КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

РОЗПІЗНАВАННЯ РУХОМИХ ОБ'ЄТІВ В ПРОСТОРІ

Дипломна робота магістра

Спеціальність: 123 «Комп'ютерна інженерія»

Лещука Олександра Сергійовича

_____(підпис)

Науковий керівник

доктор технічних наук, професор

Погорілий Сергій Дем'янович

_____(підпис)

Рецензент

кандидат фізичко-математичних наук, доцент

Кулябко Петро Петрович

_____(підпис)

До захисту допускаю

Протокол засідання кафедри від

«__»_____ 2022 р. № __

Завідувач кафедри

к.ф-м.н., доцент

Бойко Юрій Володимирович

Київ - 2022

РЕФЕРАТ

Проведено аналіз наукових публікацій та створено застосування для класифікації моделей рухомих автівок на відео. Було визначено такі класи для моделей: Audi, Chevrolet, Toyota, Other. До класу Other відносяться автівки, які не відносяться до перших трьох класів.

Було проаналізовано можливість застосування різних типів нейронних мереж до поставленої задачі і також здійснено порівняльний аналіз бібліотек для навчання нейронних мереж. Сформовано і обґрунтовано вибір моделі нейронної мережі

Виконано програмну реалізацію застосунку мовою Python для класифікації об'єктів з 4 класами на основі нейронної мережі Yolo v4. Експериментально показано, що цей тип нейронної мережі якісно вирішує поставлену задачу: класифікацію об’єктів на зображеннях/відео.

Проведено навчання мережі та досягнуто точність розпізнавання близько 93% на валідаційній вибірці і 91% на тестовій вибірці. Було проведення тестування застосування для визначення моделі автівок на світлинах та відео.

Випускна дипломна робота магістра: 43 сторінки, 8 рисунків, 1 таблиця, 1 додаток, 14 посилань.

Ключові слова: згорткові нейронні мережі, класифікація об’єктів, розпізнавання образів, комп’ютерний зір, аугментація даних, глибинне навчання, TensorFlow, Python, нейронна мережа Yolo v4.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

НМ – нейрона мережа

ГС – градієнтний спуск

CNN – Convolutional neural network (зготкова нейронна мережа)

MLP – Multilayer perceptron (багатошаровий перцептрон)

Backpropagation – метод зворотного розповсюдження помилки

TensorFlow – бібліотека машинного навчання

Loss function – функція втрат (ФВ)

Yolo v4 – нейрона мережа для класифікації об’єктів

Supervised learning – машинне навчання з учителем

Зміст

Вступ

Розпізнавання образів — це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних.

Завдяки нейронним мережам стався великий прорив в області машинного навчання. . Алгоритми, що були створені за принципом роботи людського мозку, дозволяють обробляти величезні обсяги даних. Вже сьогодні, штучний інтелект застосовується в багатьох напрямках, наприклад розпізнавання мови, жестів, тексту та проведення медичної діагностики.

Комп'ютерний зір належить до теорії та технології створення штучних систем, які отримують інформацію у вигляді зображень. Відеодані можуть бути представлені у вигляді багатьох форм, таких як відео, зображення з різних камер або тривимірними даними з медичного сканера.

Комп'ютерний зір дозволяє виявляти об’єкти, розпізнавати, генерувати зображення та їх супер-розширення. Виявлення об’єктів базується на здатності комп’ютера і систем програмного забезпечення знаходити об’єкти на зображеннях і ідентифікувати їх. Такий підхід застосовується для виявлення обличь , транспортних засобів, підрахунку пішоходів, систем безпеки і машин без водія. Прорив і швидке впровадження глибокого навчання в 2012 році призвели до появи таких сучасних і високоточних алгоритмів і методів виявлення об’єктів як R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet і високоточних MobileNet SSD і YOLO.

Нейронні мережі після автоматизованого навчання дозволяють аналізувати складні структури даних, класифікувати різні об’єкти та знаходити певні закономірності. Зважаючи на все це, доцільно розглянути застосування нейронних до поставленої задачі, а саме дослідити існуючі нейронні алгоритми

і моделі, для того щоб на їх основі створити самокеровану, автоматизовану систему для виконання функції розпізнавання об’єктів.

Існує два підходи до розпізнавання об'єктів із використанням глибокого навчання:

- Навчання моделі з нуля. Щоб навчити глибоку мережу з нуля, необхідно збирати дуже великий набір даних і розробити архітектуру мережі, яка вивчатиме характеристики і будуватиме модель. Результати можуть бути вражаючими, але цей підхід вимагає великої кількості навчальних даних, а також необхідно налаштовувати рівні та ваги згорткової нейронної мережі.
- Використання попередньо вивченої моделі глибокого навчання: більшість додатків глибокого навчання використовують підхід трансферного навчання – процес, який включає точне налаштування попередньо вивченої моделі. Спочатку береться існуюча мережа, така як AlexNet або GoogLeNet, і вводяться нові дані, що містять раніше невідомі класи. Цей метод вимагає менше часу і може забезпечити швидший результат, оскільки модель вже навчена на тисячах чи мільйонах зображень.

1. Аналіз штучних нейронних мереж для розпізнавання відео-даних та зображень.

1.1. Штучні нейронні мережі.

1.1.1. Загальні принципи побудови нейронних мереж.

Штучна нейронна мережа – це математична модель, основою для розробки якої став спосіб організації та принцип функціонування головного мозку. Кожен нейрон залежить від інших нейронів, які входять до нього, виконує обчислення і потім активується (якщо результат обчислення перевищує якийсь поріг), чи ні (якщо не перевищує).

Нейронна мережа у людському мозку - гігантська взаємопов'язана система нейронів, де сигнал, який передається одним нейроном, може передаватися у тисячі інших нейронів. Відповідно, штучні нейронні мережі складаються зі штучних нейронів, які проводять аналогічні обчислення над своїми входами. Нейронні мережі можуть вирішувати широке коло завдань, серед яких такі як розпізнавання рукописного тексту та розпізнавання обличь.

Нейронні мережі інтерпретують отримані дані та проводять їх маркування. Мережі розпізнають тільки цифрові дані, що містяться в векторах ознак, тому всі данні повинні бути переведені в такі вектори, будь то зображення, звук, текст або тимчасові ряди. Нейронні мережі групують і класифікують дані.

Навчання відбувається через повторну активацію деяких нейронних з'єднань (синапсів), що збільшує ймовірність виведення потрібного результату при відповідній вхідній інформації (сигналі). Такий вид навчання використовує зворотний зв'язок - при правильному результаті нейронні зв'язки, які виводять його, стають щільнішими.

Нейронні мережі [1] інтерпретують отримані дані та проводять їх маркування. Мережі розпізнають тільки цифрові дані, що містяться в векторах ознак, тому всі данні повинні бути переведені в такі вектори, будь то зображення, звук, текст або тимчасові ряди. Нейронні мережі групують і класифікують дані.

Як правило, такі нейронні мережі для проведення свого навчання потребують наявності навчальної вибірки даних та складаються із трьох компонентів: вхідного, прихованого (обчислювального) та вихідного шарів. Шари, у свою чергу, містять у своєму складі зв’язані між собою нейрони, на кожен з яких надходить вхідний сигнал з певною вагою.

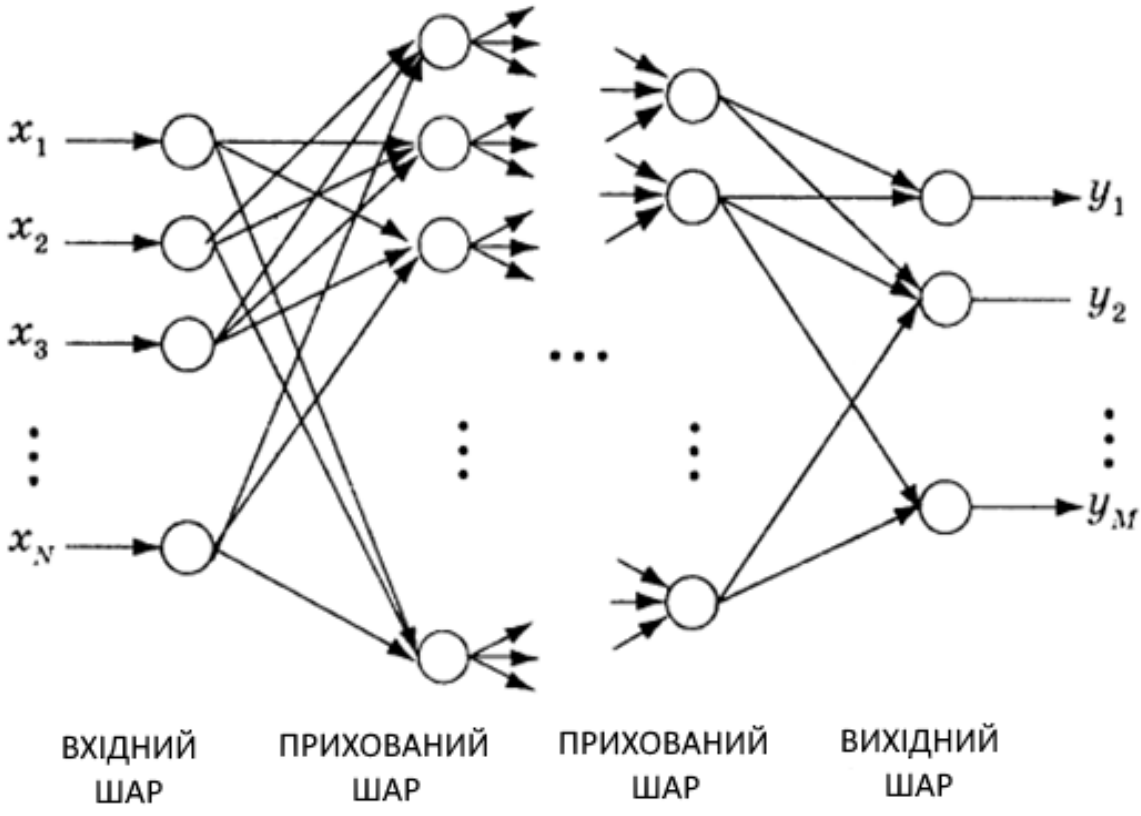

Рис. 1.1. Загальна структура нейронної мережі

Кожен нейрон примножує початкове значення на деяку вагу, підсумовує результати з іншими значеннями, що надходять у один і той же нейрон, регулює отримане число за допомогою нейрону відхилення, а потім нормалізує вихід із функцією активації.

Опис зв'язків відбувається за допомогою векторів та матриць у які записується вагові коефіцієнти ребер між нейронами. А навчання мережі полягає у знаходженні цих коефіцієнтів.

Одним з найбільш поширених способів машинного навчання є нейронних мереж прямого розповсюдження (прямого поширення), для якої характерною є прямолінійність, оскільки проходження сигналу відбувається в одному напрямку. Класичним видом нейронних мереж прямого поширення для роботи з табличними наборами даних є багатошаровий перцептрон (MLP), загальна архітектура якого продемонстрована на рис. 1.1. Це модель, в якій дані подаються на вхідний шар що сполучений з одним або декількома прихованими шарами нейронів, задача яких – забезпечення рівня абстракції. Вихідний шар є останнім з компонентів, на якому робляться прогнози.

Якщо у мережі є достатня кількість нейронів прихованого шару, то вона здатна провести навчання, використовуючи пряме розповсюдження сигналу між вхідним та вихідними шарами. Перцептрон є зручним як у випадку табличного вигляду даних, так і для класифікації графічних зображень, часових рядів, різних типів документів. MLP також застосовний не лише у ролі окремої моделі для вирішення звичайних задач машинного навчання, а й використовується у складі багатьох складних технологій.

Через свою здатність відтворювати та моделювати нелінійні процеси, нейронні мережі знайшли застосування в широкому діапазоні дисциплін: від керування транспортними засобами, передбаченнями траєкторії, перемоги

чемпіонів у іграх, оптимізації фінансових процесів до моделювання природніх та інженерних моделей, розпізнавання образів та об’єктів, медичної діагностики та багато інших.

1.1.2. Методи навчання нейронної мережі.

Машинне навчання з учителем (supervised learning) [1] включає моделювання ознак даних і відповідних даним міток. Після вибору моделі її можна використовувати для присвоєння міток новим, невідомим раніше даним. У процесі класифікації мітки являють собою дискретні категорії, тому для цього обирається навчальна вибірка.

Безпосередній процес навчання проходить у 2 етапи:

- Пряме розповсюдження сигналу;
- Зворотне розповсюдження сигналу помилки.

Оскільки в деякому розумінні нейронна мережа є обчислювальним графом, інформація в ній поширюється поступово, починаючи від вхідного шару нейронів і після проходження через приховані шари до вихідних нейронів формується результат опрацювання сигналу. Один з прикладів прямого розповсюдження сигналу показаний на рисунку 1.2 [2].

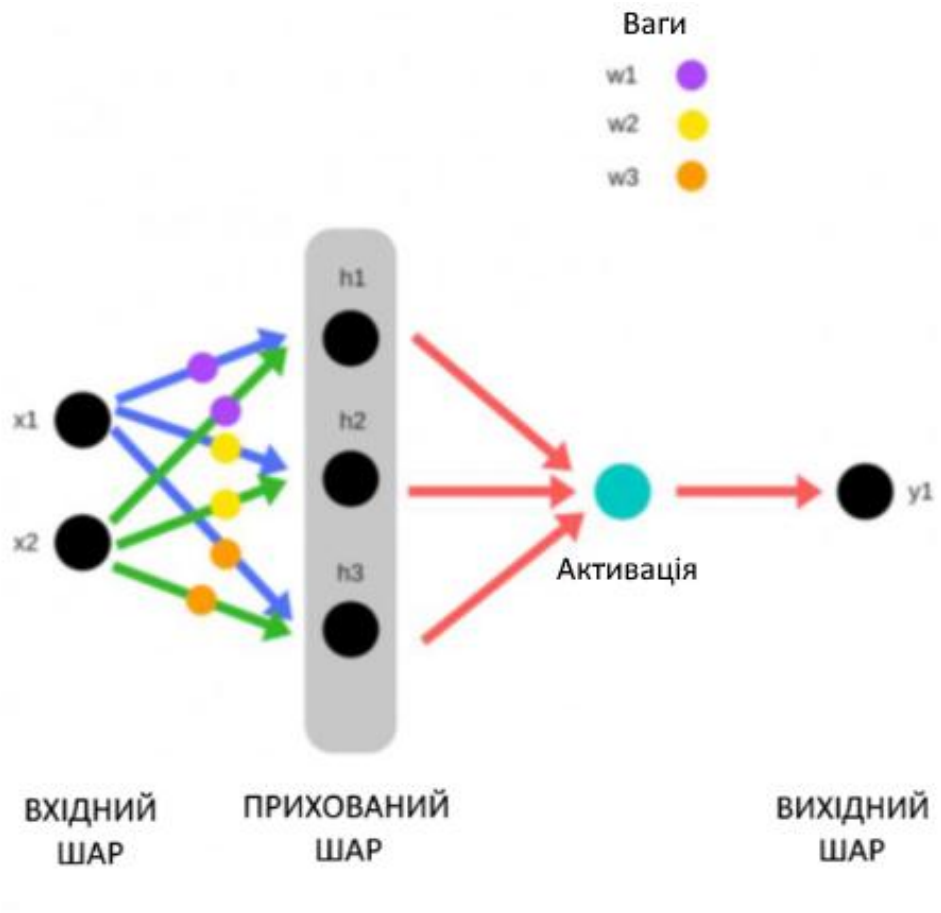

Рисунок 1.2. Пряме розповсюдження сигналу [2]

Задля мінімізації помилки та отримання бажаного результату використовується метод зворотного розповсюдження помилки (Backpropagation), основною ідеєю якого є поширення сигналів помилки від виходу мережі до її входів. Цей метод є ітеративним градієнтним алгоритмом; потребує, щоб функція нейронів була диференційованою. Недоліком застосування такого алгоритму є те, що у складних завданнях процес навчання може проходити дуже довго і в результаті не дати покращень точності через:

- Знаходження одного з локальних мінімумів (при наявності поруч декількох більш глибоких);
- вибір неоптимального розміру кроку.

Для оптимізації процесу навчання нейронних мереж використовують методи на основі знаходження градієнта функції втрат, серед яких найбільш поширеним є стохастичний градієнтний спуск. При градієнтному спуску коригування параметрів моделі використовується градієнт (зазвичай, обчислюється як сума градієнтів кожного елементу навчання), для якого необхідним є лише один прохід по навчальним даним. При стохастичному градієнтному спуску – значення градієнту функції апроксимується вартісною функцією (cost function), обрахованою лише на одному елементі. Як підсумок, параметри моделі змінюються пропорційно наближеному градієнту – після кожного об'єкта навчання. Для великих масивів даних стохастичний градієнтний спуск може дати значну перевагу в швидкості в порівнянні зі стандартним градієнтним спускок.

Нейронні мережі мають велике різноманіття архітектур, кожна з яких використовується для виконання завдань, для яких дана архітектура найкраще підходить.

В цілому можна виділити два типи нейронних мереж:

- згорткові нейронні мережі
- рекурентні нейронні мережі

1.1.3 Згорткові нейронні мережі.

Для вирішення поставленої задачі була обрана згорткова нейронна мережа (Convolutional Neural Network). Вона має таку назву, оскільки «згортка» - один з основних етапів роботи даної мережі.

Починаючи з 2011 року [3], передовою в мережах прямого поширення глибинного навчання була послідовність згорткових шарів та шарів

максимізаційного агрегування увінчаних декількома повно- або частково зв'язаними шарами, за якими йде рівень остаточної класифікації. Навчання зазвичай виконується без спонтанного попереднього навчання. Мережа, створена таким керованим методом глибинного навчання, називається згортковою та була першою, що досягла в певних задачах продуктивності, порівняної з людською. Саме такого типу й використовується нейронна мережа у цій роботі.

Згорткова нейронна [3] мережа може мати десятки або сотні шарів, кожен з яких навчається виявляти різні особливості зображення. Фільтри застосовуються до кожного навчального зображення з різною роздільною здатністю, і вихідні дані кожного згорнутого зображення використовуються як вхідні дані для наступного шару. Фільтри можуть починати обробки найпростіших характеристик зображення, таких як яскравість та краї, та ускладнюватися до характеристик, які однозначно визначають об'єкт.

Згорткові нейронні мережі виконують ідентифікацію та класифікацію зображень, тексту, звуку та відео.

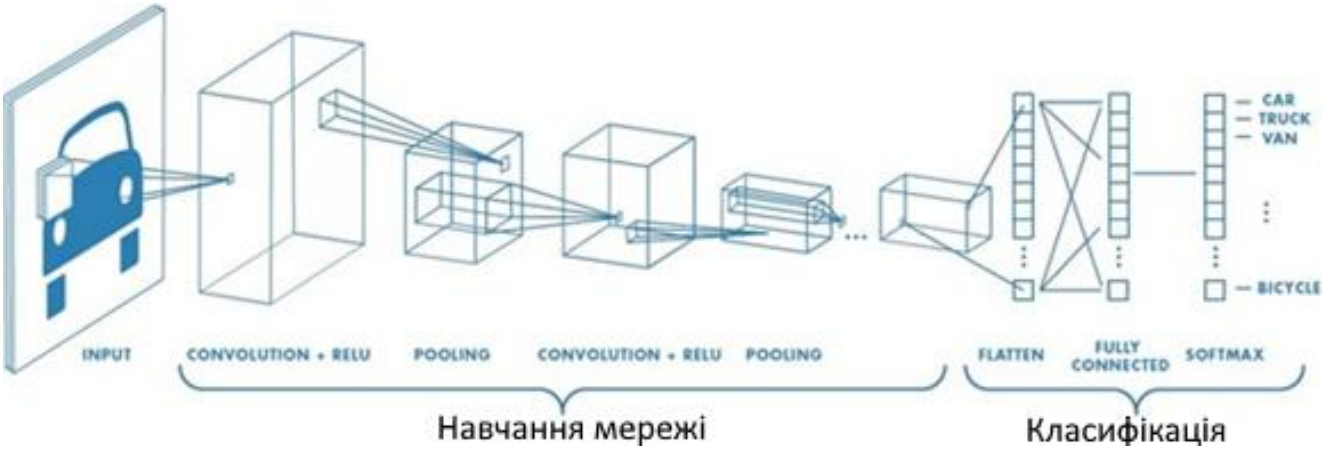

Рис 1.3. Модель згорткової нейронної мережі [3]

Як і інші нейронні мережі, згорткової нейронної мережі складається з вхідного шару, вихідного шару та безлічі прихованих шарів між ними (Рис. 1.3). Ці рівні виконують операції, які змінюють дані з вивчення характеристик, специфічних для цих даних. Три найбільш поширені рівні: згортка, активація або ReLU і об'єднання (pooling).

- Згортка пропускає вхідні зображення через набір згорткових фільтрів, кожен з яких активує певні характеристики зображень.
- Випрямлений лінійний блок (ReLU) дозволяє проводити більш швидке та ефективне навчання, відображаючи негативні значення в нуль та збереження позитивних значень. Іноді це називають активацією, тому що лише активовані характеристики переносяться на наступний рівень.
- Об'єднання (pooling) спрощує висновок, виконуючи нелінійне зниження якості зображення, зменшуючи кількість параметрів, які потрібно вивчити мережі.

Ці операції повторюються на десятках чи сотнях шарів, при цьому кожен шар вчиться визначати різні характеристики.

Після вивчення характеристик на багатьох рівнях архітектура CNN переходить до класифікації.

Передостанній шар - це повністю пов'язаний шар, який виводить вектор розміром K, де K - кількість класів, які мережа зможе передбачити. Цей вектор містить ймовірності для кожного класу будь-якого зображення, що класифікується.

Останній шар мережі архітектури використовує рівень класифікації, такий як softmax, для забезпечення виведення класифікації.

1.1.4 Рекурентні нейронні мережі

Рекурентними називають такі нейронні мережі у яких з’єднання між утворюють орієнтовний цикл. Мають такі характеристики:

- У кожного з’єднання є своя вага, вона ж пріоритет.
- Вузли поділяються на два типи, ввідні вузли і вузли приховані.
- Інформація у рекурентній нейронній мережі передається не тільки по прямій, шар за шаром, але і між самими нейронами.
- Важливою відмінною особливістю рекурентної нейронної мережі є наявність так званої «області уваги», коли можна задати певні фрагменти даних, що потребують посиленої обробки.

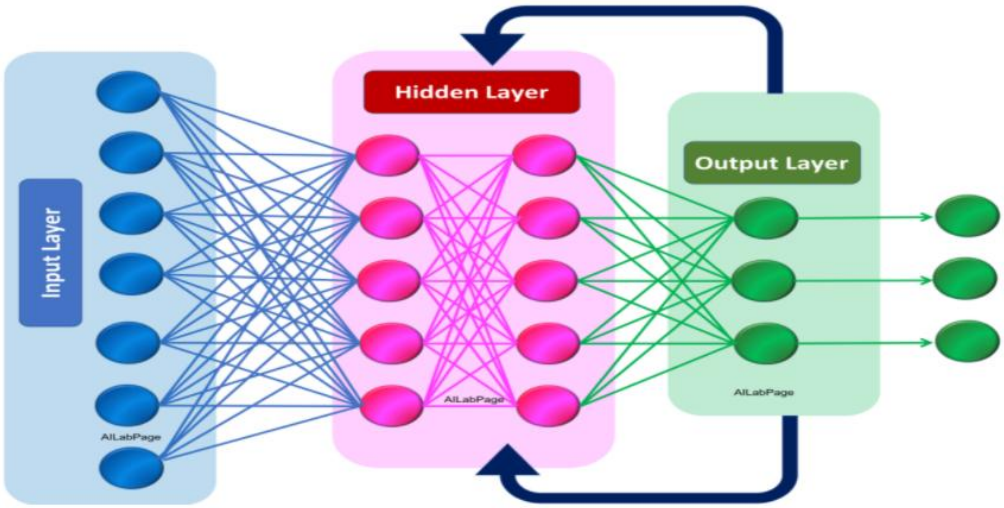

Рис 1.4. Модель рекурентної нейронної мережі [15]

1.2. Порівняльна характеристика бібліотек для роботи з нейронними мережами та засобів обробки інформації

Усі бібліотеки можна умовно поділити на символьні та несимвольні. Символьні обчислення — це перетворення і робота з математичними рівностями та формулами як з послідовністю символів. Вони відрізняються від числових розрахунків, які оперують наближеними числовими значеннями, що знаходяться поза математичними виразами.

Переваги несимвольних бібліотек:

- імперативні бібліотеки нейронних мереж, такі як torch і caffe, як правило, мають дуже схожий пристрій обчислювальної частини;
- з точки зору виразності імперативні бібліотеки влаштовані таким чином, що в них може бути інтерфейс на основі графів, як приклад - nngraph.

Недоліки несимвольних бібліотек:

- оптимізація вручну. Наприклад, операції на місці потрібно реалізувати вручну;
- більшість імперативних бібліотек поступаються символьним за виразністю.

Символьні бібліотеки мають такі переваги, як:

- можливість автоматичної оптимізації на основі графів залежностей;
- можна отримати набагато більше можливостей багаторазового використання пам'яті;
- можливість автоматично обчислювати оптимальний граф обчислень.

Основним недоліком є те, що доступні символьні бібліотеки з відкритим вихідним кодом на даний момент недостатньо розвинені і поступаються імперативним по продуктивності. [4]

Розробник	**TensorFlow**	**MXNET**	**Theano**	**PyTorch**
Мова програмування	C++, Python, Julia, Matlab, R, Scala	C++, Python	Python	Python
Підтримка CUDA	Так	Так	Так	Так
Підтримка рекурентних мереж	Так	Так	Так	Так
Підтримка згорткових мереж	Так	Так	Так	Так
Підтримка Hadoop	Ні	Так	Ні	Ні

Таблиця 1. Узагальнена порівняльна характеристика бібліотек Theano, MXNET та TensorFlow

Tensorflow — бібліотека з відкритим вихідним кодом для чисельного розрахунку з використанням графів потоку даних. Вузли графа представляють собою математичні операції, а ребра - багатовимірні масиви даних (тензори), які течуть між ними. Така гнучка архітектура дозволяє розгорнути обчислення на одному або декількох процесорах або графічних процесорах на робочому столі, сервері або мобільному пристрої без переписування коду. Розроблена корпорацією Google для роботи з тензорами, використовується для побудови нейронних мереж. При написанні програми на Python можна компілювати і запускати програму як на CPU, так і на GPU. Таким чином для запуску на GPU вам не доводиться писати на код C ++ або на рівні CUDA. На основі даної бібліотеки будуються більш високорівневі бібліотеки для роботи з нейронними мережами на рівні цілих шарів. Бібліотека Keras стала використовувати Tensorflow як основний бекенд для обчислень замість аналогічної бібліотеки Theano. Для роботи з зображеннями і відео (зі згортковими НМ), на NVIDIA використовується бібліотека cuDNN. Бібліотека використовує систему багаторівневих вузлів, яка дозволяє вам швидко налаштовувати, навчати і розгортати штучні НМ з великими наборами даних. Саме це дозволяє Google визначати предмети на фотографіях і розуміти вимовлені слова в додатках для розпізнавання усного мовлення. Є декілька навчених нейронних мереж, проте використання їх потребує імплементації нетривіального скрипту на Python. [3]

Apache MXNet — база Deep Learning, орієнтована на ефективність і гнучкість. Це дозволяє поєднувати символічне і імперативне програмування для збільшення ефективності і продуктивності. За своєю суттю MXNet містить динамічний планувальник залежностей, який «на льоту» автоматично розпаралелює як символічні, так і імперативні операції. У MXNET

передбачений інструмент caffe_converter, призначений для перетворення заздалегідь навчених моделей caffe у формат MXNET.

Theano - це розширення мови Python, що дозволяє ефективно визначати, оптимізувати, оцінювати і обчислювати математичні вирази, що містять багатовимірні масиви. Бібліотека надає базовий набір інструментів для конфігурації нейромереж і їх навчання. Найбільше визнання Theano отримала в задачах машинного навчання при вирішенні завдань оптимізації. Бібліотека дозволяє використовувати можливості GPU без зміни коду програми, що робить її незамінною при виконанні ресурсоємних завдань. Lasagne - головна платформа на основі Theano. У Lasagne дуже просто використовувати заздалегідь навчені моделі Caffe.

Можливості бібліотеки:

- тісна інтеграція з NumPy;
- можливості використання GPU;
- ефективне диференціювання змінних;
- динамічна генерація коду на C;
- розширені можливості юніт-тестування та самоперевірок;

NumPy – це бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами.

Математичні алгоритми, реалізовані на Python, часто працюють набагато повільніше тих же алгоритмів, реалізованих на компільованих мовах (наприклад, Фортран, Сі, Java). Бібліотека NumPy надає реалізації

обчислювальних алгоритмів (у вигляді функцій і операторів), призначені для роботи з багатовимірними масивами. В результаті будь-який алгоритм, який може бути виражений у вигляді послідовності операцій над масивами (матрицями) і реалізований з використанням NumPy, працює так само швидко, як еквівалентний код, що виконується в MATLAB.

SciPy – це відкрита бібліотека високоякісних наукових інструментів для мови програмування Python. SciPy містить модулі для оптимізації, інтегрування, спеціальних функцій, обробки сигналів, обробки зображень, генетичних алгоритмів, рішення звичайних диференціальних рівнянь та інших задач, які вирішуються в науці і при інженерній розробці. Дана бібліотека є аналогом MATLAB і Scilab. Для візуалізації при використанні SciPy часто застосовують бібліотеку Matplotlib, що є аналогом засобів виведення графіки MATLAB. В даний час SciPy поширюється під ліцензією BSD і його розробники спонсоруються Enthought.

Можливості:

- пошук мінімумів і максимумів функцій;
- обчислення інтегралів функцій;
- підтримка спеціальних функцій;
- обробка сигналів;
- обробка зображень;
- робота з генетичними алгоритмами;
- розв’язання звичайних диференціальних рівнянь;

Pandas – це бібліотека Python, яка є потужним інструментом для аналізу даних. Пакет дає можливість будувати зведені таблиці, виконувати групування і фільтрацію, надає зручний доступ до табличних даних, а при наявності пакета matplotlib дає можливість будувати графіки н отриманих наборах даних.

Основні можливості бібліотеки:

- Об’єкт DataFrame для маніпулювання індексованими масивами двовимірних даних
- Інструменти для обміну даними між структурами в пам'яті і файлами різних форматів
- Засоби поєднання даних і способи обробки відсутньої інформації
- Переформатування наборів даних, в тому числі створення зведених таблиць
- Зріз даних за значеннями індексу, розширені можливості індексування, вибірка з великих наборів даних
- Вставка і видалення стовпців даних
- Можливості групування дозволяють виконувати трьохетапну операції типу «поділ, зміна, об'єднання» (англ. *split-apply-combine*).
- Злиття і об'єднання наборів даних
- Ієрархічне індексування дозволяє працювати з даними високої розмірності в структурах меншої розмірності
- Робота з тимчасовими рядами: формування тимчасових періодів і зміна інтервалів

Keras – це бібліотека, яка дозволяє на високому рівні працювати з НМ, вона підтримує, як рекурентні, так і згорткові НМ, має в своєму складі реалізацію відомих архітектур НМ (наприклад, VGG16). Слої з даної бібліотеки

були інтергровані і стали доступні всередині бібліотеки Tensorflow. Існують готові функції для роботи з зображеннями та текстом. Інтегрована в Apache Spark за допомогою дистрибутива dist-keras.

. **Scikit-learn** – це бібліотека призначена для роботи з класичними алгоритмами машинного навчання. Одна з найпопулярніших бібліотек, яка підтримує багато контрольованих і не контрольованих алгоритмів навчання. Наприклад: лінійна і логістична регресії, дерева прийняття рішень, кластеризацію, k-means. Створена на основі бібліотек Python – NumPy і SciPy. В Scikit-learn додано набір алгоритмів для для розповсюджених задач машинного навчання і аналізу даних, враховуючи кластерізацію, регресію і класифікацію, виконує такі задачі, як перетворення даних і вибір функції. [6]

Після аналізу бібліотек за критеріями зручності розробки, підтримки нейромереж для обробки зображень – згорткових нейронних мереж та документації як основи подальших розробок та досліджень було обрано бібліотеку TensorFlow.

1.3. Класифікація об’єктів з використанням моделі YOLO v4.

Існує два підходи до розпізнавання об'єктів із використанням глибокого навчання:

- Навчання моделі з нуля. Щоб навчити глибоку мережу з нуля, необхідно збирати дуже великий набір даних і розробити архітектуру мережі, яка вивчатиме характеристики і будуватиме модель. Результати можуть бути вражаючими, але цей підхід вимагає великої кількості навчальних даних,

а також необхідно налаштовувати рівні та ваги згорткової нейронної мережі.

- Використання попередньо вивченої моделі глибокого навчання: більшість додатків глибокого навчання використовують підхід трансферного навчання – процес, який включає точне налаштування попередньо вивченої моделі. Спочатку береться існуюча мережа, така як AlexNet або GoogLeNet, і вводяться нові дані, що містять раніше невідомі класи. Цей метод вимагає менше часу і може забезпечити швидший результат, оскільки модель вже навчена на тисячах чи мільйонах зображень.

Глибоке навчання пропонує високий рівень точності, але потребує великої кількості даних для точних прогнозів.

У цій роботі було використано 2-й підхід, тобто: Використання попередньо вивченої моделі глибокого навчання. За основу було взято модель YOLO v4.

Модель YOLO v4 основана на згортковій нейронній мережі . Модель ділить зображення на області, а потім прогнозує граничні рамки та ймовірності для кожного регіону.

Більшість сучасних точних моделей вимагає багато графічних процесорів для навчання з великим розміром batch-пакетів. З одним графічним процесором навчання є дуже повільним і непрактичним. YOLO v4 [8] вирішує цю проблему, створюючи детекторні об'єкти, які можна обробити на одному графічному процесорі з меншим розміром batch-пакетів.

YOLO v4 є найкращою нейронною мережею для виявлення об'єктів - найточнішою нейронною мережею (55.8% AP) на датасеті Microsoft COCO серед усіх опублікованих нейронних мереж на даний момент.

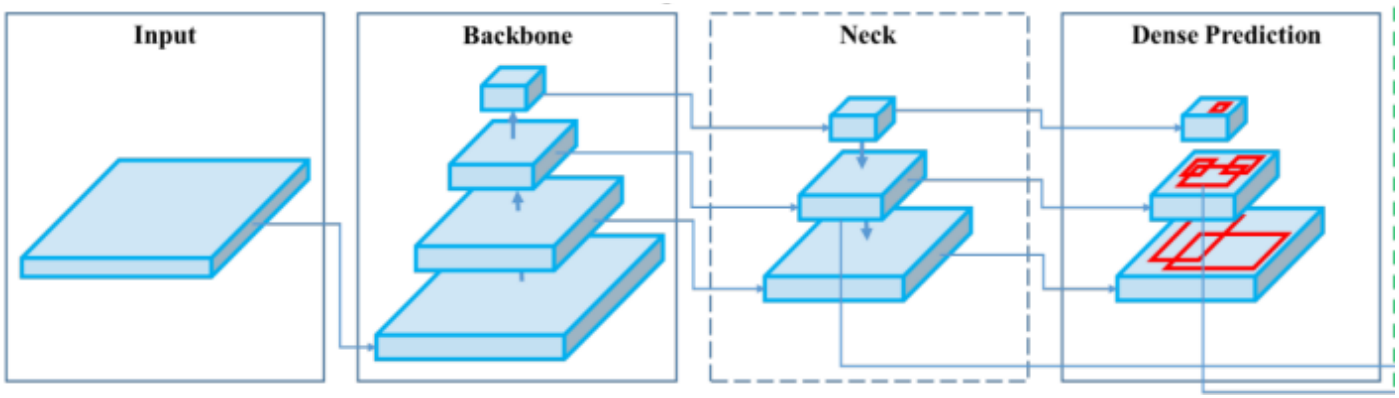

Рис 1.5. Архітектура мережі YOLO v4 [8]

На рисунку 1.5 представлена схема, яка показує архітектуру мережі YOLO v4.

Використання блоку Backbone [8] – це один із способів, за допомогою якого можна підвищити точність, спроектувати глибшу згорткову нейрону мережу і збільшити складність моделі. В архітектурі YOLOv4 використовується CSPDarknet53 як блок Backbone.

Основна мета блоку Neck – додати додаткові шари між блоком Backbone та блоком Head, щоб отримати «багатшу» просторову та семантичну інформацію. В архітектурі YOLOv4 використовуються концепція FPN , яка поступово замінюється PAN та SPP в якості блоку Neck.

У блоці Head відбувається той самий процес визначачення координат обєкта разом із оцінкою достовірності для класу. Мета полягає в тому, щоб розділити зображення на сітку, що складається з кількох частин , а потім для

кожної передбачити можливість наявності об'єкта. На виході виходить вектор з координатами обєктів та класами ймовірностей.

YOLO переосмислила завдання виявлення об'єктів у завдання регресії. Вона йде від пікселів зображення до координат bounding box та ймовірностей класів. Тим самим, єдина мережа сітки передбачає кілька bounding box і ймовірності класів для цих областей.

Зображення поділяється на сітку з рамками розміром S x S. Кожна область може містити кілька різних об'єктів для розпізнавання.

По-перше, кожна область відповідає за прогнозування кількості bounding box. Також кожна рамка прогнозує вартісне значення (confidence value) для кожної області, обмеженої bounding box. Іншими словами, це значення визначає можливість знаходження того чи іншого об'єкта в даній рамці. Тобто якщо якась область не має певного об'єкта, важливо, щоб довірче значення для цієї області було низьким.

Коли ми візуалізуємо все передбачення, ми отримуємо карту об’єктів і упорядкованих по довірчому значенню, рамки.

По-друге, кожна рамка відповідає за передбачення ймовірностей класів. Це не говорить про те, що якась область містить якийсь об'єкт, лише ймовірність знаходження об'єкта в цій області. Припустимо, якщо область передбачає автомобіль, це не гарантує, що автомобіль насправді присутній в ній. Це говорить лише про те, що якщо є об'єкт, то цей об'єкт швидше за все автомобіль.

1.4. Середовище виконання

Основним інструментом розробки програми для створення НМ було обрано інтерпретовану мову програмування Python. Ця мова програмування підтримує більшість бібліотек машинного навчання, надає можливість використання, оптимізації відкритого коду та велику кількість засобів для візуалізації даних.

Для дослідження даних було використано хмарний сервіс Google Colaboratory, який спрощує роботу в області машинного навчання. Слід зазначити, що НМ містить велику кількість параметрів, для обробки та зберігання яких необхідно мати значні обчислювальні ресурси.

Google Colaboratory дозволяє підключати графічний процесор (GPU), який суттєво прискорює обробку даних. Головними перевагами цього середовища є:

- Підтримка автоматично встановлених необхідних для роботи з НМ бібліотек Python;
- Надання безкоштовного застосування 12 гігабайт пам’яті на 12 годин;
- Створення необмеженої кількості робочих блокнотів та синхронізація зі своїми обліковими записами Google Drive та Github.

2. Розробка програмних засобів для класифікації об'єктів в просторі

2.1. Загальні вимоги та процес створення мережі для класифікації об’єктів

Необхідно було створити нейронну мережу, яка могла б класифікувати об’єкти, які знаходяться на зображенні.

Етапи створення нейронної мережі:

1. Інсталяція Python.
2. Додавання класифікатора (інструмент, який дозволяє методам машинного навчання розуміти до якого класу відноситься невідомий об’єкт.).
3. Додавання набору даних.
4. Навчання моделі (Відбувається аналіз зображень із набору за допомогою класифікатора. Програма створює спеціальні текстові файли з короткою інформацією про зображення. Весь процес навчання займає близько 6000 кроків.
5. Тестування (нейронна мережа перевіряє зображення на відповідність одній з міток і видає результат).

2.2. Підготовка даних для нейронної мережі

Найпершим кроком для навчання нейронної мережі – це створення набору даних.

Наскільки б не були хороші модель і метод навчання, якщо навчальна вибірка мала або не описує більшу частину випадків реального світу – досягти високої якості роботи буде складно.

Розмітка даних проводилася за допомогою інструмента labelImg.

Всього було розмічено близько 5000 світлин автівок.

Було обрато такі класи:

- Audi;
- Chevrolet;
- Toyota;
- Other.

До класу Other відносяться моделі автівок, які не належать першим трьом класам.

Оскільки labelImg використовує формат Pascal VOC і пише розмітку в XML-файлах, а YOLOv5 використовує TXT-файли зі строками в яких міститься клас об’єкта та координати рамки, потрібно перетворити розмітку до цьому формату.

2.4. Аугментація даних

Аугментація даних (можна перекласти як збільшення даних) – це методика створення додаткових навчаючих даних з вже наявних даних. Для досягнення хороших результатів глибинні мережі повинні навчатися на великому обсязі даних. Після цього, якщо вихідний навчальний набір містить обмежену кількість зображень, необхідно збільшити кількість зображень, щоб покращити результати моделі.

За рахунок такого моделювання деформацій можна легко досягти збільшення якості моделі та підвищити її стійкість до різних шумів у вхідних даних.

Для аугментації даних було використано інструмент roboflow [12].

Було використано такі види аугментації:

- Crop (обрізання світлини)
- Rotation (поворот світлини)
- Brightness (збільшення/зменшення яскравості)

За рахунок цього вдалось розширити набір даних до 15000 зображень.

2.5. Навчання нейронної мережі

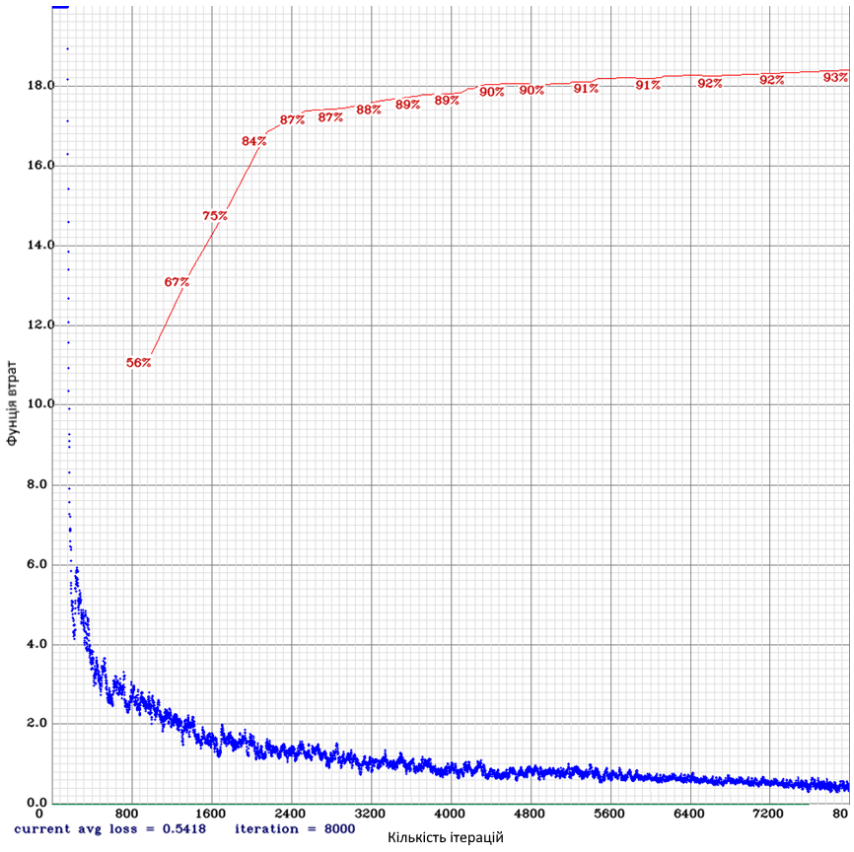

Рис 2.1. Графік значень функції втрат

На рисунку 1.5 зображено графік значень функції втрат.

З розмічених зображень 70% було обрано для навчальної вибірки, 20% для валідаційної і 10% для тестової.

Було проведено 8000 епох навчання та досягнуто точності близько 93% на валідаційній вибірці і 91% на тестовій вибірці.

2.6. Результати роботи створеного застосування

Дана нейронна мережа якісно вирішує поставлені задачі в даній роботі, а саме: класифікація об’єктів на зображеннях/відео.

Приклади розпізнавання моделі автівок на зображеннях :

Рис 2.2. Результати розпізнавання моделі Audi

Рис 2.3. Результати розпізнавання моделі Toyota

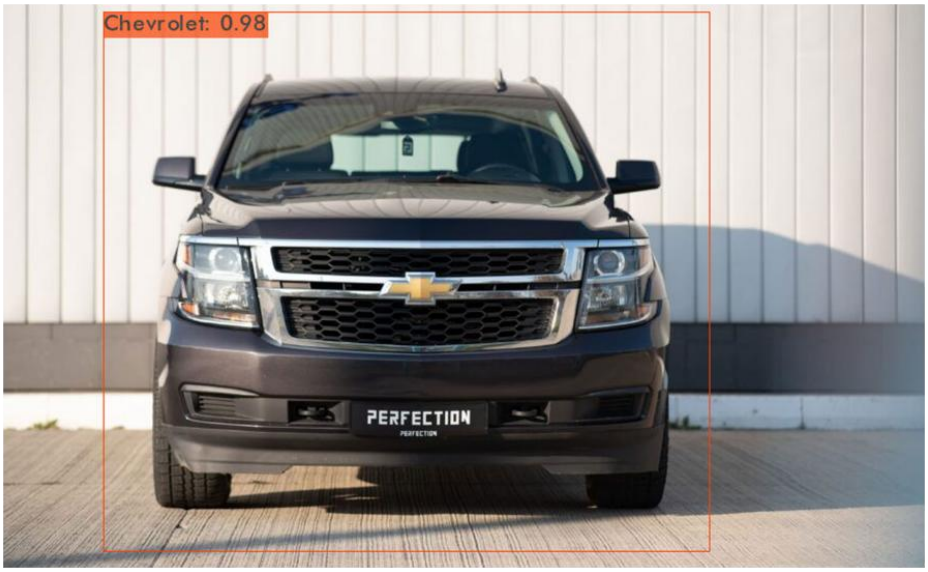

Рис 2.4. Результати розпізнавання моделі Chevrolet

Приклад розпізнавання моделей автівок на відео можна подивитись за посиланням [14].

Висновки

Проведено аналіз можливості застосування нейронних мереж для розв’язання задачі розпізнавання рухомих авто. Здійснено порівняльний аналіз бібліотек для навчання нейронних мереж. Було сформовано і обґрунтовано вибір моделі нейронної мережі

Реалізована модель для класифікації об'єктів з 4 класами на основі моделі Yolo v4. Дана нейронна мережа якісно вирішує поставлені задачі в даній роботі, а саме: класифікація об’єктів на зображеннях/відео.

Визначено такі класи для моделей автівок: Audi, Chevrolet, Toyota, Other. До класу Other відносяться автівки, які не відносяться до перших трьох класів.

Було проведено навчання мережі та досягнуто точності близько 93% на валідаційній вибірці і 91% на тестовій вибірці.

Результати тестування застосунку показали високу точність (не менше 91%) для визначення моделі автівок на світлинах та відео.

Список використаних джерел

1. Joel Grus, «Data Science from Scratch First Principles with Python», O'Reilly Media, Inc., 2019, pp.
2. Curiosity-Driven Learning [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/curiosity-driven-learning-made-easy-part-i-d3e5a2263359 (дата звернення: 25.04.2022).
3. What is a Convolutional Neural Network? [Електронний ресурс]. – Режим доступу: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html?s_tid=srchtitle (дата звернення: 25.04.2022).
4. С.Д. Погорілий, М.В. Чечула. Агрегація та аналіз графічних даних в розподіленій мережі мобільних пристроїв. Реєстрація, зберігання і обробка даних. Том 21, № 2, 2019, с.с. 21-33.
5. Офіційний відкритий репозиторій TensorFlow [Електронний ресурс] – Режим доступу: https://github.com/tensorflow/tensorflow (дата звернення: 25.04.2022
6. Порівняння бібліотек глибокого навчання [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/intel/blog/254747 (дата звернення: 25.04.2022).
7. Open Images Dataset V6 [Електронний ресурс]. – Режим доступу: https://storage.googleapis.com/openimages/web/index.html (дата звернення: 25.04.2022).
8. YOLO v4 [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/yolo-v4-optimal-speed-accuracy-for-object-detection-79896ed47b50 (дата звернення: 25.04.2022).
9. YOLOv4 Object Detection [Електронний ресурс]. – Режим доступу: https://machinelearningknowledge.ai/yolov4-object-detection-tutorial-with-image-and-video/ (дата звернення: 25.04.2022).

10. Train Custom YOLOv4 Model for Object Detection [Електронний ресурс]. – Режим доступу: https://machinelearningknowledge.ai/train-custom-yolov4-model-for-object-detection-in-google-colab/ (дата звернення: 25.04.2022).
11. YOLOv4: Optimal Speed and Accuracy of Object Detection [Електронний ресурс]. – Режим доступу: https://arxiv.org/pdf/2004.10934.pdf (дата звернення: 25.04.2022).
12. Інструмент для аугментації світлин Roboflow [Електронний ресурс]. – Режим доступу: https://roboflow.com/ (дата звернення: 25.04.2022).
13. YOLOv4 : A Machine Learning Model to Detect the Position and Type of an Object [Електронний ресурс]. – Режим доступу: https://medium.com/axinc-ai/yolov4-a-machine-learning-model-to-detect-the-position-and-type-of-an-object-4f108ed0507b (дата звернення: 25.04.2022).
14. Розпізнавання моделей автівок на відео [Електронний ресурс]. – Режим доступу: https://drive.google.com/file/d/1mlkb4-MlxModkx4Gh8iYlUoIpxam7tyc/view?usp=sharing (дата звернення: 25.04.2022).
15. A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks [Електронний ресурс]. – Режим доступу: https://builtin.com/data-science/recurrent-neural-networks-and-lstm (дата звернення: 25.04.2022).

Додаток 1. Код застосування мовою Python

```
#імпорт бібліотек
from absl import app, flags, logging
from absl.flags import FLAGS
import os
import shutil
import tensorflow as tf
from core.yolov4 import YOLO, decode, compute_loss,
decode_train
from core.dataset import Dataset
from core.config import cfg
import numpy as np
import os
from lxml import etree
import cv2
from glob import glob
import re
from core import utils
from core.utils import freeze_all, unfreeze_all
from google.colab import drive
drive.mount('/content/gdrive') #підключаємо гугл диск
```

```
!ln -s /content/gdrive/My\ Drive/ /mydrive

!git clone https://github.com/AlexeyAB/darknet

%cd darknet

!sed -i 's/OPENCV=0/OPENCV=1/' Makefile

!sed -i 's/GPU=0/GPU=1/' Makefile

!sed -i 's/CUDNN=0/CUDNN=1/' Makefile

!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

!make

!wget

#завантажуємо початкові ваги

https://github.com/AlexeyAB/darknet/releases/download/darknet_
yolo_v3_optimal/yolov4.conv.137

#завантажуємо набір даних з гугл диску

!cp /mydrive/yolo_cars/Cars.v7i.darknet.zip ../

!unzip ../C def add_txt_files_with_names_of_images(split =
"train"):

  image_files = []

  os.chdir(os.path.join("data", split))

  for filename in os.listdir(os.getcwd()):

      if filename.endswith(".jpg"):

          image_files.append(f"data/{split}/" + filename)
```

```
  os.chdir("..")
  with open(f"{split}.txt", "w") as outfile:
      for image in image_files:
          outfile.write(image)
          outfile.write("\n")
      outfile.close()
  os.chdir("..")ars.v7i.darknet.zip -d data/
for split in ["train", "valid"]:
  add_txt_files_with_names_of_images(split)
!ls data/
#завантаження конфінураційних файлів з гугл диску
!cp cfg/yolov4-custom.cfg /mydrive/yolo_cars/yolov4-obj.cfg
!cp /mydrive/yolo_cars/yolov4-obj.cfg cfg/yolov4-obj.cfg
!cp /mydrive/yolo_cars/obj.names ./data
!cp /mydrive/yolo_cars/obj.data  ./data
#навчання мережі
!./darknet detector train data/obj.data cfg/yolov4-obj.cfg
yolov4.conv.137 -dont_show –map
%cd cfg
!sed -i 's/batch=64/batch=1/' yolov4-obj.cfg
!sed -i 's/subdivisions=16/subdivisions=1/' yolov4-obj.cfg
```

```
!sed -i 's/width=416/width=608/' yolov4-obj.cfg
!sed -i 's/height=416/height=608/' yolov4-obj.cfg
%cd .
#визначення класів
classes = {Audi: 0, Toyota: 1, Chevrolet: 2, Other: 4}
def extract_txt(img_path, xml_path, txt_path):
    img = cv2.imread(img_path)
    W = img.shape[1]
    H = img.shape[0]
    tree = etree.parse(xml_path)

    root = tree.getroot()
    with open(txt_path, 'w') as f:
        for child in root:
            if child.tag == 'object':
                for object_child in child:
                    coords = []
                    if object_child.tag == 'name':

f.write(str(classes.get(object_child.text))+' ')
                    if object_child.tag == 'bndbox':
```

```
                        for bbox_child in object_child:
#перетворення міток зображеннь до потрібного формату
coords.append(int(bbox_child.text))
     xmin = coords[0]
     xmax = coords[2]
     ymin = coords[1]
     ymax = coords[3]
     w = (xmax - xmin) / W
     h = (ymax - ymin) / H
     xc = (xmin + (xmax - xmin) / 2) / W
     yc = (ymin + (ymax - ymin) / 2) / H
f.write(str(xc)+' '+str(yc)+' '+str(w)+' '+str(h)+'\n')
xmls = sorted(glob(CAR_BRANDS_PATH + '*.xml'))
for xml_path in xmls:
    img_path = re.findall('[A-Za-z0-9_/]+',
xml_path)[0]+'.jpg'
    txt_path = re.findall('[A-Za-z0-9_/]+',
img_path)[0]+'.txt'
    extract_txt(img_path, xml_path, txt_path)
def imShow(path):
  %matplotlib inline
```

```
  image = cv2.imread(path)
  height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height),
interpolation = cv2.INTER_CUBIC)
  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()
def upload():
uploaded = files.upload()
  for name, data in uploaded.items():
    with open(name, 'wb') as f:
      f.write(data)
      print ('saved file', name)
def download(path):
  files.download(path)
!./darknet detector demo data/obj.data cfg/yolov4-obj.cfg
#визначаємо кращі ваги навченої мережі
/mydrive/yolo_cars/backup/yolov4-obj_best.weights -dont_show
```

```
#розпізнавання моделей автівок на відео з гугл диску

/mydrive/yolo_cars/tr10.mp4 -i 0 -out_filename
/mydrive/yolo_cars/result.avi -out result.json -thresh 0.5

# позначеємо bounding box для об’єктів з ймовірністю
належності до класу більше 0.5

!./darknet detector test data/obj.data cfg/yolov4-obj.cfg
/mydrive/yolo_cars/backup/yolov4-obj_last.weights
/mydrive/yolo_cars/car2.jpg -thresh 0.4

imShow('predictions.jpg')

# змінюємо makefile щоб увімкнути GPU та OPENCV

!sed -i 's/OPENCV=0/OPENCV=1/' Makefile

!sed -i 's/GPU=0/GPU=1/' Makefile

!sed -i 's/CUDNN=0/CUDNN=1/' Makefile

!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

!sed -i 's/LIBSO=0/LIBSO=1/' Makefile

network, class_names, class_colors = load_network("cfg/yolov4-
obj.cfg", "data/obj.data",
"/mydrive/YOLOv4Files/backup/yolov4-obj_final.weights")

width = network_width(network)

height = network_height(network)

# допоміжна функція для запуску виявлення класів на зображенні

def drknet_help(img, width, height):
```

```
  darknet_image = make_image(width, height, 3)
  img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img_resized = cv2.resize(img_rgb, (width, height),
interpolation=cv2.INTER_LINEAR)

  # перетворення обмежувальних прямокутників до потрібного
розміру
img_height, img_width, _ = img.shape
  width_ratio = img_width/width
  height_ratio = img_height/height
# запускаємо модель для виявлення автівок в режимі реального
часу
  copy_image_from_bytes(darknet_image, img_resized.tobytes())
  detections = detect_image(network, class_names,
darknet_image)
  free_image(darknet_image)
  return detections, width_ratio, height_ratio
video_stream()
# мітки для відео
label_html = 'Capturing...'
# запускаємо відео в режимі рельно часу
```

```
detections, width_ratio, height_ratio = darknet_helper(frame,
width, height)
    for label, confidence, bbox in detections:
      left, top, right, bottom = bbox2points(bbox)
      left, top, right, bottom = int(left * width_ratio),
int(top * height_ratio), int(right * width_ratio), int(bottom
* height_ratio)
      bbox_array = cv2.rectangle(bbox_array, (left, top),
(right, bottom), class_colors[label], 2)
      bbox_array = cv2.putText(bbox_array, "{}
[{:.2f}]".format(label, float(confidence)),
                          (left, top - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                          class_colors[label], 2)

    bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0
).astype(int) * 255
    # convert overlay of bbox into bytes
    bbox_bytes = bbox_to_bytes(bbox_array)
    # update bbox so next frame gets new overlay
    bbox = bbox_bytes
```