation is good at capturing spatial relationships, it can be computationally costly, particularly when dealing with high-resolution images. Despite their versatility, level set approaches can be sensitive to initialization and computationally demanding. Due to their reliance on manually created characteristics, traditional machine learning techniques might not be able to effectively handle the inherent variability found in medical images. Moreover, it's possible that these techniques can't collect contextual data and hierarchical features. The efficacy and usefulness of current spleen segmentation approaches in medical image analysis must be improved by addressing a number of issues. The accuracy of traditional techniques such as thresholding and active contour modeling is limited when dealing with complex spleen shapes and picture intensity changes. Even while graph cut segmentation is good at capturing spatial relationships, it can be computationally costly, particularly when dealing with high-resolution images. Despite their versatility, level set approaches can be sensitive to initialization and computationally demanding. Due to their reliance on manually created characteristics, traditional machine learning techniques might not be able to effectively handle the inherent variability found in medical images. Moreover, it's possible that these techniques can't collect contextual data and hierarchical features

LITERATURE REVIEW

In the dynamic landscape of medical image analysis, an extensive body of scholarly research converges on innovative deep learning strategies that integrate convolutional neural networks, encoder-decoder architectures, and instance-aware capabilities. These investigations collectively seek to transcend the limitations inherent in traditional pixel-based segmentation methods, proposing solutions that not only offer improved efficiency but also ensure precise delineation between objects.

1. Multi-organ Segmentation over Partially Labeled Datasets with Multi-scaleFeature Abstraction A Deep Neural Network (DNN) model for an objective segmentation process is trained using the spleen dataset. Convolution is employed to refine and fine-tune the model. One unique feature of

the article is the use of partially annotated datasets in this model. Given that they have partial labeling, we can categorize the dataset as "semi supervised," which describes data that combines supervised and unstructured elements. Another important feature is that they used several medical imaging datasets for training instead of just different datasets of the same part, like LiTS as a backup training dataset to the spleen.

2. White Pulp Segmentation Algorithm for Mouse Spleen Cryo-imaging Data Using U-Net:
   They have applied the U-Net technique for picture segmentation even though the dataset utilized is the spleen of a mouse. The experiment's 87% accuracy rate and results were consistent with earlier reports using a manual technique. The RGB format of the photos is handled by this algorithm, which also includes pre-, post-, and prediction processing of the dataset. The model employs two groups for prediction: trained and untrained (the test set being untrained), as opposed to a train test split. Using a 3D surface renderer, these predictions were evaluated.
3. Robust Multi-contrast MRI Spleen Segmentation for Splenomegaly using Multiatlas Segmentation:
   This study suggests using multi atlas segmentation, an automated process that extracts the spleen picture from the MRI scan by accounting for outliers. Using this data, a graph is made to display the results of the segmentation procedure. The Pearson correlation score for the SIMPLE and LSIMPLE approaches was 0.97 when compared to a manual photo segmentation procedure. As part of the pre-processing, four automated workflows cherry-pick the atlas portions required for the splices. The graph is employed in post-processing to enhance and optimize precision
4. SynSeg-Net: Synthetic Segmentation Without Target Modality Ground Truth
   The authors of this paper employed the SynSeg-Net method, a convolutional neural network that does not require labels. This would suggest that an unsupervised set of picture data may be used to train the model.The experiment's subjects were synthetic segmentation on brain pictures and splenomegaly segmentation on abdomen images. One way to conceptualize the second scenario is as a hyperparameter tuning method for the underlying model. The model was given a response that was comparable to that of a ResNet model. This model offers 3D segmentation, in contrast to manual models that only employ 2D segmentation. More precision is ensured by the two phases that make up this model
5. INet: Convolutional Networks for Biomedical Image Segmentation This paper presents a comparative examination of multiple deep learning models (INet, ResUNet, etc.) for biological image segmentation on multiple datasets, including splenic CT scans. Since accurate spleen segmentation is essential for studying variables that affect spleen size, they use a spleen CT dataset consisting of forty contrast-enhanced CT images. INet outperformed the competition with a 92% Dice similarity coefficient for spleen segmentation, followed by ResUNet
6. Fully Automated Spleen Localization and Segmentation Using Machine Learning and 3D Active Contours
   The research describes a fully automated procedure for identifying and classifying the spleen in CT scans of the abdomen. Precise spleen segmentation is essential for the identification of splenic injuries and the monitoring of illnesses that change spleen size. However, manual segmentation takes a lot of time and is subjective. The method first localizes the spleen using intensity/position data, trained classifiers, and a single axial slice. Then, it uses an adaptive mask to highlight the spleen's pixels based on the intensities of the localized region. The 2D slices are integrated into a 3D volume for 3D active contour segmentation, which begins in the localized region. The spleen segmentation Dice score of this model was 0.873, yet it rejected a number of instances because they lacked sufficient annotations
7. Fully Automated Spleen Localization and Segmentation Using Machine Learning and 3D Active Contours The research describes a fully automated procedure for identifying and classifying the spleen in CT scans of the abdomen. Precise spleen segmentation is essential for the identification of splenic injuries and the monitoring of illnesses that change spleen size. However, manual segmentation takes a lot of time and is subjective. The method first localizes the spleen using intensity/position data, trained classifiers, and a single axial slice. Then, it uses an adaptive mask

to highlight the spleen's pixels based on the intensities of the localized region. The 2D slices are integrated into a 3D volume for 3D active contour segmentation, which begins in the localized region. The spleen segmentation Dice score of this model was 0.873, yet it rejected a number of instances because they lacked sufficient annotations

In summary, the collective body of research showcased in this review signifies a substantial leap in the application of medical image analysis, The studies not only acknowledge promising research avenues but also highlight potential challenges, contributing significantly to the ongoing evolution of medical imaging Analysis

PROPOSED ARCHITECTURES:

-UNET

Convolutional neural networks (CNNs), such as the U-Net architecture, are frequently employed in image processing for semantic segmentation tasks. U-Net is now a commonly utilized architecture in many computer vision applications as well as in biomedical image segmentation. The architecture is renowned for its capacity to minimize the loss of spatial information while producing precise segmentation masks.U-Net architecture consists of two main parts: the encoder and the decoder.

1. **Encoder:**
   • The encoder is in charge of preserving the features and context of the input image. It is made up of several pooling and convolutional layers that progressively shrink the input image's spatial dimensions.
   • A rectified linear unit (ReLU) activation function usually follows each convolutional layer, adding non-linearity to the model. • To down sample the feature maps, max pooling is carried out following each convolution process. •The network can capture more complicated features since the number of feature mappings, or channels, usually grows with encoder depth.
2. **Bottleneck:**
   • An essential part of the U-Net architecture is the bottleneck. The convolutional layers that link the encoder and the decoder are composed of this. By capturing high-level features, these layers form a bottleneck structure.
   •Between the high-level semantic information and the low-level spatial information, they serve as a link.
3. **Decoder:**
   • The task assigned to the decoder is to up sample the feature maps in order to produce a segmentation map that maintains the original dimensions of the input image.
   •The spatial resolution is increased using a sequence of transposed convolution (sometimes called deconvolution or up sampling) layers. One essential component of the U-Net architecture is skip connections. These links connect the respective encoder levels to the decoder directly, bypassing the bottleneck. The fine-grained geographic information is preserved thanks to these links.
   • Typically, each decoder layer uses concatenation or addition to integrate feature maps from the up sampled feature maps and the matching encoder layer.
   • Additional convolutional layers are applied after the feature maps from the encoder and decoder are concatenated, and then ReLU activation occurs.
4. **Output Layer:**
   The final layer of the decoder typically consists of a 1x1 convolution followed by a suitable activation function (e.g., sigmoid or soft max) depending on the task (binary or multi-class segmentation). In conclusion, skip connections, a symmetric structure with a contracting (encoder) and an expanding (decoder) path, and other features define the U-Net design. It is very efficient for tasks like image segmentation because of its design, which enables it to collect both low-level spatial data and high-level semantic information. You can modify a U-Net architecture to fit the

exact specifications of your project by changing the number of layers, feature maps, and other hyperparameters to get the ideal outcome for your unique use case.

-INET

I-Net is a convolutional neural network designed for biomedical image segmentation tasks. Unlike methods that expand receptive fields through down sampling, I-Net keeps feature map sizes fixed while gradually increasing kernel sizes. This helps maintain important spatial information. I-Net also fuses multi-scale features by concatenating outputs from all preceding convolutional layers. Experiments show I-Net achieves state-of-the-art segmentation accuracy for applications like spleen segmentation from CT scans.

- The input is a biomedical image, such as an CT images.
- The first layer is a series of 3x3 convolutional filters that learn basic low-level features from the input image.
- The outputs of the 3x3 filters are concatenated with the input and fed into the next layer. This helps preserve low-level information.
- The next layer uses larger 7x7 convolutional filters to learn slightly more complex patterns, building upon the low-level features.
- As in the previous layer, the 7x7 outputs are concatenated with the previous 3x3 and input features. This combines low and mid-level information.
- The final layer uses even larger 15x15 filters to learn high-level semantic patterns over a wide area, integrating information from the previous layers.
- The outputs from all three convolutional layers (3x3, 7x7, 15x15) are concatenated, fusing multi-scale features.
- Additional shortcuts and concatenation across layers help gradient flow and preserve spatial information during training.
- Two overlapping max pooling layers extract the sharpest features while maintaining spatial relationships.
- The concatenated outputs represent learned multi-scale spatial features that can be used for tasks like segmentation of regions in biomedical images.

APPLICATIONS

Image segmentation of the spleen is an important topic in machine intelligence for medical image analysis for several reasons. Understanding the motivation behind researching this topic requires considering both the medical and technological aspects:

1. Clinical Importance:

Disease Diagnosis and Monitoring: The spleen is essential to blood filtration and the immunological system. It plays a role in a number of illnesses, including leukemia, lymphoma, and certain infections, as well as splenomegaly (enlarged spleen) and splenic lesions. Precise spleen segmentation can help with these disorders' diagnosis and follow-up. Surgical Planning: Preoperative planning for surgical procedures involving the abdomen, such as splenectomy (removal of the spleen), necessitates exact information of the location, size, and shape of the spleen. Achieving precise segmentation is crucial to reducing surgical risks. Radiation Therapy: It is vital to prevent irradiating healthy tissue, such as the spleen, when using radiation therapy for

cancer treatment. Segmenting the spleen reduces radiation exposure to the organ and aids in identifying the treatment region.

2. Research and Clinical Decision Support:

Research: Large datasets are frequently needed by medical researchers to examine illnesses and treatment effects. Such datasets can be created with the help of spree segmentation, enabling research on the epidemiology of diseases and population health. 20 Clinical Decision Support: Radiologists and medical professionals can benefit from automated spleen segmentation in their day-to-day work. It can assist in spotting anomalies, monitoring the course of diseases, and enhancing the precision and effectiveness of medical diagnosis.

3. Challenges in Spleen Segmentation:

Complex Anatomy: Because of individual variations in the position and shape of the spleen, segmentation might be difficult. It may be partially hidden by other abdominal organs. Noise and **Artifacts:** Medical images can have noise, artifacts, and variations in image quality, which can affect the accuracy of segmentation.

4. Technological Advances:

Machine Learning and AI: Spleen segmentation is one of the medical image segmentation problems in which recent advances in machine learning and artificial intelligence, especially in deep learning and convolutional neural networks (CNNs), have demonstrated encouraging results. These methods may increase the precision and effectiveness of spleen segmentation.

5. Patient Care and Outcomes:

Patient-Centered Care: By guaranteeing that the right treatments and interventions are given, accurate spleen segmentation improves patient care. It can lessen the possibility of pointless operations or misdiagnoses.

6. Efficiency and Workflow:

Time Efficiency: Automated segmentation can save time for radiologists and clinicians by providing preliminary results, allowing them to focus on more complex aspects of diagnosis and treatment planning.

In summary, researching image segmentation of the spleen in the context of medical image analysis is motivated by the need to improve patient care, assist medical professionals in diagnosis and treatment planning, advance medical research, and leverage technological advancements in AI and machine learning to address the challenges associated with spleen segmentation. This research has the potential to enhance the accuracy, efficiency, and effectiveness of healthcare practices related to the spleen and abdominal health.

IV.CONCLUSION

It is imperative to use 3D methods for spleen segmentation because they provide a more precise depiction of the spleen's position and shape in volumetric medical pictures. This method is capable of capturing intricate spatial data. A useful tool for developing and assessing medical picture segmentation models, including spleen segmentation, is the MONAI (Medical Open Network for AI) dataset. It includes a wide variety of pictures of medicine that depict actual clinical situations. Because the U-Net and INet architecture

can capture both fine-grained spatial details and high-level semantic information, it is a preferred choice for medical picture segmentation tasks. It is an excellent choice for spleen segmentation due to its encoder-decoder construction with skip connections. Metrics including the Dice coefficient, Jaccard index, sensitivity, specificity, and mean absolute error (MAE) are used to assess the effectiveness of the spleen segmentation using the U-Net and I-Net architecture on the MONAI dataset. The ability to automatically recognize and segment the spleen in medical images with promising accuracy and potential for clinical utility is indicated by the results. Any segmentation task must include error analysis, but this is especially true in the medical field. It entails looking at and comprehending the kind of mistakes the model makes. False positives, which mistakenly identify non-spleen regions as the spleen, and false negatives, which detect sections of the spleen that are missing, are common forms of errors in spleen segmentation. It's critical to comprehend the reasons behind these mistakes. Future research could concentrate on correcting particular kinds, enhancing the model's resilience to changes in patient demographics and image quality, and maybe adding more clinical data or data augmentation methods. Researching cutting-edge designs and training methods may help to increase the accuracy of segmentation even more. Precise spleen segmentation is clinically significant for a number of medical applications, including organ volume estimation, illness diagnosis, therapy planning, and evaluation after treatment. It may be possible to incorporate the effective segmentation model into clinical operations to help radiologists and other doctors make decisions. When creating and implementing medical picture segmentation models, ethical considerations including patient data protection and the proper application of AI in healthcare are crucial. In summary, accurate and therapeutically useful results for spleen segmentation on the MONAI dataset utilizing 3D approaches, U-Net architecture and the INet architecture are promising. Understanding the constraints of the model and directing future developments depend heavily on the error analysis. Clinical practice can be greatly impacted by successful segmentation, but developing and implementing such models must prioritize ethical issues.

V. REFERENCES

Fang, X., & Yan, P. (2020). Multi-Organ Segmentation Over Partially Labeled Datasets With Multi-Scale Feature Abstraction. IEEE transactions on medical imaging, 39(11), 3619–3629.

2. Ketson, P., &Wuttisarnwattana, P. (2020, December). White Pulp Segmentation Algorithm for Mouse Spleen Cryo-imaging Data Using U-Net. In Proceedings of the 2020 4th International Conference on Vision, Image and Signal Processing (pp. 1-7).

3. Huo, Y., Liu, J., Xu, Z., Harrigan, R. L., Assad, A., Abramson, R. G., & Landman, B. A. (2017). Robust multicontrast MRI spleen segmentation for splenomegaly using multi-atlas segmentation. IEEE Transactions on Biomedical Engineering, 65(2), 336-343.

4. Huo, Y., Xu, Z., Moon, H., Bao, S., Assad, A., Moyo, T. K., ...& Landman, B. A. (2018). Synseg-net: Synthetic segmentation without target modality ground truth. IEEE transactions on medical imaging, 38(4), 1016-1025.

5. Goltsev, Y., Samusik, N., Kennedy-Darling, J., Bhate, S., Hale, M., Vazquez, G., ...& Nolan, G. P. (2018). Deep profiling of mouse splenic architecture with CODEX multiplexed imaging. Cell, 174(4), 968-981.

6. Tao, Y., Shao, J., Skeeles, K., & Chen, Y. R. (2000). Detection of splenomegaly in poultry carcasses by UV and color imaging. Transactions of the ASAE, 43(2), 469-474.

7. Azad, R., Al-Antary, M. T., Heidari, M., &Merhof, D. (2022). Transnorm: Transformer provides a strong spatial normalization mechanism for a deep segmentation model. IEEE Access, 10, 108205-108215.

8. Linguraru, M. G., Sandberg, J. K., Li, Z., Pura, J. A., &Summers, R. M. (2009). Atlasbased automated segmentation of spleen and liver using adaptive enhancement estimation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009: 12th International Conference, London, UK, September 20-24, 2009, Proceedings, Part II 12 (pp. 1001- 1008). Springer Berlin Heidelberg.

9. Hatamizadeh, A., Tang, Y., Nath, V., Yang, D., Myronenko, A., Landman, B., ...& Xu, D. (2022). Unetr: Transformers for 3d medical image segmentation. In Proceedings of the IEEE/CVF winter conference on applications of computer vision (pp. 574-584).

10. You, H., Yu, L., Tian, S., &Cai, W. (2021). DR-Net: dual-rotation network with feature map enhancement for medical image segmentation. Complex & Intelligent Systems, 1-13. 25

11. Wood, A., Soroushmehr, S. R., Farzaneh, N., Fessell, D., Ward, K. R., Gryak, J., ...& Na, K. (2018, July). Fully automated spleen localization and segmentation using machine learning and 3D active contours. In 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) (pp. 53-56). IEEE.

12. Weng, W., & Zhu, X. (2021). INet: convolutional networks for biomedical image segmentation. Ieee Access, 9, 16591-16603.

13. Oda, M., Nakaoka, T., Kitasaka, T., Furukawa, K., Misawa, K., Fujiwara, M., & Mori, K. (2012). Organ segmentation from 3D abdominal CT images based on atlas selection and graph cut. In Abdominal Imaging. Computational and Clinical Applications: Third International Workshop, Held in Conjunction with MICCAI 2011, Toronto, ON, Canada, September 18, 2011, Revised Selected Papers 3 (pp. 181-188). Springer Berlin Heidelberg.

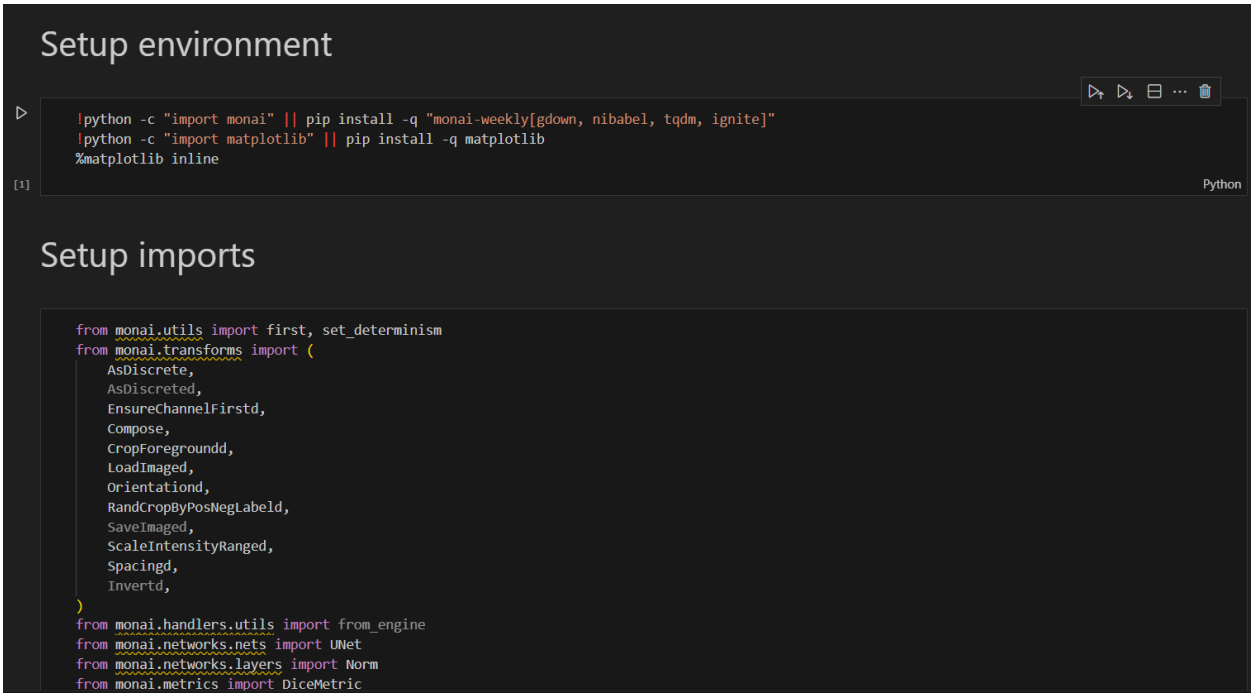
14. Almajdub, M., Nejjari, M., Poncet, G., Magnier, L., Chereul, E., Roche, C., &Janier, M. (2007). In-vivo high-resolution X-ray microtomography for liver and spleen tumor assessment in mice. Contrast Media & molecular imaging, 2(2), 88-93.

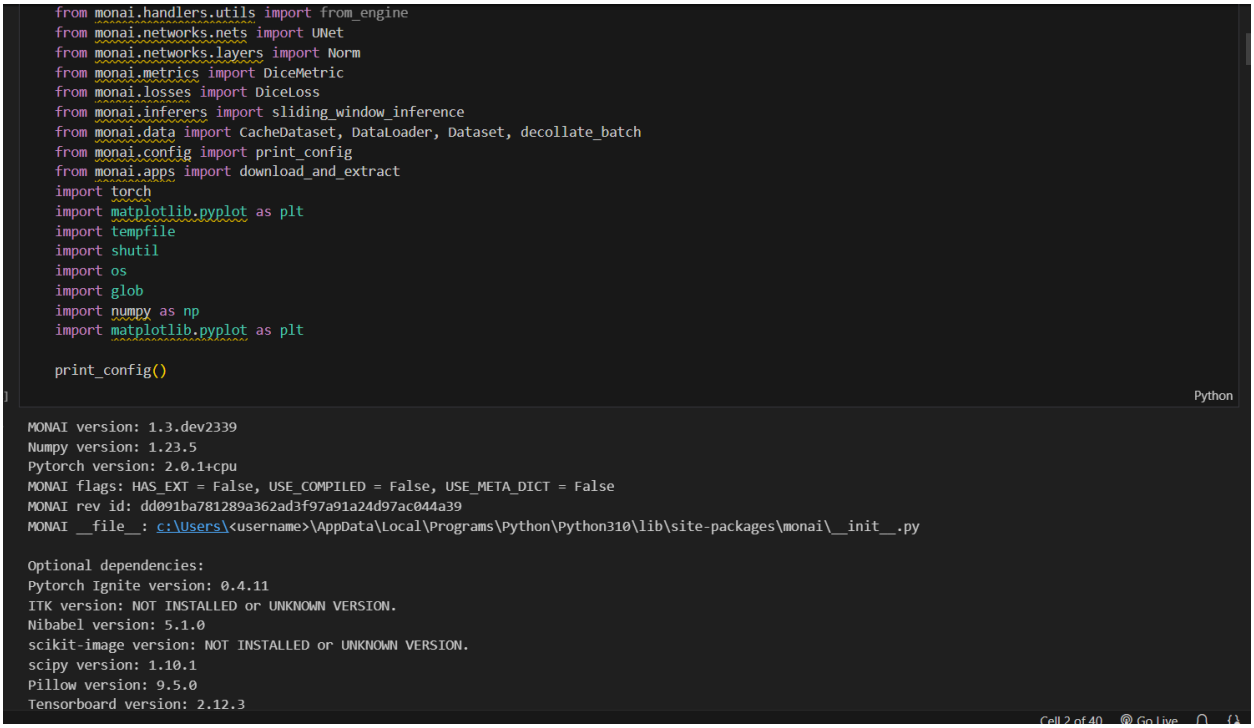
15. Yuan, Z., Puyol-Antón, E., Jogeesvaran, H., Reid, C., Inusa, B., & King, A. P. (2020). Deep Learning for Automatic Spleen Length Measurement in Sickle Cell Disease Patients. In Medical Ultrasound, and Preterm, Perinatal and Paediatric Image Analysis: First International Workshop, ASMUS 2020, and 5th International Workshop, PIPPI 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4-8, 2020, Proceedings 1 (pp. 33-41). Springer International Publishing.

APPENDIX

CODE AND OUPUT SCREENSHOTS:

UNET:





```
For details about installing the optional dependencies, please visit:
    https://docs.monai.io/en/latest/installation.html#installing-the-recommended-dependencies

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Setup data directory

You can specify a directory with the `MONAI_DATA_DIRECTORY` environment variable.
This allows you to save results and reuse downloads.
If not specified a temporary directory will be used.

```python
directory = os.environ.get("MONAI_DATA_DIRECTORY")
root_dir = tempfile.mkdtemp() if directory is None else directory
print(root_dir)
```

[3] Python

```
C:\Users\jothi\AppData\Local\Temp\tmpty2fersv
```

Download dataset

Downloads and extracts the dataset.
The dataset comes from http://medicaldecathlon.com/.

```python
resource = "https://msd-for-monai.s3-us-west-2.amazonaws.com/Task09_Spleen.tar"
md5 = "410d4a301da4e5b2f6f86ec3ddba524e"
```

Download dataset

Downloads and extracts the dataset.
The dataset comes from http://medicaldecathlon.com/.

```python
resource = "https://msd-for-monai.s3-us-west-2.amazonaws.com/Task09_Spleen.tar"
md5 = "410d4a301da4e5b2f6f86ec3ddba524e"

compressed_file = os.path.join(root_dir, "Task09_Spleen.tar")
data_dir = os.path.join(root_dir, "Task09_Spleen")
if not os.path.exists(data_dir):
    download_and_extract(resource, compressed_file, root_dir, md5)
```

4] Py

```
Task09_Spleen.tar: 1.50GB [04:37, 5.80MB/s]
2023-10-30 23:13:21,641 - INFO - Downloaded: C:\Users\jothi\AppData\Local\Temp\tmpty2fersv\Task09_Spleen.tar

2023-10-30 23:13:23,982 - INFO - Verified 'Task09_Spleen.tar', md5: 410d4a301da4e5b2f6f86ec3ddba524e.
2023-10-30 23:13:23,983 - INFO - Writing into directory: C:\Users\jothi\AppData\Local\Temp\tmpty2fersv.
```

Set MSD Spleen dataset path

```python
train_images = sorted(glob.glob(os.path.join(data_dir, "imagesTr", "*.nii.gz")))
train_labels = sorted(glob.glob(os.path.join(data_dir, "labelsTr", "*.nii.gz")))
data_dicts = [{"image": image_name, "label": label_name} for image_name, label_name in zip(train_images, train_labels)]
train_files, val_files = data_dicts[:-9], data_dicts[-9:]
```

5] Py

Set deterministic training for reproducibility

```
[6]
set_determinism(seed=0)
```
Python

Setup transforms for training and validation

Here we use several transforms to augment the dataset:

1. `LoadImaged` loads the spleen CT images and labels from NIfTI format files.
2. `EnsureChannelFirstd` ensures the original data to construct "channel first" shape.
3. `Orientationd` unifies the data orientation based on the affine matrix.
4. `Spacingd` adjusts the spacing by `pixdim=(1.5, 1.5, 2.)` based on the affine matrix.
5. `ScaleIntensityRanged` extracts intensity range [-57, 164] and scales to [0, 1].
6. `CropForegroundd` removes all zero borders to focus on the valid body area of the images and labels.
7. `RandCropByPosNegLabeld` randomly crop patch samples from big image based on pos / neg ratio.
   The image centers of negative samples must be in valid body area.
8. `RandAffined` efficiently performs `rotate`, `scale`, `shear`, `translate`, etc. together based on PyTorch affine transform.

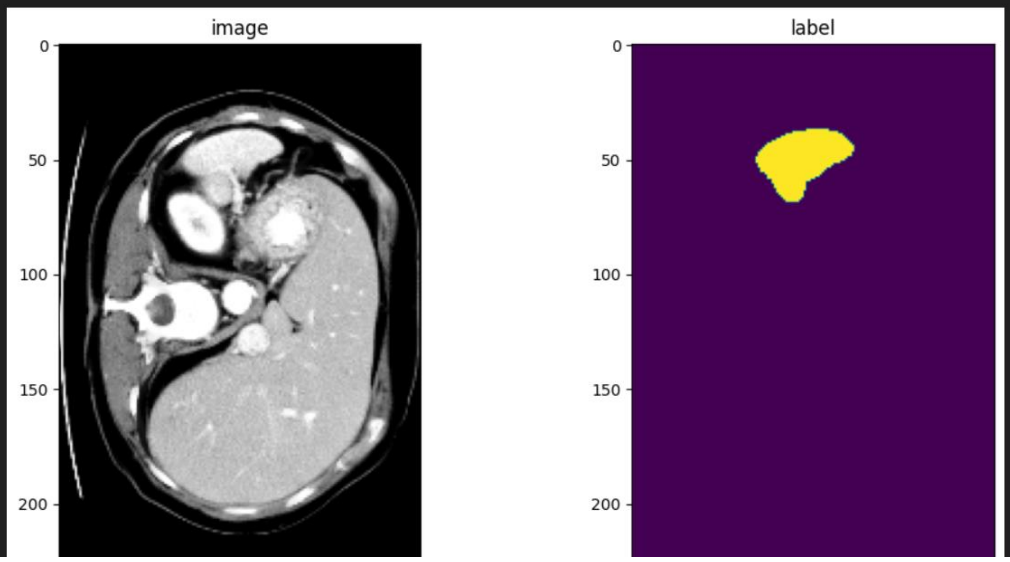
```
train_transforms = Compose(
    [
        LoadImaged(keys=["image", "label"]),
        EnsureChannelFirstd(keys=["image", "label"]),
        ScaleIntensityRanged(
            keys=["image"],
            a_min=-57,
            a_max=164,
```
Cell 2 of 40 Go Live

```
        RandCropByPosNegLabeld(
            keys=["image", "label"],
            label_key="label",
            spatial_size=(96, 96, 96),
            pos=1,
            neg=1,
            num_samples=4,
            image_key="image",
            image_threshold=0,
        ),
    ]
)
val_transforms = Compose(
    [
        LoadImaged(keys=["image", "label"]),
        EnsureChannelFirstd(keys=["image", "label"]),
        ScaleIntensityRanged(
            keys=["image"],
            a_min=-57,
            a_max=164,
            b_min=0.0,
            b_max=1.0,
            clip=True,
        ),
        CropForegroundd(keys=["image", "label"], source_key="image"),
        Orientationd(keys=["image", "label"], axcodes="RAS"),
        Spacingd(keys=["image", "label"], pixdim=(1.5, 1.5, 2.0), mode=("bilinear", "nearest")),
    ]
)
```
Python

```
    plt.title("image")
    plt.imshow(image[:, :, 80], cmap="gray")
    plt.subplot(1, 2, 2)
    plt.title("label")
    plt.imshow(label[:, :, 80])
    plt.show()
```

[12]

```
image shape: torch.Size([228, 158, 113]), label shape: torch.Size([228, 158, 113])
```



Define CacheDataset and DataLoader for training and validation

Here we use CacheDataset to accelerate training and validation process, it's 10x faster than the regular Dataset.
To achieve best performance, set `cache_rate=1.0` to cache all the data, if memory is not enough, set lower value.
Users can also set `cache_num` instead of `cache_rate`, will use the minimum value of the 2 settings.
And set `num_workers` to enable multi-threads during caching.
If want to to try the regular Dataset, just change to use the commented code below.

```
    train_ds = CacheDataset(data=train_files, transform=train_transforms, cache_rate=1.0, num_workers=4)
    # train_ds = Dataset(data=train_files, transform=train_transforms)

    # use batch_size=2 to load images and use RandCropByPosNegLabeld
    # to generate 2 x 4 images for network training
    train_loader = DataLoader(train_ds, batch_size=2, shuffle=True, num_workers=4)

    val_ds = CacheDataset(data=val_files, transform=val_transforms, cache_rate=1.0, num_workers=4)
    # val_ds = Dataset(data=val_files, transform=val_transforms)
    val_loader = DataLoader(val_ds, batch_size=1, num_workers=4)
```

[13] Python

```
Loading dataset:   0%|          | 0/32 [00:00<?, ?it/s]
Loading dataset: 100%|██████████| 32/32 [00:22<00:00,  1.40it/s]
Loading dataset: 100%|██████████| 9/9 [00:05<00:00,  1.79it/s]
```

Create Model, Loss, Optimizer

```python
# standard PyTorch program style: create UNet, DiceLoss and Adam optimizer
device = torch.device("cuda:0")
model = UNet(
    spatial_dims=3,
    in_channels=1,
    out_channels=2,
    channels=(16, 32, 64, 128, 256),
    strides=(2, 2, 2, 2),
    num_res_units=2,
    norm=Norm.BATCH,
)
loss_function = DiceLoss(to_onehot_y=True, softmax=True)
optimizer = torch.optim.Adam(model.parameters(), 1e-4)
dice_metric = DiceMetric(include_background=False, reduction="mean")
```

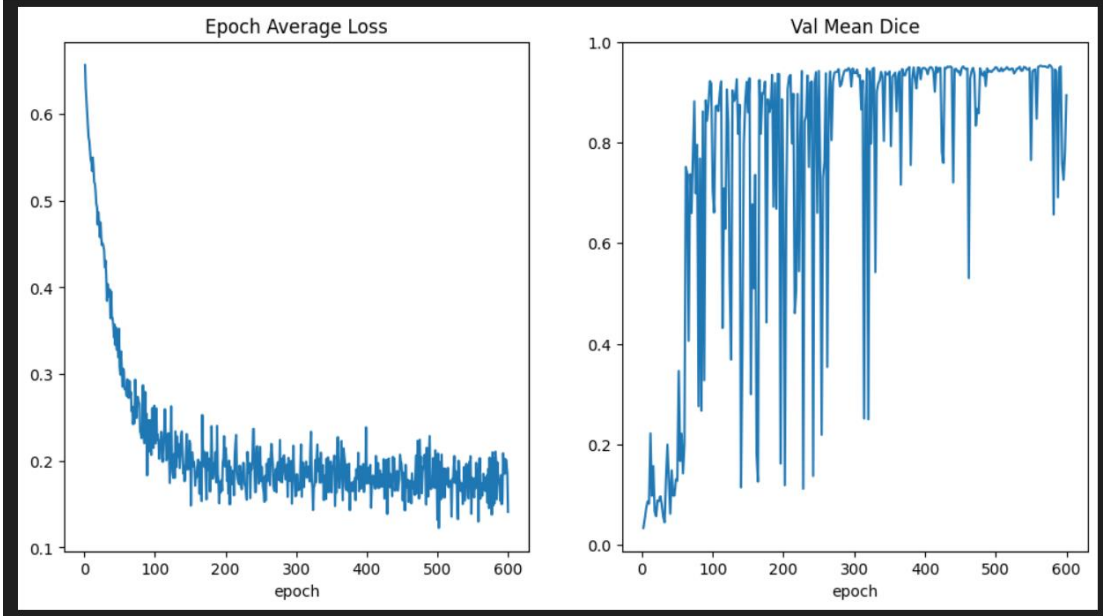
[14] Python

+ Code + Markdown

Execute a typical PyTorch training process

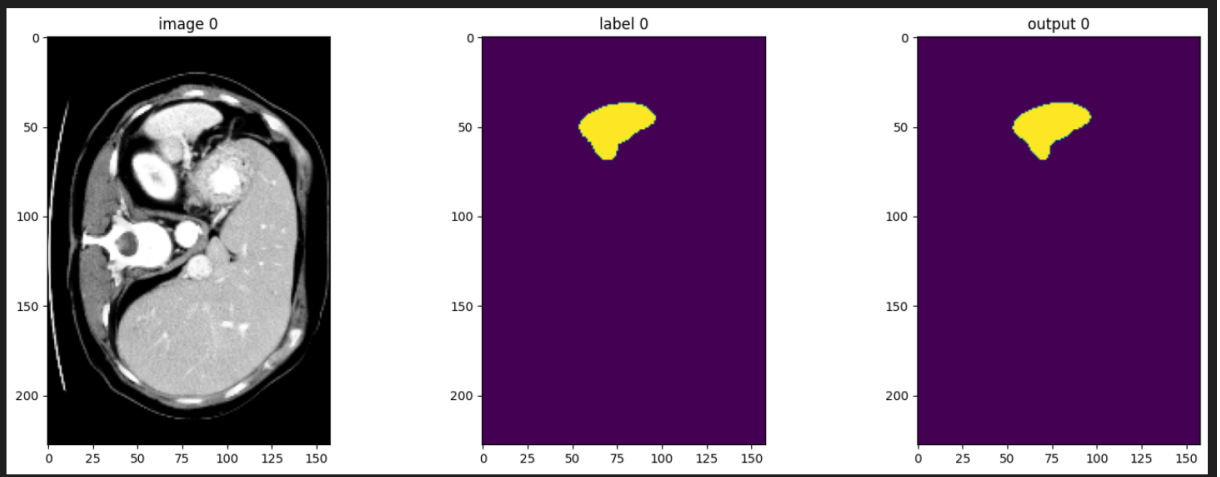
```python
max_epochs = 600
val_interval = 2
best_metric = -1
best_metric_epoch = -1
epoch_loss_values = []
metric_values = []
post_pred = Compose([AsDiscrete(argmax=True, to_onehot=2)])
post_label = Compose([AsDiscrete(to_onehot=2)])

for epoch in range(max_epochs):
    print("-" * 10)
    print(f"epoch {epoch + 1}/{max_epochs}")
    model.train()
    epoch_loss = 0
    step = 0
    for batch_data in train_loader:
        step += 1
        inputs, labels = (
            batch_data["image"],
            batch_data["label"]
        )
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = loss_function(outputs, labels)
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()
        print(f"{step}/{len(train_ds) // train_loader.batch_size}, " f"train_loss: {loss.item():.4f}")
    epoch_loss /= step
```

```
plt.plot(x, y)
plt.show()
```



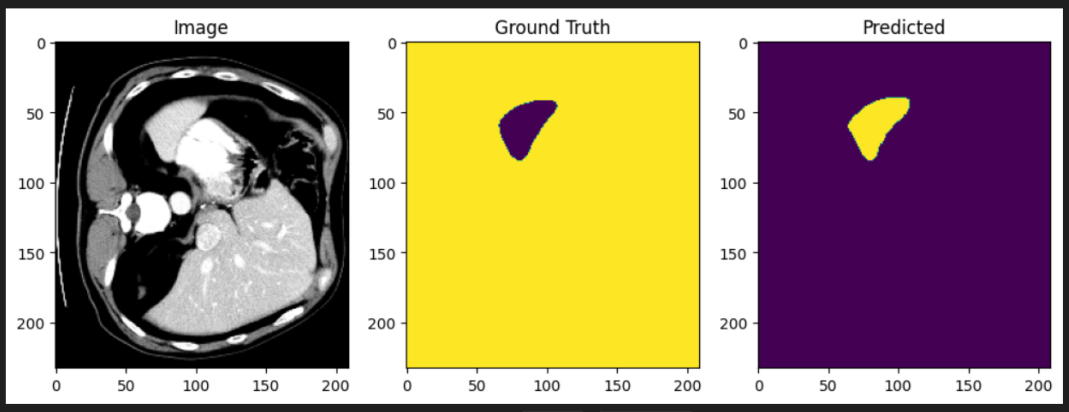

```
            plt.imshow(val_data["label"][0, 0, :, :, 80])
            plt.subplot(1, 3, 3)
            plt.title(f"output {i}")
            plt.imshow(torch.argmax(val_outputs, dim=1).detach().cpu()[0, :, :, 80])
            plt.show()
            if i == 2:
                break
```

[18] Python



```
plt.subplot(1, 3, 1)
plt.title("Image")
plt.imshow(val_data["image"][i, 0, :, :, 80], cmap="gray")
plt.subplot(1, 3, 2)
plt.title("Ground Truth")
plt.imshow(ground_truth_seg[0, :, :, 80])
plt.subplot(1, 3, 3)
plt.title("Predicted")
plt.imshow(predicted_seg[1, :, :, 80])
plt.show()
```

[2] Pyth



+ Code + Markdown

```
# Calculate the overall Dice coefficient for the dataset
mean_dice = np.mean(dice_scores)
print(f"Mean Dice coefficient: {mean_dice:.4f}")
```

[31] Python

```
Mean Dice coefficient: 0.9991
```

INET:

```python
import monai
from monai.networks.nets import UNet
import torch
import torch.nn as nn
from monai.losses import DiceLoss
from monai.metrics import DiceMetric
from monai.transforms import Compose, AsDiscrete
from monai.inferers import sliding_window_inference
from monai.data import decollate_batch


# Define device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
import torch.optim as optim
from torch.optim import Adam




class INet(nn.Module):
    def __init__(self):
        super(INet, self).__init__()



        # Use a UNet as the backbone

        self.unet = UNet(
            spatial_dims=3,
            in_channels=1,
            out_channels=2,
            channels=(16, 32, 64, 128, 256),
            strides=(2, 2, 2, 2)
```

```
            num_res_units=2
            )



        # Instance-aware layers
        self.inst_conv1 = nn.Conv3d(2, 64, kernel_size=1)
        self.inst_conv2 = nn.Conv3d(64, 64, kernel_size=1)


        self.out_layer = nn.Conv3d(64, 2, kernel_size=1)

    def forward(self, x):
        x = self.unet(x)

        # Instance embedding
        emb = self.inst_conv1(x)
        emb = self.inst_conv2(emb)

        # Final output
        x = self.out_layer(emb)

        return x

model = INet()

# Rest of the code remains the same
loss_fn = DiceLoss(to_onehot_y=True, softmax=True)
optimizer = Adam(model.parameters(), lr=1e-4)
dice_metric = DiceMetric(include_background=False, reduction="mean")
```

```python
# Rest of the code remains the same
loss_fn = DiceLoss(to_onehot_y=True, softmax=True)
optimizer = Adam(model.parameters(), lr=1e-4)
dice_metric = DiceMetric(include_background=False, reduction="mean")

# Train model
for epoch in range(10):
   max_epochs = 10
val_interval = 2
best_metric = -1
best_metric_epoch = -1
epoch_loss_values = []
metric_values = []
post_pred = Compose([AsDiscrete(argmax=True, to_onehot=2)])
post_label = Compose([AsDiscrete(to_onehot=2)])

for epoch in range(max_epochs):
    print("-" * 10)
    print(f"epoch {epoch + 1}/{max_epochs}")
    model.train()
    epoch_loss = 0
    step = 0
    for batch_data in train_loader:
        step += 1
        inputs, labels = (
            batch_data["image"],
            batch_data["label"]
        )
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = loss_fn(outputs, labels)
        loss.backward()
        optimizer.step()
```

```python
        epoch_loss += loss.item()
        print(f"{step}/{len(train_ds) // train_loader.batch_size}, " f"train_loss: {loss.item():.4f}")
    epoch_loss /= step
    epoch_loss_values.append(epoch_loss)
    print(f"epoch {epoch + 1} average loss: {epoch_loss:.4f}")

    if (epoch + 1) % val_interval == 0:
        model.eval()
        with torch.no_grad():
            val_dice_coefficients = []  # List to store Dice coefficients for this validation epoch
            for val_data in val_loader:
                val_inputs, val_labels = (
                    val_data["image"],
                    val_data["label"]
                )
                roi_size = (160, 160, 160)
                sw_batch_size = 4
                val_outputs = sliding_window_inference(val_inputs, roi_size, sw_batch_size, model)
                val_outputs = [post_pred(i) for i in decollate_batch(val_outputs)]
                val_labels = [post_label(i) for i in decollate_batch(val_labels)]
                # compute metric for current iteration
                dice_metric(y_pred=val_outputs, y=val_labels)
                val_dice_coefficients.append(dice_metric.aggregate().item())
                # Calculate and display the mean Dice coefficient for this validation epoch
                mean_dice_coefficient = np.mean(val_dice_coefficients)
                print(
                    f"current epoch: {epoch + 1} current mean dice: {mean_dice_coefficient:.4f}"
                    f"\nbest mean dice: {best_metric:.4f} at epoch: {best_metric_epoch}"
                    )

            # aggregate the final mean dice result
            metric = dice_metric.aggregate().item()
            # reset the status for next validation round
            dice_metric.reset()
```

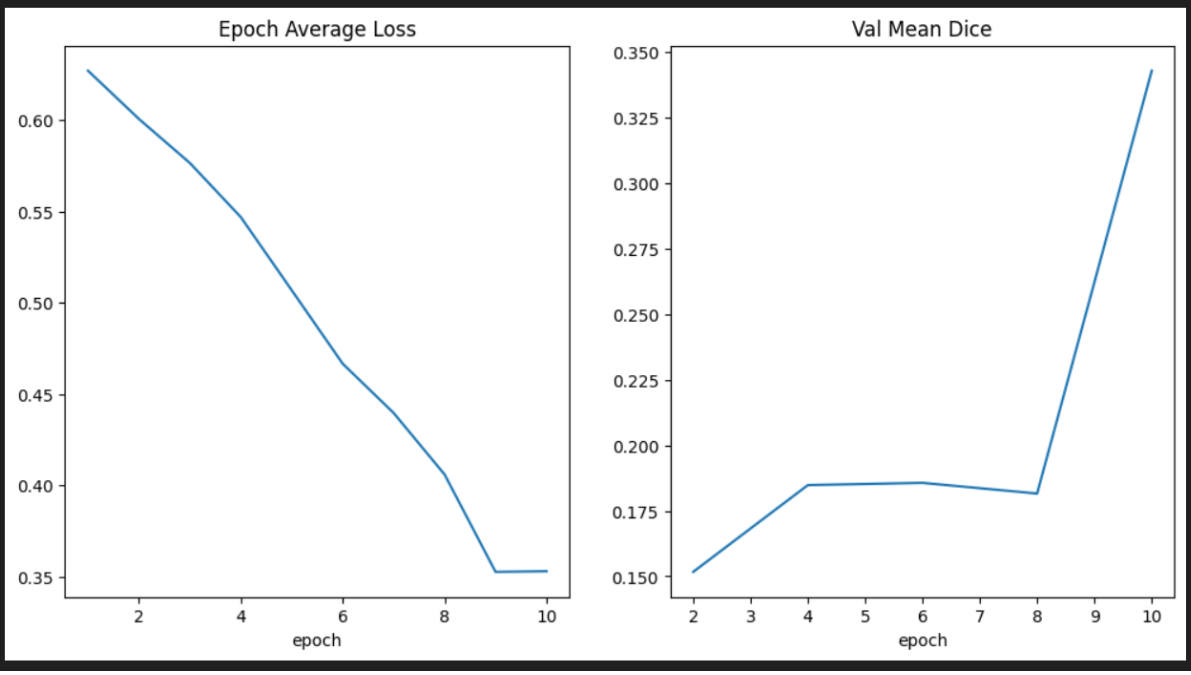
```
            # aggregate the final mean dice result
            metric = dice_metric.aggregate().item()
            # reset the status for next validation round
            dice_metric.reset()

            metric_values.append(metric)
            if metric > best_metric:
                best_metric = metric
                best_metric_epoch = epoch + 1
                torch.save(model.state_dict(), os.path.join(root_dir, "D:/Jodi/Sem7/medical/project
                print("saved new best metric model")
            print(
                f"current epoch: {epoch + 1} current mean dice: {metric:.4f}"
                f"\nbest mean dice: {best_metric:.4f} "
                f"at epoch: {best_metric_epoch}"
            )
```

```
----------
epoch 1/10
1/16, train_loss: 0.6454
2/16, train_loss: 0.6597
3/16, train_loss: 0.6335
4/16, train_loss: 0.6456
5/16, train_loss: 0.6111
6/16, train_loss: 0.5897
7/16, train_loss: 0.6374
8/16, train_loss: 0.6356
9/16, train_loss: 0.6215
10/16, train_loss: 0.6299
```

```
plt.show()
```

[13]

...


Epoch Average Loss
0.60
0.55
0.50
0.45
0.40
0.35
2
4
6
8
10
epoch
Val Mean Dice
0.350
0.325
0.300
0.275
0.250
0.225
0.200
0.175
0.150
2
3
4
5
6
7
8
9
10
epoch

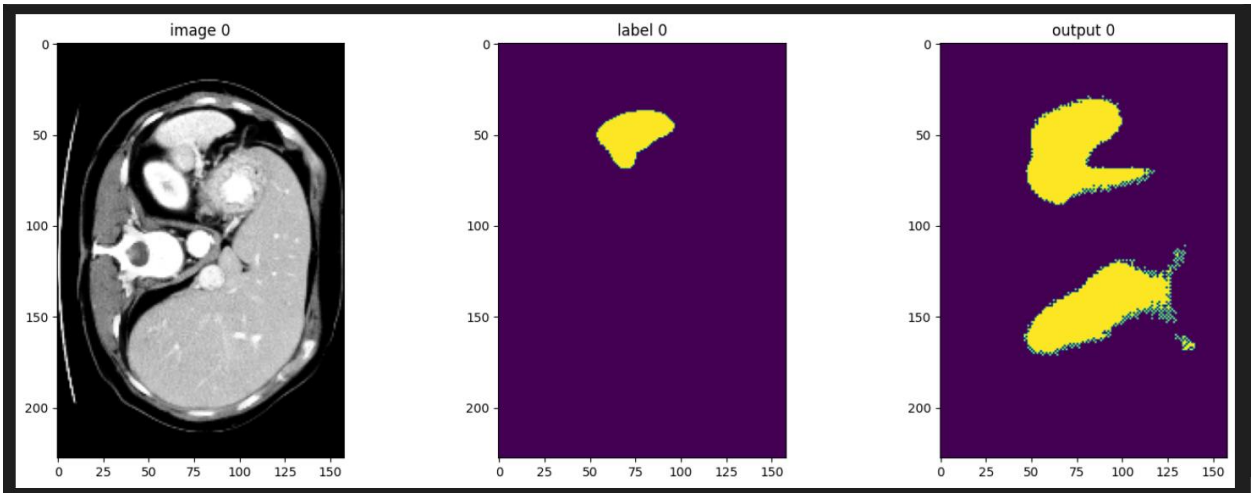


image 0
label 0
output 0
0
50
100
150
200
0
25
50
75
100
125
150

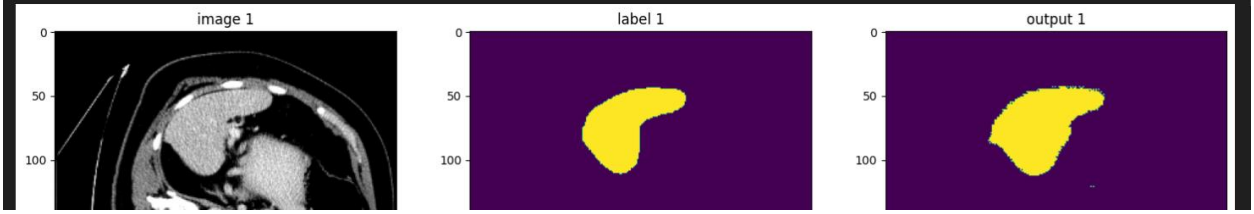
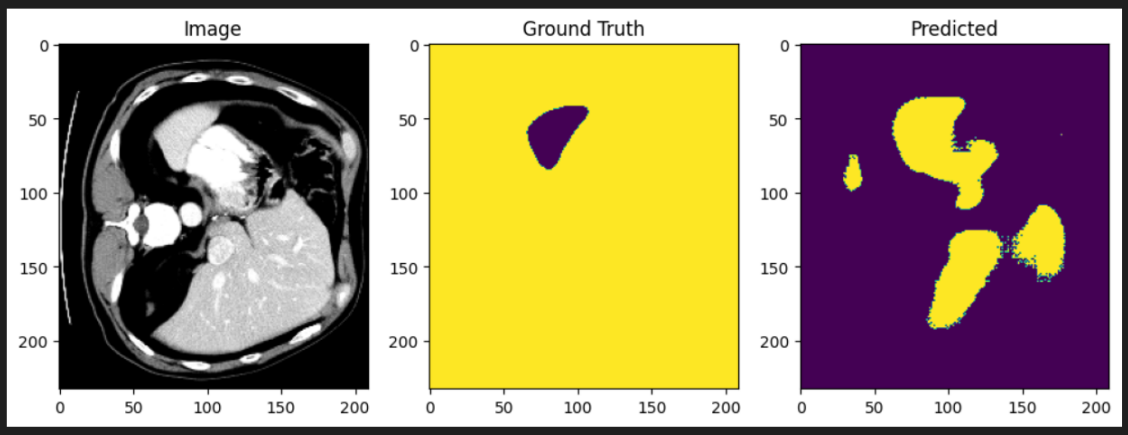


image 1
label 1
output 1
0
50
100

[22]




```
# Calculate the overall Dice coefficient for the dataset
mean_dice = np.mean(dice_scores)
print(f"Mean Dice coefficient: {mean_dice:.4f}")
```

[23]

```
Mean Dice coefficient: 0.9673
```