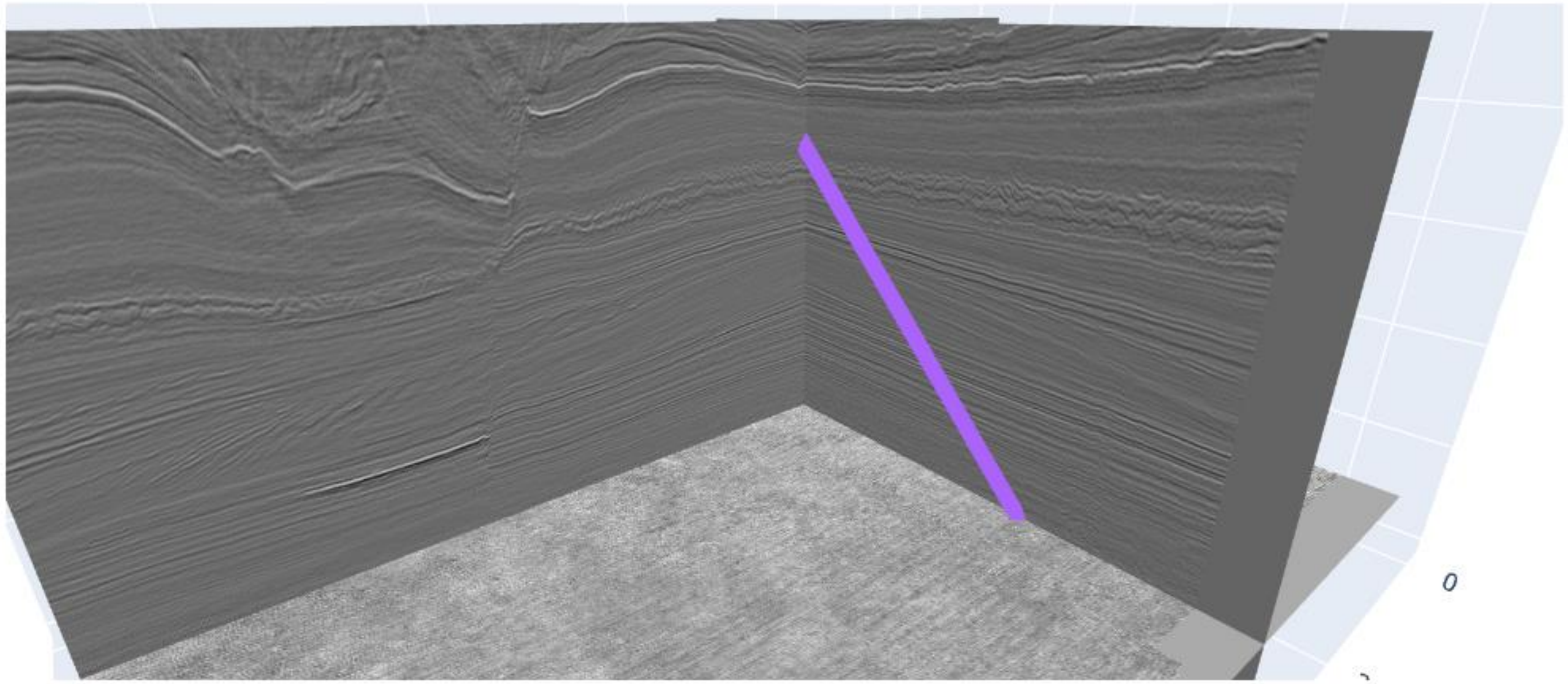
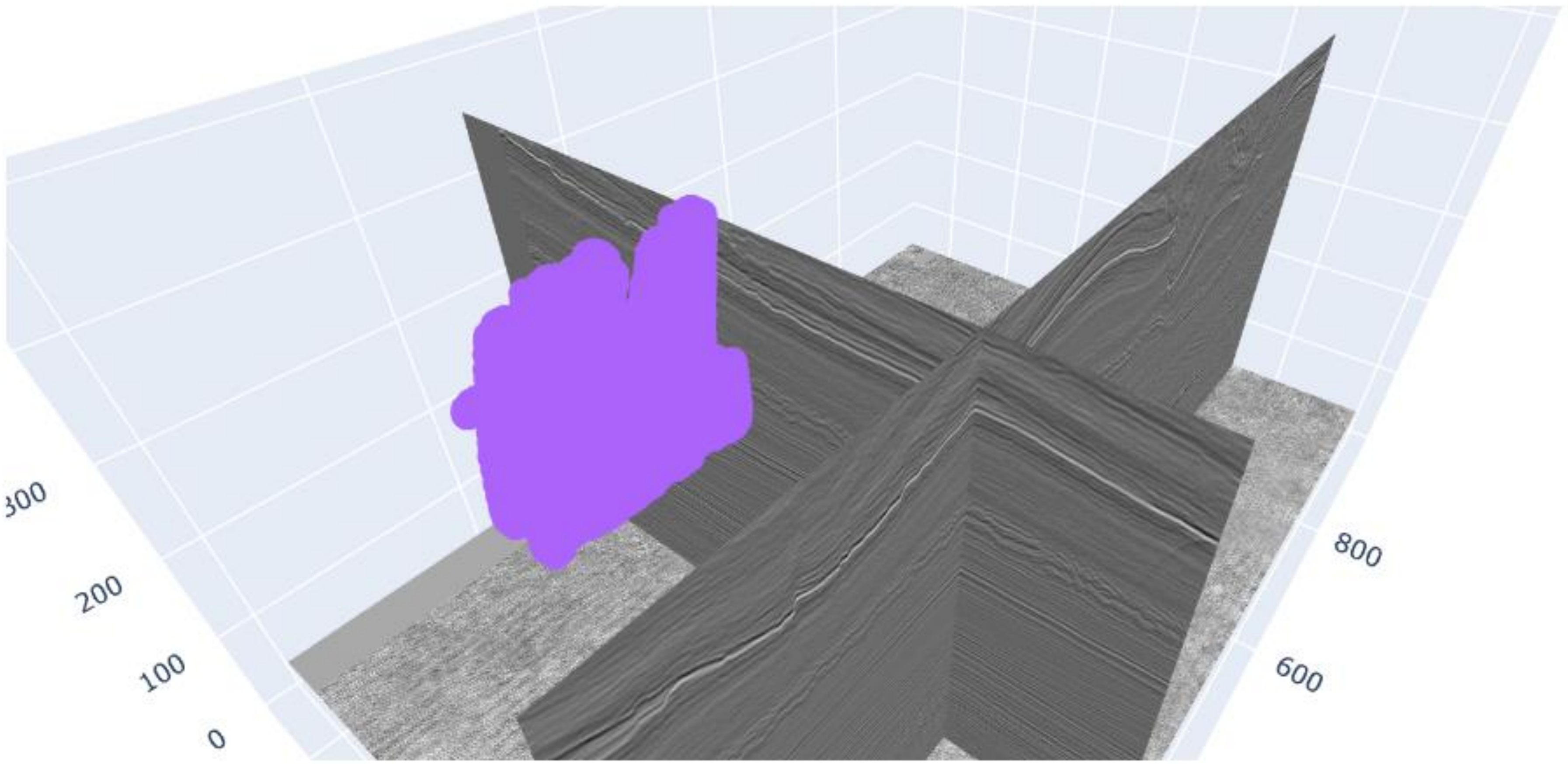
```
segy = segyio.open('data/Dutch Government_F3_entire_8bit seismic.segy')
data = segy.trace.raw[:]
cube = np.dstack(tuple(data[len(segy.xlines) * i : len(segy.xlines)*(i+1)] for i in range(data.shape[0]//len(segy.xlines))))
clip_percentile = 99
vm = np.percentile(data, clip_percentile)
cube = np.moveaxis(cube, [0, 2], [2, 0])
```

```
plot_3d_scale(cube, 300, 150, 50, [x,y,z])
```



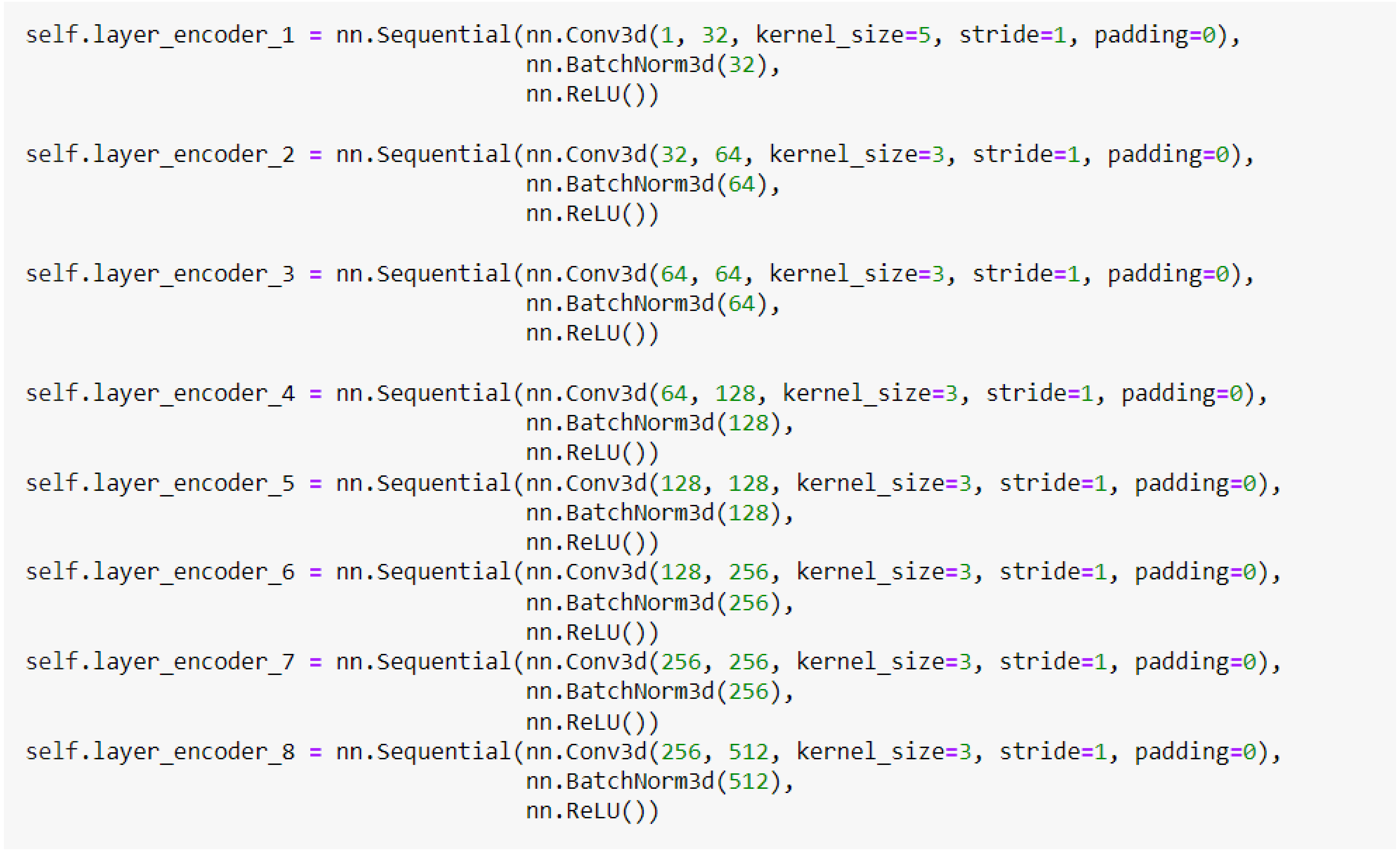

300
200
100
0
800
600

```python
class SegYDataset(Dataset):
    def __init__(self, data_type=None, size_scale=1):
        self.fault_data = np.load('reduced_fault_data.npy')
        self.seis_data = np.load('reduced_seis_data.npy')
        self.size_scale = size_scale

    def __len__(self):
        return 100

    def __getitem__(self, idx):
        while True:
            size = 128
            IL, XL, Z = self.seis_data.shape
            iline = random.randint(0, IL - size)
            xline = random.randint(0, XL - size)
            zline = random.randint(0, Z - size)
            seis_slice = self.seis_data[iline: iline+size,
                                        xline: xline+size,
                                        zline: zline+size,]

            fault_slice = self.fault_data[iline: iline+size,
                                          xline: xline+size,
                                          zline: zline+size,]
            if fault_slice.sum() > 70_000:
                X = torch.Tensor(seis_slice)
                Y = torch.Tensor(fault_slice)
                return (X[None,:], Y[None, :])
```

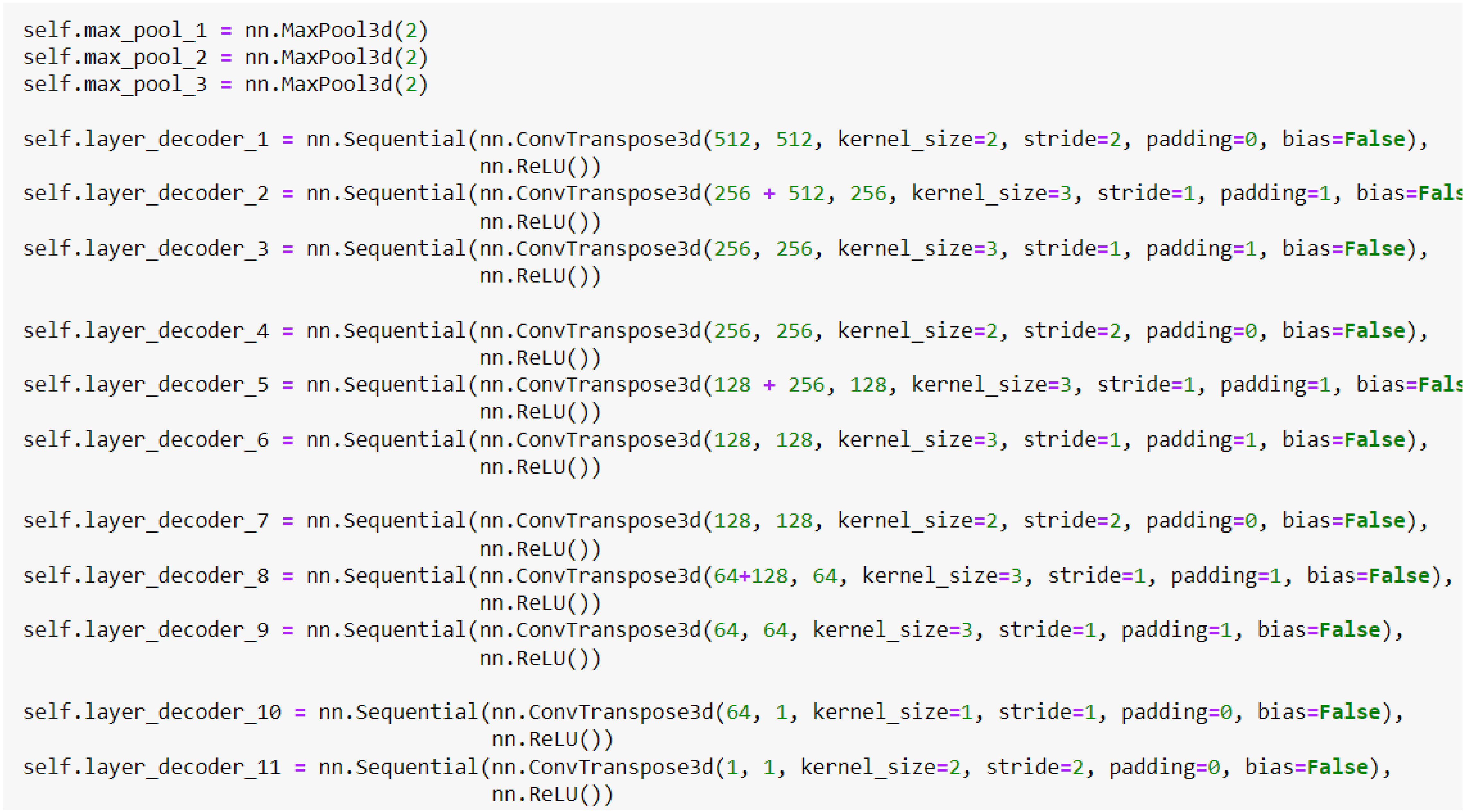


```
self.layer_encoder_1 = nn.Sequential(nn.Conv3d(1, 32, kernel_size=5, stride=1, padding=0),
                                     nn.BatchNorm3d(32),
                                     nn.ReLU())

self.layer_encoder_2 = nn.Sequential(nn.Conv3d(32, 64, kernel_size=3, stride=1, padding=0),
                                     nn.BatchNorm3d(64),
                                     nn.ReLU())

self.layer_encoder_3 = nn.Sequential(nn.Conv3d(64, 64, kernel_size=3, stride=1, padding=0),
                                     nn.BatchNorm3d(64),
                                     nn.ReLU())

self.layer_encoder_4 = nn.Sequential(nn.Conv3d(64, 128, kernel_size=3, stride=1, padding=0),
                                     nn.BatchNorm3d(128),
                                     nn.ReLU())
self.layer_encoder_5 = nn.Sequential(nn.Conv3d(128, 128, kernel_size=3, stride=1, padding=0),
                                     nn.BatchNorm3d(128),
                                     nn.ReLU())
self.layer_encoder_6 = nn.Sequential(nn.Conv3d(128, 256, kernel_size=3, stride=1, padding=0),
                                     nn.BatchNorm3d(256),
                                     nn.ReLU())
self.layer_encoder_7 = nn.Sequential(nn.Conv3d(256, 256, kernel_size=3, stride=1, padding=0),
                                     nn.BatchNorm3d(256),
                                     nn.ReLU())
self.layer_encoder_8 = nn.Sequential(nn.Conv3d(256, 512, kernel_size=3, stride=1, padding=0),
                                     nn.BatchNorm3d(512),
                                     nn.ReLU())
```



```
self.max_pool_1 = nn.MaxPool3d(2)
self.max_pool_2 = nn.MaxPool3d(2)
self.max_pool_3 = nn.MaxPool3d(2)

self.layer_decoder_1 = nn.Sequential(nn.ConvTranspose3d(512, 512, kernel_size=2, stride=2, padding=0, bias=False),
                                     nn.ReLU())
self.layer_decoder_2 = nn.Sequential(nn.ConvTranspose3d(256 + 512, 256, kernel_size=3, stride=1, padding=1, bias=Fals
                                     nn.ReLU())
self.layer_decoder_3 = nn.Sequential(nn.ConvTranspose3d(256, 256, kernel_size=3, stride=1, padding=1, bias=False),
                                     nn.ReLU())

self.layer_decoder_4 = nn.Sequential(nn.ConvTranspose3d(256, 256, kernel_size=2, stride=2, padding=0, bias=False),
                                     nn.ReLU())
self.layer_decoder_5 = nn.Sequential(nn.ConvTranspose3d(128 + 256, 128, kernel_size=3, stride=1, padding=1, bias=Fals
                                     nn.ReLU())
self.layer_decoder_6 = nn.Sequential(nn.ConvTranspose3d(128, 128, kernel_size=3, stride=1, padding=1, bias=False),
                                     nn.ReLU())

self.layer_decoder_7 = nn.Sequential(nn.ConvTranspose3d(128, 128, kernel_size=2, stride=2, padding=0, bias=False),
                                     nn.ReLU())
self.layer_decoder_8 = nn.Sequential(nn.ConvTranspose3d(64+128, 64, kernel_size=3, stride=1, padding=1, bias=False),
                                     nn.ReLU())
self.layer_decoder_9 = nn.Sequential(nn.ConvTranspose3d(64, 64, kernel_size=3, stride=1, padding=1, bias=False),
                                     nn.ReLU())

self.layer_decoder_10 = nn.Sequential(nn.ConvTranspose3d(64, 1, kernel_size=1, stride=1, padding=0, bias=False),
                                      nn.ReLU())
self.layer_decoder_11 = nn.Sequential(nn.ConvTranspose3d(1, 1, kernel_size=2, stride=2, padding=0, bias=False),
                                      nn.ReLU())
```