Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Інтелектуальний людино-машинний додаток для перетворення "мова жестів - текст"

Виконала: студентка 2 курсу, групи 2АКІТ-21м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

Марія КОЛЯДИЧ
Ім'я ПРІЗВИЩЕ

Керівник:
д.т.н., доцент, зав. кафедри КСУ
степінь, звання, посада

В'ячеслав КОВТУН
Ім'я ПРІЗВИЩЕ

«16» грудня 2022 р.

Опонент: Доцент каф.АІІТ
степінь, звання, посада

Юрій ІВАНОВ
Ім'я ПРІЗВИЩЕ

«17» грудня 2022 р.

Допущено до захисту
Зав. кафедри КСУ
В'ячеслав КОВТУН
«14» грудня 2022

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ

В'ячеслав КОВТУН

"03" жовтня 2022 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Колядич Марії Вікторівні
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальний людино-машинний додаток для перетворення "мова жестів - текст"
керівник роботи д.т.н., доцент, зав. кафедри КСУ Ковтун В.В.
затверджені наказом ВНТУ від "14" вересня 2022 року № 203
2. Термін подання студентом роботи "12" грудня 2022 року
3. Вихідні дані до роботи: нейронна мережа, вибірка даних (відеозаписи), автокодер, мова програмування *Python/Matlab/Java/C#.*
4. Зміст текстової частини: вступ, аналіз предметної області, розробка математичного апарату нейронної мережі; розробка програмного забезпечення та експериментальні дослідження, економічний розділ, список використаних джерел, висновки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) архітектура мережі; діаграма варіантів використання; робочий процес системи; приклад вихідних даних; лістинг програмного забезпечення.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Ковтун В.В., д.т.н., доцент, зав. кафедри КСУ	03.10.22	08.12.22
4	Небава М.І., професор кафедри ЕПВМ		

7. Дата видачі завдання "03" жовтня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	03.10.22	20.10.22	
2	Розробка математичного апарату нейронної мережі	20.10.22	10.11.22	
3	Розробка програмного забезпечення та експериментальні дослідження	10.11.22	20.11.22	
4	Підготовка економічного розділу	20.11.22	25.11.22	
5	Оформлення пояснювальної записки і графічного матеріалу	25.11.22	07.12.22	
6	Попередній захист роботи	12.12.22	16.12.22	
7	Остаточний захист роботи	20.12.22	23.12.22	

Студентка _____ Марія КОЛЯДИЧ
(підпис) (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи _____ В'ячеслав КОВТУН
(підпис) (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 004.514

Колядич М. В. Інтелектуальний людино-машинний додаток для перетворення “мова жестів - текст”. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інтелектуальні комп’ютерні системи. Вінниця: ВНТУ, 2022. 123 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 39; табл. 5.

У магістерській кваліфікаційній роботі розроблено систему перекладу жестової мови з потокового відео на англійську. У оглядово-аналітичній частині роботи досліджено існуючі методи та підходи до розпізнавання об’єктів, Проаналізовано існуючі архітектури згорткових нейронних мереж. У теоретично-методичній частині розроблено рішення у вигляді нейронної мережі для розпізнавання жестів із потокового відео. Розроблений продукт аналізується та тестується. У економічній частині проаналізований технічний рівень і розрахована собівартість реалізації розробки. Ілюстративна частина складається з 10 плакатів із результатами роботи.

Ключові слова: нейронна мережа, розпізнавання жестів, алгоритм, додаток.

ABSTRACT

UDC 004.514

Koliadych M.V. Intelligent human-machine application for converting “sign language to text”. Master's thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2022. – 123 p.

In Ukrainian language. Bibliographer: 21 titles; fig.: 39; tabl. 5.

In the master's qualification work, a sign language translation system from streaming video to English was developed. In the review and analytical part of the work, the existing methods and approaches to object recognition were investigated, and the existing architectures of convolutional neural networks were analyzed. In the theoretical and methodological part, a solution in the form of a neural network for recognizing gestures from streaming video is developed. The developed product is analyzed and tested. In the economic part, the technical level is analyzed and the cost of development implementation is calculated. The illustrative part consists of 10 posters with the results of the work.

Keywords: neural network, gesture recognition, algorithm, application.

Відгук
керівника магістерської кваліфікаційної роботи

студентки Колядич Марія Вікторівна група 2АКІТ-21м
на тему: Інтелектуальний людино-машинний додаток для перетворення "мова жестів - текст".

Актуальність роботи в контексті спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» доведена результатами інформаційного пошуку та аналізу літературних джерел. Додатковим підтвердженням актуальності роботи слугують опубліковані тези на Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації.

Представлені в магістерській кваліфікаційній роботі рішення обґрунтовані послідовним ланцюгом дій, які включають пошук інформації про проблему, її узагальнення, формулювання прикладних задач для її вирішення, проектування відповідного програмного засобу для вирішення поставлених задач, його тестування і формулювання висновків. Раціональність та ефективність прийнятих рішень доведені результатами тестування.

Дипломниця показала хороший рівень спеціальних знань і «м'яких» навичок. Дипломниця продемонструвала вміння: - вирішувати поставлені керівником завдання самостійно, згідно власноруч розробленої схеми заходів; - здійснювати пошук і узагальнення інформації; - комунікативні навички. Доведенням ерудиції та креативності дипломниці є вчасно представлена магістерська кваліфікаційна робота.

Основні результати, представлені в роботі отримані дипломницею самостійно. Матеріалу роботи властивий високий ступінь оригінальності, що доведено результатами перевірки на наявність запозичень.

Дипломниця працював ритмічно, без суттєвих відхилень від затвердженого графіку. Втрати зв'язку з керівником не було.

Недоліки: Автор навів лише необхідні і достатні діаграми, що описують процес проектування створеної системи. Результати багатоваріантного аналізу бажано було звести в таблицю. Автор не акцентує увагу на тім, які саме з отриманих результатів опубліковані у вигляді тез..

Загалом магістерська кваліфікаційна робота ***відповідає*** спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», заслуговує на оцінку **"А"**, а її автор ***заслуговує*** присудження кваліфікації: ступінь вищої освіти магістр, спеціальність «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Керівник магістерської кваліфікаційної роботи

Д.т.н., доцент, зав. кафедри КСУ		В'ячеслав КОВТУН
(посада, науковий ступінь, вчене звання)	(підпис)	(*Ім'я ПРІЗВИЩЕ*)

Відгук
опонента на магістерську кваліфікаційну роботу

студентки Колядич Марія Вікторівна група 2АКІТ-21м
на тему: Інтелектуальний людино-машинний додаток для перетворення "мова жестів - текст".

Актуальність роботи в контексті спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» доведена результатами інформаційного пошуку та аналізу літературних джерел. Додатковим підтвердженням актуальності роботи слугують опубліковані тези на Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації.

Перший розділ магістерської кваліфікаційної роботи повністю присвячений огляду літературних та інформаційних джерел за обраною темою. Проаналізовано не менш ніж три аналоги створюваної системи. Функціональність та форм-фактор створеної системи цілком визначені на основі результатів критичного огляду літератури.

Прийняті рішення обґрунтовані результатами огляду літератури, результатами проектування та втілені в функціонуючу програмну систему. Результати її тестування доводять правильність прийнятих рішень.

Експериментальні дослідження продумані і повні. Тести охоплюють як функції інтерфейсу створеної системи, так і доводять якість та повноту виконання нею функціонального призначення, обґрунтованого на етапі проектування.

Вміст графічної частина повною мірою репрезентує всі отримані в магістерській кваліфікаційній роботі результати. Якість рисунків в графічній частині прийнятна.

Недоліки: В роботі недостатньо обґрунтовано вихідний перелік реалізованих класифікаторів. Інтерфейс створеної системи, так само, як і метрика оцінювання якості класифікації, нестандартні.

Загалом магістерська кваліфікаційна робота ***відповідає*** спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», заслуговує на оцінку **"А"**, а її автор ***заслуговує*** присудження кваліфікації: ступінь вищої освіти магістр, спеціальність «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Опонент на магістерську кваліфікаційну роботу

Доцент кафедри АІІТ — Юрій ІВАНОВ
(посада, науковий ступінь, вчене звання) (підпис) (*Ім'я ПРІЗВИЩЕ*)

ЗМІСТ

ВСТУП

Дана магістерська робота присвячена вивченню спілкування глухих у суспільстві. Коли більшість людей говорять про мову жестів, вони мають на увазі широке поняття людей, які використовують жести, щоб показати слова чи літери. Однак єдиної жестової мови насправді не існує. Натомість існує багато мов жестів, які відрізняються самими жестами, правилами складання речень, мімікою, формою й рухами рота і губ. Існує також думка, що мова жестів якось пов'язана з розмовною мовою. Однак це не так.

Мова жестів створювалася і розвивалася окремо від будь−якої розмовної мови. Перше систематичне вивчення мови жестів почалося у Франції наприкінці XVIII століття. Після успіху у Франції метод навчання жестової мови поширився по всьому світу. Мова жестів певної країни (або навіть регіону країни) залежить від країни, з якої походить метод навчання. Таким чином, американська мова жестів має більше спільного з французькою мовою жестів, ніж з англійською, частково російською також з французькою, а частково (починаючи з Москви) з німецькою, яка сама була однією з перших двох шкіл жестової мови.

Зі стрімким розвитком сучасного суспільства та технологій з’являється все більше специфічних термінів, які також повинні бути відображені в мові жестів. Відбувається це так: спочатку термін тлумачиться буквально, потім для слова створюється символ.

Зі сказаного вище видно, що жестова мова є складною системою, яка потребує глибокого вивчення для розуміння її основи та принципів. У роботі цього майстра розкривається порівняно невелика частина масштабного розуміння всіх сурдоперекладів – процедурний переклад американської жестової мови.

Мета дослідження. Метою цього дослідження є покращення умов спілкування людей з вадами слуху шляхом удосконалення архітектури жестового перекладу.

Для досягнення мети необхідно виконати наступні завдання:

− Дослідження існуючих алгоритмів, архітектур і методів виявлення об'єктів;

− Розробка системи жестового перекладу;

− При необхідності скорегувати гіпер параметри моделі;

− аналізувати результати роботи системи;

− Робота над підвищенням точності розпізнавання та швидкості дій.

об'єкт дослідження. Процес перекладу жестової мови з потокового відео на англійську.

Тема дослідження – методи та алгоритми виявлення жестів із потокового відео та їх перекладу на англійську мову.

Практичне значення отриманих результатів. Розроблену систему можна використовувати як плагін у веб−додатку, який допоможе людям з проблемами слуху спілкуватися.

Наукова новизна отриманих результатів. Створення архітектури для системи перекладу мовою жестів із потокового відео на англійську.

Переклад жестовою мовою використовує глибоке навчання у сфері розпізнавання об’єктів у режимі потокового відео.

В рамках даної роботи досліджено існуючі методи та підходи до розпізнавання об’єктів, Проаналізовано існуючі архітектури згорткових нейронних мереж. На основі цього дослідження було розроблено рішення у вигляді нейронної мережі для розпізнавання жестів із потокового відео. Розроблений продукт аналізується та тестується. Результатом є система перекладу жестової мови з потокового відео на англійську.

Апробація і публікація результатів роботи. Результати роботи були представлені на всеукраїнській науково-практичній інтернет-конференції студентів аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи» [21].

- Колядич М.В. Класифікатор для автоматизованої автентифікації мовця за результатами візуалізації мовленнєвих сигналів [Електронний ресурс] / М. В.

Колядич // ВНТУ. – 2020. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2020/paper/view/9538.

1 АНАЛІЗ СИСТЕМ ПЕРЕКЛАДУ МОВИ ЖЕСТІВ У ПОТОКОВОМУ ВІДЕО НА АНГЛІЙСЬКУ

1.1 Опис функціональних процесів цільової системи

1.1.1 Опис процесу діяльності

Об'єктом автоматизації є процес усного перекладу з жестової мови. Вхід – відеопотік від користувача, де користувач копіює слова з мови жестів. Запис відео за допомогою веб−камери.

Ми ідентифікуємо руки (жести) із вхідного відео та використовуємо згортову нейронну мережу, щоб отримати набір атрибутів, які потім декодуються у відповідні жести. Кожен жест включається в базу даних жестів, яку потім можна змінювати, редагувати або доповнювати за потреби.

Результатом цієї процедури є відображення відповідних жестів на екрані.

Оскільки система розробляється на замовлення підприємства, існують обмеження, які встановлює замовник на розробку. Ці обмеження включають вимоги до обчислювальної потужності: система повинна мати можливість працювати в режимі реального часу на комп’ютері з вбудованою графікою та до 8 ГБ оперативної пам’яті.

1.1.2 Актори і функції

Під час використання системи користувачі можуть створювати власні жести, ініціювати розпізнавання відео та отримувати переклади. Взаємодія між користувачем і системою показано на рисунку 1.1.

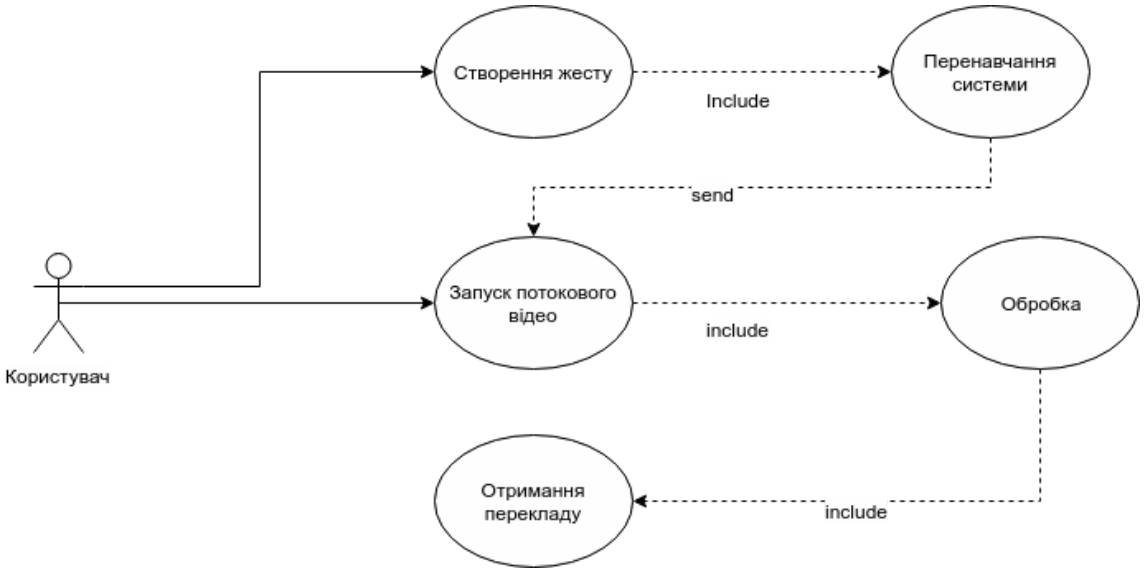


Рисунок 1.1 – Діаграма варіантів використання

1.2 Постановка задачі

Метою цього дослідження є покращення умов спілкування людей з вадами слуху шляхом удосконалення архітектури жестового перекладу.

Для досягнення цієї мети необхідно вивчити існуючі алгоритми, архітектури та методи виявлення об’єктів. Розробіть систему перекладу мовою жестів на основі вивченого. Аналізуйте результати роботи системи та працюйте над удосконаленням системи, тобто точністю розпізнавання, точністю жестового перекладу, швидкістю дій.

Для підтримки поточного перекладу система повинна мати можливість обробляти вхідне відео тривалістю не більше 5 секунд. Крім того, оброблене відео, яке продовжуватиме відображатися, не повинно містити зупинок або зависань. На відміну від більшості існуючих подібних систем, важливо, щоб система могла точно перекладати жести, а не букви чи цифри.

Для досягнення цієї мети необхідно виконати наступні завдання:

− Дослідження існуючих алгоритмів, архітектур та методів виявлення об’єктів;

− Розробка системи жестового перекладу;

− При необхідності скорегувати гіпер параметри моделі;

− аналізувати результати роботи системи;

− Робота над підвищенням точності розпізнавання та швидкості дій.

1.3 Предметний огляд аналогів

Розпізнавання мови жестів на основі відео без часової сегментації, Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, Weiping Li [1] - набір даних у сучасних моделях китайської мови жестів. Точність моделі становить 82,7%. Архітектура - ієрархічна мережа уваги (LS-HAN).

До недоліків моделі можна віднести посередню точність розпізнавання, китайську мову жестів, відсутність відкритого коду та недоступність для комерційного використання.

Використання глибоких згорткових мереж для розпізнавання жестів американською мовою жестів, Вівек Бхеда та Н. Діанна Радпур [2] - Набір даних у моделях американської мови жестів. Точність - ~90%. Архітектура - deepCNN.

Недоліки цієї моделі включають нижчу точність розпізнавання, розпізнає лише літери та цифри та відсутність відкритого вихідного коду, що робить MS-ASL: великомасштабний набір даних і орієнтир для розуміння американської мови жестів, комерційно використаний Хамідом Реза Ваезі Джозе, Оскар Коллер [3] - Набір даних у моделі RWTH− BOSTON. Точність - ~86%. Архітектура - ASL100.

До недоліків моделі можна віднести середню точність розпізнавання, відсутність відкритого коду та недоступність для комерційного використання.

1.4 Нейромережі як технологія рішення поставленої задачі

Як працюють нейронні мережі? Нейронні мережі були першими, кого прославив Google. Google першим впровадив систему, яка імітує роботу людського мозку.

Як випливає з назви, нейронна мережа – це набір взаємопов’язаних штучних нейронів, які імітують мозок. Тобто вони мають певний елемент (схожий на сома

- тіло клітини біологічних нейронів) і має сполучні лінії та точки з іншими нейронами (подібно до дендритів і синапсів відповідно).

У програмуванні нейрони – це завершені елементи програмного коду, які утворюють нейронну мережу. Кожен нейрон отримує вхідні дані, обробляє їх і передає через синапси. Якщо говорити трохи зрозуміліше, нейрон є основною одиницею штучного інтелекту. Нейронна мережа – це комп’ютерна реалізація людського мозку [4].

Розвиток Інтернету та процес глобалізації призвели до появи великого обсягу інформації, яку не може опрацювати одна людина [4]. Основні завдання нейронної мережі:

− аналіз та класифікація за визначеними параметрами;

− прогнози на основі вхідної інформації;

- Виявлення тих самих даних.

Наприклад, останній пункт використовується для систем безпеки аеропорту. Це робиться шляхом захоплення людських облич і порівняння їх із базою даних злочинців. Іншим прикладом є функція пошуку схожих зображень Google. Досить завантажити одну фотографію, і система знайде всі схожі зображення [4].

Існує кілька типів нейронних зв'язків. Найпоширенішими з них є сигма і ReLU. Sigmoid дає відповіді в діапазоні від -1 до 1 (або від -100% до 100%). Вони найчастіше використовуються для класифікації тих завдань, які вимагають однозначної відповіді «так» або «ні». У випадку ReLU відповідь коливається від 0 до нескінченності.

Щоб пояснити, як працює системний аналіз, сигмоїдна функція більш підходить, оскільки обмежений діапазон вхідної інформації є більш зрозумілим. Алгоритм розрахунку ReLU [4]:

Крок 1. Мережа введення даних Крок 2. Розрахунок ваги;

Крок 3. Передайте інформацію на наступний шар;

Крок 4. Повторюйте дію до кінця мережі;

Крок 5 виконати обробку функції активації отриманих результатів Крок 6 Опублікувати результати.

Змінюючи кількість шарів у мережі, можна змінити обчислювальну складність і, таким чином, точність прогнозів. На сьогоднішній день існують нейронні мережі з близько 1 мільйона нейронів, які можуть повністю імітувати мозок бджоли (в мозку людини близько 86 мільйонів нейронів). В даний час нейронні мережі існують навіть для легких мобільних додатків, що є свідченням еволюції технології (оскільки попередні нейронні мережі були складними і займали багато часу для обробки даних).

Важливим етапом навчання є підготовка даних. Тобто дані повинні бути приведені в певну форму. Що саме це означає?

Розглянемо наступний приклад: Аналізуємо динаміку фондового ринку. У цьому випадку ціна буде набагато більше одиниці. Тому дані можна звести до різниці в ціні, вираженої у відсотках. На виході ми отримаємо значення в діапазоні від -1 до 1[4].

Описана послідовність дій називається нормалізацією вхідних даних. Це перший і найважливіший крок перед початком машинного навчання. Система повинна отримувати інформацію у формі, яку вона може обробити [4].

Коли ви підготуєте свої дані, ви можете почати власне навчання. Перший результат, швидше за все, буде неправильним і має велику похибку. Це можна пояснити тим, що мережа ще не «навчилася». Тобто ваги мережі ближчі до випадкових ваг, ніж ваги прогнозованих результатів. Адже головне завдання нейронної мережі – підлаштувати вагові коефіцієнти під ті значення, які дають точні результати. Спочатку вагові коефіцієнти встановлюються на випадкові значення, потім, у кожну епоху навчання, вони коригуються відповідно до змін у точності відповіді.

Для того, щоб нейронна мережа правильно регулювала ваги, необхідно вказати набір операндів, щоб керувати обробкою даних і допомагати нейронній мережі правильно змінювати ваги. Кількість таких операцій може коливатися від 4 до майже 1000 в залежності від рівня мережі.

Ітерація по всіх операндах одночасно називається епохою. Кількість епох залежить від потреб і складності нейронної мережі та самих даних. З кожною епохою точність розрахунку зростає. Коли похибка не перевищує кількох відсотків, навчання вважається успішним. Однак це також залежить від самого завдання. Наприклад, такі тривіальні завдання, як визначення рукописних цифр, повинні мати похибку менше 1%, але більш складні завдання можуть мати більші похибки (бажано не більше 8-10%).

1.5 Штучний нейрон як базовий елемент нейромережі

Історично робота Уоррена С. Маккалоха та Волтера Піттса [5] вважається першою публікацією, яка заклала основу для створення штучних нейронів і нейронних мереж. У цій роботі пропонується теорія, заснована на тому, що всі аспекти нейронної активності можна моделювати за допомогою мережі елементів з двома стабільними станами [6]. Модель цього елемента показана на рисунку 1.2 і називається формальним нейроном [6].

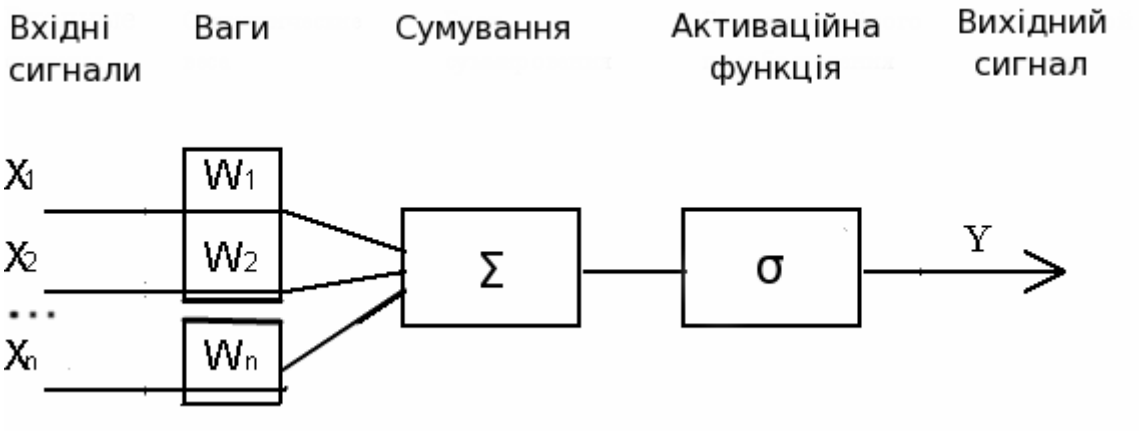


Рисунок 1.2 – Типовий штучний нейрон

Синаптична функція моделюється шляхом масштабування вхідного сигналу (x0, x1, ..., xn) з ваговими коефіцієнтами (w0, w1, ..., wn) [6]. Отриманий сигнал надходить на вхід суматора, який обробляє його за рівнянням [6]:

$$v = \sum_{i=0}^{n} w_i i x_i \quad (1.1)$$

де v − аргумент функції активації:

$$y = f(v). \quad (1.2)$$

Функція активації, запропонована в [5], є звичайною функцією з умовою більшості або меншості від нуля:

$$y = \begin{cases} 1, & v \leq 0, \\ 0, & v \geq 0 \end{cases} \quad (1.3)$$

В даний час використовується велика кількість різноманітних функцій активації. Найпоширенішим є сигмовид, який може бути дискретним і аналоговим.

Прикладом найпростішої функції активації є дискретна сигма, параметри якої називають функцією Хевісайда або тета-функцією [6]:

$$y = \begin{cases} 1, & v \leq a, \\ 0, & v \geq a \end{cases} \quad (1.4)$$

Якщо значення активації v нейрона з функцією активації Хевісайда не є

Понад значенням параметра a нейрон залишається пасивним, а при перевищенні порогу видає фіксоване значення функції як логічну одиницю [6].

Поширеним варіантом нелінійної АФ є параметрична сигмоїдна функція форми:

$$y = \frac{b}{c + e^{dv}} \tag{1.5}$$

Якщо параметри дорівнюють b = 1, c = 1, d = -1, то отримуємо:

$$y = \frac{1}{1 + e^{-v}} \tag{1.6}$$

Амплітуда вихідного сигналу нейрона із заданою функцією активації залежить від амплітуди вхідного сигналу [6]. На рисунку 1.3 видно, що нахил сигмоїдної функції залишається високим лише в певному діапазоні [6].

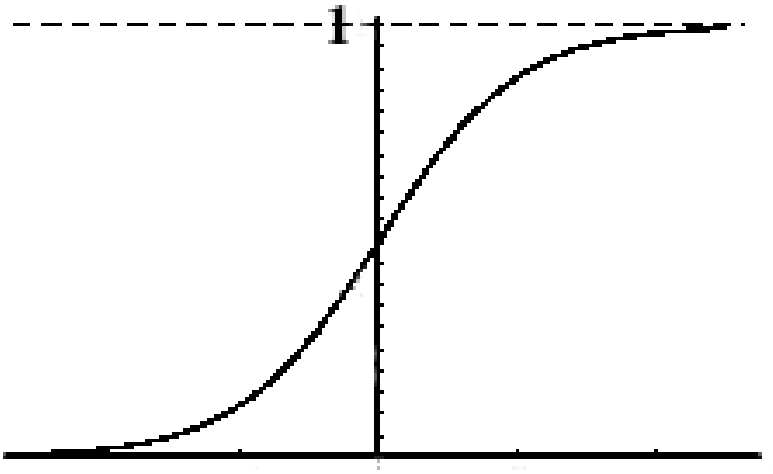


Рисунок 1.3 – Сигмоїдальна функція активації

Цю властивість вперше використав Гроссберг у [7]. Завдяки нелінійному коефіцієнту посилення нейрони можуть надійно працювати в широкому діапазоні рівнів вхідного сигналу [6]. Іншою корисною властивістю для моделювання сигмоїдних функцій активації є просте вираження диференційованості та похідних уздовж всієї осі абсцис [6]:

$$y' = y(1 - y),$$

Функція гіперболічного тангенса схожа, як показано на рисунку 1.4.

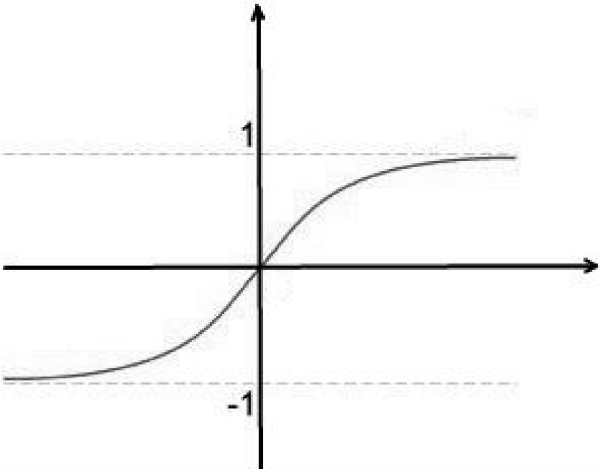


Рисунок 1.4 – Функція активації гіперболічний тангенс

Однак, на відміну від сигмоїдальної функції, гіперболічна тангенсова функція симетрична відносно початку координат. Це також дає можливість використовувати від'ємні значення.

Гіперболічна функція визначається коефіцієнтами k, m, які регулюють параметри вихідного сигналу

$$y = k * th(m * v) \qquad (1.7)$$

При m = 1, k = 1 отримаємо:

$$y = th\ (v),$$

що забезпечує поширення сигналу мережею.

Дзвонові функції також доступні в дискретному та аналоговому форматах. Дискретний формат із параметрами k, m задається рівнянням (1.8). На рисунку 1.5 графічно показано дискретний формат дзвоноподібної функції.

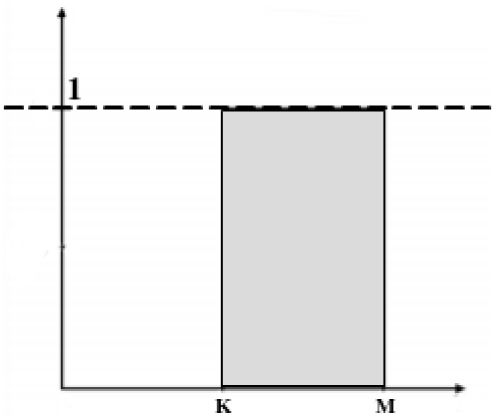


Рисунок 1.5 – Дискретна дзвонова функція активації

$$y = \begin{cases} 1, & v > k, v \leq m, \\ 0, & v \leq 0, v > m. \end{cases} \quad (1.8)$$

Функції аналогового дзвоника використовуються для аналогового моделювання нейронів із властивостями смугового фільтра [6]. Існують дзвоноподібні показникові функції першого і другого порядку.

Аналогова дзвонова функція першого порядку представлена формулою k, m, n, p, r:

$$y = \frac{1}{ke^{mv} + ne^{pv} + r}. \quad (1.9)$$

Коли k = 1, m = 1, n = 1, p = -1, r = -1, стандартна інваріантна форма функції Белла першого порядку показана на рисунку 1.6.:

$$y = \frac{1}{e^{v} + e^{-v} - 1}. \qquad (1.10)$$

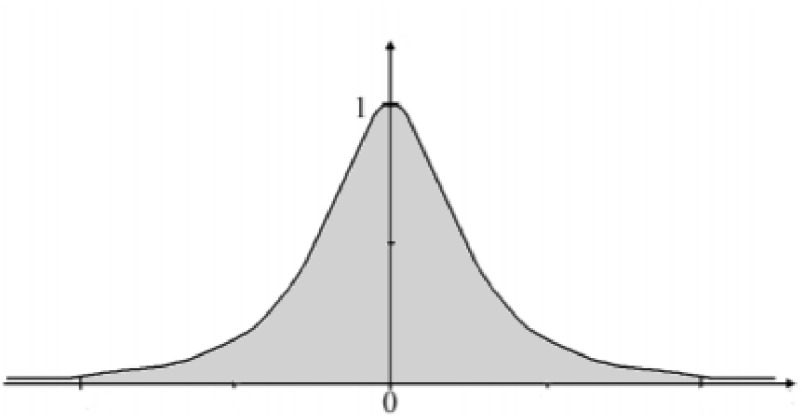


Рисунок 1.6 – Аналогова функція активації 1-го порядку

Дзвонова функція 2-го порядку має вид

$$Y = e^{-v^2}$$

Різниця між дзвоноподібною функцією першого та другого порядку полягає в швидкості наростання та спаду. Аналогова функція другого порядку показана на рисунку 1.7. Більш висока швидкість функції другого порядку дозволяє використовувати її у фільтрах з більш жорсткими параметрами.

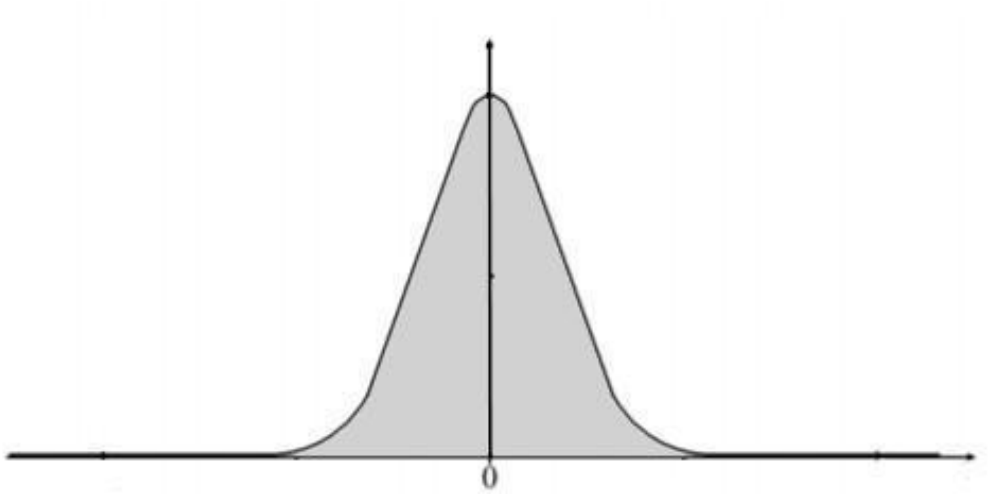


Рисунок 1.7 – Аналогова функція активації 2-го порядку

Нейронні мережі – це спосіб створення програмного забезпечення, яке вирішує проблеми без прямого опису рішення. Таким чином можна отримати рішення з більш детальним описом коефіцієнтів, ніж це можуть зробити люди.

Сьогодні кількість інформації у світі щомиті зростає в геометричній прогресії. Це зростання спонукало до створення нейронних мереж.

Коли обсяг даних, які потрібно обробити, досягає мільйонів або навіть мільярдів, обробка людьми стає неможливою, і навіть комп’ютерна обробка може вимагати значної обчислювальної потужності та часу. У результаті класичні методи та алгоритми програмування вже не здатні вирішити проблему.

Саме в цей момент виникла потреба в нейронних мережах. Зрештою, вони не просто переглядають кожне зображення піксель за пікселем, а й знаходять особливі характеристики, спільні для набору даних.

Нейронні мережі можуть працювати в будь-якій сфері і можуть допомогти вирішити багато проблем, які не під силу класичним методам програмування. Наприклад, розпізнавання людей, предметів, облич, аналіз дорожньої розмітки, обробка звуку, реконструкція зображення, аналіз нетипової поведінки тощо.

Як видно з вищесказаного, завдання розпізнавання жестової мови та її перекладу не реалізується на комерційному рівні. Є дослідження з цього питання, але всі вони закриті, і є такі недоліки, як низька швидкість обробки, низька точність і потреба у великій обчислювальній потужності. Тому створення систем

мовного перекладу є актуальною проблемою з високим потенціалом для створення проектів.

2 ІСНУЮЧІ НЕЙРОМЕРЕЖІ І МЕТОДИ ОБРОБЛЕННЯ ДАНИХ

Машинне навчання (ML) використовується для складних завдань, які люди не можуть безпосередньо кодувати. Деякі завдання настільки складні, що важко, а то й неможливо чітко описати всі нюанси та запрограмувати їх. Тому ми передаємо багато даних в алгоритм MN і дозволяємо програмі вирішити це, вивчаючи дані та знаходячи параметри моделі, які досягають того, чого хоче розробник.

Розглянемо ці два приклади:

Приклад 1. Написати програму для розпізнавання 3D-об’єктів за нових зовнішніх умов, таких як освітлення та фоновий шум, є складним завданням. Ми не знаємо, яку програму написати, тому що ми не знаємо, як це завдання реалізується в людському мозку. І навіть якщо ми розуміємо, як людський мозок обробляє це завдання, програмна реалізація цього процесу може бути надзвичайно складною.

Приклад 2: Створити програму для визначення ймовірності шахрайства з банківським рахунком дуже складно. Не існує стандартизованого способу обійти захист банків. Натомість шахраям доводиться щоразу знаходити нові способи, тому програми мають постійно змінюватися та виявляти слабкі місця захисту, навіть раніше, ніж хтось інший.

Використання MN у подібних завданнях: програма більше не повинна бути написана самим розробником у конкретній задачі. Вона повинна використовувати набір даних багатьох подібних випадків і надавати їх алгоритму MN, а алгоритм MN визначає правильне рішення для загального випадку. Програми, створені за допомогою алгоритмів навчання, можуть сильно відрізнятися від типових рукописних програм. Він може містити сотні чисел і багато параметрів, і зміни цих параметрів призводять до змін результатів. Якщо алгоритм належним чином навчений, він зможе точно передбачати випадки, яких немає в навчальній вибірці. Якщо дані змінюються, програма також повинна змінюватися, навчаючись на

нових прикладах. Крім того, розробка MN-алгоритмів менш дорога, ніж ручна обробка даних.

Основні типи задач, які найкраще вирішуються за допомогою машинного навчання:

− розпізнавання предметів, розпізнавання обличчя чи виразу обличчя, читання по губах;

− виявлення аномалій (наприклад, аномалії в метеорологічних спостереженнях, діагностика захворювань);

- Прогнозування (наприклад: прогнози цін на нерухомість, системи рекомендацій, економічні показники).

Нейронні мережі є підкласом машинного навчання. Вони являють собою набір методів і алгоритмів, які зробили революцію в області машинного навчання. Вони розроблені так, щоб нагадувати біологічні нейронні мережі, а сучасні глибокі нейронні мережі вже дуже добре відтворюють природні нейронні мережі глибиною до 107 нейронів. Нейронні мережі самі по собі є загальними наближеннями функцій, тому їх можна застосовувати майже до будь-якої задачі MN, де основною метою є знаходження характерних коефіцієнтів усіх шарів мережі.

2.1 Найпоширеніші типи нейромереж

2.1.1 Персептрон

Перше покоління нейронних мереж, перцептрон, було лише обчислювальною моделлю одного нейрона (рис. 2.1). Персептрони вперше були описані Френком Розенблатом у його статті «Персептрони: імовірнісна модель для зберігання та організації інформації в мозку» (1956) [7]. Також відомий як нейронна мережа прямого зв’язку, перцептрон отримує інформацію спереду назад. Навчання персептрона зазвичай вимагає зворотного поширення, надаючи мережі парні вхідні та вихідні набори даних. Вхідні дані надсилаються до нейронів, які

обробляються та призводять до виходів. Помилка зворотного поширення зазвичай є деякою зміною різниці між входом і виходом. Враховуючи, що мережа має достатньо прихованих нейронів, вона теоретично може завжди моделювати залежності між входами та виходами. На практиці їх використання дуже обмежене, але вони зазвичай поєднуються з іншими мережами для створення нових мереж.

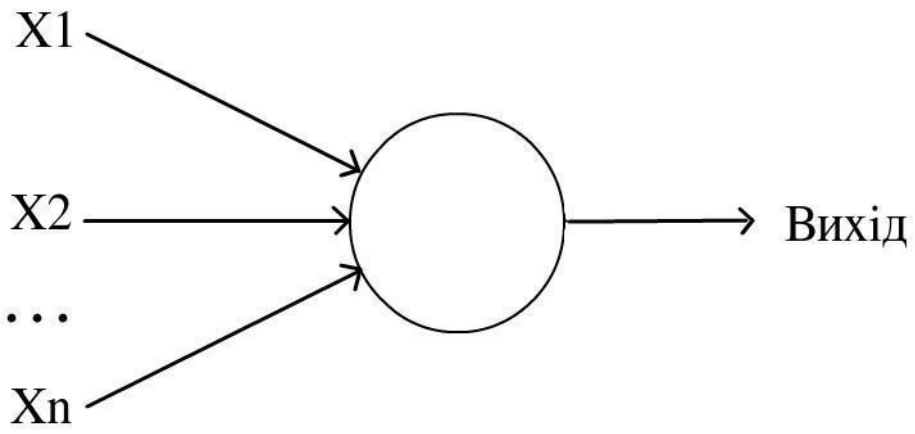


Рисунок 2.1 – Персептрон

2.1.2 Конволюційна (згорткова) нейромережа

У 1998 році Ян Лекун [9] і його колеги розробили нейронну мережу під назвою LeNet для розпізнавання рукописних цифр. Він використовує зворотне поширення з багатьма прихованими шарами, карти з багатьма повторюваними одиницями на кожному шарі, широкі мережі, які можуть обробляти кілька символів одночасно, навіть якщо вони накладаються, і розумні способи навчання всієї системи, а не лише розпізнавача. Пізніше цей метод буде названо згортковою нейронною мережею (CNN). Приклад архітектури згорткової нейронної мережі показано на рисунку 2.2.

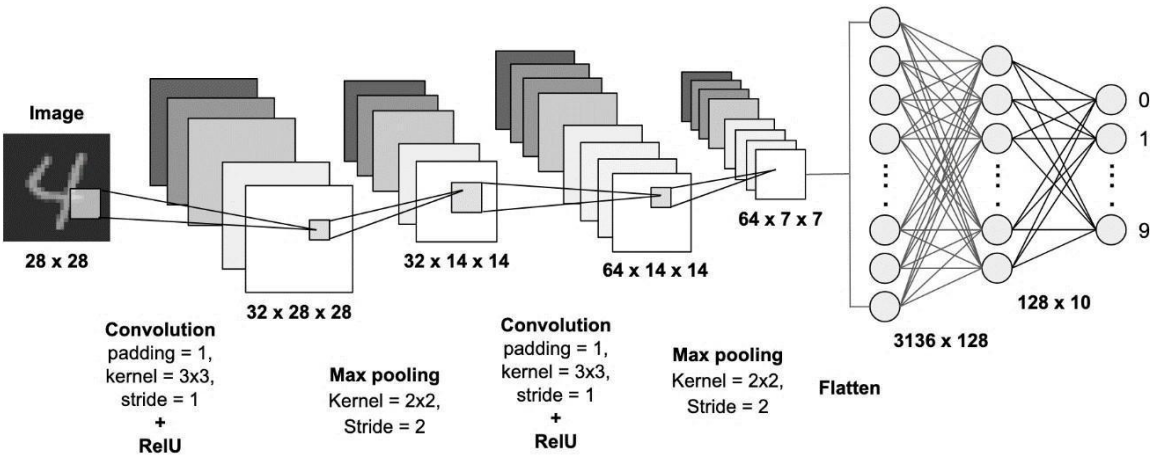


Рисунок 2.2 − Конволюційна нейромережа (CNN) [10]

Згорткові нейронні мережі найчастіше використовуються в обробці зображень, але також можуть бути застосовані до інших типів даних, таких як звукові послідовності. Саме цей тип нейронної мережі використовувався в цьому дослідженні та буде описаний більш детально в наступних параграфах. Стандартною операційною моделлю для CNN є класифікація вхідних даних. У типовій архітектурі CNN перший рівень відповідає не за аналіз даних, а за сканування даних. Для обробки зображення розміром 200 x 200 пікселів створення шару з 40 000 нейронів є надзвичайно неоптимальним. Найкращим підходом є створення вхідного сканованого шару, скажімо, 20 x 20, який забезпечує частку пікселів зображення. Після цього обробіть наступні 20 x 20 пікселів, пересуваючи вікно обробки на один піксель праворуч (або на вказану кількість пікселів, див. розділ «Крок»).

Після цього замість початкових 40 000 пікселів відфільтровані дані надсилаються на наступні шари для обробки. Шари в згорткових нейронних мережах, як правило, зменшують розмірність, але поглиблюють. Окрім згорткових шарів, CNN також мають рівні об’єднання. Приєднання – це спосіб фільтрувати деталі. Найпоширенішим варіантом є пошук максимального значення у вікні об'єднання.

2.1.3 Рекурентні нейромережі

Щоб зрозуміти RNN, потрібен короткий огляд обробки послідовностей. У більшості випадків під час використання MN для обробки послідовностей даних вам потрібно перетворити вхідну послідовність на іншу вихідну послідовність (наприклад, перетворити послідовність англійських слів на послідовність французьких слів). Наступний елемент у вхідній послідовності можна передбачити без необхідності перетворювати послідовність в іншу послідовність. Цей підхід є більш логічним, ніж спроба передбачити окремі пікселі чи окремі частинки зображення на основі решти зображення. Передбачення наступного елемента в послідовності стирає різницю між контрольованим і неконтрольованим навчанням. Він використовує методи, розроблені для навчання під наглядом, але не потребує окремого тренувального сигналу. На рисунку 2.3 показаний приклад архітектури рекурентної нейронної мережі.

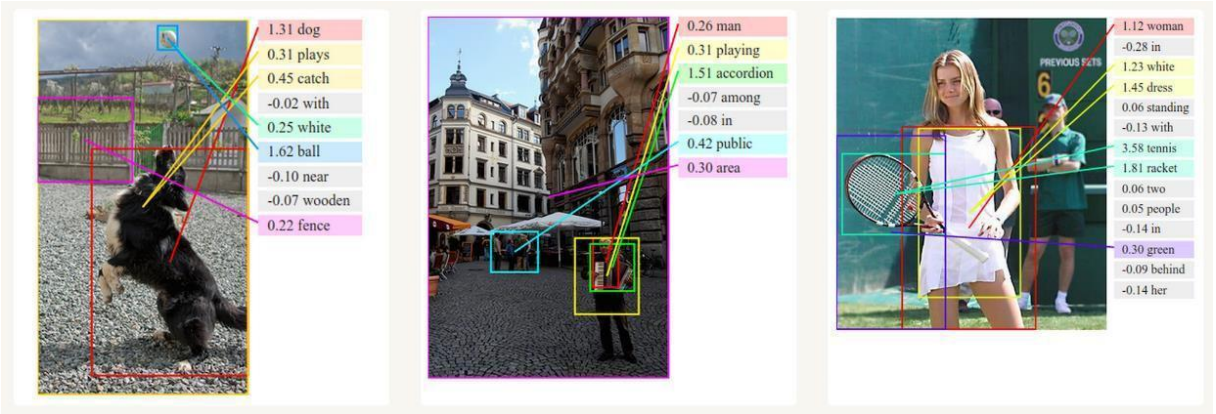


Рисунок 2.3 – Приклад роботи рекурентної нейромережі [11]

Моделі без пам’яті є стандартним підходом до MN. Нейронні мережі прямого зв’язку – це узагальнені моделі, які використовують один або кілька прихованих рівнів із нелінійними функціями активації. Однак, якщо ми надамо нашій генеративній моделі деякий прихований стан і якщо ми надамо цьому стану деяку внутрішню динаміку, ми можемо отримати більш цікаву модель: вона може зберігати інформацію тривалий час у прихованому стані. Якщо динамічні та

згенеровані приховані стани зашумлені, ймовірність може бути розподілена по простору прихованих векторів стану.

Первісно представлені Джеффрі Елманом (1990) «Пошук часу в структурі» [12], рекурентні нейронні мережі (RNN) в основному є перцептронами; однак, на відміну від перцептронів без стану, вони мають зв’язки між стрибками, Connect через час. RNN є дуже потужними, оскільки вони поєднують у собі 2 властивості:

− Розподілений прихований стан, який може зберігати великий обсяг історичної інформації;

– Нелінійна динаміка, тому оновлення прихованих шарів є більш ефективними.

За достатньої кількості шарів і часу RNN може обчислити будь-що. Вони можуть коливатися або поводитися нестабільно. Вони мають потенціал для впровадження простих паралельних програм, у той час як ці програми взаємодіють, створюючи дуже складні ефекти.

Як і бази даних, мережі RNN мають кілька реляційних варіантів. Це такі зв’язки, як «один до одного», «один до багатьох», «багато до одного» та «багато до багатьох» (рис. 2.4).

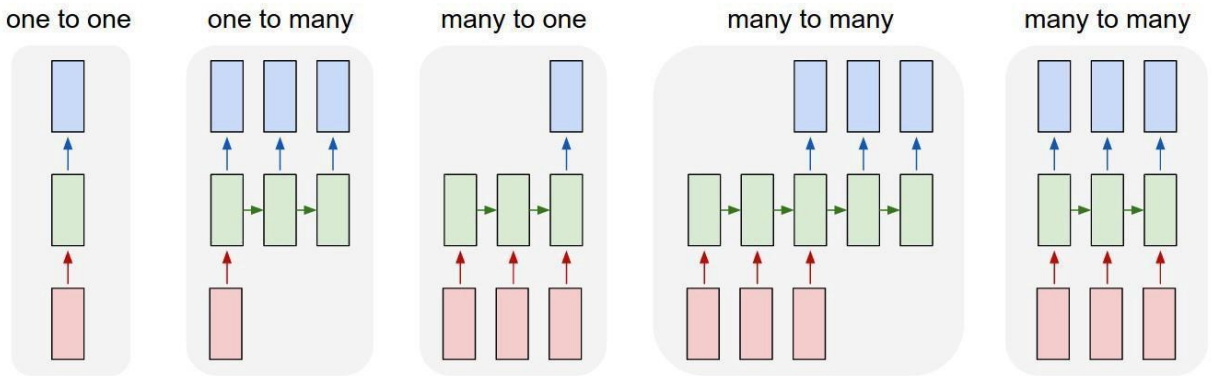


Рисунок 2.4 – Види зв’язків в рекурентних нейромережах [13]

Однією з головних проблем із RNN є проблема зникнення (або розриву) градієнтів, де значення градієнта можуть швидко наближатися до нуля з часом або дані занадто великі, залежно від використовуваного автофокусування. Інтуїтивно зрозуміло, що це невелика проблема, оскільки це лише ваги, а не стани нейронів,

але ваги зберігають історичну інформацію про стан; якщо ваги досягають значення 0 або 1 000 000, попередній стан не надає багато інформації. RNN в принципі можна використовувати багатьма способами, оскільки типи даних, які не прив’язані до часу (тобто, на відміну від відеопотоків), у більшості випадків можуть бути визначені як послідовності. Зображення або рядок тексту можна фіксувати по одному пікселю або одному символу за раз, тому, якщо ваги залежать від часу, вони використовуються для визначення елементів, розташованих раніше в послідовності, а не елементів х секунд тому. Як правило, рекурентні мережі використовуються для пошуку продовжень або додаткової інформації послідовності, такої як автозавершення.

2.1.4 Довго- / короткотривала пам’ять

Хохрайтер і Шмідхуберт у 1997 році вирішили проблему змусити RNN запам’ятовувати речі протягом тривалого часу, побудувавши так звану довгу короткочасну пам’ять (LSTM) [14] (рис. 2.5). Мережі LSTM намагаються вирішити проблему градієнтного затухання або сплесків, вводячи вентилі та чітко визначені комірки пам’яті. Комірка пам’яті зберігає попередні значення та зберігає їх, якщо «ворота забуття» не скаже комірці забути ці значення. LSTM також мають «вхідні ворота», які додають нові елементи до клітинки, і «вихідні ворота», які вирішують, коли перейти через вектор із цієї клітинки до наступного прихованого стану.

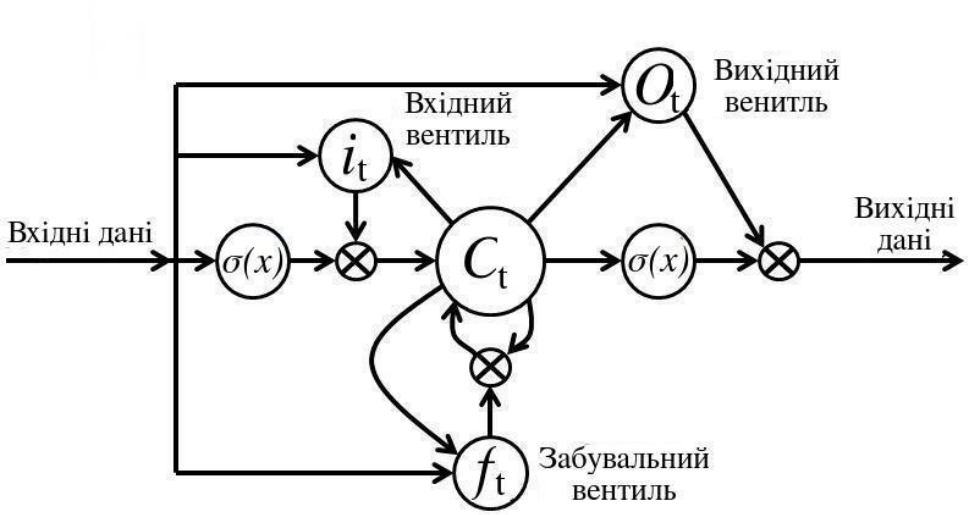


Рисунок 2.5 – Базовий вузол нейромережі LSTM

Нагадаємо, що для всіх RNN значення з початкової вибірки даних і попереднього шару використовуються для обчислення поточного прихованого шару. Тому при розрахунку коефіцієнтів наступного прихованого шару будуть використані дані з поточного прихованого шару. LSTM просто додають шар комірок, щоб забезпечити високий ступінь передачі інформації про стан між ітераціями. Іншими словами, ми хочемо пам’ятати вміст із попередніх ітерацій якомога довше, і комірки в LSTM дозволяють нам це робити. Було показано, що LSTM вивчають складні послідовності, наприклад, писати, як Мольєр, або писати оригінальну музику.

2.1.5 Керований рекурентний блок (GRU)

Gated Recurrent Units (GRU) трохи відрізняються від LSTM, описаних вище. Вони беруть як вхідні дані початкового навчання та дані з попереднього рівня. Виконайте деякі обчислення, а потім пройдіть уздовж поточного шару. У наступній ітерації використовується наступний елемент початкового набору даних, поточний рівень використовується для додаткових обчислень тощо. Вони відрізняються від LSTM тим, що GRU не потребують шару комірок для передачі значень. Обчислення в кожній ітерації гарантують, що значення поточного

прихованого шару поширюється безпосередньо, або зберігаючи дані шару в комірках пам’яті, або виконуючи стрибки, а по днях під час обробки багатьох нових даних.

Загалом GRU функціонують дуже подібно до LSTM, велика різниця полягає в тому, що GRU трохи швидші та легші для керування. У деяких випадках GRU може перевершити LSTM. Більше про GRU у статті Junyoung Chung за 2014 рік «Емпірична оцінка стробованих рекурентних нейронних мереж у моделюванні послідовностей» [15].

2.1.6 Глибока мережа довіри (Deep Belief Network)

Метод зворотного поширення помилок використовується при обчисленні функції втрат для кожного шару нейронної мережі та кожного окремого нейрона. Однак є деякі серйозні проблеми з використанням зворотного поширення. По-перше, для цього потрібні навчальні дані з мітками; в той же час майже всі дані не марковані. По-друге, час навчання погано масштабується, що означає, що він дуже повільний у мережах із кількома прихованими рівнями. По-третє, він може застрягти в малих локальних оптимумах, тому вони далекі від оптимальних для сітки глибини.

Щоб подолати обмеження зворотного розповсюдження, дослідники розглянули можливість використання неконтрольованих методів навчання. Це допомагає підтримувати ефективність і простоту використання методу градієнта для коригування ваг і використовувати його для моделювання вхідної структури. Зокрема, вони відкоригували ваги, щоб збільшити ймовірність того, що генеративна модель забезпечить сенсорний вхід. Питання в тому, яку генеративну модель ми повинні тренувати? Або поєднання двох?

Джошуа Бенгіо створив цю мережу у своїй статті «Жадібне пошарове навчання глибоких мереж» [16], яка показала ефективність відстеження. Ця техніка також відома як жадібне навчання, і жадібний означає прийняття локально оптимальних рішень для отримання хорошої, але, можливо, не оптимальної

відповіді. Мережа являє собою спрямований ациклічний граф, що складається з випадкових величин. Використовуючи мережі переконань, ви можете спостерігати деякі змінні та вирішити 2 проблеми:

− Проблеми міркування: встановлення стану немічених змінних;

− Проблема навчання: налаштуйте взаємодію між змінними, щоб мережа могла швидше генерувати дані навчання.

Мережі глибокої віри можна навчити за допомогою зворотного поширення та представити дані як імовірнісну модель.

2.1.7 Автоенкодери

Автоенкодер – це тип нейронної мережі, що спеціалізується на неконтрольованому навчанні, тобто коли навчальні дані не позначаються. Як модель стиснення даних, їх можна використовувати для кодування заданого вхідного даних для створення меншого розміру даних. Потім вхідний сигнал може бути реконструйований із закодованої версії за допомогою декодера.

Те, що вони роблять, дуже схоже на аналіз головних компонентів, який часто використовується для представлення вхідних даних з використанням меншої кількості вимірів, ніж вихідні. Так, наприклад, у НЛП, якщо ви представляєте слово як вектор із 100 чисел, ви можете використовувати PCA, щоб представити його як 10 чисел. Звичайно, це може призвести до певної втрати інформації, але якщо у вас є лише обмежена кількість вимірів для роботи, це хороший спосіб роботи з вашими даними. Крім того, це чудовий спосіб візуалізації даних, оскільки легко побудувати дані зменшеної розмірності на двовимірному графіку порівняно зі 100-вимірним вектором. Автокодери виконують подібну роботу - різниця полягає в тому, що вони можуть кодувати заданий вектор у вектор із меншими розмірами за допомогою нелінійного перетворення (порівняно з PCA, яке є лінійним перетворенням). Таким чином, він може генерувати більш складні кодування. Їх можна використовувати для зменшення розмірності, пошуку інших

нейронних мереж, генерації даних тощо. Є кілька причин для використання автоматичного шифрувальника:

- Вони забезпечують гнучке відображення двома способами;

- час навчання є лінійним (або краще);

- Остаточна модель кодування дуже компактна та швидка.

Однак оптимізація глибоких автокодерів за допомогою зворотного поширення виявилася дуже складною. Коли початкова вага мала, градієнт зворотного поширення зникає. Сьогодні вони рідко використовуються на практиці, головним чином тому, що було доведено, що навчання під керівництвом вчителів працює краще в ключових сферах, які колись вважалися проривними, наприклад, підготовка до служби.

2.1.8 Генеративні змагальні мережі

У Generative Adversarial Networks (2014) [17] Ян Гудфеллоу пропонує новий тип нейронної мережі, в якій дві мережі працюють разом. Генеративні суперницькі мережі (GAN) складаються з будь-яких двох мереж (хоча, як правило, з прямим розповсюдженням і згортковими нейронними мережами), одна з яких відповідає за генерування вмісту (генеративна), а інша відповідає за оцінку вмісту (дискримінаційна). Завдання дискримінаційної моделі полягає в тому, щоб визначити, чи виглядає дане зображення природним (зображення з набору даних) чи виглядає штучно створеним. Завдання генератора – створити природні зображення, схожі на вихідні дані. Аналогія, яка використовується в статті, полягає в тому, що генеративна модель схожа на «команду підрядників, які намагаються виготовити та використовувати фальшиві гроші», а дискримінаційна модель – «поліція намагається виявити фальшиві гроші». Генератор намагається обдурити дискримінатор, а дискримінатор намагається не бути обдуреним генератором. Оскільки моделі навчалися з перемінною оптимізацією, обидва методи вдосконалювалися до тих пір, поки «підробки не стали відрізнити від справжніх». За словами Янна Лекуна, ці мережі можуть стати наступною великою

подією. Вони є одними з небагатьох успішних методів навчання без вчителя, які швидко покращують нашу здатність виконувати генеративні завдання. Ведеться багато активних досліджень в області застосування GAN до мовних завдань, підвищення їх стабільності та зручності вивчення. Вони вже використовуються в промисловості для різних застосувань, включаючи інтерактивне редагування зображень, оцінку 3D-форми, виявлення наркотиків, напівконтрольоване навчання в робототехніці.

Нейронні мережі є однією з найкрасивіших парадигм програмування, коли-небудь створених. За допомогою звичайних методів програмування вам потрібно пояснити комп'ютеру, що робити, розбиваючи велику проблему на безліч маленьких, чітко визначених завдань, які комп'ютер може легко виконати. Навпаки, у нейронних мережах ми не говоримо комп'ютеру, як вирішити нашу проблему. Натомість на основі цих спостережень він вчиться знаходити власне вирішення проблеми.

Сьогодні глибокі нейронні мережі та глибоке навчання досягли видатних результатів у багатьох важливих проблемах, таких як комп'ютерне бачення, розпізнавання мови та обробка природної мови. Вони широко використовуються такими компаніями, як Google, Microsoft і Facebook.

2.2 Конволюційна нейромережа (CNN або ConvNet)

У загальних архітектурах нейронних мереж існує два основних типи мереж: згорткові нейронні мережі (в основному використовуються для таких завдань, як розпізнавання зображень, класифікація зображень і обробка зображень і відео) і рекурентні нейронні мережі. Виявлення об'єктів, розпізнавання облич тощо є деякими з областей, де CNN широко використовуються.

Класифікація зображень CNN беруть вхідні зображення, обробляють їх і класифікують за певними класами (наприклад, собака, кіт, тигр, лев). Комп'ютер сприймає вхідне зображення як масив пікселів, який залежить від роздільної здатності зображення. Виходячи з роздільної здатності зображення, обчисліть

параметри h x w x d (h = висота, w = ширина, d = розмір). Наприклад, зображення масиву матриці RGB 6 x 6 x 3 (3 стосується значень RGB, рис. 2.6) і зображення матриці зображення матриці градацій сірого розміром 4 x 4 x 1.

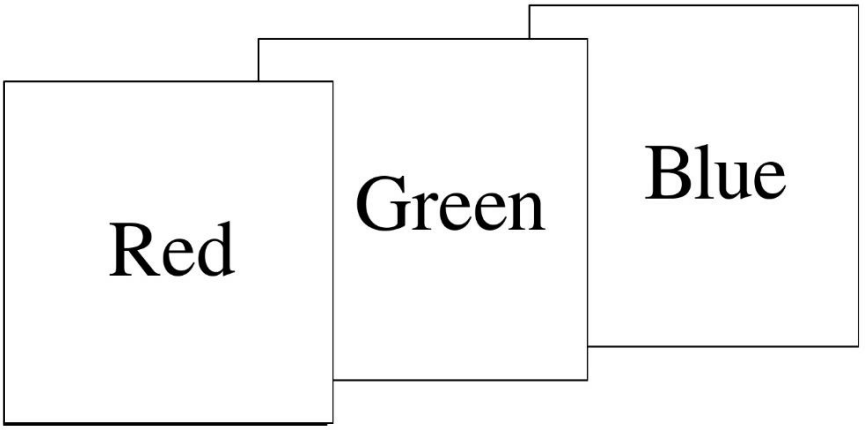


Рисунок 2.6 − Масив RGB матриць

Технічно, завдяки моделі глибокого навчання CNN для навчання та тестування, кожне вхідне зображення проходитиме серію згорткових шарів із фільтрами (ядра), об’єднанням, повністю зв’язаними шарами (FC) і застосовуватиме функцію Softmax для класифікації об’єктів зі значеннями ймовірності. між 0 і 1 На рисунку 2.7 показано повний процес CNN, який обробляє вхідне зображення та класифікує об’єкти на основі значення.

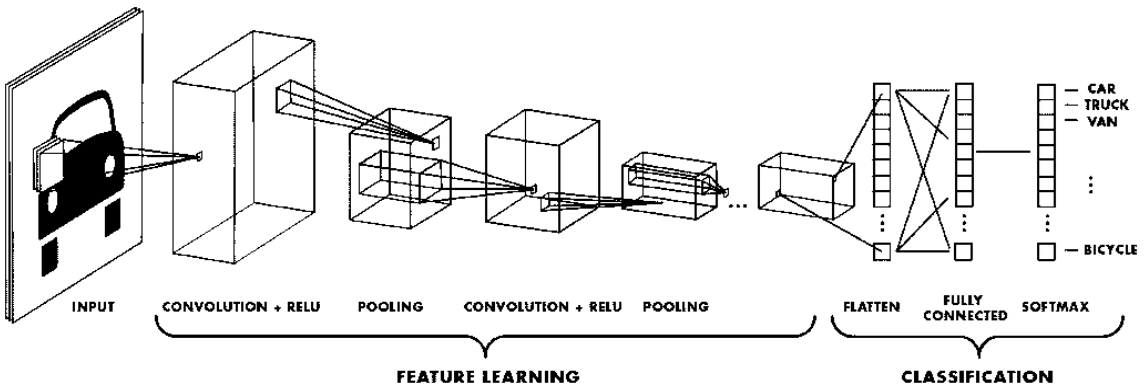


Рисунок 2.7 − Нейромережа з багатьма конволюційними шарами [18]

2.2.1 Конволюційний (згортковий) шар

Згортка - це перший шар, який витягує елементи з вхідного зображення. Convolution зберігає попіксельне з’єднання, вивчаючи особливості зображення за допомогою невеликих квадратів вхідних даних. Це математична операція, яка потребує двох вхідних даних, таких як матриця зображення та фільтр або ядро (рис. 2.8):

− матриця зображення (об’єм) розміром (h * w * d);

− фільтр розміром (fh * fw * d);

− вихідний розмір зображення (h – fh + 1) * (w – fw + 1) * 1.

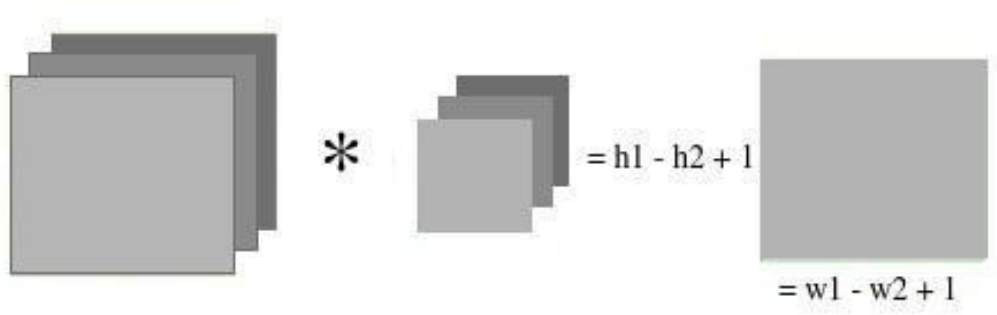


Рисунок 2.8 – Добуток матриці зображення та матриці ядра (фільтра)

Розглянемо зображення 5 х 5 зі значеннями пікселів 0, 1 і матрицею фільтра 3 х 3, як показано на рисунку 2.9:

Рисунок 2.9 – Добуток матриці зображення та матриці ядра (фільтра)

Потім згортка матриці зображення 5 х 5 множиться на матрицю фільтра 3 х 3, яка називається «картою функцій». Результат показано на рисунку 2.10:

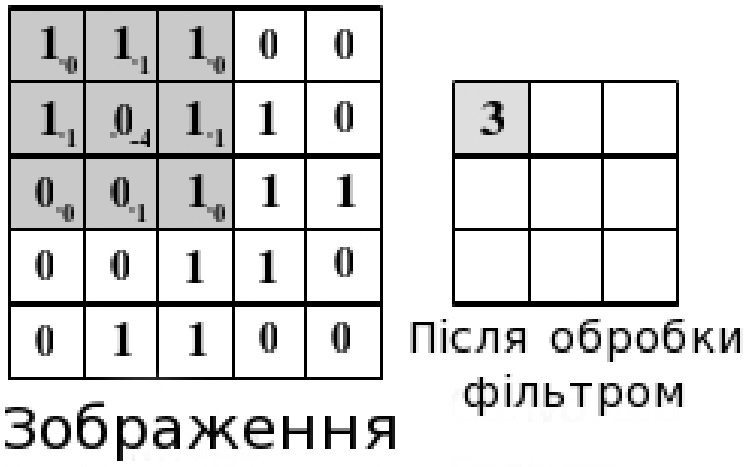


Рисунок 2.10 – Отримана матриця 3 на 3

Згортка зображення за допомогою різних фільтрів може виконувати такі операції, як виявлення країв, розмиття та підвищення різкості шляхом застосування фільтрів. На рисунку 2.11 показано ефект згортання зображення після застосування різних типів фільтрів (ядер).

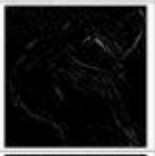
Операція	Фільтр	Оброблене зображення
Тотож-ність	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Знахо-дження кутів	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Різкість	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Квадратне розмиття	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Гауссове розмиття	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Рисунок 2.11 − Приклад роботи найбільш поширених ядер

2.2.2 Вплив кроку перекривання пік селів на функціонування CNN

Крок (Stride) – крок пікселя над вхідною матрицею. Коли крок становить 1, ми переміщуємо фільтр на 1 піксель за раз. На кроці 2 ми переміщуємо фільтр на 2 пікселі за раз і так далі. На рисунку 2.12 показано, що згортання виконуватиметься з кроком 2.

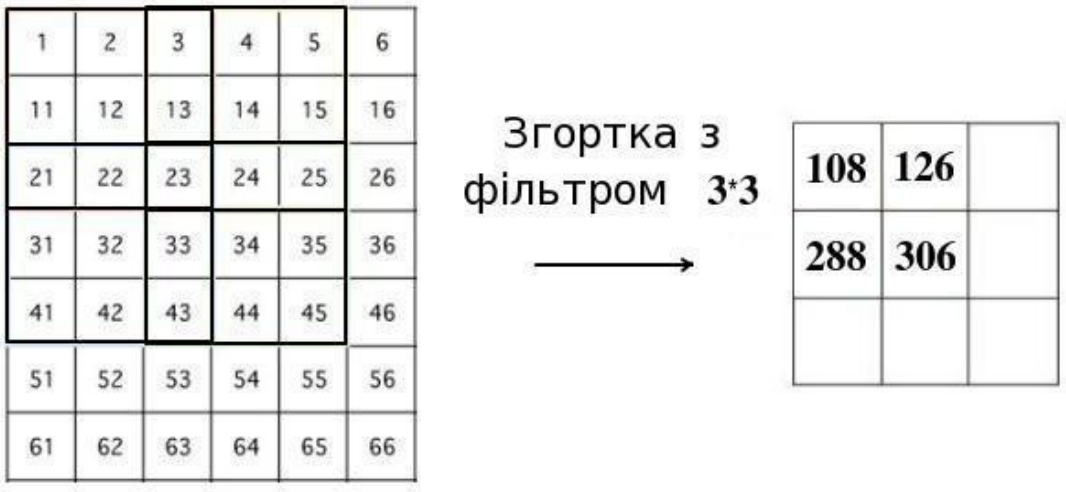


Рисунок 2.12 − Крок у 2 пікселі

2.2.3 Доповнення зображень (Padding)

Іноді фільтр не підходить для вхідного зображення. Є два варіанти вирішення цієї проблеми:

− додати до зображення нульові відступи, щоб зробити його відповідним;

- Відкиньте частини зображення, де фільтр не підходить. Це називається дійсним відступом, і воно зберігає лише дійсну частину зображення.

2.2.4 Випрямлення (ReLU)

Функція ReLU або Rectifier (Rectified Linear Unit) для вирішення нелінійних проблем. Вихід $f(x) = \max(0,x)$.

Чому ReLU важливий: Метою ReLU є впровадження нелінійності в нашу ConvNet. Оскільки наша мета полягає в тому, щоб наша ConvNet навчалася, це вимагає, щоб обробка даних була нелінійною.

Результат використання ReLU показано на рисунку 2.13:

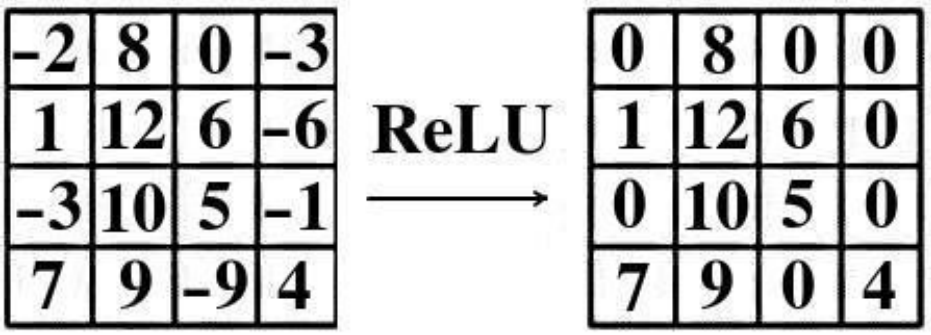


Рисунок 2.13 − Операція випрямлення

Існують інші нелінійні функції, такі як tanh або sigmoid, які можна використовувати замість ReLU. Більшість спеціалістів із обробки даних використовують ReLU, тому що продуктивність ReLU ефективніша за інші.

2.2.5 Об’єднання (Pooling)

Об’єднання шарів зменшує кількість параметрів, коли зображення занадто велике. Об’єднання, також відоме як субдискретизація або зменшення дискретизації, зменшує розмірність кожного зображення, але зберігає важливу інформацію. Асоціація може працювати на різних підставах:

− максимальне об'єднання;

− середня однорідність;

- Комбінуйте за кількістю.

Об’єднання за максимумом знаходить найбільший елемент серед чисел вибраної матриці (рис. 2.14). Функція «Об’єднання за середнім» знаходить середнє значення всіх чисел у матриці. Об'єднання за сумою дає суму всіх чисел вибраної матриці.

Рисунок 2.14 – Об’єднання за принципом «максимум»

2.2.6 Повнозв’язний шар

Цей рівень, також відомий як FC-шар, перетворює матрицю на вектор і передає її на повністю пов’язаний рівень, як нейронна мережа (рис. 2.15).

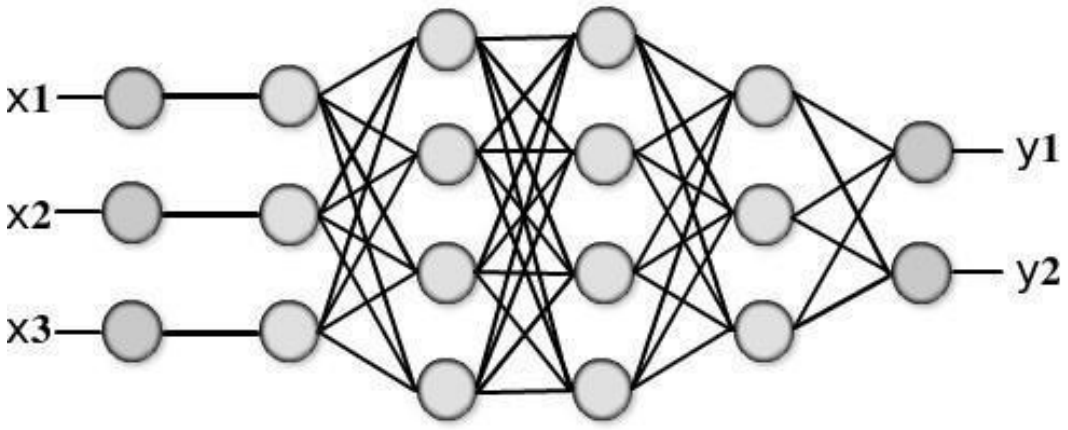


Рисунок 2.15 − Після шару об’єднання реалізується повнозв’язний шар

На зображенні нижче матрицю ознак буде перетворено на векторне представлення (x1, x2, x3, ...). Повністю з’єднані шари поєднують ці функції для створення моделі. Нарешті, є функції активації, такі як softmax або sigmoid, щоб класифікувати результати як котів, собак, автомобілів, вантажівок тощо.

Щоб налаштувати CNN для роботи, вхідне зображення потрібно подати на згортковий шар. Виберіть параметри, застосуйте фільтри кроками, прокладіть, якщо потрібно. Виконайте згортку на зображенні та застосуйте активації ReLU до матриці. Виконайте об'єднання, щоб зменшити розмірність. Додайте якомога більше згорткових шарів, поки результат не буде задовільним. Перетворіть вихідні дані в одновимірний вектор і надішліть його на повністю пов'язаний шар (FC Layer). Отримайте класи та класифікуйте зображення за допомогою функції активації (логістична регресія з функцією вартості).

2.3 Алгоритми навачання (методи градієнтного спуску)

Що таке алгоритм оптимізації? Алгоритми оптимізації (АО) – це алгоритми, завданням яких є мінімізація (або максимізація) функції втрат. Функція втрат – це звичайна математична функція, параметри якої є навчальними параметрами моделі. У свою чергу, навчальними параметрами моделі є коефіцієнти, які використовуються при обчисленні цільового значення (Y) із набору вхідних параметрів

(X) Нейронна мережа. Наприклад, значення вагових коефіцієнтів (W) і помилок (b) нейронної мережі є параметрами її навчання, які використовуються при обчисленні вихідних значень і оновлюються в напрямку оптимального рішення, тобто мінімізації втрати під час навчання мережі.

Внутрішні параметри моделі відіграють дуже важливу роль для ефективного навчання моделі та отримання точних результатів. Ось чому різні стратегії та АО застосовуються для оновлення та розрахунку відповідних та оптимальних значень параметрів для таких нейронних мереж.

Алгоритми оптимізації в основному поділяються на дві категорії:

Методи першого порядку - ці методи спрямовані на мінімізацію або максимізацію функції втрат E(x) за допомогою значень градієнта щодо параметрів. Найпоширенішим алгоритмом оптимізації в цій категорії є градієнтний спуск. Перша похідна показує, спадає чи зростає функція в даній точці.

Що таке градієнт функції? Градієнт - це просто вектор, який є багатовимірним узагальненням похідної (dy / dx). Різниця полягає в тому, що градієнт використовується для обчислення похідної функції, яка залежить від кількох змінних або кількох змінних. Використовуйте часткові похідні для обчислення градієнтів. Крім того, ще одна ключова відмінність між градієнтом і похідною полягає в тому, що градієнт функції створює векторне поле.

Градієнт, представлений матрицею Якобі, є матрицею часткових похідних першого порядку (градієнтів).

Методи другого порядку. Ці методи використовують похідну другого порядку (також відому як Гессе) для мінімізації або максимізації функції втрат відповідно. Матриця Гессе є матрицею часткових похідних другого порядку. Оскільки обчислення другої похідної є дорожчим, ніж обчислення першої похідної, вона менше використовується. Друга похідна вказує нам, зростає чи спадає перша похідна, що говорить нам про кривизну функції.

Хоча пошук та обчислення похідних Гессе може бути дещо дорогим, технічна перевага цього методу оптимізації полягає в тому, що він не ігнорує та не ігнорує кривизну поверхні. Іноді методи АО другого порядку можуть перевершувати методи градієнтного спуску (тобто часткові похідні першого порядку), оскільки вони не застряють поблизу сідлової точки, тоді як градієнтний спуск може застрягти, що є однією з найбільших проблем.

2.3.1 Метод градієнтного спуску

Градієнтний спуск є найважливішою технікою та основою для навчання та оптимізації інтелектуальних систем.

Формула для оновлення параметрів за допомогою градієнтного спуску:

$$\theta = \theta - \eta \nabla J(\theta)$$

де η – швидкість навчання, а $\nabla J(\theta)$ – градієнт функції втрат $J(\theta)$.

Це найпопулярніші алгоритми оптимізації в оптимізації нейронних мереж. Градієнтний спуск в основному використовується для оновлення вагових коефіцієнтів у нейронних мережах, тобто оновлення та коригування параметрів моделі для мінімізації функції втрат. Використовуючи техніку зворотного поширення, коли обчислення добутку вхідного сигналу та відповідних ваг спочатку поширюється вперед, а потім застосовується функція активації, яка перетворює вхідний сигнал у вихідний, з акцентом на несуттєвих використання лінійного перетворення, яке дозволяє моделі вивчати практично будь-яку довільну функціональну карту.

Потім, використовуючи зворотне поширення, поширте значення втрати та оновіть ваги за допомогою градієнтного спуску. де розраховується градієнт функції втрат E відносно ваг B, а параметри оновлюються в протилежному напрямку до градієнта функції втрат параметрів моделі.

Розглянемо варіанти градієнтного спуску.

Традиційний градієнтний спуск обчислює весь набір даних, однак обробка наборів даних, які є дуже великими та не можуть поміститися в пам’яті, може бути дуже повільною та складною. Розмір кроку градієнта визначає коефіцієнт η і відповідає за апроксимацію глобального мінімуму опуклої поверхні помилки та локального мінімуму опуклої поверхні. Крім того, при використанні цього типу градієнта виникає проблема обчислення надлишкових оновлень для великих наборів даних.

Зазначена вище проблема регулярного градієнтного спуску розв’язується в стохастичному градієнтному спуску.

Стохастичний градієнтний спуск (SGS) – лише один об’єкт випадковим чином вибирається із вибірки за одну ітерацію. Зазвичай це швидша техніка. Він виконує одне оновлення за раз.

Через часті оновлення параметри змінюються в широких межах, і сила флуктуації функції втрат також різна. Це може бути перевагою, оскільки така

поведінка може допомогти нам відкрити нові та, можливо, кращі локальні мінімуми.

Але проблема з SGS полягає в тому, що зрештою важко визначити точне мінімальне значення через часті оновлення та коливання.

Однак, якщо швидкість навчання η повільно знижується, SGS демонструє ту саму схему пошуку оптимального значення, що й стандартний градієнтний спуск. Визначення оптимальної швидкості за допомогою стохастичного градієнтного спуску показано на рисунку 2.16.

Оновлення параметрів з кожною ітерацією навчання призводить до різких коливань функції втрат, що може призвести до неможливості отримати найменше значення параметра, яке дає найменше значення втрат.

Міні-пакетний градієнтний спуск. Щоб уникнути всіх проблем і недоліків SGS і стандартного градієнтного спуску, використовується Mini Batch Gradient Descent, оскільки він використовує всі переваги обох методів. При такому підході кожна ітерація вибирає набір об’єктів із навчальної вибірки фіксованої довжини. У більшості випадків розмір дорівнює двойці.

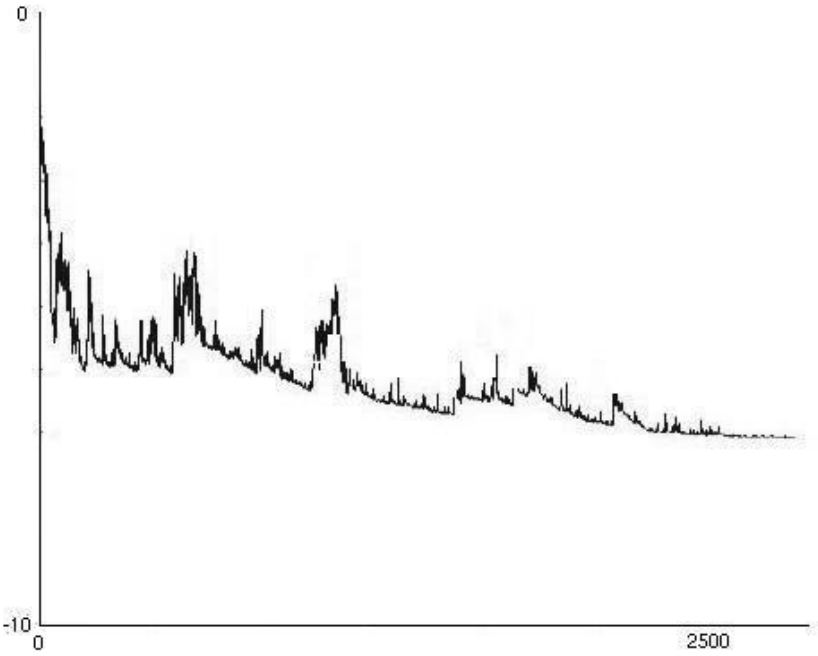


Рисунок 2.16 – Швидкість пошуку оптимуму методом градієнтного спуску

Переваги використання Mini Batch Gradient Descent:

- Це зменшує дисперсію оновлень параметрів, що зрештою може призвести до кращої та більш стабільної поведінки.

- Зазвичай розміри міні-партій становлять від 32 до 256, але вони можуть відрізнятися залежно від програми та проблеми, що вирішується.

- Сьогодні Mini Batch Gradient Descent є широко використовуваною технікою.

З градієнтним спуском і його різновидами виникає багато проблем. Серед них вибір відповідної швидкості навчання може бути важким. Занадто низька швидкість навчання призводить до повільного пошуку оптимального значення, що впливає на загальний час навчання і, таким чином, стає надто довгим. У той час як занадто висока швидкість навчання завадить знайти оптимум і спричинить коливання навколо мінімуму або навіть розбіжність.

Крім того, ще одна проблема полягає в тому, що однакова швидкість навчання застосовується до всіх параметрів.

Ще одним ключовим завданням у мінімізації характеристик функції помилок нейронної мережі є уникнення застрягання в багатьох неоптимальних локальних мінімумах. Фактично, труднощі виникають не в точці локального мінімуму, а в точці сідла, точці, де одна величина нахиляється вгору, а інша величина – вниз. Ці сідлові точки, як правило, мають однакову мінімальну помилку, що ускладнює пошук оптимального значення, оскільки градієнт близький до нуля в усіх точках.

2.3.2 Моменти другого порядку (Momentum)

Висока дисперсія ворушінь у SGD ускладнює досягнення оптимальності, тому техніку під назвою Momentum було винайдено, щоб прискорити SGD шляхом руху в правильному напрямку та згладжування ворушінь у неправильному

напрямку. Іншими словами, все, що він робить, це додає відсоток γ вектора оновлення останнього кроку до поточного вектора оновлення.

$$V(t) = \gamma V(t-1) + \eta \nabla J(\theta).$$

Нарешті, оновлюються параметри:

$$\theta = \theta - V(t).$$

Значення імпульсу γ зазвичай встановлюється рівним 0,9 або приблизно до нього. Перевага імпульсу:

- це призводить до більш швидкого і стабільного знаходження оптимального значення;

- Зменшити волатильність.

Momentum оновлює лише параметри для пов’язаних параметрів. Це зменшує непотрібні оновлення параметрів, що призводить до більш швидкої, стабільної оптимізації та менших коливань.

2.3.3 Адаптивна оцінка моментуму (Adam)

Adaptive Momentum Estimation (Adam) – ще один спосіб обчислення адаптивної швидкості навчання для кожного параметра. Адам зберігає експоненціально спадне середнє минулих градієнтів M(t), подібне до імпульсу.

Значення M(t) і V(t) першого моменту є середнім відповідно, а значення другого моменту є дисперсією нецентрованого градієнта відповідно.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Ітогова формула для оновлення значень параметрів:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Значення β1 дорівнює 0,9, 0,999 – β2, а 10e-8 – ε.

Адам добре працює на практиці та має переваги перед іншими алгоритмами адаптивного навчання, оскільки він дуже швидко знаходить оптимум, а модель навчається дуже швидко й ефективно, і він коригує кожне виникнення інших методів оптимізації. Такі проблеми, як обнулення швидкості навчання, повільне досягнення оптимальності або висока дисперсія в оновленнях параметрів, що призводить до коливань у функції втрат.

2.4 Функція втрат

У контексті алгоритмів оптимізації функція, яка використовується для оцінки ваг, є цільовою функцією, яка використовується для обчислення відхилення прогнозованих значень від справжніх значень (мітки в наборі даних).

Для розрахунку втрат необхідно мінімізувати цільову функцію, тобто знайти рішення, де сума значень цільової функції на кожній ітерації є найменшою.

Таку цільову функцію часто називають функцією витрат або функцією втрат, а обчислене значення функції втрат просто називають «втратами».

При розрахунку похибки моделі під час оптимізації необхідно вибрати функцію втрат.

Це може бути складним завданням, оскільки функція повинна фіксувати властивості проблеми та в деяких випадках визначати найважливіші функції, важливі для навчання нейронної мережі.

2.4.1 Метод максимальної правдоподібності

Метод оцінки максимальної правдоподібності (оцінка максимальної правдоподібності, або MLE) – це метод визначення найкращої статистичної оцінки параметра з вибірки навчальних даних.

Вхідними даними є навчальний набір даних однієї або кількох вхідних змінних, а для оцінки ваг параметрів моделі потрібна модель.

Перевага використання максимальної правдоподібності як основи для оцінки параметрів (ваг) моделі в нейронних мережах і машинному навчанні полягає в тому, що зі збільшенням кількості прикладів у навчальному наборі даних оцінки параметрів моделі покращуються. Це називається властивістю «послідовності».

Помилка між двома ймовірностями обчислюється за допомогою крос-ентропії відповідно до методу максимальної правдоподібності.

При моделюванні задач класифікації відображення вхідних змінних у певний клас є важливим для прогнозування ймовірності належності до кожного класу.

Тому, використовуючи максимальну ймовірність, потрібно знайти набір ваг моделі, де прогнозоване значення найменше відрізняється від фактичного значення. Це називається перехресною ентропією.

У разі регресії прогнозованих величин замість цього зазвичай використовується функція втрат середньоквадратичної помилки (MSE).

Однак у рамках оцінки максимальної правдоподібності цільова змінна використовує розподіл Гауса, а середню квадратичну помилку можна розглядати як перехресну ентропію між прогнозованим розподілом моделі та розподілом цільової змінної.

Насправді використання цього методу можна вважати новою віхою в глибинному навчанні, оскільки до появи нейронних мереж для класифікації використовувалася функція середнього квадрата втрат.

Максимальна правдоподібність використовується майже повсюдно, головним чином через результати, які вона дає. Нейронні мережі, які використовують функції сигмоїдної або softmax активації для класифікації на вихідному рівні, навчаються швидше та надійніше, коли використовують функцію втрат крос-ентропії.

2.4.2 Середня квадратична похибка

Втрата середньої квадратичної помилки (MSE) визначається як сума квадратів усіх різниць між мітками та прогнозованими значеннями, поділена на загальну кількість навчальних даних.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\left(y_i - \hat{y}_j\right)^2$$

Незалежно від знаку очікуваних і фактичних значень, результат завжди позитивний, з ідеальним значенням 0,0. Значення втрат мінімізується, хоча його можна використовувати для максимізації процесу оптимізації, роблячи оцінку негативною. Графічне відображення середньоквадратичної похибки показано на рисунку 2.17:

Рисунок 2.17 – Середня квадратична похибка

2.4.3 Перехресну ентропію втрат (Cross−Entropy Los, Log Loss)

Перехресну ентропію втрат часто називають "перехресною ентропією", «Втрата журналу», «Логістична втрата» або «Втрата журналу».

Кожна прогнозована ймовірність порівнюється з фактичним вихідним значенням класу (0 або 1), і розраховується оцінка, яка визначає величину помилки на основі відстані від очікуваного значення. Оцінки є логарифмічними: невеликі різниці (0,1 або 0,2) отримують невеликі бали, а великі різниці (0,9 або 1,0) – великі.

Ентропія перехресних втрат зведена до мінімуму, де менші значення представляють кращі моделі, ніж більші. Модель, яка передбачає ідеальні значення, має крос-ентропію 0,0.

Перехресна ентропія для задачі двійкового передбачення фактично обчислюється як середня перехресна ентропія всіх значень.

Однак важливо додати невелике значення (наприклад, 1E-15) до прогнозу, щоб уникнути значення ентропії 0,0, яке означає, що модель перетренована. Це

означає, що на практиці оптимальним значенням ентропії буде значення, дуже близьке до нуля, але не дорівнює нулю.

Перехресну ентропію можна використовувати для класифікації кількох класів. Тобто обчислити ентропію для кожного класу. Для кожної категорії є можливість належати чи не належати до неї. Перехресна ентропія обчислює значення членства для кожного класу, а потім підсумовує всі класи моделі.

2.4.4 Середня абсолютна похибка

Середня абсолютна похибка (MAE) – вимірює середню похибку між усіма прогнозованими значеннями, незалежно від напрямку поширення. Це середнє значення абсолютної різниці між прогнозованим і фактичним значенням, коли всі індивідуальні відмінності однаково зважені.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_i - \hat{y}_j|$$

Крім того, середня абсолютна похибка використовується для зменшення розмірності вхідних даних. У разі використання середньої абсолютної похибки встановіть ваговий коефіцієнт для кожного вхідного параметра. І чим вищий коефіцієнт, тим більший вплив має параметр на прогнози моделі. Отже, чим менший параметр, тим менше значення і, отже, менш важливий параметр.

Ці значення можна використовувати для визначення найменш важливих параметрів, тому їх можна усунути, щоб зменшити розмірність вхідних параметрів моделі, спрощуючи обчислення та скорочуючи час обчислень. Графічне відображення середньої абсолютної похибки показано на рисунку 2.18:

Рисунок 2.18 – Середня абсолютна похибка

Проблема полягає в тому, щоб вибрати найкращу архітектуру та найкращий оптимізатор для того, щоб нейронна мережа швидко досягла найкращого стану, а також вивчити та налаштувати внутрішні параметри належним чином, щоб мінімізувати функцію втрат.

У нейронних архітектурах найкращим рішенням є згорточна нейронна мережа, оскільки вхідними даними є відео, з якого потрібно отримати та обробити набір функцій. Тобто підходять як RNN, так і CNN, але оскільки відео є потоковим і залежить від часу, RNN не підходить. Тому CNN – найкращий вибір.

З точки зору функцій оптимізації, Adam добре працює на практиці та перевершує інші адаптивні методи.

Такі методи, як SGD, NAG і Momentum, працюють погано, якщо вхідні дані розріджені. Для розріджених наборів даних слід використовувати один із адаптивних методів навчання. Ще одна перевага полягає в тому, що немає необхідності регулювати швидкість навчання.

3 ПРОЕКТУВАННЯ СИСТЕМИ ПЕРЕКЛАДУ МОВИ ЖЕСТІВ З ПОТОКОВОГО ВІДЕО НА АНГЛІЙСЬКУ

Архітектура розробленої нейронної мережі виглядає наступним чином:

а) Максимальний згортковий шар + комбінований шар;

б) максимальний згортковий шар + комбінований шар;

в) згортковий шар + максимальний уніфікований шар;

г) сплющення;

д) щільний;

f) Шар випадання.

Параметри шару показані на рисунку 3.1:

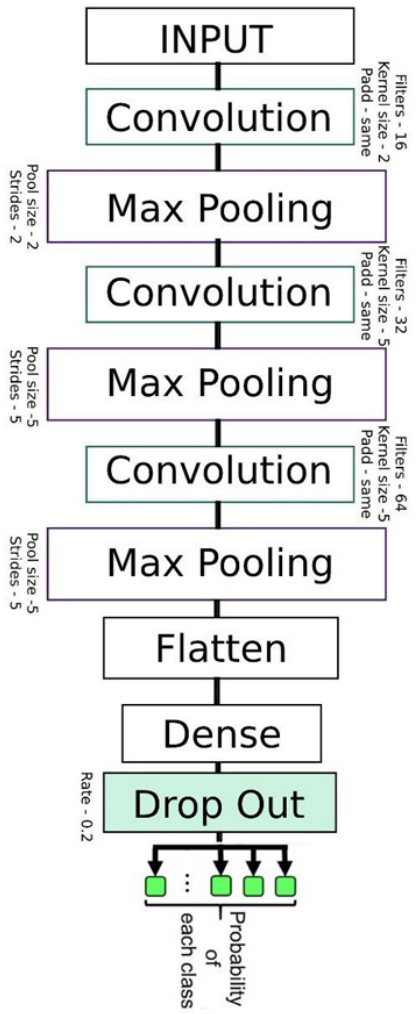


Рисунок 3.1 − Архітектура розробленої нейромережевої системи

Архітектура використовує функцію активації Softmax.

Навчання нейронної мережі включає наступні кроки: а) Завантаження набору даних;

б) створити модель;

в) запустити модель;

г) запустити функцію втрат д) запустити оптимізатор;

f) Модельне навчання.

Як набір даних використовується база даних із жестами. Через обмеження доступних пристроїв кількість жестів, які використовуються для навчання, обмежена. Однак, якщо це необхідно і обладнання, необхідне для подальшого навчання, є, просто поповніть базу даних і розмістіть модель для подальшого навчання.

3.1 Створення нейромережевої моделі

Базова формула для розрахунка розмірності конволюційного шару:

$$O = W - K + 2PS + 1,$$

де O - вихідна висота/ширина, W - вихідна висота/ширина, K - розмір фільтра (розмір ядра) (2, 3, 5 для кожного шару в розробленій моделі), P - однакове заповнення (ненульове), S - крок = 1

Вихідна формула для розрахунку розмірів шару об'єднання:

$$O = W - KS + 1,$$

де O - вихідна висота/ширина, W - вихідна висота/ширина, K - розмір фільтра (розмір ядра) (2, 3, 5 для кожного шару в розробленій моделі), S - розмір кроку = розмір фільтра

Модель ініціалізується таким чином:

model, callbacks_list = cnn_model()

Розроблена модель використовує CrossEntropy Loss як функцію витрат:

$$loss = 'categorical\_crossentropy', optimizer = sgd, metrics = ['accuracy']$$

Для оптимізації використовуються такі параметри:

1) Формула оптимізації:

$$\theta = \theta - \eta \nabla \theta,$$

де θ – параметр, η – швидкість навчання (швидкість кроку навчання), а $\nabla\theta$ – градієнт.

2) Спрощений запис формули оптимізації:

$$p = p - \alpha * \nabla \theta$$

де p – параметр, α – швидкість навчання, а $\nabla\theta$ – градієнт.

Оптимізація відбувається на кожній ітерації.

Оптимізації в розроблених системах:

$\alpha = 1e^{-2}$

sgd = optimizers.SGD(lr=$1e^{-2}$)

У загальному вигляді підсумовуючи все вище написане, виходить така послідовність операцій:

Крок 1. Перетворення вхідного потоку на тензор за допомогою градієнта Крок 2. Введення: (1, 640, 280);

Крок 3. Пряме поширення: (1, 640 * 280); Крок 4. Очистити градієнт;

Крок 5. Розрахувати дані Крок 6. Отримати комісію;

Крок 7. Нанесіть градієнт;

Крок 8. Використовуйте градієнти для зміни параметрів;

Крок 9.

Крок 10. Повторіть ітерацію (повторіть з кроку 5).

3.2 Підготовка даних для навчання моделі

Відеоролики надходять на вхід системи. Для отримання результатів сегмент обробляється для виявлення основних ознак. Послідовність дій наведена на рисунку 3.2.

Рисунок 3.2 – Підготовка даних для навчання моделі

Під час підготовки даних, отриманих із відео, спочатку нормалізуйте вхідне зображення та зменшіть його до розміру 640*480 пікселів:

cv2.resize(img, (640, 480))

Потім змініть тип зображення на двійковий. Потім ми перетворюємо зображення в тип BGR, що ми робимо за допомогою функції:

cv2.COLOR_BGR2HSV

Крім того, для видалення шуму використовуються Gaussian Blur і Average Blur:

cv2.GaussianBlur cv2.medianBlur

Після цього об'єднайте всі три шари зображення в один:

cv2.merge

Для полегшення обробки зображення зменшено до відтінків сірого:

cv2.cvtColor

Після первинної обробки зображення з нього легко отримати контури, використовуємо функцію:

cv2.findContours

3.2.1 Очистка карду зображення від шуму та його бінарізація

При визначенні контуру руки спочатку потрібен чіткий контур руки. Для цього необхідно очистити відеопотік від шумів.

Оскільки обробка відео в нейронних мережах відбувається покадрово, то відео краще також покадрово очищати від шуму. Щоб видалити шум, спочатку потрібно визначити сам шум. Для цього розроблена система порівнює попередній кадр з поточним кадром, поточний кадр з наступним кадром і так далі. Порівнювати – це знаходити відмінності між кадрами 3.1:

$$F_t(x, y) = |f_t - f_{t+1}|, \quad (3.1)$$

де Ft – кадр у момент часу t, а Ft (x, y) – різниця між двома кадрами.

Щоб визначити шум, необхідно знайти відношення всіх середніх.

Різниця в розмірі рами. Коефіцієнт розраховується за такою формулою:

$$T = \begin{cases} 0.05 * \dfrac{\sum F_t}{n(F_t) * w * h}, \text{якщо } 0.05 * \dfrac{\sum F_t}{n(F_t) * w * h} \leq 1 \\ 0.2, \text{в іншому випадку} \end{cases} \quad (3.2)$$

Серед них T – значення шуму, FT (x, y) – різниця між двома кадрами, h – висота кадру, W – ширина кадру, а n (ft) – кількість FT.

Коефіцієнти 0,05 і 0,2 визначаються дослідним шляхом.

Знаючи значення шуму, розраховане за формулою 3.2, кадр тепер можна використовувати.

$$B_t = \begin{cases} 1, \text{якщо } F_t \geq T \\ 0, \text{в іншому випадку} \end{cases}, \quad (3.3)$$

де Bt є системою подвійного значення часу T.

3.3 Порівняння моделей з різною кількістю прихованих шарів

Серед них BT є системою подвійного значення часу T. При розробці архітектури розв’язування статті було проаналізовано три архітектури згорткових нейронних мереж. Різниця між цими архітектурами полягає в кількості прихованих шарів. Виберіть згортковий шар як основний, оскільки шар об’єднання залежить від згорткового шару найбільшою мірою за будь-яких обставин.

Тому розглядається така комбінація:

a) 2 згорткових шари, тому існує до 2 уніфікованих шарів; b) 3 згорткових шари, тобто максимум 3 уніфікованих шарів; C) 4 згорткових шари, тому існує до 4 комбінованих шарів.

Параметрами для порівняння є основні показники точності, швидкості обробки відеопотоку та навантаження на систему.

Результати порівняння різних схем архітектури показано на рисунку 3.3:

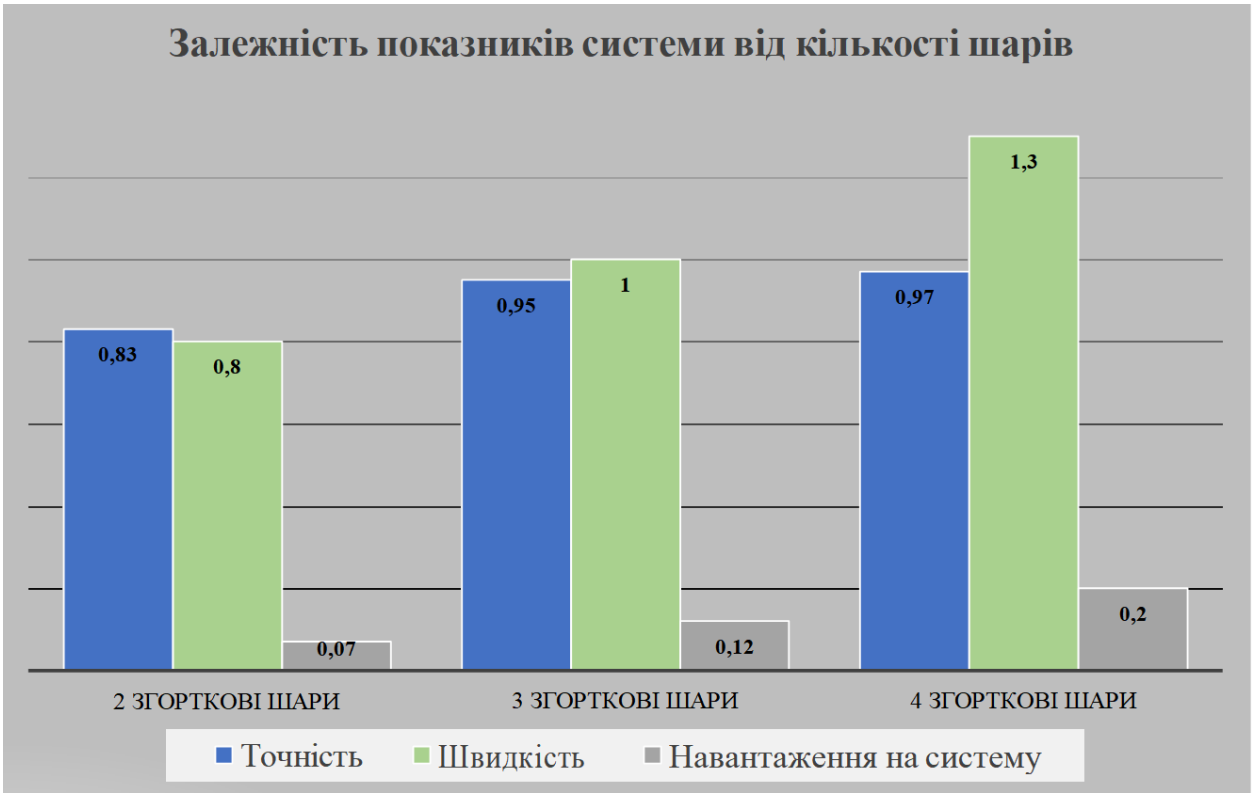


Рисунок 3.3 – Діаграма порівняння ефективності конвуляційних нейромереж для різної кількості згорткових шарів

З порівняння видно, що архітектура з 4 згортковими шарами забезпечує найвищу точність, а архітектура з двома згортковими шарами забезпечує найвищу швидкість обробки та мінімальне навантаження на систему. Однак помічено, що точність неприйнятна в архітектурі з двома згортковими шарами. У 4-рівневої архітектури, хоча точність найвища, навантаження на систему найбільше. Порівняно з архітектурою з 3 згортковими шарами, він на 8% більший, а рівень точності лише на 2% вищий.

Враховуючи всі вищезазначені ситуації, найкращим вибором є архітектура з 3 згортковими шарами, яка використовується в системі.

3.4 Налаштування гіперпараметрів

Важливим моментом розробки будь-якої нейронної мережі є вибір суперпараметра. Суперпараметр - це специфічний параметр моделі, який налаштовує і контролює модель перед навчанням.

У цій моделі суперповторне використання – це швидкість навчання, коефіцієнт виявлення шуму, прихований шар і коефіцієнт вилучення.

Швидкість навчання (α) впливає на швидкість зменшення градієнта, близького до локального мінімального значення. У цій моделі швидкість навчання становить 1E-2. Для визначення використовуйте метод пошуку в сітці.

Коефіцієнт підтвердження шуму визначається експериментально.

Кількість прихованих шарів вибирається для 3 згорткових шарів і 3 шарів об’єднання. Визначити через порівняльні показники продуктивності: швидкість обробки, точність і завантаження системи.

Коефіцієнт випадання вибирається рівним 0,2, і визначається пошук по сітці.

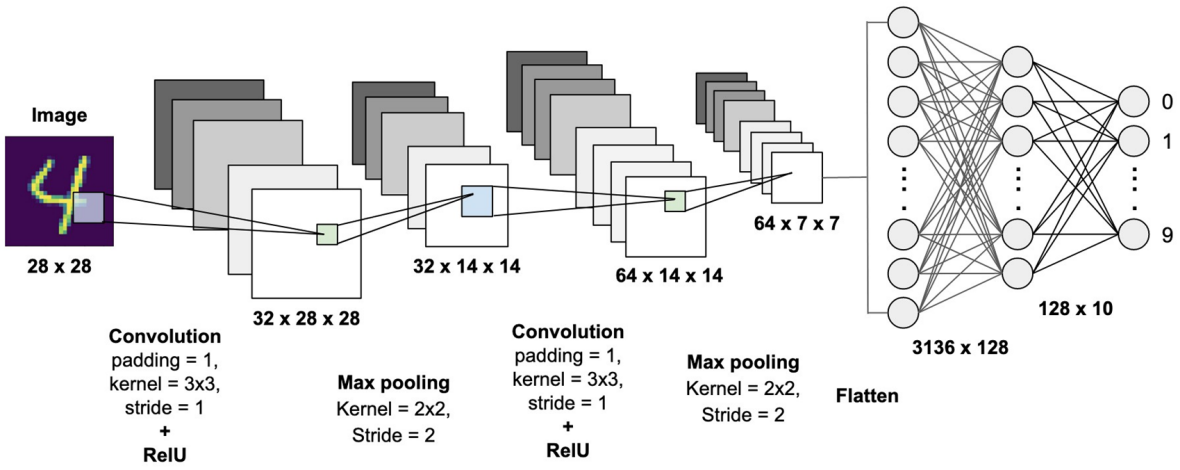


Рисунок 3.4 – Налаштована архітектура конвуляційної нейромережі

3.5 Результати тестування

Зараз ця модель може передбачати 44 знаки з американської жестової мови. Рівень точності прогнозу становить не менше 95%. Однак фон дуже важливий для точності. Що чіткіший фон, то точніший прогноз.

Для вдосконалення моделі в майбутньому ви можете додати:

-Привести словник;

-Підвищення точності визначення шумового фону.

Тому в цьому розділі описується результат розробки нейронної мережі, тобто визначення та аналіз її архітектури та супер параметрів.

Одним із надмірних спадів згорткової нейронної мережі є кількість прихованих шарів. У цій роботі аналізується продуктивність нейронних мереж з різними прихованими шарами. Основний елемент заснований на звивистих нервових прошарках. Результати показують, що найкращим вибором з точки зору точності, швидкості обробки та навантаження на систему є максимальна архітектура з 3 шарів згортки та 3 рівнів об’єднання. Ця опція фактично введена.

Виберіть CNN як тип нейронної мережі, випадкове зменшення градієнта як метод оптимізації та крос-ентропію як функцію втрат.

Видно, що створена мережа забезпечує досить високі показники точності. У порівнянні з аналогом, він може не тільки ідентифікувати літери, а й ідентифікувати слова.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту інтелектуальний людино-машинний додаток для перетворення мова жестів – текст. Задача розпізнавання мови жестів та її перекладу не реалізована на комерційному рівні. Існують дослідження з даного питання, проте всі вони закриті та мають недоліки, такі як повільна обробка, невисока точність, потреба у великих обчислювальних можливостях.

Особливістю програми є те, що дана технологія використовує вдосконалення архітектури перекладу мови жестів з потокового відео в англійську мову.

Аналогом може бути Speech(video)-to-text, ціною 120000 грн. (3000$).

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах

Продовження табл. 4.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналоги	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження табл. 4.1

Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансуванн я є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуют ься у військово-пром исловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовую
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження табл. 4.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	4	4
Наявність аналогів на ринку	4	4	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	3	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	4
Фахівці з технічної і комерційної реалізації	4	4	4
Фінансування	4	4	4
Матеріально-технічна база	4	4	4
Термін реалізації ідеї	3	3	4
Супровідна документація	4	3	4
Сума	45	43	47
Середньоарифметична сума балів	(45+43+47) / 3 = 45		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розробленого нового програмного продукту є високою, що досягається за рахунок того, що інформаційна технологія значно поліпшує умови спілкування людей з вадами слуху за рахунок вдосконалення архітектури перекладу мови жестів з потокового відео в англійську мову.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

4.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \quad , \tag{4.1}$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів в місяці, 20 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.1.

Таблиця 4.1 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	43000	2150,00	35	75250,000
Програміст	40000	2000,00	35	70000,000
Тестувальник	32000	1600,00	35	56000,000
Всього				201250,00

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розробленого програмного продукту.

4.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 15 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 15\ \% / 100\ \% \qquad (4.2)$$

$З_д = (201250,00 \cdot 15\ \% / 100\ \%) = 30187,50$ (грн.)

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$Н_з = (201250,00 + 30187,50) \cdot 22\ \% / 100\ \% = 50916,25$ (грн.)

4.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючі, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.]}. \qquad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 40000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,75 міс.

$$A_{обл} = \frac{40000}{2} \times \frac{1{,}75}{12} = 2916{,}67 \ грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки вносимо до таблиці 4.2. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$A_{н.р.} = Ц_{н.р.} * Н_а * \frac{t_{вик.}}{12} \qquad (4.5)$$

Норму амортизації $Н_а$ приймемо за 13 %.

Таблиця 4.2 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп'ютер та комп'ютерна периферія	40000	2	1,75	2916,667
Офісне обладнання	28000	4	1,75	1272,727
Приміщення	1300000	20	1,75	9479,167
Ліцензійна ОС, та спеціалізовані ліцензійні нематеріальні ресурси (Visual Studio 11, Matlab, Windows 11 Pro)	34000	-	1,75	644,583
Всього				14061,25

4.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільчих компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільчих компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п , \qquad (4.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,7 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0,9 \cdot 0,7 \cdot 8 \cdot 35 \cdot 2,01 = 354,564$ (грн.)

4.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в = 201250,00 * 105\% / 100\% = 211312,5$ (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$Н_{нзв}$ = 201250,00 * 135 % / 100 % = 271688 (грн.)

4.2.8 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$В_{заг}$ = 201250,00+30187,50+50916,25+14061,25+354,56+211312,5+271688 =
= 779769,56 грн.

4.2.9 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \qquad (5.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 779769,56 / 0,5 = 1559539 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);

- внутрішньої економічної дохідності (внутрішньої норми дохідності);
- терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.10)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

Цб – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 15000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *1000* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 3000 шт., протягом другого року – на 1800 шт., протягом третього року на 1300 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1 = (0*1000 + (15000 + 1000)*3000)* 0,8333* 0,25) * (1 - 0,18) = 7687499,693$ *грн.*

$\Delta П_2 = (0*1000 + (15000 + 1000)*(3000+1800)* 0,8333* 0,25) * (1 - 0,18) = 13119999,475$ *грн.*

$\Delta П_3 = (0*1000 + (15000 + 1000)*(3000+1800+1300)* 0,8333* 0,25) * (1 - 0,18) = 16673332,666$ *грн.*

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 37480831,83 грн.

4.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= (7687499,693/(1+0,1)[1])+(13119999,475/(1+0,1)[2])+(16673332,666/
/(1+0,1)[3]) = 6988636,08 + 10842974,8 + 12526921,6 = 30358532,47 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$PV = 2 * 1559539 = 3119078,26$ *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП – PV, \quad (4.13)$$

$E_{абс} = 30358532,47 – 3119078,26 = 27239454,21$ грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розра-хувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.14}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 27239454{,}21/3119078{,}26} - 1 = 1{,}135$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.15}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні $d = (0{,}09 \ldots 0{,}14)$;

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19 .$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{4.16}$$

$T_{ок}$ = 1 / 1,135 = 0,88 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,88 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 1559539 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,88 роки.

ВИСНОВКИ

Метою цього дослідження є покращення умов спілкування людей з вадами слуху шляхом удосконалення архітектури жестового перекладу. Як показано в даній роботі, розроблена система демонструє високу точність і швидкісні показники, задовольняючи початкове завдання.

У ході роботи вивчаються існуючі алгоритми, архітектури та методи виявлення та розпізнавання об'єктів. Крім того, під час дослідження стало зрозуміло, що необхідно більше зосередитися на виявленні меж об’єктів, а не на самому виявленні об’єктів. На основі вивченого була розроблена система жестового перекладу. Також аналізуються результати роботи.

В ході роботи виконано наступну роботу:

− Досліджено існуючі алгоритми, архітектури та методи виявлення об’єктів;

- Розроблено систему сурдоперекладу;

− Налаштувати гіперпараметри моделі;

− Аналіз результатів роботи системи;

- Робота над підвищенням точності розпізнавання та швидкості дій.

Також було проведено остаточний аналіз роботи системи для визначення виконання всіх поставлених вимог.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Video−based Sign Language Recognition without Temporal Segmentation [Електронний ресурс] / Jie Huang, Wengang Zhou, Qilin Zhang та ін.]. – 2018. – Режим доступу до ресурсу: https://arxiv.org/abs/1801.10111.

2. Vivek Bheda. Using Deep Convolutional Networks for Gesture Recognition in American Sign Language [Електронний ресурс] / Vivek Bheda, N. Dianna Radpour – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1710/1710.06836.pdf.

3. Hamid Reza Vaezi Joze. MS−ASL: A Large−Scale Data Set and Benchmark for Understanding American Sign Language [Електронний ресурс] / Hamid Reza Vaezi Joze, Oscar Koller – Режим доступу до ресурсу: https://arxiv.org/pdf/1812.01053.pdf.

4. Драгуцан А. Як працюють нейронні мережі? [Електронний ресурс] / Андрій Драгуцан // APEPS department of Igor Sikorsky KPI. – 2018. – Режим доступу до ресурсу: http://apeps.kpi.ua/neural−networks/en.

5. У.С. Мак−Каллока., В. Піттс. A logical calculus of the ideas immanent in nervous activity // Bulletin of Mathematical Biophysics – 1943. – № 5.– стр.115−133.

6. Новотарський М. А. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ: ОБЧИСЛЕННЯ [Електронний ресурс] / М. А. Новотарський, Б. Б. Нестеренко // Інститут математики НАН України. – 2004. – Режим доступу до ресурсу: http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/ShtuchnNejronMeregNester200 4.pdf.

7. Гроссберг С. Contour enhancement, short−term memory, and consistencies in reverberating neural networks // Studies in Applied Mathematics, 1973. – L11. – стр.213− 257.

8. Розенблат, Ф. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain// Cornell Aeronautical Laboratory, Psychological Review, v65 – 1958 – № 6 – стр. 386–408.

9. ЛеКун Я.., Бенджто Я. Convolutional networks for images, speech, and time− series. [Электронный ресурс] / Y. LeCun. – Режим доступу до ресурсу: http://yann.lecun.com/exdb/publis/pdf/lecun−bengio−95a.pdf.

10. MNIST Handwritten Digits Classification using a Convolutional Neural Network (CNN) [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://medium.com/@krutpatel/mnist−handwritten−digits−classification−using−a−convolutional−neural−network−cnn−af5fafbc35e9.

11. Britz D. Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs [Електронний ресурс] / DENNY Britz. – 2015. – Режим доступу до ресурсу: http://www.wildml.com/2015/09/recurrent−neural−networks−tutorial−part−1−introduction−to−rnns/20/.

12. Елман, Дж.Л.. Finding structure in time. // Cognitive Science. – 1990. – С. 179−211.

13. The Unreasonable Effectiveness of Recurrent Neural Networks [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: http://karpathy.github.io/2015/05/21/rnn− effectiveness/.

14. Hochreiter, S., Schmidhuber, J. Advances in Neural Information Processing Systems 9// MIT Press, Cambridge MA. – 1997 – С. 473−479.

15. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling [Електронний ресурс] / Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio. – 2014. – Режим доступу до ресурсу: https://arxiv.org/abs/1412.3555.

16. Greedy Layer−Wise Training of Deep Networks [Електронний ресурс] / Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle // Universit´e de Montr´eal – Режим доступу до ресурсу: https://papers.nips.cc/paper/3048−greedy−layer−wise− training−of−deep−networks.pdf.

17. I.J. Goodfellow. Generative Adversarial Networks [Електронний ресурс] / I.J. Goodfellow. – 2014. – Режим доступу до ресурсу: https://arxiv.org/abs/1406.2661.

18. Basics of the Classic CNN [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/basics−of−the−classic−cnn−a3dce1225add.

19. Стек Ю.А., Халус О.А. Метод виявлення людей та одночасне оцінювання їх поз. // Матеріали II Всеукраїнська науково−практична конференція молодих вчених та студентів – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 2019 – с. 9 – 12

20. Стек Ю.А., Халус О.А. Застосування архітектури глибокого навчання для системи розпізнавання мови жестів. // Матеріали III всеукраїнська науково−практична конференція молодих вчених та студентів – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20 − 22 листопада 2019

21. Колядич М.В. Класифікатор для автоматизованої автентифікації мовця за результатами візуалізації мовленнєвих сигналів [Електронний ресурс] / М. В. Колядич // ВНТУ. – 2022. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2020/paper/view/9538.

ДОДАТКИ

Додаток А
(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____ В'ячеслав КОВТУН
"14" грудня 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

"Інтелектуальний людино-машинний додаток для перетворення "мова жестів - текст"

08-33.МКР.002.00.000 ТЗ

Студентка групи 2АКІТ-21м

_____ Марія КОЛЯДИЧ
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник д.т.н., доцент, зав. кафедри КСУ

_____ В'ячеслав КОВТУН
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2022

1. Назва та галузь застосування

1.1. Назва – Інтелектуальний людино-машинний додаток для перетворення мова жестів - текст.

1.2. Галузь застосування – інформаційні технології.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від “14” вересня 2022 року №203

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є покращення умов спілкування людей з вадами слуху шляхом удосконалення архітектури жестового перекладу.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Vivek Bheda. Using Deep Convolutional Networks for Gesture Recognition in American Sign Language [Електронний ресурс] / Vivek Bheda, N. Dianna Radpour – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1710/1710.06836.pdf.

2. Britz D. Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs [Електронний ресурс] / DENNY Britz. – 2015. – Режим доступу до ресурсу: http://www.wildml.com/2015/09/recurrent−neural−networks−tutorial−part−1−introduction−to−rnns/20/.

3. Hochreiter, S., Schmidhuber, J. Advances in Neural Information Processing Systems 9// MIT Press, Cambridge MA. – 1997 – С. 473−479.

4. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling [Електронний ресурс] / Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio. – 2014. – Режим доступу до ресурсу: https://arxiv.org/abs/1412.3555.

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- розпізнавання жестів;
- переклад на англійську мову;
- формування результату.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 11 Pro;
- Matlab;
- Visual Studio 11.

5.2.2. Умови експлуатації системи:

- робота на веб-додатках;
- переклад мови жестів.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз систем перекладу мови жестів у потоковому відео на англійську «03»_09__ 2022 р.
2. Розробка математичного апарату нейронної мережі «05»_11_ 2022 р.
3. Розробка програмного забезпечення та експериментальні дослідження «20»_11_ 2022р.

6.2 Графічні матеріали:

1. Розробка UML-діаграми системи «10»_10_ 2022 р.
2. Розробка архітектури мережі «20»_10_ 2022 р.
3. Тестування програмного забезпечення «20»_11_ 2022 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «28»__11_ 2022 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «16»__12_ 2022 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «23»__12_ 2022р.

Додаток Б (обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Інтелектуальний людино-машинний додаток для перетворення «мова жестів – текст»»

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 99% Схожість 1%

Аналіз звіту подібності (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Галущак А.В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Колядич М.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Ковтун В.В.
(підпис) (прізвище, ініціали)

Додаток В

Лістинг програми

Файл NeuralNetwork

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.IO;

namespace iTellUtils
{
public class Vertex
{
public List<Vertex> next; public List<double> wnext; public double delta; public double value; public double threshold;

public Vertex()
{
next = new List<Vertex>(); wnext = new List<double>();
Random rand = new Random((int)DateTime.Now.Ticks); threshold = rand.NextDouble() * 0.4;
delta = 0.0;
value = 0.0;
}
}

public class Layer
{
public List<Vertex> vertex; public Layer(int size)
{
vertex = new List<Vertex>(); for (int i = 0; i < size; ++i)
vertex.Add(new Vertex());
}
}

public class NeuralNetworkException: Exception
{
public NeuralNetworkException(string message)
: base(message) { }
}

public class NeuralNetwork
```

```
{
private const double speed = 0.05; private List<Layer> layers;
static public NeuralNetwork Current = null; public int getDesision() {
double best = −1e100;
int bestv = −1; double[] x = Output();
for(int i=0; i<x.Length; ++i)
{
if (x[i] > best)
{
best = x[i]; bestv = i;
}
}
return bestv;
}
public double[] Output()
{
double[] result = new double[layers[layers.Count − 1].vertex.Count];


for(int i=0; i<result.Length; ++i)
{
result[i] = layers[layers.Count − 1].vertex[i].value;
}
return result;
}

public void Save(string filename)
{
StreamWriter sw = null; try
{
sw = new StreamWriter(filename);

sw.WriteLine(layers.Count);
for (int i = 0; i < layers.Count; ++i) sw.Write(layers[i].vertex.Count.ToString() + " ");
sw.WriteLine();

for (int i = 0; i < layers.Count; ++i)
for (int j = 0; j < layers[i].vertex.Count; ++j)
{
Vertex v = layers[i].vertex[j];
sw.Write(v.threshold.ToString() + " " + v.next.Count.ToString() + " "); for (int k = 0; k < v.next.Count; ++k)
{
```

```
sw.Write(v.wnext[k]);
sw.Write(" ");
}
sw.WriteLine();
}
}
catch (Exception e)
{
throw new NeuralNetworkException(e.Message);
}
finally
{
if (sw != null)
{
sw.Flush();
sw.Close();
}
}
}

public void Load(string filename)
{
StreamReader sw = null; try
{
sw = new StreamReader(filename); string s = sw.ReadToEnd();
string[] _ss = s.Split(' ', '\n', '\r'); List<string> ss = new List<string>(); for (int i = 0; i < _ss.Length; ++i)
if (_ss[i].Length > 0) ss.Add(_ss[i]);

int p = 0;
int n = int.Parse(ss[p++]); int[] size = new int[n];
for (int i = 0; i < n; ++i) size[i] = int.Parse(ss[p++]);

layers = new List<Layer>(); for (int i = 0; i < n; ++i)
layers.Add(new Layer(size[i]));

for (int i = 0; i < n − 1; ++i)
for (int j = 0; j < layers[i].vertex.Count; ++j)
for (int k = 0; k < layers[i + 1].vertex.Count; ++k) layers[i].vertex[j].next.Add(layers[i + 1].vertex[k]);

for (int i = 0; i < n; ++i)
```

```
{
for (int j = 0; j < size[i]; ++j)
{
Vertex v = layers[i].vertex[j]; v.threshold = double.Parse(ss[p++]); int m = int.Parse(ss[p++]);
for (int k = 0; k < m; ++k) v.wnext.Add(double.Parse(ss[p++]));
}
}
}
catch (Exception e)
{
throw new NeuralNetworkException(e.Message);
}
finally
{
if (sw != null) sw.Close();
}
}

public NeuralNetwork(string filename)
{
Load(filename);
}

public NeuralNetwork(int[] sizes)
{
Random rand = new Random((int)DateTime.Now.Ticks); layers = new List<Layer>();
for (int i = 0; i < sizes.Length; ++i) layers.Add(new Layer(sizes[i]));

for (int i = 0; i < sizes.Length − 1; ++i)
{
for (int j = 0; j < layers[i].vertex.Count; ++j)
for (int k = 0; k < layers[i + 1].vertex.Count; ++k)
{
layers[i].vertex[j].next.Add(layers[i + 1].vertex[k]); layers[i].vertex[j].wnext.Add(rand.NextDouble() * 0.4 − 0.2);
}
}
}

public void clear()
{
for (int i = 0; i < layers.Count; ++i)
for (int j = 0; j < layers[i].vertex.Count; ++j) layers[i].vertex[j].value = 0.0;
```

```
}

private double sigmoida(double x)
{
return 1.0 / (1.0 + Math.Exp(−x));
}

public int InputLayerSize { get { return layers[0].vertex.Count; } } public void Run(double[] input)
{
clear();
if (input.Length != layers[0].vertex.Count) throw new Exception("Input layer and input has different number of items");

// init first layer
for (int i = 0; i < input.Length; ++i)
{
layers[0].vertex[i].value = input[i];
for (int j = 0; j < layers[0].vertex[i].next.Count; ++j) layers[0].vertex[i].next[j].value += input[i] * layers[0].vertex[i].wnext[j];
}

// next layers
for (int t = 1; t < layers.Count; ++t)
for (int i = 0; i < layers[t].vertex.Count; ++i)
{

Vertex v = layers[t].vertex[i];
v.value = sigmoida(v.value − v.threshold); for (int j = 0; j < v.next.Count; ++j)
v.next[j].value += v.value * v.wnext[j];
}
}

public void Learn(double[] input, double[] need)
{
if (need.Length != layers[layers.Count − 1].vertex.Count)
throw new Exception("Test is something different from expected"); Run(input);
// last layer:
for (int i = 0; i < layers[layers.Count − 1].vertex.Count; ++i)
{
Vertex v = layers[layers.Count − 1].vertex[i];
v.delta = v.value * (1.0 − v.value) * (need[i] − v.value);
```

```
}
// other layers:
for (int t = layers.Count − 2; t >= 0; −−t) foreach (Vertex v in layers[t].vertex)
{
v.delta = 0.0;
for (int i = 0; i < v.next.Count; ++i) v.delta += v.next[i].delta * v.wnext[i];
v.delta *= v.value * (1.0 − v.value);
}

// learning:

for (int t = layers.Count − 1; t >= 0; −−t)
for (int i = 0; i < layers[t].vertex.Count; ++i)
{
Vertex v = layers[t].vertex[i];
for (int j = 0; j < v.next.Count; ++j)
{
Vertex y = v.next[j];
v.wnext[j] += speed * y.delta * v.value;
}
}
}
}
}
```

Файл WordDB

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.IO;
using System.Runtime.Serialization.Formatters.Binary; using System.Runtime.Serialization;

namespace iTellUtils
{

public class WordDBCantSaveException: Exception
{
public WordDBCantSaveException(string s):base(s) { }
}
public class WordDBCantLoadException : Exception
{
public WordDBCantLoadException(string s) : base(s) { }
}
```

```
[Serializable] public class WordDB
{
static private WordDB instance = null;
static public WordDB Instance { get { if (instance == null) instance = new WordDB(); return instance; } }

private SortedSet<string> words = new SortedSet<string>();

private double getProb(string s, FuzzyVector v) { if (s.Length != v.chars.Count) return 0.0; double prob = 1.0;
for(int i=0; i<s.Length; ++i)
prob *= v.chars[i][(int)(s[i]−'a')]; return prob;
}

public int WordsCount { get { return words.Count; } } private bool checkPref(string s1, string s2)
{
if (s1.Length<s2.Length) return false;
int i = 0; while (i < s2.Length) if (s1[i] != s2[i]) return false; else ++i; return true;
}
public string[] getWords(string pref)
{
string[] w = words.ToArray(); List<string> list = new List<string>(); for (int i = 0; i < w.Length; ++i)
{
if (checkPref(w[i], pref)) list.Add(w[i]);
}
return list.ToArray();
}

public void LoadFromTextFile(string filename)
{

try
{

words.Clear();
string[] w = File.ReadAllLines(filename);
for (int i = 0; i < w.Length; ++i) if (w[i].Length > 0)
{

AddWord(w[i]);
}
}
catch (Exception e)
```

```
{
throw new WordDBCantLoadException(e.Message);
}
}
public void SaveToTextFile(string filename)
{

try
{



}

string[] s = words.ToArray(); File.WriteAllLines(filename, s);

catch (Exception e)
{
throw new WordDBCantSaveException(e.Message);
}
}

public WordDB() { }
public WordDB(string[] words)
{
for (int i = 0; i < words.Length; ++i) this.words.Add(words[i]);
}
public void AddWord(string s)
{
string ss="";
for (int i = 0; i < s.Length; ++i) ss += Char.ToLower(s[i]); if (words.Contains(ss)) return;
words.Add(ss);
}
public void RemoveWord(string s)
{
if (!words.Contains(s)) return; words.Remove(s);
}

public WordDBSearchResult findBestWords(FuzzyVector word)
{



WordDBSearchResult res = new WordDBSearchResult(10); string[] W = getWords("");
```

```
for (int i = 0; i < W.Length; ++i)
{
res.AddWord(W[i], getProb(W[i], word));
}
return res;
}
}

public class WordDBSearchResult
{
private List<double> prob; private List<string> word;
public WordDBSearchResult(int limit) { prob = new List<double>();
word = new List<string>();
}

public void AddWord(string word, double prob)
{
this.prob.Add(prob); this.word.Add(word);
}
public int Count()
{return 0;}

public string GetFirstWord()
{
double best = 0.0; string res = "";
for (int i=0; i < prob.Count; ++i)
{
if (prob[i] > best)
{
best = prob[i]; res = word[i];
}
}
return res;
}
public string GetWordAt(int index)
{ return null; }
public double GetProbAt(int index)
{ return 0.0; }
}

public class FuzzyVector
{
```

```
public List<double[]> chars = new List<double[]>();
}
}
```

Файл BlockRecognizer

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Drawing;

namespace iTellUtils
{
public class BlockRecognizer
{
static private int Width; static private int Height; static int[,] bits = null;

static public bool Loaded() { return (bits != null); } static public void StartImage(int [,] img, int W, int H)
{
Width = W; Height = H; bits = img; line = 0;
column = 0;

}

static public void Reset()
{
line = column = 0;
}

public static int dx0 = 2; public static int dy0 = 0; public static int dx = 7; public static int dy = 16;

static public int line; static private int column;

static public Bitmap NextBitmapChar()
{
if ((line+1)*dy + dy0 > Height) return null; int y = line * dy + dy0;
int x = column * dx + dx0; if (x + dx > Width) {
++line; column = 0;
return NextBitmapChar();
}
Bitmap res = new Bitmap(dx, dy); int sum = 0;
for (int i = 0; i < dx; ++i) for (int j = 0; j < dy; ++j) {
//res.SetPixel(i, j, Color.Red); sum += bits[x + i, y + j];
res.SetPixel(i, j, Color.FromArgb(255 − bits[x + i, y + j], 255 − bits[x + i, y + j], 255
```

```
− bits[x + i, y + j]));
}
++column; return res;
}

static public double[] NextDoubleChar()
{
if ((line + 1) * dy + dy0 > Height) return null; int y = line * dy + dy0;
int x = column * dx + dx0; if (x + dx > Width)
{
++line; column = 0;
return NextDoubleChar();
}
double[] res = new double[dx*dy]; int sum = 0;
for (int j = 0; j < dy; ++j) for (int i = 0; i < dx; ++i)
{
res[sum++]= (double)bits[x + i, y + j]/255.0;
}
++column; return res;
}
}
}
```

Файл ImageFilter

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Drawing;

namespace iTellUtils
{
public class ImageFilter
{
static int ColorToInt(Color v)
{
return Math.Max(v.B,Math.Max(v.G,v.R));
}
static Color IntToColor(int x) {


return Color.FromArgb(x, x, x);
}
```

```
public static Bitmap DoFilteringSimple(Bitmap source) { Bitmap result = (Bitmap)source.Clone();
for (int x = 0; x < result.Width; ++x)
for (int y = 0; y < result.Height; ++y)
{
result.SetPixel(x,y,IntToColor(ColorToInt(source.GetPixel(x, y))));
}
return result;
}

public static Bitmap DoFiltering(Bitmap source)
{
Bitmap result = new Bitmap(source.Width, source.Height); int[] pixels = new int[8];
int[] dx = {−1,−1,−1,0,1,1, 1, 0};
int[] dy = {−1, 0, 1,1,1,0,−1,−1};
for (int x = 0; x < result.Width; ++x)
{
for (int y = 0; y < result.Height; ++y)
{
if (x == 317 && y == 106) { Console.WriteLine();
}
int n = 0;
for(int i=0; i<8; ++i)
{
int xx=x+dx[i]; int yy=y+dy[i];
if (xx<0 || yy<0 || xx>=source.Width || yy>=source.Height) continue; pixels[n++] =
ColorToInt(source.GetPixel(xx,yy));
}
for(int i=0; i<n−1; ++i) for(int j=0; j<n−1; ++j)
if (pixels[j]>pixels[j+1])
{

}
if (n<8) {

pixels[j]^=pixels[j+1]; pixels[j+1]^=pixels[j]; pixels[j]^=pixels[j+1];

result.SetPixel(x,y,IntToColor(ColorToInt(source.GetPixel(x,y))));
}
else
{
int X = ColorToInt(source.GetPixel(x,y));
```

```
double med = (double)(pixels[3]+pixels[4])/2.0d; double r1 = ((double)X<=med)?pixels[0]−X:X−pixels[7]; double r2 = ((double)X<=med)?pixels[1]−X:X−pixels[6]; double r3 = ((double)X<=med)?pixels[2]−X:X−pixels[5]; double r4 = ((double)X<=med)?pixels[3]−X:X−pixels[4];

if (r1>8.0 || r2>20.0 || r3>40.0 || r4>80.0)
{

}
else
{

}
}
}
}

result.SetPixel(x,y,IntToColor( (int)(1.0*med + 0.0*X + 0.5) ));



result.SetPixel(x,y,IntToColor( (int)(0.0*med + 1.0*X + 0.5) ));

return result;
}
}
}
```

Файл Recognizer

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using iTellUtils;



using System.Drawing;
using System.Drawing.Imaging;

namespace iTell
{
public class Recognizer
{
static public List<double[]> getData(int[][] images)
{
```

```
List<double[]> res = new List<double[]>(); for (int i = 0; i < images.Length; ++i)
{
double[] a = new double[NeuralNetwork.Current.InputLayerSize]; for (int r = 0; r < images[i].Length; ++r)
a[r] = (double)images[i][r] / 256.0; res.Add(a);
}
return res;
}

static public FuzzyVector GetVector(List<double[]> data)
{
FuzzyVector vect = new FuzzyVector(); for (int i = 0; i < data.Count; ++i) {
NeuralNetwork.Current.Run(data[i]);  double[] v = NeuralNetwork.Current.Output(); double sum = 0.0;
for (int j = 0; j < 26; ++j) sum += v[j]; for (int j = 0; j < 26; ++j) v[j] /= sum; vect.chars.Add(v);
}
return vect;
}

static public FuzzyVector GetVector(string word)
{
FuzzyVector vect = new FuzzyVector(); for (int i = 0; i < word.Length; ++i)
{
double[] y = new double[26]; for (int j = 0; j < 26; ++j)
y[j]=((int)(word[i]−'a')==j)?1.0:0.0;

vect.chars.Add(y);
}
return vect;
}
}
}
```

Файл MainForm

```
using System;
using System.Collections.Generic; using System.ComponentModel; using System.Data;
using System.Drawing; using System.Linq; using System.Text;
using System.Windows.Forms; using iTellUtils;
using System.IO;

namespace iTell
{
public partial class MainForm : Form
{
```

```
private ConsoleForm consoleForm = new ConsoleForm(); public void showConsole(bool yes)
{
UserSettings.Instance.ShowConsole = yes; вивестиКонсольToolStripMenuItem.Checked = yes; if (yes)
consoleForm.Show(); else
consoleForm.Hide();
}

public MainForm()
{
InitializeComponent();
}

private void LoadWordDB(string filename)
{
Console.Message("Завантаження бази слів з файлу " + filename); bool ok = true;
try { WordDB.Instance.LoadFromTextFile(filename); }
catch (WordDBCantLoadException e) { ok = false; Console.Error("LoadWordDB", "Помилка завантаження слів з файлу: " + e.Message); }
if (ok) Console.Message("База слів завантажена");
}

private void SaveWordDB(string filename)
{
Console.Message("Збереження бази слів до файлу " + filename); bool ok = true;
try { WordDB.Instance.SaveToTextFile(filename); }
catch (WordDBCantSaveException e) { ok = false; Console.Error("SaveWordDB", "Помилка збереження слів до файлу: " + e.Message); }
if (ok) Console.Message("База слів збережена");
}

private void Form1_Load(object sender, EventArgs e)
{
consoleForm.Width = 600;
consoleForm.Height = 200; Console.setForm(consoleForm);
if (UserSettings.Instance.ShowConsole)
{
consoleForm.Show(); вивестиКонсольToolStripMenuItem.Checked = true;
}
else
{
consoleForm.Hide(); вивестиКонсольToolStripMenuItem.Checked = false;
}
```

```
// todo loading words
if (UserSettings.Instance.CurrentDB != "") LoadWordDB(UserSettings.Instance.CurrentDB);
// todo loading net
if (UserSettings.Instance.CurrentNetwork != "")
NeuralNetwork.Current = new NeuralNetwork(UserSettings.Instance.CurrentNetwork); else
{
int[] sizes = new int[] { 112,100,26 }; NeuralNetwork.Current = new NeuralNetwork(sizes);
}
}

private void button1_Click(object sender, EventArgs e)
{
}

private void button2_Click(object sender, EventArgs e)
{

}

private void button1_Click_1(object sender, EventArgs e)
{
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
pictureBox1.Load(openFileDialog1.FileName);
Bitmap bits = ImageFilter.DoFilteringSimple(new Bitmap(pictureBox1.Image)); pictureBox1.Image = bits;
int [,] img = new int[bits.Width,bits.Height]; for (int i = 0; i < bits.Width; ++i)
for (int j = 0; j < bits.Height; ++j)
{
img[i, j] = 255−(int)bits.GetPixel(i, j).B;



}
BlockRecognizer.StartImage(img, bits.Width, bits.Height);
}
}

string goWork(FuzzyVector v, List<double[]> input)
{
string s = WordDB.Instance.findBestWords(v).GetFirstWord(); for (int i = 0; i < s.Length; ++i) {
int a = (int)s[i] − 'a'; double[] need = new double[26];
for (int j = 0; j < s.Length; ++j)
```

```
{
need[j] = (j == a) ? 1.0 : 0.0;
}
NeuralNetwork.Current.Learn(input[i], need);
}
return s;
}

private void button2_Click_1(object sender, EventArgs e)
{
DateTime beginTime = DateTime.Now; int total=0; BlockRecognizer.Reset();
FuzzyVector word = new FuzzyVector(); List<double[]> imgs = new List<double[]>(); StringBuilder sb = new StringBuilder();  int lastline = BlockRecognizer.line;
int last = 0; while(true) {
last = BlockRecognizer.line;
double[] x = BlockRecognizer.NextDoubleChar();
++total;
if (x == null) break;
int spaces = 0; for (int i = 0; i < x.Length; ++i) if (x[i] < 0.1) spaces++; if (spaces > x.Length * 90 / 100)
{
if (word.chars.Count > 0)
{
if (last != lastline)
{
sb.Append("\n");
}

}
}
else
{

lastline=last; sb.Append(goWork(word, imgs)+" "); word.chars.Clear();
imgs.Clear();

imgs.Add(x); NeuralNetwork.Current.Run(x);
word.chars.Add(NeuralNetwork.Current.Output());
}
}
if (word.chars.Count > 0)
{
sb.Append(goWork(word, imgs)); word.chars.Clear(); imgs.Clear();
```

```
        }
        richTextBox1.Text = sb.ToString();
        double v=(DateTime.Now − beginTime).TotalSeconds; MessageBox.Show("Затрачений час: " + v.ToString() + "
сек.\n" +
        "Всього символів: " + total.ToString() + "\n" +
        "Час обробки одного символу: " + (v / (double)total).ToString() + "
        сек/символ");
        }

        private void вивестиКонсольToolStripMenuItem_Click(object sender, EventArgs e)
        {
        showConsole(!UserSettings.Instance.ShowConsole);
        }

        private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
        {
        Close();
        }

        private void налаштуванняToolStripMenuItem_Click(object sender, EventArgs e)
        {
        SettingsForm settingsForm=new SettingsForm(this); settingsForm.ShowDialog();
        }

        private void редагуванняБазиToolStripMenuItem_Click(object sender, EventArgs e)
        {
        DBForm form = new DBForm(); form.ShowDialog();
        }

        private void імпортуватиЗФайлуToolStripMenuItem_Click(object sender, EventArgs e)
        {
        if (openFileDialog1.ShowDialog() == DialogResult.OK) LoadWordDB(openFileDialog1.FileName);
        }

        private void експортуватиВФайлToolStripMenuItem_Click(object sender, EventArgs e)
        {
        if (saveFileDialog1.ShowDialog()==DialogResult.OK) SaveWordDB(saveFileDialog1.FileName);
        }

        private void button4_Click(object sender, EventArgs e)
        {
        if (BlockRecognizer.Loaded()) { TestForm form = new TestForm(); form.ShowDialog();
```

```
}
}

private void завантажитиНейроннуМережуToolStripMenuItem_Click(object sender, EventArgs e)
{
if (openFileDialog1.ShowDialog() == DialogResult.OK)
NeuralNetwork.Current.Load(openFileDialog1.FileName);
}

private void зберегтиНейроннуМережуToolStripMenuItem_Click(object sender, EventArgs e)
{
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
NeuralNetwork.Current.Save(saveFileDialog1.FileName);
}

private void навчанняНаОсновіШаблонних СимволівToolStripMenuItem_Click(object sender, EventArgs e)
{
LearningForm form = new LearningForm(); form.ShowDialog();
}

private void button3_Click(object sender, EventArgs e)
{
if (saveFileDialog1.ShowDialog() == DialogResult.OK) {
File.WriteAllText(saveFileDialog1.FileName,richTextBox1.Text);
}
}

private void завантажитиЗображенняToolStripMenuItem_Click(object sender, EventArgs e)
{
button1_Click_1(sender, e);
}

private void зберегтиРезультатиРозпізнаванняToolStripMenuItem_Click(object sender, EventArgs e)
{
button3_Click(sender, e);
}
}
}
```

Файл LearningForm

```
using System;
using System.Collections.Generic; using System.ComponentModel; using System.Data;
```

```
using System.Drawing; using System.Linq; using System.Text;
using System.Windows.Forms; using iTellUtils;

namespace iTell
{
public partial class LearningForm : Form
{
public LearningForm()
{
InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{
if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
{
textBox1.Text = folderBrowserDialog1.SelectedPath;
}
}

Random rand = new Random((int)DateTime.Now.Ticks); private void button2_Click(object sender, EventArgs e)
{
List<double[]> bits = new List<double[]>(); List<double[]> need = new List<double[]>(); for(int i=0; i<26; ++i)
{
double[] res = new double[7*16];
Bitmap p = new Bitmap(textBox1.Text+"\\"+(char)((int)'A'+i)+".bmp"); int pos=0;
for(int y=0; y<16; ++y) for(int x=0; x<7; ++x) res[pos++] = (double)(255− p.GetPixel(x,y).B)/255.0;
bits.Add(res);
}

for(int i=0; i<26; ++i)
{
double[] a = new double[26];
for(int j=0; j<26; ++j) a[j] = (i==j)?1.0:0.0; need.Add(a);
}

StringBuilder sb = new StringBuilder(); int iter = (int)numericUpDown1.Value; int total=0;
int wins=0; progressBar1.Minimum=0; progressBar1.Maximum=iter;
for (int i = 0; i < iter; ++i)
{
progressBar1.Value = i; Application.DoEvents();
for (int a = 0; a < 100; ++a)
```

```
        {
        int x = rand.Next() % 26;

        NeuralNetwork.Current.Run(bits[x]);
        if (NeuralNetwork.Current.getDesision()==x) {sb.Append("+");++wins;} else sb.Append("o");
        ++total;
        NeuralNetwork.Current.Learn(bits[x], need[x]);
        }
        progressBar1.Value = i + 1;
        }
        richTextBox1.Text = sb.ToString() + "\n"+"["+wins.ToString()+" of "+total.ToString()+"]";
        }

        private void button4_Click(object sender, EventArgs e)


        {
        if (openFileDialog1.ShowDialog() == DialogResult.OK)
NeuralNetwork.Current.Load(openFileDialog1.FileName);
        }

        private void button3_Click(object sender, EventArgs e)
        {
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
NeuralNetwork.Current.Save(saveFileDialog1.FileName);
        }
        }
```

Додаток Г
(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

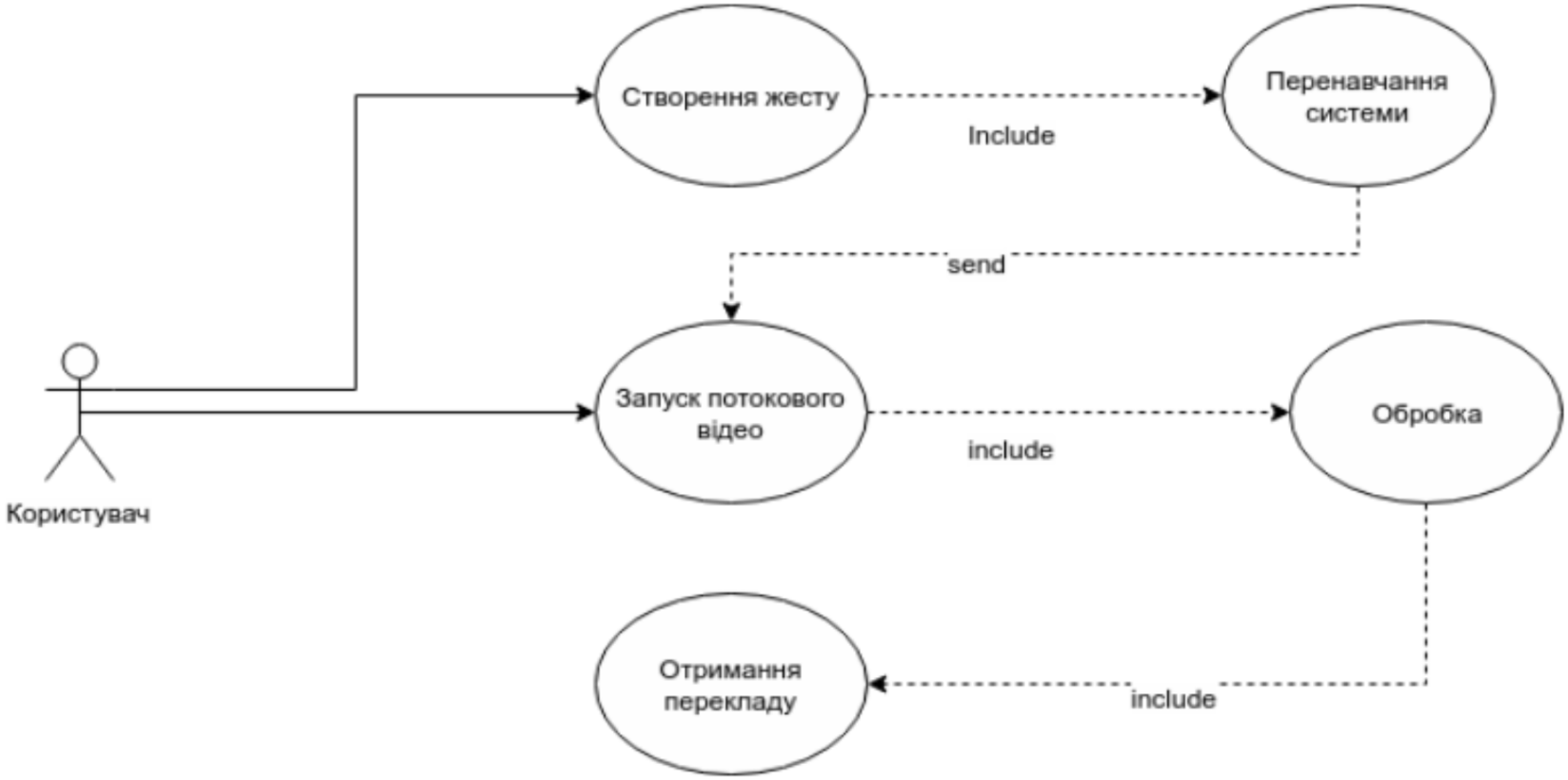
ІНТЕЛЕКТУАЛЬНИЙ ЛЮДИНО-МАШИННИЙ ДОДАТОК ДЛЯ ПЕРЕТВОРЕННЯ "МОВА ЖЕСТІВ - ТЕКСТ"

Студентка групи 2АКІТ-21м

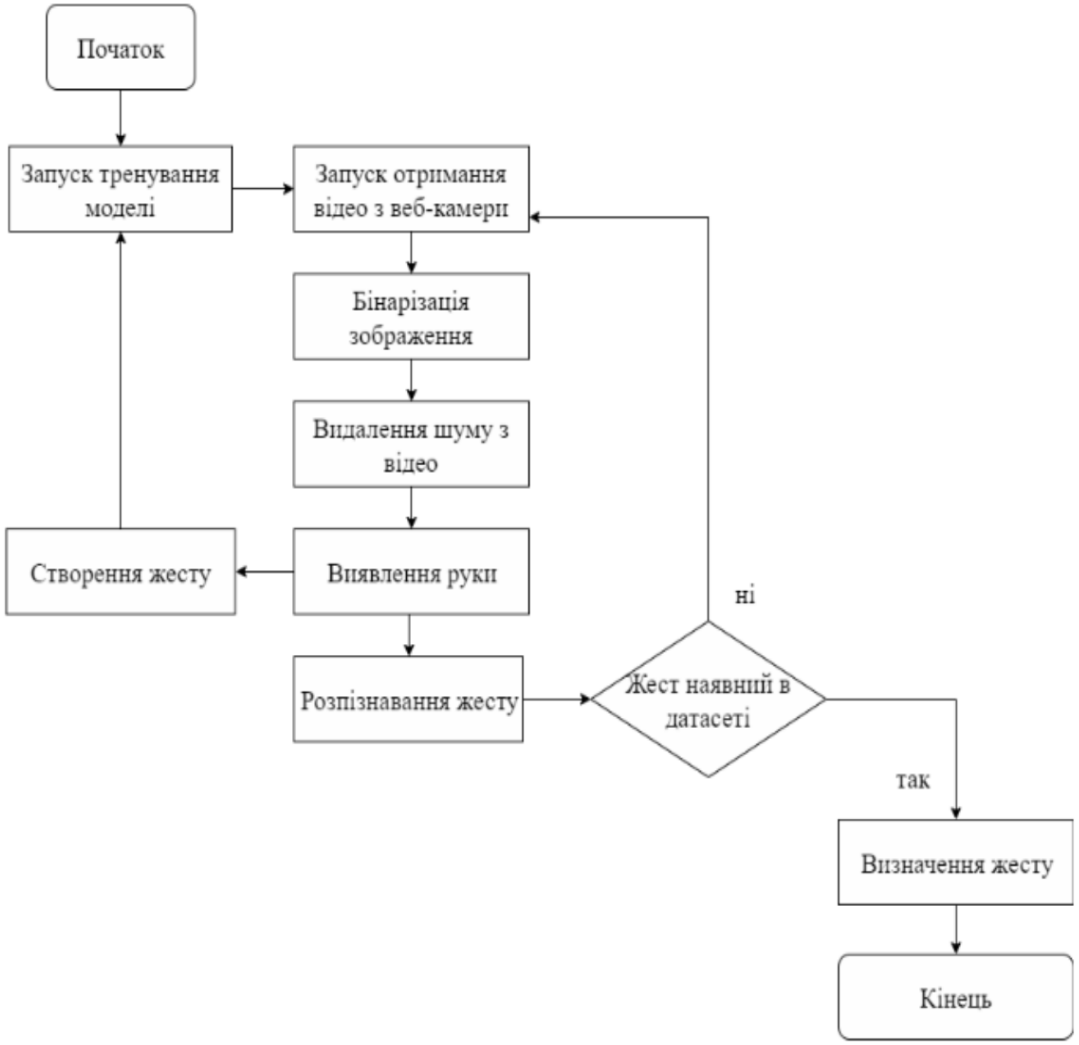
Підпис Марія КОЛЯДИЧ
Ім'я ПРІЗВИЩЕ

Керівник д.т.н., доцент, зав. кафедри КСУ

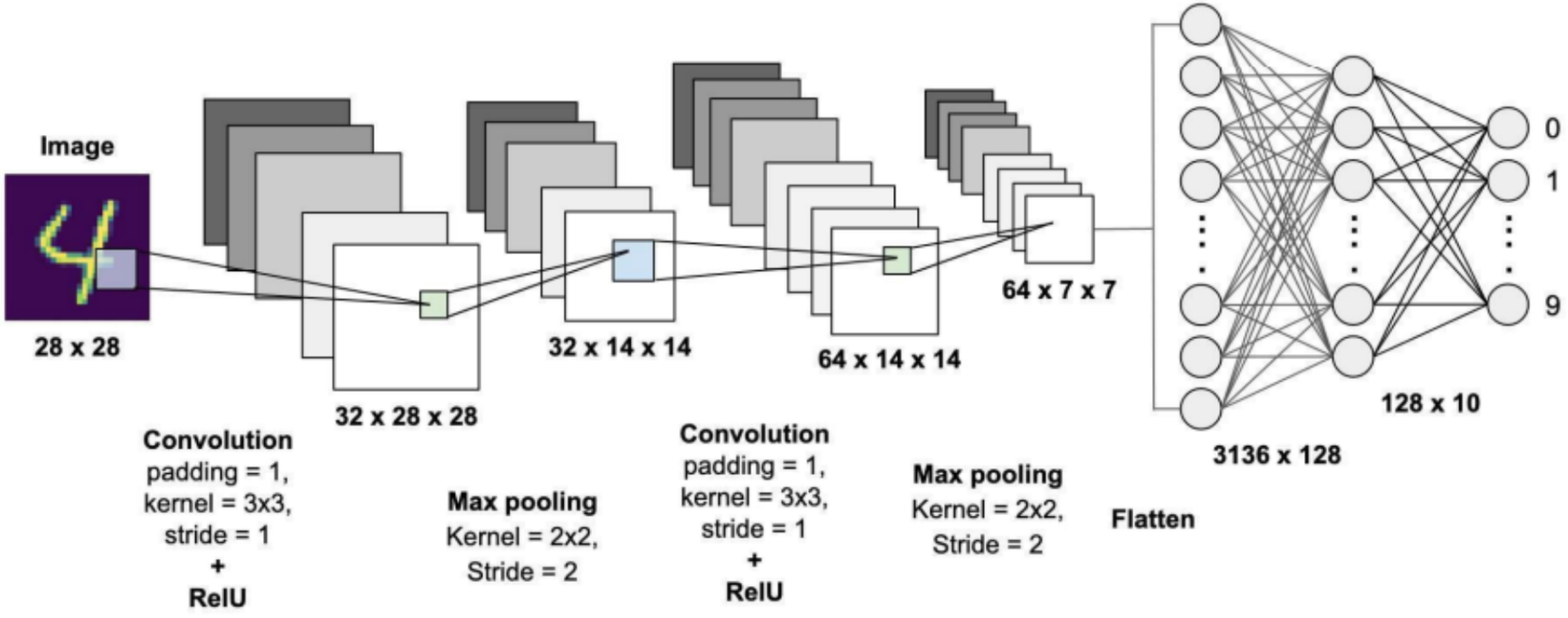
Підпис В'ячеслав КОВТУН
Ім'я ПРІЗВИЩЕ



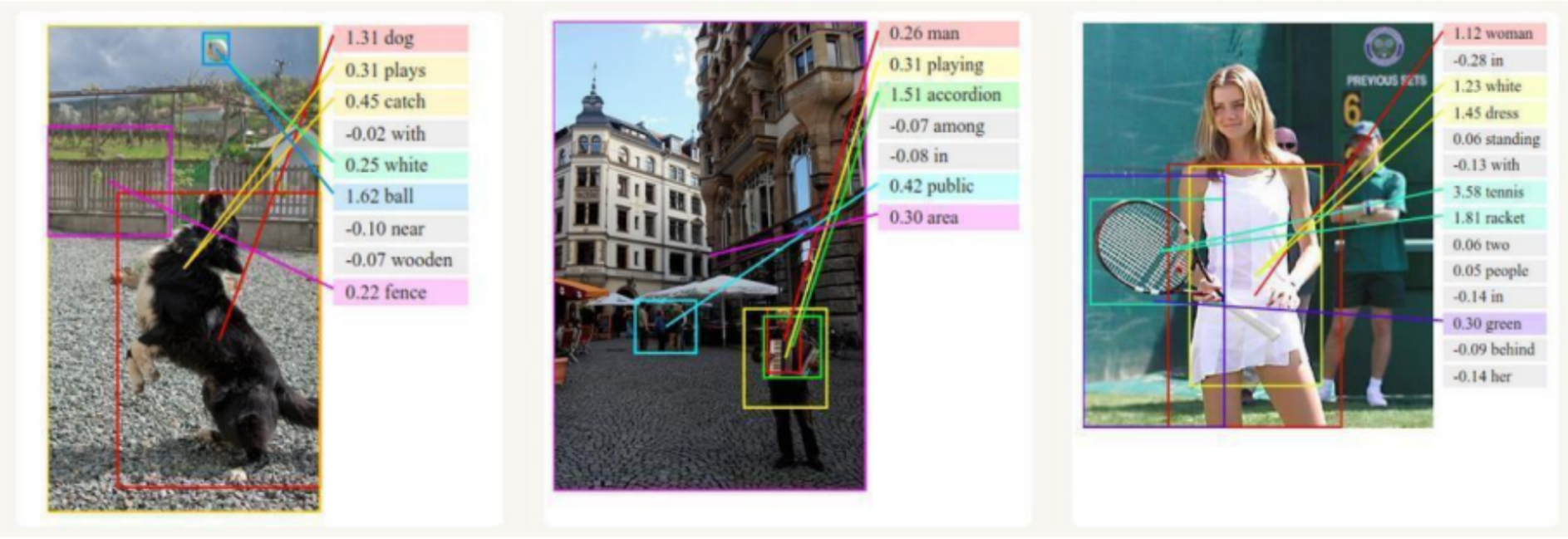
Діаграма використання системи перетворення «жест-текст»



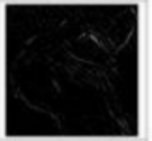
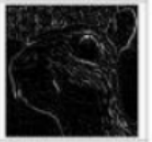
Алгоритм функціонування системи перетворення «жест-текст»



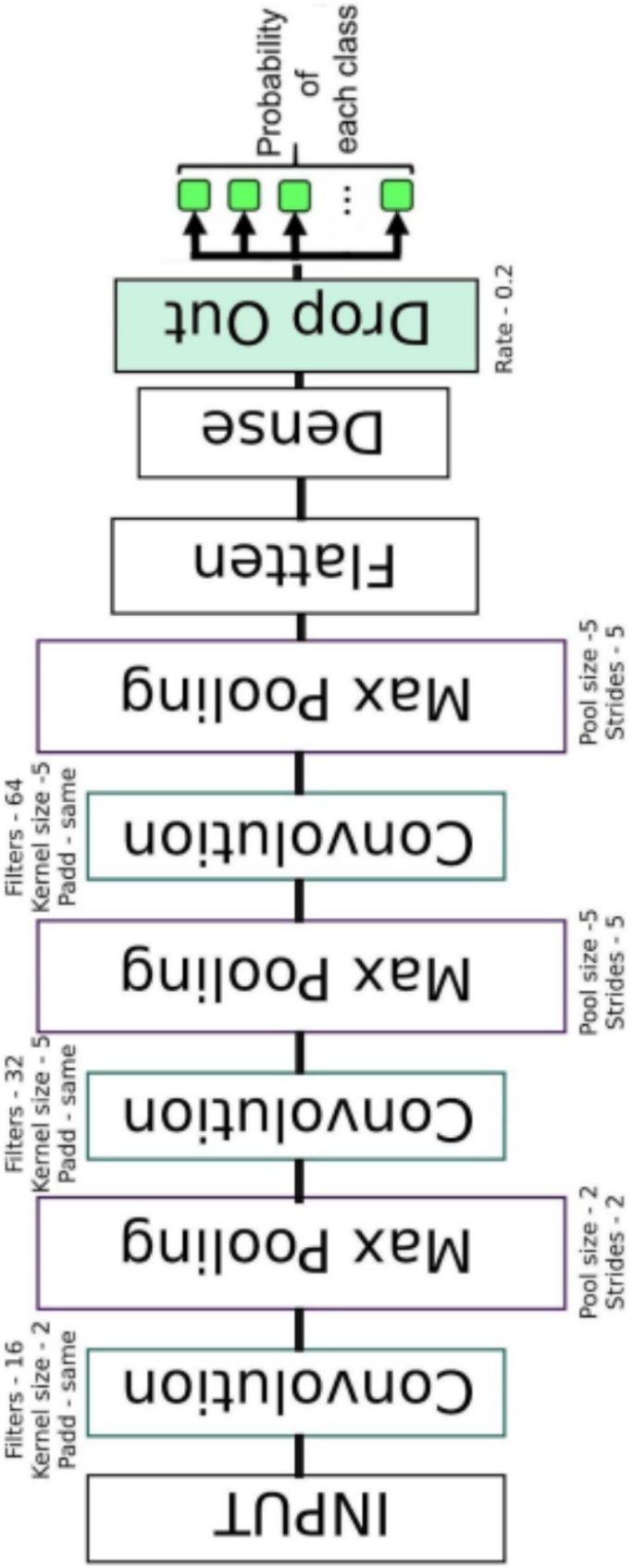
Архітектура конволюційної (згорткової) нейромережі CNN



Приклад роботи конволюційної (згорткової) нейромережі CNN

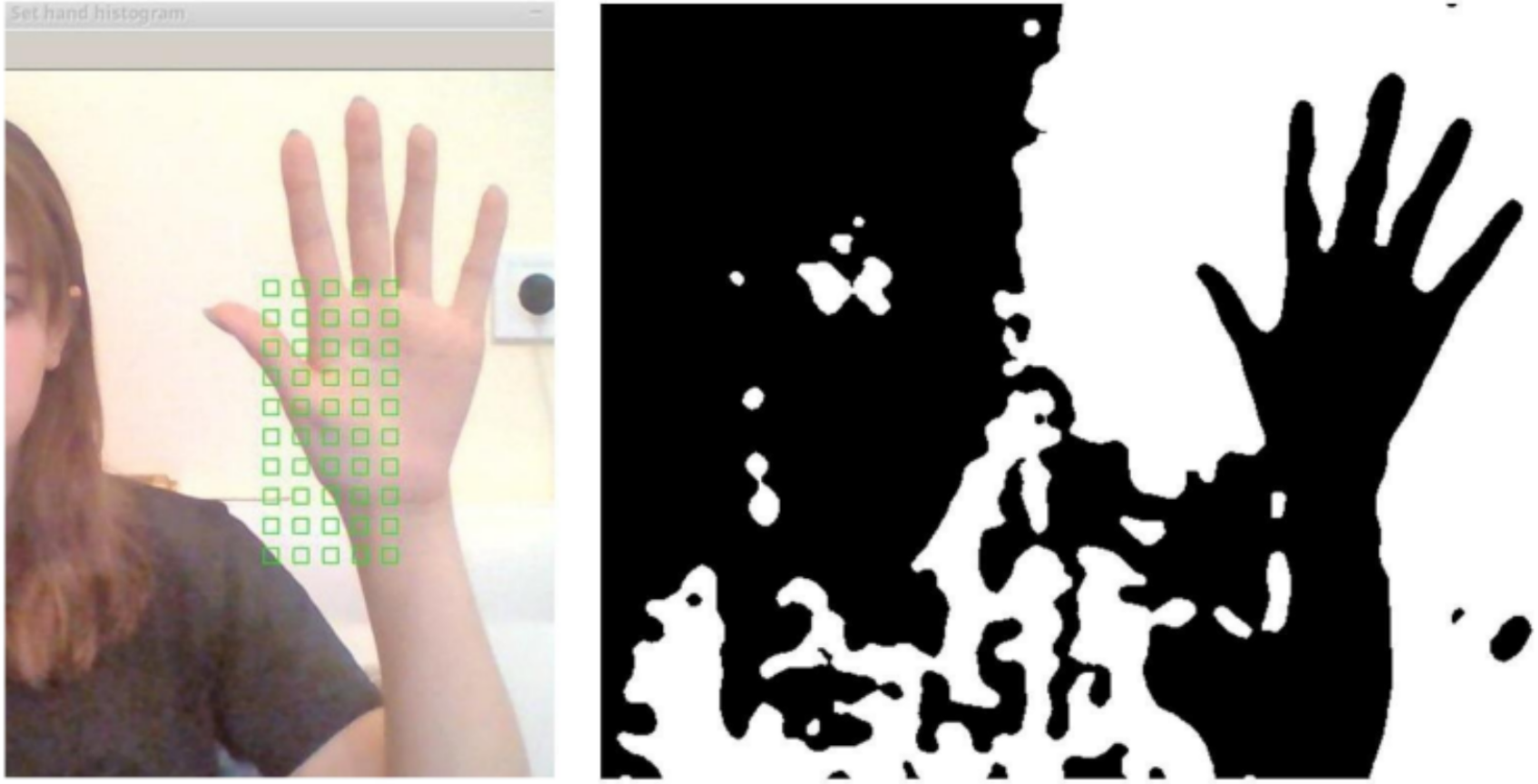
Операція	Фільтр	Оброблене зображення
Тотож-ність	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Знахо-дження кутів	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Різкість	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Квадратне розмиття	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Гауссове розмиття	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Приклад роботи найбільш поширених ядер згортки нейромережі CNN

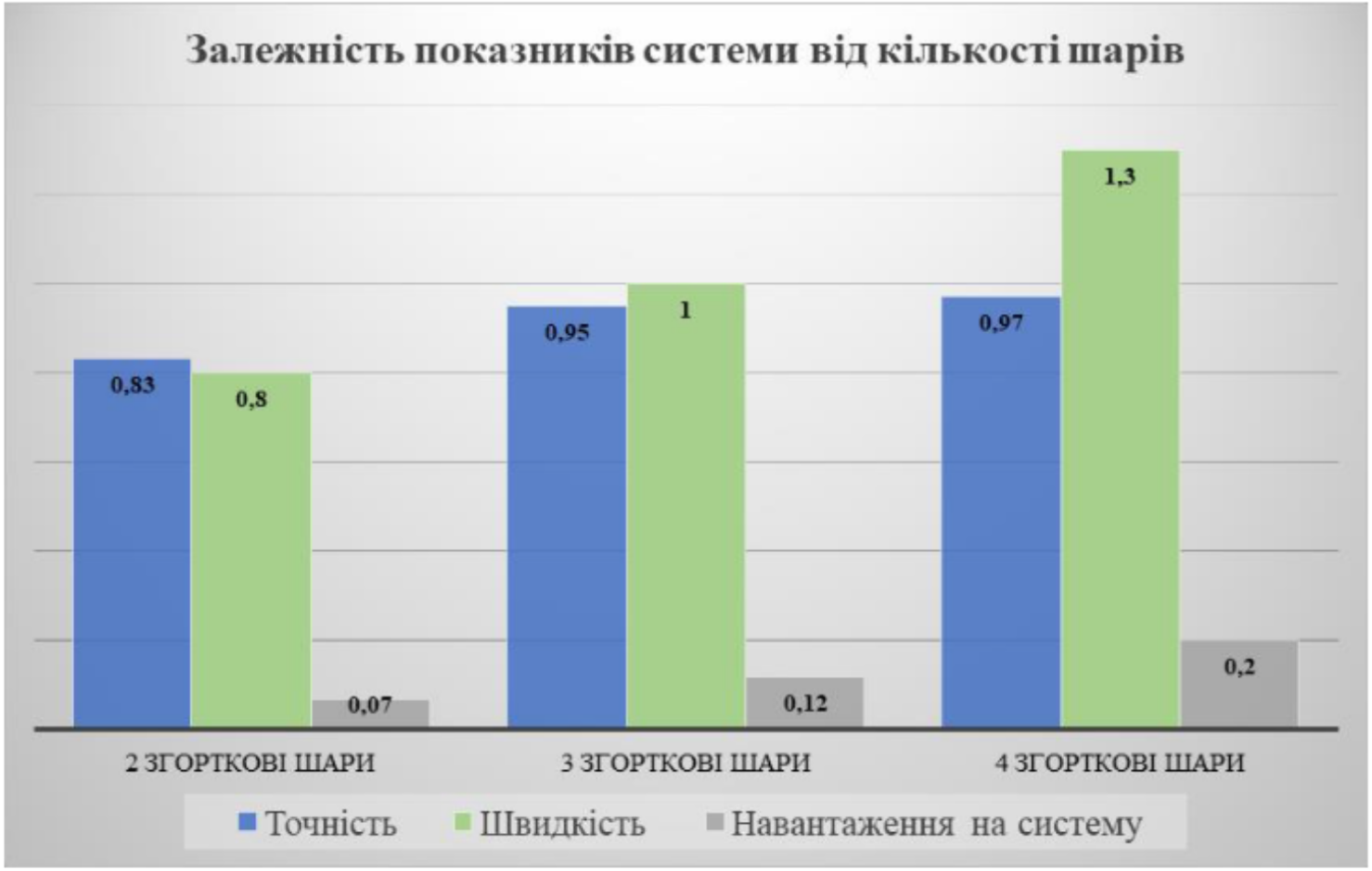


Архітектура нейромережевого класифікатора

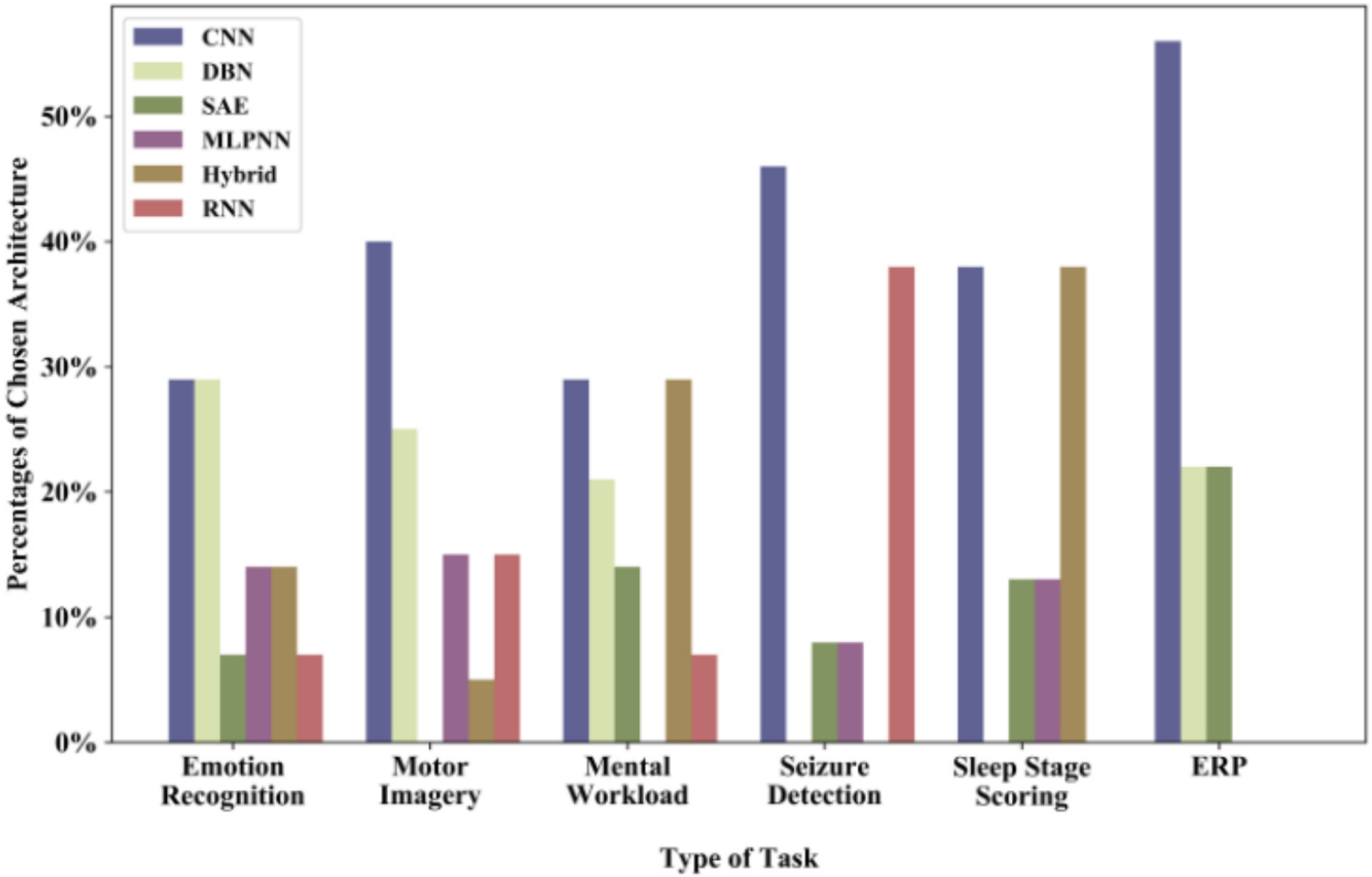
Підготовка даних для навчання нейромереж



Приклад роботи технології бінаризації



Якісні показники роботи системи в залежності від кількості конволюційних шарів в нейромережі CNN



Якісні показники роботи системи в залежності від типу нейромережі