**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**ІКНІ**

Кафедра **ПЗ**



**ЗВІТ**

до лабораторної роботи №2

**на тему:** *“Розгортання попередньо тренованої моделі GPT в локальному середовищі”*

**з дисципліни** *“Штучний інтелект в ігрових застосунках”*

**Лектор:**

асис. каф. ПЗ

Бауск О. Є.

**Виконав:**

ст. гр. ПЗ-33

Юшкевич. А.І.

**Прийняв:**

ст. викл. каф. ПЗ

Бауск О. Є.

«\_\_» \_\_\_\_\_\_\_\_\_ 2024р.

∑=\_\_\_\_\_

Львів – 2024

**Тема роботи:** Розгортання попередньо тренуваної моделі GPT в локальному середовищі.

**Мета роботи:** Ознайомитись з основами функціонування системи-обгортки для моделей глибокого навчання OLLAMA, навчитися розгортати навчені моделі.

# Теоретичні відомості

**Що таке Ollama?**

Ollama — це інструмент з відкритим вихідним кодом, призначений для простого та швидкого розгортання моделей штучного інтелекту (AI), зокрема великих мовних моделей (LLM), у локальному середовищі. Він дозволяє користувачам легко завантажувати, зберігати, керувати та запускати попередньо треновані моделі без необхідності складних налаштувань чи глибоких знань у сфері машинного навчання.

**Як працює Ollama?**

Ollama працює як система-обгортка (wrapper), яка забезпечує зручний інтерфейс для взаємодії з моделями глибокого навчання. Він автоматизує процес завантаження моделей з віддалених репозиторіїв, їх зберігання, керування версіями та запуску. Ollama підтримує різні популярні моделі, такі як GPT, LLaMA, DeepSeek та інші, дозволяючи користувачам швидко розгорнути їх локально та використовувати для вирішення різноманітних задач, включаючи генерацію тексту, відповіді на запитання, створення чат-ботів тощо.

**Що таке попередньо треновані моделі (pretrained models)?**

Попередньо треновані моделі — це моделі машинного навчання, які вже були навчені на великих наборах даних для виконання певних загальних задач. Наприклад, мовні моделі, такі як GPT, тренуються на величезних об'ємах текстових даних з інтернету, книг, статей тощо. Після такого тренування модель здатна розуміти контекст, генерувати текст, відповідати на запитання та виконувати інші завдання, пов'язані з обробкою природної мови. Користувачі можуть використовувати ці моделі без додаткового навчання або ж донавчати їх (finetuning) на власних специфічних наборах даних для покращення результатів у конкретних задачах.

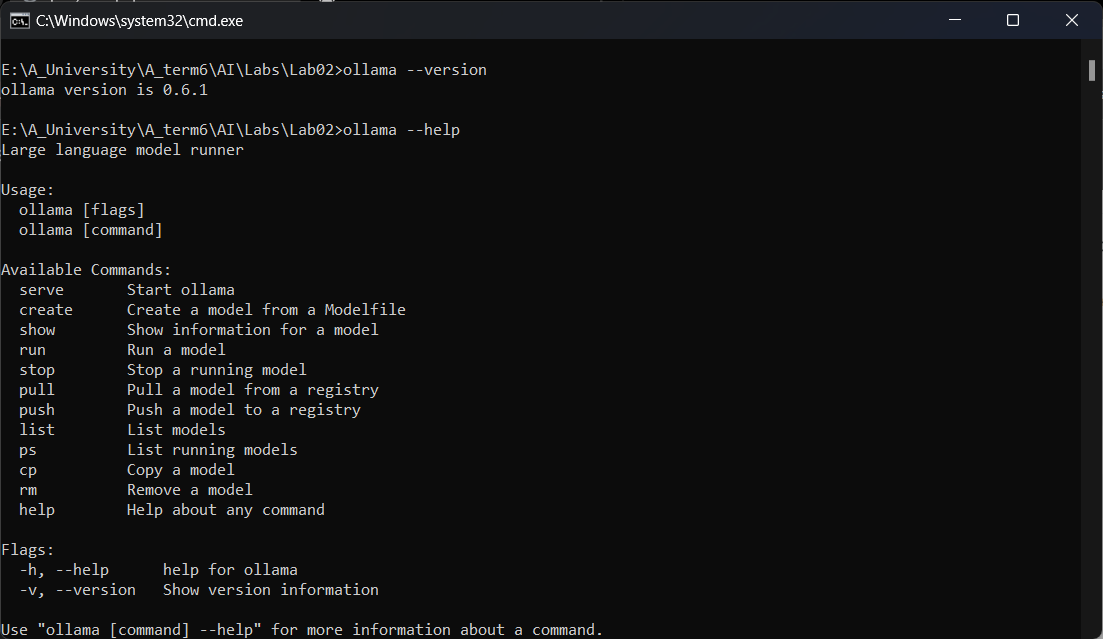
**Як зберігаються та керуються моделями в Ollama?**

Ollama зберігає моделі у вигляді спеціальних контейнерів, які містять усі необхідні файли та параметри для роботи моделі. Користувачі можуть легко завантажувати моделі з віддалених репозиторіїв за допомогою простих команд (наприклад, ollama pull <назва\_моделі> ). Після завантаження модель зберігається локально, що дозволяє використовувати її без підключення до інтернету. Ollama також надає зручні команди для перегляду списку доступних локально моделей ( ollama list ), видалення непотрібних моделей ( ollama rm <назва\_моделі> ) та запуску моделей для виконання конкретних задач. Таким чином, Ollama значно спрощує процес роботи з великими мовними моделями, роблячи їх доступними для широкого кола користувачів, включаючи тих, хто не має глибоких знань у сфері штучного інтелекту та машинного навчання.

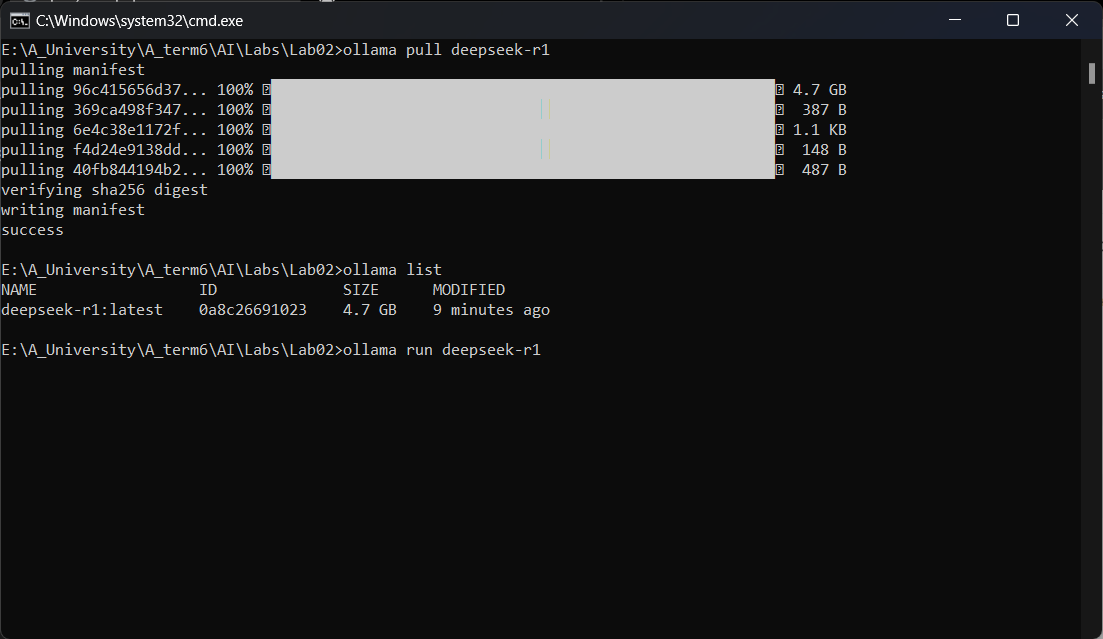
# Завдання

1. Встановити систему розгортання моделей глибокого навчання Ollama.
2. Розгорнути локально LLM модель DeepSeek-R1 (Варіант з 1B параметрів).
3. Протестувати локальне розгортання моделі.
4. Дослідити налаштування моделей при локальному розгортанні, зрозуміти різницю між використанням онлайн- сервісів з LLM моделями та власного деплоймента.

# Хід роботи



**­***Рис. 1. Втановлення Ollama*



**­***Рис. 2. Завантаження моделі Deepseek*

**Код отриманий за запитом “Write a Python script to convert JPEG images to PNG”:**

def convert\_jpeg\_to\_png(input\_dir, output\_dir, quality=100):

try:

# Iterate through all files in input directory

for filename in os.listdir(input\_dir):

# Check if the file has .jpeg extension and is a file (not directory)

if filename.lower().endswith('.jpeg') and os.path.isfile(os.path.join(input\_dir, filename)):

# Open the image

img = Image.open(os.path.join(input\_dir, filename))

# Convert to PNG with optional quality setting

img.save(

os.path.join(output\_dir, f"{filename}.png"),

format="PNG",

quality=quality

)

# Optional: Remove the original JPEG file after conversion (to keep directory structure)

if os.path.isfile(os.path.join(input\_dir, filename)):

os.remove(os.path.join(input\_dir, filename))

print(f"Conversion completed. All .jpeg files converted to .png in {output\_dir}.")

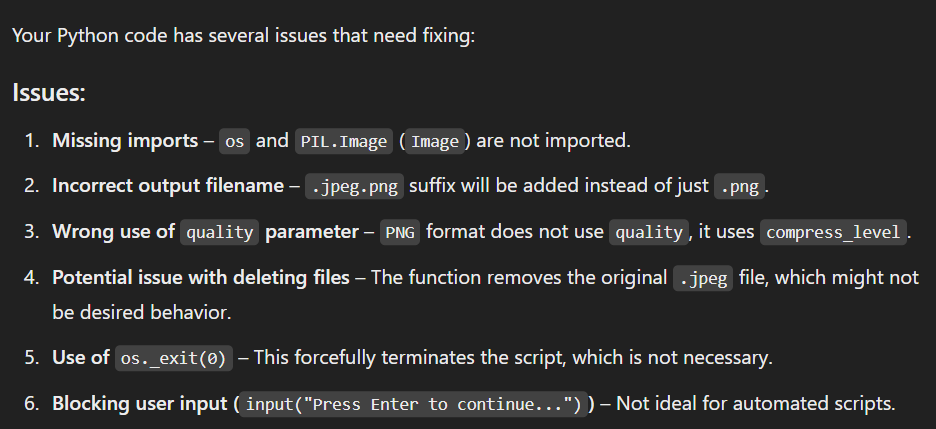
except Exception as e:

print(f"An error occurred during conversion: {str(e)}")

# Option to stop or continue processing

input("Press Enter to continue...")

os.\_exit(0)

****

**­***Рис. 3. Валідація отриманого коду*

**Питання:**

1. **Скільки параметрів має модель, яку ви використовуєте?**

Модель яку я використовую має 7 мільярдів параметрів.

1. **Як, на вашу думку, зміняться результати роботи моделі, якщо використовувати модель з більшою кількістю параметрів? Меншою кількістю параметрів?**
2. Менше параметрів (наприклад, модель 3B):

* Швидший висновок (з меншими обчислювальними витратами).
* Зменшене використання пам’яті (може працювати на невеликих пристроях).
* Нижча точність, менша здатність міркувати та слабше узагальнення.

1. Більше параметрів (наприклад, модель 13B або 65B):

* Краща точність, сильніша аргументація та більш детальне розуміння.
* Потрібна більша обчислювальна потужність (наприклад, кращі графічні процесори, більше оперативної пам’яті).
* Ризик переобладнання даних перед навчанням, якщо вони не налаштовані належним чином.

1. **Які стадії/режими роботи моделі ви використовуєте саме на своїй локальній машині, при виконанні команд ollama run та надання текстових промптів?**1) Згодовування прикладів  
   2) Постановка запиту  
   3) Перевірка отриманих даних  
   4) Коригування у разі невдачі

**Код отриманий за запитом “Write a Python script to convert JPEG images to PNG” (1.5b):**

from PIL import Image

import os

def convert\_jpeg\_to\_png():

"""Converts JPEG images to PNG files in a directory."""

print("Converting JPEG to PNG...")

# Get the list of files in the directory

input\_dir = 'path\_to\_directory'

for file\_name in os.listdir(input\_dir):

if file\_name.lower().endswith('.jpeg'):

# Open the image

img = Image.open(os.path.join(input\_dir, file\_name))

try:

# Convert to PNG with some quality level (0.95 is good)

img.save(os.path.join(input\_dir, f'{file\_name}.png'), 'PNG', quality=0.95)

print(f"Successfully converted: {file\_name}")

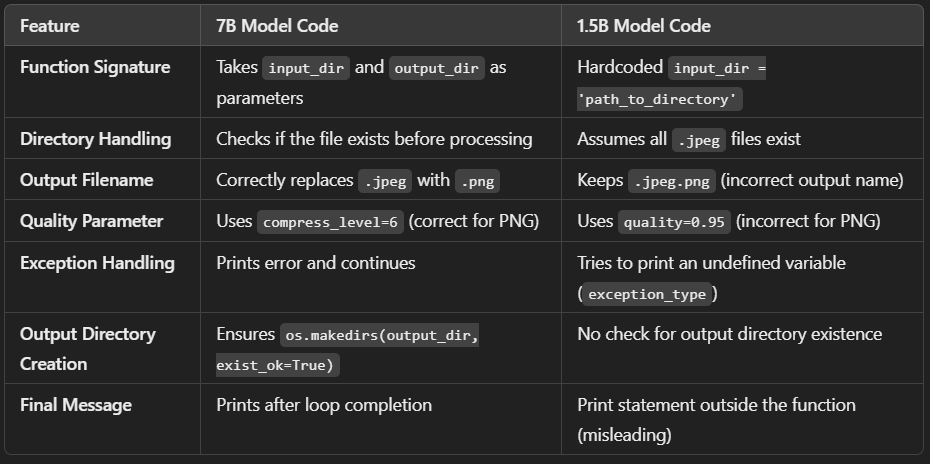
except Exception as e:

print(f"Error converting file: {file\_name}.", exception\_type=str(e))

print("Conversion complete.")

if \_\_name\_\_ == "\_\_main\_\_":

convert\_jpeg\_to\_png()

****

**­***Рис. 4. Ключові відмінності*

**Висновок:** Від кількості параметрів залежить точність отриманої відповіді та врахування необумовлених нюансів.