Deep learning and applications practice #5
- Image Restoration

2016025096 강수진

Code implementation

I implemented ‘Model1.ipynb’, ‘Model2.ipynb’, ‘Model3.ipynb’ file and I ran this file in google colab.

I set epochs_num=100, batch_sz=32.

```
epochs_num=100
batch_sz=32
```

I load image like the code below.

```
def load(filename):
   np_image = Image.open(filename)
   np_image = np.array(np_image).astype('float32')/255.0
   np_image = np.expand_dims(np_image, axis=0)
   print(np_image.shape)
   return np_image

img = load('./noisy.png')
print(img.shape)
```

filename is ‘noisy.png’ and I changed image to np array.
Since we have to consider batch size, I expand dimension of this array.

Each model has different model network and the code is like below.

Model1

```
inputs = layers.Input(shape=(None,None,3))

input_with_gaussian = layers.GaussianNoise(0.1)(inputs)

layer1 = layers.Conv2D(filters=64, kernel_size=3, padding='same',activation='relu',
                       bias_initializer='zeros')(input_with_gaussian)

layer2 = layers.Conv2D(filters=64, kernel_size=3, padding='same',activation='relu',
                       bias_initializer='zeros')(layer1)

layer3 = layers.Conv2D(filters=64, kernel_size=3, padding='same',activation='relu',
                       bias_initializer='zeros')(layer2)

layer4 = layers.Conv2D(filters=64, kernel_size=3, padding='same',activation='relu',
                       bias_initializer='zeros')(layer3)

predictions = layers.Conv2D(filters=3, kernel_size=3, padding='same',
                      bias_initializer='zeros')(layer4)

model = Model(inputs=inputs, outputs=predictions)

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

model.fit(x_train, y_train, epochs=epochs_num, batch_size=batch_sz)

model.evaluate(x_test, y_test, verbose=2)

result = model.predict(img)
```

Model2

```
inputs = layers.Input(shape=(None,None,3))
input_with_gaussian = layers.GaussianNoise(0.1)(inputs)

layer1 = layers.Conv2D(filters=64, kernel_size=3, padding='same',activation='relu')(input_with_gaussian)

layer2 = layers.Conv2D(filters=64, kernel_size=3, padding='same',activation='relu')(layer1)

layer3 = layers.Conv2D(filters=64, kernel_size=3, padding='same',activation='relu')(layer2)

layer4 = layers.Conv2D(filters=64, kernel_size=3, padding='same',activation='relu')(layer3)

layer5 = layers.Conv2D(filters=3, kernel_size=3, padding='same')(layer4)

predictions = layers.Add()([layer5, input_with_gaussian])

model = Model(inputs=inputs, outputs=predictions)

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

model.fit(x_train, y_train, epochs=epochs_num, batch_size=batch_sz)

model.evaluate(x_test, y_test, verbose=2)

result = model.predict(img)
```

Model3

```
inputs = layers.Input(shape=(None,None,3))
input_with_gaussian = layers.GaussianNoise(0.1)(inputs)

layer1 = layers.Conv2D(filters=64, kernel_size=3, padding='same')(input_with_gaussian)
layer1 = layers.BatchNormalization()(layer1)
layer1 = layers.ReLU()(layer1)

layer2 = layers.Conv2D(filters=64, kernel_size=3, padding='same')(layer1)
layer2 = layers.BatchNormalization()(layer2)
layer2 = layers.ReLU()(layer2)

layer3 = layers.Conv2D(filters=64, kernel_size=3, padding='same')(layer2)
layer3 = layers.BatchNormalization()(layer3)
layer3 = layers.ReLU()(layer3)

layer4 = layers.Conv2D(filters=64, kernel_size=3, padding='same')(layer3)
layer4 = layers.BatchNormalization()(layer4)
layer4 = layers.ReLU()(layer4)

layer5 = layers.Conv2D(filters=3, kernel_size=3, padding='same')(layer4)
predictions = layers.Add()([layer5, input_with_gaussian])
```

```
model = Model(inputs=inputs, outputs=predictions)

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

model.fit(x_train, y_train, epochs=epochs_num, batch_size=batch_sz)

model.evaluate(x_test, y_test, verbose=2)

result = model.predict(img)
```

To save an image, I used the code below.
Since the dimension of ‘result’ was 4, I used reshape to change dimension.

```
result = result.reshape(512,512,3)
result *= 255.0
result = Image.fromarray(result.astype(np.uint8), 'RGB')
result.save('Model1.png')
result
```

Results

noisy.png

model1.png

model2.png

model3.png

I could get rid of the noise of the image.