m-math_2025_009

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Факультет математики та інформатики
кафедра математичного моделювання

ВИЗНАЧЕННЯ ОРИГІНАЛЬНОСТІ ВІДЕО РЕСУРСІВ ЗАСОБАМИ ML.NET

Кваліфікаційна робота
Рівень вищої освіти – другий (магістерський)

Виконав:
студент 2 курсу, 601 групи
Пузенко Євген Володимирович
Керівник:
канд. фіз.-мат. наук, доцент
Горбатенко Микола Юрійович

До захисту допущено
на засіданні кафедри
протокол № 5 від 26 листопада 2024 р.
Зав. кафедрою _____ проф. Черевко І.М.

Чернівці – 2024

АНОТАЦІЯ

Метою даної роботи є дослідження відео ресурсів для визначення їх оригінальності. Розроблене програмне забезпечення дає змогу визначити та оцінити відео ресурс перед подальшим аналізом.

Галузі використання результатів даного дослідження: криптографія, комп'ютерна графіка, машинне навчання.

Кількість ілюстрацій - 53.

Кількість використаних джерел - 9.

Кількість додатків - 1.

Обсяг роботи - 46 сторінок.

Ключові слова: .NET, C#, WPF, ML.NET, Cryptohraphy, Researching, AnomalyDetection, MulticlassClassification.

ABSTRACT

The goal of this paper is to research video resources to determine their originality. The developed software allows to identify and evaluate video resources before further analysis.

Areas of use of the results of this research: cryptography, computer graphics, machine learning.

The number of illustrations - 53.

Number of sources used - 9.

Number of applications - 1.

The volume of work - 46 pages.

Keywords: .NET, C#, WPF, ML.NET, Cryptohraphy, Researching, AnomalyDetection, MulticlassClassification.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ Пузенко Є.В.

Зміст

Вступ

У рамках кваліфікаційної роботи на тему: “Визначення оригінальності відео ресурсів засобами ML.NET” вивчалися можливості нейронних мереж для пошуку аномалій у відеоресурсах з використанням технології ML.NET. Дослідження дозволило поглибити мої знання в галузях машинного навчання, комп’ютерної графіки, систем приховування та аналізу даних, проектування програмних застосунків.

Намагання людей видати одну інформацію за іншу тягнулися з давніх-давен і не зупинилися й по сьогодні. Як раніше, так і зараз, люди намагалися сховати справжню інформацію, змінилися тільки методи.

. Під час виконання роботи було розглянуто наступні теми:

- опис технологій, які були використані при виконанні дослідження;
- аналіз аномалії;
- мультикласова класифікація;
- пошук аномалій;
- створення набору даних;
- логістична функція;
- процес навчання й оцінювання мереж;
- аналіз отриманих результатів.

Розділ 1 - Опис технологій та теоретичні відомості

В даному розділі я описав технології що використав та теоретичні відомості якими необхідно володіти для розуміння викладеного матеріалу.

1.1 Піксель

Піксель(від англ. pixel, є скороченням від picture element) - це атомарна одиниця зображення у растровій графіці. Всі зображення в растровій графіці складаються із пікселів, які, як правило, розташовані у вигляді сітки. Кожен піксель може містити, як один колір (канал кольору), так і більше.

У зображеннях, є поняття роздільної здатності зображення - це кількість пікселів на одиницю площі екрана. Чим вища роздільна здатність екрана, тим чіткішим і детальнішим виглядає зображення.

Кожен піксель має певний колір, який може формуватися за допомогою як 1 каналів, так і декількох. Сьогодні, найбільш розповсюдженим варіантом є RGB, що складається з 3 каналів. Комбінації значень каналів формують фінальний колір пікселя. В цьому випадку:

R (red) – червоний;

G (green) – зелений;

B (blue) - синій.

Кількість пікселів у зображенні також визначає поняття роздільної здатності екрана. Наприклад 720p становить 1280x720 пікселів, де 1280 - кількість пікселів у рядків зображення, а 720 - кількість рядів зображення.

1.2 Контейнери зображень

Контейнери зображень можуть бути файлами або форматами, які містять у собі інформацію про пікселі зображення та додаткову інформацію. Серед популярних представників є формати PNG і JPEG. Ці формати мають

свої характеристики, зокрема підтримку прозорості, ступінь стиснення та збереження якості зображення.

Контейнери зображень є важливими, оскільки забезпечують оптимізацію розміру зображення, швидкість завантаження та збереження інформації.

JPEG(Joint Photographic Experts Group):

- один із найбільш використовуваних форматів для зображень з високою кількістю кольорів;
- має малий розмір файлу, оскільки використовує стиснення з втратами;
- ідеальний для фотографій, де важливо підтримувати велику кількість кольорів і малий розмір;
- не підтримує прозорість.

PNG (Portable Network Graphics):

- використовує стиснення без втрат, тобто зберігає всі пікселі зображення;
- підтримує прозорість;
- ідеальний у випадках, де треба зберегти дані без втрат.

1.3 Відео

Відео - це набір зображень, які можуть відтворюватися послідовно або випадково. Ця технологія була створена для зберігання та відтворення візуальної інформації у формі рухомих зображень.

Існують відео, де кожен кадр відео представлений у вигляді окремого зображення. Такі відео називають сирими, або raw video. На противагу їм існують відео, які містять лише базовий кадр та зміни у наступних кадрах. Такі відео називати стисненими, або compressed video.

Для досягнення мети зменшення розміру відео файлів були придумані алгоритми стискання, які заведено називати кодеками. Вони забезпечують правильний запис і стискання відео.

1.4 Відео контейнери

Відео контейнери - це формати файлів, які використовуються для збереження відео, аудіо і метаданих у файлі. Контейнер, в собі, містить закодоване відео та аудіо, в одному файлі. Збереження даних у контейнері відбувається завдяки кодекам, для відео і аудіо використовують різні кодеки залежно від потреб користувача. До популярних відео форматів можна віднести: MP4, MKV, AVI, WEBM.

Відео кодеки поділяють на ті, що змінюють дані і ті, що не змінюють їх. До популярних кодеків можна віднести: H264, H265, FFV1.

1.5 Шифрування

Шифрування - це процес перетворення даних користувача, зрозумілих для користувача, у криптографічно захищені дані. Ще з давніх часів людство використовує ті чи інші методи для захисту інформації. Одним із таких методів шифрування був шифр Цезаря. В основі всіх шифрів лежить сам алгоритм шифрування та ключ або пара ключів. Алгоритм виконує роль системи, що шифрує, а ключ вказує як системі працювати.

Шифрування використовується для забезпечення конфіденційності під час передачі або збереження даних. Процес шифрування базується на використанні алгоритмів, які змінюють структуру чи вміст даних таким чином, щоб без ключа вони були незрозумілими й важко піддавалися розшифруванню.

Сьогодні, як і в минулому, використовуються різні алгоритми шифрування, як для безпечного зберігання, так і для безпечного

транспортування. Сьогодні захист особистих даних, паролів, банківських транзакцій є тими даними, які найчастіше шифрують. Для цього в основі алгоритмів шифрування закладають сильні математичні функції та довгі ключі для забезпечення високої надійності й стійкості шифрування.

Сьогодні розділяють 2 види шифрування: симетричне і асиметричне.

Асиметричне шифрування - це такий вид шифрування, де для шифрування використовується 1 ключ, а для розшифрування інший. Інша назва цих ключів публічний і приватний, де публічний використовується для шифрування, а приватний для розшифрування.

Якщо порівнювати цих 2 види шифрування, то симетричне є класичним способом забезпечення безпеки інформації. Добре підходить для будь-яких об’ємів даних, проте має проблему із передачею ключів між користувачами. Асиметричне ж є ідеальним для передачі між користувачами по загальним мережам, оскільки, приватні ключі, тобто спосіб розшифрувати повідомлення, залишається у власника. Публічний ключ можна пересилати вільно не боячись за безпеку даних.

Зараз існує багато алгоритмів шифрування, проте найбільш вживаними є AES, RC5, HC-256, RSA, DSA.

1.6 AES

AES (Advanced Encryption Standard) - це симетричний алгоритм шифрування, розроблений у 2001 році Деймоном Д. і Ріджменом В.. Він використовується для шифрування і дешифрування даних. Цей алгоритм і досі є криптографічно стійким та з 2002 року був прийнятий, як міжнародний стандарт для шифрування даних.

Даний алгоритм базується на використанні алгоритму перестановки байтів даних з використанням ключа шифрування. Даний алгоритм підтримує ключі різної довжини, наприклад, 128, 196, 256 бітів. Такий підхід та різноманіття ключів дозволяє використовувати різну довжину ключів для

шифрування і розшифрування у випадках коли час критичний. Також різна довжина ключів забезпечує різні рівні безпеки. Чим довший ключ, тим вищий рівень безпеки, проте і час операцій довший.

1.7 C#

Така мова програмування як C# була презентована у 2001 році завдяки роботам Гейлсберга Андерса, Вільтамута Скота та Гольде Пітера. Від самого початку існування та до сьогодні має велику та активну спільноту розробників що продовжують підтримку і розвиток мови. Дана мова та технології цієї мови задали стандарти індустрії. Було розроблено безліч унікальних та цікавих підходів до розробки програмного забезпечення. C# є об’єктно орієнтованою мовою програмування. Тому всі компоненти програм є об’єктами, які регулюють свою поведінку та властивості. Дану мову використовують для розробки різних програм для різних платформ. Дана мова підтримує розробку додатків під різні платформи, наприклад, веб-розробка, десктопна розробка, мобільна розробка. Оскільки C# розроблявся, як конкурент Java, то і синтаксис в них схожий і продовжує бути Сі подібним.

Для створення об’єктів з однаковими властивостями й поведінкою використовуються шаблони які в C# називають класи та структури.

Для взаємодії між об’єктами використовують методи, конструктори, поля і властивості.

Оскільки мова C# є об’єктно орієнтованою мовою тому в ній закладено основні принципи такого виду програмування, а саме: поліморфізм, абстракція, інкапсуляція і наслідування.

1.8 ML.NET

ML.NET - це кросплатформенна бібліотека для створення, покращення, використання моделей машинного навчання. Розробником є компанія Microsoft. Була розроблена з метою полегшити розробникам платформи .net інтеграцію моделей машинного навчання із їх додатками без необхідності вивчати інші мови програмування, такі як Python. Хоч ML.NET і не покриває 100% можливостей аналогічних бібліотек в Python, проте вже дозволяє працювати із різноманітними типами задач, включаючи класифікацію, регресію, кластеризацію, пошук аномалій, систем рекомендацій та інші.

ML.NET підтримує роботу, як з перенавченими моделями, так і можливість навчити модель з нуля використовуючи користувацькі дані.

У ML.NET можна працювати, як із класичними табличними даними, так і з об’єктами, такими як зображення, або текст. Також ML.NET підтримує ONNX, що дозволяє імпортувати моделі, що були створені в інших бібліотеках, таких як PyTorch чи TensorFlow.

1.9 MulticlassClassificationTrainer

MulticlassClassificationTrainer - це клас який використовується для навчання моделей класифікації зображень. В його основі лежить глибока нейронна мережа, що дає змогу розробникам інтегрувати дану мережу у свій проект без використання TensorFlow.

Даний класифікатор використовує попередньо навчену мережу, таку як ResNet і адаптує її до нових даних. Це дозволило значно прискорити процес навчання і зробити його ефективнішим, оскільки попередньо навчена модель використовує накопичені знання під час навчання на великих наборах даних.

Даний класифікатор має підтримку GPU, що може значно прискорити до навчання.

1.10 AnomalyDetection

Time-Series in AnomalyDetection - це модуль, що дозволяє автоматично ідентифікувати незвичайні або несподівані події в послідовних даних. Зазвичай використовується для визначення незвичайних подій у фінансових транзакціях, трафіку, аналітичних даних тощо.

В ML.NET представлено два методи для виявлення аномалій.

DetectIidChangePoint - це метод у ML.NET, який використовується у виявленні точок зміни у послідовних даних. Під точками зміни розуміються різкі зміни даних, наприклад раптове зростання або спад значень, що можуть вказувати на аномальність.

DetectIidSpike - це метод у ML.NET що використовується для виявлення раптових викидів у часових рядах. Дані викиди можуть вказувати на аномальні події, або відхилення.

1.11 WPF

WPF(Windows Presentation Foundation) - це потужна технологія від Microsoft, що була створена для створення та підтримки сучасних настільних додатків під платформу Windows. Основною метою розробників WPF було створення простих, легко масштабованих інтерфейсів користувача, в тому числі графіків, анімацій і мультимедіа. WPF використовує мову XAML для опису програмних об’єктів, таких як інтерфейс користувача. Завдяки XAML користувач може відокремити логіку інтерфейсу від бізнес-логіки. Це робить процес розробки легшим і спрощує підтримку коду.

У WPF є можливість працювати, як з растровою, так і векторною графікою, що дозволяє легко створювати інтерфейси для екранів з різною роздільною здатністю.

У WPF є підтримка 2Д і 3Д графіки, що дозволяє працювати із 3Д об’єктами та дозволяє створювати складні інтерфейси з тривимірними елементами.

У WPF є можливість прив’язки даних, це дозволяє підключати UI елементи до джерел даних, наприклад баз даних, чи API сервісів. Це дозволяє забезпечити динамічне оновлення інтерфейсу в реальному часі.

1.12 OpenCvSharp4

OpenCvSharp4 - це обгортка для .NET, для бібліотеки OpenCv що використовується для роботи з комп’ютерним зором, обробкою зображень і відео. OpenCvSharp4 надає можливість використовувати можливості OpenCv у додатках на мові C#.

OpenCvSharp4 підтримує великий набір функцій для обробки зображень. В тому числі зміну розміру, роботу з пікселями, повороти, фільтрацію й інші операції, як з контентом зображення, так і метаданими.

Робота з відео, також, можлива з цією бібліотекою, вона надає можливість обробки відео, читання чи захоплення кадрів відео з зовнішніх відео потоків, створення, читання і запис відео файлів.

Однією із найпотужніших можливостей є функції які пов’язані з виявленням об’єктів таких як розпізнавання облич, виділення країв об’єктів і руху.

1.13 OxyPlot

OxyPlot - це кросплатформенна бібліотека для створення і відображення графіків, діаграм у додатках на основі .NET. OxyPlot надає можливості для побудови різноманітних типів графіків в тому числі лінійних, гістограм, кругових та інших. OxyPlot підтримує розробку під різні платформи в тому числі WPF, WinForms, Xamarin і веб розробку.

Розділ 2 – Опис програмного додатку

В даному розділі описано розроблений функціонал для програмного додатку та модулів. Технології і мови, які були обрані для виконання задачі. Внутрішню будову застосунку з описом інфраструктури проектів.

2.1 Вибір технологій

Для своєї роботи я використовував мову C# (Рисунок 2.1) та технології екосистеми .NET. Це дозволило мені працювати у відомому для мене середовищі. А завдяки великій кількості технологій, що працюють з .NET, вдалося значно підвищити швидкість і якість проведеної роботи

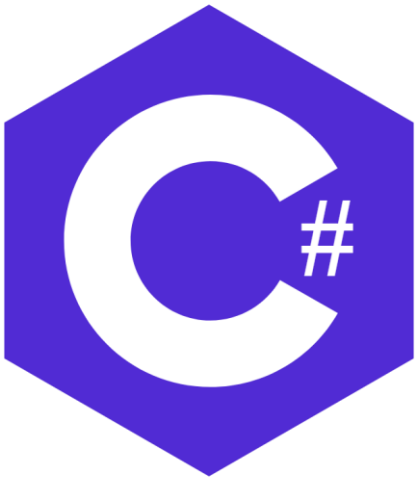

Рисунок 2.1 – Логотип мови C#

Для роботи з нейронними мережами я використовував технологію ML.NET (Рисунок 2.2). Оскільки саме ця технологія була офіційно розроблена Microsoft і рекомендується до використання у будь яких .NET проектах, де є необхідність використання нейронних мереж.

Рисунок 2.2 - Логотип ML.NET

Для створення додатка було використано технологію .NET WPF (Рисунок 2.3). Саме ця технологія є передовою й рекомендується для розробки додатків під Windows.

Рисунок 2.3 - Логотип WPF

Для роботи з графічними додатками, в тому числі з фото і відео, було використано сторонню відкриту технологію OpenCV (Рисунок 2.4), а саме її обгортку для зручної роботи в середині .NET OpenCvSharp4.

Рисунок 2.4 - Логотип OpenCV

Для створення графіків, використаних у 2 дослідженні було використано бібліотеку OxyPlot (Рисунок 2.5). Це популярне й підтримуване рішення для зручного, й швидкого створення, виведення графіків різних видів і форм.

Рисунок 2.5 - Логотип OxyPlot

2.2 Проектування додатку

Для забезпечення зрозумілої архітектури функціонал був рознесений на різні проекти. Кожен проект відповідав за свій функціонал. Було окремо створено модулі для роботи з зображеннями, відео, наборами даних, нейронною мережею і графічною оболонкою.

2.3 PixFlow

В даному проекті було описано і реалізовано базу для обробки відеоресурсів. Для забезпечення універсальності проекту мною був створений абстрактний рівень, що мав полегшити подальшу розробку та покращення функціоналу й більш керовану взаємодію в системі. Дані що були оброблені використовуючи цей проект в подальшому були використані для машинного навчання. Основним завданням даної частини додатку було забезпечити засоби для взаємодії, інтеграції й обробки відео та кадрів відео, а також забезпечити виконання базових операцій з обробки даних перед їх подачею у інші проекти. Прикладом такого проекту є MLPrep.

Основні компоненти PixFlow. Проект складається з декількох компонентів, кожен із них відповідає за свої специфічні аспекти обробки кадрів відеоресурсів.

Image: містить в собі функціонал для роботи із кадрами відео.

Interfaces: у даній папці містяться загальні інтерфейси для роботи з зображеннями, такі як IHighlighter і IInsertable.

IHighlighter – це інтерфейс, що визначає метод для виділення областей за певним параметром використовуючи 2 кадри (Рисунок 2.6).

```
4 references
public interface IHighlighter
{
    6 references
    public Mat Highlight(Mat frame1, Mat frame2);
}
```

Рисунок 2.6 - Інтерфейс IHighlighter й його методи

IInsertable – це інтерфейс (Рисунок 2.7) що визначає метод для вставки зображень за певними правилами, що можуть бути розписані в InsertOperation.

```
3 references
public interface IInsertable
{
    3 references
    public void Insert(Mat frame, IEnumerable<InsertOperation> operations);
}
```

Рисунок 2.7 - Інтерфейс IInsertable й його методи

Models: дана папка містить структури даних, моделі даних. В даному випадку, там знаходиться 1 модель для опису процесу вставлення даних у певний кадр відео.

InsertOperaion – це модель (Рисунок 2.8), що описує правила за якими інформація має бути вставлена в певний кадр. В даній моделі описується процес вставлення інформації. В будь якій моделі дані вставляються у вигляді прямокутника довільного розміру. Декілька моделей можуть бути застосовані для 1 кадру. У випадку накладання одної із моделей на іншу, будуть збережені зміни тої що була застосована останньою.

```
16 references
public record InsertOperation
{
    2 references
    public IEnumerable<byte> Bytes { get; init; }

    5 references
    public IPixelEditable Editor { get; init; }

    8 references
    public Rect Position { get; init; }

    7 references
    public InsertOperation(IEnumerable<byte> bytes, IPixelEditable editor, Rect position)...

}
```

Рисунок 2.8 - Модель InsertOperation

Services: папка з сервісами для роботи із зображеннями.

Diff – це сервіс (Рисунок 2.9), який був розроблений для отримання абсолютної різниці між 2 кадрами. В його основі лежить використання вже

готового функціоналу OpenCvSharp4. Використання цієї різниці необхідно для подальшого підсвічування аномалії, якщо така існує в зразку.

```
2 references
public sealed class Diff : IHighlighter
{
    4 references
    public Mat Highlight(Mat frame1, Mat frame2)
    {
        ArgumentNullException.ThrowIfNull(frame1, nameof(frame1));

        ArgumentNullException.ThrowIfNull(frame2, nameof(frame2));

        if (frame1.Size() != frame2.Size())
        {
            throw new ArgumentException("Input frames must be of the same size.");
        }

        return frame1 - frame2;
    }
}
```

Рисунок 2.9 - Сервіс Diff

ImageInserter – це основний сервіс, що забезпечує виконання операцій з кадром. Саме в ньому в кадр вставляються дані із моделей згідно заданих правил (моделей) (Рисунок 2.10).

```
public void Insert(Mat frame, IEnumerable<InsertOperation> operations)
{
    ArgumentNullException.ThrowIfNull(frame, nameof(frame));

    ArgumentNullException.ThrowIfNull(operations, nameof(frame));

    if (!operations.Any())
    {
        throw new ArgumentException("Operations cannot be empty.");
    }

    foreach (var operation in operations)
    {
        int setBytes = 0;

        for (int y = operation.Position.Y; y < operation.Position.Y + operation.Position.Height; y++)
        {
            for (int x = operation.Position.X; x < operation.Position.X + operation.Position.Width; x++)
            {
                var pixel = frame.Get<Vec3b>(y, x);
                var bytes = operation.Bytes.Take(operation.Editor.BytesCountForPixel).ToArray();

                if (bytes.Length != operation.Editor.BytesCountForPixel)
                {
                    throw new ArgumentException("Byte array size does not match the expected size.");
                }

                frame.Set(y, x, operation.Editor.Set(pixel, bytes));

                setBytes += operation.Editor.BytesCountForPixel;
            }
        }
    }
}
```

Рисунок 2.10 - Метод Insert з сервісу ImageInserter та процес вставлення даних у кадр

Rotator – це сервіс що забезпечує обертання навколо центру зображення. Цілю використання даного сервісу було: забезпечити більше різноманіття даних для подальшого використання їх в мережі (Рисунок 2.11).

```
1 reference
public static class Rotator
{
    1 reference
    public static Mat Rotate(Mat image, int angle)
    {
        var center = new Point2f(image.Width / 2f, image.Height / 2f);

        Mat rotationMatrix = Cv2.GetRotationMatrix2D(center, angle, 1.0);

        double cos = Math.Abs(rotationMatrix.Get<double>(0, 0));
        double sin = Math.Abs(rotationMatrix.Get<double>(0, 1));

        int newWidth = (int)(image.Height * sin + image.Width * cos);
        int newHeight = (int)(image.Height * cos + image.Width * sin);

        rotationMatrix.Set(0, 2, rotationMatrix.Get<double>(0, 2) + (newWidth / 2) - center.X);
        rotationMatrix.Set(1, 2, rotationMatrix.Get<double>(1, 2) + (newHeight / 2) - center.Y);

        Mat dst = new Mat();
        Cv2.WarpAffine(image, dst, rotationMatrix, new Size(newWidth, newHeight));

        return dst;
    }
}
```

Рисунок 2.11 - Сервіс Rotator та метод і реалізація обертання

Pixel: містить функціонал для роботи із пікселями, забезпечує вставлення інформації в пікселі. Підтримує читання інформації з пікселя.

Interfaces: дана папка містить інтерфейс IPixelEditable (Рисунок 2.12), що визначає методи вставлення і отримання даних з пікселю.

```
5 references
public interface IPixelEditable
{
    4 references
    public Vec3b Set(in Vec3b origin, in byte[] bytes);

    6 references
    public byte[] Get(in Vec3b origin);

    4 references
    public int BytesCountForPixel { get; }
}
```

Рисунок 2.12 - Інтерфейс IPixelEditable й його методи

Services: в даній папці знаходиться реалізація інтерфейсу, що був описаний вище. Changer3B2G3R, що реалізує вставлення і отримання інформації за певним алгоритмом. В даному випадку інформація записується

в 3 останніх біти синього і червоного, і 2 останніх біта зеленого кольорів. Так само і з читанням.

```
public sealed class Changer3B2G3R : IPixelEditable
{
    4 references
    public int BytesCountForPixel => 1;

    4 references
    public Vec3b Set(in Vec3b origin, in byte[] bytes)
    {
        if (bytes == null || bytes.Length < 1)
        {
            throw new ArgumentException("Invalid byte array provided.");
        }

        var newPixel = new Vec3b
        {
            Item0 = (byte)(origin.Item0 >> 3 << 3 | bytes[0] & 0b111), // B
            Item1 = (byte)(origin.Item1 >> 2 << 2 | bytes[0] >> 3 & 0b11), // G
            Item2 = (byte)(origin.Item2 & 0b11111000 | bytes[0] >> 5 & 0b111) // R
        };

        return newPixel;
    }

    6 references
    public byte[] Get(in Vec3b origin)
    {
        byte data = 0;

        data |= (byte)(origin.Item0 & 0b00000111);
        data |= (byte)((origin.Item1 & 0b00000011) << 3);
        data |= (byte)((origin.Item2 & 0b00000111) << 5);

        return [data];
    }
}
```

Рисунок 2.13 - Реалізація Changer3B2G3R й процесу вставлення даних у піксель

Video: даний компонент містить функціонал для роботи із відео файлами.

Helpers: дана папка була створена для допоміжних класів. На даний момент в ній міститься тільки 1 допоміжний клас Codecs (Рисунок 2.14), що надає доступ для одного із найбільш ефективних кодеків без втрат, оскільки, серед структурних кодеків OpenCvSharp4 його нема.

```
3 references
public static class Codecs
{
    3 references
    public static FourCC FFV1 => FourCC.FromFourChars('F', 'F', 'V', '1');
}
```

Рисунок 2.14 - Codecs й FFV1

Interfaces: тут знаходяться інтерфейси для роботи із відео, включаючи IInsertable (Рисунок 2.15) та ISplittable (Рисунок 2.16). Дані інтерфейси визначають методи для вставлення даних у відео і функціонал розділення відео на частини. Саме завдяки цим інтерфейсам є можливість працювати із вставленням чи розділенням відео на частини.

```
1 reference
public interface IInsertable
{
    4 references
    public void Insert(string videoFilepath,
        IEnumerable<VideoInsertOperation> operations,
        bool useOriginalCodec = false);
}
```

Рисунок 2.15 - Інтерфейс IInsertable й його метод

```
public interface ISplitable
{
    2 references
    IEnumerable<Mat> Split(string videoFile);
}
```

Рисунок 2.16 - Інтерфейс ISplitable й його методи

Models: Тут знаходяться моделі, що використовуються даним компонентом. А саме: VideoInsertOperation (Рисунок 2.17), де можна вказати до якого фрейму відео які операції будуть використовуватися.

```
public record VideoInsertOperation
{
    3 references
    public int FrameIndex { get; init; }

    4 references
    public IEnumerable<InsertOperation> Operations { get; init; }

    4 references
    public VideoInsertOperation(int frameIndex, IEnumerable<InsertOperation> operations)
    {
        ArgumentNullException.ThrowIfNull(operations, nameof(Operations));

        Operations = operations;

        if (frameIndex < 0)
        {
            throw new ArgumentOutOfRangeException("FrameIndex cannot be less than 0", nameof(frameIndex));
        }

        FrameIndex = frameIndex;
    }
}
```

Рисунок 2.17 - Модель VideoInsertOperation

Services: У даній теці зберігаються основні сервіси для роботи із відео:

OneFrameSplitter – (Рисунок 2.18) що забезпечує послідовне читання кадрів відео файлу й забезпечує контроль за ресурсами, що забезпечує

відсутність витоків даних, чи того, що якийсь файл буде заблокований іншим процесом.

```
0 references
public class OneFrameSpliter : ISplitable
{
    2 references
    public IEnumerable<Mat> Split(string videoFile)
    {
        var reader = new VideoCapture(videoFile);

        using var mat = new Mat();

        while (reader.Read(mat))
        {
            var copy = new Mat();

            mat.CopyTo(copy);

            yield return copy;
        }

        reader.Release();
        mat.Release();
    }
}
```

Рисунок 2.18 - Сервіс OneFrameSpliter й його метод

VideoCutter – сервіс що був створений для розділення відео на сегменти із певної кількості фреймів (Рисунок 2.19).

```
public static class VideoCutter
{
    1 reference
    public static void Cut(string inVideoFile, string outVideoFile, int frameCount)
    {
        using var reader = new VideoCapture(inVideoFile);
        using var writer = new VideoWriter(
            outVideoFile, FourCC.FromString(reader.FourCC), reader.Fps,
            new Size(reader.FrameWidth, reader.FrameHeight));

        using var mat = new Mat();

        for (int i = 0; i < frameCount && reader.Read(mat); i++)
        {
            writer.Write(mat);
        }
    }
}
```

Рисунок 2.19 - Сервіс VideoCutter й його методи

VideoInserter – це сервіс (Рисунок 2.20) що забезпечує вставлення у відео користувацьких даних згідно інструкцій в моделях.

```
public sealed class VideoInserter(Image.Interfaces.IInsertable inserter = null!) : Interfaces.IInsertable
{
    private readonly Image.Interfaces.IInsertable _inserter = inserter ?? new Image.Services.ImageInserter();

    4 references
    public void Insert(string videoFilePath, IEnumerable<VideoInsertOperation> operations, bool useOriginalCodec = false)...
}
```

Рисунок 2.20 - Сервіс VideoInserter

VideoOperations – це сервіс (Рисунок 2.21) для інших операцій над відео. Таких як зміна Fps для зручнішого дослідження відео.

```
public class VideoOperations
{
    1 reference
    public static void SetFps(string filePath, double fps)
    {
        ArgumentException.ThrowIfNullOrEmpty(filePath, nameof(filePath));

        if (fps <= 0)
        {
            throw new ArgumentOutOfRangeException(nameof(fps), "FPS must be greater than zero.");
        }

        using var reader = new VideoCapture(filePath);
        if (!reader.IsOpened())
        {
            throw new IOException($"Failed to open video file: {filePath}");
        }

        string tempFilePath = Path.GetTempFileName() + ".avi";

        using var writer = new VideoWriter(tempFilePath, FourCC.FromString(reader.FourCC), fps,
            new Size(reader.FrameWidth, reader.FrameHeight));

        reader.Release();
        writer.Release();

        File.Delete(filePath);
        File.Move(tempFilePath, filePath);
    }
}
```

Рисунок 2.21 - Сервіс VideoOperations й його методи

VideoParams – це допоміжний сервіс (Рисунок 2.22), що надає інформацію про кадри відео, такі як їх кількість і розмір.

```
4 references
public sealed class VideoParams
{
    1 reference
    private VideoParams() { }

    2 references
    public int FrameCount { get; private set; }

    3 references
    public Size FrameSize { get; private set; }

    1 reference
    public static VideoParams Get(string videoPath)
    {
        ArgumentException.ThrowIfNullOrEmpty(videoPath, nameof(videoPath));

        if(!File.Exists(videoPath))
        {
            throw new IOException($"Failed to open video file: {videoPath}");
        }

        using var reader = new VideoCapture(videoPath);

        return new VideoParams()
        {
            FrameCount = reader.FrameCount,
            FrameSize = new Size(reader.FrameWidth, reader.FrameHeight)
        };
    }
}
```

Рисунок 2.22 - сервіс VideoParams

2.4 MLPrep

Даний проект був створений для підготовки даних перед передачею їх, для навчання мережею. Його метою було взяти дані, оброблені в PixFlow і застосувати до них алгоритми для перетворення таких кадрів у зразки.

До основних компонентів даного проекту можна віднести:

DatasetPreparator: в даному модулі були створені інструменти, які забезпечили PixFlow алгоритмами і даними. Саме ці алгоритми і дані були використані для генерування наборів даних.

CustomHighlighters – (Рисунок 2.23) цей модуль містить класи які використовуються для виділення пікселів кадру, що відносяться до заданих умов. Саме тут знаходиться 3 селектор, що найкраще себе показав і був впроваджений в додаток.

```
public class V3Selector : IHighlighter
{
    private static readonly Diff _diff = new();

    4 references
    public Mat Highlight(Mat frame1, Mat frame2)
    {
        if (frame1.Empty() || frame2.Empty())
        {
            throw new ArgumentException("One or both frames are empty.");
        }

        var dif = _diff.Highlight(frame1, frame2);

        for (int y = 0; y < dif.Size().Height; y++)
        {
            for (int x = 0; x < dif.Size().Width; x++)
            {
                var pixel = dif.Get<Vec3b>(y, x);

                if (pixel.Item0 < 8 && pixel.Item0 != 0 &&
                    pixel.Item1 < 4 && pixel.Item1 != 0 &&
                    pixel.Item2 < 8 && pixel.Item2 != 0)
                {
                    dif.Set(y, x, new Vec3b(255, 255, 255));
                }
                else
                {
                    dif.Set(y, x, new Vec3b(0, 0, 0));
                }
            }
        }
```

Рисунок 2.23 - Селектор 3 й його робота по виділенню пікселів

DataProviders: в даній теці знаходяться необхідні для роботи постачальники даних.

Interfaces: містить інтерфейс IBytesProvider (Рисунок 2.24), що визначає методи для забезпечення доступу до байтів інформації.

```
4 references
public interface IBytesProvider
{
    2 references
    IEnumerable<byte> Get(int count);
    3 references
    IEnumerable<byte> Get();
}
```

Рисунок 2.24 - Інтерфейс IBytesProvider й його методи

Services: папка в якій знаходяться реалізації інтерфейсу.

RandomBytesProvider – (Рисунок 2.25), як зрозуміло з назви використовує випадкові значення для подальшого їх вставлення у кадри відео. Використання даного сервісу є корисним, оскільки, в реальному житті, зашифровані дані часто виглядають як набір випадкових значень і виглядають, як шум. Сьогодні майже всі дані, що передаються через інтернет, шифруються для забезпечення безпеки і конфіденційності, імітуючи так структуру випадкових даних.

```
public sealed class RandomBytesProvider : IBytesProvider
{
    private Random _random = new Random();

    1 reference
    public IEnumerable<byte> Get(int count)
    {
        if (count <= 0)
        {
            throw new ArgumentOutOfRangeException(nameof(count), "Count must be greater than zero.");
        }

        var bytes = new byte[count];

        _random.NextBytes(bytes);

        return bytes;
    }

    2 references
    public IEnumerable<byte> Get()
    {
        yield return (byte)_random.Next(0, 255);
    }
}
```

Рисунок 2.25 - Сервіс RandomBytesProvider й реалізація інтерфейсу

TextBytesProvider – даний сервіс (Рисунок 2.26) надає можливість перетворити текст у байти, й взаємодіяти з даними у зрозумілому для програми способами.

```
public class TextBytesProvider : IBytesProvider
{
    private readonly string _text;

    private readonly Encoding _encoding;

    private int _index = 0;
    0 references
    public TextBytesProvider(string text, Encoding encoding)
    {
        ArgumentException.ThrowIfNullOrEmpty(text, nameof(text));
        ArgumentException.ThrowIfNullOrWhiteSpace(text, nameof(text));

        _text = text;
        _encoding = encoding;
    }

    1 reference
    public IEnumerable<byte> Get(int count)
    {
        var res = _encoding.GetBytes(_text, _index, count);

        _index += count;

        return res;
    }

    2 references
    public IEnumerable<byte> Get()
    {
        return _encoding.GetBytes(_text);
    }
```

Рисунок 2.26 - Сервіс TextBytesProvider й його методи

Додаткові сервіси для обробки зображень:

Cutter – це сервіс (Рисунок 2.27), що надає можливість відокремити певну кількість кадрів починаючи з першого.

```
sealed public class Cutter
{
    0 references
    public static void Cut(string directory,string outDirectory, int frameAmount)
    {
        foreach (var file in Directory.GetFiles(directory))
        {
            VideoCutter.Cut(file, Path.Combine(outDirectory, $"{Path.GetFileNameWithoutExtension(file)};cut.avi"), frameAmount);
        }
    }
}
```

Рисунок 2.27 - Сервіс Cutter й його реалізація

Preparator – це основний сервіс (Рисунок 2.28), що відповідає за підготовку зразків для мережі. Він використовує в собі функції, що були розроблені в інших сервісах для створення зразків із заданими параметрами.

```
public class Preparator : IDisposable
{
    private readonly bool _deleteTempVideoFiles;

    private readonly string _original;

    private string? _lastOperatedVideo;

    private readonly List<string> _filesToDelete = new List<string>();

    1 reference
    public Preparator(string fileName, bool deleteTempVideoFiles = true)...

    1 reference
    public Preparator Insert(IPixelEditable editor, IBytesProvider provider, double frameInserPercent, int[]? pattern = null)
    {
        var operations = GenerateOperations(_lastOperatedVideo == null ? _original :
            _lastOperatedVideo, editor, provider, frameInserPercent, pattern);

        return Insert(operations, frameInserPercent);
    }

    1 reference
    public Preparator Insert(IEnumerable<VideoInsertOperation> operations, double frameInsertPercent)
    {
        var inserter = new VideoInserter();
        var file = _lastOperatedVideo == null ? _original : _lastOperatedVideo;
        var insertFile = $"{Path.GetFileNameWithoutExtension(file)};inserted {frameInsertPercent:0.0}.avi";
        File.Copy(file, insertFile, true);

        inserter.Insert(insertFile, operations);
        _lastOperatedVideo = insertFile;
        _filesToDelete.Add(file);

        return this;
    }
```

Рисунок 2.28 - клас Preparator й його методи по вставленню даних у відео

В ньому знаходяться декілька великих функцій, що забезпечують виконання всієї роботи по підготовці даних. 2 варіанти функції Insert для вставлення даних у кадри, згідно згенерованих згідно заданого шаблону моделей, чи переданого користувачем. Метод HighlightByPattern (Рисунок 2.29) для обробки даних селекторами. Метод ToImages (Рисунок 2.30) для перетворення відредагованої копії відео на зображення.

```
0 references
public Preparator HighlightByPattern(IHighlighter highlighter, int[] pattern)
{
    if (!File.Exists(_lastOperatedVideo))
    {
        throw new FileNotFoundException("Insert file not found.", _lastOperatedVideo);
    }

    var highlightFile = $"{Path.GetFileNameWithoutExtension(_lastOperatedVideo)};highlighted.avi";

    _filesToDelete.Add(highlightFile);

    using var reader = new VideoCapture(_lastOperatedVideo);
    using var writer = new VideoWriter(highlightFile, Codecs.FFV1, reader.Fps, new Size(reader.FrameWidth, reader.FrameHeight));

    using var frame1 = new Mat();
    using var frame2 = new Mat();

    reader.Read(frame1);
    int i = 0;
    while (reader.Read(frame2))
    {
        var patternIteration = pattern[i % pattern.Length];
        if (patternIteration == 1)
        {
            using var highlightedFrame = highlighter.Highlight(frame1, frame2);
            writer.Write(highlightedFrame);
            Cv2.CopyTo(frame2, frame1);
        }
        i++;
    }
    _lastOperatedVideo = highlightFile;

    return this;
}
```

Рисунок 2.29 - метод HighlightByPattern сервісу Preparator

```
0 references
public Preparator ToImages(ISplitable spliter, string outDirectory)
{
    if (!File.Exists(_lastOperatedVideo))...

    if (!Directory.Exists(outDirectory))...

    int counter = 1;

    foreach (var mat in spliter.Split(_lastOperatedVideo))
    {
        var imagePath = Path.Combine(outDirectory, $"{Path.GetFileNameWithoutExtension(_lastOperatedVideo)};{counter++}.png");

        mat.SaveImage(imagePath);
    }

    return this;
}
```

Рисунок 2.30 - метод ToImages сервісу Preparator

Rotator – це сервіс (Рисунок 2.31), роль якого надати можливість повертати багато зображень одразу. Використовує функціонал PixFlow.Rotator.

```
internal class Rotator
{
    0 references
    public static void Rotate(string inDirectory, string outDirectory, int[] angles)
    {
        foreach (var file in Directory.GetFiles(inDirectory, "*.png", SearchOption.AllDirectories))
        {
            foreach (var angle in angles)
            {
                using var mat = new Mat(file);

                var rotated = PixFlow.Image.Services.Rotator.Rotate(mat, angle);

                var path = $"{Path.Combine(outDirectory, $"{Path.GetFileNameWithoutExtension(file)};rotated {angle}.png")}";

                rotated.SaveImage(path);
            }

        }
    }
}
```

Рисунок 2.31 - Сервіс Rotator й його метод

2.5 V3Sel

Даний проект був створений для класифікації зображень за допомогою бібліотеки ML.NET. Він дозволяє створювати, тренувати і використовувати нейронну модель для класифікації зображень. В якості даних використовує кадри у вигляді масивів байтів.

В проекті є 2 моделі: ModelInput (Рисунок 2.32) і ModelOutput.

ModelInput має 2 властивості, а саме:

ImageSource – байтовий масив, для зображення;

Label – текстова мітка класу.

```
14 references
public class ModelInput
{
    [LoadColumn(0)]
    [ColumnName(@"Label")]
    1 reference
    public string Label { get; set; }

    [LoadColumn(1)]
    [ColumnName(@"ImageSource")]
    3 references
    public byte[] ImageSource { get; set; }

}
```

Рисунок 2.32 - ModelInput з властивостями

ModelOutput (Рисунок 2.33) має ще 2 додаткові властивості, окрім властивостей ModelInput.

PredictedLabel – назва передбаченого класу;

Score – масив із значеннями ймовірностей для кожного класу.

```
public class ModelOutput
{
    [ColumnName(@"Label")]
    0 references
    public uint Label { get; set; }

    [ColumnName(@"ImageSource")]
    0 references
    public byte[] ImageSource { get; set; }

    [ColumnName(@"PredictedLabel")]
    0 references
    public string PredictedLabel { get; set; }

    [ColumnName(@"Score")]
    1 reference
    public float[] Score { get; set; }

}
```

Рисунок 2.33 - ModelOutput з властивостями

2.5.1 Тренер і пайплайн моделі

Нейронна мережа використовує тренер ImageClassification, який є частиною ML.NET. Завдяки використанню даного тренера було навчено мережу на основі зразків що були створені в PixFlow. В даному випадку в середині використовується ResNet50.

Пайплан (Рисунок 2.34) складається з декількох етапів:

- конвертація міток Label у числові для зручності обробки;
- використання тренера для класифікації зображень;
- зворотня конвертація числових міток у текстові значення.

Для тренування використовується метод RetrainModel, а пайплайн створюється у методі BuildPipeline.

```
rence
lic static IEstimator<ITransformer> BuildPipeline(MLContext mlContext)

var pipeline = mlContext.Transforms.Conversion.MapValueToKey(outputColumnName:@"Label",inputColumnName:@"Label",addKeyValueAnnotationsAsText:false)
                        .Append(mlContext.MulticlassClassification.Trainers.ImageClassification(
                            labelColumnName:@"Label",scoreColumnName:@"Score",featureColumnName:@"ImageSource"))
                        .Append(mlContext.Transforms.Conversion.MapKeyToValue(outputColumnName:@"PredictedLabel",inputColumnName:@"PredictedLabel"));

return pipeline;
```

Рисунок 2.34 - Pipeline

Після завершення роботи мережі повертається ModelOutput з масивом оцінок. Оцінки, одразу представлені у сортованому виді. Тому першим значення буде найбільше значення та те, якому класу належить передбачене значення.

2.6 WPFApp

В даному проекті я відобразив можливість користувача взаємодіяти з моделлю для визначення оригінальності відео. Я розробив дизайн вікна (Рисунок 2.35) для зручного використання користувачем розробленого функціоналу.

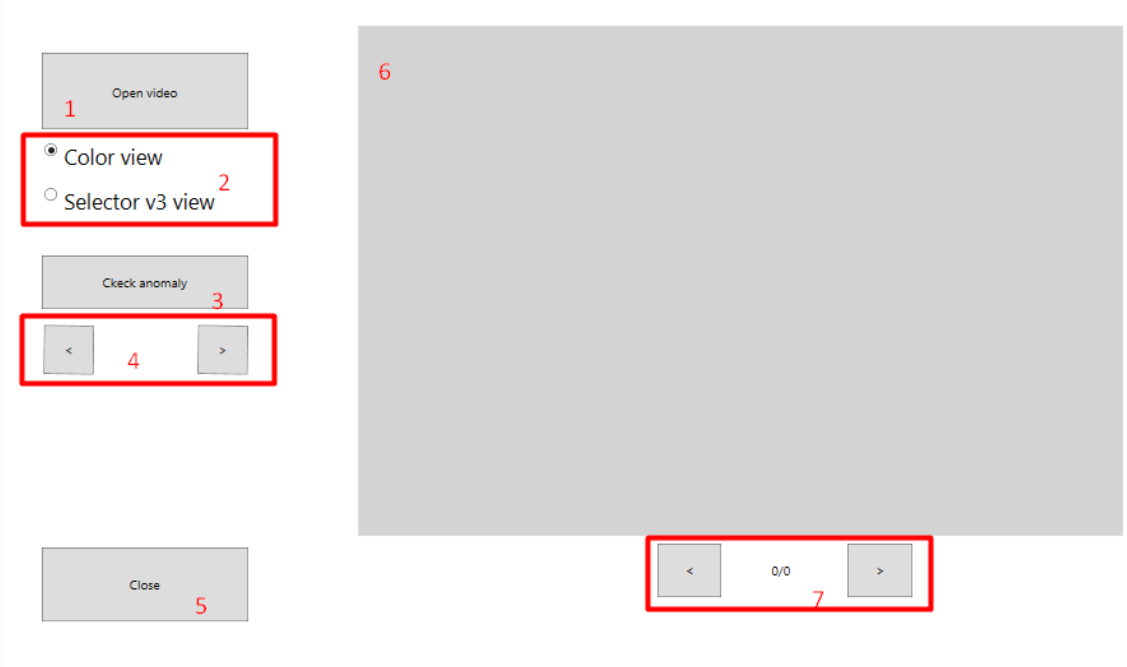

Рисунок 2.35 - Вікно для взаємодії користувача з моделлю

1. кнопка для вибору відео для перевірки;
2. набір з 2 можливостей перегляду контенту відео файлу;
3. кнопка для перевірки відео на оригінальність. Після перевірки поверне повідомлення з текстом щодо чи є відео аномальним чи ні;
4. стрілочки для того щоб перейти на наступний(>), чи попередній(<) аномальний кадр відео;
5. кнопка для того щоб закрити програму;
6. дисплей де будуть відображатися кадри відео;
7. стрілочки для переходу на наступний(>), чи попередній(<) кадр відео файлу.

В даному додатку є 2 (Рисунок 2.36) (Рисунок 2.37) способи спостерігати за відео файлом, а саме:

1. кольоровий – для перегляду відео у звичайному стані;

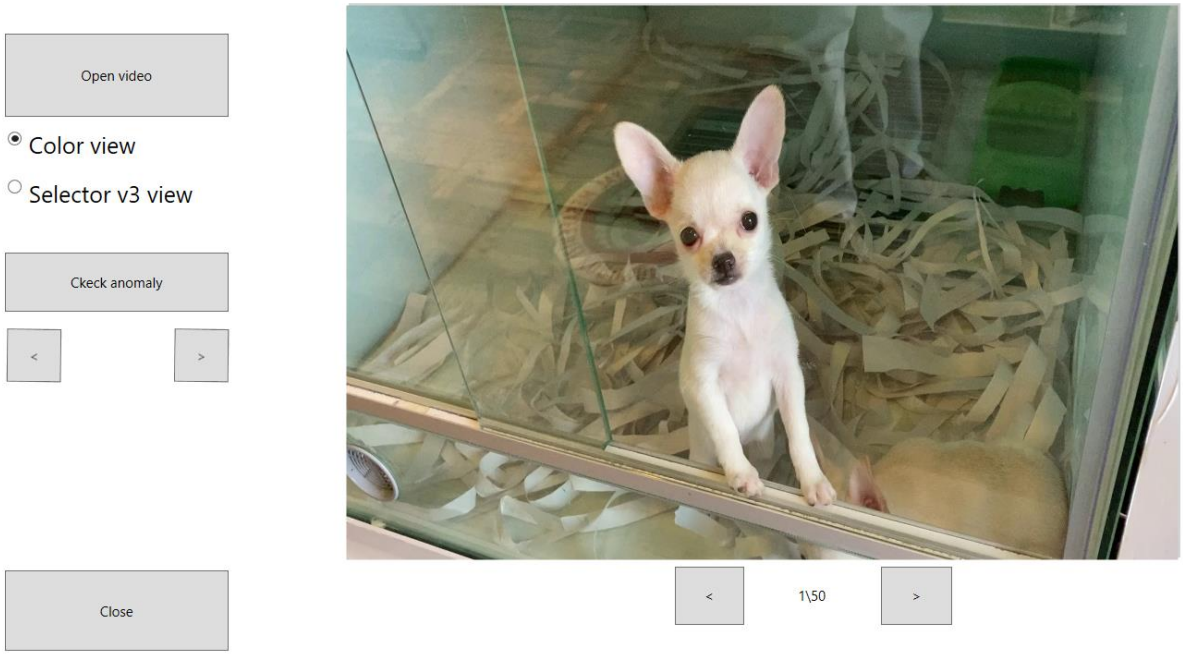

Рисунок 2.36 - Демонстрація роботи кольорового виводу зображення

2. селектор 3 виду – для перегляду відео через обробку кадру селектором.

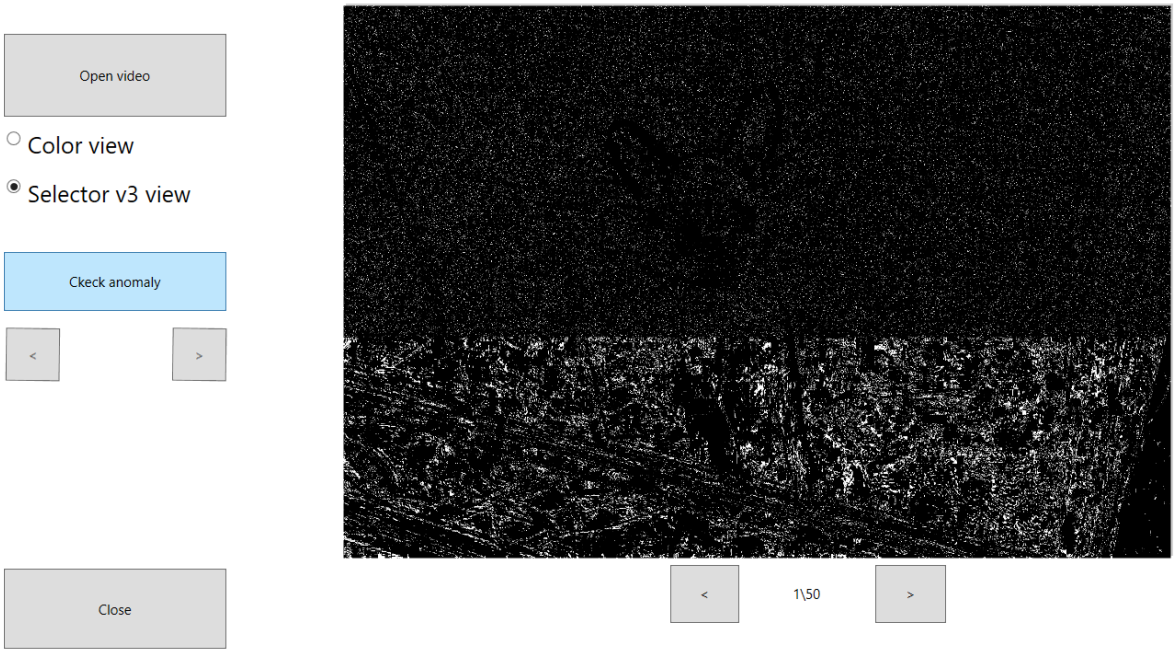

Рисунок 2.37 - Демонстрація впливу 3 селектору

Комбінування використання даних режимів може покращити аналіз відео файлів на оригінальність.

Після того як, користувач натисне на кнопку для перевірки (Рисунок 2.38) аномалії, програма почне перевірку. Це займає певний час. Чим більший відео файл тим більше часу займає перевірка.

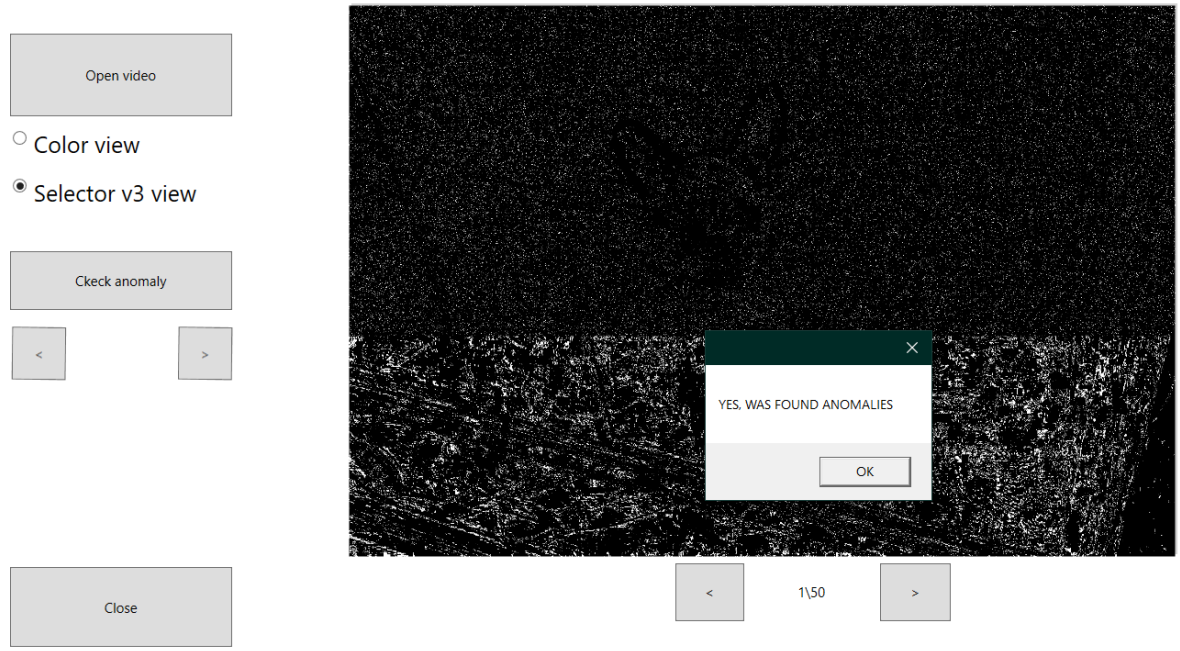

Рисунок 2.38 - Демонстрація результатів перевірки

У випадку знаходження аномалії, програма виведе на екран повідомлення про це, далі можна буде детальніше ознайомитися з вмістом файлу, а саме з його кадрами, які були помічені як аномальні.

Розділ 3 Проведення досліджень

3.1 Про аномалію

В межах даного дослідження я дослідів аномалію, яка проявляє себе, як зміна пікселів кадрів.

Відео аномалія - це певне відхилення або невідповідність у відео файлі, яке не відповідає очікуванням візуальним чи іншим характеристикам.

Аномалія може проявлятися у вигляді спотворення зображення, невідповідності пікселів, чи раптових зникнень, чи появи об’єктів. Подібні аномалії можуть виникати у результаті помилок в програмному забезпеченні, впливу зовнішніх факторів або навмисних дій.

Дане дослідження та розуміння відео аномалії відображеної у цьому дослідженні є важливим для забезпечення оригінальності й безпеки даних.

Аномалія, яку я досліджував, не є природньою або тою що виникла у результаті зовнішніх факторів, чи помилок у програмному забезпеченню. Дана аномалія може нести в собі корисне навантаження таку як текст, файли, чи просто байти даних.

Для створення даної аномалії були використані технології для роботи із вмістом відео файлу, а саме кадрами відео. Дана аномалія замінює корисним навантаженням останні біти у байті кольору згідно певного алгоритму. Таким чином, аномалія змінює колір пікселя. Для користувача візуально різниця не помітна, навіть якщо поставити поряд оригінальне і змінене зображення (Рисунок 3.1).

Рисунок 3.1 - Демонстрація різниці між не аномальним(ліворуч) й аномальним(праворуч) кадрами

Розмір аномалії залежить від обсягу інформації, який користувач хоче помістити в кадр відео. У досліджених зразках аномальна частина знаходилася у границях від 40% кадру до 100% кадру.

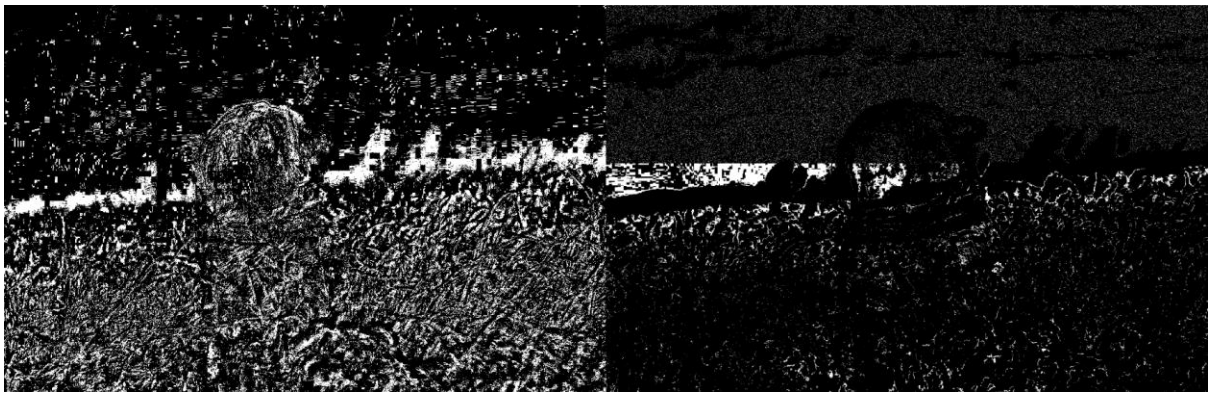

Рисунок 3.2 - Демонстрація заповнення аномалією(праворуч) простору

Дана аномалія має певні специфічні особливості, такі як, низька стійкість до пошкоджень, оскільки, популярні відео кодеки оптимізують розмір файлу змінюючи пікселі кадрів з ціллю зменшити розмір відео. В результаті роботи кодеків, потенційна інформація, збережена у кадрах може бути пошкодженою або повністю втраченою.

3.2 Дані для дослідження

Для проведення дослідження, було взято відкриті та вільні для використання відео, і та або, дані отримані на основі цих відео. Всього було

використано 19 відео. В якості аномалії: алгоритм вставлення користувацької інформації у відео файли.

3.3 Дослідження 1: Мультикласова класифікація з 2 класами

В рамках дослідження аномалії я, вирішив дослідити можливість визначення аномальних частин відео використовуючи можливості ML.NET та тренер для мультикласової класифікації. Оскільки, ML.NET не підтримує з коробки роботу із відео файлами, то було вирішено розділити відео на набір кадрів та подальші дії вже застосовувати для них.

Основною ідеєю цього дослідження було перевірити гіпотезу, що зробивши певні дії між кадрами відео, можна перевірити зразок на оригінальність.

Для початку, я вирішив спробувати використати звичайну різницю між 2 послідовними кадрами.

3.3.1 Індикатор 1: різниця між 2 кадрами

В основі ідеї для першого індикатору було використати абсолютну різницю між 2 кадрами, як потенційний показник аномальності, оскільки, 2 послідовних кадри в нормальних і реальних умовах скорше за все не будуть значно відрізнятися один від одного (Рисунок 3.3).

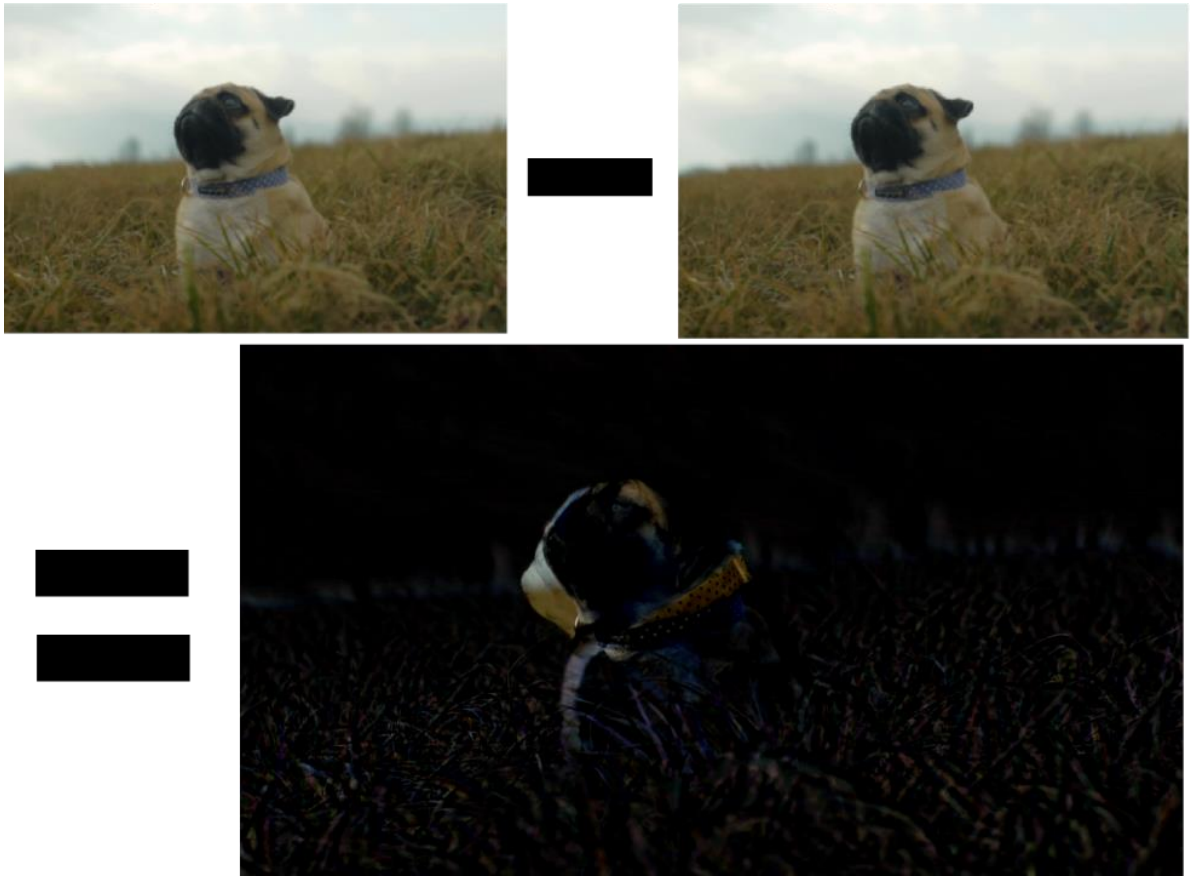

Рисунок 3.3 Демонстрація роботи 1 індикатору(селектору)

На фото можна побачити різницю між двома послідовними кадрами. Оскільки, за час між цими кадрами положення неба майже не змінилося, воно переважно чорне, тобто майже однакове для обох кадрів. Пес за цей час трохи змінив своє положення, тому, на фінальному фото його можна чітко побачити. Через вітер, що змінив положення трави між кадрами, її теж, можна візуально спостерігати на фінальному фото.

3.3.2 Набір даних

Для створення набору даних, відео були розділені на 2 частини. Перша частина відео були відібрані для тренування (14 штук), а друга для оцінювання (5 штук). Відео та подальші дані для тренування були, окремо, збережені від даних і відео для оцінювання.

З кожного відео бралися пари кадрів, які знаходилися на певній відстані один від одного, щоб створити більш різноманітні зразки із даними.

Потім, з усіх відео були відібрані пари кадрів, з яких було утворено зразки даних без аномалій використовуючи індикатор (Рисунок 3.4). Оскільки, для використання класифікації потрібно створити категорії до яких мережа буде відносити ті, чи інші дані, то не аномальні зразки були поміщені у категорію Original (Рисунок 3.4) (Рисунок 3.6).

Рисунок 3.4 - Демонстрація роботи індикатору на аномальному зразку

Для створення аномальних зразків, я обрав ті самі кадри, тільки в перший, чи в другий, чи в 2 одразу кадри було вставлено аномалію. Потім до цих кадрів застосували індикатор. Для аномальних даних назва була Inserted (Рисунок 3.5).

Рисунок 3.5 - Демонстрація структури набору даних

Оскільки, з кожного відео бралися однакові кадри, тобто, різниця між оригінальним зразком і аномальним це наявність аномалії. Це було зроблено для того, щоб покращити якість навчання мережі, оскільки, це б допомогло краще виділити аномальні частини на фоні решти даних. Таким чином я хотів навчити мережу краще бачити аномалії та її специфічний вплив на зразок.

Рисунок 3.6 - Демонстрація роботи індикатору для аномального зразку

Для навчання, всього я створив створено 1383 зразки даних. З них, до аномальних було віднесено 672, а до оригінальних 711 зразки з фреймів різних відео .

Для оцінювання 264, з них 132 аномальних й 132 оригінальних.

Співвідношення тренувальної частини до частини з оцінюванням ~80%.

3.3.3 Тренування і оцінювання

У якості тренера було обрано один із тренерів екосистеми .NET, її модуля ML.NET, а саме: MulticlassClassification. Особливістю даного тренера є використання вже навченої моделі ResNet-50. Після того як мережі було віддано для тренування навчальну частину сету, а для оцінювання

використано тестову частину набору даних, я отримав Micro-Accuracy - 0.83 - тобто мережа доволі впевнена в своєму виборі в тому до якого класу віднести зразок. Macro-Accuracy - 0.81 - тобто мережа доволі правильно навчилася відносити зразки до відповідних категорій.

Після отриманих результатів я вирішив, що отримані результати мене не влаштовують і я вирішив покращити їх. Результатом моїх трудів стало 2 версія індикатора.

3.3.4 Індикатор 2: Додаткове виділення пікселів

Ідея створення другого індикатора прийшла мені після отримання результатів навчання першого індикатора. Я вирішив спробувати покращити вже існуючий індикатор, додавши до нього функцію виділення пікселів за певних умов. Глибше досліджуючи аномалію, я помітив, що в аномальних зразках значення оригінального пікселя і значення зміненого пікселя відрізняються на три або два біти. Тому я вирішив додатково виділяти такі пікселі (Рисунок 3.7).

Ідея полягала в тому, щоб краще підсвітити для мережі й людського ока аномальну частину зразка. Я взяв вже створений набір даних й пропустив його через другий індикатор. У цьому індикаторі я додатково перевірив значення каналів пікселя на відповідність моїм вимогам. Для більш зручного представлення будемо вважати, що три біти дорівнюють числу 8, а два біти числу 4. Я перевіряв, щоб значення каналів пікселя було в необхідному мені діапазоні.

Якщо червоний канал не дорівнював нулю і був менше 8, зелений канал був менше чотирьох і не дорівнював нулю, а синій канал був менше за 8 і також не дорівнював нулю, я змінював колір цього пікселя на білий (255, 255, 255). Таким чином, я використав цей індикатор для кожного зразка в наборі даних й зберіг зміни.

Рисунок 3.7 - Різниця між 2 зразками за використання 2 індикатору

3.3.5 Тренування й оцінювання

Дані із зміненого набору даних були передані для навчання й оцінювання.

В результаті навчання було отримано Micro-Accuracy - 0.88 та Macro-Accuracy - 0.92.

Тобто вдалося покращити точність класифікації мережею даних й вірно класифікувати 92% тестових зразків. Отримані результати мене не влаштували й я вирішив ще попрацювати над даними для покращення зразків.

3.3.6 Індикатор 3: Додаткове затемнення пікселів

Ідея створення третього індикатора прийшла мені після розробки другого. Якщо в другому індикаторі я лише підсвічував пікселі, які відповідали певним умовам, в третьому я вирішив затемнити всі інші пікселі, що не відповідали умові.

Для цього, я додав додаткову умову: якщо перша умова не виконується, колір пікселя змінюється на чорний (0, 0, 0).
Для створення нового набору даних я використав уже створений набір після першого індикатору. Я пропустив зразки даних із першого індикатора через третій. В результаті, зразки, які були оброблені третім індикатором, збереглися для подальшого використання у навчанні та оцінюванні мережі (Рисунок 3.8).

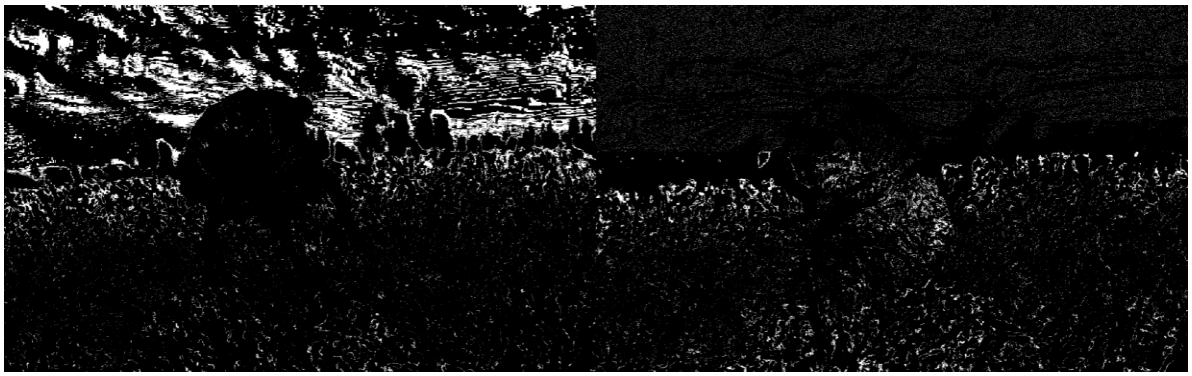

Рисунок 3.8 - Різниця між зразками за використання 3 індикатору

3.3.7 Тренування й оцінювання

Провівши навчання, а потім оцінку мережі на основі тестових даних, я отримав Micro-Accuracy - 0.94 й Macro-Accuracy - 0.95

Знову було значно краще навчено модель. Використання покращеного індикатору дозволило значно підвищити якість навчання мережі.

Додатково вдалося значно покращити візуальне відображення аномалії. На цьому я вирішив закінчити покращення даних, та завершити дослідження, оскільки, отримані результати близькі до еталонних.

3.3.8 Висновки

Під час проведення цього дослідження, я дослідив аномалію та те, як вона проявляється в даних. Для цього використовувався тренер MulticlassClassification із ML.NET, що дозволило з високою ймовірністю класифікувати аномальні та не аномальні зразки даних. Після використання третьої, останньої версії індикатора, мережу було навчено та оцінено. У результаті мережа змогла з високою ймовірністю правильно відносити дані до відповідних категорій.

3.4 Дослідження 2: Використання AnomalyDetection

Після завершення проведення першого дослідження, я вирішив спробувати знайти більш ефективне в плані ресурсів рішення. Для проведення 2 дослідження, я обрав тренер AnomalyDetection з екосистеми ML.NET. - це єдиний тренер в цій екосистемі, що не потребує проведення навчання\оцінювання, оскільки, спеціально був розроблений для пошуку аномалій в даних. В рамках цього дослідження я використав два способи представлені тренером для визначення оригінальності.

Оскільки, для використання функцій тренера дані мають бути рівномірними, щоб можна було виявити та визначити аномалії, жоден із попередніх способів представлення даних у незміненому вигляді не підходить для використання. Для розв'язування цієї проблеми я детальніше проаналізовані дані. Для аналізу були обрані зразки, оброблені третім індикатором, оскільки вони показали себе найкраще в попередньому дослідженні та найбільше пристосовані для візуального спостереження аномалій.

Я дослідив, що аномалія для нормальних даних, як правило, завжди темніша за не аномальні частини зразка. Тому було вирішено перетворити зображення у масив середньої яскравості кожного рядка. Для цього був використаний наступний алгоритм (Рисунок 3.9).

```
public float[] Get(Mat mat)
{
    var intensities = new float[mat.Height];

    for (int y = 0; y < mat.Height; y++)
    {
        float lineIntensity = 0;

        for(int x = 0;x<mat.Width; x++)
        {
            var pixel = mat.Get<Vec3b>(y, x);

            var intensity = (pixel.Item0 + pixel.Item1 + pixel.Item2) / 3f;

            lineIntensity += intensity;
        }

        intensities[y] = lineIntensity / mat.Width;
    }

    return intensities;
}
```

Рисунок 3.9 - Алгоритм для отримання масиву середньої яскравості рядку

Після перетворення зразку із набору даних в масив середньої яскравості рядків, масив був відображений на графіках (Рисунок 3.10).

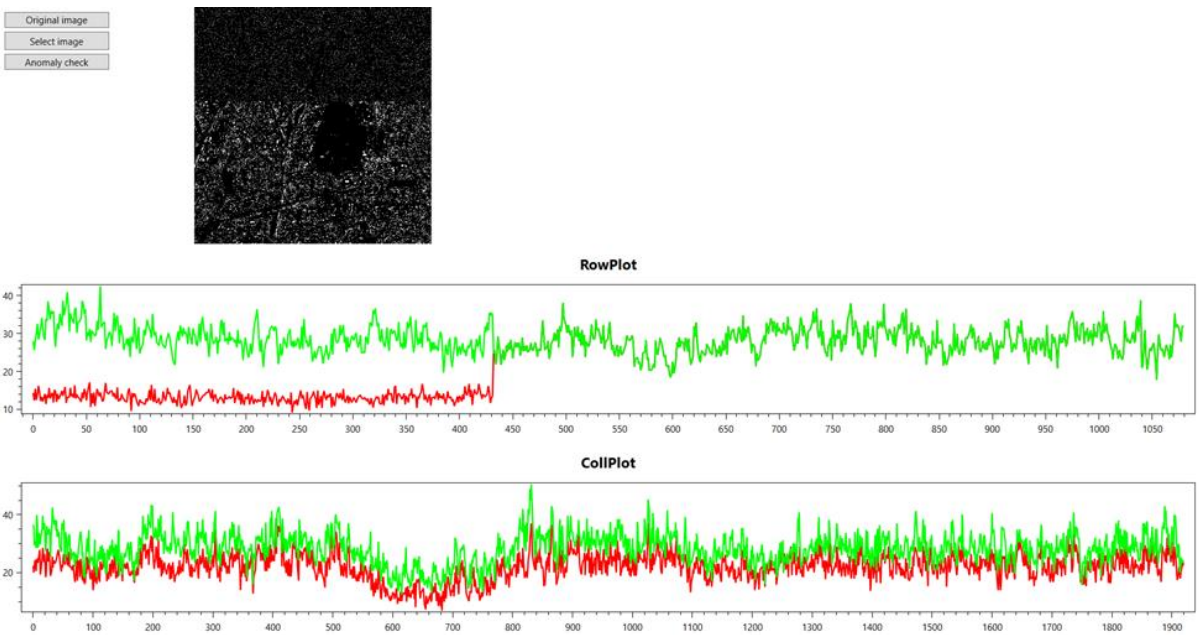

Рисунок 3.10 - Аналіз зображення з використанням розробленого алгоритму

На даних графіках можна побачити, як впливає аномалія на середню яскравість рядку.

Зеленим відображено значення в не аномальному зразку, а червоним значенням в аномальному. Як можна побачити відхилення значне (RowPlot), а також аномалія знаходиться в певному діапазоні. У мене виникла ідея

перевірити чи у всіх зразків аномалія веде себе подібним чином (Рисунок 3.11).

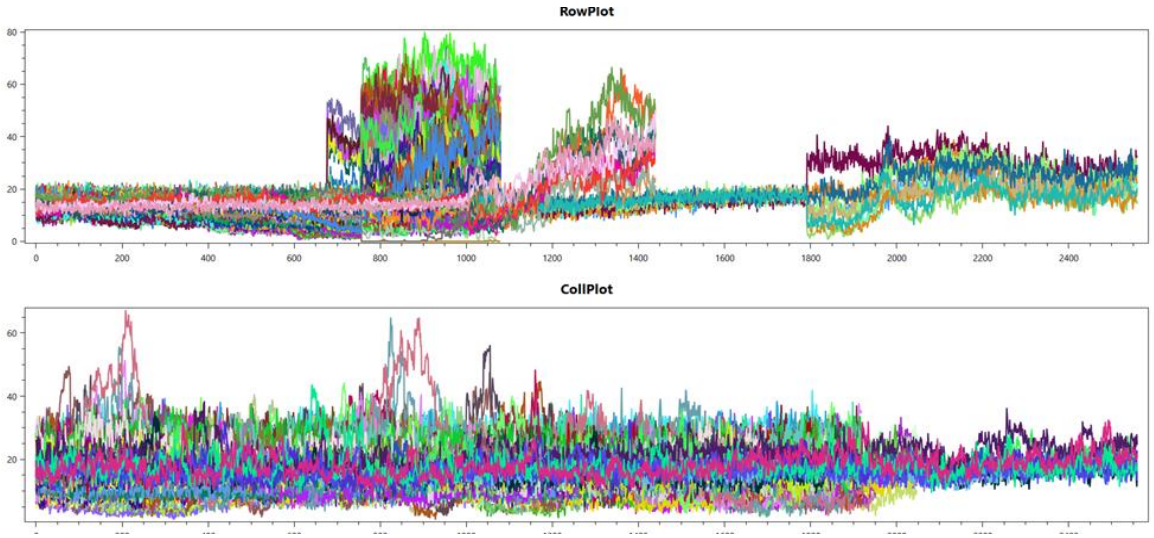

Рисунок 3.11 - Графіки для всіх аномальних зразків даних

Як можна побачити по графікам значення яскравості рядків для аномальних частин зразків коливаються у діапазоні від 3 до 25 одиниць яскравості і не сильно коливається. На основі цих результатів я вирішив спробувати використати пошук аномалій.

3.4.1 Тренування й оцінювання

Оскільки, даний тренер не підтримує навчання тому розбивати дані на частину для тренування і оцінювання не було потреби. Тому я обрав декілька зразків, які були перетворені на масиви середньої яскравості рядків та були передані до для визначення піків (Рисунок 3.12).

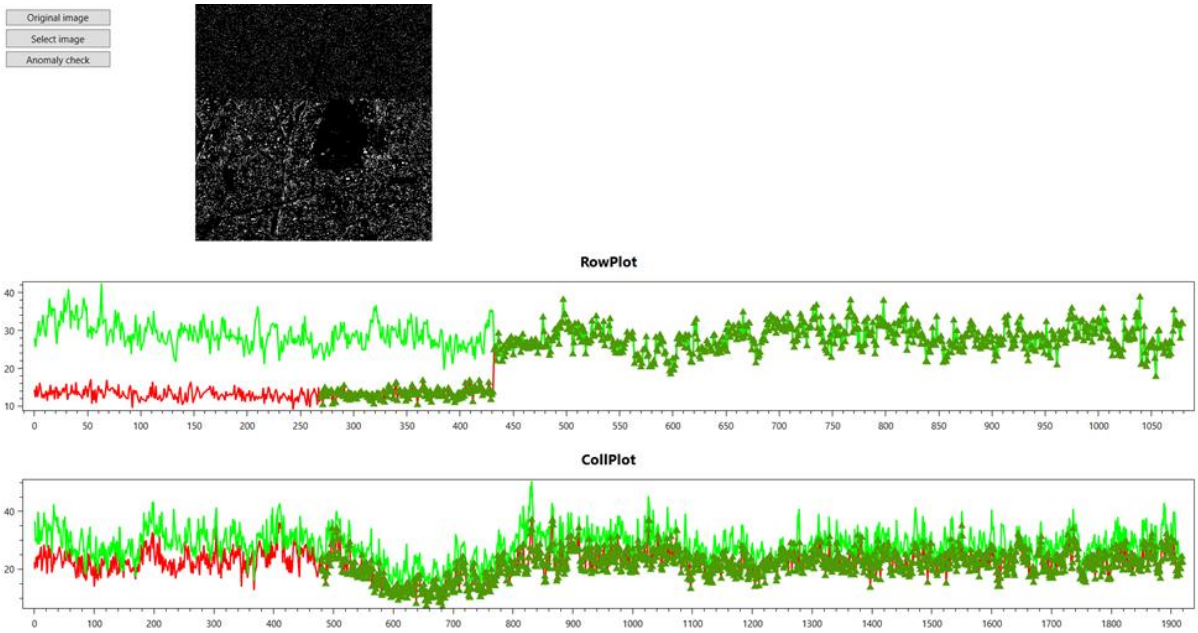

Рисунок 3.12 - Використання DetectSpikes

Хоча, пошук піків не спрацював, оскільки, дані не є рівномірно розподіленими, я вирішив скористатися другим методом тренера, а саме точкою переходу. На графіку (Рисунок 3.13) точку зміни між аномальною частиною і не аномальною частиною чітко видно, додатково виділено синім.

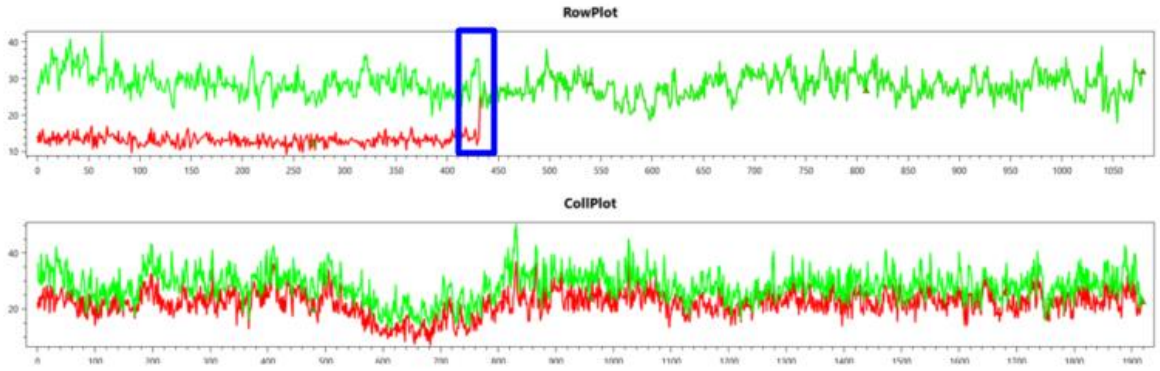

Рисунок 3.13 - Зображення точки де очікується позначення переходу

В результаті роботи тренеру та методу пошуку точки переходу чітких результатів теж отримано не було (Рисунок 3.14).

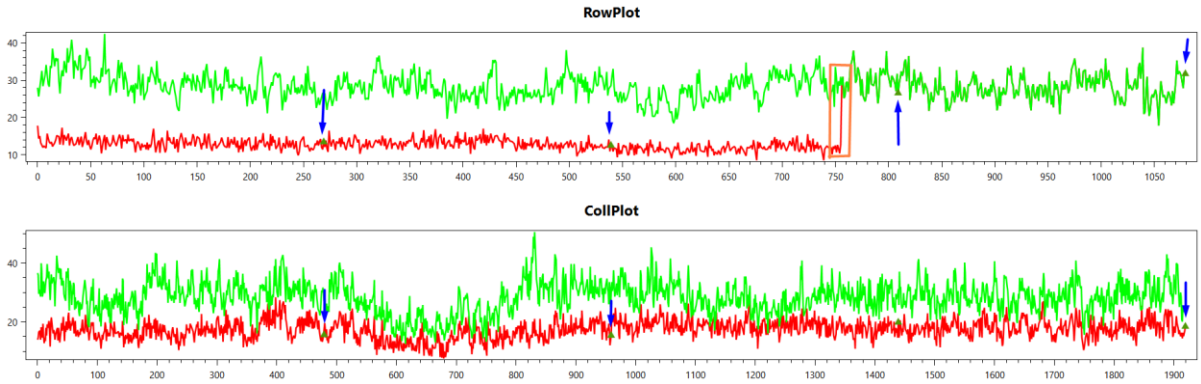

Рисунок 3.14 - Реально отримані точки переходу

Як можна побачити на графіку, точки зміни були визначені неправильно. На зображенні для кращої видимості вони були підсвічені синіми стрілками, коли, як вони очікувалися в діапазоні переходу що виділений оранжевим прямокутником.

3.4.2 Висновки

Використання тренеру пошук аномалій з ML.NET не дало очікуваних результатів. Цей результат міг бути пов'язаний з нерівномірністю використаних даних. Було використано два методи тренеру: пошук піків та пошук переходів. 2 методи не дозволили вірно перевірити оригінальність ресурсів.

3.5 Дослідження 3: Логістична функція

У даному дослідженні я вирішив використати логістичну функцію для визначення аномальних і оригінальних кадрів. Логістична функція, як відомо з назви має повернути значення 1 чи 0, де 0 - відсутність аномалії, 1 - її наявність у зразку. На основі досліджених даних 2 дослідженні, я помітив, що аномальні дані знаходяться у певному діапазоні й вирішив спробувати визначати аномалію на основі аналізу середньої яскравості рядків кадрів відео.

Під час обробки даних я вирішив рахувати інтенсивність кольорів кожного пікселя, оскільки, аномалія змінює шум в відео. Цей шум я визначав використовуючи формулу для рахування інтенсивності пікселя використану у пошуку аномалії. Для аналізу я використовував кадри, що були оброблені за допомогою 3 селектору, оскільки, саме завдяки цьому селектору мені вдалося найкраще підсвітити аномалію у мультикласовій класифікації.

Рисунок 3.15 - Приклад оригінального зразку що був позначений як аномальний

Однак, під час використання даної функції я зіткнувся з проблемою закладеною в основу селектора, а саме з неможливістю в абсолютно всіх випадках правильно підсвітити аномалію. Дана проблема виникала тоді, коли об’єкт у відео між 2 кадрами відео, сильно змінює своє положення, тоді такий об’єкт може відобразитися як чорна область.

Щоб краще зрозуміти цю проблему, я у 2 дослідженні, побудував графіки за допомогою бібліотеки OxyPlot. Додавши на графік дані з усіх зразків. Тоді я отримав наступні результати: майже всі аномальні частини зразків знаходилися у діапазоні від 3 до 25 одиниць інтенсивності і різниця між 2 послідовними числами інтенсивності була невеликою. Це наштовхнуло мене на використання цього діапазону, як критерію для логістичної функції. У функції, я вираховував середню інтенсивність для рядків зразку. Потім

шукав найдовший ланцюжок, що знаходиться в межах діапазону аномального значення. Якщо цей найдовший ланцюжок був більше 0.4(кількості стовпців кадру), то цей кадр є аномальний і функція повертає 1. Якщо менше то кадр не аномальний і повертає 0. Цей метод спрацював, але, він не є достатньо точним для обробки і оцінки достатньо складних аномалій.

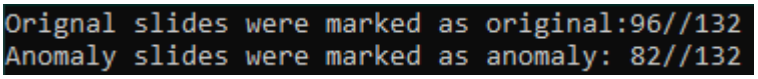

Рисунок 3.15 - Результати роботи логістичної функції

В результаті роботи алгоритму 72% оригінальних зразків було вірно оцінено, як не аномальні, а аномальних тільки 60% було оцінено вірно. Тобто точність використання даної функції була 66%.

Висновки

Під час виконання кваліфікаційної роботи рівня магістр, я дослідив аномалію й розробив шляхи для визначення оригінальності відео ресурсів.

Я дослідив можливість створення набору даних із впровадженням аномалії у дані. У якості даних було використано випадкові значення для імітації реальних зашифрованих даних. Для збереження даних було досліджено і впроваджено використання loseless алгоритмів стискання даних. Було обрано FFV1, через його швидкодію і одну з найкращих стискаючих можливостей серед інших подібних кодеків.

Було проведено дослідження з використанням ImageClassification з ML.NET. Дане дослідження було проведено з цілю навчити мережу розрізняти аномальні і не аномальні зразки даних. Провівши череду покращень, мережу було успішно навчено відрізняти аномальні від не аномальних зразків та досягнуто точності у 0.95 для тестових даних.

Було проведено дослідження з використанням AnomalyDetection з ML.NET з ціллю перевірити можливість використання даного тренеру для визначення аномальності зразків даних. Хоч і використання даного тренеру не дало очікуваних результатів, проте, було проведено більш детальний аналіз даних, частину результатів якого було використано в подальшому.

На основі попередніх результатів дослідження AnomalyDetection, було проведено ще одне дослідження, ціль якого було створити логістичну функцію для визначення оригінальних і аномальних зразків даних. Дане дослідження дало результати, оскільки, аномальна частина зразку і оригінальна, як правило, значно відрізнялися. Після калібрування параметрів функції для отримання найкращого результату, було отримано точність у 66%.

Оскільки, точність функції менша за точність роботи мережі, для впровадження у додаток було використано результати 1 дослідження.

Було розроблено додаток для демонстрації результатів найкращого із досліджень.

Список використаних джерел

1. ML dataset creation [Електронний ресурс] - https://devblogs.microsoft.com/dotnet/object-detection-ml-dotnet-model-builder/
2. AnomalyDetection ML.NET [Електронний ресурс] - https://learn.microsoft.com/en-us/dotnet/machine-learning/tutorials/sales-anomaly-detection
3. ImageClassification ML.NET [Електронний ресурс] - https://learn.microsoft.com/en-us/dotnet/api/microsoft.ml.vision.imageclassificationtrainer?view=ml-dotnet
4. Логістична функція [Електронний ресурс] - https://en.wikipedia.org/wiki/Logit
5. WPF .NET [Електронний ресурс] - https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-9.0
6. OpenCvSharp4 з OpenCv для .NET [Електронний ресурс] - https://github.com/shimat/opencvsharp
7. OpenCv [Електронний ресурс] - https://docs.opencv.org/4.x/index.html
8. Codec FFV1 [Електронний ресурс] - https://trac.ffmpeg.org/wiki/Encode/FFV1
9. Графіки в WPF [Електронний ресурс] - https://oxyplot.github.io/

Додатки

MLPrep.V3Selector.cs

```
using OpenCvSharp;
using PixFlow.Image.Filters.Interfaces;
using PixFlow.Image.Filters.Services;

namespace MyML.DatasetPreparator.CustomHighlighters
{
    public class V3Selector : IHighlighter
    {
        private static readonly Diff _diff = new();

        public Mat Highlight(Mat frame1, Mat frame2)
        {
            if (frame1.Empty() || frame2.Empty())
            {
                throw new ArgumentException("One or both frames are empty.");
            }

            var dif = _diff.Highlight(frame1, frame2);

            for (int y = 0; y < dif.Size().Height; y++)
            {
                for (int x = 0; x < dif.Size().Width; x++)
                {
                    var pixel = dif.Get<Vec3b>(y, x);

                    if (pixel.Item0 < 8 && pixel.Item0 != 0 &&
                        pixel.Item1 < 4 && pixel.Item1 != 0 &&
                        pixel.Item2 < 8 && pixel.Item2 != 0)
                    {
                        dif.Set(y, x, new Vec3b(255, 255, 255));
                    }
                    else
                    {
                        dif.Set(y, x, new Vec3b(0, 0, 0));
                    }
                }
            }

            return dif;
        }
    }
}
```

MLPrep.IBytesProvider.cs

```
namespace MyML.DatasetPreparator.DataProviders.Interfaces
{
    public interface IBytesProvider
    {
        IEnumerable<byte> Get(int count);
        IEnumerable<byte> Get();
    }
}
```

MLPrep.RandomBytesProvider.cs

```
using MyML.DatasetPreparator.DataProviders.Interfaces;

namespace MyML.DatasetPreparator.DataProviders.Services
{
    public sealed class RandomBytesProvider : IBytesProvider
    {
        private Random _random = new Random();
```

```
        public IEnumerable<byte> Get(int count)
        {
            if (count <= 0)
            {
                throw new ArgumentOutOfRangeException(nameof(count), "Count must
be greater than zero.");
            }

            var bytes = new byte[count];

            _random.NextBytes(bytes);

            return bytes;
        }

        public IEnumerable<byte> Get()
        {
            yield return (byte)_random.Next(0, 255);
        }
    }
}
```

MLPrep.TextBytesProvider.cs

```
using MyML.DatasetPreparator.DataProviders.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MLPrep.DatasetPreparator.DataProviders.Services
{
    public class TextBytesProvider : IBytesProvider
    {
        private readonly string _text;

        private readonly Encoding _encoding;

        private int _index = 0;
        public TextBytesProvider(string text, Encoding encoding)
        {
            ArgumentException.ThrowIfNullOrEmpty(text, nameof(text));
            ArgumentException.ThrowIfNullOrWhiteSpace(text, nameof(text));

            _text = text;
            _encoding = encoding;
        }

        public IEnumerable<byte> Get(int count)
        {
            var res = _encoding.GetBytes(_text, _index, count);

            _index += count;

            return res;
        }

        public IEnumerable<byte> Get()
        {
            return _encoding.GetBytes(_text);
        }
    }
}
```

MLPrep.Cutter.cs

```
using PixFlow.Video.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyML.DatasetPreparator
{
    sealed public class Cutter
    {
        public static void Cut(string directory,string outDirectory, int
frameAmount)
        {
            foreach (var file in Directory.GetFiles(directory))
            {
                VideoCutter.Cut(file, Path.Combine(outDirectory,
$"{Path.GetFileNameWithoutExtension(file)};cut.avi"), frameAmount);
            }
        }
    }
}
```

MLPrep.Preparator.cs

```
using MyML.DatasetPreparator.DataProviders;
using MyML.DatasetPreparator.DataProviders.Interfaces;
using OpenCvSharp;
using PixFlow.Image.Filters.Interfaces;
using PixFlow.Image.Models;
using PixFlow.Pixel.Interfaces;
using PixFlow.Video.Helpers;
using PixFlow.Video.Interfaces;
using PixFlow.Video.Models;
using PixFlow.Video.Services;

namespace MyML.DatasetPreparator
{
    public class Preparator : IDisposable
    {
        private readonly bool _deleteTempVideoFiles;

        private readonly string _original;

        private string? _lastOperatedVideo;

        private readonly List<string> _filesToDelete = new List<string>();

        public Preparator(string fileName, bool deleteTempVideoFiles = true)
        {
            _original = fileName;

            _lastOperatedVideo = _original;

            _deleteTempVideoFiles = deleteTempVideoFiles;
        }

        public Preparator Insert(IPixelEditable editor, IBytesProvider provider,
double frameInserPercent, int[]? pattern = null)
        {
            var operations = GenerateOperations(_lastOperatedVideo == null ?
_original :
                _lastOperatedVideo, editor, provider, frameInserPercent, pattern);
```

```
            return Insert(operations, frameInserPercent);
        }

        public Preparator Insert(IEnumerable<VideoInsertOperation> operations,
double frameInsertPercent)
        {
            var inserter = new VideoInserter();
            var file = _lastOperatedVideo == null ? _original :
_lastOperatedVideo;
            var insertFile = $"{Path.GetFileNameWithoutExtension(file)};inserted
{frameInsertPercent:0.0}.avi";
            File.Copy(file, insertFile, true);

            inserter.Insert(insertFile, operations);
            _lastOperatedVideo = insertFile;
            _filesToDelete.Add(file);

            return this;
        }

        public Preparator Highlight(IHighlighter highlighter)
        {
            if (!File.Exists(_lastOperatedVideo))
            {
                throw new FileNotFoundException("Insert file not found.",
_lastOperatedVideo);
            }

            var highlightFile =
$"{Path.GetFileNameWithoutExtension(_lastOperatedVideo)};highlighted.avi";

            _filesToDelete.Add(highlightFile);

            using var reader = new VideoCapture(_lastOperatedVideo);
            using var writer = new VideoWriter(highlightFile, Codecs.FFV1,
reader.Fps, new Size(reader.FrameWidth, reader.FrameHeight));

            using var frame1 = new Mat();
            using var frame2 = new Mat();

            reader.Read(frame1);
            while (reader.Read(frame2))
            {
                using var highlightedFrame = highlighter.Highlight(frame1,
frame2);
                writer.Write(highlightedFrame);
                Cv2.CopyTo(frame2, frame1);
            }
            _lastOperatedVideo = highlightFile;

            return this;
        }

        public Preparator HighlightByPattern(IHighlighter highlighter, int[]
pattern)
        {
            if (!File.Exists(_lastOperatedVideo))
            {
                throw new FileNotFoundException("Insert file not found.",
_lastOperatedVideo);
            }

            var highlightFile =
$"{Path.GetFileNameWithoutExtension(_lastOperatedVideo)};highlighted.avi";

            _filesToDelete.Add(highlightFile);
```

```
            using var reader = new VideoCapture(_lastOperatedVideo);
            using var writer = new VideoWriter(highlightFile, Codecs.FFV1,
reader.Fps, new Size(reader.FrameWidth, reader.FrameHeight));

            using var frame1 = new Mat();
            using var frame2 = new Mat();

            reader.Read(frame1);
            int i = 0;
            while (reader.Read(frame2))
            {
                var patternIteration = pattern[i % pattern.Length];
                if (patternIteration == 1)
                {
                    using var highlightedFrame = highlighter.Highlight(frame1,
frame2);
                    writer.Write(highlightedFrame);
                    Cv2.CopyTo(frame2, frame1);
                }
                i++;
            }
            _lastOperatedVideo = highlightFile;

            return this;
        }

        public Preparator ToImages(ISplitable spliter, string outDirectory)
        {
            if (!File.Exists(_lastOperatedVideo))
            {
                throw new FileNotFoundException("Video file not found.",
_lastOperatedVideo);
            }

            if (!Directory.Exists(outDirectory))
            {
                Directory.CreateDirectory(outDirectory);
            }

            int counter = 1;

            foreach (var mat in spliter.Split(_lastOperatedVideo))
            {
                var imagePath = Path.Combine(outDirectory,
$"{Path.GetFileNameWithoutExtension(_lastOperatedVideo)};{counter++}.png");

                mat.SaveImage(imagePath);
            }

            return this;
        }

        private static List<VideoInsertOperation> GenerateOperations(
            string videoPath,
            IPixelEditable pixelEditor,
            IBytesProvider provider,
            double insertPercent,
            int[]? pattern)
        {
            var operations = new List<VideoInsertOperation>();
            var videoParams = VideoParams.Get(videoPath);

            var bytes = provider.Get();

            Rect rect;
```

```
            bool foundEndOfPattern = false;

            if (pattern == null)
            {
                pattern = [1];
            }

            for (int i = 0; i < videoParams.FrameCount; i++)
            {
                var patternIteration = pattern[i % pattern.Length];

                if (foundEndOfPattern)
                {
                    rect = new Rect(0, 0, 0, 0);
                }
                else
                {
                    rect = patternIteration switch
                    {
                        1 => new Rect(0, 0, videoParams.FrameSize.Width,
(int)(videoParams.FrameSize.Height * insertPercent)),
                        0 => new Rect(0, 0, 0, 0),
                        -1 => StopPattern(ref foundEndOfPattern),
                        _ => throw new ArgumentException($"Unsupported pattern
iteration {patternIteration}")
                    };
                }

                operations.Add(new VideoInsertOperation(i, new[] { new
InsertOperation(bytes, pixelEditor, rect) }));
            }

            return operations;
        }

        private static Rect StopPattern(ref bool foundEndOfPattern)
        {
            foundEndOfPattern = false;
            return new Rect(0, 0, 0, 0);
        }

        public void Dispose()
        {
            if (_deleteTempVideoFiles)
            {
                _filesToDelete.ForEach(f => { if (File.Exists(f)) File.Delete(f);
});
            }
        }
    }
}
```

MLPrep.Rotator.cs

```
using OpenCvSharp;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyML.DatasetPreparator
{
    internal class Rotator
    {
        public static void Rotate(string inDirectory, string outDirectory, int[]
angles)
```

```
        {
            foreach (var file in Directory.GetFiles(inDirectory, "*.png",
SearchOption.AllDirectories))
            {
                foreach (var angle in angles)
                {
                    using var mat = new Mat(file);

                    var rotated = PixFlow.Image.Services.Rotator.Rotate(mat,
angle);

                    var path = $"{Path.Combine(outDirectory,
$"{Path.GetFileNameWithoutExtension(file)};rotated {angle}.png")}";

                    rotated.SaveImage(path);
                }

            }
        }
    }
}
```

PixFlow.IHighlighter.cs

```
using OpenCvSharp;

namespace PixFlow.Image.Filters.Interfaces
{
    public interface IHighlighter
    {
        public Mat Highlight(Mat frame1, Mat frame2);
    }
}
```

PixFlow.IInsertable.cs

```
using PixFlow.Image.Models;
using OpenCvSharp;

namespace PixFlow.Image.Interfaces
{
    public interface IInsertable
    {
        public void Insert(Mat frame, IEnumerable<InsertOperation> operations);
    }
}
```

PixFlow.InsertOperation.cs

```
using PixFlow.Pixel.Interfaces;
using OpenCvSharp;

namespace PixFlow.Image.Models
{
    public record InsertOperation
    {
        public IEnumerable<byte> Bytes { get; init; }

        public IPixelEditable Editor { get; init; }

        public Rect Position { get; init; }

        public InsertOperation(IEnumerable<byte> bytes, IPixelEditable editor,
Rect position)
        {
            ArgumentNullException.ThrowIfNull(bytes, nameof(bytes));
```

```
            ArgumentNullException.ThrowIfNull(editor, nameof(editor));

            Bytes = bytes;

            Editor = editor;

            Position = position;
        }

    }
}
```

PixFlow.Diff.cs

```
using PixFlow.Image.Filters.Interfaces;
using OpenCvSharp;

namespace PixFlow.Image.Filters.Services
{
    public sealed class Diff : IHighlighter
    {
        public Mat Highlight(Mat frame1, Mat frame2)
        {
            #region checks
            ArgumentNullException.ThrowIfNull(frame1, nameof(frame1));

            ArgumentNullException.ThrowIfNull(frame2, nameof(frame2));

            if (frame1.Size() != frame2.Size())
            {
                throw new ArgumentException("Input frames must be of the same
size.");
            }
            #endregion

            return frame1 - frame2;
        }
    }
}
```

PixFlow.ImageInserter.cs

```
using PixFlow.Image.Interfaces;
using PixFlow.Image.Models;
using OpenCvSharp;

namespace PixFlow.Image.Services
{
    public sealed class ImageInserter : IInsertable
    {
        public void Insert(Mat frame, IEnumerable<InsertOperation> operations)
        {
            ArgumentNullException.ThrowIfNull(frame, nameof(frame));

            ArgumentNullException.ThrowIfNull(operations, nameof(frame));

            if (!operations.Any())
            {
                throw new ArgumentException("Operations cannot be empty.");
            }

            foreach (var operation in operations)
            {
                int setBytes = 0;
```

```
                for (int y = operation.Position.Y; y < operation.Position.Y +
operation.Position.Height; y++)
                {
                    for (int x = operation.Position.X; x < operation.Position.X +
operation.Position.Width; x++)
                    {
                        var pixel = frame.Get<Vec3b>(y, x);
                        var bytes =
operation.Bytes.Take(operation.Editor.BytesCountForPixel).ToArray();

                        if (bytes.Length != operation.Editor.BytesCountForPixel)
                        {
                            throw new ArgumentException("Byte array size does not
match the expected size.");
                        }

                        frame.Set(y, x, operation.Editor.Set(pixel, bytes));

                        setBytes += operation.Editor.BytesCountForPixel;
                    }
                }
            }
        }
    }
}
```

PixFlow.Rotator.cs

```
using OpenCvSharp;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Formats.Asn1.AsnWriter;

namespace PixFlow.Image.Services
{
    public static class Rotator
    {
        public static Mat Rotate(Mat image, int angle)
        {
            var center = new Point2f(image.Width / 2f, image.Height / 2f);

            Mat rotationMatrix = Cv2.GetRotationMatrix2D(center, angle, 1.0);

            double cos = Math.Abs(rotationMatrix.Get<double>(0, 0));
            double sin = Math.Abs(rotationMatrix.Get<double>(0, 1));

            int newWidth = (int)(image.Height * sin + image.Width * cos);
            int newHeight = (int)(image.Height * cos + image.Width * sin);

            rotationMatrix.Set(0, 2, rotationMatrix.Get<double>(0, 2) + (newWidth
/ 2) - center.X);
            rotationMatrix.Set(1, 2, rotationMatrix.Get<double>(1, 2) + (newHeight
/ 2) - center.Y);

            Mat dst = new Mat();
            Cv2.WarpAffine(image, dst, rotationMatrix, new Size(newWidth,
newHeight));

            return dst;
        }
    }
}
```

PixFlow.IPixelEditable.cs

```
using OpenCvSharp;

namespace PixFlow.Pixel.Interfaces
{
    public interface IPixelEditable
    {
        public Vec3b Set(in Vec3b origin, in byte[] bytes);

        public byte[] Get(in Vec3b origin);

        public int BytesCountForPixel { get; }
    }

}
```

PixFlow.Changer3B2G3R.cs

```
using PixFlow.Pixel.Interfaces;
using OpenCvSharp;

namespace PixFlow.Pixel.Services
{
    public sealed class Changer3B2G3R : IPixelEditable
    {
        public int BytesCountForPixel => 1;

        public Vec3b Set(in Vec3b origin, in byte[] bytes)
        {
            if (bytes == null || bytes.Length < 1)
            {
                throw new ArgumentException("Invalid byte array provided.");
            }

            var newPixel = new Vec3b
            {
                Item0 = (byte)(origin.Item0 >> 3 << 3 | bytes[0] & 0b111), // B
                Item1 = (byte)(origin.Item1 >> 2 << 2 | bytes[0] >> 3 & 0b11), //
G
                Item2 = (byte)(origin.Item2 & 0b11111000 | bytes[0] >> 5 & 0b111)
// R
            };

            return newPixel;
        }

        public byte[] Get(in Vec3b origin)
        {
            byte data = 0;

            data |= (byte)(origin.Item0 & 0b00000111);
            data |= (byte)((origin.Item1 & 0b00000011) << 3);
            data |= (byte)((origin.Item2 & 0b00000111) << 5);

            return [data];
        }
    }
}
```

PixFlow.Codecs.cs

```
using OpenCvSharp;

namespace PixFlow.Video.Helpers
{
    public static class Codecs
    {
```

```
        public static FourCC FFV1 => FourCC.FromFourChars('F', 'F', 'V', '1');
    }
}
```

PixFlow.IInsertable.cs

```
using PixFlow.Video.Models;

namespace PixFlow.Video.Interfaces
{
    public interface IInsertable
    {
        public void Insert(string videoFilepath,
            IEnumerable<VideoInsertOperation> operations,
            bool useOriginalCodec = false);
    }
}
```

PixFlow.ISplitable.cs

```
using OpenCvSharp;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PixFlow.Video.Interfaces
{
    public interface ISplitable
    {
        IEnumerable<Mat> Split(string videoFile);
    }
}
```

PixFlow.VideoInsertOperation.cs

```
using PixFlow.Image.Models;

namespace PixFlow.Video.Models
{
    public record VideoInsertOperation
    {
        public int FrameIndex { get; init; }

        public IEnumerable<InsertOperation> Operations { get; init; }

        public VideoInsertOperation(int frameIndex, IEnumerable<InsertOperation>
operations)
        {
            ArgumentNullException.ThrowIfNull(operations, nameof(Operations));

            Operations = operations;

            if (frameIndex < 0)
            {
                throw new ArgumentOutOfRangeException("FrameIndex cannot be less
than 0", nameof(frameIndex));
            }

            FrameIndex = frameIndex;
        }
    }
}
```

PixFlow.OneFrameSpliter.cs

```
using OpenCvSharp;
using PixFlow.Video.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PixFlow.Video.Services
{
    public class OneFrameSpliter : ISplitable
    {
        public IEnumerable<Mat> Split(string videoFile)
        {
            var reader = new VideoCapture(videoFile);

            using var mat = new Mat();

            while (reader.Read(mat))
            {
                var copy = new Mat();

                mat.CopyTo(copy);

                yield return copy;
            }

            reader.Release();
            mat.Release();
        }
    }
}
```

PixFlow.VideoCutter.cs

```
using OpenCvSharp;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PixFlow.Video.Services
{
    public static class VideoCutter
    {
        public static void Cut(string inVideoFile, string outVideoFile, int
frameCount)
        {
            using var reader = new VideoCapture(inVideoFile);
            using var writer = new VideoWriter(
                outVideoFile, FourCC.FromString(reader.FourCC), reader.Fps,
                new Size(reader.FrameWidth, reader.FrameHeight));

            using var mat = new Mat();

            for (int i = 0; i < frameCount && reader.Read(mat); i++)
            {
                writer.Write(mat);
            }
        }
    }
}
```

PixFlow.VideoInserter.cs

```
using PixFlow.Video.Helpers;
using PixFlow.Video.Models;
using OpenCvSharp;

namespace PixFlow.Video.Services
{
    public sealed class VideoInserter(Image.Interfaces.IInsertable inserter =
null!) : Interfaces.IInsertable
    {
        private readonly Image.Interfaces.IInsertable _inserter = inserter ?? new
Image.Services.ImageInserter();

        public void Insert(string videoFilePath, IEnumerable<VideoInsertOperation>
operations, bool useOriginalCodec = false)
        {
            #region checks
            ArgumentException.ThrowIfNullOrEmpty(videoFilePath,
nameof(videoFilePath));

            ArgumentNullException.ThrowIfNull(operations, nameof(operations));

            if (!operations.Any())
            {
                throw new ArgumentException("Operations cannot be empty.");
            }
            #endregion

            using var reader = new VideoCapture(videoFilePath);
            if (!reader.IsOpened())
            {
                throw new IOException($"Failed to open video file:
{videoFilePath}");
            }

            string tempFilePath = Path.GetTempFileName() + ".avi";//only avi
support ffv1

            using var writer = new VideoWriter(
                tempFilePath,
                useOriginalCodec ? FourCC.FromString(reader.FourCC) : Codecs.FFV1,
                reader.Fps,
                new Size(reader.FrameWidth, reader.FrameHeight));

            var sorted = operations.OrderBy(x => x.FrameIndex);

            using var mat = new Mat();

            int writerCount = 0, readIndex = 0;
            foreach (var operation in sorted)
            {
                while (reader.Read(mat))
                {
                    if (readIndex == operation.FrameIndex)
                    {
                        _inserter.Insert(mat, operation.Operations);
                        if (operation.Operations.FirstOrDefault()!.Position != new
Rect(0, 0, 0, 0))
                        {
                            writer.Write(mat);
                            writerCount++;
                        }
                        readIndex++;
                        break;
                    }
                    writer.Write(mat);
                    readIndex++;
```

```
                }
            }

            reader.Release();
            writer.Release();

            File.Delete(videoFilePath);
            File.Move(tempFilePath, videoFilePath);
        }
    }
}
```

PixFlow.VideoOperations.cs

```
using OpenCvSharp;

namespace PixFlow.Video.Services
{
    public class VideoOperations
    {
        public static void SetFps(string filePath, double fps)
        {
            ArgumentException.ThrowIfNullOrEmpty(filePath, nameof(filePath));

            if (fps <= 0)
            {
                throw new ArgumentOutOfRangeException(nameof(fps), "FPS must be
greater than zero.");
            }

            using var reader = new VideoCapture(filePath);
            if (!reader.IsOpened())
            {
                throw new IOException($"Failed to open video file: {filePath}");
            }

            string tempFilePath = Path.GetTempFileName() + ".avi";

            using var writer = new VideoWriter(tempFilePath,
FourCC.FromString(reader.FourCC), fps,
                new Size(reader.FrameWidth, reader.FrameHeight));

            reader.Release();
            writer.Release();

            File.Delete(filePath);
            File.Move(tempFilePath, filePath);
        }
    }
}
```

PixFlow.VideoParams.cs

```
using OpenCvSharp;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PixFlow.Video.Services
{
    public sealed class VideoParams
    {
        private VideoParams() { }
```

```
        public int FrameCount { get; private set; }

        public Size FrameSize { get; private set; }

        public static VideoParams Get(string videoPath)
        {
            ArgumentException.ThrowIfNullOrEmpty(videoPath, nameof(videoPath));

            if(!File.Exists(videoPath))
            {
                throw new IOException($"Failed to open video file: {videoPath}");
            }

            using var reader = new VideoCapture(videoPath);

            return new VideoParams()
            {
                FrameCount = reader.FrameCount,
                FrameSize = new Size(reader.FrameWidth, reader.FrameHeight)
            };
        }
    }
}
```

V3sel.V3sel.consumption.cs

```
// This file was auto-generated by ML.NET Model Builder.
using Microsoft.ML;
using Microsoft.ML.Data;
using System;
using System.Linq;
using System.IO;
using System.Collections.Generic;
namespace V3sel_ConsoleApp1
{
    public partial class V3sel
    {
        public class ModelInput
        {
            [LoadColumn(0)]
            [ColumnName(@"Label")]
            public string Label { get; set; }

            [LoadColumn(1)]
            [ColumnName(@"ImageSource")]
            public byte[] ImageSource { get; set; }

        }

        public class ModelOutput
        {
            [ColumnName(@"Label")]
            public uint Label { get; set; }

            [ColumnName(@"ImageSource")]
            public byte[] ImageSource { get; set; }

            [ColumnName(@"PredictedLabel")]
            public string PredictedLabel { get; set; }

            [ColumnName(@"Score")]
            public float[] Score { get; set; }

        }
```

```
        private static string MLNetModelPath = Path.GetFullPath("V3sel.mlnet");

        public static readonly Lazy<PredictionEngine<ModelInput, ModelOutput>>
PredictEngine = new Lazy<PredictionEngine<ModelInput, ModelOutput>>(() =>
CreatePredictEngine(), true);


        private static PredictionEngine<ModelInput, ModelOutput>
CreatePredictEngine()
        {
            var mlContext = new MLContext();
            ITransformer mlModel = mlContext.Model.Load(MLNetModelPath, out var
_);
            return mlContext.Model.CreatePredictionEngine<ModelInput,
ModelOutput>(mlModel);
        }

        public static List<IOrderedEnumerable<KeyValuePair<string, float>>>
PredictAllLabels(ModelInput[] input)
        {
            var predEngine = PredictEngine.Value;

            var res = new List<IOrderedEnumerable<KeyValuePair<string, float>>>();
            foreach (var model in input)
            {
                res.Add(GetSortedScoresWithLabels(predEngine.Predict(model)));
            }

            return res;
        }

        public static IOrderedEnumerable<KeyValuePair<string, float>>
GetSortedScoresWithLabels(ModelOutput result)
        {
            var unlabeledScores = result.Score;
            var labelNames = GetLabels(result);

            Dictionary<string, float> labledScores = new Dictionary<string,
float>();
            for (int i = 0; i < labelNames.Count(); i++)
            {
                // Map the names to the predicted result score array
                var labelName = labelNames.ElementAt(i);
                labledScores.Add(labelName.ToString(), unlabeledScores[i]);
            }

            return labledScores.OrderByDescending(c => c.Value);
        }

        private static IEnumerable<string> GetLabels(ModelOutput result)
        {
            var schema = PredictEngine.Value.OutputSchema;

            var labelColumn = schema.GetColumnOrNull("Label");
            if (labelColumn == null)
            {
                throw new Exception("Label column not found. Make sure the name
searched for matches the name in the schema.");
            }

            var keyNames = new VBuffer<ReadOnlyMemory<char>>();
            labelColumn.Value.GetKeyValues(ref keyNames);
            return keyNames.DenseValues().Select(x => x.ToString());
        }

        public static ModelOutput Predict(ModelInput input)
```

```
        {
            var predEngine = PredictEngine.Value;
            return predEngine.Predict(input);
        }
    }
}
```

V3sel.V3sel.training.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.ML.Data;
using Microsoft.ML.Vision;
using Microsoft.ML;

namespace V3sel_ConsoleApp1
{
    public partial class V3sel
    {
        public static IDataView LoadImageFromFolder(MLContext mlContext, string
folder)
        {
            var res = new List<ModelInput>();
            var allowedImageExtensions = new[] { ".png", ".jpg", ".jpeg", ".gif"
};
            DirectoryInfo rootDirectoryInfo = new DirectoryInfo(folder);
            DirectoryInfo[] subDirectories = rootDirectoryInfo.GetDirectories();

            if (subDirectories.Length == 0)
            {
                throw new Exception("fail to find subdirectories");
            }

            foreach (DirectoryInfo directory in subDirectories)
            {
                var imageList = directory.EnumerateFiles().Where(f =>
allowedImageExtensions.Contains(f.Extension.ToLower()));
                if (imageList.Count() > 0)
                {
                    res.AddRange(imageList.Select(i => new ModelInput
                    {
                        Label = directory.Name,
                         ImageSource = File.ReadAllBytes(i.FullName),
                    }));
                }
            }
            return mlContext.Data.LoadFromEnumerable(res);
        }

        public static ITransformer RetrainModel(MLContext mlContext, IDataView
trainData)
        {
            var pipeline = BuildPipeline(mlContext);
            var model = pipeline.Fit(trainData);

            return model;
        }

        public static IEstimator<ITransformer> BuildPipeline(MLContext mlContext)
        {
            var pipeline =
mlContext.Transforms.Conversion.MapValueToKey(outputColumnName:@"Label",inputColum
nName:@"Label",addKeyValueAnnotationsAsText:false)
```

```
.Append(mlContext.MulticlassClassification.Trainers.ImageClassification(

labelColumnName:@"Label",scoreColumnName:@"Score",featureColumnName:@"ImageSource"
))

.Append(mlContext.Transforms.Conversion.MapKeyToValue(outputColumnName:@"Predicted
Label",inputColumnName:@"PredictedLabel"));

            return pipeline;
        }
    }
 }
```

WpfApp.IFrameSelector.cs

```
using System.Windows.Media.Imaging;

namespace WpfApp
{
    interface IFrameSelector
    {
        public BitmapSource Get(SelectorModel model);
    }
}
```

WpfApp.MainWindow.xaml.cs

```
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApp
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void VideoCheckerWindow_Button_Click(object sender,
RoutedEventArgs e)
        {
            var videoChecker = new VideoChecker();

            videoChecker.Left = Left;
            videoChecker.Top = Top;

            videoChecker.Show();

            Close();
        }
    }
}
```

WpfApp.OriginalSelector.cs

```
using OpenCvSharp;
using OpenCvSharp.WpfExtensions;
using System.Windows.Media.Imaging;

namespace WpfApp
{
    class OriginalSelector(string filepath) : IFrameSelector
    {
        private readonly string _filepath = filepath;

        public BitmapSource Get(SelectorModel model)
        {
            var original = model as OriginalSelectorModel;

            if (original == null)
            {
                throw new Exception(nameof(model));
            }

            using VideoCapture capture = VideoCapture.FromFile(_filepath);

            using Mat frame = new Mat();

            capture.Set(VideoCaptureProperties.PosFrames, original.Index);

            capture.Read(frame);

            return frame.ToBitmapSource();
        }
    }
}
```

WpfApp.V3SelectorAsBitmapSource

```
using OpenCvSharp;
using OpenCvSharp.WpfExtensions;
using System.Windows.Media.Imaging;

namespace WpfApp
{
    class V3SelectorAsBitmapSource(string filepath): IFrameSelector
    {
        private readonly string _filepath = filepath;

        public BitmapSource Get(SelectorModel model)
        {
            var v3Model = model as V3SelectorModel;

            if(v3Model == null)
            {
                throw new Exception(nameof(model));
            }

            using VideoCapture capture = VideoCapture.FromFile(_filepath);

            using Mat frame0 = new Mat();

            capture.Set(VideoCaptureProperties.PosFrames, v3Model.PrevIndex);

            capture.Read(frame0);

            using Mat frame1 = new Mat();

            capture.Set(VideoCaptureProperties.PosFrames, v3Model.NextIndex);

            capture.Read(frame1);
```

```
            using var dif = (Mat)(frame0 - frame1);

            Mat res = new Mat(frame0.Size(), frame0.Type(), new Scalar(0, 0, 0));
            for (int y = 0; y < dif.Size().Height; y++)
                for (int x = 0; x < dif.Size().Width; x++)
                {
                    var pixel = dif.Get<Vec3b>(y, x);

                    if (pixel.Item0 < 8 && pixel.Item0 != 0 && pixel.Item1 < 4 &&
pixel.Item1 != 0 && pixel.Item2 < 8 && pixel.Item2 != 0)//BGR
                    {
                        res.Set(y, x, new Vec3b(255, 255, 255));
                    }
                    else
                    {
                        res.Set(y, x, new Vec3b(0, 0, 0));
                    }
                }

            return res.ToBitmapSource();
        }
    }
}
```

WpfApp.V3SelectorModel.cs

```
namespace WpfApp
{
    record V3SelectorModel :SelectorModel
    {
        public int PrevIndex { get;set; }
        public int NextIndex { get;set; }
    }
}
```

WpfApp.VideoChecker.xaml.cs

```
using Microsoft.Win32;
using OpenCvSharp;
using OpenCvSharp.WpfExtensions;
using PixFlow.Video.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Shapes;
using static V3sel_ConsoleApp1.V3sel;
using MyML.DatasetPreparator.CustomHighlighters;

namespace WpfApp
{
    public partial class VideoChecker : System.Windows.Window, IDisposable
    {
        private string _videoPath = "";

        private int _totalFrames = 0;

        private VideoCapture? _video = null;

        private int _videoIndex = 0;
```

```
        private (bool isAnomaly, int index, float value)[]? _anomalyCheckingResult
= null;

        public VideoChecker()
        {
            InitializeComponent();
        }

        public void Dispose() => _video?.Dispose();

        private void CloseToMenu_Button_Click(object sender, RoutedEventArgs e)
        {
            MainWindow mainWindow = new MainWindow();

            mainWindow.Show();

            Close();
        }

        private void OpenVideo_Click(object sender, RoutedEventArgs e)
        {
            var openFileDialog = new OpenFileDialog { Filter = "Video files
(*.avi)|*.avi|All files (*.*)|*.*;" };

            if (openFileDialog.ShowDialog() == true)
            {
                _videoPath = openFileDialog.FileName;
                _video?.Dispose();
                _video = new VideoCapture(_videoPath);
                _totalFrames = _video.FrameCount;
                _videoIndex = 0;

                ShowFrameV2(_videoIndex);
                SetCurrentFrameInLabel(_videoIndex + 1);
            }
            else
            {
                MessageBox.Show("File not selected");
            }
        }

        private void MoveToNextFrame_Button_Click(object sender, RoutedEventArgs
e)
        {
            if (_video == null || _videoIndex + 1 >= _totalFrames) return;

            _videoIndex++;
            ShowFrameV2(_videoIndex);
            SetCurrentFrameInLabel(_videoIndex + 1);
        }

        private void SetCurrentFrameInLabel(int currentFrame)
        {
            FrameOfTotalFrames_Label.Content = $"{currentFrame}\\{_totalFrames}";
        }

        private void MoveToPrevFrame_Button_Click(object sender, RoutedEventArgs
e)
        {
            if (_video == null || _videoIndex <= 0) return;

            _videoIndex--;
            ShowFrameV2(_videoIndex);
            SetCurrentFrameInLabel(_videoIndex + 1);
        }
```

```
        private void ShowFrameV2(int showIndex)
        {
            //if (_video == null || showIndex < 0 || showIndex >= _totalFrames)
return;

            //_video.Set(VideoCaptureProperties.PosFrames, showIndex);
            //using var mat = new Mat();
            //_video.Read(mat);

            //if (!mat.Empty())
            //{
            //    VideoFrames_Image.Source = mat.ToBitmapSource();
            //}

            ImageSource image = null!;
            if ((bool)ColorView_RadioButton.IsChecked!)
            {
                image = new OriginalSelector(_videoPath).Get(new
OriginalSelectorModel() { Index = showIndex });
            }
            if((bool)SelectorV3View_RadioButton.IsChecked!)
            {
                if(showIndex == 0)
                {
                    image = new V3SelectorAsBitmapSource(_videoPath).Get(new
V3SelectorModel() { PrevIndex = 0, NextIndex = 1 });

                }
                else
                {
                    image = new V3SelectorAsBitmapSource(_videoPath).Get(new
V3SelectorModel() { PrevIndex = showIndex - 1, NextIndex = showIndex });
                }
            }
            VideoFrames_Image.Source = image;
        }

        private void ColorView_RadioButton_Checked(object sender, RoutedEventArgs
e)
        {
            if (_video != null)
            {
                ShowFrameV2(_videoIndex);
            }
        }

        private void SelectorV3View_RadioButton_Checked(object sender,
RoutedEventArgs e)
        {
            if (_video != null)
            {
                ShowFrameV2(_videoIndex);
            }
        }

        private void AnomalyChecking_Button_Click(object sender, RoutedEventArgs
e)
        {
            bool haveAnomalies = false;

            var inputModels = new List<ModelInput>();

            using var reader = new VideoCapture(_videoPath);

            using var prev = new Mat();
```

```
            reader.Read(prev);

            using var cur = new Mat();
            while (reader.Read(cur))
            {
                inputModels.Add(new ModelInput() { ImageSource = new
V3Selector().Highlight(prev, cur).ToBytes() });

                cur.CopyTo(prev);
            }

            _anomalyCheckingResult =
CheckAnomalies(inputModels.ToArray()).ToArray();

            if((float)_anomalyCheckingResult.Count(x=>x.isAnomaly ==
true)/_anomalyCheckingResult.Length > 0.1f)
            {
                MessageBox.Show("YES, WAS FOUNDED ANOMALIES");
            }
            else
            {
                MessageBox.Show("NO, NOTHING WAS FOUNDED");
            }
        }

        private List<(bool isAnomaly, int index, float value)>
CheckAnomalies(ModelInput[] inputModels)
        {
            var predicted = V3sel_ConsoleApp1.V3sel.PredictAllLabels(inputModels);

            var list = new List<(bool isAnomaly, int index, float value)>();
            int index = 0;
            foreach (var pairs in predicted)
            {
                var score = pairs.ToArray()[0];

                if (score.Key.Equals("Original"))
                {
                    list.Add((false, index++, score.Value));
                }
                else if(score.Key.Equals("Inserted"))
                {
                    list.Add((true, index++, score.Value));
                }
            }

            return list;
        }

        private void MoveToPrevAnomaly_Click(object sender, RoutedEventArgs e)
        {
            if (_anomalyCheckingResult == null) return;

            int cur = _videoIndex;

            while(cur > 0)
            {
                cur--;
                if (_anomalyCheckingResult[cur].isAnomaly)
                {
                    ShowFrameV2(cur);
                    SetCurrentFrameInLabel(cur - 1);

                    _videoIndex = cur;
```

```
                    return;
                }
            }
        }

        private void MoveToNextAnomaly_Click(object sender, RoutedEventArgs e)
        {
            if (_anomalyCheckingResult == null) return;

            int cur = _videoIndex;

            while (cur < _anomalyCheckingResult.Length -1)
            {
                cur++;
                if (_anomalyCheckingResult[cur].isAnomaly)
                {
                    ShowFrameV2(cur);
                    SetCurrentFrameInLabel(cur + 1);

                    _videoIndex = cur;

                    return;
                }
            }
        }
    }
}
```