МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)
Кафедра дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

ДИПЛОМНА РОБОТА

РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЯ ПЛАЗУНІВ
ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

Виконала: студентка групи ПМІ-45
спеціальності 122 – комп'ютерні науки
(шифр і назва спеціальності)
Пац А.Б.
(підпис) (прізвище та ініціали)
Керівник Колос Н.М.
(підпис) (прізвище та ініціали)
Рецензент
(підпис) (прізвище та ініціали)

2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики
Кафедра Дискретного аналізу та інтелектуальних систем
Спеціальність 122 – Комп'ютерні науки
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____

"31" серпня 2022 року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТА

Пац Анастасії Богданівни
(прізвище, ім'я, по батькові)

1 .Тема роботи "Розпізнавання та класифікація рептилій за допомогою нейронних мереж"

керівник роботи Колос Надія Мирославівна, кандидат фізико-математичних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені Вченою радою факультету від "13" вересня 2022 року № 15

2. Строк подання студентом роботи 13.06.2023 р.

3. Вихідні дані до роботи: мова програмування Python, середовища розробки Google Colab та Jupyter notebook, бібліотеки Numpy, Matplotlib, Pandas, OpenCV, графічний інструмент анотації зображень LabelImg, нейронна мережа YOLOv5s.

4. Зміст дипломної роботи (перелік питань, які потрібно розробити): збір даних у вигляді зображень, обробка даних, тренування моделі нейронної мережі, створення віконного застосунку для тестування моделі.

4. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): зображення моделі нейронної мережі, результати навчання нейронної мережі, результати тестування натренованої моделі в застосунку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання **31 серпня 2022 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми	20.09.22	
2	Складання плану роботи	19.10.22	
3	Пошук можливих варіантів виконання завдання	10.11.22	
4	Збір даних у вигляді зображень	14.01.23	
5	Первинна обробка даних	29.01.23	
6	Анотація зображень за допомогою LabelImg	15.02.23	
7	Вторинна обробка анотованих даних	09.03.23	
8	Пошук відповідної моделі для тренування даних	21.03.23	
9	Тренування даних та вибір найкращої моделі	30.03.23	
10	Написання вступу та першого розділів дипломної роботи	09.04.23	
11	Написання другого розділу дипломної роботи	13.04.23	
12	Створення віконного застосунку для тестування моделі	16.04.23	
13	Написання третього та четвертого розділів дипломної роботи	22.04.23	
14	Завершення написання дипломної роботи (висновки та література)	02.05.23	
15	Подача науковому керівнику	12.05.23	
16	Коригування дипломної роботи	16.05.23	
17	Оформлення кінцевого варіанту роботи	25.05.23	
18	Подання роботи до захисту	12.06.23	

Студент _____ Пац А.Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Колос Н.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пац А. Б. “Розпізнавання та класифікація плазунів за допомогою нейронних мереж”. Керівник роботи — кандидат фіз.-мат. наук, доцент кафедри дискретного аналізу та інтелектуальних систем Колос Н. М. Дипломна робота бакалавра: 35 сторінок, 23 рисунки, 1 таблиця.

Ключові слова: нейромережі, штучний інтелект, розпізнавання об’єктів, рептилії, плазуни, python, YOLO.

Метою є тренування моделі штучної нейронної мережі, яка б з високою точністю розпізнавала та класифікувала різні види рептилій на зображеннях. Головним об’єктом дослідження є використання штучного інтелекту, а саме нейронних мереж для розпізнавання рептилій на зображеннях.

ЗМІСТ

ВСТУП

Одним із основних способів розуміння та сприйняття світу є зір. Зображення є величезною частиною нашого життя і відіграє чи не найактивнішу роль у розвитку людської свідомості. Важливість сприйняття візуальної інформації як для окремо взятого індивіда, так і для всієї цивілізації, є доволі великою.

Від початку народження цивілізації, людина старалася систематизувати та видобувати певні знання із візуальної інформації, яку вона отримала із навколишнього середовища: наскельні малюнки, пізніше ілюстровані манускрипти, ще пізніше цілі енциклопедії з детальними зображеннями флори і фауни, а із появою камери процес отримання візуальних даних спростився в рази і надалі кількість цих даних експоненційно росла.

Ефективним методом спостереження за об’єктами навколишнього середовища стали розробка та впровадження у виробничі процеси штучного інтелекту, зокрема штучних нейронних мереж, що, відтворюючи роботу людського мозку, мають властивість навчатися і розпізнавати певні закономірності та малюнки, навіть ті, які не піддаються опрацюванню людиною. Основними перевагами штучних нейронних мереж над людським мозком є здатність за короткий час засвоювати набагато більшу кількість інформації.

Завдяки застосуванню широкого спектру програм, які базуються на використанні штучного інтелекту, процес машинного розпізнавання об’єктів на зображеннях створив умови для більш інтенсивного розвитку у найрізноманітніших сферах, як от медицина, автомобільна промисловість, ігри, соціальна комунікація. Також особливо актуальною у XXI столітті стала природоохоронна галузь, для якої використання штучного інтелекту — це вагома

перевага, адже чим раніше буде виявлена проблема, тим швидше науковці зможуть взятись за її вирішення, і час, який економить нейромережа, буде доданий до часу здорового функціонування нашої планети.

Актуальність роботи

Актуальність теми роботи полягає у тому, що автоматичне виявлення рептилій на зображеннях за допомогою штучного інтелекту може бути корисним для багатьох сфер цієї ж таки природоохоронної галузі. Ось кілька прикладів:

Збереження фауни: багато видів рептилій знаходяться під загрозою зникнення через втрату первісного середовища існування, зміну клімату та інші негативні фактори. Використовуючи штучний інтелект для виявлення та моніторингу рептилій у їхньому природному середовищі існування, природоохоронці можуть краще зрозуміти особливості природних потреб цих тварин, прослідкувати зміну їхньої поведінки, еволюцію виду при мутації навколишнього середовища та розробити ефективніші стратегії їхнього збереження.

Сільське господарство: деякі рептилії, наприклад змії, можуть завдавати шкоди посівам і худобі. Використовуючи штучний інтелект для виявлення та відстеження рептилій у сільськогосподарських угіддях, фермери зможуть сформувати знання про їхню поведінку, виокремити типових представників “шкідників” та розробити систему заходів, щоб запобігти збитку і захистити свою власність.

Екологія: вивчення поведінки рептилій може дати розуміння низки наукових питань зі сфери екології, наприклад наслідків зміни клімату. Використовуючи штучний інтелект для виявлення та відстеження рептилій у польових умовах, дослідники можуть збирати дані, які було б важко або неможливо отримати будь-якими іншими способами, і завдяки ним прослідкувати, як

впливають зміни клімату на представників цього виду, чим загрожують ці зміни в майбутньому рептиліям та іншим представникам тваринного світу, які закономірності цих змін можна прослідкувати у довготривалій перспективі.

Герпетологія: герпетологи вивчають рептилій і земноводних, а виявлення рептилій за допомогою штучного інтелекту може допомогти їм ідентифікувати різних представників цього виду, вивчити їх поведінку та дізнатися більше про їхнє життя у різних умовах навколишнього середовища. Це може допомогти покращити наше розуміння цих тварин та їхньої ролі серед інших тварин у фауні.

Загалом, виявлення рептилій за допомогою штучного інтелекту має потенціал для вирішення низки важливих питань екології, сільського господарства та наукових досліджень.

Мета

Метою цієї роботи є тренування моделі штучної нейронної мережі, яка б з високою точністю розпізнавала та класифікувала різні види рептилій на зображеннях.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПРИНЦИП РОБОТИ НЕЙРОННИХ МЕРЕЖ

1.1 Штучні нейронні мережі

Штучна нейронна мережа (ШНМ) — обчислювальна система, побудована за принципом функціонування біологічних нейронних мереж живих організмів (людей та тварин).

Біологічні нейрони ієрархічно з'єднані в мережі, де вихід одних нейронів є входом для інших нейронів. Ми можемо представити такі мережі у вигляді з'єднаних шарів з вузлами.

Кожен вузол приймає зважений вхід, після обчислення зваженої суми він передає її до функції активації, та генерує вихід.

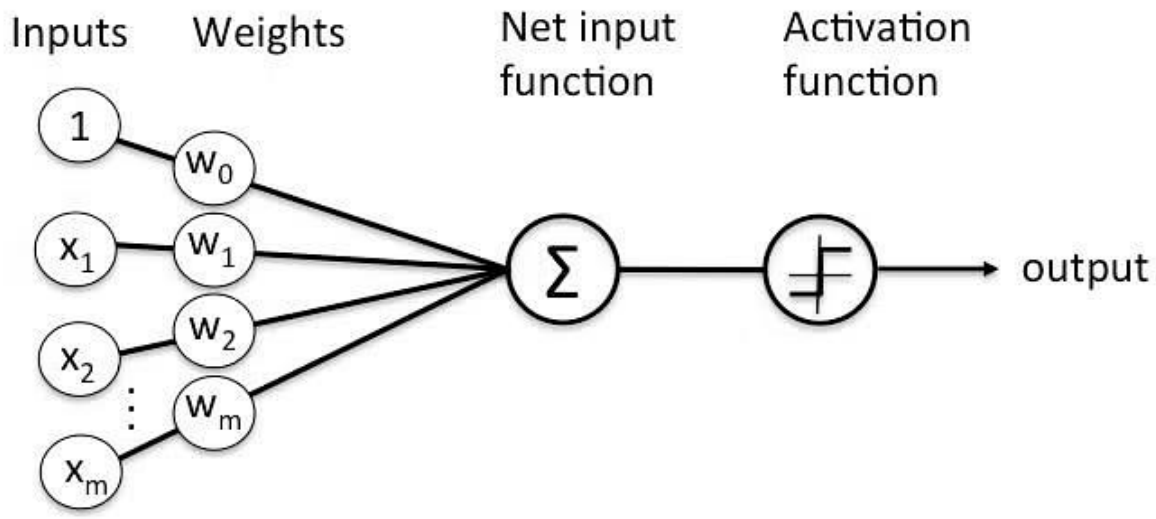

Рисунок 1.1 Штучний нейрон

Зважений вхід у вузол має вигляд:

$w_0+w_1 x_1+w_2 x_2+...+w_m x_m$,

де w_i — числові значення ваги. Вони нам потрібні, оскільки це значення, які будуть змінюватись протягом процесу навчання, а w_0 потрібна для забезпечення правильного навчання нейронної мережі, оптимізуючи нелінійну функцію активації.

У задачах активаційна функція повинна мати характеристику "тригера". Іншими словами, якщо вхід більше, ніж деяке значення, то вихід повинен змінювати стан, наприклад з 0 на 1 або -1 на 1. Це імітує "включення" біологічного нейрону.

Штучна нейронна мережа — це, власне, система з’єднаних і взаємодіючих між собою штучних нейронів, що організовуються в прошарки. Кожен такий вузол має справу не лише з сигналами, які він періодично отримує, але й сигналами, які він посилає іншим штучним нейронам.

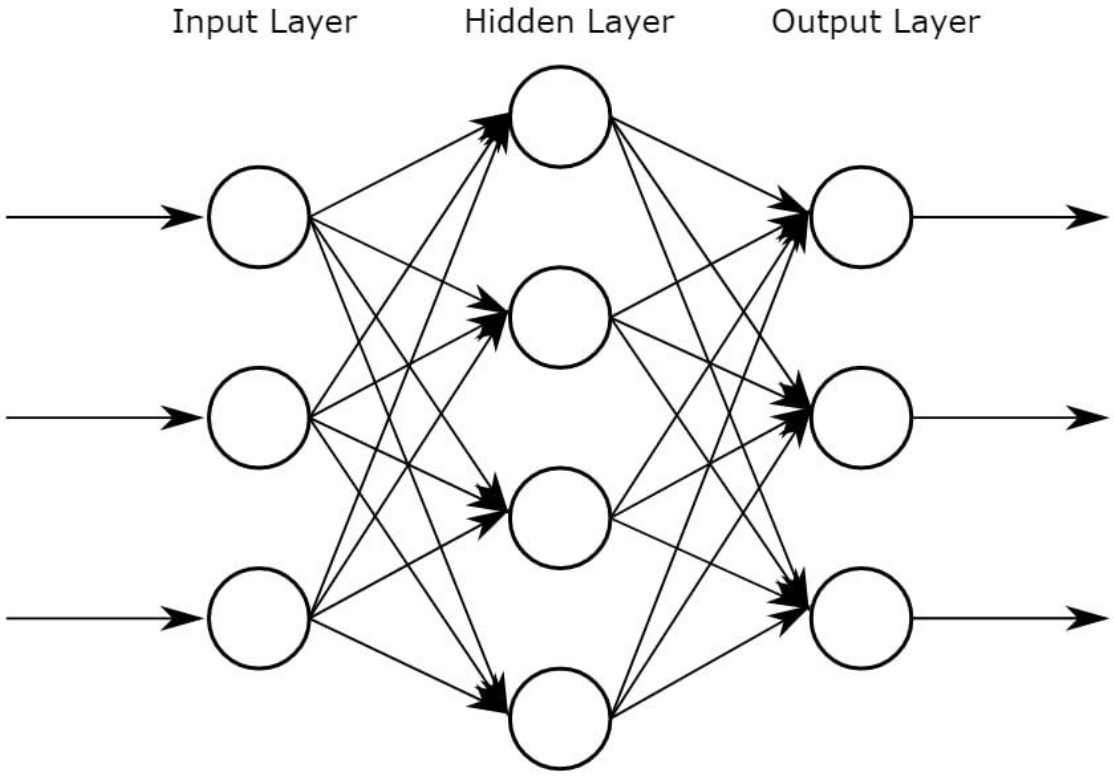

Рисунок 1.2 Двошарова штучна нейронна мережа

Вхідні вузли обробляють дані, аналізують або класифікують їх і передають наступному прошарку. Приховані шари отримують дані від вхідного шару або інших прихованих шарів, що так само аналізують вихідні дані попереднього шару, обробляють його та передають далі.

Вихідний рівень дає кінцевий результат обробки всіх даних штучною нейронною мережею. Він може мати один або кілька вузлів. Наприклад, якщо у нас є бінарна проблема класифікації, вихідний рівень матиме один вихідний вузол, який дасть результат як 1 або 0. Однак, якщо у нас є проблема багатокласової класифікації, вихідний рівень може складатися з більш ніж одного вихідного вузла.

Задача класифікації — задача, що містить множину об’єктів, поділених певним чином на класи. Алгоритми класифікації дозволяють розділити об'єкти відповідно до зазначених заздалегідь класів, наприклад розділити кішок і собак, чи багатокласова задача — класифікувати 33 літери українського алфавіту з рукописного тексту.

1.2 Згорткові нейронні мережі

Згорткові нейронні мережі (Convolutional neural network, CNN) — це мережева архітектура для алгоритмів глибокого навчання, яка спеціально використовується для розпізнавання зображень і завдань, які включають обробку піксельних даних. Такий тип нейронної мережі може розкривати ключову інформацію як у часових рядах, так і в зображеннях. З цієї причини він дуже цінний для завдань, пов’язаних із зображеннями, наприклад класифікація об’єктів і розпізнавання образів. Щоб ідентифікувати шаблони в зображенні, CNN використовує принципи лінійної алгебри. Згорткові нейронні мережі також можуть класифікувати аудіо та сигнальні дані.

Згорткові нейронні мережі глибокого навчання складаються з трьох рівнів: згорткового шару (Convolutional layer), шару об’єднання (Pooling layer) та повністю

зв’язного шарів (Fully connected, FC). Згортковий шар є першим шаром, тоді як шар FC є останнім.

Більшість обчислень відбувається на згортковому рівні, який є основним будівельним блоком CNN. Другий згортковий шар може слідувати за початковим згортковим шаром. Процес згортання включає в себе ядро або фільтр всередині цього шару, який переміщається по рецептивним полям зображення, перевіряючи, чи присутня певна ознака в зображенні. За кілька ітерацій ядро охоплює все зображення. Після кожної ітерації обчислюється скалярний добуток між вхідними пікселями та фільтром. Остаточний результат серії точок називається картою об’єктів (feature map). Зрештою, зображення перетворюється на числові значення на цьому шарі, що дозволяє CNN інтерпретувати зображення та витягувати з нього відповідні шаблони.

Під час прямого проходу ядро ковзає по висоті та ширині зображення, створюючи представлення зображення цієї сприятливої області. Це створює двовимірне представлення зображення, відоме як мапа активації, яка дає відповідь ядра на кожну просторову позицію зображення. Розмір ковзання ядра називається кроком.

Якщо ми маємо вхід розміром W x W x D і D-out кількістю ядер із просторовим розміром F із кроком S і розміром заповнення P, тоді розмір вихідного об’єму можна визначити за такою формулою:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Це дасть вихідний обсяг розміром W-out x W-out x D-out:

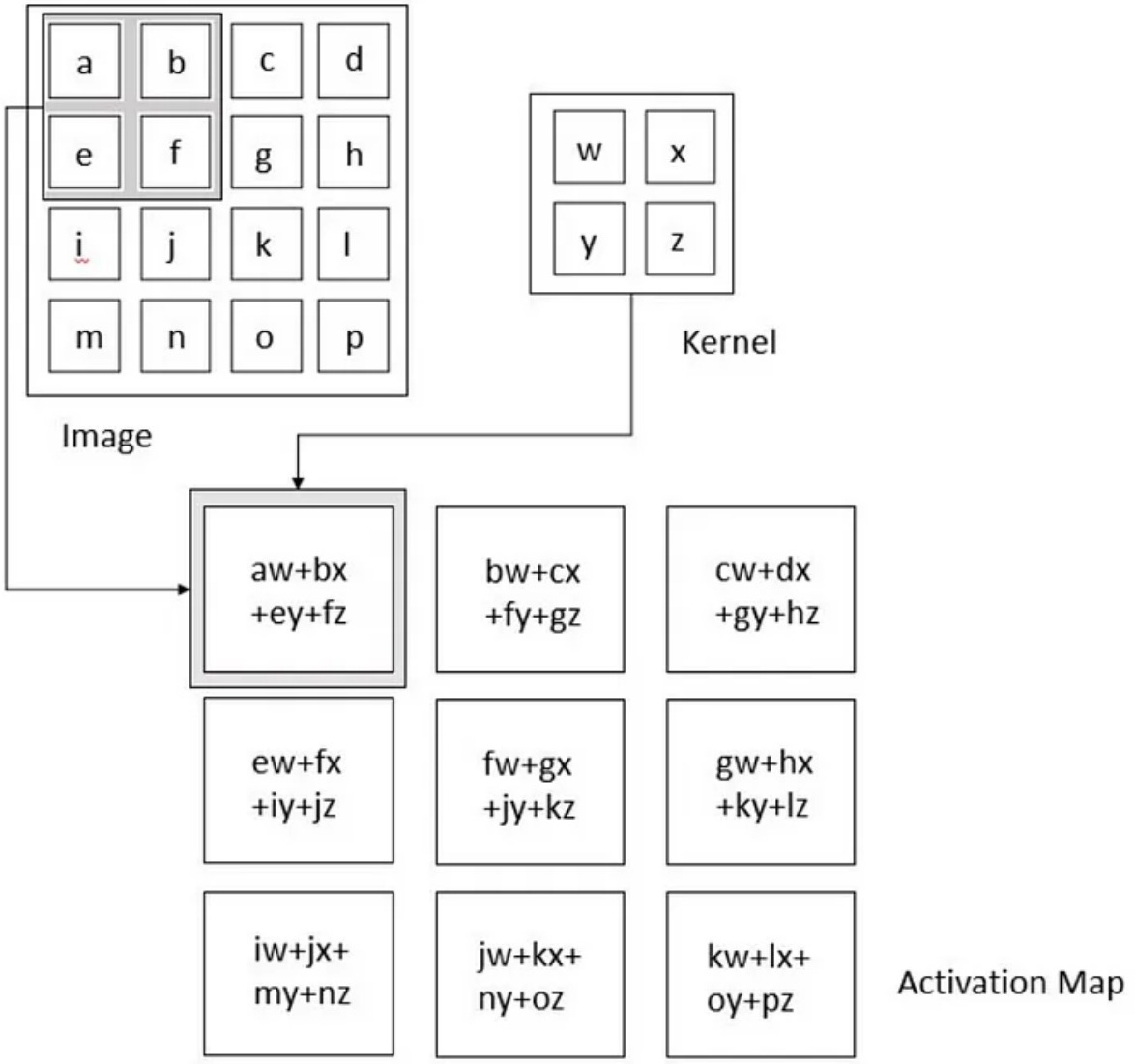

Рисунок 1.3 Мапа активації

Від згорткового до повністю об’єднаного шару складність CNN зростає. Саме ця зростаюча складність дозволяє CNN послідовно ідентифікувати більші частини та складніші характеристики зображення, поки він нарешті не ідентифікує об’єкт повністю.

Подібно до згорткового рівня, шар об’єднання також розгортає ядро або фільтр по вхідному зображенню. Але на відміну від згорткового рівня, шар об’єднання зменшує кількість параметрів у вхідних даних і також призводить до

втрати деякої інформації. Позитивним є те, що цей рівень зменшує складність і підвищує ефективність CNN.

Існує кілька функцій об’єднання, наприклад середнє значення прямокутного околу, норма прямокутного околу та зважене середнє на основі відстані від центрального пікселя. Однак найпопулярнішим процесом є максимальне об’єднання (max pooling), яке повідомляє про максимальний результат із околу.

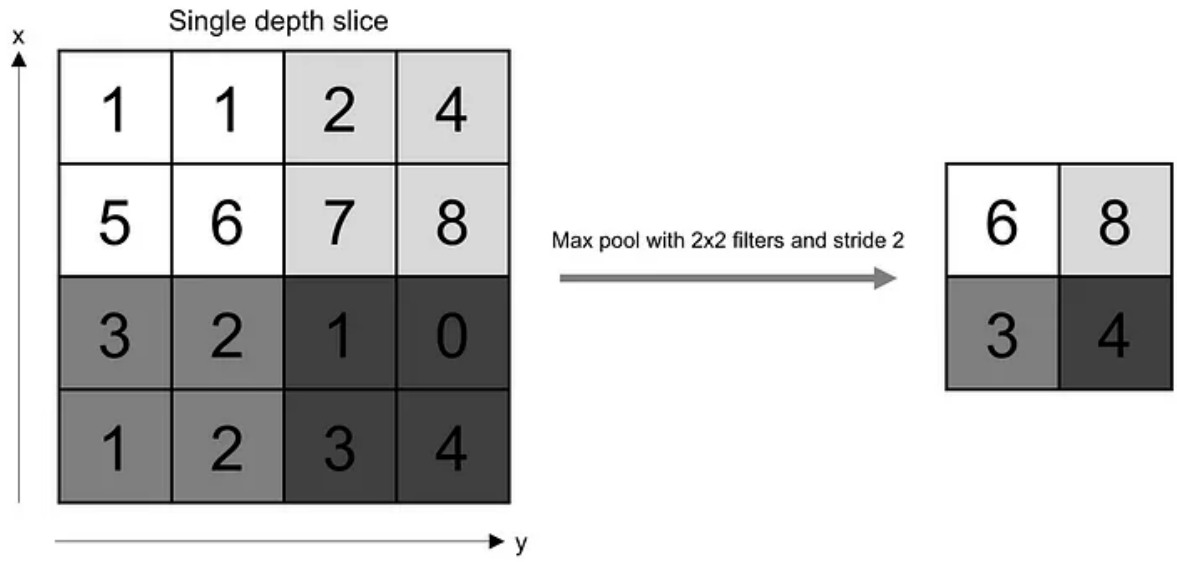

Рисунок 1.4 Операція максимального об’єднання

Якщо у нас є карта активації розміром W x W x D, об’єднуюче ядро просторового розміру F і крок S, тоді розмір вихідного обсягу можна визначити за такою формулою:

$$W_{out} = \frac{W - F}{S} + 1$$

Це дасть вихідний обсяг розміром W-out x W-out x D.

Повністю зв'язаний шар – це місце, де відбувається класифікація зображень на основі ознак, витягнутих із попередніх шарів. Тут повне зв'язаність означає, що всі входи або вузли з одного шару підключені до кожного блоку активації або вузла наступного шару.

Усі рівні в згорткових нейронних мережах зв’язані не повністю, тому що це призведе до надмірно щільної мережі. Це також призвело б до збільшення втрат і вплинуло б на якість виведення.

Згорткова нейронна мережа може мати кілька рівнів, кожен з яких навчається виявляти різні характеристики вхідного зображення. Фільтр або ядро застосовуються до кожного зображення, щоб створити результат, який стає все кращим і деталізованим після кожного шару. На нижніх рівнях фільтри можуть починатися як прості функції.

Для програм розпізнавання зображень, класифікації зображень і комп’ютерного зору (CV) CNN особливо корисні, оскільки вони забезпечують високоточні результати, особливо коли задіяно багато даних. CNN також вивчає особливості об’єкта в послідовних ітераціях, коли дані об’єкта переміщуються через численні шари нейромережі. Це пряме і глибоке навчання усуває потребу в ручному видаленні функцій.

CNN можна перенавчити для нових завдань розпізнавання та побудувати на основі вже існуючих мереж. Ці переваги відкривають нові можливості використання згорткових нейронних мереж для реальних додатків без збільшення обчислювальної складності чи витрат.

2 ВИКОРИСТАНІ ТЕХНОЛОГІЇ

2.1 Python

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Його високорівневі вбудовані структури даних у поєднанні з динамічною типізацією та динамічним зв’язуванням роблять його дуже привабливим для швидкої розробки додатків, а також для використання як мови сценаріїв або з’єднувальної мови для з’єднання існуючих компонентів. Python підтримує різноманітні бібліотеки, модулі та пакети.

2.2 Jupyter notebook

Jupyter Notebook надає середовище розробки на основі браузера, який зручний для співпраці, обміну та навіть публікації результатів наукових досліджень. Jupyter Notebook насправді є окремим випадком ширшої структури ноутбуків Jupyter, яка охоплює ноутбуки для Julia, R та інших мов програмування.

2.3 Google Colab

Colaboratory, або скорочено «Colab», є продуктом Google Research. Colab дозволяє будь-кому писати та виконувати довільний код Python через браузер і особливо добре підходить для машинного навчання та аналізу даних. Технічно кажучи, Colab — це розміщена служба для ноутбуків Jupyter, яка не потребує налаштування, але забезпечує безкоштовний доступ до обчислювальних ресурсів, включаючи графічні процесори. Також Colab дозволяє використовувати блокноти Jupyter і ділитися ними з іншими без необхідності завантажувати, інсталювати чи запускати будь-що.

2.4 LabelImg

LabelImg — це графічний інструмент анотації зображень, який дозволяє малювати візуальні рамки навколо об’єктів на кожному зображенні, а також автоматично зберігає XML-файли зображень з мітками.

2.5 NumPy

NumPy — це розширення мови Python, формати даних якого включають матриці та багатовимірні масиви. NumPy може виконувати математичні операції з такими масивами, як статистичні, алгебраїчні та тригонометричні процедури. Він забезпечує високофункціональний багатовимірний масив і навіть необхідні інструменти для обчислення та регулювання масивів.

2.6 Matplotlib

Matplotlib — бібліотека для мови Python для 2D-графіків і зображень, який переважно використовується для візуалізації наукових, інженерних і фінансових даних. Програмне забезпечення можна безкоштовно завантажити, використовувати та розповсюджувати. Вона має широкий спектр застосування. Більшість користувачів знайомі з інтерфейсом командного рядка для інтерактивного створення графіків і зображень. Цей інтерфейс відображає дані та обробляє їх у простому спливаючому вікні.

2.7 Pandas

Pandas — це програмна бібліотека, написана для мови програмування Python для зручної обробки даних та їхнього аналізу. Вона, зокрема, пропонує структури даних та операції для роботи з числовими таблицями та часовими рядами.

2.8 OpenCV

Комп'ютерний зір — це сфера яка в даний момент потужно розвивається, частково як результат випуску доступних і якісних камер, частково через доступність до обчислювальних потужностей, а частково через те, що алгоритми зору все більше і більше вдосконалюються. Сам OpenCV зіграв певну роль у розвитку комп'ютерного зору, даючи можливість тисячам людей робити дослідження в даній сфері. Орієнтуючись на машинний зір в реальному часі, OpenCV допомагає студентам та професіоналам ефективно реалізовувати проекти та виконувати дослідження, надаючи їм комп'ютерний зір та інфраструктуру машинного навчання, які раніше були доступні лише для кількох дослідницьких лабораторій. В бібліотеці OpenCV реалізовано і оптимізовано понад 2000 алгоритмів, що включає в себе як класичні алгоритми машинного зору так і сучасні. Ці алгоритми використовуються для ідентифікації та виявлення облич, розпізнаванні об’єктів, класифікації дій людини на відео, відстеження рухів камери, відстеження рухомих об’єктів, створення 3D моделей об’єктів, об’єднання зображень для отримання високої роздільної здатності зображення цілої сцени(панорамні знімки), виявлення ідентичних зображення в базі даних, видалення ефекту червоних очей із фото, зроблених за допомогою спалаху, відслідковування руху очей, тощо.

2.9 YOLO (You Only Look Once)

YOLO (You Look Only Once) — система виявлення об’єктів у реальному часі, розроблена компанією Ultralytics. YOLOv5 — це найсучасніша модель виявлення об’єктів у реальному часі, яка пропонує покращену продуктивність і точність порівняно з попередніми версіями. Компанія Ultralytics розробила та випустила

YOLOv5 як проект із відкритим вихідним кодом, зробивши його доступним для спільноти комп’ютерного зору та розробників у всьому світі. Проєкт отримав значну популярність завдяки своїй простоті, універсальності та високій продуктивності в задачах виявлення об’єктів. Він був широко прийнятий і використовується в різних дослідницьких проектах, академічних дослідженнях і практичних застосуваннях.

YOLOv5 побудовано на архітектурі глибокої нейронної мережі, яка поєднує згорткові нейронні мережі (CNN) і рівні виділення функцій для виявлення та локалізації об’єктів на зображенні. Він використовує єдину мережу для прямого прогнозування обмежувальних рамок і ймовірностей класів для кількох об’єктів за один прохід вперед.

3 ПІДГОТОВКА ДАНИХ ТА ТРЕНУВАННЯ МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

3.1 Збір даних

Пошук даних, а саме зображень різних видів плазунів, для тренування та тестування моделі, було здійснено за допомогою відкритих джерел, зокрема пошукової системи Google а також дочірньої платформи Google — Kaggle, що дозволяє користувачам знаходити набори даних, які вони хочуть використовувати для побудови моделей штучного інтелекту. Згодом потрібно було вручну відсортувати та помістити зображення в каталоги по категоріях: “chameleon”, “crocodile”, “frog”, “gecko”, “iguana”, “lizard”, “snake”, “turtle”.

3.2 Обробка даних

Кожне зображення окремо було оброблене в графічному інструменті для анотації зображень LabelImg.

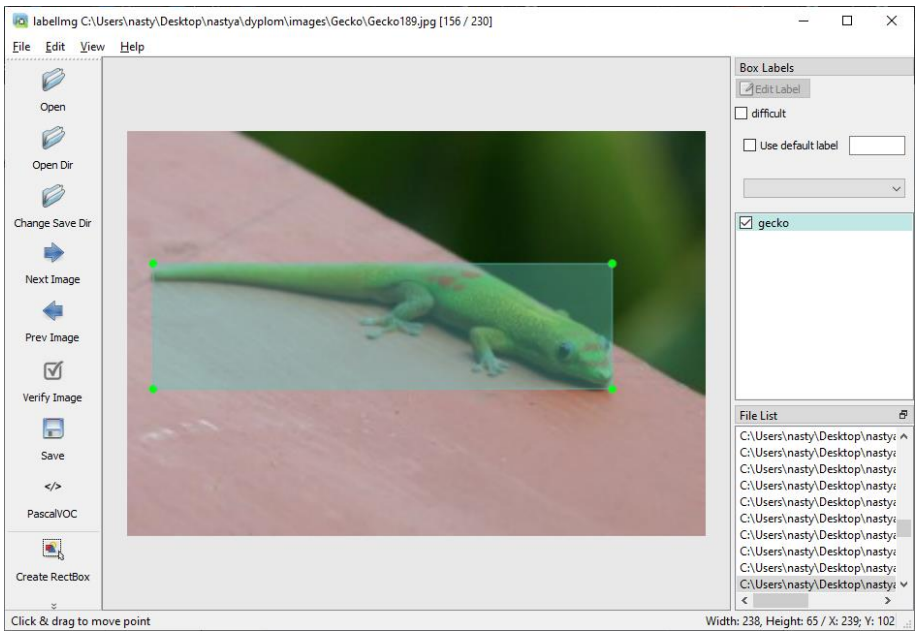

Рисунок 3.1 Приклад анотації зображення в LabelImg

На виході отримуємо xml-файл з інформацією про саме фото (довжина, ширина, глибина), визначеними координатами рамки, назвою позначеного об’єкту і тому подібне. Витягуємо дані в таблицю, також розраховуємо додаткові параметри, наприклад середина рамки, її довжина та ширина:

	filename	width	height	name	xmin	xmax	ymin	ymax	center_x	center_y	w	h
0	Gecko189.jpg	300	210	gecko	13	252	69	134	0.441667	0.483333	0.796667	0.309524
1	Gecko190.jpg	300	200	gecko	32	283	72	144	0.525000	0.540000	0.836667	0.360000
2	Gecko191.jpg	207	300	gecko	17	193	14	286	0.507246	0.500000	0.850242	0.906667
3	Gecko193.jpg	300	169	gecko	106	258	62	120	0.606667	0.538462	0.506667	0.343195
4	Gecko218.jpg	271	300	gecko	34	245	17	226	0.514760	0.405000	0.778598	0.696667

Рисунок 3.2

Оскільки нам не потрібна вся інформація, беремо лише потрібні нам дані з кожного xml-файлу та записуємо у звичайний текстовий файл. Id класу до якого відноситься об’єкт, центр рамки по x та по y, ширина та висота рамки.

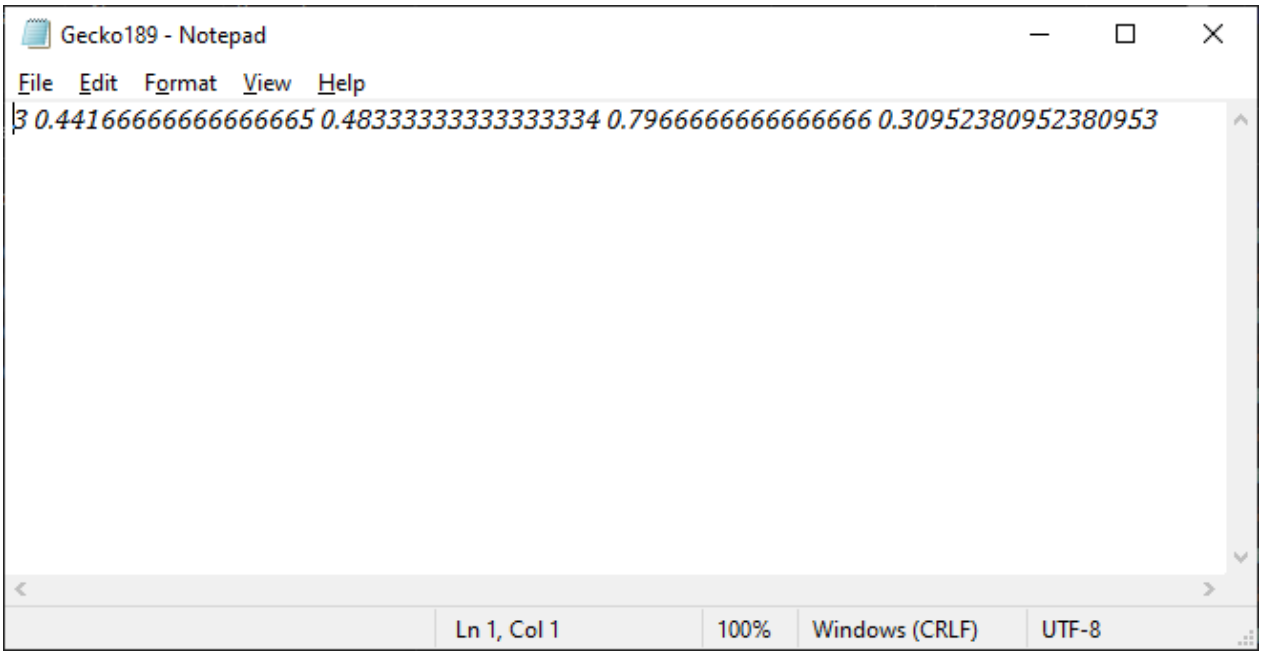

Рисунок 3.3

Ділимо всі елементи в папці на тренувальні (80%) та тестувальні (20%) дані та завантажуємо у відповідні папки на GoogleDrive.

Повторюємо ці процедури з іншими зображеннями рептилій кожного виду. Загалом таким чином було оброблено 1135 фотографій 8-ми видів рептилії, 906 з яких в тренувальній папці, а 229 в тестувальній.

Таблиця 3.1 - Назви класів та кількість даних в них

Назва	**Тренувальна**	**Тестувальна**
Chameleon	139	33
Crocodile	108	28
Frog	101	23
Gecko	103	29
Iguana	106	26
Lizard	137	38
Snake	105	26
Turtle	107	26
Загалом	**906**	**229**

Кількість появ класів плазунів може відрізнятись від кількості зображень плазунів, оскільки на одному фото могло бути більше одного плазуна, або ж декілька їх видів.

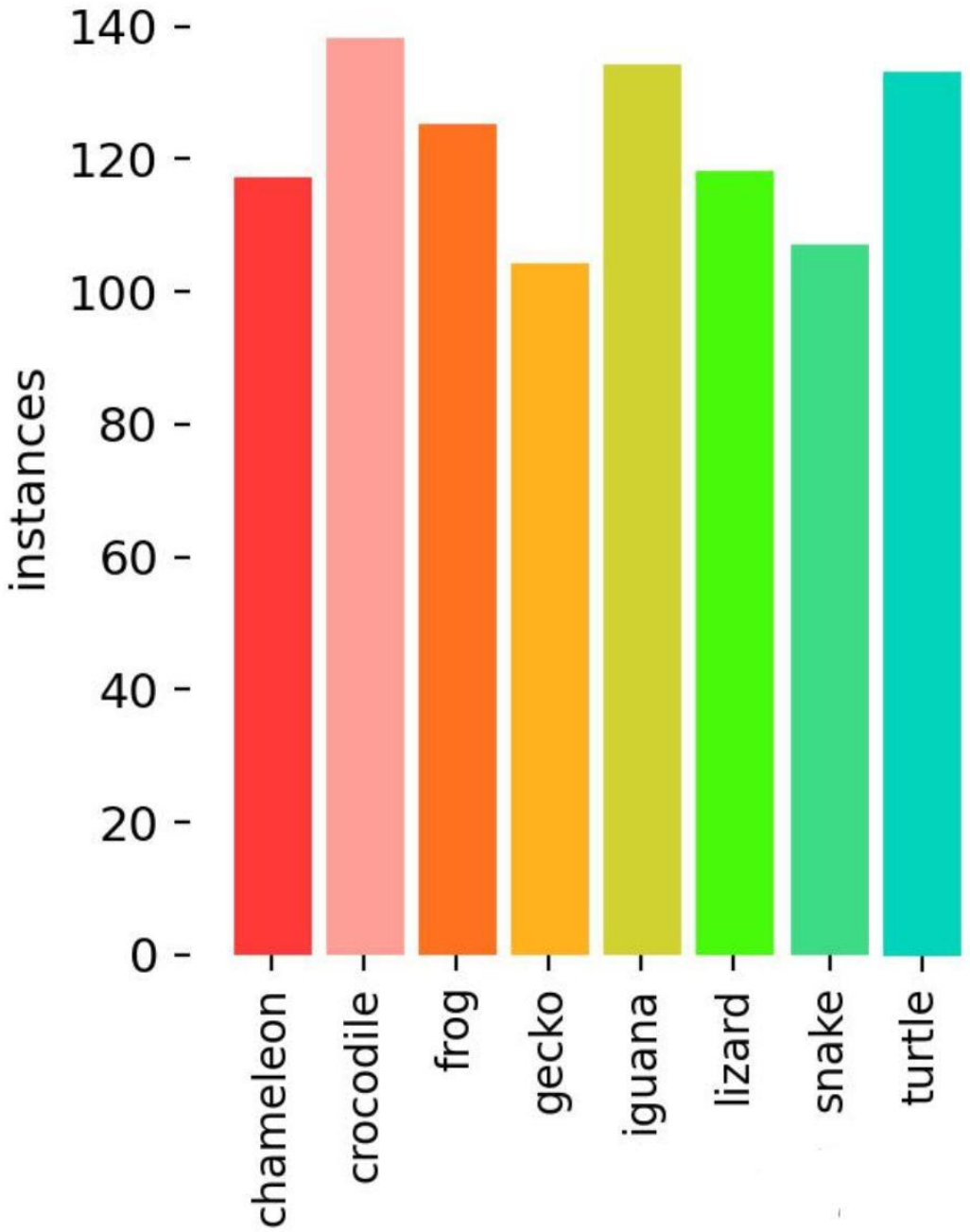

Рисунок 3.4 Представлення даних у вигляді діаграми

3.4 Архітектура моделі

Модель YOLOv5s має 24 шари:

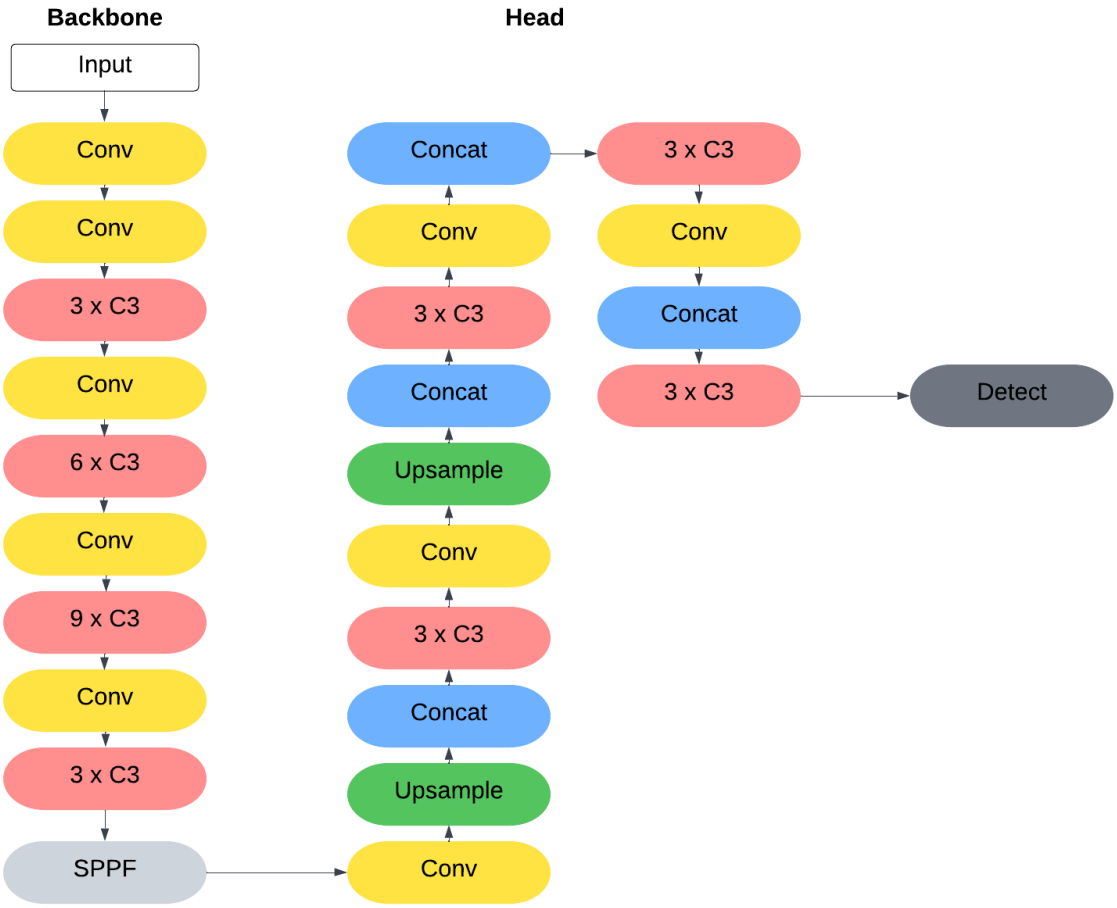

Рисунок 3.5 Архітектура моделі YOLOv5s

Conv — застосовує функцію Sigmoid Linear Unit (SiLU) поелементно;

C3 — CSP Bottleneck з 3 згортками;

SPPF — Spatial Pyramid Pooling - Fast шар для YOLOV5 від Glenn Jocher;

Upsample — збільшує дискретизацію заданих багатоканальних 1D (часових), 2D (просторових) або 3D (об’ємних) даних;

Concat — об'єднує список тензорів (скаляри, вектори, лінійні оператори) уздовж розміру.

3.5 Тренування моделі

Тренування моделі відбувалось на платформі Google Colab, адже там можна під’єднатись до Google Drive де знаходяться всі необхідні, попередньо завантажені дані.

Репозиторій YOLOv5 було клоновано з GitHub за допомогою «!git clone https://github.com/ultralytics/yolov5».

У каталозі YOLOv5 є файл «requirements.txt», який містить список необхідних бібліотек і версій для запуску моделі YOLOv5. Потрібну бібліотеку було встановлено за допомогою команди bash «!pip install -r requirements.txt». Варто згадати, що «pip» — це менеджер пакунків Python.

Було вибрано відповідні параметри для навчання нещодавно розробленого набору даних, шляхом практичного підбору. Для навчання моделі було обрано 49 епох, що зайняло приблизно 30 хв.

Тренування моделі:
«!python train.py --data data.yaml --cfg yolov5s.yaml --batch-size 10 --name Model --epochs 50»

У файлі data.yaml написані відповідні каталоги для тренувальних і тестувальних зображень, кількість класів - 8, та їхні назви. Модель — YOLOv5s, кількість навчальних прикладів, які використовуються в одній ітерації — 10, кількість епох — 50. Тренування останньої моделі зайняло приблизно 30 хвилин.

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       0/49       2.2G    0.07885    0.02889    0.05921         19        640: 100% 91/91 [00:27<00:00,  3.32it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.11it/s]
                   all        229        240     0.0506      0.239     0.0589     0.0197

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       1/49      2.89G     0.0539    0.02465    0.05355         22        640: 100% 91/91 [00:24<00:00,  3.71it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.61it/s]
                   all        229        240       0.13      0.175     0.0834     0.0291

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       2/49      2.89G    0.04895    0.02153    0.04959         12        640: 100% 91/91 [00:24<00:00,  3.72it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.62it/s]
                   all        229        240       0.24       0.23      0.117     0.0458

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       3/49      2.89G    0.04808    0.02151    0.04611         22        640: 100% 91/91 [00:24<00:00,  3.68it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.89it/s]
                   all        229        240      0.163      0.292      0.168     0.0722

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       4/49      2.89G     0.0475    0.02059    0.04588         15        640: 100% 91/91 [00:24<00:00,  3.68it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.54it/s]
                   all        229        240      0.182       0.26      0.158      0.065
```

Рисунок 3.6 Початок тренування моделі

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      45/49      2.89G     0.0256    0.01467    0.01532         16        640: 100% 91/91 [00:24<00:00,  3.67it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  5.01it/s]
                   all        229        240      0.795      0.801      0.831      0.503

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      46/49      2.89G    0.02682    0.01524    0.01386         18        640: 100% 91/91 [00:24<00:00,  3.77it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.42it/s]
                   all        229        240      0.819      0.783      0.842      0.502

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      47/49      2.89G    0.02428    0.01474    0.01454         14        640: 100% 91/91 [00:24<00:00,  3.67it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.69it/s]
                   all        229        240      0.834      0.796      0.846      0.496

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      48/49      2.89G     0.0237     0.0153    0.01349         14        640: 100% 91/91 [00:24<00:00,  3.75it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.73it/s]
                   all        229        240      0.793        0.8       0.84      0.514

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      49/49      2.89G    0.02396    0.01502    0.01361         20        640: 100% 91/91 [00:24<00:00,  3.70it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 12/12 [00:02<00:00,  4.34it/s]
                   all        229        240      0.801      0.823      0.851      0.517
```

Рисунок 3.7 Кінець тренування моделі

```
50 epochs completed in 0.388 hours.
Optimizer stripped from runs/train/Model14/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/Model14/weights/best.pt, 14.4MB

Validating runs/train/Model14/weights/best.pt...
Fusing layers...
YOLOv5s summary: 157 layers, 7037095 parameters, 0 gradients
                 Class     Images  Instances          P          R
                   all        229        240      0.801      0.823
             chameleon        229         31      0.859      0.806
             crocodile        229         33      0.926      0.757
                  frog        229         29      0.832      0.931
                 gecko        229         32      0.751      0.753
                iguana        229         30      0.772      0.789
                lizard        229         28      0.715      0.893
                 snake        229         26      0.742      0.846
                turtle        229         31      0.809      0.806
Results saved to runs/train/Model14
```

Рисунок 3.8 Опис результатів тренування моделі

Точність натренованої моделі (precision) — 0.801 (в класах варіюється від 0.715 до 0.926)

Повнота натренованої моделі (recall) — 0.823 (в класах варіюється від 0.757 до 0.931)

TP = True Positives (позитивний і правильний прогноз)
FP = False Positives (позитивний, але неправильний прогноз)
FP = False Negatives (негативний і неправильний прогноз)

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

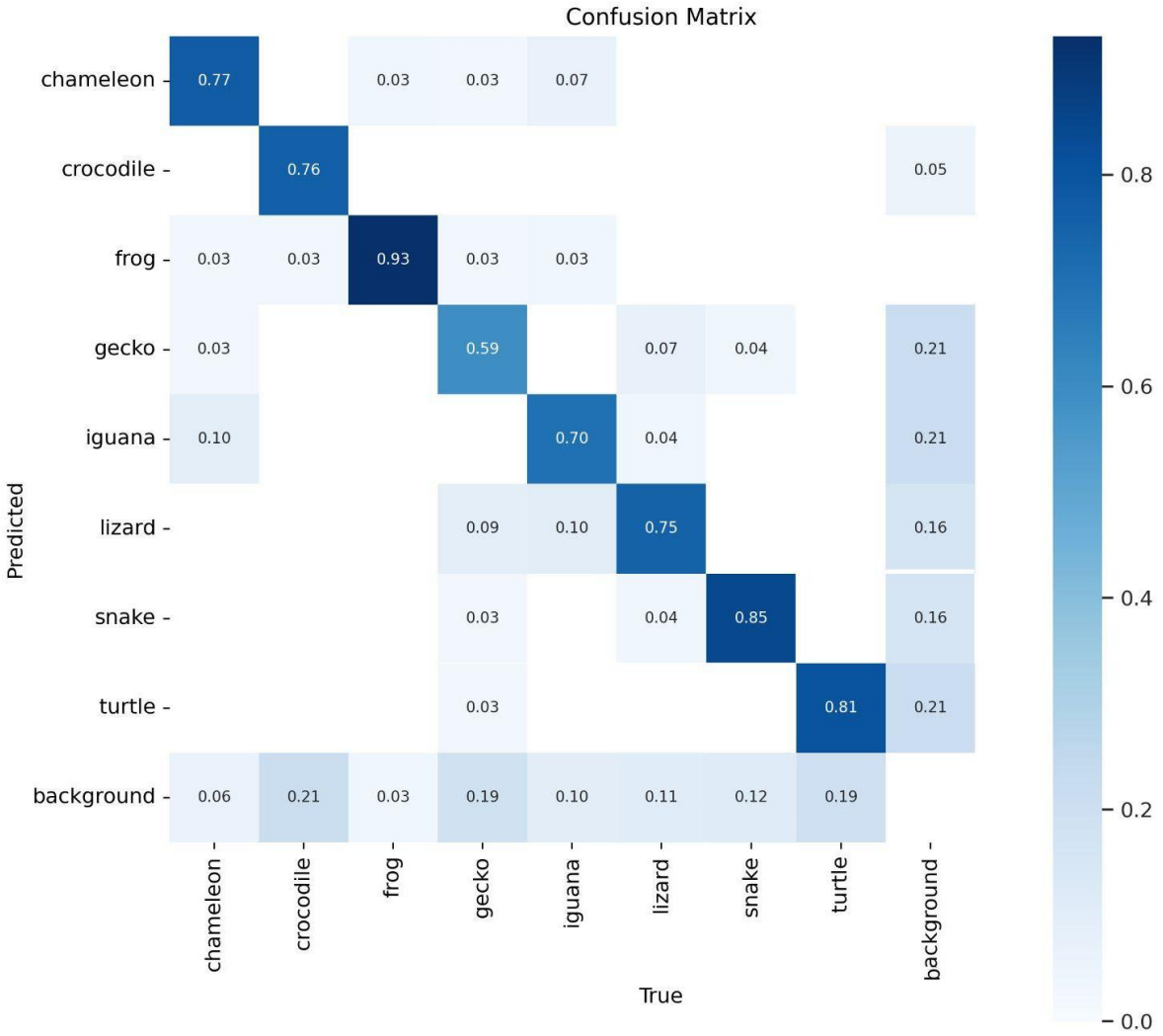

Рисунок 3.9 Матриця невідповідностей

Виявлені рептилії на тестових зображеннях після тренування моделі:

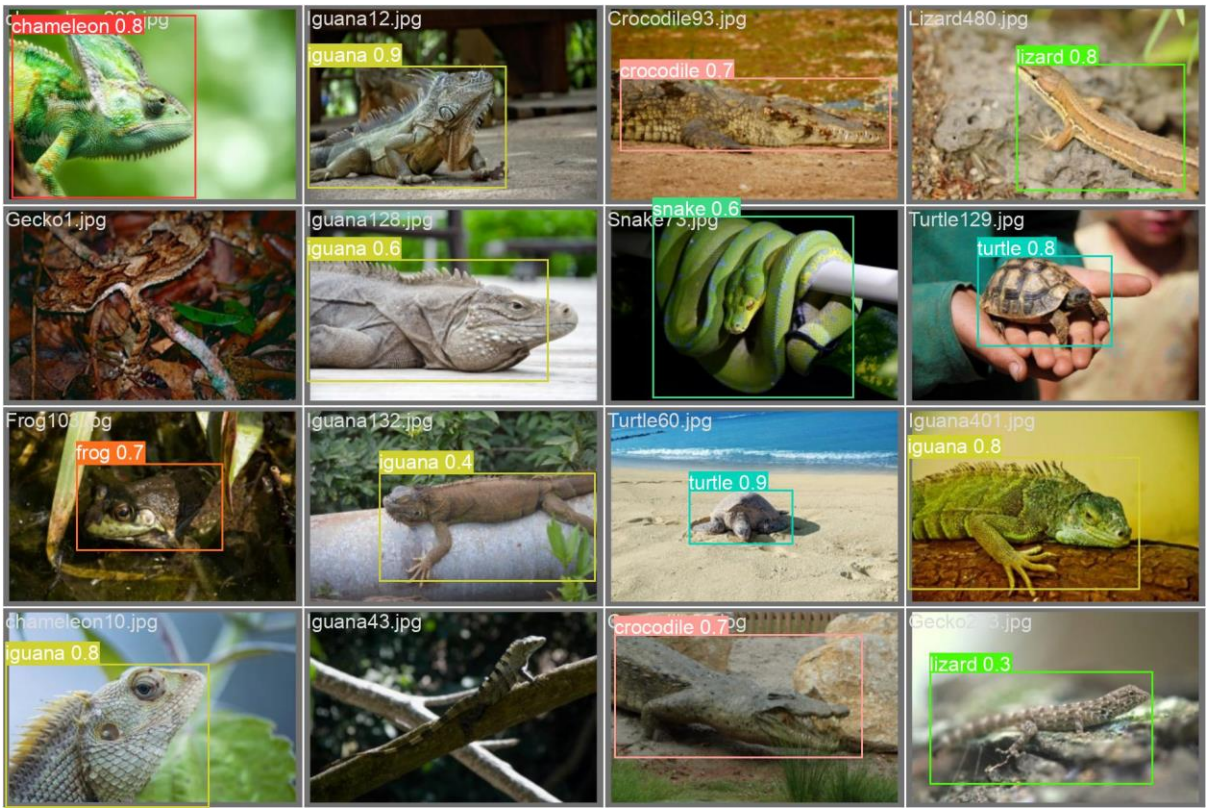

Рисунок 3.10

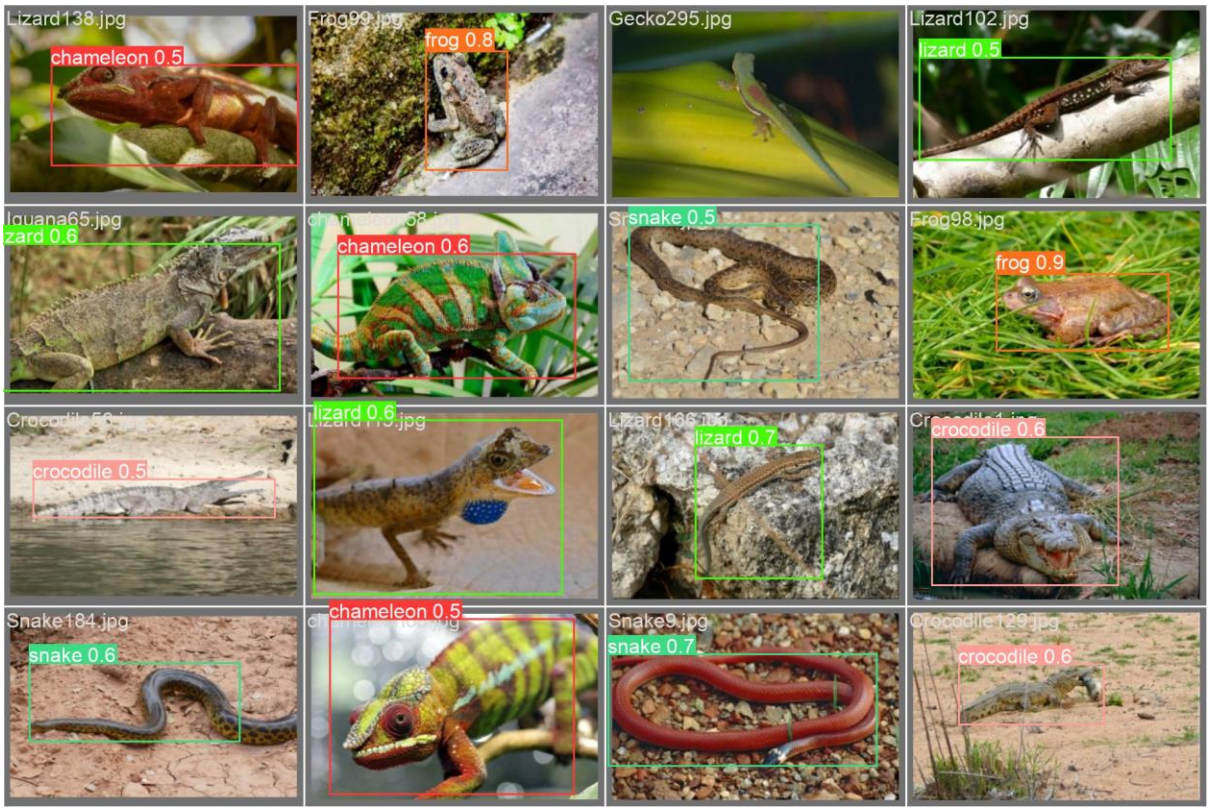

Рисунок 3.11

4 АПРОБАЦІЯ

Після збереження натренованої моделі, я створила віконний застосунок на мові Python, використавши графічну бібліотеку Tkinter для зручного використання моделі для розпізнавання рептилій на фото. Зображення для тестування були взяті з пошукової системи Google. Кнопкою «select an image» завантажуємо зображення з комп’ютера та після натискання «detect» з’являється нове вікно з обраним зображенням та окресленими рамками рептиліями на ньому, якщо такі були знайдені.

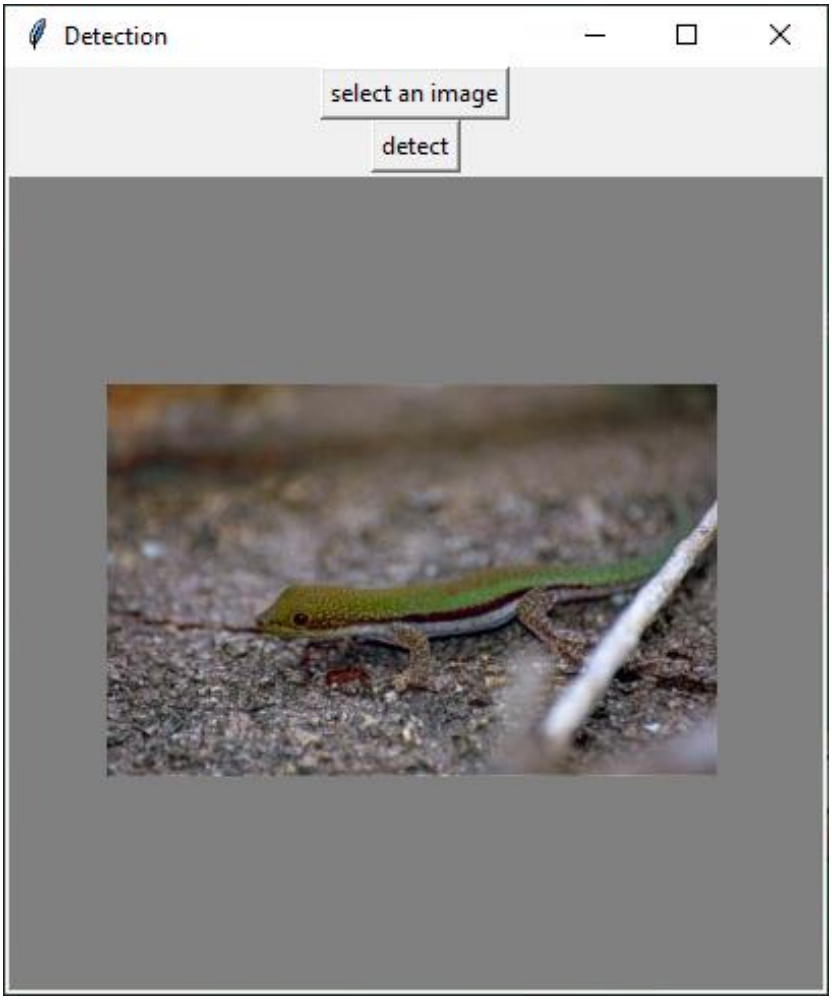

Рисунок 4.1 Віконний застосунок

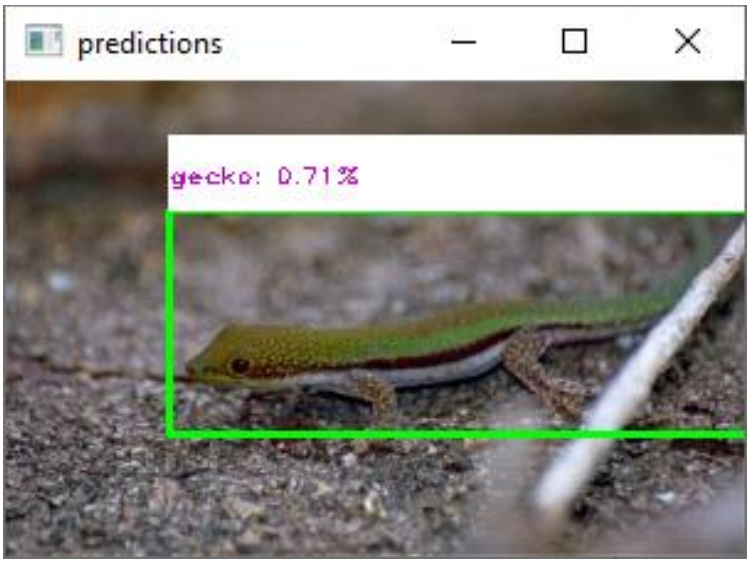

Рисунок 4.2 Результати розпізнавання в застосунку

Нижче наведено ще декілька прикладів розпізнавання рептилій на зображеннях в застосунку:

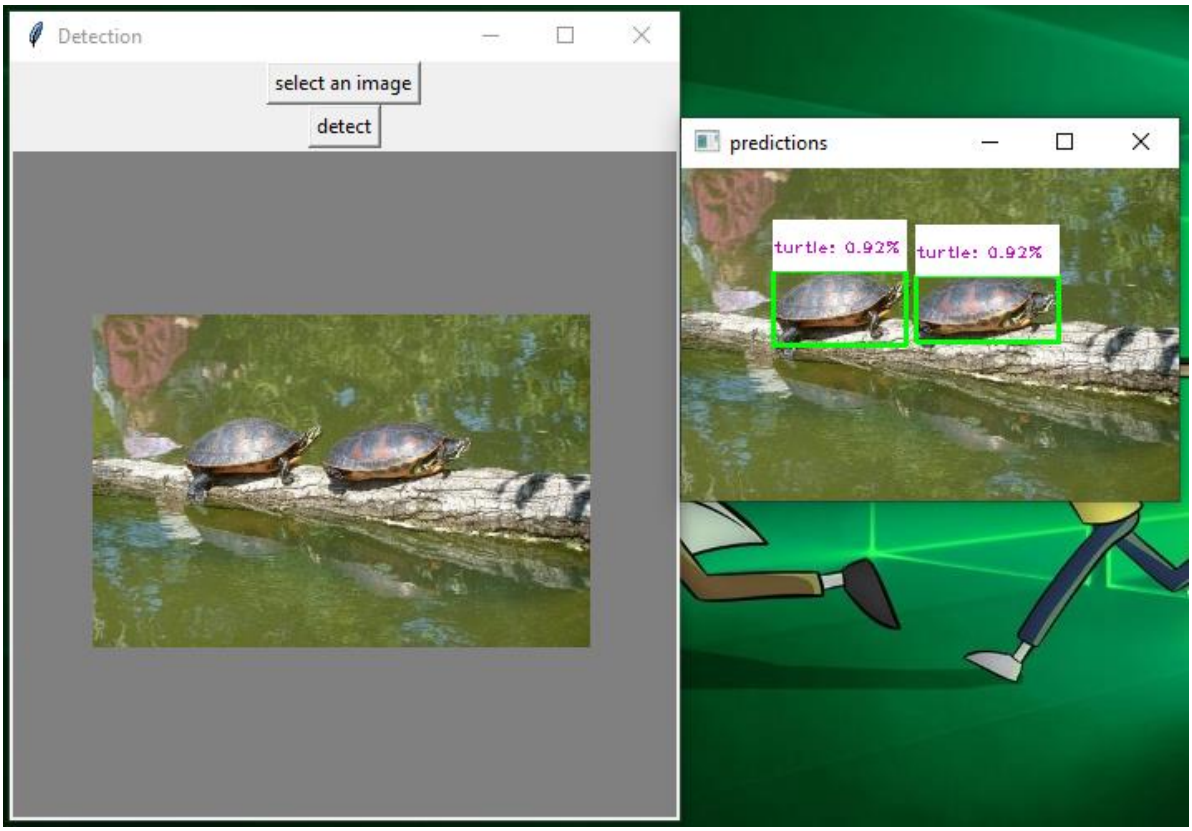

Рисунок 4.3

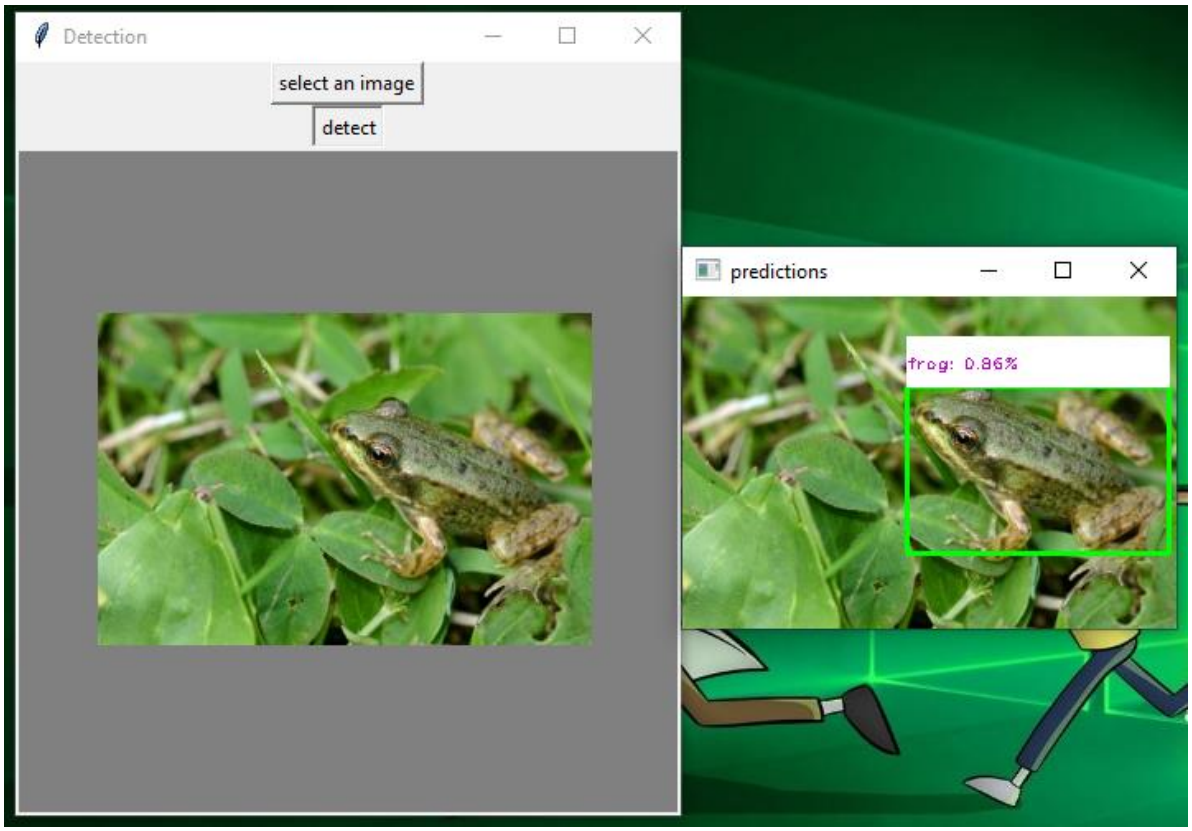

Рисунок 4.4

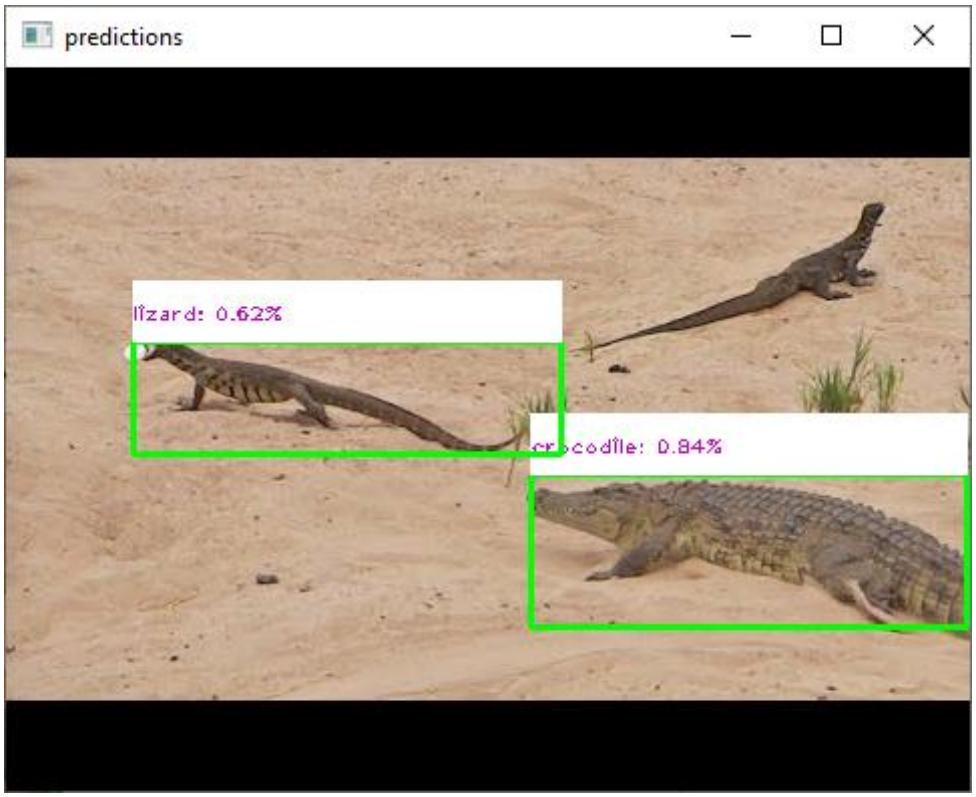

Рисунок 4.5

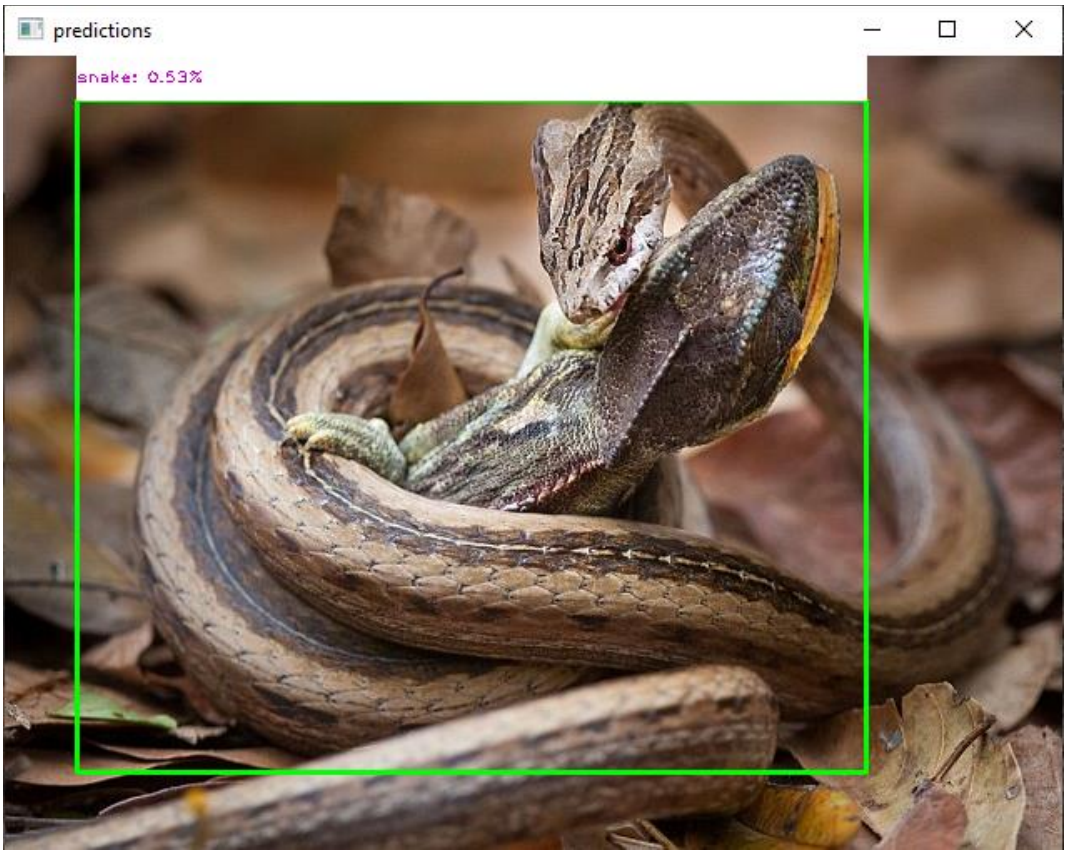

Рисунок 4.6

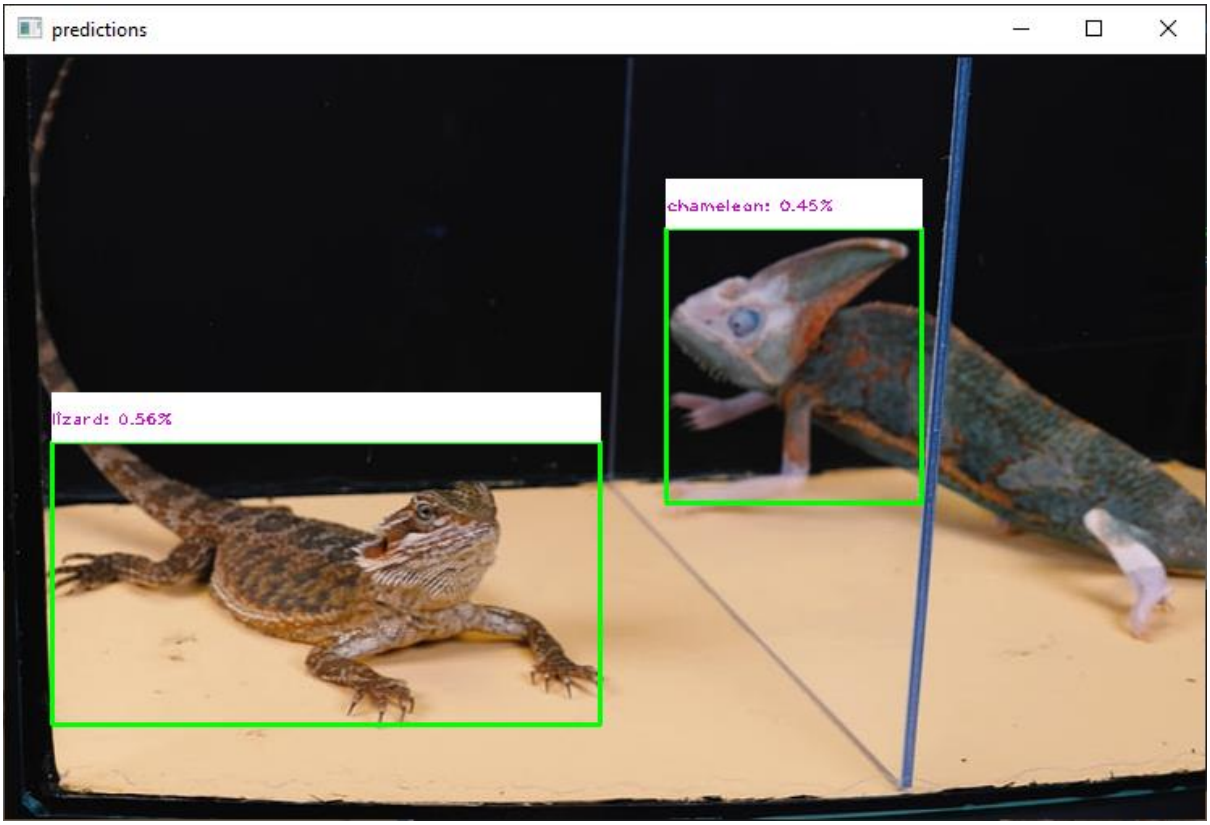

Рисунок 4.7

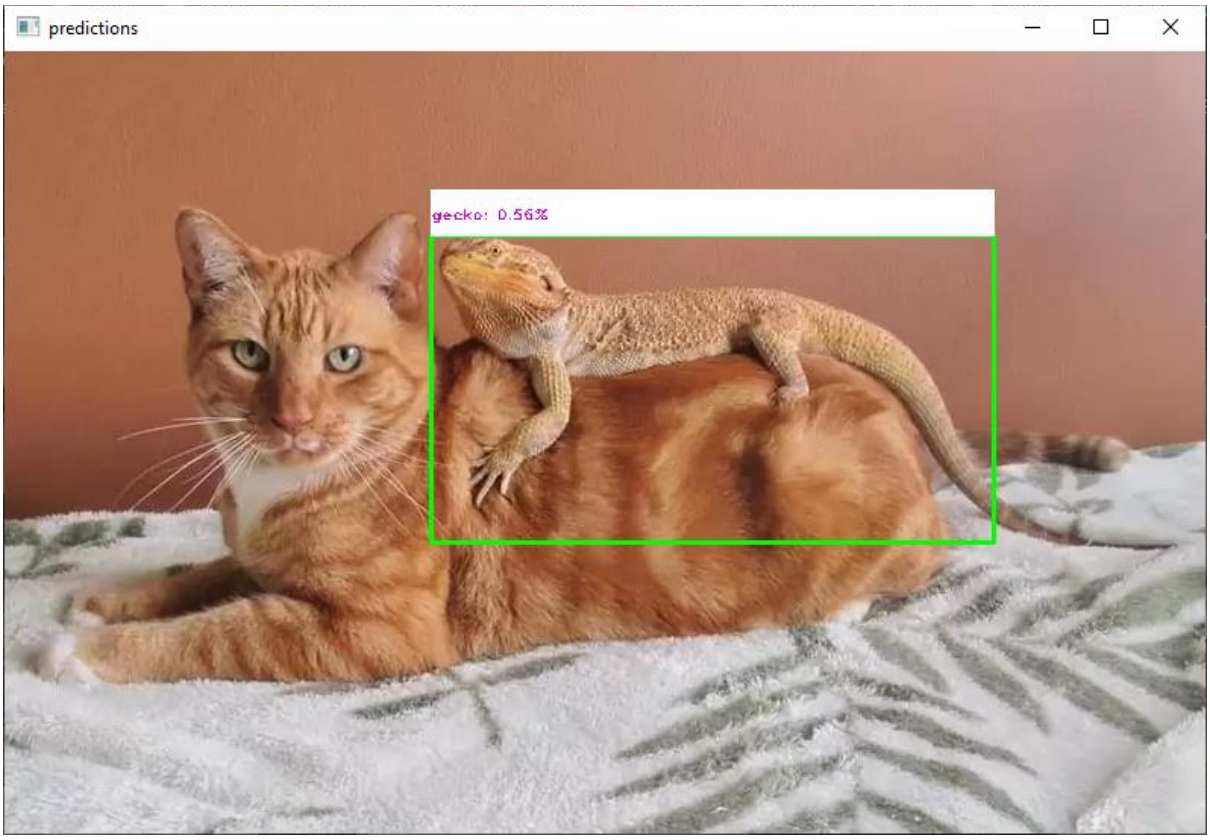

Рисунок 4.8

ВИСНОВКИ

Під час написання роботи було зібрано дані у вигляді зображень різних видів рептилій. Для реалізації цієї задачі використано нейронну мережу YOLOv5, що складається з 24 шарів, в тому числі згорткових. Натреновано модель для розпізнавання та класифікації плазунів на вхідних зображеннях з точністю до 85%. В результаті на зображеннях можна виявити до восьми видів рептилій: хамелеонів, крокодилів, жаб, геконів, ігуан, ящірок, змій та черепах.

Також було створено віконний застосунок для практичного застосування моделі з можливістю завантажити фото з комп’ютера та отримати результат у вигляді зображення з окресленими на ньому знайденими плазунами. Код програми доступний за першим посиланням у списку використаних джерел.

Ця робота може знайти своє застосування у вивченні та виявленні плазунів в дослідницьких, наукових та сільськогосподарських сферах. Бути корисною для подальшого індивідуального користування, або ж основою для майбутніх покращених застосунків для розпізнавання та класифікації обраних рептилій та інших їх видів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. URL: https://github.com/nastyaPats/ReptilesDetection.git
2. Yao, J.; Qi, J.; Zhang, J.; Shao, H.; Yang, J.; Li, X. A Real-Time Detection Algorithm for Kiwifruit Defects Based on YOLOv5. Electronics 2021, 10, 1711. URL: https://doi.org/ 10.3390/electronics10141711
3. Zelinsky, A. Learning OpenCV — Computer Vision with the OpenCV Library, 2009.
4. VanderPlas, J. Python Data Science Handbook: O'Reilly. 2016.
5. Python 3.9.0 documentation URL: https://docs.python.org/release/3.9.0/
6. URL:https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
7. URL: https://github.com/ultralytics/yolov5
8. Ian Goodfellow, Yoshua Bengio, and Aaron Courville — Deep Learning (Adaptive Computation and Machine Learning series), 2016.
9. Fei-Fei Li, Justin Johnson, Serena Yeung — Convolutional Neural Network for Visual Recognition