Нейроинформатика. Лабораторная работа 5

Сети с обратными связями

Целью работы является исследование свойств сети Элмана, алогритмов обучения, а также применения сетей в задачах распознавания статических и динамических образов.

Выполнил Лисин Роман, М8О-406Б-20. Вариант 12.

```
import numpy as np

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.layers import Layer
from keras import backend as back

import matplotlib.pyplot as plt
```

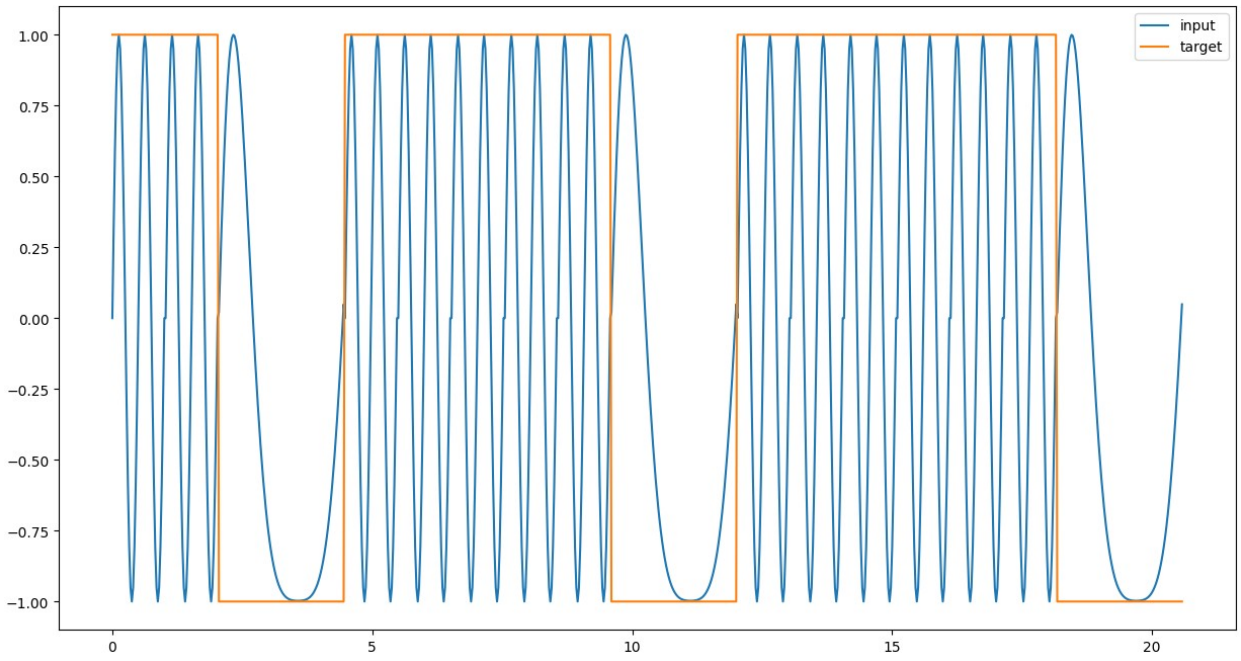
Зададим два сигнала - основной сигнал (p_1) и сигнал, подлежащий распознаванию (p_2)

```
def fp1(k):
    return np.sin(4 * np.pi * k)

def fp2(k):
    return np.sin(2 * k * k - 6 * k + 3)


k1_begin = 0
k1_end = 1

k2_begin = -0.02
k2_end = 2.36

h = 0.025

r = (2, 5, 6)
```

Объединим сигналы в один массив - сначала идет r[0] повторений сигнала p1, затем одно повторение сигнала p2, потом r[1] повторений сигнала p1, одно p2 и т.д.

```
k1 = np.arange(k1_begin, k1_end+h, h)
k2 = np.arange(k2_begin, k2_end+h, h)

p1 = fp1(k1)
p2 = fp2(k2)
```

```
p = np.concatenate((
    np.tile(p1, r[0]),
    p2,
    np.tile(p1, r[1]),
    p2,
    np.tile(p1, r[2]),
    p2,
))

t = np.concatenate((
    np.ones(len(p1) * r[0]),
    -np.ones(len(p2)),
    np.ones(len(p1) * r[1]),
    -np.ones(len(p2)),
    np.ones(len(p1) * r[2]),
    -np.ones(len(p2)),
))

x = np.arange(len(p)) * h

plt.figure(figsize=(15, 8))

plt.plot(x, p, label='input')
plt.plot(x, t, label='target')
plt.legend()

plt.show()
```



Сгенерируем датасет для обучения. В качестве входных признаков будем брать несколько подряд идущих элементов. Таргеты - тип сигнала в каждой точке промежутка

```
delay = 5

p_train = np.array([np.hstack([p[i:i+delay]]) for i in range(len(p) -
delay)])
t_train = np.array([np.hstack([t[i:i+delay]]) for i in range(len(t) -
delay)])

p_train.shape, t_train.shape
```

```
((819, 5), (819, 5))
```

Для обучения будем использовать сеть Элмана. Реализуем слой для нее

```
class ElmanLayer(Layer):
    def __init__(self, output_dim, **kwargs):
        self.output_dim = output_dim
        self.prev = tf.Variable(tf.zeros((1, output_dim)))
        super(ElmanLayer, self).__init__(**kwargs)

    def build(self, input_shape):
        self.w1 = self.add_weight(
            name='w1',
            shape=(input_shape[1], self.output_dim),
            initializer='uniform',
            trainable=True,
        )

        self.w2 = self.add_weight(
            name='w2',
            shape=(self.output_dim, self.output_dim),
            initializer='uniform',
            trainable=True,
        )

        self.b = self.add_weight(
            name='b',
            shape=(self.output_dim, ),
            initializer='uniform',
            trainable=True,
        )

        super(ElmanLayer, self).build(input_shape)

    def call(self, inputs):
        res = inputs @ self.w1 + self.b
        res += self.prev @ self.w2
        res = tf.keras.activations.tanh(res)
        self.prev.assign(res)
        return res
```

Подготовим модель. Модель будет состоять из комбинации слоя Элмана и линейного слоя

```
model = keras.Sequential([
    ElmanLayer(output_dim=8),
    keras.layers.Dense(delay),
])
```

Обучим ее

```
model.compile(
    loss='mse',
    optimizer='Adam',
)

train_info = model.fit(
    p_train,
    t_train,
    batch_size=1,
    epochs=150,
)
```

```
Epoch 1/150
819/819 [==============================] - 11s 3ms/step - loss: 0.8927
Epoch 2/150
819/819 [==============================] - 3s 4ms/step - loss: 0.8460
Epoch 3/150
819/819 [==============================] - 2s 2ms/step - loss: 0.8488
Epoch 4/150
819/819 [==============================] - 2s 2ms/step - loss: 0.8469
Epoch 5/150
819/819 [==============================] - 2s 2ms/step - loss: 0.8429
Epoch 6/150
819/819 [==============================] - 2s 2ms/step - loss: 0.8405
Epoch 7/150
819/819 [==============================] - 2s 3ms/step - loss: 0.8387
Epoch 8/150
819/819 [==============================] - 3s 3ms/step - loss: 0.8350
Epoch 9/150
819/819 [==============================] - 2s 2ms/step - loss: 0.8339
Epoch 10/150
819/819 [==============================] - 2s 3ms/step - loss: 0.8269
Epoch 11/150
819/819 [==============================] - 2s 2ms/step - loss: 0.8271
Epoch 12/150
819/819 [==============================] - 2s 2ms/step - loss: 0.8237
Epoch 13/150
819/819 [==============================] - 2s 3ms/step - loss: 0.8161
Epoch 14/150
819/819 [==============================] - 2s 3ms/step - loss: 0.8070
Epoch 15/150
```

```
819/819 [==============================] - 2s 2ms/step - loss: 0.8054
Epoch 16/150
819/819 [==============================] - 2s 2ms/step - loss: 0.7960
Epoch 17/150
819/819 [==============================] - 2s 2ms/step - loss: 0.7886
Epoch 18/150
819/819 [==============================] - 2s 2ms/step - loss: 0.7834
Epoch 19/150
819/819 [==============================] - 2s 3ms/step - loss: 0.7693
Epoch 20/150
819/819 [==============================] - 2s 3ms/step - loss: 0.7713
Epoch 21/150
819/819 [==============================] - 2s 2ms/step - loss: 0.7644
Epoch 22/150
819/819 [==============================] - 2s 2ms/step - loss: 0.7521
Epoch 23/150
819/819 [==============================] - 2s 2ms/step - loss: 0.7513
Epoch 24/150
819/819 [==============================] - 2s 3ms/step - loss: 0.7367
Epoch 25/150
819/819 [==============================] - 3s 3ms/step - loss: 0.7383
Epoch 26/150
819/819 [==============================] - 2s 3ms/step - loss: 0.7260
Epoch 27/150
819/819 [==============================] - 2s 3ms/step - loss: 0.7180
Epoch 28/150
819/819 [==============================] - 2s 2ms/step - loss: 0.7109
Epoch 29/150
819/819 [==============================] - 2s 2ms/step - loss: 0.6969
Epoch 30/150
819/819 [==============================] - 2s 2ms/step - loss: 0.6799
Epoch 31/150
819/819 [==============================] - 4s 5ms/step - loss: 0.6882
Epoch 32/150
819/819 [==============================] - 2s 3ms/step - loss: 0.6745
Epoch 33/150
819/819 [==============================] - 2s 3ms/step - loss: 0.6605
Epoch 34/150
819/819 [==============================] - 2s 2ms/step - loss: 0.6559
Epoch 35/150
819/819 [==============================] - 2s 2ms/step - loss: 0.6477
Epoch 36/150
819/819 [==============================] - 3s 3ms/step - loss: 0.6349
Epoch 37/150
819/819 [==============================] - 2s 3ms/step - loss: 0.6258
Epoch 38/150
819/819 [==============================] - 2s 2ms/step - loss: 0.6118
Epoch 39/150
819/819 [==============================] - 2s 2ms/step - loss: 0.6139
```

```
Epoch 40/150
819/819 [==============================] - 2s 2ms/step - loss: 0.6080
Epoch 41/150
819/819 [==============================] - 2s 2ms/step - loss: 0.6019
Epoch 42/150
819/819 [==============================] - 3s 3ms/step - loss: 0.5908
Epoch 43/150
819/819 [==============================] - 2s 3ms/step - loss: 0.5753
Epoch 44/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5728
Epoch 45/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5673
Epoch 46/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5612
Epoch 47/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5487
Epoch 48/150
819/819 [==============================] - 3s 3ms/step - loss: 0.5488
Epoch 49/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5404
Epoch 50/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5416
Epoch 51/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5302
Epoch 52/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5245
Epoch 53/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5239
Epoch 54/150
819/819 [==============================] - 3s 3ms/step - loss: 0.5267
Epoch 55/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5162
Epoch 56/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5016
Epoch 57/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5026
Epoch 58/150
819/819 [==============================] - 2s 2ms/step - loss: 0.5022
Epoch 59/150
819/819 [==============================] - 2s 3ms/step - loss: 0.4950
Epoch 60/150
819/819 [==============================] - 3s 3ms/step - loss: 0.4831
Epoch 61/150
819/819 [==============================] - 2s 2ms/step - loss: 0.4767
Epoch 62/150
819/819 [==============================] - 2s 3ms/step - loss: 0.4587
Epoch 63/150
819/819 [==============================] - 2s 2ms/step - loss: 0.4508
Epoch 64/150
```

```
819/819 [==============================] - 2s 2ms/step - loss: 0.4368
Epoch 65/150
819/819 [==============================] - 2s 3ms/step - loss: 0.4218
Epoch 66/150
819/819 [==============================] - 2s 3ms/step - loss: 0.4101
Epoch 67/150
819/819 [==============================] - 2s 2ms/step - loss: 0.3891
Epoch 68/150
819/819 [==============================] - 2s 3ms/step - loss: 0.3773
Epoch 69/150
819/819 [==============================] - 2s 3ms/step - loss: 0.3620
Epoch 70/150
819/819 [==============================] - 2s 2ms/step - loss: 0.3495
Epoch 71/150
819/819 [==============================] - 2s 3ms/step - loss: 0.3337
Epoch 72/150
819/819 [==============================] - 2s 3ms/step - loss: 0.3256
Epoch 73/150
819/819 [==============================] - 2s 2ms/step - loss: 0.3178
Epoch 74/150
819/819 [==============================] - 2s 2ms/step - loss: 0.3081
Epoch 75/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2983
Epoch 76/150
819/819 [==============================] - 2s 3ms/step - loss: 0.2951
Epoch 77/150
819/819 [==============================] - 3s 3ms/step - loss: 0.2853
Epoch 78/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2767
Epoch 79/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2732
Epoch 80/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2699
Epoch 81/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2650
Epoch 82/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2585
Epoch 83/150
819/819 [==============================] - 3s 3ms/step - loss: 0.2452
Epoch 84/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2481
Epoch 85/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2437
Epoch 86/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2398
Epoch 87/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2370
Epoch 88/150
819/819 [==============================] - 2s 3ms/step - loss: 0.2307
```

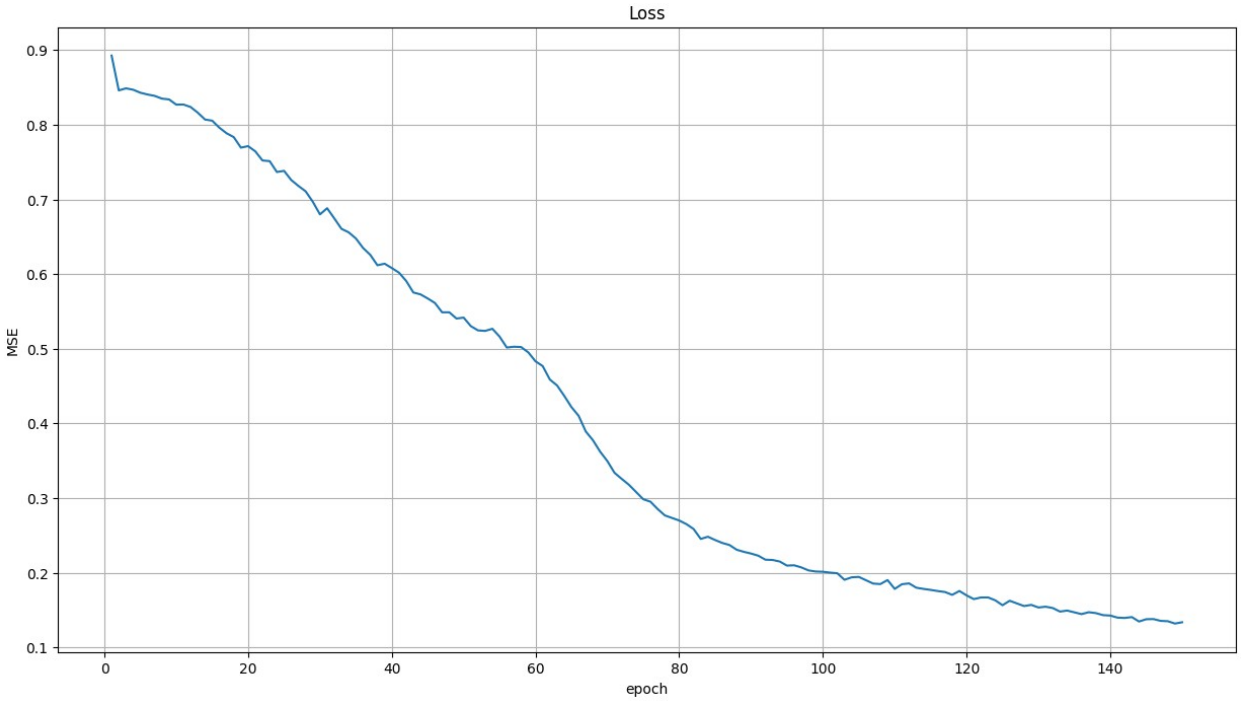
```
Epoch 89/150
819/819 [==============================] - 3s 3ms/step - loss: 0.2279
Epoch 90/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2256
Epoch 91/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2228
Epoch 92/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2174
Epoch 93/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2170
Epoch 94/150
819/819 [==============================] - 2s 3ms/step - loss: 0.2149
Epoch 95/150
819/819 [==============================] - 3s 3ms/step - loss: 0.2094
Epoch 96/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2099
Epoch 97/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2070
Epoch 98/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2030
Epoch 99/150
819/819 [==============================] - 2s 2ms/step - loss: 0.2015
Epoch 100/150
819/819 [==============================] - 2s 3ms/step - loss: 0.2012
Epoch 101/150
819/819 [==============================] - 3s 3ms/step - loss: 0.2001
Epoch 102/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1992
Epoch 103/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1905
Epoch 104/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1937
Epoch 105/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1943
Epoch 106/150
819/819 [==============================] - 3s 3ms/step - loss: 0.1898
Epoch 107/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1854
Epoch 108/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1847
Epoch 109/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1901
Epoch 110/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1783
Epoch 111/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1845
Epoch 112/150
819/819 [==============================] - 3s 3ms/step - loss: 0.1856
Epoch 113/150
```

```
819/819 [==============================] - 2s 2ms/step - loss: 0.1799
Epoch 114/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1783
Epoch 115/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1769
Epoch 116/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1753
Epoch 117/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1741
Epoch 118/150
819/819 [==============================] - 3s 3ms/step - loss: 0.1703
Epoch 119/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1755
Epoch 120/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1696
Epoch 121/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1646
Epoch 122/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1667
Epoch 123/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1668
Epoch 124/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1629
Epoch 125/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1564
Epoch 126/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1624
Epoch 127/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1589
Epoch 128/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1553
Epoch 129/150
819/819 [==============================] - 3s 3ms/step - loss: 0.1568
Epoch 130/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1534
Epoch 131/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1544
Epoch 132/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1526
Epoch 133/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1479
Epoch 134/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1492
Epoch 135/150
819/819 [==============================] - 3s 3ms/step - loss: 0.1469
Epoch 136/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1445
Epoch 137/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1470
```

```
Epoch 138/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1458
Epoch 139/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1431
Epoch 140/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1425
Epoch 141/150
819/819 [==============================] - 3s 3ms/step - loss: 0.1398
Epoch 142/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1393
Epoch 143/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1405
Epoch 144/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1345
Epoch 145/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1376
Epoch 146/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1379
Epoch 147/150
819/819 [==============================] - 3s 3ms/step - loss: 0.1354
Epoch 148/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1350
Epoch 149/150
819/819 [==============================] - 2s 2ms/step - loss: 0.1318
Epoch 150/150
819/819 [==============================] - 2s 3ms/step - loss: 0.1336
```

Посмотрим на лосс

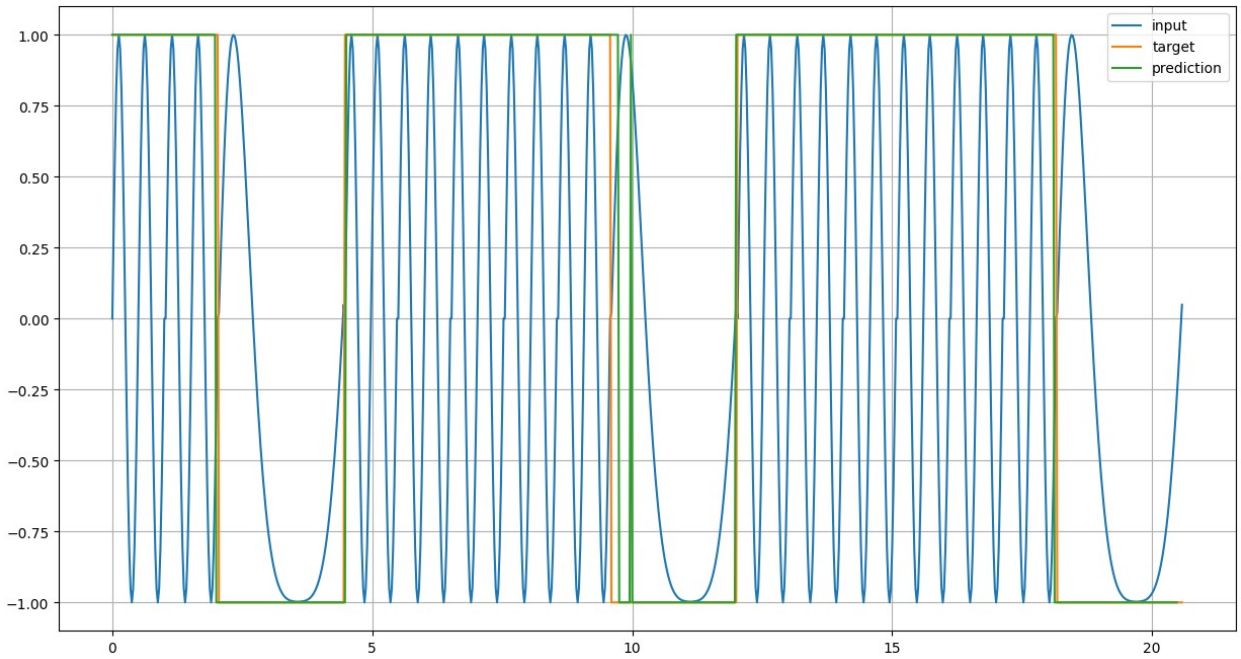
```
plt.figure(figsize=(15, 8))

loss_history = train_info.history['loss']
plt.xlabel('epoch')
plt.ylabel('MSE')
plt.plot(range(1, len(loss_history) + 1), loss_history)
plt.title('Loss')
plt.grid()

plt.show()
```



Посмотрим, как справляется обученная сеть с заданием

```
plt.figure(figsize=(15, 8))

preds = np.hstack((
    np.sign(model.predict(
        np.expand_dims(p_train[i], axis=0), verbose=0
    ))
    for i in range(0, len(p_train), delay)
))[0]

plt.plot(x, p, label='input')
plt.plot(x, t, label='target')
plt.plot(x[:-delay+1], preds, label='prediction')

plt.grid()
plt.legend()
plt.show()
```

```
<ipython-input-12-00848489f1bd>:3: FutureWarning: arrays to stack must be passed as a "sequence" type such as list or tuple. Support for non-sequence iterables such as generators is deprecated as of NumPy 1.16 and will raise an error in the future.
  preds = np.hstack((
```



Сеть почти всегда правильно угадывает таргет, но с некоторыми погрешностями на границах сигналов

Вывод

В данной лабораторной работе я познакомился с одним примером сети с обратными связями - сетью Элмана. Я использовал ее для решения задачи определения типа сигнала. Построенная сеть хорошо справляется с задачей