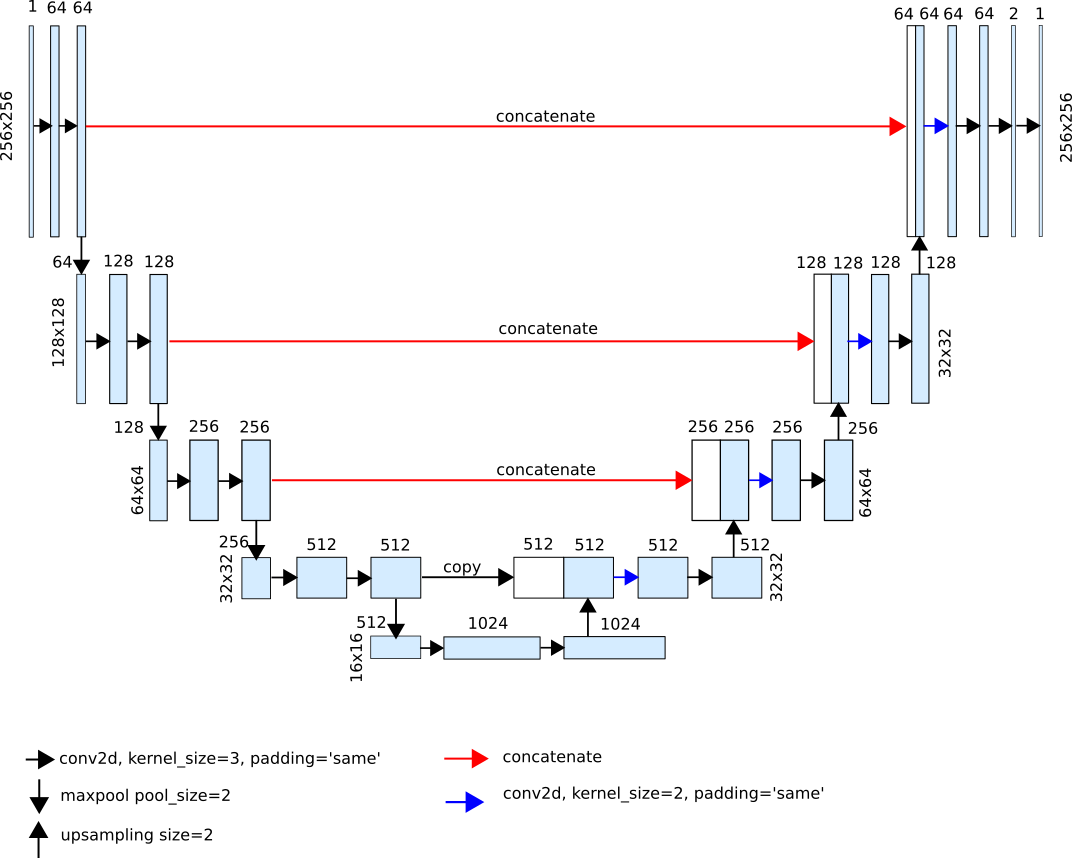
LAB 10: Segmentation using deep learning models.

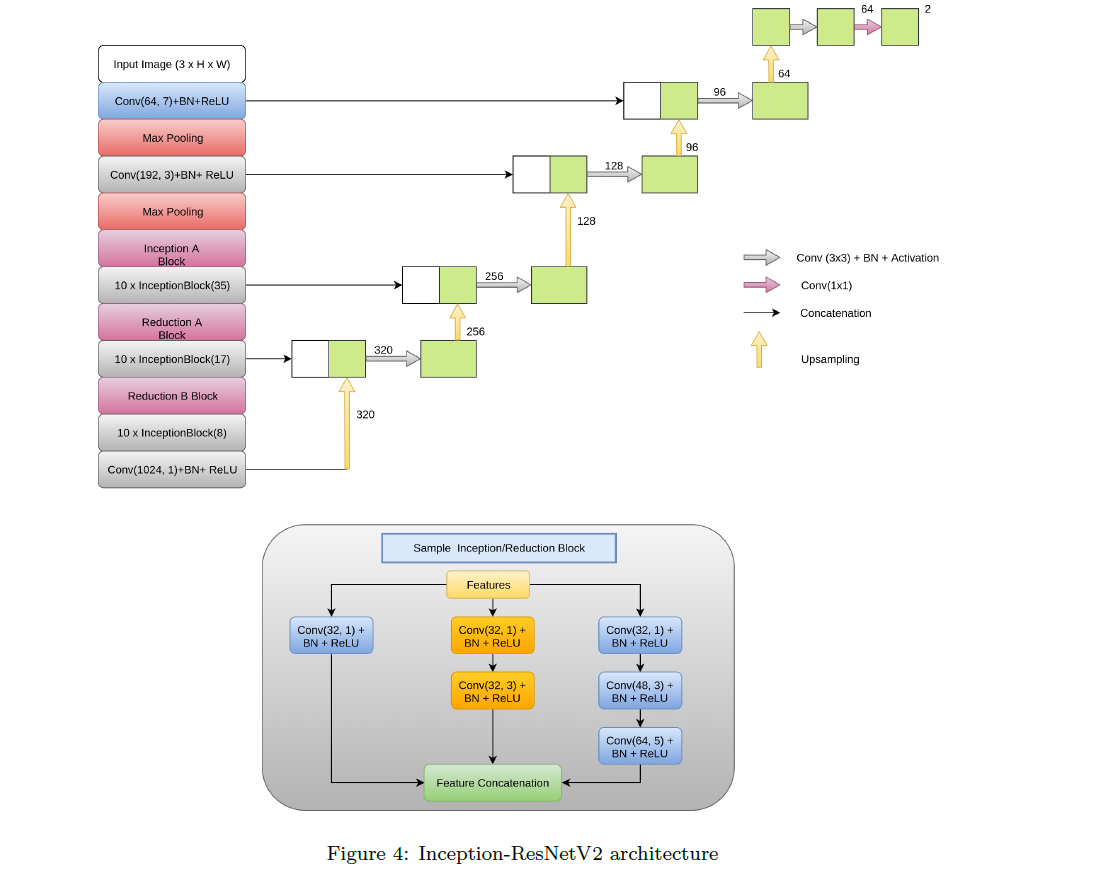
Q1: Check the 2D transpose convolutional layer, 2D upsampling convolution layer and concatenation layer in keras.

Q2:If the input size of the tensor is data=(16,224,224,3), labels=(16,224,224,4), what will be output shape or size of model output?

Q3: Please construct the following UNet model and check the shape of each layer from start to end.



Q4: Please check the shape of every layer of the following model and try to change the feature maps after every upsampling block as shown yellow arrow in the figure below.



Q5: Design segmentation model using the inception modules in encoder and ResNet module at decoder side.The input image size of the model should be 224x224x3.

Note: Use the following code for reference.

Input image size(224x224x3)

Inception\_Module a

Inception\_Module b

Inception\_Module c

Inception\_Module d

ResNet Block1

ResNet Block2

ResNet Block3

ResNet Block4

Input image size(224x224x3)

# residual block may can apply with any number of layers.

def convolution\_block(x, filters, size, strides=(1,1), padding='same', activation=True):

    x = Conv2D(filters, size, strides=strides, padding=padding)(x)

    x = BatchNormalization()(x)

    if activation == True:

        x = LeakyReLU(alpha=0.1)(x)

    return x

def residual\_block(blockInput, num\_filters=16):

    x = LeakyReLU(alpha=0.1)(blockInput)

    x = BatchNormalization()(x)

    blockInput = BatchNormalization()(blockInput)

    x = convolution\_block(x, num\_filters, (3,3) )

    x = convolution\_block(x, num\_filters, (3,3), activation=False)

    x = Add()([x, blockInput])

    return x

# inception function

def inception\_module(x,

                     filters\_1x1,

                     filters\_3x3\_reduce,

                     filters\_3x3,

                     filters\_5x5\_reduce,

                     filters\_5x5,

                     filters\_pool\_proj,

                     name=None):

    conv\_1x1 = Conv2D(filters\_1x1, (1, 1), padding='same', activation='relu', kernel\_initializer=kernel\_init, bias\_initializer=bias\_init)(x)

    conv\_3x3 = Conv2D(filters\_3x3\_reduce, (1, 1), padding='same', activation='relu', kernel\_initializer=kernel\_init, bias\_initializer=bias\_init)(x)

    conv\_3x3 = Conv2D(filters\_3x3, (3, 3), padding='same', activation='relu', kernel\_initializer=kernel\_init, bias\_initializer=bias\_init)(conv\_3x3)

    conv\_5x5 = Conv2D(filters\_5x5\_reduce, (1, 1), padding='same', activation='relu', kernel\_initializer=kernel\_init, bias\_initializer=bias\_init)(x)

    conv\_5x5 = Conv2D(filters\_5x5, (5, 5), padding='same', activation='relu', kernel\_initializer=kernel\_init, bias\_initializer=bias\_init)(conv\_5x5)

    pool\_proj = MaxPool2D((3, 3), strides=(1, 1), padding='same')(x)

    pool\_proj = Conv2D(filters\_pool\_proj, (1, 1), padding='same', activation='relu', kernel\_initializer=kernel\_init, bias\_initializer=bias\_init)(pool\_proj)

    output = concatenate([conv\_1x1, conv\_3x3, conv\_5x5, pool\_proj], axis=3, name=name)

    return output

# these the inception blocks

x = inception\_module(x,

                     filters\_1x1=64,

                     filters\_3x3\_reduce=96,

                     filters\_3x3=128,

                     filters\_5x5\_reduce=16,

                     filters\_5x5=32,

                     filters\_pool\_proj=32,

                     name='inception\_3a')

x = inception\_module(x,

                     filters\_1x1=128,

                     filters\_3x3\_reduce=128,

                     filters\_3x3=192,

                     filters\_5x5\_reduce=32,

                     filters\_5x5=96,

                     filters\_pool\_proj=64,

                     name='inception\_3b')

x = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='max\_pool\_3\_3x3/2')(x)

x = inception\_module(x,

                     filters\_1x1=192,

                     filters\_3x3\_reduce=96,

                     filters\_3x3=208,

                     filters\_5x5\_reduce=16,

                     filters\_5x5=48,

                     filters\_pool\_proj=64,

                     name='inception\_4a')

x1 = AveragePooling2D((5, 5), strides=3)(x)

x1 = Conv2D(128, (1, 1), padding='same', activation='relu')(x1)

x1 = Flatten()(x1)

x1 = Dense(1024, activation='relu')(x1)

x1 = Dropout(0.7)(x1)

x1 = Dense(10, activation='softmax', name='auxilliary\_output\_1')(x1)

x = inception\_module(x,

                     filters\_1x1=160,

                     filters\_3x3\_reduce=112,

                     filters\_3x3=224,

                     filters\_5x5\_reduce=24,

                     filters\_5x5=64,

                     filters\_pool\_proj=64,

                     name='inception\_4b')

x = inception\_module(x,

                     filters\_1x1=128,

                     filters\_3x3\_reduce=128,

                     filters\_3x3=256,

                     filters\_5x5\_reduce=24,

                     filters\_5x5=64,

                     filters\_pool\_proj=64,

                     name='inception\_4c')

x = inception\_module(x,

                     filters\_1x1=112,

                     filters\_3x3\_reduce=144,

                     filters\_3x3=288,

                     filters\_5x5\_reduce=32,

                     filters\_5x5=64,

                     filters\_pool\_proj=64,

                     name='inception\_4d')