МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по практической работе №7

по дисциплине «Искусственные нейронные сети»

Тема: Мониторинг моделей глубокого обучения средствами библиотеки Keras

Студентка гр. 8383 _____ Максимова А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

Необходимо построить рекуррентную нейронную сеть, которая будет прогнозировать значение некоторого периодического сигнала.

К каждому варианту предоставляется код, который генерирует последовательность.

Для выполнения задания необходимо:

1. Преобразовать последовательность в датасет, который можно подавать на вход нейронной сети (можно использовать функцию gen_data_from_sequence из примера)
2. Разбить датасет на обучающую, контрольную и тестовую выборки
3. Построить и обучить модель
4. Построить график последовательности, предсказанной на тестовой выборке. Данный график необходимо также добавить в pr

Также, в файлах с кодом вариантов есть функция draw_sequence, которая позволяет нарисовать часть последовательности.

Задание

5 вариант

Код для генерации последовательности:

```
import numpy as np
import random
import math
import matplotlib.pyplot as plt

def gen_sequence(seq_len = 1000):
    seq = [math.cos(i/5)/(1.1 +
math.sin(i/6))/6 + random.normalvariate(0,
0.04) for i in range(seq_len)]
    return np.array(seq)

def draw_sequence():
    seq = gen_sequence(250)
    plt.plot(range(len(seq)),seq)
    plt.show()

draw_sequence()
```

Выполнение работы

1. Исходная последовательность, генерируемая функцией gen_sequence(), была преобразована в датасет с помощью функции gen_data_from_sequence из примера:

```
def gen_data_from_sequence(seq_len=1000, lookback=10):
    seq = gen_sequence(seq_len)

    past = np.array([[[seq[j]] for j in range(i, i + lookback)] for i in range(len(seq) - lookback)])
    future = np.array([[seq[i]] for i in range(lookback, len(seq))])

    return (past, future)
```

2. Полученный датасет был разбит на обучающую, контрольную и тестовую выборки в соотношении 70%:15%:15%.

```
def getDatasets(all_data, all_labels):
    train_size = (len(all_data) // 10) * 7  # 70%
    val_size = (len(all_data) - train_size) // 2  # 15%

    train_data, train_labels = all_data[:train_size], all_labels[:train_size]
    val_data, val_labels = all_data[train_size:train_size+val_size], all_labels[train_size:train_size+val_size]
    test_data, test_labels = all_data[train_size+val_size:], all_labels[train_size+val_size:]

    return train_data, train_labels, val_data, val_labels, test_data, test_labels
```

3. Была построена и обучена модель искусственной нейронной сети. Модель, представленная в примере, была немного модифицирована, в результате чего значения потерь сети изменились с 0.0068 до 0.0049. Параметр return_sequences отвечает за возвращение последовательности, полученной в ходе обучения, так как когда мы используем более одного рекуррентного слоя, нужно подавать на следующий слой не одно значение, а последовательность.

```
def buildModel():
    model = models.Sequential()

    model.add(layers.GRU(48, recurrent_activation='sigmoid', input_shape=(None, 1), return_sequences=True))
    model.add(layers.LSTM(48, activation='relu', input_shape=(None, 1), return_sequences=True, dropout=0.35, recurrent_dropout=0.2))
    model.add(layers.GRU(32, input_shape=(None, 1), recurrent_dropout=0.2))
    model.add(layers.Dense(1))

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

```
model = buildModel()
history = model.fit(
    train_X, train_Y,
    epochs=70,
    validation_data=(val_X, val_Y)
)
```

4. Были написаны функции для построения графика потерь сети и графика последовательности, предсказанной на тестовой выборке.

```
def plotLoss(loss, val_loss, epochs):
    plt.plot(epochs, loss, label="Training loss", linestyle='--', linewidth=2, color='green')
    plt.plot(epochs, val_loss, 'b', label='Validation loss', color='red')

    plt.title('Training and Validation loss')
    plt.xlabel("Epochs")
    plt.ylabel("Loss")

    plt.legend()
    plt.grid()
    plt.savefig("Loss.png", format="png", dpi=240)
    plt.show()
```

```
def plotPredict(predict_res, test_Y, epochs):
    plt.clf()

    plt.plot(epochs, predict_res, label="Predicted sequences", linestyle='--', linewidth=2, color='green')
    plt.plot(epochs, test_Y, 'b', label="Generated sequences", color='red')

    plt.title('Predicted and Generated sequences')
    plt.xlabel("x")
    plt.ylabel("Sequence")

    plt.legend()
    plt.grid()
    plt.savefig("Predict.png", format="png", dpi=240)
    plt.show()
```

Результаты обучения рекуррентной нейронной сети

График потерь нейронной сети на обучающих и тестовых данных

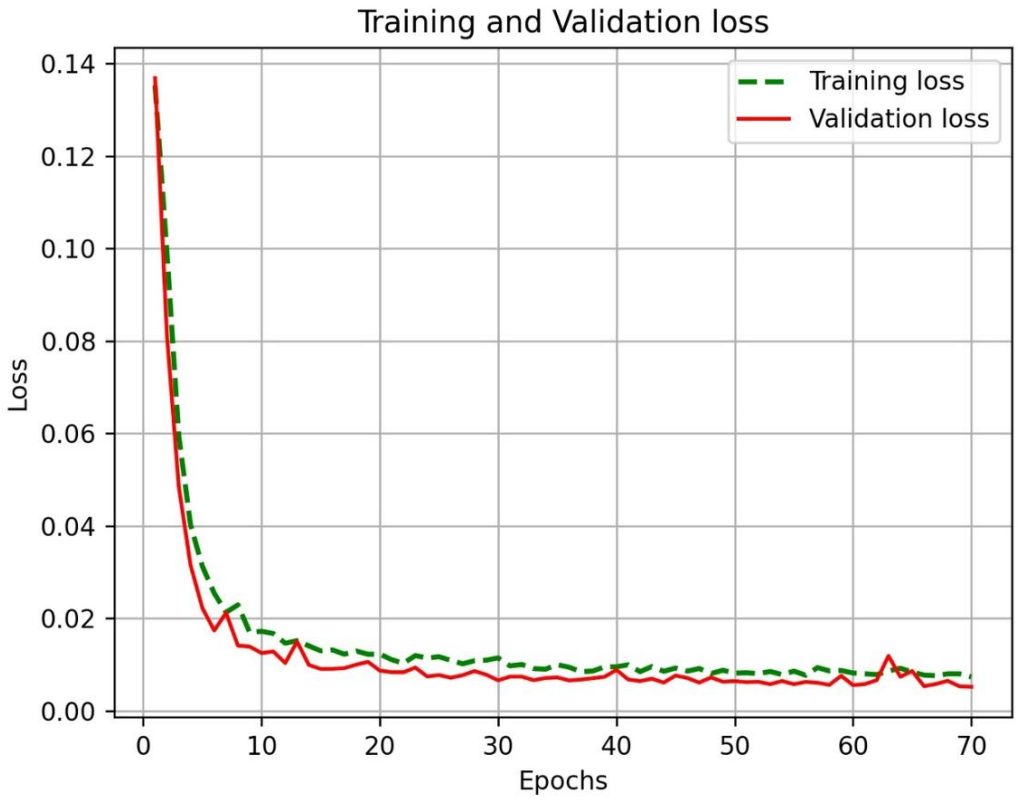
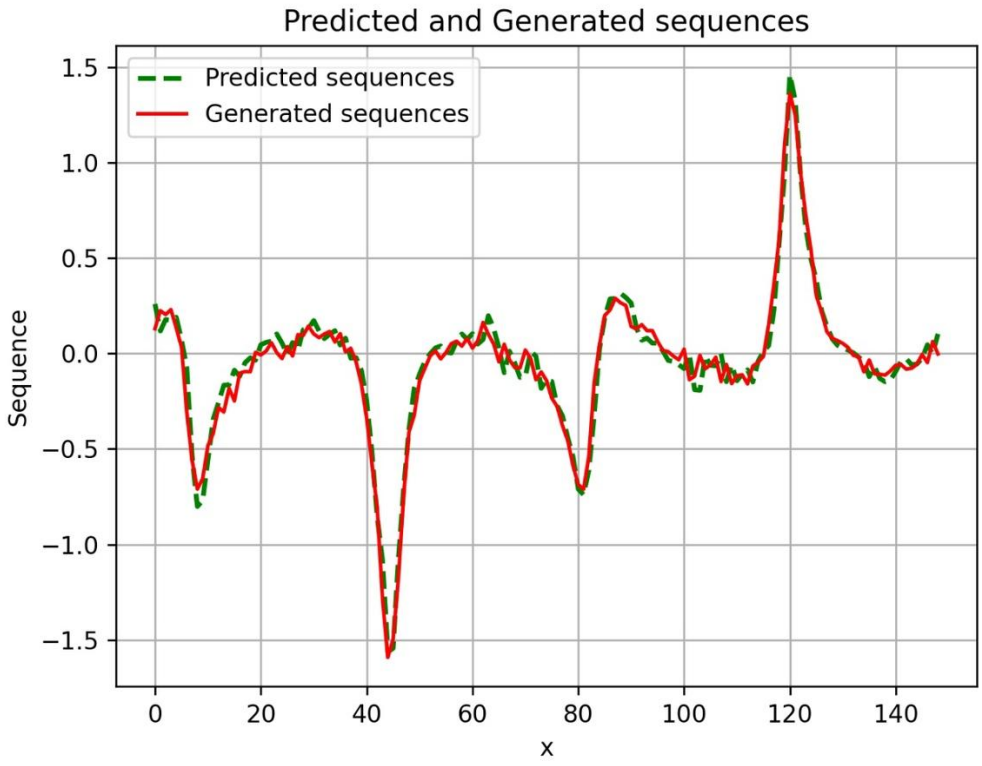


График последовательности, предсказанной на тестовой выборке



Выводы

Была построена рекуррентная нейронная сеть, решающая задачу прогнозирования значения некоторого периодического сигнала. Наименьшее значение потерь сети составило 0.0049.