МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна технологія розпізнавання об'єктів. Машинне навчання бортової системи розпізнавання наземних об'єктів»

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

Керівник роботи **Довбиш А.С.**

Студент гр. ІН-73 **Савченко Т.Р.**

Суми 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп’ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

“_____” _____2021 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-73 спеціальності “Комп'ютерні науки” денної форми навчання Савченко Т.Р.

Тема: “**Інформаційна технологія розпізнавання об’єктів. Машинне навчання бортової системи розпізнавання наземних об’єктів** ”

Затверджена наказом по СумДУ

№_____ от _____2021 р.

Зміст пояснювальної записки: 1) Аналіз предметної області; 2) Дослідження сучасного стану і перспектив розвитку бортових систем; 3) Визначення вимог до системи; 4) Оцінка функціональної ефективність машинного навчання бортової системи розпізнавання в режимі екзамену;

Дата видачі завдання “_____”_____2021 р.

Керівник випускної роботи _____ Довбиш А.С.

Завдання прийняв до виконання _____ Савченко Т.Р.

РЕФЕРАТ

Записка: 67 стор., 33 рис., 4 табл., 1 додаток, 17 джерел.

Об’єкт дослідження – слабоформалізований процес розпізнавання об’єктів на місцевості.

Мета роботи - підвищення функціональної ефективності бортової системи безпілотного авіаційного комплексу для розпізнавання об’єктів в змодельованому середовищі.

Методи дослідження — детерміновано-статистичні методи розпізнавання образів, математичний аналіз вхідних даних, інформаційно-екстремальна інтелектуальна технологія.

Результати – розроблено і програмно реалізовано алгоритм машинного навчання, на основі інформаційно-екстремальної інтелектуальної технології, для безпілотного літального апарату. Додатково було висунуто і експериментально підтверджено гіпотезу про доцільність створення системи контрольних допусків на базі класу з найбільшою дисперсією, бо із-за своєї різноманітності ознак він буде найближчим сусідом до будь-якого іншого класу з алфавіту. Для перевірки ефективності машинного навчання була змодельована 3D-модель місцевості, яка відповідає реальним умовам. Окрім цього було здійснене динамічне випробування, де безпілотник під час свого польоту повинен був здійснювати ідентифікацію інфраструктурної, неперервної мережі типу «дорога».

ЗМІСТ

ВСТУП

У наш час активно розвиваються геоінформаційні системи (ГІС), їх головною задачею є візуальне представлення і опис певної місцевості. Як правило, це вузько спеціалізоване програмне забезпечення (ПЗ), яке дозволяє проводити аналіз окремо виділеної території.

У рамках бакалаврської роботи, нас цікавлять ГІС, які використовують для створення динамічних карт, головною метою яких є - оперативне отримання результатів аналізу території. Такі системи застосовуються для інспектування неперервних інфраструктурних мереж та пошуку їх несправності. Наприклад: аналіз цілісності дороги, труб водопостачання тощо.

Воєнна галузь приділяє особливу увагу розробці та впровадженню ГІС, бо саме під час воєнних дій вкрай необхідно швидко отримати актуальну геоінформацію з території супротивника. Це може бути: кількість та тип ворожої техніки; місце знаходження ворожих таборів; план місцевості, який допоможе організувати наступ;

Для швидкого і безпечного аналізу, відносно невеликої території, використовуються безпілотні авіаційні комплекси (БАК). Як правило, вони оснащенні високо роздільними камерами різного типу та, завдяки потужним процесорам, здатні виконувати складні обчислення.

Головною метою бакалаврської роботи є створення бортової системи розпізнавання (БСР) наземних об’єктів. Для цього необхідно вирішити дві глобальні задачі:

- отримати адекватний вхідний математичний опис місцевості, який повинен відповідати реальним умовам;
- розробити БСР, ефективність якої можна буде перевірити під час безпосередньої ідентифікації наземних об’єктів;

Кінцевим результатом буде симулятор, який дозволе моделювати бажану місцевість, де буде формуватися вхідний математичний опис і проводитимуться випробування.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Дослідження сучасних систем інтелектуального аналізу геоінформаційних даних

ГІС будується на основі, так званої, «базової карти», яка представляє із себе первинний опис тахеометричних даних певної місцевості. У найпростішому вигляді це може бути звичайна топографічна карта (Рис.1.1)

Рисунок 1.1 – Приклад топографічної карти міста Суми

Головним правилом при побудові такого плану місцевості (Рис.1.1) – є використання заздалегідь визначених умовних позначень. Тобто необхідно щоб у будь-якій частині карти, наприклад, ліс зображувався одним і тим самим чином.

Топографічні карти містять основну географічну інформацію території, але, зазвичай, цього замало для її повного аналізу. Річ у тім, що дослідження будь-якої місцевості потрібно проводити комплексно, з використанням розмаїтих статистичних даних, які залежать від поставленої задачі. Саме цим і займається геоінформаційна система, поєднанням найрізноманітнішої інформації про обране місце. Дуже часто базою для ГІС є топографічні карти або будь-яке інше модельне зображення території.

У геоінформаційних системах дані поділяються, за типом, на: просторові та атрибутивні.

Просторовими, або метричними, називають позиційні дані реального об’єкту, тобто це відображення його безпосереднього положення на місцевості. Інформацію цього типу розділяють на дві абстрактні категорії:

- дискретні (будинки, територіальні зони);
- неперервні (рельєф, рівень опадів, середньорічна температура);

У геоінформаційних системах, для відображення просторової інформації дискретних об’єктів, використовуються растрові зображення. Як правило, це базова карта, рисунок 1.2.

Рисунок 1.2 – Приклад растрової карти міста Суми

Для позначення неперервних об’єктів використовують векторний спосіб подачі інформації. Простіше кажучи, за допомогою геометричних примітивів. Це значно спрощує роботу та зменшує загальний розмір ГІС. До речі, дуже часто у GPS-навігаторах використовується саме такий спосіб представлення просторових даних (Рис.1.3).

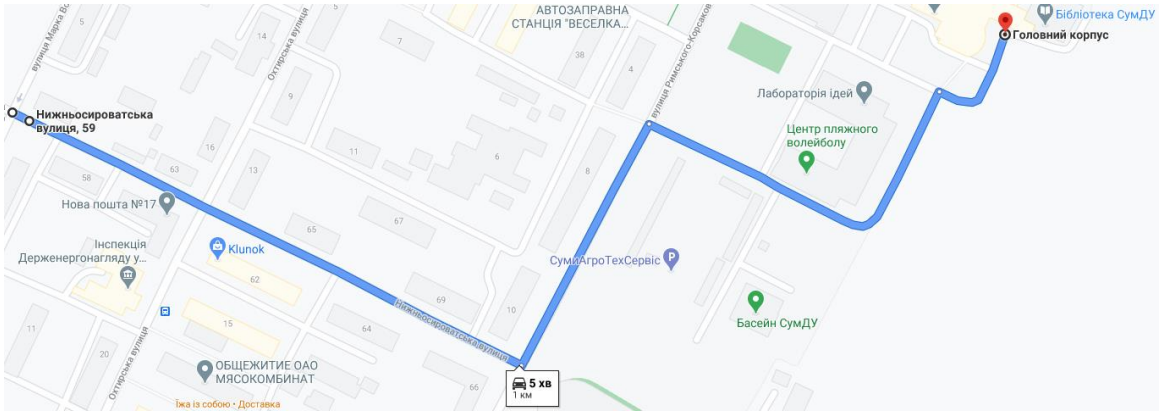


Рисунок 1.3 – Приклад GPS-карти з використанням векторних об’єктів

Окрім фізичних координат певного об’єкту, ГІС надає ще додаткову інформацію про нього. Це атрибутивний тип даних, форма і вид якого залежить від задач, які покладенні на геоінформаційну систему. Наприклад, можна розмалювати територію України зелено-червоним градієнтом. Таким чином, щоб регіони з найбільшою кількістю хворих на коронавірус позначалися насиченим червоним кольором, області з мінімальною кількістю захворілих мали зелене забарвлення. Вкрай важливо, щоб ГІС містила пояснення для таких умовних позначень.

Будь-яка геоінформаційна система повинна вміти фіксувати, зберігати, модифікувати, керувати, аналізувати і відображати, описані вище, просторові й атрибутивні дані (Рис.1.4). Інтерфейс, масштаб та особливості подання інформації залежать від вимог до ГІС.

Рисунок 1.4 – Приклад візуалізації еміграції населення України за допомогою ГІС Global Migration Data

Зазвичай сучасні, вузькоспеціалізовані ГІС додатково укомплектовані засобами інтелектуального аналізу даних. Їх вид та спосіб використання залежить від предметної галузі та типу даних з якими вони працюють. Наприклад, сільськогосподарська ГІС може мати систему підтримки прийняття рішень (СППР), яка дозволить висунути припущення про можливу врожайність території, на основі наявної інформації.

Незважаючи на те, що більшість геоінформаційних систем вирішують вузькопрофільні завдання, можна виділити ряд типових задач [1], які ставляться до сучасних ГІС:

1) проектування об’єктів на визначеній території;
2) аналіз географічної інформації та її подальша обробка;
3) вирішення транспортних задач;
4) ідентифікація об’єктів на місцевості;
5) оформлення результатів аналізу даних у вигляді карт різного типу;
6) аналіз та відображення змін об’єктів у часі;
7) підтримка при прийнятті рішень;
8) моделювання місцевості та подій на ній;

Сучасні геоінформаційні системи приймають найрізноманітніший вигляд і можуть виконувати широкий спектр задач в галузях, де необхідна просторова інформація про об’єкт. Проведемо невеликий огляд сфер діяльності в яких застосовуються ГІС:

- ***сільське господарство***: у цій сфері ГІС (Рис.1.5) використовуються для зберігання картографічних даних, які необхідні при аналізі введення сільського господарства. Наприклад: оцінка якості ґрунтів, їх агроекологічний стан, характер ґрунтового покриву, розподіл землі, степінь її забруднення, тощо. Комплексний аналіз цих факторів дозволяє зменшити ризики та максимізувати потенційну врожайність.

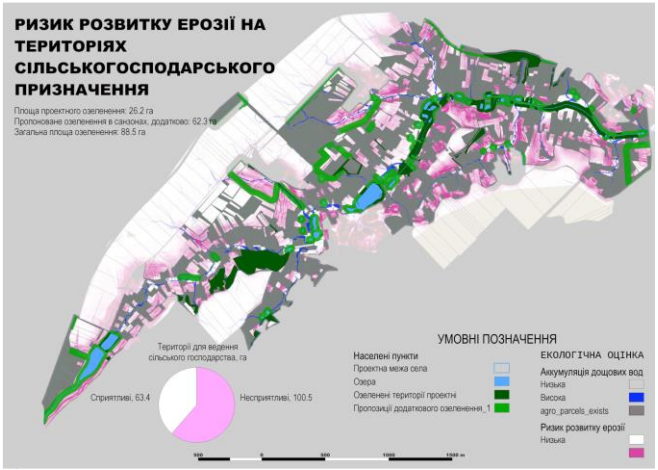


Рисунок 1.5 – Приклад сільськогосподарської ГІС

- ***екологія та природокористування***: характерною особливістю цих сфер є робота з даними, які були отримані з великих територій, на які впливають безліч чинників. ГІС допомагає прогнозувати майбутній стан довкілля та підтримувати його стабільність.
- ***воєнна галузь:*** приділяє надзвичайно велике значення комплексному аналізу геоінформаційних даних. При цьому вкрай важливо постійно отримувати актуальну інформацію, адже від неї може залежати результат конфлікту і людські життя. Дуже часто ГІС, які використовують воєнні (Рис.1.6), мають можливість створювати 3д-макети місцевості та моделювати, у реальному часі, атаки чи наступи.

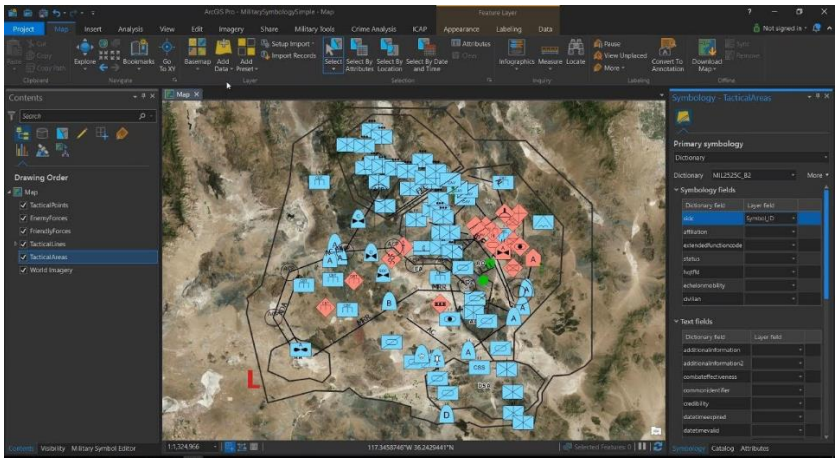


Рисунок 1.5 – Приклад воєнної ГІС

- ***транспортні мережі та комунікація***: для аналізу сучасних інфраструктурних мереж в ГІС використовуються динамічні карти, де в реальному часі можна прослідкувати рух об’єктів. Це дозволяє легко регулювати трафік та локалізувати несправності чи перешкоди.
- ***управління бізнесом:*** у випадку, коли підприємство має мережевий характер, а його філіали знаходяться в різних куточках країни, а то й світу, виникають складні управлінські проблеми. Необхідно постійно оперувати великою кількістю інформації з різних місць, аналізувати діяльність конкурентів. У таких випадках, при управлінні бізнесом, зручно впровадити ГІС.

- ***криміналістика***: правоохоронні органи використовують ГІС для ведення кримінальної картографії (Рис.1.6), щоб візуалізувати, в зручному вигляді, великі масиви даних, які містять інформацію про скоєнні правопорушення. Таким чином, при їх аналізі, вдається розробити план попередження майбутніх потенційних злочинів

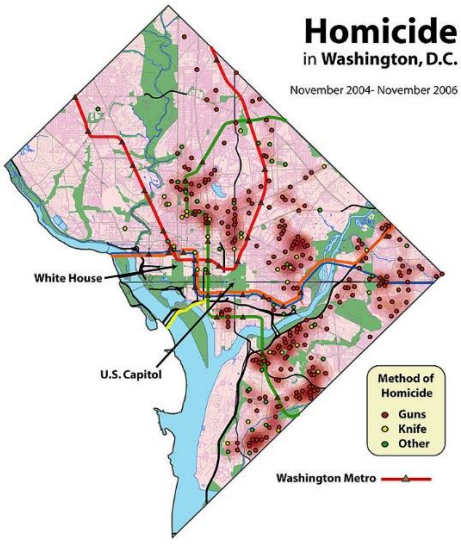


Рисунок 1.6 – Приклад кримінальної карти

Це зовсім не весь перелік галузей, де використовуються ГІС, але саме в них відбувається найактивніший розвиток геоінформаційних систем.

Можна помітити, що ГІС поділяються на загальні та вузькопрофільні. Наприклад, якщо нам необхідно лише зберігати і самостійно аналізувати дані певної території, то достатньо використати вже готові конструктори, наприклад QGIS [2]. Проте коли річ йде про специфічні проблеми для певної галузі, то виникає необхідність у створені унікальних ГІС, які будуть спроектовані для виконання вузькопрофільних завдань. Наприклад: інтелектуальний аналіз території; моделювання фізичних процесів на місцевості; ідентифікація об’єктів; прогнозування за допомогою інтелектуальних систем, тощо.

1.2 Сучасний стан і перспективи розвитку бортових систем безпілотних авіаційних комплексів для розпізнавання наземних об’єктів

Головним завданням бакалаврської роботи є розробка бортової системи розпізнавання (БСР) для безпілотного авіаційного комплексу (БАК), який є частиною військової геоінформаційної системи. БАК складається із:

- безпілотного літального апарату (БПЛА);
- пункту дистанційного керування;

Зазвичай, БАК здійснює керування одразу декількох БПЛА. При цьому дані передаються по зашифрованому каналу, таким чином, щоб зловмисник не мав доступу до цієї інформації.

Безпілотні літальні апарати мають різне призначення, але нас цікавить їх застосування у воєнній галузі. Військові використовують БПЛА для проведення розвідки, корегування ударів по наземних цілях або їх безпосередньої атаки. Розглянемо основні характеристики БАК, які пропонують сучасні виробники:

Розвідувально-ударний БПЛА MQ-9 Reaper – американський безпілотник, зовнішній вигляд якого показано на рисунку 1.7 [3].

Рисунок 1.7 – Зовнішній вигляд БПЛА MQ-9 Reaper

Апарат здатний знаходитися в повітрі до 27 годин, підніматися на висоту 15 000 м, розвивати швидкість понад 400 кілометрів на годину і спроможний

підняти вантаж вагою до 1700 кг. MQ-9 Reaper має на борту оптично-електрону систему спостереження та наведення AN/AAS-52 (Рис.1.8), яка виготовляється компанією The Raytheon. Вона складається із камери видимого та інфрачервоного діапазону, електронно-оптичного перетворювача (ЕОП), лазерного дальноміра-цілевказівника. За необхідності можна встановити датчик руху цілі або радар із синтезованою апаратурою, для наведення керованих бомб типу JDAM.

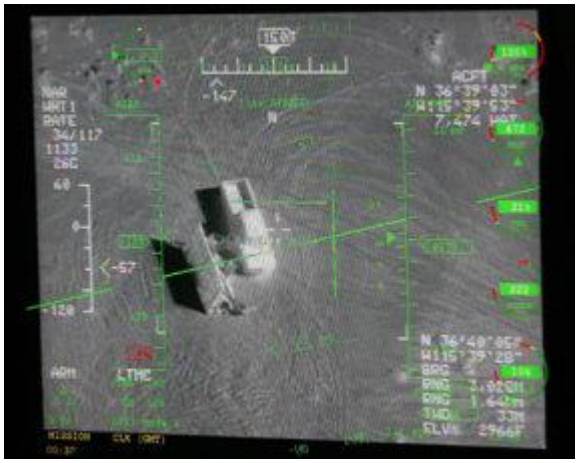


Рисунок 1.8 – Приклад зображення, яке було отримане з борту MQ-9 Reaper

MQ-9 Reaper експлуатується армією з 2007 року. Основні країни, які мають ці БПЛА на озброєнні: США, Великобританія, Італія, Туреччина, Франція.

Байрактар ТБ2 – турецький розвідувальний безпілотний літальний апарат, розроблений компанією Baykar Makina. Зовнішній вигляд безпілотника зображено на рисунку 1.9 [4].

Рисунок 1.9 – Загальний вигляд БПЛА Байрактар ТБ2

Керування усіма модулями апарату відбувається з наземної станції. Байрактар оснащений двигуном внутрішнього згорання, потужністю 100 к.с., це дозволяє йому розвинути швидкість понад 222 км/год і максимальну висоту 8200 м. Безпілотник здатний підіймати в повітря вантаж вагою до 55 кг. Керувати байрактаром можна в радіусі 150 км.

БПЛА оснащений декількома камерами і системою спостереження, що дозволяє виконувати розвідку території та ідентифікацію об'єктів. Крім того, Байрактар може нести на борту чотири керовані протитанкові ракети UMTAS з лазерним наведенням або високоточні авіабомби Roketsan MAM-C, MAM-L, які здатні вражати ціль на відстані до 8 км [4][5].

На даний момент офіційно використовують Байрактари ТБ2 наступні країни: Туреччина, Лівія, Катар, Україна, Азербайджан.

А1-СМ «Фурія» - це український, багатоцільовий безпілотний авіаційний комплекс, який призначений для ведення повітряної розвідки вдень та вночі, визначення координат цілі, коригування артилерійського вогню, конвоювання. На рисунку 1.10 показаний зовнішній вигляд БПЛА «Фурія».

Рисунок 1.10 – Загальний вигляд БПЛА «Фурія»

Апарата має два цифрових канали керування, інформація по яких шифрується за допомогою алгоритму AES 256. Супроводження цілі та визначення її координат здійснюється автоматично з використанням супутникових систем: GPS, GLONASS, GALILEO. «Фурія» має електронний

двигун, який робить її політ майже безшумним. Максимально можливий радіус, для здійснення керування, 30 км.

Слід зазначити, що головною задачею цього безпілотника – є розвідка, тому БПЛА має невеликий розмір. Його не чутно і майже не помітно в небі. На рисунку 1.11 можна побачити приблизні габарити безпілотника.

Рисунок 1.11 – Запуск БПЛА «Фурія»

«Фурія» містить два оптичних модуля: денний, на базі Sony FSB-H11, і нічний, який використовує Flir Tau 640x480 30Hz. Крім цього, БПЛА обладнаний тепловізором і фотокамерою IXUS 285 HS з 20.2 mpx. CMOS.

Квадрокоптер THOR. Зазвичай найдешевшим БПЛА є квадрактоптери. Вони набагато менші від планерів і можуть зависати у повітрі, що досить корисно для конвоювання або трекінгу техніки.

Прикладом такого коптеру є ізраїльський THOR (Рис.1.12), який був розроблений фірмою Elbit Systems [6].

Рисунок 1.12 – Загальний вигляд квадрокоптеру THOR

Головним призначенням THOR є спостереження за цілю та розвідка території. Безпілотник здатний знаходитися в повітрі при температурах від мінус 40 градусів за Цельсієм до плюс 65. THOR може зберігати стабільність при сильному вітру, в хмарах пилу та під час дощу – ще недавно це могли робити лише відносно габаритні БПЛА.

Квадрокоптер THOR має наступні характеристики: тривалість безперервного польоту 75 хвилин; вага вантажу до 3 кг; час розгортання 3 хв; максимальна швидкість досягає 65 км/год, робоча приблизно 40 км/год; висота польоту 610 метрів. При цьому на борту безпілотника є сенсорні та оптичні системи.

Проаналізувавши різні типи безпілотних літальних апаратів можна зробити наступні висновки:

1) Сучасні БПЛА, у воєнній галузі, зазвичай, використовуються як транслятори зображень наземних об’єктів, які аналізуються операторами з метою розв’язання конкретних задач. Наприклад: розпізнавання об’єкту, корегування атаки, конвоювання, тощо.

2) Орієнтація на глобальні і локальні мережі GPS робить БПЛА слабо захищеним від засобів радіоелектронної протидії (РЕП) і створює сприятливі умови для організації вторгнень і кібератак. Виникає потреба у наданні бортовим системам властивості автономного функціонування.

3) З метою розширення функціональних можливостей БАК та їх захищеності, актуальною задачею є створення автономних БСР наземних природних, інфраструктурних та інших малогабаритних об’єктів на основі сучасних методів інтелектуального аналізу даних. Крім того, розв’язання цієї задачі дозволить підвищити функціональну ефективність бортової системи для розпізнавання навігаційних перешкод і повітряних загроз.

Відомо, що США розробляє перспективний БАК з умовною назвою GS-2, основною перевагою якого буде наявність автономної БСР. Завершення цього проекту планується в 2030 році.

1.3 Огляд методів інтелектуального аналізу даних

Основна задача, яку, у рамках бакалаврської роботи, повинна вирішувати БСР – ідентифікація наземних об’єктів. Це можна зробити лише за допомогою методів розпізнавання образів, які, в свою чергу, базуються на статистичній теорії прийняття рішень.

Розпізнавання образів є лише частиною методів інтелектуального аналізу даних, тому, в сучасній літературі, прийнято використовувати загальний термін: «автоматична класифікація». Її головною цілю є створення комп’ютерних систем, які здатні навчатися, розрізняти та класифікувати образи. Згідно П.К. Анохіну [7] на фізіологічному рівні розпізнавання об’єкту, для будь-якої живої істоти, складається із двох етапів:

- процес навчання, де здійснюється виокремлення і запам’ятовування характерних ознак;
- етап екзамену, безпосереднє розпізнавання об’єкту;

Враховуючи це сформуємо загальну постановку задачі для ідентифікації функціонального стану системи розпізнавання (СР). Нехай, на етапі навчання нам потрібно знайти оптимальне, в інформаційному розумінні, розбиття простору ознак розпізнавання (ОР) на скінчену кількість класів. Під час екзамену від системи вимагається, через обмежене число випробувань, у режимі функціонування СР, прийняти високо достовірне рішення про належність вектору-реалізацій образу, що розпізнається, до деякого класу, з апріорно визначеної множини $\left\{X_m^*\right\}$.

Переважна більшість сучасних алгоритмі класифікації образів базуються на гіпотезі компактності. Згідно неї реалізації одного класу відображаються в просторі ознак як геометрично близькі точки утворюючи при цьому «компактні» згущення. Існують різні метрики компактності, але у найпростішому випадку можна вважати, що реалізації э компактними, коли відстань між ними не перевищує деяке порогове значення.

Основою більшості сучасних інтелектуальних алгоритмів є статистистичні методи. Вони базуються на байесівському вирішальному правилі: порівняння обчислених, у процесів машинного навчання системи прийняття рішень (СПР), 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, $m,l = \overline{1,M}$.

Головною задачею статистистичних методів можна вважати набір репрезентативних вибірок і створення на їх основі апріорних ймовірностей $p(\mu_m / \gamma_l)$, які можна перетворити в апостеріорні за відомою 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 відповідною теоремою: $p(\gamma_l) = \sum_{m=1}^{M} p(\mu_m)\, p(\gamma_l / \mu_m)$.

Застосування класифікатора Байеса, на практиці, є ускладненим, оскільки для прикладних задач вигляд функцій щільності рідко буває відомий заздалегідь. Для подолання цієї проблеми існує два підходи:

- *Параметричний.* Функції щільності подаються у аналітичному вигляді, але це не завжди можливо зробити;
- *Не параметричний*. Замість, отримання апріорної інформації про вигляд функції щільності ймовірностей, достатньо розрахувати оцінки основних її величин через методи: математичної статистики, парзенівських вікон, ядра щільності, найближчих сусідів, тощо.

Етап навчання, для статистичної теорії розпізнавання, характеризується значною обчислювальною трудомісткістю, яка пов’язана з необхідністю виконання умов збіжності алгоритму. Оперативність роботи системи, на основі відомих модифікацій стохастичної апроксимації, наприклад градієнтних методів, коливається в значних межах [8].

Головною метою статистичного підходу можна вважати знаходження роздільної функції, яка задає спосіб розбиття на класи розпізнавання. Таким чином, задача навчання трактується як пошук екстремального значення показника ефективності, в якості якого розглядається мінімум середнього ризику помилкової класифікації.

Статистичні методи мають низьку достовірність класифікації під час екзамену, це пов’язано з невизначеністю або відсутністю апріорної інформації про об’єкт дослідження. Тобто найкращого результату вдасться досягти в тих галузях людської діяльності, де можливо створити набір елементів для репрезентативної вибірки. До того ж вкрай важливо, щоб обрані експериментальні дані були однорідними та стійкими.

Останнім часом значного поширення, для задач адаптивного пошуку та багатопараметричної оптимізації, набули методи, які відносяться до класу генетичних алгоритмів (ГА) [17]. Їх суть полягає у копіюванні біологічних механізмів спадковості та природного відбору, для здійснення випадкового пошуку глобального екстремуму. Дуже часто генетичні алгоритми поєднуються з іншими методами, що дозволяє спростити побудову оптимального класифікатору.

На практиці, у системах розпізнавання зображень, найчастіше використовуються екстремально-кореляційні СР. Зробимо огляд окремих методів такого типу:

1) Ванільний градієнтний підйом (Vanilla Gradient Ascent).

За допомогою модифікованої версії зворотного розповсюдження помилки дозволяє обчислити як на результат ідентифікації впливає одночасно кожен окремий піксель [10].

Градієнт – це вектор, який може відображати наскільки зміна ваги одного елементу впливає на кінцевий результат. Це означає, що він показує які компоненти об’єкту є найбільш важливими для його ідентифікації. Під час реалізації класичного зворотного розповсюдження помилки приймається від’ємне значення градієнту і мінімізуються втрати на етапі навчання. Замість

цього градієнтний підйом використовує відношення оцінки класу до вхідних пікселів, помічаючи найбільш важливі для класифікації зображення. Таким чином, лише за допомогою одного проходу можна створити теплову карту, яка покаже нам пріоритетність кожного окремого пікселю. (Рис.1.13).

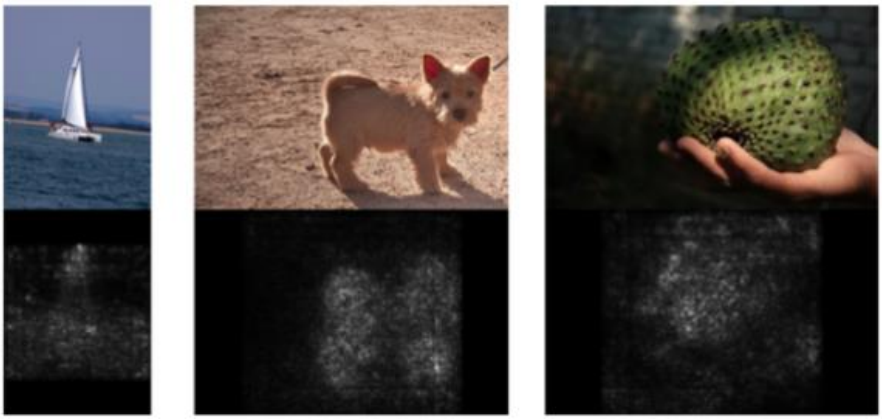


Рисунок 1.13 – Приклад результату роботи градієнтного підйому

Незважаючи на працездатність алгоритму, він має один суттєвий недолік: розповсюдження від'ємного градієнту, який створює шум на результуючому зображенні (Рис.1.14).

а *б*

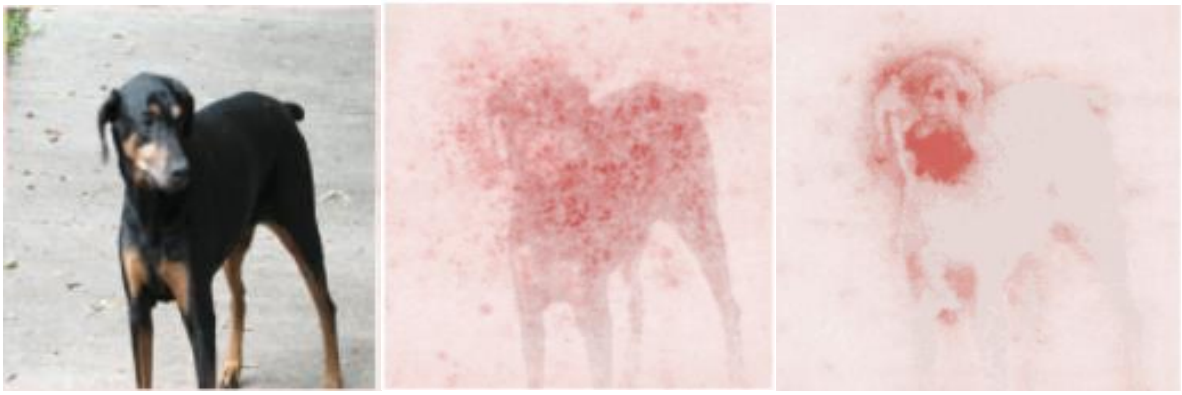
Рисунок 1.14 – Процес оптимізації вхідного математичного опису:
а - початкове зображення; б – результат градієнтного підйому

2) Інтегровані градієнти (Integrated Gradients).

У роботі [11] автори зосереджують свою увагу на теорії інтерпретації образів, зокрема на дві аксіоми про чутливість та інваріантність реалізацій, яким, на їх погляд, повинний відповідати будь-який метод розпізнавання.

Метод інтегрованих градієнтів (ІГ) починається з базового зображення, як правило, воно повністю або частково затемнене. Після цього поступово яскравість збільшується. Градієнти оцінок класів по відношенню до вхідних елементів обчислюється окремо для кожного пікселю і усереднюються. Таким чином, вдається отримати значення цінності для кожного пікселя.

Слід зазначити, що в такий спосіб можна подолати проблему ванільного градієнтного підйому [10]. Річ у тім, що градієнти є локальними, вони не відображають глобальну цінність пікселя, лише його чутливість в конкретній точці входу. Змінюючи яскравість зображення і обчислюючи градієнти на кожному кроці можна отримати більш повну картину цінності кожного окремого пікселя (Рис.1.15).



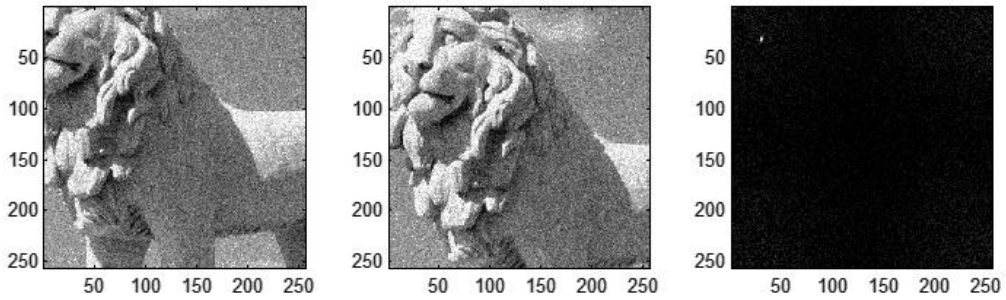
а *б* *в*

Рисунок 1.15 – Порівняння алгоритмів: а - базове зображення; б – результат ванільного підйому; в – результат інтегрованих градієнтів

Незважаючи на те, що здебільшого при використанні ІГ можна отримати більш точні карти чутливості пікселів, цей метод працює досить повільно. Крім цього, він вводить два нових параметра: обрання базового зображення та кількість кроків, за якими створюються інтегровані градієнти. Від їх вибору буде залежати ефективність та тривалість алгоритму.

3) Алгоритм фазової кореляції. Метод базується на припущенні, що зображення спотворено гаусівським білим шумом, який при зіставлені з самим собою (від одного зображення до іншого) дасть помітний пік у області, де зсув майже нульовий.

Припустимо, що у нас э два зображення рис.1.16.а і рис.1.16.б з гаусівським шумом і відносним зміщенням між ними в (30,33) пікселя. У результаті фазової кореляції було отримано чіткий пік рис.17.г приблизно у координатах (30,33).



а *б* *в*

Рисунок 1.16 – Процес фазової кореляції: а - базове зображення; б – зміщене зображення; в – фазова кореляція

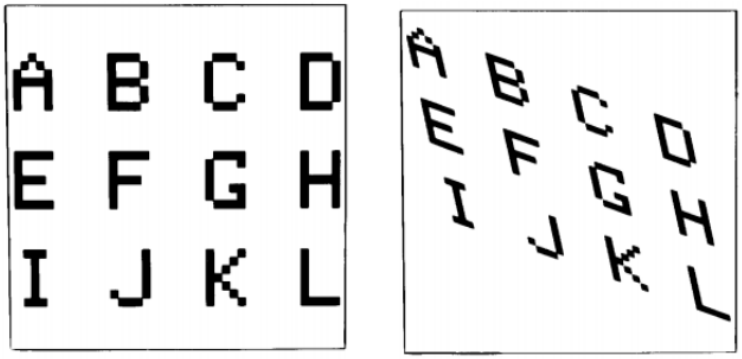
Отже, метод зводиться до обчислення кореляції за фазовими складовими спектрів зображень. Це робиться тому, що фаза спектру несе більше корисної інформацію про сигнал порівняно із амплітудою. Алгоритм показує найбільшу ефективність при наявності вузькополосного шуму.

4) Метод інваріантних моментів.

Основною перевагою цього методу є його нечутливість до положення об’єкту в кадрі, що робить його вкрай ефективним при розпізнаванні образу з довільною орієнтацією. Проте алгоритм відзначається надзвичайною обчислювальною трудомісткістю, особливо, коли на зображенні присутні усі типи афінних перетворень (зсув, поворот, масштабування).

Інваріант – це величина, яка залишається не зміною після деякого її перетворення. Під час екзамену образи можуть відрізнятися від свого еталону масштабом, поворотом або зсувом. Для об’єктів, які відносяться до одного класу, можна стверджувати, що при послідовному виконанні афінних перетворень і порівняні результату з еталоном, вдасться отримати параметри, які встановлять найвищу степінь подібності між зображеннями.

Наприклад на рисунку 1.17 можна помітити, що після виконання деяких геометричних перетворень нам вдасться із деформованого зображення рис. 1.17.б отримати його початковий вигляд 1.17.а.



а *б*

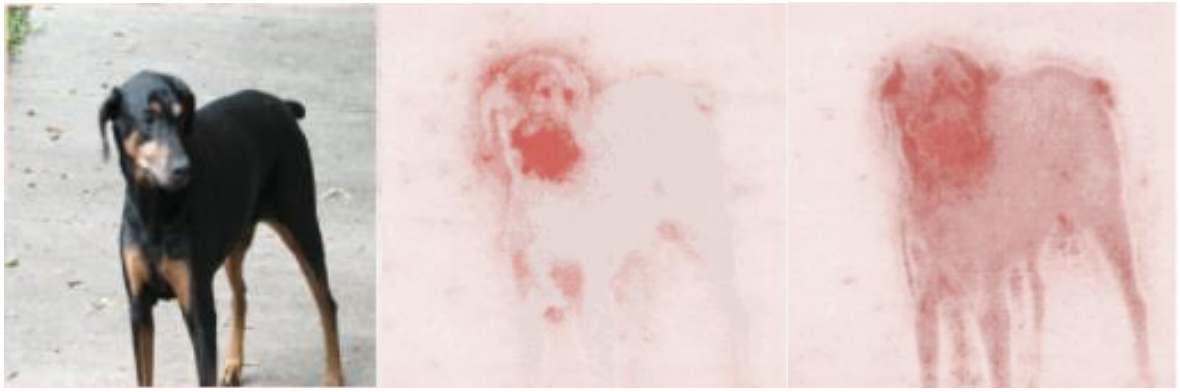
Рисунок 1.17 – Приклад інваріантів: а - початкове зображення; б – моментні інваріанти

Тобто суть алгоритму полягає у визначенні кореляції між статистичними моментами функції розподілу зображень, що порівнюються.

4) Розмиті інтегровані градієнти (Blur Integrated Gradients).

Є одним із найновіших градієнтних методів, який вдалося знайти у вільній літературі [12]. Він представляє із себе покращений варіант методу, який був описаний у роботі [11].

Суть алгоритму полягає у поступовому розмиті початкового зображення, з обчисленням градієнтів для кожного, та усередненням кінцевого результату (Рис.1.18).



а *б* *в*

Рисунок 1.18 – а - базове зображення; б – результат для інтегрованих градієнтів; в – результат при розмитих інтегрованих градієнтах

Хоча процес не сильно відрізняється від того, що описаний для методу інтегрованих градієнтів [11], де зображення було повністю затемнене а потім поступово поверталося до свого початкового вигляду, результат позбувається ряду візуальних артефактів. Автори стверджують, що такий підхід є більш теоретично виправдний, бо розмиття не призводить до появи нових дефектів.

6) Ієрархічно-кореляційні алгоритми кластеризації.

Сукупність методів, які, за певним правилом, упорядковують дані у деяку структуру з вкладених кластерів [9]. Алгоритми цього типу прийнято поділяти на два класи:

- *Дивезійні методи*, де нові кластери створюються шляхом ділення більших груп на менші;
- *Агломеративні методи*, де менші групи об’єднуються в більші;

Алгоритми ієрархічної кластеризації доцільно застосовувати до даних, які характеризуються деякою зв’язністю.

Сучасними науковцями активно розвиваються штучні нейроні мережі. Їх слід віднести до окремої групи методів розпізнавання, які намагаються програмно відтворити нейронні структури людського мозку. Нейрони реалізуються у вигляді перимикачів з різною вагою, вони поєднуються таким чином, щоб створити нерозривну систему. Навчання відбувається через повторну активацію деяких з’єднань. В результаті, збільшується вірогідність правильного результату при відповідній вхідній інформації. Нажаль цей підхід має ряд недоліків, які проявляються підчас обробленні великих масивів даних, які сформовані при довільних умовах.

Окрім зазначених способів реалізації СР є ще досить специфічні. Наприклад, до них можна віднести методи нечіткої класифікації, які засновані на теорії нечітких множин Л.Заде. Науковцям вдалося навіть досягти деяких успіхів у цьому напрямку, особливо у розпізнаванні літер і цифр, але цей підхід ще потребує доопрацювання.

Не зважаючи на активний розвиток галузі автоматичної класифікації, значних зрушень у підвищенні ефективності навчання СР, для слабо

формалізованих задач, все ще не відбулося. До сих пір відсутні універсальні методи ідентифікації об'єктів, які б наблизилися до інтелектуальних можливостей людини. Сьогодні можливо 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 [13];

1.4. Формалізована постановка задачі інформаційного синтезу бортової системи розпізнавання

Нехай сформовано алфавіт $\{X_m^o \mid m=\overline{1,M}\}$ класів розпізнавання, які характеризують кадри зображення місцевості. Для кожного із них 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 виконується перетворення вхідної навчальної матриці Y в робочу бінарну X, яка змінюється в процесі машинного навчання. Тому для простору Хеммінга задано вектор параметрів функціонування, які впливають на ефективність машинного навчання БСР розпізнавати реалізації класу X_m^o:

$$g_m =< x_m, d_m, \delta >, \tag{1.1}$$

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 дорівнює половині симетричного поля

контрольних допусків на ознаки розпізнавання, якими є значення яскравості в пікселях.

На параметри функціонування системи, які будемо далі називати 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 перевірити функціональну ефективність машинного навчання. Тобто потрібно оптимізувати відповідні параметри (1.1), які забезпечують максимальне 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’язання поставленої задачі слід виконати такі завдання:

- розробити математичні моделі функціонування системи розпізнавання в режимах машинного навчання та екзамену;

- розробити алгоритми машинного навчання СР образів з оптимізацією системи контрольних допусків на ознаки розпізнавання;
- розробити інформаційний критерій і, за допомогою нього, оцінити функціональну ефективність машинного навчання системи;
- побудувати вирішальні правила;
- розробити алгоритм функціонування системи в режимі екзамену і оцінити його функціональну ефективність;

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

2. ОПИС МЕТОДУ ДОСЛІДЖЕННЯ

2.1. Основні положення інформаційно екстремальної технології аналізу даних

Основна ідея машинного навчання системи розпізнавання, у рамках IEI-технології, згідно до праць [13][14], полягає в побудові, в рамках геометричного підходу, високо достовірних вирішальних правил шляхом оптимізації 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. Шляхом оптимізації контрольних допусків на ознаки розпізнавання здійснюється перетворення вхідної навчальної матриці в робочу бінарну, яка в процесі машинного навчання змінюється з метою адаптації вхідного математичного опису до максимальної достовірності класифікаційних рішень.

Побудова високо достовірних вирішальних правил, в рамках IEI-технології, здійснюється за багатоциклічною процедурою пошуку максимального граничного, усередненого за алфавітом $\{X_m^o\}$, 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 параметрів машинного навчання системи розпізнавати реалізації класу X_m^o; G_ξ – допустима область значень ξ -ї ознаки розпізнавання; G_E – допустима область визначення функції інформаційного критерію.

На алгоритм навчання (2.1) накладаються такі 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 $Card\,\tilde{\Re} = M$;

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

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

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 визначається розмірністю вектору параметрів навчання. При цьому, внутрішній цикл реалізує, так званий, базовий алгоритм, призначенням якого є:

1) обчислення на кожному кроці навчання інформаційного критерію і пошук глобального максимуму його функції;

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 максимального граничного значення.

2.2. Математичні моделі машинного навчання

Розглянемо категорійну модель машинного навчання розробленої СР у вигляді орієнтованого графа, де задіяні множини, що відображаються одна на одну відповідними операторами.

Категорійна модель включає вхідний математичний опис системи розпізнавання, який подано як структуру (2.7):

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

де G – 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, яка 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.

При цьому, декартів добуток $G \times T \times \Omega \times Z$ утворює універсум випробувань, який є джерелом інформації для формування вхідної навчальної матриці Y. Категорійну модель інформаційно-екстремального машинного навчання БСР з оптимізацією відповідних параметрів – координат структурованого вектору (1.1) показано на рисунку 2.1.

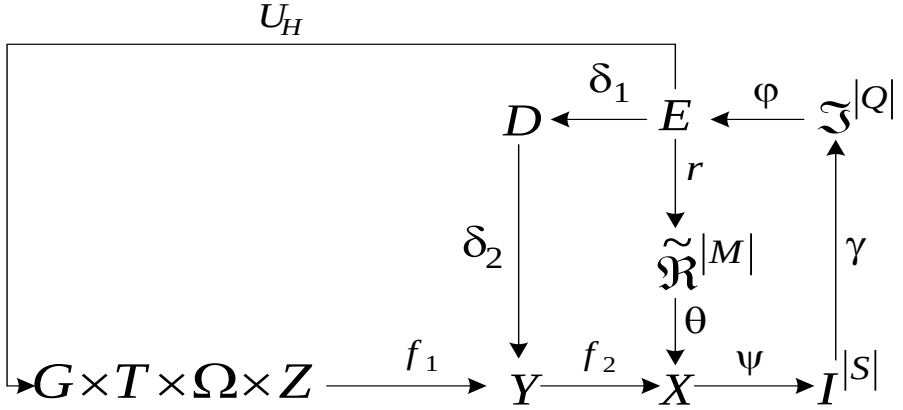


Рисунок 2.1 – Категорійна модель машинного навчання СР

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, перевіряє основну статистичну гіпотезу $\gamma_1: x_{m,i}^{(j)} \in X_m^o$. Оператор γ визначає множину $\Im^{|Q|}$ точнісних характеристик класифікаційних рішень, де $Q = S^2$, а оператор φ обчислює множину значень E інформаційного критерію оптимізації, який є функціоналом від точнісних характеристик. Контур оптимізації контрольних допусків на ознаки розпізнавання замикається через терм-множину D, елементами якої є значення системи контрольних допусків (СКД) на ознаки розпізнавання. Оператор u регламентує процес машинного навчання.

Перевірка функціональної ефективності, інформаційно-екстремального машинного навчання, здійснюється при функціонуванні системи розпізнавання образів в режимі екзамену.

Категорійна модель подана у вигляді орієнтованого графа відображень множин, які застосовуються при функціонуванні розробленної системи в режимі класифікації об’єкту, показано на рис. 2.2.

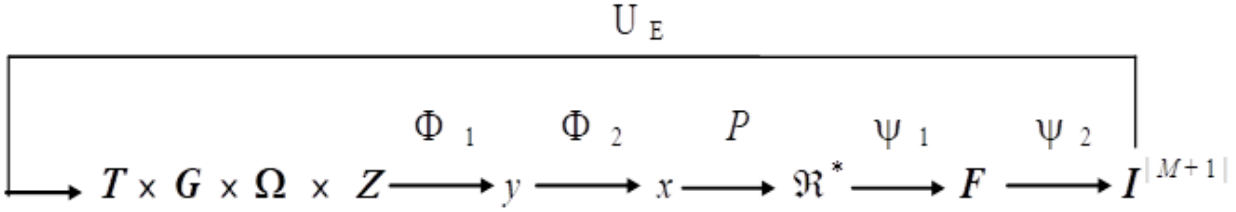


Рисунок 2.2 – Категорійна модель СР в режимі екзамену

У категорійній моделі (рис. 2.2) оператор Φ_1, із джерела інформації, формує екзаменаційну реалізацію класу, що розпізнається, аналогічну, за структурою, навчальній матриці. Оператор Φ_2, за допомогою отриманих на етапі машинного навчання оптимальними контрольними допусками на ознаки

розпізнавання, формує двійкову реалізацію x, а оператор P відображає вектор-реалізацію, що розпізнається, на побудоване, під час етапу машинного навчання, оптимальне розбиття $\Re^*$ класів розпізнавання. Оператор Ψ_1 для кожної реалізації обчислює значення, побудованих на етапі машинного навчання, вирішальних правил і формує терм-множину F, а оператор Ψ_2 за максимальним значенням вирішального правила відносить вектор, що розпізнається, до одного із класів заданого алфавіту $\{X_m^o\}$. Призначенням оператора U_E є регламентація процесу екзамену.

Розглянуті категорійні моделі відбивають притаманні людині перетворення інформації, які мають місце при когнітивних процесах формування та прийняття 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

Для оцінки функціональної ефективності машинного навчання системи розпізнавання, що навчається, найчастіше використовуються ентропійний критерій Шеннона та інформаційна міра Кульбака. У праці [13] розглядається критерій Кульбака як добуток логарифмічного відношення обчислених на k-му кроці навчання повної ймовірності $P_{t,m}^{(k)}$ правильного прийняття рішень про належність класу X_m^o реалізації, що розпізнається, до повної ймовірності $P_{f,m}^{(k)}$ помилкового прийняття рішень і різниці цих імовірностей:

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

Згідно з теоремою про повну ймовірність для двохальтернативної системи оцінок рішень при допущенні, згідно із принципом 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, яка визначає радіуси гіперсферичних контейнерів, побудованих в радіальному базисі Хеммінга.

Після підстановки виразу (2.9) в формулу (2.8) отримаємо

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

Виразимо першу і другу достовірності:

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

Після підстановки виразів (2.11) в формулу (2.10) отримаємо:

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

Нормована модифікація критерію (2.12) має 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).

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

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

де $K_{1,m}^{(k)}$ – кількість подій, при яких реалізації, що належать класу X_m^o, помилково до нього не відносяться; $K_{3,m}^{(k)}$ – кількість подій, при яких помилково відносяться, до класу розпізнавання X_m^o, реалізації сусіда X_c^o; $n_{\min}$ – мінімальний обсяг навчальної вибірки, який визначається згідно з працею [13].

Після підстановки відповідних оцінок точнісних характеристик (2.14) у вираз (2.12) одержимо робочу формулу для обчислення критерію Кульбака:

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

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

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

Обчислення коефіцієнтів $K_{1,m}^{(k)}$ і $K_{2,m}^{(k)}$, при класифікації j-ої реалізації, здійснювалося 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 здійснюється за правилом:

1) обчислюється кодова відстань $d[x_m \oplus x_c^{(j)}]$;

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 точнісних характеристик класифікаційних рішень, які в свою чергу залежать від дистанційних параметрів. Тобто, вище наведені модифікації критерію Кульбака можна розглядати як узагальнення відомих статистичних і дистанційних критеріїв близькості класів розпізнавання.

2.4. Лінійний алгоритм машинного навчання інформаційно екстремальної технології для системи розпізнавання

Базовий алгоритм, як правило, не забезпечує високої функціональної ефективності машинного навчання, оскільки контрольні допуски можуть бути не оптимальними в інформаційному розумінні. Згідно з категорійною моделю (Рис. 2.1) інформаційно-екстремальний алгоритм машинного навчання системи розпізнавання з оптимізацією системи контрольних допусків (2.16) подано у вигляді ітераційної процедури пошуку глобального максимуму усередненого за алфавітом класів розпізнавання інформаційного критерію в робочій (допустимій) області визначення його 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 показано двобічне симетричне поле контрольних допусків.

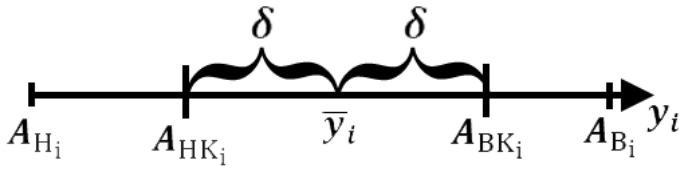


Рисунок 2.3 – Поле контрольних допусків на ознаку розпізнавання

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

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

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

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

2) $m : m + 1$;

3) онулення лічильника зміни параметра δ поля контрольних допусків: $\delta := 0$;

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

5) обчислюються нижні $A_{H,i}$ і верхні $A_{B,i}$ контрольні допуски на ознаки розпізнавання, відповідно до правил:

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

6) онулення лічильника кроків зміни радіуса гіперсферичного контейнера: $k := 0$;

7) $k := k = 1$;

8) формується тривимірний масив бінарної навчальної матриці X, елементи якої обчислюються за правилом:

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

9) формування масиву усереднених двійкових векторів-реалізацій $\{x_m\}$, елементи яких визначаються за правилом

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

де ρ_m – рівень квантування координат двійкового вектору x_m, який за замовчуванням дорівнює 0,5.

10) розбиття 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) обчислюється інформаційний критерій;

12) якщо $k \le N$, то виконується пункт 7, інакше – пункт 13;

13) якщо $\delta < \delta_H$, то виконується пункт 4, інакше – пункт 14;

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

15) якщо виконується умова $m < M - 1$, то реалізується пункт 2, інакше – пункт 16;

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

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

Отримані в процесі паралельної оптимізації екстремальні значення параметрів машинного навчання є квазіоптимальними, оскільки вони змінювалися, при кожному кроці навчання, на однакову величину для всіх ознак одночасно. Для підвищення функціональної ефективності СР було реалізовано алгоритм машинного навчання з послідовною оптимізацією контрольних допусків. При цьому, отримані на етапі паралельної оптимізації контрольні допуски приймалися як стартові при послідовній оптимізації, яка здійснювалася 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 стартовими величинами для всіх ознак; $\otimes$ – символ операції повторення;

Машинне навчання СР, з паралельно-послідовною оптимізацією контрольних допусків, дозволяє підвищити достовірність класифікаційних рішень і при цьому суттєво підвищується оперативність машинного навчання, оскільки пошук глобального максимуму критерію здійснюється тільки в робочій області визначення його функції.

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

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

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

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

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

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

Таким чином, на екзамені визначається, за вирішальними правилами (2.18), належність реалізації класу, що розпізнається, одному із класів, які задані алфавітом. При цьому, вирішальні правила, через малу обчислювальну трудомісткість, відрізняються високою оперативністю.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ НАЗЕМНИХ ОБ'ЄКТІВ

3.1 Вхідний математичний опис бортової системи розпізнавання

На рисунку 3.1 показана 3D-модель місцевості, розміром 1846 на 1046 пікселів, за якою буде здійснюватися подальше розпізнавання об'єктів. Слід зазначити, що змодельований план регіону містить типові зони інтересу для безпілотника і максимально наближається до реальних умов.

Рисунок 3.1 – 3D-модель місцевості

Вхідний математичний опис БСР починається із формування алфавіту класів розпізнавання, для якого слід обрати зображення об'єктів, які система повинна ідентифікувати. У результаті аналізу рисунку 3.1 було виділено чотири класи місцевості: X_1^0 - техногенні об'єкти; X_2^0 - сільська дорога; X_3^0 - рідкий ліс; X_4^0 - ґрунт;

На рисунку 3.2 показано кадри ділянок, розміром 55 на 55 пікселів, які були обрані як класи розпізнавання місцевості (рис.3.1).

а *б* *в* *г*

Рисунок 3.2. Класи розпізнавання: а – клас X_1^0; б – клас X_2^0; в – клас X_3^0; г – клас X_4^0;

Серед обраних, на рисунку 3.2, зображень рис.3.2.г – відноситься до типу «текстура», тобто це стаціонарний клас, де невелика частина зображення буде містити достатньо інформації для ідентифікації. Проте інші кадри є не стаціонарними за яскравістю і на етапі екзамену об'єкти, які вони містять, можуть займати довільне положення.

Враховуючи концепції IEI-технології, яка була обрана як метод дослідження, сформуємо зміст вхідного математичного опису БСР:

1) словник ознак розпізнавання;

2) алфавіт класів розпізнавання;

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

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

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

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

3.2 Визначення базового класу

Під час машинного навчання, згідно принципів IEI-технології, формується система контрольних допусків. При цьому обирається базовий

клас, на основі якого буде створюватися СКД. Від цього етапу залежить подальша точність створених вирішальних правил. Оптимальна, в інформаційному розумінні, СКД повинна описувати таку кількість унікальних ознак обраних класів, щоб мінімізувати їх перетин. Було висунуто припущення, що доцільно обирати базовий клас за дисперсією. Для цього знаходимо усереднене значення яскравості кожної ознаки алфавіту класів $\{X_m^o \mid m=\overline{1,M}\}$ за формулою (3.1):

$$\Theta_j = \frac{1}{N}\sum_{i=1}^{N}\theta_i , \tag{3.1}$$

де Θ_j – усереднене значення яскравості пікселів, що входять до *j*-ї ознаки**;** θ_i – значення яскравості RGB-складової в *i*-му пікселі рецепторного поля зображення кадру; *N* – кількість ознак;

Дисперсію шукаємо як міру відхилення кожної окремої ознаки від усередненого значення яскравості $\overline{\Theta}$, за формулою (3.2):

$$S_m = \frac{1}{N-1}\sum_{i=1}^{N}(\Theta_i - \overline{\Theta})^2 , \tag{3.2}$$

де S_m – значення дисперсії для класу *m*, $m=\overline{1,M}$; *N* – кількість ознак; Θ_i – усереднене значення яскравості пікселів, що входять до *i*-ї ознаки; $\overline{\Theta}$ - усереднене значення Θ_i;

Для прикладу візьмемо вже обрані нами класи (рис.3.2) та усереднимо значення яскравості їх пікселів, що входять до однієї ознаки, рисунок 3.3.

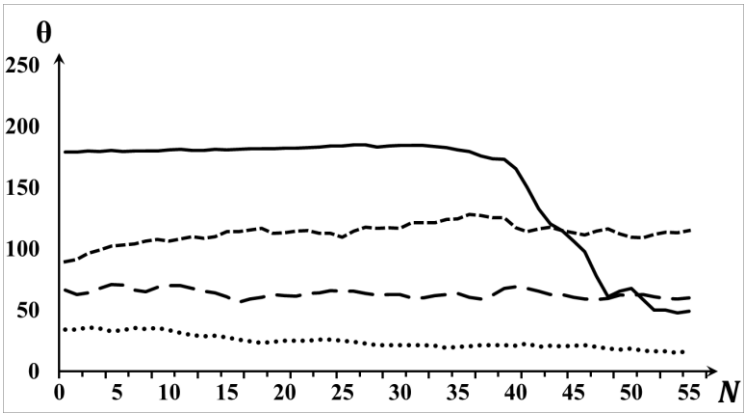
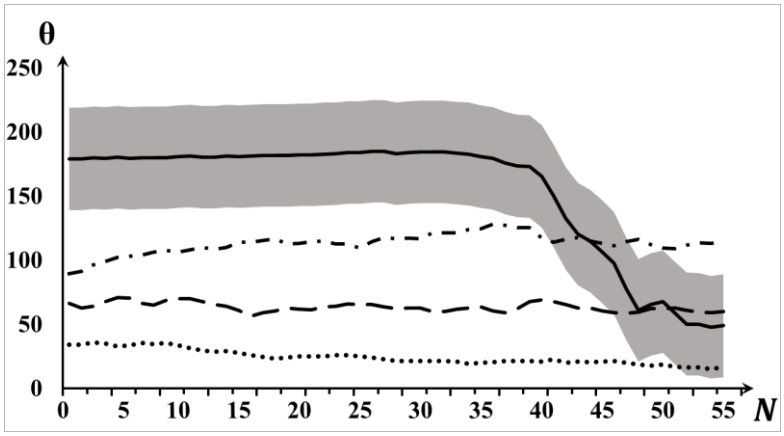


Рисунок 3.3 – Графік усередненого значення яскравості ознак:
суцільна лінія – X_1^0; штрих-пунктир – X_2^0; пуктир- X_3^0; штрихована лінія - X_4^0;

При цьому якщо обчислити дисперсії для кожного із класів, то отримуємо наступні значення: $S_1 = 2191.47$ для класу X_1^0; $S_2 = 61.89$ для класу X_2^0; $S_3 = 33.29$ для класу X_3^0; $S_4 = 11.6$ для класу X_4^0; Обираємо клас із максимальною дисперсією, в даному випадку це X_1^0, і будуємо СКД при значенні $\delta = 40$. Візуально це демонструє рисунок 3.4. на якому зафарбована зона і є система контрольних допусків.



Графік 3.4 – Графік СКД на основі класу з найбільшою дисперсією

Аналіз рис.3.4 показує, що обраний клас є оптимальним, бо побудована на основі нього СКД описує максимальну кількість ознак алфавіту класів.

Гіпотеза була експерементально підтверджена при реалізації базового алгоритму інформаційно-екстримального машинного навчання, в процеці якого було побудовано СКД на основі кожного із класів розпізнавання при визначеному значенні параметра поля контрольних допусків δ=40. Максимальне значення критерію (2.15) було отримано для класу X_1^0, тобто він є найближчим сусідом для будь-якого елементу алфавіту і водночас має найбільшу дисперсію яскравості ознак.

3.3 Результати моделювання

Для реалізації інформаційно-екстремального машинного навчання бортової системи розпізнавання, шляхом оброблення кадрів зображення місцевості в декартовій системі координат, була сформована вхідна навчальна

матриця. Як критерій оптимізації було обрано модифіковану міру Кульбака в своїй нормованій формі (2.13).

На рисунку 3.5 показано графік залежності усередненого, за алфавітом класів розпізнавання, нормованого критерію (2.13) від параметра δ поля контрольних допусків, отриманих при їх паралельній оптимізації. Тут і далі робочу область визначення функції критерію виділено темним кольором.

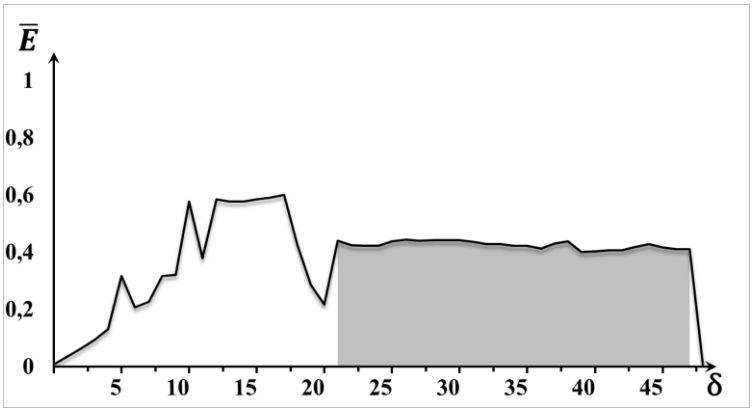


Рисунок 3.5 – Графік залежності інформаційного критерію (2.13) від параметра поля контрольних допусків на ознаки розпізнавання

Аналіз рисунку 3.5 показує, що оптимальне значення параметра поля контрольних допусків дорівнює $\delta^* = 27$ (в градаціях яскравості) при максимальному значенні інформаційного критерію $\bar{E}^* = 0.45$. Для підвищення функціональної ефективності системи розпізнавання додатково було реалізовано алгоритм послідовної оптимізації (2.17). На рисунку 3.6 показано зміну нормованого критерію (2.13) в процесі послідовної оптимізації контрольних допусків на ознаки розпізнавання.

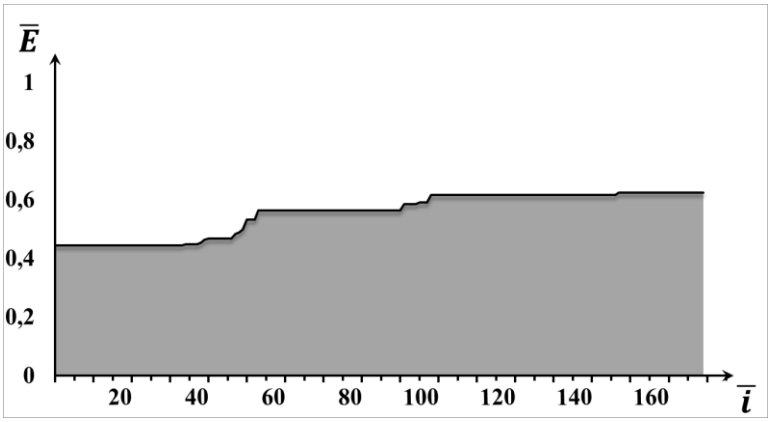
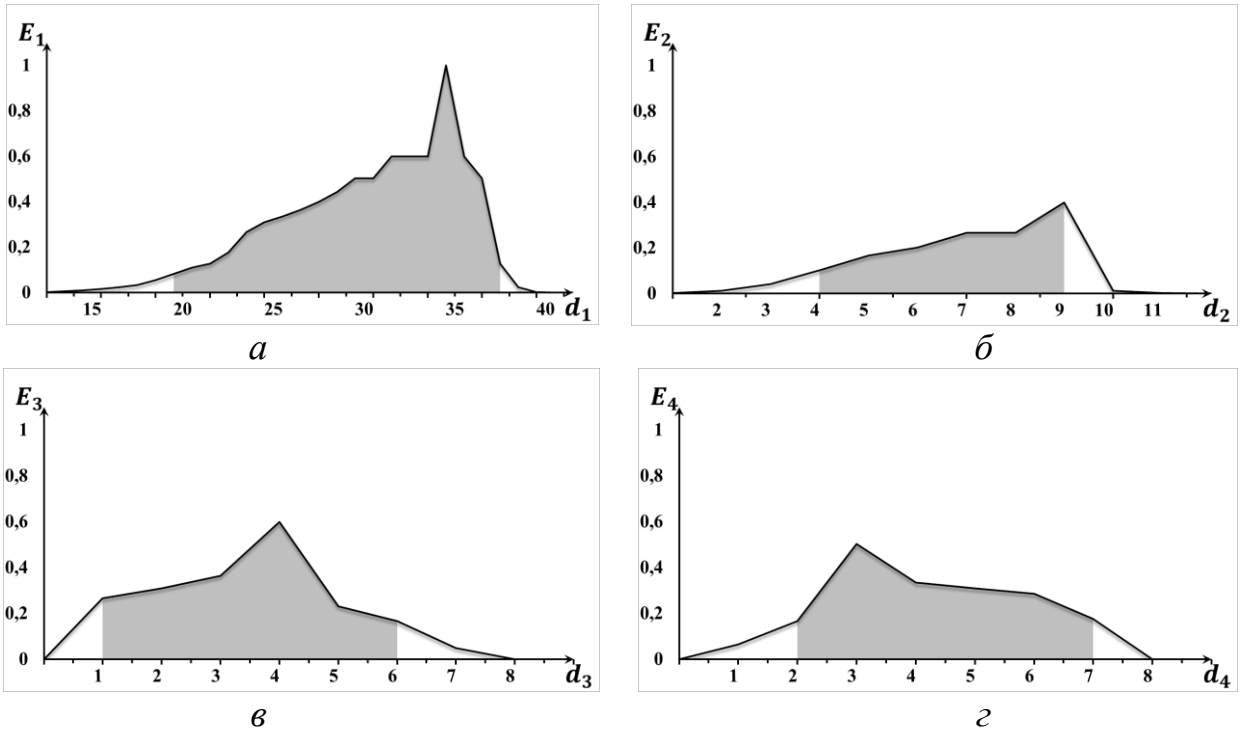


Рисунок 3.6 – Графік зміни інформаційного критерію в процесі послідовної оптимізації контрольних допусків на ознаки розпізнавання

Аналіз рисунку 3.6 показує, що інформаційний критерій оптимізації на третьому прогоні, який визначається відношенням кількості ітерацій $\bar{i}$ до кількості ознак *N*, досягнув значення $\bar{E}^* = 0.63$.

На рисунку 3.7 показано графіки залежності інформаційного критерію (2.13) від радіусів контейнерів розпізнавання, оптимальні значення яких дозволяють побудувати вирішальні правила (2.18).



а *б*

в *г*

Рисунок 3.7 – Графіки залежності критерію (2.13) від радіусів контейнерів класів розпізнавання: а – клас X_1^o; б – клас X_2^o; в – клас X_3^o; г – клас X_4^o

Аналіз рисунку 3.7 показує, що оптимальні значення радіусів контейнерів класів розпізнавання дорівнюють: $d_1^* = 34$ (тут і далі в кодових одиницях) – для класу X_1^0; $d_2^* = 9$ – для класу X_2^0; $d_3^* = 4$ – для класу X_3^0; $d_4^* = 3$ – для класу X_4^0;

Перевірка ефективності машинного навчання бортової системи розпізнавання здійснювалось в режимі екзамену, за результатами якого було отримане оцифроване зображення місцевості, рисунок 3.8.

Рисунок 3.8 – Результат класифікації кадрів

Візуальний огляд отриманого, на рисунку 3.8, результату показує, що ідентифікація є достовірною, адже у більшості випадках кадри правильно класифікуються.

Через те, що вирішальні правила є не безпомилковими за навчальною матрицею, тобто не вдалося досягти максимально граничного значення інформаційного критерію, необхідно збільшити глибину машинного навчання шляхом оптимізації додаткових параметрів, включаючи розмір кадру зображення [15].

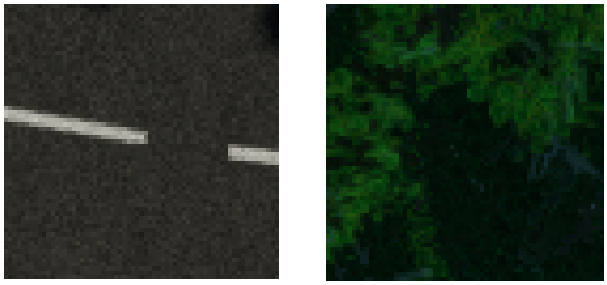
Оскільки об’єкти, які бортова система розпізнавання повинна ідентифікувати, зазвичай, є не стаціонарними і можуть приймати довільне положення у кадрі, то доцільно виконувати оброблення зображень в полярній системі координат. Як показано в праці [16] це дозволить створити інваріантні вирішальні правила, які будуть менш чутливі до положення об’єкту.

3.4 Результати динамічного випробування

Для того, щоб наблизитися до реальних умов, був розроблений алгоритм, який дозволяє ідентифікувати фрейми відео-потоку, отриманого з

камери безпілотника. Додатков було змодельовано окрему ділянку місцевості, яка представляє із себе автомагістраль.

Ідентифікація буде проводитися у спрощеному вигляді, через бінеризацію карти на кадри, що відносяться до класу автомобільна дорога (рис.3.9.а) і не належать йому (рис.3.9.б).



а *б*

Рисунок 3.9. а – клас X_1^0; б – клас X_2^0;

На рисунку 3.10 можна побачити графік залежності інформаційного критерію (2.13) від параметра δ поля контрольних допусків, отриманого під час їх паралельної оптимізації.

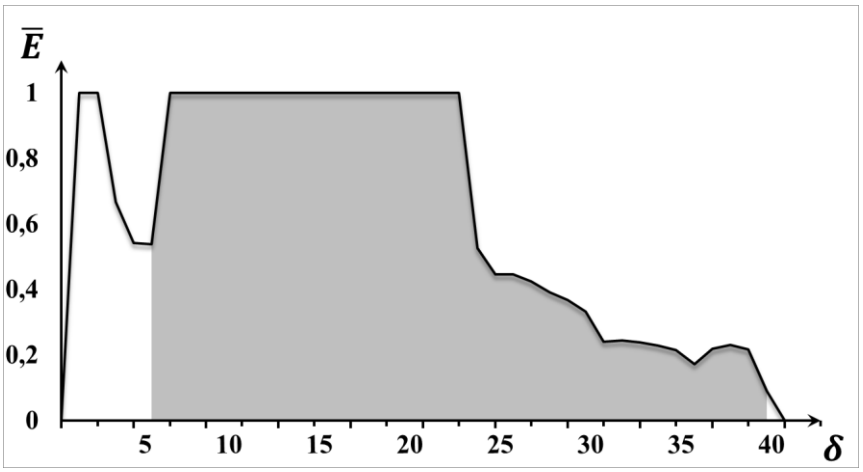
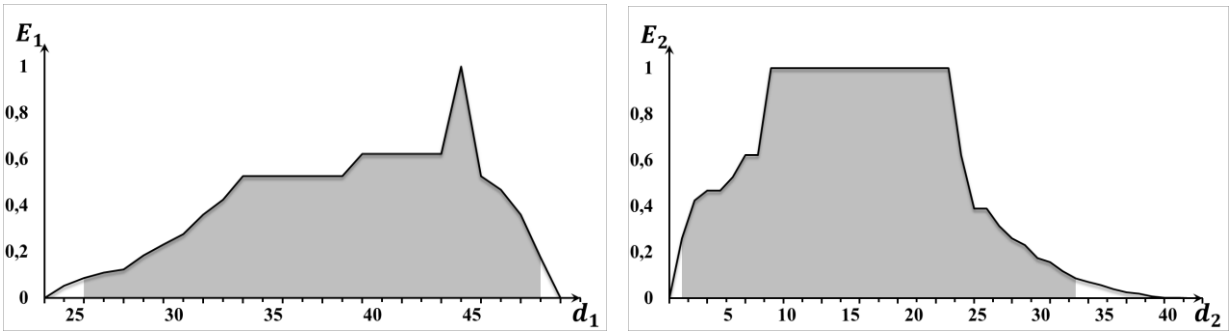


Рисунок 3.10 – Графік залежності інформаційного критерію (2.13) від параметра поля контрольних допусків на ознаки розпізнавання

Як можна побачити з графіку (рис.3.10) на етапі навчання було досягнуто максимально граничного значення інформаційного критерію (2.13), тобто, як мінімум, за навчальною матрицею побудовані вирішальні правила є без помилковими. Це говорить про те, що нема необхідності у послідовній оптимізації контрольних допусків.

Залежність інформаційного критерію (2.13) від радіусів контейнерів розпізнавання, можна побачити на рисунку 3.11.



а *б*

Рисунок 3.11 – Залежності критерію (2.13) від радіусів контейнерів класів розпізнавання: а – клас X_1^o; б – клас X_2^o;

Згідно рисунку 3.11 оптимальними є наступні значення радіусів контейнерів розпізнавання: $d_1^* = 44$ – для класу X_1^0; $d_3^* = 8$ – для класу X_2^0;

Етап екзамену проводився шляхом розбиття відео-потоку на фрейми з їх подальшою класифікацією. Приклад результату показано на рисунку 3.12.

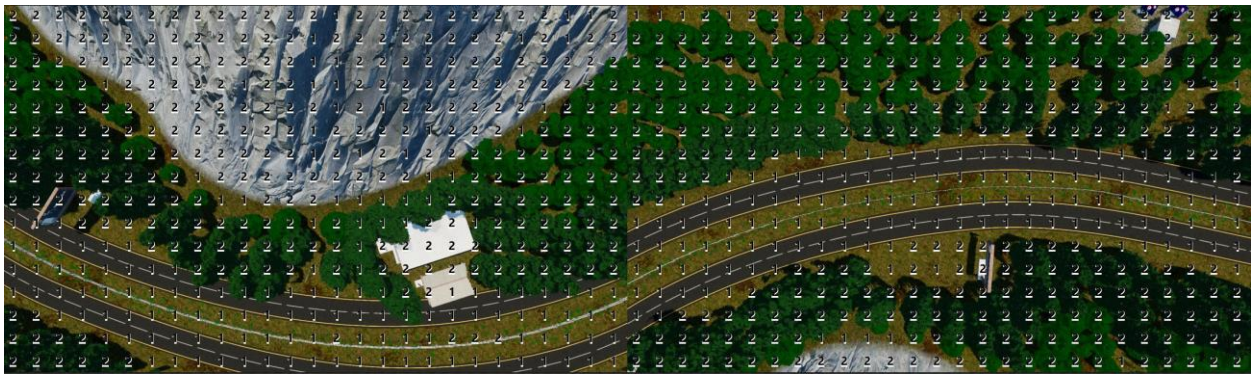


Рисунок 3.12 – Результат класифікації відео-потоку

Слід зазначити, що на рисунку 3.12 показано ідентифікацію лише двох фреймів, на практиці їх кількість обмежується лише обсягом пам’яті БСР. Окремо був розроблений програмний модуль, який склеює отримані результати в одне відео, щоб спростити візуальний аналіз результату класифікації.

3.5 Короткий опис програмної реалізації

Бортова система розпізнавання була написана на мові програмування *C#*. Для зручності сприйняття опис коду, розробленого програмного забезпечення, занесений до таблиць і в Додаток.

Клас decartCoord містить в собі інформацію про методи обробки та взаємодії з класами, що були обрані для машинного навчання. Його опис представлений в табл. 3.1.

Таблиця 3.1 Опис складових класу decartCoord

ЗМІНІ			
Назва	Тип	Модифікатор доступу	Призначення
img	Bitmap	private	Зберігає клас у вигляді зображення
bitMap	int[,]	private	Зберігає клас у вигляді матриці яскравості пікселів
`binaryMap`	`int[,]`	private	Зберігає бінарну матрицю класу
`etalonVektor`	`int[]`	private	Зберігає еталонний вектор класу
`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 колонок зображення (N)
`public int getRows()`			Повертає кількість строк зображення (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 виконує безпосереднє навчання в рамках інформаційно-екстремальної інтелектуальної технології. Його опис представлений в табл. 3.2.

Таблиця 3.2 Опис складових класу teachDecart

ЗМІНІ			
Назва	Тип	Модифікатор доступу	Призначення
`decarts`	`List<>`	private	Колекція класів на яких навчається ІС
`para_d`	`int[]`	private	Відстань до сусіда
`para`	`int[]`	private	Список 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 критерію для класів
`my_do`	`int[]`	private	Значення радіусів вирішальних правил
`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()`			Отримання граничного значення дельта
`public float getPointParallel(int i, int j)`			Отримання результату паралельної оптимізації за його індексами
`public int getDelta()`			Отримання значення дельта

`public float getPointFinally_KFE(int k, int i, int j)`	Отримання результату оптимізації радіусів за його індексами
`public Bitmap getClass(int i)`	Отримання зображення із класу
`public decartCoord getDecart(int i)`	Отримання класу
`public float getEM(int i)`	Отримання КФЕ для певного класу
`public int getRadius(int i)`	Отримання радіусу гіперповерхні для певного класу
`public int getElemVD(int i)`	Отримання значення верхнього допуску за його індексом
`public int getElemND(int i)`	Отримання значення нижнього допуску за його індексом
`public int[] getVD()`	Отримання масиву верхніх допусків
`public int[] getND()`	Отримання масиву нижніх допусків
`void realization()`	Процес машинного навчання, який виконує створення бінарної карти, 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()`	Розбиття 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 взаємодіє з teachDecart, виконує екзамен. Опис методів і змінних наведено в таблиці 3.3.

Таблиця 3.3 Опис складових класу machineLearn

ЗМІНІ			
Назва	Тип	Модифікатор доступу	Призначення
`decarts`	`List<>`	private	Колекція класів

`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)`			Отримує значення яскравості пікселів із зображення
`public int examsDecart(Bitmap bitmap)`			Виконує алгоритм екзамену. Повертає значення класу до якого належить зображення.

Клас FFMPEG клас, який дозволяє взаємодіяти з відео, шляхом використання бібліотеки FFmpeg. Опис коду наведено в таблиці 3.4.

Таблиця 3.4 Опис складових класу FFMPEG

ЗМІНІ			
Назва	Тип	Модифікатор доступу	Призначення
`ffmpeg`	`Process`	private	Запуск локальної утиліти ffmpeg.exe
Методи			
Назва			Призначення
`private void startProcess()`			Запуск та налаштування процесу ffmpeg.exe
`public void createFrames(string input, string output, string fps)`			Розбиття відео на фрейми
`public void createVideo(string input, string output, string fps)`			Склеювання кадрів у відео

ВИСНОВКИ

1) У результаті виконання бакалаврської роботи було реалізовано бортову систему розпізнавання наземних об’єктів, з використання інформаційно-екстремального машинного навчання. У рамках обраної технології була виконана оптимізація системи контрольних допусків на ознаки розпізнавання, що дозволило отримати оптимальні геометричні параметри контейнерів класів розпізнавання.

2) За отриманими, в процесі машинного навчання, оптимальними геометричними параметрами контейнерів класів розпізнавання було побудовано вирішальні правила, які продемонстрували високу достовірність і оперативність при прийнятті класифікаційних рішень.

3) Була видвинута і експериментально підтверджена гіпотеза про обрання базового класу за дисперсією. Цей підхід дозволив, з мінімальними витратами часу, підвищити функціональну ефективність системи розпізнавання.

4) Для перевірки працездатності бортової системи розпізнавання було змодельовано місцевість, яка наближається до реальних умов роботи безпілотника. Таким чином, вдалося перевірити результат класифікації об’єктів, у рамках динамічного випробування.

5) У подальшому необхідно забезпечити інваріантність вирішальних правил до довільного положення наземного об’єкту у кадрі зони інтересу. Для цього, наприклад, можна удосконалити метод формування вхідної навчальної матриці шляхом оброблення кадрів цифрового зображення регіону в полярній системі координат.

6) Результати роботи впроваджено в Науково-дослідному центрі ракетних військ і артилерії Збройних сил України. Оформлено свідоцтво на авторський твір.

СПИСОК ЛІТЕРАТУРИ

1. Цветков В.Я. Геоинформационные системы и технологии: «Финансы и статистика», 1997. - 290 с.
2. Discover QGIS. https://qgis.org/en/site/about/index.html
3. Allen McDuffee, "Launch This New 9-Hour Solar-Powered Drone from Your Shoulder" Archived 2014-03-16 at the Wayback Machine, Wired, 13 August 2013.
4. BAYKAR Unmanned Aerial Vehicle Systems. https://baykardefence.com/uav-15.html
5. MAM-C Mini Akıllı Mühimmat – Roketsan. https://www.roketsan.com.tr/urun/mam-c-mini-akilli-muhimmat/
6. Israel's THOR Drone Is Ready for Just About Anything. https://nationalinterest.org/blog/middle-east-watch/israels-thor-drone-ready-just-about-anything-167192
7. Анохин П.К. Биология и нейрофизиология условного рефлекса — М.: Медицина, 1968. – 547 с.
8. Краснопоясовський А.С. Інформаційний синтез інтелектуальних систем керування: підхід, що грунтується на методі функціонально-статистичних випробувань.– Суми: Видавництво СумДУ, 2004.– 261 с.
9. Жамбю М. Иерархический кластер-анализ и соответствия. — М.: Финансы и статистика, 1988. — 345 с.
10. Karen Simonyan, Andrea Vedaldi, Andrew Zisserman. Visualizing Image Classification Models and Saliency Maps [2013]. https://arxiv.org/abs/1312.6034
11. Mukund Sundararajan, Ankur Taly, Qiqi Yan. Axiomatic Attribution for Deep Networks [2017]. https://arxiv.org/abs/1703.01365
12. Shawn Xu, Subhashini Venugopalan, Mukund Sundararajan. Attribution in Scale and Space. https://arxiv.org/pdf/2004.03383.pdf
13. Довбиш А.С. Основи проектування інтелектуальних систем: Навчальний посібник.– Суми: Видавництво Сум ДУ, 2009.– 171 с.

14. Довбиш А.С. Інтелектуальні інформаційні технології в електронному навчанні / А.С. Довбиш, А.В. Васильєв, В.О. Любчак. – Суми: Видавництво СумДУ, 2013.– 177 с.
15. 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
16. 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.
17. Banzhaf, Wolfgang; Nordin, Peter; Keller, Robert; Francone, Frank (1998). Genetic Programming – An Introduction

ДОДАТОК

Клас decartCoord.

```
using System;
using System.Drawing;

namespace machineLearning
{
    public class decartCoord
    {
        public Bitmap img;
        int[,] decart_map;
        int[,] binary_map;
        int[] etalon_vektor;

        int N;
        int n;
        float ro = 0.5f;
        float dispertion;
        float avr;

        public int[,] RGB;
        public decartCoord(Bitmap source)
        {
            img = source;

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

            etalon_vektor = new int[N];
            binary_map = new int[N, n];
            decart_map = new int[N, n];

            RGB = new int[N, 3];

            make_RGB();
        }
        public int getElem_EV(int i) {
            return etalon_vektor[i];
        }
        public int getElem_BM(int i, int j) {
            return binary_map[i, j];
        }
        public int getColumns() {
            return N;
        }
        public int getRows()
        {
            return n;
        }
```

```
        public int getPixel(int i, int j) {
            return decart_map[i, j];
        }
        public void setPixel(int i, int j, int value) {
            decart_map[i, j] = value;
        }
        public float getDispertion()
        {
            return dispertion;
        }
        public float getAvr() {
            return avr;
        }
        public void make_etalon_vektor()
        {
            for (int i = 0; i < N; i++)
            {
                float ev_sum = 0;
                for (int j = 0; j < n; j++)
                    ev_sum += binary_map[i, j];
                if ((ev_sum / n) >= ro) etalon_vektor[i] = 1;
                else etalon_vektor[i] = 0;
            }
        }
        public void make_binary_map(int[] vd, int[] nd)
        {
            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    if (decart_map[i, j] >= nd[i] && decart_map[i, j]
<= vd[i])
                        binary_map[i, j] = 1;
                    else
                        binary_map[i, j] = 0;
                }
            }
        }
        void make_RGB()
        {
            avr = 0;
            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;

                avr += (RGB[i, 0] + RGB[i, 1] + RGB[i, 2]) /3;
            }
            //середнє значення яскравості
            avr /= N;

            float distance = 0;
            for (int i = 0; i < N; i++)
                distance += (float)Math.Pow(((RGB[i, 0] + RGB[i, 1] +
RGB[i, 2])/3) - avr, 2);
            //дисперсія
            dispertion = distance / N;
        }
    }
}
```

Клас teachDecart

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

namespace machineLearning
{
    public class teachDecart
    {
        List<decartCoord> decarts = new List<decartCoord>();
        int[] para_d;
        int[] para;
        int[,] sk;
        int[,] sk_para;
        float[] em;
        int[] my_do;

        int[] vd;
        int[] nd;

        int limit_d = 80;
        int d;
        int used_class = 0;
        int[] d_posl;

        List<float[]> points_posledovatel_KFE;
        float[,] points_parallel_KFE;
        float[,,] points_finally_KFE;
        public teachDecart(List<decartCoord> sender)
        {
            foreach (var item in sender)
```

```
                decarts.Add(item);
            sk = new int[2, decarts[0].getRows()];
            sk_para = new int[2, decarts[0].getRows()];

            para_d = new int[decarts.Count];
            para = new int[decarts.Count];

            em = new float[decarts.Count];
            my_do = new int[decarts.Count];

            vd = new int[decarts[0].getColumns()];
            nd = new int[decarts[0].getColumns()];

            d_posl = new int[decarts[0].getColumns()];

            points_parallel_KFE = new float[limit_d, 2];
            points_finally_KFE = new float[decarts.Count,
decarts[0].getColumns(), 2];
            points_posledovatel_KFE = new List<float[]>();

            //пошук правильноъ послідовності классів
            searchOptimClass();
        }
        public int getLimitD() {
            return limit_d;
        }
        public int getRunsSequential() {
            return points_posledovatel_KFE.Count *
points_posledovatel_KFE[0].Length;
        }
        public int getDelta() {
            return d;
        }
        public float getPointParallel(int i, int j)
        {
            return points_parallel_KFE[i, j];
        }
        public List<float[]> getPointsSequential()
        {
            return points_posledovatel_KFE;
        }
        public int getColumnsFinally_KFE() {
            return points_finally_KFE.GetLength(1);
        }
        public float getPointFinally_KFE(int k, int i, int j)
        {
            return points_finally_KFE[k, i, j];
        }
        public Bitmap getClass(int i) {
            return decarts[i].img;
        }
```

```
        public decartCoord getDecart(int i)
        {
            return decarts[i];
        }
        public float getEM(int i) {
            return em[i];
        }
        public int getRadius(int i) {
            return my_do[i];
        }
        public int getVD(int i) {
            return vd[i];
        }
        public int getND(int i)
        {
            return nd[i];
        }
        void realization()
        {
            foreach (var decart in decarts)
                decart.make_binary_map(vd, nd);
            for (int k = 0; k < decarts.Count; k++)
                decarts[k].make_etalon_vektor();

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

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

                vd[i] = sum + delta;
                nd[i] = sum - delta;

                if (vd[i] > 255 || nd[i] < 0)
                {
                    limit_d = delta;
                    return false;
                }
            }
            return true;
        }
        void make_para()
        {
            for (int k = 0; k < decarts.Count; k++)
```

```
            {
                para_d[k] = 0;
                int ev_sum1 = decarts[k].getColumns();
                for (int k1 = 0; k1 < decarts.Count; k1++)
                {
                    if (k1 != k)
                    {
                        int ev_sum = 0;
                        for (int i = 0; i < decarts[k].getColumns();
i++)
                        {
                            //считаем расстояние между эталонными
векторами
                            if (decarts[k].getElem_EV(i) !=
decarts[k1].getElem_EV(i))
                                ev_sum++;
                        }
                        //минимальное растояние до соседа
                        if (ev_sum <= ev_sum1)
                        {
                            ev_sum1 = ev_sum;
                            para[k] = k1;
                            para_d[k] = ev_sum1;
                        }
                    }
                }
            }
        }
        double INFK(int INFK_d, ref float INFK_d1, ref float
INFK_betta, int k)
        {
            // Слагаемое для формулы критерия
            int k1 = 0, k4 = 0;
            //Начальный алгоритм
            for (int INFK_j = 0; INFK_j < decarts[k].getRows();
INFK_j++)
            {
                //кол.  своих реализаций в сфере
                if (sk[0, INFK_j] <= INFK_d) k1++;
                //кол.  соседних реализаций в сфере
                if (sk[1, INFK_j] <= INFK_d) k4++;
            }
            //Характеристики
            //1-а достовірність
            INFK_d1 = (float)k1 / (float)decarts[k].getRows();
            //помилка 2-го роду
            INFK_betta = (float)k4 / (float)decarts[k].getRows();

            float d1_b = INFK_d1 - INFK_betta;
            double best_res = 1 * Math.Log((1 + 1 + 0.001) / (1 - 1 +
0.001));
```

```
            //Значення КФЕ
            //return d1_b * Math.Log((1 + d1_b + 0.1) / (1 - d1_b +
0.1)) / Math.Log(2);
            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 < decarts[k].getRows(); j++)
            {
                //расстояние от сосед реализ до своего EV
                sk[1, j] = 0;
                //расстояние от своей реализ до своего EV
                sk[0, j] = 0;
                //расстояние от сосед реализ до своего EV
                sk_para[1, j] = 0;
                //расстояние от своей реализ до соседнего EV
                sk_para[0, j] = 0;
                for (int i = 0; i < decarts[k].getColumns(); i++)
                {
                    if (decarts[k].getElem_EV(i) !=
decarts[para[k]].getElem_BM(i, j)) sk[1, j] += 1;
                    if (decarts[k].getElem_EV(i) !=
decarts[k].getElem_BM(i, j)) sk[0, j] += 1;
                    if (decarts[para[k]].getElem_EV(i) !=
decarts[para[k]].getElem_BM(i, j)) sk_para[1, j] += 1;
                    if (decarts[para[k]].getElem_EV(i) !=
decarts[k].getElem_BM(i, j)) sk_para[0, j] += 1;
                }
            }
        }
        void make_myDo()
        {
            //достоверности
            float t_d1 = 0, t_betta = 0;
            points_finally_KFE = new float[decarts.Count,
decarts[0].getColumns(), 2];

            for (int k = decarts.Count - 1; k >= 0; k--)
            {
                em[k] = 0;
                /*->*/
                my_do[k] = 0;
                //считаем растояние к своим и соседним реализациям
                make_sk(k);
                float d_correct = 999;
                float d_tmp;
                //перебираем радиусы
                for (int d = 1; d < decarts[k].getColumns(); d++)
                {
                    //вычисляем КФЭ для каждого радиуса
```

```
                    double e = INFK(d, ref t_d1, ref t_betta, k);
                    if (e >= em[k] && t_d1 >= 0.5 && t_betta <= 0.5 &&
d < para_d[k])
                    {
                        // em[k] = (float)e;
                        if (e > em[k])
                            my_do[k] = d;
                        em[k] = (float)e;

                        d_tmp = (float)d / para_d[k];
                        if (e == em[k] && d_tmp < d_correct) {
                            d_correct = d_tmp;
                            my_do[k] = d;
                        }
                    }
                    if (e > em[k] && my_do[k] == 0)
                    {
                        d_correct = (float)d / para_d[k];
                        em[k] = (float)e;
                    }
                    //координаты для графика
                    if (t_d1 >= 0.5 && t_betta <= 0.5 && d <
para_d[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;
        }
        public float average_KFE()
        {
            float avr = 0;
            for (int k = 0; k < decarts.Count; k++)
                avr += em[k];
            avr /= decarts.Count;
            return avr;
        }
        void searchOptimClass() {
            ////нахождение оптимального класса
            //края розподілу
            float min = 999, max = -1;
            for (int i = 0; i < decarts.Count; i++)
```

```
            {
                if (decarts[i].getAvr() < min)
                    min = decarts[i].getAvr();
                if (decarts[i].getAvr() > max)
                    max = decarts[i].getAvr();
            }
            //якщо краї близько один до одного, то обераємо більшу
дисперсію. Якщо далеко, то меншу.
            for (int i = 0; i < decarts.Count - 1; i++)
            {
                for (int j = 0; j < decarts.Count - i - 1; j++)
                {
                    if (Math.Abs(max - min) < 255 / 2)
                    {
                        if (decarts[j + 1].getDispertion() >
decarts[j].getDispertion())
                        {
                            decartCoord temp = decarts[j + 1];
                            decarts[j + 1] = decarts[j];
                            decarts[j] = temp;
                        }
                    }
                    else
                    {
                        if (decarts[j + 1].getDispertion() <
decarts[j].getDispertion())
                        {
                            decartCoord temp = decarts[j + 1];
                            decarts[j + 1] = decarts[j];
                            decarts[j] = temp;
                        }
                    }
                }
            }
        }

        public void parallelOptimization()
        {
            //delta для поля допусков
            float em_last_d = 0;
            float radius = 0;

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

                float em_now = average_KFE();
                //проверка радиусов контейнеров

                if (propusk_KFE(my_do[used_class], used_class))
```

```
                {
                    points_parallel_KFE[delta - 1, 1] = em_now;

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

                if (em_now > em_last_d &&
propusk_KFE(my_do[used_class], used_class))
                {
                    em_last_d = em_now;
                    d = delta;
                    radius = my_do[used_class];
                }
                else if (em_now == em_last_d && radius <
my_do[used_class] && propusk_KFE(my_do[used_class], used_class))
                {
                    em_last_d = em_now;
                    d = delta;
                    radius = my_do[used_class];
                }
            }

            make_dopusk(used_class, d);
            realization();
        }
        bool local_dopusk(int i, int delta)
        {
            int sum = 0;
            for (int j = 0; j < decarts[used_class].getRows(); j++)
                sum += decarts[used_class].getPixel(i, j);

            sum /= decarts[used_class].getRows();

            vd[i] = sum + delta;
            nd[i] = sum - delta;

            if (vd[i] > 255 || nd[i] < 0)
                return false;
            return true;
        }
        public void sequentialOptimization()
        {
            float last_res_G = 0, now_res_G = 0;
            for (int i = 0; i < decarts[used_class].getColumns(); i++)
                d_posl[i] = d;

            now_res_G = average_KFE();
```

```
            do
            {
                float[] points = new
float[decarts[used_class].getColumns()];
                last_res_G = now_res_G;

                for (int i = 0; i < decarts[used_class].getColumns();
i++)
                {
                    float last_res_L = 0;
                    float radius = 0;

                    for (int delta = 1; delta < limit_d; delta++)
                    {
                        //изменение допуска одного признака
                        if (!local_dopusk(i, delta)) break;
                        realization();

                        float now_res_L = average_KFE();
                        if (now_res_L > last_res_L)
                        {
                            last_res_L = now_res_L;
                            d_posl[i] = delta;

                            radius = my_do[used_class];
                        }
                        else if (now_res_L == last_res_L && radius <
my_do[used_class])
                        {
                            last_res_L = now_res_L;
                            d_posl[i] = delta;

                            radius = my_do[used_class];
                        }
                    }
                    //зміна допуску однієї ознаки
                    local_dopusk(i, d_posl[i]);
                    realization();
                    points[i] = average_KFE();
                }
                //збереження інформації при оптимізації всіх допусків
                points_posledovatel_KFE.Add(points);

                now_res_G = average_KFE();
            } while (Math.Abs(now_res_G - last_res_G) > 0.01);

            realization();
        }
    }
}
```

Клас machineLearn

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

namespace machineLearning
{
    public class machineLearn
    {
        //колекція класів
        List<decartCoord> decarts;
        //навчання в декартовій системі
        public teachDecart teach_decart;
        public machineLearn()
        {
            //колекція класів
            decarts = new List<decartCoord>();
        }
        public void realization_decart()
        {
            teach_decart = new teachDecart(decarts);

            teach_decart.parallelOptimization();
            teach_decart.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);
                }
            }
            decarts.Add(decart);
        }
    }
}
```

Клас FFMPEG

```
using System.Diagnostics;
namespace machineLearning
{
    class FFMPEG
    {
        Process ffmpeg;
        void startProcess() {
            //налаштування FFMPEG
```

```
            ffmpeg.StartInfo.FileName = "utils/ffmpeg.exe";
            ffmpeg.StartInfo.UseShellExecute = false;
            ffmpeg.StartInfo.RedirectStandardOutput = true;
            ffmpeg.StartInfo.RedirectStandardError = true;
            ffmpeg.StartInfo.CreateNoWindow = true;
        }
        //розбиття відео на кадри
        public void createFrames(string input, string output, string
fps)
        {
            if (fps == null) fps = "3";

            ffmpeg = new Process();
            //виконання скрипту для FFMPEG
            //приклад
            //-i video.webm -vf fps=3 thumb%04d.jpg -hide_banner
            ffmpeg.StartInfo.Arguments = " -i " + input + " " + "-vf
fps=" + fps + " -qscale:v 2 " + output + "frame%d.jpg" + " -
hide_banner";
            startProcess();

            //початок процесу
            ffmpeg.Start();
            ffmpeg.WaitForExit();
            ffmpeg.Close();
        }
        //створення відео із кадрів
        public void createVideo(string input, string output, string
fps)
        {
            if (fps == null) fps = "3";
            ffmpeg = new Process();
            //виконання скрипту для FFMPEG
            //приклад
            //-f image2 -r 24 -i frame%04d.jpg -sameq -y -an -r 24
out.flv
            string text = "-f image2 -r " + fps + " -i " + input +
"frame%d.png -y -an " + output;
            ffmpeg.StartInfo.Arguments = "-f image2 -r " + fps + " -i
" + input + "frame%d.png -y -an " + output;
            startProcess();

            //початок процесу
            ffmpeg.Start();
            ffmpeg.Close();
        }
    }
}
```