

A Step-by-Step Tutorial for Radiologists on Building a Neural Network for Segmentation

Masoom Haider, MD, FRCPC FSAR

Professor of Radiology, University of Toronto

Clinician Scientist, Ontario Institute of Cancer Research

Senior Scientist, Lunenfeld Tanenbaum Research Institute

Director Machine Learning and Radiomics Lab

Director Sinai Health System Research MRI

Co-Director – Artificial Intelligence Center

Joint Dept of Medical Imaging, UHN, SHS, WCH

mahaider@radfiler.com



Navigating this e-poster

- If clickable links in this presentation do not work please download a pdf from this link:

<https://www.haiderlab.ca>

You want to “get under the hood” and build and AI Segmentation Model

This is meant to be a reference and a list of resources for those taking the journey to gain competence in coding for AI in medical imaging





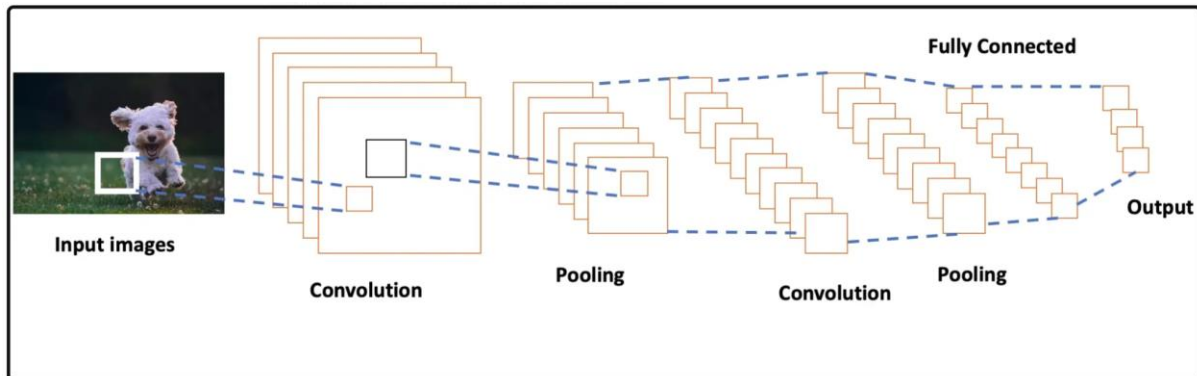
Required Skills

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

2×4 4×3 2×3

- Python coding
- Basic linear algebra proficiency
- A basic knowledge of deep neural network theory/design



Where can I get these skills online?

- Python
 - Python for Beginners <https://www.pythonforbeginners.com/>
 - Learn Python <https://www.learnpython.org/>
 - Real Python <https://realpython.com/>
- Pytorch
 - <https://www.udemy.com/course/deep-learning-with-pytorch-for-medical-image-analysis/>
- Pytorch Lightning
 - <https://www.youtube.com/@PyTorchLightning>

Where can I get these skills online?

- Github
 - <https://docs.github.com/en/get-started>
- AI Fundamentals
 - <https://www.coursera.org/learn/ai-for-medical-diagnosis>
- Radiology-specific AI certificate program by RSNA
 - <https://www.rsna.org/ai-certificate>
 - Annual tutorials/workshops at RSNA
- More education programs for our field are likely forthcoming

Setting up your Development Environment

- Hardware
 - PC – i7/i9 CPU and Nvidia GPU (more VRAM the better)
- Operating System
 - Windows or Linux
 - Linux greater compatibility with open source
 - CUDA (11.7)
 - <https://developer.nvidia.com/cuda-11-7-0-download-archive>
- Open source repository <https://github.com/>



Setting up your Development Environment Software

Cloud options not covered

- Annotation tools
 - 3D Slicer
 - OHIF
 - ITK Snap

<https://www.slicer.org/>

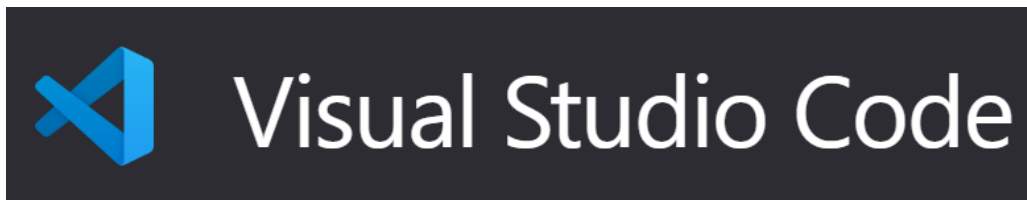
<https://ohif.org/>

<http://www.itksnap.org>



Setting up your Development Environment Software Stack

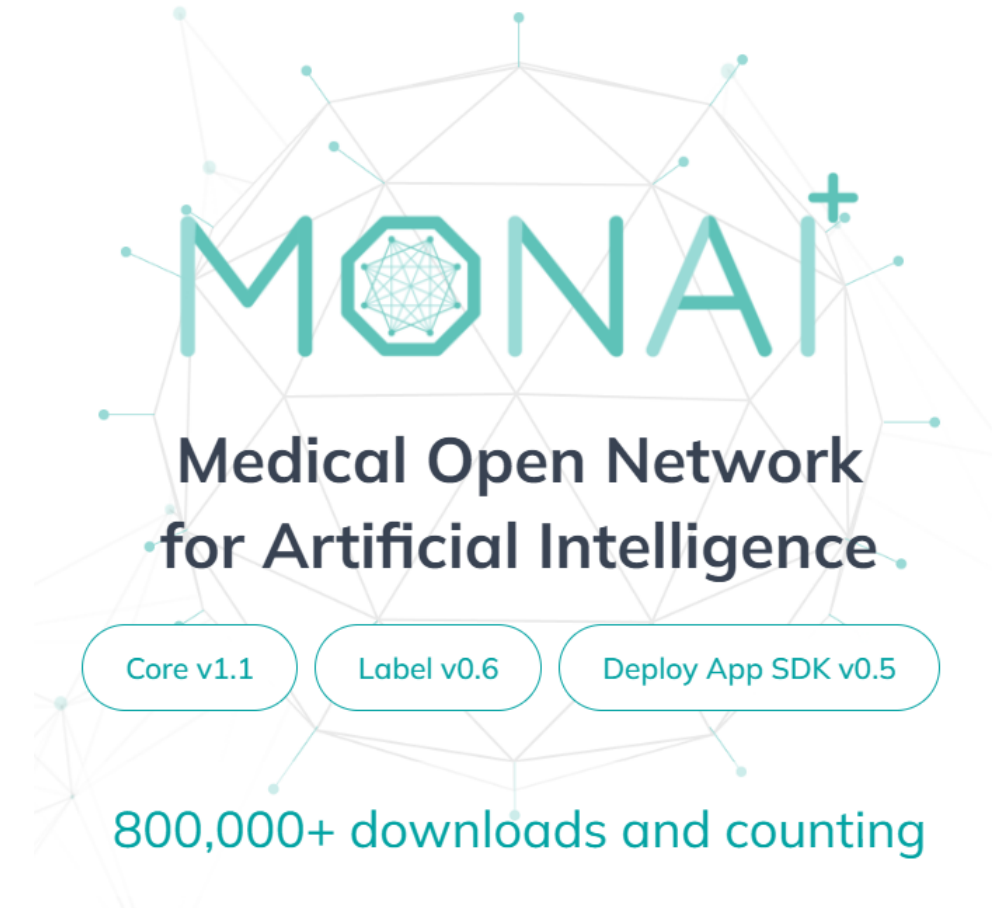
- Environment manager
 - MiniConda <https://docs.conda.io/en/latest/miniconda.html>
- Code editor
 - VSCode <https://code.visualstudio.com/Download>
- Jupyter Notebook for prototyping/documentation
 - Jupyter Notebooks



What is MONAI?

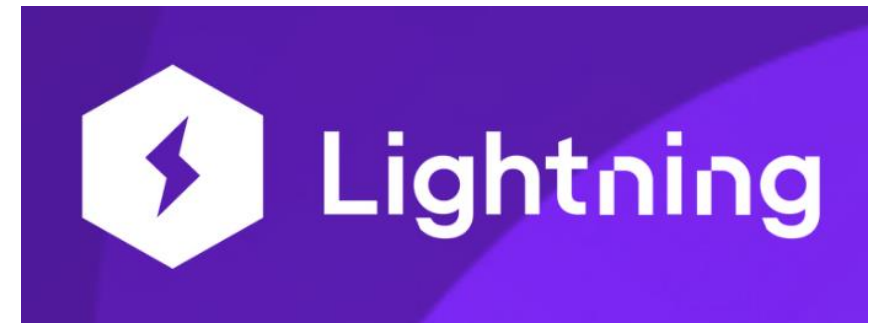
<https://monai.io/>

a set of open-source, freely available collaborative frameworks built for accelerating research and clinical collaboration in Medical Imaging. The goal is to accelerate the pace of innovation and clinical translation by building a robust software framework that benefits nearly every level of medical imaging, deep learning research, and deployment.



Setting up your Development Environment Software Stack

- Setup environment in Anaconda (Miniconda)
 - See Conda cheat sheet
- Install software stack
 - Pytorch
 - Pytorch Lightning
 - MONAI



Windows Batch Script

Create your environment in Anaconda

(.bat file)

SET ENV_NAME=monai10 *the conda environment name you wish to assign*

call conda create -n %ENV_NAME% python=3.8 -y

call conda activate %ENV_NAME%

call conda install jupyter -y

conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia -y

call pip install pytorch-lightning

call pip install monai

call pip install -r requirements-dev.txt *this file from the monai git and installs additional libraries*

call conda install jupyter -y

Watch the MonAI Youtube Channel

- Basics 2023

- <https://www.youtube.com/watch?v=U77UPifZ1Uw&t=1170s>

- Bootcamp 2021

- <https://www.youtube.com/playlist?list=PLtoSVSQ2XzyCobzE6NvwjNpITsQyPUtf>

Project MONAI
@ProjectMONAI
1.76K subscribers

HOME VIDEOS PLAYLISTS COMMUNITY CHANNELS ABOUT

MONAI Overview ▶ Play all
Overview of MONAI

MONAI Label Deep Dive Series ▶ Play all
Learn more about MONAI Label, including installation, extension overviews, annotation methods, training your model, and creating your own custom MONAI Label application.

MONAI Label - Installation with PyPI, Docker, and GitHub
Project MONAI
2K views • 7 months ago

MONAI Label - 3D Slicer Module Overview
Project MONAI
2.4K views • 6 months ago

MONAI Label - Training from Scratch
Project MONAI
3.1K views • 6 months ago

MONAI Label - Scribbles Annotation
Project MONAI
861 views • 6 months ago

Clone the Monai Tutorial Git


- Download the spleen segmentation tutorial code
 - https://github.com/Project-MONAI/tutorials/blob/main/3d_segmentation/spleen_segmentation_3d_lightning.ipynb
- Save your prostate segmentations and source DICOM data in nifti format using one of the segmentation software tools (i.e. itksnap or 3d slicer)
- Customize the `prepare_data` function in the tutorial

Use Case





- Fully automated segmentation of the prostate on T2 MRI
- For use in PSA density calculation
- For use in TRUS/MRI fusion biopsy system
- Radiologist or technologist aid
- Input: Axial T2 images following PiRads standard
- Output: Segmentation mask of prostate

The UNET Model

- MONAI provides basic models
- UNET is particularly good for simple segmentation tasks



[What's New](#)
[Highlights](#)
[API Reference](#)
[Installation Guide](#)
[Development](#)
[More ▾](#)

Ctrl + K

Section Navigation

- Applications
- Auto3dseg
- Federated Learning
- Model Bundle
- Transforms
- Loss functions
- Network architectures**
- Metrics
- Optimizers
- Data
- Engines
- Inference methods
- Event handlers
- Visualizations
- Utilities

`monai.networks.nets.Dynunet`

alias of `DynUNet`

UNet

```
class monai.networks.nets.UNet(spatial_dims, in_channels, out_channels, channels,
strides, kernel_size=3, up_kernel_size=3, num_res_units=0, act='PRELU',
norm='INSTANCE', dropout=0.0, bias=True, adn_ordering='NDA', dimensions=None)
```

Enhanced version of UNet which has residual units implemented with the ResidualUnit [\[source\]](#) class. The residual part uses a convolution to change the input dimensions to match the output dimensions if this is necessary but will use nn.Identity if not. Refer to: https://link.springer.com/chapter/10.1007/978-3-030-12029-0_40.

Each layer of the network has an encode and decode path with a skip connection between them. Data in the encode path is downsampled using strided convolutions (if *strides* is given values greater than 1) and in the decode path upsampled using strided transpose convolutions. These down or up sampling operations occur at the beginning of each block rather than afterwards as is typical in UNet implementations.

To further explain this consider the first example network given below. This network has 3 layers with strides of 2 for each of the middle layers (the last layer is the bottom connection which does not down/up sample). Input data to this network is immediately reduced in the spatial dimensions by a factor of 2 by the first convolution of the residual unit defining the first layer of the encode part. The last layer of the decode part will upsample its input (data from the previous layer concatenated with data from the skip connection) in the first convolution. This ensures the final output of the network has the same shape as the input.

Padding values for the convolutions are chosen to ensure output sizes are even divisors/multiples of the input sizes if the *strides* value for a layer is a factor of the input sizes. A typical case is to use *strides* values of 2 and inputs that are multiples of powers of 2. An input

On this page

- Blocks
- Layers
- Nets**
- AHNet
- DenseNet
- DenseNet121
- DenseNet169
- DenseNet201
- DenseNet264
- EfficientNet
- BlockArgs
- EfficientNetBN
- EfficientNetBNFeatures
- SegResNet
- SegResNetVAE
- ResNet
- SENet
- SENet154
- SEResNet50
- SEResNet101
- SEResNet152
- SEResNext50
- SEResNext101
- HighResNet
- DynUNet
- UNet**
- UNet

SOP for Data Storage

- Structure your data storage using a standard so that you can use similar DataLoader logic for many applications

PROSTATRESEG

<examid>

<segid>_<pulseseq>_<reader initials>.nii.gz

<pulseseq>.nii.gz

```
+---+
AIPR_10000009_19010101_2ec9998ef7f6bb8
|   |   adc.nii.gz
|   |   axt2.nii.gz
|   |   b0.nii.gz
|   |   b1600.nii.gz
|   |   dce.nii.gz
|   |   dwi.nii.gz
|   |   p_axt2_ig.nii.gz
|   |   p_b0_ig.nii.gz
```

Example of data_setup()

```

mSourceDir = r"D:\PSEG" # directory containing data
seriesname='axt2'
segname='p'

def prepare_data(self):
    self filenames = []
    mFolders = [ f.path for f in os.scandir(self.mSourceDir) if f.is_dir() ]
    seriesfname = self.seriesname + ".nii.gz"
    segfname = self.segname + "_" + self.seriesname + "*" + ".nii.gz"
    for mFolder in mFolders:
        # get prostate images and segmentations
        mseriesfname = os.path.join(mFolder, seriesfname)
        mseriesfname = glob.glob(mseriesfname)
        msegfname = os.path.join(mFolder, segfname)
        msegfname = glob.glob(msegfname)
        if len(mseriesfname) > 0 and len(msegfname)>0:
            self.filenames.append({self.fn_keys[0]: mseriesfname[0], self.fn_keys[1]: msegfname[0]})
    # partition files
    test = partition_dataset(self.filenames, ratios=(0.9,0.1), shuffle=True, seed=0, drop_last=False, even_divisible=False)
    self.train_files = test[0]
    self.val_files = test[1]
    # define transforms # put your transforms below
...

```

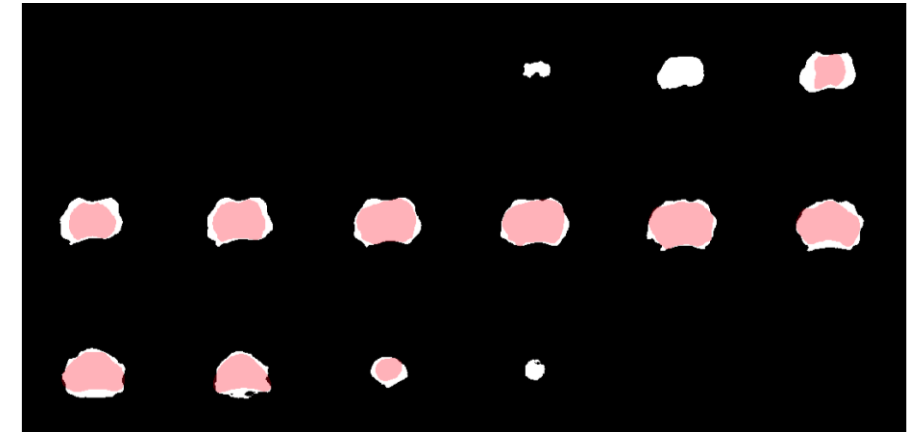
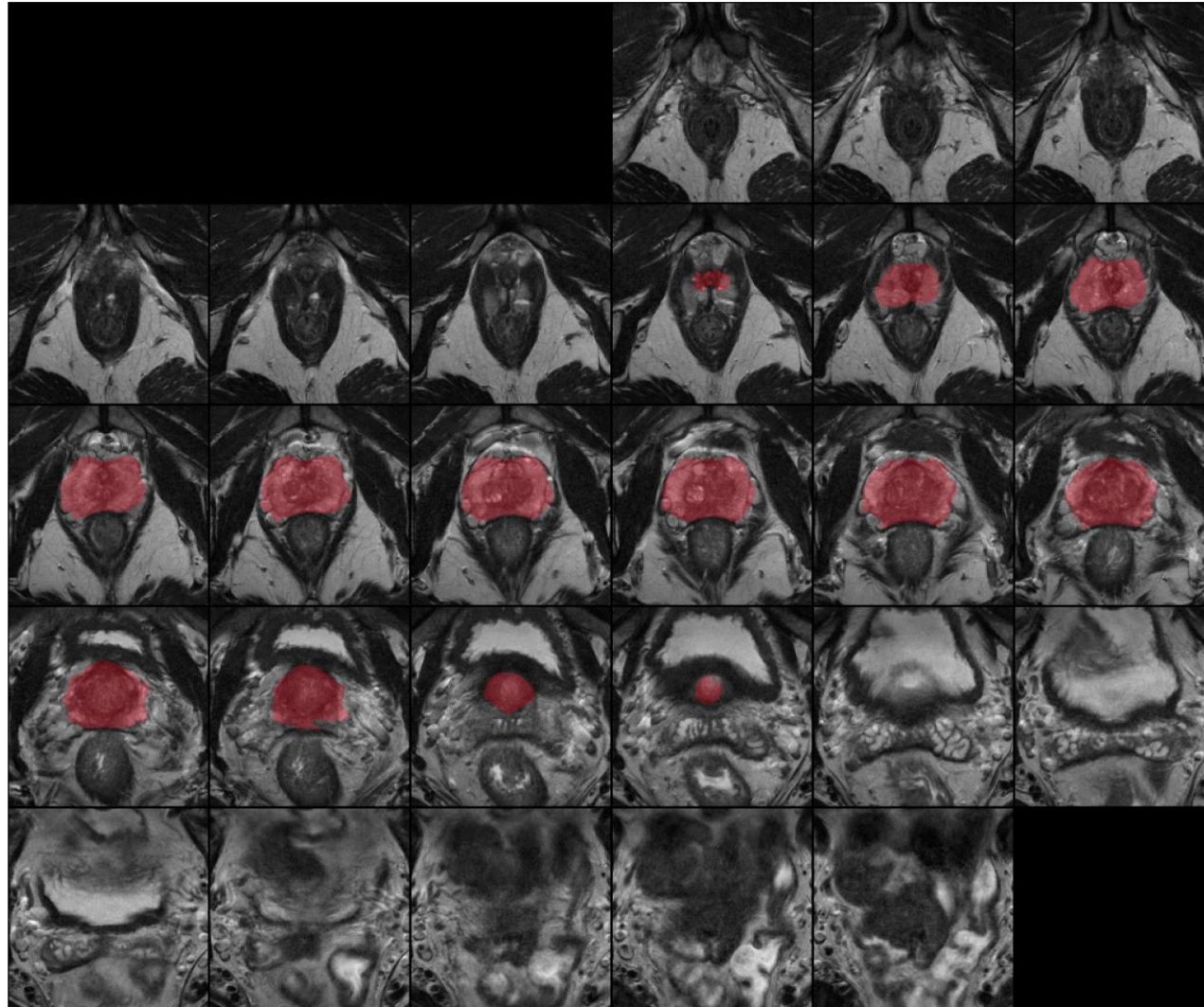
- Run the remaining cells in the notebook
- Training may take a while
- Video from the MONAI workshops are particularly helpful and the associated git tutorial discussion groups are helpful resources

Prediction Code

```
net1.eval()
device = torch.device("cuda:0")
net1.to(device)
dl = net1.val_dataloader()
fn = net1 filenames
with torch.no_grad():
    for i, pred_data in enumerate(dl):
        roi_size = net1.roi_size
        sw_batch_size = 1
        pred_outputs = sliding_window_inference(pred_data["image"].to(device), roi_size, sw_batch_size, net1)
        a = pred_data["image"]
        sw_batch_size=a.shape[0]
        for j in range(sw_batch_size):
            print(f'Working on exam: {j}')
            im = pred_data["image"][j,:,:,:]
            seg = pred_data["label"][j,:,:,:]
            segcalc = torch.argmax(pred_outputs, dim=1).detach().cpu()[j:j+1,:,:,:]
            overlayim = blend_images(segcalc,seg, alpha=0.7, cmap='hsv', rescale_arrays=True)
            matshow3d(overlayim,figsize=(100, 100), frame_dim=-1, show=True, channel_dim=0)
```

Prediction

```
{'image': 'D:\\PSEG\\AIPR_10000036_19010000_2ec9998ef7ff89f\\axt2.nii.gz',  
  'label': 'D:\\PSEG\\AIPR_10000036_19010000_2ec9998ef7ff89f\\p_axt2_ig.nii.gz'}  
Dice : 0.71546304
```



Overlay of radiologist and
predicted segmentation

Summary

- This is an overview of the journey in learning how to get into the world of deep convolutional neural network coding.
- It requires a dedicated learning plan
- Tools continue to improve rapidly and it is expected that things will become easier
- MONAI has greatly accelerated progress for those in the medical imaging arena

m.haider@utoronto.ca
<https://www.haiderlab.ca>



**Lunenfeld-Tanenbaum
Research Institute**
Sinai Health System



**University
Medical Imaging
Toronto**

