Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему *Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 2 курсу, група КНм-23-2* — Підпис — *Валерій ПОСТРИБАЙЛО* (Ім'я, ПРІЗВИЩЕ)
Курс, група виконавця

Керівник: *зав. кафедри КН, д.т.н., проф.* — Підпис — *Олександр БАРМАК* (Ім'я, ПРІЗВИЩЕ)
Науковий ступінь, посада

Нормоконтроль *к.т.н., доцент кафедри КН* — Підпис — *Руслан БАГРІЙ* (Ім'я, ПРІЗВИЩЕ)
Науковий ступінь, посада

До захисту допускаю:

зав. кафедри КН, д.т.н., професор — Підпис — *Олександр БАРМАК* (Ім'я, ПРІЗВИЩЕ)

10 грудня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *магістр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*
(підпис)
д.т.н., професор Олександр БАРМАК
«*02*» *вересня* *2024* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

1. Тема кваліфікаційної роботи магістра: *«Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання»*

2. Завдання видано студенту *Валерію ПОСТРИБАЙЛУ*
(Ім'я, ПРІЗВИЩЕ)

3. Керівник роботи *д.т.н., проф. Олександр БАРМАК*
(посада, Ім'я, ПРІЗВИЩЕ)

4. Затверджені наказом університету від «*26*» *серпня* *2024* р. № *60*

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета кваліфікаційної роботи магістра – підвищення якості класифікації комах-шкідників на зображеннях із зерносховищ за моделлю глибокого навчання. Для досягнення поставленої мети визначено наступні задачі: проаналізувати існуючі методи класифікації шкідників у зерносховищах на основі зображень; обрати та налаштувати архітектуру глибокої нейронної мережі для класифікації малих об'єктів, підготувати датасет для донавчання моделі, включаючи різні класи шкідників; здійснити донавчання та оптимізацію моделі для підвищення стабільності роботи; провести валідацію та аналіз отриманих експериментальних результатів за стандартними метриками якості.

Вихідні дані: вдосконалена модель, спеціалізований датасет з зображеннями комах-шкідників, набір гіперпараметрів.

Реферат

Кваліфікаційна робота магістра присвячена розробці та вдосконаленню методу класифікації комах-шкідників у зерносховищах на основі глибоких нейронних мереж, що спрямований на забезпечення точного та стабільного виявлення шкідників за зображеннями в умовах реального часу.

Актуальність теми. Сучасні досягнення у сфері комп'ютерних наук, зокрема у галузі штучного інтелекту та глибокого навчання, активно інтегруються у різноманітні галузі, від медицини до промисловості. Одним із напрямків, де ці технології демонструють значний потенціал, є сільське господарство. Використання моделей глибокого навчання дозволяє автоматизувати складні процеси обробки та аналізу даних, які раніше вимагали значних людських ресурсів та часу. Одним із прикладів таких задач є виявлення та класифікація шкідників зернових культур під час зберігання.

Традиційні методи контролю якості зернових культур часто не дозволяють своєчасно і надійно виявляти наявність шкідників, що може призвести до значних економічних втрат. Впровадження сучасних методів машинного навчання, зокрема глибоких нейронних мереж, дозволяє автоматизувати процес моніторингу стану зерносховищ, забезпечуючи високу точність та швидкість класифікації шкідників. Це сприяє зменшенню втрат продукції та зниженню потреби у використанні хімічних засобів для боротьби з шкідниками.

Мета і задачі роботи. Метою роботи є підвищення якості класифікації комах-шкідників на зображеннях із зерносховищ за моделлю глибокого навчання. Для досягнення поставленої мети потрібно виконати наступні завдання:

- провести аналіз: предметної області для проблеми виявлення шкідників у зерносховищах; існуючих публікацій щодо класифікації шкідників у зерносховищах; існуючих програмних рішень та моделей глибокого навчання для виявлення об’єктів малого розміру на зображеннях;
- вдосконалити метод виявлення та класифікації шкідників у зерносховищах на зображеннях за моделлю глибокого навчання;
- підготувати відповідний набір даних для донавчання нейронної мережі;

– донавчити попередньо навчену нейронну мережу виявляти шкідників у зерносховищах;

– імплементувати вдосконалений метод у прототипну інформаційну систему;

– визначити якість запропонованого методу за відомими статистичними показниками та навести порівняння з іншими відомими підходами.

Об’єкт дослідження. процес виявлення та класифікації комах-шкідників на зображеннях із зерносховищ за моделлю глибокого навчання.

Предмет дослідження. Моделі глибокого навчання.

Методи дослідження. Моделі глибокого навчання для аналізу зображень з дрібними об’єктами.

Наукова новизна одержаних результатів. Вдосконалено метод класифікації комах-шкідників у зерносховищах за їх зображенням засобами глибокого навчання, який відрізняється від існуючих донавчанням моделі на спеціалізованих наборах даних, додаванням шарів Dropout для контролю за перенавчанням та оптимізацією гіперпараметрів, що дозволило працювати з меншим об’ємом даних, більшою швидкодією, підвищити точність класифікації шкідників при різних умовах фотофіксації, частковому перекритті об’єктів та збільшити кількість видів комах-шкідників, які загрожують зерносховищам.

Апробація результатів кваліфікаційної роботи магістра та публікації.
За темою роботи підготувлені тези до наукової конференції: Бармак О.В., Пострибайло В.О. Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання. Збірник наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2024». - Хмельницький, 2024. - С. 425-427. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2024-corpuspaper.pdf

Структура та обсяг роботи. Кваліфікаційна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 51 найменування та 3 додатків. Загальний обсяг кваліфікаційної роботи магістра становить 124 сторінки, з них 86 сторінок основного тексту та 38 сторінок додатків. У роботі наведено 21 рисунок, 37 формул та 9 таблиць.

Ключові слова: класифікація шкідників, зерносховища, глибокі нейронні мережі, YOLOv8, донавчання, оптимізація гіперпараметрів, регуляризація, набір даних, точність класифікації.

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
CNN	Convolutional Neural Network
CPU	Central Processing Unit
F1 Score	F-measure (метрика гармонічного середнього між точністю та повнотою)
GPU	Graphics Processing Unit
ROC	Receiver operating characteristic
SGD	Stochastic Gradient Descent
YOLO	You Only Look Once
YOLOv8	You Only Look Once version 8
CNN	Convolutional Neural Network

Вступ

Актуальність теми. Сучасні досягнення у сфері комп'ютерних наук, зокрема у галузі штучного інтелекту та глибокого навчання, активно інтегруються у різноманітні галузі, від медицини до промисловості. Одним із напрямків, де ці технології демонструють значний потенціал, є сільське господарство. Використання моделей глибокого навчання дозволяє автоматизувати складні процеси обробки та аналізу даних, які раніше вимагали значних людських ресурсів та часу. Одним із прикладів таких задач є виявлення та класифікація шкідників зернових культур під час зберігання.

Традиційні методи контролю якості зернових культур часто не дозволяють своєчасно і надійно виявляти наявність шкідників, що може призвести до значних економічних втрат. Впровадження сучасних методів машинного навчання, зокрема глибоких нейронних мереж, дозволяє автоматизувати процес моніторингу стану зерносховищ, забезпечуючи високу точність та швидкість класифікації шкідників. Це сприяє зменшенню втрат продукції та зниженню потреби у використанні хімічних засобів для боротьби з шкідниками.

Мета і задачі роботи. Метою роботи є підвищення якості класифікації комах-шкідників на зображеннях із зерносховищ за моделлю глибокого навчання. Задачами роботи є проведення аналізу предметної області для проблеми виявлення шкідників у зерносховищах; існуючих публікацій щодо класифікації шкідників у зерносховищах; існуючих програмних рішень та моделей глибокого навчання для виявлення об’єктів малого розміру на зображеннях. Також задачею роботи є вдосконалення методу виявлення та класифікації шкідників у зерносховищах на зображеннях за моделлю глибокого навчання та підготування відповідного набору даних для донавчання нейронної мережі, донавчання попередньо навченої нейронної мережі виявляти шкідників у зерносховищах. Окрім цього задачами роботи є імплементування вдосконаленого методу у прототипну інформаційну систему та визначення якості запропонованого методу за відомими статистичними показниками, наведення порівняння з іншими відомими підходами.

Об’єкт дослідження процес виявлення та класифікації комах-шкідників на зображеннях із зерносховищ за моделлю глибокого навчання.

Предмет дослідження: Моделі глибокого навчання.

Методи дослідження. Моделі глибокого навчання для аналізу зображень з дрібними об’єктами.

Наукова новизна одержаних результатів. Вдосконалено метод класифікації комах-шкідників у зерносховищах за їх зображенням засобами глибокого навчання, який відрізняється від існуючих донавчанням моделі на спеціалізованих наборах даних, додаванням шарів Dropout для контролю за перенавчанням та оптимізацією гіперпараметрів, що дозволило працювати з меншим об’ємом даних, більшою швидкодією, підвищити точність класифікації шкідників при різних умовах фотофіксації, частковому перекритті об’єктів та збільшити кількість видів комах-шкідників, які загрожують зерносховищам.

Апробація результатів кваліфікаційної роботи магістра та публікації.
За темою роботи підготувлені тези на наукову конференцію: Бармак О.В., Пострибайло В.О. Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання. Збірник наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2024». - Хмельницький, 2024. - С. 425-427. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2024-corpuspaper.pdf

Структура та обсяг роботи. Кваліфікаційна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 51 найменування та 3 додатків. Загальний обсяг кваліфікаційної роботи магістра становить 124 сторінки, з них 86 сторінок основного тексту та 38 сторінок додатків. У роботі наведено 21 рисунок, 37 формул та 9 таблиць.

Ключові слова: класифікація шкідників, зерносховища, глибокі нейронні мережі, YOLOv8, донавчання, оптимізація гіперпараметрів, регуляризація, набір даних, точність класифікації.

Розділ 1

Аналіз сучасного стану у використанні методів глибокого навчання для класифікації шкідників у зерносховищах за зображеннями

1.1 Аналіз предметної області

Проблема шкідників у зерносховищах є однією з головних загроз для сільського господарства, що може призвести до серйозних економічних втрат. Основні шкідники, такі як Sitophilus granarius (зерновий довгоносик), Rhyzopertha dominica (зерновий точильник) і Tribolium castaneum (червоний борошноїд), не тільки пошкоджують зерно на різних етапах його зберігання, але й можуть суттєво зменшити його обсяги та знизити якість. Це призводить до фінансових втрат для фермерів і підприємств, що займаються зберіганням і торгівлею зерном. Окрім цього, збільшення застосування хімічних засобів для боротьби з комахами загрожує забрудненням навколишнього середовища і продуктів харчування, а також підвищує виробничі витрати.

Запровадження сучасних технологій для виявлення та ідентифікації шкідників на основі глибокого навчання значно покращує процеси боротьби з комахами в зерносховищах. Використання системи автоматизованого виявлення дозволяє швидко та точно визначати присутність шкідників на ранніх стадіях зараження. Це надає можливість своєчасно вживати заходи для запобігання їхньому поширенню, що в результаті зменшує втрати продукції та витрати на боротьбу з шкідниками.

Система на базі штучного інтелекту забезпечує автоматизований моніторинг стану зерносховищ без постійної участі людини. Вона здатна аналізувати великий обсяг даних про стан зерна та наявність шкідників, що дозволяє не тільки швидко реагувати на зараження, але й прогнозувати ризики появи шкідників у майбутньому. Це допомагає раціонально керувати умовами зберігання та уникати непередбачених втрат.

Одним із ключових переваг таких систем є можливість зниження залежності від хімічних засобів захисту, адже точне визначення виду шкідників дозволяє

застосовувати більш екологічно безпечні методи боротьби або обмежити використання пестицидів лише в необхідних зонах. Це не тільки знижує негативний вплив на довкілля, а й сприяє збереженню якості продукції.

Окрім того, автоматизовані системи моніторингу дозволяють виявляти різні типи шкідників на різних етапах їх розвитку, що робить процес боротьби більш гнучким та адаптивним. Наприклад, виявлення первинних шкідників, таких як Sitophilus granarius, дозволяє запобігти появі вторинних, які зазвичай з'являються після пошкодження зерна. Це сприяє збереженню довгострокової якості продукції та покращує умови зберігання.

1.2 Аналіз існуючих публікацій щодо класифікації шкідників у зерносховищах

Оскільки проблема не нова, довкола неї встиг зібратися великий пласт різних досліджень та рішень, які сильно різняться своїми методами та ефективністю. Неефективність ручного виявлення шкідників описано у статті [1], стаття показує що ручне виявлення шкідників ефективне лише за умови їх розміщення на листках і коли шкідник великого розміру, якщо ж шкідники не великі і не на поверхні, а тим паче вже в середині зібраного врожаю, ефективність виявлення значно менше.

Досліджуючи питання різних новітніх підходів до цієї проблеми, стаття [2] аналізує сучасний прогрес у методах виявлення шкідників серед зернових культур. Огляд включає традиційні методи, такі як візуальний огляд, пастки для шкідників та інші, але основний акцент зроблено на новітніх технологіях. Зокрема, розглядаються машинний зір, тепловізійна зйомка, аналіз метаболітів, молекулярні техніки (PCR, ДНК-баркодинг), біоакустичний аналіз, а також спектроскопія ближнього інфрачервоного випромінювання і гіперспектральна візуалізація. Порівнюючи більш сучасні підходи, підкреслюється значна перевага у зручності та ефективності більш сучасних рішень.

Специфічність моніторингу шкідників і проблематика старих методів дослідження підкреслена у статті [3], де наголошується ускладнення і інколи повна

неможливість відслідковування і боротьби зі шкідниками через їх постійне пересування, умови зберігання зерна, розмноження, різні стадії життя та умови зберігання продовольства.

Стаття [4] розглядає виявлення шкідників та їхнього впливу на зовнішнє середовище, яке вони собою заражають. Підкреслено ефективність та високу точність машинного зору, аналізу метаболітів, біоакустичного аналізу та термографії, але недоліком цих методів зазначене дороговартісне спеціалізоване обладнання і необхідний попередній досвід у застосуванні описаних методів.

У статті [5] про сучасний підхід виявлення та класифікації шкідників у зерносховищах Китаю розповідається про цілу національну систему моніторингу та раннього виявлення шкідників. Поєднання сучасних технологічних методів показало високу ефективність сучасних методів не зважаючи на їх дороговартісне обслуговування. Загострення необхідності максимальної оптимізації процесів моніторингу та класифікації шкідників ускладнилося нещодавньою пандемією та геополітичними кризами, про що розповідається у статті [6]. У статті [7] порівнюють акустичні, візуальні, та молекулярні методи з методами штучного інтелекту та зазначаються найвищу ефективність, високу оптимізацію та точність саме методів на основі машинного навчання.

Стаття [8] шукає підхід до актуальних викликів в сільському господарстві, серед яких пандемія, шкідники, неврожайність, зростання населення зміни клімату, та проблеми збирання і обробки великих, складних даних. Поєднання сенсорних технологій та безпілотних платформ у поєднанні з машинним навчанням показали значні результати у зборі величезної кількості високоякісних зображень полів та їх аналізі включаючи картування земель, класифікацію культур, виявлення та класифікацію шкідників, моніторинг біотичного/абіотичного стресу та прогнозування врожаю. У статті [9] оглядаються сучасні недеструктивні технології, які використовуються для виявлення пошкоджень плодів і овочів шкідниками під час пост-урожайних умов зберігання. В огляді розглянуто кілька методів, включно зі спектроскопією, зображенням, акустичними сенсорами та хімічними взаємодіями, з особливим акцентом на недеструктивні методи.

Деякі підприємства мають досвід використання спеціальних акустичних систем для боротьби зі шкідниками. Стаття [10] розповідає про досвід користування такими системами, необхідне обладнання та досвід роботи з такими ще з 1900-х років і показує високу їх ефективність та екологічну безпеку. У статті [11] розповідається, що акустичні системи дають доволі непогані результати у виявленні личинок та дорослих рисових хробаків у зерні. Проте автори статті зазначають, що використання залишається обмеженим через високу вартість пристроїв і необхідність спеціальних навичок для інтерпретації даних. У статті [12] автори оптимізували використання акустичних пристроїв і показали що можна застосовувати більш дешеве і просте обладнання, проте їх рішення оптимізоване лише під зовсім невеликі сховища. У статті [13] створена спеціальна акустична система яку можна поміщати безпосередньо у зерно і яка інформує користувача за допомогою пошти про найменші ознаки зараження зерна, проте розробники зіткнулися з проблемою занадто шумного аграрного середовища, що ускладнює диференціацію дуже тихих звуків шкідників.

Автори статті [14] також підкреслюють ефективність та перевагу акустичних методів, особливо для виявлення личинок комах, які харчуються зерном, оскільки їх важко помітити традиційними методами. Але такі методи виявлення також доволі дорого та складно обслуговувати. Використання гіперспектральних сенсорних систем в статті [15] показує їх зручність в інтеграції з мобільними пристроями та оперативність реагування на перші прояви захворювання рослин та запасів зерна.

Метод Раманової спектроскопії в статті [16] презентує себе як доволі чутливий спосіб аналізу хімічного складу та структури різних зразків. Метод добре ідентифікує різні види комах, їх вік та стать. Розвиток методу потребує тісної міждисциплінарної співпраці між ентомологами, хімометриками та спектроскопістами для покращення аналізу складних зразків та визначення невідомих компонентів, що забезпечить кращу основу для боротьби з шкідливими комахами. Стаття [17] показує можливість оптимізації і дешевого обслуговування гіперспектрального методу, але варто наголосити, що метод використовувався лише на шкідниках, що були розташовані на листі рослин, а не безпосередньо на зібраному зерні. Інтеграція БПЛА та гіперспектральна зйомка у статті [18] показала високу точність прогнозування до

85%. Але варто зауважити що умови зйомки на БПЛА у відкритому полі і в умовах зберігання зерна сильно різняться. Використання ближньої інфрачервоної спектроскопії у статті [19] показало майже стовідсотковий результат під час ідентифікації відібраних видів шкідників, метод гарно працює з деталями і не заважає збереженню культур. З недоліків методу, знову ж таки висока собівартість.

Використання технології електронного носа (E-nose), яка імітує систему нюху тварин, відносно нещодавно стало доволі перспективним напрямком і описане у статті [20]. Ця технологія дозволяє здійснювати недеструктивне виявлення, має високу чутливість, низьку вартість, можливість реального аналізу, простоту в експлуатації та зручність у транспортуванні. Електронний ніс здатний виявляти летючі органічні сполуки (VOC), які рослини виділяють під час поширення шкідників, для залучення природних ворогів шкідників або відлякування їх. У статті також розглядається використання датчиків металевих оксидів (MOS) у складі E-nose, які мають переваги в чутливості та низькій ціні. Метод класифікації шкідників за допомогою їх специфічних запахів і феромонів висвітлено у статті [21]. Метод дозволяє проводити ранню фіксацію, є екологічно чистим і може бути інтегрованим у більшу систему, проте роботі методу може перешкоджати постійна необхідність калібрування, вплив фонових запахів та технічні обмеження. У статті [22] описують систему раннього виявлення шкідників, яка використовує технології цифрових камер, сенсорних мереж та віддаленого моніторингу в поєднанні з методами машинного навчання. Обговорюється використання ближньої інфрачервоної спектроскопії (NIR) та недорогого електронного носа (e-nose) для виявлення присутності шкідників та їх взаємодії з рослинами. Результати показали високу точність моделей класифікації (96,5–99,3% для NIR та 94,2–99,2% для e-nose).

Стаття [23] пропонує розробку інверсної задачі, яка враховує біологічні особливості, для створення карт присутності шкідників, використовуючи тимчасові сигнали від мережі сенсорів феромонів. В інверсну задачу вводяться біологічні знання шляхом специфічного штрафу, що використовує залишки рівнянь динаміки популяцій. Автори показали, що інверсні задачі, які використовують дані з мережі сенсорів феромонів і включають біологічну інформацію як регуляризаційний

елемент, дозволяють підвищити точність прогнозування та виявлення шкідників на ранніх стадіях.

В основі системи зі статті [24] лежить використання електронного носа (e-nose) для аналізу запахів, що виділяються Fall Armyworm в навколишнє середовище. Крім того, використовуються технології обробки зображень, зокрема сегментація зображень, для виявлення наявності шкідника через пошкодження які вони залишають. У статті [25] обговорюється використання технології електронного носа (E-nose) для виявлення шкідників у зберіганні рису. В експерименті використовувався прилад E-nose (Cyranose 320) з 32 сенсорами для виявлення летких компонентів, які виділяються в зараженому рисі. Результати показали, що E-nose не зміг відрізнити чистий рис від зараженого рисом злаковим жуком, але продемонстрував високу реакцію на леткі компоненти, які виробляються рисовим жуком та червоним борошняним жуком після чотирьох тижнів зберігання. Таким чином, E-nose зміг розрізнити чистий і заражений рис лише через чотири тижні зберігання при температурі 30 °C.

Стаття [26] пропонує використовувати різні технології газового датчика для аналізу летючих органічних сполук, які виділяються рослинами в умовах стресу. Аналіз летючих органічних сполук (ЛОС), які виділяють рослини та інші біологічні матеріали, може бути застосований для виявлення шкідників у зерні, що зберігається оскільки під час зараження комахи можуть також продукувати специфічні летючі речовини. Це дозволяє використовувати електронний ніс (e-nose) для раннього виявлення інфекцій чи шкідників, схожим чином, як він застосовується для рослин.

Цікаве дослідження, що спирається на температурні показники [27] було проведено у Китаї. Середня точність правильного визначення проблем з якістю та інвентаризацією зерна склала 94%.

Особливе місце серед всіх існуючих методів в останні роки займають методи з використанням штучного інтелекту та глибокого навчання. Стаття [28] показує потенціал та перевагу підходів з інтегрованим штучним інтелектом та глибоким навчанням перед традиційними та іншими сучасними підходами. У [29] автори провели дослідження 92 публікації, опублікованих в період з 2016 по 2022 рік. З

акцентом на класифікацію, моніторинг та виявлення комах у пастках. Вони виділили два основні підходи: стандартний і адаптований для виявлення шкідників, а також розглянули різні архітектури та детектори. Основні виклики, згадані у статті, включають обмеження алгоритмів для малих об'єктів і незбалансованість даних.

В цій статті [30] автори підкреслюють машинне і глибоке як методи які дозволяються максимально оптимізувати моніторинг та класифікацію шкідників без впливу людського фактору та інших сторонніх подразників. У статті [31] наведено огляд нових трендів у виявленні шкідників із використанням нейронних мереж, підкреслюючи переваги та виклики цієї технології. Особлива увага приділяється використанню ансамблів нейронних мереж, баз даних шкідників, сучасного програмного забезпечення та інноваційних архітектур для підвищення ефективності виявлення, підкреслює важливість технологій комп'ютерного зору для раннього виявлення шкідників у збережених зернах, що є особливо актуальним у країнах, що розвиваються.

У статті [32] аналізуються чинники, які спричиняють пошкодження зерна, а також наведені результати порівняння різних стандартних методів глибокого навчання, які можуть допомогти в контролі за поширенням комах і зменшити втрати під час зберігання. Стаття [33] оглядає методи глибинного навчання (DL) та їхнє застосування для розумного моніторингу шкідників (SPM), зосереджуючись на класифікації та виявленні шкідників на основі польових зображень. Вона описує ключові етапи обробки даних, починаючи від отримання зображень і завершуючи побудовою моделей. У статті розглядаються виклики і можливості використання DL у боротьбі зі шкідниками, а також пропонується загальна структура для майбутнього впровадження цих технологій. Дослідження статті [34] розповідає про досвід застосування таких комерційних інструментів як Plantix та See & Spray, що показали як застосування сучасних підходів може покращити роботу сільського господарства. Автори наголошуються, що попри те, що машинне навчання в аграрному секторі зараз ще на ранніх стадіях, вони показують обнадійливі результати в покращенні продуктивності сільського господарства та зменшенні шкоди для довкілля. Стаття [35] підкреслює, що використання машинного навчання не лише обіцяє покращити

ефективність моніторингу шкідників, але й зменшить залежність від хімічних пестицидів. Втрати рису від шкідників у Індії становлять близько 3,51% щороку, тому у статті [36] пропонуються методи інтегрованого управління шкідниками (IPM), які є більш екологічними та ефективними. Ці методи (фізичні, хімічні, культурні, біологічні, моніторингові, економічні) поєднують різні стратегії контролю, зменшуючи залежність від небезпечних хімікатів. Про поєднання різних методів контролю та боротьби зі шкідниками включаючи і машинне навчання пишуть у статті [37]. Зазначається, що сучасні методи приносять все кращі і кращі результати, проте потребують ще ретельного вивчення і досліджень особливо у країнах що розвиваються.

У праці [38] поєднуються штучний інтелект, технології розпізнавання зображень, сенсори довкілля та Інтернет речей (IoT) для ідентифікації шкідників. У статті [39] описують автоматизовану систему моніторингу шкідників, що покращує прогнозування шкідників в агрономії. Система поєднує автоматичний підрахунок шкідників за зображеннями та статистичний аналіз для встановлення кореляції з ручними підрахунками. Результати показали, що автоматичні підрахунки шкідників відображають схожі тенденції з ручними, хоча автоматична система має нижчий поріг виявлення, рахуючи лише видимих шкідників. Це підкреслює ефективність комп'ютерного зору у моніторингу, але і разом з тим необхідність поєднання його з іншими методами. У статті [40] пропонується установка сенсорних пристроїв з камерами на пастках, які фотографують і передають зображення в Інтернет для аналізу. Дослідження враховує погодні умови, температуру та відносну вологість. Розроблено модель машинного навчання, яка може прогнозувати появу і активність шкідників на щоденній основі з урахуванням температури повітря та вологості. Були застосовані різні алгоритми машинного навчання, і їх точність для прогнозування виявлення шкідників досягла 86,3%.

У результаті аналізу статей, розглянуто кілька основних підходів до виявлення та класифікації шкідників у зерносховищах та сільському господарстві в цілому. Традиційні методи, такі як візуальні перевірки або пастки, ефективні для великих шкідників, однак їх застосування є трудомістким і обмеженим для малих шкідників

або тих, що знаходяться всередині зерна. Вони часто не забезпечують необхідної швидкості та точності для великих площ.

Недеструктивні методи, що базуються на використанні спектроскопії, акустичних сенсорів та хімічного аналізу (наприклад, електронний ніс для аналізу летючих органічних сполук), пропонують більшу точність та швидкість виявлення шкідників. Однак ці методи можуть бути дорогими та залежати від умов навколишнього середовища, що обмежує їхню застосовність у польових умовах.

Феромонні методи є більш екологічними та безпечними, оскільки вони використовують природні хімічні сигнали для приваблення або відлякування шкідників. Вони демонструють високу ефективність, але їх обмежує потреба у постійному калібруванні та складність впровадження на великих територіях.

Найбільш перспективними є методи на основі глибокого навчання. Використання таких моделей, як YOLO та ResNet, дозволяє автоматизувати процес виявлення шкідників із високою точністю, навіть для малих або прихованих шкідників. Глибокі нейронні мережі забезпечують оптимізацію процесів і усувають вплив людського фактора. Однак, такі методи потребують значних обчислювальних ресурсів і великих наборів даних для навчання моделей.

Отже, підходи, що базуються на глибокому навчанні, демонструють найкращі результати завдяки їхній високій точності, швидкості й можливості масштабування, що робить їх найбільш ефективними для сучасного сільського господарства.

1.3 Аналіз моделей глибокого навчання для виявлення об’єктів малого розміру на зображеннях

Вибір відповідної архітектури глибокого навчання є критично важливим для досягнення високої точності і швидкості в таких задачах. Існує безліч моделей, кожна з яких має свої переваги та недоліки, що робить їх підходящими для різних сценаріїв використання. При виборі моделі важливо враховувати специфіку задачі, доступні ресурси, а також вимоги до швидкості обробки і точності.

Faster R-CNN [41] поєднує в собі точність і гнучкість, використовуючи двоетапний підхід для виявлення об’єктів. Спочатку модель генерує регіональні пропозиції за допомогою спеціального алгоритму, відомого як Region Proposal Network (RPN), який автоматично визначає області, ймовірно, що містять об’єкти. Ці пропозиції потім класифікуються та уточнюються в другому етапі, де модель використовує детальніше обчислення для визначення класу об’єкта та корекції обмежувальних рамок. Завдяки цій структурі, Faster R-CNN демонструє високу точність, особливо при виявленні об'єктів з різними розмірами та формами. Ця архітектура особливо корисна для виявлення малих об’єктів, оскільки вона може адаптуватися до різних масштабів завдяки використанню глибоких нейронних мереж. Вдосконалення архітектури, такі як впровадження Feature Pyramid Networks (FPN), дозволяє моделі зберігати деталі зображення на різних рівнях обробки, що значно підвищує точність виявлення на різних масштабах. Завдяки гнучкості у налаштуванні параметрів RPN та інтеграції з FPN, Faster R-CNN може більш ефективно справлятися з різними сценаріями, включаючи виявлення об'єктів в умовах складного фону або при низькій роздільній здатності, що робить цю модель однією з найбільш популярних у задачах комп'ютерного зору. Крім того, Faster R-CNN може бути адаптована для роботи з різними базовими мережами, такими як ResNet або ResNeXt, що дозволяє користувачам обирати оптимальні архітектури відповідно до вимог проекту. Це робить її дуже гнучкою та ефективною для використання в різних застосуваннях, від автономних транспортних засобів до медичної діагностики.

RetinaNet [42] є одноетапною моделлю для виявлення об'єктів, яка поєднує швидкість виявлення з високою точністю завдяки унікальному підходу, який включає Focal Loss. Ця функція втрат була розроблена для вирішення проблеми дисбалансу класів, яка часто виникає в задачах виявлення об'єктів, особливо коли рідкісні об'єкти можуть бути легко упущені через велику кількість негативних прикладів. Focal Loss надає більше значення рідкісним об'єктам, зменшуючи вагу легких для класифікації негативних прикладів, що робить RetinaNet особливо ефективною для виявлення малих об'єктів, які можуть бути затоплені великою кількістю фону. Модель складається з двох основних компонентів: Backbone, який відповідає за витягнення

ознак з зображення, та Head, який генерує обмежувальні рамки та прогнози класів. RetinaNet використовує багаторівневу архітектуру, що дозволяє їй виявляти об'єкти на різних масштабах. Це досягається за рахунок використання Feature Pyramid Networks (FPN), які об'єднують інформацію з різних рівнів згорткової мережі, забезпечуючи багатошаровий контекст для виявлення об'єктів. Цей підхід забезпечує гнучкість та адаптивність моделі в складних умовах, де об'єкти можуть мати різні розміри, форми та кольори. RetinaNet також демонструє високу швидкість обробки завдяки одноетапній природі, що робить її підходящою для реальних застосувань, таких як відеоспостереження, автоматизоване управління транспортом та мобільні пристрої. Додатковою перевагою RetinaNet є її простота у використанні. На відміну від моделей з двома етапами, таких як Faster R-CNN, RetinaNet може бути швидше навченим та інтегрованим у різні системи, зберігаючи при цьому конкурентоспроможну точність виявлення. Це робить RetinaNet популярним вибором серед дослідників та розробників, які прагнуть поєднати швидкість та точність у своїх проектах виявлення об'єктів.

SSD (Single Shot MultiBox Detector) [43] – це ще одна потужна одноетапна модель, яка дозволяє виявляти об’єкти за один прохід через мережу. Цей підхід спрощує процес виявлення, оскільки модель не потребує окремих етапів для генерації регіональних пропозицій, як у випадку з двоетапними моделями. SSD використовує кілька масштабів для генерації обмежувальних рамок на різних рівнях, що робить її ефективною для виявлення об'єктів різного розміру, включаючи маленькі. Архітектура SSD включає базову мережу (backbone), таку як VGG16, яка спочатку витягує ознаки з вхідного зображення. Потім модель додає кілька додаткових згорткових шарів, які виконують виявлення на різних масштабах. Це дозволяє SSD одночасно прогнозувати обмежувальні рамки та класи об'єктів, використовуючи кілька масштабів особливостей з різних рівнів. Завдяки цій структурі SSD має здатність виявляти об'єкти з різними характеристиками, що є критично важливим для складних сцен. SSD також демонструє високу швидкість обробки, що робить її придатною для застосувань у реальному часі, таких як відеонагляд, автономні транспортні засоби та робототехніка. Завдяки своїй архітектурі, яка оптимізована для

швидкого виявлення, SSD може досягати високої продуктивності навіть на обмежених обчислювальних ресурсах. Особливо важливим аспектом є здатність SSD використовувати додаткові шари для підвищення роздільної здатності виявлення малих об’єктів. Це дозволяє моделі точно розпізнавати об'єкти, які можуть бути вкрай маленькими або знаходитися на фоні, таким чином, забезпечуючи якість виявлення в умовах, де деталі є критично важливими. Завдяки своїй простоті та ефективності, SSD став популярним вибором серед дослідників та розробників, які потребують швидкого та точного виявлення об'єктів у реальному часі, без значних витрат на обчислювальні ресурси. Ця модель продовжує використовуватися в різноманітних галузях, таких як безпека, медична діагностика та автоматизація виробництва.

Mask R-CNN [44] розширює можливості Faster R-CNN, додаючи компонент сегментації для кожного об’єкта, що робить його ідеальним для завдань, де важлива точність. Ця архітектура забезпечує не тільки виявлення об'єктів, але й детальну сегментацію, яка дозволяє виділити межі об'єкта на зображенні. Це особливо корисно для виявлення малих об’єктів у складних сценах, де їх можна легко сплутати з фоном або іншими об'єктами. Mask R-CNN реалізує цю функцію сегментації через додаткову голову (branch) для кожного регіону, яка генерує маски, що представляють форми виявлених об'єктів. Це дозволяє моделі не лише визначити, де знаходиться об'єкт, але й точно охопити його контури, що підвищує точність та якість виявлення. Завдяки своїй точності та можливості адаптації під різні архітектури основи, Mask R-CNN демонструє чудові результати у виявленні об'єктів малих розмірів. Модель може використовувати різні основні мережі, такі як ResNet, для витягання ознак, а також включати Feature Pyramid Networks (FPN) для покращення здатності виявлення об'єктів різних розмірів. Ця архітектура є особливо корисною в медицині, де необхідно точно сегментувати об'єкти, такі як пухлини на медичних знімках, або в автономних системах, де точність виявлення об'єктів може впливати на безпеку. Mask R-CNN демонструє високий рівень гнучкості та адаптивності, що робить її популярним вибором серед дослідників та розробників у багатьох сферах, включаючи відеоспостереження, робототехніку та автоматизовану обробку зображень.

EfficientDet [45] – це одна з найсучасніших архітектур для виявлення об’єктів, яка використовує EfficientNet як базову модель для досягнення високої ефективності при збереженні якості. Основна мета EfficientDet полягає в оптимізації співвідношення між швидкістю, точністю та ресурсами, що робить її надзвичайно привабливою для практичного використання. Завдяки своїй збалансованій архітектурі, EfficientDet демонструє хорошу продуктивність при виявленні малих об’єктів, оптимізуючи параметри для досягнення високої швидкості та точності. Ключовим елементом EfficientDet є використання BiFPN (Bidirectional Feature Pyramid Network), яке дозволяє моделі інтегрувати ознаки з різних рівнів надійніше та ефективніше. BiFPN забезпечує двосторонню пропускну здатність, що дозволяє моделі ефективно використовувати як верхні, так і нижні шари, завантажуючи деталі та контекст зображення. Це критично важливо для роботи з об’єктами, які мають невеликі розміри на зображеннях, оскільки дозволяє покращити виявлення на різних масштабах та забезпечує точність розпізнавання навіть в умовах високої складності. Крім того, EfficientDet відзначається своєю модульністю, що дозволяє адаптувати її до конкретних завдань, включаючи різні конфігурації архітектури, щоб задовольнити вимоги різних застосувань. Ця гнучкість, разом із перевагами ефективності та продуктивності, робить EfficientDet популярним вибором у сферах, де важливо швидко обробляти зображення з високою точністю, таких як автономні транспортні засоби, безпека та промислове виробництво. Завдяки цим характеристикам, EfficientDet демонструє можливості для подальшого розвитку та вдосконалення в галузі виявлення об'єктів.

YOLO (You Only Look Once) [46] – це модель, яка здійснила революцію у виявленні об’єктів завдяки здатності обробляти зображення в реальному часі. Її унікальна архітектура розбиває зображення на сітку та для кожної клітинки визначає обмежувальні рамки та ймовірності класів. Цей підхід дозволяє швидко та ефективно знаходити об’єкти, оскільки всі обчислення виконуються за один етап, на відміну від традиційних методів, що потребують кількох стадій. Вдосконалення моделі YOLOv8, значно покращили її здатність розпізнавати малі об’єкти. Завдяки впровадженню технологій, таких як Feature Pyramid Networks (FPN), модель тепер може ефективніше

працювати з об'єктами різного масштабу. FPN зберігає деталі зображення на різних рівнях абстракції, що критично важливо для виявлення малих об'єктів, які можуть бути частково перекриті іншими.

Окрім того, нові версії YOLO також використовують такі методи, як Focal Loss, щоб зосередитися на менш представлених класах під час навчання, що робить модель більш стійкою до дисбалансу класів. Це особливо важливо при роботі з малими об'єктами, які можуть становити невелику частину загальної вибірки, адже модель, що спирається на стандартну функцію втрат, може ігнорувати ці об'єкти через їх рідкісність. Завдяки своїй швидкості, точності та адаптивності до різних умов, YOLO став популярним вибором у багатьох практичних застосуваннях, таких як відеоспостереження, автономні транспортні засоби та обробка зображень у реальному часі. Ця модель продовжує розвиватися, демонструючи перспективи для подальшого вдосконалення в галузі виявлення об'єктів.

У результаті проведеного аналізу архітектур для виявлення об'єктів, було обрано модель YOLO через її високі показники швидкості та точності, що є критично важливими для задач виявлення об'єктів у реальному часі. Використання вдосконалених технологій, таких як Feature Pyramid Networks та Focal Loss, дозволяє цій моделі ефективно справлятися з об'єктами різних розмірів, включаючи об’єкти малих розмірів. З огляду на її надійність та популярність у практичних застосуваннях, YOLO є оптимальним вибором для реалізації проекту, спрямованого на виявлення об'єктів у складних умовах.

1.4 Постановка задачі

Метою кваліфікаційної роботи магістра є підвищення якості класифікації комах-шкідників на зображеннях із зерносховищ за моделлю глибокого навчання. Для досягнення поставленої мети потрібно виконати наступні завдання:

– провести аналіз: предметної області для проблеми виявлення шкідників у зерносховищах; існуючих публікацій щодо класифікації шкідників у зерносховищах;

існуючих програмних рішень та моделей глибокого навчання для виявлення об’єктів малого розміру на зображеннях;

– вдосконалити метод виявлення та класифікації шкідників у зерносховищах на зображеннях за моделлю глибокого навчання;

– підготувати відповідний набір даних для донавчання нейронної мережі;

– донавчити попередньо навчену нейронну мережу виявляти шкідників у зерносховищах;

– імплементувати вдосконалений метод у прототипну інформаційну систему;

– визначити якість запропонованого методу за відомими статистичними показниками та навести порівняння з іншими відомими підходами.

Розділ 2
Вдосконалення методу класифікації комах-шкідників за зображенням

2.1 Опис моделі глибокого навчання для класифікації комах

Архітектура YOLOv8 забезпечує результативне виявлення об'єктів завдяки уніфікованій структурі шарів та модулів. Базуючись на принципах попередніх версій, ця модель містить удосконалення для підвищення точності обробки різних типів об'єктів. На рис. 2.1 зображена архітектура YOLOv8, яка побудована за схемою, подібною до YOLOv5, але замість класичного CSPLayer застосовує модуль C2f. Цей модуль, розроблений для покращення передачі характеристик між шарами, поєднує високорівневі ознаки з контекстною інформацією, що сприяє точному виявленню об'єктів, зокрема дрібних, таких як комахи-шкідники у зерносховищах.

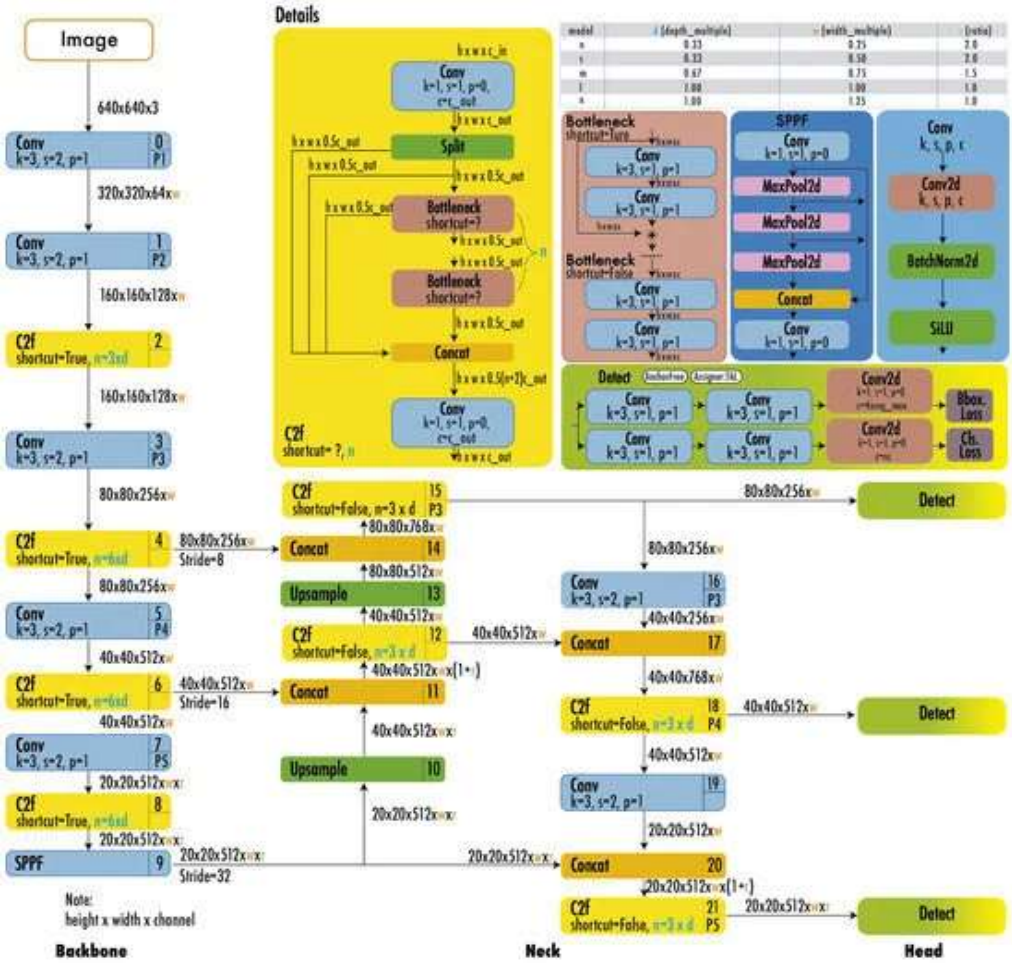


Рисунок 2.1 – Архітектура YOLOv8m [47]

YOLOv8 використовує вдосконалений варіант CSPDarknet53 як основу (backbone), яка раніше була застосована у YOLOv4 і модифікована для потреб YOLOv8. CSPDarknet53 базується на структурі Cross-Stage Partial (CSP), що поділяє потік характеристик на окремі частини, знижуючи обчислювальні витрати та підвищуючи ефективність навчання за рахунок зменшення навантаження на пам’ять. Це дозволяє архітектурі одночасно зберігати точність при мінімізації затримок, що є важливим для завдань реального часу в умовах зерносховищ.

У YOLOv8 модуль C2f оптимізований для зменшення обчислювальних витрат і покращення ефективності передачі інформації між рівнями. Використання двох згорток у модулі C2f забезпечує інтеграцію локальних ознак із більш загальною контекстуальною інформацією, що дозволяє моделі ефективно розрізняти об'єкти на складному фоні та в умовах високої щільності, як це характерно для зерносховищ. Це поєднання забезпечує високу швидкість обробки без втрати якості розпізнавання, що робить YOLOv8 надзвичайно ефективним інструментом для задач виявлення шкідників у зерні.

Для оптимізації швидкості обчислень у YOLOv8 інтегровано шар Spatial Pyramid Pooling Fast (SPPF). Цей шар призначений для збору ознак з різних частин вхідного зображення, генеруючи фіксовану карту ознак незалежно від початкового розміру зображення. Такий підхід дозволяє зберегти просторову ієрархію ознак, що покращує точність виявлення об'єктів на зображеннях різних масштабів, особливо актуально для моніторингу зерносховищ, де необхідно розпізнавати як дрібні, так і крупні об'єкти.

Кожен згортковий шар в YOLOv8 оснащено нормалізацією за партіями та активацією SiLU (Sigmoid-weighted Linear Unit). Нормалізація за партіями забезпечує стабільність під час навчання, нормалізуючи розподіл вхідних даних для кожного шару, а активація SiLU надає мережі необхідну нелінійність для розпізнавання складних структур у даних. Це дозволяє моделі ефективно розрізняти різноманітні особливості об'єктів, що є важливим при виявленні та класифікації шкідників серед маси зерна.

Головний блок YOLOv8 поділено на окремі частини для обробки різних задач: детектування об'єкта, класифікації та регресії координат. Такий підхід дозволяє спеціалізувати кожну частину мережі на конкретне завдання, що підвищує її загальну точність і ефективність. Завдяки незалежній обробці завдань кожна частина мережі може бути налаштована на оптимальну роботу для свого типу обчислень, що підвищує точність розпізнавання шкідників у складних умовах зерносховищ.

Удосконалення кожного компонента, від структури шарів до кінцевих блоків обробки, дозволяють YOLOv8 досягати високих показників точності та швидкості, що робить її ідеальною для застосувань у промислових умовах.

Однак при застосуванні для класифікації дрібних об'єктів, таких як комахи-шкідники, YOLOv8 стикається з певними труднощами. Основні обмеження архітектури пов'язані з тим, що невеликі або частково перекриті об'єкти можуть залишатися нерозпізнаними у складному середовищі, наприклад, серед зернових мас. Це може зумовити помилкові класифікації або пропуски при виявленні комах, особливо якщо об'єкти подібні до фону за розміром або кольором.

2.2 Вдосконалення архітектури нейромережі для класифікації комах

Для підвищення якості нейромережі в задачах класифікації комах-шкідників необхідно впровадити низку вдосконалень, спрямованих на підвищення точності та стабільності моделі. Основним підходом є донавчання моделі на специфічних наборах даних, що відображають різноманітні види комах. Для запобігання перенавчанню запропоновано механізми контролю, які дозволять автоматично відслідковувати та коригувати процес навчання. Окрім того, за допомогою оптимізації гіперпараметрів здійснюватиметься пошук найбільш продуктивних конфігурацій, що забезпечить максимальну продуктивність моделі в умовах реального середовища.

Отже, для адаптації YOLOv8 під специфічні вимоги до класифікації шкідників у зерносховищах запропоновано наступні зміни до базової моделі:

– *донавчання моделі на спеціалізованих наборах даних*. Для точного розпізнавання комах модель потрібно донавчити на спеціалізованому, для цієї мети, датасеті, що включає зображення шкідників у зернових масах. Такий підхід сприяє підвищенню точності класифікації, дозволяючи моделі адаптуватися до специфічних особливостей структури та кольору шкідників у складних фонових умовах;

– *додавання шарів Dropout для регуляризації.* Використання шарів Dropout у архітектурі сприятиме запобіганню перенавчанню, випадковим чином відключаючи нейрони під час навчання, що знижує ймовірність перенавчання на специфічних даних і дозволяє моделі краще узагальнювати інформацію. Для додаткового контролю за перенавчанням запропоновано зберігати метрики навчання і забезпечити автоматичний вибір найкращої моделі на основі показника fitness. Це дозволяє своєчасно втручатися у навчальний процес у разі появи ознак перенавчання, забезпечуючи точніше розпізнавання комах у складних середовищах;

– *оптимізація гіперпараметрів*. За допомогою оптимального вибору гіперпараметрів, таких як швидкість навчання, розмір батчу та тип оптимізатора, можливо знайти найбільш результативну комбінацію параметрів для досягнення найкращих показників точності та стабільності. Це підвищує продуктивність YOLOv8, роблячи її більш адаптованою для виявлення дрібних об'єктів у складних умовах, що особливо важливо для завдань з класифікації шкідників у зернових масах.

Завдяки цим вдосконаленням модель буде краще підготовлена для виконання завдань з виявлення комах-шкідників у зерносховищах, забезпечуючи високу точність та стабільність роботи навіть у складних умовах. Запропоноване вдосконалення архітектури та налаштування гіперпараметрів дозволяють більш результативно використовувати модель в умовах реального світу, забезпечуючи високий рівень якості розпізнавання дрібних шкідників навіть на тлі зерна та інших елементів середовища.

Далі розглянемо детальніше запропоновані вдосконалення.

2.2.1. Донавчання моделі глибокого навчання

Донавчання моделі є необхідним етапом для адаптації попередньо навченої моделі до специфічної задачі класифікації. У задачі класифікації комах-шкідників у зерносховищах донавчання YOLOv8 дозволяє використовувати вже наявні знання моделі для розпізнавання нових класів об'єктів, забезпечуючи більш точні результати.

Нехай попередньо навчена модель з параметрами $\theta_{pretrained}$, отриманими шляхом навчання на великому датасеті $D_{pretrained}$. Ці параметри включають ваги та зміщення кожного шару нейронної мережі, що дозволяють моделі виконувати базові задачі класифікації.

Для адаптації моделі до нового завдання – класифікації комах-шкідників – використовується новий датасет:

$$D_{new} = \{(x_i, y_i)\}_{i=1}^{N}, \tag{2.1}$$

де x_i є вхідним зображенням, а y_i – відповідним класом комахи-шкідника. Завдання полягає в оптимізації параметрів моделі з метою мінімізації втрат на новому датасеті, що дозволить підвищити її точність у специфічних умовах.

Для задачі класифікації застосовується крос-ентропійна функція втрат, яка оцінює розбіжність між прогнозованими ймовірностями та фактичними класами:

$$L(\theta) = \frac{1}{N}\sum_{i=0}^{n}\sum_{k=0}^{K} y_{i,k}\log(p_{i,k}) \tag{2.2}$$

де N – кількість прикладів у датасеті, K – кількість класів, $y_{i,k}$ – індикатор належності прикладу i до класу k (1, якщо належить, і 0 – інакше), а $p_{i,k}$ – ймовірність того, що модель з параметрами θ класифікує приклад ii як клас k.

Для оновлення параметрів застосовуються алгоритми стохастичного градієнтного спуску (SGD). Оновлення параметрів відбувається за наступним правилом:

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta L_{reg}(\theta_t), \tag{2.3}$$

де:

θ_t – параметри на ітерації t;

η – швидкість навчання;

$\nabla_\theta L_{reg}(\theta_t)$ – градієнт регуляризованої функції втрат відносно параметрів.

Стохастичний градієнтний спуск є методом оптимізації для знаходження оптимальних значень цільової функції. Основною метою використання SGD є визначення таких параметрів моделі, які забезпечать високу точність як для навчальних, так і для тестових даних. У цьому методі градієнт є вектором напрямку, що вказує на шлях максимального зменшення функції втрат у певній точці. На відміну від класичного градієнтного спуску, який здійснює багаторазові обчислення градієнтів для подібних прикладів у великих наборах даних, стохастичний градієнтний спуск уникає цих надлишкових обчислень, оновлюючи параметри моделі після кожного кроку. Це дозволяє значно прискорити процес оптимізації, що обумовило його вибір для даного дослідження.

Алгоритм SGD переміщується у просторі параметрів у напрямку найбільшого нахилу функції втрат, що дозволяє ефективно наближатися до оптимальних значень. На рис. 2.2 показано приклад роботи SGD. У процесі стохастичного градієнтного спуску одним із найважливіших параметрів є швидкість навчання. Якщо вона надто висока, алгоритм може «перестрибувати» вузькі улоговини мінімумів, а якщо занадто низька – алгоритм може застрягти в одному з локальних мінімумів.

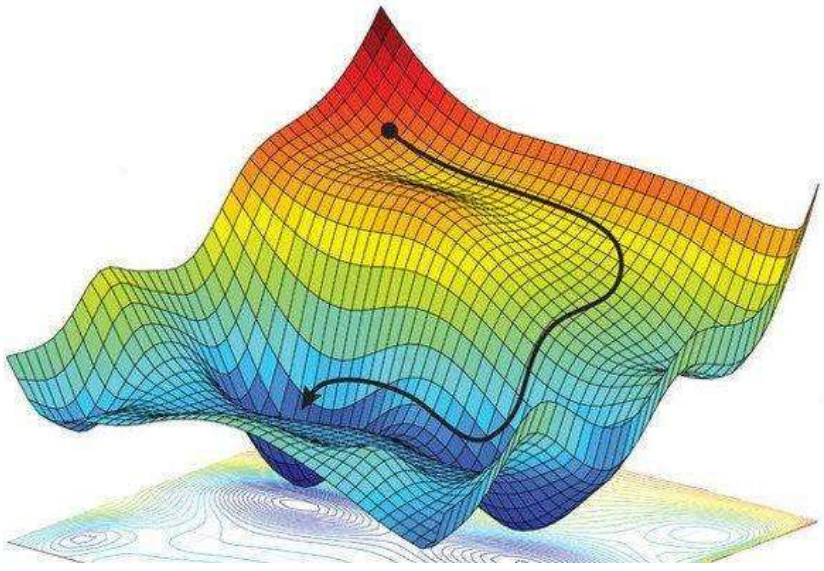


Рисунок 2.2 – Стохастичний градієнтного спуску [48]

Для навчання нейронної мережі використовуються відповідні набори даних, які створені самостійно. Мережа переглядає набір даних неодноразово: кожне повне опрацювання навчальної вибірки називається епохою. Кожне проходження набору даних мережею є ітерацією, і кількість епох задається вручну на етапі початкового налаштування.

При донавчанні використовується концепція передачі навчання. Параметри нижніх шарів мережі, які відповідають за витяг характеристик, залишаються фіксованими або оновлюються з меншою швидкістю навчання, тоді як верхні шари, що відповідають за класифікацію, оновлюються повністю.

Позначимо $\theta = [Q_{base}, Q_{head}]$, де Q_{base} – параметри базової частини моделі, а Q_{head} – параметри класифікатора. Тоді оновлення параметрів можна записати наступним чином:

$$\theta_{base}^{t+1} = \theta_{base}^{t} - \eta_{base} \nabla_{\theta_{base}} L_{reg}(\theta_t), \tag{2.4}$$

$$\theta_{head}^{t+1} = \theta_{head}^{t} - \eta_{head} \nabla_{\theta_{head}} L_{reg}(\theta_t), \tag{2.5}$$

де $\eta_{base} < \eta_{head}$ або $\eta_{base} = 0$ (якщо базові шари не оновлюються).

Для покращення здатності моделі до узагальнення застосовується метод Dropout, який випадковим чином «відключає» нейрони під час навчання з певною ймовірністю p, що дозволяє зменшити ризик перенавчання.

Математично, вихід кожного нейрона під час навчання масштабується за випадковою змінною $z \sim Bernoulli(1-p)$:

$$h_i^{(dropout)} = z_i h_i \tag{2.6}$$

де h_i – вихід нейрона без Dropout, а z_i – випадкова змінна, що дорівнює 0 з ймовірністю p і 1 з ймовірністю $1-p$. Під час оцінювання (inference) Dropout не застосовується, проте виходи нейронів масштабується, щоб компенсувати вплив Dropout у процесі навчання:

$$h_i^{(eval)} = (1-p)\, h_i \tag{2.7}$$

Оскільки завдання полягає в класифікації комах-шкідників, які можуть не бути в початковому датасеті, необхідно адаптувати вихідний шар моделі до нової кількості класів K_{new}.

Якщо початкова модель мала K_{old} вихідних нейронів, то останній шар замінюється на новий, який відповідає K_{new} класам. Також, якщо останній повнозв'язний шар моделі має параметри $W_{old} \in R^{K_{old} \cdot d}$ та $b_{old} \in R^{K_{old}}$, де d – розмірність вхідного вектора. Новий шар у свою чергу визначається за параметрами: $W_{new} \in R^{K_{new} \cdot d}$ а $b_{new} \in R^{K_{new}}$.

Якщо датасет є незбалансованим (наприклад, різна кількість прикладів для кожного класу), застосовується зважена функція втрат:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{n} w_{y_i} \sum_{k=1}^{K} y_{i,k} \log(p_{i,k}) , \quad (2.8)$$

де w_{y_i} – ваги класів, обернено пропорційні кількості прикладів у класі y_i. Це дозволяє моделі приділяти більше уваги класам із меншою кількістю даних, що підвищує її здатність до узагальнення.

Для пришвидшення збіжності використовується метод імпульсу (Momentum):

$$v_{t+1} = \mu v_t - \eta \nabla_\theta L_{reg}(\theta_t) , \quad (2.9)$$

$$\theta_{t+1} = \theta_t + v_{t+1} , \quad (2.10)$$

де v_t – накопичений градієнт, а μ – коефіцієнт імпульсу. Цей підхід допомагає стабілізувати процес навчання, зменшуючи коливання градієнтів і пришвидшуючи пошук оптимального рішення.

Отже, донавчання моделі YOLOv8 для класифікації комах-шкідників у зерносховищах включає в себе адаптацію попередньо навченої моделі до створеного датасету шляхом оновлення її параметрів з використанням оптимізаційних алгоритмів та регуляризаційних технік. Математичний апарат, що лежить в основі цього процесу, базується на мінімізації функції втрат відносно параметрів моделі з використанням методів градієнтного спуску та врахуванням особливостей даних і задачі.

2.2.2. Контроль за перенавчанням

Перенавчання (overfitting) є однією з найбільш поширених проблем при побудові моделей глибокого навчання, зокрема для завдань класифікації зображень, як у випадку класифікації комах-шкідників у зерносховищах. При перенавчанні модель надмірно адаптується до навчальних даних, втрачаючи здатність до узагальнення на нових даних. Це призводить до високої точності на навчальних даних і низької точності на тестових.

Для контроля за перенавчанням в даній роботі було реалізовано декілька методів, зокрема через dropout та використання weight decay.

Термін «dropout» позначає процес випадкового вимкнення нейронів (входів та прихованих шарів) у нейронній мережі. При цьому всі прямі та зворотні зв’язки з вимкненими нейронами тимчасово видаляються, що дозволяє формувати нову архітектуру на основі початкової мережі. Імовірність вимкнення нейронів визначається параметром *p*, який називається ймовірністю dropout.

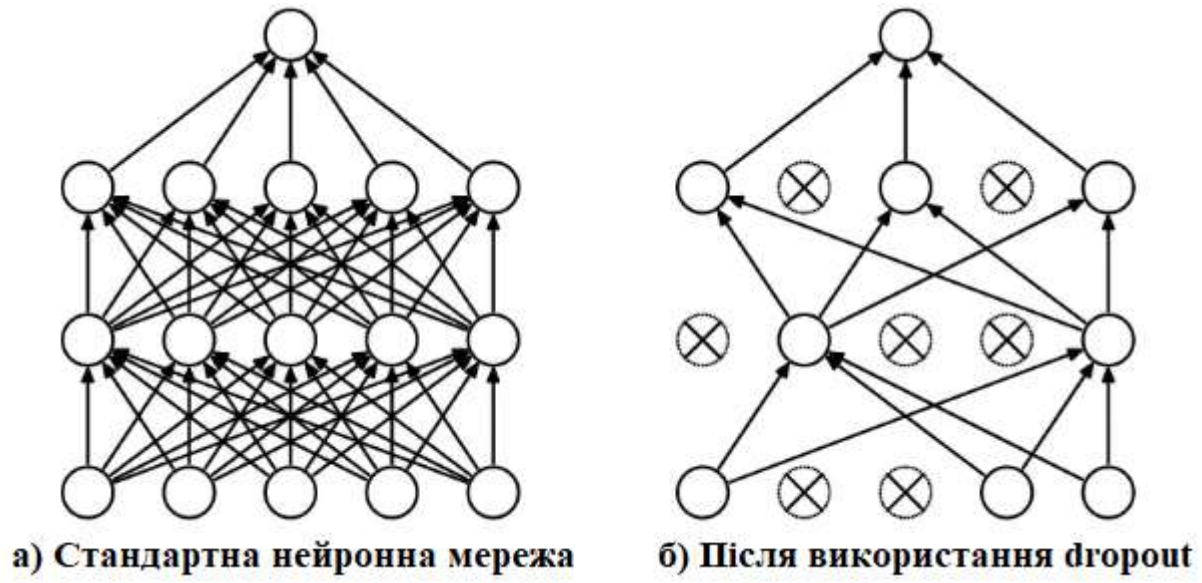


Рисунок 2.3 – Використання dropout для стандартної нейронної мережі [49]

Розглянемо приклад із вхідним вектором x={1,2,3,4,5}, що подається на повнозв’язний шар з налаштованим шаром dropout, де ймовірність вимкнення становить p=0.2p=0.2 (або ймовірність збереження нейрона – 0.8). Під час прямого

поширення (forward propagation) у процесі навчання з вхідного вектора xx буде вимкнено 20% нейронів, тож, наприклад, xx може набути вигляду {1,0,3,4,5}{1,0,3,4,5} або {1,2,0,4,5}{1,2,0,4,5}. Аналогічний підхід застосовується і до прихованих шарів.

Для прикладу, якщо приховані шари містять 1000 нейронів, і застосовано dropout з ймовірністю вимкнення 0.5, то у кожній ітерації навчання випадковим чином буде вимкнено 500 нейронів. У загальному випадку для вхідних шарів зазвичай рекомендується зберігати ймовірність утримання нейронів ближче до 1, тобто 1−p≈0.81−p≈0.8, що є оптимальним значенням відповідно до рекомендацій авторів методу. Для прихованих шарів із більшою ймовірністю вимкнення модель стає більш розрідженою; оптимальним значенням для них є p=0.5p=0.5, що передбачає вимкнення 50% нейронів, створюючи таким чином компроміс між збереженням інформації та запобіганням перенавчанню.

У проблемі перенавчання модель схильна до вивчення статистичного шуму, що заважає їй узагальнювати на нових даних. Основна мета навчання моделі полягає в мінімізації функції втрат, враховуючи всі елементи (нейрони). Однак у випадку перенавчання окремі нейрони можуть адаптуватися таким чином, що вони компенсують помилки інших нейронів. Це створює складні взаємозалежності, що ускладнюють здатність моделі до узагальнення, оскільки така складна співадаптація зазвичай не спрацьовує на невідомих даних.

Застосування Dropout допомагає запобігти цим співадаптаціям, оскільки кожен нейрон у процесі навчання може бути вимкнений із певною ймовірністю, роблячи його присутність непередбачуваною. Таким чином, випадкове вимкнення частини нейронів примушує інші шари брати на себе більшу відповідальність за обробку вхідних даних, використовуючи ймовірнісний підхід.

Цей метод сприяє узагальненню моделі, знижуючи ризик перенавчання. На прикладі видно (рис. 2.4), що прихований шар із застосуванням Dropout здатен вивчати загальні риси даних, уникнувши створення співадаптацій, які характерні для шару без Dropout. Очевидно, що Dropout руйнує взаємозв’язки між нейронами, тим самим сприяючи загальному узагальненню моделі.

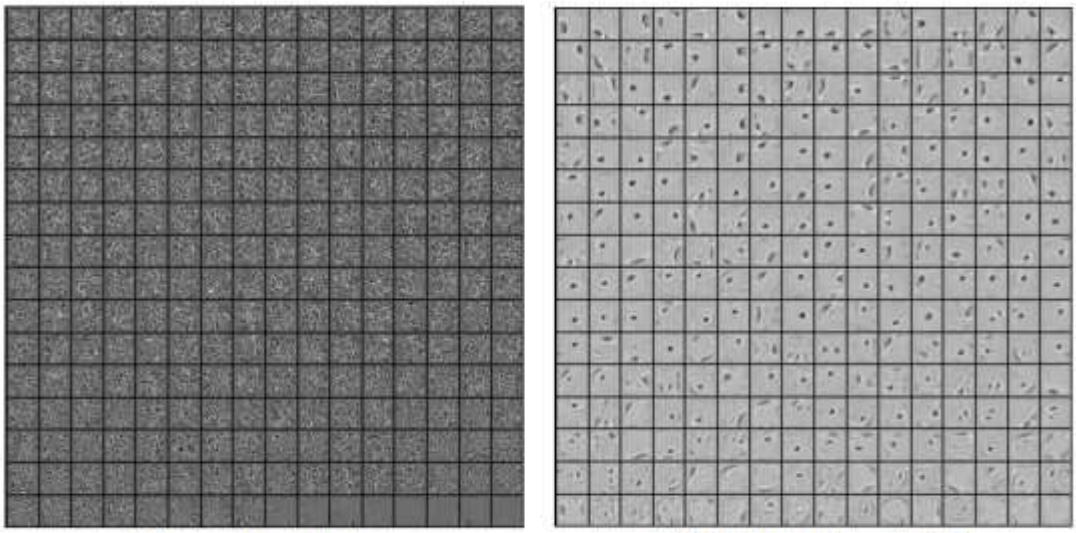


Рисунок 2.4 – Функції прихованого шару з та без dropout [49]

З аналізу на рис. 2.4 чітко видно, що прихований шар із використанням Dropout здатен навчатися більш узагальненим ознакам, ніж шар без Dropout, який схильний до формування співадаптацій. Dropout руйнує такі внутрішньошарові зв'язки між нейронами, що дозволяє моделі зосередитися на загальних ознаках, сприяючи її здатності до узагальнення.

У класичній реалізації шару Dropout, під час навчання кожен нейрон у шарі зберігається з певною ймовірністю, яка дорівнює $1-p$, де p – це ймовірність вимкнення нейрона. Це створює розріджену архітектуру для кожного конкретного навчального бетчу, причому ця архітектура змінюється при кожній ітерації.

У стандартній нейронній мережі, під час прямого поширення сигналу використовуються такі рівняння:

$$z_i^{(l+1)} = w_i^{(l+1)} y^l + b_i^{(l+1)} , \tag{2.11}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)}) , \tag{2.12}$$

де:

z – вектор виходу зі шару (l+1) до активації;

y – вектор виходів зі шару l;

w – вага шару l;

b – зміщення (bias) шару l.

За допомогою функції активації значення z перетворюються у вихідні значення для шару (l+1). Це забезпечує необхідне нелінійне перетворення, що сприяє навчальній здатності моделі. Dropout додає варіативність у ці обчислення, забезпечуючи підвищену стійкість моделі до перенавчання.

При використанні Dropout рівняння прямого поширення модифікуються наступним чином:

$$r_j^{(l)} \sim Bernoulli(p) \ , \tag{2.13}$$

$$\tilde{y}^{(l)} = r^{(l)} \cdot y^{(l)} \ , \tag{2.14}$$

$$z_i^{(l+1)} = w_i^{(l+1)} \tilde{y}^{(l)} + b_i^{(l+1)} \ , \tag{2.15}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)}) \ , \tag{2.16}$$

де $r_j^{(l)}$ – випадкова величина, яка підпорядковується розподілу Бернуллі з ймовірністю p набувати значення 1. Перед обчисленням zz вхідні значення шару проходять процес випадкового відбору, при якому елементно множаться на незалежні випадкові змінні Бернуллі. У цьому випадку r виконує роль маски для вхідного вектора $y^{(l)}$, забезпечуючи збереження лише обраних нейронів відповідно до ймовірності збереження Dropout.

Така операція дозволяє формувати "розріджені" вихідні значення $\tilde{y}^{(l)}$, які подаються на вхід наступного шару під час прямого поширення. Цей підхід забезпечує ефективне усереднення внеску різних нейронів, що сприяє загальному узагальненню моделі та знижує ризик перенавчання.

Окрім контролю за перенавчанням за допомогою Dropout, було використано регуляризацію шляхом зменшення ваг (weight decay), яка знижує складність моделі та запобігає надмірній адаптації ваг до специфічних даних навчального набору. Dropout дозволяє формувати "розріджені" вихідні значення $\tilde{y}^{(l)}$, що подаються на вхід наступного шару під час прямого поширення, сприяючи узагальненню моделі. Таке поєднання Dropout та weight decay сприяє ефективному усередненню внеску різних

нейронів, що знижує ризик перенавчання і допомагає уникнути надмірної залежності між нейронами.

Для теоретичного обґрунтування використаного методу продемонстровано, що пошарове зменшення ваг є інваріантним щодо пошарового масштабування мережі. Масштабування, або репараметризація, дозволяє змінювати ваги з'єднань без впливу на обчислювану функцію, завдяки чому можна зберегти функціональність мережі. Це масштабування легко вивести при використанні певних функцій активації *f*, таких як лінійна функція або ReLU, які задовольняють властивість $f(ax) = af(x)$ для будь-якого додатного скаляра a.

Як показано на рис. 2.5, один із найпростіших прикладів пошарового масштабування реалізується шляхом зміни ваг W_{l-1} на aW_{l-1} і ваг W_l на $\frac{1}{a}W_l$; у такому випадку значення x_{l+1} залишається незмінним. Оскільки масштабування не впливає на функціональність мережі, воно не повинно впливати на оновлення мережі в процесі навчання. Якщо ж вплив від масштабування відчутний, це свідчить про те, що спосіб оновлення мережі залежить від масштабу кожної ваги, що може призвести до нестабільності процесу навчання.

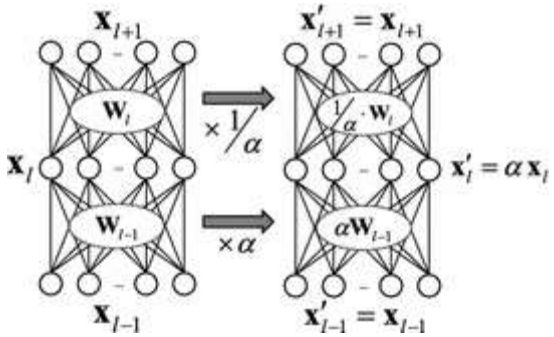


Рисунок 2.5 – Функції прихованого шару з та без dropout [50]

Доведено, що сила регуляризації змінюється при застосуванні постійного зменшення ваг, але залишається інваріантною при використанні пошарового зменшення ваг. Слід зазначити, що у цьому дослідженні акцент зроблено на балансі між градієнтом і регуляризацією ваг, на відміну від існуючих робіт, які зосереджуються лише на масштабі градієнта.

Припустімо, що мережа масштабована, як показано на рис. 2.5. Оскільки будь-яке пошарове масштабування може бути відображене через комбінацію такого типу масштабування, вивчення цього випадку є достатнім для демонстрації інваріантності методу до масштабування. Позначимо x_l' і W_l' як вхідні значення і ваги зв'язків ll-го шару після масштабування відповідно. У цьому контексті можна вивести вирази для $\Delta W_{l-1}'$ та $\Delta\Delta W_l'$:

$$\Delta W_{l-1}' = x_{l-1}'^{T}\Delta y_{l-1}' = x_{l-1}^{T}\left(\frac{1}{a}\right)\Delta y_{l-1}' = \frac{1}{a}\,\Delta W_{l-1}\,, \tag{2.17}$$

$$\Delta W_l' = x_l'^{T}\Delta y_l' = a x_l^{T}\Delta y_l = a\Delta W_l\,. \tag{2.18}$$

Підставляючи ці рівняння у формулу оновлення ваг:

$$W_l^{t+1} - \eta\left(\frac{\partial E}{\partial W_l}|W_l^t + \lambda W_l^t\right), \tag{2.19}$$

де W_l^t описує ваги зв'язків у l-му шарі (де l=1,…,L) мережі після t-го оновлення, η – швидкість навчання, E – цільова функція, яку потрібно мінімізувати (наприклад, функція втрат із багатокласовою логістичною регресією), а λ – коефіцієнт weight decay, що відображає силу регуляризації для W_l^t. Це значення використовується для обчислення прямого поширення і зворотного поширення помилки в l-му шарі.

Оновлення ваг масштабованої мережі при постійній величині weight decay матиме вигляд:

$$\Delta W_{l-1}' + \lambda W_{l-1}' = \frac{1}{a}\Delta W_{l-1} + \lambda a W_{l-1}, \tag{2.20}$$

$$\Delta W_l' + \lambda W_l = a W_l + \frac{\lambda}{a} W_l. \tag{2.21}$$

За великих значень аа регуляризаційний член weight decay домінує в оновленні W_{l-1}, тоді як у випадку W_l градієнтний член стає домінуючим. Це свідчить про те, що масштабування впливає на силу weight decay.

На відміну від цього, при пошаровому зменшенні ваг (layer-wise weight decay) значення коефіцієнтів регуляризації змінюватимуться залежно від масштабування за такими виразами:

$$\lambda'_{l-1} = \frac{scale(\Delta W'_{l-1})}{scale(W'_{l-1})}\lambda = \frac{\frac{1}{a}scale(\Delta W_{l-1})}{a\ scale(W_{l-1})} = \frac{\lambda_{l-1}}{a^2}, \quad (2.22)$$

$$\lambda'_l = \frac{scale(\Delta W'_l)}{scale(W'_l)}\lambda = \frac{a\ scale(\Delta W_l)}{\frac{1}{a}scale(W_l)} = a^2\lambda_l. \quad (2.23)$$

На основі наведених виразів можна вивести значення оновлень для масштабованої мережі у випадку пошарового зменшення ваг:

$$\Delta W'_{l-1} + \lambda'_{l-1}W'_{l-1} = \frac{1}{a}\Delta W_{l-1} + \frac{\lambda_{l-1}}{a^2}aW_{l-1} = \frac{1}{a}(W_{l-1} + \lambda_{l-1}W_{l-1}), \quad (2.24)$$

$$\Delta W'_l + \lambda'_l W'_l = a\Delta W_l + \frac{a^2\,\lambda_l}{a}W_l = a(\Delta W_l + \lambda_l W_l). \quad (2.25)$$

Рівняння показують, що сила регуляризації шляхом зменшення ваг є інваріантною до масштабування в межах запропонованого методу. Така інваріантність випливає з належного урахування масштабів градієнтів, що залежать від масштабів ваг у верхніх шарах. Хоча масштаб оновлення можна компенсувати адаптивною швидкістю навчання або іншими складними методами оптимізації, можна зробити висновок, що пошарове зменшення ваг забезпечує інваріантність навчання мережі відносно масштабування.

Таким чином, застосовані методи контролю за перенавчанням, включаючи Dropout і пошарове зменшення ваг, дозволяють значно підвищити здатність моделі до узагальнення. Dropout сприяє уникненню співадаптацій між нейронами, тоді як пошарове зменшення ваг забезпечує стійкість регуляризації до масштабування. Поєднання цих підходів сприяє стабільному і надійному навчанню моделі, ефективно запобігаючи перенавчанню і зберігаючи високий рівень узагальнення на невідомих даних.

2.2.3. Оптимізація гіперпараметрів

Оптимізація гіперпараметрів є одним із ключових етапів процесу навчання глибоких нейронних мереж. Гіперпараметри визначають як структуру моделі, так і її навчальні характеристики, впливаючи на збіжність алгоритму та здатність моделі до узагальнення. У контексті класифікації комах-шкідників у зерносховищах із

використанням моделі YOLOv8 правильний вибір гіперпараметрів є вирішальним для досягнення високих показників точності та стабільності.

У цьому дослідженні буде використовуватись оптимізація таких гіперпараметрів, як:

– швидкість навчання (η);

– розмір міні-бетчу (B);

– оптимізатор.

Ці параметри мають суттєвий вплив на процес навчання моделі та якість отриманих результатів.

Швидкість навчання є гіперпараметром, який визначає розмір кроку, з яким ваги моделі оновлюються під час кожної ітерації навчання. Правило оновлення вагів при використанні стохастичного градієнтного спуску формулюється таким чином:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L_{reg}(\theta_t)\,, \tag{2.26}$$

де:

θ_t – значення параметрів моделі на ітерації t;

$\nabla_\theta L_{reg}(\theta_t)$ – градієнт регуляризованої функції втрат на ітерації t.

Вибір оптимального значення швидкості навчання є надзвичайно важливим для успішного навчання моделі. Якщо значення η занадто велике, модель може стати нестабільною, оскільки кожен крок оновлення буде надто великим, що призведе до пропускання мінімуму функції втрат. У такому випадку алгоритм збіжності не досягне. З іншого боку, надто мале значення η значно уповільнює процес навчання і може спричинити застрягання в локальному мінімумі, обмежуючи здатність моделі досягти глобального мінімуму.

У рамках цього дослідження розглядаються значення $\eta \in \{0.001, 0.0001\}$, які обираються з урахуванням необхідності забезпечення стабільної збіжності та запобігання коливанням градієнта під час навчання.

Для кожної комбінації гіперпараметрів у процесі дослідження виконуватимуться такі кроки:

– ініціалізація моделі: завантажується попередньо навчена модель YOLOv8, що містить початкові параметри, здатні забезпечити базовий рівень точності;

– налаштування гіперпараметрів: встановлюються значення швидкості навчання η, розміру міні-бетчу B та вибраного оптимізатора. Значення цих гіперпараметрів визначають ефективність і стабільність процесу навчання моделі;

– навчання моделі: проводиться тренування моделі на навчальному наборі даних протягом заданої кількості епох, що дозволяє моделі пристосувати свої параметри до специфічних характеристик завдання;

– оцінка моделі: обчислюються метрики на валідаційному наборі для оцінки якості поточної моделі з обраними гіперпараметрами. Цей етап дає змогу виявити, наскільки модель здатна узагальнювати нові дані;

– збереження метрик: результати навчання фіксуються для подальшого аналізу, що дозволяє порівняти різні комбінації гіперпараметрів і вибрати найкращу;

– оновлення найкращої моделі: якщо поточна модель демонструє вищі показники за обраним критерієм, вона зберігається як новий найкращий варіант.

Після тренування моделей з усіма можливими комбінаціями гіперпараметрів найкраща модель обирається на основі значення метрики «fitness».

Нехай $Fitness_k$ позначає значення метрики «fitness» для моделі, яка використовує комбінацію гіперпараметрів k. Тоді обирається комбінація з максимальним значенням за формулою:

$$k^* = \arg max_k \, Fitness_k \,, \tag{2.27}$$

де k^* – індекс комбінації, що забезпечує найвищу продуктивність.

Модель із комбінацією гіперпараметрів k^* зберігається для подальшого використання, що забезпечує оптимальну ефективність на досліджуваному датасеті.

При великих значеннях η модель може стати нестабільною, оскільки надмірно великі кроки в оновленні параметрів призводять до того, що модель пропускає як локальні, так і глобальні мінімуми функції втрат. В результаті модель не може досягти оптимальної точки мінімізації. Математично, якщо крок оновлення занадто великий, це можна виразити так:

$$||\theta_{t+1} - \theta_t|| = \eta||\nabla_\theta L_B(\theta_t) \gg 0\,, \tag{2.28}$$

де θ_{t+1} і θ_t позначають параметри моделі на ітераціях t+1 і t відповідно, а $\nabla_\theta L_B(\theta_t)$ – градієнт функції втрат на ітерації t. Високе значення швидкості навчання збільшує відстань між значеннями параметрів на сусідніх ітераціях, що створює ризик для стабільності алгоритму.

Якщо значення η занадто мале, процес навчання буде дуже повільним, оскільки кроки оновлення параметрів стають мізерно малими. Це призводить до того, що модель може застрягнути в локальному мінімумі, оскільки вона не матиме достатньо «імпульсу» для подолання локальних мінімумів функції втрат. Математично, маленький крок можна відобразити так:

$$||\theta_{t+1} - \theta_t|| = \eta||\nabla_\theta L_B(\theta_t) \approx 0\,, \tag{2.29}$$

що свідчить про те, що модель практично не змінює свої параметри між ітераціями, що значно уповільнює її збіжність до глобального мінімуму.

Розмір міні-бетчу B визначає кількість зразків, які обробляються моделлю за одну ітерацію, перед оновленням параметрів. Цей параметр впливає на стабільність градієнта та потреби в пам’яті, а також може допомогти уникнути локальних мінімумів.

При малому значенні B градієнт функції втрат має вищу варіативність, що означає більшу дисперсію між значеннями градієнтів у різних ітераціях. Така варіативність градієнта може допомогти моделі уникнути локальних мінімумів і сприяти кращому узагальненню результатів. Математично це виражається так:

$$Var[\nabla_\theta L_B(\theta_t)] = \frac{\sigma^2}{B}\,, \tag{2.30}$$

де σ^2 – дисперсія градієнта по всьому датасету. При зменшенні B значення дисперсії зростає, що підвищує варіативність градієнта.

Велике значення B забезпечує стабільніший градієнт, що робить процес навчання менш мінливим. Це знижує ймовірність коливань у процесі навчання та дозволяє моделі робити більш обґрунтовані кроки до мінімізації функції втрат. Однак

збільшення розміру міні-бетчу вимагає значно більше пам'яті, що може стати обмеженням при роботі з великими моделями або обмеженими ресурсами.

Оптимізація гіперпараметрів є ключовим етапом у досягненні високої продуктивності моделей глибокого навчання, оскільки правильний вибір таких параметрів, як швидкість навчання, розмір міні-бетчу та оптимізатор, суттєво впливає на здатність моделі навчатися ефективно та узагальнювати на нових даних. У цьому дослідженні продемонстровано процес оптимізації гіперпараметрів за допомогою методу повного перебору, а також проаналізовано вплив кожного гіперпараметра на результати навчання моделі. Цей підхід дозволяє покращити якість класифікації комах-шкідників та забезпечити надійність моделі в реальних умовах використання, створюючи умови для її точного функціонування навіть за умов варіативності даних у реальному середовищі зерносховищ.

2.3 Опис метрик оцінки якості навченої моделі

Оцінка якості навченої моделі дозволяє зрозуміти, наскільки добре модель узагальнює закономірності з навчальних даних та як вона поводиться на нових, невідомих даних. У контексті класифікації комах-шкідників у зерносховищах, правильна оцінка моделі гарантує її ефективність та надійність у реальних умовах. Нижче описано метрики, використані для оцінки якості моделі.

Матриця помилок є інструментом для візуалізації кількості правильних і неправильних передбачень для кожного класу. Вона являє собою квадратну матрицю розміру $K \times K$, де K – загальна кількість класів. Кожен елемент матриці C_{ij} відображає кількість зразків, що належать класу ii, але були передбачені як клас j.

Математично елементи матриці помилок визначаються за наступною формулою:

$$C_{ij} = |\{x \in D | y = i \wedge \hat{y} = j\}|, \tag{2.31}$$

де:

D – тестовий набір даних,

y – істинна мітка класу,

$\hat{y}$ – передбачена мітка класу.

Матриця помилок дозволяє ідентифікувати, між якими класами модель найчастіше помиляється, що є корисним для виявлення слабких місць моделі. Наприклад, якщо модель часто плутає певні пари класів, це може свідчити про схожість між ознаками цих класів або про необхідність додаткового навчання моделі для розрізнення цих специфічних класів.

Accuracy є основною метрикою для оцінки загальної точності моделі класифікації, яка вказує на частку правильно класифікованих зразків від загальної кількості. Ця метрика обчислюється як відношення суми всіх правильних передбачень (True Positives і True Negatives) до загальної кількості передбачень і формується за наступною формулою:

$$Accuracy = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} (TP_k + FP_k + FN_k)}, \tag{2.32}$$

де K – кількість класів, TP_k – кількість істинно позитивних для класу k, FP_k – кількість хибнопозитивних для класу k, FN_k – кількість хибнонегативних для класу k.

Тобто, загальне значення Accuracy для багатокласової класифікації – це частка правильно класифікованих зразків серед загальної кількості зразків для всіх класів.

Precision (Точність) є метрикою, що оцінює кількість правильних передбачень серед усіх передбачень для конкретного класу, тобто, частку істинно позитивних результатів серед усіх позитивних передбачень. Ця метрика зосереджується на кількості хибнопозитивних передбачень, і її обчислення здійснюється за формулою:

$$\text{Precision}_k = \frac{TP_k}{TP_k + FP_k}, \tag{2.33}$$

де TP_k – істинно позитивні передбачення для класу k, а FP_k – хибнопозитивні передбачення для цього ж класу. *Precision* особливо важлива для задач, де критично важливо уникати помилкових тривог, як у випадках, коли хибнопозитивні передбачення можуть призводити до негативних наслідків або додаткових витрат. Зважене середнє значення *Precision* для усіх класів можна обчислити, враховуючи

кількість зразків у кожному класі, для забезпечення більш точної оцінки у задачах багатокласової класифікації.

Повнота для кожного класу *k* визначає, наскільки добре модель знаходить усі зразки цього класу, мінімізуючи пропущені виявлення. Іншими словами, повнота (*Recall*) показує частку правильних передбачень для класу серед усіх зразків, які фактично належать до цього класу. Математично повнота для класу *k* обчислюється за формулою:

$$\text{Recall}_k = \frac{TP_k}{TP_k + FN_k}, \tag{2.34}$$

де:

TP_k (True Positives) – кількість зразків класу *k*, які модель правильно розпізнала як цей клас;

FN_k (False Negatives) – кількість зразків класу *k*, які модель не розпізнала і передбачила як інший клас.

Зважена середня повнота, що обчислюється для усього набору даних, дозволяє оцінити загальну здатність моделі до класифікації, враховуючи частку кожного класу:

$$\text{Recall}_w = \frac{1}{N}\sum_{k=1}^{K} n_k \cdot \text{Recall}_k, \tag{2.35}$$

де *N* – загальна кількість зразків, а n_k – кількість зразків класу *k*.

F1-міра є гармонійним середнім між точністю та повнотою для кожного класу. Ця метрика враховує як помилкові тривоги (False Positives), так і пропущені виявлення (False Negatives), забезпечуючи збалансовану оцінку продуктивності моделі. F1-міра для класу *k* обчислюється за формулою:

$$\text{F1}_k = 2 \cdot \frac{Precision_k \cdot \text{Recall}_k}{Precision_k + \text{Recall}_k}, \tag{2.36}$$

де $Precision_k$ – точність для класу *k*, а Recall_k – повнота для класу *k*.

Зважена середня F1-міра обчислюється за допомогою:

$$\text{F1}_w = \frac{1}{N}\sum_{k=1}^{K} n_k \cdot \text{F1}_k, \tag{2.37}$$

що дозволяє врахувати різницю в кількості зразків кожного класу, забезпечуючи точніше оцінювання якості роботи моделі для задач класифікації.

Метрики точності, зокрема *Precision* та *Recall*, є ключовими для задачі класифікації комах-шкідників, оскільки вони дозволяють оцінити здатність моделі як мінімізувати помилкові тривоги, так і знижувати ризик пропусків. Високе значення *Precision* означає, що модель рідко помиляється при ідентифікації комах-шкідників, що є критичним для уникнення хибного позначення нешкідливих комах як шкідників. У свою чергу, високе значення *Recall* забезпечує повне охоплення всіх шкідників, мінімізуючи ймовірність пропусків важливих об’єктів. Для досягнення балансу між *Precision* і *Recall* використовується *F1-міра*, яка поєднує обидві метрики, забезпечуючи збалансовану оцінку продуктивності моделі. Високий рівень F1-міри свідчить про гармонійний розподіл між точністю та повнотою, що робить модель надійною для практичного використання. Аналіз цих метрик дозволяє визначити слабкі місця, на яких модель демонструє нижчу ефективність, та сприяє оптимізації гіперпараметрів для підвищення продуктивності. Крім того, ретельний огляд результатів за допомогою матриці помилок може підказати необхідність збору додаткових даних або застосування методів балансування датасету для покращення роботи моделі.

Висновки до розділу 2

У розділі було обґрунтовано теоретичні підходи для вдосконалення методу класифікації комах-шкідників за зображенням. Спершу детально розглянуто архітектуру YOLOv8, яка має відповідні переваги для задач виявлення дрібних об’єктів, а також наведено можливі обмеження, які необхідно враховувати для оптимізації продуктивності в специфічних умовах. У контексті дослідження обрано стратегію донавчання моделі YOLOv8, що включає адаптацію попередньо навчених параметрів до створеного датасету із зразками комах-шкідників, забезпечуючи моделі попередні знання про об’єкти нового класу.

Для зниження ризиків перенавчання передбачено застосування методів регуляризації, зокрема Dropout та weight decay, що сприятиме підвищенню стабільності моделі у подальших експериментах. Крім того, теоретично обґрунтовано вибір параметрів оптимізації, таких як швидкість навчання, розмір міні-бетчу та оптимізатор, для досягнення оптимального балансу між точністю та продуктивністю. Оцінка якості навченої моделі здійснюватиметься на основі метрик Accuracy, Precision, F1-міри й Recall, що дозволить комплексно оцінити результативність класифікації для кожного класу.

Отже, розроблені в цьому розділі теоретичні обґрунтування створюють основу для подальших експериментів, спрямованих на підвищення результативності моделі та адаптації її до умов використання у реальному середовищі.

Розділ 3

Основні кроки вдосконаленого методу для виявлення комах-шкідників у зерносховищах

У розділі наведено основні етапи вдосконаленого методу для виявлення комах-шкідників у зерносховищах, що базується на глибокому навчанні. Спочатку описано кроки для отримання вдосконаленої моделі на базі YOLOv8, включаючи обробку вхідних даних, додавання шарів регуляризації та оптимізацію гіперпараметрів, що підвищують точність і стабільність моделі. Далі наведено кроки для класифікації за вдосконаленою та навченою моделлю, які охоплюють підготовку до класифікації, перевірку коректності вхідних даних та обробку результатів. Кожна з описаних послідовностей кроків доповнена схемами для наочного розуміння послідовності дій. Завдяки цьому забезпечується структурований підхід до побудови методу, який здатний розпізнавати шкідників навіть у складних умовах зерносховищ.

3.1 Кроки для отримання вдосконаленої моделі глибокого навчання

Процес донавчання моделі YOLO8 для класифікації комах-шкідників у зерносховищах є необхідним для адаптації нейронної мережі до специфічних умов цієї задачі. Використання попередньо навченої моделі дає змогу скоротити час на навчання та покращити точність за рахунок використання вже сформованих базових характеристик. Проте через те, що базова модель не враховує унікальні особливості зображень шкідників у зернових масах, донавчання дозволяє пристосувати її параметри до нових класів об'єктів, підвищуючи якість розпізнавання.

Рис. 3.1 демонструє кроки процесу вдосконалення моделі YOLO8, включаючи етапи обробки даних, налаштування гіперпараметрів, і регуляризації, що забезпечують високу узагальнюючу здатність та стабільність моделі.

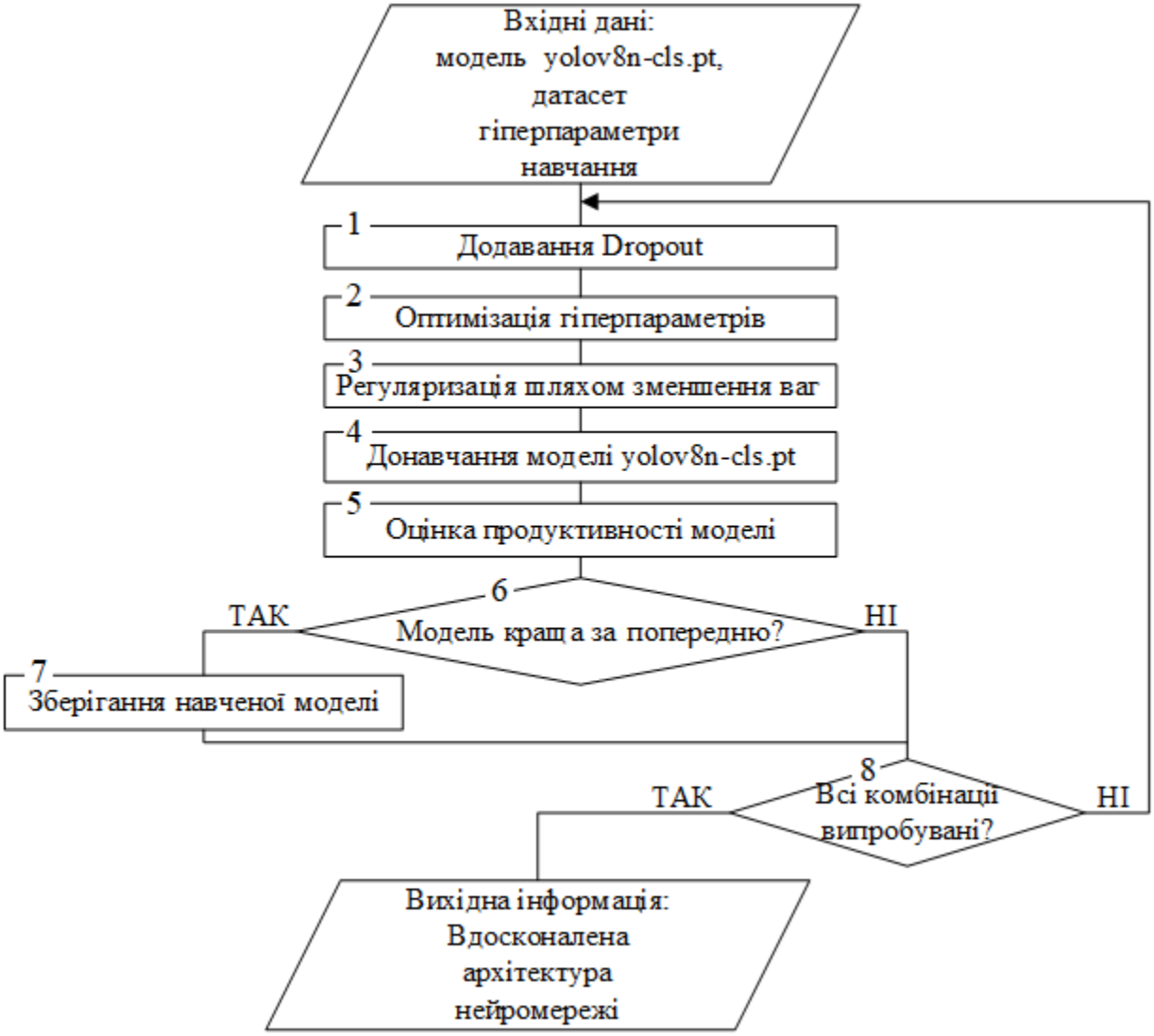


Рисунок 3.1 – Кроки для вдосконалення моделі

Процес навчання моделі вдосконаленим методом складається з наступних послідовних кроків:

Вхідна інформація: попередньо навчена модель YOLO (yolov8n-cls.pt), підготовлений датасет з даними для тренування та валідації, а також набір гіперпараметрів, які включають значення швидкості навчання, розміри батчу та оптимізатор.

Крок 1. Для зменшення ризику перенавчання додаються шари Dropout. Dropout випадково відключає певні нейрони під час навчання, що дозволяє моделі уникнути надмірної адаптації до тренувальних даних і покращує її здатність до узагальнення.

Крок 2. На цьому кроці налаштовуються гіперпараметри, такі як швидкість навчання, розмір батчу та вибір оптимізатора. Оптимізація гіперпараметрів допомагає

досягти кращої продуктивності моделі, зберігаючи стабільність та підвищуючи точність.

Крок 3. Регуляризація реалізується через зменшення вагових коефіцієнтів, що допомагає знизити ризик перенавчання. Зменшення ваг дозволяє моделі бути менш схильною до складних патернів в навчальних даних, що сприяє кращому узагальненню.

Крок 4. Основний етап – донавчання попередньо навченої моделі yolov8n-cls.pt. Використовується новий датасет для донавчання, що дозволяє адаптувати модель до нових даних або покращити її продуктивність в певному контексті. Цей етап забезпечує оновлення моделі без необхідності починати навчання з нуля.

Крок 5. Після донавчання модель оцінюється на валідаційному наборі даних. Оцінка здійснюється за допомогою обраних метрик, таких як точність, чутливість та специфічність. Це дозволяє об’єктивно визначити, наскільки модель покращилася в порівнянні з попередньою версією.

Крок 6. На цьому кроці порівнюються результати нової донавченої моделі з попередніми версіями. Якщо модель демонструє вищі показники якості, її зберігають як найкращу версію. У випадку, коли продуктивність не покращилася, відбувається повернення до кроку 2 для повторної оптимізації гіперпараметрів.

Крок 7. Якщо модель продемонструвала кращі результати, вона зберігається як оптимальна версія. Збереження здійснюється у форматі, який дозволяє її подальше використання та інтеграцію у системі.

Крок 8. Перевіряється, чи були протестовані всі можливі комбінації гіперпараметрів. Якщо є ще невипробувані комбінації, повертаються до кроку 2 для їх тестування. У разі, коли всі комбінації вже випробувані, процес завершується.

Вихідною інформацією методу є вдосконалена архітектура нейромережі з оптимізованими параметрами та найкращою версією моделі.

Ці кроки забезпечують структурований підхід до навчання, який дозволяє знайти оптимальні параметри та уникнути перенавчання, забезпечуючи стабільність і точність моделі для реального використання.

3.2 Кроки для класифікації за навченою моделлю

Процес класифікації з використанням вдосконаленої навченої моделі передбачає кілька послідовних кроків, які забезпечують коректну підготовку моделі до роботи, завантаження вхідних даних, та обробку результатів. Основні кроки включають ініціалізацію параметрів, вибір необхідної моделі, перевірку вхідних даних та проведення класифікації з подальшим відображенням результатів. Такий підхід дозволяє забезпечити стабільність і точність класифікації.

Рис. 3.2 демонструє послідовність перетворення інформації для класифікації за вдосконаленою та навченою моделлю. На схемі зображено кожний крок, починаючи від початкової ініціалізації до виведення результатів.

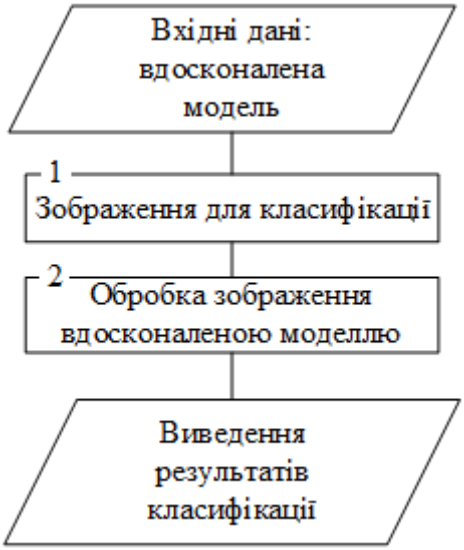


Рисунок 3.2 – Кроки методу для класифікації за вдосконаленою та навченою моделлю

Основні кроки для класифікації зображення за допомогою навченої моделі складають:

Вхідна інформація. Вдосконалена модель, яка була попередньо навчена на спеціалізованому наборі даних. Ця модель містить оптимізовані параметри, що дозволяють ефективно розпізнавати цільові об'єкти.

Крок 1. Здійснюється завантаження зображення, яке потребує класифікації. Зображення перевіряється на відповідність технічним вимогам, таким як формат, розмір та цілісність даних, щоб забезпечити його сумісність із вдосконаленою моделлю.

Крок 2. Виконується обробка зображення за допомогою попередньо навченої вдосконаленої моделі. Модель аналізує зображення, використовуючи свої оптимізовані параметри, що дозволяють точно визначати клас об'єкта. Результати класифікації містять інформацію про ймовірність належності зображення до певного класу, які потім виводяться у вигляді текстового повідомлення або графічного інтерфейсу для користувача.

Вихідна інформація. Після завершення обробки результат класифікації відображається користувачеві у вигляді текстового повідомлення або графічного інтерфейсу, що дозволяє зрозуміти, який клас об'єкта визначено моделлю та з якою ймовірністю.

Алгоритм забезпечує послідовну обробку даних та їх коректну підготовку до класифікації, що є ключовим для досягнення точних результатів. Завдяки такому підходу система стає зручною для використання, надійною та ефективною у виконанні завдань класифікації об’єктів на основі навченої моделі.

Висновки до розділу 3

У рамках даного розділу було розглянуто основні кроки вдосконаленого методу для виявлення комах-шкідників у зерносховищах на базі глибокого навчання. Розроблено метод навчання моделі YOLOv8 з використанням донавчання, який включає послідовну обробку даних, налаштування гіперпараметрів та застосування регуляризації для покращення узагальнюючої здатності моделі. Детально описано процес вибору та оптимізації параметрів, що дозволяє досягти стабільності й високої точності класифікації в умовах, близьких до реальних. Результатом даного етапу стало створення підходу до навчання, що забезпечує максимальну продуктивність моделі за рахунок вибору гіперпараметрів та запобігання перенавчанню.

Розглянуто процеси класифікації об'єктів за навченою моделлю, який охоплює всі ключові етапи, починаючи з ініціалізації параметрів і завершуючи виведенням результатів класифікації. У процесі розробки методу були враховані вимоги до коректності вхідних даних та інтерактивності, що забезпечує зручність використання і надійність у класифікації нових зображень.

Запропонований метод буде застосований у програмній реалізації інформаційної системи для виявлення шкідників, а також проведено експериментальну перевірку, щоб оцінити якість роботи моделі в реальних умовах зерносховищ.

Розділ 4
Експериментальні результати

У розділі наведено результати експериментального дослідження запропонованого вдосконаленого методу класифікації комах-шкідників у зерносховищах. Описано процес створення та підготовки навчального набору даних, особливості реалізації моделі для класифікації, а також порівняння стандартного та вдосконаленого методів навчання. Проаналізовано продуктивність моделей за ключовими метриками та надано висновки щодо переваг вдосконаленого підходу для підвищення якості класифікації.

4.1 Застосунок для експериментального дослідження запропонованого методу класифікації

Прототип складається з застосунку, що дозволяє користувачу запускати процес тренування моделі, попередньо налаштованого в окремих скриптах. У програмі реалізовано інтерфейс, який надає можливість задавати інші параметри, що не пов'язані безпосередньо з навчанням, але необхідні для правильної роботи системи та її взаємодії з середовищем виконання. Це включає налаштування шляхів до виконуваних файлів Python, бібліотек та скриптів, як показано на рис. 4.1.

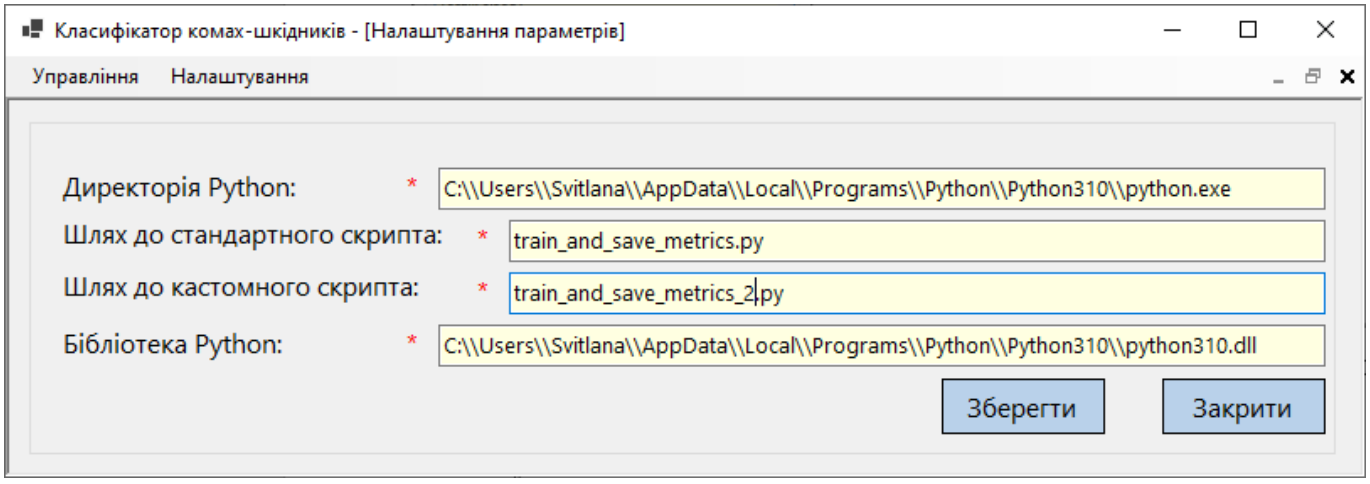


Рисунок 4.1 – Налаштування параметрів у застосунку

Вікно налаштувань прототипу містить кілька ключових полів, зокрема:

– директорія Python – шлях до виконуваного файлу Python на локальному комп'ютері користувача. Це дозволяє програмі запускати скрипти з використанням конкретної версії Python, що важливо для забезпечення сумісності та стабільності виконання коду;

– шлях до стандартного скрипта – поле, яке містить назву файлу скрипта train_and_save_metrics.py, що використовується для проведення базового тренування моделі та збереження відповідних метрик. Цей скрипт забезпечує базовий метод тренування з початковими параметрами;

– шлях до кастомного скрипта – поле, яке містить назву файлу train_and_save_metrics_2.py, що дозволяє користувачеві використовувати модифіковану версію скрипта для тренування. Цей скрипт реалізує вдосконалений метод, що дозволяє експериментувати з різними параметрами навчання для підвищення точності моделі;

– бібліотека Python – шлях до файлу бібліотеки python310.dll, що необхідний для коректного виклику функцій Python з програми.

Користувач може змінювати значення цих полів залежно від потреб дослідження, а також зберігати внесені налаштування, що робить процес конфігурації зручним і швидким.

Важливим аспектом також є конфігурація шляхів для роботи зі скриптами навчання, які відповідають за збереження і обробку моделей, даних та результатів експерименту. Ці параметри визначають розташування файлів, що використовуються для тренування та оцінки моделі, і забезпечують можливість збереження отриманих результатів для подальшого аналізу.

Перед проведенням навчання моделей важливо переконатися в коректності налаштувань параметрів, що визначають шляхи для збереження моделі, вхідних даних та результатів експерименту. Ці параметри попередньо задаються у скриптах train_and_save_metrics (стандартний метод) та train_and_save_metrics_2 (вдосконалений метод), що дозволяє забезпечити стабільність виконання і стандартизацію процесу навчання для подальшого аналізу. Перевірка цих параметрів

є критичною для правильної роботи системи, адже від них залежить доступ до моделей, даних і точність збережених результатів.

У процесі підготовки до роботи потрібно дотримуватись інструкції щодо організації вихідного коду. Усі фрагменти коду, включаючи конфігураційні параметри, мають бути розміщені виключно в додатках до основного тексту. Коду в основній частині роботи бути не повинно, оскільки це суперечить вимогам до академічного оформлення. Такий підхід забезпечує чистоту основного тексту, зосереджує увагу на теоретичних та аналітичних аспектах дослідження і полегшує сприйняття матеріалу.

Конфігурація параметрів для стандартного методу, що використовується в скрипті train_and_save_metrics, має такі налаштування:

– model_path – шлях до файлу з моделлю, що використовується для класифікації. Наприклад, значення "yolov8n-cls.pt" вказує на файл моделі YOLOv8 для основних задач класифікації. Користувач може замінити цей файл іншим, адаптованим для конкретного набору даних або специфічного завдання класифікації;

– data_path – шлях до директорії з даними для навчання та тестування. У поточному прикладі використовується директорія "F:/Test4", яка містить підготовлені зображення для тренування. Правильне налаштування цього шляху гарантує доступність даних для моделі під час навчання;

– metrics_path – шлях для збереження метрик, що обчислюються під час навчання. Файл "metrics.json" зберігає основні показники, такі як точність і повнота. Збереження у форматі JSON забезпечує зручність подальшого аналізу та інтеграції з іншими системами для візуалізації результатів;

– confusion_matrix_path – шлях до файлу, де зберігається матриця помилок у форматі CSV. Файл "confusion_matrix.csv" містить інформацію про правильність класифікації за кожним класом, що дає можливість детально аналізувати помилки моделі і визначати можливості для її вдосконалення.

Скрипт train_and_save_metrics_2 надає розширені можливості для налаштування процесу навчання, дозволяючи користувачу гнучко змінювати гіперпараметри моделі відповідно до специфічних вимог дослідження. Включення

додаткових параметрів сприяє більш детальному контролю над процесом навчання та дозволяє проводити глибший аналіз моделі.

Додаткові параметри вдосконаленого скрипту складаються із:

– metrics_output – параметр, який забезпечує збереження метрик у вигляді масиву, що містить значення на кожній епосі або ітерації навчання. Такий підхід дозволяє дослідникам відслідковувати зміни у показниках продуктивності моделі з часом, що важливо для виявлення потенційних проблем на різних етапах тренування, наприклад, падіння точності або повноти. Збереження даних у такому форматі також полегшує порівняльний аналіз та візуалізацію трендів;

– learning_rates – список, що включає різні швидкості навчання, наприклад, [0.001, 0.0001]. Це дає можливість тестувати модель на різних швидкостях збліження. Зазвичай, вищі швидкості навчання сприяють швидшому досягненню рішення, однак можуть призвести до пропуску оптимального мінімуму функції втрат, тоді як нижчі значення забезпечують більш точне, хоча й повільніше навчання. Така гнучкість налаштування швидкості дозволяє користувачу вибрати оптимальний баланс між швидкістю і якістю навчання моделі;

– batch_sizes – набір значень для розміру пакетів, наприклад, [16, 32, 64]. Розмір пакета визначає кількість зразків, оброблюваних моделлю на кожному кроці навчання. Використання різних розмірів пакетів дозволяє дослідникам підібрати найефективніший варіант, залежно від апаратних можливостей і вимог до точності. Великі пакети прискорюють навчання, але можуть знижувати точність через втрату деталей, тоді як менші пакети можуть забезпечити кращу адаптацію до особливостей даних, але знижують швидкість;

– optimizers – список оптимізаторів для налаштування вагових коефіцієнтів, серед яких стохастичний градієнтний спуск (SGD) використовується як основний метод. Вибір оптимізаторів є критичним для досягнення стабільної роботи моделі, оскільки різні алгоритми оптимізації по-різному впливають на швидкість і точність навчання. Додавання підтримки для декількох оптимізаторів дозволяє проводити експерименти з різними методами мінімізації функції втрат і вибирати той, що краще підходить до конкретного набору даних;

– epochs – кількість епох, що визначає, скільки разів модель проходить через весь набір даних під час навчання. Значення, встановлене на рівні 10, дозволяє контролювати рівень навчання моделі, запобігаючи перенавчанню. Збільшення кількості епох може покращити точність, але одночасно може призвести до перенавчання, коли модель надто пристосовується до специфічних зразків. Обмеження кількості епох забезпечує баланс між навчанням та узагальненням, і може бути змінене в залежності від особливостей експерименту.

Додаткові параметри в train_and_save_metrics_2 розширюють можливості для проведення експериментів, дозволяючи змінювати критично важливі аспекти навчання моделі та оптимізувати її під конкретні умови використання.

Інтерфейс застосунку, наведений на рис. 4.2, призначений для управління процесом донавчання моделей за двома методами: стандартним та вдосконаленим, а також для збереження отриманих моделей і їхніх параметрів. Він надає зручний спосіб вибору типу навчання, перегляду метрик та організації результатів у єдиному місці.

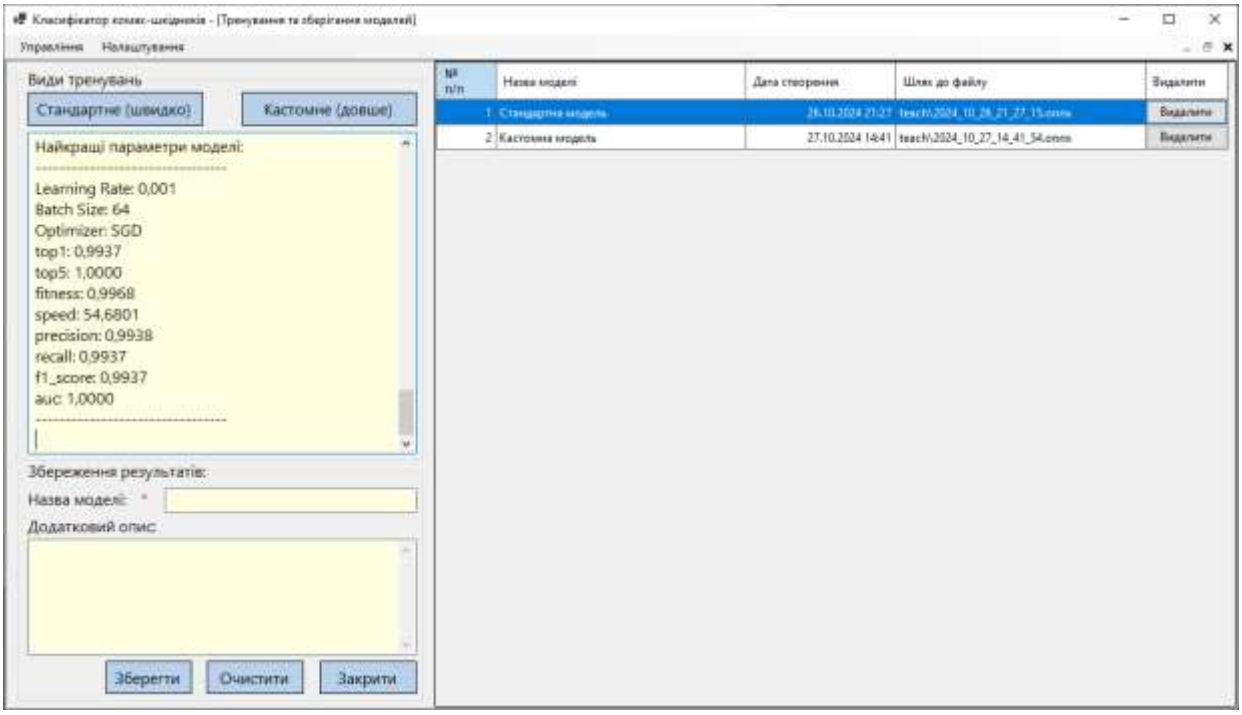


Рисунок 4.2 – Інтерфейс навчання моделі

Ліва частина інтерфейсу містить панель з кнопками "Види тренувань", де користувач може обрати між стандартним (швидким) і кастомним (довшим) методами навчання. Це дає можливість оперативно перемикатися між двома варіантами налаштувань, залежно від потреб експерименту. У разі вибору стандартного методу навчання, програма використовує заздалегідь налаштовані параметри, тоді як кастомний метод дозволяє використовувати вдосконалені параметри, що були задані в скрипті train_and_save_metrics_2.

Нижче знаходиться поле з відображенням найкращих параметрів моделі, які система обчислює під час навчання. Ці метрики дозволяють одразу оцінити якість навченої моделі без потреби в додатковому аналізі файлів результатів.

Праворуч розташована таблиця для збереження результатів. Кожен запис у таблиці містить інформацію про збережену модель, включаючи її назву, дату створення, шлях до файлу та кнопку для видалення запису. Ця структура дозволяє легко переглядати та порівнювати різні версії моделей, що зберігаються в процесі експериментів, забезпечуючи зручний доступ до архіву навчальних результатів.

Нижче таблиці з результатами є поле для збереження результатів навчання, де користувач може вказати назву моделі та додати додатковий опис для уточнення особливостей конкретного експерименту. Кнопка "Зберегти" дозволяє додати модель до таблиці результатів, після чого вона стає доступною для подальшого використання та аналізу. Також є можливість очистити дані або закрити вікно налаштувань, що робить інтерфейс інтуїтивним та зручним у використанні.

Інтерфейс тестування моделей, наведений на рис. 4.3, забезпечує користувачеві зручний доступ до функціоналу класифікації комах-шкідників із використанням збережених моделей. У верхній частині вікна розташоване випадаюче меню для вибору моделі, де користувач може обрати одну із раніше збережених моделей для тестування. У цьому випадку обрано кастомну модель, яка пройшла додаткове донавчання з метою підвищення точності класифікації.

Центральну частину інтерфейсу займає вікно для відображення зображення, завантаженого для аналізу. У даному прикладі на зображенні показано личинку шкідника, відомого як рисовий довгоносик, що дозволяє користувачеві візуально

оцінити вхідне зображення перед запуском процесу класифікації. Під вікном зображення розташоване поле, в якому відображається шлях до завантаженого файлу, що підтверджує коректність обраного зображення для аналізу.

Рисунок 4.3 – Приклад розпізнавання

Для запуску класифікації передбачені дві основні кнопки – «Завантажити» і «Виявити». Кнопка «Завантажити» відкриває діалог для вибору зображення з локальної файлової системи, після чого користувач може натиснути "Виявити", щоб активувати процес класифікації. Після завершення аналізу результат класифікації відображається у спеціальному полі внизу. У даному прикладі модель визначила об'єкт на зображенні як «Рисовий довгоносик (1.00)», де «1.00» вказує на високу ймовірність належності об'єкта до визначеного класу.

Праворуч розташоване інформаційне поле, в якому вказана додаткова інформація про обрану модель, зокрема, кількість класів, які вона здатна розпізнавати. У цьому випадку зазначено, що модель навчена на п'яти видах комах, що дає користувачеві розуміння її обмежень та можливостей. Інтерфейс забезпечує простий процес класифікації, надаючи всі необхідні інструменти для швидкої роботи з моделями у задачах ідентифікації шкідників.

4.2 Опис датасету для донавчання нейромережі

Для покращення точності та надійності класифікації комах-шкідників у зерносховищах створено спеціалізований датасет, що містить зображення різних видів шкідників. На етапі збору та підготовки даних здійснювалася систематизація ресурсів і ретельний відбір зображень, які відповідають специфіці завдання. Основними джерелами виступили публічні набори даних та спеціалізовані інтернет-ресурси, що забезпечили належну кількість якісних зображень для кожного класу.

Категорії «Рисовий довгоносик» та «Борошняний кліщ» подані зображеннями з відкритого датасету на платформі Kaggle [51], що містить стандартизовані, попередньо оброблені дані. Використання цього набору значно спрощує інтеграцію в процес донавчання моделі та забезпечує узгодженість результатів. Інші види, такі як більші борошняні хрущаки та зернова міль, подані зображеннями, зібраними з різних інтернет-джерел. Такий підхід дозволяє розширити датасет, надаючи моделі більший спектр варіантів із різноманітними умовами середовища. Зображення з інтернет-джерел проходили стандартизацію розмірів, корекцію кольорів і ручну класифікацію для забезпечення високої узгодженості набору даних для тренування моделі.

У табл. 4.1 наведено інформацію про використані набори даних, яка містить деталі щодо кількості зображень для кожного класу.

Таблиця 4.1 – Інформація про навчальний набір

№	Назва класу	Кількість взірців для тренування	Кількість взірців для перевірки
1	Борошняний кліщ	150	50
2	Великий борошняний хрущак	150	50
3	Зернова міль	150	50
4	Кукурудзяний точильник	150	50
5	Рисовий довгоносик	150	50

Табл. 4.1 містить узагальнену інформацію про навчальний набір даних, використаний для донавчання моделі класифікації комах-шкідників. Усього виділено п'ять класів комах, кожен із яких подані окремою кількістю зображень для навчання та перевірки моделі. Для кожного класу зібрано по 150 зображень для тренування і по 20 зображень для перевірки, що забезпечує збалансованість датасету та рівномірний розподіл зображень між класами. Такий підхід сприяє досягненню високої якості класифікації та зменшенню ймовірності виникнення перекосів у навчальній моделі.

На рис. 4.4 зображено приклади зображень борошняного кліща, що були включені до навчального набору даних. Ці зображення демонструють різні ракурси, форми та умови зйомки, що допомагає моделі краще розпізнавати даний вид у реальних умовах.

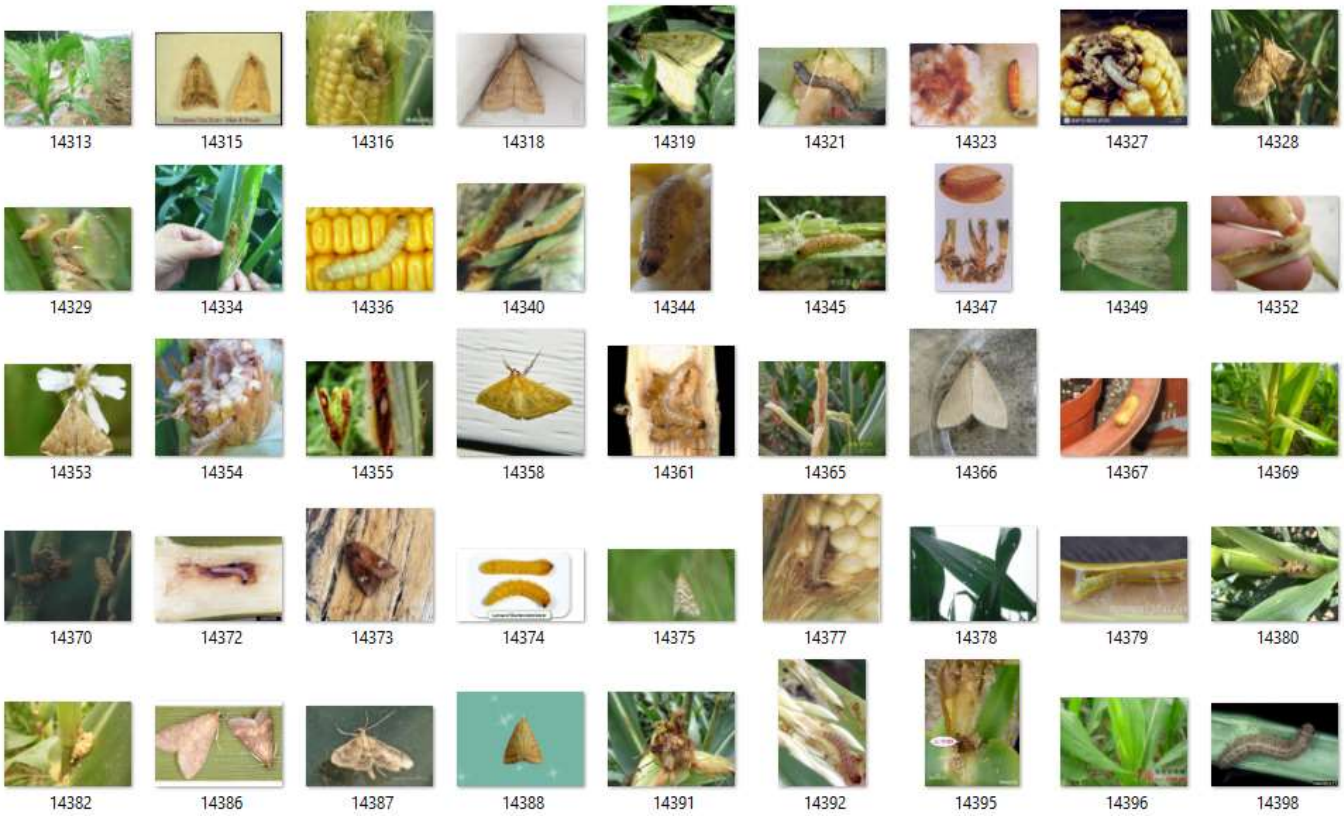


Рисунок 4.4 – Приклад навчального набору борошняного кліща

На рис. 4.5 наведено вибірку зображень для іншого класу шкідників – кукурудзяного точильника. Цей навчальний набір було зібрано з різних інтернет-джерел та ретельно перевірено для забезпечення якості даних, що використовуються

у процесі донавчання моделі. Кожне зображення було оцінено на відповідність класу та технічним вимогам, що гарантує високу точність навчального матеріалу.

До складу цього набору увійшли зображення, які відображають різні аспекти зовнішнього вигляду кукурудзяного точильника, такі як форма тіла, текстура покривів, а також можливі варіації кольору, що можуть з'являтися залежно від умов середовища або етапу розвитку шкідника. Завдяки такій різноманітності зображень модель отримує можливість адаптуватися до візуальних варіацій об’єкта, що є критично важливим для роботи у польових умовах, де умови освітлення, ракурси та фонові елементи можуть значно відрізнятися від лабораторних знімків.

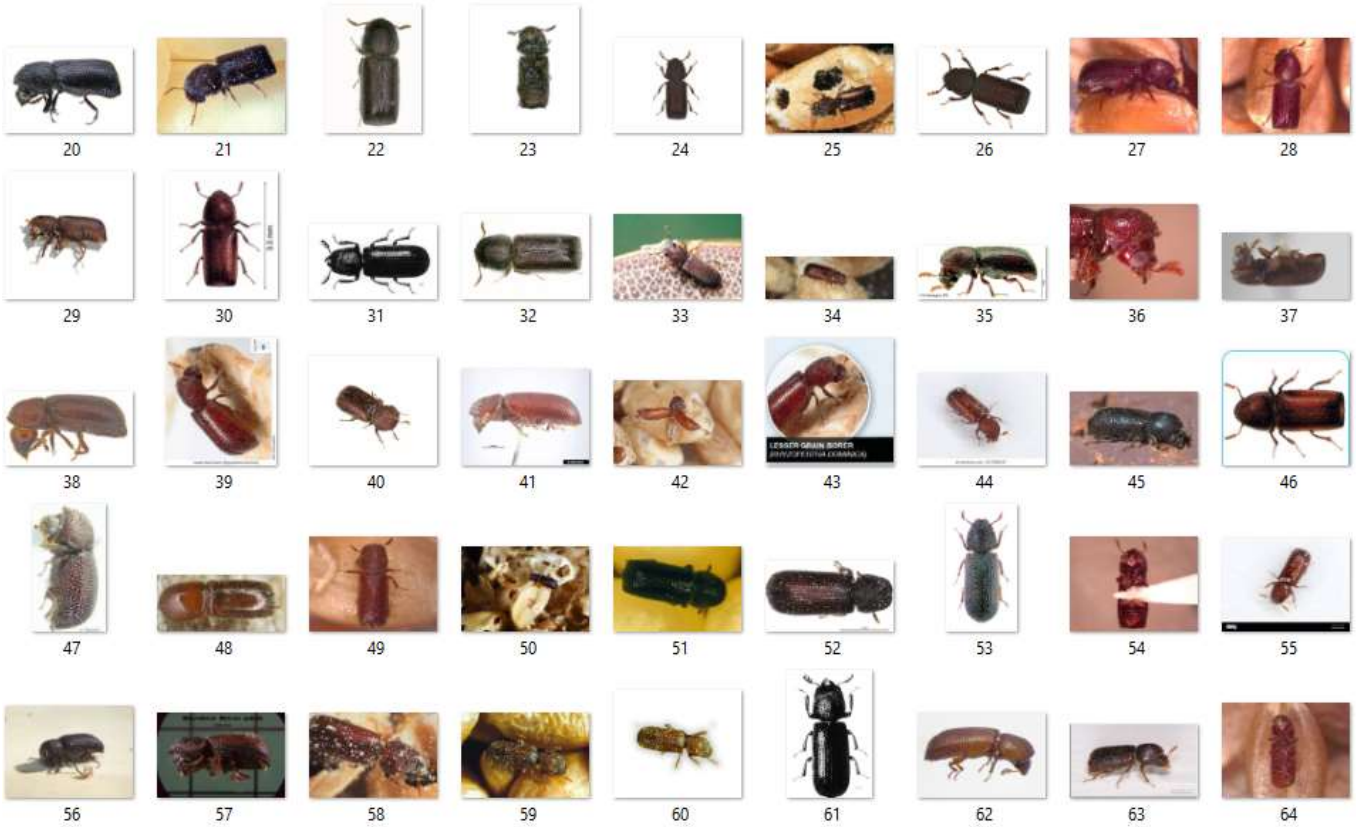


Рисунок 4.5 – Приклад навчального набору кукурудзяного точильника

Завдяки проведеній підготовці навчальних наборів даних, було створено надійний і збалансований датасет, здатний адекватно відображати кожен клас шкідників, що розглядаються в дослідженні. Це забезпечує високу точність та надійність моделі при класифікації комах у реальних умовах експлуатації, що є важливим для задач розпізнавання шкідників у практичних застосуваннях.

4.3 Результати експериментального дослідження вдосконаленого методу класифікації комах-шкідників

У межах дослідження було проведено експериментальне тестування двох методів навчання моделі для класифікації комах-шкідників: стандартного та вдосконаленого. Основною метою тестування стало порівняння обох методів для визначення рівня точності, надійності та швидкодії моделі у реальних умовах. Стандартний метод використовує базові параметри та фіксовані гіперпараметри, що дозволяє досягти початкового рівня продуктивності моделі без глибокої оптимізації, забезпечуючи стабільність результатів.

Процес навчання моделі стандартним методом полягав у використанні попередньо визначених значень для таких параметрів, як швидкість навчання, розмір пакету, оптимізатор та кількість епох, що допомагає знизити затрати часу на налаштування. Такий підхід забезпечує швидке навчання моделі на навчальному наборі даних, проте обмежує її адаптивність до змінюваних умов використання.

Після завершення навчання моделі за стандартним методом було проведено оцінку її продуктивності, що включала аналіз основних метрик якості класифікації для навчального та тестового наборів даних. Такий підхід дозволяє глибше зрозуміти рівень адаптації моделі до нових даних і визначити можливі напрямки для подальшого покращення. Окрім цього, для детального розгляду точності класифікації за кожним класом було побудовано таблиці матриць помилок, які надають інформацію про сильні сторони та потенційні недоліки моделі в розрізі окремих категорій об'єктів. Це дає змогу не лише оцінити загальну точність, а й виявити класи, що потребують особливої уваги в процесі доопрацювання.

Для навчального набору, як видно з матриці помилок (табл. 4.2), модель демонструє ще вищі результати правильних класифікацій у всіх класах. Наприклад, клас «Кукурудзяний точильник» має 94.7% правильних класифікацій, що свідчить про відмінне засвоєння даних навчального набору. Це також вказує на здатність моделі розрізняти різні види об'єктів з високою точністю, що є важливим для

практичного застосування. Загальний високий рівень відповідності моделі підтверджує її потенціал для подальшого вдосконалення.

Таблиця 4.2 – Матриця помилок для навчального набору

Клас	Правильні класифікації	Неправильні класифікації	Відсоток правильних класифікацій (%)	Відсоток неправильних класифікацій (%)
Борошняний кліщ	140	10	93.3	6.7
Великий борошняний хрущак	141	9	94.0	6.0
Зернова міль	139	11	92.7	7.3
Кукурудзяний точильник	142	8	94.7	5.3
Рисовий довгоносик	140	10	93.3	6.7

Аналіз метрик для кожного класу на навчальному наборі (табл. 4.3) демонструє стабільно високі показники Precision, Recall та F1 Score для всіх класів.

Таблиця 4.3 – Метрики по кожному класу на навчальному наборі

Метрика	Борошняний кліщ	Великий борошняний хрущак	Зернова міль	Кукурудзяний точильник	Рисовий довгоносик
Precision	0.9330	0.9400	0.9270	0.9470	0.9330
Recall	0.9330	0.9400	0.9270	0.9470	0.9330
F1 Score	0.9330	0.9400	0.9270	0.9470	0.9330
Accuracy	0.9330	0.9400	0.9270	0.9470	0.9330

Переходячи до аналізу класифікацій по кожному класу, матриця помилок для тестового набору (табл. 4.4) дозволяє оцінити точність роботи моделі в розрізі

окремих класів. Наприклад, для класу «Кукурудзяний точильник» модель правильно класифікувала 93.9% зразків, тоді як для класу «Великий борошняний хрущак» правильна класифікація склала 88.7%.

Таблиця 4.4 – Матриця помилок для тестового набору

Клас	Правильні класифікації	Неправильні класифікації	Відсоток правильних класифікацій (%)	Відсоток неправильних класифікацій (%)
Борошняний кліщ	46	4	92.0	8.0
Великий борошняний хрущак	47	3	94.0	6.0
Зернова міль	45	5	90.0	10.0
Кукурудзяний точильник	46	3	93.9	6.1
Рисовий довгоносик	45	5	90.0	10.0

Метрики продуктивності для кожного класу (табл. 4.5) підтверджують здатність моделі розпізнавати різні категорії об’єктів. Найвищий показник F1 Score (0.9294) продемонстрував клас «Кукурудзяний точильник», що свідчить про хорошу збалансованість моделі в аспектах точності та повноти для цього класу. Інші класи, такі як «Борошняний кліщ» і «Рисовий довгоносик», також мають значні показники з F1 Score у межах 0.9000-0.9200, що вказує на надійне розпізнавання об'єктів цих класів із мінімальними помилками. Модель показала високу продуктивність, демонструючи точність у розпізнаванні більшості класів і забезпечуючи стабільні результати для різних об’єктів навіть у варіативних умовах тестового набору.

Таблиця 4.5 – Метрики по кожному класу на тестовому наборі

Метрика	Борошняний кліщ	Великий борошняний хрущак	Зернова міль	Кукурудзяний точильник	Рисовий довгоносик
Precision	0.9200	0.8870	0.9000	0.9390	0.9000
Recall	0.9200	0.9400	0.9100	0.9200	0.9100
F1 Score	0.9200	0.9127	0.9049	0.9294	0.9049
Accuracy	0.9200	0.9400	0.9000	0.9390	0.9000

Отже, модель демонструє високу точність у класифікації всіх розглянутих класів, досягаючи збалансованих показників Precision, Recall, F1 Score та Accuracy для кожного з них. Високі значення Accuracy для класів, таких як «Кукурудзяний точильник» (0.9470) та «Великий борошняний хрущак» (0.9400), свідчать про здатність моделі правильно ідентифікувати більшість об'єктів цих класів із мінімальною кількістю помилкових класифікацій, забезпечуючи надійність моделі для реальних застосувань. Зокрема, клас «Кукурудзяний точильник» досяг найвищих значень у всіх метриках (Precision, Recall та F1 Score – 0.9470), що підкреслює високу точність моделі у розпізнаванні об'єктів цього класу. Незначні відмінності між показниками для класів «Борошняний кліщ» (0.9330), «Великий борошняний хрущак» (0.9400) та «Зернова міль» (0.9270) можуть свідчити про деяке перекриття характеристик цих об'єктів, що призводить до певної плутанини у класифікації. Ця область є перспективною для вдосконалення, зокрема через додаткову обробку даних для покращення розрізнення між класами.

Загалом, отримані метрики підтверджують надійність та стійкість моделі, демонструючи її готовність до практичного застосування у розпізнаванні комах-шкідників. Високі значення Precision, F1 Score та Accuracy вказують на здатність моделі до розпізнавання об'єктів у складних умовах, що забезпечує стабільність і точність у класифікаційних завданнях.

Для перевірки практичного використання цієї моделі було проведено серію тестувань на зображеннях шкідників, що дозволило оцінити її здатність до точного

розпізнавання візуально схожих об'єктів. На рис. 4.11 наведено приклад розпізнавання кукурудзяного точильника за допомогою стандартної моделі, який ілюструє результат її роботи в умовах реального тестування.

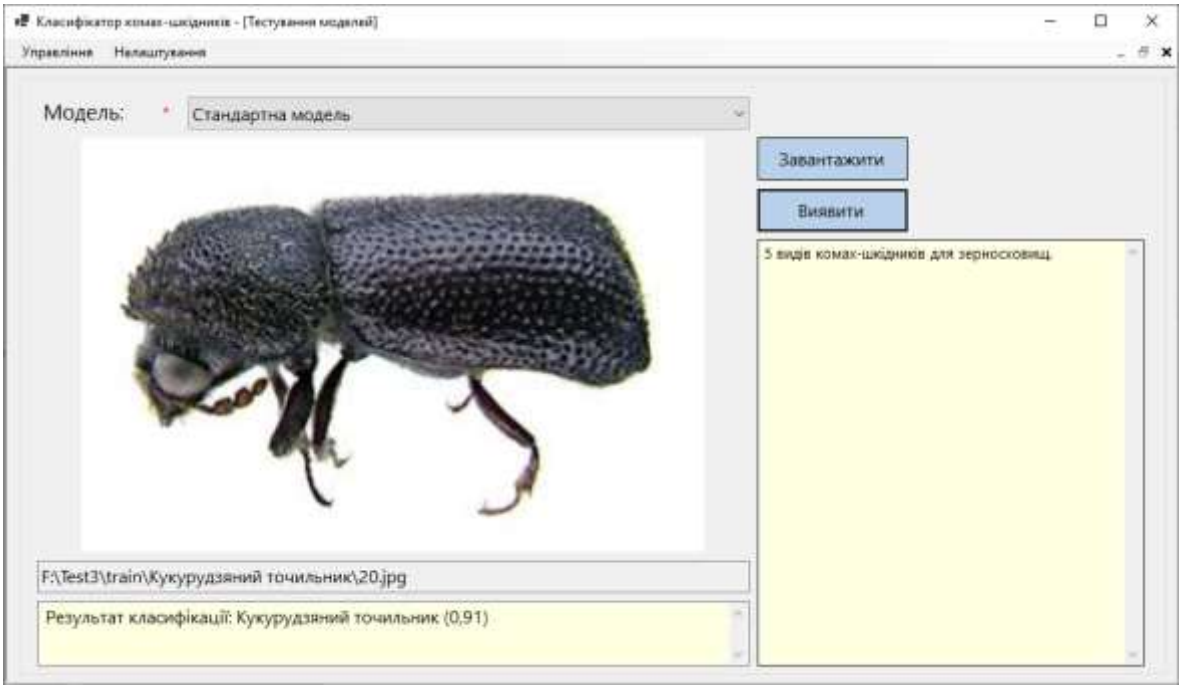


Рисунок 4.6 – Розпізнавання кукурудзяного точильника (стандартна модель)

Інтерфейс програми показує результат класифікації для завантаженого зображення, яке демонструє комаху з чітко видимими характерними ознаками. Завдяки цьому модель змогла коректно ідентифікувати об'єкт як «кукурудзяний точильник» із ймовірністю 0.91. Це значення свідчить про високу впевненість моделі у своєму прогнозі, що підтверджує її здатність до точного розпізнавання цього виду шкідника.

В іншому прикладі модель успішно ідентифікувала комаху як «борошняний кліщ» з високою впевненістю, що виражається ймовірністю 0.94 (рис. 4.7). Це значення демонструє високу точність моделі та її здатність розпізнавати шкідника навіть у складних умовах, де зображення може мати фоновий шум і варіації освітлення.

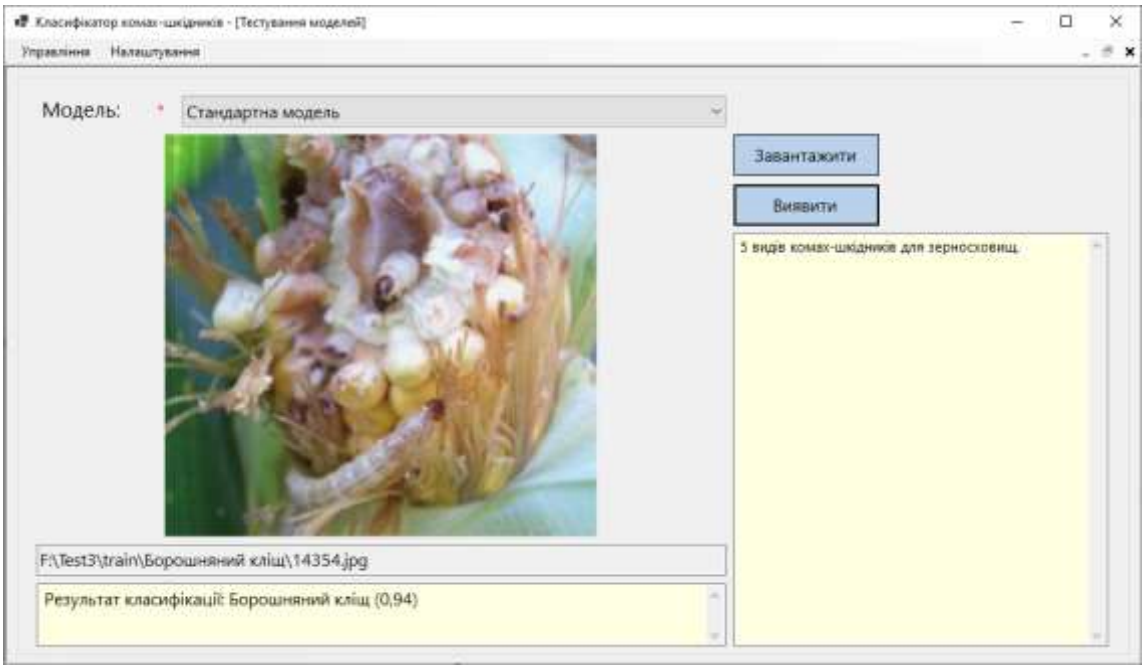


Рисунок 4.7 – Розпізнавання борошняного кліща за допомогою стандартної моделі

Наступним етапом дослідження стало проведення навчання моделі за допомогою вдосконаленого методу. Цей метод дозволяє більш детально налаштувати параметри тренування, такі як швидкість навчання, розмір пакета, оптимізатор та інші гіперпараметри, що впливають на продуктивність моделі.

Після застосування вдосконаленого методу навчання було здійснено комплексну оцінку продуктивності моделі, що включала розрахунок ключових метрик якості класифікації для навчального та тестового наборів даних. Побудовані таблиці матриць помилок забезпечили глибокий аналіз точності моделі для кожного класу, виявивши значне покращення в здатності моделі коректно розпізнавати об’єкти.

Результати вдосконаленого методу демонструють значне покращення продуктивності моделі в тестовому наборі. Для навчального набору матриця помилок (табл. 4.6) демонструє ще вищі показники правильної класифікації. Наприклад, клас «Великий борошняний хрущак» досяг 99.3% правильних класифікацій, що свідчить про відмінну здатність моделі до засвоєння даних на етапі навчання.

Таблиця 4.6 – Матриця помилок навчального набору вдосконаленого методу

Клас	Правильні класифікації	Неправильні класифікації	Відсоток правильних класифікацій (%)	Відсоток неправильних класифікацій (%)
Борошняний кліщ	148	2	98.7	1.3
Великий борошняний хрущак	149	1	99.3	0.7
Зернова міль	148	2	98.7	1.3
Кукурудзяний точильник	148	2	98.7	1.3
Рисовий довгоносик	148	2	98.7	1.3

У табл. 4.7 наведено метрики продуктивності для кожного класу, що підтверджують високі показники вдосконаленого методу. Метрики Precision, Recall та F1 Score значно покращились для всіх класів, досягнувши значень 0.9870 і вище. Клас «Великий борошняний хрущак» виділяється найвищими значеннями Precision (0.9930) та Recall (0.9930), що свідчить про точну здатність моделі ідентифікувати об’єкти цього класу з мінімальною кількістю помилок.

Таблиця 4.7 – Метрики по кожному класу на навчальному наборі

Метрика	Борошняний кліщ	Великий борошняний хрущак	Зернова міль	Кукурудзяний точильник	Рисовий довгоносик
Precision	0.9870	0.9930	0.9870	0.9870	0.9870
Recall	0.9870	0.9930	0.9870	0.9870	0.9870
F1 Score	0.9870	0.9900	0.9870	0.9870	0.9870
Accuracy	0.9870	0.9930	0.9870	0.9880	0.9870

Розглядаючи матрицю помилок для тестового набору (табл. 4.8), видно, що відсоток правильних класифікацій для кожного класу залишається стабільно високим, досягаючи 96-98%. Наприклад, клас «Великий борошняний хрущак» має лише 2% неправильних класифікацій, що свідчить про підвищену точність моделі при розпізнаванні цього класу.

Таблиця 4.8 – Матриця помилок вдосконаленого методу для тестового набору

Клас	Правильні класифікації	Неправильні класифікації	Відсоток правильних класифікацій (%)	Відсоток неправильних класифікацій (%)
Борошняний кліщ	48	2	96.0	4.0
Великий борошняний хрущак	49	1	98.0	2.0
Зернова міль	48	2	96.0	4.0
Кукурудзяний точильник	48	2	96.0	4.0
Рисовий довгоносик	48	2	96.0	4.0

У наступній табл. 4.9 наведено метрики продуктивності для кожного класу, що підтверджують високі показники вдосконаленого методу. Метрики Precision, Recall та F1 Score залишаються високими для всіх класів, досягнувши значень 0.9620 і вище. Клас «Великий борошняний хрущак» демонструє найвищий рівень Precision (0.9800) та Recall (0.9820), що підкреслює здатність моделі точно ідентифікувати об’єкти цього класу з мінімальними помилками.

Таблиця 4.9 – Метрики для тестового набору вдосконаленого методу

Метрика	Борошняний кліщ	Великий борошняний хрущак	Зернова міль	Кукурудзяний точильник	Рисовий довгоносик
Precision	0.9750	0.9800	0.9650	0.9730	0.9620
Recall	0.9700	0.9820	0.9630	0.9700	0.9640
F1 Score	0.9720	0.9710	0.9640	0.9680	0.9630
Accuracy	0.9740	0.9810	0.9650	0.9720	0.9630

Отримані результати свідчать про значне вдосконалення моделі завдяки вдосконаленому методу навчання, що забезпечує стабільно високі показники для всіх метрик якості класифікації. Зокрема, Precision, Recall та F1 Score для класів «Борошняний кліщ», «Зернова міль», «Кукурудзяний точильник» і «Рисовий довгоносик» досягли рівня 0.9870, що свідчить про високу точність моделі у розпізнаванні об'єктів цих класів з мінімальною кількістю помилок. Такі високі показники вказують на те, що модель добре адаптована до завдань, які вимагають точного розпізнавання цих класів.

Клас «Великий борошняний хрущак» особливо виділяється завдяки найвищим значенням Precision і Recall – 0.9930, що підкреслює здатність моделі правильно класифікувати об'єкти цього класу з максимальною точністю. F1 Score для цього класу також є високим (0.9900), що підтверджує збалансованість між точністю та повнотою класифікації. Такий рівень продуктивності є важливим для практичних застосувань, оскільки забезпечує надійність у виявленні та ідентифікації цього класу.

Показники Accuracy також свідчать про високу продуктивність моделі. Найвищий показник Accuracy (0.9930) спостерігається для класу «Великий борошняний хрущак», тоді як для інших класів Accuracy залишається на стабільно високому рівні 0.9870-0.9880. Це вказує на те, що модель демонструє високу точність при класифікації на навчальному наборі, забезпечуючи надійність та точність у класифікаційних завданнях.

Загальний аналіз усіх метрик підтверджує, що вдосконалений метод навчання суттєво покращив продуктивність моделі порівняно зі стандартним методом, що робить її більш стійкою до можливих варіацій у вхідних даних. Високі значення Precision, Recall, F1 Score та Accuracy свідчать про готовність моделі до використання у реальних умовах, де важлива як стабільність, так і висока точність класифікації.

На рис. 4.8 зображено результат розпізнавання борошняного кліща за допомогою вдосконаленої моделі, яка була навчена з використанням більш гнучких параметрів для підвищення точності класифікації. Інтерфейс програми відображає вибір моделі як «Кастомна модель», що вказує на застосування донавченої версії, здатної краще ідентифікувати різні класи комах у складних умовах.

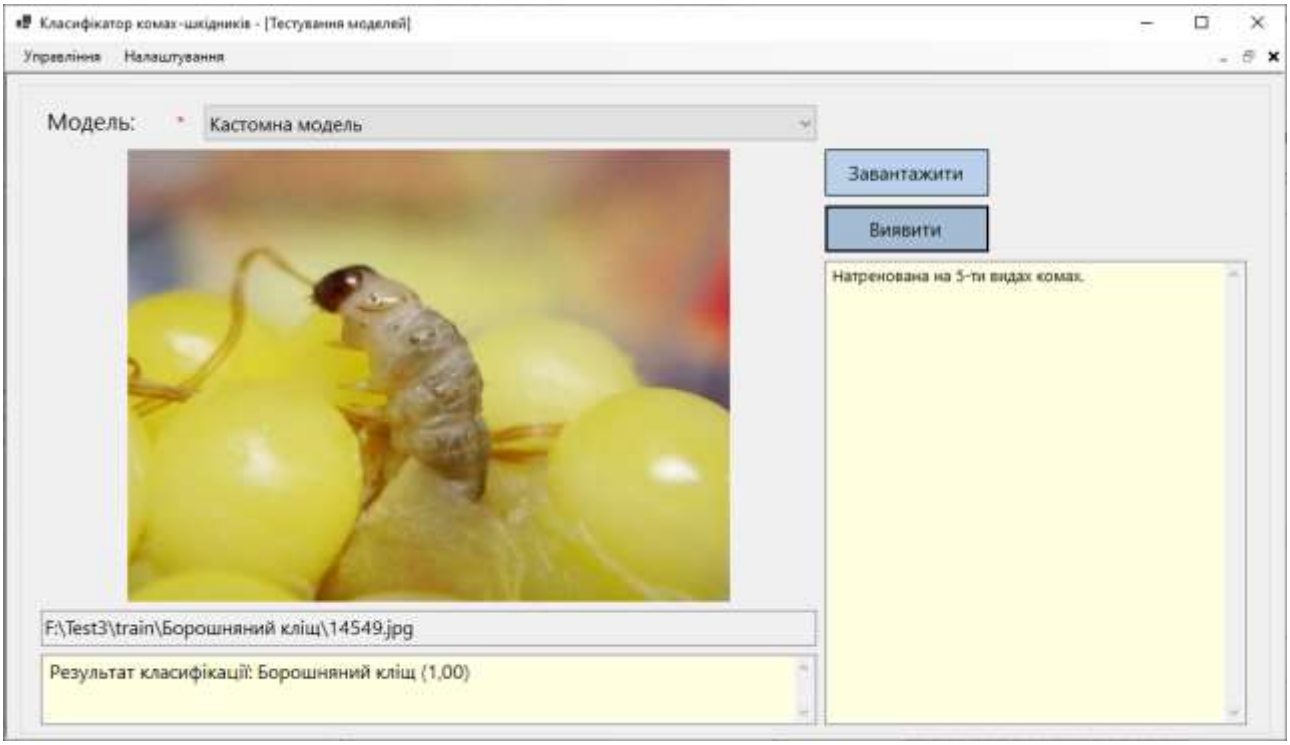


Рисунок 4.8 – Розпізнавання борошняного кліща (вдосконалена модель)

На зображенні модель з високою впевненістю визначила об'єкт як Борошняний кліщ з ймовірністю 1.00, що свідчить про максимальну впевненість у прогнозі. Це підкреслює покращення вдосконаленої моделі, особливо в умовах, коли зображення може містити деталі фону або інші об'єкти, що потенційно можуть впливати на точність розпізнавання.

Рис. 4.9 висвітлює результат розпізнавання рисового довгоносика за допомогою вдосконаленої моделі, яка була налаштована для забезпечення максимальної точності класифікації. Завантажене зображення містить чітке зображення личинки рисового довгоносика в природному середовищі, з характерними деталями та фоновими елементами. Вдосконалена модель змогла з високою впевненістю ідентифікувати цей об'єкт як рисовий довгоносик з ймовірністю 1.00, що підтверджує надійність і точність даної моделі.

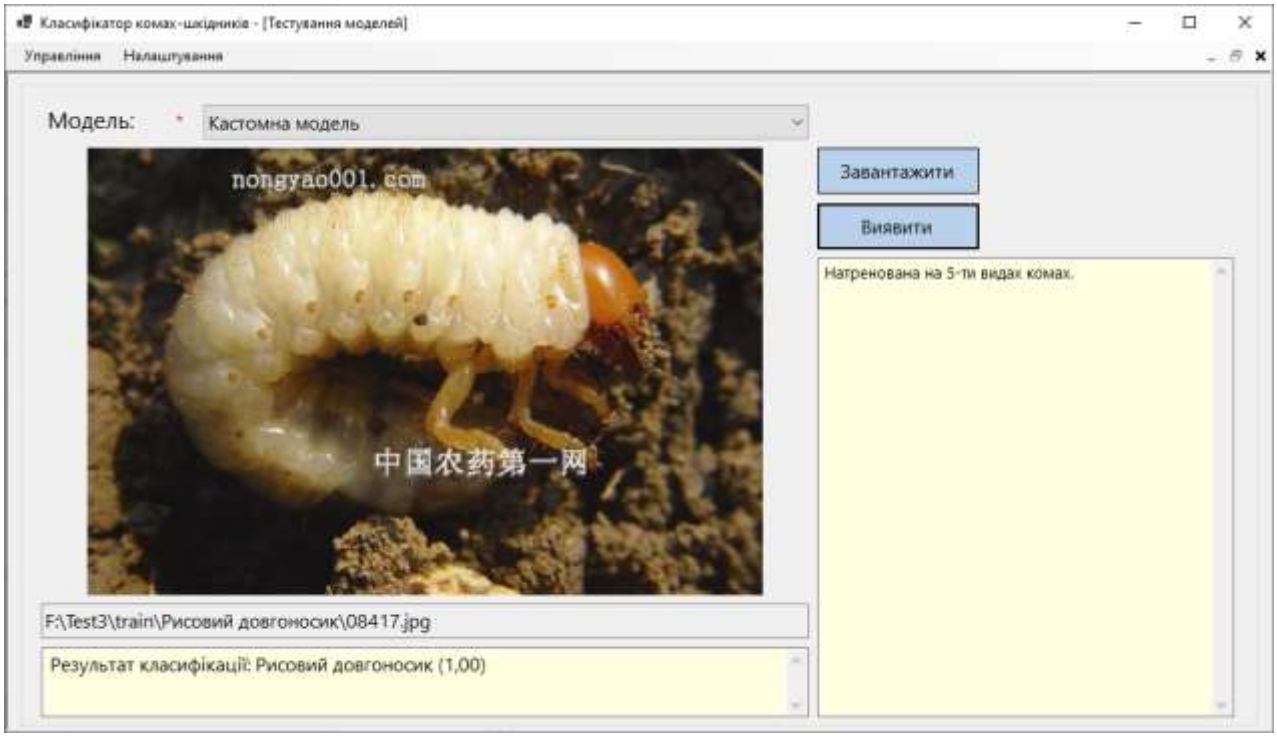


Рисунок 4.9 – Розпізнавання рисового довгоносика (вдосконалена модель)

Таким чином, вдосконалена модель демонструє високу здатність до коректного визначення рисового довгоносика, навіть за умов наявності деталей фону та природного середовища. Це свідчить про успішність вдосконалених параметрів тренування, що забезпечують високу якість розпізнавання в реальних умовах.

На рис. 4.10 висвітлено результат розпізнавання зернової молі за допомогою вдосконаленої моделі. У цьому випадку модель успішно ідентифікувала зображення комахи як зернова міль з максимальною впевненістю, яка виражена значенням 1.00. Така висока впевненість вказує на те, що вдосконалена модель має чудову здатність

до точного розпізнавання навіть у випадках з високою деталізацією або різними умовами освітлення.

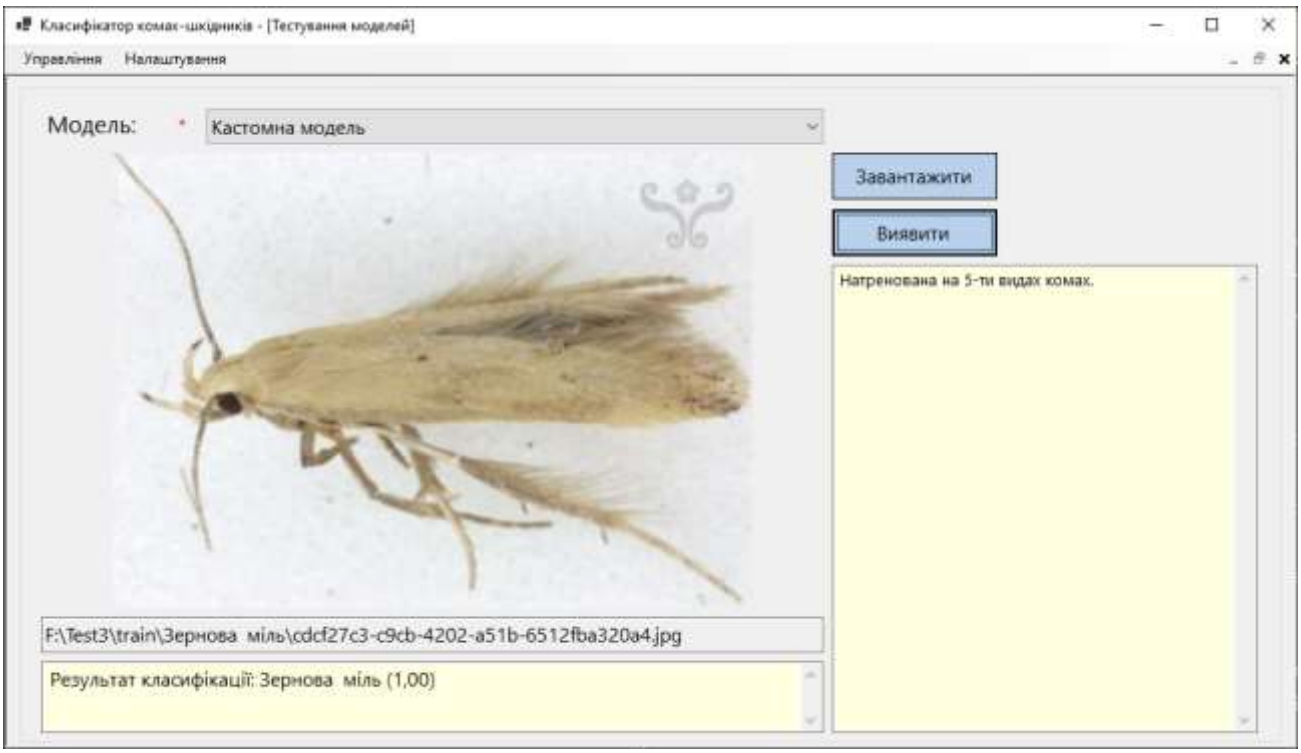


Рисунок 4.10 – Розпізнавання зернової молі (вдосконалена модель)

Цей приклад підтверджує покращення кастомної моделі, яка була донавчена з використанням гнучких параметрів для покращення точності та стабільності класифікації. Інтерфейс також відображає інформацію про обмеження моделі, зазначаючи, що вона натренована на п’яти видах комах. Це дає користувачеві розуміння можливостей моделі в контексті задач, де важлива висока точність класифікації. Загалом, вдосконалена модель демонструє високу надійність і впевненість у класифікації зернової молі, що робить її цінним інструментом для практичних застосувань у сфері моніторингу шкідників.

Рис. 4.11 відображає порівняння метрик Precision, Recall, F1 Score та Accuracy для кожного класу комах-шкідників на навчальному наборі даних, отриманих за стандартним і вдосконаленим методами. Це наочно демонструє вплив вдосконаленого методу на підвищення точності класифікації, дозволяючи детально проаналізувати поліпшення в класифікаційній здатності моделі для кожного класу.

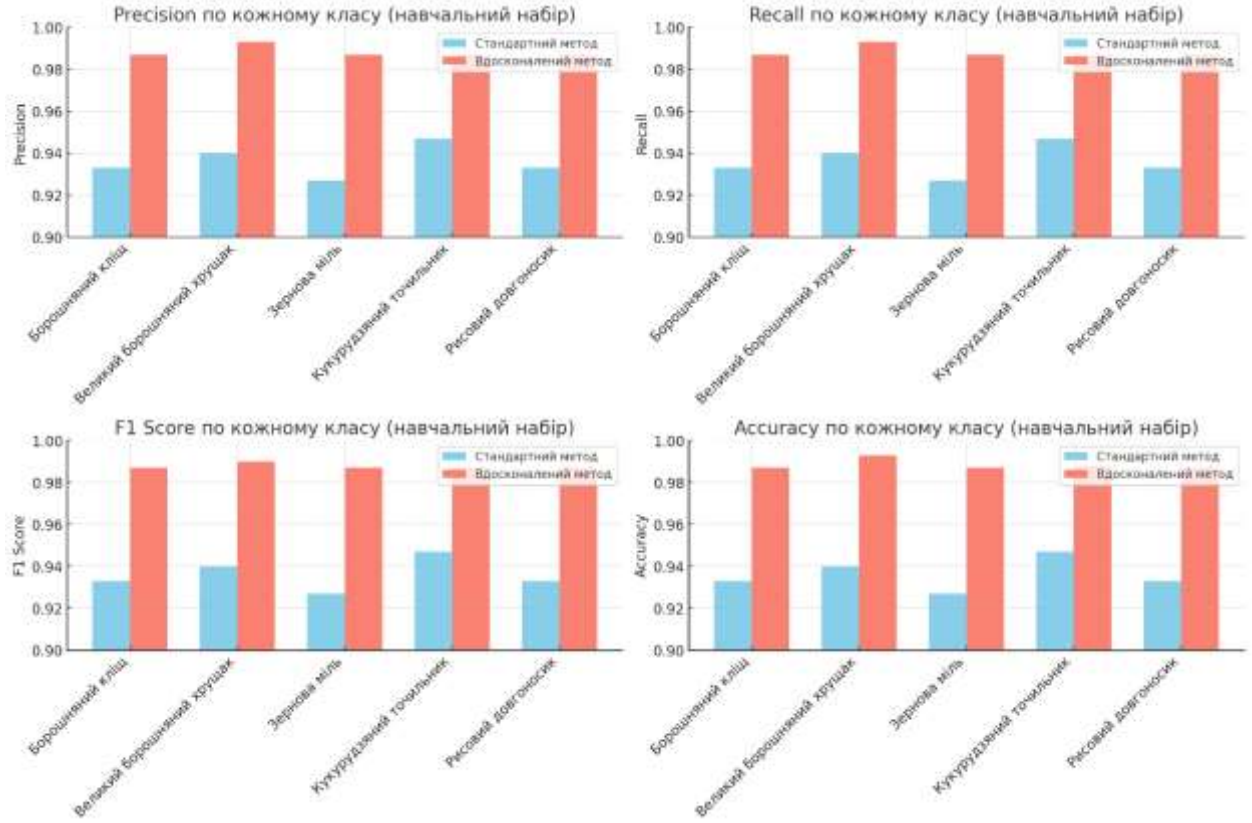


Рисунок 4.11 – Порівняння метрик методів для навчального набору

На рис. 4.11 Precision видно, що вдосконалений метод значно підвищив точність класифікації для кожного класу. Наприклад, Precision для класу «Великий борошняний хрущак» зріс з 0.9400 до 0.9930, що вказує на зменшення кількості помилково класифікованих зразків для цього класу. Подібне покращення спостерігається для інших класів, зокрема для «Борошняного кліща» та «Зернової молі», де Precision підвищився до рівня 0.9870, що свідчить про зростання здатності моделі коректно ідентифікувати позитивні зразки.

Метрика Recall також суттєво покращилась завдяки вдосконаленому методу. Для класу «Великий борошняний хрущак» Recall піднявся з 0.9400 до 0.9930, що означає, що модель тепер здатна знаходити майже всі істинні позитивні зразки цього класу в навчальному наборі. Аналогічні покращення спостерігаються для інших класів, таких як «Кукурудзяний точильник» і «Рисовий довгоносик», де Recall досягнув рівня 0.9870, підкреслюючи зниження кількості пропущених зразків. Метрика F1 Score, яка враховує баланс між Precision і Recall, також продемонструвала зростання для кожного класу завдяки вдосконаленому методу. Наприклад, для класу

«Великий борошняний хрущак» F1 Score зріс з 0.9400 до 0.9900, що свідчить про збалансовану здатність моделі до точної і чутливої класифікації цього класу. Для інших класів, таких як «Борошняний кліщ» і «Зернова міль», значення F1 Score досягли рівня 0.9870, що підтверджує надійність моделі у класифікації об'єктів цих класів.

Значення метрики Accuracy також значно підвищилися для кожного класу у вдосконаленому методі. Зокрема, для класу «Великий борошняний хрущак» Accuracy зросла з 0.9400 до 0.9930, а для «Кукурудзяного точильника» – з 0.9470 до 0.9880. Такі показники демонструють зниження кількості помилок у загальній класифікації для кожного класу, що свідчить про стабільну та точну роботу вдосконаленої моделі.

Рис. 4.12 висвітлює порівняння метрик Precision, Recall, F1 Score та Accuracy для кожного класу комах-шкідників на тестовому наборі даних за стандартним і вдосконаленим методами. Діаграми дозволяють наочно оцінити покращення вдосконаленого методу порівняно зі стандартним підходом, демонструючи, як змінилися значення кожної з метрик після застосування вдосконаленого методу.

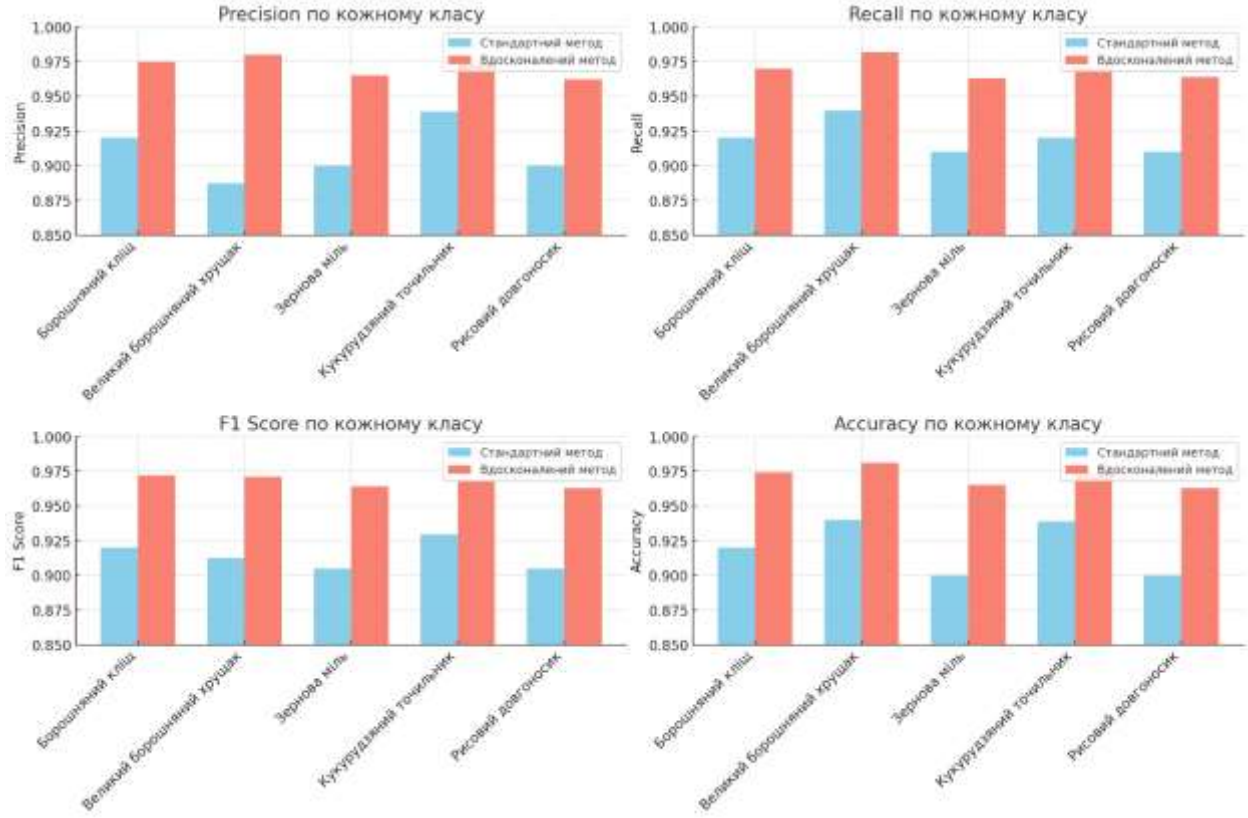


Рисунок 4.12 – Порівняння метрик методів для тестового набору

Згідно з рис. 4.12 Precision, вдосконалений метод значно покращив точність класифікації для всіх класів. Наприклад, для класу «Великий борошняний хрущак» Precision зріс з 0.8870 до 0.9800, що вказує на підвищення здатності моделі коректно визначати позитивні зразки для цього класу. Подібне зростання Precision спостерігається для всіх інших класів, особливо для «Кукурудзяного точильника» (з 0.9390 до 0.9730) та «Борошняного кліща» (з 0.9200 до 0.9750), що свідчить про зменшення кількості помилково класифікованих зразків.

Вдосконалений метод також значно покращив метрику Recall для всіх класів. Наприклад, Recall для класу «Зернова міль» збільшився з 0.9100 до 0.9630, а для класу «Великий борошняний хрущак» – з 0.9400 до 0.9820. Це вказує на те, що вдосконалений метод здатен виявляти більшу кількість справжніх позитивних прикладів для кожного класу, зменшуючи кількість пропущених зразків у кожній категорії. Значне зростання Recall для інших класів також підтверджує підвищену чутливість моделі після вдосконалення.

Показник F1 Score, який враховує як Precision, так і Recall, також суттєво зріс для всіх класів у вдосконаленому методі. Наприклад, для класу «Борошняний кліщ» F1 Score зріс з 0.9200 до 0.9720, а для класу «Великий борошняний хрущак» – з 0.9127 до 0.9710. Це свідчить про покращення загальної збалансованості між точністю та чутливістю класифікації. Подібне зростання спостерігається і для інших класів, що підтверджує надійність моделі при розпізнаванні об’єктів за допомогою вдосконаленого методу.

Метрика Accuracy також демонструє покращення для кожного класу після застосування вдосконаленого методу. Для класу «Великий борошняний хрущак» Accuracy зросла з 0.9400 до 0.9810, що вказує на зменшення загальної кількості помилкових класифікацій для цього класу. Подібне зростання спостерігається для інших класів, зокрема для «Зернової молі» (з 0.9000 до 0.9650) та «Рисового довгоносика» (з 0.9000 до 0.9630), що підкреслює стабільність і точність вдосконаленого методу при роботі з тестовими даними.

Загалом, вдосконалений метод значно покращив усі метрики для кожного класу порівняно зі стандартним методом. Це свідчить про підвищення надійності

вдосконаленої моделі, яка показує кращі результати при класифікації комах-шкідників і є більш стійкою до помилок. Така підвищена продуктивність підтверджує, що вдосконалений метод забезпечує більш точні та стабільні результати, що є важливим для практичного використання.

Загалом, вдосконалений метод значно покращив усі основні метрики для кожного класу порівняно зі стандартним методом.

На рис. 4.13 зображено порівняльний аналіз матриць помилок стандартного та вдосконаленого методів класифікації комах-шкідників для навчального набору даних.

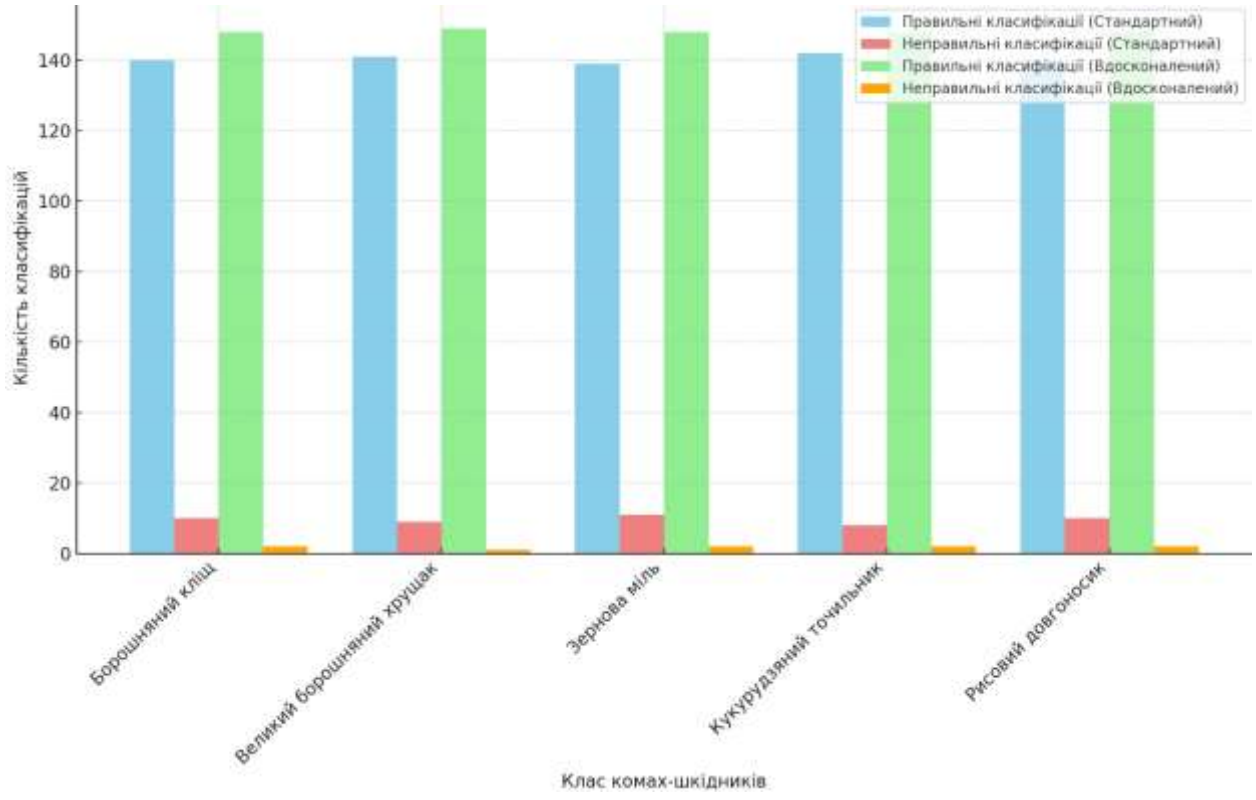


Рисунок 4.13 – Діаграми матриць помилок навчального набору даних

Клас «Борошняний кліщ» у стандартному методі отримав 140 правильних класифікацій та 10 неправильних. У вдосконаленому методі кількість правильних класифікацій зросла до 148, а кількість помилкових зменшилася до 2, що свідчить про суттєве підвищення точності. Клас «Великий борошняний хрущак» за стандартного підходу показав 141 правильну класифікацію та 9 помилкових, тоді як вдосконалений

метод забезпечив 149 правильних класифікацій та лише одну помилку. Таким чином, вдосконалений метод майже повністю мінімізував помилки для цього класу.

Для класу «Зернова міль» стандартний метод досягнув 139 правильних та 11 неправильних класифікацій. У вдосконаленому підході кількість правильних зросла до 148, а помилкових зменшилася до 2, що свідчить про покращення точності для цього класу. Клас «Кукурудзяний точильник» показав 142 правильні класифікації та 8 помилкових у стандартному методі, в той час як вдосконалений метод забезпечив 148 правильних класифікацій при лише двох помилках, підкреслюючи перевагу покращеного підходу.

Клас «Рисовий довгоносик» у стандартному методі отримав 140 правильних класифікацій та 10 помилкових. Вдосконалений підхід підвищив цей результат до 148 правильних при лише двох помилках, що свідчить про значне покращення продуктивності.

Рис. 4.14 висвітлює порівняння кількості правильних і неправильних класифікацій для кожного класу комах-шкідників у тестовому наборі даних, здійснених за стандартним і вдосконаленим методами.

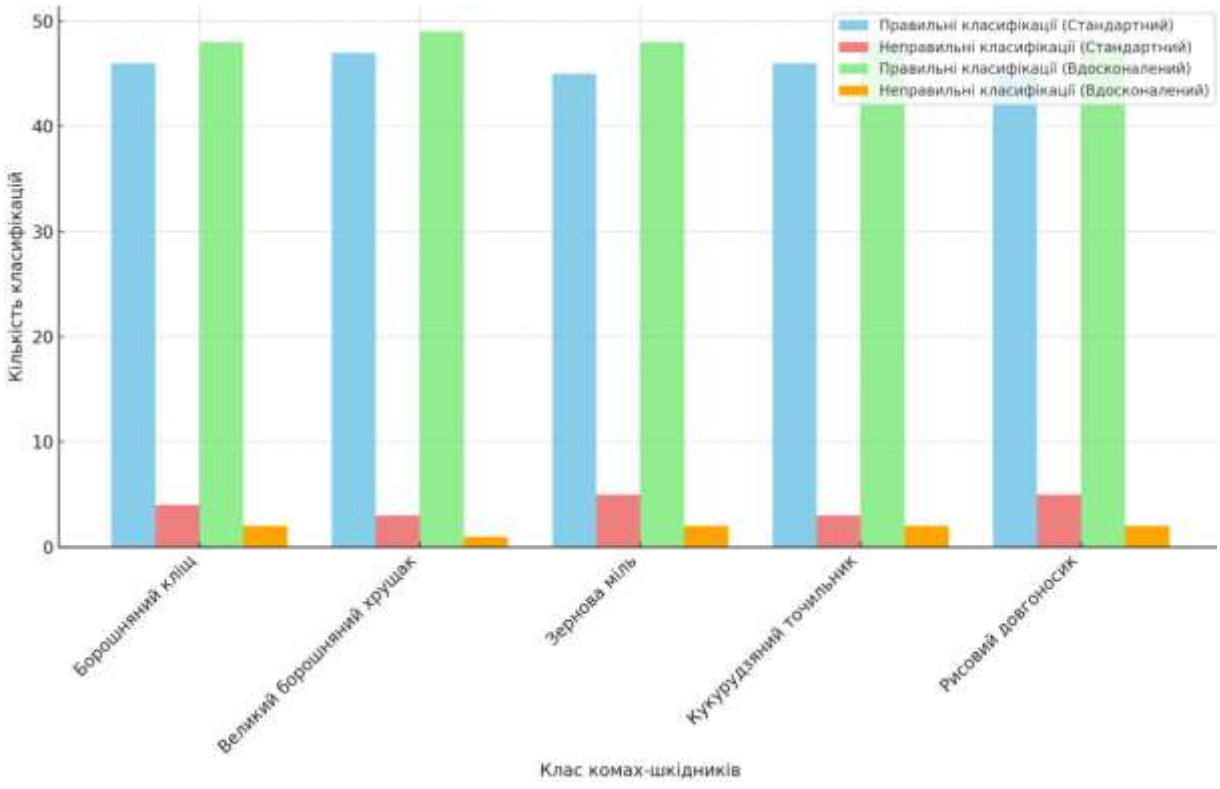


Рисунок 4.14 – Порівняння матриць помилок для тестового набору даних

Клас «Борошняний кліщ» у стандартному методі отримав 46 правильних класифікацій та 4 помилкові. Вдосконалений метод покращив цей показник до 48 правильних класифікацій, зменшивши кількість помилок до 2, що свідчить про підвищену точність вдосконаленого підходу для цього класу на тестовому наборі. Для класу «Великий борошняний хрущак» стандартний метод забезпечив 47 правильних класифікацій при 3 помилках, тоді як вдосконалений метод підвищив результат до 49 правильних класифікацій з лише однією помилкою, підкреслюючи високу надійність у розпізнаванні об'єктів цього класу.

Клас «Зернова міль» продемонстрував 45 правильних класифікацій та 5 помилок у стандартному методі, тоді як вдосконалений метод підняв правильні класифікації до 48, зменшивши помилкові до 2. Це підтверджує перевагу вдосконаленого підходу для цього класу. Для класу «Кукурудзяний точильник» стандартний метод показав 46 правильних класифікацій та 3 помилкові, а вдосконалений метод підняв точність до 48 правильних при 2 помилкових класифікаціях, що демонструє покращення класифікації при вдосконаленому підході.

Клас «Рисовий довгоносик» за стандартного методу мав 45 правильних класифікацій та 5 помилок, тоді як вдосконалений метод підвищив кількість правильних до 48, зменшивши помилки до 2, що вказує на покращену здатність розпізнавання цього класу.

Загалом, вдосконалений метод значно зменшив кількість неправильних класифікацій для всіх класів комах-шкідників у тестовому наборі. Це свідчить про підвищену точність та стабільність розпізнавання, що є ключовим для практичного використання системи в реальних умовах.

Отже, вдосконалений метод значно скоротив кількість помилкових класифікацій для всіх класів комах-шкідників у навчальному наборі. Він забезпечив суттєве підвищення точності класифікації, мінімізувавши кількість помилок і забезпечивши стабільні та надійні результати для кожного класу.

Висновки до розділу 4

У розділі наведено результати експериментальних досліджень запропонованого методу класифікації комах-шкідників у зерносховищах за допомогою донавченої моделі глибокого навчання. Детально описано розроблений прототип системи, який включає можливість вибору та запуску моделей для класифікації на основі нейронних мереж. Було створено спеціальний навчальний набір даних, який охоплює п'ять класів шкідників: борошняний кліщ, великий борошняний хрущак, зернова міль, кукурудзяний точильник та рисовий довгоносик. Кожен клас представлений збалансованою кількістю зображень для тренування і тестування, що дозволило забезпечити рівномірну підготовку моделі до розпізнавання кожного виду комах.

В процесі експериментів були використані два методи навчання: стандартний та вдосконалений. Проведено порівняння їхньої продуктивності за основними метриками для навчального набору, де результати стандартного методу показали Accuracy на рівні 0.9330 та F1 Score 0.9330, тоді як вдосконалений метод забезпечив підвищення цих показників до 0.9870 та 0.9870 відповідно. На тестовому наборі вдосконалений метод також перевершив стандартний, зокрема, Accuracy зросла з 0.9200 до 0.9740, а F1 Score – з 0.9200 до 0.9720. Це свідчить про покращену збалансованість моделі в аспекті точності та повноти для кожного класу.

Аналіз матриць помилок підтвердив, що вдосконалений метод значно зменшив кількість помилкових класифікацій для більшості класів. Наприклад, для класу «Рисовий довгоносик» кількість неправильних класифікацій у тестовому наборі знизилася з 5 до 2, а для класу «Зернова міль» – з 5 до 2, що підтверджує кращу адаптивність та здатність вдосконаленого методу до точнішої класифікації. Подібні покращення спостерігалися для інших класів, що забезпечило стабільні та точні результати при класифікації всіх типів комах-шкідників у навчальному та тестовому наборах.

Отримані результати свідчать про те, що вдосконалений метод навчання дає кращі показники для класифікації комах-шкідників у зерносховищах, оскільки він

забезпечує підвищену точність та стабільність класифікації. Використання цих результатів дозволяє покращити процес автоматизованого моніторингу шкідників та забезпечує більш надійне і швидке виявлення різних класів комах, що сприяє посиленню заходів захисту зерносховищ.

Загальні висновки

Метою кваліфікаційної роботи було підвищення якості класифікації комах-шкідників на зображеннях із зерносховищ завдяки розробці та впровадженню вдосконаленого методу на основі глибокого навчання. Для досягнення поставленої мети виконано наступні завдання:

– проведено ретельний аналіз предметної області, зокрема проблеми шкідників у зерносховищах, та здійснено огляд сучасних наукових підходів до класифікації шкідників, що дозволило визначити переваги та недоліки наявних рішень. На основі порівняння обрано модель YOLO як найбільш перспективну для застосування завдяки її здатності забезпечувати високі показники швидкодії та точності, що критично важливо для практичного застосування в реальному часі.

– вдосконалено метод класифікації шляхом донавчання моделі YOLOv8 на спеціально підготовленому датасеті, що відображає специфіку зображень із комахами-шкідниками у зерносховищах. Впроваджено комплекс удосконалень, серед яких регуляризаційні методи (dropout, weight decay) та оптимізація гіперпараметрів (швидкість навчання, розмір батчу, вибір оптимізатора), що забезпечило підвищення стабільності моделі та її здатності узагальнювати нові дані.

– розроблено метод класифікації, які були реалізовані у вигляді прототипної інформаційної системи. Прототип системи забезпечує автоматизацію процесів аналізу зображень шкідників та спрощує процес управління запасами зернових. Інтерфейс системи дозволяє користувачам здійснювати класифікацію з високим рівнем зручності та точності.

– для підтвердження підвищення якості класифікації проведено експерименти на розширеному тестовому наборі даних, що включає зображення комах-шкідників різних класів (борошняний кліщ, великий борошняний хрущак, зернова міль, кукурудзяний точильник та рисовий довгоносик). Результати експериментів підтвердили, що вдосконалена модель показує стабільно високі показники точності та чутливості, зокрема, значення Precision становить 0.9620–0.9800, а Recall – 0.9630–

0.9820. Це значно перевищує показники базової моделі та свідчить про суттєве покращення якості класифікації.

Експериментальні результати демонструють, що вдосконалений метод забезпечує не лише високі кількісні показники, але й підвищує стабільність та надійність класифікації в реальних умовах застосування. Зокрема, вдосконалена модель продемонструвала надійність у виявленні малих об'єктів і зберегла високі показники точності навіть за умов, максимально наближених до практичних. Це свідчить про її здатність адаптуватися до різноманітних типів даних і зберігати точність, що є важливою перевагою для застосування у системах автоматичного моніторингу.

Практичне значення отриманих результатів полягає у можливості автоматизувати процес ідентифікації шкідників, що сприятиме зниженню ризиків втрат зернових запасів та поліпшенню їхнього обліку в умовах зберігання. Застосування запропонованого методу дозволить значно зменшити залежність від людського фактору та прискорити процеси контролю, що особливо важливо у великих зерносховищах. Даний метод зможе забезпечити основу для подальшого розвитку автоматизованих систем, які можуть включати додаткові функції моніторингу та безпеки. Таким чином, запропонований метод створює перспективи для більш надійного управління запасами зерна, зберігаючи їх якість та зменшуючи ризики втрат у довгостроковій перспективі.

Перелік джерел

1. Estimating probability of visual detection of exotic pests and diseases in the grains industry–An expert elicitation approach / E. Arndt et al. *Frontiers in ecology and evolution*. 2022. Vol. 10. URL: https://doi.org/10.3389/fevo.2022.968436 (date of access: 17.11.2024).

2. Current progress on innovative pest detection techniques for stored cereal grains and thereof powders / L. Zhu et al. *Food chemistry*. 2022. Vol. 396. P. 133706. URL: https://doi.org/10.1016/j.foodchem.2022.133706 (date of access: 17.11.2024).

3. Gerken A. R., Campbell J. F. Spatial and temporal variation in stored-product insect pest distributions and implications for pest management in processing and storage facilities. *Annals of the entomological society of america*. 2021. URL: https://doi.org/10.1093/aesa/saab049 (date of access: 17.11.2024).

4. Environmental risk assessment for plant pests: a procedure to evaluate their impacts on ecosystem services / G. Gilioli et al. *Science of the total environment*. 2014. Vol. 468-469. P. 475–486. URL: https://doi.org/10.1016/j.scitotenv.2013.08.068 (date of access: 20.11.2024).

5. Research on and application of crop pest monitoring and early warning technology in china. *Frontiers of agricultural science and engineering*. 2021. P. 0. URL: https://doi.org/10.15302/j-fase-2021411 (date of access: 17.11.2024).

6. Chidege M. Y., Venkataramana P. B., Ndakidemi P. A. Enhancing food grains storage systems through insect pest detection and control measures for maize and beans: ensuring food security post-covid-19 tanzania. *Sustainability*. 2024. Vol. 16, no. 5. P. 1767. URL: https://doi.org/10.3390/su16051767 (date of access: 17.11.2024).

7. Management of stored grain pests in organic systems. *Penn State Extension | The Pennsylvania State University*. URL: https://extension.psu.edu/management-of-stored-grain-pests-in-organic-systems (date of access: 17.11.2024).

8. A review of deep learning in multiscale agricultural sensing / D. Wang et al. *Remote sensing*. 2022. Vol. 14, no. 3. P. 559. URL: https://doi.org/10.3390/rs14030559 (date of access: 17.11.2024).

9. Non-Destructive technologies for detecting insect infestation in fruits and vegetables under postharvest conditions: a critical review / A. A. Adedeji et al. *Foods*. 2020. Vol. 9, no. 7. P. 927. URL: https://doi.org/10.3390/foods9070927 (date of access: 17.11.2024).

10. *Entomology Journal | Journal of Entomology and Zoology Studies*. URL: https://www.entomoljournal.com/archives/2020/vol8issue3/PartAI/8-3-343-345.pdf (дата звернення: 17.11.2024).

11. Automated applications of acoustics for stored product insect detection, monitoring, and management / R. Mankin et al. *Insects*. 2021. Vol. 12, no. 3. P. 259. URL: https://doi.org/10.3390/insects12030259 (date of access: 17.11.2024).

12. Performance of a low-cost acoustic insect detector system with sitophilus oryzae (coleoptera: curculionidae) in stored grain and tribolium castaneum (coleoptera: tenebrionidae) in flour / R. W. Mankin et al. *Journal of economic entomology*. 2020. Vol. 113, no. 6. P. 3004–3010. URL: https://doi.org/10.1093/jee/toaa203 (date of access: 17.11.2024).

13. A method for acoustic storage pest detection and its challenges / M.-B. Christina et al. URL: https://www.openagrar.de/receive/openagrar_mods_00090546 (date of access: 17.11.2024).

14. Automated applications of acoustics for stored product insect detection, monitoring, and management / R. Mankin et al. *Insects*. 2021. Vol. 12, no. 3. P. 259. URL: https://doi.org/10.3390/insects12030259 (date of access: 17.11.2024).

15. Mobile computing for pest and disease management using spectral signature analysis: a review / N. N. Che'Ya et al. *Agronomy*. 2022. Vol. 12, no. 4. P. 967. URL: https://doi.org/10.3390/agronomy12040967 (date of access: 17.11.2024).

16. Gao Y., Xu M.-L. Recent research progress and outlook on applications of Raman scattering in pest detection and control: raman spectroscopy and its application in entomology. *Applied spectroscopy reviews*. 2024. P. 1–20. URL: https://doi.org/10.1080/05704928.2024.2407112 (date of access: 17.11.2024).

17. Differentiation of wheat diseases and pests based on hyperspectral imaging technology with a few specific bands / L. Yuan et al. *Phyton*. 2023. Vol. 92, no. 2. P. 611–628. URL: https://doi.org/10.32604/phyton.2022.023662 (date of access: 17.11.2024).

18. Detection of wheat Fusarium head blight using UAV-based spectral and image feature fusion / H. Zhang et al. *Frontiers in plant science*. 2022. Vol. 13. URL: https://doi.org/10.3389/fpls.2022.1004427 (date of access: 17.11.2024).

19. Identifying stored-grain insects using near-infrared spectroscopy / F. E. Dowell et al. *Journal of economic entomology*. 1999. Vol. 92, no. 1. P. 165–169. URL: https://doi.org/10.1093/jee/92.1.165 (date of access: 17.11.2024).

20. Zheng Z., Zhang C. Electronic noses based on metal oxide semiconductor sensors for detecting crop diseases and insect pests. *Computers and electronics in agriculture*. 2022. Vol. 197. P. 106988. URL: https://doi.org/10.1016/j.compag.2022.106988 (date of access: 17.11.2024).

21. FundurulicJorge M. S. Faria A., Faria J. M. S., Inácio M. L. Advances in electronic nose sensors for plant disease and pest detection. *MDPI*. URL: https://doi.org/10.3390/CSAC2023-14890 (date of access: 19.11.2024).Early detection of aphid infestation and insect-plant interaction assessment in wheat using a low-cost electronic nose (e-nose), near-infrared spectroscopy and machine learning modeling / S. Fuentes et al. *Sensors*. 2021. Vol. 21, no. 17. P. 5948. URL: https://doi.org/10.3390/s21175948 (date of access: 18.11.2024).

22. Biology-Informed inverse problems for insect pests detection using pheromone sensors. *index - INRAE - Institut national de recherche pour l'agriculture, l'alimentation et l'environnement*. URL: https://hal.inrae.fr/hal-04572831 (date of access: 19.11.2024).

23. A multi-level smart monitoring system by combining an e-nose and image processing for early detection of FAW pest in agriculture / S. A. R. M. Ahouandjinou et al. *SpringerLink*. URL: https://link.springer.com/chapter/10.1007/978-3-030-51051-0_2 (date of access: 19.11.2024).

24. Feasibility of detection of infested rice using an electronic nose / M. Zhou et al. *Journal of stored products research*. 2021. Vol. 92. P. 101805. URL: https://doi.org/10.1016/j.jspr.2021.101805 (date of access: 18.11.2024).

25. MacDougall S., Bayansal F., Ahmadi A. Emerging methods of monitoring volatile organic compounds for detection of plant pests and disease. *Biosensors*. 2022. Vol. 12, no. 4. P. 239. URL: https://doi.org/10.3390/bios12040239 (date of access: 18.11.2024).

26. Digital monitoring of grain conditions in large-scale bulk storage facilities based on spatiotemporal distributions of grain temperature / W. Wu et al. *Biosystems engineering*. 2021. Vol. 210. P. 247–260. URL: https://doi.org/10.1016/j.biosystemseng.2021.08.028 (date of access: 18.11.2024).

27. Liu J., Wang X. Plant diseases and pests detection based on deep learning: a review. *Plant methods*. 2021. Vol. 17, no. 1. URL: https://doi.org/10.1186/s13007-021-00722-9 (date of access: 18.11.2024).

28. A systematic review on automatic insect detection using deep learning / A. C. Teixeira et al. *Agriculture*. 2023. Vol. 13, no. 3. P. 713. URL: https://doi.org/10.3390/agriculture13030713 (date of access: 18.11.2024).

29. Chithambarathanu M., Jeyakumar M. K. Survey on crop pest detection using deep learning and machine learning approaches. *Multimedia tools and applications*. 2023. URL: https://doi.org/10.1007/s11042-023-15221-3 (date of access: 18.11.2024).

30. New trends in detection of harmful insects and pests in modern agriculture using artificial neural networks. a review / D. Popescu et al. *Frontiers in plant science*. 2023. Vol. 14. URL: https://doi.org/10.3389/fpls.2023.1268167 (date of access: 18.11.2024).

31. Insect and pest detection in stored grains: analysis of environmental factors and comparison of deep learning methods / D. P. R. et al. *Wseas transactions on environment and development*. 2022. Vol. 18. P. 759–768. URL: https://doi.org/10.37394/232015.2022.18.71 (date of access: 18.11.2024).

32. Classification and detection of insects from field images using deep learning for smart pest management: a systematic review / W. Li et al. *Ecological informatics*. 2021. Vol. 66. P. 101460. URL: https://doi.org/10.1016/j.ecoinf.2021.101460 (date of access: 18.11.2024).

33. Domingues T., Brandão T., Ferreira J. C. Machine learning for detection and prediction of crop diseases and pests: a comprehensive survey. *Agriculture*. 2022. Vol. 12,

no. 9. P. 1350. URL: https://doi.org/10.3390/agriculture12091350 (date of access: 18.11.2024).

34. Barbedo J. G. A. Detecting and classifying pests in crops using proximal images and machine learning: a review. *Ai*. 2020. Vol. 1, no. 2. P. 312–328. URL: https://doi.org/10.3390/ai1020021 (date of access: 18.11.2024).

35. Integrated Approaches for Managing Stored Grain Insect Pests of Rice. *NRRI Climate Resilient*. URL: https://icar-nrri.in/wp-content/uploads/2022/05/Book-NRRI-Climate-Resilient.pdf#page=335 (date of access: 19.11.2024).

36. REVIEW ON MAJOR STORAGE INSECT PESTS OF CEREALS AND PULSES - European Repository. *Welcome to European Repository - European Repository*. URL: http://go7publish.com/id/eprint/3391/ (date of access: 19.11.2024).

37. An aiot based smart agricultural system for pests detection / C.-J. Chen et al. *IEEE access*. 2020. Vol. 8. P. 180750–180761. URL: https://doi.org/10.1109/access.2020.3024891 (date of access: 18.11.2024).

38. An automatic system for pest recognition and forecasting / R. Wang et al. *Pest management science*. 2021. Vol. 78, no. 2. P. 711–721. URL: https://doi.org/10.1002/ps.6684 (date of access: 18.11.2024).

39. Prediction of pest insect appearance using sensors and machine learning. *MDPI*. URL: https://www.mdpi.com/1424-8220/21/14/4846 (date of access: 19.11.2024).

40. Faster R-CNN – Torchvision main documentation. *PyTorch*. URL: https://pytorch.org/vision/main/models/faster_rcnn.html (date of access: 19.11.2024).

41. PhD E. G. RetinaNet: advancing object detection in computer vision. *Medium*. URL: https://medium.com/@evertongomede/retinanet-advancing-object-detection-in-computer-vision-719ceb744308 (date of access: 19.11.2024).

42. SSD: single shot multibox detector / W. Liu et al. *SpringerLink*. URL: https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2 (date of access: 19.11.2024).

43. Potrimba P. What is mask R-CNN? The ultimate guide. *Roboflow Blog*. URL: https://blog.roboflow.com/mask-rcnn/ (date of access: 19.11.2024).

44. EfficientDet - NVIDIA Docs. *NVIDIA Docs*. URL: https://docs.nvidia.com/tao/tao-toolkit-archive/tao-30-2202/text/object_detection/efficientdet.html (date of access: 19.11.2024).

45. You only look once (YOLO): what is it?. *Data Science Courses | DataScientest*. URL: https://datascientest.com/en/you-only-look-once-yolo-what-is-it (date of access: 19.11.2024).

46. Terven J., Córdova-Esparza D., Romero-González J. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. Machine Learning and Knowledge Extraction. 2023. No. 5. P. 1680–1716.

47. Stochastic Gradient Descent: A Basic Explanation. URL: https://mohitmishra786687.medium.com/stochastic-gradient-descent-a-basic-explanation-cbddc63f08e0 (date of access: 26.10.2024).

48. Dropout in Neural Networks. URL: https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9 (date of access: 26.10.2024).

49. Layer-Wise Weight Decay for Deep Neural Networks. URL: https://link.springer.com/chapter/10.1007/978-3-319-75786-5_23 (date of access: 26.10.2024).

50. IP102-Dataset. *kaggle*. URL: http://ww.kaggle.com/datasets/rtlmhjbn/ip02-dataset?resource=download (date of access: 26.10.2024).

ДОДАТКИ

Додаток А

Скрипти навчання моделей

Код скрипту «train_and_save_metrics»

```
import json
import pandas as pd
import numpy as np
import os
import sys
from glob import glob
from ultralytics import YOLO
from sklearn.metrics import confusion_matrix, classification_report, precision_score, recall_score, f1_score, roc_auc_score
import torch

# Шляхи для моделі, даних та результатів
model_path = "yolov8n-cls.pt"
data_path = "F:/Test3"
metrics_path = "metrics.json"
confusion_matrix_path = "confusion_matrix.csv"

# Завантаження моделі
model = YOLO(model_path)

# Тренування моделі
model.train(data=data_path, epochs=10, imgsz=640)

# Збереження моделі у форматі .pt з назвою new_teach.pt
model.save("new_teach.pt")

# Експорт моделі у форматі ONNX з назвою new_teach.onnx
# Переключаємо модель в режим оцінки
model.model.eval()

# Створюємо вхідний тензор з потрібними розмірами
dummy_input = torch.randn(1, 3, 640, 640)

# Експортуємо модель у формат ONNX
torch.onnx.export(
    model.model,              # Модель для експорту
    dummy_input,              # Фіктивний вхід
    "new_teach.onnx",         # Шлях для збереження
    opset_version=12,         # Версія opset
    input_names=['input'],
    output_names=['output']
)

# Оцінка на валідаційному датасеті
val_results = model.val(data=data_path)

# Збір основних метрик після валідації
```

```
metrics = {
    "top1": getattr(val_results, "top1", None),
    "top5": getattr(val_results, "top5", None),
    "fitness": getattr(val_results, "fitness", None),
    "speed": getattr(val_results, "speed", {}).get("inference", None)
}

# Підготовка до обчислення матриці помилок та додаткових метрик
valid_dir = os.path.join(data_path, 'val')

# Перевірка існування папки
if not os.path.exists(valid_dir):
    print(f"Директорія {valid_dir} не існує. Перевірте шлях до валідаційних даних.")
    sys.exit(1)

class_dirs = [d for d in os.listdir(valid_dir) if os.path.isdir(os.path.join(valid_dir, d))]

# Перевірка наявності класів у валідаційній папці
if not class_dirs:
    print(f"У папці {valid_dir} не знайдено класів. Перевірте структуру ваших даних.")
    sys.exit(1)

class_to_idx = {cls_name: idx for idx, cls_name in enumerate(sorted(class_dirs))}
idx_to_class = {v: k for k, v in class_to_idx.items()}

image_paths = []
true_labels = []

for cls_name in class_dirs:
    cls_dir = os.path.join(valid_dir, cls_name)
    image_files = glob(os.path.join(cls_dir, '*.*'))
    if not image_files:
        print(f"У класі {cls_name} не знайдено зображень.")
        continue
    image_paths.extend(image_files)
    true_labels.extend([class_to_idx[cls_name]] * len(image_files))

# Перевірка наявності валідаційних зображень
if not image_paths:
    print("Не знайдено валідаційних зображень для обробки.")
    sys.exit(1)

# Перетворення на numpy масив
true_labels = np.array(true_labels)

# Передбачення на валідаційному наборі
results = model.predict(source=image_paths, imgsz=640, batch=32, verbose=False)

# Отримання передбачених міток
predicted_labels = np.array([r.probs.top1 for r in results])

# Обчислення матриці помилок
conf_matrix = confusion_matrix(true_labels, predicted_labels)
```

```
conf_matrix_df = pd.DataFrame(
    conf_matrix,
    index=[idx_to_class[i] for i in range(len(class_dirs))],
    columns=[idx_to_class[i] for i in range(len(class_dirs))]
)
conf_matrix_df.to_csv(confusion_matrix_path, index=True)

# Обчислення додаткових метрик
precision = precision_score(true_labels, predicted_labels, average='weighted')
recall = recall_score(true_labels, predicted_labels, average='weighted')
f1 = f1_score(true_labels, predicted_labels, average='weighted')

# Отримання ймовірностей
probabilities = np.array([r.probs.data.cpu().numpy() for r in results])

# Обчислення AUC
try:
    n_classes = len(class_dirs)
    if n_classes == 2:
        # Двокласова класифікація
        y_score = probabilities[:, 1]
        # Переконаємося, що мітки класів є 0 та 1
        if set(true_labels) != {0, 1}:
            print(f"Мітки класів перед перекодуванням: {set(true_labels)}")
            unique_labels = sorted(set(true_labels))
            label_encoder = {old_label: new_label for new_label, old_label in enumerate(unique_labels)}
            true_labels = np.array([label_encoder[label] for label in true_labels])
            predicted_labels = np.array([label_encoder[label] for label in predicted_labels])
        auc = roc_auc_score(true_labels, y_score)
    else:
        # Багатокласова класифікація
        auc = roc_auc_score(true_labels, probabilities, multi_class='ovr', average='weighted')
except Exception as e:
    auc = None
    print(f"Не вдалося обчислити AUC: {e}")

# Додавання додаткових метрик до словника metrics
metrics['precision'] = precision
metrics['recall'] = recall
metrics['f1_score'] = f1
metrics['auc'] = auc

# Збереження метрик у файл JSON
with open(metrics_path, "w") as f:
    json.dump(metrics, f, indent=4)

# Обчислення детального звіту по метрикам
report = classification_report(
    true_labels,
    predicted_labels,
    target_names=[idx_to_class[i] for i in range(len(class_dirs))],
    output_dict=True
)
```

```
# Збереження детального звіту у файл JSON
with open('classification_report.json', 'w') as f:
    json.dump(report, f, indent=4, ensure_ascii=False)

print("Модель збережено як 'new_teach.pt' та експортовано у 'new_teach.onnx'.")
print("Метрики збережено у 'metrics.json' та 'classification_report.json', матриця помилок збережена у 'confusion_matrix.csv'.")
```

Код скрипту «train_and_save_metrics_2»

```
import json
import pandas as pd
import numpy as np
import os
from ultralytics import YOLO
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, roc_auc_score
from glob import glob
import sys
import torch
from torch import nn

# Шляхи для моделі, даних та результатів
model_path = "yolov8n-cls.pt"  # Використовуємо модель
data_path = "F:/Test4"
metrics_output = []

learning_rates = [0.001, 0.0001]
batch_sizes = [16, 32, 64]
optimizers = ['SGD']
epochs = 10

# Кращі результати
best_metrics = None
best_params = None
best_fitness = -np.inf

# Функція для додавання Dropout
def add_dropout(model, p=0.2):
    for name, module in model.model.named_children():
        if isinstance(module, nn.Sequential):
            new_layers = []
            for layer in module:
                new_layers.append(layer)
                if isinstance(layer, (nn.Conv2d, nn.Linear)):
                    new_layers.append(nn.Dropout(p=p))
            setattr(model.model, name, nn.Sequential(*new_layers))
        elif isinstance(module, (nn.Conv2d, nn.Linear)):
            new_module = nn.Sequential(
                module,
                nn.Dropout(p=p)
            )
            setattr(model.model, name, new_module)
```

```
# Перебір всіх комбінацій гіперпараметрів
for lr in learning_rates:
    for bs in batch_sizes:
        for opt in optimizers:
            print(f"\nТренування моделі з параметрами: learning_rate={lr}, batch_size={bs}, optimizer={opt}")

            # Завантаження базової моделі
            model = YOLO(model_path)

            # Додавання Dropout шарів
            add_dropout(model, p=0.2)

            # Налаштування параметрів навчання
            model.overrides['lr0'] = lr  # початкова швидкість навчання
            model.overrides['batch'] = bs  # розмір батчу
            model.overrides['optimizer'] = opt  # оптимізатор без .lower()

            # Додавання регуляризації через weight_decay
            model.overrides['weight_decay'] = 0.0005

            # Тренування моделі
            model.train(data=data_path, epochs=epochs, imgsz=640)

            # Оцінка на валідаційному датасеті
            val_results = model.val(data=data_path)

            # Збір основних метрик після валідації
            metrics = {
                "learning_rate": lr,
                "batch_size": bs,
                "optimizer": opt,
                "top1": getattr(val_results, "top1", None),
                "top5": getattr(val_results, "top5", None),
                "fitness": getattr(val_results, "fitness", None),
                "speed": getattr(val_results, "speed", {}).get("inference", None)
            }

            # Підготовка до обчислення додаткових метрик
            # Отримання списку валідаційних зображень та їх справжніх міток
            valid_dir = os.path.join(data_path, 'val')  # Змініть на 'valid', якщо ваша папка називається інакше
            if not os.path.exists(valid_dir):
                print(f"Директорія {valid_dir} не існує. Перевірте шлях до валідаційних даних.")
                sys.exit(1)

            class_dirs = [d for d in os.listdir(valid_dir) if os.path.isdir(os.path.join(valid_dir, d))]
            if not class_dirs:
                print(f"У папці {valid_dir} не знайдено класів. Перевірте структуру ваших даних.")
                sys.exit(1)

            class_to_idx = {cls_name: idx for idx, cls_name in enumerate(sorted(class_dirs))}
            idx_to_class = {v: k for k, v in class_to_idx.items()}
```

```
image_paths = []
true_labels = []

for cls_name in class_dirs:
    cls_dir = os.path.join(valid_dir, cls_name)
    image_files = glob(os.path.join(cls_dir, '*.*'))
    if not image_files:
        print(f"У класі {cls_name} не знайдено зображень.")
        continue
    image_paths.extend(image_files)
    true_labels.extend([class_to_idx[cls_name]] * len(image_files))

if not image_paths:
    print("Не знайдено валідаційних зображень для обробки.")
    sys.exit(1)

# Перетворення на numpy масив
true_labels = np.array(true_labels)

# Передбачення на валідаційному наборі
results = model.predict(source=image_paths, imgsz=640, batch=bs, verbose=False)

# Отримання передбачених міток
predicted_labels = np.array([r.probs.top1 for r in results])

# Обчислення матриці помилок
conf_matrix = confusion_matrix(true_labels, predicted_labels)
conf_matrix_df = pd.DataFrame(
    conf_matrix,
    index=[idx_to_class[i] for i in range(len(class_dirs))],
    columns=[idx_to_class[i] for i in range(len(class_dirs))]
)
# Збереження матриці помилок у файл CSV (опціонально)
conf_matrix_path = f"confusion_matrix_lr{lr}_bs{bs}_opt{opt}.csv"
conf_matrix_df.to_csv(conf_matrix_path, index=True)

# Обчислення додаткових метрик
precision = precision_score(true_labels, predicted_labels, average='weighted')
recall = recall_score(true_labels, predicted_labels, average='weighted')
f1 = f1_score(true_labels, predicted_labels, average='weighted')

# Отримання ймовірностей
probabilities = np.array([r.probs.data.cpu().numpy() for r in results])

# Обчислення AUC
try:
    n_classes = len(class_dirs)
    if n_classes == 2:
        # Двокласова класифікація
        y_score = probabilities[:, 1]
        # Переконаємося, що мітки класів є 0 та 1
        if set(true_labels) != {0, 1}:
```

```
                unique_labels = sorted(set(true_labels))
                label_encoder = {old_label: new_label for new_label, old_label in enumerate(unique_labels)}
                true_labels = np.array([label_encoder[label] for label in true_labels])
                predicted_labels = np.array([label_encoder[label] for label in predicted_labels])
            auc = roc_auc_score(true_labels, y_score)
        else:
            # Багатокласова класифікація
            auc = roc_auc_score(true_labels, probabilities, multi_class='ovr', average='weighted')
    except Exception as e:
        auc = None
        print(f"Не вдалося обчислити AUC: {e}")

    # Додавання метрик до словника metrics
    metrics['precision'] = precision
    metrics['recall'] = recall
    metrics['f1_score'] = f1
    metrics['auc'] = auc

    # Збереження метрик для порівняння
    metrics_output.append(metrics)

    # Оновлення найкращих метрик
    if val_results.fitness > best_fitness:
        best_fitness = val_results.fitness
        best_metrics = metrics
        best_params = {'learning_rate': lr, 'batch_size': bs, 'optimizer': opt}

        # Встановлення базової назви для найкращої моделі
        base_model_name = "new_teach"

        # Збереження найкращої моделі у форматі .pt
        model.save(f"{base_model_name}.pt")

        # Експорт найкращої моделі у формат ONNX
        # Переключаємо модель в режим оцінки
        model.model.eval()

        # Створюємо фіктивний вхідний тензор з потрібними розмірами
        dummy_input = torch.randn(1, 3, 640, 640)

        # Експортуємо модель у формат ONNX з бажаною назвою файлу
        torch.onnx.export(
            model.model,              # Модель для експорту
            dummy_input,              # Фіктивний вхід
            f"{base_model_name}.onnx",  # Шлях для збереження
            opset_version=12,         # Версія opset
            input_names=['input'],
            output_names=['output']
        )

        print(f"Найкраща модель збережена у форматі .pt як: {base_model_name}.pt")
        print(f"Найкраща модель збережена у форматі .onnx як: {base_model_name}.onnx")
```

```
            print(f"Поточні метрики: {metrics}")
# Збереження всіх метрик у файл
with open('hyperparameter_optimization_results.json', 'w') as f:
    json.dump(metrics_output, f, indent=4)

print(f"\nНайкращі параметри: {best_params}")
print(f"Найкращі метрики: {best_metrics}")
```

Додаток Б

Лістинги програмного забезпечення

Лістинг 1. Код класу «TrainingAndStorageForm.cs»

```
using ClassificationInsectApp.AppCode;
using ClassificationInsectApp.Providers;
using Newtonsoft.Json.Linq;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Python.Runtime;

namespace ClassificationInsectApp {
  public partial class TrainingAndStorageForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private ModelsProvider _ModelsProvider = new ModelsProvider();
    private List<Models> _ModelsList = new List<Models>();
    private bool _IsModelTrain = false;
    private string _ModelPath = "new_teach.onnx";
    private PathSettingProvider _PathSettingProvider = new PathSettingProvider();
    private PathSetting _selPathSetting = new PathSetting();


    public TrainingAndStorageForm() {
      InitializeComponent();
      DataLoad();
      Load();
    }

    private void Load() {
      _selPathSetting = _PathSettingProvider.LoadCredentials();
    }

    private async void StandartTrainBtn_Click(object sender, EventArgs e) {
      DisableBtn();
      await Task.Run(() => {
        try {
          // Використання Invoke для оновлення тексту в основному потоці
          this.Invoke((MethodInvoker)(() => {
            ResultTBox.Text = "Тренування моделі....\r\n";
            ResultTBox.Text += "Дочекайтесь закінчення процесу.\r\n";
          }));
```

```
    // Запуск Python скрипта для навчання та збереження метрик
    var psi = new ProcessStartInfo {
      FileName = _selPathSetting.PythonFileName,
      Arguments = _selPathSetting.StandartScript,  // Шлях до Python скрипта
      RedirectStandardOutput = true,
      UseShellExecute = false,
      CreateNoWindow = true
    };

    using (var process = Process.Start(psi)) {
      using (var reader = process.StandardOutput) {
        string result = reader.ReadToEnd();
        Console.WriteLine(result); // Вивід результату навчання у консоль
      }
    }

    // Другий блок `try-catch` для читання метрик з JSON файлу
    try {
      // Читання метрик з JSON файлу
      string metricsContent = File.ReadAllText("metrics.json");
      var metrics = JsonConvert.DeserializeObject<Dictionary<string, object>>(metricsContent);

      // Формування повідомлення з усіма метриками
      string message = "Метрики моделі:\r\n";
      foreach (var metric in metrics) {
        // Перевірка типу метрики (числове значення чи вкладений об'єкт)
        if (metric.Value is double || metric.Value is float || metric.Value is decimal) {
          message += $"{metric.Key}: {Convert.ToDouble(metric.Value):F4}\r\n";
        } else if (metric.Value is JObject nestedMetrics) {
          message += $"{metric.Key}:\r\n";
          foreach (var nestedMetric in nestedMetrics) {
            message += $"  {nestedMetric.Key}: {Convert.ToDouble(nestedMetric.Value):F4}\r\n";
          }
        }
      }
      // Виведення метрик у TextBox у головному потоці
      this.Invoke((MethodInvoker)(() => ResultTBox.Text += message));
      _IsModelTrain = true;

    } catch (Exception ex) {
      this.Invoke((MethodInvoker)(() => MessageBox.Show($"Помилка: {ex.Message}")));
    }
  } catch (Exception ex) {
    this.Invoke((MethodInvoker)(() => MessageBox.Show($"Загальна помилка: {ex.Message}")));
  }
});

Runtime.PythonDLL = _selPathSetting.PythonDLL;

// Ініціалізуємо Python середовище
PythonEngine.Initialize();
```

```
    // Викликаємо Python код
    using (Py.GIL())  // GIL потрібен для потокової роботи Python
    {
     dynamic yolo = Py.Import("ultralytics"); // Імпортуємо бібліотеку YOLOv8
     dynamic model = yolo.YOLO("new_teach.pt"); // Завантажуємо модель
     model.export(format: "onnx"); // Експортуємо модель у ONNX
    }

    // Завершуємо Python середовище
    PythonEngine.Shutdown();

    MessageBox.Show("Модель успішно експортована в ONNX формат!");

    EnableBtn();
   }


  private async void CustomTrainBtn_Click(object sender, EventArgs e) {
    DisableBtn();
    await Task.Run(() => {
     try {
      // Очищення та встановлення тексту в процесі тренування
      this.Invoke((MethodInvoker)(() => {
       ResultTBox.Text = "Тренування моделі....\r\n";
       ResultTBox.Text += "Дочекайтесь закінчення процесу.\r\n";
      }));

      // Запуск Python скрипта для навчання та збереження метрик
      var psi = new ProcessStartInfo {
       FileName = _selPathSetting.PythonFileName,
       Arguments = _selPathSetting.CustomScript,
       RedirectStandardOutput = true,
       UseShellExecute = false,
       CreateNoWindow = true
      };

      using (var process = Process.Start(psi)) {
       using (var reader = process.StandardOutput) {
        string result = reader.ReadToEnd();
        Console.WriteLine(result); // Вивід результату навчання у консоль
       }
      }

      // Читання та обробка метрик з JSON файлу
      try {
       // Читання JSON файлу
       string metricsContent = File.ReadAllText("hyperparameter_optimization_results.json");
       var metricsList = JsonConvert.DeserializeObject<List<Dictionary<string,
object>>>(metricsContent);

       // Знаходження найкращих параметрів
       Dictionary<string, object> bestMetrics = null;
       double bestScore = double.MinValue;
```

```
        foreach (var metrics in metricsList) {
          // Перевірка наявності ключа "fitness" для обчислення оцінки
          if (metrics.ContainsKey("fitness")) {
            double currentScore = Convert.ToDouble(metrics["fitness"]);
            if (currentScore > bestScore) {
              bestScore = currentScore;
              bestMetrics = metrics;
            }
          }
        }

        // Формування повідомлення з усіма метриками для кожного конфігураційного набору
        string message = "Результати тренувань з різними параметрами:\r\n";
        foreach (var metrics in metricsList) {
          message += "--------------------------------\r\n";
          message += $"Learning Rate: {metrics.GetValueOrDefault("learning_rate", "N/A")}\r\n";
          message += $"Batch Size: {metrics.GetValueOrDefault("batch_size", "N/A")}\r\n";
          message += $"Optimizer: {metrics.GetValueOrDefault("optimizer", "N/A")}\r\n";

          // Відображення метрик, обробка відсутніх ключів
          foreach (var metric in metrics) {
            if (metric.Key != "learning_rate" && metric.Key != "batch_size" && metric.Key !=
"optimizer") {
              message += $"{metric.Key}: {Convert.ToDouble(metric.Value):F4}\r\n";
            }
          }
          message += "--------------------------------\r\n";
        }

        // Додавання найкращих параметрів з надписом
        if (bestMetrics != null) {
          message += "\r\nНайкращі параметри моделі:\r\n";
          message += "--------------------------------\r\n";
          message += $"Learning Rate: {bestMetrics.GetValueOrDefault("learning_rate", "N/A")}\r\n";
          message += $"Batch Size: {bestMetrics.GetValueOrDefault("batch_size", "N/A")}\r\n";
          message += $"Optimizer: {bestMetrics.GetValueOrDefault("optimizer", "N/A")}\r\n";

          foreach (var metric in bestMetrics) {
            if (metric.Key != "learning_rate" && metric.Key != "batch_size" && metric.Key !=
"optimizer") {
              message += $"{metric.Key}: {Convert.ToDouble(metric.Value):F4}\r\n";
            }
          }
          message += "--------------------------------\r\n";
        }

        // Виведення всіх метрик у TextBox
        this.Invoke((MethodInvoker)(() => ResultTBox.Text = message));
        _IsModelTrain = true;
      } catch (Exception ex) {
        this.Invoke((MethodInvoker)(() => MessageBox.Show($"Помилка при читанні метрик:
{ex.Message}")));
```

```
        }
      } catch (Exception ex) {
        this.Invoke((MethodInvoker)(() => MessageBox.Show($"Загальна помилка: {ex.Message}")));
      }
    });
    EnableBtn();
    Runtime.PythonDLL = _selPathSetting.PythonDLL;

    // Ініціалізуємо Python середовище
    PythonEngine.Initialize();

    // Викликаємо Python код
    using (Py.GIL())  // GIL потрібен для потокової роботи Python
    {
      dynamic yolo = Py.Import("ultralytics"); // Імпортуємо бібліотеку YOLOv8
      dynamic model = yolo.YOLO("new_teach.pt"); // Завантажуємо модель
      model.export(format: "onnx"); // Експортуємо модель у ONNX
    }

    // Завершуємо Python середовище
    PythonEngine.Shutdown();

    MessageBox.Show("Модель успішно експортована в ONNX формат!");

    EnableBtn();
  }


  private void SaveBtn_Click(object sender, EventArgs e) {
    if (IsDataEnteringCorrect()) {
      // Знаходження наступного унікального ModelsId
      int modelsId = (_ModelsList.Count > 0) ? _ModelsList.Max(m => m.ModelsId) + 1 : 1;

      // Копіювання моделі та отримання нового шляху
      string newPathName = CopyModelToTeachFolder(_ModelPath);

      // Перевірка, чи успішно скопійовано
      if (newPathName != null) {
        // Збереження інформації про модель з новим шляхом
        Models models = new Models(modelsId, ModelsNameTBox.Text, DateTime.Now, newPathName,
DescriptionTBox.Text);
        _ModelsList.Add(models);
        _ModelsProvider.SaveModels(_ModelsList);

        // Оновлення відображення даних та очищення контролів
        DataLoad();
        ClearAllControls();
      } else {
        MessageBox.Show("Не вдалося скопіювати модель. Перевірте шлях до початкового файлу.");
      }
    }
  }
```

```
    private void Clear_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void ModelsGView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && e.ColumnIndex == 5) {
        // Отримуємо ModelsId для вибраного рядка
        int modelsId = Convert.ToInt32(ModelsGView[0, e.RowIndex].Value);

        // Підтвердження видалення
        var confirmResult = MessageBox.Show("Ви дійсно хочете видалити цей запис?",
"Підтвердження видалення", MessageBoxButtons.YesNo);
        if (confirmResult == DialogResult.Yes) {
          // Видаляємо модель із списку
          var modelToRemove = _ModelsList.FirstOrDefault(m => m.ModelsId == modelsId);
          if (modelToRemove != null) {
            _ModelsList.Remove(modelToRemove);

            // Зберігаємо оновлений список
            _ModelsProvider.SaveModels(_ModelsList);

            // Оновлюємо дані у DataGridView
            LoadDataInModelsGView(_ModelsList);
            MessageBox.Show("Запис успішно видалено.");
          }
        }
      }
    }


    private void DataLoad() {
      int firstRowIndex = 0;
      if (ModelsGView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = ModelsGView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _ModelsList = _ModelsProvider.GetAllData();
        LoadDataInModelsGView(_ModelsList);
        if (_selectedRowIndex == ModelsGView.Rows.Count) {
          _selectedRowIndex = ModelsGView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          ModelsGView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          ModelsGView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch (Exception ex) {
        MessageBox.Show(ex.ToString());
      }
```

```
}

private string GenerateFileName() {
  DateTime now = DateTime.Now;
  string fileName = string.Format("{0}_{1}_{2}_{3}_{4}_{5}",
      now.Year, now.Month, now.Day, now.Hour, now.Minute, now.Second);

  return fileName;
}

// Метод копіювання, який повертає шлях до нового файлу
private string CopyModelToTeachFolder(string modelPath) {
  // Перевірка існування файлу
  if (!File.Exists(modelPath)) {
    Console.WriteLine("Файл моделі не знайдено.");
    return null;
  }

  // Створення папки teach, якщо вона не існує
  string destinationDirectory = "teach";
  if (!Directory.Exists(destinationDirectory)) {
    Directory.CreateDirectory(destinationDirectory);
  }

  // Генерація нового імені файлу з розширенням .onnx
  string newFileName = GenerateFileName() + ".onnx";
  string destinationPath = Path.Combine(destinationDirectory, newFileName);

  // Копіювання файлу з новою назвою
  File.Copy(modelPath, destinationPath);

  Console.WriteLine($"Модель скопійовано до {destinationPath}");
  return destinationPath; // Повернення шляху до нового файлу
}

private void DisableBtn() {
  StandartTrainBtn.Enabled = false;
  SaveBtn.Enabled = false;
  CustomTrainBtn.Enabled = false;
}

private void EnableBtn() {
  StandartTrainBtn.Enabled = true;
  SaveBtn.Enabled = true;
  CustomTrainBtn.Enabled = true;
}

private void LoadDataInModelsGView(List<Models> ModelsList) {
  ModelsGView.DataSource = null;
  ModelsGView.Columns.Clear();
  ModelsGView.AutoGenerateColumns = false;
  ModelsGView.RowHeadersVisible = false;
```

```
  ModelsGView.DataSource = ModelsList;

  DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
  DetailIdColumn.DataPropertyName = "ModelsId";
  ModelsGView.Columns.Add(DetailIdColumn);
  ModelsGView.Columns[0].Visible = false;

  DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
  numberColumn.HeaderText = "№ п/п";
  numberColumn.DataPropertyName = "Number";
  numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
  numberColumn.Width = 60;
  ModelsGView.Columns.Add(numberColumn);

  DataGridViewColumn ModelsNameColumn = new DataGridViewTextBoxColumn();
  ModelsNameColumn.HeaderText = "Назва моделі";
  ModelsNameColumn.DataPropertyName = "ModelsName";
  ModelsNameColumn.Width = 250;
  ModelsGView.Columns.Add(ModelsNameColumn);

  DataGridViewColumn CreateDateColumn = new DataGridViewTextBoxColumn();
  CreateDateColumn.HeaderText = "Дата створення";
  CreateDateColumn.DataPropertyName = "CreateDate";
  CreateDateColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
  CreateDateColumn.Width = 150;
  ModelsGView.Columns.Add(CreateDateColumn);

  DataGridViewColumn FileModelColumn = new DataGridViewTextBoxColumn();
  FileModelColumn.HeaderText = "Шлях до файлу";
  FileModelColumn.DataPropertyName = "FileModel";
  FileModelColumn.Width = 250;
  ModelsGView.Columns.Add(FileModelColumn);

  DataGridViewButtonColumn IsResidesBtn = new DataGridViewButtonColumn();
  IsResidesBtn.HeaderText = "Видалити";
  IsResidesBtn.Text = "Видалити";
  IsResidesBtn.UseColumnTextForButtonValue = true;
  IsResidesBtn.ToolTipText = "Видалити";
  IsResidesBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
  ModelsGView.Columns.Add(IsResidesBtn);


  for (int i = 0; i < ModelsGView.Columns.Count; i++) {
    ModelsGView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
  }

 }

 private void ClearAllControls() {
  ModelsNameTBox.Text = String.Empty;
  DescriptionTBox.Text = String.Empty;
  ResultTBox.Text = String.Empty;
  _IsModelTrain = false;
```

```
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(ModelsNameTBox.Text)) {
        ModelsNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ModelsNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (!_IsModelTrain) {
        MessageBox.Show("Модель поки не навчено!", "Увага");
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг 2. Код класу «TestForm.cs»

```
using ClassificationInsectApp.Providers;
using Compunet.YoloV8;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace ClassificationInsectApp {
  public partial class TestForm : Form {
    private ModelsProvider _ModelsProvider = new ModelsProvider();
    private Models _SelectedModels = new Models();
    private List<Models> _ModelsList = new List<Models>();
    private bool _IsModelsLoad = false;
    private string _filter = "Image Files(*.jpg; *.jpeg; *.gif; *.bmp; *.png)|*.jpg; *.jpeg; *.gif; *.bmp;
*.png";
    private string _FileName = "";

    public TestForm() {
      InitializeComponent();
      LoadAllData();
    }

    private void LoadAllData() {
      _ModelsList = _ModelsProvider.GetAllData();
      ModelsCBox.DataSource = _ModelsList;
      ModelsCBox.ValueMember = "ModelsId";
      ModelsCBox.DisplayMember = "ModelsName";
      _IsModelsLoad = true;
      ModelsCBox_SelectedValueChanged(ModelsCBox, EventArgs.Empty);
    }

    private void LoadBtn_Click(object sender, EventArgs e) {
```

```
      OpenFileDialog openFileDialog1 = new OpenFileDialog();
      openFileDialog1.Filter = _filter;
      if (openFileDialog1.ShowDialog() == DialogResult.OK) {
        PredictPictureBox.SizeMode = PictureBoxSizeMode.Zoom;
        PredictPictureBox.Image = new Bitmap(openFileDialog1.FileName);
        _FileName = openFileDialog1.FileName;
        FileNameTBox.Text = _FileName;
      }
    }

    private void PredictBtn_Click(object sender, EventArgs e) {
      if (_SelectedModels != null && !string.IsNullOrEmpty(_SelectedModels.FileModel) &&
!string.IsNullOrEmpty(_FileName)) {
        // Завантажуємо модель із файлу ONNX
        var modelPath = _SelectedModels.FileModel; // шлях до ONNX файлу моделі
        try {
          using var predictor = new YoloPredictor(modelPath);

          // Виконуємо передбачення
          var result = predictor.Classify(_FileName);
          // Виводимо результат у TextBox
          ResultTBox.Text = $"Результат класифікації: {result.ToString()}";
        } catch (Exception ex) {
          MessageBox.Show($"Помилка при прогнозуванні: {ex.Message}");
        }
      } else {
        MessageBox.Show("Будь ласка, завантажте зображення та виберіть модель.");
      }
    }

    private void ModelsCBox_SelectedValueChanged(object sender, EventArgs e) {
      if (_IsModelsLoad) {
        int selectedModelId = Convert.ToInt32(ModelsCBox.SelectedValue);
        _SelectedModels = _ModelsList.FirstOrDefault(m => m.ModelsId == selectedModelId);

        if (_SelectedModels != null) {
          // Виводимо інформацію про обрану модель
          DescriptionTBox.Text = _SelectedModels.Description;
          ResultTBox.Text = $"Обрано модель: {_SelectedModels.ModelsName}\nШлях до моделі:
{_SelectedModels.FileModel}";
        }
      }
    }
  }
}
```

Лістинг 3. Код класу «SettingsForm.cs»

```
using ClassificationInsectApp.AppCode;
using ClassificationInsectApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Security.Cryptography.Xml;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ClassificationInsectApp {
  public partial class SettingsForm : Form {
    private PathSettingProvider _PathSettingProvider = new PathSettingProvider();
    private ValidationMy _validation = new ValidationMy();
    private PathSetting _selPathSetting = new PathSetting();

    public SettingsForm() {
      InitializeComponent();
      Load();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        // Створення екземпляру класу з даними для збереження
        var credentials = new PathSetting {
          PythonFileName = PythonFileNameTBox.Text,
          StandartScript = StandartScriptTBox.Text,
          CustomScript = CustomScriptTBox.Text,
          PythonDLL = PythonDLLTBox.Text,
        };
        _PathSettingProvider.SaveCredentials(credentials);
        MessageBox.Show("Дані збережено!");
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void Load() {
      _selPathSetting = _PathSettingProvider.LoadCredentials();
      if (_selPathSetting != null) {
        PythonFileNameTBox.Text = _selPathSetting.PythonFileName;
        StandartScriptTBox.Text = _selPathSetting.StandartScript;
        CustomScriptTBox.Text = _selPathSetting.CustomScript;
        PythonDLLTBox.Text = _selPathSetting.PythonDLL;
      }
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(PythonFileNameTBox.Text)) {
        PythonFileNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PythonFileNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
```

```
      }
      if (_validation.IsDataEntering(StandartScriptTBox.Text)) {
        StandartScriptValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        StandartScriptValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(CustomScriptTBox.Text)) {
        CustomScriptValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CustomScriptValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг 4. Код класу «PathSettingProvider»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;

namespace ClassificationInsectApp.Providers {
  internal class PathSettingProvider {
    // Метод для збереження даних у XML файлі
    string _FileName = "settings_py.xml";
    public void SaveCredentials(PathSetting credentials) {
      var serializer = new XmlSerializer(typeof(PathSetting));
      using (var stream = new FileStream(_FileName, FileMode.Create)) {
        serializer.Serialize(stream, credentials);
      }
    }

    // Метод для завантаження даних з XML файлу
    public PathSetting LoadCredentials() {
      // Перевірка, чи існує файл
      if (!File.Exists(_FileName)) {
        MessageBox.Show($"Файл не знайдено: {_FileName}", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        return null; // або повернути новий об'єкт PathSetting з параметрами за замовчуванням
      }

      var serializer = new XmlSerializer(typeof(PathSetting));
      using (var stream = new FileStream(_FileName, FileMode.Open)) {
        return (PathSetting)serializer.Deserialize(stream);
      }
    }

  }
```

```
}

public class PathSetting {
  private string _PythonFileName;
  private string _StandartScript;
  private string _CustomScript;
  private string _PythonDLL;

  public PathSetting() {
    _PythonFileName = String.Empty;
    _StandartScript = String.Empty;
    _CustomScript = String.Empty;
    _PythonDLL = String.Empty;
  }
  public string PythonFileName {
    set { _PythonFileName = value; }
    get { return _PythonFileName; }
  }
  public string StandartScript {
    set { _StandartScript = value; }
    get { return _StandartScript; }
  }
  public string CustomScript {
    set { _CustomScript = value; }
    get { return _CustomScript; }
  }
  public string PythonDLL {
    set { _PythonDLL = value; }
    get { return _PythonDLL; }
  }
}
```

Лістинг 5. Код класу «ModelsProvider»

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Formatters.Binary;
using System.Text;
using System.Threading.Tasks;
using System.Text.Json;
using System.Runtime.InteropServices.JavaScript;
using System.Xml.Linq;

namespace ClassificationInsectApp.Providers {
    internal class ModelsProvider {
   // Збереження списку моделей у файл у форматі JSON
   public void SaveModels(List<Models> listModels) {
     // Використання потоку для запису JSON з режимом Create
     using (FileStream fs = new FileStream("Models.dat", FileMode.Create, FileAccess.Write)) {
       JsonSerializer.Serialize(fs, listModels);
       Console.WriteLine("Об'єкт серіалізовано у JSON формат.");
     }
```

```
    }


    // Отримання даних з файлу у вигляді списку моделей
    public List<Models> GetAllData() {
      // Перевірка, чи існує файл і чи він не порожній
      if (!File.Exists("Models.dat") || new FileInfo("Models.dat").Length == 0) {
        return new List<Models>(); // Повертаємо порожній список, якщо файл не існує або порожній
      }

      List<Models> allModelsList = new List<Models>();
      using (FileStream fs = new FileStream("Models.dat", FileMode.Open, FileAccess.Read)) {
        // Десеріалізація JSON в список моделей
        allModelsList = JsonSerializer.Deserialize<List<Models>>(fs) ?? new List<Models>();
      }

      // Присвоєння номера кожній моделі
      for (int i = 0; i < allModelsList.Count; i++) {
        allModelsList[i].Number = i + 1;
      }

      return allModelsList;
    }

  }
}

public class Models {
  private int _Number;
  private int _ModelsId;
  private string _ModelsName;
  private DateTime _CreateDate;
  private string _FileModel;
  private string _Description;
  private string _Message;

  public Models() {
    _Number = 0;
    _ModelsId = 0;
    _FileModel = string.Empty;
    _CreateDate = new DateTime();
    _ModelsName = string.Empty;
    _Description = string.Empty;
    _Message = string.Empty;
  }
  public Models(int ModelsId, string ModelsName, DateTime CreateDate, string FileModel, string
Description) {
    _ModelsId = ModelsId;
    _ModelsName = ModelsName;
    _CreateDate = CreateDate;
    _FileModel = FileModel;
    _Description = Description;
  }
```

```
  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int ModelsId {
    set { _ModelsId = value; }
    get { return _ModelsId; }
  }
  public string FileModel {
    set { _FileModel = value; }
    get { return _FileModel; }
  }
  public DateTime CreateDate {
    set { _CreateDate = value; }
    get { return _CreateDate; }
  }
  public string ModelsName {
    set { _ModelsName = value; }
    get { return _ModelsName; }
  }
  public string Description {
    set { _Description = value; }
    get { return _Description; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Додаток В

Світлини наукових публікацій, виконаних при роботі над кваліфікаційною роботою магістра

(*скріншоти титульної сторінки, сторінки змісту та всіх сторінок із публікацією*)

Наукова публікація:

Бармак О.В., Пострибайло В.О. Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання. Збірник наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2024». - Хмельницький, 2024. - С. 425-427.

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ

за матеріалами XVI Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2024»

15-16 листопада 2024

Хмельницький 2024

УДК 004.4

Пострибайло В.О., Бармак О.В.

Хмельницький національний університет

МЕТОД КЛАСИФІКАЦІЇ КОМАХ-ШКІДНИКІВ У ЗЕРНОСХОВИЩАХ ЗА МОДЕЛЛЮ ГЛИБОКОГО НАВЧАННЯ

Розроблено метод за вдосконаленою моделлю глибокого навчання для класифікації комах-шкідників у зерносховищах. Запропонований метод використовує оптимізовану модель YOLOv8, яка дозволяє ідентифікувати шкідників на зображеннях із високою точністю в умовах реального часу. Модель здатна аналізувати зображення різних масштабів і виявляти комах навіть на складних фонах або при малих розмірах об'єктів, що сприяє зниженню втрат зерна та підвищенню рівня продовольчої безпеки.

The method is developed using an advanced deep learning model for classifying insect pests in grain storage facilities. The proposed method uses the improved YOLOv8 model, which allows to identify pests in images with high accuracy in real time. The model is able to analyze images of various scales and detect insects even on complex backgrounds or with small object sizes, which helps reduce grain losses and increase food security.

З розвитком нових технологій та підвищенням рівня інформатизації суспільства задача виявлення шкідників зернових запасів набуває особливої актуальності [1]. У сільському господарстві точне та своєчасне виявлення шкідників відіграє ключову роль у попередженні втрат врожаю та забезпеченні продовольчої безпеки. Методи штучного інтелекту, зокрема моделі глибокого навчання, такі як YOLO, пропонують ефективні рішення для розпізнавання шкідників навіть у складних умовах зберігання зерна [2]. Такі рішення можуть застосовуватись для моніторингу в реальному часі, що дозволяє раніше виявлення шкідників, зменшує ручну працю та підвищує ефективність виявлення шкідників [3]. У міру розвитку технологій, що базуються на глибокому навчанні, зростає потреба вдосконалювати ці моделі для забезпечення високої точності в різних умовах зберігання та для різних видів шкідників [4].

Метою роботи є підвищення якості класифікації комах-шкідників на зображеннях шляхом вдосконалення моделі глибокого навчання YOLOv8.

Архітектура YOLOv8, обрана як основа для розв'язання задачі, забезпечує оптимальну продуктивність завдяки вдосконаленому модулям обробки характеристик, зокрема C2f та CSPDarknet53 [5]. Модуль C2f поліпшує передачу контексту між шарами, що сприяє розпізнаванню дрібних об'єктів на складному

фоні, характерному для зерносховищ. У YOLOv8 також інтегровано шар Spatial Pyramid Pooling Fast (SPPF), який зберігає просторову ієрархію ознак, що дозволяє одночасно розпізнавати дрібні та великі об'єкти на зображеннях різних масштабів [6].

Для адаптації моделі під специфічні умови роботи в зерносховищах було застосовано донавчання на спеціалізованих наборах даних, що включають різноманітні зображення шкідників у зернових масах. Такі набори даних дозволяють моделі адаптуватися до особливостей текстури та кольорових відмінностей між комахами й навколишнім середовищем, що є характерним для зерносховищ. Додавання шарів Dropout та контролю за перенавчанням сприяє покращенню здатності моделі узагальнювати інформацію, знижуючи ймовірність надмірної адаптації до специфічних прикладів і мінімізуючи помилки класифікації. Ці шари забезпечують стійкість моделі при обробці різних сценаріїв, включаючи зміни умови освітлення та часткове перекриття об'єктів. Оптимізація гіперпараметрів, таких як швидкість навчання, розмір батчу та тип оптимізатора, проводилася з урахуванням складності завдання та вимог до обчислювальних ресурсів. Важливість точного вибору гіперпараметрів зумовлена необхідністю забезпечити стабільну збіжність і надійність моделі в умовах обмежених обчислювальних можливостей, де максимізація точності та швидкості обробки є критично важливими для поставленої роботи.

Оцінка якості навченої моделі проводилася за основними метриками: точністю, повнотою, F1-мірою та ROC-AUC-аналізом. Отримані результати демонструють високу точність і здатність до узагальнення моделі в умовах промислових зерносховищ. Зокрема, значення метрик показали:

– Precision: 0.9938, що означає високу точність розпізнавання та мінімальну кількість хибнопозитивних спрацювань;

– Recall: 0.9937, який свідчить про повне охоплення всіх класів шкідників;

– F1-міра: 0.9937, що вказує на гармонійний баланс між точністю та повнотою;

– AUC-ROC: 0.9999, що свідчить про високу здатність моделі розрізняти різні класи.

Отримані результати вказують на значний потенціал розробленого методу для вирішення задачі класифікації комах у специфічних умовах зерносховищ, де часто спостерігаються неоднорідність фону та різні масштаби об'єктів. Завдяки донавчанню моделі YOLOv8 на спеціалізованих наборах даних, система показала здатність працювати у реальних умовах, забезпечуючи стабільність і точність моніторингу в режимі реального часу. Це дозволяє не тільки знизити втрати зерна через своєчасне виявлення шкідників, але й оптимізувати управління запасами, що в

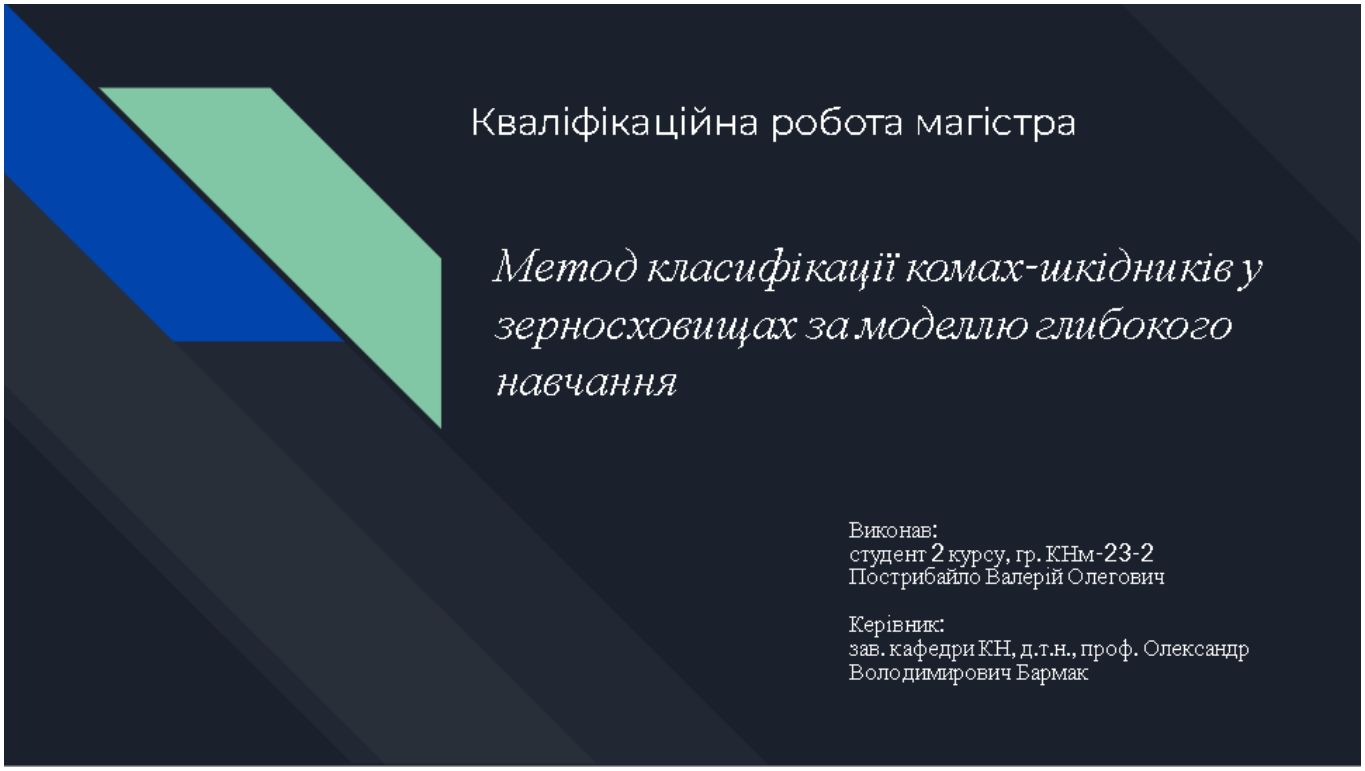
цілому сприяє підвищенню рівня продовольчої безпеки і зменшенню витрат на контроль якості зберігання зернових продуктів.

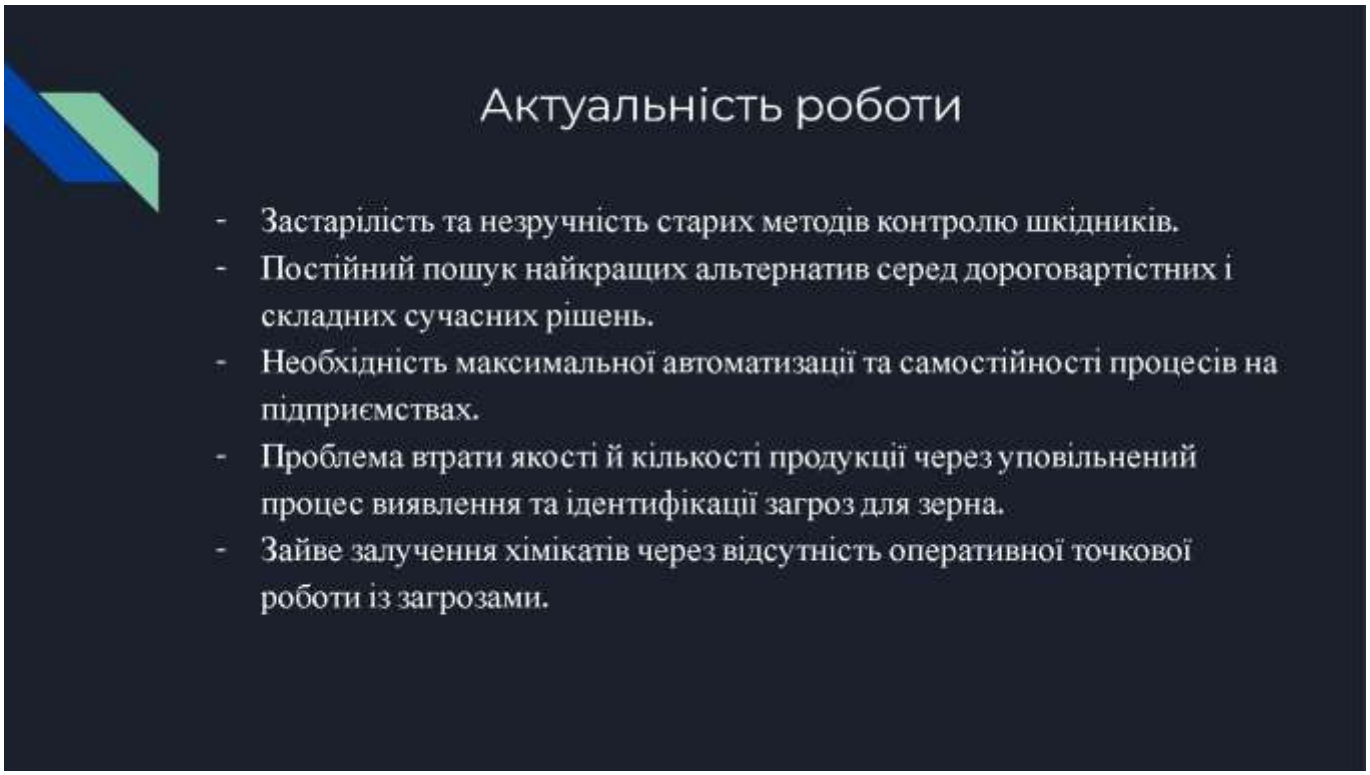
Перелік посилань

1. Deep learning in agriculture. URL: https://medium.com/@saiwadotai/deep-learning-in-agriculture-3bbcb504aa41
2. Detection of Agricultural Pests Based on YOLO. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2560/1/012013/pdf
3. Object detection in agricultural contexts: A multiple resolution benchmark and comparison to human. URL: https://www.sciencedirect.com/science/article/abs/pii/S016816992100421X
4. Detection of stored-grain insects using deep learning. URL: https://www.sciencedirect.com/science/article/abs/pii/S016816991730769X.
5. Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. Machine Learning and Knowledge Extraction, 5(4), 1680-1716.
6. Wang, Z., Chen, J., Yu, P., Feng, B., & Feng, D. SC-YOLOv8 Network with Soft-Pooling and Attention for Elevator Passenger Detection. Applied Sciences. 2024. 14(8), 3321

Додаток Г

Презентаційний матеріал





Мета і задачі роботи

Метою роботи є підвищення якості класифікації комах-шкідників на зображеннях із зерносховищ за моделлю глибокого навчання. Для досягнення поставленої мети потрібно виконати наступні завдання:

1) провести аналіз: предметної області для проблеми виявлення шкідників у зерносховищах; існуючих публікацій щодо класифікації шкідників у зерносховищах; існуючих програмних рішень та моделей глибокого навчання для виявлення об'єктів малого розміру на зображеннях;
2) вдосконалити метод виявлення та класифікації шкідників у зерносховищах на зображеннях за моделлю глибокого навчання;
3) підготувати відповідний набір даних для донавчання нейронної мережі;

Мета і задачі роботи

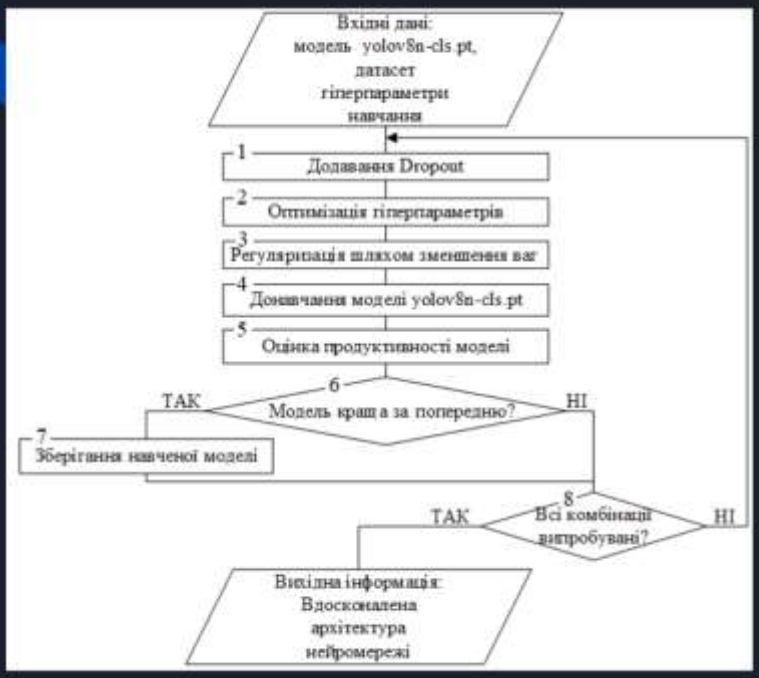
4) донавчити попередньо навчену нейронну мережу виявляти шкідників у зерносховищах;
5) імплементувати вдосконалений метод у прототипну інформаційну систему.
6) визначити якість запропонованого методу за відомими статистичними показниками та навести порівняння з іншими відомими підходами.

Об'єкт дослідження. процес виявлення та класифікації комах-шкідників на зображеннях із зерносховищ за моделлю глибокого навчання.

Предмет дослідження. Моделі глибокого навчання.

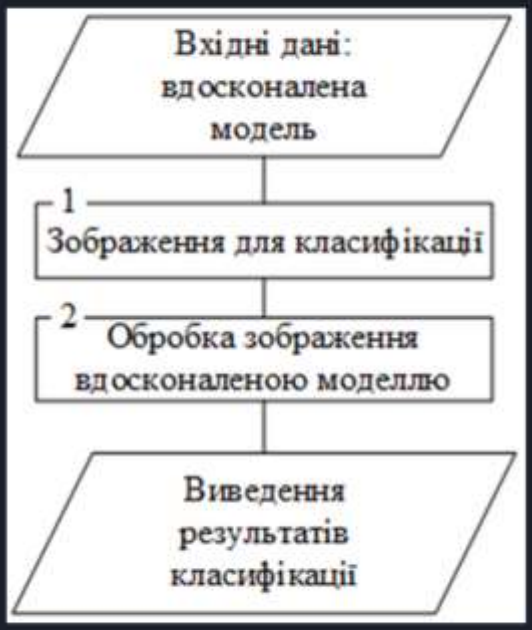
Методи дослідження. Моделі глибокого навчання для аналізу зображень з дрібними об'єктами.

Наукова новизна одержаних результатів. Вдосконалено метод класифікації комах-шкідників у зерносховищах за їх зображенням засобами глибокого навчання, який відрізняється від існуючих донавчанням моделі на спеціалізованих наборах даних, додаванням шарів Dropout для контролю за перенавчанням та оптимізацією гіперпараметрів, що дозволило працювати з меншим об'ємом даних, більшою швидкодією, підвищити точність класифікації шкідників при різних умовах фотофіксації, частковому перекритті об'єктів та збільшити кількість видів комах-шкідників, які загрожують зерносховищам.

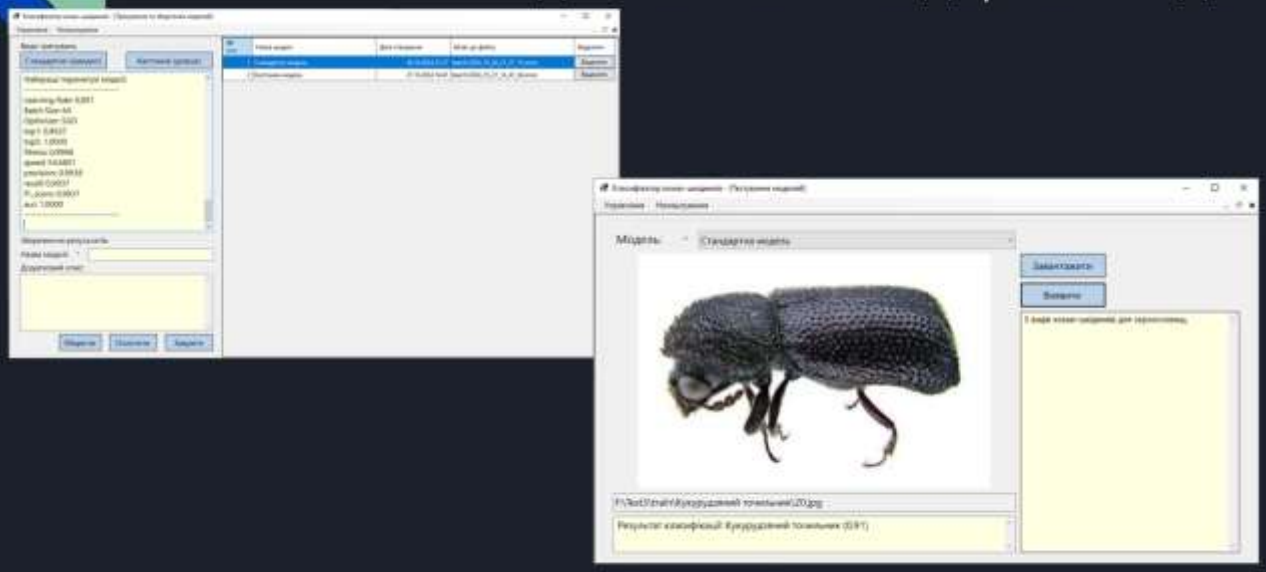


Кроки для отримання вдосконаленої моделі глибокого навчання

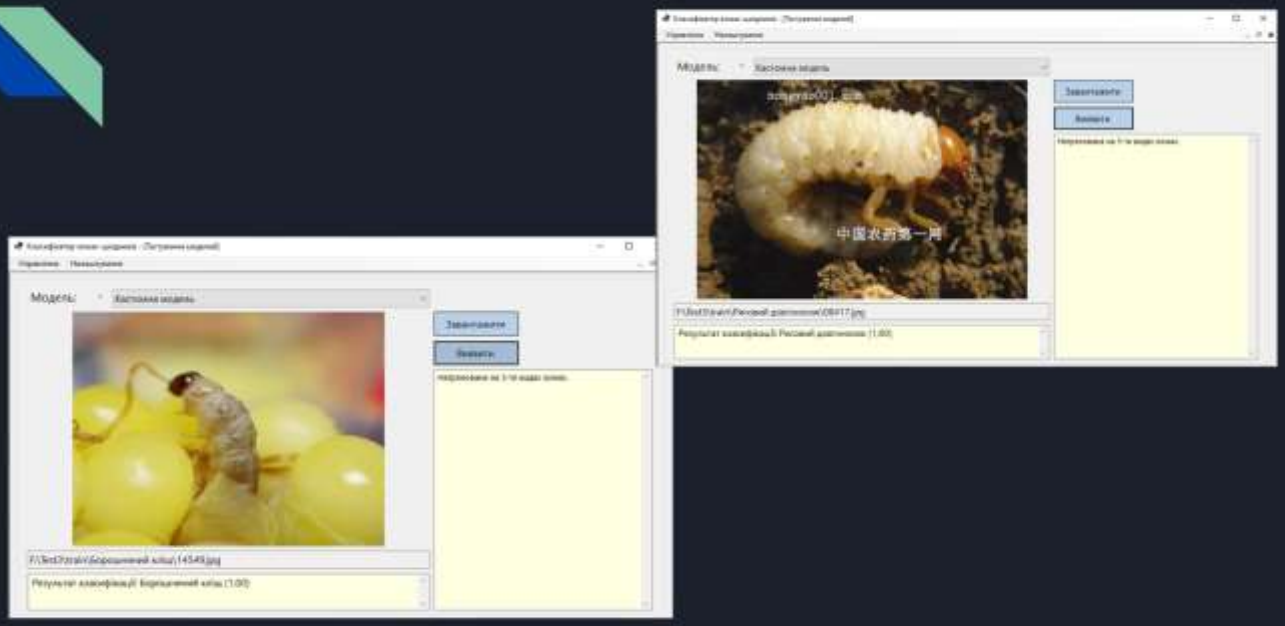
Кроки методу для класифікації за вдосконаленою та навченою моделлю



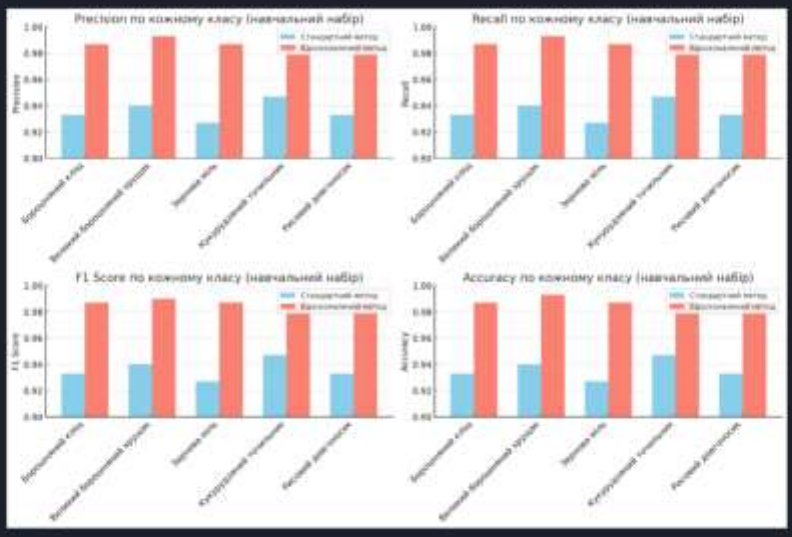
Інтерпретація отриманих результатів Розпізнавання за допомогою стандартної моделі



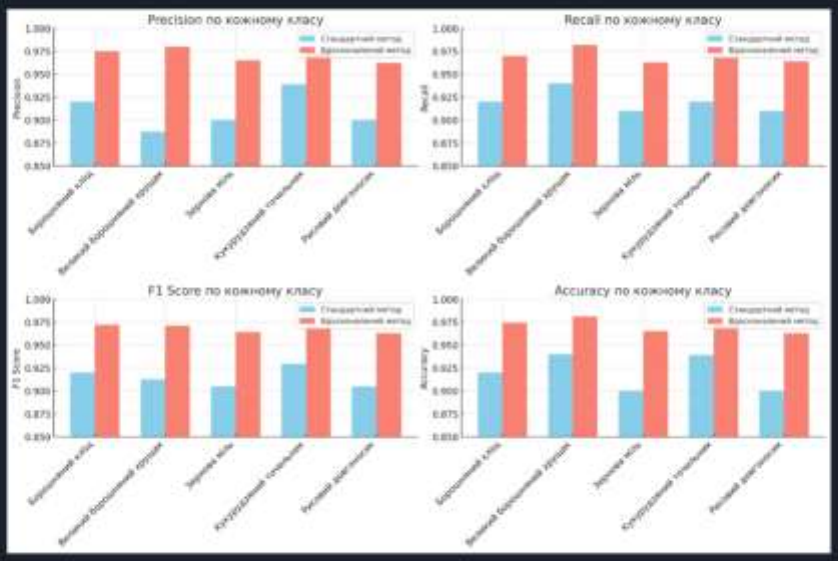
Розпізнавання за допомогою вдосконаленої моделі



Дослідження ефективності
Порівняння метрик методів для навчального набору



Дослідження ефективності
Порівняння метрик методів для тестового набору



Висновки

Метою кваліфікаційної роботи було підвищення якості класифікації комах-шкідників на зображеннях із зерносховищ завдяки розробці та впровадженню вдосконаленого методу на основі глибокого навчання. Для досягнення поставленої мети виконано наступні завдання:

1. Проведено ретельний аналіз предметної області, зокрема проблеми шкідників у зерносховищах, та здійснено огляд сучасних наукових підходів до класифікації шкідників, що дозволило визначити переваги та недоліки наявних рішень. На основі порівняння обрано модель YOLO як найбільш перспективну для застосування завдяки її здатності забезпечувати високі показники швидкодії та точності, що критично важливо для практичного застосування в реальному часі.

2. Вдосконалено метод класифікації шляхом донавчання моделі YOLOv8 на спеціально підготовленому датасеті, що відображає специфіку зображень із комахами-шкідниками у зерносховищах. Впроваджено комплекс удосконалень, серед яких регуляризаційні методи (dropout, weight decay) та оптимізація гіперпараметрів (швидкість навчання, розмір батчу, вибір оптимізатора), що забезпечило підвищення стабільності моделі та її здатності узагальнювати нові дані.

3. Розроблено метод класифікації, які були реалізовані у вигляді прототипної інформаційної системи. Прототип системи забезпечує автоматизацію процесів аналізу зображень шкідників та спрощує процес управління запасами зернових. Інтерфейс системи дозволяє користувачам здійснювати класифікацію з високим рівнем зручності та точності.

4. Для підтвердження підвищення якості класифікації проведено експерименти на розширеному тестовому наборі даних, що включає зображення комах-шкідників різних класів (борошняний клщ, великий борошняний хрущак, зернова міль, кукурудзяний точильник та рисовий довгоносик). Результати експериментів підтвердили, що вдосконалена модель показує стабільно високі показники точності та чутливості, зокрема, значення Precision становить 0.9620–0.9800, а Recall – 0.9630–0.9820. Це значно перевищує показники базової моделі та свідчить про суттєве покращення якості класифікації.

Практичне значення отриманих результатів полягає у можливості автоматизувати процес ідентифікації шкідників, що сприятиме зниженню ризиків втрат зернових запасів та поліпшенню їхнього обліку в умовах зберігання. Застосування запропонованого методу дозволить значно зменшити залежність від людського фактору та присвятити процеси контролю, що особливо важливо у великих зерносховищах. Даний метод зможе забезпечити основу для подальшого розвитку автоматизованих систем, які можуть включати додаткові функції моніторингу та безпеки. Таким чином, запропонований метод створює перспективи для більш надійного управління запасами зерна, зберігаючи їх якість та зменшуючи ризики втрат у довгостроковій перспективі.

Наукові публікації

Дякую за увагу

Thu Dec 05 15:02:23 EET 2024, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.258 Educational

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 154826 Назва: КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА на тему Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання Додано в БД: 2024-12-05 Автора: Валерій ПОСТРИБАЙЛО Керівники: Олександр БАРМАК Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	116513	1711	4158 (4%)	66 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Валерій ПОСТРИБАЙЛО

Співавтор:

Назва: Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання

Науковий керівник: Олександр БАРМАК, д.т.н., проф.

Підрозділ: Кафедра комп'ютерних наук

Коефіцієнт подібності 1:3.8%

Коефіцієнт подібності 2:1.3%

Мікропробіли: 0

Заміна букв: 84

Інтервали: 0

Білі знаки: 1

Дата **створення звіту:** 2024-12-05 14:19:21.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 5.12.2024 експерт Петровський Р.Р.

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА ДО ЗАХИСТУ ЗА РЕЗУЛЬТАТАМИ АНАЛІЗУ ЗВІТУ ПОДІБНОСТІ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод класифікації комах-шкідників у зерносховищах за модедлю глибокого навчання

Автор: Валерій ПОСТРИБАЙЛО

Спеціальність: 122 – Комп'ютерні науки

Освітня програма: освітньо-професійна

Науковий керівник: д.т.н., професор, зав. кафедри КН Олександр БАРМАК

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є допустимими і не є плагіатом, оскільки:

1) за програмою Anti-Plagiarism виявлені 3% запозичень вказують на відомі терміни.
2) За програмою StrikePlagiarism виявлені запозичення (КП1=3.8% та КП2=1.3%) є фрагментарними – містять поширені конструкції та схеми (у роботі вказані посилання на ці джерела), загальновідомі терміни, скорочення та визначення.

Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.

Керівник роботи _____ О. В. Бармак

Гарант ОП _____ Р. В. Багрій

Завідувач кафедри КН _____ О. В. Бармак

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу магістра

гр. КНм-23-2 Валерія ПОСТРИБАЙЛА за темою: *Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання*

1. Актуальність теми

Сучасні досягнення у сфері комп'ютерних наук, зокрема у галузі штучного інтелекту та глибокого навчання, активно інтегруються у різноманітні галузі, від медицини до промисловості. Одним із напрямків, де ці технології демонструють значний потенціал, є сільське господарство. Використання моделей глибокого навчання дозволяє автоматизувати складні процеси обробки та аналізу даних, які раніше вимагали значних людських ресурсів та часу. Одним із прикладів таких задач є виявлення та класифікація шкідників зернових культур під час зберігання.

2. Відповідність роботи предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до наукових робіт

Тема кваліфікаційної роботи студента повністю відповідає предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до кваліфікаційної роботи магістра. Робота вирішує науково-прикладну задачу у сфері комп'ютерних наук (а саме застосування інтелектуальних методів обробки інформації (моделей глибокого навчання) до класифікації комах-шкідників у зерносховищах). У роботі проведено дослідження, яке розвиває існуючі знання та процедури.

3. Професійні та особистісні якості магістранта

Під час опрацювання магістерської роботи магістрант продемонстрував такі професійні якості, як вміння ставити задачі, проводити їх аналіз і декомпозицію, здійснювати підбір і застосування наукових методів для вирішення часткових завдань дослідження; вміння знаходити причинно-наслідкові залежності та формувати висновки, вміння застосовувати сучасні інформаційні технології. Крім цього, магістрант проявив такі особистісні якості, як наполегливість, відповідальність, доброчесність, працелюбність, коректність.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

У ході роботи магістрант проявив достатній ступінь самостійності. Зокрема, ним особисто було: проведено аналіз предметної області для проблеми виявлення шкідників у зерносховищах; вдосконалено метод виявлення та класифікації шкідників у зерносховищах на зображеннях за моделлю глибокого навчання; підготовлено відповідний набір даних для донавчання нейронної мережі; донавчено попередньо навчену нейронну мережу виявляти

шкідників у зерносховищах; імплементувано вдосконалений метод у прототипну інформаційну систему; визначено якість запропонованого методу за відомими статистичними показниками та навести порівняння з іншими відомими підходами.

5. Наукова новизна та оригінальність запропонованих підходів

Магістерська робота характеризується наявністю наукової новизни. Відповідні положення сформульовані коректно та відображають її сутність.

Результати магістерської роботи достатньо апробовані. Зокрема, матеріали роботи доповідалися на науковій конференції «Актуальні проблеми комп'ютерних наук».

6. Ступінь оволодіння методами дослідження

У магістерській роботі магістрантом реалізовано комплексний підхід щодо застосування загальнонаукових і спеціальних методів наукових досліджень. Зокрема, використано методи застосування згорткової нейронної мережі, сучасних інформаційних технологій, вищої математики, теорії ймовірностей і математичної статистики. Під час їх застосування до вирішення окремих завдань дослідження магістрант продемонстрував достатній ступінь володіння ними.

7. Повнота та якість розкриття теми роботи

Тема роботи розкрита достатньо повно та якісно. Визначена мета роботи досягнута.

8. Логічність, послідовність, аргументованість, літературна грамотність викладу матеріалу

Викладення матеріалу магістерської роботи є достатньо логічним, послідовним і має достатній рівень аргументації. Робота написана літературною мовою, граматична якість викладення матеріалу роботи задовільна.

9. Можливість практичного застосування кваліфікаційної роботи, окремих її частин

Робота має теоретико-прикладний характер. Результати роботи дозволяють автоматизувати процес моніторингу стану зерносховищ, забезпечуючи високу точність та швидкість класифікації шкідників. Це сприяє зменшенню втрат продукції та зниженню потреби у використанні хімічних засобів для боротьби з шкідниками.

10. Висновок про можливість допуску кваліфікаційної роботи до захисту, на яку оцінку заслуговує робота

Рекомендую допустити роботу до захисту.

Кваліфікаційна робота заслуговує на оцінку «Добре».

Науковий керівник _____ д.т.н., проф. Олександр Бармак

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МОН УКРАЇНИ

кафедра комп'ютерних наук

ВІДГУК ОПОНЕНТА

на кваліфікаційну роботу магістра

гр. КІНм-23-2 Валерія ПОСТРИБАЙЛА за темою: *Метод класифікації комах-шкідників у зерносховищах за моделлю глибокого навчання*

1. Актуальність обраної теми

Обрана магістрантом тема є достатньо актуальною на даний час, особливо з урахуванням тенденцій щодо розвитку штучного інтелекту. Актуальність обраної теми достатньо обґрунтована в роботі.

2. Відповідність роботи предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до наукових робіт

Тема кваліфікаційної роботи студента повністю відповідає предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до кваліфікаційної роботи магістра. Робота вирішує науково-прикладну задачу у сфері комп'ютерних наук (а саме застосування інтелектуальних методів обробки інформації (моделей глибокого навчання) до класифікації комах-шкідників у зерносховищах). У роботі присутні дослідження, які розвивають існуючі підходи, та отримані нові результати, які за певними показниками – вищі, ніж в аналогічних підходах.

3. Повнота розкриття мети та завдань дослідження

Часткові завдання дослідження сформульовані коректно та повністю відповідають меті роботи.

4. Наявність наукової новизни

Магістерська робота характеризується наявністю наукової новизни. Положення наукової новизни сформульовані коректно та відображають їх сутність.

Матеріали магістерської роботи апробовані на науковій конференції «Актуальні проблеми комп'ютерних наук».

5. Зміст кожного розділу роботи

Робота складається з чотирьох розділів.

У першому розділі – «Аналіз сучасного стану у використанні методів глибокого навчання для класифікації шкідників у зерносховищах за зображеннями» – автором проведено детальний аналіз предметної області, огляд існуючих публікацій, присвячених класифікації шкідників, а також розглянуто моделі глибокого навчання, що застосовуються для виявлення об'єктів малого розміру на зображеннях. На основі цього аналізу сформульовано задачу дослідження.

У другому розділі – «Вдосконалення методу класифікації комах-шкідників за зображеннями» – магістрантом описано обрану модель глибокого навчання, запропоновано вдосконалення її архітектури, виконано доналаштування моделі, контроль за перенавчанням та оптимізацію гіперпараметрів. Крім того, визначено метрики для оцінки якості навченої моделі.

У третьому розділі – «Основні кроки вдосконаленого методу для виявлення комах-шкідників у зерносховищах» – автором представлено покроковий опис процесу отримання вдосконаленої моделі та процесу класифікації з використанням навченої моделі.

У четвертому розділі – «Експериментальні результати» – магістрантом проведено оцінку ефективності запропонованого методу. Як результати розглянуто порівняння між вдосконаленим методом і існуючими моделями. Отримані результати підтвердили переваги запропонованого методу.

6. Ступінь розкриття теми роботи

Тема роботи розкрита достатньою мірою та мета досягнута.

7. Якість оформлення кваліфікаційної роботи

Робота написана літературною мовою та оформлена згідно відповідних вимог достатньо якісно.

8. Недоліки кваліфікаційної роботи

Разом з тим, у роботі є і певні недоліки. Так, у роботі відсутні відповіді на питання щодо ефективності авторського удосконалення на інших наборах даних, крім тих, що досліджені в магістерській роботі.

Проте зазначений недолік загалом не впливає на комплексну позитивну оцінку роботи.

9. Загальний висновок (допускається чи не допускається до захисту), якої оцінки заслуговує кваліфікаційна робота.

Рекомендую допустити роботу до захисту.

Кваліфікаційна робота заслуговує на оцінку «Добре».

Опонент (прізвище, ім'я, по батькові, посада, місце роботи)

Бедратюк Леонід Петрович, зав. кафедри ІПЗ

«10» грудня 2024 р.

Підпис