copy from https://blog.naver.com/ckdgus1433/221443838135

```
%tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

```
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

# MNIST 로딩 (라벨은 필요없기 때문에 버림)
(x_train, _), (x_test, _) = mnist.load_data()

# 데이터 정규화 및 Reshape
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = np.reshape(x_train, (len(x_train), 784))
x_test = np.reshape(x_test, (len(x_test), 784))

# 원본데이터에 Noise 추가
noise_factor = 0.5
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_test.shape)
x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)

# Noise가 추가된 데이터 확인
n = 10
plt.figure(figsize=(20, 2))
for i in range(1,n):
    ax = plt.subplot(1, n, i)
    plt.imshow(x_test_noisy[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()

# 모형 구성
model = Sequential()
model.add(Dense(128, activation='relu', input_dim=784))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(784, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy')

# 모형 학습
model.fit(x_train_noisy, x_train,
          nb_epoch=100,
```

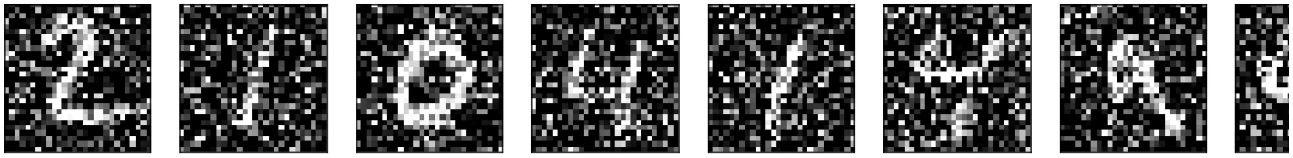
```
            batch_size=256,
            shuffle=True,
            validation_data=(x_test_noisy, x_test))

# 결과 확인
decoded_imgs = model.predict(x_test)
n = 10
plt.figure(figsize=(20, 6))
for i in range(1, n):
    # display original
    ax = plt.subplot(3, n, i)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # display noisy
    ax = plt.subplot(3, n, i + n)
    plt.imshow(x_test_noisy[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # display reconstruction
    ax = plt.subplot(3, n, i + 2*n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```



```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resource_vari
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/nn_impl.py:18
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:The `nb_epoch` argument in `fit` has been renamed `epochs`.
Train on 60000 samples, validate on 10000 samples
Epoch 1/100
60000/60000 [==============================] - 10s 168us/sample - loss: 0.2566 - val_loss: 0.
Epoch 2/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1776 - val_loss: 0.16
Epoch 3/100
60000/60000 [==============================] - 1s 25us/sample - loss: 0.1594 - val_loss: 0.15
Epoch 4/100
60000/60000 [==============================] - 1s 25us/sample - loss: 0.1510 - val_loss: 0.14
Epoch 5/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1451 - val_loss: 0.14
Epoch 6/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1411 - val_loss: 0.13
Epoch 7/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1381 - val_loss: 0.13
Epoch 8/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1356 - val_loss: 0.13
Epoch 9/100
60000/60000 [==============================] - 1s 25us/sample - loss: 0.1334 - val_loss: 0.13
Epoch 10/100
60000/60000 [==============================] - 1s 25us/sample - loss: 0.1317 - val_loss: 0.13
Epoch 11/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1303 - val_loss: 0.12
Epoch 12/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1290 - val_loss: 0.12
Epoch 13/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1279 - val_loss: 0.12
Epoch 14/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1270 - val_loss: 0.12
Epoch 15/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1261 - val_loss: 0.12
Epoch 16/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1252 - val_loss: 0.12
Epoch 17/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1244 - val_loss: 0.12
Epoch 18/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1235 - val_loss: 0.12
Epoch 19/100
60000/60000 [==============================] - 1s 25us/sample - loss: 0.1227 - val_loss: 0.12
Epoch 20/100
60000/60000 [==============================] - 1s 25us/sample - loss: 0.1221 - val_loss: 0.12
Epoch 21/100
60000/60000 [==============================] - 1s 24us/sample - loss: 0.1215 - val_loss: 0.12
Epoch 22/100
```