Постановка задачи.

Вариант 7

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

Изучить пример простой рекуррентной сети, которая предсказывает значение зашумленной синусоиды. Модель из примера не является наилучшей, а лишь демонстрирует пример построения сети со слоями GRU и LSTM.

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

Реализация

Сначала был подключен файл генерация входных данных, сгенерированы данные для обучения модели в количестве 1000 единиц, затем данные были разбиты на тренировочные и тестовые.

```
def gen_data_from_sequence(seq_len=1006, 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)


data, res = gen_data_from_sequence()

dataset_size = len(data)
train_size = (dataset_size // 10) * 7
```

```
val_size = (dataset_size - train_size) // 2
train_data, train_res = data[:train_size], res[:train_size]
val_data, val_res = data[train_size:train_size + val_size],
res[train_size:train_size + val_size]
test_data, test_res = data[train_size + val_size:], res[train_size + val_size:]
```

Затем была создана и обучена модель нейронной сети. Конфигурация модели:

```
model = Sequential()
model.add(layers.GRU(64, recurrent_activation='sigmoid', input_shape=(None,
1), return_sequences=True))
model.add(layers.LSTM(32, activation='relu', input_shape=(None, 1),
return_sequences=True, dropout=0.2))
model.add(layers.GRU(16, input_shape=(None, 1), recurrent_dropout=0.2))
model.add(layers.Dense(1))

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

Данная модель показывает следующие результаты, представленные на рис. 1 и 2.

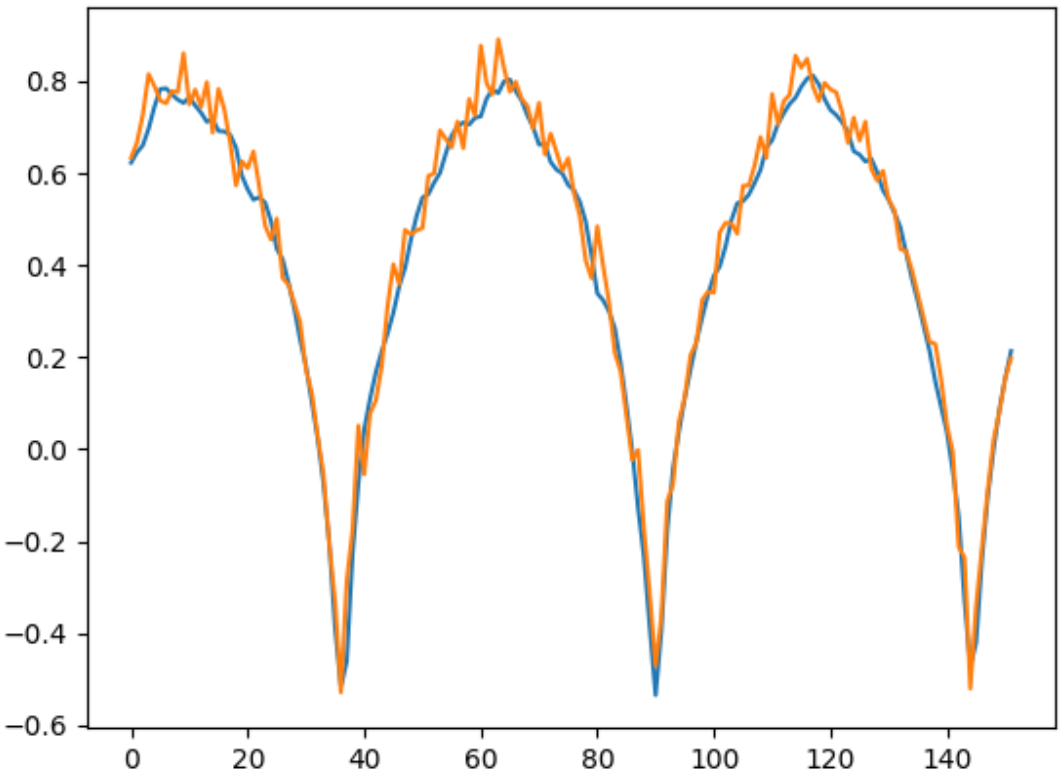

Рисунок 1 – Сигнал и предсказание моделью.

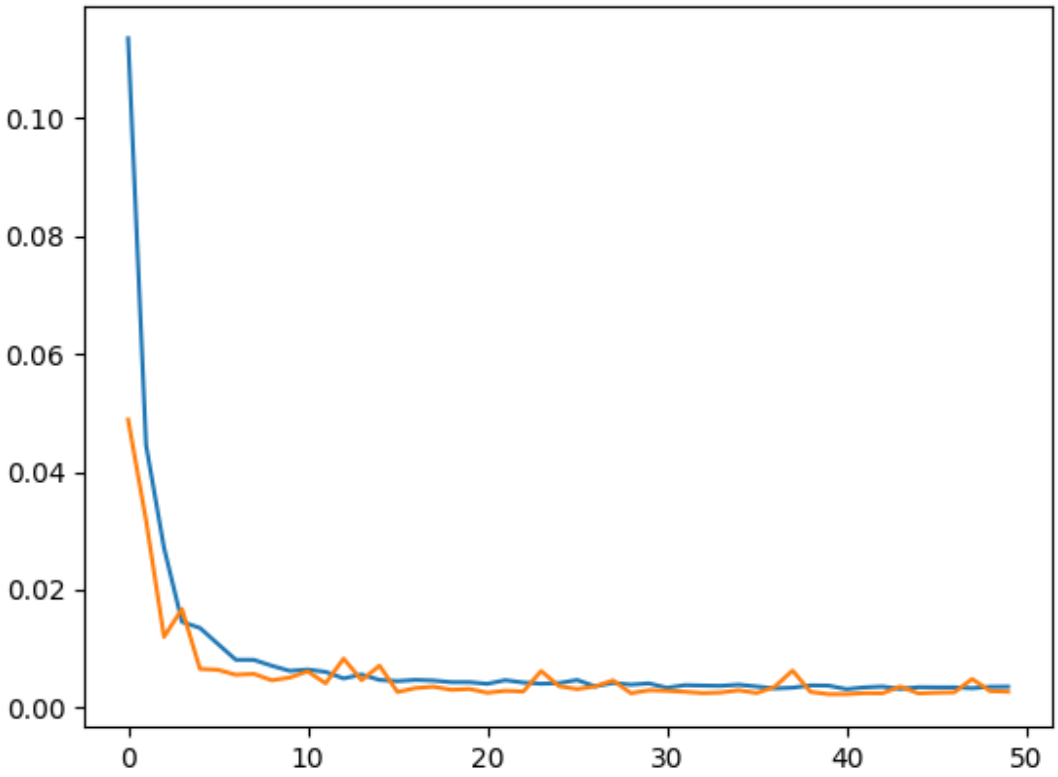

Рисунок 2 – ошибка на обучающих и валидационных данных

Ошибка на обучающих данных – 0.0035, на валидационных - 0.0027.