1. Постановка задачи.

Приспособить порождающую состязательную нейронную сеть (НС) для генерации рукописных цифр.
В отличие от MNIST, цифры отображаются на белом фоне.
Обучающее множество формируется на основе обучающего множества MNIST.
Условия обучения:
- на вход генератора подаются изображения из `x_train`;
- на вход дискриминатора - изображения из `x_reversed` и изображения, порожденные генератором.
Кроме прочих изменений, код дополняется блоком, предусматривающим загрузку модели обученного генератора и генерацию изображений на основе тестового множества MNIST.

2. Параметры модели.

Число эпох – 5001

Размер обучающего пакета – 128

Оптимизатор – Adam, шаг обучения – 0.0002

Функция потерь дискриминатора – бинарная перекрёстная энтропия, функция потерь генератора – mean square logarithmic error

3. Изменения в изначальной программе.

В качестве обучающей и тестовой выборки мы возьмём изображения восьмёрок из набора MNIST. В качестве реальных примеров для дискриминатора будем брать изображения восьмёрок на белом фоне. Таким образом, изменяется функция m_inp:

```
def m_inp(x, x_r, b_s):  # Вход дискриминатора и генератора:
                         # x – изображения 8 из набора MNIST; x_r – реверсированные
                         # изображения
    idx = np.random.randint(0, x.shape[0], b_s)
    d_n = x_r[idx]
    g_n = x[idx]
    return d_n, g_n
```

В качестве генератора берётся модель автокодировщика из ЛР-7.

Она содержит 9 полносвязных слоёв. На входе и выходе – изображения – одномерный массив длиной 784. Функция активации выходного слоя – ‘sigmoid’.

Дискриминатор не изменяется.

После обучения, сохранённая ранее модель генератора выгружается из файла (models.load_model), она используется для генерации изображений из проверочной выборки.

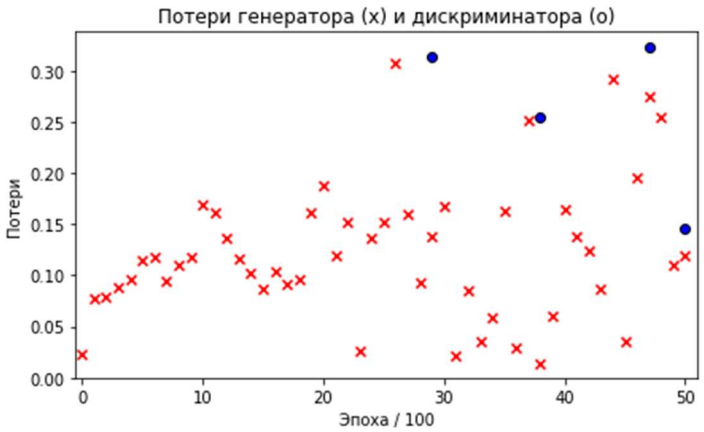
4. model.summary() всех компонентов НС

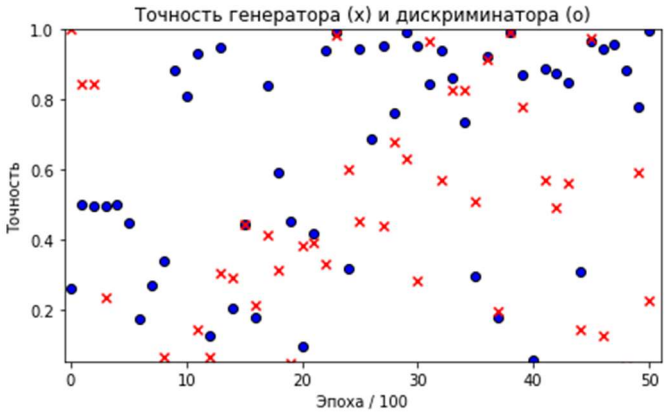
Model: "model" – генератор

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
_________________________________________________________________
Model: "model_1" – дискриминатор
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 784)]             0

dense_10 (Dense)             (None, 512)               401920

leaky_re_lu_9 (LeakyReLU)    (None, 512)               0

dense_11 (Dense)             (None, 256)               131328

leaky_re_lu_10 (LeakyReLU)   (None, 256)               0

dense_12 (Dense)             (None, 1)                 257

=================================================================
Total params: 533,505
Trainable params: 533,505
Non-trainable params: 0
_________________________________________________________________
Model: "model_2" – combined
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 784)]             0

model (Functional)           (None, 784)               1153712

model_1 (Functional)         (None, 1)                 533505

=================================================================
Total params: 1,687,217
Trainable params: 1,153,712
Non-trainable params: 533,505
_________________________________________________________________
```

5. Графики обучения.





6. Изображения, полученные в результате тестирования генератора

Заключение: модель GAN, состоящая из полносвязных слоёв, не даёт стабильного улучшения качества генерируемых изображений при увеличении числа эпох. Вцелом, практика использования модели показала, что для каждой цифры в наборе MNIST требуется свой оптимальный набор гиперпараметров для того, чтобы генерировались более реалистичные изображения. Практика показала, что модель работает крайне нестабильно, если использовать набор не из примеров конкретной цифры, а набор из всех цифр. Также модель может выдавать хорошие результаты для конкретной цифры при числе эпох – 3500, а при числе эпох 5000 – результат ухудшается.