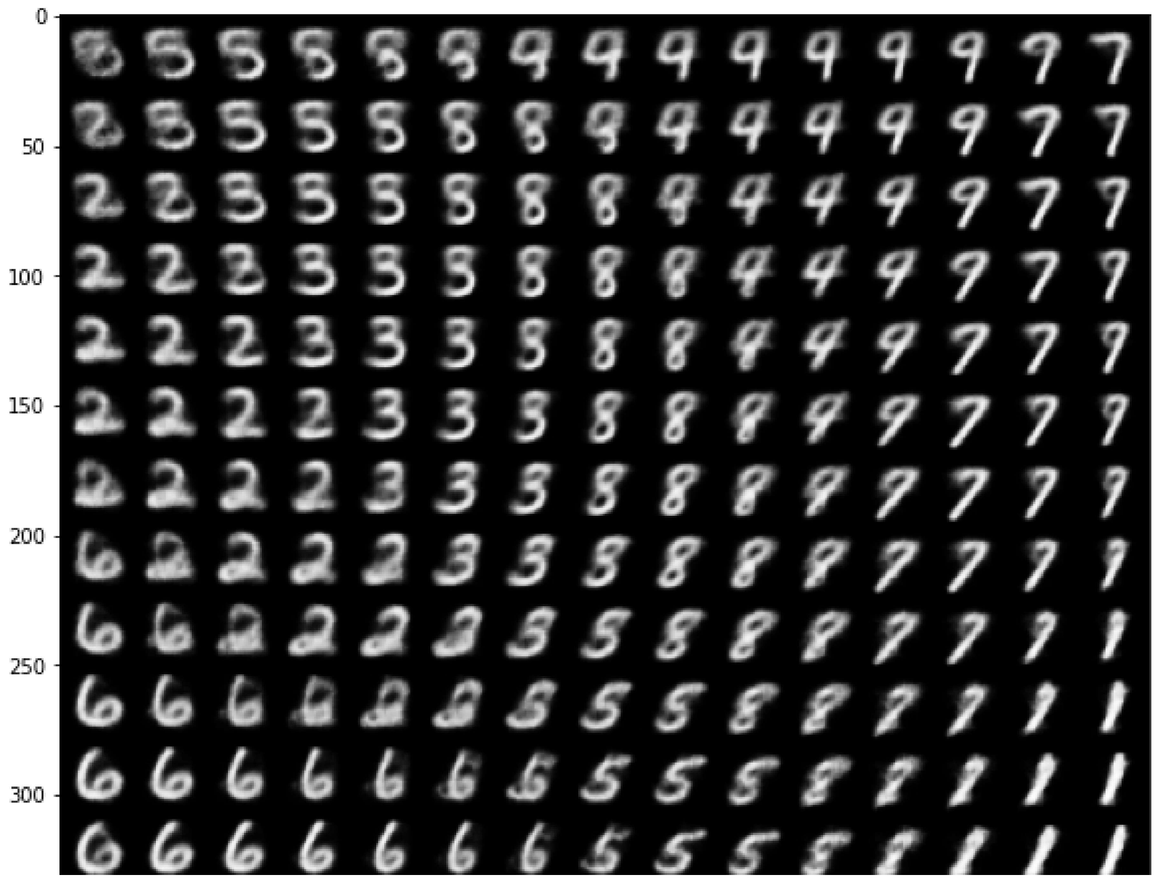
```
”””
既存の画像を変分オートエンコーダ（VAE）という手法で学習させて、
全く新しい画像を生成してもらう
”””

import keras
from keras import layers
from keras import backend as K
import numpy as np
from keras.models import Model
from keras.datasets import mnist
from scipy.stats import norm
import matplotlib.pyplot as plt

#  VAEエンコーダネットワーク
img_shape = (28, 28, 1)
latent_dim = 2
batch_size = 16

input_img = keras.Input(shape=img_shape)
x = layers.Conv2D(32, 3, padding='same', activation='relu')(input_img)
x = layers.Conv2D(64, 3, padding='same', activation='relu', strides=(2,2))(x)
x = layers.Conv2D(64, 3, padding='same', activation='relu')(x)
x = layers.Conv2D(64, 3, padding='same', activation='relu')(x)

shape_before_flattening = K.int_shape(x)

x = layers.Flatten()(x)
x = layers.Dense(32, activation='relu')(x)

z_mean = layers.Dense(latent_dim)(x)
z_log_var = layers.Dense(latent_dim)(x)

#  潜在空間サンプリング関数
def sampling(args):
    z_mean, z_log_var = args
```

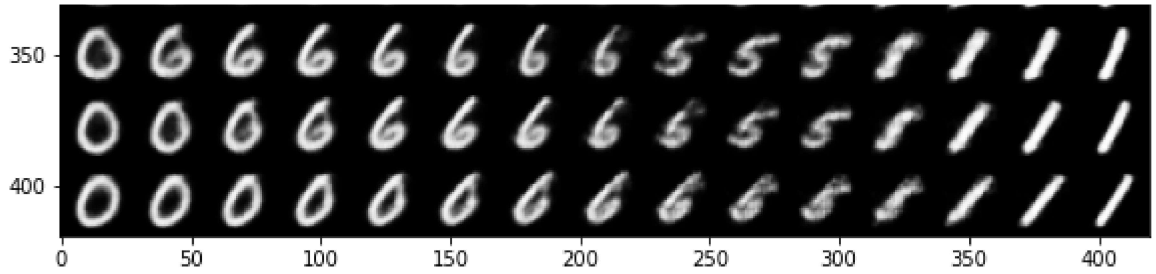
```
    epsilon = K.random_normal(shape=(K.shape(z_mean)[0], latent_dim),
                              mean=0, stddev=1)
    return z_mean + K.exp(z_log_var) * epsilon

z = layers.Lambda(sampling)([z_mean, z_log_var])

# 潜在空間の点を画像にマッピングするVAEデコーダネットワーク
decoder_input = layers.Input(K.int_shape(z)[1:])

x = layers.Dense(np.prod(shape_before_flattening[1:]), activation='relu')(decoder_input)
x = layers.Reshape(shape_before_flattening[1:])(x)
x = layers.Conv2DTranspose(32, 3, padding='same', activation='relu', strides=(2,2))(x)
x = layers.Conv2D(1, 3, padding='same', activation='sigmoid')(x)

decoder = Model(decoder_input, x)

z_decoded = decoder(z)

# VAEの損失関数を計算するためのカスタム層
class CustomVariationalLayer(keras.layers.Layer):
    def vae_loss(self, x, z_decoded):
        x = K.flatten(x)
        z_decoded = K.flatten(z_decoded)
        xent_loss = keras.metrics.binary_crossentropy(x, z_decoded)
        kl_loss = -5e-4 * K.mean(z_log_var - K.square(K.exp(z_log_var)) - K.square(z_mean) + 1, axis=-1)
        return K.mean(xent_loss + kl_loss)

    def call(self, inputs):
        x = inputs[0]
        z_decoded = inputs[1]
        loss = self.vae_loss(x, z_decoded)
        self.add_loss(loss, inputs=inputs)
        return z_decoded

y = CustomVariationalLayer()([input_img, z_decoded])

# VAEの訓練
vae = Model(input_img, y)
```

```
vae.compile(optimizer='rmsprop', loss=None)
vae.summary()

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_train = x_train.reshape(x_train.shape + (1,))
x_test = x_test.astype('float32') / 255.
x_test = x_test.reshape(x_test.shape + (1,))

vae.fit(x=x_train,y=None, shuffle=True, epochs=10, batch_size=batch_size, validation_data=(x_test, None))

#  ２次元の潜在空間から点のグリッドを抽出し、画像にデコード
n = 15
digit_size = 28
figure = np.zeros((digit_size * n, digit_size * n))

grid_x = norm.ppf(np.linspace(0.05, 0.95, n))
grid_y = norm.ppf(np.linspace(0.05, 0.95, n))

for i, xi in enumerate(grid_x):
    for j ,yj in enumerate(grid_y):
        z_sample = np.array([xi, yj]).reshape(1, latent_dim)
        digit = decoder.predict(z_sample, batch_size=1)
        digit = digit[0].reshape(digit_size, digit_size)
        figure[j * digit_size: (j + 1) * digit_size,
               i * digit_size: (i + 1) * digit_size] = digit

plt.figure(figsize=(10, 10))
plt.imshow(figure, cmap='Greys_r')
plt.show
```

```
/usr/local/lib/python3.6/dist-packages/keras/engine/training_utils.py:819: UserWarning: Output custom_variational_layer_5 missing from loss
  'be expecting any data to be passed to {0}.'.format(name))
Model: "model_9"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_14 (InputLayer)           (None, 28, 28, 1)    0
__________________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, 28, 28, 32)   320         input_14[0][0]
__________________________________________________________________________________________________
conv2d_39 (Conv2D)              (None, 14, 14, 64)   18496       conv2d_38[0][0]
__________________________________________________________________________________________________
conv2d_40 (Conv2D)              (None, 14, 14, 64)   36928       conv2d_39[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, 14, 14, 64)   36928       conv2d_40[0][0]
__________________________________________________________________________________________________
flatten_9 (Flatten)             (None, 12544)        0           conv2d_41[0][0]
__________________________________________________________________________________________________
dense_26 (Dense)                (None, 32)           401440      flatten_9[0][0]
__________________________________________________________________________________________________
dense_27 (Dense)                (None, 2)            66          dense_26[0][0]
__________________________________________________________________________________________________
dense_28 (Dense)                (None, 2)            66          dense_26[0][0]
__________________________________________________________________________________________________
lambda_7 (Lambda)               (None, 2)            0           dense_27[0][0]
                                                                 dense_28[0][0]
__________________________________________________________________________________________________
model_8 (Model)                 (None, 28, 28, 1)    56385       lambda_7[0][0]
__________________________________________________________________________________________________
custom_variational_layer_5 (Cus [(None, 28, 28, 1),  0           input_14[0][0]
                                                                 model_8[1][0]
==================================================================================================
Total params: 550,629
Trainable params: 550,629
Non-trainable params: 0
__________________________________________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 22s 361us/step - loss: 0.2113 - val_loss: 0.1953
Epoch 2/10
60000/60000 [==============================] - 21s 352us/step - loss: 0.1930 - val_loss: 0.1898
Epoch 3/10
```

```
Epoch 3/10
60000/60000 [==============================] - 21s 354us/step - loss: 0.1886 - val_loss: 0.1871
Epoch 4/10
60000/60000 [==============================] - 22s 361us/step - loss: 0.1860 - val_loss: 0.1839
Epoch 5/10
60000/60000 [==============================] - 21s 347us/step - loss: 0.1843 - val_loss: 0.1854
Epoch 6/10
60000/60000 [==============================] - 21s 358us/step - loss: 0.1832 - val_loss: 0.1818
Epoch 7/10
60000/60000 [==============================] - 21s 352us/step - loss: 0.1821 - val_loss: 0.1810
Epoch 8/10
60000/60000 [==============================] - 21s 353us/step - loss: 0.1813 - val_loss: 0.1804
Epoch 9/10
60000/60000 [==============================] - 21s 351us/step - loss: 0.1807 - val_loss: 0.1810
Epoch 10/10
60000/60000 [==============================] - 22s 359us/step - loss: 0.1802 - val_loss: 0.1801
<function matplotlib.pyplot.show>
```




350
400
0
50
100
150
200
250
300
350
400