Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Нейромережеві ігрові моделі для розвитку логічного, творчого мислення та креативності у дітей

Виконав:
Студент 2 курсу, групи КІТм-21-2
Колода Д. Д.
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології
Керівник доц. Борисенко В.П.

Допускається до захисту _____

Зав. кафедри О. Г. Руденко

2022 р.

Харківський національний університет радіоелектроніки

Факультет	Комп'ютерної інженерії та управління
Кафедра	Комп'ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп'ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп'ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:
Зав. кафедри

(підпис)
«17» грудня 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Колоді Дарині Дмитрівні

1. Тема роботи (проекту) Нейромережеві ігрові моделі для розвитку логічного, творчого мислення та креативності у дітей
затверджена наказом університету від "07" 11 2022р. № 1455СТ
2. Термін подання студентом роботи до екзаменаційної комісії 01 _12_2022р.
3. Вихідні дані до роботи (проекту)
малюнки;
зображення з відео-камери;
перелік використаних програмних засобів: ОС Windows 10;
інтегроване середовище: Visual Studio Code.

4. Перелік питань, що потрібно опрацювати в роботі
- Аналіз предметної області та постановка задачі;
- вибір та створення архітектури нейронної мережі;
- додаткові етапи до створення системи;
- висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням кафедри)
Слайд-презентація - 23 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявністю консультантів згідно до наказу, зазначеному у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

1 КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін	Примітка
1	Аналіз літератури	08.11.2022 - 15.11.2022	виконано
2	Аналіз предметної області та постановка задачі	16.11.2022 - 18.11.2022	Виконано
3	Вибір та створення необхідної архітектури	19.11.2022 - 22.11.2022	Виконано
4	Навчання графової нейромережі	23.11.2022 - 27.11.2022	Виконано
5	Тренування моделі розпізнавання долоні	28.11.2022 - 30.11.2022	Виконано
6	Програмна реалізація	01.12.2022 - 05.12.2022	Виконано
7	Тестування програми	06.12.2022 - 08.12.2022	Виконано
8	Написання пояснювальної записки	09.12.2022 - 15.12.2022	Виконано
9	Подання кваліфікаційної роботи керівникові та	16.12.2022 - 16.12.2022	Виконано
10	Рецензування кваліфікаційної роботи	17.12.2022-18.12.2022	Виконано

Дата видачі завдання 08.11.2022 р.

Студент Колода Д. Д.

Керівник роботи доц. Борисенко В. П.

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 88 сторінок, 60 рисунків, 2 додатків, 20 джерел.

Темою роботи є нейромережеві ігрові моделі для розвитку логічного, творчого мислення та креативності у дітей.

Предметом дослідження є алгоритми виявлення, відслідковування та розпізнавання об’єктів на основі глибинного навчання.

Об’єктом дослідження є малюнки користувачів, відеоматеріали з відеокамери комп'ютеру.

Метою роботи є розвиток логічного, творчого мислення та креативності у дітей через нейромережеві ігрові моделі.

Методами дослідження є нейромережеві ігрові моделі, програмні бібліотеки для роботи із зображеннями та відеоматеріалами.

Це новий та поки що не дуже популярний метод розвитку логічного, творчого мислення та креативності у дітей за допомогою нейромережевих ігрових моделей.

НЕЙРОМЕРЕЖЕВІ ІГРОВІ МОДЕЛІ, ПРОГРАМНІ БІБЛІОТЕКИ, ЗОБРАЖЕННЯ, РОЗПІЗНАВАННЯ, ВІДЕОМАТЕРІАЛИ, АЛГОРИТМИ ВИЯВЛЕННЯ, ГРАФОВІ НЕЙРОМЕРЕЖІ

ABSTRACT

Explanatory note of the qualification work: 88 pages, 60 figures, 2 appendices, 20 sources.

The topic of the work is neural network game models for the development of logical, creative thinking and creativity in children.

The subject of research is the algorithms for detection, tracking and recognition of objects based on deep learning.

The object of the research is user drawings, video footage from a computer video camera.

The purpose of the work is the development of logical, creative thinking and creativity in children through neural network game models.

Research methods include neural network game models, software libraries for working with images and video materials.

This is a new and still not very popular method of developing logical, creative thinking and creativity in children using neural network game models.

NEURAL NETWORK GAME MODELS, SOFTWARE LIBRARIES, IMAGES, RECOGNITION, VIDEO MATERIALS, DETECTION ALGORITHMS, GRAPHICAL NEURAL NETWORKS

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Нейромережеві ігрові моделі для розвитку логічного, творчого мислення та креативності у дітей

Виконав: студент 2 курсу, групи КІТм-21-2
Колода Д. Д.
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології
Керівник ___доц. Борисенко В.П._____

2022 р.

АНОТАЦІЯ

Колода Д. Д. Нейромережеві ігрові моделі для розвитку логічного, творчого мислення та креативності у дітей. – Магістерська кваліфікаційна робота.

У магістерській кваліфікаційній роботі вирішено актуальну проблему розвитку логічного, творчого мислення та креативності у дітей за допомогою нового більш програмного способу.

Метою кваліфікаційної роботи є розвиток логічного, творчого мислення та креативності у дітей через нейромережеві ігрові моделі.

Об'єктом дослідження цієї роботи є малюнки користувачів, відеоматеріали з відеокамери комп'ютеру.

Предметом дослідження є алгоритми виявлення, відслідковування та розпізнавання об’єктів на основі глибинного навчання.

Дитяче мислення, як і сприйняття, увага, пам’ять має свої особливості розвитку. У молодшому шкільному віці вона займає центральну ланку у системі психічних процесів. З розвитком мислення відбувається розвиток та вдосконалення не лише цих процесів, а й інтелектуальних здібностей.

Існує велика кількість способів розвитку логічного, творчого мислення та креативності у дітей. Наприклад: настільні ігри, конструктор, головоломки, ігри в усній формі, ребуси та анаграми та інші. Але всі вони не є ідеальними рішеннями.

Проблема полягає в тому, що деякі способи вимагають економічних витрат, інші - взаємодію з дітьми або батьками. Але часом не буває таких можливостей, наприклад під час епідемій. У наші часи дистанційного навчання, віддаленої роботи треба мати альтернативний варіант для розвитку логічного, творчого мислення та креативності у дітей. Тому на допомогу приходить новий додатковий підхід для розвитку дитячого мислення.

Цей спосіб має такі переваги як:

- не потребує економічних витрат;
- не потребує взаємодії з іншими дітьми або батьками (ШІ виповнює їх функцію);
- простота у використанні.

Але має і такий недолік як немає можливості розвитку живого спілкування, побудови діалогу між дитиною та іншою людиною.

Тож цей спосіб також не є ідеальним, але є гарним доповненням до всіх інших прийомів розвитку логічного, творчого мислення та креативності у дітей.

Були проаналізовані існуючі рішення. Нажаль, на сьогодні не має різноманітності у програмах з нейромережею. Протестована та виявлені недоліки та переваги програми QuickDraw від Google. Зроблені висновки та поставлена задача для реалізації. Для того, щоб дитина розвивала обидві півкули не виходячи з дому, зроблена програма з нейромережевими ігровими моделями. Головним функціоналом застосунку є розвиток логічного, творчого мислення та креативності у дітей.

Нейромережеві ігрові моделі складаються з трьох ігор:

- “Намалюйка” - користувач малює зображення, а нейромережа відгадує по начерках за допомогою графової нейромережи MGT (Multi-Graph Transformer) для розпізнавання начерків;
- “Камінь-ножиці-папір” - гра з противником у вигляді штучного інтелекту. ШІ сам обирає свій хід, аналізуючі ходи користувача. Це виходить за допомогою моделі HandPose, яка застосовується на TensorFlow і допомагає знаходити руку та пальці, та за допомгою FingerPose, яка працює на основі даних HandPose і дозволяє легко знаходити саме жести, наприклад «кулак» або «ножиці»;
- “Відгадайка” - ШІ загадує число та користувач намагається

відгадати за найменшу кількість спроб.

Проведений аналіз видів нейронних мереж, розглянуті їх архітектури, функції та застосування. Обрані графові нейромережі, бо вони використовуються для класифікації тексту, виявлення об'єктів на зображеннях, задачі комбінаторної оптимізації, якраз те, що потрібно у цьому проекті.

Обрана графова нейромережа MGT (Multi-Graph Transformer) для розпізнавання начерків для гри “Намалюйка” та моделі HandPose (допомагає знаходити руку та пальці) та FingerPose, (дозволяє легко знаходити жести).

Проведене тренування моделі розпізнавання долоні. Для цього були пройдені такі кроки:

1. Підготовка датасету та розділення на тренувальну, валідаційну та тестову частини.
2. Зміна розмірів зображеннь долонь та координати ключових точок.
3. Створення хітмапів (heatmaps, теплові карти) на основі координат ключових точок.
4. Підготовка даних (отримання батчів даних з попередніх зображень).
5. В межах кожної епохи робити стадії тренування та оцінки.
6. Отримання хітмапів.
7. Додаткова постобробка.
8. Розрахунок середньозважених x та y за всіма пікселями хітмапи.
9. Оцінка роботи моделі на тестовому сеті.
10. Розрахунок помилки для кожної ключової точки окремо.
11. Розрахунок середньої помилки для ключової точки по всьому тестовому сету.

Результати тренування були дуже гарні. Була натренерована модель і отримана така точність: середня помилка для ключової точки на тестовому сеті - 4,5% розміру зображення, або 6 пікселів для картинок 128x128, або 8 пікселів для картинок 224x224. Це дуже гарні результати, лише приблизно 5% ця модель

дає неточність.

Обрана бібліотека ReactJS на мові програмування JavaScript, тому що вона добре підходить для розробки алгоритмів машинного навчання, та має декілька необхідних додаткових бібліотек.

Для того, щоб користувач мав можливість запускати нейромережеві ігрові моделі на веб-сторінці, було прийняте рішення використовувати HTML та CSS для відображення.

Існує безліч наборів даних осіб, які можуть бути використані для оцінки алгоритмів. Є кілька популярних наборів даних для адаптації розпізнавання малюнків, наприклад quicklyDraw.

Для даного проекту, щоб впоратися з завданням, була обрана саме ця бібліотека та частково додані власні малюнки.

Також були обрані додаткові бібліотеки Tensorflow, HandPose, FingerPose і Brain.JS, які полегшують роботу з нейромережею.

На думку автора, даний проект - це сучасне, комп'ютеризоване доповнення до існуючих рішень розвитку логічного, творчого мислення та креативності у дітей.

НЕЙРОМЕРЕЖЕВІ ІГРОВІ МОДЕЛІ, ПРОГРАМНІ БІБЛІОТЕКИ, ЗОБРАЖЕННЯ, РОЗПІЗНАВАННЯ, ВІДЕОМАТЕРІАЛИ, АЛГОРИТМИ ВИЯВЛЕННЯ, ГРАФОВІ НЕЙРОМЕРЕЖІ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ - штучний інтелект

FF чи FFNN - нейронні мережі прямого поширення (feed forward neural networks)

P - перцептрони (perceptrons)

RBF - мережі радіально-базових функцій

HN - нейронна мережа Хопфілда (Hopfield network)

MC або DTMC - Ланцюги Маркова (Markov chains або discrete time Markov Chains)

BM - Машина Больцмана (Boltzmann machine)

RBM - Обмежена машина Больцмана (restricted Boltzmann machine)

AE - Автокодувальник (autoencoder)

SAE - Розріджений автокодувальник (sparse autoencoder)

VAE - Варіаційні автокодувальники (Variational autoencoder)

DAE - Шумопригнічуючі автокодувальники (denoising autoencoder)

DBN - Мережа типу "deep belief"

CNN - Згорткові нейронні мережі (convolutional neural networks)

DCNN - глибинні згорткові нейронні мережі (deep convolutional neural networks)

DN - Розгорткові нейронні мережі (deconvolutional networks)

GNN - Графові нейромережі (англ. Graph Neural Network)

GCN - Граф згорткової мережі (Graph Convolutional Network)

GAT - Графічна мережа уваги (Graph Attention Network)

GGNN - Нейронна мережа Gated Graph (Gated Graph Neural Network)

MGT - Графова нейромережа (Multi-Graph Transformer).

ВСТУП

Для людини дуже важливо навантажувати та розвивати обидві півкулі мозку. Мислення дитини надзвичайно пластичне, саме тому цей час чудово присвятити розвитку логічного та творчого мислення. У багатьох дослідженнях показано, що за розвиток логічного мислення відповідає ліва півкуля мозку, а за творче мислення – права.

Логіка в навчанні надзвичайно важлива. Логіка веде до строгості й обґрунтованості кожного кроку, сприяє розвитку критичності розуму, але сама по собі не приводить до творчого мислення. Без сумніву, для успішного вирішення будь-яких завдань потрібні знання, що пов’язані певними логічними зв’язками. Але для творчого мислення цього замало. Щоб сформувати творчо орієнтовану особистість з нестандартними підходами до розв’язання проблем і самостійністю суджень, потрібна інтеграція обох півкуль. Тому неправильно вважати, що один тип мислення абсолютно виключає інший. Звичайно, можна взятися виключно за тренування одного типу мислення і не чіпати інший. Але ж тоді, якщо провести паралелі з фізичними тренуваннями, це начебто тренування однієї правої ноги в футболіста, адже вона частіше використовується.

Світ є цілісним. Кожна особистість - цілісна. Саме тому потрібен цілеспрямований розвиток як логічної складової мислення, так і творчої. Найправильніше «добудувати» творчу складову на логічному мисленні, як на міцному фундаменті. З цим якнайкраще справляються ігри для розвитку обидвох півкуль. Дитина краще засвоює нові знання в ігровій формі. З цим добре допомагають нейромережеві ігрові моделі. Ідея нейромережі полягає в тому, щоби зібрати складну структуру з дуже простих елементів. Насправді нейромережа – це просто набір правил, за якими обробляється інформація. «Штучним нейроном», або «перцептроном», називається лише кілька

арифметичних дій. Штучні нейронні мережі зараз перебувають у піку популярності. Можна запитати, чи гучна назва зіграла свою роль у маркетингу та застосуванні цієї моделі. Поза всяким сумнівом, штучні нейромережі — популярний метод, і це очевидно завдяки їх успіху в багатьох сферах застосування: розпізнавання зображень, обробка природних мов, автоматизований трейдинг і автономні автомобілі.

Нейронні мережі увійшли до практики скрізь, де потрібно вирішувати завдання прогнозування, класифікації чи управління. Такий вражаючий успіх визначається кількома причинами:

- Багаті можливості. Нейронні мережі – виключно потужний метод моделювання, що дозволяє відтворювати надзвичайно складні залежності. Зокрема, нейронні мережі нелінійні за своєю природою. Протягом багатьох років лінійне моделювання було основним методом моделювання в більшості областей, оскільки для нього добре розроблені процедури оптимізації.
- Простота у використанні. Нейронні мережі навчаються на прикладах. Користувач нейронної мережі підбирає представницькі дані, а потім запускає алгоритм навчання, який автоматично сприймає структуру даних. При цьому від користувача, звичайно, потрібно якийсь набір евристичних знань про те, як слід відбирати та готувати дані, вибирати потрібну архітектуру мережі та інтерпретувати результати, проте рівень знань, необхідний для успішного застосування нейронних мереж, набагато скромніший, ніж, наприклад, під час використання традиційних методів статистики.

У майбутньому розвиток таких нейробіологічних моделей може призвести до створення дійсно мислячих комп'ютерів.

Метою магістерської кваліфікаційної роботи є розробка нейромережевих ігрових моделей для розвитку логічного творчого мислення та креативності у дітей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз існуючих підходів до розвитку логічного, творчого мислення та креативності у дітей

Дитяче мислення, як і сприйняття, увага, пам’ять має свої особливості розвитку. У молодшому шкільному віці вона займає центральну ланку у системі психічних процесів. З розвитком мислення відбувається розвиток та вдосконалення не лише цих процесів, а й інтелектуальних здібностей. Вивченням особливостей розвитку мислення у дітей займалися зарубіжні та вітчизняні вчені, дослідники, психологи та педагоги. Цінний внесок у вивчення проблеми розвитку мислення зробив Ж. Піаже. Теорія розвитку інтелекту у дитинстві, що він запропонував, послужила основою низки експериментальних і теоретичних робіт вітчизняних учених: Л.С.Виготского, А.Н.Леонтьева, В.В.Давидова, П.Я. Гальперіна [1].

Думкова діяльність - незамінна частина розвитку людини. Дуже часто труднощі, що виникають у школі, пов’язані з недостатністю розвитку основних розумових операцій. Насамперед, це аналіз та синтез, узагальнення, порівняння, конкретизація та абстрагування.

Реалії теперішнього часу такі, що існує величезна кількість шкільних програм, але, на жаль, не всі сприяють тому, щоб дитина навчилася думати і розмірковувати. Логіка допомагає швидше обробляти та аналізувати інформацію, правильно формулювати питання та робити висновки. Це основа успіху у навчанні та побудові професійної кар’єри.

Людина з розвиненим логічним та абстрактним мисленням ширше мислить. Він здатний нестандартно вирішувати складні завдання та пропонувати варіанти, яких не бачать інші. Логіка має на увазі гнучкість і

пластичність розуму. У століття стрімких змін розумові навички допомагають завжди залишатися в тренді і швидко освоювати нові компетенції.

Навички логічного мислення потрібні кожній людині незалежно від його занять. На нас з усіх боків навалюються величезні потоки суперечливої інформації, розібратися в істинності та хибності суджень та думок може тільки той, хто вміє оперувати логічними конструкціями.

Численні теорії та концепції розвитку мислення дозволяють зробити висновок про те, що крім природного розвитку та проходження певних стадій розвитку індивідуального мислення, його розвиток можна стимулювати шляхом індивідуальної чи групової роботи.

Також є багато інших способів для розвитку мислення у дітей.

1.1.1 Настільні ігри

Нині сфера настільних ігор добре розвинена. Є багато різних варіантів з простими правилами, яскравими картинками та гарним сюжетом, близьким до дитини. До того ж, існують ігри, розроблені спеціально для маленьких дітей, щоб розвинути їхню здатність до логіки. Плюс настільних ігор у тому, що їх можна використовувати і не за призначенням: за допомогою фігурок можна вчитися розрізняти кольори та форми, а з кубиком – вивчати рахунок.

1.1.2 Конструктор

Конструювання - вид діяльності, який може надовго зайняти як дитину, так і її батьків. Потрібно лише виявити трохи уяви, щоб купка кубиків перетворилася на щось цікаве. Зараз є безліч різних видів конструкторів: від звичайного набору кубиків та пірамідок до «Лего», за допомогою яких дитина вчиться працювати зі схемами, а також розвиває творче мислення.

1.1.3 Головоломки

Загадки – один із найпопулярніших та найвідоміших способів розвитку нестандартного мислення та логіки дитини. Збірники з головоломками продаються в будь-якому книжковому, а під окремі їхні види (лабіринти та «знайди відмінність») відводяться цілі рубрики у дитячих журналах. Також дуже добре розвивають мислення в дітей віком просторові головоломки, такі як кубик Рубика.

Можна використовувати ігри з сірниками (у школі замість них використовують лічильні палички), де потрібно змінити положення кількох сірників, щоб отримати потрібну фігуру. Також зараз дуже популярними є металеві головоломки. Вони зазвичай виглядають як пара зігнутих спеціальним чином предметів, які потрібно роз’їздити. У цьому добре розвивається як мислення, а й дрібна моторика.

1.1.4 Ігри в усній формі

Цей варіант хороший тим, що для нього не потрібно нічого купувати, а працює він не менш добре, ніж книжка із загадками. Існує безліч ігор, спрямованих на розвиток дитячого мислення та різних здібностей. Можна грати в асоціації, в їстівне-неїстівне, додумувати фразу до кінця, підбирати риму і таке інше.

Улюблена багатьма з дитинства гра «Холодно – гаряче» теж чудово підходить для розвитку мислення. Суть зводиться до того, що дитина шукає заховані предмети, задаючи навідні питання, на які батьки повинні відповідати «холодно», «тепло» або «гаряче» залежно від того, наскільки близька дитина до знахідки.

1.1.5 Ребуси та анаграми

Такі види завдань підходять для старших діток, які вже вміють читати. Подібні ігри часто пропонують у школі та у збірниках завдань для початкової школи. Ребуси розвивають мислення та концентрацію, до того ж допомагають запам’ятати написання різних слів.

1.1.6 Порівняння існуючих рішень

Існує велика кількість способів розвитку логічного, творчого мислення та креативності у дітей, але всі вони не є ідеальними. Деякі вимагають економічних витрат, інші - взаємодію з дітьми або батьками. Але часом не буває таких можливостей, тому на допомогу приходить новий додатковий підхід для розвитку логічного, творчого мислення та креативності у дітей за допомогою нейромережевих ігрових моделей.

Цей спосіб має такі переваги як:

- не потребує економічних витрат;
- не потребує взаємодії з іншими дітьми або батьками (ШІ виповнює їх функцію);
- простота у використанні.

Але має і такий недолік як немає можливості розвитку живого спілкування, побудови діалогу між дитиною та іншою людиною.

Тож цей спосіб також не є ідеальним, але є гарним доповненням до всіх інших прийомів розвитку логічного, творчого мислення та креативності у дітей.

1.1.7 Аналіз існуючого рішення з нейромережею

Для розробки нейромережевих ігрових моделей для розвитку логічного,

творчого мислення та креативності у дітей були розглянуті декілька існуючих рішень, наприклад QuickDraw від Google [2].

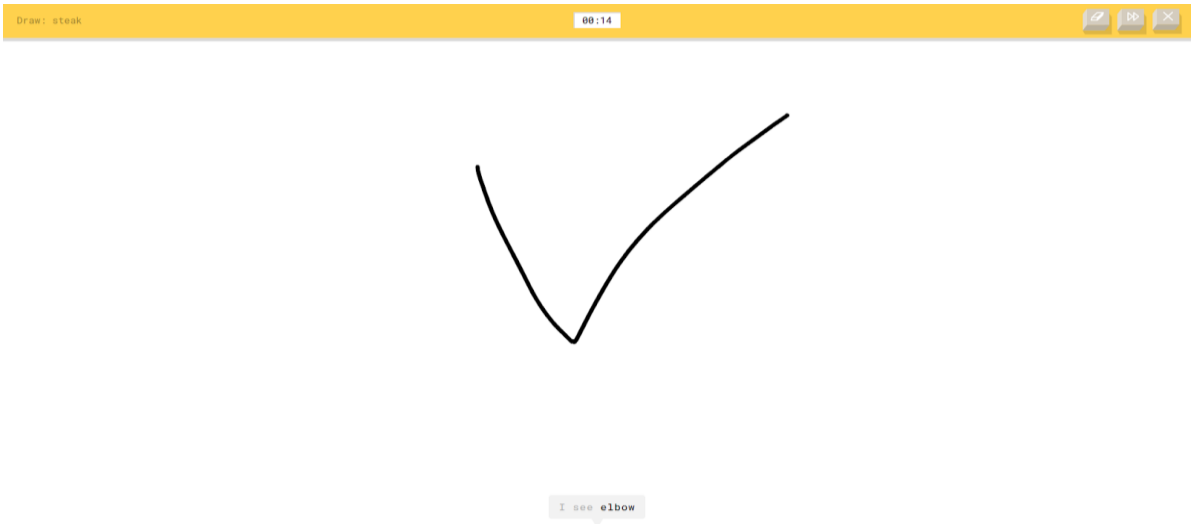


Рисунок 1.1 – Інтерфейс програми QuickDraw

Були виявлені такі переваги:

- програма проста та зрозуміла у використанні;
- приємний інтерфейс;
- нейромережа чітко вгадує малюнок користувача.

Але був знайдений і такий недолік, як маленький вибір ігор. Хотілось розширень у вигляді інших розваг.

Після тестування даних програм були виявлені переваги та недоліки, проведений порівняльний аналіз, зроблені висновки та поставлена задача для реалізації.

1.2 Опис завдання

Дитяча гра — це діяльність, спрямована на орієнтування в предметній і

соціальній дійсності, в якій дитина відображає враження від їх пізнання. Мати дитинство — це передусім мати право на розвиток власної ігрової діяльності, яка є важливою складовою дитячої субкультури. Водночас гра є могутнім виховним засобом, у ній, за словами К. Ушинського, реалізується потреба людської природи. У системі навчання і виховання дошкільників активно використовуються дидактичні (навчальні) ігри, які розвивають спостережливість, уяву, пам'ять, мислення, мовлення, сенсорні орієнтації дітей у розмірах, формах, кольорах, максимально задіюють інтелектуальний потенціал у пізнанні світу і себе.

Для того, щоб дитина розвивала обидві півкули не виходячи з дому, можна зробити програму з нейромережевими ігровими моделями. Головним функціоналом застосунку є розвиток логічного, творчого мислення та креативності у дітей. Нейромережеві ігрові моделі складаються з трьох ігор: «Намалюйка», «Камінь-ножиці-папір», «Відгадайка».

1.2.1 Гра “Намалюйка”

Гра складається з десятьох рівнів складності малюнків. Вона починається з того, що загадується слово, наприклад кіт, яке дитині треба намалювати за певний час. Нейромережа повинна відгадати цю назву за допомогою пошуку схожих зображень вже в існуючій бібліотеці. Рівень закінчується, коли нейромережа відгадує загадане слово і бал зараховується дитині за те, що вона намалювала вгадуваний малюнок, або якщо вийшов час та нейромережа не впізнала. Користувач переходить на наступний рівень і все повторюється, поки не закінчаться всі десять рівнів. Дитина виграє, якщо нейромережа вгадала більшість малюнків.

Ця гра вимагає від дитини креативності, творчого підходу, розуміння як виглядає певний об'єкт, як можна намалювати це слово, які особливості для

швидкого впізнавання нейромережею воно має.

1.2.2 Гра “Камінь-ножиці-папір”

Для цієї гри є вимога — включення веб-камери на комп'ютері або ноутбуці. Правила цієї дитячої гри всі знають, але бувають такі моменти, коли немає можливості пограти в неї, бо не знайшовся противник. У цьому випадку допоможе нейромережа. Вона визначає положення долоні користувача та розпізнає жест та намагається обіграти. В ігрі нескінченні рівні. Задача дитини – включити логічне мислення, передбачити наступний хід противника та обіграти. Гра допомагає дитині розвинути здатність аналізувати, оцінювати та передбачати інформацію.

1.2.3 Гра “Відгадайка”

Нейромережа загадує число, яке користувач повинен відгадати за найменшу кількість спроб за допомогою підказок «більше» та «менше». Ця гра веде до строгості й обґрунтованості кожного кроку, допомагає дитині почати логічно мислити, обмірковувати який варіант буде найближчий до правильної відповіді.

1.3 Постановка задачі

Вибрати нейронну мережу, навчити її і реалізувати програму, яка складатиметься з трьох ігор для розвитку логічного, творчого мислення та креативності у дітей за допомогою нейромережевих ігрових моделей:

- «Намалюйка» - користувач малює зображення, а нейромережа відгадує;
- «Камінь-ножиці-папір» - гра з противником у вигляді штучного інтелекту

та нейромережею, яка розпізнає долоню та жест користувача;

- «Відгадайка» – ШІ загадує число та користувач намагається відгадати за найменшу кількість спроб.

Інтерфейс повинен бути чіткий, зрозумілий та цікавий для дитини.

Для виконання поставлених задач буде використовуватися MGT ігрова модель та TensorFlow.

MGT це графова нейромережа для розпізнавання начерків, яка допоможе впізнавати малюнки користувача.

TensorFlow – це ML-framework від Google, який призначений для проектування, створення та вивчення моделей глибокого навчання.

Глибоке навчання - це область машинного навчання, алгоритми в якій були натхненні структурою та роботою мозку.

Також будуть використовуватися додаткові бібліотеки та фреймворки для полегшення роботи з нейромережею.

2 ВИБІР ТА СТВОРЕННЯ АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ

2.1 Аналіз нейронних мереж

Нейронна мережа — це метод штучного інтелекту, який вчить комп'ютери обробляти дані таким же способом, як і людський мозок. Це тип процесу машинного навчання, що називається глибоким навчанням, який використовує взаємопов'язані вузли або нейрони в шаруватій структурі, що нагадує людський мозок. Він створює адаптивну систему, за допомогою якої комп'ютери навчаються на своїх помилках та постійно вдосконалюються. Таким чином, штучні нейронні мережі намагаються вирішувати складні завдання, такі як резюмування документів або розпізнавання осіб з більш високою точністю.

Нейронні мережі допомагають комп'ютерам приймати розумні рішення з обмеженою участю людини. Вони можуть вивчати та моделювати відносини між нелінійними та складними вхідними та вихідними даними. Наприклад, нейронні мережі можуть виконувати такі завдання.

Нейронні мережі можуть розуміти неструктуровані дані та робити загальні спостереження без спеціального навчання. Наприклад, вони можуть розпізнати, що дві різні вхідні пропозиції мають однакове значення:

- Чи не підкажете як зробити оплату?
- Як мені переказати гроші?

Нейронна мережа зрозуміє, що обидві пропозиції означають те саме. Також вона може визначити, що Бакстер-роуд – це місце, а Бакстер Сміт – це ім'я людини.

Існує безліч видів архітектур нейромереж, які можна використовувати для розв’язання поставленої задачі. Розглянемо їх [4].

2.1.1 Нейронні мережі прямого поширення та перцептрони

Ці нейронні мережі дуже прямолінійні, вони передають інформацію від входу до виходу. Нейронні мережі часто описуються у вигляді торта, де кожен шар складається з вхідних, прихованих або вихідних клітин. Клітини одного шару пов'язані між собою, а сусідні шари зазвичай повністю пов'язані. Найпростіша нейронна мережа має дві вхідні клітини і одну вихідну, і може використовуватися як модель логічних вентилів. FFNN зазвичай навчається за методом зворотного поширення помилки, в якому мережа отримує безліч вхідних та вихідних даних.

Цей процес називається навчанням з учителем, і він відрізняється від навчання без вчителя тим, що у другому випадку безліч вихідних даних мережу складає самостійно. Вищезгадана помилка є різницею між введенням та висновком. Якщо мережа має достатню кількість прихованих нейронів, вона теоретично здатна змоделювати взаємодію між вхідним і вихідними даними. Практично такі мережі використовуються рідко, але часто комбінують з іншими типами для отримання нових.

Рисунок 2.1 – Архітектура FFNN

2.1.2 Мережі радіально-базових функцій (RBF)

Це FFNN, яка використовує радіальні базові функції як функції активації.

Більше вона нічим не виділяється.

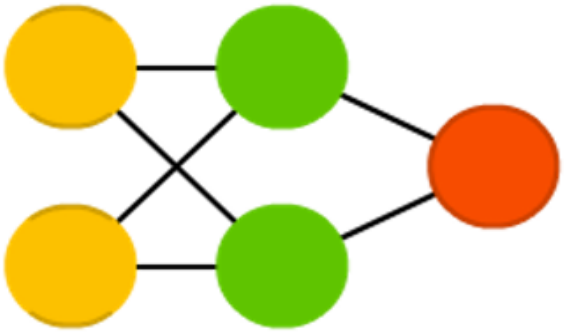


Рисунок 2.2 – Архітектура RBF

2.1.3 Нейронна мережа Хопфілда

Це повнозв'язкова нейронна мережа із симетричною матрицею зв'язків. Під час отримання вхідних даних кожен вузол є входом, у процесі навчання стає прихованим, а потім стає виходом. Мережа навчається так: значення нейронів встановлюються відповідно до бажаного шаблону, після чого обчислюються ваги, які надалі не змінюються. Після того, як мережа навчилася на одному або декількох шаблонах, вона завжди зводитиметься до одного з них (але не завжди до бажаного). Вона стабілізується залежно від загальної «енергії» та «температури» мережі. У кожного нейрона є свій поріг активації, що залежить від температури, при проходженні якого нейрон набуває одного з двох значень (зазвичай -1 або 1, іноді 0 або 1). Така мережа часто називається мережею з асоціативною пам'яттю; як людина, бачачи половину таблиці, може уявити другу половину таблиці, і ця мережа, отримуючи таблицю, наполовину зашумленную, відновлює до повної.

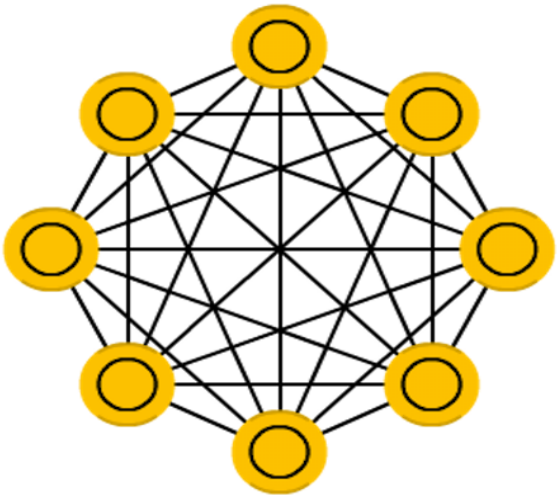


Рисунок 2.3 – Архітектура HN

2.1.4 Ланцюги Маркова (DTMC)

Це попередники машин Больцмана (BM) та мереж Хопфілда (HN). Їхній сенс можна пояснити так: які мої шанси потрапити в один із наступних вузлів, якщо я перебуваю в цьому? Кожен наступний стан залежить лише від попереднього. Хоча насправді ланцюги Маркова є CP, вони дуже схожі. Також ланцюги Маркова не обов'язково повні.

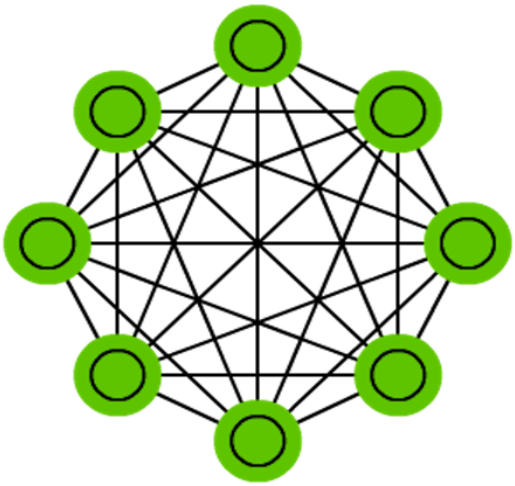


Рисунок 2.4 – Архітектура DTMC

2.1.5 Машина Больцмана

Дуже схожа на мережу Хопфілда, але в ній деякі нейрони позначені як вхідні, а деякі як приховані. Вхідні нейрони надалі стають вихідними. Машина Больцмана – це стохастична мережа. Навчання відбувається за методом зворотного поширення помилки або алгоритмом порівняльної розбіжності. Загалом процес навчання дуже схожий на такий у мережі Хопфілда.

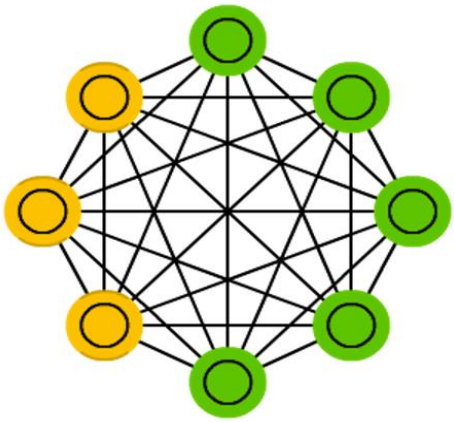


Рисунок 2.5 – Архітектура BM

2.1.6 Обмежена машина Больцмана

Схожа на машину Больцмана і, отже, на мережу Хопфілда. Єдиною різницею є її обмеженість. У ній нейрони одного типу пов'язані між собою. Обмежену машину Больцмана можна навчати як FFNN, але з одним нюансом: замість прямої передачі даних та зворотного розповсюдження помилки потрібно передавати дані спершу у прямому напрямку, потім у зворотному. Після цього проходить навчання за методом прямого та зворотного поширення помилки.

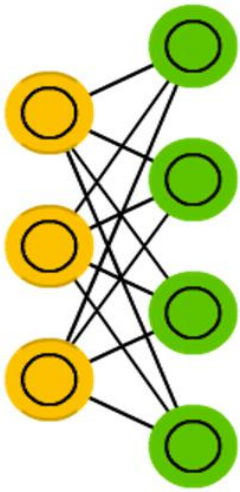


Рисунок 2.6 – Архітектура RBM

2.1.7 Автокодувальник

Чимось схожий на FFNN, тому що це скоріше інший спосіб використання FFNN, аніж фундаментально інша архітектура. Основною ідеєю є автоматичне кодування у сенсі стиснення (не шифрування) інформації. Сама мережа формою нагадує пісочний годинник, у ній приховані шари менше вхідного і вихідного, причому вона симетрична. Мережу можна навчити методом зворотного поширення помилки, подаючи вхідні дані та задаючи помилку рівної різниці між входом та виходом.

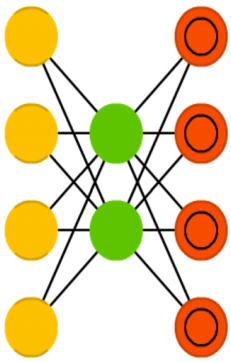


Рисунок 2.7 – Архітектура AE

2.1.8 Розріджений автокодувальник

У певному сенсі протилежність звичайного. Замість того, щоб навчати мережу відображати інформацію в меншому обсязі вузлів, ми збільшуємо їх кількість. Замість того, щоб звужуватись до центру, мережа там роздмухується. Мережі такого типу корисні до роботи з великою кількістю дрібних властивостей набору даних. Якщо навчати мережу як звичайний автокодувальник, нічого корисного не вийде. Тому, крім вхідних даних, подається ще й спеціальний фільтр розрідженості, який пропускає лише певні помилки.

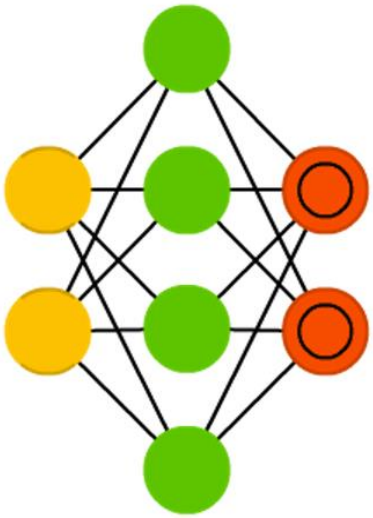


Рисунок 2.8 – Архітектура SAE

2.1.9 Варіаційні автокодувальники

Мають схожу з АЕ архітектурою, але навчають їх іншому: наближенню ймовірнісного розподілу вхідних зразків. У цьому беруть початок від машин

Больцмана. Тим не менш, вони спираються на байєсовську математику, коли йдеться про ймовірні висновки та незалежності, які інтуїтивно зрозумілі, але складні в реалізації. Якщо узагальнити, то можна сказати, що ця мережа бере до уваги вплив нейронів. Якщо щось одне відбувається в одному місці, а щось інше — в іншому, ці події не обов'язково пов'язані, і це має враховуватися.

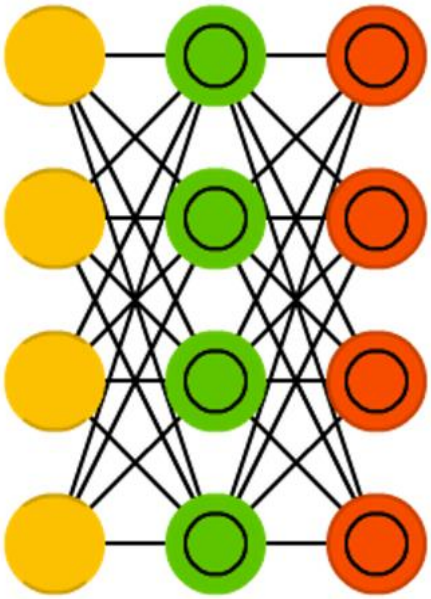


Рисунок 2.9 – Архітектура VAE

2.1.10 Шумопригнічуючі автокодувальники

Це AE, які вхідні дані подаються в зашумленном стані. Помилки ми обчислюємо так само, і вихідні дані порівнюються із зашумленими. Завдяки цьому мережа вчиться звертати увагу більш широкі властивості, оскільки маленькі можуть змінюватися разом із шумом.

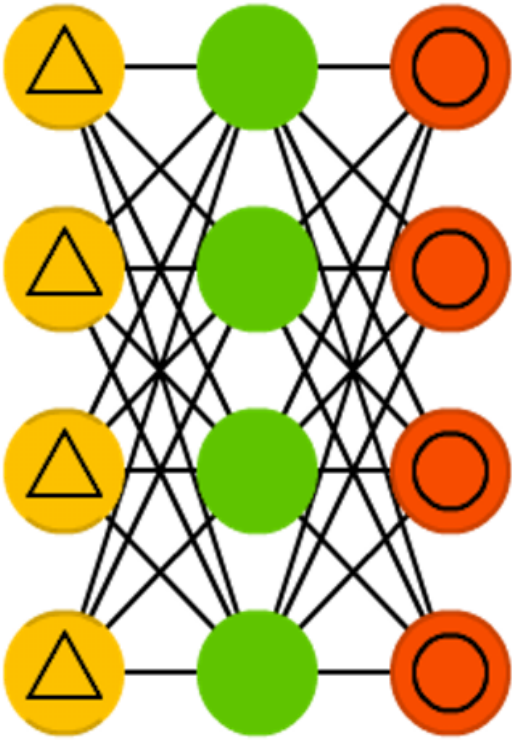


Рисунок 2.10 – Архітектура DAE

2.1.11 Мережа типу "deep belief"

Це назва, яку отримав тип архітектури, в якій мережа складається з кількох з'єднаних RBM або VAE. Такі мережі навчаються побічно, причому кожному блоку потрібно лише вміти закодувати попередній. Така техніка називається «жадібним навчанням», яка полягає у виборі локальних оптимальних рішень, що не гарантують оптимальний кінцевий результат. Також мережу можна навчити (методом зворотного поширення помилки) відображати дані як імовірнісної моделі. Якщо використовувати навчання без вчителя, стабілізовану модель можна використовувати для створення нових даних.

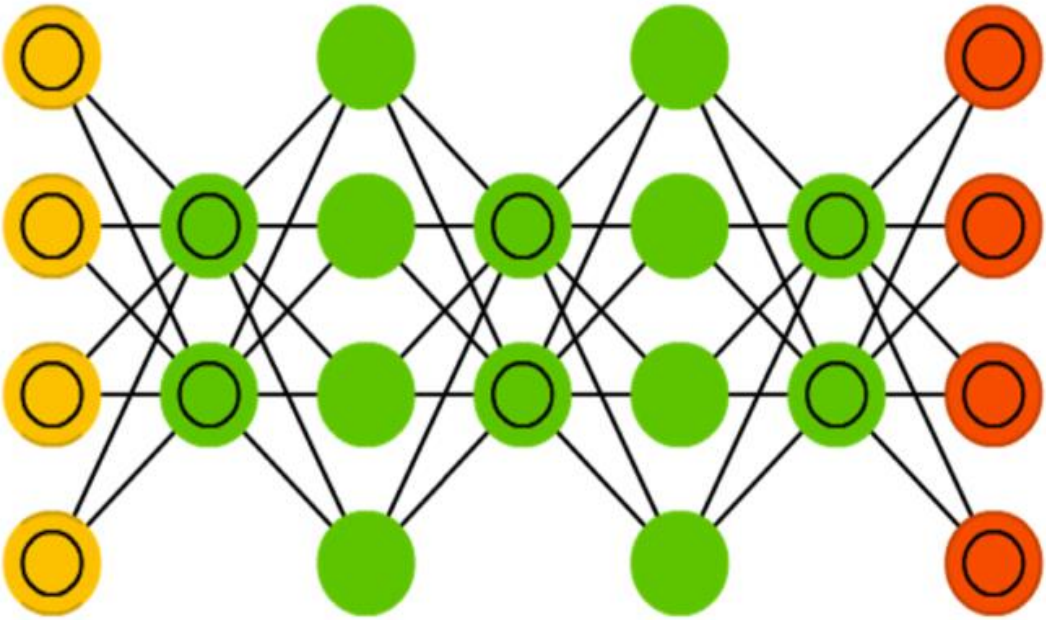


Рисунок 2.11 – Архітектура DBN

2.1.12 Згорткові нейронні мережі та глибинні згорткові нейронні мережі

Зазвичай вони використовуються для обробки зображень, рідше для аудіо. Типовим способом застосування CNN є класифікація зображень: якщо на зображенні є кішка, мережа видасть кішка, якщо є собака собака. Такі мережі зазвичай використовують «сканер», який не париться всі дані за один раз. Наприклад, якщо у вас є зображення 200×200, ви не відразу оброблятимете всі 40 тисяч пікселів. Замість це мережа рахує квадрат розміру 20 x 20 (зазвичай з лівого верхнього кута), потім зрушить на 1 піксель і рахує новий квадрат, і т.д. Ці вхідні дані передаються через згорткові шари, в яких не всі вузли з'єднані між собою. Ці шари мають властивість стискатися з глибиною, причому часто використовуються ступені двійки: 32, 16, 8, 4, 2, 1. На практиці до кінця CNN прикріплюють FFNN для подальшої обробки даних. Такі мережі називаються глибинними (DCNN).

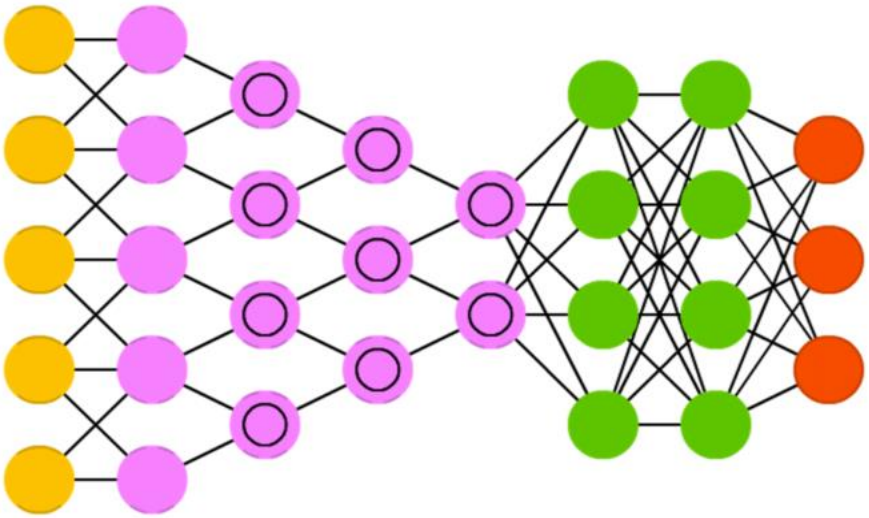


Рисунок 2.12 – Архітектура DCNN

2.1.13 Розгорткові нейронні мережі

Також звані зворотними графічними мережами, є зворотним до згорткових нейронних мереж. Уявіть, що ви передаєте мережі слово "кішка", а вона генерує картинки з кішками, схожі на реальні зображення котів. DNN також можна поєднувати з FFNN. Варто зауважити, що у більшості випадків мережі передається не рядок, а який бінарний вектор: наприклад, <0, 1> - це кішка, <1, 0> - собака, а <1, 1> - і кішка, і собака.

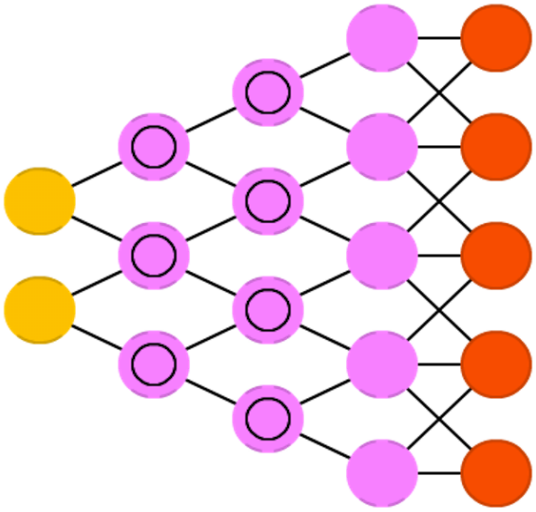


Рисунок 2.13 – Архітектура DN

2.1.14 Графові нейромережі

Графова нейронна мережа — тип нейронної мережі, яка працює зі структурою графа. Типовим застосуванням GNN є класифікація вузлів.

Графи, не володіючи регулярною структурою як зображення (кожний піксель має 8 сусідів) або тексти (послідовність слів), тривалий час залишалися поза увагою класичних нейронних моделей, які набули широкого поширення в галузі машинного навчання та штучного інтелекту [5]. Більшість моделей векторизації графів (побудови векторного представлення вершин у графі) були досить повільними та використовували алгоритми на основі матричної факторизації чи спектральної декомпозиції графа. У 2015-16 роках з'явилися ефективніші моделі (DeepWalk, Line, Node2vec, Hope) на основі випадкових блукань. Однак і вони мали обмеження, тому що ніяк не торкалися при побудові векторної моделі графа додаткових ознак, які можуть зберігатися на вершинах або на ребрах. Поява графових нейронних мереж стала логічним продовженням досліджень у галузі графових ембеддингів та дозволила уніфікувати під єдиним

фреймворком попередні підходи.

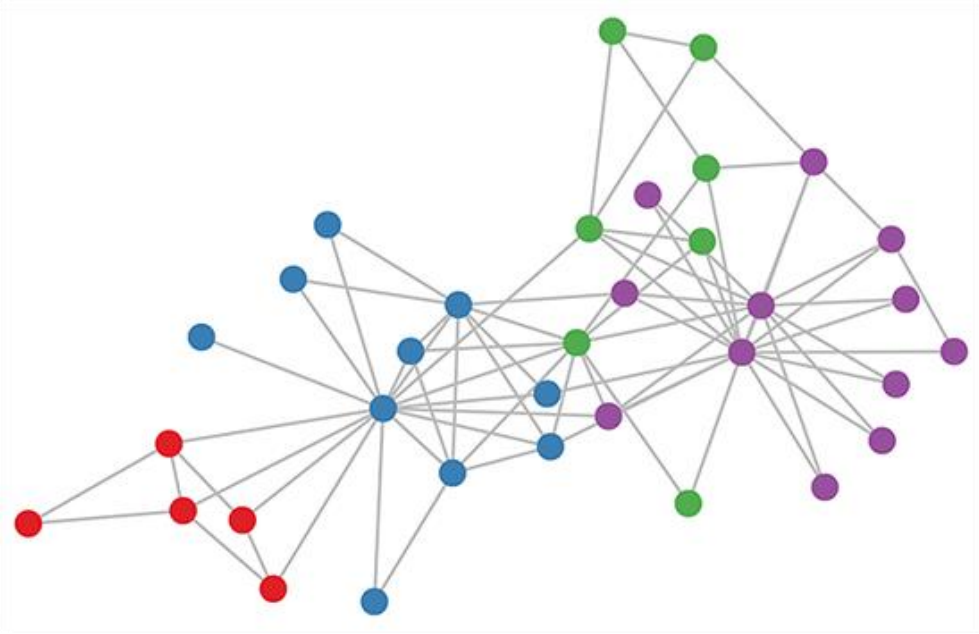


Рисунок 2.14 – Архітектура GNN

Використання GNN дозволяє працювати з даними графів без попередньої обробки. Такий підхід дає змогу зберегти топологічні відносини між вузлами графа.

В основі GNN закладено механізм розповсюдження інформації. Граф обробляється набором модулів, які пов'язані між собою відповідно до зв'язків графа. Також кожен із модулів пов'язані з вузлами графа. У процесі навчання модулі оновлюють свої статки та обмінюються інформацією. Це триває доти, доки модулі не досягнуть стійкої рівноваги (для того, щоб була гарантія того, що такий стійкий стан існує, цей механізм поширення обмежений). Вихідні дані GNN обчислюються з урахуванням стану модуля кожному вузлі (2.1).

□□□□□ □□□□□h□□□h□□□ □□□□□□□□□□□□□

$$h_{\square}^{\square} = \square\left(\square_{\square} \sum_{\square \square \square(\square)} \frac{h_{\square}^{\square-1}}{|\square(\square)|} + \square_{\square} h_{\square}^{\square-1}\right)$$

$$\frac{\square\square\square\ \square\square\square\square h\square\square\square h\square\square\square\ \square\square\square\square\square\square\square\square\square\square\square\square}{h_{\square}^{\square} = \square\left(\square_{\square}\sum_{\square\ \square\ \square\ (\square)\cup\square}\frac{h_{\square}^{\square-1}}{\sqrt{|\square(\square)||\square(\square)|}}\right)} \quad (2.1)$$

2.2 Вибір архітектури нейронної мережі для проекту

Проведений аналіз архітектур нейронних мереж показав, що для даного проекту підійде архітектура, яка влаштовує по характеристикам та застосуванню, це графові нейромережі. Один шар графової нейромережі - це звичайний повнозв'язний шар (fully-connected layer) нейронної мережі, але ваги в ньому застосовуються не до всіх вхідних даних, а тільки до тих, які є сусідами конкретної вершини в графі, на додаток до її власного подання з попереднього шару. Ваги для сусідів і самої вершини можуть задаватися загальною матрицею ваги або двома окремими. Можуть додаватись нормалізації для прискорення збіжності; можуть змінюватись нелінійні функції активацій, але загальна конструкція залишається схожою. При цьому графові згорткові мережі отримали свою назву завдяки агрегації інформації від своїх сусідів, хоча набагато ближче до цього визначення стоять графові механізми уваги (GAT) або індуктивна модель навчання (GraphSAGE).

2.2.1 Розвиток GCN

В останні роки системи, засновані на варіантах графових нейронних мереж, таких як GCN (Graph Convolutional Network), GAT (Graph Attention Network), GGNN (Gated Graph Neural Network), продемонстрували високу продуктивність при вирішенні багатьох завдань [6].

Для тестування алгоритмів часто використовують датасети наборів даних

мережі цитування, такі як Citeseer, Cora and Pubmed. Вони вузли — це документи, а ребра — це посилання цитати. GCN на цих даних вдається досягти точності в районі 70.3%, 81.5%, 79.0% відповідно, а результати GAC становлять 83.0%, 72.5%, 79.0%.

При вирішенні реальних завдань, дані алгоритми застосовуються в наступних областях:

- GCN застосовують у завданнях класифікації текстових даних, зображень, хвороб та прогнозуванні побічних ефектів;
- GAT використовують для класифікації тексту, виявлення об'єктів на зображеннях, задачі комбінаторної оптимізації;
- GGNN застосовують для нейронного машинного перекладу, аналізу соціальних відносин.

2.2.2 Застосування графових нейромереж

Графові нейромережі мають велике застосування: розпізнавання зображень, обробка природних мов, автоматизований трейдинг і автономні автомобілі. Нейронні мережі увійшли до практики скрізь, де потрібно вирішувати завдання прогнозування, класифікації чи управління. Розглянемо декілька зразків використання цих нейромереж.

2.2.3 Рекомендаційні системи

Графи розвиваються у контексті взаємодії користувачів із продуктами на платформах електронної торгівлі. В результаті багато компаній використовують графові нейромережі для створення рекомендаційних систем. Зазвичай за допомогою графів моделюють взаємодію користувачів з товарами, навчають ембеддінгів з урахуванням правильно підібраної негативної вибірки, і за

допомогою ранжування результатів вибирають персоналізовані пропозиції щодо товарів і в реальному часі показують конкретним користувачам. Одним із перших сервісів з таким механізмом став Uber Eats: нейромережа GraphSage підбирає рекомендації продуктів харчування та ресторанів.

Хоча у випадку з рекомендаціями продуктів харчування графи виходять відносно невеликими через географічні обмеження, проте в деяких компаніях застосовуються нейромережі з мільярдами зв'язків. Наприклад, китайський гігант Alibaba запустив в експлуатацію графові ембеддінги та графові нейромережі стосовно мільярдів користувачів та товарів. Лише створення таких графів — кошмар для розробників. Але завдяки конвеєру Aligraph можна лише за п'ять хвилин збудувати граф на 400 млн вузлів. Вражає. Aligraph підтримує ефективне, розподілене графове сховище, оптимізовані оператори вибірок та купу власних графових нейромереж. Зараз цей конвеєр використовується для рекомендацій та персоналізованого пошуку за численними продуктами компанії.

Pinterest запропонувала модель PinSage, яка ефективно підбирає сусідні вузли за допомогою персоналізованого PageRank та оновлює ембеддінги вершин за допомогою агрегування інформації від сусідів. Наступна модель PinnerSage вже може працювати з мультиембеддінг, щоб враховувати різні смаки користувачів. Це лише кілька примітних прикладів у сфері рекомендаційних систем. Можете ще почитати про дослідження Amazon графів знань та графових нейромереж, або про використання компанією Fabula AI графових нейромереж для визначення фальшивих новин. Але і без цього очевидно, що графові нейромережі демонструють багатообіцяючі результати при значному сигналі від взаємодій користувача [7].

2.2.4 Комбінаторна оптимізація

Вирішення завдань комбінаторної оптимізації лежать в основі багатьох важливих продуктів у сфері фінансів, логістики, енергетики, природничих наук та проектування електроніки. Більшість цих завдань описується за допомогою графів. І за останнє століття було витрачено дуже багато зусиль створення більш ефективних алгоритмічних рішень. Проте революція машинного навчання дала нам нові, переконливі підходи.

Команда Google Brain використовувала графові нейромережі для оптимізації енергоспоживання, площі та продуктивності чіпів для нового обладнання на зразок Google TPU. Комп'ютерний процесор можна представити у вигляді графа пам'яті та компонентів логіки, кожен зі своїми координатами та типом. Визначення розташування для кожного компонента з урахуванням обмежень за щільністю розміщення та маршрутизації навантаження досі є трудомістким процесом, витвором мистецтва інженерів-електронників. Поєднання графової моделі з політикою та навчанням із підкріпленням дозволяє знаходити оптимальне розміщення мікросхем та створювати більш продуктивні чіпи порівняно з розробленими людьми.

Інший підхід передбачає інтеграцію моделі машинного навчання до вже існуючих інструментів рішення. Наприклад, колектив під керівництвом М. Гасса запропонував графову мережу, яка навчається політикам вибору змінних за методом гілок та кордонів: це критично важлива операція в інструментах рішення на основі частково-цілочисельних лінійних програм (MILP). В результаті вивчені уявлення намагаються мінімізувати тривалість роботи інструментів рішення та демонструють гарний компроміс між швидкістю виведення та якістю рішень.

У більш свіжій спільній роботі DeepMind та Google графові мережі використані у двох ключових підзавданнях, що вирішуються MILP-інструментами: спільному присвоєнні змінних та обмеження цільових значень. Запропонований підхід на основі нейромереж виявився в 2-10 разів швидше в

порівнянні з існуючими інструментами рішення при використанні величезних наборів даних, у тому числі застосовуваних у Google систем упаковки продукції та планування.

2.2.5 Комп'ютерний зір

Об'єкти у світі глибоко взаємопов'язані, тому зображення цих об'єктів можна успішно обробляти з допомогою графових нейромереж. Наприклад, можна сприймати вміст зображення через графи сцени – набір об'єктів на картинці зі своїми взаємозв'язками. Графи сцен застосовуються для пошуку зображень, розуміння їх вмісту та осмислення, додавання субтитрів, відповідей на візуальні питання та генерування зображень. Ці графи дозволяють сильно підвищити продуктивність моделей.

В одній із робіт Facebook описано, що можна помістити в кадр об'єкти, задати їх позиції та розміри, і на основі цієї інформації буде створено граф сцени. З його допомогою графова нейромережа визначає ембеддінги об'єктів, з яких, у свою чергу, згорткова нейромережа створює маски об'єктів, рамки та контури. Кінцеві користувачі можуть просто додавати до графа нові вузли (визначаючи відносне положення та розміри вузлів), щоб нейромережі могли генерувати зображення з цими об'єктами [8].

Інше джерело графів у комп'ютерному зорі - зіставлення двох взаємозалежних зображень. Це класичне завдання, для вирішення якого раніше створювали вручну дескриптори. Компанія Magic Leap, що спеціалізується на тривимірній графіці, створила архітектуру на основі графових нейромереж під назвою SuperGlue. Ця архітектура дозволяє в реальному часі зіставляти відеозаписи для тривимірного відтворення сцен, розпізнавання місць, одночасної локалізації та побудови картки (SLAM). SuperGlue складається з графової нейромережі на основі механізму уваги. Вона вчить знаходити

ключові точки зображення, які потім передаються на оптимальний транспортний шар для зіставлення. На сучасних відеокартах модель здатна працювати в реальному часі і може бути інтегрована до SLAM-систем.

2.2.6 Фізика і хімія

Подання взаємодій між частинками або молекулами у вигляді графів і прогнозуванням властивостей нових матеріалів і речовин за допомогою графових нейромереж дозволяє вирішувати різні природничі завдання. Наприклад, у рамках проекту Open Catalyst Facebook та CMU шукають нові способи зберігання відновлюваної енергії сонця та вітру. Одне з можливих рішень полягає у перетворенні цієї енергії за допомогою хімічних реакцій на інші види палива, скажімо, на водень. Але для цього потрібно створити нові каталізатори високо інтенсивних хімічних реакцій, а відомі сьогодні методи типу DFT дуже дорогі. Автори проекту виклали найбільшу добірку каталізаторів, DFT-загасань та базових рівнів для графових нейромереж. Розробники сподіваються знайти нові дешеві симуляції молекул, які доповнять поточні дорогі симуляції, що виконуються протягом днів, ефективними оцінками енергії та міжмолекулярних сил, які обчислюються протягом мілісекунд.

Дослідники з DeepMind також застосували графові нейромережі для емуляції динаміки комплексних систем частинок, таких як вода та пісок. Прогнозуючи на кожному кроці відносний рух кожної частки можна правдоподібно відтворити динаміку всієї системи та більше дізнатися про закони, які керують цим рухом. Наприклад, так намагаються вирішити найцікавіше з невирішених завдань у теорії твердого тіла – перехід у склоподібний стан. Графові нейромережі не тільки дозволяють емулювати динаміку під час переходу, але й допомагають краще зрозуміти, як частинки

впливають одна на одну залежно від часу та відстані.

Американська фізична лабораторія Fermilab працює над застосуванням графових нейромереж у Великому адронному колайдері для обробки мільйонів даних та пошуку тих з них, які можуть бути пов'язані з відкриттям нових частинок. Автори хочуть реалізувати графові нейромережі в програмованих логічних інтегральних схемах і вбудувати їх у процесори для збору даних, щоб можна було використовувати графові нейромережі віддалено з будь-якого куточка світу [8].

2.2.7 Розробка ліків

Фармацевтичні компанії активно шукають нові методи розробки ліків, жорстко конкуруючи один з одним та витрачаючи на дослідження мільярди доларів. У біології можна з допомогою графів представляти взаємодії різних рівнях. Наприклад, на молекулярному рівні зв'язку між вузлами позначатимуть міжатомні сили в молекулі, або взаємодія між амінокислотними основами в білку. У більшому масштабі графи можуть представляти взаємодію між протеїнами і РНК або продуктами обміну речовин. Залежно від рівня абстракції графи можна застосовувати для цільової ідентифікації, прогнозування властивостей молекул, високопродуктивного скринінгу, проектування нових ліків, конструювання протеїнів та перепрофілювання ліків [8].

Найімовірнішим результатом використання графових нейромереж у цій сфері стала робота дослідників з MIT (Массачусетський технологічний інститут), опублікована в Cell у 2020-му. Вони застосували модель глибокого навчання під назвою Chemprop, яка прогнозувала антибіотичні властивості молекул: пригнічення розмноження кишкової палички. Після навчання всього лише на 2500 молекул з бібліотеки, схваленої Управлінням з контролю за продуктами та ліками, Chemprop застосували до більшого набору даних, у тому

числі Drug Repurposing Hub, що містить молекулу Halicin, перейменовану на честь II HAL 9000 з фільму «Космічна одіссея 2001 року». Примітно, що до цього Halicin вивчали тільки стосовно лікування діабету, тому що її структура сильно відрізняється від відомих антибіотиків. Але клінічні експерименти in vitro та in vivo показали, що Halicin є антибіотиком широкого спектру. Широке зіставлення з сильними нейромережевими моделями наголосило на важливості виявлених за допомогою графових нейромереж властивостей Halicin. Крім практичної ролі цієї роботи архітектура Chemprop цікава й іншим: на відміну від багатьох графових нейромереж вона містить 5 шарів та 1600 прихованих вимірів, що набагато більше за типові параметри графових нейромереж для таких завдань..

2.3 Графова нейромережа MGT для розпізнавання начерків

Також у цьому проекті не обійтися без графової нейромережі MGT для розпізнавання начерків для гри “Намалюйка”. MGT - це архітектура нейромережі, яка адаптована для розпізнавання начерків. Модель обробляє начерки у вигляді графів. MGT вивчає геометричні та часові ознаки малюнків. Запропонований підхід протестували на датасеті Google QuickDraw. Результати MGT можна порівняти з підходом, заснованим на CNN: 72.80% точність розпізнавання проти 74.22%. При цьому модель обійшла підхід, який ґрунтується на RNN. Дослідники заявляють, що раніше скетчі не представлялися як графи і оброблялися з допомогою графових архітектур.

2.3.1 Навчання графової нейромережі MGT для розпізнавання начерків

Навчання уявленням малюнків є складним завданням через високу абстрактність начерків та розрідженість ліній. Існуючі підходи фокусуються на

статичності скетчів за допомогою згорткових мереж (CNNs). Крім цього, скетчі можуть розглядатись як послідовності ліній. Тоді їхнього представлення використовують рекурентні мережі (RNNs). Multi-Graph Transformer (MGT) працює зі скетчами як із графами. Графи захоплюють локальну і світову структуру ліній структури [9].

Архітектура MGT надихнута трансформером. Кожен шар моделі складається з двох компонентів:

- Multi-Graph Multi-Head Attention (MGMHA): підшар із механізмом уваги;
- Повнозв'язковий підшар із кодуванням позиції (positional encoding).

2.3.2 Тестування роботи моделі

Щоб перевірити роботу MGT, дослідники взяли завдання розпізнавання скетч на датасеті Google QuickDraw. Google QuickDraw складається з 414 тисяч зображень нарисів. Як базові моделі дослідники відібрали варіації state-of-the-art CNN архітектур і двонаправлені RNN. Нижче видно, що CNN-підходи видавали більш точні прогнози, ніж MGT. Незважаючи на це, MGT обійшла RNN-підходи.

Network	Configurations	Recognition Accuracy			Parameter Amount
		acc.@1	acc.@5	acc.@10	
Bi-directional LSTM #1	4D Input, $\vec{d} = 256, L = 4, Dropout_{LSTM} = 0.5, Dropout_{MLP} = 0.15$	0.6665	0.8820	0.9189	5,553,241
Bi-directional LSTM #2	4D Input, $\vec{d} = 256, L = 5, Dropout_{LSTM} = 0.5, Dropout_{MLP} = 0.15$	0.6524	0.8697	0.9133	7,130,201
Bi-directional GRU	4D Input, $\vec{d} = 256, L = 5, Dropout_{GRU} = 0.5, Dropout_{MLP} = 0.15$	0.6768	0.8854	0.9234	5,419,097
AlexNet (Krizhevsky et al., 2012)	Standard architecture and configurations	0.6808	0.8847	0.9203	58,417,305
VGG-11 (Simonyan & Zisserman, 2014)		0.6743	0.8814	0.9191	130,179,801
Inception V3 (Szegedy et al., 2016)		0.7422	0.9189	0.9437	25,315,474
ResNet-18 (He et al., 2016)		0.7031	0.9030	0.9351	11,353,497
ResNet-34 (He et al., 2016)		0.7009	0.9010	0.9347	21,461,657
ResNet-152 (He et al., 2016)		0.6924	0.8973	0.9312	58,850,713
DenseNet-201 (Huang et al., 2017)		0.7050	0.9013	0.9331	18,755,673
MobileNet V2 (Sandler et al., 2018)		0.7310	0.9161	0.9429	2,665,817
Vanilla Transformer (Vaswani et al., 2017)	$\vec{d} = 256, L = 4, I = 8, Dropout = 0.1$, Fully-connected graph	0.5249	0.7802	0.8486	14,029,401
MGT (Base)	$\vec{d} = 128, L = 4, I = 24, Dropout = 0.1, \mathbf{A}^{1\text{-hop}}, \mathbf{A}^{2\text{-hop}}, \mathbf{A}^{\text{global}}$ graphs	0.7070	0.9030	0.9351	10,096,601
MGT (Large)	$\vec{d} = 256, L = 4, I = 24, Dropout = 0.25, \mathbf{A}^{1\text{-hop}}, \mathbf{A}^{2\text{-hop}}, \mathbf{A}^{\text{global}}$ graphs	0.7280	0.9106	0.9387	39,984,729

Рисунок 2.15 – Результати порівнянь

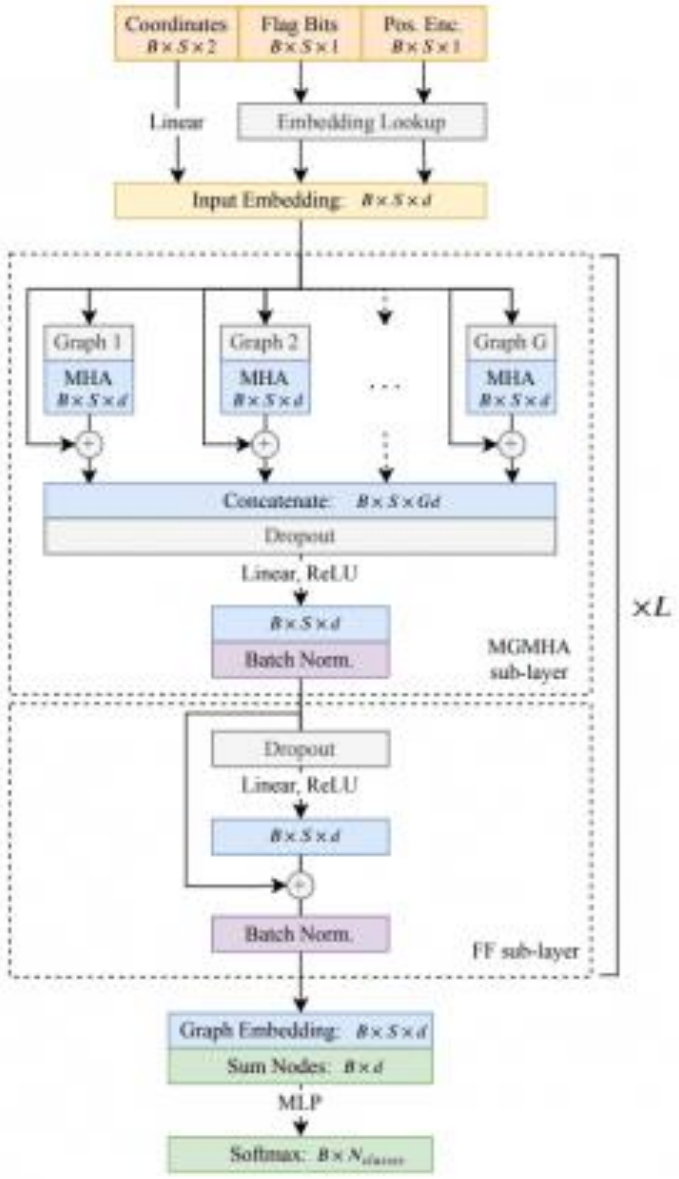


Рисунок 2.16 – Візуалізація складових частин MGT

2.4 Створення архітектури системи

Система складається з декількох основних частин або модулів, кожен з яких виконує свою унікальну функцію:

- модуль машинного навчання, який відповідає за цикли навчання нейронної мережі шляхом прийняття на вхідні дані зображення різних малюнків користувачів. Потім вихідні ваги між нейронами записуються в окремий файл;
- модуль обробки зображення, який масштабує, обрізає і нормалізує зображення;
- модуль включення відеокамери комп'ютеру, та розпізнавання долоні користувача.

2.5 Підготовка датасету до тренування моделі розпізнавання долоні

1. Розділити датасет на тренувальну, валідаційну та тестову частини. Як завжди, на тренувальній частині ми тренуємо модель, за допомогою валідаційної вибираємо, коли зупинити тренування, щоб не перенавчитися, а тестова частина – для оцінки точності моделі.
2. Змінити розмір зображення до 128×128. Рука - відносно простий об'єкт, тому такого маленького розміру буде достатньо. Також змінити розмір для координат ключових точок (рисунок 2.18).
3. Спочатку значення картинки в інтервалі [0,255]. Перевести у формат [0,1].
4. Нормалізувати зображення, використовуючи середні та стандартні відхилення, розраховані на тренувальному сеті. Для кожного колірного каналу (R, G, B) буде окреме середнє та окреме стандартне відхилення. Середні та стандартні відхилення вважаються за всіма пікселями колірного каналу у всіх зображеннях.

Створити хітмапи (heatmaps, теплові карти) на основі координат ключових точок. Хітмапи - це дуже популярний підхід для розпізнавання 2D-пози руки та пози людини (Human Pose Estimation). Хітмапи буквально є у кожному пейпері, може, з невеликими модифікаціями.

Отже, потрібен окремий heatmap для кожної ключової точки, разом 21 хітмапа для кожного зображення.

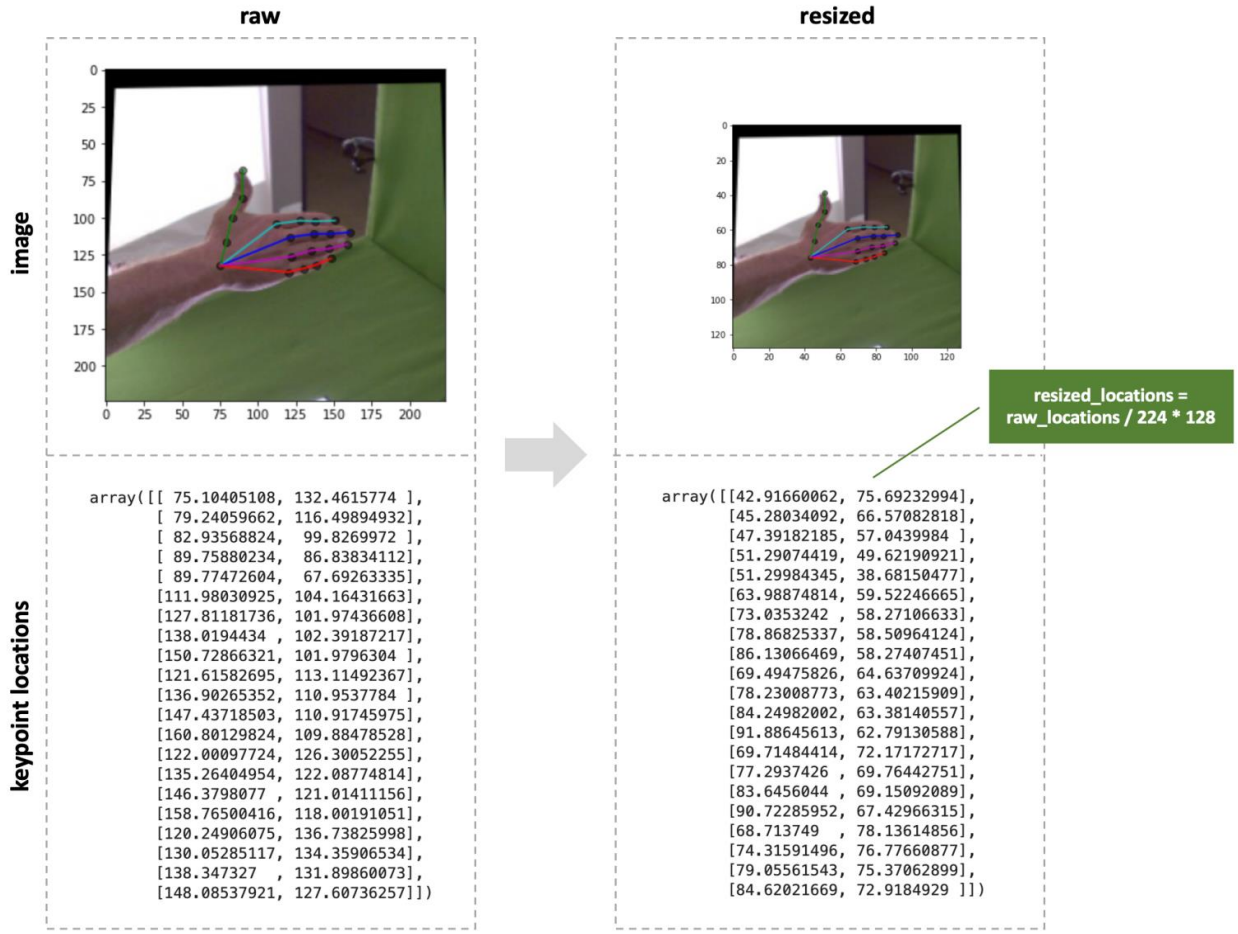


Рисунок 2.17 – Змінення координат ключових точок

Хітмапи потрібно розмивати, щоб запобігти перенавченню моделі і зробити процес тренування стабільнішим і швидшим. Параметри розмиття не такі важливі, тут працює правило «на око»: точка на хіт-мапі не повинна бути ні дуже великою, ні дуже маленькою. Нормалізація Min-Max потрібна, тому що у фінальному шарі нейронної мережі ми використовуватимемо сигмоїду.

Разом маємо:

- X - зображення розміру 3×128×128.

- *Y* - масив розміру 21×128×128, що складається із послідовно складених хітмап. У цих хітмап теж є порядок: перша хітмапа відповідає за розташування зап'ястя, остання - кінчика мізинця.

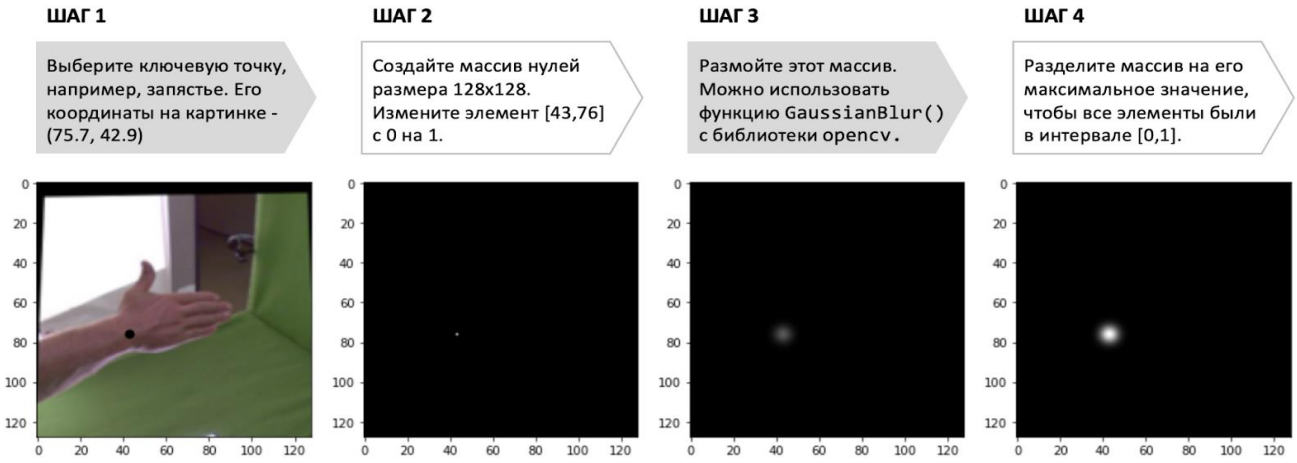


Рисунок 2.18 – Heatmap на основі координати ключової точки

2.6 Підготовка даних до тренування

Треба розділити датасет на тренувальну, валідаційну та тестову частини, як на схемі внизу. FreiHAND датасет виглядає перемішаним, тому треба ділити за індексами (індекси в назві файлів).

Рисунок 2.19 – Поділ датасету

Щоб вантажити та обробляти дані, нам потрібен клас DataLoader. Він збиратиме і завантажуватиме в оперативну пам'ять батчі даних і попередньо їх оброблятиме.

DataLoader. Власне клас, який формує батчі і послідовно завантажує їх в оперативну пам'ять.

Тепер циклом проходитись по даталодеру та отримувати батчі даних.

Dataset. У цьому класі прописується логіка, де саме брати зображення та розмітку та як їх попередньо обробляти.

2.7 Розрахунки для тренування

Більшість пейперів використовують IoU Loss для хітмап. IoU Loss використовується в задачах виявлення об'єктів (Object Detection), а після певної адаптації – для сегментації (приклад цього пейпера). Хітмапи схожі на сегментаційні маски, тому IoU Loss вважаємо за формулами нижче. І з таким IoU Loss модель тренується добре [10].

$$\square_{\square_0\square} = 1 - \square_0\square$$

$$\square_0\square = \frac{1}{\square}$$

$$\square = \sum_{\square} (\square_{\square} * \square_{\square})$$

$$\square = \sum_{\square} (\square_{\square} * \square_{\square}) + \sum_{\square} (\square_{\square} * \square_{\square}) - \sum_{\square} (\square_{\square} * \square_{\square}),$$

(2.1)

де $\square_{\square}$ – передбачення, $\square_{\square}$ – фактичні значення пікселів у хітмапах. IoU Loss рахується спочатку для кожної хітмапи окремо, потім усереднюється по 21

хітмапі, а потім усереднюється по всіх зображеннях в батчі.

2.8 Практика. Тренування моделі

1. В межах кожної епохи робити стадії тренування та оцінки (evaluation). Перед стадією тренування потрібно вручну переключити модель на стан «тренування» — командою model.train(), а перед оцінкою (чи передбаченнями) на стан «оцінка» — model.eval(). Це потрібно, тому що деякі шари поводяться по-різному під час тренування та передбачень, наприклад Dropout та BatchNorm.

2. Під час оцінки (і передбачень) використовувати команду torch.no_grad(). Це скаже моделі "Не рахуй градієнти зараз", і код буде працювати швидше і використовувати менше пам'яті.

3. За правилами PyTorch дані та модель повинні бути на одному девайсі. За замовчуванням дані та модель створюються на CPU, і їх потрібно буде перенести на GPU. Розмітку (labels) теж, інакше ви не зможете порахувати IoU Loss, адже передбачення моделі будуть на GPU.

4. Вибираємо розмір батча, який міститься у пам'яті (я брала 48), та кількість батчів за епоху «на око» (я брала 50). Початкову швидкість навчання (learning rate) ставила 0,1 і знижувати щоразу, коли тренувальний IoU Loss перестає знижуватися. Зупиняю тренування, якщо валідаційне IoU Loss не знижується протягом, наприклад, 10 епох.

5. З моїми параметрами тренування та на GPU модель натренувалася за 2 години та десь 200 епох. Фінальний IoU Loss на тренувальному та валідаційному сеті – 0,437 та 0,476, відповідно.

2.9 Передбачення моделі

Зараз модель віддає на вихід хітмапи. Але хітмапи - це ще не координати, тому потрібна додаткова постобробка [10].

Для кожної ключової точки руки одержати (x, y) координати, тобто розташування ключової точки на зображенні руки. Координати (x, y) будуть або в інтервалі [0, размер_картинки], або в інтервалі [0,1].

Так виглядають прогнози моделі. Легко можна визначити координати ключової точки при погляді на хіт-мапу. Ключова точка руки розташована десь у центрі "білої точки" на хітмапі. Та ж логіка використовується у постобробці.

Варіантів два:

- Простий. Знайти на хіт-мапі піксель з найбільшим значенням. Взяти його (x, y) координати. Це і будуть координати ключової точки руки.
- Складне, але надійне. Розрахувати середньозважені x та y за всіма пікселями хітмапи. Обираємо складний варіант (рисунок 2.23).

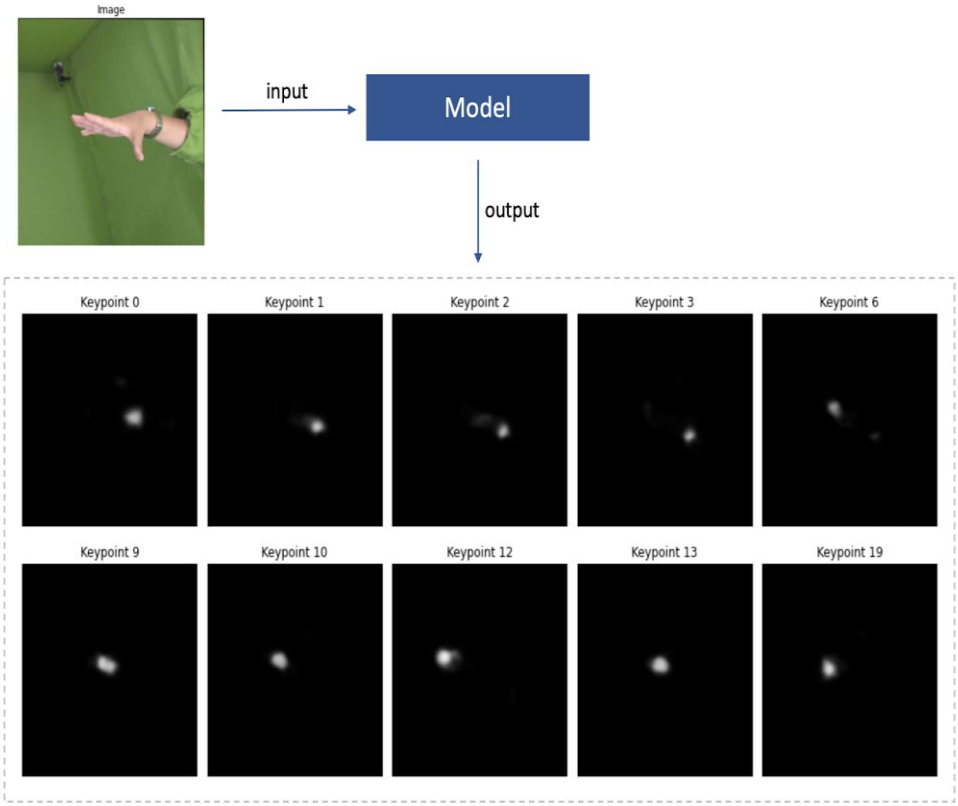


Рисунок 2.20 – Передбачення моделі для випадкового зображення руки

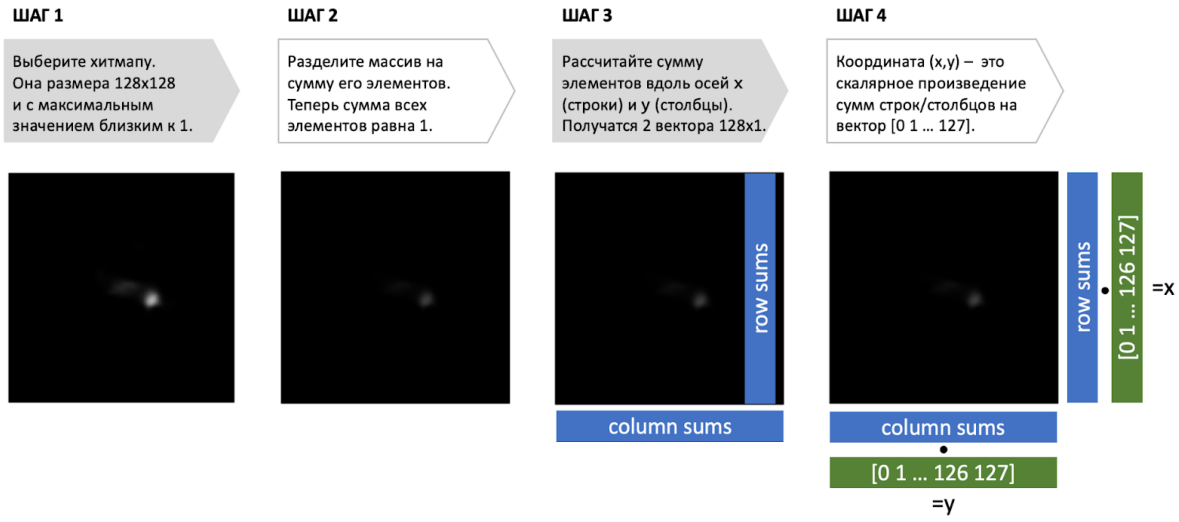


Рисунок 2.21 – Координати ключової точки з хітмапи

2.10 Оцінка точності моделі

Хороша практика - це завжди і візуально переглядати передбачення та розраховувати метрики. Найважливіше оцінити роботу моделі на тестовому сеті, тому що саме з такою точністю модель і працюватиме у продакшені на нових даних. Нижче наведено приклад, в якому форматі візуально переглядати передбачення. Можна для випадкових зображень відобразити фактичні координати ключових точок і поруч - передбачення моделі (вже після постачання). Позу руки зручно візуалізувати через скелет із різнокольоровими пальцями.

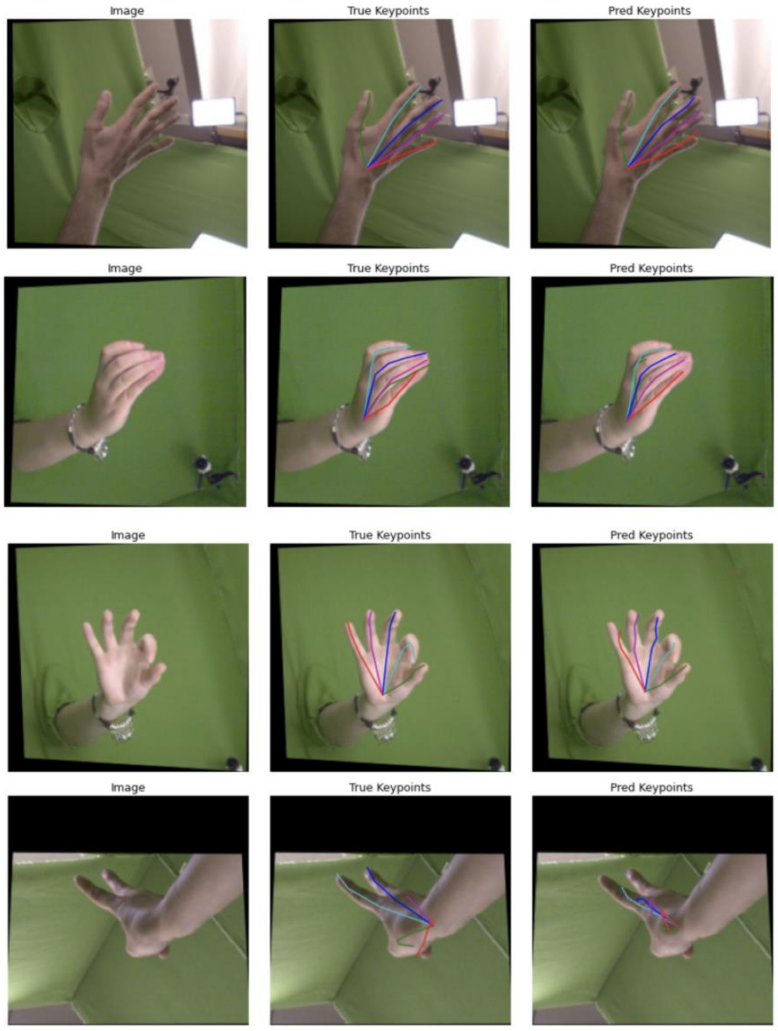


Рисунок 2.22 – Візуалізація передбачень моделі на тестовому сеті

Найголовніша метрика, яку важливо розрахувати, це середня помилка (error) для ключової точки по всьому тестовому сету [11].

Спочатку треба розрахувати помилку для кожної ключової точки окремо - як евклідова відстань між фактичними та передбаченими (x, y) координатами. Координати можна брати в пікселях - в інтервалі [0, розмір_зображення], або у відсотках від картинки - в інтервалі [0,1]. Знаходимо помилку по всіх ключових точках на зображенні і по всіх зображень в датасеті.

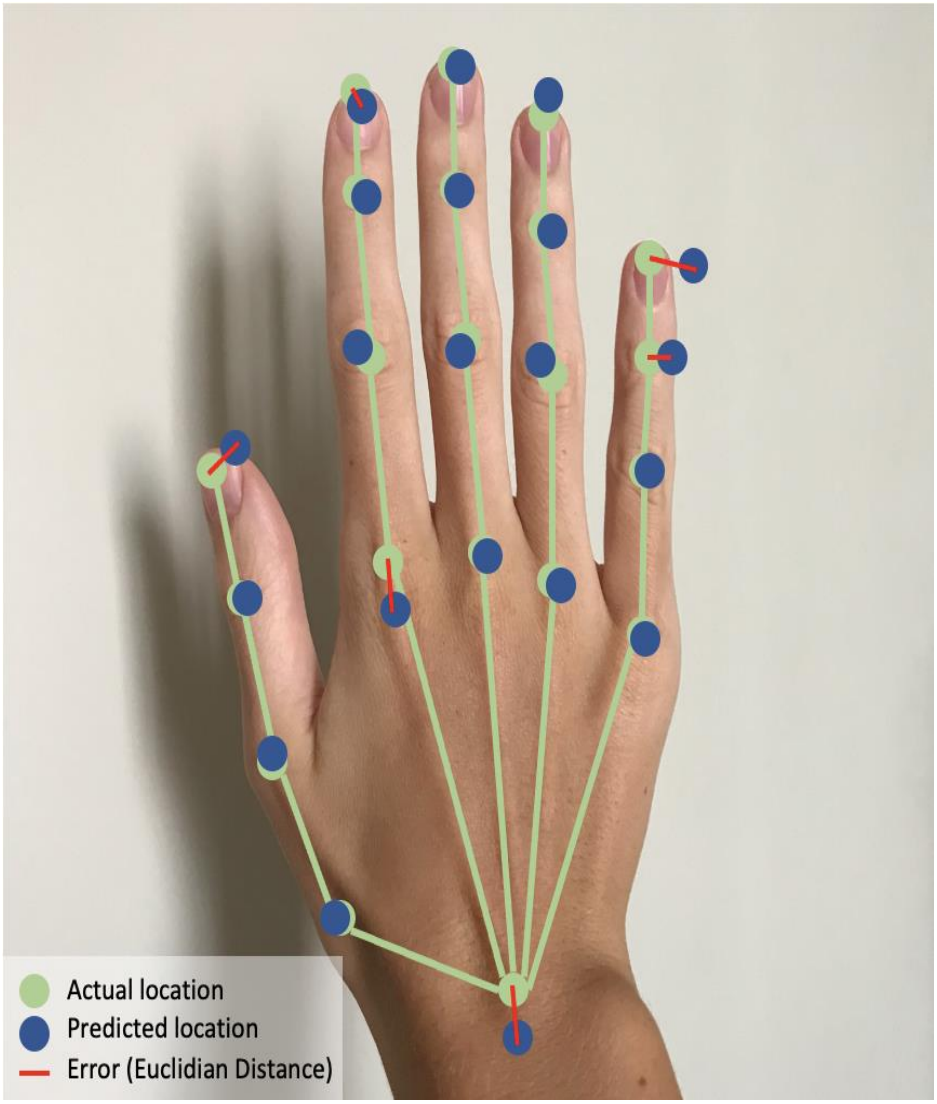


Рисунок 2.23 – Розрахунок помилки розпізнавання 2D-пози руки

Необхідно розрахувати середню помилку в датасеті по кожній з 21 ключової точки, або середню помилку на пальцях або суглобах, як іноді роблять у пейперах.

Таким чином, була натренована модель і отримана така точність: середня помилка для ключової точки на тестовому сеті - 4,5% розміру зображення, або 6 пікселів для картинок 128x128, або 8 пікселів для картинок 224x224. Це дуже гарні результати, лише приблизно 5% ця модель дає неточність.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір мови програмування

Для розробки обрано ОС Windows 10. Ця операційна система добре поширена, має сильну підтримку з боку виробника.

Обрана бібліотека ReactJS [12] на мові програмування JavaScript, тому що вона добре підходить для розробки алгоритмів машинного навчання, та має декілька необхідних додаткових бібліотек [13].

ReactJS – це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду – «компонентів». React може бути використаний як основа в розробці односторінкових (SPA) або мобільних застосунків, оскільки ця бібліотека забезпечує найшвидший спосіб отримання та відображення користувачу швидко змінюваних даних [14].

Програми вимагають використання додаткових бібліотек для управління станом, маршрутизації та взаємодії з API. React спрощує створення інтерактивних інтерфейсів, дозволяє створювати прості відображення для кожного стану програми та ефективно оновлює та відтворює лише потрібні компоненти, коли дані змінюються.

Використання React дозволяє створювати інкапсульовані компоненти, які керують власним станом, а потім повторно використовувати їх для створення складних інтерфейсів користувача. Оскільки логіка компонентів написана в JavaScript замість шаблонів, можливо легко передати багато даних через програму і зберегти її стан з DOM.

Бібліотека React Router – це API для застосунків, які використовують бібліотеку React. Маршрутизація («роутинг») – це процес, який відповідає за визначення обробника для конкретної запитуваної адреси [15].

У будь-якому реальному веб-застосунку потрібні маршрути, це забезпечує можливість бачити, де користувач знаходиться на сайті в будь-який момент часу, у адресному рядку браузера. Отже, веб-застосунок повинен вміти зіставляти певний URL з відповідною йому сторінкою. На даний момент є кілька популярних бібліотек для маршрутизації: react-router, router5, aviator тощо.

Більшість сучасних веб-застосунків написано з використанням React Router 4. Маршрутизатор React використовує динамічну маршрутизацію (тобто маршрутизацію, яка здійснюється під час рендерингу програми, а не в конфігурації застосунка). Маршрутизатор React та динамічна маршрутизація на стороні клієнта дозволяє створити односторінковий сайт з навігацією, без зайвого перезавантажування сторінки, під час навігації користувача.

Для виклику компонентів маршрутизатор React використовує їх структуру, а далі компоненти відображають відповідну інформацію. Роутер React також дозволяє користувачеві використовувати такі функції браузера, як кнопка «Назад» та «Оновити сторінку», зберігаючи правильний вигляд програми. Це дозволяє уникнути проблем, що характерні при використанні для фонового обміну між клієнтом та сервером технології AJAX, яка для забезпечення клієнтської навігації потребує окремого використання History API. Бібліотека react-router користується попитом серед розробників, оскільки надає такі можливості: навігація по кліку (компонент Link), перенаправлення (компонент Redirect), маршрутизація (компонент Route), історія (властивість history).

Для того, щоб користувач мав можливість запускати нейромережеві ігрові моделі на веб-сторінці, було прийняте рішення використовувати HTML та CSS для відображення [16].

Для мови програмування JavaScript добре підходить середа розробки VSCode.

3.2 Вибір бази даних для навчання нейронної мережі

Існує безліч наборів даних осіб, які можуть бути використані для оцінки алгоритмів. Є кілька популярних наборів даних для адаптації розпізнавання малюнків, наприклад quicklyDraw.
Для даного проекту, щоб впоратися з завданням, була обрана саме ця бібліотека та частково додані власні малюнки.

3.3 Додаткові бібліотеки

Було обрано бібліотеки Tensorflow, HandPose, FingerPose і Brain.JS.

3.3.1 TensorFlow

TensorFlow – це популярна бібліотека для машинного навчання, яку можна використовувати як у Python, так і JavaScript. Це бібліотека програмного забезпечення з відкритим кодом для чисельних розрахунків з використанням графів потоку даних [TensorFlow]. Вузли графу представлені у вигляді математичних операцій, в той час як ребра графу представляються багатовимірними масивами даних (тензорами), що передаються між вузлами [17].

TensorFlow був створений і підтримується командою Google Brain в рамках дослідницької організації Google Machine Intelligence для ML та DL. Зараз він випускається під ліцензією відкритого коду Apache 2.0. Інтерфейси програмування TensorFlow включають Python та C ++ з планами для API Java, GO, R та Haskell, також підтримуються хмарні середовища Google та Amazon. TensorFlow призначений для широкомасштабних розподілених тренувань, сюди входить 200 стандартних операцій, включаючи математичні, маніпуляції з

масивом, управління потоками та операції з управління станом, написані на C ++. На відміну від інших бібліотек DL, які в основному орієнтовані на дослідження (наприклад, Theano), TensorFlow був розроблений для використання як у системах досліджень, так і у розробці та виробництві ПЗ. TensorFlow підтримується процесорами, графічними процесорами, мобільними пристроями і широкомасштабними розподіленими системами, які складаються з сотень вузлів. Крім того, існує TensorFlow Lite – легке рішення TensorFlow для мобільних та вбудованих пристроїв [TensorflowLite]. Це дає змогу випускати ML-орієнтовані рішення на пристрої користувачів дуже швидко та з невеликим двійковим розміром, але має підтримку обмеженого набору систем. Також підтримується апаратне прискорення за допомогою Android Neural Networks API.

Сильні сторони:

- На сьогоднішній день найпопулярніший інструмент DL, з відкритим кодом, швидким залученням, який добре підтримується сильною промисловою компанією (Google).

- Потужна чисельна бібліотека для програмування потоків даних, яка є основою для досліджень та розробок DL.

- Ефективно працює з математичними виразами, що включають багатовимірні масиви.

- Дуже добре задокументована.

- Обчислення на GPU / CPU, мобільні обчислення, висока масштабованість обчислень на різних машинах та величезні набори даних.

- Вищий шар абстракції, ніж Theano.

Слабкі сторони:

API нижчого рівня важко використовувати безпосередньо для створення моделей DL.

Кожен обчислювальний потік повинен бути побудований як статичний

графік (хоча пакет Tensorflow Fold намагається полегшити цю проблему), а також відсутні символьни цикли.

3.3.2 HandPose

HandPose – заздалегідь навчена модель, яка застосовується на TensorFlow і допомагає знаходити руку та пальці.

Примітка: ця модель може розпізнавати лише одну руку під час введення - виявлення кількох рук з’явиться в майбутньому випуску [18].

MediaPipe Handpose — це легкий конвеєр ML, що складається з двох моделей: детектора долоні та моделі відстеження пальців скелета руки. Він передбачає 21 ключову точку тривимірної руки на кожну виявлену руку.

За вхідних даних модель передбачає, чи містить вона руку. Якщо так, модель повертає координати обмежувальної рамки навколо руки, а також 21 ключову точку всередині руки, що окреслює розташування кожного суглоба пальця та долоні.

3.3.3 FingerPose

FingerPose працює на основі даних HandPose і дозволяє легко знаходити саме жести, наприклад «кулак» або «ножиці».

Класифікатор жестів пальців для орієнтирів рук, виявлених MediaPipe Handpose. Він розпізнає такі жести, як «Перемога» або «Великий палець» +1 у вихідному зображенні чи відеопотоці. Ви можете визначити додаткові жести, використовуючи прості описи жестів [19].

Розпізнавання жестів складається з трьох кроків:

- Визначте орієнтири руки всередині відеозображення
- Оцінка напрямку та вигину кожного окремого пальця

- Порівняння результату з набором описів жестів.

3.3.4 Brain.JS

Brain.JS – дуже проста бібліотека для машинного навчання. Простіше кажучи, нейронна мережа – це метод навчання машини, який працює аналогічно до людського мозку. Зважаючи на правильну відповідь на запитання (наприклад, «який варіант буде вибрати цей користувач?»), він повільно вивчає шаблон і взаємозв'язок між входами та відповідями. Одним із прикладів нейронної мережі є система розпізнавання обличчя Facebook, Deepface [20]. Але через складну доменну мову нейронних мереж і, здавалося б, крутою кривою навчання, може бути важко почати.

3.4 Програмна реалізація розпізнавання нейромережею руки користувача

Поза руки визначається розташуванням ключових точок. Тобто під "розпізнати позу руки" мається на увазі "знайти координати ключових точок руки". У руки є 21 ключова точка (рисунок 3.1): зап'ясток плюс 5 пальців * 4 крапки на палець. І знаючи розташування кожної з точок (на картинці і щодо один одного), ми можемо однозначно визначити, який жест показує рука — «камінь», «ножиці» або «папір».

Порядок має значення. Потрібно не просто знайти хмару координат ключових точок - потрібно визначити координати зап'ястя, координати кінчика вказівного пальця, тощо.

В програмній реалізації координати пальців виглядають наступним чином:

```
const fingerPoints = {
    thumb: [0, 1, 2,  3, 4],
```

```
    indexFinger: [0, 5, 6, 7, 8],
    middleFinger: [0, 9, 10, 11, 12],
    ringFinger: [0, 13, 14, 15, 16],
    pinky: [0, 17, 18, 19, 20],
};
```

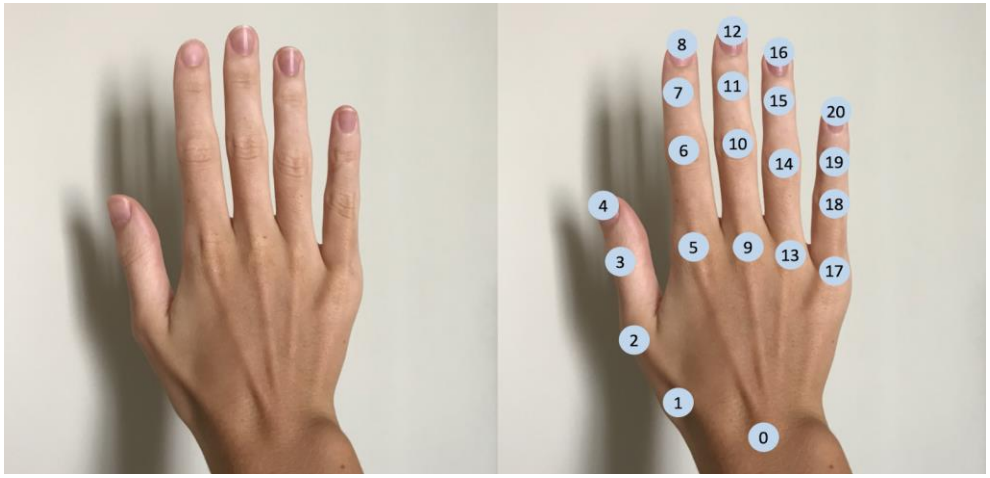


Рисунок 3.1 – Ключові точки руки за обраним порядком

Типова модель розпізнавання 2D-пози руки виглядає так:

- На вхід: картинка однієї руки. Немає інших частин тіла, великої площі із заднім фоном або ще рук.
- Вихід: список (□, □) координат ключових точок. Координати можна представити як піксельні координати, коли значення □ та □ в інтервалі [0, розмір_картинки], або як нормалізовані координати, коли значення x та y в інтервалі [0, 1]. Нормалізовані координати — це піксельні координати, розділені на розмір картинки.

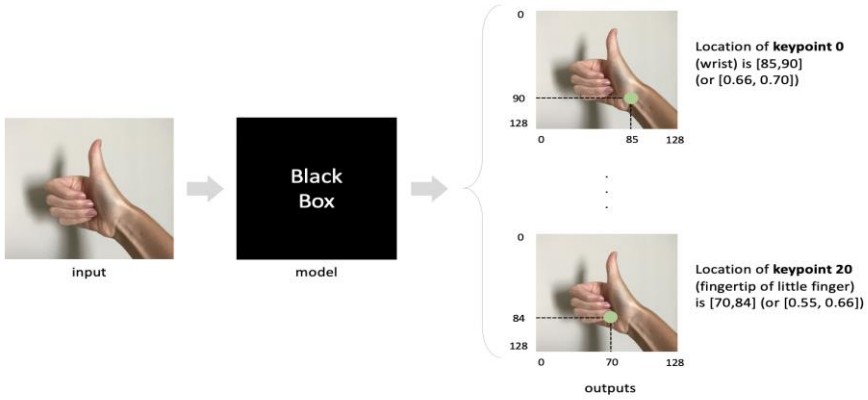
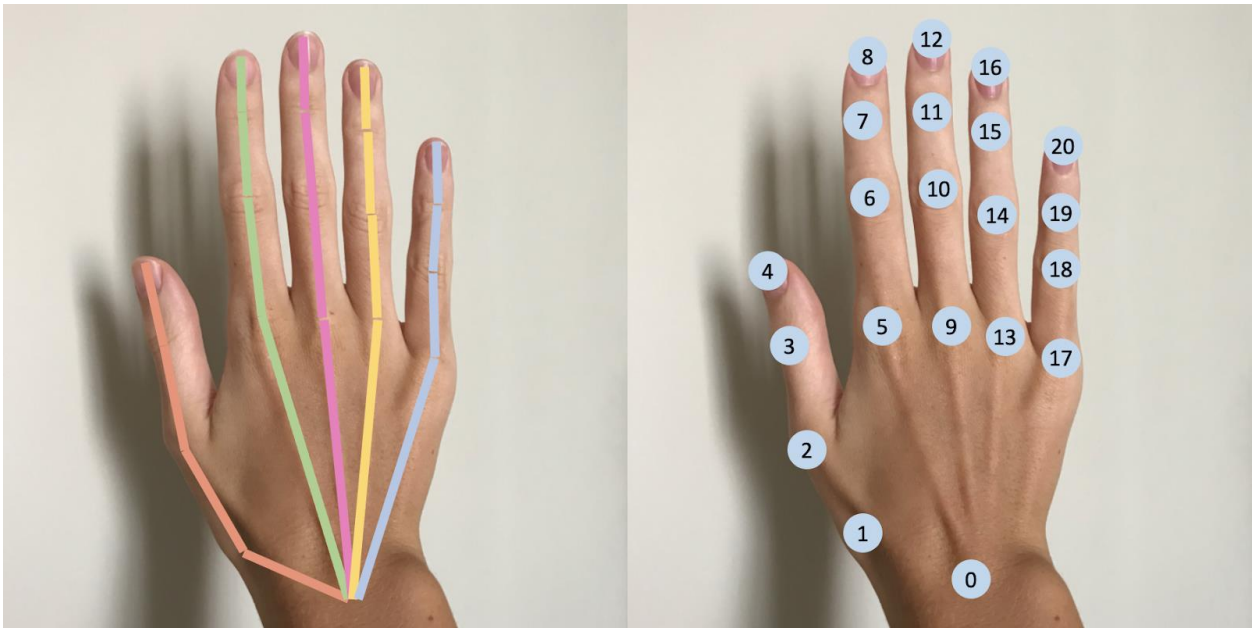


Рисунок 3.2 – Типова модель розпізнавання 2D-пози руки



а) б)

Рисунок 3.3 – Поза руки: а) візуалізація; б) ідентифікація

Позу руки показують як скелет з різнокольорових пальців. Скелет - це лише візуалізація, а поза руки все одно представлена як координати ключових точок. І знаючи координати кожної точки, нейромережа без проблем малює скелет.

Приклад програмного коду візуалізації пози руки:

```
export const drawHand = (predictions, ctx) => {
    if (predictions.length>0) {
        predictions.forEach((prediction)=> {
            const landmarks = prediction.landmarks;
            //переборка пальців рук
            for (let i = 0; i < Object.keys(fingerPoints).length;
i++) {
                //визначення пальця
                let finger = Object.keys(fingerPoints)[i];
                //переборка точок на пальцях
                for (let p = 0; p < fingerPoints[finger].length-
1;p++) {
                    const firstIndex = fingerPoints[finger][p];
                    const secondIndex = fingerPoints[finger][p+1];
                    //малюється лінія
                    ctx.beginPath();
                    ctx.moveTo(
                        landmarks[firstIndex][0],
                        landmarks[firstIndex][1]
                    );
                    ctx.lineTo(
                        landmarks[secondIndex][0],
                        landmarks[secondIndex][1]
                    )
                    ctx.strokeStyle = "plum";
                    ctx.lineWidth = 4;
                    ctx.stroke();
                }
            }
        })
    }
}
```

Таким чином, за допомогою моделі HandPose та бібліотеки FingerPose, проаналізовано модель для розпізнавання 2D-пози руки та реалізоване розпізнавання нейромережею руки користувача для гри “Камінь-ножиці-папір”.

ВИСНОВКИ

Робота присвячена актуальній проблемі – розробці комп’ютерних рішень для розвитку логічного, творчого мислення та креативності дітей. А це дуже важливо навантажувати та розвивати обидві півкулі мозку, особливо для дітей.

Існує велика кількість способів розвитку логічного, творчого мислення та креативності у дітей. Наприклад: настільні ігри, конструктор, головоломки, ігри в усній формі, ребуси та анаграми та інші. Але все вони не є ідеальними рішеннями. Ці способі більш застарілі.

Були розглянуті існуючі рішення з нейромережею. Нажаль, на сьогодні не має різноманітності у програмах з нейромережею. Була протестована програма QuickDraw від Google, проведений аналіз, порівняння, виявлені переваги та недоліки та зроблені висновки.

Була поставлена задача – вибрати нейронну мережу і реалізувати програму, яка складатиметься з трьох ігор для розвитку логічного, творчого мислення та креативності у дітей за допомогою нейромережевих ігрових моделей:

- гра “Намалюйка” - користувач малює зображення, а нейромережа відгадує;
- гра “Камінь-ножиці-папір” - гра з противником у вигляді штучного інтелекту та нейромережею, яка розпізнає долоню та жест користувача;
- Гра “Відгадайка” – штучний інтелект загадує число та користувач намагається відгадати за найменшу кількість спроб.

Проведено аналіз видів нейронних мереж, розглянуті їх архітектури, функції та застосування. Обрані графові нейромережі, бо вони використовуються для класифікації тексту, виявлення об'єктів на зображеннях, задачі комбінаторної оптимізації, якраз те, що потрібно у цьому проекті.

Обрано графову нейромережу MGT (Multi-Graph Transformer) для розпізнавання начерків для гри “Намалюйка” та моделі HandPose (допомагає знаходити руку та пальці) та FingerPose, (дозволяє легко знаходити жести).

Проведено тренування моделі розпізнавання долоні та отримано наступні результати: середня помилка для ключової точки на тестовому сеті - 4,5% розміру зображення, або 6 пікселів для картинок 128x128, або 8 пікселів для картинок 224x224. Похибка точності моделі 5%.

Для реалізації поставленої задачі використано бібліотеку React та доповнення до цього JavaScript-фреймворка. Для допомоги в роботі з нейромережею були обрані бібліотеки: Tensorflow, HandPose, FingerPose і Brain.JS.

Поставлені в роботі задачі виконані повністю. Розроблено програмний додаток для розвитку логічного, творчого мислення та креативності у дітей за допомогою нейромережевих ігрових моделей. Є різноманітність в іграх, зрозумілий інтерфейс.

Таким чином, результат даної кваліфікаційної роботи може бути суттєвим комп'ютеризованим та сучасним доповненням до вже існуючих рішень розвитку логічного, творчого мислення та креативності у дітей.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розвиток мислення у дітей. URL: https://sfp.org.ua/rozvitok-mislennya-u-ditej/ (дата звернення 28.11.2022).
2. QuickDraw від Google. URL: https://quickdraw.withgoogle.com/ (дата зернення 28.11.2022).
3. Субботін С. О. Нейронні мережі: теорія та практика: навч. посіб. Житомир: Вид. О. О. Євенок, 2020. 184 с.
4. Заенцев И.В. Нейронные сети: основные модели. Воронеж, 1999. 76 с.
5. Троцько В. В. Методи штучного інтелекту: навчально-методичний і практичний посібник. Київ, 2020. 86 с.
6. Графові нейронні мережі. Розвиток GCN. URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%93%D1%80%D0%B0%D1%84%D0%BE%D0%B2%D1%8B%D0%B5_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8 (дата зернення 28.11.2022).
7. Каллан Р., Вільямс І. Д. Нейронні мережі: Короткий довідник, 2017. 288 с.
8. Що таке графові нейронні мережі. URL: https://habr.com/ru/company/vk/blog/557280/ (дата звернення 28.11.2022).
9. Ніколенко С., Кадурін А., Архангельська Е. Глибоке навчання. СПб, 2018, 480 с.
10. Редько В. Г. Еволюція, нейронні мережі, інтелект: Моделі і концепції еволюційної кібернетики. М .: Ленанд, 2019. 224 с.
11. Хайкін С. Нейронні мережі: повний курс. М .: Діалектика, 2019. 1104 с.
12. ReactJS. URL: https://reactjs.org/ (дата звернення 28.11.2022).
13. Фаулер, Мартин. Рефакторинг кода на JavaScript: улучшение проекта существующего кода, 2-е изд. СПб.: ООО «Диалектика», 2019. 464 с.

14. React. URL: https://reactdev.ru/handbook/tutorial/ (дата звернення 28.11.2022).
15. React Router. URL: http://fpm.dnu.dp.ua/2019/12/react-router/ (дата звернення 28.11.2022).
16. Роббинс, Дж. HTML5, CSS3 и JavaScript. Исчерпывающее руководство. М.: Эксмо, 2014. 528 с.
17. TensorFlow. URL: https://www.tensorflow.org/ (дата звернення 28.11.2022).
18. HandPose. URL: https://github.com/tensorflow/tfjs-models (дата звернення 28.11.2022).
19. FingerPose. URL: https://github.com/andypotato/fingerpose (дата звернення 28.11.2022).
20. Brain.JS. URL: https://brain.js.org/#/ (дата звернення 28.11.2022).