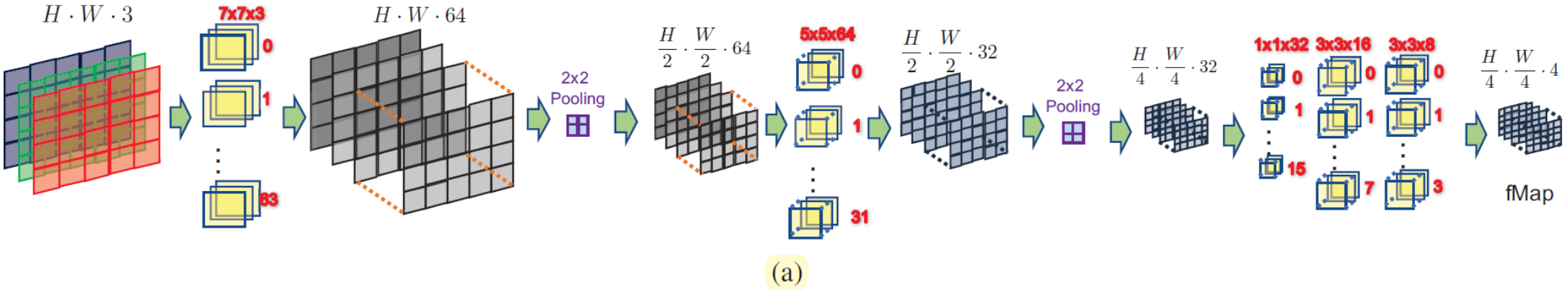
# VCIP2017




- Compression ratio: dimension of the image after compression/dimension before compression

- $\frac{\frac{H}{4}\times\frac{W}{4}\times 4}{H\times W\times 3} = \frac{1}{12}$

- Plot the PSNR curves versus varying compression ratios, for example: 1/32, 1/16, 1/12, 1/8, ¼.

# How to decompress the image from fMap?

```
input_coarse = Input(shape = (b, b, 3))

e = Conv2D(128, kernel_size=(5, 5), padding = "SAME", strides = (4,4), activation='relu', input_shape=(b, b, 3))(input_coarse)
e = Conv2D(64, kernel_size=(5, 5), padding = "SAME", strides = (1,1), activation='relu')(e)
e = Conv2D(3, kernel_size=(5, 5), padding = "SAME", strides = (1,1), activation='relu')(e)

d = Conv2DTranspose(64, kernel_size=(5, 5), padding = "SAME", strides = (1,1), activation='relu')(e)
d = Conv2DTranspose(128, kernel_size=(5, 5), padding = "SAME", strides = (1,1), activation='relu')(d)
d = Conv2DTranspose(3, kernel_size=(5, 5), padding = "SAME", strides = (4, 4), activation='relu')(d)

coarse_model = Model(inputs = input_coarse, outputs = d)
coarse_model.summary()
coarse_model.compile(optimizer='adam', loss='mse')
```

# How to decompress the image from fMap?

```
input_coarse = Input(shape = (b, b, 3))

e = Conv2D(128, kernel_size=(5, 5), padding = "SAME", strides = (4,4), activation='relu', input_shape=(b, b, 3))(input_coarse)
e = Conv2D(64, kernel_size=(5, 5), padding = "SAME", strides = (2,2), activation='relu')(e)
e = Conv2D(3, kernel_size=(5, 5), padding = "SAME", strides = (1,1), activation='relu')(e)
e = Conv2D(3, kernel_size=(5, 5), padding = "SAME", strides = (1,1), activation='relu')(e)

d = Conv2DTranspose(64, kernel_size=(5, 5), padding = "SAME", strides = (1,1), activation='relu')(e)
d = Conv2DTranspose(64, kernel_size=(5, 5), padding = "SAME", strides = (1,1), activation='relu')(e)
d = Conv2DTranspose(128, kernel_size=(5, 5), padding = "SAME", strides = (2,2), activation='relu')(d)
d = Conv2DTranspose(3, kernel_size=(5, 5), padding = "SAME", strides = (4, 4), activation='relu')(d)

coarse_model = Model(inputs = input_coarse, outputs = d)
coarse_model.summary()
coarse_model.compile(optimizer='adam', loss='mse')
```

# How to decompress the image from fMap?

```
input_layer = Input(shape=(b,b,3)) # block size: bxb
c1 = Conv2D(128, kernel_size=(3,3), activation='relu', padding='same', strides = (1,1))(input_layer)
l = MaxPool2D(strides=(2,2))(c1) # b/2 x b/2
c2 = Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same')(l)
l = MaxPool2D(strides=(2,2))(c2) # b/4 x b/4
c3 = Conv2D(filters=1, kernel_size=(3,3), activation='relu', padding='same')(l)

l = Conv2D(filters=1, kernel_size=(3,3), activation='relu', padding='same')(c3)
l = UpSampling2D(size=(2,2))(l)
l = Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same')(l)
l = UpSampling2D(size=(2,2))(l)
l = Conv2D(filters=128, kernel_size=(3,3), activation='relu', padding='same')(l)
d = Conv2D(filters=3, kernel_size=(1,1), activation='linear')(l)

coarse_model = Model(inputs = input_layer, outputs = d)
coarse_model.summary()
coarse_model.compile(optimizer='adam', loss='mse')
```

# VCIP2017

- *II.A Residual Encoder: not required*
- *II.B Predictive Encoder: not required*
- *II.C Quantization and Entropy Coding: no need*

- *If the frame size is too big, you can divide each frame into small patches (blocks) and design a block-based video encoder and decoder system. For example, use block size 32x32*
- *Then for testing, you also need to divide the test frames into blocks of the same size. But PSNR is calculated after re-grouping all blocks into the original big frame.*