

# Stereo Matching

Quang-Vinh Dinh  
Ph.D. in Computer Science

# Stereo Matching

## ❖ Using deep learning



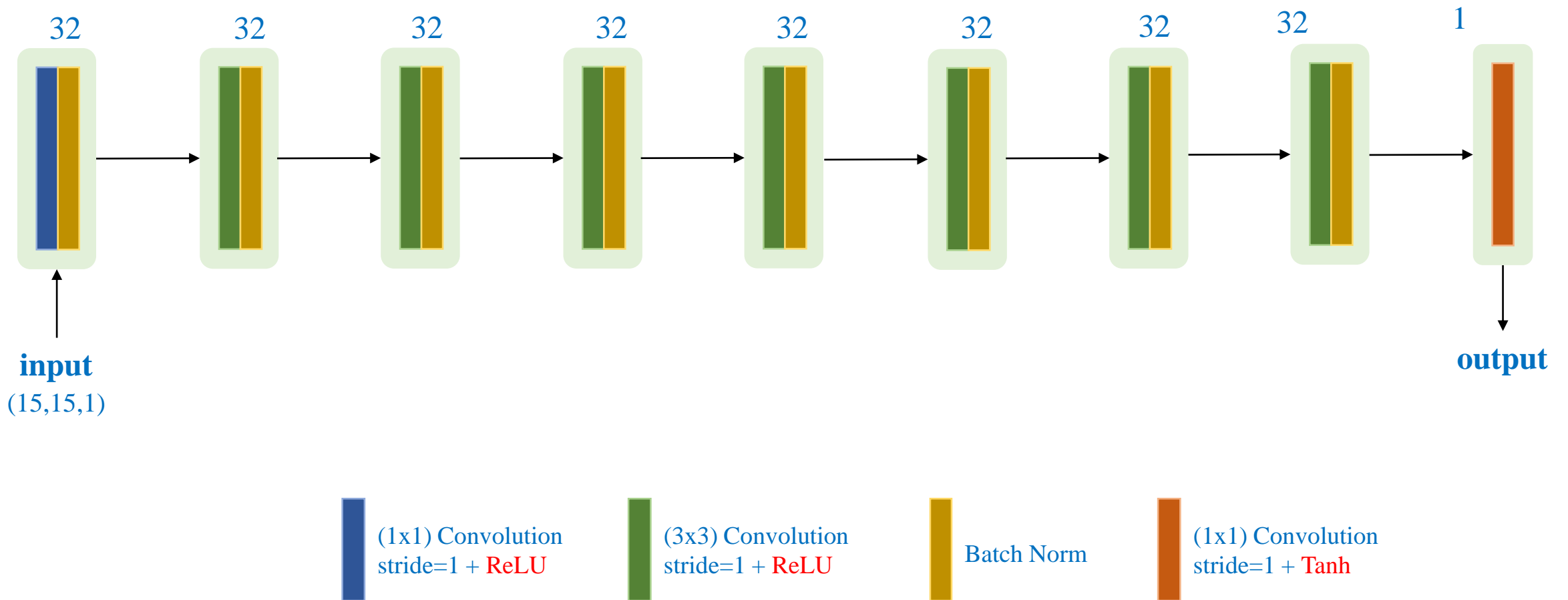
(a)

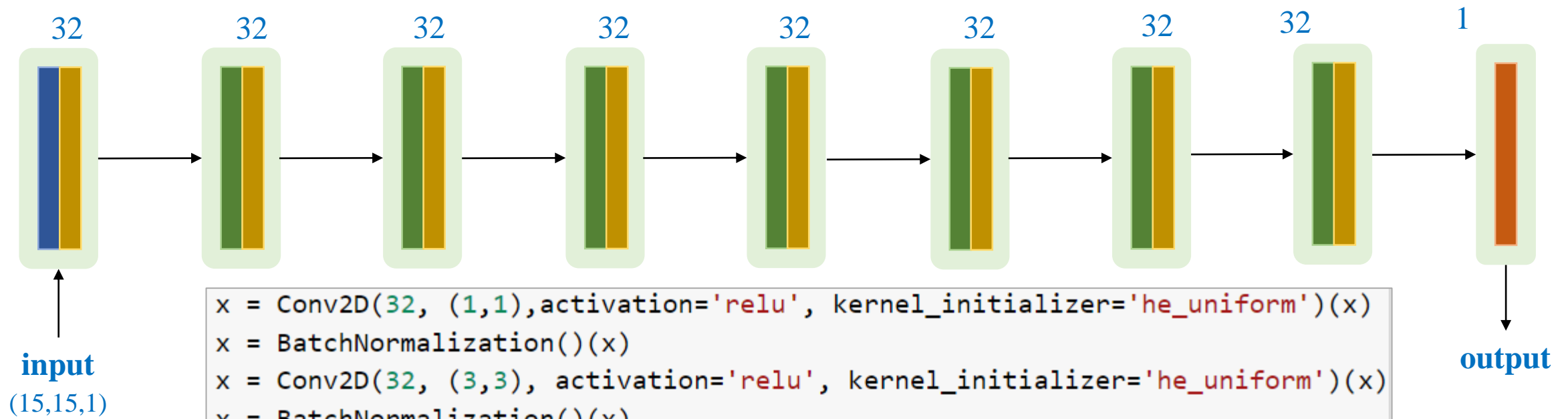


(b)

# Stereo Matching

## ❖ Using deep learning





```

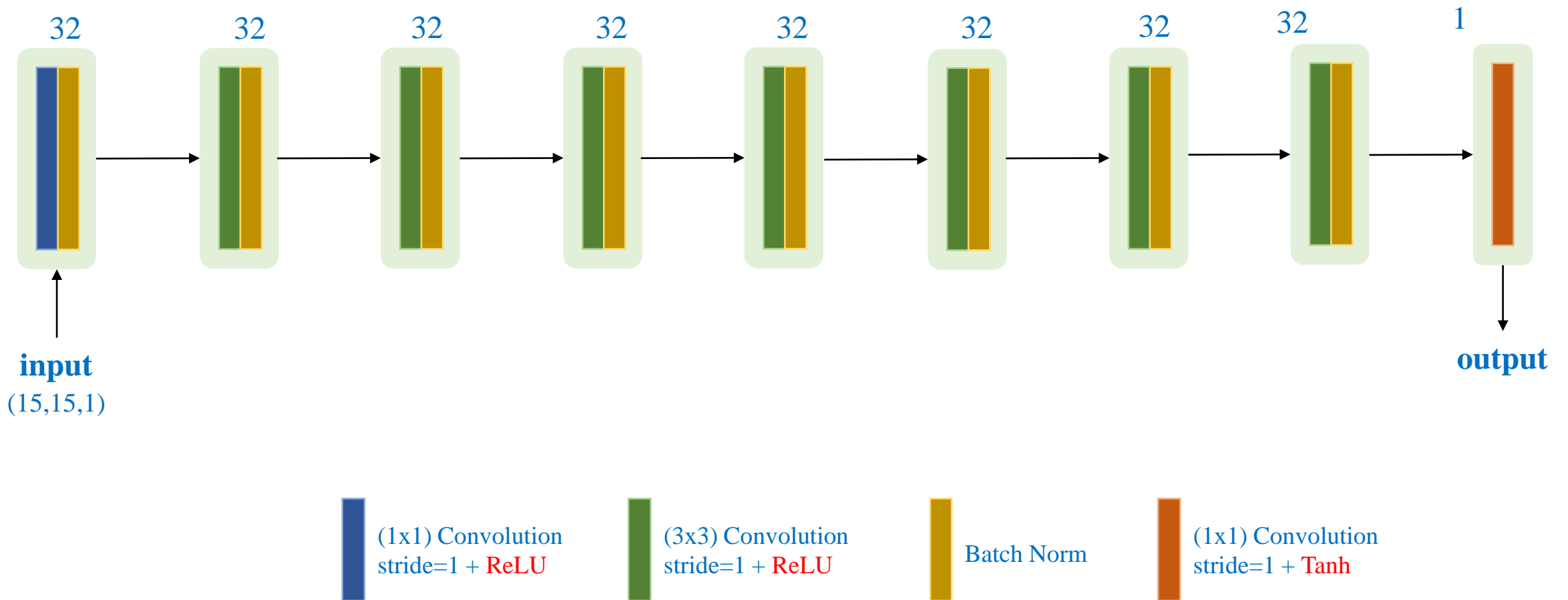
x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
x = BatchNormalization()(x)
x = Conv2D(1, (1,1), activation='tanh')(x)

```

# Stereo Matching

## ❖ Implementation

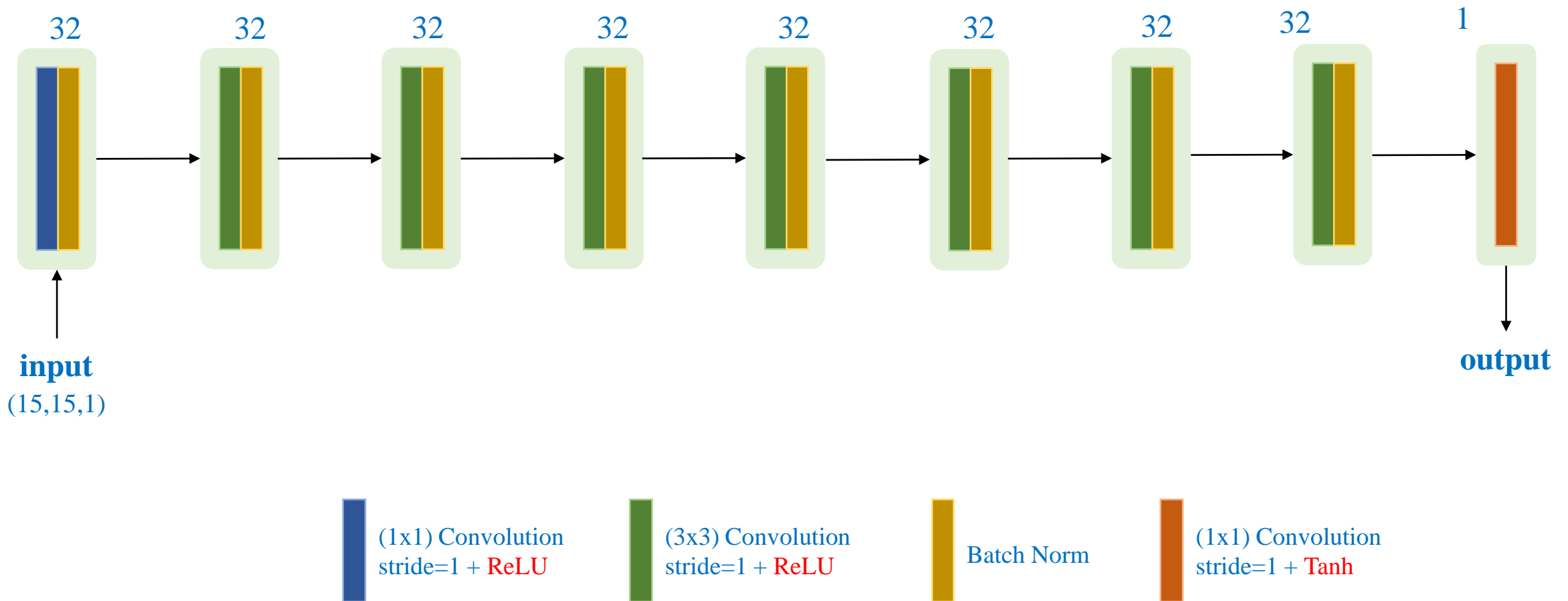
### ❖ Problem with shapes



# Stereo Matching

## ❖ Implementation

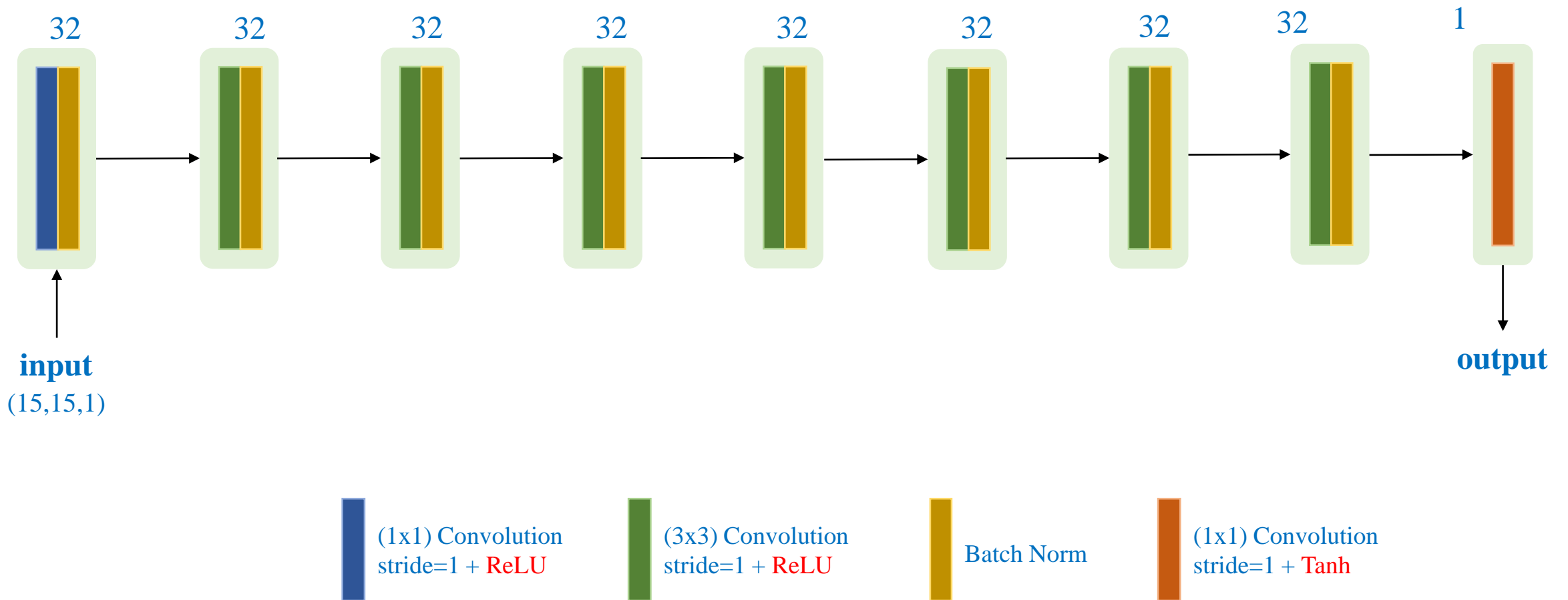
### ❖ Problem with fixed sizes



# Stereo Matching

## ❖ Implementation

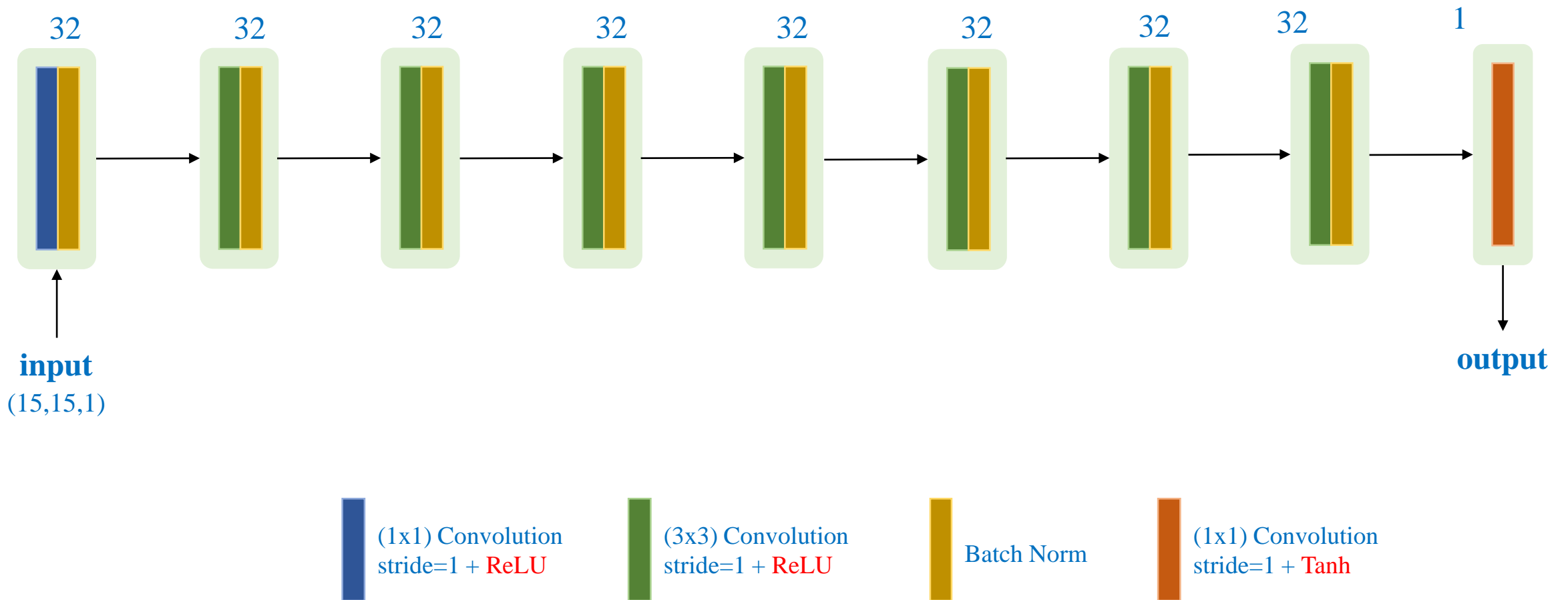
### ❖ Using the fit function



# Stereo Matching

## ❖ Implementation

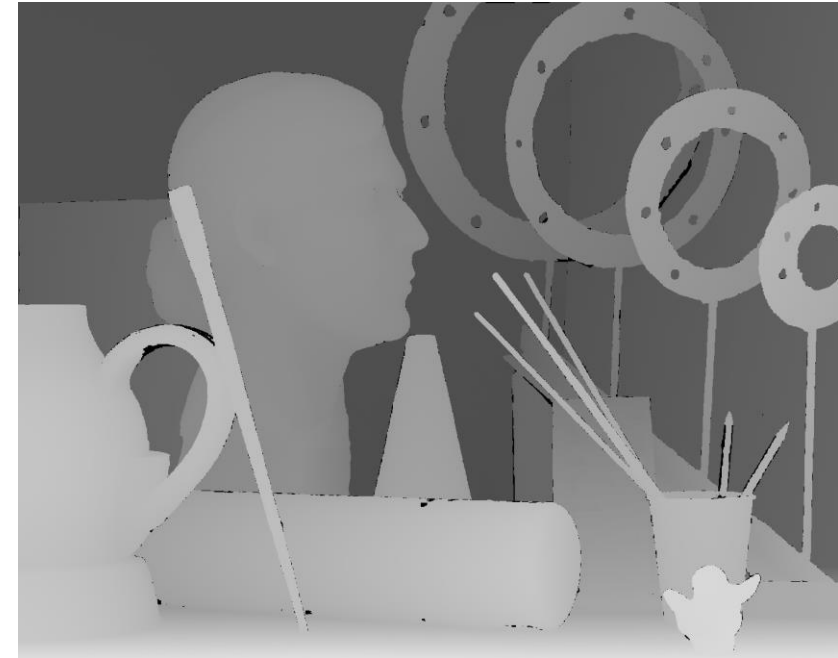
### ❖ Using GradientTape





# Stereo Matching

## ❖ Dataset



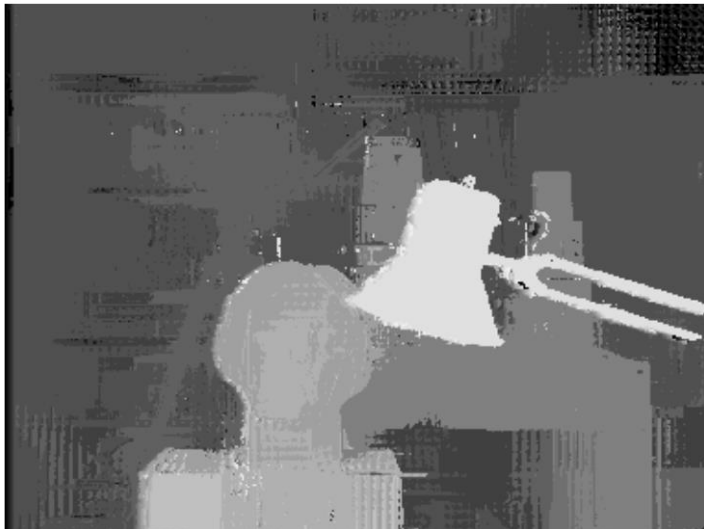
# Stereo Matching

## ❖ Post-processing methods

$w=3$

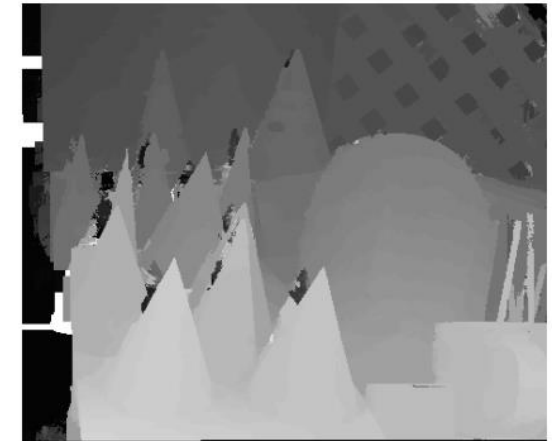
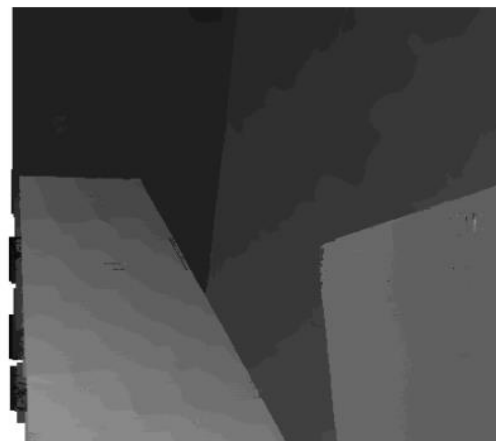
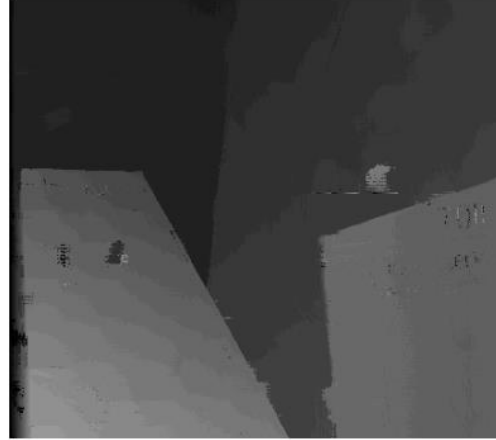
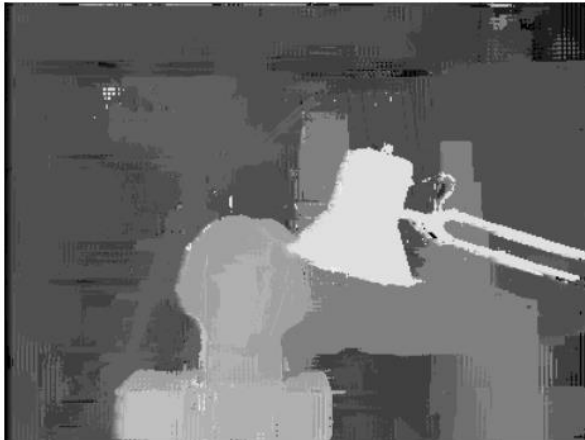


$w=5$



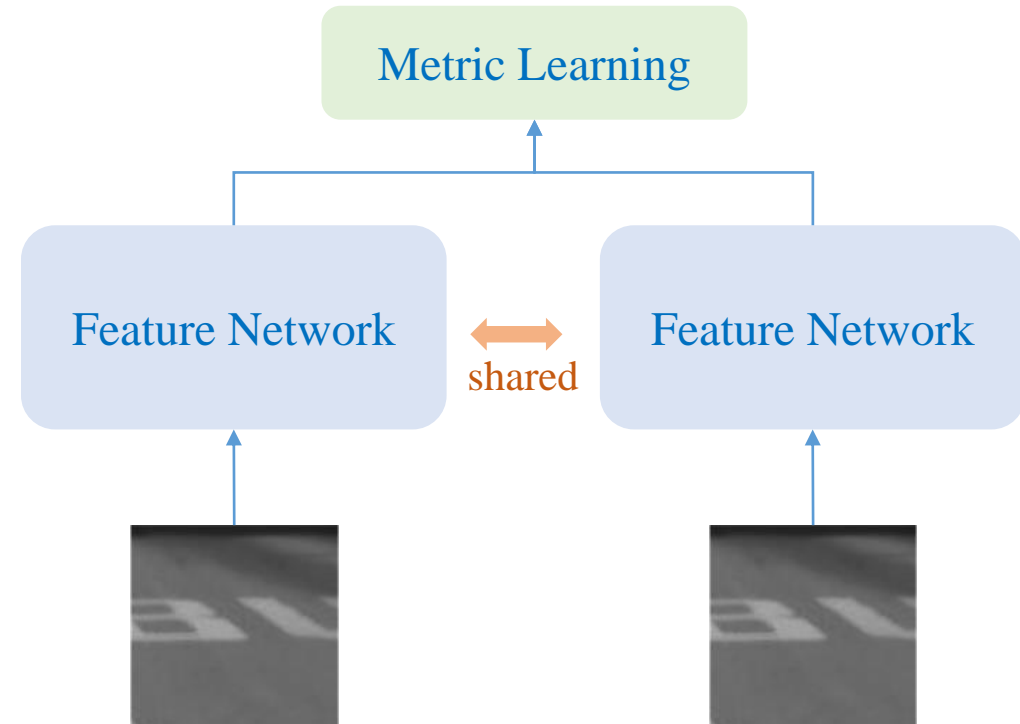
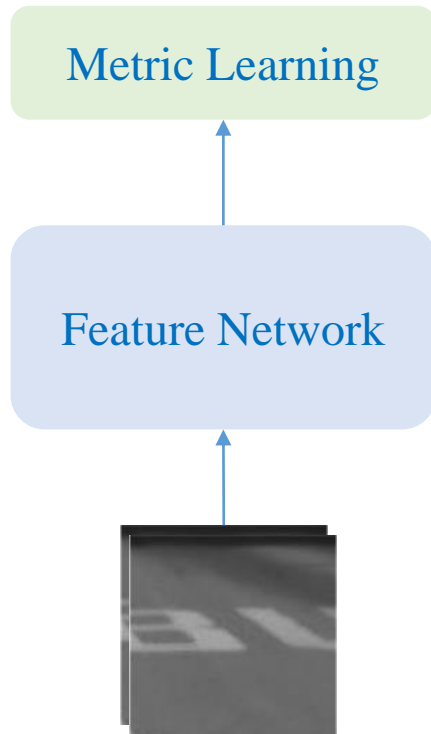
# Stereo Matching

## ❖ Post-processing methods



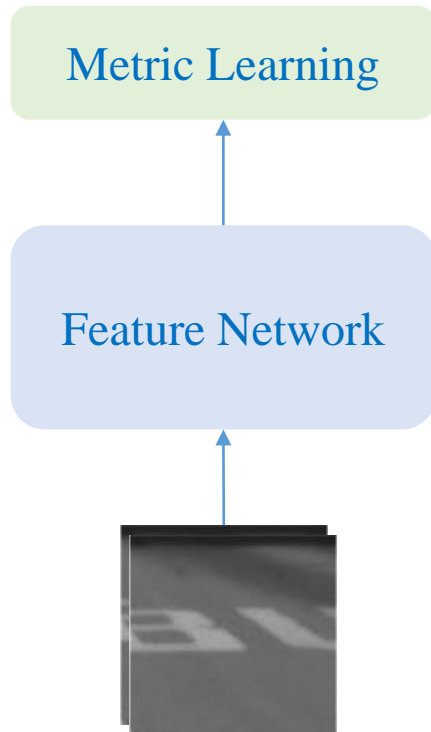
# Stereo Matching

## ❖ Siamese Networks



# Stereo Matching

## ❖ Siamese Networks

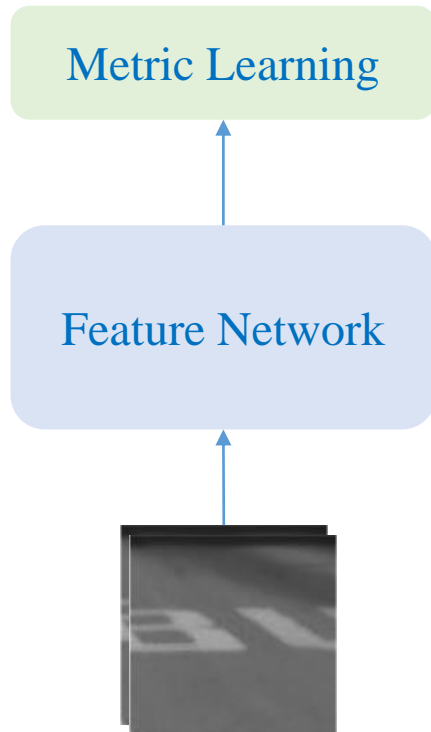


```
1 from tensorflow.keras.models import Model
2 from tensorflow.keras.layers import Input, Dense, Conv2D, BatchNormalization
3 import tensorflow as tf
4
5 def base_model(input_shape):
6     inputs = Input(shape=input_shape)
7     x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(inputs)
8     x = BatchNormalization()(x)
9     x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
10    x = BatchNormalization()(x)
11    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
12    x = BatchNormalization()(x)
13    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
14    x = BatchNormalization()(x)
15    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
16    x = BatchNormalization()(x)
17    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
18    x = BatchNormalization()(x)
19    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
20    x = BatchNormalization()(x)
21    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
22    x = BatchNormalization()(x)
23    x = Conv2D(1, (1,1), activation='tanh')(x)
24    x = tf.squeeze(x)
25    model = Model(inputs=inputs, outputs=x)
26    return model
```



# Stereo Matching

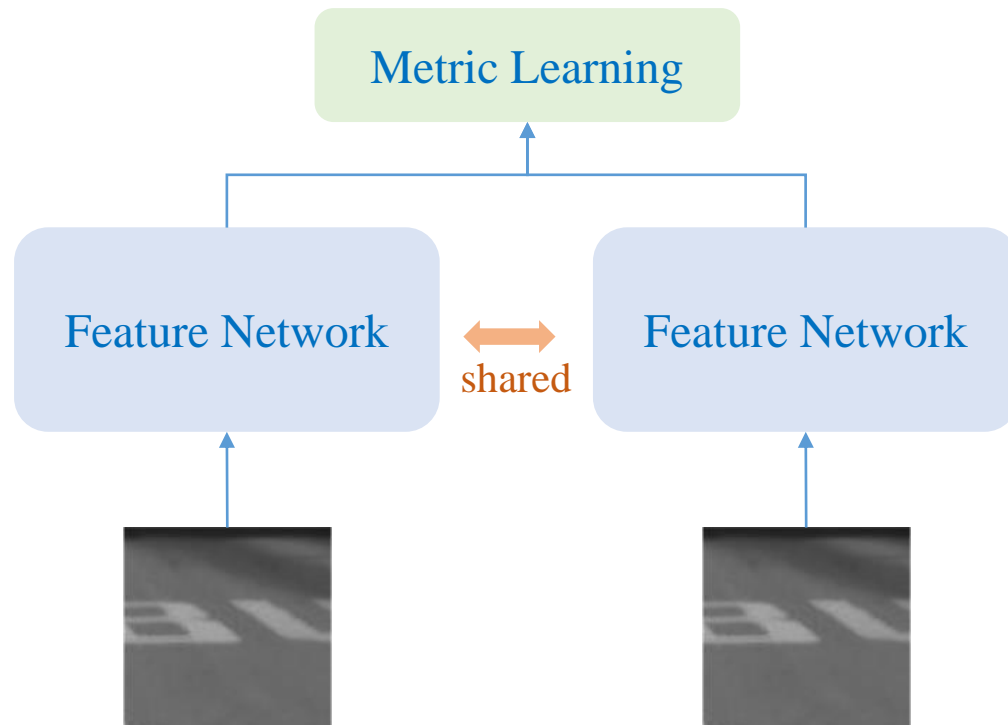
## ❖ Siamese Networks



```
1 def base_model(input_shape):
2     inputs = Input(shape=input_shape)
3
4     # feature network
5     x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(inputs)
6     x = BatchNormalization()(x)
7     x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
8     x = BatchNormalization()(x)
9     x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
10    x = BatchNormalization()(x)
11    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
12    x = BatchNormalization()(x)
13    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
14    x = BatchNormalization()(x)
15    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
16    x = BatchNormalization()(x)
17    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
18    x = BatchNormalization()(x)
19    x = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform')(x)
20    x = BatchNormalization()(x)
21
22    # metric learning
23    x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(x)
24    x = BatchNormalization()(x)
25    x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(x)
26    x = BatchNormalization()(x)
27    x = Conv2D(1, (1,1), activation='tanh')(x)
28    x = tf.squeeze(x)
29
30    model = Model(inputs=inputs, outputs=x)
31    return model
```

# Stereo Matching

## ❖ Siamese Networks

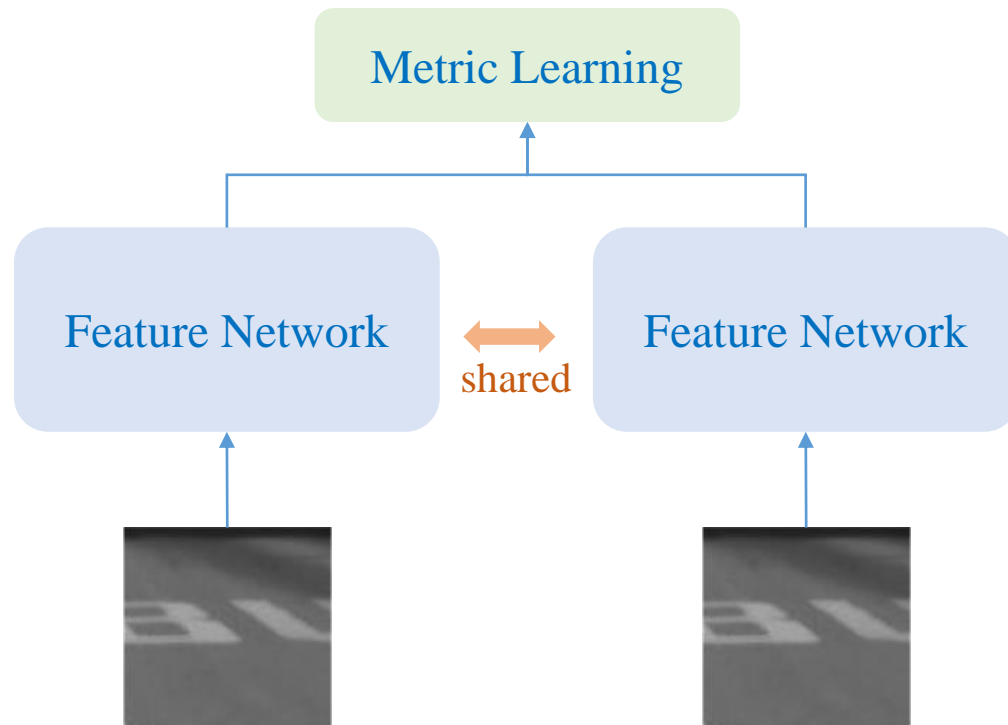


```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Conv2D, BatchNormalization
import tensorflow as tf

shared_weights = tf.keras.Sequential([
    Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform'),
    BatchNormalization(),
    Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform'),
    BatchNormalization(),
    Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform'),
    BatchNormalization(),
    Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform'),
    BatchNormalization(),
    Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform'),
    BatchNormalization(),
    Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform'),
    BatchNormalization(),
    Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform'),
    BatchNormalization(),
    Conv2D(32, (3,3), activation='relu', kernel_initializer='he_uniform'),
    BatchNormalization(),
])
```

# Stereo Matching

## ❖ Siamese Networks



```
def base_model(input_shape):  
    inputs_l = tf.keras.layers.Input(shape=input_shape)  
    inputs_r = tf.keras.layers.Input(shape=input_shape)  
  
    x_l = inputs_l  
    x_r = inputs_r  
  
    x_l = shared_weights(x_l)  
    x_r = shared_weights(x_r)  
  
    # combined  
    concat_input = tf.keras.layers.Concatenate()  
    x = concat_input([x_l, x_r])  
  
    x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(x)  
    x = Conv2D(1, (1,1), activation='sigmoid')(x)  
    x = tf.squeeze(x)  
    model = Model(inputs=[inputs_l, inputs_r], outputs=x)  
    return model
```



