Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій
Кафедра системного проектування

Звіт
Про виконання лабораторної роботи №6
З курсу «Системи машинного навчання»
Прогнозування часових послідовностей

Виконала:
Студентка групи ФЕС-32
Філь Дарина

Перевірив:
Доцент Колич І.І.

Львів 2024

Мета: навчитися будувати та навчати моделі для прогнозування часових послідовностей за допомогою повнозв'язних нейронних мереж.

Теоретичні відомості

Часові послідовності — це серії даних, зібрані або записані з певними часовими інтервалами. Прогнозування часових послідовностей має на меті передбачити майбутні значення на основі попередніх даних.

Повнозв'язні нейронні мережі складаються з декількох шарів нейронів, де кожен нейрон одного шару з'єднаний з усіма нейронами наступного шару. Такі мережі використовуються для різноманітних задач, включаючи прогнозування часових послідовностей.

Хід роботи

Завдання

1. Підготовка середовища

1.1 Встановіть необхідні бібліотеки, якщо вони ще не встановлені:

- PyTorch
- Keras
- Matplotlib
- Pandas
- Scikit-learn

1.2 Імпортуйте необхідні бібліотеки в Python.

2. Завантаження та підготовка даних

2.1 Завантажте набір даних часових послідовностей:

- Використайте публічний набір даних, наприклад, дані про енергоспоживання з UCI Machine Learning Repository.

2.2 Підготуйте дані:

- Обмежте кількість даних, якщо набір надто великий
- Виберіть дані для передбачення: Voltage
- Перетворіть дані у формат, придатний для використання з нейронними мережами.
- Нормалізуйте дані за допомогою MinMaxScaler для підвищення ефективності навчання.

3. Створення архітектури нейронної мережі

3.1 Створіть модель повнозв'язної нейронної мережі за допомогою Keras:

- Використайте послідовну (Sequential) модель.
- Додайте один або кілька повнозв'язних (Dense) шарів.

3.2 Сконфігуруйте модель для навчання:

- Виберіть оптимізатор (наприклад, Adam).
- Вкажіть функцію втрат (наприклад, mean_squared_error).
- Вкажіть метрику для оцінки моделі (наприклад, mean_absolute_error).

4. Навчання моделі на наборі даних

4.1 Розділіть дані на тренувальну та тестову вибірки:

- Використайте 80% даних для навчання та 20% для тестування.

4.2 Навчіть модель на тренувальних даних:

- Вкажіть кількість епох.
- Вкажіть розмір пакета (batch size).
- Збережіть історію навчання для подальшої візуалізації.

5. Оцінка та візуалізація результатів

5.1 Оцініть модель на тестових даних:

- Обчисліть метрики якості прогнозування на тестових даних.

5.2 Візуалізуйте результати навчання та оцінки:

- Побудуйте графіки втрат та точності на тренувальних та тестових даних.
- Візуалізуйте фактичні та прогнозовані значення часових рядів.

6. Оформити звіт

```
import torch
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
```

Рис. 1 Імпортування усіх бібліотек

```
url = r"C:\Users\Дарина\OneDrive\Робочий стіл\3 курс 1 семестр\Машинне\machine\household_power_consumption.csv"
df = pd.read_csv(url, sep=',', low_memory=False, parse_dates={'datetime': ['Date', 'Time']}, infer_datetime_format=True, na_values=['nan', '?'])



df = df[['datetime', 'Voltage']].dropna()


scaler = MinMaxScaler(feature_range=(0, 1))
df['Voltage'] = scaler.fit_transform(df[['Voltage']])
```

Рис. 2 Підготовка до роботи з дата-сетом: форматування його у датафрейм, видалення всіх значень NaN та застосування MinMaxScaler

```
model = Sequential()
model.add(Dense(64, input_dim=1, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))



model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mean_absolute_error'])
```

Рис. 3 Використання послідовної моделі

```
X = df['Voltage'].values.reshape(-1, 1)
y = df['Voltage'].values


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)



history = model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_test, y_test))
```

Рис. 4 Тренування моделі

```
29s 559us/step - loss: 3.6935e-04 - mean_absolute_error: 0.0017 - val_loss: 3.0713e-06 - val_mean_absolute_error: 0.0017
29s 563us/step - loss: 3.1113e-07 - mean_absolute_error: 3.0142e-04 - val_loss: 4.1071e-08 - val_mean_absolute_error: 1.9761e-04
28s 555us/step - loss: 2.3595e-07 - mean_absolute_error: 2.6144e-04 - val_loss: 3.5841e-10 - val_mean_absolute_error: 1.7254e-05
28s 547us/step - loss: 1.9994e-07 - mean_absolute_error: 2.2979e-04 - val_loss: 1.3467e-09 - val_mean_absolute_error: 3.6090e-05
28s 554us/step - loss: 1.7252e-07 - mean_absolute_error: 2.2851e-04 - val_loss: 1.6354e-10 - val_mean_absolute_error: 1.2382e-05
```

Рис. 5 Результат її тренування на 5 епохах, кожна епоха опрацьовувала 51232 значення

```
Test Loss: 1.6354366472182136e-10
Test MAE: 1.238246386492392e-05
Test R-squared (Accuracy): 0.9999999788568531
```

Рис. 6 Метрики точності моделі, як можемо бачити модель навчилась дуже гарно що фактично призвело до ідеального результату

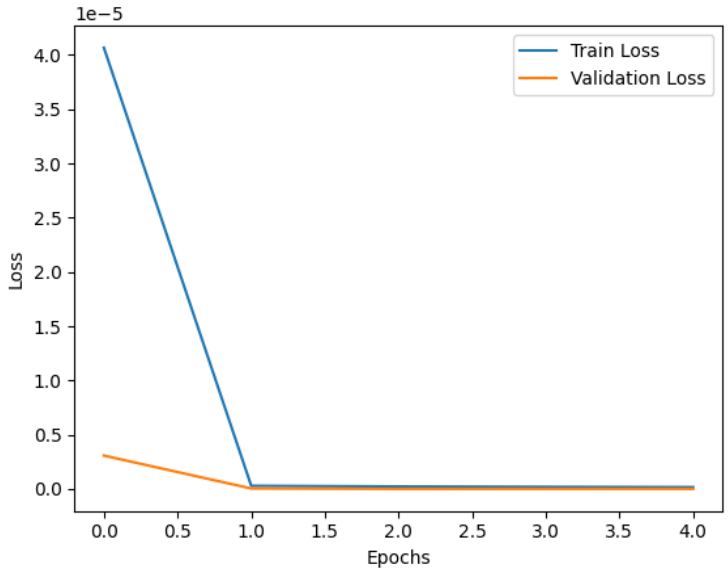

Рис. 7 Графік втрат відносно епохи

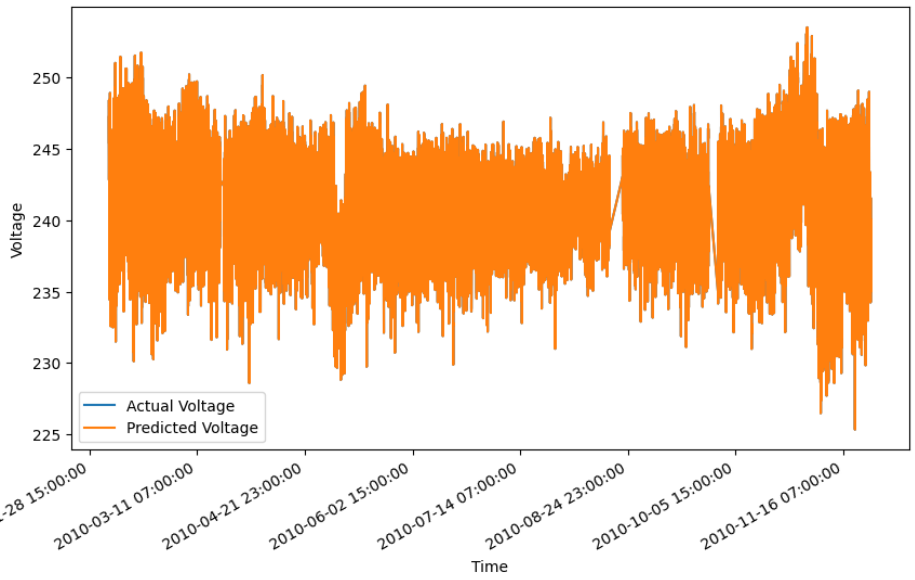

Рис. 8 Графік передбачених значень порівняно зі справжніми на часовій шкалі для всіх значень у дата-сеті

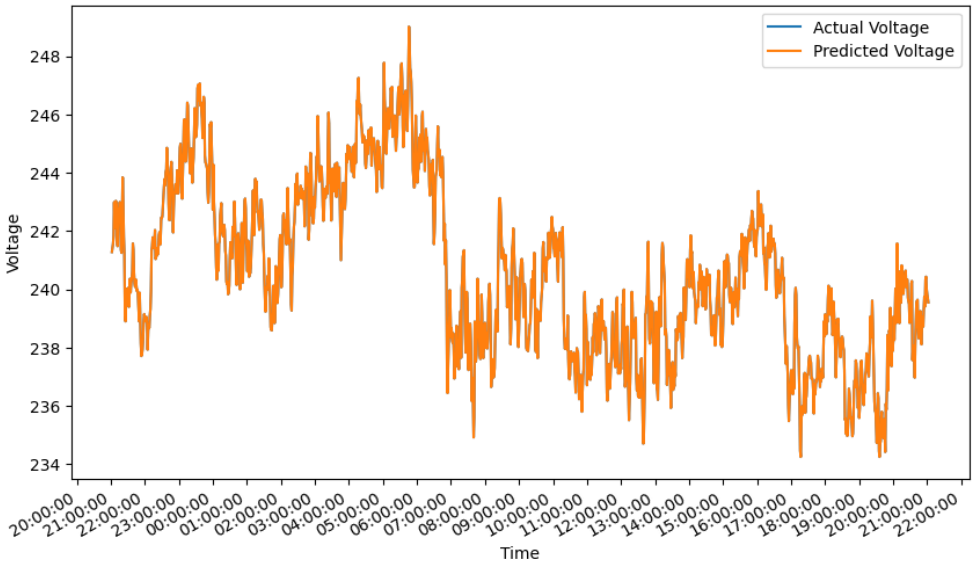

Рис. 9 Графік передбачених значень відносно дійсних протягом 1 дня(зі значень дата-сету)

```
Test Loss: 2.9432685551000 76e-06
Test MAE: 0.0015317910583689809
R-squared: 0.9997912928484904
```

Рис. 10 Обмежила значення до 1000, замість використання усіх даних в дата-фреймі й отримав наступний результат.

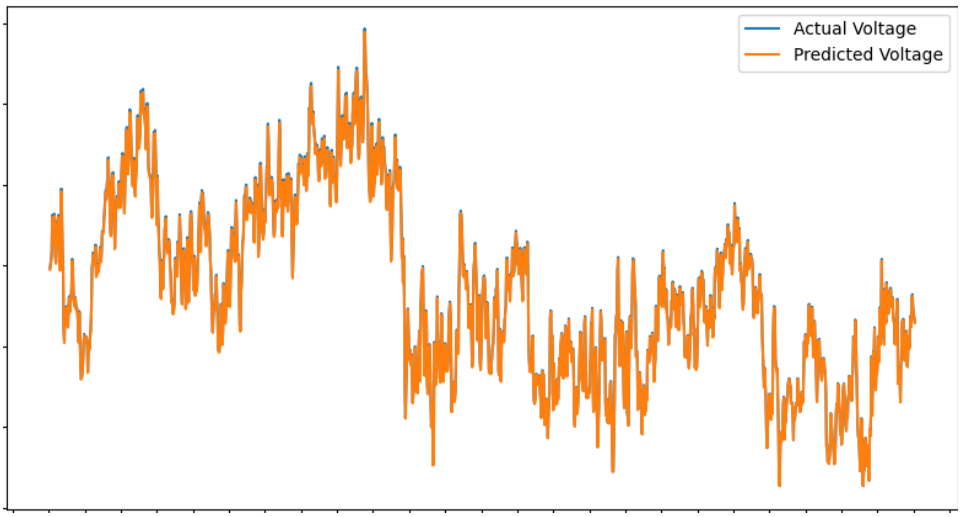

Рис. 11 Графік передбачених значень порівняно з дійсними протягом однієї доби. Можна побачити, що зменшення обсягу даних не спричинило значного зростання похибки

```
df = df[['datatime', 'Voltage']]

df['Voltage'] = df['Voltage'].interpolate(method='linear')

df['Voltage'] = df['Voltage'].fillna(method='fill').fillna(method='bfill')
```

Рис. 12 Замість видалення значень NaN, я вирішила заповнити їх за допомогою вбудованих методів бібліотеки Pandas

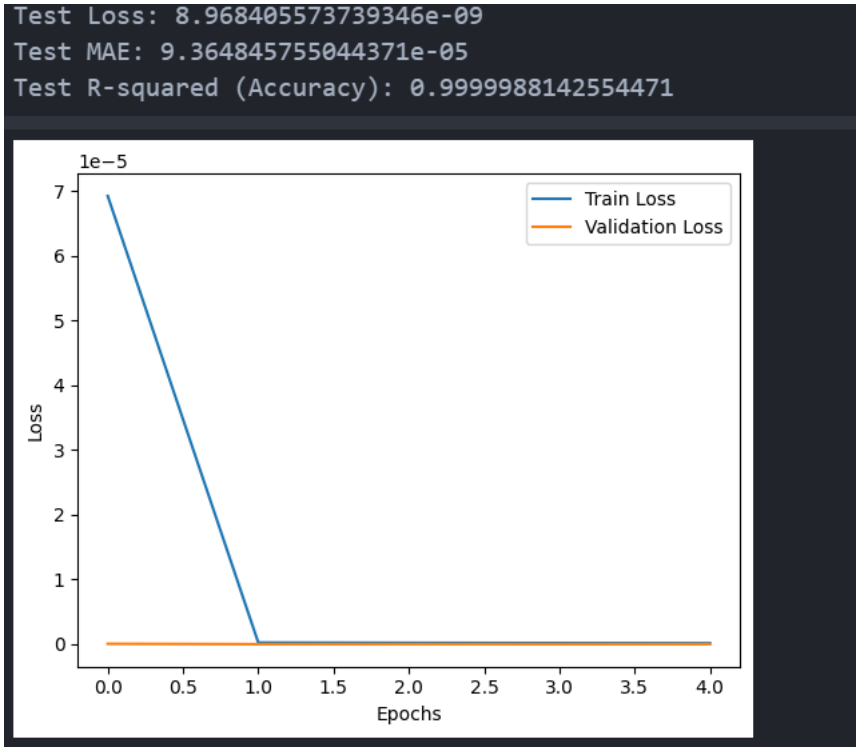

Рис. 13 Це призводить до подібних значень похибки та втрат

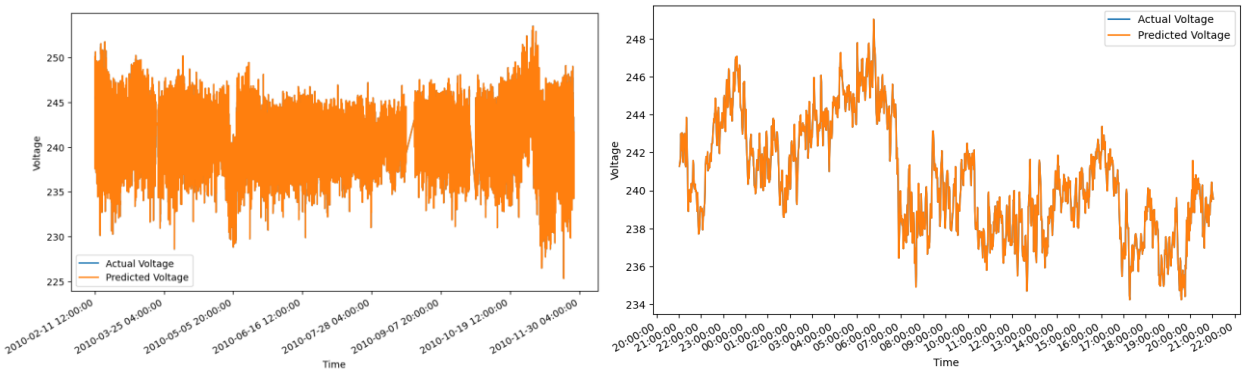

Рис. 14 Графіки актуальних та передбачених значень теж мають схожий вигляд.

Висновок: у цій лабораторній роботі я використала повнозв'язні нейронні мережі для навчання моделі, яка прогнозуватиме часові послідовності. Як видно з результатів, створена модель є дуже точною для набору даних, що містить інформацію про споживання електроенергії серед побутових споживачів.