inference

September 16, 2022

1 Initialize

[ ]:
```
import sys
sys.path.append("/media/hdd/viscent/SR-UNet")
from utils.inference_utils import *
```

[ ]:
```
dhcp_train_loader, dhcp_test_loader, dhcp_val_loader =␣
 ↪get_dataloader('dhcp',100,modality='t2')
hcp_train_loader, hcp_test_loader, hcp_val_loader =␣
 ↪get_dataloader('hcp',100,modality='t2')

dhcp_t2_1500 = load_model('/media/hdd/viscent/SR-UNet/pretrained_models/
 ↪dhcp_t2_1500.pth')
dhcp_t2_1500_aug = load_model('/media/hdd/viscent/SR-UNet/pretrained_models/
 ↪dhcp_t2_1500_aug.pth')
hcp_t2_140_aug = load_model('/media/hdd/viscent/SR-UNet/pretrained_models/
 ↪hcp_t2_140_aug.pth')

hcp_t2_140_aug.eval()
hcp_t2_140_aug.cuda()
dhcp_t2_1500.eval()
dhcp_t2_1500.cuda()
dhcp_t2_1500_aug.eval()
dhcp_t2_1500_aug.cuda()
```

```
[09/16/22 17:36:22] INFO     colossalai - root - INFO: Creating dataset with 80␣
 ↪examples

                    INFO     colossalai - root - INFO: length of list_images_t1:␣
 ↪80

  0%|          | 0/80 [00:00<?, ?it/s]
[09/16/22 17:36:25] INFO     colossalai - root - INFO: Creating dataset with 20␣
 ↪examples
```

```
                    INFO     colossalai - root - INFO: length of list_images_t1:␣
↪20

 0%|          | 0/20 [00:00<?, ?it/s]

                    INFO     colossalai - root - INFO: Creating dataset with 80␣
↪examples

 0%|          | 0/80 [00:00<?, ?it/s]

[09/16/22 17:36:27] INFO     colossalai - root - INFO: Creating dataset with 20␣
↪examples

 0%|          | 0/20 [00:00<?, ?it/s]
```

[ ]:
```
BUNet3D(
  (encoders): ModuleList(
    (0): Encoder(
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 1, eps=1e-05, affine=True)
          (conv): Conv3d(1, 8, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 8, eps=1e-05, affine=True)
          (conv): Conv3d(8, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
      )
    )
    (1): Encoder(
      (pooling): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 16, eps=1e-05, affine=True)
          (conv): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 16, eps=1e-05, affine=True)
          (conv): Conv3d(16, 32, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
```

```
          (ReLU): ReLU(inplace=True)
        )
      )
    )
    (2): Encoder(
      (pooling): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 32, eps=1e-05, affine=True)
          (conv): Conv3d(32, 32, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 32, eps=1e-05, affine=True)
          (conv): Conv3d(32, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
      )
    )
    (3): Encoder(
      (pooling): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 64, eps=1e-05, affine=True)
          (conv): Conv3d(64, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 64, eps=1e-05, affine=True)
          (conv): Conv3d(64, 128, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
      )
    )
    (4): Encoder(
      (pooling): MaxPool3d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 128, eps=1e-05, affine=True)
          (conv): Conv3d(128, 128, kernel_size=(3, 3, 3), stride=(1, 1, 1),
```

```
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 128, eps=1e-05, affine=True)
          (conv): Conv3d(128, 256, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
      )
    )
  )
  (decoders): ModuleList(
    (0): Decoder(
      (upsampling): InterpolateUpsampling()
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 384, eps=1e-05, affine=True)
          (conv): Conv3d(384, 128, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 128, eps=1e-05, affine=True)
          (conv): Conv3d(128, 128, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
      )
    )
    (1): Decoder(
      (upsampling): InterpolateUpsampling()
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 192, eps=1e-05, affine=True)
          (conv): Conv3d(192, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 64, eps=1e-05, affine=True)
          (conv): Conv3d(64, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
      )
    )
```

```
    (2): Decoder(
      (upsampling): InterpolateUpsampling()
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 96, eps=1e-05, affine=True)
          (conv): Conv3d(96, 32, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 32, eps=1e-05, affine=True)
          (conv): Conv3d(32, 32, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
      )
    )
    (3): Decoder(
      (upsampling): InterpolateUpsampling()
      (basic_module): DoubleConv(
        (SingleConv1): SingleConv(
          (groupnorm): GroupNorm(1, 48, eps=1e-05, affine=True)
          (conv): Conv3d(48, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
        (SingleConv2): SingleConv(
          (groupnorm): GroupNorm(1, 16, eps=1e-05, affine=True)
          (conv): Conv3d(16, 16, kernel_size=(3, 3, 3), stride=(1, 1, 1),
padding=(1, 1, 1), bias=False)
          (ReLU): ReLU(inplace=True)
        )
      )
    )
  )
  (final_conv): Conv3d(16, 1, kernel_size=(1, 1, 1), stride=(1, 1, 1))
  (mu): Linear(in_features=256, out_features=1, bias=True)
  (logvar): Linear(in_features=256, out_features=1, bias=True)
  (latent_to_decode): Linear(in_features=1, out_features=256, bias=True)
  (transform): Sequential(
    (0): RandomRotation3D(RandomRotation3D(degrees=(15.0, 20.0, 20.0), p=0.5,
p_batch=1.0, same_on_batch=False, resample=bilinear, align_corners=False))
    (1): RandomMotionBlur3D(RandomMotionBlur3D(kernel_size=3, angle=35.0,
direction=0.5, p=0.4, p_batch=1.0, same_on_batch=False, border_type=constant,
resample=nearest))
    (2): RandomAffine3D(RandomAffine3D(degrees=(15.0, 20.0, 20.0), shears=None,
translate=None, scale=None, p=0.4, p_batch=1.0, same_on_batch=False,
```

```
resample=bilinear, align_corners=False))
  )
)
```

[ ]:
```
data_root = '/media/hdd/viscent/FLYWHEEL_BROWN/BROWN/SUBJECTS'
subject_list = os.listdir(data_root)
# Exclude hidden files
subject_list = [x for x in subject_list if not x.startswith('.')]
t2_file_list = []
for subject in subject_list:
    tmp = os.listdir(os.path.join(data_root, subject, 'SESSIONS'))
    tmp = [x for x in tmp if not x.startswith('.')]
    session = tmp[0]
    tmp = os.listdir(os.path.join(data_root, subject, 'SESSIONS', session,␣
 ↪'ACQUISITIONS'))
    tmp = [x for x in tmp if x.find('T2') != -1 and x.find('AXI') != -1 and not␣
 ↪x.startswith('.')]
    if len(tmp) == 0:
        continue
    nifti_dir = os.path.join(data_root, subject, 'SESSIONS', session,␣
 ↪'ACQUISITIONS', tmp[0], 'FILES')
    nifti_file = os.listdir(nifti_dir)
    nifti_file = [x for x in nifti_file if x.endswith('.nii.gz') and not x.
 ↪startswith('.')][0]
    t2_file_list.append(os.path.join(nifti_dir, nifti_file))
```

1.1 Preprocessing

[ ]:
```
import shutil

for i,t2_file in enumerate(t2_file_list):
    shutil.copy(t2_file, '/media/hdd/viscent/SR-UNet/inference/
 ↪Inference_FLYWHEEL_BROWN/input_raw/%03d.nii.gz'%i)
    os.system('bet2 /media/hdd/viscent/SR-UNet/inference/
 ↪Inference_FLYWHEEL_BROWN/input_raw/%03d.nii.gz /media/hdd/viscent/SR-UNet/
 ↪inference/Inference_FLYWHEEL_BROWN/input_brain/%03d.nii.gz'% (i,i))
```

[ ]:
```
for i in range(113):
    os.system('flirt -in /media/hdd/viscent/SR-UNet/inference/
 ↪Inference_FLYWHEEL_BROWN/input_brain/%03d.nii.gz -ref /media/hdd/viscent/
 ↪SR-UNet/inference/Template_dHCP.nii.gz -out /media/hdd/viscent/SR-UNet/
 ↪inference/Inference_FLYWHEEL_BROWN/input_reg/%03d.nii.gz'% (i,i))
```

```
Final result:
0.097415 -0.075573 -0.679350 123.527223
-0.042392 0.753587 -0.149819 17.827600
```

```
0.788458 0.067552 0.104333 -1.524559
0.000000 0.000000 0.000000 1.000000


Final result:
-0.012858 -0.239369 -0.720638 173.624255
-0.046686 0.764112 -0.145254 10.248861
0.778178 0.054591 0.046926 2.086636
0.000000 0.000000 0.000000 1.000000


Final result:
-0.044277 -0.014296 -0.729924 134.591992
0.003228 0.749314 -0.030424 0.589731
0.742170 0.000962 -0.046207 19.306903
0.000000 0.000000 0.000000 1.000000


Final result:
-0.161719 0.046834 -0.710888 138.715748
0.019304 0.699681 -0.015276 8.823828
0.749340 -0.013568 -0.179409 23.651455
0.000000 0.000000 0.000000 1.000000


Final result:
0.095646 0.104757 -0.679562 115.541731
-0.018467 0.755144 0.006462 4.772346
0.761878 0.007360 0.069567 8.075729
0.000000 0.000000 0.000000 1.000000


Final result:
0.640717 0.173220 0.014730 2.084022
0.133769 -0.513423 -0.545093 173.133921
-0.147562 0.414933 -0.544700 98.251680
0.000000 0.000000 0.000000 1.000000


Final result:
0.054232 -0.114829 -0.664596 149.349148
0.090567 0.722395 -0.132845 11.285448
0.769344 -0.083908 0.065130 10.242936
0.000000 0.000000 0.000000 1.000000


Final result:
-0.037744 -0.006239 -0.693868 133.549075
```

```
-0.040852 0.822099 0.016804 5.691403
0.792663 0.049815 -0.036975 6.491916
0.000000 0.000000 0.000000 1.000000


Final result:
0.057672 0.006395 -0.695112 130.762477
-0.015163 0.772189 -0.030001 0.397969
0.730625 0.016222 0.034060 9.414057
0.000000 0.000000 0.000000 1.000000


Final result:
-0.135876 -0.021997 -0.656666 141.201230
-0.001987 0.720623 -0.002199 1.654473
0.773239 0.018896 -0.122416 18.482433
0.000000 0.000000 0.000000 1.000000


Final result:
0.012352 -0.002705 -0.697568 153.604003
0.044215 0.808258 -0.045275 2.800556
0.713054 -0.076707 0.018746 20.563773
0.000000 0.000000 0.000000 1.000000


Final result:
-0.126375 0.032079 -0.611724 132.208097
-0.093219 0.702197 0.015042 13.043375
0.737576 0.123495 -0.087519 9.716991
0.000000 0.000000 0.000000 1.000000


Final result:
0.555688 -0.268730 0.308964 24.617188
0.123336 0.676688 0.187373 -22.883612
-0.385756 -0.168550 0.557743 64.252510
0.000000 0.000000 0.000000 1.000000


Final result:
0.041239 0.011091 -0.689919 129.399880
0.010272 0.800154 0.005429 -3.859201
0.746691 -0.027675 0.037032 13.502270
0.000000 0.000000 0.000000 1.000000


Final result:
```

```
-0.010260 0.001394 -0.701168 151.525682
-0.021752 0.799727 -0.019120 5.605549
0.848783 0.027706 -0.012040 6.110654
0.000000 0.000000 0.000000 1.000000


Final result:
0.023570 -0.000566 -0.687443 143.739182
-0.007447 0.801841 -0.005275 6.294485
0.791701 -0.015473 0.019022 10.631143
0.000000 0.000000 0.000000 1.000000


Final result:
-0.097751 -0.101418 -0.652380 147.291502
-0.218029 0.753641 -0.128237 35.844178
0.764542 0.201207 -0.114297 2.912575
0.000000 0.000000 0.000000 1.000000


Final result:
0.019648 0.117294 -0.663795 131.036945
-0.001306 0.724402 0.103356 -4.560207
0.752488 0.001459 0.001014 12.973491
0.000000 0.000000 0.000000 1.000000


Final result:
0.685602 -0.038161 0.235763 5.939477
0.050537 0.720401 -0.058504 5.832625
-0.153485 -0.046420 0.788339 50.079534
0.000000 0.000000 0.000000 1.000000


Final result:
0.087749 0.088672 -0.756210 114.746007
0.025432 0.798152 0.098351 -7.640558
0.765234 -0.038363 0.083240 16.677734
0.000000 0.000000 0.000000 1.000000


Final result:
0.035270 -0.011225 -0.710085 132.212882
-0.111571 0.841284 -0.001877 15.743334
0.688890 0.104718 0.026201 7.282804
0.000000 0.000000 0.000000 1.000000
```

```
Final result:
0.099037 0.062471 -0.816934 141.740692
-0.044195 0.791882 -0.023780 2.747650
0.794317 0.038569 0.165572 -9.555716
0.000000 0.000000 0.000000 1.000000


Final result:
0.646563 -0.005677 -0.243844 48.995448
-0.043525 0.747458 -0.035927 5.124596
0.290073 -0.022011 0.683129 -13.072387
0.000000 0.000000 0.000000 1.000000


Final result:
0.568321 -0.150511 0.376740 5.500373
0.089609 0.727124 0.034753 -8.289189
-0.387344 -0.036522 0.570299 59.718231
0.000000 0.000000 0.000000 1.000000


Final result:
-0.000517 -0.174071 -0.717887 138.641779
-0.097832 0.787686 -0.246409 32.712360
0.729333 0.075817 -0.016916 5.251236
0.000000 0.000000 0.000000 1.000000


Final result:
0.339087 -0.271549 0.782209 -17.469671
0.531984 -0.483313 -0.559151 127.931081
0.555165 0.637356 -0.048568 -11.506388
0.000000 0.000000 0.000000 1.000000


Final result:
0.174630 0.127219 -0.689288 107.757369
-0.103949 0.722485 0.047400 10.892567
0.795008 0.076935 0.182732 -13.598302
0.000000 0.000000 0.000000 1.000000


Final result:
-0.239281 0.062882 -0.690502 140.092139
0.093249 0.802561 0.014538 -3.381015
0.794237 -0.080032 -0.237632 32.786604
0.000000 0.000000 0.000000 1.000000
```

```
Final result:
-0.114055 -0.009540 -0.697487 138.044826
0.022634 0.694116 -0.019491 5.694333
0.736580 -0.018113 -0.093242 22.162212
0.000000 0.000000 0.000000 1.000000


Final result:
0.657089 -0.141871 0.163341 20.327799
0.146148 0.694898 -0.063551 1.200411
-0.099850 0.034028 0.669971 21.908412
0.000000 0.000000 0.000000 1.000000


Final result:
-0.023885 -0.146593 -0.667693 147.085691
-0.017407 0.719249 -0.142754 13.189686
0.732493 -0.014577 0.003023 11.618427
0.000000 0.000000 0.000000 1.000000


Final result:
0.030189 -0.197395 -0.644108 158.134433
0.073956 0.745269 -0.252729 25.069653
0.766125 -0.070404 0.047397 18.517086
0.000000 0.000000 0.000000 1.000000


Final result:
-0.019212 -0.234523 -0.618302 160.153148
0.074451 0.662965 -0.272078 20.464691
0.759935 -0.104768 -0.019345 25.388849
0.000000 0.000000 0.000000 1.000000


Final result:
0.045349 -0.104233 -0.642637 146.204233
-0.029654 0.752303 -0.121565 22.149213
0.703079 0.063421 0.051938 8.728045
0.000000 0.000000 0.000000 1.000000


Final result:
-0.050450 -0.028919 -0.737804 154.251529
-0.021714 0.764189 -0.088443 15.343819
0.801785 0.020409 -0.037422 7.613261
0.000000 0.000000 0.000000 1.000000
```

```
Final result:
0.201166 -0.037151 0.588462 -0.036409
-0.032914 0.783092 -0.047974 5.774308
-0.667431 -0.066062 0.205471 121.708884
0.000000 0.000000 0.000000 1.000000


Final result:
0.000938 0.000118 -0.721497 135.508031
-0.009961 0.795152 -0.035351 5.963869
0.846422 0.022409 -0.028167 6.542873
0.000000 0.000000 0.000000 1.000000


Final result:
0.120916 -0.135143 -0.661622 153.775424
-0.102902 0.780957 -0.147525 21.277867
0.781524 0.086154 0.106504 0.145022
0.000000 0.000000 0.000000 1.000000


Final result:
0.695389 -0.255334 0.191003 21.445319
0.240536 0.742294 0.009989 -17.167045
-0.202652 -0.013460 0.671411 25.459141
0.000000 0.000000 0.000000 1.000000


Final result:
0.038462 0.256990 -0.702083 102.066300
-0.124835 0.699760 0.181899 11.150230
0.813407 0.115301 0.053119 -3.495235
0.000000 0.000000 0.000000 1.000000


Final result:
0.554687 -0.191565 0.394994 6.769395
0.128070 0.725909 0.070361 -13.845151
-0.433764 -0.010621 0.628687 53.534122
0.000000 0.000000 0.000000 1.000000


Final result:
0.056423 0.092065 -0.696828 123.741998
-0.024058 0.787556 0.034413 9.942969
0.783792 0.035324 0.056783 4.315150
```

```
0.000000 0.000000 0.000000 1.000000


Final result:
0.822215 0.036618 0.475696 -49.706219
-0.168408 0.856848 -0.091272 14.355533
-0.397034 -0.041706 0.768331 49.670760
0.000000 0.000000 0.000000 1.000000


Final result:
0.054365 0.007383 -0.724817 133.775545
0.034753 0.779598 -0.084857 4.825512
0.741171 -0.012367 0.046402 19.304907
0.000000 0.000000 0.000000 1.000000


Final result:
0.026667 0.004052 -0.722016 145.949973
0.006776 0.858547 -0.039519 -3.693487
0.795356 0.013483 0.011558 11.011546
0.000000 0.000000 0.000000 1.000000


Final result:
-0.032327 -0.205829 -0.643049 149.381618
0.004385 0.747588 -0.242558 28.182088
0.774736 -0.024101 -0.016379 10.642182
0.000000 0.000000 0.000000 1.000000


Final result:
0.306935 0.481135 -0.437030 37.794264
-0.793631 0.479927 0.013058 98.607338
0.264001 0.337837 0.632078 -35.738643
0.000000 0.000000 0.000000 1.000000


Final result:
-0.220045 -0.008583 0.709961 9.987785
-0.021549 0.794426 -0.137135 12.646153
-0.779527 -0.011269 -0.257965 169.022129
0.000000 0.000000 0.000000 1.000000


Final result:
0.058698 0.064916 -0.620440 130.483648
-0.057475 0.735307 0.046371 5.726998
```

```
0.794558 0.056300 0.054190 -2.421339
0.000000 0.000000 0.000000 1.000000


Final result:
0.674895 -0.086508 0.233476 3.558886
0.102582 0.753098 -0.098797 3.906553
-0.212255 0.061713 0.710170 17.914219
0.000000 0.000000 0.000000 1.000000


Final result:
-0.026301 0.009926 -0.728584 140.649131
0.000980 0.757057 0.018638 -1.934249
0.695752 0.011625 -0.016568 16.738864
0.000000 0.000000 0.000000 1.000000


Final result:
-0.001116 -0.000064 -0.660288 136.692081
0.034674 0.742852 0.012783 1.148430
0.763019 -0.017462 -0.000114 14.622050
0.000000 0.000000 0.000000 1.000000


Final result:
-0.019117 -0.140116 -0.690760 137.487922
-0.085729 0.725654 -0.155329 25.611730
0.691242 0.086020 -0.013987 12.621040
0.000000 0.000000 0.000000 1.000000


Final result:
-0.150434 -0.003636 -0.752692 149.454906
-0.030733 0.769016 -0.024205 1.799180
0.758539 0.018184 -0.096050 16.127384
0.000000 0.000000 0.000000 1.000000


Final result:
0.024877 0.004460 -0.695690 148.344013
-0.016139 0.782320 -0.040624 6.244761
0.739755 0.032140 0.013496 14.727247
0.000000 0.000000 0.000000 1.000000


Final result:
-0.059145 0.002514 -0.655897 138.336592
```

```
-0.060014 0.732005 -0.078341 16.850682
0.760242 0.071697 -0.072221 13.858690
0.000000 0.000000 0.000000 1.000000


Final result:
-0.005576 0.002372 -0.640069 133.130452
0.099235 0.795030 -0.079203 0.452020
0.713302 -0.102162 -0.010308 30.335826
0.000000 0.000000 0.000000 1.000000


Final result:
-0.044999 -0.174323 -0.664359 154.104065
-0.078966 0.693748 -0.161826 19.101191
0.790494 0.079337 -0.043089 5.927990
0.000000 0.000000 0.000000 1.000000


Final result:
0.007378 0.000598 -0.636138 136.287499
-0.112960 0.727903 -0.038789 23.305263
0.709966 0.097544 0.023763 4.655212
0.000000 0.000000 0.000000 1.000000


Final result:
0.171618 0.507189 -0.412471 61.856887
0.239628 -0.497981 -0.640753 176.090570
-0.702064 -0.025321 -0.293654 167.373130
0.000000 0.000000 0.000000 1.000000


Final result:
0.035294 -0.041901 0.761947 18.480702
-0.004151 0.832245 -0.148941 8.095265
-0.775820 -0.014747 -0.011064 134.282886
0.000000 0.000000 0.000000 1.000000


Final result:
0.039643 0.007493 -0.687151 147.246063
-0.023898 0.760758 -0.007685 11.161406
0.814471 0.012974 0.060921 3.772544
0.000000 0.000000 0.000000 1.000000


Final result:
```

```
0.105711 -0.046586 0.637453 0.618757
0.100838 0.739400 -0.099455 3.621997
-0.705841 0.105056 0.132778 116.652271
0.000000 0.000000 0.000000 1.000000


Final result:
0.574343 0.032019 0.465711 -22.159201
0.056326 -0.717885 -0.085035 163.026161
0.453002 0.031203 -0.493690 86.121538
0.000000 0.000000 0.000000 1.000000


Final result:
0.014863 -0.104775 -0.687844 151.830228
-0.056502 0.737993 -0.117888 18.202740
0.738699 0.074785 0.017043 7.536736
0.000000 0.000000 0.000000 1.000000


Final result:
0.010249 0.253231 -0.777672 104.775289
-0.055661 0.821992 0.213844 -9.316393
0.751361 0.061015 0.049647 9.275378
0.000000 0.000000 0.000000 1.000000


Final result:
0.632711 -0.039161 0.320875 -1.397056
0.029974 0.729111 -0.052193 5.295600
-0.307703 -0.002809 0.627113 50.821602
0.000000 0.000000 0.000000 1.000000


Final result:
-0.020619 -0.118548 -0.719037 157.761889
-0.026729 0.796516 -0.132828 14.649007
0.832395 0.017883 -0.000159 6.709993
0.000000 0.000000 0.000000 1.000000


Final result:
0.044781 0.110795 -0.666435 130.052780
-0.031557 0.719563 0.110744 -0.685600
0.816432 0.051642 0.038141 -1.594557
0.000000 0.000000 0.000000 1.000000
```

```
Final result:
0.167107 0.056010 -0.830132 146.162734
-0.031321 0.874155 0.052729 -2.151453
0.777234 0.003902 0.146370 13.877170
0.000000 0.000000 0.000000 1.000000


Final result:
-0.033021 -0.274540 -0.614606 164.201614
0.075222 0.652605 -0.347641 38.747063
0.767328 -0.088963 0.006499 18.984753
0.000000 0.000000 0.000000 1.000000


Final result:
-0.023953 -0.153943 -0.688901 152.227754
-0.037181 0.715312 -0.130874 22.099060
0.817017 0.036417 -0.021557 8.356155
0.000000 0.000000 0.000000 1.000000


Final result:
-0.035608 -0.005600 -0.745081 142.232273
-0.006310 0.753687 -0.004117 4.664528
0.766500 0.000731 -0.036663 18.821504
0.000000 0.000000 0.000000 1.000000


Final result:
-0.094304 -0.008380 -0.717882 150.867897
-0.032717 0.766461 -0.083384 10.953551
0.804151 0.017982 -0.031132 9.359940
0.000000 0.000000 0.000000 1.000000


Final result:
0.055292 0.130776 -0.731839 126.042429
-0.034132 0.766335 0.095253 -1.018578
0.778449 0.045387 0.068090 -3.246869
0.000000 0.000000 0.000000 1.000000


Final result:
-0.019567 0.004377 -0.675173 137.380483
0.034832 0.731688 0.000174 3.555001
0.736010 -0.025840 -0.017194 18.299058
0.000000 0.000000 0.000000 1.000000
```

```
Final result:
0.042300 -0.007500 -0.767653 145.828534
0.005914 0.735518 -0.010644 5.058652
0.843872 0.000283 0.043282 0.955158
0.000000 0.000000 0.000000 1.000000


Final result:
0.684496 -0.197646 -0.079645 46.161787
0.199396 0.723594 -0.045379 -13.413937
0.146723 0.004756 0.733230 -10.608080
0.000000 0.000000 0.000000 1.000000


Final result:
0.111246 -0.121023 -0.704291 140.789573
0.034747 0.814675 -0.131077 7.824518
0.782069 -0.002103 0.094645 12.974397
0.000000 0.000000 0.000000 1.000000


Final result:
0.456354 -0.047578 -0.764158 129.055958
0.127782 0.837934 -0.047760 -13.043259
0.699411 -0.092097 0.511172 -16.040647
0.000000 0.000000 0.000000 1.000000


Final result:
-0.027057 -0.005356 -0.695397 135.879222
0.020205 0.776084 -0.060355 3.000969
0.727639 -0.006313 -0.048549 20.114278
0.000000 0.000000 0.000000 1.000000


Final result:
0.069556 -0.108746 -0.678223 135.484559
-0.069827 0.737755 -0.161981 21.194128
0.797465 0.078966 0.048386 2.895462
0.000000 0.000000 0.000000 1.000000


Final result:
0.028948 0.018890 -0.727078 133.261466
-0.024208 0.762970 -0.027300 12.301800
0.741982 0.040344 0.094185 9.451470
0.000000 0.000000 0.000000 1.000000
```

```
Final result:
-0.003527 0.001049 -0.717134 139.148773
-0.006588 0.737722 -0.048707 10.519392
0.806683 0.013561 0.001391 9.675962
0.000000 0.000000 0.000000 1.000000


Final result:
-0.163830 0.072365 -0.685199 126.821367
0.043914 0.720272 -0.011451 2.680912
0.755363 -0.041367 -0.166734 30.417591
0.000000 0.000000 0.000000 1.000000


Final result:
0.030905 0.018887 -0.682855 137.130221
0.021198 0.778983 -0.016299 5.774340
0.715586 -0.007524 0.044945 17.103042
0.000000 0.000000 0.000000 1.000000


Final result:
-0.080950 0.015939 -0.697104 133.778028
-0.037291 0.682498 -0.006234 15.404702
0.771388 0.059959 -0.016618 3.989694
0.000000 0.000000 0.000000 1.000000


Final result:
0.127350 -0.228188 -0.686194 137.419630
-0.104878 0.687438 -0.313094 43.212757
0.733326 0.140878 0.092538 -6.833525
0.000000 0.000000 0.000000 1.000000


Final result:
0.001174 0.000415 -0.705875 142.486326
-0.117334 0.754669 -0.052414 16.666359
0.777155 0.135732 0.009138 0.367736
0.000000 0.000000 0.000000 1.000000


Final result:
-0.014866 -0.204271 -0.701759 155.510358
-0.016097 0.733112 -0.229812 29.337773
0.682410 0.023518 -0.007371 17.424125
```

```
0.000000 0.000000 0.000000 1.000000


Final result:
-0.056198 0.007790 -0.757517 137.567230
-0.042100 0.757583 -0.112073 11.435212
0.808506 0.046439 -0.040666 9.007487
0.000000 0.000000 0.000000 1.000000


Final result:
0.673326 0.051996 -0.085848 24.523405
-0.036345 0.704997 0.176875 -8.125800
0.149910 -0.205814 0.705162 19.916467
0.000000 0.000000 0.000000 1.000000


Final result:
0.238863 0.513323 -0.295139 37.235170
0.119715 -0.364400 -0.675459 177.900420
-0.685236 0.138742 -0.235850 146.919957
0.000000 0.000000 0.000000 1.000000


Final result:
-0.018648 -0.294711 -0.618956 171.998643
-0.058317 0.660707 -0.302389 49.824007
0.741531 0.044417 -0.038311 15.775670
0.000000 0.000000 0.000000 1.000000


Final result:
0.095880 -0.185483 -0.653243 137.467243
0.061528 0.754319 -0.226226 20.149954
0.765638 -0.050233 0.105443 6.745923
0.000000 0.000000 0.000000 1.000000


Final result:
0.022354 -0.004453 -0.703788 138.559739
0.004431 0.772781 0.003039 -2.890363
0.758194 -0.018694 0.043969 11.297594
0.000000 0.000000 0.000000 1.000000


Final result:
0.042530 0.005690 -0.638827 127.440736
-0.084167 0.738955 -0.046553 15.492238
```

```
0.753578 0.075957 0.033149 3.429785
0.000000 0.000000 0.000000 1.000000


Final result:
-0.139261 0.043204 -0.713003 137.730206
0.040406 0.712050 -0.020110 5.734775
0.802989 -0.018336 -0.124768 18.482561
0.000000 0.000000 0.000000 1.000000


Final result:
0.058948 -0.065012 -0.668393 151.387967
-0.084959 0.752992 -0.048479 12.185184
0.746029 0.088934 0.048181 -1.726612
0.000000 0.000000 0.000000 1.000000


Final result:
0.005117 -0.047571 -0.725277 137.356969
0.011817 0.722035 -0.080890 11.761375
0.720434 -0.003386 0.015720 17.358024
0.000000 0.000000 0.000000 1.000000


Final result:
-0.024898 -0.022548 -0.739751 151.808695
-0.097725 0.792346 -0.065933 15.496571
0.731356 0.106923 -0.040800 8.508114
0.000000 0.000000 0.000000 1.000000


Final result:
0.006959 0.002985 -0.743545 136.394325
0.034532 0.807100 -0.090970 -3.007761
0.828660 -0.029454 0.026189 10.752342
0.000000 0.000000 0.000000 1.000000


Final result:
0.684570 -0.133386 0.328476 3.326473
0.130359 0.738639 -0.069035 -2.870298
-0.311859 0.045151 0.679954 42.383802
0.000000 0.000000 0.000000 1.000000


Final result:
-0.043095 -0.211036 -0.694409 158.221493
```

```
0.010493 0.747022 -0.233649 16.477126
0.782834 -0.016445 -0.026427 16.024056
0.000000 0.000000 0.000000 1.000000


Final result:
-0.072509 -0.059248 -0.670063 135.767024
-0.141742 0.779613 -0.068102 21.092334
0.713100 0.155524 -0.081799 11.697903
0.000000 0.000000 0.000000 1.000000


Final result:
-0.025677 -0.122781 -0.648301 158.183875
0.124240 0.720134 -0.116646 6.508607
0.795147 -0.152170 -0.013718 25.984897
0.000000 0.000000 0.000000 1.000000


Final result:
0.000536 -0.000149 -0.695490 140.570695
0.051946 0.800891 -0.048724 6.738908
0.745475 -0.052421 -0.004072 18.056939
0.000000 0.000000 0.000000 1.000000


Final result:
-0.109979 0.098916 -0.700750 140.367150
-0.023351 0.722576 0.062663 2.748039
0.761527 0.037622 -0.080696 16.560830
0.000000 0.000000 0.000000 1.000000


Final result:
0.003178 -0.235580 -0.640161 157.230431
-0.017850 0.724967 -0.283331 37.496369
0.793686 0.001566 -0.018047 12.571557
0.000000 0.000000 0.000000 1.000000


Final result:
-0.190261 0.072456 -0.619935 154.773824
0.006926 0.755435 0.051737 -3.319584
0.767325 0.025952 -0.173670 30.217615
0.000000 0.000000 0.000000 1.000000


Final result:
```

```
-0.000563 -0.000192 -0.696315 139.316075
-0.096939 0.735157 -0.035076 14.880966
0.778957 0.109245 -0.000357 5.789122
0.000000 0.000000 0.000000 1.000000


Final result:
-0.046183 -0.001431 -0.672497 148.834015
-0.024356 0.767174 -0.044769 7.440775
0.783621 0.025708 -0.037950 12.384301
0.000000 0.000000 0.000000 1.000000


Final result:
0.033279 0.009380 -0.764737 126.000875
-0.036411 0.805219 -0.055847 14.442616
0.813960 0.034303 0.044749 -1.594940
0.000000 0.000000 0.000000 1.000000
```

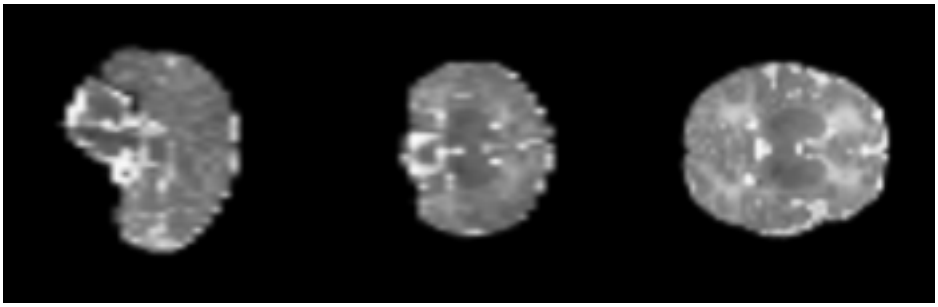
[ ]:
```
images = []
image_tensors = []
t2_file_list = os.listdir('/media/hdd/viscent/SR-UNet/inference/
 ↪Inference_FLYWHEEL_BROWN/input_reg/')
t2_file_list = [os.path.join('/media/hdd/viscent/SR-UNet/inference/
 ↪Inference_FLYWHEEL_BROWN/input_reg/',x) for x in t2_file_list]
for t2_file in t2_file_list:
    images.append(sitk.ReadImage(t2_file))
    subject = tio.Subject(t2=tio.ScalarImage(t2_file))
    transform_1 = tio.Compose([
        tio.transforms.RescaleIntensity(0., 1.),
        tio.transforms.ToCanonical(),
        tio.transforms.Resample((1.,1.,1.)),
    ])
    subject = transform_1(subject)
    edge_max = max(subject.t2.data.shape)
    padding = ((edge_max - subject.t2.data.shape[1]) // 2,
               (edge_max - subject.t2.data.shape[2]) // 2,
                   (edge_max - subject.t2.data.shape[3]) // 2)
    transform_2 = tio.Compose([
        tio.Pad(padding),
        tio.transforms.Resize((160,160,160)),
    ])
    subject = transform_2(subject)
    image_tensor = subject.t2.data.unsqueeze(0).float()
    image_tensors.append(image_tensor)
```

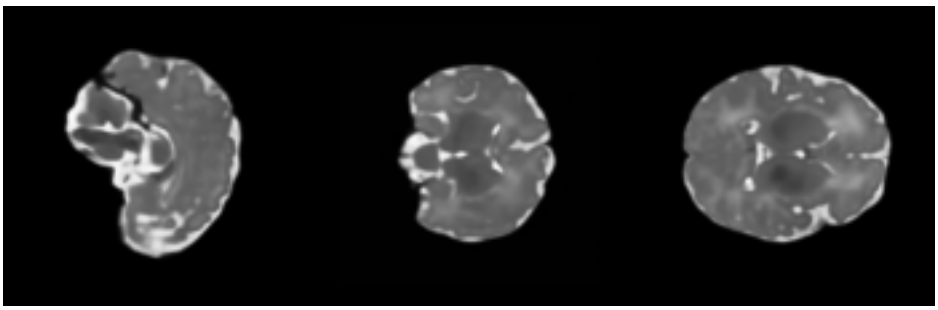
```
/home/viscent/anaconda3/envs/bunet/lib/python3.10/site-
```

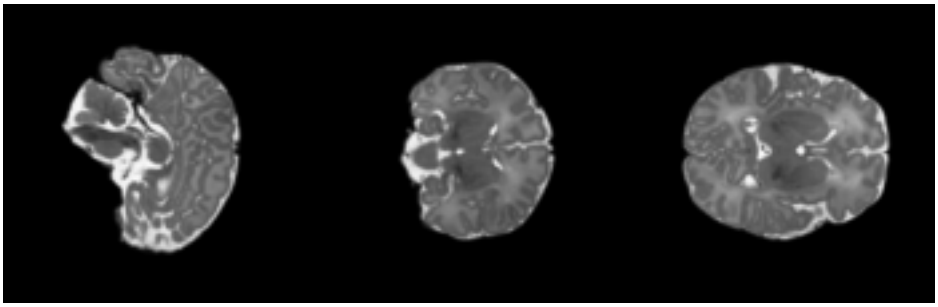
```
packages/torchio/transforms/preprocessing/intensity/rescale.py:99:
RuntimeWarning:

Rescaling image "t2" not possible because all the intensity values are the same
```

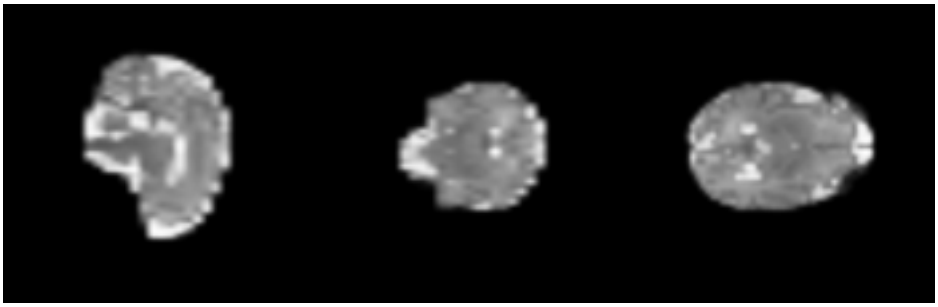
1.2 Sanity Check on Training Data

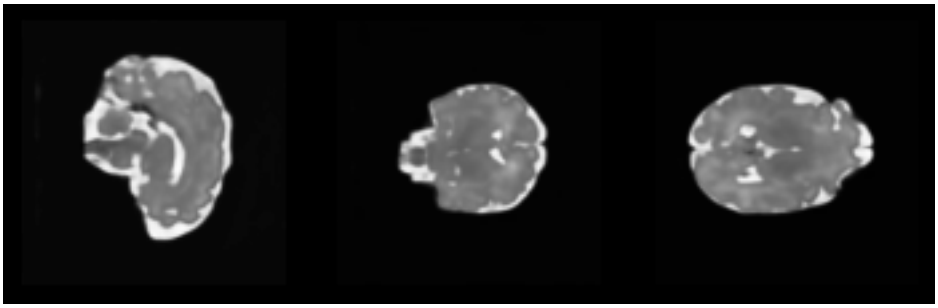
[ ]:
```
for image_tensor, target_tensor in dhcp_val_loader:
    image_tensor = image_tensor.cuda()
    target_tensor = target_tensor.cuda()
    output_tensor = dhcp_t2_1500(image_tensor).detach().cpu().numpy()
    display_multiplanar_center(image_tensor.detach().cpu().numpy()[0,0])
    display_multiplanar_center(output_tensor[0,0])
    display_multiplanar_center(target_tensor.detach().cpu().numpy()[0,0])
```

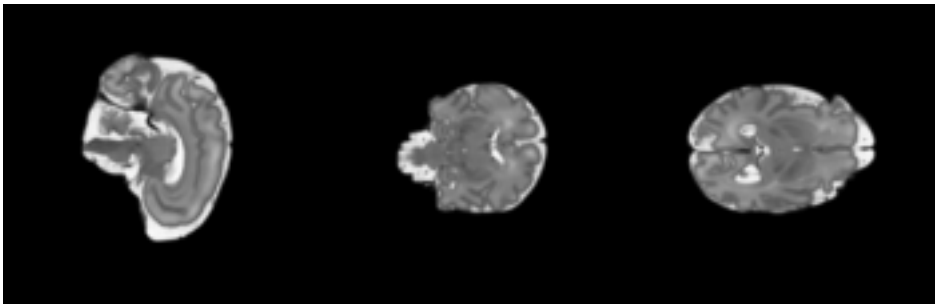


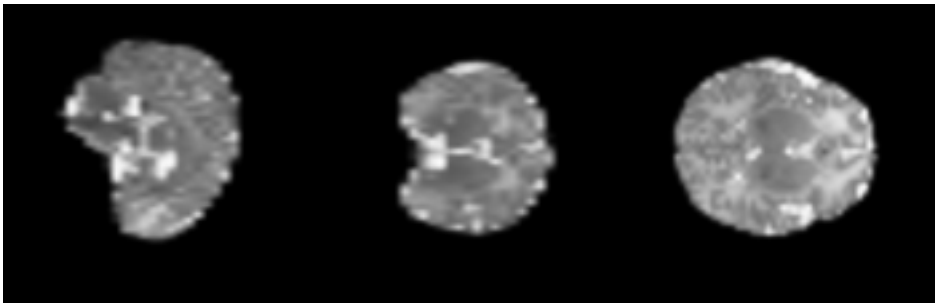


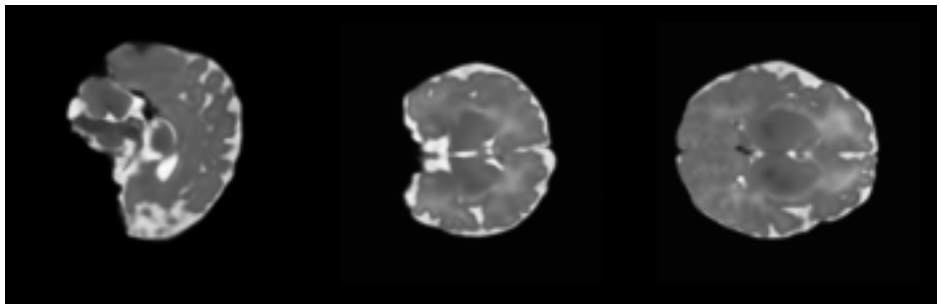


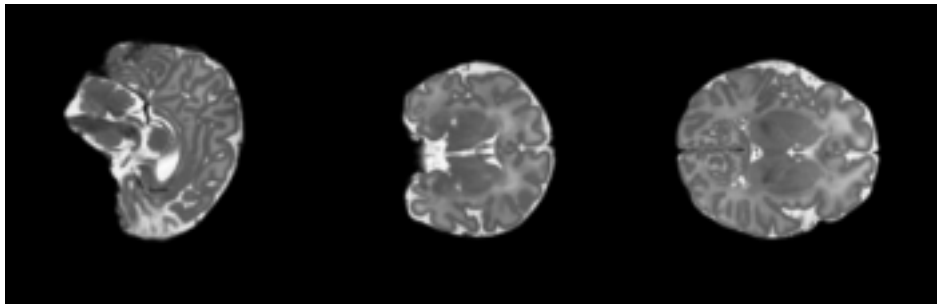


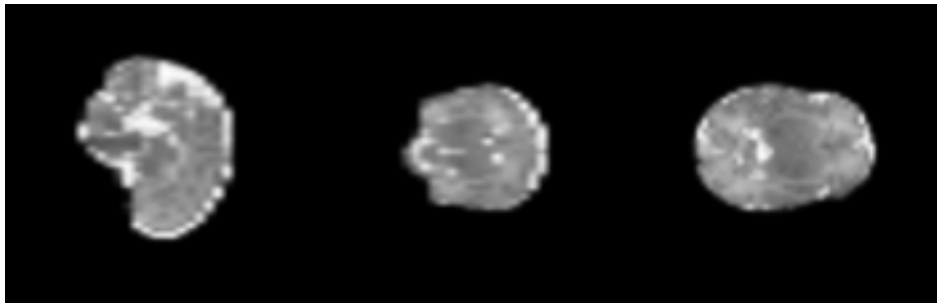


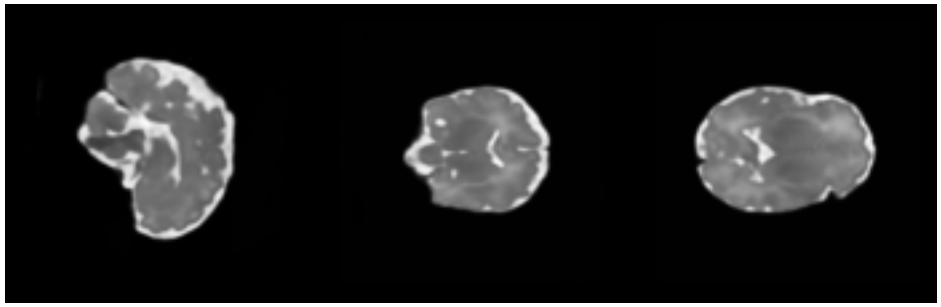


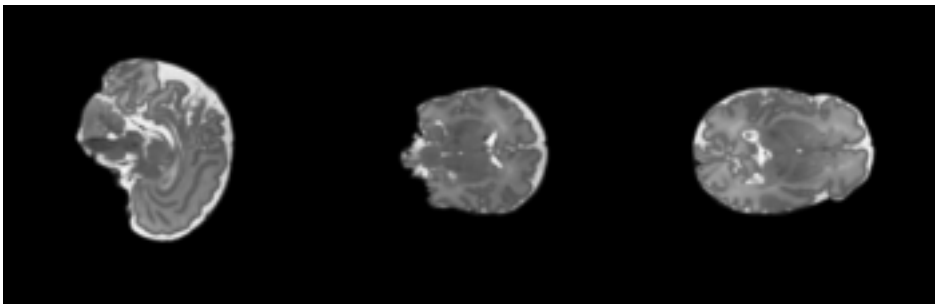


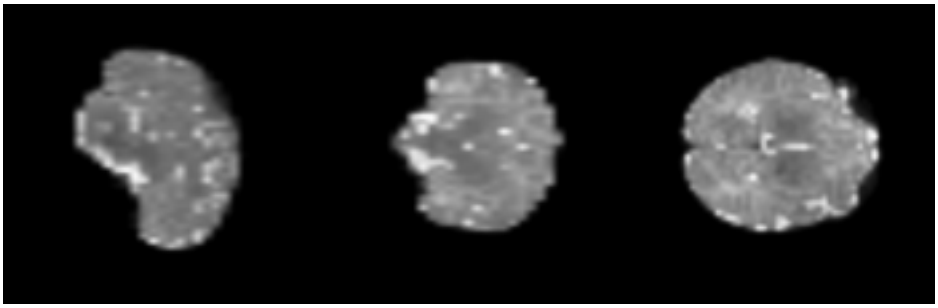


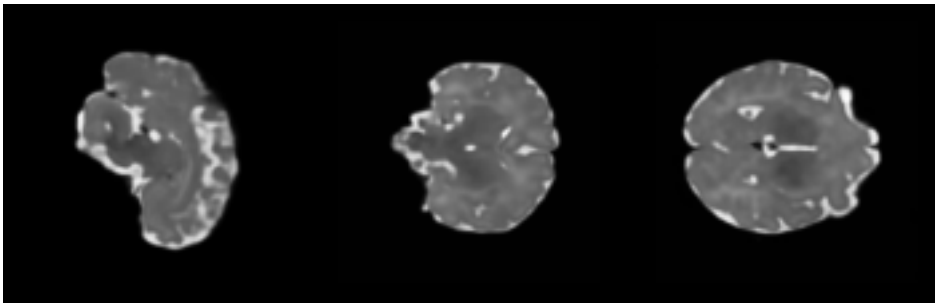


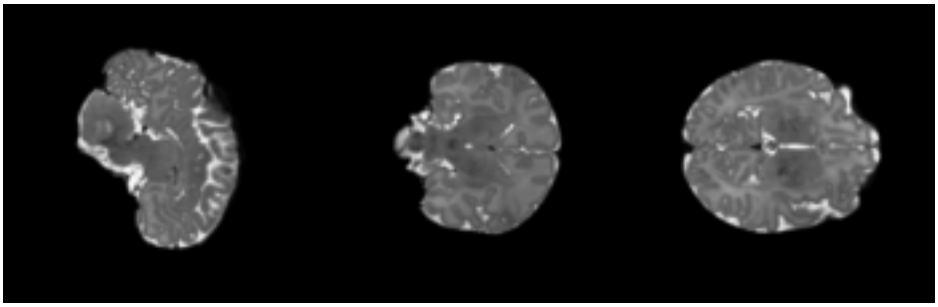


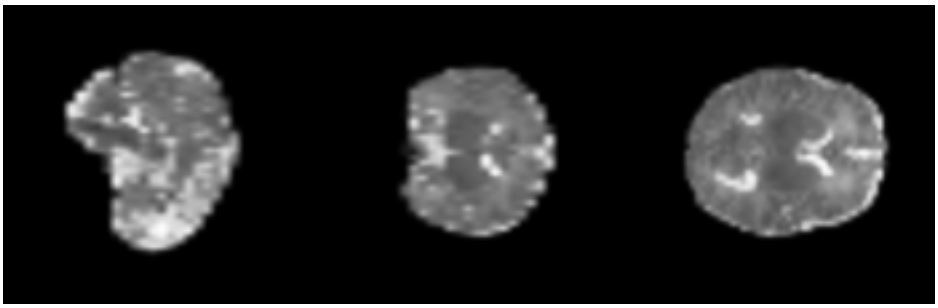


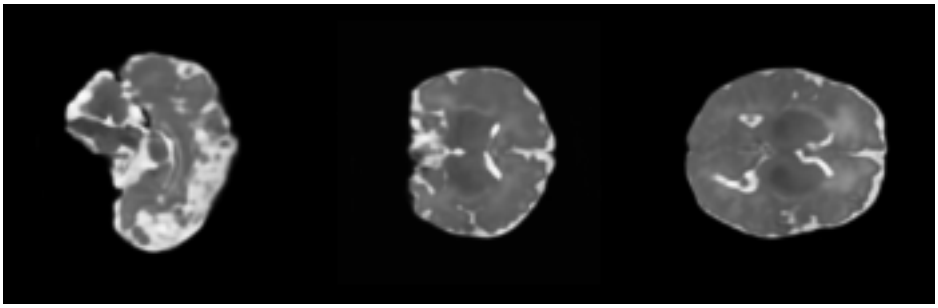


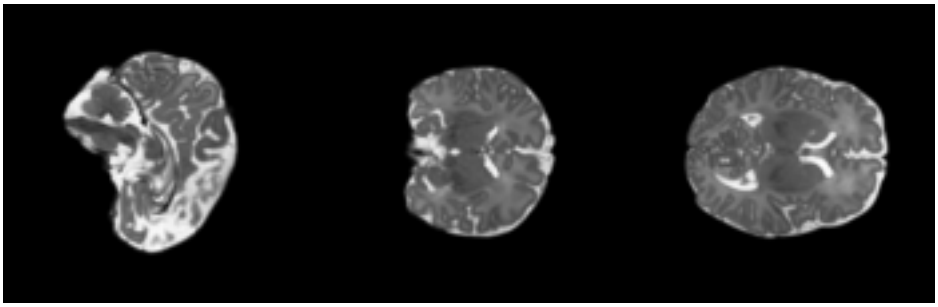


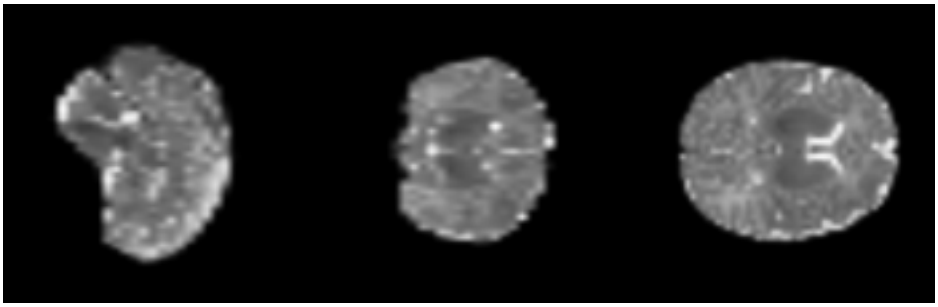


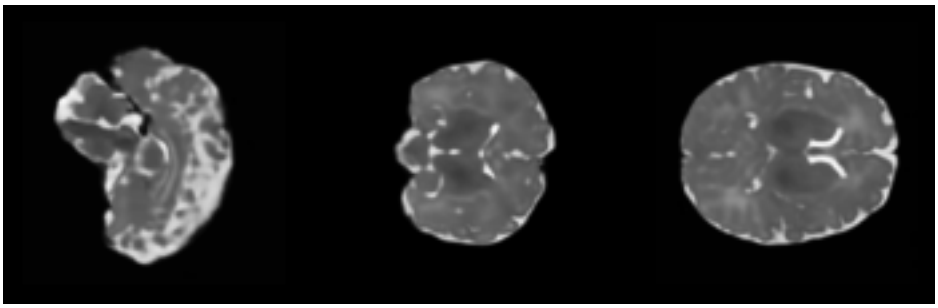


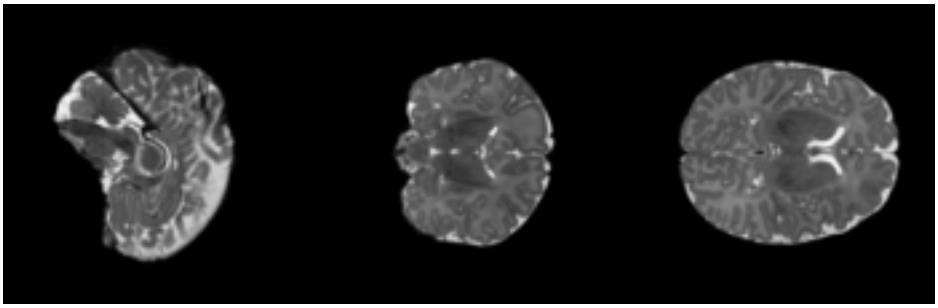


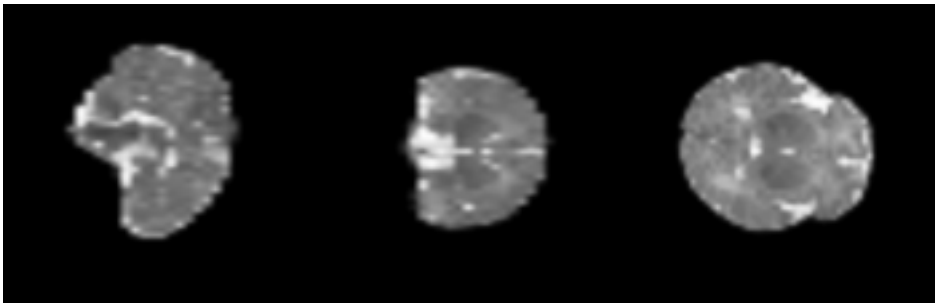


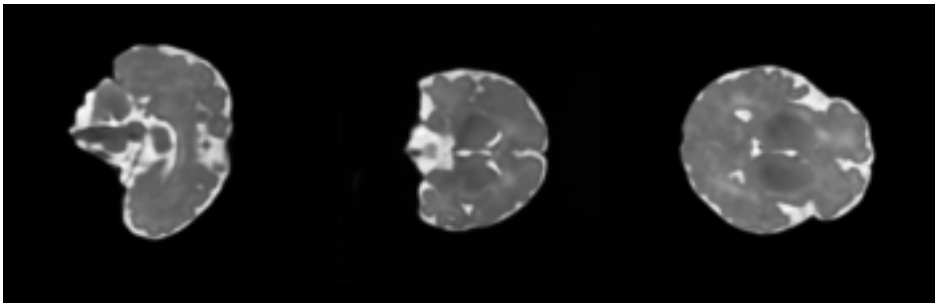


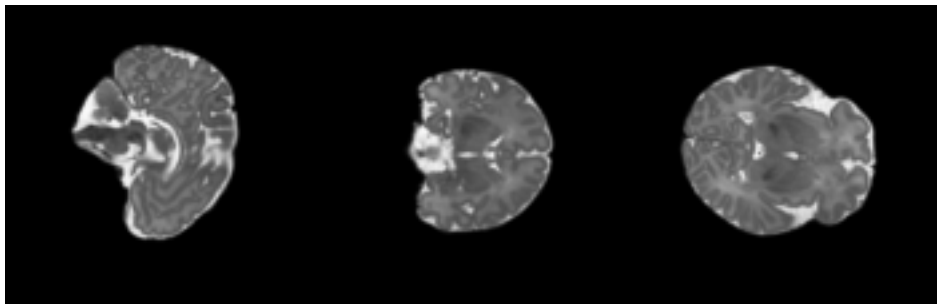


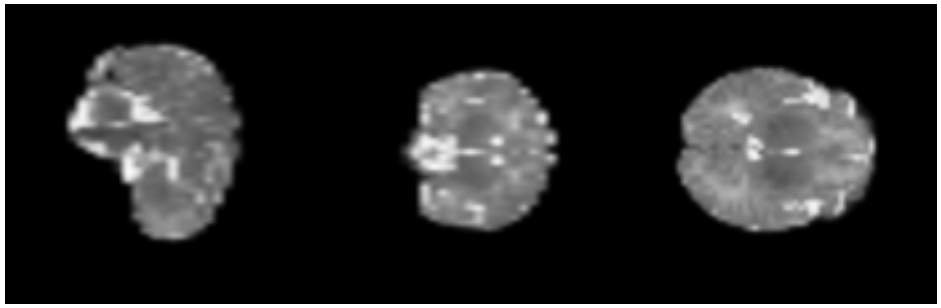


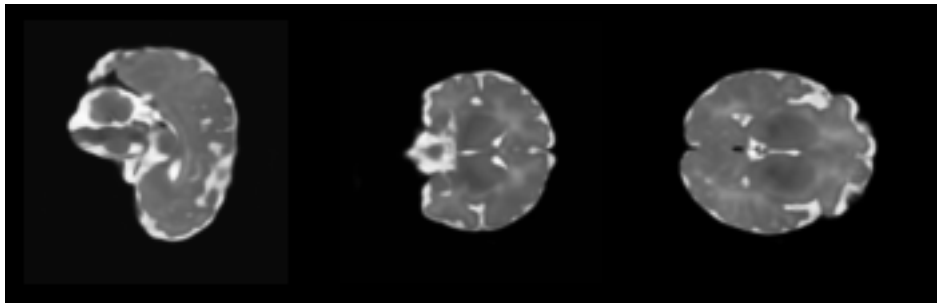


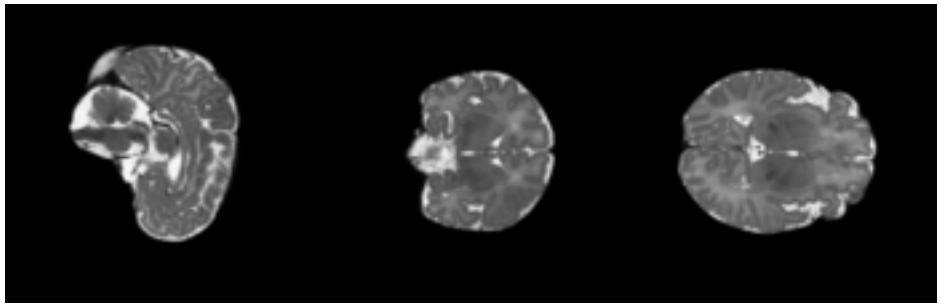


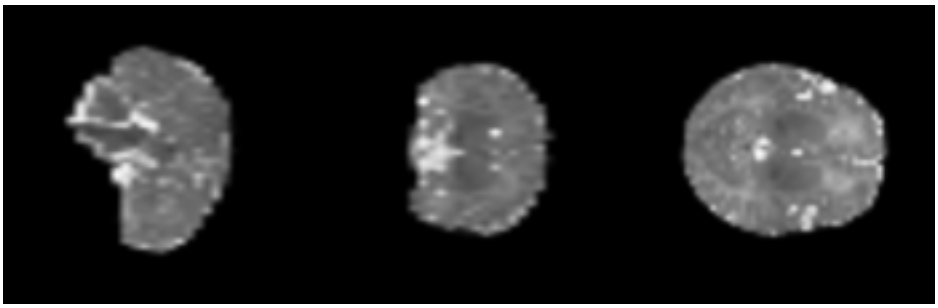


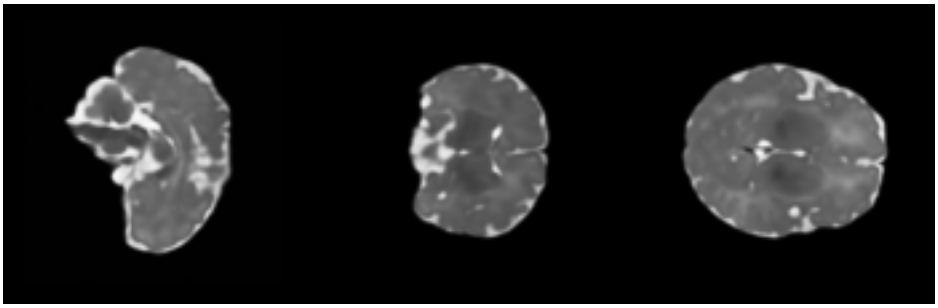


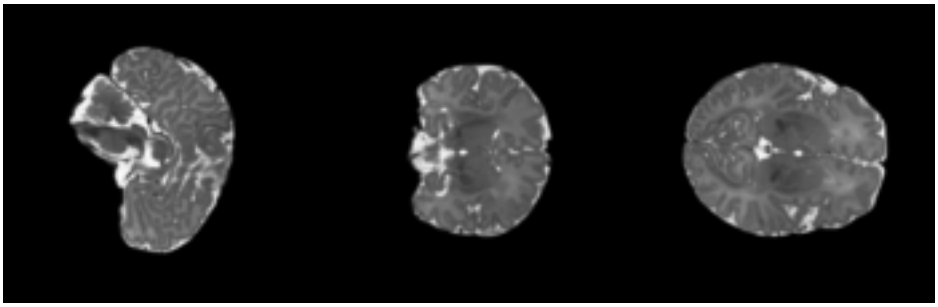


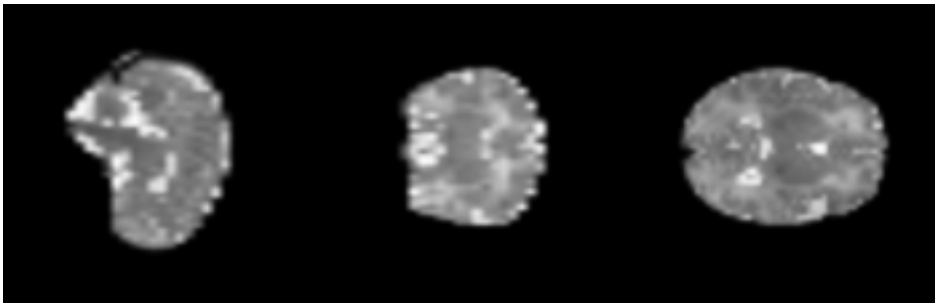


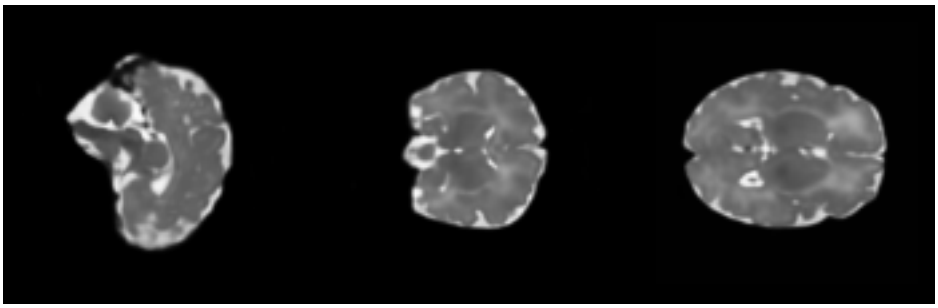


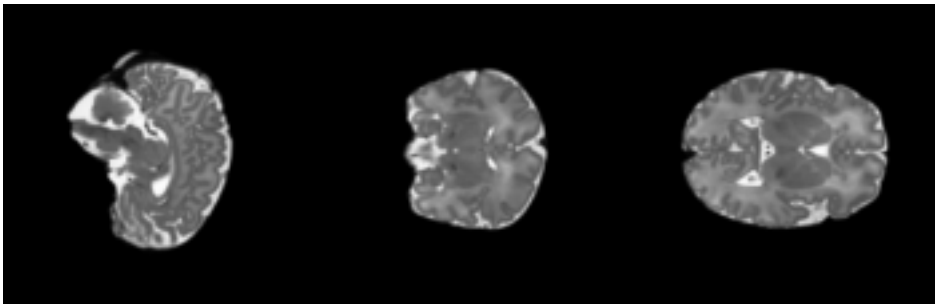


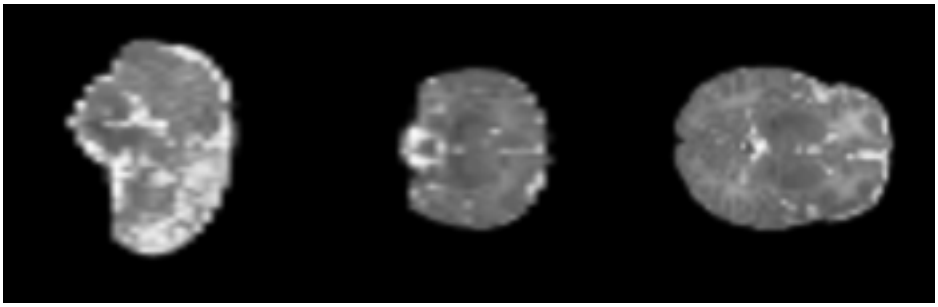


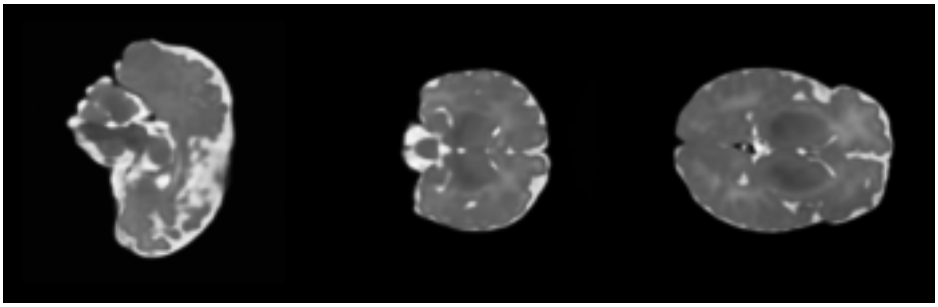


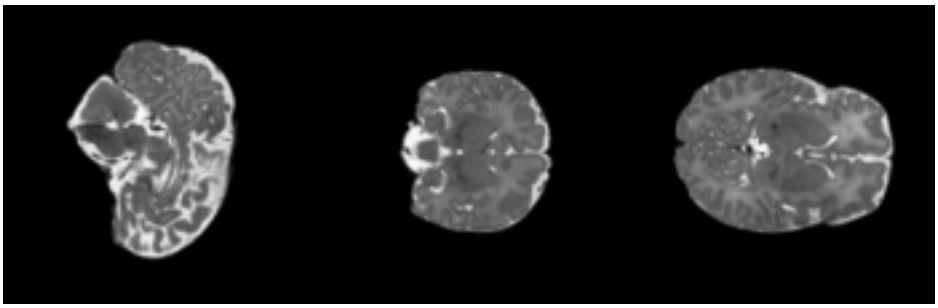


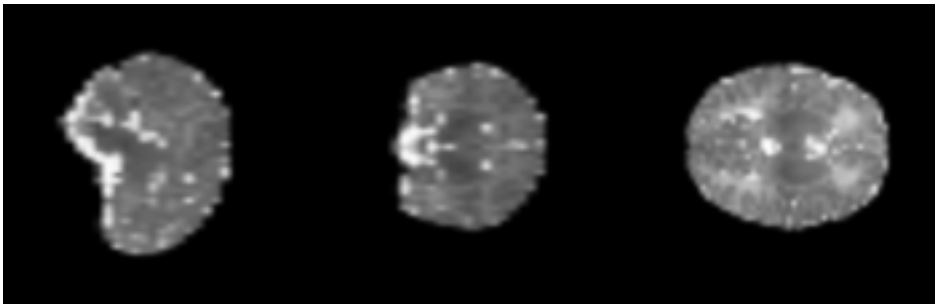


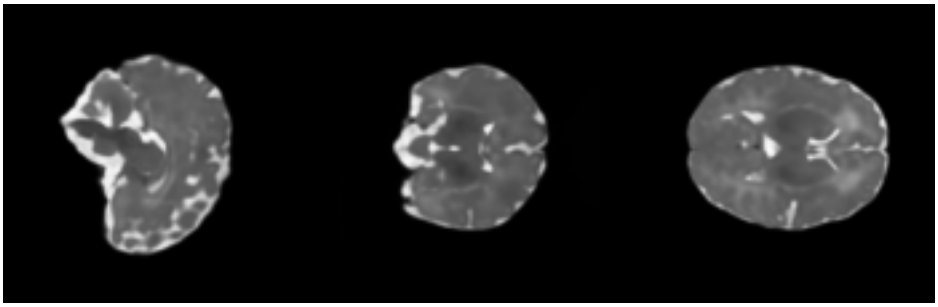


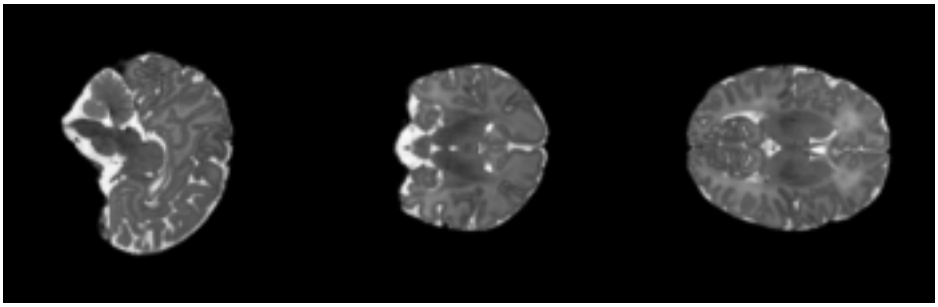


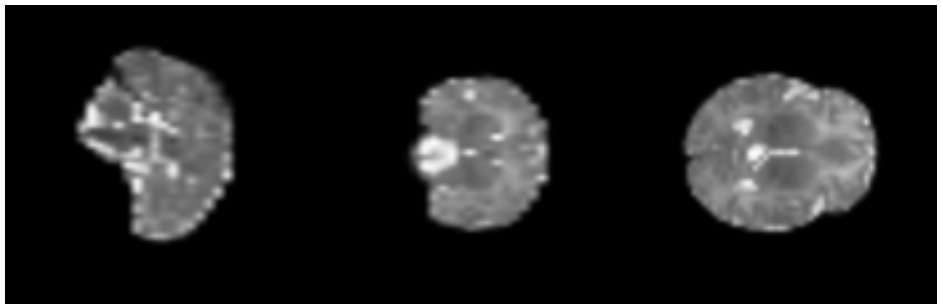


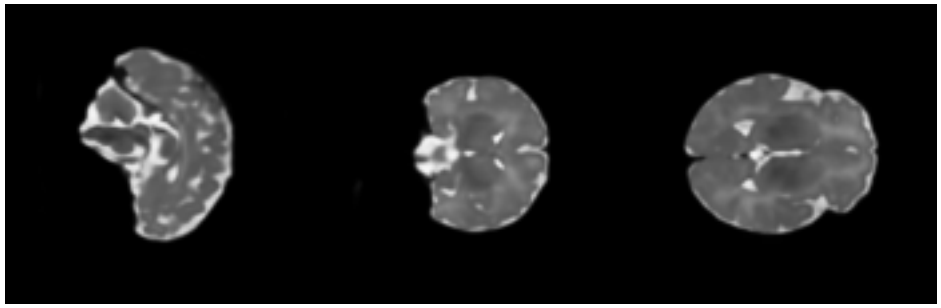


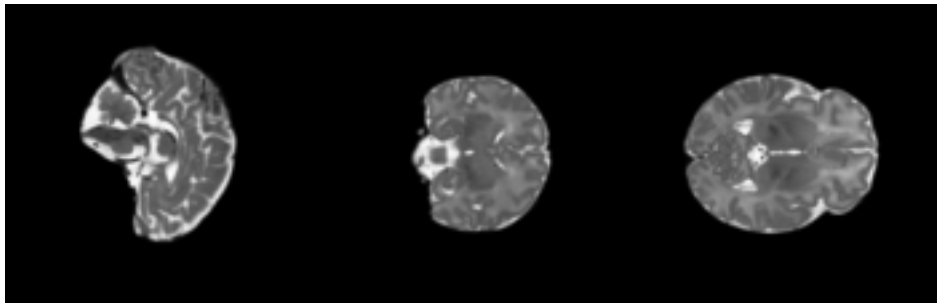


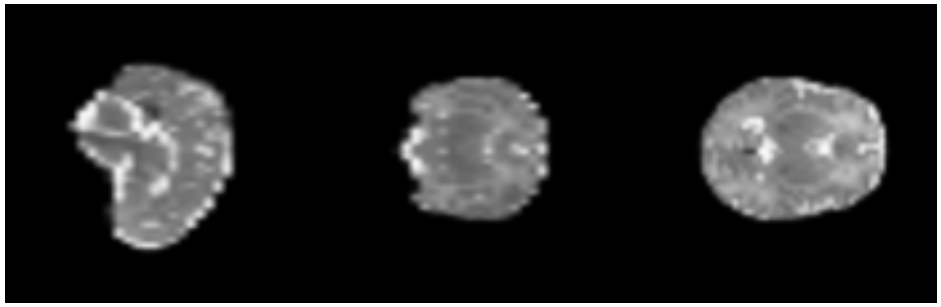


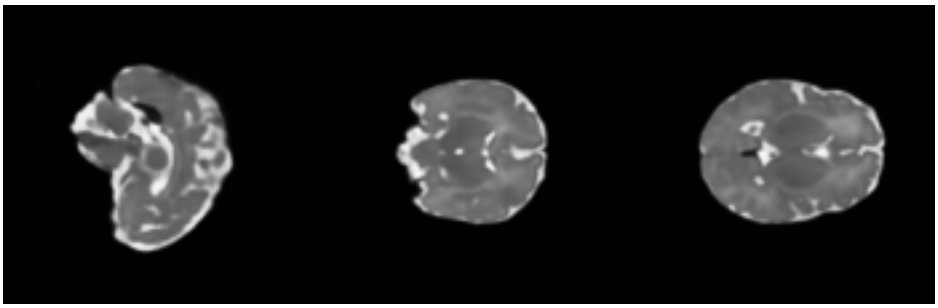


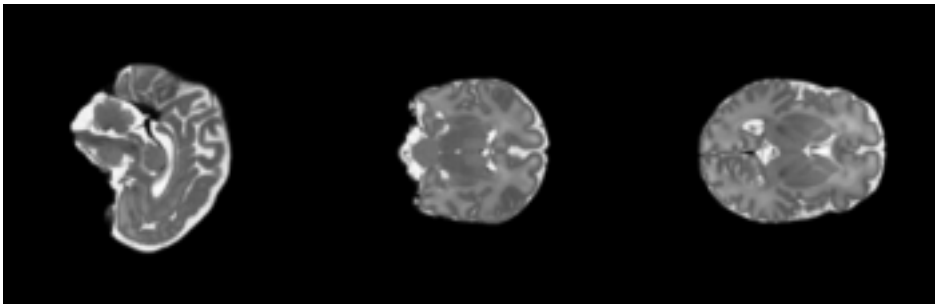


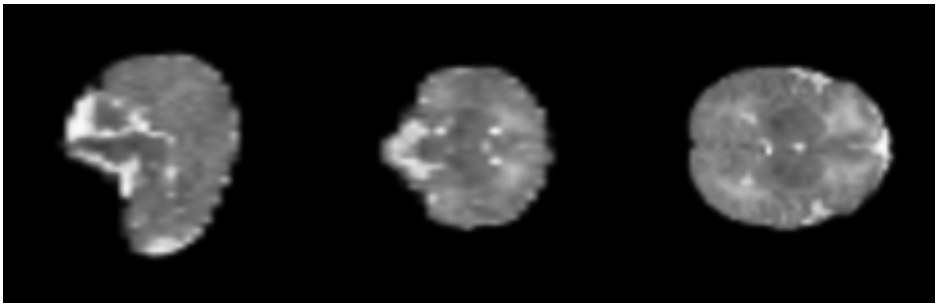


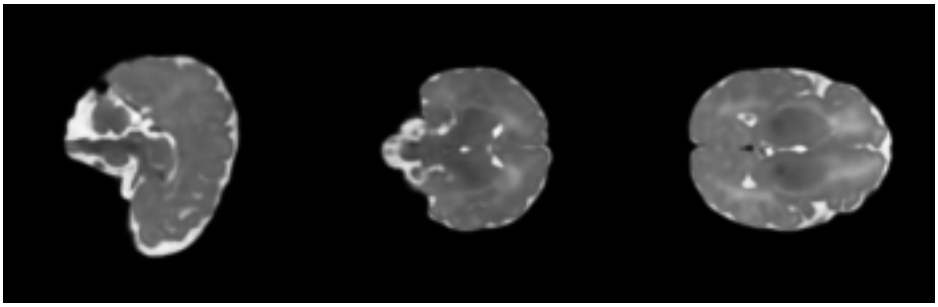


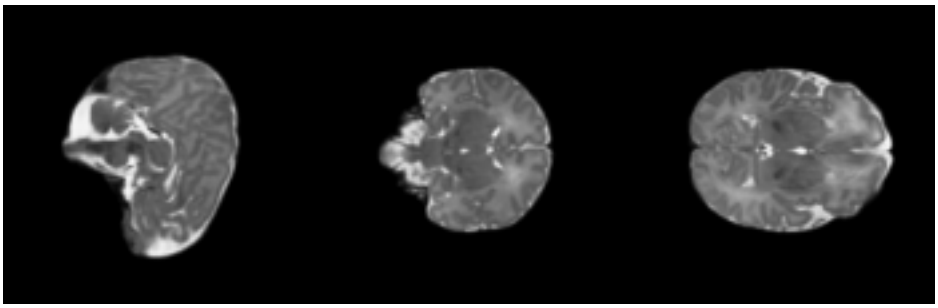


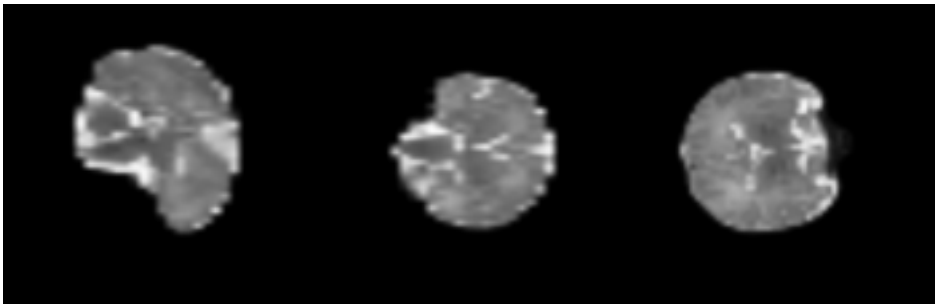


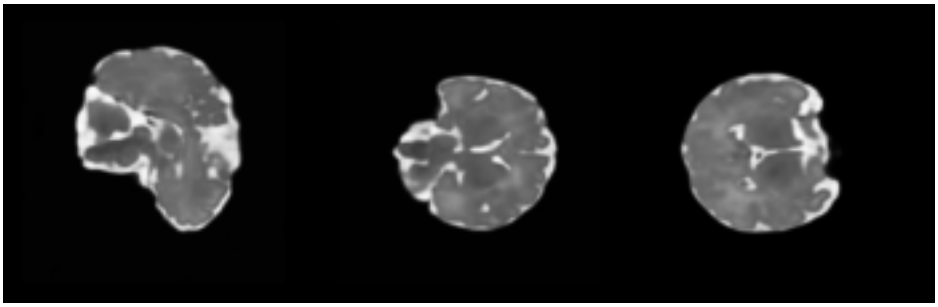


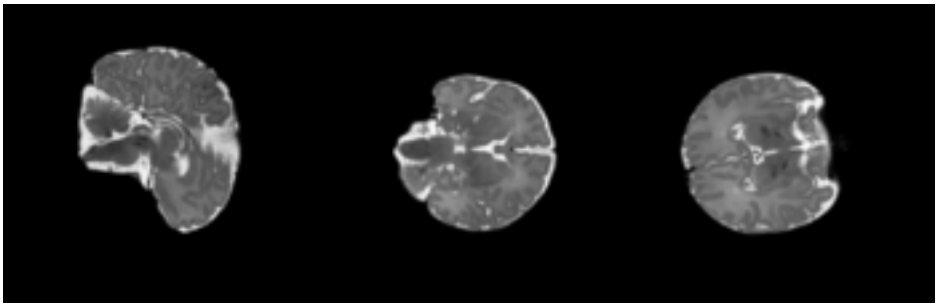


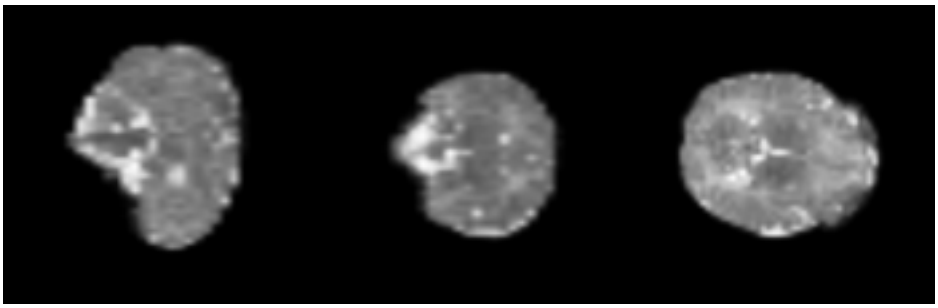


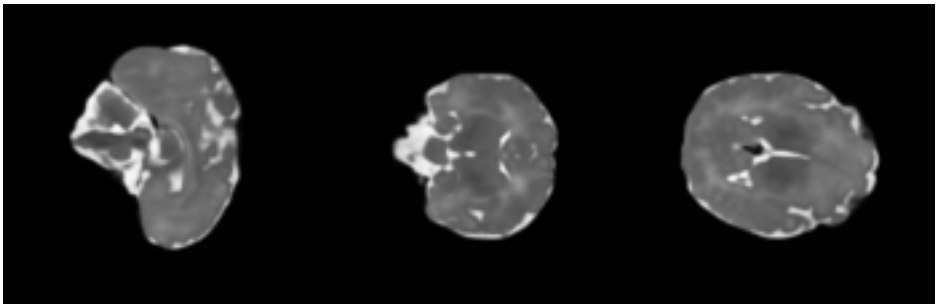


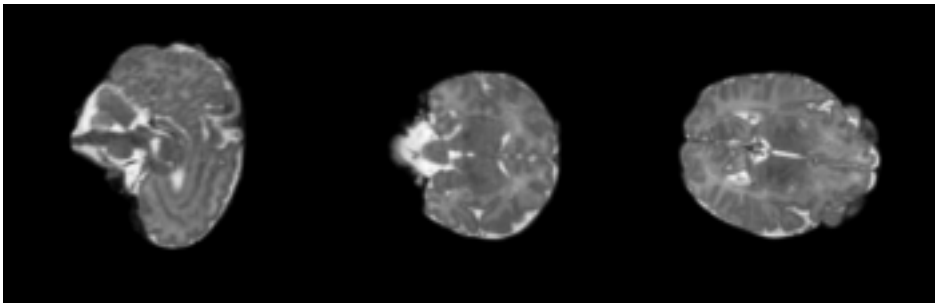


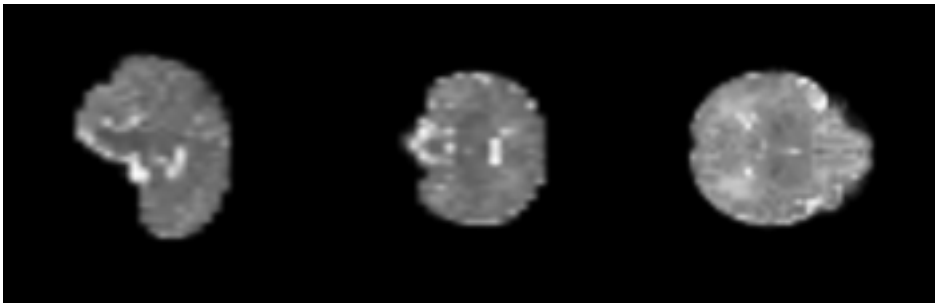


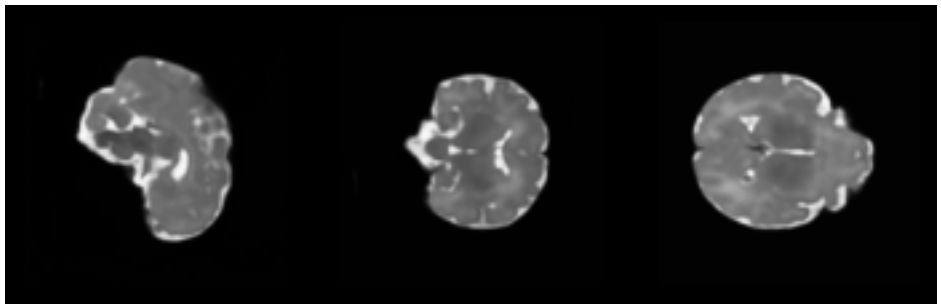


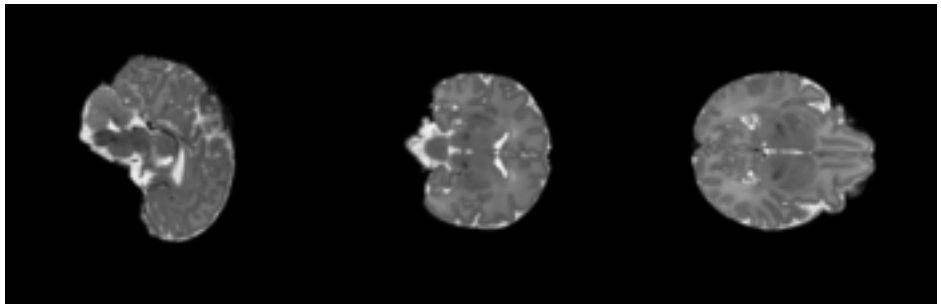


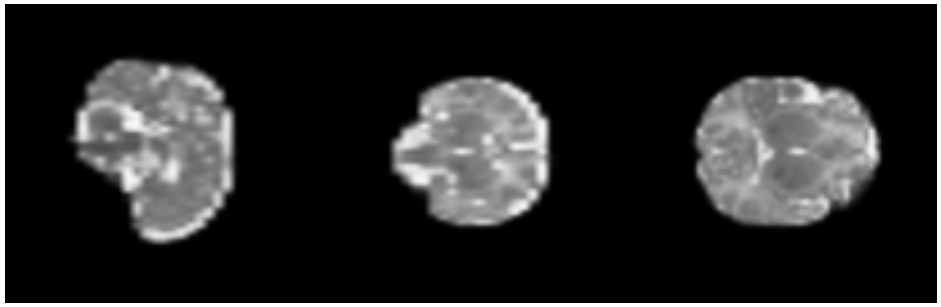


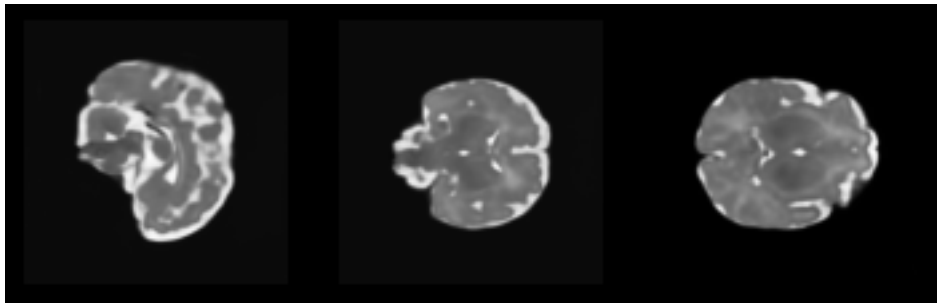


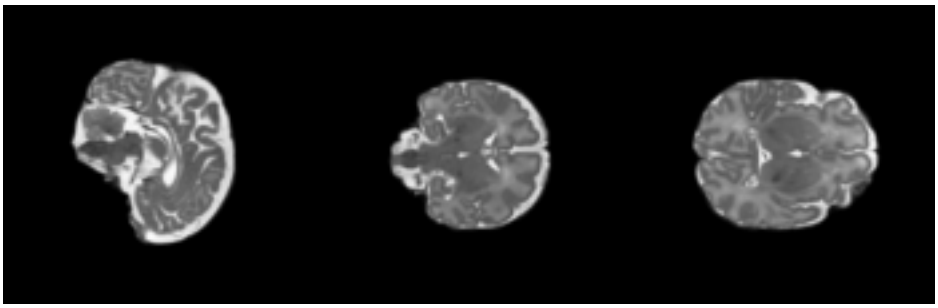












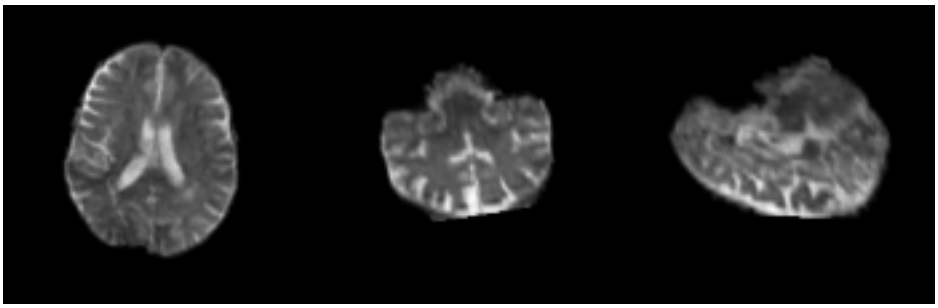
1.3 Testing on FLYWHEEL_BROWN data

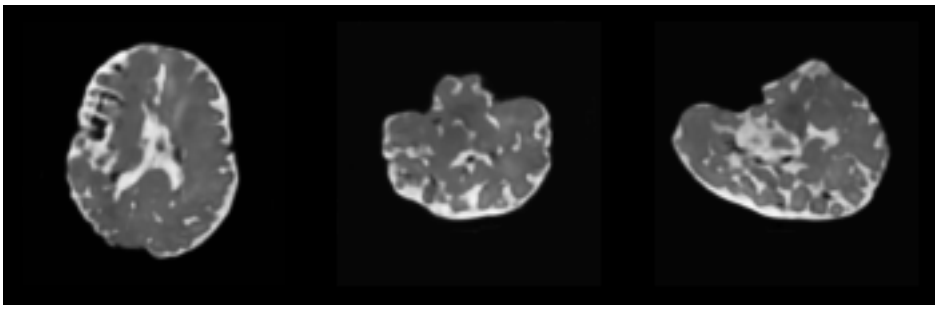
[ ]:
```
import os


output_tensors_dhcp_1500 = []
output_tensors_dhcp_1500_aug = []
output_tensors_hcp_140_aug = []

data_root = '/media/hdd/viscent/FLYWHEEL_BROWN/'

outdir_dhcp_1500 = os.path.join(data_root, 'output_dhcp_1500')
outdir_dhcp_1500_aug = os.path.join(data_root, 'output_dhcp_1500_aug')
outdir_hcp_140_aug = os.path.join(data_root, 'output_hcp_140_aug')
# os.mkdir(outdir_dhcp_1500)
# os.mkdir(outdir_dhcp_1500_aug)
# os.mkdir(outdir_hcp_140_aug)

i=0
for image_tensor in tqdm(image_tensors):
    print('##################{}##################'.format(i))
    output_tensor_dhcp_1500 = dhcp_t2_1500(image_tensor.cuda()).cpu().detach()
    output_tensor_dhcp_1500_aug = dhcp_t2_1500_aug(image_tensor.cuda()).cpu().
 ↪detach()
    output_tensor_hcp_140_aug = hcp_t2_140_aug(image_tensor.cuda()).cpu().
 ↪detach()
    output_tensors_dhcp_1500.append(output_tensor_dhcp_1500)
    output_tensors_dhcp_1500_aug.append(output_tensor_dhcp_1500_aug)
    output_tensors_hcp_140_aug.append(output_tensor_hcp_140_aug)

    display_multiplanar_center(image_tensor.cpu().numpy()[0,0])
    display_multiplanar_center(output_tensor_dhcp_1500[0,0])
    # display_multiplanar_center(output_tensor_dhcp_1500_aug[0,0])
    # display_multiplanar_center(output_tensor_hcp_140_aug[0,0])
```

```
    sitk.WriteImage(sitk.GetImageFromArray(output_tensor_dhcp_1500[0,0].
↪numpy()), os.path.join(outdir_dhcp_1500, 'output.nii.gz'))
    sitk.WriteImage(sitk.GetImageFromArray(output_tensor_dhcp_1500_aug[0,0].
↪numpy()), os.path.join(outdir_dhcp_1500_aug, 'output.nii.gz'))
    sitk.WriteImage(sitk.GetImageFromArray(output_tensor_hcp_140_aug[0,0].
↪numpy()), os.path.join(outdir_hcp_140_aug, 'output.nii.gz'))
    print('####################################')
    i+=1
```

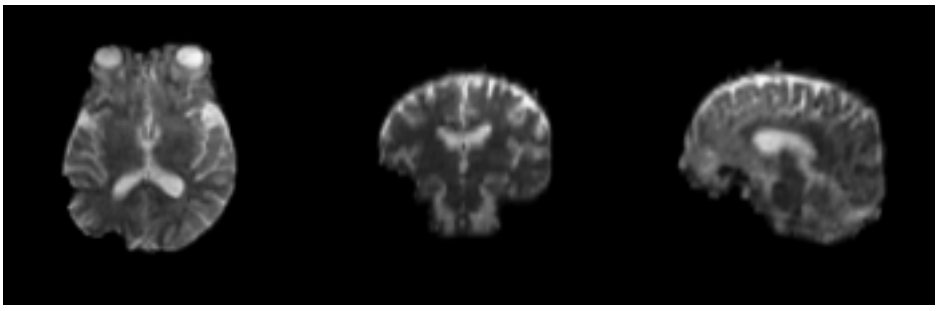
```
  0%|          | 0/113 [00:00<?, ?it/s]
```

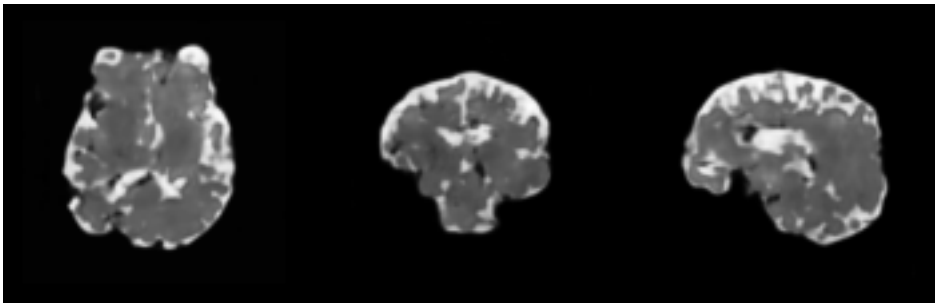
```
##################0#################
```



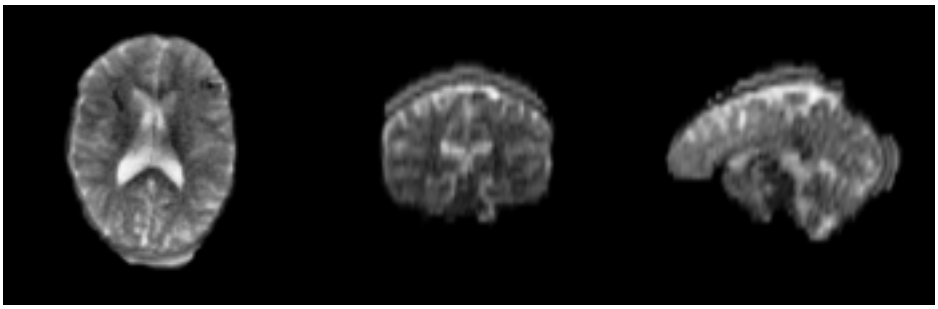


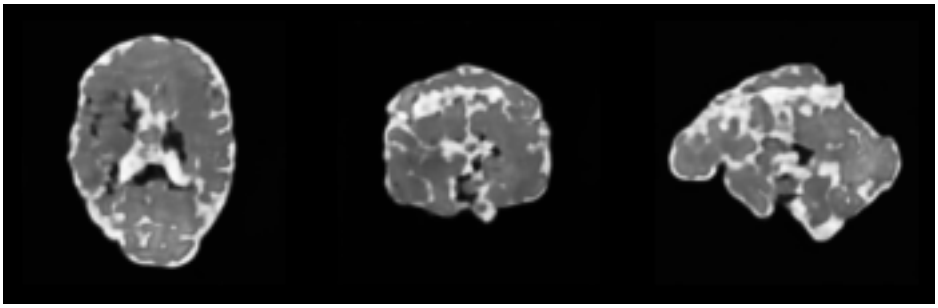
```
####################################
##################1#################
```



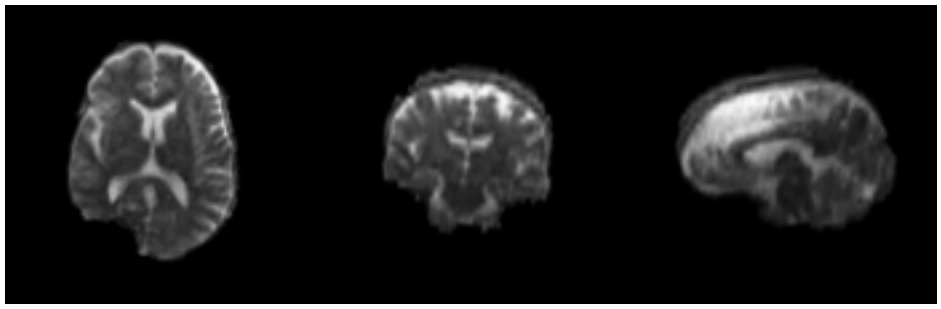


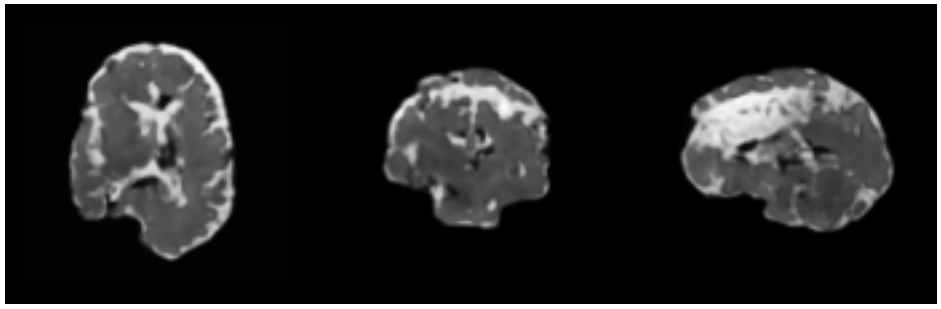
####################################
#################2#################



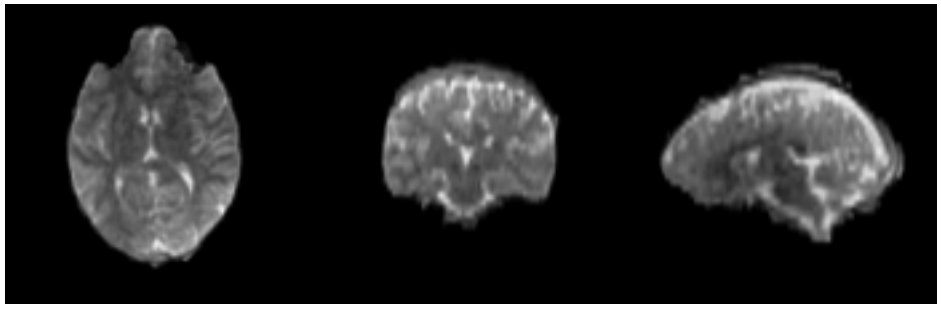


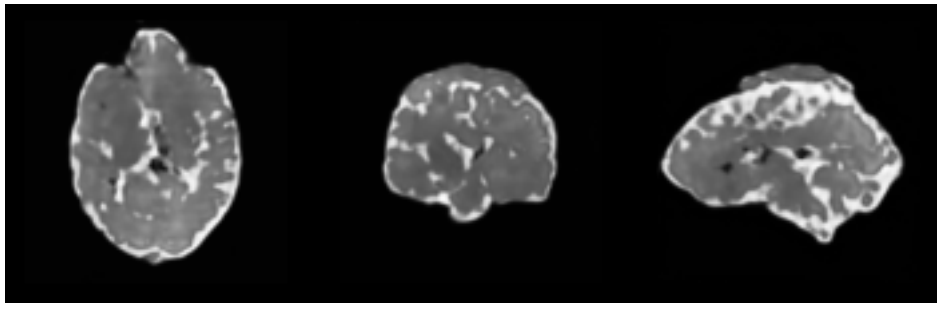
####################################
#################3#################



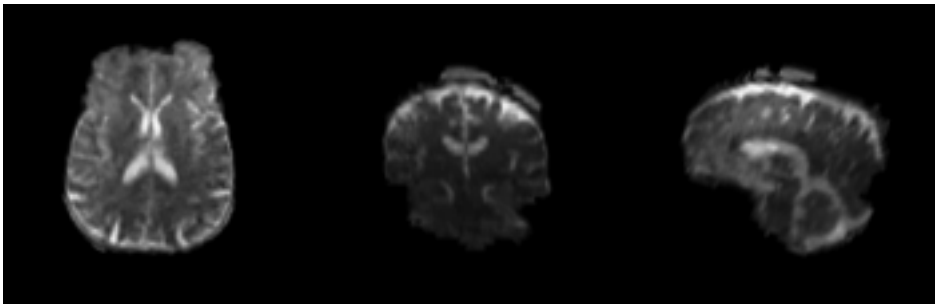


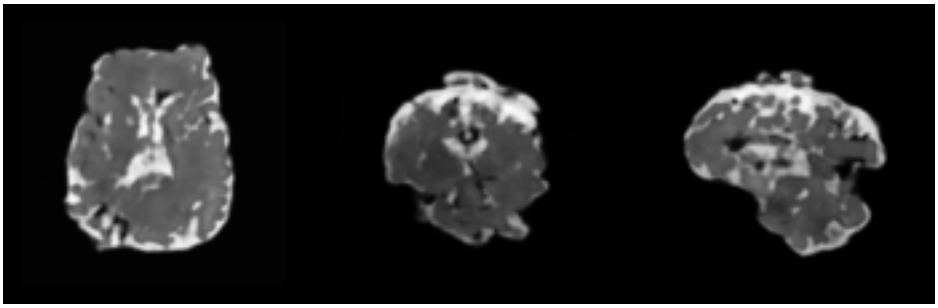
####################################
##################4#################



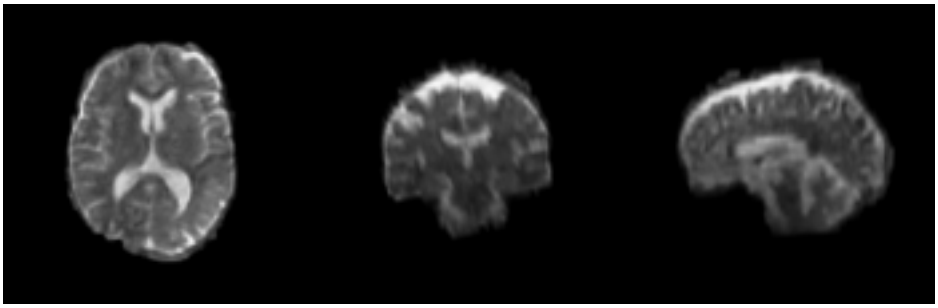


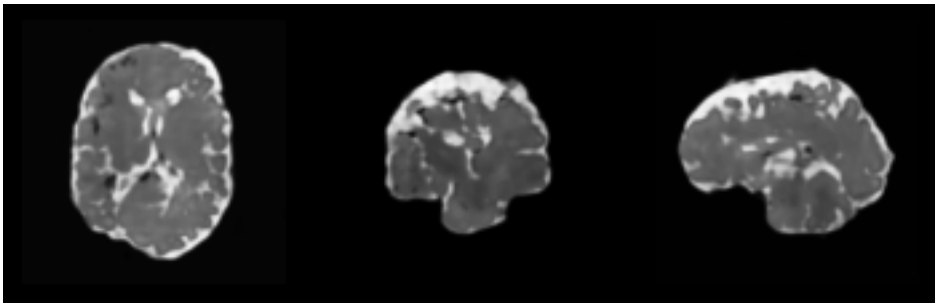
####################################
#################5#################



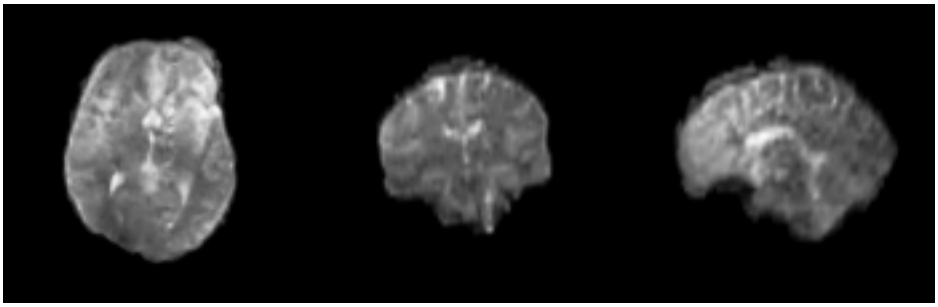


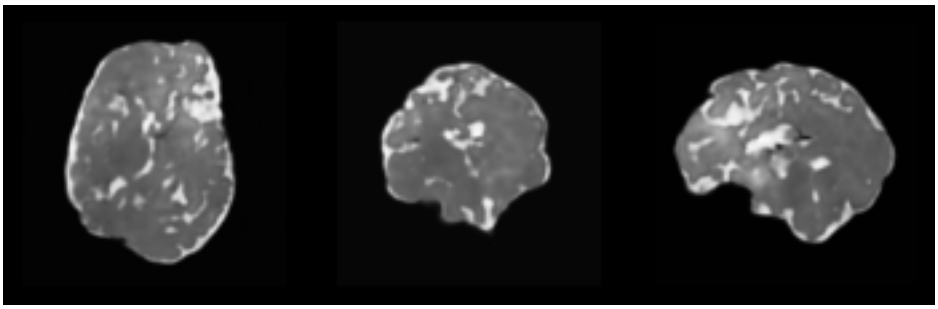
####################################
#################6#################



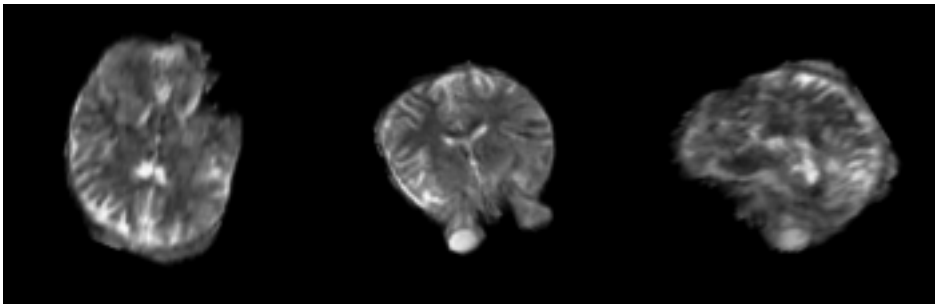


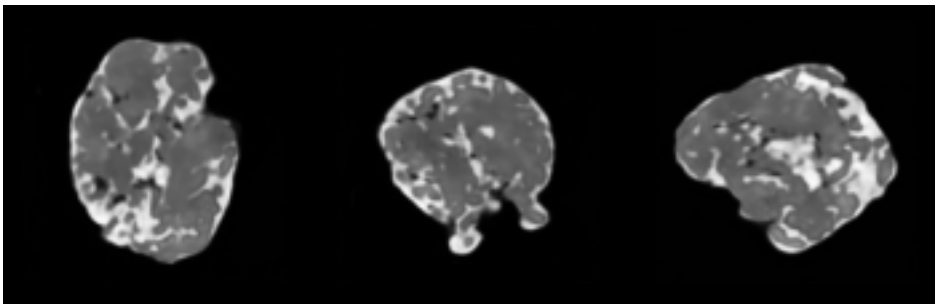
####################################
#################7#################



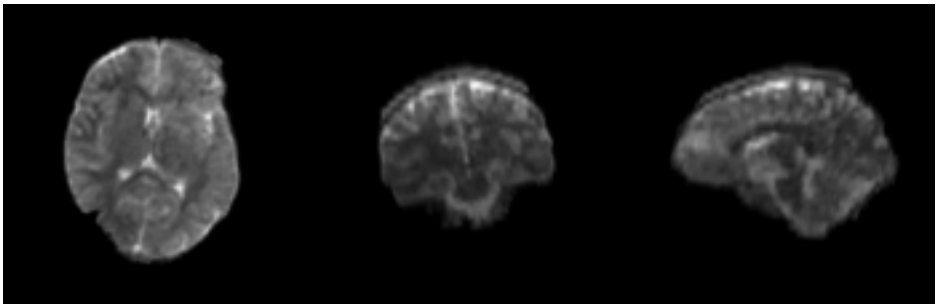


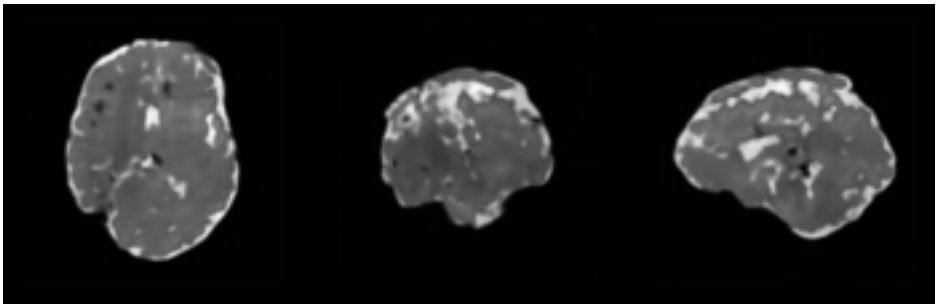
####################################
#################8#################



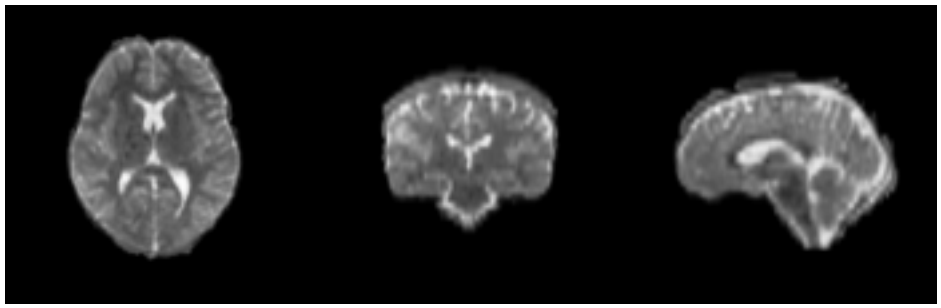


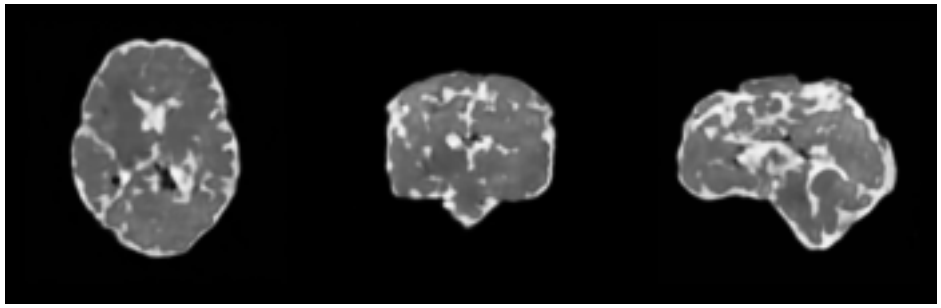
####################################
#################9##################



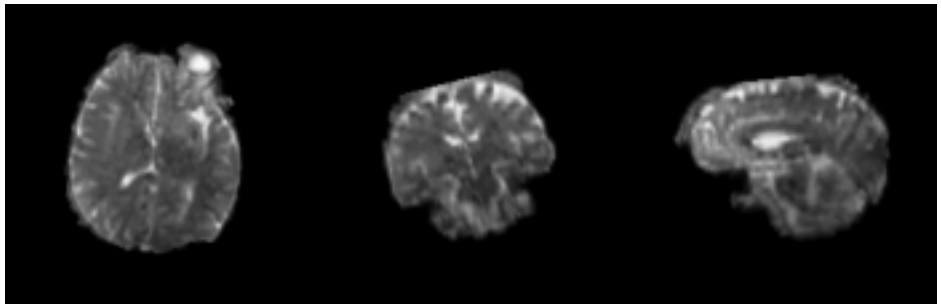


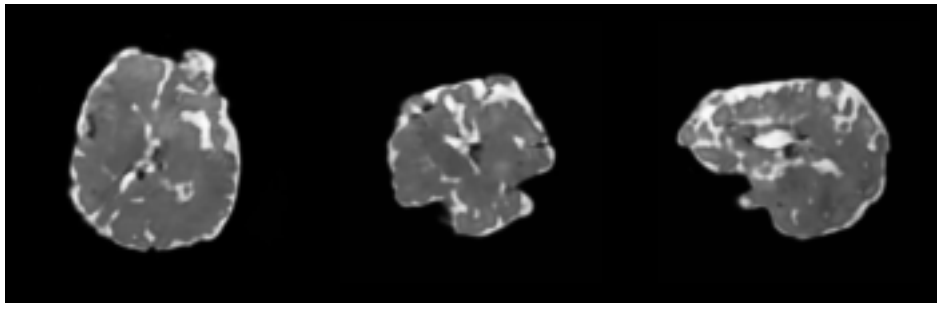
####################################
#################10#################



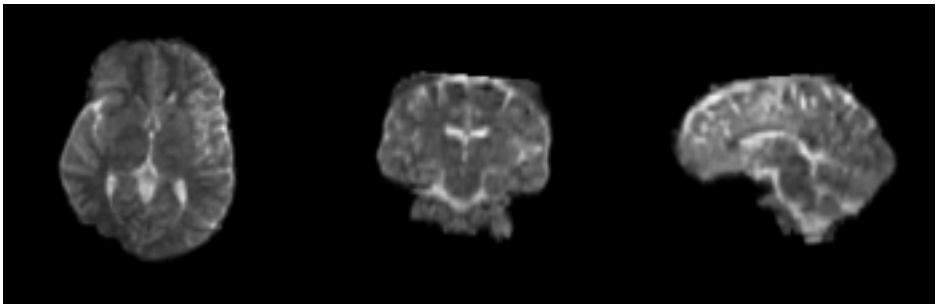


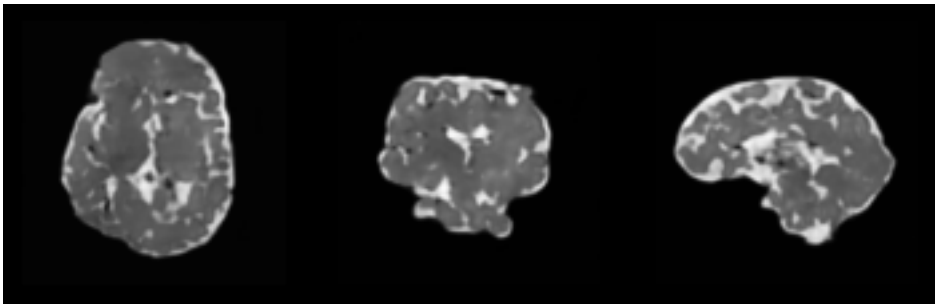
####################################
##################11#################



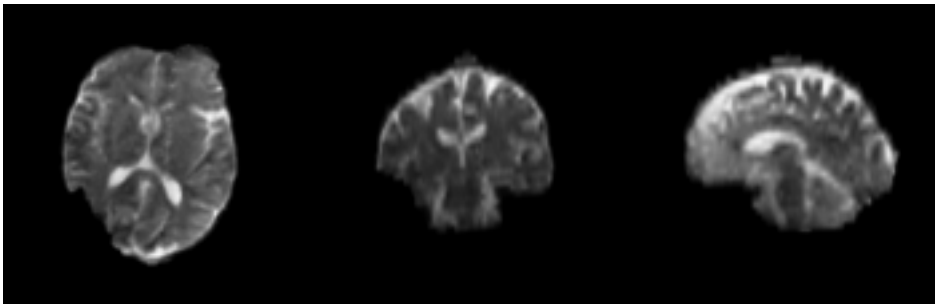


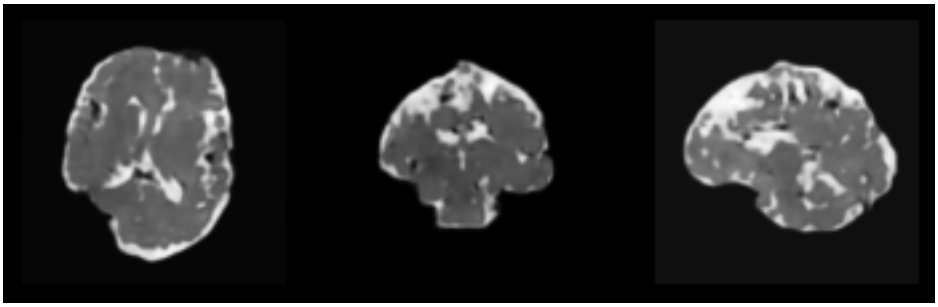
##################################
#################12###############



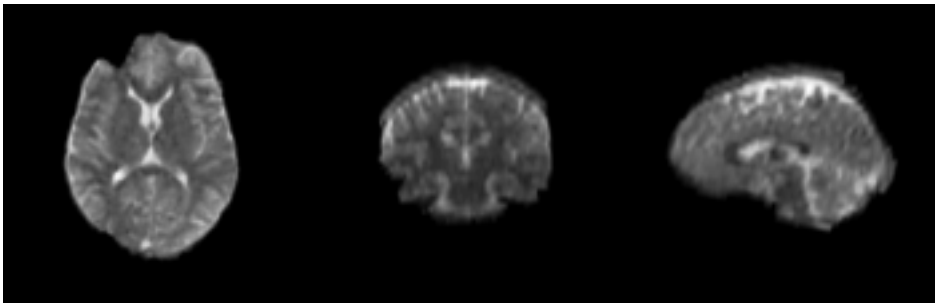


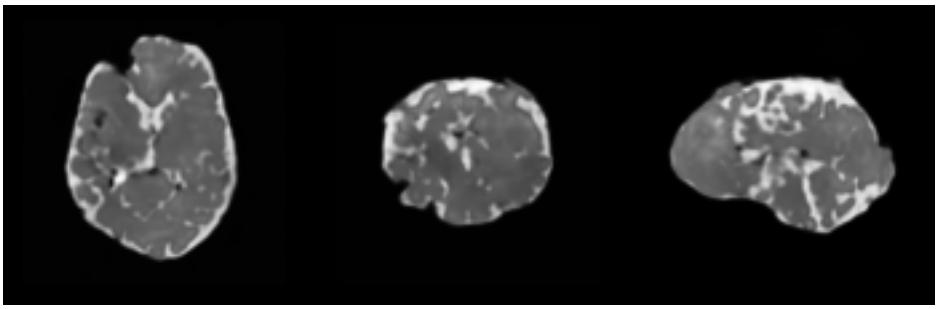
##################################
#################13###############



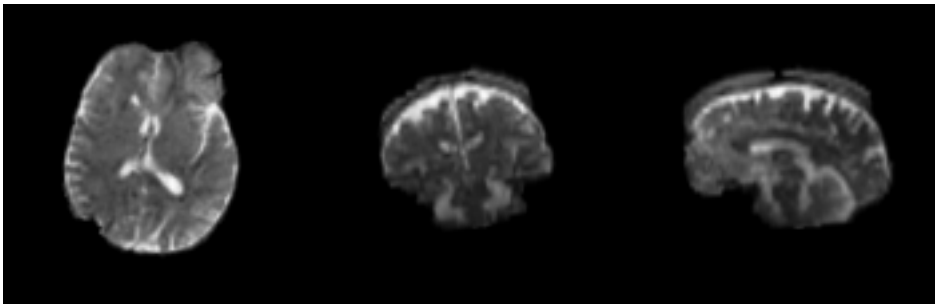


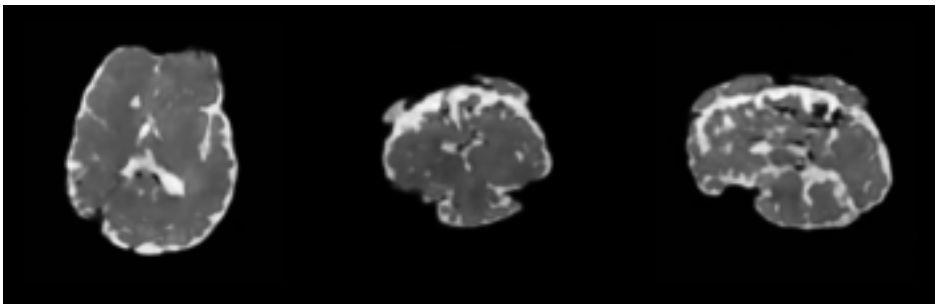
####################################
#################14#################



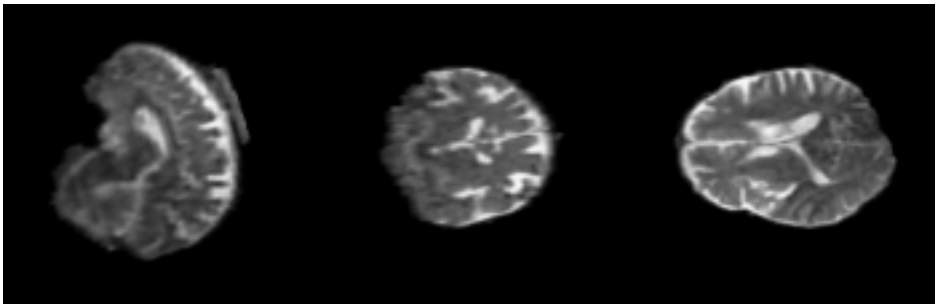


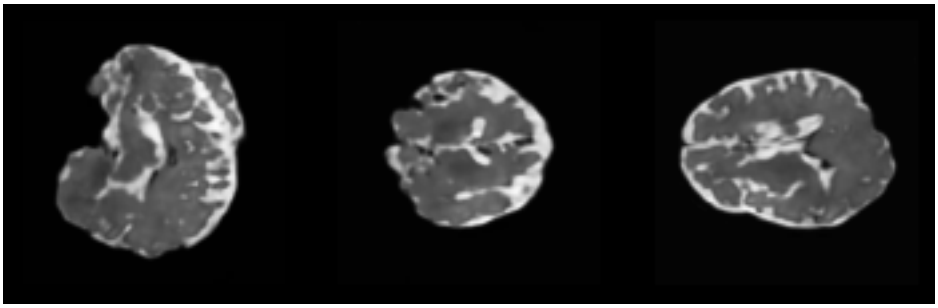
####################################
#################15#################



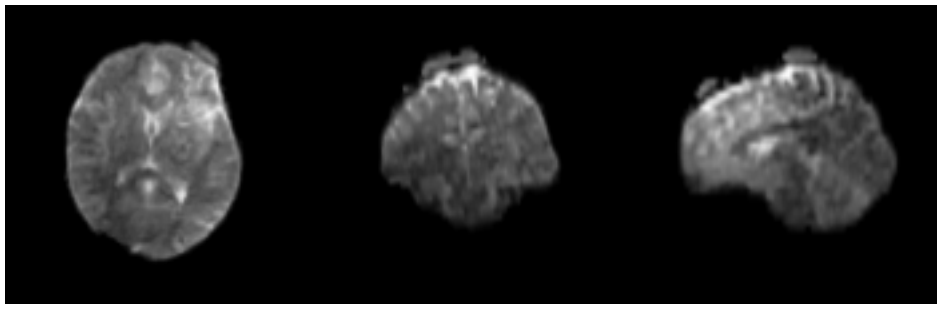


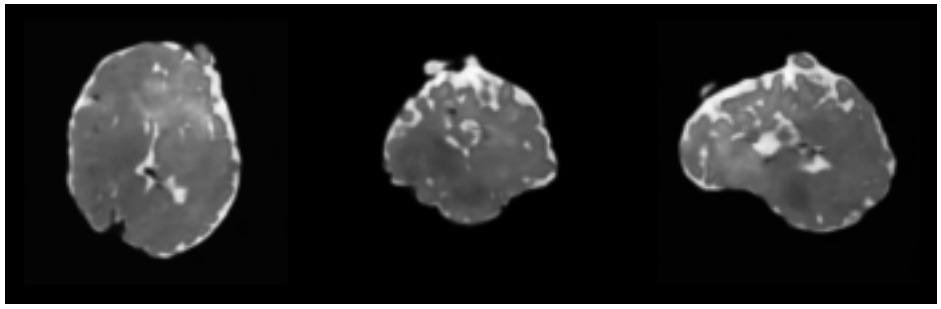
####################################
#################16#################



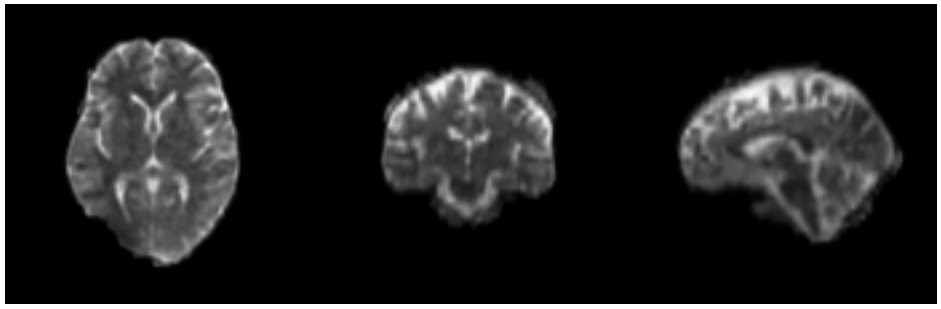


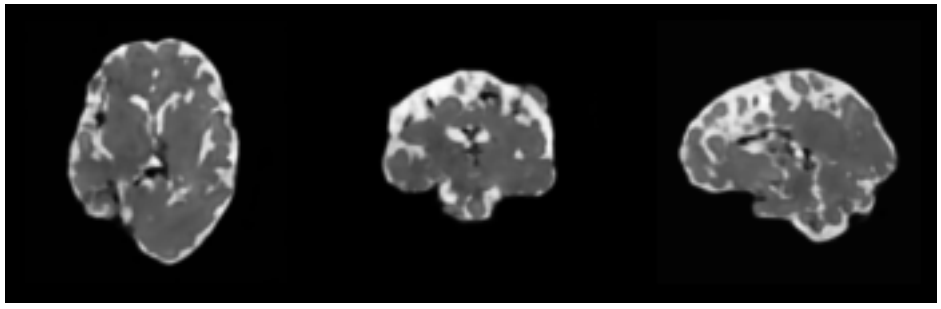
####################################
#################17#################



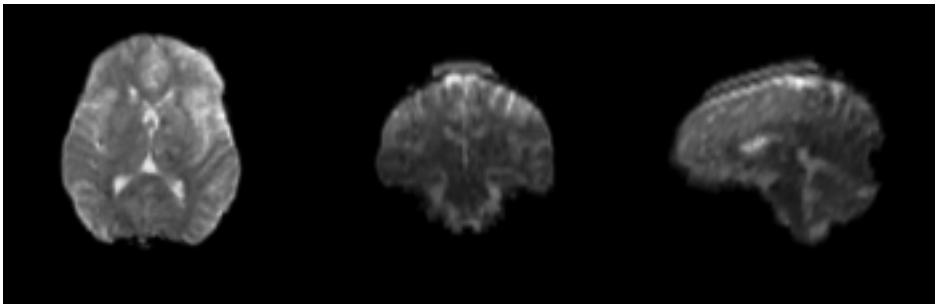


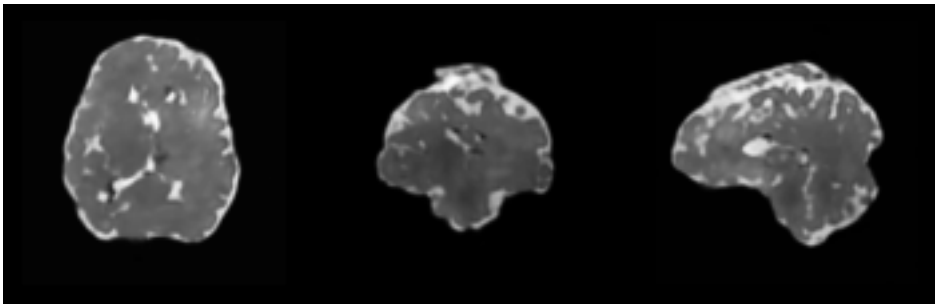
####################################
##################18#################



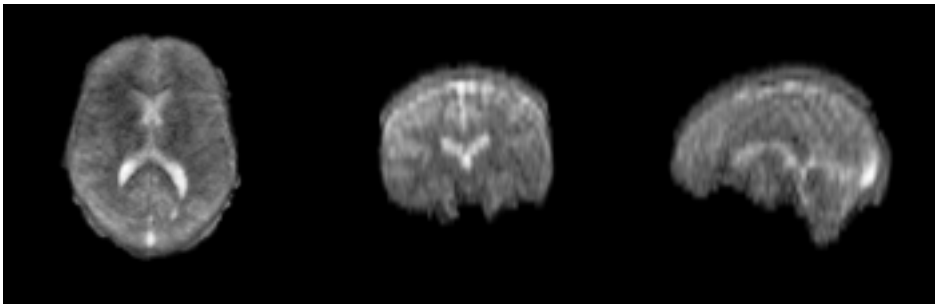


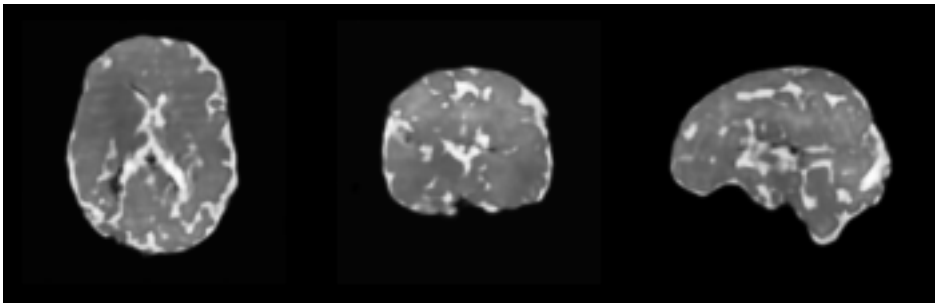
####################################
##################19#################



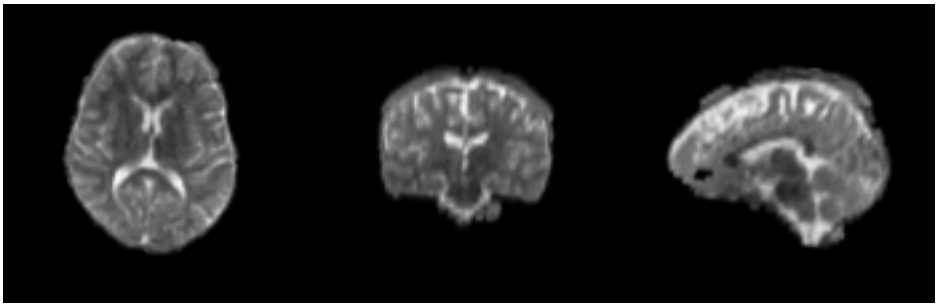


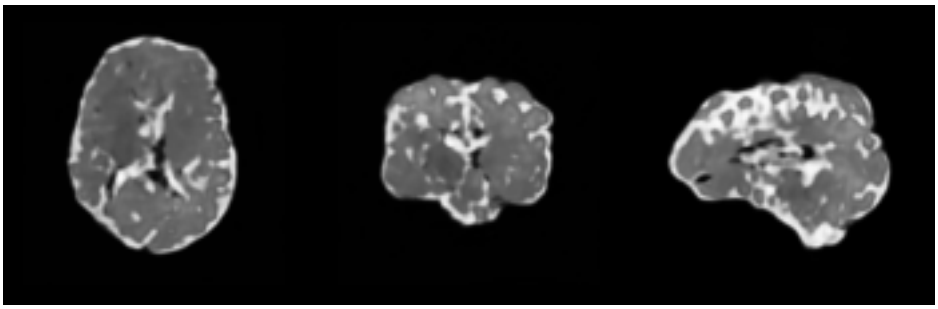
####################################
##################20#################



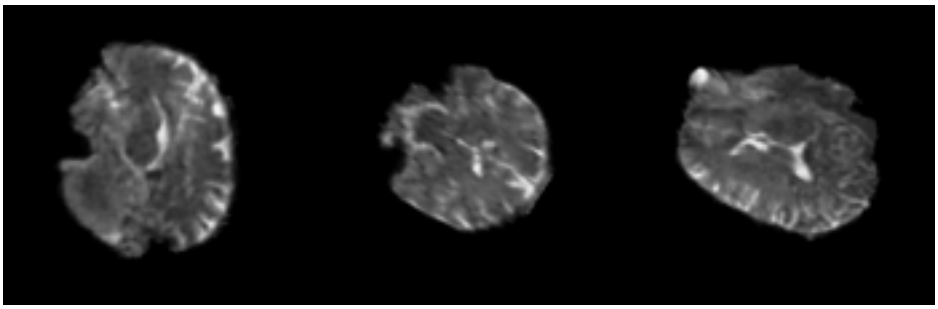


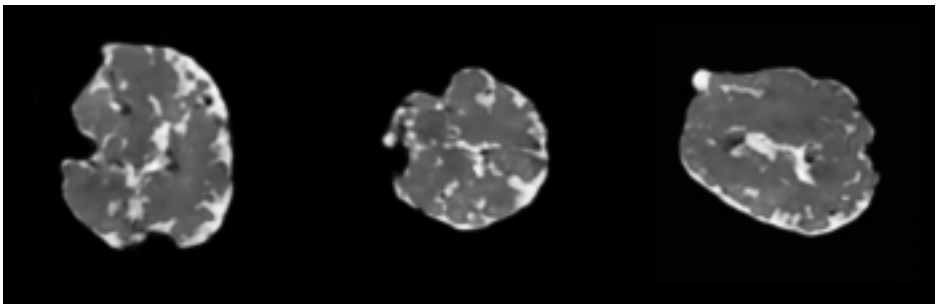
####################################
#################21#################



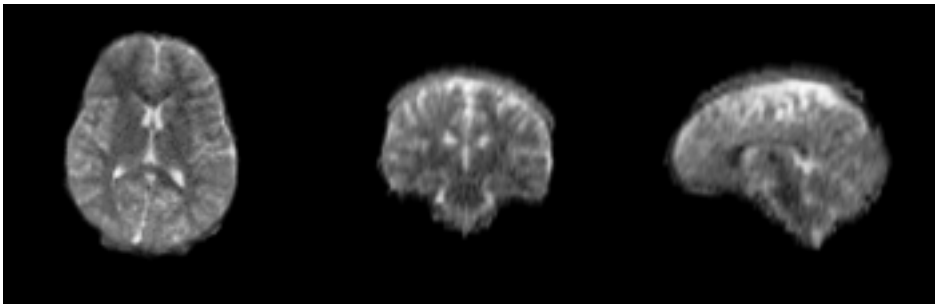


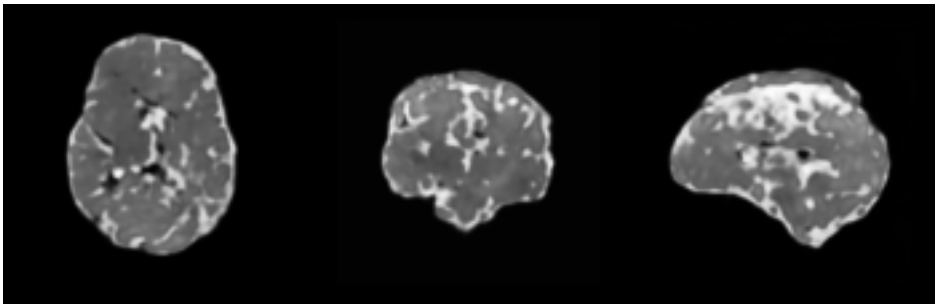
####################################
#################22#################



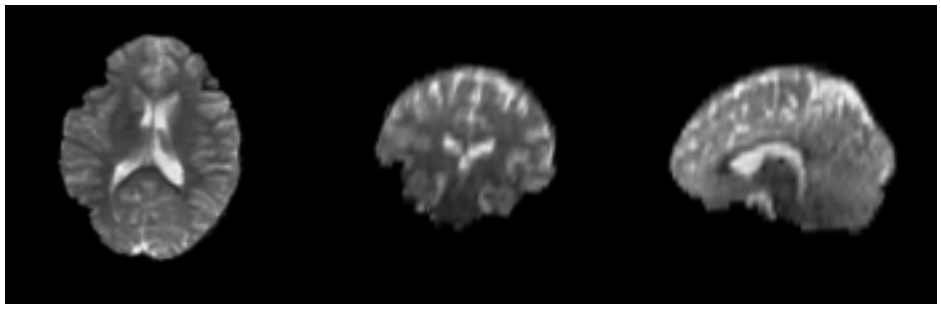


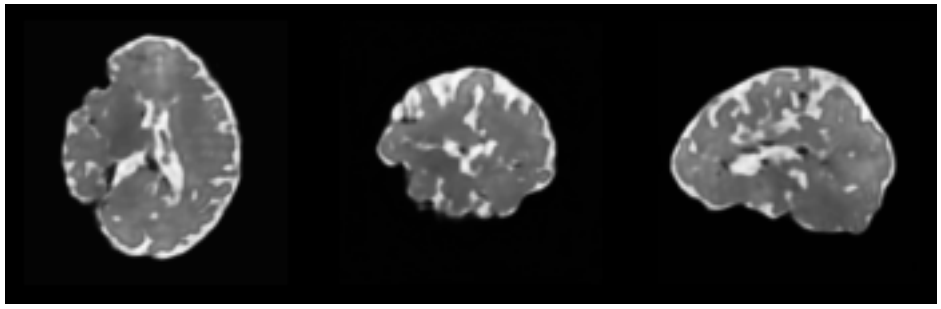
####################################
##################23#################



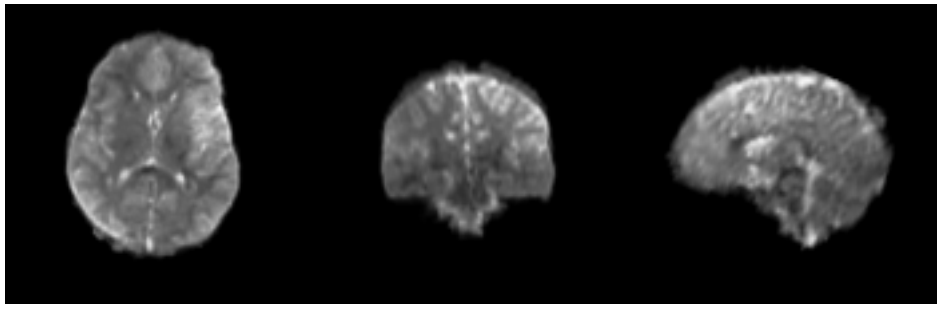


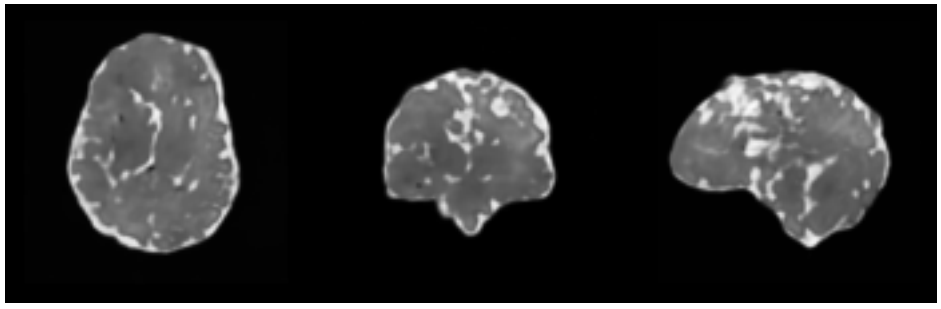
####################################
##################24#################



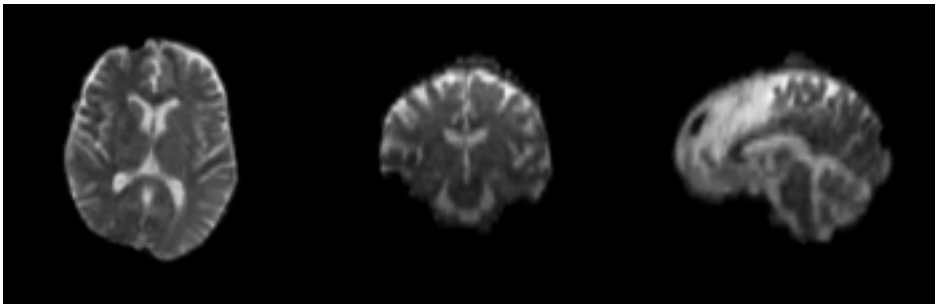


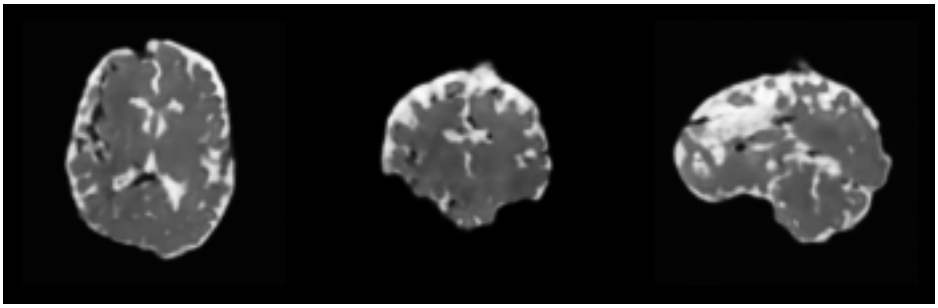
####################################
##################25#################



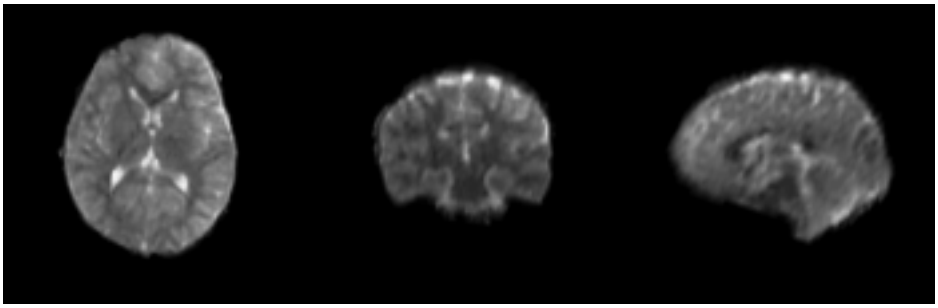


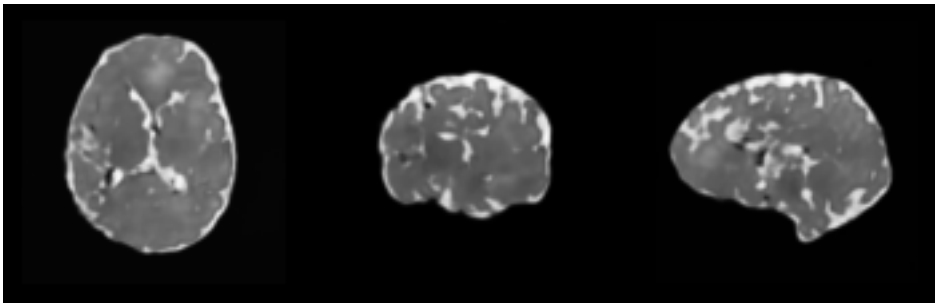
####################################
#################26#################



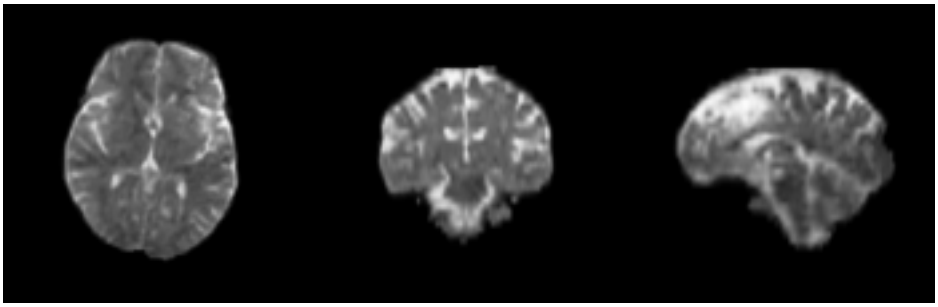


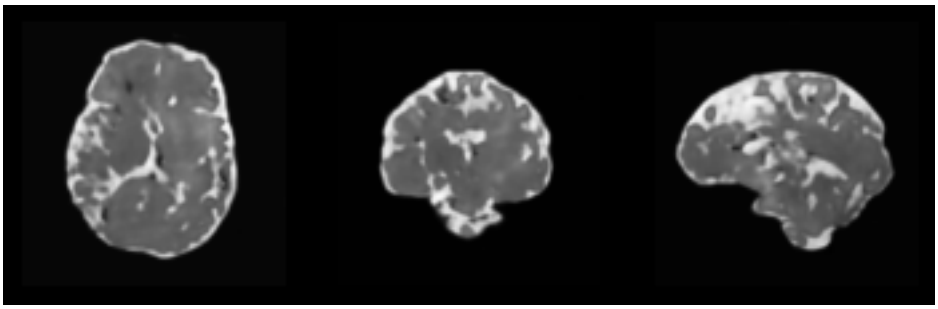
####################################
#################27#################



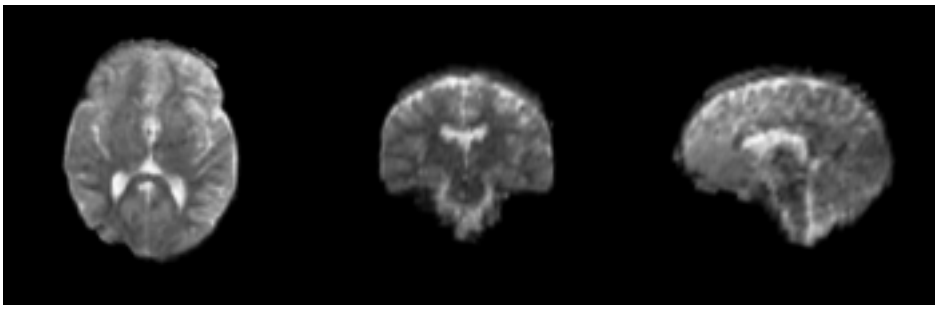


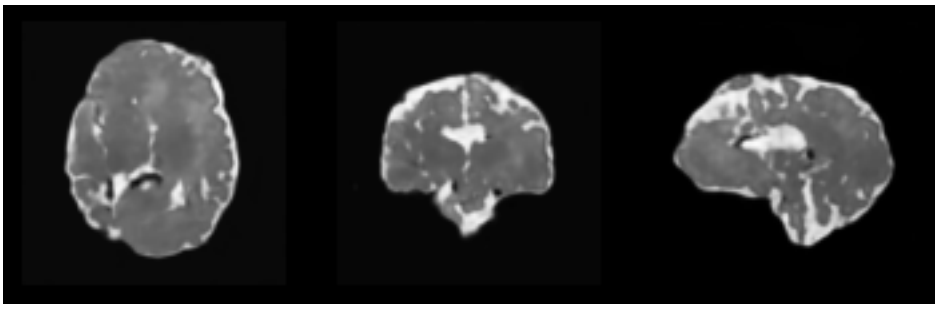
####################################
##################28#################



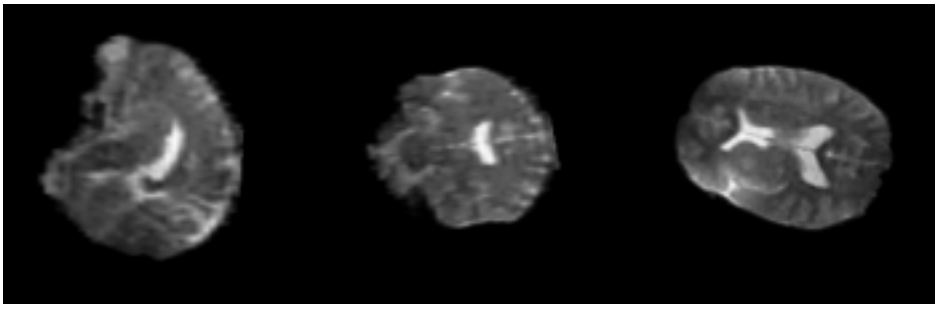


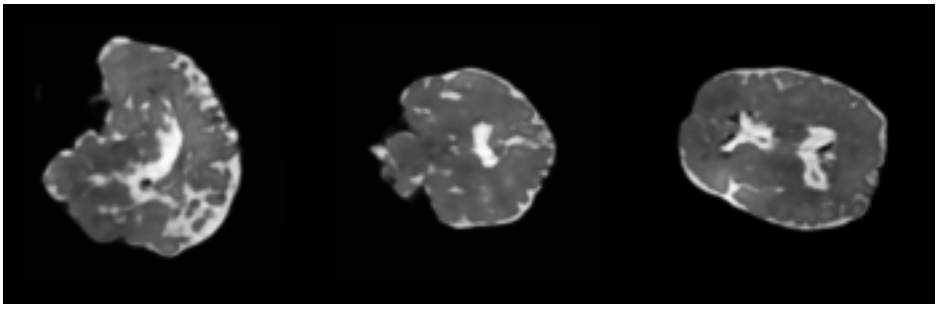
####################################
##################29#################



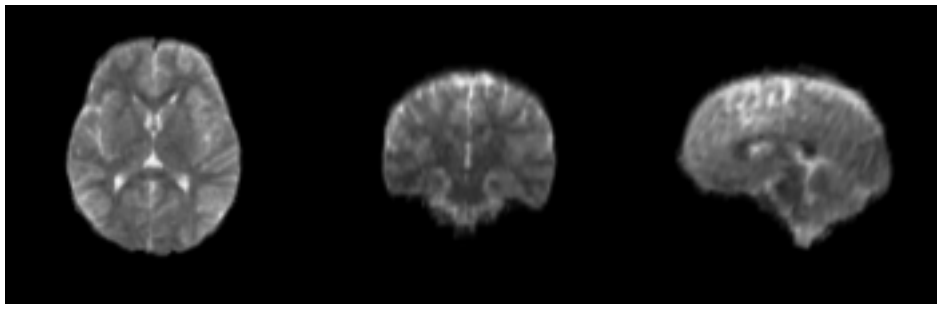


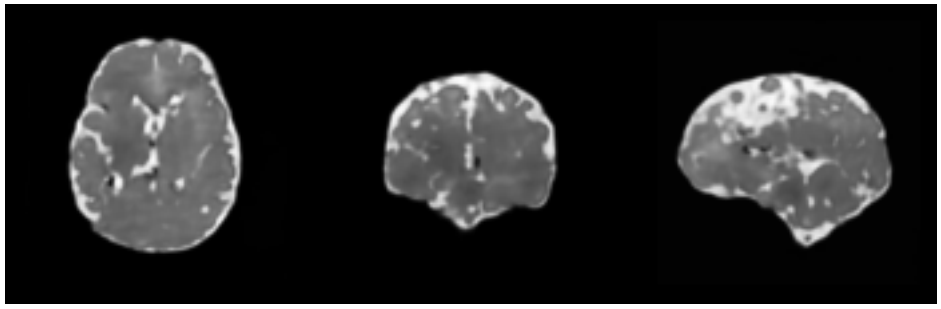
####################################
#################30#################



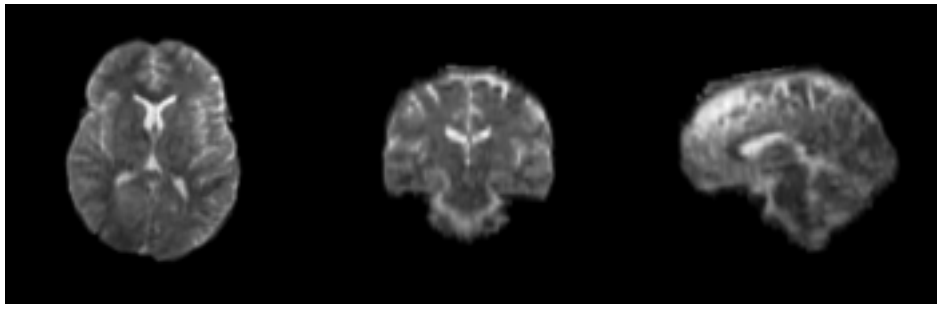


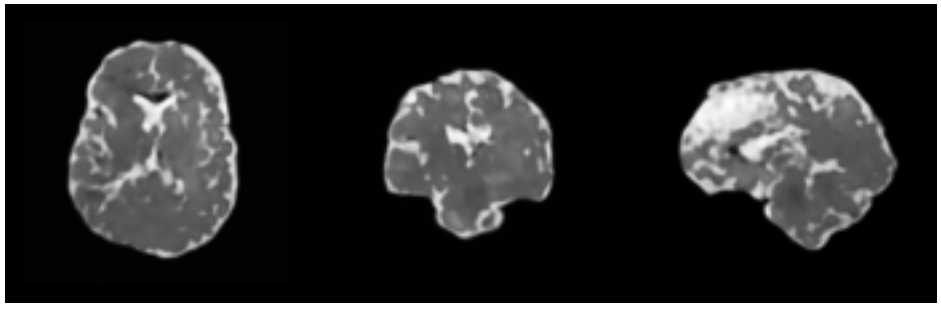
####################################
#################31#################



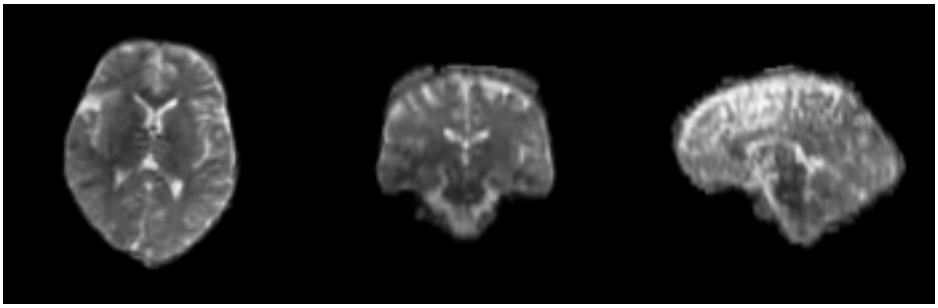


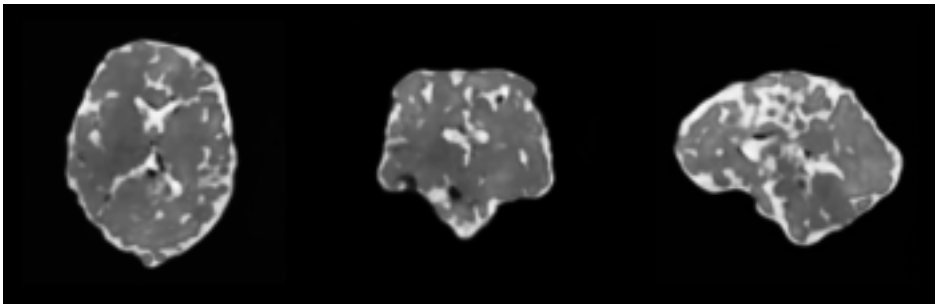
####################################
#################32#################



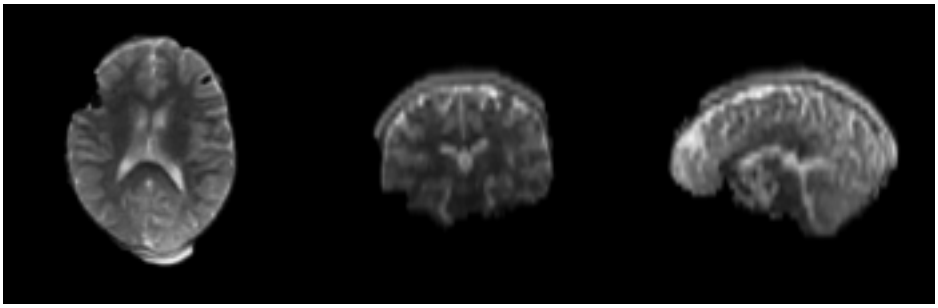


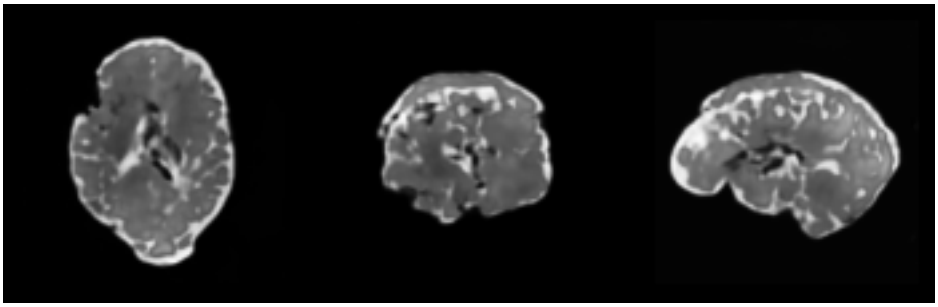
####################################
#################33#################



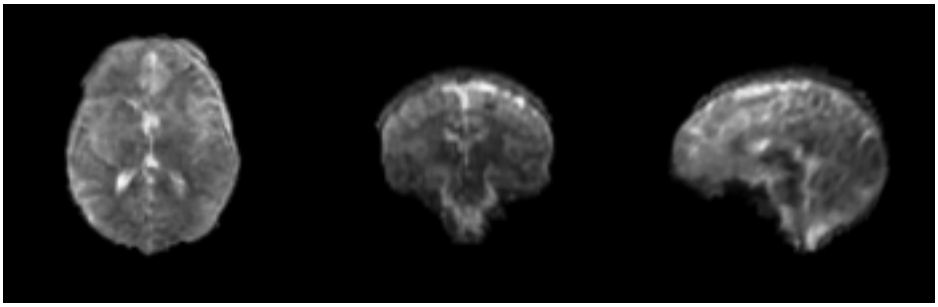


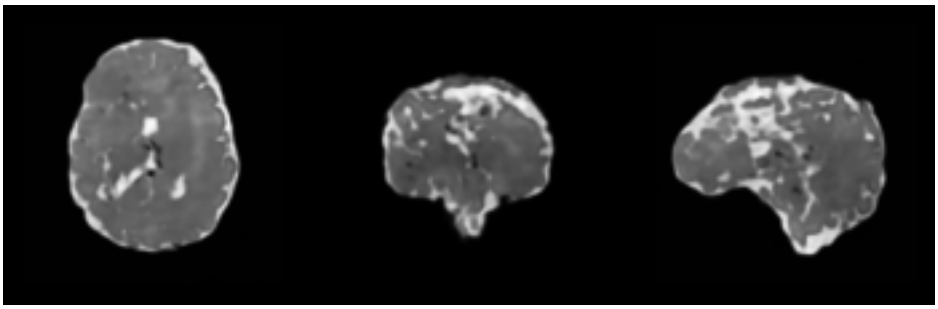
####################################
#################34#################



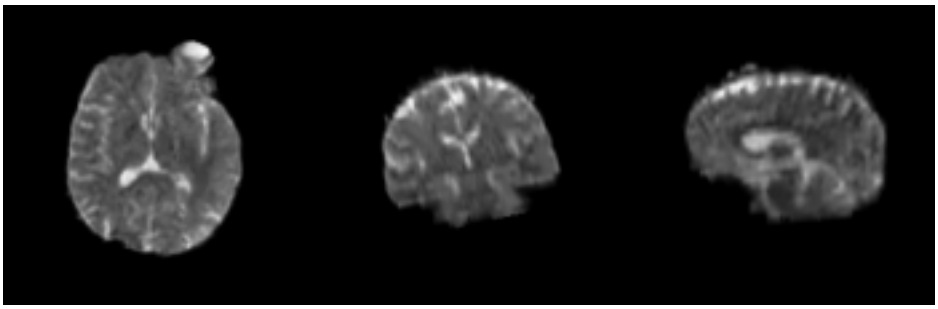


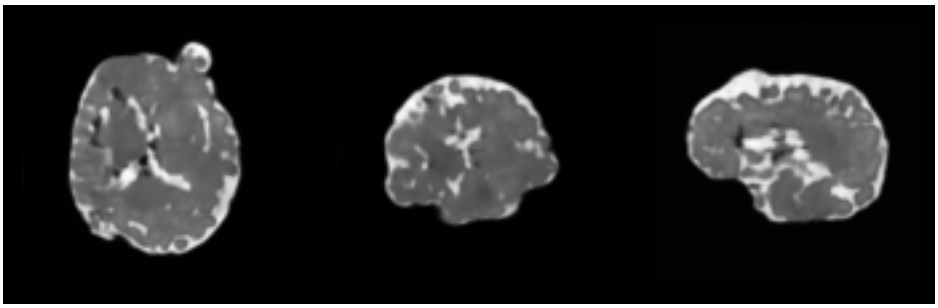
####################################
#################35#################



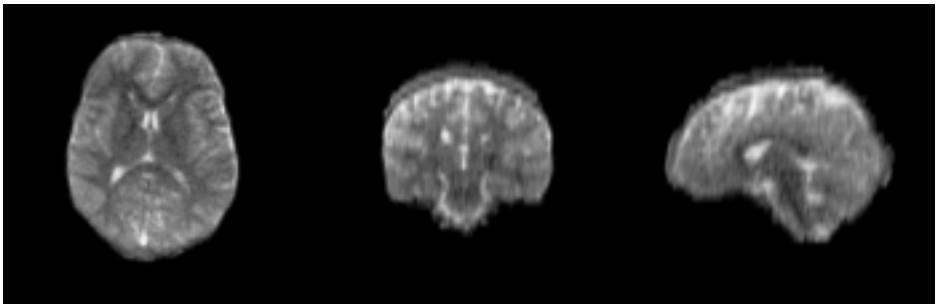


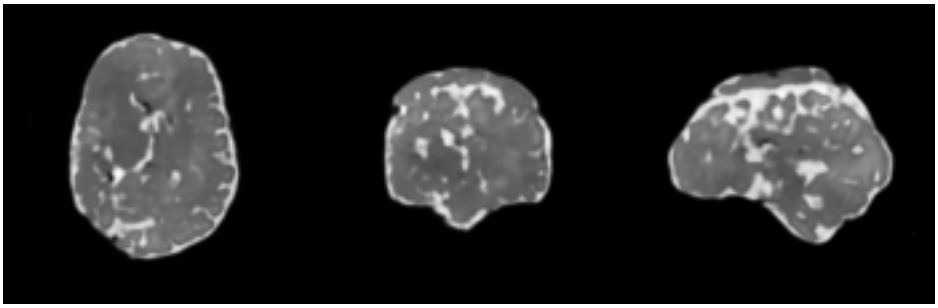
####################################
#################36#################



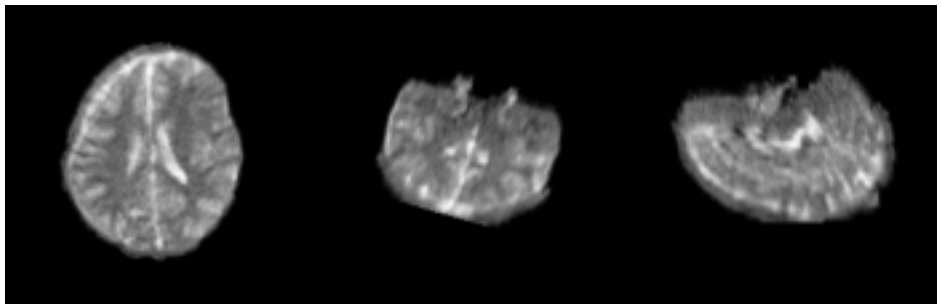


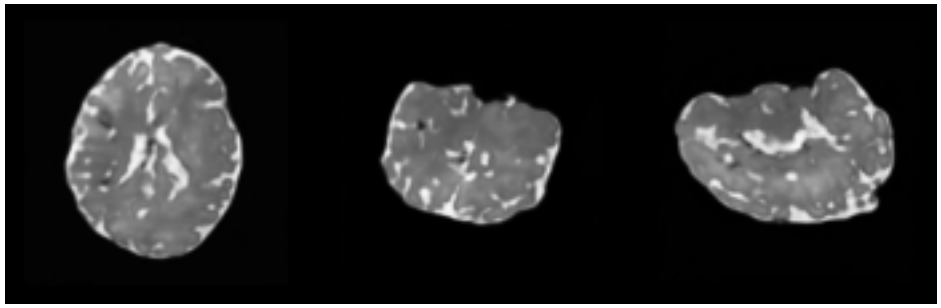
####################################
#################37#################



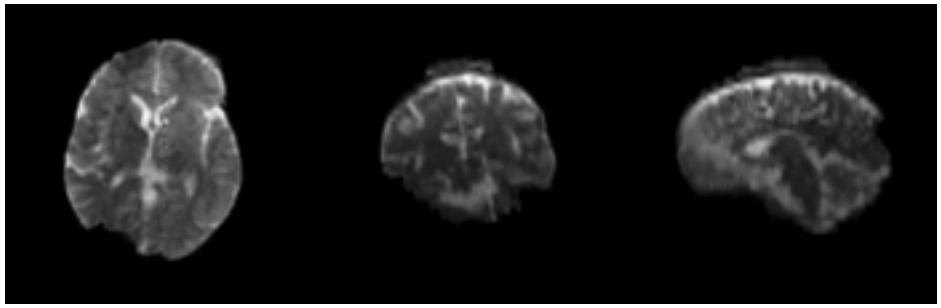


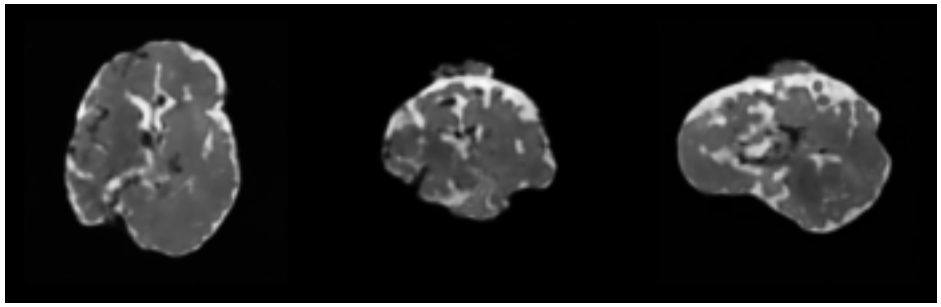
####################################
#################38#################



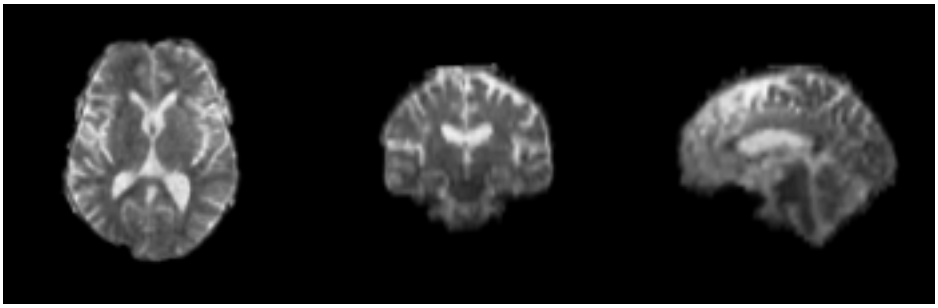


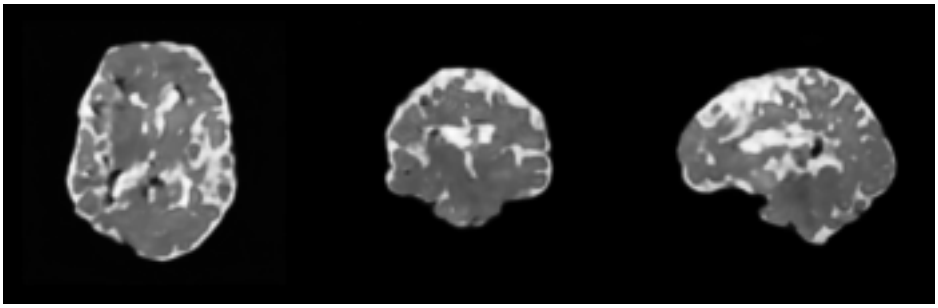
####################################
##################39#################



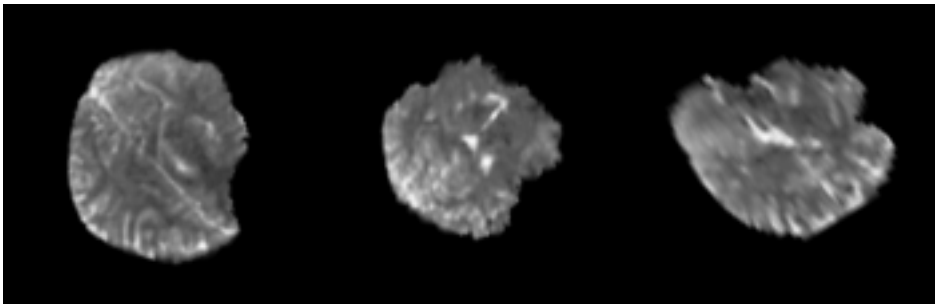


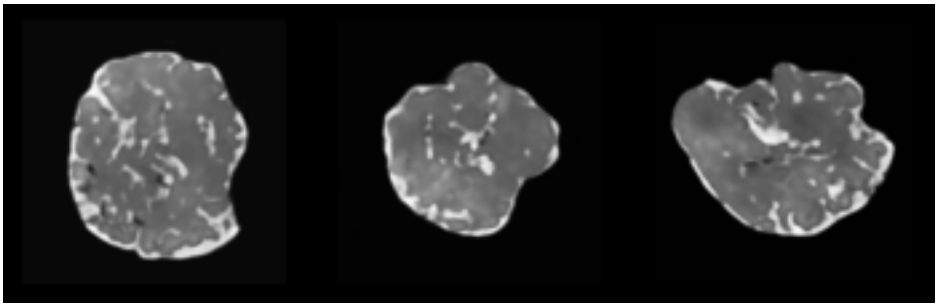
####################################
##################40################



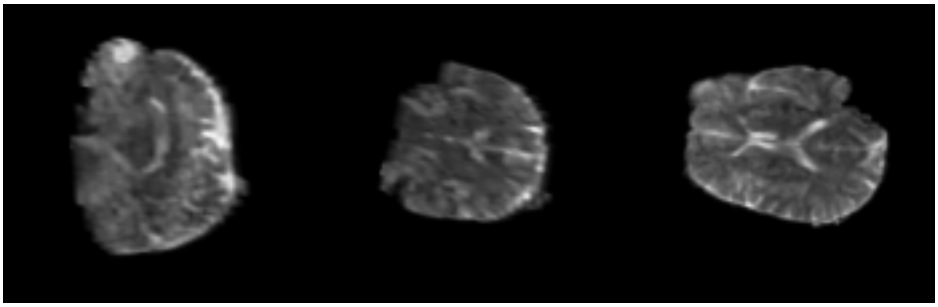


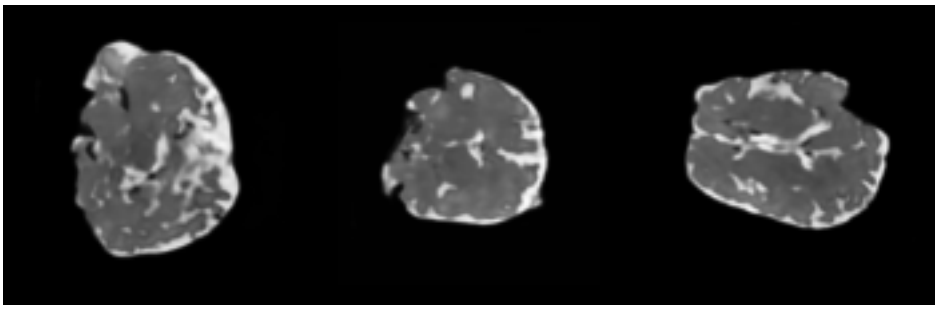
####################################
##################41################



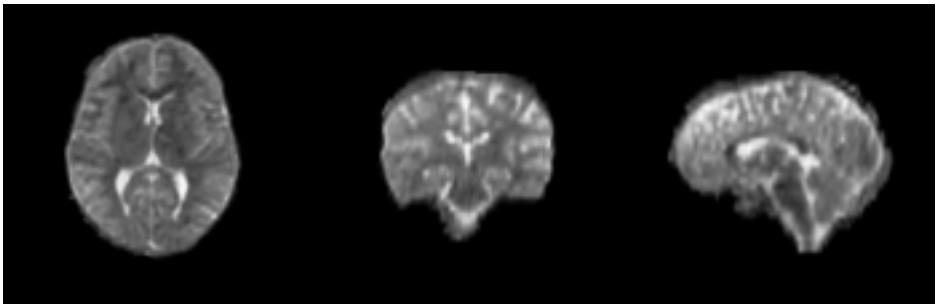


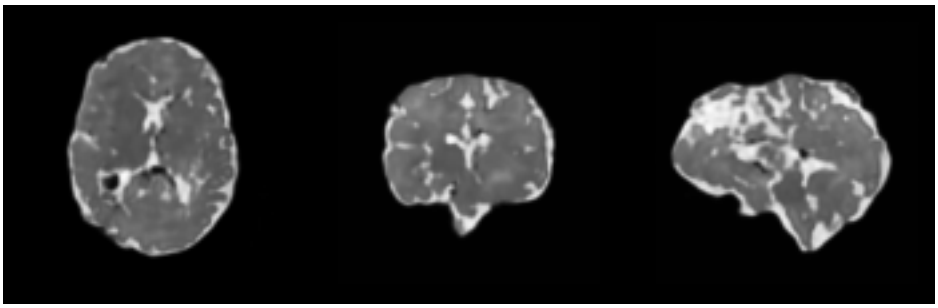
####################################
##################42#################



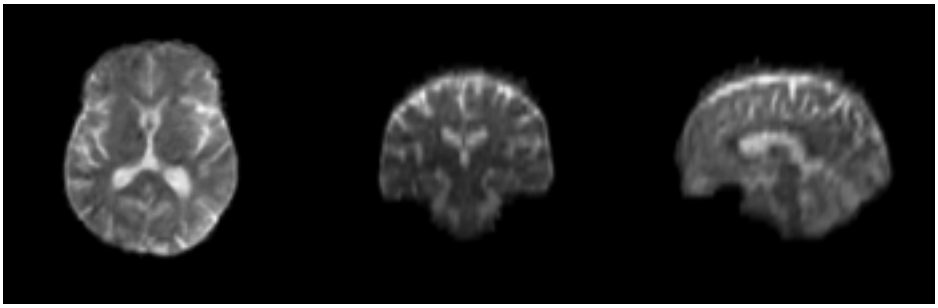


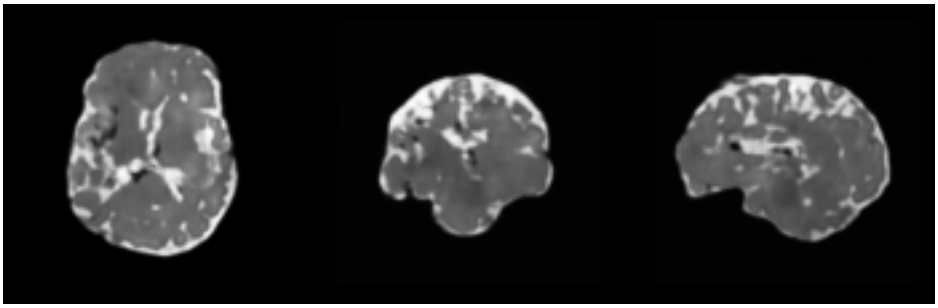
####################################
##################43#################



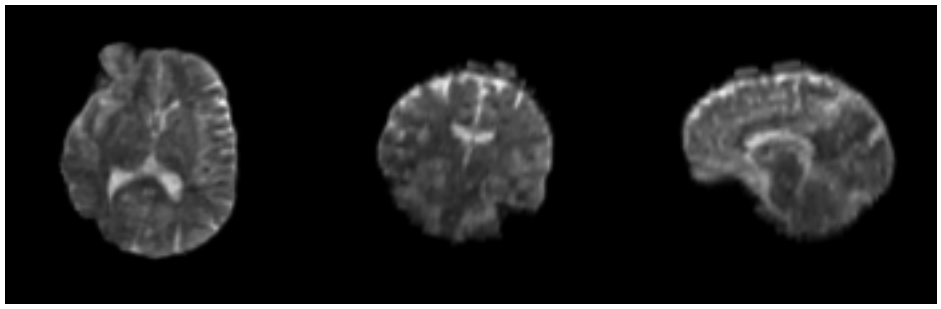


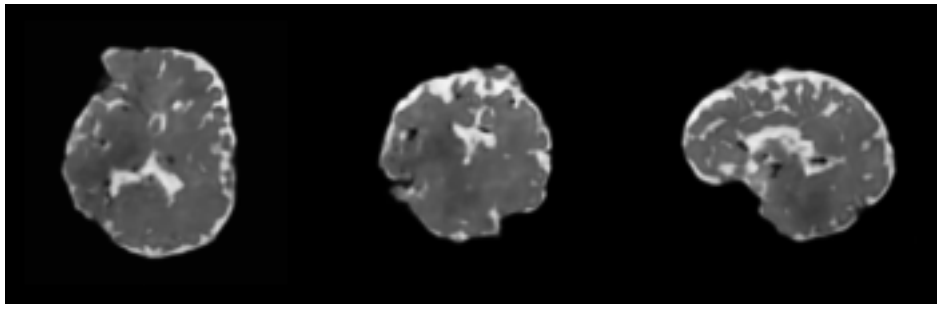
####################################
##################44################



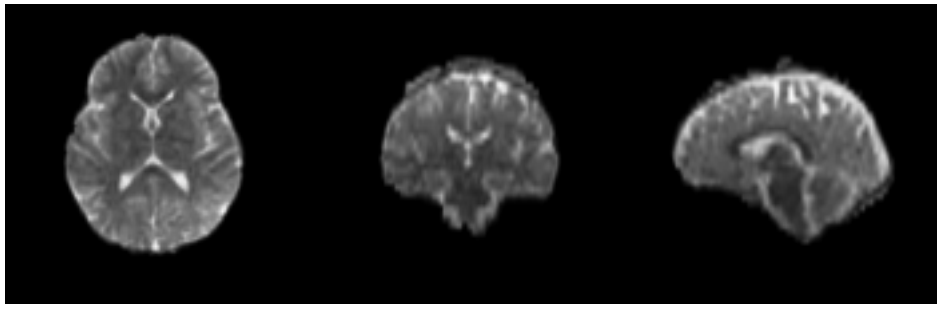


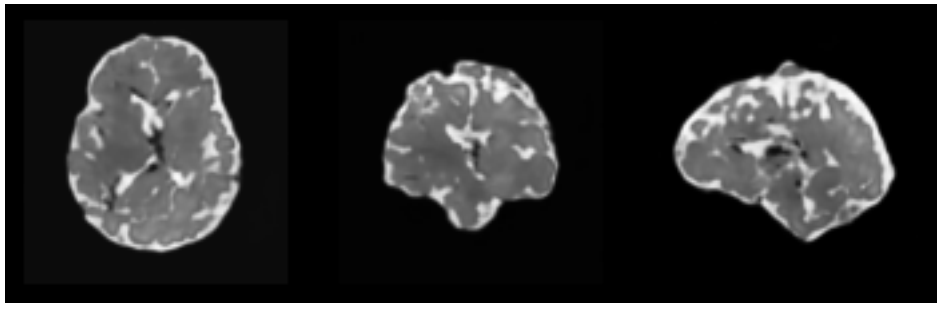
####################################
##################45################



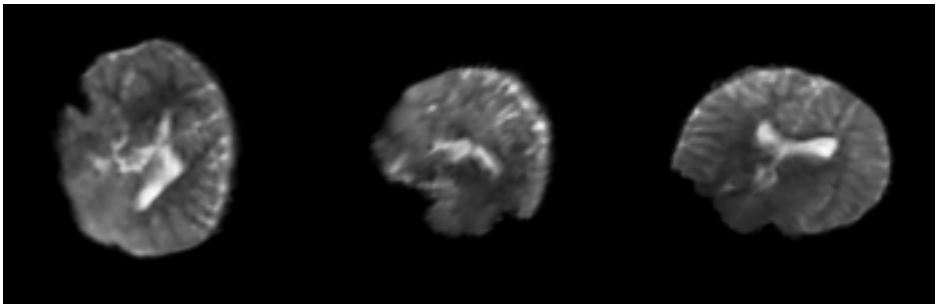


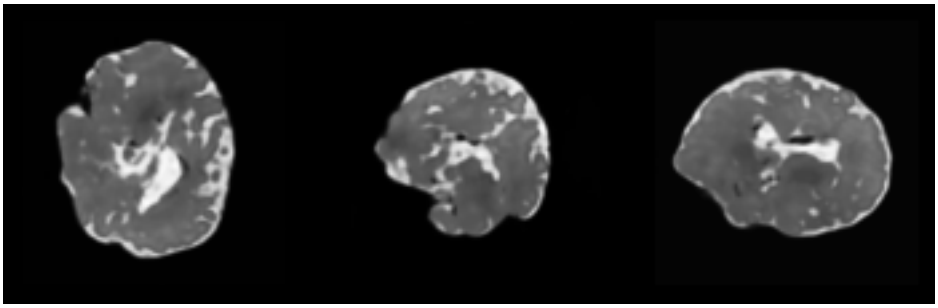
####################################
##################46#################



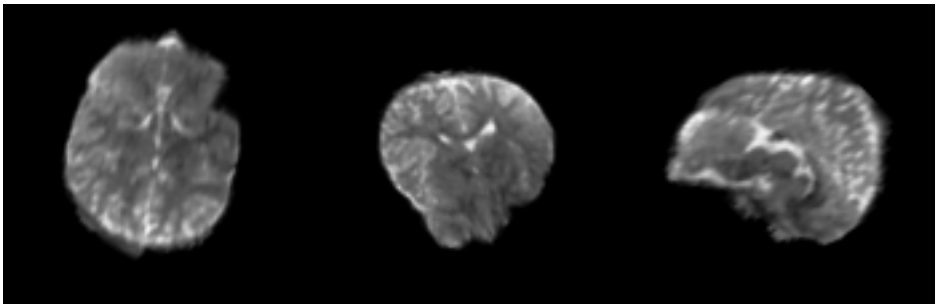


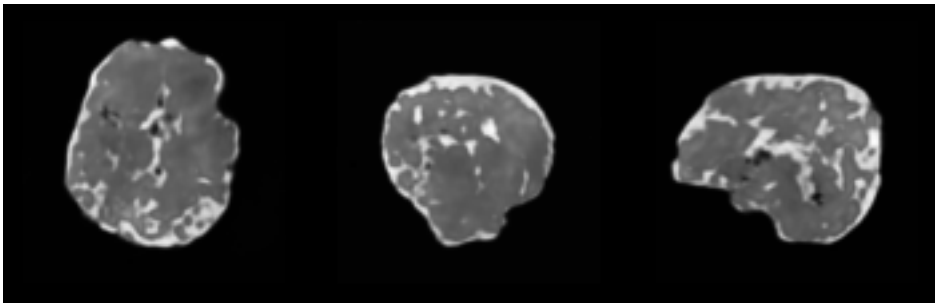
####################################
##################47#################



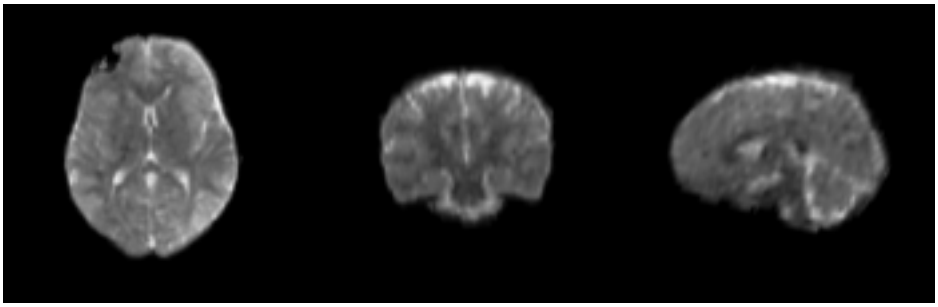


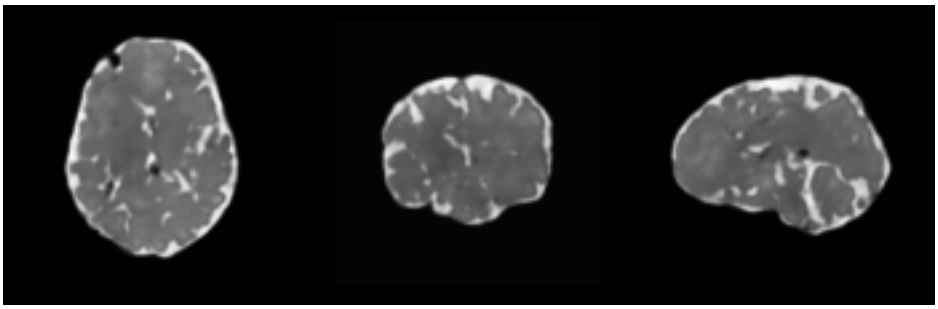
####################################
##################48#################



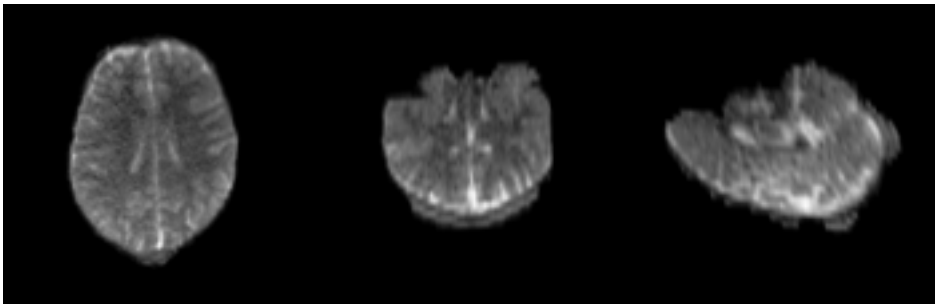


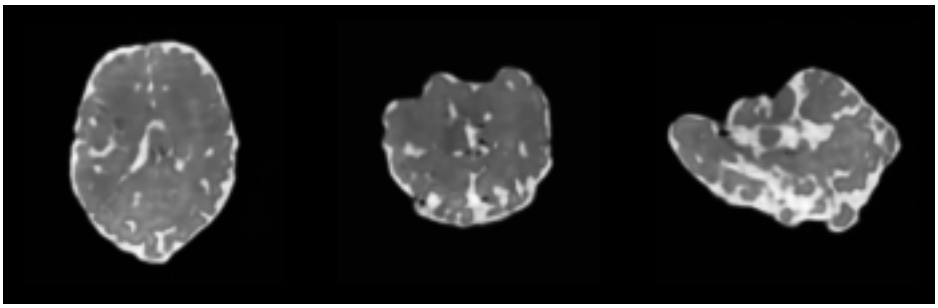
####################################
##################49################



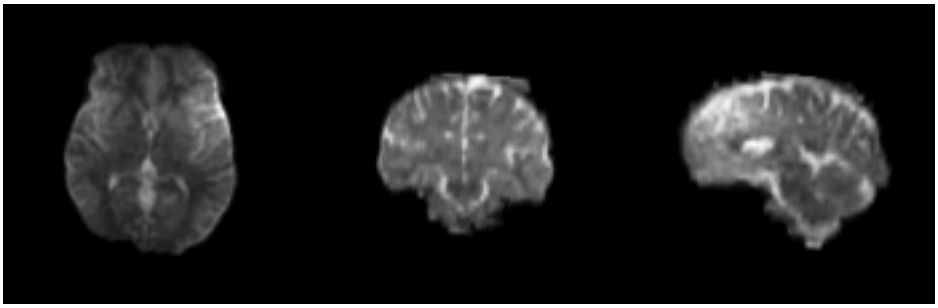


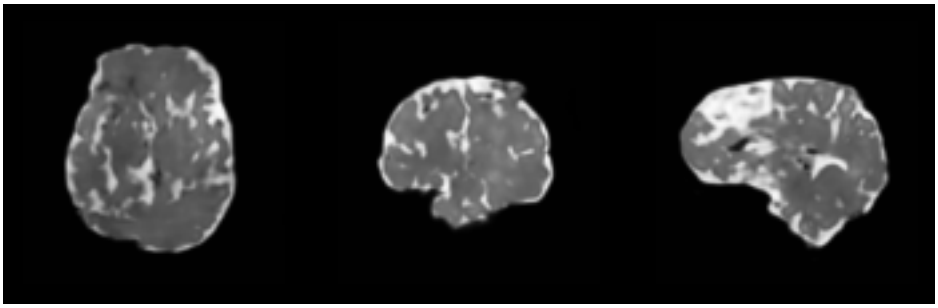
####################################
##################50################



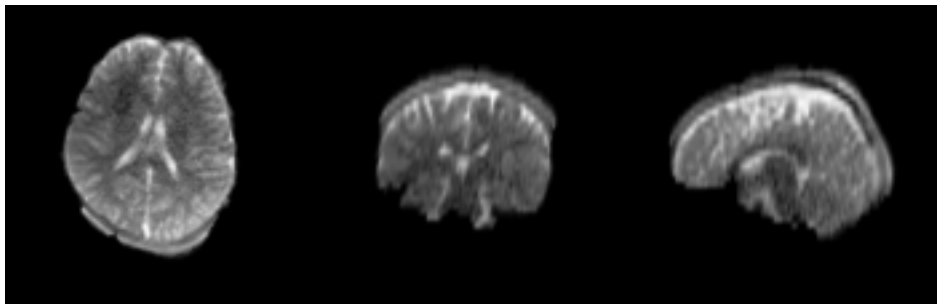


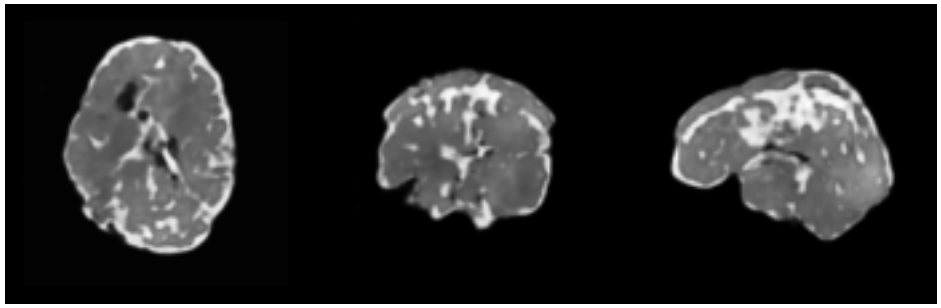
####################################
#################51#################



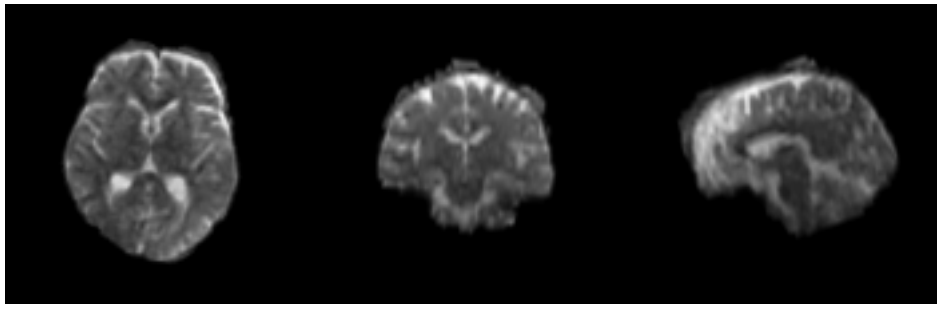


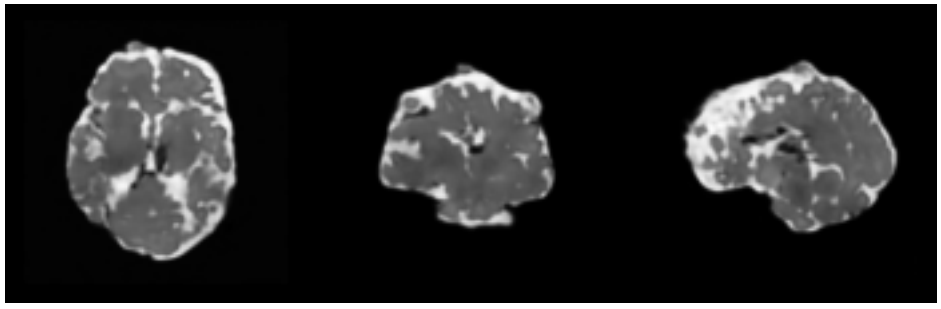
####################################
#################52#################



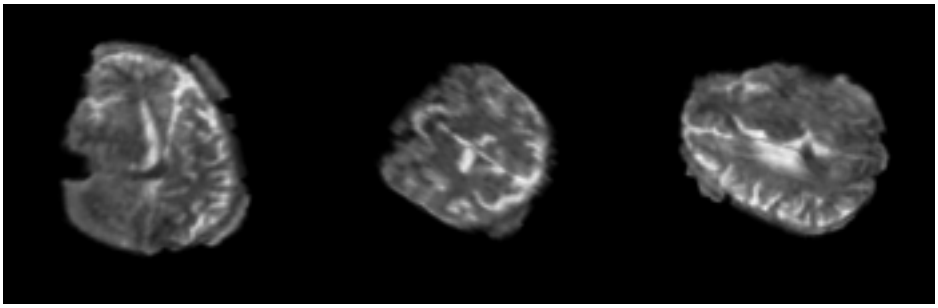


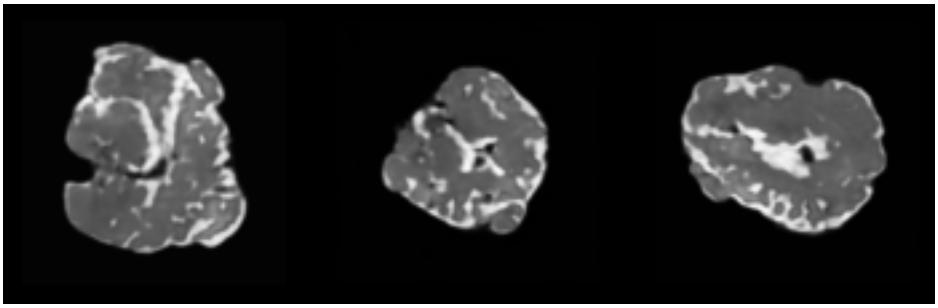
####################################
#################53#################



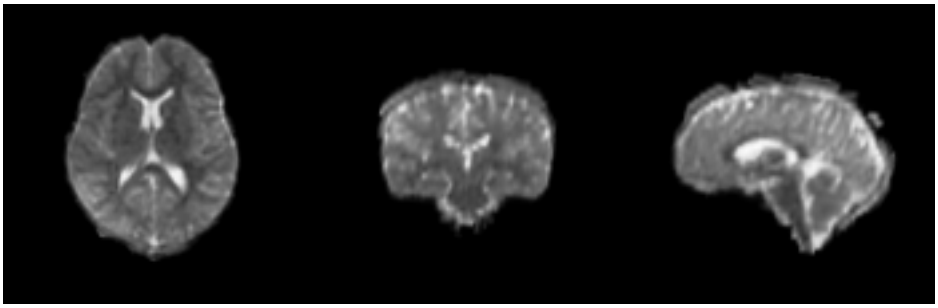


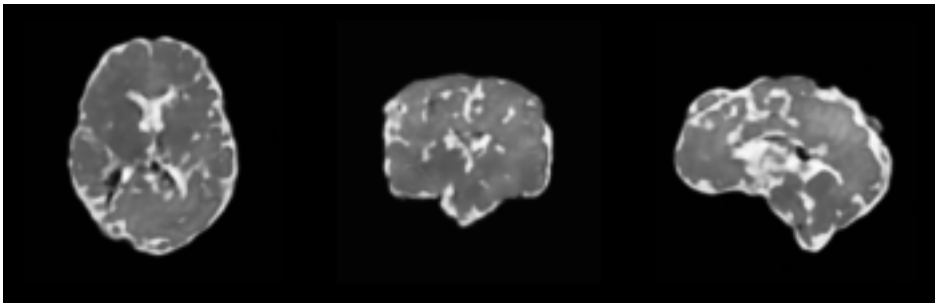
####################################
#################54#################



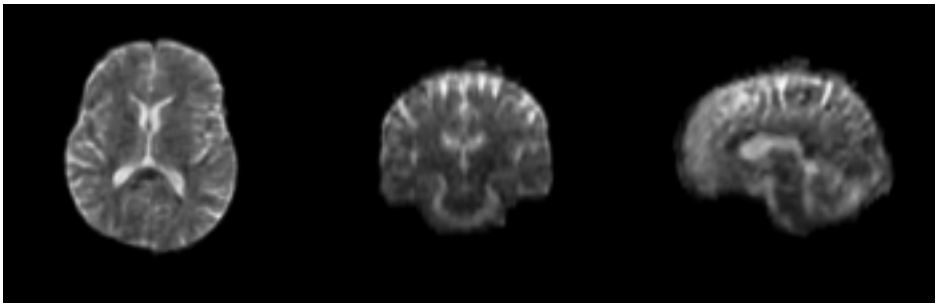


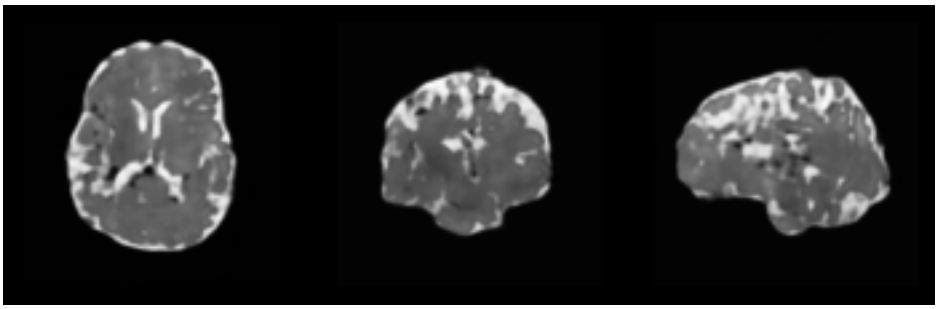
####################################
#################55#################



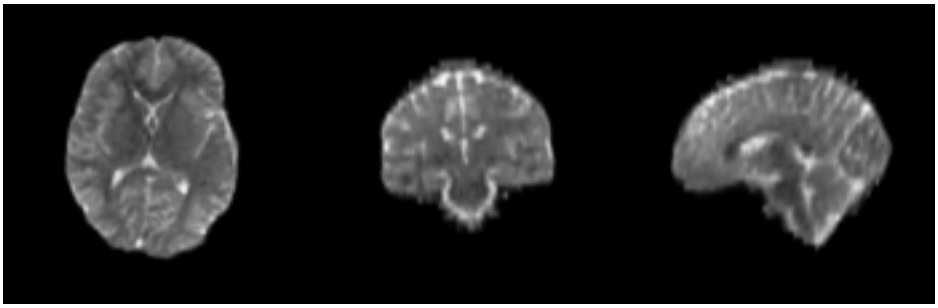


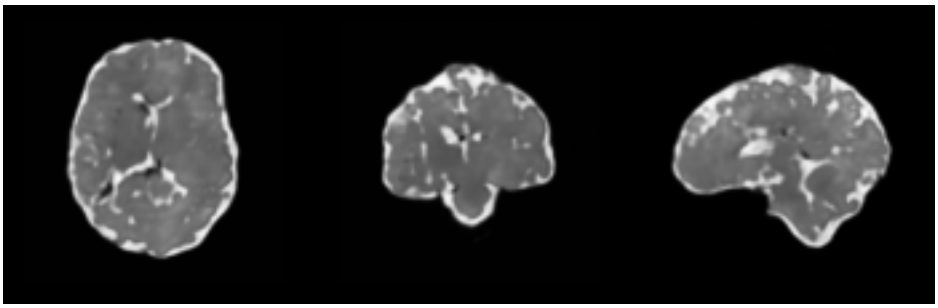
####################################
##################56#################



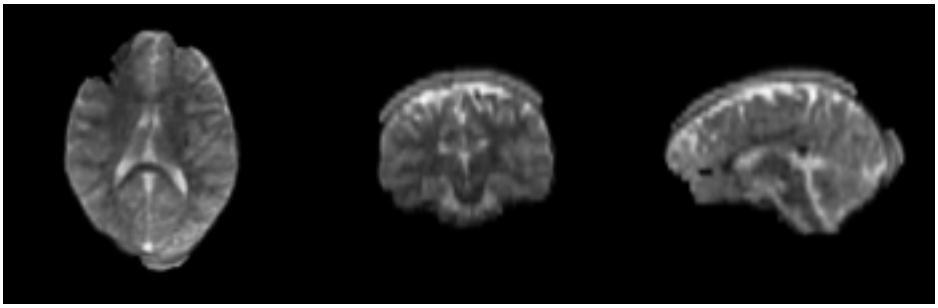


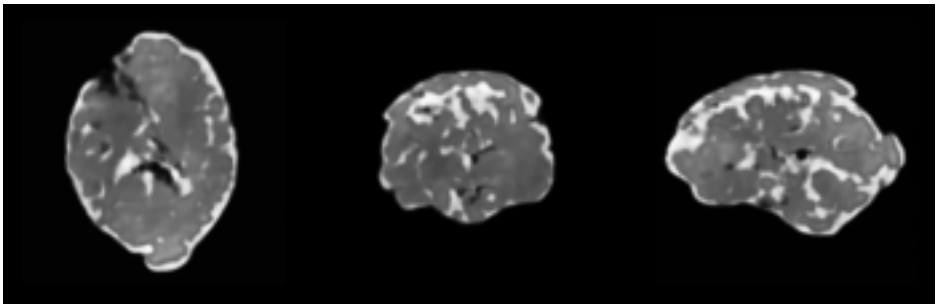
####################################
##################57#################



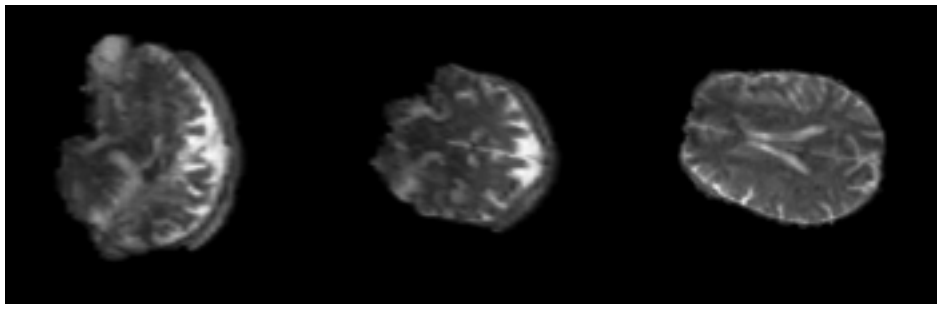


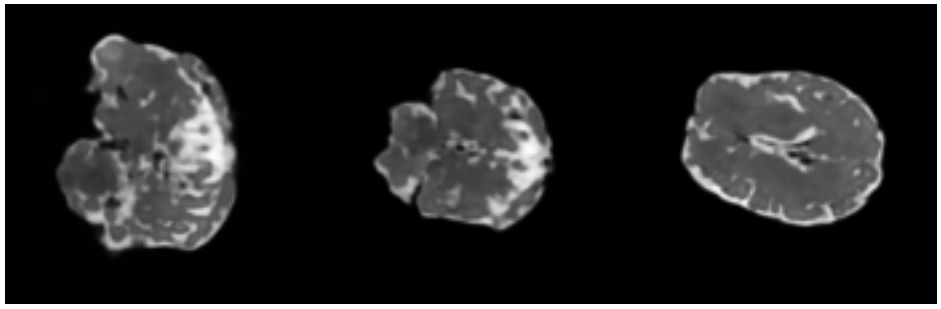
####################################
##################58################



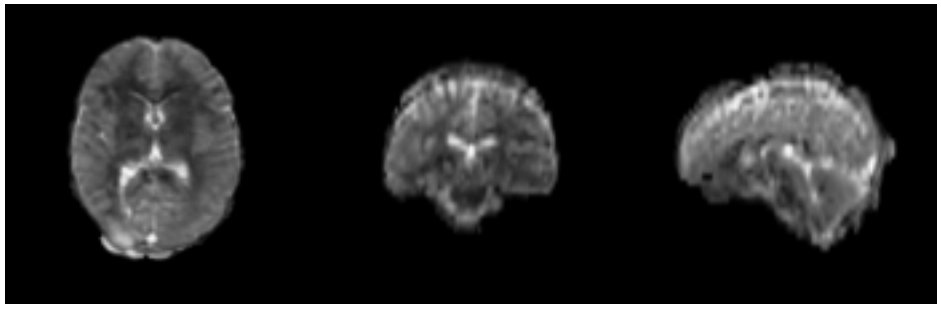


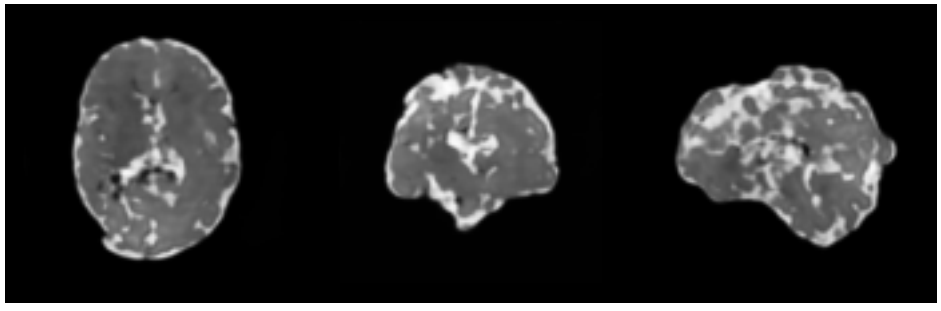
####################################
##################59################



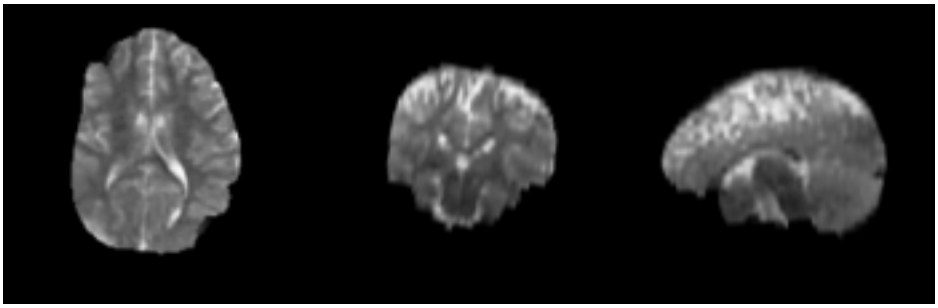


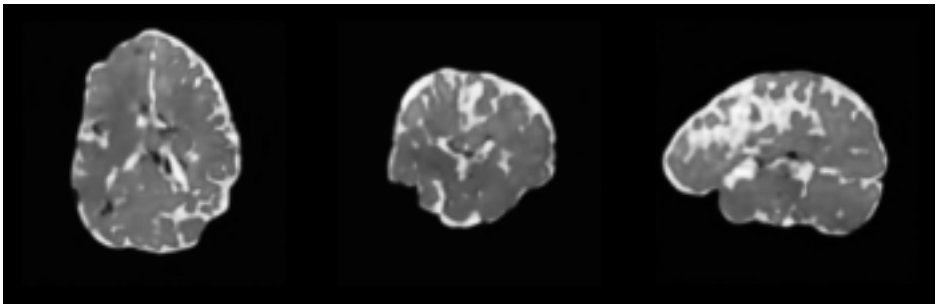
####################################
##################60################



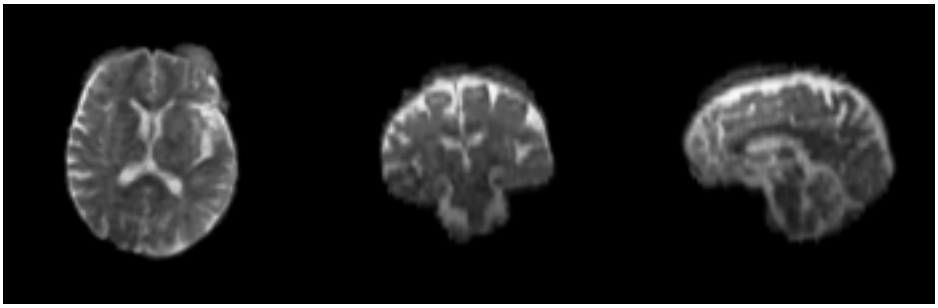


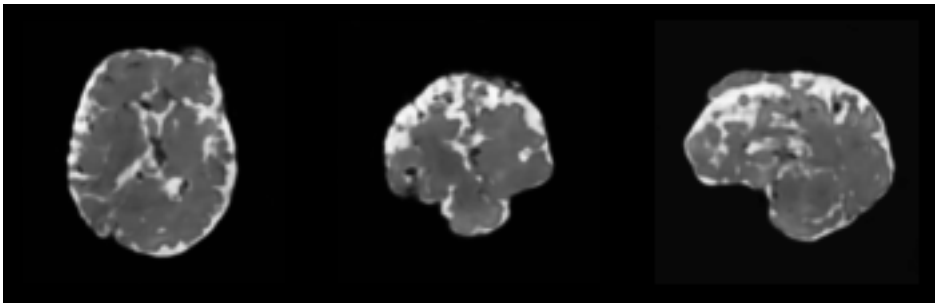
####################################
##################61#################



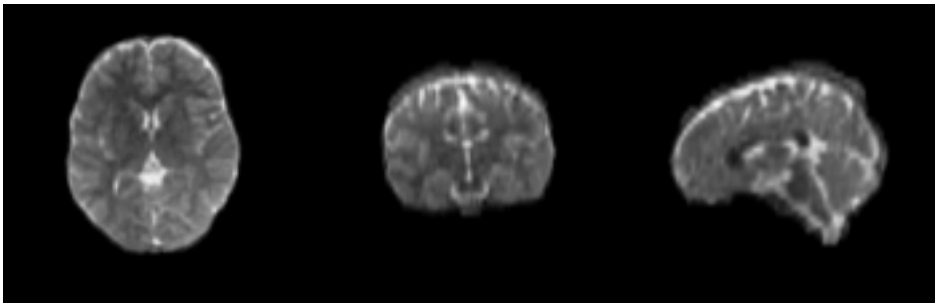


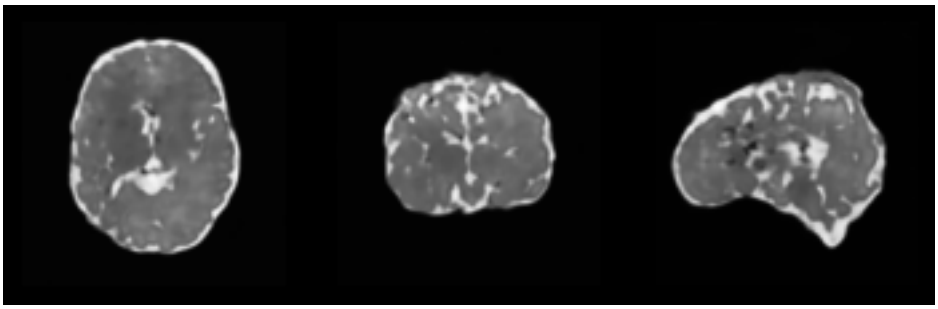
####################################
##################62#################



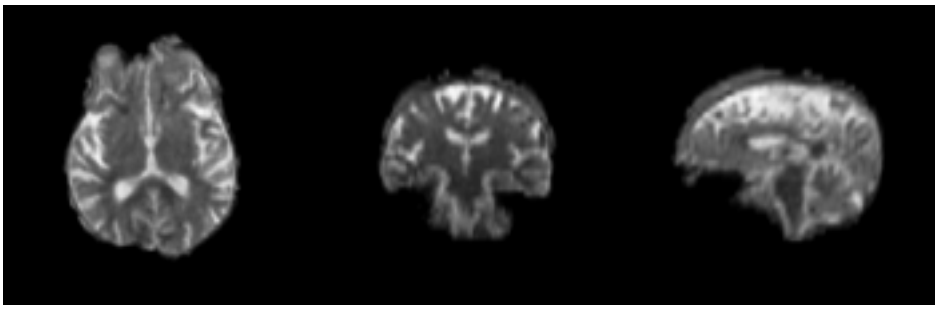


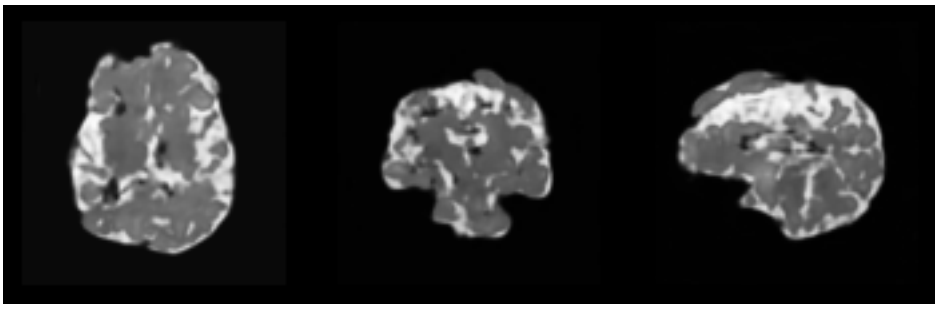
####################################
#################63#################



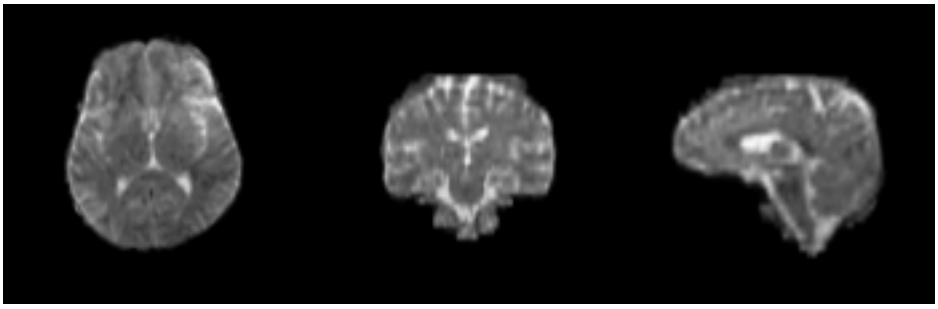


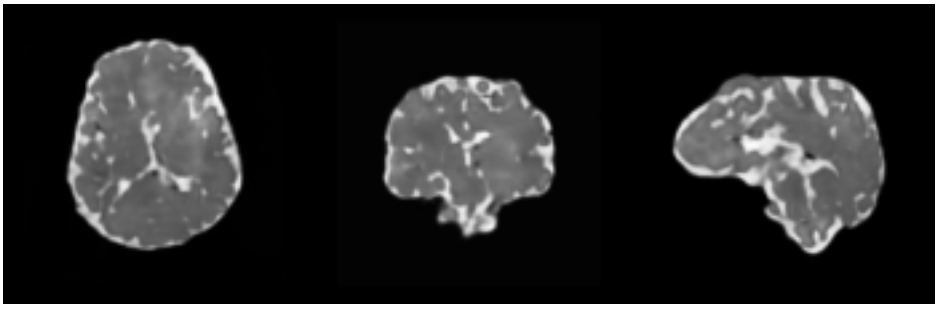
####################################
#################64#################



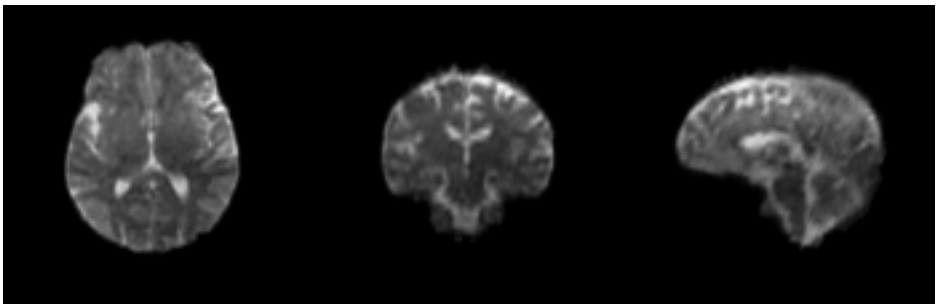


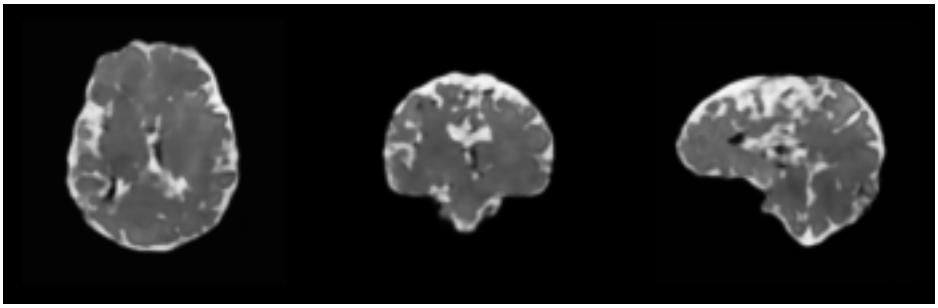
####################################
##################65#################



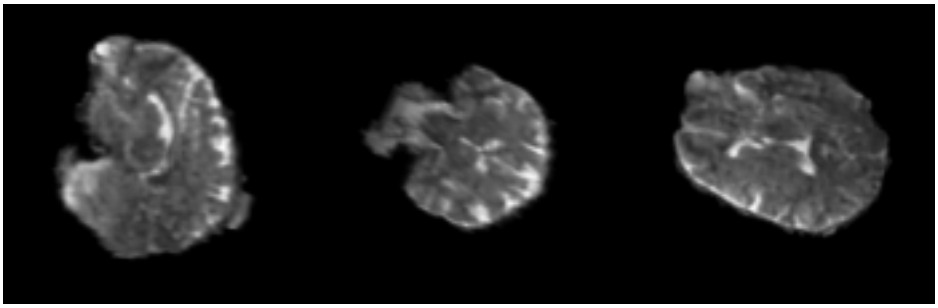


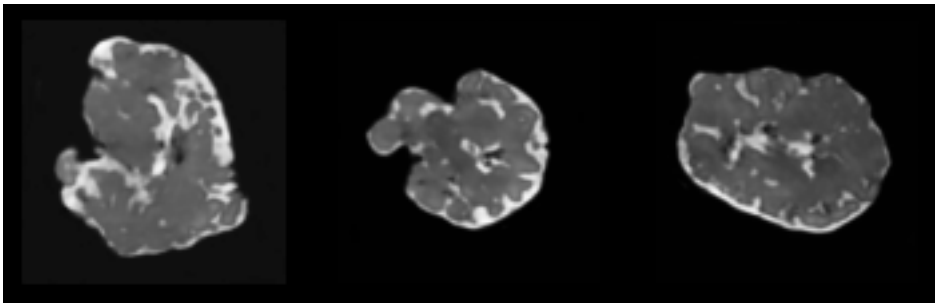
####################################
##################66#################



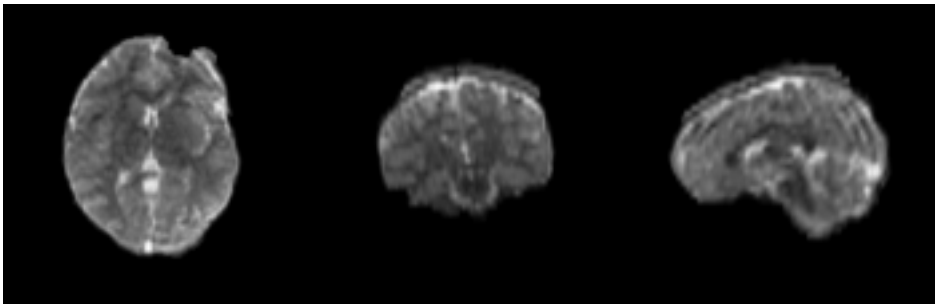


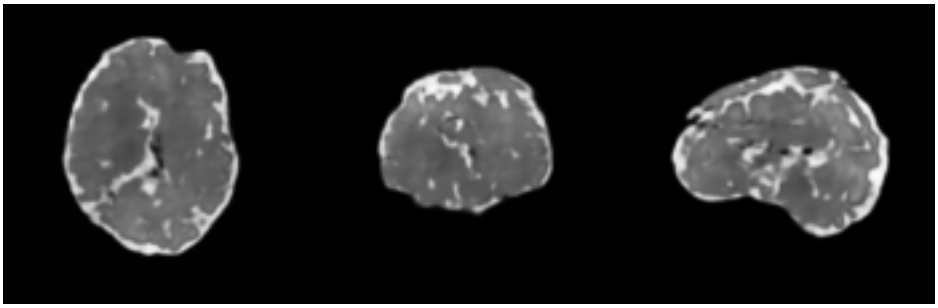
####################################
#################67#################



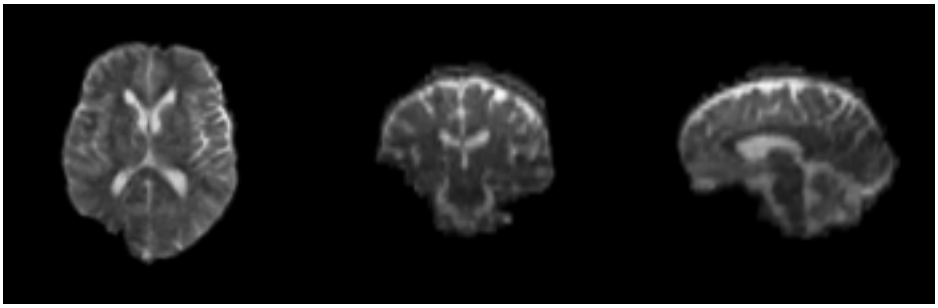


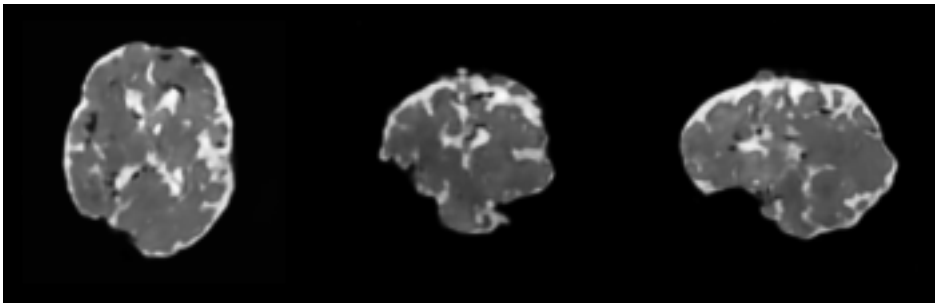
####################################
##################68################



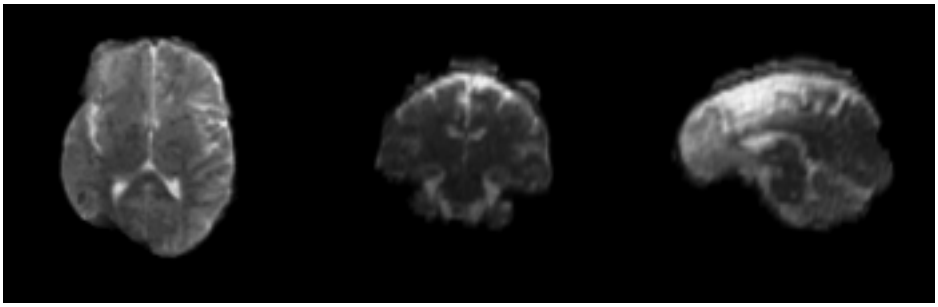


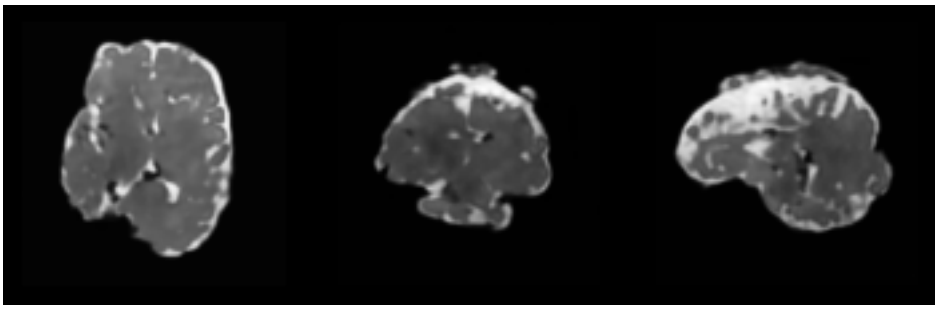
####################################
##################69################



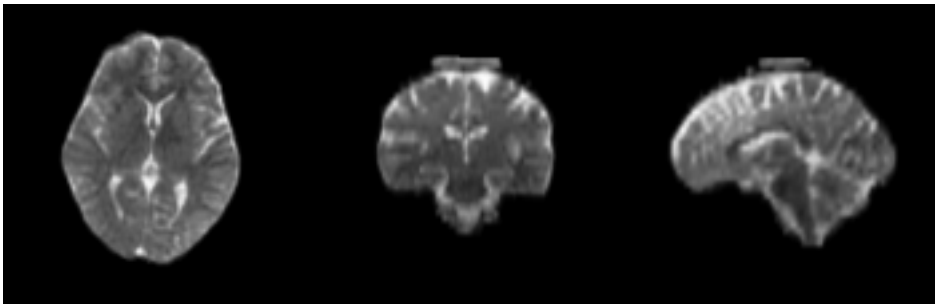


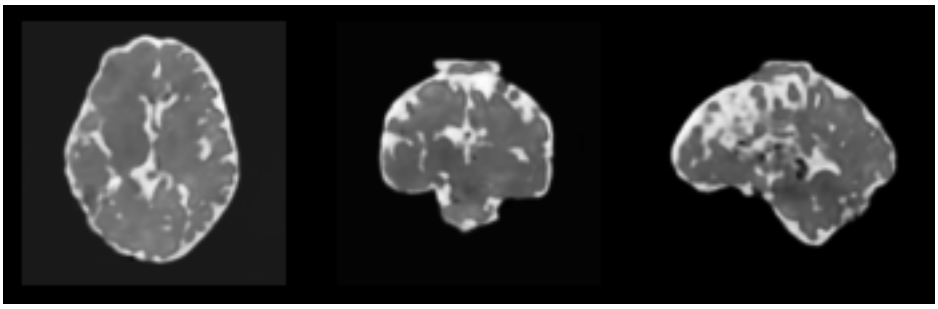
##################################
#################70#################



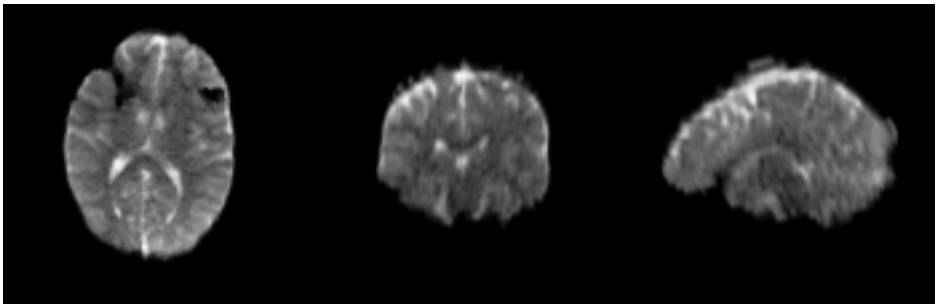


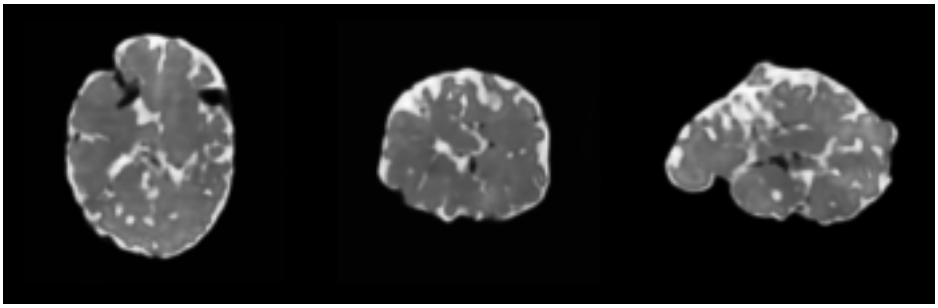
##################################
#################71#################



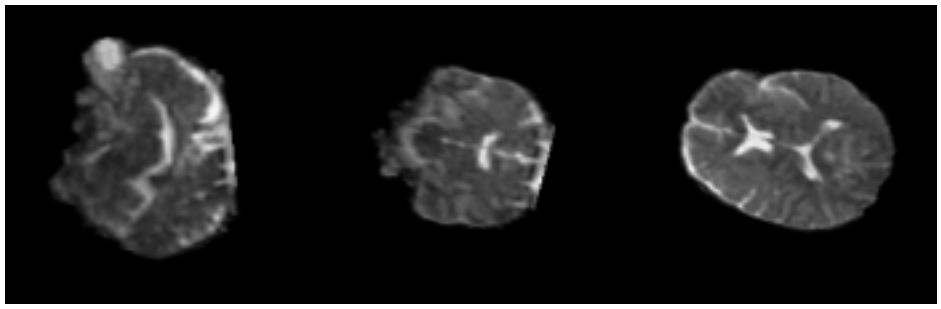


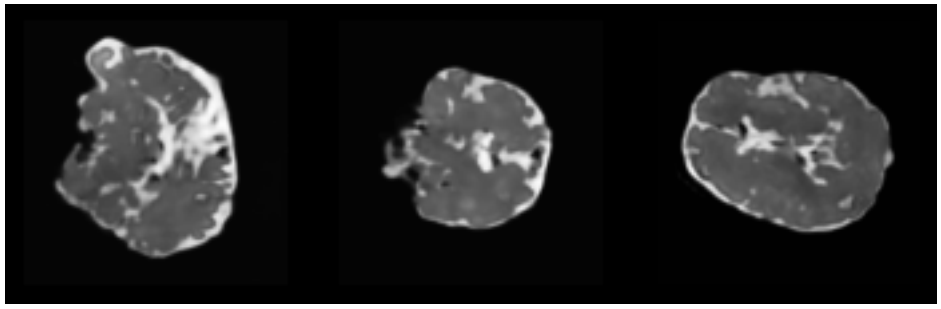
####################################
#################72#################



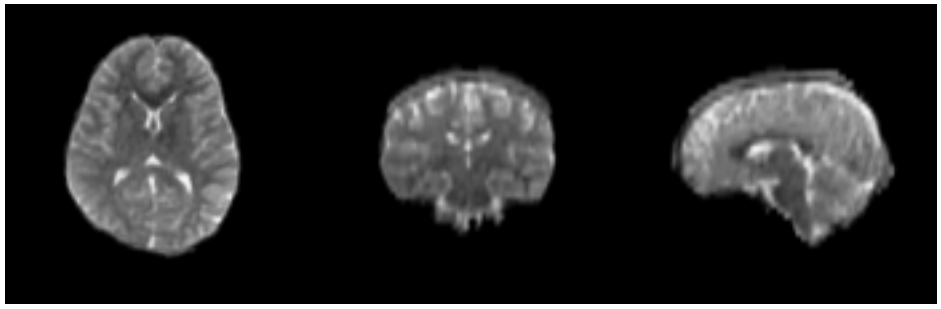


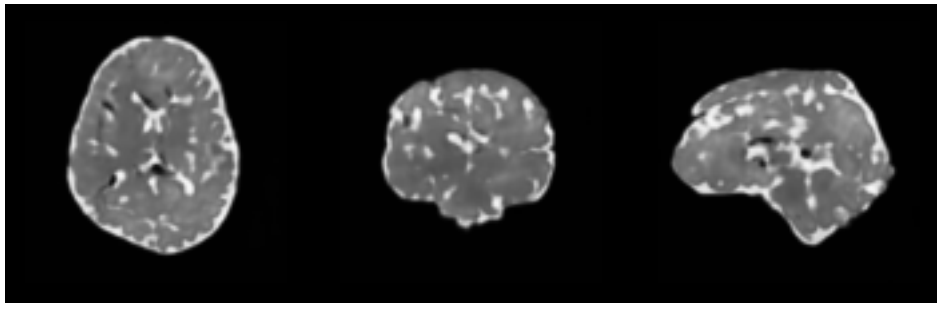
####################################
#################73#################



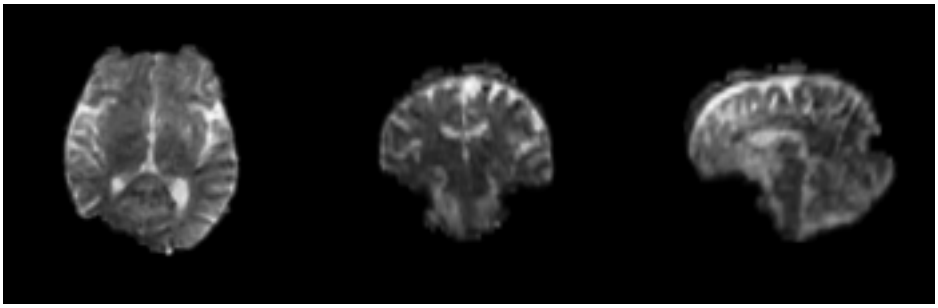


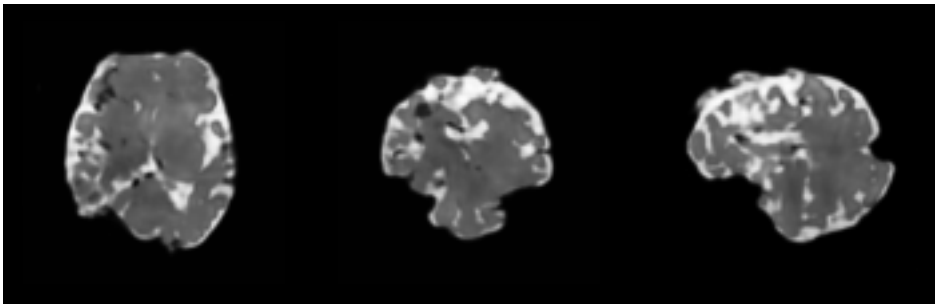
####################################
##################74################



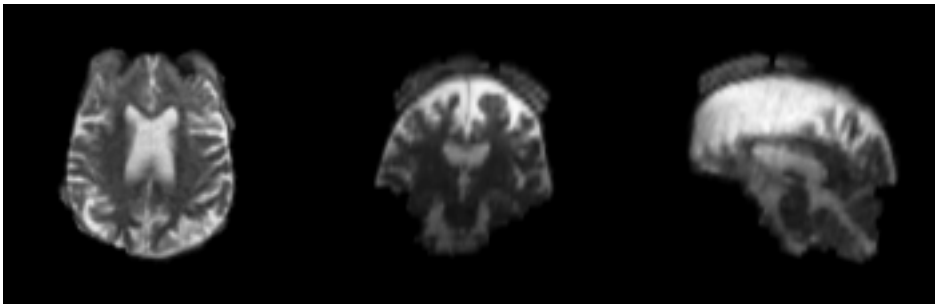


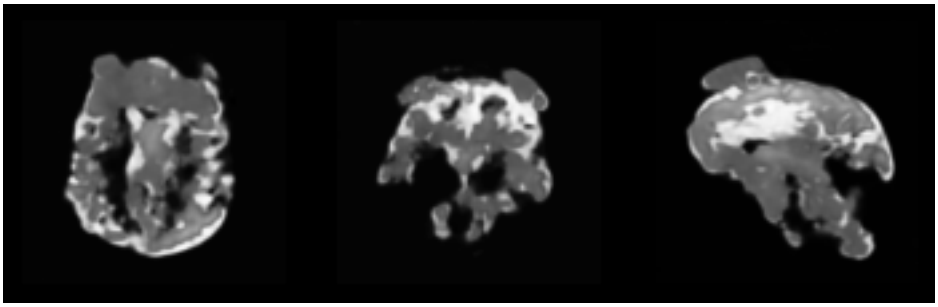
####################################
#################75#################



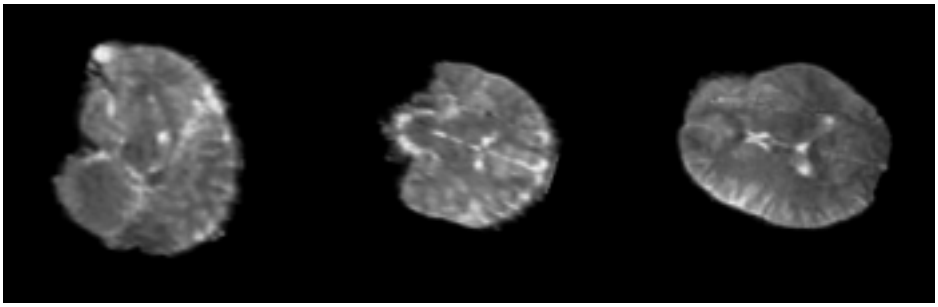


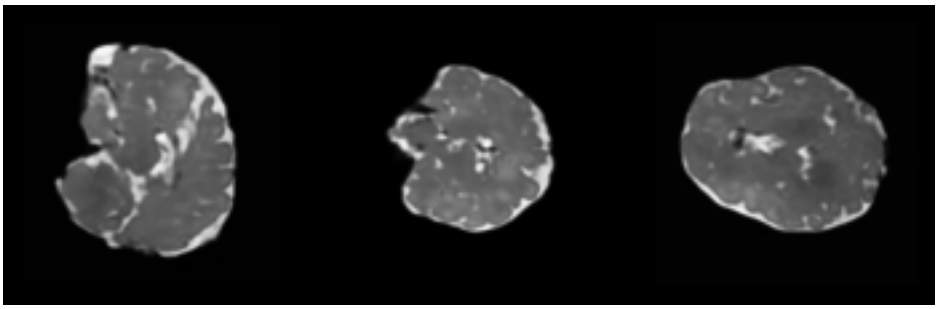
####################################
#################76#################



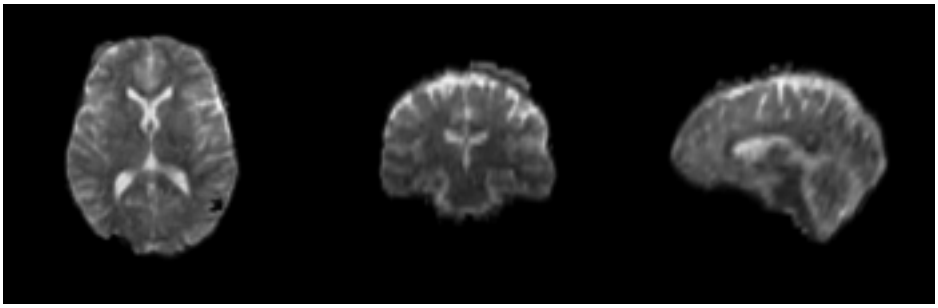


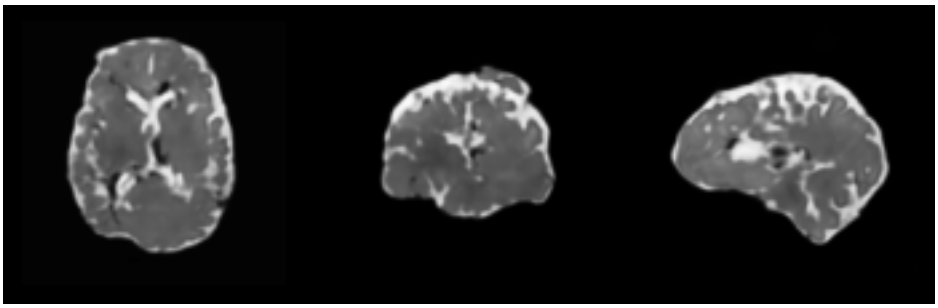
####################################
#################77#################



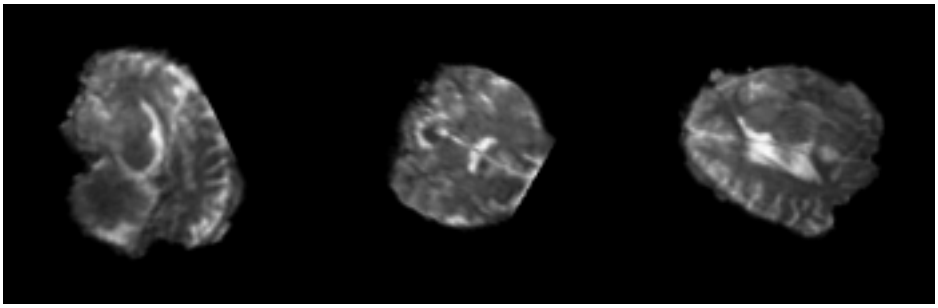


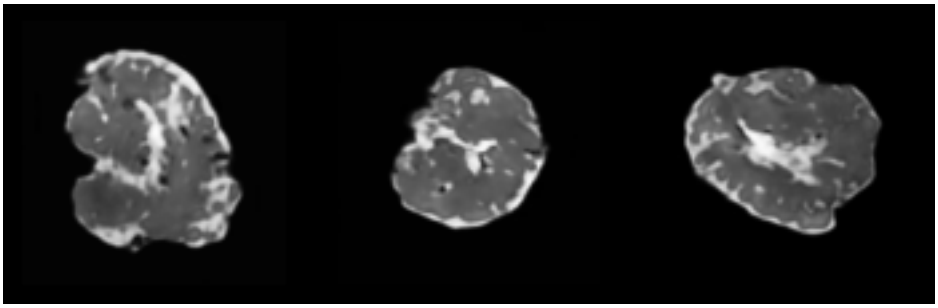
####################################
#################78#################



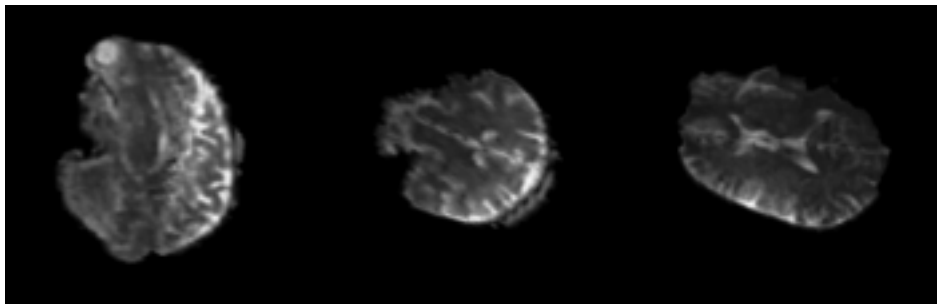


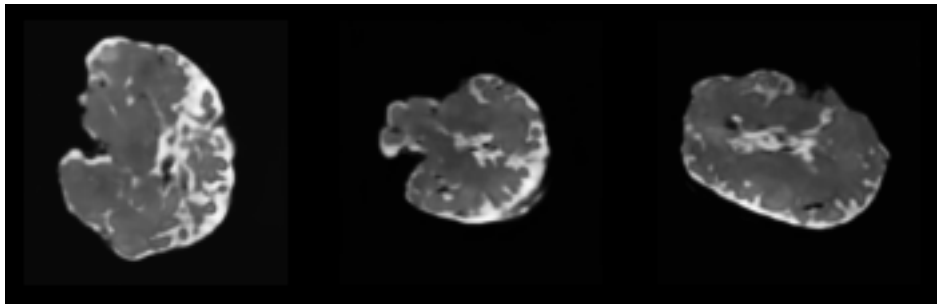
####################################
##################79#################



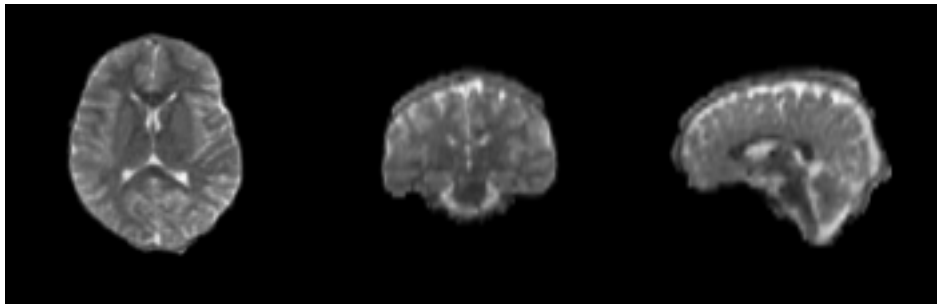


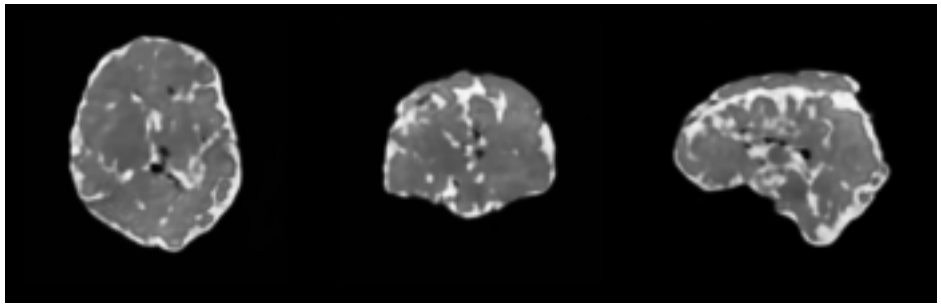
####################################
##################80#################



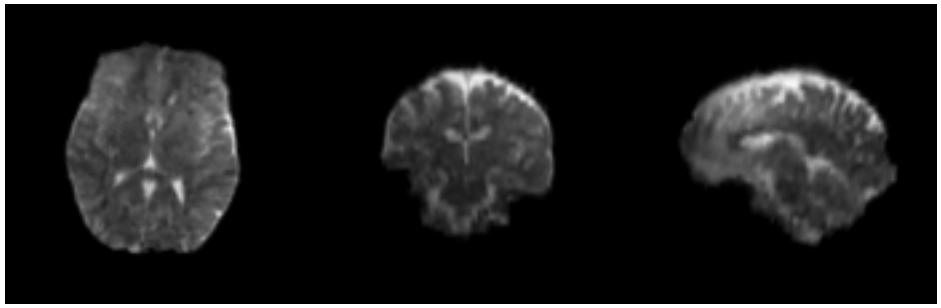


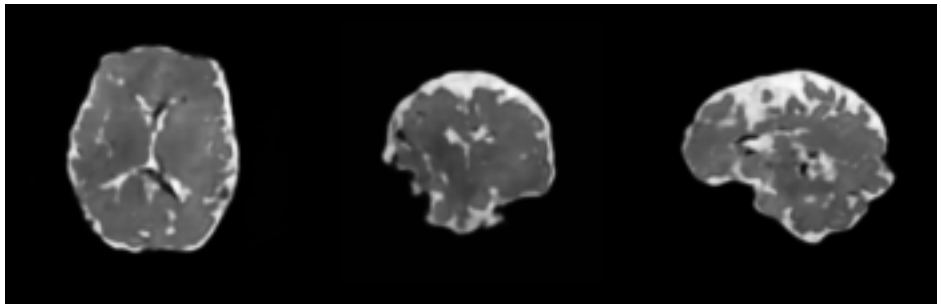
####################################
#################81#################



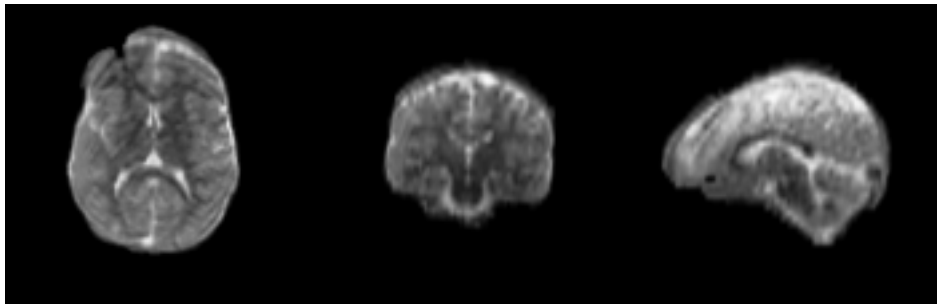


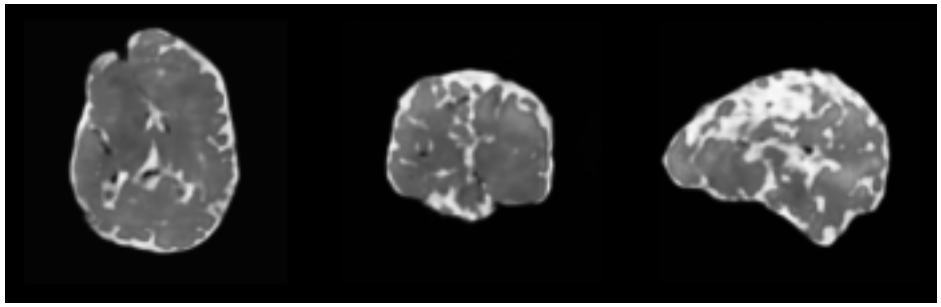
##################################
#################82#################



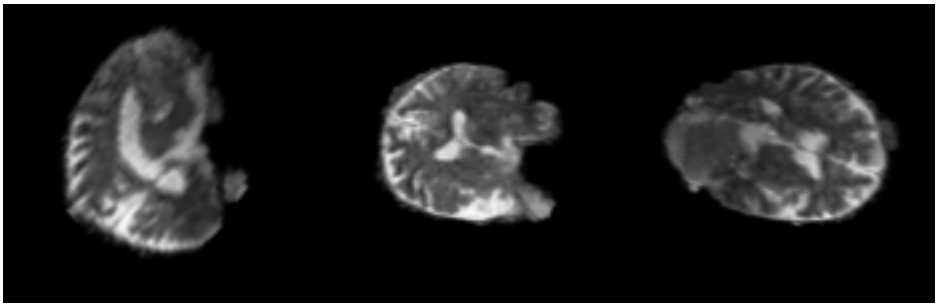


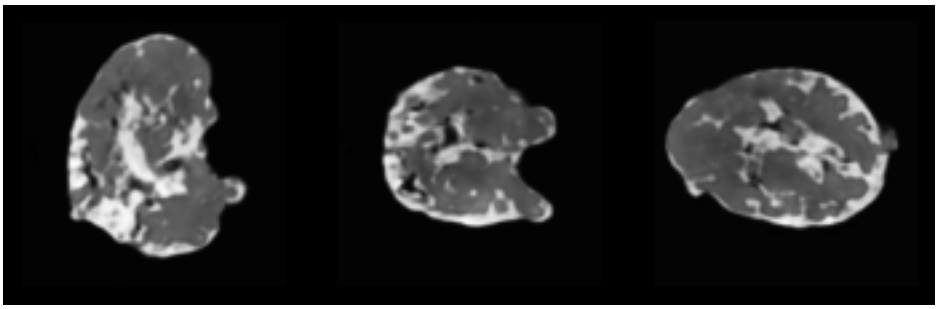
##################################
#################83#################



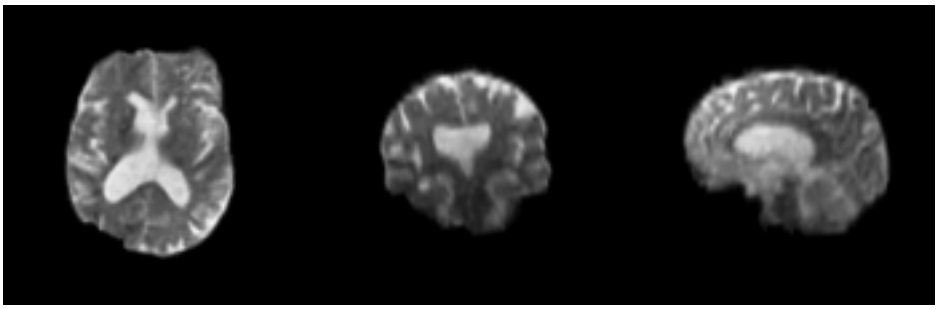


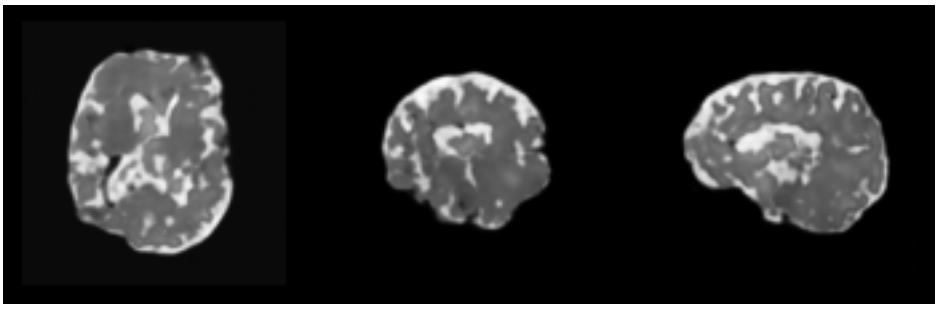
####################################
##################84##################



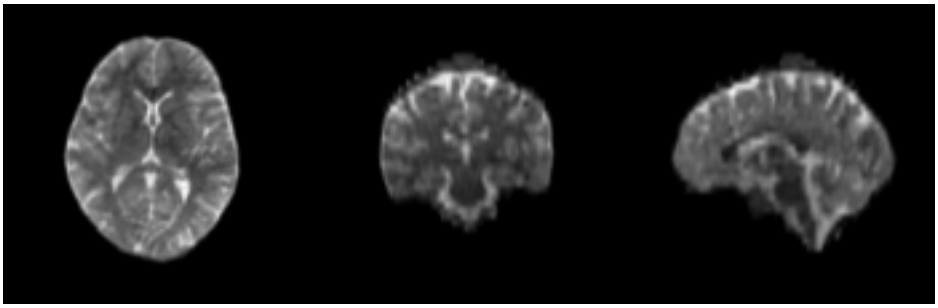


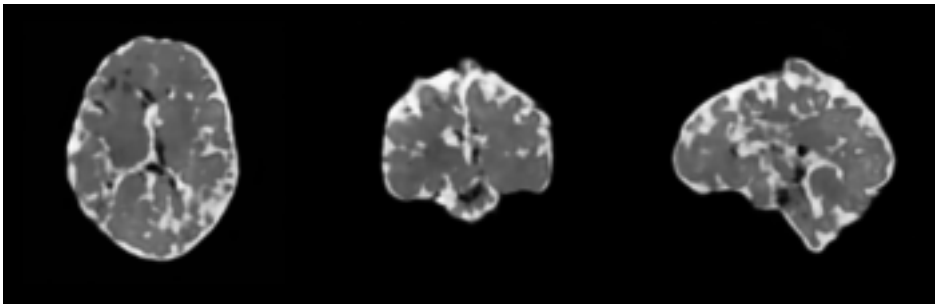
####################################
##################85##################



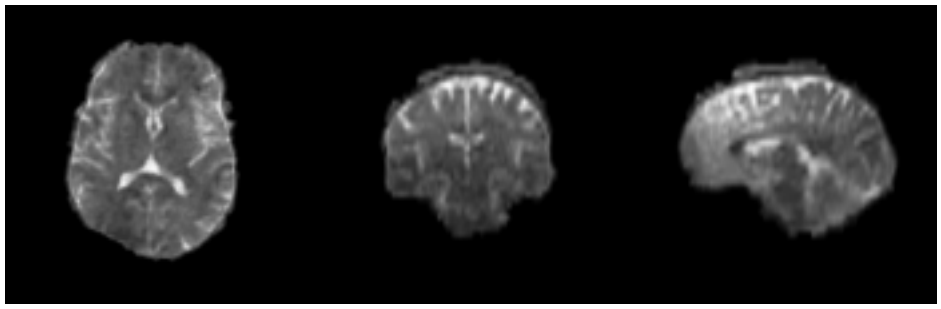


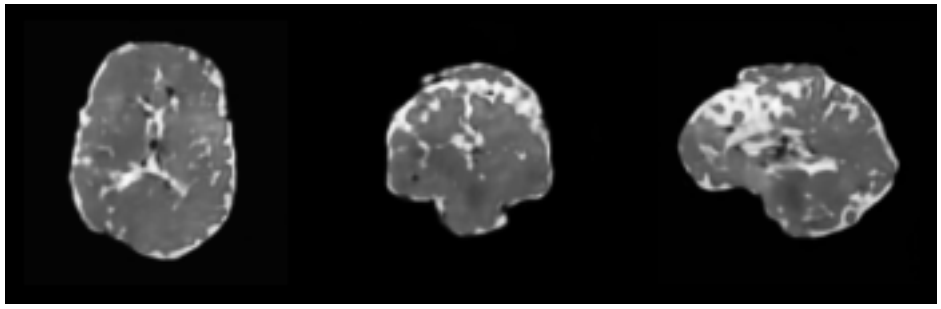
####################################
##################86#################



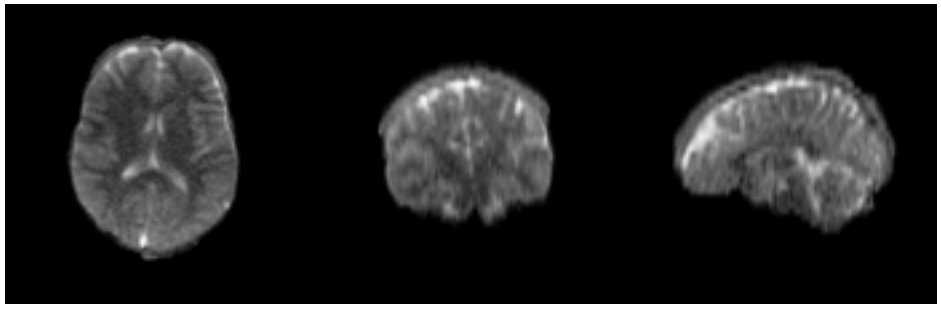


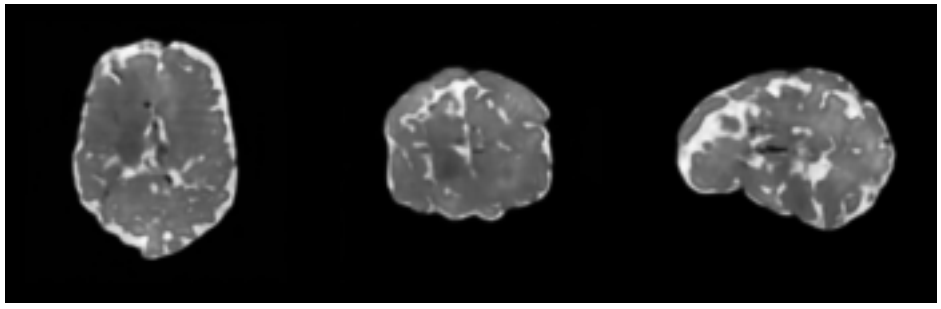
####################################
##################87#################



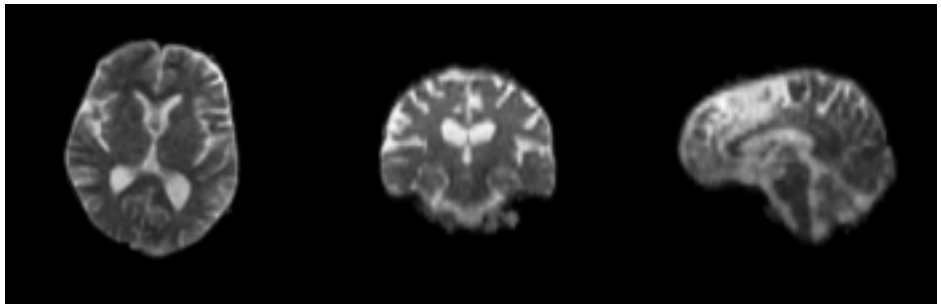


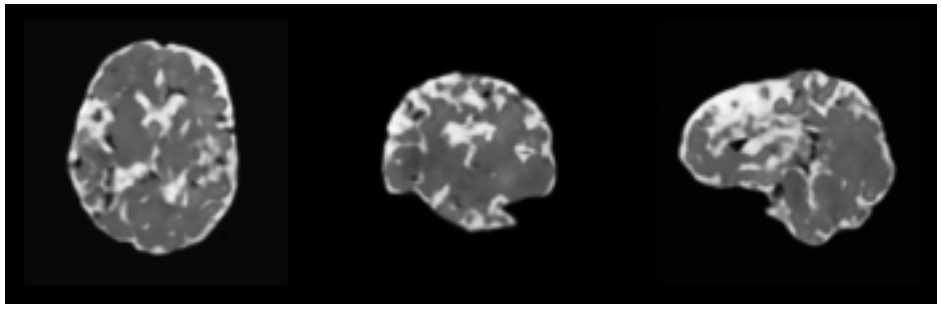
####################################
##################88#################



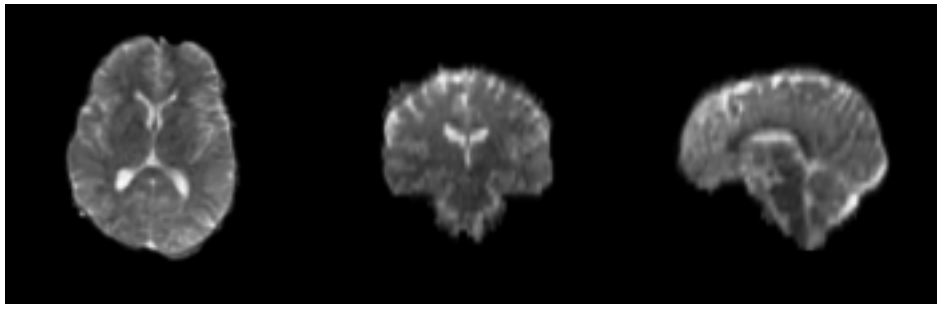


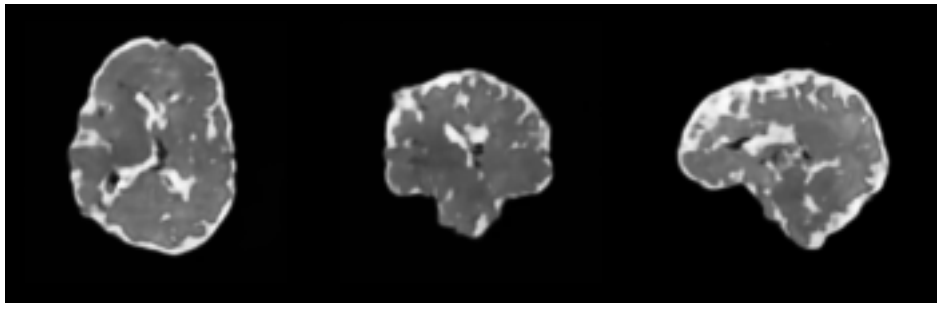
####################################
##################89#################



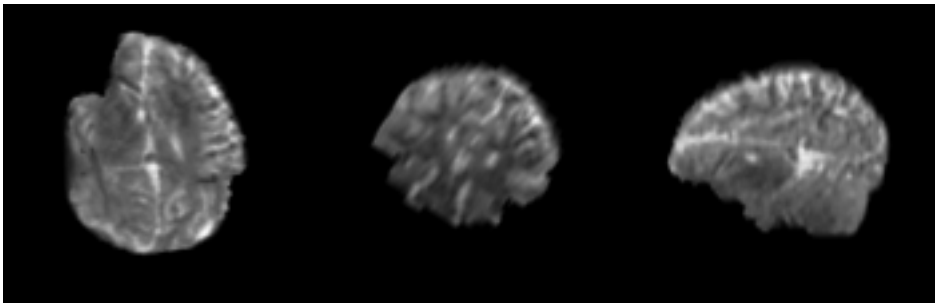


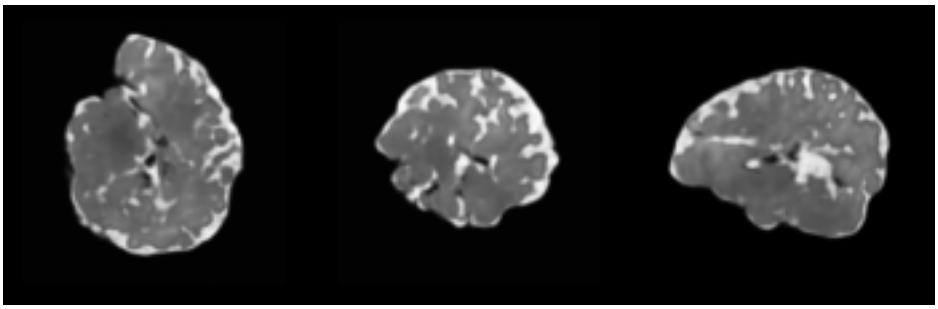
####################################
##################90#################



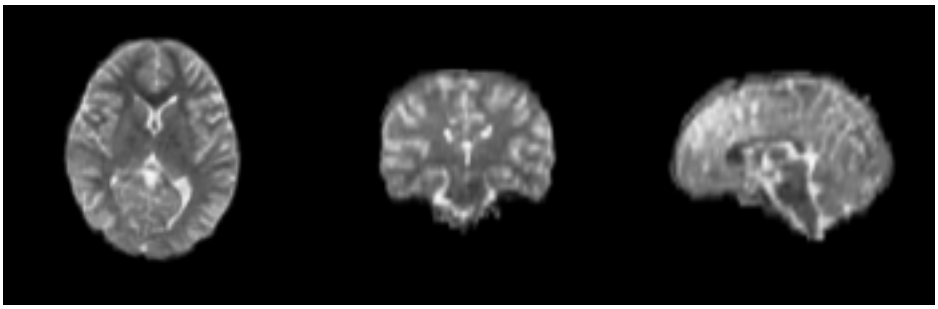


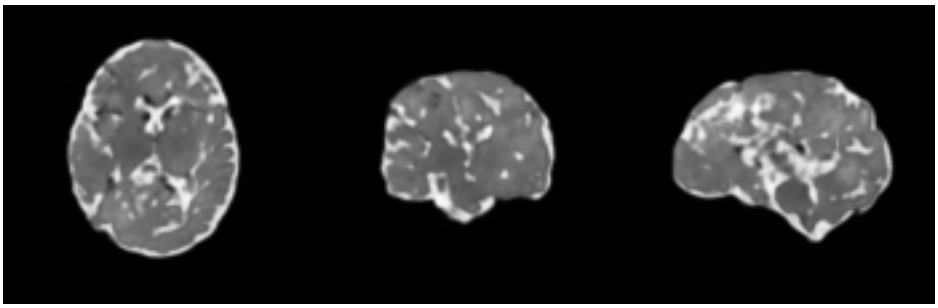
####################################
##################91#################



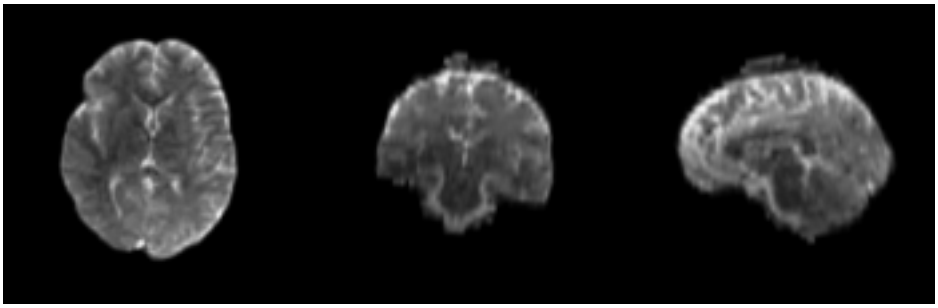


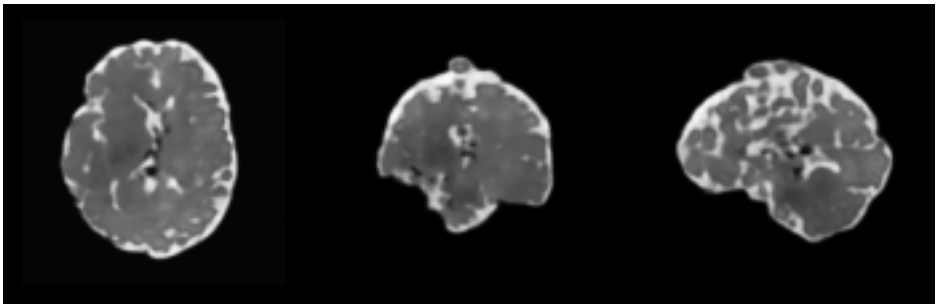
####################################
##################92#################



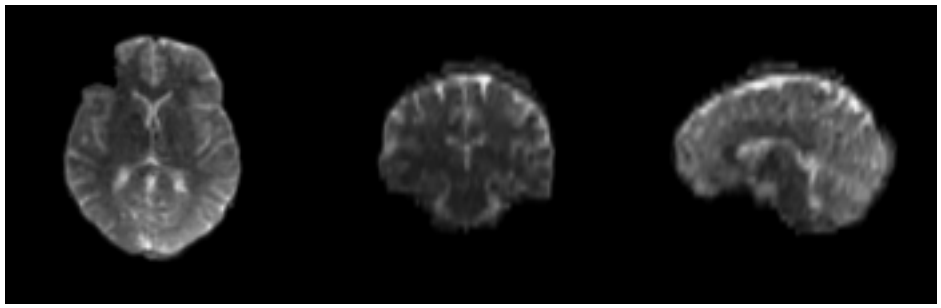


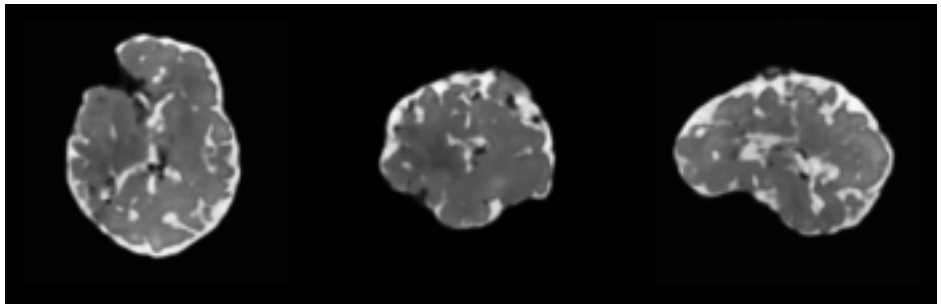
####################################
#################93#################



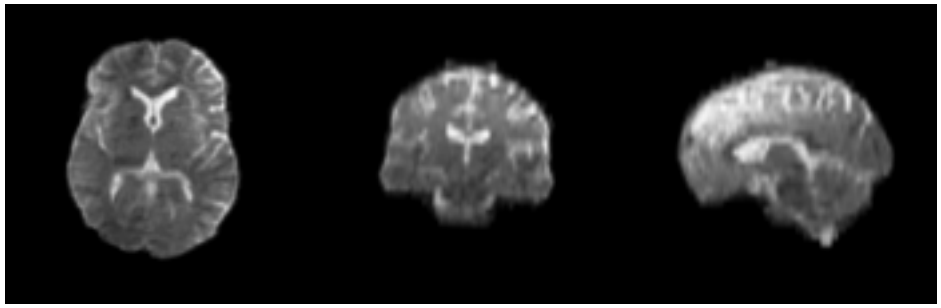


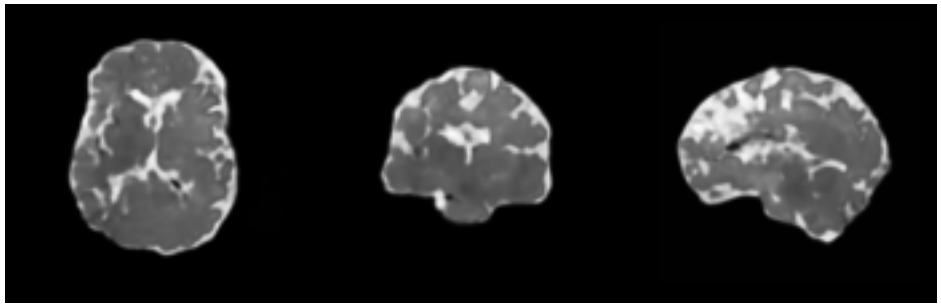
####################################
#################94#################



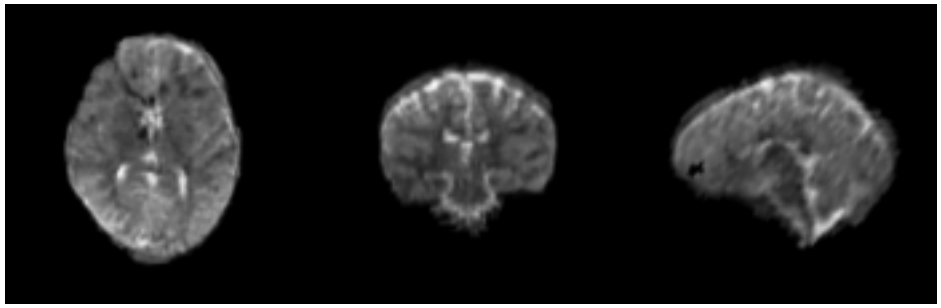


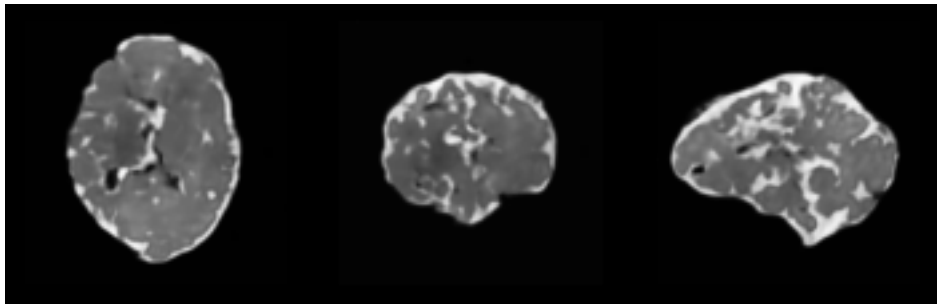
####################################
#################95#################



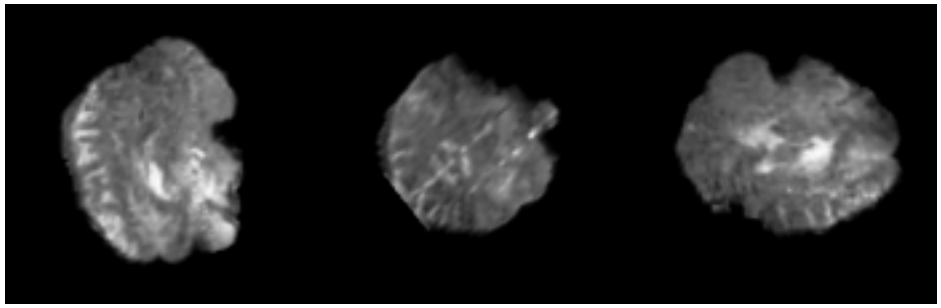


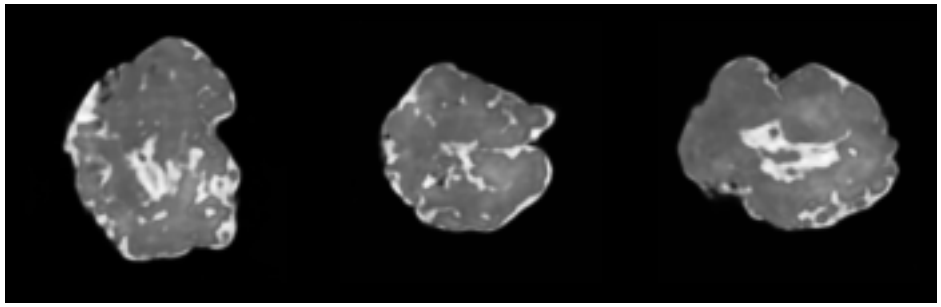
####################################
##################96#################



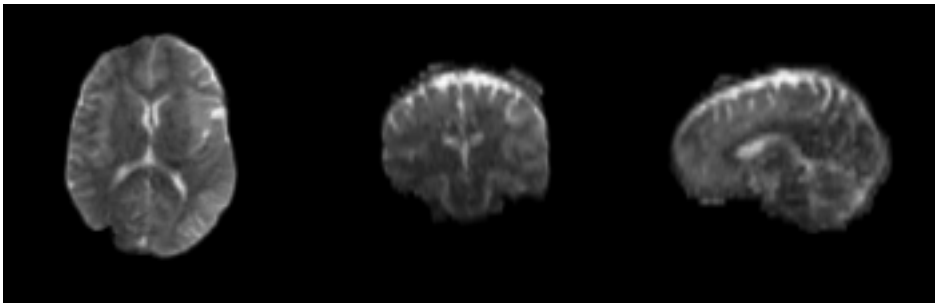


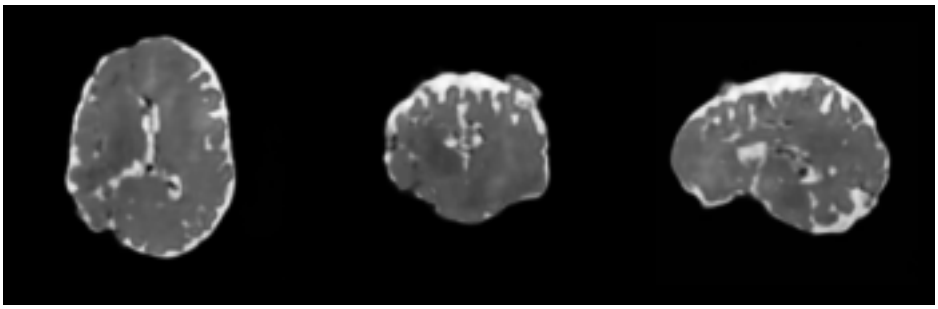
####################################
##################97#################



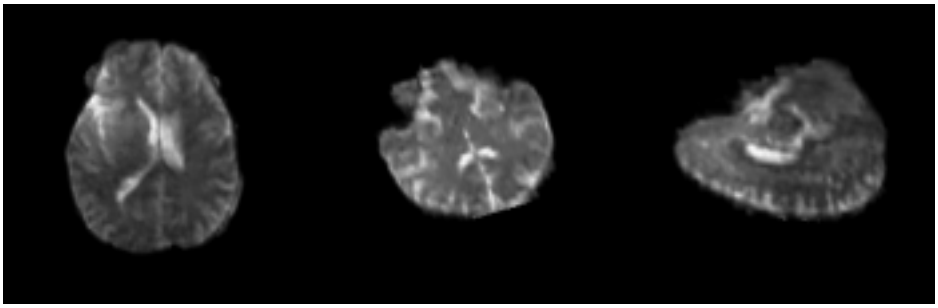


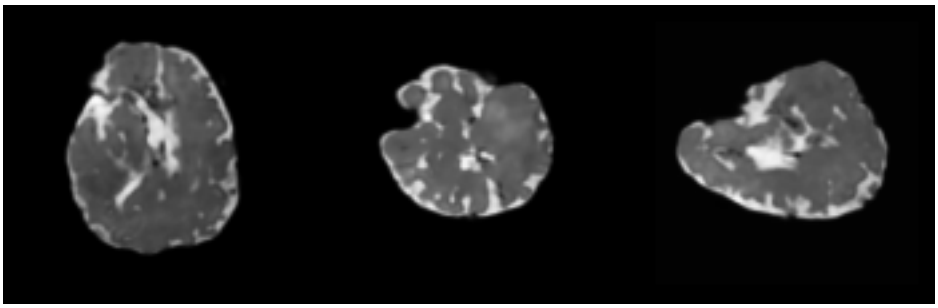
####################################
##################98################



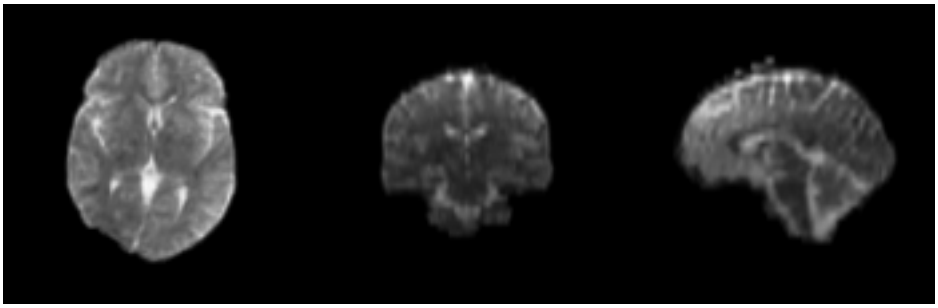


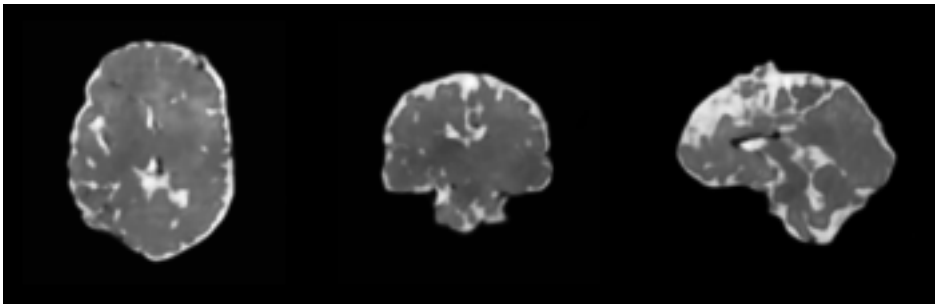
####################################
##################99################



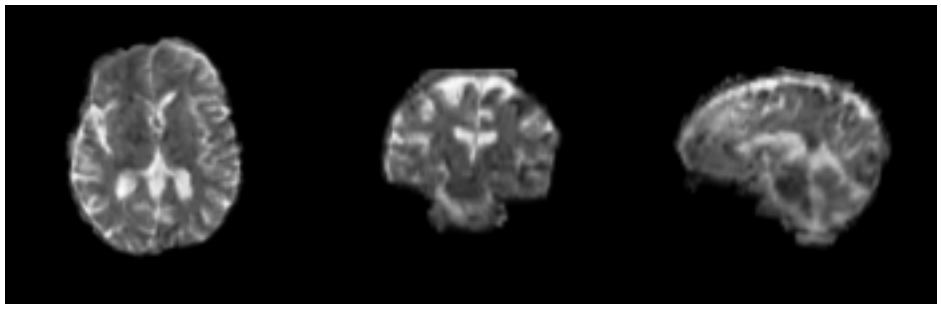


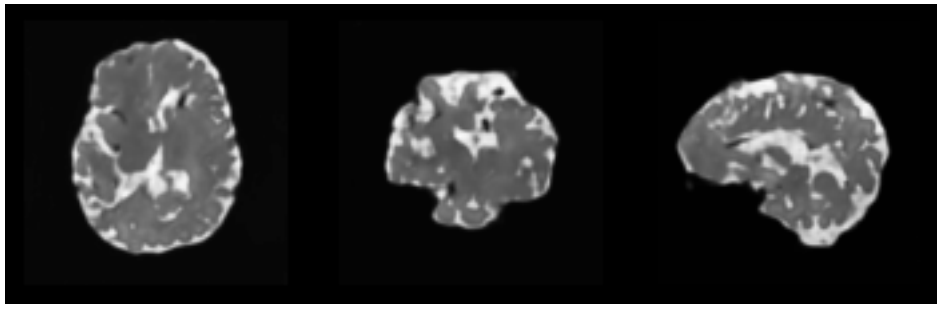
####################################
##################100#################



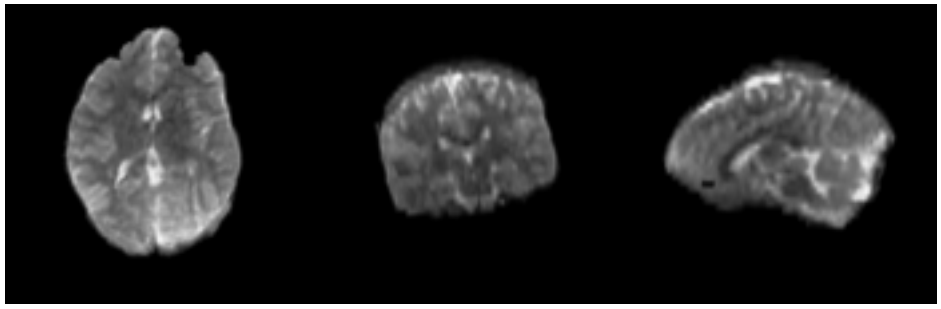


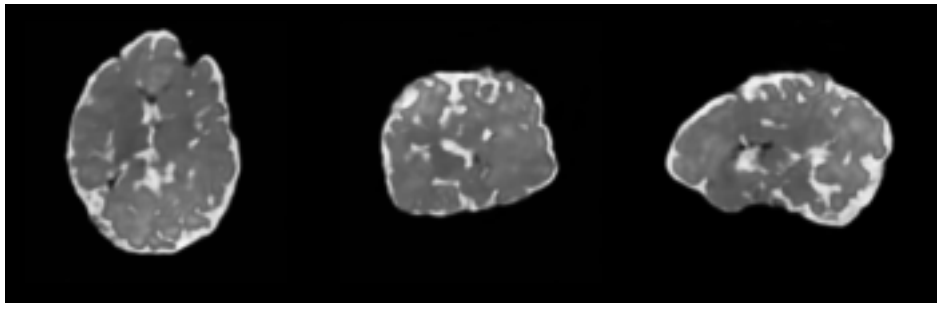
####################################
##################101#################



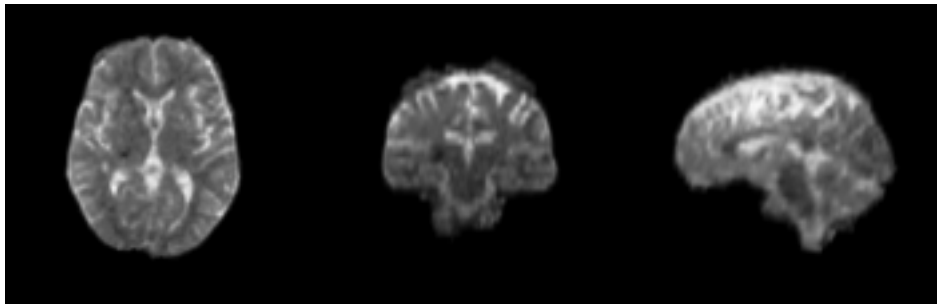


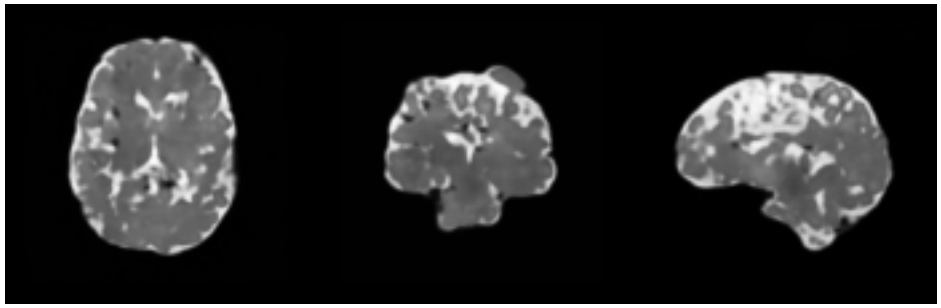
####################################
#################102#################



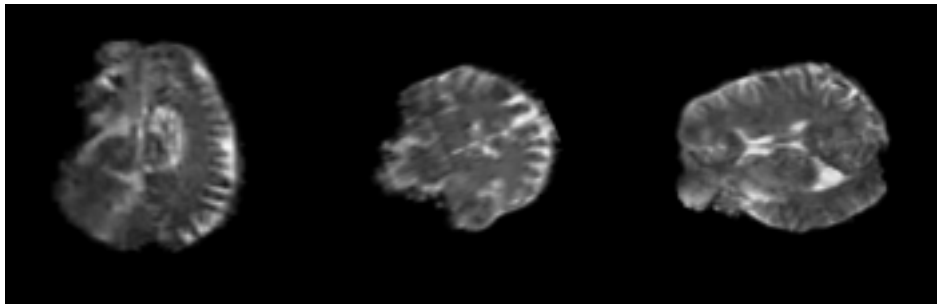


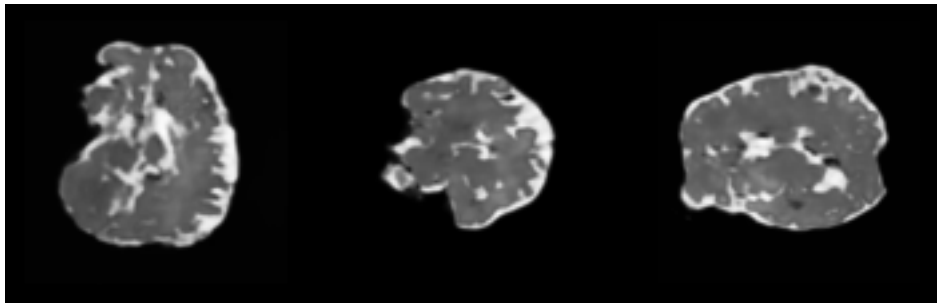
####################################
##################103#################



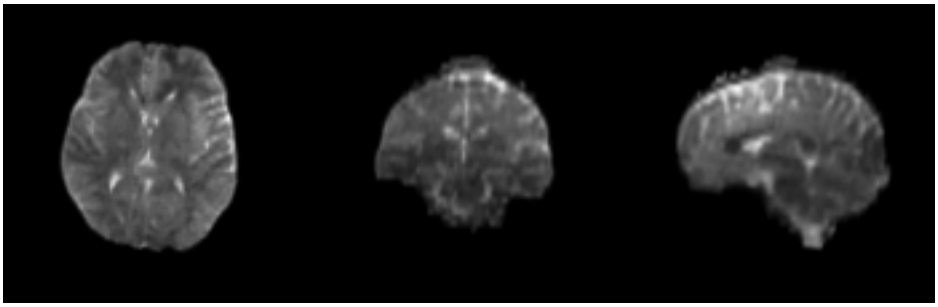


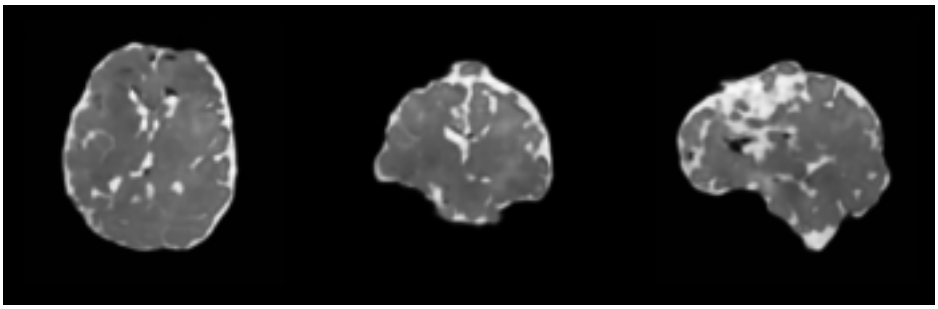
####################################
##################104#################



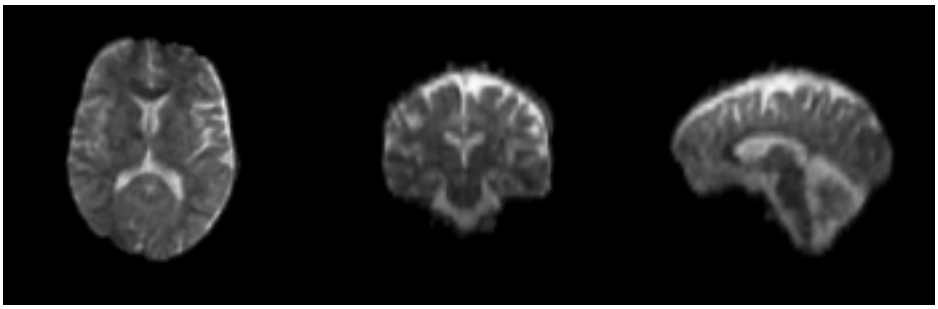


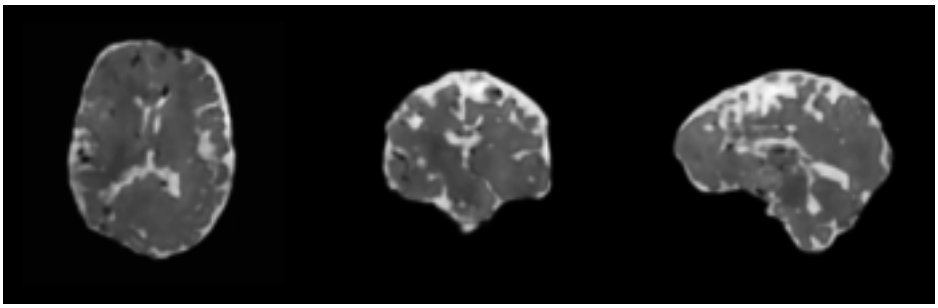
####################################
#################105#################



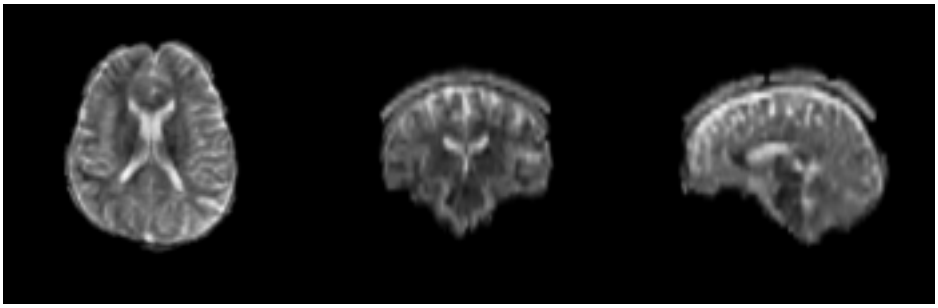


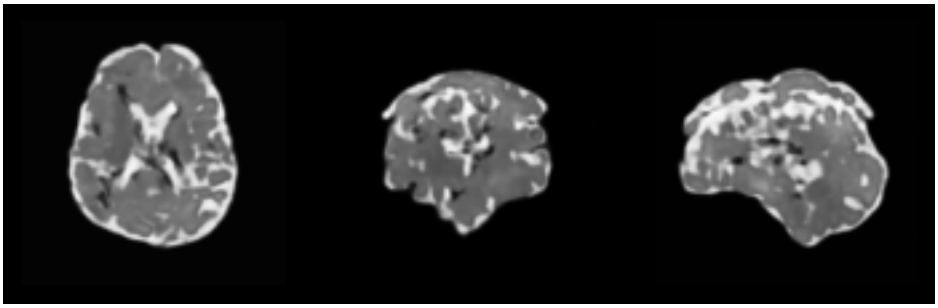
####################################
#################106#################



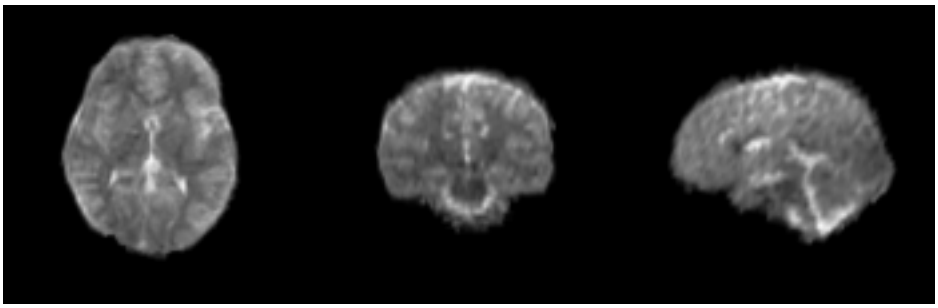


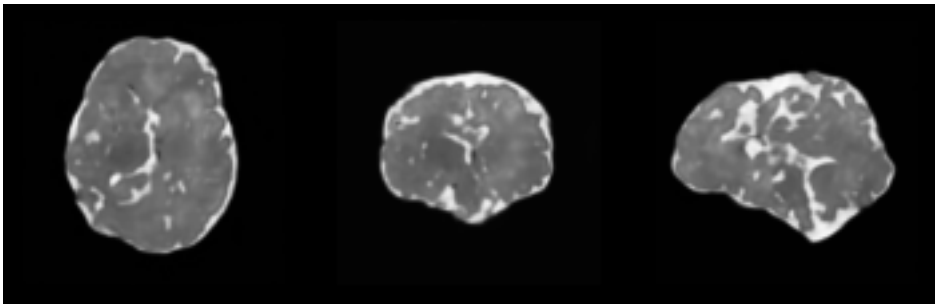
####################################
#################107#################



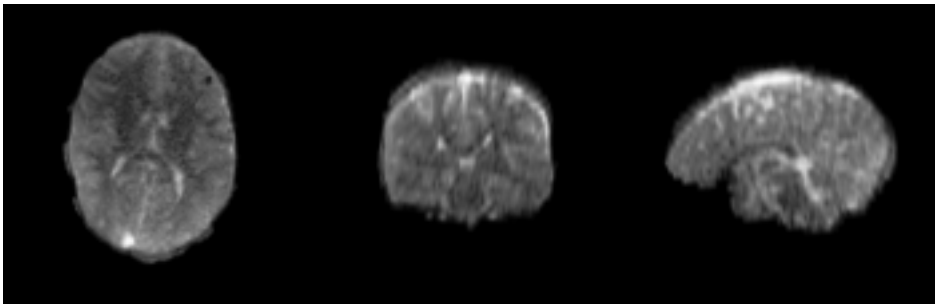


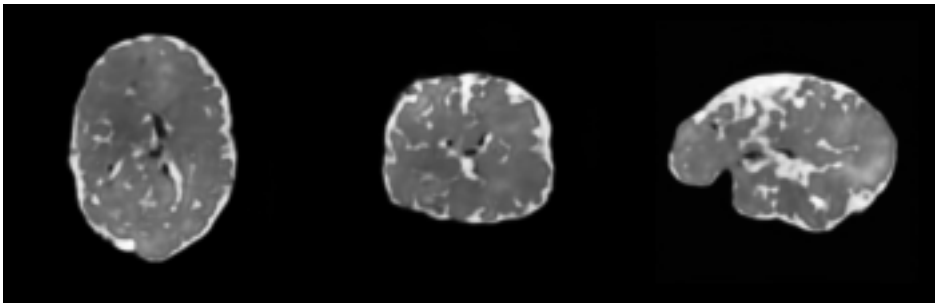
####################################
#################108#################



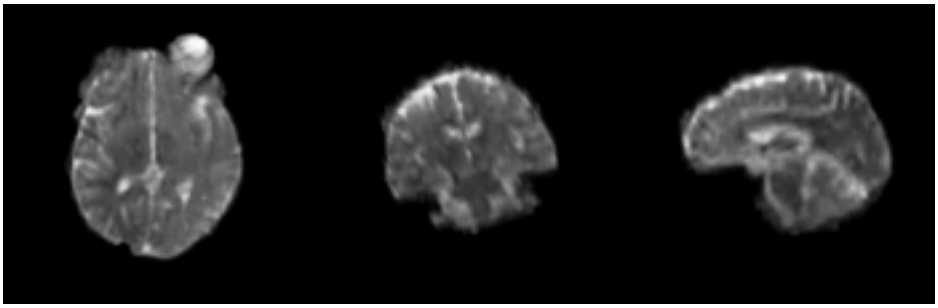


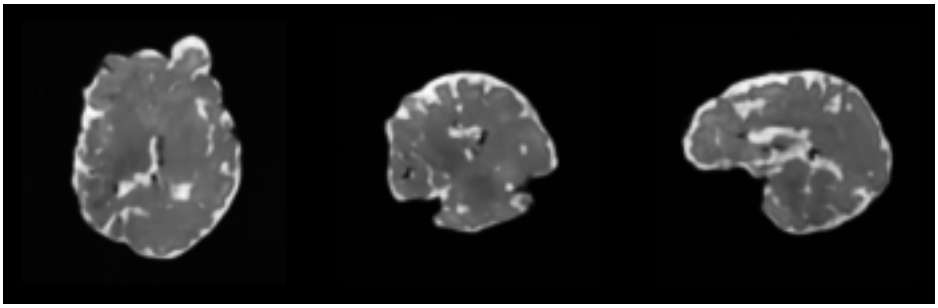
####################################
#################109################



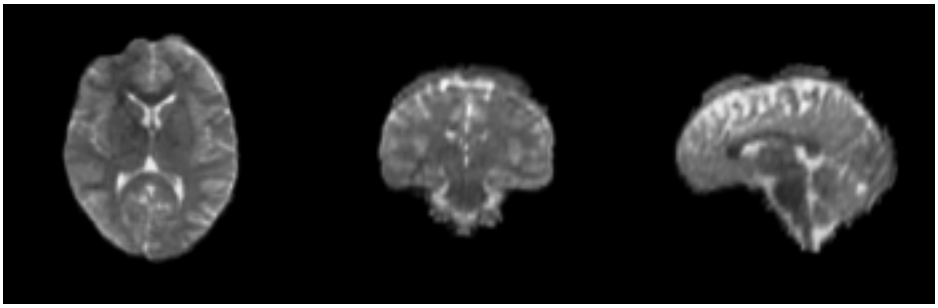


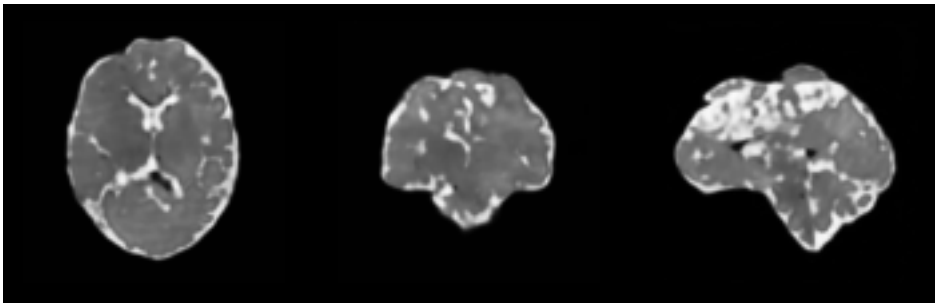
##################################
################110################



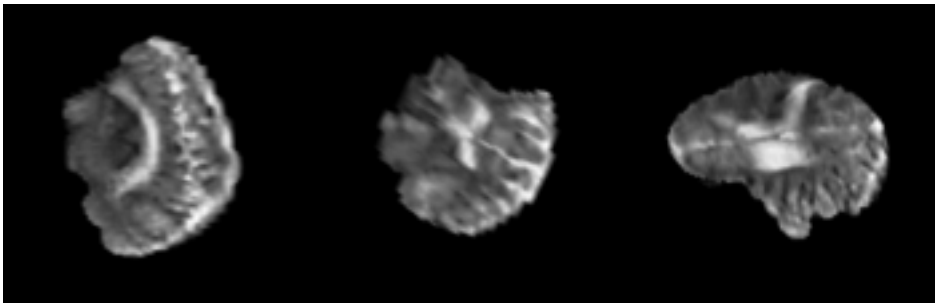


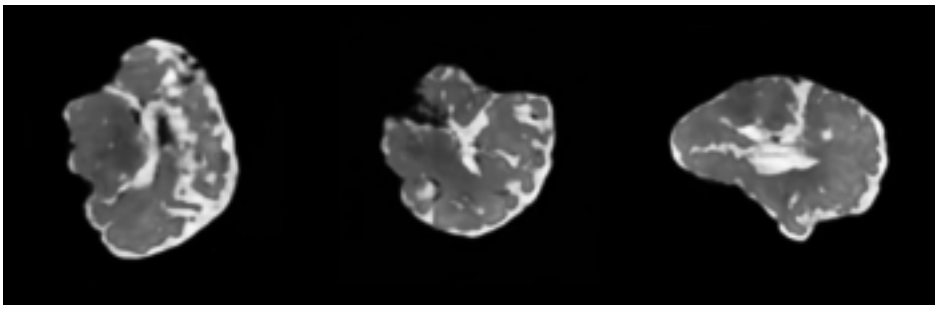
##################################
################111################





####################################
##################112#################





####################################