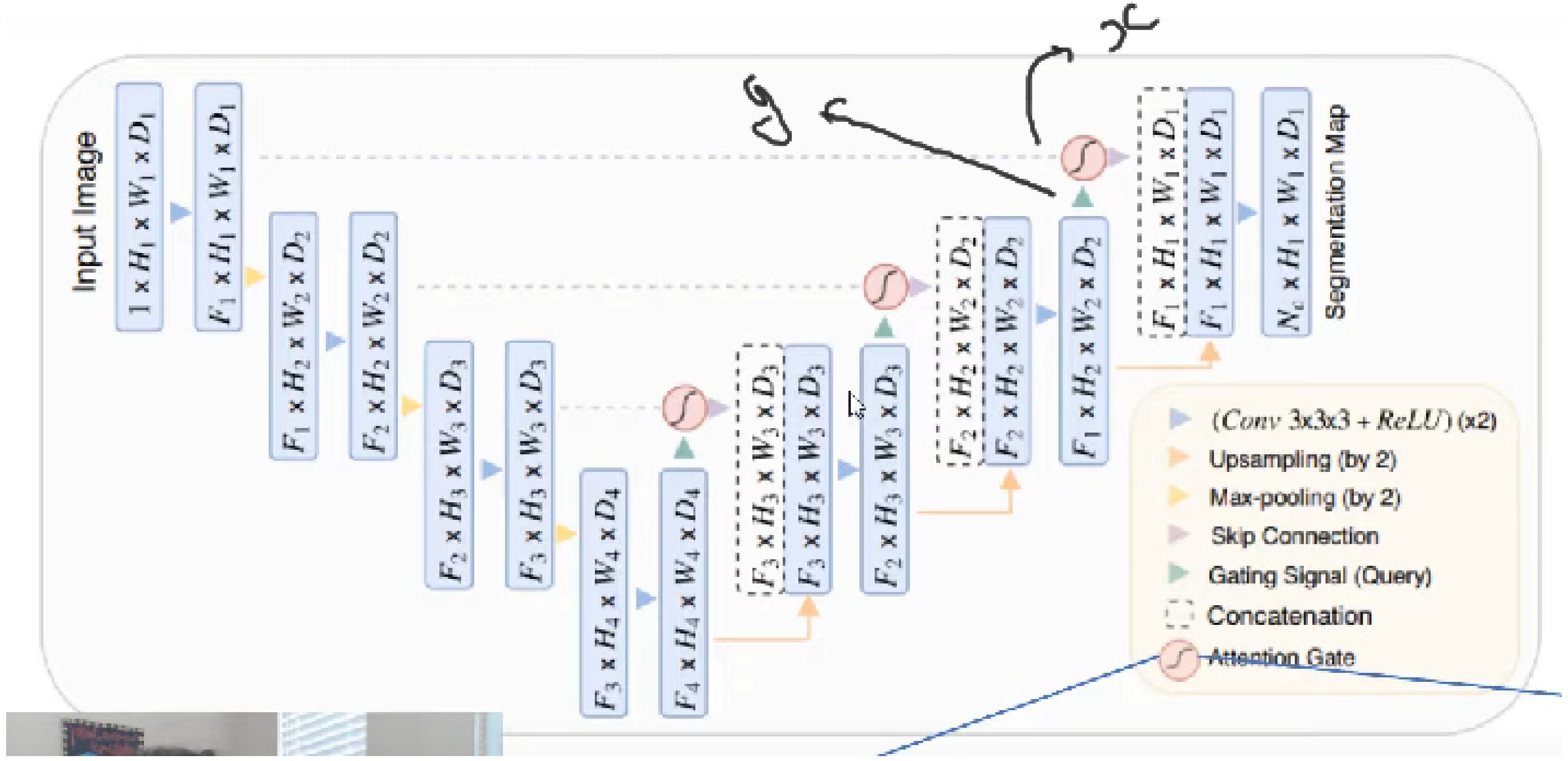
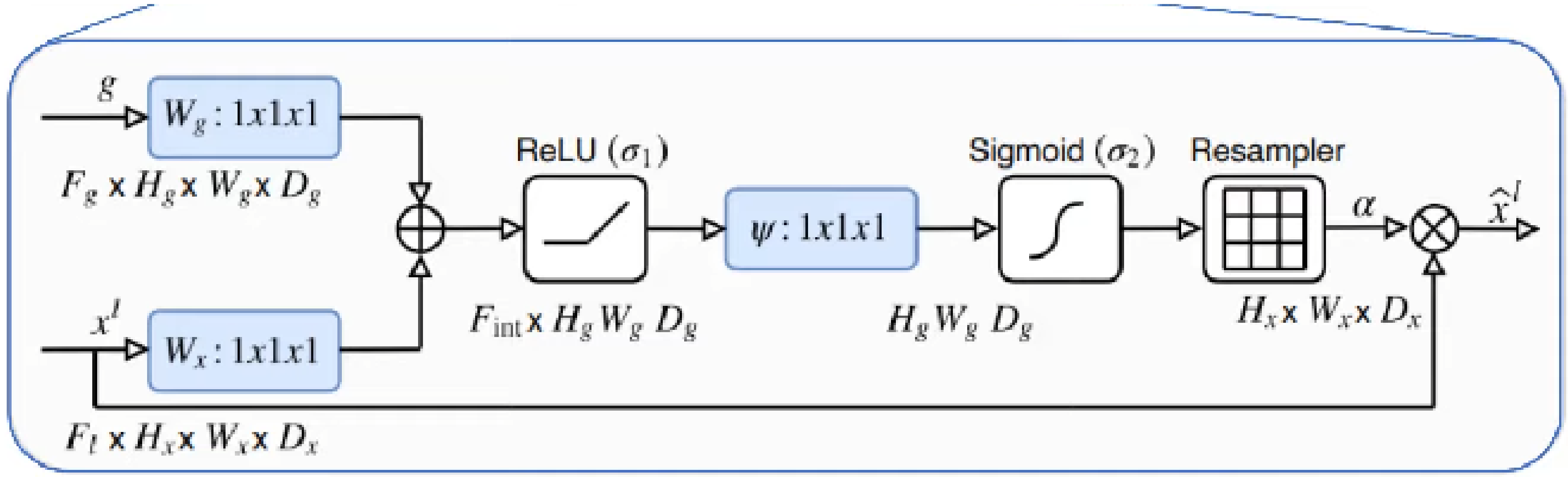




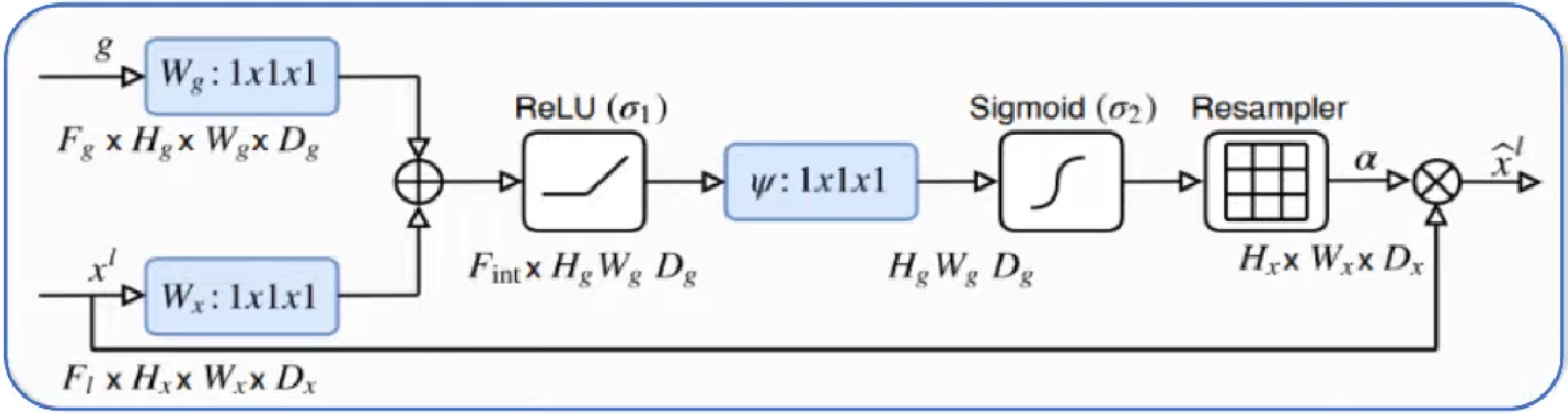
attention gate takes x & g

x ⇒ comes from skip connection

↳ it provide better information

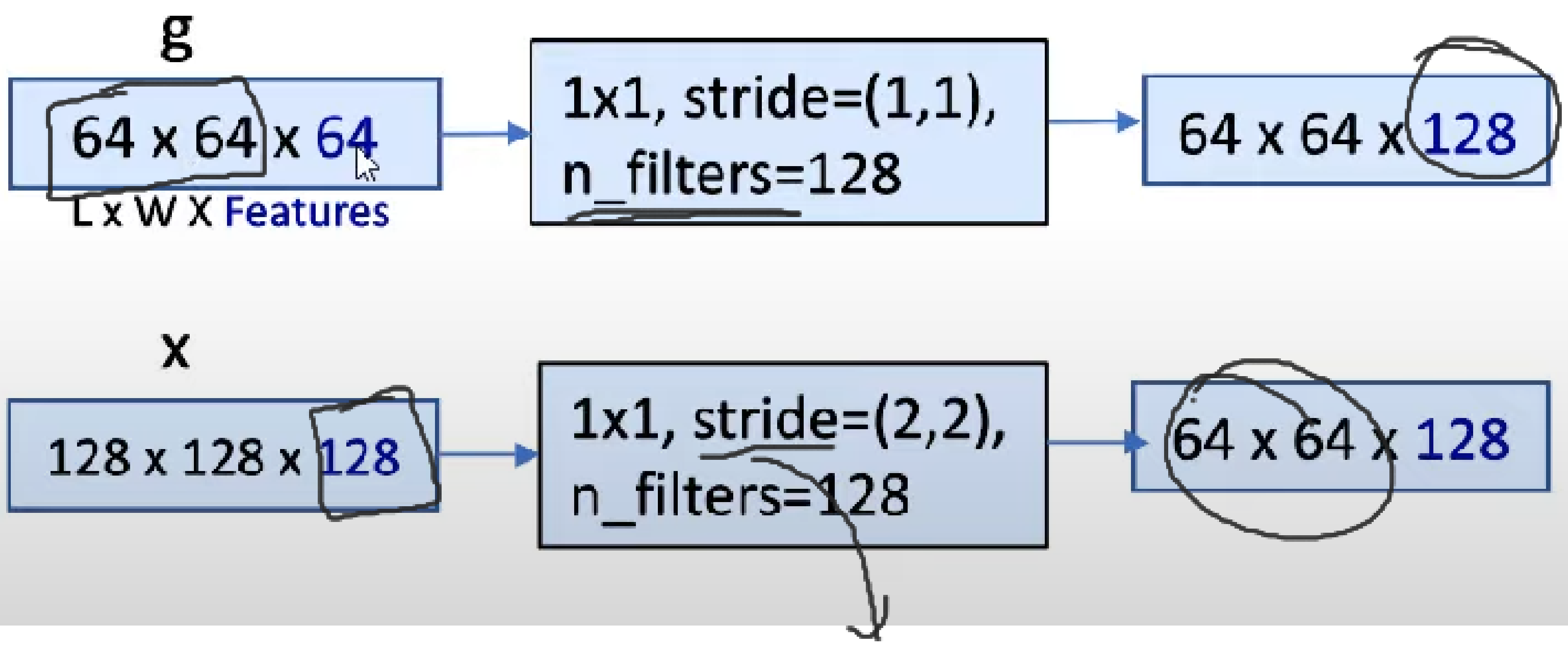
g ⇒ gating signal ⇒ comes from lowest layer of network

↳ it provide better representation

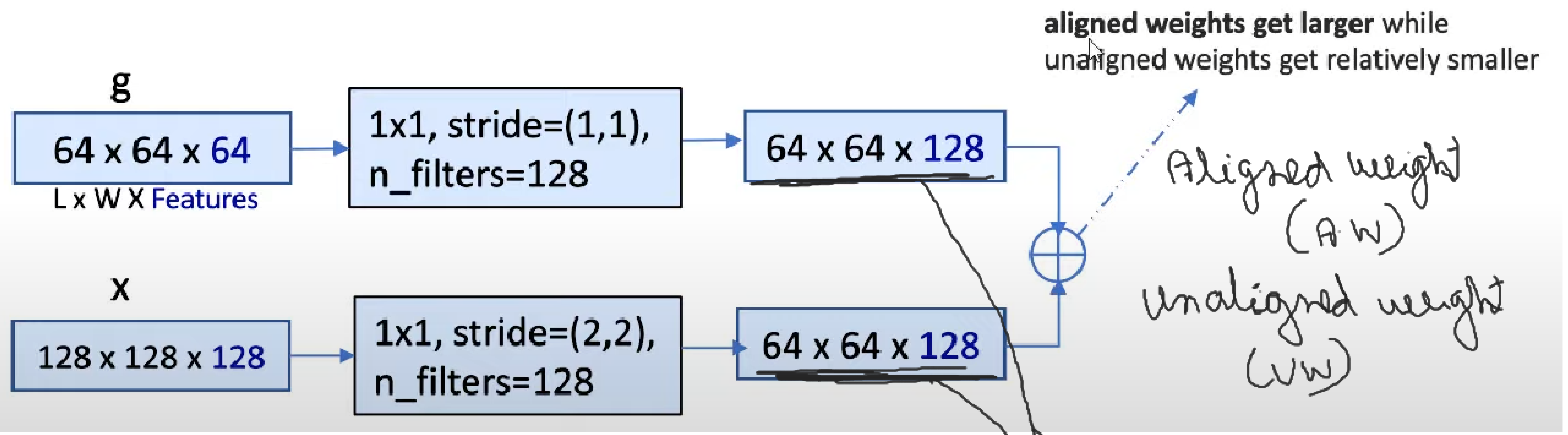


Taking size of g [64 x 64]

Taking Channel of x [128]



size become half



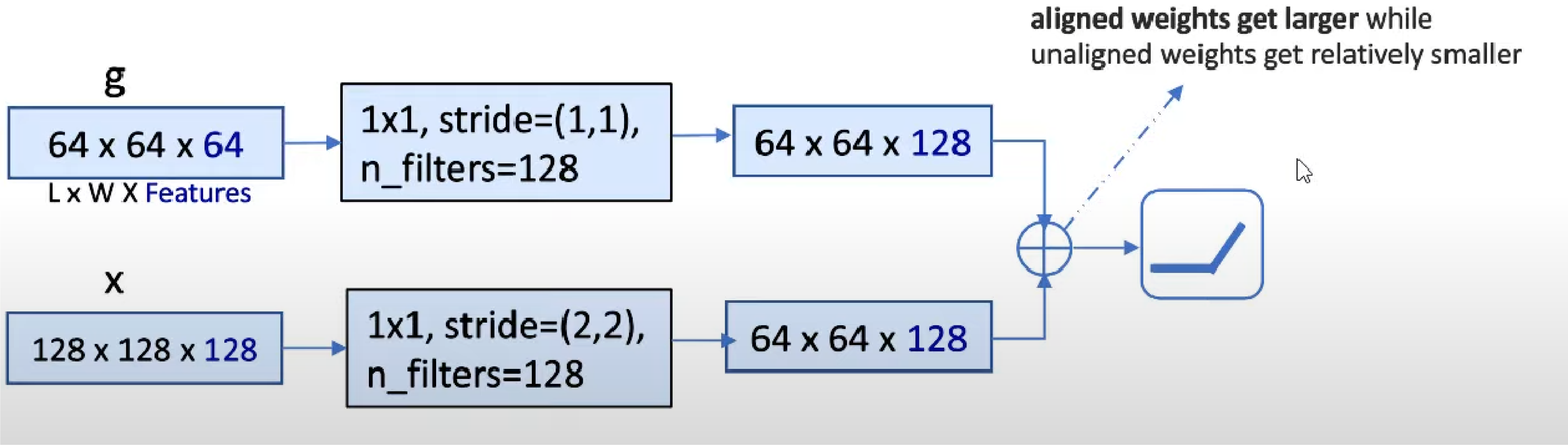
as they are of same size with channel we could add them

The reason to add both is that ↑ aligned weight which makes weight of unaligned weight much smaller,

eg → $A.W = 0.9 \rightarrow 0.9 + 0.9 \Rightarrow 1.8$

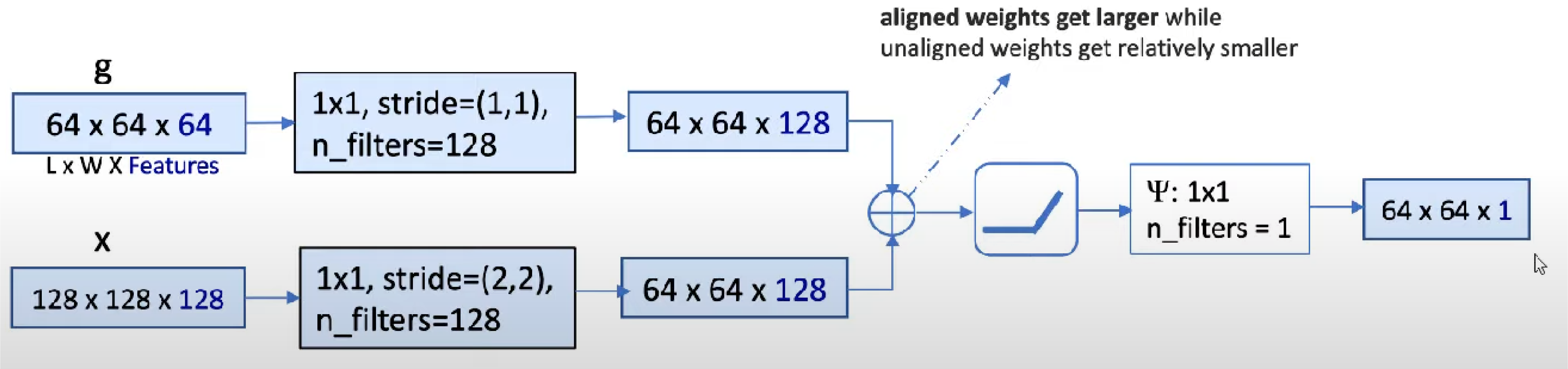
$U.W = 0.1 \rightarrow 0.1 + 0.1 \Rightarrow 0.2$

relatively smaller



After adding, passing it through ReLU activation function which represent

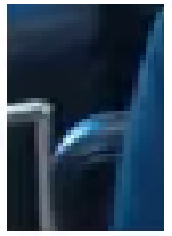
$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$



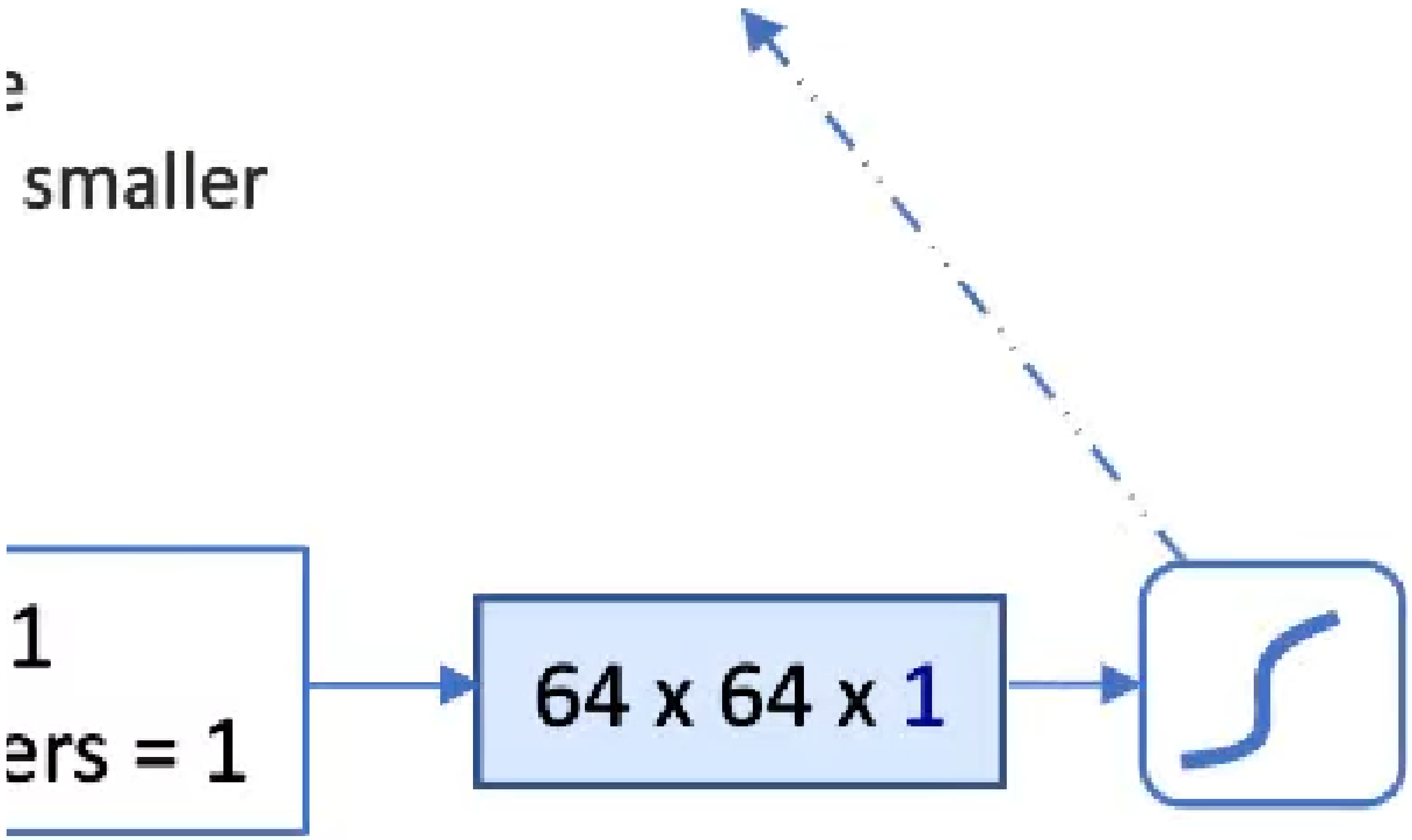
passing through Ψ function with filters = 1
which gives output as 64 x 64 x 1

These are nothing but weights but its range could be
zero → ∞ as we are using ReLU

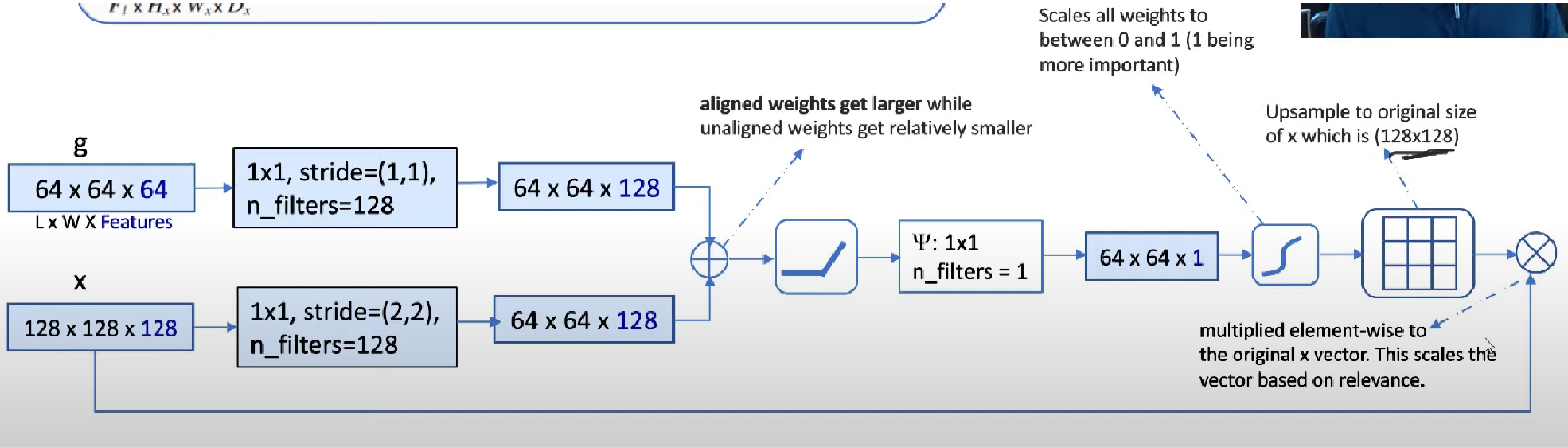
Scales all weights to between 0 and 1 (1 being more important)



e
smaller



So with the help of sigmoid activation function
all value comes in range
from 0 to 1

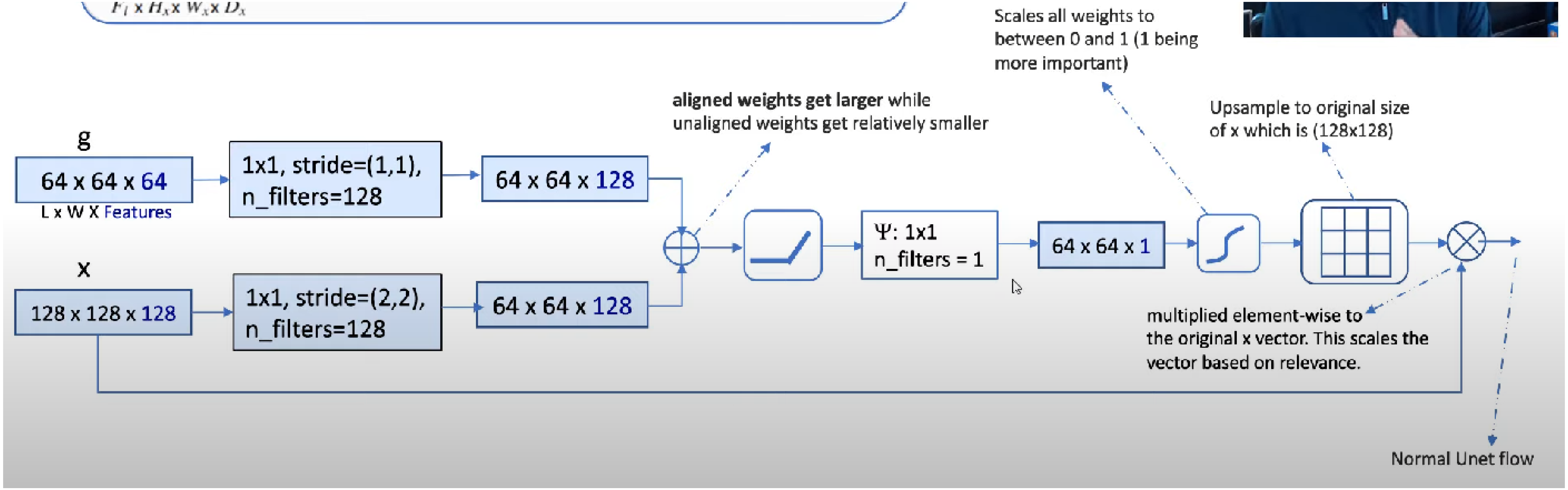


as we get all value in range $0 \to 1$ we have to upsample to the original size of x

And multiple with x, which scales the relevent part of image

We can say that, at each pixel that coming from x we multiplying the pixel value with the weight value (that we calculated using g)

And output goes to next layer at normal U-net flow

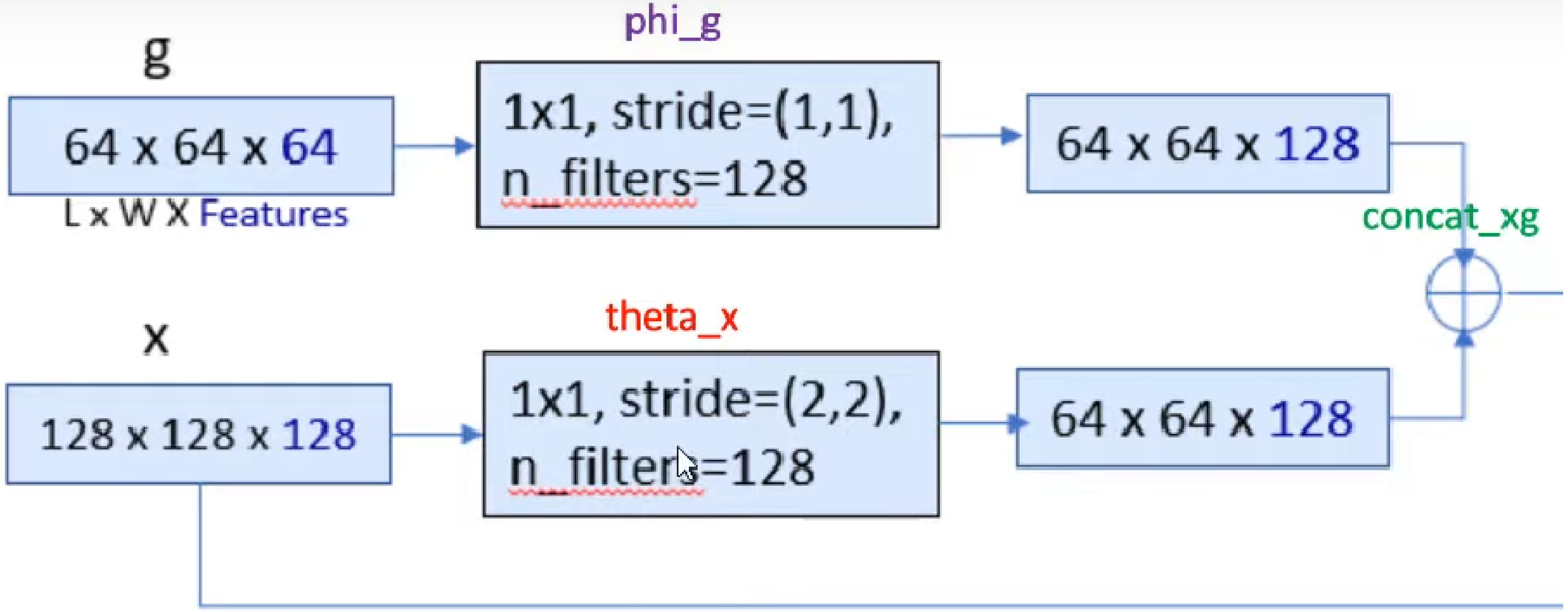


F_l x H_x x W_x x D_x
Scales all weights to between 0 and 1 (1 being more important)
aligned weights get larger while unaligned weights get relatively smaller
Upsample to original size of x which is (128x128)
g
64 x 64 x 64
L x W X Features
1x1, stride=(1,1), n_filters=128
64 x 64 x 128
x
128 x 128 x 128
1x1, stride=(2,2), n_filters=128
64 x 64 x 128
Ψ: 1x1 n_filters = 1
64 x 64 x 1
multiplied element-wise to the original x vector. This scales the vector based on relevance.
Normal Unet flow

Code Implementation

of

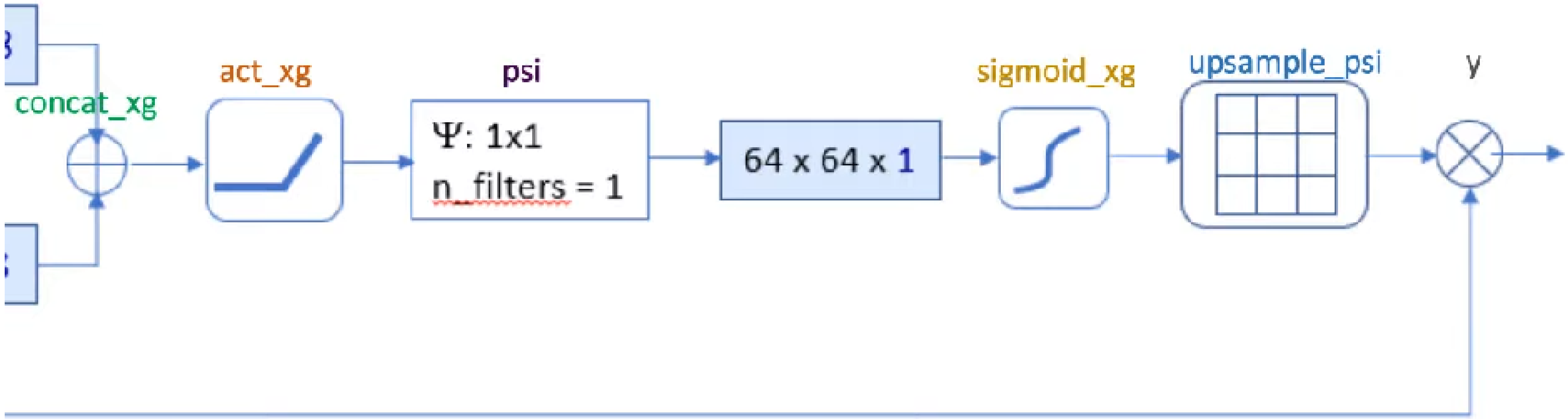
Attention U-Net




```
def attention_block(x, gating, inter_shape):
    shape_x = K.int_shape(x)
    shape_g = K.int_shape(gating)

# Getting x to the same shape as the gating signal
    theta_x = layers.Conv2D(inter_shape, (1, 1), strides=(2, 2), padding='same')(x)
    shape_theta_x = K.int_shape(theta_x)

# Getting the gating signal to the same number of filters as the inter_shape
    phi_g = layers.Conv2D(inter_shape, (1, 1), padding='same')(gating)
```



)(x)

```
concat_xg = layers.add([phi_g, theta_x])

  act_xg = layers.Activation('relu')(concat_xg)

  psi = layers.Conv2D(1, (1, 1), padding='same')(act_xg)

  sigmoid_xg = layers.Activation('sigmoid')(psi)
  shape_sigmoid = K.int_shape(sigmoid_xg)

  upsample_psi = layers.UpSampling2D(size=(shape_x[1] // shape_sigmoid[1], shape_x[2] //
shape_sigmoid[2]))(sigmoid_xg)


  y = layers.multiply([upsample_psi, x])

  result = layers.Conv2D(shape_x[3], (1, 1), padding='same')(y)
  result_bn = layers.BatchNormalization()(result)

  return result_bn
```