МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ»**

Виконав: студент 2 курсу, групи 8.1211-1іпз

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

В.В. Лобачов
(ініціали та прізвище)

Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри комп'ютерних наук, професор, д.т.н. Чопоров С.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент
_____ Лісняк А.О.
(підпис)

« ____ » _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Лобачову Володимиру Володимировичу
(прізвище, ім'я та по-батькові)

1. Тема роботи (проєкту) Розробка системи розпізнавання дорожніх знаків

керівник роботи (проєкту) Лісняк Андрій Олександрович, к.ф.-м.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 4 » травня 2022 року № 500-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік питань до розробки.
3. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Розробка тестових випадків.
3. Розробка системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану виконання кваліфікаційної	23.07.2022	
	роботи магістра.		
2.	Збір вихідних даних та аналіз предметної	10.08.2022	
	області.		
3.	Обробка методичних та теоретичних джерел.	25.08.2022	
4.	Специфікація вимог до системи. Робота над	06.09.2022	
	першим розділом.		
5.	Проектування системи. Робота над другим	21.09.2022	
	розділом.		
6.	Реалізація та тестування системи. Робота над	22.09.2022	
	третім розділом.		
7.	Розробка керівництва користувача. Робота	17.10.2022	
	над додатками.		
8.	Оформлення та нормоконтроль	25.11.2022	
	кваліфікаційної роботи магістра.		
9.	Захист кваліфікаційної роботи магістра.	15.12.2022	

Студент _____ В.В. Лобачов
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка системи розпізнавання дорожніх знаків»: 59 с., 21 рис., 30 джерел.

ВИЗНАЧЕННЯ ЕМОЦІЙ, ГЛИБОКЕ НАВЧАННЯ, ЕПОХА, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, КОМП'ЮТЕРНА СИСТЕМА, НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ, OPEN CV.

Мета і завдання дослідження полягають у вивченні сучасних підходів до визначення дорожніх знаків, а також у розробці комп'ютерної системи, що буде ефективно вирішувати цю задачу.

У процесі дослідження була розглянута проблема визначення дорожніх знаків та підходи глибинного навчання до її вирішення. Як результат, була розроблена та навчена оптимальна модель глибокої нейронної мережі. Дана мережа розпізнає дорожні знаки.

SUMMARY

Master’s Qualification Paper «Development of the Road Sign Recognition System»: 59 pages, 21 figures, 30 references.

DETECTION OF EMOTIONS, DEEP LEARNING, ERA, CONVOLUTIONAL NEURAL NETWORK, COMPUTER SYSTEM, NEURAL NETWORK, ROAD SIGNES RECOGNITION, OPEN CV.

The aim of the research is to study modern approaches to the road signs recognition, as well as to develop a computer system that will effectively solve this problem.

In the process of research, the problem of determining road signs and deep learning approaches to its solution were considered. As a result, an optimal deep neural network model was developed and trained. This network recognizes road signs.

ЗМІСТ

ВСТУП

В сучасних умовах розвитку інноваційних технологій у всіх галузях і сферах діяльності людини з’явилися нові наукові напрями. Швидкого та інтенсивного піднесення за останні роки зазнала інформатика, що виросла з класу теоретичних фундаментальних дисциплін та значно розширила практичні сфери свого застосування.

Розвиток штучних нейронних мереж тісно пов'язаний з біологією. Штучний нейрон – це спрощена модель біологічного нейрона. Математично він представляє собою деяку нелінійну функцію (функцію активації) від одного аргументу, що є лінійною комбінацією вхідних сигналів. Зв’язки між нейронами, за аналогією зі зв’язками між природними нейронами, називаються синапсами.

Штучний нейрон має єдиний вихід, який інколи називають аксоном. Штучні нейрони об’єднують, утворюючи при цьому штучні нейронні мережі.

Важливою властивістю нейронних мереж, що свідчить про їх великий потенціал і широкі прикладні можливості – паралельна обробка інформації одночасно великою кількістю нейронів. Завдяки цьому досягається значне пришвидшення обробки інформації. Іншою не мнеш важливою особливістю нейронних мереж є здатність до навчання та узагальнення інформації. Таким чином досягається деяка схожість з роботою головного мозку людини.

Останнім часом спостерігається тенденція зростання інтересу до використання нейронних мереж для вирішення різних завдань і застосування їх в різних сферах людського життя.

З використанням нейронних мереж відкрилися можливості проведення обчислень в сферах, що до цього відносилися лише до сфери людського інтелекту. З’явилися можливості створення систем, які здатні вчитися, запам'ятовувати та аналізувати інформацію, що дуже нагадує розумові здібності людини.

Типовими задачами, що можуть бути вирішеними за допомогою нейронних мереж та нейрокомп'ютерів є: задача класифікації, автоматизація прогнозування, автоматизація процесу ухвалення рішень, управління, кодування і декодування інформації, розпізнавання образів та ін.

Нейронні мережі можуть використовуватися майже в усіх галузях і сферах діяльності людини: економіці, медицині, зв’язку і безпеці охоронних систем, обробці інформації.

У галузі безпеки і охоронних системах нейронні мережі необхідні для ідентифікації особи, розпізнавання голосу, осіб в натовпі, розпізнавання автомобільних номерів, аналіз аэрокосмічних знімків, моніторингу інформаційних потоків, виявлення підробок.

У галузі обробка інформації нейронні мережі можуть застосовуватися для обробки чеків, розпізнавання підписів, відбитків пальців і голосу.

Розроблені італійською фірмою R Informati нейромережеві пакети серії FlexR d, використовуються для розпізнавання і автоматичного введення рукописних платіжних документів і податкових декларацій. У першому випадку вони застосовуються для розпізнавання не тільки кількості товарів і їх вартості, але також і формату документа. У разі податкових декларацій розпізнаються фіскальні коди і суми податків.

Отже, в сучасному світі нейронні мережі це не далеке майбутнє. Нейроінфоматикою та дослідженнями нейромереж у різних галузях займаються науковці з усього світу. За допомогою штучних нейронних мереж можна опрацьовувати, аналізувати та узагальнювати інформації, що аналогічно роботі головного мозку людини. Нейтронні мережі використовуються у економіці, медицині, зв'язку, безпеці та охоронних системах, введенні та обробці інформації. Безумовно, даний перелік не повний, проте він дозволяє отримати уявлення про характер застосування нейромережевих технологій.

1 ОГЛЯД ТА ХАРАКТЕРИСТИКА НЕЙРОННИХ МЕРЕЖ

1.1 Загальний огляд проблеми

Сьогодні у великих містах на дорогах можна спостерігати багато незадовільнених людей через те, що доводиться деякий час чекати у заторах. За даними щорічного дослідження Traffic Index у 2020 році Харків займає тринадцяте місце серед міст світу із самими великими заторами на дорогах, до цього списку входить більше чотирьох ста міст. Так через затори у пікові години поїздка у середньому збільшується на двадцять хвилин [2]. Транспортна мережа явно досягає своїх меж. Це було предметом громадського обговорення протягом десятиліть. Але як можна уникнути пробок, оптимізувати транспортний потік і знизити число дорожньо-транспортних пригод зі смертельними наслідками? Загальновідомо, що зміна правил дорожнього руху сама по собі не вирішить проблему.

Штучний інтелект може використовуватися як вибірково, так і комплексно для дорожнього руху і особливо для водіння. Деякі з функцій, в яких успішно використовується ШІ – це, наприклад, автоматичне розпізнавання відстані або паркування. Ці системи стають все більш складними і точними завдяки великим обсягам навчальних даних.

Штучний інтелект володіє величезним потенціалом в області дорожнього руху. Це може зробити водіння не тільки більш комфортним і безпечним, але також більш екологічним і інтелектуальним. Передові алгоритми, засновані на принципах штучного інтелекту, машинного навчання і великих даних, враховують обсяг трафіку в режимі реального часу для оптимізації транспортного потоку і підвищення індивідуальної мобільності. В результаті користувач досягає місця призначення максимально швидко, безпечно та з комфортом.

Багато процесів дорожнього руху можуть бути значно поліпшені. Кожен водій, якому доводиться чекати на світлофорі кілька хвилин поспіль, навіть якщо для цього немає видимої причини – за винятком того, що система світлофора працює за фіксованою схемою, яка повністю не залежить від поточної дорожньої ситуації, може мати до цього відношення. Використання штучного інтелекту для підтримки руху трафіку відповідно до поточної ситуації має багато переваг:

– безперервний рух, без пробок, це корисно для навколишнього середовища;
– дозволяє оптимізувати багато бізнес-процесів, такі як поставки, що приносить велику користь економіці.

Всі ці фактори сприяють оптимізації загальної транспортної системи. Це вигідно кожному учаснику дорожнього руху, навіть тим, хто раніше міг брати участь у дорожньому русі лише в обмеженій мірі без допомоги цифрових інструментів. На якість програмного забезпечення, призначеного для використання в дорожньому русі, впливає, з одного боку, програмування алгоритмів, але також в значній мірі кількість і якість навчальних даних. Чим надійніше і реалістичніше навчальні дані для машинного навчання, тим більше можливостей для безпечного проектування дорожнього руху.

Машинне розпізнавання об'єктів на зображеннях полягає в здатності автоматично класифікувати вихідні дані і відносити їх до певного класу за допомогою виділення характерних ознак об'єкта. Проблема автоматичного розпізнавання об'єктів на зображеннях є відносно новою і отримала свій розвиток у другій половині 20 століття. У перших системах розпізнавання використовувалися прості геометричні моделі, і була потрібна участь адміністратора, який виробляв виділення ознак об'єкта на зображенні. Потім система виконувала чисельні вимірювання розмірів і відстаней виділених ознак щодо контрольних точок.

В даний час розпізнавання об'єктів на зображеннях здійснюється на основі складних математичних уявлень про існуючі процеси і є дуже затребуваною в областях комп'ютерного зору, обробки і аналізу зображень, біометрії, систем

безпеки та відео-контролю. Сучасні методи, які вирішують задачу розпізнавання об'єктів, застосовуються для вирішення широкого кола завдань: розпізнавання осіб, відбитків пальців, сітківки ока, друкованих символів, автомобільних номерних знаків, маркування на поверхнях різних об'єктів і т.д.

На сьогоднішній день досягнуто значних успіхів при вирішенні задач розпізнавання об'єктів і символів на зображеннях, проте існує ряд складнощів, які істотно знижують надійність застосування сучасних методів:

– низька роздільна здатність зображень, в результаті чого, ознаки об'єктів можуть бути погано помітні;
– наявність складної фонової структури на зображеннях – це має на увазі наявність на зображеннях сторонніх об'єктів, які можуть мати візуальні ознаки, схожі з шуканим об'єктом;
– різні спотворення, отримані в процесі реєстрації зображень.

Реєстрація зображень може проводитися при невдалих ракурсах, поганих погодних умовах, різних кутах і умов освітлення. В результаті цього, на зображеннях можуть бути різні шумові перешкоди, символи можуть бути схильні до аффінним і проекційним спотворень; кількість шуканих об'єктів на зображенні заздалегідь не відомо. Все це вимагає застосування різних алгоритмів попередньої обробки, що в свою чергу ускладнює процес розпізнавання, робить його більш громіздким, збільшує обсяг і час обчислювальних процесів.

Тому на сьогоднішній день до сих пір існує потреба в розробці методів і алгоритмів, які вирішують перераховані вище проблеми.

1.2 Архітектура нейронних мереж

Будь-яка архітектура ШНМ складається з штучних нейронів – елементів обробки, мають структуру трьох пов'язаних один з одним шарів: вхідним, що складається з одного або кількох шарів, прихованим і вихідним (рис. 1.1).

Вхідний шар складається з вхіднихнейронів, які передають інформацію в

приховані шари. Прихований шар в свою чергу передає інформацію в вихідний. Кожен нейрон має входи з вагами – синапсами, функцію активації, визначальну вихідну інформацію при заданої вхідної, і один вихід. Синапси – регульовані параметри, що конвертують нейронну мережу в параметризрвану систему.Для розгляду поняття «тріангуляції» приведемо деякі визначення.

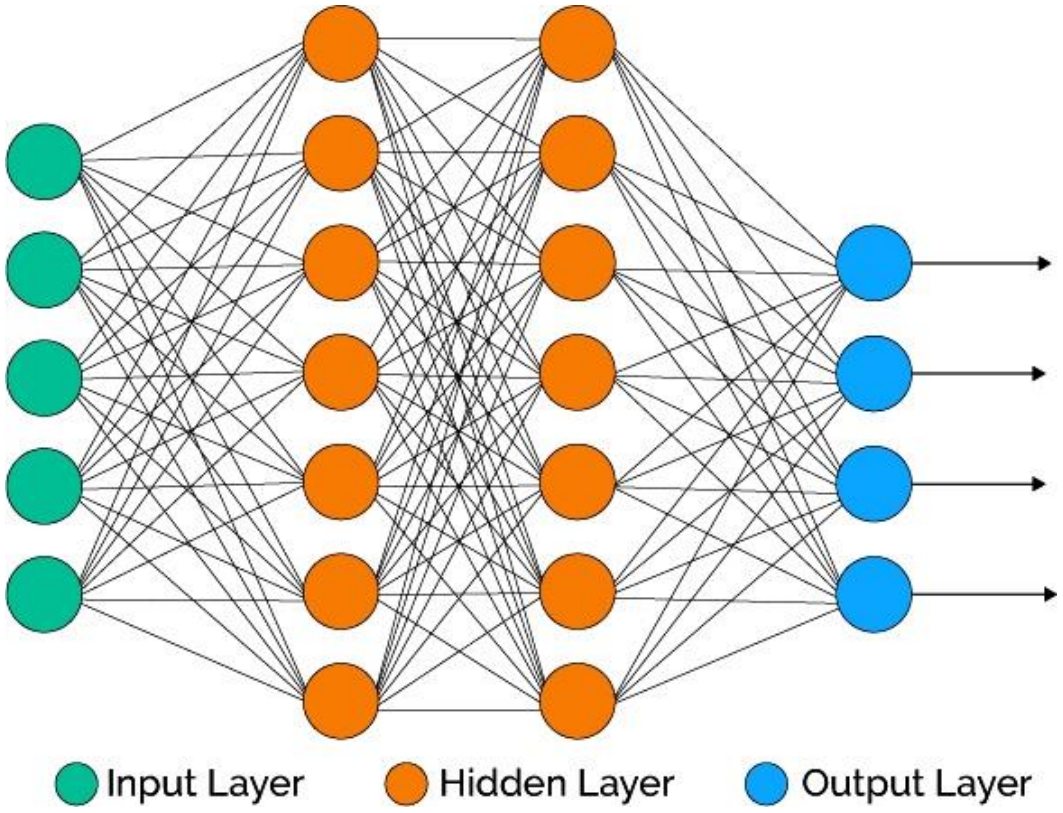

Рисунок 1.1 – Архітектура штучної нейронної мережі

Навчання (або тренування) – процес оптимізації ваг, в якому мінімізується помилка передбачення, і мережа досягає необхідного рівня точності. Найбільш використовуваний метод для визначення внеску в помилку кожного нейрона – зворотнє поширення помилки, за допомогою якого обчислюють градієнт. Це одна з модифікацій методу градієнтного спуску.

За допомогою додаткових прихованих шарів можливо зробити систему більш гнучкою і потужною. ШНМ з багатьма прихованими шарами називаються глибокими нейронними мережами; вони створюють складні нелінійні зв'язки.

Розглянемо популярні архітектури нейронних мереж, які добре показали

себе в задачах нейролінгвістичного програмування і рекомендуються до використання.

1.3 Методи навчання нейронних мереж

Основні методи навчання нейронних мереж, такі як: зворотне поширення помилки (Back-propagation); стохастичний градієнтний спуск (Stochastic Gradient Descent); адаптація швидкості навчання (Learning Rate Decay); випадкове видалення ваг (Dropout); дискретизація на основі вибірки (Max Pooling); пакетна нормалізація (Batch Normalization); довга короткотермінова пам'ять (Long Short-Term Memory); безперервний пакет словникової моделі (Continuous Bag Of Words) та передавання навчання (Transfer Learning).

Першим з методів навчання нейронних мереж є зворотне поширення помилки (Back-propagation) – це метод для обчислення часткових похідних (або градієнтів) функції, яка має форму функціональної композиції (як у нейронних мережах). При вирішенні задач оптимізації, використовуючи градієнтні методи (градієнтний спуск є лише одним з них), обчислюється градієнт функції на кожній ітерації.

1.4 Ефективність підходів глибинного навчання до розпізнавання об’єктів

Може виникнути питання: чому підходи глибинного навчання для розпізнавання об’єктів, є настільки ефективними у порівнянні з класичними методами комп’ютерного зору? Це питання розкривається у дослідженні [8].

Традиційним підходом комп’ютерного зору до розпізнавання об’єктів є використання дескрипторів ознак (HOG та інших) [9]. До появи глибинного навчання для вирішення таких задач, як класифікація зображень, виконувався

крок вилучення ознак. Ознаки – це невеликі інформативні області на зображеннях, які однозначно описують той чи інший об’єкт. На цьому кроці можуть бути задіяні декілька алгоритмів комп’ютерного зору: виявлення країв, виявлення кутів чи сегментація порогових значень. Вилучені із зображень ознаки формують визначення (опис) об’єктів кожного класу. Дескриптор ознак намагається знайти ці визначення на зображеннях, які подаються йому на вхід. Якщо значна кількість певних ознак присутня на вхідному зображенні, він класифікує його як таке, що містить об’єкт певного класу. Наприклад, для дескриптора ознак на основі гістограм напрямлених градієнтів (HOG) ознаками є розподіл градієнтних векторів (їх напрям та величина) на зображенні.

Складність традиційного підходу полягає в тому, що інженер повинен обрати, які саме ознаки найкраще описують той чи інший клас об’єктів. Це тривалий процес спроб та помилок. Зі збільшенням кількості класів об’єктів вилучення ознак стає більш громіздким. Більше того, процес вилучення ознак використовує безліч параметрів, кожен із яких повинен бути точно налаштований інженером.

Глибинне навчання – це концепція наскрізного навчання, у якій модель глибокої нейронної мережі навчається виявляти основні закономірності різних класів об’єктів на навчальній вибірці. Тобто, нейронна мережа не програмується під конкретну задачу, а навчається її вирішувати, як це роблять люди. Добре встановлено, що глибокі нейронні мережі працюють набагато краще, ніж традиційні алгоритми комп’ютерного зору, хоча і з компромісами щодо обчислювальних вимог та часу, необхідного для навчання моделі. Зважаючи на це, можна констатувати, що процес роботи інженера в області комп’ютерного зору кардинально змінився: знання та досвід у вилученні створених «вручну» ознак об’єктів були замінені знаннями та досвідом вибору чи створення найкращої архітектури нейронної мережі для вирішення задачі.

Розробка архітектури нейронної мережі, яка навчається вилучати ознаки (характеристики) об’єктів на зображенні, а саме згорткової нейронної мережі (Convolutional Neural Network), призвела до прориву у вирішенні задачі

класифікації об’єктів. Архітектура згорткової нейронної мережі була описана ще у 1998 році Яном Лекуном та його командою у статті, але її активне використання почалося лише з 2012 року, коли інженери у дослідженні [10, 11] розробили та навчили модель глибокої згорткової нейронної мережі AlexNet класифікувати 1.2 мільйони зображень на 1000 різних класів. Стрімкий розвиток методів глибинного навчання для класифікації, а пізніше – для розпізнавання об’єктів, обумовлений зростанням обчислювальної потужності та збільшенням обсягу даних, необхідних для навчання нейронних мереж.

1.5 Згорткові нейронні мережі

Згорткові нейронні мережі були вперше описані Яном Лекуном та його командою у дослідженні [10], де вони вирішували задачу розпізнавання рукописних цифр із датасета MNIST. Через недостатню обчислювальну потужність та кількість даних для навчання згорткові нейронні мережі не набували широкого розповсюдження. Усе змінилося у 2012 році, коли Алекс Крижевський та його команда розробили модель AlexNet, здатну класифікувати 1.2 мільйони зображень на 1000 різних класів з похибками top-1 та top-5 у 37.5 % та 17 % відповідно[11]. У статті [17] наведений детальний огляд принципу роботи згорткової нейронної мережі.

Згорткова нейронна мережа – це нейронна мережа прямого поширення спеціального виду, яка займається розпізнаванням ознак певних класів об’єктів на зображенні. У порівнянні з нейронною мережею прямого поширення згорткова нейронна мережа використовує значно меншу кількість параметрів. На рисунку 1.2 зображена її архітектура.

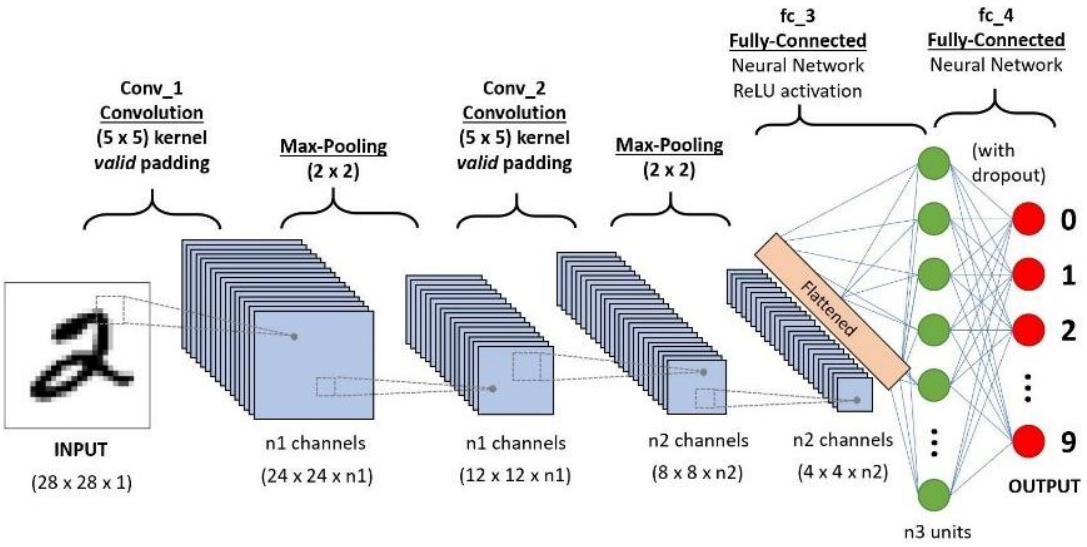

Рисунок 1.2 – Архітектура згорткової нейронної мережі

Перший шар згорткової нейронної мережі – згортковий (convolution layer). Згорткова нейронна мережа працює на основі фільтрів, які займаються розпізнаванням ознак певних класів об’єктів на зображенні: прямих ліній, кривих певного типу тощо. Фільтр – це матриця чисел (ваг), які навчаються виявляти ознаки певних об’єктів на зображенні. Фільтр переміщається уздовж зображення та виявляє, чи присутня деяка ознака у його частині. Для отримання такої відповіді здійснюється операція згортки (convolution operation), яка математично визначається як сума добутків елементів фільтру та частини вхідного зображення. Назва згорткової нейронної мережі походить саме від назви цієї операції. На рисунку 1.3 зображена операція згортки.

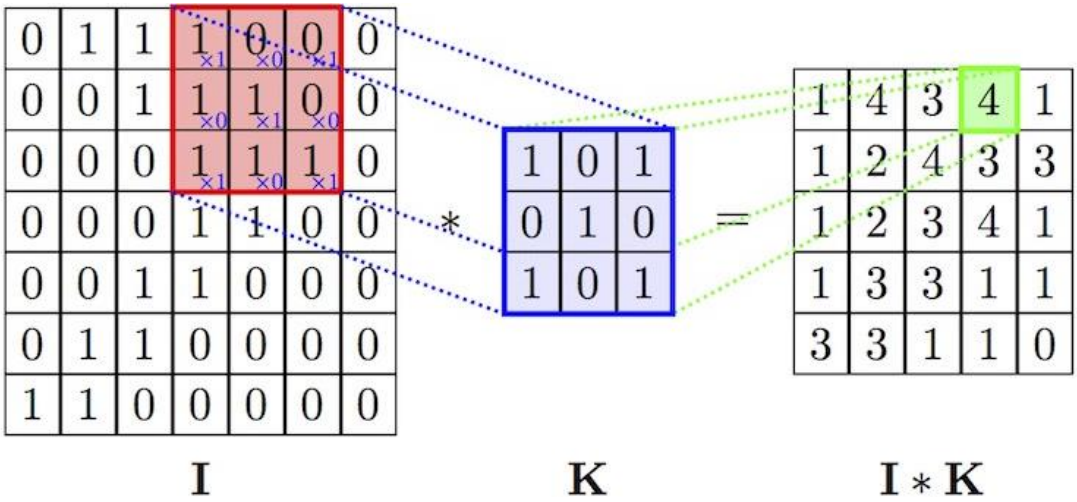

Рисунок 1.3 – Операція згортки

Якщо ознака присутня у певній частині зображення, в результаті згортки ми отримаємо велике значення. В іншому випадку, ми отримаємо або мале значення, або 0. Цей факт наочно продемонстрований на рисунку 1.4. Фільтр займається розпізнаванням напрямлених праворуч кривих. У першому випадку він виявив шукану криву, оскільки в результаті ми отримали досить велике значення: 50×30+50×30+50×30+20×30+50×30=6600. У другому випадку шукана крива не була виявлена, оскільки ми отримали 0.

Для виявлення характеристик на кольоровому (RGB) зображенні фільтр представляє собою тривимірну матрицю, кожен елемент якої містить значення для трьох каналів: червоного, зеленого та синього. Фільтр можна переміщувати уздовж зображення з кроком, відмінним від одиниці. Крок переміщення фільтру називається страйдом (stride).

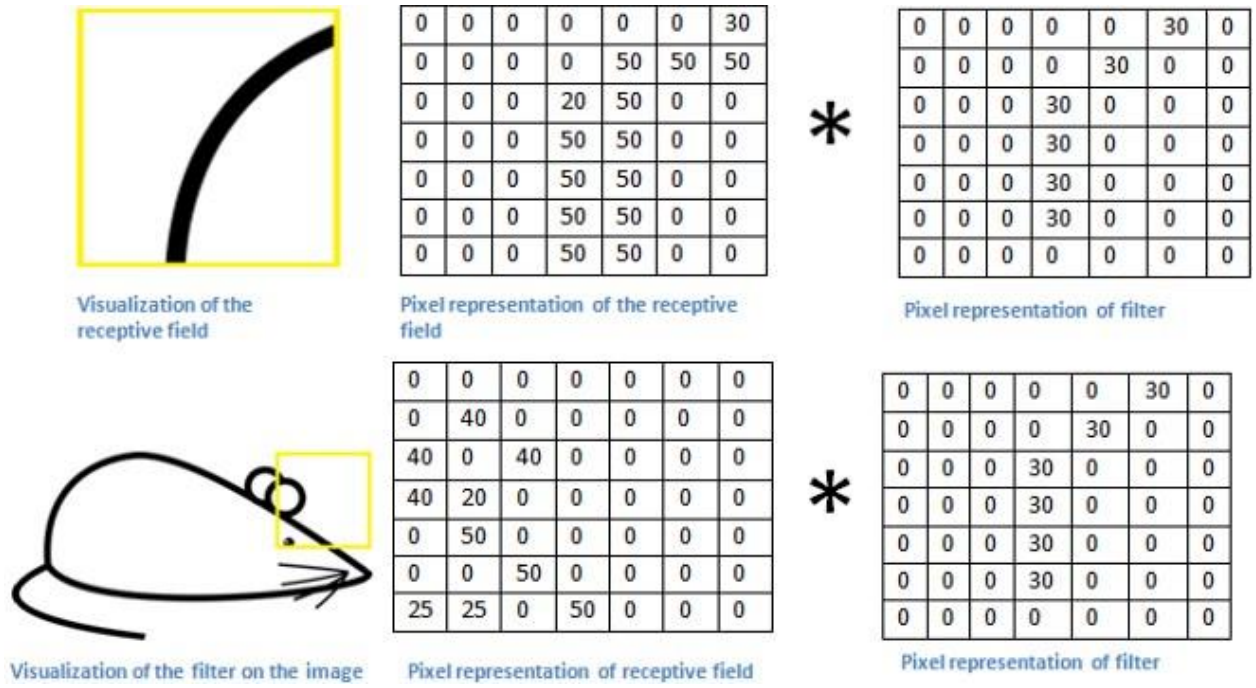

Рисунок 1.4 – Фільтр згорткової нейронної мережі

Елементи матриці, отриманої після операції згортки, складаються з величиною зміщення (bias) та пропускаються через нелінійну функцію активації. Оскільки вхідні дані (наприклад, рукописні цифри) нелінійні, модель повинна це врахувати. Часто в якості такої функції використовують ReLU.

Зазвичай у згортковому шарі використовується декілька фільтрів. Ознаки, отримані в результаті операції згортки, об’єднуються, формуючи n-вимірну матрицю.

Наступний після згорткового шар – шар підвибірки (pooling layer). З метою прискорення процесу навчання та зменшення обсягу пам’яті, що споживає нейронна мережа, виконують операцію зниження розмірності (downsampling) вхідних даних (матриці ознак). Існує два способи зниження розмірності: максимальне об’єднання (max pooling) та середнє об’єднання (average pooling). Під час операції об’єднання уздовж вхідних даних переміщається так зване вікно просіювання. З даних, що потрапляють в його «поле зору», знаходиться максимальне чи середнє значення, яке переміщається в результуючу матрицю. На рисунку 1.5 продемонстровані операції максимального та середнього об’єднання.

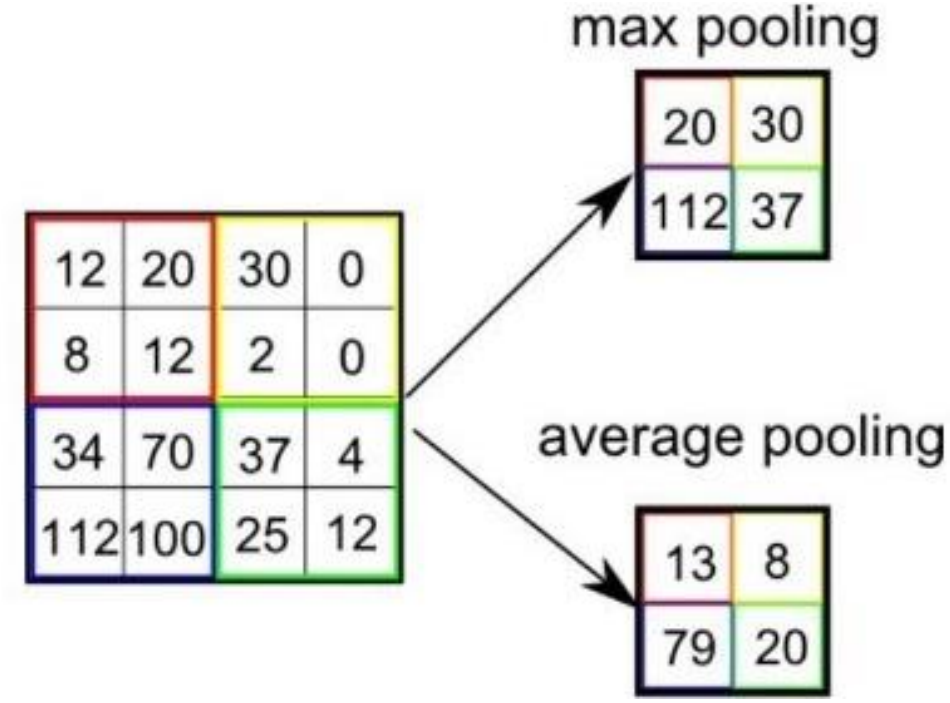

Рисунок 1.5 – Операції максимального та середнього об’єднання

Перевагою операції об’єднання є те, що воно «змушує» нейронну мережу зосередитись на декількох нейронах замість усіх, що має регуляризуючий ефект, зменшуючи вірогідність перенавчання (overfitting).

Після згорткових шарів та шарів підвибірки йде один чи декілька повністю пов’язаних шарів (fully-connected layer). Отримані в результаті проходження через попередні шари ознаки (матриці) розгортаються у довгий вектор, який проходить через декілька повністю пов’язаних шарів. У кожному такому шарі вектор ознак множиться на ваги шару, підсумовується зі значенням зміщення та проходить через нелінійну функцію активації.

Останній шар згорткової нейронної мережі – вихідний (output layer). Вихідний шар відповідає за обчислення вірогідності приналежності вхідних даних до певного класу об’єкта, наприклад, рукописної цифри «3». Кількість нейронів у вихідному шарі зазвичай дорівнює кількості класів об’єктів, які ми хочемо розпізнати. Виходи останнього повністю пов’язаного шару пропускаються через функцію активації, наприклад Softmax, яка формує вектор ймовірностей приналежності вхідного зображення до того чи іншого класу об’єктів.

1.6 Неглибокі (shallow) нейронні мережі

Неглибокі моделі, як і глибокі нейронні мережі, теж популярні і корисні інструменти. Наприклад, word2vec група неглибоких двошарових моделей, яка використовується для створення векторних уявлень слів (word embeddings). Представлена в Efficient Estimation of Word Representations in Vector Space, word2vec приймає на вході великий корпус тексту і створює векторний простір. Кожному слову в цьому корпусі приписується відповідний вектор в цьому просторі. Відмітна властивість – слова із загальних текстів в корпусі розташовані близько один до одного в векторному просторі.

1.7 Фільтрація зашумлених зображень

Попередня обробка зображення – процес поліпшення якості зображення, що ставить за мету отримання на основі оригіналу максимально точного і адаптованого для автоматичного аналізу зображення.

Серед дефектів цифрового зображення можна виділити наступні види:

– цифровий шум;
– кольорові дефекти (недостатні або надлишкові яскравість і контраст, неправильний колірний тон);
– розмитість (расфокусировка).

Методи попередньої обробки зображень залежать від завдань досліджень і можуть включати наступні види робіт:

– фільтрація зашумленних зображень;
– корекція яскравості і контрасту.

Цифровий шум зображення – дефект зображення, внесений фотосенсором і електронікою пристроїв, які їх використовують. Для його придушення використовують такі методи:

– лінійне усереднення точок по сусідах;
– розмиття по Гаусу;
– медіанна фільтрація;
– морфологічні перетворення.

1.8 Ефективність підходів глибинного навчання до розпізнавання об’єктів

Традиційним підходом комп’ютерного зору до розпізнавання об’єктів є використання дескрипторів ознак [9]. До появи глибинного навчання для вирішення таких задач, як класифікація зображень, виконувався крок вилучення ознак. Ознаки – це невеликі інформативні області на зображеннях, які

однозначно описують той чи інший об’єкт. На цьому кроці можуть бути задіяні декілька алгоритмів комп’ютерного зору: виявлення країв, виявлення кутів чи сегментація порогових значень. Вилучені із зображень ознаки формують визначення (опис) об’єктів кожного класу. Дескриптор ознак намагається знайти ці визначення на зображеннях, які подаються йому на вхід. Якщо значна кількість певних ознак присутня на вхідному зображенні, він класифікує його як таке, що містить об’єкт певного класу. Наприклад, для дескриптора ознак на основі гістограм напрямлених градієнтів (HOG) ознаками є розподіл градієнтних векторів (їх напрям та величина) на зображенні.

Складність традиційного підходу полягає в тому, що інженер повинен обрати, які саме ознаки найкраще описують той чи інший клас об’єктів. Це тривалий процес спроб та помилок. Зі збільшенням кількості класів об’єктів вилучення ознак стає більш громіздким. Більше того, процес вилучення ознак використовує безліч параметрів, кожен із яких повинен бути точно налаштований інженером.

Глибинне навчання – це концепція наскрізного навчання, у якій модель глибокої нейронної мережі навчається виявляти основні закономірності різних класів об’єктів на навчальній вибірці. Тобто, нейронна мережа не програмується під конкретну задачу, а навчається її вирішувати, як це роблять люди. Добре встановлено, що глибокі нейронні мережі працюють набагато краще, ніж традиційні алгоритми комп’ютерного зору, хоча і з компромісами щодо обчислювальних вимог та часу, необхідного для навчання моделі. Зважаючи на це, можна констатувати, що процес роботи інженера в області комп’ютерного зору кардинально змінився: знання та досвід у вилученні створених «вручну» ознак об’єктів були замінені знаннями та досвідом вибору чи створення найкращої архітектури нейронної мережі для вирішення задачі.

Розробка архітектури нейронної мережі, яка навчається вилучати ознаки (характеристики) об’єктів на зображенні, а саме згорткової нейронної мережі (Convolutional Neural Network), призвела до прориву у вирішенні задачі класифікації об’єктів. Архітектура згорткової нейронної мережі була описана ще

у 1998 році Яном Лекуном та його командою у статті [10], але її активне використання почалося лише з 2012 року, коли інженери у дослідженні [11] розробили та навчили модель глибокої згорткової нейронної мережі AlexNet класифікувати 1.2 мільйони зображень на 1000 різних класів. Стрімкий розвиток методів глибинного навчання для класифікації, а пізніше – для розпізнавання об’єктів, обумовлений зростанням обчислювальної потужності та збільшенням обсягу даних, необхідних для навчання нейронних мереж. Принцип роботи згорткової нейронної мережі розглянутий у розділі 2 кваліфікаційної роботи.

1.9 Висновки з розділу 1

В данному розділі було розглянуто загальну інформацію про глибинні нейронні мережі, їх структуру та випадки, в яких їх доручно використовувати. Плюсами таких мереж можна вважати навчання без учителя, а також розпізнавання більш глибоких, деколи неочевидних, закономірностей в данних. В таких мережах за кожен рівень ознак відповідає свій шар і вони чудово себе показують в задачах розпізнавання образів на зображенні та в задачах розпізнавання голосу. Недоліками таких мереж є перенавчання і великий час обчислень. Зрозуміло, що для вирішення задач, поставлених в межах дослідження данної дисертації доцільно використовувати саме CNN, адже вони чудово себе показують в задачах розпізнавання образів на зображеннях.

2 ЗАГАЛЬНІ МЕТОДИ РОЗРОБКИ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ РОЗПІЗНАВАННЯ ДОРОЖНИХ ЗНАКІВ

2.1 Особливості та переваги використання мови програмування Python

Python проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм, ніж shell. З іншого боку, вона краще за C обробляє помилки, і, будучи мовою дуже високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на C потребує значних витрат часу. Завдяки більш загальним типам даних, Python застосовують до більш широкого кола задач, ніж Awk і навіть Perl, у той ж час багато речей на мові Python робляться настільки ж просто. Python дозволяє розбивати програми на модулі, що потім можуть бути використані в інших програмах. Python поставляється з великою бібліотекою стандартних модулів, які можна використовувати як основу для нових програм або як наприклад при вивченні мови. Стандартні модулі надають засоби для роботи з файлами, системними викликами, мережевими з'єднаннями і навіть інтерфейсами до різних графічних бібліотек.

Python – інтерпретована мова, що дозволяє заощадити значну кількість часу, що зазвичай витрачається на компіляцію. Інтерпретатор можна використовувати інтерактивно, що дозволяє експериментувати з можливостями мови, писати шаблони програм або тестувати функції при розробці «знизу-вверх». Він також зручний як настільний калькулятор. Python дозволяє писати дуже компактні й зручні для читання програми. Програми, написані мовою Python, звичайно значно коротші еквівалента на C або C++ з декількох причин:

– типи даних високого рівня дозволять Вам виразити складні операції однією інструкцією;

– групування інструкцій виконується за допомогою відступів замість фігурних дужок;
– немає необхідності в оголошенні змінних.

Python розширювана мова: знання C дозволяє додавати нові функції, що вбудовуються, або модулі для виконання критичних операцій з максимальною швидкістю або написання інтерфейсу до комерційних бібліотек, доступним тільки у двійковій формі. Інтерпретатор мови Python може бути вбудований у програму, написану на C, і використовувати його як розширення або командну мову для цієї програми. Python використовується в даний час десятками тисяч програмістів в усьому світі, і число людей, що використовують його, швидко зростає, подвоюється і потроюється щороку. Python приваблює користувачів з ряду причин. Він використовується для розробки програм і дозволяє провести розробку набагато швидше, ніж традиційні мови типу C, C++ або Java . Ця мова працює однаково добре на Windows, UNIX, Macintosh, і OS/2, може використовуватися, для легкої розробки як малих додатків чи сценаріїв, так і для розгортання великих програм. Python пропонує доступ до могутнього і легкого у використанні комплекту інструментальних засобів графічного інтерфейсу користувача. Традиційні машинні мови типу C і Pascal мають ряд характеристик, наприклад, сувора типізація, базові типи, складні (і звичайно довгі) цикли, і потреба у великих кількостях кодів для виконання відносно малих задач. Java досить новий, але розділяє більшість характеристик, включених у цей перелік. Програмісти, знайомі з традиційними мовами погодяться, що відсутність суворої типізації полегшує роботу з Python.

Прямих методів розроблено небагато (через обмежену можливість їх використання); всі вони умовно можуть бути розділені на дві тісно пов'язані групи: методи на основі шаблонів і методи відображення. Методи на основі шаблонів увазі розбиття областей заданого виду (паралелепіпед, куля, циліндр). Відповідно, для кожного виду області використовується свій шаблон, тобто принцип розміщення вузлів і встановлення зв'язків між ними.

Методи відображення дозволяють перейти від областей строгої

геометричної форми до областей довільного виду. Якщо можливо побудувати взаємооднозначне відображення між заданою областю та областю будь-якої простої геометричної форми, то, розбивши останню, можна відобразити отриману сітку на вихідну область. Недоліком цього підходу, крім складності реалізації, є спотворення сітки при відображенні, яке може істотно знизити якість тріангуляції.

Особливим підкласом прямих методів є методи дроблення, які засновані на принципі подрібнення елементів вже побудованої грубої сітки. У двовимірному випадку ці методи дуже ефективні, оскільки будь який трикутник можна розбити на 4 або 9 однакових і подібних йому трикутників (відповідно, втрати якості при дробленні не буде). На жаль, в тривимірному випадку ці методи майже не застосовні, оскільки тетраедр подібним чином розбити можна (тетраедр можна розбити на менші тетраедри декількома способами, однак при цьому будуть отримані тетраедри гіршої форми, що призведе до зниження якості сітки).

Сітки, отримані прямими методами, є структурованими, тобто їх топологія повністю визначається деяким набором правил. Це означає, що за індексами вузла можна визначити всіх його сусідів, а також обчислити координати. Це важлива властивість дозволяє істотно економити комп'ютерні ресурси.

2.2 Python у порівнянні з іншими мовами

Відмінностей Python від інших мов доволі багато, перерахуємо основні з них:

– керування пам'яттю – цілком автоматичне – не потрібно хвилюватися щодо розподілу або звільнення пам'яті. Немає загрози «небезпечного посилання». Java – єдина мова, що пропонує таку концепцію;
– типи зв'язані з об'єктами, а не зі змінними. Це означає, що змінній може бути призначене значення будь-якого типу, і що (наприклад) масив

може містити об'єкти різних типів. Традиційні мови не надають такої можливості;

– операції звичайно виконуються в більш високому рівні абстракції. Це частково результат того, як написана мова, і частково результат розширеної стандартної бібліотеки кодів, що поставляється разом з Python.

Ці та інші особливості Python роблять розгортання додатків надзвичайно швидким. Продуктивність створеного додатку залежить від його особливостей. Звичайно, для чисельного алгоритму, що виконує звичайну арифметику цілого числа в циклі 'for', неважливо, на якій мові він написаний. Але для «середнього» додатка, збільшення продуктивності може бути просто дивовижним.

Один недолік Python, у порівнянні з найбільш традиційними мовами, полягає в тому, що це – не цілком компільована мова; замість цього, вона частково транслює програму до внутрішньої форми байт-коду, і цей байт-код виконується інтерпретатором Python. Однак, у перспективі – сучасні комп'ютери мають так багато не використовуваного обчислювального потенціалу, що для 90% додатків швидкодія зв'язана з вибором мови. Java теж компілюється в байт-код, але в даний час працює повільніше ніж Python у більшості випадків. Крім того, дуже просто об'єднати Python з модулями, написаними на C або C++, які можна використовувати, щоб збільшити швидкість роботи програм в критичних ділянках.

2.3 Огляд бібліотек і фреймворків, що реалізують алгоритми глибинного навчання

Одним із розділів машинного навчання є глибинне навчання (Deep Learning). Глибинне навчання займається розробкою алгоритмів для навчання на основі ознак даних із використанням багатошарових графів, що отримали назву

штучних нейронних мереж. Архітектури глибинного навчання використовуються в таких прикладних задачах:

- розпізнавання мовлення та обробка природної мови;
- комп’ютерний зір;
- механізми фільтрації контенту в соціальних мережах;
- машинний переклад;
- пошукові алгоритми;
- біоінформатика;
- розробка ліків тощо.

Штучна нейронна мережа – математична модель, принцип роботи якої нагадує роботу мережі біологічних нейронів. Кожен шар представлений множиною вузлів – штучних нейронів, які видобувають ознаки все більшого рівня, поки останній шар не скомбінує ці ознаки, щоб зробити передбачення. Штучний нейрон – це модель, яка складається із шару вхідних сигналів , кожен із яких має вагу. Ваги відображають значимість кожного входу штучного нейрону. На основі вхідних активацій та їх ваг у вихідний шар обчислюється сумуюча функція, яка враховуючи вхідні сингали та їх ваги обчислює деяке значення. Далі результат сумуючої функції передається в активаційну функцію, яка в залежності від значення сумуючої функції обчислює вихідний результат нейрона.

Однією із необхідних умов ефективної роботи нейронної мережі є її паралельність обчислень. Послідовні обчислення центрального процесора персонального комп’ютера значно сповільнюють роботу нейронної мережі, особливо багатошарової з великою кількістю нейронів. Тому слід розглянути сучасні підходи при розробці нейронних мереж (як правило, дані підходи поєднуються):

- використання апаратних розподілених систем – нейрокомп’ютерів, графічних процесорів;
- створення паралельних обчислювальних систем на базі персональних комп’ютерів;

– використання хмарних сервісів.

Широке використання графічних процесорів для паралельних обчислень та створення відповідних додатків стало можливе завдяки технології CUDA – платформі для паралельних обчислень на графічних процесорах від Nvidia. Для реалізації задач глибинного навчання розроблена бібліотека cuDNN, що забезпечує такі стандартні процедури нейронних мереж як пряме поширення, метод зворотного поширення помилки, об'єднуючий (pooling), нормуючий (normalization) та активаційний (activation) шари. Дана бібліотека підтримує такі фреймворки як TensorFlow, Caffe2, MATLAB, Microsoft Cognitive Toolkit, Theano, PyTorch.

Використання графічних процесорів з підтримкою технології CUDA для проведення досліджень з паралельних обчислень та зокрема глибинного навчання є хорошим рішенням. Проте доцільно звернути увагу на альтернативну можливість, наприклад, використання хмарних сервісів. Лідерами в напрямку хмарних технологій, зокрема і в проектуванні нейронних мереж є Google Cloud Platform, AWS, Nvidia GPU Cloud.

2.4 Фреймворк Keras

Keras – це відкритий фреймворк для глибинного навчання, написаний на мові програмування Python, що є надбудовою над бібліотекою TensorFlow. TensorFlow – це відкрита бібліотека для машинного навчання від компанії Google, написана на мові програмування C++. Головним розробником Keras є Франсуа Шоллє, інженер із Google. При розробці фреймворка головний акцент був зроблений на надання дослідникам можливостей для швидкого експериментування.

Фреймворк Keras можна інсталювати на Linux, Windows чи macOS. Як і Darknet, він дозволяє навчати нейронну мережу на декількох графічних процесорах одночасно. Основні структури даних, які використовує Keras – це шари

(layers) та моделі (models). Найпростішою моделлю є Sequential, яка представляє собою послідовність із шарів нейронної мережі. Фреймворк надає безліч готових шарів нейронної мережі: згортковий шар, шар підвибірки, повністю пов’язаний шар тощо. Для побудови більш складних архітектур Keras надає Functional API, який дозволяє будувати графи із шарів та створювати власні моделі (models). Keras включає у себе навчені моделі нейронних мереж (MobileNet [22], VGG [29] тощо), а також деякі датасети для різних задач (датасет рукописних цифр MNIST, датасет з відгуками про фільми з сайту IMDB тощо).

2.5 Фреймворк PyTorch

PyTorch – це відкритий фреймворк для машинного навчання, написаний на мовах програмування C, C++ та Python, що розроблений на базі бібліотеки Torch. Torch – це відкрита MATLAB-подібна бібліотека для мови програмування Lua, що реалізує велику кількість алгоритмів для глибинного навчання. PyTorch розроблюється лабораторією штучного інтелекту компанії Facebook.

Фреймворк PyTorch можна інсталювати на Linux, Windows чи macOS. PyTorch забезпечує дві основні функціональності: матричні обчислення з прискоренням на графічних процесорах та використання глибоких нейронних мереж, побудованих на базі системи автоматичного диференціювання. Фреймворк складається з трьох модулів: autograd, optim та nn. Модуль autograd включає у себе систему автоматичного диференціювання, модуль optim реалізує алгоритми оптимізації, які використовуються при побудові нейронних мереж, а модуль nn надає високорівневі абстракції над модулем autograd. Як і Darknet та Keras, PyTorch включає у себе навчені моделі нейронних мереж (AlexNet [11], SqueezeNet [27] тощо). Як і Keras, фреймворк включає у себе датасети для різних задач (COCO [20], датасет рукописних цифр MNIST тощо).

2.6 MatConvNet

MatConvNet – це реалізація з відкритим кодом конволюційних нейронних мереж (CNN) з глибокою інтеграцією в середовище MATLAB. Панель інструментів розроблена з акцентом на простоту та гнучкість. Це розкриває будівельні блоки CNN як прості у використанні функції MATLAB, забезпечуючи підпрограми для обчислення згортків з банками фільтрів, об'єднання функцій, нормалізацію та набагато більше. MatConvNet можна легко розширити, часто використовуючи лише код MATLAB, що дозволяє швидко прототипувати нові CNN архітектури. У той же час він підтримує ефективність обчислення на процесорі та графічному процесорі, що дозволяють тренувати складні моделі на великих датасетах, таких як ImageNet ILSVRC, що містять мільйони навчальних прикладів.

2.7 Neon

Заявляється розробниками як найшвидший фреймворк для CNN і глибинного навчання з підтримкою обчислень на GPU і CPU. Фронтенд виконаний на мові Python, в той час як самі алгоритми реалізовані на спеціально розробленому шейдерний асемблері. Розроблено компанією Nervana Systems, яка була куплена Intel.

2.8 TensorFlow

TensorFlow – відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для вирішення завдань побудови і тренування нейронної мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття. Застосовується як для досліджень, так

і для розробки власних продуктів Google. Основний API для роботи з бібліотекою реалізований для Python, також існують реалізації для C Sharp, C ++, Haskell, Java, Go і Swift. Підтримує обчислення на CPU, GPU і спеціально розробленими компанією Google TPU (Tensor processing units).

2.9 AlexNet

Перша робота, яка популяризувала згорткові нейронні мережі в напрямку комп'ютерного зору, розроблена Alex Krizhevsky, Ilya Sutskever і Geoff Hinton. AlexNet була протестована на змаганні ImageNet ILSVRC в 2012 році і значно випередила конкурента на другому місці (відсоток помилок: 16 % проти 26 %). Архітектура мережі була дуже схожа на архітектуру LeNet, але була глибше, більше і використовувала послідовності згорткових шарів (до цього було стандартною практикою використовуватися тільки комбінацію, в якій вісля згорткового шару відразу слідував шар підвибірки).

2.10 ResNet

Residual Network розроблена Каймінгом Хе і іншими, стала переможцем ILSVRC. Інтенсивно використовує пакетну нормалізацію і спеціальні skipconnections. В кінці немає повнозв'язних шарів. ResNet на сьогодні є справжнім витвором мистецтва в світі CNN і використовується найбільш часто. ЇЇ архітектрура показана на рисунку 2.1.

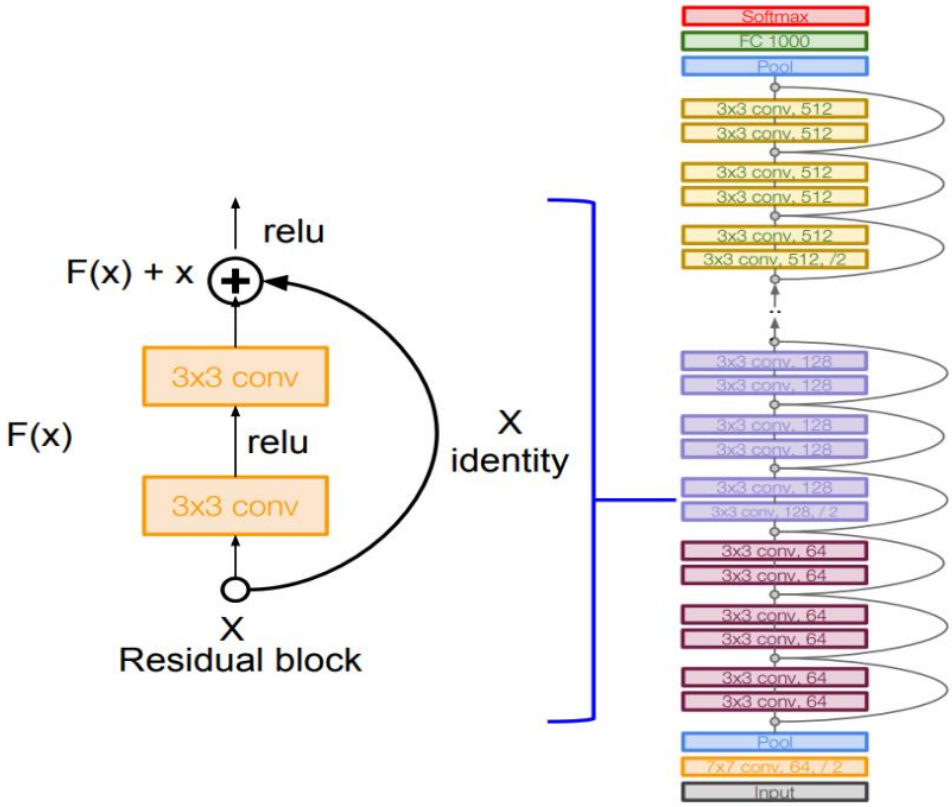

Рисунок 2.1 – Архітектура ResNet

2.11 GoogLeNet

GoogLeNet збільшив очікувану середню точність виявлення об'єктів до 0.439329, і знизив похибку класифікації до 0.06656, найкращого результату на той момент. Його мережа застосовувала понад 30 шарів. Число параметрів склало всього 4млн. Ця продуктивність згорткових нейронних мереж у завданнях ImageNet була близькою до людської.

2.12 Моделі для розпізнавання об’єктів у реальному часі

Серед моделей нейронних мереж для розпізнавання об’єктів у реальному часі, які здатні працювати на пристроях з обмеженими ресурсами, були також розглянуті SSD [23] та SqueezeNet [27]. У статті [28] висвітлені основні принципи

їхньої роботи.

Детектор SSD є своєрідним продовженням детектора YOLO [15], оскільки він перейняв від нього багато принципів роботи (наприклад, алгоритм non-maximum suppression). Відмінною особливістю SSD є розпізнавання об’єктів за один прохід за допомогою заданої сітки вікон на піраміді зображень (різна розмірність зображень). Піраміда зображень закодована у виходах при послідовних операціях згортки та підвибірки. У такий спосіб SSD розпізнає як великі, так і маленькі об’єкти за один прохід. В оригінальній статті [23] для вилучення ознак дослідники використовували модель VGG [29], яка потребує багато обчислювальних ресурсів через велику кількість параметрів. Використання MobileNet у якості блоку для вилучення ознак дозволяє SSD працювати на пристроях з обмеженими ресурсами, однак такий підхід програє за точністю моделі YOLOv4-tiny, як було описано вище.

SqueezeNet – це маленька, але точна модель. Вона, як і MobileNet, може бути використана в якості блоку для вилучення ознак на зображенні та працювати на мобільних пристроях. Відмінною особливістю SqueezeNet є те, що дані спочатку стискаються до чотирьох 1×1 згорткових фільтрів, а потім розширюються до чотирьох 1×1 і чотирьох 3×3 згорткових фільтрів. Одна ітерація стиснення-розширення називається Fire Module. Модель SqueezeNet досягає точності AlexNet, використовуючи при цьому у 50 разів менше параметрів. Команда у дослідженні [30] займалася розробкою моделі Tinier-YOLO – покращеної версії YOLOv3-tiny, яка використовує ідеї SqueezeNet (Fire Module) та має точність 34 % на датасеті COCO [20], програючи YOLOv4-tiny.

2.13 Датасет Open Images

Open Images (The Open Images Dataset V4) [31] – це масштабний датасет, що містить близько 9.2 мільйонів анотованих зображень для навчання нейронної мережі класифікувати об’єкти, розпізнавати об’єкти чи виявляти візуальні

стосунки між об’єктами. Open Images V4 включає у себе 19.8 тисяч класів об’єктів для класифікації, 600 класів об’єктів для розпізнавання та 57 класів об’єктів для виявлення візуальних стосунків. Зображення містять складні сцени з у середньому 8 об’єктами. Одним із класів об’єктів, як і в COCO, є клас Person – «людина». Зображення для задачі розпізнавання об’єктів мають наступні атрибути: GroupOf (група із більше, ніж 5 об’єктів одного класу, які сильно перекривають один одного), Partially occluded (об’єкт частково перекритий іншим об’єктом), Truncated (частина об’єкта виходить за рамки зображення), Depiction (не справжній об’єкт, наприклад, статуя), Inside (зображення зроблено усередині об’єкта, наприклад, салону авто).

Завантажити датасети для навчання, валідації та тестування можна з офіційного сайту Open Images. Зображення, а також анотації для кожної з перерахованих вище задач завантажуються окремо. Для полегшення цього процесу окремі розробники створили програмний засіб – OIDv4 ToolKit. Він дозволяє завантажувати зображення окремих категорій у потрібній кількості з відповідного датасета, завантажувати анотації до зображень, обирати директорію для завантаження, фільтрувати зображення за атрибутами (GroupOf, Truncated та інші) тощо. Даний засіб можна використовувати тільки для задач класифікації та розпізнавання. Не менш важливим є те, що завантаження відбувається у різних потоках, що значно пришвидшує цей процес. Також засіб надає можливість відображати завантажені зображення із анотованими об’єктами.

2.14 Висновки з розділу 2

У цьому розділі подано опис популярних фреймворків, розглянуто їх плюси тамінуси. Також розглянуті найбільш популярні бібліотеки для роботи з зображеннями, їх переваги та недоліки. Та зроблено огляд датасетів з наборами даних для вирішення поставленої задачі.

3 РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ ДЛЯ РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ

3.1 Опис задачі

Для зображення, що складається з дорожнього знака, необхідно спрогнозувати рамку навколо дорожнього знака і визначити тип дорожнього знака. Ці знаки можуть належати чотирма різним класам:

- світлофор;
- стоп;
- обмеження швидкості;
- пішохідний перехід.

Це називається багатозадачне завдання навчання, оскільки вона включає виконання двох завдань: регресія для знаходження координат bounding box'a, класифікація визначення типу дорожнього знака.

Датасет складається із 877 зображень (рис. 3.1). Це досить незбалансований набір даних, більшість зображень відноситься до класу обмеження швидкості.

Описи кожного зображення зберігаються в окремих файлах XML. Зробимо такі кроки для підготовки даних для тренування:

1) пройдемося по директорії annotations щоб отримати всі файли .xml;
2) прочитаємо потрібну нам інформацію з кожного файлу .xml, використовуючи xml.etree.ElementTree;
3) творимо словник, що містить filepath (шлях до картинки), width, height, (xmin, xmax, ymin, ymax) (кординати bounding box'a) і class і додамо словник до списку;
4) створимо pandas dataframe, використовуючи список словників з минулого пункту.

```
images_path = Path('../input/road-sign-detection/images')
anno_path = Path('../input/road-sign-detection/annotations')


def filelist(root, file_type):
    return [os.path.join(directory_path, f) for directory_path, directory_name,
            files in os.walk(root) for f in files if f.endswith(file_type)]
def generate_train_df (anno_path):
    annotations = filelist(anno_path, '.xml')
    anno_list = []
    for anno_path in annotations:
        root = ET.parse(anno_path).getroot()
anno = {}
        anno['filename'] = Path(str(images_path) + '/'+ root.find("./filename").text)
        anno['width'] = root.find("./size/width").text
        anno['height'] = root.find("./size/height").text
        anno['class'] = root.find("./object/name").text
        anno['xmin'] = int(root.find("./object/bndbox/xmin").text)
        anno['ymin'] = int(root.find("./object/bndbox/ymin").text)
        anno['xmax'] = int(root.find("./object/bndbox/xmax").text)
        anno['ymax'] = int(root.find("./object/bndbox/ymax").text)
        anno_list.append(anno)
    return pd.DataFrame(anno_list)
```

In [3]:
```
df_train = generate_train_df(anno_path)
df_train
```

Out[3]:

	filename	width	height	class	xmin	ymin	xmax	ymax
0	../input/road-sign-detection/images/road732.png	300	400	speedlimit	99	97	159	157
1	../input/road-sign-detection/images/road518.png	300	400	speedlimit	157	165	184	191
2	../input/road-sign-detection/images/road717.png	300	400	speedlimit	90	219	109	240
3	../input/road-sign-detection/images/road362.png	300	400	speedlimit	82	164	151	232
4	../input/road-sign-detection/images/road492.png	300	400	speedlimit	110	137	182	208
...	...	...	...	...	...	...	...	...
872	../input/road-sign-detection/images/road248.png	300	400	speedlimit	167	205	192	229
873	../input/road-sign-detection/images/road227.png	300	400	speedlimit	141	181	153	193
874	../input/road-sign-detection/images/road660.png	300	400	speedlimit	132	189	193	250
875	../input/road-sign-detection/images/road620.png	300	400	speedlimit	123	222	150	250

Рисунок 3.1 – Дорожні знаки з датасету

In [4]:
```
df_train['class'].value_counts()
```

Out[4]:
```
speedlimit      652
crosswalk        88
stop             76
trafficlight     61
Name: class, dtype: int64
```

Рисунок 3.2 – Розподіл знаків за категоріями

Далі необхідно перетворити мітки у класи.

In [5]:
```
class_dict = {'speedlimit': 0, 'stop': 1, 'crosswalk': 2, 'trafficlight': 3}
df_train['class'] = df_train['class'].apply(lambda x:  class_dict[x])

print(df_train.shape)
df_train.head()
```

Рисунок 3.3 – Створення класів

Out[5]:

	filename	width	height	class	xmin	ymin	xmax	ymax
0	../input/road-sign-detection/images/road732.png	300	400	0	99	97	159	157
1	../input/road-sign-detection/images/road518.png	300	400	0	157	165	184	191
2	../input/road-sign-detection/images/road717.png	300	400	0	90	219	109	240
3	../input/road-sign-detection/images/road362.png	300	400	0	82	164	151	232
4	../input/road-sign-detection/images/road492.png	300	400	0	110	137	182	208

Рисунок 3.4 – Таблиця з класифікацією зображень

3.2 Дорожні знаки

На даний момент в Україні дорожні знаки розділені на вісім груп:попереджувальні знаки;

- знаки пріоритету;
- заборонні знаки;
- розпорядчі знаки;
- знаки особливих приписів;
- інформаційні знаки;
- знаки сервісу;
- знаки додаткової інформації;

Розглянемо основні з них.

Основними знаками в Україні є заборонні дорожні знаки. Заборонні дорожні знаки – забороняють водієві здійснити будь-яку дію, що пропонує на знак. Найчастіше аварійні ситуації трапляються через, недотримання цих знаків. Прикладом будуть знаки обмеження швидкості, заборони обгону, в'їзду, стоянки і зупинки. Як відомо, червоний колір відразу ж звертає на себе увагу, тому що забороняють знаки мають круглу форму і яскраво червоний колір, для того щоб водій зміг його побачити в будьякому стані і погодних умовах. На рисунку 3.5 можна побачити приклад заборонних дорожніх знаків:

Рисунок 3.5 – Приклад заборонних знаків

Попереджувальні дорожні знаки (рис. 3.6) – основною функцією цих знаків є попередження водіїв про небезпеку, і застерігати від дорожньо-транспортних пригод. Такі знаки зазвичай встановлюють на небезпечних ділянках дороги. В основі знака в більшості випадків лежить трикутник з червоним обведенням.

Рисунок 3.6 – Приклад попереджувальних знаків

Розпорядчі дорожні знаки – використовуються для того, щоб показувати водіям єдино дозволений дію, будь то проїзд тільки прямо, або тільки поворот наліво (але в цьому випадку дозволено і розворот). Такі знаки дозволяють виконувати тільки певні дії тільки певним учасникам дорожнього руху. На рисунку 3.7 можна побачити приклад розпорядчих знаків.

Рисунок 3.7 – Приклад розпорядчих знаків

Знаки пріоритету – встановлюють черговість проїзду перехресть, перехрещень доріг або вузьких ділянок дороги. Знаки особливих розпоряджень можуть поєднувати елементи як розпорядчих, так і заборонних знаків. Приклад знаків особливих приписів зображено на рисунку 3.8.

Рисунок 3.8 – Приклад знаків особливих розпоряджень

Розпізнавати буде найрозумніше саме ці групи знаків тому, що саме ці знаки мають істотний вплив на дорожню ситуацію. У той же час, з перших п'яти груп найбільшу важливість представляють забороняють дорожні знаки, як, наприклад, знаки обмеження швидкості або заборони обгону. Саме ігнорування

заборонних дорожніх знаків, згідно зі статистикою, є найчастішою причиною виникнення ДТП.

3.3 Розробка методу розпізнавання об'єктів на зображенні

Оскільки для навчання моделі комп'ютерного зору зображення повинні мати однаковий розмір, нам необхідно змінити розмір наших зображень та відповідних обмежуючих рамок. Змінити розмір зображення просто, але змінити розмір bounding box'a трохи складніше, тому що кожен прямокутник залежить від зображення та його розмірів. Перетворимо bounding box на зображення (маску) того ж розміру, що і відповідне цьому прямокутнику зображення. Ця маска буде просто мати 0 для заднього фону та 1 для області покриваючої bounding box. Спочатку прочитаємо зображення:

```
def read_image(path):
    return cv2.cvtColor(cv2.imread(str(path)), cv2.COLOR_BGR2RGB)

def create_mask(bb, x):
    """Створюємо маску для bounding box'a такого ж шейпу як і зображення"""
    rows,cols,*_ = x.shape
    Y = np.zeros((rows, cols))
    bb = bb.astype(np.int)
    Y[bb[0]:bb[2], bb[1]:bb[3]] = 1.
    return Y
def mask_to_bb(Y):
    """Конвертуємо маску Y в bounding box'a, приймаючи 0 як фоновий
ненульовой об’єкт """
    cols, rows = np.nonzero(Y)
    if len(cols) == 0:
```

```
        return np.zeros(4, dtype=np.float32)
    top_row = np.min(rows)
    left_col = np.min(cols)
    bottom_row = np.max(rows)
    right_col = np.max(cols)
    return np.array([left_col, top_row, right_col, bottom_row], dtype=np.float32)
def create_bb_array(x):
    """Генерируємо масив bounding box'а з стовпця train_df"""
    return np.array([x[5],x[4],x[7],x[6]])

def resize_image_bb(read_path, write_path, bb, sz):
    """Ресайзимо зображення та його bounding box та записуємо зображення в новий шлях"""
    im = read_image(read_path)
    im_resized = cv2.resize(im, (sz, sz))
    Y_resized = cv2.resize(create_mask(bb, im), (sz, sz))
    new_path = str(write_path/read_path.parts[-1])
    cv2.imwrite(new_path, cv2.cvtColor(im_resized, cv2.COLOR_RGB2BGR))
    return new_path, mask_to_bb(Y_resized)
```

Застосуємо всі наші функції:

```
IM_SIZE = 300
new_paths = []
new_bbs = []
train_path_resized = Path('./images_resized')
Path.mkdir(train_path_resized, exist_ok=True)

for index, row in df_train.iterrows():
```

```
    new_path,new_bb = resize_image_bb(row['filename'], train_path_resized,
create_bb_array(row.values), IM_SIZE)
    new_paths.append(new_path)
    new_bbs.append(new_bb)
df_train['new_path'] = new_paths
df_train['new_bb'] = new_bbs

df_train.head()
```

Out[8]:

filename	width	height	class	xmin	ymin	xmax	ymax	new_path	new_b
../input/road-sign-detection/images/road732.png	300	400	0	99	97	159	157	images_resized/road732.png	[72.0, 99.0, 117.0, 158.0]
../input/road-sign-detection/images/road518.png	300	400	0	157	165	184	191	images_resized/road518.png	[123.0 157.0, 143.0, 183.0]
../input/road-sign-detection/images/road717.png	300	400	0	90	219	109	240	images_resized/road717.png	[164.0 90.0, 179.0, 108.0]
../input/road-sign-detection/images/road362.png	300	400	0	82	164	151	232	images_resized/road362.png	[123.0 82.0, 173.0, 150.0]

Рисунок 3.9 – Розподіл зображень зі знаками

Приклад семпла, що вийшов:

```
im = cv2.imread(str(df_train.values[58][0]))
bb = create_bb_array(df_train.values[58])
print(im.shape)

Y = create_mask(bb, im)
mask_to_bb(Y)

plt.imshow(im)
```

```
Out[9]:
        <matplotlib.image.AxesImage at 0x7f2c46c21c10>
```

Рисунок 3.10 – Приклад зображення знаку

3.4 Аугментація даних

Аугментація даних - це метод, що дозволяє краще узагальнити нашу модель шляхом створення нових навчальних зображень за допомогою різних варіантів існуючих зображень. У нашому поточному навчальному наборі всього 800 зображень, тому збільшення даних дуже важливо, щоб наша модель не перенавчилася. Для цього завдання будемо використовувати переворот, поворот, центральне обрізання та випадкове обрізання.

Єдине, що тут потрібно пам'ятати, – це переконатися, що рамка, що обмежує, також трансформується так само, як і зображення. Для цього ми дотримуємося того ж підходу, що і зміна розміру – перетворюємо рамку, що обмежує, в маску, застосовуємо ті ж перетворення до маски, що і вихідне зображення, і витягуємо координати рамки, що обмежує.

```
# Вирізаємо шматок з зображення
def crop(im, r, c, target_r, target_c):
   return im[r:r+target_r, c:c+target_c]
```

```
# Центральне вирізання
def center_crop(x, r_pix=8):
    r, c,*_ = x.shape
    c_pix = round(r_pix*c/r)
    return crop(x, r_pix, c_pix, r-2*r_pix, c-2*c_pix)

def rotate_cv(im, deg, y=False, mode=cv2.BORDER_REFLECT):
    """ Розвертаємо наше зображення"""
    r,c,*_ = im.shape
    M = cv2.getRotationMatrix2D((c/2,r/2),deg,1)
    if y:
return cv2.warpAffine(im, M, (c, r), borderMode=cv2.BORDER_CONSTANT)
    return cv2.warpAffine(im, M, (c, r), borderMode=mode,
flags=cv2.WARP_FILL_OUTLIERS)

def random_cropXY(x, Y, r_pix=8):
    """ Повертає випадкове вирізання"""
    r, c,*_ = x.shape
    c_pix = round(r_pix * c/r)
    rand_r = random.uniform(0, 1)
    rand_c = random.uniform(0, 1)
    start_r = np.floor(2 * rand_r * r_pix).astype(int)
    start_c = np.floor(2 * rand_c * c_pix).astype(int)
    xx = crop(x, start_r, start_c, r - 2*r_pix, c - 2*c_pix)
    YY = crop(Y, start_r, start_c, r - 2*r_pix, c - 2*c_pix)
    return xx, YY
# Трансформуємо нашу картинку
def transformsXY(path, bb, is_transforms):
    x = cv2.imread(str(path)).astype(np.float32)
    x = cv2.cvtColor(x, cv2.COLOR_BGR2RGB) / 255
```

```
    Y = create_mask(bb, x)
    if is_transforms:
        rdeg = (np.random.random()-.50) * 20
        x = rotate_cv(x, rdeg)
        Y = rotate_cv(Y, rdeg, y=True)
        if np.random.random() > 0.5:
            x = np.fliplr(x).copy()
            Y = np.fliplr(Y).copy()
        x, Y = random_cropXY(x, Y)
    else:
        x, Y = center_crop(x), center_crop(Y)
    return x, mask_to_bb(Y)
def create_corner_rect(bb, color='red'):
    bb = np.array(bb, dtype=np.float32)
    return plt.Rectangle((bb[1], bb[0]), bb[3]-bb[1], bb[2]-bb[0], color=color,
                         fill=False, lw=3)

def show_corner_bb(im, bb):
    plt.imshow(im)
    plt.gca().add_patch(create_corner_rect(bb))
```

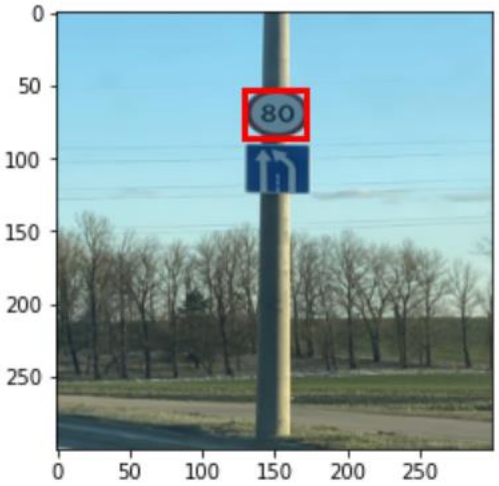

Рисунок 3.11 – Приклад зображення, оригінал

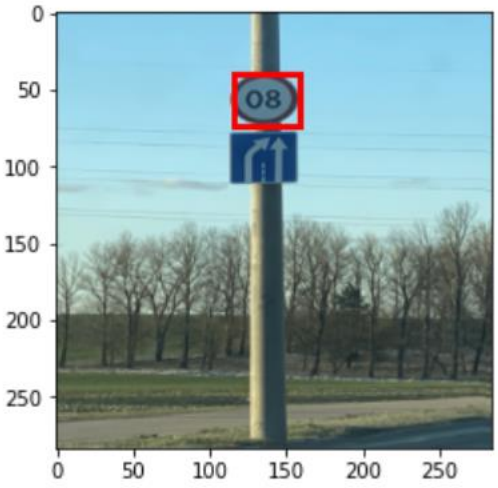

Рисунок 3.12 – Після трансформації

3.5 Датасет

Тепер, коли ми маємо додатки до даних, ми можемо створити набір даних PyTorch. Ми нормалізуємо зображення за допомогою статистики ImageNet, тому що ми будемо використовувати заздалегідь навчену модель ResNet і застосуємо аугментацію даних у нашому наборі даних під час навчання.

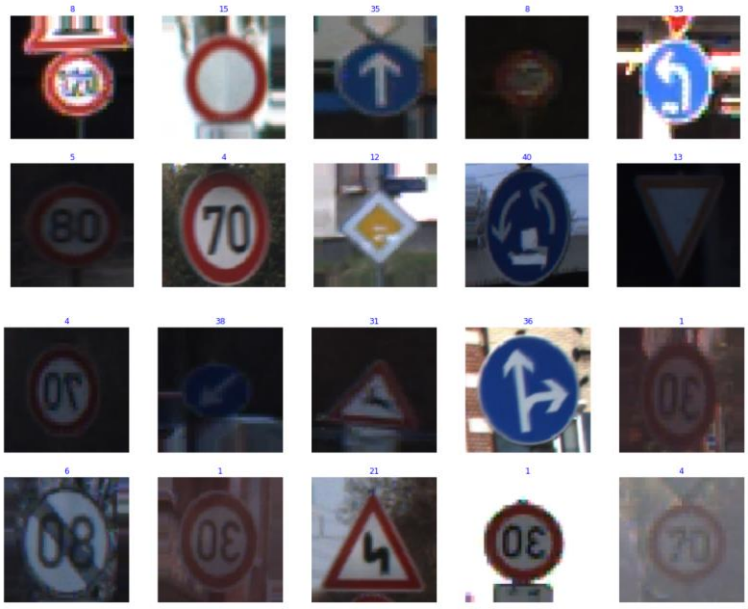

Рисунок 3.13 – Зображення з датасету

```
df_train = df_train.reset_index()
X = df_train[['new_path', 'new_bb']]
Y = df_train['class']
X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size=0.2,
random_state=42)



def normalize(im):
    """Нормалізація даних за допомогою статистики ImageNet"""
    imagenet_stats = np.array([[0.485, 0.456, 0.406], [0.229, 0.224, 0.225]])
    return (im - imagenet_stats[0]) / imagenet_stats[1]
class RoadDataset(Dataset):
    def __init__(self, paths, bb, y, is_transforms=False):
        self.is_transforms = is_transforms
        self.paths = paths.values
        self.bb = bb.values
        self.y = y.values

    def __len__(self):
        return len(self.paths)

    def __getitem__(self, idx):
        path = self.paths[idx]
        y_class = self.y[idx]
        x, y_bb = transformsXY(path, self.bb[idx], self.is_transforms)
        x = normalize(x)
x = np.rollaxis(x, 2)
        return x, y_class, y_bb
```

```
train_ds = RoadDataset(X_train['new_path'], X_train['new_bb'], y_train,
is_transforms=True)
valid_ds = RoadDataset(X_val['new_path'], X_val['new_bb'], y_val)
```

Завантажемо все це в наш даталоадер.

```
batch_size = 16
train_dl = DataLoader(train_ds, batch_size=batch_size, shuffle=True)
valid_dl = DataLoader(valid_ds, batch_size=batch_size)
```

3.6 Визначення моделі

Як модель ми будемо використовувати дуже просту попередньо навчену модель resNet-34. Оскільки тут у нас є два завдання, є два останні шари - регресія рамки, що обмежує, і класифікатор зображень.

```
class BB_model(nn.Module):
    def __init__(self):
        super(BB_model, self).__init__()
        resnet = models.resnet34(pretrained=True)
        layers = list(resnet.children())[:8]
        self.features = nn.Sequential(*layers)
        self.classifier = nn.Sequential(nn.BatchNorm1d(512), nn.Linear(512, 4))
        self.bb = nn.Sequential(nn.BatchNorm1d(512), nn.Linear(512, 4))

    def forward(self, x):
        x = self.features(x)
        x = F.relu(x)
        x = nn.AdaptiveAvgPool2d((1,1))(x)
        x = x.view(x.shape[0], -1)
        return self.classifier(x), self.bb(x)
```

```
resnet = models.resnet34(pretrained=True)
list(resnet.children())[:8]
```

3.7 Навчання та тестування

Для розрахунку втрат нам потрібно взяти до уваги як втрату класифікації, так і втрату регресії прямокутника, що обмежує, тому ми використовуємо комбінацію кросентропії і L1-втрати (сума всіх абсолютних відмінностей між істинним значенням і прогнозованими координатами).

```
model = BB_model().cuda()
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.Adam(params, lr=0.006)
epochs = 15
model
```

```
def train():
    for i in range(epochs):

        model.train()
        total = 0
        sum_loss = 0

        for x, y_class, y_bb in train_dl:
            len_batch = y_class.shape[0]
            x = x.cuda().float()
            y_class = y_class.cuda()
            y_bb = y_bb.cuda().float()
            out_class, out_bb = model(x)
```

```
    # losses
    loss_class = F.cross_entropy(out_class, y_class, reduction="sum")
    loss_bb = F.l1_loss(out_bb, y_bb, reduction="sum")

    loss = loss_class + loss_bb

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    total += len_batch
    sum_loss += loss.item()

train_loss = sum_loss / total

# Eval
model.eval()
val_total = 0
val_sum_loss = 0
correct = 0

for x, y_class, y_bb in valid_dl:
    len_batch = y_class.shape[0]
    x = x.cuda().float()
    y_class = y_class.cuda()
    y_bb = y_bb.cuda().float()

    out_class, out_bb = model(x)
```

```
            loss_class = F.cross_entropy(out_class, y_class, reduction="sum")
            loss_bb = F.l1_loss(out_bb, y_bb, reduction="sum")
            loss = loss_class + loss_bb

            _, pred = torch.max(out_class, 1)
            correct += (pred == y_class).sum().item()

            val_sum_loss += loss.item()
            val_total += len_batch

        val_loss = val_sum_loss / val_total
        val_acc = correct / val_total

        print(f"Epoch [{i+1}/{epochs}]. train_loss {train_loss:.3f} val_loss {val_loss:.3f}
val_acc {val_acc:.3f}")
```

Тестування.

```
# resizing test image
im = read_image('./images_resized/road789.png')
Path.mkdir(Path('./road_signs_test'), exist_ok=True)
cv2.imwrite('./road_signs_test/road789.jpg', cv2.cvtColor(im,
cv2.COLOR_RGB2BGR))
```

```
# test Dataset
test_ds = RoadDataset(
    pd.DataFrame([{'path':'./road_signs_test/road789.jpg'}])['path'],
    pd.DataFrame([{'bb':np.array([0,0,0,0])}])['bb'],
    pd.DataFrame([{'y':[0]}])['y']
)
```

```
x, y_class, y_bb = test_ds[0]


xx = torch.FloatTensor(x[None,])
xx.shape
```

```
# prediction
out_class, out_bb = model(xx.cuda())
out_class, out_bb
```

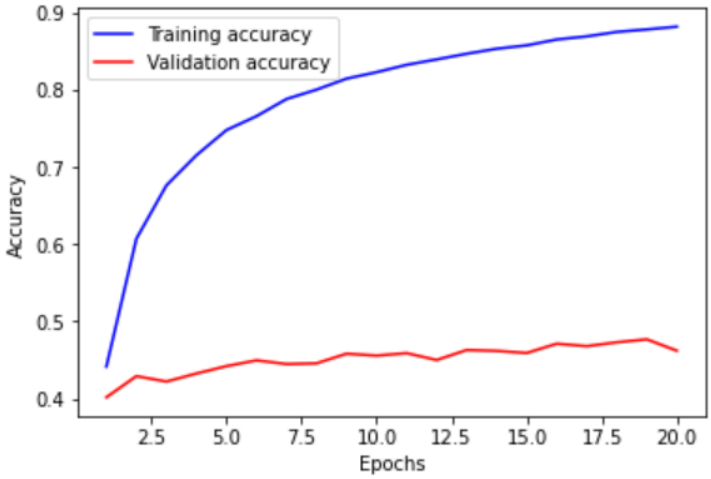

Рисунок 3.14 – Перенавчена модель

Ми бачимо, що мережа перенавчена: точність перевірки значно нижча, ніж у навчальної; однак навіть точність навчання недостатньо велика. Це означає, що немає сенсу застосовувати техніку регуляризації.

Перші шари згорткової мережі виявляють шаблони загального призначення, які корисні для великої різноманітності завдань, тоді як останні шари вивчають шаблони для конкретних завдань. Отже, очевидна ідея спробувати: отримати вихідні дані перших шарів згорткової основи та використати їх як вхідні дані для класифікатора.

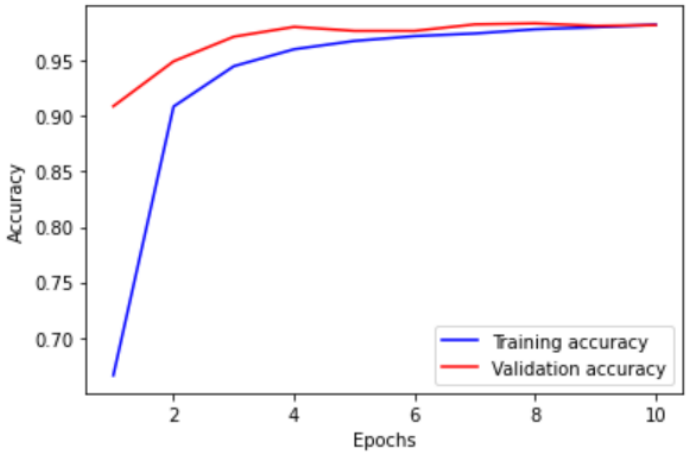

Рисунок 3.15 – Навчання на 10 епохах

Це виглядає краще: 98,2 % точності перевірки та навчання.

ВИСНОВКИ

Досліджена проблема визначення дорожних знаків: розглянуті переваги та недоліки існуючих технологій та систем для її вирішення, сфери застосування таких систем та сучасні підходи комп’ютерного зору до її розв’язання.

Досліджені засоби для вирішення поставленої проблеми – розглянуті моделі нейронних мереж, придатні для розпізнавання дорожних знаків по зображенню знака.

Проведено навчання моделі нейронної мережі для розпізнавання дорожних знаків: налаштоване середовище для навчання, підготовлені зображення знаків з відповідного датасету, створені необхідні для тренування файли та успішно виконано навчання моделі.

Розроблена готова до впровадження комп’ютерна система, яка визначає дорожні знаки по зображенню.

Досліджені результати навчання та точність навчених моделей нейронних мереж: модель, якій було задано 15 епох навчання показала більшу точність при навчанні ніж нейронні мережі з 10 та 20 епохами.

Розроблена штучна нейронна мережа дозволяє прогнозувати дорожні знаки. Абсолютна відсоткова похибка становить не більше 2 %. При цьому виявлено, що точність розробленої системи залежить від якості навчання моделі нейронної мережі, яка у свою чергу залежить від якості датасету.

Поставлена проблема визначення дорожніх знаків повністю вирішена з допустимою точністю.

ПЕРЕЛІК ПОСИЛАНЬ

1. Акулов П. В. Решение задач прогнозирования с помощью нейронных сетей. URL: www.dgtu.donetsk.ua (дата звернення 21.10.2022).

2. Оссовский С. Нейронные сети для обработки информации / пер. с польского И. Д. Рудинского. Москва: Финансы и статистика, 2002. 344 с.

3. Кальченко Д. Нейронные сети: на пороге будущего. *КомпьютерПресс*. 2005. № 1. URL: http //www.compr.ru (дата звернення: 21.10.2022).

4. Rosenblatt F. The Perceptron a perceiving and recognizing automaton: report No. 85-460-1. New York: Cornell Aeronautical Laboratory, 1957. 33 p. URL: https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf (дата звернення: 25.09.2022).

5. McCulloch W. S., Pitts W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*. 1943. Vol. 5, Issue 4. P. 115.

6. Donald O. H. The Organization of Behavior: A Neuropsychological Theory. Wiley, 1949. 335 p.

7. Fukushima K., Miyake S., Ito T. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Transaction on Systems, Man and Cybernetics.* 1983. Vol. SMC-13, no. 5. P. 826–834.

8. Lowe D. Object recognition from local scale-invariant features. URL: https://www.researchgate.net/publication/2373439_Object_Recognition_from_Local_Scale-Invariant_Features (дата звернення: 25.09.2022).

9. Dalal N., Triggs B. Histograms of Oriented Gradients for Human Detection. *CVPR'05*: proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2005. P. 886–893. DOI: 10.1109/CVPR.2005.177.

10. Vapnik, Vladimir N. The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1995. 334 p.

11. Kemelmacher-Shlizerman I., Seitz S. M., Miller D., Brossard E. The MegaFace Benchmark: 1 Million Faces for Recognition at Scale. *CVPR*: proceedings

of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 4873–4882. DOI: 10.1109/CVPR.2016.527.

12. Mahony N., Campbell S., Carvalho A. Deep Learning vs Traditional Computer Vision. *Advances in Computer Vision*: proceedings of the 2019 Computer Vision Conference (CVC). 2019. Vol. 1. P. 128–144. DOI: 10.1007/978-3-030-17795-9.

13. Lecun Y., Bengio Y., Haffner P. Gradient-Based Learning Applied to Document Recognition. *IEEE*: proceedings of the IEEE. 1998. Vol. 86, No. 11. P. 2278–2324. DOI: 10.1109/5.726791.

14. Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*. 2012. Vol. 25, Issue 2. P. 1097.

15. Girshick R., Donahue J., Darrell T., Malik J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *CVPR:* proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. Columbus, 2014. P. 580–587. DOI: 10.1109/CVPR.2014.81.

16. Girshick R. Fast R-CNN. *ICCV:* proceedings of the 2015 IEEE International Conference on Computer Vision. Santiago, 2015. P. 1440–1448. DOI: 10.1109/ICCV.2015.169.

17. Shaoqing R., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2016. Vol. 39, Issue 6. P. 1137.

18. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *CVPR*: proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 779–788. DOI: 10.1109/CVPR.2016.91.

19. Bochkovskiy A., Wang C., Liao H. YOLOv4: Optimal Speed and Accuracy of Object Detection. 2020. URL: https://arxiv.org/pdf/2004.10934.pdf (дата звернення: 25.09.2022).

20. Convolutional Neural Networks from the ground up. URL:

https://to+wardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1 (дата звернення: 18.11.2022).

21. Liu W., Anguelov D., Erhan D. SSD: Single Shot MultiBox Detector. *ECCV 2016*: proceedings of the European Conference on Computer Vision. 2016. P. 21–37.

22. Howard A., Sandler M., Chu G. Searching for MobileNetV3. *ICCV 2019*: proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, 2019. P. 1314–1324.

23. Depthwise Separable Convolution. URL: https://www.youtube.com/watch?v=T7o3xvJLuHk (дата звернення: 20.11.2022).

24. Iandola F., Han S., Moskewicz M. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. 2016. URL: https://arxiv.org/pdf/1602.07360.pdf (дата звернення: 21.10.2022).

25. Нейросети: куда это все движется. URL: https://habr.com/ru/post/482794/ (дата звернення: 21.10.2022).

26. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR 2015*: proceedings of the International Conference on Learning Representations. San Diego, 9 May 2015. URL: https://www.robots.ox.ac.uk/~vgg/publications/2015/Simonyan15/simonyan15.pdf (дата звернення: 18.10.2022).

27. Fang W., Wang L., Ren P. Tinier-YOLO: A Real-time Object Detection Method for Constrained Environments. *IEEE Access*. 2019. Vol. 8. P. 1935.

28. Kuznetsova A., Rom H., Alldrin N. The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale. *International Journal of Computer Vision*. 2020. Vol. 128, Issue 4. P. 1956.

29. Braun M., Krebs S., Flohr F., Gavrila D. EuroCity Persons: A Novel Benchmark for Person Detection in Traffic Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2019. Vol. 41, Issue 8. P. 1844.

30. Fisher Y., Chen H., Wang X. A Diverse Driving Dataset for Heterogeneous Multitask Learning. *CVPR*: proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. P. 2636–2645.