МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: «**ІНТЕЛЕКТУАЛЬНА СИСТЕМА МОНІТОРИНГУ ЗОВНІШНІХ ОБ'ЄКТІВ ДЛЯ БЕЗПЛОТНИХ ТРАНСПОРТНИХ ЗАСОБІВ В АПК**»

Виконав: здобувач 6 курсу групи Іт-62
Спеціальності 126 «Інформаційні системи та технології»
Задолинний В.І.

Керівник: Чаплига В.М.
Рецензент:

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Освітній ступінь «Магістр» за спеціальністю –
126 – «Інформаційні системи та технології»

“ЗАТВЕРДЖУЮ”
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
“____”_____2024_ р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту
Задолинного Владислава Ігоровича

1. Тема роботи: **«Інтелектуальна система моніторингу зовнішніх об’єктів для безпілотних транспортних засобів в АПК»**.

Керівник роботи Чаплига Вячеслав Михайлович, д.т.н., професор.

Затверджені наказом по університету від «_12_» __09__ 2024 р. № 616/кс

2. Строк подання студентом роботи: 15.12.2024 року.

3. Початкові дані до роботи: Принципи побудови та застосування нейронних мереж третього покоління для моніторингу зовнішніх об’єктів для безпілотних транспортних засобів в АПК.

4. Зміст пояснювальної записки:

Вступ

Розділ 1 Аналіз особливостей моніторингу зовнішніх об’єктів для безпілотних транспортних засобів в.

Розділ 2 Дослідження методів та технологій спайкових нейронних мереж для моніторингу зовнішніх об’єктів безпілотних транспортних засобів в АПК

Розділ 3 Розробка концептуальної моделі спайкової нейронної мережі для моніторингу зовнішніх об’єктів безпілотних транспортних засобів в АПК

Розділ 4 Охорона праці та безпека в надзвичайних ситуаціях

Розділ 5 Розрахунок економічної ефективності Інтелектуальної системи моніторингу зовнішніх об’єктів для безпілотних транспортних засобів в АПК.

Висновки та пропозиції

Список використаних джерел

5. Перелік графічного матеріалу – презентація.

6. Консультанти з розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3, 5	*Чаплига В.М., професор кафедри інформаційних технологій*			
4	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*			

7. Дата видачі завдання «__12___» _____09_____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Написання Вступу, першого розділу та означення головних завдань роботи*	12.09.2024 - 25.09.2024	
2	*Виконання другого розділу та формування початкових даних*	26.09.2024 - 16.10.2024	
3	*Виконання третього розділу та узагальнення отриманих результатів роботи*	17.10.2024 - 01.11.2024	
4.	*Розроблення та обґрунтування пропозицій щодо реалізації результатів роботи. Розроблення питань з охорони праці. Написання економічної частини (4- 5 розділи роботи).*	2.11.2024 – 24.11.2024	
5	*Кінцеве оформлення кваліфікаційної роботи та оформлення ілюстративних матеріалів, таблиць, здача роботи на перевірку на плагіат та на рецензування.*	25.11.2024 – 01.12.2024	
6	*Підготовка до захисту в ЕК (написання доповіді). Пробний захист на кафедрі, виправлення зауважень. Завершення кваліфікаційної роботи в цілому*	2.12.2024 – 05.12.2024	
7	*Перевірка на плагіат*	06.12 2024	

Студент _____ Задолинний В. І.

Керівник роботи _____ Чаплига В. М.

УДК 635.1

Р Е Ф Е Р А Т

Інтелектуальна система моніторингу зовнішніх об’єктів для безпілотних транспортних засобів в АПК.
Задолинний В. І. Кафедра ІТ – Дубляни, Львівський НУП, 2024.
Кваліфікаційна робота: 79 с. текст. част., 31 Рисунок, 5 табл., 15 джерел.

Робота присвячена вирішенню актуальної задачі, пов'язаної з оптимізацією швидкості та точності моделей моніторингу виявлення та розпізнавання об'єктів в реальному часі зовнішніх об’єктів безпілотними транспортними засобами в АПК.

Об'єкт дослідження - процес роботи нейронних мереж в задачах моніторингу та розпізнавання зовнішніх об’єктів для безпілотних технологічних засобів в АПК. Предмет дослідження - методи перетворення нейронних мереж до нейронних мереж для моніторингу та розпізнавання зовнішніх об’єктів безпілотними технологічними засобами в АПК. Мета дослідження - розроблення та реалізація концептуальних моделей інтеграції спайкових нейронних мереж із вже існуючою моделлю YOLOv3 для виявлення та розпізнавання зовнішніх об'єктів у режимі реального часу в системах моніторингу безпілотних транспортних засобів АПК. В роботі удосконаленна концептуальна модель згорткової нейронної мережі, дістало подальшого розвитку нейронна мережа YOLOv3 та досліджено вплив LIF активаторів на характеристики YOLOv3, що на практиці дає можливість застосовувати точні нейронні мережі з використанням менш потужних графічних інтерфейсів для розпізнавання зовнішніх об’єктів для безпілотних транспортних засобів.

Сформовано заходи щодо охорони праці та безпеки в надзвичайних ситуаціях. Визначено параметри ефективності отриманих в роботі рішень.

Ключові слова: нтелектуальна система, моніторинг, зовнішній об’єкт, безпілотний транспортний засіб, АПК.

UDC 635.1

SUMMARY

An intelligent system for monitoring external objects for unmanned vehicles in the agricultural sector.
V. I. Zadolinnyi, Department of IT - Dublyany, Lviv National University of Technology, 2024.
Qualification work: 79 p. text. chast., 31 figures, 5 tables, 15 sources.

The work is devoted to solving a pressing problem related to optimizing the speed and accuracy of monitoring models for real-time detection and recognition of external objects by unmanned vehicles in the agricultural sector. The object of the study is the process of neural networks in the tasks of monitoring and recognizing external objects for unmanned technological means in the agricultural sector. The subject of the study is methods for converting neural networks to neural networks for monitoring and recognizing external objects by unmanned technological means in the agricultural sector. The purpose of the study is to develop and implement conceptual models for integrating spike neural networks with the existing YOLOv3 model for real-time detection and recognition of external objects in monitoring systems for unmanned vehicles in the agricultural sector. The work improved the conceptual model of the convolutional neural network, further developed the YOLOv3 neural network, and investigated the influence of LIF activators on the characteristics of YOLOv3, which in practice makes it possible to use accurate neural networks using less powerful graphical interfaces.

Measures for labor protection and safety in emergency situations were formed. The parameters of the effectiveness of the solutions obtained in the work were determined.

Keywords: intelligent system, monitoring, external object, unmanned vehicle, agricultural industry.

ЗМІСТ

ВСТУП

Актуальність. Інтелектуальні системи моніторингу зовнішніх об’єктів є фундаментальною складовою для впровадження безпілотних транспортних засобів в АПК (див. Рисунок В1 – В2)

Рисунок В.1 - Безпілотний вантажний автомобіль VERA від компанії Volvo Trucks.

Рисунок В.2 - Безпілотний трактор від компанії John Deere.

Вони не лише забезпечують безпечну та ефективну роботу автономних систем, але й сприяють підвищенню продуктивності, екологічної стійкості та економічної ефективності аграрного виробництва. Ці системи формують основу для сталого розвитку "Розумного сільського господарства" та інтеграції інноваційних рішень в агропромисловий сектор.

За останні роки досягнення у галузі виявлення та розпізнавання об'єктів вражають, особливо, завдяки моделі YOLO v3. Однак існують виклики, пов'язані з оптимізацією швидкості та точності цих моделей в реальному часі, особливо у сфері безпілотних транспортних засобів (БТЗ) в АПК.

Інтеграція алгоритму YOLOv3 (You Only Look Once, версія 3) із спайковими нейронними мережами (SNN, Spiking Neural Networks) є перспективним рішенням у задачах моніторингу зовнішніх об’єктів для безпілотних транспортних засобів (БТЗ) в агропромисловому комплексі (АПК). Ця комбінація об’єднує швидкість і точність YOLOv3 із енергоефективністю та адаптивністю SNN, що дозволяє не тільки вирішувати виклики, пов'язані із збереженням точності, але й відкриє можливість для покращення швидкості та зменшення енергоспоживання в системах виявлення і моніторингу зовнішніх об’єктів для БТЗ.

Об'єкт дослідження: процес роботи спайкових нейронних мереж в задачах моніторингу та розпізнавання зовнішніх об’єктів для безпілотних транспортних засобів в АПК.

Предмет дослідження: перетворення згорткових нейронних мереж до спайкових нейронних мереж для моніторингу та розпізнавання зовнішніх об’єктів безпілотними транспортними засобами в АПК.

Мета дослідження: розроблення та реалізація концептуальних моделей інтеграції спайкових нейронних мереж із вже існуючою моделлю YOLOv3 для виявлення та розпізнавання зовнішніх об'єктів у режимі реального часу в системах моніторингу безпілотних транспортних засобів АПК.

Завдання, які поставлені та виконані для досягнення мети:

- проаналізувати особливості характеристикк спайкових нейронних мереж типу LIF (Leaky Integrate-and- Fire), зокрема їхню специфіку у порівнянні із згортковими нейронними мережами;

- підготувати тренувальні дані, розробити та використовувати спеціалізований датасет для автопілоту , враховуючи унікальні особливості цього застосування;
- підготувати концептуальну модель спайкових мереж для моніторингу та розпізнавання зовнішніх об’єктів безпілотними транспортними засобами в АПК.
- створити архітектуру моделі, яка буде ефективно інтегруватися з YOLOv3, забезпечуючи оптимальну точність та швидкість роботи в задачах моніторингу та розпізнавання зовнішніх об’єктів для безпілотних транспортних засобів в АПК;
- провести тестування моделі, здійснивши серію тестів для оцінки продуктивності, точності та стійкості нової моделі в реальних умовах.
- проаналізувати роботу вихідної мережі і порівняти результати нової моделі із YOLOv3, визначити вплив інтеграції спайкових нейронних мереж на точність та швидкість виявлення зовнішніх об’єктів для безпілотних транспортних засобів в АПК.

Методи дослідження. У роботі знайшли своє застосування теорія виявлення та розпізнавання об'єктів, теорія спайкових нейронних мереж типу LIF та роботи їх активаторів, теорія роботи моделі виявлення та розпізнавання об'єктів

Інформаційною основою роботи слугували науково-практичні монографії і статті спеціалістів, а також підручники по темі кваліфікаційної роботи.

Наукова новизна роботи полягає в удосконаленні концептуальної моделі згорткової нейронної мережі, подальшому розвитку нейронної мережі YOLOv3 та дослідженню впливу LIF активаторів на характеристики YOLOv3 в задачах моніторингу та розпізнавання зовнішніх об’єктів для безпілотних транспортних засобів в АПК.

Практичне значення отриманих результатів дає можливість застосовувати точні нейронні мережі з використанням менш потужних графічних

інтерфейсів для розпізнавання зовнішніх об’єктів для безпілотних транспортних засобів.

Апробація результатів роботи. Основні теоретичні та практичні результати кваліфікаційної роботи доповідались на наукових семінарах кафедри ІТ та на Міжнародному студентському науковому форумі, жовтень 2024, Львів.

Публікації здобувача за темою кваліфікаційної роботи.

Задолинний В. І. Система моніторингу зовнішніх об’єктів для безпілотних транспортних засобів в АПК на основі спайкових нейронних мереж. Студентська молодь і науковий прогрес: тези доп. Міжнар. студ. наук. форуму, 02–04 жовт. 2024 р. [Електронний ресурс]. Львів, 2024. С. 376.

Структура та обсяг кваліфікаційної роботи. Робота містить вступ, п’ять розділів, висновки та пропозиції, список використаної літератури та додатки**.**

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ МОНІТОРИНГУ ТА ВИЯВЛЕННЯ ЗОВНІШНІХ ОБ’ЄКТІВ АВТОПІЛОТОМ ТРАНСПОРТНИХ ЗАСОБІВ В АПК

1.1 Аналіз особливостей побудови спайкових нейронних мереж для виявлення зовнішніх об'єктів автопілотом транспортного засобу в АПК

Особливості архітектури SNN полягають у тому, що шар сенсорного сприйняття: інтегрує дані з камер, лідара, ультразвукових сенсорів чи інших датчиків; кодери спайків перетворюють аналогові або цифрові сигнали в серії спайків (наприклад, за допомогою кодерів на основі амплітуди чи частоти сигналу); обчислювальні шари формують та обробляють сигнали, наприклад, використовуючи алгоритми навчання STDP (Spike-Timing-Dependent Plasticity); шар класифікації: аналізує патерни спайків для виявлення зовнішніх об'єктів (наприклад, людей, тварин, техніки чи перешкод).

Серед робіт з даної тематики відзначимо статті Сейджун Кім, Сонсік Парк, Бюнггук На, Сангро Юн [1]. Їх модель теж базувалася на схожій вище наведеній архітектурі, але попри це вони тестували лише на даних з COCO dataset та VOC datasets. В даній роботі використовується більш «заточений» на виявлення і моніторинг зовнішніх об’єктів для БТЗ датасет. Те саме стосується роботи Сейджун Кім, Сонсік Парк, Бюнггук На, Сангро Юн [2]. Вона також використовує функції активатори подібні до даної роботи але вони теж протестовані лише на загальному наборі даних. Модель Бісваділ Чакраборті та Сюеюань Ше [3] показала високи результати, але вони зосереджені лише на роботі з Tiny YOLO а не з Yolo.

Побудова спайкових нейронних мереж (SNN, Spiking Neural Networks) для виявлення зовнішніх об'єктів автопілотом транспортного засобу базується на

використанні принципів активації нейронів лише при досягненні певного порогового рівня. Ці мережі імітують роботу людського мозку та працюють за подієво-орієнтованим підходом, що особливо ефективно для обробки великих потоків сенсорної інформації в реальному часі. Серед основних особливостей побудови SNN для автопілота можна виділити наступні.

На відміну від традиційних нейронних мереж, де обробка здійснюється в кожному такті, SNN працюють на основі подій: нейрони активуються тільки у відповідь на значущі події (спайки). Це дозволяє знизити енергоспоживання, що є критичним для автономних транспортних засобів.

Така подієво-орієнтована обробка ідеально підходить для динамічних сцен, таких як виявлення об'єктів на дорозі, де SNN можуть миттєво реагувати на зміни в навколишньому середовищі.

Оскільки SNN активуються лише при досягненні певного порогу збудження, вони споживають менше енергії порівняно з традиційними глибинними нейронними мережами. Це знижує потребу в ресурсах і збільшує тривалість роботи електронних систем автопілота. Такий підхід дозволяє автопілоту транспортного засобу обробляти дані ефективно в режимі реального часу, що особливо важливо для вантажних електромобілів в АПК.

SNN використовують моделі нейронів, такі як модель Ліфковича (LIF, Leaky Integrate-and-Fire) та більш складна модель Ходжкіна-Хакслі, які імітують збудження та спрацьовування біологічних нейронів. Це дає змогу створювати гнучкіші системи, що пристосовуються до складних змін середовища. Синаптичні зв’язки між нейронами можуть мати параметри ваги, які змінюються на основі принципів пластичності (наприклад, STDP – Spike-Timing-Dependent Plasticity), що дозволяє системі навчатися на основі часових закономірностей у надходженні інформації.

SNN добре підходять для роботи з такими сенсорами, як камера подієвого бачення (event camera), які фіксують тільки зміни в полі зору. Це забезпечує

обробку подій у реальному часі та дозволяє автопілоту миттєво реагувати на зміни об'єктів навколо. Такий підхід підвищує здатність системи виявляти об’єкти навіть при низькій освітленості або в умовах складних погодних умов, оскільки SNN оперують подіями, а не безперервним потоком інформації.

Спайкові нейронні мережі можуть враховувати часові залежності між спайками, що робить їх придатними для ідентифікації об'єктів, які змінюють положення або форму. Це дає змогу автопілоту визначати швидкість і траєкторію руху об'єктів, таких як пішоходи, транспортні засоби чи перешкоди. SNN забезпечують швидку реакцію на раптові зміни (наприклад, різка поява об’єкта на дорозі), що сприяє безпеці під час руху.

Для автопілотів БТЗ в АПК ефективно використовувати гібридний підхід, комбінуючи SNN з традиційними нейронними мережами або методами машинного навчання, такими як згорткові нейронні мережі (CNN). Це дозволяє поєднати енергоефективність SNN із високою точністю CNN при розпізнаванні складних об'єктів. Гібридні моделі допомагають зменшити обчислювальні витрати, дозволяючи автопілоту обробляти різні типи даних від численних сенсорів (камери, лідари, радари) для формування цілісної картини оточення.

Використання принципів пластичності, як STDP, дає змогу SNN навчатися на основі нового досвіду та адаптуватися до раніше невідомих об'єктів або змін у навколишньому середовищі. Завдяки цьому автопілот може «вчитися на льоту», що є цінним при розпізнаванні нових або нестандартних об’єктів, наприклад, нестандартного поведінкового патерну водія на дорозі.

SNN мають високу стійкість до шумів і артефактів, що важливо в умовах непередбачуваного середовища, де зображення можуть мати перешкоди (погодні умови, недостатнє освітлення). У порівнянні з традиційними моделями, які можуть створювати помилкові спрацьовування на основі шумових сигналів, SNN забезпечують більш стабільні результати за рахунок порогових значень для спайків.

Спайкові нейронні мережі забезпечують енергоефективне, стійке до шуму та подієво-орієнтоване розпізнавання об’єктів у складних умовах навколишнього середовища. Використання SNN для автопілотів може значно підвищити точність і швидкість реагування на зовнішні об'єкти, що є критичним для безпечної та надійної роботи автономних транспортних засобів.

Однією з найновіших систем (нейромереж) для виявлення зовніних об’єктів є розроблена Джозефом Редмоном мережа YOLO, яка працює з великою швидкістю за принципом, ілюстрованим на Рисунок 1.1.

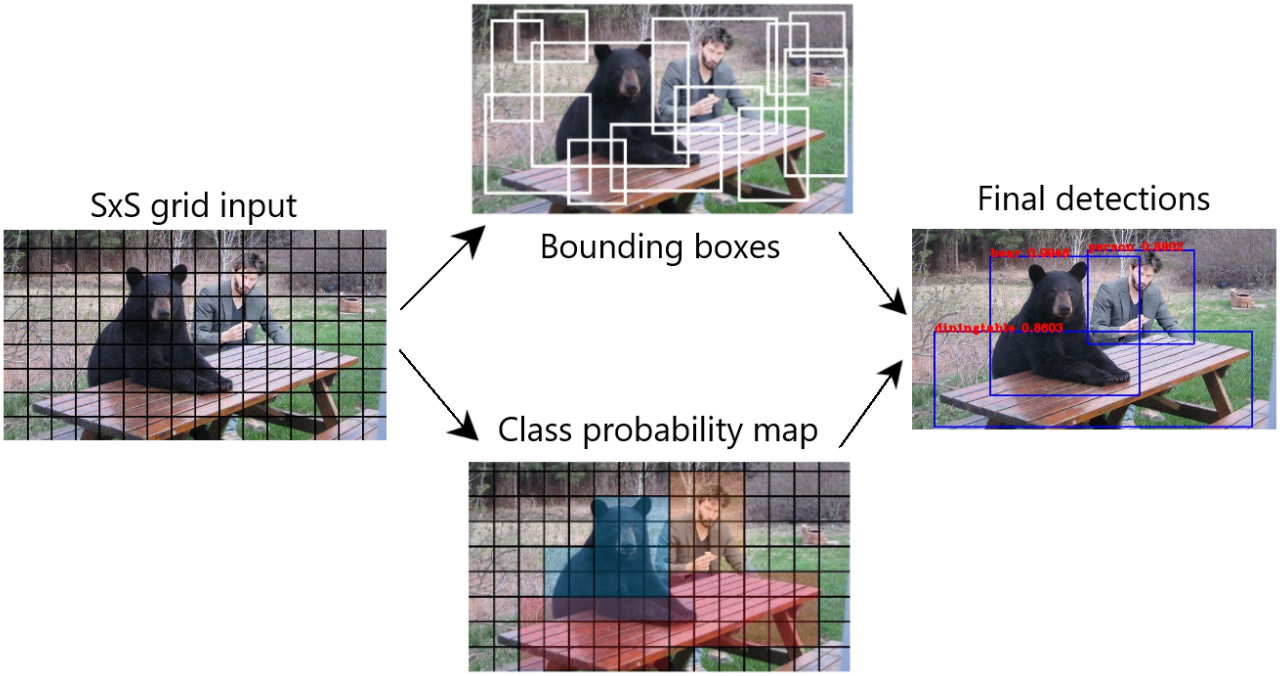


Рисунок 1.1 – Ілюстрація принципу роботи нейромережі YOLOv3

Реалізація покращеної версії YOLOv3 базується на використанні в нейромережі тільки згорточних шарів, яких налічується 53. Тоді створюється функція, до якої надходитимуть важливі параметри, що послідовно змінюються в шарах мережі (див. рисунок 1.2)

Зважаючи на хороші результати виявлення зовнішних об’єктів мережею YOLOv3 вчені практики постійно працюють над її вдосконаленням.

Так, робота [1] спрямована на покращення ефективності у завданнях виявлення об'єктів за допомогою БПЛА шляхом додавання шару SPP в кінці darknet- 53.

У цьому дослідженні було проведено тренування різних моделей: YOLOv3 із шаром SPP, YOLOv3, та YOLOv3-tiny, використовуючи набір даних Visdrone2019- Det. Оцінка ефективності цих моделей здійснювалася за допомогою валідаційного набору при вхідному масштабі зображень 640x640.

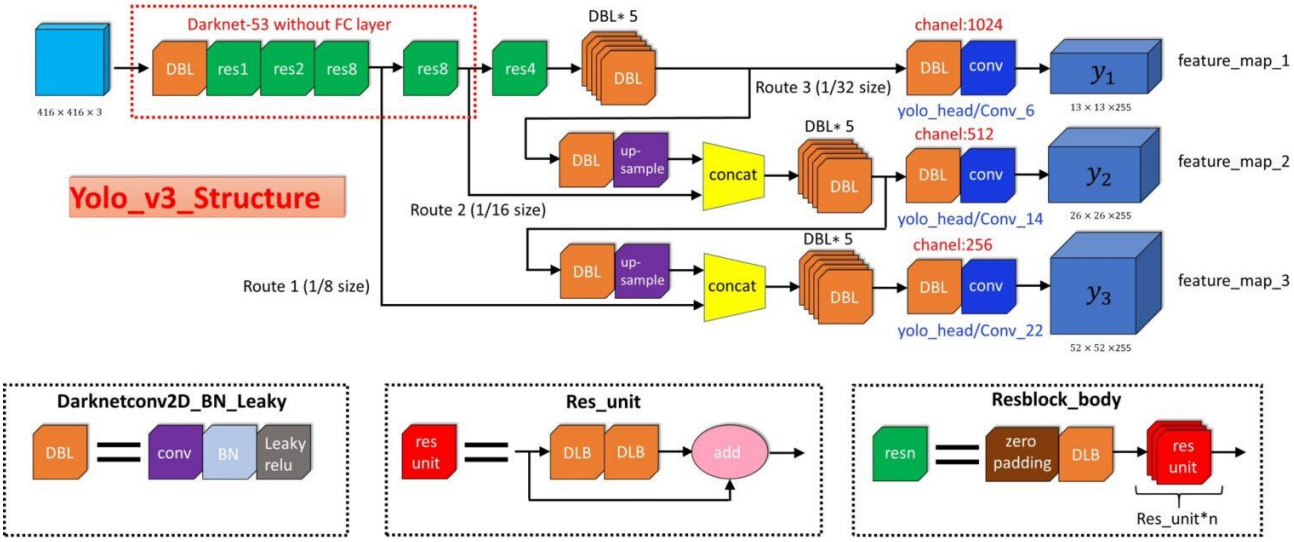


Рисунок 1.2 – Структура нейромережі YOLOv3.

Результати дослідження показали, що YOLOv3 із шаром SPP виявив значне покращення ефективності моделі виявлення об'єктів. Точність mAP (mean Average Precision) цієї моделі виявилася на 0,6% вищою, ніж у YOLOv3, та на 26,6% вищою, ніж у YOLOv3-tiny. Важливо відзначити, що YOLOv3 із шаром SPP здатний зберігати високу точність навіть при різних вхідних масштабах. Наприклад, результати показали, що на вхідному масштабі 960x960 точність mAP цієї моделі була вищою на 0,8%, а на вхідному масштабі 1280x1280 вона виявила покращення на 0,9% у порівнянні із YOLOv3.

У той час як YOLOv3-tiny залишався менш ефективним на обох масштабах порівняно із YOLOv3 із шаром SPP та звичайним YOLOv3. Ці результати

підтверджують, що додавання шарів SPP до YOLOv3 може значно підвищити ефективність моделей виявлення об'єктів, особливо при використанні даних, отриманих від БПЛА, і при різних вхідних масштабах зображення.

Результати YOLOv3 з SPP свідчать, що додавання шарів SPP до YOLOv3 може покращити ефективність моделей виявлення об'єктів з даних, отриманих від БПЛА, навіть при різних вхідних масштабах зображення.

Автори Сейджун Кім, Сонсік Парк, Бюнггук На, Сангро Юн [2] представили новаторську модель штучної нейронної мережа Spiking-YOLO, яка визначається за використанням спайкових нейронів для ефективного виявлення об'єктів на зображеннях.

1.2. Аналіз методик виявлення об'єктів автопілотом транспортного засобу в АПК

Виявлення об'єктів (object detection) полягає в тому, що комп'ютер переглядає зображення чи відео та намагається знайти та з'ясувати, що на ньому є різні речі. Задача полягає як в пошуку та і класифікації об'єктів на зображенні. Це допомагає ідентифікувати задані класи об'єктів для подальшого їх опрацювання.

Виявлення об'єкта є однієї із складових компонентів самостійного керування автомобіля. Її використання описане в стандартизованому документі SAE J3016 для даної системи. Згідно неї рівні автоматизації автопілот автомобіля , ділиться на п'ять категорій.

Перші два рівня (0-2) означають, що автомобілем керує людина, яка активно стежить за дорожньою обстановкою і несе відповідальність за всі прийняті дії. Однак навіть на цих рівнях автомобіль може давати деякі підказки і допомагати, наприклад, у режимі круїз-контролю.

На третьому рівні система може взяти на себе керування при заданих умовах. Це відбувається завдяки використовуючи передові системи допомоги

водієві (ADAS). В даній ситуації водій має бути готовий взяти на себе керування при можливість виникнення небезпечної ситуації.

Четвертий рівень пропонує повністю автоматизацію водіння. Попри це водій має лишатися за кермом для можливості взяти управління транспортним засобом на себе.

П'ятий рівень автопілот, при якому всі аспекти керування автоматизовані. При цьому вже немає необхідності водієві перебувати на водійському місці.

Узагальнена інформація по стандарту SAE J3016 2018 представлена у таблиці 1.1.

Описання даних рівнів допомагає розробникам та користувачам, як автомобілі мають працювати з навколишнім середовищем та водієм і в якому напрямку має рухатися дана сфера.

Враховуючи це виявлення об'єктів повинне виконувати наступні завдання:

- знаходження, визначення положення та відслідковування певних об'єктів довкола машини.
- класифікація знайдених і вибраних об'єктів відповідно до заданих міток.

В самокерування вони можуть виконувати важливі функції. Це є як і пошук транспортних засобів та пішоходів біля машини для знаходження їх в середовищі так і аналіз світлофорів та різних дорожніх знаків.

Таблиця 1.1. Опис рівнів автоматизації водіння транспортним засобом в стандарті SAE J3016

Рівень	Опис
0	Людина керує автомобілем, немає автоматизації.
1	Людина керує автомобілем, але може отримувати деякі підказки та допомогу від системи, наприклад, круїз-контроль.

Продовження таблиці 1.1

2	Людина керує автомобілем, але може використовувати систему в деяких умовах, наприклад, автопілот в певних ситуаціях.
3	Система бере на себе керування автомобілем у визначених умовах (використовуючи системи ADAS), але водій повинен бути готовий взяти управління у небезпечних ситуаціях.
4	Повністю автоматизоване водіння, але водій повинен залишатися за кермом та готовий взяти управління в будь-який момент.
5	Повністю автоматизоване водіння, водій вже не потрібен за кермом, всі аспекти керування автоматизовані.

Завдяки технології об'єднання сенсорів та стрімкому вдосконаленню штучного інтелекту, автономні транспортні засоби починають набувати визнання як реальної можливості майбутнього. Прогнозується, що до 2030 року приблизно 12% реєстрацій транспортних засобів по всьому світу буде припадати саме на автономні.

1.3 Аналіз моделей глибокого навчання конволюційних нейронних мереж

Найпопулярнішим типом нейронної мережі для роботи із відео та зображенням є згорткові (конволюційні - convolutional neural network, CNN) нейронні мережі.

Нейронна мережа визначається як система взаємопов'язаних нейронів. Нейрони, або нервові клітини, є основними будівельними блоками мозку та є

біологічними нейронними мережами. Структура нейрона така, як показано на Рисунок 1.3.

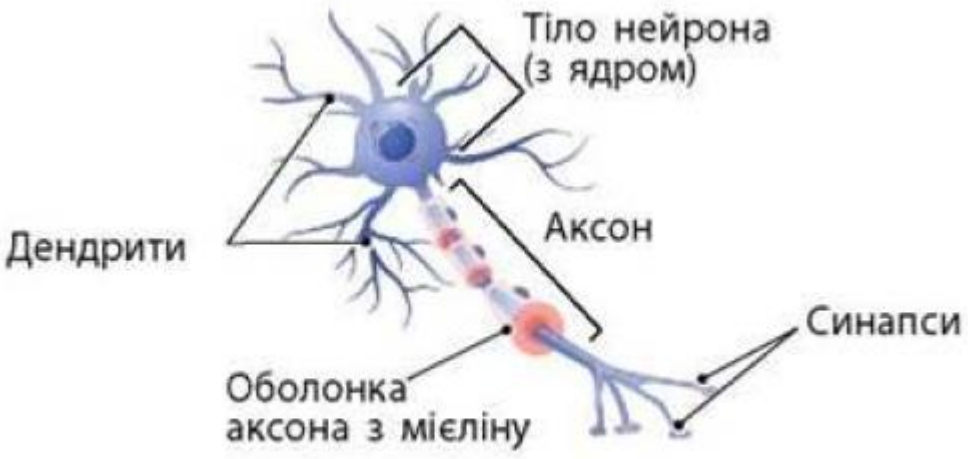


Рис. 1.3 - Схема біологічного нейрона

Обчислювальні системи складаються з багатьох простих і добре взаємопов'язаних елементів обробки, які обробляють зовнішню вхідну інформацію за допомогою динамічних відповідей стану. Нейрони мають здатність виробляти лінійні або нелінійні реакції. Вхід і вихід нелінійної системи не пропорційні, як показано на Рисунок 1.4.

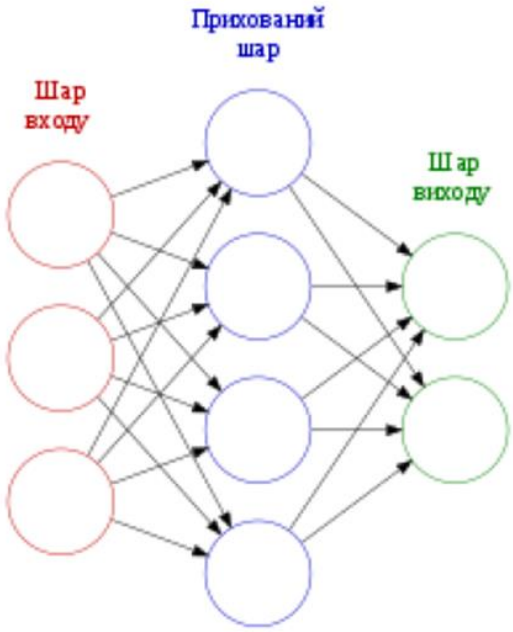


Рисунок 1.4 - Ілюстрація будови нейронних мереж.

Глибокі нейронні мережі мають велику кількість шарів, зазвичай більше трьох. Це означає, що вони мають багато прихованих шарів між вхідними і вихідними шарами. Згорткова нейронна мережа — це архітектура нейронної мережі глибокого навчання, яка зазвичай використовується в комп'ютерному зорі.

CNN зазвичай має три рівні: згортковий рівень, рівень агрегування та повноз'єднаний рівень (див Рисунок 1.5).

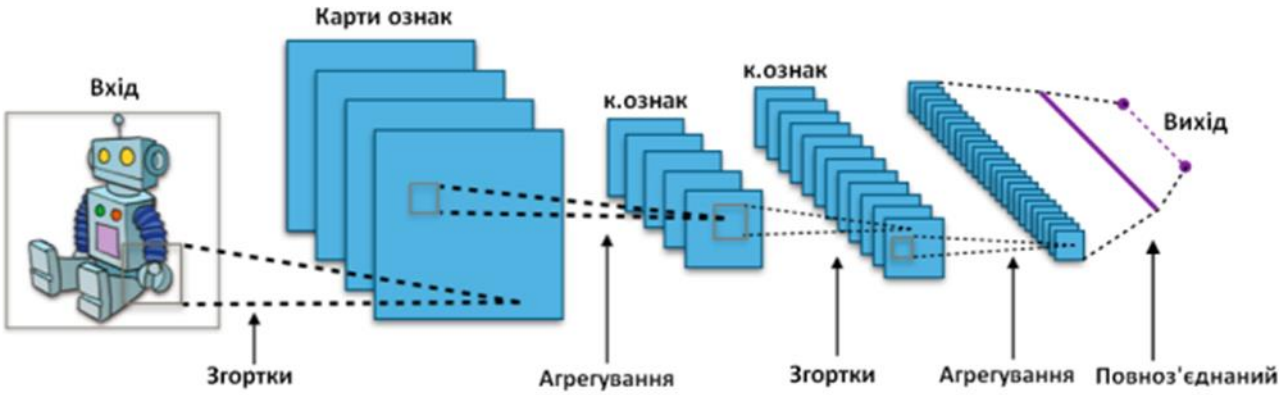


Рисунок 1.5 - Архітектура згорткової (конволюційної - convolutional neural network, CNN) нейронні мережі.

Згорткові шари є основними будівельними блоками CNN. Вони виконують більшу частину обчислювального навантаження мережі.

Цей рівень виконує скалярний добуток між двома матрицями, де одна матриця є набором параметрів, що вивчаються (також називається ядром), а інша матриця є обмеженою частиною рецептивного поля. Ядро просторово менше зображення, але глибше. Це означає, що якщо зображення складається з трьох (RGB) каналів, висота і ширина комірки будуть просторово малими, але глибина буде розподілена по всіх трьох каналах. Цей процес показано на Рисунок 1.6.

Під час прямого проходу комірки ковзає по висоті та ширині зображення, створюючи представлення зображення цієї області прийому. Це створює двовимірне представлення зображення, яке називається картою активації, яка дає реакцію комірка на кожне просторове розташування зображення.

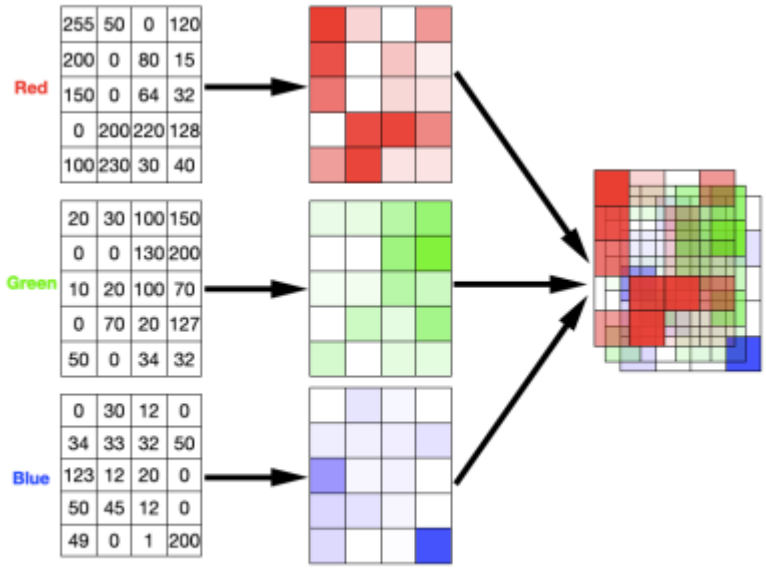


Рисунок 1.6 - Скалярний добуток в CNN

Під час прямого проходу комірки ковзає по висоті та ширині зображення, створюючи представлення зображення цієї області прийому. Це створює двовимірне представлення зображення, яке називається картою активації, яка дає реакцію комірка на кожне просторове розташування зображення. Величина ковзання комірки називається кроком.

Розмір вихідного об'єму можна визначити за допомогою:

$$W_{out} = \frac{W-F+2P}{S} \quad (1.1)$$

де:

W-ширина вхідної карти активації

F- кількість ядр із розміром простору

S- розмір кроку

P – розмір заповнення

Шар агрегації замінює вихід мережі в заданому місці отриманням сумарної статистики для найближчого виходу. Це дає змогу зменшити просторовий розмір

подання і скоротити кількість необхідних обчислень і ваг. Операція об'єднання обробляється окремо для кожного фрагмента подання.

Існує кілька функцій об'єднання, включно з усередненням за прямокутними околицями,

L2-нормою прямокутних околиць і зваженим усередненням на основі відстані до центрального пікселя. Однак найпоширенішим процесом є максимальне об'єднання, яке дає змогу отримати максимальний результат з околиць.

З огляду на карту активації розміром W x W x D, розмір вихідного об'єму можна визначити за такою формулою:

$$W_{out} = \frac{W-F}{S} \qquad (1.2)$$

де:

W- ширина вхідної карти активації

F- ядро об'єднання просторового розміру

S- розмір кроку

У всіх випадках злиття забезпечує певну трансляційну інваріантність, що означає можливість розпізнавання об'єктів, де б вони не знаходилися в кадрі.

Шар FC служить для відображення уявлення між вхідним і вихідним даними.

FC мають повний зв'язок з усіма нейронами попереднього і наступного шарів. Тому ефект зсуву з подальшим перемноженням матриць можна обчислити звичайним чином.

Оскільки згортка є лінійною операцією, а зображення далеко не лінійне, шар нелінійності часто розміщують одразу після шару згортки, щоб додати нелінійність у карту активації.

Існує кілька типів нелінійних операцій, але найчастіше використовують такі сигмоїдальні нелінійності мають математичну форму:

$$\sigma(x) = \frac{1}{(1+e^{-x})} \tag{1.3}$$

При цьому беруться дійсні числа і "звужується" діапазон від 0 до 1. Однак вкрай небажаною властивістю сигмоїда є те, що під час активації на будь-якому хвості градієнт стає практично нульовим. Якщо локальний градієнт стає дуже маленьким, то це фактично "вбиває" градієнт у процесі зворотного поширення. Крім того, якщо дані, що надходять на нейрон, завжди позитивні, то на виході сигмоїди будуть або всі позитивні, або всі негативні дані, що призведе до зигзагоподібної динаміки в оновленні ваг градієнта.

Tanh стискає дійсні числа в діапазон [-1, 1]. Як і в сигмоїдальних нейронів, активація насичується, але, на відміну від сигмоїдальних нейронів, вихідне значення зосереджене на нулі.

Випрямляючі лінійні блоки (ReLU) стали дуже популярними в останні кілька років. Вони обчислюють функцію:

$$f(k) = \max(0, k) \tag{1.4}$$

Іншими словами, активація - це просто поріг, що дорівнює нулю.

Таким чином, мережева штучних SNN значно спрощена порівняно з реальними біологічними мережами. У зв'язку з цим доцільно припустити, що модельовані спайкові нейрони мають суто порогову динаміку (на відміну від рефрактерної, гістерезисної та резонансної динаміки).

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ПІДХОДІВ ДО РОЗРОБЛЕННЯ КОНЦЕПТУАЛЬНОЇ МОДЕЛІ SNN ДЛЯ ВИЯВЛЕННЯ ЗОВНІШНИХ ОБ'ЄКТІВ

2.1. Дослідження архітектури моделі YOLOv3

YOLOv3 — це алгоритм виявлення об'єктів, який використовує одну нейронну мережу для одночасного прогнозування обмежувальних прямокутників разом із класовими ймовірностями для кожного об'єкта на зображенні. Однією з основних характеристик YOLOv3 є його здатність ефективно виявляти об'єкти в реальному часі, забезпечуючи високу швидкість і точність.

YOLOv3 використовує архітектуру Darknet-53 як основу для виявлення об'єктів. Darknet-53 складається з 53 конволюційних шарів. Вони відзначається високою швидкістю разом з ефективністю порівняно з попередніми версіями. Дана архітектура стала основним елементом успіху YOLOv3, оскільки вона забезпечує високу продуктивність і точність виявлення об'єктів.

Тривалість подій на трьох різних рівнях виявлення в YOLOv3 розкривається через використання трьох гілок від моделі darknet-53. Шлях до отримання вихідного шару y1 включає додавання кількох шарів згорткової обробки після останнього шару darknet-53. Цей шар має розмір, що в 32 рази менше, ніж фактичне вхідне зображення. Подальший процес включає збільшення шару у 2 рази та конкатенацію з другим вихідним шаром darknet-53, а результат - вихідний шар y2. Такий самий підхід використовується для отримання вихідного шару y3.

Вибір трьохрівневого виявлення виникає з потреби YOLOv3 у точному визначенні об'єктів різних розмірів. Шар y1 дозволяє зменшити вихідний розмір на 32 рази порівняно з фактичним зображенням, y2 - в 16 разів, а y3 - в 8 разів. Таким чином, великі, середні та малі об'єкти ефективно виявляються на відповідних шарах y1, y2 і y3. Цей технічний підхід до трьохрівневого виявлення

сприяє покращенню ефективності YOLOv3 в порівнянні з попередніми варіантами, надаючи системі можливість виявлення об'єктів різних масштабів з високою точністю.

YOLO оперує наступними поняттями.

• Objectness Score - показник служить як бал оцінки впевненості в тому, чи містить даний блок центр якого-небудь об'єкта на фактичному зображенні і цей бал впевненості не залежить від конкретного об'єкта.

• Class Probabilities визначають ймовірність того, що виявлений об'єкт належить певному класу.

• Anchor Box передбачає зсуви до заздалегідь визначених стандартних обрамлень, які називають як anchor boxes. YOLOv3 використовує різні anchors на різних масштабах. Модель YOLOv3 передбачає обрамлення на трьох масштабах, і в кожному масштабі призначає три anchors. Отже, у цілому для цієї мережі використовується дев'ять anchor boxes. Ці anchors визначаються за допомогою алгоритму кластеризації K-means на наборі даних.

• Decode Processing - коли вихідні шари YOLOv3 передбачають атрибути, такі як координати обрамлення, розміри, бал впевненості та ймовірності класу.

• Центральні Координати – коли передаються значення tx, ty у функцію сигмоїди, яка перетворює значення в діапазон між 0 і 1. Потім додаються координати верхнього лівого кута *cx, cy* для передбачення фактичних координат нашого обрамлення.

• Розмір Обрамлення передбачаються за допомогою логарифмічного просторового перетворення виводу, а потім множаться на розмір anchor box. Завдяки концепції «Дивишся тільки один раз -You Only Look Once» (YOLO), YOLOv3 є вдосконаленою моделлю виявлення об'єктів, що використовує сучасні підходи для ефективного виявлення об'єктів на зображеннях. Однією з ключових особливостей архітектури YOLOv3 є його робота з сіткою та "anchor boxes".

Приклад фотографії, як бачить її YOLO показано на рисунку 2.1

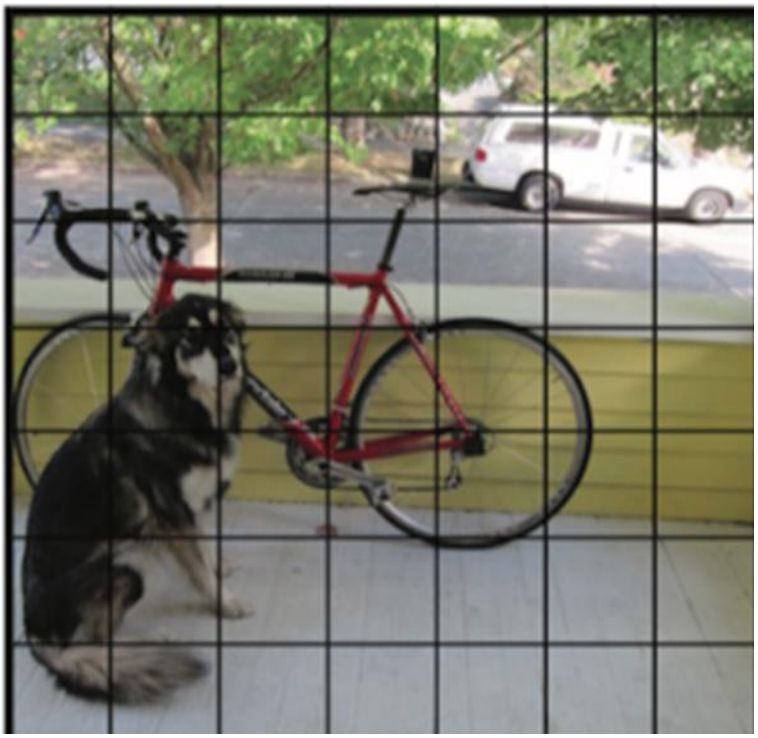


Рисунок 2.1 - Приклад фотографії, як бачить її YOLOv3

YOLOv3 реалізує множинне виведення на різних рівнях сітки, що дозволяє моделі ефективно виявляти об'єкти різних розмірів. Кожен рівень має свої власні anchor boxes та відповідає за виявлення об'єктів конкретного масштабу.

Якщо порівняти YOLov3 з попередньою YOLOv2 та наступною моделлю YOLOv4 можна зробит висновки що:

• YOLOv3 включає додаткові масштаби для виявлення об'єктів різних розмірів та має більшу точність виявлення об'єктів порівняно з YOLOv2.

• YOLOv4 має покращену швидкість та точність в порівнянні з YOLOv3 та використовує CSPDarknet53, подальший розвиток Darknet-53. Крім того вона містить додаткові оптимізації, такі як PANet, для підвищення ефективності виявлення об'єктів на зображеннях.

Треба зазаначити, що у липні 2022 році була випущена новіша версія YOLOv7, яка заснована на обмежувальних прямокутниках та імовірностях

віднесення зовнішнього об’єкту до певного класу, що дозволяє виявляти зовнішні об’єкти в реальному часі і тому вона стає галузевим стандартом в цій області (див. Рисунок 2.2).

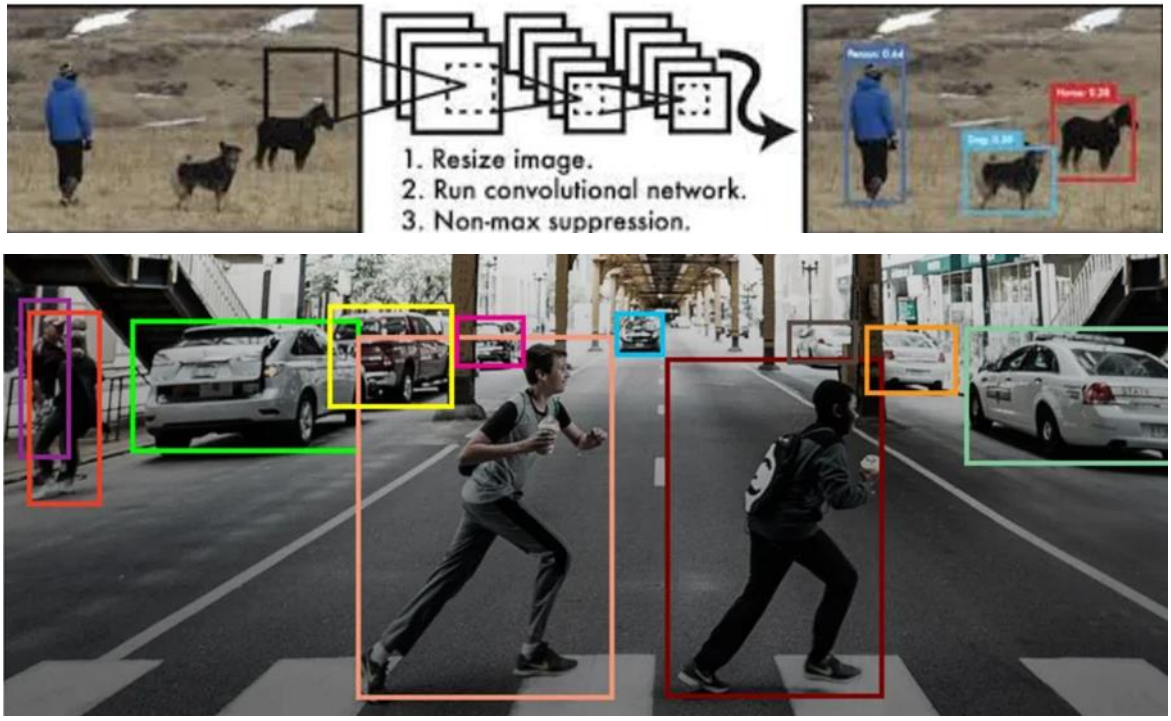


Рисунок 2.2 – Прицип роботи нейромережі реального часу YOLOv7.

А у 2024 році компанія Ultralytics розробила версію YOLOv8.

2.2. Дослідження функції втрат в моделі YOLOv3

Використання функції втрат IoU YOLOv3 для оптимізації має недолік. Вона працює лише тоді, коли огороджувальні рамки мають перекриття, і будь-які рухливі градієнти для випадків без перекриття не надаються. Функція втрат DistanceIoU (DIoU) додає штрафний елемент на основі функції втрат IoU з метою мінімізації відстані між центральними точками двох огороджувальних рамок.

Однак огороджувальні рамки мають три елементи: відношення перекриття, відстань між центральними точками та відношення сторін. Функція втрат CIoU ґрунтується на функції DIoU. Потім до функції втрат CIoU додається умова щодо

сталості відношення сторін для вирішення цієї проблеми.

Діагональну відстань мінімальної замкненої області, яка включає як прогнозну, так і фактичну рамку

$$R_{CIoU} = \frac{p^2(b,b^{gt})}{c^2} + \alpha v \quad (2.1)$$

де b та b^{gt} позначають центральні точки B та B^{gt}

$\rho(\cdot)$ - евклідова відстань

c - діагональна довжина найменшої огороджувальної рамки, що охоплює дві рамки

d - це відстань між o та o’ , при цьому :

o- центр прогнозної рамки

o’ - центр фактичної рамки.

α - це позитивний параметр компромісу, а v вимірює сталість відношення сторін. α визначається як

$$\alpha = \frac{v}{(1-IoU+v)} \quad (2.2)$$

v визначається як

$$v = \frac{4}{\pi^2} * \left(\arctan\frac{\omega^{gt}}{h^{gt}} - \arctan\frac{\omega}{h}\right)^2 \quad (2.3)$$

де:

$\frac{\omega^{gt}}{h^{gt}}$ представляє відношення сторін прогнозної рамки, $\frac{\omega}{h}$ - відношення сторін реальної рамки. Функція втрат визначається як:

$$\text{LossCIou} = 1 - \text{CIoU} = 1 - IoU + \frac{p^2(b,b^{gt})}{c^2} + av \quad (2.4)$$

Також необхідно вказати $\frac{\partial v}{\partial w}$ та $\frac{\partial v}{\partial h}$

$$\frac{\partial v}{\partial w} = \frac{\frac{8}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)h}{w^2 + h^2} \quad (2.5)$$

де w2 + h2- це невелике значення для випадків, коли h та w знаходяться в [0, 1], щоб уникнути вибуху градієнту. Порівняно із функцією втрат IoU, функція втрат CIoU має не тільки той самий показник перекриття, що й IoU, але також має відстань між центральними точками та відношення сторін, яких немає в IoU.

Щодо Моделі Ходжкіна-Хакслі, то вона біологічно точна (обмежена тим, що описує тільки канали та іонні струми всередині нейронів під час генерації спайків), але вимагає великих обчислювальних ресурсів і не реалізовується у великомасштабному моделюванні.

Модель "витік-інтегральне збудження", у якій вхідний струм інтегрується за часом до досягнення порогового мембранного потенціалу без урахування поведінки біологічних іонних каналів, називається інтегральною моделлю збудження (IF). LIF відображає дифузію іонів через мембрану, коли в клітині не досягається певна рівновага, і вводить термін "витік" у модель IF. Завдяки своїй простоті та низькій обчислювальній вартості модель LIF та її варіанти є одними з найбільш широко вживаною із моделей нейронів.

2.3. Моделювання спайкової мережі за допомогою SnnTroch

Усі клітини, включаючи нейрони, оточені тонкою мембраною. Ця мембрана представляє собою ліпідний біляр, який відокремлює провідний солений розчин всередині нейрона від зовнішнього середовища. З електричної точки зору обидва провідники, які розділені ізолятором, функціонують як конденсатор.

Одна з ключових функцій цієї мембрани - контроль над тим, що входить і виходить з клітини (наприклад, іони, такі як Na). Зазвичай мембрана є непроникною для іонів, що блокує їх переміщення в та з нейрона. Але в мембрані існують специфічні канали, які відкриваються при введенні струму в нейрон. Цей рух заряду електрично моделюється за допомогою резистора.

Тепер представимо собі, що до нейрона вводиться який-небудь часовий струм $I_{in}(t)$, чи то через електричну стимуляцію, чи від інших нейронів. Загальний струм в ланцюзі зберігається, тож:

$$I_{in}(t) = I_R + I_C \tag{2.7}$$

Законом Ома потенціал мембрани, вимірюваний між внутрішнім та зовнішнім середовищем нейрона, пропорційний струму через резистор:

$$I_R(t) = \frac{U_{mem}(t)}{R} \tag{2.8}$$

Де C - конденсатор, константа пропорційності між збереженою на конденсаторі зарядом Q та $U_{mem}(t)$:

$$Q = CU_{mem}(t) \tag{2.9}$$

Швидкість зміни заряду визначає ємнісний струм:

$$\frac{dQ}{dt} = I_C(t) = C\frac{dU_{mem}(t)}{dt} \tag{2.10}$$

Отже:

$$I_{in}(t) = \frac{U_{mem}(t)}{R} + C\frac{dU_{mem}(t)}{dt} \Rightarrow RC\frac{dU_{mem(t)}}{dt} = -U_{mem}(t) + rRI_{in}(t) \tag{2.11}$$

Права сторона рівняння має одиниці Напруга. Зліва термін $\frac{dU_{mem(t)}}{dt}$ має одиниці *RC* Напруга/Час. Щоб вирівняти його з лівою стороною (тобто напругою),

$$\tau\frac{dU_{mem(t)}}{dt} = -U_{mem}(t) + rRI_{in}(t) \tag{2.12}$$

має бути одиницею [Час]. Ми називаємо τ=RC часовою константою ланцюга:

Отже, пасивну мембрану описує лінійне диференційне рівняння.

Щоб похідна функції мала такий самий вигляд, як і вихідна функція, тобто

$$\frac{dU_{mem(t)}}{dt} \infty U_{mem}(t) \tag{2.13}$$

це підтверджує, що розв'язок є експоненційним із часовою константою.

Скажімо, що нейрон починає з якогось значення U_0 без додаткового введення, тобто $I_{in}(t) = 0$

Розв’язок лінійного диференціального рівняння матиме наступний вигляд

$$U_{mem}(t) = U_0 e^{-\frac{t}{\tau}}$$

Можна відзначити, що YOLOv3, як алгоритм виявлення об'єктів на зображеннях, виявився ефективним та потужним у реальних системах. Його здатність працювати в реальному часі, використання Darknet-53 архітектури для високої продуктивності, трьохрівневий підхід для точного виявлення об'єктів різних розмірів, а також використання anchor boxes для передбачення розмірів та положення об'єктів роблять його достатньо хорошим рішенням рішенням.

У контексті вдосконалення функції втрат важливим є впровадження нових підходів, таких як DistanceIoU (DIoU) та Compound IoU (CIoU). DIoU вирішує проблему безперекриття огороджувальних рамок, додаючи врахування відстані між їх центральними точками. CIoU враховує відношення сторін та вводить діагональну відстань мінімальної замкненої області, що поліпшує точність моделі в умовах безперекриття.

Загалом, YOLOv3 і його вдосконалення функції втрат відображають постійний розвиток у сфері виявлення об'єктів, роблячи його ідеальним вибором для завдань, де важливість швидкості та точності є вирішальною. Важливо відзначити, що розглянуті нейромодельні підходи, такі як модель Ходжкіна-Хакслі та модель «витік-інтегральне збудження», разом із моделями синаптичної взаємодії, надають повне уявлення про функціонування нейронних систем.

РОЗДІЛ 3
ІНТЕЛЕКТУАЛЬНА СИСТЕМА МОНІТОРИНГУ ЗОВНІШНІХ ОБ'ЄКТІВ ДЛЯ БЕЗПІЛОТНИХ ТРАНСПОРТНИХ ЗАСОБІВ В АПК

3.1. Реалізація моделі YOLO з використанням фреймворку PyTorch

Як було заначено, в даній роботі використовується модель YOLOv3 з архітектурою Darknet-54, яка використовує 54 шари і блоки Convolution (згортковий) та Residual (залишковий). (див. Рисунок 3.1).

Type	Filters	Size	Output
Convolutional	32	3x3	256 x 256
Convolutional	64	3x3/2	128 x128
Convolutional	32	1 x 1	
Convolutional	64	3x3	
Residual			128 x128
Convolutional	128	3x3/2	64x64
Convolutional	64	1 x 1	
Convolutional	128	3x3	
Residual			64x64
Convolutional	256	3x3/2	32x32
Convolutional	128	1 x 1	
Convolutional	256	3x3	
Residual			32x32
Convolutional	512	3x3/2	16x 16
Convolutional	256	1 x 1	
Convolutional	512	3x3	
Residual			16x 16
Convolutional	1024	3x3/2	8x8
Convolutional	512	1 x 1	
Convolutional	1024	3x3	
Residual			8x8
Avgpool		Global	
Connected		1000	
Softmax			

Рисунок 3.1 - Архітектура Darknet-54 (з кількістю шарів 54 і блоками)

Модель з PyTorch замість використання softmax використовує LeakyRelu. Граф спайкової моделі, побудованої з tensorboardX представлена на рисунку 3.2.

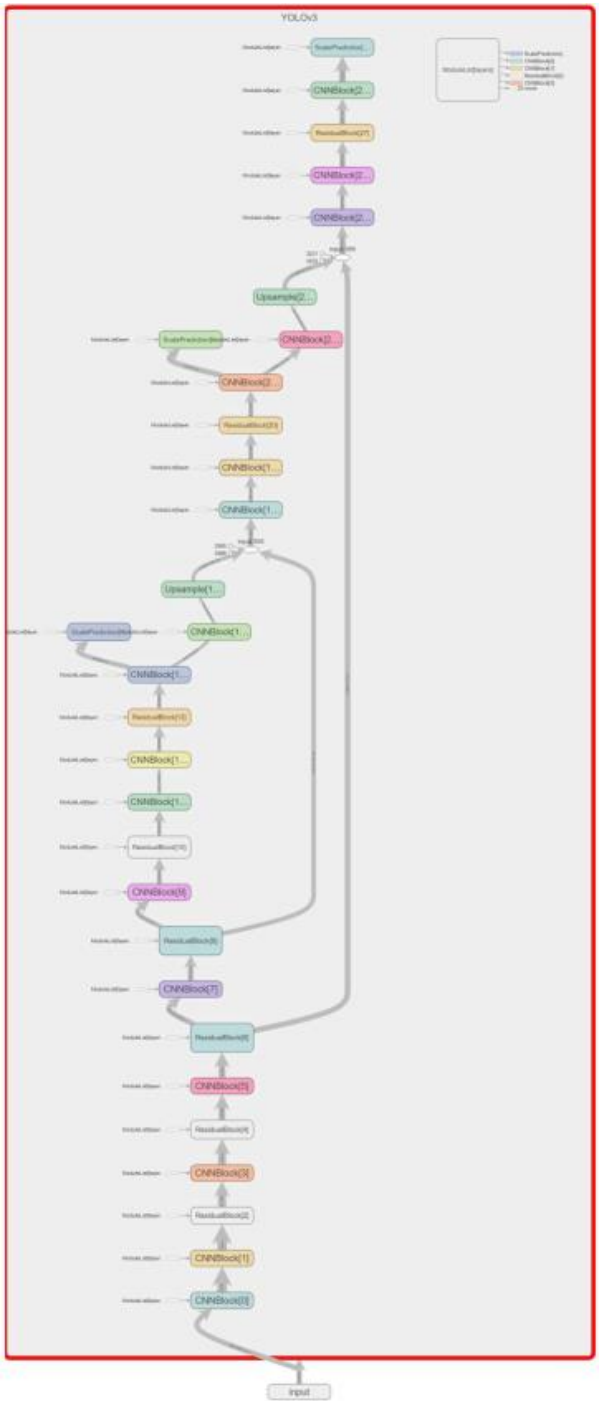


Рисунок 3.2 - Граф спайкової моделі, побудованої за допомогою tensorboardX.

Блок Convolutian (згортковий) блок виконує ключові функції у YOLOv3, проводячи операції з обробки зображення та виділяючи основну інформацію для подальшого аналізу та обробки. Його будова в PyTorch складається з загорткового шару, пакетної нормалізації та активатора. Перший виконує операцію з шарами.

Після опрацювання отримані дані нормалізуються. В кінці дані перетворюються в нейрон за допомогою активатора. Блок має можливість не використовувати нормалізаційний та активаційний шари. В PyTorch згортковий шар та пакетна нормалізація виконується за допомогою nn.Conv2d та nn.BatchNorm2d, відповідно. (див. рисунку 3.3.)

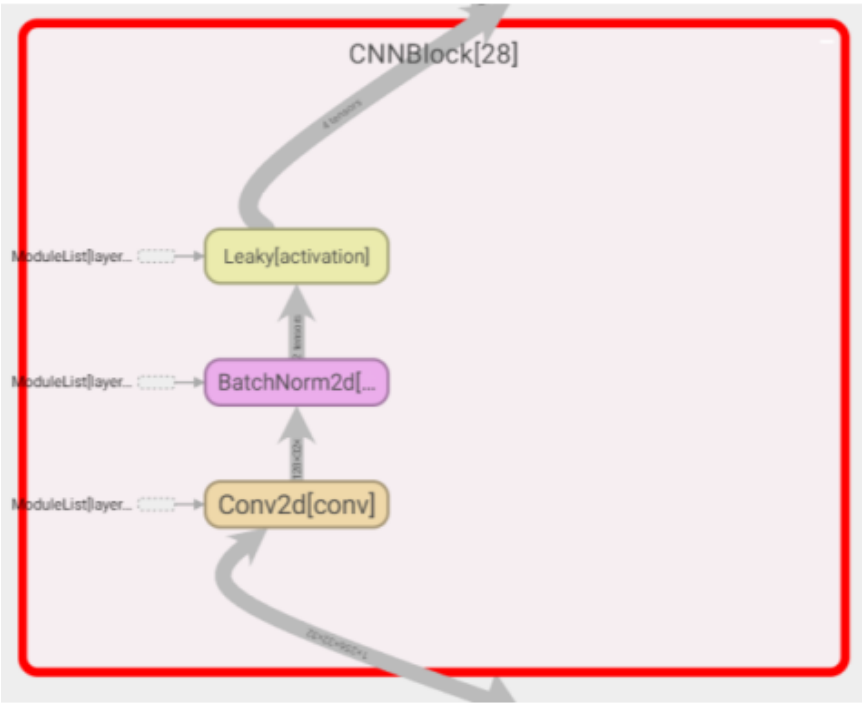


Рисунок 3.3 - Графічне представлення згорткового CNN блоку.

Залишковий блок - виконує функцію покращення навчання. Це виконується завдяки пошуку різниці між вхідними та вихідними сигналами (даними). Ця різниця, або залишок, передається до вихідних сигналів для запобігання проблеми градієнту, де нові дані стають менш важливіші за попередні. Він містить задану (1, 2, 4 або 8 разів) повторюваність пар згорткового, нормалізованого та активаційного шарів (див. рисунок 3.4.).

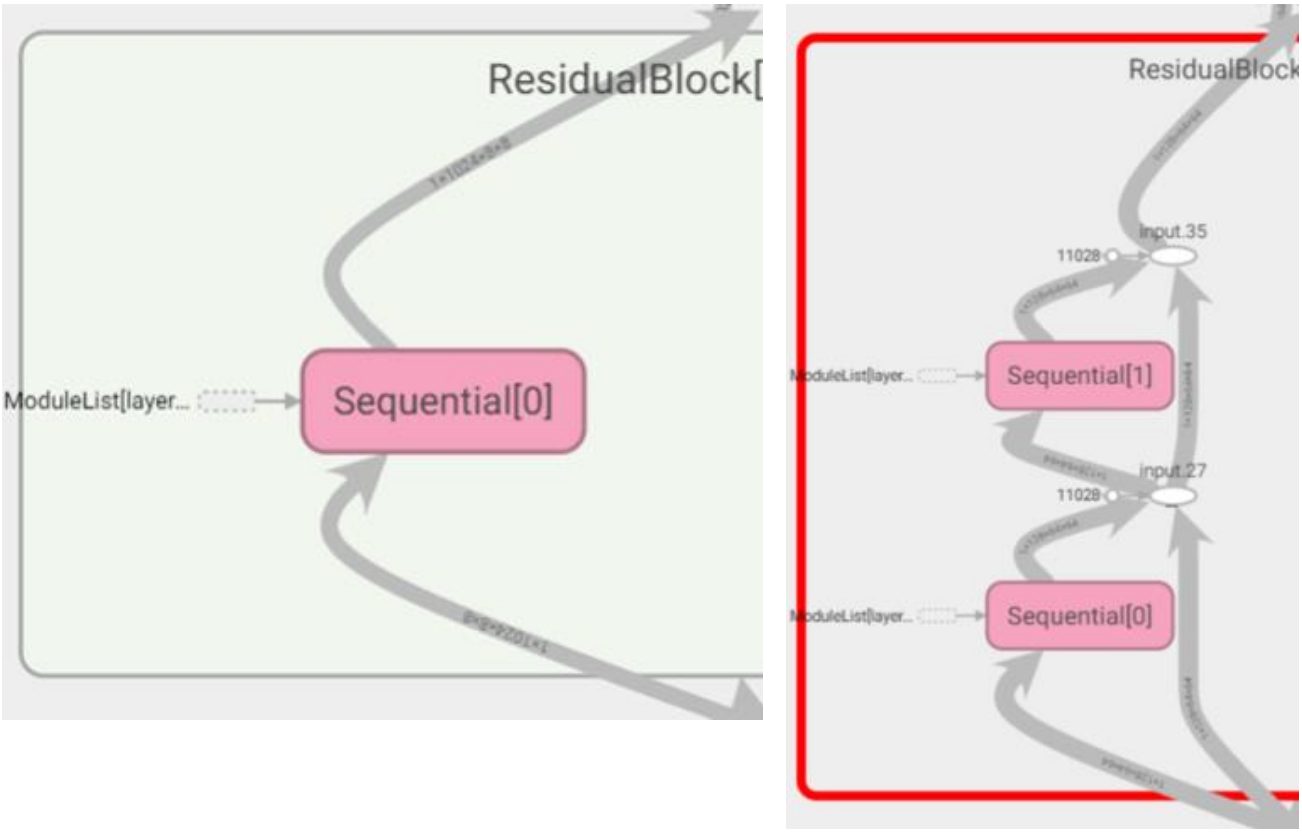


Рисунок 3.4 - Представлення залишкового блоку (Residual) без повторення та з повторенням.

Блок scaleprediction займається створенням вихідних сигналів. Він викликається три рази для трьох різних розмірів боксів. Для цього використовуються ті самі шари що для згорткового блоку. Після цього відбувається переформування та перестановка в вихідній матриці шарів відповідно до YOLO формату.

Розроблений програмний код для цих блоків представлено на рисунках 3.5 – 3.6.

```
class CNNBlock(nn.Module):
    def __init__(self, in_channels, out_channels, use_batch_norm=True, **kwargs):
        super().__init__()
        self.conv = nn.Conv2d(in_channels, out_channels, bias=not use_batch_norm,
**kwargs)
        self.bn = nn.BatchNorm2d(out_channels)
        self.activation =snn.Leaky(beta=0.7,init_hidden=True)
        self.use_batch_norm = use_batch_norm
    def forward(self, x):
        x = self.conv(x)
        if self.use_batch_norm:
            x = self.bn(x)
            x = self.activation(x)
            return x
        else:
            return x
```

Рисунок 3.5 – Програмний код блоку Convolutian для спайкової мережі

```
class ResidualBlock(nn.Module):
    def __init__(self, channels, use_residual=True, num_repeats=1):
        super().__init__()
        res_layers = []
        for _ in range(num_repeats):
            res_layers += [
                nn.Sequential(
                    nn.Conv2d(channels, channels // 2, kernel_size=1),
                    nn.BatchNorm2d(channels // 2),
                    snn.Leaky(beta=0.7,init_hidden=True),
                    nn.Conv2d(channels // 2, channels, kernel_size=3, padding=1),
                    nn.BatchNorm2d(channels),
                    snn.Leaky(beta=0.7,init_hidden=True) )
```

```
            ]
        self.layers = nn.ModuleList(res_layers)
        self.use_residual = use_residual
        self.num_repeats = num_repeats
    def forward(self, x):
        utils.reset(self.layers)
        for layer in self.layers:
            residual = x
            x = layer(x)
            if self.use_residual:
                x = x + residual
        return x
```

Рисунок 3.6 - Програмний код блоку Residual для спайкової мережі

Окрім цих двох блоків є ще блок ScalePrediction, який займається створенням вихідних сигналів. Він викликається три рази для трьох різних розмірів боксів. Для цього використовуються ті самі шари що для згорткового блоку. Після цього відбувається переформування та перестановка в вихідної матриці шарів відповідно до YOLO формату. Програмний код блоку ScalePrediction представлено на Рисунок 3.7.

```
class ScalePrediction(nn.Module):
    def __init__(self, in_channels, num_classes):
        super().__init__()
        lif1=snn.Leaky(beta=0.7,init_hidden=True)
        self.pred = nn.Sequential(
            nn.Conv2d(in_channels, 2*in_channels, kernel_size=3, padding=1),
            nn.BatchNorm2d(2*in_channels),
            lif1,
        nn.Conv2d(2*in_channels, (num_classes + 5) * 3, kernel_size=1),)
```

```
        self.num_classes = num_classes
    def forward(self, x):
        utils.reset(self.pred)
        output = self.pred(x)
        output = output.view(x.size(0), 3, self.num_classes + 5, x.size(2), x.size(3))
        output = output.permute(0, 1, 3, 4, 2)
        return output
```

Рисунок 3.7. Програмний код блоку ScalePredict для спайкової мережі.

Програмний код основного блоку розробленої моделі спайкової мережі представлено в Додатку 1.

3.2. Реалізація додаткових функцій моделі та робота з даними

В даній роботі використовано додаткові функції: iou, nms або алгоритм видалення немаксимальних значень та convert_cells_to_bboxes, ancher box.

Функція **Iou** обчислює значення перекриття площини двох рамок. Це може бути як дві реальні рамки так і реальна із передбаченою. Для знаходження коефіцієнта визначається площина перетину та площу об'єднання. Після чого коефіцієнт визначається як відношенням площин.

Ця функція iou (Intersection over Union) визначає міру подібності між двома прямокутниками, які представлені у форматі [x, y, ширина, висота]. ЇЇ робота показано на Рисунок 3.8.

Функція має дві гілки - одна для випадку прогнозу (is_pred=True), а інша для випадку, коли потрібно порівняти прямокутники за їхніми ширинами та висотами (is_pred=False).

У випадку прогнозу (is_pred=True), функція обчислює координати чотирьох кутів рамки (прогнозу та мітки), визначає координати області перетину та рахує площу об'єднання та перетину. На основі цього обчислюється метрика IoU, яка

вказує на ступінь перекриття прямокутників.

Рисунок 3.8 - Взірець роботи фунції iou.

У випадку, коли is_pred=False, функція розраховує IoU на основі ширини та висоти прямокутників без розрахунку координат. Це може бути корисним, наприклад, при порівнянні двох областей за їхніми габаритами. Ця функція корисна в задачах комп'ютерного зору та виявлення об'єктів для вимірювання точності та подібності між прогнозами та реальними мітками.

Функція **NMS** реалізує алгоритм не-максимального придушення основі їхньої впевненості та ступеня перекриття (IoU). Основні етапи виконання функції наступні.

• Фільтрація за порогом впевненості, коли області, у яких впевненість нижче порогового значення (threshold), відфільтровуються.

• Сортування за впевненістю -залишені області сортуються за впевненістю в порядку спадання.

• Ініціалізація списку після пригнічення найбільшого значення, коли

створюється порожній список, який буде заповнюватися областями після пригнічення найбільшого значення.

• У циклі пригнічення найбільшого значення доки є області відфільтрованого та відсортованого списку, виконуються наступні операції.

1. Вилучається перша область (найвпевненіша) зі списку.

2. Ця область порівнюється з усіма іншими областями, щоб визначити їхнє перекриття (IoU).

3. Якщо перекриття менше заданого порогу (iou_threshold) або мітки областей відрізняються, то додається ця область до списку областей після пригнічення найбільшого значення. Далі повертається список областей після пригнічення найбільшого значення.

Приклад використання даного алгоритму показано на рисунку 3.9.

Рис.3.9 - Реалізація алгоритму не-максимального придушення

Ця функція корисна в задачах обробки великої кількості областей та відбору лише найвпевненіших із них за допомогою алгоритму пригнічення найбільшого значення. Вона використовується для:

- видалення області довіри або рамки якщо їх значення довіри є мешим

ніж вказане значення threshold.

- сортування за довірою, коли області сортуються в порядку спадання розрахованої довіри.

- не-максимальне придушення провидить ітерацію по областям довіри, під час якої викликається функція iou, яка використовується для аналізу накладання рамок.

Функція convert_cells_to_bboxes - конвертує передбачені моделлю дані в координати рамки які можна використовувати при побудові рамок на зображення.

Основні операції функції наступні.

• Отримання розмірності, коли визначається розмірність прогнозів (кількість клітин у ширину і висоту).

• Виділення та обробка прогнозів щодо координат, впевненості та класів з загального тензору прогнозів:

1. Якщо вхід - це прогнози, то обробляються координати x, y та ширини/висоти, застосовуючи сигмоїду для x та y і експоненту для ширини/висоти.

2. Обчислюються індекси клітин.

3. Розраховуються координати x, y та ширини/висоти з відповідним масштабуванням.

• Поетапне формування результату.

1. Перетворення результатів у зручний формат - клас, впевненість, координати x, y, ширини та висоти.

2. Формування кінцевого тензору областей і його подальша конвертація в список.

Програмна реалзіція цих функцій представлена на Рисунок 3.10 та в Додатку 2.

```
    Функція nms
def nms(bboxes, iou_threshold, threshold):
    bboxes = [box for box in bboxes if box[1] > threshold]
    bboxes = sorted(bboxes, key=lambda x: x[1], reverse=True)
    bboxes_nms = []
    while bboxes:
        first_box = bboxes.pop(0)
        for box in bboxes:
            if box[0] != first_box[0] or iou(
                torch.tensor(first_box[2:]),
                torch.tensor(box[2:]),
            ) < iou_threshold:
                if box not in bboxes_nms:
                    bboxes_nms.append(box)
    return bboxes_nms
```

Рисунок 3.10 - Програмна реалзіція функції Nms.

Anchor box - це заздалегідь визначені рамки для боксів різних розмірів, які використовуються в навчанні. Вони використовуються під час навчання і допомагають краще запам'ятовувати об'єкти різних розмірів, та оптимізують процес навчання. Вона має вигляд матриці (3x3x2) і вираховується за допомогою кластеризації. При спробі вирахувати дані параметри для власного датасету, Anchor box показав числа дуже малі, які не можна використовувати. Тому використовувалися рамки наведені у оригінальній статті, оскільки вони показали хороші результати особливо для PyTorch .

База даних використовується BDD100K, у якій містяться відео та зображення для виявлення об'єктів та їх сегментації. Матеріали зроблені з передньої камери автомобіля тому вони підходять для створення безпілотного

транспортного засобу в АПК. Ці дані містять зображення при різних погодних умовах та в різних проміжки часу, а саме, вдень і вночі, що максимально близько до реальних умов. Їх приклад показано на рисунку 3.11.

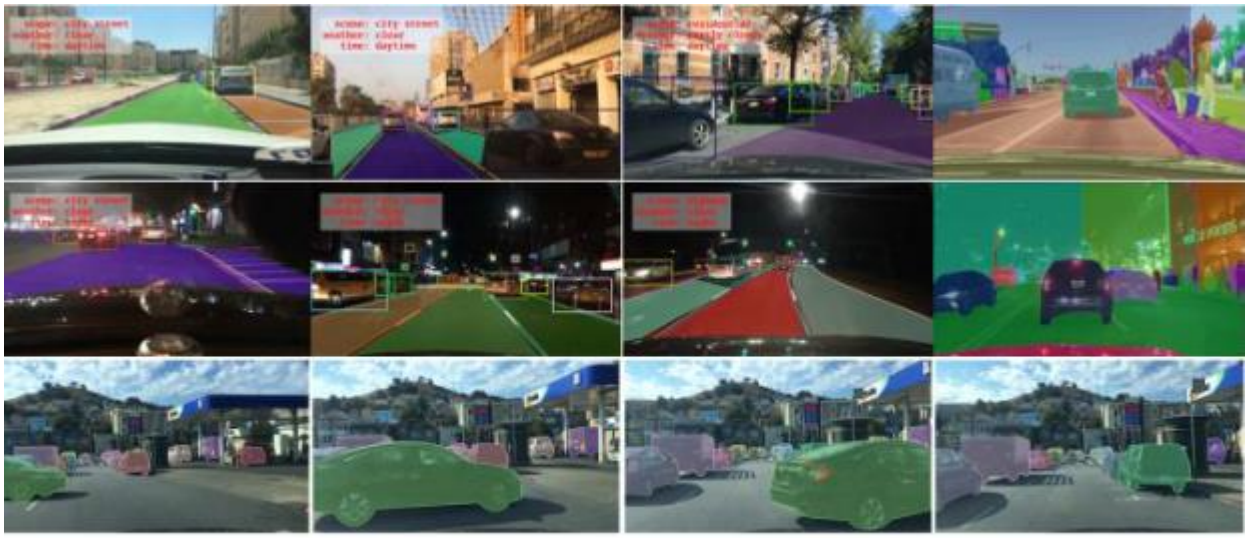


Рисунок 3.4 - Приклад зображень з BDD100K

Як показана на Рисунок 3.12 є різні типи зображення та типи мітки для виявлення об'єктів та сегментації. Для даної роботи використовувалися 100k images та Detection 2020 labels. Detection 2020 містить покращені мітки порівняно з оригінальними, які знаходяться в Labels.

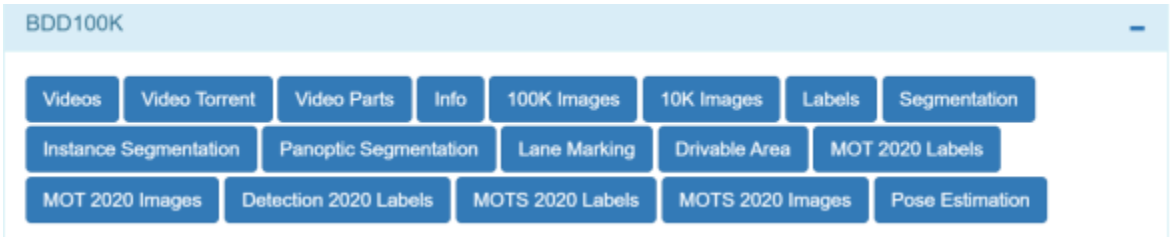


Рисунок 3.12 - Дані з бази даних-BDD100K

Зображення в Detection 2020 розподілені на сімдесят тисяч тренувальних, двадцять тисяч валідаційних та решта -для тестування. Мітки містяться в двох json файлах. Перший для тренувальних зображень, другий - валідації. Координати

боксів вказані у форматі близькому до Pascal VOC, а саме точки верхнього лівого та нижнього кут бокса. У python для одної рамки це реалізовано наступним чином:

```
x, y, x2, y2 = bbox
x_center = (x + x2/ 2) / image_width
y_center = (y + y2 / 2) / image_height
yolo_width = x2/ image_width
 yolo_height = y2 / image_height
```

Налічується 13класів об'єктів, а саме, світлофор, дорожній знак, автомобіль, пішохід, автобус, вантажівка, водій, велосипед, мотоцикл, поїзд, інший транспортний засіб, інша людина та трейлер.

Оскільки задана модель повторює YOLO, то вона працює із відповідним форматом, який містить назву моделі, та складається з назви класу, дві координати на центр боксу та його ширина і довгота. Для цієї операції пишеться код, який зчитує дані з відповідних файлів, визначає відповідні змінні та записує їх в текстовий txt файл , який містить ім'я зображення. Для простішої ітерації ім'я файлів з мітками та зображення записуються в csv файли для тренування та валідації.

При оригінальному маркуванні даних у 2020 році кількість об'єктів у датасеті було розподілено наступним чином (див. рисунок 3.13).

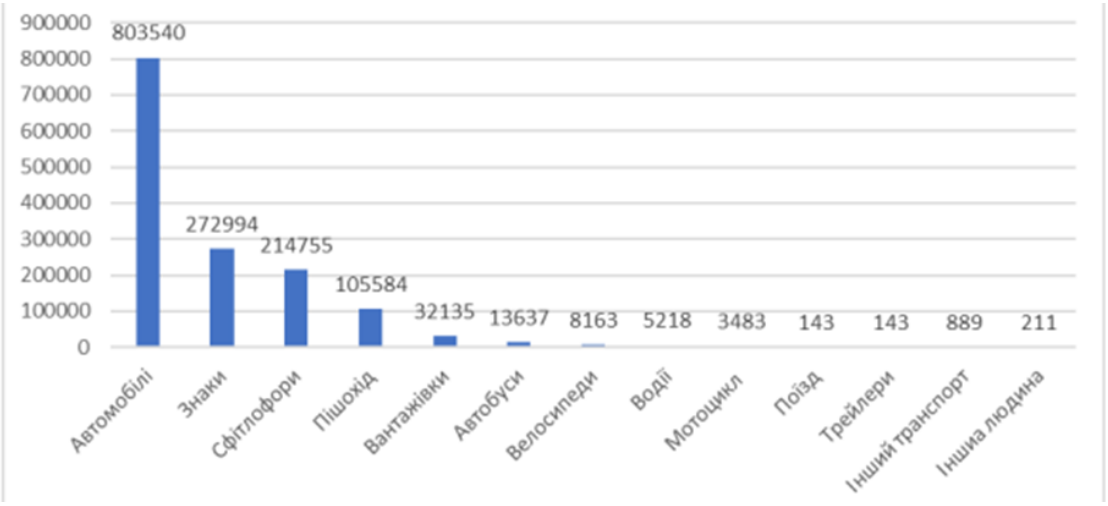


Рисунок 3.13 – Розподіл по кількості об’єктів в датасеті.

Завантаження баз даних відбувається через відповідний клас PyTorch torch.utils.data.Dataset. Датасет окремо робиться для тренувальних, валідаційних та тренувальних даних. Вхідний параметр для них це шляхи до відповідних папок із зображеннями, мітками та csv таблицею, розмір зображення, яке має бути на виході, функції трансформації, ancher box та розмір сітки. Оскільки спайкові мережі використовують багато ресурсів на тренування, розмір тренувальних даних зменшено до п'ятнадцяти тисяч, а валідаційні - не застосовуються при тренування. Це саме стосується розмір пакетів , який рівний - 19. Як було зазначено dataset приймає функцію трансформації. Вона займається перетворення зображення, зменшує розмір, доповнює залишкові частини зображення нулями, редагує кольоровий фільтр, перетворює в torch масив та нормалізує його. Крім цього, після перевірки координат відбувається їх перетворення до матриці, яка використовується під час тренування. Також відбувається перевірка чи дійсно координати в правильному форматі.

Для завантаження даних на графічний процесор та тренування використовується завантажувач даних. Використання завантажувача даних спрощує процес доступу та переміщення даних з різних джерел в центральне місце, таке як хмарний склад даних. Завантажувач даних підтримує швидке, обсягне завантаження даних. Він прискорює обробку даних, дозволяючи завантажувати практично будь- які дані, у будь-якому форматі, з будь-якої системи, будь-якого обсягу та швидкості. Також автоматично адаптується до змін вихідних даних та схеми для отримання даних у реальному часі.

Його функціональні можливості:

- Планування - завантажувач даних повинен надавати можливість планування завдань при створенні або редагування існуючі завдання для запуску за розкладом для платформ, таких як Snowflake, Google BigQuery, Azure Synapse, AWS Redshift та Databricks. Вибереться часовий пояс та частоту для планування, з можливістю налаштування відповідно до змін потреб бізнесу.

- Пакетне завантаження даних є важливим у багатьох випадках, наприклад, під час операції "підняти та перемістити". Завантажувач даних Informatica використовує паралельну обробку для ефективного виконання пакетних завантажень від джерела до цілі. Він пропонує більше 30 коннекторів для різних джерел даних та хмарних цілей, таких як Snowflake, Databricks, Google BigQuery, Azure Synapse та Amazon Redshift.

- Автоматизація відіграє важливу роль в різних аспектах завантаження даних, включаючи відправку сповіщень та відстеження використання. Автоматизується надсилання сповіщень на кілька електронних адрес зі статусом виконання завдань - чи це невдача, попередження чи успіх. Функція автоматизованого відліку дозволяє відстежувати використання, надаючи інформацію щодо кількості оброблених рядків під час циклу. Це усуває необхідність ручного відстеження та надає зрозумілий огляд даних, оброблених з різних джерел до цілей.

- ETL (Видобуток, Трансформація, Завантаження) - надійний завантажувач даних повинен сприяти процесу ETL. Завантажувач даних від Informatica дозволяє настроювати та докладати зусилля щодо джерела даних, використовуючи власні властивості. Крім того, можна виключати конкретні поля з кожного об'єкта в одному завданні, визначати власні фільтри та додавати первинні ключі та поля водяного знаку для проведення завантаження до цілі.

Завантажувач даних використовує розмір пакетів - 19. Під час ітерації вони повертають як вхідні дані (зображення) так і вихідні (мітки). Модель та ці дані необхідно закріпити на графічному процесорі для швидкого тренування.

3.3. Інтелектуальна система моніторингу зовнішніх об’єктів для безпілотних транспортних засобів в АПК на основі розробленої спайкової моделі

Для перетворення моделі в першу чергу необхідно змінити активатори. Для цього використовується активатор, подібний до LeakyRelu, а саме Leaky. Leaky - це модель нейрона із простою інтеграцією та витоком першого порядку. Припускається, що вхід - це ін’єкція струму. Мембранний потенціал експоненційно спадає зі швидкістю beta, якщо:

$$U[T] > U_{thr} \Rightarrow S[T+1] = 1 \tag{3.1}$$

де:

U – мембрамбний потенціал,

Uthr – мембранний поріг.

Також

$$U[t+1] = \beta U[t] + I_{in}[t+1] - RU_{tr} \tag{3.2}$$

де:

U – мембрамбний потенціал

U_{thr} – мембранний поріг

B - швидкість розпаду потенціалу мембрани

I_{in} – вхідний струм

R – механізм скидання, якщо активний R=1, інакше R=0

Якщо механізм скидання ("reset_mechanism") = 0, то буде встановлюватися 0 кожного разу, коли нейрон видає спайк:

$$U[t+1] = \beta U[t] + I_{syn}[t+1] - R(\beta U[t] + I_{in}[t+1]) \tag{3.3}$$

Він використовує параметр beta для встановлення швидкості розпаду мембрани. Також за замовчуванням активатор повертає вихідний спайк, який містить дані та наступний мембранний потенціал для кожного елемента в партії.

Оскільки не використовується мембранний потенціал, в активаторі додається опція init_hidden=True, яка вказує, що потрібно повертати лише спайк.

У фінальному вигляді даний шар має наступний вигляд

snn.Leaky(beta=0.7,init_hidden=True),

де

- beta (float або torch.tensor) - швидкість розпаду мембранного потенціалу обмежена від 0 до 1 під час прямого проходження. Може бути одновимірним тензором (тобто з однаковою швидкістю розпаду для всіх нейронів у шарі) або багатовимірним (один ваговий коефіцієнт на нейрон).
- init_hidden (bool, необов'язковий) - Ініціалізує змінні стану як змінні екземпляра. За замовчуванням встановлено False.

Для можливості тренування необхідно також необхідно мати snntorch utils.reset() функцію. Вона допомагає скинути внутрішні приховані параметри до нуля. Без неї тренування неможливе. Це потрібно як і в циклі тренування, так і в блоках моделях.

Таким чином, проведено детальний аналіз створення моделі YOLOv3 з використанням архітектури Darknet-54 для виявлення об'єктів на зображеннях та відео. Модель використовує різні типи блоків, такі як Convolution (згортковий), Residual (залишковий) та scaleprediction, для ефективного створення вихідних сигналів.

Для ефективного навчання моделі використовуються різні функції втрат, такі як nn.BCEWithLogitsLoss() для параметрів л noobj, nn.MSELoss() для X_obj та л coord, і nn.CrossEntropyLoss() для X_class. Це дозволяє моделі навчатися з високою точністю та ефективністю. Оптимізатор optim.Adam з PyTorch використовується для оптимізації параметрів моделі.

База даних BDDK100 виступає в якості набору даних для тренування та тестування, забезпечуючи різноманітність умов та сценаріїв, які можуть виникнути у реальному світі.

Для досягнення високої точності виявлення об'єктів використовуються допоміжні функції, такі як iou (Intersection over Union), nms (non-maximum suppression) та convert_cells_to_bboxes. Функція iou важлива для обчислення міри подібності між областями, що сприяє оцінці точності моделі. Функція nms відділяє впевнені області, щоб уникнути подвійного виявлення. Функція convert_cells_to_bboxes перетворює вихідні дані моделі у формат областей (bounding boxes), що полегшує аналіз та візуалізацію результатів.

Узагальнюючи, робота над моделлю YOLOv3 використовує комплексний підхід до виявлення об'єктів у візуальних даних, використовуючи потужність згорткових та залишкових блоків, а також ефективні функції втрат для навчання моделі.

Дана інформація також включає в себе опис створення LIF snn моделі за допомогою бібліотеки snntorch, що базується на PyTorch. Перетворення моделі в snn включає зміни в активаторах, зокрема, використання Leaky замість LeakyRelu. Описані параметри, такі як швидкість розпаду мембранного потенціалу та інші, дозволяють ефективно моделювати поведінку нейрона.

Для можливості тренування необхідно використовувати функцію reset() з snntorch utils, яка допомагає скинути внутрішні параметри до початкового стану. Втрати, оптимізатор та завантажувач даних підтримуються обома моделями без необхідності внесення змін.

Втрати, оптимізатор та завантажувач даних підтримуються обома моделями і не потребують жодних модифікацій. Це зручно, оскільки не треба робити зайвих змін в системі.

Для тренування та тестування моделей використовувалося Google Colab. Colab є версією Jupyter Notebook, яка пропонується як сервіс, що дозволяє писати та виконувати Python-код через свій веб-браузер.

Jupyter Notebook є безкоштовним, відкритим та створеним у рамках проекту Jupyter. Jupyter Notebook схожий на інтерактивний науковий журнал, який

включає не лише нотатки та дані, але й код, який може обробляти ці дані. Код можна виконувати безпосередньо в межах блокноту, який, в свою чергу, може фіксувати результати виконання коду. Подібний підхід вперше впроваджений в програмах, таких як Matlab та Mathematica, але на відміну від них, Jupyter є веб-застосунком, заснованим на браузері.

Google Colab базується на коді проекту Jupyter і господарює блокноти Jupyter без необхідності встановлення локального програмного забезпечення. Проте, хоча блокноти Jupyter підтримують кілька мов, включаючи Python, Julia та R, Colab наразі підтримує лише Python. Вигялд даних блокнотів можна побачити на Рисунок 3.14 – 3.15.

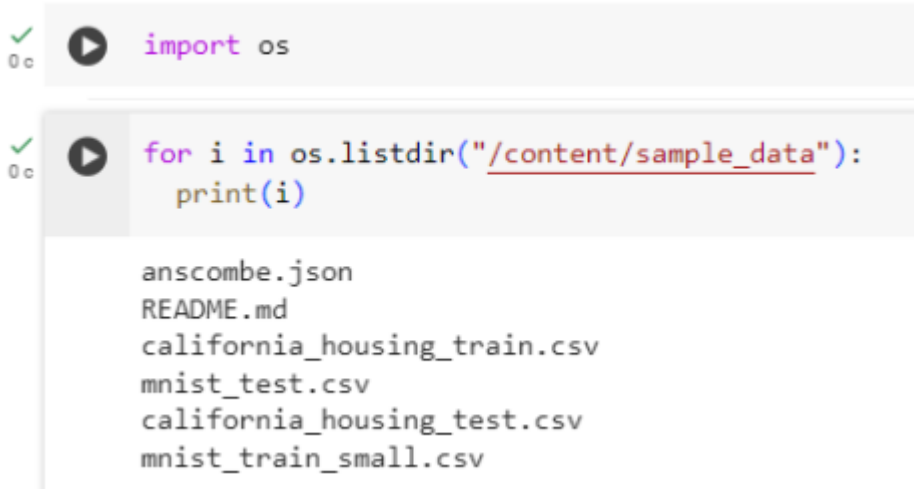


Рисунок 3.14 – Вигляд блокноту Colab

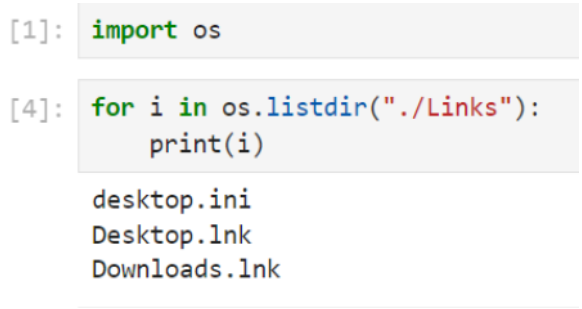


Рисунок 3.15 – Вигляд блокноту Jupyter

Блокноти Colab зберігаються в обліковому записі Google Drive та можуть

бути спільно використані іншими користувачами, аналогічно як і для інших файлів Google Drive. У блокнотах також існує функція автоматичного збереження, але вони не підтримують одночасне редагування, тому спільна робота здійснюється послідовно, а не паралельно.

Colab є безкоштовним, але має свої обмеження. Деякі типи коду, такі як обслуговування медіа та криптовалютний майнінг, заборонені. Ресурси обмежені і змінюються залежно від попиту, хоча Google Colab пропонує платну версію з надійнішими ресурсами. Вартість даних послуг показано в таблиці 3.1

Таблиця 3.1

Тарифний план Google Colab

Тарифний план	Ціна
Pay as Go	Безкоштовний доступ до лімітованих ресурсів
Colab Pro	11, 99
Colab Pro+	55,99
Colab Enterprise	Не фіксована ціна

За тарифним планом Pay As You Go оплата здійснюється лише за фактичне використання обчислювальних одиниць. Термін їх дії 90 днів. Якщо потрібно, можна докупити додаткові одиниці. Підписка не обов'язкова, і оплата здійснюється лише за реальне використання. Цей тариф надає можливість переходити на більш потужні графічні процесори за потреби.

Colab Pro пропонує 100 обчислювальних одиниць щомісяця, і термін їх дії також становить 90 днів. При необхідності можна докупити додаткові обчислювальні одиниці. Цей тариф рекомендований для тих, хто шукає більше обчислювальної потужності. Користувач може переходити на більш потужні графічні процесори за своїми потребами. Також цей тариф надає доступ до машин з найвищим обсягом пам'яті.

Colab Pro+ містить усі переваги тарифного плану Pro і додає ще 500 обчислювальних одиниць на місяць (усього 400). Термін їх дії також складає 90 днів, і за потреби їх можна докупити. :Користувач отримує пріоритетний доступ для оновлення графічних процесорів до більш потужних преміум-версій. За допомогою обчислювальних одиниць можна обробляти дані ще протягом 24 годин, навіть після закриття веб-переглядача.

Colab Enterprise - повністю інтегровано із сервісами Google Cloud, такими як BigQuery і Vertex AI. Замість записників Google Drive клієнтиі використовують записники GCP, які зберігаються і доступні для передавання в хмарні консолі. Включає також виконання і генерацію коду за допомогою штучного інтелекту.

Дана робота виконується за безкоштовним планом Pay as Go. Вона пропонує графічний процесор Tesla T4.

Tesla T4 - це професійна відеокарта від NVIDIA, заснована на технології 12 нм і заснована на графічному процесорі TU104 у його варіанті TU104-895-A1, карта підтримує DirectX 12 Ultimate. Графічний процесор TU104 - це великий чіп з площею кристала 545 мм2 та 13 600 мільйонами транзисторів. Його вигляд показано на Рисунок 3.16.

Рисунок 3.16. Зовнішний вигляд відео карти Tesla T4.

Карта має 2560 шейдерних блоків, 160 блоків текстурного відображення і 64 блоки ROP.

Також процесор містить 320 тензорних ядр, які покращують швидкість застосувань машинного навчання. Карта також обладнана 40 ядрами прискорення для трасування променів. NVIDIA поєднала 16 ГБ відео пам'яті GDDR6 з Tesla T4, які з'єднані за допомогою 256-бітного інтерфейсу пам'яті.

Графічний процесор працює на частоті 585 МГц, яку можна підняти до 1590 МГц, а пам'ять працює на частоті 1250 МГц (ефективна швидкість 10 Гбіт/с). Завдяки своєму однослотовому дизайну, відеокарта не вимагає додаткового живлення, її споживана потужність оцінюється на максимальний рівень 70 Вт.

Оцінка тренування SNN моделі в задачі виявлення і моніторигу зовнішних об'єктів щодо беспілотного автотранспорту в АПК.

При початку тренування можна побачити різницю в використанні ресурсів. Це видно на Рисунок 3.17 – 3.18. Згідно них спайкова мережа використовує вдвічі більше графічної пам'яті. Також ці два результати є більшими, ніж результати для оригінальної YOLOv3, яка вимагає лише 4 Гігабайт пам'яті/

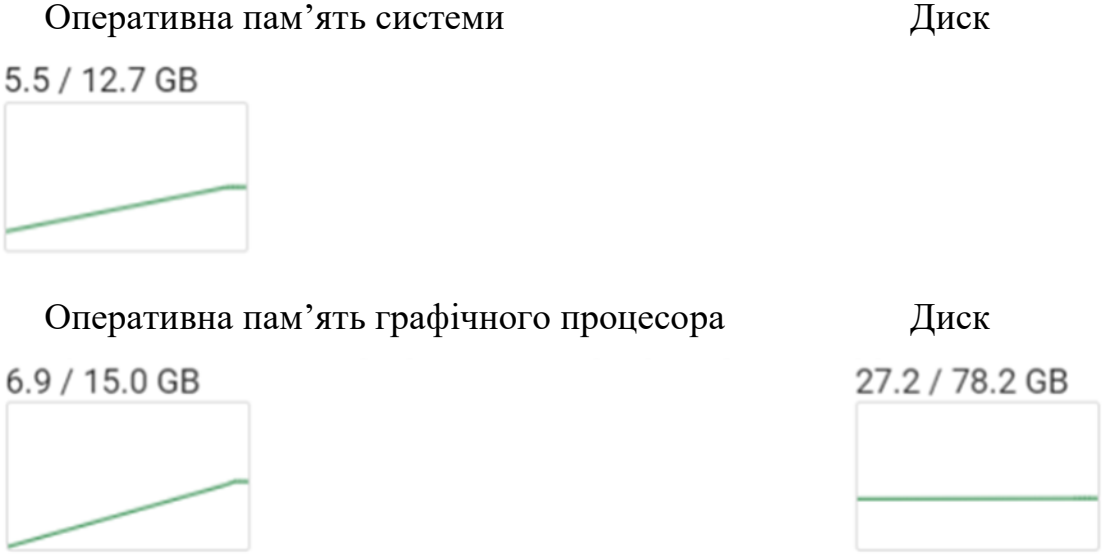


Рисунок 3.17 - Використання апаратних ресурсів при тренуванні моделі PyTorch

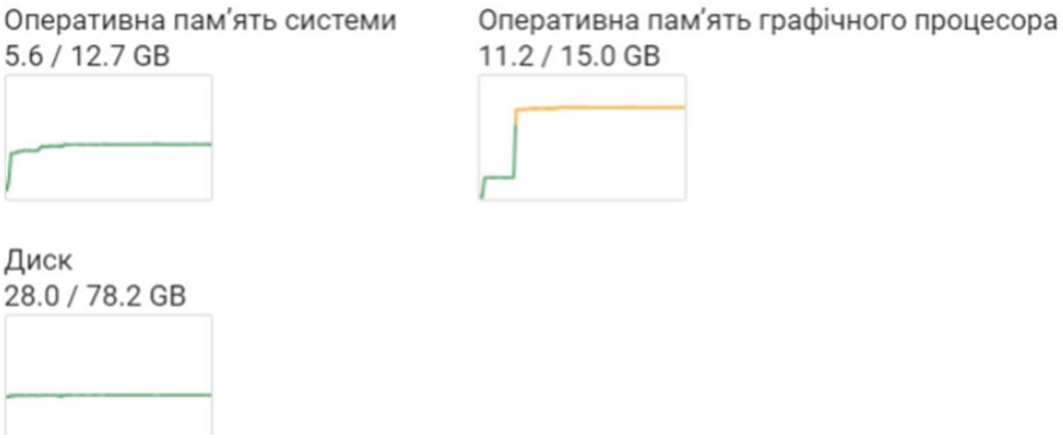
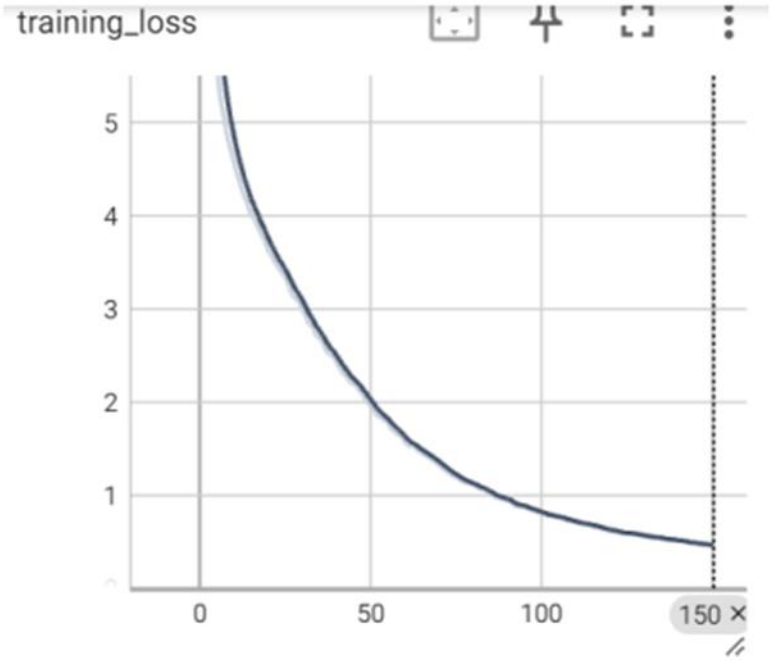


Рисунок 3.18 - Використання апаратних ресурсів при тренуванні моделі snntorch

При тренуванні втрати обчислюються однаково. Вони записуються в логи TensorBoard.

TensorBoard - це відкритий інструментарій, який дозволяє вивчати прогрес тренування та оптимізувати продуктивність моделі, шляхом внесення змін у гіперпараметри. У наборі інструментів TensorBoard реалізовано інтерактивний інтерфейс, де можна візуалізувати журнали у вигляді графіків, зображень, гістограм, тексту та інших компонентів. Крім того, він сприяє відстеженню різних показників, таких як градієнти, втрати, метрики та проміжні виводи.

Їх кількість показана на Рисунок 3.19 – 3.20. Втрати нотувалися однаково за допомогою бібліотеки TensorBoard.



Run ↑	Smoothed	Value	Step	Relative
.	0,4684	0,4621	150	2.26 hr

Рис. 3.19 - Втрати під час тренування моделі PyTorch

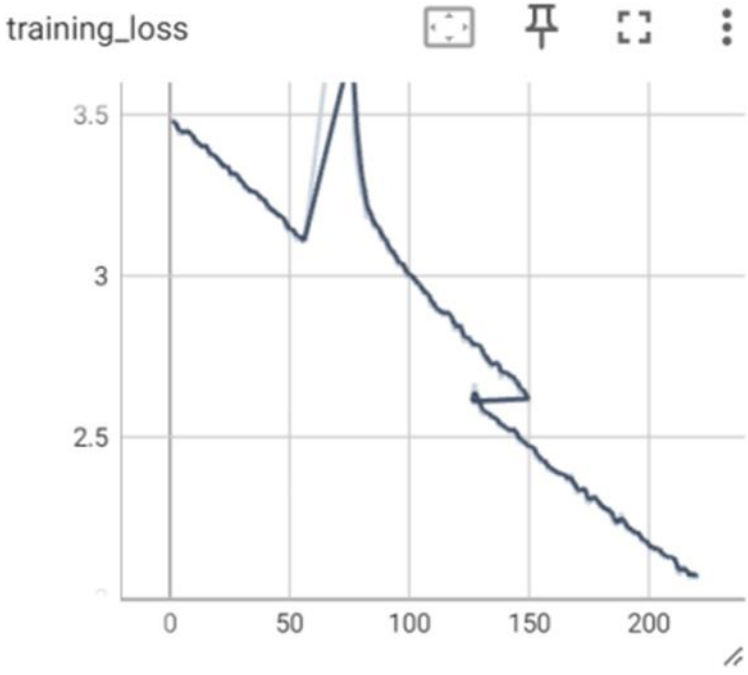


Рис. 2.20 - Втрати під час тренування snntorch

З цього можна зробити висновок, що для моделі PyTorch втрати на початку є вищі, але досить швидко вони опускаються до позначки 0,4. Спайкову мережу SnnTorch треба ще тренувати, для неї 180 епох виявилось недостатньо.

Проведемо порівняння роботи розробленої моделі з існуючими аналогами.

Моделі порівнюються за наступними параметрами.

- FPS - частота, з якою послідовні зображення (кадри) захоплюються або відображаються на екрані. Вона вимірюється в кадрах на секунду (FPS) і характеризує швидкісь роботи моделі.

FPS дорівнює:

$$FPS = \frac{1}{t} \tag{3.4}$$

де t - середній час обробки фотографії моделлю.

За допомого t час можна знайти як:

```
fps=[]
for i, data, in enumerate(custom_dataset2)
        x_val = data[0].to(device)
        x_val=x_val.unsqueeze(0)
        a=time.time() output=model(x_val) b=time.time() fps.append(b-a)
sum(fps)/len(fps)
```

Цей код працює для обох моделей. Однак, для згорткової моделі він повернув значення 0.05, а для іншої - 0.07.

Розрахувавши FPS за заданою формулою отримаємо данимі представлені в таблиці 3.2.

Таблиця 3.2.

Характеристики натренованих моделей YOLOv3

Параметр	PyTorch	SnnT
FPS	20	14

За цими даними можна оцінити FPS для виявлення в реальному часі. Мінімальний рівень, який підходить для виявлення мережі складає 15 фреймів. Спайковій мережі трохи не вистачає швидкості для проходження цього порогу.

Для виявлення зображення необхідно отримані дані перетворити передбачені координати у координати для створення графіку бібліотекою matplotlib.pyplot. Результат можна побачити на Рисунок 3.21-3.22. Оскільки спайкова мережа гірше навчена, її дані менш точні.

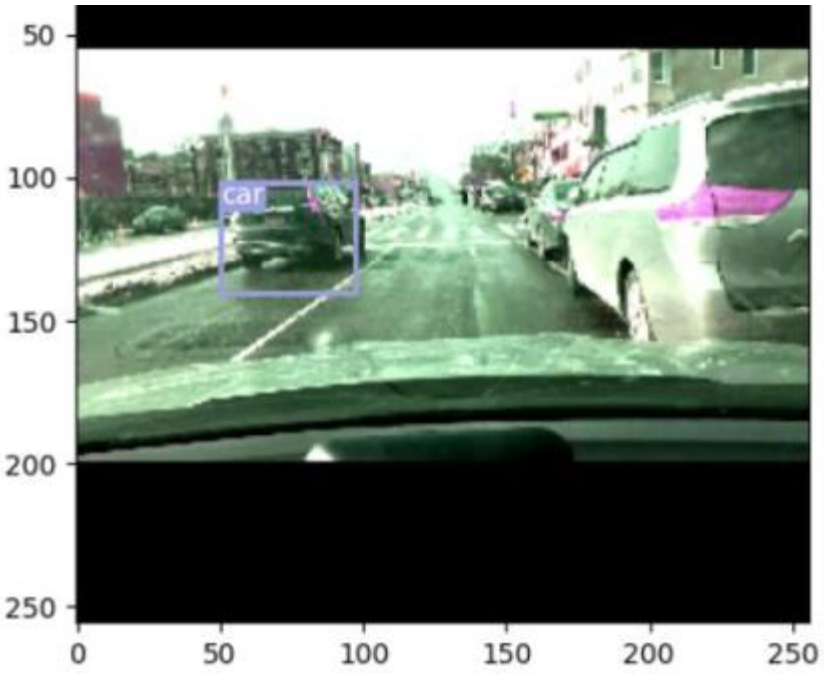


Рисунок 3.21 – Результат роботи моделі PyTorch.

Рисунок 3.22 - Результат роботи моделі SnnTorch.

Результати використання апаратних ресурсів при використанні моделі PyTorch представлені на графіках (див. рисунок 3.23.)

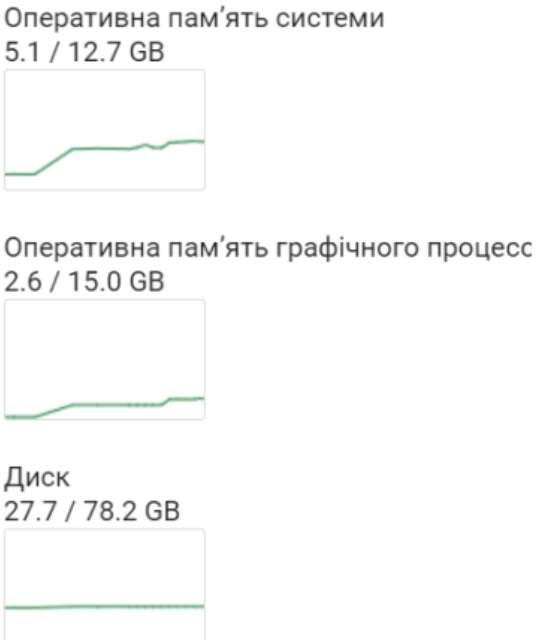


Рисунок 3.23 - Графік використання апаратних ресурсів при використанні моделі PyTorch.

Тут на точність могли попливати розмір зображення 256x256, які є досить малими, але економлять ресурси під час тренування. Оригінальний розмір який виокристовували під час тренування є 416x416. Крім того, слід зазначити, що в оригіналі FPS YOLOv3 при таких розмірах зображення була 22 FPS . У роботі падіння FPS пояснюється тим, що мережа виконує більш складні обчислення, ніж згорткова мережа. В результаті цього вона витрачає більше часу на обробку зображення, що призводить до зниження FPS.

Також представлені графіки використання апаратних ресурсів під час виконання тестування спайкової мережі (див. Рисунок 3.24).

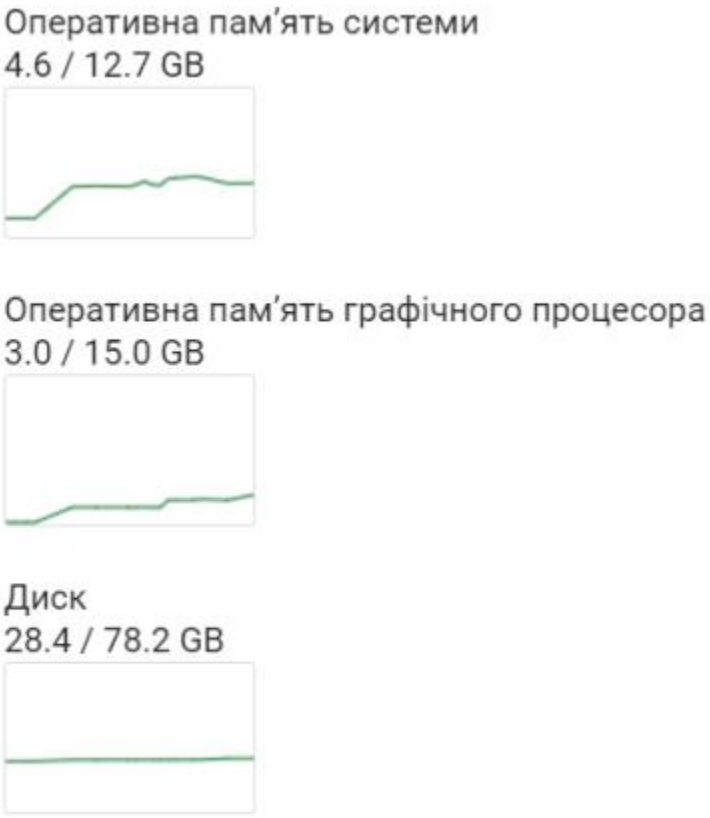


Рис.3.24 Графік використання апаратних ресурсів для спайкових мереж.

З цих графіків можна зробити висновок, що моделі використовують приблизно однаково кількість пам'яті графічного процесора. Але це не вказує, що їх електроспоживання теж однакове.

Аналіз електроспоживання. Розробка програмного забезпечення для систем автопілоту транспортниз засобів АПК, яке ефективно використовує енергію, є важливою частиною стратегій зменшення споживання електроенергії, особливо в умовах обмежених ресурсів у сфері транспортної техніки.

Найточніший і найменш впливаючий на результати є метод вимірювання споживання енергії з використанням ватт-годинного лічильника, який безпосередньо підключений до розетки. Цей підхід дозволяє вимірювати загальне споживання електроенергії пристрою, дозволяючи розраховувати витрати на енергію. Оцінюючи напряму електропостачання, цей метод надає коиплексне та надійне вимірювання, роблячи його вибірковим для отримання точних даних про споживання енергії та відповідних розрахунків вартості.

Крім даного методу є ще різні іннструментальні програмні рішення.

- Інструмент Intel Power Gadget виступає як програмний засіб моніторингу використання електроенергії, спеціально розроблений для процесорів Intel Core від другого до сьомого покоління. Зазначимо, що процесори Intel Atom не підтримуються. Цей інструмент, сумісний як з операційною системою Windows, так і з Mac OS X, включає додаток, драйвер та бібліотеки для моніторингу та оцінки поточної інформації про споживання електроенергії процесорного пакету в ватах, використовуючи лічильники енергії в самому процесорі. Останнє оновлення розширює його функціональність для оцінки інформації про енергоспоживання на різних платформах, включаючи ноутбуки, настільні комп'ютери та сервери.

- Powerstat - це ще один інструмент, який вимірює споживання електроенергії комп'ютера від джерела живлення батареї або підтримує інтерфейс RAPL (Running Average Power Limit). Подібно до vmstat, вивід powerstat включає статистику споживання електроенергії. Після завершення роботи powerstat обчислює середнє значення, стандартне відхилення, мінімум, максимум та геометричне середнє зібраних даних. Запуск powerstat в якості кореневого

користувача надає додаткову інформацію про активність процесів fork(2), exec(2) та exit(2).

- PowerTOP - це ще один потужний інструмент для вимірювання та моніторингу споживання енергії, що має важливу перевагу - здатність оцінювати споживання енергії в пристроях з процесорами AMD. Його основна функція полягає в вимірюванні споживання енергії, але PowerTOP виходить за рамки цього, надаючи інтерактивний режим. У цьому режимі користувачі можуть точно налаштовувати параметри управління енергією в їхній системі Linux. Хоча ця функція іноді може здатися складною, PowerTOP виступає як комплексна та цінна альтернатива для оптимізації ефективності споживання енергії в системах на процесорах AMD.
- Likwid, інший інструмент, який використовує інтерфейс RAPL, розроблений Intel, отримує вимірювання енергії та потужності з різних областей ЦП, забезпечуючи швидкий і простий доступ до даних зі споживання енергії від процесорів Intel.
- Для відеокарт Nvidia є інструмент nvidia-smi командного рядка, який надає інформацію про споживання електроенергії разом з іншими показниками, такими як використання пам'яті, температура та інше. Це особливо корисно для вимірювання споживання енергії.

Оскільки Colab містить лімітований доступ до апаратної частини, а на хостовій машині встановелно AMD процесор з графічною карткою AMD, на якому не вдалося встановити всі необхідні бібілотеки для підтримки тренування, вичислити її енергоспоживання не вдалося.

Загалом, отримані результати підтверджують, що, незважаючи на те, що витрати на тренування в PyTorch можуть бути трошки вищими у порівнянні з іншими платформами, це виправдовується високою гнучкістю та простотою використання цієї бібліотеки. Використання спайкових мереж дозволяє досягти

певного поліпшення продуктивності, що впливає на точність та здатність моделі працювати з великими обсягами даних.

Навіть при збільшених витратах на тренування, швидкість кадрів на секунду залишається на прийнятному рівні, а використання ресурсів під час тестування не суттєво зростає. Це свідчить про ефективність отриманої моделі та можливість успішного впровадження її у реальне середовище.

Для тренування та тестування моделей ми використовували Google Colab, що є версією Jupyter Notebook і пропонується як веб-сервіс. Colab надає зручний інтерфейс для написання та виконання Python-коду, використовуючи ресурси Google, а самі блокноти зберігаються в обліковому записі Google Drive. Використання Colab є безкоштовним, проте є обмеження та платні тарифні плани для більшого доступу до ресурсів та функціоналу.

Зазначимо, що у 2022 році була випущена версія YOLO (You Only Look Once) v7, яка є однією з найсучасніших і ефективних реалізацій сімейства YOLO для задач виявлення об'єктів. Версія v7 містить ряд покращень у порівнянні з попередніми версіями, зокрема YOLOv3, YOLOv4, YOLOv5 та YOLOv6. Основними перевагами її є YOLOv7 наступні.

- Удосконалена Рекурсивна модульна структура архітектури. В YOLOv7 інтегровані покращені блоки, такі як ELAN (Extended ELAN), які дозволяють ефективніше розподіляти та комбінувати інформацію між шарами, що покращує навчання глибоких моделей; нова архітектура "підсиленого передавання" інформації, яка забезпечує краще поєднання локальних і глобальних особливостей, що підвищує її продуктивність у складних задачах розпізнавання.
- Підвищена швидкість обробки з меншою кількістю обчислювальних витрат на аналогічному апаратному забезпеченні, забезпечуючи швидке виявлення об'єктів у реальному часі за рахунок GPU-оптимізаці і вдосконаленню використання CUDA та TensorRT, модель працює більш ефективно на сучасних графічних процесорах

- YOLOv7 підтримує мультизадачність, таких як виявлення, сегментація та класифікація, що робить її універсальною для складних задач комп’ютерного зору з тонким налаштуванням для специфічних задач.

- Ефективність у використанні ресурсів у різноманітних моделях що підходять для пристроїв з обмеженими ресурсами (наприклад, мобільних телефонів чи вбудованих систем) за рахунок зменшення обсягу параметрів, хоча YOLOv7 при цьому досягає високої точності порівняно з сучасними моделями.

- Адаптивне навчання анотацій (label assignment) за рахунок механізму "адаптивного призначення міток", що дозволяє покращити обробку анотацій об'єктів різного розміру і здійснювати ефективне масштабування моделі:

- Застомування сучасні технологій оптимізації з використанням спеціальних допоміжних механізмів навчання, які підвищують стабільність навчального процесу та знижують ймовірність перенавчання та краще зберігання попередньо навчених знанньь, що робить її ідеальною для задач transfer learning.

- Кросплатформенність та зручність використання за рахунок підтримки широкого спектру фреймворків, таких як PyTorch і TensorFlow, що спрощує інтеграцію в існуючі системи.

- Зручність адаптації завдяки відкритому коду та зрозумілому дизайну, YOLOv7 легко модифікується для різних задач комп’ютерного зору.

- Вища продуктивність на COCO Benchmark, про що свідчить досягнення 56.8 mAP (mean Average Precision) при 702 FPS (кадрів за секунду), що перевищує показники YOLOv3, YOLOv4, YOLOv5, та інших конкурентів.

Таким чином, YOLOv7 відрізняється високою продуктивністю, точністю та ефективністю. Вона є ідеальним вибором для задач виявлення об’єктів у реальному часі завдяки своїй універсальності, низькому енергоспоживанню та зручності використання. Але для розуміння методу і процесів виявлення та моніторингу зовнішних об’єктів в беспілотних транспортних засобах АПК достатньо дослідити версію YOLOv3.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Нормативно-документи з охорони праці та безпеки в надзвичайних ситуаціях в АПК

Національна система нормативно-правових актів України з охорони праці та безпеки в надзвичайних ситуаціях включає в себе закони, постанови, накази та інші акти, які регулюють права та обов'язки працівників і роботодавців щодо забезпечення безпечних умов праці і захисту від надзвичайних ситуацій. Ось декілька основних нормативно-правових актів у цій галузі:

Закон України "Про охорону праці" (від 14 грудня 1992 року № 2694-XII) - цей закон встановлює загальні принципи та вимоги щодо охорони праці в Україні.

Закон України "Про надзвичайні ситуації та станом надзвичайної ситуації" (від 21 грудня 1992 року № 2693-XII) - цей закон регулює організацію та управління діяльністю в галузі захисту населення та територій від надзвичайних ситуацій.

Закон України "Про цивільний захист" (від 5 лютого 1993 року № 3206-XII) - цей закон визначає порядок організації цивільного захисту та заходи щодо захисту населення від надзвичайних ситуацій.

Закон України "Про працю" (від 10 грудня 1971 року № 322-VIII) - цей закон встановлює основні права та обов'язки працівників і роботодавців, включаючи вимоги до охорони праці та безпеки на робочому місці.

Постанова Кабінету Міністрів України "Про затвердження Порядку розслідування нещасних випадків на виробництві та професійних захворювань" (від 23 жовтня 1996 року № 1248) - ця постанова визначає процедуру розслідування нещасних випадків на виробництві та професійних захворювань.

Постанова Кабінету Міністрів України "Про затвердження Положення про організацію та проведення заходів з охорони праці" (від 10 грудня 2003 року № 1913) - ця постанова встановлює загальні вимоги до організації та проведення заходів з охорони праці в підприємствах та організаціях.

Накази Державної служби України з надзвичайних ситуацій (ДСНС) та інших відповідних органів, які регулюють конкретні аспекти безпеки та охорони праці в різних сферах діяльності.

Це лише загальні приклади нормативно-правових актів, які стосуються охорони праці та безпеки в надзвичайних ситуаціях в Україні. При вирішенні конкретних питань, пов'язаних з цими питаннями, важливо враховувати чинне законодавство та консультуватися з фахівцями з охорони праці та безпеки.

4.2. Фактори, що впливають на організацію охорони праці та безпеку в надзвичайних ситуаціях при автоматизації системи моніторингу безпілотного автотранспорту АПК

Охорона праці та безпека в надзвичайних ситуаціях важливі в контексті автоматизації системи моніторингу безпілотного автотранспорту АПК, оскільки забезпечення безпеки робочого середовища та готовності до можливих надзвичайних ситуацій є основними аспектами в управлінні ризиками. Основними факторами, які мають бути враховані при організації охорони праці та безпека в надзвичайних ситуаціях для процесів сертифікації медіа контенту є наступними:

- визначення та розробка процедур охорони праці для працівників, що працюють у сфері безпілотних автотранспортних засобів АПК
- навчання персоналу щодо правил та процедур безпеки
- оцінка та управління ризиками для ідентифікації потенційних небезпек та розробка стратегій їх управлінням
- визначення заходів безпеки та проактивна реакція на ризики

- розробка та впровадження планів надзвичайних ситуацій для випадків, таких як пожежі, аварії, природні катастрофи та інші небезпечні події
- тренування персоналу щодо ефективного реагування на надзвичайні ситуації
- регулярна перевірка та технічне обслуговування обладнання, яке використовується в процесі автоматизації системи моніторингу безпілотного автотранспорту АПК
- забезпечення безпеки робочого середовища, включаючи вентиляцію, освітлення та інші параметри
- впровадження заходів забезпечення безпеки персональних даних, оскільки у процесі автоматизації системи моніторингу безпілотного автотранспорту АПК може використовуватися чутлива інформація
- регулярне навчання персоналу з охорони праці та безпеки в надзвичайних ситуаціях
- створення системи навчання та тестувань для перевірки реакції персоналу на різні сценарії розвитку подій ризику
- визначення та впровадження процедур, що забезпечують безпеку під час сертифікації медіа-ресурсів, зокрема при використанні спеціалізованого обладнання
- установлення контактів та механізмів співпраці з екстреними службами та організаціями надзвичайного реагування
- розробка планів відновлення діяльності та відновлення робочого процесу після надзвичайних ситуацій
- впровадження та дотримання стандартів безпеки та регуляцій, які стосуються галузі медіа-ресурсів.

Запропоновані заходи сприяють створенню безпечного та здорового робочого середовища для тих, хто займається атоматизацією системи моніторингу

безпілотного автотранспорту АПК, та допомагають зменшити ризики в надзвичайних ситуаціях.

РОЗДІЛ 5
ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ВИЯВЛЕННЯ ТА МОНІТОРИНГУ ЗОВНІШНИХ ОБ'ЄКТІВ ДЛЯ БЕЗПІЛОТНИХ ТРАНСПОРТНИХ ЗАСОБІВ В АПК НА ОСНОВІ РОЗРОБЛЕНОЇ СПАЙКОВОЇ МОДЕЛІ

5.1. Обґрунтування економічної доцільності використання спайкових нейронних мереж для інтелектуальної системи виявлення та моніторингу зовнішних об'єктів для безпілотних транспортних засобів в АПК

Нейронні мережі є об'єктом інтенсивного вивчення в контексті розвитку систем штучного інтелекту. Один із перспективних напрямків у цій області - це використання Спайкові Нейронних Мереж , що наслідують поведінку біологічних нейронів. Особливість SNN полягає у їхній здатності ефективно обробляти тимчасову інформацію, оскільки кожен нейрон випромінює сплеск лише тоді, коли його мембранний потенціал досягає порогового значення.

SNN мають безліч переваг. Одне з них було вказано раніше, а саме робота з тимчасовою інформацією. Це надає млжливість обробляти динамічні та зашумлені входи краще, ніж іншим моделям як CNN, які покладається на фіксовані функції активації.

Попри це SNN є складнішим і менш зрозумілим, ніж CNN, що ускладнює його розробку, навчання та оптимізацію. Для ефективної роботи SNN також потрібне спеціалізоване апаратне та програмне забезпечення, оскільки не всі готові рішення пристосовані до архітектури даної моделі.

З економічної точки зору, SNN має компроміси між продуктивністю та вартістю навчання. SNN має потенціал для досягнення вищої точності , меншої затримки роботи та споживання електроенергії. Попри це вона вимагає великі ресурси на навчання моделі.

Даний продукт буде виконуватися на NVIDIA Jetson AGX Xavier . Це рішення для розгортання та тестування моделей комп'ютерного зору для автопілоту та роботів. Вона містить файлову систему подібну до Linux. Вона дозволяє встановлювати додаткові бібліотеки, як snntorch для SNN. Крім цього є вже встановлені бібліотеки для моделей, сенсорів. Також він підтримує SSD для збільшення розміру вбудованої пам'яті в 30 гігабайтів. Для тестування буде достатньо камери реєстратора яка передаватиме зображення з автомобіля подібне до тестових файлів. Модель буде тренуватися в хмарі з великою кількістю gpu. З цим буде отримана можливість створювати великі і точні моделі.

5.2. Розрахунок прибутку від розробки та зменшення коштів порівняно з аналогами інтелектуальної системи виявлення та моніторингу зовнішних об'єктів для безпілотних транспортних засобів в АПК

Очікуваний прибуток - це прибуток який очікується від розробки та збереження коштів від аналогів.

Згідно з дослідженнями IBM Research можна припустити, SNN зменшить споживання електроенергії на 80%. Це може призвести до економії до кількох тисяч гриванів щорічно (залежно від вартості електроенергії та обсягу використання). Це стосується не тільки використання енергії при споживання готового продукту але й ефективного роботою з низькою кількістю даних, що може призвести до зекономлених витрат на збір та обробку даних. Зекономлені кошти теж рівні кільком тисяч гривням. Якщо взяти що ціна за оренду моделі 30000 і вважати, що є 10 покупців, то очікуваний прибуток буде розраховуватися так:

$$D_{\text{міс}} = 30000 * 10 = 300000$$

За рік вона буде рівна:

$$D_{\text{рік}} = 300000 * 12 = 360000$$

$$T_{\text{ок}} = \frac{K_{\text{кап}} + K_{\text{екс}}}{D_{\text{рік}}} \approx 3 \text{ років}$$

Можна зробити висновок, що даний проект фінансувати недоцільно.

Оскільки він містить великі витрати як капітальні та експлуатаційні, а термін окупності становить 3 роки.

ВИСНОВКИ

Виявлення та моніторинг зовнішних об'єктів визначається як ключова складова для забезпечення ефективності та безпеки інтелектуальної системи для безпілотних транспортних засобів в АПК

Стандарт SAE J3016 визначає рівні автоматизації автопілоту від ручного керування до повної автоматизації в різних умовах експлуатації.

Аналіз аналогічних досліджень розкриває досягнення в області виявлення об'єктів за допомогою специфічних моделей, які використовуються у безпілотних літальних апаратах (БПЛА) з нейромережами. У роботі, присвяченій використанню YOLOv3 з Spatial Pyramid Pooling для виявлення об'єктів за допомогою БПЛА, виділяється покращена ефективність моделі, особливо в стійкості до різних масштабів зображень. Це вказує на перспективи впровадження даної технології в реальних умовах.

Дослідження моделі Spiking-YOLO приводить до висновку, що використання спайкових нейронів може стати ключовим елементом для створення енергоефективних систем виявлення об'єктів. Важливо відзначити, що ця модель не лише досягає порівнянних результатів із традиційними моделями виявлення об'єктів, але також вирізняється значно меншим споживанням енергії. Це може мати важливе значення для розвитку нейромережних технологій, особливо з огляду на потребу в енергоефективних рішеннях у сферах автономних систем та вбудованих пристроїв.

Помічено також високі результати моделі FSHNN, що використовує гібридний підхід із навчанням на основі STDP та навчанням зворотнім розповсюдженням. Ця модель демонструє інноваційний підхід до об'єктного виявлення в реальному часі та досягає вражаючих результатів, перевершуючи стандартні методи, такі як RetinaNet.

Усі ці висновки свідчать про динамічний розвиток та потенціал використання різноманітних підходів та моделей для виявлення об'єктів. Проте в контексті техніко-економічної ефективності важливо враховувати витрати та переваги кожної конкретної моделі, зокрема їхній вплив на капітальні та експлуатаційні витрати для досягнення оптимальних результатів в сферах застосування

Методи виявлення об'єктів можна класифікувати на одноетапні та двоетапні. Перші, представлені YOLO та SSD, забезпечують швидке виявлення об'єктів без додаткових етапів, тоді як другі, такі як R-CNN та Faster-RCNN, використовують два етапи для більш точного визначення та класифікації.

Застосування нейронних мереж, зокрема згорткових нейронних мереж виявляється найбільш ефективним для обробки візуальних даних. Структура нейрона, інспірована біологічними нейронними мережами, дозволяє виробляти як лінійні, так і нелінійні реакції.

Було розглянуто роботи, подібні до даної наукової роботи, їх особливості та які результатів вони змогли досягти.

У даній роботі було розглянуто модель YOLOv3 як ефективний алгоритм виявлення об'єктів. Використання Darknet-54 забезпечує високу продуктивність та точність виявлення. Незважаючи на це, виявлено недолік функції втрат IoU, що призводить до представлення DistanceIoU (DIoU) як покращеної альтернативи.

Модель Ходжкіна-Хакслі і модель LIF представляють біологічно точні, але різні підходи до моделювання спайкових нейронів. Модель AdEx поєднує експоненціальну залежність напруги з адаптивною змінною, що моделює поріг адаптації.

Досліджено процес тренування моделей на PyTorch та використання спайкових мереж. Отримані результати свідчать про високу гнучкість та простоту використання PyTorch, а також ефективність спайкових мереж у забезпеченні точності та здатності працювати з великими обсягами даних.

У економічному розділі були розглянуті сильні та слабкі сторони спайкової моделі для автопілотів транспортних засобів в АПК. Компенсаційні витрати, такі як обладнання, інтеграція та оплата праці, грали ключову роль у визначенні вартості системи. Важливо врахувати, що використання такої моделі вимагає значних витрат як у капітальному, так і у експлуатаційному плані, і термін окупності становить 3 роки.

Підкреслено важливість розробки програмного забезпечення для систем автопілотів транспортних засобів, що ефективно використовує енергію. Подано інформацію про різні методи вимірювання споживання енергії, включаючи використання ватт- годинного лічильника та різні програмні рішення, такі як Intel Power Gadget, Powerstat, PowerTOP, Perf, Likwid, та nvidia-smi для вимірювання споживання електроенергії компонентами системи, включаючи процесори та відеокарти.

Враховуючи висновки з аналізу тренування моделей на PyTorch та використання спайкових мереж, важливо підкреслити успіх у досягненні високої гнучкості та ефективності PyTorch, а також точності та здатності спайкових мереж працювати з великими обсягами даних. Однак, в контексті техніко-економічного аспекту, виявлено значні витрати на впровадження спайкових моделей для автопілотів, що вимагає обґрунтованого підходу та ретельного розгляду всіх аспектів для забезпечення оптимальної результативності в сфері автопілотів.

В цьому плані перспективним є застосування найновішої моделі YOLOv7, яка по своїм характеристикам перевершує усі попередні версії моделей виявлення та моніторингу зовнішніх об’єктів безпілотним автотранспортом АПК.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. [2305.12344] YOLOv3 with Spatial Pyramid Pooling for Object Detection with Unmanned Aerial Vehicles (arxiv.org)

2. https://www.labellerr.com/blog/how-object-detection-works-in-self-driving-cars-using-deep-learning/

3. https://encord.com/blog/object-detection/

4. https://scholarworks.gvsu.edu/cgi/viewcontent.cgi?article=1916&context=the ses

5. https://www.linkedin.com/pulse/single-shot-multibox-detector-ssd-ayoub-kirouane

6. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9313413/#B5-brainsci-12-
00863

7. https://cnvrg.io/spiking-neural-networks/

8. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

9. https://uk.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87%D0%B
D%D0%B0_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0 %B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0

10. https://www.google.com/url?sa=i&url=https%3A%2F%2Flivingfo.com%2Fs hcho-take-nejronni-merezhi-ta-iak-vony-pratsiuiut%2F&psig=AOvVaw34YXzFqp1TVVQsrpAX8P7l&ust=1695653510161000&s
ource=images&cd=vfe&opi=89978449&ved=0CBIQjhxqFwoTCID22Nq_w4EDFQAAA AAdAAAAABAE

11. https://uahistory.co/pidruchniki/zadorozhnij-biology-deep-level-8-class-

2021/39.php

12. https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/
13. https://arxiv.org/abs/1903.06530
14. https://arxiv.org/abs/2201.07706
15. https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html

Додаток 1

Основний блок моделі

```
class YOLOv3(nn.Module):
    def __init__(self, in_channels=3, num_classes=20):
        super().__init__()
        self.num_classes = num_classes
        self.in_channels = in_channels
        self.layers = nn.ModuleList([
            CNNBlock(in_channels, 32, kernel_size=3, stride=1, padding=1),
            CNNBlock(32, 64, kernel_size=3, stride=2, padding=1),
            ResidualBlock(64, num_repeats=1),
            CNNBlock(64, 128, kernel_size=3, stride=2, padding=1),
            ResidualBlock(128, num_repeats=2),
            CNNBlock(128, 256, kernel_size=3, stride=2, padding=1),
            ResidualBlock(256, num_repeats=8),
            CNNBlock(256, 512, kernel_size=3, stride=2, padding=1),
            ResidualBlock(512, num_repeats=8),
            CNNBlock(512, 1024, kernel_size=3, stride=2, padding=1),
            ResidualBlock(1024, num_repeats=4),
            CNNBlock(1024, 512, kernel_size=1, stride=1, padding=0),
            CNNBlock(512, 1024, kernel_size=3, stride=1, padding=1),
            ResidualBlock(1024, use_residual=False, num_repeats=1),
            CNNBlock(1024, 512, kernel_size=1, stride=1, padding=0),
            ScalePrediction(512, num_classes=num_classes),
            CNNBlock(512, 256, kernel_size=1, stride=1, padding=0),
            nn.Upsample(scale_factor=2),
            CNNBlock(768, 256, kernel_size=1, stride=1, padding=0),
            CNNBlock(256, 512, kernel_size=3, stride=1, padding=1),
            ResidualBlock(512, use_residual=False, num_repeats=1),
            CNNBlock(512, 256, kernel_size=1, stride=1, padding=0),
```

```
    ScalePrediction(256, num_classes=num_classes),
    CNNBlock(256, 128, kernel_size=1, stride=1, padding=0),
    nn.Upsample(scale_factor=2),
    CNNBlock(384, 128, kernel_size=1, stride=1, padding=0),
    CNNBlock(128, 256, kernel_size=3, stride=1, padding=1),
    ResidualBlock(256, use_residual=False, num_repeats=1),
    CNNBlock(256, 128, kernel_size=1, stride=1, padding=0),
    ScalePrediction(128, num_classes=num_classes)])

def forward(self, x):
    utils.reset(self.layers)
    outputs = []
    route_connections = []
    for layer in self.layers:
        if isinstance(layer, ScalePrediction):
            outputs.append(layer(x))
            continue
        x = layer(x)
        if isinstance(layer, ResidualBlock) and layer.num_repeats == 8:
            route_connections.append(x)
        elif isinstance(layer, nn.Upsample):
            x = torch.cat([x, route_connections[-1]], dim=1)
            route_connections.pop()
    return outputs
```