Stereo Matching

Quang-Vinh Dinh
Ph.D. in Computer Science

Stereo Matching

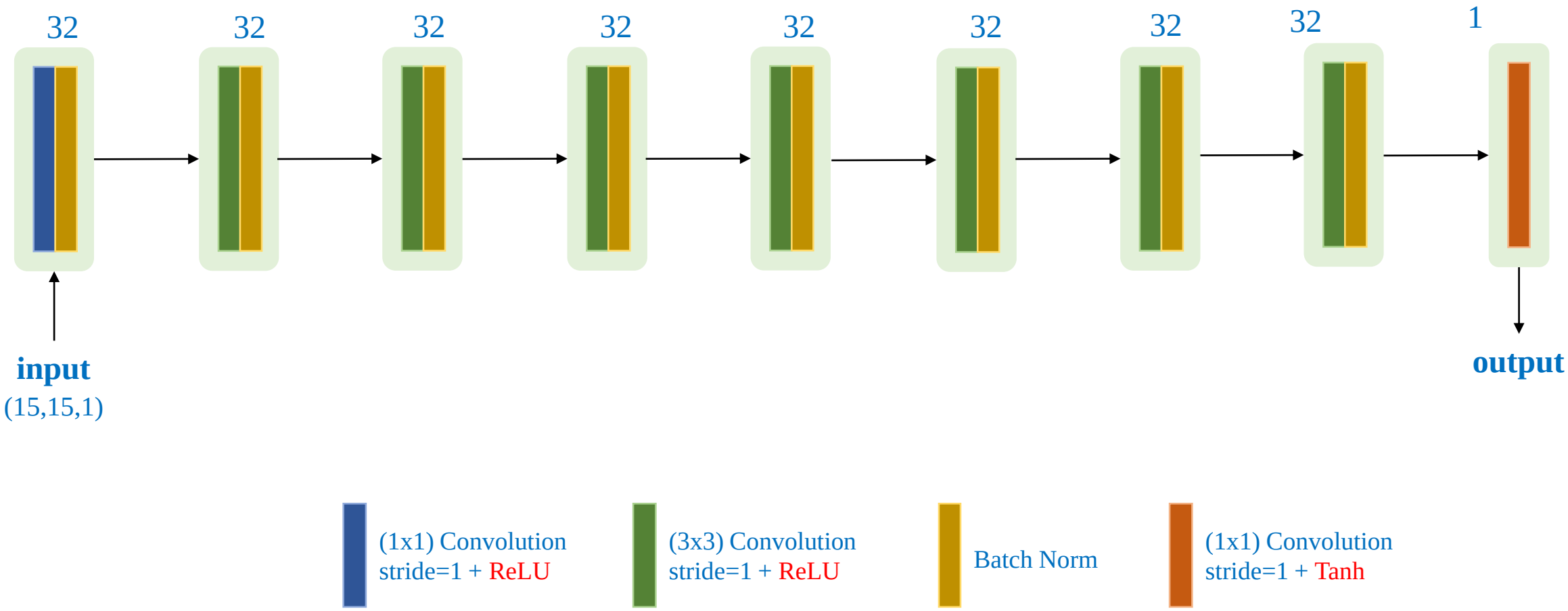
❖ Using deep learning

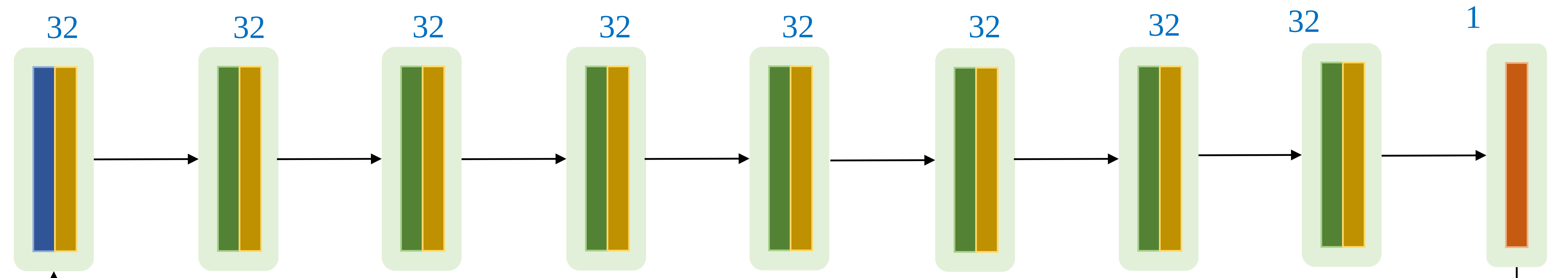
(a)

(b)

Stereo Matching

❖ Using deep learning





input
(15,15,1)

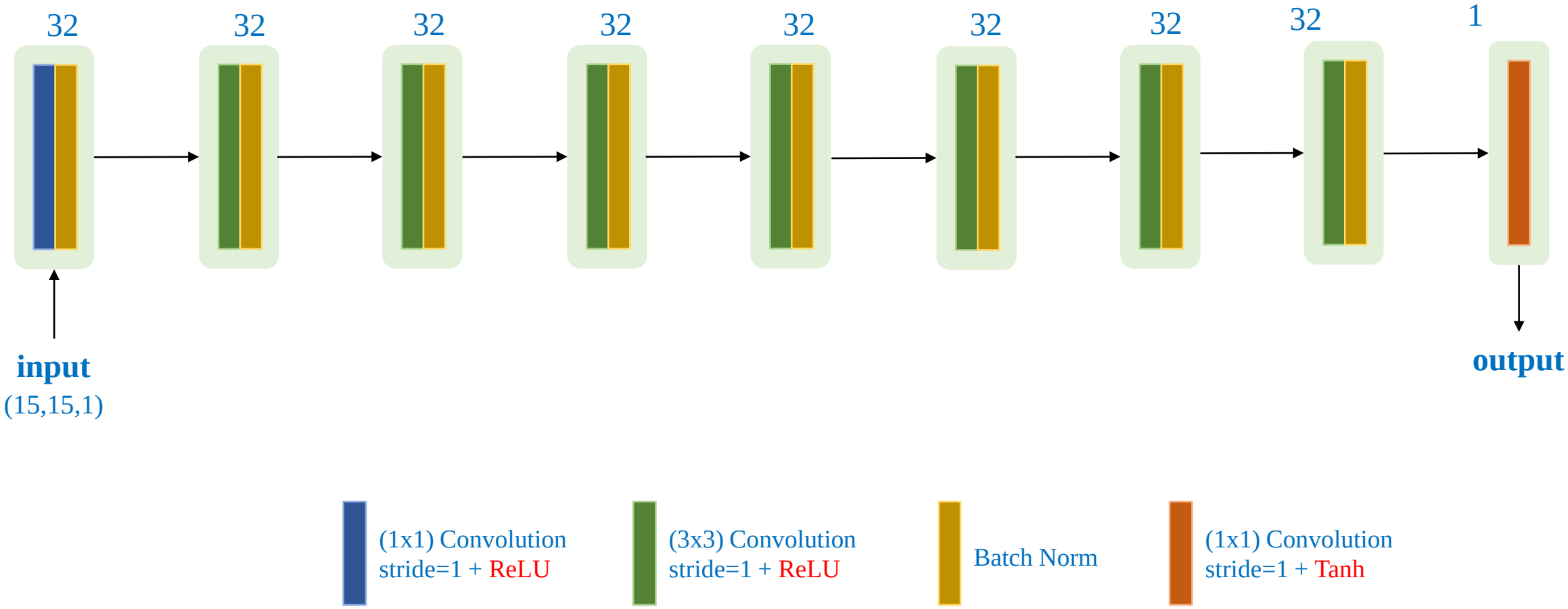
output

```
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

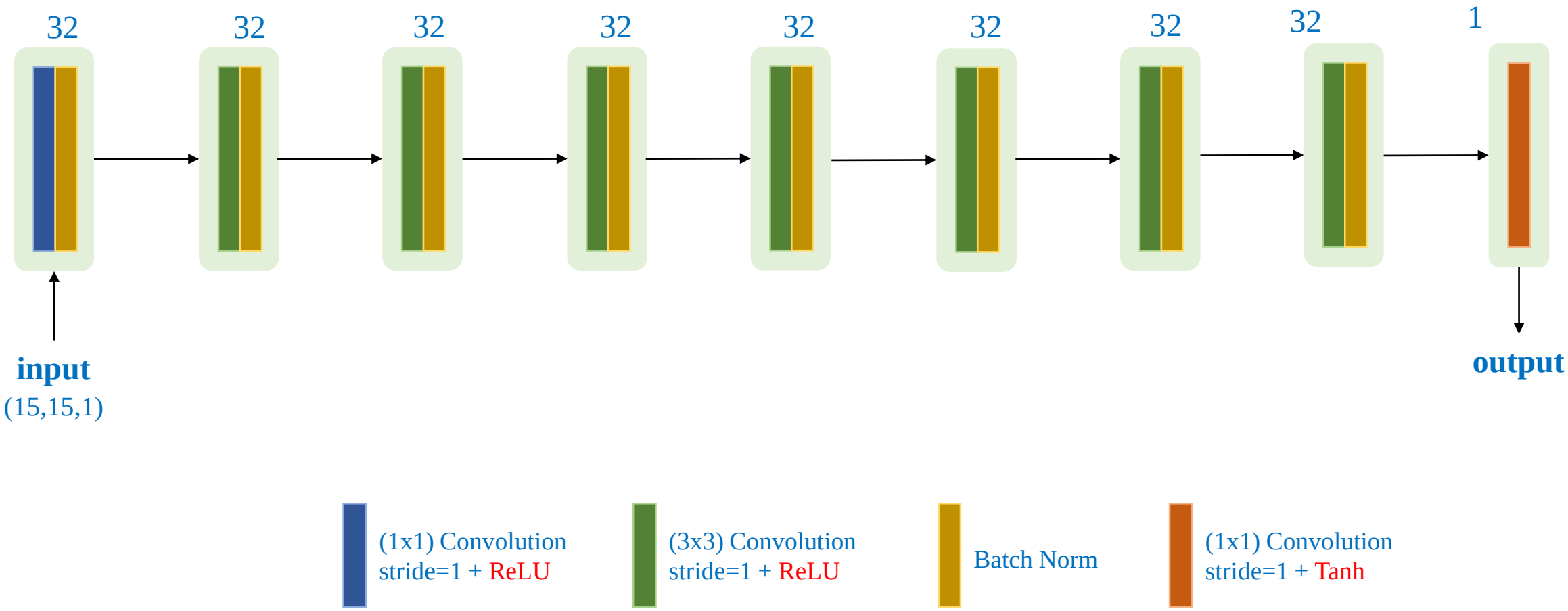
❖ Problem with shapes



Stereo Matching

❖ Implementation

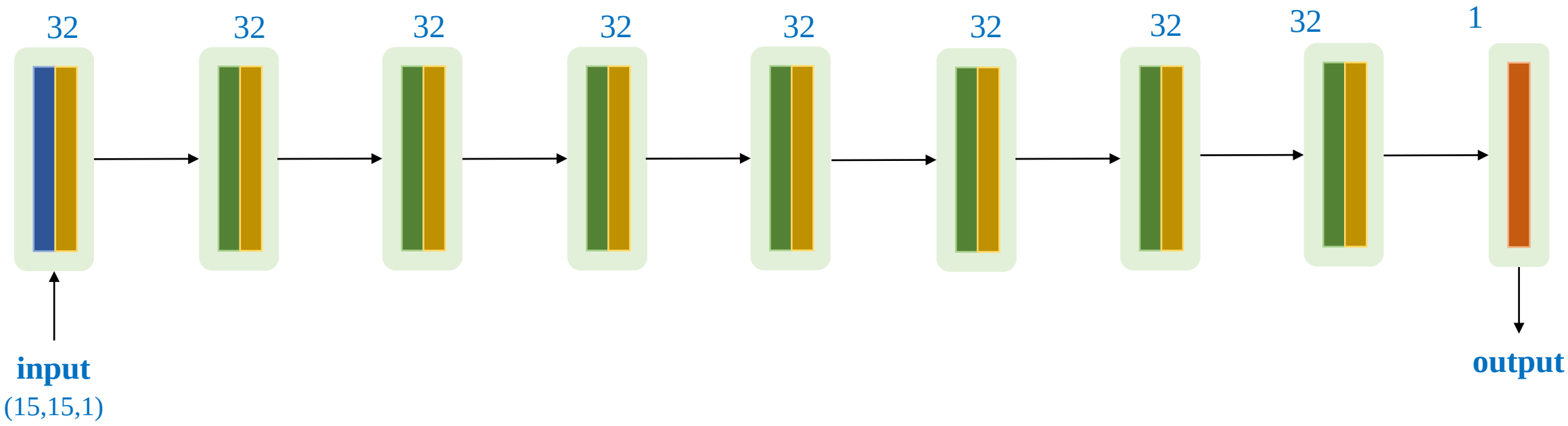
❖ Problem with fixed sizes



Stereo Matching

❖ Implementation

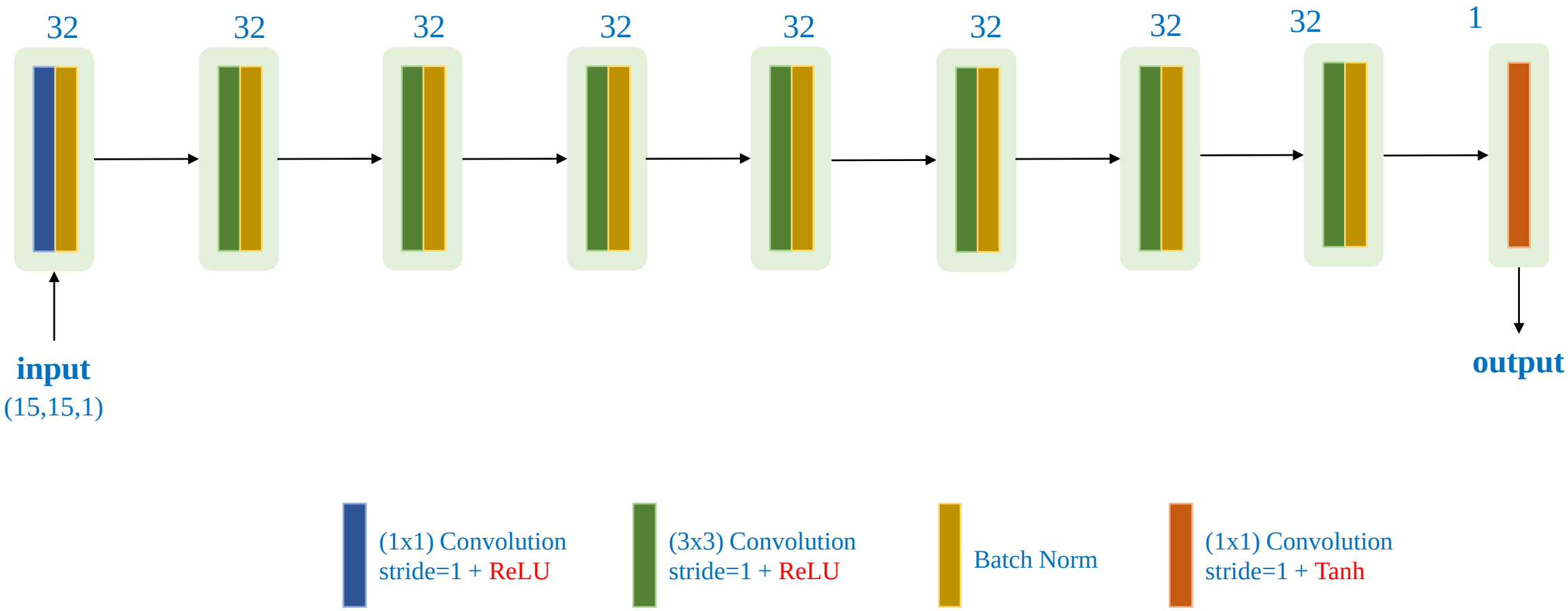
❖ Using the fit function



Stereo Matching

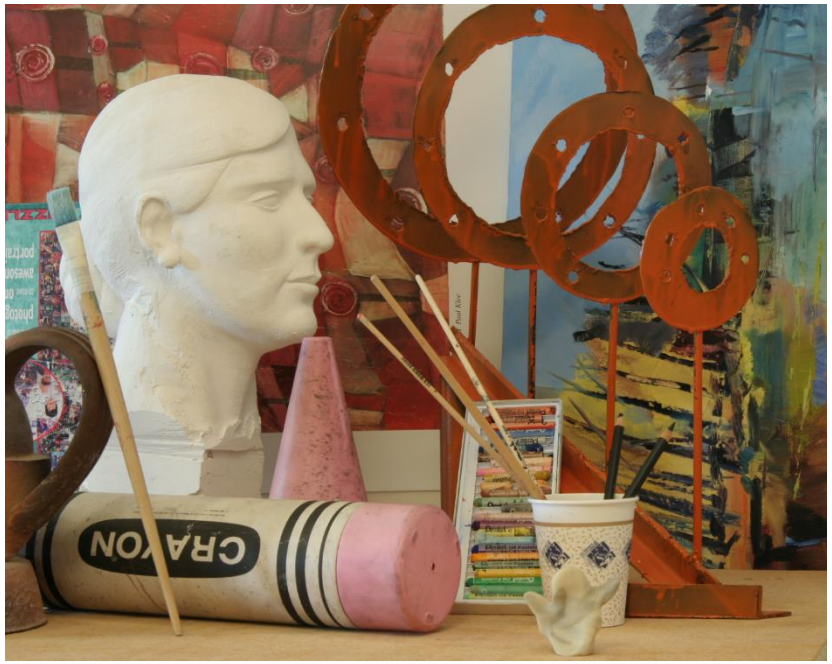
❖ Implementation

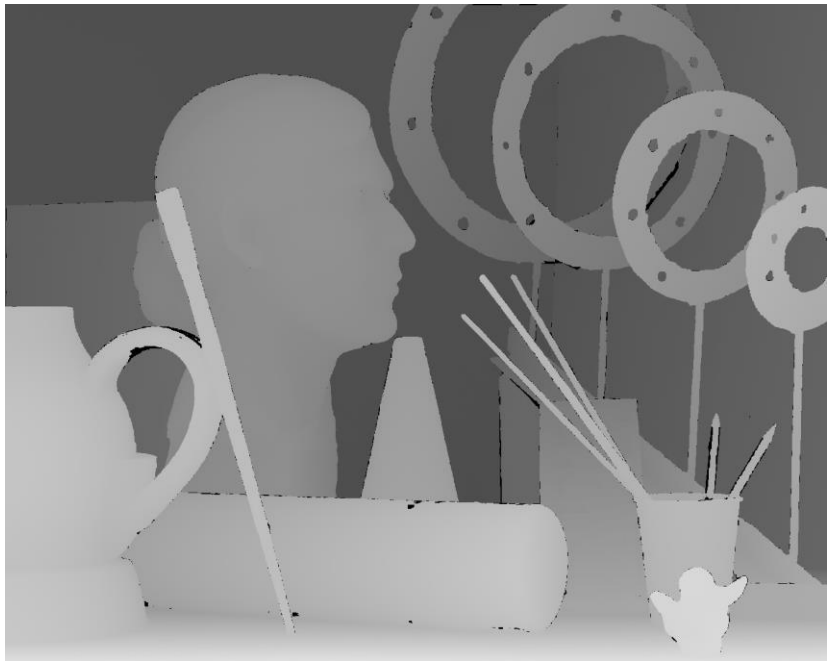
❖ Using GradientTape



Stereo Matching

❖ Dataset

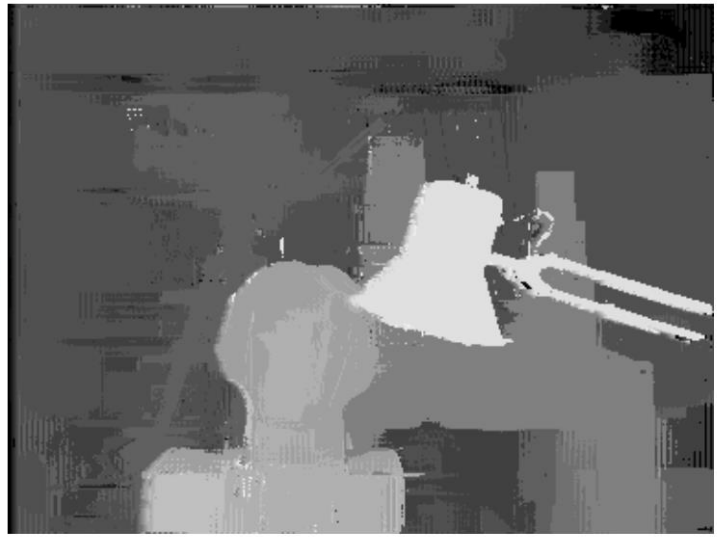


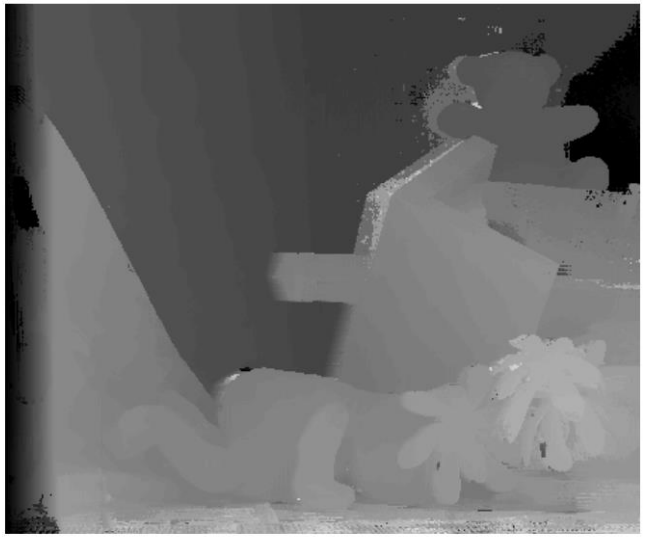


Stereo Matching

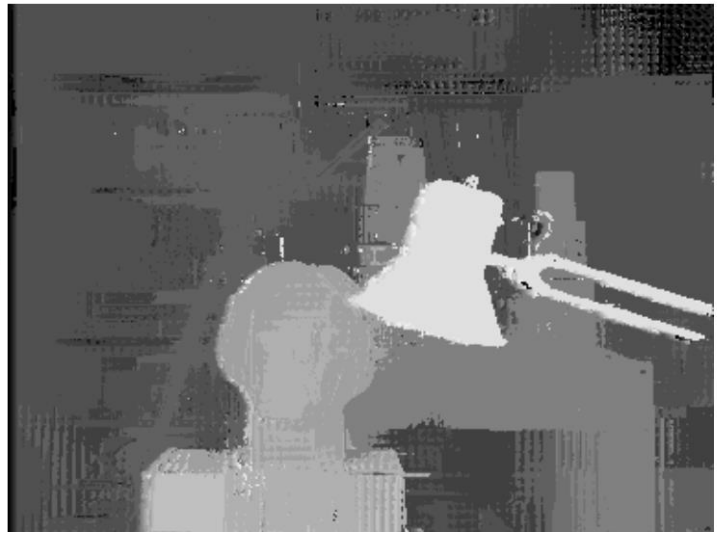
❖ Post-processing methods

w=3



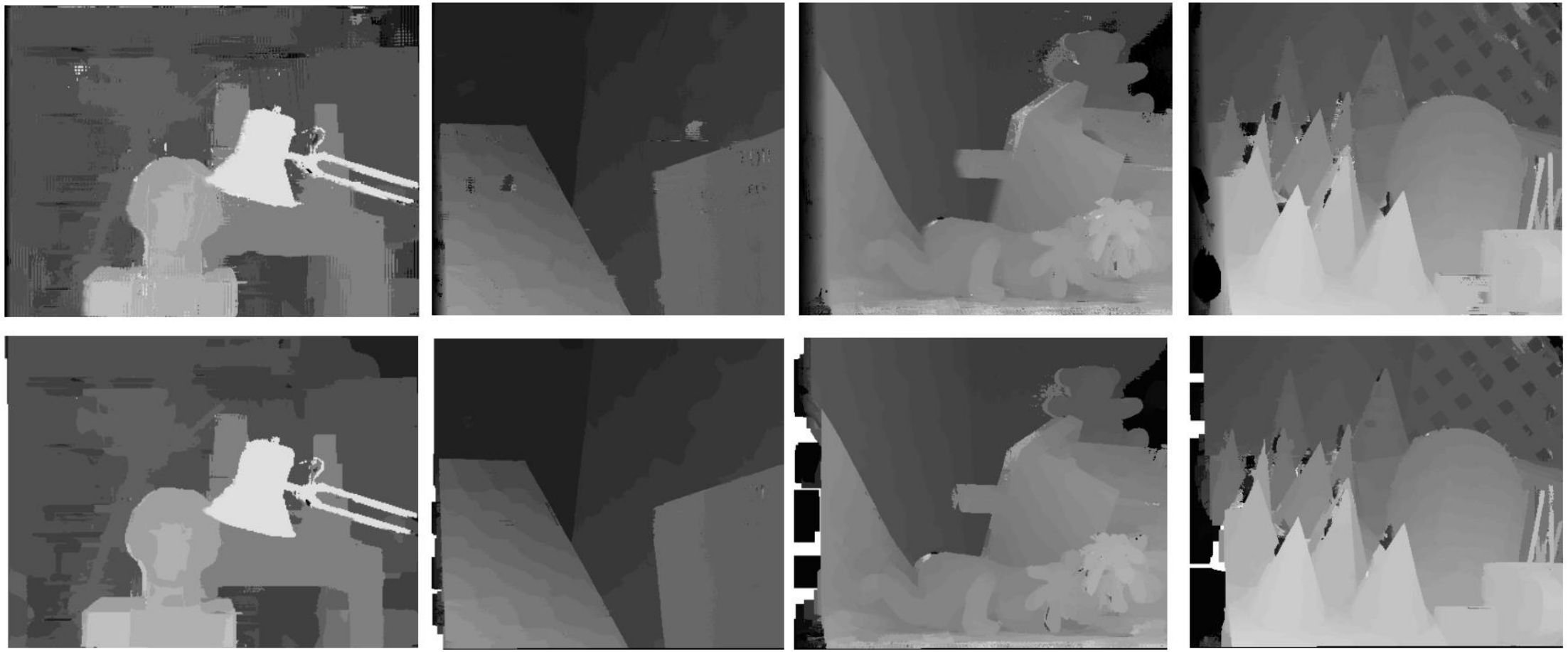


w=5



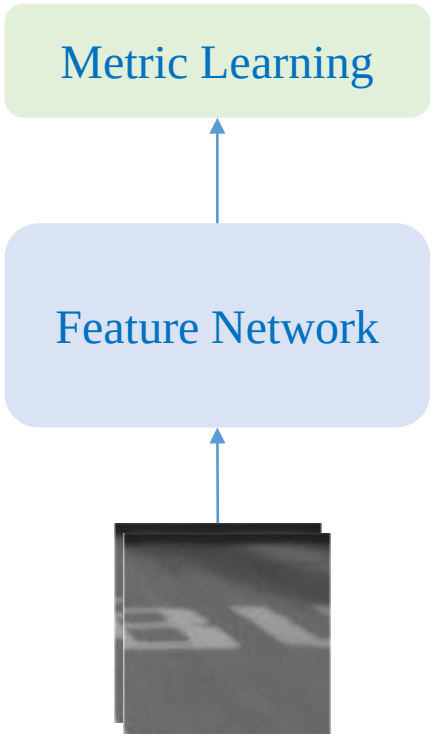
Stereo Matching

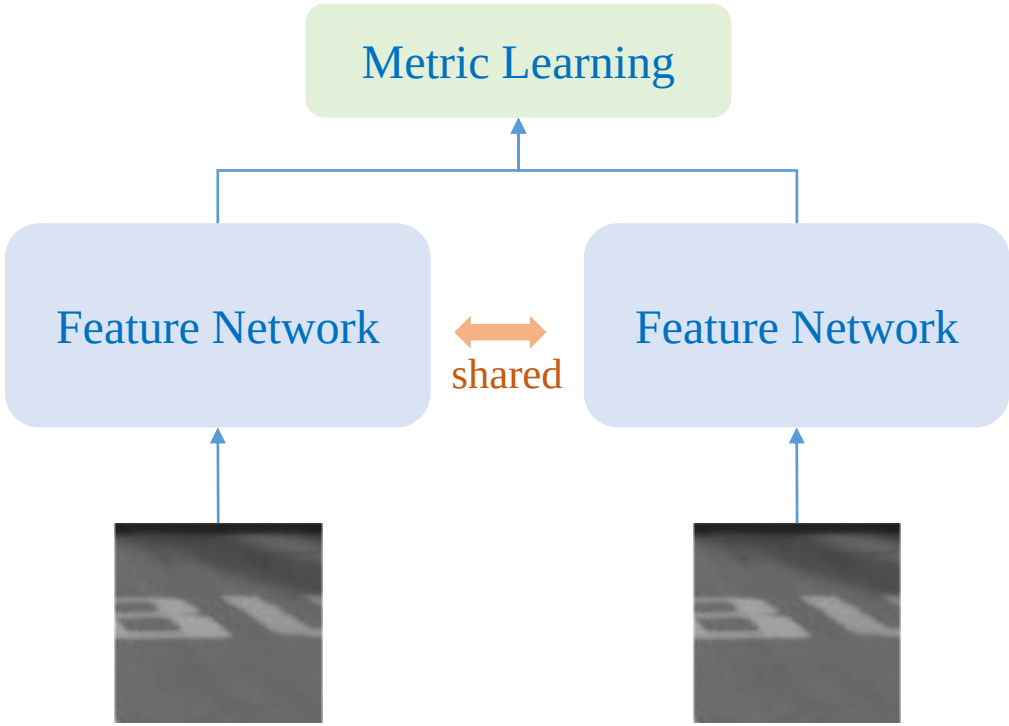
❖ Post-processing methods



Stereo Matching

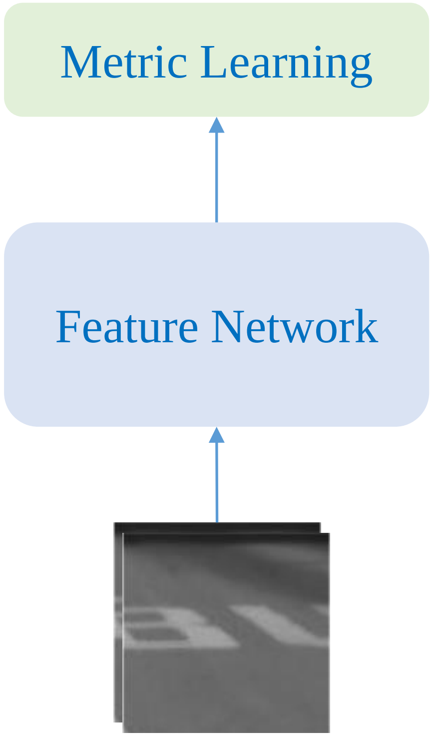
❖ Siamese Networks





Stereo Matching

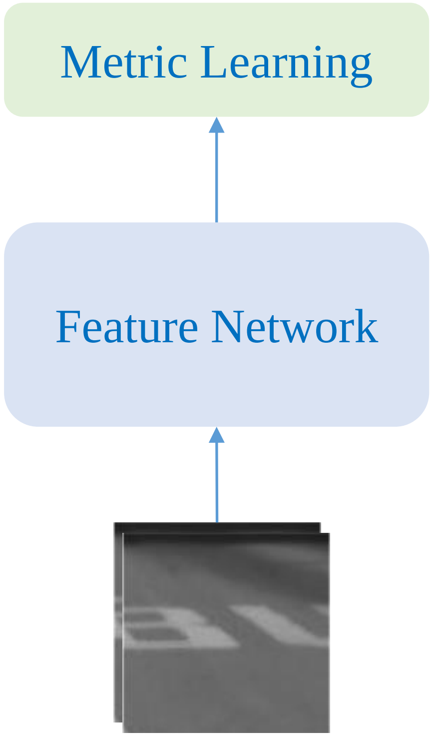
❖ Siamese Networks




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

def base_model(input_shape):
    inputs = Input(shape=input_shape)
    x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(inputs)
    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)
    x = tf.squeeze(x)
    model = Model(inputs=inputs, outputs=x)
    return model
```

Stereo Matching

❖ Siamese Networks

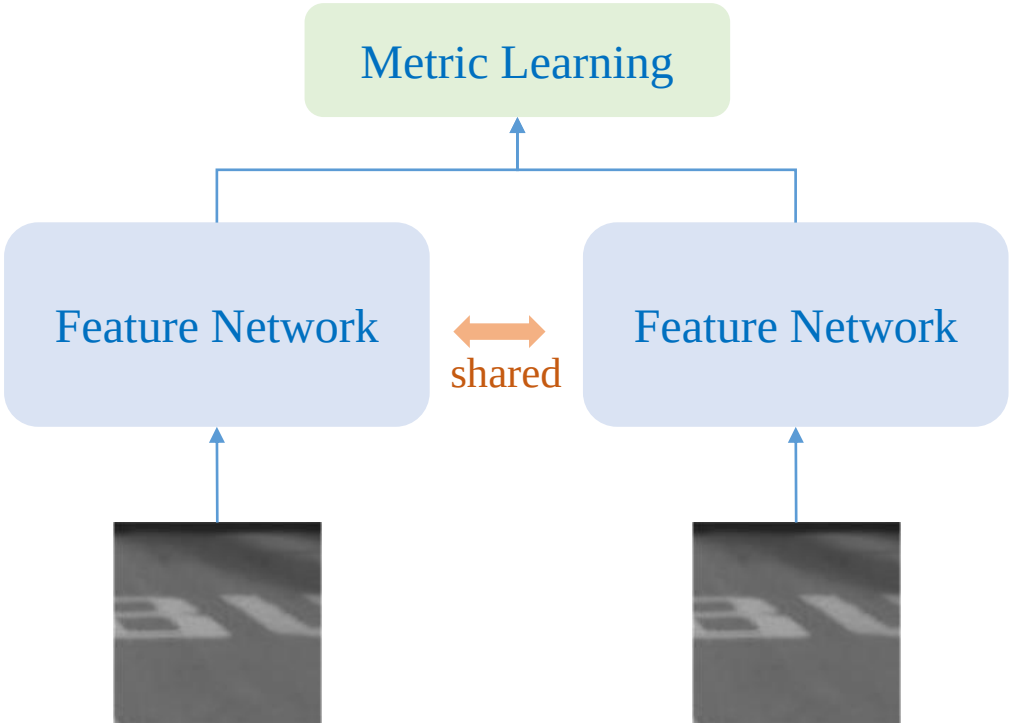



```
def base_model(input_shape):
    inputs = Input(shape=input_shape)

    # feature network
    x = Conv2D(32, (1,1),activation='relu', kernel_initializer='he_uniform')(inputs)
    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)

    # metric learning
    x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(x)
    x = BatchNormalization()(x)
    x = Conv2D(32, (1,1), activation='relu', kernel_initializer='he_uniform')(x)
    x = BatchNormalization()(x)
    x = Conv2D(1, (1,1), activation='tanh')(x)
    x = tf.squeeze(x)

    model = Model(inputs=inputs, outputs=x)
    return model
```

Year 2020

Stereo Matching

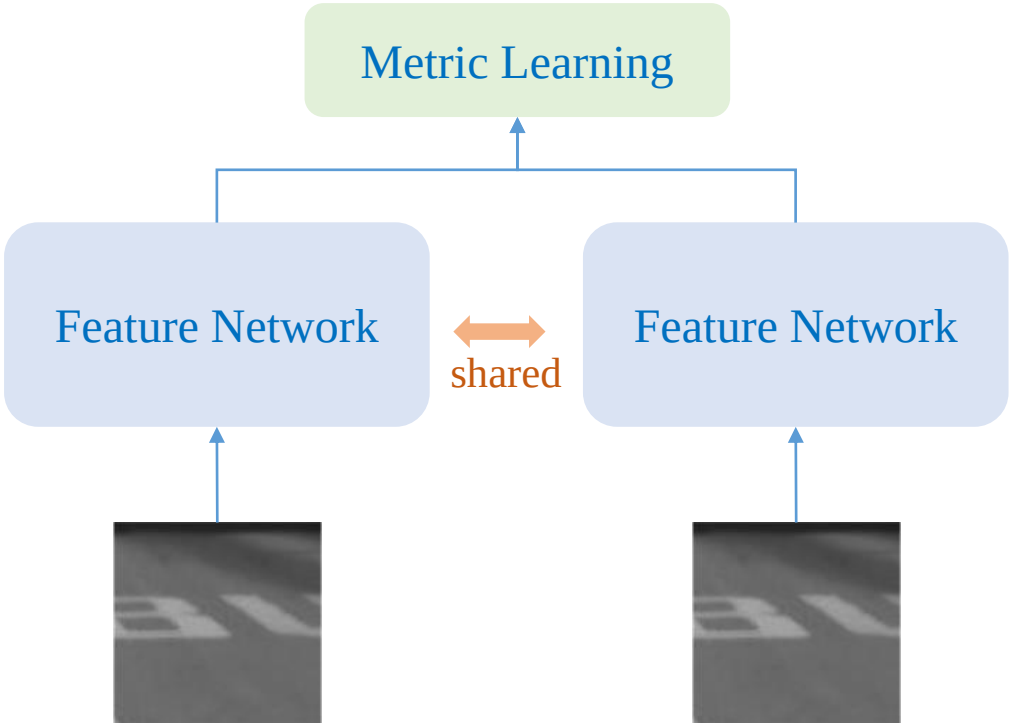
❖ Siamese Networks




```
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()
])
```

Stereo Matching

❖ Siamese Networks




```python
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
```


Thank you!