НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи і методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки»

на тему: «Система розпізнавання заборонених для перевезення об’єктів в багажі пасажирів літака»

Виконала:
студентка IV курсу, групи КІ-01
Столярчук Єлизавета Олександрівна _____

Керівник:
асистент кафедри ШІ,
Кухарєв С. О. _____

Консультант з економічного розділу:
професор, д.е.н., проф. кафедри ЕК ФММ,
Шевчук Олена Володимирівна _____

Консультант з нормоконтролю:
фахівець першої категорії кафедри ШІ,
Кравець П.В. _____

Рецензент:
доцент кафедри СП,
Безносик О.Ю. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студентка _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В. о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«31» січня 2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Столярчук Єлизаветі Олександрівні

1. Тема роботи «Система розпізнавання заборонених для перевезення об’єктів в багажі пасажирів літака», керівник роботи Кухарєв Сергій Олександрович, асистент кафедри ШІ, затверджені наказом по університету від «31» травня 2024 р. № 2240-С

2. Термін подання студентом роботи «10» червня 2024 року.

3. Вихідні дані до роботи: ренгтгенівські-знімки багажу з анотацією небезпечних об’єктів (положення на зображенні та клас предмету).

4. Зміст роботи: дослідження небезпеки при авіаподорожах та способи її уникнення, аналіз існуючих технологій, аналіз підходів до оптимізації зображень для спільної задачі, вибір мови програмування та бібліотек розробки, створення архітектури та навчання нейронної мережі для системи розпізнавання та класифікації потенційно небезпечних об’єктів в багажі пасажирів літака, аналіз отриманих результатів, порівняння роботи власного алгоритму з існуючими стандартними алгоритмами, функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу: презентація до захисту роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук Олена Анатоліївна, професор, д. е. н.		

7. Дата видачі завдання «05» лютого 2024 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Вивчення літератури за темою роботи.	15.04.2024 25.04.2024	Виконано
2	Підготовка першого розділу.	25.04.2024 29.04.2024	Виконано
3	Підготовка другого розділу.	30.04.2024 02.05.2024	Виконано
4	Розробка програмного продукту.	03.05.2024 13.05.2024	Виконано
5	Підготовка третього розділу.	14.05.2024 18.05.2024	Виконано
6	Підготовка економічної частини.	19.05.2024 25.05.2024	Виконано
7	Підготовка четвертого розділу.	26.05.2024 30.05.2024	Виконано
8	Оформлення дипломної роботи відповідно до нормконтролю.	31.05.2024 06.06.2024	Виконано
9	Підготовка презентації доповіді.	07.06.2024 10.06.2024	Виконано

Студент Єлизавета СТОЛЯРЧУК

Керівник Сергій КУХАРЄВ

РЕФЕРАТ

Дипломна робота: 99 с., 26 рис., 9 табл., 40 посилань, 1 додаток.

АВІАТРАНСПОРТ, РЕНГТЕН-ЗНІМОК, ЗАПОБІГАННЯ ТЕРОРИЗМУ, НЕЙРОННІ МЕРЕЖІ, РОЗПІЗНАВАННЯ, КЛАСИФІКАЦІЯ, СКРІНІНГ, ОПТИМІЗАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – система контролю безпеки в аеропортах.

Предмет дослідження – використання штучного інтелекту для підвищення ефективності перевірки ручної поклажі та багажу.

Мета роботи – розробити та оцінити ефективність системи, яка використовує штучний інтелект для зменшення ризику помилок при перевірці багажу, зокрема для виявлення небезпечних об’єктів в умовах підвищеної загрози терористичних атак.

Авіаційний транспорт є одним з найбільших роботодавців у світі, підтримуючи 87,7 мільйона робочих місць. Після відкриття повітряного простору України основною проблемою стане висока ймовірність терористичних атак, зумовлена воєнним станом та можливими провокаціями.

Впровадження штучного інтелекту на етапі перевірки багажу є важливим завданням для допомоги працівникам аеропортів, що дозволить зменшити ризик помилок при перевірці багажу. Розроблена архітектура системи включає модуль латерального гальмування та класифікатор підтримуючих векторів. LIM обробляє зображення, виділяючи важливі межі та усуваючи нерелевантні дані. CNN виділяє ключові ознаки зображень, а SVM забезпечує точну класифікацію на основі цих ознак. Зворотне розповсюдження використовується для оптимізації ваг мережі, що підвищує точність.

За результатами нейронна мережа на 75% ефективно визначає небезпечні об’єкти в багажі та має пряме практичне застосування в якості першого етапу перевірки багажу під час скринінгу.

ABSTRACT

Bachelor's thesis: 99 p., 26 figures, 9 tables, 40 references, 1 appendix.

AIR TRANSPORT, X-RAY IMAGE, TERRORISM PREVENTION, NEURAL NETWORKS, RECOGNITION, CLASSIFICATION, SCREENING, OPTIMIZATION, ARTIFICIAL INTELLIGENCE.

The object of the study is the airport security control system.

The subject of research is the use of artificial intelligence to improve the efficiency of hand luggage and baggage screening.

The purpose of the work is to to develop and evaluate the effectiveness of a system that uses artificial intelligence to reduce the risk of errors in baggage screening, in particular to identify dangerous objects in the face of an increased threat of terrorist attacks.

Air transport is one of the largest employers in the world, supporting 87.7 million jobs. After Ukraine's airspace is reopened, the main problem will be the high probability of terrorist attacks due to martial law and possible provocations. Passenger and luggage security at airports prevents dangerous items from being brought on board.

The introduction of artificial intelligence is an important task to help airport employees reduce the risk of errors during baggage checks. The developed system architecture includes a lateral inhibition module and a support vector classifier. LIM processes images by highlighting important boundaries and eliminating irrelevant data. CNN identifies key image features, and SVM provides accurate classification based on these features. Backpropagation is used to optimize the network weights, which improves accuracy.

As a result, the neural network is 75% effective in identifying dangerous objects in luggage and has direct practical application as the first stage of baggage screening.

ЗМІСТ

ВСТУП

Авіаційний транспорт є одним із найбільших роботодавців у світі, підтримуючи 87.7 мільйона робочих місць [1]. З початком повномасштабної війни в Україні повітряний простір для цивільної авіації було закрито. Однак, авіакомпанії активно намагаються змінити ситуацію та відновити роботу сфери цивільної авіації [2].

У разі відкриття цивільного повітряного простору в Україні основною проблемою стане висока ймовірність терористичних атак в авіатранспорті. Це зумовлено воєнним станом та можливими провокаціями з боку Росії, а також тим, що у багатьох громадян України може бути зброя та небезпечні предмети.

Безпека пасажирів і багажу в аеропортах є одним з найважливіших аспектів авіаційної безпеки. Вона запобігає пронесенню на борт літаків предметів і матеріалів, які можуть бути використані для вчинення актів незаконного втручання. Система контролю безпеки включає рентгенівські сканери, прохідні металодетектори та спеціалізоване програмне забезпечення.

Таким чином, важливою задачею є впровадження штучного інтелекту для допомоги працівникам аеропорту, що дозволить зменшити ризик помилок при перевірці ручної поклажі та багажу.

Розроблена архітектура включає в себе два структурні елементи – модуль латерального гальмування та класифікатор підтримуючих векторів. LIM виконує обробку зображень, виділяючи важливі межі та усуваючи нерелевантні дані. Після цього CNN виділяє ключові ознаки зображень, а SVM забезпечує точну класифікацію на основі цих ознак. Окрім того, зворотне розповсюдження використовується для оптимізації ваг мережі для досягнення більш високої точності.

За результатами нейронна мережа на 75% ефективно визначає небезпечні об’єкти в багажі та має пряме практичне застосування в якості першого етапу перевірки багажу під час скринінгу.

Результати роботи було представлено на V Міжнародній науково-практичній конференції “Інформаційні, моделюючі технології, системи та комплекси” (ІМТСК-2024) та опубліковано в збірнику тез [12].

РОЗДІЛ 1 НЕБЕЗПЕКА ПРИ АВІАПОДОРОЖАХ ТА СПОСОБИ ЇЇ УНИКНЕННЯ

1.1 Роль авіаційного транспорту в сучасному світі

Повітряний транспорт забезпечує значні економічні та соціальні вигоди. Він сприяє розвитку туризму, торгівлі, сполучення, стимулює економічне зростання, забезпечує робочі місця, підвищує рівень життя, зменшує рівень бідності, забезпечує зв'язок з віддаленими громадами та дозволяє швидко реагувати на катастрофи.

Авіаційний сектор – один із найбільших роботодавців у світі, забезпечуючи 87,7 мільйонів робочих місць у всьому світі і забезпечуючи 11,3 мільйона прямих робочих місць. Авіація забезпечує 3,5 трильйона доларів світового валового внутрішнього продукту. Якби авіація була країною, вона була б 17-ю за величиною економікою у світі, підтримуючи майже 3,5 трильйона доларів економічного впливу [1].

Авіаперевезення сприяють розвитку туризму. Близько 58% усіх міжнародних туристів подорожують до місця призначення повітряним транспортом. Повітряний транспорт дозволяє людям шукати пригод у нових країнах, відпочивати на тропічних пляжах, налагоджувати ділові стосунки та відвідувати друзів і родину. Оскільки наша глобальна економіка стає все більш взаємопов'язаною, авіація є тим фактором, який об'єднує людей.

З вибухом повномасштабної війни в Україні повітряний простір для цивільної авіації став недоступним. Протягом понад двох років мешканцям не доводиться вибирати повітряний транспорт для подорожей через конфлікт між Україною та Росією. У липні 2023 року компанія Ryanair оголосила, що планує відновити авіарейси з України до кінця року з трьох міст західної України. Ситуація наразі залишається напруженою, але переговори між авіакомпаніями та керівництвами країн тривають [2].

У випадку відновлення цивільного повітряного простору в Україні головною проблемою стане висока ймовірність терористичних атак на борту повітряних суден. Це обумовлено як наявністю військового конфлікту та можливими провокаціями з боку Росії, так і розповсюдженням зброї та небезпечних предметів серед широкого кола українських громадян.

1.2 Небезпека під час авіаподорожей

Повітряний тероризм – це міжнародне явище, і воно зовсім не нове на світовій арені: перший приклад політично мотивованого викрадення реактивного лайнера стався в липні 1968 року, а перший вибух бомби на борту літака стався в травні 1949 року. До 1967 року кількість терористичних атак, спрямованих проти цивільної авіації, була мінімальною. Однак у 1967-77 роках частота таких терористичних атак стрімко зросла. Найбільша кількість терористичних актів проти цивільних авіалайнерів була зафіксована в 1977-86 роках.

Нинішня щорічна частота таких подій приблизно вдвічі менша, ніж на початку 1980-х років, але потенційні наслідки сучасних повітряних терористичних атак є більш драматичними і трагічними. Цілями терористів у таких операціях є пасажирські літаки, аеропорти або офіси авіакомпаній. Події 11 вересня 2001 року показали, що терористи можуть раптово завдати удару по ключових об'єктах – приватних будинках, державних установах, військових об'єктах тощо – в будь-якій точці земної кулі. Руйнування символічних об'єктів може послабити довіру громадян до уряду країни, оскільки держава не здатна забезпечити безпеку власного народу. Атаки 11 вересня також довели, що принаймні деякі терористи без вагань будуть використовувати реактивні лайнери як “живі ракети” для досягнення своїх цілей.

Неодноразові терористичні атаки призвели до посилення заходів авіаційної безпеки, і приблизно 50 мільярдів доларів США щорічно витрачається в усьому світі на стримування або запобігання терористичним атакам на авіацію.

Безпека пасажирів та їх багажу у системі аеропортового контролю є ключовим аспектом забезпечення безпеки повітряних перевезень. Ця система має за мету запобігти проникненню на борт літака предметів та матеріалів, що можуть бути використані для здійснення незаконного втручання. Елементами системи контролю безпеки є рентгенівські сканери, металодетектори та спеціалізоване програмне забезпечення [3].

За період з 2010 по 2020 рік загалом було зафіксовано 338 терористичних актів, з яких найбільша кількість, а саме 235, становили вибухи. На другому місці за кількістю були збройні напади, кількість яких склала 65 випадків [4]. Детальна статистика представлена на рисунку 1.1.

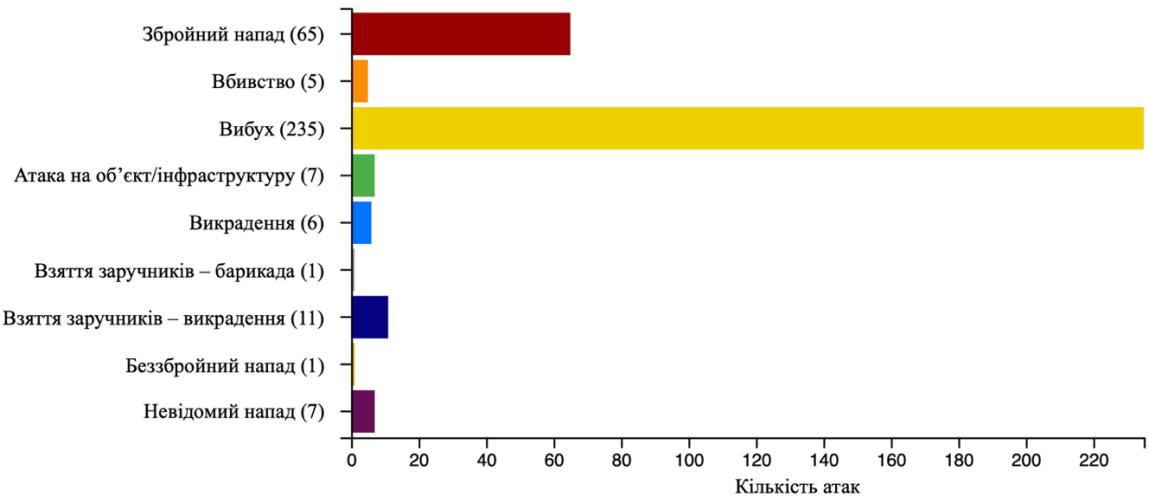

Рисунок 1.1 – Гістограма кількості терористичних атак в авіатранспорті в період з 2010 по 2020 роки (згрупована за типом нападу)

Відповідно найпоширенішими типами використаної зброї стали вибухівка та вогнепальна зброя. З кількостями використання в 258 та 78 разів відповідно.

Ці типи зброї мали бути виявленими на етапі сканування багажу та ручної поклажі пасажирів. Однак ключовим елементом системи контролю на безпеку є людина – інспектор з контролю на безпеку (SSc), яка може допустити помилку при огляді. Графік використаної зброї під час терористиних атак зображено на рис. 1.2, його було сформовано на основі даних з Глобальної бази даних про тероризм (Global Terrorism Database).

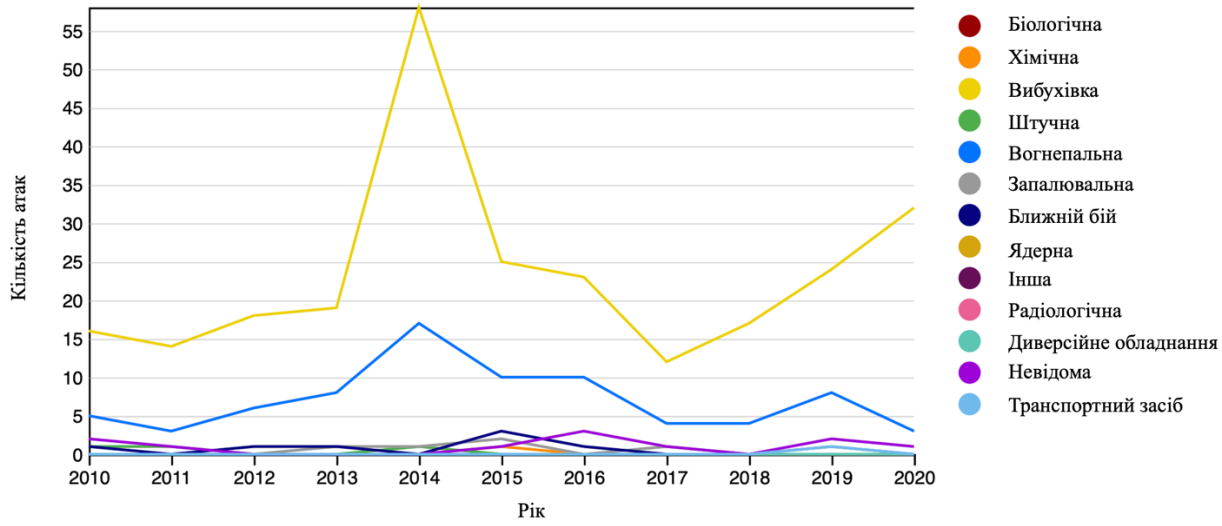

Рисунок 1.2 – Графік кількості терористичних атак в авіатранспорті в період з 2010 по 2020 роки згрупована за типом зброї

1.3 Перевірка багажу за допомогою рентген-сканів

Рентгенівські сканери є важливим компонентом захисту, оскільки вони використовуються для виявлення об'єктів загрози та запобігання їх проникненню в стратегічно важливі будівлі. Хоча існують й інші методи перевірки (терагерцовий, ультразвуковий тощо), рентгенівське випромінювання зазвичай використовується для сканування багажу, посилок і навіть вантажів.

На сьогоднішній день майже всі комерційно доступні системи рентгенівської візуалізації засновані на ослабленні рентгенівського

випромінювання. При цьому використовується різниця між властивостями ослаблення досліджуваної деталі і навколишнього фону для створення контрастності зображення. Властивості ослаблення виражаються за допомогою коефіцієнта ослаблення матеріалу μ.

Контраст ослаблення деталі визначається визначається за формулою 1.1.

$$C_{att} = \frac{I_1 - I_2}{I_1} \quad (2.1)$$

де I_1 – інтенсивність рентгенівського випромінювання в тіні деталі;
I_2 – інтенсивність рентгенівського випромінювання в навколишньому просторі.

Схематичний принцип дії зображено на рис. 1.3. Для того, щоб бути виявленою, деталь повинна зупинити (або відхилитися на кут, достатній для того, щоб вивести її з початкового променя) більше (або менше) рентгенівських променів, ніж навколишній фон.

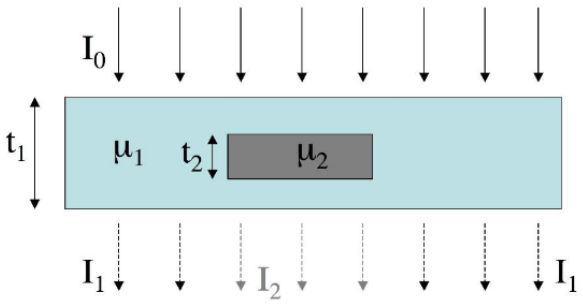

Рисунок 1.3 – Принцип роботи звичайної рентгенівської візуалізації на основі ослаблення [1]

Працівники, які займаються аналізом та інтерпретацією рентгенівських знімків, повинні мати певні знання про основи роботи рентгенівських систем,

[1] Джерело зображення: https://www.researchgate.net/publication/253538472/figure/fig1/AS:298192672444417@1448106081875/Working-principle-of-conventional-attenuation-based-x-ray-imaging-The-detail-must-stop.png

оскільки розуміння цих фундаментальних елементів полегшує завдання рентгенівського скринінгу і, в кінцевому рахунку, підвищує ефективність виявлення.

Перш за все, важливим фактом є те, що рентгенівське зображення є чорно-білим. Також використовується термін “зображення у відтінках сірого”. Саме так більшість людей зазвичай сприймає рентгенівські знімки, оскільки вони схильні думати про чорно-білі рентгенівські знімки, які вони бачили в кабінеті свого лікаря. Апарати, що використовуються для виявлення предметів, показують об'єкти в різних кольорах, щоб полегшити їх розпізнавання.

Для отримання кольорових рентгенівських зображень необхідна технологія двоенергетичної візуалізації (промені високої і низької енергії). Вона дозволяє апарату аналізувати атомний склад різних матеріалів, що містяться в об'єкті. Потім до кожного різного атомного складу застосовується певний колір. Приклад такого зображення наведено на рис. 1.4.

Рисунок 1.4 – Приклад кольорового рентгенівського зображення багажу [5]

Органічні матеріали зображені в помаранчевих відтінках, а метали – в синіх, тоді як неорганічні та змішані матеріали – в зелених. Але існують деякі винятки з цього правила, якщо, наприклад, склад не є на 100% зрозумілим, як у випадку з сумішшю.

Наприклад, цинкові сплави можуть виглядати зеленими, як на рис. 1.5. Аналогічно, різні типи матеріалів, що накладаються один на одного, також можуть бути зображені зеленим кольором.

Рисунок 1.5 – Приклад кольорового рентгенівського зображення автоматичної гвинтівки M4, яка виготовлена з різних сплавів металів [5]

При ідентифікації небезпечних предметів також дуже важливо враховувати, що предмет видно тільки під певним кутом або він змішаний з іншими вантажами, особливо якщо вони накладені один на одного. Приклад різного просторового розміщення зображено на рис. 1.6.

Рисунок 1.6 – Приклад рентгену зброї під різними кутами [5]

Коли об'єкт просвічується рентгеном, частина рентгенівських променів поглинається об'єктом, тоді як інші рентгенівські промені проходять крізь нього. Тому саме щільність і товщина матеріалів впливають на те, наскільки легко рентгенівські промені проходять крізь об'єкти. Чим менша щільність матеріалу, тим прозорішим він є для рентгенівських променів, і тим світлішим і чіткішим буде об'єкт на рентгенівському зображенні. З іншого боку, чим

щільніший і товстіший об'єкт, тим темнішим він буде виглядати на зображенні і тим складнішою буде ідентифікація об'єктів і потенційних аномалій.

Якщо рентгенівське випромінювання не може проникнути крізь предмет, з'являється темно-синій або чорний колір, також відомий як “темна тривога” або “темна зона”, що означає, що матеріал настільки щільний, що рентгенівські промені не можуть проникнути крізь нього в прийнятній мірі, приклад наведено на рис. 1.7. У таких випадках зазвичай рекомендується або повторити сканування під іншим кутом, якщо це можливо, або перейти до ручного огляду.

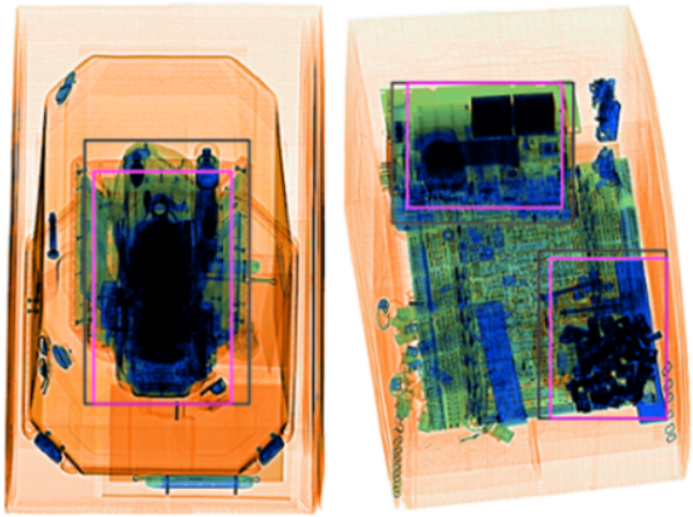

Рисунок 1.7 – Приклад кольорового рентгенівського зображення багажу з наявними темними зонами [5]

Варто зазначити, що точне забарвлення матеріалів може дещо відрізнятися у різних виробників систем, але основні принципи залишаються незмінними.

Незважаючи на розвиток технологій, фактичне рішення про те, чи містить рентгенівське зображення сумки або вантажівки заборонений предмет або аномалію, приймає людина-оператор, рентгенолог. В останні роки основна увага приділяється тому, як підвищити кваліфікацію оператора, оскільки помилки при перевірці рентгенівських зображень можуть призвести до серйозних наслідків.

Зокрема, перевірка транспортних засобів є надзвичайно складним завданням. Вона вимагає аналізу значної кількості даних, зібраних на одному рентгенівському знімку протягом лише кількох хвилин. Ідентифікація заборонених предметів стає ще складнішою, коли мова йде про пропорційно невеликі предмети загрози, такі як зброя, а тим більше, коли зброя була розкладена на частини, навмисно заховані в різних відсіках транспортного засобу або серед вантажу. Що стосується методів приховування, то постійно мінливі способи дій додають ще один рівень складності до і без того складного завдання [5].

Для підвищення ефективності роботи людино-машинної системи необхідно враховувати два фактори: відбір і підготовку рентгенівських сканерів.

Дослідження рентгенівського контролю пасажирів в аеропортах показало, що не кожна людина має потенціал для того, щоб стати хорошим рентгенівським сканером. Для правильної інтерпретації рентгенівських знімків дуже важливими є специфічні здібності до обробки візуальної інформації, такі як мисленнєве обертання, розділення фігури і землі та візуальний пошук специфічних патернів.

Отже, важливим завданням є впровадження штучного інтелекту для підтримки працівників аеропорту, що сприятиме зменшенню можливості помилок при перевірці ручного багажу та вантажу. Це також допоможе знизити ризик терористичних загроз у повітряному транспорті.

1.4 Постановка задачі

Задачею дипломної роботи є розробка та тренування нейронної мережі для аналізу зображень багажу, яка буде виокремлювати потенційно небезпечні об’єкти на знімках та класифікувати їх за категоріями. Окрім того, нейронна

мережа має виділяти “темні зони” як потенційно небезпечні, що в подальшому сигналізуватиме співробітнику аеропорту проінспектувати підозрілу ділянку на предмет небезпеки. Аналіз зображень має відбуватись з початковою оптимізацією.

Висновки до розділу 1

У першому розділі було розглянуто важливість авіаційних подорожей для сучасного суспільства та потенційна небезпека його використання. Було виявлено слабке місце у безпеці. Також, було оглянуто принцип роботи рентгенівських сканерів при перевірці багажу. За допомогою наведених фактів було обгрунтовано мету даної дипломної роботи та актуальність обраної теми досліджень.

РОЗДІЛ 2 ХАРАКТЕРИСТИКА ОБРАНОГО ІНСТРУМЕНТАРІЮ

2.1 Нейронні мережі

Кожна нейронна мережа складається з шарів вузлів, або штучних нейронів – вхідного шару, одного або декількох прихованих шарів і вихідного шару. Кожен вузол з'єднується з іншими і має власну вагу та поріг спрацьовування. Якщо вихід будь-якого окремого вузла перевищує вказане порогове значення, цей вузол активується, надсилаючи дані на наступний рівень мережі. В іншому випадку дані не передаються на наступний рівень мережі [6]. Приклад нейронної мережі зображено на рис. 2.1.

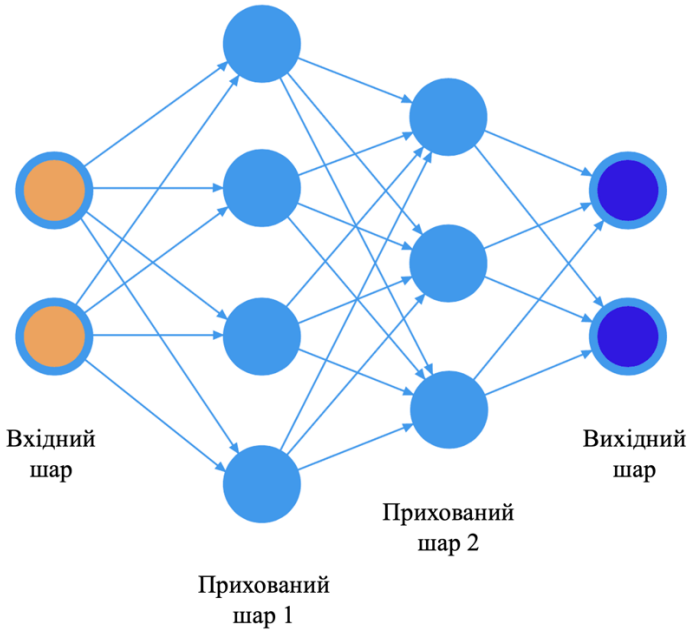

Рисунок 2.1 – Приклад кольорового рентгенівського зображення багажу з наявними темними зонами [6]

Нейронні мережі покладаються на навчальні дані, щоб навчатися і покращувати свою точність з часом. Після того, як вони точно налаштовані на точність, вони стають потужними інструментами в інформатиці та штучному

інтелекті, дозволяючи нам класифікувати і кластеризувати дані з високою швидкістю. Завдання з розпізнавання мовлення або розпізнавання зображень можуть займати хвилини, а не години, якщо порівнювати з ручною ідентифікацією, яку проводять люди-експерти. Одним з найвідоміших прикладів нейронної мережі є пошуковий алгоритм Google.

Нейронні мережі іноді називають штучними нейронними мережами (ШНМ) або імітаційними нейронними мережами (ІМНМ). Вони є підмножиною машинного навчання і лежать в основі моделей глибокого навчання.

Нейронні мережі класифікуються в залежності від їх архітектури та способу структурування. Приклад класифікації зображено на рис. 2.2.

2.2 Глибинне навчання та його методи

Глибинне навчання – це частина машинного навчання, яка використовує багатошарові нейронні мережі, так звані глибокі нейронні мережі, для імітації складної здатності людського мозку до прийняття рішень. Певна форма глибинного навчання лежить в основі більшості систем штучного інтелекту (ШІ) в нашому житті сьогодні.

За строгим визначенням, глибока нейронна мережа, або DNN, – це нейронна мережа з трьома або більше шарами. На практиці більшість DNN мають набагато більше шарів. DNN навчаються на великих обсягах даних, щоб ідентифікувати і класифікувати явища, розпізнавати закономірності і взаємозв'язки, оцінювати можливості, робити прогнози і приймати рішення. У той час як одношарова нейронна мережа може робити корисні, приблизні прогнози і рішення, додаткові шари в глибокій нейронній мережі допомагають уточнити і оптимізувати ці результати для більшої точності.

Глибоке навчання є основою багатьох додатків і сервісів, що автоматизують аналітичні та фізичні процеси без потреби втручання людини.

Воно використовується у повсякденних продуктах і послугах, таких як цифрові асистенти, голосові пультів телевізорів, виявлення шахрайства з кредитними картками, а також у новітніх технологіях, наприклад, у безпілотних автомобілях і генеративному штучному інтелекті.

Основними проблемами навчання глибоких мереж є зникнення градієнтів та вибухове зростання градієнтів, повільне навчання та схильність до перенавчання.

Під час просування алгоритму зворотного розповсюдження помилок у глибоких нейронних мережах, градієнти часто поступово зменшуються по мірі руху від вищих до нижчих шарів. Це може призвести до того, що ваги зв'язків у нижніх шарах залишаються практично незмінними під час градієнтного спуску, що ускладнює навчання та може призвести до невдачі у досягненні оптимального розв'язку.

З іншого боку, може виникнути і протилежна проблема, коли просування алгоритму зворотного розповсюдження помилок до більш нижніх шарів мережі призводить до зростання значень градієнтів. Внаслідок цього наступні шари отримують високі значення приростів ваг, що може призвести до розходження алгоритму градієнтного спуску.

Узагальнюючи, глибокі нейронні мережі можуть страждати від змінливості градієнтів, коли різні шари навчаються з різною швидкістю.

Причиною вказаних проблем може бути невдалий вибір функції активації. Більшість дослідників спочатку вважали, що найкращим варіантом є сигмоїдна функція активації, оскільки вона приблизно відтворює активаційні процеси в біологічних нейронах. Сигмоїдна функція має складності в розрізненні між активацією з великою силою, наприклад, 6 ($\sigma(6) = 0{,}994$ при $\beta = 1$), та активацією з ще більшою силою, наприклад, 10 ($\sigma(10) = 0{,}999$).

Сигмоїдна функція добре розрізняє “достатньо активовані” нейрони, коли їхні результати активації близькі до одиниці, від “недостатньо активованих”, коли ці результати наближені до нуля.

Однак виявилося, що у глибоких нейронних мережах набагато краще справляється функція активації ReLU (Rectified Linear Unit). ReLU має кращі властивості для глибокого навчання через свою простоту та відсутність проблеми з виципленням градієнту у порівнянні з сигмоїдною функцією. Крім того, ReLU може сприяти прискоренню навчання через свою швидку збіжність та здатність уникати проблеми з виципленням градієнту [7].

Поняття “сили активації” краще виражається за допомогою функції ReLU. Розглянемо функцію активації, яка є сумою нескінченного ряду сигмоїдних функцій, представлену у формулі 2.2.

$$S(x) = \sigma(x + 1/2) + \sigma(x - 1/2) + \sigma(x - 3/2) + \cdots \qquad (2.2)$$

де кожен сигмоїд у цій сумі зміщений вправо відносно попереднього на одну одиницю, і нехай β = 1.

Якщо x → −∞, то S → 0. Сигмоїди, розташовані праворуч від нуля із центром правіше від x, будуть швидше прямувати до нуля, тоді як ті, що розташовані ліворуч від x, будуть наближатися до одиниці. Таким чином, активація за допомогою цієї групи сигмоїдів визначає значення x.

Концепцію "сили активації" краще виражає набір нескінченного числа S(x) сигмоїдних функцій, ніж окремий сигмоїд, і може бути апроксимована функціями SoftPlus(x) і ReLU(x)..

По-друге, мережі з нейронами ReLU можуть вміщувати значно більше число нейронів, ніж мережі з більш складними функціями активації при однакових умовах.

По-третє, активація за допомогою ReLU точніше відтворює процеси, які відбуваються у мозку людини. Енергоефективність мозку досягається завдяки розрідженості активації його нейронів. Використовуючи функцію ReLU з регуляризацією, можна досягти розрідженості активації, на відміну від

сигмоїдної функції. Для сигмоїдної функції в будь-який момент часу активується приблизно половина нейронів.

2.3 Згорткові нейронні мережі

Згорткова нейронна мережа (CNN) – це тип нейронних мереж, який спеціалізується на обробці даних з сітчастою структурою, таких як зображення. Цифрове зображення – це бінарне представлення візуальних даних, що складається з набору пікселів, розташованих у формі сітки, де кожен піксель має значення для позначення яскравості та кольору. Наприклад, як показано на рис. 2.2.

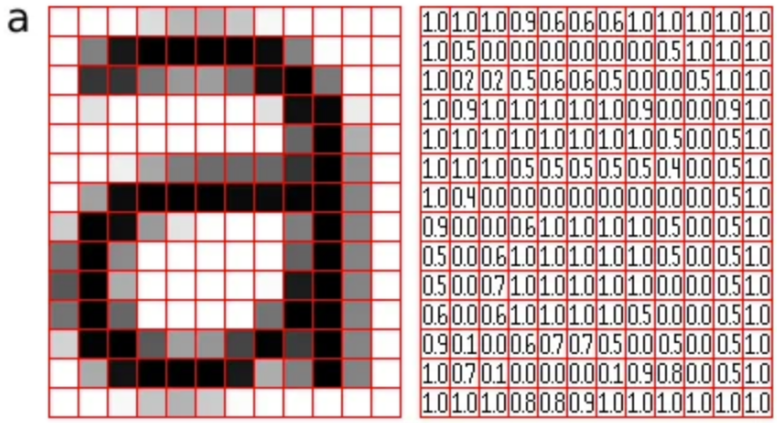

Рисунок 2.2 – Представлення зображення у вигляді сітки пікселів[2]

Людський мозок обробляє величезну кількість інформації в ту секунду, коли ми бачимо зображення. Кожен нейрон працює у своєму власному рецептивному полі і пов'язаний з іншими нейронами таким чином, що вони охоплюють все поле зору. Подібно до того, як кожен нейрон реагує на стимули

[2] Джерело зображення: https://pippin.gimp.org/image_processing/images/sample_grid_a_square.png

лише в обмеженій області зорового поля, що називається рецептивним полем у біологічній системі зору, кожен нейрон у CNN також обробляє дані лише у своєму рецептивному полі. Шари розташовані таким чином, що спочатку вони виявляють простіші патерни (лінії, криві тощо), а далі – складніші (обличчя, об'єкти тощо). Використовуючи CNN, можна зробити зір комп'ютерам.

На відміну від ШНМ, старіші форми нейронних мереж часто обробляли візуальні дані фрагментарно, використовуючи сегментовані або з низькою роздільною здатністю вхідні зображення. Комплексний підхід CNN до розпізнавання зображень дозволяє їм перевершувати традиційні нейронні мережі в низці завдань, пов'язаних із зображеннями, і, меншою мірою, з обробкою мови та аудіо.

CNN зазвичай має три шари: шар згортки, шар об'єднання та повністю з'єднаний шар. Схематичний приклад архітектури зображено на рис. 2.3.

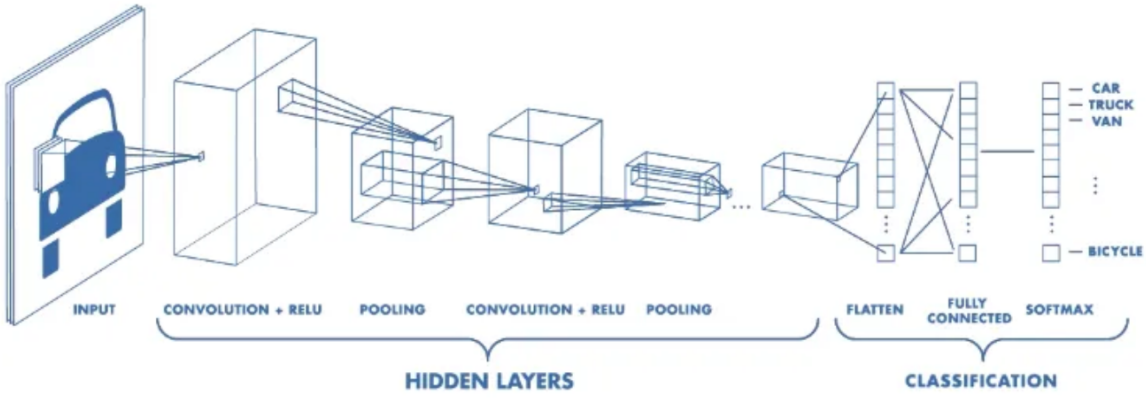

Рисунок 2.3 – Архітектура згорткової нейронної мережі[3]

Згортковий шар є основним структурним елементом CNN і саме в ньому відбувається більшість обчислень. Цей шар використовує фільтр або ядро – невелику матрицю ваг – для переміщення по сприйнятливому полю вхідного зображення, щоб виявити наявність специфічних особливостей.

[3] Джерело зображення: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

Процес починається з переміщення ядра по ширині та висоті зображення, зрештою, за кілька ітерацій воно охоплює все зображення. У кожній позиції обчислюється точковий добуток між вагами ядра та значеннями пікселів зображення під ядром. Це перетворює вхідне зображення на набір карт ознак або згорнутих ознак, кожна з яких представляє наявність та інтенсивність певної ознаки в різних точках зображення [8]. Приклад операції згортки зображено на рис. 2.4.

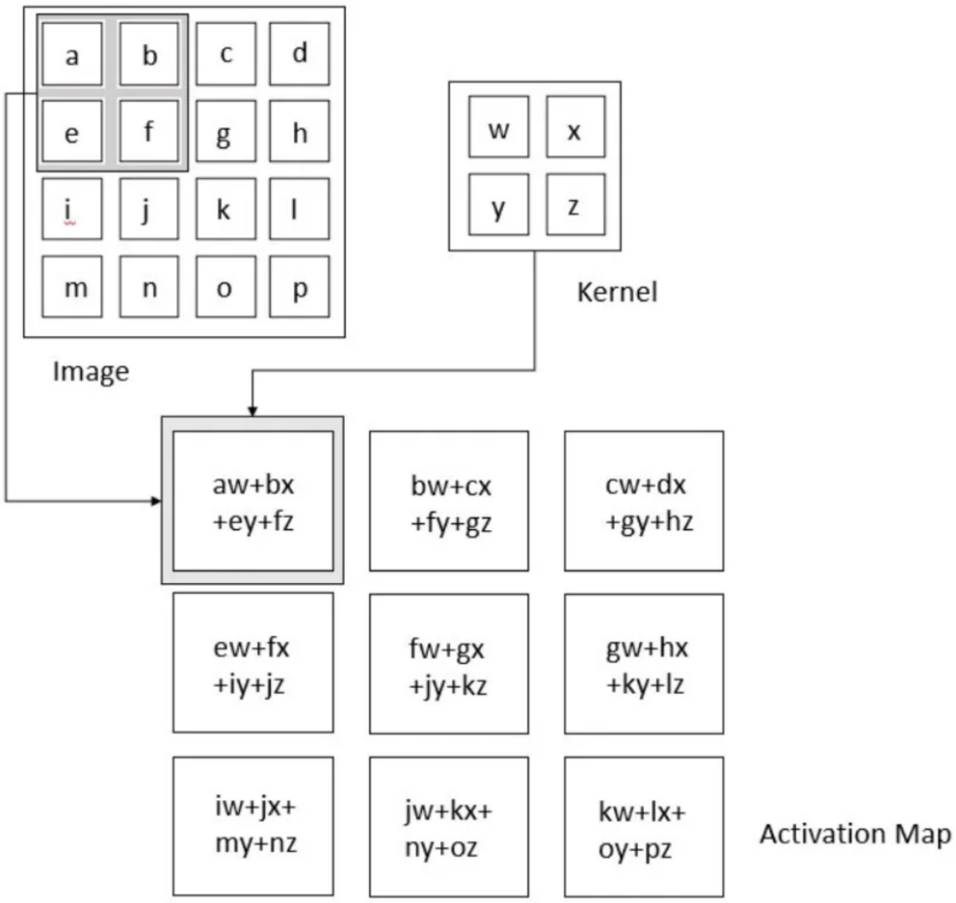

Рисунок 2.4 – Приклад операції згортки [8]

CNN часто складаються з декількох згорнутих шарів, що накладаються один на одного. Завдяки такій багаторівневій архітектурі CNN поступово інтерпретує візуальну інформацію, що міститься у вихідних даних зображення. На ранніх шарах CNN ідентифікує основні характеристики, такі як краї, текстури або кольори. Більш глибокі шари отримують вхідні дані з карт

особливостей попередніх шарів, що дозволяє їм виявляти більш складні патерни, об'єкти і сцени.

Шар пулінгу в CNN є критично важливим компонентом, який слідує за згортковим шаром. Подібно до шару згортки, операції шару об'єднання включають процес розгортання вхідного зображення, але його функція відрізняється.

Мета шару об'єднання – зменшити розмірність вхідних даних, зберігаючи при цьому критично важливу інформацію, таким чином покращуючи загальну ефективність мережі. Зазвичай це досягається шляхом зменшення вибірки: зменшення кількості точок даних на вході.

Для CNN це зазвичай означає зменшення кількості пікселів, що використовуються для представлення зображення. Найпоширенішою формою об'єднання є максимальне об'єднання, яке зберігає максимальне значення в межах певного вікна (тобто розмір ядра), відкидаючи інші значення. Інша поширена методика, відома як середнє об'єднання, використовує подібний підхід, але використовує середнє значення замість максимального.

Даунсемплінг значно зменшує загальну кількість параметрів та обчислень. На додаток до підвищення ефективності, це посилює здатність моделі до узагальнення. Менш складні моделі з високорівневими характеристиками зазвичай менш схильні до перенавчання – явища, яке виникає, коли модель засвоює шум і надто специфічні деталі у своїх навчальних даних, що негативно впливає на її здатність узагальнювати нову, небачену інформацію.

Зменшення просторового розміру представлення має потенційний недолік, а саме втрату частини інформації. Однак для таких задач, як виявлення об'єктів та класифікація зображень, зазвичай достатньо вивчити лише найпомітніші ознаки вхідних даних. Схематичний приклад операції пулінгу зображено на рис. 2.5.

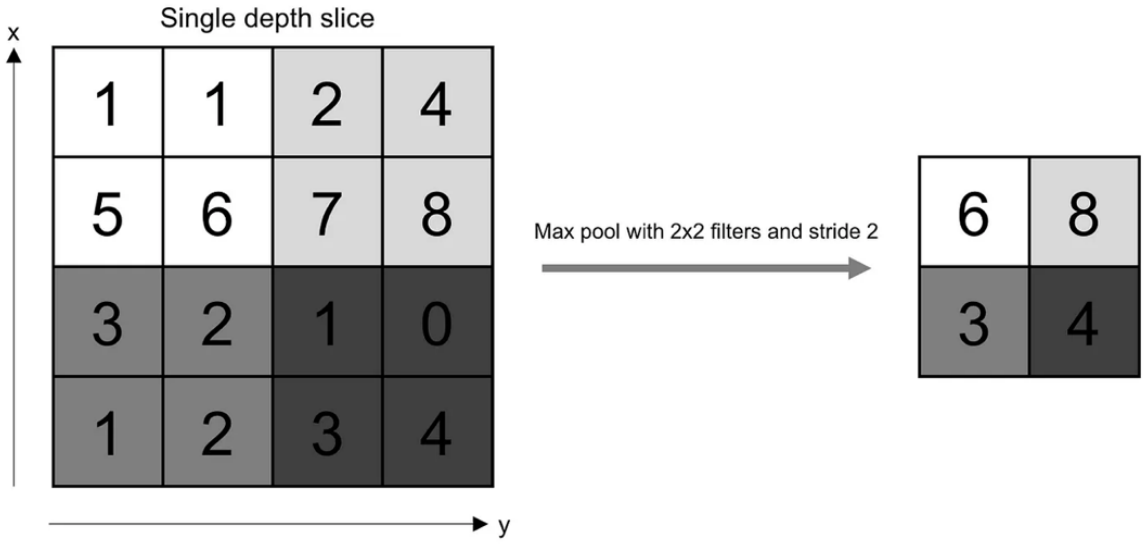

Рисунок 2.5 – Приклад операції пулінгу (об’єднання) [9]

Повністю зв'язаний шар відіграє важливу роль на заключних етапах ШНМ, де він відповідає за класифікацію зображень на основі ознак, вилучених у попередніх шарах. Термін “повністю пов’язаний” означає, що кожен нейрон в одному шарі пов'язаний з кожним нейроном у наступному шарі.

Повністю зв'язаний шар інтегрує різні ознаки, витягнуті з попередніх згорткових шарів і шарів об'єднання, і зіставляє їх з конкретними класами або результатами. Кожен вхід з попереднього шару з'єднується з кожною одиницею активації в повністю пов'язаному шарі, що дозволяє CNN одночасно враховувати всі ознаки при прийнятті остаточного рішення про класифікацію [20].

Не всі шари в CNN є повністю пов'язаними. Оскільки повністю пов'язані шари мають багато параметрів, застосування цього підходу до всієї мережі створить непотрібну щільність, збільшить ризик перенавчання і зробить мережу дуже дорогою для навчання з точки зору пам'яті та обчислень. Обмеження кількості повністю з'єднаних шарів балансує між обчислювальною ефективністю та здатністю до узагальнення і здатністю до навчання складних патернів.

CNN відрізняються від традиційних нейронних мереж кількома ключовими особливостями. Важливо, що в CNN не кожен вузол в шарі з'єднаний з кожним вузлом у наступному шарі. Оскільки їхні згорткові шари мають менше параметрів порівняно з повністю з'єднаними шарами традиційної нейронної мережі, CNN ефективніше справляються із завданнями обробки зображень.

CNN використовують техніку, відому як спільне використання параметрів, що робить їх набагато ефективнішими в обробці даних зображень. У згорткових шарах один і той самий фільтр – з фіксованими вагами – використовується для сканування всього зображення, що значно зменшує кількість параметрів порівняно з повністю зв'язаним шаром традиційної нейронної мережі. Об'єднувальні шари ще більше зменшують розмірність даних, щоб підвищити загальну ефективність та узагальнюваність CNN.

CNN особливо корисні для задач комп'ютерного зору, таких як розпізнавання та класифікація зображень, оскільки вони призначені для вивчення просторових ієрархій ознак шляхом захоплення основних ознак у ранніх шарах і складних патернів у більш глибоких шарах. Однією з найважливіших переваг згорткових нейронних мереж є їхня здатність виконувати автоматичне вилучення ознак або навчання ознак. Це усуває необхідність вилучення ознак вручну, що історично було трудомістким і складним процесом.

CNN також добре підходять для навчання з перенесенням, коли попередньо навчена модель налаштовується для нових завдань. Таке багаторазове використання робить цей вид нейронних мереж універсальними та ефективними, особливо для задач з обмеженими навчальними даними. Побудова на основі вже існуючих мереж дозволяє розробникам машинного навчання розгортати CNN в різних реальних сценаріях, мінімізуючи при цьому обчислювальні витрати [23].

Як описано вище, CNN є більш ефективними в обчислювальному плані, ніж традиційні повністю пов'язані нейронні мережі, завдяки використанню

спільного доступу до параметрів. Завдяки спрощеній архітектурі CNN можна розгортати на широкому спектрі пристроїв, включаючи мобільні пристрої, такі як смартфони, і в сценаріях периферійних обчислень.

Ця ефективність дозволяє CNN виконувати складні завдання, такі як розпізнавання образів і класифікація об'єктів, в режимі реального часу, зберігаючи при цьому високу точність і швидкість роботи. Використання CNN також сприяє зменшенню споживання енергії, що є важливим фактором для пристроїв з обмеженими ресурсами.

2.4 Пакетна нормалізація

Нормалізація – це техніка попередньої обробки, що використовується для стандартизації даних. Якщо не нормалізувати дані перед навчанням, це може викликати проблеми в мережі, значно ускладнюючи процес навчання і знижуючи його швидкість.

Пакетна нормалізація – це техніка нормалізації, яка застосовується між шарами нейронної мережі, а не до вихідних даних. Вона здійснюється вздовж міні-партій, а не на всьому наборі даних. Ця техніка прискорює навчання та дозволяє використовувати більш високі темпи навчання, що полегшує процес навчання [10].

Визначити пакетну нормалізацію можна за допомогою формули 2.3.

$$z^N = \left(\frac{z - m_z}{s_z}\right) \tag{2.3}$$

де m_z – середнє значення виходу нейронів;

s_z – стандартне відхилення виходу нейронів.

На рис. 2.6 проілюстровано нейронну мережу прямого поширення, червоною лінією показано пакетну нормалізацію, яка застосовується безпосередньо перед застосуванням функції активації.

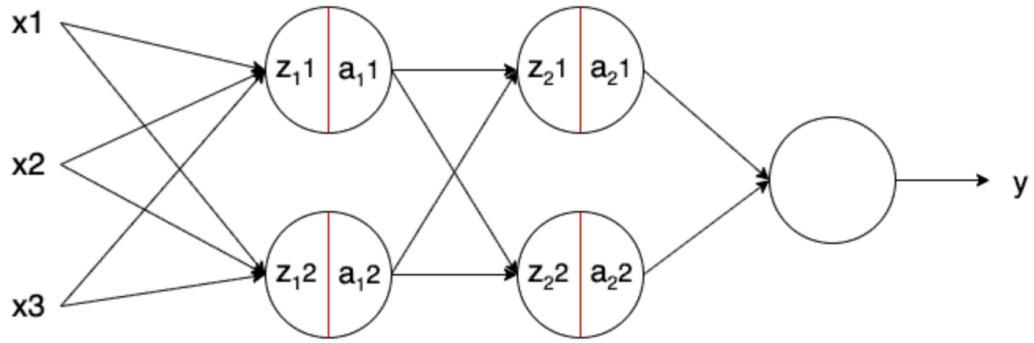

Рисунок 2.6 – Пакетна нормалізація в нейронній мережі прямого поширення, де x_i – входи, z – виходи нейронів, a – вихідні функції активації, y – вихід мережі [10]

Пакетна нормалізація забезпечує очевидну перевагу в тому, що нормалізація дозволяє пришвидшити навчання нейронної мережі та зменшити ймовірність її перенавчання.

По-друге, пакетна нормалізація зменшує коваріаційний зсув мережі. Коваріаційний зсув – це зміна в розподілі даних. Внутрішній коваріаційний зсув – це зміна розподілу вхідних даних у внутрішньому шарі нейронної мережі. Для нейронів внутрішнього шару вхідні дані, отримані від попереднього шару, постійно змінюються. Це пов'язано з численними обчисленнями, зробленими перед ним, і вагами в процесі навчання. Застосування пакетної норми гарантує, що середнє значення і стандартне відхилення входів шару завжди залишатимуться однаковими. Таким чином, кількість змін у розподілі входів шарів зменшується. Глибші шари мають більш надійне підґрунтя щодо того, якими будуть вхідні значення, що допомагає під час процесу навчання [24].

Пакетна нормалізація також має ефект регуляризації. Оскільки вона обчислюється для міні-партій, а не для всього набору даних, розподіл даних, який бачить модель, завжди містить певний рівень шуму. Це може служити

регуляризатором, допомагаючи уникнути перенавчання і покращуючи процес навчання. Проте доданий шум є досить незначним, тому для ефективної регуляризації його часто недостатньо самостійно, і зазвичай його використовують разом з Dropout.

Пакетна нормалізація працює аналогічно і в згорткових нейронних мережах, проте враховуються властивості згортки. У згортках використовуються спільні фільтри, які проходять уздовж карт характеристик вхідних даних (у зображеннях це зазвичай висота і ширина). Ці фільтри однакові на кожній карті характеристик. Тому нормалізація вихідних даних виконується аналогічно, розподіляючи їх по картах характеристик. Це означає, що параметри, які використовуються для нормалізації, обчислюються окремо для кожної карти. У стандартній пакетній нормалізації кожен об'єкт мав би різні середнє значення і стандартне відхилення. У випадку CNN карта об'єктів матиме єдине середнє значення і стандартне відхилення, яке використовується для всіх об'єктів, що містяться на ній [21].

Таким чином, пакетна нормалізація є важливою технікою в глибокому навчанні, яка покращує продуктивність моделі, пришвидшує навчання та допомагає в регуляризації. Вона широко використовується як у прямих, так і в згорткових нейронних мережах для підвищення ефективності навчання та запобігання надмірному пристосуванню.

2.5 Методи розповсюдження в згорткових нейронних мережах

Поширення в згорткових нейронних мережах передбачає оновлення ваг і дельт під час зворотного проходу. Для ваг оновлення обчислюється як добуток швидкості навчання на від'ємний градієнт функції втрат по відношенню до ваг. Для дельт оновлення обчислюється як добуток швидкості навчання на від'ємний градієнт функції втрат по відношенню до дельт.

Під час прямого проходу операція згортки в ШНМ передбачає ковзання ядра по вхідній карті ознак, обчислення добутку між кожним елементом ядра та елементом вхідної карти ознак, який він перекриває. Цей процес повторюється з використанням різних ядер, щоб сформувати стільки вихідних карт ознак, скільки потрібно. Операція згортки гарантує, що ядро робить внесок у всі елементи вихідної карти ознак.

Під час зворотного розповсюдження градієнти обчислюються за правилом ланцюга. Для градієнта втрат відносно входу градієнти обчислюються шляхом згортки фільтра з розширеною версією тензора вихідного градієнта. Для градієнта втрат відносно фільтра градієнти обчислюються шляхом згортки тензора вхідного градієнта з тензором градієнта фільтра [22].

У випадку об'єднання шарів, помилка, як правило, присвоюється “одиниці-переможцю” при максимальному об'єднанні, або множиться на коефіцієнт об'єднання при середньому об'єднанні і присвоюється всьому об'єднаному блоку. Потім градієнти поширюються назад через мережу, оновлюючи ваги і дельти для мінімізації функції втрат.

Рекурентне поширення є високоефективним і гнучким, вимагаючи мінімальної кількості налаштувань, окрім кількості вхідних даних. Це означає, що навіть новачки можуть швидко налаштувати модель для отримання корисних результатів. Крім того, цей метод потребує меншої кількості налаштувань параметрів порівняно з іншими оптимізаційними алгоритмами, що є особливо корисним при роботі з великими мережами. Зменшення кількості необхідних налаштувань параметрів значно полегшує процес розробки та тестування.

Цей алгоритм використовує градієнтне навчання, яке коригує ваги і зсуви, щоб мінімізувати різницю між прогнозованими та фактичними результатами. Він навчається на основі помилок, поширюючи їх назад через мережу, щоб коригувати ваги і зсуви, що зменшує похибку і підвищує

продуктивність з часом. Такий підхід дозволяє моделі поступово покращувати свої прогнози, оскільки вона постійно адаптується до нових даних.

Ці переваги роблять зворотне поширення потужним інструментом для навчання нейронних мереж прямого поширення, включаючи згорткові нейронні мережі, і сприяють його широкому використанню в різних додатках, включаючи класифікацію зображень.

2.6 Оптимізація зображень в задачі виявлення та класифікації об’єктів на рентгенівських зображеннях

Програмне забезпечення рентгенівських сканерів допомагає операторам двома способами. По-перше, воно покращує рентгенівські та КТ-зображення, які оператори аналізують, а по-друге, автоматично виявляє об'єкти на зображеннях. Сучасні алгоритми зосереджені на вдосконаленні та інтерпретації зображень.

Один із найпоширеніших методів покращення зображень полягає у знешумленні та псевдозабарвленні. Для знешумлення використовують гістограми, виявлення країв, розмиття за Гаусом та фільтри Габора. Початкові спроби поєднували низько- та високоенергетичні рентгенівські зображення і застосовували віднімання фону для зменшення шуму. Для адаптивного покращення зображень використовується багатошаровий персептрон, який прогнозує найкращу техніку покращення на основі вхідних та вихідних зображень [25].

Для зображень з низькою щільністю використовується перетворення Радона для вибору порогового значення, що дозволяє видаляти завади з складних рентгенівських знімків. Стандартні схеми псевдозабарвлення використовуються для одноенергетичних зображень, покращуючи ефективність виявлення та настороженість операторів порівняно з зображеннями у відтінках

сірого. Нові схеми кольорового кодування, які використовують оцінку ефективного атомного номера та інформацію про щільність, ще більше підвищують ефективність виявлення загроз [26].

Сучасні методи розпізнавання зображень включають розпізнавання об'єктів та їх виявлення. Ранні роботи зосереджувалися на обробці зображень, використовуючи спрощені піксельні сегментації з фіксованими порогами та групуванням регіонів. Подальші дослідження зосереджувалися на попередній сегментації за допомогою методу найближчого сусіда, видаленні фону та остаточній класифікації. Лу і Коннерс (2006) використовували згладжуючий фільтр для усунення шуму і регіональну сегментацію для виявлення вибухових речовин [27].

Однією з перших робіт з виявлення об'єктів при багаторакурсному рентгенівському скануванні багажу була робота Мері (2011), яка виявляла бритви та леза за допомогою дескриптора Scale Invariant Feature Transform (SIFT) [28]. Цей метод інваріантний до перекладу, масштабу, обертання і стійкий до шуму. Він порівнює SIFT-дескриптори еталонних об'єктів з SIFT-дескрипторами сегментованих областей на зображенні, об'єднуючи знайдені об'єкти з різних точок зору шляхом відстеження розріджених SIFT-об'єктів. Метод досягає 94% точності та 6% хибнопозитивних результатів, хоча тестування проводилося на зображеннях з невеликим захаращенням.

Приклад виявлення зброї на рентгенівському скані багажу зображено на рис. 2.7.

Замість використання інформації про форму та методів, розроблених для зображень видимого спектру, деякі підходи застосовують хімічні властивості рентгенівських знімків (ослаблення). Heitz і Chechik (2010) запропонували метод розділення об'єктів у рентгенівських зображеннях, використовуючи властивість адитивності в логарифмічному просторі, де логарифмічне ослаблення в пікселі є сумою логарифмічних ослаблень усіх об'єктів, через які проходить рентгенівське випромінювання [29].

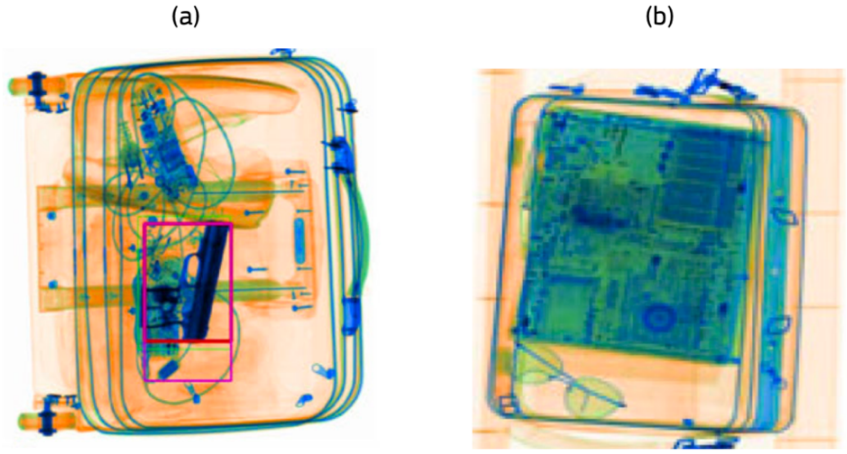

Рисунок 2.7 – Приклад завдання виявлення об'єктів: (а) виявлений пістолет в рамці, і (б) сумка без загрози

Цей метод показав обнадійливі результати, зменшуючи середньоквадратичну похибку на 23% порівняно з двома базовими методами, зазвичай використовуваними для природних зображень: методом сегментації (Felzenszwalb & Huttenlocher, 2004) та кластеризацією за допомогою k-середніх. Проте, цей метод був розроблений для обробки зображень з невеликою кількістю об'єктів і обмежений багажем з низьким рівнем завад [30].

До домінування глибокого навчання в цій галузі, незважаючи на відсутність текстури на рентгенівських знімках, широко використовувався підхід мішків візуальних слів (BoVW), який добре працює з багатими на текстуру зображеннями. BoVW – це метод, в якому локальні особливості з набору зображень об'єднуються в скінченну кількість кластерів. Центроїди кластерів формують словник, який використовується для кодування ознак зображень у векторному квантованому поданні. Центроїди кластерів називаються візуальними словами, а модель "мішок слів" представляє зображення як гістограму цих візуальних слів. Метод складається з п'яти кроків: виявлення ознак, опис ознак, побудова словника, обчислення мішка слів (BoW) і класифікація.

Однією з перших спроб використання BoVW була робота Баштана та ін. (2011), які застосували цей метод для виявлення пістолетів за чотирма ракурсами двоенергетичних рентгенівських зображень сумок на відносно невеликому наборі даних: 52 позитивних і 156 негативних зображень для навчання (52 сумки) і 764 зображення (190 сумок) для тестування [31]. Вони використовували різні детектори ознак: різниця гауссіанів (DoG), Гессен-Лапласа, детектор кутів Харріса, ознаки з методу прискореного сегментного тесту (FAST) та детектори ознак STAR на основі ознак CenSurE. Було проведено експерименти з трьома різними дескрипторами: SIFT (Lowe, 2004) [32], Speeded Up Robust Features (SURF), який є більш обчислювально ефективним, та дескриптор двійкових робастних незалежних елементарних ознак (BRIEF). Для генерації словника використовувався метод k-середніх, а для класифікації – SVM-класифікатор. Результати оцінювали за допомогою показників recall, precision та average precision. Найкращі результати показали детектор DoG і дескриптор SIFT, але продуктивність зростала при використанні двох різних детекторів (Harris та DoG) та при об'єднанні ключових точок з низькоенергетичних, високоенергетичних і кольорових зображень. Вони досягли TPR близько 70% і середньої точності 57%. Приклад зображено на рис. 2.8.

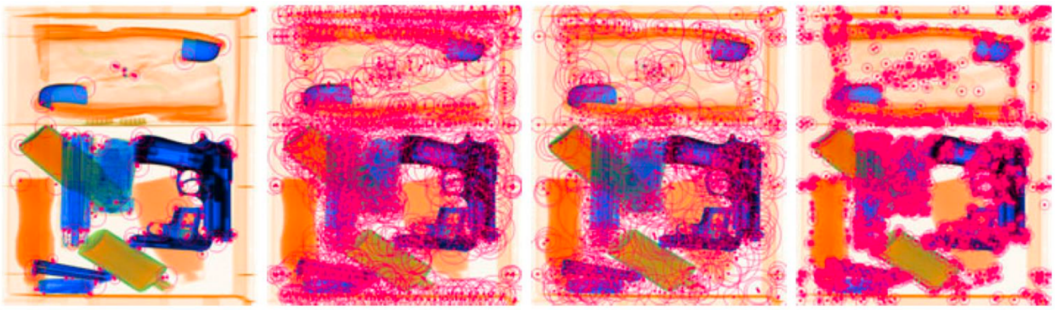

Рисунок 2.8 – Детектори виділених точок на кольоровому рентгенівському зображенні з використанням OpenCV 2.2 реалізацій Harris, DoG від SIFT, Hessian-Laplace від SURF та FAST відповідно

Baştan та ін. (2013) здійснили виявлення трьох рентгенівських об'єктів (ноутбуки, пістолети та скляні пляшки), маючи приблизно в п'ять разів більше негативних зразків, ніж позитивних [33]. На відміну від їхньої попередньої роботи (Baştan та ін., 2011) [31], вони використали інший дескриптор і об'єднали ознаки з кожного окремого вигляду у спільний простір ознак. Використання нового дескриптора зображення SPIN в області інтенсивності, який представляє собою 2D-гістограму значень інтенсивності в круговій околиці точки та є інваріантним до обертання, покращило виявлення. Вони також представили результати за показниками точності, пригадування та середньої точності, дійшовши висновку, що окремі текстурні ознаки працюють недостатньо ефективно, особливо для менш текстурованих об'єктів (наприклад, пляшок). Ефективність значно покращується з додаванням кольорових ознак (інформації про матеріал) і виявлення за кількома видами, особливо коли виявлення за одним виглядом не є оптимальним, як у випадку пістолетів та пляшок. mAP для виявлення пістолетів становила 66%, для ноутбуків – 87%, а для пляшок – 64%.

Baştan та ін. (2011, 2013) [31, 33] і Franzel та ін. (2012) [34] порівнювали ефективність виявлення об'єктів за одним та декількома видами і дійшли висновку, що поєднання декількох видів покращує результати. Натхненні роботою Baştan та ін. (2011) [31], Turcsany та ін. [35] представили унікальний підхід BoWV для класифікації рентгенівської вогнепальної зброї шляхом вилучення специфічних для класу ознак. Вони модифікували традиційну генерацію кодових книг у BoWV, кластеризуючи ознаки окремо для позитивних і негативних класів, як це було зроблено раніше (Perronnin et al., 2006) [36], що полегшувало завдання класифікатора. Це також дозволяло впливати на співвідношення між кількістю візуальних слів для позитивного і негативного класів. Вони згенерували рівну кількість візуальних слів для представлення обох класів.

Використання детектора та дескриптора ознак SURF разом із RBF SVM показало найкращі результати. Метод був протестований на 2500 зображеннях,

з яких було вирізано 850 зображень вогнепальної зброї та 10000 фрагментів зображень, проведено 3-кратну перехресну перевірку за допомогою ROC-кривої, варіюючи розмір ядра σ. Було досягнуто низької FPR в 4% і високої TPR в 99%, що значно покращило показники порівняно з TPR в 70% у Baştan та ін. (2011) [31]. Це покращення пояснюється більшою кількістю навчальних даних, кластеризацією ознак для кожного класу окремо та збільшеною кількістю візуальних слів для кодової книги.

У роботі Mery та ін. (2016) [28] BoVW також використовували для виявлення чотирьох об'єктів: пістолетів, сюрікенів, лез для гоління та скріпок. Як і у Turcsany та ін. (2013) [35], для кожного класу формували словники з дескрипторів ознак випадково обрізаних ділянок зображення SIFT (Lowe, 2004) [32]. Під час навчання кількість вибірок зменшували за допомогою алгоритму k-середніх на вже кластеризованих вибірках. На етапі тестування відбирали найкращі тестові ділянки (найбільш дискримінативні на основі розподілу коефіцієнтів) за допомогою індексу концентрації розрідженості. Кожен відібраний тестовий патч представлявся "адаптивним" розрідженим представленням, обчисленим на основі "найкращого" репрезентативного словника кожного класу. Тестові патчі класифікувалися за методологією "класифікації розрідженого представлення". Тестове зображення класифікувалося за більшістю голосів класів, призначених тестовим патчам. Набір даних складався зі 100 зображень для кожного класу, отриманих із загальнодоступної бази даних GDXray. Використовувалася стратегія "залишити один об'єкт поза увагою": для кожного класу обирали 51 випадково вибране зображення, 50 для навчання і одне для тестування, повторюючи це 100 разів для обчислення середньої точності. Найкращі результати показала комбінація дескрипторів LBP та SIFT, досягаючи понад 95% точності для кожного класу і 85% у випадку оклюзії, яка становила менше 30% об'єкта. Приклад даних зображено на рис. 2.9.

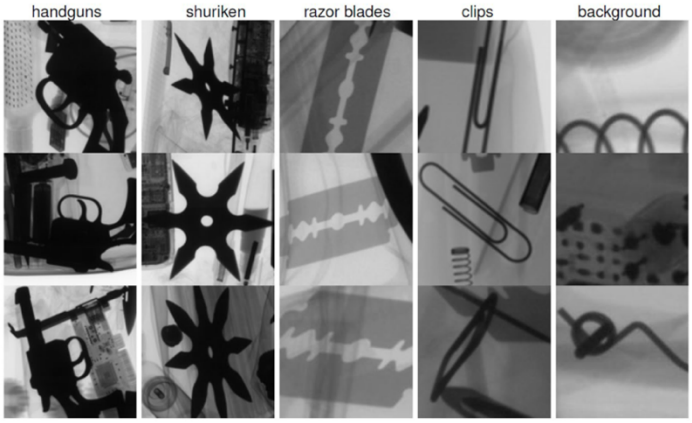

Рисунок 2.9 – Приклад трьох зображень на клас з бази даних GDXray [11]

Дані з оклюзією були створені шляхом заміни пікселів зображення квадратами постійних значень сірої шкали. Варто зазначити, що дослідження проводилося на вже обрізаних об'єктах або фонових зображеннях, замість того, щоб шукати і класифікувати об'єкти в повних зображеннях, що є більш складним завданням.

У своїй роботі Баштан (Baştan, 2015) [33] ретельно проаналізував кілька детекторів ознак (Гарріса-Лапласа, Гарріса-афінний, Гессіана-Лапласа, Гессіана-афінний), щоб оцінити застосовність та ефективність розріджених локальних ознак (SIFT та SPIN) для виявлення об'єктів на рентгенівських знімках багажу, використовуючи підхід "мішка ознак" (Bag-of-Features). В цьому дослідженні також вивчалася роль матеріальної інформації, наданої рентгенівськими зображеннями через кольорове мапування та багаторакурсну рентгенівську візуалізацію, в ефективності виявлення. Було застосовано багаторакурсне виявлення з використанням структурованого навчання (структурні SVM), що покращило просторовий розподіл значень достовірності класифікації. Ця робота є найбільш схожою на дослідження Franzel та ін. [34], але з ключовою відмінністю: замість окремих детекторів для кожного виду та об'єднання їхніх результатів у 3D, Baştan (2015) виконував виявлення безпосередньо в 3D, використовуючи квантовані локальні ознаки з усіх видів зображень, як у їхній роботі 2013 року (Baştan et al., 2013) [31]. Набір даних

включав три види зображень на об'єкт: низькоенергетичне, високоенергетичне та псевдокольорове зображення для виявлення трьох об'єктів загрози: ноутбуків, пістолетів та скляних пляшок. Характерні точки визначалися на псевдокольорових рентгенівських знімках, оскільки кольорові зображення містять більше інформації та текстур порівняно з низько- та високоенергетичними зображеннями, які також є більш зашумленими.

Було використано великий масив даних: 669 сканів (2676 зображень) пістолетів, 250 сканів (1000 зображень) ноутбуків і 280 сканів (720 зображень) скляних пляшок. Для оцінки ефективності виявлення об'єктів використовувалися стандартні показники точності, точності та середньої точності, засновані на критерії PASCAL, що передбачає перекриття 50% площі для правильного розпізнавання. Найкращих результатів було досягнуто за допомогою багатопоглядового та багатофункціонального підходу, який поєднував детектори ознак Гессіана-Лапласа та Гарріса-Лапласа з SIFT, кольоровим SPIN-дескриптором та суперпіксельною дискретизацією. Середня точність виявлення зброї склала 66%, ноутбуків – 87%, а пляшок – 64%. Гірші результати для пістолетів і пляшок пояснюються їхнім безтекстурним зовнішнім виглядом.

Кундегорскі та ін. (2016) всебічно оцінили різні дескриптори характерних точок у задачі класифікації зображень на основі BoVW [37]. Їхній оціночний набір даних складався з майже 20 000 зразків рентгенівських знімків, вже вирізаних з повних зображень (з подвійною енергією, з помилковим кольором, від різних виробників). Комбінація FAST-SURF, навчена SVM-класифікатором, виявилася найкращою комбінацією детектора ознак і дескриптора для завдання виявлення вогнепальної зброї з точністю 0,94, істинно-позитивним результатом 83% і хибно-позитивним результатом 3,3%.

Метод розрідженої реконструкції на основі ШНМ був представлений Svec (2016). Він працював із зображеннями, витягнутими з бази даних GDXray, з об'єктами, що належать до чотирьох класів: пістолети, сюрікени, бритви та фон. Вони використовували ключові точки SIFT, витягнуті із зображення, для

відбору ознак використовувався метод прямого пошуку, а для створення словників для кожного класу використовувалася кластеризація за методом k-середніх. На етапі тестування для кожного класу використовувався класифікатор k найближчих сусідів (KNN), а клас об'єкта прогнозувався за допомогою м'якого голосування. Метод було оцінено з використанням даних GDXray та за допомогою показників точності, пригадування та достовірності. Запропонований метод досяг 97% точності для пістолетів, 99% точності для сюрікенів і 92% точності для бритв на валідаційному наборі (набір, який використовувався для налаштування параметрів). Крім того, дані були обрізані з більших зображень, що містять лише об'єкт, який нас цікавить, як у Mery et al. (2016) [28], що робить завдання класифікації простішим, ніж пошук і розпізнавання об'єктів загрози на повних зображеннях пасажирських сумок.

Інший метод, що використовує базу візуального словника, використовуючи одноракурсні та одноенергетичні зображення сумок з предметами загрози (пістолети, сюрікени, бритви), зображеними в різних позах з використанням повних зображень сумок для тестування. У цій роботі вони використовували стандартну модель SIFT для виявлення та опису ключових точок і запропонували "адаптовану неявну модель форми" (AISM) для кластеризації та виявлення об'єктів. AISM базується на добре відомому методі "неявної моделі форми" (ISM) (Leibe et al., 2008), з тією принциповою різницею, що ISM використовує тільки входження з найвищою оцінкою схожості як достовірне виявлення, в той час як AISM використовує поріг для визначення того, які входження будуть використані [38]. Крім того, AISM не вимагає апріорних знань про кількість об'єктів, які потрібно виявити, на відміну від ISM. Метод було протестовано на 100 лезах для гоління, 100 сюрікенах і 200 пістолетах через більшу міжкласову варіацію для пістолетів. Тестування проводилося на 200 рентгенівських знімках з різною кількістю об'єктів всередині (0-2). Оцінка включала ROC-аналіз, на основі якого було виявлено, що виявлення лез для гоління та сюрікенів було дуже ефективним, що призвело до майже ідеальної ROC-кривої. В обох випадках були отримані високі AUC і

TPR та низька FPR: AUC понад 98%, TPR понад 98% і FPR дорівнює 2% для лез для гоління і 6% для сюрікенів. Результати виявлення пістолетів дещо нижчі: AUC 90%, TPR близько 85% і FPR близько 20%. Враховуючи, що асиметричні об'єкти мають дуже розрізнені входження відносно реального центру об'єкта, найкращі результати отримано для симетричних об'єктів, таких як леза для гоління та сюрікени. Отже, метод не буде дуже ефективним для закритих об'єктів, які, незалежно від їхньої справжньої форми, завжди будуть виглядати асиметричними. Однак цей підхід було розроблено з використанням одноенергетичних зображень і зображень з одного ракурсу, і він не має переваг від багаторакурсного виявлення та двоенергетичних кольорових характеристик.

Незважаючи на домінування BoVW в минулому, інші методи комп'ютерного зору/машинного навчання також вивчалися для класифікації рентгенівських об'єктів. Franzel та ін. (2012) запропонували метод, що використовує стандартне ковзне вікно з гістограмою, орієнтованою на градієнт (HOG), класифікатор SVM і новий метод об'єднання результатів класифікації з кожного з чотирьох поглядів [34]. Автори використовували три раунди бутстрапінгу для SVM і перенавчання, зосереджуючись на складних негативних прикладах. Виявлення декількох поглядів, що вимагає знання геометрії сканера, здійснюється шляхом нанесення ліній у 3D-просторі, і якщо вони перетинаються (або проходять дуже близько один до одного), наприклад, класифікація для заданого обмежувального поля збігається з цими поглядами, то впевненість класифікації для кожного з них збільшується.

Таким чином, кількість хибних виявлень зменшується. Середня точність зросла для виявлення зброї з 46% до 66%, для виявлення ноутбуків з 85% до 86% і для виявлення пляшок з 54% до 64%. Недоліком цього підходу є те, що він втрачає обмежувальні рамки об'єктів і виводить лише центри об'єктів. Тому оцінка ефективності базується на відстані між центрами виявлених і базових об'єктів, а не на загальновживаному перекритті площ обмежувальних рамок об'єктів.

Шмідт-Хакенберг та ін. (2012) запропонували новий метод виявлення зброї, використовуючи ознаки, натхненні зоровою корою людини, SLF-HMAX і V1-подібні, у поєднанні з лінійним SVM-класифікатором [39].

Ознаки SLF-HMAX – це, по суті, вейвлет-фільтри Габора, з додаванням інкорпорованої розрідженості та біологічно вмотивованих ознак складних зорових клітин, тоді як модель V1-подібних ознак представляє прості клітини, як пояснюється в Pinto et al. (2008) [40].

Метод порівнювався з домінуючим на той час підходом, який використовує SIFT і пірамідальну гістограму візуальних слів (PHOW) з кластеризацією за допомогою k-середніх в рамках BoVW і лінійного SVM-класифікатора. Вони використовували стандартний підхід ковзного вікна, застосований до необрізаних багатоенергетичних кольорових одноракурсних зображень.

Отримані результати зображено на рис. 2.10.

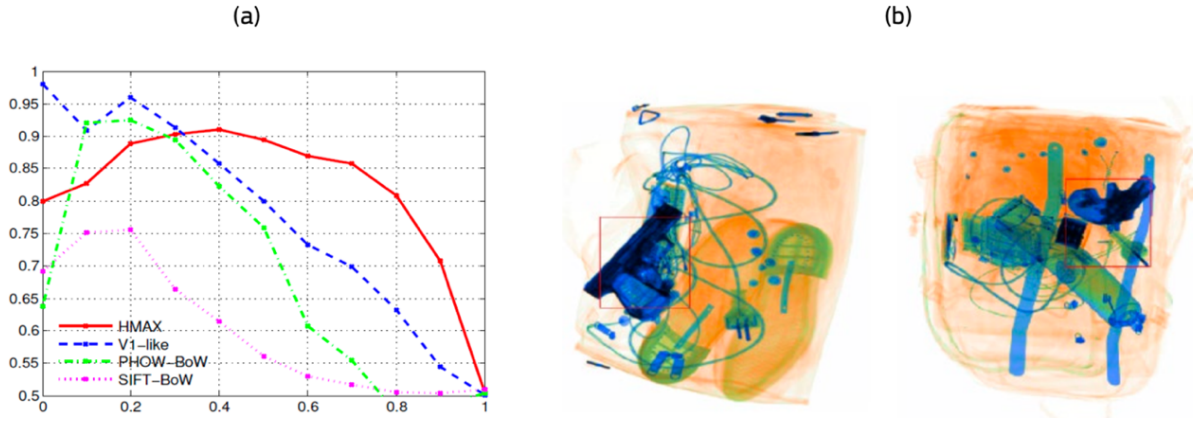

Рисунок 2.10 – (а) Крива "точність-відгук" для всіх протестованих ознак, і (б) правильно виявлений пістолет у двох пакетах з різною орієнтацією

Загальний багатоетапний підхід передбачає виявлення об'єктів у 3D на основі багаторакурсних 2D-зображень за умови відомої геометрії сканера. Подібна обробка 3D-зображень була розглянута в іншій роботі. Цей метод включає: витягування ознак за допомогою дескрипторів і класифікатора k-NN,

зіставлення ключових точок на послідовних зображеннях з різних ракурсів та аналіз багаторакурсних даних, де ключові точки з двох послідовних зображень зіставляються для формування 3D-точок через структуру від руху. Після кластеризації 3D-точки перепроектуються назад у 2D-ключові точки, які потім класифікуються за допомогою класифікатора k-NN. Метод був протестований на 120 зразках, включаючи кліпси, пружини та леза, і показав найкращі результати для 15 зразків лез з 100% точністю і 93% розпізнавання, а також для 75 зразків пружин з 96% точністю і 85% розпізнавання.

Висновки до розділу 2

У другому розділі було розглянуто роботу нейронних мереж та глибинне навчання. За основу було обрано згорткову нейронну мережу та детальніше описано логіку її роботи. Описано методи розповсюдження в нейронних мережах та їхню роль в підвищенні точності. Також, було оглянуто пакетну нормалізацію. В останньому підрозділі було описано різні підходи в оптимізації зображень в задачі виявлення розпізнавання та класифікації небезпечних об’єктів, які було використано в інших роботах.

РОЗДІЛ 3 ОГЛЯД ЗАСОБІВ РОЗРОБКИ ТА РЕЗУЛЬТАТИ РОБОТИ СИСТЕМИ РОЗПІЗНАВАННЯ НЕБЕЗПЕЧНИХ ОБ'ЄКТІВ

3.1 Навчальна вибірка рентгенівських зображень

У якості навчальної вибірки використано набір даних HiXray – набір даних високоякісних рентгенівських зображень. Він містить в собі 45364 зображень знімків багажу з 102928 найпоширенішими забороненими предметами 8 категорій.

Розпізнавання заборонених об'єктів на рентгенівських знімках є дуже важкою задачею для пересічної людини. У HiXray кожен екземпляр аналізувався вручну з анотацією на рівні поля професійними інспекторами служби безпеки аеропорт з великим досвідом роботи. Усі інспектори дотримувалися однакових рекомендацій щодо анотування – що анотувати, як анотувати межі, як анотувати оклюзію тощо. Крім того, точність кожної анотації перевіряв ще один інспектор, включаючи перевірку на наявність пропущених об'єктів для забезпечення вичерпного маркування.

У середньому на одне зображення в наборі даних HiXray припадає 2,27 екземпляра. Статистика розподілу кількості зображень відображена на таблиці 3.1, де категорії "ПЗ1", "ПЗ2", "Вода", "НБ", "МТ", "ПЛ", "КМ" та "НЗ" відповідають «Портативний зарядний пристрій 1», «Портативний зарядний пристрій 2», «Вода», «Ноутбук», «Мобільний телефон», «Планшет», «Косметика» та «Неметалева запальничка»

Таблиця 3.1 – Основні параметри програмного продукту

Категорія	*ПЗ1*	*ПЗ2*	*Вода*	*НБ*	*МТ*	*ПЛ*	*КМ*	*НЗ*
Навчальна вибірка	9919	6216	2471	8046	43204	3921	7969	706
Тестова вибірка	2502	1572	621	1996	10631	997	1980	177
Всього	12421	7788	3092	10042	53835	4918	9949	883

Кількість екземплярів даних в навчальний та тестовій вибірці мають відношення 4:1. Всі зображення зберігаються у форматі JPG з роздільною здатністю в середньому 1200*900. Максимальна роздільна здатність зразків досягає 2000*1040.

Різні моделі рентгенівських апаратів можуть мати деякі відмінності в передачі кольору, що було описано в розділі 1. В наборі даних було використано одну з найбільш класичних стратегій передачі кольору. Кольори об'єктів під рентгенівським випромінюванням в основному визначаються їхнім хімічним складом, який детально представлений у таблиці 3.2.

Таблиця 3.2 – Основні параметри програмного продукту

Колір	*Матеріал*	*Приклади*
Помаранчевий	Органічні матеріали	Пластик, продукти харчування, одяг
Блакитний	Неорганічні матеріали	Ювелірні вироби, скло та кераміка, батареї та акумулятори
Зелений	Суміші органічних та неорганічних матеріалів	Електронні пристрої, годинники

Приклад зображення наведено на рисунку 3.1.

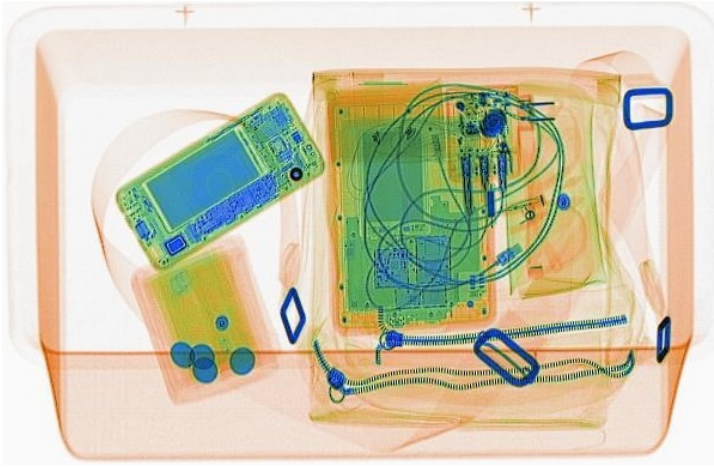

Рисунок 3.1 – Приклад рентгенівського зображення ручної поклажі пасажира з набору даних HiXray

За допомогою функції imread бібліотеки OpenCV відбувається зчитування файлу, після чого за допомогою функції resize було здійснено приведення зображень до одного розміру (300*300). Далі, якщо вказано цільове перетворення, застосовує його до цільових даних. Після чого оброблене зображення було перетворено на тензор PyTorch.

3.2 Архітектура моделі

Модуль Латерального Гальмування (Lateral Inhibition Module, LIM) та Класифікатор Підтримуючих Векторів (Support Vector Machine, SVM) є важливими інструментами в області обробки зображень та машинного навчання. Кожен з них виконує специфічні функції, які можуть доповнювати одна одну, забезпечуючи підвищення точності класифікації та сегментації даних. У цьому розділі розглянемо структуру та функції LIM та SVM, а також їх спільне використання для покращення результатів обробки зображень.

Модуль Латерального Гальмування (LIM) є спеціалізованою структурою нейронної мережі, розробленою для покращення здатності моделі відрізняти різні регіони, зокрема для підвищення точності виявлення меж і ідентифікації об'єктних регіонів. LIM складається з двох основних компонентів: механізму двонаправленої пропагації та активації меж.

Двонаправлена пропагація має на меті фільтрацію шумової інформації та зменшення впливу сусідніх регіонів на об'єктні регіони. Цей механізм дозволяє інформації поширюватися в обох напрямках вздовж меж регіонів, тим самим підсилюючи чіткість меж об'єктів. Це сприяє балансуванню інформації, що надходить від сусідніх регіонів, ефективно пригнічуючи шум та нерелевантні дані. Це зазвичай досягається за допомогою фільтрів або згорткових шарів, які обробляють вхідні дані в прямому і зворотному напрямках. Такий підхід

допомагає згладити переходи між різними регіонами та підсилити контраст на межах.

Активація меж призначена для специфічного виділення та активації меж об'єктів у зображенні або наборі даних. Активація меж робить межі більш помітними та легшими для ідентифікації мережею. Це є важливим для завдань, таких як сегментація, де необхідна точна межа. Активація меж часто включає застосування спеціальних функцій активації або додаткових шарів, чутливих до країв. Ці шари можуть використовувати техніки, такі як градієнтне виявлення країв або інші методи посилення країв, щоб забезпечити чітке позначення меж.

Класифікатор Підтримуючих Векторів (SVM) є потужним інструментом для класифікації даних, який використовується у різних галузях машинного навчання. Основна мета SVM полягає в пошуку оптимальної гіперплощини, яка розділяє дані на різні класи з максимальною відстанню між найближчими точками різних класів (підтримуючими векторами).

SVM працює за допомогою наступних етапів: спочатку здійснюється перетворення вхідних даних у більш високий простір ознак за допомогою ядрових функцій (kernel functions), таких як лінійне, поліноміальне, радикально-базисне (RBF) ядро та інші. Після цього алгоритм шукає гіперплощину в цьому просторі, яка максимально розділяє класи. Важливою особливістю SVM є його здатність ефективно працювати з високовимірними даними та забезпечувати гарну узагальнювальну здатність.

Комбінування LIM та SVM може дати значні переваги у завданнях обробки зображень. LIM може бути використаний для попередньої обробки зображень, покращуючи виділення меж та зменшуючи шум, що створює більш чисті та чіткі ознаки для подальшої класифікації. Після обробки LIM отримані ознаки можуть бути передані на вхід SVM для подальшої класифікації.

Наприклад, у завданнях сегментації зображень LIM може спочатку обробляти зображення, виділяючи важливі межі та усуваючи нерелевантні дані. Потім, використовуючи отримані ознаки, SVM може класифікувати пікселі або

регіони зображення з високою точністю. Це дозволяє досягти більш точних результатів у порівнянні з використанням лише одного з методів.

На практиці LIM та SVM можуть бути інтегровані у різні архітектури нейронних мереж для завдань обробки зображень, таких як виявлення об'єктів, сегментація та класифікація. Покращені можливості виявлення меж та придушення шуму, що надаються LIM, підвищують загальну продуктивність цих моделей, роблячи їх більш стійкими та точними при ідентифікації та визначенні об'єктів у складних зображеннях.

LIM та SVM є потужними інструментами, що доповнюють один одного у завданнях обробки зображень. Використання LIM для попередньої обробки зображень дозволяє покращити якість виділення ознак, що, у свою чергу, підвищує точність класифікації за допомогою SVM. Така комбінація методів забезпечує високу ефективність та точність у різних задачах машинного навчання та комп'ютерного зору.

Розроблена архітектура включає в себе ці два елементи, разом зі згортковою нейронною мережею. LIM виконує обробку зображень, виділяючи важливі межі та усуваючи нерелевантні дані. Після цього CNN виділяє ключові ознаки зображень, а SVM забезпечує точну класифікацію на основі цих ознак. Окрім того, зворотне розповсюдження використовується для оптимізації ваг мережі для досягнення більш високої точності.

3.3 Вибір мови програмування та бібліотек

Python, універсальна та потужна мова програмування, завоювала величезну популярність у різних сферах завдяки своїй простоті, читабельності та великим бібліотекам. Спочатку розроблена Гвідо ван Россумом наприкінці 1980-х років, Python перетворилася на мову, якою користуються як початківці, так і професіонали та дослідники.

Шлях Python почалася на початку 1990-х років, коли він був офіційно випущений як Python 1.0. За ці роки Python зазнав значних оновлень та вдосконалень, остання стабільна версія – Python 3.x. Розвиток мови відбувається під керівництвом Python Software Foundation, що забезпечує її відкритий характер та зростання, кероване спільнотою [13].

Успіх Python можна пояснити кількома ключовими особливостями:

- читабельність – синтаксис Python підкреслює читабельність та простоту, що робить її легкою для вивчення та розуміння;
- широкі бібліотеки – Python може похвалитися великою стандартною бібліотекою та численними сторонніми бібліотеками для різноманітних функцій, таких як аналіз даних (NumPy, Pandas), машинне навчання (TensorFlow, PyTorch), веб-розробка (Django, Flask) та багато іншого;
- інтерпретована природа – це мова, що дозволяє швидко створювати прототипи та інтерактивне кодування;
- крос-платформна сумісність – Python працює на різних платформах, таких як Windows, macOS та Linux, що забезпечує портативність;
- підтримка спільноти – спільнота Python є активною і підтримує, пропонуючи ресурси, навчальні посібники та форуми для розробників усіх рівнів.

Python знаходить застосування в широкому спектрі галузей:

- веб-розробка – такі фреймворки, як Django та Flask, дозволяють швидко розробляти веб-додатки;
- наука про дані (data science) – такі бібліотеки, як NumPy, Pandas, Matplotlib та Scikit-learn роблять Python найкращим вибором для аналізу даних та машинного навчання;
- автоматизація – простота Python робить її ідеальною для написання сценаріїв та автоматизації повторюваних процесів;

- наукові обчислення – такі інструменти, як SciPy та Jupyter Notebooks, ефективно підтримують наукові обчислення;
- освіта – зрозумілість мови Python та її дружній характер для початківців роблять її популярним вибором для навчання концепціям програмування.

З розвитком технологій Python не втрачає своєї актуальності та впливу в різних галузях. Завдяки постійним оновленням, внеску спільноти та потужній екосистемі бібліотек і фреймворків, Python залишається найкращим вибором для розробників, які прагнуть ефективності, продуктивності та універсальності у своїх проектах [14].

Окрім того, для мови програмування Python створено дуже велику кількість зручних бібліотек, що дозволяються створювати та навчати нейронні мережі, що стає вирішальним аргументом в підтримку вибору цієї мови.

PyTorch є однією з найпопулярніших бібліотек для створення та тренування нейронних мереж у Python. Вона пропонує потужний та гнучкий інструментарій для роботи з тензорами, автоматичного обчислення градієнтів та оптимізації моделей [15].

Основна структура даних у PyTorch – це тензор. Тензори нагадують масиви NumPy, але мають додаткову перевагу у вигляді підтримки обчислень на GPU, що значно прискорює процеси навчання моделей. Вони можуть створюватися з різних джерел даних, таких як списки Python або NumPy масиви, і можуть мати довільну кількість вимірів.

Однією з ключових особливостей PyTorch є система автоматичного обчислення градієнтів, що дозволяє ефективно виконувати зворотне розповсюдження помилки (backpropagation). Це досягається завдяки модулю autograd, який автоматично обчислює градієнти тензорів, використовуючи метод зворотного проходу. Таким чином, розробники можуть легко оптимізувати параметри моделей, не турбуючись про складні математичні обчислення [16].

PyTorch також включає модуль nn, який використовується для створення та тренування нейронних мереж. За допомогою цього модуля можна визначати різноманітні шари нейронних мереж та комбінувати їх у складні архітектури. Моделі в PyTorch зазвичай створюються шляхом наслідування класу nn.Module, що дозволяє зручно організовувати та управляти різними компонентами моделі.

Для оптимізації моделей PyTorch пропонує різні алгоритми оптимізації, такі як стохастичний градієнтний спуск (SGD) та його модифікації (Adam, RMSprop тощо). Вони доступні через модуль torch.optim, який надає зручні інтерфейси для налаштування та застосування оптимізаторів до параметрів моделей.

Щоб ефективно тренувати нейронні мережі, важливо правильно організувати процес навчання, включаючи завантаження та обробку даних. Для цього PyTorch має модуль torch.utils.data, який містить класи для створення та управління наборами даних і завантажувачами даних. Це дозволяє легко інтегрувати процеси аугментації даних, батчинг та шифтування даних під час тренування [17].

Таким чином, PyTorch є потужним інструментом для науковців та інженерів, що працюють з машинним навчанням. Його гнучкість та зручний інтерфейс дозволяють швидко розробляти, тренувати та тестувати складні нейронні мережі, забезпечуючи високу продуктивність та ефективність.

Також, NumPy є фундаментальною бібліотекою для наукових обчислень у Python. Вона забезпечує ефективну роботу з багатовимірними масивами і матрицями, а також включає велику кількість високорівневих математичних функцій для обробки цих масивів.

Основна структура даних у NumPy – це ndarray (N-dimensional array), який представляє собою багатовимірний масив однорідних елементів. Цей масив може мати будь-яку кількість вимірів, що дозволяє зручно працювати з різними типами даних, такими як вектори, матриці та тензори. NumPy масиви є дуже ефективними у використанні пам'яті та швидкими у виконанні операцій завдяки реалізації на базі мови C [18].

NumPy надає широкий спектр математичних функцій для роботи з масивами. Це включає базові операції, такі як додавання, віднімання, множення та ділення, а також більш складні операції, такі як лінійна алгебра, статистика та функції для обробки сигналів. Завдяки цим функціям, користувачі можуть виконувати складні обчислення з мінімальним кодом і високою продуктивністю.

Однією з найважливіших особливостей NumPy є можливість роботи з векторизованими операціями. Це означає, що операції можуть бути застосовані до всього масиву одночасно, без необхідності використання циклів. Такий підхід значно підвищує швидкість обчислень і робить код більш чистим та зрозумілим.

NumPy також підтримує численні способи маніпуляції масивами, такі як зміна форми, обрізка, об'єднання та розділення масивів. Це дозволяє легко пристосовувати дані до необхідного формату для подальшої обробки або аналізу. Крім того, бібліотека має зручні інструменти для генерації випадкових чисел, що є корисним для моделювання і тестування.

Завдяки інтеграції з іншими бібліотеками для наукових обчислень та машинного навчання, такими як SciPy, pandas, Matplotlib та інші, NumPy є ключовим компонентом екосистеми Python для наукових досліджень і аналізу даних. Його гнучкість, ефективність та багатий набір функцій роблять його незамінним інструментом для інженерів, науковців та розробників, які працюють з великими обсягами даних та складними математичними моделями [19].

3.4 Результати роботи

Було розроблено програмний код запропонованого алгоритму, під який було адаптовано стандартний Single Shot MultiBox Detector. SSD архітектурою

для виявлення об'єктів на зображеннях у режимі реального часу. Вона об'єднує етапи виділення ознак і виявлення об'єктів в єдину нейронну мережу, дозволяючи швидко і точно визначати межі об'єктів та їх класи. SSD використовує множинні рівні ознак (multi-scale feature maps) і пропонує велику кількість предикторів для кожного рівня ознак, що забезпечує високу ефективність і точність навіть на малих об'єктах.

На рисунках 3.2, 3.3 та 3.4 наведено приклади успішного розпізнавання та класифікації об’єктів на рентгенівських знімках.

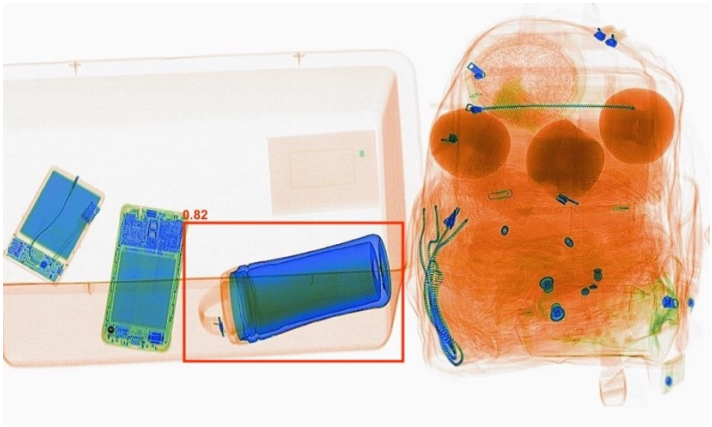

Рисунок 3.2 – Приклад розпізнавання та класифікації об’єкту класу “Вода”

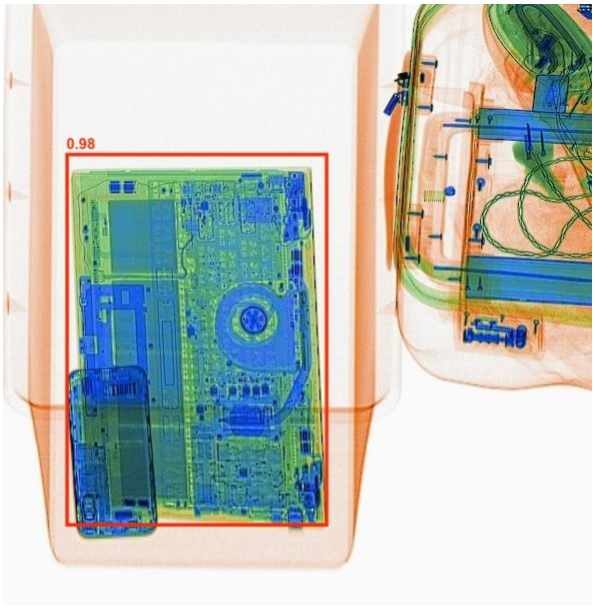

Рисунок 3.3– Приклад розпізнавання та класифікації об’єкту класу “Ноутбук”

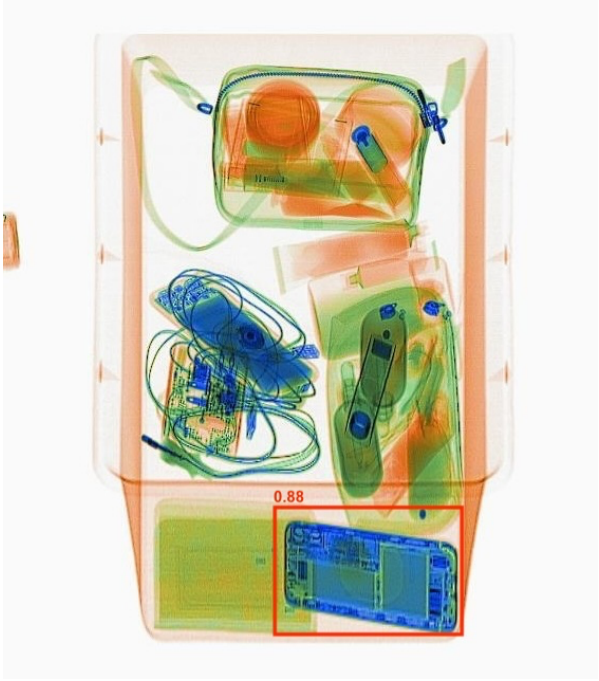

Рисунок 3.4 – Приклад розпізнавання та класифікації об’єкту класу “Портативний зарядний пристрій 1”

В результаті тестування було встановлено, що застосування рекурентного розповсюдження та запропонованої архітектури підвищило відсоток успішного виявлення об’єктів на рентгенівських знімках багажу приблизно на 5%, до середнього показника 75% для кожного типу матеріалів. Модель також враховує випадки з чорними вікнами та зонами затемнення, де об’єкти неможливо розпізнати. Такі області будуть завжди підсвічуватись як потенційно небезпечні.

На таблиці 3.3 зображено результати експериментів з розпізнавання та класифікації об’єктів на рентгенівських знімках на стандартному алгоритмі SSD, використання SSD з прямим розповсюдженням, застосування SSD з рекурентним розповсюдження та модифікація SSD під власну архітектуру. Жирним виділено найкращі отримані показники.

Таблиця 3.3 – Результати класифікації об’єктів

Алгоритм моделі	*Середнє значення*	*ПЗ1*	*ПЗ2*	*Вода*	*НБ*	*МТ*	*ПЛ*	*КМ*	*НЗ*
SSD	70.1	86.6	81.1	81.2	96.5	93.2	91.3	30.9	0.02
SSD + ПР	70.6	87.2	81.5	82.6	**98.3**	94.4	91.5	29.3	0.06
SSD + РР	71.2	88.4	82.2	76.6	97,7	94.5	91.4	38.7	**0.1**
SSD + власна архітектура	74.6	89.5	**83.9**	**84.8**	98.2	**95.1**	**92.7**	**52.5**	**0.1**

Варто зауважити, що показники особливо низькі у двох класах (КМ – косметика та НЗ – неметалева запальничка) через те, що порівняно з іншими категоріями, такі об’єкти набагато складніше розпізнати.

Предмети категорії “Неметалева запальничка” дуже малі за розміром і складаються з невеликого шматка заліза та пластикового корпусу. На рентгенівському знімку пластик має помаранчевий колір, який майже зливається з фоном.

Для предметів з категорії “Косметика” основна проблема полягає в тому, що існують великі відмінності у формах косметичних засобів, таких як круглі та квадратні, які легко сплутати з іншими видами предметів.

Таким чином, розроблена архітектура нейронної мережі з 75% ефективністю виявляє небезпечні об’єкти в багажі, що може бути корисним доповненням для роботи аеропортових співробітників під час скринінгу.

Висновки до розділу 3

У третьому розділі описано вхідний набір рентгенівських знімків HiXray, який було використано для навчання нейронної мережі. Засобами розробки було обрано мову програмування Python та його бібліотеки PyTorch, Numpy,

Scikit Learn та інші. Такий підхід дозволяє максимально швидко та ефективно здійснювати розробку.

Розроблену архітектуру мережі було порівняно зі стандартний алгоритмом SSD, SSD з прямим розповсюдженням та SSD з рекурентним розповсюдження. На результатах було показано, що власна архітектура досягає точності в 75% в задачі розпізнавання та класифікації небезпечних об’єктів.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ СИСТЕМИ РОЗПІЗНАВАННЯ НЕБЕЗПЕЧНИХ ОБ'ЄКТІВ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація. Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

- F_1 – вибір мови програмування;
- F_2 – вибір способу реалізацій алгоритмів;
- F_3 – вибір середовища програмування.

Кожна з цих функцій має декілька варіантів реалізації:

– функція F_1:
  - Python;
  - C++;
– функція F_2:
  - використання готових бібліотек обробки та аналізу зображень;
  - створення власних алгоритмів обробки та аналізу зображень;
– функція F_3:
  - Jupyter Notebook;
  - PyCharm.

Варіанти реалізації основних функцій наведені у морфологічній карті системи на рис. 4.1.

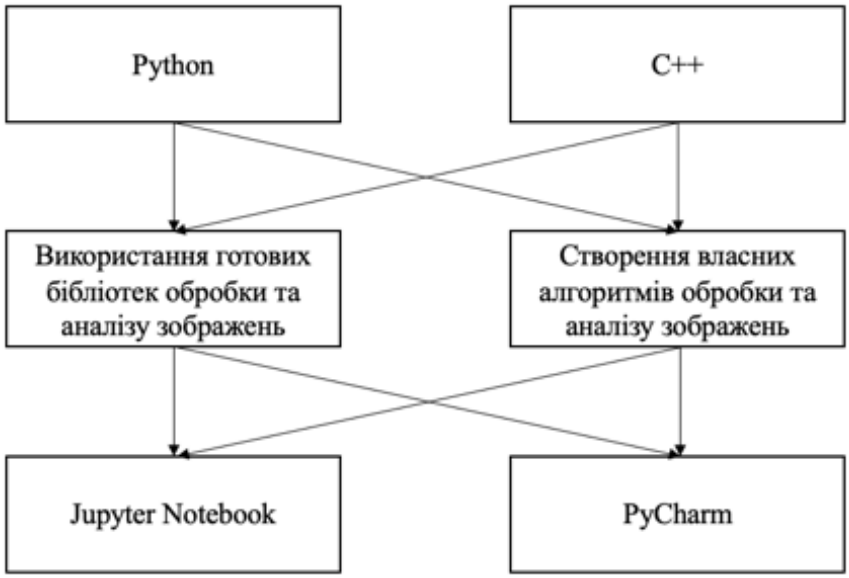

Рисунок 4.1 – Морфологічна карта системи

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функції	*Варіанти реалізації*	*Переваги*	*Недоліки*
F_1	А	Швидка розробка, чіткий і зрозумілий синтаксис, велика стандартна бібліотека, крос-платформеність, широке застосування	Повільна швидкість виконання, велике використання пам'яті, має обмеження у реалізації мультипоточних програм
	Б	Висока продуктивність, контроль над пам'яттю, гнучкість, мультиплатформенність	Проблеми з безпекою пам'яті, складність використання та написання програм
F_2	А	Доступність та легкість у використанні, перевірені та протестовані алгоритми	Менша гнучкість, виконують раніше поставлену задачу
	Б	Реалізація кастомного функціоналу, який потрібен програмі	Великий час реалізації, потреба у додатковому етапі тестування
F_3	А	Інтерактивність середовища, візуалізація помилок, підтримка багатьох мов програмування, легка інтеграція з хмарними сервісами	Обмежені можливості налагодження коду, великі блокноти стають важкими в управлінні
	Б	Автодоповнення коду, підсвічування синтаксису, можливість інтеграції з Git	Ресурсомісткість, вартість, складність освоєння

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1*:* перевагу даємо загальнодоступності та швидкості реалізації; для спрощення роботи по написанню коду варіант Б має бути відкинутий. Функція F_2*:* перевагу даємо доступності в реалізації та меншим часовим

витратам на розробку програмного продукту; для спрощення роботи по написанню коду варіант Б має бути відкинутий. Функція F_3*:* програма допускає обрання обох варіантів; можливо використати варіанти А чи Б.

Таким чином, будемо розглядати такий варіанти реалізації програмного продукту з формул 4.1 та 4.2.

$$F_1 a - F_2 a - F_3 a \quad (4.1)$$

$$F_1 a - F_2 a - F_3 б \quad (4.2)$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X_1 – швидкодія мови програмування;
- X_2 – об’єм пам’яті для обчислень та збереження даних;
- X_3 – час навчання даних;
- X_4 – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметру	*Умовне позначення*	*Одиниці виміру*	*Значення параметра*		
			гірші	*середні*	*кращі*
Швидкодія мови програмування	X_1	млн оп/мс	5	70	200
Об’єм пам’яті	X_2	Гб	10	2	0.7
Час обробки 1-го зображення	X_3	мс	150	100	20
Потенційний об’єм програмного коду	X_4	кількість рядків коду	1000	500	300

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

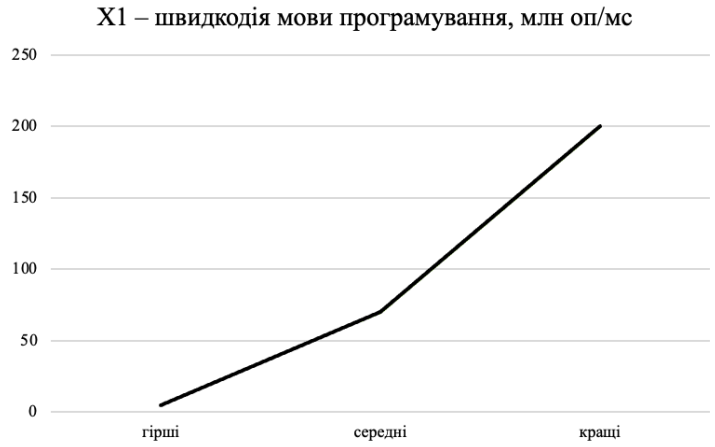

Рисунок 4.2 – Графік значень параметру X1 (швидкодія мови), млн оп/мс

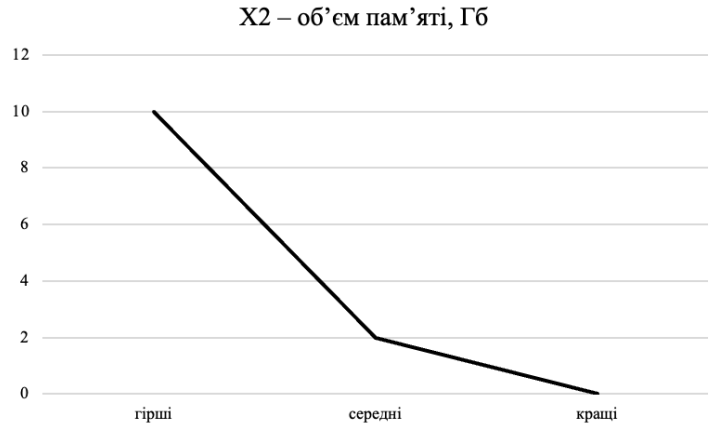

Рисунок 4.3 – Графік значень параметру X2 (об’єм пам’яті), Гб

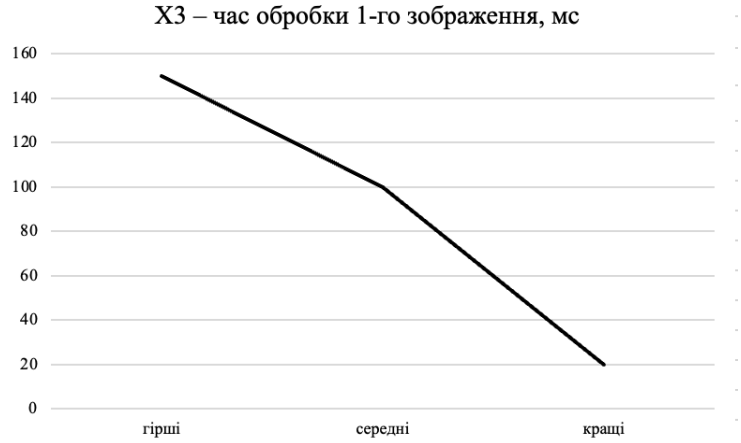

Рисунок 4.4 – Графік значень параметру X3 (час обробки зображення), мс

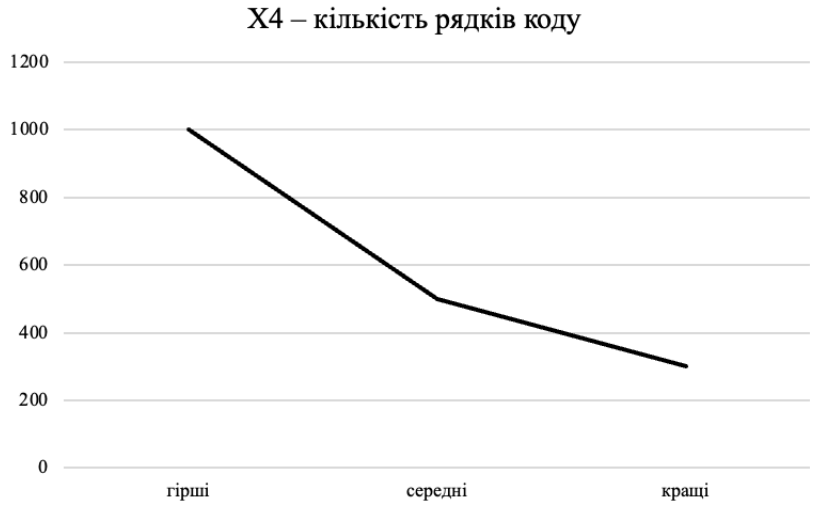

Рисунок 4.5 – Графік значень параметру X4 (кількість рядків коду)

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Основні параметри програмного продукту

Назва параметру та одиниці виміру	*Умовне позначення*	*Ранг параметра за оцінкою експерта*							*Сума рангів* R_i	Δ_i	Δ_i^2
		1	*2*	*3*	*4*	*5*	*6*	*7*			
Швидкодія мови програмування, млн оп/мс	X_1	2	1	1	2	3	2	3	12	-3,5	12,25
Об’єм пам’яті, Гб	X_2	3	3	2	3	2	3	2	19	0,5	0,25
Час обробки 1-го зображення, мс	X_3	1	2	3	1	1	1	1	11	-7,5	56,25
Потенційний об’єм програмного коду, кількість рядків коду	X_4	4	4	4	4	4	4	4	28	10,5	110,25
Разом		10	10	10	10	10	10	10	70	0	179

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів визначається за формулою 4.3:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70 \quad (4.3)$$

де N – число експертів,

n – кількість параметрів

R_{ij} – ранг і-го параметру проставлений j-им експертом

б) середня сума рангів визначається за формулою 4.4:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \quad (4.4)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів за формулою 4.5:

$$\Delta_i = R_i - T \quad (4.5)$$

Сума відхилень по всім параметрам повинна дорівнювати 0.

г) загальна сума квадратів відхилення за формулою 4.6:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 179 \quad (4.6)$$

Порахуємо коефіцієнт узгодженості за формулою 4.7:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 179}{7^2(4^3-4)} = 0{,}7306 > W_k = 0{,}67 \quad (4.7)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Умовне позначення	*Експерти*							*Кінцева оцінка*	*Числове значення*
	1	*2*	*3*	*4*	*5*	*6*	*7*		
X_1 і X_2	<	<	<	<	>	<	>	<	0,5
X_1 і X_3	>	<	<	>	>	>	>	>	1,5
X_1 і X_4	<	<	<	<	<	<	<	<	0,5
X_2 і X_3	>	>	<	>	>	>	>	>	1,5
X_2 і X_4	<	<	<	<	<	<	<	<	0,5
X_3 і X_4	<	<	<	<	<	<	<	<	0,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається за формулою 4.8:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \tag{4.8}$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$. Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за формулами 4.9 та 4.10:

$$K_{\text{в}i} = \frac{b_i}{\sum_{i=1}^{n} b_i} \tag{4.9}$$

$$b_i = \sum_{i=1}^{N} a_{ij} \tag{4.10}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за формулами 4.11 та 4.12:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'} \quad (4.11)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (4.12)$$

На таблиці 4.5 зображено розрахунок вагомості параметрів на трьох ітераціях. На цьому етапі різниця значень коефіцієнтів вагомості параметрів не перевищує 2%, тому більшої кількості ітерацій розрахунків немає потреби проводити.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	*Параметри* x_j				*Перша ітерація*		*Друга ітерація*		*Третя ітерація*	
	X_1	X_2	X_3	X_4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X_1	1	0,5	1,5	0,5	3,5	0,22	12,25	0,21	44,875	0,21
X_2	1,5	1	1,5	0,5	4,5	0,28	16,25	0,28	59,125	0,27
X_3	0,5	0,5	1	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X_4	1,5	1,5	1,5	1	5,5	0,34	21,25	0,35	77,875	0,36
Всього					16	1	59	1	216	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (об'єм пам'яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується за формулою 4.13.

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} \quad (4.13)$$

де n – кількість параметрів;

$K_{вi}$ – коефіцієнт вагомості і-го параметра;

B_i – оцінка і-го параметра в балах.

Розрахунки коефіцієнтів вагомості параметрів та коефіцієнтів рівня якості для основних функцій наведено в таблиці 4.6.

Таблиця 4.6 – Розрахунок вагомості параметрів

Функція	*Варіант реалізації функції*	*Параметер*	*Абсолютне значення параметра*	*Бальна оцінка параметра*	*Коефіцієнт вагомості параметра*	*Коефіцієнт рівня якості*
F_1	А	X_1	70	10	0,21	2,1
F_2	А	X_2	0.7	5	0,27	1,35
F_3	А	X_4	20	9	0,16	1,44
	Б	X_4	300	3	0,36	1,08

За даними таблиці 4.6 можемо використати формулу 4.14 для розрахунку рівня якості кожнго з варіантів.

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}] \quad (4.14)$$

Числові розрахунки двох варіантів наведено в формулах 4.15 та 4.16.

$$K_{K1} = 2{,}1 + 1{,}35 + 1{,}44 = 4{,}89 \quad (4.15)$$

$$K_{K2} = 2{,}1 + 1{,}35 + 1{,}08 = 4{,}53 \quad (4.16)$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня K_{K1} має більше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

- розробка проекту програмного продукту;
- розробка програмної оболонки.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється за формулою 4.17.

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М} \quad (4.17)$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

$K_М$ – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P = 30 людино-днів. Поправочний коефіцієнт, який враховує вид

нормативно-довідкової інформації для першого завдання: $К_П$ = 1.9. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.9. Тоді загальна трудомісткість програмування першого завдання розраховується як $Т_1 = 30 \cdot 1{,}9 \cdot 0{,}9 = 51{,}3$ людино-днів.

Для другого завдання (використовується алгоритм третьої групи складності, ступінь новизни Б) – $Т_Р$ =35 людино-днів, $К_П$ =0.8, $К_{СК}$ = 1, $К_{СТ}$ =0.8. Таким чином, загальна трудомісткість $Т_2 = 35 \cdot 0{,}8 \cdot 0{,}8 = 22{,}4$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість, розрахунки наведено в формулах 4.18 та 4.19.

$$Т_I = (51{,}3 + 22{,}4 + 4{,}8 + 22{,}4) \cdot 8 = 807{,}2 \text{ людино-годин} \quad (4.18)$$

$$Т_{II} = (51{,}3 + 22{,}4 + 7{,}9 + 22{,}4) \cdot 8 = 832 \text{ людино-годин} \quad (4.19)$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 20000 грн., один аналітик в області даних з окладом 25000. Визначимо середню зарплату за годину за формулою 4.20.

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.} \quad (4.20)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

Розрахунки наведено формулою 4.21.

$$С_Ч = \frac{20000+20000+25000}{3 \cdot 21 \cdot 8} = 128{,}97 \text{ грн.} \quad (4.21)$$

Загальна заробітна плата розраховується за формулою 4.22.

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д \text{ грн.} \quad (4.20)$$

де $С_Ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплату розробників за варіантами наведено в формулах 4.21 та 4.22.

$$С_{ЗП\,I} = 128{,}97 \cdot 807{,}2 \cdot 1{,}2 = 124925{,}5 \text{ грн.} \quad (4.21)$$

$$С_{ЗП\,II} = 128{,}97 \cdot 832 \cdot 1{,}2 = 128763{,}65 \text{ грн.} \quad (4.22)$$

Відрахування на єдиний соціальний внесок становить 22%, розрахунки наведено в формулах 4.23 та 4.24.

$$С_{ВІД\,I} = С_{ЗП\,I} \cdot 0{,}22 = 124925{,}5 \cdot 0{,}22 = 27483{,}61 \text{ грн.} \quad (4.21)$$

$$С_{ВІД\,II} = С_{ЗП\,II} \cdot 0{,}22 = 128763{,}65 \cdot 0{,}22 = 28328 \text{ грн.} \quad (4.22)$$

Тепер визначимо витрати на оплату однієї машино-години ($С_М$). Так як одна ЕОМ обслуговує одного програміста з окладом 20000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо результат наведений формулою 4.23.

$$С_Г = 12 \cdot M \cdot К_З = 12 \cdot 20000 \cdot 0{,}2 = 48000 \text{ грн.} \quad (4.23)$$

З урахуванням додаткової заробітної плати отримаємо формулу 4.24.

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 48000 \cdot (1 + 0{,}2) = 57600 \text{ грн.} \quad (4.24)$$

Відрахування на соціальний внесок розраховано за формулою 4.25.

$$С_{ВІД} = С_{ЗП} \cdot 0{,}22 = 57600 \cdot 0{,}22 = 12672 \text{ грн.} \quad (4.25)$$

Амортизаційні відрахування розраховуємо при амортизації 25% та середньої вартості електронно-обчислювальної машини – 18000 грн. Розрахунки наведено формулою 4.26.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1{,}4 \cdot 0{,}25 \cdot 18000 = 6300 \text{ грн.} \quad (4.26)$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$ – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо формулою 4.27.

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1{,}4 \cdot 18000 \cdot 0{,}08 = 2016 \text{ грн.} \quad (4.27)$$

де $К_Р$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою 4.28.

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot К_В = 1{,}4 \cdot 18000 \cdot 0{,}08 = 627{,}2 \text{ год} \quad (4.27)$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою 4.28.

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 627{,}2 \cdot 0{,}2 \cdot 0{,}3 \cdot 5{,}23 = 196{,}82 \text{ грн.} \quad (4.28)$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнт зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії для юридичних осіб.

Накладні витрати розраховуємо за формулою 4.29.

$$С_Н = Ц_{ПР} \cdot 0{,}67 = 18000 \cdot 0{,}67 = 12060 \text{ грн.} \quad (4.28)$$

Тоді, річні експлуатаційні розраховуємо за формулами 4.29 та 4.30.

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \text{ грн.} \quad (4.29)$$

$$С_{ЕКС} = 57600 + 12672 + 6300 + 2016 + 197 + 12060 = 90845 \text{ грн.} \quad (4.30)$$

Собівартість однієї машино-години ЕОМ розраховуємо за формулою 4.31.

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 90890 / 627{,}2 = 144{,}84 \text{ грн.} \quad (4.31)$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, розраховуємо за формулами 4.32 та 4.33.

$$C_{М\,I} = C_{М\text{-}Г} \cdot T_I = 144{,}84 \cdot 807{,}2 = 116914{,}85 \text{ грн.} \quad (4.32)$$

$$C_{М\,I} = C_{М\text{-}Г} \cdot T_{II} = 144{,}84 \cdot 832 = 120506{,}88 \text{ грн.} \quad (4.33)$$

Накладні витрати складають 67% від заробітної плати. Таким чином, можемо розрахувати їх за формулами 4.34 та 4.35 для двох варіантів реалізації ПП.

$$C_{Н\,I} = C_{ЗП\,I} \cdot 0{,}67 = 124925{,}5 \cdot 0{,}67 = 83700{,}09 \text{ грн.} \quad (4.34)$$

$$C_{Н\,I} = C_{ЗП\,I} \cdot 0{,}67 = 128763{,}65 \cdot 0{,}67 = 86271{,}65 \text{ грн.} \quad (4.35)$$

Отже, вартість розробки ПП за варіантами розраховуємо формулами 4.36, 4.37 та 4.38.

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н \quad (4.36)$$

$$C_{ПП\,I} = 124926 + 27484 + 116915 + 83700 = 353025 \text{ грн.} \quad (4.37)$$

$$C_{ПП\,II} = 128764 + 28328 + 120507 + 86272 = 363871 \text{ грн.} \quad (4.38)$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулами 4.39, 4.40 та 4.41

$$K_{ТЕРj} = K_{Кj} / C_{Фj} \quad (4.39)$$

$$К_{ТЕР1} = 4{,}89 \,/\, 353025 = 1{,}385 \cdot 10^{-5} \quad (4.40)$$

$$К_{ТЕР2} = 4{,}53 \,/\, 363871 = 1{,}245 \cdot 10^{-5} \quad (4.41)$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}=1{,}385 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 1{,}385 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– вибір мови програмування Python;
– використання готових бібліотек обробки та аналізу зображень;
– використання Jupyter Notebook як інтерфейсу розробки.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

Висновки до розділу 4

В даній розділі було проведено повний функціонально-вартісний аналіз системи розпізнавання небезпечних об’єктів в багажі пасажирів літака. Також було оцінено основні функції програмного продукту.

В першій частині функціонально-вартісного аналізу було проаналізовано можливі варіанти розробки системи. Було проведено оцінку різних варіантів

реалізації. Внаслідок чого було здійснено розрахунок коефіцієнту технічого рівня та обрано найкращу альтернативу.

Друга частина функціонально-вартісного аналізу була присвячена оцінці вартісної складової реалізації програмного продукту. В рамках неї враховано заробітну плату робітників, витрати на електроенергію та ЕОМ тощо. Результатом було обрано найефективніший варіант, який потребує менших витрат.

ВИСНОВКИ

В даній дипломній роботі було розроблено та досліджено систему розпізнавання небезпечних об’єктів у багажі пасажирів літака за допомогою методів глибинного навчання. Це дослідження є особливо актуальним у світлі сучасних загроз, пов'язаних з тероризмом та іншими злочинами. Забезпечення безпеки авіапасажирів є надзвичайно важливим завданням, і автоматизація процесу виявлення небезпечних об'єктів за допомогою нейронних мереж дозволяє значно підвищити ефективність і швидкість перевірок багажу.

Для реалізації системи було обрано мову програмування Python та бібліотеки PyTorch, Numpy, Scikit Learn, які забезпечують гнучкість та потужні можливості для обробки і аналізу зображень. Використання Jupyter Notebook як інтерфейсу розробки сприяло зручності роботи та візуалізації результатів. У ході дослідження була створена архітектура згорткової нейронної мережі, яка продемонструвала високі показники точності в задачі розпізнавання і класифікації небезпечних об'єктів. Навчання моделі здійснювалося на наборі рентгенівських знімків HiXray, що дозволило досягти точності 75%.

Проведено функціонально-вартісний аналіз різних варіантів реалізації системи. Було розраховано коефіцієнт техніко-економічного рівня (КТЕР) для кожного варіанту та обрано оптимальний. Найкращим виявився варіант з $К_{ТЕР1}=1{,}385 \cdot 10^{-5}$, який поєднує високу ефективність та мінімальні витрати. Аналіз вартості реалізації програмного продукту враховував заробітну плату робітників, витрати на електроенергію та обчислювальне обладнання. Обраний варіант реалізації показав найкращі результати з точки зору економічної ефективності.

Практичне значення даної роботи полягає в тому, що розроблена система може бути впроваджена в аеропортах для автоматичного виявлення небезпечних об'єктів у багажі пасажирів. Це значно підвищить рівень безпеки та зменшить ризики терористичних актів. Виконане дослідження та розробка

підтвердили ефективність застосування методів глибинного навчання для вирішення задач безпеки в авіації, що відкриває нові перспективи для подальшого вдосконалення та впровадження подібних систем у практику.

Таким чином, дане дослідження підтверджує, що використання сучасних технологій і методів глибинного навчання є ефективним засобом для підвищення безпеки в авіаційній галузі. Це відкриває нові можливості для розробки й впровадження високотехнологічних рішень, що сприятимуть забезпеченню безпеки пасажирів та персоналу аеропортів.

Результати роботи було представлено на V Міжнародній науково-практичній конференції “Інформаційні, моделюючі технології, системи та комплекси” (ІМТСК-2024) та опубліковано в збірнику тез [12].

ПЕРЕЛІК ПОСИЛАНЬ

1. H. Dodd, “Supporting economic & social development | ATAG,” atag.org. https://atag.org/industry-topics/supporting-economic-social-development.
2. “Перші авіарейси з України найімовірніше запустять до кінця року з трьох міст – Ryanair,” Економічна правда. https://www.epravda.com.ua/news/ 2023/07/25/702560/ (дата звернення: 18.05.2024).
3. D. Anderson, “Optimising multi-layered security screening,” Journal of Transportation Security, vol. 14, no. 3–4, pp. 249–273, Nov. 2021, doi: https://doi.org/10.1007/s12198-021-00237-3.
4. Global Terrorism Database, “Global Terrorism Database,” University of Maryland, 2022. https://www.start.umd.edu/gtd/.
5. “The challenges of X-ray image analysis and the value of training,” WCO News. https://mag.wcoomd.org/magazine/wco-news-96/the-challenges-of-x-ray-image-analysis-and-the-value-of-training/.
6. S. Cristina, “Calculus in Action: Neural Networks,” Machine Learning Mastery, Aug. 22, 2021. https://machinelearningmastery.com/calculus-in-action-neural-networks/.
7. Н.І. Недашківська, “Методи і моделі аналізу інтелектуального аналізу даних” https://ela.kpi.ua/server/api/core/bitstreams/14d83a9f-88b8-41c9-a81b-342f55dbae94/content (дата звернення: 18.05.2024).
8. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, Massachusetts: The Mit Press, 2016. Available: http://imlab.postech.ac.kr/dkim/class/csed514_2019s/DeepLearningBook.pdf
9. “Pooling operation – Machine Learning with Swift [Book],” www.oreilly.com. https://www.oreilly.com/library/view/machine-learning-with/9781787121515/20e6811a-81ee-47ac-8809-f72808928557.xhtml (дата звернення: 18.05.2024).

10. M. Riva, “Batch Normalization in Convolutional Neural Networks | Baeldung on Computer Science,” www.baeldung.com, Oct. 29, 2020. https://www.baeldung.com/cs/batch-normalization-cnn
11. D. Mery, E. Svec, and M. Arias, “Object Recognition in X-ray Testing Using Adaptive Sparse Representations,” Journal of nondestructive evaluation, vol. 35, no. 3, Jul. 2016, doi: https://doi.org/10.1007/s10921-016-0362-8.
12. Є.О. Столярчук, С.О.Кухарєв, “СИСТЕМА РОЗПІЗНАВАННЯ НЕБЕЗПЕЧНИХ ОБ’ЄКТІВ В БАГАЖІ ПАСАЖИРІВ ЛІТАКА,” in ІНФОРМАЦІЙНІ МОДЕЛЮЮЧІ ТЕХНОЛОГІЇ, СИСТЕМИ ТА КОМПЛЕКСИ (ІМТСК-2024), Черкаси, Україна: Черкаський національний університет імені Богдана Хмельницького, квітень 2024, с. 188–190. URL: https://fotius.cdu.edu.ua/wp-content/uploads/2024/05/Book_IMTCK_2024.pdf
13. Rawat, A. (2020). A Review on Python Programming. International Journal of Research in Engineering, Science and Management, 3(12), 8-11.
14. Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. Information, 11(4), 193.
15. Ketkar, N., Moolayil, J., Ketkar, N., & Moolayil, J. (2021). Introduction to pytorch. Deep learning with python: learn best practices of deep learning models with PyTorch, 27-91.
16. Stevens, E., Antiga, L., & Viehmann, T. (2020). Deep learning with PyTorch. Manning Publications.
17. Pajankar, A., & Joshi, A. (2022). Hands-on Machine Learning with Python: Implement Neural Network Solutions with Scikit-learn and PyTorch. apress.
18. Grout, I. (2020, March). Realization of NumPy Tensordot using the Field Programmable Gate Array for Embedded Machine Learning Applications. In 2020 8th International Electrical Engineering Congress (iEECON) (pp. 1-4). IEEE.

19. Fuhrer, C., Solem, J. E., & Verdier, O. (2021). Scientific Computing with Python: High-performance scientific computing with NumPy, SciPy, and pandas. Packt Publishing Ltd.
20. Chua, L. O. (1998). CNN: A paradigm for complexity (Vol. 31). World Scientific.
21. Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of big Data, 8, 1-74.
22. Li, H., Zhao, R., & Wang, X. (2014). Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. arXiv preprint arXiv:1412.4526.
23. Chua, L. O., & Roska, T. (1993). The CNN paradigm. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 40(3), 147-156.
24. Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. Advances in neural information processing systems, 31.
25. Sinha, T., Verma, B., & Haidar, A. (2017, November). Optimization of convolutional neural network parameters for image classification. In 2017 IEEE symposium series on computational intelligence (SSCI) (pp. 1-7). IEEE.
26. Ren, Y., & Cheng, X. (2019, March). Review of convolutional neural network optimization and training in image processing. In Tenth International Symposium on Precision Engineering Measurements and Instrumentation (Vol. 11053, pp. 788-797). SPIE.
27. Lu, Q., & Conners, R. W. (2006). Using Image Processing Methods to Improve the Explosive Detection Accuracy. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), 36(6), 750–760. https://doi.org/10.1109/TSMCC.2005.855532
28. Mery, D., Riffo, V., Zscherpel, U., Mondragón, G., Lillo, I., Zuccar, I., Lobel, H., Carrasco, M. (2015). GDXray: The Database of X-ray Images for

Nondestructive Testing. Journal of Nondestructive Evaluation, 34(4), 42. https://doi.org/10.1007/s10921-015-0315-7

29. Heitz, G., & Chechik, G. (2010). Object separation in x-ray image sets. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2093–2100. https://doi.org/10.1109/CVPR.2010.5539887
30. Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. International Journal of Computer Vision, 59(2), 167–181. https://doi.org/10.1023/B:VISI.0000022288.19776.77
31. Baştan, M., Yousefi, M. R., & Breuel, T. M. (2011). Visual Words on Baggage X-Ray Images. In P. Real, D. Diaz-Pernil, H. Molina-Abril, A. Berciano, & W. Kropatsch (Eds.), Computer Analysis of Images and Patterns, Vol. 6854, pp. 360–368. https://doi.org/10.1007/978-3-642-23672-3_44
32. Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60(2), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94
33. Baştan, M., Byeon, W., & Breuel, T. (2013). Object Recognition in Multi-View Dual Energy X-ray Images. Proceedings of the British Machine Vision Conference 2013, 130.1-130.11. https://doi.org/10.5244/C.27.130
34. Franzel, T., Schmidt, U., & Roth, S. (2012). Object Detection in Multi-view X-Ray Images. In A. Pinz, T. Pock, H. Bischof, & F. Leberl (Eds.), Pattern Recognition 7476, 144–154. https://doi.org/10.1007/978-3-642-32717-9_15
35. Turcsany, D., Mouton, A., & Breckon, T. P. (2013). Improving feature-based object recognition for X-ray baggage security screening using primed visualwords. 2013 IEEE International Conference on Industrial Technology (ICIT), 1140–1145. https://doi.org/10.1109/ICIT.2013.6505833
36. Perronnin, F., Dance, C., Csurka, G., & Bressan, M. (2006). Adapted Vocabularies for Generic Visual Categorization. In A. Leonardis, H. Bischof, & A. Pinz (Eds.), Computer Vision – ECCV 2006, Vol. 3954, 464–475. https://doi.org/10.1007/11744085_36

37. Kundegorski, M. E., Akçay, S., Devereux, M., Mouton, A., & Breckon, T. P. (2016). On using Feature Descriptors as Visual Words for Object Detection within X-ray Baggage Security Screening. 7th International Conference on Imaging for Crime Detection and Prevention (ICDP 2016), 1–6. https://doi.org/10.1049/ic.2016.0080

38. Leibe, B., Leonardis, A., & Schiele, B. (2008). Robust Object Detection with Interleaved Categorization and Segmentation. International Journal of Computer Vision, 77(1–3), 259–289. https://doi.org/10.1007/s11263-007-0095-3

39. Schmidt-Hackenberg, L., Yousefi, M. R., & Breuel, T. M. (2012). Visual cortex inspired features for object detection in X-ray images. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), 2573–2576. Tsukuba, Japan: IEEE.

40. Pinto, N., Cox, D. D., & DiCarlo, J. J. (2008). Why is Real-World Visual Object Recognition Hard? PLoS Computational Biology, 4(1), e27. https://doi.org/10.1371/journal.pcbi.0040027

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
import torch
import torch.nn as nn

class BoundaryAggregation(nn.Module):
    def __init__(self, in_channels):
        super(BoundaryAggregation, self).__init__()
        self.convolution = nn.Conv2d(in_channels * 5, in_channels, 1)

    def forward(self, input_tensor: torch.Tensor):
        in_channels, height, width = input_tensor.size()[1:4]
        rotated_tensor = torch.rot90(input_tensor, -1, [2, 3])
        vertical_down = self.aggregate_top_to_bottom(input_tensor, in_channels,
height)
        vertical_up = self.aggregate_bottom_to_top(input_tensor, in_channels, height)
        horizontal_right = self.aggregate_left_to_right(rotated_tensor, in_channels,
width)
        horizontal_left = self.aggregate_right_to_left(rotated_tensor, in_channels,
width)
        concatenated_tensor = torch.cat((input_tensor, vertical_down, vertical_up,
horizontal_right, horizontal_left), 1)
        merged_tensor = self.convolution(concatenated_tensor)
        return merged_tensor

    def aggregate_left_to_right(self, rotated_tensor: torch.Tensor, in_channels: int,
height: int):
        x = torch.clone(rotated_tensor)
        x = self.aggregate_top_to_bottom(x, in_channels, height)
        x = torch.rot90(x, 1, [2, 3])
```

```
        return x

    def aggregate_right_to_left(self, rotated_tensor: torch.Tensor, in_channels: int,
height: int):
        x = torch.clone(rotated_tensor)
        x = self.aggregate_bottom_to_top(x, in_channels, height)
        x = torch.rot90(x, 1, [2, 3])
        return x

    def aggregate_bottom_to_top(self, input_tensor: torch.Tensor, in_channels: int,
height: int):
        x = torch.clone(input_tensor)
        for i in range(height - 1, -1, -1):
            x[:, :, i] = torch.max(x[:, :, i:], 2, True)[0].squeeze(2)
        return x

    def aggregate_top_to_bottom(self, input_tensor: torch.Tensor, in_channels: int,
height: int):
        x = torch.clone(input_tensor)
        for i in range(height):
            x[:, :, i] = torch.max(x[:, :, :i + 1], 2, True)[0].squeeze(2)
        return x


if __name__ == "__main__":
    test_tensor = torch.randn(2, 2, 3, 3)
    in_channels = test_tensor.size()[1]
    model = BoundaryAggregation(in_channels)
    output_tensor = model(test_tensor)
    print(output_tensor.shape)
```

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.autograd import Variable
from layers import *
from data import voc, coco
import os
from PIL import Image
import cv2
import time
from boundary_aggregation import BoundaryAggregation

class SSD(nn.Module):
    def __init__(self, phase, size, base, num_classes, head_fpn):
        super(SSD, self).__init__()
        self.phase = phase
        self.num_classes = num_classes
        self.cfg = (coco, voc)[num_classes == 21]
        self.priorbox = PriorBox(self.cfg)
        self.priors = Variable(self.priorbox.forward(), volatile=True)
        self.size = size

        # SSD network
        self.vgg = nn.ModuleList(base)
        self.L2Norm1 = L2Norm(512, 20)
        self.L2Norm3 = L2Norm(256, 40)
        self.L2Norm4 = L2Norm(128, 80)

        self.loc_fpn = nn.ModuleList(head_fpn[0])
```

```
        self.conf_fpn = nn.ModuleList(head_fpn[1])

        # Boundary Aggregation
        self.P3_aggregation = BoundaryAggregation(256)
        self.P4_aggregation = BoundaryAggregation(256)
        self.P5_aggregation = BoundaryAggregation(256)

        self.P3_conv = nn.Conv2d(512, 256, kernel_size=1, stride=1, padding=0)
        self.P4_conv = nn.Conv2d(1024, 256, kernel_size=1, stride=1, padding=0)
        self.P5_conv = nn.Conv2d(2048, 256, kernel_size=1, stride=1, padding=0)
        self.P6_conv = nn.Conv2d(256, 256, kernel_size=3, stride=2, padding=1)
        self.P7_conv = nn.Conv2d(256, 256, kernel_size=3, stride=2, padding=1)

        self.gamma_3 = nn.Parameter(torch.zeros(1))
        self.gamma_4 = nn.Parameter(torch.zeros(1))
        self.gamma_5 = nn.Parameter(torch.zeros(1))

    def forward(self, x):
        sources = list()
        loc = list()
        conf = list()

        for k in range(23):
            x = self.vgg[k](x)
        s = self.L2Norm1(x)
        sources.append(s)

        for k in range(23, len(self.vgg)):
            x = self.vgg[k](x)
        sources.append(x)
```

```
        P3_x = self.P3_conv(sources[0])
        P4_x = self.P4_conv(sources[1])
        P5_x = self.P5_conv(sources[2])

        P3_dual = self.P3_aggregation(P3_x)
        P4_dual = self.P4_aggregation(P4_x)
        P5_dual = self.P5_aggregation(P5_x)

        O3_x = self.gamma_3 * P3_dual + (1 - self.gamma_3) * P3_x
        O4_x = self.gamma_4 * P4_dual + (1 - self.gamma_4) * P4_x
        O5_x = self.gamma_5 * P5_dual + (1 - self.gamma_5) * P5_x

        P6_x = self.P6_conv(P5_x)
        P7_x = self.P7_conv(P6_x)

        return [O3_x, O4_x, O5_x, P6_x, P7_x]



class RegressionModel(nn.Module):
    def __init__(self, num_features_in, num_anchors=9, feature_size=256):
        super(RegressionModel, self).__init__()

        self.conv1 = nn.Conv2d(num_features_in, feature_size, kernel_size=3,
padding=1)
        self.act1 = nn.ReLU()
        self.conv2 = nn.Conv2d(feature_size, feature_size, kernel_size=3, padding=1)
        self.act2 = nn.ReLU()
        self.conv3 = nn.Conv2d(feature_size, feature_size, kernel_size=3, padding=1)
        self.act3 = nn.ReLU()
```

```
        self.conv4 = nn.Conv2d(feature_size, feature_size, kernel_size=3, padding=1)
        self.act4 = nn.ReLU()
        self.output = nn.Conv2d(feature_size, num_anchors * 4, kernel_size=3,
padding=1)

    def forward(self, x):
        out = self.conv1(x)
        out = self.act1(out)

        out = self.conv2(out)
        out = self.act2(out)

        out = self.conv3(out)
        out = self.act3(out)

        out = self.conv4(out)
        out = self.act4(out)

        out = self.output(out)

        out = out.permute(0, 2, 3, 1)
        return out.contiguous().view(out.shape[0], -1, 4)

import os
import time
import argparse
import shutil
import torch
import torch.backends.cudnn as cudnn
from torch.autograd import Variable
```

```
from data import HiXrayDetection, HiXrayAnnotationTransform,
BASE_TRANSFORM, labelmap
from ssd_MuBo import build_ssd

parser = argparse.ArgumentParser(description='SSD Evaluation')
parser.add_argument('--trained_model',
default='weights/ssd300_mAP_77.43_v2.pth', type=str, help='Trained state_dict file
path to open')
parser.add_argument('--save_folder', default='eval/', type=str, help='File path to save
results')
parser.add_argument('--confidence_threshold', default=0.01, type=float,
help='Detection confidence threshold')
parser.add_argument('--top_k', default=5, type=int, help='Further restrict the number
of predictions to parse')
parser.add_argument('--cuda', default=True, type=str2bool, help='Use CUDA to train
model')
parser.add_argument('--HiXray_root', default=VOC_ROOT, help='Location of
HiXray root directory')
parser.add_argument('--cleanup', default=True, type=str2bool, help='Cleanup and
remove results files following eval')
parser.add_argument('--imagesetfile', default='/mnt/cvpr_dataset/test/test_name.txt',
type=str, help='Imageset file path to open')

args = parser.parse_args()

if torch.cuda.is_available():
    if args.cuda:
        torch.set_default_tensor_type('torch.cuda.FloatTensor')
    else:
        torch.set_default_tensor_type('torch.FloatTensor')
```

```
else:
    torch.set_default_tensor_type('torch.FloatTensor')

annopath = os.path.join(args.HiXray_root, 'test_annotation', '%s.xml')
imgpath = os.path.join(args.HiXray_root, 'test_image', '%s.jpg')
devkit_path = args.save_folder
dataset_mean = (104, 117, 123)
set_type = 'test'


class Timer:
    """A simple timer."""

    def __init__(self):
        self.total_time = 0.
        self.calls = 0
        self.start_time = 0.
        self.diff = 0.
        self.average_time = 0.

    def start(self):
        """Start the timer."""
        self.start_time = time.time()

    def stop(self, average=True):
        """Stop the timer and calculate the elapsed time."""
        self.diff = time.time() - self.start_time
        self.total_time += self.diff
        self.calls += 1
        self.average_time = self.total_time / self.calls
```

```
        return self.average_time if average else self.diff


def parse_annotation(filename):
    """Parse a HiXray xml annotation file."""
    filename = filename.replace('.xml', '.txt')
    imagename = filename.replace('annotations', 'images')
    objects = []
    with open(filename, 'r') as f:
        for line in f:
            data = line.strip().split()
            obj_struct = {}
            obj_struct['name'] = data[0]
            obj_struct['pose'] = 'Unspecified'
            obj_struct['truncated'] = int(data[1])
            obj_struct['difficult'] = int(data[2])
            obj_struct['bbox'] = [int(data[3]), int(data[4]), int(data[5]), int(data[6])]
            objects.append(obj_struct)
    return objects


def test_net(save_folder, net, cuda, dataset, transform, top_k, im_size=300,
thresh=0.05):
    num_images = len(dataset)
    all_boxes = [[[] for _ in range(num_images)] for _ in range(len(labelmap) + 1)]

    for i in range(num_images):
        img = dataset.pull_image(i)
        x = torch.from_numpy(transform(img).unsqueeze(0)).permute(0, 3, 1, 2)
        x = Variable(x.cuda()) if cuda else Variable(x)
```

```
        y = net(x).data
        detections = y
        for j in range(1, detections.size(1)):
            dets = detections[0, j, :]
            mask = dets[:, 0].gt(thresh).expand(5, dets.size(0)).t()
            dets = torch.masked_select(dets, mask).view(-1, 5)
            if dets.size(0) == 0:
                continue
            boxes = dets[:, 1:]
            boxes[:, 0] *= img.shape[1]
            boxes[:, 2] *= img.shape[1]
            boxes[:, 1] *= img.shape[0]
            boxes[:, 3] *= img.shape[0]
            scores = dets[:, 0].cpu().numpy()
            cls_dets = np.hstack((boxes.cpu().numpy(), scores[:,
np.newaxis])).astype(np.float32, copy=False)
            all_boxes[j][i] = cls_dets

    with open(os.path.join(save_folder, 'detections.pkl'), 'wb') as f:
        pickle.dump(all_boxes, f, pickle.HIGHEST_PROTOCOL)

    print('Evaluating detections')
    evaluate_detections(all_boxes, save_folder, dataset)


def evaluate_detections(box_list, output_dir, dataset):
    write_voc_results_file(box_list, dataset)
    do_python_eval(output_dir)
```

```
def reset_args():
    global args
    saver_root = 'save/'
    if not os.path.exists(saver_root):
        os.mkdir(saver_root)
    args.save_folder = saver_root + 'best_result_epoch/'

    if not os.path.exists(args.save_folder):
        os.mkdir(args.save_folder)
    else:
        shutil.rmtree(args.save_folder)
        os.mkdir(args.save_folder)

    global devkit_path
    devkit_path = args.save_folder



if __name__ == '__main__':
    reset_args()

    # Load net
    num_classes = len(labelmap) + 1  # +1 for background
    net = build_ssd('test', 300, num_classes)  # initialize SSD
    net.load_state_dict(torch.load(args.trained_model))
    net.eval()

    # Load data
    dataset = HiXrayDetection(args.HiXray_root, args.imagesetfile,
HiXrayAnnotationTransform(), phase='test')
    if args.cuda:
```

```
        net = net.cuda()

import os
import torch
import argparse
from torch.utils.data import DataLoader
from ssd_MuBo import build_ssd
from data import HiXrayDetection, HiXrayAnnotationTransform,
BASE_TRANSFORM

def main_training():
    parser = argparse.ArgumentParser(description='Train SSD model')
    parser.add_argument('--root_dir', default=VOC_ROOT, help='Path to dataset root
directory')
    parser.add_argument('--epochs', default=100, type=int, help='Total number of
training epochs')
    parser.add_argument('--batch', default=32, type=int, help='Training batch size')
    parser.add_argument('--lr', default=1e-3, type=float, help='Starting learning rate')
    parser.add_argument('--decay', default=5e-4, type=float, help='Weight decay rate')
    parser.add_argument('--use_cuda', default=True, type=bool, help='Enable CUDA
for training')
    parser.add_argument('--output_dir', default='weights/', type=str, help='Directory to
save model checkpoints')
    args = parser.parse_args()

    if torch.cuda.is_available() and args.use_cuda:
        torch.set_default_tensor_type('torch.cuda.FloatTensor')
    else:
        torch.set_default_tensor_type('torch.FloatTensor')
```

```
    dataset = HiXrayDetection(args.root_dir, transform=BASE_TRANSFORM(300,
(104, 117, 123)))
    data_loader = DataLoader(dataset, batch_size=args.batch, shuffle=True,
num_workers=4)

    ssd_network = build_ssd('train', 300, 21)
    model = ssd_network

    if args.use_cuda:
        model = model.cuda()

    optimizer = torch.optim.SGD(model.parameters(), lr=args.lr, momentum=0.9,
weight_decay=args.decay)
    loss_function = nn.MultiBoxLoss()

    model.train()

    for epoch in range(args.epochs):
        for step, (images, targets) in enumerate(data_loader):
            if args.use_cuda:
                images = images.cuda()
                targets = [target.cuda() for target in targets]

            optimizer.zero_grad()
            outputs = model(images)
            loc_loss, conf_loss = loss_function(outputs, targets)
            total_loss = loc_loss + conf_loss
            total_loss.backward()
            optimizer.step()
```

```
            if step % 10 == 0:
                print(f'Epoch [{epoch+1}/{args.epochs}], Step
[{step+1}/{len(data_loader)}], Loss: {total_loss.item()}')

        torch.save(ssd_network.state_dict(), os.path.join(args.output_dir,
f'ssd300_epoch_{epoch+1}.pth'))

if __name__ == '__main__':
    main_training()
```