import tensorflow as tf

def attention\_block(x, g, inter\_channel):

theta\_x = tf.layers.conv2d(x, inter\_channel, [1, 1], padding='SAME')

phi\_g = tf.layers.conv2d(g, inter\_channel, [1, 1], padding='SAME')

f = tf.nn.relu(tf.add(theta\_x, phi\_g))

psi\_f = tf.layers.conv2d(f, 1, [1, 1], padding='SAME')

sigmoid\_psi\_f = tf.nn.sigmoid(psi\_f)

att\_x = tf.multiply(x, sigmoid\_psi\_f)

return att\_x

def down\_block(x, filters, kernel\_size=(3, 3), padding="same", strides=1):

c = tf.layers.conv2d(x, filters, kernel\_size, padding=padding, strides=strides)

c = tf.layers.batch\_normalization(c)

c = tf.nn.relu(c)

p = tf.layers.max\_pooling2d(c, pool\_size=(2, 2), strides=(2, 2))

return c, p

def up\_block(x, skip, filters, kernel\_size=(3, 3), padding="same", strides=1):

us = tf.layers.conv2d\_transpose(x, filters, kernel\_size, padding=padding, strides=strides)

concat = tf.concat([us, skip], axis=3)

c = tf.layers.conv2d(concat, filters, kernel\_size, padding=padding, strides=1)

c = tf.layers.batch\_normalization(c)

c = tf.nn.relu(c)

return c

def attention\_unet(input\_shape=(256, 256, 3), num\_classes=1):

inputs = tf.keras.layers.Input(input\_shape)

c1, p1 = down\_block(inputs, 64)

c2, p2 = down\_block(p1, 128)

c3, p3 = down\_block(p2, 256)

c4, p4 = down\_block(p3, 512)

x = down\_block(p4, 1024)

x = up\_block(x, c4, 512)

x = up\_block(x, c3, 256)

x = up\_block(x, c2, 128)

x = up\_block(x, c1, 64)

attention = attention\_block(c4, x, 256)

outputs = tf.layers.conv2d(attention, num\_classes, [1, 1], padding='SAME', activation='sigmoid')

model = tf.keras.Model(inputs, outputs, name='Attention\_U\_Net')

return model