Выполнила Кузина Анастасия 8382

Вариант 3

Задание:

Необходимо построить рекуррентную нейронную сеть, которая будет прогнозировать значение некоторого периодического сигнала.
К каждому варианту предоставляется код, который генерирует последовательность. Для выполнения задания необходимо:

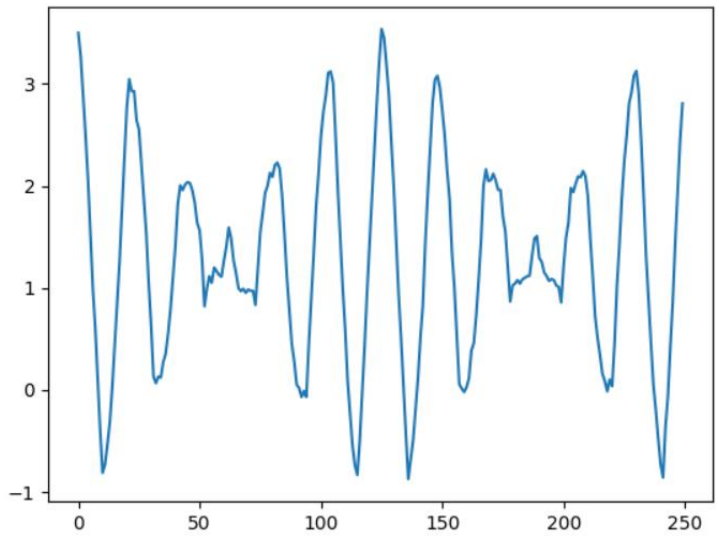
1. Преобразовать последовательность в датасет, который можно подавать на вход нейронной сети (можно использовать функцию gen_data_from_sequence из примера)
2. Разбить датасет на обучающую, контрольную и тестовую выборку
3. Построить и обучить модель
4. Построить график последовательности, предсказанной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

Выполнение:

Данные генерируются согласно данному варианту:

```
def func(i):
    i = i % 21
    return abs(i - 10)/4

def gen_sequence(seq_len = 1000):
    seq = [math.cos(i/4) + func(i) + random.normalvariate(0, 0.04) for i in range(seq_len)]
    return np.array(seq)
```

И затем отображаются в виде графика-участка последовательности:



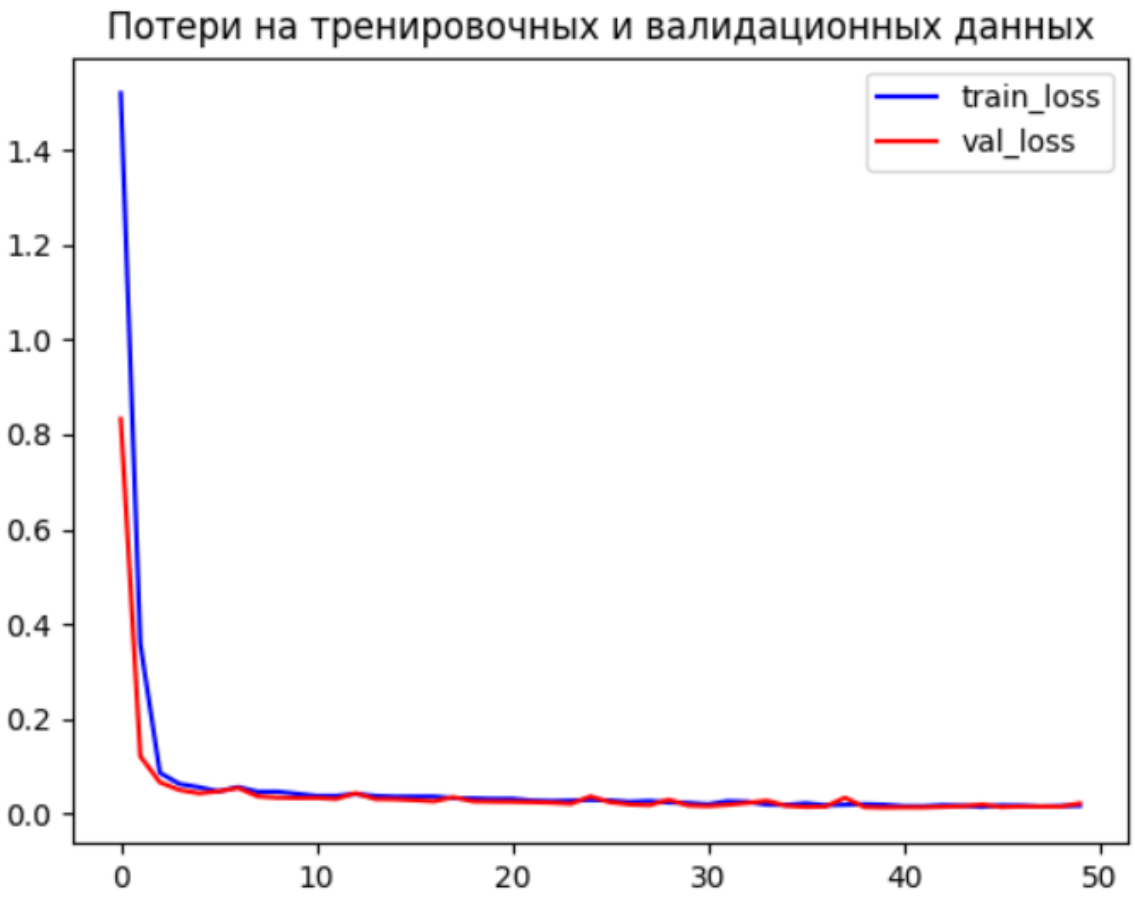
Затем последовательность преобразуется в датасет с ипользованием кода из примера и разделяется на обучающую и тестовую выборки:

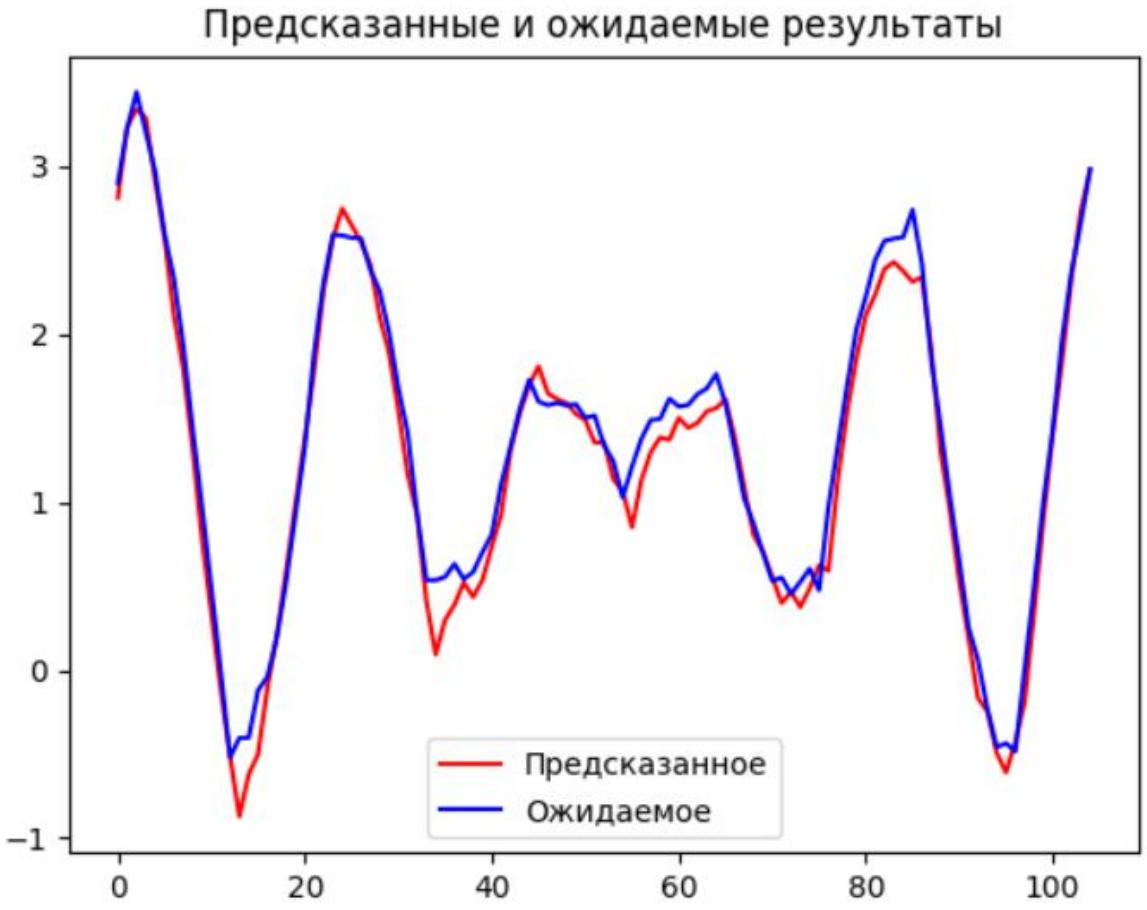
```
seq_len = 1006
lookback = 10
seq = var3.gen_sequence(seq_len)
data = np.array([[[seq[j]] for j in range(i,i+lookback)] for i in range(len(seq) - lookback)])
res = np.array([[seq[i]] for i in range(lookback,len(seq))])


dataset_size = len(data)
train_size = (dataset_size // 10) * 9

train_data, train_res = data[:train_size], res[:train_size]
test_data, test_res = data[train_size:], res[train_size:]
```

Затем создается модель сети, модель компилируется и обучается в течение 50 эпох:

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

model.compile(optimizer='nadam', loss='mse')
history = model.fit(train_data, train_res, epochs=50, validation_split=0.15)
```

Из слоя LSTM было убрано прореживание, что повысило итоговую точность сети. Затем были отрисованы графики потерь на тренировочном и валидационном множествах и график предсказаний сети в сравнении с изначальной последовательностью:


Потери на тренировочных и валидационных данных
train_loss
val_loss
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50


Предсказанные и ожидаемые результаты
3
2
1
0
−1
0
20
40
60
80
100
Предсказанное
Ожидаемое