1. Постановка задачи.

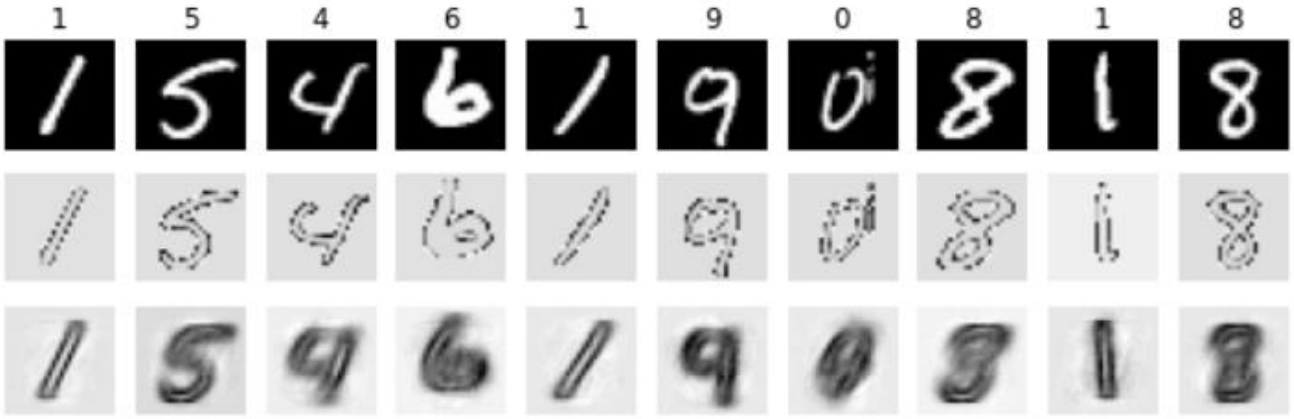
Обучить автокодировщик генерировать рукописные цифры, аналогичные получаемым с помощью ImageDataGenerator при задании featurewise_center = True.

2. Модель НС.

```
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 784)]             0
_________________________________________________________________
dense (Dense)                (None, 512)               401920
_________________________________________________________________
leaky re lu (LeakyReLU)      (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense 1 (Dense)              (None, 256)               131328
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dropout 2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense 3 (Dense)              (None, 64)                8256
_________________________________________________________________
leaky re lu 3 (LeakyReLU)    (None, 64)                0
_________________________________________________________________
dropout 3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 32)                0
_________________________________________________________________
dense 5 (Dense)              (None, 64)                2112
_________________________________________________________________
leaky re lu 5 (LeakyReLU)    (None, 64)                0
_________________________________________________________________
dropout 4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               8320
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 128)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense 7 (Dense)              (None, 256)               33024
_________________________________________________________________
leaky re lu 7 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dropout 6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 512)               131584
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_9 (Dense)              (None, 784)               402192
=================================================================
Total params: 1,153,712
Trainable params: 1,153,712
Non-trainable params: 0
_________________________________________________________________
```

3. **Изображения.**

Исходные, целевые и генерируемые.



4. **Код программы.**

```
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.datasets import mnist
from keras.layers import Dense, Input, Flatten, Reshape, Dropout, LeakyReLU
from keras.models import Model
from tensorflow.keras.optimizers import Adam
import string


pathToData = 'mnist/'
num_classes = 10
img_cols = img_rows = 28
x_train_shape_0 = 60000
x_test_shape_0 = 10000


# Загрузка EMNIST
def load_data():
    print('Загрузка данных из двоичных файлов...')
    with open(pathToData + 'imagesTrain.bin', 'rb') as read_binary:
        x_train = np.fromfile(read_binary, dtype=np.uint8)
    with open(pathToData + 'labelsTrain.bin', 'rb') as read_binary:
        y_train = np.fromfile(read_binary, dtype=np.uint8)
    with open(pathToData + 'imagesTest.bin', 'rb') as read_binary:
        x_test = np.fromfile(read_binary, dtype=np.uint8)
    with open(pathToData + 'labelsTest.bin', 'rb') as read_binary:
        y_test = np.fromfile(read_binary, dtype=np.uint8)
    x_train = np.array(x_train, dtype='float32')
    x_test = np.array(x_test, dtype='float32')
    x_train = x_train.reshape(-1, img_rows, img_cols, 1)
    x_test = x_test.reshape(-1, img_rows, img_cols, 1)
    return x_train, y_train, x_test, y_test


def makeNames():
    return list(string.digits)


x_train, y_train, x_test, y_test = load_data()

# Все параметры имеют заданные по умолчанию значения
datagen = ImageDataGenerator(featurewise_center = True, data_format = 'channels_last')
```

```
print('Настройка генератора...')
datagen.fit(x_train)
x_y_train = datagen.flow(x_train, y_train, batch_size = x_train_shape_0, shuffle = False)
genimg_train = x_y_train[0][0].astype('uint8')
genlab_train = x_y_train[0][1]
print(type(x_train))
print(type(genimg_train))
print(genimg_train.shape)

# вывод 10 рандомных изображений
names = makeNames()
n = 10
f = plt.figure(figsize=(n, 2))
for i in range(n):
    j = np.random.randint(0, high=x_train_shape_0, dtype=int)
    ind = genlab_train[j]
    let = names[ind]
    sp = f.add_subplot(2, n, i + 1)
    sp.axis('Off')
    img = x_train[j]
    img = img.reshape(img_rows, img_cols)
    plt.imshow(img, cmap=plt.get_cmap('gray'))
    sp.set_title(let)

    sp1 = f.add_subplot(2, n, i + 1 + n)
    sp1.axis('Off')
    img = genimg_train[j]
    img = img.reshape(img_rows, img_cols)
    plt.imshow(img, cmap=plt.get_cmap('gray'))
plt.show()


def one_part(units, x):
    x = Dense(units)(x)
    x = LeakyReLU()(x)
    return Dropout(0.25)(x)


latent_size = 32 # Размер латентного пространста
inp = Input(shape = (784, ))
# x = Reshape((-1, ))(inp)
x = one_part(512, inp)
x = one_part(256, x)
x = one_part(128, x)
x = one_part(64, x)
x = Dense(latent_size)(x)
encoded = LeakyReLU()(x)
x = one_part(64, encoded)
x = one_part(128, x)
x = one_part(256, x)
x = one_part(512, x)
decoded = Dense(784, activation = 'sigmoid')(x)
# x = Dense(784, activation = 'sigmoid')(x)
# decoded = Reshape((28, 28, 1))(x)
model = Model(inputs = inp, outputs = decoded)
model.summary()


model.compile(optimizer = Adam(learning_rate=0.001), loss = 'msle')

x_train = x_train.reshape(-1, img_cols * img_rows) / 255.0
genimg_train = genimg_train.reshape(-1, img_cols * img_rows) / 255.0
```

```
epochs = 10
batch_size = 100
model.fit(x_train, genimg_train, epochs = epochs, batch_size = batch_size, shuffle = False)


pred_train = model.predict(x_train)

f = plt.figure(figsize=(n, 3))
for i in range(n):
    j = np.random.randint(0, high=len(pred_train), dtype=int)
    sp = f.add_subplot(3, n, i+1)
    sp.axis('Off')
    img = x_train[j]  # .astype('uint8')
    img = img.reshape(img_rows, img_cols)
    plt.imshow(img, cmap = plt.get_cmap('gray'))
    sp1 = f.add_subplot(3, n, i+1+n)
    sp1.axis('Off')
    img = genimg_train[j]  # .astype('uint8')
    img = img.reshape(img_rows, img_cols)
    plt.imshow(img, cmap = plt.get_cmap('gray'))
    sp2 = f.add_subplot(3, n, i+1+2*n)
    sp2.axis('Off')
    img = pred_train[j]  #.astype('uint8')
    img = img.reshape(img_rows, img_cols)
    plt.imshow(img, cmap = plt.get_cmap('gray'))
    sp.set_title(names[genlab_train[j]])
plt.show()
```