Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Моніторинг та аналіз пошкоджень сільськогосподарських земель засобами ШІ
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-3
Крістіна Сільванович
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ст. викл. Олена Гриньова
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Сільванович Крістіні Валеріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Моніторинг та аналіз пошкоджень сільськогосподарських земель засобами ШІ

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 червня 20 25 р.

3. Вихідні дані до роботи Зображення сільськогосподарських земель, типи пошкоджень, агрокліматичні дані, розмічені маски, моделі ШІ.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі

2) Архітектура системи

3) Програмна реалізація моделей штучного інтелекту

4) Інтерфейс користувача та тестування моделей

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	21.05.2025	виконано
3	Огляд існуючих метрик оцінювання якості	25.05.2025	виконано
4	Аналіз підходів вирішення задачі моніторингу	27.05.2025	виконано
5	Експериментальне моделювання та навчання		
	моделі	30.05.2025	виконано
6	Написання пояснювальної записки	31.05.2025	виконано
7	Перевірка на академічний плагіат	02.06.2025	виконано
8	Нормоконтроль	04.06.2025	виконано
9	Підготовка презентації та доповіді	08.06.2025	виконано
10	Попередній захист	10.06.2025	виконано
11	Рецензування	13.06.2025	виконано
12	Захист перед ЕК	17.06.2025	виконано

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Олена Гриньова
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 95 с., 8 рис., 10 дод., 18 джерел.

АГРОЕКОСИСТЕМИ, ГЛИБОКЕ НАВЧАННЯ, МОНІТОРИНГ ҐРУНТІВ, ПРОГНОЗ ВІДНОВЛЕННЯ, СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, AGROECOSYSTEM RECLAMATION, CONVOLUTIONAL NEURAL NETWORK, DECISION SUPPORT SYSTEM, LONG SHORT-TERM MEMORY, SOIL DAMAGE DETECTION, TEMPORAL FUSION TRANSFORMER, U-NET.

Об’єкт дослідження – процес оцінки та відновлення сільськогосподарських земель, що зазнали пошкодження внаслідок зовнішніх руйнівних впливів, зокрема бойових дій.

Предмет дослідження – застосування алгоритмів штучного інтелекту для автоматизованого моніторингу стану ґрунтів на основі аерофото- та супутникових знімків.

Мета роботи – розробити підхід до моніторингу та прогнозування стану агроекосистем після пошкодження з використанням глибоких нейронних мереж та архітектури U-Net, та впровадити систему підтримки прийняття рішень для оцінки строків відновлення земель.

Методи дослідження – машинне навчання, глибинне навчання, обробка зображень, методи сегментації, аналіз метаданих.

Протягом написання роботи досліджуються сучасні методи автоматизованої класифікації пошкоджених земель за допомогою системи з трьох моделей: детекція пошкоджень, семантична сегментація та прогноз відновлення. Реалізується базова структура Decision Support System (DSS) і застосовується архітектура U-Net для точного визначення меж уражень. Використовуються аерофото- та супутникові знімки й методи комп’ютерного зору.

ABSTRACT

Bachelor’s thesis contains: 95 pp., 8 fig., 10 ann., 18 references.

AGROECOSYSTEM RECLAMATION, CONVOLUTIONAL NEURAL NETWORK, DECISION SUPPORT SYSTEM, DEEP LEARNING, IMAGE SEGMENTATION, LONG SHORT-TERM MEMORY, RESTORATION FORECASTING, SOIL DAMAGE DETECTION, TEMPORAL FUSION TRANSFORMER, U-NET.

Object of study – the process of assessing and restoring agricultural lands damaged due to external destructive impacts, particularly military actions.

Subject of study – the application of artificial intelligence algorithms for automated soil condition monitoring based on aerial and satellite imagery.

Objective of the study – to develop an approach for monitoring and forecasting the condition of agroecosystems after damage using deep neural networks and the U-Net architecture, and to implement a decision support system for estimating land restoration timelines.

Research methods – machine learning, deep learning, image processing, segmentation techniques, and metadata analysis.

During the development of the project, modern methods of automated classification of damaged lands were investigated using a system of three models: damage detection, semantic segmentation, and restoration forecasting. A basic structure of a Decision Support System (DSS) was implemented, and the U-Net architecture was applied for precise delineation of affected areas. Aerial and satellite imagery, along with computer vision techniques, were utilized.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Агроекосистема – екосистема, що складається з сільськогосподарських земель та пов’язаних біологічних компонентів;

Аерофотознімки – зображення, отримані з повітряних або космічних апаратів, що використовуються для аналізу земель;

Глибинне навчання – підмножина машинного навчання, що використовує багатошарові нейронні мережі для аналізу даних;

M1 – модель класифікації пошкоджених ділянок;

M2 – модель сегментації пошкоджень;

M3 – модель прогнозування часу відновлення;

МШІ – модель штучного інтелекту;

Рекультивація – комплекс заходів із відновлення пошкоджених земель для їхнього подальшого використання;

ШІ – штучний інтелект;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

LSTM – Long Short-Term Memory – довга короткострокова пам’ять, тип рекурентної нейронної мережі;

.NET – платформа розробки від Microsoft;

ML.NET – бібліотека машинного навчання для .NET;

ONNX – Open Neural Network Exchange – формат для обміну нейронними мережами;

PReLU – Parametric Rectified Linear Unit – параметрична модифікована лінійна одиниця активації;

Softmax – активатор Softmax – функція активації для інтерпретації ймовірностей класів;

TFT – Temporal Fusion Transformer – трансформер для прогнозування часових рядів;

U-Net – сегментаційна нейронна мережа типу U-подібної архітектури.

ВСТУП

Збройні конфлікти залишають глибокий слід не лише в суспільстві, а й у довкіллі. Сільськогосподарські землі, які десятиліттями забезпечували продовольчу стабільність, сьогодні перетворюються на нестабільні, потенційно небезпечні зони, де ґрунт може бути змінений до невпізнаваності: вибухи, уламки, важкі метали та інші наслідки воєнних дій руйнують родючі шари ґрунту, роблячи землю непридатною для вирощування культур. Ці пошкодження загрожують не лише врожаю та економіці, а й самому життю агроекосистем, адже без здорової землі немає майбутнього для продовольчої безпеки держави.

Щоб повернути ці поля до життя, потрібно спочатку зрозуміти, що з ними сталося: чи можна сіяти, чи земля потребує серйозного відновлення. У минулому для цього проводили польові обстеження, брали зразки ґрунтів, відправляли їх у лабораторії – ці методи забирали багато часу, ресурсів і не могли охопити великі території. Особливо це стає критичним у післявоєнний період, коли йдеться про сотні тисяч гектарів, що постраждали одночасно. У таких умовах традиційні підходи виявляються неефективними – потрібні сучасні, масштабовані та швидкодіючі рішення.

І саме тут на допомогу приходять сучасні технології – штучний інтелект, комп’ютерний зір і алгоритми глибокого навчання. Використовуючи зображення з супутників або з дронів, ці системи можуть точно виявити, де є пошкодження, окреслити межі уражених ділянок, класифікувати тип пошкодження (механічне, хімічне, термічне) і навіть оцінити, скільки часу знадобиться для повного відновлення родючості. Система може, наприклад, визначити, що конкретна зона має сліди забруднення, а інша – повністю зруйнований верхній шар ґрунту. Це дає змогу створити точні карти ураження і ухвалити рішення: де можна сіяти, а де потрібна рекультивація.

Створення такої «розумної» системи, яка автоматично аналізує великі обсяги знімків, виявляє пошкодження, оцінює масштаби і прогнозує відновлення, відкриває новий рівень ефективності у відновленні сільськогосподарських земель, є актуальною задачею. Це дозволить фермерам, агрономам і органам влади отримувати об’єктивну картину ситуації у реальному часі, зменшуючи залежність від польових виїздів, суб’єктивних оцінок та людського фактору.

Однак самі по собі зображення – лише сировина. Щоб перетворити їх на корисну інформацію, необхідно обробити дані, виділити ключові характеристики, прибрати шуми, порівняти з базовими станами. Усе це вручну зробити складно, особливо коли мова йде про тисячі або десятки тисяч знімків. Саме тому ключовим стає впровадження автоматизованих систем аналізу, побудованих на принципах машинного навчання – систем, що здатні адаптуватися, вчитися на прикладах і робити висновки з нових даних.

Такий підхід не лише дозволяє зекономити час і ресурси – він відкриває можливість перейти від реагування на проблему до її прогнозування. Прогнозування термінів відновлення, врахування типів пошкоджень, кліматичних чинників, сезонності – усе це сприяє створенню реалістичних і ефективних стратегій для аграрної відбудови. Сільське господарство отримує шанс не просто вижити, а перезапуститися на якісно новому рівні – з точністю, швидкістю й технологічною підтримкою, які ще кілька років тому здавалися фантастикою.

Саме тому розробка таких систем – це не лише технічне завдання. Це внесок у продовольчу безпеку, у післявоєнне відновлення країни та в збереження природного потенціалу, який ми не маємо права втратити.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Сучасні підходи до оцінки стану ґрунтів

Сільськогосподарські землі – це не просто поля, де ростуть культури, а основа продовольчої безпеки країни. Від їхнього стану залежить, чи матимемо ми достатньо їжі, чи зможемо підтримувати економіку й екосистему. Але в умовах війни, коли поля зазнають руйнувань від вибухів, забруднення важкими металами чи ущільнення ґрунту, збереження та відновлення цих земель стає справжнім викликом. Ураження можуть бути різними: від величезних вирв після обстрілів до ерозії чи порушення водного балансу через пошкодження дренажних систем. І щоб зрозуміти, як повернути ці землі до життя, потрібно спочатку оцінити, наскільки сильно вони постраждали.

Раніше для оцінки стану ґрунтів фермери й науковці покладалися на перевірені, але доволі громіздкі методи. Наприклад:

– польові дослідження: вчені виїжджали на поле, брали зразки ґрунту і відправляли їх до лабораторії. Там аналізували, чи є в ґрунті токсичні речовини, наскільки він родючий, чи не втратив поживних елементів;

– геохімічні аналізи: ці методи допомагали визначити, чи не забруднені ґрунти важкими металами, як-от свинець чи кадмій, що можуть потрапляти туди після вибухів;

– картографування: за допомогою геопросторових даних створювали мапи, на яких позначали деградовані ділянки – наприклад, місця, де ґрунт ущільнився чи почалася ерозія;

– візуальні огляди: фахівці ходили полями, фіксували вирви, спалені ділянки чи інші видимі пошкодження.

Ці методи працюють, але мають серйозні недоліки. По-перше, вони займають багато часу – від збору зразків до отримання результатів можуть пройти тижні. По-друге, потрібні значні ресурси: люди, обладнання,

транспорт. І найголовніше – такі підходи не дозволяють швидко оцінити стан величезних територій, особливо коли йдеться про сотні чи тисячі гектарів постраждалих земель. У воєнний чи післявоєнний період, коли кожна хвилина на рахунку, це стає критичною проблемою.

На щастя, сучасні технології дозволяють нам подивитися на проблему з іншого боку. Дистанційне зондування землі (ДЗЗ) у поєднанні з алгоритмами штучного інтелекту відкриває нові можливості. Замість того, щоб тижнями ходити полями, ми можемо за лічені години отримати детальну картину стану земель. Ось які інструменти для цього використовують:

– супутникові знімки: супутники, такі як Sentinel-2 чи Landsat 8, роблять знімки Землі з космосу. Ці зображення дозволяють побачити зміни на полях – наприклад, де ґрунт став менш родючим, де з’явилися вирви чи де рослинність зникла;

– аерофотознімки з дронів: безпілотники можуть літати на низькій висоті й робити надточні фото. Вони особливо корисні для невеликих ділянок, де потрібна висока деталізація;

– ГІС-шари: дані з відкритих джерел, як-от OpenStreetMap чи GlobCover, допомагають додати контекст – наприклад, інформацію про типи ґрунтів чи розташування водойм;

– метеорологічні дані: погода впливає на стан ґрунтів, тож інформація про опади, температуру чи вологість допомагає точніше оцінити ситуацію.

Але самі по собі ці дані – лише сирий матеріал. Щоб вони стали корисними, потрібні розумні алгоритми, які можуть обробити величезні обсяги інформації й видати чіткі висновки. І тут на сцену виходить штучний інтелект.

Сучасні алгоритми глибокого навчання, такі як згорткові нейронні мережі, дозволяють не просто аналізувати зображення, а й знаходити на них конкретні проблеми. Наприклад, ШІ може:

– виявляти пошкоджені ділянки: алгоритми визначають, де на полі є вирви, спалена земля чи ерозія;

– сегментувати уражені зони: це означає, що система не просто показує, що поле пошкоджене, а й чітко окреслює межі цих ділянок;

– оцінювати масштаби: ШІ може підрахувати, яка саме площа постраждала, і навіть класифікувати тип пошкодження;

– прогнозувати відновлення: спеціальні моделі, аналізують дані про клімат, тип ґрунту та ступінь ураження, щоб передбачити, скільки часу знадобиться для рекультивації.

Наприклад, уявімо поле, яке зазнало обстрілів. Супутник Sentinel-2 робить знімок, дрон уточнює деталі, а ШІ за лічені хвилини видає звіт: «На ділянці площею 50 гектарів виявлено 12 вирв, 30% території забруднено, відновлення займе приблизно 2 роки за умови рекультивації». Такі висновки дозволяють фермерам і владі швидко планувати подальші дії – від очищення ґрунту до вибору культур, які можна вирощувати.

1.2 Штучний інтелект у моніторингу екосистем

Останні десять років стали справжнім проривом для технологій штучного інтелекту. Вони проникли в усі сфери життя, і особливо помітно це в природоохоронних та аграрних дослідженнях. Сьогодні за допомогою розумних алгоритмів ми можемо стежити за станом лісів, полів, річок і озер, помічати, де природа страждає, і навіть прогнозувати, що буде далі, якщо не втрутитися. Це ніби мати супермозок, який бачить усе одразу – від змін у ґрунті до наслідків людської діяльності чи природних катаклізмів.

Один із найцікавіших напрямів – це аналіз зображень, які надходять із супутників, дронів чи аерофотозйомки. Такі фото допомагають зрозуміти, що відбувається з рослинністю, чи вистачає ґрунту вологи, чи немає слідів забруднення, як відновлюються поля після пожеж або бойових дій, і навіть

чи не загрожує землі ерозія чи заболочення. Завдяки сучасним технологіям ми можемо не лише побачити ці зміни, а й детально їх проаналізувати.

Ключову роль у цьому відіграють згорткові нейронні мережі – розумні програми, які вміють знаходити закономірності на зображеннях. Наприклад, архітектури на кшталт U-Net чи DeepLabv3+ дозволяють із неймовірною точністю визначати, де на полі є проблема. Вони можуть показати, де з’явилися вирви від вибухів, де вигоріла трава, де ґрунт постраждав від хімікатів чи радіації. Це ніби дати машині лупу, яка бачить кожен сантиметр землі.

Щоб усе працювало як слід, зображення спочатку готують: прибирають шум, роблять кольори чіткішими, додають різноманітності до даних, щоб програма могла працювати з різними типами фото. А ще ці технології часто поєднують із геоінформаційними системами, які допомагають зрозуміти, де саме на мапі розташовані проблемні місця.

Такі системи вже активно використовуються. Наприклад, за допомогою супутникових даних і розумних алгоритмів можна оцінити, наскільки здорові посіви, просто подивившись на показники, як-от NDVI. Або, скажімо, дрони роблять детальні знімки поля, і програма на основі U-Net одразу показує, де ґрунт постраждав від пожежі чи вибуху. Є навіть моделі, які аналізують, як змінюється природа з часом, і можуть спрогнозувати, скільки знадобиться, щоб поле знову зазеленіло. Для цього використовують кліматичні дані та спеціальні інструменти, які вміють працювати з часовими рядами.

Особливо цінними ці технології стають там, де працювати на землі важко чи небезпечно – наприклад, у регіонах, постраждалих від війни. Уявіть: поля заміновані, дороги зруйновані, а людей для досліджень бракує. У таких умовах вийти в поле з лопатою і пробірками просто неможливо. Але супутники й дрони можуть зібрати дані, а розумні програми – швидко показати, які ділянки постраждали найбільше, де потрібно починати очищення, і як краще відновлювати землю. Вони навіть можуть створювати

детальні карти, які показують, що і де відбувається, і давати поради, як повернути родючість.

Звісно, не все ідеально. Іноді бракує якісних даних, щоб навчити програми розпізнавати специфічні проблеми, як-от сліди від вибухів чи хімічного забруднення. Буває, що знімки з супутників і дронів дуже відрізняються, і це ускладнює роботу. А ще моделі потрібно постійно адаптувати, щоб вони працювали в різних кліматичних умовах і в реальному часі. Але попри ці виклики, напрям розвивається шаленими темпами.

Сьогодні ми стоїмо на порозі створення систем, які не просто знаходять проблеми, а й допомагають їх вирішувати. Це як мати розумного помічника, який не лише діагностує, а й пропонує план дій. Такі технології дають надію, що навіть після найважчих криз ми зможемо повернути життя нашим полям, лісам і річкам, зробити їх знову зеленими й родючими.

1.3 Мета роботи та постановка задачі

Ця програма має за мету допомогти фермерам, екологам і управлінцям земельних ресурсів оцінювати стан сільськогосподарських земель, які постраждали під військових конфліктів, і планувати їхнє відновлення. Додаток має бути простим, зрозумілим й дозволяв кожному – від фермера до спеціаліста – легко проаналізувати земельну ділянку: завантажити знімок, побачити вирви від вибухів чи забруднення від зброї та дізнатися, коли земля знову буде готова до посівів. Мета – створити інструмент, який допоможе відновлювати землі після війни й у майбутньому стане частиною великих систем, що підтримуватимуть відбудову сільського господарства країни.

Користувач зможе завантажити зображення поля – наприклад, фото з дронів чи супутникові знімки Sentinel-2 – і кількома кліками запустити аналіз. Програма покаже, де на полі є пошкодження від зброї, наскільки вони серйозні, і підкаже, скільки часу може знадобитися для відновлення

землі. Усе це буде подано ясно: карти з виділеними проблемними ділянками, цифри про площу пошкоджень або звіти з порадами, що робити далі. Програма розробляється так, щоб її можна було покращувати й підключати до більших систем, які стежать за станом земель.

Програма виконує три головні завдання: знаходити уражені ділянки, визначати масштаби пошкоджень і прогнозувати, коли поле можна буде використовувати знову. Для цього було створено три основні частини, кожна з яких матиме свою роль.

Було постановлено задачі:

– розробка методики виявлення пошкоджених сільськогосподарських земель за допомогою супутникових знімків Sentinel-2 та аерофотознімків, використовуючи алгоритми комп'ютерного зору, згорткові нейронні мережі (CNN) та методи семантичної сегментації;

– оцінка ступеня пошкодження земель, аналіз характеру руйнувань, класифікація їх за рівнем критичності та визначення потенційні ризики для аграрної діяльності;

– розробка прогнозної моделі для визначення термінів відновлення ґрунту, використовуючи алгоритми машинного навчання для аналізу історичних даних про стан ґрунтів та кліматичних факторів;

– інтеграція всіх моделей у єдину систему підтримки прийняття рішень, що дозволяє аграрним підприємствам та державним органам ефективно планувати заходи з відновлення земель. Тестування та оцінка точності моделей, визначення ефективності їхнього застосування у реальних умовах та можливі шляхи вдосконалення.

Перша частина програми – це інструмент, який переглядає зображення й вирішує, чи є на полі сліди від зброї, наприклад, вирви чи забруднення. Запропоновано використання штучних нейронних мереж, що застосовує техніку регуляризації згорткової ШНМ з процедурою скорочення шляхів (Shortcut), а також дані дистанційного зондування (супутникові знімки, таких як знімки Sentinel-2,

аерофотознімки з дронів) [1], [2]. Для активації шарів згорткової нейронної мережі для детекції (ЗНМ-Д), а також повнозв'язного шару, ми обрали функцію активації PReLU (Parametric Rectified Linear Unit). Вона допомагає уникнути проблеми згасання градієнта, забезпечує плавне оновлення ваг і робить процес навчання швидшим. Для вихідного шару використано функцію активації Softmax, яка гарантує точне визначення ймовірностей для кожного класу. Для створення згорткової нейронної мережі для детекції (ЗНМ-Д) застосовано фреймворк ML.NET – офіційна бібліотека машинного навчання від Microsoft, яка інтегрується з C# та дозволяє реалізовувати моделі глибокого навчання безпосередньо в середовищі розробки. На цьому етапі програма розпізнає картографічні зображення сільськогосподарських ділянок, щоб знайти райони, пошкоджені зброєю. Ця частина швидко перевіряє знімки і розділяє їх на «уражені» й «неуражені», щоб користувач знав, де треба діяти.

Друга частина – це модель сегментації, яка детально показує, де саме на полі є пошкодження, їх масштаб та серйозність. Для розв'язання цієї задачі інтегровано алгоритм комп'ютерного зору Convolutional Neural Networks (CNN), застосовуючи архітектуру U-Net. Модель, завдяки своїй енкодер-декодерній архітектурі, забезпечить точну піксельну сегментацію, визначаючи навіть дрібні деталі пошкоджень, що критично важливо для планування рекультивації. Це допоможе одразу побачити проблему і почати планування, як повернути полю родючість.

Третя частина – це модуль прогнозування, який оцінює, скільки часу знадобиться для відновлення землі після ураження зброєю. Він враховуватиме такі фактори, як площа ураження, тип зброї, тип ґрунту та кліматичні умови. Використано комбінацію Temporal Fusion Transformer (TFT) для обробки статичних даних, таких як тип зброї чи геолокація, і Long Short-Term Memory (LSTM) для аналізу динамічних змін, наприклад, сезонних чи кліматичних факторів, що забезпечить точніші прогнози. Поєднання TFT і LSTM забезпечує точні та стабільні прогнози:

TFT враховує поточні умови, тоді як LSTM відстежує довготривалі зміни. Для підвищення ефективності застосовується оптимізація гіперпараметрів.

Для створення програми обрано мову програмування C# і платформу .NET, бо це надійні інструменти, які дозволяють робити зручні й гнучкі програми. З ними легко працювати й додавати нові функції. Інтерфейс зробимо через WPF – усе буде просто: кілька кнопок, щоб завантажити знімок, запустити аналіз і побачити результати у вигляді карти чи звіту.

Для аналітичних функцій використовувався ML.NET – бібліотеку від Microsoft, яка допомагає створювати інструменти для аналізу даних. Вона ідеальна для нашого прототипу, бо може працювати з невеликою кількістю даних і підтримує все, що нам потрібно: перевірку зображень, оцінку пошкоджень і прогнози. Для складніших частин, як-от детальний аналіз чи прогнози, спочатку створено інструменти в Python, використовуючи PyTorch чи TensorFlow, а потім перенесено їх у C# через формат ONNX. Наприклад, перша частина була створена в Python, збережена як ONNX-файл і підключена до C# для швидкої роботи.

Дані, як-от знімки чи інформація про зброю й площу пошкоджень, були взяті з відкритих джерел і зберігаються в простих CSV-файлах – їх легко зібрати навіть вручну. У майбутньому, якщо програма стане більшою, є можливість підключення бази даних, наприклад, Microsoft SQL Server, щоб працювати з великими обсягами інформації чи кількома користувачами.

Програма являє собою три моделі, інтегровані у систему підтримки прийняття рішень та поєднає в собі інструменти для аналізу зображень, розумні алгоритми та зручний інтерфейс, щоб допомогти оцінювати стан аграрних ділянок, пошкоджених внаслідок військових атак, і планувати їхнє відновлення. Вона стане першим кроком до створення системи, яка підтримуватиме тих, хто працює над рекультивацією земель, допомагаючи повернути родючість полям після війни.

2 АРХІТЕКТУРА СИСТЕМИ

Цей розділ описує архітектуру системи, яка поєднує три ключові моделі для оцінки стану сільськогосподарських земель, пошкоджених унаслідок військових дій. Кожна модель відповідає за своє завдання: виявлення пошкоджених ділянок, сегментацію пошкоджень і прогнозування часу відновлення. Система побудована так, щоб бути простою у використанні, масштабованою та ефективною навіть за обмежених даних. Нижче детально описано кожну модель, її архітектуру, принципи роботи та інтеграцію в загальну систему.

2.1 Трирівнева клієнт-серверна модель (UI – логіка – моделі)

Під час розробки програмного забезпечення для системи було обрано трирівневу клієнт-серверну архітектуру, що дозволяє чітко розмежувати логіку користувача, бізнес-логіку та обчислювальні модулі штучного інтелекту. Такий підхід забезпечує високу модульність, спрощує підтримку, масштабування системи та можливість подальшого розширення без зміни всієї структури.

Трирівнева архітектура в даному проєкті поділяється на декілька рівнів.

Презентаційний рівень (UI – User Interface): це перший рівень взаємодії користувача із системою. У системі інтерфейс розроблено з використанням Windows Presentation Foundation (WPF) – сучасного інструменту від Microsoft, що дозволяє будувати гнучкі, масштабовані та стильні десктопні застосунки на мові C#.

Інтерфейс надає користувачеві змогу:

– завантажувати зображення з локального комп’ютера;

– запускати класифікацію зображення;

– виконувати сегментацію пошкоджених ділянок;

– здійснювати прогнозування наслідків;

– переглядати результати обробки;

– зберігати результати в зручному форматі (PDF або DOCX).

Елементи керування (кнопки, комбо-бокси, текстові блоки, зображення тощо) розміщено на формі MainWindow.xaml. Для кожного елемента передбачено обробник подій у відповідному .cs-файлі.

Логічний рівень (Business Logic Layer): відповідає за опрацювання всієї логіки взаємодії між користувачем і моделями. Він реалізований у коді MainWindow.xaml.cs, де:

– обробляються події натискання кнопок;

– здійснюється перевірка вхідних даних;

– координується виклик моделей ONNX;

– обробляються результати й передаються в UI;

– формується звіт.

Клас MainWindow також містить змінні для зберігання проміжних результатів: шлях до зображення, результат класифікації, згенерована маска та прогнозовані значення. Всі моделі запускаються асинхронно, щоб не блокувати інтерфейс користувача. Приклад логіки показано у лістингу 2.1.

Лістинг 2.1 – Програмний код активації відповідних елементів керування відповідно до результатів класифікації

```
if (result.Contains("Уражена"))
{
    SegmentButton.Visibility = Visibility.Visible;
    PredictButton.Visibility = Visibility.Visible;
}
else{
    SegmentButton.Visibility = Visibility.Collapsed;
    PredictButton.Visibility = Visibility.Collapsed;
    MaskImage.Source = null;
    ForecastTextBlock.Text = "";
}
```

Рівень моделей (AI Models / Data Layer): цей рівень відповідає за всю обробку даних штучним інтелектом. До нього входять класи, які завантажують та запускають моделі ONNX, збережені у форматі, сумісному з ONNX Runtime. Для кожної моделі є окремий клас:

– OnnxModelRunner – виконує класифікацію зображення на дві категорії: уражене / неуражене поле (M1);

– UnetModelRunner – реалізує сегментацію зображення за допомогою моделі U-Net, формуючи маску пошкодженої області (M2);

– LstmOnnxRunner – використовується для прогнозування ймовірної площі ураження та часу відновлення посівів на основі послідовних аграрних і погодних даних (M3).

Всі моделі попередньо створено та навчені в середовищі Python (з використанням PyTorch, pandas, sklearn), а потім експортовано у формат ONNX (Open Neural Network Exchange). Це дозволяє легко запускати їх у .NET-застосунках без втрати продуктивності.

2.2 Модель для виявлення пошкоджених ділянок

Першою частиною є модель для виявлення пошкоджених ділянок призначена для швидкої класифікації зображень сільськогосподарських полів, щоб визначити, чи є на них сліди ураження зброєю, такі як вирви, забруднення від осколків чи хімічних речовин. Тут використовується згорткова нейронна мережа (CNN) із технікою регуляризації Shortcut, для стабільного й ефективного навчання. Ця техніка дозволяє уникнути проблеми згасання градієнта, додаючи прямі зв’язки між шарами, що допомагають інформації легше проходити між шарами, щоб модель не втрачала важливі деталі.

Для активації шарів, а також для повнозв’язного шару, обрано функцію PReLU (Parametric Rectified Linear Unit) – вона розумніша за звичайну ReLU, бо сама підлаштовується, щоб навчання йшло швидше й без

збоїв. На виході модель видає ймовірності через функцію Softmax, що забезпечує коректну інтерпретацію ймовірностей класів. Після класифікації зображень на наступному етапі аналізуються зображення з класу «уражені ділянки». Це дозволяє одразу зрозуміти, чи варто аналізувати знімок детальніше. Архітектуру системи можна побачити на рисунку 2.1.

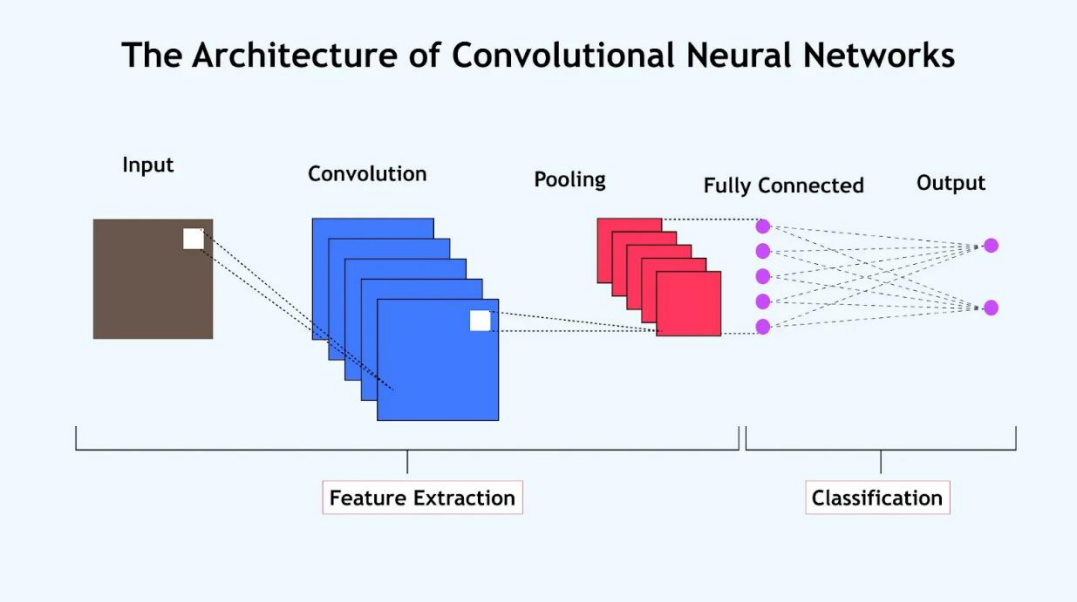

Рисунок 2.1 – Архітектура згорткової нейронної мережі

Створення моделі відбуватиметься за допомогою ML.NET – зручної бібліотеки від Microsoft, яка добре працює із C#. Але оскільки обробка зображень – справа складна, спочатку тренується модель у Python через TensorFlow, бо він має більше інструментів для таких завдань. Потім готова модель переводиться у формат ONNX і підключається до C#, щоб вона швидко працювала прямо в програмі.

Знімки для аналізу беруться із супутників Sentinel-2 (особливо смугу B4, яка добре показує зміни в рослинності) або з дронів [3]. Перед тим, як пустити зображення в модель, вони проходять попередню підготовку: нормалізацію для вирівнювання яскравості та контрастності, а також

аугментацію (наприклад, повороти чи зміну масштабу) для покращення узагальнюючої здатності моделі та її стійкості до варіацій у даних.

Ця модель інтегрується в систему як перший етап аналізу. Користувач завантажує зображення через інтерфейс WPF, і модель за лічені секунди видає результат: чи є на полі пошкодження. Якщо ділянка класифікується як «уражені», зображення передається до наступної моделі для детальнішого аналізу. Результати зберігаються у CSV-файлі для подальшого використання або передачі до бази даних, якщо система масштабуватиметься.

2.3 Модель для сегментації пошкоджень

Друга модель відповідає за сегментацію і призначена для детального аналізу зображень, класифікованих як «уражені», щоб визначити точне розташування, форму, площу та інтенсивність пошкоджень. Для цього впроваджено архітектуру U-Net – згорткову нейронну мережу з енкодер-декодерною структурою, яка широко застосовується для семантичної сегментації, зокрема в задачах аналізу супутникових зображень. U-Net складається з енкодера, який поступово зменшує розмірність зображення, і декодера, що відновлює просторову інформацію, дозволяючи точно визначити контури, площу, тип та рівень інтенсивності уражень ґрунту, що є критично важливим для подальшого аналізу та прийняття рішень. Дана архітектура ефективно виділяє контури уражених ділянок, таких як вирви чи забруднені зони, завдяки своїй здатності зберігати просторову інформацію через пропускні з’єднання між енкодером і декодером. Крім того, U-Net демонструє високу стійкість до шумів на зображеннях, що є особливо важливим при роботі з аерофотознімками та супутниковими даними низької якості. Завдяки цьому модель забезпечує точні результати навіть за умов обмеженої роздільної здатності вхідних зображень. Дану архітектуру представлено на рисунку 2.2.

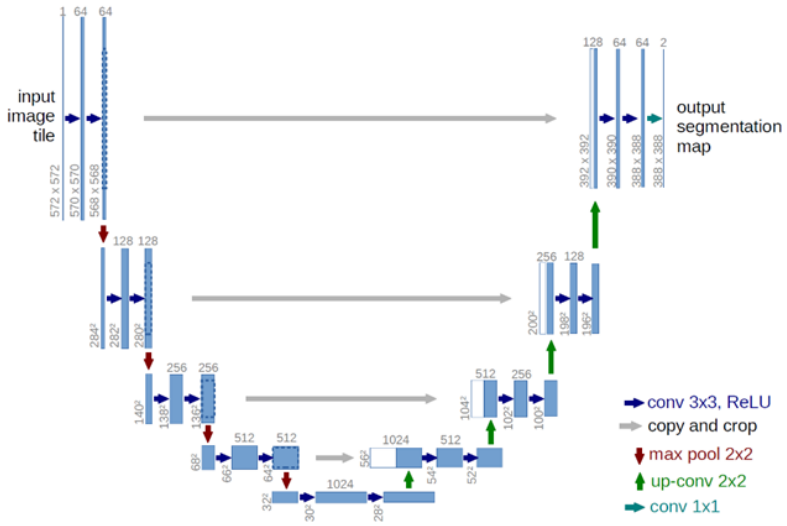

Рисунок 2.2 – Архітектура U-Net

Енкодерна частина U-Net складається з кількох згорткових шарів, які поступово зменшують розмірність зображення, виділяючи ключові ознаки, такі як краї вирв чи зміни в текстурі ґрунту. Декодерна частина відновлює зображення до початкової роздільної здатності, створюючи піксельну маску, де кожен піксель позначений як «пошкоджений» або «непошкоджений». Для активації шарів знову використовується PReLU, щоб забезпечити стабільне навчання.

Модель тренується в Python за допомогою PyTorch, оскільки ця бібліотека пропонує гнучкі інструменти для роботи з U-Net і обробки великих зображень. Для навчання беруться датасети, що включають супутникові знімки Sentinel-2 і аерофотознімки з дронів, з анотаціями, які позначають пошкоджені ділянки. Смуга B4 Sentinel-2 є особливо ефективною для виявлення змін у рослинності, тому використовуємо її як основний канал, додаючи смуги B2 і B3 для покращення контрастності [3]. Попередня обробка в цій моделі потребує формування датасету з фотографій пошкоджених земель та метаданих, таких як тип використаної зброї, площа ураження та час, необхідний для відновлення. Метадані структуруються та стандартизуються для забезпечення коректності

подальшого аналізу. Також виконується попередня обробка зображень, зокрема нормалізація яскравості та контрастності, видалення шуму за допомогою гаусівського фільтра, щоб уніфікувати дані.

Наступним кроком є тестування та валідація МШІ M_2. Використовуються метрики Mean Squared Error (MSE) для оцінки точності прогнозування часу, Intersection over Union (IoU) – для визначення площі ураження, а також Recall, Precision та F1-score для класифікації пошкоджень. Верифікація моделі M_2 виконується на тестовому наборі даних, включаючи раніше невикористані зображення, щоб оцінити її узагальнену здатність.

Концептуальний аналіз МШІ M_2 зосереджується на виявленні помилок, їх усуненні через донавчання та ансамблеві методи. Оцінюється вплив погодних умов і якості зображень, а також можливість інтеграції супутникових та дронових даних для покращення прогнозування.

Для оцінки якості сегментації застосовуються Mean Squared Error (MSE) для оцінки точності прогнозування часу, Intersection over Union (IoU) – для визначення площі ураження, а також Recall, Precision та F1-score для класифікації пошкоджень [4]. Ці метрики дозволяють переконатися, що модель точно виділяє межі пошкоджень із мінімальними помилками. Після тренування модель експортується у формат ONNX і інтегрується в C# через ONNX Runtime, що забезпечує швидку обробку зображень у реальному часі.

У системі ця модель працює як другий етап: після того, як перша модель визначила зображення як «уражені», U-Net створює детальну піксельну карту пошкоджень, чітко окреслюючи контури уражених зон. Ця карта відображається в інтуїтивному інтерфейсі WPF, де користувач може побачити білі області пошкоджень на чорному тлі разом із числовими даними про площу ураження в квадратних метрах або гектарах. Така візуалізація допомагає фермерам чи екологам швидко оцінити масштаби проблеми та планувати подальші дії.

2.4 Модель для прогнозу часу відновлення

Третя модель відповідає за прогнозування і оцінює, скільки часу знадобиться для відновлення землі після ураження зброєю, враховуючи такі фактори, як площа пошкодження, тип зброї (наприклад, вибухи, хімічне забруднення), тип ґрунту та кліматичні умови. Для цього було використано комбінацію двох архітектур: Temporal Fusion Transformer (TFT) для обробки статичних і різнорідних даних, таких як тип зброї чи геолокація, і Long Short-Term Memory (LSTM) для аналізу динамічних змін, наприклад, сезонних чи погодних факторів. Цей підхід дозволяє створювати точні прогнози навіть у динамічних умовах [4].

TFT – це гібридна модель, яка поєднує механізми уваги (attention) і обробку часових рядів. Ця модель продивляється поточні дані, наприклад, тип ґрунту чи місце розташування, і робить висновки. Вона ефективно аналізує статичні дані (наприклад, тип ґрунту чи координати поля) і змінні, що залежать від часу (наприклад, опади чи температура), створюючи узагальнені прогнози. LSTM, у свою чергу, відстежує довгострокові залежності, такі як поступове відновлення родючості ґрунту, що залежить від сезонних циклів. Разом ці моделі забезпечують точні оцінки часу рекультивації, враховуючи як поточний стан поля, так і його динаміку.

Модель тренується в Python за допомогою бібліотеки PyTorch, яка підтримує складні архітектури, такі як TFT і LSTM. Дані беруться з результатів сегментації (площа та інтенсивність пошкоджень) і додаємо інформацію з відкритих джерел, наприклад, тип ґрунту чи кліматичні показники, збережені у CSV-файлах. Щоб модель була точною, встановлюються різні налаштування за допомогою KerasTuner, щоб знайти найкращі значення для розміру шарів і швидкості навчання [4]. Для оцінки моделі використано Mean Absolute Error (MAE)), щоб перевірити, наскільки точно прогнозується час відновлення, а також R^2 для оцінки загальної якості моделі. Готову модель переведено в ONNX і підключається до C# через

ONNX Runtime для швидкого виконання прогнозів. У програмі користувач бачить звіт: скільки часу треба на відновлення і що можна зробити, наприклад, які методи рекультивації спрацюють найкраще. Усе це зберігається в CSV, щоб можна було повернутися до даних пізніше. Приклад вигляду даної комбінації представлено на рисунку 2.3.

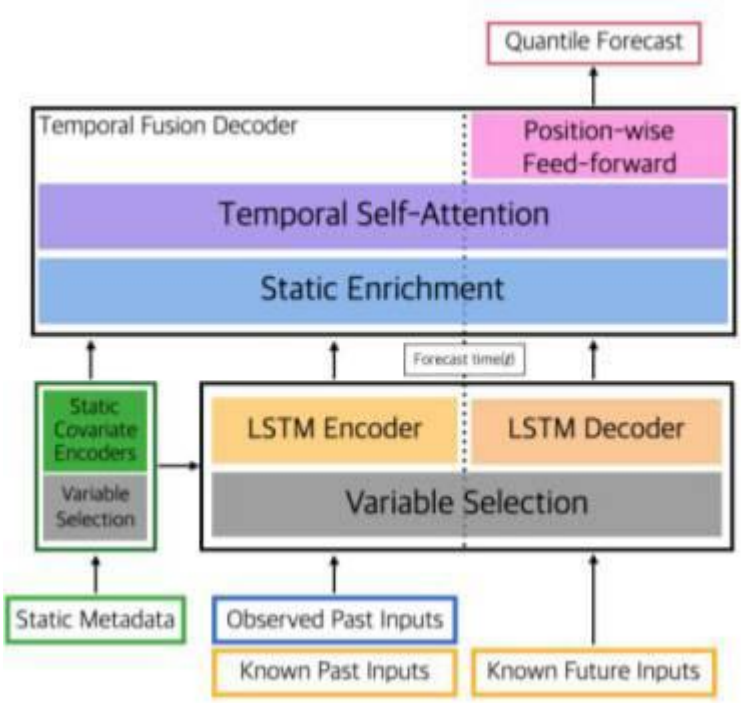

Рисунок 2.3 – Приклад комбінації двох архітектур: TFT та LSTM

У системі ця модель завершує цикл аналізу: після сегментації пошкоджень користувач отримує звіт із прогнозованим часом відновлення, який відображається в інтерфейсі WPF. Звіт включає рекомендації, наприклад, які методи рекультивації будуть найефективнішими, і зберігається у CSV-файлі для подальшого використання. Для підвищення точності прогнозів модель враховувала додаткові фактори, такі як кліматичні умови та тип ґрунту, отримані з відкритих джерел У майбутньому модель може бути вдосконалена шляхом інтеграції з базами даних, таких як Microsoft SQL Server, для роботи з великими обсягами даних і підтримки кількох користувачів.

2.5 Інтеграція моделей у систему

Усі три моделі об’єднуються в єдину систему підтримки прийняття рішень: перша модель класифікує зображення, друга створює детальну карту пошкоджень, а третя прогнозує час відновлення. Усе це користувач бачить у простому інтерфейсі на WPF: завантажив знімок, натиснув пару кнопок, запустив аналіз і отримав результат у вигляді карт, графіків і текстових звітів з порадами. Дані між частинами передаються через CSV-файли, що забезпечує простоту й гнучкість.

Для масштабування системи у майбутньому можна буде підключити базу даних, наприклад, Microsoft SQL Server, щоб зберігати більше інформації чи працювати з кількома користувачами. ML.NET і ONNX дозволяють усе робити швидко, а тренування в Python дає змогу пробувати нові ідеї. Така система не лише допомагає оцінити, що сталося з полями, а й закладає фундамент для більших проєктів, наприклад, щоб підключити дану програму до геоінформаційних систем чи аналізувати не лише поля, а й ліси чи інші землі.

2.6 Послідовність роботи моделей (класифікація → сегментація → прогноз)

У системі реалізовано послідовну обробку зображення, що забезпечує логічне проходження кожного етапу аналізу: від загального визначення наявності пошкоджень до побудови маски ураження та прогнозу можливих наслідків. Усі три моделі штучного інтелекту (M1, M2, M3) працюють узгоджено, кожна запускається лише після успішного завершення попередньої.

Робота починається із завантаження зображення користувачем, після чого система автоматично запускає класифікаційну модель (M1). Користувач завантажує зображення земельної ділянки, після чого модель

визначає, чи містить воно ознаки ураження. Якщо пошкодження відсутнє, система завершить аналіз, зекономивши ресурси. Якщо ж зображення класифікується як «уражене», автоматично активується другий етап.

У випадку виявлення пошкоджень активується модель сегментації (M2), яка формує піксельну маску, що окреслює межі ураженої зони. Ця маска дозволяє точно локалізувати ураження, розрахувати їхню площу та візуалізувати результат у вигляді карти. Маска зберігається у форматі PNG та відображається в інтерфейсі.

Третім етапом є прогнозування наслідків (M3). Ця модель оцінює ймовірні наслідки пошкодження, включаючи орієнтовну площу ураження та час, необхідний для відновлення ділянки. У розрахунках враховуються як результати сегментації, так і додаткові дані: тип ґрунту, кліматичні умови, сезонність, ймовірний тип пошкоджень. Результати прогнозу подаються у текстовому вигляді та додаються до фінального звіту.

Передача результатів між моделями реалізується через внутрішні змінні та допоміжні структури, у тому числі тимчасові файли. Усі етапи обробки є асинхронними, що дозволяє забезпечити плавну взаємодію з інтерфейсом і не блокувати роботу застосунку під час обчислень. Завдяки чіткому розмежуванню між модулями, послідовність роботи моделей зберігає гнучкість, дозволяючи в майбутньому доповнити її новими функціональними блоками.

Архітектура системи забезпечує повну автоматизацію аналізу зображень, дозволяючи швидко обробляти дані та формувати практичні висновки й рекомендації для користувачів, таких як фермери чи екологи. Усі етапи обробки інтегровані в єдиний процес у рамках системи підтримки прийняття рішень (DSS), що робить систему повноцінним аналітичним інструментом. Завдяки модульній структурі система залишається гнучкою, дозволяючи легко додавати нові функції, наприклад, інтеграцію з геоінформаційними системами.

2.7 Потік даних у системі та структура DSS

У системі було реалізовано чіткий і структурований потік даних, що забезпечував безперебійну обробку інформації та злагоджену взаємодію між усіма компонентами програми: інтерфейсом користувача, логікою обробки, моделями штучного інтелекту (M1, M2, M3) та модулем генерації звітів. Такий підхід дозволяв системі підтримки прийняття рішень (DSS) не лише проводити аналіз стану сільськогосподарських земель, а й надавати користувачу практичні рекомендації для подальших дій, роблячи процес інтуїтивно зрозумілим навіть для людей без технічної підготовки.

Потік даних у системі розпочинався із завантаження користувачем зображення земельної ділянки у форматі .jpg або .png через інтерфейс WPF. Після цього зображення автоматично передавалися до логічного ядра програми, яке координувало подальшу обробку. На першому етапі дані надходили до моделі M1, яка виконувала класифікацію, визначаючи, чи є на зображенні ознаки ураження, такі як вирви чи забруднення. Результат класифікації – мітка «уражена» або «неуражена» – зберігався у тимчасовому CSV-файлі та відображався в інтерфейсі як текстове повідомлення. Якщо ділянка визнавалася неураженою, обробка завершувалася, а користувачу пропонувалося завантажити нове зображення.

У разі виявлення уражень потік даних переходив до моделі M2, яка здійснювала сегментацію зображення за допомогою архітектури U-Net. Модель створювала піксельну маску, де білим кольором позначалися пошкоджені ділянки, а чорним – непошкоджені. Крім того, M2 обчислювала числові показники, такі як загальна площа ураження, які також зберігалися у CSV-файлі разом із шляхом до маски у форматі .png. Ці дані виводилися в інтерфейсі у вигляді графічної карти та текстових характеристик, що дозволяло користувачу одразу оцінити масштаби пошкоджень.

На завершальному етапі модель M3 отримувала результати сегментації (площу та інтенсивність уражень) разом із додатковими

метаданими, зібраними з відкритих джерел, наприклад, інформацією про тип ґрунту, кліматичні умови чи тип використаної зброї. Використовуючи комбінацію Temporal Fusion Transformer (TFT) і Long Short-Term Memory (LSTM), М3 прогнозувала час, необхідний для відновлення ділянки, наприклад, «2 роки» або «15 місяців». Прогноз супроводжувався рекомендаціями щодо рекультиваційних заходів, таких як очищення ґрунту чи внесення добрив. Усі результати зберігалися у CSV-файлі та відображалися в інтерфейсі у вигляді зрозумілого текстового звіту, що містив числові дані та практичні поради.

Структура DSS у системі була побудована за трирівневою архітектурою:

– рівень інтерфейсу користувача: забезпечував зручне завантаження зображень, відображення результатів у вигляді тексту, графічних карт і звітів. Інтерфейс був інтуїтивним, щоб фермери, екологи чи держслужбовці могли легко працювати з програмою без спеціальних знань;

– рівень логіки обробки: координував потік даних між моделями, виконував попередню обробку зображень (нормалізацію, фільтрацію шумів) і керував збереженням даних у CSV-файлах. Цей рівень також відповідав за контроль винятків, наприклад, перевірку якості зображень чи наявності уражень;

– рівень моделей ШІ: включав три моделі (М1, М2, М3), які обробляли дані та генерували результати. Моделі були інтегровані через ONNX Runtime, що забезпечувало швидку обробку в середовищі C#.

Для забезпечення надійності потоку даних у системі було реалізовано механізми контролю якості. Наприклад, якщо зображення було низької роздільної здатності або не відповідало вимогам, система видавала повідомлення про помилку, пропонуючи користувачу перевірити дані.

Така організація потоку даних у поєднанні зі структурою DSS робила ефективним інструментом для оцінки стану земель і планування їхнього відновлення. Система не лише аналізувала поточний стан ділянок, а й

надавала конкретні рекомендації, що допомагали фермерам і екологам приймати обґрунтовані рішення. У майбутньому потік даних міг бути вдосконалений шляхом інтеграції з базами даних, наприклад, Microsoft SQL Server, для роботи з більшими обсягами інформації або підключення до реальних супутникових потоків для аналізу в реальному часі.

2.8 Структура даних та зберігання

Структура даних розробленої системи організована таким чином, щоб забезпечити зручність обміну між компонентами, швидкий доступ до проміжних результатів та можливість формування фінального звіту без втрати інформації. Враховуючи поетапну обробку та модульність архітектури, дані зберігаються у вигляді окремих змінних у коді, тимчасових файлів, а також графічних об'єктів.

Після завантаження зображення його шлях зберігається у вигляді рядка (imagePath), а саме зображення завантажується у вигляді об’єкта BitmapSource для подальшої обробки в пам’яті. Результати класифікації (currentPrediction) і прогнозу (currentForecast) зберігаються окремо у вигляді текстових змінних, які відображаються у відповідних текстових полях інтерфейсу.

Маска, отримана в процесі сегментації, зберігається двічі: як об’єкт у пам’яті (currentMask типу BitmapSource) і як зображення у форматі .png, що дозволяє використати її в PDF/DOCX звіті. Усі ці об'єкти передаються до модуля генерації звітів у структурованому вигляді, без необхідності повторної обробки.

Для забезпечення узгодженості між модулями використовувалися тимчасові файлові структури, такі як CSV-файли, де зберігалися результати класифікації, площа пошкодження, координати маски та прогноз, що спрощувало логіку обробки та дозволяло легко масштабувати систему чи інтегрувати її з базами даних або хмарними сервісами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Використані технології та компоненти

Для створення системи було використано сучасний набір технологій і інструментів, що забезпечували ефективну реалізацію функціоналу машинного навчання, зручного десктопного інтерфейсу, обробки зображень і формування звітів. Архітектура системи була спроектована з акцентом на продуктивність, модульність і можливість масштабування, що дозволяло легко адаптувати її до нових вимог і забезпечувало стабільну роботу моделей штучного інтелекту (ШІ) у реальних умовах.

3.1.1 Інструменти Python (PyTorch, ONNX, Pandas)

Основним середовищем для створення, навчання та тестування моделей ШІ був Python, який завдяки своїй універсальності та широкій екосистемі бібліотек ідеально підійшов для роботи з машинним навчанням. Використовувалися такі ключові бібліотеки:

– PyTorch: ця бібліотека стала основою для побудови всіх трьох моделей системи – згорткової нейронної мережі (CNN) для класифікації (M1), архітектури U-Net для сегментації (M2) та комбінації Long Short-Term Memory (LSTM) і Temporal Fusion Transformer (TFT) для прогнозування (M3). PyTorch цінувався за гнучкість у роботі з динамічними обчислювальними графами, що спрощувало експерименти з архітектурою моделей, а також за підтримку експорту в універсальний формат ONNX, що забезпечував кросплатформеність;

– Pandas: використовувалася для обробки табличних даних, таких як кліматичні показники (температура, опади), характеристики ґрунтів чи метадані про типи уражень. Завдяки Pandas обробка великих обсягів даних

була швидкою та зручною, а результати зберігалися у структурованих CSV-файлах;

– NumPy: застосовувалася для виконання числових операцій, зокрема нормалізації та трансформації даних перед подачею їх у моделі;

– scikit-learn: використовувалася для крос-валідації моделей і оцінки їхньої якості за метриками, такими як Precision, Recall, F1-score (для класифікації та сегментації) і Mean Absolute Error (MAE) для прогнозування;

– Albumentations: ця бібліотека забезпечувала аугментацію зображень (обертання, зміна яскравості, додавання шумів), що дозволяло підвищити стійкість моделей до різних умов освітлення чи якості знімків.

Після завершення навчання моделі експортувалися у формат ONNX (Open Neural Network Exchange), що дозволяло використовувати їх у C# додатку без залежності від Python. Такий підхід забезпечував швидку інтеграцію моделей у десктопне середовище та оптимізував продуктивність системи.

3.1.2 Інструменти C# (.NET, WPF, ONNX Runtime)

Клієнтська частина була реалізована на платформі .NET 8.0, що гарантувала стабільність, сучасний функціонал і підтримку асинхронного програмування. Використовувалися такі технології:

– WPF (Windows Presentation Foundation): ця платформа стала основою для створення інтуїтивного графічного інтерфейсу користувача. WPF дозволяла будувати сучасні інтерфейси з підтримкою асинхронної обробки подій, що зберігало чуйність програми навіть під час виконання складних обчислень. У інтерфейсі відображалися зображення ділянок, маски сегментації, текстові результати класифікації та прогнози, а також елементи керування для завантаження файлів і перегляду звітів;

– ONNX Runtime for .NET: використовувалася для запуску моделей ШІ у форматі ONNX безпосередньо в C# середовищі. Це дозволяло

обходитися без Python залежностей, забезпечуючи високу швидкість виконання моделей і їхню інтеграцію з десктопним додатком. ONNX Runtime оптимізувала обчислення, використовуючи апаратне прискорення (CPU або GPU, якщо доступно), що робило обробку зображень швидкою навіть на стандартних комп'ютерах;

– System.Drawing та System.Windows.Media.Imaging: ці бібліотеки відповідали за обробку зображень, зокрема завантаження вхідних знімків, рендеринг масок сегментації та їх збереження у форматі .png для звітів. Вони забезпечували якісну візуалізацію результатів у реальному часі;

– асинхронне програмування (async/await): використовувалося для паралельного виконання обчислень (наприклад, обробки зображень моделями) і оновлення інтерфейсу, що дозволяло уникнути затримок і забезпечувало плавну взаємодію користувача з системою.

3.1.3 Зовнішні джерела даних (супутники, кліматичні API, відкриті набори)

Для повноцінної роботи моделі прогнозування (М3) та забезпечення точності аналізу система використовувала або передбачала інтеграцію з зовнішніми джерелами даних, що робило систему адаптованою до реальних умов:

– супутникові знімки: система була розрахована на роботу із зображеннями від Sentinel-2 і Landsat [5], які надають високоякісні дані для аналізу сільськогосподарських земель. Ці джерела могли використовуватися для отримання знімків у реальному часі або з архівів, що дозволяло відстежувати стан ділянок у динаміці;

– відкриті погодні API: для прогнозування часу відновлення земель використовувалися дані з таких сервісів, як Open-Meteo і Meteostat [6], які надавали часові ряди кліматичних показників (опади, температура,

вологість). Ці дані були критично важливими для моделі М3, оскільки впливали на оцінку рекультиваційних заходів;

– публічні аграрні датасети: для навчання моделей використовувалися набори даних, такі як FAO (Food and Agriculture Organization) і CropHarvest, що містили приклади уражених земель, типи ґрунтів і характеристики рослинності. Це дозволяло моделям «навчатися» на реальних сценаріях і підвищувало їхню точність.

Інтеграція зовнішніх джерел робила систему гнучкою і готовою до роботи в різних умовах, наприклад, у регіонах із різними кліматичними чи ґрунтовими характеристиками. У майбутньому систему можна було б розширити, додавши підтримку інших API (наприклад, для моніторингу забруднень чи аналізу хімічного складу ґрунтів) або підключивши реляційні бази даних для збереження історії аналізів.

Поєднання Python (для створення моделей) та C# (.NET + WPF для інтерфейсу та запуску моделей) забезпечило баланс між гнучкістю машинного навчання і стабільністю та зручністю десктопного застосунку. Обрані інструменти дозволили реалізувати комплексну, автономну систему, яка може працювати локально без підключення до серверів, а також масштабуватися до веб або хмарної архітектури в майбутньому.

3.2 Класифікація пошкоджених ділянок (M1)

Модель M1 відповідає за початкову обробку зображення, тобто за визначення того, чи є на ньому пошкоджена ділянка. Ця класифікація дозволяє ефективно відсіювати неінформативні зображення, економлячи обчислювальні ресурси для подальшої обробки лише зображень із ознаками ураження. Побудована на основі згорткової нейронної мережі (CNN) із технікою регуляризації Shortcut і функцією активації PReLU, модель забезпечує високу точність навіть на знімках із різними умовами освітлення.

3.2.1 Архітектура CNN-моделі

Для класифікації зображень було використано згорткову нейронну мережу (CNN), побудовану на основі ResNet-18 [7]. Архітектура була модифікована для роботи з невеликим набором зображень і забезпечення високої точності:

– змінено останній шар: fc = nn.Linear(512, 2);

– додано Dropout перед повнозв'язним шаром для зменшення перенавчання;

– використано PReLU як активацію для адаптації до темних зон.

Програмний код для навчання моделі класифікації наведено у лістингу 3.1.

Лістинг 3.1 – Програмний код для навчання моделі класифікації

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
model =
models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K
_V1)
model.fc = nn.Sequential(
    nn.Dropout(0.5),  # Add Dropout
    nn.Linear(model.fc.in_features, NUM_CLASSES)
)
device = torch.device("cuda" if torch.cuda.is_available()
else "cpu")
model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001,
weight_decay=1e-4)
```

3.2.2 Навчання, оцінка, експорт ONNX

Модель тренувалася у середовищі PyTorch на невеликому датасеті: 36 зображень у train, 10 зображень у validation. Код навчання моделі наведено у додатку А. Через обмежену кількість даних було застосовано аугментації (RandomFlip, ColorJitter) та знижено learning rate (0.0001).

Ключові параметри:

– Loss: CrossEntropyLoss;

– Optimizer: Adam;

– Epoch: 15;

– Batch Size: 4;

– Augmentation: доступна в train_transforms;

– EarlyStopping: активований (patience=3).

Програмний код для циклів навчання представлено у лістингу 3.2.

Лістинг 3.2 – Програмний код для циклів навчання

```
for epoch in range(EPOCHS):
    model.train()
    running_loss = 0.0
    for images, labels in train_loader:
        images, labels = images.to(device),
labels.to(device)
        optimizer.zero_grad()
        output = model(images)
        loss = criterion(output, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    avg_train_loss = running_loss / len(train_loader)
    print(f"Epoch {epoch+1}/{EPOCHS}, Train Loss:
{avg_train_loss:.4f}")
```

Отримані результати демонструють загалом успішне навчання моделі. На валідаційному наборі було досягнуто 100% точності, що свідчить про високу ефективність класифікації саме на доступних прикладах. Мінімальне значення втрат на валідації (Val Loss) зафіксовано на 10-й епосі і це лише 0.0158, що є дуже низьким показником.

Такого результату вдалося досягти завдяки використанню регуляризації Dropout, а також аугментацій зображень, що дозволили штучно збільшити варіативність навчальних даних. Низький learning rate забезпечив стабільне навчання без різких коливань у значеннях втрат.

Однак важливо враховувати і обмеження. Найсуттєвішим є дуже невеликий розмір валідаційного набору – лише 10 зображень. Це створює ризик перенавчання, коли модель не узагальнює закономірності, а просто запам’ятовує конкретні приклади. Для остаточних висновків потрібне тестування на більшій та незалежній вибірці. Вже проаналізовані результати навчання моделі по епохах видно нижче на рисунках 3.1 та 3.2.

```
Epoch 1/15, Train Loss: 0.6888
Val Loss: 0.3277, Val Accuracy: 80.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 2/15, Train Loss: 0.3701
Val Loss: 0.1979, Val Accuracy: 90.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 3/15, Train Loss: 0.0945
Val Loss: 0.1474, Val Accuracy: 90.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 4/15, Train Loss: 0.1298
Val Loss: 0.1481, Val Accuracy: 80.00%
Epoch 5/15, Train Loss: 0.0382
Val Loss: 0.1138, Val Accuracy: 100.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 6/15, Train Loss: 0.0171
Val Loss: 0.0819, Val Accuracy: 100.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 7/15, Train Loss: 0.5185
Val Loss: 0.0479, Val Accuracy: 100.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 8/15, Train Loss: 0.0557
Val Loss: 0.0341, Val Accuracy: 100.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 9/15, Train Loss: 0.0082
Val Loss: 0.0200, Val Accuracy: 100.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 10/15, Train Loss: 0.0111
Val Loss: 0.0158, Val Accuracy: 100.00%
Збережено кращі ваги в m1_model_weights.pth
```

Рисунок 3.1 – Результати навчання моделі класифікації

```
Epoch 11/15, Train Loss: 0.0432
Val Loss: 0.0149, Val Accuracy: 100.00%
Збережено кращі ваги в m1_model_weights.pth
Epoch 12/15, Train Loss: 0.1109
Val Loss: 0.0192, Val Accuracy: 100.00%
Epoch 13/15, Train Loss: 0.0063
Val Loss: 0.0274, Val Accuracy: 100.00%
Epoch 14/15, Train Loss: 0.0273
Val Loss: 0.0290, Val Accuracy: 100.00%
Early stopping triggered!
Модель експортовано в m1_model.onnx
```

Рисунок 3.2 – Друга частина результатів навчання моделі класифікації

3.2.3 Підключення в C# через OnnxRuntime

Після успішного навчання та експорту моделі у формат ONNX, вона була інтегрована в десктопний застосунок на мові C# із використанням бібліотеки Microsoft.ML.OnnxRuntime. Цей формат дозволяє виконувати нейронну мережу незалежно від середовища навчання (наприклад, без запуску Python), що ідеально підходить для .NET-додатків.

У програмі реалізовано окремий клас OnnxModelRunner, відповідальний за завантаження моделі, передобробку зображення та отримання результату класифікації. Модель читається з файлу m1_model.onnx, який розміщується в директорії Models/. Програмний код для завантаження та запуску Onnx моделі представлено у лістингу 3.3.

Лістинг 3.3 – Програмний код для завантаження та запуску Onnx моделі

```
using Microsoft.ML.OnnxRuntime;
using Microsoft.ML.OnnxRuntime.Tensors;
using System;
using System.Collections.Generic;
```

Продовження лістингу 3.3

```
using System.Drawing;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

public class OnnxModelRunner
{
    private readonly InferenceSession _session;
    private readonly int _inputWidth;
    private readonly int _inputHeight;
    public OnnxModelRunner(string modelPath)
    {
        _session = new InferenceSession(modelPath);
        var inputMeta = _session.InputMetadata["input"];
        _inputHeight = inputMeta.Dimensions[2];
        _inputWidth = inputMeta.Dimensions[3];
    }
    public string Predict(string imagePath)
    {
        var input = PreprocessImage(imagePath);
        var inputs = new List<NamedOnnxValue>
        {
            NamedOnnxValue.CreateFromTensor("input",
input)
        };
        using var results = _session.Run(inputs);
        var output =
results.First().AsEnumerable<float>().ToArray();
        int predictedIndex = Array.IndexOf(output,
output.Max());
        return predictedIndex == 0 ? "⚠ Уражена ділянка"
: "✅ Неуражена ділянка";
    }
}
```

Зображення, яке завантажується користувачем, спочатку конвертується у формат тензора (Tensor<float>). Перед подачею в модель воно:

– масштабується до розміру 224×224 пікселі;

– нормалізується (перетворюється у значення від 0 до 1);

– переводиться з формату Bitmap до багатовимірного масиву DenseTensor<float>.

Передобробка реалізується в окремому методі PreprocessImage, який також входить до складу OnnxModelRunner.cs. Повний код внесення моделі у C# код наведено у додатку Б.

Результат у вигляді двох чисел (ймовірностей для класу «уражена» і «неуражена») зчитується з виходу моделі. Далі проводиться порівняння: якщо ймовірність класу «уражена» вища – виводиться відповідне повідомлення. Це повідомлення відображається у інтерфейсі WPF за допомогою елемента ResultTextBlock. Код для цього має вигляд:

```
ResultTextBlock.Text = modelRunner.Predict(imagePath);
```

Також результат зберігається у змінній currentPrediction, що дає змогу використати його пізніше – наприклад, у звітах у форматі PDF або DOCX.

Цей модуль забезпечує швидке та стабільне розгортання моделі класифікації прямо у графічному інтерфейсі, без необхідності мати Python або сторонні інтерпретатори. Завдяки використанню OnnxRuntime, передбачення виконується протягом часток секунди навіть на звичайних офісних комп’ютерах.

3.3 Сегментація зображень (M2)

Сегментація пошкоджених ділянок є другим етапом в системі. Після класифікації зображення як «ураженого» за допомогою моделі M1, система переходить до побудови точного просторового окреслення ураженої зони. Цей процес виконується за допомогою моделі M2, яка реалізована на основі

архітектури U-Net – однієї з найпоширеніших і найрезультативніших структур для семантичної сегментації [8], особливо у задачах медичної та супутникової обробки зображень.

3.3.1 Архітектура U-Net

Модель U-Net складається з двох частин: стискуючої (енкодер) та розширюючої (декодер). В енкодері вхідне зображення проходить через кілька рівнів згорткових шарів із функцією активації PReLU, після чого стискається за допомогою операції max pooling. Це дозволяє зменшити розмір зображення, зберігаючи при цьому найбільш значущі ознаки. Декодер виконує зворотну операцію – поступово відновлює просторову роздільну здатність за допомогою транспонованих згорткових шарів (ConvTranspose2d) і об’єднує (через skip connections) відповідні рівні з енкодера. У фіналі модель формує карту пікселів, де кожен піксель має значення від 0 до 1 – ймовірність, що він належить до пошкодженої зони.

Реалізація архітектури в проєкті була виконана за допомогою фреймворку PyTorch. Код навчання моделі сегментації наведено у додатку В. Клас моделі було збудовано вручну, без використання зовнішніх бібліотек, що дозволяє повністю контролювати структуру і її змінюваність у майбутньому.

3.3.2 Обробка зображень і побудова маски

На етапі навчання для кожного зображення, що міститься в каталозі, виконується попередня обробка. Всі зображення масштабуються до фіксованого розміру 128×128 пікселів – це оптимальний розмір для швидкого навчання на малих об’ємах даних. Після цього виконується нормалізація: значення пікселів діляться на 255, щоб привести їх до діапазону [0, 1], що є стандартом для нейронних мереж.

Поряд із зображенням обробляється і відповідна маска – це чорно-біле зображення з тією ж роздільною здатністю, де білим кольором позначені пошкоджені області, а чорним – фон. Маска також масштабуються до 128×128, конвертується у одноканальний тензор [1, H, W] та нормалізується.

Далі зображення та маска передаються у модель у вигляді тензорів типу torch.Tensor. U-Net обробляє зображення, проходячи крізь усі згорткові та декодувальні шари, після чого формує вихідну карту ймовірностей, де кожному пікселю призначається значення від 0 до 1, яке інтерпретується як ймовірність того, що цей піксель належить до пошкодженої ділянки.

В результаті сегментації формується бінарна маска, яка зберігається в окремий файл (segmented_mask.png) і водночас передається до інтерфейсу користувача в WPF для візуалізації. Повний код навчання моделі для генерації масок наведено у додатку Г. У WPF-модулі ця маска прив’язується до елементу MaskImage, де користувач може переглянути сегментовану ділянку прямо на екрані.

Функціональною реалізацією цього етапу в C# є клас UnetModelRunner, що відкриває модель unet_model.onnx, проводить необхідні перетворення зображення, запускає модель та повертає об’єкт типу Bitmap, який далі передається у Image.Source. Код завантаження моделі у C# частину програми:

```
var seg = new UnetModelRunner("Models/unet_model.onnx");
var mask = segmenter.Segment(imagePath);
MaskImage.Source = BitmapFromBitmap(mask); // показ у WPF
```

Таким чином, уся логіка повністю автоматизована та прихована від користувача. Код реалізації моделі у C# додатку наведено у додатку Д.

Під час навчання моделі на обмеженому наборі даних модель демонструвала поступове покращення. Втрати на тренувальному наборі (loss) зменшилися з 4.18 у першій епосі до 2.11 у десятій. Хоча значення втрат трохи коливалися. Як приклад, від 2.9 до 3.3, а потім до 2.2.

Але загальна динаміка була позитивною. Це підтверджує, що навіть на малому наборі даних мережа здатна навчитися узагальнювати просторові патерни, характерні для пошкоджених ділянок. Вже проаналізовані результати навчання моделі по епохах видно нижче на рисунку 3.3.

```
Навчання моделі U-Net...
100%|##########| 6/6 [00:16<00:00,  2.72s/it]
Епоха 1, Втрата: 4.1852
100%|##########| 6/6 [00:06<00:00,  1.12s/it]
  0%|          | 0/6 [00:00<?, ?it/s]Епоха 2, Втрата: 3.7997
100%|##########| 6/6 [00:07<00:00,  1.21s/it]
  0%|          | 0/6 [00:00<?, ?it/s]Епоха 3, Втрата: 2.9014
100%|##########| 6/6 [00:22<00:00,  3.75s/it]
  0%|          | 0/6 [00:00<?, ?it/s]Епоха 4, Втрата: 3.3143
100%|##########| 6/6 [00:12<00:00,  2.11s/it]
  0%|          | 0/6 [00:00<?, ?it/s]Епоха 5, Втрата: 3.3679
100%|##########| 6/6 [00:10<00:00,  1.71s/it]
  0%|          | 0/6 [00:00<?, ?it/s]Епоха 6, Втрата: 3.2461
100%|##########| 6/6 [00:09<00:00,  1.60s/it]
  0%|          | 0/6 [00:00<?, ?it/s]Епоха 7, Втрата: 2.6442
100%|##########| 6/6 [00:08<00:00,  1.44s/it]
  0%|          | 0/6 [00:00<?, ?it/s]Епоха 8, Втрата: 2.2143
100%|##########| 6/6 [00:08<00:00,  1.36s/it]
  0%|          | 0/6 [00:00<?, ?it/s]Епоха 9, Втрата: 2.6495
100%|##########| 6/6 [00:08<00:00,  1.35s/it]
Епоха 10, Втрата: 2.1142
```

Рисунок 3.3 – Результати навчання моделі U-Net

Після завершення навчання модель експортувалася у формат ONNX для використання у .NET-застосунку. Програмний код для збереження моделі в Onnx представлено у лістингу 3.4.

Лістинг 3.4 – Програмний код для збереження моделі в Onnx

```
dummy_input = torch.randn(1, 3, IMG_SIZE,
IMG_SIZE).to(DEVICE)
torch.onnx.export(model, dummy_input, "unet_model.onnx",
```

Продовження лістингу 3.4

```
                    input_names=["input"],
output_names=["output"],
                    dynamic_axes={"input": {0: "batch"},
"output": {0: "batch"}},
                    opset_version=11)
```

Файл unet_model.onnx зберігається у директорії Models/ і використовується в десктопному застосунку на етапі сегментації.

3.4 Прогноз часу відновлення (M3)

Після класифікації та сегментації, третім завершальним етапом обробки є прогноз часу відновлення пошкодженої ділянки. Завданням моделі M3 є регресійний аналіз на основі результатів сегментації, а також зовнішніх даних, таких як тип ґрунту, кліматичні умови та характер пошкоджень. Результатом є орієнтовний час, необхідний для повного відновлення земельної ділянки, представлений у місяцях або роках. Цей етап побудований на складній архітектурі, що поєднує можливості TFT (Temporal Fusion Transformer) і LSTM (Long Short-Term Memory).

3.4.1 Комбінована архітектура TFT + LSTM

Модель була реалізована у фреймворку PyTorch як комбінація двох підходів до аналізу часових рядів:

– LSTM відповідає за обробку послідовних змін параметрів у часі. Наприклад, зміни вологості, опадів або температури, що впливають на швидкість відновлення ґрунту;

– TFT (Temporal Fusion Transformer) забезпечує обробку як динамічних, так і статичних вхідних ознак із використанням механізму

уваги (attention), що дає змогу фокусуватися на ключових змінах у часовому ряді та адаптивно зважувати інформацію.

Архітектура включає кілька входів:

– нормалізовану площу ураження (із сегментації);

– кліматичні характеристики (метеодані за останні 12 місяців);

– тип ґрунту (категоріальний вхід);

– тип зброї (означений за площинними шаблонами або за класифікатором).

Ці входи об’єднуються в єдиний багатовимірний тензор, який подається на вхід LSTM та декодується TFT-блоком. Програмний код для реалізації комбінації LSTM та TFT представлено у лістингу 3.5.

Лістинг 3.5 – Програмний код для реалізації комбінації LSTM та TFT

```
class SimpleLSTMTransformer(nn.Module):
    def __init__(self, input_size, hidden_size,
output_size=1):
        super(SimpleLSTMTransformer, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size,
batch_first=True)
        self.linear = nn.Linear(hidden_size, hidden_size)
        self.transformer = nn.TransformerEncoder(

nn.TransformerEncoderLayer(d_model=hidden_size, nhead=2,
batch_first=True),
            num_layers=1
        )
        self.output = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x, _ = self.lstm(x)
        x = self.linear(x)
        x = self.transformer(x)
        return self.output(x[:, -1, :])
```

3.4.2 Обробка динамічних і статичних ознак

На вході модель отримує комбінацію статичних параметрів (тип ґрунту, геолокація) та динамічних часових рядів (наприклад, вологість, температура, кількість опадів, площа пошкоджень у часі). Дані було згенеровано шляхом моделювання та симуляції, з урахуванням реальних шаблонів клімату.

Всі вхідні значення масштабувалися до діапазону [0, 1] за допомогою MinMaxScaler із бібліотеки sklearn. Для time-series входу було сформовано масив розмірності (batch, time_steps, features), де time_steps=50 і features=2.

Подача даних у модель виконувалася наступним чином:

```
input_size = X_tensor.shape[2]
model = SimpleLSTMTransform(size=input_size, hid_size=32)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)
```

3.4.3 Побудова регресійної моделі

Навчання моделі проводилося протягом 150 епох. У процесі втрати зменшувалися поступово – з початкового значення 10.8050 до мінімуму 0.8116 на 120-й епосі. Зниження втрат демонструє успішне навчання моделі. Графік втрат мав деякі коливання (наприклад, зростання на 140-й епосі до 1.08), однак загальний тренд залишався низхідним. Повний код навчання моделі прогнозування наведено у додатку Е.

Результати навчання комбінованої моделі прогнозування свідчать про поступове покращення здатності моделі робити точні прогнози. Хоча в останні епохи втрати трохи зросли, модель вже досягла хороших значень точності, і зростання могло бути зумовлене перенавчанням або обмеженим обсягом даних.

Вигляд результатів та процес навчання моделі представлено на рисунку 3.4.

```
Epoch 0, Loss: 10.8050
Epoch 10, Loss: 6.4145
Epoch 20, Loss: 4.6539
Epoch 30, Loss: 3.7811
Epoch 40, Loss: 3.9530
Epoch 50, Loss: 3.2916
Epoch 60, Loss: 2.9025
Epoch 70, Loss: 1.9986
Epoch 80, Loss: 2.0203
Epoch 90, Loss: 1.4920
Epoch 100, Loss: 1.0534
Epoch 110, Loss: 0.9757
Epoch 120, Loss: 0.8116
Epoch 130, Loss: 0.9940
Epoch 140, Loss: 1.0839
```

Рисунок 3.4 – Результати навчання комбінованої моделі прогнозування

Після навчання модель експортується у формат ONNX, щоб інтегрувати її у C# додаток. Код збереження моделі представлено у лістингу 3.6.

Лістинг 3.6 – Програмний код для збереження навченої моделі прогнозування

```
model.eval()
dummy_input = torch.randn(1, 1, input_size)
torch.onnx.export(
    model, dummy_input, "model_lstm_tft.onnx",
    input_names=["input"], output_names=["output"],
    dynamic_axes={"input": {0: "batch"}, "output": {0:
"batch"}},
    opset_version=14
)
```

Отримане значення – прогнозований час відновлення у місяцях. Це число автоматично передається до інтерфейсу користувача і зберігається у змінній currentForecast. Після чого результат зберігається у звіт. Код використання моделі прогнозування у C# додатку надано у лістингу 3.7.

Лістинг 3.7 – Програмний код використання моделі прогнозування у C# додатку

```
public float[] Predict(float[,] inputSequence)
{
    int batch = inputSequence.GetLength(0);
    int featureDim = inputSequence.GetLength(1);

    var tensor = new DenseTensor<float>(new[] { batch, 1,
featureDim });

    for (int i = 0; i < batch; i++)
    {
        for (int j = 0; j < featureDim; j++)
        {
            tensor[i, 0, j] = inputSequence[i, j];
        }
    }
    var inputs = new List<NamedOnnxValue>
    {
        NamedOnnxValue.CreateFromTensor("input", tensor)
    };
    using var results = _session.Run(inputs);
    var outputTensor = results.First().AsTensor<float>();

    return outputTensor.ToArray();
}
```

Повний код реалізації моделі прогнозування наведено у додатку Ж.

4 ІНТЕРФЕЙС КОРИСТУВАЧА ТА ТЕСТУВАННЯ МОДЕЛЕЙ

4.1 Інтерфейс користувача (UI)

Система реалізована як WPF-додаток, побудований на мові C# з використанням платформи .NET. Головне вікно додатку побудоване за принципом трирівневої логіки: ліворуч розміщені кнопки керування, а праворуч – вікна для відображення результатів обробки.

Після запуску додатку користувач бачить панель з такими основними етапами:

– завантаження зображення – відкриває діалогове вікно для вибору фото;

– класифікація (M1) – визначає, чи зображення є ураженим;

– сегментація (M2) – при потребі створює маску пошкоджень;

– прогноз (M3) – оцінює орієнтовний час відновлення;

– формування звіту – дозволяє зберегти результати у форматі PDF або DOCX.

Код для кнопки класифікації, яка є однією з основних 5 кнопок, наведено у лістингу 4.1.

Лістинг 4.1 – Програмний код для кнопки класифікації

```
private async void Classify_Click(object sender,
RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(imagePath)) {
        ResultTextBlock.Text = "❗ Спочатку завантажте
зображення!";
        return;
    }

    LoadingOverlay.Visibility = Visibility.Visible;
```

Продовження лістингу 4.1

```
    await Task.Delay(300);
    try
    {
        var classifier = new
OnnxModelRunner("Models/m1_model.onnx");
        var result = await Task.Run(() =>
classifier.Predict(imagePath));
        currentPrediction = result;
        ResultTextBlock.Text = result;
        if (result.Contains("Уражена"))
        {
            SegmentButton.Visibility = Visibility.Visible;
            PredictButton.Visibility = Visibility.Visible;
        }
        else
        {
            SegmentButton.Visibility =
Visibility.Collapsed;
            PredictButton.Visibility =
Visibility.Collapsed;
            MaskImage.Source = null;
            ForecastTextBlock.Text = "";
        }
    }
    catch (Exception ex)
    {
        ResultTextBlock.Text = $"⚠ Помилка класифікації:
{ex.Message}";
    }
    finally
    {
        LoadingOverlay.Visibility = Visibility.Collapsed;
    }
}
```

Усі результати обробки відображаються в інтерфейсі у вигляді тексту та графіки (маска, карта пошкоджень, числові значення). Всі етапи виконуються покроково, і кожен з них вмикає наступну кнопку, лише якщо попередній пройдено успішно. Повний код функціональності та візуальності інтерфейсу наведено у додатках И та К відповідно.

На рисунку 4.1 показано інтерфейс програми після закінчення всіх процесів обробки зображення.

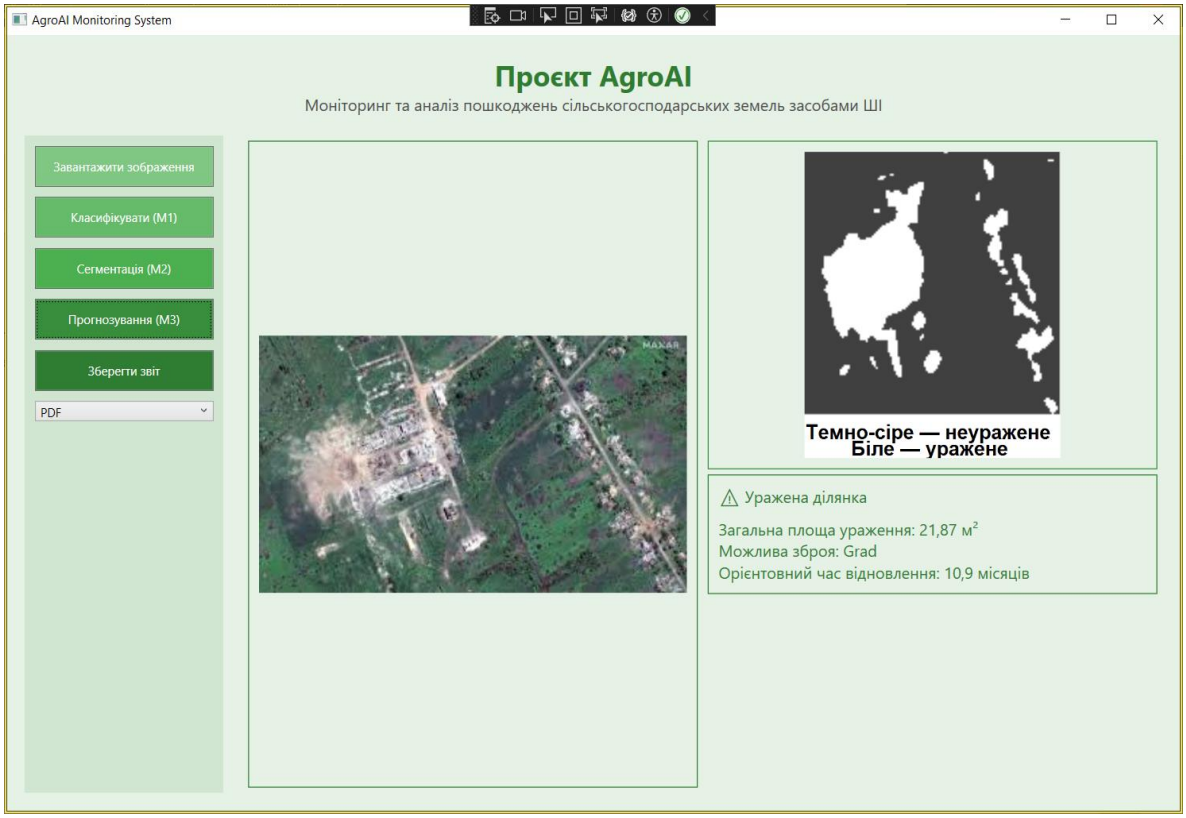

Рисунок 4.1 – Інтерфейс програми після закінчення всіх процесів обробки зображення

4.2 Інтеграція моделей у систему підтримки прийняття рішень (DSS)

Система реалізована як локальний модуль підтримки прийняття рішень (DSS), що дозволяє спеціалістам – агрономам, екологам або фермерам – оперативно отримати комплексну оцінку стану земельної

ділянки, яка зазнала пошкоджень. Завдяки інтеграції трьох моделей штучного інтелекту, процес аналізу побудований логічно та послідовно: від первинної класифікації до отримання прогнозу відновлення.

На першому етапі модель класифікації (M1) визначає, чи містить зображення ознаки ураження. Якщо пошкодження виявлено, система автоматично переходить до другого етапу, де модель сегментації (M2) будує детальну карту пошкоджених зон і обчислює площу ураження. Після цього активується третя модель (M3), яка аналізує отримані дані разом із додатковими характеристиками ділянки (наприклад, тип ґрунту чи погодні умови) та формує прогноз часу, необхідного для її повного відновлення.

Усі результати цих етапів – включно з визначеним типом пошкодження, оціненою площею ураження, можливим типом застосованої зброї та прогнозованим терміном відновлення – об’єднуються в єдиний звіт. Цей звіт також містить карту пошкоджень у вигляді сегментованого зображення, що надає користувачу не лише числову інформацію, а й візуальне представлення наслідків. У підсумку користувач отримує чітку й структуровану аналітичну довідку, яка може бути збережена у форматі PDF або Word і використана для подальших дій, включаючи планування рекультиваційних заходів.

4.3 Тестування

Тестування моделей, що входять до складу системи, проводилось окремо для кожного з етапів аналізу, щоб забезпечити максимальну контрольованість результатів та ідентифікувати можливі слабкі місця ще на ранніх стадіях.

На першому етапі була протестована модель класифікації (M1), яка працювала з набором із 46 зображень, поділених на тренувальну та валідаційну частини. Під час навчання модель досягла 100% точності на валідаційному наборі, що формально свідчить про її високу ефективність.

Однак важливо враховувати, що обсяг даних був обмеженим, і така точність може бути наслідком переобучення – модель могла запам’ятати приклади замість того, щоб навчитися узагальнювати закономірності.

Для моделі сегментації (М2) оцінка якості проводилася візуально: отримані маски накладалися на відповідні зображення, і вручну перевірялося, наскільки точно модель відтворює контури пошкоджених ділянок. У більшості випадків маски виявилися інформативними, а межі уражених зон – чіткими, що дозволяє з впевненістю використовувати їх у подальшому аналізі. Проте ефективність сегментації виявлялася чутливою до якості вхідного зображення: на зображеннях зі слабким контрастом або складною текстурою точність зменшувалась.

Тестування регресійної моделі прогнозу (М3) проводилося на синтетично згенерованих часових рядах, що дозволило відстежити зміну втрат під час навчання. Протягом 150 епох спостерігалося стабільне зниження функції втрат – з початкового значення близько 10 до фінального рівня нижче одиниці. Це вказує на поступове покращення якості прогнозу. Отримані результати узгоджувалися з логікою: при більшій площі пошкодження модель прогнозувала триваліший час відновлення, що підтверджує її коректну поведінку. Водночас слід враховувати, що модель не враховує важливі зовнішні чинники, зокрема економічні ресурси, політичні рішення чи наявність техніки для рекультивації – її прогноз базується виключно на фізичних характеристиках.

У процесі тестування було виявлено низку обмежень. Зокрема, класифікаційна модель через малий обсяг навчальних даних схильна до переобучення. Якість сегментації залежить від вхідного зображення – в ідеальних умовах вона висока, але може погіршуватися при зміні умов зйомки. А модель прогнозу, попри хороші результати на тестових даних, потребує подальшої адаптації для реальних сценаріїв, де до фізичних параметрів додаються соціальні та економічні фактори.

ВИСНОВКИ

Результатом розробки є інтелектуальна система, яка стала відповіддю на нагальну потребу автоматизованої оцінки стану сільськогосподарських земель, пошкоджених через бойові дії. Робота над проєктом включала ґрунтовне дослідження предметної області, аналіз сучасних методів обробки супутникових і аерофотознімків, а також вивчення викликів, із якими стикаються фермери, екологи та аграрії в умовах війни. Це допомогло не лише зрозуміти проблему, а й запропонувати рішення, яке реально працює та враховує потреби користувачів.

Окрім того, застосування таких інструментів сприяє ефективнішому плануванню використання земельних ресурсів і прискоренню процесів рекультивації, що напряму підтримує досягнення цілі сталого розвитку 15 – «Захист і відновлення екосистем суші». Це підкреслює значення технологій штучного інтелекту не лише в контексті оперативного реагування на воєнні наслідки, а й як стратегічного інструменту для довгострокового сталого розвитку сільського господарства [16].

Система була побудована навколо трьох ключових етапів обробки зображень із застосуванням моделей штучного інтелекту:

– класифікація пошкоджень за допомогою згорткової нейронної мережі (CNN);

– сегментація уражених ділянок із використанням архітектури U-Net;

– прогнозування часу відновлення земель на основі комбінації Temporal Fusion Transformer (TFT) і Long Short-Term Memory (LSTM).

Така послідовність дозволяла системі:

– швидко визначати, чи є на зображенні пошкодження;

– створювати детальну карту уражених зон із точними контурами;

– прогнозувати термін відновлення з урахуванням площі ураження, типу ґрунту, кліматичних умов і характеру пошкоджень;

– формувати зручний звіт у форматі PDF або DOCX, який користувач міг одразу використати для планування рекультивації.

Моделі ШІ були розроблені в середовищі Python із застосуванням бібліотек PyTorch і ONNX, що забезпечило їхню гнучкість і високу точність. Для інтеграції в десктопний додаток на C# із сучасним інтерфейсом WPF використано ONNX Runtime, що дозволило запускати моделі без залежності від Python і забезпечило швидку обробку даних. Інтерфейс програми був спроектований так, щоб бути інтуїтивно зрозумілим: результати класифікації, маски сегментації та прогнози відображалися в реальному часі, а керування системою не вимагало спеціальних технічних знань.

Дослідження підтвердило, що штучний інтелект є потужним інструментом для аналізу великих обсягів візуальних даних, особливо в умовах обмеженого доступу до територій через мінування чи активні бойові дії. Навіть із невеликими наборами даних система досягала високої точності завдяки продуманій аугментації зображень, регуляризації моделей і використанню сучасних архітектур нейронних мереж. Це стало важливим висновком, адже в реальних умовах доступ до великих обсягів даних часто обмежений.

Розробка враховувала практичні потреби кінцевих користувачів:

– потребу в швидкій оцінці стану полів без ризикованих виїздів на місця;

– обмеження в доступі до точних даних через воєнні обставини;

– запит на просту та зручну програму, якою могли б користуватися фермери, екологи чи представники місцевих адміністрацій.

Для реалізації системи було обрано технології, що поєднували гнучкість і масштабованість: платформа .NET із WPF для створення зрозумілого та сучасного інтерфейсу, ONNX Runtime для швидкої та ефективної роботи моделей штучного інтелекту, а також CSV-файли для структурованого зберігання й обміну даними. Такий підхід зробив систему

модульною, надійною та готовою до подальшого розвитку, дозволяючи легко адаптувати її до нових вимог чи інтегрувати з іншими платформами.

У процесі роботи над проєктом сформувалося чітке розуміння вимог до програмного продукту в рамках дипломної роботи: він мав бути модульним, щоб підтримувати додавання нових функцій; надійним, щоб уникати помилок у критичних сценаріях; зручним для користувачів без спеціальної підготовки; і, головне, аналітично глибоким, щоб надавати цінну інформацію для прийняття обґрунтованих рішень. Особлива увага приділялася простоті використання, адже цільовою аудиторією були не лише технічні спеціалісти, а й фермери, екологи та представники місцевих адміністрацій, які потребували швидких і зрозумілих результатів.

У підсумку, розроблена система не лише довела свою ефективність у вирішенні поставлених завдань, а й стала прикладом того, як сучасні технології можуть допомогти у відновленні аграрного сектору України після війни. Система має значний потенціал для розвитку: її можна інтегрувати з супутниковими сервісами, такими як Sentinel-2, для аналізу даних у реальному часі, підключити до геоінформаційних систем (GIS) для створення детальних аграрних карт або синхронізувати з державними реєстрами для ширшої аналітики. Наприклад, додавання модуля для порівняння знімків у динаміці могло б допомогти відстежувати зміни стану земель, а інтеграція з базами даних про мінну небезпеку підвищила б безпеку планування рекультиваційних робіт.

Робота стала важливим кроком у цифровізації сільського господарства України, показавши, як технології штучного інтелекту можуть підтримати фермерів і екологів у складні часи [17], [18]. Розроблена система є практичним інструмент, який допомагає повернути життя на постраждалі землі, надаючи чіткі рекомендації для їхнього відновлення, закладає фундамент для створення нових рішень, які могли б охопити ширші аспекти аграрного сектору, та відкриває нові перспективи для подальших досліджень і розробок у сфері післявоєнної відбудови.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методи комп'ютерного зору і глибинних нейронних мереж для еколого-економічного аналізу : монографія / Н. М. Куссуль та ін. Київ : Наук. думка, 2024. 474 с.

2. Екологічний моніторинг ландшафтних ділянок з використанням регуляризованих штучних нейронних мереж. / С. Удовенко та ін. *Біоніка інтелекту*. 2022. *Т. 1 № 94*. С. 13–22. URL: https://doi.org/10.30837/bi.2020.1(94).03 (дата звернення: 20.05.2025).

3. Drozd S., Kussul N., Shelestov A. Satellite-Based Analysis of Forest Damage in Ukraine's Protected Areas. *13th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*. Gliwice, Poland. 4–6 September, 2025.

4. Mennour R., Blikaze S., Benaissa S. A Novel Decentralized Federated Incremental Learning Framework for ECG and EEG Signal Analysis. *International Journal of Computing*. 2025. Vol. 19, no. 3. P. 1–10.

5. EarthExplorer. *EarthExplorer*. URL: https://earthexplorer.usgs.gov/ (дата звернення: 30.05.2025).

6. JSON API | Meteostat Developers. *Meteostat Developers*. URL: https://dev.meteostat.net/api/ (дата звернення: 30.05.2025).

7. Геніцой П. О., Куссуль Н. М. Аналіз ефективності методів машинного навчання для задач розпізнавання. *XIX Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики»*, м. Київ. 2022. С. 290–295. URL: https://ela.kpi.ua/handle/123456789/52547 (дата звернення: 20.05.2025).

8 Застосування генеративно-змагальних мереж для покращення якості сегментації супутникових знімків / О. В. Шкаліков та ін. *XIX Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та*

інформатики», м. Київ. 2022. С. 375–378. URL: https://ela.kpi.ua/handle/123456789/52532 (дата звернення: 20.05.2025).

9. TensorFlow.js Підручник. Уроки для початківців. W3Schools українською. *W3Schools українською. Безплатні уроки онлайн для початківців, школярів та студентів*. URL: https://w3schoolsua.github.io/ai/ai_tensorflow_intro.html (дата звернення: 21.05.2025).

10. PyTorch Forecasting Documentation – pytorch-forecasting documentation. *PyTorch Forecasting Documentation – pytorch-forecasting documentation*. URL: https://pytorch-forecasting.readthedocs.io/en/stable/ (дата звернення: 21.05.2025)..

11. Aramendia A. I. The U-Net : A Complete Guide. *Medium*. URL: https://medium.com/@alejandro.itoaramendia/decoding-the-u-net-a-complete-guide-810b1c6d56d8 (дата звернення: 26.05.2025).

12. Collect Earth Online Home. *Collect Earth Online - Satellite Image Viewing & Interpretation Systemâ□□*. URL: https://www.collect.earth/ (дата звернення: 25.05.2025).

13. Search · Russia · WarSpotting. *Losses · Russia · WarSpotting – documented material losses in Russo-Ukrainian war*. URL: https://ukr.warspotting.net/search/?model=16&belligerent=2 (дата звернення: 25.05.2025).

14. Armed Forces of Ukraine destroyed the Russian Grad multiple rocket launcher with a drone in the Donetsk region. *Militarnyi*. URL: https://militarnyi.com/en/news/armed-forces-of-ukraine-destroyed-the-russian-grad-multiple-rocket-launcher-with-a-drone-in-the-donetsk-region/ (дата звернення: 25.05.2025).

15. Institute for the Study of War. *Institute for the Study of War*. URL: https://www.understandingwar.org/backgrounder/ukraine-conflict-updates (дата звернення: 25.05.2025).

16. 17 Цілей сталого розвитку | Global Compact. *Global Compact*. URL: https://globalcompact.org.ua/tsili-stijkogo-rozvytku/ (дата звернення: 01.06.2025).

17. Сільванович К. В., Гриньова О. Є. Моніторинг та відновлення сільськогосподарських земель засобами штучного інтелекту. *Радіоелектроніка та молодь у XXI столітті*, м. Харків. 2025. С. 51–53. URL: https://openarchive.nure.ua/entities/publication/6a9d7017-8bdb-4118-a8c8-8ed170ce91a8 (дата звернення: 01.06.2025).

18. Silvanovych K., Hrynova O. Leveraging ai for agricultural land monitoring and reclamation. *Information Systems and Technology: Results and Prospects*, Kyiv. 2025. P. 295–297. URL: https://ist.fit.knu.ua/_files/ugd/016074_36d0f427916c46abb6491a7572bb63ec.pdf (дата звернення: 01.06.2025).