Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 4
з дисципліни «Методи та технології штучного інтелекту»
На тему:
«Моделювання фушкий двох змінних з двома входами і одним виходом на основі нейронних мереж»

Виконала:
студентка групи ІС-12.
Мельникова К.О.

Перевірив:
Шимкович В. М.

Київ – 2023

Мета роботи: Дослідити структуру та принцип роботи нейронної мережі. За допомогою нейронної мережі змоделювати функцію двох змінних.

Варіант:

11.	$y=\sin((x-2)/2)\cdot x\cdot\sin(x-3)$ $z=\cos(y+x/2)$	11.	20.	9.

Завдання:

За допомогою програмних засобів моделювання або мови програмування високого рівня створити та дослідити вплив кількості внутрішніх шарів та кількості нейронів на середню відносну помилку моделювання для різних типів мереж (feed forward backprop, cascade - forward backprop, elman backprop):

1. Тип мережі: feed forward backprop:

a) 1 внутрішній шар з 10 нейронами;

b) 1 внутрішній шар з 20 нейронами;

2. Тип мережі: cascade - forward backprop:

a) 1 внутрішній шар з 20 нейронами;

b) 2 внутрішніх шари по 10 нейронів у кожному;

3.Тип мережі: elman backprop:

a) 1 внутрішній шар з 15 нейронами;

b) 3 внутрішніх шари по 5 нейронів у кожному;

4. Зробити висновки на основі отриманих даних.

Хід роботи:

1) Спочатку створимо методи, що визначають функції та розіб'ємо дані на навчальний та тестовий датасети в оптимальному відношенні 3:1:

```python
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error

def y_func(x):
    return np.sin((x-2)/2)*x+np.sin(x-3)

def z_func(x, y):
    return np.cos(y+x/2)

x_train, x_test = np.linspace(0, 12, 150), np.linspace(12, 16, 50)
x_combined = np.concatenate((x_train, x_test))
y_train, y_test = y_func(x_train), y_func(x_test)
z_train, z_test = z_func(x_train, y_train), z_func(x_test, y_test)
```

[36] ✓ 0.0s Python

+ Code + Markdown

2) Змоделюємо функцію:

```python
def model_func_graph(title, z_predicted=None):
    plt.plot(x_train, z_train, label="train")
    plt.plot(x_test, z_test, label="test")
    if z_predicted is not None:
        plt.plot(x_test, z_predicted, label="predicted")
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.show()

model_func_graph("Змодельована функція")
```

[37] ✓ 0.0s Python

3) Тепер створимо функції моделей для різних типів мереж: feed forward backprop, cascade - forward backprop та elman backprop:

```python
def feed_forward_backprop(neurons_per_layer):
    model = tf.keras.models.Sequential()
    model.add(tf.keras. layers.Dense(neurons_per_layer, activation='relu', input_shape=(2,)))
    model.add(tf.keras.layers.Dense(1))
    model.compile(loss="mean_squared_error", optimizer="adam")
    return model

def cascade_forward_backprop(neurons_per_layer, hidden_layers=1):
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(neurons_per_layer, activation='relu'))
    for i in range(hidden_layers):
        model.add(tf.keras. layers.Dense(neurons_per_layer, activation='relu'))
    model.add(tf. keras. layers.Dense(1))
    model.compile(loss="mean_squared_error", optimizer="adam")
    return model

def elman_backprop(neurons_per_layer, hidden_layers=1):
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense (neurons_per_layer, activation='relu', input_shape=(1, 2)))
    for _ in range(hidden_layers):
        model.add(tf.keras.layers.SimpleRNN(neurons_per_layer, activation='relu', return_sequences=True))
    model.add(tf.keras.layers.SimpleRNN(neurons_per_layer, activation='relu'))
    model.add(tf.keras.layers.Dense(1))
    model.compile(loss="mean_squared_error", optimizer="adam")
    return model
```

[38] ✓ 0.0s Python

4) Тепер використавши всі три моделі з різними параметрами, виведемо графіки та обрахуємо середньоквадратичні похибки для кожного випадку:

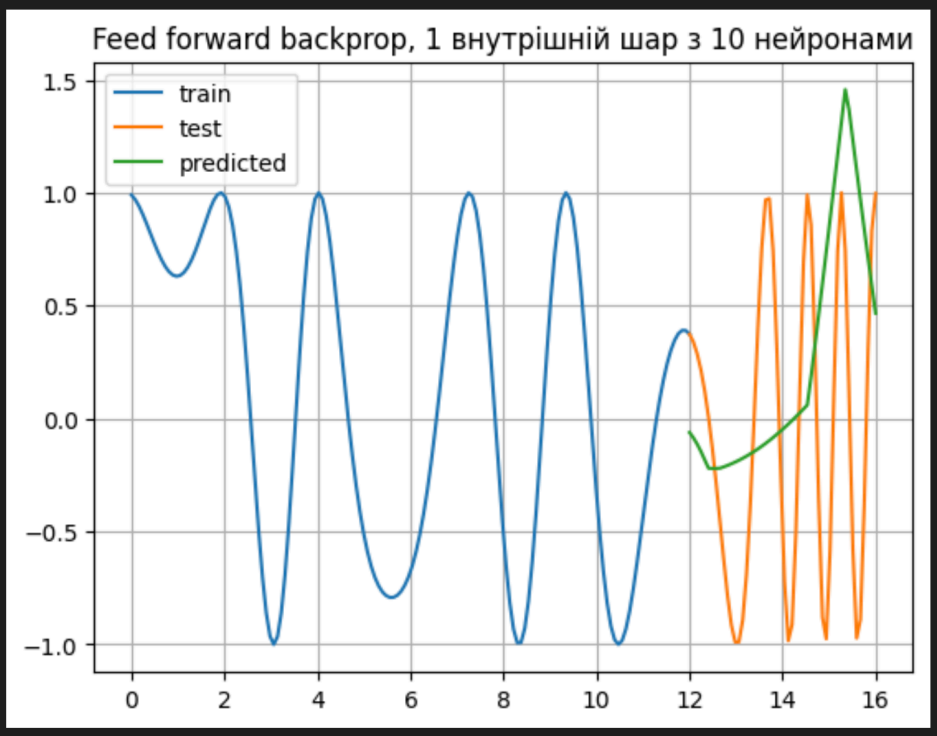
```python
def plot_models(model, neurons_per_layer, title, hidden_layers=None):
    if(hidden_layers):
        obj = model(neurons_per_layer, hidden_layers)
    else:
        obj = model(neurons_per_layer)

    train_values = np.vstack((x_train, y_train)).T
    test_values = np.vstack((x_test, y_test)).T
    if model == elman_backprop:
        train_values = np.expand_dims(train_values, axis=1)
        test_values = np.expand_dims(test_values, axis=1)
    obj.fit(train_values, z_train, epochs=1000, verbose=0)
    predicted_data = obj.predict(test_values)
    model_func_graph(title=title, z_predicted=predicted_data)
    print("\nMSE: "  + str(mean_squared_error(z_test, predicted_data)))

plot_models(feed_forward_backprop, 10, "Feed forward backprop, 1 внутрішній шар з 10 нейронами")
plot_models(feed_forward_backprop, 20, "Feed forward backprop, 1 внутрішній шар з 20 нейронами")
plot_models(cascade_forward_backprop, 20, "cascade – forward backprop, 1 внутрішній шар з 20 нейронами")
plot_models(cascade_forward_backprop, 10, "cascade – forward backprop, 1 внутрішній шар з 20 нейронами", 2)
plot_models(elman_backprop, 15, "elman backprop, 1 внутрішній шар з 15 нейронами")
plot_models(elman_backprop, 5, "elman backprop, 1 внутрішній шар з 15 нейронами", 3)
```

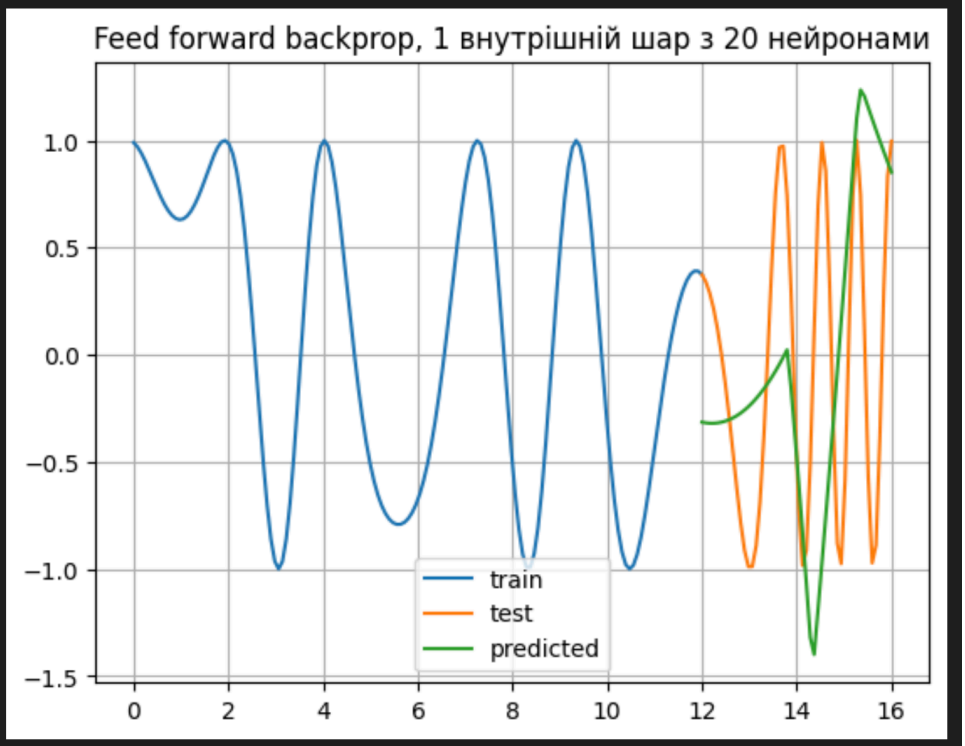
[39] ✓ 17.7s Python

```
2/2 [==============================] - 0s 864us/step
```



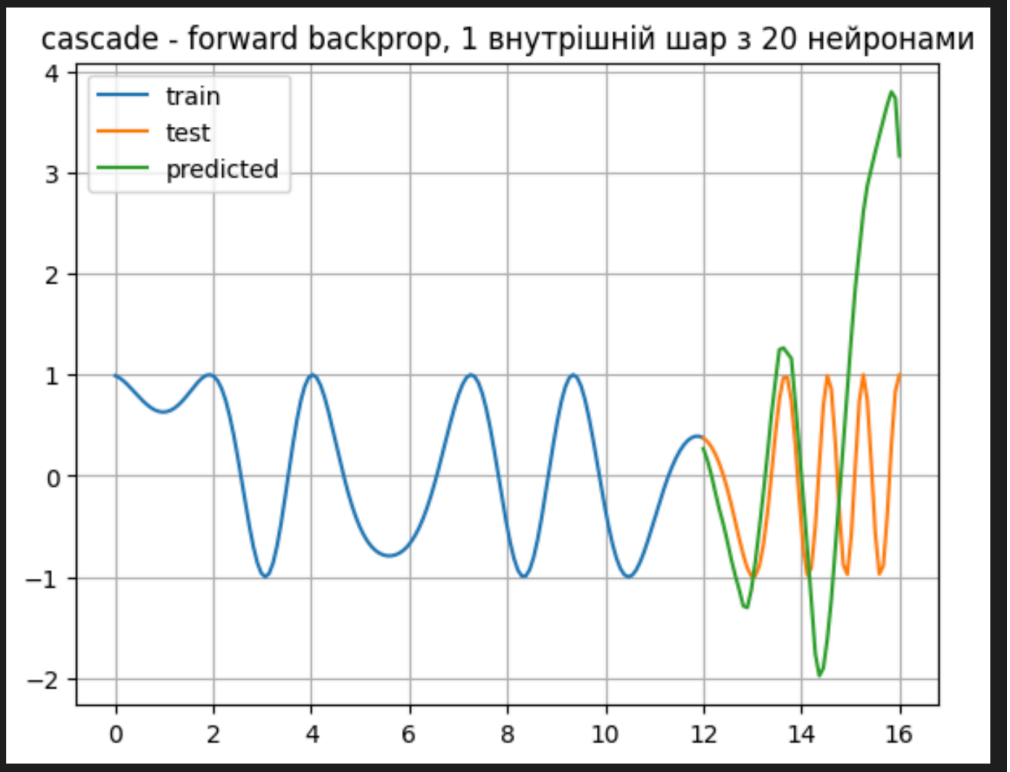

```
MSE: 0.7321474457775463
```

```
2/2 [==============================] - 0s 991us/step
```



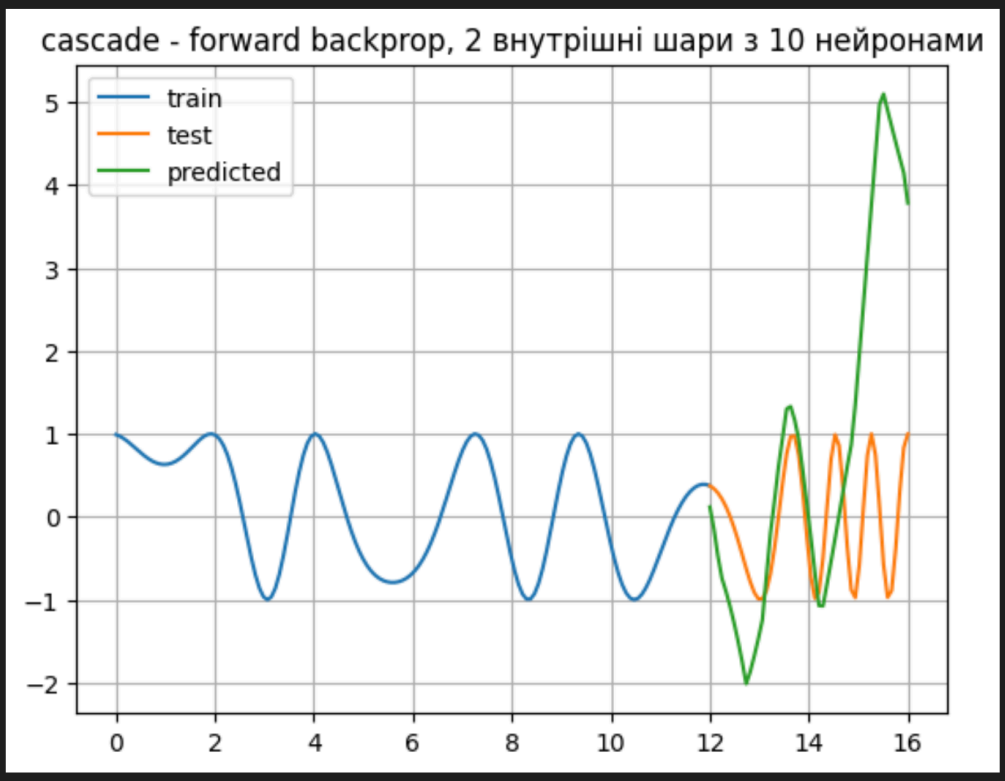

```
MSE: 0.8023460690318233
```

```
2/2 [==============================] - 0s 904us/step
```



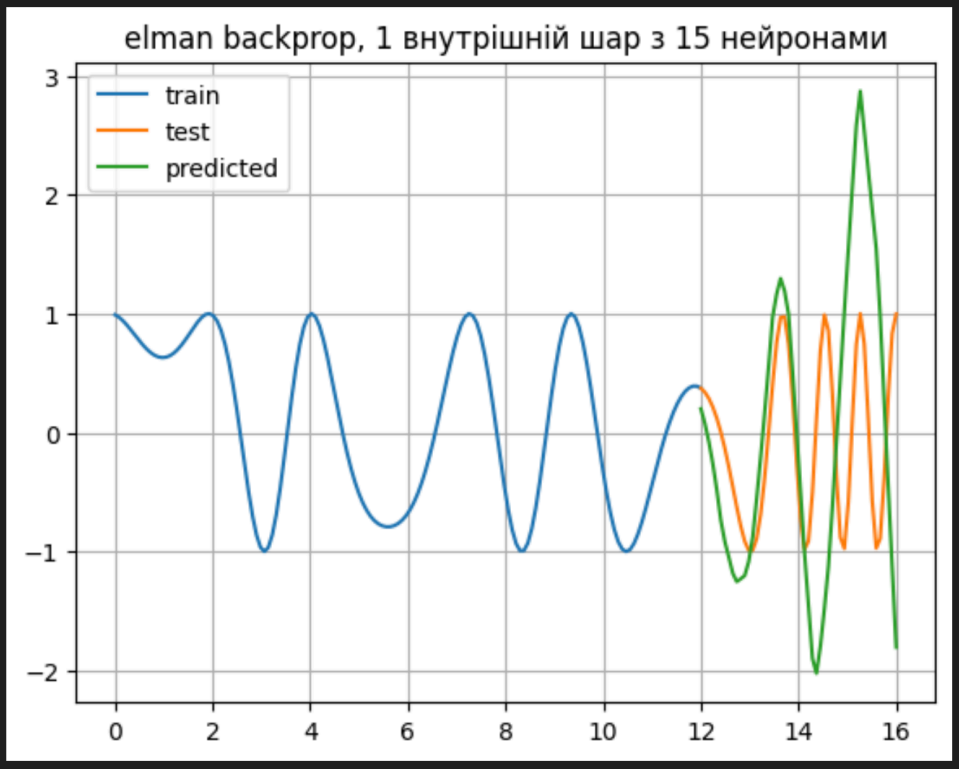

```
MSE: 3.076220412845537
```

```
2/2 [==============================] - 0s 862us/step
```



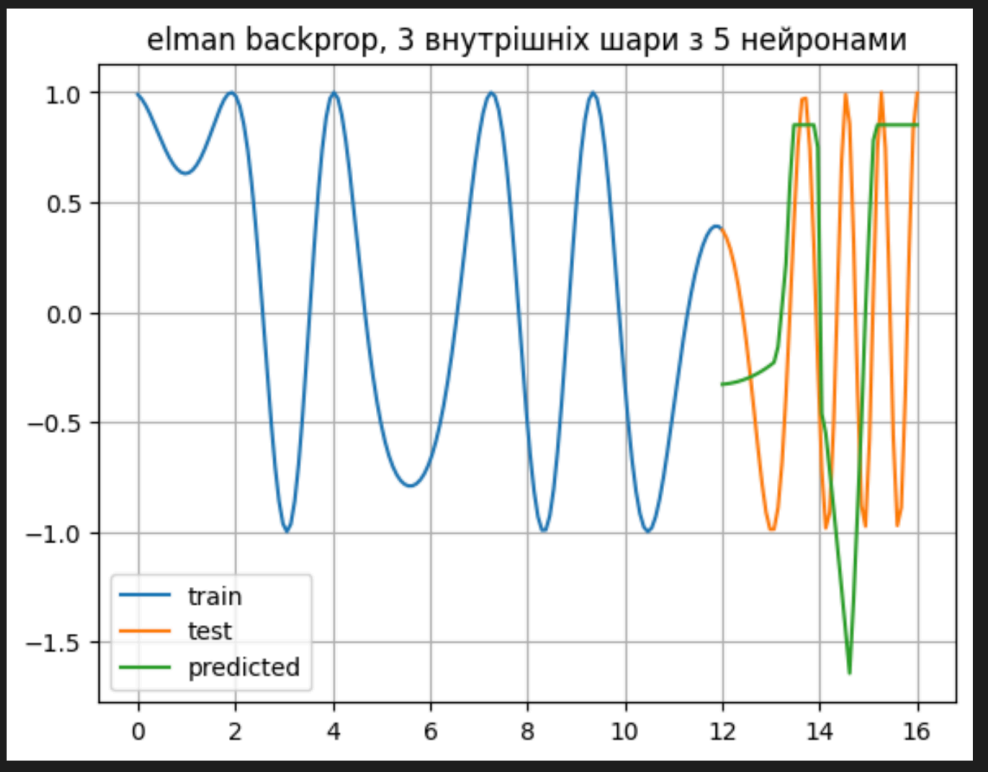

```
MSE: 4.972254690872408
```

```
2/2 [==============================] - 0s 1ms/step
```




```
MSE: 1.7328843741913849
```

```
2/2 [==============================] - 0s 1ms/step
```




```
MSE: 0.8295896911394837
```

Висновок:

Отже, під час виконання даної лабораторної роботи я дослідила структуру та принцип роботи нейронної мережі. З використанням даної нейронної мережі я змоделювала функцію двох змінних. Під час проведення дослідження було використано 3 моделі: feed forward backprop, cascade - forward backprop та elman backprop. З різними параметрами. Дослідження показало, що всі експерименти дали досить невелику похибку, проте найкращого результату з найменшим значенням похибки було досягнуто при використанні моделі Feed forward backprop з 1 внутрішнім шаром з 10 нейронами, найгірший результат же дала модель cascade - forward backprop з 2 внутрішніми шарами з 10 нейронами у кожному.