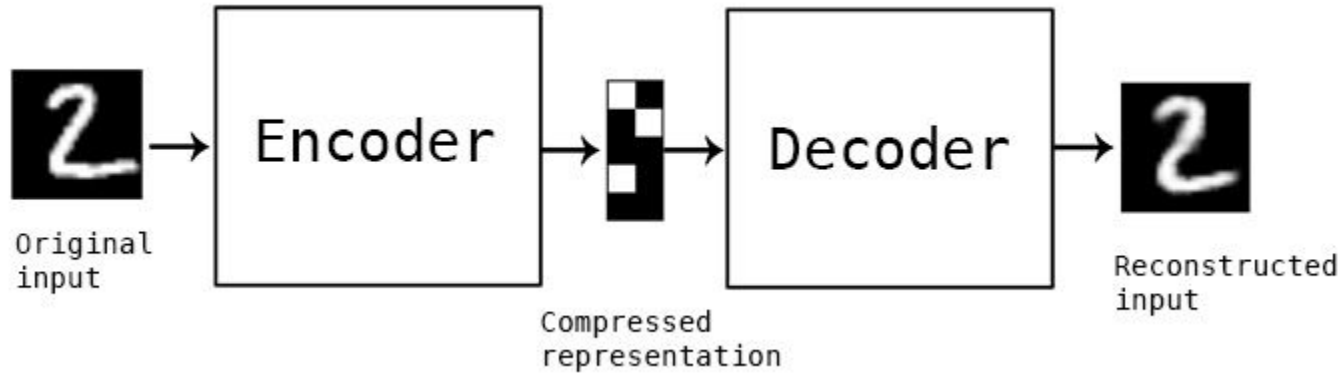


Building Autoencoders in Keras

Autors:

Filip Grześkowiak
Adam Ćwian

What are autoencoders?



Autoencoding

Algorithm in which data compression and decompression functions are:

- data-specific - only able to compress data similar to what they have been trained on
- lossy - decompressed outputs will be degraded compared to original
- learned automatically from data examples - require only appropriate training data

What are they used for?

Two most popular practical applications of autoencoders are:

- data denoising
- dimensionality reduction for data visualization e.g. with **t-SNE** (**t-distributed stochastic neighbor embedding**)

t-SNE



Basic autoencoder

```
input_img = Input(shape=(784,))  
# "encoded" is the encoded representation of the input  
encoded = Dense(encoding_dim, activation='relu')(input_img)  
# "decoded" is the lossy reconstruction of the input  
decoded = Dense(784, activation='sigmoid')(encoded)  
  
autoencoder = Model(input_img, decoded)
```

Basic autoencoder

```
encoder = Model(input_img, encoded)

encoded_input = Input(shape=(encoding_dim,))
# retrieve the last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]
# create the decoder model
decoder = Model(encoded_input, decoder_layer(encoded_input))
```

Basic autoencoder

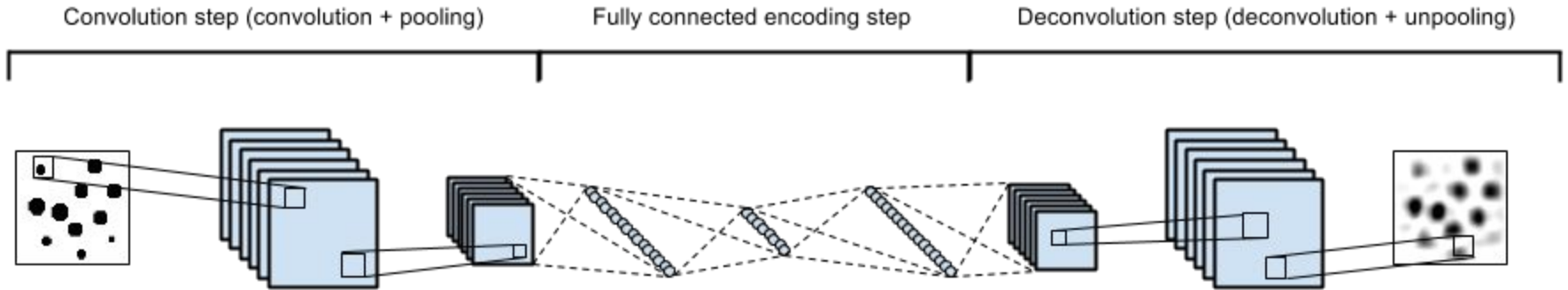
```
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
autoencoder.fit(x_train, x_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))

encoded_imgs = encoder.predict(x_test)
decoded_imgs = decoder.predict(encoded_imgs)
```

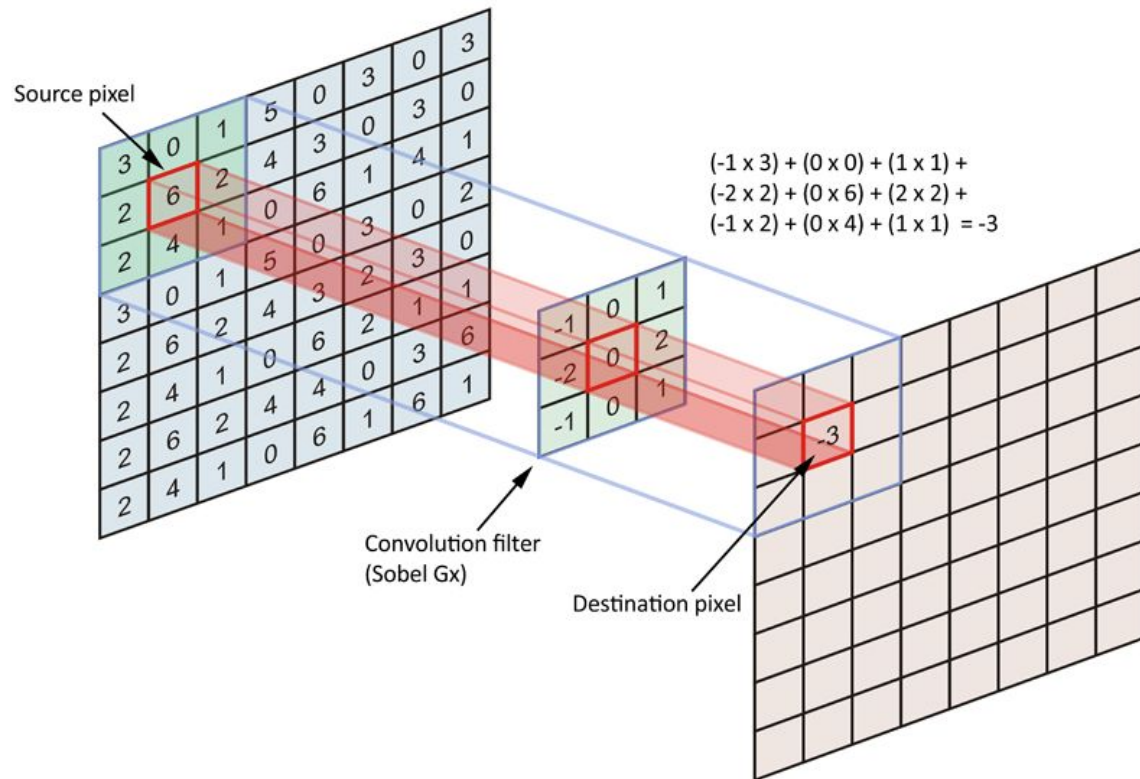

Digits reconstruction



Convolutional autoencoder



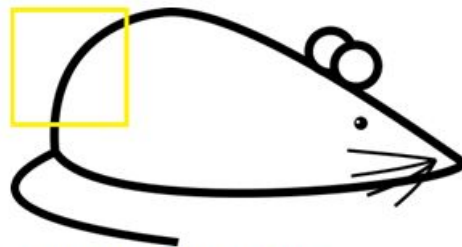
Convolution



Convolution



Original image



Visualization of the filter on the image



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

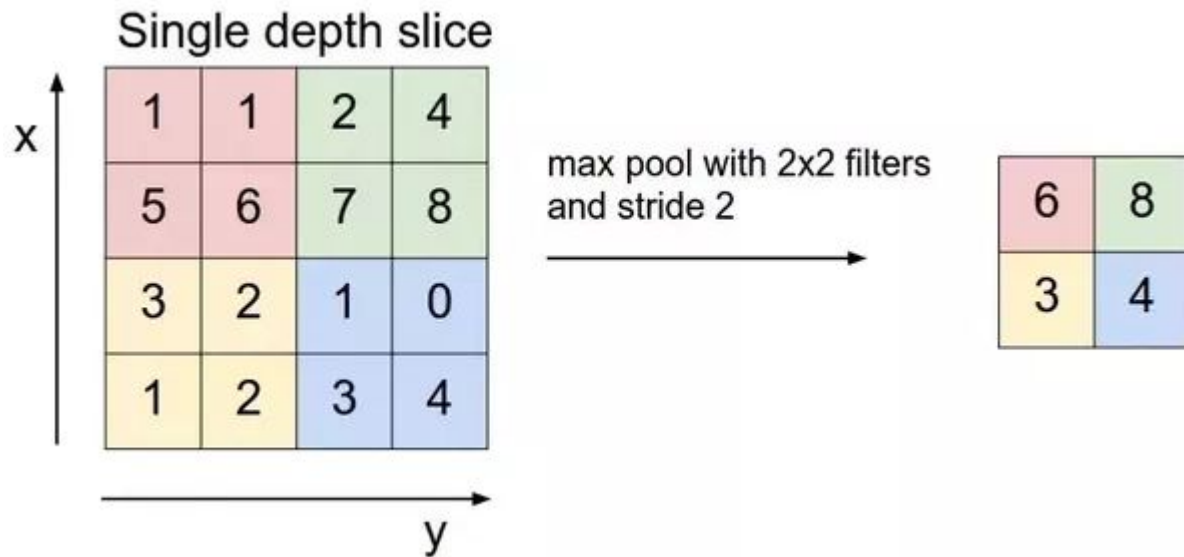
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = $(50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600$ (A large number!)

Max pooling



CNN Implementation

```
input_img = Input(shape=(28, 28, 1))

x = Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

# at this point the representation is (4, 4, 8) i.e. 128-dimensional

x = Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(16, (3, 3), activation='relu')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelata', loss='binary_crossentropy')
```



Image denoising

```
autoencoder.fit(x_train_noisy, x_train,  
               epochs=100,  
               batch_size=128,  
               shuffle=True,  
               validation_data=(x_test_noisy, x_test),  
               callbacks=[TensorBoard(log_dir='/tmp/tb', histogram_freq=0, write_graph=False)])
```

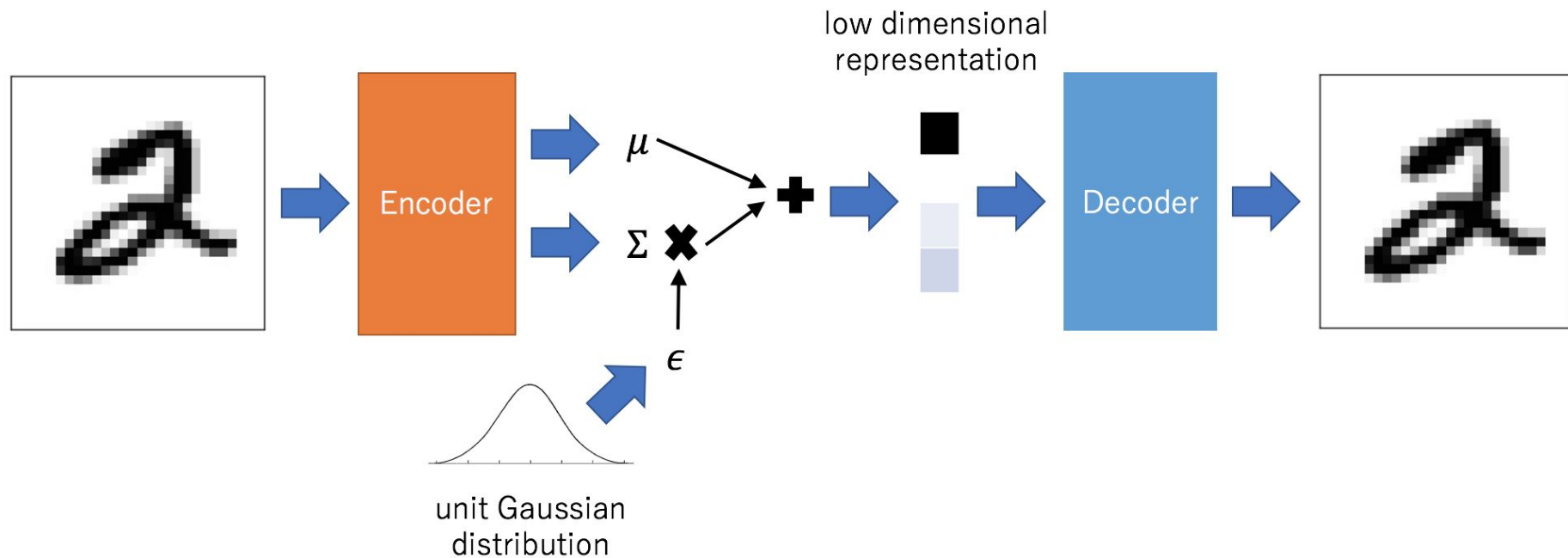
Image denoising



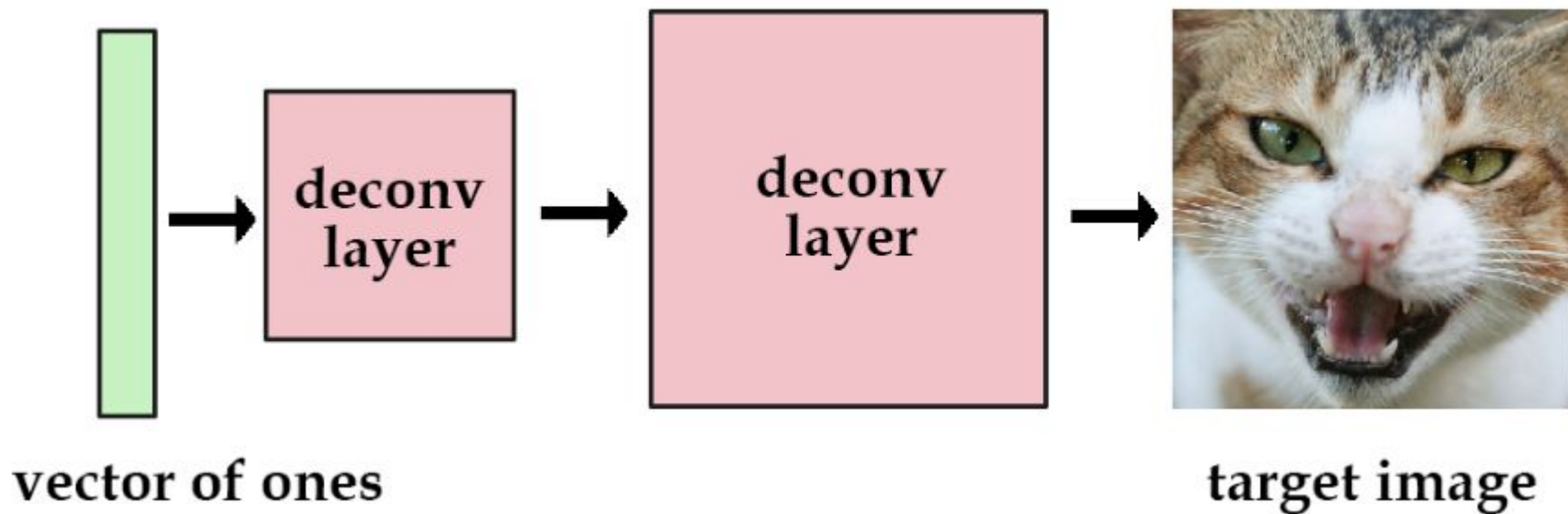
Documents denoising

There exist several methods to design forms with fields to fields may be surrounded by bounding boxes, by light rectangles. These methods specify where to write and, therefore, minimize overlapping with other parts of the form. These guides can be on a separate sheet of paper that is located below the form or they can be on the form. The use of guides on a separate sheet is much better for the quality of the scanned image, but requires giving more instructions. This restricts its use to tasks where this type of acquisition is used. Light rectangles on the form are more commonly used for this reason. Light rectangles are more easily removed with filters than dark lines whenever the handwritten text is present. Nevertheless, other practical issues must be taken into account.

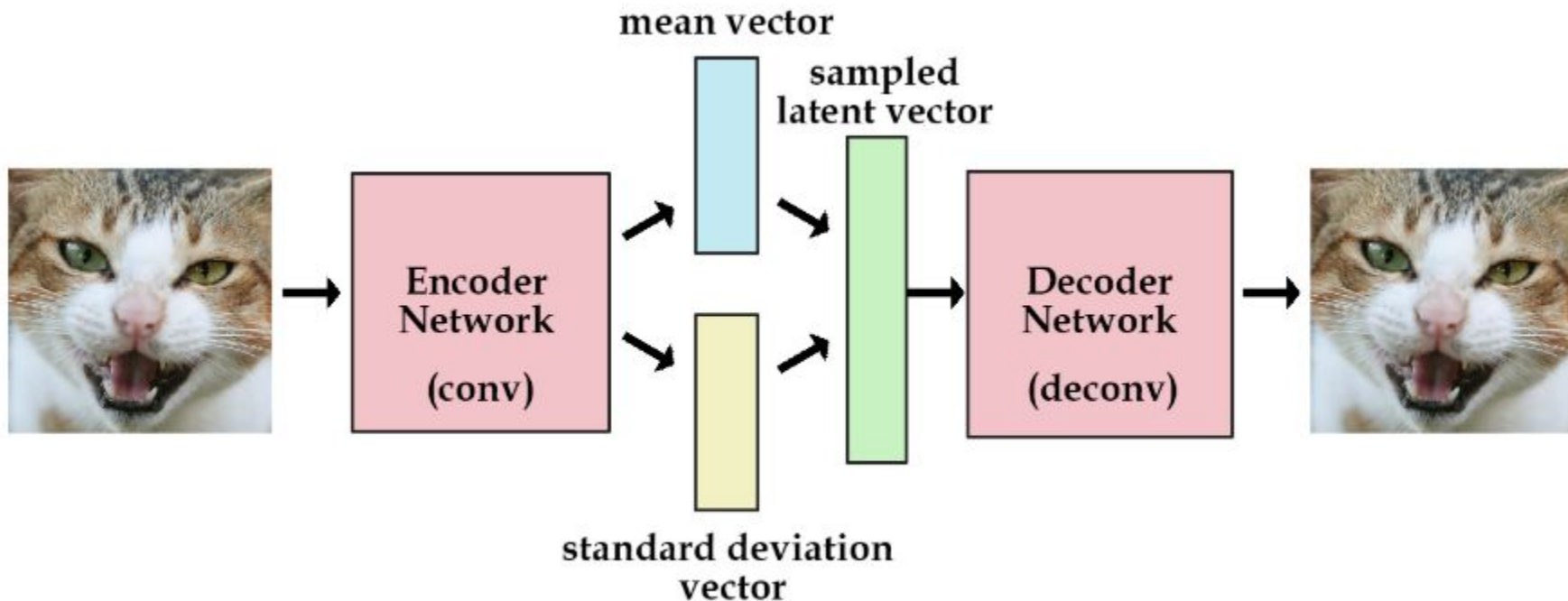
Variational autoencoder



Decoder



We add a constraint on the encoding network, that forces it to generate latent vectors that follow a unit gaussian distribution



Generative model result



Latent space visualization

