

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

ЗВІТ

до лабораторної роботи № 1

по дисципліні «Програмування інтелектуальних інформаційних систем»

Тема: «Розроблення Web – застосунку з елементами штучного інтелекту»

Виконав:

студент групи ЗПІ-зп01

Лебідь Олександр

Захищено з балом _____

Перевірив:

доцент, к.т.н.

Катін Павло Юрійович

Київ 2022

Мета роботи: набуття знань, умінь та навичок програмування інтелектуальних інформаційних систем.

Завдання

Розробити інтелектуальну інформаційну систему для опрацювання файлів, що містять дані JPEG. Створена програма має повідомляти користувача про найменування об'єкта, якій міститься на зображенні.

Виконання лабораторної роботи

Для виконання даної лабораторної роботи мною шляхом використання інтегрованого середовища розробки PyCharm було створено новий проект Lab_Art_Intelligence (рис. 1)

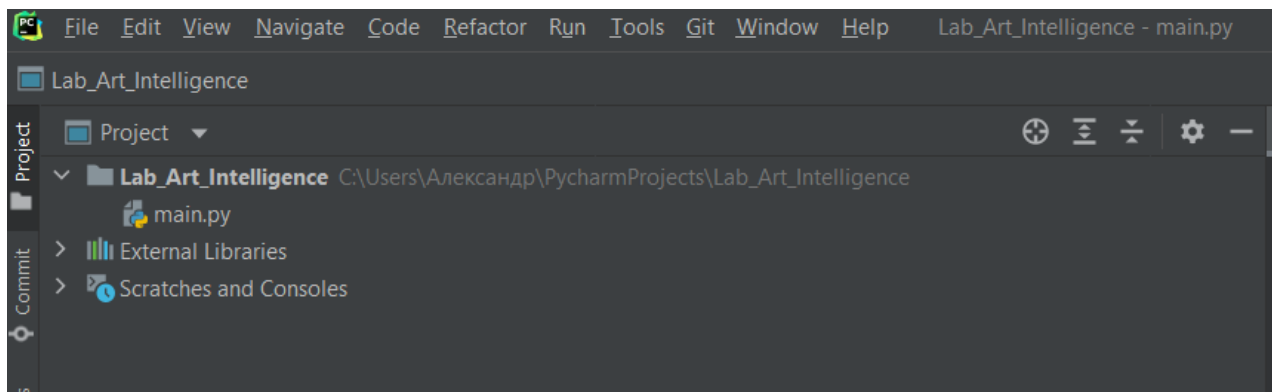


Рис. 1

Для можливості подальшого виконання даного проекту було завантажено через термінал PyCharm бібліотеки TensorFlow та Streamlit.

Завантаження здійснено шляхом виконання команд:

```
pip install tensorflow
```

```
pip install streamlit
```

TensorFlow - відкрита програмна бібліотека для машинного навчання, розроблена Google для вирішення завдань побудови та тренування нейронної

мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття.

Бібліотека спочатку розроблена для Python і найчастіше використовується з ним. Проте існують реалізації TensorFlow для інших мов: C#, C++ , Go , Java, Swift і так далі. Вони використовуються рідше за основну — головним чином для написання коду під специфічні платформи. Сама бібліотека написана мовою Python із використанням швидкого та продуктивного C++ для вирішення математичних завдань. Тому вона ефективно працює зі складними обчисленнями. Сама бібліотека включає безліч інструментів для різних напрямків ML, але найчастіше використовується для роботи з нейронними мережами.

TensorFlow має високий рівень абстракції, тобто бібліотека написана так, що не потрібно думати про технічну реалізацію абстрактних понять. Можна зосередитись на описі логіки програми та на математиці, а спосіб реалізації обчислень – завдання TensorFlow, а не програміста. Це полегшує розробку та дозволяє сконцентруватися на важливих завданнях.

Приклади використання технології – розпізнавання природної мови, зображень та рукописних текстів, різноманітні завдання класифікації чи кластеризації, обробка великих даних.

У нашій лабораторній роботі TensorFlow використовується для роботи із нейронною мережею EfficientNet-b0.

EfficientNet-b0 - згортова нейронна мережа, яка навчена більш ніж на мільйоні зображень від бази даних ImageNet. Мережа може класифікувати зображення до 1000 категорій об'єктів, таких як клавіатура, миша, олівець, різні тварини. Мережа має вхідний розмір зображень 224 x 224.

Нейронні мережі — математичні моделі та їхнє програмне втілення, засновані на будові людської нервової системи.

Зокрема, нервова система живої істоти складається з нейронів – клітин, які накопичують та передають інформацію у вигляді електричних та хімічних імпульсів. У нейронів є аксон - основна частина клітини, і дендрит - довгий

відросток на її кінці, який може досягати сантиметра в довжину. Дендрити передають інформацію з однієї клітини на іншу і працюють як «дроти» для нервових імпульсів.

Як приклад можна навести будь-яку усвідомлену дію. Наприклад, людина вирішує підняти руку: імпульс спочатку з'являється у його мозку, потім через мережу нейронів інформація передається від однієї клітини до іншої. По дорозі вона перетворюється і зрештою досягає клітин у руці. Рука піднімається. Так працює більшість процесів в організмі — тих, що керуються мозком.

Але головна особливість нейронних мереж – здатність навчатися. І саме вона лягла в основу машинних нейромереж.

Структуру нейрона відтворили за допомогою коду. Як «аксон» використовується комірka, яка зберігає в собі обмежений діапазон значень. Інформація про «нервові імпульси» зберігається у вигляді математичних формул і чисел.

Зв'язки між нейронами також реалізовані програмно. Один із них передає іншому на вхід якусь обчислену інформацію, той отримує її, обробляє, а потім передає результат уже своїх обчислень далі. Таким чином, інформація поширюється по мережі, коефіцієнти всередині нейронів змінюються - відбувається навчання.

Коли нейромережу навчають, їй «показують» дані, за якими необхідно щось передбачити, і еталонні правильні відповіді для них. Це називається навчальною вибіркою. Інформації має бути багато — вважається, що мінімум удесятеро більше, ніж кількість нейронів у мережі.

Під час навчання нейромережі показують будь-яку інформацію й кажуть, що це таке, тобто дають відповідь. Усі дані надаються не за допомогою слів, а за допомогою формул та числових коефіцієнтів. Наприклад, зображення жінки відповідає "1", а зображення чоловіка - "0". Це найпростіший приклад; реальні мережі влаштовані складніше.

Так, вхідні нейрони отримують інформацію, перетворюють її та передають далі. Зміст інформації автоматично обробляється з допомогою формул і перетворюється на математичні коефіцієнти. Приблизно як те, що ми бачимо, перетворюється на нервові імпульси і передається в мозок. Він їх обробляє, і людина розуміє, що довкола нього. Тут принцип схожий.

Кожен нейрон має «вагу» — число всередині нього, розраховане за особливими алгоритмами. Він показує, наскільки показання нейрона значимі для всієї мережі. Відповідно, під час навчання ваги нейронів автоматично змінюються та балансуються.

В результаті складається ситуація, коли певні нейрони реагують, наприклад, на силует людини — і видають інформацію, яка перетворюється на відповідь: «Це людина». При цьому людину не потрібно описувати як набір математичних постатей — під час навчання нейронна мережа сама задає значення ваги, які визначають її.

Виведенням нейронної мережі стає набір формул і чисел, які перетворюються на відповідь. Наприклад, якщо зображення чоловіка - "0", а жінки - "1", то результат 0,67 означатиме щось на кшталт "Швидше за все, це жінка". Нейросітка через свою структуру не може дати абсолютно точної відповіді — лише ймовірність. І через закритість та нестабільність нейронів її показання можуть відрізнитися навіть для однакових вибірок.

Ми не можемо сказати, за якими критеріями програма «вирішує», що на зображенні зображена людина або що текст є віршем. Усе це відбувається автоматично. Завдання розробника - правильно описати структуру та задати формули. Приблизно так само ми не можемо достовірно сказати, що саме відбувається в людському мозку, чому він розуміє, що собака це собака, навіть якщо вперше бачить незнайому породу.

Доцільно зазначити, що кожен нейрон ніяк не пов'язаний із процесом роботи інших. Так, вони отримують один від одного інформацію, але їхня внутрішня діяльність не залежить від інших елементів. Тому навіть якщо один нейрон вийде з ладу, інший продовжить працювати — це важливо у питанні стійкості до відмови. Подібна стійкість властива і біологічним нейронним

мережам, які продовжують працювати, навіть якщо вони пошкоджені. Але в незалежності є і недолік: через неї рішення виявляються багатоступінчастими і часом хаотичними, їх складно передбачити та вплинути на них.

Так як нейрони самі підбирають критерії і не залежать один від одного, нейромережі гнучкіші, ніж інші моделі машинного навчання. Їхня архітектура успадкувала важливі властивості біологічної нервової системи: здатність самонавчатися і пристосовуватися до нових даних, можливість ігнорувати «шуми» та неважливі деталі вхідної інформації. Як жива людина зможе розрізнити знайомого в натовпі, так нейромережа можна навчити виділяти потрібне та відкидати непотрібне. Як результат, нейромережі здатні вирішувати широкий спектр завдань, і їх можна адаптувати практично за будь-яких обставин.

У нейромереж є спільні риси - наприклад, наявність вхідного шару, який приймає інформацію на вхід. Але багато й відмінностей. Для кожного завдання потрібна своя нейронна мережа. Вони відрізнятимуться структури, архітектура, типи нейронів та багато іншого. Створити універсальний алгоритм неможливо, принаймні поки що, тому мережі окремо оптимізують під певні спектри завдань.

Наприклад, серед інших виділяють такі нейромережі, як:

- *односпрямовані* - працюють в одному напрямку. Це означає, що вони не мають «пам'яті», а потік інформації передається тільки в один бік. Односпрямовані мережі добре підходять для розпізнавання завдань. Суть приблизно та сама, що й у випадку зі сприйняттям навколишнього світу реальним мозком. Органи почуттів отримують інформацію та передають її в одному напрямку, та в процесі трансформується та розпізнається. Мозок робить висновок: "я бачу собаку", "чути рок-музику", "на вулиці холодно". Односпрямована модель працює за тим самим принципом, але спрощено. Ще один варіант їх застосування – прогнозування.

- *рекурентні* - ці мережі мають ефект «пам'яті» завдяки тому, що дані передаються у двох напрямках, а не в одному. У результаті вони сприймають попередню отриману інформацію та можуть глибше її «аналізувати». Це

корисно, якщо перед мережею стоїть складне завдання на кшталт перекладу тексту.

- *згорткові* - це окрема категорія нейронних мереж менш закрита, ніж інші, завдяки принциповій багатошаровості. Багатошаровими називаються нейронні мережі, в яких нейрони згруповані у шари. При цьому кожен нейрон попереднього шару пов'язаний з усіма нейронами наступного шару, а всередині шарів зв'язку між нейронами відсутні. Згорткові мережі використовують для розпізнавання образів. У них особлива структура шарів: частина займається «згортанням», перетворенням картинки, а частина — угрупованням та розпізнаванням маленьких дискретних елементів, створених на шарах згортання. Таких шарів кілька. Результат — більш висока точність та якісне сприйняття інформації.

Цікавий факт: якщо звичайні нейромережі були засновані на нейронах у головному мозку, так згорткові - на структурі зорової кори. Це та частина мозку, яка відповідає за сприйняття картинок. У ній чергуються «прості» та «складні» клітини: перші реагують на певні лінії та контури, другі — на активацію конкретних простих клітин. Так відбувається процес розпізнавання образів у мозку, і приблизно так само влаштована сітка.

Як вже зазначалося вище, у цій лабораторній роботі використовується саме попередньо навчана згорткова нейронна мережа - **EfficientNet-b0**.

Streamlit - це фреймворк Python з відкритим вихідним кодом, який використовується для розгортання моделей машинного навчання в красивих веб-застосунках всього у кілька рядків коду.

Даний фреймворк був вибраний у межах цієї лабораторної роботи, оскільки він досить зручний та легкий у використанні. Так, якщо внести будь-які зміни до свого коду, то можна одразу побачити автоматичні оновлення у веб-додатку.

Далі мною було написано код програми із поясненнями, який має наступний вигляд:

```

import io
import streamlit as st
from PIL import Image
import numpy as np
from tensorflow.keras.applications import EfficientNetB0 # підключаємо модель
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.efficientnet import preprocess_input,
decode_predictions

@st.cache(allow_output_mutation=True)
# завантажуюмо попередньо навчану модель EfficientNetB0
def load_model():
    return EfficientNetB0(weights='imagenet')

# попередня обробка зображення
def preprocess_image(img):
    img = img.resize((224, 224)) # перетворюємо зображ. у визнач. розмір

    # стандартні операції для підготовки зображ. до опрацювання в tensorflow
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    return x

# завантажник файлів streamlit
def load_image():
    uploaded_file = st.file_uploader(label='Виберіть зображення для опрацювання')
    if uploaded_file is not None: # якщо користувач завант. файл
        image_data = uploaded_file.getvalue() # зчитуємо файл. Отримуємо дані в байтах
        st.image(image_data) # перетворюємо в зображення за доп. зас. streamlit
        return Image.open(io.BytesIO(image_data)) # створюємо зображ. у форматі PIL та повертаємо
    else:
        return None

# друкуємо ТОП-3 класа з самою великою ймовірністю
def print_predictions(preds):
    classes = decode_predictions(preds, top=3)[0] # декодуємо прогноз моделі
    for cl in classes:
        st.write(cl[1], cl[2]) # друкування назви класу та ймовірності

# код для побудови веб сторінки
model = load_model()

st.title('Класифікація зображень') # виводимо заголовок
img = load_image() # виводимо картинку на сторінці
result = st.button('Опрацювати зображення') # кнопка розпізнан. зображення

if result:
    x = preprocess_image(img) # викликаємо метод для попередньої обробки зображ.
    preds = model.predict(x) # запускаємо розпізнання даних, отриманих з попер. опрацювання зображ.
    st.write('**Результати опрацювання**') # відображаємо зазначений текст
    print_predictions(preds) # друкуємо отриманий результат

```


Суть програми полягає у тому, щоб користувач мав змогу у Web – застосунку через кнопку обрати будь – яке заздалегідь збережене фото із розширенням .jpg, завантажити його у додатку та у подальшому отримати результат у вигляді текстового повідомлення, який об’єкт зображений на фото. При цьому користувач отримує три найбільш вірогідних варіанти у порядку спадання ймовірності.

Опрацювання зображення та отримання результату відбувається за допомогою бібліотеки для машинного навчання Tensorflow та попередньо навченої нейронної мережі EfficientNet-b0. Представлення моделі машинного навчання у Web – застосунку здійснюється за допомогою використання фреймворку Streamlit.

Для того, щоб запустити Web – застосунок, у терміналі необхідно виконати команду:

```
streamlit run main.py
```

де main.py – це файл, який ми вказуємо Streamlit відкрити.

У результаті за посиланням Local URL: <http://localhost:8501> завантажується браузер із Web – застосунком, створеним за допомогою Streamlit (рис. 2)

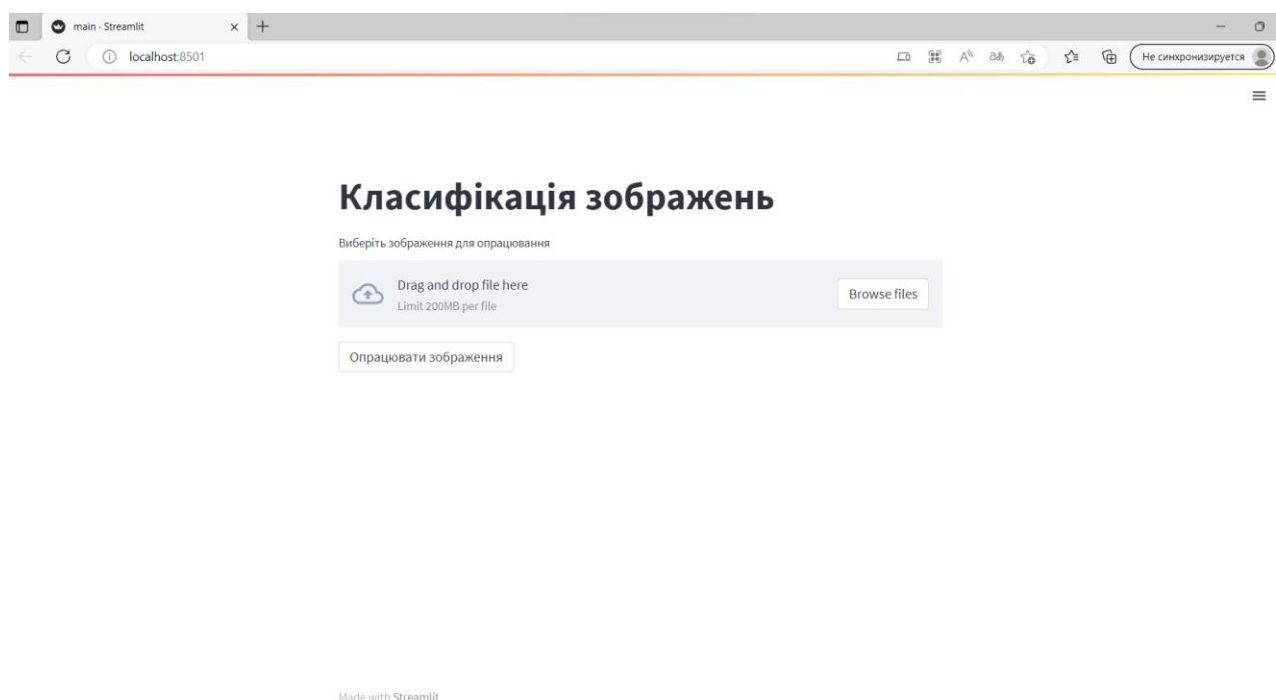



Рис. 2


Далі попередньо збережений файл із розширенням .jpg завантажуюємо у Web - застосунку (рис. 3).


Класифікація зображень

Виберіть зображення для опрацювання

 Drag and drop file here
Limit 200MB per file

Browse files


 3.jpg 5.0KB ×




Опрацювати зображення


Рис. 3

Після завантаження файлу натискаємо кнопку із написом “Опрацювати зображення” та отримуємо результат опрацювання штучним інтелектом даного файлу (Рис. 4)

 Drag and drop file here
Limit 200MB per file

Browse files

 3.jpg 5.0KB ×



Опрацювати зображення

Результати опрацювання:

Labrador_retriever 0.76496524

Rhodesian_ridgeback 0.02008495

golden_retriever 0.011016229

Рис. 4

Як бачимо, після опрацювання маємо результат, що із ймовірністю 76% на фото зображено пес породи Лабрадор - ретривер. Тобто програма відпрацювала коректно та отримано результат із досить високою точністю.

Висновок

Під час виконання даної лабораторної роботи я набув знань, умінь та навичок з технології розроблення інтелектуальної інформаційної системи. Ознайомився із бібліотекою для машинного навчання Tensorflow, бібліотекою для розгортання моделей машинного навчання у Web – застосунку Streamlet, навчився використовувати згорткову нейронну мережу EfficientNet-b0.

Створений проект розташовано у репозиторій на GitHub за посиланням https://github.com/lebid-ol/Lab_Artificial-intelligence.git