Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Аналіз методів створення нейронних мереж з використанням мови програмування JavaScript

Виконав: студент VI курсу, групи СТм-61
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

_____ Сербін В. С.
(підпис) (прізвище та ініціали)

Керівник _____ Готович В. А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Осухівська Г. М.
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« 28 » грудня 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Студенту Сербіну Володимиру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Аналіз методів створення нейронних мереж з використанням мови програмування JavaScript»

Керівник роботи Готович Володимир Анатолійович, кандидат технічних наук
доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » листопада 2023 року № 4/7-1097

2. Термін подання студентом завершеної роботи 29 грудня 2023р.

3. Вихідні дані до роботи базуються на основі вхідних даних та результатах дослідження предметної області, використаних літературних джерелах, розглянутих інтернет-ресурсах, статтях, незалежних дослідженнях на тему методів створення нейронних мереж.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Дослідження предметної області. 1.1 Обґрунтування актуальності розробки нейронних мереж. 1.2 Інтеграція поширених моделей нейронних мереж у секторі безпеки. 1.3 Аналіз результатів прогнозування нейронних мереж. 1.4 Дослідження тенденцій поширення нейронних мереж в IT та інших сферах діяльності. 1.5 Аналіз ризиків впливу нейронних мереж на життєдіяльність людини. 1.6 Висновок до першого розділу. 2 Аналіз методів створення нейронних мереж. 2.1 Обґрунтування вибору платформи JavaScript. 2.2 Аналіз методів створення та навчання нейронних мереж. 2.3 Дослідження фреймворків для розробки нейронних мереж на мові програмування JavaScript. 2.4 Висновок до другого розділу. 3 Створення нейронних мереж на мові програмування JavaScript. 3.1 Визначення вимог до розробки нейронних мереж. 3.2 Процес створення нейронних мереж на мові програмування JavaScript. 3.3 Висновок до третього розділу. 4 Охорона праці та безпека в надзвичайних ситуаціях. 4.1 Охорон праці. 4.2 Безпека в надзвичайних ситуаціях. 4.3 Висновок до четвертого розділу. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження. 3 Завдання дослідження. 4 Актуальність дослідження. 5 Аналіз основних моделей нейронних мереж. 6 Дослідження методів створення нейронних мереж на мові програмування JavaScript. 7 Інструменти створення нейронних мереж. 8 Розробка алгоритмів для створення нейронних мереж. 9. Аналіз результатів створення нейронних мереж. 10 Висновки. 11 Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 24 листопада 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.11.2023	*Виконано*
2.	Підбір наукових джерел про поширені моделі нейрон-	26.11.2023-28.11.2023	*Виконано*
	них мереж та методи їх створення з використанням мо-		
	ви програмування JavaScript.		
3.	Опрацювання наукових публікацій та збір даних по	29.11.2023-1.12.2023	*Виконано*
	темі роботи		
4.	Виконання дослідження згідно мети кваліфікаційної	2.12.2023-4.12.2023	*Виконано*
	роботи		
5.	Оформлення розділу «Дослідження предметної обла-	5.12.2023-7.12.2023	*Виконано*
	сті»		
6.	Оформлення розділу «Аналіз методів створення ней-	8.12.2023-10.12.2023	*Виконано*
	ронних мереж»		
7.	Оформлення розділу «Створення неронних мереж на	11.12.2023-13.12.2023	*Виконано*
	мові програмування JavaScript»		
8.	Виконання завдання до підрозділу «Охорона праці»	14.12.2023-15.12.2023	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	16.12.2023-17.12.2023	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	18.12.2023-19.12.2023	*Виконано*
11.	Нормоконтроль	19.12.2023-20.12.2023	*Виконано*
12.	Перевірка на плагіат	21.12.2023	*Виконано*
13.	Попередній захист кваліфікаційної роботи	22.12.2023	*Виконано*
14.	Захист кваліфікаційної роботи	29.12.2023	

Студент _____ Сербін В. С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Готович В. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Аналіз методів створення нейронних мереж з використанням мови програмування JavaScript // Кваліфікаційна робота освітнього рівня «Магістр» // Сербін Володимир Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СТм-61 // Тернопіль, 2023 // С. 93, рис. – 12, табл. – 0, кресл. – 11, додат. – 3, бібліогр. – 50.

Ключові слова: нейронні мережі, моделі нейронних мереж, методи розробки нейронних мереж, згорткові нейронні мережі, рекурентні нейронні мережі, штучний інтелект, TensorFlow.js, Brain.js.

Кваліфікаційна робота присвячена аналізу методів створення нейронних мереж. В результаті виконання кваліфікаційної роботи здійснено огляд тенденцій розвитку предметної області та проведено аналіз сучасних методів розробки нейронних мереж на прикладі мови програмування JavaScript, в порівнянні з іншими аналогами.

Мета роботи полягає в аналізі наявних на даний момент методів розробки нейронних мереж та визначенні можливості їх застосування при створенні нейронної мережі на мові програмування JavaScript, з врахуванням переваг та недоліків кожного методу, а також переваг та обмежень фреймворків мови JavaScript.

В першому розділі кваліфікаційної роботи проведено дослідження предметної області на тему актуальності розробки нейронних мереж. Виділені основні сфери застосування програмних засобів на базі нейронних мереж та виконано аналіз результатів їх застосування. Наведено приклади успішного застосування згорткових, рекурентних та генеративно-змагальних моделей нейронних мереж. Також проведено аналіз впливу нейронних мереж на їх користувачів на основі використання останніми можливостей різних моделей.

В другому розділі кваліфікаційної роботи проведено аналіз існуючих методів розробки нейронних мереж та можливості їх застосування для створення рішень штучного інтелекту на основі мови програмування JavaScript. Виконано дослідження моделей розробки нейроних мереж на основі їх параметрів з обґрунтуванням переваг та недоліків кожного, а також можливостей їх використання при розробці нейронної мережі на базі фреймворків мови програмування JavaScript.

В третьому розділі кваліфікаційної роботи наведено приклад практичного застосування розглянутих інструментів та проаналізованих методів розробки нейронних мереж при створенні програмного рішення. Виконано тестування створеного програмного рішення та аналіз отриманих результатів.

В четвертому розділі кваліфікаційної роботи описані основні аспекти охорони праці в предметній області. Розглянуто способи виконання завдання з підвищення стійкості роботи об’єктів приладобудування у воєнний час.

ANNOTATION

Analysis of neural networks using JavaScript programming language // The educational level "Master" qualification work // Serbin Volodymyr Sergiyovich // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, STm-61 group // Ternopil, 2023 // P. 93, fig. – 12, tables – 0, posters – 11, annexes – 3, ref. – 50.

Key words: neural networks, neural network models, neural network development methods, convolutional neural networks, recurrent neural networks, artificial intelligence, TensorFlow.js, Brain.js.

The qualification work is devoted to the analysis of methods of creating neural networks. As a result of the qualification work, an overview of the development trends of the subject area was carried out and an analysis of modern methods of developing neural networks was carried out using the example of the JavaScript programming language, in comparison with other analogues.

The purpose of the work is to analyze the currently available methods of developing neural networks and determine the possibility of their application when creating a neural network in the JavaScript programming language, taking into account the advantages and disadvantages of each method, as well as the advantages and limitations of JavaScript language frameworks.

In the first section of the qualification work, a study of the subject area was carried out on the topic of the relevance of the development of neural networks. The main areas of application of software tools based on neural networks are highlighted and the results of their application are analyzed. Examples of successful application of convolutional, recurrent, and generative-competitive models of neural networks are given. An analysis of the influence of neural networks on their users was also carried out based on the latter's use of the capabilities of various models.

In the second section of the qualification work, an analysis of the existing methods of developing neural networks and the possibility of their application for

creating artificial intelligence solutions based on the JavaScript programming language was carried out. The study of neural network development models based on their parameters with justification of the advantages and disadvantages of each, as well as the possibilities of their use in the development of a neural network based on the frameworks of the JavaScript programming language, was carried out.

In the third section of the qualification work, an example of the practical application of the considered tools and analyzed methods of developing neural networks in the creation of a software solution is given. Testing of the created software solution and analysis of the obtained results was performed.

The fourth chapter of the qualification work describes the main aspects of labor protection in the subject area. Ways of performing the task of increasing the stability of the operation of instrument-making objects in wartime are considered.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – application programming interface (прикладний програмний інтерфейс).

GPU – graphics processing unit (графічний процесор).

JS – JavaScript (скриптова мова прогармування).

JSON – JavaScript Object Notation (текстовий формат обміну даними між комп’ютерами).

NLP – natural language processing (обробка природної мови).

ІТ – інформаційні технології.

КНР – Китайська Народна Республіка.

ПЗ – програмне забезпечення.

США – Сполучені Штати Америки.

ШНМ – штучні нейронні мережі.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Станом на дві тисячі двадцять третій рік нейронні мережі є однією з найбільш актуальних та перспективних розробок галузей інформаційних технологій та науки. Нейронні мережі та похідний від них штучний інтелект мають широкий спектр застосування в різних сферах життєдіяльності людини: від гаджетів до секторів безпеки, від перевірки правопису до виконання складних багатовимірних обчислень, від класифікації об’єктів до прогнозування різного роду процесів.

Нейронні мережі ефективно використовують накопичені людством великі масиви структурованих та неструктурованих даних для вирішення різноманітних завдань. Наприклад, виконуючи процес аналітики Big Data штучний інтелект здатен швидко обробляти інформацію та, застосовуючи аналітичні методи пошуку зв’язків і закономірностей, виконувати прогнозування в різних сферах діяльності. У дві тисячі сімнадцятому році ООН та Twitter спільно запустили нейронну мережу, основною метою якої було використання машинного навчання на базі Big Data – загальнодоступних даних з Twitter – для допомоги у розробці інституційної політики проти ксенофобії, дискримінації та расизму по відношенню до мігрантів та біженців [1].

Не останню роль у процесі стрімкого росту нейронних мереж відіграють постійно зростаючі обсяги ресурсів та потужності електронних обчислювальних машин. Розвиток обчислювальних систем дає можливість розробляти та практично застосовувати нові, складніші алгоритми, патерни чи архітектури для вирішення поставлених завдань. Потужніші обчислювальні машини створили нейронним мережам умови для досягнення значного прогресу в напрямку обробки нейронної мови – NLP. Наприклад, transformer-модель машинного навчання дозволила ChatGPT-4 оволодіти навичками створення реалістичні форми висловлень у відповідь на повідомлення користувачів платформи OpenAI.

Крім того, виробники потужних обчислювальних пристроїв активно долучаються до створення нейронних мереж з метою вирішення глобальних проблем. Компанія NVIDIA, для прикладу, намагається посприяти вирішенню

кліматичних проблем за посередництва створення нейронної мережі на базі суперкомп'ютера «Earth-2», що буде моделювати кліматичні зміни на планеті з метою передбачення наслідків шкідливого впливу мінливого клімату та їх пом'якшення.

Популярні з дві тисячі двадцять першого року мультимодальні нейронні мережі продовжують активний розвиток в секторі обробки та інтеграції бінарних типів даних. Найбільшої популярності в якості представника такого типу нейронних мереж досягнув DALL-E, що здатен створювати чи змінювати різного роду зображення, в тому числі й реалістичні, на основі текстового опису будь-якого виду.

Комп'ютерний зір – інша сфера застосування нейронних мереж з високого рівня результативністю використання машинного навчання та штучного інтелекту. Застосування сучасних патернів обробки об'єктів дало можливість навчити програмні засоби ефективно виконувати сематичну сегментацію з метою розпізнавання різних об'єктів, стилів та навіть облич людей.

Провідну роль нейронні мережі виконують і в сфері науки. Так, нейронна мережа AlphaFold здатна передбачати тривимірну структуру білків за їх амінокислотними послідовностями, а SchNet – вміє моделювати властивості та взаємодію молекул за їхніми атомними координатами.

Більшість з поданих типів нейронних мереж можуть бути реалізовані у вигляді веб-інтерейсу та засобами мов програмування сфери веб-розробки, такими як JavaScript, що є дуже актуальним з огляду на зростаючі обсяги використання веб-сервісів. Одна з найпопулярніших, на даний момент, нейронних мереж – ChatGPT – використовується, переважно, у вигляді веб-сторінки. Аналоги ChatGPT, DALL-E, StableDiffusion та ніших популярних представників NLP та мультимодальних типів нейронних мереж також, в основному, використовуються у вигляді веб-сайту. Тим не менше, розробка програмної обчислювальної частини цих сервісів відбувається на платформі Python, що може створювати певні труднощі при інтеграції з веб-сервісами та запуску на веб-серверах. З точки зору сталої популярності JavaScript у розробці веб-сервісів та наявності у цієї мови програмування необхідних фреймворків

формування нейронних мереж, найбільш оптимальним варіантом для реалізації нейронних мереж у вигляді веб-сторінок було б використання платформи JavaScript.

Окрім вищесказаного, на сьогоднішній день існує велика кількість досліджень, літератури, статей на тему застосування нейронних мереж, перспектив подальшого розвитку та покращення методів машинного навчання, розробки нейронних мереж, ризиків широкого впливу нейронних мереж на сфери застосування, тощо, які регулярно доповнюються новими об’єктами. Це також свідчить про актуальність та зацікавленість в створенні та поширенні нейронних мереж.

Мета і задачі дослідження. Мета роботи полягає в аналізі наявних на даний момент методів розробки нейронних мереж та визначенні можливості їх застосування при створенні нейронної мережі на мові програмування JavaScript, з врахуванням переваг та недоліків кожного методу, а також переваг та обмежень фреймворків мови JavaScript.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1) Провести дослідження предметної області в контексті розкриття актуальності розробки нейронних мереж та подальших перспектив розвитку, аналізу застосування поширених моделей нейронних мереж у різних сферах діяльності.

2) Здійснити огляд моделей нейронних мереж та варіантів їх застосування.

3) Здійснити аналіз методів розробки нейронних мереж та машинного навчання.

4) Навести порівняльну характеристику фреймворків мови програмування JavaScript, які здатні виконувати завдання створення нейронних мереж.

5) Розробити програмну реалізацію рішення, що функціонує на основі нейронних мереж на мові програмування JavaScript та обраного фреймворка.

Для виконання наведених завдань буде застосовано спеціальне програмне забезпечення у вигляді окремих пошукових систем, текстових редакторів, мови програмування JavaScript та відповідних їй фреймворків.

Об’єкт дослідження моделі нейронних мереж.

Предмет дослідження. Методи створення нейронних мереж.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у здійсненні порівняльного аналізу методів створення нейронних мереж на базі інструментів мови програмування JavaScript.

Практичне значення одержаних результатів полягає у демонстрації наявності конкурентно здатних альтернатив мові програмування Python в сфері розробки нейронних мереж шляхом аналізу методів їх створення на мові програмування JavaScript, висвітлення переваг та недоліків альтернативних фреймворків мови програмування JavaScript.

Апробація результатів магістерської роботи. Результати проведених досліджень обговорювались на конференціях:

- VI міжнародна студентська науково-технічна конференція «Природничі та гуманітарні науки. Актуальні питання» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2023 р.);
- IV міжнародна науково-практична конференція учених та студентів «Цифрова економіка як фактор інновацій та сталого розвитку суспільства» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2023 р.);
- X науково-технічна конференція «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2022 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у трьох працях конференцій (додатки А, Б, В, Д).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 50 найменувань та 4 додатків. Загальний обсяг кваліфікаційної роботи складає 91 сторінку, з них 64 сторінки основного тексту, що містить 12 рисунків.

1 АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ ПО ТЕМІ ДОСЛІДЖЕННЯ

1.1 Обґрунтування актуальності розробки нейронних мереж

Нейронні мережі є одним з найбільш актуальних і перспективних напрямків розвитку сфери інформаційних технологій. Нейронні мережі є програмним (оцифрованим) втіленням біологічних нейронів людського мозку, які імітують поведінку останніх при обробці отримуваної інформації. Нейронні мережі мають здатність навчатися на основі вхідних даних, адаптуватися до змінних умов і вирішувати складні завдання, які традиційні алгоритми, наприклад, Маркова та Гауса, не можуть ефективно виконувати.

Актуальність нейронних мереж обумовлена рядом факторів. В першу чергу, стрімким зростанням обсягів і різноманітності даних. Цей фактор пов’язаний з тим, що в сучасному світі створюється та накопичується величезна кількість даних. Це може бути результатом:

1) життєдіяльності людини (соціальні мережі, інтернет-ресурси та сервіси);

2) спостережень сектору безпеки (записи камер, датчиків руху, мікрофонів);

3) наукових експериментів та досліджень:

4) обчислень, інформація з давачів, супутників тощо.

Зазвичай, перед обчислювальними системами стоїть завдання швидкої та результативної обробки цих даних. Однак, традиційні алгоритми не здатні забезпечити гарантований результат виконаної процедури, який може мати значну похибку чи взагалі не відобразитись через неповноту даних, їх велику розмірність, складність, шум, або неможливість комбінування, включаючи інші дані в обчислення.

За даними анонімного статистичного порталу, у 2020 році обсяг Big Data склав трохи більше 64 зеттабайт, а до кінця 2025 року очікується, що він зросте до 460 зеттабайт. Ці дані містять важливу інформацію, яка може бути використана для покращення якості життя, науки, бізнесу, безпеки і інших сфер, однак, її ручна обробка може зайняти величезну кількість часу. Для прикладу,

навіть найбільш популярні соціальні мережі вже давно не здатні впоратись з обсягом завантажуваного користувачами контенту: відстань між публікацією користувачем фотографії та її модерацією на предмет порушення правил користування сервісом може складати від трьох діб до тижня часу. Що вже казати про багаторівневі обчислення чи комплексні дослідження, результати яких можуть розглядатись тижнями, а то й місяцями.

З метою ефективного використання цих даних потрібно мати комплекс засобів для їх збирання, зберігання, сортування, аналізу передачі чи поширення і візуалізації.

Штучний інтелект – це технологя загального призначення, яка вважається однєю з найперспективнших, оскільки здатна обробити цю величезну кількість даних, використовуючи свою здатність до самонавчання, адаптації, абстрагування, та використати їх для релевантного пошуку інформації, створення нового мультимедійного чи текстового контенту, математичного аналізу, прогнозування різноманітних процесів, вдосконалення різного роду систем, тощо [2].

Нейронні мережі можуть працювати з різними типами даних, такими як зображення, звук, текст, відео, числа, символи, графи і багато інших, вивчаючи закладені в них закономірності, залежності, особливості, структуру, не вимагаючи для цього, передуючим процесу вивчення, обробки, визначення правил тощо.

Наприклад, наявність значної кількості даних про споживання газу дозволяє ефективно застосовувати методи машинного навчання та прийняття рішень удосконалити задачі обробки та прогнозування споживання газу. Крім того, методи прийняття рішень є також використовується для допомоги операторам у прийнятті обґрунтов них рішень щодо оптимізації газу процеси споживання. Ці методи доводять свою ефективність у моніторингу споживання газу, зменшуючи витрати електроенергії та мінімізуючи вплив на навколишнє середовище [3].

Крім того, нейронні мережі здатні вирішувати проблему забезпечення надійності роботи електропостачання будівлі та аналізу її спроживання. У

науково-технічних роботах дослідження проблеми якості електропостачання та аналіз споживання електроенергії розглядається як дві окремі та незалежні області. При цьому ці сектори взаємопов'язані і однаково важливі для забезпечення надійності функціонування електроспоживання закладів [4].

Нейронні мережі ефективно вирішують завдання класифікації, регресії, кластеризації, асоціації, генерації, перетворення, пошуку, оптимізації та багато інших. Різні архітектури, у вигляді згорткових, рекурентних, графових, капсульних, генеративно-змагальних, уважних, можуть застосовуватись з метою підвищення гнучкості обробки результатів залежно від особливостей даних та характеру завдань.

Нейронні мережі здатні виконати моделювання різноманітних циклічних сигналів та їх обробку за допомогою цифрових систем, що є важливим завданням. Моделювання дозволяє визначити можливості відомих і створених методів обробки циклічних сигналів на різних етапах аналізу сигналів, тестування, наприклад, їх використання модельовані реалізації. Крім того, імітація циклічних сигналів дозволяє навчати нейронні мережі заново [5].

Відомо багато математичних моделей циклічних сигналів, які розроблені в рамках як детерміністичного, так і стохастичного підходів до їх опису [6].

Таким чином, нейронні мережі є потужним інструментом для обробки великих і різноманітних даних, об’єм яких зростає з кожним днем.

Продовжуючи тему факторів актуальності нейронних мереж, варто відзначити продовження розвивитку електронної обчислювальної техніки, яка дозволяє створювати більш потужні та здібні до вирішення задач нейронні мережі. Цей фактор пов’язаний з тим, що для навчання і застосування нейронних мереж потрібно мати достатньо обчислювальних ресурсів, таких як пам’ять, процесори, графічні ядра тощо.

В ході останніх десятиліть обчислювальна техніка досягла значного розвитку, що дало можливість створювати більш складні і ефективні нейронні мережі. Станом на дві тисячі двадцять третій рік, нейронні мережі починають інтегровувати в мобільні процесори. Це обумовлено тим, що, за даними Moore’s Law, кількість транзисторів на інтегральній схемі збільшується вдвоє приблизно

кожні півтора роки, що експоненціально збільшує потужність кишенькових пристроїв.

Історично, графічні процесори (англ. Graphics processing unit, GPU) розроблялись та вдосконалювались з метою обробки графічних об’єктів. Однак, з плином часу, вони виявилися продуктивними для використання у сфері машинного навчання. GPU володіють великою кількістю паралельних ядер, що здатні реалізувати однотипні операції над великими масивами даних. Такий підхід є характерним і для нейронних мереж.

Згідно інформації виробника графічних процесорів, компанії NVIDIA, найпотужніший GPU, випущений компанією на кінець дві тисячі двадцять третього року, NVIDIA A100, має 54 мільярди транзисторів, 6912 ядер CUDA, 40 ГБ пам’яті і може досягати 19,5 терафлопс одинарної точності. Такі потужності дозволяють виконувати процес навчання та запуску дуже складних нейронних мереж, наприклад, GPT-4, що включає в себе більше як 200 мільярдів параметрів, що застосовуються для точної та природної генерації текстової форми мовлення.

Постійна поява нових напрямків застосування нейронних мереж, де вони демонструють свої переваги над традиційними методами вирішення задач, також є важливим фактором актуальності. Нейронні мережі використовуються в таких галузях, як медицина, освіта, економіка, безпека, мистецтво, IT і багатьох інших.

Нейронні мережі здатні виявляти характерні особливості хвороб при їх діагностиці, виконувати аналіз поведінки вірусів, їх структури та зарактеристик для створення ефективних лікувальних засобів, або рекомендувати ліки на основі характеру хвороби та специфічних особливостей організму людини, на кшталт алергічних реакцій. Наприклад, вони можуть виявляти рак шкіри з точністю 95%, що перевищує точність лікарів.

Також, нейронні мережі можуть допомогти в персоналізації навчання, генерації питань та оцінці знань на основі інтересів, когнетивних особливостей кожного учня, темпу вивчення матеріалу.

Глибоке навчання використовується в економічній сфері для вивчення та генерування нових торгових стратегій з метою пошуку найкращої формули максимізації прибутку та адаптації до мінливих умов ринку. В цей час згорткова

модель застосовується нейронними мережами для розпізнавання облич, голосу, підпису, відбитків пальців, що підвищує безпеку автентифікації при отриманні прав доступу до конфіденційної інформації, банківських рахунків, фінансових операцій, гаджетів тощо.

Нейронні мережі застосовуються для виявлення аномалій, загроз, вразливостей, шпигунського ПЗ в складних програмних чи апаратних системах та здатна згенерувати алгоритм вирішеня проблем. Наприклад, використовують власні методи шифрування та дешифрування інформації, захисту даних для запобігання зловмисникам, кіберзлочинності, хакінгу, шпигунству тощо.

Алгоритми глибокого навчання використовуються з метою обробки великих потоків даних, що надходять з від записуючих пристроїв, датчиків, супутників, виявляючи, в процесі, будь-які відхилення від норми, що сможуть бути передвісниками небезпеки. Алгоритми генеративно-змагальних нейронних мереж, в свою чергу, застосовуються для створення синтетичних даних, які можуть бути використані для підвищення рівня безпеки або навпаки – для проведення атак.

Нейронні мережі значною мірою впливають і на соціальну сферу. Так, штучний інтелект може використовувати рекомендаційні системи, які враховують історію, поведінку, інтереси, потреби і побажання кожного користувача, що може зіграти важливу роль в сегментації аудиторії, таргетингу, ретаргетингу, персоналізації, генерації контенту, впливу, лояльності тощо. Основним напрямком використання цієї опції є генерація привабливого для конкретного користувача, переконливого та оригінального рекламного контенту в текстовому, графічному чи звуковому форматі.

Нейронні мережі давно зайняли свою нішу в мистецтві своєю здатністю до генерації графічного контенту: останні два роки мережу Інтернет заполонили згенеровані різними AI-інструментами картини. Найбільшого успіху в цій сфері досягли DALL-E та StableDiffusion, що здатні створювати графічний контент практично без помилок. Крім того, рекурентні та згорткові нейронні мережі дедалі частіше застосовують в музикальному сегменті для створення текстів пісень, нот та повноцінних композицій. Нейронні мережі також здатні до

створення коротких анімаційних роликів, ігор, розповідей, діалогів на будь-яку тему тощо.

Таким чином, на основі поверхневого дослідження методів застосування нейронних мереж в різних сферах життєдіяльності людини, можна зробити проміжний висновок щодо високого рівня актуальності теми дослідження.

Нейронні мережі володіють великим потенціалом для ефективного та продуктивного розв'язання завдань різного типу та ступеню складності, що робить цю технологію однією з найбільш інноваційних та перспективних. Штучний інтелект здатен покращити рівень та якість життя людей, зберегти час на обробку великої кількості чи, навіть, банальний пошук інформації, забезпечити віртуальну та фізичну безпеку, оптимізувати економічні та бізнес-процеси тощо.

1.2 Інтеграція поширених моделей нейронних мереж у секторі безпеки

В результаті виконаного аналізу актуальності нейронних мереж справедливим буде твердження, що ця технологія може бути використана будь-де. Однак, серед всіх сфер виділяються ті, в яких штучний інтелект досягнув великих успіхів, активно розвивається та ефективно використовується людьми. Для більш глибокого дослідження предметної області виконається розгляд найбільш важливих напрямків використання нейронних мереж.

Використання нейронних мереж у секторі безпеки є дуже перспективним і актуальним напрямком, оскільки нейронні мережі здібні до виявлення аномалій, загроз, вразливостей, шпигунського ПЗ в комплексних програмних чи апаратних системах, а також здатні згенерувати алгоритм вирішеня проблем. Як згадувалось раніше, штучний інтелект може використовувати власні методи шифрування та дешифрування інформації, захисту даних для запобігання зловмисникам, кіберзлочинності, хакінгу, шпигунству тощо. В цій задачі нейронних мережам допомагає здатність до обробки великих масивів даних.

В спектрі забезпечення кібербезпеки нейронні мережі можуть використовувати автокодувальники, що допомагають у виявленні аномалій в

патернах поведінки, зовнішніх втручань в системні процеси, внутрішніх вразливостей, шкідливого коду і багато іншого. Згенеровані нейронними мережами синтетичні дані застосовуються для тестування системи на предмет вразливостей, навчання в запобіганні загрозам, побудови захисту системи та здійсненні атак на аналогічні системи.

Найчастіше для забезпечення безпеки комп'ютерних систем та мереж застосовуються згорткові нейронні мережі. Їх основною функцією в системах захисту є аналіз мережевого трафіку, обробка та класифікація пакетів, що передаються мережею, виявлення відхилень в потоках трафіку. Дієвим прикладом такої нейронної мережі є розробка США під назвою DeepArmor. Програмний комплекс на базі згорткових та генеративно-змагальних нейронних мереж здатен до виявлення та блокування різних типів шкідливого ПЗ: вірусів, програм віддаленого стеження та моніторингу, ботнетів, криптомайнерів тощо. Принцип роботи системи кібербезпеки полягає у ізоляції файлів, їх аналізі шляхом розбору структури, сигнатури, поведінки, зв'язків та залежностей, після чого відбувається процес генерації шкідливих синтетичних файлів для самонавчання запобіганню подальшим загрозам даного типу.

Окрім загроз апаратних та програмних, нейронні мережі здатні передбачати та запобігати загрозам реальним. Для прикладу, в сфері контролю систем відеоспостереження використовуються згорткові нейронні мережі, що виконують функції збору інформації з камер, мікрофонів, датчиків руху, інших сенсорів та обробку інформації з метою розпізнавання облич, емоцій, жестів, дій осіб; аналізу полодження предметів, загального плану навколишнього середовища, подій тощо. Такий підхід дає можливість ідентифікувати людей, які перебувають у розшуку, підозрілих поведінкових патернів, що можуть призвести до скоєння злочину незаконних дій, аварій, катастроф.

В противагу згортковим нейронним мережам, які застосовуються, переважно, для обробки вже готових даних, рекурентна модель здатна аналізувати відеопотік у реальному часі, що дозволяє прискорити виявлення змін людської поведінки, аномалій в поведінці навколишнього середовища, навдення

причинно-наслідкових зв’язків подій на основі виявлення залежностей та закономірностей поведінкових патернів [7].

Варті уваги і методи застосування генеративно-змагальних мереж, які працюють в одному потоці з рекурентними та згортковими. Головною особливістю ткого підходу є те, що отримане зображення з камер відеоспостереження може бути покращено в аспектах якості картинки: збільшення роздільної здатності, видалення ефекту «шуму», відновлення пошкоджених фрагментів, приблизне відтворення кольорової палітри на основі сірих тонів чорно-білих записів тощо.

Для прикладу розглядається система відеоспостереження BriefCam, що використовується в США з метою аналізу відеоданих для виявлення скоєних злочинів. BriefCam дозволяє переглядати великі обсяги відеоданих за короткий час, використовуючи технологію VIDEO SYNOPSIS, яка стискує години подій у короткий відеоролик [8].

Крім того, BriefCam дає можливість оперативно виконувати пошук та ідентифікацію людей чи об’єктів за різними характеристиками. Департаменту поліції США вдалось налаштувати систему сповіщень про критичні події в режимі реального часу, основну роль в яких грає рекурентна сторона нейронної мережі, та збору аналітики про отриманий відеопотік.

Програмний продукт BriefCam отримав поширення не лише в правоохоронних органах, але й в транспортній галузі, службі надзвичайних ситуацій та міському плануванні. BriefCam здатен використовувати нейронні мережі для аналізу відео на предмет виявлення аварій, аномалій і інших ситуацій, які вимагають уваги. Це допомагає службі 911 швидко реагувати на критичні події ще до отримання запитів від цивільних осіб.

Варто відзначити, що нейронна мережа виконує обчислення з використанням ядер GPU, що прискорює продуктивність обробки відеопотоків та точність результату.

На противагу рекурентним моделям, тандем згорткової з генеративно-змагальнох моделей нейронних мереж, окрім вищезгаданих можливостей, також здатні до розпізнавання людської ДНК чи сітківки ока. Це значною мірою

підвищує ймовірність вірної ідентифікації особи, забезпечує вищий рівень захисту аутентифікації при спробі отримання доступу до захищених даних, апаратури, приміщень тощо [9].

Прикладом використання біометричної аутентифікації на основі нейронних мереж є Face++. Розроблена в КНР, згорткова нейронна мережа займається деталізованим аналізом зображень та відео:

- розпізнавання облич та їх виділення;
- висвітлення характерних рис облич та формування характеристики людини, наприклад, вік особи, її стать, расова приналежність, емоційний фон, рівень симпатії, колір та довжина волосся, наявність додаткової атрибутики у вигляді прикрас на голові та шкірі;
- визначення рівня схожості людей між собою в прямому порівнянні портретів та виявлення людей з найбільшим ступенем подібності;
- виділення ключових точок на обличчі, таких як розміщення вух, очей, брів, носу, губ, підборіддя, волосся тощо;
- аналіз позиції рук в просторі та жестикуляції.

На відміну від всіх попередніх наведених програмних комплексів, розробниками Face++ представлено доступні API і SDK для розробників і користувачів, які дозволяють використовувати її можливості в різних сценаріях і платформах. Завдяки цьому система використовується у великій кількості галузей: фінансова, соціальна, освітня, мадична, розважальна та інші [8].

Хоч закритість американських розробок є недоліком, він значною мірою компенсується ефективністю цих систем. Так, ще з дві тисячі дев’ятого року департамент поліції США почав дослідження та тестування нейронних мереж для прогнозування та попередження злочинності. В перехідних період між нульовими та десятими роками приватна американська компанія «Palantir Technologies», відома своєю плідною співпрацею з Центральним розвідувальним управлінням та урядом США, завершила роботу над потужним програмним засобом під кодовою назвою «Palantir». Основна мета розробки, що знаходилась під прикриттям стартапу, була в створенні системи прогнозування поширення злочинності.

Компанія у співпраці з владою Нового Орлеану, до якої увійшли вузьке коло урядовців, включно з мером Мітчем Ландре, таємно організувала застосування власної розробки прогнозування правопорушень в реальних міських умовах. Секретна програма, на базі рекурентної нейронної мережі, виконувала функції пошуку та дослідження зв'язків між членами кримінальних угрупувань, аналізу профілів користувачві у соціальних мережаж з метою прогнозування ймовірності скоєння злочинів чи ймовірності постати в ролі жертви певними особами.

Основний операційний функціонал цієї програми був спрямований на візуалізацію великих масивів інформації, що допомагає працівникам правоохоронних органів встановлювати причино-наслідковий зв'язок між поведінкою осіб та скоєними ними правопорушеннями.

Після значної кількості успішних тестів та внесення в програму певних корективів, в дві тисячі дванадцятому році програмний засіб був переданий Новому Орлеану у вигляді неофіційної благодійної допомоги. За інформацією з відкритих джерел, більшість урядовців міста підтвердили, що не були обізнані у використанні цього проекту до офіційного висвітлення [7].

Окрім «Palantir», в США існує аналогічна система під назвою «Готем». Вона використовується поліцейським департаментом США для виявлення потенційних злочинців та відстежування їх діяльності. Працює програмний засіб згідно наступного принципу: протоколи затримань та матеріали кримінальних справ формуються в єдину базу даних, яка виконує обробку інформації та створює перелік осіб, що мають чи можуть мати відношення до представників криміналітету.

Не дивлячись на стрімке поширення використання нейронних мереж в секторі безпеки та широкий спектр застосування, працівники департаменту поліції, програмні інженери та дослідники застерігають, що нейронні мережі не можуть стати вирішенням всіх проблем, наприклад, причин та передумов скоєння злочинів. Однак, ніхто не заперечує факту, що використання нейронних мереж зробило роботу правоохоронців швидшою, якіснішою та дешевшою відносно традиційних методів стеження, прогнозування тощо.

1.3 Аналіз результатів прогнозування із застосуванням нейронних мереж

В економічній сфері лідерство у застосуванні нейронних мереж також утримує США. Найбільш поширені мережі банків у Штатах застосовують обчислювальні потужності нейронних мереж з метою управління кредитними ризиками, наприклад, для оцінки ймовірності повернення кредиту позичальником.

Припустімо, особа претендує на кредит з умовною сумою. Тоді нейронні мережі витягують кредитну історію людини, відомості про доходи, джерела доходів та стабільність їх отримання, за необхідності, виконують оцінку майна. Після збору інформації про особу штучний інелект виконує побудову графіків ймовірності несвоєчасного повернення коштів у перспективі виділеного часу, або його конкретних відрізків.

На основі цього нейронні мережі оцінюють можливі збитки, які понесе банк, надавши кредит. Результат всіх операцій обробки даних та обчислень фінансових факторів перетворюється в структуру, що називається фінансовою надійністю позичальника. Таким чином, технологія здатна швидко попередньо визначити ненадійних позичальників та зібрати підстави для відмови у кредитуванні.

Окрім банківського сектору нейронні мережі вже активно застосовуються у прогнозуванні змін на фондовому ринку.

Наприклад, Anfis Editor та nnTool – інструменти Matlab, призначені для тонкого налаштування, навчання, тренування та експлуатації нейронної мережі під назвою «Елмана» в середовищі додатку – зарекомендували себе при створенні прогнозів ситуації на міжбанківському валютному ринку Forex, моделюванні індексу фондової біржі першої фондової торгівельної системи – ПФТС.

Результат виконання однієї з вищеописаних операцій з допомогою нейронних мереж подано на рисунку 1.1.

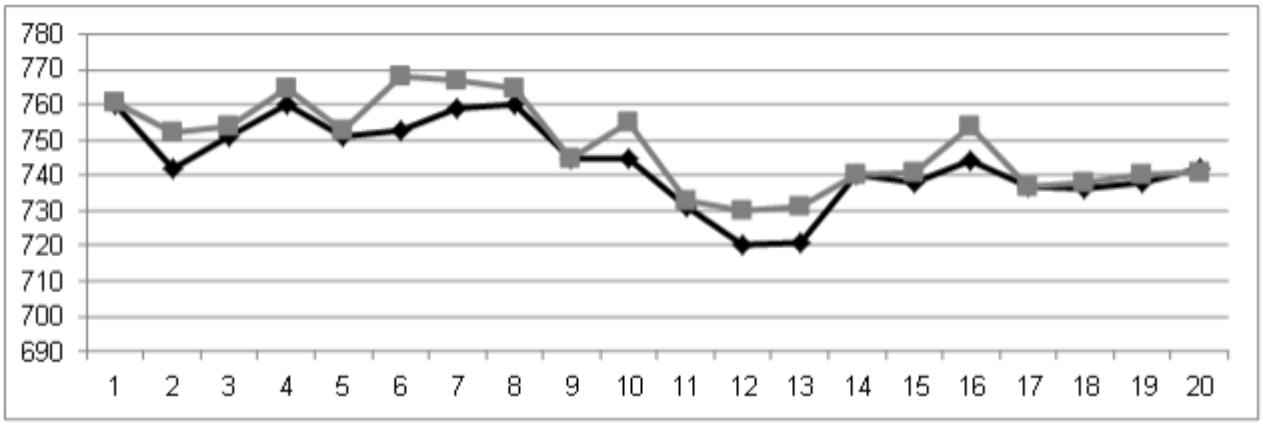

Рисунок 1.1 – Результат моделювання індексу фондової біржі [10]

Згідно результатів моделювання, виконаних нейронною мережею (сірий колір відрізків), індекс фондової біржі ПФТС протягом більшості днів періоду з першого січня дві тисячі п’ятнадцятого року по двадцяте січня того ж року мав бути вищим за фактичний. Тим не менше, похибка виявилась не значною в порівнянні з фактичними даними (чорний колір відрізків). З двадцяти точок, присутніх на графіку, прогноз нейронної мережі збігся з реальною ситуацією в одинадцяти, що станом на дві тисячі п’ятнадцятий рік вважалось успішним результатом.

Інший програмний продукт – BrainMaker – був виготовлений на замовлення Пентагону в якості нейропакету військового призначення [11].

Принцип роботи нейропакету був заснований на поширеній сьогодні моделі імітації роботи біологічних нейронів, з’єднаних між собою в багатошарову структуру. Кожен окремий нейрон мережі мав власні входи та вихід. Вхід застосовувався для отримання інформації від інших нейронів, якщо поточний нейрон знаходився в одному з внутрішніх шарів, або із зовнішнього джерела, якщо нейрон знаходився на початку мережі.

Будь-які вхідні дані отримували своє значення цінності – вагу – для поточного нейрона. Далі нейрон виконував процес розрахунку суми добутків отриманих даних та їх цінності. На основі отриманого результату визначався подальший маршрут даних в мережі (нейрон-отримувач). Тоді вихід нейрону використовувався для передачі інформації іншим нейронам наступних шарів, або вихідному шару.

На основі системи ваг і відбувалось навчання нейронної мережі: на вхід нейронів представлялись вхідні дані з вже готовим вихідним результатом. Завданням нейрону було виконувати регулювання ваг цих даних до мінімізації можливості здійснити хибне судження чи досягнення мінімальної різниці між фактичним та очікуваним результатом.

Ця складна розробка, як і, свого часу, Інтернет-мережа чи супутниковий зв'язок, швидко перестала бути власністю виключно американського уряду та набула поширення в інших сферах.

Так, ще в одна тисяча дев'ятсот дев'яностому році пакет був переформатований для бізнес-орієнтованого програмного забезпечення та почав виконувати функції аналізу фондових ринків, прогнозування падіння чи зростання цін на акції, нерухомість тощо. Сьогодні програмний засіб є одним з найбільш популярних в фінансовій сфері США та був удостоєний безлічі нагород за свою інноваційність.

Одним із прикладів застосування нейропакету є трейдерське програмне забезпечення NeuroShell Day Trader. Ця програмна платформа, розроблена для створення та розгортання торгових систем, володіє розширеним функціоналом для тонкого налаштування кожної моделі нейронної мережі, вибору даних та параметрів для її навчання, тренування та експлуатації в цілях автоматизації, аналізу вартості акцій, валютних курсів, індексів тощо.

Всі результати виконаних завдань ПЗ демонструвало у вигляді графіків чи таблиць з підкріпленими до них даними статистики. Маючи свій власний графічний інтерфейс, програмний засіб також міг бути інтегрованим з іншими табличними ПЗ, наприклад, MS Excel.

Як правило, торгові платформи, такі як NeuroShell, можуть пропонувати певні можливості інтеграції, але ступінь і особливості можуть відрізнятися. Наприклад, можливість експортувати дані з NeuroShell для подальшого аналізу в інших програмах або імпортувати зовнішні дані на платформу.

Демонстрацію застосування NeuroShell Day Trader з метою визначення волатильності однієї з фондових бірж подано на рисунку 1.2.

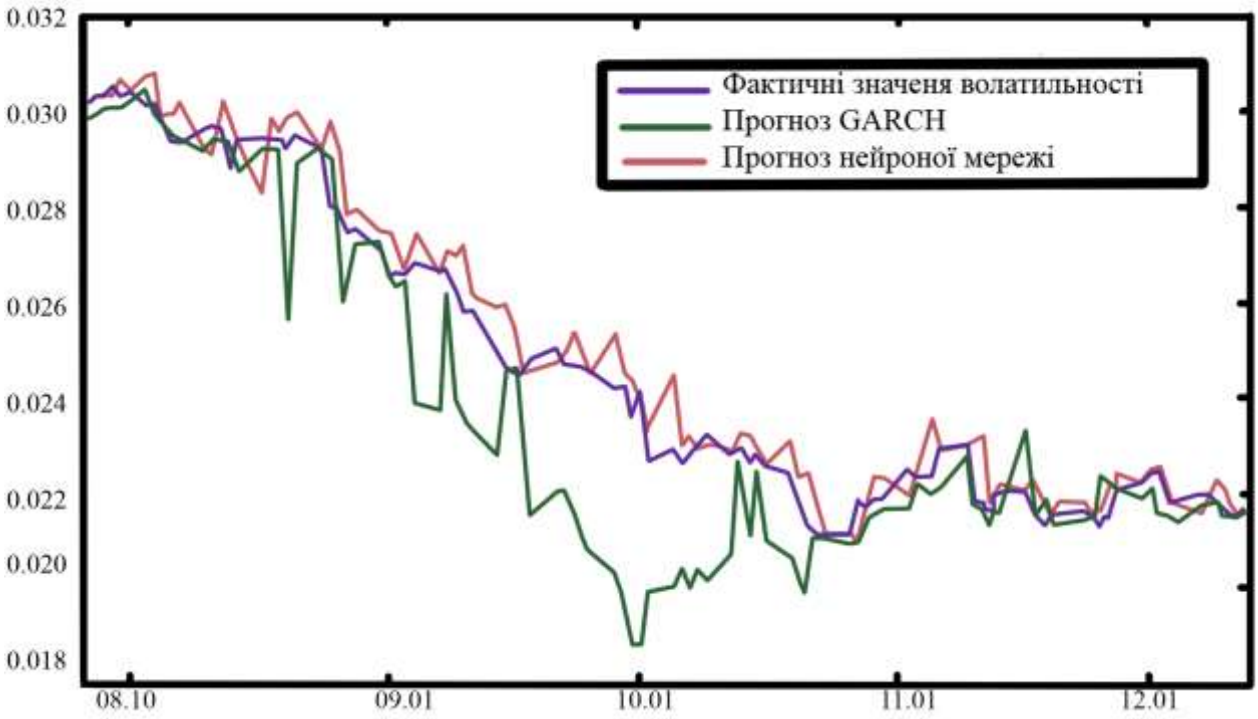

Рисунок 1.2 – Прогнозування ситуації на фондовому ринку [10]

Результат порівняння прогнозів нейронної мережі, моделі прогнозування GARCH та реальної ситуації свідчить про набагато вищу стабільність прогнозу саме зі сторони нейронної мережі, хоча в більшості випадків остання завищувала значення індексу. Тим не менше, похибка нейропакету була не настільки суттєвою в порівнянні з моделю GARCH, яка навпаки – часто і відчутно занижувала свої значення в своєму прогнозі.

Економічна сфера не обмежується лише банківськими операціями та фондовими ринками. Вона охоплює всі сфери, які впливають на фінансові потоки. Так, системи масового обслуговування широко застосовують нейронні мережі з метою оптимізації торгівлі, транспортних потоків, роботи з клієнтами тощо. Наприклад, рекурентні нейронні мережі виконують завдання обробки та сортування заявок користувачів на придбання товару, здійснюють перевірку виконання великої кількості транзакцій на предмет їх успішності, збирають аналітичні дані використання систем обслуговування та формують оцінку їх ефективності, продуктивності тощо.

Згорткові нейронні мережі, в свою чергу, застосовуються для виконання аналізу зібраних даних про роботу системи в різні періоди часу та їх порівняння в контексті виявлення закономірностей, причинно-наслідкових зв’язків появи

тих чи інших подій, формування подальших тенденцій та прогнозів, що можуть бути використаними для планування та реалізації подальших кроків оптимізації, модифікації, інших покращень системи [12].

В системах масового обслуговування, що застосовуються в торгівельній галузі, нейронні мережі використовують свої алгоритми аналізу поведінки користувачів для формування рекомендацій, стратегії ціноутворення, рекламних кампаній, підвищення привабливості товару.

Наприклад, компанія Amazon впровадила в своїй веб-системі обслуговування клієнтів рекурентні та генеративно-змагальні моделі нейронних мереж для моніторингу історії покупок, переглядів і пошукових запитів кожного користувача з цілюю виявлення основних типів товарів, які найчастіше він замовляє, об’єднання користувачів в окремі групи за інтересами, що пришвидшує поширення кращих рекомендацій та релевантних результатів пошуку. Вони також аналізують відгуки про товар з метою виділення найкращих, якісних чи популярних об’єктів покупки.

Крім того, Amazon використовує нейронні мережі для налагодження логістики переміщення товарів, в тому числі й в приміщеннях їх зберігання та підготовки до транспортування. Хоч станом на дві тисячі двадцять третій рік роботи не є поширеним явищем та все ще вважаються інноваційною технологією, вони вже давно виконують свої фукнкції в зв’язку зі штучним інтелектом, забезпечуючи продуктивність в задачах, переміщення сортування та збереження товарів. За інформацією з вільних аналітичних джерел, використання роботів на складах Amazon скоротило витрати часу на завдання, в порівнянні з їх виконанням людиною, в середньому на 20-30%.

Приклад пристроїв, що вже виконують свої завдання під управлінням нейронних мереж на складі Amazon, зображені на рисунку 1.3.

Представлені приклади застосування нейронних мереж є лише невеликою частиною з всіх можливих варіантів використання цієї технології в економічній сфері. Тим не менше, їх більш ніж достатньо для демонстрації рівня інтеграції штучного інтелекту в однин з найважливіших секторів діяльності людини.

Рисунок 1.3 – Пристрої під управлінням нейронних мереж Amazon

Перейдемо до питання тенденцій поширення нейронних мереж в різних сферах людської діяльності.

1.4 Дослідження тенденцій поширення нейронних мереж в сфері інформаційних технологій та в інших сферах

Галузь IT значною мірою переплітається з іншими сферами діяльності людини. Наприклад, функціонування вищезгаданих систем масового обслуговування буде значною мірою ускладнене, якщо взагалі можливе, без засобів комунікації. Нейронні мережі широко застосовуються в проектуванні комп’ютерних мереж та мереж зв’язку. Основними завданнями штучного інтелекту в цьому напрямку є оптимізація прокладених маршрутів передачі даних, контроль за маршрутизацією потоків інформації, аналізу пакетів, що передаються мережею, на предмет пошкоджених фрагментів, шкідливого ПЗ.

Іншою сферою, де програмна продукція IT в зв’язці з нейронними мережами часто залучаються, є транспортна галузь. Станом на дві тисячі двадцять третій рік нейронні мережі рухаються в напрямку гравця ключової ролі у побудові транспортних шляхів та інфраструктури. Своєю здатністю до швидкої обробки Big Data нейронні мережі забезпечують короткі терміни аналізу всіх факторів проекту: витрат на робочу силу, матеріали, цикл та часові проміжки виконання, ризики, тенденції тощо. По завершенню аналізу результати можуть бути візуалізовані у будь-якому зручному вигляді.

Відомим прикладом застосування нейронних мереж для розрахунку тенденцій витрат на будівництво в довгостроковій перспективі є зведення нової автомагістралі в штаті Луїзіана, США. Протягом 17 років, за які планувалось виконати її будівництво, нейронні мережі передбачили подвоєння витрат відносно початково закладеної урядом суми. Таким чином, замовник робіт в уособленні влади Луїзіани був підготовлений до ймовірного розвитку подій та зміг створити резервний фонд, що, згодом, дозволить знизити вплив здорожчання робіт підрядника на бюджет штату.

Окрім того, доцільно використовувати нейронні мережі у виборі оптимальних маршрутів перевезення вантажів та пасажирів територією України в контексті входження у транс-європейські коридори та напрямки «Шовкового шляху». На даний момент потужності нейронних мереж можна використовувати з метою швидкого перенаправлення торгівельних потоків іншими внутрішніми та міждержавними напрямками, наприклад, для вирішення питання систематичного перекриття кордону зі сторони Польщі.

Нейронні мережі дають можливість виконувати короткострокове прогнозування транспортних потоків та віртуальне моделювання магістралей на основі макродинамічної моделі транспортного потоку. Нейронні мережі можуть аналізувати історичні дані трафіку, беручи до уваги різні фактори, такі як погода, час доби та особливі події, щоб прогнозувати затори. Потім ці прогнози можна використовувати для планування альтернативних маршрутів, щоб уникнути інтенсивного руху.

Прикладом застосування нейронними мережами вищеописаних можливостей є компанія UPS. Її розробка під назвою ORION (On-Road Integrated Optimization and Navigation) здатна, на основі макродинамічної моделі транспортного потоку, надавати водіям актуальні дані про ситуацію на визначеному дорожньому маршруті та пропонувати найбільш оптимальні шляхи руху. Згідно статистичних даних, зібраних самим розробником системи, ORION дає можливість скоротити довжину маршрутів на п’ятдесят мільйонів годин робочого часу в рік, тим самим заощадивши від дев’яноста до ста десяти

мільйонів галонів палива, що еквівалентно рівню викидів вуглекислого газу в атмосферу мінімум в дев’яносто тисяч тонн.

Іншим програмним продуктом, що демонструє можливості нейронних мереж в напрямку налагодження логістичних маршрутів є система масового обслуговування Google Maps. Для підтримки якісної роботи своєї розробки компанія застосовує зв’язки із супутниками, геолокаційні дані положення смартфонів в просторі, дані мережевого трафіку, доступні камери відеонагляду, датчики руху, Інтернет-ресурси та інші джерела передачі інформації. Після обробки цього масиву даних нейронні мережі Google відображають в додатку актуальну інформацію про погодні умови, стан дороги тощо, та надають рекомендації щодо альтернативних маршрутів руху транспорту.

Приклад виконання процесу побудови просторових даних трафіку району аеропорту міста Ріад, Саудівська Аравія, подано на рисунку 1.4.

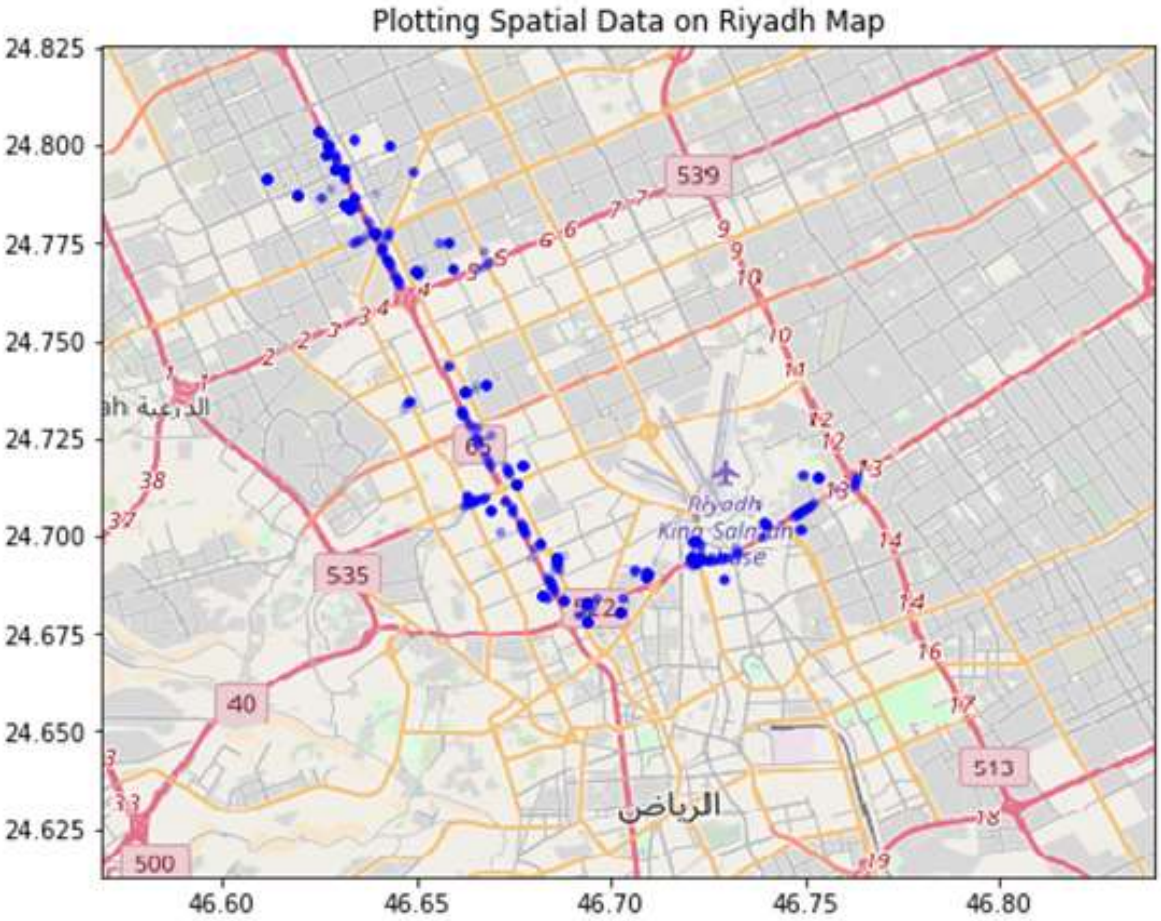

Рисунок 1.4 – Графік потоку трафіку

Для звичайних користувачів такий підхід дає можливість оминути небезпечні чи складнопрохідні ділянки визначеного маршруту, незалежно від типу транспорту, яким вони користуються.

Для транспортних чи торгівельних корпорацій, в свою чергу, застосування додатку на базі нейронних мереж дозволило прискорити доставку товарів. За інформацією численних незалежних аналітиків витрати часу та пального за умови використання оптимальних маршрутів Google Maps скорочуються, в середньому, до відмітки в двадцять та двадцять п’ять відсотків, відповідно.

Також, Google використовує нейронні мережі для своєї системи Google Duplex, яка здатна вести природну розмову з людьми по телефону в режимі реального часу, виконуючи завдання бронювання квитків на різні події, транспорт, місця тимчасового перебування, замовлення доставки їжі, продуктів чи інших товарів, запису на відвідування барбершопу, перукарні тощо [13].

На цьому можливості нейронних мереж не завершуються, оскільки вони можуть бути залучені до розробки інших програмних засобів в якості помічників, допомагаючи на різних етапах виконання завдань. Станом на дві тисячі двадцять третій рік штучний інтелект активно застосовується в якості IDE-плагінів, що спияють пришвидшенню написання програмного коду, створюючи окремі його фрагменти, аналізуючи помилки та виконуючи відлагодження. Також, додаткове ПЗ на базі нейронних мереж може аналізувати вимоги до проекту, створювати список завдань та документацію.

Крім того, нейронні мережі здатні до аналізу багатьох аспектів та атрибутів проекту: сематика, синтаксис, стиль, контекст, залежності тощо. Прикладом такої нейронної мережі може бути застосунок DeepCode. Він широко застосовується для аналізу коду на предмет наявності в ньому помилок, вразливостей, недоліків та здатен самостійно виправити їх або запропонувати варіанти вирішення допущеної проблеми.

Для успішності технологічного процесу розробки програмного забезпечення (ПЗ) важливим є вирішення завдань підтримки якості як програмного коду зокрема, так і програмного продукту в цілому, а також постачання чергових версій програмного продукту до споживача [14].

Найбільшого поширення нейронні мережі досягли в сфері мистецтва. Можливості генерації нового контенту створили велику кількість програмного забезпечення з різними функціями.

На основі існуючих стилів та жанрів, тематик та авторів такі програмні продукти, як DELL-E, здатні створювати зображення високої якості. Нейронна мережа, при цьому, вимагає лише текстовий опис рисунку, використовуючи ключові слова з нього для генерації частин зображення.

Сучасний метод створення зображень застосовує генеративно-змагальні нейронні мережі, оскільки вони взаємодіють між собою з метою якісного виконання завдання: нейронна мережа виконує завдання створення зображення, а дескримінатор порівнює отриманий результат з вхідним зображенням на предмет збігу. В залежності від цілі завдання та результату, дескримінатор повертає відповідь.

Крім того, нейронні мережі здатні допрацьовувати вже створені витвори мистецтва, аналізуючи та повторюючи стиль художника. Наприклад, в якості реставраторів старих картин з метою їх відновлення чи вдосконалення. Аналіз картин нейронною мережею може бути використаний для встановлення автора невідомих витворів мистецтва на основі стилю та характерних елементів, для подальшої модифікації об’єкту аналізу.

Одним з основних типів нейронних мереж, що використовуються в сфері мистецтва, є згорткові нейронні мережі. Він може виявляти об’єкти на різних рівнях абстракції та зберігати просторову інформацію, що робить його придатним для обробки зображень.

Згорткові нейронні мережі здатні виконувати завдання аналізу та класифікації зображення. Аналіз включає в себе наступні фактори: атрибути зображення у вигляді емоційної палітри об’єктів та характерних елементів стилю, жанр, автора. Наприклад, чи є зображення портретом, пейзажем, натюрмортом, абстракцією, приналежністю до імпресіонізму, експресіонізму, сюрреалізму і т.д. До методів аналізу та класифікації зображень відноситься згорткова нейронна мережа, що привертає увагу. Такий метод застосовує механіку фокусування уваги для виділення ключових частин зображення.

Згорткові нейронні мережі можуть застосовуватись в мистецтві з разними задачами. Наприклад, щоб надати зображенню стиль, транспортувати стиль одного зображення безпосередньо на інше, зберігаючи вміст. Припустімо, будь-який користувач може створити зображення, стиль якого буде повторювати картини Ван Гога або Пікассо. Алгоритм використовує згорткові нейронні мережі з метою аналізу вмісту та стилю зображення, оптимізуючи нове зображення, щоб мінімізувати різницю між ними. Цей алгоритм є відносно новим, оскільки був запропонований Леоном Гатіссом у дві тисячі п’ятнадцятому році.

Іншим типом нейронних мереж, які використовуються в мистецтві, однак не художньому, є рекурентні нейронні мережі. Вона представляє себе як мережа зворотного зв’язку, що може зберігати попередній стан та виконувати обробку послідовних даних, таких як текст або медіа-файли. В мистецтві присутні сценарії використання таких нейронних мереж, однак, вони не є поширеними.

Перш за все, нейронні мережі виконують завдання генерації змістовного та граматично вірного тексту. Наприклад, дають можливість згенерувати текст з імітацією стилю автора конкретного твору, жанру, тематики. Вони здатні створювати різні типи літературних творів, в тому числі: пісні, вірші, оповідання, романи. Також, нейронні мережі здатні створити сценарій твору, відео або повноцінного фільму. Основним методом, що застосовується для виконання вище описаних завдань є моделі мови, що виконують завдання вивчення ймовірності наступного слова в тексті на основі попередніх.

Окрім тексту, нейронні мережі здатні відтворити звуки інструментів, або створити нові; здатні до генерації голосів, звукових ефектів, коротких мелодій, повноцінних композицій та пісень. Метод синтезу, який часто застосовується для виконання таких завдань, є глибокою нейронною мережею з умвідомленням часу. Вони застосовуються рекурентними нейронними мережами для відтворення з’язків та залежностей між звуковими сигналами.

В галузі медицини нейронні мережі, переважно, застосовуються для діагностики захворювань [15].

Прикладом використання такого типу нейронної мережі може бути програмний пакет від компанії R Informati, що виконує завдання діагностики показників серця на наявність відхилень. Такого роду системи успішно застосовуються в більшості медичних закладів Англії з метою попередження інфаркту міокарда та інших серцево-судинних захворювань, знижуючи їх рівень [16].

Використання нейронних мереж впроваджують навіть на рівні державних інституцій з метою управління, контролю та аналізу. Наприклад, у Китайській народній республіці нейронні мережі застосовуються для боротьби з корупцією. В обмеженому колі регіонів КНР з дві тисячі дванадцятого року працює національна система безпеки, яка виконує завдання моніторингу за дотриманням правопорядку із застосування функції ідентифікації громадян за посередництва моделей нейронних мереж для аналізу та розпізнавання обличчя. Аналогічні технології були впроваджені для контролю за виконанням працівниками правоохоронних органів своїх обов’язків за нормами правового поля та виявлення випадків перевищення службових повноважень. Також, нейронні мережі офіційно впроваджені в системи захисту державних баз даних від зовнішнього проникнення.

В тому ж дві тисячі дванадцятому році в КНР запровадили систему нульової довіри «Zero Trust». Розроблена на базі нейронних мереж Китайською академією наук вона виконує функції внутрішнього контролю державних службовців на базі аналізу інформації з більш ніж ста п’ятидесяти захищених баз даних та оцінки якості виконання службових обов’язків, процесів особистого життя, загальної поведінки. Таким чином, нейронна мережа здатна виявити підозрілі операції набуття чи відчуження власності, факти незаконного будівництва, придбання землі або знесення будинків, незаконного збагачення з використанням тіньових схем [7].

Проаналізовані типи нейронних мереж та методи їх застосування займають лише незначну частину загального спектру варіантів використання в різних сферах. Окрім вищепредставлених існують автокодери, меморіальні мережі, трансформери тощо. Тим не менше, представлених прикладів цілком достатньо

для демонстрації глибини проникнення нейронних мереж у види діяльності людини.

1.5 Аналіз ризиків впливу нейронних мереж на життєдіяльність людини

Частина досліджень на тему нейронних мереж висловлює певний песимізм щодо зростання безробіття через втручання нової технології у всі процеси життєдіяльності. Найвідоміше з них виконали Фрей та Осборн в Оксфордському університеті десять років назад. Дослідження стверджує, що автоматизація робочих місць може до дві тисячі тридцять третього року позбавити працездатне населення США близько половини видів діяльності – сорока семи відсотків. Ймовірність цього, згідно розрахунків авторів, перевищує сімдесят відсотків.

Подібна ситуація була спрогнозована і для Німеччини – сорок два відсотки видів діяльності. Автори дослідження припускають, що робота на основі частково чи повністю повторюваних завданнях з необхідністю аналізу великої кількості даних когнітивно вимогливих видів діяльності в зоні найбільшого ризику. Подібні прогнози були озвучені й іншими дослідженнями.

Прогнози Фрей та Осборн подають оцінку саме на тему видів діяльності, які може повністю взяти на себе штучний інтелект, але не в контексті кількості робочих місць. Тим не менше, ці оцінки ставляться під сумнів еволюцією професій ще з часів дев’ятнадцятого століття. Для прикладу, ще до цифрової трансформації світу та створення штучного інтелекту професія сажотруса значною мірою адаптувала інструменти виконання завдань, однак, продовжує виконуватись людиною з більшою швидкістю, якістю та контролем. Відповідно до цієї думки, всі сучасні дослідження спростовують результати аналізу Оксфордського університету. Наприклад, Федеральне міністерство праці та соціальних справ опублікувало статистику, згідно якої дев’ять відсотків робочих місць у США під загрозою повної автоматизації.

Відповідно до іншого результату аналізу майбутніх перспектив робочих місць Всесвітнього економічного форуму за дві тисячі двадцятий рік, станом на

дві тисячі двадцять п’ятий рік близько дев’яноста мільйонів робочих місць буде передано у виконання сучасним технологіям. Однак, наявність та впровадження цих технологій дозволить створити до ста мільйонів робочих місць, що свідчить про хибність теорії безробіття через технологічний прогрес. Думка про завищення показників безробіття через технології станом на дві тисячі двадцять третій рік стала більш аргументованою та апелює до рівності нових посад з потенційними робочими місцями чи позитивних прогнозів збільшення можливостей працевлаштування, хоч останні і не враховують когнитивні вимоги нових робочих місць та поділу потенційних робітників на класи [17].

Прогнозується, що до дві тисячі тридцятого року якість програмних продуктів та рівень використання такого роду технологій значно зросте, а їх залучення розширить можливості та галузі застосування FP&A в корпоративному управлінні. Базовим елементом в цьому процесі стане централізована база даних, що об’єднує актуальні та постійно оновлювані дані з зовнішніх та внутрішніх ресурсів. В реальному часі адміністратори процесів, фінансові працівники, планувальники та аналітики отримують цілісне уявлення про поточний стан компанії, а нейронні мережі зможуть виконати глибокий аналіз даних, сформувати її подальші перспективи та бізнес-стратегії [17].

Тим не менше, станом на дві тисячі двадцять третій рік впровадження нейронних мереж в індустріальній сфері залишається на початковому етапі нової виробничої парадигми на основі даних та алгоритмів [17].

До чинників, на які впливають нейронні мережі, окрім економічних також входять соціальні та психологічні. Станом на дві тисячі двадцять третій рік, все частіше люди вдаються до спілкування з нейронними мережами, такими як ChatGPT, використовуючи цей винахід для спрощення виконання своїх робочих завдань чи досягнення особистих цілей. Для прикладу, одним із найпоширеніших способів використання учнями ChatGPT є домашнє завдання. ChatGPT допомагає учням з різних предметів, таких як математика, фізика, хімія, біологія, історія, географія, література тощо. ChatGPT може надавати студентам поради, пояснення, приклади, ресурси, огляди, відгуки тощо [18].

З іншого боку, це впливає на соціальну поведінку людини. Особи, що користуються соціальними мережами, такими як Twitter, Facebook, Instagram можуть піддаватись впливу нейронних мереж в ході соціальної взаємодії з іншими людьми. Прикладами можуть слугувати прохання: «Придумай жарт», «Напиши вірш», «Згенеруй картинку», «Виправ фото» і т.д. Особливо це стосується розробки Microsoft – Bing – нейронна мережа якої, під час взаємодії з користувачем, сама вносить такого роду пропозиції.

Соціальні мережі, в свою чергу, активно застосовують свої бази даних користувачів для машинного навчання з метою опанування нейронними мережами здатності до розуміння ключових атрибутів особи: інтересів, настрою, емоцій, поведінки, впливу, взаємодії. Постійний аналіз цієї інформації використовується для подальшого генерування релевантних рекламних пропозицій та новин, представлення персоналізованих сервісів.

Звісно, такий підхід має свої ризики, оскільки ставить під сумнів конфіденційність та безпеку даних кожного користувача. Приватність та свободи можуть бути порушені зловмисниками, що можуть використовувати технологію з метою викрадення даних, шпигунства, шантажу, маніпуляцій, дезінформації, кібератак. Нейронні мережі здатні порушити встановлені у світі права, свободи та цінності, оскільки базово вони не мають розуміння або не враховують інтереси людини, потреби, бажання, етику, мораль.

Відповідно до цього, нейронні мережі здатні по-різному впливати на психічне та емоційне здоров’я людини, самопочуття, її задоволення життям, розуміння сенсу існування, емпатію. Наприклад, легкодоступні відповіді можуть знижувати зусилля для виконання певних завдань та погіршувати концентрацію на виконанні задачі, негативно відобразитись на здатності запам’ятовувати інформацію та створювати залежність від технології, знижувати рівень активності когнетивних процесів та перенести людину в стан фрустрації, погіршити логіку та критичне мислення. За таких умов може постраждати і якість знань, відповідно – фаховість спеціалістів в різних напрямках.

Застосування нейронних мереж в творчих процесах, загалом, знецінює працю справжніх митців, письменників, музикантів. При виконанні важливих

завдань «цифровий помічник» може створювати тривогу та стрес через невідповідність відповіді поставленому завданню, перенавантаження користувача інформацією, не виправдані очікування, втрату часу на очікування відповіді, критику, неможливість пояснити своє бажання і т.д.

Разом з тим, нейронні мережі здатні позитивно відобразитись на розвитку інтелекту та його окремих аспектах, підвищуючи, підтримуючи та покращуючи їх. Технологія представляє різні можливості для навчання, покращення когнетивних функцій, саморозвитку, виконання досліджень та пошуку інноваційних рішень.

У висновку, нейронні мережі можуть мати позитивний або негативний вплив на людський інтелект, залежно від того, як вони використовуються, контролюються, регулюються і оцінюються.

1.6 Висновок до першого розділу

В першому розділі кваліфікаційної роботи виконано дослідження предметної області визначеної тематики в контексті розкриття актуальності розробки нейронних мереж та тенденцій їх розвитку.

Детально розглянуто сфери застосування нейронних мереж та представлено приклади успішного вирішення поставлених перед ними задач. Розкрито принцип роботи нейронних мереж в кожному з представлених напрямків застосування технології.

Розглянуто вплив на соціальне становище людини в контексті працевлаштування. Виконано аналіз впливу нейронних мереж на життєдіяльність людини, її ментальність, висвітлено ризики застосування технології, переваги та недоліки.

2 АНАЛІЗ МЕТОДІВ СТВОРЕННЯ НЕЙРОННИХ МЕРЕЖ

2.1 Обґрунтування вибору платформи JavaScript

Однією з причин використання JavaScript для розробки нейронних мереж є те, що JavaScript – це універсальна, легка і загальна мова програмування, яку можна запускати як на стороні клієнта, так і на стороні сервера [19]. Це означає, що нейронні мережі можна створювати для запуску в браузері або Node.js і використовувати для різних цілей, включаючи веб-розробку, мобільну розробку, розробку ігор, Інтернет речей і машинне навчання [20].

JavaScript також є мовою програмування високого рівня з динамічною типізацією, успадкуванням прототипів і функціональним програмуванням. Фактично, вона базується на засадах логіки та алгоритмів на заміну складних конструкцій типізації, що робить її гнучкою, виразною і легкою у вивченні та використанні. Один і той самий код на JavaScript можна використовувати на різних платформах, що полегшує розробку і тестування.

Ще одна причина використовувати JavaScript для розробки нейронних мереж полягає в тому, що існує багато високоякісних бібліотек і фреймворків, які надають потужні та практичні інструменти для створення та навчання нейронних мереж.

Brain.js – це вбудована в браузер бібліотека нейронних мереж Node.js, яка підтримує різні типи мереж, такі як повністю зв'язані, рекурентні, LSTM і GRU, а також різні функції активації, оптимізації, втрати та інші [21]. Вона також використовує графічні процесори для прискорення обчислень і надає простий та інтуїтивно зрозумілий синтаксис для створення та навчання мереж.

Synaptic.js – це браузерна бібліотека нейронних мереж, яка підтримує різні типи мереж, включаючи повністю зв’язані, рекурентні, LSTM, GRU та HopField, а також різні функції активації, оптимізації, втрати тощо. Вона також надає незалежні від бібліотеки архітектури та алгоритми і гнучкий модульний синтаксис для побудови та навчання мереж.

TensorFlow.js – фреймворк для створення нейронних мереж у браузері та Node.js, заснований на TensorFlow, одному з найпопулярніших і найпотужніших фреймворків для машинного навчання. Графічні процесори та WebGL можна використовувати для прискорення обчислень і передачі моделей з Python на JavaScript і навпаки. Він також надає високорівневі та низькорівневі API для побудови та навчання мереж, а також попередньо підготовлені моделі для різних завдань, таких як класифікація зображень, розпізнавання мовлення та генерація тексту.

Відповідно до можливостей бібліотек, JavaScript підтримує рекурентні та згорткові нейронні мережі. Він дуже потужний і ефективний для обробки послідовних і просторових даних, таких як текст, аудіо, зображення та відео. Цей тип нейронної мережі можна використовувати для вирішення складних і творчих завдань, таких як класифікація, розпізнавання, генерація, трансляція та синтез.

Іншою причиною використання JavaScript для розробки нейронних мереж є те, що це інтерактивна, експериментальна і доступна мова програмування, яка дозволяє бачити і змінювати результати своєї роботи в режимі реального часу. Це означає, що платформа надає можливість тестувати, налагоджувати, оптимізувати і демонструвати нейронні мережі в браузері за допомогою графічних і звукових елементів, а також взаємодіяти з ними за допомогою миші, клавіатури або сенсорного екрану. Нейронна мережа здатна використовувати динамічні та інтерактивні веб-сторінки для розпізнавання тексту, генерації зображень, перекладу мови та інших завдань. В цей час швидкодоступна та ефективна база клієнтської частини сервісу забезпечить обробку масивів даних та виконання глибокого аналізу.

Таким чином, JavaScript – це мова, яка дозволяє використовувати нейронні мережі у десктопному браузері без встановлення додаткового програмного забезпечення та залучення ресурсів власного сервера. Це робить нейронні мережі швидкими, зручними та доступними для широкого кола користувачів і застосунків.

Також, можна легко поширити нейронну мережу іншим споживачам за допомогою URL-адреси, QR-коду, соціальних мереж тощо. Крім того, підтримується здатність вбудовувати їх у веб-сайти, блоги, презентації тощо.

Варто вказати, що JavaScript – це відкрита і безкоштовна мова програмування, яка не вимагає ліцензії, плати або реєстрації, що робить її доступною для всіх, хто хоче вчитися і експериментувати з нейронними мережами.

2.2 Аналіз методів створення та навчання нейронних мереж

Машинне навчання та глибоке навчання – це два напрямки штучного інтелекту, які використовують дані для дослідження та вирішення складних проблем. Машинне навчання використовує алгоритми, які навчаються на основі даних, щоб передбачати або класифікувати події на основі набору взаємодій між змінними, які називаються ознаками або атрибутами. Глибинне навчання розширює алгоритми машинного навчання нейронних мереж для вивчення складних завдань, які важко виконувати комп’ютерам, таких як розпізнавання облич і розуміння мови [22].

Нейронні мережі є одним з основних типів алгоритмів машинного навчання, які використовуються для створення штучного інтелекту [23]. Нейронні мережі називаються так тому, що вони засновані на структурі та функціях людського мозку, який складається з мільярдів нервових клітин, які називаються нейронами. Нейрони з’єднані між собою синапсами, які передають сигнали від одного нейрона до іншого [24]. Кожен нейрон отримує сигнали від багатьох інших нейронів, обробляє їх і посилає власний сигнал. Нейронні мережі моделюють цей процес за допомогою математичних об’єктів, які називаються штучними нейронами [25].

Штучний нейрон – це функція, яка приймає кілька чисел, які називаються вхідними даними, і виводить одне число, яке називається вихідними даними [26]. Вхідними даними можуть бути будь-які дані, які ви хочете навчити нейронну мережу розпізнавати або обробляти, наприклад пікселі зображення, звукові

хвилі або текстові слова. Результатом може бути будь-який вихід нейронної мережі, наприклад: мітки класів, значення ймовірності, числа тощо [26].

Штучні нейрони обчислюють результат у два етапи: підсумовування та активація. На першому етапі нейрон множить всі вхідні дані на відповідні ваги і додає всі добутки. Вага - це число, яке визначає важливість кожного входу для результату. Ваги вивчаються нейронною мережею під час навчання. До суми також додається ще одне число, яке називається зміщенням. Його також вивчає нейронна мережа. Зсув дозволяє нейронам регулювати вихідний сигнал.

На другому кроці нейрони застосовують функцію активації до суми і перетворюють суму на вихід. Функція активації – це нелінійна функція, яка додає складності та гнучкості нейронній мережі. Існує багато типів активаційних функцій: наприклад, сигмоїда, гіперболічний тангенс, ReLU, softmax тощо. Вибір функції активації залежить від типу завдання і результатів, які ви хочете отримати.

Нейронна мережа складається з ряду штучних нейронів, розташованих шарами. Кожен шар містить групу нейронів, які отримують вхідні дані від попереднього шару і передають вихідні дані наступному шару. Перший шар називається вхідним, останній шар називається вихідним, а всі шари між ними називаються прихованими. Шари та кількість нейронів у кожному шарі визначають архітектуру нейронної мережі. Архітектура нейронної мережі залежить від типу завдання і даних, які потрібно обробити.

Нейронні мережі навчаються за допомогою процесу, який називається зворотним поширенням помилок. Зворотне поширення помилок – це алгоритм, який порівнює вихід нейронної мережі з очікуваними даними (такими як правильні мітки класу) і обчислює помилку. Помилка – це різниця між вихідними та очікуваними даними, виміряна за допомогою функції втрат. Функція втрат – це функція, яка штрафує нейронну мережу за неточні прогнози. Існує багато типів функцій втрат: середня квадратична помилка, взаємна ентропія, втрати зв’язку тощо.

Вибір функції втрат залежить від типу завдання та результату, який потрібно отримати [26]. Зворотне поширення помилок використовує ланцюгові

правила для розподілу помилок між усіма нейронами та ваговими коефіцієнтами в мережі [26].

Ланцюгове правило – це математичне правило, яке дозволяє обчислити похідну складної функції за допомогою похідних її компонентів. Похідна – це міра зміни функції відносно її аргументу. Похідна показує, наскільки вихід нейрона залежить від вхідних даних і ваг. Зворотне поширення помилок використовує похідні для оновлення вагових коефіцієнтів і вирівнювання нейронів у напрямку, що зменшує помилку.

Зворотне поширення помилки працює у зворотному порядку від вихідного рівня до вхідного. На кожному рівні обчислюється градієнт помилки за допомогою зворотного поширення помилки. Це вектор, що містить часткові похідні помилки відносно кожної ваги та зсуву на рівні. Градієнт похибки показує, наскільки змінюється похибка при зміні ваг і зсувів шару. Зворотне поширення помилки використовує правило оновлення, яке віднімає добуток градієнта помилки і швидкості навчання для оновлення ваг і зміщень шарів.

Швидкість навчання – це гіперпараметр, який визначає швидкість навчання нейронної мережі. Швидкість навчання підбирається емпірично. Помилка зворотного поширення повторюється багато разів для кожного набору вхідних і очікуваних даних, які називаються навчальними прикладами. Навчальні приклади збираються в наборі даних, який поділяється на три частини: навчання, перевірка і тестування. Навчальний набір даних використовується для навчання нейронної мережі з використанням зворотного поширення помилки. Валідаційний набір даних використовується для перевірки якості нейронної мережі під час навчання та для налаштування гіперпараметрів, таких як кількість шарів, кількість нейронів, функція активації, функція втрат та швидкість навчання. Тестові набори даних використовуються для оцінки якості нейронної мережі після навчання та порівняння її з іншими моделями [26].

Зображення та відео є одними з найбільш часто використовуваних типів даних у комп’ютерному зорі. Зображення та відео можуть містити багато інформації про об’єкти, сцени, дії, емоції тощо [27]. Глибоке навчання дозволяє використовувати цю інформацію для різноманітних завдань, таких як

класифікація зображень, розпізнавання облич, розпізнавання дій і створення пояснень [27].

Згорткові нейронні мережі (ЗНМ) – це особливий тип нейронних мереж, які ефективно обробляють просторово структуровані дані, такі як зображення, і тому часто використовуються в контексті глибокого навчання. ШНМ складається з набору шарів, які виконують різні операції над вхідними даними, такі як згортка, об’єднання, нормалізація та активація.

Згортання – це процес застосування невеликих фільтрів до вхідних даних, щоб підкреслити такі особливості, як краї, кути і текстури.

Шар згортки є шаром, який приймає вхідний тензор (наприклад, зображення), застосовує набір фільтрів, які сканують різні області вхідного тензора, і виконує основні операції, такі як додавання, множення та сума. Результатом є вихідний тензор, що містить карту ознак, що представляє виявлені особливості.

Фільтр представлений, як малий тензор із такою ж глибиною, що й вхідний тензор, але з меншою висотою та шириною. Фільтр застосовується до кожного розділу вхідного тензора, виконуючи покомпонентне множення та підсумовування. Фільтри можна використовувати для виявлення певних типів об’єктів, наприклад горизонтальних або вертикальних країв.

Карта ознак – це тензор, що містить значення, що відповідають наявності чи відсутності певної функції в кожній області вхідного тензора. Карти функцій можна використовувати для подальшого аналізу та візуалізації [28].

Об’єднання є процесом зменшення розміру вхідних даних шляхом вибору максимального значення, середнього значення тощо з певного діапазону.

Регулювання виконує операцію масштабування вхідних даних для уникнення надмірної підгонки та нестабільності [29].

Активація – це застосування нелінійної функції до вхідних даних для додавання складності та виразності моделі ШНМ, як і традиційні нейронні мережі, навчаються за допомогою зворотного поширення помилок, але їхня особлива архітектура вимагає меншої кількості параметрів та обчислень ШНМ

застосовуються для класифікації, розпізнавання, сегментації, генерації облич, розпізнавання та інших задач обробки зображень [29].

Генеративно-змагальні нейронні мережі (GAN) є іншим типом генеративної моделі, яка використовується для створення нових даних, подібних до вхідних даних. GAN складається з двох частин: генератора і дискримінатора [30]. Генератор бере випадковий вектор і перетворює його в синтетичні дані. Дискримінатор отримує синтетичні або реальні дані і визначає, чи є вони реальними чи ні. GAN навчаються через змагання між генераторами та дискримінаторами. Генератор намагається обдурити дискримінатор, створюючи реалістичні синтетичні дані. Дискримінатор намагається відрізнити синтетичні дані від реальних [31].

Моделювання ймовірності, невизначеності даних та прогнозування відбувається з використанням байєсівських систем, методів максимальної правдоподібності, ентропії максимальної правдоподібності, теорії інформації, інших теорій ймовірності та статистичних методів, які враховують розподіл даних та шум в процесі генерації та оцінювання [32].

Глибоке навчання також є потужним інструментом для NLP, оскільки воно дозволяє моделям вивчати складні та абстрактні мовні особливості з великих наборів даних [33]. Це передбачає застосування глибоких нейронних мереж до різних задач NLP, таких як класифікація тексту, генерація тексту, переклад мови, розпізнавання мови, аналіз сентименту та інші [34].

Один із кількох способів розглядати нейронні мережі – це складні математичні функції, які отримують два або більше числових значень на вході і видають одне або більше числових значень на виході [35]. Чітке розуміння механізмів входу-виходу нейронних мереж має важливе значення для розуміння того, як працюють системи прогнозування ШІ.

Припустімо, у нас є демонстраційна програма, що створює мережу 3-4-2. Це означає, що є три вхідні значення, чотири так звані приховані вузли, де відбувається більшість обчислень, і два вихідні значення.

Знімок екрана, зображений на рисунку 2.1, показує демонстрацію процесу введення-виведення нейронної мережі.

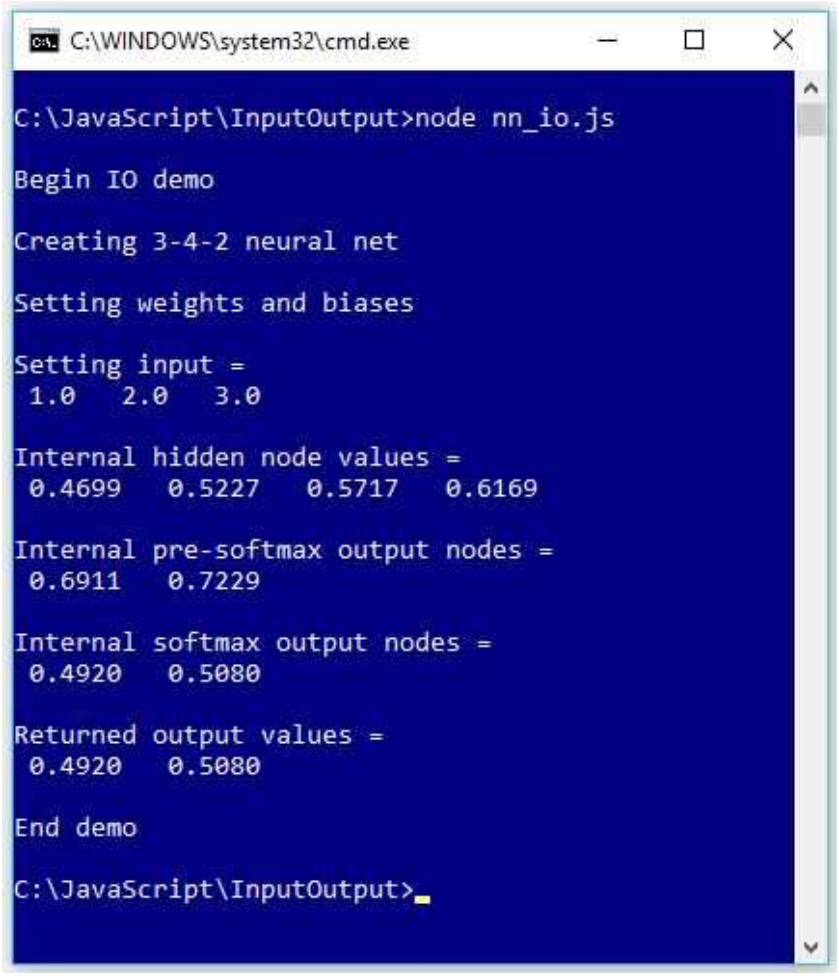

Рисунок 2.1 – Демонстрація введення-виведення нейромережі

У фоновому режимі нейромережа налаштовується шляхом встановлення значень 12 прихованих вхідних ваг, 4 прихованих зсувів, 8 прихованих вихідних ваг і 2 вихідних зсувів (загалом 26 ваг і зсувів). Після встановлення ваг і зсувів демонстраційна програма встановлює три вхідні значення: (1.0, 2.0 і 3.0). Ці значення надсилаються до мережі, і останні два вихідні значення становлять (0.4920, 0.5080). Варто звернути увагу, що сума вихідних значень дорівнює 1.0, що не є випадковістю.

Демонстраційна програма відображає обчислені значення чотирьох внутрішніх прихованих вузлів і попередні значення вихідних вузлів (0,6911, 0,7229) перед застосуванням важливої функції під назвою активація softmax [36].

У нейронних мережах важливо прийти до розуміння механізму входу-виходу. Наведена нижче схема відповідає демонстраційній програмі (рис. 2.2).

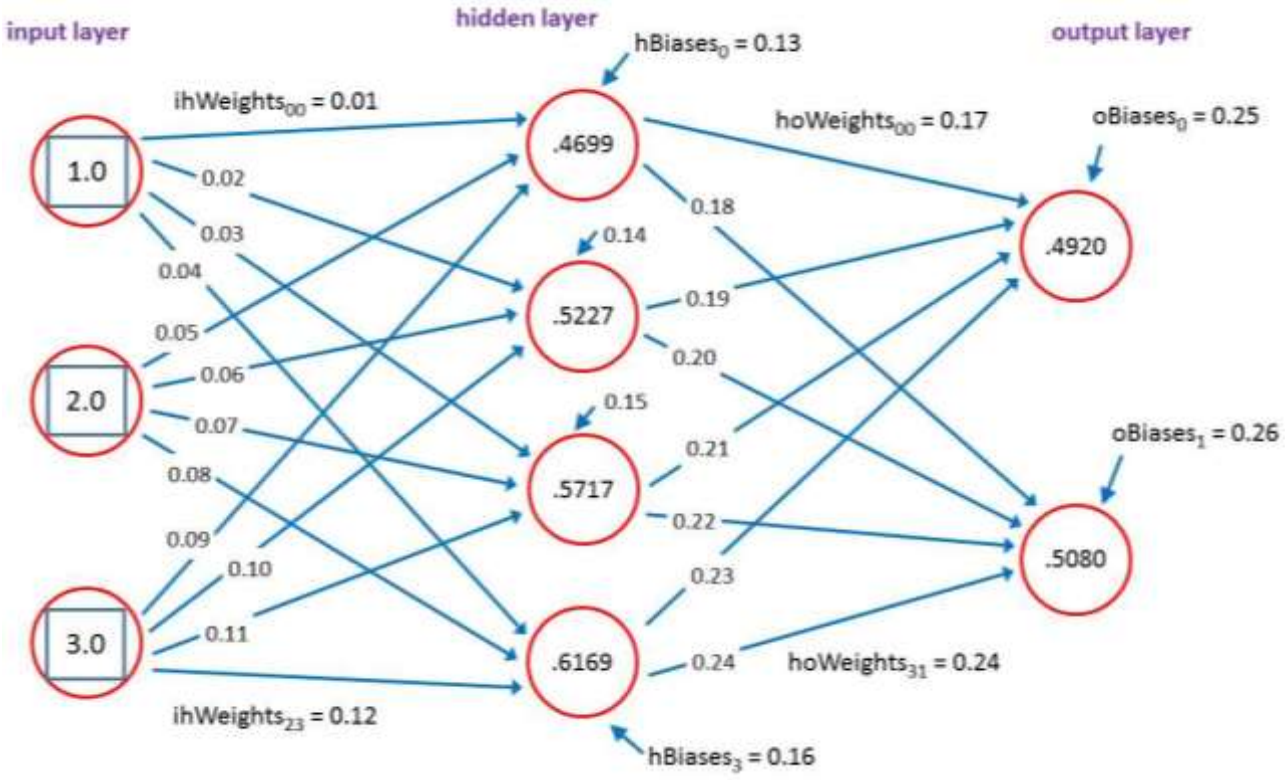

Рисунок 2.2 – Шари нейронної мережі демо-програми

Значення вхідної вершини відповідають числам: 1.0, 2.0, 3.0. Кожна лінія, що з’єднує приховану вершину входу і приховану вершину виходу, являє собою числову константу, яка називається вагою. Якщо вершина [0] індексується з 0 на початку, то вага від input[0] до hidden[0] дорівнює 0.01, а від hidden[3] до output[1] дорівнює 0.24. Ваги можуть бути додатними або від’ємними. До кожної прихованої вершини та кожної вихідної вершини (крім вхідних) додається спеціальна вага, яка називається зміщенням: значення зміщення для hidden[3] дорівнює 0.16, а значення зміщення для output[0] дорівнює 0.25.

Зауважимо, що якщо є ni вхідних вузлів, nh прихованих вузлів і немає вихідних вузлів, загальна кількість вузлів дорівнює (ni * nh) + (nh * no) + nh + ваги і немає зсуву У демонстраційній нейронній мережі 3-4-2, (3 * 4) + (4 * 2) + 4 + 2 = (3 * 4) + (4 * 2) + 4 + 2 = 26 ваг і зсувів.

Щоб обчислити значення прихованої вершини, кожне вхідне значення множиться на відповідну вхідну вагу для прихованої вершини, добуток додається, додаються значення зсуву і до суми застосовується функція гіперболічного тангенса кута нахилу (скорочено tanh). Для прихованої вершини [0] це відбувається наступним чином [36]:

```
sum[0] = (1.0)(0.01) + (2.0)(0.05) + (3.0)(0.09) + 0.13 = 0.5100
hidden[0] = tanh(0.5100) = 0.4699
```

Функція гіперболічного тангенса, яка використовується таким чином, називається функцією активації прихованого шару. Функція активації Tanh встановлює значення всіх прихованих вузлів між -1,0 і +1,0. Значення продуктивності логістичної сигми коливаються від 0,0 до 1,0. Вихідні вершини обчислюються подібним чином, але замість активації tanh, логістичної сигмоіди або ReLU використовується спеціальна функція під назвою softmax. Цю функцію найкраще пояснити на прикладі.

Розрахункова сума значень зсуву вихідних даних "output[0]" і "output[1]" виглядає наступним чином:

```
sum[0] = (0.4699)(0.17) + (0.5227)(0.19) + (0.5717)(0.21) +
(0.6169)(0.23) + 0.25 = 0.6911
sum[1] = (0.4699)(0.18) + (0.5227)(0.20) + (0.5717)(0.22) +
(0.6169)(0.24) + 0.26 = 0.7229
```

Дільник (divisor) обчислюється шляхом застосування функції exp() до кожного елемента суми, а потім підсумовування цих значень.

```
Дільник (divisor) = exp(0,6911) + exp(0,7229) = 1,9960 + 2,0604 =
4,0563
```

Функція exp(x) – це значення x, зведене до числа Ейлера, приблизно 2,71828. Результатом softmax є exp() кожного члена, поділеного на член дільника.

```
output [0] = 1,9960 / 4,0563 = 0,4920
output [1] = 2,0604 / 4,0563 = 0,5080
```

Метою активації softmax є масштабування вихідного значення до 1.0, щоб його можна було вільно інтерпретувати як ймовірність. Припустимо, що демонстрація відповідає завданню, яке має на меті передбачити, чи є людина чоловіком або жінкою на основі трьох предикторних змінних, таких як річний дохід, роки навчання та зріст [36]. Якщо чоловіки кодуються як (1, 0), а жінки як

(0, 1), прогноз буде жіночим, оскільки друге вихідне значення (0,5080) більше за перше (0,4920).

Незважаючи на те, що кодування (1, 0) і (0, 1) для вирішення задач двійкової класифікації використовується відносно рідко, цей приклад відповідає демонстраційній архітектурі нейронної мережі.

Тепер, щоб краще зрозуміти, як працюють нейронні мережі, розглянемо компоненти нейронних мереж та їх параметри з точки зору статті Віталія Кравченка, опублікованої на інтернет-ресурсі LivingFo.

Нейрон – це обчислювальна одиниця, яка отримує інформацію, виконує над нею прості обчислення та передає її далі. Їх можна класифікувати за трьома основними типами: вхідні (сині), приховані (зелені) і вихідні (червоні): Існують також нейрони зміщення та контекстні нейрони.

Коли нейронна мережа складається з великої кількості нейронів, вводиться термін «рівень». Отже, є вхідний рівень, який отримує інформацію, n прихованих рівнів (зазвичай до 3), які її обробляють, і вихідний рівень, який виводить результати.

Кожен нейрон має два основні параметри: вхідні дані та вихідні дані.

Для вхідних нейронів вхідні дані дорівнюють вихідним. Для інших нейронів вхідне поле містить сукупну інформацію про всі нейрони на попередньому рівні, яка нормалізується функцією активації (тут називається просто f(x)), а потім передається у вихідне поле.

Важливо пам’ятати, що нейрони оперують з числами в діапазоні [0,1] або [-1,1]. Щоб обробити числа поза цим діапазоном, необхідно розділити 1 на це число. Цей процес називається нормалізацією і часто використовується в нейронних мережах.

Синапс – це з’єднання між двома нейронами. Синапси мають параметр, який називається вагою. Завдяки цьому, коли вхідна інформація передається від одного нейрона до іншого, вхідна інформація змінюється. Наприклад, припустимо, що є три нейрони, які передають інформацію наступному нейрону. Тоді є три ваги, що відповідають кожному з цих нейронів [36]. Нейрон з

найбільшою вагою має найбільше інформації і домінує над наступним нейроном (наприклад, змішування кольорів).

По суті, набір терезів нейронної мережі або їх матриця є свого роду мозком всієї системи. Завдяки цим вагам вхідна інформація обробляється і перетворюється в результат [36].

У своїй статті Віталій Кравченко також описує біологічні основи нейронних мереж і накладає на це суть програмування: у мозку є нервові клітини. Їх приблизно 86 мільярдів. З біологічної точки зору нейрони – це клітини, які з'єднані з іншими клітинами цього типу. Усе це разом утворює нейронну мережу. Кожна клітина отримує сигнали від інших клітин. Потім він обробляє їх і самостійно посилає сигнали іншим клітинам. Простіше кажучи, нейрони отримують сигнали (інформацію), обробляють їх (виконують обчислення, «мислять») і надсилають свої відповіді. Стрілками зображені зв'язкові прикріплення, які передають інформацію (рис. 2.3) [37].

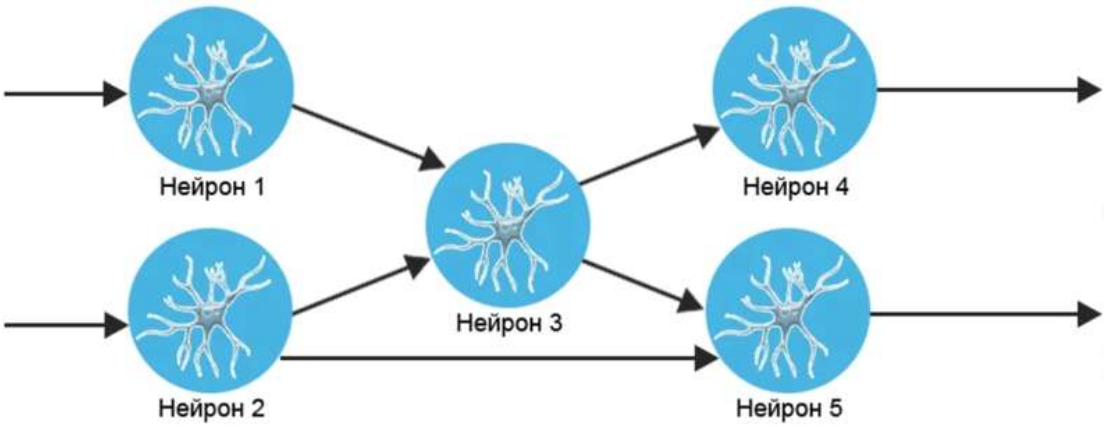

Рисунок 2.3 – Схема функціонування нейронної мережі

Таким чином, передаючи сигнали один одному, нейронні мережі приймають кожне рішення. Висновок полягає в тому, що кожне рішення, яке ми приймаємо, є результатом колективної дії мільярда нейронів.

На схемі, показаній раніше, стрілки вказують на зв'язки між нейронами. Існують різні зв'язки. Наприклад, нижня стрілка, що з'єднує нейрон 2 з нейроном 5, довга [37]. Це означає, що сигнал від нейрона 2 до нейрона 5 проходить

довший шлях, ніж сигнал від нейрона 3, довжина якого вдвічі менша за довжину стрілки. У деяких випадках сигнал може затухати і надходити слабкіше.

В IT всі ці процеси спрощуються і будується «спрощена модель». Ця модель складається з двох основних елементів [37].

Алгоритм – у біології нейрон думає. У програмуванні «мислення» замінюється алгоритмом – тобто набором команд. Наприклад – якщо на вхід прийшла 1, у відповідь відправлється 0.

Вага рішення – усі зв'язки, згасання та інше вирішили замінити «вагою». Вага це як сила рішення, його важливість. Це просто величина, найчастіше число. Наш нейрон приходить рішення з певною вагою, наш нейрон приходить число. І якщо воно більше іншого числа, що прийшло, то воно важливіше. Це як приклад.

Іншими словами, є алгоритм і ваги рішень. Це все, що потрібно для побудови простої нейронної мережі.

Штучна нейронна мережа складається з трьох компонентів: вхідний шар; приховані (обчислювальні) шари; вихідний шар [36]. Схему простої штучної нейронної мережі продемонстровано на рисунку 2.4.

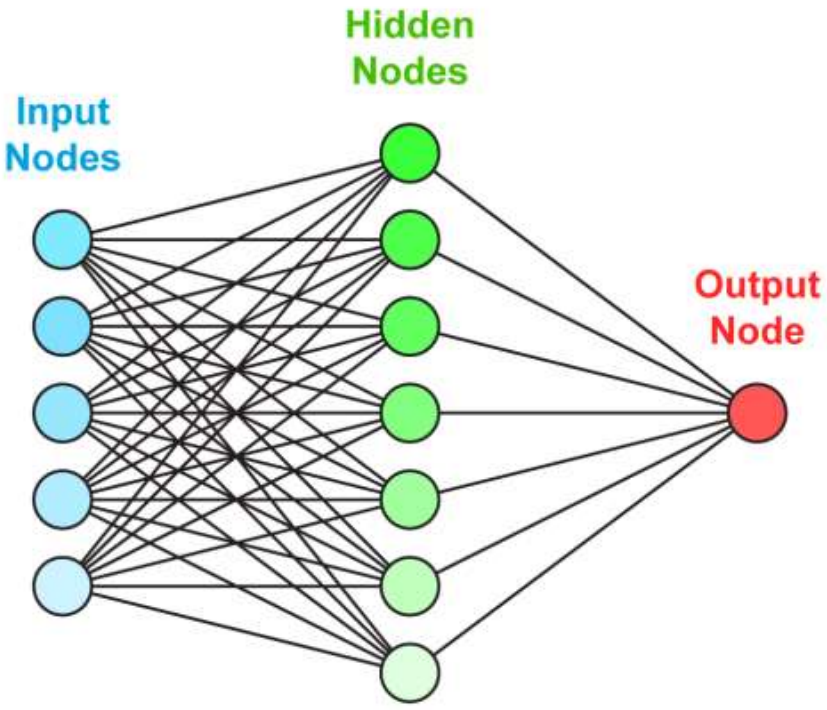

Рисунок 2.4 – Схема простої нейронної мережі

Така нейронна мережа навчається у два етапи: пряме поширення помилки та зворотне поширення помилки. Під час прямого поширення помилки робиться прогноз відповіді. Зворотне поширення помилок мінімізує похибку між фактичними та прогнозованими відповідями.

Одним із найважливіших і критичних критеріїв є те, чи можна навчити нейронну мережу. Загалом, нейронна мережа – це набір нейронів, які посилають сигнали. Сигнал подається на вхід і проходить через тисячі нейронів, але вихід невідомий. Щоб отримати бажані результати з конвертацією, потрібно змінити налаштування мережі.

Вхідний сигнал не може бути змінений. Суматор виконує функцію підсумовування, але не може змінити свій вміст або видалити щось із системи. Залишається одне – використовувати коефіцієнти або корелюючі функції та застосовувати їх на ваги зв’язків. Тут можна дати визначення навчання нейронної мережі: це необхідно для того, щоб знайти набір вагових коефіцієнтів, які дозволять нам отримати потрібний сигнал через суматор.

Це концепція, яку використовує наш мозок – він використовує синапси для посилення або послаблення вхідних сигналів. Люди навчаються, змінюючи синапси, коли електрохімічні імпульси проходять через нейронні мережі мозку.

Однак є одне застереження. Якщо ваговий коефіцієнт встановлюється вручну, нейромережа запам’ятовує правильний вихідний сигнал. У цьому випадку виведення інформації відбувається миттєво, і може здатися, що нейромережа швидко навчилася. Однак незначна зміна вхідного сигналу призведе до того, що на виході будуть неправильні, нелогічні відповіді.

Тому замість того, щоб вказувати конкретні коефіцієнти для одного вхідного сигналу, можна використовувати зразки для створення узагальнених параметрів.

За допомогою таких зразків можна навчити мережу отримувати правильні результати. На цьому етапі навчання нейронної мережі можна розділити на контрольоване і неконтрольоване навчання [37].

Під навчанням у цьому методі мається на увазі концепція подачі зразків вхідного сигналу на нейронну мережу, отримання вихідного сигналу і

порівняння його з готовим рішенням. Наприклад, для розпізнавання обличчя створюється вибірка від 5000 до 10 000 фотографій (вхідний сигнал) і визначається, які з них містять людське обличчя (вихід, правильний сигнал).

Вхідними сигналами може бути як стан ринку загалом, і конкретні дні. Вчителем не обов'язково є людина. Мережа потрібно тренувати сотнями та тисячами годин, тому у 99% випадків тренуванням займається комп'ютерна програма [37].

Окрім стандартних нейронних мереж, що складаються з трьох шарів, також існують глибокі нейронні мережі [38], приклад архітектури якої зображено нижче (рис. 2.5).

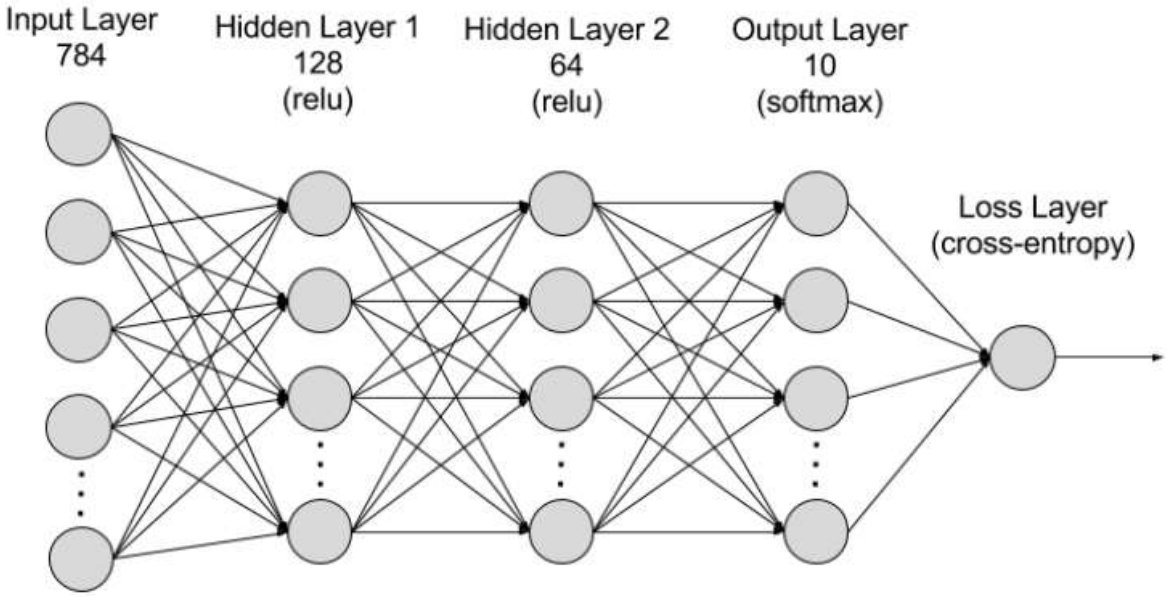

Рисунок 2.5 – Архітектура глибокої нейронної мережі

Глибока нейронна мережа або мережа глибокого навчання може містити кілька закритих шарів із мільйонами підключених штучних нейронів.

Кількість ваг вказує на зв'язок між одним вузлом та іншим. Вага є додатним числом, якщо вузол активує інший вузол, і від'ємним числом, якщо вузол блокує інший вузол. Чим вище значення ваги, тим більше один вузол впливає на інші вузли. Теоретично глибокі нейронні мережі можуть поєднувати будь-які вхідні дані з будь-якими вихідними [36]. Однак потрібно пам'ятати, що глибокі нейронні мережі вимагають набагато складнішого навчання, ніж інші

методи машинного навчання. Тому нам потрібні мільйони прикладів навчальних даних замість сотень або тисяч, як у випадку простих мереж [36].

2.3 Дослідження фреймворків для розробки нейронних мереж на мові програмування JavaScript

Для спрощення та прискорення розробки нейронних мереж на JavaScript існують спеціальні фреймворки, які надають готові бібліотеки, інструменти, алгоритми і інтерфейси для роботи з нейронними мережами. Фреймворки також дозволяють підвищити якість, продуктивність, надійність і безпеку нейронних мереж, а також сприяють їх масштабуванню, тестуванню, відлагодженню і документуванню.

TensorFlow.js – це відкритий фреймворк для машинного навчання та нейронних мереж на JavaScript, який базується на популярній платформі TensorFlow [29]. TensorFlow.js дозволяє створювати, навчати, оцінювати, запускати і використовувати нейронні мережі в браузері або на сервері за допомогою Node.js. TensorFlow.js має багато переваг, таких як:

1) Висока продуктивність, завдяки використанню WebGL для апаратного прискорення обчислень.

2) Гнучкість, завдяки можливості використовувати різні рівні абстракції, від низькорівневих операцій до високорівневих шарів і моделей.

3) Сумісність, завдяки можливості імпортувати та експортувати моделі з інших платформ, таких як Keras, TensorFlow, PyTorch.

4) Інтерактивність, завдяки можливості вбудовувати нейронні мережі в веб-сторінки і додатки, а також використовувати різні типи даних, такі як зображення, звук, відео, текст.

5) Спільнота, завдяки наявності великої кількості документації, прикладів, туторіалів, курсів, блогів, форумів.

Бібліотека TensorFlow.js підтримує різні типи нейронних мереж, такі як згорткові, рекурентні, трансформерні, генеративно-змагальні і багато інших. TensorFlow.js також надає різні алгоритми навчання, такі як градієнтний спуск,

стохастичний градієнтний спуск, адам, RMSProp. TensorFlow.js може застосовуватися в різних сферах, таких як комп’ютерний зір, обробка мови, розпізнавання мови, рекомендаційні системи, генерація тексту, генерація зображень [39].

Це найбільш розвинений і популярний фреймворк, який має велику спільноту, багато документації, прикладів і ресурсів, а також підтримує найбільшу кількість типів нейронних мереж і алгоритмів навчання. TensorFlow.js також має високу сумісність з іншими платформами, такими як Keras, TensorFlow, PyTorch і багато інших. TensorFlow.js є відмінним вибором для розробників, які хочуть створювати складні, інноваційні і функціональні нейронні мережі на JavaScript

Brain.js – це відкритий фреймворк для нейронних мереж на JavaScript, який простий у використанні і має мінімальні залежності. Brain.js дозволяє створювати, навчати і використовувати нейронні мережі в браузері або на сервері за допомогою Node.js. Brain.js має такі переваги, як:

1) Простота, завдяки використанню JSON-формату для представлення даних і моделей, а також використанню простих функцій для створення і навчання нейронних мереж.

2) Швидкість, завдяки використанню GPU.js для апаратного прискорення обчислень на графічних процесорах.

3) Універсальність, завдяки можливості використовувати різні типи даних, такі як числа, масиви, об'єкти, булеві значення, рядки.

4) Модульність, завдяки можливості використовувати різні типи нейронних мереж, такі як звичайні, згорткові, рекурентні, LSTM, GRU.

Бібліотека Brain.js підтримує різні типи нейронних мереж, такі як звичайні, згорткові, рекурентні, LSTM, GRU. Brain.js також надає різні алгоритми навчання, такі як градієнтний спуск, стохастичний градієнтний спуск, адам, RMSProp. Brain.js може застосовуватися в різних сферах, таких як класифікація, регресія, кластеризація, генерація, перетворення, пошук, рекомендація.

У підсумку, Brain.js – простий і швидкий фреймворк, який має мінімальні залежності, а також підтримує багато типів нейронних мереж і алгоритмів

навчання. Brain.js також має високу інтерактивність, оскільки дозволяє вбудовувати нейронні мережі в веб-сторінки і додатки, а також використовувати різні типи даних. Brain.js є хорошим вибором для розробників, які хочуть створювати прості, швидкі і універсальні нейронні мережі на JavaScript. Однак, варто врахувати, що вона не підтримує деякі складні архітектури, такі як трансформери, має обмежену документацію та функціональність, та може мати проблеми зі стабільністю та сумісністю.

Synaptic – це відкритий фреймворк для нейронних мереж на JavaScript, який має високу гнучкість і можливість самоорганізації. Synaptic дозволяє створювати, навчати і використовувати нейронні мережі в браузері або на сервері за допомогою Node.js. Synaptic має такі переваги, як:

1) Гнучкість, завдяки можливості створювати нейронні мережі будь-якої архітектури, топології, функції активації, функції втрати.

2) Самоорганізація, завдяки можливості використовувати алгоритми, які дозволяють нейронним мережам самостійно змінювати свою структуру, параметри, навчання і поведінку, такі як генетичні алгоритми, алгоритми навчання без учителя, алгоритми навчання з підсиленням.

3) Простота, завдяки використанню JSON-формату для представлення даних і моделей, а також використанню простих функцій для створення і навчання нейронних мереж.

4) Швидкість, завдяки використанню GPU.js для апаратного прискорення обчислень на графічних процесорах.

Synaptic підтримує різні типи нейронних мереж, такі як звичайні, згорткові, рекурентні, LSTM, GRU, хопфільдові, перцептрони, лінійні, нелінійні, багатошарові, одношарові і багато інших. Synaptic також надає різні алгоритми навчання, такі як градієнтний спуск, стохастичний градієнтний спуск, адам, RMSProp, генетичні алгоритми, алгоритми навчання без учителя, алгоритми навчання з підсиленням. Synaptic може застосовуватися в різних сферах, таких як класифікація, регресія, кластеризація, генерація, перетворення, пошук, рекомендація.

Synaptic – це гнучкий і самоорганізований фреймворк, який дозволяє створювати нейронні мережі будь-якої архітектури, топології, функції активації, функції втрати і багато іншого. Synaptic також дозволяє використовувати алгоритми, які дозволяють нейронним мережам самостійно змінювати свою структуру, параметри, навчання і поведінку, такі як генетичні алгоритми, алгоритми навчання без учителя, алгоритми навчання з підсиленням і багато іншого. Synaptic є цікавим вибором для розробників, які хочуть створювати гнучкі, самоорганізовані і експериментальні нейронні мережі на JavaScript. Однак, вона не підтримує GPU, має обмежену документацію та спільноту, та не оновлюється з 2017 року

Вибираючи найкращий фреймворк для розробки нейронних мереж за допомогою JavaScript, необхідно врахувати такі фактори, як мета, досвід, ресурси, очікування та вимоги.

Для початківців, або з метою навчання основам нейронних мереж, або для створення і навчання простої нейронної мережі використовуються Synaptic або Brain.js. Вони мають простий та інтуїтивний синтаксис, але не передбачають високої продуктивності, гнучкості або підтримки.

Для високорівневої роботи зі створення професійної нейронної мережі в браузері чи Node.js, варто використати TensorFlow.js, який має потужні та зручні інструменти розробки, навчання та впровадження. Власне, саме через описані фактори цей фреймворк можна вважати найкращим, за наявності досвіду та ресурсів. Він дозволяє працювати з різними типами нейронних мереж, використовувати GPU та WebGL для прискорення обчислень, переносити моделі з Python на JavaScript та навпаки, використовувати предтреновані моделі для різних завдань тощо. Він також надає високорівневі та низькорівневі API для створення та навчання мереж, але вони можуть бути складними та незручними для початківців [40].

Фреймворки для розробки нейронних мереж на JavaScript мають багато переваг, таких як висока продуктивність, гнучкість, сумісність, інтерактивність, модульність і багато іншого. Фреймворки для розробки нейронних мереж на JavaScript підтримують різні типи нейронних мереж, такі як згорткові,

рекурентні, трансформерні, генеративно-змагальні. Також, вони можуть застосовуватися в різних сферах, таких як комп’ютерний зір, обробка мови, розпізнавання мови, рекомендаційні системи, генерація тексту, генерація зображень.

Фреймворки для розробки нейронних мереж на JavaScript є потужними інструментами для розробників, які хочуть створювати інтелектуальні, інноваційні і функціональні веб-сторінки і додатки. Вони відкривають нові можливості в сфері IT [33].

В загальному понятті, модель – це абстрактне представлення нейронної мережі, що визначає її архітектуру, параметри, функції активації, функції втрат, оптимізатори та інші компоненти. Щоб створити модель у TensorFlow.js, потрібно виконати ряд кроків.

Імпортувати необхідні бібліотеки та дані: TensorFlow.js надає різні модулі для роботи з даними, моделями, шарами, оптимізації та візуалізації. Дані можна імпортувати з локальних файлів, URL-адрес або з вбудованих наборів даних, таких як MNIST, CIFAR-10 або IMDB.

Далі потрібно визначити архітектуру моделі. Послідовні методи дозволяють створювати моделі, що складаються з серії шарів, кожен з яких отримує вхідні дані від попереднього шару. Функціональні методи дозволяють створювати моделі з більш складною топологією, наприклад, з гілками, циклами, спільними шарами і т.д. TensorFlow.js надає багато типів шарів, включаючи Dense, Conv2D, LSTM, Dropout і BatchNormalisation TensorFlow.js надає багато різних типів шарів, включаючи.

Подальша компіляція моделі визначає функцію втрат, оптимізатори та метрики, які використовуються для навчання та оцінки моделі. Функція втрат визначає, наскільки точно модель прогнозує очікувані результати.

Оптимізатори оновлюють параметри моделі, використовуючи градієнтний спуск або його різновид, щоб зменшити значення функції втрат. Метрики – це числові показники для відстеження прогресу моделі під час навчання і тестування; TensorFlow.js надає багато вбудованих функцій втрат, оптимізаторів і метрик, таких як категоріальна крос-ентропія, і точність [41].

Навчання моделі означає пристосування моделі до набору даних, що містить приклади вхідних даних (набори) і відповідних вихідних.

Дані можуть бути нормалізовані, стандартизовані, аугментовані, закодовані, векторизовані та іншим чином оброблені, щоб покращити їх якість та сумісність з моделлю даних [42].

Навчання моделі зазвичай здійснюється за допомогою методу припасування. Для цього потрібен навчальний набір даних, кількість епох, розмір партії, валідаційний набір даних та інші параметри; метод підгонки повертає історію навчання з функцією втрат і метричними значеннями для кожної епохи.

Запуск методу evaluate, який приймає тестовий набір даних та повертає значення функції втрати та метрик для моделі. Метод evaluate дозволяє нам порівнювати різні моделі та визначати, яка з них краще виконує наше завдання.

Запуск методу predict, який приймає вхідні дані та повертає вихідні дані, які модель генерує. Метод predict дозволяє нам перевірити, як модель генерує нові дані, які ми можемо візуалізувати, аналізувати та використовувати для наших цілей.

Застосування додаткових метрик та технік, які специфічні для нашого завдання. Наприклад, якщо ми хочемо навчити модель писати, ми можемо використовувати метрики, такі як BLEU, ROUGE, METEOR, які вимірюють якість тексту, який модель генерує. Або якщо ми хочемо навчити модель малювати, ми можемо використовувати метрики, такі як FID, IS, SSIM, які вимірюють якість зображень, які модель генерує.

Розгортання та використання нашої моделі означає інтеграцію нашої моделі в наш веб-додаток або іншу платформу, де ми хочемо використовувати її для генерації нових даних. Розгортання та використання нашої моделі з TensorFlow.js включає такі кроки:

TensorFlow.js надає різні способи збереження та завантаження нашої моделі, такі як локальне сховище, індексована база даних, HTTP-сервер, Google Cloud Storage та інші. Збереження та завантаження нашої моделі дозволяє нам

зберегти нашу модель для подальшого використання або передати її іншим людям або платформам.

TensorFlow.js надає різні інструменти та бібліотеки, які дозволяють нам вбудовувати нашу модель в наш веб-додаток, такі як tfjs-vis, tfjs-react, tfjs-node та інші [37]. Вбудовування нашої моделі в наш веб-додаток дозволяє нам використовувати нашу модель для генерації нових даних на стороні клієнта або сервера, залежно від наших потреб та ресурсів.

TensorFlow.js надає різні можливості для оновлення та покращення нашої моделі, такі як перенавчання, трансферне навчання, активне навчання, федеративне навчання та інші. Оновлення та покращення нашої моделі дозволяє нам підтримувати нашу модель в актуальному стані та покращувати її якість за допомогою нових даних або нових моделей [42].

2.4 Висновок до другого розділу

В другому розділі кваліфікаційної роботи проведено огляд відомих методів розробки нейронних мереж. Наведено опис застосування кожного методу, їх характеристик, особливостей моделі. Також розкрито питання машинного навчання, методів навчання, алгоритмів.

Виконано обґрунтування використання мови програмування JavaScript для розробки нейронних мереж, розгляд фреймворків, які можуть бути застосовані для їх створення. В ході аналізу виділено особливості кожного фреймворка, здійснено порівняльний аналіз їх атрибутів, можливостей. Крім тогог сформовано підсумок за напрямками застосування кожного з фреймворків.

3 СТВОРЕННЯ НЕЙРОННИХ МЕРЕЖ НА МОВІ ПРОГРАМУВАННЯ JAVASCRIPT

3.1 Визначення вимог до розробки нейронних мереж

Коротко опишемо вимоги до розробки нейронних мереж. До них належать:

1) Імпортування необхідних бібліотек та даних. TensorFlow.js, Brain.js та інші надають різні модулі для роботи з даними, моделями, шарами, оптимізацією, візуалізацією і т.д. Дані можуть бути завантажені з локальних файлів, URL-адрес або вбудованих наборів даних, таких як MNIST, CIFAR-10, IMDB тощо.

2) Визначення архітектури моделі. Модель – це абстрактне представлення алгоритму машинного навчання, яке визначає його архітектуру, параметри, функції активації, функцію втрати, оптимізатор та інші компоненти. Визначення архітектури моделі з Scikit-Learn, Keras та TensorFlow включає такі кроки:

3) Вибір типу моделі, який відповідає завданню. Існує багато типів моделей машинного навчання, таких як лінійні моделі, дерева рішень, ансамблі, нейронні мережі, генеративні моделі та інші.

4) Вибір фреймворку, який підтримує наш тип моделі. Не всі фреймворки підтримують всі типи моделей. Наприклад, Brain.js підтримує більшість традиційних моделей машинного навчання, але не підтримує глибокі нейронні мережі. TensorFlow підтримують більшість типів нейронних мереж, але не підтримують деякі генеративні моделі.

5) Компіляція моделі означає визначення функції втрати, оптимізатора та метрик, які будуть використовуватися для навчання та оцінки моделі. Функція втрати вимірює, наскільки добре модель передбачає очікувані результати. Оптимізатор використовує градієнтний спуск або його варіації для оновлення параметрів моделі таким чином, щоб зменшити значення функції втрати. Метрики – це числові показники, які дозволяють нам відстежувати прогрес моделі під час навчання та тестування. Навчання моделі означає підгонку моделі до набору даних, який містить приклади входів та відповідних виходів [43].

6) Навчання моделі з TensorFlow. Навчання моделі включає такі кроки:

1. Розбиття даних на тренувальний, валідаційний та тестовий набори. Тренувальний набір використовується для оновлення параметрів моделі. Валідаційний набір використовується для перевірки якості моделі під час навчання та налаштування гіперпараметрів. Тестовий набір використовується для оцінки якості моделі після навчання.

2. Підготовка даних до навчання. Дані можуть бути нормалізовані, стандартизовані, аугментовані, закодовані, векторизовані та іншим чином оброблені, щоб покращити їх якість та сумісність з моделлю.

3. Запуск процесу навчання. Навчання моделі зазвичай виконується за допомогою методу fit, який приймає тренувальний набір даних, кількість епох, розмір пакета, валідаційний набір даних та інші параметри. Метод fit повертає історію навчання, яка містить значення функції втрати та метрик для кожної епохи.

7) Оцінка та перевірка моделі означає перевірку того, наскільки добре модель генерує нові дані, які схожі на тестовий набір даних, який містить приклади входів та відповідних виходів, які модель не бачила під час навчання. Оцінка та перевірка моделі з TensorFlow включає такі кроки:

1. Запуск методу score, який приймає тестовий набір даних та повертає значення функції втрати та метрик для моделі. Метод score дозволяє нам порівнювати різні моделі та визначати, яка з них краще виконує наше завдання.

2. Запуск методу predict, який приймає вхідні дані та повертає вихідні дані, які модель генерує. Метод predict дозволяє нам перевірити, як модель генерує нові дані, які ми можемо візуалізувати, аналізувати та використовувати для наших цілей.

3. Застосування додаткових метрик та технік, які специфічні для завдання. Наприклад, якщо ми хочемо навчити модель класифікувати зображення, ми можемо використовувати метрики, такі як точність, повнота, матриця помилок, крива ROC, площа під кривою ROC тощо [44].

Або якщо ми хочемо навчити модель генерувати текст, ми можемо використовувати метрики, такі як BLEU, ROUGE, METEOR, які вимірюють якість тексту, який модель генерує [43].

Перейдемо до опису процесу розробки нейронних мереж з використанням мови JavaScript.

3.2 Процес розробки нейронних мереж на мові програмування JavaScript

Для демонстрації можливостей мови програмування JavaScript у контексті створення нейронних мереж виконується створення програмної логіки нейронної мережі на базі мови програмування JavaScript. В якості завдання обробника призначено класифікацію зображень: розпізнавання фото з метою фільтрації контенту.

Відповідно до цього, спочатку, створюється директорія проекту. В якості бази для нейронної мережі буде використано простий шаблон згорткової моделі, що складається з трьох шарів.

Перший рівень використовується для «вирівнювання» вхідних даних. Для зображень, зроблених камерою або завантажених користувачем, зображення «зрівнюється» в 1D-вектор.

Другий повнозв’язаний шар, де кожен нейрон поточного шару з’єднаний з кожним нейроном попереднього шару. Функція активації ReLU використовується тут для вирішення проблеми зникнення градієнта, тобто ситуації, коли градієнти функції втрачаються або занадто зменшуються під час зворотного поширення у глибоких нейронних мережах. Ця проблема стає особливою актуальною в глибоких архітектурах, де кількість шарів велика, оскільки градієнти передаються назад через багато шарів під час навчання мережі [45].

Коли градієнти стають дуже малими, вони не можуть ефективно оновлювати ваги нейронів у вихідних шарах та, таким чином, не ведуть до змін

в поведінці мережі [46]. Це може призводити до того, що нейрони у глибоких частках мережі навчаються дуже повільно або навіть не навчаються взагалі.

Основні причини проблеми зникнення градієнта включають велику кількість шарів у мережі, використання активаційних функцій, які призводять до зменшення вихідного градієнта (наприклад, сигмоїда або тангенс гіперболічний), а також випадкова ініціалізація ваг.

Останній повнозв'язаний шар, який має один нейрон з функцією активації Sigmoid, що застосовується для бінарної класифікації, де вихідний нейрон може вказати ймовірність того, що зображення містить кота.

Визначившись з типом та архітектурою нейронної мережі, а також її призначенням, можна виконати етап наповнення директорії проекту файлами. Так, для початку створюються index.html, який буде містити форму для завантаження фотографії та відображення результатів її обробки, mod.js – файл зберігання коду згорткової моделі для навчання нейронної мережі та script.js – що буде містити код для обробки зображення, завантаженого у форму на index-сторінці.

Перелік створених файлів проекту подано на рисунку 3.1.

Рисунок 3.1 – Файли проекту

З метою навчання нейронної мережі можна використовувати власний датасет з відповідним контентом, або ж готові рішення у вигляді навченої моделі. На базі TensorFlow.js. Hub є можливість виконати завантаження навченої моделі.

На практиці в переважній більшості випадків користувачам не доведеться займатися створенням нових моделей і навчанням їх з нуля на стороні клієнта.

Найчастіше доведеться створювати моделі на основі вже існуючих. Ця техніка називається Transfer Learning.

Принцип Transfer Learning простий. Спочатку модель навчається з використанням великого набору навчальних даних [47]. У процесі навчання нейронна мережа витягує велику кількість корисних характеристик (особливостей) конкретної розв'язуваної задачі, які можуть бути використані як основа для нової, яка буде навчатися на невеликій кількості навчальних даних для більш конкретне, але схоже завдання (рис. 3.2).

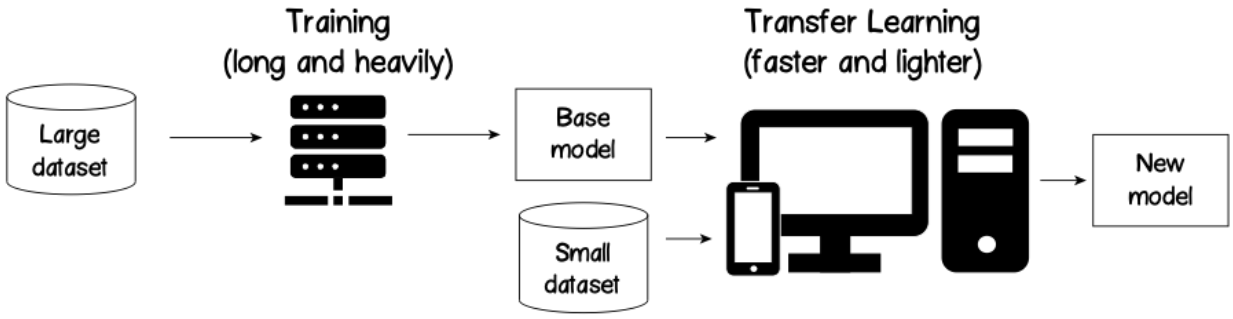

Рисунок 3.2 – Структурна схема Transfer Learning

Таким чином, перенавчання може відбуватися на пристроях з обмеженими ресурсами за відносно менший час.

Для завантаження моделі в TensorFlow використовується спеціальний формат JSON, який, у свою чергу, може бути двох типів: graph-model або layers-model (рис. 3.3).

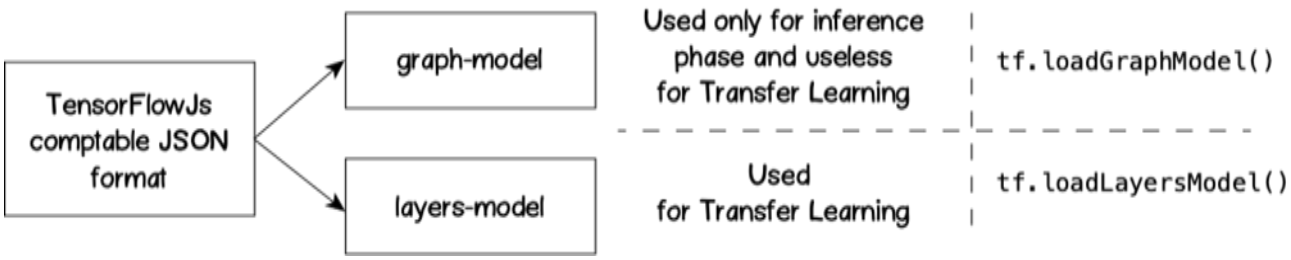

Рисунок 3.3 – Класифікація моделей

Щоб завантажити попередньо навчену модель у форматі моделі шарів, TensrFlowJS надає метод tf.loadLayersModel для API, який завантажує топологію

моделі та її ваги, отримані в результаті тривалого процесу навчання. Топологія моделі вказана у форматі json, який також містить поле weightsManifest, що вказує шляхи у двійковому форматі, що містить усі ваги з'єднань навченої нейронної мережі [47].

Для прикладу, код обробника mod.js із підключенням до навченої моделі виглядатиме, як подано у лістингу 3.1.

Лістинг 3.1 – Підключення навченої моделі:

```
async function loadModel() {
  const model = await
tf.loadLayersModel('https://tfhub.dev/google/tfjs-
model/imagenet/mobilenet_v2_130_224/classification/4/default/1', {
fromTFHub: true });
  return model;
}
```

Продемонстрований вище код завантажить MobileNetV2, який навчений на ImageNet, для класифікації зображень.

Стрічка коду async function loadModel() – це оголошення асинхронної функції loadModel, яка буде використовуватися для завантаження моделі. Функція використовує ключове слово async, оскільки завантаження моделі може зайняти час. Операція await застосовується для очікування завершення завантаження.

Стрічка коду const model = await tf.loadLayersModel('*посилання*', { fromTFHub: true }); завантажує навчену модель. В процесі використовується бібліотека TensorFlow.js (tf), і функція loadLayersModel завантажує модель з вказаного URL. У цьому випадку відбувається завантаження передньо навченої моделі MobileNetV2 для класифікації зображень.

Після завантаження моделі функція повертає її – return model – для подальшого використання.

Проте, важливо відзначити, що використана у прикладі модель буде класифікувати зображення в 1000 класів ImageNet, і не всі ці класи можуть включати необхідні елементи.

Після отримання в код навченої моделі необхідно використати її для цілей обробника. Щоб забезпечити це необхідно модифікувати код mod.js.

Новий скрипт буде розділений на дві частини. Перший фрагмент коду встановлює обробник подій, який викликається при повному завантаженні сторінки (window.onload). При цьому спрацьовує функція loadModel, яка завантажує модель та присвоює її змінній catModel. Це гарантує, що модель буде готова до використання, коли користувач буде готовий використовувати функціонал. Фрагмент модифікованого коду подано у лістингу 3.2.

Лістинг 3.2 – Модифікований код обробника mod.js:

```
let catModel;

// Виклик при завантаженні сторінки
window.onload = async function () {
  // Завантаження н-моделі при завантаженні сторінки
  catModel = await loadModel();
};
```

Наступна функція викликається при виборі користувачем файлу для завантаження. Спочатку вона отримує перший вибраний файл, створює об’єкт зображення (Image), та присвоює йому шлях до завантаженого файлу. Потім встановлює обробник подій, який викликається при повному завантаженні зображення.

В створеному обробнику подій зображення перетворюється в тензор за допомогою tf.browser.fromPixels, змінюється розмір до [224, 224] та розширюється на нову ось за допомогою expandDims.

Застосовується навчена модель для отримання прогнозу. Отримується значення прогнозу з тензору. Відображення результату прогнозу на веб-сторінці: якщо значення більше 0.5, то вважається, що на зображенні ідентифікований об’єкт, і виводиться відповідне повідомлення (див. лістинг 3.3).

Лістинг 3.3 – Модифікований код обробника mod.js:

```
// Обробка завантаженого зображення
async function handleFileUpload(event) {
  // Отримання першого вибраного користувачем файлу з події
  const file = event.target.files[0];
```

```
  // Створення нового об’єкта Image та присвоєння йому шляху до
завантаженого файлу
  const img = new Image();
  img.src = URL.createObjectURL(file);

  // Виклик функції при повному завантаженні зображення
  img.onload = async function () {
    // Перетворення зображення в тензор за допомогою TensorFlow.js
    const tensor =
tf.browser.fromPixels(img).resizeNearestNeighbor([224,
224]).expandDims();

    // Застосування завантаженої моделі для отримання прогнозу
    const prediction = catModel.predict(tensor);

    // Отримання значення з тензору
    const result = prediction.dataSync()[0];
    // Відображення результату тензору на веб-сторінці
    document.getElementById('result').innerText = result > 0.5 ?
'Порушення правил використання!' : 'Без порушень';

    // Відображення зображення
    document.getElementById('preview').src = img.src;
  };
```

Навчання згорткової моделі для класифікації зображень – це складний та витратний процес, який вимагає великого датасету та обчислювальних ресурсів. Поданий вище код демонструє використання TensorFlow.js для роботи з навченими моделями у веб-середовищі. Однак TensorFlow.js дає можливість виконати цей процес самостійно для більш тонкого налаштування і подальшого використання у власних цілях.

Спочатку необхідно підготувати датасет. В залежності від завдань, кількість директорій з відсортованими даними може бути різною. Умвоно, зображення поділені на допустимі та ідентифіковані об’єкти: папки normal та blocked, відповідно.

Для виконання процесу навчання створеної нейронної мережі на основі заготовленого датасету можна виконати в модулі mod.js наступний фрагмент JavaScript коду (див. лістинг 3.4).

Можливість налаштувати кількість фільтрів, розмір ядра та пулінгу моделі відповідно до поставленого завдання є беззаперечною перевагою поданого методу.

Лістинг 3.4 – Навчання на основі датасету:

```
// Створення згорткової нейронної мережі
const model = tf.sequential();
model.add(tf.layers.conv2d({
  inputShape: [224, 224, 3],
  filters: 32,
  kernelSize: 3,
  activation: 'relu'
}));
model.add(tf.layers.maxPooling2d({ poolSize: 2, strides: 2 }));
model.add(tf.layers.flatten());
model.add(tf.layers.dense({ units: 128, activation: 'relu' }));
model.add(tf.layers.dense({ units: 1, activation: 'sigmoid' }));

// Компіляція моделі
model.compile({
  optimizer: 'adam',
  loss: 'binaryCrossentropy',
  metrics: ['accuracy']
});
```

Відповідно до представленго коду, модель складається зі згорткового шару, повнозв’язаного шару та шару активації. Модель компілюється з оптимізатором Adam і функцією втрати бінарної крос-ентропії.

Створюється згорткова нейронна мережа за допомогою tf.sequential() для класифікації зображень на дві групи. Мережа має згорткові та повнозв’язані шари з активацією ReLU та sigmoid. Додається згортковий шар (conv2d) з 32 фільтрами, ядро якого має розмір 3x3 та активацією ReLU.

Після цього додаються шари максимального зведення (maxPooling2d) для зменшення розмірності зображення, розгортання (flatten), який перетворює вихід з попереднього шару у вектор та два повнозв’язаних шари (dense) з активацією ReLU та sigmoid для вирішення задачі бінарної класифікації.

Модель компілюється з оптимізатором Adam та втратою бінарної крос-ентропії. Далі завантажується та підготовлюється датасет із зображень двох категорій: зображення змінюються розміром до [224, 224] та розширюються на нову ось. Датасети об’єднуються, а мітки створюються як вектори [1, 1, ..., 1, 0, 0, ..., 0] для різних типів зображень, відповідно.

Навчається модель за допомогою функції fit, передаючи датасет та мітки. Мітки створюються у вигляді векторів. Модель навчається на датасеті протягом 10 епох (проходів через усі елементи навчального датасету), зберігаючи

результати у TensorBoard, та нарешті, зберігається у файлі "my_model" для подальшого використання.

Визначається оптимізатор (adam), функція втрат (binaryCrossentropy) та метрика (accuracy) для компіляції моделі.

Після навчання модель зберігається у файл mod для подальшого використання. Друга частина програмного коду описана у лістингу 3.5.

Лістинг 3.5 – Розмітка другої моделі:

```
// Завантаження та підготовка датасету
const catsDataset =
tf.data.imageFromDirectory('img').concatMap(img =>
tf.image.resizeBilinear(img, [224, 224]).expandDims());
const nonCatsDataset =
tf.data.imageFromDirectory('path').concatMap(img =>
tf.image.resizeBilinear(img, [224, 224]).expandDims());

const dataset = catsDataset.concatenate(nonCatsDataset);
const labels = tf.concat([tf.ones([catsDataset.size]),
tf.zeros([nonCatsDataset.size])]);

// Навчання моделі
model.fit(dataset, labels, {
  epochs: 10,
  shuffle: true,
  callbacks: tf.node.tensorBoard('logs')  // Можна використовувати
TensorBoard для візуалізації
}).then(info => {
  console.log('Навчання завершено:', info);
  // Збереження навченої моделі
  model.save('downloads://mod');
});
```

Далі, в index.html додається розмітка сторінки, що буде використовуватись з метою завантаження фотографії, описана у лістингу 3.6.

Лістинг 3.6 – Розмітка другої моделі:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
  <title> Фільтрація рисунків </title>
</head>
```

```
<body>
  <input type="file" accept="image/*"
onchange="handleFileUpload(event)">
  <img id="preview" style="max-width: 300px;">
  <p id="result"></p>
  <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
  <script src="main.js"></script>
</body>
</html>
```

Елемент script в даному випадку відіграє роль посилання на підключення до бібліотеки TensorFlow: <script src="https://cdn.jsdelivr.net/npm/@tensorflow /tfjs"></script>.

Отже, практика підтвердила, що алгоритми JavaScript у співпраці зі TensorFlow здатні до створення нейронних мереж на базі згорткової моделі.

Тепер розглянемо програмний код наступного типу нейронних мереж, поданий у лістингу 3.7.

Лістинг 3.7 – Програмна логіка рекурентної моделі:

```
// Створення рекурентного шару LSTM
const lstmLayer = tf.layers.lstm({
  units: 64,                // Кількість LSTM-нейронів
  inputShape: [10, 1],      // Форма вхідних даних (припустимо, що
текст має 10 символів)
  returnSequences: true     // Повертати послідовності на виході
});
// Створення звичайного повнозв'язаного шару
const denseLayer = tf.layers.dense({
  units: 1,                 // Один вихідний нейрон (фільтрація: 1
- фільтр, 0 - не фільтр)
  activation: 'sigmoid'     // Визначення ймовірності фільтрації
});
// Збірка моделі
const model = tf.sequential();
model.add(lstmLayer);
model.add(denseLayer);
// Компіляція моделі
model.compile({
  optimizer: 'adam',        // Оптимізатор Adam
  loss: 'binaryCrossentropy'// Функція втрати - бінарна крос-
ентропія для задачі бінарної класифікації
});
// Генерація синтетичних даних для навчання
const generateData = () => {
  const data = [];
  for (let i = 0; i < 100; i++) {
```

```
    const isFiltered = i % 2 === 0;  // Фільтрація через символ
    data.push({ x: i / 10, y: isFiltered ? 1 : 0 });
  }
  return data;
};
```

У цьому випадку нейронна мережа на базі рекурентного типу застосовується для фільтрації тексту. Модель вивчає, які частини тексту слід фільтрувати (1) та які залишати (0). Тут застосовується LSTM-шар для врахування контексту тексту. Цей шар має 64 LSTM-нейрони та приймає вхідні дані у вигляді текстового ряду з 10 символами. Параметр returnSequences вказує, що вихід має містити послідовності, а не лише останнє значення.

Наступний за ним – повнозв’язаний шар з одним вихідним нейроном та активаційною функцією сигмоїди – застосовується з метою визначення ймовірності фільтрації.

Далі, подібно до згорткової ШНМ, модель компілюється з оптимізатором Adam та функцією втрати бінарної крос-ентропії для задачі бінарної класифікації.

Нарешті, функція generateData генерує синтетичний текстовий ряд, в якому кожен другий символ буде фільтруватись (1), а інші залишатись (0).

Друга частина коду, розроблена для навчання рекурентної нейронної мережі подана у лістингу 3.8.

Лістинг 3.8 – Процес навчання нейронної мережі:

```
// Підготовка навчання (дані)
const trainingData = generateData();
const xs = tf.tensor2d(trainingData.map(item => [item.x]));  //
Вхідні дані - часова компонента
const ys = tf.tensor2d(trainingData.map(item => [item.y]));  //
Вихідні дані - мітки фільтрації
// Навчання моделі
model.fit(xs, ys, { epochs: 50 }).then(() => {
  // Передбачення на основі навченої моделі
  const newDataPoint = { x: 10, y: null };  // Представлення
нового текстового символу
  const inputTensor = tf.tensor2d([[newDataPoint.x]]);
  const prediction = model.predict(inputTensor);
  newDataPoint.y = Math.round(prediction.dataSync()[0]); //
Округлення до 0 або 1
```

```
  console.log('Передбачення x=10:', newDataPoint.y === 1 ?
'Порушення правил спільноти!' : 'Все чисто');
});
```

За процесом виконання, дані збираються у формат, зручний для нейронної мережі, та модель навчається протягом 50 епох.

Вибір кількості епох є гіперпараметром, тобто це параметр, який не навчається моделлю, а встановлюється дослідником на підставі досвіду та вимог конкретного завдання. Кількість епох впливає на те, як добре модель вчиться на навчальних даних. Однак немає універсального значення, і оптимальна кількість епох може залежати від різних факторів, таких як розмір датасету, складність завдання, архітектура моделі і т.д.

Зазвичай велика кількість епох може призвести до перенавчання (overfitting), коли модель стає дуже доброю на навчальних даних, але погано узагальнюється на нові дані. У той час як невелика кількість епох може призвести до недонавчання (underfitting), коли модель не вивчає всі особливості даних.

Таким чином, кількість епох визначається шляхом експериментів. Зазвичай рекомендується розділити дані на тренувальний і валідаційний набори, щоб слідкувати за показниками ефективності моделі під час навчання та визначити оптимальну кількість епох, яка забезпечує найкращі результати на валідаційному наборі.

На основі навченої моделі робиться передбачення для нового текстового символу. Результат виводиться в консоль як "Порушенн правил спільноти" чи "Все чисто".

Тепер необхідно здійснити прив’язку нейронної мережі до HTML-розмітки сторінки з полем внесення інформації, вміст якого нейронна мережа буде обробляти на предмет фільтрації. Перша частина розмітки подана у лістингу 3.9.

Лістинг 3.9 – Розмітка сторінки:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
  <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
  <title>Фільтрація тексту</title>
  <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
</head>
<body>

<h1>Фільтрація тексту</h1>

<label for="textInput">Enter Text:</label>
<input type="text" id="textInput" oninput="filterText()">

<p id="result"></p>

</body>
</html>
```

В кінець розмітки вноситься елемент script з обробником, як описано у лістингу 3.10.

Лістинг 3.10 – Внесення скрипту

```
  const lstmLayer = tf.layers.lstm({
    units: 64,
    inputShape: [10, 1],
    returnSequences: true
  });
  const denseLayer = tf.layers.dense({
    units: 1,
    activation: 'sigmoid'
  });
  const model = tf.sequential();
  model.add(lstmLayer);
  model.add(denseLayer);
  model.compile({
    optimizer: 'adam',
    loss: 'binaryCrossentropy'
  });
  // Функція для фільтрації тексту
  async function filterText() {
    const inputElement = document.getElementById('textInput');
    const resultElement = document.getElementById('result');
    // Отримання тексту з поля вводу
    const inputText = inputElement.value;
    // Перетворення тексту в тензор
    const inputTensor =
tf.tensor2d([Array.from(inputText)].map(char =>
[char.charCodeAt(0) / 255]));
    // Передбачення на основі навченої моделі
    const prediction = model.predict(inputTensor);
    // Отримання значення прогнозу з тензору
```

```
    const filterResult = Math.round(prediction.dataSync()[0]);
    // Відображення результату на веб-сторінці
    resultElement.innerText = filterResult === 1 ? 'Заборонене
слово!' : 'Все чисто';
  }
</script>
```

У цьому прикладі, ми використовуємо LSTM-модель для фільтрації тексту, де ми визначили, що текст буде фільтруватись, якщо вихід моделі більше 0.5, інакше текст буде допущений. Результат виводиться під полем вводу.

Окрім розглянутих, існують і інші типи нейронних мереж, які можна створити на мові програмування JavaScript. Наприклад, нейронні мережі прямого зв’язку – найпростіша форма нейронних мереж, де інформація переміщується вперед від входу до виходу без зворотного зв'язку. Вони складаються з вхідного шару, прихованих шарів і вихідного шару, як покзано на лістингу 3.11.

Лістинг 3.11– Найпростіший запис нейронної мережі:

```
const model = tf.sequential();
model.add(tf.layers.dense({inputShape: [inputSize], units:
hiddenSize, activation: 'relu'}));
model.add(tf.layers.dense({units: outputSize, activation:
'softmax'}));
```

Також, генеративно-змагальні нейронні мережі також розробляються на платформі JavaScript та успішно застосовуються для генерації нових зображень або даних. Вони складаються з генератора, який створює нові зразки, та дискримінатора, який визначає, чи це реальні зразки чи штучно створені. Тим не менше, імплементація такого типу нейронних мереж може бути складною та вимагати більше ресурсів.

Це лише декілька прикладів типів нейронних мереж, які можна створити з використанням JavaScript та TensorFlow.js., або інших фреймворків Різноманітність цих мереж дозволяє вирішувати різні завдання у сферах від класифікації до генерації контенту.

За результатами дослідження та практичної реалізації, можна сформулювати висновки щодо можливостей, інструментів, переваг та недоліків JavaScript у цій сфері.

Можливості JavaScript у створенні нейронних мереж:

- TensorFlow.js є потужним інструментом для створення нейронних мереж у веб-середовищі. Вона надає API для визначення, навчання та використання моделей нейронних мереж, включаючи підтримку різних типів шарів та оптимізаторів.
- JavaScript дозволяє створювати як згорткові, так і рекурентні нейронні мережі. Згорткові мережі ефективно працюють з візуальними даними, тоді як рекурентні використовуються для обробки послідовних даних, таких як текст чи звук.
- Завдяки можливостям JavaScript, нейронні мережі можуть бути вбудовані безпосередньо в веб-додатки, надаючи користувачам інтерактивність та реальний час аналізу даних.

В основі успіху реалізацій нейронних мереж JavaScript лежить бібліотека TensorFlow.js, яка відкриває доступ до потужного функціоналу машинного навчання прямо у веб-середовищі.

Крім того, JavaScript може виконуватися як у веб-браузерах, так і на сервері за допомогою Node.js, що розширює можливості розгортання нейронних мереж на різних платформах.

До переваг розробки нейронних мереж на JavaScript відноситься:

1) Легкість використання: JavaScript має низький поріг входження, що дозволяє розробникам швидко починати роботу зі створенням нейронних мереж.

2) Широка розповсюдженість: за рахунок широкої розповсюдженості браузерів, JavaScript доступний для більшості користувачів та розробників.

3) Інтерактивність: можливість вбудовувати нейронні мережі в веб-додатки надає можливість для інтерактивного аналізу даних.

До недоліків розробки нейронних мереж на JavaScript відносяться:

1) Обмежені ресурси: у веб-середовищі обчислювальні ресурси можуть бути обмеженими, особливо на пристроях з обмеженими характеристиками.

2) Навчання на клієнті: навчання складних моделей на клієнтському боці може призвести до навантаження на ресурси пристрою користувача.

3) Залежність від браузера: деякі функціональності можуть залежати від можливостей конкретного браузера.

Сучасна область машинного навчання невпинно розвивається, і JavaScript стає все більш важливим інструментом для створення та розгортання нейронних мереж різних типів.

Більше того, JavaScript стає ключовим інструментом у сфері створення нейронних мереж, забезпечуючи легкість використання та інтерактивність. TensorFlow.js розширює можливості розробки у веб-середовищі, дозволяючи реалізовувати різноманітні типи нейронних мереж. На хвилі активного розвитку машинного навчання, JavaScript відкриває нові перспективи для застосування інтелектуальних систем прямо в онлайн-середовищі. Однак слід звертати увагу на обмежені ресурси та вибирати оптимальні стратегії розгортання в залежності від конкретних вимог проекту.

3.3 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи виконано загальний опис вимог до створення нейронних мереж. Вказано орієнтири для вибору правильного фреймворку з відповідними прикладами. Висвітлено базові поняття та складові процесу навчання нейронної мережі.

На основі використання одного з вказаних фреймворків JavaScript продемонстровано можливість підключення його до проекту та подальшого використання.

Описано процес створення програмних рішень на основі рекурентної та згорткової моделей нейронних мереж, з поясненням використаного програмного коду. Визначено та реалізовано основний напрямок функціонування нейронних мереж кожного типу. Відповідно до останнього, створено веб-інтерфейс для демонстрації можливості інтеграції нейронної мережі у веб-середовище та її застосування там.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Тема кваліфікаційної роботи освітнього рівня «Магістр» присвячена аналізу методів створення нейроних мереж з використанням мови програмування JavaScript. Тому доцільно розглянути питання вимог до ергономікиробочого місця оператора ПК, вимог до безпеки з охорони праці та профілактичних медичних оглядів для працівників ПК.

Робота з комп'ютером характеризується значною розумовим та нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи та досить великим навантаженням на м'язи рук при роботі з клавіатурою ПК. Велике значення має раціональна конструкція та розташування елементів робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора [48].

У процесі роботи з комп'ютером необхідно дотримуватися правильного режиму праці та відпочинку. В іншому випадку у персоналу відзначаються значне напруження зорового апарату з появою скарг на незадоволеність роботою, головний біль, дратівливість, порушення сну, втому і хворобливі відчуття в очах, в попереку, в області шиї та руках [49].

Вимоги щодо організації та обладнання робочих місць при розробці програмного забезпечення включає в себе площу, відведену на одне робоче місце не менше 6 м^2. Конструкція робочого місця повинна забезпечувати підтримання оптимальної робочої пози, тобто такої, яка дозволяє працівникові при написанні коду на ПК виконувати роботу з мінімальним напруженням тіла, і яка дозволяє уникнути перевтоми в ході і після закінчення робочого процесу [50].

Раціональна робоча поза має важливе значення для збереження здоров'я працівника, оскільки тривале перебування його в незручній і напруженій позі може призвести до таких захворювань, як сколіоз, варикозне розширення вен, плоскостопість тощо. Установлено, що робота в зігнутому положенні збільшує затрати енергії на 20%, а при значному нахиленні — на 45% порівняно з прямим

положенням корпуса.

За потреби особливої концентрації уваги під час виконання робіт з написання коду для проекту суміжні робочі місця необхідно відділяти одне від одного перегородками висотою 1,5 – 2 м [51].

Забарвлення приміщень та меблів має сприяти створенню сприятливих умов для зорового сприйняття та позитивного настрою.

Джерела світла, такі як світильники та вікна, які дають відображення від поверхні екрана, значно погіршують точність знаків і спричиняють перешкоди фізіологічного характеру, які можуть виразитися у значній напрузі, особливо при тривалій роботі.

Недостатність освітлення призводить до напруги зору, послаблює увагу, призводить до настання передчасної втоми. Надмірно яскраве освітлення викликає засліплення, роздратування та різь в очах. Неправильний напрямок світла робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань, тому настільки важливим є правильний розрахунок освітленості.

Відображення, включаючи відображення від вторинних джерел світла, має бути мінімальним. Для захисту від надмірної яскравості вікон можуть бути використані штори та екрани.

Робочі місця слід розташовувати відносно джерела природного світла, тобто вікон, таким чином, щоб світло падало на клавіатуру програміста збоку, переважно зліва. Також робоче місце для роботи на ПК має відповідати сучасним вимогам ергономіки [52]:

- стіл повинен мати висоту поверхні 680 - 800 мм, ширину 600 - 1400 мм і глибину 800 - 1000 мм. Такі параметри забезпечують можливість виконання операцій в зоні досяжності працівника;
- робочий стілець робочий стілець має бути підйомно-поворотним, з можливістю регулювання висоти, бажано зі стаціонарними або змінними підлікотниками і напівм’якою нековзкою поверхнею сидіння, що легко чиститься і не електризується;

- екран комп’ютера має розташовуватися на оптимальній відстані від користувача з урахуванням літерно-цифрових знаків і символів. Стандартне значення становить 600 – 700 мм.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість монітору, зручність ручного керування пристроєм введення-виведення інформації.

Техніка безпеки для програміста – це запорука довготривалої та безпроблемної роботи такого фахівця. Техніка безпеки програмістів регулюється «Інструкцією з охорони праці», де все розкладено за пунктами та дуже докладно описано. Знати її потрібно, якщо програміст працює у великій офісній будівлі, де до комп’ютера мають непрямий доступ кілька людей. У цьому випадку він зобов’язаний слідувати інструкціям техніки безпеки, щоб не наражати на небезпеку своє здоров’я і здоров’я оточуючих його колег. Плюс програміст просто повинен знати, як поводитися під час надзвичайних ситуацій.

Зазвичай техніка безпеки програміста зачитується фахівцями з безпеки праці кожної організації, де працюють програмісти, чи вона доступна прочитання кожним співробітником. А щорічно, іноді й частіше, всі співробітники розписуються, що ознайомлені з технікою безпеки. Фактично техніку безпеки мало хто читає, мало хто знає і мало хто дотримується, тому що все обмежується тим, що програмісти просто розписуються в журналі, ніби вони «ознайомлені» і спокійно працюють далі.

Коли програміст працює віддалено з дому, вся відповідальність за його безпеку лежить на його плечах. Коли програміст працює в офісі, то відповідальність за його безпеку лежить на плечах програміста та окремого спеціаліста, який має слідкувати за дотриманням техніки безпеки. І в тому, і в іншому випадку програміст повинен знати основи охорони праці, описані нижче.

Вся техніка безпеки для програміста поділяється на кілька етапів: до початку роботи, під час роботи, після закінчення роботи; у разі аварійної ситуації.

Програміст перед стартом своєї роботи повинен:

1) провести огляд свого робочого місця;

2) провести регулювання освітлення, щоб екран був добре видно і не відображав світло;

3) проконтролювати коректне підключення електричних частин комп’ютера до мережі;

4) проконтролювати відсутність оголених частин проведення на електричних проводах комп’ютера;

5) провести перевірку цілісності столу, стільця, підставки для ніг, висувної частини столу для клавіатури тощо; якщо потрібно, програміст повинен відрегулювати всі ці моменти.

Під час своєї роботи програміст повинен:

1) стежити за чистотою свого робочого місця;

2) не закривати вентиляційні вікна комп’ютера;

3) коректно припиняти роботу комп'ютера, коли це необхідно;

4) стежити за дотриманням свого графіка роботи та відпочинку;

5) правильно та за призначенням експлуатувати комп’ютер та всі його частини;

6) вчасно виконувати фізичні вправи для очей, шиї, рук та тулуба;

7) стежити за своїм розташуванням на робочому місці: правильна постава, відстань до та екрану тощо.

Після закінчення робочого дня або свого робочого часу програміст повинен:

1) правильно завершити роботу всіх запущених програм та пристроїв;

2) перевірити відсутність у дисководах дисків чи дискет;

3) відключити системний блок від електромережі;

4) вимкнути додаткові пристрої від електромережі;

5) оглянути своє робоче місце і привести його до ладу, якщо це необхідно.

Перш за все, відповідно до ст. 169 Кодексу законів про працю України та ст. 17 Закону України «Про охорону праці» від 2002 року роботодавець зобов’язаний за свої кошти організувати проведення попереднього (при прийнятті на роботу) і періодичних (протягом трудової діяльності) медичних оглядів працівників, зайнятих на важких роботах, роботах зі шкідливими чи

небезпечними умовами праці або таких, де є потреба у професійному доборі, а також щорічного обов’язкового медичного огляду осіб віком до 21 року.

Метою проведення обов’язкових профілактичних медичних оглядів є запобігання розповсюдженню інфекційних та небезпечних захворювань, динамічне спостереження за станом здоров'я працюючого населення.

Такий вид оглядів передбачений статтею 21 Закону України «Про захист населення від інфекційних хвороб» та статтею 26 Закону України «Про забезпечення санітарного та епідемічного благополуччя населення».

Таким чином працівники окремих професій, виробництв та організацій, діяльність яких пов’язана з обслуговуванням населення і може призвести до поширення інфекційних хвороб, повинні проходити обов’язкові попередні (до прийняття на роботу) і періодичні профілактичні медичні огляди.

4.2 Безпека в надзвичайних ситуаціях

Тема кваліфікаційної роботи освітнього рівня «Магістр» присвячена аналізу методів створення нейроних мереж з використанням мови програмування JavaScript. Тому доцільно розглянути питання підвищення стійкості роботи об’єктів приладобудування у воєнний час.

Проблематика стійкості об’єктів приладобудування у воєнний час є критичним фактором, який впливає на безпеку та ефективність систем, що покладаються на точні дані та контроль.

За загальною теоретикою, стійкість об’єктів приладобудування залежить від багатьох факторів: якість компонентів, процедури ремонту та обслуговування, аналіз навантаження та надмірностей, застосування методологій надання послуг, захищеність від надзвичайних ситуацій, здатність інженерно-технічного комплексу протистояти руйнуванню, надійність постачання енергетики та інших ресурсів, підготовлення до аварійно-рятовних та відновлюваних робіт. У воєнний час це означає, що об’єкти приладобудування повинні бути стабільними та надійними у будь-яких умовах, навіть якщо вони

знаходиться під загрозою, або пошкоджене ворожим нападом, або умовами навколишнього середовища.

Моделі забезпечення стійкості об’єктів приладобудування – це способи описати та аналізувати поведінку і функціонування таких об’єктів у різних умовах і ситуаціях. Вони можуть мати різну складність і глибину, але загальною метою є визначити потенційні ризики і надзвичайні ситуації, що можуть загрожувати стабільності і надії об’єктів приладобудування.

Існує ряд методів покращити стабільність об’єктів приладобудування у воєнний час.

Проектування інженерно-технічних заходів цивільного захисту (ІТЗЦ), які передбачають врахування можливих надзвичайних ситуацій та їх наслідків для об’єкта, а також розробку ефективних дій для запобігання або зменшення їх впливу [53].

Використання спеціальних захисних споруд і особливих конструкцій на радіаційно-, вибухо- і пожежонебезпечних об’єктах, будівництво дамб і обвалування в районах можливих затоплень, укріплення схилів у районах з підвищеним ризиком зсувів.

Також до конструкційної стійкості можна віднести розташування об’єктів важливих для оборони у місцях з природними або штучними перешкодами, які ускладнюють атаку. Використання твердих матеріалів та конструкцій, які можуть витримувати великі навантаження. Застосування технік маскування для затруднення визначення місцезнаходження об’єкта [54].

Крім того, можна розташування багато функціональних об’єктів поруч з цільовим об’єктом для створення плутанини та ускладнення визначення цілей.

Застосування нових технологій, матеріалу для покращення якості ефективності обладнання і систем приладобудування, наприклад: електронне управління, автоматизації, механизмизації, оптимизації.

Використання прогресивного управління для передбачення потенційних проблем на полісі і запровадження заходів для їх запобігання. Наприклад, регулярний аналіз споживання запасних частин і перевірка їх сумісності з всіма об’єктами на полісі.

Надання послуг для оптимізації ремонту та обслуговування приладів. Наприклад, автоматизована інспекція та моніторинг для виявлення та усунення несправностей. Сюди ж можна додати застосування мобільних та легко пересувних систем для уникнення ворожих атак та ускладнення визначення місцезнаходження. Використання техніки та транспортних засобів з підвищеною мобільністю.

Моделювання для аналізу та оптимізації параметрів і поведінки приладів. Наприклад, використовувати математичні методи та комп'ютерне моделювання для проектування та симуляції систем.

У воєнний час забезпечення економічної та соціальної стійкості об’єктів приладобудування може виявитися складним завданням через можливі економічні труднощі та соціальні виклики. Деякі моделі та методи для забезпечення стійкості у воєнний час включають економічне планування та ресурсне управління, соціальну політику та соціальний захист, створення резервів та запасів, господарську диверсифікацію, інфраструктурну стійкість, економічне збалансування та управління ризиками. Застосування цих моделей спрямоване на створення комплексного підходу до забезпечення стійкості у воєнний час, охоплюючи економічний та соціальний аспекти. Специфічні заходи можуть залежати від конкретних умов та вимог об’єкта приладобудування.

Метою економічного планування та управління ресурсами є забезпечення оптимального використання ресурсів і фінансів для підтримки стабільності економічної системи. Результат досягається шляхом ретельного планування бюджету, аналізу ризиків, диверсифікації джерел економіки та впровадження стратегій скорочення витрат.

Соціальна політика та соціальний захист спрямовані на зменшення негативного впливу воєнних подій на населення та соціальну сферу. Це відбувається через запровадження систем соціального забезпечення, надання гуманітарної допомоги, розробки програм зайнятості та навчання [55].

Створення резервів і техніки спрямоване на забезпечення доступу до стратегічних ресурсів і техніки у воєнний час. Результат підходу може бути

досягнутий шляхом створення національного запасу критично важливих ресурсів, які можна використовувати для подолання економічних труднощів.

Економічна диверсифікація має на меті зменшити залежність від окремих галузей чи ринків і забезпечити економічну стабільність. Це досягається за рахунок розвитку різних секторів економіки, просування інновацій та відкриття нових ринків.

Відмовостійкість інфраструктури спрямована на забезпечення функціональності критичної інфраструктури під час війни. Для цього необхідні такі заходи, як захист критично важливих об’єктів, розробка планів аварійного відновлення та створення резервних систем зв’язку та електроенергії.

Економічний баланс і управління ризиками спрямовані на мінімізацію економічних ризиків і забезпечення стабільності в неспокійному економічному середовищі. Це досягається шляхом проактивного управління ризиками, аналізу та адаптації до мінливих економічних умов.

4.3 Висновок до четвертого розділу

В четвертому розділі кваліфікаційної роботи описані основні аспекти охорони праці в предметній області: описано вимоги ергономіки робочого місця, оформлення робочого кабінету та необхідних умов для роботи за ПК. Описані фактори та умови, які позитивно та негативно впливають на людину при роботі. Також висвітлені теми вимог безпеки та проведення профілактичних медичних оглядів для операторів ПК.

Розглянуто способи виконання завдання з підвищення стійкості роботи об’єктів приладобудування у воєнний час.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи досліджено питання актуальності нейронних мереж, їх застосування та впливу в найважливіших сферах життєдіяльності людського суспільства. Виконано огляд тенденцій розвитку предметно області та аналіз сучасних методів розробки нейронних мереж на прикладі мови програмування JavaScript.

Була досягнута мета роботи, що полягала у аналізі наявних на актуальний момент часу методів розробки нейронних мереж, визначенні можливості їх застосування при створенні нейронної мережі на мові програмування JavaScript, розгляді переваг та недоліків кожного методу, а також переваг та обмежень фреймворків JS, що використовуються для реалізації задачі розробки та навчання нейронних мереж.

В першому розділі кваліфікаційної роботи виконано завдання з дослідження предметної області визначеної тематики в контексті розкриття актуальності розробки нейронних мереж та тенденцій їх розвитку.

Детально розглянуто сфери застосування нейронних мереж та представлено приклади успішного вирішення поставлених перед ними задач. Розкрито принцип роботи нейронних мереж в кожному з представлених напрямків застосування технології.

Розглянуто вплив на соціальне становище людини в контексті працевлаштування. Виконано аналіз впливу нейронних мереж на життєдіяльність людини, її ментальність, висвітлено ризики застосування технології, переваги та недоліки.

В другому розділі кваліфікаційної роботи виконано огляд відомих методів розробки нейронних мереж. Ведеться опис застосування кожного методу, їх характеристик, особливостей моделі. Розкривається питання машинного навчання, методів навчання, алгоритмів.

Наведено обґрунтування використання мови програмування JavaScript для розробки нейронних мереж, розгляд фреймворків, які можуть бути застосовані для їх створення. В ході аналізу виділяються особливості кожного фреймворка,

порівняльний аналіз їх атрибутів, можливостей. Сформовано підсумок за напрямками застосування кожного з фреймворків.

В третьому розділі кваліфікаційної роботи описано вимоги до розробки нейронної мережі на основі використання одного з вказаних фреймворків JavaScript.

Створено програмного рішення із застосуванням нейронної мережі та висвітлено модель її навчання.

В розділі «Охорона праці та безпека в надзвичайних ситуаціях» описані основні аспекти охорони праці в предметній області: описано вимоги ергономіки робочого місця, оформлення робочого кабінету та необхідних умов для роботи за ПК. Описані фактори та умови, які позитивно та негативно впливають на людину при роботі. Також висвітлені теми вимог безпеки та проведення профілактичних медичних оглядів для операторів ПК.

Розглянуто способи виконання завдання з підвищення стійкості роботи об’єктів приладобудування у воєнний час.

ПЕРЕЛІК ДЖЕРЕЛ

1. ШІ назавжди: роль машинного навчання у реагуванні на гуманітарні кризи. [Електронний ресурс] // AIBusiness – 2022. – Режим доступу: https://aibusiness.com/ml/ai-for-good-the-role-of-machine-learning-in-responding-to-humanitarian-crises

2. Фабіо Ругге. ШІ в епоху кібербезладу. Актори, тенденції та перспективи / Фабіо Ругге. – Мілан: Ledizioni, 2020. – 5 с.

3. Gas Consumption Forecasting Using Machine Learning Methods and Taking into Account Climatic Indicators. Shymchuk, G., Lytvynenko, I., Hromyak, R., Lytvynenko, S., Hotovych, V. The 1st International Workshop on Computer Information Technologies in Industry 4.0, CITI 2023. Ternopil 14 -16 June 2023. Vol. 3468, pp. 156-163. URL: https://ceur-ws.org/Vol-3468/short8.pdf

4. Simulation of cyclic signals (generalized approach). Lupenko, S., Lytvynenko, I., Hotovych, V. 4th International Conference on Informatics and Data-Driven Medicine, IDDM 2021. Valencia. 19 November 2021. CEUR Workshop Proceedings. Vol. 3038, P. 86-92. ISSN 1613-0073 URL: https://ceur-ws.org/Vol-3038/short2.pdf

5. Lupenko S., Lytvynenko Ia., Hotovych V., Zozulia A., Chizoba N., Volyanyk O. (2021) Concept of design, requirements and generalized architectures of components of the integrated onto-oriented information environment of simulation and processing of cyclic signals. Scientific Journal of TNTU (Tern.), vol 102, no 2, pp. 147–160.

6. Volodymyr Gotovych, Oleg Nazarevych, Leonid Shcherbak (2018) Mathematical modeling of the regular-mode electric power supply and electric power consumption processes of the organization. Scientific Journal of TNTU (Tern), vol 91, No 3, pp. 134–142. URL: https://doi.org/10.33108/visnyk_tntu2018.03.134

7. Штучні нейронні мережі: перспективи використання в правоохоронній діяльності. [Електронний ресурс] // Маєтний М. І. – 2021. – Режим доступу до ресурсу: http://surl.li/fcyiy

8. FacePlusPlus. [Електронний ресурс] // MEGVII – 2023. – Режим доступу до ресурсу: https://www.faceplusplus.com.cn/

9. Y. Tiumentsev, M. Egorchev. Neural Network Modeling and Identification of Dynamical Systems. / Y. Tiumentsev, M. Egorchev. – Cambridge: Academic Press, 2019. – 4 p.

10. Використання нейронних мереж для прогнозування у фінансовій сфері. [Електронний ресурс] // Мозолевська М. О., Ставицький О. В. – 2017. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/22609/1/2017-11_5-07.pdf

11. Mingxi Wu. Intelligent Warfare: Prospects Of Military Development In The Age of AI. / Mingxi Wu. – London: Routledge, 2023. – 201 p.

12. Ефективність використання штучних нейронних мереж в економіці. [Електронний ресурс] // Василенко О. О., Іващенко Р. М., Бурлєєв О. Л. – 2021. – Режим доступу: http://surl.li/nbluh

13. Що таке нейронна мережа: простими словами. [Електронний ресурс] // FutureNow. – 2023. – Режим доступу: https://futurenow.com.ua/shho-take-nejronna-merezha-prostymy-slovamy/

14. Готович В.А., Мачужак А.В. Застосування методології CI/CD для автоматизації процесів тестування та розгортання програмного забезпечення. Матеріали XI Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 7-8 грудня 2022 року. С.131-132.

15. Peter Lee, Carey Goldberg, Isaac Kohane. The AI Revolution in Medicine: GPT-4 and Beyond. / Peter Lee, Carey Goldberg, Isaac Kohane. – London: Pearson, 2023. – 304 p.

16. Використання нейронних мереж – перспективна сфера науки і суспільства. [Електронний ресурс] // Neasmo. – 2021. – Режим доступу: http://oldconf.neasmo.org.ua/node/139

17. Кнаппертсбуш І., Гондлах К. Робота і AI 2030. Виклики та стратегії для завтрашньої роботи. / Кнаппертсбуш І., Гондлах К. – Франкфурт: Springer, 2023. – 4 с.

18. Chat-GPT. Chat-GPT & AI Technology Application for Students. / Chat-GPT. Ver: Artificial Intelligence Advisor, 2023. – 8 p.

19. Mike McGrath. HTML, CSS and JavaScript in Easy Steps. / Mike McGrath. – Royal-Leamington-Spa: In Easy Steps, 2020. – 10 p.

20. John Dean. Web Programming with HTML5, CSS, and JavaScript. / John Dean. – Boston: Jones & Bartlett Publishers, 2019. – 20 p.

21. Бібліотека Brain.js – [Електронний ресурс] // W3SchoolsUA. – 2023. – Режим доступу: https://w3schoolsua.github.io/ai/ai_brainjs.html#gsc.tab=0

22. Ekaba Bisong. Building Machine Learning and Deep Learning Models on Google Cloud Platform with JavaScript. / Ekaba Bisong: – New-York: Apress, 2019. – 4 p.

23. Igor Farkaš, Paolo Masulli, Stefan Wermter. Artificial Neural Networks and Machine Learning – ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part I. / Igor Farkaš, Paolo Masulli, Stefan Wermter – Frankfurt: Springer, 2020. – 103 p.

24. Igor Farkaš, Paolo Masulli, Stefan Wermter. Artificial Neural Networks and Machine Learning – ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part II. / Igor Farkaš, Paolo Masulli, Stefan Wermter – Frankfurt: Springer, 2020. – 48 p.

25. Patel Ankur A. Praxisbuch Unsupervised Learning: Machine-Learning-Anwendungen für ungelabelte Daten mit Python programmieren. / Patel Ankur A. – Sebastopol: O’Reilly Media, 2020. – 10 p.

26. Ronald T. Kneusel. How AI Works From Sorcery to Science by Ronald T. Kneusel Chapter 2. / Ronald T. Kneusel. – San-Francisco: No Starch Press, 2023. – 8 p.

27. Rajalingappaa Shanmugamani, Abdul Ghani Abdul Rahman, Stephen Maurice Moore, Nishanth Koganti. Deep Learning for Computer Vision. / Rajalingappaa Shanmugamani, Abdul Ghani Abdul Rahman, Stephen Maurice Moore, Nishanth Koganti. – Bermingem: Packt Publishing, 2018 – 23 p.

28. James McCaffrey. Neural Networks with JavaScript Succinctly. / James McCaffrey. Morrisville: Syncfusion, 2019. – 8 p.

29. Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. / Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. – Kitchener: D2L, 2021 – 12 p.

30. David Foster. Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play. / David Foster – Sebastopol: O'Reilly Media, 2019 – 9 p.

31. Rafael Valle. Hands-On Generative Adversarial Networks with Keras. / Rafael Valle. – Bermingem: Packt Publishing, 2019 – 22 p.

32. Marc Peter Deisenroth, A. Aldo Faisal та Cheng Soon Ong. Mathematics for Machine Learning. / Marc Peter Deisenroth, A. Aldo Faisal та Cheng Soon Ong. – Cambridge: Cambridge University Press, 2019. – 5 p.

33. Delip Rao, Brian McMahan. Natural Language Processing with PyTorch. / Delip Rao, Brian McMahan. – Sebastopol: O'Reilly Media, 2019. – 11 с.

34. Sebastian Raschka, Vahid Mirjalili. Machine Learning mit Python und Keras, TensorFlow 2 und Scikit-Learn. / Sebastian Raschka, Vahid Mirjalili. – Cambridge: MIT Press, 2021. – 11 p.

35. Маккафрі Д. Нейронні мережі з JavaScript. Коротко. – М.: Syncfusion, 2019. – 3 с.

36. Що таке нейронна мережа? [Електронний ресурс] – Режим доступу: https://aws.amazon.com/ru/what-is/neural-network/

37. Кравченко В. Що таке нейронні мережі та як вони працюють? [Електронний ресурс] – Режим доступу: http://surl.li/fcypt

38. Gant Laborde. Practical TensorFlow.js: Deep Learning in Web App Development. Gant Laborde. – New-York: Apress, 2021 – 15 p.

39. Juan De Dios Santos Rivera. Learning Tensorflow.js: Powerful Machine Learning in JavaScript. / Juan De Dios Santos Rivera. – Sebastopol: O'Reilly Media, 2020. – 31 p.

40. Practical Machine Learning in JavaScript: TensorFlow.js for Web Developers. / Charlie Gerard. – New-York: Apress, 2020 – 12 p.

41. Чень Хуіань. Підручник із веб-дизайну JavaScript і застосування штучного інтелекту TensorFlow.js. / Чень Хуіань. – Шанхай: Qi Feng, 2020. – 5 с.

42. Shanqing Cai, Stan Bileschi, Eric Nielsen. Deep Learning with JavaScript: Neural networks in TensorFlow.js. / Shanqing Cai, Stan Bileschi, Eric Nielsen. – New-York: Manning Publications, 2020. – 15 p.

43. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. / Aurélien Géron. – Sebastopol: O'Reilly Media, Sebastopol, 2019. – 17 p.

44. Jorge D. Rios, Alma Y. Alanis, Nancy Arana-Daniel, Carlos Lopez-Franco. Neural Networks Modeling and Control: Applications for Unknown Nonlinear Delayed Systems in Discrete Time. / Jorge D. Rios, Alma Y. Alanis, Nancy Arana-Daniel, Carlos Lopez-Franco – Cambridge: Academic Press, 2020. – 18 p.

45. Проблема зникання градієнту [Електронний ресурс] // Wikipedia. – 2023. – Режим доступу до ресурсу: https://cutt.ly/IwDgUkbY

46. Градієнтний спуск [Електронний ресурс] // robot_dreams. – 2023. – Режим доступу до ресурсу: https://robotdreams.cc/uk/blog/331-gradiyentniy-spusk-algoritm-ta-priklad-na-python

47. Transfer Learning [Електронний ресурс] // brain_leo – 2020. – Режим доступу до ресурсу: https://habr.com/ru/articles/526786/

48. Умови праці працівників, які використовують у роботі персональні комп'ютери [Електронний ресурс] // Золочів.нет. – 2019. – Режим доступу до ресурсу: https://zolochiv.net/umovy-pratsi-pratsivnykiv-iaki-vykorystovuiut-u-roboti-personal-ni-komp-iutery/

49. Робота за комп'ютером, наслідки та поради [Електронний ресурс] // АПАУ. – 2021. – Режим доступу до ресурсу: https://cutt.ly/twDgKv06

50. Організація робрчого місця оператора з обробки інформації та програмного забезпечення [Електронний ресурс] // Сонгрова О. В. – 2021. – Режим доступу до ресурсу: https://naurok.com.ua/organizaciya-robrchogo-miscya-operatora-z-obrobki-informaci-ta-programnogo-zabezpechennya-249780.html

51. Організація робочого місця оператора ПК [Електронний ресурс] // Studcon. – 2021. – Режим доступу до ресурсу: http://studcon.org/organizaciya-robochogo-miscya-operatora-pk

52. Охорона праці в офісі. Вимоги до робочого місця офісного працівника [Електронний ресурс] // AGN International. – 2021. – Режим доступу до ресурсу: https://gc.ua/uk/oxorona-praci-v-ofisi-vimogi-do-robochogo-miscya-ofisnogo-pracivnika/

53. Указ президента України 479/2021 [Електронний ресурс] // Зеленський В. – 2021. – Режим доступу до ресурсу: https://www.president.gov.ua/documents/4792021-40181

54. Маскування військ та об’єктів [Електронний ресурс] // G7 Сиили ТрО ЗСУ. – 2022. – Режим доступу до ресурсу: https://sprotyvg7.com.ua/lesson/maskuvannya-vijsk-ta-obyektiv

55. Указ президента України [Електронний ресурс] // Зеленський В. – 2021. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/479/2021#Text

ДОДАТКИ

Додаток А

Тези конференції

Міністерство освіти і науки України,
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VI МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

27-28 квітня 2023 р.

(збірник тез конференції)

Тернопіль 2023

УДК 004.4

Сербін В. – ст. гр. СТм-51
Тернопільський національний технічний університет імені Івана Пулюя

РОЛЬ НЕЙРОННИХ МЕРЕЖ У РЕГУЛЮВАННІ ГУМАНІТАРНИХ КРИЗ

Serbin V.
Ternopil Ivan Puluj National Technical University

THE ROLE OF NEUTRAL NETWORKS IN THE REGULATION OF HUMANITARIAN CRISES

Ключові слова: нейронні мережі, штучний інтелект, системи підтримки прийняття рішень, гуманітарна криза.

Keywords: neural networks, artificial intelligence, decision support systems, humanitarian crisis.

На сьогоднішній день нейронні мережі представляють собою уособлення інструменту, який може вирішити значну кількість технічно складних та різнонаправлених завдань: від розв'язку простих математичних алгоритмів чи формування частин програмного коду і до виконання функції захисту інформаційних систем від хакерських атак. Від недавна нейронні мережі активно застосовують для створення об'єктів творчості: музики, коротких відео чи картин. Також штучний інтелект допомагає вирішувати складні економічні питання, формувати відносно успішні бізнес-плани. Навіть правоохоронні органи США використовують нейромережі з метою запобігання злочинності, а державні установи КНР – задля контролю над високопосадовцями та виявлення серед них корупціонерів.

Як демонструють приклади, спектр використання дійсно широкий, однак, чи здатне людство використати цей інструмент для покращення методів з надання допомоги при ліквідації гуманітарних катастроф? Дані ООН та різних дослідницьких організацій останніх років свідчать про те, що людство стикається з демографічними, кліматичними та екологічними проблемами великого масштабу, як ніколи раніше. На щастя, на сьогоднішній день вже існують нові програмні рішення, які можуть допомогти подолати ці незаплановані та часто катастрофічні події. Вони побудовані на системах штучного інтелекту, які можуть зрозуміти надто складні для людини шаблони даних.

Наприклад, глобальне потепління є досить серйозною проблемою останніх десятиліть, оскільки є неконтрольованим процесом і може спричинити катастрофічні наслідки для планети. Тому такі організації, як Nvidia, активно долучаються до вирішення кліматичної проблеми шляхом створення нового суперкомп'ютера зі штучним інтелектом під назвою «Earth-2». Його метою буде створення цифрового двійника планети Земля для допомогти світовим лідерам передбачити та пом'якшити наслідки шкідливого впливу зміни клімату.

NetHope, технологічний консорціум з майже 60 провідних некомерційних організацій світу, є ще однією організацією, яка вивчає вплив штучного інтелекту на гуманітарний сектор. Вона змогла визначити широкий спектр переваг нейронних мереж, у тому числі здатність охоплювати більше людей послугами та важливою інформацією. Також була відзначена здатність до передбачення надзвичайних ситуацій перед їх виникненням та поширенням, приймати швидкі рішення та покращувати результати.

NetHope продемонструвала кілька прикладів користі технології навіть на ранніх етапах її розвитку та впровадження. Для прикладу, Датська рада у справах біженців використовує штучний інтелект та відкриті дані для прогнозування вимушеного переміщення в таких місцях, як Буркіна-Фасо, Малі та Нігерія [1]. Тоді як чат-бот Норвезької ради з питань біженців допомагає венесуельським мігрантам у Колумбії вивчати сучасну імміграційну політику та закони.

Також, у 2017 році ООН запустила проект, що використовує технологію «Великих даних» у зв'язці з нейронними мережами для покращення процесу прийняття рішень: використовуючи машинне навчання та загальнодоступні дані з Twitter, організація хоче допомогти розробити інституційну політику проти ксенофобії, дискримінації та расизму щодо мігрантів і біженців [1].

Використання машинного навчання для пошуку гуманітарних знань (knowledge) є добре відомою практикою, у якій моніторинг платформ соціальних мереж здійснюється для отримання, кластеризації, анотування та ранжування актуальної інформації про поточні кризи. У цих програмах машинне навчання використовується для класифікації вмісту соціальних мереж на основі терміновості, важливості, серйозності тощо. На основі цих класифікацій планується та виконується реакція осіб, які приймають гуманітарні рішення.

Системи підтримки прийняття рішень можна використовувати і для визначення пріоритетів розподілу ресурсів на основі попиту та фактичних потреб постраждалого населення на місцях виникнення кризових ситуацій. Ці системи на базі нейронних мереж надають обґрунтовану інформацію тим організаціям та особам, які приймають рішення в ситуації гуманітарних криз щодо розподілу ресурсів.

Збалансований розподіл рятує життя та ресурси, оскільки дозволяє тим, хто вкрай їх потребує, мати пріоритетний доступ до необхідної матеріальної допомоги, і водночас захищає ці предмети від марнотратства та розподілу не тим одержувачам, у яких, насправді, може бути відсутня гостра необхідність в них [2].

Одним з таких інструментів є програма товариств Червоного Хреста та Червоного Півмісяця з швидкого розподілу гуманітарних ресурсів та фінансів на основі прогнозування для попередньої реалізації заходів. Ця програма використовує різні джерела даних, такі як метеорологічні дані та аналіз ринку, щоб визначити, коли і куди слід розподіляти гуманітарні ресурси [3].

Залежно від вибраних моделей, архітектури та програмної реалізації нейронні мережі можуть виконувати значну допоміжну функцію у вирішенні чи попередженні певного роду гуманітарних проблем та криз, однак, результати обчислень штучного інтелекту не рекомендується вважати виключно вірним рішенням, оскільки набір алгоритмів навіть з можливістю навчання не може врахувати фактор контексту поданої інформації.

Література:

1. ШІ назавжди: роль машинного навчання у реагуванні на гуманітарні кризи [Електронний ресурс] – Режим доступу: https://aibusiness.com/ml/ai-for-good-the-role-of-machine-learning-in-responding-to-humanitarian-crises.
2. SEmHuS: семантично вбудований гуманітарний простір [Електронний ресурс] – Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9990040/.
3. Використання потенціалу штучного інтелекту для гуманітарної діяльності: можливості та ризики [Електронний ресурс] – Режим доступу: https://international-review.icrc.org/articles/harnessing-the-potential-of-artificial-intelligence-for-humanitarian-action-919.

Додаток Б

Тези конференції

Тернопільський національний технічний університет імені Івана Пулюя, Україна
Львівський національний університет імені Івана Франка, Україна
Вінницький національний аграрний університет, Україна
Харківський національний економічний університет імені Семена Кузнеця, Україна
Донецький національний університет імені Василя Стуса, Україна
Тернопільський національний педагогічний університет імені Володимира Гнатюка, Україна
Краківський аграрний університет, Польща
Університет Вища Школа Бізнесу в Домброві-Гурничі, Польща

ТЕЗИ ДОПОВІДЕЙ

IV міжнародної науково-практичної конференції учених та студентів
«ЦИФРОВА ЕКОНОМІКА ЯК ФАКТОР ІННОВАЦІЙ ТА СТАЛОГО РОЗВИТКУ СУСПІЛЬСТВА»

7-8 грудня 2023року

ТЕРНОПІЛЬ, УКРАЇНА 2023

УДК 004.8, 339.9
В. Сербін, студент групи СТм-61
Тернопільський національний технічний університет імені Івана Пулюя, Україна
Науковий керівник:Д. Дмитрів, канд.техн.наук, доц. Д.В.

«ШОВКОВИЙ ШЛЯХ» ТА РОЛЬ НЕЙРОННИХ МЕРЕЖ В ПРОКЛАДАННІ ОПТИМАЛЬНИХ МАРШРУТІВ

V.Serbin
Ternopil Ivan Puluj National Technical University, Ukraine
Supervisor: D.Dmytriv, Ph.D, Assos Prof.

THE SILK ROAD AND THE ROLE OF NEURAL NETWORKS IN THE CONSTRUCTION OF OPTIMAL ROUTES

Шовковий шлях є однією з найбільших транспортних магістралей, що з'єднує Європу та Азію. Стародавнім торговим шляхом великі цивілізації Риму та Китаю могли виконувати обмін товарами, ідеями та культурою: саме через стародавній Шовковий шлях до Китаю прийшли християнство, буддизм, а до Заходу – шовк.

Ця стародавня дорога стала поштовхом для Організації Об'єднаних Націй щодо побудови нової трансазіатської магістралі. В цей час Економічно-соціальна комісія ООН для Азії та Тихого океану (UNESCAP) запропонувала залізничний аналог цієї дороги [1].

Протягом десятиліть планувань, будівництва магістралі та змін економіко-політичної ситуації у проміжних державах, напрямок дороги «Нового Шовкового шляху» зазнавав змін. На рисунку 1 відображені запропоновані в різні періоди часу варіації Нового Шовкового шляху на фоні стародавньої дороги.

Рис. 1 – Проекти «Шовкового шляху»

В останній варіації Нового Шовкового шляху він оминає Україну. З економічної точки зору це є невигідним рішенням, однак, для формування точних висновків звернемось до практики.

Опираючись на аналітичні дані логістичних компаній, у 2021 році транзит через Україну становив лише 2% обсягів контейнерних перевезень у західному напрямку на Новому Шовковому шляху, що є дуже незначною часткою загального обсягу залізничних вантажних перевезень між Китаєм і Європою. Це пояснюється застарілою інфраструктурою,

що не відповідає стандартам транспортних шляхів Європи та Азії. Міністр інфраструктури України, Олександр Курбаков, так коментував стан справ: «Наразі ми не можемо повною мірою використовувати фактор нашого географічного розташування, тому що наша залізнична та портова інфраструктура все ще застаріла» [2]. Після перенесення бойових дій зі східного регіону України на всю її територію, використання транспортної інфраструктури ще більше ускладнилось, а транзитні можливості держави значною мірою погіршились.

Навіть з урахуванням всіх недоліків, масштаб транспортної інфраструктури України та її транзитні можливості здатні забезпечити значний потік товарів. Так, станом на 2021 рік обсяги перевезення вантажів в Україні становили 619,9 млн. тонн, з яких 314,3 млн. припадали на автомобільний транспорт, а 222,6 млн. тонн – залізничний транспорт [3].

Згідно даних Міністерства фінансів України, в тому ж 2021-му році доходи державного бюджету України за статтею «Податки на міжнародну торгівлю та зовнішні операції» склали 38177,2 млн. грн., з яких:

- ввізне мито – 36854,9 млн. грн.;
- вивізне мито – 1322,3 млн. грн.

Такий результат складає 2.94% від загальних доходів державного бюджету [4].

За платежами, сплату яких до бюджету контролює Державна митна служба, надходження становили 409,3 млрд грн, в тому числі 380,3 млрд грн у вигляді податку на додану вартість та 26,6 млрд грн у вигляді ввізного мита [5]. Ці платежі відображають обсяг торгівлі з іншими країнами, включаючи ті, що входять до Шовкового шляху.

За даними Державної служби статистики України, у 2021 році обсяг транспортних послуг, що надаються українськими перевізниками, становив 223,8 млрд грн, що на 16,9% більше, ніж у 2020 році. З них 56,7 млрд грн припадало на зовнішні перевезення, а 167,1 млрд грн на внутрішні. За видами транспорту, найбільший обсяг послуг надавався автомобільним транспортом (97,4 млрд грн), залізничним транспортом (67,2 млрд грн) та трубопровідним транспортом (35,5 млрд грн). Морський транспорт надав послуг на суму 10,9 млрд грн, повітряний транспорт – на 9,7 млрд грн, а річковий транспорт – на 3,1 млрд грн.

За даними Міністерства інфраструктури України, у 2021 році Україна збільшила обсяг перевезень вантажів по Шовковому шляху на 20% у порівнянні з 2020 роком. Зокрема, за даними Укрзалізниці, у 2021 році було перевезено 1 200 контейнерів з Китаю до Європи через територію України, що на 50% більше, ніж у 2020 році. Також було здійснено перший експортний рейс з України до Китаю з 41 контейнером зерна.

Отже, Україна і справді має потенціал для налагодження потужної транспортної логістики та наповнення власного бюджету за рахунок транзиту чи експорту товарів. Однак, для вирішення поточних проблем з інфраструктурою: її планування, побудови чи перебудови та налагодження, необхідно використати всі наявні та актуальні в час цифрових технологій ресурси.

Не останню роль у побудові транспортних шляхів та інфраструктури відіграють нейронні мережі. З допомогою можливості швидкого аналізу великих масивів даних нейронні мережі здатні в короткі терміни проаналізувати ті чи інші аспекти проекту та видати результат у вигляді аналітики. При будівництві автомагістралей нейронні мережі здатні розраховувати витрати на будівельні матеріали, робочу силу, обладнання. Результати використання різних моделей демонструють, що вони здатні відтворити як минулі тенденції витрат на будівництво з розумною точністю, так і оцінити майбутні витрати. Наприклад, при будівництві шосе в штаті Луїзіана, США, нейронні мережі змогли спрогнозувати зростання ціни будівництва за 17 років вдвічі [6].

Окрім того, доцільно використовувати нейронні мережі у виборі оптимальних маршрутів перевезення вантажів та пасажирів територією України в контексті входження у транс-європейські коридори та напрямки «Шовкового шляху». На даний момент потужності нейронних мереж можна використовувати з метою швидкого перенаправлення торгівельних потоків іншими внутрішніми та міждержавними напрямками, наприклад, для вирішення

питання систематичного перекриття кордону зі сторони Польщі. Нейронні мережі дають можливість виконувати короткострокове прогнозування транспортних потоків та віртуальне моделювання магістралей на основі макродинамічної моделі транспортного потоку. Нейронні мережі можуть аналізувати історичні дані трафіку, беручи до уваги різні фактори, такі як погода, час доби та особливі події, щоб прогнозувати затори. Потім ці прогнози можна використовувати для планування альтернативних маршрутів, щоб уникнути інтенсивного руху. Також, нейронні мережі можна використовувати для виявлення аномалій у моделях трафіку, оптимізовуючи рух транспорту.

Література

1. The Editors of Encyclopedia Britannica. Silk Road. URL: https://www.britannica.com/money/topic/Silk-Road-trade-route
2. Majorie van Leijen. How important is Ukraine on the New Silk Road? URL: https://www.railfreight.com/specials/2022/02/25/how-important-is-ukraine-on-the-new-silk-road/?gdpr=deny
3. DIA. Cargo transportation in Ukraine has halved in 2022. URL: https://dia.dp.gov.ua/en/cargo-transportation-in-ukraine-has-halved-in-2022/
4. Мінфін. Доходи держбюджету України. URL: https://index.minfin.com.ua/ua/finance/budget/gov/income/2021/
5. Міністерство фінансів України. URL: https://mof.gov.ua/
6. Chester G. Wilmot, Bing Mei. Neural Network Modeling of Highway Construction Costs. URL: https://www.researchgate.net/publication/239386874_Neural_Network_Modeling_of_Highway_Construction_Costs

Додаток В

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

МАТЕРІАЛИ

X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

7–8 грудня 2022 року

ТЕРНОПІЛЬ
2022

УДК 004.4
В. Сербін
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОЇ ОБРОБКИ ДАНИХ В МОВІ ПРОГРАМУВАННЯ JAVASCRIPT

UDC 004.4
V. Serbin

IMPLEMENTATION OF PARALLEL DATA PROCESSING IN JAVASCRIPT PROGRAMMING LANGUAGE

Станом на сьогодні, мова програмування JavaScript більше не обмежується браузером, а використовується для створення програмного забезпечення у будь-якому пристрої, від серверів до пристроїв Інтернету речей. Багато з написаних програм є важкими і можуть отримати значне підвищення продуктивності від використання паралельних обчислень. В загальному, причина розпаралелювати JS така ж, як і причина розпаралелювати будь-яку іншу мову: закон Мура вмирає, а багатоядерна архітектура захоплює світ [1].

Використання API Web Workers, ймовірно, є єдиним способом досягти справжньої мультипроцесорності в Javascript [2]. Web Worker дозволяють користувачеві паралельно запускати JavaScript без втручання в інтерфейс користувача. Робочий сценарій буде завантажено та запущено у фоновому режимі, повністю незалежно від сценаріїв інтерфейсу користувача. Це означає, що Workers не мають доступу до елементів інтерфейсу користувача, таких як DOM і поширених функцій JS, як-от getElementById (але можуть здійснювати виклики AJAX). Основний варіант використання API Web Worker полягає у виконанні обчислювально дорогих завдань у фоновому режимі, без процесу переривання або переривання взаємодії користувача [1].

Щоб реалізувати Worker, необхідно створити його екземпляр за допомогою коду, який він запускатиме [2]:

```
const worker = new Worker(«worker.js»);
```

Web Workers спілкуються з основним документом/основним потоком через техніку передачі повідомлень. Передача повідомлень здійснюється за допомогою API postMessage [1]:

```
worker.postMessage(num);
```

Згідно зі специфікацією існує два типи Web Worker: спільні та виділені.

Web Worker за замовчуванням називається виділеним. Цей тип робочого файлу доступний лише зі сценарію, який його створив, тоді як спільний робочий файл можна отримати з кількох сценаріїв.

Принцип взаємодії Web Workers та основного потоку наведено на рисунку 1.

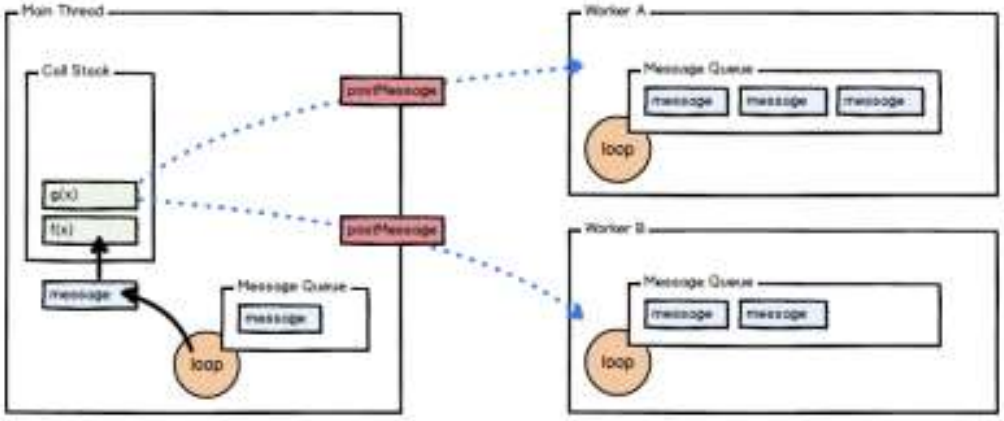

Рисунок 1. Схема взаємодії Web Workers та основного потоку

Для спільного Web Worker потрібен інший конструктор: SharedWorker. Спільний робочий файл доступний кількома сценаріями, навіть якщо до них звертаються різні вікна, iframe або навіть робочі.

Прослуховування події відбувається через обробник «onmessage()», в якому e.data міститиме передане значення

Цей процес працює в обох напрямках, тож можна повернути деякі дані назад із робочого коду у основний потік [2]. Для цього використаємо у коді worker-у «postMessage()» з результатом, а у основному потоці використовуємо «worker.onmessage» для прийому даних.

Отже, потоки Web Worker допомагають нам розвантажити інтенсивні завдання ЦП з циклу подій, щоб виконувати їх паралельно без блокування. Робочий потік виконує частину коду відповідно до вказівок батьківського потоку окремо від батьківського та інших робочих потоків. Кожен робочий потік має власне ізольоване середовище, цикл подій, чергу подій тощо. Робочий і батьківський потік можуть спілкуватися один з одним через канал обміну повідомленнями, а також робочі потоки дають нам можливість запускати кілька потоків в одному процесі [3]. Крім утримання циклу подій від виконання трудомістких операцій ЦП можемо використовувати пул робочих потоків для розділення та паралельного виконання важких операцій ЦП з ціллю підвищення продуктивності нашої програми [3].

Література

1. Розпаралелювання JavaScript для розваги та прибутку. URL: https://www.codementor.io/@madhugnadig/parallelizing-javascript-for-fun-and-profit-naxmo4lam.
2. Паралельна обробка в JS. URL: https://advancedweb.hu/parallel-processing-in-js/.
3. Паралельна обробка в Node.js з використанням робочих потоків. URL: https://deepsource.io/blog/nodejs-worker-threads/.