Load data:

```
# import data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# scale data
X_train = X_train / 255
X_test = X_test / 255
```

Generate noisy data:

```
# generate noisy data
X_train_noisy = np.clip(X_train
    + np.random.normal(loc=0.0, scale=0.5, size=X_train.shape), 0., 1.)
X_test_noisy = np.clip(X_test
    + np.random.normal(loc=0.0, scale=0.5, size=X_test.shape), 0., 1.)
```

Build convolutional autoencoder--undercomplete to extract the most useful features. The autoencoder reconstructs noisy images as clean images:

```
# build convolutional autoencoder
conv_encoder = models.Sequential([
    layers.Reshape([28, 28, 1], input_shape=[28, 28]),
    layers.Conv2D(16, (3, 3), padding='same', activation='selu'),
    layers.MaxPool2D((2, 2)),
    layers.Conv2D(8, (3, 3), padding='same', activation='selu'),
    layers.MaxPool2D((2, 2))
    ])

conv_decoder = models.Sequential([
    layers.Conv2D(8, (3, 3), padding='same', activation='selu',
                                          input_shape=[7, 7, 8]),
    layers.UpSampling2D((2, 2)),
    layers.Conv2D(4, (3, 3), padding='same', activation='selu'),
    layers.UpSampling2D((2, 2)),
    layers.Conv2D(1, (3, 3), padding='same', activation='sigmoid'),
    layers.Reshape([28, 28])
    ])

conv_ae = models.Sequential([conv_encoder, conv_decoder])
conv_ae.compile(loss='binary_crossentropy', optimizer='adam')
history = conv_ae.fit(X_train_noisy, X_train, epochs=20,
                          validation_data=(X_test_noisy, X_test))
```

Results--noisy images (top) and clean autoencoder reconstructions (bottom):