Практическое задание 5
Вариант 2, цель 5 (X+4+e).

Задание

Необходимо в зависимости от варианта сгенерировать датасет и сохранить его в формате csv. Построить модель, которая будет содержать в себе автокодировщик и регрессионную модель. Обучить модель и разбить обученную модель на 3: Модель кодирования данных, модель декодирования данных, и регрессионную модель.

Выполнение

Генерация датасета и сохраниение его в формате csv:

```
def f1(x): return np.power(-x, 3)
def f2(x): return np.log(np.abs(x))
def f3(x): return np.sin(3 * x)
def f4(x): return np.exp(x)
def f5(x): return x + 4
def f6(x): return -x + np.sqrt(np.abs(x))
def f7(x): return x

def gen_data(data_size=1000):
    X = np.random.normal(-5, 10, data_size)
    e = np.random.normal(0, 0.3, data_size)

    data = np.array([f1(X)+e, f2(X)+e, f3(X)+e, f4(X)+e, f6(X)+e, f7(X)+e]).T
    labels = np.array([f5(X)+e]).T

    return data, labels
```

```
train_data, train_labels = gen_data()
test_data, test_labels = gen_data(200)
np.savetxt('train_data.csv', train_data, delimiter=',')
np.savetxt('train_labels.csv', train_labels, delimiter=',')
np.savetxt('test_data.csv', test_data, delimiter=',')
np.savetxt('test_labels.csv', test_labels, delimiter=',')
```

Модель, которая содержит в себе автокодировщик и регрессионную модель:

```
input_layer = Input(shape=(6,))

encoded = Dense(32, activation='relu')(input_layer)
encoded = Dense(16, activation='relu')(encoded)
encoded = Dense(8, activation='relu')(encoded)

decoded = Dense(32, activation='relu')(encoded)
decoded = Dense(64, activation='relu')(decoded)
decoded = Dense(6, name='decoded')(decoded)

regression = Dense(32, activation='relu')(encoded)
regression = Dense(64, activation='relu')(regression)
regression = Dense(64, activation='relu')(regression)
regression = Dense(64, activation='relu')(regression)
regression = Dense(1, name='regression')(regression)
```

Обучение модели и разбиение обученной модели на 3: Модель кодирования данных, модель декодирования данных, и регрессионную модель.

```
model = Model(inputs=[input_layer], outputs=[decoded, regression])
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
history = model.fit([train_data], [train_data, train_labels], epochs=200, batch_size=5)

encoded_model = Model(inputs=[input_layer], outputs=[encoded])

decoded_model = Model(inputs=[input_layer], outputs=[decoded])

regression_model = Model(inputs=[input_layer], outputs=[regression])
```

Тестирование моделей и сохранение результатов и самих моделей в csv файлы:

```
encoded_prediction = encoded_model.predict(test_data)
decoded_prediction = decoded_model.predict(test_data)
regression_prediction = regression_model.predict(test_data)

encoded_model.save('encoded_model.h5')
decoded_model.save('decoded_model.h5')
regression_model.save('regression_model.h5')

np.savetxt('encoded.csv', encoded_prediction, delimiter=',')
np.savetxt('decoded.csv', decoded_prediction, delimiter=',')
np.savetxt('regression.csv', regression_prediction, delimiter=',')
```