# Lung Cancer Segmentation

Deep Neural Networks Final Project

Salem AlAthari, Zehui Chen, Kat Desai, Manjit Singh

May 2, 2024

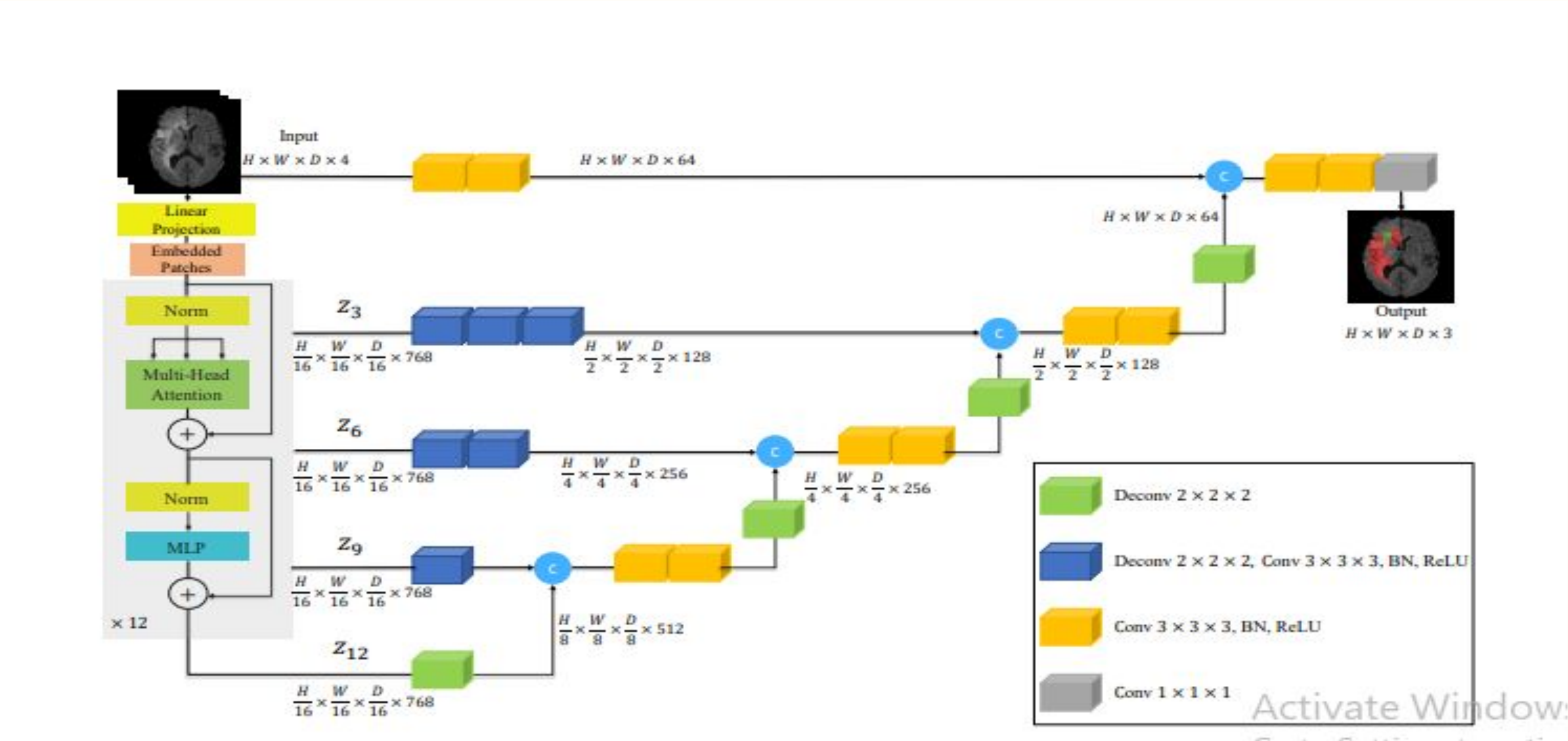
# Research Statement

Comparative Analysis on Multiple Methods to Identify and Segment Lung Cancer Tumors.

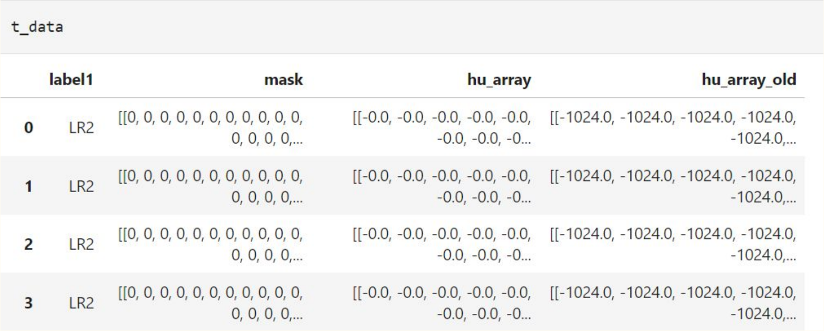
# Current Progress

- Working on a 3D Convolutional Network
- Working on a 2D Convolutional Network
- Data Augmentation

# UNETR

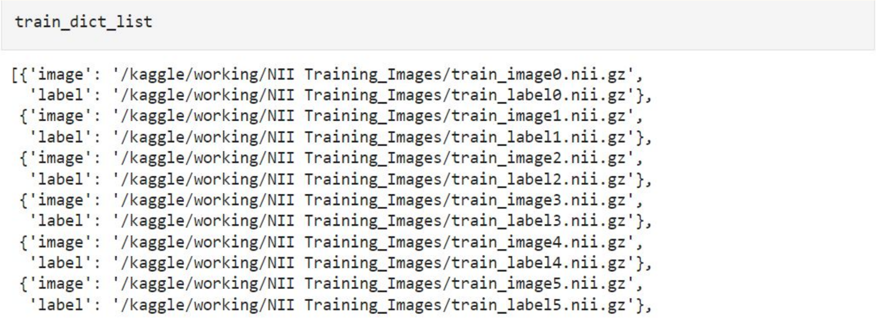

Input
$H \times W \times D \times 4$
Linear Projection
Embedded Patches
Norm
Multi-Head Attention
Norm
MLP
× 12
$H \times W \times D \times 64$
$z_3$
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 128$
$z_6$
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4} \times 256$
$z_9$
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{8} \times \frac{W}{8} \times \frac{D}{8} \times 512$
$z_{12}$
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
C
$\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4} \times 256$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 128$
$H \times W \times D \times 64$
Output
$H \times W \times D \times 3$
🟩 Deconv 2 × 2 × 2
🟦 Deconv 2 × 2 × 2, Conv 3 × 3 × 3, BN, ReLU
🟨 Conv 3 × 3 × 3, BN, ReLU
⬜ Conv 1 × 1 × 1

# 2-D Pkl to 3-D Spatial NII



```
t_data
```

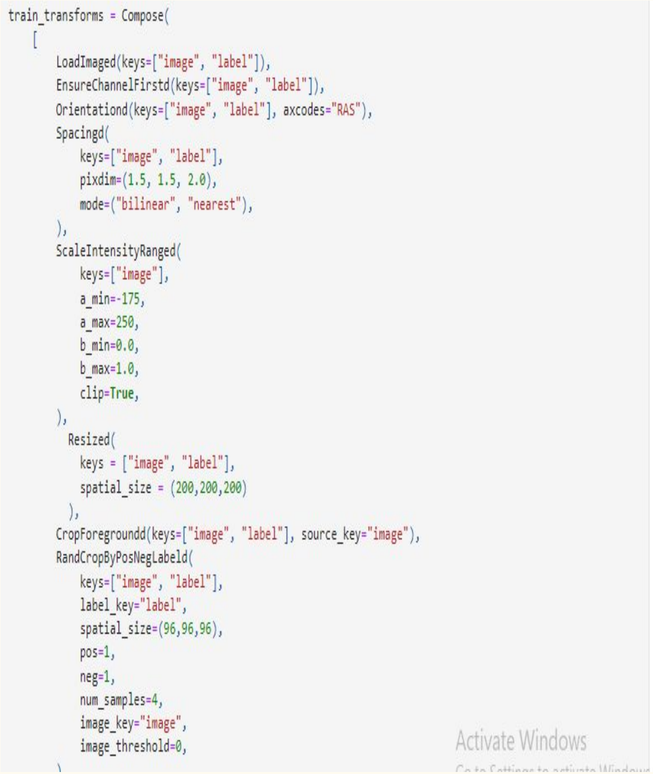
| | label1 | mask | hu_array | hu_array_old |
|---|---|---|---|---|
| **0** | LR2 | [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,... | [[-0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0... | [[-1024.0, -1024.0, -1024.0, -1024.0, -1024.0,... |
| **1** | LR2 | [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,... | [[-0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0... | [[-1024.0, -1024.0, -1024.0, -1024.0, -1024.0,... |
| **2** | LR2 | [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,... | [[-0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0... | [[-1024.0, -1024.0, -1024.0, -1024.0, -1024.0,... |
| **3** | LR2 | [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,... | [[-0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0... | [[-1024.0, -1024.0, -1024.0, -1024.0, -1024.0,... |



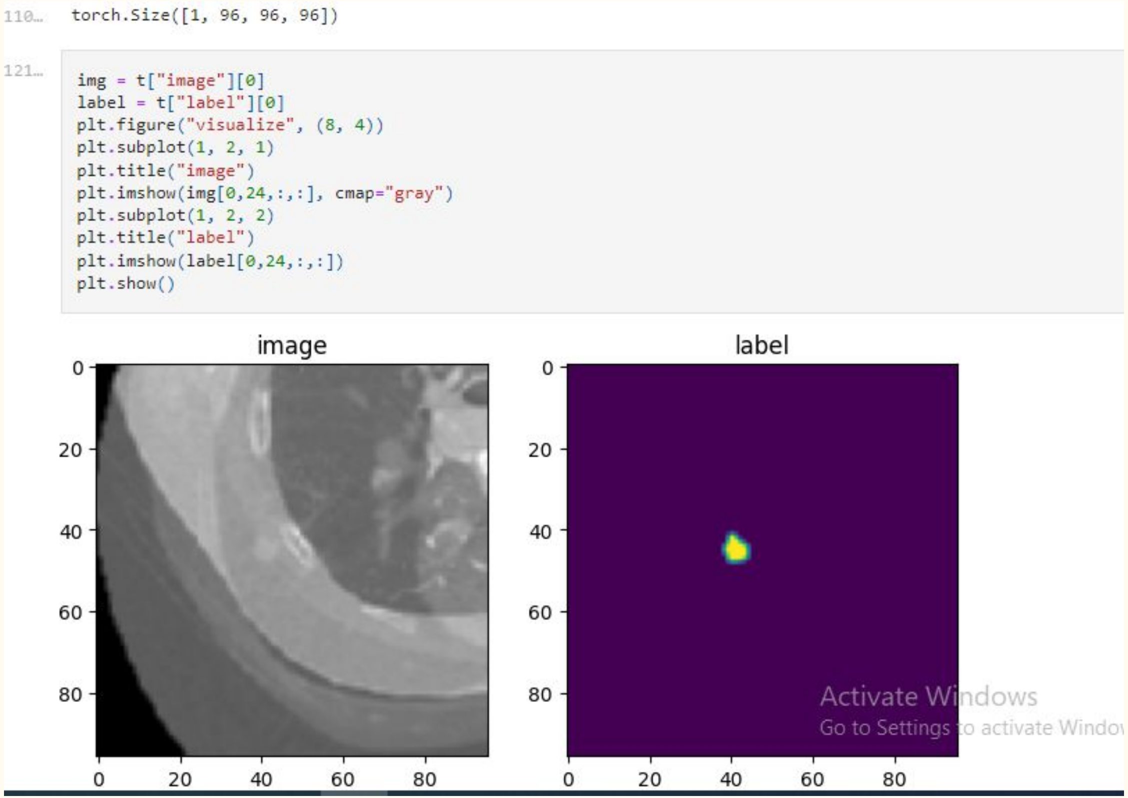
```
train_dict_list
```

```
[{'image': '/kaggle/working/NII Training_Images/train_image0.nii.gz',
  'label': '/kaggle/working/NII Training_Images/train_label0.nii.gz'},
 {'image': '/kaggle/working/NII Training_Images/train_image1.nii.gz',
  'label': '/kaggle/working/NII Training_Images/train_label1.nii.gz'},
 {'image': '/kaggle/working/NII Training_Images/train_image2.nii.gz',
  'label': '/kaggle/working/NII Training_Images/train_label2.nii.gz'},
 {'image': '/kaggle/working/NII Training_Images/train_image3.nii.gz',
  'label': '/kaggle/working/NII Training_Images/train_label3.nii.gz'},
 {'image': '/kaggle/working/NII Training_Images/train_image4.nii.gz',
  'label': '/kaggle/working/NII Training_Images/train_label4.nii.gz'},
 {'image': '/kaggle/working/NII Training_Images/train_image5.nii.gz',
  'label': '/kaggle/working/NII Training_Images/train_label5.nii.gz'},
```

# Transforming and Partitioning Voxels

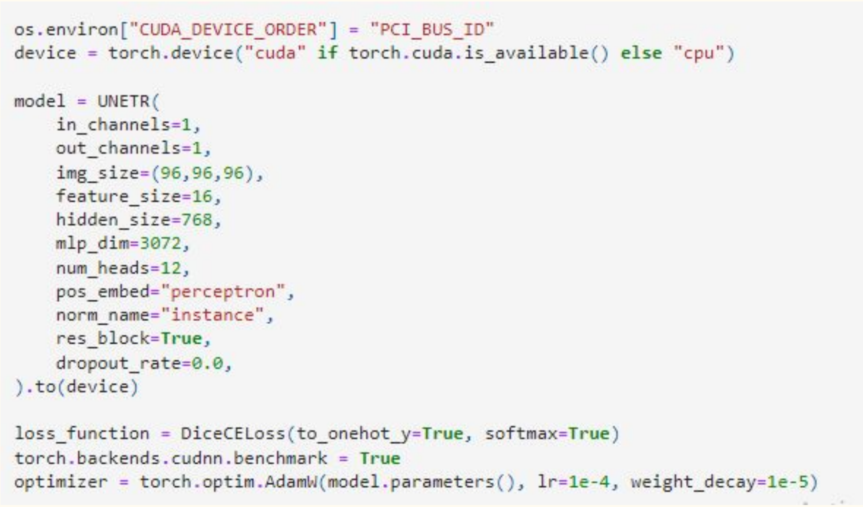


```
train_transforms = Compose(
    [
        LoadImaged(keys=["image", "label"]),
        EnsureChannelFirstd(keys=["image", "label"]),
        Orientationd(keys=["image", "label"], axcodes="RAS"),
        Spacingd(
            keys=["image", "label"],
            pixdim=(1.5, 1.5, 2.0),
            mode=("bilinear", "nearest"),
        ),
        ScaleIntensityRanged(
            keys=["image"],
            a_min=-175,
            a_max=250,
            b_min=0.0,
            b_max=1.0,
            clip=True,
        ),
          Resized(
            keys = ["image", "label"],
            spatial_size = (200,200,200)
          ),
        CropForegroundd(keys=["image", "label"], source_key="image"),
        RandCropByPosNegLabeld(
            keys=["image", "label"],
            label_key="label",
            spatial_size=(96,96,96),
            pos=1,
            neg=1,
            num_samples=4,
            image_key="image",
            image_threshold=0,
        ),
```

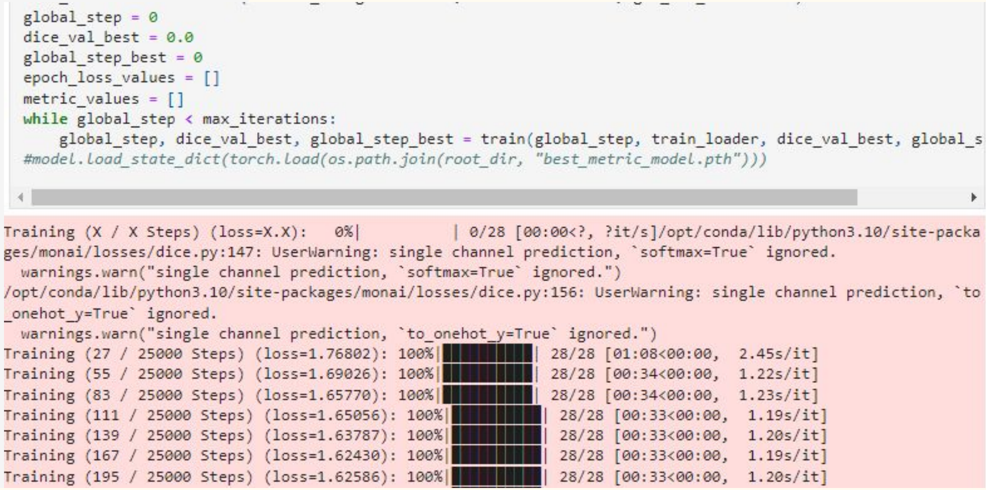


```
110…  torch.Size([1, 96, 96, 96])

121…  img = t["image"][0]
      label = t["label"][0]
      plt.figure("visualize", (8, 4))
      plt.subplot(1, 2, 1)
      plt.title("image")
      plt.imshow(img[0,24,:,:], cmap="gray")
      plt.subplot(1, 2, 2)
      plt.title("label")
      plt.imshow(label[0,24,:,:])
      plt.show()
```

# UNET-R Model



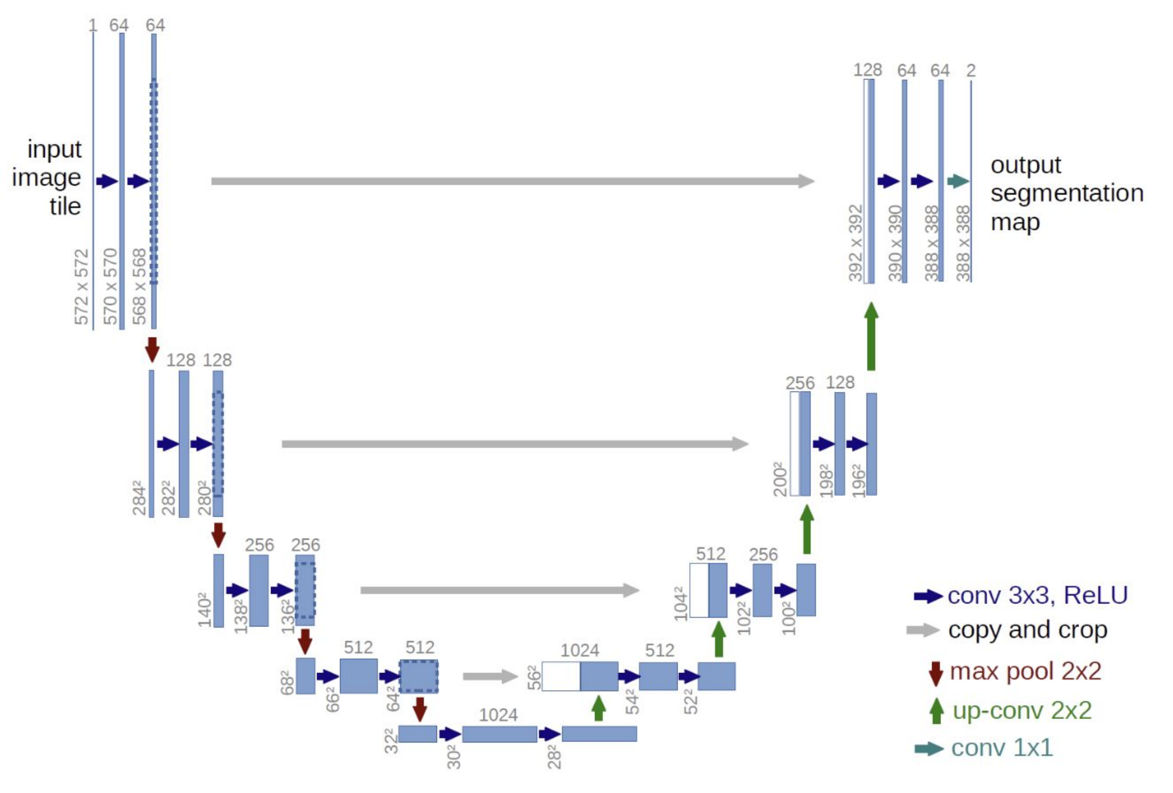
```
os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model = UNETR(
    in_channels=1,
    out_channels=1,
    img_size=(96,96,96),
    feature_size=16,
    hidden_size=768,
    mlp_dim=3072,
    num_heads=12,
    pos_embed="perceptron",
    norm_name="instance",
    res_block=True,
    dropout_rate=0.0,
).to(device)

loss_function = DiceCELoss(to_onehot_y=True, softmax=True)
torch.backends.cudnn.benchmark = True
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-4, weight_decay=1e-5)
```



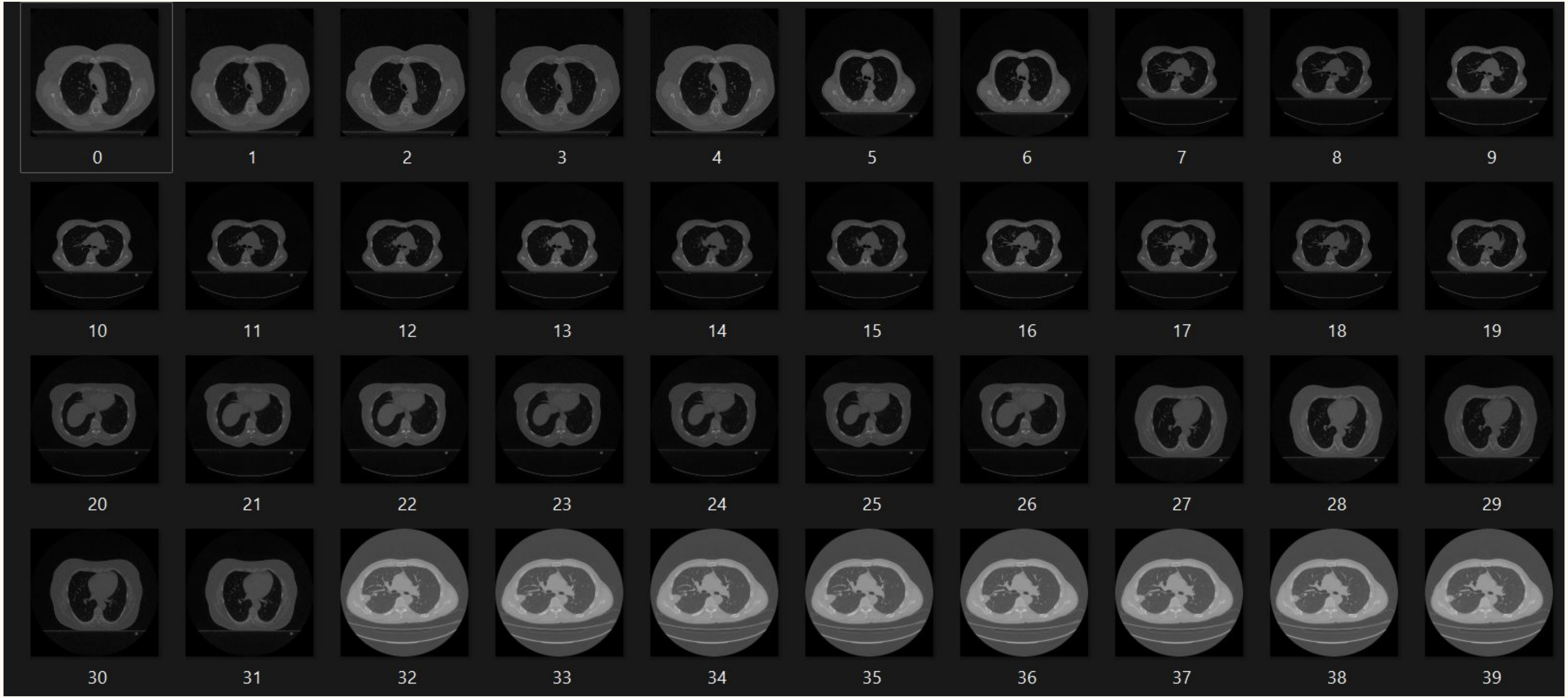
```
global_step = 0
dice_val_best = 0.0
global_step_best = 0
epoch_loss_values = []
metric_values = []
while global_step < max_iterations:
    global_step, dice_val_best, global_step_best = train(global_step, train_loader, dice_val_best, global_s
#model.load_state_dict(torch.load(os.path.join(root_dir, "best_metric_model.pth")))
```

```
Training (X / X Steps) (loss=X.X):   0%|          | 0/28 [00:00<?, ?it/s]/opt/conda/lib/python3.10/site-packa
ges/monai/losses/dice.py:147: UserWarning: single channel prediction, `softmax=True` ignored.
  warnings.warn("single channel prediction, `softmax=True` ignored.")
/opt/conda/lib/python3.10/site-packages/monai/losses/dice.py:156: UserWarning: single channel prediction, `to
_onehot_y=True` ignored.
  warnings.warn("single channel prediction, `to_onehot_y=True` ignored.")
Training (27 / 25000 Steps) (loss=1.76802): 100%|██████████| 28/28 [01:08<00:00,  2.45s/it]
Training (55 / 25000 Steps) (loss=1.69026): 100%|██████████| 28/28 [00:34<00:00,  1.22s/it]
Training (83 / 25000 Steps) (loss=1.65770): 100%|██████████| 28/28 [00:34<00:00,  1.23s/it]
Training (111 / 25000 Steps) (loss=1.65056): 100%|██████████| 28/28 [00:33<00:00,  1.19s/it]
Training (139 / 25000 Steps) (loss=1.63787): 100%|██████████| 28/28 [00:33<00:00,  1.20s/it]
Training (167 / 25000 Steps) (loss=1.62430): 100%|██████████| 28/28 [00:33<00:00,  1.19s/it]
Training (195 / 25000 Steps) (loss=1.62586): 100%|██████████| 28/28 [00:33<00:00,  1.20s/it]
```

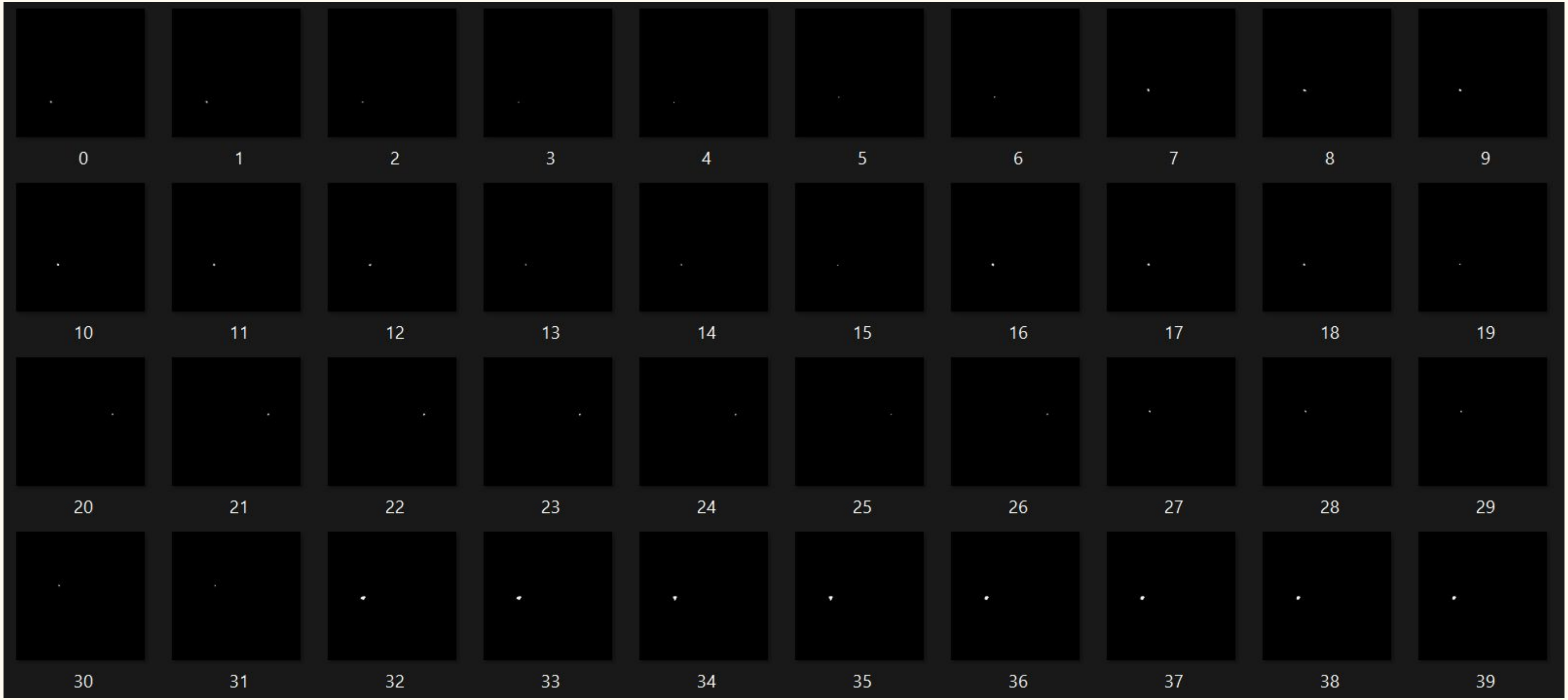
# UNET Model for 2D Convolutional Network



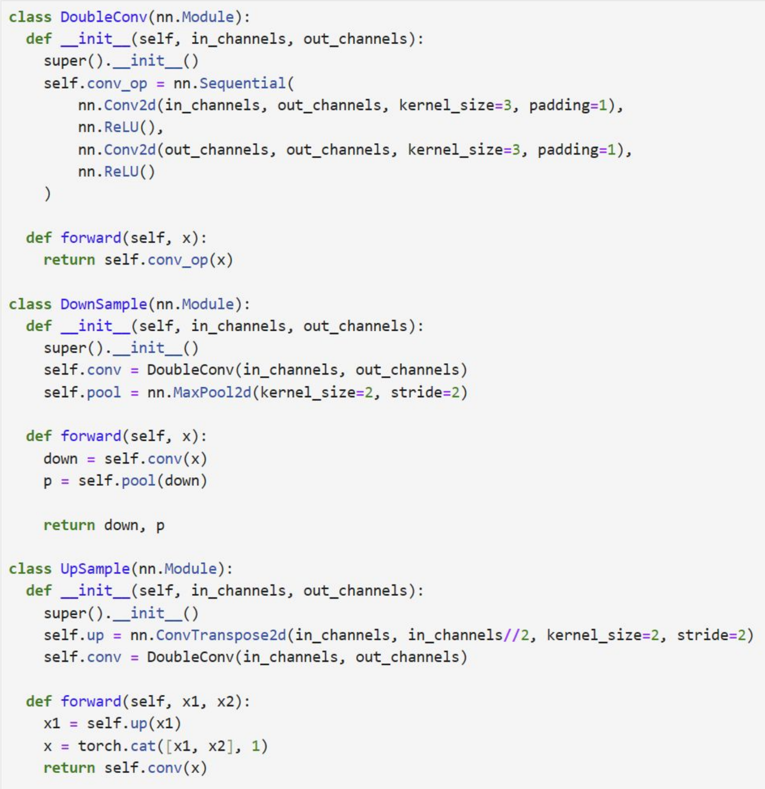
# Input Images



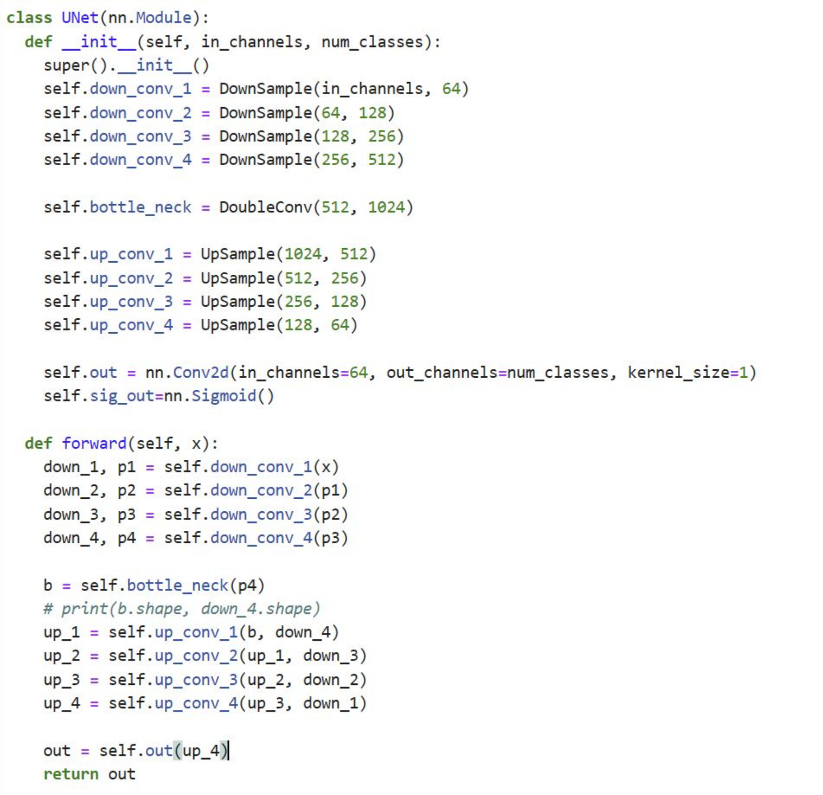
# Output Image



# UNET Model 1:

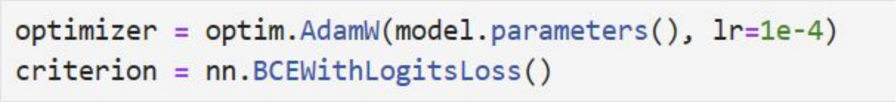


```
class DoubleConv(nn.Module):
  def __init__(self, in_channels, out_channels):
    super().__init__()
    self.conv_op = nn.Sequential(
        nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1),
        nn.ReLU()
    )

  def forward(self, x):
    return self.conv_op(x)

class DownSample(nn.Module):
  def __init__(self, in_channels, out_channels):
    super().__init__()
    self.conv = DoubleConv(in_channels, out_channels)
    self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

  def forward(self, x):
    down = self.conv(x)
    p = self.pool(down)

    return down, p

class UpSample(nn.Module):
  def __init__(self, in_channels, out_channels):
    super().__init__()
    self.up = nn.ConvTranspose2d(in_channels, in_channels//2, kernel_size=2, stride=2)
    self.conv = DoubleConv(in_channels, out_channels)

  def forward(self, x1, x2):
    x1 = self.up(x1)
    x = torch.cat([x1, x2], 1)
    return self.conv(x)
```

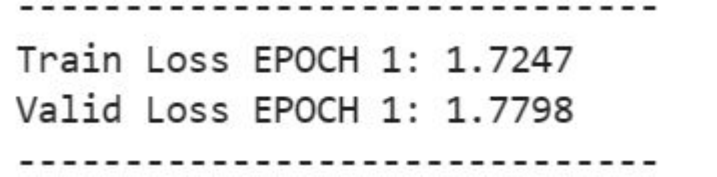


```
class UNet(nn.Module):
  def __init__(self, in_channels, num_classes):
    super().__init__()
    self.down_conv_1 = DownSample(in_channels, 64)
    self.down_conv_2 = DownSample(64, 128)
    self.down_conv_3 = DownSample(128, 256)
    self.down_conv_4 = DownSample(256, 512)

    self.bottle_neck = DoubleConv(512, 1024)

    self.up_conv_1 = UpSample(1024, 512)
    self.up_conv_2 = UpSample(512, 256)
    self.up_conv_3 = UpSample(256, 128)
    self.up_conv_4 = UpSample(128, 64)

    self.out = nn.Conv2d(in_channels=64, out_channels=num_classes, kernel_size=1)
    self.sig_out=nn.Sigmoid()

  def forward(self, x):
    down_1, p1 = self.down_conv_1(x)
    down_2, p2 = self.down_conv_2(p1)
    down_3, p3 = self.down_conv_3(p2)
    down_4, p4 = self.down_conv_4(p3)

    b = self.bottle_neck(p4)
    # print(b.shape, down_4.shape)
    up_1 = self.up_conv_1(b, down_4)
    up_2 = self.up_conv_2(up_1, down_3)
    up_3 = self.up_conv_3(up_2, down_2)
    up_4 = self.up_conv_4(up_3, down_1)

    out = self.out(up_4)
    return out
```

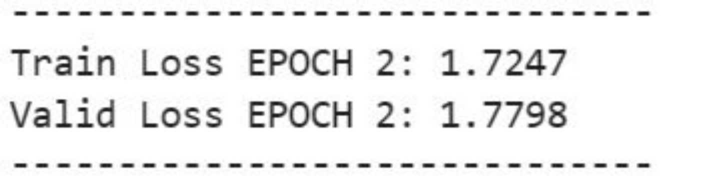
# UNET MODEL 1: Result



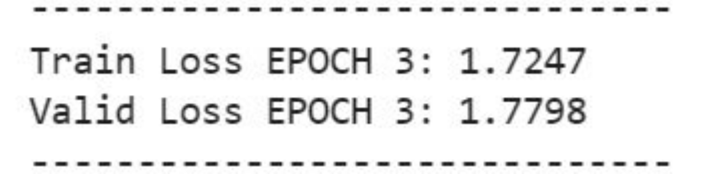
```
optimizer = optim.AdamW(model.parameters(), lr=1e-4)
criterion = nn.BCEWithLogitsLoss()
```



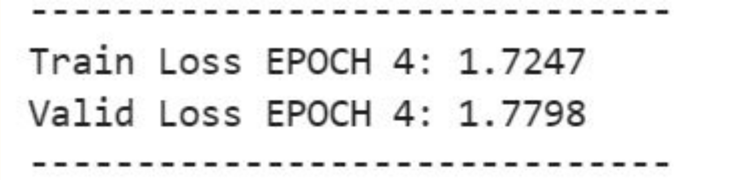
```
--------------------------------
Train Loss EPOCH 1: 1.7247
Valid Loss EPOCH 1: 1.7798
--------------------------------
```



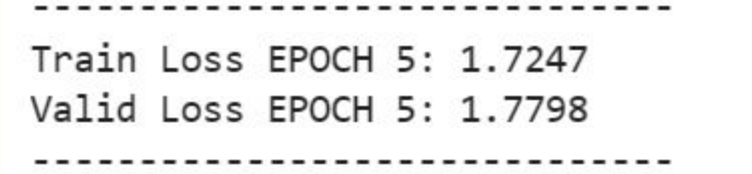
```
--------------------------------
Train Loss EPOCH 2: 1.7247
Valid Loss EPOCH 2: 1.7798
--------------------------------
```



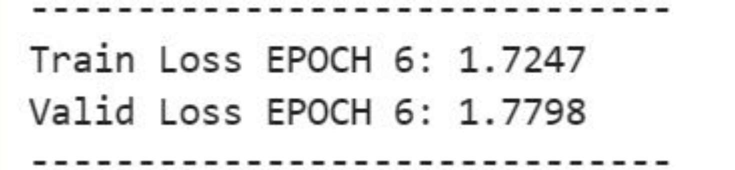
```
--------------------------------
Train Loss EPOCH 3: 1.7247
Valid Loss EPOCH 3: 1.7798
--------------------------------
```



```
--------------------------------
Train Loss EPOCH 4: 1.7247
Valid Loss EPOCH 4: 1.7798
--------------------------------
```

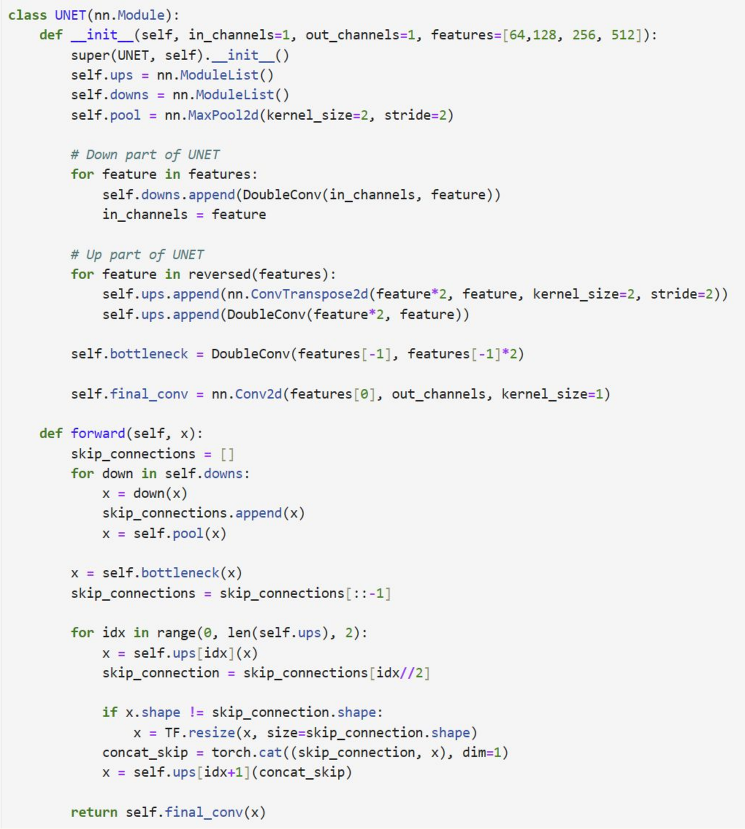


```
--------------------------------
Train Loss EPOCH 5: 1.7247
Valid Loss EPOCH 5: 1.7798
--------------------------------
```

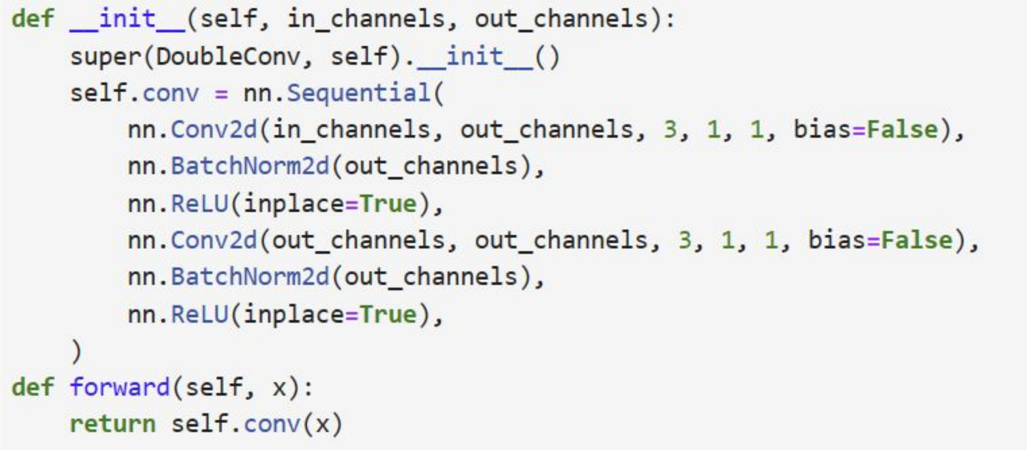


```
--------------------------------
Train Loss EPOCH 6: 1.7247
Valid Loss EPOCH 6: 1.7798
--------------------------------
```

# UNET MODEL 2:

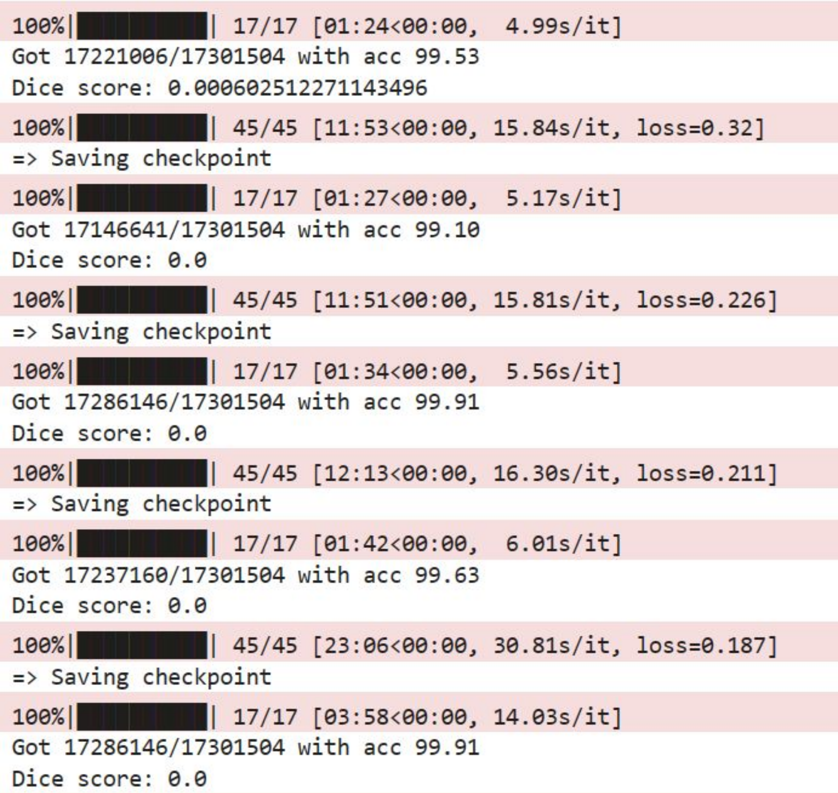


```
class UNET(nn.Module):
    def __init__(self, in_channels=1, out_channels=1, features=[64,128, 256, 512]):
        super(UNET, self).__init__()
        self.ups = nn.ModuleList()
        self.downs = nn.ModuleList()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        # Down part of UNET
        for feature in features:
            self.downs.append(DoubleConv(in_channels, feature))
            in_channels = feature

        # Up part of UNET
        for feature in reversed(features):
            self.ups.append(nn.ConvTranspose2d(feature*2, feature, kernel_size=2, stride=2))
            self.ups.append(DoubleConv(feature*2, feature))

        self.bottleneck = DoubleConv(features[-1], features[-1]*2)

        self.final_conv = nn.Conv2d(features[0], out_channels, kernel_size=1)

    def forward(self, x):
        skip_connections = []
        for down in self.downs:
            x = down(x)
            skip_connections.append(x)
            x = self.pool(x)

        x = self.bottleneck(x)
        skip_connections = skip_connections[::-1]

        for idx in range(0, len(self.ups), 2):
            x = self.ups[idx](x)
            skip_connection = skip_connections[idx//2]

            if x.shape != skip_connection.shape:
                x = TF.resize(x, size=skip_connection.shape)
            concat_skip = torch.cat((skip_connection, x), dim=1)
            x = self.ups[idx+1](concat_skip)

        return self.final_conv(x)
```



```
def __init__(self, in_channels, out_channels):
    super(DoubleConv, self).__init__()
    self.conv = nn.Sequential(
        nn.Conv2d(in_channels, out_channels, 3, 1, 1, bias=False),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(inplace=True),
        nn.Conv2d(out_channels, out_channels, 3, 1, 1, bias=False),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(inplace=True),
    )
def forward(self, x):
    return self.conv(x)
```

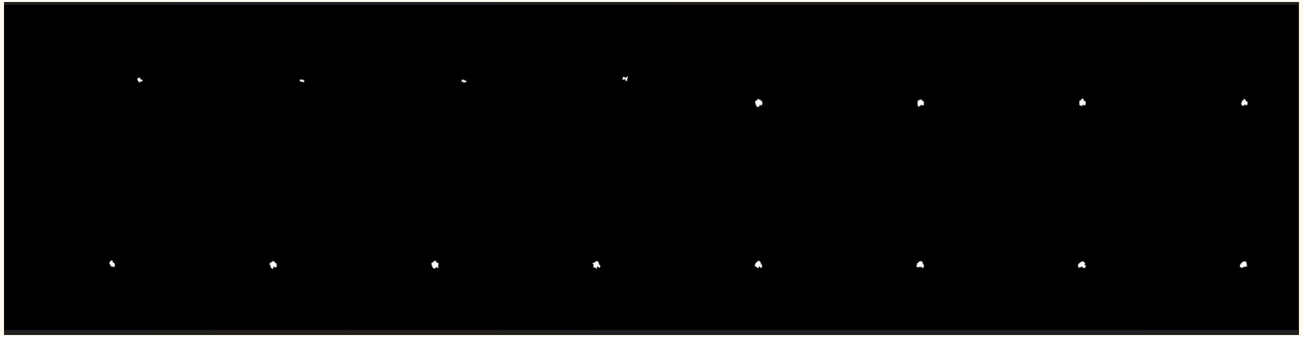
# UNET Model 2: training results



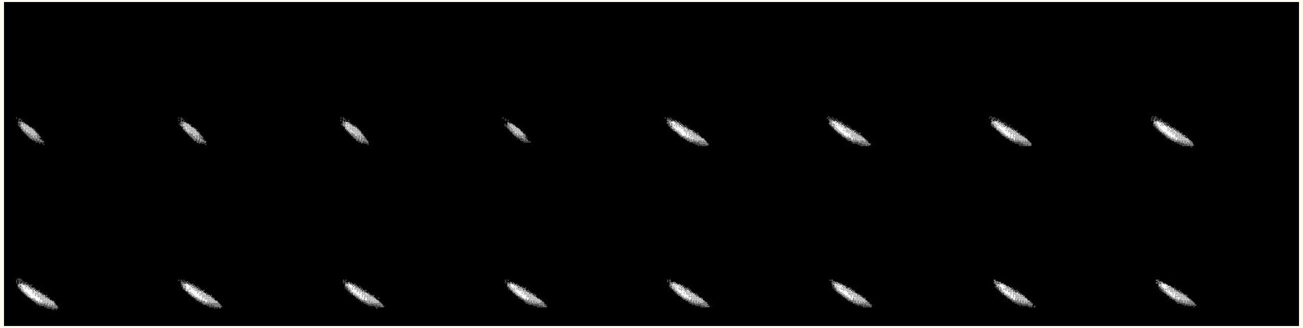
```
100%|██████████| 17/17 [01:24<00:00,  4.99s/it]
Got 17221006/17301504 with acc 99.53
Dice score: 0.000602512271143496
100%|██████████| 45/45 [11:53<00:00, 15.84s/it, loss=0.32]
=> Saving checkpoint
100%|██████████| 17/17 [01:27<00:00,  5.17s/it]
Got 17146641/17301504 with acc 99.10
Dice score: 0.0
100%|██████████| 45/45 [11:51<00:00, 15.81s/it, loss=0.226]
=> Saving checkpoint
100%|██████████| 17/17 [01:34<00:00,  5.56s/it]
Got 17286146/17301504 with acc 99.91
Dice score: 0.0
100%|██████████| 45/45 [12:13<00:00, 16.30s/it, loss=0.211]
=> Saving checkpoint
100%|██████████| 17/17 [01:42<00:00,  6.01s/it]
Got 17237160/17301504 with acc 99.63
Dice score: 0.0
100%|██████████| 45/45 [23:06<00:00, 30.81s/it, loss=0.187]
=> Saving checkpoint
100%|██████████| 17/17 [03:58<00:00, 14.03s/it]
Got 17286146/17301504 with acc 99.91
Dice score: 0.0
```

# UNET Model 2: sample predictions

Ground Truth:



Prediction:



# Future Plans

- Continuing developing current models
- Create a multi channel 2D convolutional network
- Work on an classifier for semantic segmentation
- Explore and test models

# Tentative Development Schedule

| | |
|---|---|
| **Apr 11-18** | Dataset sourcing and literature review |
| **Apr 18-25** | Finetune a pretrained model with this dataset |
| **Apr 25-May 2** | Develop baseline models |
| **May 2-9** | Finish testing baseline models with results |
| **May 9-16** | Presentation & Paper |