Project Report on

Automatic Multi-Organ Segmentation for Pre-Surgery Analysis

Submitted in partial fulfillment of the course of

PG-Diploma

in

Big Data Analytics

From **C-DAC ACTS (Bangalore)**

Guided by
Mr. Abhay Mane

Presented by

Pankaj Sahu	PRN: 220950125060
Patki Sagar Sanjay	PRN: 220950125063
Potdar Shailesh Hemant	PRN: 220950125065
Prachee Adey	PRN: 220950125066
Raja Gupta	PRN: 220950125071

CANDIDATES' DECLARATION

We hereby certify that the work being presented in the report titled: **Automatic Multi-Organ Segmentation for Pre-Surgery Analysis**, in partial fulfillment of the requirements for the award of PG Diploma Certificate and submitted to the department of PG-DBDA of the C-DAC ACTS Bangalore, is an authentic record of our work carried out during the period, 15th September 2022 to 15th March 2023 under the supervision of Mr. Abhay Mane, C-DAC Bangalore. The matter presented in the report has not been submitted by us for the award of any degree of this or any other Institute/University.

Name and Signature of Candidate:

Pankaj Sahu	PRN: 220950125060
Patki Sagar Sanjay	PRN: 220950125063
Potdar Shailesh Hemant	PRN: 220950125065
Prachee Adey	PRN: 220950125066
Raja Gupta	PRN: 220950125071

Counter Signed by

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Pankaj Sahu

Patki Sagar Sanjay

Potdar Shailesh Hemant

Prachee Adey

Raja Gupta

Have successfully completed their project on

Automatic Multi-Organ Segmentation for Pre-Surgery Analysis

Under the guidance of

Mr. Abhay Mane

Mr. Abhay Mane	Mr. Arun Shankar
(Project Guide)	(Course Coordinator)

ACKNOWLEDGEMENT

This project **“Automatic Multi Organ Segmentation for Pre-Surgery Analysis”** was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We all are very glad to mention the name of **<u>Mr. Abhay Mane</u>** for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during project work.

Our most heartfelt thanks go to **<u>Mr. Arun Shankar</u>** (Course Coordinator, **PG-DBDA**) who gave all the required support and kind coordination to provide all the necessities like extra Lab hours to complete the project and throughout the course up to the last day at C-DAC ACTS, Bangalore.

From,

The Whole Team.

INDEX

ABSTRACT

Organ segmentation is an important step in medical image analysis, particularly for pre-surgery analysis. Pre-surgery analysis involves studying medical images such as CT or MRI scans of a patient's body to identify the location, size and extent of a tumor or other abnormalities that may require surgery. Organ segmentation refers to the process of identifying and delineating the boundaries of organs or structures within medical images.

Regarding the difficulties that we can encounter when using traditional image processing tools, deep learning has emerged as the primary solution in the healthcare field because medical images are more difficult to process than standard images (dense contrast, a wide range of variations in the human body). Deep learning is used for classification, object detection, and especially segmentation tasks.

Deep learning is used to segment human organs such as the liver, lung and or to segment tumors from different parts of the body. There are many different types of medical images, such as MRI (which is mostly used for brain tumor segmentation), CT scans, PET scans. So, we know that performing a deep learning task necessitates several steps, one of which is data pre-processing, which is the first thing we must do before launching a training. This is the topic of this article; we will discuss the tools available to us for performing this pre-process. Preparing the data varies depending on the task; for example, classification is the easiest because we only need to prepare the images, whereas object detection and segmentation require us to prepare the image as well as the labels (bounding boxes or masks for segmentation).

We will use segmentation as an example, which can be used for tumor or organ segmentation. In medical imaging, we can work with 2D images, which can be dicoms, JPGs, or PNGs, or 3D volumes, which are groups of slices, each slice is a 2D file (most of the time dicoms), and this group is a nifti file that represents the entire patient or only a portion of his body.

INTRODUCTION

Deep learning can be used to detect diseases like cancer by finding tumour cells through medical images. Deep learning also has the potential to improve the quality of medical care by segmenting organs during surgery or scanning patients for signs of cancer or other ailments. Deep learning in computer vision can be used for a variety of tasks, the most common of which is image classification, which is one of the first deep learning applications.

The image classification technique simply determines whether an image contains the object that we are looking for. Then there is object detection, which is another extremely useful technique in computer vision. This technique entails drawing a box around the object we are looking for in the image.

Image segmentation, on the other hand, is a technique that combines image classification and object detection. This technique not only detects an object in an image, but it also segments the object's correct pixels.

There are two types of image segmentation, the first is referred to as semantic segmentation and the second is referred to as instance segmentation. The distinction between these two types is that semantic segmentation only produces pixel probabilities about whether a pixel belongs to one of two classes. The goal of instance segmentation is to index objects even if they belong to the same class. In our case, semantic segmentation will be used because we need to know whether the slice contains a liver or not.

Accurate organ segmentation is essential for pre-surgery analysis because it allows doctors to better understand the anatomy of the patient and plan the surgical procedure accordingly. For example, if a tumor is located close to a critical organ, the surgeon may need to modify the surgical approach to avoid damaging the organ during the procedure. Additionally, organ segmentation can help to minimize the risk of complications during surgery such as bleeding or nerve damage, by allowing the surgeon to visualize the precise location and extent of the abnormality.

Organ segmentation can also be used to generate 3D models of the patient's organs or structures, which can be used for surgical simulation or preoperative planning. By simulating the surgery beforehand, the surgeon can identify potential complications and develop a plan to mitigate them, ultimately leading to a safer and more effective surgery.

DATASET

Basics of Medical Images

Medical images are visual representations of the internal structures and functions of the human body. They are typically captured using various imaging techniques, such as X-rays, magnetic resonance imaging (MRI), computed tomography (CT), ultrasound, and positron emission tomography (PET).

Here are some basics of medical images:

- **X-rays**: X-rays are a form of electromagnetic radiation that can pass through the body and produce images of the internal structures. X-rays are commonly used to diagnose bone fractures, dental problems, and chest conditions such as pneumonia.
- **MRI**: MRI uses a powerful magnet and radio waves to produce detailed images of the body's internal structures. MRI is useful for diagnosing a wide range of conditions, including brain and spinal cord disorders, joint problems, and cancers.
- **CT**: CT uses X-rays and computer technology to produce detailed cross-sectional images of the body. CT is useful for diagnosing conditions such as tumors, blood clots, and bone fractures.
- **Ultrasound**: Ultrasound uses high-frequency sound waves to produce images of the body's internal structures. It is commonly used to diagnose conditions such as pregnancy, heart problems, and liver disease.
- **PET**: PET uses a radioactive substance called a tracer to produce images of the body's internal functions. PET is useful for diagnosing conditions such as cancer, heart disease, and neurological disorders.

Medical images are typically interpreted by a trained radiologist or other medical specialist. These professionals use their expertise to identify abnormalities or areas of concern and make a diagnosis or recommend further testing or treatment.

Need of Labeled Dataset

Labeled data is crucial for image segmentation because it is necessary to train machine learning algorithms to accurately segment objects within an image. Image segmentation involves partitioning an image into multiple segments or regions, where each segment corresponds to a particular object or background.

Labeled data provides a set of predefined ground truth values, which are used to train machine learning

models to accurately segment images. This training involves the machine learning model learning to identify the objects in an image based on their characteristics, such as color, texture, and shape.
Without labelled data, it is difficult to accurately train machine learning models to perform image segmentation. This is because machine learning models require large amounts of data to learn how to accurately segment objects in an image. Labelled data provides this necessary training data by providing pre-defined ground truth values that can be used to train the machine learning model.
Furthermore, labelled data is essential for validating the accuracy of the image segmentation algorithms. Once a model has been trained on a labelled dataset, it can be tested on a separate set of labelled images to ensure that it accurately segments objects in new images. Without labelled data, it would be impossible to validate the accuracy of the segmentation algorithm.

DICOM

DICOM stands for Digital Imaging and Communications in Medicine, which is a standard used in medical imaging for the communication and management of medical image data. DICOM files are digital files that contain medical images and associated patient data in a standardized format.
DICOM files are used to transfer, store, and display medical images, such as X-rays, CT scans, MRIs, and ultrasound images. The standard specifies the format and structure of the data, including information such as patient demographics, study and series information, and image pixel data. It also provides guidelines for the communication of this data between different medical imaging devices and systems.
DICOM files are beneficial because they enable interoperability between different medical imaging systems and allow medical professionals to easily access and share patient data and images. This improves patient care and facilitates collaboration between medical professionals. DICOM files can be viewed using specialized software that can display and manipulate medical images. Many medical imaging systems and electronic health record systems support DICOM files, making them a widely used standard in the medical industry.

NIfTI

NIfTI stands for Neuroimaging Informatics Technology Initiative, which is a standard used in the field of neuroimaging to store and share neuroimaging data, such as magnetic resonance imaging (MRI) and functional MRI (fMRI) data. NIfTI files are digital files that contain neuroimaging data in a standardized format.

NIfTI files are used to store image data in 3D and 4D (i.e., time-series) formats. They can contain various types of neuroimaging data, including structural MRI, diffusion MRI, and functional MRI. NIfTI files are beneficial because they provide a standardized format for neuroimaging data, allowing researchers and clinicians to easily share and analyze data across different imaging platforms and software tools. This improves collaboration between different research groups and facilitates the development of new analysis methods and algorithms. NIfTI files can be viewed using specialized software, such as 3D Slicer and ITK Snap, that can display and manipulate neuroimaging data. Many neuroimaging research tools and software packages support NIfTI files, making them a widely used standard in the field of neuroimaging.

File Conversion

Converting NIfTI files to DICOM and vice versa is useful for medical professionals and researchers who need to share or integrate neuroimaging data between different software platforms or medical imaging systems. Here are some methods for converting between NIfTI and DICOM file formats:

Converting NIfTI files to DICOM

One way to convert NIfTI files to DICOM is to use specialized software such as 3D Slicer, which is a free, open-source tool that can convert NIfTI files to DICOM format. This tool can be used from the command line or via a graphical user interface (GUI).

We used **pydicom** package to convert nifti files to dicom.

Converting DICOM files to NIfTI

One way to convert DICOM files to NIfTI is to use specialized software such as 3D Slicer or MRIConvert, which can convert DICOM files to NIfTI format. These tools can be used from the command line or via a GUI.

When converting between file formats, it is important to note that some information may be lost or altered in the process, such as patient information, metadata, or image quality. It is also important to ensure that the converted files are properly formatted and validated before using them for clinical or research purposes.

We used **SimpleITK** package to convert dicom files to nifti.

METHODOLOGY

Need for the conversion

The framework we are using is MONAI, which takes input as Nifti files only. The CT scan Nifti files used in the dataset are collected from various sources. Thus, each file contains a different number of slices. It can cause errors while training. Hence, we first need to convert Nifti files to Dicom and then make groups/folders containing the same number of slices.
Then we have to convert all folders containing dicoms to nifti files.

Creating Groups of 65 Slices

When it is a public dataset or even it is its own data, the number of slices for each patient differs. And if it is not dealt with, this discrepancy before heading to the training, it could become an issue. Keeping in mind that we are doing 3D segmentation, input volumes should all be the same size. A dataset that includes the exact number of slices for all the patients is never received.
There is no need to establish the groups if all the patients have a similar number of slices. It is because we executed a resize in the pre-processing section, all the inputs have the same number of slices. However, if a patient has 300 slices and it is reduced to 100 or 60 slices, there is chance of losing a significant number of slices that may carry vital information about the body.
The training was very slow and your GPU/CPU memory may not handle that size of inputs. If a patient with several slices equal to 60 into 300 slices is enlarged, all the information will be lost. As a result, we needed to make small groups of slices that each represent a small portion of a patient, and then combine them to acquire the entire volume.
Steps are:

1- Convert all your nifti files into dicoms
If we already have dicom files, there is no need to convert them to nifti right away; otherwise, we need to convert them to dicom to establish tiny groups. There is no special python script that can perform this conversion. The problem is that it gives all dicom series the same index, which means all the slices will have the same place in the body and we cannot convert them back to nifties.
Thus, we can perform this conversion with the software 3D slicer.

2- Move every 65 slices into a folder

We can start putting a specified number of slices in a folder that represents a small portion of the body after we have all the patients in dicom format, 65 slices, but can be randomly chosen; try not to pass the 100 slices by a sub patient.

Data Augmentation

Data augmentation is a technique used in machine learning and deep learning to increase the amount of training data by artificially creating new samples from the existing dataset. It involves applying various transformations or manipulations to the original data to create new data samples that are similar but not identical to the original samples.

The purpose of data augmentation is to improve the robustness and generalization of the trained model. By creating additional training samples, the model is exposed to more variations and patterns in the data, making it more capable of accurately recognizing and classifying new, unseen data.

Some common data augmentation techniques include:

- Rotation: rotating the image by a certain degree.
- Flipping: flipping the image horizontally or vertically.
- Scaling: resizing the image to a different size.
- Translation: shifting the image horizontally or vertically.
- Shearing: tilting the image along the x or y axis.
- Adding noise: adding random noise to the image.

Data augmentation can be applied to different types of data, such as images, audio, and text. It is a popular technique in computer vision, where large datasets are often required to train deep learning models effectively.

Algorithm for Prediction

Convolutional Neural Networks (CNNs) are a type of deep neural network that are commonly used for image recognition, computer vision, and natural language processing tasks. CNNs are designed to process inputs with a grid-like topology, such as images or sound waves. The key components of a CNN include:

- Convolutional Layers: These layers use filters to extract features from the input data. The filters slide over the input data, performing element-wise multiplication and summation to create a feature map. Multiple filters are typically used to extract different types of features.
- Pooling Layers: These layers downsample the feature maps generated by the convolutional layers, reducing the spatial dimensions of the data. This helps to reduce the number of parameters in the model and prevent overfitting.
- Activation Functions: These functions introduce non-linearity to the model, allowing it to learn complex patterns in the data.
- Fully-Connected Layers: These layers are used to make the final predictions based on the features learned by the convolutional and pooling layers.
- Dropout Layers: These layers randomly drop out a fraction of the neurons during training, which can help to prevent overfitting.

The training of a CNN involves feeding it a large dataset of labeled images, and optimizing the network parameters (such as the weights and biases) using an optimization algorithm such as stochastic gradient descent. The performance of the network is then evaluated on a separate test dataset.

Overall, CNNs have been highly successful in a wide range of image recognition and computer vision tasks, and continue to be an active area of research in the field of machine learning.

Steps to apply CNN

Applying a Convolutional Neural Network (CNN) typically involves the following steps:

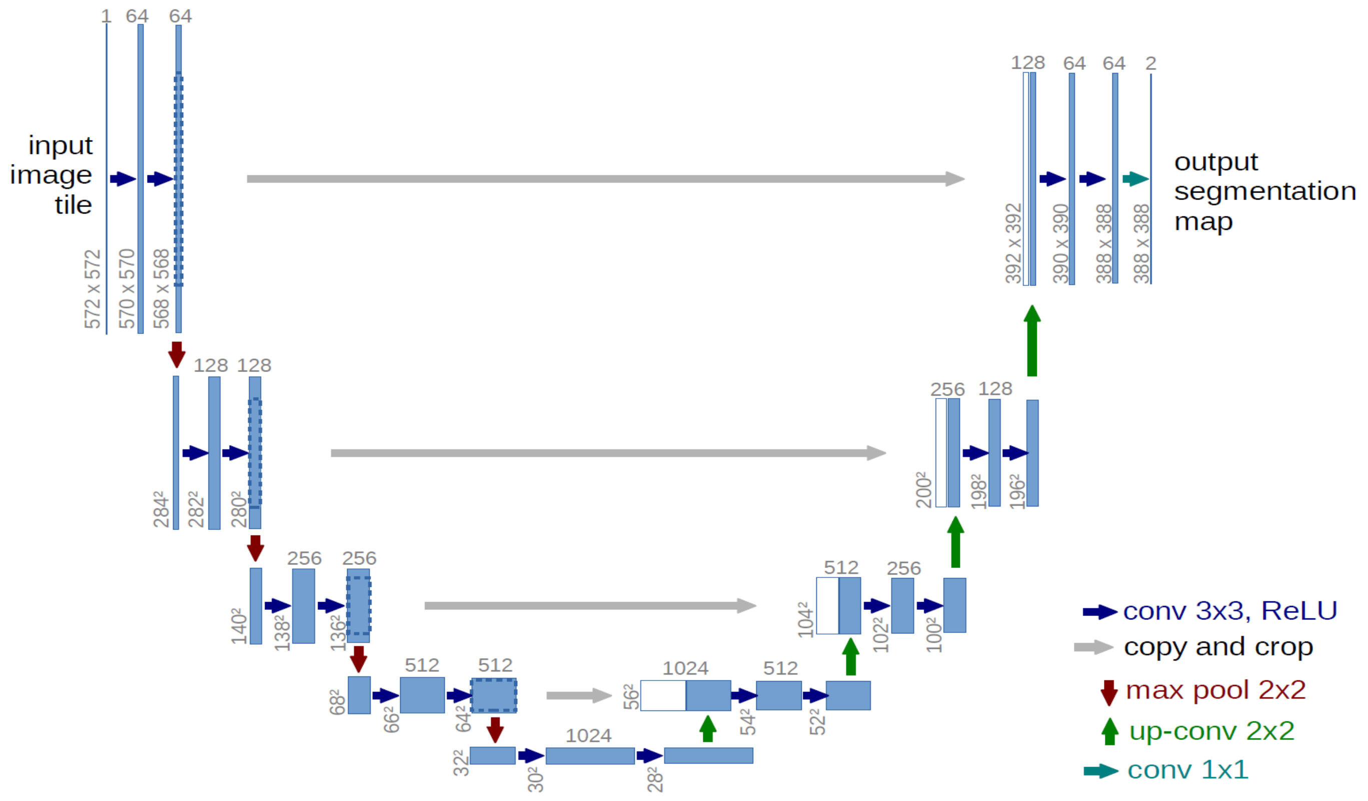
1. Data Preparation: The first step is to prepare your data for training the CNN. This includes selecting the images or data samples to use, cleaning and preprocessing the data, and splitting it into training, validation, and test sets.
2. Model Selection: Choose a CNN architecture that is appropriate for your problem. Some popular CNN architectures include VGG, ResNet, Inception, and MobileNet.
3. Model Compilation: Compile the CNN model by selecting an optimizer, a loss function, and evaluation metrics. The optimizer is used to update the weights of the network during training, while the loss function measures how well the network is performing on the training data.
4. Model Training: Train the CNN model using the training data. During training, the CNN learns to

recognize patterns in the input data and adjust its weights accordingly.

5. Model Evaluation: Evaluate the performance of the CNN model on the validation data. This will give you an idea of how well the model is generalizing to new data.
6. Model Testing: Finally, test the CNN model on the test data to get a final measure of its performance. This will give you an idea of how well the model will perform in the real world.
7. Model Tuning: Iterate on the above steps to improve the performance of the CNN model. You can try different CNN architectures, adjust the hyperparameters of the model, or use data augmentation techniques to improve the model's performance.

U-Net: Convolutional Networks for Biomedical Image Segmentation

The typical use of convolutional networks is on classification tasks, where the output to an image is a single class label. However, in many visual tasks, especially in biomedical image processing, the desired output should include localization, i.e., a class label is supposed to be assigned to each pixel. Moreover, thousands of training images are usually beyond reach in biomedical tasks. Hence, it is trained in a network in a sliding-window setup to predict the class label of each pixel by providing a local region (patch) around that pixel.



U-net architecture (example for 32x32 pixels in the lowest resolution).

Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

The U-Net architecture is a type of convolutional neural network (CNN) that was first proposed for medical image segmentation tasks in 2015. It is named "U-Net" because of its U-shaped architecture, which consists of a contracting path and an expansive path.

The contracting path is made up of convolutional and max-pooling layers that gradually reduce the spatial resolution of the input data, while increasing the number of feature maps. This allows the network to capture local information about the input image.

The expansive path consists of up-convolutional and concatenation layers that gradually increase the spatial resolution of the data, while decreasing the number of feature maps. This allows the network to combine information from different scales, and generate segmentation masks that are aligned with the original input image.

The key innovation of the U-Net architecture is the addition of skip connections that connect corresponding layers in the contracting and expansive paths. These skip connections allow the network to reuse features from earlier layers, and propagate them to later layers, thus preserving fine-grained information about the input image.

Overall, the U-Net architecture has been highly successful in a wide range of medical image segmentation tasks, and has been adapted for use in other computer vision tasks as well. Its ability to capture both local and global information, while preserving fine-grained details, makes it a powerful tool for image analysis and understanding.

Steps for Hyperparameter tuning

Hyperparameter tuning involves selecting the best values for the hyperparameters of a machine learning model. The following are the steps for hyperparameter tuning:

- Define the hyperparameters: Determine which hyperparameters need to be tuned for the model. This will depend on the specific algorithm and implementation being used. Some examples of hyperparameters include learning rate, regularization strength, number of layers, number of nodes per layer, batch size, etc.
- Define a search space: Define a range of values that each hyperparameter can take. This search space can be defined using discrete or continuous values.
- Select a search strategy: Choose a strategy for searching the hyperparameter space. This can be done using a random search, grid search, or other methods such as Bayesian optimization.
- Train and evaluate the model: Train and evaluate the model using different combinations of hyperparameters. Use a portion of the data set as a validation set to determine which

hyperparameter values perform the best.

- Select the best hyperparameters: Once the hyperparameters have been searched, select the combination that performs the best on the validation set.
- Test the model: Finally, test the performance of the model using the selected hyperparameters on the test set.
- Refine the search space: If the model performance is not satisfactory, refine the search space and repeat the hyperparameter tuning process until the desired performance is achieved.

The hyperparameters which we can tune are:

In Preprocessing-

1. ScaleIntensityRanged:
    a. a_min- Intensity original range min
    b. a_max- Intensity original range max
    c. b_min- Intensity target range min
    d. b_max- Intensity target range max

In Training-

Optimizer:

1. LearningRate
2. WeightDecay

PyTorch

PyTorch is an open-source machine learning framework that is primarily used for deep learning applications. It was developed by Facebook's artificial intelligence research group and was first released in 2016. PyTorch is built on top of the Torch library, which is a popular machine learning library in the Lua programming language.

PyTorch provides a Python interface for building and training deep neural networks. It is particularly popular among researchers and academics due to its flexibility and ease of use.

Some of the key features of PyTorch include:

- Dynamic Computational Graphs: PyTorch allows for dynamic computation graphs, which means that the graph is created on the fly as the code is executed. This makes it easier to write and debug complex neural network architectures.
- GPU Acceleration: PyTorch provides support for running computations on GPUs, which can significantly speed up training times for deep learning models.
- Easy Debugging: PyTorch provides easy-to-use debugging tools that help users to identify and

resolve errors in their code.

- TorchScript: PyTorch provides a way to convert Python code to a more efficient, lower-level representation called TorchScript. This can be useful for deploying PyTorch models in production environments.
- Large Community: PyTorch has a large and active community of developers and users, which means that there is a wealth of resources and support available online.

Overall, PyTorch is a powerful and flexible machine learning framework that is well-suited for deep learning tasks. Its ease of use and large community make it a popular choice among researchers and practitioners alike.

MONAI

MONAI (Medical Open Network for AI) is an open-source deep learning framework that is specifically designed for medical imaging applications. It was developed by a team of researchers at the Center for Biomedical Image Computing and Analytics at the University of Pennsylvania, and was first released in 2019.

MONAI provides a Python interface for building and training deep learning models for medical imaging tasks, such as segmentation, classification, and registration. Some of the key features of MONAI include:

- Preprocessing Pipelines: MONAI provides a set of pre-built preprocessing pipelines that can be used to prepare medical images for deep learning tasks. These pipelines include tools for data normalization, cropping, resampling, and augmentation.
- 3D Support: MONAI has native support for processing 3D medical images, which is important for many medical imaging applications.
- Model Zoo: MONAI provides a model zoo that contains pre-trained deep learning models that can be used for various medical imaging tasks. These models can be fine-tuned or adapted for specific use cases.
- Visualization Tools: MONAI provides tools for visualizing medical images and model predictions, which can help with understanding and interpreting the results of deep learning models.
- Distributed Training: MONAI provides support for distributed training, which allows for training deep learning models on multiple GPUs or even multiple machines.

Overall, MONAI is a powerful and specialized deep learning framework that is designed specifically for medical imaging applications. Its pre-processing pipelines, 3D support, model zoo, visualization tools, and distributed training capabilities make it a valuable tool for researchers and practitioners in the field of medical imaging.

Loading dataset

There are two approaches to loading the data:

The first is to load the images and masks individually, this is the way it can be used for image classification but it works also for segmentation.

The second method is to create a Python dictionary with two columns, one for the image paths and one for the label paths. The path of the image with the corresponding mask is then entered in each row.

When we apply the transforms, we select just the key work of either the image or the label, or both, rather than creating transforms for the images and transforms for the labels (masks). Creating a variable that contains the path to the entire data set is necessary. Once the dictionary is created, if it is referred to the first item in the dictionary, index 0 is used in a normal array, but the first item will have two columns, the first of which is the path to the image and the second of which is the path to the label.

To apply multiple transforms to the same patient, we used Monai's **COMPOSE** function, which allows combining any desired transformations (the ones defined in Monai documentation).

When using Monai, the primary transforms are Load image to load the nifty files and **ToTensor** to convert the transformed data into torch tensors so that we can use it for training.

- AddChanneld: This function adds a channel to our image and label (the volume of multiple slices), because when tumor segmentation is done, we need a channel that plays the role of background or tumor.
- Spacingd: This function assists us in changing the voxel dimensions because we do not know if a dataset of medical images was acquired with the same scan or with different scans, so they may have different voxel dimensions (width, height, depth). Thus, we need to generalize all of them to the same dimensions.
- ScallIntensityRanged: This function assists us in performing two tasks at the same time: the first is to change the contrast from that dense vision into something more visible, and the second is to normalize the voxel values and place them between 0 and 1 so that the training will be faster.
- CropForegroundd: This function assists us in cropping out the empty regions of the image that we do not require, leaving only the region of interest.
- Resized: Finally, this function is optional, but it is required if we use the crop foreground function, because the function that will do the crop will have the output with random dimensions depending on each patient, so if we do not add an operation to give the same dimensions to all patients, our model does not work.

Now, as with any deep learning code, we load the data with its transform before we begin training, two essential functions are used:

- Dataset: This function defines the data along with its transforms, so if training and validation sets are there, we need to create two "dataset" functions, one to combine the training data with its transforms and the other to combine the validation data with its transforms. If the same transforms are applied to the training and validation sets, the same transforms in the parameter 'transform' in the function "dataset" are used.

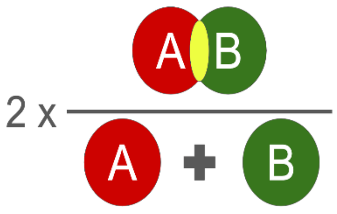
- Data-loader: This is the function that loads the data into RAM with a specific batch-size. It creates another channel to specify the index of the batch size during training. There are two data loaders, one for training and one for validation.

Difference between Dice Coefficient and Dice Loss

The Dice coefficient and Dice loss are both commonly used metrics in medical image segmentation to evaluate the accuracy of segmentation results. The main difference between the two is that the Dice coefficient is a measure of similarity between two sets, while the Dice loss is a measure of dissimilarity or error between two sets.

The Dice coefficient, also known as the Sørensen–Dice coefficient, is a statistical measure of the similarity between two sets of data. In the context of medical image segmentation, the Dice coefficient is used to measure the similarity between the ground truth segmentation mask and the predicted segmentation mask produced by a model. The Dice coefficient ranges from 0 to 1, where a value of 1 indicates perfect agreement between the two sets, and a value of 0 indicates no agreement.

$$\text{Dice} = 2\text{x} \frac{y \cap y_{\text{-pred}}}{y + y_{\text{-pred}}}$$



Dice coefficient

The Dice loss, on the other hand, is a measure of the dissimilarity or error between two sets of data. It is commonly used as a loss function in deep learning models for medical image segmentation, and is defined as 1 minus the Dice coefficient. The Dice loss measures the discrepancy between the predicted segmentation mask and the ground truth segmentation mask, and is used to guide the optimization of the deep learning model during training.

$$\text{Dice Loss} = 1 - \text{Dice}$$

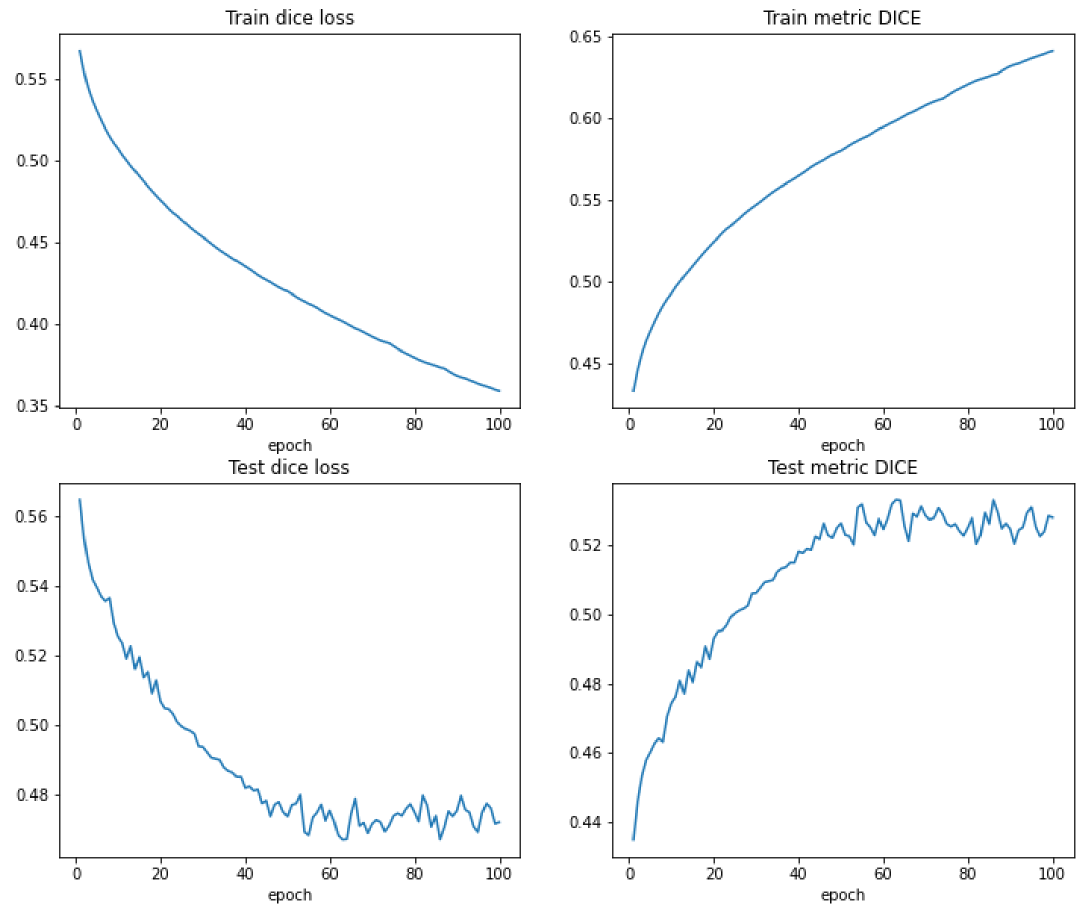
Dice Loss

In summary, the Dice coefficient and Dice loss are both important metrics for evaluating the accuracy of medical image segmentation results. The Dice coefficient measures the similarity between two sets of data, while the Dice loss measures the dissimilarity or error between two sets of data.

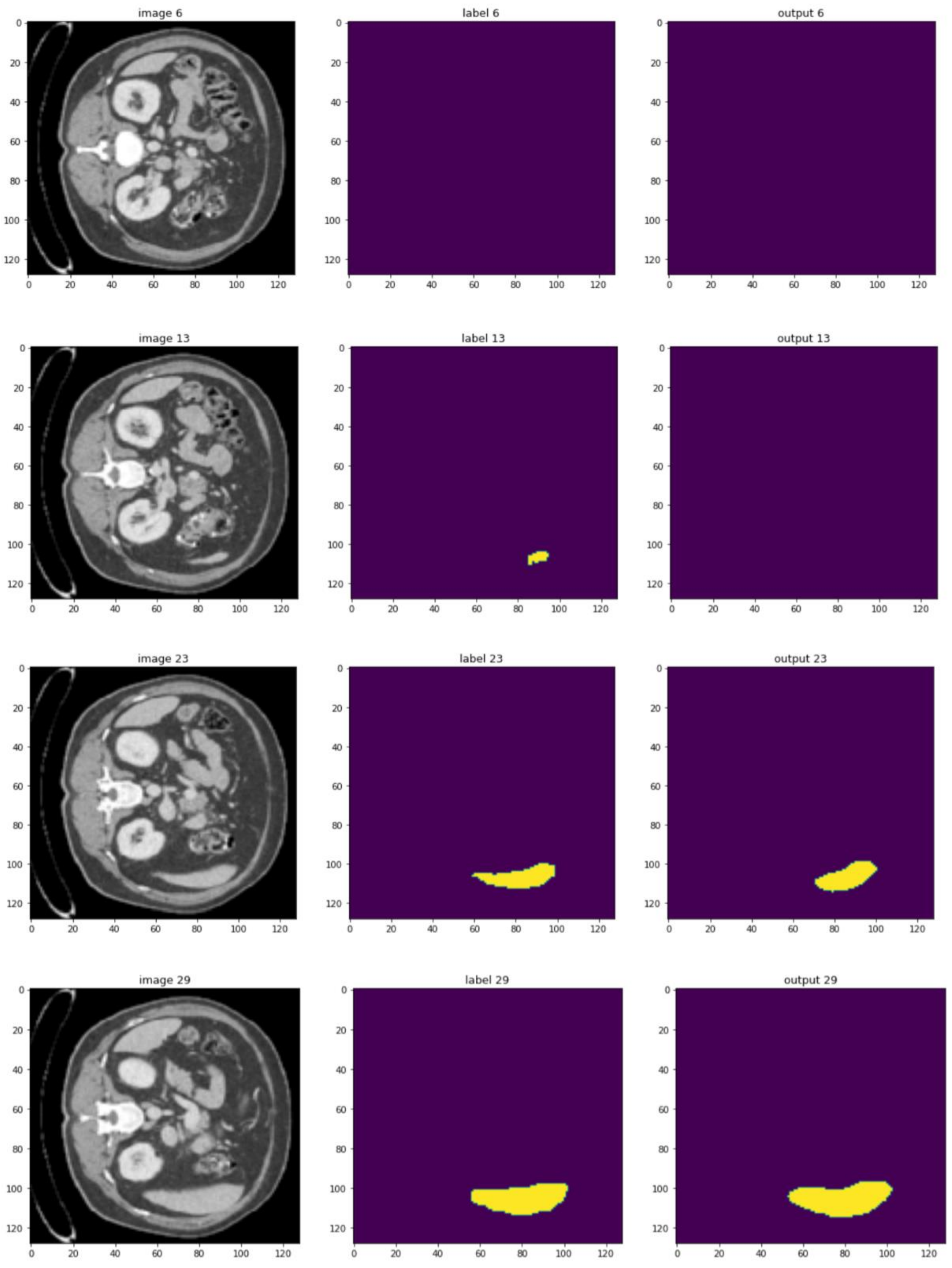
RESULTS

To test the model, first load the U-Net model as you did for training, then there is a function that allows you to load the weights to the created model, and finally you can start passing the test patients to see if the model is segmenting well or not.

These are the metrics for the training and the testing data:



Train and Test metrics



Images **Actual Segmentation** **Predicted Segmentation**

CHALLENGES AND LIMITATIONS

Medical Dataset

First, it is difficult to find medical dataset which is organized in proper format. The medical data format is unconventional, which means it is not like regular images. We need to make many conversions and transformations before training.

Semantic segmentation:

Semantic segmentation is commonly defined as the process of partitioning an image into multiple segments/regions. To this end, one or multiple labels are assigned to every pixel such that pixels with the same label share certain characteristics. Semantic segmentation can therefore also be regarded as pixel-level classification. As in image-classification problems, predicted class probabilities are typically calculated for each pixel deciding on the class affiliation based on a threshold over the class scores.

In semantic segmentation problems, the pixel-level classification is typically followed by a post-processing step, in which connected components are defined as objects, and object boundaries are created accordingly. Semantic segmentation metrics
can roughly be classified into three classes:

1. single-threshold counting metrics or overlap-based

metrics, for measuring the overlap between the reference annotation and the prediction of the algorithm,

2. distance-based metrics, for measuring the distance between object boundaries,
3. problem-specific metrics, measuring, for example, the volume of objects.

Pitfalls Related to Image-Level Classification

Most issues related to classification metrics are related to one of the following properties of the underlying biomedical problem:

- High class imbalance
- Presence of more than two classes
- Unequal importance of classes
- Interdependencies between classes
- Lack of stratification
- Missing prevalence correction

Pitfalls Related to Segmentation

All pitfalls compiled for this work and relevant for semantic or instance segmentation. This section focuses on limitations for semantic segmentation, but some of them are also transferable to other problem categories, as indicated in the table. Limitations of metrics are typically related to the following properties:

- Small size of structures relative to pixel size
- High variability of structure sizes
- Complex shapes of structures
- Importance of structure volume
- Importance of structure center
- Importance of structure boundaries
- Possibility of multiple labels per unit
- High inter-rater variability
- Possibility of outliers in reference annotation
- Possibility of reference or prediction without the target structure
- Preference for over- vs. under segmentation

SYSTEM AND SOFTWARE REQUIREMENTS

3D Slicer

3D Slicer is a free and open-source software package for medical image analysis and visualization. It was originally developed by the Surgical Planning Laboratory at Harvard Medical School, and has since become a widely-used tool for medical imaging research and clinical applications.

3D Slicer allows users to import, visualize, and manipulate medical image data from a variety of sources, including CT and MRI scans. It provides a range of tools for image segmentation, registration, and visualization, which can be used for a variety of applications, including surgical planning, radiation therapy planning, and research.

One of the key features of 3D Slicer is its extensive library of modules, which can be used to perform a variety of image analysis and processing tasks. These modules include tools for volume rendering, image registration, segmentation, and diffusion tensor imaging analysis, among others. Additionally, 3D Slicer has a large and active community of users and developers, who contribute to the development and improvement of the software.

Overall, 3D Slicer is a powerful and flexible tool for medical image analysis and visualization, which can be used for a wide range of applications in both research and clinical settings. Its user-friendly interface, extensive library of modules, and active community make it a valuable resource for anyone working with medical image data.

Google Colab

Google Collaboratory, also known as "Google Colab" or "Colab" for short, is a cloud-based platform that provides free access to a Jupyter notebook environment and allows users to run Python code on Google's servers. Colab is designed to make it easy for researchers, students, and developers to collaborate on data analysis and machine learning projects.

One of the main advantages of Colab is that it provides free access to powerful hardware resources, including CPUs, GPUs, and TPUs, which can be used to accelerate computations for machine learning tasks. Colab also provides access to a variety of libraries and frameworks, including TensorFlow, PyTorch, and scikit-learn, which can be used for a range of data analysis and machine learning tasks.

In addition to providing access to powerful hardware and software resources, Colab also makes it easy to share notebooks and collaborate with others. Notebooks can be saved to Google Drive, and users can collaborate on a single notebook in real-time.

Colab also provides a variety of tools for visualizing data and displaying results, which can help with understanding and interpreting the output of machine learning models.

Overall, Colab is a powerful and flexible platform for data analysis and machine learning, which is particularly useful for researchers, students, and developers who need access to powerful computing resources but may not have access to their own high-performance computing infrastructure. Its ease of use, collaboration features, and access to powerful hardware and software make it a valuable tool for anyone working with data and machine learning.

CUDA Toolkit

Compute Unified Device Architecture Toolkit is a software development platform for building GPU-accelerated applications. Developed by NVIDIA, CUDA is designed to take advantage of the parallel computing capabilities of NVIDIA graphics processing units (GPUs) to accelerate computational tasks and improve performance.

The CUDA Toolkit includes a suite of libraries, tools, and APIs that enable developers to write high-performance parallel code using popular programming languages such as C, C++, and Python. The toolkit provides a range of features, including:

- CUDA Compiler: A compiler that converts CUDA source code into executable code for GPUs.
- CUDA Runtime: A library that provides a set of low-level APIs for managing CUDA devices, executing CUDA kernels, and transferring data between the CPU and GPU.
- CUDA Math Library: A library that provides optimized math functions for common computational tasks.
- CUDA Deep Neural Network Library (cuDNN): A library that provides optimized primitives for deep learning, such as convolution and pooling.
- CUDA Tools: A suite of tools for debugging, profiling, and analyzing CUDA applications, including NVIDIA Visual Profiler and NVIDIA Nsight.

The CUDA Toolkit is widely used for a variety of applications, including scientific computing, data analytics, machine learning, and deep learning. It enables developers to harness the power of GPU-accelerated computing to accelerate performance and improve efficiency, making it a valuable tool for anyone working with large-scale computational tasks.

CONCLUSION

Medical image segmentation is a critical task in many areas of medical imaging, including diagnosis, treatment planning, and disease monitoring. Deep learning models have shown great promise in improving the accuracy and efficiency of medical image segmentation, and there are now a variety of frameworks available for building and training these models, including PyTorch and MONAI.

A successful medical image segmentation project typically involves several key steps, including data preparation, model development, hyperparameter tuning, and model evaluation. The choice of data and the preprocessing pipeline is an important factor in the success of the project, as it determines the quality of the data used for training and testing the model. The development and optimization of the deep learning model is another critical step, where the choice of the model architecture, loss function, optimizer, and learning rate can have a significant impact on the performance of the model.

Finally, the evaluation of the model is an important step in the project, where various metrics such as Dice coefficient and Dice Loss is used to evaluate the accuracy of the model. It is also important to assess the clinical utility of the segmentation results by comparing them to ground truth data and evaluating their impact on downstream clinical tasks.

Overall, medical image segmentation is a complex and challenging task, but with the development of deep learning models and the availability of frameworks like PyTorch and MONAI, there is great potential to improve the accuracy and efficiency of medical imaging and to make significant advances in the diagnosis and treatment of medical conditions.

REFERENCES

1. Shadi AlZu'bi, Mohammed Shehab, Mahmoud Al-Ayyoub, Yaser Jararweh, and Brij Gupta. 2020. Parallel implementation for 3d medical volume fuzzy segmentation. Pattern Recognition Letters 130 (2020), 312–318.
2. Saeid Asgari Taghanaki, Kumar Abhishek, Joseph Paul Cohen, Julien Cohen-Adad, and Ghassan Hamarneh. 2021. Deep semantic segmentation of natural and medical images: a review. Artificial Intelligence Review 54, 1 (2021), 137–178.
3. D Bamira and MH Picard. 2018. Imaging: Echocardiology—Assessment of Cardiac Structure and Function. (2018).
4. Bernice B Brown. 1968. Delphi process: a methodology used for the elicitation of opinions of experts. Technical Report. Rand Corp Santa Monica CA. Yanmin Sun, Andrew K. C. Wong, and Mohamed S. Kamel. 2009. Classification of Imbalanced Data: a Review. Int. J. Pattern Recognit. Artif. Intell. 23 (2009), 687–719.
5. Abdel Aziz Taha and Allan Hanbury. 2015. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. BMC medical imaging 15, 1 (2015), 1–28.
6. Abdel Aziz Taha, Allan Hanbury, and Oscar A Jimenez del Toro. 2014. A formal method for selecting evaluation metrics for image segmentation. In 2014 IEEE international conference on image processing (ICIP). IEEE, 932–936.
7. Ciresan, D.C., Gambardella, L.M., Giusti, A., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: NIPS. pp. 2852{2860 (2012)
8. Dosovitskiy, A., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with convolutional neural networks. In: NIPS (2014)
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
10. Hariharan, B., Arbelez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and _ne-grained localization (2014), arXiv:1411.5752 [cs.CV]
11. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into recti_ers: Surpassing human-level performance on imagenet classi_cation (2015), arXiv:1502.01852 [cs.CV]
12. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Ca_e: Convolutional architecture for fast feature embedding (2014), arXiv:1408.5093 [cs.CV]
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classi_cation with deep convolutional neural networks. In: NIPS. pp. 1106{1114 (2012)
14. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.,Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation 1(4), 541{551 (1989)
15. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation (2014), arXiv:1411.4038 [cs.CV]
16. Maska, M., (...), de Solorzano, C.O.: A benchmark for comparison of cell tracking algorithms. Bioinformatics 30, 1609{1617 (2014)
17. Seyedhosseini, M., Sajjadi, M., Tasdizen, T.: Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks. In: Computer Vision(ICCV), 2013 IEEE International Conference on. pp. 2168{2175 (2013)
18. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014), arXiv:1409.1556 [cs.CV]
19. WWW: Web page of the cell tracking challenge, http://www.codesolorzano.com/celltrackingchallenge/Cell_Tracking_Challenge/Welcome