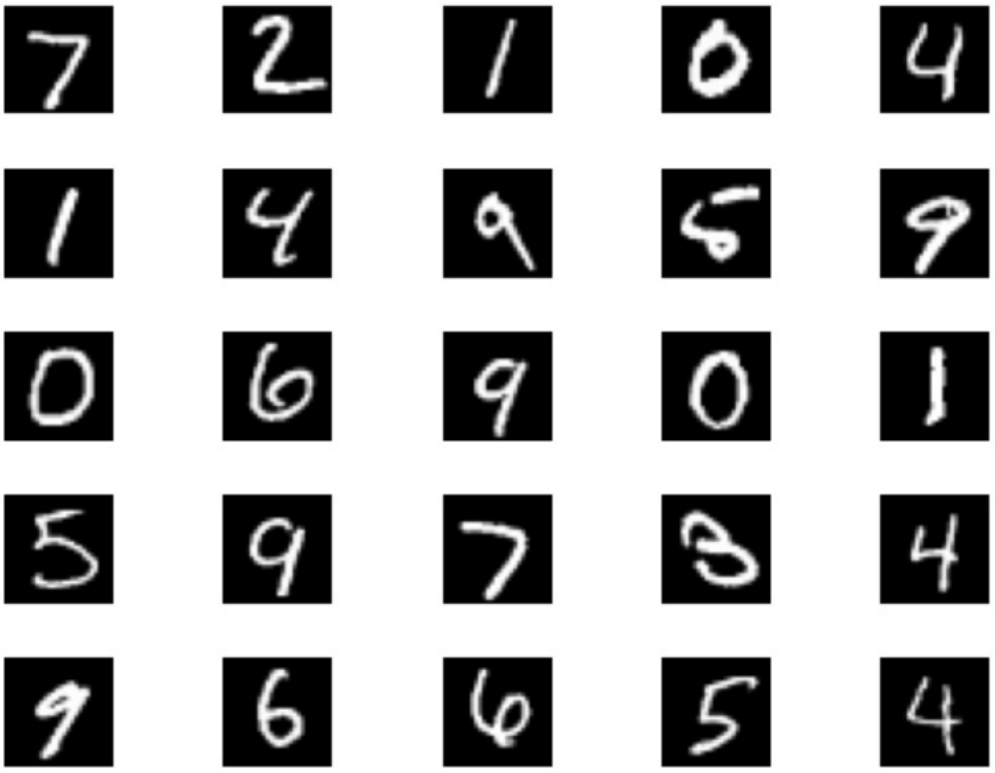
1.Задача

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

2. Модель НС (model.summary()).

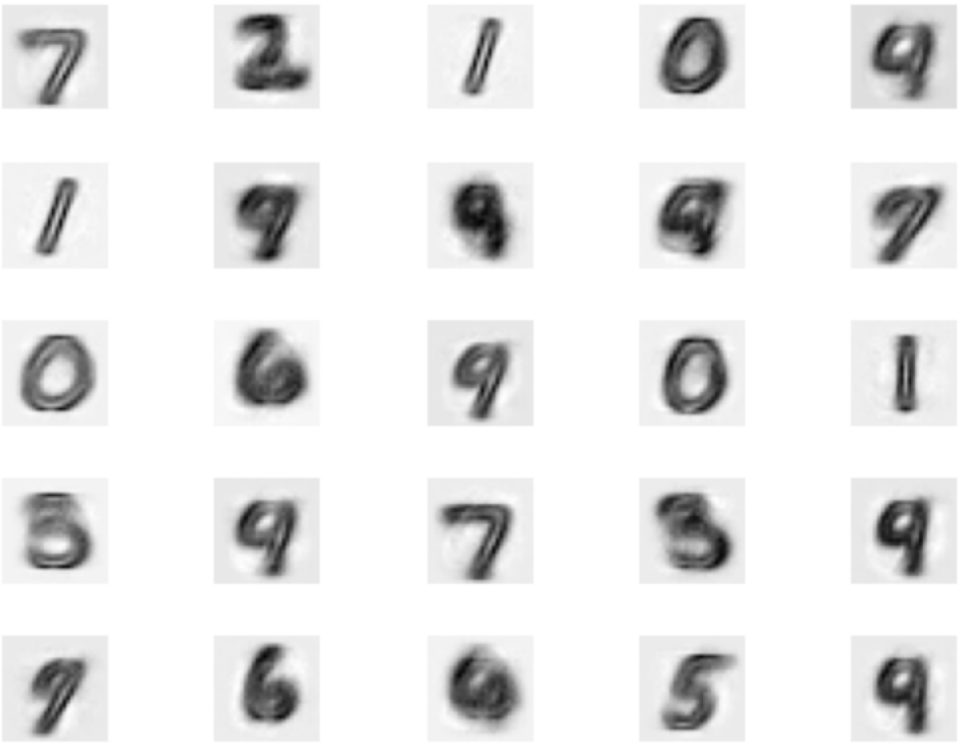
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 784)]             0
_________________________________________________________________
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. Примеры целевых изображений.

4. Примеры генерируемых изображений.



5. Код программы.

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Input, Dense, LeakyReLU, Dropout
from matplotlib import pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Для очистки области вывода в IPython, Jupyter, Colab
#from IPython import display
import numpy as np

epochs = 20
(x_trn, y_trn), (x_tst, y_tst) = mnist.load_data()
x_trn = x_trn.reshape(x_trn.shape[0], 28, 28, 1)
#Используем ImageDataGenerator
generation = ImageDataGenerator(featurewise_center= True)
generation.fit(x_trn)
x_y = generation.flow(x_trn, y_trn, batch_size = len(y_trn),
```

```
shuffle=False)[0][0][:].astype('uint8')

#Кодер и декодер автокодировщика построены из блоков,
#формируемых процедурой one_part:
def one_part(units, x):
   x = Dense(units)(x)
   x = LeakyReLU()(x)
   return Dropout(0.3)(x)
def some_plts(imgs):
   fig, axs = plt.subplots(5, 5)
   k = -1
   for i in range(5):
       for j in range(5):
           k += 1
           img = imgs[k].reshape(28, 28)
           axs[i, j].imshow(img, cmap = 'gray')
           axs[i, j].axis('off')
   plt.subplots_adjust(wspace = 0.5, hspace = 0.5)
   plt.show()
some_plts(x_trn)
some_plts(x_y)

#Формирование модели автокодировщика

latent_size = 32 # Размер латентного пространста
inp = Input(shape = (784))
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)
model = Model(inputs = inp, outputs = decoded)
model.compile('adam', loss = 'binary_crossentropy') # nadam
model.summary()
#нормализация данных
x_y = x_y.reshape(-1, 784) / 255.
x_trn = x_trn.reshape(-1, 784) / 255.
#Обучаем модель
model.fit(x_trn, x_y, epochs = epochs, batch_size = 256)
```

```
x_tst = x_tst.reshape(x_tst.shape[0], 28, 28, 1)
generation_test = ImageDataGenerator(featurewise_center=True)
generation_test.fit(x_tst)

x_tst1 = generation.flow(x_tst, y_tst, batch_size=len(y_tst),
shuffle=False)[0][0][:].astype('uint8')
x_tst = x_tst.reshape(-1, 784) / 255.0
some_plts(x_tst)
some_plts(x_tst1)
#Прогноз автокодировщика
predicted_images = model.predict(x_tst)
some_plts(predicted_images)
```