

オートエンコーダーと異常検知入門

スカイマインド株式会社

本橋 和貴

自己紹介

▶ 本橋 和貴 @kmotohas

- スカイマインド株式会社
 - Deep Learning Engineer (前職ではDL+ROS)
- 素粒子物理学実験 (LHC-ATLAS実験) 出身
 - 博士 (理学)
- 好きな定理 : ネーターの定理

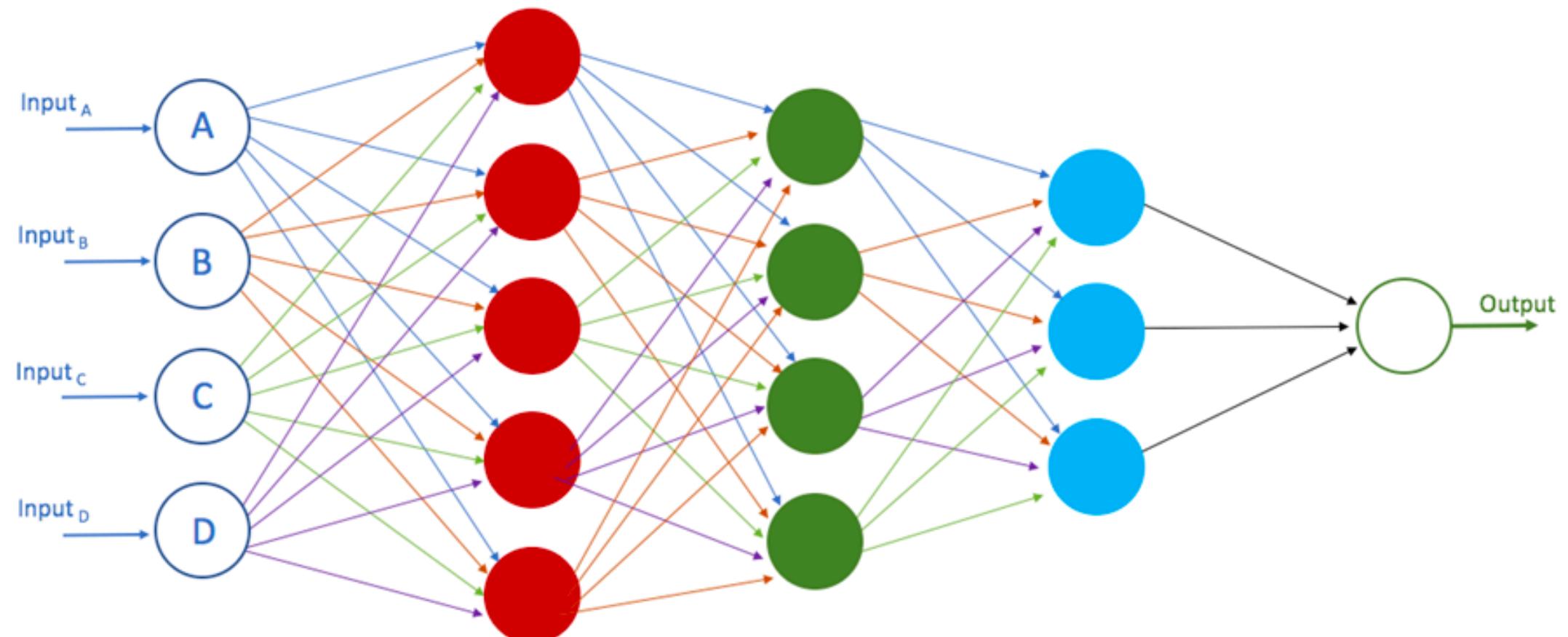


系に連続的な**対称性**がある場合はそれに対応する**保存則**が存在すると述べる定理である。(Wikipedia)

今回話すこと

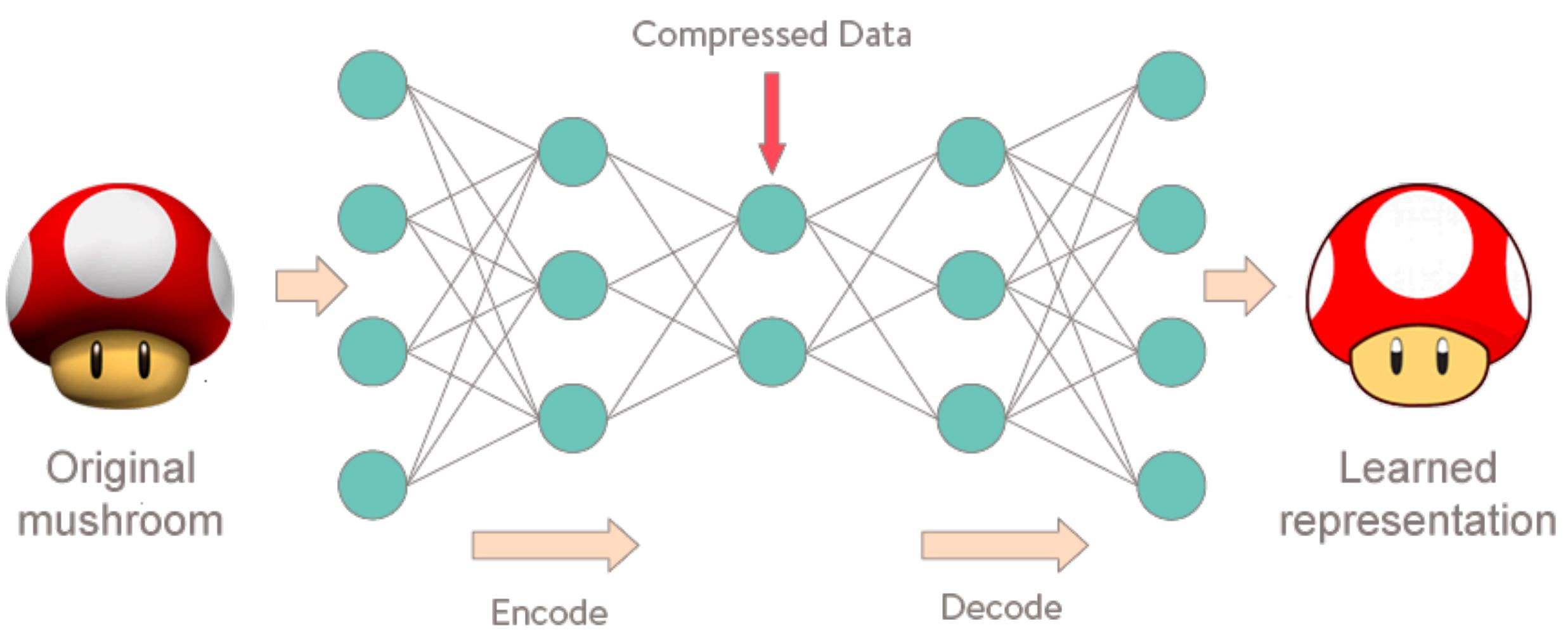
- ▶ オートエンコーダーとは何か
 - Convolutional Autoencoder
- ▶ 異常検知への適用
 - テーブルデータ
 - 画像データ

オートエンコーダーとは



一般的なニューラルネットワーク

$$f(x) = y$$

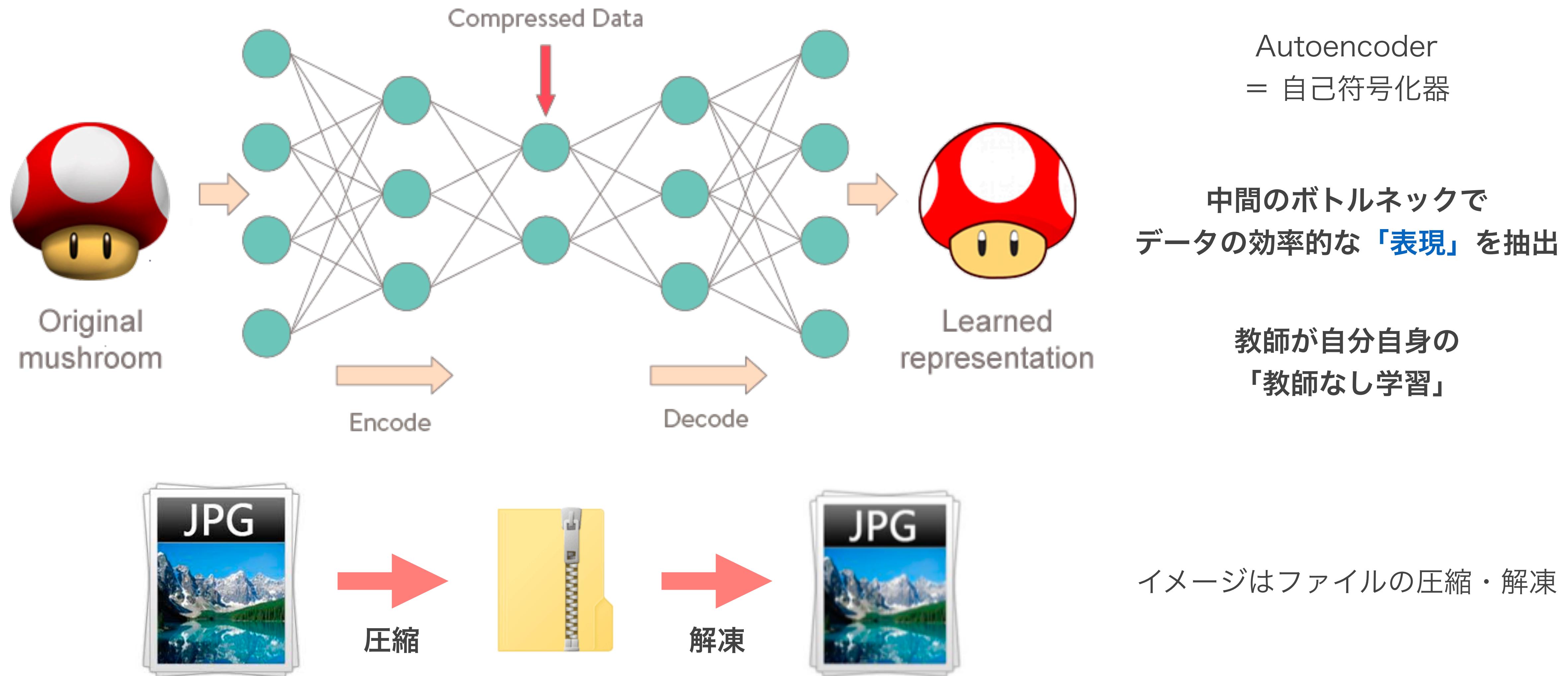


<https://medium.com/@curiously/credit-card-fraud-detection-using-autoencoders-in-keras-tensorflow-for-hackers-part-vii-20e0c85301bd>

オートエンコーダー

$$f(x) = x$$

オートエンコーダーのイメージ



Tensorflow (Keras) での実装例

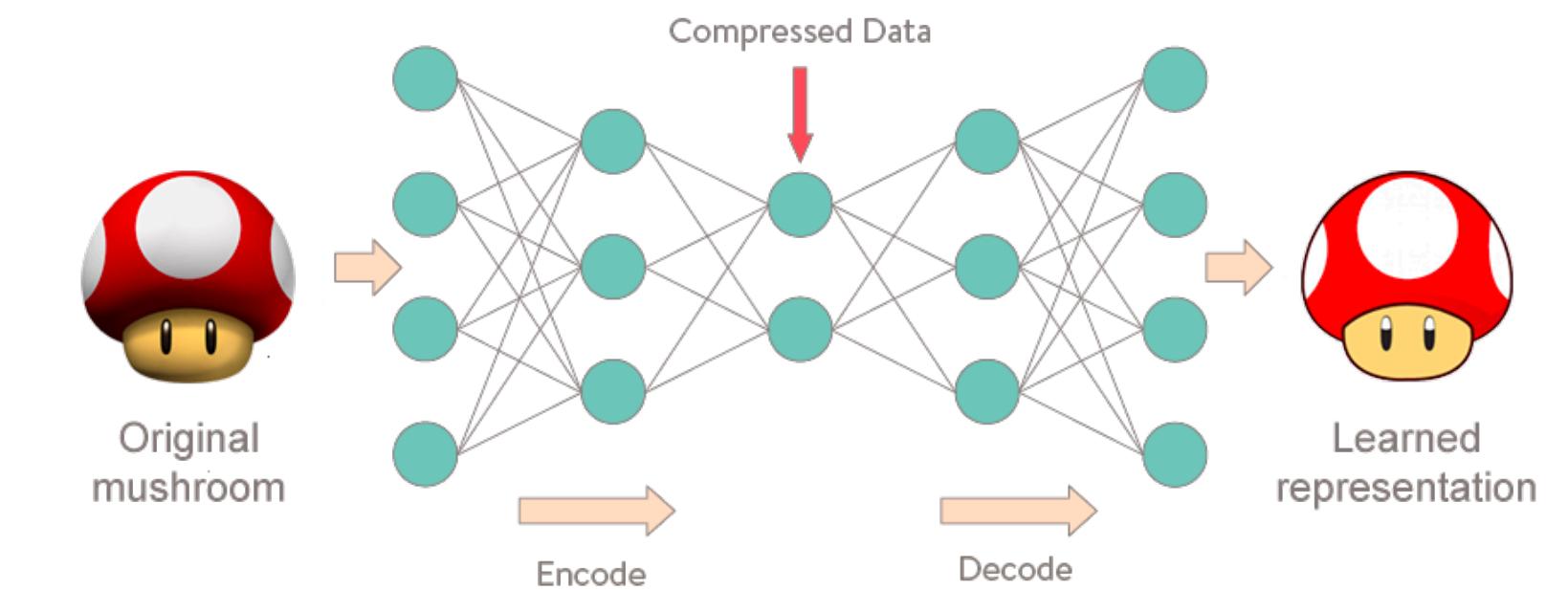
```
from tensorflow.keras.layers import Input, Dense  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.datasets import mnist
```

← 用いるモジュールのインポート

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()  
x_train = x_train / 255.  
x_train = x_train.reshape(len(x_train), 28*28)
```

← データの読み込み、正規化
二次元の画像を一次元のベクトルに変換

```
model = Sequential()  
model.add(Dense(256, activation='relu', input_shape=(28*28,), name='encoder_1'))  
model.add(Dense(100, activation='relu', name='encoder_2'))  
model.add(Dense(256, activation='relu', name='decoder_1'))  
model.add(Dense(28*28, activation='sigmoid', name='decoder_2'))  
model.summary()
```



```
model.compile(optimizer='adam', loss='mean_squared_error')  
model.fit(x_train, x_train, epochs=16, batch_size=128)
```

← 入力と目標の出力はどちらも訓練データとして学習

テストデータの復元

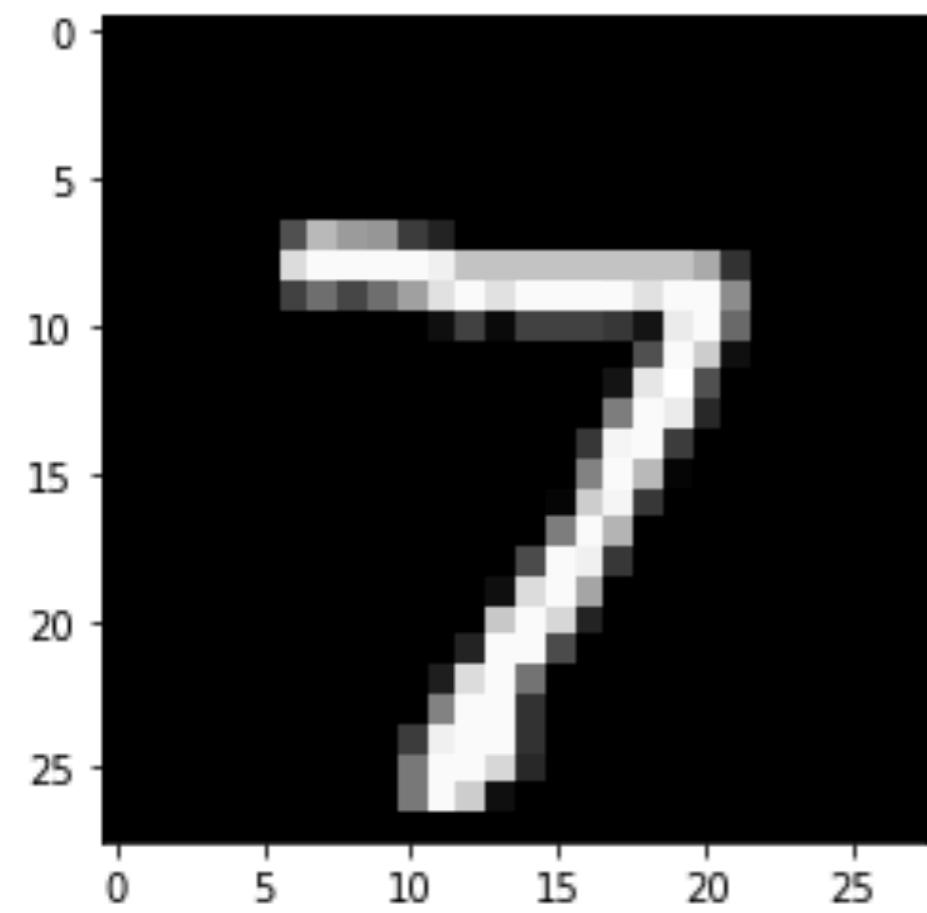
```
import numpy as np  
decoded = model.predict(x_test.reshape(len(x_test), 28*28))
```

← テストデータを一次元ベクトルにしてモデルに入力

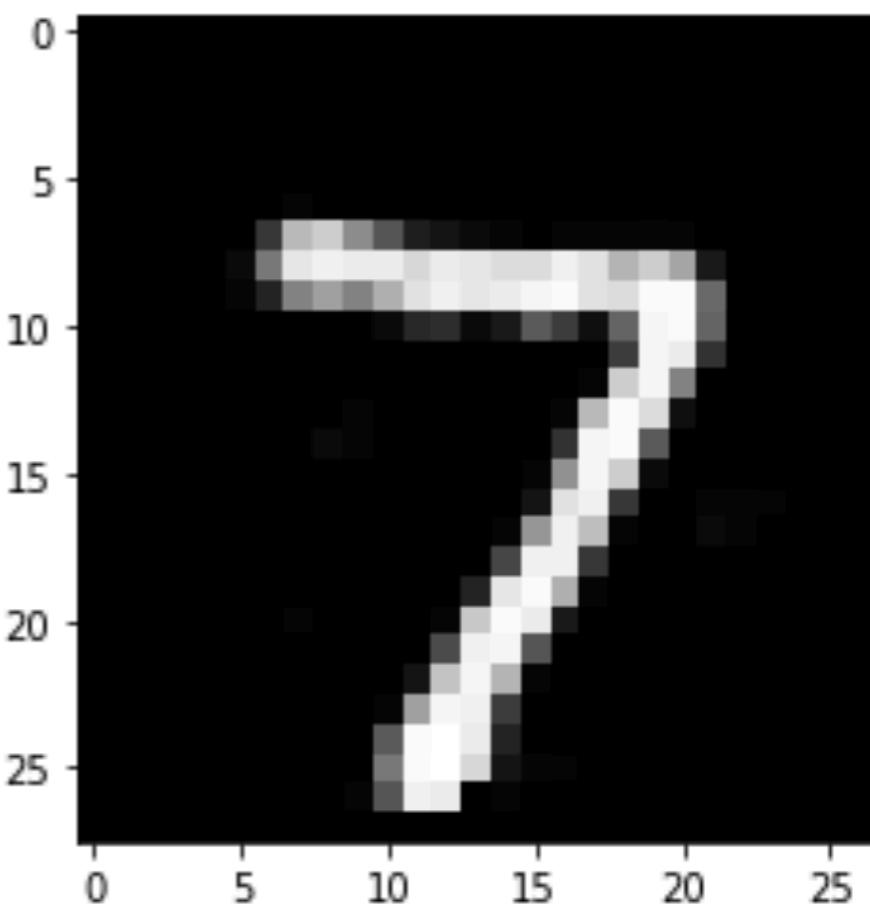
```
import matplotlib.pyplot as plt  
fig, (axL, axR) = plt.subplots(ncols=2, figsize=(10,4))  
axL.imshow(x_test[0].reshape(28, 28), 'gray')  
axR.imshow(decoded[0].reshape(28, 28), 'gray')
```

← matplotlibで一枚めの画像の復元結果を確認

<matplotlib.image.AxesImage at 0x7f8c0556d908>



元データ

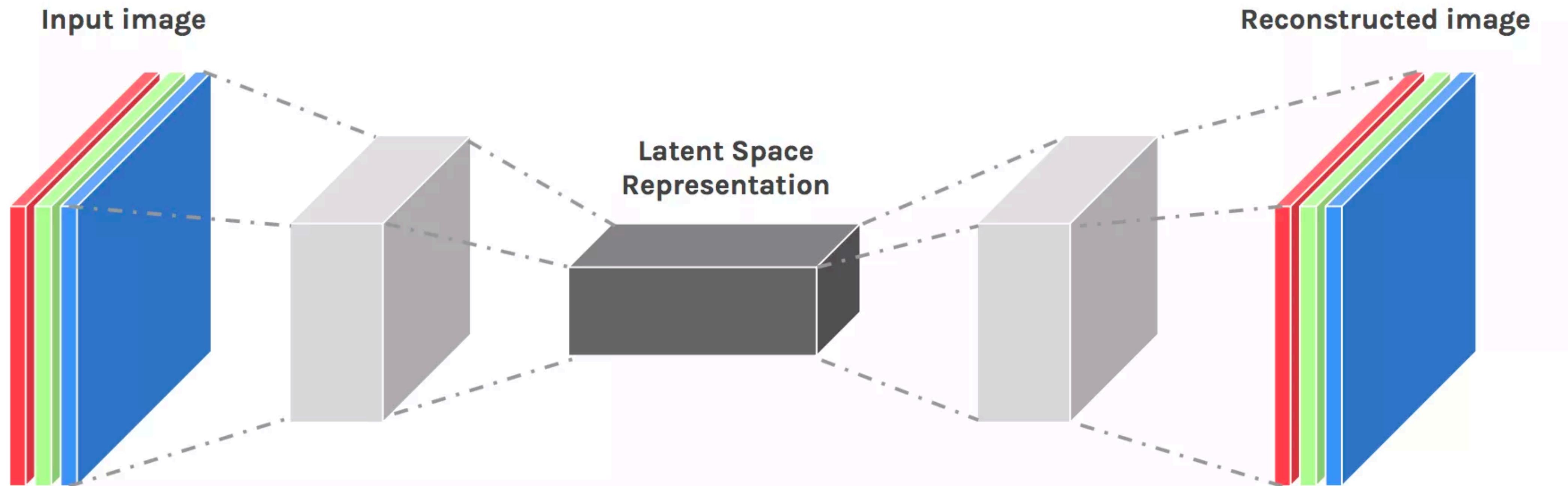


復元されたデータ

28×28 = 784 から 100 次元まで情報圧縮しても
データをうまく復元できている
(符号化しても情報がほとんど失われていない)

CAE (Convolutional AutoEncoder)

全結合層ではなく畳み込み層を用いたオートエンコーダー



<https://sefiks.com/2018/03/23/convolutional-autoencoder-clustering-images-with-neural-networks/>

Keras を用いた CAE の実装例

```
model = Sequential()

# 1st convolution layer
model.add(Conv2D(16, (3, 3), padding='same', activation='relu',
                 input_shape=(256, 256, 3)))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))

# 2nd convolution layer
model.add(Conv2D(8, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))

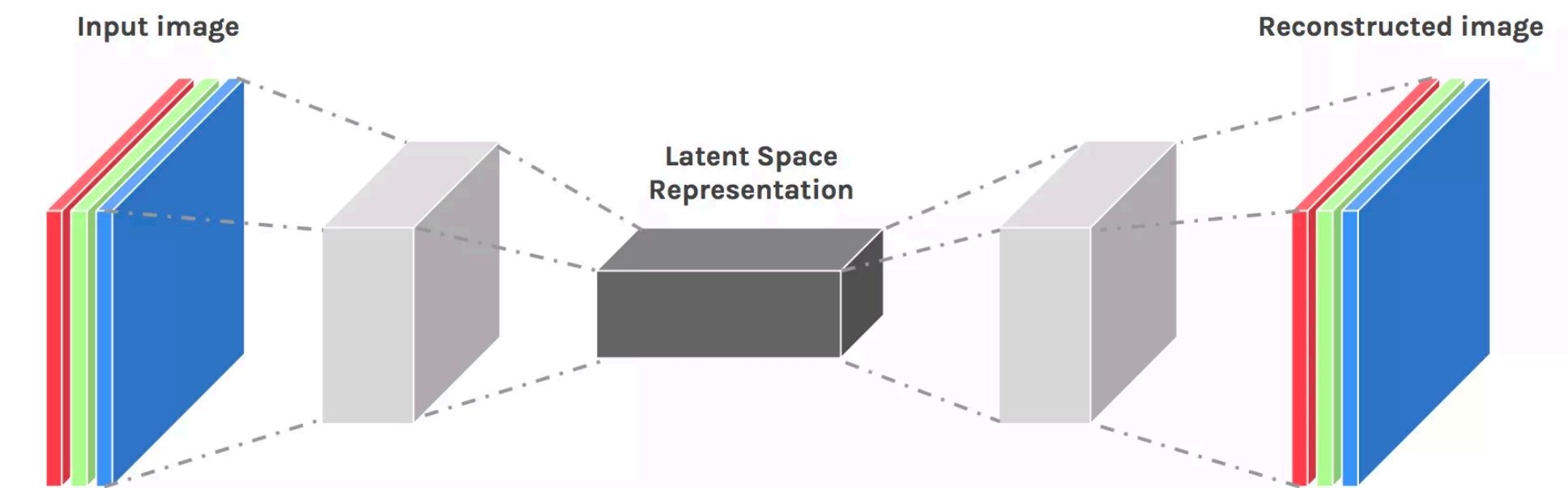
# latent space representation

# 3rd convolution layer
model.add(Conv2D(8, (3, 3), padding='same', activation='relu'))
model.add(UpSampling2D((2, 2)))

# 4th convolution layer
model.add(Conv2D(16, (3, 3), padding='same', activation='relu'))
model.add(UpSampling2D((2, 2)))

model.add(Conv2D(3, (3, 3), padding='same', activation='sigmoid'))
```

← MaxPooling2D で画像サイズを小さくする



← UpSampling2D で画像サイズを大きくする

Keras を用いた CAE の実装例

```
model = Sequential()

# 1st convolution layer
model.add(Conv2D(16, (3, 3), padding='same', activation='relu',
                 input_shape=(256, 256, 3)))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))

# 2nd convolution layer
model.add(Conv2D(8, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))

# latent space representation

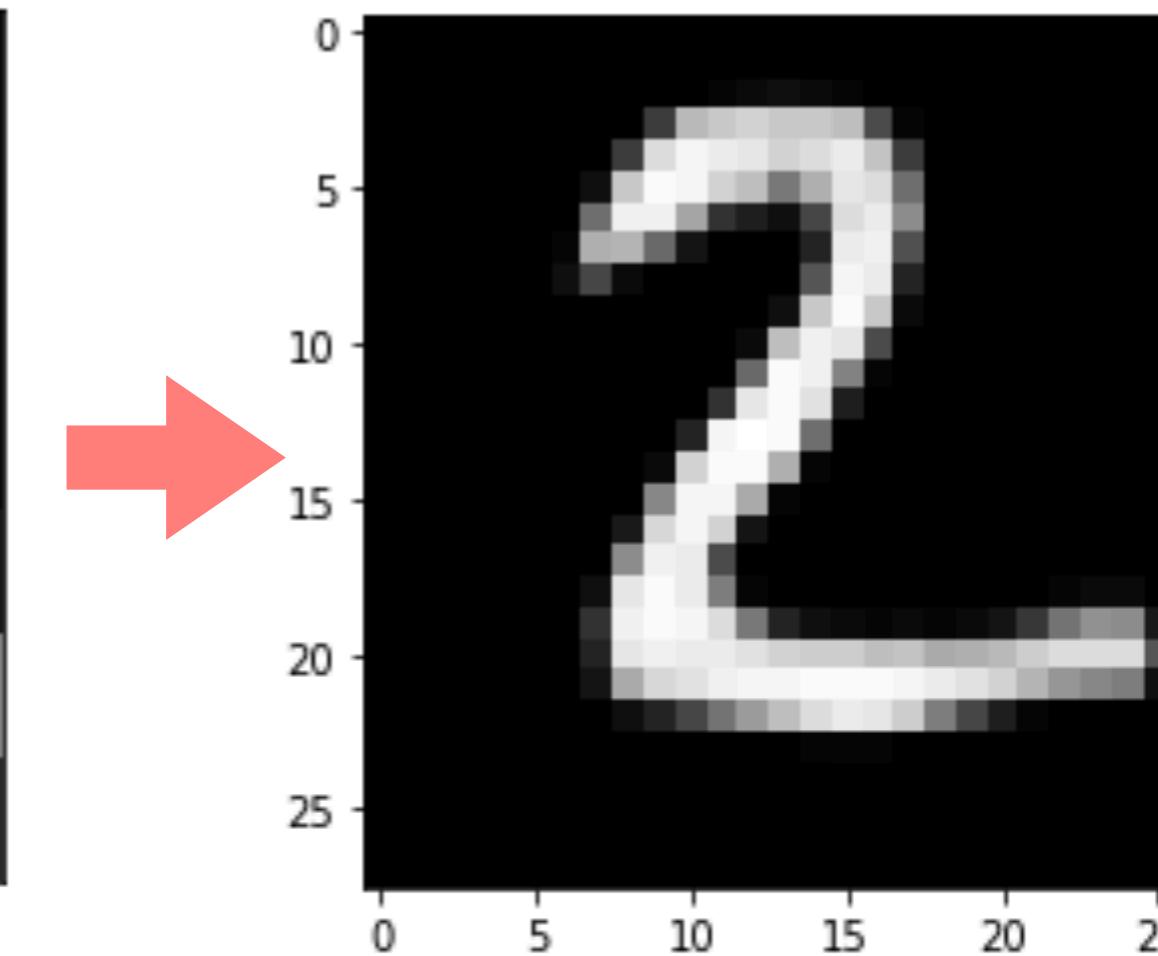
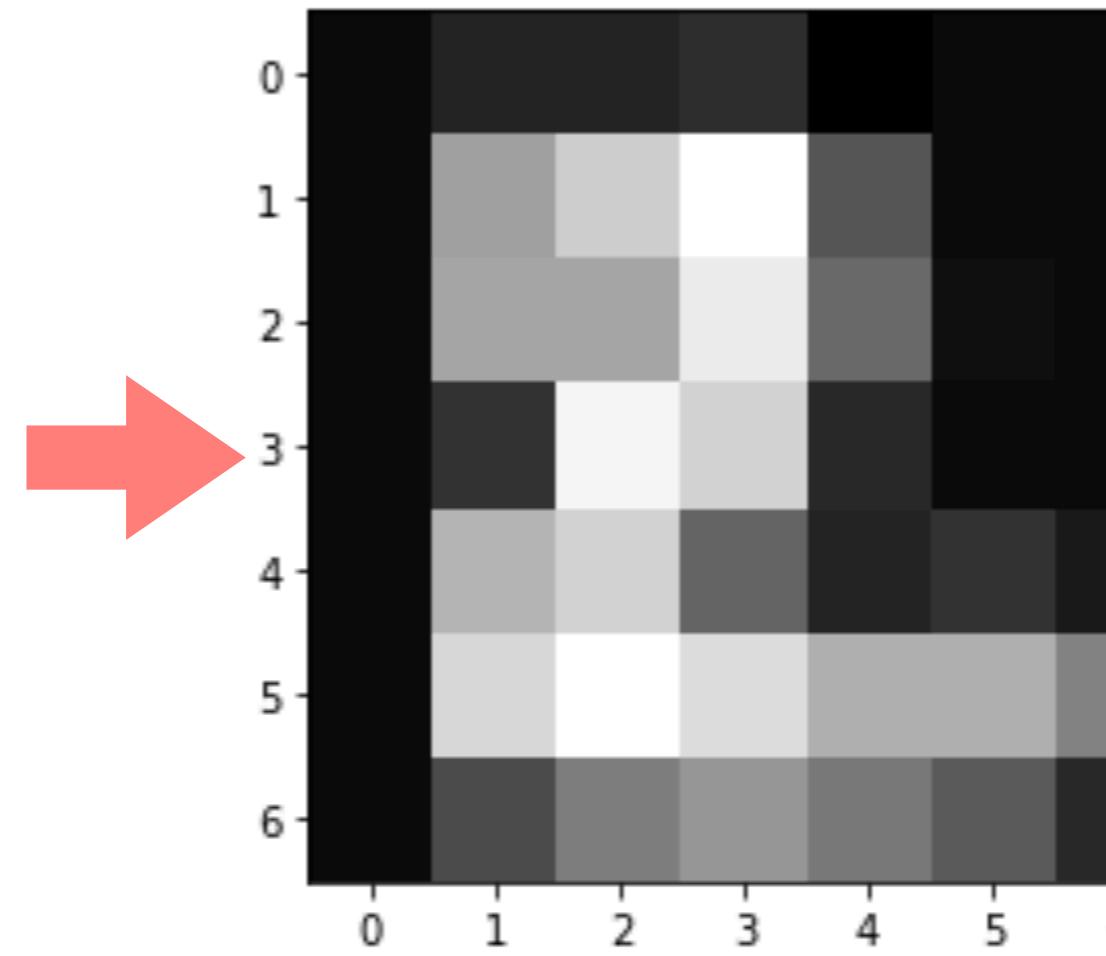
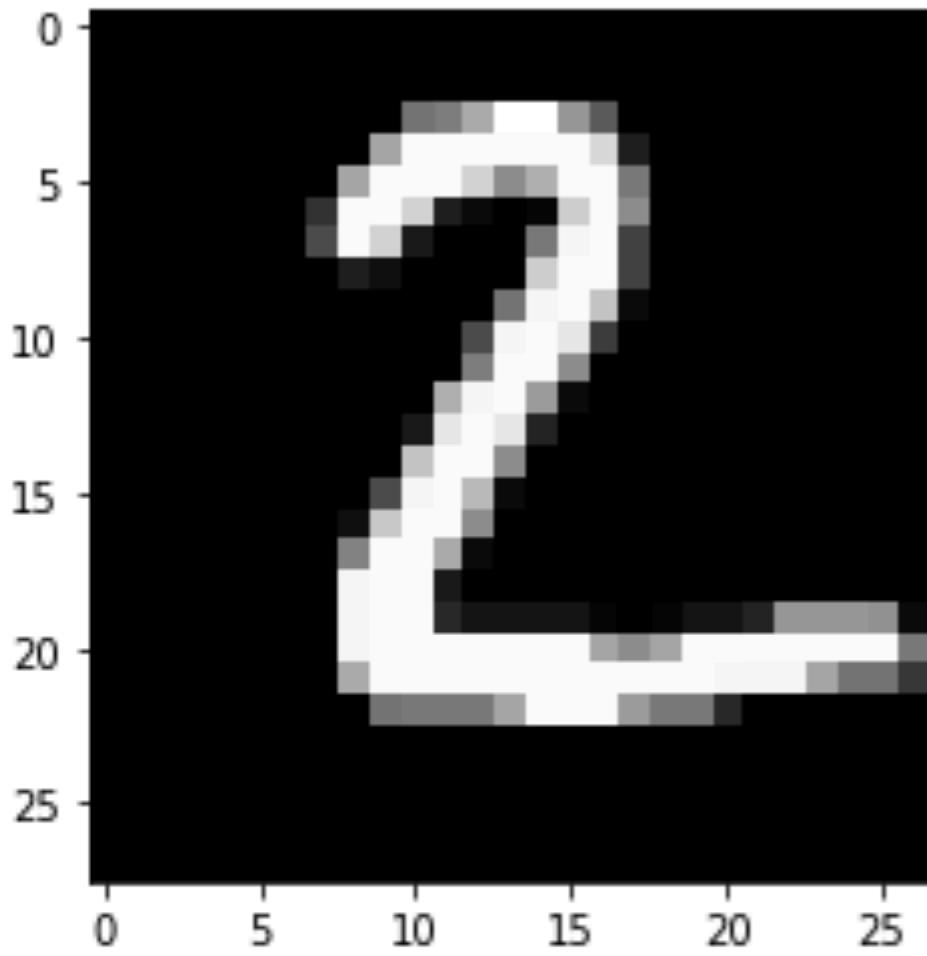
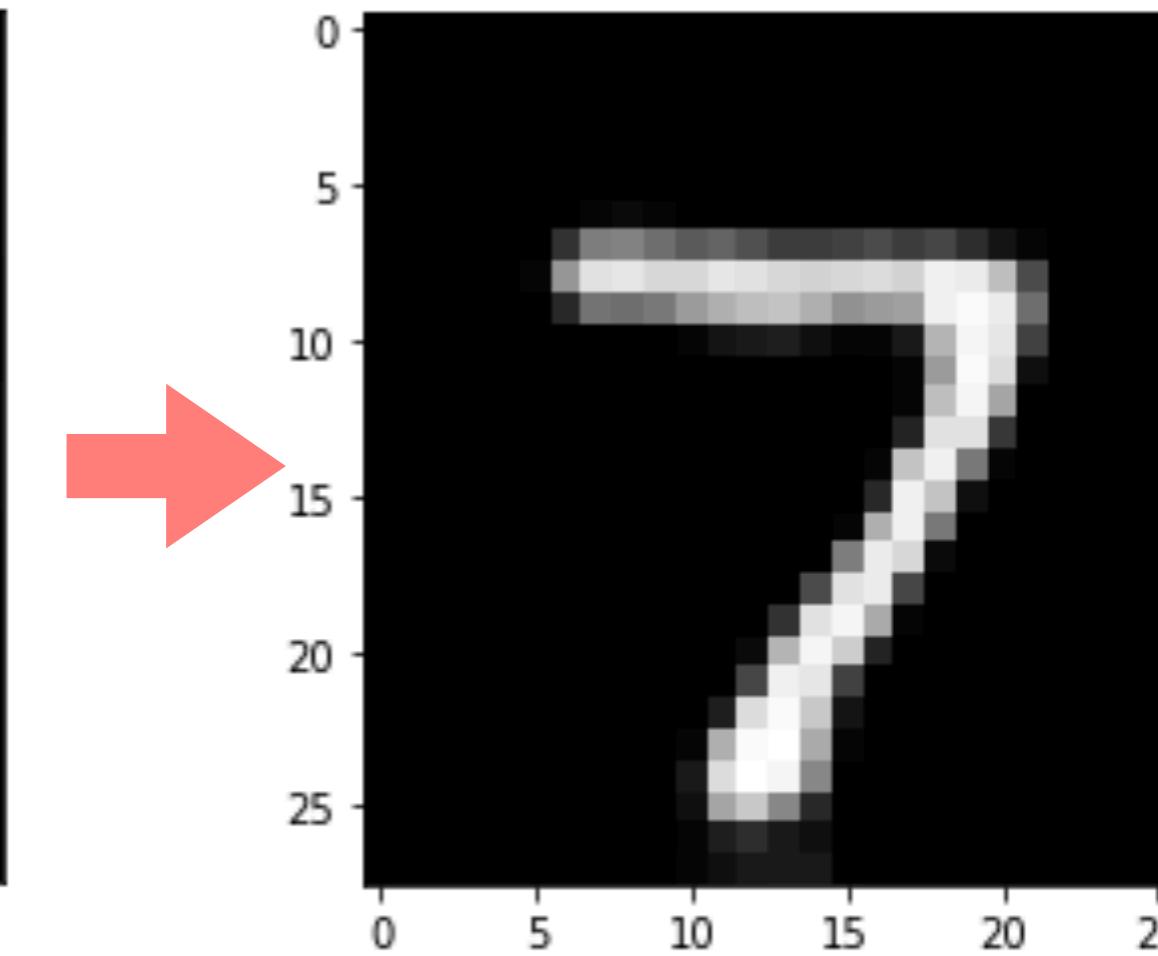
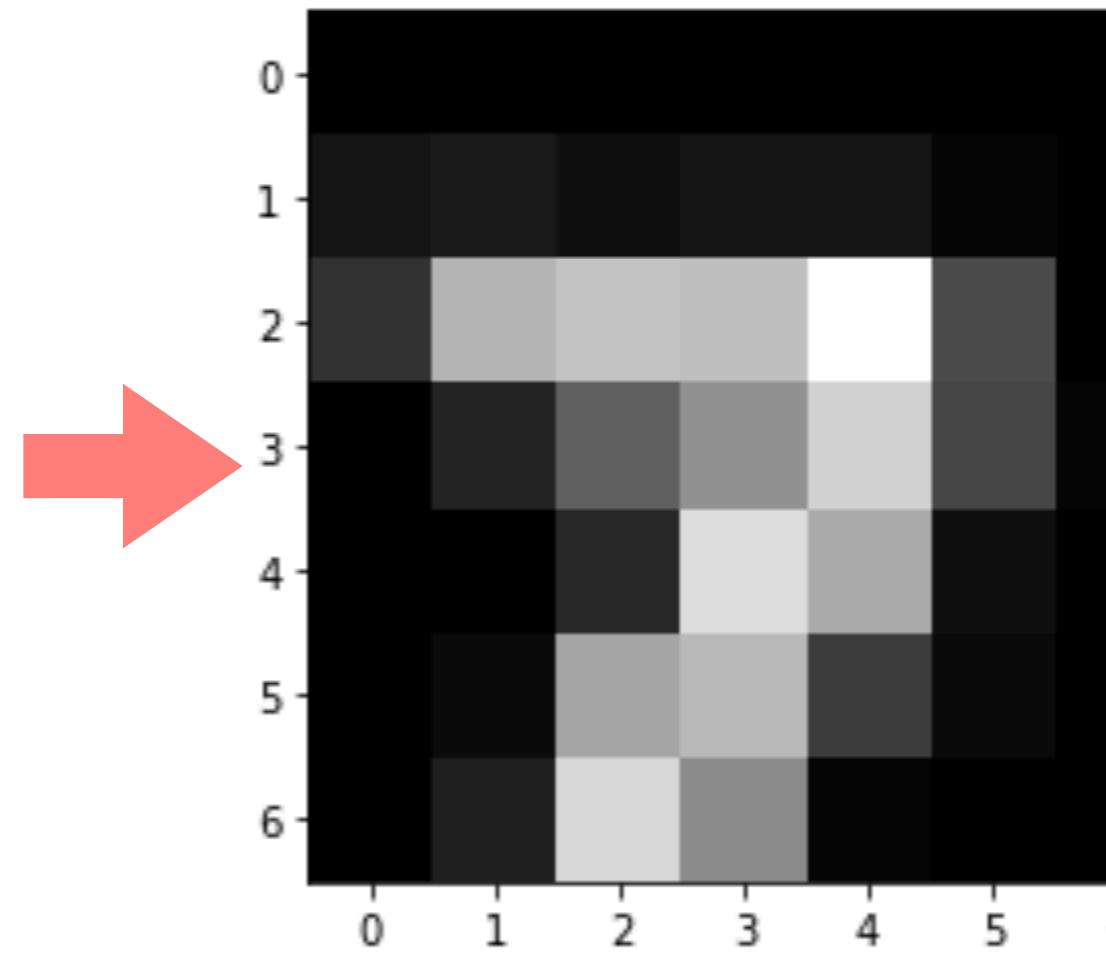
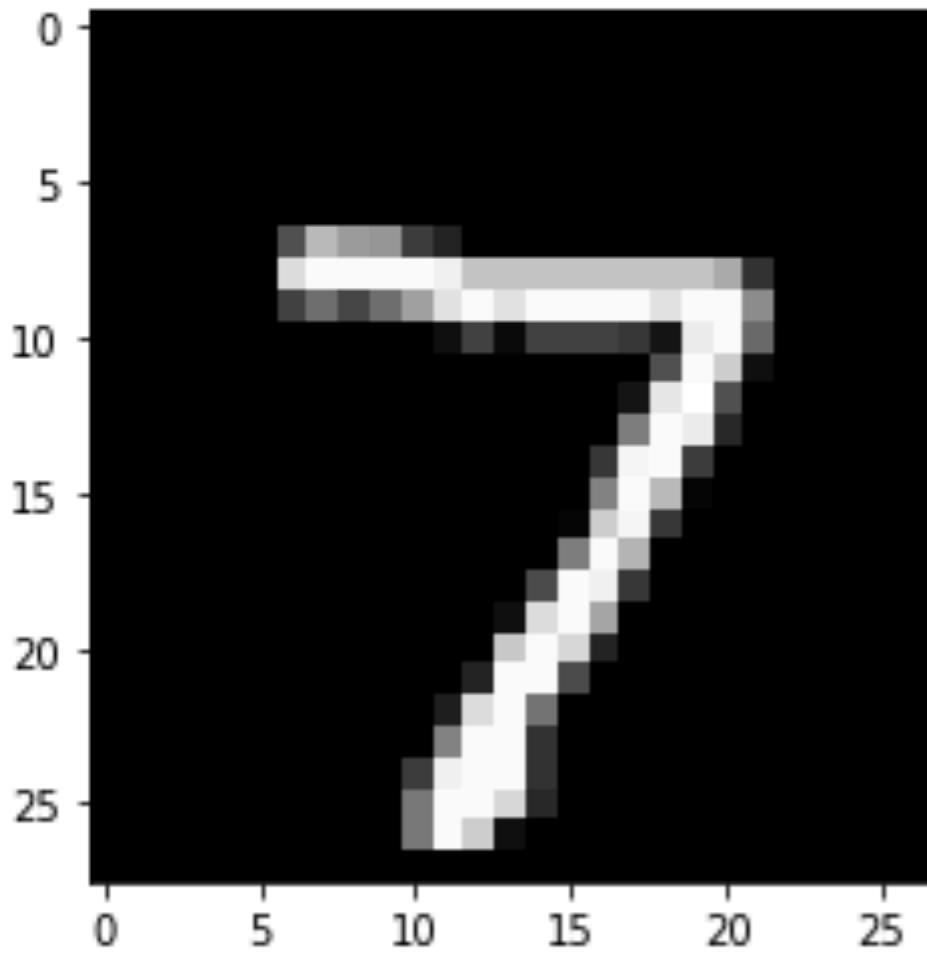
# 3rd convolution layer
model.add(Conv2D(8, (3, 3), padding='same', activation='relu'))
model.add(UpSampling2D((2, 2)))

# 4th convolution layer
model.add(Conv2D(16, (3, 3), padding='same', activation='relu'))
model.add(UpSampling2D((2, 2)))

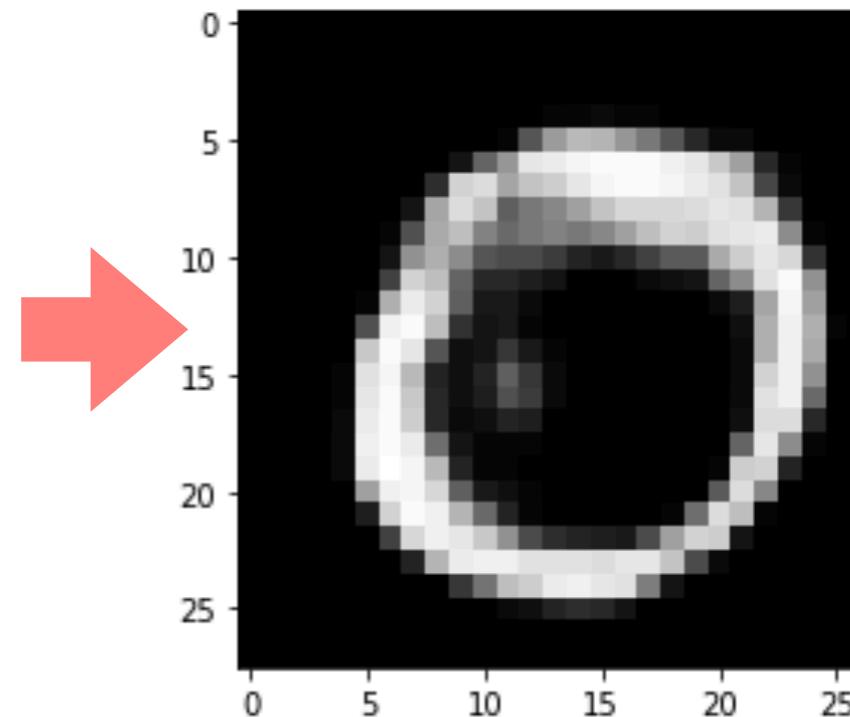
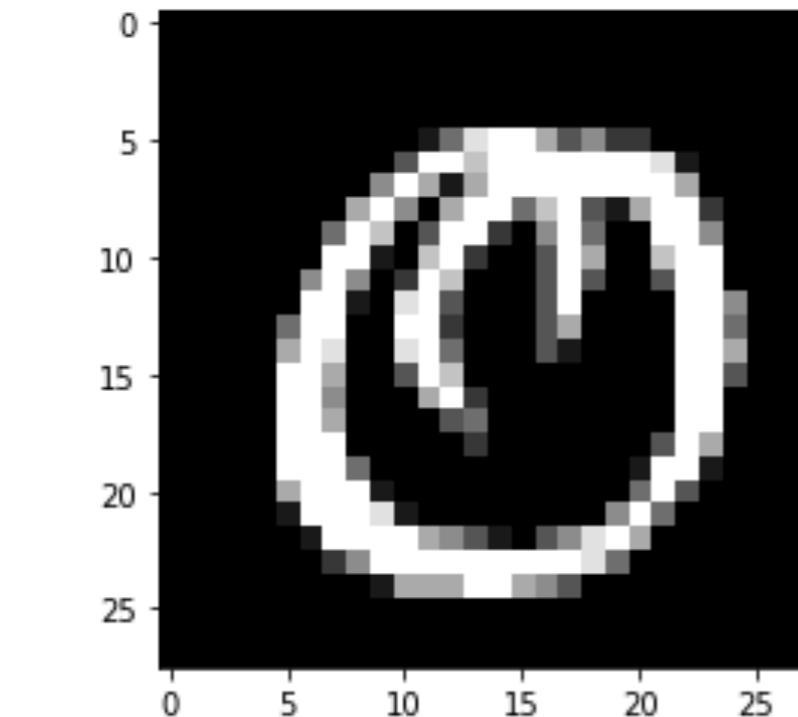
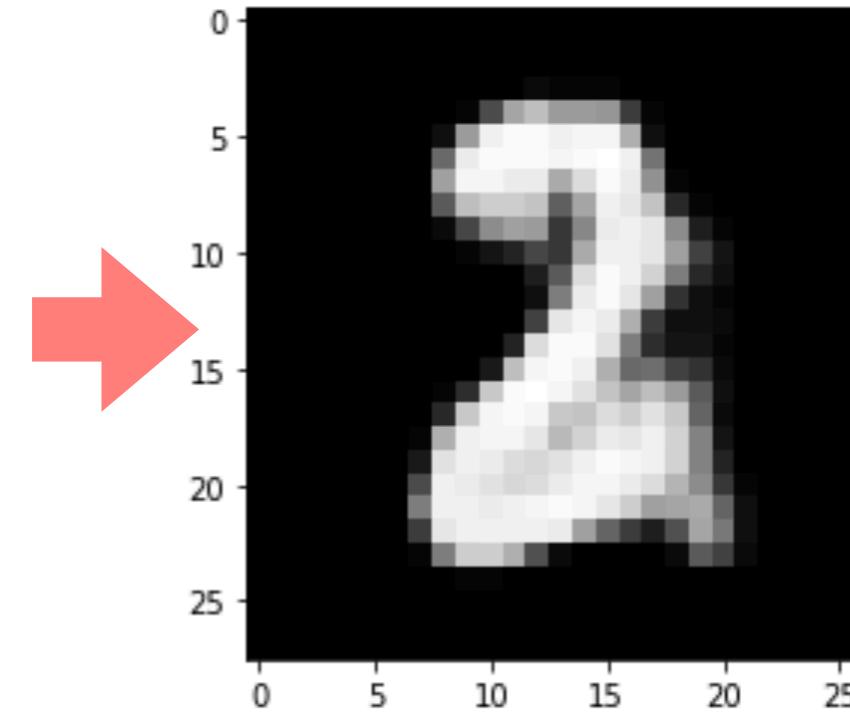
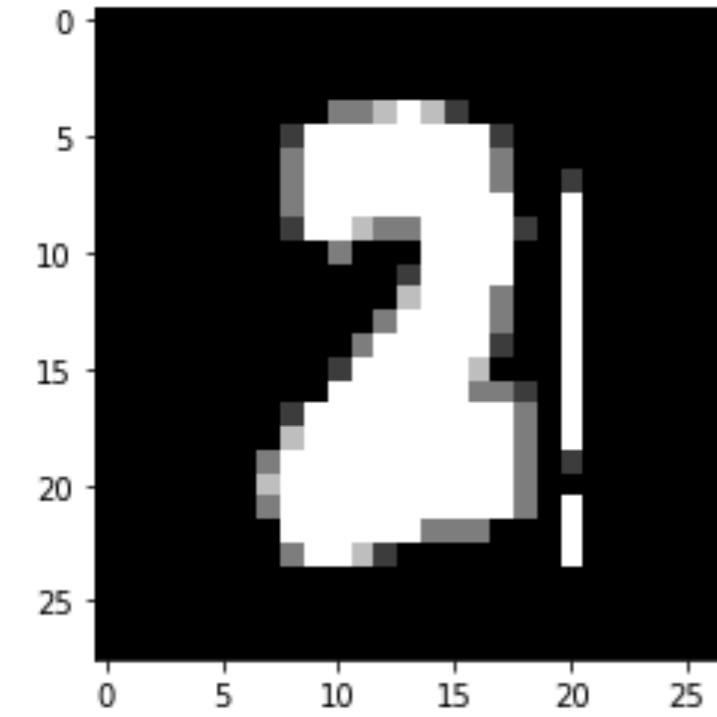
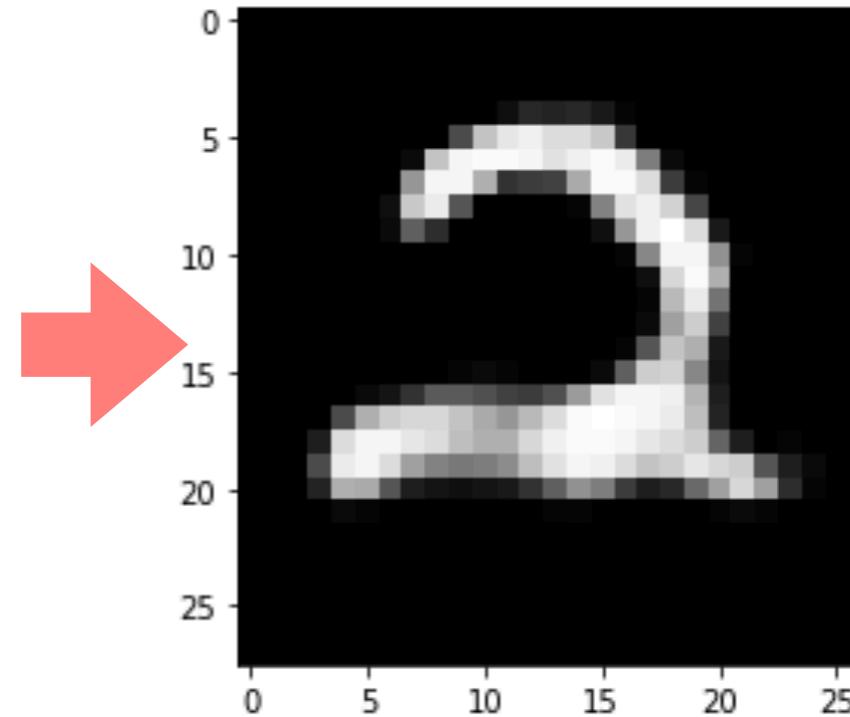
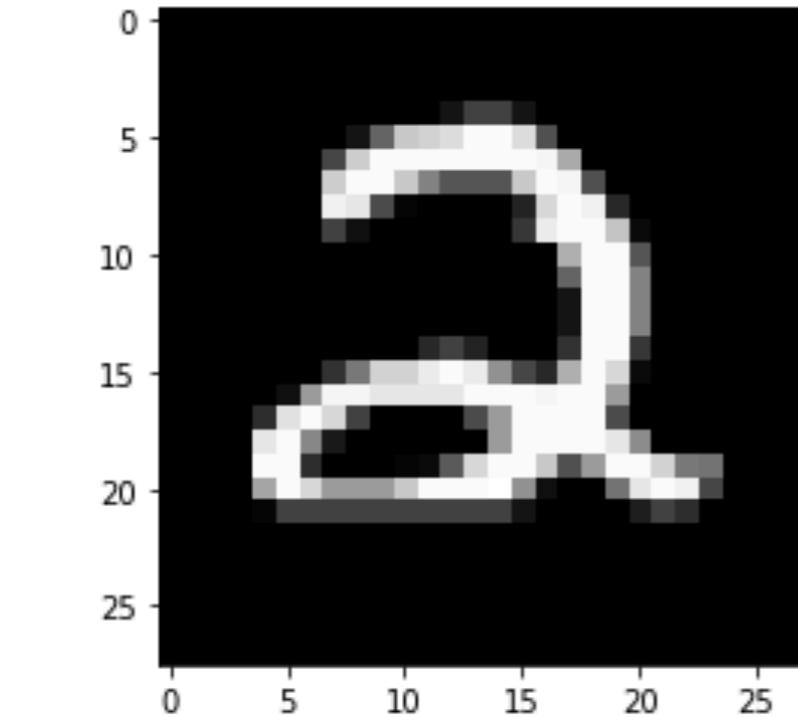
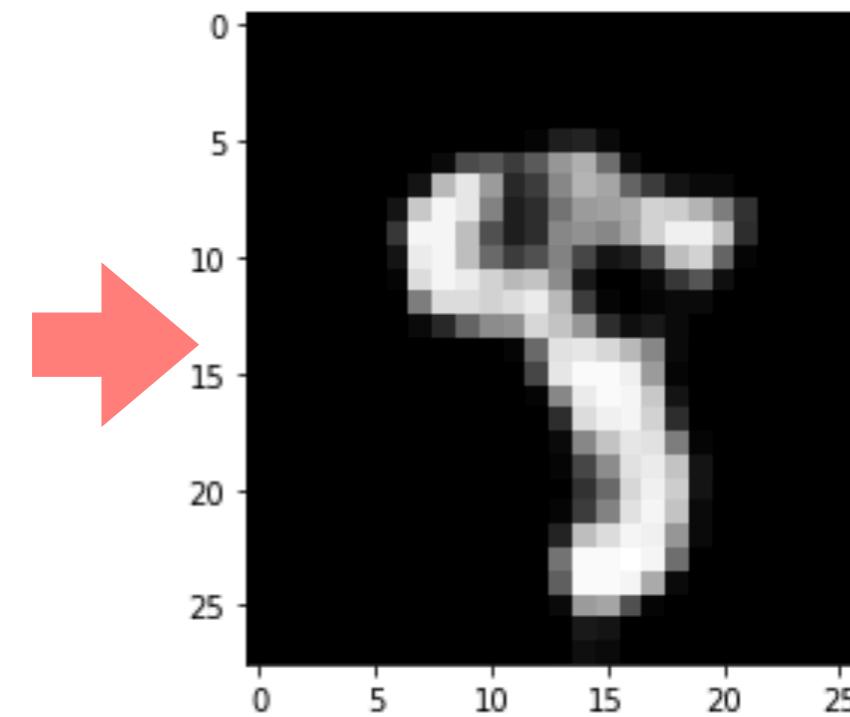
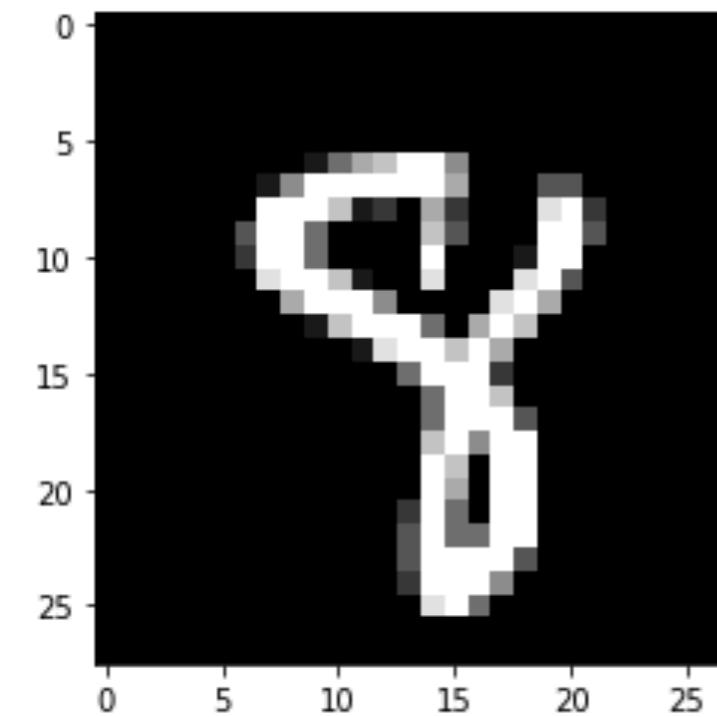
model.add(Conv2D(3, (3, 3), padding='same', activation='sigmoid'))
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 256, 256, 16)	448
max_pooling2d_1 (MaxPooling2D)	(None, 128, 128, 16)	0
conv2d_2 (Conv2D)	(None, 128, 128, 8)	1160
max_pooling2d_2 (MaxPooling2D)	(None, 64, 64, 8)	0
conv2d_3 (Conv2D)	(None, 64, 64, 8)	584
up_sampling2d_1 (UpSampling2D)	(None, 128, 128, 8)	0
conv2d_4 (Conv2D)	(None, 128, 128, 16)	1168
up_sampling2d_2 (UpSampling2D)	(None, 256, 256, 16)	0
conv2d_5 (Conv2D)	(None, 256, 256, 3)	435
Total params: 3,795		
Trainable params: 3,795		
Non-trainable params: 0		

CAE による画像の符号化と復号化



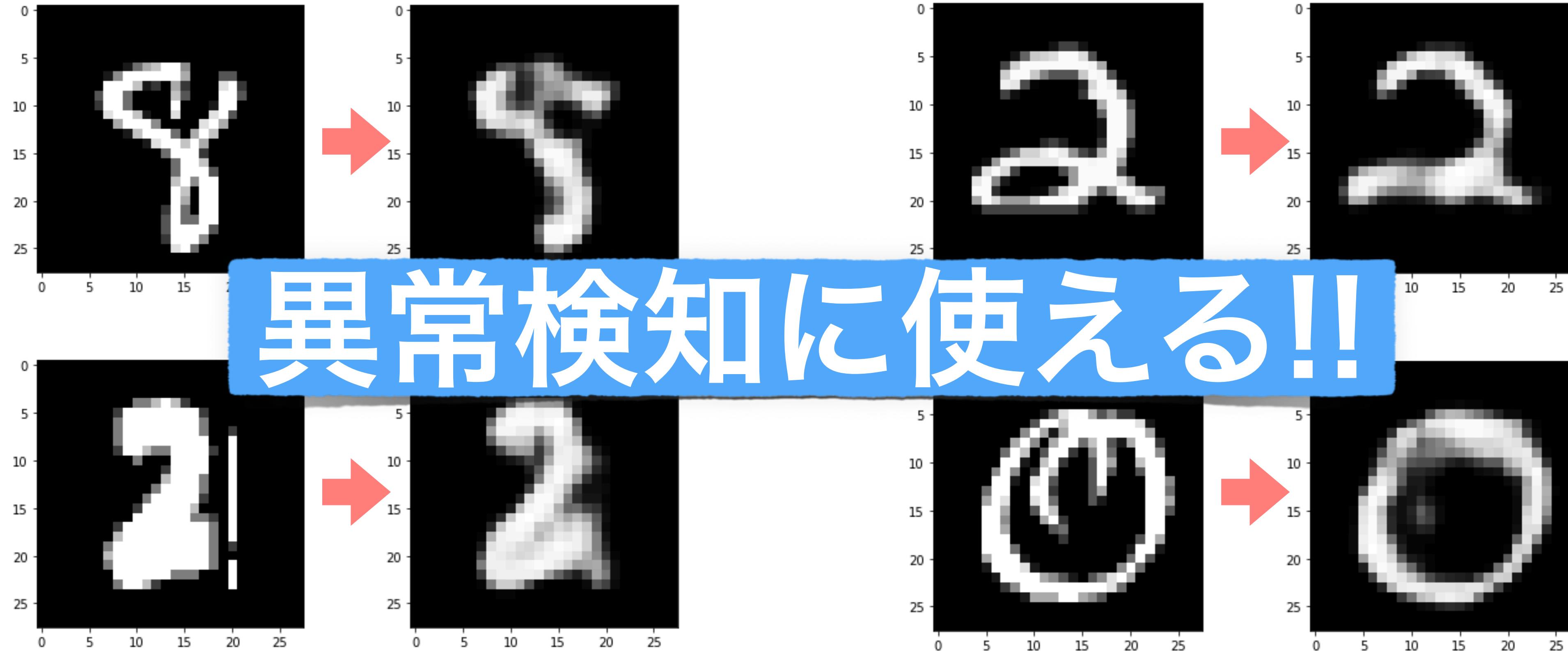
同じモデルでもうまく復号できないケース



訓練データセットと異なる分布のデータはうまく復号できない

ノイズが乗っていたり形が特殊だったり

同じモデルでもうまく復号できないケース



訓練データセットと異なる分布のデータはうまく復号できない

ノイズが乗っていたり形が特殊だったり

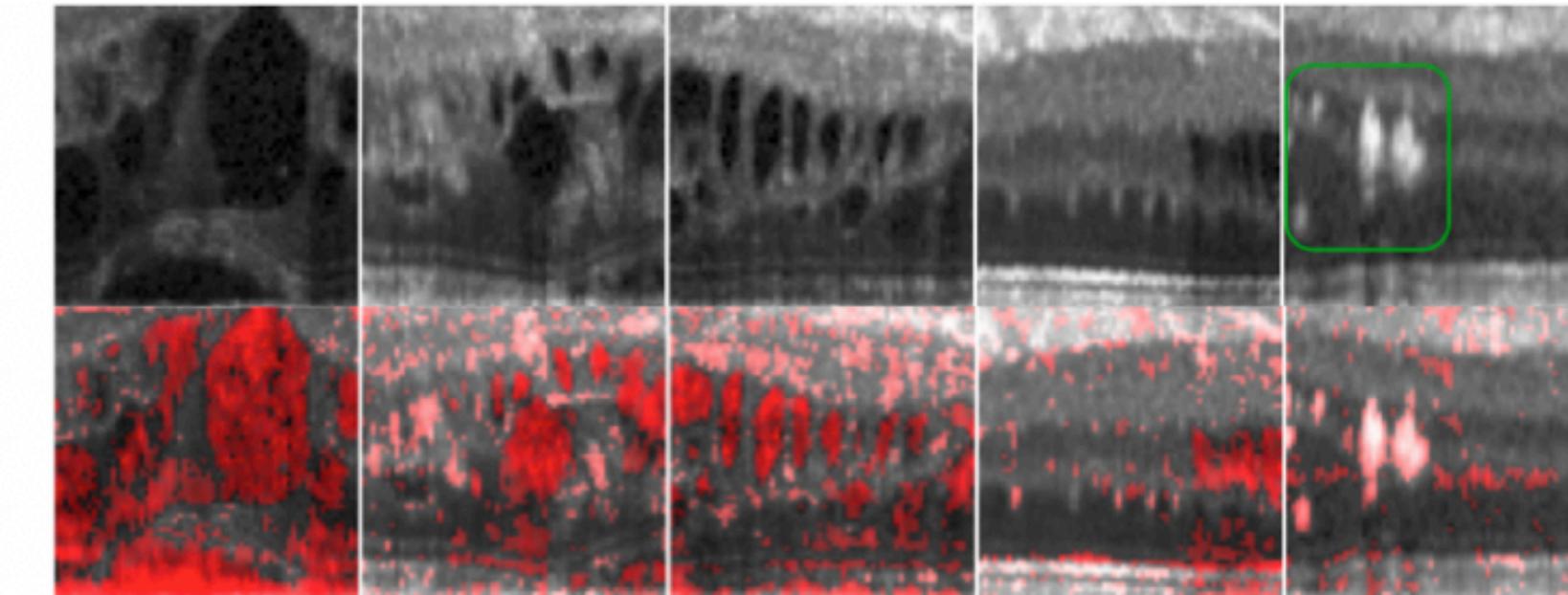
異常検知タスクの例

動画



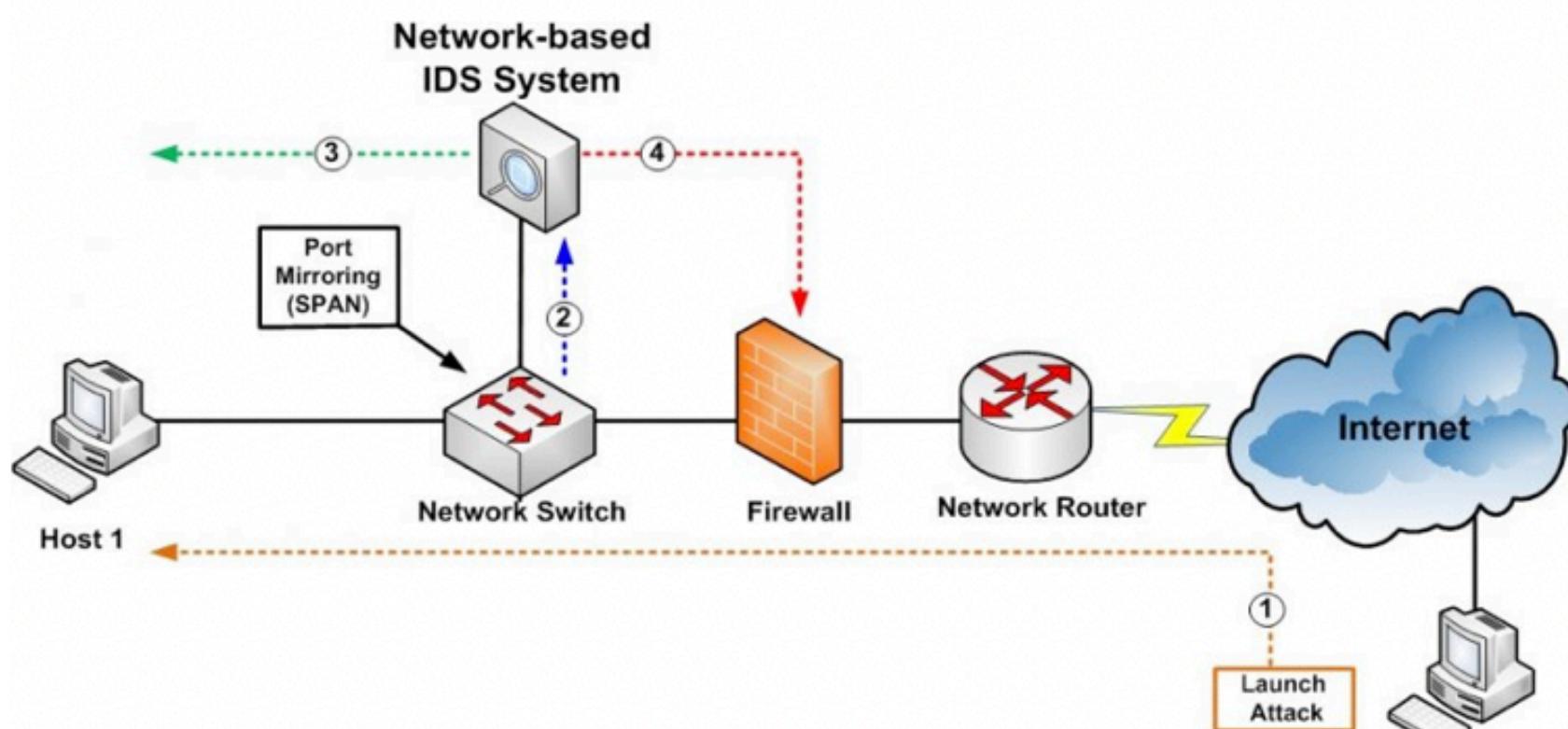
(a) Illegal Traffic Flow detection

画像



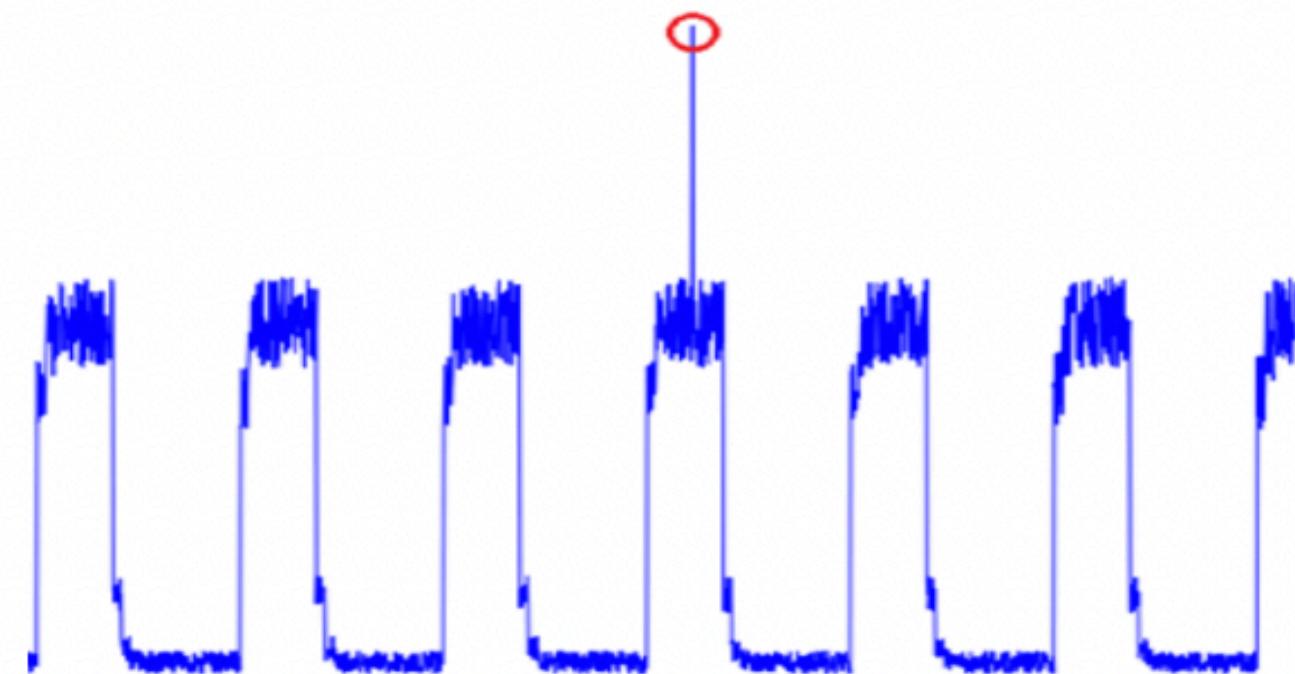
(b) Detecting Retinal Damage

ログ
(テキスト)



(c) Cyber-Network Intrusion detection

時系列



(d) Internet Of Things (IoT) Big-Data
Anomaly detection

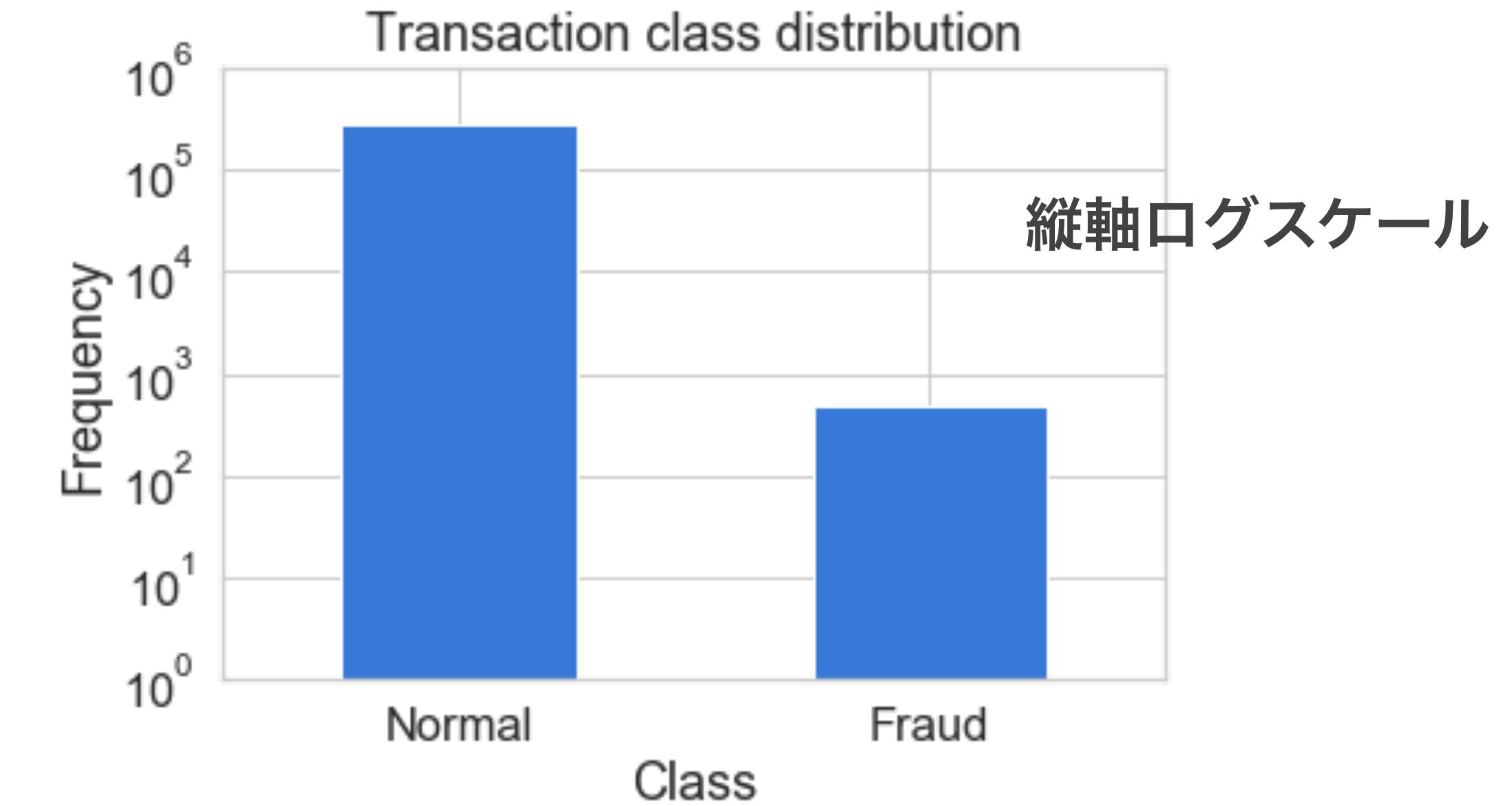
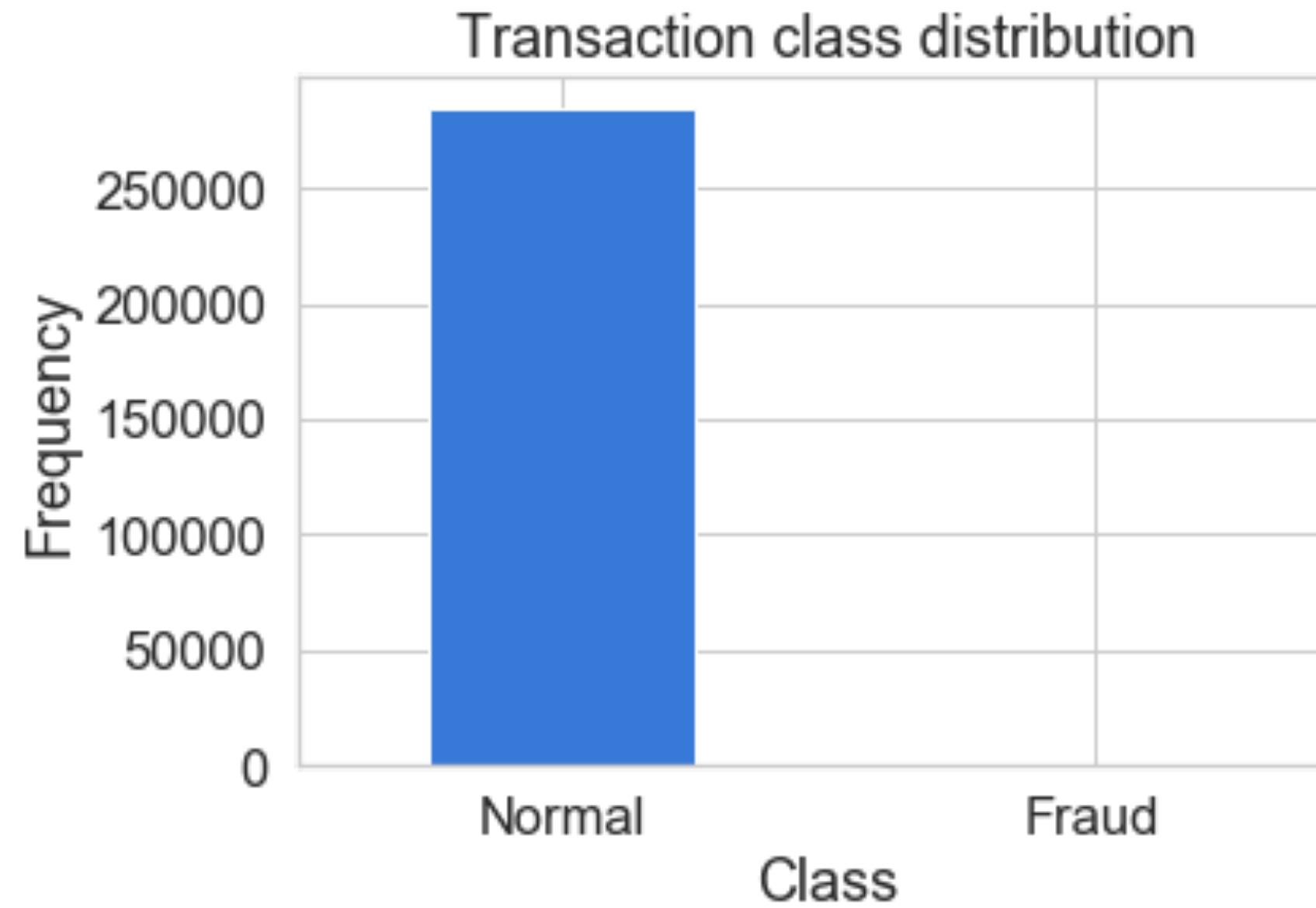
Deep Learning for Anomaly Detection: A Survey

異常検知タスクと教師なし学習

- ▶ 「異常」というくらいなのでデータ数が少ない
 - 一般的に「正常データ数 >> 異常データ数」
- ▶ 分類問題として**教師あり学習**で解くのは難しい
- ▶ **教師なし学習**（もしくは半教師あり学習）で解くことが多い
 - 正常データのみで学習すると異常データの再構成誤差が大きくなることを利用

クレジットカード不正利用検知の例

- ULB (ブリュッセル自由大学) 提供のテーブルデータセット
 - <https://www.kaggle.com/mlg-ulb/creditcardfraud>
- PCAで匿名化された特徴量28個と時間・取引金額・正常/不正の二値ラベル



正常データ (>> 不正データ) のみを用いてオートエンコーダーを学習

オートエンコーダーの実装例

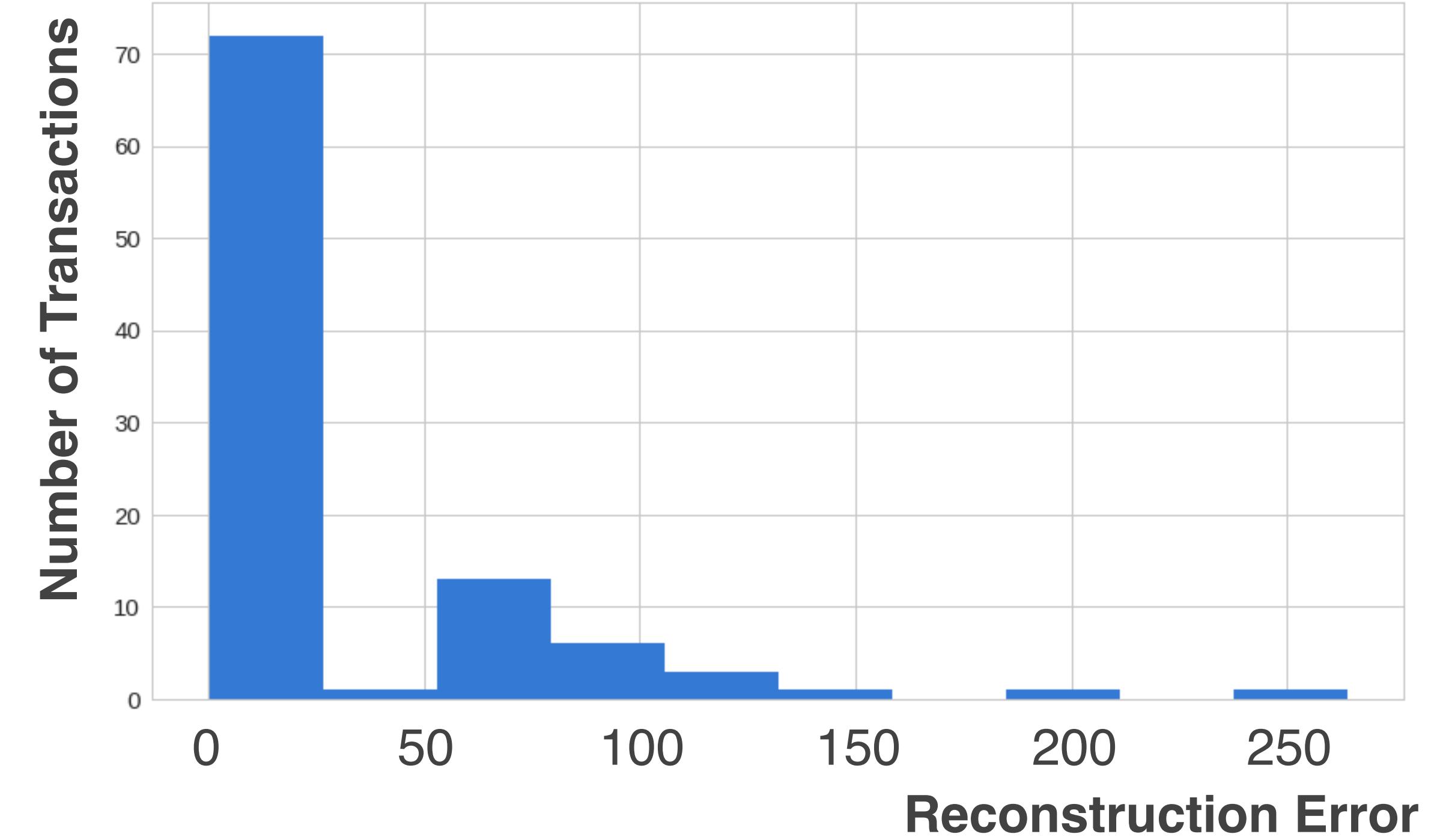
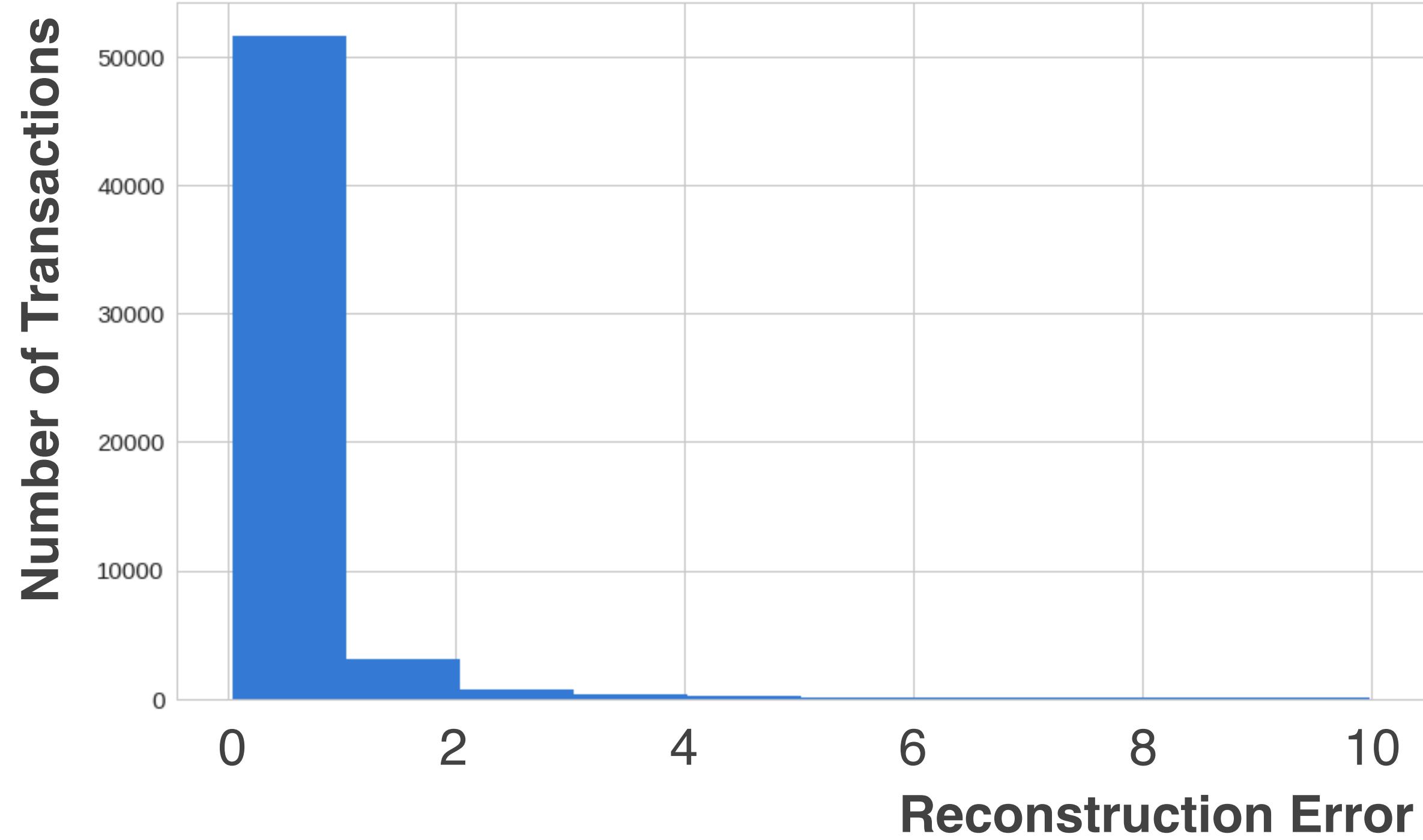
- ▶ 以降3ページ、DATASCIENCE.COMのチュートリアルより拝借
 - <https://www.datascience.com/blog/fraud-detection-with-tensorflow>

```
nb_epoch = 100
batch_size = 128
input_dim = train_x.shape[1] #num of columns, 30
encoding_dim = 14
hidden_dim = int(encoding_dim / 2) #i.e. 7
learning_rate = 1e-7

input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="tanh",
                 activity_regularizer=regularizers.l1(learning_rate))(input_layer)
encoder = Dense(hidden_dim, activation="relu")(encoder)
decoder = Dense(hidden_dim, activation='tanh')(encoder)
decoder = Dense(input_dim, activation='relu')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
```

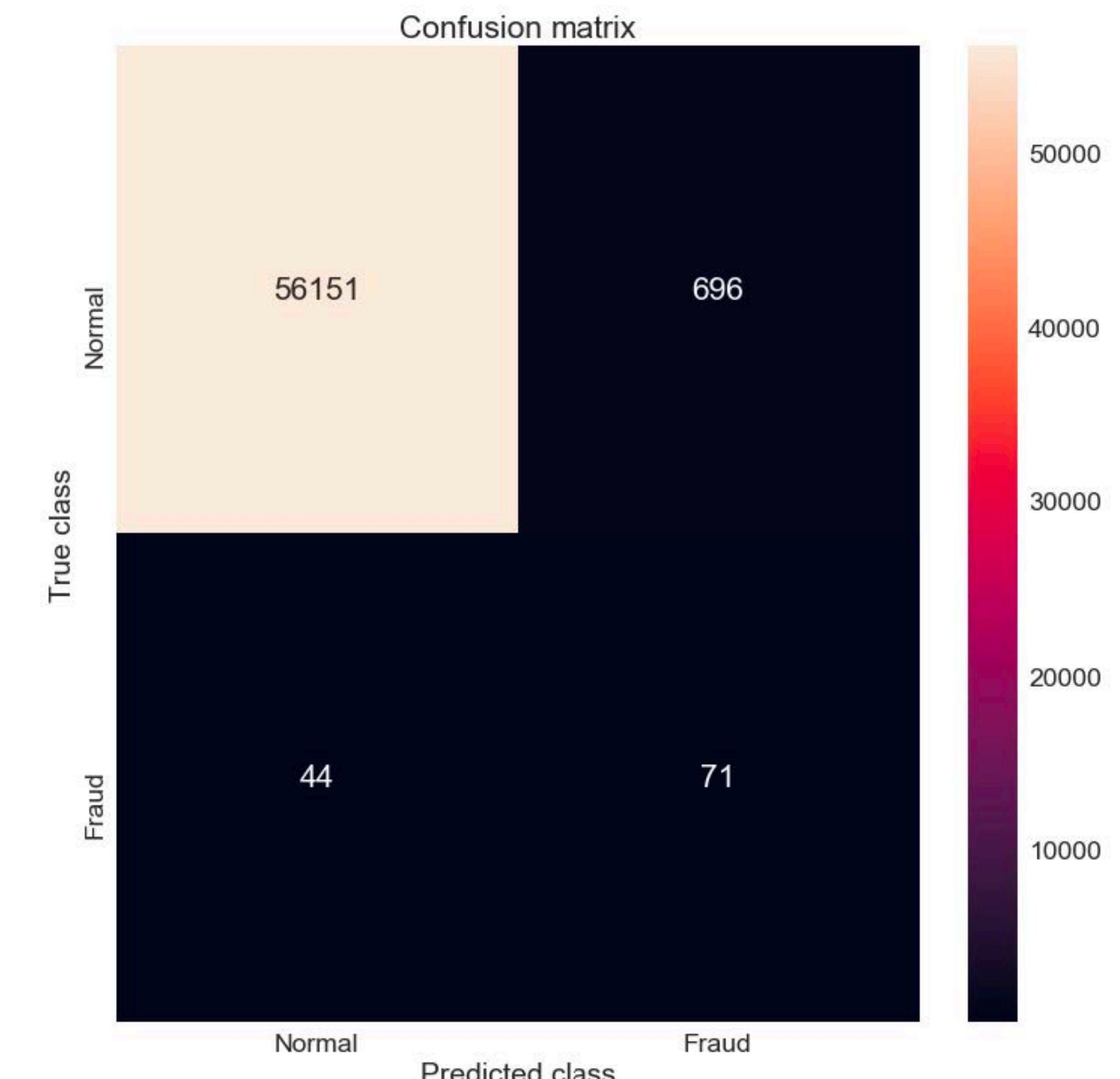
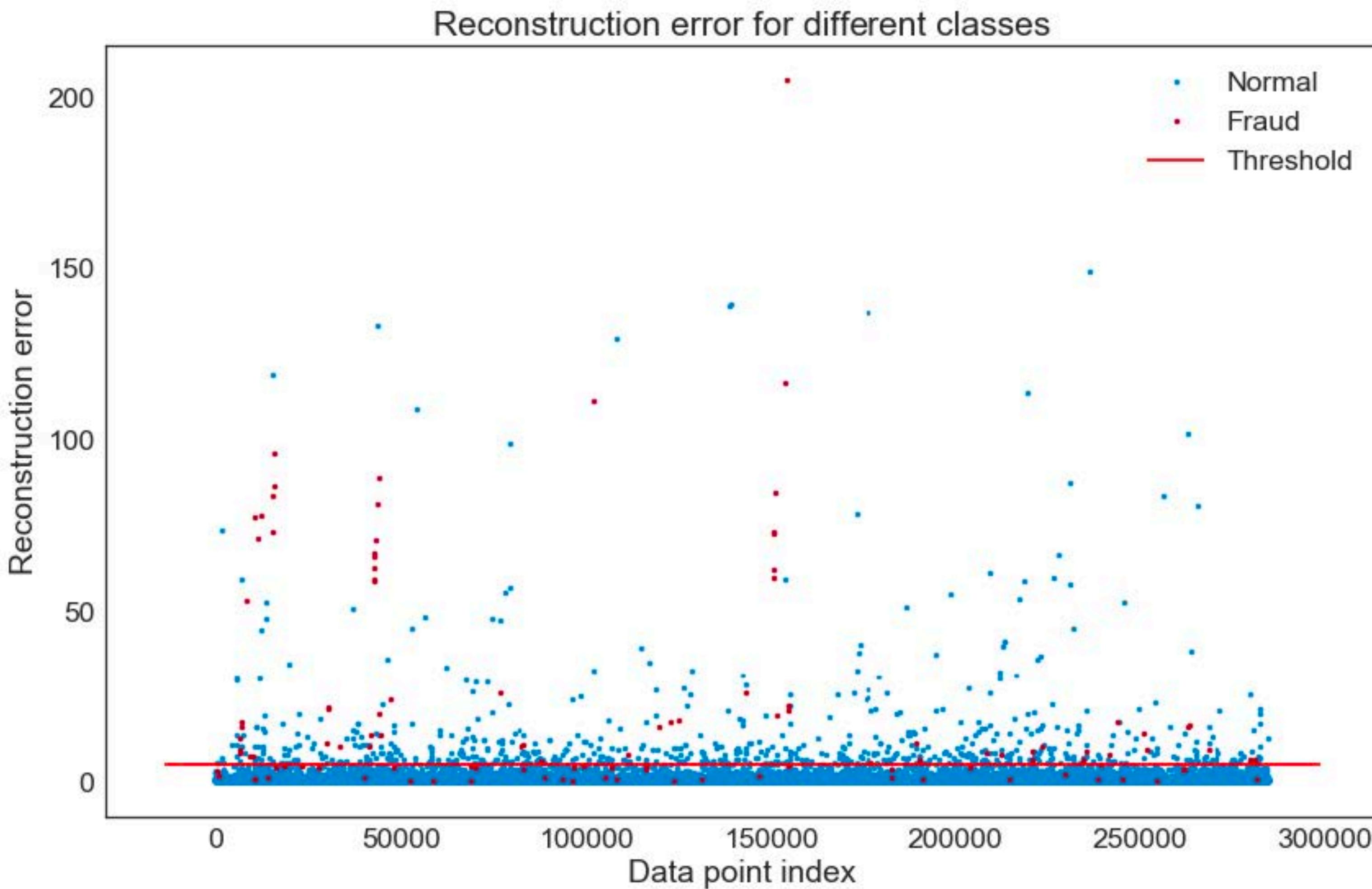
再構成誤差 (MSE) の分布

<https://medium.com/@curiously/credit-card-fraud-detection-using-autoencoders-in-keras-tensorflow-for-hackers-part-vii-20e0c85301bd>

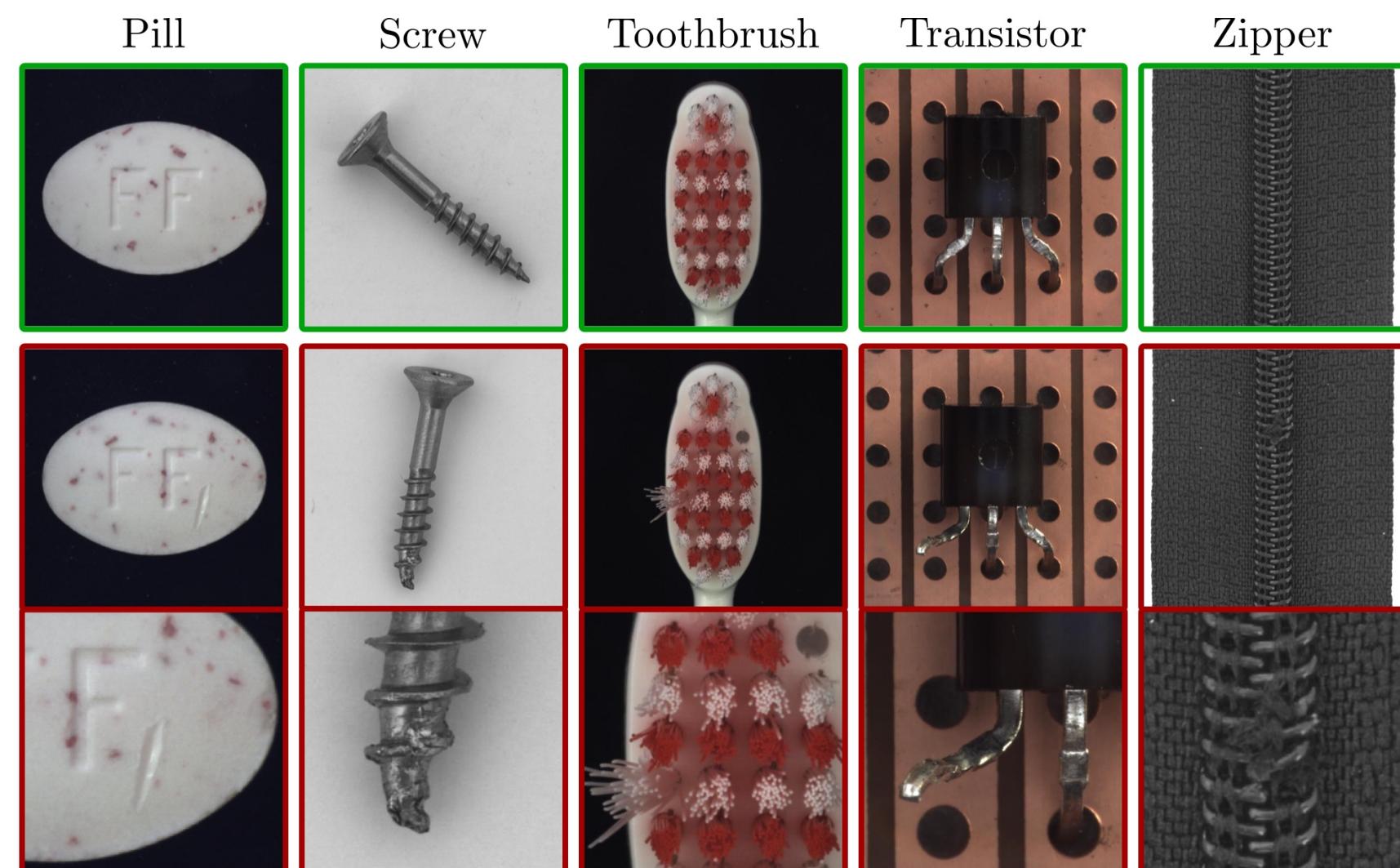
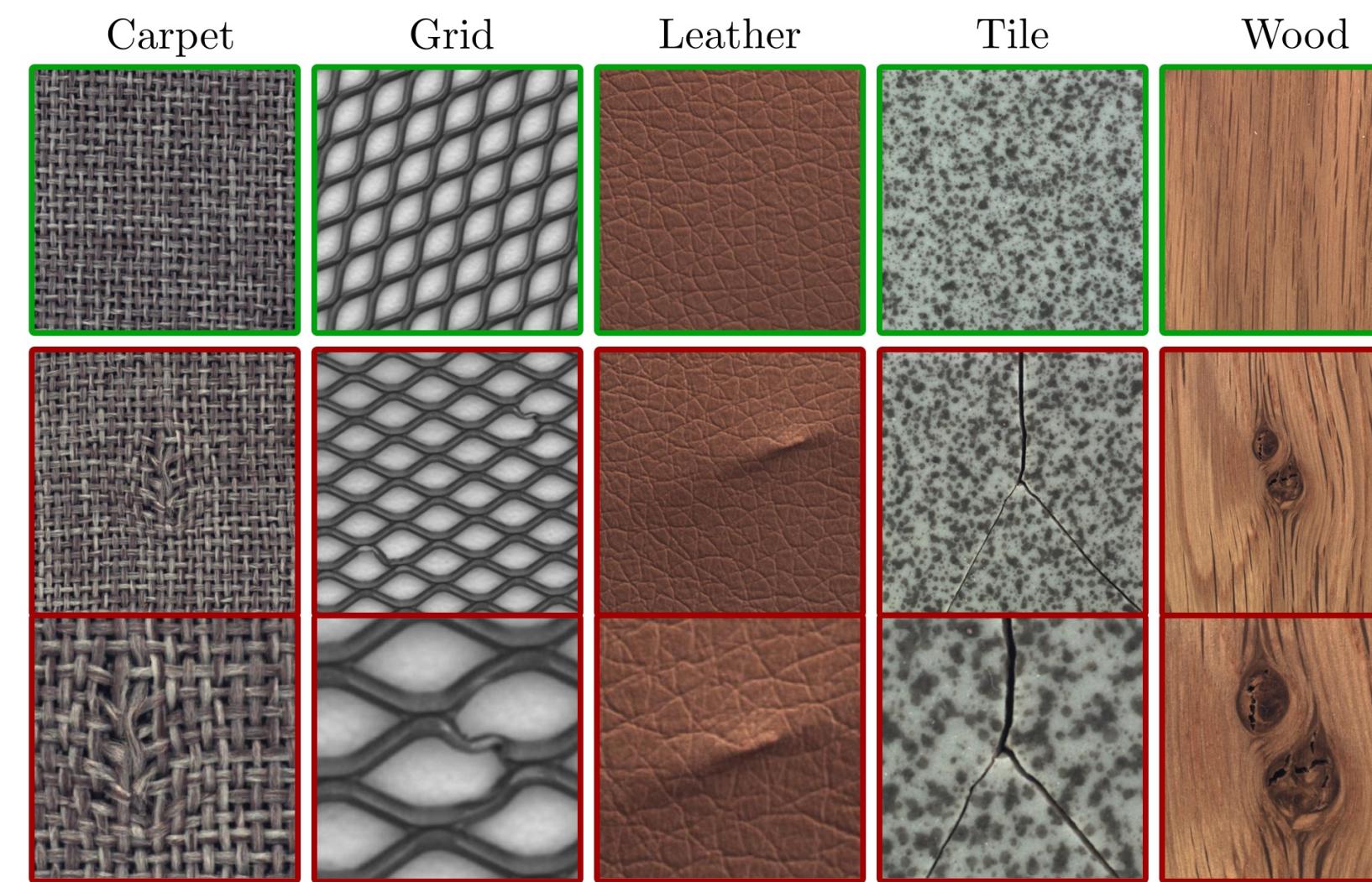
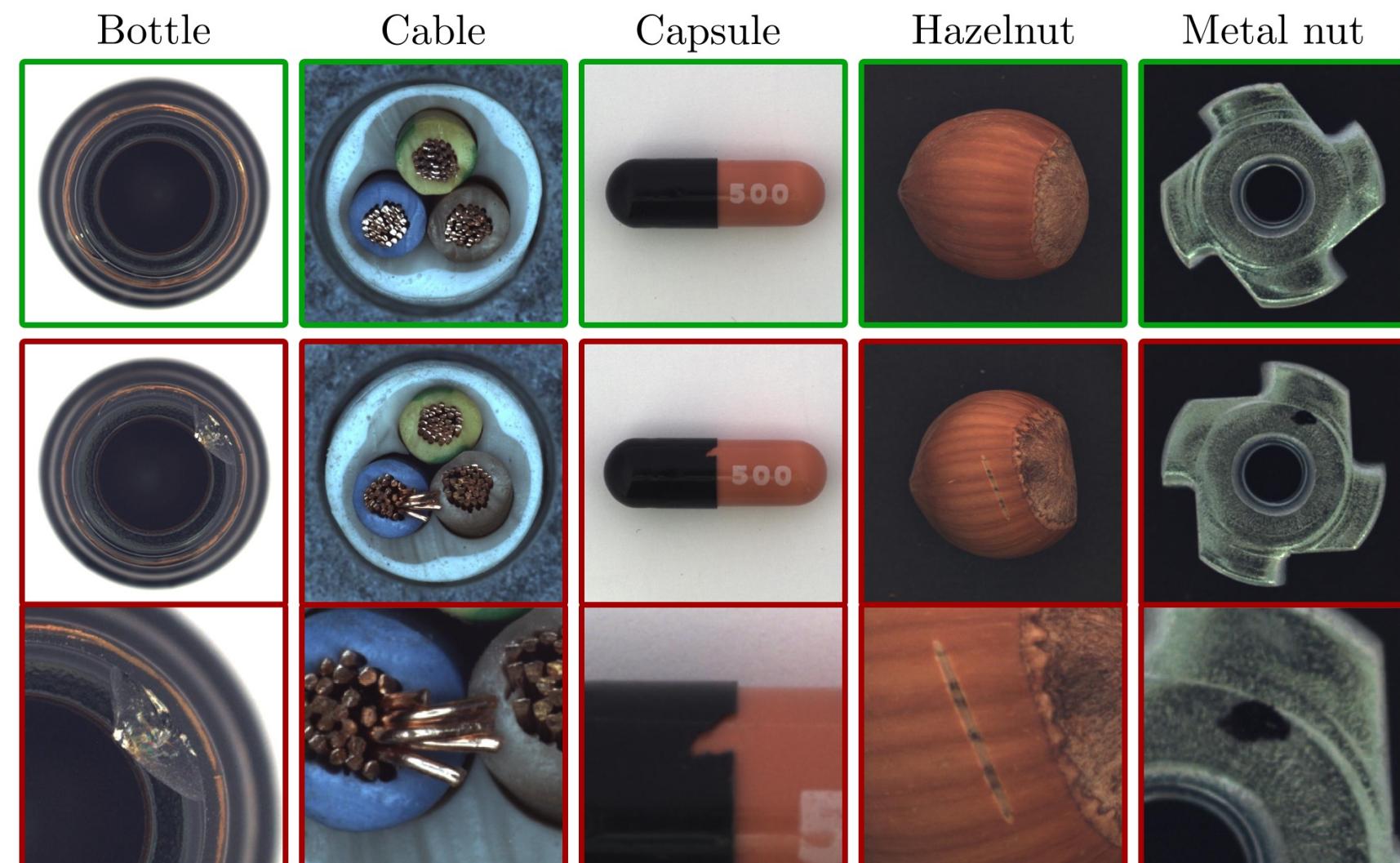


分布形状が優位に異なっている → 閾値を決めて正常/異常判別

各データに対する再構成誤差の散布図と混合行列



MVTec Anomaly Detection Dataset (MVTec AD)



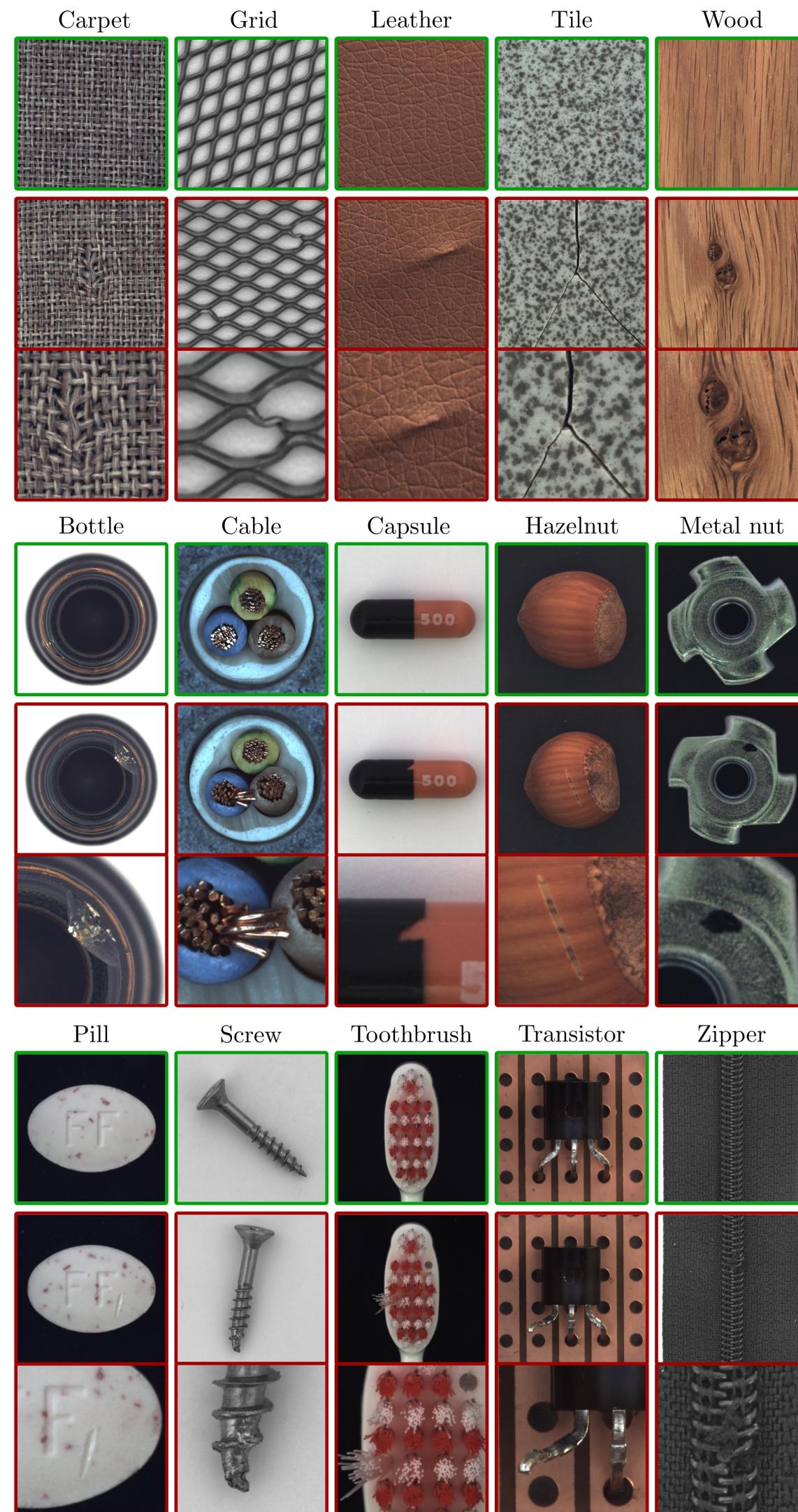
MVTecはドイツ・ミュンヘンの画像処理ソフトウェア開発会社

5,000枚以上の異常検知用画像データセットを公開
15種類の異なる物体とテクスチャの正常/異常データを含む

解説論文はコンピュータビジョンのトップ会議 CVPR 2019 で採択

<https://www.mvtec.com/company/research/datasets/>

MVTec Anomaly Detection Dataset (MVTec AD)



Category	AE (SSIM)	AE (L2)	AnoGAN	CNN Feature Dictionary	Texture Inspection	Variation Model
Textures	Carpet	0.43 0.90	0.57 0.42	0.82 0.16	0.89 0.36	0.57 0.61
	Grid	0.38 1.00	0.57 0.98	0.90 0.12	0.57 0.33	1.00 0.05
	Leather	0.00 0.92	0.06 0.82	0.91 0.12	0.63 0.71	0.00 0.99
	Tile	1.00 0.04	1.00 0.54	0.97 0.05	0.97 0.44	1.00 0.43
	Wood	0.84 0.82	1.00 0.47	0.89 0.47	0.79 0.88	0.42 1.00
	Bottle	0.85 0.90	0.70 0.89	0.95 0.43	1.00 0.06	- 1.00
	Cable	0.74 0.48	0.93 0.18	0.98 0.07	0.97 0.24	- -
	Capsule	0.78 0.43	1.00 0.24	0.96 0.20	0.78 0.03	- 0.03
	Hazelnut	1.00 0.07	0.93 0.84	0.83 0.16	0.90 0.07	- -
	Metal nut	1.00 0.08	0.68 0.77	0.86 0.13	0.55 0.74	- 0.32
Objects	Pill	0.92 0.28	1.00 0.23	1.00 0.24	0.85 0.06	- 1.00
	Screw	0.95 0.06	0.98 0.39	0.41 0.28	0.73 0.13	- 1.00
	Toothbrush	0.75 0.73	1.00 0.97	1.00 0.13	1.00 0.03	- 0.60
	Transistor	1.00 0.03	0.97 0.45	0.98 0.35	1.00 0.15	- -
	Zipper	1.00 0.60	0.97 0.63	0.78 0.40	0.78 0.29	- -

上段: 正常データの正解率
下段: 異常データの正解率

全体的にオートエンコーダーを用いたモデルが強い

AE(L2) と AE(SSIM)

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, x_train, epochs=16, batch_size=128)
```

損失関数は必ず自乗誤差でなくてはいけないわけでもない

L2誤差 (自乗誤差)

$$L_2(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{r=0}^{h-1} \sum_{c=0}^{w-1} (\mathbf{x}(r, c) - \hat{\mathbf{x}}(r, c))^2$$



<https://dftalk.jp/?p=18111>

SSIM (Structual SIMilarity) index [2004]

<http://visualize.hatenablog.com/entry/2016/02/20/144657>

$$\text{SSIM}(\mathbf{p}, \mathbf{q}) = \frac{(2\mu_{\mathbf{p}}\mu_{\mathbf{q}} + c_1)(2\sigma_{\mathbf{pq}} + c_2)}{(\mu_{\mathbf{p}}^2 + \mu_{\mathbf{q}}^2 + c_1)(\sigma_{\mathbf{p}}^2 + \sigma_{\mathbf{q}}^2 + c_2)}$$

人間が感じる違い

- ・画素値(輝度値)の変化
- ・コントラストの変化
- ・構造の変化

が符号化前後でどれくらい変化したかを表す指標
(画像を小さいWindowに分けて計算)

AnoGAN [2017]

<https://arxiv.org/abs/1703.05921>

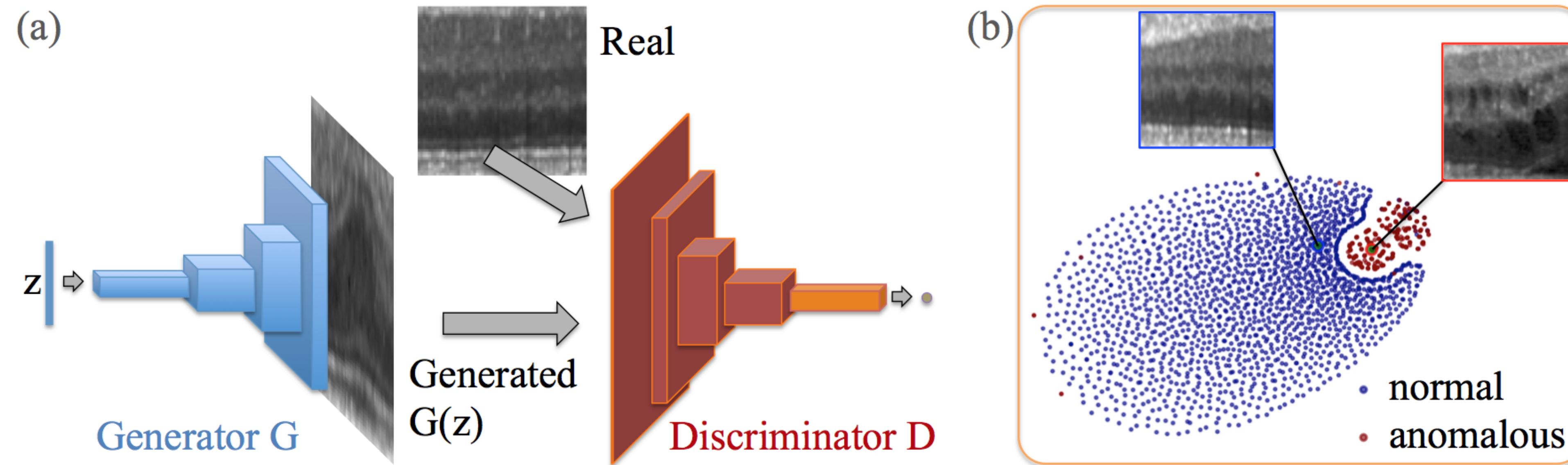
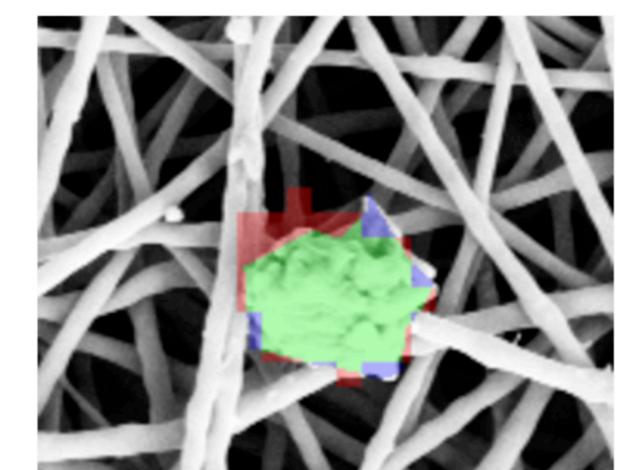


Fig. 2. (a) Deep convolutional generative adversarial network. (b) t-SNE embedding of normal (blue) and anomalous (red) images on the feature representation of the last convolution layer (orange in (a)) of the discriminator.

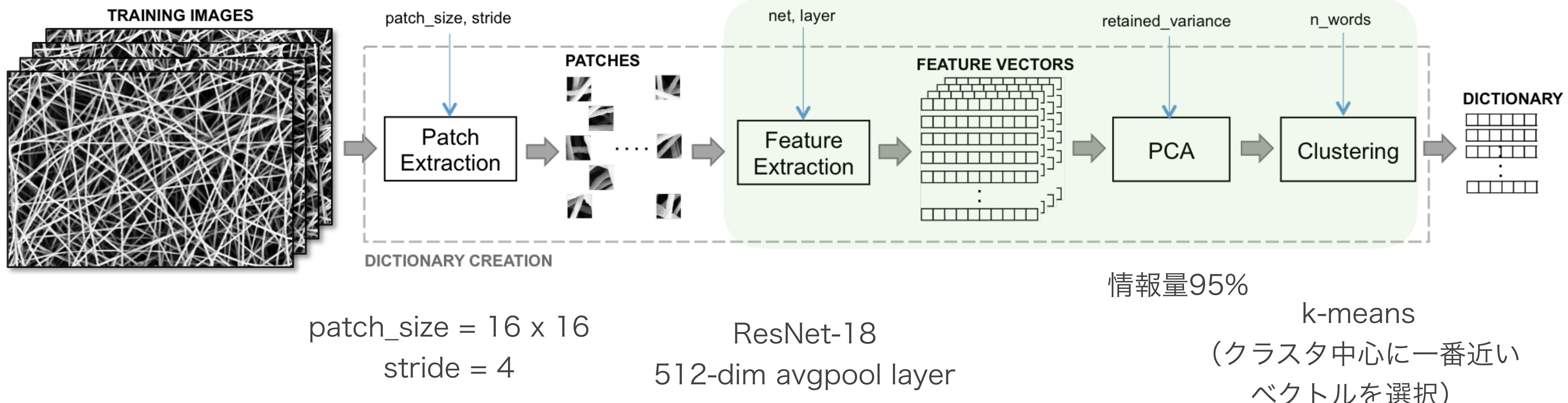
GANを正常データで訓練し、正常データの生成モデル（確率分布）を構築

CNN Feature Dictionary [2018]

<https://www.ncbi.nlm.nih.gov/pubmed/29329268>

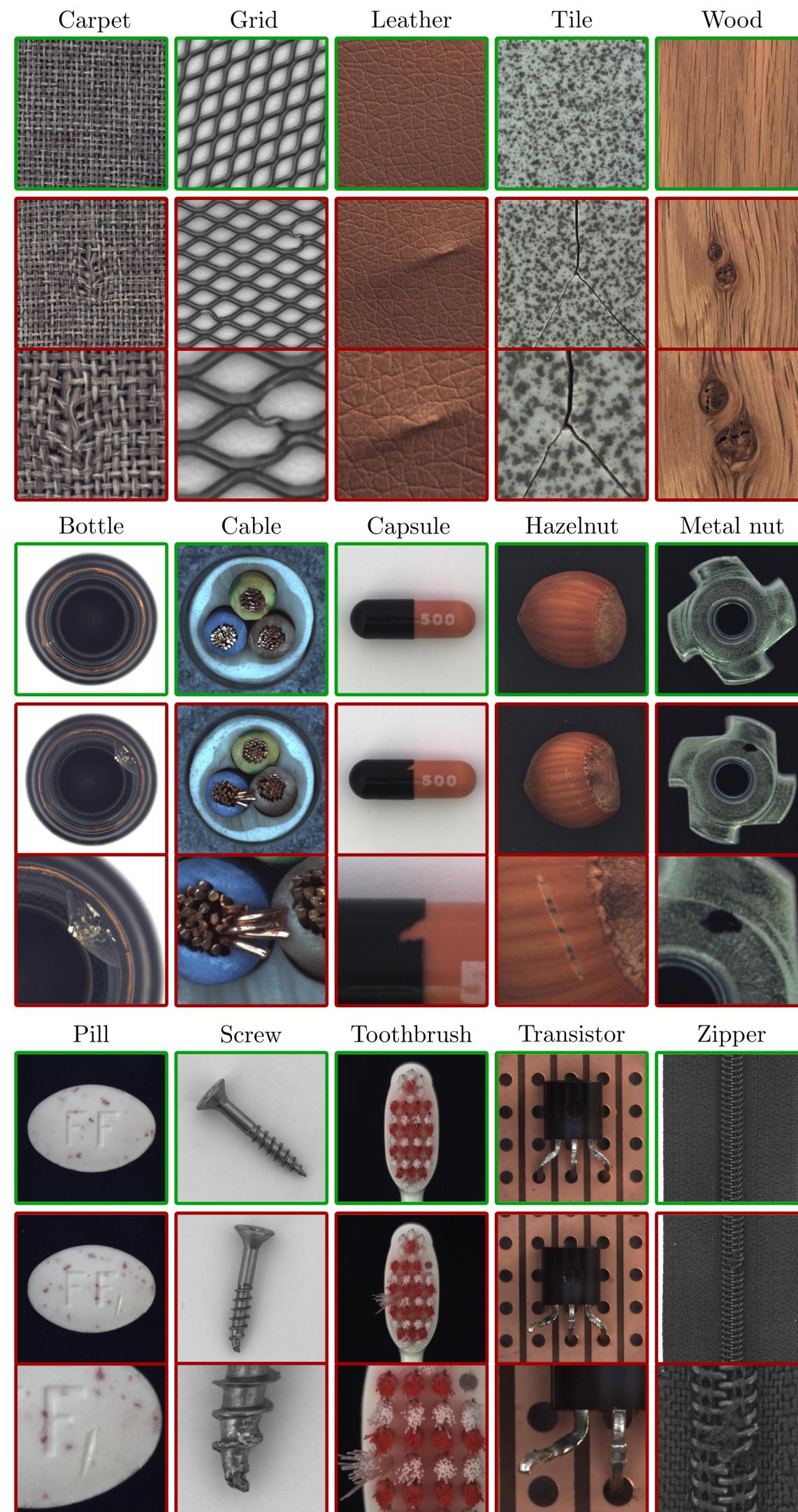


次元削減を行なっている部分



特徴量ベクトルのディクショナリの中で互いの距離がある閾値より遠いベクトルを異常領域とする

MVTec Anomaly Detection Dataset (MVTec AD)



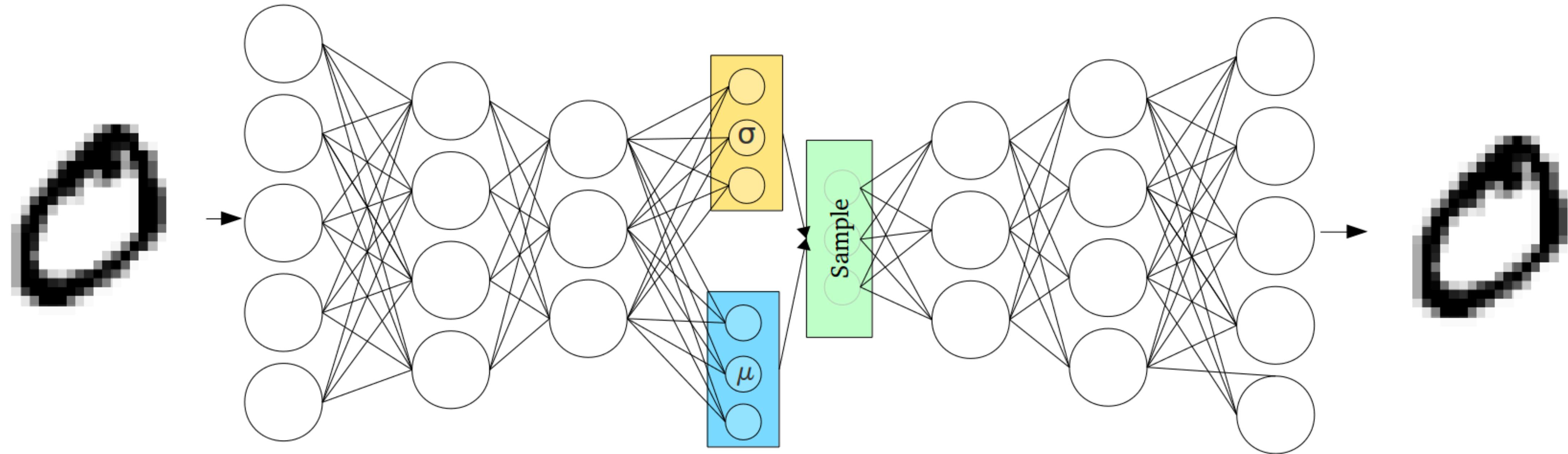
Category	AE (SSIM)	AE (L2)	AnoGAN	CNN Feature Dictionary	Texture Inspection	Variation Model
Textures	Carpet	0.43 0.90	0.57 0.42	0.82 0.16	0.89 0.36	0.57 0.61
	Grid	0.38 1.00	0.57 0.98	0.90 0.12	0.57 0.33	1.00 0.05
	Leather	0.00 0.92	0.06 0.82	0.91 0.12	0.63 0.71	0.00 0.99
	Tile	1.00 0.04	1.00 0.54	0.97 0.05	0.97 0.44	1.00 0.43
	Wood	0.84 0.82	1.00 0.47	0.89 0.47	0.79 0.88	0.42 1.00
	Bottle	0.85 0.90	0.70 0.89	0.95 0.43	1.00 0.06	- 1.00
	Cable	0.74 0.48	0.93 0.18	0.98 0.07	0.97 0.24	- -
	Capsule	0.78 0.43	1.00 0.24	0.96 0.20	0.78 0.03	- 0.03
	Hazelnut	1.00 0.07	0.93 0.84	0.83 0.16	0.90 0.07	- -
	Metal nut	1.00 0.08	0.68 0.77	0.86 0.13	0.55 0.74	- 0.32
Objects	Pill	0.92 0.28	1.00 0.23	1.00 0.24	0.85 0.06	- 1.00
	Screw	0.95 0.06	0.98 0.39	0.41 0.28	0.73 0.13	- 1.00
	Toothbrush	0.75 0.73	1.00 0.97	1.00 0.13	1.00 0.03	- 0.60
	Transistor	1.00 0.03	0.97 0.45	0.98 0.35	1.00 0.15	- -
	Zipper	1.00 0.60	0.97 0.63	0.78 0.40	0.78 0.29	- -

上段: 正常データの正解率
下段: 異常データの正解率

全体的にオートエンコーダーを用いたモデルが強い

まとめ

- ▶ オートエンコーダーは入力と目的の出力に同じデータを用いてデータの効率的な「表現」を学習
- ▶ Max Pooling - Up Sampling を用いて畳み込み層を用いたオートエンコーダーも実現できる
- ▶ 再構成誤差を指標に教師なし学習で異常検知を行うことができる
- ▶ 「再構成誤差」の定義は色々ある
- ▶ MVTec AD データセットは実用的に面白そう



```

# VAE model = encoder + decoder
# build encoder model
inputs = Input(shape=input_shape, name='encoder_input')
x = Dense(intermediate_dim, activation='relu')(inputs)
z_mean = Dense(latent_dim, name='z_mean')(x)
z_log_var = Dense(latent_dim, name='z_log_var')(x)

# use reparameterization trick to push the sampling out as input
# note that "output_shape" isn't necessary with the TensorFlow backend
z = Lambda(sampling, output_shape=(latent_dim,), name='z')([z_mean, z_log_var])

# instantiate encoder model
encoder = Model(inputs, [z_mean, z_log_var, z], name='encoder')
encoder.summary()
plot_model(encoder, to_file='vae_mlp_encoder.png', show_shapes=True)

# build decoder model
latent_inputs = Input(shape=(latent_dim,), name='z_sampling')
x = Dense(intermediate_dim, activation='relu')(latent_inputs)
outputs = Dense(original_dim, activation='sigmoid')(x)

# instantiate decoder model
decoder = Model(latent_inputs, outputs, name='decoder')
decoder.summary()
plot_model(decoder, to_file='vae_mlp_decoder.png', show_shapes=True)

# instantiate VAE model
outputs = decoder(encoder(inputs)[2])
vae = Model(inputs, outputs, name='vae_mlp')

```

異常検知とは

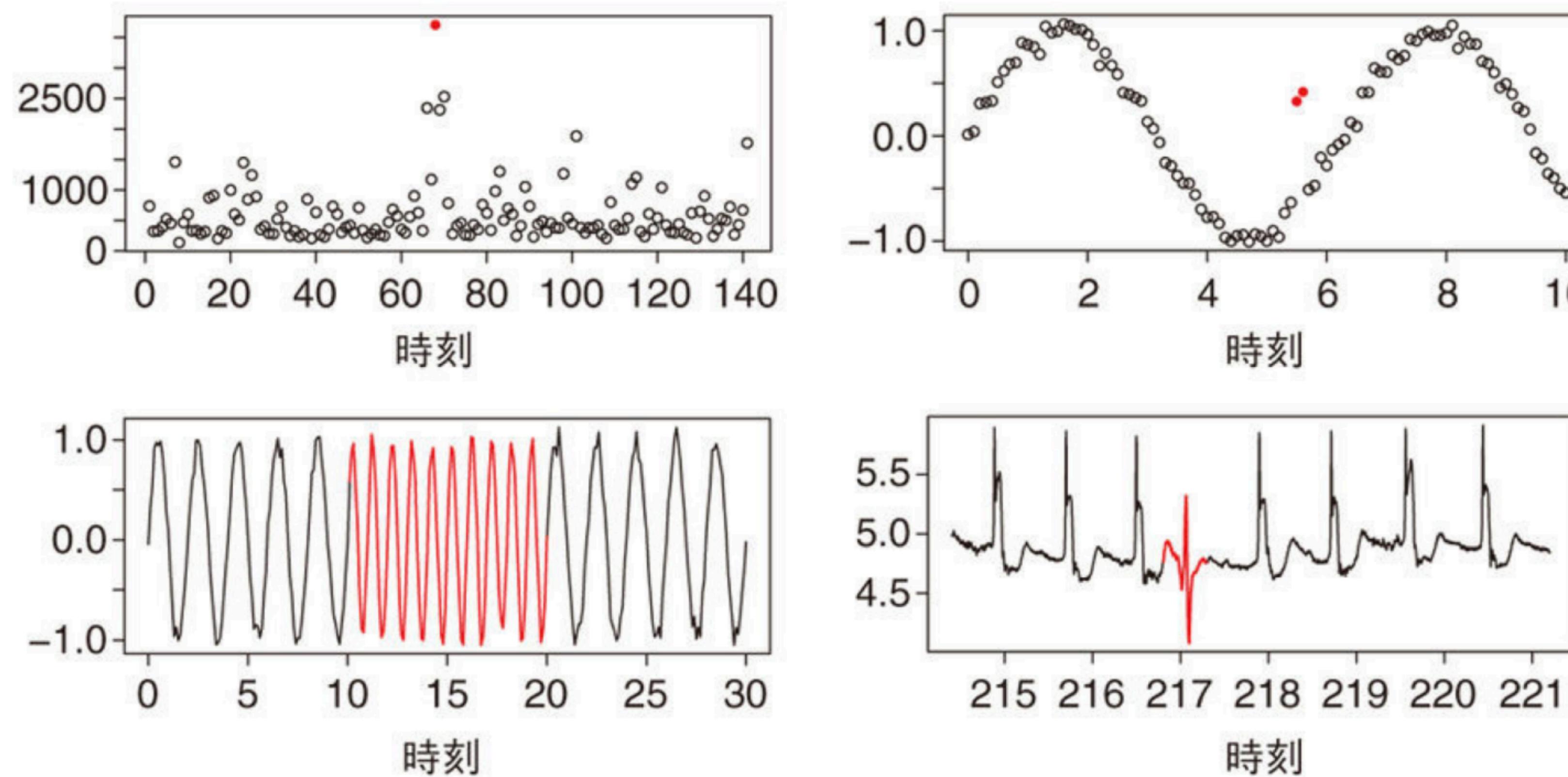


図 1.1 時系列データのさまざまな異常の例. 上段: 外れ値 (左), 時系列的外れ値 (右). 下段: 変化点 (左: 周波数変化データ), 変化点または異常部位 (右: 心電図データ).