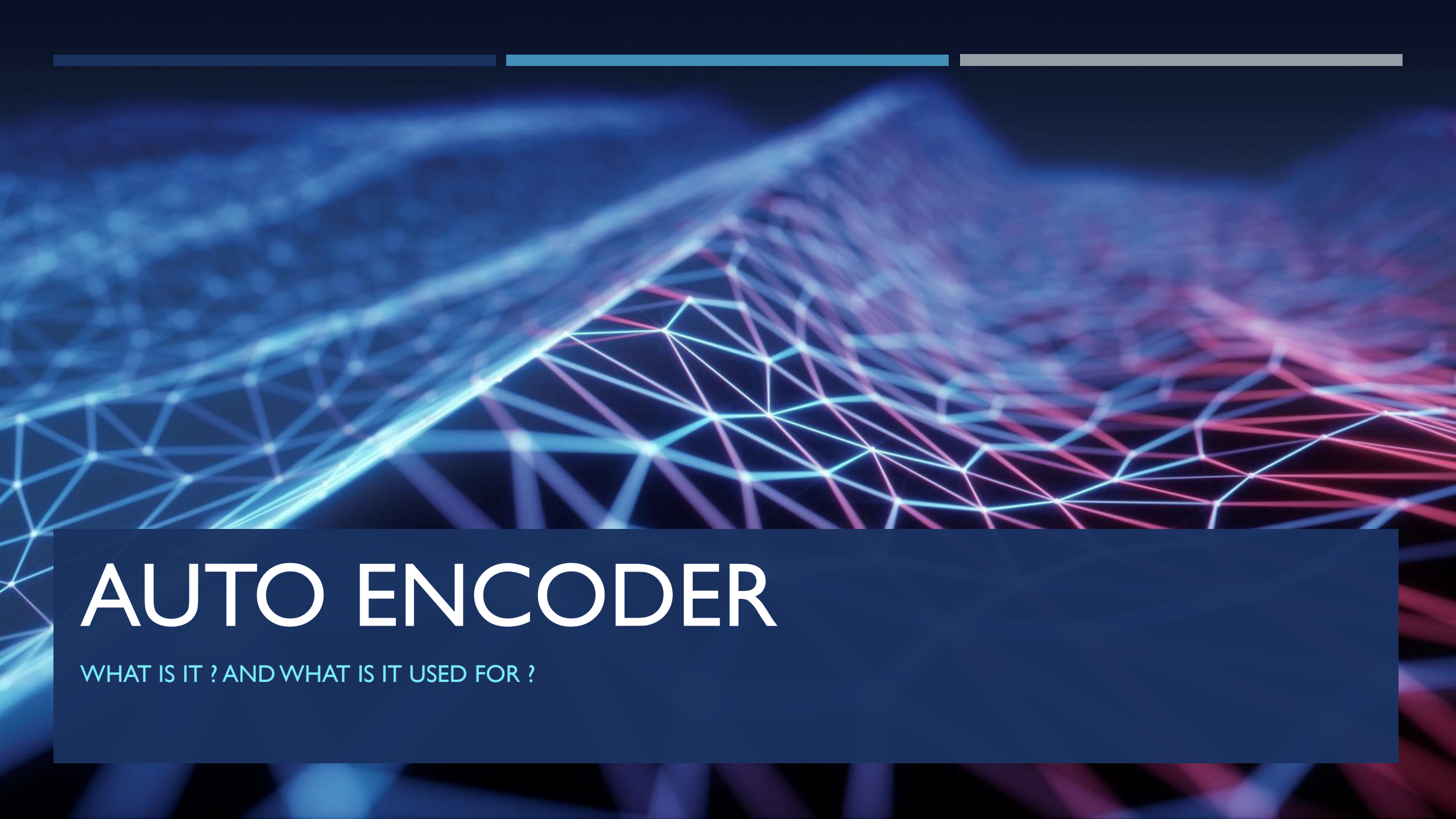


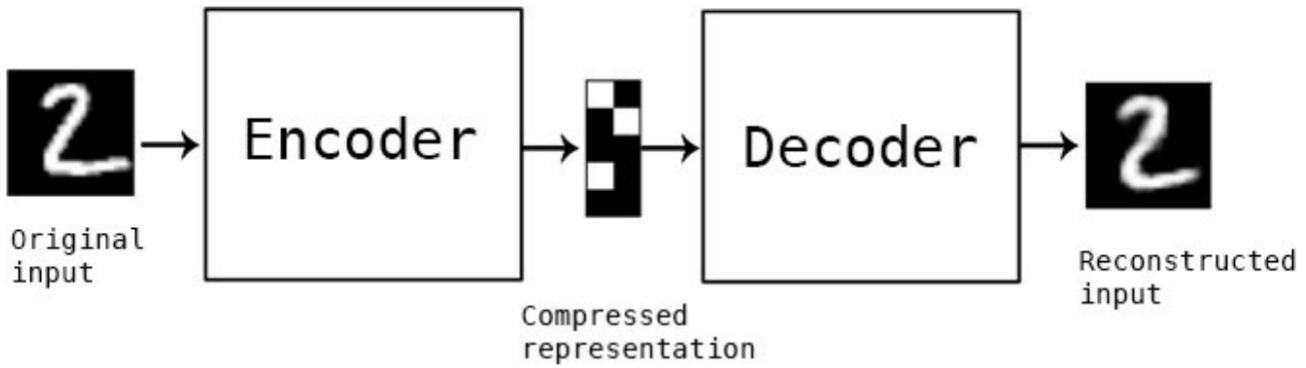
AUTO ENCODER

WHAT IS IT ? AND WHAT IS IT USED FOR ?

AUTOENCODER IS AN **UNSUPERVISED** ARTIFICIAL NEURAL NETWORK

- شبکه عصبي خود رمز نگار نوعي از شبکه هاي بدون نظارت است که به صورت موثري فشرده سازي و رمز نگاري ورودي را ياد ميگيرد و همچنين ياد مي گيرد چگونه داده فشرده و رمز شده را بازسازي کند به گونه اي که بيشترين ميزان شباهت را به ورودي داشته باشد .

- خودرمزنگار ها به خاطر مدل طراحي شون ابعاد داده را به کمك يادگيريِ نحوه نادیده گرفتن نویز ها, کاهش میدهند .



Autoencoder for MNIST

دیتاست MINST مجموعه اي از تصاویر اعداد به صورت دست نویس است که حدود 60000 تصویر 28x28 درآن وجود دارد .

AUTO ENCODER COMPONENTS

Encoder	که در آن مدل یاد می‌گیرد که چگونه ابعاد ورودی را کاهش داده و داده‌های ورودی را در یک نمایش رمزگذاری شده فشرده کند.
BottleNeck	لایه ای است که شامل نمایش فشرده داده های ورودی است. که این کمترین ابعاد ممکن براي داده های ورودی است.
Decoder	که در آن مدل یاد می گیرد چگونه داده ها را از نمایش رمزگذاری شده بازسازی کند که تا حد امکان به ورودی اصلی نزدیک باشد.
Reconstruction Loss	این روشی است که میزان عملکرد دیکدر و نزدیک بودن خروجی به ورودی اصلی را اندازه گیری می کند.

چرا باید شبکه عصبي اي رو آموزش بديم که خروجي شبيه به ورودي توليد کند ؟

ساده اس , مثلا براي حذف نويز
يا تشخيص Anomaly

AUTOENCODER FOR ANOMALY DETECTION

```
import numpy as np
import keras
from keras.datasets import mnist
from keras.models import Sequential, Model
from keras.layers import Dense, Input
from keras import optimizers
from keras.optimizers import Adam

(x_train, y_train), (x_test, y_test) = mnist.load_data()
train_x = x_train.reshape(60000, 784) / 255
val_x = x_test.reshape(10000, 784) / 255

autoencoder = Sequential()
autoencoder.add(Dense(512, activation='elu', input_shape=(784,)))
autoencoder.add(Dense(128, activation='elu'))
autoencoder.add(Dense(10, activation='linear', name="bottleneck"))
autoencoder.add(Dense(128, activation='elu'))
autoencoder.add(Dense(512, activation='elu'))
autoencoder.add(Dense(784, activation='sigmoid'))
autoencoder.compile(loss='mean_squared_error', optimizer = Adam())

trained_model = autoencoder.fit(train_x, train_x, batch_size=1024,
epochs=10, verbose=1, validation_data=(val_x, val_x))
encoder = Model(autoencoder.input,
autoencoder.get_layer('bottleneck').output)
encoded_data = encoder.predict(train_x) # bottleneck representation

decoded_output = autoencoder.predict(train_x) # reconstruction
encoding_dim = 10

# return the decoder
encoded_input = Input(shape=(encoding_dim,))
decoder = autoencoder.layers[-3](encoded_input)
decoder = autoencoder.layers[-2](decoder)
decoder = autoencoder.layers[-1](decoder)
decoder = Model(encoded_input, decoder)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 6s 103us/step - loss: 0.0757 -
val_loss: 0.0505
Epoch 2/10
60000/60000 [==============================] - 6s 96us/step - loss: 0.0420 -
val_loss: 0.0355
Epoch 3/10
60000/60000 [==============================] - 6s 95us/step - loss: 0.0331 -
val_loss: 0.0301
Epoch 4/10
60000/60000 [==============================] - 6s 96us/step - loss: 0.0287 -
val_loss: 0.0266
Epoch 5/10
60000/60000 [==============================] - 6s 95us/step - loss: 0.0259 -
val_loss: 0.0244
Epoch 6/10
60000/60000 [==============================] - 6s 96us/step - loss: 0.0240 -
val_loss: 0.0228
Epoch 7/10
60000/60000 [==============================] - 6s 95us/step - loss: 0.0226 -
val_loss: 0.0216
Epoch 8/10
60000/60000 [==============================] - 6s 97us/step - loss: 0.0215 -
val_loss: 0.0207
Epoch 9/10
60000/60000 [==============================] - 6s 96us/step - loss: 0.0207 -
val_loss: 0.0199
Epoch 10/10
60000/60000 [==============================] - 6s 96us/step - loss: 0.0200 -
val_loss: 0.0193
```

- اگر داده های ورودی همبستگي داشته باشند , روش رمزگذار خودکار بسیار خوب کار می کند .

زیرا عملیات رمزگذاری برای فشرده سازی داده ها به ویژگی های مرتبط متکی است.

همانطور که در خروجی مشاهده می شود ، آخرین خطای بازسازی برای مجموعه اعتبارسنجی 0.0193 است که خوب است. حال، اگر هر تصویر معمولی اي از مجموعه داده MNISTارسال شود،

میزان ضرر بازسازی بسیار کم خواهد بود (< 0.02) اما اگر هر تصویر متفاوت دیگری (پرت یا نویزي) ارسال شود، مقدار تلفات بازسازی بالایی دریافت خواهد شد،

زیرا شبکه نمیتواند تصویر/ورودی را که یک ناهنجاری تلقی می‌شود، بازسازی کند.

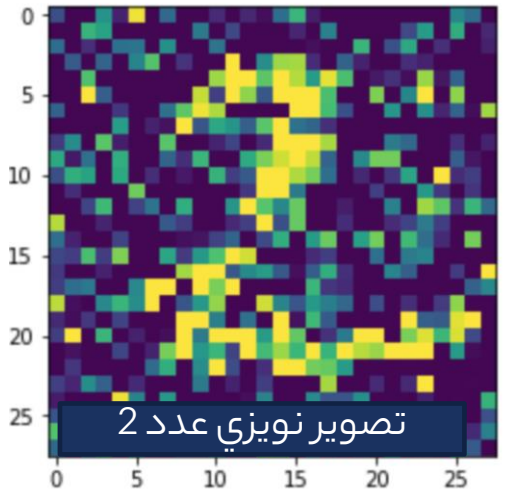
AUTOENCODER FOR ANOMALY DETECTION

```
%matplotlib inline
from keras.preprocessing import image
# if the img.png is not one of the MNIST dataset that the model was trained on, the error will be very high.
img = image.load_img("./img.png", target_size=(28, 28), color_mode = "grayscale")
input_img = image.img_to_array(img)
inputs = input_img.reshape(1,784)
target_data = autoencoder.predict(inputs)
dist = np.linalg.norm(inputs - target_data, axis=-1)
print(dist)
```

- حال ،از شبکه براي تشخيص ناهنجاری استفاده میکنیم.
- کد رو به رو از دو تصویر مختلف برای پیش‌بینی امتیاز ناهنجاری (خطای بازسازی) با استفاده از شبکه رمزگذار خودکار که در اسلاید قبل توضیح دادیم استفاده می‌کند.
- تصویر اول از MNISTو نتیجه 5.43209 است. این بدان معنی است که تصویر یک ناهنجاری نیست.
- تصویر دوم, یک تصویر کاملا تصادفی است که به مجموعه داده آموزشی تعلق ندارد و نتیجه: 6789.4907.
- این خطای بالا به این معنی است که تصویر یک ناهنجاری است. همین مفهوم در مورد هر نوع مجموعه داده صدق می کند.

AUTOENCODER FOR DENOISING

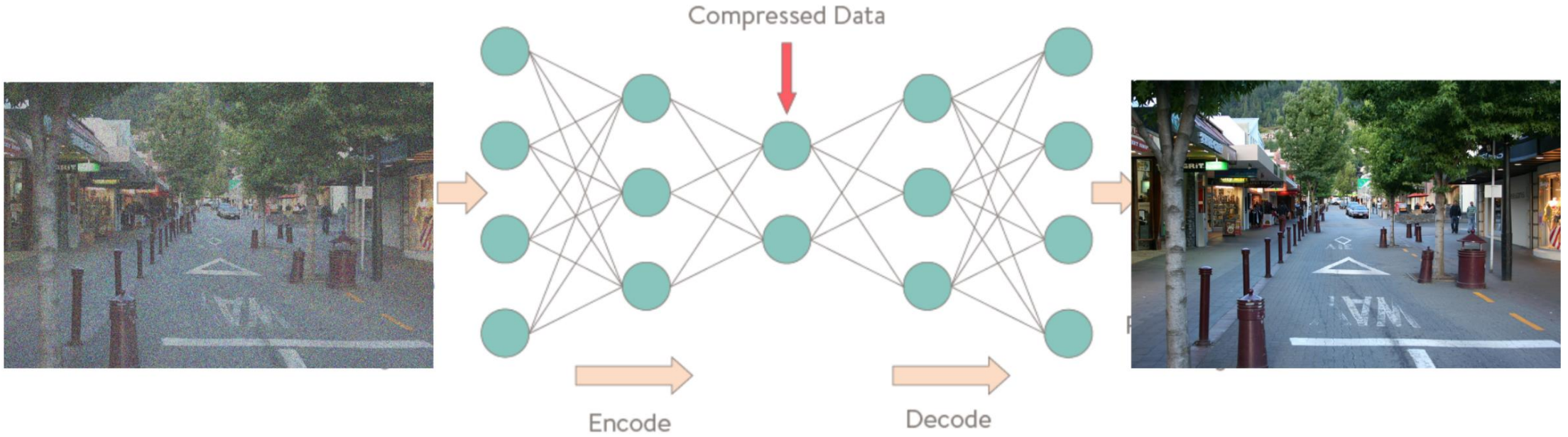
```
# The code below is from the Keras Blogs
# https://blog.keras.io/building-autoencoders-in-keras.html

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.)

#Print one image to see the noise
plt.imshow(x_test_noisy[1].reshape(28, 28))
```



تصویر نویزی عدد 2

- حذف نویز یا کاهش نویز فرآیند حذف نویز از یك سیگنال است.
- این سیگنال می تواند یک تصویر، صدا یا یک سند باشد.
- می توان یک شبکه Autoencoderرا آموزش داد تا نحوه حذف نویز از تصاویر را بیاموزد.
- برای آزمایش این مورد، باید دوباره از مجموعه داده MNIST استفاده کنیم و مقداری نویز مصنوعی در مجموعه داده ایجاد کنیم.
- کد رو به رو به سادگی مقداری نویز به مجموعه داده اضافه می کند.

EXAMPLE OF DENOISING INPUT WITH AUTO ENCODER



AUTOENCODER FOR DENOISING

```
input_img = Input(shape=(28, 28, 1))
nn = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
nn = MaxPooling2D((2, 2), padding='same')(nn)
nn = Conv2D(32, (3, 3), activation='relu', padding='same')(nn)
encoded = MaxPooling2D((2, 2), padding='same')(nn)
```

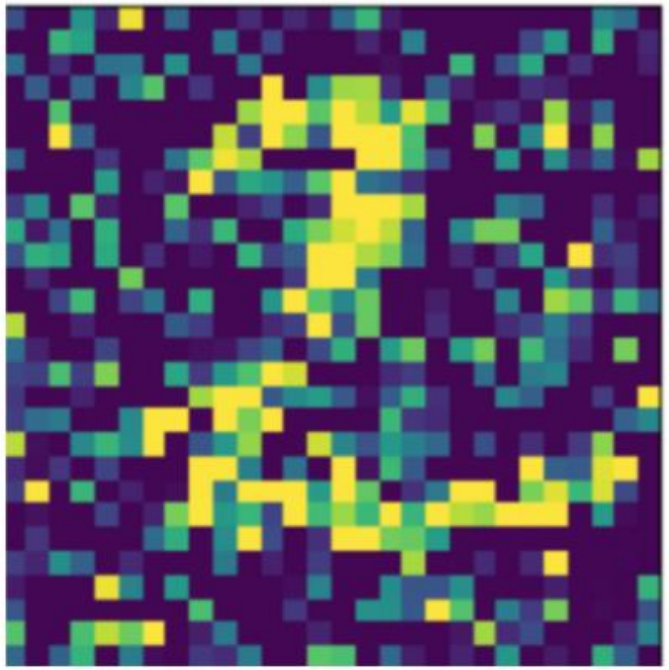
- از تابع MaxPoolingبرای کاهش ابعاد استفاده شده است. (28، 28، 32) با ضریب دو کاهش می یابد، بنابراین پس از MaxPoolingاول (14، 14، 32) و پس از MaxPoolingدوم (7، 7، 32) خواهد بود. این نمایش کدگذاری شده تصویر است.

```
nn = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)
nn = UpSampling2D((2, 2))(nn)
nn = Conv2D(32, (3, 3), activation='relu', padding='same')(nn)
nn = UpSampling2D((2, 2))(nn)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(nn)
```

- کد رو به رو قسمت بازسازی ارقام اصلی است. اینجاست که شبکه در واقع یاد می گیرد که چگونه نویز را از تصاویر ورودی حذف کند. از تابع UpSampling برای بازسازی تصاویر به ابعاد اصلی استفاده می کنیم (28، 28)

```
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelta',loss='binary_crossentropy')
autoencoder.fit(x_train_noisy, x_train,
                epochs=50,
                batch_size=256,
                validation_data=(x_test_noisy, x_test))
```

- آخرین مرحله ایجاد مدل، کامپایل آن و شروع آموزش است.



پس از اتمام آموزش، تصویر نویزي را از طریق شبکه عبور مي دهیم و نتیجه کاملاً چشمگیر است، نویز تصویر به طور کامل حذف شده است .

THANK YOU

PEYMAN SHOBEIRI

DANIAL NAHOFTE

ARYAN FARMAN