Super-resolution and Deblurring

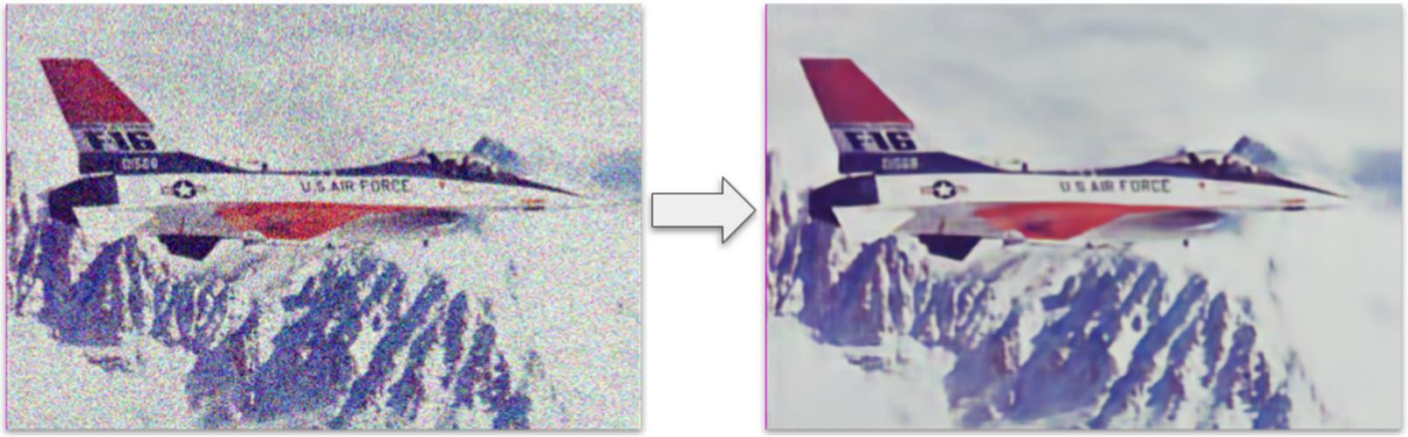
Prof. Seungchul Lee

Industrial AI Lab.

Image Restoration

- Image restoration
  - to recover original image from degraded one with prior knowledge of degradation process.
- The sources of corruption in digital images arise during image acquisition (digitization) and transmission.
  - Imaging sensors can be affected by ambient conditions.
  - Interference can be added to an image during transmission.

Image Restoration



Degraded image **Restored image**

Inverse Problem

- Inverse problems involve modeling of degradation and applying the inverse process in order to recover the original image from inadequate observations.
- The observations contain incomplete information about the target parameter or data due to physical limitations of the measurement devices.
- Consequently, solutions to inverse problems are non-unique.

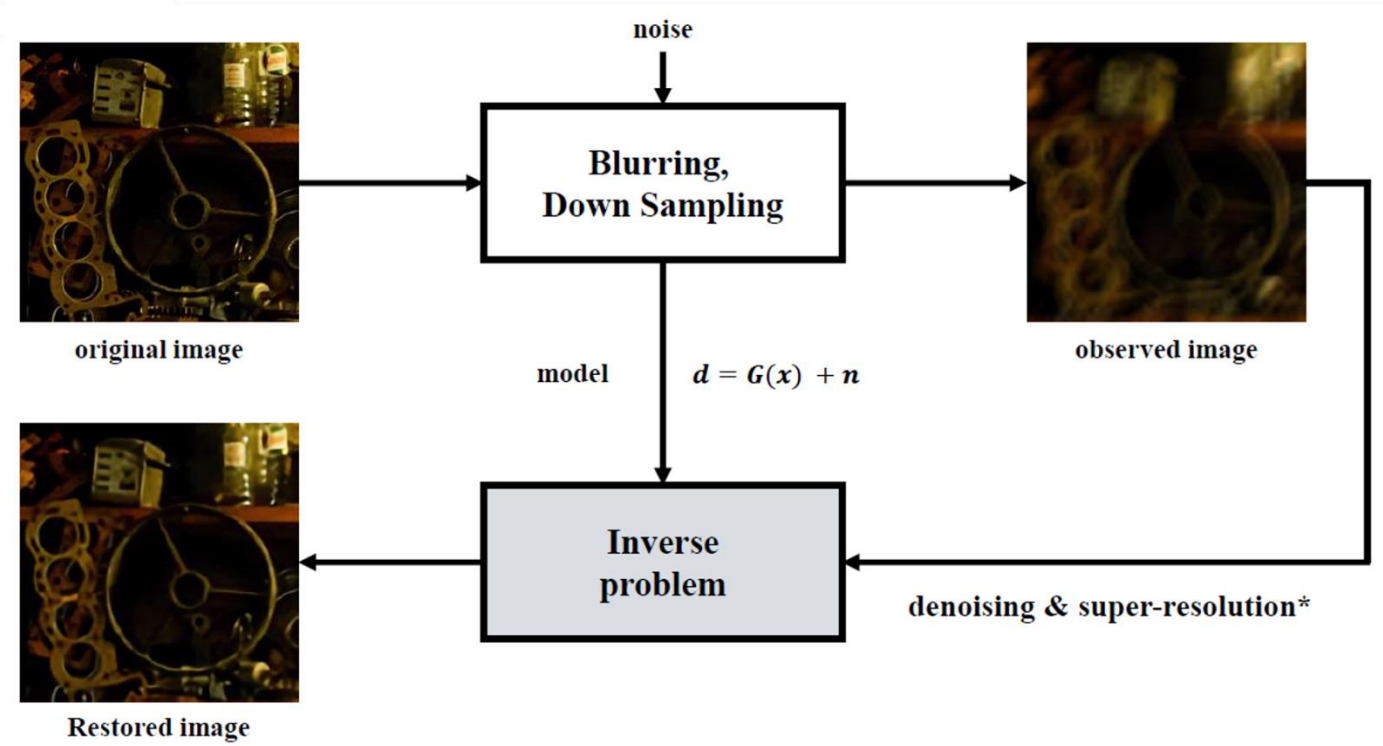
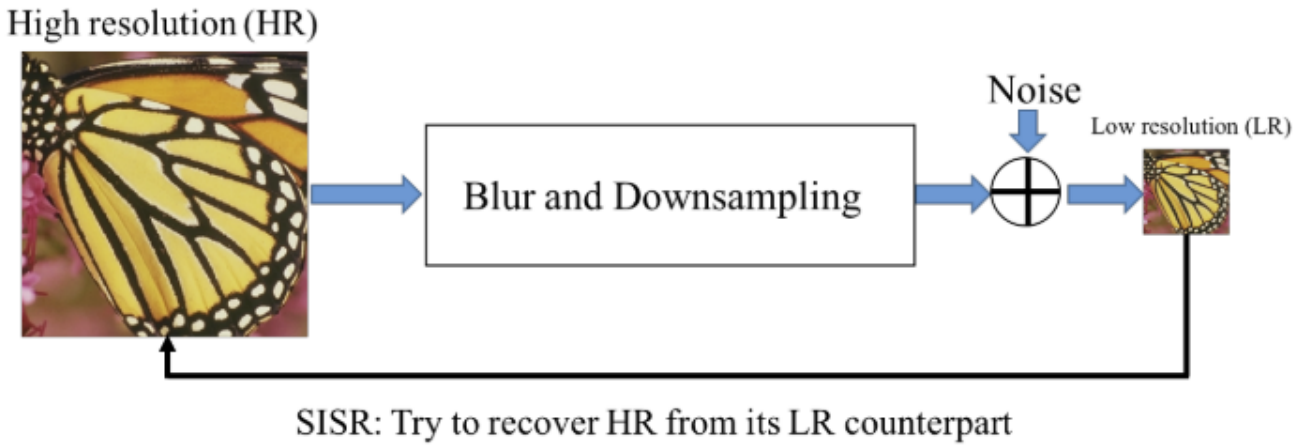
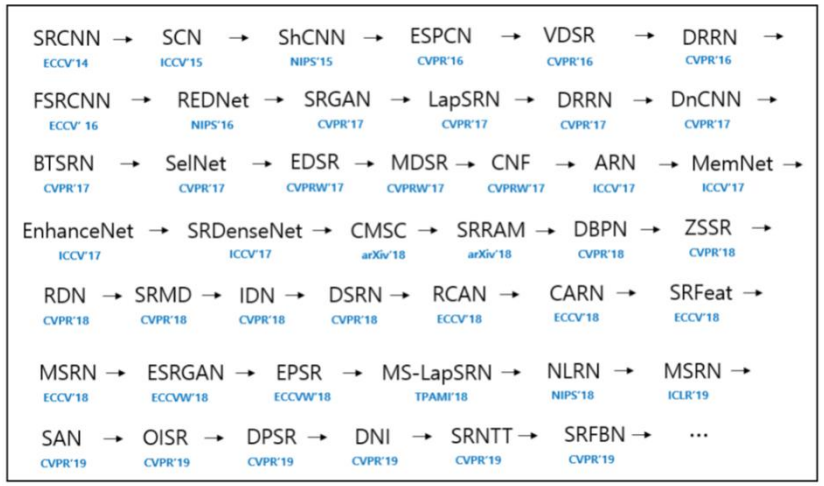


Image Super-resolution

- Restore High Resolution (HR) image from Low Resolution (LR) image

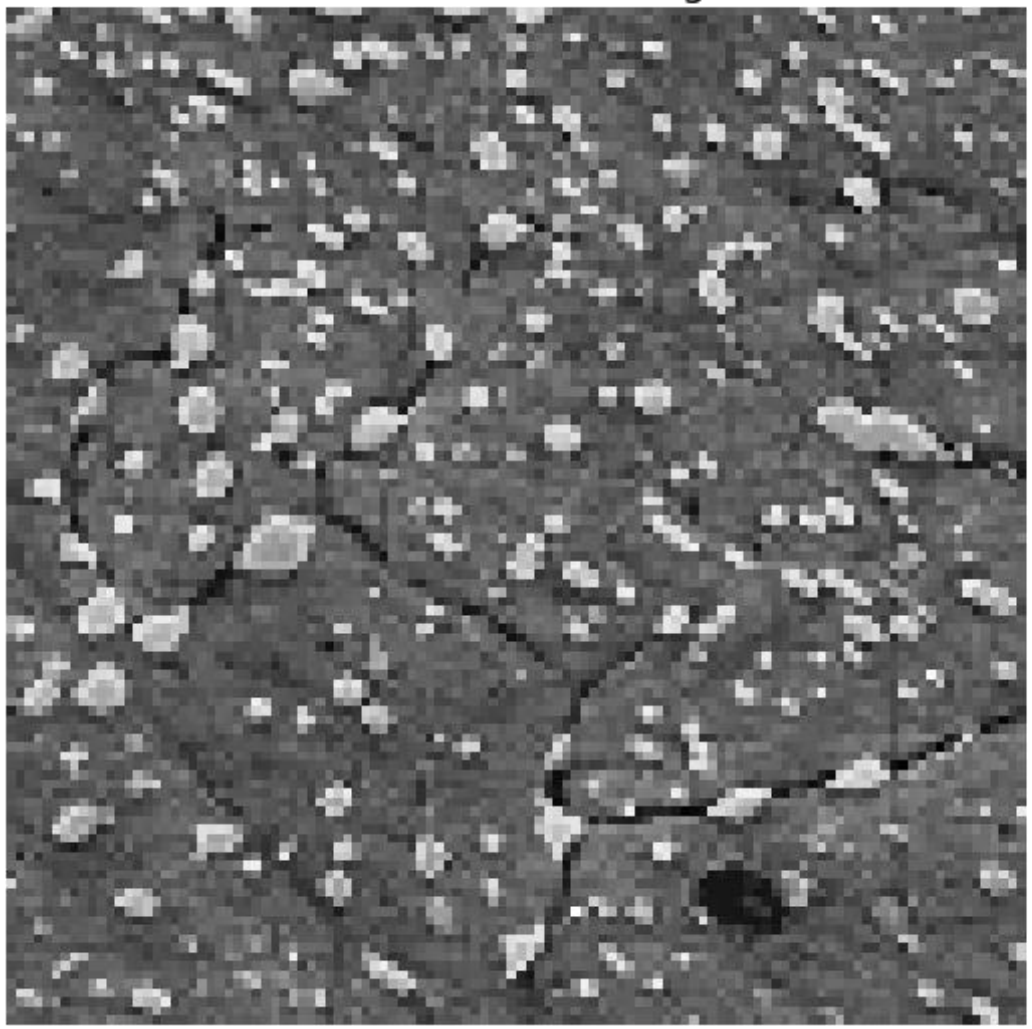


- Numerous learning-based SR approaches

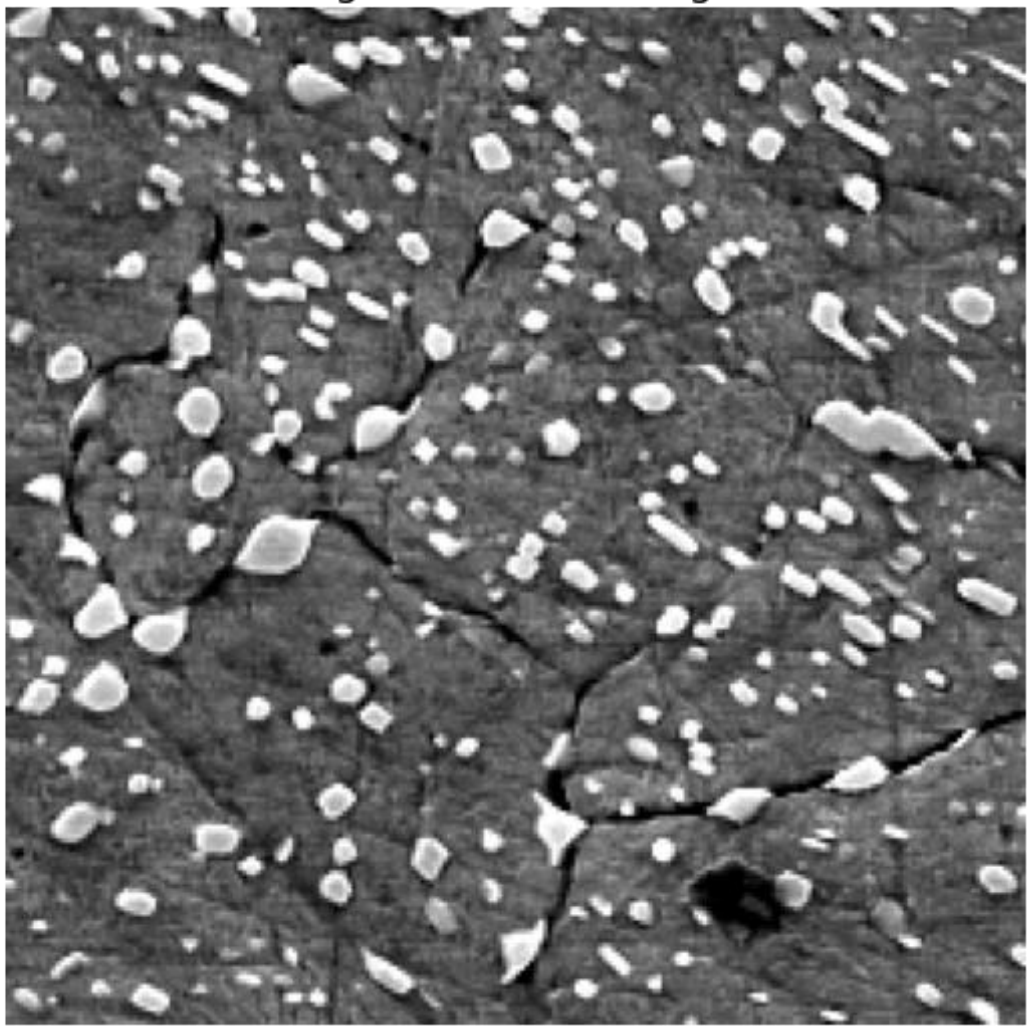


Lab: SR on Material Images

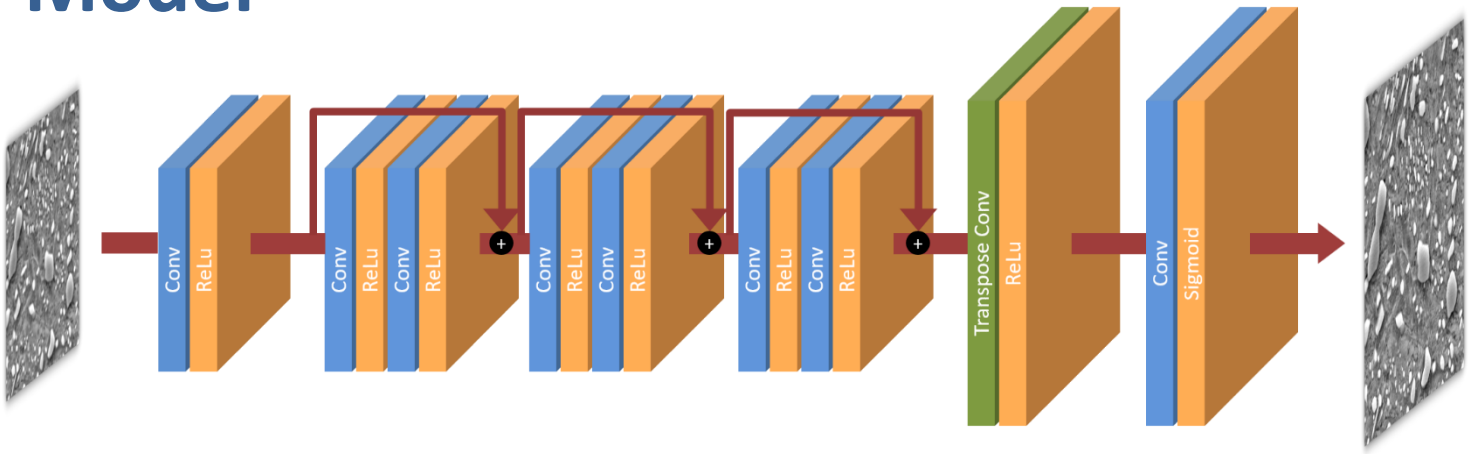
Low-resolution image



High-resolution image



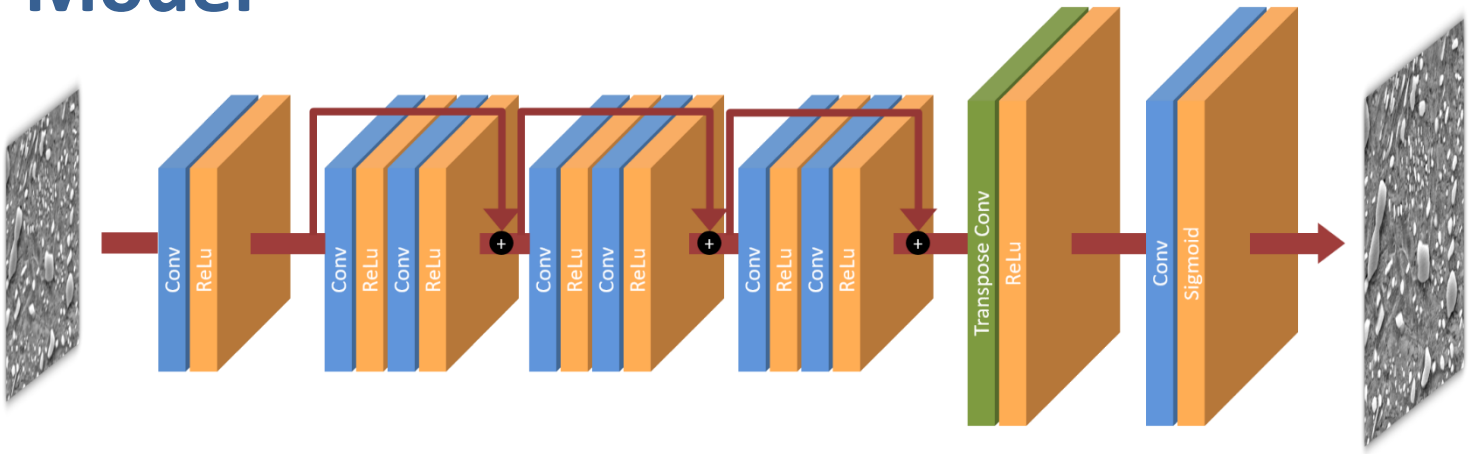
Build a FCN Model




```
inputs = tf.keras.Input(shape = (112, 112, 1))

# 3x3 convolutional layer
x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(inputs)
```

```
# first residual block
x_skip = x
x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Add()([x_skip, x])
```

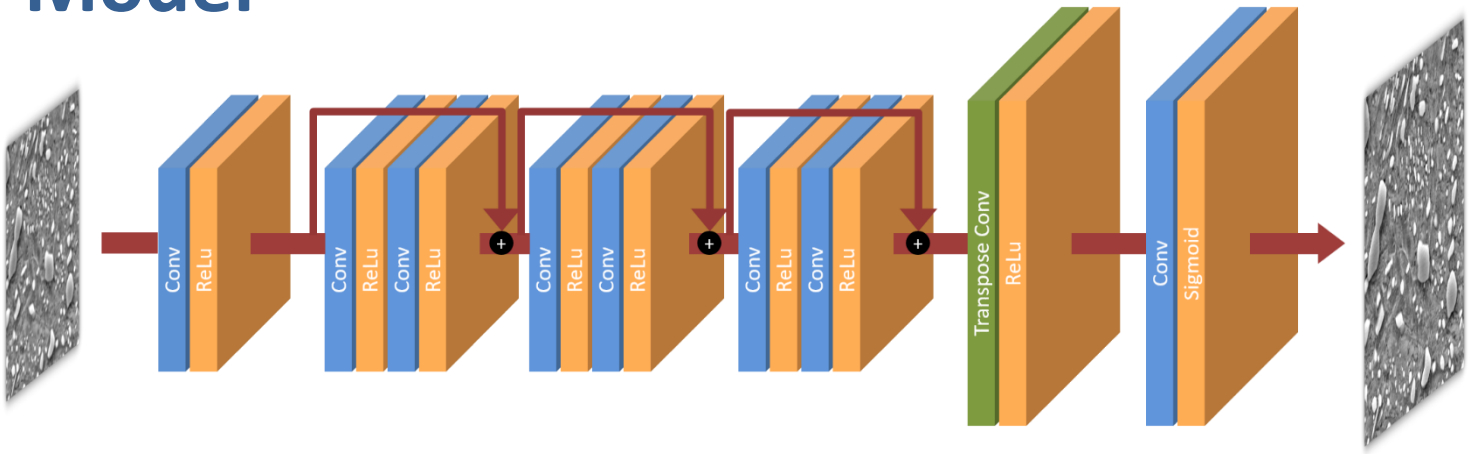
Build a FCN Model




```
# second residual block
x_skip = x
x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Add()([x_skip, x])
```

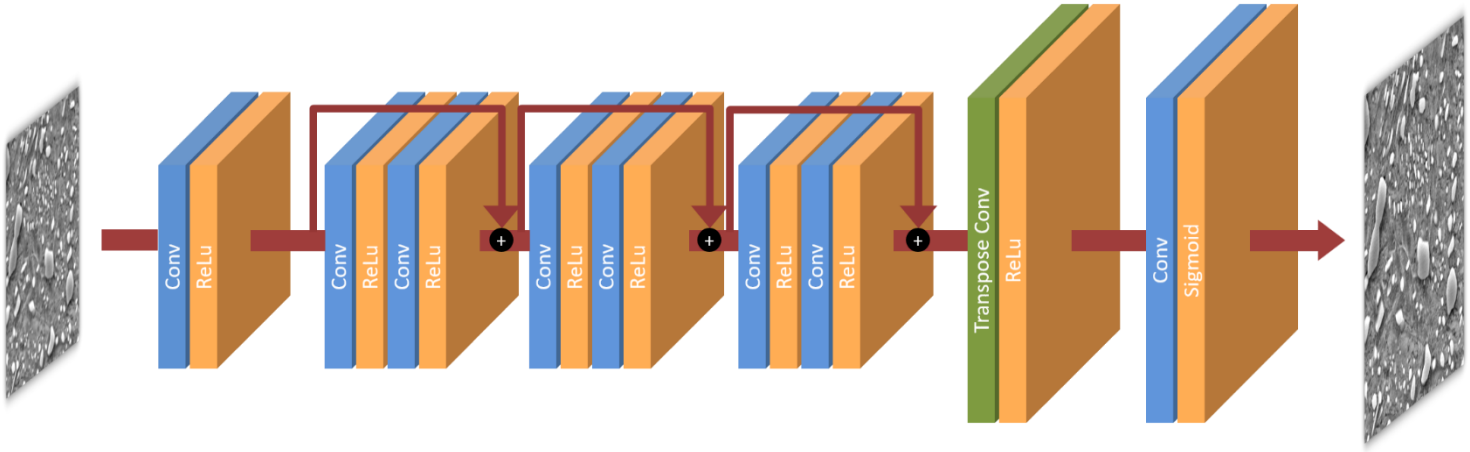
```
# third residual block
x_skip = x
x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Add()([x_skip, x])
```

Build a FCN Model




```
# upsampling layer
x = tf.keras.layers.Conv2DTranspose(
                    filters = 16,
                    kernel_size = (4,4),
                    strides = (2,2),
                    padding = 'SAME',
                    activation = 'relu')(x)

# 3x3 convolutional layer
outputs = tf.keras.layers.Conv2D(
                    filters = 1,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'sigmoid')(x)

model = tf.keras.Model(inputs, outputs)
```

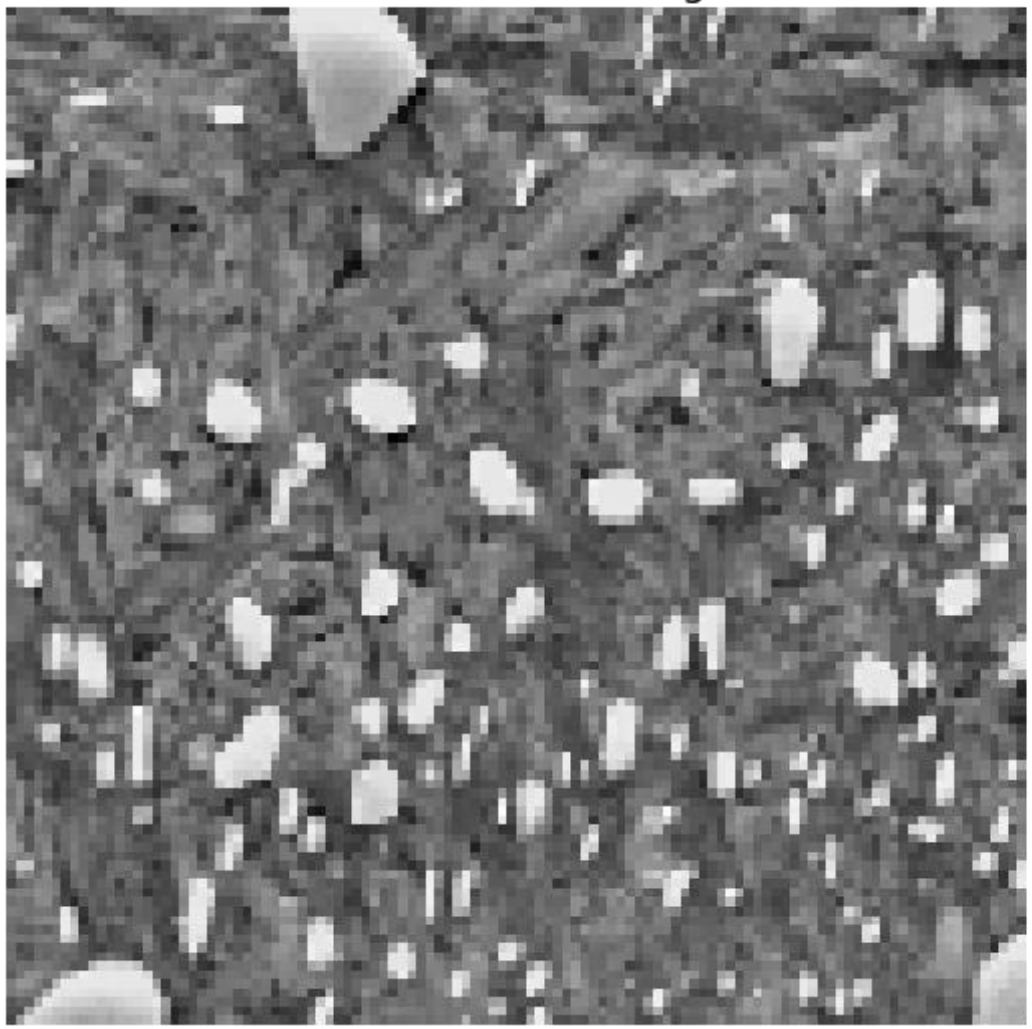
Training




```
model.compile(optimizer = 'adam',
              loss = 'mean_absolute_error',
              metrics = ['mean_squared_error'])
```

Result

Low-resolution image



Super-resolved image

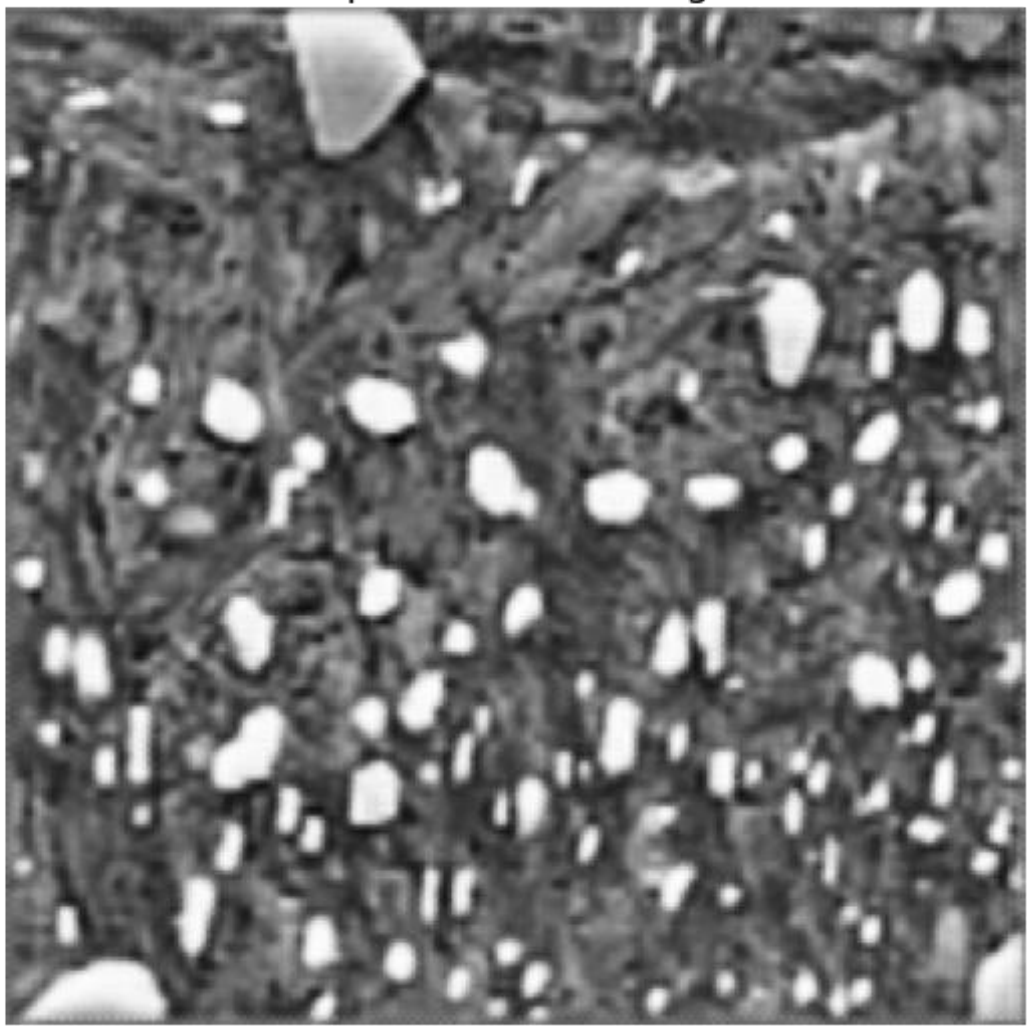
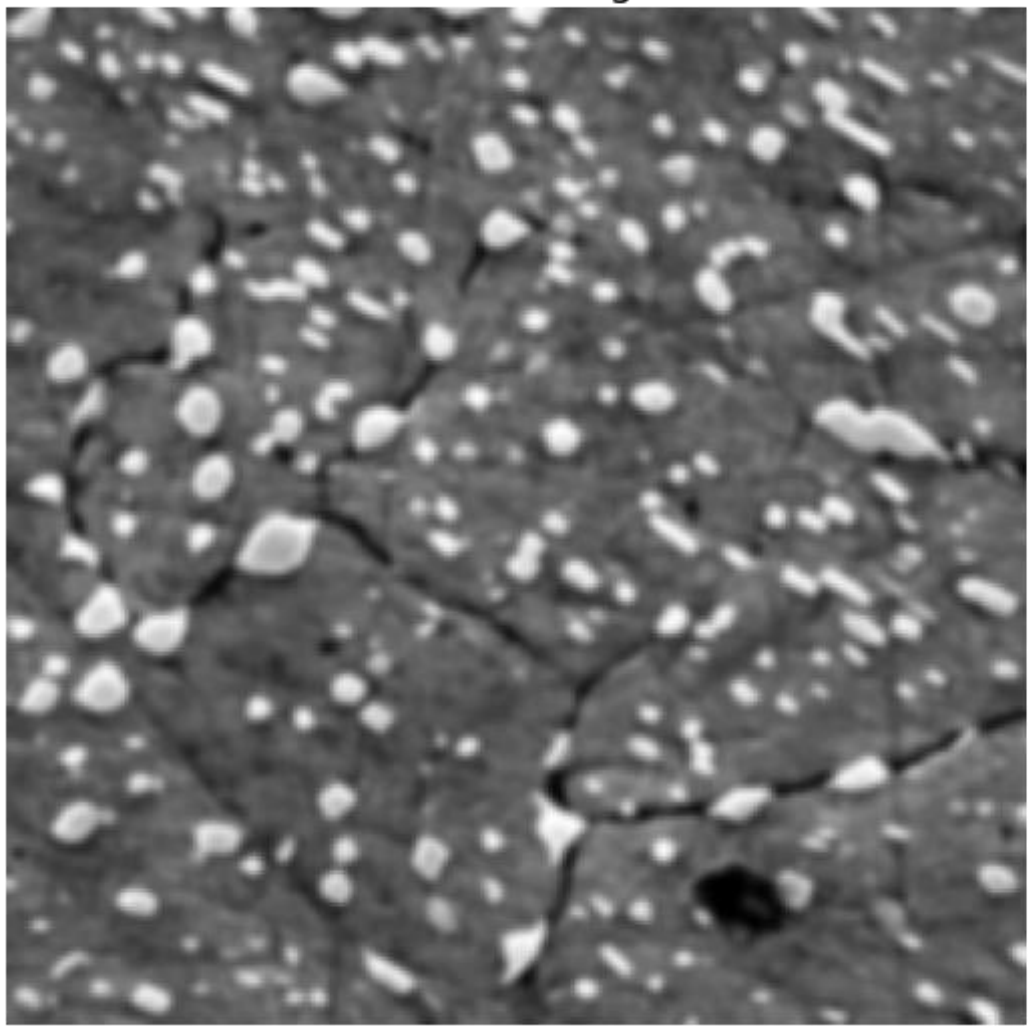
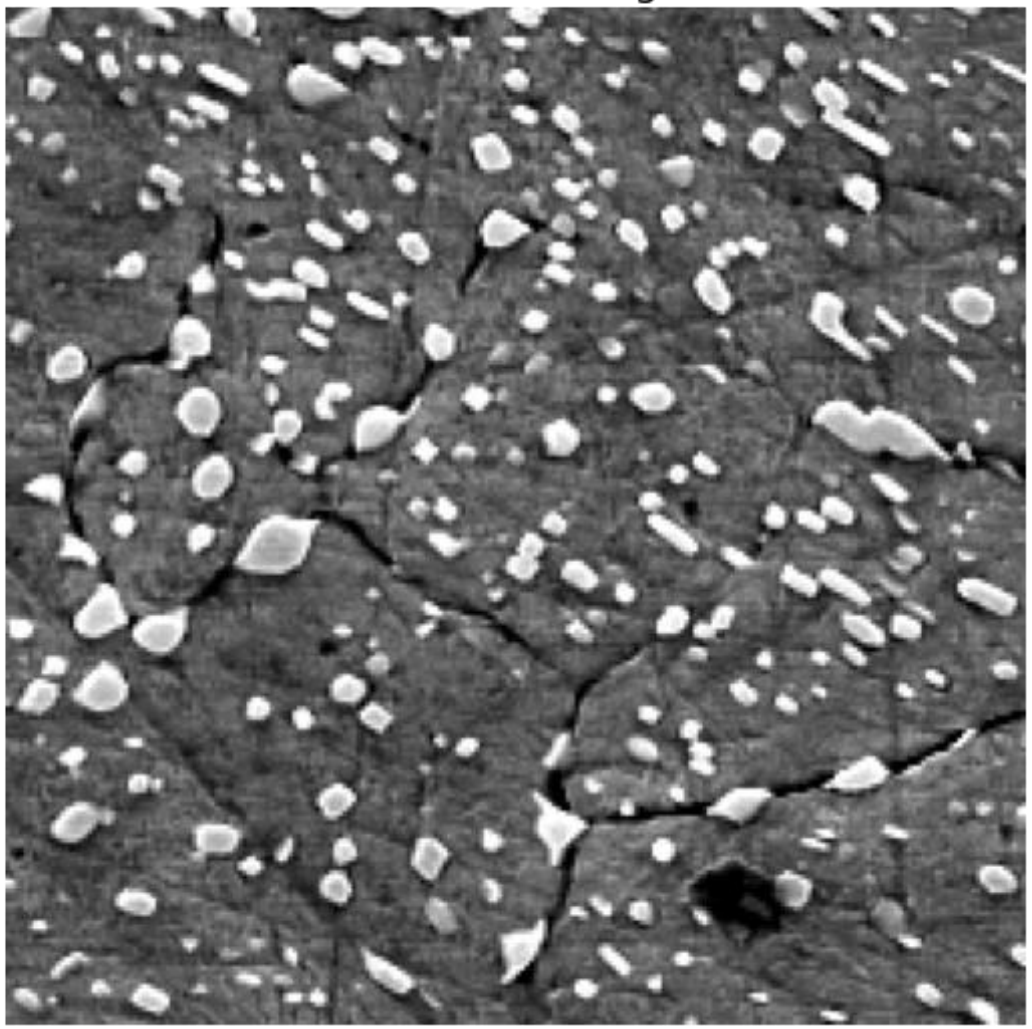


Image Deblurring

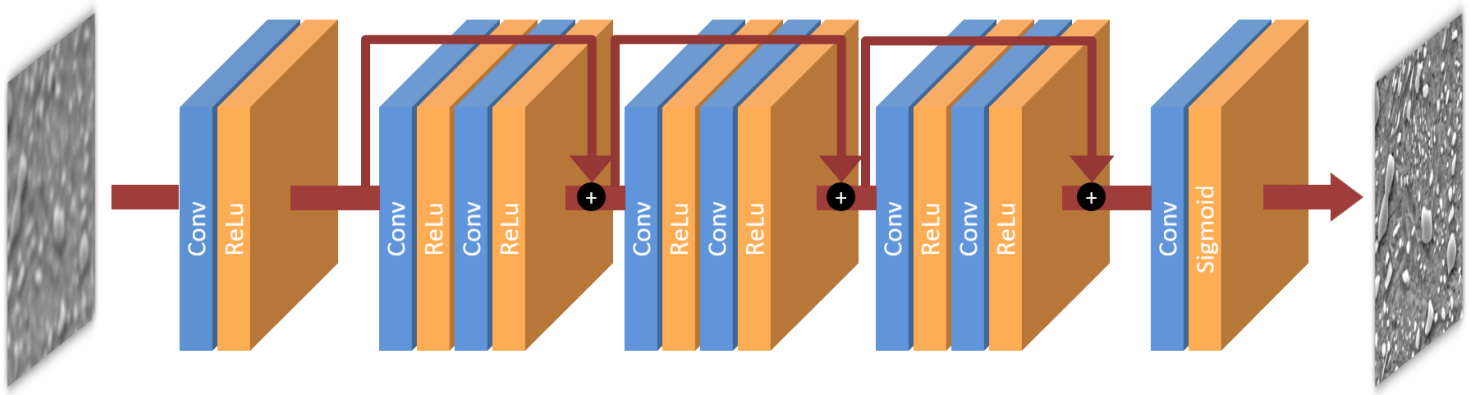
Blurred image



Deblurred image



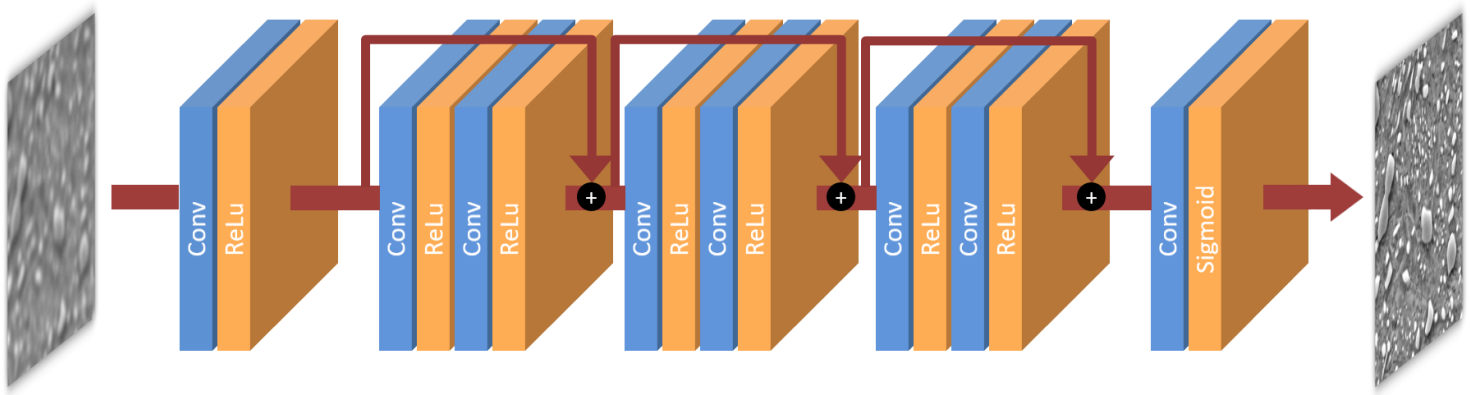
Build a FCN Model




```
inputs = tf.keras.Input(shape = (224, 224, 1))

# 3x3 convolutional layer
x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(inputs)
```

```
# first residual block
x_skip = x
x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Add()([x_skip, x])
```

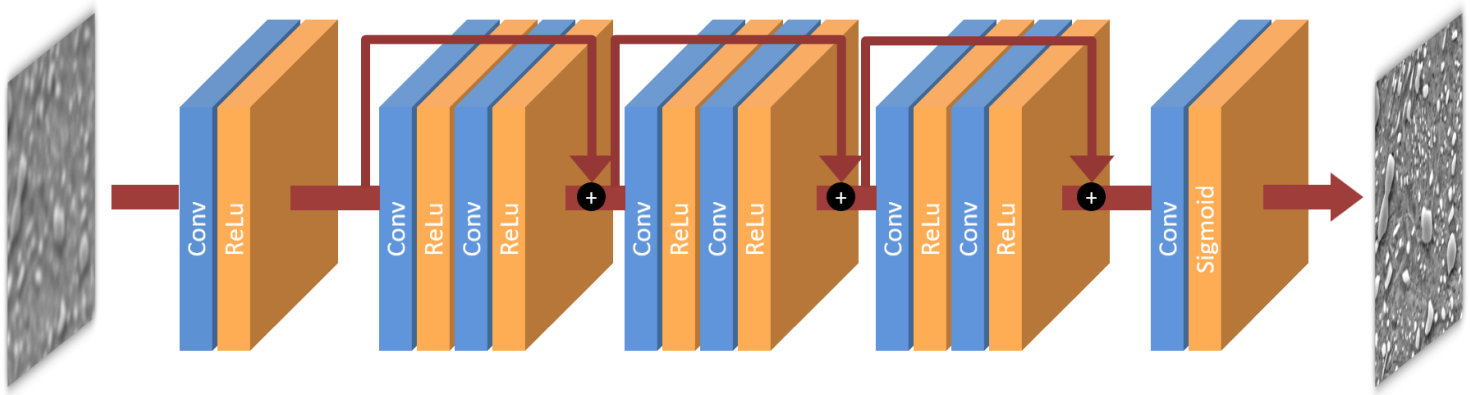
Build a FCN Model




```
# second residual block
x_skip = x
x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Add()([x_skip, x])
```

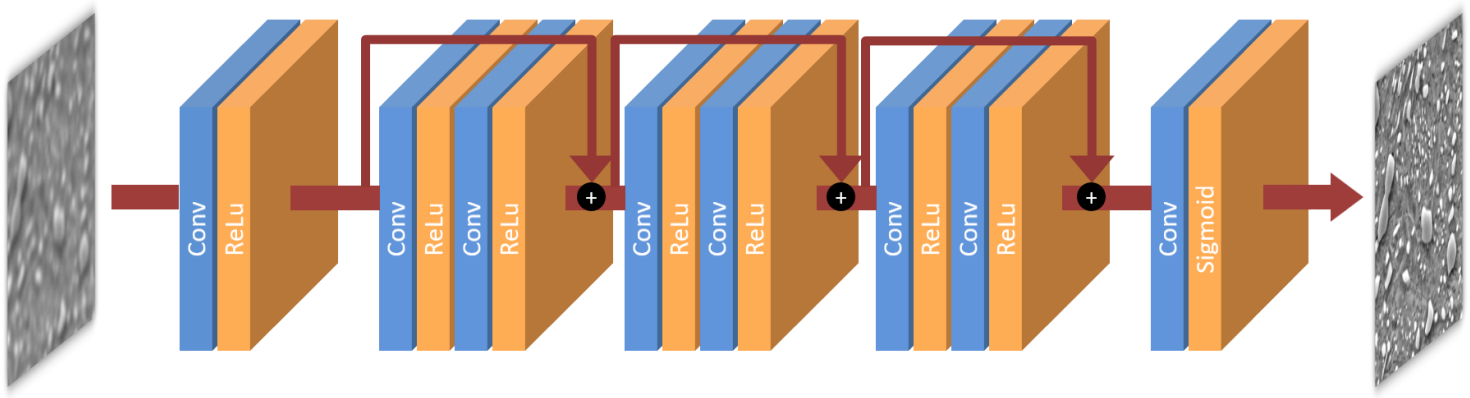
```
# third residual block
x_skip = x
x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Conv2D(
                    filters = 16,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'relu')(x)

x = tf.keras.layers.Add()([x_skip, x])
```

Build a FCN Model




```
# 3x3 convolutional layer
outputs = tf.keras.layers.Conv2D(
                    filters = 1,
                    kernel_size = (3,3),
                    padding = 'SAME',
                    activation = 'sigmoid')(x)

model = tf.keras.Model(inputs, outputs)
```

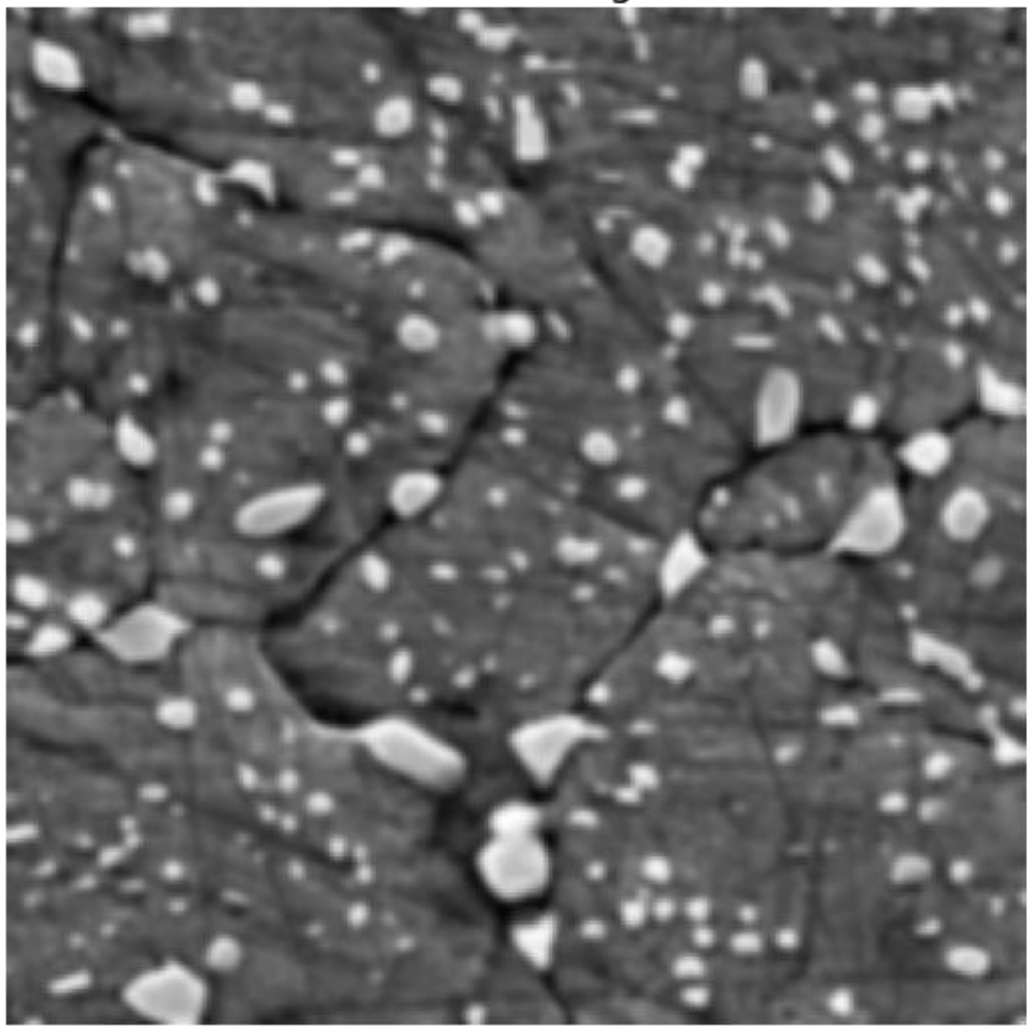
Training




```
model.compile(optimizer = 'adam',
              loss = 'mean_absolute_error',
              metrics = ['mean_squared_error'])
```

Result

Blurred image



Deblurred image