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

ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗПІЗНАВАННЯ ЦИФР

Виконала: студентка групи ПМі-43с
спеціальності
122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Іванкова А.О.
(підпис) (прізвище та ініціали)

Керівник Позднякова І.В.
(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 Притула М.М.

"31 "серпня 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ У РОБОТУ СТУДЕНТУ

Іванковій Анастасії Олегівній
(прізвище, ім'я, по батькові)

1. Тема роботи «Використання машинного навчання для розпізнавання цифр»

керівник роботи Позднякова Інна Володимирівна, канд. фіз.-мат. 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 **13.06.2023р.**

3. Вихідні дані до роботи документація по мові програмування Python, Keras, MatPlotLib, TensorFlow, Seaborn, Yaml, Scikit-Learn, Random, CV2, Pandas PyTorch, та NumPy, інтернет ресурси, відкриті набори даних з цифровим зображенням HandWritten_Character та standard OCR dataset, веб-середовище розробки Kaggle

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) Огляд існуючих досліджень, перегляд наборів даних, створення нейронної мережі для ідентифікації цифр на зображенні, дослідження тонкощів використання машинного навчання для розробки нейронних мереж.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) ілюстрації алгоритму YOLO, MNIST dataset, елементи модифікованого датасету, графіки втрат під час навчання нейромережі, графіки результатів навчання нейронної мережі, матриця похибок моделі у ході навчання моделі, отримані результати навчання, тестування та використання навченої моделі машинного навчання

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.	Аналіз та постановка задачі	10.10.2022	
2.	Вивчення літератури по обраній темі	15.11.2022	
3.	Огляд існуючих програмних рішень задачі	11.12.2022	
4.	Аналіз алгоритмів розпізнавання об’єктів	10.01.2023	
5.	Огляд веб-середовища Kaggle	18.02.2023	
6.	Проектування модифікації датасету	31.03.2023	
7.	Розробка процесу навчання нейромережі	10.04.2023	
8.	Розробка процесу тренування нейромережі	20.04.2023	
9.	Оформлення дипломної роботи	31.05.2023	

Студент _____ Іванкова А.О
(підпис) (прізвище та ініціали)

Керівник роботи _____ Позднякова І.В._
(підпис) (прізвище та ініціали)

РЕФЕРАТ

У даній бакалаврській дипломній роботі було досліджено тему «Використання машинного навчання для розпізнавання цифр».

Мета роботи – дослідження алгоритмів розпізнавання об’єктів на зображеннях та розробка застосунку, який із допомогою обраного алгоритму здатен розпізнавати цифри та зберігати отримані результати.

У ході роботи було досліджено програмні рішення та технології, які застосовуються для ідентифікації об’єктів на зображеннях. Особливу увагу було приділено алгоритму YOLO (You Only Look Once), його особливостям застосування, перевагам та недолікам.

Результат роботи – протестована та коректно навчена модель машинного навчання для задачі розпізнавання цифр, опис процесу модифікації датасету на базі двох різних наборів даних, опис результатів навчання та тренування нейронної мережі.

Для розробки застосунку було використано мову програмування Python та веб-платформу Kaggle.

Обсяг дипломної роботи 38 сторінок, 31 рисунок, 13 використаних джерел.

Ключові слова: глибинне навчання, згорткова нейронна мережа, набір даних, датасет, розпізнавання об’єктів, ідентифікація цифр, Kaggle, Keras, TensorFlow, YOLO, MNIST.

ЗМІСТ

ВСТУП 6
1 ПОСТАНОВКА ЗАДАЧІ 8
2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ 9
2.1 ЛОГІСТИЧНА РЕГРЕСІЯ 9
2.2 K-MEANS КЛАСТЕРИЗАЦІЯ 10
2.3 ДЕРЕВА РІШЕНЬ 10
2.4 МЕТОДИ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ. 11
3 АНАЛІЗ АЛГОРИТМІВ РОЗПІЗНАННЯ ОБ'ЄКТІВ 12
3.1 ЗАДАЧА КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ 12
3.2 ЗГОРТКОВА НЕЙРОННА МЕРЕЖА 14
3.3 Region-based Convolutional Neural Networks (R-CNN) 16
3.5. YOLO (You Only Look Once) 16
4 РЕАЛІЗАЦІЯ АЛГОРИТМУ 20
4.1 KAGGLE 20
4.2 DATASETS 21
4.3 МОДИФІКАЦІЯ ДАТАСЕТУ 23
4.4 НАВЧАННЯ НЕЙРОМЕРЕЖІ 27
4.5 ТЕСТУВАННЯ МОДЕЛІ 30
РЕЗУЛЬТАТИ 31
ВИСНОВКИ 36
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 37

ВСТУП

В еру цифрових технологій, жодна сфера людської життєдіяльності не обходиться без технічного, а зокрема – цифрового супроводу. Системи, які займаються зчитуванням, ідентифікації даних, обробки та використанням цифрової інформації активно впроваджуються у приватні підприємства та державні структури.

Актуальність обраної мною теми дослідження підтверджується великою кількістю досліджень у сфері науки про дані та штучного інтелекту, а сама розробка алгоритмів із використанням машинного навчання для розпізнавання цифр є однією із найбільш уживаних у галузі машинного навчання - комп'ютерному зорі.

Нам важливо, аби наша праця, пошук, були зручними та зрозумілими. Таким чином, будь яка робота з інформацією піддається автоматизації. Будь-який пошук, обробка та збереження інформації не обходиться без набору алгоритмів, які забезпечують весь процес. Ми шукаємо, описуємо та навчаємо алгоритми діяти так, як нам буде вигідно. Останнім питанням займаються спеціалісти із систем штучного інтелекту та, зокрема, з нейронних мереж. За допомогою алгоритмів ми можемо знаходити об'єкти, людей, знаходити відмінності між об'єктами та їхню подібність.

Розпізнавання об'єктів із застосуванням машинного навчання використовується в різних областях, включаючи розпізнавання символів на зображеннях, автоматизоване розпізнавання тексту та рукописних заміток, розпізнавання номерних знаків на автомобілях та інших.

Метою моєї дипломної роботи є дослідження та розробка застосунку із використанням алгоритму машинного навчання для розпізнавання цифр на зображеннях. Робота складається із декількох етапів: вивчення теоретичних аспектів комп'ютерного зору та розпізнавання об'єктів, підготовка даних, розробка та налаштування нейронної мережі, навчання нейро-мережі, тестування та оцінка результатів.

Одним із головних завдань у розробці мого застосунку є тестування розробленої моделі машинного навчання на зображеннях із друкованими, писаними числами та числах із шумом. Адже у чистому вигляді цифри ідентифікувати дуже просто, а на зашумлених даних – у рази важче, а деколи навіть неможливо. На таких примірниках нейромережі роблять велику кількість помилок.

Тож, використання машинного навчання для розпізнавання цифр має важливе значення для покращення різноманітних процесів, які потребують автоматизованого розпізнавання цифрової інформації.

Для розробки застосунку мною було використано мову програмування Python та веб-платформу Kaggle для розробки та змагань із машинного навчання.

1 ПОСТАНОВКА ЗАДАЧІ

Задача знаходження цифр на зображеннях зводиться до пошуку та ідентифікації об’єктів на невеличких масштабованих частинках вихідного зображення. (рис. 1.1)

Для організації задачі можна використовувати такі способи, як логістична регресія, дерева рішень, або нейронні мережі. Для вирішення задачі було обрано нейромережі.

З огляду проблематики задачі та її поширення на різноманітні зразки символів, включаючи друковані та рукописні цифри, зокрема на зашумлених зображеннях, моєю задачею було створення алгоритму ідентифікації вищеперерахованих об’єктів із застосуванням моделі машинного навчання YOLOv5.

Для забезпечення якісного та різноманітного навчання алгоритму, головним із завдань був вибір бази даних із наведеними типами цифр та створення шумів на отриманих фото. У ході виконання дипломної роботи було модифіковано два бібліотечних набори даних в один.

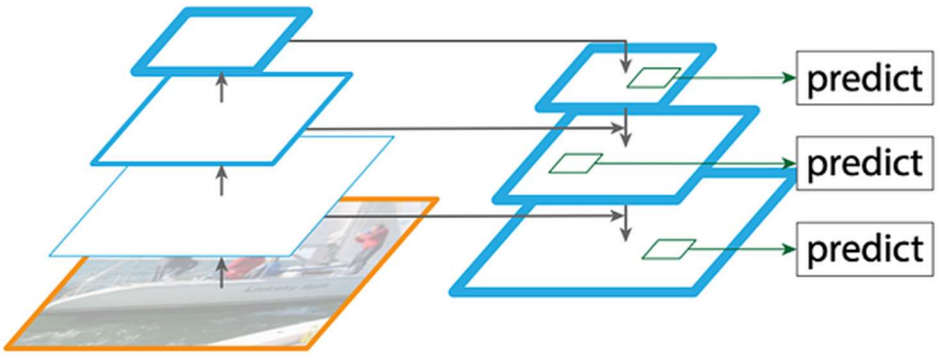


Рис. 1.1 Знаходження об’єктів на зображенні

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ

Задача ідентифікації цифр на зображеннях – це базова задача у машинному навчанні. Використовується у всіх алгоритмах, що займаються пошуком символів на зображеннях.

Існує безліч методів для виявлення об'єктів, вибір певних методів напряму залежить від конкретного застосування та наявних технічних ресурсів. Часто використовують наступні методи для задач виявлення об'єктів: логістична регресія, дерева рішень, методи на основі глибокого навчання.

2.1 ЛОГІСТИЧНА РЕГРЕСІЯ

Логістична регресія є статистичним алгоритмом, який використовується для прогнозування ймовірності належності об'єкта до певного класу. Для моделювання залежності між вхідними ознаками і ймовірністю належності до класу, регресія використовує логістичну функцію.

Логістична регресія обчислює лінійну комбінацію вхідних ознак з вагами моделі. Опісля чого за допомогою логістичної функції, вхідні ознаки перетворюються у лінійну комбінацію ймовірності належності до класу за значенням, що варіюється від 0 до 1. Після цього ваги моделі оптимізуються шляхом максимізації ймовірності правильної класифікації навчальних прикладів, наприклад методом градієнтного спуску. Для прийняття рішення про належність до класу, встановлюється поріг ймовірності, відповідно, якщо ймовірність перевищує поріг – об'єкт класифікується як належний до класу, в іншому випадку – як не належний.

Перевагою алгоритму є простота в реалізації та інтерпретації. Добре справляється із задачами бінарної класифікації.

Основним недоліком даного рішення є проблема локальних мінімумів, при потраплянні у які алгоритм зациклюється і перестає прогнозувати коректні ймовірності.

2.2 K-MEANS КЛАСТЕРИЗАЦІЯ

Це алгоритм машинного навчання, який забезпечує навчання мережі без учителя, використовується для групування точок даних в кластери на основі їх схожості. Алгоритм розділює дані на K відмінних кластерів, де K - це попередньо визначена кількість, встановлена користувачем. Широке застосування отримав у сфері дослідження даних, сегментації клієнтів, стиснення зображень тобто у задачах виявлення вбудованих патернів даних.

Алгоритм K-means кластеризації здатний ефективно обробляти великі обсяги даних, також він відносно простий у розумінні та реалізації. Основною перевагою є простий підхід до кластеризації даних без необхідності приєднання міток до навчальних даних.

Проте головним недоліком є чутливість до початкового розташування центроїд, що, як наслідок можуть призвести до різних призначень кластерів та, відповідно, різних результатів.

2.3 ДЕРЕВА РІШЕНЬ

Даний алгоритм є достатньо популярним у сфері машинного навчання, використовується для задач класифікації та регресії. Дерева рішень утворюють структуру, схожу на блок-схему, у якій кожний внутрішній вузол відповідає певній ознаці, кожна гілка слугує правилом прийняття рішення, а кожен листовий вузол відображає результат або мітку класу.

Дерева рішень мають декілька переваг, включаючи інтерпретовність, оскільки отримане дерево можна легко візуалізувати та зрозуміти. Вони можуть обробляти як категоріальні, так і числові дані, враховують втрату значень у наборах даних, а також нелінійні залежності між ознаками.

Проте дерева рішень страждають від перенавчання, у випадках, коли модель утворює занадто складну схему, як наслідок нові дані і погано узагальнюються. Також недоліком є нестабільність через чутливість до змін у тренувальній вибірці, що може сильно змінити структуру дерева і втратити відповідність до задачі.

2.4 МЕТОДИ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ.

Рішення на основі глибокого навчання, зокрема згорткові нейронні мережі (Convolutional Neural Networks, CNN), зробили прорив у задачах із виявлення об'єктів.

Моделі, такі як Single Shot MultiBox Detector (SSD), You Only Look Once (YOLO) та Region-based CNNs (R-CNN), досягли високої ефективності та точності. Дані методи побудовані на основі навчання глибоких нейронних мереж на великих попередньо розмічених наборах даних, дана умова необхідна задля вивчення ознак для виявлення об'єктів алгоритмом.

Реалізація нейромереж глибокого навчання можуть бути реалізовані на базі OpenCV, TensorFlow (містить попередньо навчені згорткові нейронні мережі для певних задач), PyTorch (надає широкий спектр інструментів для побудови та тренування нейронних мереж), scikit-learn (містить різні класифікатори, для виконання задач класифікації, такі як метод опорних векторів-SVM), Keras (високорівневий API для побудови і тренування нейронних мереж), Microsoft Cognitive Services (колекція хмарних API та сервісів для використання функції комп’ютерного зору), Fastai (високорівнева бібліотека машинного навчання), Caffe (фреймворк, розроблений для швидкого розпізнавання об'єктів), Amazon Rekognition (хмарний сервіс, який надає API для розпізнавання зображень), YOLO (популярний алгоритмом розпізнавання об'єктів, також застосовується для ідентифікації об’єктів у режимі реального часу) та інші.

Основними недоліками моделей глибокого навчання є високі вимоги до обчислювальних ресурсів, адже для якісного навчання нейромережі потребують потужних процесорів та вільної оперативної пам’яті, необхідність великого набору даних для навчання. І, зокрема, нейронні мережі – є складними моделями з великою кількістю параметрів, що утворює проблеми у їх розумінні та розробці.

Проте – це найпотужніші інструменти на даний час, які отримали широке застосування у багатьох сферах, і які здатні самонавчатися у ході використання.

3 АНАЛІЗ АЛГОРИТМІВ РОЗПІЗНАННЯ ОБ’ЄКТІВ

3.1 ЗАДАЧА КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

Штучним інтелектом (ШІ) називають здатність цифрового комп’ютера або керованого комп’ютером робота, виконувати ряд завдань, що зазвичай асоціюють із розумними істотами. Також даний термін можуть застосовувати до проектів розробки систем наділених характерними для людей інтелектуальними процесами, такими як: здатність міркувати, відкривати сенс, узагальнювати чи вчитися на минулому досвіді.

Однією з найуживаніших на теперішній час галузей штучного інтелекту є – комп’ютерний зір. Computer Vision (CV) – це наука про комп’ютери та програмні системи, тобто алгоритми, які мають на меті розпізнавати об’єкти та розуміти вхідні зображення. В основі систем комп’ютерного зору зазвичай лежать алгоритми на базі машинного навчання, з допомогою яких системи отримують здатність навчатися відрізняти між собою об'єкти, виокремлювати шаблони та розкривати закономірності.

CV складається з різних галузей, а саме: розпізнавання зображень, виявлення об’єктів на зображеннях, створення зображень, робота із супер-роздільною здатністю зображень, тощо.

Розпізнавання об’єктів, мабуть, є найглибшим аспектом комп’ютерного зору через велику кількість випадків практичного використання, включаючи задачі виявлення обличь/транспортних засобів, підрахунку пішоходів, веб-зображень, систем безпеки та автомобілів без водіїв, загальна реєстрація зображень, побудова панорам, пошук схожих зображень, класифікація і категоризація зображень.

Людина навчається розпізнавати образи в ході знайомства зі навколишнім середовищем, як-от дитина запам’ятовує, чим відрізняється кішка від собаки. А комп’ютерний інтелект «мислить» по-іншому: задля навчання алгоритму, йому необхідно «згодувати» датасет із наперед розміченими даними, в якому наочно буде показано відмінності одного об'єкта від іншого.

Розпізнавання об’єктів — це форма комп’ютерного зору, в якій модель навчається ідентифікації і знаходження розташування одного або кількох класів об’єктів на зображенні.

Розпізнавання об’єктів складається з двох компонентів:

1. Класифікація об’єктів – отримання міток класу кожного об’єкта, виявленого на зображенні. Тут ми дізнаємося що саме і в якій кількості містить зображення.
2. Локалізація об’єкта – знаходження розташування кожного об’єкта на зображенні, зазначене як координати обмежувальної рамки, що охоплює об’єкт.

Класифікація об’єктів – є фундаментальним завданням, яке намагається осягнути ціле зображення в цілому.

Мета – класифікувати об’єкт, присвоївши йому певну мітку. Тож, розпізнавання об’єктів включає завдання як класифікації, так і локалізації, і використовується для аналізу більш реалістичних випадків, коли на зображенні може існувати кілька об’єктів. Прикладом слугує алгоритм пошуку літаку на вхідному зображенні, де пошукових об’єктів є – декілька. (рис. 3.1.1).

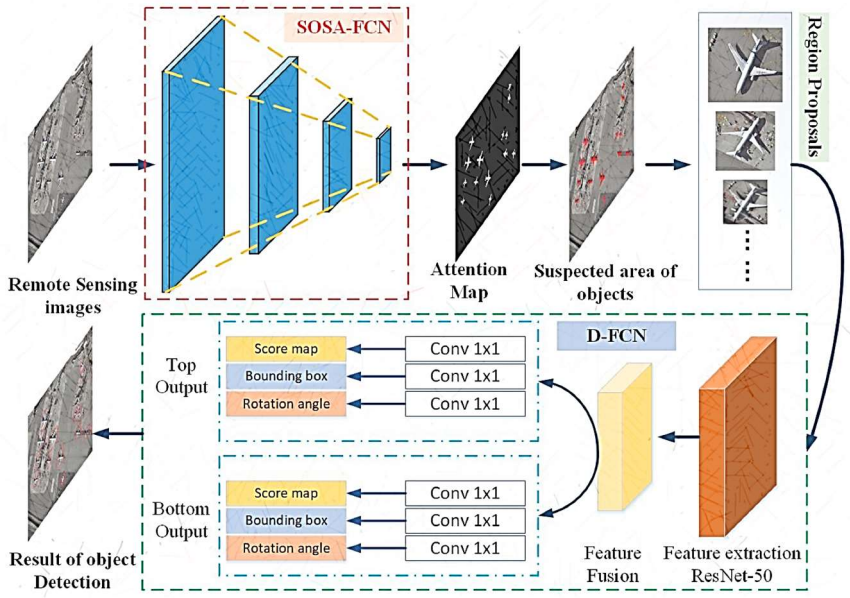


Рис. 3.1.1 Класифікація зображень

Ранні реалізації моделей виявлення об’єктів передбачали собою використання класичних алгоритмів з популярної бібліотеки комп’ютерного зору, OpenCV. Проте дані класичні алгоритми так і не змогли досягти достатньої продуктивності для роботи в різних умовах.

Прорив і швидке впровадження глибокого навчання в 2012 році створили сучасні та високоточні алгоритми і методи виявлення об’єктів, такі як Region-based Convolutional Neural Networks (R-CNN), Fast-RCNN, Faster-RCNN, RetinaNet і швидкі, але високоточні, такі як YOLO (You Only Look Once), Single Shot Detector (SSD).

Сучасні методи можна розділити на два основних типи: одно-етапні та двоетапні. Одно-етапні методи надають пріоритет швидкості висновку, а приклади моделей включають, і RetinaNet. Двоступеневі методи визначають пріоритет точності виявлення, і приклади моделей включають Fast Region-Based Convolutional Network(Fast R-CNN), Faster Region-Based Convolutional Network (Faster R-CNN), Histogram of Oriented Gradients (HOG), Region-based Fully Convolutional Network (R-FCN), Spatial Pyramid Pooling (SPP-net).

3.2 ЗГОРТКОВА НЕЙРОННА МЕРЕЖА

Згорткова нейронна мережа (Convolutional Neural Network, CNN), є одною з передових нейронних мереж, за використанням у галузі комп'ютерного зору та обробки зображень. CNN спеціалізується на обробці даних, які мають просторову архітектуру, наприклад зображення.

Згорткова нейронна мережа складається із шарів згортки (Convolutional Layers), що використовують ядра для виконання операції згортки над вхідним зображенням. Завдяки даним шарам мережа здатна автоматично виявляти локальні особливості зображень.

Наступною складовою – є приховані шари (Pooling Layers), які зменшують розмір виходу попереднього шару, зберігаючи лише найважливіші ознаки. Ця

особливість допомагає зменшити кількість параметрів та обчислювальний обсяг мережі.

Повнозв'язані шари (Fully Connected Layers): Ці шари забезпечують зв'язок між виходами з попередніх шарів та виходом мережі. Вони використовуються для класифікації або регресії на основі отриманих ознак.

До переваг згорткових нейронних мереж варто віднести здатність автоматично вивчати репрезентативні ознаки зображення, зменшення кількості параметрів за рахунок спільного використання шарами фільтрів, а також універсальність згорткових операцій, що дозволяє працювати зі зображеннями різних розмірів.

Варто зауважити те, що окрім позитивних критеріїв, згорткова нейронна мережа має негативні характеристики, до яких відноситься чутливість до змін в навчальній вибірці. Згідно з чим, навіть малі помилки у вибірці чи недостатній обсяг вхідних навчальних даних, впливає на результат навчання нейромережі, яка зовсім не буде підходити для поставленої першочергово задачі.

Загалом, неможливо передбачити хід навчання нейронних мереж, а самі темпи навчання різних нейронних мереж можуть відрізнятися, у залежності від типу нейромережі та обладнання, на якому вона працюватиме та навчатиметься. Можна сказати, що процес навчання – рандомний, адже деколи мережа може натрапити на локальні мінімуми у розв’язках і зациклитись. Таким чином, ми не отримаємо успішного результату.

На сьогодні, існує чимало способів покращення роботи нейромереж. Основними є вибір потужних систем для обчислень та збільшення якості побудованої системи для навчання нейромереж. Наприклад, використання найновіших комп’ютерів (кластери) із потужними процесорами та відеокартами, чи використання хмарних API, що дозволить зекономити чимало ресурсів, зокрема час. А покращити якість навчання ми можемо за рахунок збільшення розміру

навчальної та тренувальної вибірок, що дозволить забезпечити широкий спектр розпізнавання об’єктів.

3.3 Region-based Convolutional Neural Networks (R-CNN)

У 2014 році група дослідників з Каліфорнійського університету в Берклі розробила глибоку згорткову мережу під назвою R-CNN (регіональна згорткова нейронна мережа). Вона може виявити 80 різних типів об’єктів на зображеннях.

Основною перевагою використання архітектури R-CNN є те, що вона дозволяє отримувати переваги від двовимірної структури вхідних даних, якими і є зображення. І виходячи з отриманих даних вже проводити чітку класифікацію та локалізацію об’єктів зображень.

Великим мінусом, або навіть необхідністю для створення та використання машинного навчання – є потужні процесори та відеокарта, інакше наше навчання нейромережі буде затягуватися надовго, а деколи навіть і зовсім буде неосяжним.

3.5. YOLO (You Only Look Once)

Алгоритм YOLO (You Only Look Once) був створений Джозефом Редмоном і Алі Фархаді. Оригінальна версія YOLO, відома як YOLOv1, була представлена Джозефом Редмоном спільно з його науковим керівником Алі Фархаді в 2016 році.

Вони розробили YOLO як алгоритм реального часу для виявлення та класифікації об'єктів на зображеннях і відео. Мотивацією створення YOLO було вирішення обмежень попередніх підходів до виявлення об'єктів, які часто потребували великих обчислювальних витрат та кількох проходів по зображенню.

Модель YOLO здатна виявляти багато об'єктів на зображенні одночасно без необхідності виконання додаткових обчислень для кожного окремого об'єкту, що виокремлює її з-поміж інших алгоритмів.

Зокрема YOLO пропонує новий підхід до розв’язання задачі ідентифікації об'єктів, розглядаючи її як задачу регресії, це дозволяє здійснювати прогнози для класів об'єктів, присвоюючи їм відповідні мітки класів та виділення координат

обмежувальних рамок об'єктів безпосередньо за один прохід нейронної мережі. Це значно прискорило роботу YOLO порівняно з іншими методами виявлення об'єктів на той час. Також, основною перевагою моделі YOLO є те, що вона добре працює з об'єктами різних розмірів та форм, включаючи об'єкти зі змінними пропорціями, адже алгоритм підлаштовується і змінює розмір вхідного зображення у ході роботи, під необхідні критерії.

До недоліків варто віднести складну структуру прихованих шарів, через що модель може втрачати пильність до дрібних деталей на зображеннях, схильність до помилок у детекції перекритих одних об'єктів іншим, через що модель виявляє їх некоректно. Також, як і решту мереж глибинного навчання модель YOLO вимагає потужних обчислювальних ресурсів для ефективної роботи, через свою складну архітектуру та обчислювальні вимоги, що тільки ростуть із нововведеннями.

Перейдемо до самої архітектури моделі, яка зображена на рисунку (рис. 3.5.1).

Щільний блок містить кілька шарів згортки, причому кожен шар H_i складається з пакетної нормалізації нейронного вузла і супроводжується згорткою. Замість того, щоб використовувати вихідні дані лише останнього шару, шар H_i приймає як вихідні дані всіх попередніх шарів, а також оригінал. тобто x_0, x_1, … та x_{i-1}. Кожен шар H_i нижче виводить чотири карти функцій. Тому на кожному шарі кількість карт властивостей збільшується на чотири.

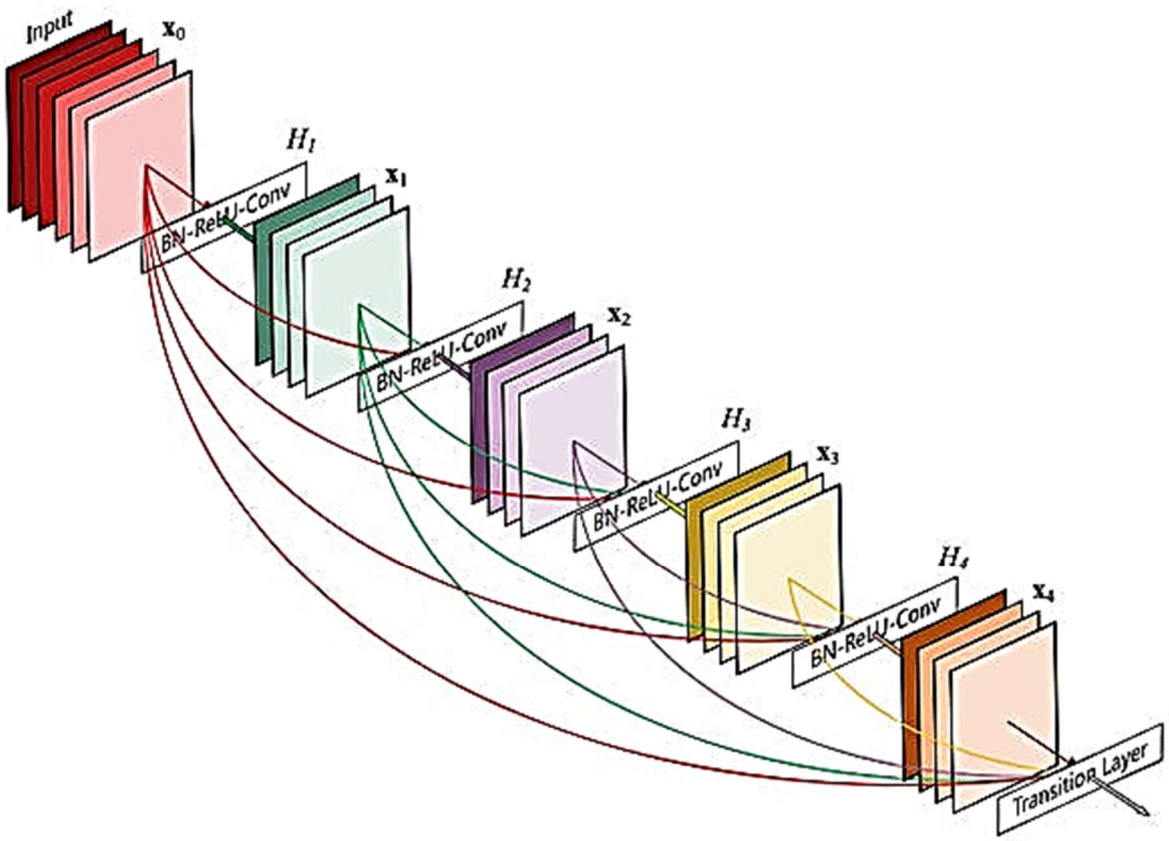


Рис. 2.5.1 Алгоритм YOLO

З моменту свого запуску, YOLO пройшов чимало покращень, на даний час існують нові версії, такі як YOLOv7n(nano), YOLOv7s,YOLOv7m та інші. Кожна з версій отримала оновлення для досягнення кращої точності та продуктивності в завданнях виявлення об'єктів.

Мною було обрано модель YOLOv5s, яка реалізована на бібліотеці машинного навчання PyTorch. За дослідженнями розробників моделі, за частотою обробки кадрів, для зображення з глибиною 640 пікселів, у секунду для різних версіях YOLO (рис. 3.5.2 та рис. 3.5.3), мною була обрана модель YOLOv5s, що отримала хороші показники частоти обробки даних, як на CPU так і на GPU.

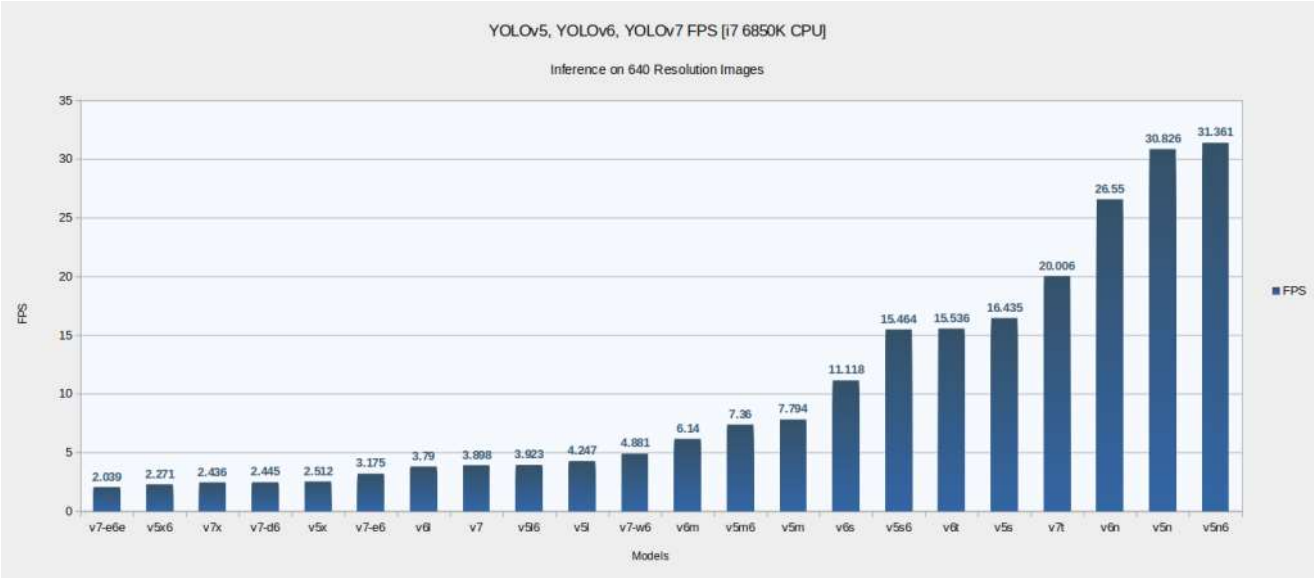


Рис. 3.5.2 Inference on 640 Resolution Image

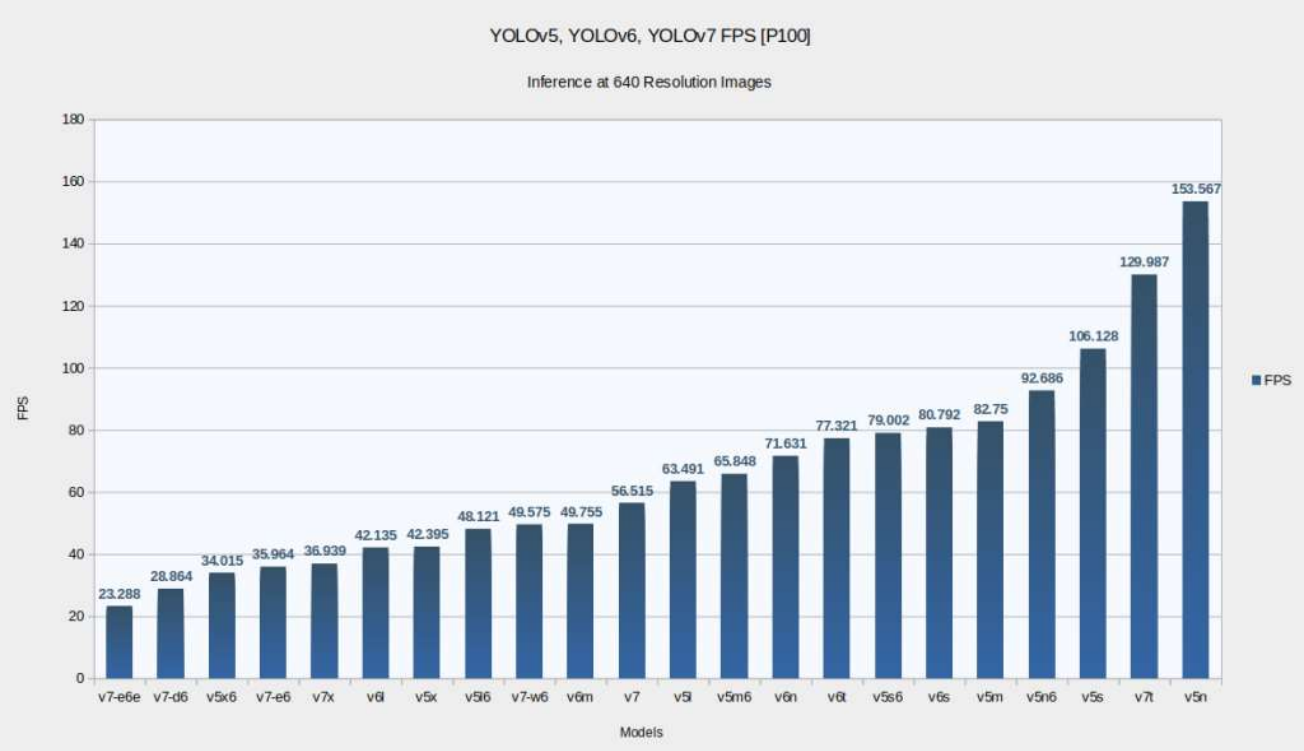


Рис. 3.5.3 Inference on 640 Resolution Image

4 РЕАЛІЗАЦІЯ АЛГОРИТМУ

4.1 KAGGLE

Головним вибором перед реалізацією застосунку стояло питання вибору середовища розробки та роботи. Адже більшість нейромереж для якісного навчання вимагають потужного обладнання із високими характеристиками. Зокрема такі пристрої потребують великого обсягу оперативної пам'яті, новітні процесори (найкраще підходять наприклад intel core i7 та вище, на рівні із його прототипами), та вільної кеш-пам’яті. Варто зазначити, що мій ноутбук отримав лише intel core i5, що вплинуло на роботу року давності, тут навчання нейромережі для розпізнання символів на зображеннях, займало у стані спокою – від 15ти годин.

Мною було прийнято рішення використати середовище для розробки та навчання нейронної мережі Kaggle. Kaggle. Вона була заснована у 2010 році, а у 2017 році – придбана компанією Google. Зараз Kaggle є достатньо популярною платформою для студентів та науковців усього світу, де вони мають змогу знайти цікаві дані та, зокрема, конкурувати один з одним у змаганнях з машинного навчання. Окрім цього, Kaggle також пропонує навчальні ресурси, які допомагають початківцям у сфері науки про дані та машинного навчання навчитися використовувати усі технології платформи у своїй роботі.

Kaggle містить широкий вибір бібліотек та інструментів для розробки нейронних мереж машинного навчання. Включно з такими бібліотеками, як: TensorFlow, Keras, PyTorch – це передові інструменти для якісного створення якісних застосунків. Окрім цього, середовище надає потужні обчислювальні ресурси, які можна використовувати для тренування нейронних мереж, що вимагають високі характеристики по обладнанню. Це і дозволило у подальшому зекономити час на тренування нейромережі та покращити точність моєї моделі. Та ще одною перевагою платформи є – різноманітна кількість наборів даних, що в подальшому можна використовувати для навчання та тестування моделей.

Тож, для отримання переваги у швидкості виконання алгоритмом необхідних для навчання та тренування кроків, було використано GPU100 – Tesla P100 із обсягом оперативної пам’яті 16 GB.(рис. 4.1.1). Скріншот використання GPU100 у ході навчання нейромережі зображено на рис.4.1.2

```
Setup complete. Using torch 2.0.0 (Tesla P100-PCIE-16GB)
```

Рис. 4.1.1 Встановлення GPU100 – Tesla P100

Run

3368.0s - GPU P100

Рис. 4.1.2 Час використання GPU100 для навчання та запуску моделі

4.2 DATASETS

Наступним кроком необхідно було вибрати датасети, які пізніше використовувалися для навчання нейромережі. Приклади деяких датасетів наведено в табличці(Рис. 4.2.1)[1].

Handwriting and character recognition [edit]

Google Translate

Dataset name	Brief description	Preprocessing	Instances	Format	Default Task	Created (updated)	Reference	Creator
Artificial Characters Dataset	Artificially generated data describing the structure of 10 capital English letters.	Coordinates of lines drawn given as integers. Various other features.	6000	Text	Handwriting recognition, classification	1992	[91]	H. Guvenir et al.
Letter Dataset	Upper case printed letters.	17 features are extracted from all images.	20,000	Text	OCR, classification	1991	[92][93]	D. Slate et al.
CASIA-HWDB	Offline handwritten Chinese character database. 3755 classes in the GB 2312 character set.	Gray-scaled images with background pixels labeled as 255.	1,172,907	Images, Text	Handwriting recognition, classification	2009	[94]	CASIA
CASIA-OLHWDB	Online handwritten Chinese character database, collected using Anoto pen on paper. 3755 classes in the GB 2312 character set.	Provides the sequences of coordinates of strokes.	1,174,364	Images, Text	Handwriting recognition, classification	2009	[95][94]	CASIA
Character Trajectories Dataset	Labeled samples of pen tip trajectories for people writing simple characters.	3-dimensional pen tip velocity trajectory matrix for each sample	2858	Text	Handwriting recognition, classification	2008	[96][97]	B. Williams
Chars74K Dataset	Character recognition in natural images of symbols used in both English and Kannada		74,107		Character recognition, handwriting recognition, OCR, classification	2009	[98]	T. de Campos
EMNIST dataset	Handwritten characters from 3600 contributors	Derived from NIST Special Database 19. Converted to 28x28 pixel images, matching the MNIST dataset.[99]	800,000	Images	character recognition, classification, handwriting recognition	2016	EMNIST dataset [100] Documentation [101]	Gregory Cohen, et al

Рис. 4.2.1 Датасети писаних літер та цифр

[1] Взята із списку популярних датасетів з вікіпедії. Див. Список використаних джерел[10]

Моїм вибором стали HandWritten_Character та standard OCR dataset. Перший складається із набору писаних літер та цифр та базується на MNIST dataset (Рис. 4.2.2), що складається з 80 000 одиниць, писаних арабських цифр, які зберігаються у форматі .jpg, 28*28 пікселя, написаних чорним по білому, для чіткого розпізнавання знаків. Другий містить зображення друкованих цифр, що зберігаються в аналогічному вигляді.

Окрім обсягу, великою перевагою даних датасетів є те, що вони використовується у бібліотеці Tenserflow, найбільшій бібліотеці машинного навчання Python.

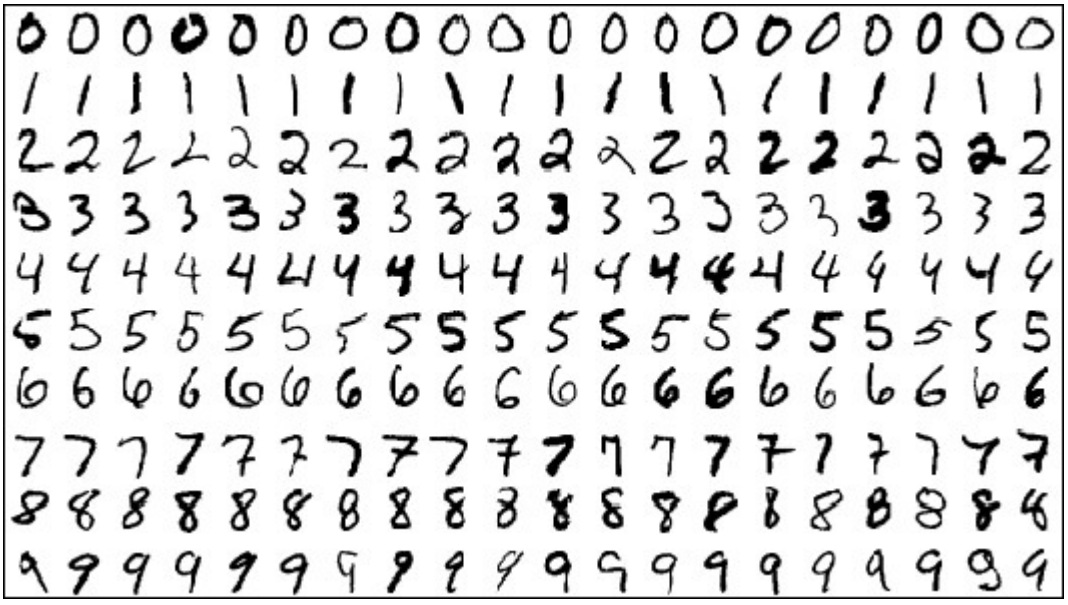


Рис. 4.2.2 Демонстрація MNIST dataset

Наступним кроком було встановлення необхідних пакетів та бібліотек для навчання нейромережі та для алгоритмів для її роботи загалом. У дипломній роботі я використала такі бібліотеки для машинного навчання, як: MatPlotLib, Keras, TensorFlow, Seaborn, Yaml, Scikit-Learn, Random, CV2, PyTorch, Pandas та NumPy.

4.3 МОДИФІКАЦІЯ ДАТАСЕТУ

Основна частина зусиль моєї дипломної роботи була зосереджена на цьому розділі. Зокрема, необхідність модернізації відчутна у процесі забезпечення різноманітного навчання моєї нейронної мережі. У загальному розумінні, потрібний датасет, має поєднувати символи обидвох вищезгаданих мною бібліотек (див. 4.2 DATASETS), у набори зображень, із різною кількістю цифр та фоном.

Отже, опісля встановлення необхідних інструментів, переходимо до створення набору зображень для навчання та тестування мережі. Спочатку рандомним чином обирається кількість цифр, які будуть розміщені на зображеннях, відповідно до чого формується тло заготовок. Опісля чого ці зображення піддаються маніпуляціям із зашумленням фону. Для цього було використано метод Гауса. Далі на отриману заготовку рівним чином (25*25) зображуються цифри, взяті із бібліотечних датасетів, що перед зображенням були позбавленні фону. Варто зауважити, що не всі цифри були чітко визначені алгоритмом і відібрані із тла, таким чином, деякі картинки датасетів, із сірим тлом, зображуються із рідним фоном на шаблонах. Приклад формування набору показано на рис. 4.3.1.

Одночасно із формуванням набору даних, формувалися текстові файли, що містять розташування цифр, відносно центру цифр та осей «x та y» зображень.(рис. 4.3.2)

Наступним кроком був формування набору даних із створених зображень. Для навчання було відібрано 4000 зображень, а для тренування – 400.

На цьому кроці можна різницю в часі по виконанню роботи алгоритмом на базі процесора ноутбуку(intel core i5) та GPU100 на Kaggle, а саме час створення тестових зображень та формування робочого датасету.(рис.4.3.3 та 4.3.4)

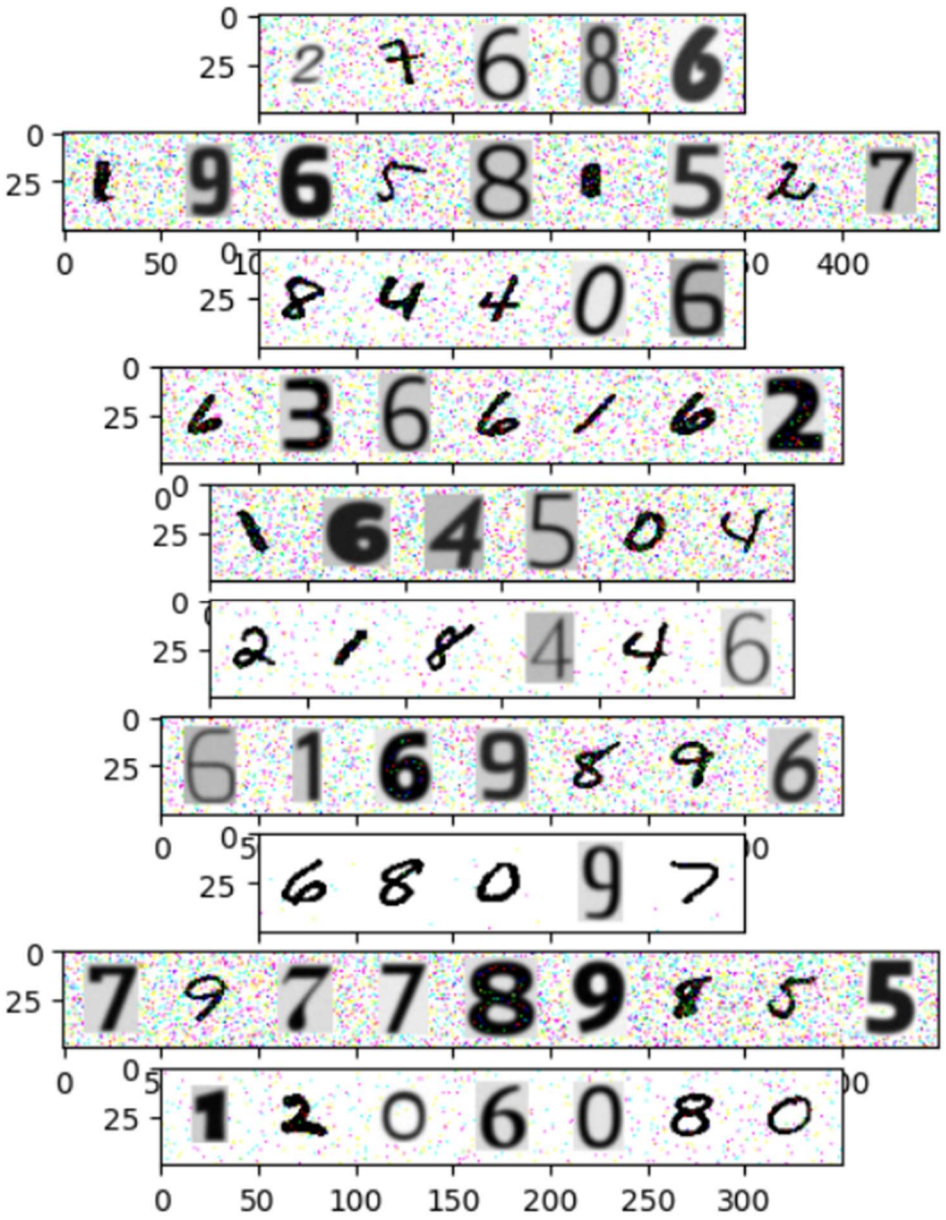
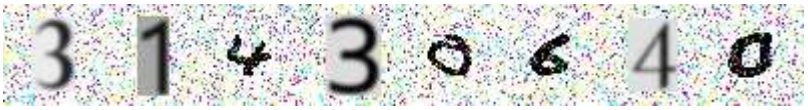


Рис. 3.3.1 Витяг із набору даних



```
3 0.0625 0.49 0.055 0.74
1 0.1875 0.51 0.04 0.78
4 0.3125 0.5 0.08 0.64
3 0.4375 0.5 0.07 0.8
0 0.5625 0.5 0.08 0.64
6 0.6875 0.5 0.08 0.64
4 0.81125 0.5 0.0625 0.72
0 0.9375 0.5 0.08 0.64
```

4.3.2 Елемент навчального набору

```
create_dataset(4000, 400)
```

```
100%|██████████| 4000/4000 [07:23<00:00,  9.02it/s]
100%|██████████| 400/400 [00:42<00:00,  9.51it/s]
```

Рис. 4.3.3 Виконання створення та формування набору даних на базі процесора intel CORE i5

```
100%|██████████| 4000/4000 [09:13<00:00,  7.23it/s]
100%|██████████| 400/400 [00:53<00:00,  7.41it/s]
```

Рис. 4.3.4 Виконання створення та формування набору даних на базі процесора GPU100

На рисунку (рис. 4.3.5) зображено початок процесу створення зображень навчальної вибірки, із 4000 елементів.

```
  0%|          | 0/4000 [00:00<?, ?it/s]  0%|          | 1/4000 [00:00<14:02,  4.75it/s]  0%|          | 2/4000
[00:00<12:03,  5.53it/s]  0%|          | 3/4000 [00:00<12:36,  5.29it/s]  0%|          | 4/4000 [00:00<13:54,
4.79it/s]  0%|          | 6/4000 [00:01<11:20,  5.87it/s]  0%|          | 7/4000 [00:01<13:16,  5.02it/s]  0%|
      | 8/4000 [00:01<13:07,  5.07it/s]  0%|          | 9/4000 [00:01<11:37,  5.73it/s]  0%|          | 10/4000
[00:01<10:52,  6.11it/s]  0%|          | 11/4000 [00:01<10:17,  6.46it/s]  0%|          | 12/4000 [00:02<10:32,
6.31it/s]  0%|          | 14/4000 [00:02<08:48,  7.54it/s]  0%|          | 15/4000 [00:02<08:59,  7.38it/s]  0%|
```

Рис. 4.3.5 Початок створення зображень для навчального набору даних

На даному рисунку (рис. 4.3.6) показано завершення формування навчального набору даних із 4000 елементів.

```
[09:11<00:02,  5.99it/s]100%|██████████| 3984/4000 [09:11<00:02,  6.02it/s]100%|██████████| 3985/4000
[09:11<00:02,  6.11it/s]100%|██████████| 3987/4000 [09:11<00:01,  8.02it/s]100%|██████████| 3988/4000
[09:11<00:01,  7.85it/s]100%|██████████| 3990/4000 [09:11<00:01,  7.92it/s]100%|██████████| 3991/4000
[09:12<00:01,  8.21it/s]100%|██████████| 3992/4000 [09:12<00:01,  7.89it/s]100%|██████████| 3993/4000
[09:12<00:00,  7.40it/s]100%|██████████| 3994/4000 [09:12<00:00,  7.48it/s]100%|██████████| 3995/4000
[09:12<00:00,  7.32it/s]100%|██████████| 3997/4000 [09:12<00:00,  8.49it/s]100%|██████████| 3998/4000
[09:13<00:00,  7.17it/s]100%|██████████| 3999/4000 [09:13<00:00,  6.98it/s]100%|██████████| 4000/4000
[09:13<00:00,  6.98it/s]100%|██████████| 4000/4000 [09:13<00:00,  7.23it/s]
```

Рис. 4.3.6 Завершення створення зображень для навчального набору даних

На рисунках 4.3.7 та 4.3.8 показано аналогічні процеси, тільки для тестового набору даних, що складається із 400 елементів.

```
  0%|          | 0/400 [00:00<?, ?it/s]  0%|          | 1/400 [00:00<00:51,  7.71it/s]  0%|          | 2/400
[00:00<00:59,  6.64it/s]  1%|          | 3/400 [00:00<00:59,  6.67it/s]  1%|          | 5/400 [00:00<00:41,
9.62it/s]  2%|          | 6/400 [00:00<00:50,  7.85it/s]  2%|          | 7/400 [00:00<00:47,  8.26it/s]  2%|
       | 8/400 [00:01<00:52,  7.52it/s]  2%|          | 9/400 [00:01<00:56,  6.88it/s]  2%|          | 10/400
[00:01<00:52,  7.39it/s]  3%|          | 12/400 [00:01<00:51,  7.59it/s]  3%|          | 13/400 [00:01<00:56,
6.85it/s]  4%|          | 14/400 [00:02<01:09,  5.54it/s]  4%|          | 15/400 [00:02<01:03,  6.03it/s]  4%|
          | 17/400 [00:02<00:56,  6.76it/s]  5%|          | 19/400 [00:02<00:45,  8.31it/s]  5%|          |
20/400 [00:02<00:51,  7.32it/s]  5%|          | 21/400 [00:03<01:02,  6.09it/s]  6%|          | 23/400
[00:03<00:55,  6.83it/s]  6%|          | 24/400 [00:03<00:51,  7.32it/s]  6%|          | 25/400 [00:03<00:53,
```

Рис. 4.3.7 Початок створення зображень для навчального набору даних

```
383/400 [00:51<00:02,  6.72it/s] 96%|          | 384/400 [00:51<00:02,  7.11it/s] 96%|          | 385/400
[00:51<00:02,  6.97it/s] 96%|          | 386/400 [00:51<00:02,  6.81it/s] 97%|          | 387/400 [00:52<00:01,
 6.61it/s] 97%|          | 388/400 [00:52<00:01,  6.26it/s] 97%|          | 389/400 [00:52<00:01,  6.31it/s]
98%|          | 390/400 [00:52<00:01,  6.25it/s] 98%|          | 391/400 [00:52<00:01,  5.73it/s] 98%|
          | 392/400 [00:52<00:01,  5.79it/s] 98%|          | 393/400 [00:53<00:01,  6.19it/s] 99%|          |
395/400 [00:53<00:00,  7.62it/s] 99%|          | 396/400 [00:53<00:00,  7.20it/s] 99%|          | 397/400
[00:53<00:00,  7.54it/s]100%|          | 398/400 [00:53<00:00,  7.03it/s]100%|          | 399/400 [00:53<00:00,
 7.37it/s]100%|          | 400/400 [00:53<00:00,  7.19it/s]100%|          | 400/400 [00:53<00:00,  7.41it/s]

642.8s    5     Setup complete. Using torch 2.0.0 (Tesla P100-PCIE-16GB)
```

Рис. 4.3.8 Завершення створення зображень для навчального набору даних

На рис. 4.3.9 зображено приклад набору навчальних даних, із 100% передбаченням, та їх вигляд у зібраному форматі,

Рис. 4.3.9 Приклад навчального набору даних

4.4 НАВЧАННЯ НЕЙРОМЕРЕЖІ

Нейромережа, а саме модель Yolo_v5, глибокого навчання, яку я застосувала для власного завдання – використовує фреймворк PyTorch, який є потужним інструментом взаємодії мови програмування Python у ході машинного навчання.

Для навчання нейромережі було обрано навчання з підсиленням, або іншими словами – навчання зі вчителем. Такий варіант найоптимальніший, адже ми не очікуємо того, що наша мережа визначатиме приховані залежності самостійно, це економить час і вимагає менше інших ресурсів, таких як кількість навчальних прикладів. Такий процес навчання реалізується за допомогою Епох, іншими словами – періодів навчання, чи уроків. Коли нейромережа проходить 1 епоху вона визначає кількість посилок, зроблених на даному етапі і виправляє їх за допомогою найбільш поширеного способу коригування навчання інтелектуальних систем – метод зворотного поширення помилки, коли ваги зв'язків мережі коригуються у зворотному порядку, починаючи від результатів, закінчуючи початковим шаром.

В середньому для навчання нейромережі спеціалісти беруть 50-ть епох. Я ж обрала 30, бо вже на 50-ти епохах мережа стикалася із проблемою перенавчання.

Особливістю використання моделі Yolo_v5 є те, що для навчання, алгоритм, потребує окремого файлу Yaml, який містить назви лейблів та розташування датасету. Формування такого файлу зображено на рис. 4.4.1.

```
671.4s   27   names:
671.4s   28   - '0'
671.4s   29   - '1'
671.4s   30   - '2'
671.4s   31   - '3'
671.4s   32   - '4'
671.4s   33   - '5'
671.4s   34   - '6'
671.4s   35   - '7'
671.4s   36   - '8'
671.4s   36   - '8'
671.4s   37   - '9'
671.4s   38   nc: 1
671.4s   39   train: /kaggle/working/dataset/images/train
671.4s   40   val: /kaggle/working/dataset/images/valid
```

Рис. 4.4.1 Процес формування спеціального файлу .yaml

Отже, наступним кроком був запуск навчання нашої нейромережі.(рис. 4.4.2) Навчання проводиться за допомогою скрипта train.py з ключами: data (шлях до датасету), img (дозвіл картинки), epochs (кількість епох), cfg (конфігурація розміру s/m/l), weights (стартові ваги).

```
# train
!WANDB_MODE="dryrun" python train.py --img 640 --batch 16 --epochs 30 --data /kaggle/workin
g/my_data.yaml --weights yolov5s.pt
```

Рис. 4.4.2 Запуск навчання нейромережі

Зауважу, що навчання відбувалося у декілька разів, адже зі зміною даних необхідно починати навчання мережі заново. Процес навчання повторювався більше семи разів. В більшості, головною потребою було те, що при оновленні даних, чи методу навчання, сам процес навчання необхідно було починати заново.

На рис. 4.4.3 можна побачити початок процесу навчання нейромережі на 30ти епохах, у меню log Message середовища Kaggle.

Time	#	Log Message
699.2s	96	Plotting labels to runs/train/exp/labels.jpg...
704.4s	97	Image sizes 640 train, 640 val
704.4s	98	Using 2 dataloader workers
704.4s	99	Logging results to [1mruns/train/exp [0m
704.4s	100	Starting training for 30 epochs...
704.4s	101	
704.4s	102	Epoch GPU_mem box_loss obj_loss cls_loss Instances Size
		0%| | 0/250 [00:00<?, ?it/s] 0/29 3.82G 0.1284 0.08089 0.06833 261
		640: 0/29 3.82G 0.1284 0.08089 0.06833 261 640: 0/29 3.87G
		0.1276 0.07889 0.06819 230 640: 0/29 3.87G 0.1276 0.07889 0.06819
		230 640: 0/29 3.87G 0.1274 0.07395 0.06806 195 640:

Рис. 4.4.3 Початок процесу навчання нейромережі на 30ти епохах

На рис. 4.4.4 та 4.4.5 показано процес навчання нейромережі у вигляді таблиць та підсумки на рис. 4.4.6.

Результатом є модель, що навчилася визначати цифри на навчальній вибірці із вірогідністю 0,977.

Зокрема, цифру 0 із вірогідністю – 0,979; 1 – 0,982; 2 – 0,966; 3 – 0,976; 4 – 0,981; 5 – 0,982; 6 - 0,983; 7 – 0,975; 8 – 0,978; 9 - 0,981.

```
Logging results to runs/train/exp
Starting training for 30 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       0/29      3.87G    0.06805    0.07934     0.0639        258        640: 1
                 Class     Images  Instances          P          R      mAP50
                   all        400       2796      0.162      0.331      0.167     0.0719

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       1/29      4.66G    0.04757    0.06133    0.06059        264        640: 1
                 Class     Images  Instances          P          R      mAP50
                   all        400       2796       0.36      0.365      0.358      0.23

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       2/29      4.66G    0.03973    0.05766    0.05762        250        640: 1
                 Class     Images  Instances          P          R      mAP50
                   all        400       2796       0.58      0.578      0.629     0.389
```

Рис. 4.4.4. Початок навчання нейромережі

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      27/29      4.66G   0.008276    0.02217   0.003189        242        640: 1
                 Class     Images  Instances          P          R      mAP50
                   all        400       2796      0.992      0.989      0.995     0.979

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      28/29      4.66G   0.007758    0.02143   0.003265        255        640: 1
                 Class     Images  Instances          P          R      mAP50
                   all        400       2796      0.994      0.992      0.995     0.978

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      29/29      4.66G   0.007428    0.02062    0.00312        269        640: 1
                 Class     Images  Instances          P          R      mAP50
                   all        400       2796      0.995       0.99      0.995     0.977

30 epochs completed in 0.718 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.3MB
```

Рис. 4.4.5 Завершення навчання нейромережі

```
Model summary: 157 layers, 7037095 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all        400       2796      0.992      0.989      0.995     0.979
                     0        400        300      0.996      0.993      0.995     0.982
                     1        400        275          1      0.992      0.995     0.966
                     2        400        288      0.991      0.976      0.994     0.982
                     3        400        258      0.995      0.992      0.995     0.976
                     4        400        304      0.992      0.997      0.994     0.981
                     5        400        274      0.989      0.984      0.995     0.982
                     6        400        280      0.989      0.989      0.994     0.983
                     7        400        288      0.982      0.993      0.993     0.975
                     8        400        234       0.99      0.996      0.995     0.978
                     9        400        295      0.994       0.98      0.995     0.981
```

Рис. 4.4.6 Підсумки навчання нейромережі

4.5 ТЕСТУВАННЯ МОДЕЛІ

Опісля успішного навчання мережі, наступним кроком є її тестування. У моєму випадку це проведення тестів на наборі із 400 зображень, сформованих у ході роботи алгоритму. На рисунку (рис.4.5.1) зображено запуск тестування навченої мережі (detect.py):

```
In [15]:
# detect valid images
!python detect.py --source '/kaggle/working/dataset/images/valid' --weights 'runs/train/exp/weig
hts/best.pt' --img 550 --save-txt --save-conf --exist-ok
```

Рис. 4.5.1 Запуск тестування навченої мережі

Усі отримані матеріали тестування, а саме результати передбачення збереглися у відповідній папці. Також, на поданих нижче рисунках можна спостерігати за процесом тестування моделі на тестовому наборі даних та ідентифікацією цифр на зображеннях, де зображення вибираються рандомним чином (рис. 4.5.2, рис. 4.5.3).

```
image 1/400 /kaggle/working/dataset/images/valid/0.jpg: 96x576 2 0s, 1 1, 1 3, 1 4, 2 9s,
22.2ms
image 2/400 /kaggle/working/dataset/images/valid/1.jpg: 96x576 2 0s, 1 4, 1 6, 3 8s, 1 9,
16.5ms
image 3/400 /kaggle/working/dataset/images/valid/10.jpg: 96x576 2 1s, 2 2s, 1 3, 1 4, 1
5, 1 6, 1 9, 16.3ms
image 4/400 /kaggle/working/dataset/images/valid/100.jpg: 128x576 2 1s, 1 7, 1 8, 1 9, 2
1.6ms
```

Рис. 4.5.2 Початок процесу тестування моделі

```
image 399/400 /kaggle/working/dataset/images/valid/98.jpg: 64x576 1 1, 2 3s, 2 6s, 3 7s,
1 9, 16.4ms
image 400/400 /kaggle/working/dataset/images/valid/99.jpg: 96x576 2 0s, 1 1, 1 2, 2 4s, 1
5, 1 7, 1 8, 16.4ms
Speed: 0.2ms pre-process, 16.6ms inference, 1.1ms NMS per image at shape (1, 3, 576, 576)
Results saved to runs/detect/exp
400 labels saved to runs/detect/exp/labels
```

Рис. 4.5.3 Завершення процесу тестування

РЕЗУЛЬТАТИ

Результати навчання та тренування зображено на рис 5.1.

У лівому верхньому куті зображено графік "train/box_loss", що відображає значення втрат "box_loss" протягом тренування. Він надає інформацію про те, як змінювалися втрати моделі щодо передбачення обмежувальних рамок під час тренування. Зменшення втрат "box_loss" до значення 0,1 свідчить про те, що модель навчилася краще локалізувати об'єкти та точно передбачати обмежувальні рамки.

Наступним є графік "train/obj_loss", який показує значення втрат "obj_loss" протягом тренування, інформує про втрати моделі відносно виділення об'єктів під час тренування. Зменшення втрат "obj_loss" свідчить про те, що модель навчилася краще виділяти об'єкти, відносно початку навчання. Аналогічно, зменшення втрат "cls_loss" на третьому графіку "train/cls_loss" вказує на покращення класифікації об'єктів протягом тренування.

Спостерігаючи за графіками "metric/precision" та "metric/recall", можна оцінити, як точність та чутливість моделі змінювалася протягом тренування. Високі показники точності (precision) свідчить про те, що більшість передбачень моделі були правильними, тоді як висока чутливість (recall) свідчить про те, що модель здатна виявляти більшість реальних об'єктів.

Ідентичні висновки можна зробити про нижній ряд графіків, який відповідає за результати тренування. А саме: у лівому нижньому куті зображено графік "val/box_loss", який говорить про те, що похибки виявлення обмежувальних рамок об’єктів варіювалися від 0,01 до 0,04. Наступний графік "val/obj_loss", показує зниження значень "obj_loss" що говорить про покращення ефективності моделі, хоча й недосконаліть у передбаченнях уже відчутна(від 0,11 до 0,16), хоч і не катастрофічно. Аналогічно, зменшення втрат "cls_loss" на нижньому третьому графіку, показує прогрес у ефективності виявлення та локалізації об’єктів натренованої моделі.

А вже графіки "metric/mAP_0.5" та "metric/mAP_0.5:0.95", що знаходяться у правому нижньому куті рис. 5.1 відносяться до метрик середньої точності виявлення об'єктів (mean Average Precision) для моделі YOLOv5. Тут "metric/mAP_0.5" відображає середню точність при порогові виходу моделі 0.5, отже, при виявленні об'єктів, що мають показник впевненості (confidence score) більше 0.5, розраховується точність і обчислюється середнє значення для всіх класів об'єктів. Графік "metric/mAP_0.5:0.95" відображає середню точність при порогові виходу моделі, що змінюється від 0.5 до 0.95 з кроком 0.05, що дозволяє більш детально отримати уявлення про точність натренованої моделі машинного навчання при різних порогах виходу і допомагає оцінити її продуктивність на різних рівнях впевненості.

Ці метрики допомагають оцінити ефективність моделі YOLOv5 у виявленні об'єктів з різними порогами виходу і надають усереднені значення точності для кожного порогу. Чим вищі значення mAP, тим краще продуктивність моделі виявлення об'єктів, що і показали виведені графіки, результати яких покращилися приблизно до показника 1.0

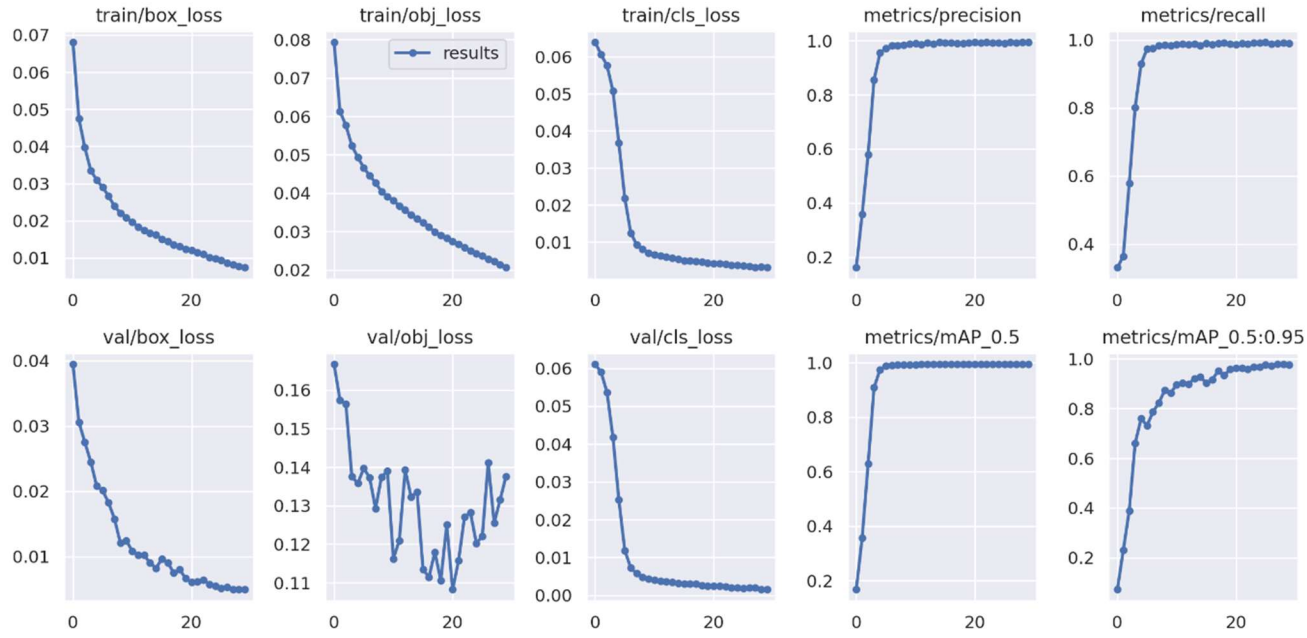


Рис. 5.1 Результати навчання та тренування мережі

Ще одним результуючим графіком, яким забезпечує нас модель YOLOv5 є, показаний на рис. 5.2 графік F1 Confidence Curve, що показує залежність значень F1-показника від рівня впевненості (confidence). Використовується для оцінки продуктивності, що враховує точність-precision (вісь y) і чутливість-recall (вісь x), моделі виявлення об'єктів, тобто графік F1 відображає впевненість мережі у своїх передбаченнях для різних порогів впевненості. У моєму випадку пороговим значенням є 0,787, який є достатнім результатом для задачі ідентифікації цифр.

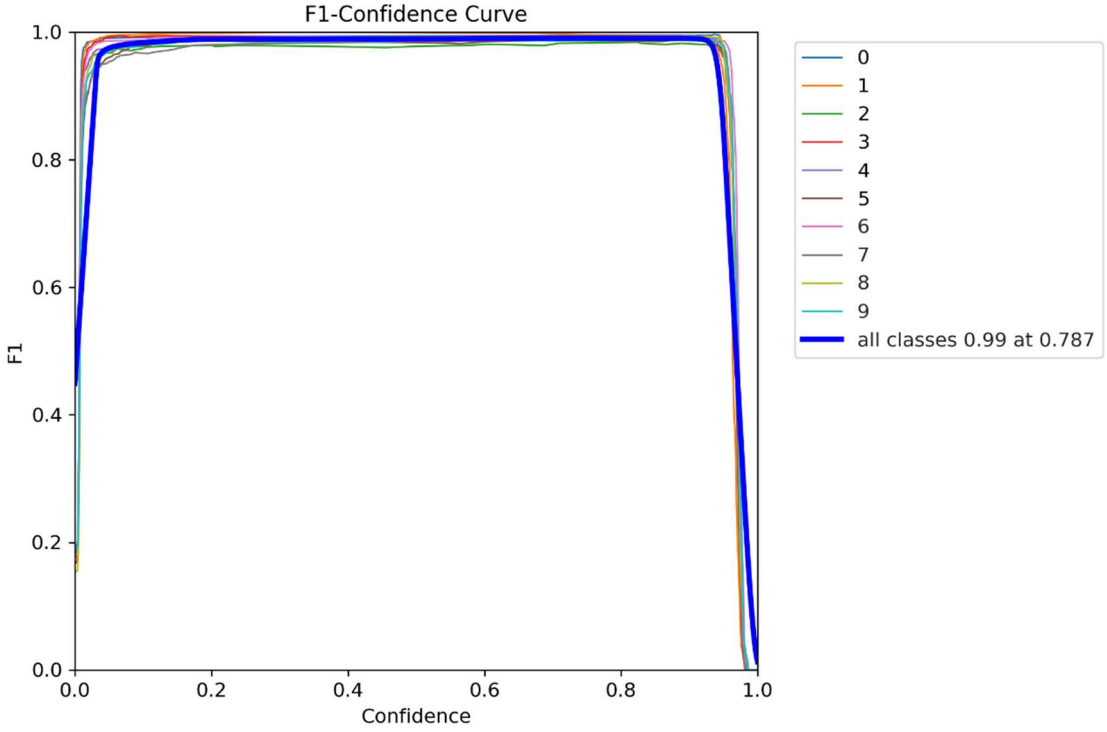


Рис. 5.2 F1-Confidence curve

На наступному рисунку (рис. 5.3) зображено матрицю похибок (Confusion Matrix) моделі у ході навчання. Матриця помилок подана у вигляді квадратної матриці, в якої рядки відповідають за передбачені цифри, а стовпці представлені у формі правдивих відповідей. Елементи по головній діагоналі матриці вказують на правильно передбачені об’єкти, в той час, як елементи поза діагоналлю представляють неправильно передбачення. Хочу зосередити увагу також на тому, що похибки мережа зробила на зашумлених

зображеннях, де градієнт background, який знаходиться справа від матриці похибок – показує рівень зашумлення даних.

Можна зробити висновок із матриці похибок: модель зробила похибку при ідентифікації цифри «2»(вказавши, що це «5» чи «6» на певних зображеннях), також помилково було обрано «7»(замість «1» та «9»), та, окрім цього, відхилення у передбаченнях отримали цифри «4» та «9» замінивши одна одну. Проте, дані випадки були одиничними, а мережа справилася із більшістю поставлених завдань.

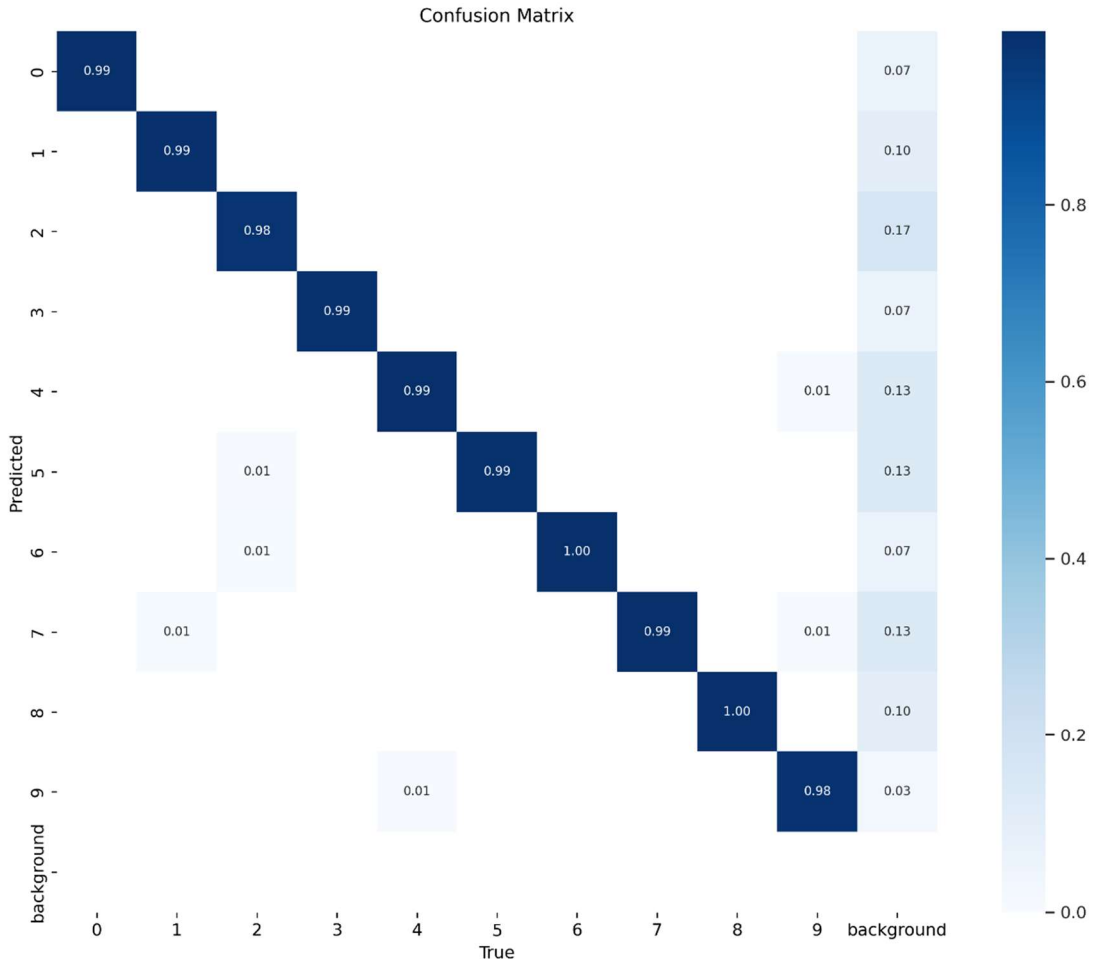


Рис. 5.3 Матриця помилок (Confusion Matrix)

На наступних зображеннях подано результати тестування моделі на вибірково обраному елементові із тестової вибірки даних. (рис. 5.4)

На фото показано дві цифри справа у кольоровій рамці, у яких ліва – мітка, яку призначила нейромережа об’єкту, права – впевненість мережі у своєму виборі. Єдиним мінусом такого виводу результатів є те, що близьке розміщення даних на зображенню – не дозволяє повністю розгледіти отримані результати. Проте, по отриманих даних можна зробити висновок, що – модель показує хороші результати ідентифікації цифр, на різних форматах фонів.

Рис. 5.4 Передбачення моделі на тестовому елементу вибірки

Також отримано результат на реальному прикладі, модель визначила усі цифри правильно.

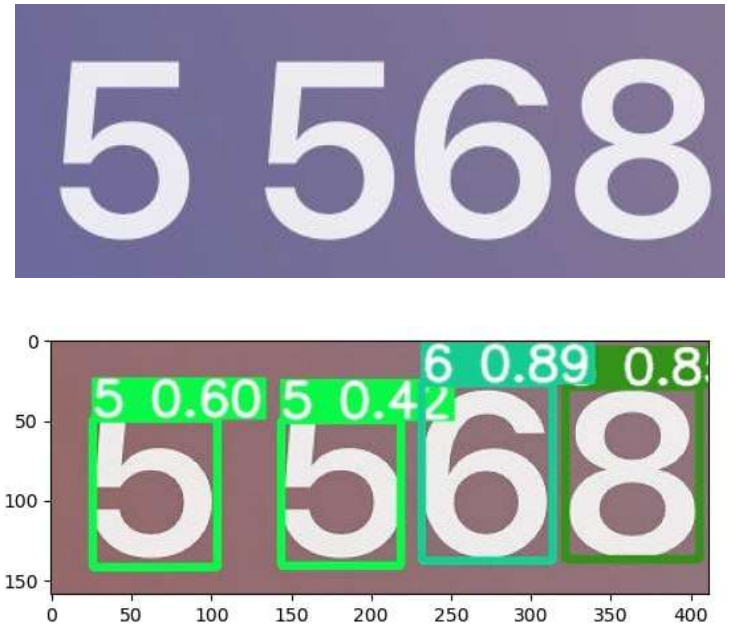


Рис. 5.5 Передбачення моделі на реальному зображенні

ВИСНОВКИ

У даній дипломній роботі було досліджено способи розв’язання задачі ідентифікації цифр на зображеннях.

Увагу було приділено середовищу розробки Kaggle, яке дозволяє використовувати надпотужні програмні інструменти, хмарне обладнання, різноманіття навчальних та тренувальних наборів даних та, зокрема, бібліотек машинного навчання.

Як спосіб побудови системи знаходження та ідентифікації цифр на зображеннях було застосовано модель глибинної згорткової нейронної мережі YOLO_v5. Було розглянуто особливості використання моделі, її історію, переваги та недоліки.

У ході дослідження, а саме під час тренування та використання навченої нейромережі було зроблено висновки за ступенем навченості, запам’ятовування та відстеження похибок під час навчання та тренування мережі.

Варто зауважити, що навчена модель машинного навчання повністю справляється із завданням ідентифікації цифр на зображенні. Проте, якщо розглядати зображення із дуже дрібними цифрами, то мережа, через свою складну структуру просто не вважатиме їх за об’єкт.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Practical Convolutional Neural Networks: Implement advanced deep learning. / Mohit Sewak, Md. Rezaul Karim, Pradeep Pujari. – Packt Publishing, 2018. – 7- 46с.
2. Computer Vision with Python 3 / Saurabh Kapur – BIRMINGHAM – MUMBAI, 2017. – 80 -109с.
3. Computer Vision: Algorithms and Applications / Richard Szeliski. – Springer, 2022. – 235-343с.
4. Deep Learning / Ian Goodfellow, Yoshua Bengio, Aaron Courville. – The MIT Press, 2016. – 95-267с.
5. Practical Computer Vision: Extract insightful information from images using TensorFlow, Keras, and OpenCV / Abhinav Dadhich. – Kindle Edition, 2018.
6. Handwritten Digit Recognition with a Back-Propagation Network / [Yann LeCun, Bernhard E. Boser, John S. Denker, etc]. – NIPS, 1989. – 396- 404с.
7. ХНУРЕ, ПЗЕОМ, Основні положення теорії штучних нейронних мереж [Електронний ресурс] / Шатовська Т. Б., Рєпка В. Б. – Харків, 2010. – Режим доступу: https://dl.nure.ua/pluginfile.php/634/mod_resource/content/2/content/content1.html
8. Top 8 Algorithms For Object Detection [Електронний ресурс] / Ambika choudhury – Mystery Vault, 2020. – Режим доступу: https://analyticsindiamag.com/top-8-algorithms-for-object-detection/
9. Сучасні підходи до розв’язання задач комп’ютерного зору [Електронний ресурс] / Р.М. ТИМЧИШИН, О.Є. ВОЛКОВ, О.Ю. ГОСПОДАРЧУК // Control systems and computers, 2018. – 46-69с. – Режим доступу: пhttp://dspace.nbuv.gov.ua/bitstream/handle/123456789/161565/06-Tymchyshyn.pdf?sequence=1

10. Basics of Yolo v5 - Balloon Detection. [Електронний ресурс] – VBOOKSHELF, 2021. – Режим доступу: https://www.kaggle.com/code/vbookshelf/basics-of-yolo-v5-balloon-detection/notebook
11. YOLOv5 PyTorch TXT [Електронний ресурс] / Roboflow. – Roboflow, Inc, 2020 – Режим доступу: https://roboflow.com/formats/yolov5-pytorch-txt
12. Як працюють нейронні мережі? [Електронний ресурс] / Igor Sikorsky. – Режим доступу: http://apeps.kpi.ua/neural-networks/en
13. Нейронні мережі - шлях до глибинного навчання. [Електронний ресурс] / Codeguida, 2017 – Режим доступу: https://codeguida.com/post/739