Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

«Допущено до захисту»
В.о. завідувача кафедри БІСТ
Мелкозьорова О.М.
_____
« » 2024 р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Перспективи використання штучного інтелекту у кібербезпеці»

Оцінка « »
Голова ЕК
Лемешко О.В. _____

Керівник к.т.н. Мелкозьорова О.М.

Рецензент д.т.н., Краснобаєв В.А.

Виконавець: студент групи КБ-42

Перевозник Д.Ф.

Харків – 2024

РЕФЕРАТ

Дипломна робота присвячена розробці засобів захисту системи аутентифікації на основі аналізу. Робота складається зі вступу, трьох розділів, висновків до кожного розділу, переліку посилань та додатків. Загальний обсяг основного тексту становить 43 сторінок, включаючи 17 рисунків та 2 таблиці. Додатково в роботі наведено 4 додатки. Перелік посилань включає 12 найменувань. Загальний обсяг дипломної роботи становить 49 сторінок.

Актуальність теми дослідження полягає в збільшенні обсягів цифрової інформації та зростанні кількості користувачів інтернету, зростає і кількість кіберзагроз, які стають дедалі складнішими і витонченішими. Системи аутентифікації, що забезпечують доступ до інформаційних ресурсів, є ключовими елементами захисту даних, і їхній захист потребує постійного вдосконалення.

Метою дипломної роботи є дослідження сучасних методів і технологій штучного інтелекту, зокрема нейронних мереж, для захисту систем аутентифікації та підвищення рівня кібербезпеки.

Об’єктом дослідження дипломної роботи є системи аутентифікації та їхні засоби захисту від кіберзагроз з використанням технологій штучного інтелекту.

Предметом розробки є нейромережеві моделі та алгоритми машинного навчання, які застосовуються для підвищення ефективності та надійності систем аутентифікації у контексті кібербезпеки.

Методи дослідження: вимірювання ефективності нейромережевих моделей у виявленні та протидії кіберзагрозам, аналіз даних, моделювання та тестування розроблених систем.

Результатами проведеної роботи є розробка та впровадження ефективної нейромережевої моделі для захисту системи аутентифікації, яка демонструє високу точність і швидкість у виявленні спроб несанкціонованого доступу.

Ключові слова: НЕЙРОМЕРЕЖА, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, БЕЗПЕКА, НЕЙРОНИ, АУТЕНТИФІКАЦІЯ.

ABSTRACT

The thesis is devoted to the development of means of protection of the authentication system based on the analysis. The work consists of an introduction, three chapters, conclusions to each chapter, a list of references and appendices. The main body of the thesis comprises 43 pages, including 17 figures and 2 tables. Additionally, the thesis includes 4 appendixes. The list of references contains 12 entries. The total length of the diploma thesis is 49 pages.

The relevance of the research topic lies in the increase in the amount of digital information and the growth in the number of Internet users, and the number of cyber threats, which are becoming more and more complex and sophisticated, is also increasing. Authentication systems that provide access to information resources are key elements of data protection, and their protection requires constant improvement.

The aim of the thesis is to research modern methods and technologies of artificial intelligence, in particular neural networks, to protect authentication systems and increase the level of cyber security.

The object of research of the thesis is authentication systems and their means of protection against cyber threats using artificial intelligence technologies.

The subject of development is neural network models and machine learning algorithms, which are used to improve the efficiency and reliability of authentication systems in the context of cyber security.

Research methods: measuring the effectiveness of neural network models in detecting and countering cyber threats, data analysis, modeling and testing of developed systems.

The results of the work are the development and implementation of an effective neural network model to protect the authentication system, which demonstrates high accuracy and speed in detecting unauthorized access attempts.

Keywords: NEURAL NETWORK, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, SECURITY, NEURONS, AUTHENTICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІ	–	Штучний Інтелект
МН	–	Машинне Навчання
CNN	–	Convolutional Neural Network
RNN	–	Recurrent Neural Network
ANI	–	Artificial Narrow Intelligence
AGI	–	Artificial General Intelligence
ReLU	–	Rectified Linear Unit;
ГН	–	Глибинне Навчання
ANN	–	Artificial Neural Network
ШНМ	–	Штучні Нейронні Мережі
SOM	–	Self-organizing Map
CISO	–	Chief Information Security Officer
CEO	–	Chief Executive Officer
PDE	–	Processing Development Environment
GPU	–	Graphics Processing Unit
API	–	Application Programming Interface

ВСТУП

Захист інформаційних систем є однією з найважливіших задач сучасного суспільства, адже цифровізація охоплює всі аспекти життя – від особистих даних до критичних інфраструктур. За останні роки кількість кібератак значно зросла, причому методи, які використовують зловмисники, стають все складнішими і небезпечнішими. Традиційні методи кіберзахисту, які ґрунтуються на фільтрації трафіку та використанні сигнатур, часто не здатні ефективно протидіяти новітнім загрозам, які можуть змінювати свої характеристики в режимі реального часу.

Провідні компанії в галузі кібербезпеки активно використовують штучний інтелект (ШІ) для розробки новітніх захисних систем. Деякі, наприклад, спеціалізуються на виявленні та нейтралізації шкідливого ПЗ за допомогою глибокого навчання, тоді як інші зосереджується на аналізі поведінкових моделей для виявлення аномалій.

Аналіз вітчизняної та зарубіжної науково-технічної літератури показує, що існуючі методи кіберзахисту не завжди здатні впоратися з сучасними загрозами. Більшість традиційних методів, таких як фільтрація на основі сигнатур, не можуть виявити новітні та складні атаки, які використовують методи маскування та швидко змінюються. У цій галузі є значні прогалини, які можуть бути заповнені за допомогою ШІ, зокрема нейронних мереж та методів машинного навчання.

Захист інформаційних систем від кіберзагроз стає все більш актуальним питанням у світі, де цифрові технології відіграють ключову роль у всіх сферах життя. Зростання складності та частоти кібератак вимагає нових підходів до кібербезпеки. ШІ надає можливість створювати адаптивні системи, які можуть автоматично виявляти та реагувати на загрози в режимі реального часу. Це особливо важливо в умовах зростаючої складності та динамічності кіберзагроз, де швидкість реакції має вирішальне значення.

Використання ШІ у кібербезпеці дозволяє створювати системи, які здатні аналізувати великі обсяги даних, виявляти аномалії та прогнозувати можливі атаки. Це робить ШІ потужним інструментом у боротьбі з кіберзлочинністю та

підвищує загальний рівень безпеки інформаційних систем. Розробка ефективних методів та алгоритмів на основі ШІ є важливим напрямком досліджень, що має значний потенціал для впровадження в різних галузях.

Основною метою даного дослідження є аналіз можливостей застосування ШІ для забезпечення кібербезпеки та розробка методів і алгоритмів, які підвищують ефективність захисту інформаційних систем. Особливу увагу буде приділено розробці нейронних мереж для аналізу поведінкових даних користувачів і виявлення аномалій, що можуть свідчити про потенційні загрози.

Галузь застосування результатів дослідження охоплює різні аспекти кібербезпеки, включаючи виявлення та реагування на загрози, прогнозування кібератак та автоматизацію захисних заходів. Розроблені методи та алгоритми можуть бути використані в різних сферах, таких як фінансові установи, державні органи, підприємства та інші організації, які потребують надійного захисту своїх інформаційних систем.

Захист інформаційних систем від кібератак є критично важливим завданням у сучасному цифровому світі. Використання ШІ для вирішення цих проблем відкриває нові можливості для створення адаптивних та ефективних систем захисту. Дане дослідження спрямоване на розробку та впровадження методів та алгоритмів на основі ШІ для підвищення рівня кібербезпеки, що має велике значення для захисту інформаційних систем у різних галузях.

1 ТЕРМІНОЛОГІЧНИЙ БАЗИС ПРЕДМЕТНОЇ ОБЛАСТІ, ЩО РОЗГЛЯДАЄТЬСЯ

1.1 Ключові поняття штучного інтелекту

Штучний інтелект (ШІ) — це широка галузь комп'ютерної науки, що зосереджується на створенні систем, здатних виконувати завдання, які зазвичай вимагають людського інтелекту. До таких завдань належать навчання, розпізнавання мови, візуальне сприйняття, прийняття рішень, переклад мов та інші. Також, це галузь комп'ютерної науки, яка зосереджується на розробці алгоритмів та систем, що здатні виконувати інтелектуальні завдання.

Машинне навчання (МН) — підрозділ ШІ, що займається розробкою алгоритмів, які дозволяють комп'ютерним системам навчатися на основі даних та поліпшувати свою продуктивність з досвідом без явного програмування.

Глибинне навчання (ГН) — підмножина машинного навчання, яка використовує багатошарові нейронні мережі для аналізу та навчання з великих обсягів даних.

Нейронна мережа — обчислювальна модель, натхненна біологічними нейронами, що складається з вузлів (нейронів), з'єднаних між собою і здатних до адаптивного навчання.

Навчання з підкріпленням (НП) — тип машинного навчання, в якому агент навчається взаємодіяти з навколишнім середовищем, отримуючи винагороду або покарання за свої дії.

Обробка натуральної мови (ОНМ) — створення алгоритмів, що дозволяють комп'ютерам розуміти, інтерпретувати та генерувати людську мову.

1.2 Поняття навчання в штучному інтелекті

ШІ включає різні підтипи, серед інших: машинне навчання , комп’ютерне бачення , нечітка логіка і обробка природної мови. МН базується на алгоритмах, навчених для прийняття рішень, які навчаються на проаналізованих даних. Алгоритми МН можна класифікувати на основі типу отриманого зворотного

зв’язку. По-перше, це контрольоване навчання, яке отримує попередньо каталогізовані дані як вхідні дані. Ще одна категорія – навчання без нагляду. Різниця з попереднім полягає в тому, що навчальні дані не каталогізовані, і система повинна розпізнавати та маркувати той самий тип даних. У напівконтрольованому навчанні виконується комбінація двох попередніх алгоритмів. Таким чином, система повинна враховувати як позначені, так і непомічені елементи. Інший тип МН представлений навчанням з підкріпленням, яке може вчитися на своїх успіхах і помилках. Натомість глибоке навчання (ГН) — це підгрупа МН, заснована на алгоритмах, які використовують штучні нейронні мережі (ANN), організовані в кілька шарів, щоб імітувати те, як людський мозок інтерпретує інформацію та робить висновки з неї. ГН характеризується кількома прихованими шарами вузлів, які вивчають представлення даних, абстрагуючи їх різними способами.

Комп’ютерне бачення — це галузь штучного інтелекту, яка дозволяє комп’ютерам розпізнавати зображення та розрізняти окремі елементи зображення, приписуючи їм значення. Нечітка логіка використовує недвійкові значення для вирішення проблем, які потребують роботи з більшою кількістю значень, які класична логіка не може вирішити. Нарешті, обробка природної мови є підтипом ШІ, який намагається зрозуміти природну мову для спілкування між машинами та людьми.

Діаграма, що відображає взаємозв’язки між штучним інтелектом, машинним навчанням і глибоким навчанням наведена на рисунку 1.1 [1].

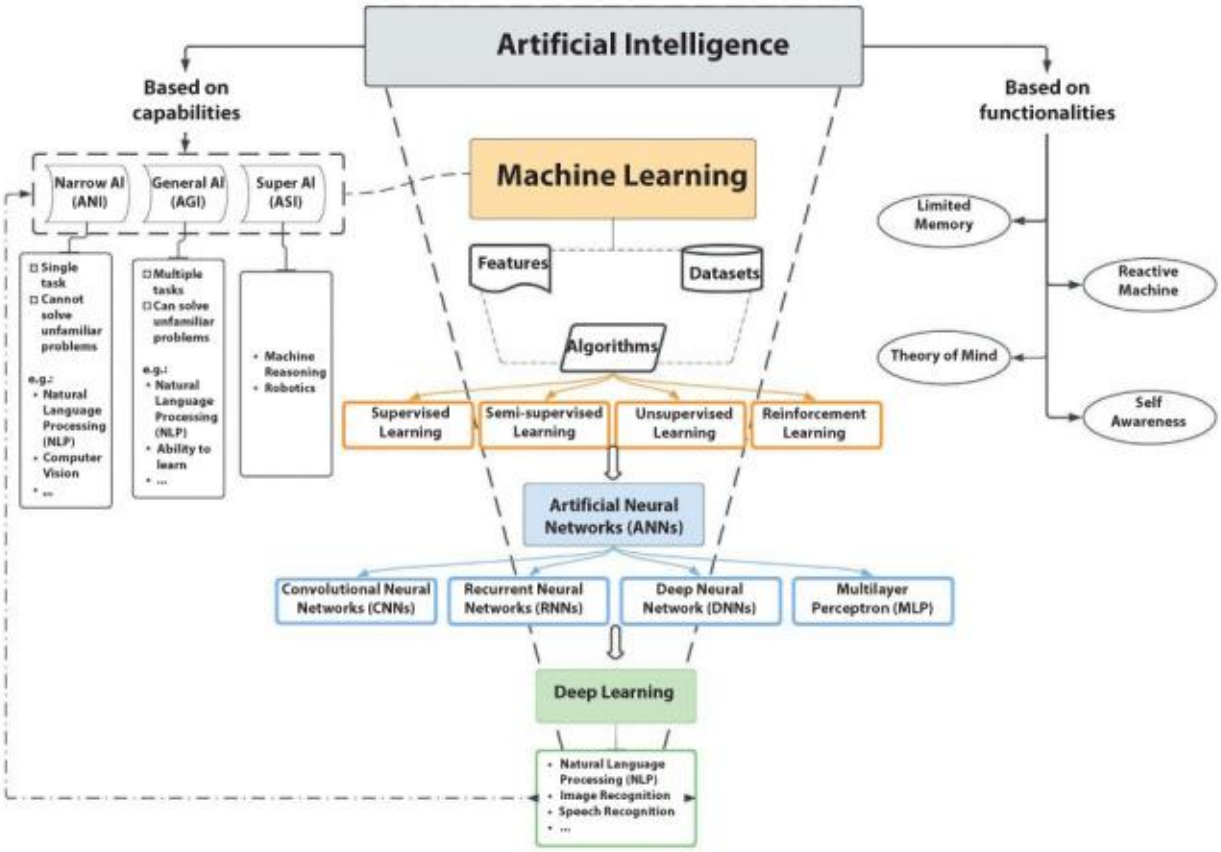

Рисунок 1.1 – Взаємозв’язки між штучним інтелектом, машинним навчанням і глибоким навчанням

1.3 Рівні інтелекту в ШІ

Існує кілька різних підходів до створення ШІ, зокрема:

- Правила та логіка: Перші системи ШІ базувалися на правилах і логічних висновках, що дозволяло їм виконувати прості завдання на основі заданих алгоритмів.
- Евристика: Системи, які використовують евристики, спираються на правила, що допомагають скоротити час на пошук рішення, але не завжди гарантують оптимальний результат.
- Машинне навчання: Найсучасніший підхід до ШІ, який включає алгоритми, здатні вчитися на основі даних. Цей підхід включає підмножини, такі як глибоке навчання та нейронні мережі.

Перші системи ШІ базувалися на жорстко запрограмованих правилах і логічних висновках. Такі системи можуть виконувати прості завдання,

дотримуючись заздалегідь визначених алгоритмів. Наприклад, експертні системи використовувалися для діагностики хвороб на основі набору правил, складених експертами в цій галузі. Хоча ці системи можуть бути ефективними для вирішення специфічних завдань, вони мають обмежену здатність адаптуватися до нових даних і ситуацій.

Евристики використовуються для скорочення часу на пошук рішення за рахунок використання наближених методів і правил, які базуються на попередньому досвіді. Цей підхід дозволяє швидше знаходити рішення, хоча і не завжди оптимальні. Евристики широко застосовуються в задачах оптимізації та прийняття рішень, де важливо знайти прийнятне рішення в обмежений час.

Машинне навчання є одним з найбільш сучасних і ефективних підходів до ШІ. Воно включає розробку алгоритмів, які дозволяють системам вчитися на основі даних. МН дозволяє комп'ютерам розпізнавати патерни і робити прогнози без необхідності жорсткого програмування. Алгоритми МН можуть бути застосовані в різних областях, включаючи розпізнавання образів, обробку природної мови, медичну діагностику та багато іншого.

ШІ можна класифікувати за різними критеріями, зокрема за рівнем його інтелектуальних можливостей та за типами алгоритмів, що використовуються.

Вузький або слабкий ШІ (Artificial Narrow Intelligence, ANI) спеціалізується на виконанні конкретних завдань і не має загальних інтелектуальних здібностей. Приклади ANI включають системи розпізнавання мови, чат-боти, рекомендаційні системи та інші спеціалізовані додатки. Вузький ШІ демонструє високий рівень ефективності в певних завданнях, але не може вийти за межі своєї спеціалізації.

Загальний або сильний ШІ (Artificial General Intelligence, AGI) прагне досягти рівня інтелекту, подібного до людського, з можливістю виконувати будь-яке завдання, що вимагає інтелекту. AGI має здатність до самонавчання і адаптації до нових ситуацій без необхідності перенавчання для кожного нового завдання. Хоча AGI є предметом інтенсивних досліджень, на даний момент такі системи ще не існують.

Суперінтелект (Artificial Superintelligence, ASI) є гіпотетичним рівнем ШІ, який перевершує людський інтелект у всіх аспектах. ASI має потенціал для здійснення революційних змін у всіх сферах життя, але також викликає занепокоєння щодо можливих ризиків та етичних питань.

ШІ демонструє високу ефективність у розпізнаванні образів, що дозволяє системам комп'ютерного зору ідентифікувати об'єкти на зображеннях і відео. Це знайшло застосування в медицині для діагностики хвороб на основі зображень, у безпекових системах для розпізнавання облич і в автономних транспортних засобах для навігації. Сучасні системи розпізнавання образів здатні аналізувати величезні обсяги даних і виділяти з них значущі патерни, що значно підвищує точність і швидкість діагностики [2].

Завдяки розвитку технологій обробки природної мови (NLP), ШІ здатний розуміти, інтерпретувати і генерувати людську мову. Це включає в себе створення чат-ботів, систем автоматичного перекладу, аналізу тексту і голосових асистентів, таких як Siri і Alexa. Ці технології дозволяють автоматизувати багато рутинних завдань, підвищуючи ефективність і зручність для користувачів.

Алгоритми ШІ використовуються для аналізу великих обсягів даних з метою виявлення тенденцій і прогнозування майбутніх подій. Це знаходить застосування у фінансових ринках, де ШІ допомагає прогнозувати цінові зміни, у маркетингу для аналізу поведінки споживачів та у промисловості для прогнозування технічних збоїв обладнання. Використання ШІ в аналітиці дозволяє компаніям приймати більш обґрунтовані рішення на основі точних даних і прогнозів.

Активне застосування ШІ у сфері кібербезпеки дозволяє ефективніше виявляти і реагувати на кіберзагрози. Системи, засновані на ШІ, можуть аналізувати мережевий трафік, виявляти аномалії і попереджувати про можливі атаки. Це дозволяє організаціям швидко реагувати на нові загрози і мінімізувати ризики, пов'язані з кіберзлочинністю. Наприклад, ШІ може виявляти підозрілу активність у реальному часі і автоматично приймати заходи для захисту системи.

Розробка автономних систем, таких як безпілотні автомобілі та дрони, базується на використанні ШІ для прийняття рішень в режимі реального часу. Ці системи використовують дані з різних сенсорів для навігації і виконання завдань без втручання людини. Автономні транспортні засоби можуть аналізувати навколишнє середовище, визначати найкращі маршрути і уникати перешкод, що робить їх безпечними і ефективними.

1.4 Нейронні мережі

Нейронні мережі — це одна з основних технологій штучного інтелекту, яка імітує роботу людського мозку для обробки даних і прийняття рішень. Вони відіграють ключову роль у сучасній кібербезпеці, оскільки дозволяють виявляти і передбачати кіберзагрози з набагато більшою точністю, ніж традиційні методи.

Нейронні мережі складаються з шарів взаємопов'язаних "нейронів", які обробляють вхідні дані через серію математичних операцій. Кожен нейрон отримує вхідні дані, обробляє їх і передає результати наступним нейронам аж до того, як буде отримано вихідний сигнал. Навчання нейронних мереж відбувається через процес, званий "зворотнім поширенням помилки", де мережа постійно коригує свої ваги на основі розбіжностей між її прогнозами та реальними результатами.

Важливо розуміти важливість нейронних мереж у двох словах, 3 шари нейронної мережі додають суті її існування та повної значущості.

Починаючи з 1-го рівня, нейронні мережі поєднують у собі силу наших нейронних здібностей для обробки інформації та створення результатів. Подібним чином штучні нейронні мережі також містять ці 3 рівні для організованої обробки інформації та початку виконання завдань. Ось такі шари нейронної мережі обговорюються коротко.

Перший рівень нейронної мережі є вхідним. Для обробки будь-якої інформації комп’ютеру важливо вводити інформацію, яка підлягає обробці. Прихований рівень у нейронній мережі також відомий як рівень обробки. Цей шар є найважливішим з усіх шарів і обробляє вхідні дані, які отримує наш мозок.

Як тільки вхідні дані отримані першим рівнем, вони надсилаються на наступний рівень мережі. Тут інформація інтерпретується, обробляється та розбивається на більш дрібні компоненти, щоб мозок міг зрозуміти її.

Останній рівень нейронної мережі є вихідним. Починаючи з вхідного рівня, мережа починає обробку інформації, яка в кінцевому підсумку призводить до виходу. Цей результат є таким самим, як когнітивна відповідь, яку ми надаємо іншій людині. Оскільки прихований рівень обробляє інформацію, він створює вихід для генерування відповіді з нашої сторони. Подібним чином штучні нейронні мережі також створюють вихідні дані, коли їх просять виконати певне завдання або обчислення з використанням вихідного рівня в нейронній мережі. Це останній шар нейронної мережі.

Оскільки ми тепер зрозуміли основи нейронних мереж і те, як вони працюють, тепер розберемося в перевагах нейронних мереж.

Ефективний візуальний аналіз. Найперша перевага нейронних мереж полягає в тому, що вони забезпечують ефективний візуальний аналіз. Оскільки штучна нейронна мережа схожа на нейронну мережу людини, вона здатна виконувати більш складні завдання та дії порівняно з іншими машинами.

Це передбачає аналіз візуальної інформації та розділення її на різні категорії. Типовим прикладом цієї переваги є коли будь-який веб-сайт, який ви відвідуєте, просить вас підтвердити, чи є ви роботом.

Роботи не можуть ефективно аналізувати візуальну інформацію, тоді як люди можуть успішно це робити. Це доводить, що будь-яка особа, яка входить на веб-сайт, є людиною, оскільки вона повинна розрізняти різні зображення та об’єднувати зображення певного типу.

Обробка неорганізованих даних. Ще однією з найбільших переваг нейронних мереж є те, що вони здатні обробляти неорганізовані дані. Шляхом обробки, сегрегації та класифікації неорганізованих даних штучні нейронні мережі або ШНМ можуть дуже добре організовувати дані.

У співпраці з аналітикою великих даних неорганізовані дані можна структурувати за подібним шаблоном і, у свою чергу, організовувати. З появою ШНМ завдання організації невпорядкованих даних стало набагато легшим.

На відміну від традиційних часів, коли командам кваліфікованих людей доводилося витрачати свої дні на класифікацію невпорядкованих даних, сьогодні комп'ютери можуть виконувати ту саму функцію за кілька хвилин, якщо не секунд.

Адаптивна структура. Третя перевага нейронних мереж полягає в тому, що їх структура має адаптивний характер. Це означає, що для якої б мети не застосовувалася ШНМ, вона змінює свій курс структури відповідно до мети.

Від розвитку когнітивних можливостей машини до виконання складних програм структура нейронних мереж може змінюватися. Це на відміну від досить жорстких структур численних алгоритмів і програм машинного навчання.

На відміну від незмінних структур, штучні нейронні мережі швидко трансформуються, адаптуються та пристосовуються до нових умов і відповідно демонструють свої навички. Це також вказує на те, що вид навчання, яке входить у навчання цих мереж, є порівняно меншим і більш адаптивним.

Зручний інтерфейс. Остання перевага серед інших полягає в тому, що вони мають зручний інтерфейс. Щоб будь-яка машина чи штучне обладнання мало успіх, її інтерфейс і зручність у використанні повинні бути зручними для користувача.

Так само він не повинен бути надто складним для роботи та простим у використанні. Це стосується якнайкращого опису штучних нейронних мереж. Завдяки зручному інтерфейсу ШНМ можна навчити без зайвих складнощів.

Крім того, вони здатні адаптувати свої структури, що робить їх ще більш корисними для роботи напівкваліфікованих і кваліфікованих професіоналів. Це одна з найбільших переваг цієї концепції, оскільки її можна легко адаптувати до будь-якої команди професіоналів, яка хоче з нею працювати.

Існує безліч різноманітних мереж, які функціонують незалежно в нейронній мережі та виконують багато підзавдань. Немає жодних вимог до будь-якої взаємодії один з одним під час процесу обчислення [3].

Незважаючи на те, що переваги нейронних мереж перевищують їх недоліки, важливо враховувати їх і навіть глибоко досліджувати їх місцезнаходження. Давайте тепер прочитаємо про деякі з відомих недоліків нейронних мереж.

Вимоги до обладнання. Незважаючи на їх здатність швидко пристосовуватися до мінливих вимог мети, для якої вони повинні працювати, нейронні мережі можуть бути трохи важкими для організації та організації. Це означає, що їм потрібні важкі машини та апаратне обладнання для роботи в будь-якому застосуванні.

Для початківців або тих, хто має обмежений бюджет, це може бути однією з перешкод нейронних мереж. Крім того, це також може означати, що потрібно інвестувати в додаткові речі більше, ніж в основний компонент процесу.

Таким чином, штучні нейронні мережі можуть бути дещо проблематичними, коли мова заходить про їх апаратне забезпечення, організацію та розміщення.

Другий недолік нейронних мереж полягає в тому, що вони часто можуть створювати неповні результати. Оскільки ШНМ навчені адаптуватися до мінливих застосувань нейронних мереж, їх часто не навчають для всього процесу. Хоча це здається досить легким аспектом, коли мова йде про переваги ШНМ, він може швидко перетворитися на недолік, як тільки настане час для виведення. Через неповні результати ШНМ багато часу були предметом розмов у місті. За допомогою численних теорем для таких мереж можна розрахувати лише ймовірне значення або оцінку.

Придатність даних. Ще одна з проблем нейронних мереж полягає в тому, що вони сильно залежать від даних, які їм надаються. Це означає, що ефективність будь-якої нейронної мережі прямо пропорційна кількості даних, які вона отримує для обробки. Більше того, ШНМ також страждають, якщо надані їм дані недостатньо придатні. Таким чином, алгоритми штучної нейронної мережі

можуть помилитися під час аналізу даних, доступних у невеликих кількостях і тих, які вони не можуть легко інтерпретувати. Навіть коли ці мережі навчаються, їх слід наповнювати величезною кількістю даних, щоб підготувати їх до майбутнього. Якщо ні, то результати можуть виявитися помилковими та можуть спотворити фактичні результати обчислень, застосування чи просто завдання.

Мінімальний контроль. Хоча програми штучних нейронних мереж є досить вигідними, коли мова заходить про організацію невпорядкованих даних, вони також можуть бути дуже шкідливими. Це стосується мінімального контролю, який тренери мають над фактичною продуктивністю та загальним функціонуванням ШНМ.

Від ймовірної цінності до невідомих етапів роботи, штучні нейронні мережі значною мірою приховані у своїй фактичній структурі. Це може означати, що на ці мережі не можна здійснити зовнішній вплив чи контроль, щоб запустити їх відповідно до зручності користувача.

Використання нейромереж в кібербезпеці:

- Виявлення аномалій: Нейронні мережі ефективно використовуються для виявлення аномалій у мережевому трафіку, що може свідчити про присутність зловмисних дій, таких як DDoS-атаки чи проникнення. Вони аналізують шаблони трафіку і виявляють відхилення, які відрізняються від норми.
- Класифікація шкідливого програмного забезпечення: Шкідливе ПЗ може бути класифіковане нейронними мережами на основі його поведінки, коду, та інших характеристик. Це допомагає системам кібербезпеки швидше ідентифікувати і реагувати на загрози.
- Прогнозування кібератак: За допомогою аналізу великих масивів даних, нейронні мережі можуть передбачати потенційні кібератаки перш ніж вони стануться. Вони навчаються з історичних даних про атаки і можуть прогнозувати майбутні тренди або атаки [4].

Нейромережі не можна вважати оптимальним рішенням для всіх обчислювальних завдань. У багатьох випадках традиційні комп'ютери та класичні

обчислювальні методи є більш придатними. Сучасні цифрові обчислювальні машини перевершують людей у здатності виконувати числові та символьні обчислення. Проте, люди здатні без зусиль вирішувати складні задачі, пов'язані зі сприйняттям зовнішніх даних (наприклад, розпізнавання обличчя в натовпі), з такою швидкістю і точністю, що значно перевищує можливості навіть найпотужніших комп'ютерів.

У таблиці 1.1 [5] наведено порівняння традиційного комп’ютера з біологічною нейронною системою.

Таблиця 1.1 – Порівняння традиційного комп’ютера з біологічною нейронною системою

	Комп’ютер з архітектурою фон Неймана	Біологічна нейрона система
Процесор	Складний Високошвидкісний Один чи декілька	Простий Низькошвидкісний Велика кількість
Пам'ять	Відділена від процесора Локалізована адресація за адресою	Інтегрована в процесор Розподілена адресація по змісту
Обчислення	Централізовані Послідовні збережені програми	Розподілені паралельні Самонавчання
Надійність	Висока вразливість	Живучість
Спеціалізація	Числові й символьні операції	Проблеми сприйняття
Середовище функціонування	Строго визначене Строго обмежене	Погано визначене Без обмежень

Продовження таблиці 1.1

Функції	Логічно, через правила, концепції, обчислення	Через зображення, рисунки, керування
Метод навчання	За правилами (дидактично)	За прикладами (сократично)
Застосування	Числова та символьна обробка інформації	Розпізнавання мови, розпізнавання образів, розпізнавання текстів

1.5 Архітектура нейронних мереж

Існує багато компонентів архітектури нейронної мережі. Кожна нейронна мережа має кілька спільних компонентів (рис.1.2).

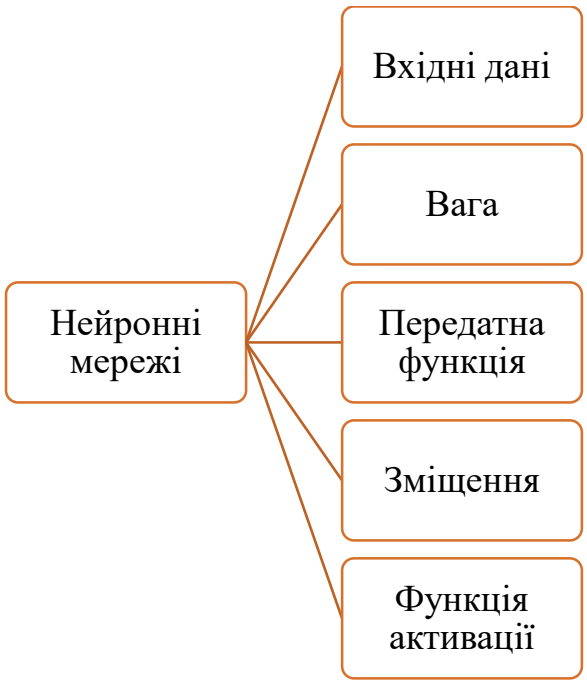

Рисунок 1.2 – Спільні компоненти нейронної мережі

Вхідні дані – це дані, які вводяться в модель з метою навчання та навчання.

Вага – вага допомагає впорядкувати змінні за важливістю та впливом внеску.

Передатна функція. Передатна функція – це коли всі вхідні дані підсумовуються та об’єднуються в одну вихідну змінну.

Функція активації. Роль функції активації полягає в тому, щоб вирішити, чи потрібно активувати певний нейрон. Це рішення ґрунтується на тому, чи буде вхід нейрона важливим для процесу передбачення.

Зміщення – зміщення зсуває значення, задане функцією активації.

У таблиці 1.2 наведено типові архітектури нейронних мереж.

Таблиця 1.2 – Типові архітектури нейронних мереж

Мережі прямого поширення	Рекурентні мережі
Перцептрони Мережа Back Propagation Мережа зустрічного поширення Карта Кохонена	Мережа Хопфілда Мережа Хемінга Мережа адаптивної резонансної теорії Двоскерована асоціативна пам'ять

Мережі прямого поширення належать до статичних, оскільки вхідні сигнали на нейрони не залежать від попереднього стану мережі.

Персептрон – нейронна мережа, яка застосовує математичну операцію до вхідного значення, надаючи вихідну змінну.

Мережа Back Propagation – це багатошаровий перцептрон, де для навчання використовується метод зворотного поширення помилки. Ця архітектура дозволяє вирішувати складніші задачі нелінійної класифікації та регресії.

Мережа зустрічного поширення – це модифікація стандартної мережі зворотного поширення, де використовуються додаткові шари або з'єднання для покращення точності та швидкості навчання.

Карта Кохонена – це вид самоорганізуючої карти (SOM), що використовується для візуалізації та кластеризації даних. Карти Кохонена

допомагають у виявленні прихованих структур у даних шляхом їх проектування на низьковимірні простори.

Рекурентні мережі є динамічними через наявність зворотних зв'язків (петель), які змінюють вхідні сигнали на нейрони з часом, що призводить до змін у станах мережі.

Мережа Хопфілда – це тип асоціативної пам'яті з повнозв’язними нейронами, де кожен нейрон пов'язаний з кожним іншим нейроном. Використовується для збереження і відтворення патернів.

Мережа Хемінга – це мережа, що використовує Хемінгову відстань для класифікації вхідних даних. Вона порівнює вхідні вектори з еталонними і вибирає найближчий до них.

Мережа адаптивної резонансної теорії – це мережа, що використовується для кластеризації і розпізнавання образів, дозволяючи мережі навчатися нових даних без забування старих. Вона складається з двох підмереж: орієнтовної і резонансної.

Двоскерована асоціативна пам'ять – це мережа асоціативної пам'яті, що має двонаправлені зв'язки між шарами нейронів. Використовується для зберігання і відтворення пар даних (патернів) у двох напрямках.

Оригінальність нейромереж, які імітують біологічний мозок, полягає в їхній здатності навчатися на прикладах з навчальної множини. Процес навчання нейромереж передбачає налаштування архітектури та вагових коефіцієнтів синаптичних зв'язків відповідно до даних навчальної множини, що дозволяє ефективно розв'язувати поставлені задачі.

Висновки за розділом 1

У першому розділі було проведено огляд основних термінів і понять, що стосуються предметної області штучного інтелекту. Визначено, що штучний інтелект є галуззю, яка займається створенням систем, здатних виконувати завдання, що зазвичай вимагають людського інтелекту, включаючи розпізнавання образів, розуміння природної мови, прийняття рішень та самонавчання. ШІ

охоплює підгалузі, такі як машинне навчання, обробка природної мови, робототехніка та експертні системи. Нейронні мережі описуються як обчислювальні моделі, натхненні біологічними нервовими системами, які складаються з вузлів та зв’язків і використовуються для вирішення складних задач, таких як класифікація, регресія, кластеризація та генерація даних. Їхня основна перевага полягає в здатності до навчання на основі прикладів, що дозволяє їм адаптуватися до нових даних. Було також обговорено різні типи архітектур нейронних мереж, включаючи мережі прямого поширення та рекурентні мережі.

2 АНАЛІЗ ПРИНЦИПУ РОБОТИ ШТУЧНОГО ІНТЕЛЕКТУ, ЙОГО СХОЖІСТЬ ТА ВІДМІННІСТЬ ВІД МОЗКУ ЛЮДИНИ

2.1 Мозок людини, нейрони, принцип їх роботи

Клітинний автомат – дискретна модель, яка описує регулярну решітку комірок, можливі стани комірок та правила зміни цих станів [6].

Представимо наступний клітинний апарат. Розташуємо елементи на регулярній решітці, для кожного елементу визначимо окіл з радіусом R, котра буде областю слідкування цього елементу (рис 2.1).

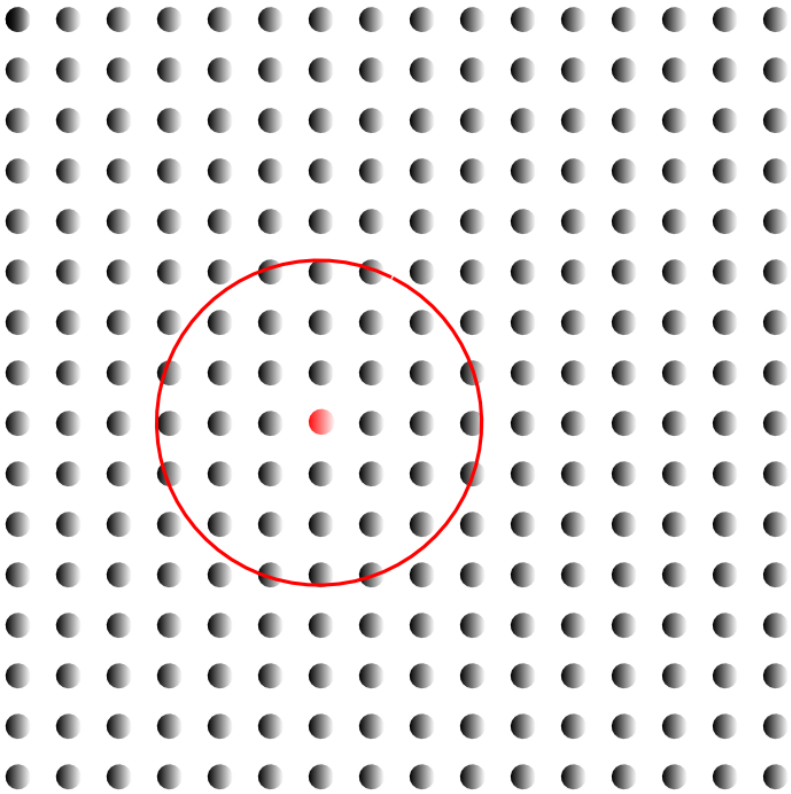

Рисунок 2.1 – Клітинний апарат

Пустимо на плоскість автомату такий патерн активності, що усі активні елементи знаходяться в області радіусом R (рис 2.2).

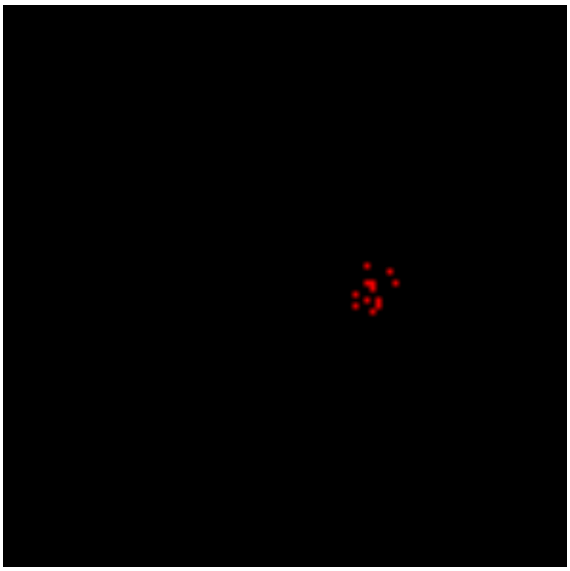

Рисунок 2.2 – Апарат з радіусом активних елементів

Зараз підрахуємо, скільки активних елементів знаходяться в полі стеження кожного з елементів (рис. 2.3).

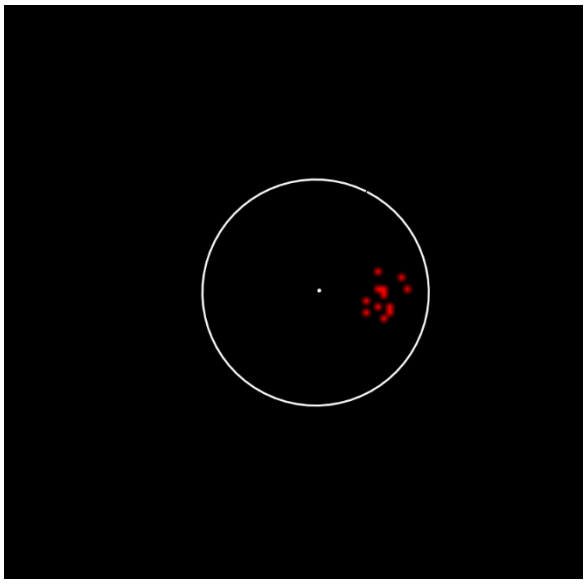

Рисунок 2.3 – Розрахунок активних елементів

Задаємо правило – кожен елемент, що знаходиться в стані спокою, у якого кількість активних елементів в радіусі слідкування буде перебільшувати деякий поріг, з вірогідність p (для прикладу візьмемо 3%) буде переходити у стан збудження. Запам’ятаємо персонально для цього елементу його вибір та картину активності в полі його слідкування.

У результаті навколо патерна початкової активності утвориться оточення з активних елементів, котрі утворять випадково згенерований візерунок (рис 2.4). Така активність називається хвильовою.

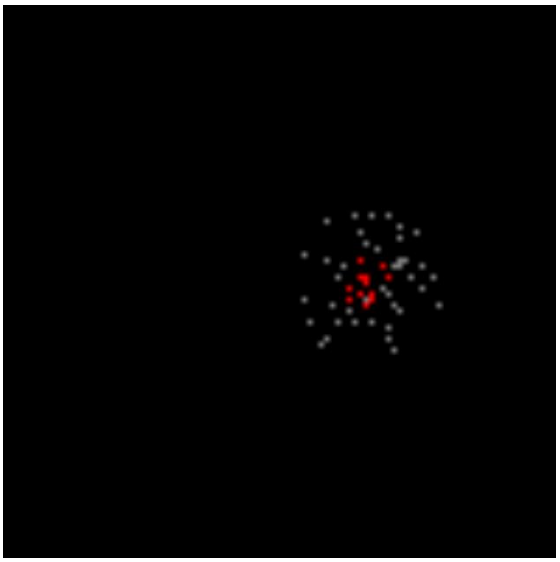

Рисунок 2.4 – Випадково згенерований візерунок

Перейдемо к наступній ітерації. На другому такту моделювання елементи, що знаходяться по периметру зони хвильової активності, «побачать» активність, що утворилася недавно. Повторимо для них процедуру активації. Елементи, які перейшли у стан збудження у минулій ітерації переведемо в стан релаксації, тобто вимкнемо їх активність і на деякий час заблокуємо їх можливість збудження від поширюваного сигналу (рис. 2.5).

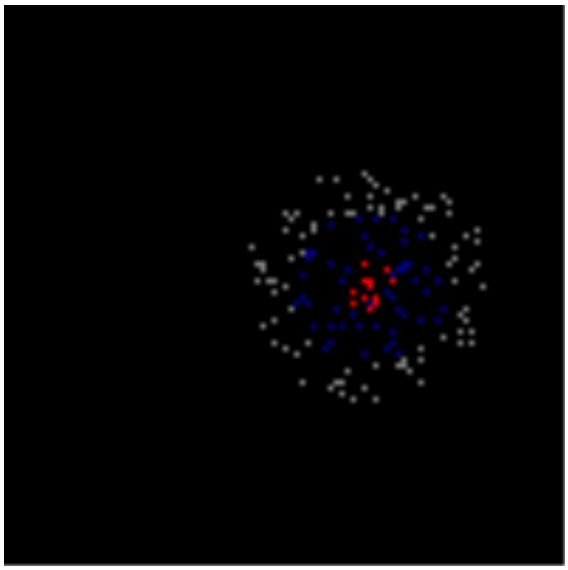

Рисунок 2.5 – Повторення процедури активації

Повторюючи кроки моделювання, ми отримаємо поширювану по автомату активність з деяким унікально створеним візерунком (рис. 2.6).

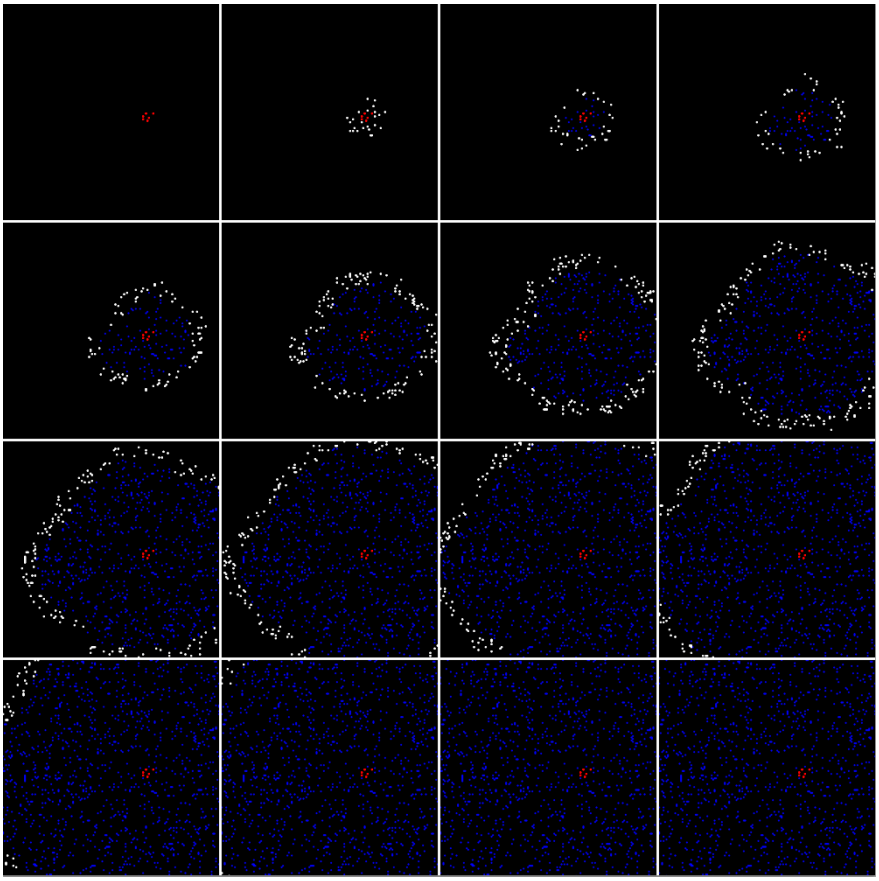

Рисунок 2.6 – Унікально створений візерунок

Якщо створити в автоматі інший патерн активності, то він так само створить порюваний від нього фронт хвилі. Але, що важливо, новий візерунок буде унікальний та відмінний від попереднього. Любий візерунок початкової активності буде створювати унікальну, характерну тільки для нього хвилю.

Змінюючи початковий візерунок, ми кожен раз будемо отримувати нову хвилю. При цьому, елементи автомату будуть пам’ятати, які візерунки через них все проходили. Введемо правило хвильового збудження. Кожен елемент, який виявив високий рівень активності навколо себе, повинен буде перевірити, чи не схожа картина цієї активності на один із запам’ятованих раніше візерунків. Якщо картина виявиться йому знайома, йому слід поводитися так само, як і при першому проходженні такої хвилі.

Підсумкову логіку роботи нашого автомата можна описати так: якщо елемент бачить незнайомий сигнал, то він випадково визначається спрацьовувати йому чи ні. При цьому він запам'ятовує і візерунок хвилі, і свій вибір. Якщо сигнал йому знайомий, він повторює вибір, зроблений при першому знайомстві з цим сигналом. Спільне спрацьовування елементів породжує фронт хвилі, що розходиться від патерну початкової активності. Стан спокою забезпечує односпрямоване поширення хвилі від місць, де активність вже була в бік, де її ще не було.

Для кожного випромінюючого патерну хвиля буде, по-перше, мати унікальний відмінний від інших хвиль візерунок поширення, а по-друге, цей візерунок буде завжди один і той же для одного і того ж старту патерна. Це означає, що якщо ми введемо словник понять і зіставимо кожне поняття з певним кодуючий його патерн, ми зможемо передавати інформацію про активність будь-якого поняття по всій поверхні клітинного автомата. Так як кожен патерн породжує хвилю з унікальним візерунком, то в будь-якому місці клітинного автомата по рисунку хвилі можна судити про те, яке поняття поширюється цією хвилею. На рисунку 2.7 видно, як розрізняються візерунки різних хвиль, проходячи через одне й те місце.

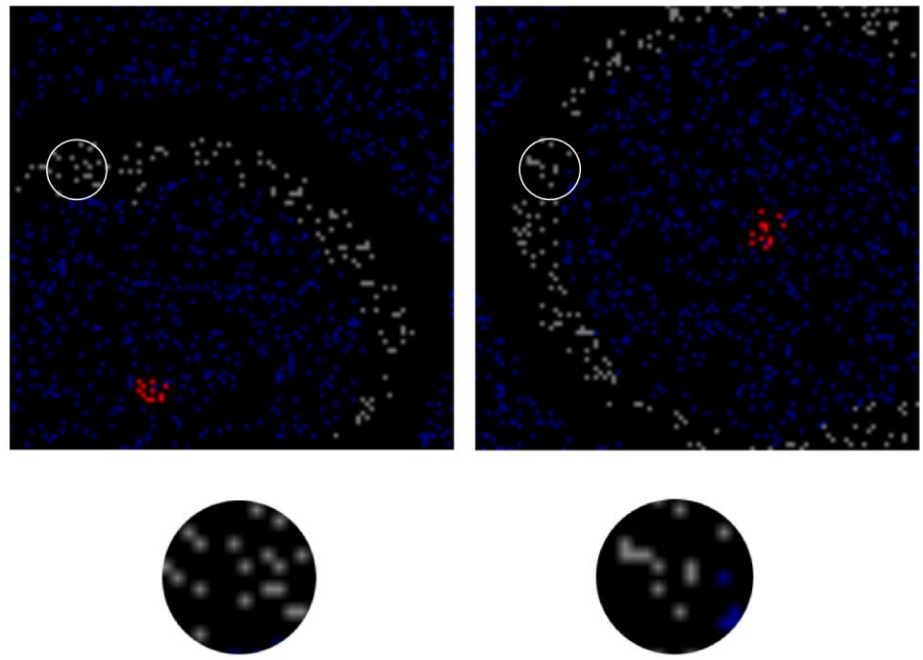

Рисунок 2.7 – Візерунки різних хвиль

Якщо сигнал кодується досить малим відсотком активних елементів, кілька сигналів можуть поширюватися по автомату одночасно, зберігаючи свою індивідуальність і інтерферуючи друг з одним. При одночасному поширенні кількох хвиль фронти цих хвиль можуть проходити крізь друга, не змінюючи свого візерунка (рис. 2.8).

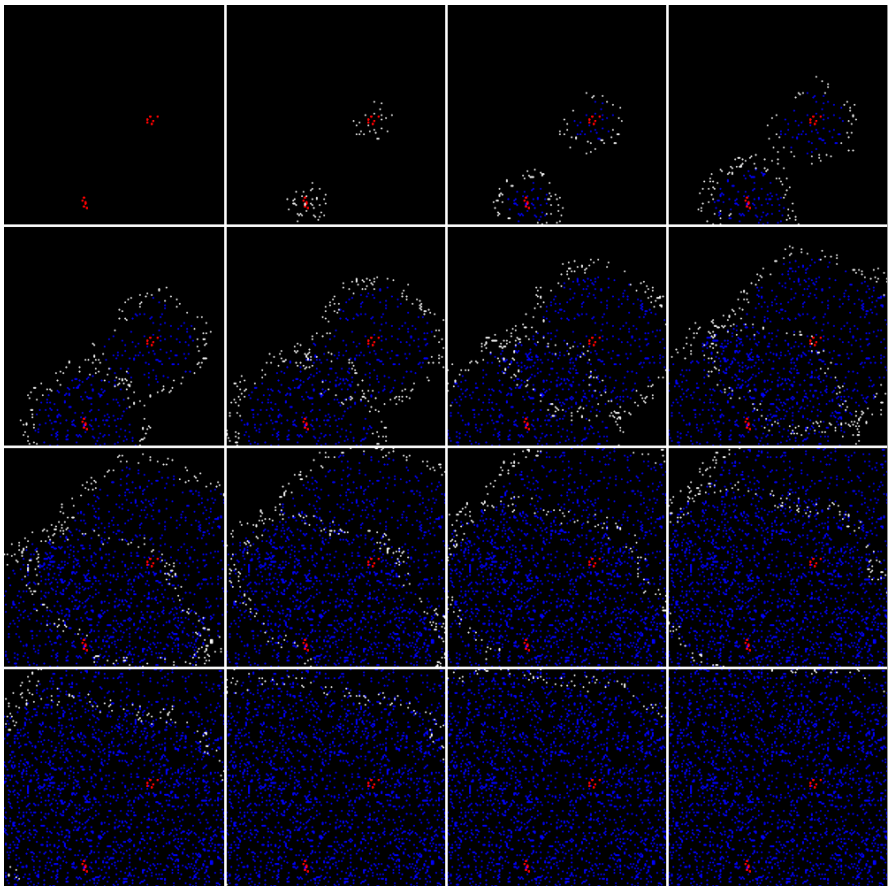

Рисунок 2.8 – Проходження фронтів хвиль

Одночасно автоматом можуть поширюватися кілька хвиль, відповідні різним сигналам. Однак зі збільшенням їх кількості рано чи пізно настає момент, коли накладення кількох хвильових візерунків призводить до появи хибних спрацьовувань. Помилкові спрацьовування лавиноподібно збільшують кількість активних елементів. З якогось моменту виникає режим самозбудження, картина якого нагадує картину епілептичного нападу.

Слід зазначити, що сигнали, закодовані таким автоматом, набувають властивість дуалізму, що відповідає дуалізму елементарних частинок. Так само,

як і частинки, які виявляють одночасно і корпускулярні властивості і хвильові, інформаційний сигнал в описуваній моделі – це одночасно і патерн, який запускає хвилю, і хвиля, яка в кожній фазі свого шляху переходить у патерн, який, у свою чергу, випромінює продовження хвилі.

Описаний алгоритм формування хвиль має значний недолік. Коли елемент бачить хвилю вперше він робить вибір брати участь у хвилі чи ні, та був запам'ятовує зроблений вибір разом із видимим йому візерунком хвилі. Це дозволяє наступного разу отримати повторюваність хвильового візерунка. Але при цьому елемент може пізнавати хвилю тільки в тому випадку, коли вона приходить з того ж боку, що й у момент запам'ятовування. Нескладно модифікувати алгоритм, щоб позбутися цього недоліку. Досить небагато відкласти момент запам'ятовування нової хвилі. Замість запам'ятовування візерунка в той момент, коли нова хвиля підійшла до елемента, можна дати хвилі трохи поширитися далі і запам'ятати візерунок навколо елемента. Змінюється і процедура подальшого впізнавання хвилі. Хвиля вважається впізнаною, якщо поруч із елементом виникло повторення частини запам'ятованого візерунка [7].

В результаті елементи автомата стають інваріантними до напряму поширення хвиль. Тобто той самий візерунок хвилі відтворюється однаково, незалежно від того, звідки прийшов фронт цієї хвилі.

Така доробка автомата додає йому дуже важливої якості. Візьмемо довільне знайоме автомату поняття. Цьому поняттю буде відповідати хвиля з унікальним візерунком. Якщо в будь-якому місці площини автомата відтворити фрагмент такого візерунка, то з цього місця поширюється хвиля, що відтворює унікальний візерунок початкової хвилі по всьому шляху свого прямування. Наприклад, якщо на автоматі (рис. 2.9) в обведеній лінією області (1) створити певний візерунок, то дійшовши до місця (2) фронт хвилі створить унікальний для цієї хвилі візерунок.

Рисунок 2.9 – Створення певного візерунку

Якщо тепер в області (2) відтворити візерунок, який був там при проходженні фронту хвилі, то з цього місця вийде хвиля, що повторює вихідний візерунок (рис. 2.10). Дійшовши до області (1) вона створить візерунок, тотожний тому, що колись запускав вихідну хвилю.

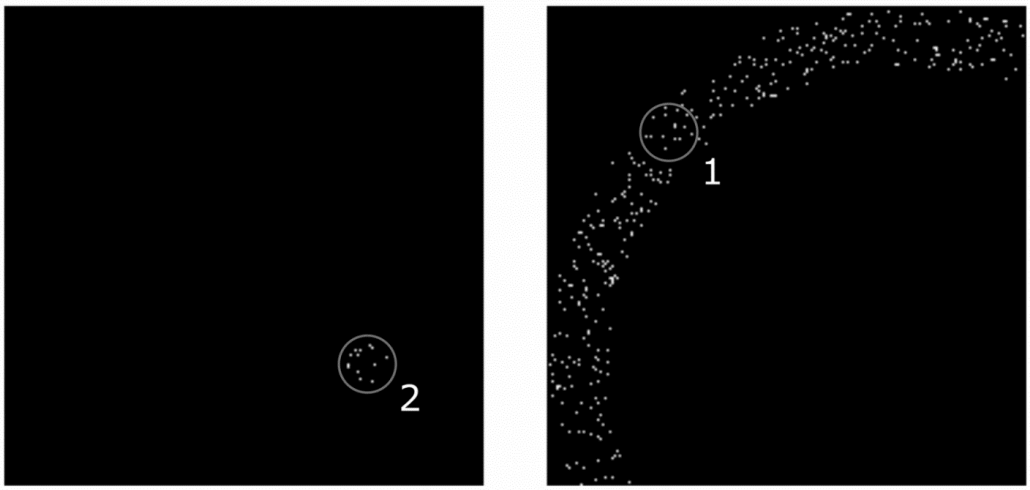

Рисунок 2.10 – Повторення вихідного візерунку

Таким чином, виходить повна пов'язаність всього простору автомата. Тобто, потенційно будь-яке його місце може запустити інформаційну хвилю і при цьому всі ділянки автомата кудись дійде відповідна хвиля отримають доступ до цієї інформації.

2.2 Нейрони нейромережі

У попередньому пункті ми розібрали як приблизно працює мозок людини – якщо виключити усю біологію з опису, ми отримаємо подібний автомат. Але що ж з штучним інтелектом? Нейромережі (для прикладу візьмемо мережу для ідентифікації зображеної тварини) мають архаїчне представлення функціювання нейрону (рис. 2.11).

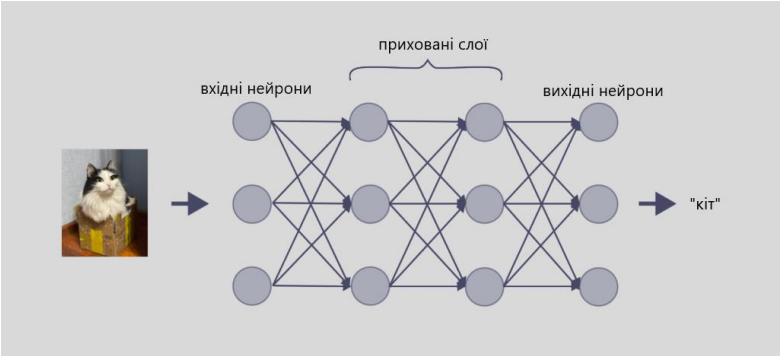

Рисунок 2.11 – Мережа для ідентифікації зображеної тварини

За цією теорією, нейрони являють собою щось схоже на сумматор, котрий приймає в себе сигнал(або сигнали), оброблює їх та передає або на наступний шар, або на вихідні нейрони. Ця теорія та її подальша імплементація з’явилися в результаті невірного висновку після аналізу нейрону мозка людини, але через зовсім інші задачі, котрі виконують нейромережі, дана конструкція дозволила робити їх дуже гнучкими.

Штучний нейрон в моделі нейронної мережі можна розглядати як простий процесор для обчислень, який отримує, обробляє та передає інформацію [8]. Основні компоненти штучного нейрона:

- Вхідні сигнали;

Кожен нейрон у нейронній мережі отримує один або кілька вхідних сигналів. Ці сигнали можуть походити від зовнішніх даних або від виходів інших нейронів у мережі. Вхідний сигнал до нейрона часто представлений у вигляді числових значень, які можуть відображати різноманітні типи даних, такі як зображення, звук, текстові дані тощо.

- Ваги;

Кожен вхідний сигнал має асоційовану з ним вагу, яка є параметром нейрона і вказує на значущість кожного входу. Ваги регулюють те, як сильно кожен вхід впливає на активність нейрона. Налаштування цих ваг відбувається під час процесу навчання нейронної мережі і є ключовим для здатності мережі адаптуватися та навчатися.

- Суматор;

Ця частина нейрона обчислює зважену суму вхідних сигналів, яка є основою для визначення того, чи буде нейрон активовано. Сума вхідних сигналів, помножених на їхні ваги, формує загальний вхідний сигнал для нейрона.

- Активаційна функція;

Активаційна функція використовується для визначення виходу нейрона на основі зваженої суми вхідних сигналів. Вона вирішує, чи достатньо сильний вхідний сигнал для активації нейрона. До популярних активаційних функцій належать сигмоїдна функція, гіперболічний тангенс та ReLU (Rectified Linear Unit). Ця функція може допомагати мережі уникнути лінійності та дозволяти їй вчитися більш складним шаблонам.

- Вихід;

Вихід нейрона є результатом активаційної функції і може передаватися як вхід до наступних нейронів у мережі або бути частиною виходу мережі. Виходи з кінцевих нейронів можуть інтерпретуватися як рішення або прогнози нейронної мережі.

Кожен з цих компонентів працює разом для того, щоб нейронна мережа могла виконувати складні завдання, такі як класифікація даних, регресійний аналіз, розпізнавання образів та багато іншого. Через велику кількість нейронів та зв'язків між ними, нейронні мережі здатні "вчитися" і адаптуватися до нових умов, змінюючи ваги своїх зв'язків.

2.3 Штучний інтелект у кібербезпеці

Штучний інтелект і машинне навчання стали критично важливими технологіями в інформаційній безпеці, оскільки вони здатні швидко аналізувати мільйони подій і виявляти багато різних типів загроз – від зловмисного програмного забезпечення, що використовує вразливості нульового дня, до виявлення ризикованої поведінки, яка може призвести до фішингу. атака або завантаження шкідливого коду. Ці технології з часом навчаються, спираючись на минуле, щоб визначити нові типи атак зараз. Історії поведінки створюють профілі користувачів, активів і мереж, дозволяючи ШІ виявляти відхилення від встановлених норм і реагувати на них [9].

ШІ ідеально підходить для вирішення деяких наших найскладніших проблем, і кібербезпека, безперечно, потрапляє в цю категорію. З огляду на сучасні кібератаки, що постійно розвиваються, і поширення пристроїв, машинне навчання та штучний інтелект можна використовувати, щоб «не відставати від поганих хлопців», автоматизуючи виявлення загроз і реагуючи ефективніше, ніж традиційні підходи, керовані програмним забезпеченням.

Результатом є нові рівні інтелекту, які живлять людські команди в різних категоріях кібербезпеки, зокрема:

- Інвентаризація IT-активів – отримання повної точної інвентаризації всіх пристроїв, користувачів і програм із будь-яким доступом до інформаційних систем. Категоризація та вимірювання критичності для бізнесу також відіграють велику роль в інвентаризації.
- Викриття загроз – хакери, як і всі інші, слідкують за тенденціями, тому те, що модно у хакерів, регулярно змінюється. Системи кібербезпеки на основі штучного інтелекту можуть надати сучасні знання про глобальні та галузеві загрози, щоб допомогти прийняти важливі рішення про пріоритетність не лише на основі того, що може бути використано для атаки на підприємство.
- Ефективність засобів контролю – важливо розуміти вплив різних засобів безпеки та процесів , які ви використовуєте для підтримки надійної безпеки.

Штучний інтелект може допомогти зрозуміти, де ваша програма інформаційної безпеки має сильні сторони, а де — недоліки.

- Прогнозування ризику злому – враховуючи інвентаризацію IT-активів, виявлення загроз і ефективність засобів контролю, системи на основі штучного інтелекту можуть передбачити, як і де вас найімовірніше буде зломлено, щоб ви могли планувати розподіл ресурсів і інструментів на слабкі місця. Рекомендовані відомості, отримані в результаті аналізу штучного інтелекту, можуть допомогти вам налаштувати та вдосконалити елементи керування та процеси для найбільш ефективного підвищення кіберстійкості вашої організації.
- Реагування на інциденти – системи на базі штучного інтелекту можуть надавати покращений контекст для встановлення пріоритетів і реагування на сповіщення системи безпеки, для швидкого реагування на інциденти та виявлення першопричин, щоб пом’якшити вразливі місця та уникнути майбутніх проблем.
- Зрозумілість – ключ до використання штучного інтелекту для посилення команд інформаційної безпеки людей – це зрозумілість рекомендацій і аналізу. Це важливо для отримання підтримки від зацікавлених сторін у всій організації, для розуміння впливу різних програм інформаційної безпеки та для надання відповідної інформації всім залученим зацікавленим сторонам, включаючи кінцевих користувачів, відділи безпеки, CISO, аудиторів, CIO, CEO та правління директорів.

Використання штучного інтелекту у кібербезпеці має такі переваги:

Автоматизація повторюваних завдань: кібербезпека вимагає значного збору даних, аналізу, керування системою та інших повторюваних завдань, які забирають час і ресурси аналітиків. ШІ має потенціал для автоматизації цих дій, дозволяючи персоналу служби безпеки зосередити свої зусилля там, де вони найбільше потрібні.

Покращене виявлення загроз і реагування: штучний інтелект ідеально підходить для збору величезних обсягів даних, їх аналізу та реагування на основі

отриманої інформації. Ці можливості можуть покращити виявлення загроз організації та реагування на них шляхом прискорення та масштабування виявлення та реагування на кібератаки, зменшуючи шкоду, яку зловмисники можуть завдати організації.

Покращена ситуаційна обізнаність і прийняття рішень: часто співробітники служби безпеки страждають від перевантаження даними, оскільки їм більше інформації, ніж вони можуть ефективно обробити та використати. Штучний інтелект чудово справляється зі збором і обробкою даних, а інформація, яку він надає, може покращити ситуаційну обізнаність співробітників служби безпеки та здатність приймати рішення на основі даних.

Використання штучного інтелекту у кібербезпеці має такі недоліки, зображених на рисунку 2.12.

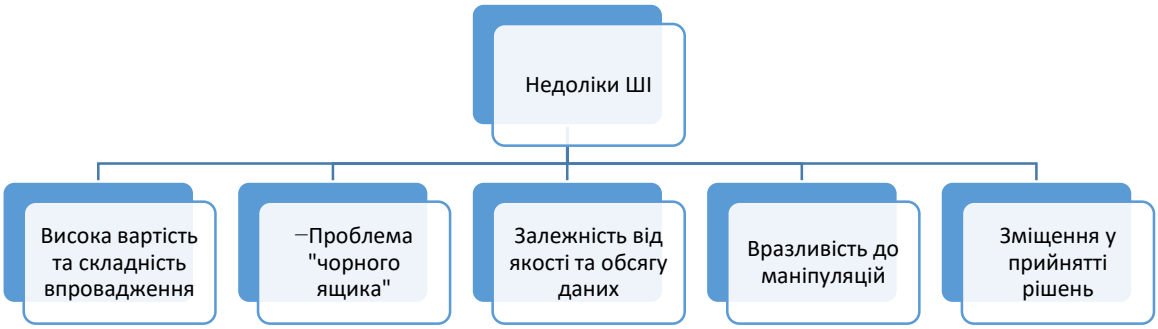

Рисунок 2.12 – Недоліки використання штучного інтелекту у кібербезпеці

Попри всі переваги, розробка та інтеграція ШІ в системи кібербезпеки може бути пов’язана з великими витратами та технічними складнощами. Не кожна організація може дозволити собі великі початкові інвестиції та потребу в кваліфікованих спеціалістах для розробки та підтримки таких систем.

ШІ часто критикують за відсутність прозорості в рішеннях, які він приймає. Це може створювати проблеми, коли потрібно точно зрозуміти, за якими критеріями було прийнято рішення, особливо у випадках, коли від цього залежать критичні для безпеки аспекти. І ця проблема не є одиничною або

нерозповсюдженою – навіть розробники малих рішень не можуть точно сказати, як саме їх продукт приймає рішення, тому що його логіка знаходиться глибоко у шарах, котрих може бути сотні тисяч.

Ефективність ШІ безпосередньо залежить від обсягу та якості тренувальних даних. Некоректно підібрані, застарілі або предвзяті дані можуть призвести до неправильних висновків і дій, що знижує надійність системи. Це причина того, що крупні компанії, накшталт Facebook або Google, мали змогу набагато раніше ніж інші запропонувати та реалізувати свої – вони з самого початку мали великі об’єми інформації для аналізу та навчання.

«Якщо ви подивитесь на сфери застосування технології глибинного вивчення, то побачите, що вона процвітає там, де може знайти багато даних; Єдиний вихід – використовувати алгоритми, яким потрібно менше ресурсів» - Ніл Лоуренс, професор машинного навчання Шеффілдського університету та співробітник підрозділу, що відповідає за ШІ компанії Amazon.

ШІ може бути підданий атакам з боку зловмисників, які навмисно вводять спотворені або маніпулятивні дані для обходу системи безпеки. Такі атаки можуть не тільки знизити ефективність ШІ, але й призвести до серйозних безпекових інцидентів.

Якщо тренувальні дані містять зміщення, то ШІ також буде відтворювати це зміщення у своїх рішеннях. Це може призвести до несправедливих або недоречних дій, що можуть мати серйозні наслідки для організацій та осіб.

Висновки за розділом 2

У другому розділі було проведено аналіз принципу роботи штучного інтелекту та його порівняння з мозком людини, а також розглянуто застосування штучного інтелекту в кібербезпеці. Штучний інтелект, зокрема нейронні мережі, значно відрізняється від людського мозку у швидкості та способах обробки інформації. Сучасні цифрові обчислювальні машини можуть виконувати мільйони обчислень за секунду, що перевершує можливості людського мозку в числових і символьних обчисленнях. Використання алгоритмів машинного

навчання дозволяє виявляти загрози та атаки з високою точністю і швидкістю, що значно підвищує рівень захисту інформаційних систем. Використання ШІ в кібербезпеці демонструє його потенціал у вирішенні складних і важливих задач, що підкреслює значимість подальших досліджень та розвитку цієї технології.

3 ПРОЕКТУВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАСОБІВ ЗАХИСТУ СИСТЕМИ АУТЕНТИФІКАЦІЇ НА ОСНОВІ АНАЛІЗУ

3.1 Обґрунтування обраних технологій та засобів розробки захисту системи аутентифікації

Для розробки було обрано наступні технології та засоби розробки систем:

- Python;
- Tensorflow;
- Keras.

Розглянемо переваги обраних технологій.

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Вона має репутацію мови, зручної для початківців, замінивши Java як найпоширенішу початкову мову, оскільки вона справляється з більшою частиною складності для користувача, дозволяючи початківцям зосередитися на повному розумінні концепцій програмування, а не на найменших деталях [10].

Python використовується для веб-розробки на стороні сервера, розробки програмного забезпечення, математики та системних сценаріїв, і популярна для швидкої розробки додатків, а також як мова сценаріїв або з’єднувальна мова для зв’язування існуючих компонентів завдяки своїм високорівневим вбудованим структурам даних, динамічному типу і динамічному зв'язуванню. Витрати на технічне обслуговування програми зменшуються завдяки Python завдяки легкому освоєнню синтаксису та акценту на зручності читання. Крім того, підтримка Python модулів і пакетів полегшує модульні програми та повторне використання коду. Python — це мова спільноти з відкритим кодом, тому багато незалежних програмістів постійно створюють бібліотеки та функціональні можливості для неї.

Особливості та переваги Python:

- сумісний із різними платформами, включаючи Windows, Mac, Linux, Raspberry Pi та інші;
- використовує простий синтаксис, який можна порівняти з англійською мовою, що дозволяє розробникам використовувати менше рядків, ніж інші мови програмування;
- працює на системі інтерпретатора, яка дозволяє негайно виконувати код, швидко відстежуючи прототипування;
- можна обробляти процедурним, об’єктно-орієнтованим або функціональним способом.

TensorFlow є однією з найпопулярніших бібліотек для створення моделей машинного навчання та глибокого навчання. Її розроблено компанією Google, і вона використовується як для досліджень, так і для промислових застосувань.

TensorFlow конкурує з такими фреймворками, як PyTorch і Apache MXNet, може навчати та запускати глибокі нейронні мережі для класифікації рукописних цифр, розпізнавання зображень, вбудовування слів, рекурентних нейронних мереж, моделей послідовності для машинного перекладу, обробки природної мови та моделювання на основі PDE (рівняння в частинних похідних). TensorFlow підтримує масштабне прогнозування виробництва з тими самими моделями, що використовуються для навчання [11].

Keras — це бібліотека-огортка Python, яка надає обгортки для інших бібліотек DL, таких як TensorFlow, CNTK, Theano, MXNet і Deeplearning4.27 Її було розроблено з метою швидкого експериментування та випущено за ліцензією MIT. Keras працює на Python від 2.7 до 3.6 і забезпечує підтримку GPU та CPU [12]. Keras розроблено та підтримується відповідно до чотирьох керівних принципів:

- Бібліотека з відкритим вихідним кодом, що швидко розвивається, із серверними інструментами, які підтримують такі потужні промислові компанії, як Google і Microsoft.
- Популярний API для DL із великою спільнотою та вичерпною документацією.

– Зручний і простий спосіб визначення DL-моделей у серверах, таких як Tensor-Flow, CNTK або Theano.

3.2 Тестова нейромережа для кібербезпеки

Тепер спробуємо створити власну нейромережу. Обраний стек технологій – Python, Tensorflow. Представимо ситуацію – нам потрібна нейромережа, котра буде аналізувати усі спроби логіну, оцінювати їх підозрілість в залежності від пристрою, локації та часу. Припустимо, що з сайту на наш сервер передається незашифрований файл формату .json, який має в собі 5 полів (рис. 3.1).

Рисунок 3.1 – Передача незашифрованого файлу

Поля “mostused” та “current“ відповідають за найбільш вживаний девайс для логіну та поточний кожний у своїй категорії відповідно, поле “time” відповідає за час логіну.

З двох створених папок датасетів «безпечного» та «підозрілого» зчитуються приклади відповідних прикладів файлів. Після цього, у коді перевіряється, чи дорівнюють поля “mostused” та “current“ у своїх категоріях один одному, та в залежності від результату видається 1 якщо так, та 0 – якщо ні. Так само робимо з часом – якщо час логіну між 8:00 та 17:59, то для нейромережі ми передаємо 1, якщо ні – 0 (рис. 3.2).

```python
def preprocess_data(data):
    ip_equal = [1 if d['ip']['mostused'] == d['ip']['current'] else 0 for d in data]
    mac_equal = [1 if d['mac']['mostused'] == d['mac']['current'] else 0 for d in data]
    times = [int(d['time'].replace(':', '')) for d in data]

    processed_times = []
    for t in times:
        hour = int(str(t)[:2])
        if 8 <= hour < 18:
            processed_times.append(1)
        else:
            processed_times.append(0)

    processed_data = np.column_stack((ip_equal, mac_equal, processed_times))
    return processed_data
```

Рисунок 3.2 – Зчитування прикладів датасетів

Чому саме 0 та 1? Це називається стандартизація – процес, коли усі дані приводяться до одного виду. Так само працюють, наприклад, нейромережі ідентифікації – спочатку перевіряються лінії (контури тіла), потім – особливості (чи має людина окуляри, чи має тварина вуса), тобто замість даних «у нього є окуляри» «у нього є вуса», мережа відправляє дані в відповідні нейрони, котрі ідентифікують необхідні ознаки та видають відповідний результат.

Сама мережа наша мережа складається з 3х шарів – вхідного, обчислювального, вихідного. Вхідний та обчислювальний мають по 64 нейрони, функції активації використовуємо “Relu”, вихідний – один нейрон та функцію сигмоїд.

Після тренування ми отримуємо готову для використання нейромережу, яка з високою точністю (0.8071 згідно метрикам, рис. 3.3) зможе ідентифікувати загрозу та попередити про неї.

```
Epoch 99/100
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 465us/step - accuracy: 0.8026 - loss: 0.4289
Epoch 100/100
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 469us/step - accuracy: 0.8071 - loss: 0.4256
```

Рисунок 3.3 – Точність ідентифікування

Після цього створюємо окрему функцію, котра визиває модель та «скормлює» їй файл, котрий ми хочемо перевірити, після чого модель видає або «безпечно», або «підозріло».

Для реалізації системи аутентифікації з використанням нейронної мережі були використані такі технології: Flask для серверної частини та HTML, CSS, JavaScript для фронтенду.

Для серверної частини було обрано мікрофреймворк Flask, який дозволяє швидко та ефективно розробляти веб-додатки на Python. Серверна частина виконує наступні функції:

- Обробка запитів від клієнта.

- Завантаження предобученої моделі нейронної мережі для класифікації спроб логіну.

- Попередня обробка вхідних даних (JSON) та їх перетворення у формат, придатний для моделі.

- Виконання передбачення за допомогою моделі та повернення результату (класифікація логіну як "безпечний" або "підозрілий").

Фронтенд частина реалізована з використанням HTML, CSS та JavaScript і забезпечує взаємодію користувача з системою. Основні елементи інтерфейсу включають:

- Поля введення для логіну та паролю.

- Елементи управління для вибору параметрів (IP-адреса, MAC-адреса та час).

- Область для відображення поточних значень JSON та результату передбачення.

Користувач вводить логін та пароль, вибирає параметри поточної спроби логіну, і дані надсилаються на сервер через AJAX-запит. Сервер обробляє запит, робить передбачення за допомогою нейронної мережі та повертає результат, який відображається на веб-сторінці.(рис 3.4).

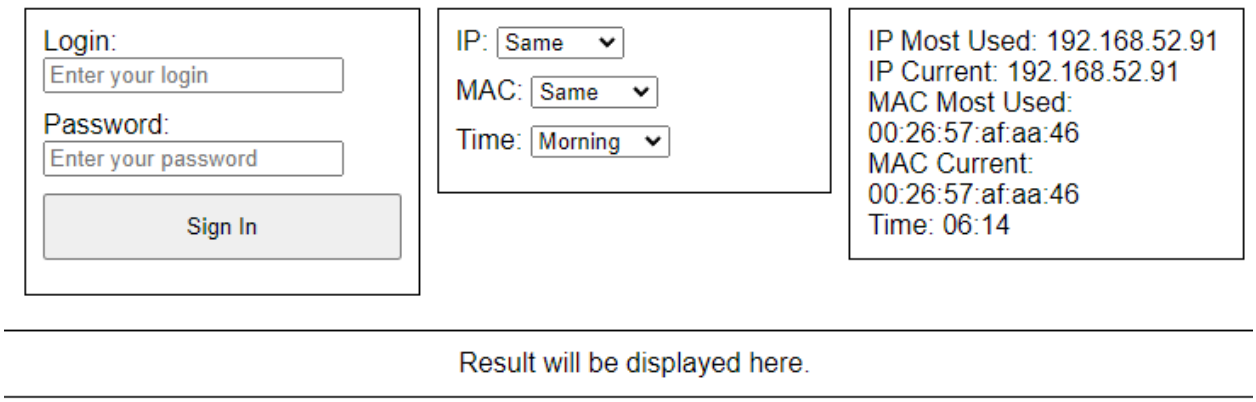

Рисунок 3.4 – Веб-сторінка

Висновки за розділом 3

В третьому розділі проведено проектування та практичну реалізації засобів захисту системи аутентифікації на основі аналізу, включаючи обґрунтування обраних технологій та розробку тестової нейромережі для кібербезпеки. Розробка тестової нейромережі для кібербезпеки показала її здатність ефективно ідентифікувати спроби несанкціонованого доступу і виявляти аномальні дії в системі аутентифікації. Нейромережі дозволяють виявляти складні шаблони поведінки, які можуть вказувати на спроби компрометації системи, що традиційні методи не завжди можуть забезпечити.

ВИСНОВОК

У ході виконання дипломної роботи було проведено глибокий аналіз сучасних методів і технік, які використовуються для ідентифікації та протидії кіберзагрозам за допомогою штучного інтелекту. Розглянуто застосування нейронних мереж, машинного навчання та глибокого навчання в контексті кібербезпеки, що дозволило оцінити їхню ефективність та потенціал у забезпеченні захисту інформаційних систем.

Технології штучного інтелекту демонструють значну ефективність у виявленні та аналізі кіберзагроз завдяки їх здатності обробляти великі обсяги даних і виявляти складні патерни у поведінці систем. Це дозволяє значно підвищити точність і швидкість реагування на потенційні загрози. ШІ автоматизує багато процесів, пов'язаних з моніторингом і реагуванням на інциденти, що зменшує людський фактор і підвищує оперативність дій у випадку виявлення аномалій або загроз. Автоматизація процесів також дозволяє зменшити навантаження на команди кібербезпеки, дозволяючи їм зосередитися на стратегічних завданнях.

Проте використання штучного інтелекту у кібербезпеці має певні обмеження. Одним з ключових аспектів є залежність від якості та обсягу навчальних даних. Без достатньо великої і якісної вибірки даних нейронні мережі та алгоритми машинного навчання можуть демонструвати недостатню точність або навіть робити хибні висновки. Крім того, алгоритми ШІ можуть бути вразливими до специфічних атак, таких як атаки на алгоритми ШІ, які можуть призвести до неправильних рішень системи безпеки. Також важливим є врахування етичних аспектів та питань приватності, оскільки збір та обробка великих обсягів даних можуть порушувати права користувачів.

Розвиток технологій глибокого навчання та їх інтеграція з іншими областями, такими як хмарні технології та інтернет речей, обіцяє подальше зростання ефективності систем кібербезпеки. Ці інтеграції дозволяють створювати більш гнучкі та масштабовані рішення, які можуть ефективно

реагувати на нові види загроз та адаптуватися до змін у навколишньому середовищі. Впровадження ШІ в сферу кібербезпеки відкриває нові можливості для захисту інформаційних систем від сучасних загроз, забезпечуючи більш проактивний і динамічний підхід до управління безпекою.

Однак, для повноцінного використання потенціалу штучного інтелекту в кібербезпеці потрібен глибокий аналіз ризиків і розробка комплексних рішень, які враховують не тільки технічні аспекти, але й правові та етичні питання. Це включає розробку політик і стандартів, які забезпечать захист приватності даних користувачів та етичне використання технологій ШІ. Необхідно також забезпечити стійкість алгоритмів до атак та створити механізми для постійного моніторингу та вдосконалення систем безпеки. Лише комплексний підхід дозволить максимально ефективно використовувати штучний інтелект для забезпечення кібербезпеки в умовах постійно змінюваного ландшафту загроз.

СПИСОК ПОСИЛАНЬ ДЖЕРЕЛ

1. Основні ідеї нейронних мереж. Режим доступу: http://masters.donntu.org/2006/kita/kornev/library/l6.html.
2. Tariq Rashid Make your own neural network /T. Rashid. – 2016 – 222 с.
3. Esteva A, Kuprel B, Novoa RA, et al. Dermatologist-level classification of skin cancer with deep neural networks? Nature. – 2017. № 4. - С. 25-35.
4. А. Редозубов. «Логіка свідомості». Режим доступу: https://habr.com/articles/326334/.
5. Роль штучного інтелекту в кібербезпеці. Режим доступу: https://www.education.ua/blog/48113/.
6. Liu X, Liu C, Huang R, et al. Long short-term memory recur-rent neural network for pharmacokinetic-pharmacodynamic modeling. Int J Clin Pharmacol er. 2021.
7. Melanie M. An Introduction to Genetic Algorithms (Fifth printing ed. Vol3). A Bradford Book The MIT Press, Cambridge, 221s. - 1999.
8. McCarthy J. What is artificial intelligence. 2004. Accessible. Режим доступу: http://www-formal.stanford.edu/jmc/whatisai.html.
9. Інформаційна сторінка структури нейронних мереж. Режим доступу: https://proglib.io/p/about-neuralnetworks .
10. Python would rather not use anything else. Режим доступу: https://www.python.org/about/.
11. Get started with TensorFlow. Режим доступу: https://www.tensorflow.org/tutorials .
12. Основна ідея бібліотеки Keras. Режим доступу: https://www.sciencedirect.com/topics/computer-science/keras.

ДОДАТОК А

tf.py

```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import json
import os

def read_json_data(folder_path):
    data = []
    for filename in os.listdir(folder_path):
        if filename.endswith('.json'):
            with open(os.path.join(folder_path, filename), 'r') as file:
                json_data = json.load(file)
                data.append(json_data)
    return data

def preprocess_data(data):
    ip_equal = [1 if d['ip']['mostused'] == d['ip']['current'] else 0 for d in data]
    mac_equal = [1 if d['mac']['mostused'] == d['mac']['current'] else 0 for d in data]
    times = [int(d['time'].replace(':', '')) for d in data]

    processed_times = []
    for t in times:
        hour = int(str(t)[:2])
        if 8 <= hour < 18:
            processed_times.append(1)
        else:
            processed_times.append(0)
```

```
        processed_data = np.column_stack((ip_equal, mac_equal, processed_times))
        return processed_data

    safe_data = read_json_data('datasets/safe')
    suspicious_data = read_json_data('datasets/suspicious')

    safe_data_processed = preprocess_data(safe_data)
    suspicious_data_processed = preprocess_data(suspicious_data)
    X = np.vstack((safe_data_processed, suspicious_data_processed))
    y = np.hstack((np.zeros(len(safe_data_processed)),
np.ones(len(suspicious_data_processed))))

    model = models.Sequential([
        layers.Dense(64, activation='relu', input_shape=(X.shape[1],)),
        layers.Dense(64, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer='nadam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

    model.fit(X, y, epochs=100, batch_size=64)

    model.save('json_classifier_model.h5')
```

ДОДАТОК Б

predict.py

```
import os
import json
import numpy as np
from tensorflow.keras.models import load_model

def read_json_data(folder_path):
    data = []
    for filename in os.listdir(folder_path):
        if filename.endswith('.json'):
            with open(os.path.join(folder_path, filename), 'r') as file:
                json_data = json.load(file)
                data.append(json_data)
    return data

def preprocess_data(data):
    ip_equal = [1 if d['ip']['mostused'] == d['ip']['current'] else 0 for d in data]
    mac_equal = [1 if d['mac']['mostused'] == d['mac']['current'] else 0 for d in data]
    times = [int(d['time'].replace(':', '')) for d in data]

    processed_times = []
    for t in times:
        hour = int(str(t)[:2])
        if 8 <= hour < 18:
            processed_times.append(1)
        else:
            processed_times.append(0)

    processed_data = np.column_stack((ip_equal, mac_equal, processed_times))
```

```
    return processed_data

test_data = read_json_data('datasets/test')
test_data_processed = preprocess_data(test_data)

model = load_model('json_classifier_model.h5')

predictions = model.predict(test_data_processed)

categories = ['safe' if pred < 0.5 else 'suspicious' for pred in predictions]

for filename, category in zip(os.listdir('datasets/test'), categories):
    print(f'{filename}: {category}')
```

ДОДАТОК В

server.py

```
from flask import Flask, request, jsonify, render_template
from tensorflow.keras.models import load_model
import numpy as np
import json

app = Flask(__name__)
model = load_model('json_classifier_model.h5')

def preprocess_data(data):
    ip_equal = 1 if data['ip']['mostused'] == data['ip']['current'] else 0
    mac_equal = 1 if data['mac']['mostused'] == data['mac']['current'] else 0
    time = int(data['time'].replace(':', ''))
    hour = int(str(time)[:2])
    time_processed = 1 if 8 <= hour < 18 else 0

    processed_data = np.array([[ip_equal, mac_equal, time_processed]])
    return processed_data

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    processed_data = preprocess_data(data)
    prediction = model.predict(processed_data)
    category = 'safe' if prediction < 0.5 else 'suspicious'
```

```
    return jsonify({'result': category})

if __name__ == '__main__':
    app.run(debug=True)
```

ДОДАТОК Г

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JSON Classifier</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            align-items: center;
            margin: 20px;
        }
        #container {
            display: flex;
            justify-content: center;
            align-items: flex-start;
        }
        #json-viewer {
            flex: 1;
            border: 1px solid #000;
            padding: 10px;
            margin-left: 10px;
        }
        #controls {
            flex: 1;
            border: 1px solid #000;
```

```
            padding: 10px;
        }
        #login-container {
            flex: 1;
            border: 1px solid #000;
            padding: 10px;
            margin-right: 10px;
        }
        .control-group {
            margin-bottom: 10px;
        }
        button {
            display: block;
            width: 100%;
            padding: 10px;
            margin-bottom: 10px;
        }
        #result-viewer {
            border: 1px solid #000;
            padding: 10px;
            width: 100%;
            text-align: center;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <div id="container">
        <div id="login-container">
            <div class="control-group">
```

```
            <label>Login:</label>
            <input type="text" id="login" placeholder="Enter your login">
        </div>
        <div class="control-group">
            <label>Password:</label>
            <input type="password" id="password" placeholder="Enter your
password">
        </div>
        <button id="sign-in">Sign In</button>
    </div>
    <div id="controls">
        <div class="control-group">
            <label>IP:</label>
            <select id="ip-select">
                <option value="same">Same</option>
                <option value="different">Different</option>
            </select>
        </div>
        <div class="control-group">
            <label>MAC:</label>
            <select id="mac-select">
                <option value="same">Same</option>
                <option value="different">Different</option>
            </select>
        </div>
        <div class="control-group">
            <label>Time:</label>
            <select id="time-select">
                <option value="morning">Morning</option>
                <option value="afternoon">Afternoon</option>
```

```
            <option value="evening">Evening</option>
            <option value="night">Night</option>
        </select>
    </div>
</div>
<div id="json-viewer"></div>
</div>
<div id="result-viewer">Result will be displayed here.</div>

<script>
    document.addEventListener('DOMContentLoaded', () => {
        const jsonViewer = document.getElementById('json-viewer');
        const ipSelect = document.getElementById('ip-select');
        const macSelect = document.getElementById('mac-select');
        const timeSelect = document.getElementById('time-select');
        const signInButton = document.getElementById('sign-in');
        const loginField = document.getElementById('login');
        const passwordField = document.getElementById('password');
        const resultViewer = document.getElementById('result-viewer');

        const defaultData = {
            "ip": {
                "mostused": "192.168.52.91",
                "current": "192.168.52.91"
            },
            "mac": {
                "mostused": "00:26:57:af:aa:46",
                "current": "00:26:57:af:aa:46"
            },
            "time": "06:14"
```

```
};

function updateJsonViewer() {
    jsonViewer.innerHTML = `
        <div>IP Most Used: ${defaultData.ip.mostused}</div>
        <div>IP Current: ${defaultData.ip.current}</div>
        <div>MAC Most Used: ${defaultData.mac.mostused}</div>
        <div>MAC Current: ${defaultData.mac.current}</div>
        <div>Time: ${defaultData.time}</div>
    `;
}

function getTimeValue(option) {
    switch(option) {
        case 'morning': return '06:14';
        case 'afternoon': return '12:00';
        case 'evening': return '18:00';
        case 'night': return '22:00';
        default: return '06:14';
    }
}

ipSelect.addEventListener('change', () => {
    const value = ipSelect.value;
    defaultData.ip.current = value === 'same' ? defaultData.ip.mostused :
'192.168.52.92';
    updateJsonViewer();
});

macSelect.addEventListener('change', () => {
```

```
            const value = macSelect.value;
            defaultData.mac.current = value === 'same' ? defaultData.mac.mostused
: '00:26:57:af:aa:47';
            updateJsonViewer();
        });

        timeSelect.addEventListener('change', () => {
            const value = timeSelect.value;
            defaultData.time = getTimeValue(value);
            updateJsonViewer();
        });

        signInButton.addEventListener('click', () => {
            const login = loginField.value.trim();
            const password = passwordField.value.trim();

            if (!login || !password) {
                alert('Login and Password cannot be empty.');
                return;
            }

            fetch('/predict', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(defaultData)
            })
            .then(response => response.json())
            .then(data => {
```

```
                resultViewer.textContent = `Result: ${data.result}`;
            })
            .catch(error => {
                console.error('Error:', error);
            });
        });

        updateJsonViewer();
    });
  </script>
</body>
</html>
```