МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія оптимізації вхідного математичного опису бортової системи розпізнавання безпілотного літального апарату при зміні висоти польоту»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Олексієнко Г.А.**

Студент гр.ІН.мз-01с **Литвиненко А.М.**

Суми 2021

Сумський державний університет

Факультет ЦЗДВН Кафедра Комп'ютерних наук

Спеціальність 122-Комп'ютерні науки

Затверджую:
зав.кафедрою
Довбиш А.С.
_____
«___» _____2021р.

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТОВІ

Литвиненку Андрію Миколайовичу

1. **Тема роботи:** «Інформаційна технологія оптимізації вхідного математичного опису бортової системи розпізнавання безпілотного літального апарату при зміні висоти польоту»

Затверджую наказом по інституту від «__» _____ 2021 р. № _____

2. Термін здачі студентом закінченої роботи _____

3. Вхідні данні до роботи _____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Інформаційний огляд предметної області дослідження, постановка задачі; 2) Опис методів дослідження; 3) Програмна реалізація інтелектуальної системи розпізнавання наземних об'єктів; Висновки; Списки використаних джерел; Додатки*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Аналіз предметної області. Постановка задачі дослідження		
2.	Моделювання тривимірної моделі місцевості		
3.	Програмна реалізація інтелектуальної схеми		
4.	Оформлення пояснювальної записки до дипломної роботи		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 48 стор., 19 рис., 3 таблиці, 1 додаток., 20 джерел.

Об’єкт дослідження – слабоформалізований процес розпізнавання об’єктів на місцевості під час зміни висоти безпілотного літального апарату.

Мета роботи – створення автоматизованої бортової системи безпілотного авіаційного комплексу для розпізнавання об’єктів інтересу в змодельованому середовищі.

Методи дослідження — детерміновано-статистичні методи розпізнавання образів, математичний аналіз вхідних даних, інформаційно-екстремальна технологія, 3Д-моделювання динамічного середовища та процесів.

Результати – розроблено та програмно реалізовано бортову систему розпізнавання наземних об’єктів, яка використовує інформаційно-екстремальне машинне навчання. У рамках обраної технології було здійснено паралельно-послідовну оптимізація системи контрольних допусків на ознаки розпізнавання. Для формування вхідного математичного опису та перевірки працездатності бортової системи розпізнавання було змодельовано 3Д-макет місцевості. Це дозволило перевірити результати класифікації об’єктів під час динамічного випробування безпілотника. Було висунуто і експериментально підтверджено припущення про недоцільність використання вхідного математичного опису, який був зроблений на різній висоті польоту безпілотника. Реалізовано алгоритм зміни розміру класів розпізнавання, при зміні висоти польоту безпілотного літального апарату.

ІНТЕЛЕКТУАЛЬНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, СИСТЕМА РОЗПІЗНАВАНЯ, МОДЕЛЮВАННЯ МІСЦЕВОСТІ, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ГЕОІНФОРМАЦІЙНА СИСТЕМА

ЗМІСТ

ВСТУП

З розвитком воєнної стратегії виникла потреба у створенні нових методів ведення бойових дій. Так, наприклад, широкого використання отримали багатофункціональні безпілотні літальні апарати (БПЛА), які є на озброєнні більшості країн, що входять в склад НАТО.

Сучасні безпілотники здатні виконувати широкий спектр оперативно-тактичних задач [1], наприклад: конвоювання транспортного засобу, корегування артилерійського вогню, геоінформаційний аналіз території тощо. Окремо слід виділити ударні БПЛА [2], які використовуються як бойові одиниці.

Для функціонування безпілотників необхідно встановлювати зв’язок з командним пунктом [3, 4], де відбувається керування та на який, по криптозахищеному каналу, передається зображення місцевості. Інакше кажучи БПЛА виступає у ролі ретранслятору відеопотоку. Це робить безпілотник вкрай вразливим до радіоелектронної протидії (РЕП). Одним із перспективних, але складних, способів подолання цієї проблеми є створення автономної бортової системи розпізнавання, за допомогою методів машинного навчання.

Основною метою кваліфікаційної роботи є розроблення бортової системи розпізнавання (БСР) різноманітних наземних об’єктів, яка б адаптувала вхідний математичний опис до висоти польоту безпілотного літального апарату. Для виконання цього завдання важливо вирішити дві головні задачі:

- отримати адекватний вхідний математичний опис місцевості, котрий повністю має відповідати реальним умовам;
- розробити бортову систему розпізнавання, ефективність котрої можна буде експериментально перевірити під час безпосередньо ідентифікації наземних об’єктів;

1 АНАЛІЗ ПРОБЛЕМ ІНФОРМАЦІЙНОГО СИНТЕЗУ ІНТЕЛЕКТУАЛЬНИХ ГЕОІНЙОРМАЦІЙНИХ СИСТЕМ

1.1 Огляд сучасних систем аналізу геоінформаційних даних

Сучасні геоінформаційні системи (ГІС) здатні зберігати, аналізувати і графічно подавати отриманий результат. В деякому сенсі ГІС можна представити як інструмент, що використовується для отримання комплексної інформації про певну місцевість та об’єкти на ній.

Принцип роботи геоінформаційних систем [5] полягає у тому, що кожному об’єкту місцевості ставиться у відповідність певний запис у базі даних, який містить атрибутну інформацію. Візуально усі дані зберігаються у вигляді набору тематичних шарів, рисунок 1.1, які об’єднанні на основі їх географічного положення. Це хоч і простий, але надзвичайно ефективний спосіб маніпулювання з просторовими даними.

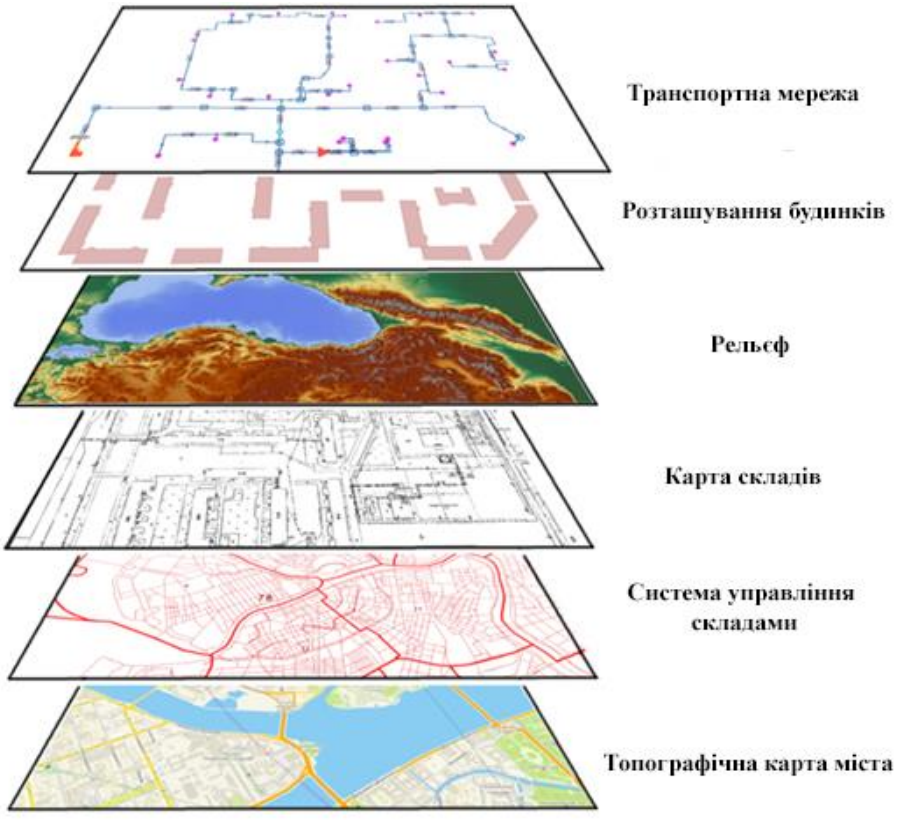


Рисунок 1.1 – Приклад багатошаровості геоінформаційної системи

Інформація, яка використовується в геоінформаційних системах поділяється на:

- позиційні: місцезнаходження об’єкту на місцевості, його фізичні координат.
- непозиційні: будь-яка інформація, яка описує об’єкт, наприклад: графічні дані, 3д-моделі, електронні документи тощо.

Сучасні ГІС здатні приймати найрізноманітніший вигляд і виконувати широкий спектр завдань. Так, наприклад, вони часто використовуються в наступних галузях:

- сільське господарство: виконання контролю за земельними масивами, аналіз майбутньої врожайності;
- екологія та природокористування: підтримання стану довкілля та контроль за ним. Забезпечення комфортних природних умов;
- просторова навігація: є базовою сферою використання ГІС, адже дозволяє комплексно орієнтуватися у просторі (рис. 1.2).
- будівництво: будь-які будівельні роботи потребують комплексного аналізу території. Ґрунти, ландшафт, щільність існуючих транспортних потоків, наявність інфраструктури тощо. Ці фактори впливають на будівельний проект.
- медико-екологічна: сфера використання, яка із-за пандемії є досить актуальною. ГІС дозволяє збирати інформацію про захворювання та темпи його поширення [6], (рис.1.3).
- управління бізнесом: більшість підприємств мають складну та розгалужену інфраструктуру, знання якої необхідне для оптимізації роботи бізнесу.
- воєнна: ГІС використовується для швидкого та комплексного аналізу території, передбачення дій супротивника та займання найвигіднішої позиції. Дуже часто програмні засоби мають можливість створення 3д-макетів місцевості та моделювання бойових дій [7] (рис. 1.4).

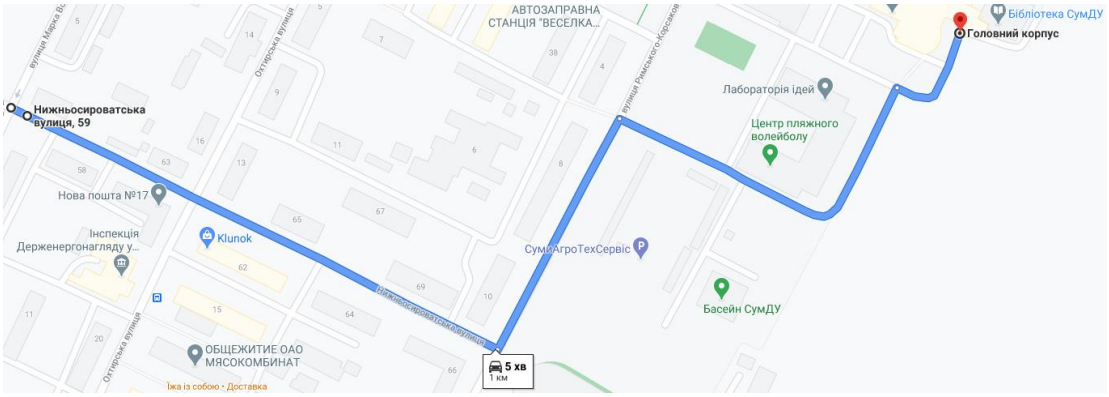


Рисунок 1.2 – Приклад GPS-карти з використанням векторних об’єктів

Рисунок 1.3 – Приклад ГІС, яка демонструє поширення вірусу COVID-19

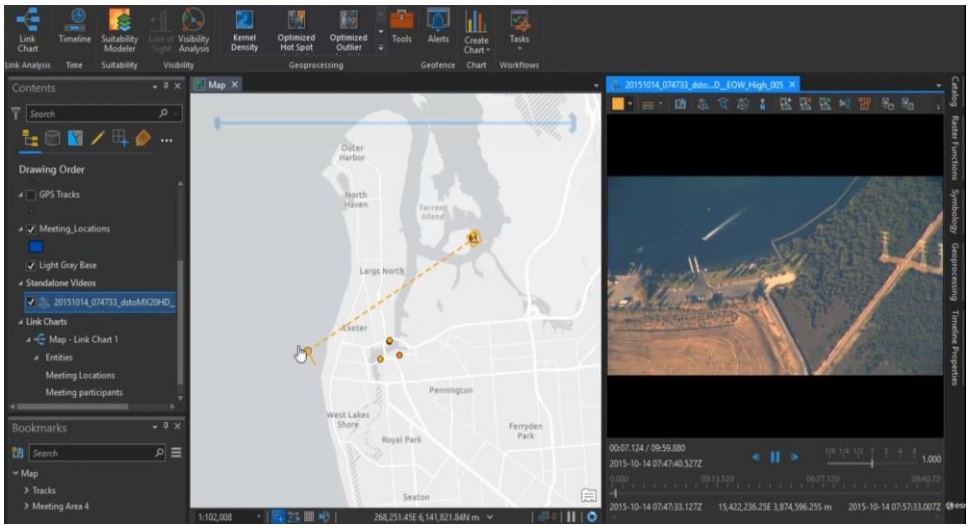


Рисунок 1.4 – Приклад роботи воєнної ГІС у реальному часі

Більшість проблем, які ставлять перед геоінформаційними системами, є шаблонними і їх можна вирішити за допомогою готових програм-конструкторів, таких як QGIS [8]. Але коли річ стосується специфічних вимог чи не типових завдань, виникає потреба у створенні вузькопрофільних рішень. Наприклад: інтелектуальний аналіз території; прогнозування; моделювання можливих результатів дій; відтворення динамічних процесів.

1.2 Огляд методів інтелектуального аналізу даних

В процесі розпізнавання наземних об’єктів бортові системи використовують інтелектуальні методи, які базуються на статистичній теорії прийняття рішень. Для узагальнення усіх алгоритмі інтелектуального аналізу даних, у сучасній літературі, використовують термін: «автоматична класифікація».

Мозок людини можна розглядати як біологічну інтелектуальну систему. Так, наприклад, згідно П.К. Анохіну [9] на фізіологічному рівні розпізнавання образів, для будь-якої живої істоти, можна поділити на два етапи:

- навчання – процес виокремлення та запам’ятовування унікальних ознак об’єкту;
- екзамен – процес розпізнавання об’єкту на основі запам’ятованих, на етапі навчання, ознак;

Згідно цього твердження можна сформувати загальну постановку задачі для функціонування системи розпізнавання (СР). Припустимо, що під час етапу навчання потрібно знайти оптимальне, з інформаційної точки зору, розбиття простору ознак розпізнавання (ОР) на скінченну кількість класів. На стадії екзамену система повинна, за обмежене число випробувань, прийняти класифікаційне рішення про належність вектору-реалізації образу, що розпізнається, до класу із множини, що апріорно визначена $\{X_m^*\}$.

Класифікація множини об’єктів можлива із-за існування гіпотези компактності [10]. Річ у тім, що схожі ознаки створюють «компактні» згущення в розглядаємому просторі, рисунок 1.5. Перш за все це означає, що межу цих класів можна розрахувати за деякою формулою, яку в подальшому можливо використати під час ідентифікації інших об’єктів.

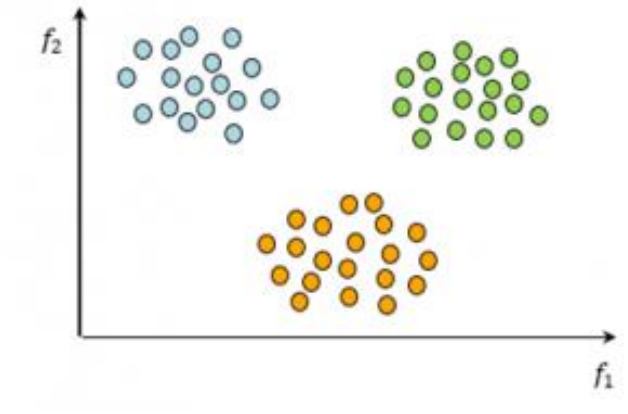


Рисунок 1.5 – Приклад виконання гіпотези компактності

Переважна більшість сучасних інтелектуальних алгоритмів базуються на статистичних методах, які, в свою чергу, побудовані на основі загально відомого байесівського вирішального правила, яке можна сформулювати наступним чином: у процесі машинного навчання виконується порівняння обчислених апостеріорних ймовірностей $p(\mu_m / \gamma_l)$, де μ_m – подія, що відображає дійсну належність реалізації класу X_m^o; γ_l – гіпотеза про належність реалізації класу X_m^o, $m,l = \overline{1,M}$.

Під час роботи статистичних методів виконується набір репрезентативної вибірки і створення на її основі апріорних ймовірностей $p(\mu_m / \gamma_l)$, які можливо перевести в апостеріорні за формулою Байеса:

$$p(\mu_m / \gamma_l) = \frac{p(\mu_m)\, p(\gamma_l / \mu_m)}{p(\gamma_l)}$$

де $p(\gamma_l)$ − це повна ймовірність прийняття гіпотези, яка визначається за наступною теоремою: $p(\gamma_l) = \sum_{m=1}^{M} p(\mu_m)\, p(\gamma_l / \mu_m)$.

На практиці використання класифікатору Байеса [11], а отже і статистичних методів, ускладняються відсутністю заздалегідь відомої функції щільності. Э два підходи, які дозволяють подолати цей недолік:

- параметричний: подання функції щільності у аналітичній формі;
- не параметричний: представлення функції щільності через оцінку її основних величин. Це можна зробити за допомогою деякої вибіркової інформації, яка точно належить генеральній сукупності, та відомих методів аналізу даних;

Процес навчання, для статистичних методів, характеризується високою трудомісткістю. Це можна пояснити за допомогою умови збіжності чисельних методів. Доведено, що оперативність роботи відомих алгоритмів стохастичної апроксимації, до яких можна віднести градієнтні методи, коливається у значних межах [12].

Результатом роботи статистичних методів, у рамках інтелектуального аналізу даних, є знаходження деякої роздільної функції. Її головним призначенням є розбиття простору ознак на ділянки, які представляють із себе класи розпізнавання. Іншими словами, задача навчання зводиться до пошуку екстремального значення коефіцієнту ефективності, який представляє із себе мінімальний усереднений ризик для помилкової класифікації.

Отже, статистичні методи характеризуються низькою достовірністю класифікації на етапі екзамену, бо на практиці заздалегідь відсутня апріорна інформація про об’єкт, або її не достатньо. Це означає, що найкращого результату можливо досягти лише в галузях, де є можливість отримати набір даних для репрезентативної вибірки. Хоча навіть так потрібно бути впевненим, що обрані елементи є однорідними та стійкими.

Для вирішення задач адаптивного пошуку та багатопараметричної оптимізації все частіше використовуються генетичні алгоритми (ГА) [13]. Вони беруть за основу принципи біологічної спадковості та природного відбору, що дозволяє ГА виконувати випадковий пошук глобального екстремуму [14]. На практиці досить часто генетичні алгоритми об’єднують з іншими методами, наприклад, для попереднього аналізу вхідних даних. Це дозволяє побудувати більш оптимальний класифікатор.

У загальному випадку генетичний алгоритм можна розбити на наступні етапи:

1) Формування початкової популяції;
2) Схрещування (обмін інформації) та/або мутація (зміна інформації);
3) Відбір найбільш придатних даних за допомогою селекції;
4) Формування нового покоління;
5) Якщо виконується критерій зупинки, то завершуємо алгоритм, інакше переходимо до пункту (2);

Для того, щоб задачу можна було вирішити за допомогою генетичного алгоритму потрібно подати ознаки у вигляді масиву, який буде виступати в ролі аналогу біологічної хромосоми. Сукупність початкових елементів створюється випадковим чином, у рамках ГА її називають «популяцією». Далі, за допомогою деякої функції належності, проводиться оцінка кожного масиву. Ті, що отримали найбільше значення критерію допускаються до етапу схрещування, де за допомогою генетичних операторів відбувається перехід до нового покоління. Таким чином виконується «еволюція» популяції, яка продовжується до виконання критерію зупинки.

Більшість сучасних науковців віддають перевагу нейроподібним системам [15]. Це окрема група методів, які намагаються програмно відтворити складну структуру людського мозку. Таким чином нейрони реалізуються у вигляді перемикачів, які поєднуються між собою зв’язками з різною вагою, щоб утворити нерозривну систему. Навчання проходить за допомогою повторної активації деяких з’єднань. Це дозволяє збільшити

вірогідність отримання коректного результату при умові, що на вхід буде подана подібна інформація. Проте цей підхід є чутливим до багатовимірності алфавіту класів розпізнавання. Системі вкрай складно ідентифікувати великі масиви даних, які зроблені при довільних умовах. Це призводить до зниження оперативності роботи, що вкрай небажано для цілого пласту задач.

Окрім розглянутих підходів створення СР, є ще досить специфічні. До них можна віднести нечітку класифікацію, яка базується на теорії нечітких множин Л.Заде [16]. Хоч методи цієї груп давно використовуються в інтелектуальних системах, але для задач розпізнавання образів вони потребують доопрацювання.

Отже, не зважаючи на досить стрімкий розвиток галузі автоматичної класифікації, все ще відсутні універсальні алгоритми розпізнавання образів для слабоформалізованих задач. На сьогоднішній день слід виділити наступні науково-методологічні проблеми інтелектуальних систем (ІС):

- модальний характер більшості досліджень, що, звісно, становить, певну цінність, але є не придатним для практичного застосування;
- незавершеність теорії ефективного машинного навчання інтелектуальних систем [17];

1.3 Формалізована постановка задачі інформаційного синтезу бортової системи розпізнавання

Нехай э апріорно визначений алфавіт класів розпізнавання $\{X_m^o \mid m=\overline{1,M}\}$, де кожен елемент характеризує кадр зображення місцевості. Для кожного побудовано тривимірну навчальну матрицю $\| y_{m,i}^{(j)} \|$ яскравості, в якій рядок $\{y_{m,i}^{(j)} \mid i=\overline{1,N}\}$, де N – кількість ознак розпізнавання, є структурованим вектором ознак, а стовпчик матриці – випадкова навчальна вибірка $\{y_{m,i}^{(j)} \mid j=\overline{1,n}\}$ i-ї ознаки з обсягом n.

Відомо, що при використанні ІЕІ-технології відбувається перетворення вхідної навчальної матриці *Y* в робочу бінарну *X*, котра перетворюється в

процесі машинного навчання. Саме тому для простору Хеммінга задано вектор параметрів функціонування, котрі впливають на ефективність машинного навчання БСР розпізнавати реалізації класу X_m^o:

$$g_m =< x_m, d_m, \delta >, \quad (1.1)$$

де x_m – усереднений вектор реалізацій, вершина якого визначає центр гіперсферичного контейнера класу розпізнавання X_m^o; d_m – радіус контейнеру розпізнавання X_m^o, який відновлюються в радіальному базисі простору ознак; δ – параметр, величина якого рівна половині симетричного поля контрольних допусків на ознаки розпізнавання, якими є значення яскравості в пікселях.

На параметри функціонування системи, які будуть в подальшому називатися параметрами машинного навчання, накладаються відповідні обмеження:

- область значень яскравості пікселів знаходиться в інтервалі [0;255] градацій яскравості;
- область значень радіуса контейнера класу X_m^o задається нерівністю:

$$d_m < d(x_m \oplus x_c),$$

де $d(x_m \oplus x_c)$ - міжцентрова відстань між реалізацією x_m і найближчою до неї x_c сусіднього класу X_c^o;

- область значень параметра δ задається нерівністю:

$$\delta < \delta_H / 2,$$

де δ_H - нормоване поле допусків на ознаки розпізнавання;

При функціонуванні розробленої СР в режимі екзамену необхідно виконати перевірку функціональної ефективності машинного навчання. Тобто потрібно оптимізувати відповідні параметри (1.1), котрі мають забезпечувати максимально можливе значення інформаційного критерію оптимізації в робочій (допустимій) області визначення його функції:

$$\overline{E}^{*} = \frac{1}{M}\sum_{m=1}^{M}\max_{G_E \cap \{k\}} E_m^{(k)}, \qquad (1.2)$$

де $E_m^{(k)}$ – значення інформаційного критерію, обчислене на k-му кроці машинного навчання; G_E – робоча область обчислення інформаційного критерію; $\{k\}$ –множина кроків машинного навчання.

Для розв'язку поставленої задачі потрібно виконати наступні завдання:

- реалізувати алгоритми машинного навчання СР образів з оптимізацією системи контрольних допусків на ознаки розпізнавання;
- побудувати вирішальні правила в просторі ознак класів розпізнавання;
- побудувати 3д-модель місцевості для формування вхідного математичного опису СР;
- реалізувати алгоритм функціонування системи в режимі екзамену і надати оцінку його функціональній ефективності;

Таким чином, завдання інформаційного синтезу здатної навчатися системи розпізнавання полягає в оптимізації параметрів її машинного навчання методом наближення глобального максимуму інформаційного критерію (1.2) до його максимального граничного значення.

Для перевірки ефективності розробленого алгоритму буде використана 3д-модель місцевості. У такий спосіб вдасться поставити експеримент функціонування БСР у умовах, які максимально наближені до реальних.

2 ОПИС МЕТОДУ ДОСЛІДЖЕННЯ

2.1 Основні положення інформаційно екстремальної технології аналізу даних

Згідно праць [17-18], основна ідея машинного навчання, при використанні IEI-технології, полягає в побудові, у рамках геометричного підходу, високо достовірних класифікаційних правил шляхом оптимізації відповідних параметрів машинного навчання системи. При цьому здійснюється цілеспрямований пошук глобального максимуму функції статистичного інформаційного критерію в робочій області її визначення, під час процесу відновлення, в радіальному базисі, бінарного простору ознак контейнерів класів розпізнавання. Перетворення вхідної навчальної матриці здійснюється через оптимізацію контрольних допусків на ознаки розпізнавання. У результаті отримуємо робочу бінарну, яка, в процесі машинного навчання, змінюється шляхом її адаптації до максимальної достовірності класифікаційних рішень

Процес побудови вирішальних правил, у рамках IEI-технології, здійснюється за допомогою багатоциклічної процедури, де відбувається пошук максимально можливого, усередненого за алфавітом $\{X_m^o\}$, граничного значення інформаційного критерію оптимізації параметрів машинного навчання (2.1):

$$g_\xi^* = \arg \max_{G_\xi} \{ \max_{G_{\xi-1}} \{ ... \{ \max_{G_1 \cap G_E} \frac{1}{M} \sum_{m=1}^{M} E_m \} ... \} \}, \quad (2.1)$$

де E_m – інформаційний критерій оптимізації системи розпізнавати реалізацій класу X_m^o; G_ξ – деяка допустима область значень ξ-ї ознаки розпізнавання; G_E – деяка допустима область визначення функції інформаційного критерію.

На алгоритм навчання (2.1) накладаються деякі обмеження:

$$\left(\forall X_m^o \in \tilde{\Re}^{|M|} \right) \left\lfloor X_m^o \neq \varnothing \right\rfloor, \quad (2.2)$$

де $\tilde{\Re}^{|M|}$ – розбиття простору ознак на класи з потужністю $Card\,\tilde{\Re} = M$;

$$\left(\exists X_k^o \in \tilde{\Re}^{|M|}\right)\left(\exists X_l^o \in \tilde{\Re}^{|M|}\right)\left[X_k^o \neq X_l^o \to X_k^o \bigcap X_l^o \neq \varnothing\right]; \quad (2.3)$$

$$\left(\forall X_k^o \in \tilde{\Re}^{|M|}\right)\left(\forall X_l^o \in \tilde{\Re}^{|M|}\right)\left[X_k^o \neq X_l^o \to Ker X_k^o \bigcap Ker X_l^o = \varnothing\right], \quad (2.4)$$

де $Ker\, X_k^o$ – ядро класу розпізнавання X_k^o; $Ker\, X_l^o$ – ядро класу розпізнавання; X_l^o – ядро найближчого сусіда для класу X_k^o;

$$\left(\forall X_k^o \in \tilde{\Re}^{|M|}\right)\left(\forall X_l^o \in \tilde{\Re}^{|M|}\right)[X_k^o \neq X_l^o \to (d_k^* < d(x_k \oplus x_l)) \,\&$$
$$\&\,(d_l^* < d(x_k \oplus x_l))], \quad (2.5)$$

де d_k^* – оптимальний радіус контейнера класу розпізнавання X_k^o; d_l^* – оптимальний радіус для контейнера X_l^o; $d(x_k \oplus x_l)$ – кодова відстань між вектором x_k, усередненим за реалізаціями класу розпізнавання X_k^o, та аналогічним вектором x_l для X_l^o;

$$\bigcup_{X_m^o \in \tilde{\Re}} X_m^o \subseteq \Omega_Б;\ k \neq l;\ k,l,m = \overline{1,M}\ , \quad (2.6)$$

де $\Omega_Б$ – бінарний простір ознак розпізнавання.

Глибина машинного навчання залежить від розмірності вектору параметрів оптимізації. Які адаптуються до прийняття класифікаційних рішень у межах, так званого, базового алгоритму, який виконує наступні функції:

1) Обчислення інформаційного критерію та пошук його максимального значення;

2) визначення оптимальних координат векторів $\{x_m^* \mid m = 1, M\}$ і радіусів контейнерів класів розпізнавання $\{d_m^* \mid m = 1, M\}$;

Таким чином, інформаційно-екстремальне машинне навчання полягає в наближенні на кожному кроці навчання інформаційного критерію (1.2) до його максимально можливого граничного значення.

2.2 Математичні моделі машинного навчання

Розроблену систему розпізнавання можна розглянути у вигляді орієнтованого графа. В якому задіяні множини, котрі беруть участь у процесі машинного навчання. При цьому вони відображені одна на одну за допомогою відповідних операторів.

Вхідний математичний опис категорійної моделі можна подати наступним чином (2.7):

$$\Delta_B = < G, T, \Omega, Z, Y, X; f_1, f_2 >, \tag{2.7}$$

де G – певний простір вхідних сигналів (факторів); T – множина моментів часу одержання інформації; Ω – простір ознак розпізнавання; Z – простір станів системи, який визначає алфавіт класів розпізнавання; Y – вибіркова множина, котра утворює вхідну багатовимірну навчальну матрицю; X – бінарна навчальна матриця; $f_1 : G \times T \times \Omega \times Z \to Y$ – оператор формування вхідної навчальної матриці Y; $f_2 : Y \to X$ – оператор трансформації вхідної навчальної матриці Y в бінарну X.

Слід зазначити, що декартів добуток $G \times T \times \Omega \times Z$ утворює універсум випробувань, який є основою для формування вхідної навчальної матриці Y.

Загальний вигляд категорійної моделі БСР, яка використовую інформаційно-екстремальне машинне навчання з оптимізацією відповідних параметрів структурованого вектору (1.1) зображено на рисунку 2.1.

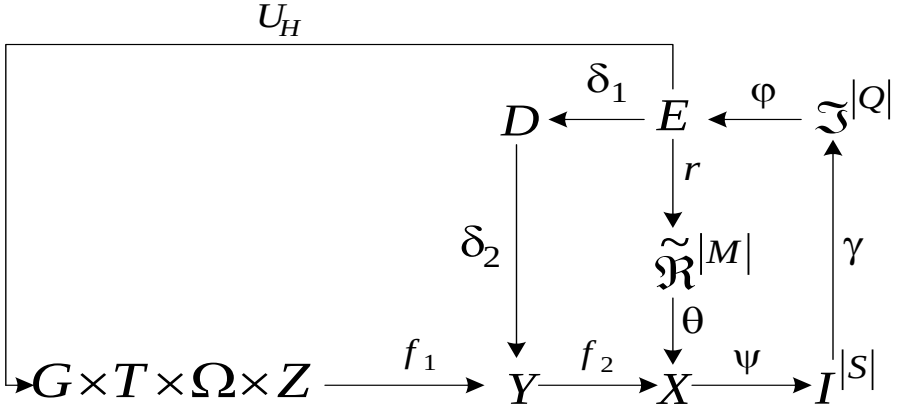


Рисунок 2.1 – Категорійна модель машинного навчання СР

На рисунку 2.1 терм-множина E складається зі значень інформаційного критерію, які обчислювалися на кожному кроці машинного навчання і є загальними для всіх контурів оптимізації. Оператор $r: E \to \tilde{\Re}^{|M|}$ будує на кожному кроці навчання розбиття $\tilde{\Re}^{|M|}$, яке відображається оператором θ на нечіткий розподіл двійкових реалізацій матриці X. Далі оператор $\psi: X \to I^{|S|}$, де $I^{|S|}$ – множина S гіпотез, виконує перевірку основної статистичної гіпотези $\gamma_1 : x_{m,i}^{(j)} \in X_m^o$. Оператор γ визначає множину $\Im^{|Q|}$ точнісних характеристик класифікаційних рішень, де $Q = S^2$, а оператор φ розраховує множину значень E інформаційного критерію оптимізації, який є функціоналом від точнісних характеристик. Контур оптимізації контрольних допусків на ознаки розпізнавання замикається через терм-множину D, частинами якої є значення системи контрольних допусків (СКД) на ознаки розпізнавання. Оператор u регламентує процес машинного навчання.

Для перевірки функціональної ефективності вирішальних правил, які були отримані у процесі інформаційно-екстремального машинного навчання, було реалізовано контрольний екзамен. У межах якого відбувається функціонування системи розпізнавання. Категорійну модель цього процесу, у вигляді орієнтованого графа, зображено на рисунку 2.2.

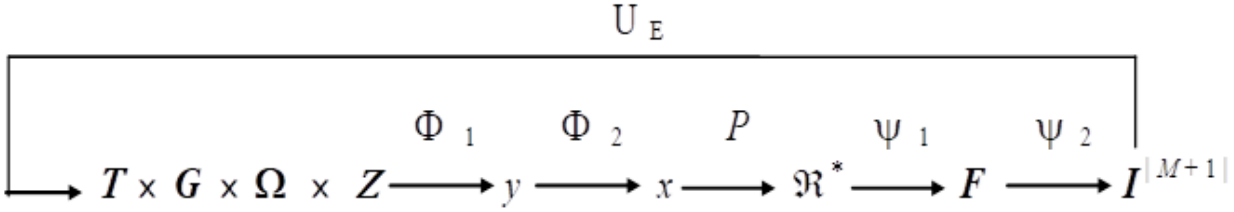


Рисунок 2.2 – Категорійна модель СР в режимі екзамену

У цій моделі (рис. 2.2) оператор Φ_1, із джерела інформації, формує екзаменаційну реалізацію класу, що розпізнається, яка є аналогічною, за своєю будовою, навчальній матриці. Оператор Φ_2, за допомогою отриманих

на етапі машинного навчання оптипальних параметрів системи контрольних допусків на ознаки розпізнавання, формується двійкова реалізація x, а оператор P відображає вектор-реалізацію, що розпізнається, на побудовану, під час етапу машинного навчання, оптимальне розбиття $\Re^*$ класів розпізнавання. Оператор Ψ_1 для кожної реалізації розраховує значення оптимальних вирішальних правил і будує терм-множину F, а оператор Ψ_2 за максимальним значенням відносить вектор, який розпізнається, до одного із класів із заданого алфавіту $\{X_m^o\}$. Призначенням оператора U_E є регламентація процесу екзамену.

Розглянуті категорійні моделі відображають властиві людям перетворення інформації, які виконуються при когнітивних процесах, зокрема при прийнятті класифікаційних рішень. Тому вони розглядаються як узагальнені структурні схеми алгоритмів інформаційно-екстремального машинного навчання системи розпізнавання образів.

2.3 Оцінка функціональної ефективності машинного навчання системи розпізнавання

У межах інформаційно-екстремального машинного навчання, оцінка функціональної ефективності системи розпізнавання, використовують ентропійний критерій Шеннона та інформаційну міру Кульбака. Так у роботі [17] критерій Кульбака описується як результат множення логарифмічного відношення обчислених на k-му кроці навчання повної ймовірності $P_{t,m}^{(k)}$ вірного прийняття рішень про належність класу X_m^o реалізації, що розпізнається, до повної ймовірності $P_{f,m}^{(k)}$ хибного прийняття рішень і різниці цих вірогідностей:

$$E_m^{(k)} = \log_2 \frac{P_{t,m}^{(k)}}{P_{f,m}^{(k)}} * [P_{t,m}^{(k)} - P_{f,m}^{(k)}]. \qquad (2.8)$$

Відповідно до теореми про повну ймовірність для двох-альтернативної системи оцінок прийнятих рішень при допущенні, яке базується принципах Лапласа-Бернуллі, що $p(\mu_m) = p(\mu_c) = 0{,}5$, маємо:

$$P_{t,m}^{(k)} = 0{,}5D_{1,m}^{(k)}(d) + 0{,}5D_{2,m}^{(k)}(d); P_{f,m}^{(k)} = 0{,}5\alpha_m^{(k)}(d) + 0{,}5\beta_m^{(k)}(d), \quad (2.9)$$

де $D_{1,m}^{(k)}(d)$ – перша достовірність прийняття рішення на k-му кроці навчання; $D_{2,m}^{(k)}(d)$– друга достовірність; $\alpha_m^{(k)}(d)$ – помилка першого роду; $\beta_m^{(k)}(d)$ – помилка другого роду; d – дистанційна міра, котра визначає радіуси гіперсферичних контейнерів, збудованих в радіальному базисі Хеммінга.

В результаті підстановки рівнянь (2.9) в формулу (2.8) одержимо

$$E_m^{(k)} = 0{,}5\log_2\left(\frac{D_{1,m}^{(k)}(d) + D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)}\right) * \{[D_{1,m}^{(k)}(d) + D_{2,m}^{(k)}(d)] - [\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)]\}. \quad (2.10)$$

Виразимо першу та другу достовірності:

$$D_{1,m}^{(k)}(d) = 1 - \alpha_m^{(k)}(d); D_{2,m}^{(k)}(d) = 1 - \beta_m^{(k)}(d). \quad (2.11)$$

В результаті підстановки рівнянь (2.11) в формулу (2.10) одержимо:

$$E_m^{(k)} = \log_2\left(\frac{2 - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d))}{\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)}\right) * \left[1 - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d))\right]. \quad (2.12)$$

Нормована модифікація критерію (2.12) має наступний вигляд (2.13)

$$E_m^{(k)} = \frac{E_{K\,m}^{(k)}}{E_{K\max}^{(k)}}, \quad (2.13)$$

де $E_{\max}^{(k)}$ – значення інформаційного критерію, яке приймається при підстановці $D_{1,m}^{(k)}(d) = D_{2,m}^{(k)}(d) = 1$ і $\alpha_m^{(k)}(d) = \beta_m^{(k)}(d) = 0$ у формулу (2.12).

Оскільки, інформаційний критерій (2.12) є функціоналом від точнісних характеристик, то при репрезентативному, але обмеженому обсязі навчальної вибірки, користуються їх оцінками:

$$\alpha_m^{(k)}(d) = \frac{K_{1,m}^{(k)}}{n_{\min}}; \quad \beta_m^{(k)}(d) = \frac{K_{2m}^{(k)}}{n_{\min}}, \tag{2.14}$$

де $K_{1,m}^{(k)}$ – число подій, під час яких реалізації, що належать класу X_m^o, помилково до нього не відносяться; $K_{3,m}^{(k)}$ – число подій, при яких хибно відносяться, до класу розпізнавання X_m^o, реалізації сусіда X_c^o; $n_{\min}$ – мінімальний обсяг навчальної вибірки, котрий розраховується відповідно до праці [17].

В результаті підстановки відповідних оцінок точнісних характеристик (2.14) у рівняння (2.12) отримаємо працюючу формулу для розрахунку критерію Кульбака:

$$E_m^{(k)} = \frac{1}{n}\log_2\left\{\frac{2n + 10^{-r} - \left[K_1^{(k)} + K_2^{(k)}\right]}{\left[K_1^{(k)} + K_2^{(k)}\right] + 10^{-r}}\right\}\left[n - (K_{12}^{(k)} + K_3^{(k)})\right], \tag{2.15}$$

де 10^{-r} –достатньо мале значення, що додається для уникнення поділу на нуль.

На практиці у формулі (2.15) значення параметра r обирається із інтервалу $1 < r \le 3$, тобто є рівним кількості знаків мантиси критерію $E_m^{(k)}$.

Розрахунок коефіцієнтів $K_{1,m}^{(k)}$ і $K_{2,m}^{(k)}$, під час класифікації j-ої реалізації, здійснювався за процедурами:

$$if\ x_m^{(j)} \notin X_m^o\ then\ K_1(j) := K_1(j-1) + 1;\ if x_c^{(j)} \in X_m^o\ then\ K_3(j) := K_3(j-1) + 1$$

При цьому віднесення, наприклад, реалізації $x_c^{(j)}$ до класу розпізнавання X_m^o виконується у такий спосіб:

1) розраховується кодова відстань між центрами класів розпізнавання

$$d[x_m \oplus x_c^{(j)}];$$

2) порівняння: якщо $d[x_m \oplus x_c^{(j)}] \le d_m$, то $x_c^{(j)} \in X_m^o$, інакше – $x_c^{(j)} \notin X_m^o$;

Таким чином, інформаційні критерії (2.12) і (2.13) є функціоналами від деяких точнісних характеристик класифікаційних рішень, котрі залежать від дистанційних параметрів класів розпізнавання. Тобто, наведені вище модифікації критерію Кульбака можливо розглядати як підсумовування відомих статистичних та дистанційних критеріїв близькості класів розпізнавання у просторі ознак.

2.4 Лінійний алгоритм машинного навчання інформаційно екстремальної технології для системи розпізнавання

У більшості випадках базовий алгоритм машинного навчання не забезпечує необхідної якості функціонування системи розпізнавання. Це пов’язано із тим, що контрольні допуски, в інформаційному розумінні, можуть бути цілком не оптимальними. Відповідно до категорійної моделі, рисунок 2.1, інформаційно-екстремальний алгоритм машинного навчання з оптимізацією контрольних допусків (2.16) подано у вигляді ітераційної процедури пошуку глобального максимуму усередненого, за алфавітом класів розпізнавання, нормованого інформаційного критерію в його робочій (допустимій) зоні визначення функції.

$$\delta_K^* = \arg\max_{G_\delta}\{\max_{G_E \cap \{k\}} \overline{E}^{(k)}\}, \qquad (2.16)$$

де δ_K^* – оптимальний параметр поля контрольних допусків; G_δ – допустима область значень параметра δ поля контрольних допусків; $\{k\}$ – впорядкована, за часом множина, кроків машинного навчання.

На рисунку 2.3 зображено двобічне симетричне поле контрольних допусків.

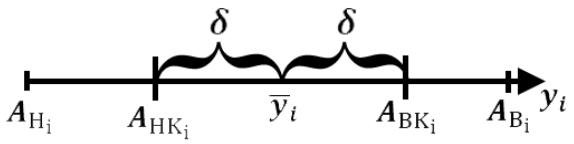


Рисунок 2.3 – Поле контрольних допусків на ознаку розпізнавання

Згідно рисунку 2.3 приймаємо такі позначення: $\overline{y_i}$ – номінальне (усереднене) значення ознаки y_i; $A_{H,i}, A_{B,i}$ – верхні та нижні нормовані допуски на ознаку y_i; $A_{HK,i}, A_{BK,i}$ – верхні та нижні контрольні допуски на ознаку y_i; δ – параметр, який рівний половині симетричного поля контрольних допусків.

Вхідною інформацією для алгоритму машинного навчання є масив $\{y_{m,i}^{(j)}\}$ і система яка складається з полів нормованих допусків $\{\delta_{\text{H,i}}\}$ на ознаки розпізнавання, котра задає ділянку значень СКД.

Розглянемо схему алгоритму машинного навчання з оптимізацією контрольних допусків, за процедурою (2.16):

1) ініціалізація лічильника класів розпізнавання: $m := 0$;

2) $m : m + 1$;

3) ініціалізація параметра δ поля контрольних допусків: $\delta := 0$;

4) $\delta := \delta + 1$;

5) обчислення нижніх $A_{H,i}$ та верхніх $A_{B,i}$ контрольних допусків на ознаки розпізнавання, відповідно правил:

$$A_{H,i} = y_i - \delta; \quad A_{B,i} = y_i + \delta;$$

6) ініціалізація лічильника кроків перетворення радіусів гіперсферичного контейнера: $k := 0$;

7) $k := k = 1$;

8) перетворення навчальної матриці в робочу бінарну X, частки якої розраховуються за правилом:

$$x_{m,i}^{(j)}[k] = \begin{cases} 1, \text{ якщо } A_{HK,i}[k] < y_{m,i}^{(j)} < A_{BK,i}[k]; \\ 0, \text{ якщо інакше.} \end{cases}$$

9) створення двійкових векторі-реалізацій $\{x_m\}$, частки яких розраховуються за правилом

$$x_{m,i} = \begin{cases} 1, & if \ \ \frac{1}{n}\sum_{j=1}^{n} x_{m,i}^{(j)} > \rho_m; \\ 0, & if \ \ \ else, \end{cases}$$

де ρ_m – рівень квантування координат двійкового вектору x_m, котрий за замовчуванням є рівним 0,5.

10) поділ множини векторів $\{x_m\}$ на пари найближчих “сусідів” $\Re_m^{|2|} =< x_m, x_c >$, де x_c – усереднений вектор сусіднього класу X_c^o;

11) розраховується інформаційний критерій;

12) за умови $k \le N$, то виконується пункт 7, в іншому разі – пункт 13;

13) за умови $\delta < \delta_H$, то виконується пункт 4, в іншому разі – пункт 14;

14) розраховується максимальне значення інформаційного критерію в робочій зоні визначення функції, де перша і друга достовірності більше 0,5;

15) за умови якщо виконується $m < M - 1$, то реалізується пункт 2, в іншому разі – пункт 16;

16) розраховується глобальний максимум деякого усередненого інформаційного критерію $\overline{E}^*$ в робочій зоні визначення його функції;

17) розраховуються оптимальні значення параметра δ^* і відповідних верхніх $A_{H,i}^*$ і нижніх $A_{B,i}^*$ контрольних допусків на всі ознаки розпізнавання;

Одержані в процесі паралельної оптимізації екстремальні значення параметрів машинного навчання є квазіоптимальними, так-як вони видозмінювалися, при кожному кроці навчання, на деяку однакову величину для всієї кількості ознак одночасно. Для збільшення функціональної ефективності СР було реалізовано алгоритм машинного навчання з послідовною оптимізацією контрольних допусків. При цьому, одержані на етапі паралельної оптимізації контрольні допуски приймалися як початкові під час послідовної оптимізації, яка виконувалася за процедурою (2.17).

$$\delta_{K,i}^* = \arg \bigotimes_{l=1}^{L} \left\{ \max_{G_{\delta i}} \left[\frac{1}{M} \sum_{m=1}^{M} \max_{G_{Em} \cap \{k\}} E_m^{(l)}(d_m) \right] \right\}, i = \overline{1, N}, \qquad (2.17)$$

де L – кількість прогонів процедури послідовної оптимізації контрольних допусків, обумовлених неоптимальними початковими величинами для всіх ознак; $\otimes$ – символ операції повторення;

Машинне навчання СР, з паралельно-послідовною оптимізацією контрольних допусків, дає можливість збільшити достовірність класифікаційних рішень і при цьому досить суттєво підвищується оперативність машинного навчання, так-як пошуки глобального максимуму критерію виконуються тільки в робочій зоні визначення функції.

За отриманими оптимальними геометричними параметрами контейнерів класів розпізнавання було побудовано продукційні вирішальні правила, які мають наступний вигляд:

$$(\forall X_m^o \in \Re^{|M|})(\forall x^{(j)} \in \Re^{|M|})[if\ \ (\mu_m > 0)\ \&\ (\mu_m = \max\{\mu_m\})$$
$$then\ \ x^{(j)} \in X_m^o\ \ \ else\ \ x^{(j)} \notin X_m^o], \qquad (2.18)$$

де $x^{(j)}$ – вектор, що розпізнається; μ_m – функція належності вектора $x^{(j)}$ до контейнеру класу розпізнавання X_m^o.

У рівнянні (2.18) функція належності, для гіперсферичного контейнера класу розпізнавання X_m^o, розраховується за формулою (2.19)

$$\mu_m = 1 - \frac{d(x_m^* \oplus x^{(j)})}{d_m^*}, \qquad (2.19)$$

де x_m^*, d_m^* – оптимальні параметри машинного навчання: усереднена двійкова реалізація і радіус гіперсферичного контейнеру, відповідно.

Тож таким чином, під час контрольної перевірки визначається, за вирішальними правилами (2.18), належність реалізації класу, який розпізнається, одному із класів, котрі задані алфавітом. При цьому, вирішальні правила, через невелику розрахункову трудомісткість, відрізняються значною оперативністю.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ НАЗЕМНИХ ОБ'ЄКТІВ

3.1 Вхідний математичний опис для бортової системи розпізнавання

Для перевірки та експериментального підтвердження теоретичних припущень було змодельовано 3D-макет місцевості. На рисунку 3.1. подано зображення, яке було зроблено з борту безпілотного літального апарату типу квадракоптер приблизно на висоті 100 м.

Рисунок 3.1 – 3D-модель місцевості

За допомогою комп'ютерної симуляції буде проведена імітація знаходження зони інтересу розпізнавання, у даному випадку автомагістралі за наступним сценарієм:

1) Безпілотник на своїй максимальній висоті польоту повинен ідентифікувати зону інтересу;
2) Поступово наблизитися до неї;
3) У подальшому БПЛА буде здійснювати пошук потрібних об'єктів, які знаходяться в зоні інтересу;

Оскільки нас цікавить оптимізація вхідного математичного опису при зміні висоти польоту, тому розглянемо задачу ідентифікації у вигляді бінаризації карти. Таким чином на рисунку 3.1. було виділено два класи розпізнавання: X_1^0 – автомагістраль (інфраструктурні об’єкти); X_2^0 – густий ліс (об’єкти природного походження);

На рисунку 3.2 зображено обрані класи розпізнавання БСР. Розмір кожного кадру становить 75 на 75 пікселів.

а *б*

Рисунок 3.2 – Класи розпізнавання: а – клас X_1^0; б – клас X_2^0;

Враховуючи обраний метод дослідження, маємо наступний зміст вхідного математичного опису БСР:

1) словник різноманітних ознак розпізнавання;

2) алфавіт можливих класів розпізнавання;

3) вхідна навчальна матриця яскравості $\| y_{m,i}^{(j)} | m = \overline{1,M}; i = \overline{1,N},\ j = \overline{1,n} \|$

4) робоча, бінарна навчальна матриця $\| x_{m,i}^{(j)} \|$, яка в процесі машинного навчання адаптується до максимальної повної ймовірності прийняття вірних класифікаційних рішень;

5) нормоване поле допусків δ_H на яскравості ознак розпізнавання, котре визначає ділянку значень параметра δ поля контрольних допусків;

6) рівень селекції ρ_m координат усередненого двійкового вектору-реалізації класу розпізнавання X_m^o, котрий є рівнем квантування реалізацій вхідної навчальної матриці. За замовчуванням приймається $\rho_m = 0{,}5$.

3.2 Результати моделювання

Процес машинного навчання бортової системи розпізнавання відбувався за допомогою інформаційно-екстремальної технології. Це означає, що вхідна навчальна матриця була сформована за допомогою оброблення класів розпізнавання в декартовій системі координат. Як критерій оптимізації використовувалася модифікована міра Кульбака в нормованій формі (2.13).

На рисунку 3.3 продемонстровано графік залежності нормованого критерію оптимізації (2.13) від параметра δ, який визначає поле контрольних допусків і змінюється одночасно для всіх реалізацій. На цьому і на кожному подальшому графіку робочу область визначення функції критерію виділено темно сірим кольором.

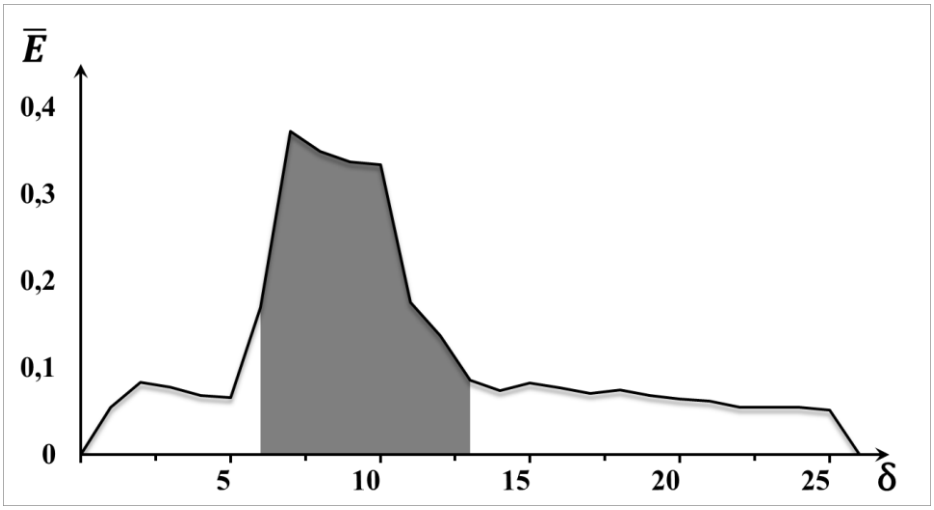


Рисунок 3.3 – Графік залежності інформаційного критерію (2.13) від параметра поля контрольних допусків при паралельній оптимізації СКД

При аналізі рисунку 3.3 можна помітити, що оптимальне значення параметру $\delta^* = 7$ (в градаціях яскравості). При цьому усереднений інформаційний критерій становить $\bar{E}^* = 0.37$. Оскільки не було досягнуто граничного значення, то додатково було прийнято рішення реалізувати алгоритм послідовної оптимізації (2.17). Під час якого параметр δ оптимізується окремо для кожної реалізації. Рисунок 3.4 показує зміну

нормованого критерію (2.13) при послідовній оптимізації поля контрольних допусків на ознаки розпізнавання

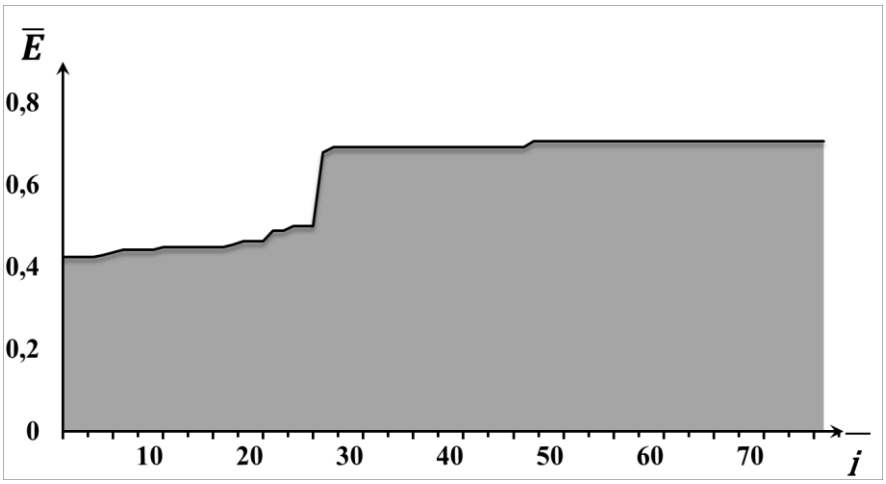
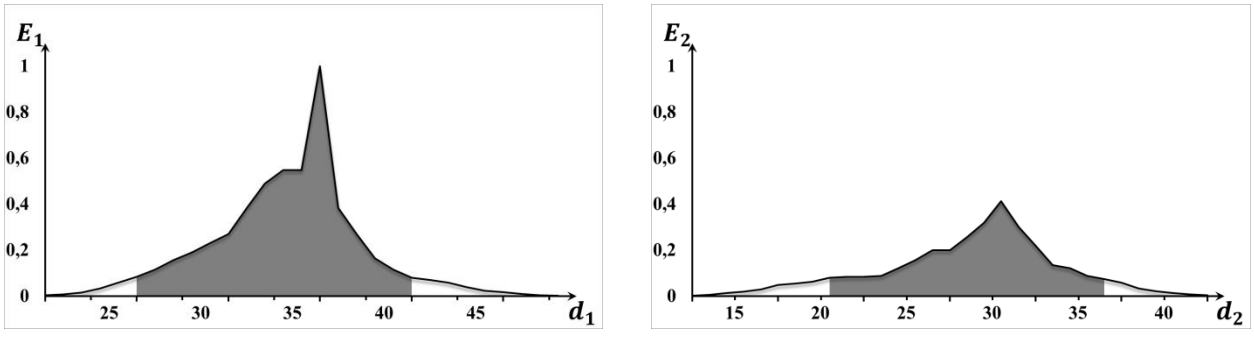


Рисунок 3.4 – Графік зміни усередненого інформаційного критерію в процесі послідовної оптимізації СКД

На рисунку 3.4 можна помітити, що усереднений інформаційний критерій оптимізації ще на першому прогоні, котрий розраховується відношенням кількості ітерацій $\bar{i}$ до кількості ознак *N*, досягнув значення в $\bar{E}^* = 0.7$.

Під час машинного навчання вдалося побудувати оптимальні вирішальні правила (2.18) в бінарному просторі Хеммінга. На рисунку 3.5 показано графіки залежності нормованого інформаційного критерію (2.13) від радіусів контейнерів розпізнавання.



а *б*

Рисунок 3.5 – Графіки залежності критерію (2.13) від радіусів контейнерів класів розпізнавання: а – клас X_1^o; б – клас X_2^o;

На рисунку 3.5 чітко видно, що оптимальні значення радіусів контейнерів класів розпізнавання становлять: $d_1^* = 36$ (тут і далі в кодових одиницях) – для класу X_1^0; $d_2^* = 30$ – для класу X_2^0;

Для перевірки ефективності проведеного машинного навчання було реалізовано алгоритм екзамену бортової системи розпізнавання. При цьому ця перевірка виконувалася у рамках комп'ютерної симуляції. Результатом цього екзамену є оцифроване зображення місцевості, рисунок 3.6.

Рисунок 3.6 – Результат класифікації кадрів

Візуальний аналіз рисунку 3.6. показує, що дорога розпізнається не достатньо чітко, але на даному етапі БПЛА достатньо просто ідентифікувати зону інтересу та почати наближатися до неї. Фотознімок місцевості з рисунку 3.6 був зроблений приблизно на висоті 100 метрів.

На рисунку 3.7. показано зображення місцевості, яке зробив безпілотник на висоті 30 метрів.

Рисунок 3.7 – 3D-модель місцевості

На рисунку 3.8, показано результат ідентифікації з використанням попередньо отриманих вирішальних правил.

Рисунок 3.8 – Результат класифікації кадрів

Можна помітити, що отриманий результат не є достовірним. Бортова система майже не змогла ідентифікувати автомагістраль. Це пов’язано із тим, що безпілотник змінив свою висоту польоту, а отже змінився масштаб фотознімків. Інакше кажучи вхідний математичний опис, рисунок 3.2, який

був зроблений на висоті 100 метрів не підходить для ідентифікації зображення, яке зроблено на висоті 30 метрів.

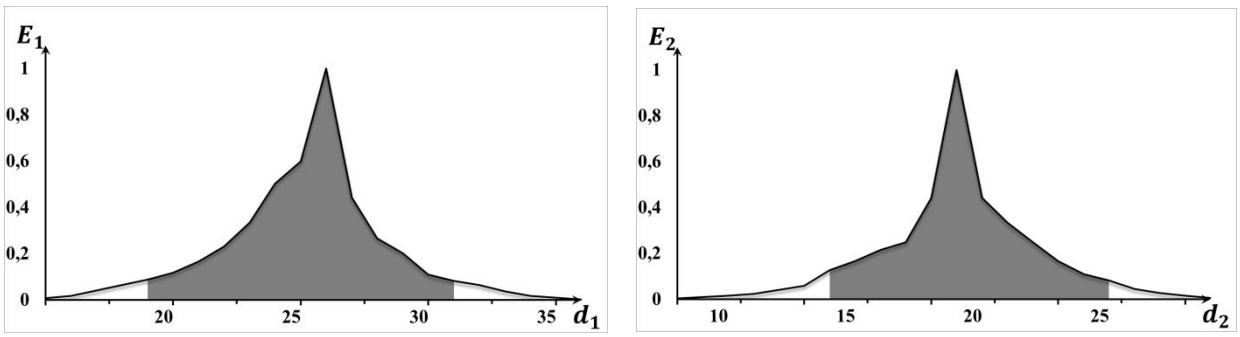
Одним із можливих, і не трудомістких, шляхів вирішення цієї проблеми є оптимізація вже отриманого вхідного математичного опису, рисунок 3.2. Наприклад за допомогою зміни розміру кадру розпізнавання [19].

На рисунку 3.9 показані кадри ділянок, рисунок 3.2, які були зменшені до розміру 55 на 55 пікселів.

а *б*

Рисунок 3.9 – Класи розпізнавання: а – клас X_1^0; б – клас X_2^0;

Було проведене перенавчання БСР з використанням паралельно-послідовної оптимізацією контрольних допусків. На рисунку 3.10 зображено графіки залежності нормованого інформаційного критерію (2.13) від радіусів контейнерів розпізнавання. Їх оптимальні значення дозволяють побудувати вирішальні правила (2.18) у рамках інформаційно-екстремального машинного навчання.



а *б*

Рисунок 3.10 – Графіки залежності критерію (2.13) від радіусів контейнерів класів розпізнавання: а – клас X_1^o; б – клас X_2^o;

На рисунку 3.5 чітко видно, що значення контейнерів для класів розпізнавання дорівнюють: $d_1^* = 26$ – для класу X_1^0; $d_2^* = 20$ – для класу X_2^0;

На рисунку 3.11 показаний результат екзамену, у вигляді оцифрованої карти місцевості.

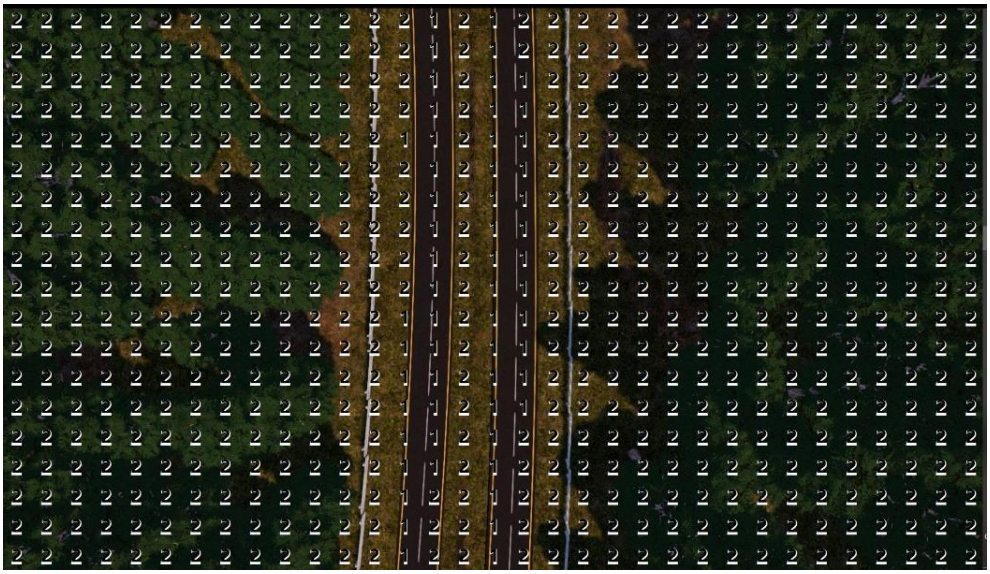


Рисунок 3.11 – Результат класифікації кадрів

Після візуального огляду рисунку 3.11 можна дійти висновку, що отриманий результат є високо достовірним. Це пов’язано із тим, що у більшості випадках зона інтересу ідентифікується без помилково.

Оптимізація вхідного математичного опису шляхом зміни розміру кадру [19] – є простим способом підвищення достовірності машинного навчання. Більше того цей метод можна використати для автономної бортової системи розпізнавання, яка б автоматично визначала оптимальний розмір кадру місцевості в залежності від висоти польоту безпілотника.

Із-за специфіки роботи БСР об’єкти, які вона повинна розпізнавати, досить часто є не стаціонарними і можуть займати довільне положення у кадрі. У такому випадку доцільно виконувати формування вхідної навчальної матриці в полярній системі координат. Як це, наприклад, показано в праці [20]. Такий підхід дозволить побудувати інваріантні вирішальні правила, а це зробить їх менш чутливими до положення об’єкту розпізнавання у кадрі.

3.3 Короткий опис програмної реалізації

Для написання розробленої бортової системи розпізнавання наземних об’єктів була використана мова програмування C#. 3д-макет місцевості та перевірка функціонування БСР під час екзамену проводилася при використанні Unreal Engine 4.

Клас decartCoord (табл. 3.1) зберігає інформації про класи розпізнавання.

Таблиця 3.1 – Опис складових класу decartCoord

ЗМІНІ			
Назва	Тип	Модифікатор доступу	Призначення
img	Bitmap	private	Клас розпізнавання у вигляді зображення
bitMap	int[,]	private	Вхідна навчальна матриця
`binaryMap`	`int[,]`	private	Бінарна матриця
`etalonVektor`	`int[]`	private	Еталонний вектор класу або центр контейнеру розпізнавання
`N`	`int`	private	Кількість ознак (стовпці)
`n`	`int`	private	Кількість реалізацій (строки)
`ro`	`float`	private	Рівень квантування координат двійкового вектору
Методи			
Назва			Призначення
`public decartCoord(Bitmap source)`			Конструктор класу. Приймає зображення і виконує первинну ініціалізацію змінних
`public int getElem_EV(int i)`			Повертає значення еталонного вектору за його індексом.
`public int getElem_BM(int i, int j)`			Повертає значення бінарної карти за індексом.
`public int getColumns()`			Повертає кількість ознак класу розпізнавання (N)
`public int getRows()`			Повертає кількість реалізацій класу розпізнавання (n)
`public int getPixel(int i, int j)`			Повертає значення яскравості пікселя за його індексом.

`public void setPixel(int i, int j, int value)`	Встановлює значення яскравості пікселя за його індексом.
`public void make_etalon_vektor()`	Створює еталонний вектор
`public void make_binary_map(int[] vd, int[] nd)`	Створює бінарну карту за наявними допусками

Клас teachDecart (табл 3.2) відповідає за етап навчання в рамках інформаційно-екстремальної технології.

Таблиця 3.2 – Опис складових класу teachDecart

ЗМІНІ			
Назва	Тип	Модифікатор доступу	Призначення
`decarts`	`List<>`	private	Колекція класів на яких навчається БСР
`para_d`	`int[]`	private	Відстань до найближчого сусіда
`para`	`int[]`	private	Список усіх сусідів
`sk`	`int[,]`	private	0-й стовпчик – це відстані від сусідніх реалізацій до свого EV 1-й стовпчик – це відстані від свої реалізацій до свого EV
`sk_para`	`int[,]`	private	0-й стовпчик – це відстані від сусідніх реалізацій до сусіднього EV 1-й стовпчик – це відстані від свої реалізацій до сусіднього EV
`em`	`float[]`	private	Значення інформаційного критерію для усіх класів
`my_do`	`int[]`	private	Значення радіусів контейнерів для усіх класів
`vd`	`int[]`	private	Верхні допуски
`nd`	`int[]`	private	Нижні допуски
`limit_d`	`int`	private	Максимальне можливе значення дельта
`d`	`int`	private	Дельта
`used_class`	`int`	private	Базовий клас
`points_parallelKFE`	`float[,]`	private	Результати паралельної оптимізації
`points_finallyKFE`	`float[,,]`	private	Результати оптимізації радіусів
Методи			
Назва			Призначення
`public teachDecart(List<decartCoord> sender)`			Конструктор класу. Приймає класи і виконує первинну ініціалізацію змінних
`public int getLimitD()`			Одержання граничного значення

	дельта
`public float getPointParallel(int i, int j)`	Одержання результату паралельної оптимізації за його індексами
`public int getDelta()`	Одержання значення дельта
`public float getPointFinally_KFE(int k, int i, int j)`	Одержання результату оптимізації радіусів за його індексами
`public Bitmap getClass(int i)`	Одержання зображення із класу
`public decartCoord getDecart(int i)`	Одержання класу
`public float getEM(int i)`	Одержання КФЕ для певного класу
`public int getRadius(int i)`	Одержання радіусу гіперповерхні для певного класу
`public int getElemVD(int i)`	Одержання значення верхнього допуску за його індексом
`public int getElemND(int i)`	Одержання значення нижнього допуску за його індексом
`public int[] getVD()`	Одержання масиву верхніх допусків
`public int[] getND()`	Одержання масиву нижніх допусків
`void realization()`	Процес машинного навчання, під час якого відбувається створення бінарної матриці, еталонних векторів, розбиття на пари і оптимізацію радіусів
`bool make_dopusk(int k, int delta)`	Змінна допусків на основі k-го класу
`void make_para()`	Поділ класів на пари
`double INFK(int INFK_d, ref float INFK_d1, ref float INFK_betta, int k)`	Розрахунок критерію функціональної ефективності
`void make_sk(int k)`	Знаходження відстаней для своїх і сусідніх реалізацій
`void make_myDo()`	Оптимізація радіусів
`bool propusk_KFE(int radius, int k)`	Перевірка на робочу область
`public float average_KFE()`	Усереднене значення критерію
`public void parallelOptimization()`	Паралельна оптимізація СКД

Клас machineLearn (табл 3.3) відповідає за етап екзамену та зміну вхідного математичного опису.

Таблиця 3.3 – Опис складових класу machineLearn

ЗМІНІ			
Назва	Тип	Модифікатор доступу	Призначення
`decarts`	`List<>`	private	Усі класи розпізнавання
`teach_decart`	`teachDecart`	public	Екземпляр класу, який виконує машинне навчання і зберігає результати по ньому.

Методи	
Назва	Призначення
`public machineLearn()`	Конструктор класу. Виконує первинну ініціалізацію змінних
`public void realization_decart()`	Послідовно викликає методи оптимізації вирішальних правил.
`public void make_fragment(Bitmap bitmap)`	Одержує значення яскравості пікселів із зображення
`public int examsDecart(Bitmap bitmap)`	Виконує алгоритм екзамену. Повертає значення класу до якого належить зображення.
`private void optimal_size()`	Зміна розміру кадру розпізнавання
`private void loadClass()`	Формування вхідного математичного опису
`private Bitmap resize_Image(int coord, int Nwith, int NHeigt, Bitmap image)`	Фізична зміна розміру зображення

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено бортову систему розпізнавання наземних об’єктів, яка використовує інформаційно-екстремальне машинне навчання. У рамках обраної технології було здійснено паралельно-послідовну оптимізація системи контрольних допусків на ознаки розпізнавання. Такий підхід дозволив отримати оптимальні геометричні параметри контейнерів класів розпізнавання.

За допомогою оптимальних геометричних параметрів контейнерів класів розпізнавання було утворено вирішальні правила, які показали значну оперативність при прийнятті класифікаційних рішень.

Для формування вхідного математичного опису та перевірки працездатності бортової системи розпізнавання було змодельовано 3Д-макет місцевості. Це дозволило перевірити результати класифікації об’єктів під час динамічного випробування безпілотника.

Було висунуто та експериментально доведено припущення про недоцільність використання вхідного математичного опису, який був зроблений на різній висоті польоту безпілотника.

Реалізовано алгоритм зміни розміру класів розпізнавання, при зміні висоти польоту безпілотного літального апарату.

У подальшому необхідно наділити вирішальні правила властивістю інваріантності до довільного положення об’єкту у кадрі. Це можна зробити, наприклад, через удосконалення методу формування вхідної навчальної матриці методом оброблення кадрів цифрового зображення регіону в полярній системі координат.

Наступним етапом дослідження є створення динамічного випробування, під час якого безпілотник спочатку, на максимальній своїй висоті, знаходить зону інтересу. Наближається до неї. А потім виконує перехід до набору інших вирішальних правил, за допомогою яких виконується пошук об’єкту інтересу.

СПИСОК ЛІТЕРАТУРИ

1. "A Brief History of UAVs". Howstuffworks.com. 22 July 2008. Retrieved 8 January 2015. Електронний доступ: https://science.howstuffworks.com/reaper.htm

2. Ударні БПЛА України. Електронний доступ: https://defence-ua.com/weapon_and_tech/udarni_bpla_ukrajini-456.html

3. Слюсар, Вадим. (2010). Передача данных с борта БПЛА: стандарты НАТО.. Электроника: наука, технология, бизнес. – 2010. - № 3. с. С. 80 – 86.

4. Слюсар, Вадим. (2013). Радиолинии связи с БПЛА: примеры реализации.. Электроника: наука, технология, бизнес. – 2010. - № 5. с. С. 56 – 60.

5. Tomlin, C. Dana (1990). Geographic information systems and cartographic modeling. Prentice Hall series in geographic information science. Prentice Hall. Retrieved 5 January 2017.

6. National Security and Defense Council of Ukraine. Електронний доступ: https://covid19.rnbo.gov.ua/

7. ArcGIS Solutions. Електронний доступ:

https://www.arcgis.com/apps/solutions/index.html?gallery=true&industry=Defense&sortField=relevance&sortOrder=desc#home

8. Discover QGIS. Електронний доступ: https://qgis.org/en/site/about/index.html

9. Анохин П.К. Биология и нейрофизиология условного рефлекса — М.: Медицина, 1968. – 547 с.

10. Аркадьев А. Г., Браверман Э. М. Обучение машины распознаванию образов. — М.: Наука, 1964.

11. Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: классификация и снижение размерности. — М.: Финансы и статистика, 1989.

12. Краснопоясовський А.С. Інформаційний синтез інтелектуальних систем керування: підхід, що грунтується на методі функціонально-статистичних випробувань.– Суми, 2004.– 261 с.

13. Banzhaf, Wolfgang; Nordin, Peter; Keller, Robert; Francone, Frank (1998). Genetic Programming – An Introduction

14. Саймон Д. Алгоритмы эволюционной оптимизации. — М.: ДМК Пресс, 2020. — 940 с.

15. Ян Лекун. Как учится машина. Революция в области нейронных сетей и глубокого обучения, 2021.

16. Заде, Л.А. (1965). Нечеткие множества. Информация и управление (8), стр. 338–353.

17. Довбиш А.С. Основи проектування інтелектуальних систем: Навчальний посібник.– Суми: Видавництво Сум ДУ, 2009.– 171 с.

18. Довбиш А.С. Інтелектуальні інформаційні технології в електронному навчанні / А.С. Довбиш, А.В. Васильєв, В.О. Любчак. – Суми: Видавництво СумДУ, 2013.– 177 с.

19. Protsenko O., Savchenko T., Myronenko M., Prikhodchenko O. Informational and extreme machine learning for onboard recognition system of ground objects. Proceedings - 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies, DESSERT 2020. Pages: 213-218

20. Dovbysh, A., Naumenko, I., Myronenko, M., Savchenko, T. Information-extreme machine learning on-board recognition system of ground objects with the adaptation of the input mathematical description. 3rd International Workshop on Computer Modeling and Intelligent Systems, CMIS 2020; National University "Zaporizhzhia Polytechnic "Zaporizhzhia; Ukraine; 27 April 2020 to 1 May 2020; CEUR Workshop Proceedings, Volume 2608, 2020, Pages 913-925.

ДОДАТОК А

```
using System.Drawing;

namespace MagWork.machineLearning
{
   class decartCoord
   {
      Bitmap img;
      int[,] bitMap;
      int[,] binaryMap;
      int[] etalonVektor;

      int N;
      int n;
      float ro = 0.5f;
      int[,] RGB;
      public decartCoord(Bitmap source)
      {
         img = source;

         n = img.Height;
         N = img.Width;

         etalonVektor = new int[N];
         binaryMap = new int[N, n];
         bitMap = new int[N, n];

         RGB = new int[N, 3];
         make_RGB();
      }
      public Bitmap getImg()
      {
         return img;
      }
      public int getColumns()
      {
         return N;
      }
      public int getRows()
      {
         return n;
      }
      public void setBitMap(int i, int j, int val)
      {
         bitMap[i, j] = val;
      }
      public int getElemBitMap(int i, int j)
      {
         return bitMap[i, j];
      }
      public int[,] getBitmap()
      {
         return bitMap;
      }
      public int[,] getBM()
      {
         return binaryMap;
      }
      public int getRGB(int i, int j)
      {
         return RGB[i, j];
      }
      public void setBitmap(int[,] bitmap)
      {
         for (int i = 0; i < N; i++)
            for (int j = 0; j < n; j++)
               bitMap[i, j] = bitmap[i, j];
      }
      public int getElemEV(int i)
      {
         return etalonVektor[i];
      }
      public int getElemBM(int i, int j)
      {
         return binaryMap[i, j];
      }
      public void makeEV()
      {
         for (int i = 0; i < N; i++)
         {
            float ev_sum = 0;
            for (int j = 0; j < n; j++)
               ev_sum += binaryMap[i, j];
            if ((ev_sum / n) >= ro) etalonVektor[i] = 1;
            else etalonVektor[i] = 0;
         }
      }
```

```
        public void makeBM(int[] vd, int[] nd)
        {
            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    if (bitMap[i, j] >= nd[i] && bitMap[i, j] <= vd[i])
                        binaryMap[i, j] = 1;
                    else
                        binaryMap[i, j] = 0;
                }
            }
        }
        void make_RGB()
        {
            for (int i = 0; i < N; i++)
            {
                int R = 0, G = 0, B = 0;
                for (int j = 0; j < n; j++)
                {
                    Color color = img.GetPixel(i, j);

                    R += color.R;
                    G += color.G;
                    B += color.B;
                }
                RGB[i, 0] = R / n;
                RGB[i, 1] = G / n;
                RGB[i, 2] = B / n;
            }
        }
    }
}

            using System;
            using System.Collections.Generic;

            namespace CourseWork.machineLearning
            {
                class teachDecart
                {
                    List<decartCoord> decartsClasses = new List<decartCoord>();
                    int[] paraD;
                    int[] paraClass;
                    int[,] distance;
                    int[,] distance_para;
                    float[] informationM;
                    int[] radiuses;

                    int[] vDopusk;
                    int[] nDopusk;

                    int limitForD = 80;
                    int optimalD;
                    float[,] points_parallel_KFE { get; set; }
                    float[,,] points_finally_KFE { get; set; }
                    public teachDecart(List<decartCoord> sender)
                    {
                        foreach (var item in sender)
                            decartsClasses.Add(item);
                        distance = new int[2, decartsClasses[0].getRows()];
                        distance_para = new int[2, decartsClasses[0].getRows()];

                        paraD = new int[decartsClasses.Count];
                        paraClass = new int[decartsClasses.Count];

                        informationM = new float[decartsClasses.Count];
                        radiuses = new int[decartsClasses.Count];

                        vDopusk = new int[decartsClasses[0].getColumns()];
                        nDopusk = new int[decartsClasses[0].getColumns()];

                        points_parallel_KFE = new float[limitForD, 2];
                        points_finally_KFE = new float[decartsClasses.Count, decartsClasses[0].getColumns(), 2];
                    }
                    public decartCoord getDecart(int i)
                    {
                        return decartsClasses[i];
                    }
                    public int getColumns()
                    {
                        return decartsClasses[0].getColumns();
                    }
                    public int getRows()
                    {
                        return decartsClasses[0].getRows();
                    }
                    public int getRadius(int i)
                    {
```

```
    return radiuses[i];
}
public int[] getVD()
{
    return vDopusk;
}
public int[] getND()
{
    return nDopusk;
}
public int getDelta()
{
    return optimalD;
}
public float getEM(int i)
{
    return informationM[i];
}
public int getLimitD()
{
    return limitForD;
}
public float getPointFinallyKFE(int k, int i, int j)
{
    return points_finally_KFE[k, i, j];
}
public float getPointParallelKFE(int i, int j)
{
    return points_parallel_KFE[i, j];
}
public int getElemRGB(int k, int i, int j)
{
    return decartsClasses[k].getRGB(i, j);
}
void realization()
{
    foreach (var decart in decartsClasses)
        decart.makeBM(vDopusk, nDopusk);
    for (int k = 0; k < decartsClasses.Count; k++)
        decartsClasses[k].makeEV();

    make_para();
    make_myDo();
}
bool make_dopusk(int k, int delta)
{
    for (int i = 0; i < decartsClasses[k].getColumns(); i++)
    {
        int sum = 0;
        for (int j = 0; j < decartsClasses[k].getRows(); j++)
            sum += decartsClasses[k].getElemBitMap(i, j);

        sum /= decartsClasses[k].getRows();

        vDopusk[i] = sum + delta;
        nDopusk[i] = sum - delta;

        if (vDopusk[i] > 255 || nDopusk[i] < 0)
        {
            limitForD = delta;
            return false;
        }

    }
    return true;
}
void make_para()
{
    for (int k = 0; k < decartsClasses.Count; k++)
    {
        paraD[k] = 0;
        int ev_sum1 = decartsClasses[k].getColumns();
        for (int k1 = 0; k1 < decartsClasses.Count; k1++)
        {
            if (k1 != k)
            {
                int ev_sum = 0;
                for (int i = 0; i < decartsClasses[k].getColumns(); i++)
                {
                    if (decartsClasses[k].getElemEV(i) != decartsClasses[k1].getElemEV(i))
                        ev_sum++;
                }
                if (ev_sum <= ev_sum1)
                {
                    ev_sum1 = ev_sum;
                    paraClass[k] = k1;
                    paraD[k] = ev_sum1;
                }
            }
```

```
        }
    }
}
double INFK(int INFK_d, ref float INFK_d1, ref float INFK_betta, int k)
{
    int k1 = 0, k4 = 0;
    for (int INFK_j = 0; INFK_j < decartsClasses[k].getRows(); INFK_j++)
    {
        if (distance[0, INFK_j] <= INFK_d) k1++;
        if (distance[1, INFK_j] <= INFK_d) k4++;
    }
    INFK_d1 = (float)k1 / (float)decartsClasses[k].getRows();
    INFK_betta = (float)k4 / (float)decartsClasses[k].getRows();

    float d1_b = INFK_d1 - INFK_betta;
    double best_res = 1 * Math.Log((1 + 1 + 0.001) / (1 - 1 + 0.001));
    return d1_b * Math.Log((1 + d1_b + 0.001) / (1 - d1_b + 0.001)) / best_res;
}
void make_sk(int k)
{
    for (int j = 0; j < decartsClasses[k].getRows(); j++)
    {
        distance[1, j] = 0;
        distance[0, j] = 0;
        distance_para[1, j] = 0;
        distance_para[0, j] = 0;
        for (int i = 0; i < decartsClasses[k].getColumns(); i++)
        {
            if (decartsClasses[k].getElemEV(i) != decartsClasses[paraClass[k]].getElemBM(i, j)) distance[1, j] += 1;
            if (decartsClasses[k].getElemEV(i) != decartsClasses[k].getElemBM(i, j)) distance[0, j] += 1;

            if (decartsClasses[paraClass[k]].getElemEV(i) != decartsClasses[paraClass[k]].getElemBM(i, j)) distance_para[1, j] += 1;
            if (decartsClasses[paraClass[k]].getElemEV(i) != decartsClasses[k].getElemBM(i, j)) distance_para[0, j] += 1;
        }
    }
}
void make_myDo()
{
    float t_d1 = 0, t_betta = 0;
    points_finally_KFE = new float[decartsClasses.Count, decartsClasses[0].getColumns(), 2];

    for (int k = decartsClasses.Count - 1; k >= 0; k--)
    {
        informationM[k] = 0;
        radiuses[k] = 0;
        make_sk(k);
        float d_correct = 999;
        float d_tmp;
        for (int d = 1; d < decartsClasses[k].getColumns(); d++)
        {
            double e = INFK(d, ref t_d1, ref t_betta, k);
            if (e >= informationM[k] && t_d1 >= 0.5 && t_betta <= 0.5 && d < paraD[k])
            {
                // em[k] = (float)e;
                if (e > informationM[k])
                    radiuses[k] = d;
                informationM[k] = (float)e;

                d_tmp = (float)d / paraD[k];
                if (e == informationM[k] && d_tmp < d_correct)
                {
                    d_correct = d_tmp;
                    radiuses[k] = d;
                }
            }
            if (e > informationM[k] && radiuses[k] == 0)
            {
                d_correct = (float)d / paraD[k];
                informationM[k] = (float)e;
            }
            if (t_d1 >= 0.5 && t_betta <= 0.5 && d < paraD[k])
                points_finally_KFE[k, d, 1] = (float)e;
            points_finally_KFE[k, d, 0] = (float)e;
        }
    }
}
bool propusk_KFE(int radius, int k)
{
    float t_d1 = 0, t_betta = 0;

    INFK(radius, ref t_d1, ref t_betta, k);
    if (t_d1 >= 0.5 && t_betta <= 0.5)
        return true;
    return false;
}
float average_KFE()
{
    float avr = 0;
    for (int k = 0; k < decartsClasses.Count; k++)
```

```
                avr += informationM[k];
            avr /= decartsClasses.Count;
            return avr;
        }
        public void parallelOptimization(int now_class)
        {
            float em_last_d = 0;
            float radius = 0;

            for (int delta = 1; delta <= limitForD; delta++)
            {
                if (!make_dopusk(now_class, delta)) break;
                realization();

                float em_now = average_KFE();
                if (propusk_KFE(radiuses[now_class], now_class))
                {
                    points_parallel_KFE[delta - 1, 1] = em_now;

                    float t_d1 = 0, t_betta = 0;
                    INFK(radiuses[now_class], ref t_d1, ref t_betta, now_class);
                }
                points_parallel_KFE[delta - 1, 0] = em_now;

                if (em_now > em_last_d && propusk_KFE(radiuses[now_class], now_class))
                {
                    em_last_d = em_now;
                    optimalD = delta;
                    radius = radiuses[now_class];
                }
                else if (em_now == em_last_d && radius < radiuses[now_class] && propusk_KFE(radiuses[now_class], now_class))
                {
                    em_last_d = em_now;
                    optimalD = delta;
                    radius = radiuses[now_class];
                }
            }

            make_dopusk(now_class, optimalD);
            realization();
        }
    }
}

using System.Collections.Generic;
using System.Drawing;

namespace machineLearning
{
    public class machineLearn
    {
        List<decartCoord> decarClasses;
        public teachDecart teachDecart;
        public machineLearn()
        {
            decarClasses = new List<decartCoord>();
        }
        public void realization_decart()
        {
            teachDecart = new teachDecart decarClasses);

            teachDecart.parallelOptimization();
            teachDecart.sequentialOptimization();
        }
        public void make_fragment(Bitmap bitmap)
        {
            decartCoord decart = new decartCoord(bitmap);
            for (int i = 0; i < bitmap.Width; i++)
            {
                for (int j = 0; j < bitmap.Height; j++)
                {
                    Color color = bitmap.GetPixel(i, j);
                    decart.setPixel(i, j, (color.R + color.G + color.B) / 3);
                }
            }
            decarClasses.Add(decart);
        }
        private void optimal_size() {
            int countClasses = machine.getCountClass();
            machine = new machine_learn();
            int i_max = 20, x_max = 0;

            for (int k = 0; k < countClasses; k++)
            {
                var pictureBox = tabControl2.TabPages[0].Controls["panel1"].Controls["panelClass" + (k + 1)].Controls["pictureBoxClass" + (k + 1)] as
PictureBox;

                loadClassMachine(resize_Image(0, pictureBox.Image.Width - i_max, pictureBox.Image.Height - i_max, (Bitmap)pictureBox.Image));
            }
```

```
                machine.realization_decart(true, false);

                for (int k = 0; k < machine.getCountClass(); k++)
                {
                    var pictureBox = tabControl2.TabPages[0].Controls["panel1"].Controls["panelClass" + (k + 1)].Controls["pictureBoxClass" + (k + 1)] as
PictureBox;
                    loadClassPicture(pictureBox, resize_Image(x_max, pictureBox.Image.Width - i_max, pictureBox.Image.Height - i_max,
(Bitmap)pictureBox.Image));
                }

                machine = new machine_learn();
                loadClass();
            }

            private Bitmap resize_Image(int coord, int Nwith, int NHeigt, Bitmap image) {
                Rectangle rectangle = new Rectangle(coord, coord, Nwith, NHeigt);

                return image.Clone(rectangle, PixelFormat.Format32bppPArgb);
            }

            private void loadClass() {
                for (int i = 0; i < countClasses; i++)
                {
                    var pictureBox = tabControl2.TabPages[0].Controls["panel1"].Controls["panelClass" + (i + 1)].Controls["pictureBoxClass" + (i + 1)] as
PictureBox;
                    loadClassMachine(new Bitmap(pictureBox.Image));
                }
            }
        }
    }
```