# Supplementary Material 1

## Initial Step from Segmentation task to translation

The foundational idea is that if a deep learning model is capable of accurately identifying and positioning organs for segmentation, then it could ostensibly learn to correct an image in desire style by effectively utilizing the right activation functions, loss functions, and an appropriate architectural design. So, for the first step: Could a model, trained on any images, learn to produce an acceptable output by using the same image as both input and target, focusing initially on visual acceptability rather than quantitative metrics?

We utilized CT images as samples before accessing the original data. The experimental setup involved using these images as both the training inputs and targets inputs, aiming to fine-tune the model’s hyperparameters to achieve visually satisfactory outputs. This stage was primarily about understanding the influence of various parameters on the initial results and was not concerned with the precision of error metrics.

Figure 1 and Table 1 in this supplementary section illustrates some of the outputs. This stage served an educational purpose, helping us to understand the foundational dynamics of deep learning applications in corrected images.

Table 1: Some specification of training approach

|  |  |
| --- | --- |
| crop\_size | (512, 512, 32) |
| transforms | ScaleIntensityRanged(keys**=**["image", "target"],a\_min**=-**1024, a\_max**=**2048, b\_min**=**0.0, b\_max**=**1.0, clip**=True**),  Orientationd(keys**=**["image", "target"], axcodes**=**"RAS"),  Spacingd(keys**=**["image", "target"], pixdim**=**(1.5, 1.5, 2.0)),  Resized(keys**=**["image", "target"], spatial\_size**=**crop\_size, mode**=**'bilinear'),  CenterSpatialCropd(keys**=**["image", "target"], roi\_size**=**crop\_size), |
| batch\_size | 4 |
| model | UNet(spatial\_dims**=**3,in\_channels**=**1,out\_channels**=**1,channels**=**(16, 32, 64),act**=**(nn**.**ReLU, {"inplace": **True**}), strides**=**(2, 2), num\_res\_units**=**2, norm**=**Norm**.**BATCH) |
| loss\_function | torch**.**nn**.**MSELoss() |
| optimizer | torch**.**optim**.**Adam(model**.**parameters(), 1e-6) |

A close-up of an x-ray

Description automatically generatedA graph of loss and loss

Description automatically generated

Figure 1: One slice of output and raining loss, from the left to right: input, target and output of the model.

## Different Models

### 3D-Unet-Model

Following the initial phase, we progressed to applying the developed model to the Ga dataset.

To adapt the model for our dataset, several transformations and optimization of hyperparameters tuned to better process the specific profiles of Ga images. First, we checked the model just for one patient data.

Figure 2 and Table 2 in this section detail the variables and outputs from this phase of the project.

A graph showing a line graph

Description automatically generated with medium confidence

A white oval with black dots

Description automatically generated

Figure 2: top: Training and validation loss for 3D-Unet model, bottom: One slice of output.

And then we tried it for a portion of data (20 patient):

A white object with a black circle

Description automatically generated with medium confidence

Figure 3: One slice of output.

Table 2: Some specification of training approach

|  |  |
| --- | --- |
| crop\_size | (180, 180, 312) |
| train\_transforms | Spacingd(keys**=**["image", "target"], pixdim**=**(1.5, 1.5, 2.0)), Resized(keys**=**["image", "target"], spatial\_size**=**crop\_size, ode**=**'bilinear') |
| val\_transforms | Spacingd(keys**=**["image", "target"], pixdim**=**(1.5, 1.5, 2.0)),  Resized(keys**=**["image", "target"], spatial\_size**=**crop\_size, mode**=**('bilinear')) |
| batch\_size | 2 |
| model | UNet(spatial\_dims**=**3, in\_channels**=**1, out\_channels**=**1, channels**=**(16, 32, 64), act**=**(nn**.**ReLU6, {"inplace": **True**}), strides**=**(2, 2), num\_res\_units**=**2,  norm**=**Norm**.**BATCH, |
| loss\_function | torch**.**nn**.**MSELoss() |
| optimizer | torch**.**optim**.**Adam(model**.**parameters(), 1e-4) |
| max\_epochs | 50 |

As it is obvious there was still some patch pattern on the image, and it means there are parameters need to be changed.

As it is mentioned in table 3 after adapting the spacing, dimensions and other parameters for loading the data appropriate for our dataset, and using all dataset, the Figure 4 concluded.

Table 3: Some specification of training approach

|  |  |
| --- | --- |
| roi\_size | [168, 168, 320] |
| train\_transforms | Spacingd(keys=["image", "target"], pixdim=(4.07, 4.07, 3.00)),  SpatialPadd(keys=["image", "target"], spatial\_size=(200, 200, 350), mode='constant'),  CenterSpatialCropd(keys=["image", "target"], roi\_size=roi\_size), |

A graph of a person

Description automatically generated A close-up of several images of a person's body

Description automatically generated

Figure 4: top: Training and validation loss for 3D-Unet model, bottom: One slice of output.

### Patched-3D Unet:

In the initial phase of our research, we attempted to use full-body 3D PET data as single inputs for training our deep learning model. This approach, however, presented significant challenges. The limit number of available data and the limit computational resources required to process full-body 3D data.

Most researchers in this field typically use a 2D slice-wise approach using data-frame images, which significantly reduces the computational demand. Others utilize a smaller section of the data-frame, training their models patch-wise to manage resource constraints effectively. Considering these factors, we opted to focus on using image patches exclusively in the axial direction, and fixed boundaries in the coronal and sagittal dimensions 168 and 168 with each patch containing 32 axial slices. This approach effectively increased our data tenfold, facilitating more extensive training under limited resource conditions.

The outcomes of this method, presented in Figure 5 and Table 4. These results underline the adaptability of our approach in optimizing data usage and computational resources while still enabling robust model training.

Table 4: Some specification of training approach

|  |  |
| --- | --- |
| roi\_size | [168, 168, 320] |
| train\_transforms | Spacingd(keys**=**["image", "target"], pixdim**=**(4.07, 4.07, 3.00), mode**=** 'trilinear'),  SpatialPadd(keys**=**["image", "target"], spatial\_size**=**(168, 168, 320), mode**=**'constant'), *# Pad to ensure minimum size*  RandCropByPosNegLabeld(keys**=**["image", "target"],label\_key**=**"target", spatial\_size**=**(168, 168, 32), pos**=**1,neg**=**1,num\_samples**=**4,image\_key**=**"image",image\_threshold**=**0, |
| val\_transforms | Spacingd(keys**=**["image", "target"], pixdim**=**(4.07, 4.07, 3.00), mode**=** 'trilinear'),  SpatialPadd(keys**=**["image", "target"], spatial\_size**=**(168, 168, 320), mode**=**'constant'), *# Pad to ensure minimum size* |
| batch\_size | 16 |
| model | UNet(spatial\_dims**=**3,in\_channels**=**1,out\_channels**=**1,  channels**=**(32, 64, 128, 256),act**=**(nn**.**ReLU6, {"inplace": **True**}),strides**=**(2, 2, 2, 2), num\_res\_units**=**2, |
| optimizer | torch**.**optim**.**Adam(model**.**parameters(), lr**=**1e-4, weight\_decay**=**0.0) |
| scheduler | torch**.**optim**.**lr\_scheduler**.**StepLR(optimizer, 40, 0.1) |
| max\_epochs | 150 |

A graph with blue and orange lines

Description automatically generated

A group of images of a person's body

Description automatically generatedA group of images of a person's body

Description automatically generated

Figure 5: top: Training and validation loss for 3D-Unet model, bottom: Two sample slices of outputs, Best Metric: 0.2328, Epoch: 148

## 2D-Unet

In addition to search for the best match model to get lower loss and better quality of images, we evaluated a 2D-Unet model training approach.

This 2D U-Net architecture was mostly similar to the previous model. The model training was optimized using an Adam optimizer with a specifically tailored learning rate schedule, which adjusted the learning rate based on the epoch count to enhance training stability and performance.

Some key variables and results detailed in Figure 4 and Table 4.

Table 5: Some specification of training approach

|  |  |
| --- | --- |
| train\_transforms | Spacingd(keys=["image", "target"], pixdim=(4.07, 4.07), mode= 'bilinear'),  spatial\_size=[168, 168], mode='constant' |
| model | monai.networks.nets.UNet(spatial\_dims=2,in\_channels=1,out\_channels=1,channels=(16, 32, 64, 128),strides=(2, 2, 2, 2),num\_res\_units=2, |
| loss\_function | = torch.nn.MSELoss() |
| optimizer | torch.optim.Adam(model.parameters(), lr=learning\_rate, betas=(0.5, 0.999)) |
| max\_epoch | 300 |
| lr\_lambda | DecayLR(epochs=max\_epochs, offset=0, decay\_epochs=decay\_epoch).step |
| scheduler | torch.optim.lr\_scheduler.LambdaLR(optimizer, lr\_lambda=lr\_lambda) |

A close-up of a brain

Description automatically generatedA graph of a graph

Description automatically generated with medium confidence

Figure 6: top: Training and validation loss for 2D-Unet model, bottom: Sample slice of output, Best Metric: 0.206, Epoch: 48

### DyUnet:

In parallel with 2D evaluation, we implemented the DynUNet architecture, an advanced and dynamic variant of the traditional U-Net designed specifically for biomedical image segmentation.

DynUNet introduces several key enhancements over the standard U-Net, including the option for deep supervision. This feature allows the network to output additional intermediate layers' predictions and facilitate the learning process by ensuring that gradients are effectively propagated back through the network, enhancing the training dynamics and enabling the model to learn detailed representations without significant overfitting.

With compatible configurations of kernel sizes and strides and depth of artitecture, model enables to effectively capture relevant features at different scales.

Key configuration parameters of DynUNet are listed in table 6 and there is one sample output from out initial implementation in the Figure 5.

Table 6: Some specification of training approach

|  |  |
| --- | --- |
| patch\_size | [168, 168, 16] |
| spacing | [4.07, 4.07, 3.00] |
| spatial\_size | (168, 168, 320) |
| train\_transforms | Spacingd(keys=["image", "target"], pixdim= spacing, mode= 'trilinear'),SpatialPadd(keys=["image", "target"], spatial\_size=spatial\_size, mode='constant'), RandSpatialCropSamplesd(keys=["image", "target"], roi\_size=self.patch\_size, num\_samples=4), |
| val\_transforms | CenterSpatialCropd(keys=["image", "target"], roi\_size=self.spatial\_size) |
| Model | DynUNet( spatial\_dims=3, in\_channels=1, out\_channels=1, kernel\_size=kernels, strides=strides, upsample\_kernel\_size=strides[1:], norm\_name="INSTANCE", deep\_supervision=True, deep\_supr\_num=2,) |

After these improvements, finally we could decrease the validation loss from around 0.2 at the initially trials to the 0.0664.

To enhance the robustness of our model, we implemented specific data augmentations. These included adding rotations of ±15 degrees and increasing the number of samples per patient from 4 to 20.

A graph of a person with blue and orange lines

Description automatically generatedA graph with a number of lines

Description automatically generated with medium confidence

A collage of images of a person's body

Description automatically generated

Figure 7: top: Training and validation loss for 2D-Unet model, bottom: Sample slice of output, Best Metric: 0.206, Epoch: 48

Here are some other metric errors from the beginning of this research:

|  |  |
| --- | --- |
| At early stage | ME: 0.64  MAE: 0.95  RE: 193.7%  ARE: 199.0% |
| Using Unet | Mean Error (SUV): -0.32 ± 0.1032  Mean Absolure Error (SUV): 0.33 ± 0.0868  Relative Error (SUV%): -55.49 ± 15.6193  Absolure Relative Error (SUV%): 56.98 ± 13.3306  Root Mean Squared Error: 0.48 ± 0.1741  Peak Signal-to-Noise Ratio: 23.92 ± 6.4356  Structual Similarity Index: 0.63 ± 0.1537 |
| Using DynUnet | mean\_error: -0.43 ± 0.3433  mean\_absolute\_error: 0.54 ± 0.2896  relative\_error: -23.92 ± 14.8091  absolute\_relative\_error: 35.36 ± 7.7831  rmse: 1.13 ± 0.8008  psnr: 32.57 ± 4.2616  ssim: 0.87 ± 0.0568 |
| DynUnet, ADCM method | Mean Error (SUV): -0.42 ± 0.0783  Mean Absolure Error (SUV): 0.42 ± 0.0767  Relative Error (SUV%): -72.41 ± 10.2247  Absolure Relative Error (SUV%): 72.65 ± 9.9125  Root Mean Squared Error: 0.57 ± 0.1856  Peak Signal-to-Noise Ratio: 22.53 ± 6.7792  Structual Similarity Index: 0.44 ± 0.1617 |