Medical/Bio Research Topics Ⅱ: Week 06 (19.10.2023)

Lesion segmentation artificial intelligence models (2): model construction

**(병변 분할 인공지능 모델 개발 연습 (2):
예측 모델 구성)**

Medical Open Network for AI (MONAI)

[https://github.com/Project-MONAI]

- Project MONAI
  - Initiative started initially by NVIDIA and King's College London to establish an inclusive community of AI researchers to develop and exchange best practices for AI in healthcare imaging across academia and enterprise researchers
    - Evolved into a growing consortium currently with 16 different universities and industrial partners
  - Aimed to define, standardize, develop, and exchange best practices for AI in healthcare by unifying the fragmented healthcare AI software field resulted from the disjointed development of several platforms such as NiftyNet [Gibson, et al., 2018], DLTK [Pawlowski, et al., 2017], DeepNeuro [Beers, et al., 2021], NVIDIA Clara [https://www.nvidia.com/en-us/clara/], and Microsoft Project InnerEye [https://www.microsoft.com/en-us/research/project/medical-image-analysis/]

- Released multiple open-source PyTorch-based frameworks for annotating, building, training, deploying, and optimizing AI workflows in healthcare
  - MONAI Core: for training AI models for healthcare imaging
  - MONAI Label: for quickly annotating new datasets
  - MONAI Deploy App ADK: for integrating AI models into clinical workflows
  - MONAI Model Zoo: for sharing a collection of medical imaging models in the MONAI Bundle format

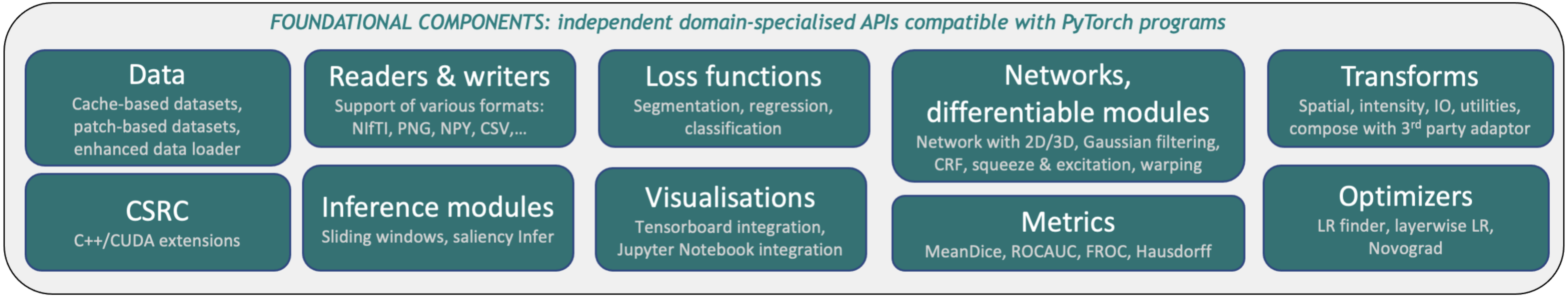
MONAI Core

- PyTorch-based, open-source, freely available, community-supported, and consortium-led framework (suite of libraries, tools, and SDKs) for deep learning in healthcare
  - Extends PyTorch to support medical data, with a particular focus on imaging, video, and other forms of structured data, as part of PyTorch ecosystem (tools, libraries, and more built up around the core PyTorch functionality to support, accelerate, and explore AI development)

- Provides field-specific and domain-optimized functionality that is not often supported by general-purpose frameworks such as Tensorflow and PyTorch
  - Medical images are often stored in complex formats with rich meta-information, with the data volumes being high-dimensional and requiring carefully designed manipulation procedures
  - If healthcare AI models are to be built using a general-purpose framework, significant functionality would need to be developed and tested, thus increasing the length of and the risks associated with the full research and development life-cycle

- Provides wrappers and adaptors that allow popular healthcare AI tools to be used from within MONAI
  - Developed with minimal required dependencies, namely PyTorch and NumPy
- Key design principles:
  - Looks and feels like PyTorch
  - Opt-in and incremental over PyTorch
  - Fully integrates with the PyTorch ecosystem
- Installation
  - $ `pip install monai`

- Modules:
  - **`monai.data`**: datasets, readers/writers, and synthetic data
  - **`monai.losses`**: classes defining loss functions
  - **`monai.networks`**: network definitions, component definitions, and PyTorch-specific utilities
  - **`monai.transforms`**: classes defining data transforms for pre-processing and post-processing
  - **`monai.csrc`** and **`monai.extensions`**: C++/CUDA extensions for MONAI core Python APIs
  - **`monai.visualize`**: utilities for data visualization
  - **`monai.metrics`**: metric tracking and analysis tools
  - **`monai.optimizers`**: classes defining optimizers



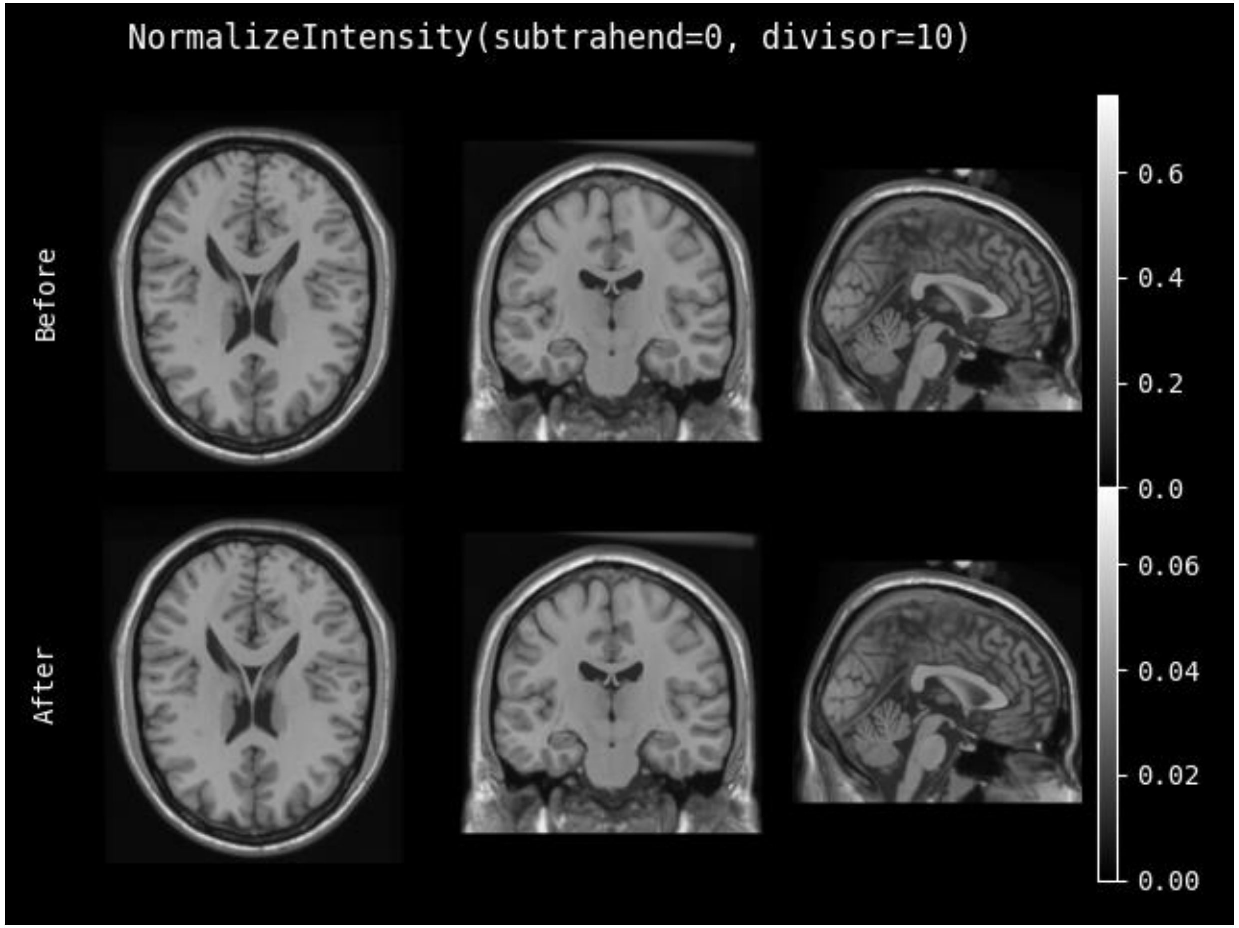
[Cardoso et al., 2021]

MONAI Core modules

- `transforms` module: Vanilla Transforms
  - IO
    - `LoadImage` class: loads the image file or files from the provided path
      - Chooses default readers based on the supported suffixes if not specified
        - » nii, nii.gz: NibabelReader
        - » png, jpg, bmp: PILReader
        - » npz, npy: NumpyReader
        - » nrrd: NrrdReader
        - » DICOM file: ITKReader
    - `SaveImage` class: saves the image (in the form of torch tensor or numpy ndarray) and metadata dictionary into files

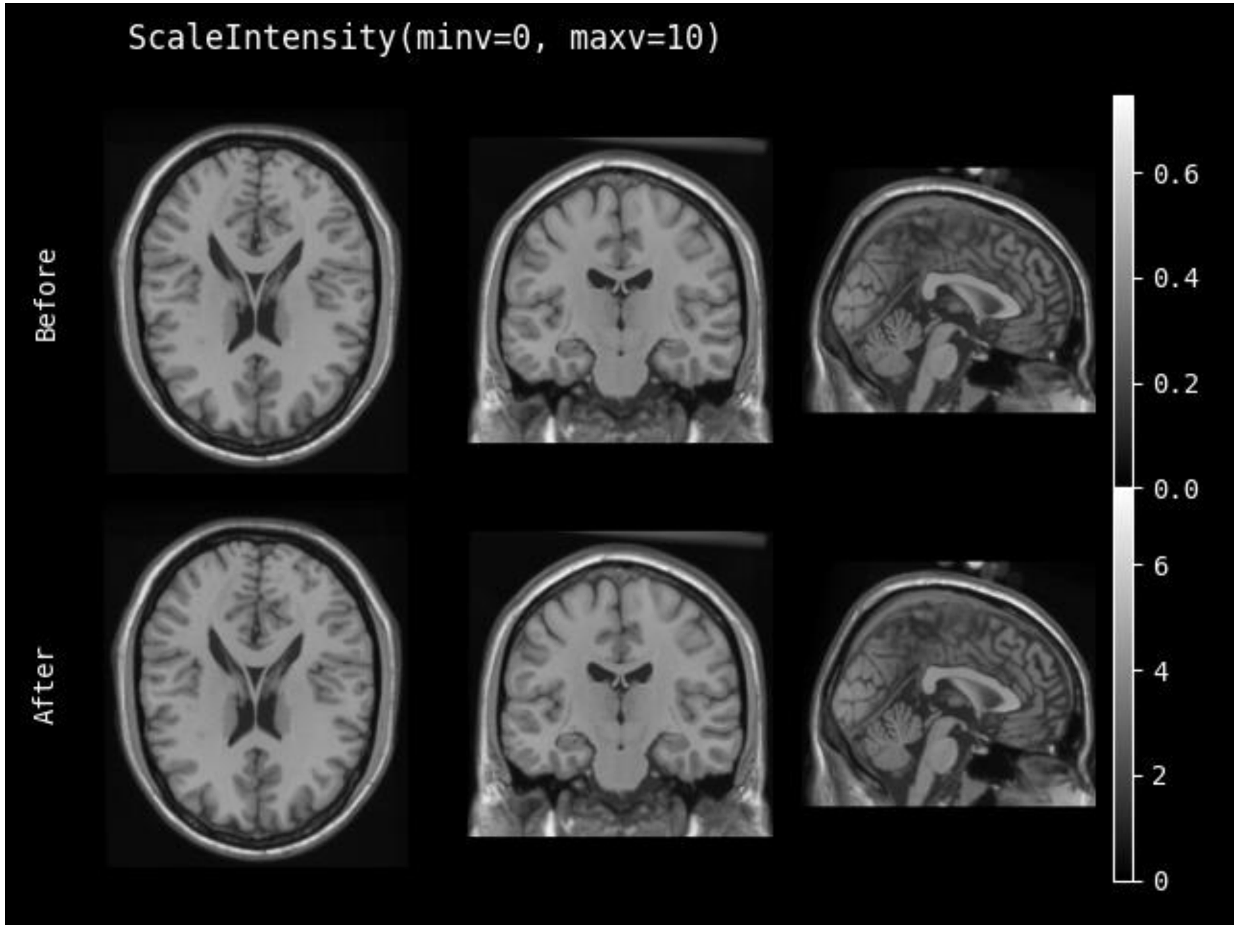
– Intensity

- `NormalizeIntensity` class: normalizes the image based on the mean and standard deviation
- `ScaleIntensity` class: scales the intensity of the image to the given value range
- `RandGaussianNoise` class: adds Gaussian noise to the image
- `GaussianSmooth` class: applies a Gaussian filter to the image
- `MedianSmooth` class: applies a median filter to the image



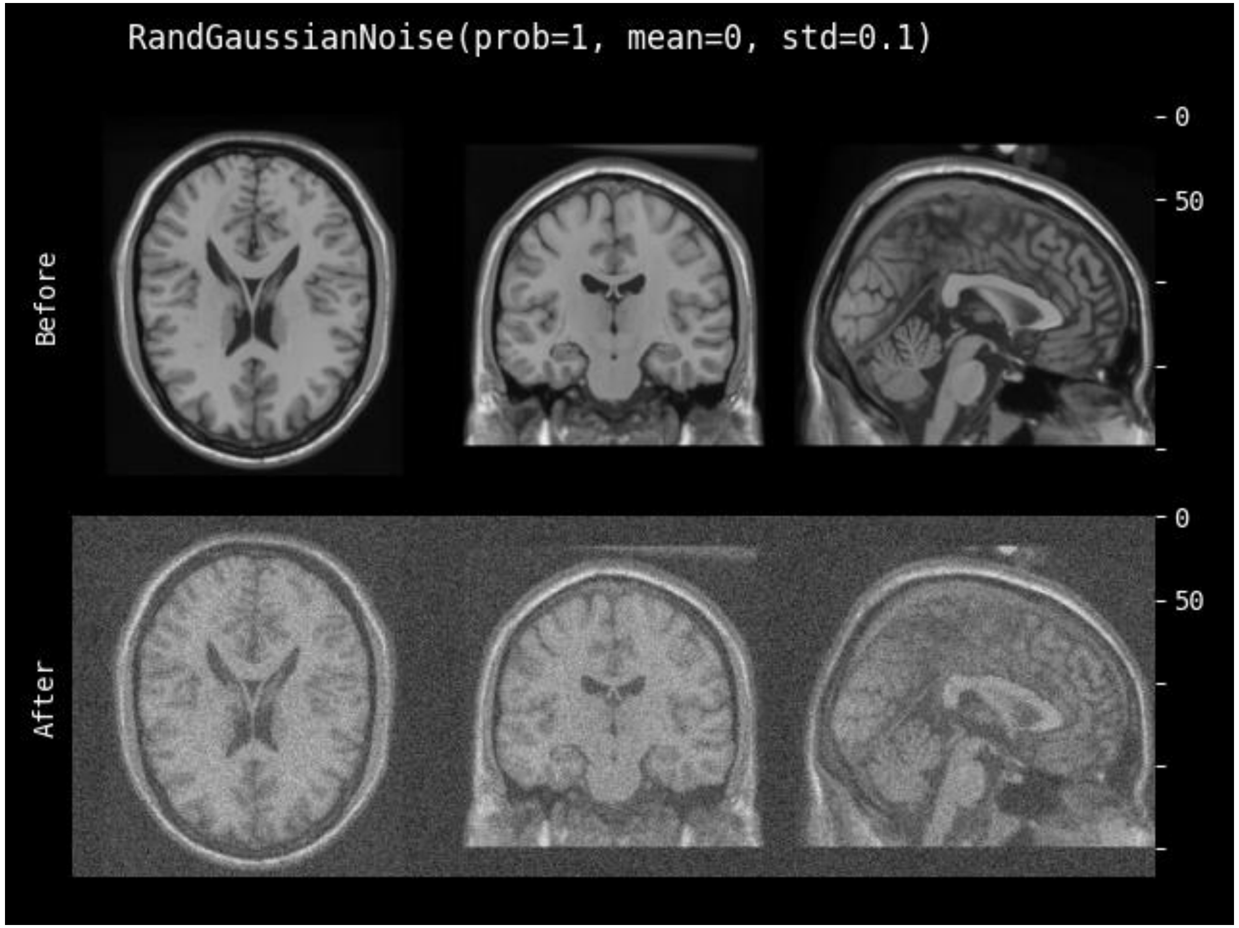
[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.NormalizeIntensity` class



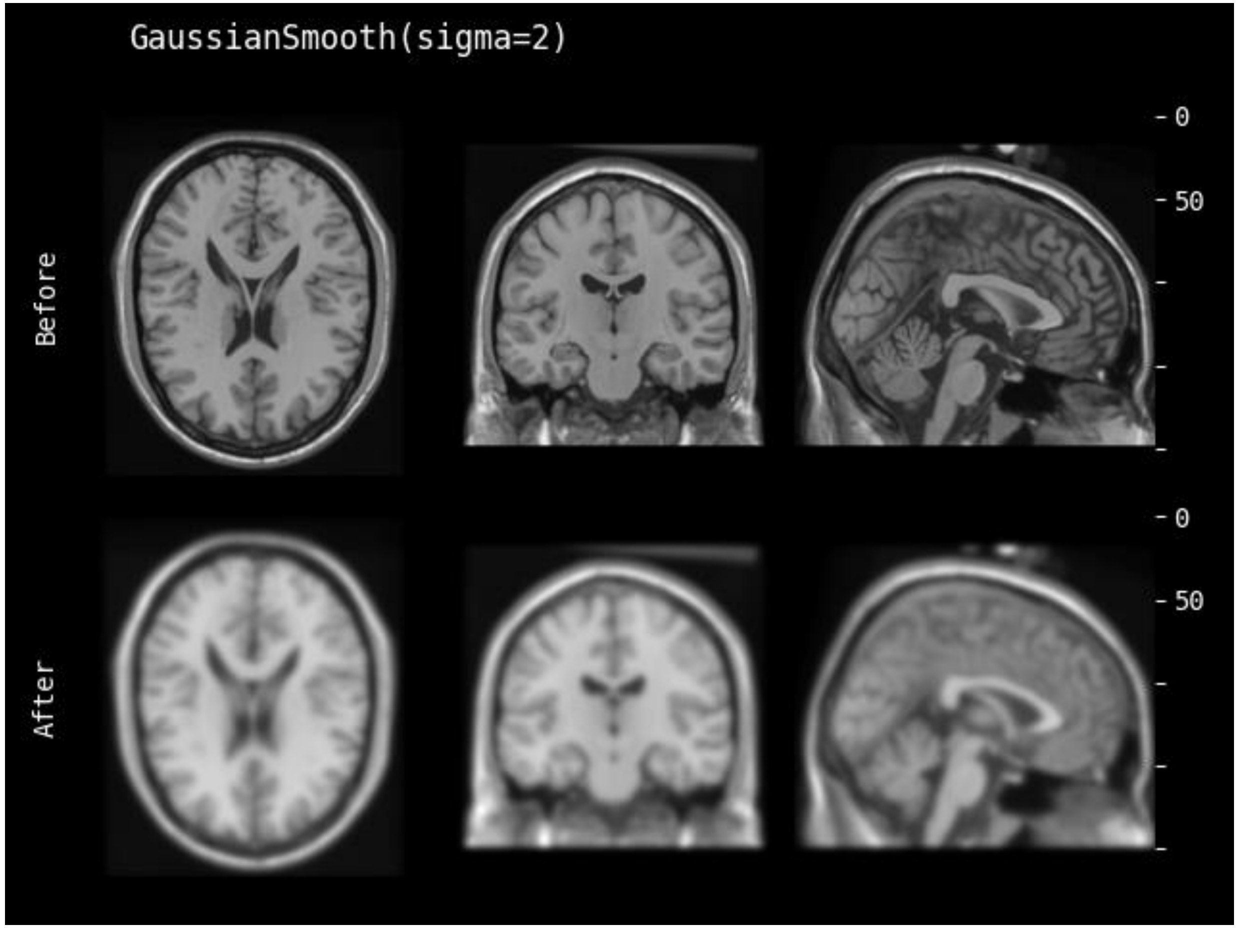
[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.ScaleIntensity` class



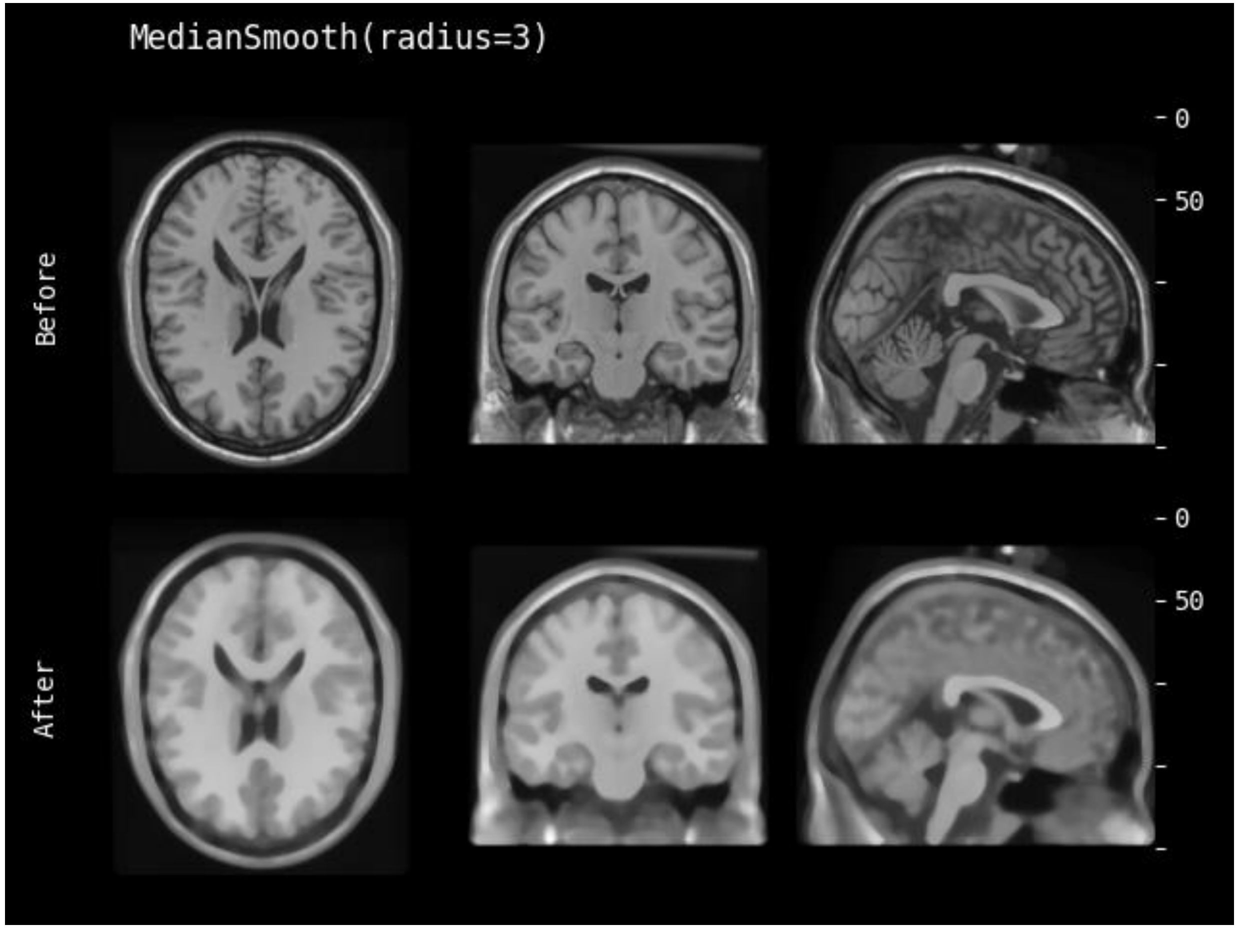
[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.RandGaussianNoise` class



[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.GaussianSmooth` class

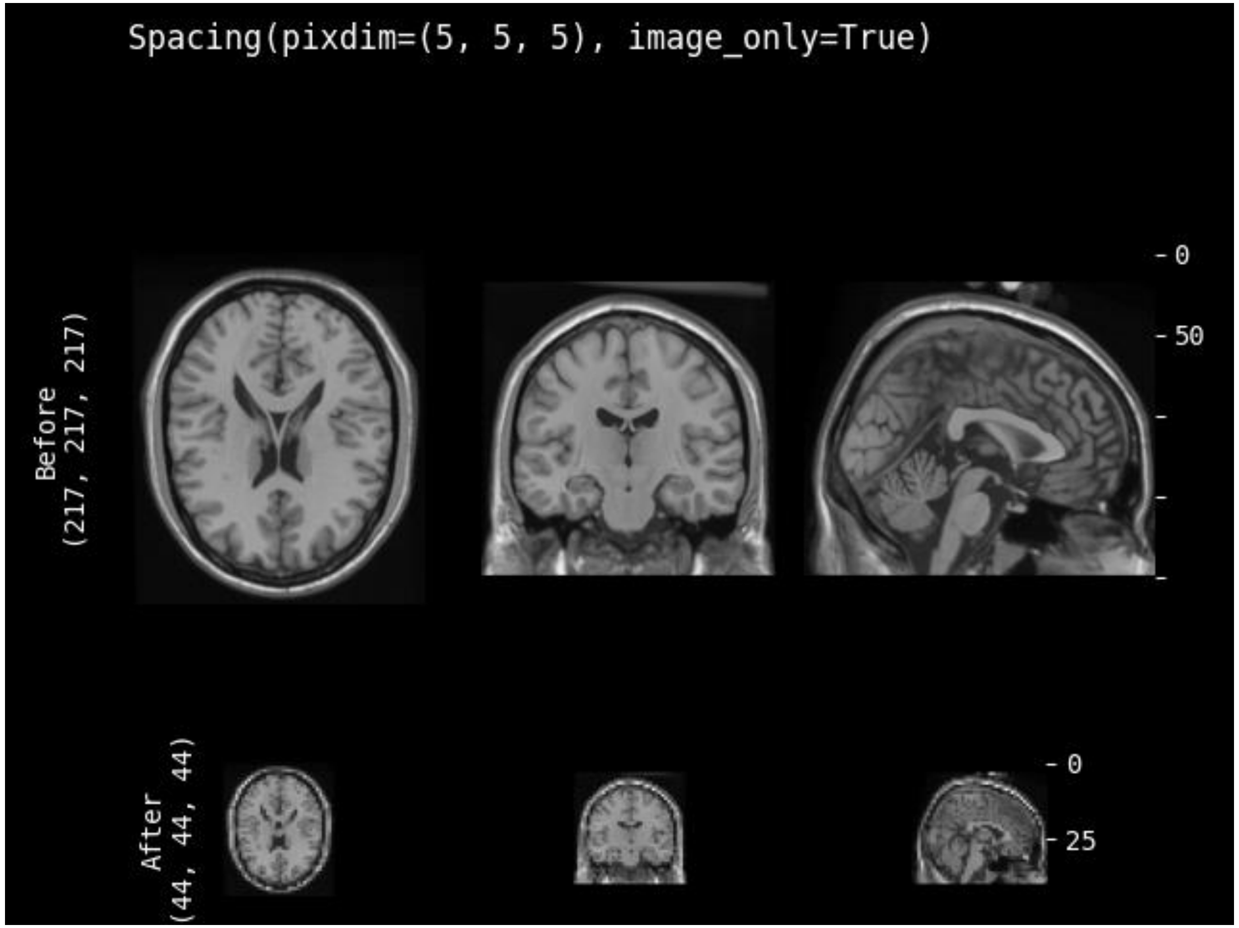


[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.MedianSmooth` class

– Spatial

- `Spacing` class: resamples the image into the specified voxel spacing
- `Resize` class: resizes the image to the specified spatial size (with scaling, not cropping/padding)
- `Orientation` class: changes the image's orientation into the specified orientation code (e.g., 'RAS')
- `Flip` class: reverses the order of elements of the image along the specified spatial axis
- `Rotate` class: rotates the image by the specified angle
- `Zoom` class: zooms the image
- `Affine` class: transforms the image based on the specified affine parameter

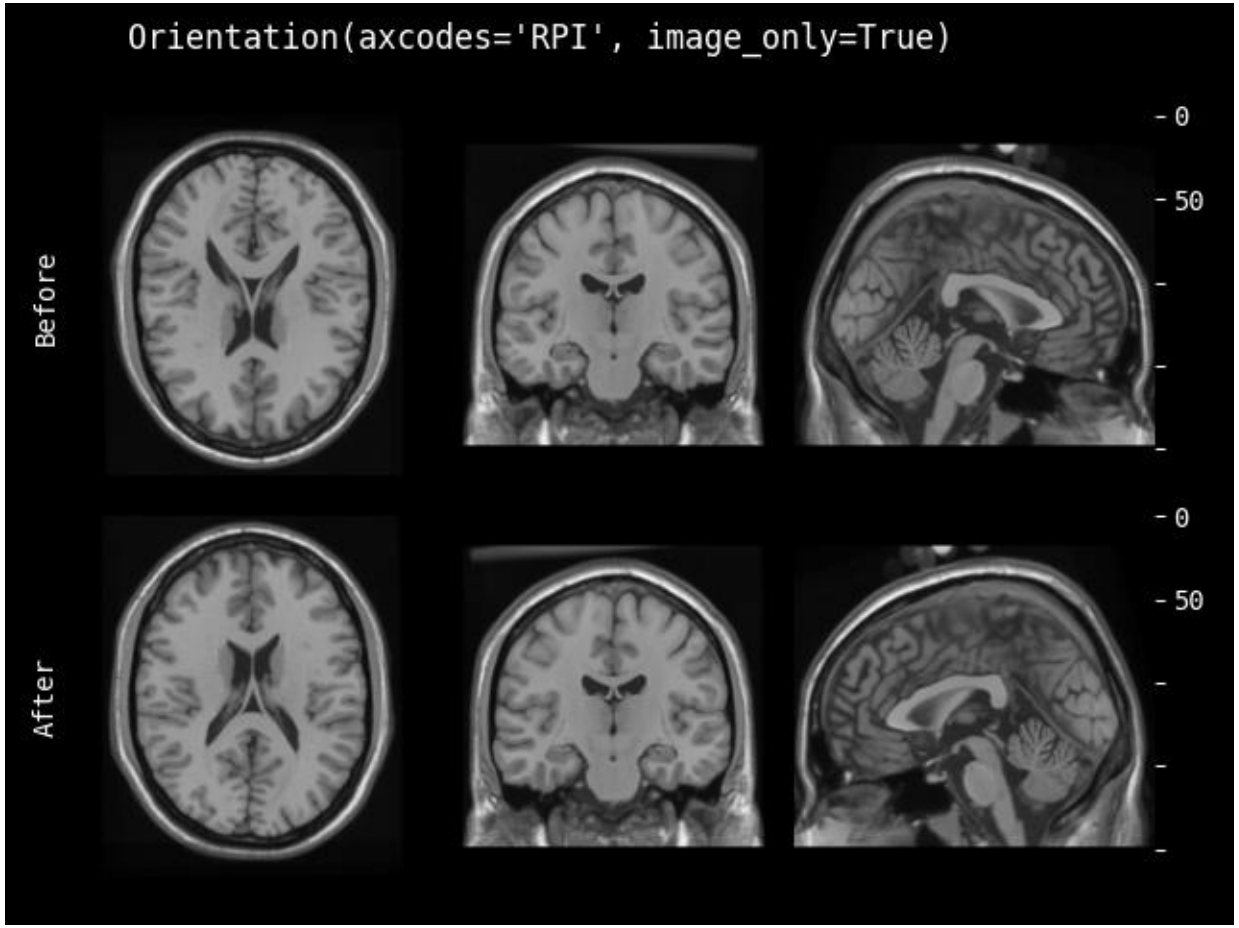


[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.Spacing` class

[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.Resize` class

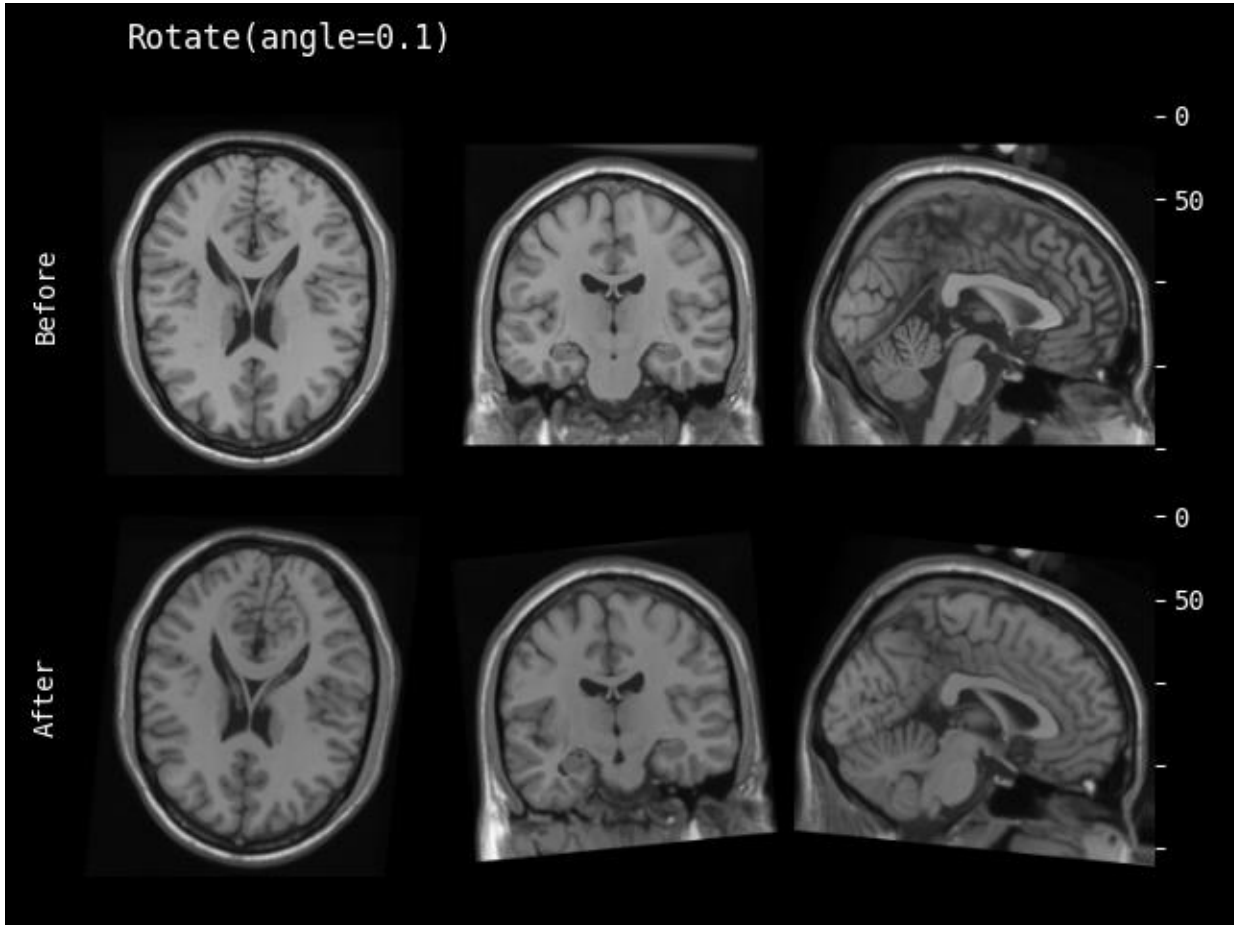


[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.Orientation` class

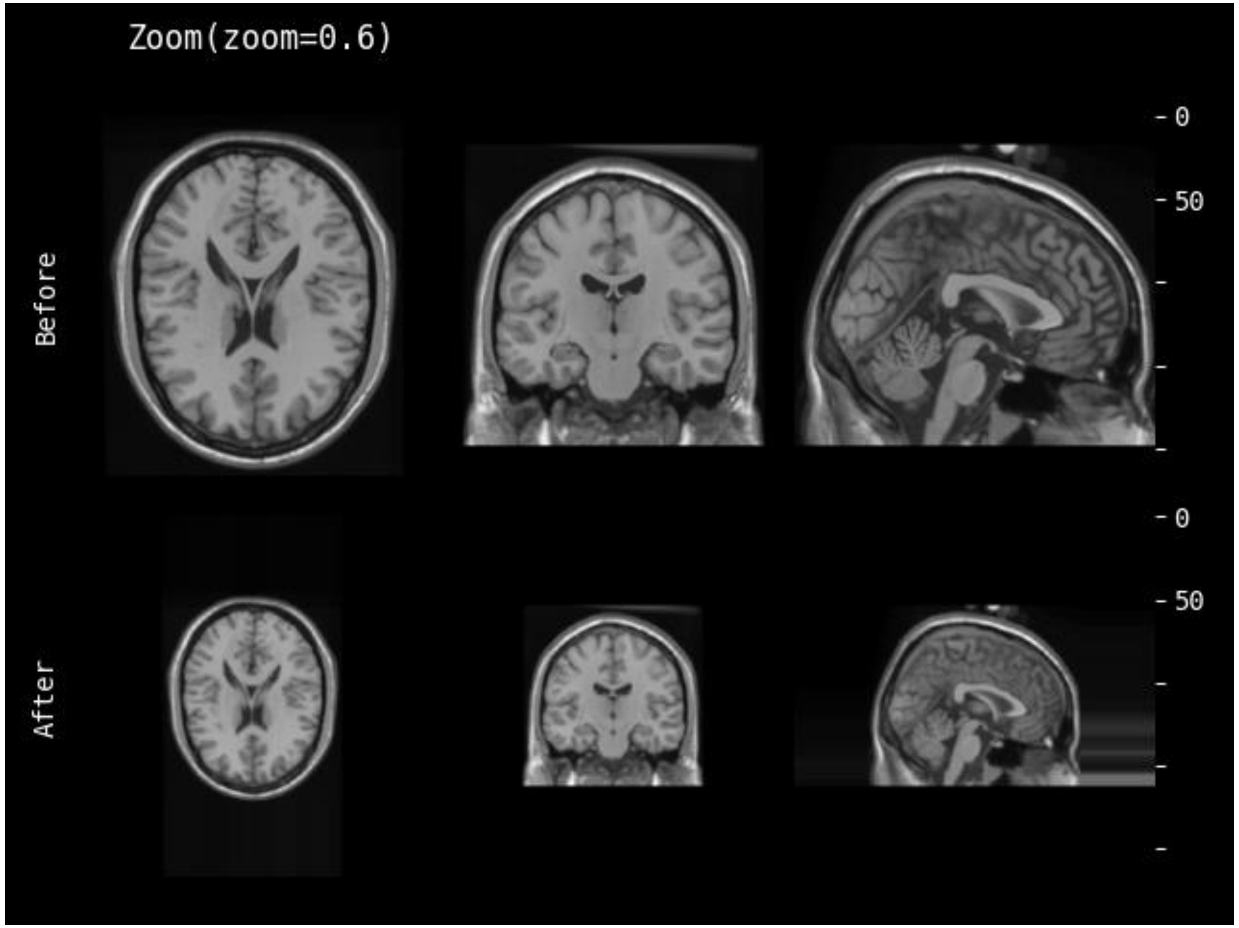
[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.Flip` class



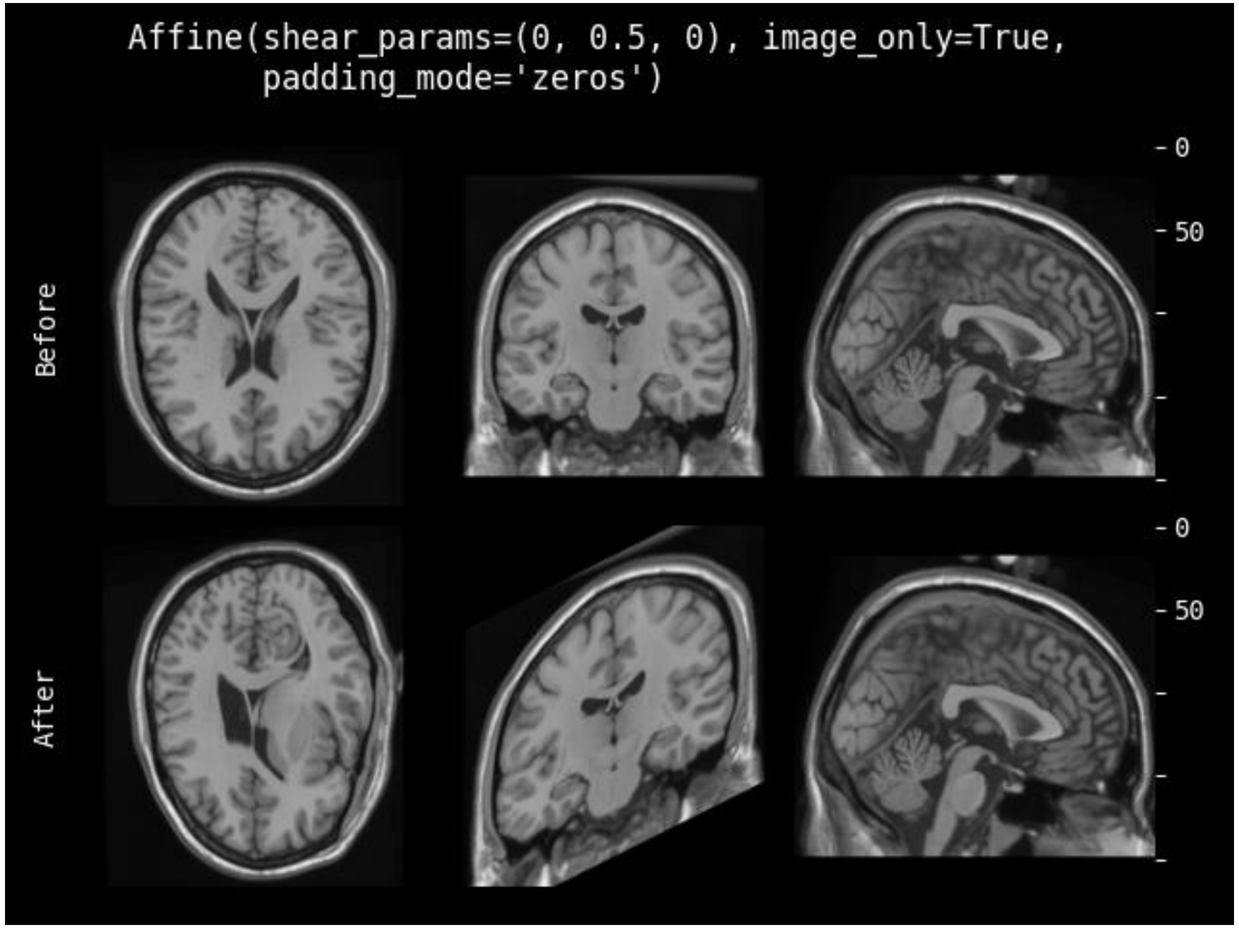
[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.Rotate` class



[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.Zoom` class

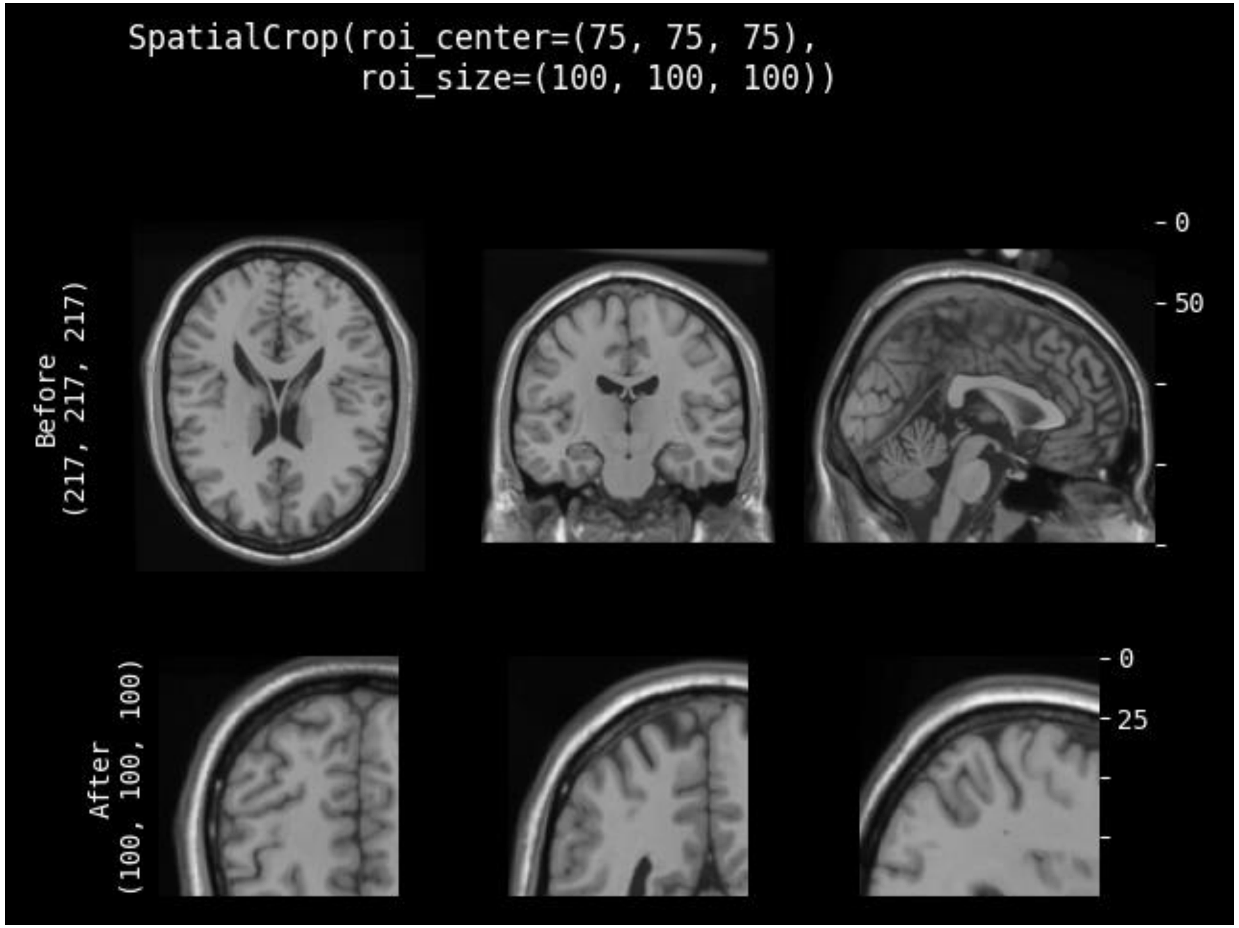


[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.Affine` class

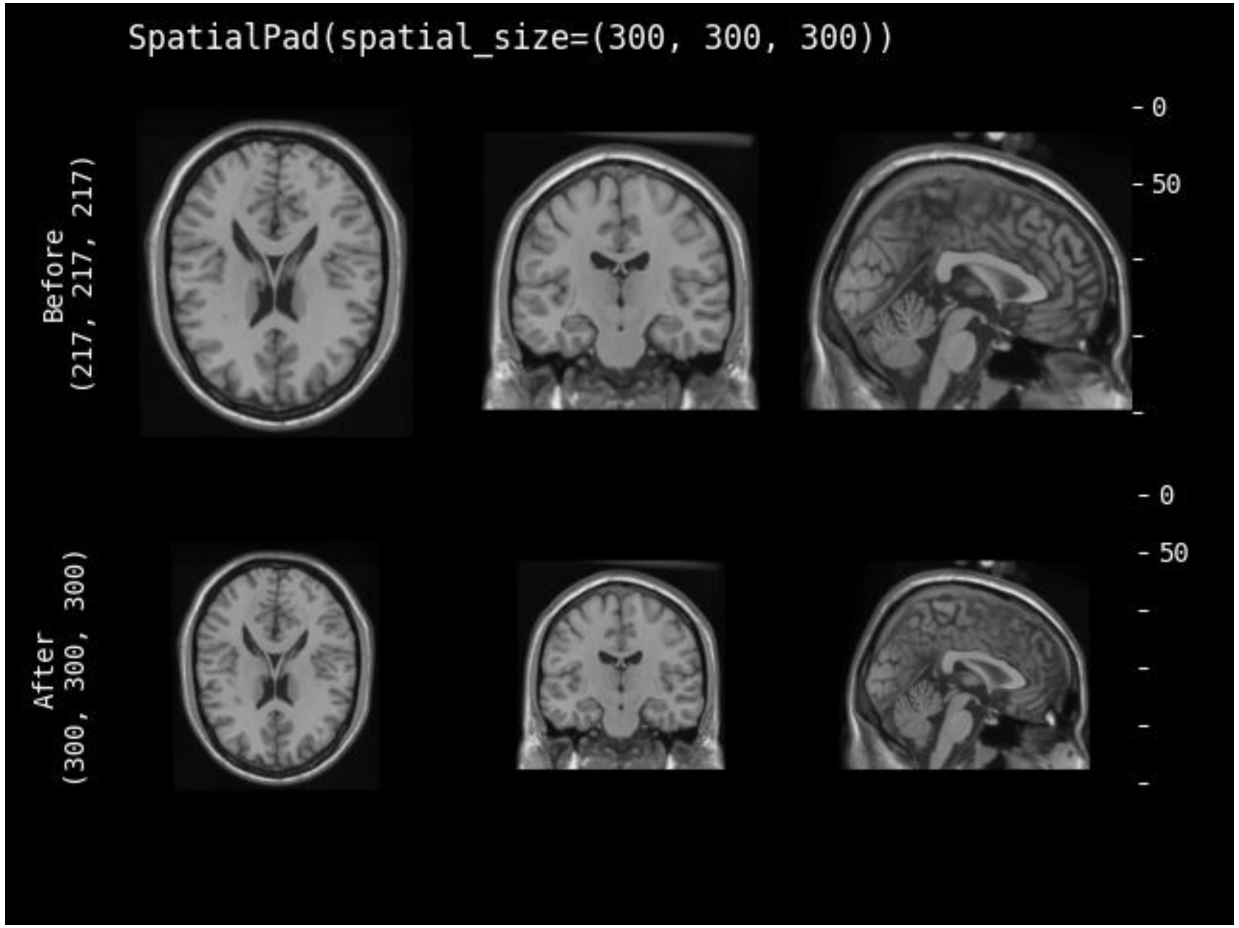
– Crop and Pad

- `SpatialCrop` class: produces a sub-volume region of interest
- `SpatialPaD` class: performs padding to the data



[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.SpatialCrop` class



[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.SpatialPad` class

– Utility

- `EnsureChannelFirst` class: adjusts or adds the channel dimension of the input data to ensure *channel_first* shape
  - A 1-size first dimension is to be added if no dimension is the channel
- `EnsureType` class: ensures the input data to be a PyTorch Tensor or numpy array
- `SqueezeDim` class: squeezes a unitary dimension
- `SplitDim` class: returns a list of length X containing images, given an image of size X along a certain dimension
- `ToTensor` class: converts the input data to a tensor without applying any other transformations
- `ToNumpy` class: converts the input data to a numpy array
- `ToCupy` class: converts the input data to a CuPy array
- `ToDevice` class: moves the PyTorch Tensor to the specified device

- Post-processing
  - `Activations` class: activation operations, typically *Sigmoid* or *Softmax*.
  - `AsDiscrete` class: converts the input tensor/array into discrete values
    - argmax
    - Thresholds the value to a binary value
    - Converts the value to the One-Hot format
    - Rounds the value to the closest integer

[https://docs.monai.io/en/stable/transforms.html]

Application of the `monai.transforms.AsDiscrete` class

- `transforms` module: Generic Interfaces
  - `Compose` class: provides the ability to chain a series of data processing steps together in a sequential manner
    - For example, `c = Compose([Flip(…), Rotate90(…), Zoom(…), RandRotate(…), Resize(…)])`

- `networks` module: Layers
  - `Dropout` class
  - `Act` class
  - `Norm` class
  - `Pad` class
  - `Pool` class
  - `SkipConnection` class
  - `Flatten` class
  - `Reshape` class

- `networks` module: Nets
  - `AHNet` class: based on "Anisotropic hybrid network" [https://arxiv.org/abs/1711.08580]
  - `DenseNet` class: based on "Densely connected convolutional networks" [https://arxiv.org/abs/1608.06993]
  - `EfficientNet` class: based on "Rethinking model scaling for convolutional neural networks" [https://arxiv.org/abs/1905.11946]
  - `SENet` class: based on "Squeeze-and-excitation networks" [https://arxiv.org/abs/1709.01507]
  - `BasicUNet` class: based on "U-Net – deep learning for cell counting, detection, and morphometry" [http://dx.doi.org/10.1038/s41592-018-0261-2]
  - `UNet` class: based on "Left-ventricle quantification using residual U-Net" [https://link.springer.com/chapter/10.1007/978-3-030-12029-0_40]

- `AttentionUNet` class: based on "Attention U-Net: learning where to look for the pancreas" [https://arxiv.org/abs/1804.03999]
- `DynUNet` class: based on "nnU-Net: self-adapting framework for U-Net-based medical image segmentation" [https://arxiv.org/abs/1809.10486]
- `VNet` class: based on "Fully convolutional neural networks for volumetric medical image segmentation" [https://arxiv.org/abs/1606.04797]
- `ResNet` class: based on "Deep residual learning for image recognition" [https://arxiv.org/abs/1512.03385] and "Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet?" [https://arxiv.org/abs/1711.09577]
- `SegResNet` class, `SegResNetVAE` class : based on "3D MRI brain tumor segmentation using autoencoder regularization" [https://arxiv.org/abs/1810.11654]

- `VarAutoEncoder` class: based on "Auto-encoding variational Bayes" [https://arxiv.org/abs/1312.6114]
- `GlobalNet` class: based on "Label-driven weakly-supervised learning for multimodal deformable image registration" [https://arxiv.org/abs/1711.01666]
- `LocalNet` class: based on "Weakly-supervised convolutional neural networks for multimodal image registration" [https://doi.org/10.1016/j.media.2018.07.002] and "Label-driven weakly-supervised learning for multimodal deformable image registration" [https://arxiv.org/abs/1711.01666]
- `ViT` class: based on "An image is worth 16x16 words: transformers for image recognition at scale" [https://arxiv.org/abs/2010.11929]
- `FullyConnectedNet` class

- `networks` module: Utilities
  - `convert_to_onnx`: converts the model into ONNX model
  - `convert_to_torchscript`: converts the model into a TorchScript model

- `losses` module: Segmentation Losses
  - `DiceLoss` class: computes the average Dice loss between two tensors
  - `TverskyLoss` class: computes the Tversky loss defined in "Tversky loss function for image segmentation using 3D fully convolutional deep networks" [https://arxiv.org/abs/1706.05721]
  - `ContrastiveLoss` class: computes the contrastive loss defined in "A simple framework for contrastive learning of visual representations" [http://proceedings.mlr.press/v119/chen20j.html]
  - `HausdorffDTLoss` class: computes the channel-wise binary Hausdorff loss defined in "Reducing the Hausdorff distance in medical image segmentation with convolutional neural networks" [https://arxiv.org/abs/1904.10030]

- `losses` module: Registration Losses
  - `BendingEnergyLoss` class: computes the bending energy based on the second-order differentiation of predictions
  - `LocalNormalizedCrossCorrelationLoss` class: computes local squared zero-normalized cross-correlation
  - `GlobalMutualInformationLoss` class: computes the differentiable global mutual information loss

- `losses` module: Reconstruction Losses
  - `SSIMLoss` class: computes the loss function based on the structural similarity index measure (SSIM) metric
  - `PatchAdversarialLoss` class: computes the adversarial loss on a patch discriminator or a multi-scale patch discriminator
  - `PerceptualLoss` class: computes the perceptual loss using features from pretrained deep neural networks
  - `JukeboxLoss` class: computes the spectral component based on the magnitude of fast Fourier transform (FFT)

- `metrics` module
  - `DiceMetric` class, `DiceHelper` class: compute the average Dice score for a set of pairs of predictions and ground truths
  - `compute_iou`: computes the intersection over union (IoU) score metric from a batch of predictions
  - `ROCAUCMetric` class, `compute_roc_auc`: compute the area under the receiver operating characteristic curve (ROC AUC)
  - `ConfusionMatrixMetric` class, `get_confusion_matrix`, `compute_confusion_matrix_metric`: compute confusion matrix-related metrics

- `SurfaceDistanceMetric` class, `compute_average_surface_distance`: compute the average surface distance
- `RMSEMetric` class: computes the root mean squared error
- `MSEMetric` class: computes the mean squared error
- `MAEMetric` class: computes the mean absolute error
- `PSNRMetric` class: computes the peak signal to noise ratio
- `regression.SSIMMetric` class: computes the SSIM

- Practical applications: supervised segmentation training workflow
  - Training a segmentation network with ground truth annotated data
    - The network is tasked with predicting the segmentation image from the input image, and this is compared using a loss such as Dice against the known actual segmentation
  - Typical PyTorch training workflow
    - Training loop
      - Feeds data into the network and optimizes its parameters
    - Evaluation loop
      - Uses validation data in conjunction with a metric to assess training progress

PyTorch Training Loop

```
for epoch in range(max_epochs):
    network.train()
    for inputs, labels in train_loader:
        optimizer.zero_grad()
        outputs = network(inputs)
        loss = loss_function(outputs, labels)
        loss.backward()
        optimizer.step()

    network.eval()
    with torch.no_grad():
        for val_inputs, val_labels in val_loader:
            val_outputs = network(val_inputs)
            metric(y_pred=val_outputs, y=val_labels)

        metric = metric.aggregate().item()
        print("Validation result:", metric)
```

[Cardoso et al., 2021]

PyTorch training workflow

- MONAI training workflow using types inherited from PyTorch-Ignite (high-level library to help with training and evaluating networks in PyTorch) [https://pytorch.org/ignite/]
  - Types that encapsulate the training process offer ease of use at the expense of some flexibility but represent a significant acceleration of development
  - Aimed to simplify and regularize how workflows are created to make the process quicker, easier, and more reproducible

MONAI Training Loop

```
evaluator = SupervisedEvaluator(
    val_data_loader=val_loader,
    network=network,
    key_val_metric={ "metric": metric },
    ...
)

trainer = SupervisedTrainer(
    max_epochs=num_epochs,
    train_data_loader=train_loader,
    network=network,
    optimizer=optimizer,
    loss_function=loss_function,
    train_handlers=[ValidationHandler(1,evaluator)],
    ...
)

trainer.run()  # do the training run for 10 epochs
```

[Cardoso et al., 2021]

MONAI training workflow

Network Architectures for Lesion Segmentation

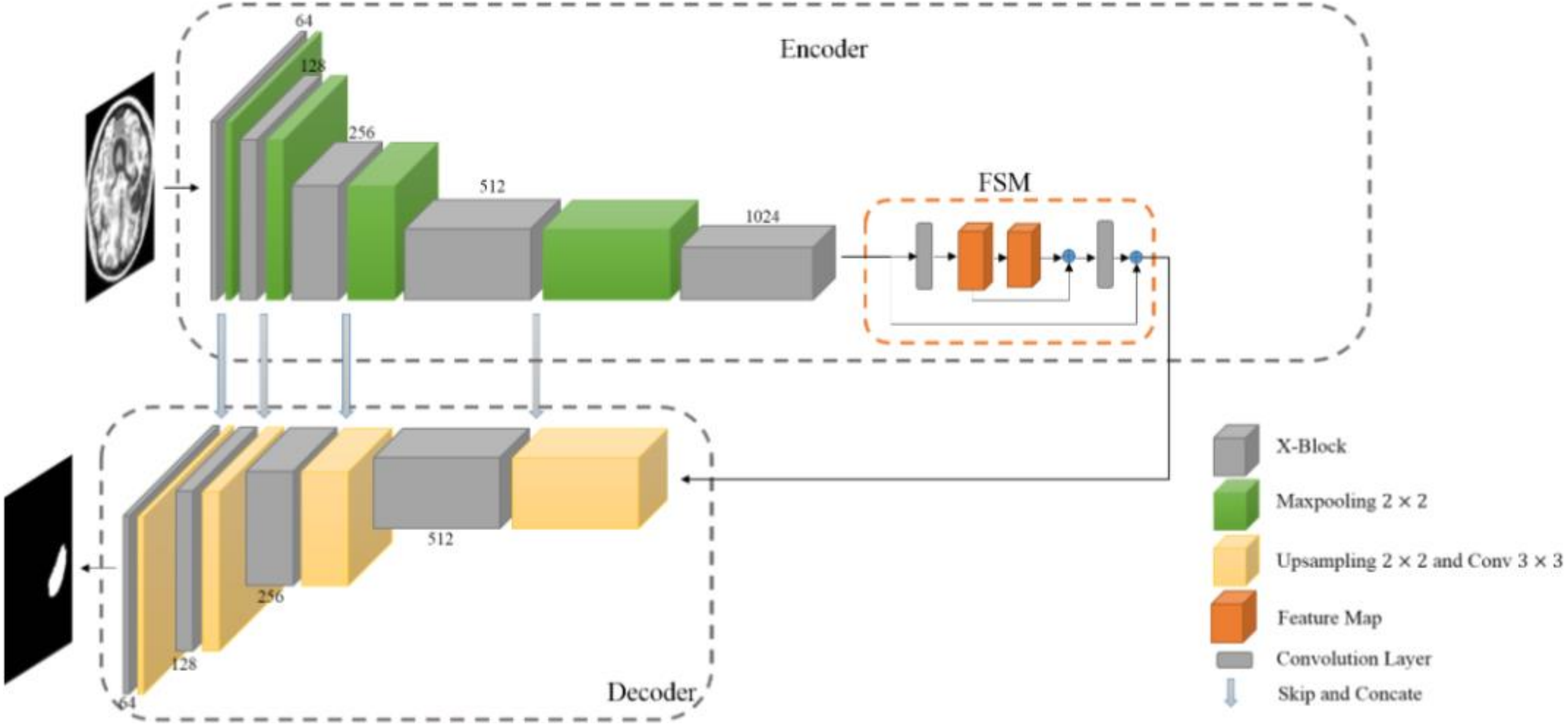
- U-Net
  - Chen et al., 2018
    - Explored the feasibility of using auto-encoder-based deep generative models, such as variational and adversarial auto-encoders, for abnormality detection in medical imaging
    - Reported the Dice similarity coefficient (DSC) of 0.50
      - ATLAS v1.2 dataset
      - Slice input (128×128 or 256×256)
      - U-Net

Models	Latent variables z	BraTS-T2w (whole tumor) AUC	BraTS-T2w (whole tumor) mDSC	ATLAS-T1w AUC	ATLAS-T1w mDSC
mean	-	0,65	0.20	0.46	0.02
AE	256	0.63	0.41	0.49	0.03
DAE (σ=0.5)	256	0.59	0.29	0.41	0.06
VAE-128	(2,2,64)	0.69	0.42	0.64	0.08
VAE-BBB-128	(2,2,64)	0.69	0.40	0.67	0.05
VAE-256	(4,4,64)	0.67	0.40	0.66	0.08
AAE-128	(2,2,64)	0.70	0.41	0.63	0.06
AAE-256	(4,4,64)	0.67	0.38	0.60	0.04
α-GAN-128	128	0.66	0.35	0.60	0.05
α-GAN-256	256	0.67	0.37	0.60	0.04
GMM (λ_{out}=0.01)	-	0.80	0.22	0.78	0.17
GMM (λ_{out}=0.001)	-	0.79	0.21	0.77	0.17
U-net (supervised)	-	-	0.80	-	0.50

[Chen et al., 2018]

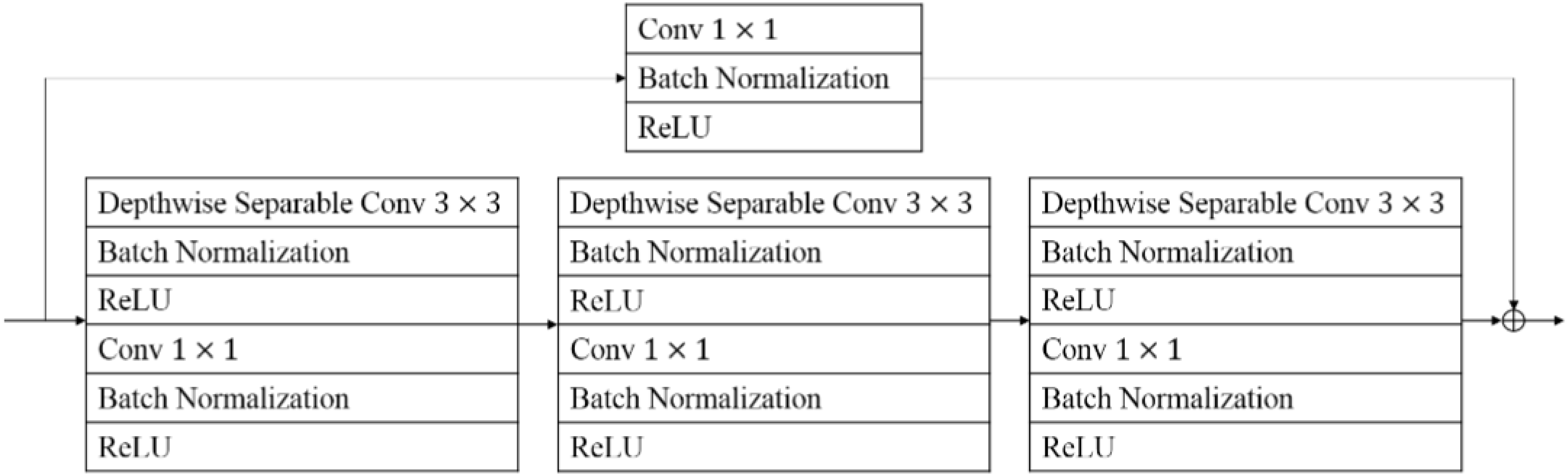
Deep generative models vs. Gaussian mixture model (GMM) and U-Net

- X-Net
  - Qi et al., 2019
    - Proposed a depth-wise separable convolution-based X-Net
    - Employed a feature similarity module (FSM) to capture long-range spatial contextual information
    - Reported the DSC of 0.4867
      - ATLAS v1.2 dataset
      - Slice input (224 × 192)
      - X-Net with an FSM
    - https://github.com/Andrewsher/X-Net



[Qi et al., 2019]

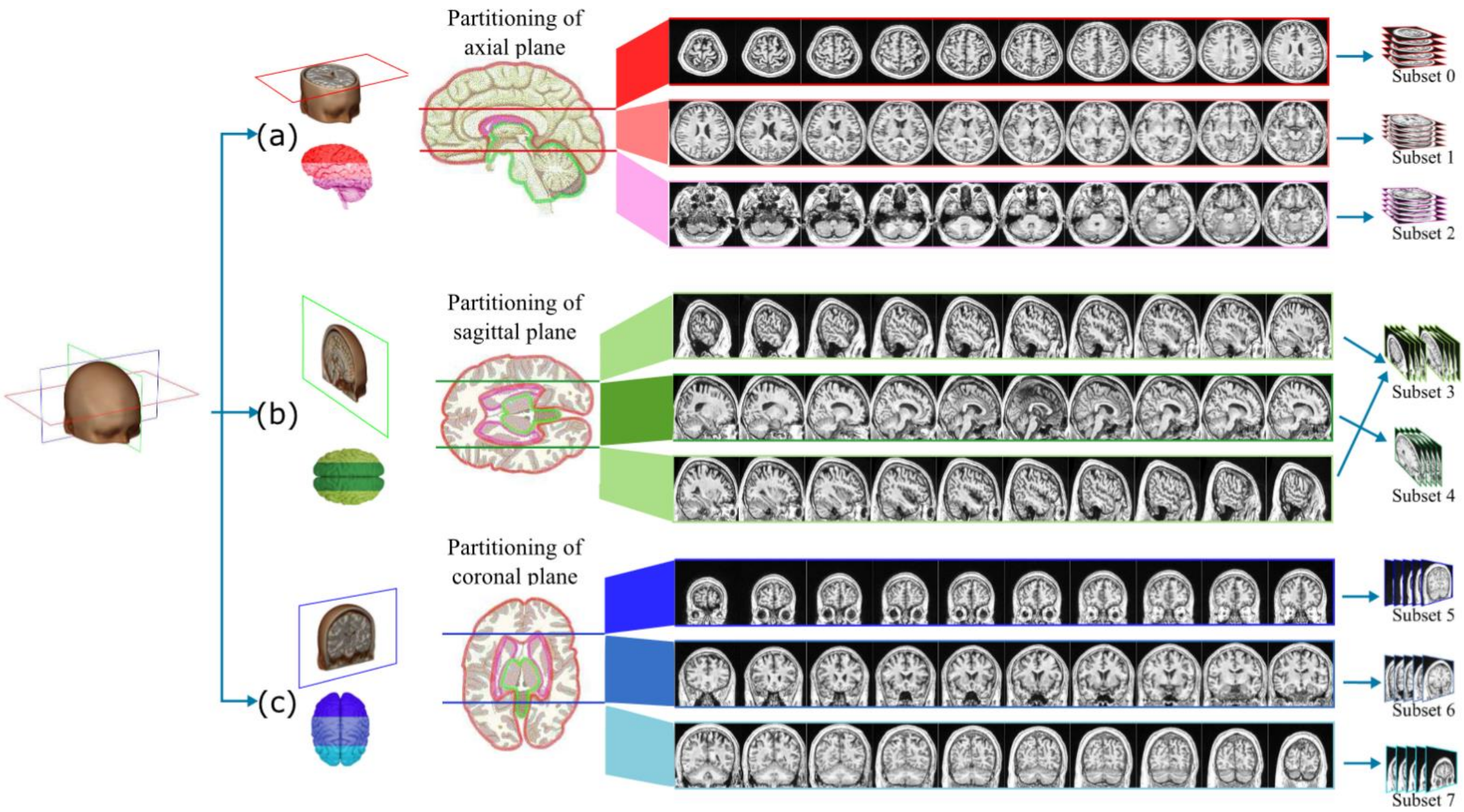
X-Net that employs X-blocks with depth-wise separable convolution layers



[Qi et al., 2019]

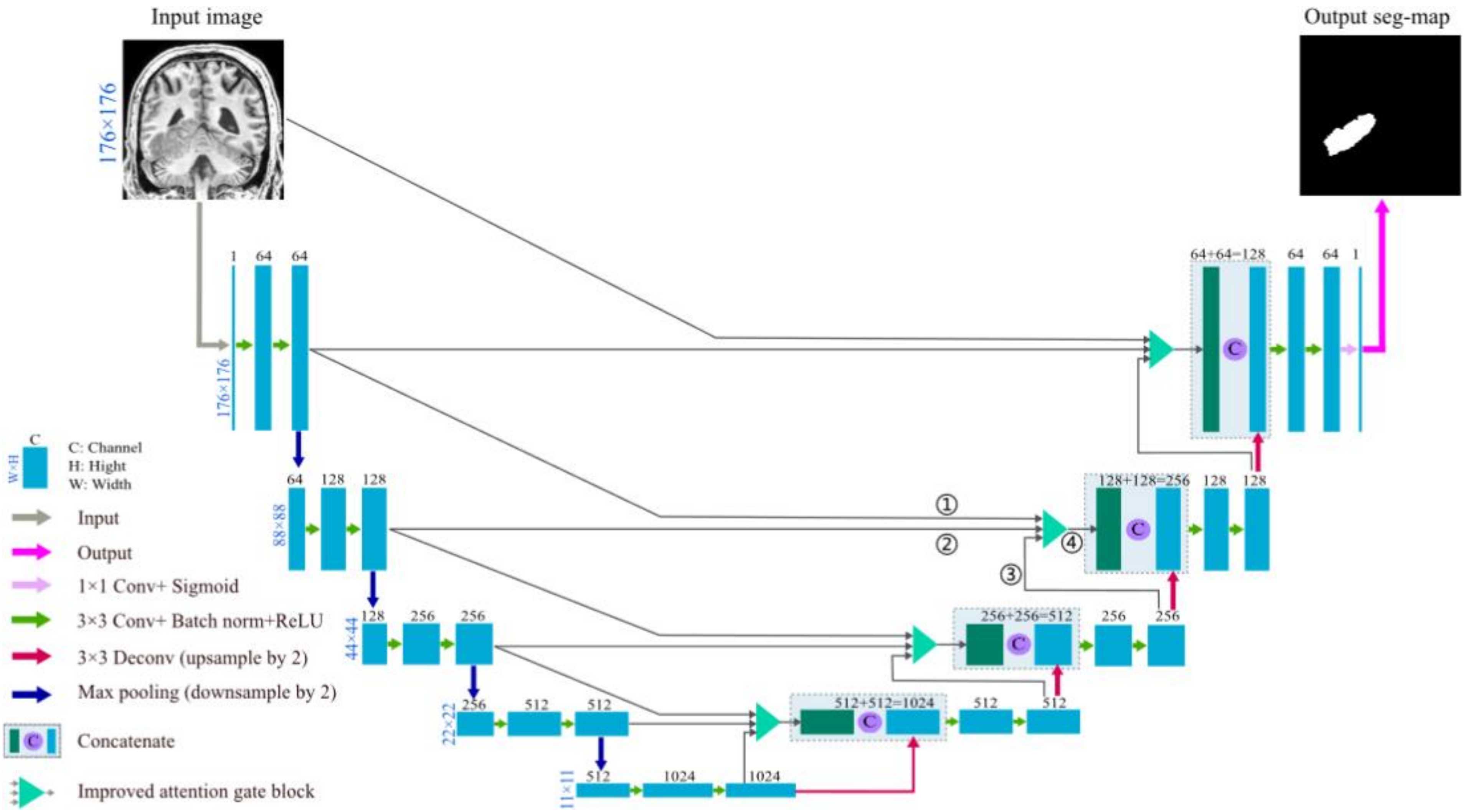
Details of a X-block

- Attention U-Net
  - Hui et al., 2020
    - Proposed a partitioning-stacking prediction fusion (PSPF) method based on an improved attention U-net
    - Reported the DSC of 0.593
      - ATLAS v1.2 dataset
      - Slice input (176 × 176)
      - PSPF method + U-Net



[Hui et al., 2020

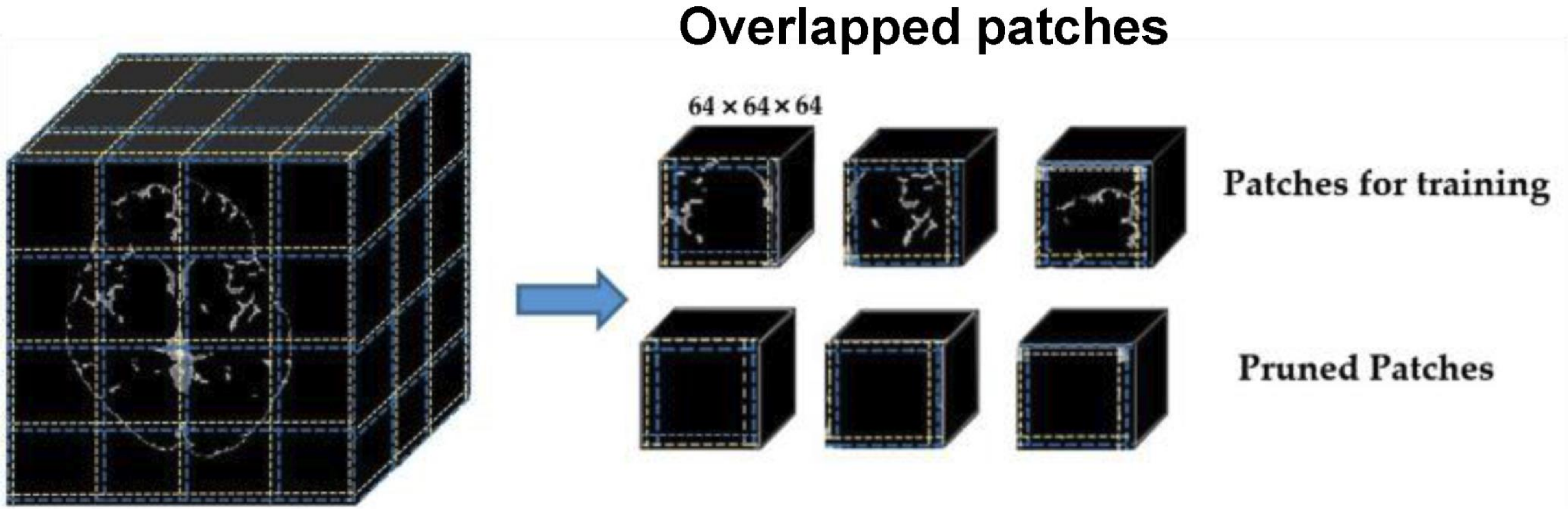
PSPF for three orthogonal planes



[Hui et al., 202]0

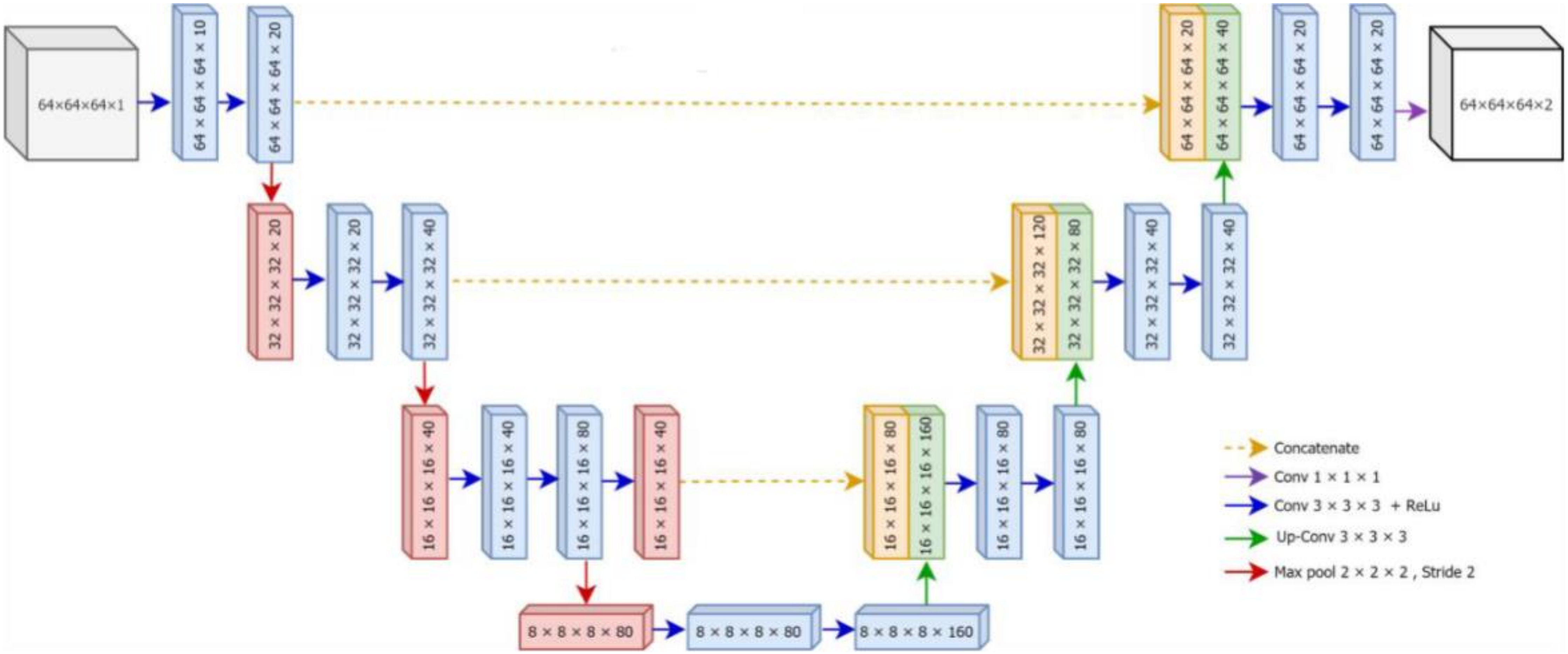
Attentioin U-Net that integrates a attention gate into a U-Net

- 3D U-Net
  - Paing et al., 2021
    - Proposed variational mode decomposition (VMD) as a preprocessing task
    - Used an overlapped patches strategy to reduce the workload of the deep-learning-based segmentation task
    - Reported the DSC of 0.668
      - ATLAS v1.2 dataset
      - Patch input ($64 \times 64 \times 64$)
      - VMD + 3D U-Net



[Paing et al., 2021]

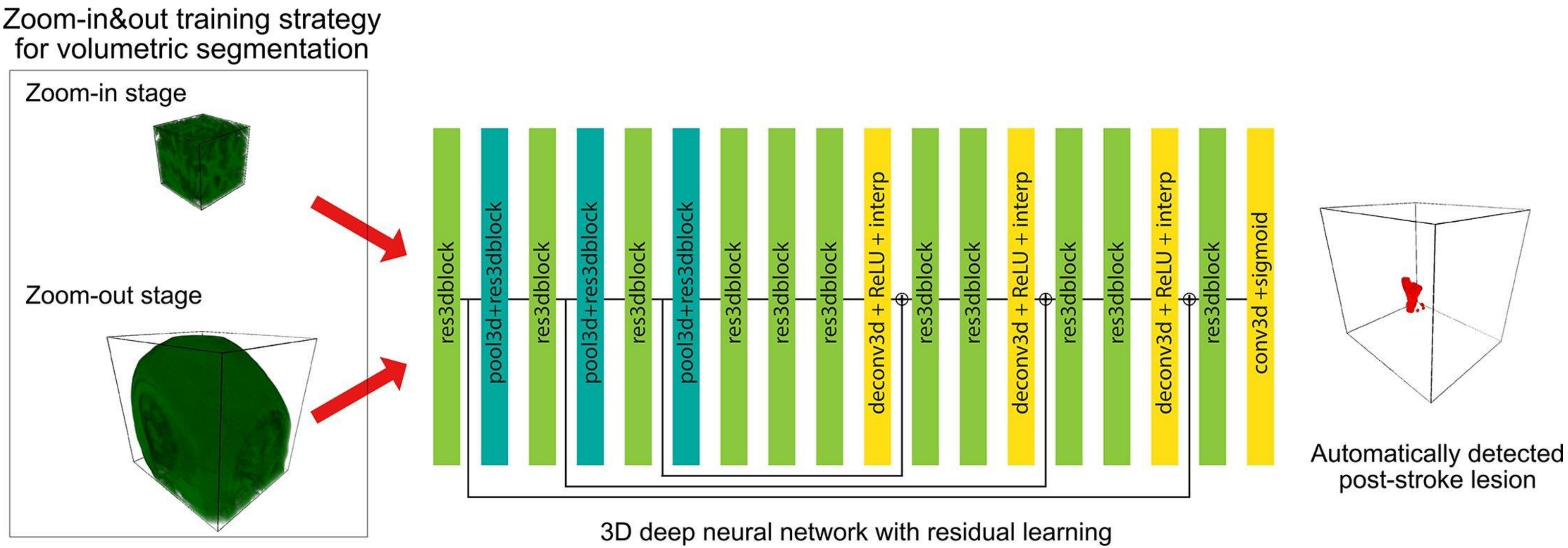
Overlapped patches strategy



[Paing et al., 2021]

3D U-Net for patch-wise segmentation

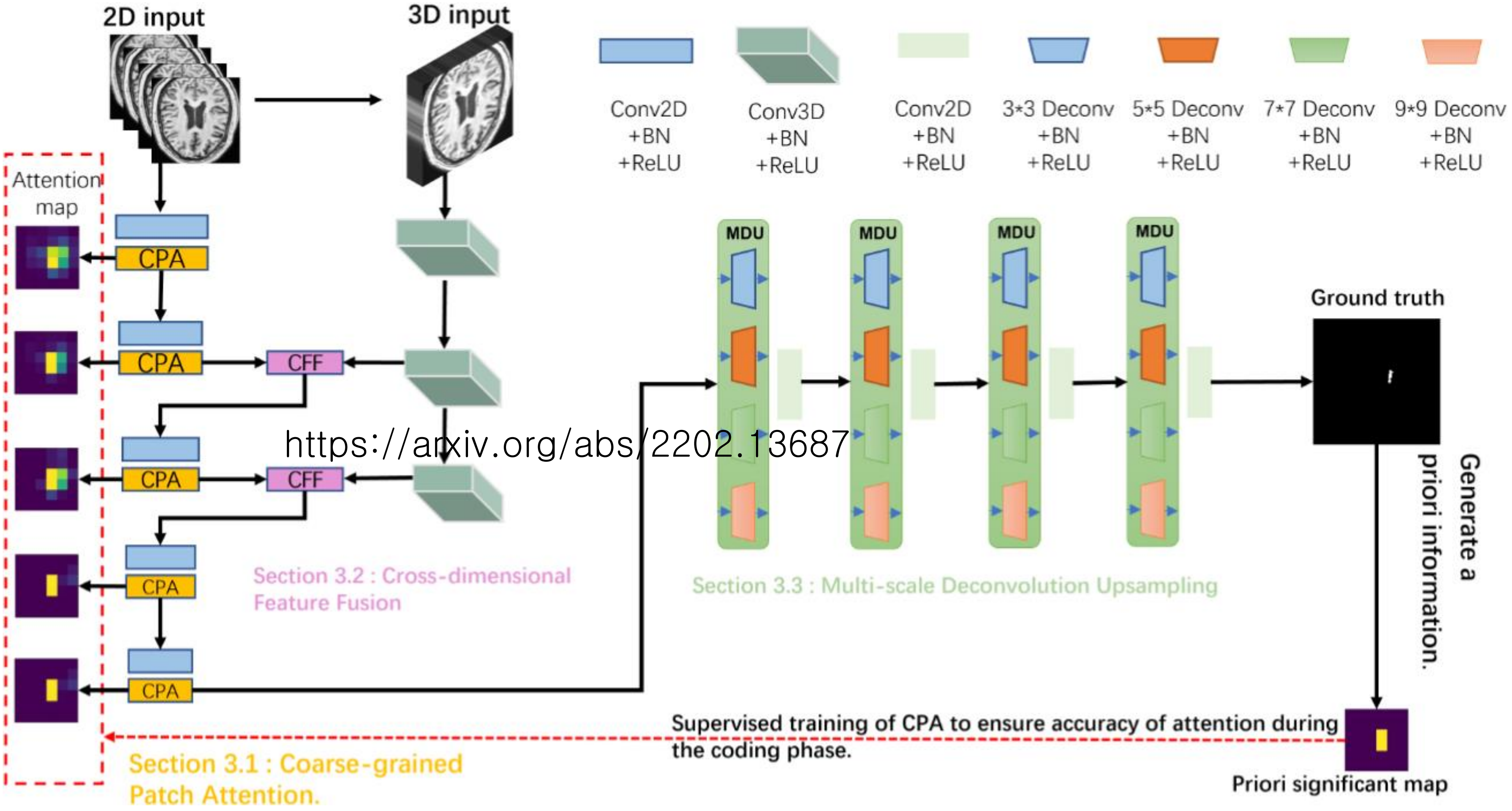
- 3D residual U-Net
  - Tomita et al., 2020
    - Proposed a two-stage zoom-in&out training strategy to first train the model on small volumes and then fine-tune it on larger volumes
    - Reported the DSC of 0.64 (0.51–0.76)
      - ATLAS v1.2 dataset
      - Volume input (128 × 128 × 128 for the zoom-in stage and 144 × 172 × 168 for the zoom-out stage)
      - Zoom-in&out training strategy + 3D residual U-Net
    - https://github.com/BMIRDS/3dMRISegmentation



[Tomita et al., 2020]

3d residual U-Net that incorporates residual connections in forward convolutional layers

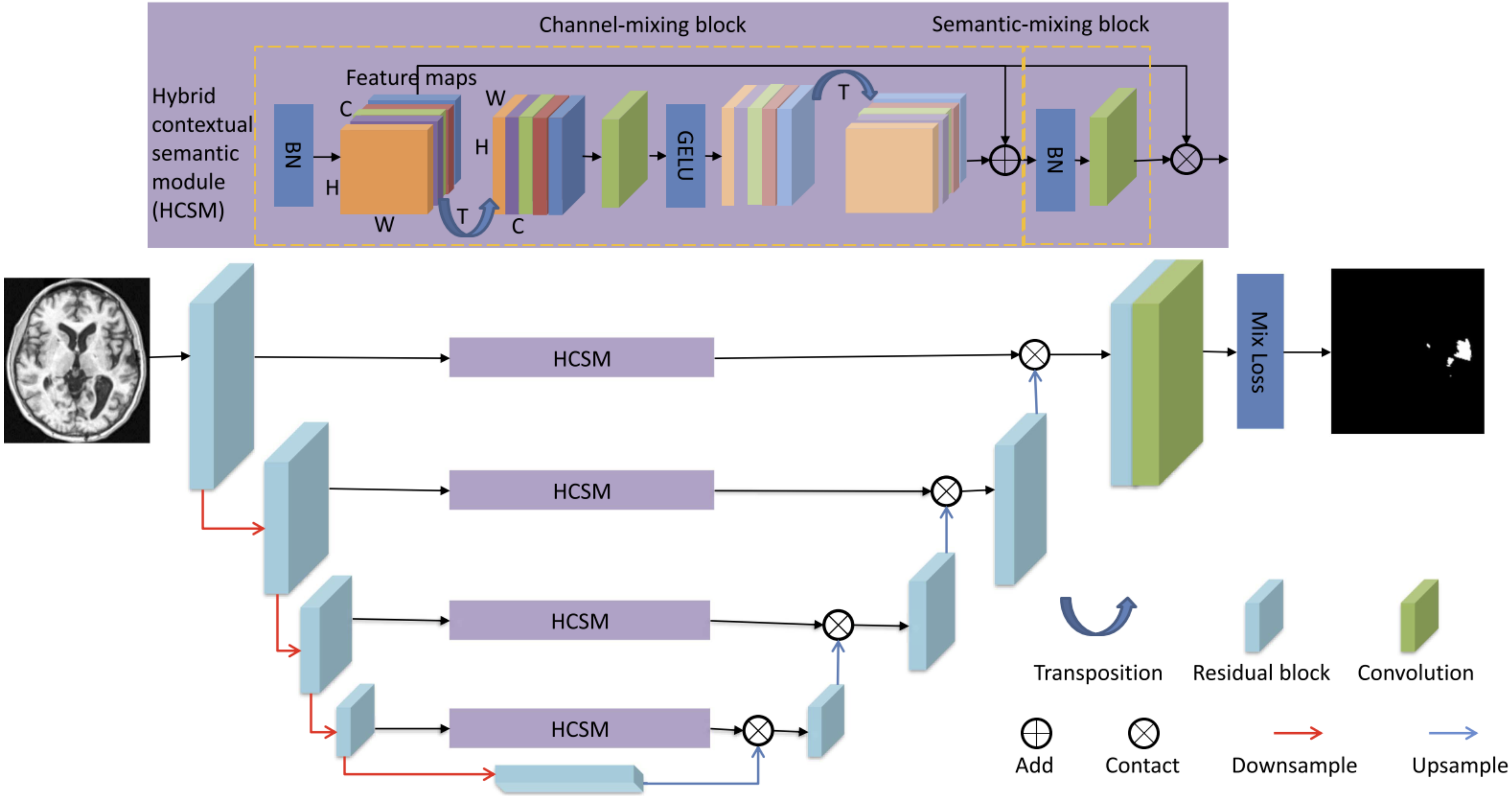
- Attention-guided multiscale recovery (AGMR)-Net
  - Du et al., 2022
    - Proposed a coarse-grained patch attention (CPA) module and a cross-dimensional feature fusion (CFF) module in the encoding phase and a multiscale deconvolution up-sampling (MDU) in the decoding phase
    - Reported the DSC of 0.594
      - ATLAS v1.2 dataset
      - Four consecutive slices input (192 × 192 × 4)
      - AGMR-Net with a CPA module, a CFF module, and an MDU



[Du et al., 2022]

AGMR-Net that employs a CPA module, a CFF module, and an MDU

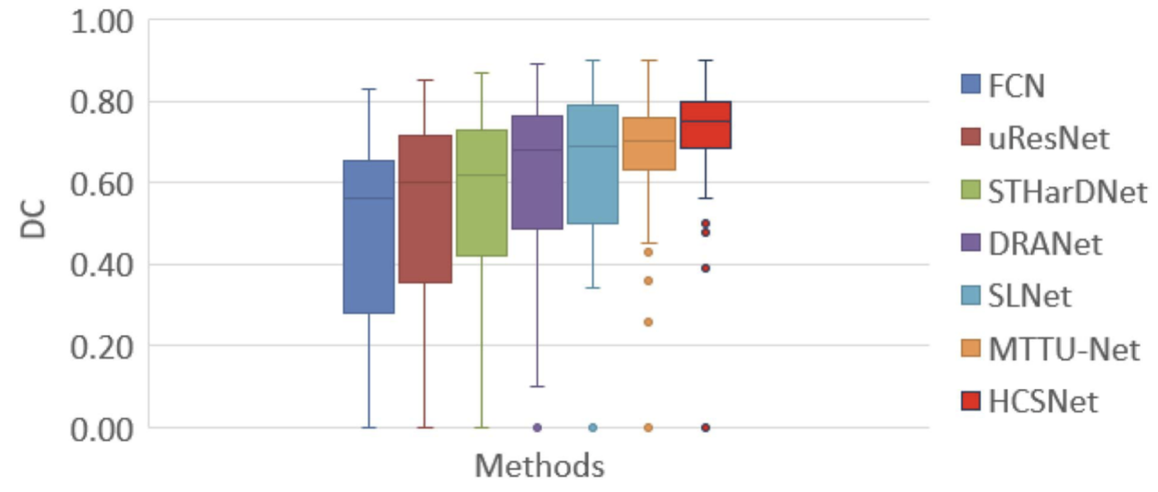
- Hybrid Contextual Semantic (HCS)-Net
  - Liu et al., 2023
    - To accurately and simultaneously segment and detect small-size stroke lesions
    - Applied a hybrid contextual semantic module (HCSM)
    - Proposed a mixing-loss function to optimize the model for unbalanced small-size lesions
    - Reported the DSC of 0.6972
      - ATLAS v2.0 dataset
      - Slice input (240 × 240)
      - HCS-Net with a HCSM



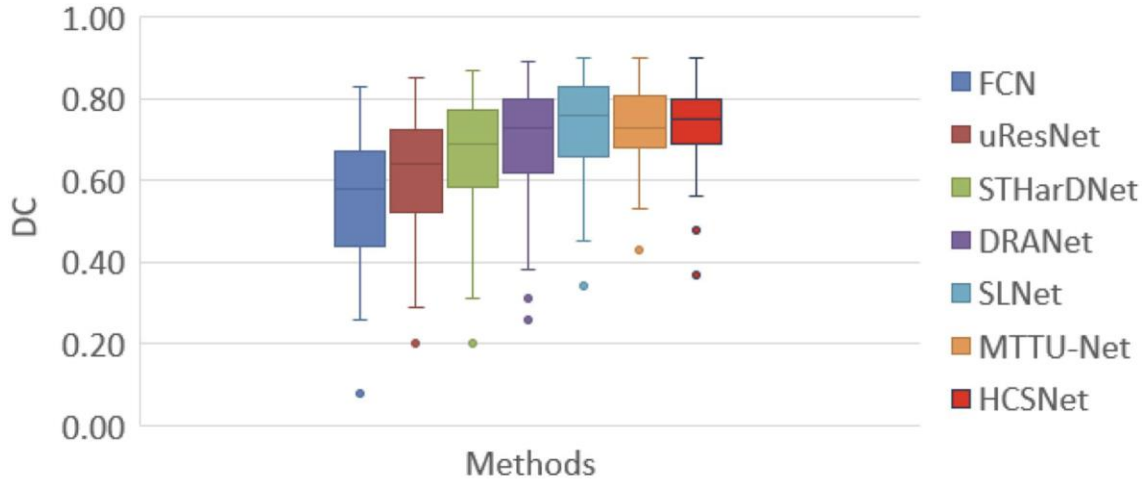
[Liu et al., 2023]

HCS-Net that consists of a U-shaped backbone network and four HCSMs

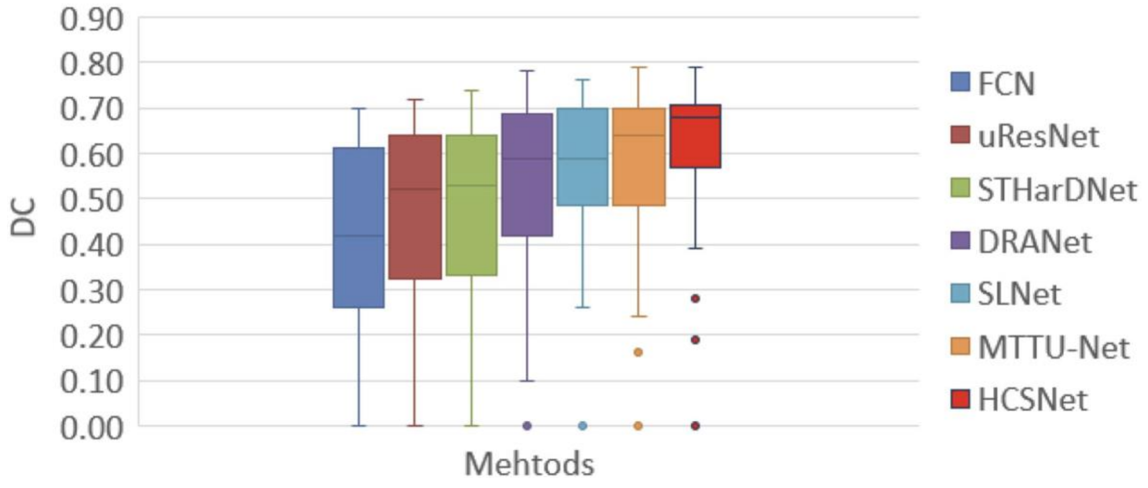
All lesions



Large-size lesions



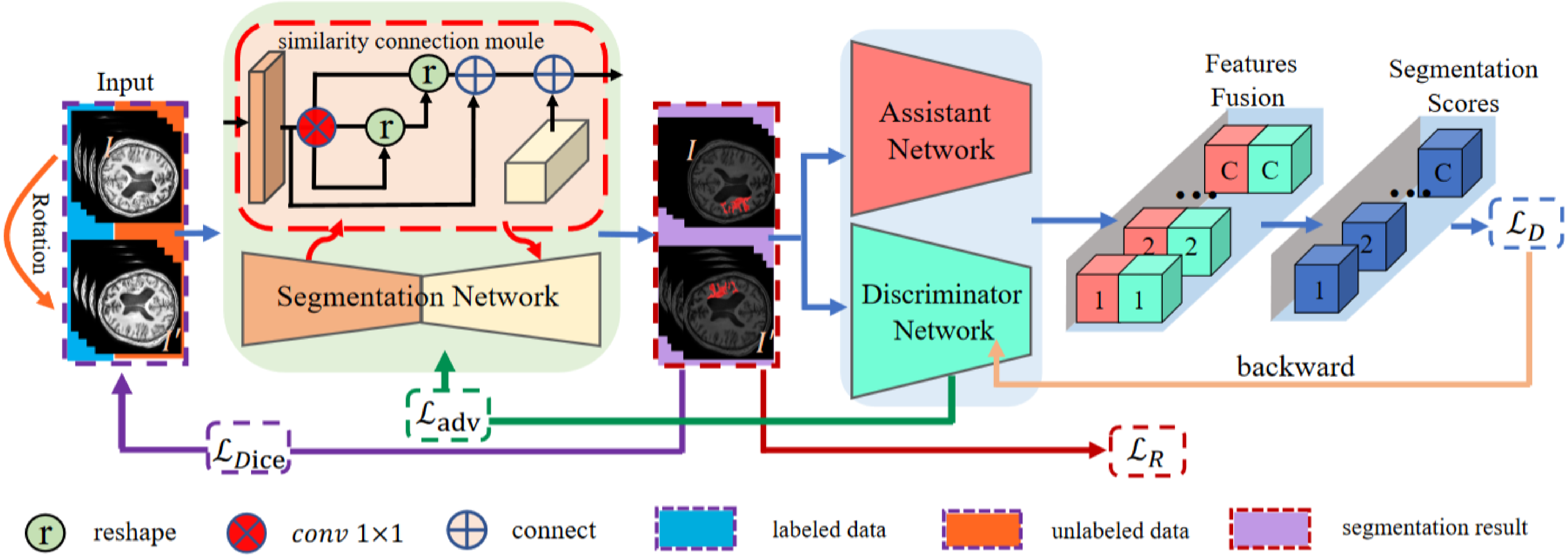
Small-size lesions



[Liu et al., 2023]

Performance for stroke lesions of different sizes

- Consistent Perception Generative Adversarial Network (CPGAN)
  - Wang et al., 2021
    - Proposed a consistent perception strategy for semi-supervised stroke lesion segmentation
    - Employed a similarity connection module (SCM) to extract a wide range of sensitive position information and multi-scale features
    - Reported the DSC of 0.617
      - ATLAS v1.2 dataset
      - Slice input (256 × 256)
      - CPGAN with a SCM



[Wang et al., 2021]

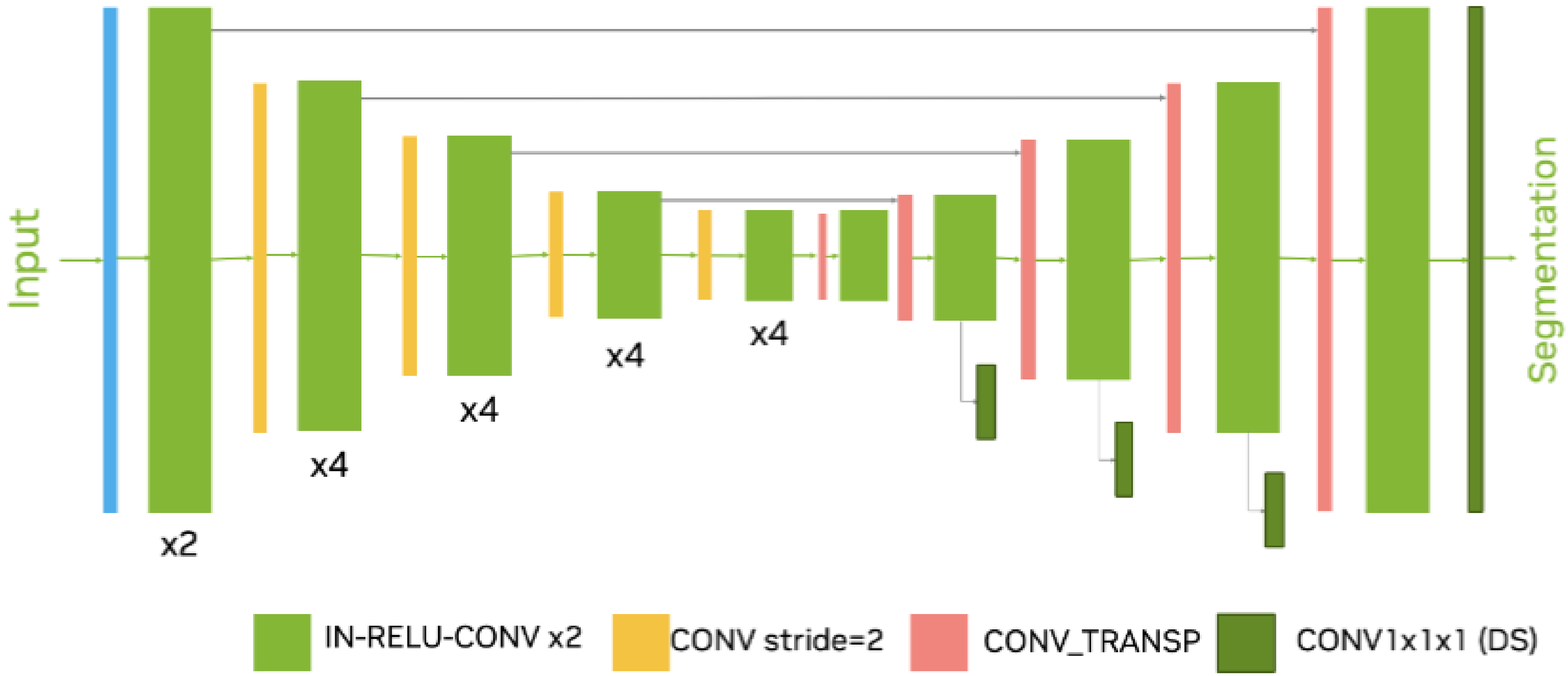
CPGAN that employs an assistant network and a discriminator to make a joint decision

Labeled/Full	Dic	Jac	Acc	Sen	Spe
1	0.617	0.581	0.638	0.556	0.705
0.8	0.613	0.544	0.625	0.531	0.657
0.6	0.544	0.512	0.583	0.529	0.649
0.4	0.502	0.433	0.536	0.523	0.611
0.2	0.457	0.392	0.496	0.477	0.541

[Wang et al., 2021]

Performance for different rates of labeled images

- SegResNet
  - Siddique et al., 2022
    - Solution to the Ischemic Stroke Lesion Segmentation challenge (ISLES 2022)
    - Used two MRI modalities of a diffusion weighted image (DWI) and an apparent diffusion coefficient (ADC) map as the input
    - Reported the DSC of 0.824
      - ISLES dataset
      - Volume input (192 × 192 × 128)
      - SegResNet



[Siddique et al., 2022]

SegResNet that uses repeated ResNet blocks and deep supervision in the decoder branch