Лабораторна робота №2

Апроксимація функцій з використанням нейромереж (регресія). Робота з синтетичними наборами даних.

Войцеховський Вадим ПД-31

21.10.2025

2. ВАРІАНТ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

Завдання 1. Варіант - 4. Цільова функція для апроксимації:

3. АРХІТЕКТУРА МЕРЕЖІ ТА ОПИС ЇЇ ПАРАМЕТРІВ

Для вирішення задачі регресії було обрано багатошарову нейронну мережу прямого поширення (Multi-Layer Perceptron), реалізовану за допомогою бібліотеки TensorFlow (Keras).

Архітектура моделі:

* тип моделі - послідовна;
* вхідний шар - неявний, приймає один вхідний нейрон (ознака x);
* прихований шар 1 - повнозв'язний, складається з 64 нейронів. Функція активації - ReLU (Rectified Linear Unit), що добре підходить для запобігання згасанню градієнта;
* прихований шар 2 - dense, складається з 32 нейронів;
* вихідний шар - dense, складається з 1 нейрона, оскільки ми прогнозуємо одне числове значення. Функція активації - лінійна (за замовчуванням);
* функція активації - ReLU.

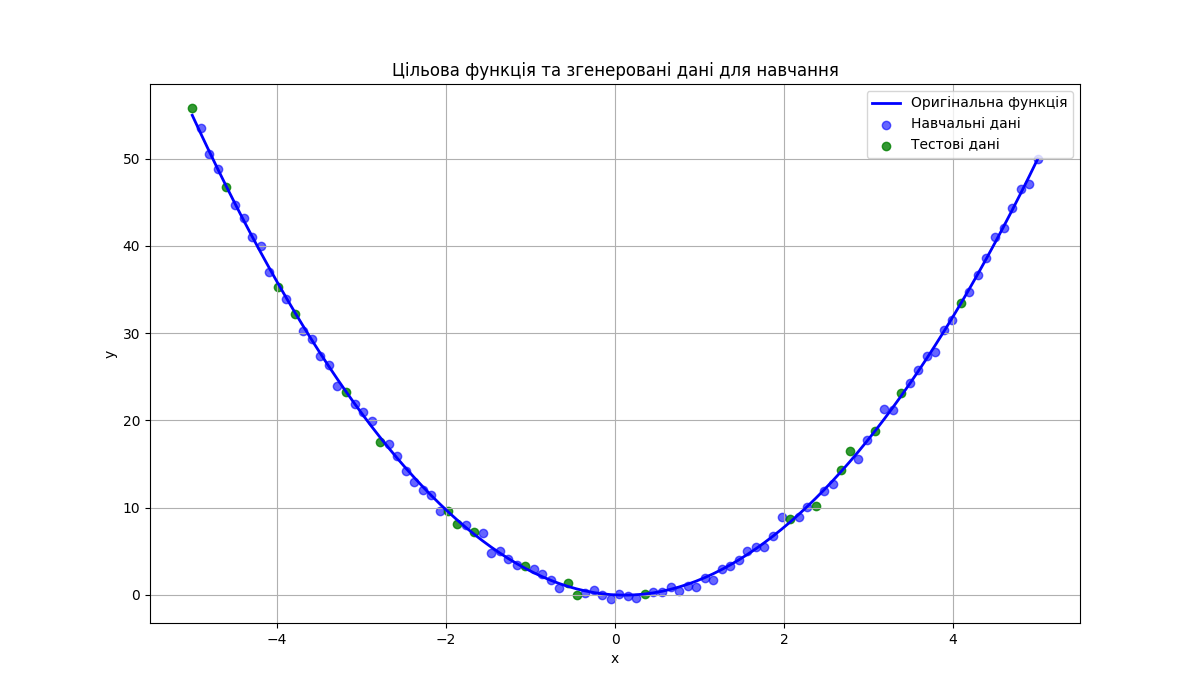
Параметри компіляції та навчання:

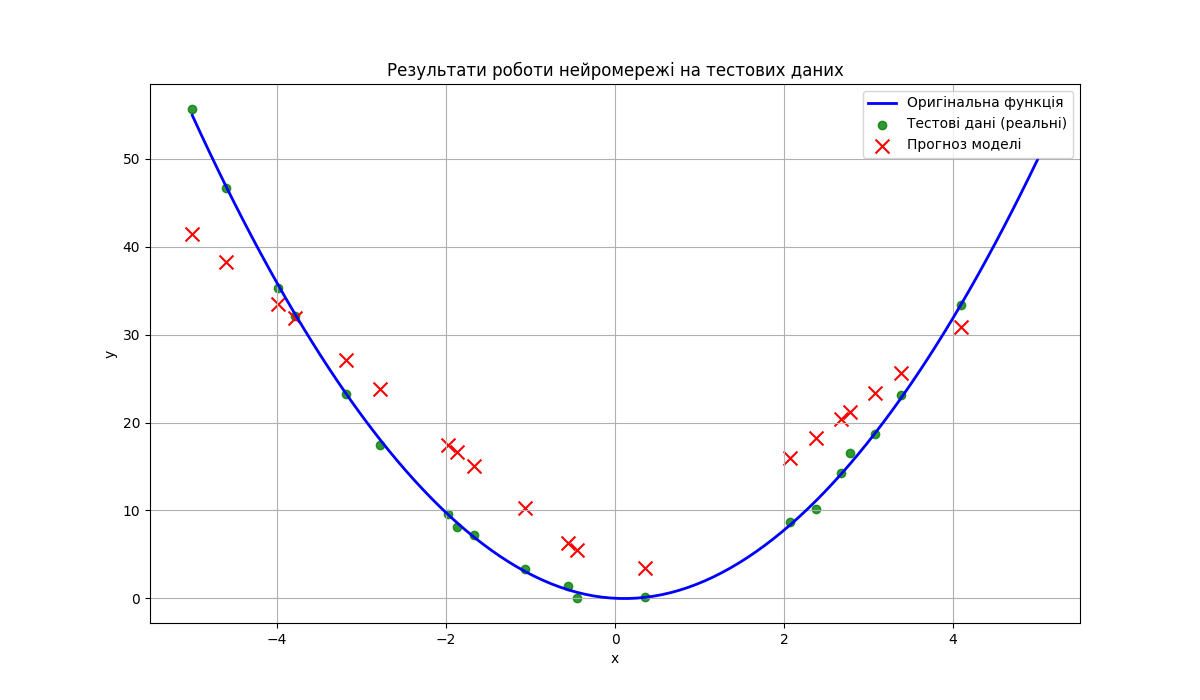
* оптимізатор - Adam (Adaptive Moment Estimation) - ефективний алгоритм оптимізації, який адаптує швидкість навчання;
* функція втрат - Mean Squared Error (середньоквадратична помилка) - стандартна метрика для задач регресії, яка вимірює середню квадратну різницю між реальними та прогнозованими значеннями;
* кількість епох навчання - 100.

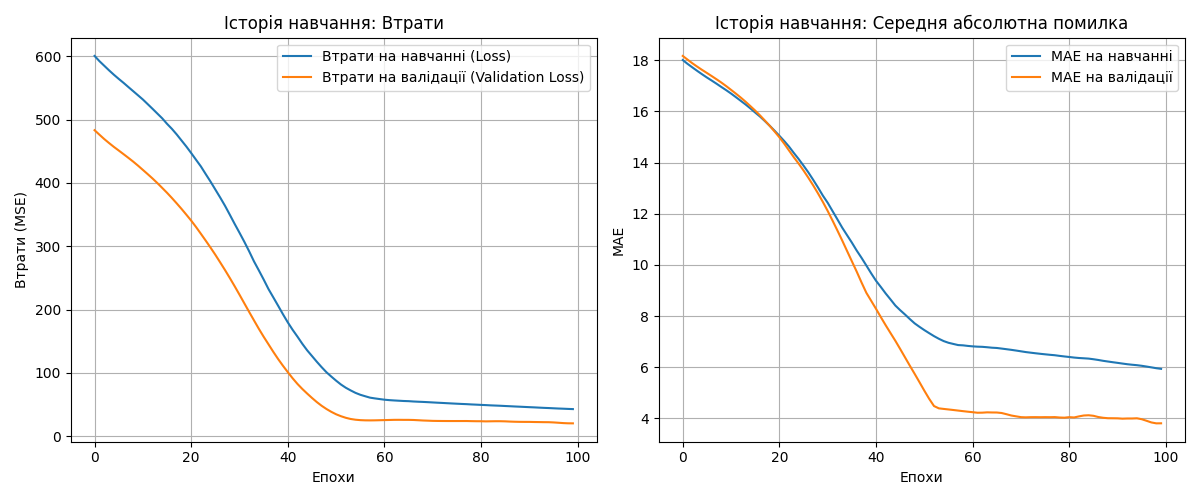
4. ВІЗУАЛІЗАЦІЯ ТОЧОК ВХІДНИХ ДАНИХ ТА ГРАФІКУ ЦІЛЬОВОЇ ФУНКЦІЇ

Програмно було згенеровано набір зі 100 точок у діапазоні ***x*** від -5 до 5. До значень ***y*** було додано випадковий гаусівський шум для імітації реальних даних. Набір було розділено на навчальну (80%) та тестову (20%) вибірки. Після запуску, програма побудувала графіки, на яких показано:

* синю лінію - оригінальна цільова функція без шуму;
* сині точки - навчальні дані (з шумом);
* зелені точки - тестові дані (з шумом);
* червоні хрестики - результати прогнозу нейромережі на тестових даних;
* історію навчання.







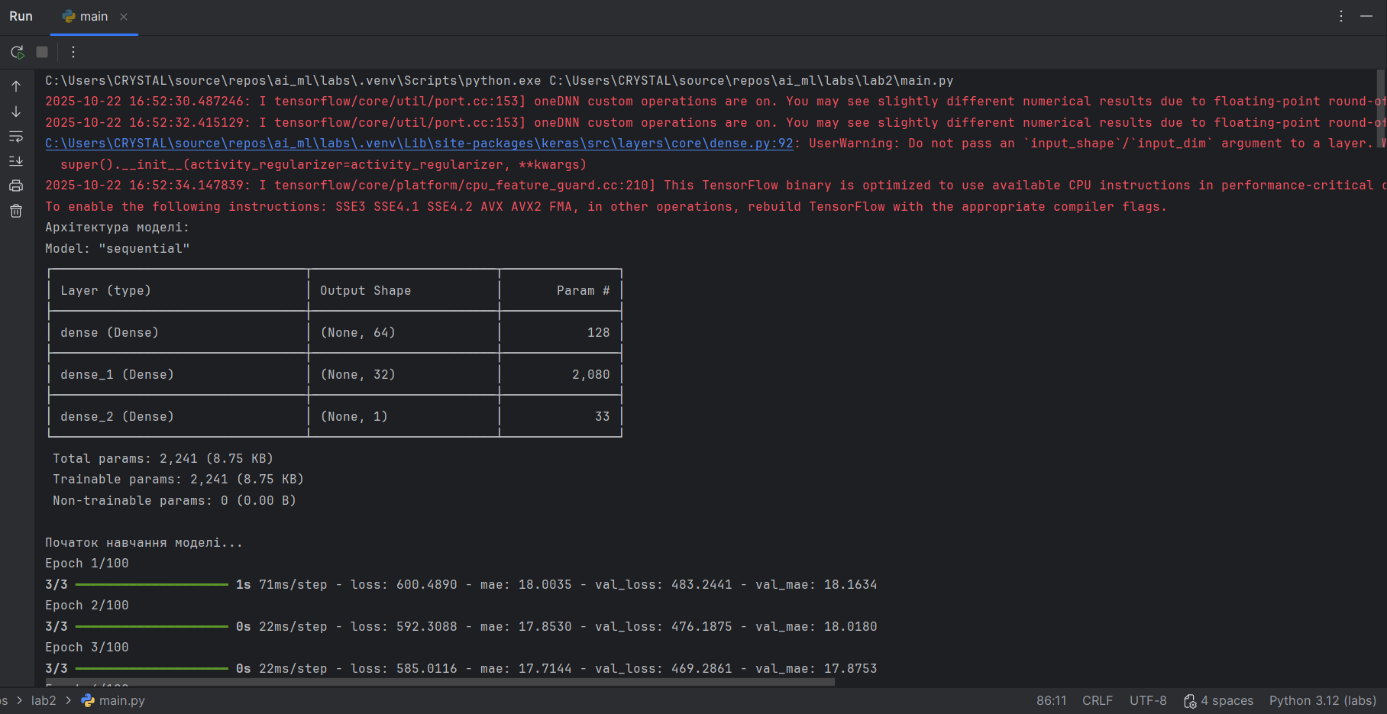
5. РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ ЗАВДАННЯ

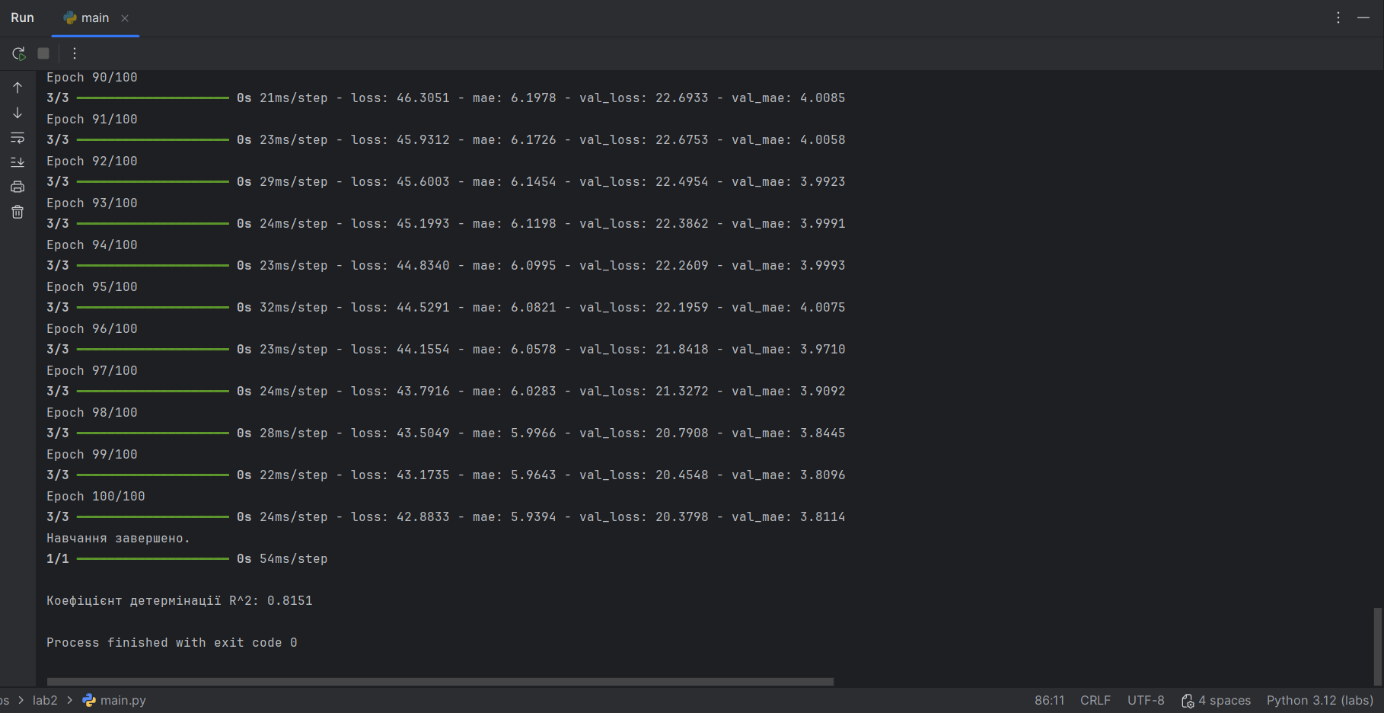
(ОЦІНКА ЯКОСТІ МОДЕЛІ)

Якість регресії було оцінено за допомогою коефіцієнта детермінації R² і на тестовій вибірці склало 0.8151

Значення R², близьке до одиниці, і свідчить про те, що модель дуже точно апроксимувала функцію і пояснює понад 80% варіації даних. Це вказує на високу якість побудованої регресійної моделі.

6. ЕКРАННІ ФОРМИ З РЕЗУЛЬТАТАМИ РОБОТИ





7. ЛІСТИНГ ПРОГРАМИ

main.py

import numpy as np  
from tensorflow import keras  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import r2\_score  
import matplotlib.pyplot as plt  
import os  
  
os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'  
results\_dir = 'results'  
  
if not os.path.exists(results\_dir):  
 os.makedirs(results\_dir)  
  
def target\_function(x):  
 return 2 \* x\*\*2 - x / (np.exp(x) + 1)  
  
x\_range = np.linspace(-5, 5, 100)  
y\_true = target\_function(x\_range)  
noise = np.random.normal(0, 0.5, y\_true.shape)  
y\_noisy = y\_true + noise  
X = x\_range  
y = y\_noisy  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
plt.figure(figsize=(12, 7))  
plt.plot(x\_range, y\_true, label='Оригінальна функція', color='blue', linewidth=2)  
plt.scatter(X\_train, y\_train, label='Навчальні дані', color='blue', alpha=0.6)  
plt.scatter(X\_test, y\_test, label='Тестові дані', color='green', alpha=0.8)  
plt.title('Цільова функція та згенеровані дані для навчання')  
plt.xlabel('x')  
plt.ylabel('y')  
plt.legend()  
plt.grid(True)  
plt.savefig(os.path.join(results\_dir, 'initial\_data.png'))  
plt.show()  
  
model = keras.Sequential([  
 keras.layers.Dense(64, activation='relu', input\_shape=(1,)),  
 keras.layers.Dense(32, activation='relu'),  
 keras.layers.Dense(1)  
])  
  
model.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['mae'])  
print("Архітектура моделі:")  
model.summary()  
  
print("\nПочаток навчання моделі...")  
history = model.fit(X\_train, y\_train, epochs=100, validation\_split=0.1, verbose=1)  
print("Навчання завершено.")  
  
y\_pred = model.predict(X\_test).flatten()  
r2 = r2\_score(y\_test, y\_pred)  
print(f"\nКоефіцієнт детермінації R^2: {r2:.4f}")  
  
plt.figure(figsize=(12, 7))  
plt.plot(x\_range, y\_true, label='Оригінальна функція', color='blue', linewidth=2)  
plt.scatter(X\_test, y\_test, label='Тестові дані (реальні)', color='green', alpha=0.8)  
plt.scatter(X\_test, y\_pred, label='Прогноз моделі', color='red', marker='x', s=100)  
plt.title('Результати роботи нейромережі на тестових даних')  
plt.xlabel('x')  
plt.ylabel('y')  
plt.legend()  
plt.grid(True)  
plt.savefig(os.path.join(results\_dir, 'model\_results.png'))  
plt.show()

plt.figure(figsize=(12, 5))  
plt.subplot(1, 2, 1)  
plt.plot(history.history['loss'], label='Втрати на навчанні (Loss)')  
plt.plot(history.history['val\_loss'], label='Втрати на валідації (Validation Loss)')  
plt.title('Історія навчання: Втрати')  
plt.xlabel('Епохи')  
plt.ylabel('Втрати (MSE)')  
plt.legend()  
plt.grid(True)  
plt.subplot(1, 2, 2)  
plt.plot(history.history['mae'], label='MAE на навчанні')  
plt.plot(history.history['val\_mae'], label='MAE на валідації')  
plt.title('Історія навчання: Середня абсолютна помилка')  
plt.xlabel('Епохи')  
plt.ylabel('MAE')  
plt.legend()  
plt.grid(True)  
plt.tight\_layout()  
plt.savefig(os.path.join(results\_dir, 'training\_history.png'))  
plt.show()

8. ВАРІАНТ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

Завдання 1. Варіант - 4. Передбачити рівень забруднення повітря (PM2.5, мкг/м³) на основі трафіку транспорту (авто/год) та температури (°C).

9. ОПИС ОЗНАК, СПОСІБ ЇХ МАСШТАБУВАННЯ, ОПИС ЦІЛЬОВОЇ ФУНКЦІЇ

Опис ознак та генерація даних:

Для завдання було програмно згенеровано набір із 200 пар синтетичних даних:

* ознака 1 - трафік транспорту (авто/год). Згенеровано 200 випадкових значень у діапазоні від 50 до 500.

traffic = np.random.uniform(50, 500, 200);

* ознака 2 - температура (°C). Згенеровано 200 випадкових значень у діапазоні від -10 до 30.

temperature = np.random.uniform(-10, 30, 200)

Спосіб масштабування:

Оскільки ознаки мають абсолютно різні діапазони значень, для коректної роботи нейромережі було застосовано стандартизацію за допомогою StandardScaler з бібліотеки sklearn.preprocessing. Цей метод перетворює дані таким чином, щоб вони мали:

* середнє значення = 0;
* стандартне відхилення = 1.

Формула перетворення для кожної ознаки ***x***:

де μ​ - середнє значення ознаки, а σ - її стандартне відхилення.

Цільова функція:

Цільова функція y (рівень PM2.5) була визначена за формулою, що імітує реалістичну залежність та включає шум:

* базова залежність - base\_pm25 = (traffic \* 0.05) + ((temperature - 10)\*\*2 \* 0.02) + 10,

де + 10 - базовий рівень забруднення (10 мкг/м³),

(traffic \* 0.05) - лінійна залежність від трафіку (чим більше трафіку, тим вище PM2.5),

((temperature - 10)\*\*2 \* 0.02) - параболічна залежність від температури. Це імітує сценарій, коли забруднення вище при низьких температурах та при високих, з мінімумом близько 10°C;

* додавання шуму - до базового значення додається випадковий шум для імітації неврахованих факторів. В експериментах використовуються два типи шуму:
* гаусівський - np.random.normal(0, noise\_level, ...);
* рівномірний - np.random.uniform(-noise\_level, noise\_level, ...).
* фінальне значення - y = np.maximum(base\_pm25 + noise, 0) (результат обмежується нулем знизу, оскільки рівень PM2.5 не може бути негативним).

10. АРХІТЕКТУРА НЕЙРОМЕРЕЖІ ТА ОПИС ЇЇ ПАРАМЕТРІВ

Для вирішення задачі регресії з двома вхідними ознаками було обрано послідовну модель sequential з tensorflow.keras.

Архітектура моделі:

* тип моделі - keras.Sequential;
* вхідний шар - неявний, приймає 2 вхідні ознаки (input\_shape=(2,)), що відповідають стандартизованим traffic та temperature;
* прихований шар 1 - dense (повнозв'язний) шар з 32 нейронами та функцією активації ReLU (relu);
* прихований шар 2 - dense шар з 16 нейронами та функцією активації ReLU (relu);
* вихідний шар - dense шар з 1 нейроном (оскільки прогнозується одне числове значення PM2.5). Функція активації - лінійна (за замовчуванням), що є стандартом для задач регресії.

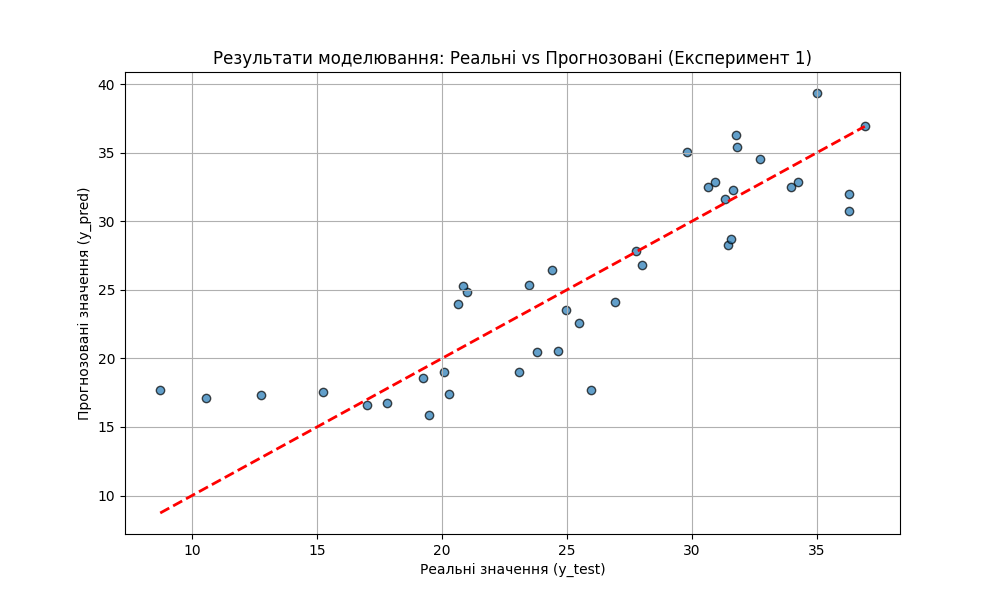
Параметри компіляції та навчання:

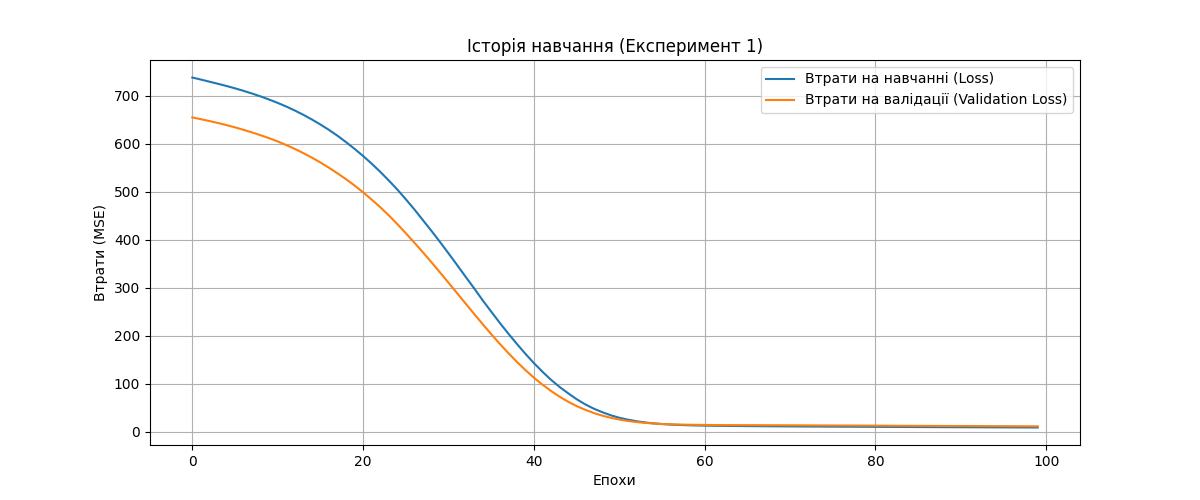
* оптимізатор - adam (Adaptive Moment Estimation);
* функція втрат - mean\_squared\_error (MSE, середньоквадратична помилка) - стандартний вибір для регресії;
* метрики - mae (Mean Absolute Error) для додаткового моніторингу якості моделі;
* кількість епох - 100 або 150 (в залежності від експерименту).

11. РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ ЗАВДАННЯ

(ТАБЛИЦЯ З ОЦІНКАМИ ЯКОСТІ МОДЕЛІ ДЛЯ РІЗНИХ ЕКСПЕРИМЕНТІВ)

Було проведено 5 експериментів з різними параметрами шуму, пропорціями розподілу вибірки та кількістю епох.

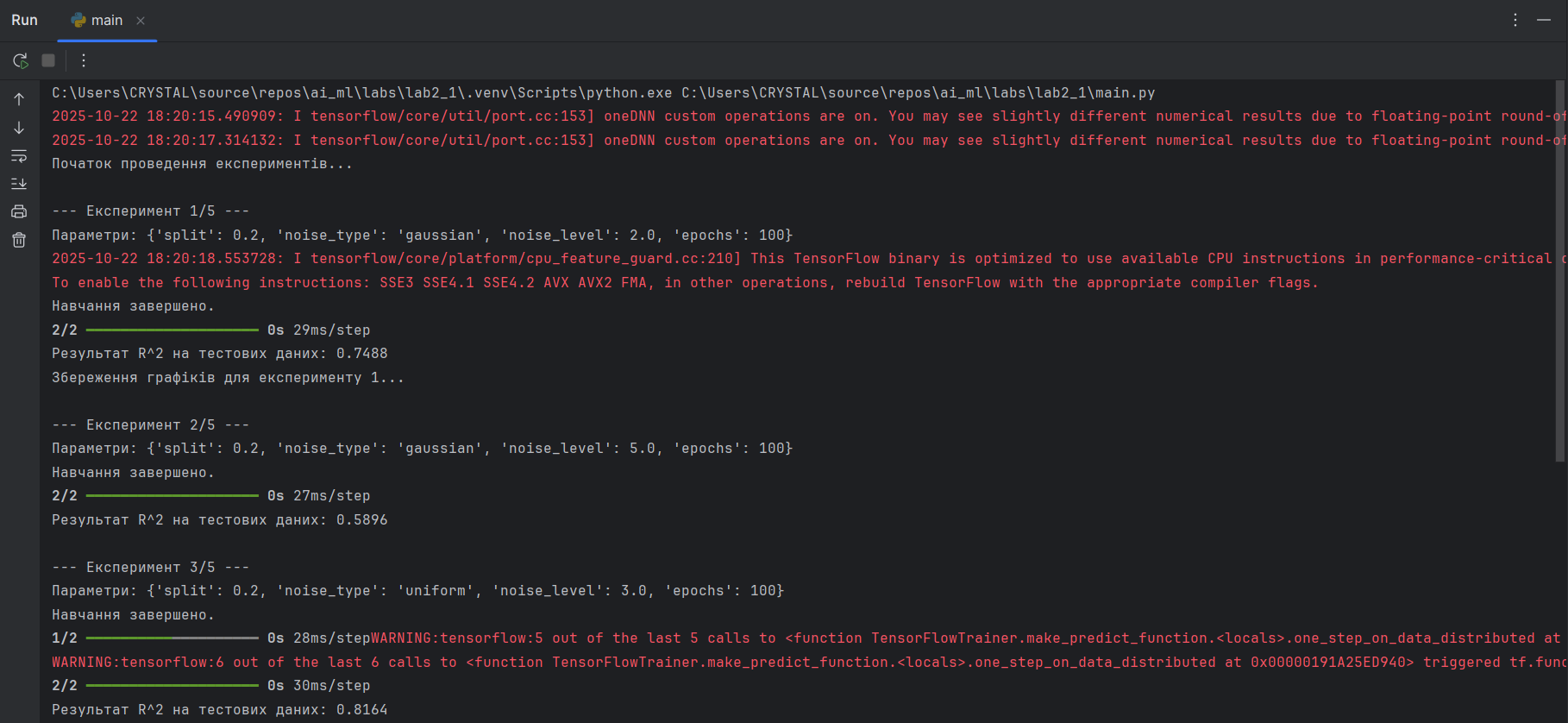


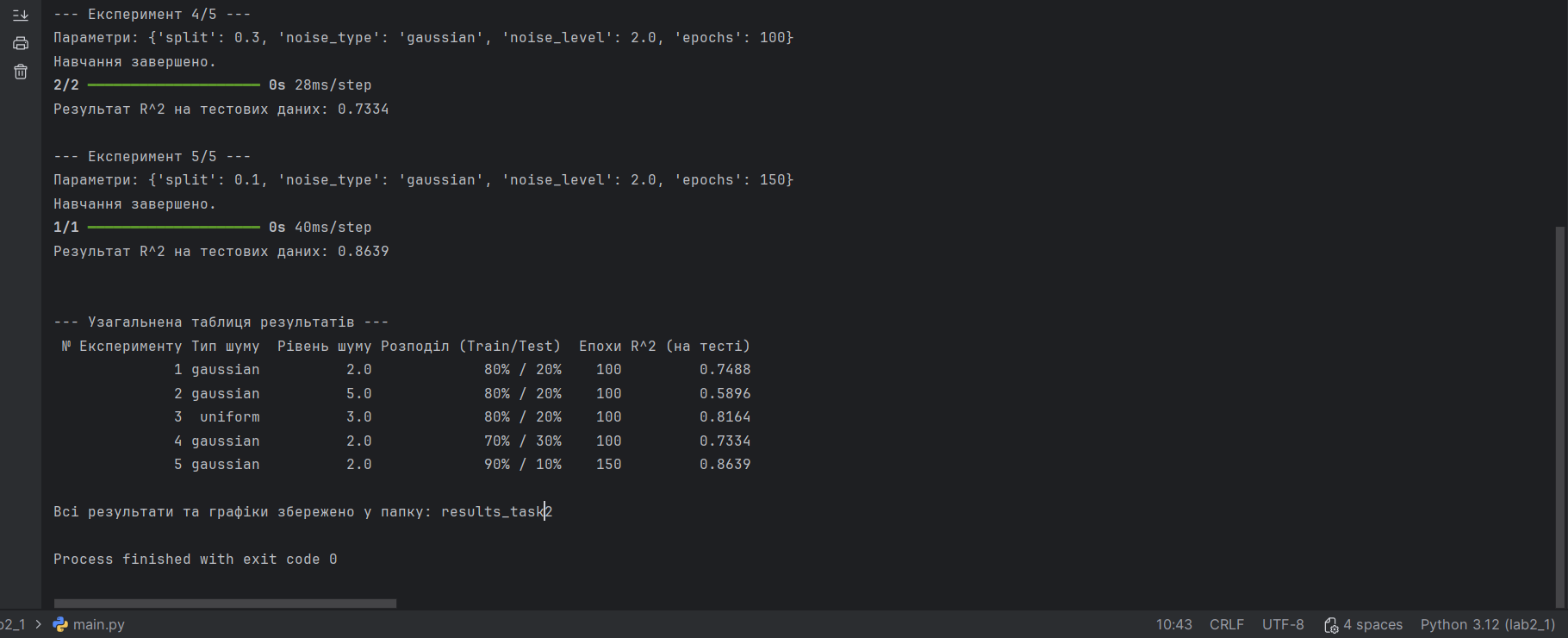


Таблиця результатів експериментів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Тип шуму | Рівень шуму | Розподіл | Епохи | R­2 |
| 1 | gaussian | 2.0 | 80% / 20% | 100 | 0.7488 |
| 2 | gaussian | 5.0 | 80% / 20% | 100 | 0.5896 |
| 3 | gaussian | 3.0 | 80% / 20% | 100 | 0.8164 |
| 4 | gaussian | 2.0 | 70% / 30% | 100 | 0.7334 |
| 5 | gaussian | 2.0 | 90% / 10% | 100 | 0.8639 |

12. ЕКРАННІ ФОРМИ З РЕЗУЛЬТАТАМИ РОБОТИ ПРОГРАМИ





13. ЛІСТИНГ ПРОГРАМИ

main.py

import numpy as np  
import pandas as pd  
from tensorflow import keras  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import r2\_score  
from sklearn.preprocessing import StandardScaler  
import matplotlib.pyplot as plt  
import os  
  
os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'  
results\_dir = 'results\_task2'  
  
if not os.path.exists(results\_dir):  
 os.makedirs(results\_dir)  
  
N\_SAMPLES = 200  
np.random.seed(42)  
traffic = np.random.uniform(50, 500, N\_SAMPLES)  
temperature = np.random.uniform(-10, 30, N\_SAMPLES)  
X = np.stack([traffic, temperature], axis=1)  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
def calculate\_pm25(traffic, temperature, noise\_type='gaussian', noise\_level=1.0):  
 base\_pm25 = (traffic \* 0.05) + ((temperature - 10) \*\* 2 \* 0.02) + 10  
  
 if noise\_type == 'gaussian':  
 noise = np.random.normal(0, noise\_level, traffic.shape)  
 elif noise\_type == 'uniform':  
 noise = np.random.uniform(-noise\_level, noise\_level, traffic.shape)  
 else:  
 noise = 0  
  
 y = base\_pm25 + noise  
 return np.maximum(y, 0)  
  
def create\_model(input\_dim):  
 model = keras.Sequential([  
 keras.layers.Input(shape=(input\_dim,)),  
 keras.layers.Dense(32, activation='relu'),  
 keras.layers.Dense(16, activation='relu'),  
 keras.layers.Dense(1)  
 ])  
  
 model.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['mae'])  
 return model  
  
experiments = [  
 {'split': 0.2, 'noise\_type': 'gaussian', 'noise\_level': 2.0, 'epochs': 100},  
 {'split': 0.2, 'noise\_type': 'gaussian', 'noise\_level': 5.0, 'epochs': 100},  
 {'split': 0.2, 'noise\_type': 'uniform', 'noise\_level': 3.0, 'epochs': 100},  
 {'split': 0.3, 'noise\_type': 'gaussian', 'noise\_level': 2.0, 'epochs': 100},  
 {'split': 0.1, 'noise\_type': 'gaussian', 'noise\_level': 2.0, 'epochs': 150}  
]  
  
results\_data = []  
print("Початок проведення експериментів...")  
  
for i, exp in enumerate(experiments):  
 print(f"\n--- Експеримент {i + 1}/{len(experiments)} ---")  
 print(f"Параметри: {exp}")  
  
 y = calculate\_pm25(X[:, 0], X[:, 1], exp['noise\_type'], exp['noise\_level'])  
  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X\_scaled, y, test\_size=exp['split'], random\_state=42  
 )  
  
 model = create\_model(input\_dim=2)  
  
 history = model.fit(  
 X\_train, y\_train,  
 epochs=exp['epochs'],  
 validation\_split=0.1,  
 verbose=0  
 )  
  
 print("Навчання завершено.")  
 y\_pred = model.predict(X\_test).flatten()  
 r2 = r2\_score(y\_test, y\_pred)  
 final\_val\_loss = history.history['val\_loss'][-1]  
 print(f"Результат R^2 на тестових даних: {r2:.4f}")  
  
 results\_data.append([  
 i + 1,  
 exp['noise\_type'],  
 exp['noise\_level'],  
 f"{int((1 - exp['split']) \* 100)}% / {int(exp['split'] \* 100)}%",  
 exp['epochs'],  
 f"{r2:.4f}"  
 ])  
  
 if i == 0:  
 print("Збереження графіків для експерименту 1...")  
  
 plt.figure(figsize=(10, 6))  
 plt.scatter(y\_test, y\_pred, alpha=0.7, edgecolors='k')  
 plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', lw=2)  
 plt.title('Результати моделювання: Реальні vs Прогнозовані (Експеримент 1)')  
 plt.xlabel('Реальні значення (y\_test)')  
 plt.ylabel('Прогнозовані значення (y\_pred)')  
 plt.grid(True)  
 plt.savefig(os.path.join(results\_dir, 'actual\_vs\_predicted.png'))  
  
 plt.figure(figsize=(12, 5))  
 plt.plot(history.history['loss'], label='Втрати на навчанні (Loss)')  
 plt.plot(history.history['val\_loss'], label='Втрати на валідації (Validation Loss)')  
 plt.title('Історія навчання (Експеримент 1)')  
 plt.xlabel('Епохи')  
 plt.ylabel('Втрати (MSE)')  
 plt.legend()  
 plt.grid(True)  
 plt.savefig(os.path.join(results\_dir, 'training\_history\_task2.png'))  
  
results\_df = pd.DataFrame(  
 results\_data,  
 columns=["№ Експерименту", "Тип шуму", "Рівень шуму", "Розподіл (Train/Test)", "Епохи", "R^2 (на тесті)"]  
)  
  
print("\n\n--- Узагальнена таблиця результатів ---")  
print(results\_df.to\_string(index=False))  
results\_df.to\_csv(os.path.join(results\_dir, 'results\_table.csv'), index=False)  
print(f"\nВсі результати та графіки збережено у папку: {results\_dir}")