Lampiran

1.1 Lampiran Source Code

1. Requirements

```
python == 3.10.12
tensorflow == 2.15.0
itk == 5.3.0
itkwidgets == 0.32.6
imageio == 2.31.6
keras == 2.15.0
ipywidgets == 7.7.1
IPython == 7.34.0
Numpy == 1.23.5
Pandas == 1.5.3
nibabel == 4.0.2
matplotlib ==3.7.1
mayavi == 4.8.1
```

2. Arsitektur RSU U^2 Net+

```
from tensorflow.keras.layers import Input, Conv3D, BatchNormalization, Activation, MaxPooling3D, UpSampling3D, Concatenate, Add

tf.keras.backend.set_image_data_format('channels_first')


def conv_block(inputs, out_ch, rate=1):
    x = Conv3D(out_ch, 3, padding="same", dilation_rate=1)(inputs)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    return x


def RSU_L(inputs, out_ch, int_ch, num_layers, rate=2):
    x = conv_block(inputs, out_ch)
    init_feats = x


    skip = []
    x = conv_block(x, int_ch)
```

```
    skip.append(x)
    for i in range(num_layers - 2):
        x = MaxPooling3D((2, 2, 2))(x)
        x = conv_block(x, int_ch)
        skip.append(x)

    x = conv_block(x, int_ch, rate=rate)

    skip.reverse()

    x = Concatenate(axis=1)([x, skip[0]])
    x = conv_block(x, int_ch)

    for i in range(num_layers - 3):
        x = UpSampling3D(size=(2, 2, 2))(x)
        x_concat = Concatenate(axis=1)([skip[i + 1], UpSampling3D(size=(2,
2, 2))(skip[i])])
        x = Concatenate(axis=1)([x, x_concat])
        x = conv_block(x, int_ch)

    x = UpSampling3D(size=(2, 2, 2))(x)
    x = Concatenate(axis=1)([x, skip[-1]])
    x = conv_block(x, out_ch)
    x = Add()([x, init_feats])
    return x
def RSU_4F(inputs, out_ch, int_ch):
    x0 = conv_block(inputs, out_ch, rate=1)

    x1 = conv_block(x0, int_ch, rate=1)
    x2 = conv_block(x1, int_ch, rate=2)
```

```
    x3 = conv_block(x2, int_ch, rate=4)

    x4 = conv_block(x3, int_ch, rate=8)

    x = Concatenate(axis=1)([x4, x3])
    x = conv_block(x, int_ch, rate=4)

    x = Concatenate(axis=1)([x, x2])
    x = conv_block(x, int_ch, rate=2)

    x = Concatenate(axis=1)([x, x1])
    x = conv_block(x, out_ch, rate=1)

    x = Add()([x, x0])
    return x

def u2net(input_shape, out_ch, int_ch, num_classes=3):
    inputs = Input(input_shape)
    s0 = inputs

    s1 = RSU_L(s0, out_ch[0], int_ch[0], 6)
    p1 = MaxPooling3D((2, 2, 2))(s1)
    s2 = RSU_L(p1, out_ch[1], int_ch[1], 5)
    p2 = MaxPooling3D((2, 2, 2))(s2)
    s3 = RSU_L(p2, out_ch[2], int_ch[2], 4)
    p3 = MaxPooling3D((2, 2, 2))(s3)

    s4 = RSU_4F(p3, out_ch[3], int_ch[3])
    p4 = MaxPooling3D((2, 2, 2))(s4)
```

```
b1 = RSU_4F(p4, out_ch[4], int_ch[4])
b2 = UpSampling3D(size=(2, 2, 2))(b1)
d1 = Concatenate(axis=1)([b2, s4])
d1 = RSU_4F(d1, out_ch[5], int_ch[5])
u1 = UpSampling3D(size=(2, 2, 2))(d1)

d2 = Concatenate(axis=1)([u1, s3])
d2 = RSU_L(d2, out_ch[6], int_ch[6], 4)
u2 = UpSampling3D(size=(2, 2, 2))(d2)

d3 = Concatenate(axis=1)([u2, s2])
d3 = RSU_L(d3, out_ch[7], int_ch[7], 5)
u3 = UpSampling3D(size=(2, 2, 2))(d3)

d4 = Concatenate(axis=1)([u3, s1])
d4 = RSU_L(d4, out_ch[8], int_ch[8], 6)

y1 = Conv3D(num_classes, 3, padding="same")(d4)
y2 = Conv3D(num_classes, 3, padding="same")(d3)
y2 = UpSampling3D(size=(2, 2, 2))(y2)
y3 = Conv3D(num_classes, 3, padding="same")(d2)
y3 = UpSampling3D(size=(4, 4, 4))(y3)
y4 = Conv3D(num_classes, 3, padding="same")(d1)
y4 = UpSampling3D(size=(8, 8, 8))(y4)
y5 = Conv3D(num_classes, 3, padding="same")(b1)
y5 = UpSampling3D(size=(16, 16, 16))(y5)
y6 = Concatenate(axis=1)([y1, y2, y3, y4, y5])
y6 = Conv3D(num_classes, 1, padding="same")(y6)
```

```
    y6 = Activation("sigmoid")(y6)
    model = tf.keras.models.Model(inputs, outputs=y6)
    return model


def build_u2net(input_shape, num_classes=3):
    out_ch = [32, 64, 128, 256, 512, 256, 128, 64, 32]
    int_ch = [32, 32, 64, 128, 256, 128, 64, 32, 32]
    model = u2net(input_shape, out_ch, int_ch, num_classes=num_classes)
    return model

model = build_u2net((4, 160, 160, 16))
model.summary()
```