Національний університет
«КИЄВО–МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики

КУРСОВА РОБОТА НА ТЕМУ:
«РОЗПІЗНАВАННЯ ЕМОЦІЙ У ВІДЕО–ПОТОЦІ ЗА ДОПОМОГОЮ КОМП'ЮТЕРНОГО ЗОРУ»

Виконала
Студентка БП КН–3
Факультету Інформатики
Пархоменко Анастасія Олександрівна
Науковий керівник:
к.т.н. Бучко Олена Андріївна

Київ – 2023

Тема: Розпізнавання емоцій у відео-потоці за допомогою комп’ютерного зору

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	21.06.2022	
2.	Аналіз матеріалів за темою	14.01.2023	
3.	Розробка та програмування алгоритму	10.03.2023	
4.	Написання текстової частини до курсової роботи	07.05.2023	
5.	Коригування виконаної роботи	07.05.2023	
6.	Створення слайдів для доповіді та написання доповіді.	10.05.2023	
7.	Остаточне оформлення роботи та слайдів	15.05.2023	
8.	Захист курсової роботи	23.05.2023	

Пархоменко А. О. _____

Бучко О. А. _____

“_____”_____

ЗМІСТ

ВСТУП

Людські емоції – тема, яка обростає десятками теорій, вчені різних галузей здавна досліджують їх, намагаючись пов’язати зовнішні прояви емоцій з внутрішніми чинниками та подразниками. Рання психологічна теорія емоцій, запропонована незалежно один від одного психологом Вільямом Джеймсом і фізіологом Карлом Ланге, припускає, що наші емоційні переживання є відповіддю на наші фізіологічні реакції на стимули. Іншими словами, ми відчуваємо емоції в результаті спостереження за реакціями нашого тіла (наприклад, ми відчуваємо страх, тому що помічаємо, як прискорено б'ється наше серце). А от Стенлі Шахтер і Джером Сінгер припустили, що емоції визначаються двома факторами: фізіологічним збудженням і когнітивною інтерпретацією. Згідно з цією теорією, спочатку ми відчуваємо фізіологічне збудження, але саме наша когнітивна інтерпретація ситуації визначає емоцію. Наприклад, прискорене серцебиття можна інтерпретувати як хвилювання або страх, залежно від контексту. Також поширеною є гіпотеза зворотного зв'язку з обличчям, що пов'язана з Чарльзом Дарвіном і розвинута Полом Екманом(та іншими), стверджує, що вираз нашого обличчя може впливати на наші емоційні переживання. Іншими словами, ми посміхаємося не лише тому, що щасливі, але й тому, що посмішка може змусити нас почуватися щасливими. Більш сучасний підхід, розроблений Полом Екманом, припускає, що деякі аспекти емоцій є універсальними і біологічно обумовленими, в той час як інші аспекти є культурно специфічними. Ця теорія виникла під впливом крос–культурних досліджень Екмана, присвячених міміці. Її підтримала Ліза Фельдман Барретт, і в своєму дослідженні стверджує, що емоції не є універсально запрограмованими в нашому мозку і тілі, а конструюються на основі нашого досвіду, контексту і культури. Це деякі з основних теорій емоцій, які мали вплив на психологію. Кожна теорія пропонує свій погляд на те, як визначаються емоції, і всі вони зробили свій внесок у наше розуміння людських емоцій.

Ця курсова робота присвячена розбору підходу до розпізнавання емоцій, що базується на дослідженнях фізіології обличчя, тобто активно послуговується теорією

фізіологічного зворотного зв'язку та нейрокультурною теорією емоцій. Конкретно, ми вивчатимемо, чи ефективно використовувати систему кодування дій обличчя (FACS), яку розробили Екман і Фрізен, щоб кодувати рухи обличчя і асоціювати їх з конкретними емоціями. Теорія фізіологічного зворотного зв'язку стверджує, що наші фізіологічні відгуки, зокрема вирази обличчя, впливають на наші емоційні переживання. Згідно з нейрокультурною теорією, деякі емоції і вирази обличчя є універсальними і базуються на біології, але інші варіюють від культури до культури.

Актуальність: у сучасному світі, з розвитком інформаційних технологій та комп'ютерного зору, розпізнавання емоцій людини із відео–потоку відіграє важливу роль у різних сферах діяльності. Від освіти до медицини, психології, реклами та інтерактивних розваг – можливість точно визначати емоції людини дає нові перспективи у взаємодії між людьми та технологіями.

Мета роботи: ознайомлення з релевантними моделями комп'ютерного зору, їх застосуванням, випробування системи, здатної зчитувати емоції людини із відео–потоку за допомогою імплементованої в програму технології FACS. Робота має на меті дослідити теоретичні аспекти розпізнавання емоцій та алгоритми, що застосовуються у комп'ютерному зорі для аналізу обличчя, а також реалізувати програмний застосунок, який демонструє роботу розробленої моделі на практиці.

Завдання роботи: вивчити теоретичні аспекти розпізнавання емоцій та класифікацію емоцій, ознайомитися з принципами роботи технології FACS та її застосуванням у розпізнаванні емоцій, дослідити технології та інструменти для аналізу відео–потоку та розпізнавання обличчя, розробити моделі для зчитування емоцій із відео–потоку з використанням технології FACS та без її використання, реалізувати програмний застосунок для демонстрації роботи розробленої моделі.

Об'єкт дослідження: процес розпізнавання емоцій людини із відео–потоку.

Предмет дослідження: методи та технології, що дозволяють реалізувати точне розпізнавання емоцій на основі аналізу міміки обличчя з використанням технології FACS або за допомогою інших систем.

Методи дослідження:

- Аналіз та систематизація наукової літератури та джерел з теми дослідження, огляд та порівняння існуючих методів та алгоритмів розпізнавання емоцій.
- Експериментальне дослідження для побудови та оцінки ефективності розроблених моделей зчитування емоцій із відео–потоку.
- Розробка програмного коду та візуалізація результатів на практиці.

Розділ 1. Теоретичні аспекти розпізнавання емоцій

1.1. Визначення емоцій та їх класифікація

Емоції – це комплексні психофізіологічні процеси, що характеризуються змінами в стані свідомості, організмі та поведінці людини. Емоції відіграють важливу роль під час соціальної інтеракції, в процесі прийняття рішень та в повсякденній життєдіяльності. Загальноприйнята класифікація емоцій включає в себе радість, смуток, здивування, страх, злість та відраза.

Теоретики дискретних емоцій багато років вивчали вираз обличчя як "шлях до розуміння емоцій". Більшість їхніх досліджень щодо універсальності так званих базових емоцій ґрунтується на вивченні міміки. Ці теорії стверджують, що існує лише обмежена кількість фундаментальних або базових емоцій і що для кожної з них існує прототиповий, вроджений та універсальний вираз обличчя. У цій традиції процес змішування базових патернів експресії пояснює варіативність вираження емоцій, яку ми часто спостерігаємо. Згідно з Екманом[1], інтерпретація виразу обличчя повинна спиратися на постульовані конфігурації, а не на окремі мімічні дії.

Існує чимало доказів[2], що вказують на чіткі прототипи мімічних сигналів, які можна надійно розпізнати в різних культурах як такі, що відповідають емоціям щастя, смутку, здивування, відрази, гніву і страху. Ці закономірності були виявлені в дослідженнях з використанням фотографій виразів обличчя в позі, а також в дослідженнях з зрячими і незрячими дітьми[3] у вираженні базових емоцій. Це досить цікаве спостереження, яке в основному підтримує нейрокультурну теорію емоцій, розроблену Полом Екманом. Згідно з цією теорією, деякі аспекти емоцій, включаючи вирази обличчя, є універсальними і базуються на біології, тоді як інші варіюють від культури до культури. Екман провів ряд досліджень, включаючи дослідження з різними культурами, і виявив, що існує ряд універсальних виразів обличчя, що асоціюються з певними емоціями. Ці емоції включають щастя, страх, смуток, гнів, відразу і здивування, і вирази обличчя, що асоціюються з ними, виявляються у людей з різних культур. Схожість між виразами обличчя зрячих і незрячих дітей вказує на те, що ці вирази

обличчя мають біологічну основу, а не просто навчаються через спостереження. Незрячі діти не можуть бачити вирази обличчя інших людей, щоб їх імітувати, але все одно виявляють схожі вирази обличчя при переживанні цих емоцій. Це вказує на те, що ці вирази обличчя є інстинктивними або вродженими реакціями на певні емоційні стани. Однак ці висновки не дозволили дослідникам інтерпретувати вираз обличчя як однозначний індикатор емоцій у спонтанних взаємодіях[4].

У спонтанній взаємодії поведінка обличчя супроводжує емоційні епізоди, що розгортаються, і зміни в конфігурації обличчя можуть відбуватися дуже швидко. Багато мімічних патернів, які виникають під час взаємодії, не є "істинними", спонтанними вираженнями внутрішніх емоційних процесів. Люди часто використовують експресивну поведінку більш–менш свідомо, щоб досягти соціальної мети, наприклад, привернути увагу або отримати підтримку. При цьому суб'єктивні відчуття людини та її вираз обличчя можуть не обов'язково збігатися. Відсутність відповідності між почуттями і виразом обличчя може бути також результатом процесів управління виразом обличчя, що слугують самопрезентації[5], і процесів контролю за виразом обличчя, що вимагаються соціокультурними нормами, такими як "правила демонстрації"[2][6][7]. Ця точка зору є ключовою в соціальній теорії конструкції емоцій, яка стверджує, що емоції формуються під впливом суспільства, культури і особистих взаємодій. Вона враховує те, що люди часто контролюють і регулюють свої емоції згідно з соціальними та культурними нормами. Це може включати в себе контроль над вираженням емоцій на обличчі, що може не завжди відображати внутрішні емоційні стани. Контроль експресії важливий не лише з точки зору міжіндивідуальної регуляції, але й як внутрішньо психічна стратегія подолання афектів, що переважають над силою. У всіх культурах існують правила контролю експресії не тільки для зменшення інтерактивних конфліктів, але й як неявний інструмент для навчання дітей контролювати "афективні сплески"[8]. Викликані цими різними процесами контролю і регуляції, мімічні індикатори емоційних процесів часто є дуже тонкими (наприклад, роль "мікромоментних виразів" і "невербального витоку", як обговорювали Екман і Фрізен).

1.2. Моделі та методи розпізнавання емоцій

Розпізнавання емоцій – це задача машинного зору, яка полягає в ідентифікації та класифікації емоцій на основі аналізу фізичних ознак, таких як вираз обличчя, тон голосу та жестикуляції. Існує багато моделей та методів розпізнавання емоцій, від простих алгоритмів порівняння шаблонів до складних нейромереж та глибинного навчання. Одним з найпоширеніших методів розпізнавання емоцій є метод аналізу рис і ознак, отриманих за допомогою детекторів, які визначають вираз обличчя та його параметри, такі як положення очей, рота та бровей. Ці признаки потім використовуються для класифікації емоцій за допомогою методів машинного навчання, таких як метод опорних векторів (SVM) та нейромережі. Інші методи розпізнавання емоцій, такі як методи з використанням глибинного навчання та нейромереж, також досить популярні. Глибинне навчання і нейромережі використовуються для розпізнавання складних емоцій та емоцій з використанням жестикуляції та міміки обличчя. Для розпізнавання емоцій у відео зображеннях застосовують методи з використанням аналізу послідовностей та рекурентних нейромереж.

Розпізнавання емоцій може бути здійснено за допомогою різних методів, таких як аналіз мови, аналіз тексту, аналіз звуків та аналіз обличчя. У контексті даної роботи основний акцент ставиться на розпізнаванні емоцій за допомогою аналізу обличчя.
Розпізнавання емоцій комп'ютерним зором – це процес автоматичного визначення емоцій людини за зображенням обличчя, звуком голосу або жестами. Цей процес включає в себе використання комп'ютерної обробки зображень, машинного навчання та інших технологій для аналізу сигналів, що передають емоційну інформацію.

Одним з найбільш популярних методів розпізнавання емоцій комп'ютерним зором є використання згорткових нейронних мереж (CNN). Ці моделі зазвичай навчаються на великих наборах даних з зображеннями облич людей, що мають відповідні емоційні мітки. Після тренування моделі можуть автоматично розпізнавати емоції на нових

зображеннях, аналізуючи різноманітні функції обличчя, такі як форма очей, губ, брів тощо.

Інший метод полягає у використанні комбінованих методів, які включають в себе використання камер, мікрофонів та датчиків руху для збору інформації про емоційний стан людини. За допомогою цих пристроїв можна збирати інформацію про жести, тон голосу та фізичні сигнали, що свідчать про емоції. Потім, ці дані можна обробляти за допомогою машинного навчання для визначення емоцій.

1.3. Технологія FACS (Facial Action Coding System): історія, принципи роботи та застосування

Facial Action Coding System – це система класифікації та опису виразів обличчя, яка була розроблена у 1978 році науковцями Полом Екманом та Воллесом Фрізеном. Історія створення FACS пов'язана з бажанням науковців створити об'єктивний метод оцінки виразів обличчя, що було б корисним для наукових досліджень і застосування в практиці, наприклад, в психології, криміналістиці, медицині та інших галузях. Принцип роботи цієї системи полягає в класифікації виразів обличчя на основі рухів певних м'язів[9]. FACS визначає, які м'язи обличчя задіяні при виразі, і присвоює цьому виразу унікальний код. Наприклад, вираз "посмішка" може складатися з рухів м'язів, які піднімають кутики губ, знижують верхню губу та змушують очі звужуватися. Застосування FACS досить широке: технологія використовується для дослідження емоцій та виразів обличчя, а також для психологічної діагностики та лікування, наприклад, для розуміння емоційних реакцій пацієнтів з психічними порушеннями. Крім того, FACS застосовується в криміналістиці та судовій медицині для аналізу виразів обличчя підозрюваних та свідків в справах про злочини. А також у рекламі та маркетингу для аналізу реакції споживачів на рекламні ролики та інші види реклами.

1.4. Інші технології та моделі машинного навчання для розпізнавання емоцій

Будування моделі розпізнавання емоцій без використання FACS може бути здійснено шляхом застосування згорткових нейронних мереж (CNN) безпосередньо до зображень облич. Загалом, цей підхід включає в себе:

- попередню обробку зображень, тобто їх нормалізацію до однакового розміру, перетворення в одноканальне зображення тощо;
- тренування моделі, тобто згорткова нейронна мережа може бути навчена розпізнавати емоції просто на основі великого набору даних, що містить зображення облич людей з відповідними емоційними мітками;
- прогнозування.

Розділ 2. Дослідження технологій для аналізу та визначення емоцій

2.1. Технології та інструменти для аналізу відео-потоку

Відео-потік – це послідовність зображень, які змінюються з певною частотою кадрів (fps – frames per second) і відображають рухомі об'єкти в часі. Обробка відео-потоку дозволяє виконувати ряд операцій, які включають: попередню обробку (прибирання шумів, фільтрація, калібрування камери та корекція кольору), сегментація (розділення зображення на регіони, що містять об'єкти інтересу), виявлення об'єктів (знаходження об'єктів інтересу на кадрі), відслідковування руху (визначення руху об'єктів між кадрами та аналіз траєкторії руху), аналіз змін у часі (виявлення змін, які відбуваються в зображеннях протягом часу, таких як зміна освітлення або зміна сцени). Розпізнавання облич – це задача машинного зору, яка полягає в ідентифікації та класифікації особистостей за допомогою їхніх фотографій або відео зображень. Існує багато моделей та методів розпізнавання облич, від простих алгоритмів порівняння шаблонів до складних нейромереж та глибинного навчання. Одна з найпоширеніших моделей розпізнавання облич – це метод Віоли–Джонса, який базується на каскадній класифікації з використанням Хаарових ознак. Цей метод широко використовується у системах відеоспостереження та автоматичного розпізнавання відбитків пальців. OpenFace – це високопродуктивна бібліотека, призначена для обробки та аналізу зображень обличчя в реальному часі. Вона може обробляти як статичні зображення, так і відео-потоки. Для читання та обробки відеофайлів та відео–потоків в реальному часі використовується бібліотека OpenCV, яка надає широкий спектр функцій для обробки зображень та відео.

OpenFace використовує модель Cascade Convolutional Neural Network (CNN) для цієї мети. Модель Cascade CNN була навчена виявляти обличчя на зображеннях, враховуючи різні розміри та орієнтації обличчя. Після його виявлення, OpenFace відслідковує рухи в часі. Це особливо важливо для відео–потоків, де обличчя може рухатися та змінювати свою орієнтацію з кадру в кадр. Для цього OpenFace

використовує алгоритм Kernelized Correlation Filters (KCF), який використовує властивості кернелів для створення фільтра, що відслідковує рух об'єкта.

Для аналізу міміки обличчя необхідно виявити ключові точки обличчя, такі як кути очей, ніс, рот та інші відмінності. Виявлення ключових точок може бути виконано за допомогою різних методів, наприклад, Active Appearance Models (AAM), Constrained Local Models (CLM) або глибоких нейронних мереж (DNN).

Ключові точки обличчя – це набір точок, що представляють характеристичні елементи обличчя, такі як кути очей, носа, рота, брів та контури обличчя. Виявлення ключових точок обличчя є важливим етапом для аналізу міміки та розпізнавання емоцій. Найпоширеніші методи виявлення ключових точок обличчя включають: ASM (Active Shape Models) і AAM (Active Appearance Models): ці статистичні моделі використовуються для представлення форми та текстури обличчя і можуть бути навчені для виявлення ключових точок обличчя з певною точністю. Dlib – бібліотека, що надає інструменти для виявлення ключових точок обличчя за допомогою методу, заснованого на глибокому навчанні. Глибокі нейронні мережі, такі як CNN (Convolutional Neural Networks) та FAN (Face Alignment Network), також можуть бути навчені для виявлення ключових точок обличчя з високою точністю.

OpenFace використовує модель Constrained Local Neural Fields (CLNF) для визначення рис обличчя. CLNF – це модель, що використовує структуровані нейронні поля для моделювання та відслідковування рис обличчя. Ця модель була навчена визначати ключові точки на обличчі, які представляють важливі риси, такі як очі, ніс і рот. Це здійснюється за допомогою мінімізації енергетичної функції, що включає в себе згладжувальний та конформаційний члени, які беруть до уваги внутрішню структуру обличчя та його вигляд відповідно.

Одиниці руху (AUs) – це стандартні одиниці мімічних рухів, що відображають активність мімічних м'язів обличчя – використовуються у системі кодування FACS (Facial Action Coding System). OpenFace використовує підхід, заснований на методі опорних векторів (SVMs), для визначення одиниць руху обличчя. В даному випадку,

SVM було навчено розпізнавати певні рухи обличчя, які відповідають конкретним одиницям дії в системі FACS. SVM, в основному, навчається розділяти дані на дві категорії в багатовимірному просторі за допомогою гіперплощини з максимальним розривом.

Відслідковування облич у відео–потоці важливе для аналізу динамічних змін міміки та емоцій у часі. Існують різні методи відслідковування, які базуються на порівнянні кадрів, оцінці оптичного потоку та кореляції між ключовими точками обличчя:

- Калманівські фільтри: статистичний метод для оцінки динамічних станів об'єктів, таких як позиція та орієнтація обличчя, на основі послідовності спостережень та вимірювань.
- Оптичний потік: метод для визначення руху об'єктів у відео-потоці на основі змін у яскравості пікселів між сусідніми кадрами.
- DeepSORT: алгоритм відслідковування, який використовує глибокі нейронні мережі для визначення асоціацій між спостереженнями та об'єктами відслідковування.

Можна використовувати комбінації цих методів для підвищення точності та стабільності відстеження і полягає у визначенні позиції та орієнтації обличчя у відео–потоці протягом часу.

2.2. Розробка моделі для зчитування емоцій із відео–потоку з використанням одиниць руху

На основі вимог до моделі, вибираються відповідні алгоритми та технології для розробки моделі. Вибір може включати:

- Алгоритми для виявлення обличчя та ключових точок обличчя (наприклад, MTCNN, Dlib, ASM, AAM)
- Алгоритми для розпізнавання одиниць руху(AUs) за допомогою FACS (наприклад, SVM, Random Forests, Deep Learning)

- Технології для відслідковування облич в просторі та часі (наприклад, Калманівські фільтри, оптичний потік, DeepSORT)

Після вибору алгоритмів та технологій, розробляється модель для зчитування емоцій із відео–потоку з використанням FACS. Процес розробки включає:

- Збір та підготовка навчальних даних (фото– і відео–потоки з різними емоціями та одиницями руху)
- Навчання моделі на навчальних даних з використанням вибраних алгоритмів та технологій
- Оцінка якості та точності моделі на тестових даних
- Оптимізація моделі для досягнення кращої точності та продуктивності (наприклад, налаштування гіперпараметрів, використання ансамблів алгоритмів, підгонка архітектури глибокого навчання)

Після розробки та оптимізації моделі, вона інтегрується з програмним забезпеченням для зчитування емоцій із відео–потоку в реальному часі. Інтеграція може включати:

- Розробка API (Application Programming Interface) для взаємодії з моделлю
- Реалізація функціональності для роботи з різними джерелами відео (камери, відеофайли, відео–потоки з інтернету)
- Розробка графічного інтерфейсу для відображення результатів аналізу емоцій

Для розпізнавання емоцій за допомогою FACS було розроблено декілька моделей машинного навчання. Деякі з них включають:

- FACET (Facial Action Coding System–based Expert Training): ця модель використовує експертні знання про одиниці руху та EMFACS для класифікації емоцій. Вона використовує ансамблі алгоритмів машинного навчання, таких як опорні вектори (SVM), для навчання на даних, отриманих з ручного анотування відео.
- EmoNet: Ця модель використовує конволюційні нейронні мережі (CNN) для навчання на відео–потоках з обличчями та емоціями, отриманими з датасетів,

таких як DISFA (Denver Intensity of Spontaneous Facial Action). Вона класифікує одиниці руху та емоції на основі EMFACS.

- OpenFace: модель машинного навчання, що використовує глибокі нейронні мережі для виявлення ключових точок обличчя, відслідковування обличь у відео–потоці та розпізнавання одиниць руху за допомогою FACS[15]. Ця модель є відкритим програмним забезпеченням і може бути легко інтегрована з різними додатками та системами. OpenFace використовує сучасні методи глибокого навчання та машинного навчання для вирішення складних завдань, пов'язаних з обличчями. Від виявлення обличчя за допомогою моделі Cascade CNN до визначення рис обличчя за допомогою CLNF і визначення одиниць руху обличчя за допомогою SVM, OpenFace показує, як можна застосувати різні технології машинного навчання для розуміння та аналізу облич. Однак, важливо зазначити, що, хоча OpenFace надзвичайно потужний, він має деякі обмеження. Зокрема, він може мати проблеми з виявленням обличчя в складних умовах освітлення або коли обличчя знаходиться під нестандартним кутом. Додатково, точність визначення FACS може варіюватися в залежності від якості вхідних даних та руху особи. Незважаючи на це, OpenFace залишається одним з найпотужніших інструментів для роботи з обличчями. Його гнучкість та масштабованість роблять його ідеальним вибором для багатьох застосувань, від академічних досліджень до комерційних продуктів[14].
- Deep Affect: Ця модель використовує рекурентні нейронні мережі (RNN), такі як LSTM (Long Short–Term Memory) або GRU (Gated Recurrent Unit), для аналізу послідовностей одиниць руху та емоцій у відео–потоці. Вона враховує часові залежності між одиницями руху та емоціями для покращення точності розпізнавання емоцій на основі EMFACS.

Ці моделі машинного навчання можуть бути використані як основа для розробки власної системи розпізнавання емоцій з відео–потоку за допомогою FACS. При виборі моделі

слід враховувати її точність, швидкість, адаптивність до різних сценаріїв та можливість інтеграції з іншими системами та технологіями.

2.3. Реалізація розпізнавання емоцій за допомогою FACS та моделей машинного навчання

На основі аналізу, проведеного в попередніх підрозділах, я обрала відповідні інструменти та бібліотеки для реалізації програмного застосунку:

- Мова програмування: Python та C++
- Бібліотеки для машинного навчання: TensorFlow, Keras
- Бібліотеки для обробки відео–потоку: OpenCV, FFmpeg, Dlib
- Бібліотеки для розпізнавання одиниць руху: OpenFace

Кожен компонент може бути розроблений як окремий клас або модуль, що спрощує розробку та тестування програмного застосунку. Після реалізації всіх компонентів програмного застосунку, вони повинні бути інтегровані в єдину систему, що забезпечує зчитування емоцій із відео–потоку. Це може включати налаштування зв'язків між компонентами, обробку помилок та забезпечення стабільної роботи програмного застосунку. У процесі розробки важливо проводити регулярне тестування та оптимізацію програмного застосунку для забезпечення високої продуктивності, точності розпізнавання емоцій та коректної роботи графічного інтерфейсу користувача. Після завершення розробки програмного застосунку рекомендується провести оцінку його ефективності на різних відео–потоках та емоційних станах.

Отже, моя робота послуговується високопродуктивними бібліотеками, які я зазначила вище, такими як: Dlib, OpenCV, OpenFace, Tensorflow, Keras, які використовуються в комп'ютерному зорі і машинному навчанні, я розробила окремі модулі для обробки фото- і відео-потоку, визначення одиниць руху і передбачення емоцій.

Спочатку, я використала OpenCV та DLib для обробки відео-потоку. OpenCV – це високопродуктивна бібліотека комп'ютерного зору, яка надає широкий спектр

алгоритмів для обробки зображень і відео. DLib, з іншого боку, є потужною бібліотекою для машинного навчання, яка включає функції для виявлення облич. Після обробки відео-потоку, я використала OpenFace – відкритий інструмент для визначення одиниць руху на обличчі. Задля належного подальшого тренування моделі машинного навчання, я вирішила застосувати CK+ датасет[12] – набір даних для емоційного розпізнавання облич, що був створений для навчання та тестування систем розпізнавання емоцій. Цей датасет базується на відомій базі даних Cohn–Kanade (CK), яка містить зображення облич людей за 23 різними категоріями емоцій, такими як радість, гнів, сум, страх тощо.

Датасет CK+ в свою чергу містить зображення облич 7 категорій емоцій, включаючи радість, гнів, сум, страх, здивування, огиду та нейтральну емоцію. Зображення були отримані від 123 добровольців віком від 18 до 30 років, які викликали різні емоції за допомогою спеціальних стимулів. Набір даних складається з 593 зображень, включаючи 327 зображень емоцій, що змінюються в часі, та 266 – незмінних емоцій. Кожне зображення містить інформацію про положення облич у кадрі та емоцію, яку воно виражає. CK+ використовували для навчання та тестування нейромереж, які використовуються у відеоспостереженні, медичній діагностиці та інших областях. Однак, цей датасет надає лише значення пікселів. Оскільки зображення були представлені у вигляді піксельних значень, мені довелось конвертувати їх назад у формат зображень, які потім оброблялись за допомогою OpenFace.

Після того, як я отримала аналітику на основі великого універсального набору даних, я зібрала датасет з власних фотографій, які, на мою думку, відображають 7 емоцій, які я досліджувала перед цим. Першим чином я спробувала використати модель, яку я попередньо навчила методом опорних векторів, але не отримала бажаної точності. Результати були непередбачуваними та нестабільними. Очевидно, що для вирішення цієї проблеми потрібно підійти більш гнучко. Я вирішила перейти до моделі глибокого навчання, яка відома своєю здатністю до високої адаптивності та ефективного вирішення складних проблем. Використовуючи багатошаровий підхід, я спробувала «довчити» модель на моєму, «специфічному» датасеті – моїх власних даних. «Довчання»

моделі – це процес, який включає в себе використання вже навченої моделі як основи, а потім додавання нових шарів та навчання моделі на новому наборі даних. Цей підхід дозволяє нам використовувати вже навчені властивості моделі для нового завдання, знижуючи загальний час навчання та покращуючи точність прогнозування.

Однак, не всі мої спроби вдосконалення моделі були успішними. Я експериментувала з моделлю повільного навчання, яка відома своєю здатністю до пристосування до складних даних за допомогою повільного зміщення ваг. Однак, в цьому випадку цей підхід лише погіршив результати розпізнавання. Це дослідження було невеликим нагадуванням про те, що не завжди більша складність або більше навчання означає кращі результати. Іноді, простота і повторення можуть бути більш ефективними.

В кінцевому підсумку, я зрозуміла, що застосування лише значень одиниць руху для передбачення емоцій недостатньо. Це важливе відкриття, яке вплине на мої подальші дослідження. В моїх наступних спробах я планую розглянути інші варіанти, що можуть включати комбінацію різних джерел даних, моделей та технік обробки. Важливим аспектом у цьому процесі є постійний пошук та експериментування. Можливо, комбінація глибокого навчання з іншими методами машинного навчання може принести кращі результати. Варто також дослідити інші джерела даних, такі як аудіо–записи або текст, який супроводжує відео. Ці додаткові контексти можуть допомогти моделі краще розуміти емоційні стани.

2.4. Розпізнавання емоцій за допомогою натренованої моделі та з додатковою обробкою зображень

Розпізнавання емоцій може бути виконане безпосередньо з зображень обличчя, не використовуючи систему кодування обличчевих дій (FACS). Замість аналізу окремих рухів обличчя (як у FACS), цей метод зазвичай включає аналіз всього зображення обличчя та визначення емоційного стану, заснованого на загальних шаблонах. Для цієї частини дослідження я вирішила змінити підхід та набір даних – я використала відкритий датасет — Face Emotion Recognition (FER)[13] і створила згорткову нейронну

мережу (CNN) для розпізнавання емоцій. Емоції також класифіковані на сім класів: щастя, сум, страх, відраза, гнів, нейтральність та здивування.

Модель, що була розроблена, представляє собою шестишарову згорткову нейронну мережу (CNN), створену в Keras. Ця мережа бере на вхід багатоканальні зображення і виконує згортку (множення та додавання) вхідного зображення з набором мас, які називаються фільтрами (або ще ядрами). Шари CNN мають два основних типи нейронів: згорткові та пулінгові. Згорткові нейрони виконують операцію згортки на вхідних даних, використовуючи маси, які навчаються під час тренування. Пулінгові нейрони виконують операцію пониження роздільної здатності, вибираючи максимальне або середнє значення вхідних даних. Окрім використання CNN, я вирішила покращити продуктивність моделі шляхом аугментації зображень[11]. Аугментація даних - це техніка, яка дозволяє збільшити кількість тренувальних даних за допомогою різноманітних випадкових перетворень вихідних зображень (вони продукують подібні, але все ж відмінні від оригіналу екземпляри) таких як зсув, зум, обертання тощо. Для тестування програми я вирішила скоротити набір емоцій до трьох найвиразніших, аби побачити, чи може взагалі модель визначати емоції хоча б з приблизною точністю.

Відмінність цього підходу від FACS полягає в тому, що він аналізує обличчя як ціле, а не розбиває його на окремі частини або рухи. Це може бути перевагою, особливо коли емоції виражаються у складних шаблонах, які включають багато частин обличчя. Проте, цей метод також має свої недоліки. Одним з них є те, що він може бути менш точним для деяких типів емоцій, особливо тих, які виражаються через дуже тонкі або складні рухи обличчя. Крім того, CNN вимагає великої кількості даних для тренування, і вони можуть бути важкими для розробки та налаштування.

ВИСНОВКИ

Основною задачею моєї курсової було дослідження можливостей комп’ютерного зору в поєднанні з машинним навчанням в розрізі взаємодії з людиною, а конкретно – чи піддається людська система експресії емоцій певній формалізації та категоризації, наскільки просто застосувати систему кодування міміки обличчя в одиниці руху в навчальних проектах, і чи буде вона кращою, ніж звичайне багатошарове глибоке навчання. В ході дослідження я використала декілька широко відомих наборів даних, різні методи перетворення зображень задля отримання більш якісних результатів, та зробила висновок, що зчитування міміки, кодування їх у одиниці руху та використання виключно значення одиниць руху є недостатнім інструментом для визначення емоції і не дає гарантованих результатів. Наступний метод – метод згорткових нейронних мереж на основі саме зображень(а не значень одиниць руху) дав більш задовільний результат. У відео-потоці були помітні вдосконалення визначення однієї і тієї ж емоції з кожним наступним кадром. Узагальнюючи отриману інформацію, можна сказати, що система FACS дозволяє аналізувати емоції на обличчі на більш деталізованому рівні. FACS базується на ідентифікації окремих одиниць дії обличчя (AU), які відповідають конкретним рухам м'язів обличчя. Це може допомогти виявити більш тонкі та складні емоції, які можуть бути пропущені при аналізі зображення обличчя як цілого. FACS вимагає відносно невеликого обсягу даних для тренування, оскільки вона здатна аналізувати обличчя на рівні окремих м'язів. Вона також дозволяє отримати більш детальну інформацію про емоції, таку як інтенсивність та динаміку виразу обличчя. Проте, FACS також має свої недоліки. Одним з них є те, що її може бути важко застосувати до зображень низької якості або зображень, на яких обличчя не повернуті прямо до камери. Крім того, FACS може вимагати значного обсягу обчислень, що може бути недосяжним для деяких застосувань.

Виходячи з цих спостережень, можна припустити, що ідеальний підхід до розпізнавання емоцій може полягати у поєднанні обох методів. Використання FACS може допомогти виявити більш тонкі емоції, що відображаються в мікровиразах м'язів

обличчя, в той час як загальний аналіз зображень обличчя може виявити більш очевидні емоційні стани. Крім того, використання обох методів разом може допомогти уникнути деяких обмежень кожного окремого методу.

Наприклад, у разі зниженої якості зображення або поганого освітлення, які можуть ускладнити визначення мікровиразів обличчя за допомогою FACS, загальний аналіз зображення може все ж таки виявити головні емоційні стани. З іншого боку, FACS може допомогти покращити точність визначення емоцій, коли основні емоційні стани є невиразними або змішаними.

Таким чином, стратегія, що включає обидва цих методи, може допомогти досягти більш високої точності та надійності при розпізнаванні емоцій. Проте, варто зазначити, що реалізація такого поєднання може вимагати більш складних алгоритмів та більшого обсягу обчислень, що може стати викликом для деяких застосувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Ekman, P., and Friesen, W.V. (1975). Unmasking the face. New Jersey, Prentice-Hall Inc.
2. Ekman, P, Friesen, WV., & Ellsworth, P. (1982). Methodological decisions. In P. Ekman (Ed), Emotion in the human face (pp. 56-97). Cambridge: Cambridge University Press.
3. Goodenough, F.L. (1932). Expression of the emotions in a blind-deaf child. Journal of Abnormal and Social Psychology, 27, 328-333
4. Russell, J. A. (1991). Culture and categorization of emotions. Psychological Bulletin, 110, 426–450.
5. Leventhal, H. (1974). Emotions: A basic problem for social psychology. In C. Nemeth (ed.) Social Psychology: Classic and Contemporary Integrations, pp. 1–51. Chicago: RandMcNally
6. Goffman, E. (1967). Interaction Ritual. Garden City, NJ: Doubleday/Anchor
7. Levenson, R. W. (1999). The intrapersonal functions of emotions. Cognition and Emotion, 13(5), 481–504.
8. Malatesta-Magai, C., Izard, C. E. & Camras, L. (1991). Conceptualizing early infant affect: Emotionsasfact, fiction or artifact? In K. T. Strongman (ed.) International Review of Studies on Emotion, Vol. 1, pp. 1–36. Chichester, UK: John Wiley & Sons
9. Ekman, P., & Friesen, W.V. (1978). Facial Action Coding System (FACS): A technique for the measurement offacial action. Palo Alto: Consulting Psychologists Press.
10. T. Baltrusaitis, A. Zadeh, Y. C. Lim and L. -P. Morency (2018)."OpenFace 2.0: Facial Behavior Analysis Toolkit. 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, China (2018).
11. https://github.com/ageitgey/face_recognition
12. https://www.kaggle.com/datasets/davilsena/ckdataset
13. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data
14. https://imotions.com/blog/learning/research-fundamentals/facial-action-coding-system/
15. https://github.com/TadasBaltrusaitis/OpenFace

ДОДАТКИ:

```
#include <dlib/image_processing/frontal_face_detector.h>
#include "LandmarkCoreIncludes.h"

#include <Face_utils.h>
#include <FaceAnalyser.h>
#include <GazeEstimation.h>
#include <RecorderOpenFace.h>
#include <RecorderOpenFaceParameters.h>
#include <SequenceCapture.h>
#include <ImageCapture.h>
#include <Visualizer.h>
#include <VisualizationUtils.h>

#ifndef CONFIG_DIR
#define CONFIG_DIR "~"
#endif

int static constexpr capacity = 48;

#define INFO_STREAM( stream ) \
std::cout << stream << std::endl

#define WARN_STREAM( stream ) \
std::cout << "Warning: " << stream << std::endl

#define ERROR_STREAM( stream ) \
std::cout << "Error: " << stream << std::endl

static void printErrorAndAbort(const std::string & error)
{
  std::cout << error << std::endl;
}

#define FATAL_STREAM( stream ) \
printErrorAndAbort( std::string( "Fatal error: " ) + stream )

std::vector<std::string> get_arguments(int argc, char **argv)
{
  std::vector<std::string> arguments;

  // First argument is reserved for the name of the executable
  for (int i = 0; i < argc; ++i)
  {
    arguments.push_back(std::string(argv[i]));
  }
  return arguments;
}

std::vector<std::vector<cv::Mat>> parseCSVtoMat(const std::string&
    filename) {
  std::ifstream file(filename);

  if(!file.is_open())
    throw std::runtime_error("Could not open file");

  std::string line;
  std::vector<std::vector<cv::Mat>> data;
```

Додаток 1.1. Програмний код для обробки CK+ датасету засобами OpenFace, Dlib, OpenCV

```
57
58    std::getline(file, line);
59
60    while(std::getline(file, line)) {
61      std::stringstream ss(line);
62      std::string cell;
63
64      std::getline(ss, cell, ',');
65      int emotion = std::stoi(cell);
66
67      if(emotion >= data.size()) data.resize(emotion + 1);
68
69      std::getline(ss, cell, ',');
70      std::stringstream pixelStream(cell);
71      std::string pixel;
72      cv::Mat image(capacity, capacity, CV_8UC1);
73
74      for(int i = 0; i < capacity; ++i) {
75        for(int j = 0; j < capacity; ++j) {
76          std::getline(pixelStream, pixel, ' ');
77          image.at<uchar>(i, j) = std::stoi(pixel);
78        }
79      }
80      cv::Mat resizedImage;
81      cv::resize(image, resizedImage, cv::Size(200, 200));
82      data[emotion].push_back(resizedImage);
83    }
84
85    return data;
86  }
87
88  int main(int argc, char **argv)
89  {
90    std::vector<std::string> arguments = get_arguments(argc, argv);
91
92    // no arguments: output usage
93    if (arguments.size() == 1)
94    {
95      return 0;
96    }
97    // Prepare for image reading
98    Utilities::ImageCapture image_reader;
99
100   // The sequence reader chooses what to open based on command line
        arguments provided
101   if (!image_reader.Open(arguments))
102   {
103     std::cout << "Could not open any images" << std::endl;
104     return 1;
105   }
106
107   // Load the models if images found
108   LandmarkDetector::FaceModelParameters det_parameters(arguments);
109
110   // The modules that are being used for tracking
111   std::cout << "Loading the model" << std::endl;
112   LandmarkDetector::CLNF face_model(det_parameters.model_location);
```

Додаток 1.2. Програмний код для обробки CK+ датасету засобами OpenFace, Dlib, OpenCV

```
113
114   if (!face_model.loaded_successfully)
115   {
116     std::cout << "ERROR: Could not load the landmark detector" <<
      std::endl;
117     return 1;
118   }
119
120   std::cout << "Model loaded" << std::endl;
121
122   // Load facial feature extractor and AU analyzer (make sure it is
       static)
123   FaceAnalysis::FaceAnalyserParameters face_analysis_params(
      arguments);
124   face_analysis_params.OptimizeForImages();
125   FaceAnalysis::FaceAnalyser face_analyser(face_analysis_params);
126
127   // If bounding boxes not provided, use a face detector
128   cv::CascadeClassifier classifier(det_parameters.
      haar_face_detector_location);
129   dlib::frontal_face_detector face_detector_hog = dlib::
      get_frontal_face_detector();
130   LandmarkDetector::FaceDetectorMTCNN face_detector_mtcnn(
      det_parameters.mtcnn_face_detector_location);
131
132   // If can't find MTCNN face detector, default to HOG one
133   if (det_parameters.curr_face_detector == LandmarkDetector::
      FaceModelParameters::MTCNN_DETECTOR && face_detector_mtcnn.
      empty())
134   {
135     std::cout << "INFO: defaulting to HOG-SVM face detector" << std
      ::endl;
136     det_parameters.curr_face_detector = LandmarkDetector::
      FaceModelParameters::HOG_SVM_DETECTOR;
137   }
138
139   // A utility for visualizing the results
140   Utilities::Visualizer visualizer(arguments);
141
142   auto CRplusDS = parseCSVtoMat("D://Studies//course_work//
      ckextended.csv");
143
144   if (!face_model.eye_model)
145   {
146     std::cout << "WARNING: no eye model found" << std::endl;
147   }
148
149   if (face_analyser.GetAUClassNames().size() == 0 && face_analyser.
      GetAUClassNames().size() == 0)
150   {
151     std::cout << "WARNING: no Action Unit models found" << std::
      endl;
152   }
153
154   if (!CRplusDS.empty())
155   {
156     // Detect landmarks around detected faces
```

Додаток 1.3. Програмний код для обробки CK+ датасету засобами OpenFace, Dlib, OpenCV

```
157     int face_det = 0;
158
159     // For reporting progress
160     double reported_completion = 0;
161
162     Utilities::RecorderOpenFaceParameters recording_params(
        arguments, true, false, //is sequence == true, web cam == false
163       image_reader.fx, image_reader.fy, image_reader.cx,
        image_reader.cy);
164
165     if(!face_model.eye_model)
166     {
167       recording_params.setOutputGaze(false);
168     }
169     std::string image_name = "emotion";
170     int iter = 0;
171     INFO_STREAM("Starting tracking");
172     for(int16_t j = 0; j < CRplusDS.size(); ++j)
173     {
174
175       Utilities::RecorderOpenFace open_face_rec(image_name.append(
        std::to_string(j)), recording_params, arguments);
176       for(int16_t i = 0; i < CRplusDS[j].size(); ++i)
177       {
178         visualizer.SetImage(CRplusDS[j][i], image_reader.fx,
        image_reader.fy, image_reader.cx, image_reader.cy);
179
180         // Detect faces in an image
181         std::vector<cv::Rect_<float>> face_detections;
182
183         if(image_reader.has_bounding_boxes)
184         {
185           face_detections = image_reader.GetBoundingBoxes();
186         }
187         else
188         {
189           if(det_parameters.curr_face_detector == LandmarkDetector
          ::FaceModelParameters::HOG_SVM_DETECTOR)
190           {
191             std::vector<float> confidences;
192             LandmarkDetector::DetectFacesHOG(face_detections,
            CRplusDS[j][i], face_detector_hog, confidences);
193           }
194           else if(det_parameters.curr_face_detector ==
          LandmarkDetector::FaceModelParameters::HAAR_DETECTOR)
195           {
196             LandmarkDetector::DetectFaces(face_detections, CRplusDS
            [j][i], classifier);
197           }
198           else
199           {
200             std::vector<float> confidences;
201             LandmarkDetector::DetectFacesMTCNN(face_detections,
            CRplusDS[j][i], face_detector_mtcnn, confidences);
202           }
203         }
204         bool det_sux = LandmarkDetector::DetectLandmarksInImage(
```

Додаток 1.4. Програмний код для обробки CK+ датасету засобами OpenFace, Dlib, OpenCV

```
        CRplusDS[j][i], face_detections[0], face_model, det_parameters,
         CRplusDS[j][i]);
205
206         cv::Vec6d pose_estimate = LandmarkDetector::GetPose(
        face_model, image_reader.fx, image_reader.fy, image_reader.cx,
        image_reader.cy);
207
208         // Gaze tracking, absolute gaze direction
209         cv::Point3f gaze_direction0(0, 0, -1);
210         cv::Point3f gaze_direction1(0, 0, -1);
211         cv::Vec2f gaze_angle(0, 0);
212
213         if(face_model.eye_model)
214         {
215           GazeAnalysis::EstimateGaze(face_model, gaze_direction0,
        image_reader.fx, image_reader.fy, image_reader.cx, image_reader
        .cy, true);
216           GazeAnalysis::EstimateGaze(face_model, gaze_direction1,
        image_reader.fx, image_reader.fy, image_reader.cx, image_reader
        .cy, false);
217           gaze_angle = GazeAnalysis::GetGazeAngle(gaze_direction0,
        gaze_direction1);
218         }
219
220         // Do face alignment
221         cv::Mat sim_warped_img;
222         cv::Mat_<double> hog_descriptor; int num_hog_rows = 0,
        num_hog_cols = 0;
223
224         // Perform AU detection and HOG feature extraction, as this
         can be expensive only compute it if needed by output or
        visualization
225         if(recording_params.outputAlignedFaces() ||
        recording_params.outputHOG() || recording_params.outputAUs() ||
         visualizer.vis_align || visualizer.vis_hog)
226         {
227           face_analyser.PredictStaticAUsAndComputeFeatures(CRplusDS
        [j][i], face_model.detected_landmarks);
228           face_analyser.GetLatestAlignedFace(sim_warped_img);
229           face_analyser.GetLatestHOG(hog_descriptor, num_hog_rows,
        num_hog_cols);
230         }
231
232         // Displaying the tracking visualizations
233         visualizer.SetObservationFaceAlign(sim_warped_img);
234         visualizer.SetObservationHOG(hog_descriptor, num_hog_rows,
        num_hog_cols);
235         visualizer.SetObservationLandmarks(face_model.
        detected_landmarks, 1.0, face_model.GetVisibilities()); // Set
        confidence to high to make sure we always visualize
236         visualizer.SetObservationPose(pose_estimate, 1.0);
237         visualizer.SetObservationGaze(gaze_direction0,
        gaze_direction1, LandmarkDetector::CalculateAllEyeLandmarks(
        face_model), LandmarkDetector::Calculate3DEyeLandmarks(
        face_model, image_reader.fx, image_reader.fy, image_reader.cx,
        image_reader.cy), face_model.detection_certainty);
238         visualizer.SetObservationActionUnits(face_analyser.
```

Додаток 1.5. Програмний код для обробки CK+ датасету засобами OpenFace, Dlib, OpenCV

```
        GetCurrentAUsReg(), face_analyser.GetCurrentAUsClass());
239
240         // Setting up the recorder output
241         open_face_rec.SetObservationHOG(face_model.
        detection_success, hog_descriptor, num_hog_rows, num_hog_cols,
        31); // The number of channels in HOG is fixed at the moment,
        as using FHOG
242         open_face_rec.SetObservationActionUnits(face_analyser.
        GetCurrentAUsReg(), face_analyser.GetCurrentAUsClass());
243         open_face_rec.SetObservationLandmarks(face_model.
        detected_landmarks, face_model.GetShape(image_reader.fx,
        image_reader.fy, image_reader.cx, image_reader.cy),
244           face_model.params_global, face_model.params_local,
        face_model.detection_certainty, face_model.detection_success);
245         open_face_rec.SetObservationPose(pose_estimate);
246         open_face_rec.SetObservationGaze(gaze_direction0,
        gaze_direction1, gaze_angle, LandmarkDetector::
        CalculateAllEyeLandmarks(face_model), LandmarkDetector::
        Calculate3DEyeLandmarks(face_model, image_reader.fx,
        image_reader.fy, image_reader.cx, image_reader.cy));
247         open_face_rec.SetObservationFaceAlign(sim_warped_img);
248         open_face_rec.SetObservationFaceID(0);
249         open_face_rec.WriteObservation();
250       }
251       open_face_rec.Close();
252
253       if(recording_params.outputAUs())
254       {
255         INFO_STREAM("Postprocessing the Action Unit predictions");
256         face_analyser.PostprocessOutputFile(open_face_rec.
        GetCSVFile());
257       }
258     }
259
260     // Reset the models for the next video
261     face_analyser.Reset();
262     face_model.Reset();
263   }
264
265   return 0;
266 }
```

Додаток 1.6. Програмний код для обробки CK+ датасету засобами OpenFace, Dlib, OpenCV

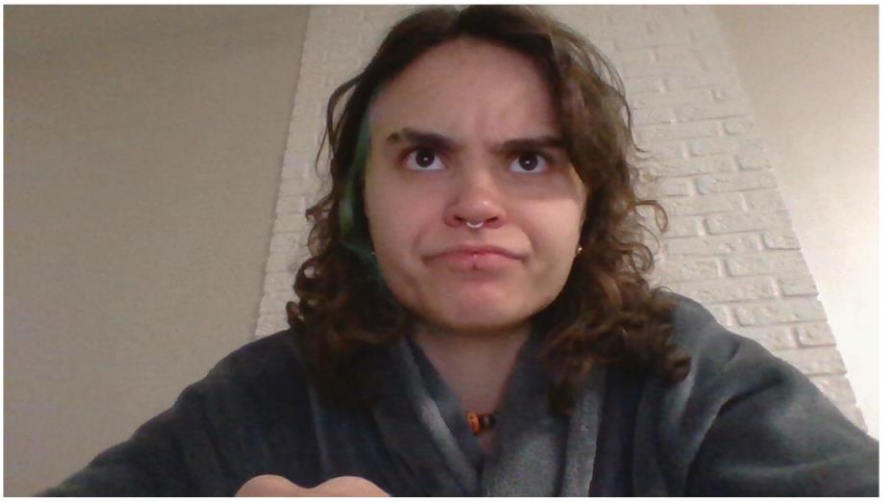
```python
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

data = pd.read_csv('wholeCK.csv')
real_data_anger = pd.read_csv('anger.csv')
real_data_contempt = pd.read_csv('contempt.csv')
real_data_disgust = pd.read_csv('disgust.csv')
real_data_fear = pd.read_csv('fear.csv')
real_data_happiness = pd.read_csv('happiness.csv')
real_data_neutral = pd.read_csv('neutral.csv')
real_data_sadness = pd.read_csv('sadness.csv')
real_data_surprise = pd.read_csv('surprise.csv')

labels = data['emotion']
features = data.drop(columns=['emotion'])

features_train, features_test, labels_train, labels_test =
    train_test_split(features, labels, test_size=0.2, random_state
    =42)

model = SVC()
model.fit(features_train, labels_train)

labels_pred = model.predict(features_test)
labels_pred_anger = model.predict(real_data_anger)
labels_pred_contempt = model.predict(real_data_contempt)
labels_pred_disgust = model.predict(real_data_disgust)
labels_pred_fear = model.predict(real_data_fear)
labels_pred_happiness = model.predict(real_data_happiness)
labels_pred_neutral = model.predict(real_data_neutral)
labels_pred_sadness = model.predict(real_data_sadness)
labels_pred_surprise = model.predict(real_data_surprise)

real_data_anger['emotion'] = labels_pred_anger
real_data_contempt['emotion'] = labels_pred_contempt
real_data_disgust['emotion'] = labels_pred_disgust
real_data_fear['emotion'] = labels_pred_fear
real_data_happiness['emotion'] = labels_pred_happiness
real_data_neutral['emotion'] = labels_pred_neutral
real_data_sadness['emotion'] = labels_pred_sadness
real_data_surprise['emotion'] = labels_pred_surprise

display(real_data_anger)
display(real_data_contempt)
display(real_data_disgust)
display(real_data_fear)
display(real_data_happiness)
display(real_data_neutral)
display(real_data_sadness)
display(real_data_surprise)
```

Додаток 2. Програмний код для тренування моделі на основі результатів роботи попереднього модуля і спроба передбачення емоцій

```
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.utils import to_categorical
from IPython import display

emotions = ['anger', 'disgust', 'fear', 'happiness', 'sadness', '
    surprise', 'neutral', 'contempt']

data = pd.read_csv('wholeCK.csv')
my_data = pd.read_csv('whole.csv')

labels = data['emotion']
features = data.drop(columns=['emotion'])
labels = to_categorical(labels)
features_train, features_test, labels_train, labels_test =
    train_test_split(features, labels, test_size=0.1, random_state
    =42)

model = models.Sequential()
model.add(layers.Dense(512, activation='relu', input_shape=(
    features_train.shape[1],)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(8, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.SGD(learning_rate=1e-4, momentum
    =0.9),
              metrics=['accuracy'])

model.fit(features_train, labels_train, epochs=50, batch_size=16,
    validation_data=(features_test, labels_test))

for layer in model.layers[:2]:
    layer.trainable = False

x = model.output
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(8, activation='softmax')(x)

model_new = models.Model(model.input, x)

model_new.compile(loss='categorical_crossentropy',
              optimizer=optimizers.SGD(learning_rate=1e-4, momentum
    =0.9),
              metrics=['accuracy'])

labels = my_data['emotion']
features = my_data.drop(columns=['emotion'])


labels = to_categorical(labels)
features_train, features_test, labels_train, labels_test =
     train_test_split(features, labels, test_size=0.1, random_state
     =42)

model_new.fit(features_train, labels_train, epochs=50, batch_size
     =16, validation_data=(features_test, labels_test))

predicted_classes = np.argmax(model_new.predict(features_test),
     axis=1)

print(type(features_test))
features_test['predicted_emotion'] = predicted_classes
```

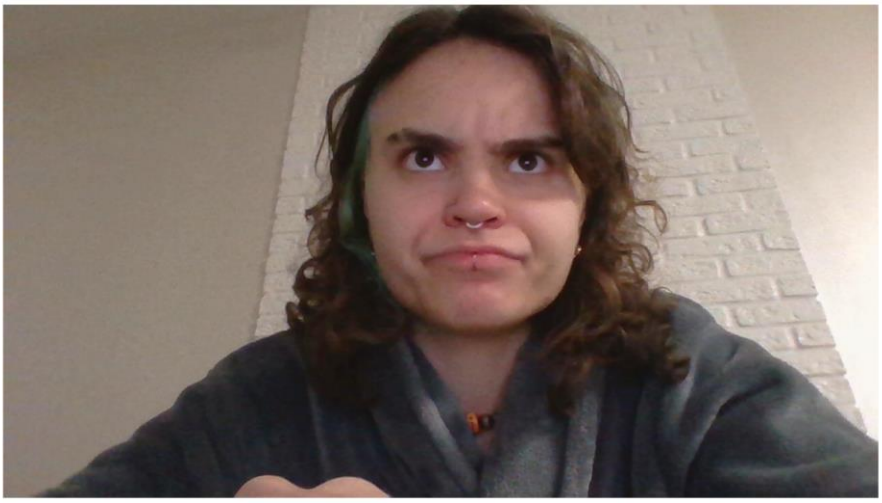
Додаток 3. Програмний код для глибокого машинного навчання на обох датасетах – CK+ та власного

```
print(emotions[int(features_test['predicted_emotion'].iloc[4])])
```



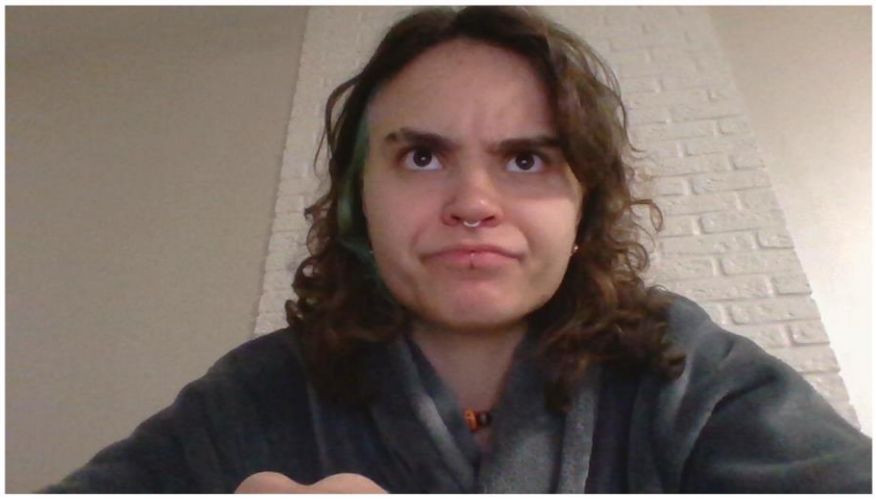
```
happiness
```

```
print(emotions[int(features_test['predicted_emotion'].iloc[4])])
```



```
disgust
```

```
print(emotions[int(features_test['predicted_emotion'].iloc[4])])
```



```
fear
```

Додаток 4. Демонстрація нестабільності передбачення емоцій програмою

```h
#pragma once
#include <chrono>
#include <memory>
#include <vector>
#include <string>

enum class Emotions
{
  Neutral,
  Happy,
  Sad,
  userAway
};

enum class AsisstantAction
{
  nothing,
  takeBreak,
  keepMood,
  cheerUp,
  userAway
};

class UserStateObserver
{
public:
  virtual ~UserStateObserver() {};
  virtual void update(AsisstantAction action) = 0;
};

class UserStateSubject {
public:
  virtual ~UserStateSubject() {};
  virtual void attach(UserStateObserver* observer) = 0;
  virtual void detach() = 0;
  virtual void notify() = 0;
};

class EmotionAnalyzer : public UserStateSubject //subject
{
  public:
    EmotionAnalyzer();

    ~EmotionAnalyzer() { delete observer; };

    void attach(UserStateObserver* observer) override;

    void detach() override;

    void notify() override;

    void setEmotion(Emotions emotion);

    void setTiredStateTimeFrame(int seconds);

    std::chrono::duration<double> countEmotionDuration(Emotions
    emotion);
```

Додаток 5.1. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, .h

```
58      AsisstantAction emotionValidation(std::chrono::duration<double>
         duration, Emotions emotion);
59
60      void determineAction(Emotions emotion);
61
62    private:
63
64      UserStateObserver* observer;
65      AsisstantAction currentAction;
66      std::chrono::steady_clock::time_point startPoint;
67      std::chrono::duration<double> tiredStateDuration;
68      std::chrono::duration<double> emotionDuration{};//zero-
        initialize
69      Emotions previousEmotion = Emotions::Neutral;
70  };
71
72  class ActReactTranslator : public UserStateObserver //observer
73  {
74  public:
75    ~ActReactTranslator();
76
77    void update(AsisstantAction action) override;
78
79    void setAnalyzer(EmotionAnalyzer& analyzer);
80
81    inline const char* resultedAction()
82    {
83      return action;
84    }
85
86
87  private:
88    const char* action = "nothing";
89    EmotionAnalyzer analyzer;
90  };
91
92  namespace Emotion
93  {
94  class EmotionRecognizer
95  {
96  public:
97    EmotionRecognizer();
98
99    void initModel(const char*);
100
101   std::vector<std::tuple<const char*, float>> emotion(unsigned char
        * img, int height, int width, bool RGB = true, bool hasAlpha =
        false);
102
103   const char* getAction();
104
105   void setTiredStateTimeFrame(int seconds);
106
107 private:
108
109   class Impl
```

Додаток 5.2. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, .h

```
110   {
111   public:
112       ActReactTranslator* translator;
113
114   public:
115     Impl();
116     ~Impl()
117     {
118       delete analyzer;
119       delete translator;
120       delete sadVietnamFlashbacks;
121     }
122
123     void initModel(const char* path);
124
125     std::vector<std::tuple<const char*, float>> emotion(unsigned
      char* img, int height, int width, bool RGB, bool hasAlpha);
126
127     const char* getAction();
128
129     void setTiredStateTimeFrame(int seconds);
130
131   private:
132     EmotionAnalyzer* analyzer;
133     unsigned char* sadVietnamFlashbacks;
134
135   private:
136     Emotions getEmotion(int maxValue);
137   };
138
139   std::unique_ptr<Impl> pImpl;
140 };
141 }
```

Додаток 5.3. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, .h

```cpp
#include "anotherMethod.h"
#include "opencv2/core.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/objdetect.hpp"
#include "opencv2/dnn/dnn.hpp"
#include "opencv2/opencv.hpp"

#include <algorithm>

ActReactTranslator::~ActReactTranslator()
{
  analyzer.detach();
  delete action;
}

void ActReactTranslator::setAnalyzer(EmotionAnalyzer& analyzer)
{
  analyzer.attach(this);
}

void ActReactTranslator::update(AsisstantAction actionToDo)
{
  switch (actionToDo)
  {
  case AsisstantAction::cheerUp:
    this->action = "cheerUp";
    break;
  case AsisstantAction::keepMood:
    this->action = "keepMood";
    break;
  case AsisstantAction::takeBreak:
    this->action = "takeBreak";
    break;
  case AsisstantAction::userAway:
    this->action = "userAway";
    break;
  default:
    this->action = "nothing";
    break;
  };
}

EmotionAnalyzer::EmotionAnalyzer()
  : startPoint(std::chrono::steady_clock::now()),
    tiredStateDuration(std::chrono::duration<double>(std::chrono::
    seconds(5)))
{}

void EmotionAnalyzer::attach(UserStateObserver* observer)
{
  this->observer = observer;
}

void EmotionAnalyzer::detach()
{
```

Додаток 5.4. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, .cpp

```
56    this->observer = nullptr;
57  }
58
59  void EmotionAnalyzer::notify()
60  {
61    this->observer->update(currentAction);
62  }
63
64  std::chrono::duration<double> EmotionAnalyzer::countEmotionDuration
        (Emotions emotion)
65  {
66    if (emotion != previousEmotion)
67    {
68      emotionDuration = std::chrono::steady_clock::now() - startPoint
        ;
69      startPoint = std::chrono::steady_clock::now();
70      setEmotion(emotion);
71    }
72    emotionDuration = std::chrono::steady_clock::now() - startPoint;
73    return emotionDuration;
74  }
75
76  AsisstantAction EmotionAnalyzer::emotionValidation(std::chrono::
        duration<double> duration, Emotions emotion)
77  {
78    if (Emotions::userAway == emotion)
79    {
80      return AsisstantAction::userAway;
81    }
82    else if (duration >= std::chrono::seconds(3))
83    {
84      if (Emotions::Neutral == emotion && duration >=
        tiredStateDuration)
85      {
86        startPoint = std::chrono::steady_clock::now();
87        return AsisstantAction::takeBreak;
88      }
89      else if (Emotions::Happy == emotion)
90      {
91        return AsisstantAction::keepMood;
92      }
93      else if (Emotions::Sad == emotion)
94      {
95        return AsisstantAction::cheerUp;
96      }
97    }
98    else return AsisstantAction::nothing;
99  }
100
101 void EmotionAnalyzer::determineAction(Emotions emotion)
102 {
103   if (observer != nullptr)
104     observer->update(emotionValidation(countEmotionDuration(emotion
        ), emotion));
105 }
106
107 void EmotionAnalyzer::EmotionAnalyzer::setEmotion(Emotions emotion)
```

Додаток 5.5. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, .cpp

```
{
  previousEmotion = emotion;
}

void EmotionAnalyzer::setTiredStateTimeFrame(int seconds)
{
  this->tiredStateDuration = std::chrono::duration<double>(std::
    chrono::seconds(seconds));
}

using namespace cv;
using namespace cv::dnn;

namespace Utils
{
    Net net;
    Point minLoc, maxLoc = { -1, -1 };
    CascadeClassifier faceCascade;
    std::vector<std::string> emotToStr = { "Neutral", "Happy", "Sad
    " };
    Mat imgCopy, faceImage, gray, reshaped, outImage;

    std::vector<Rect> haarDetect(Mat img)
    {
        std::vector<Rect> faces;

        if(faceCascade.empty())
        {
            std::cout << "\033[1;31mERROR: Couldn't open
    haarcascade_frontalface_default.xml\033[0m\n";
            return std::vector<Rect>();
        }

        faceCascade.detectMultiScale(img, faces, 1.1, 10);
        return faces;
    }
}

Emotion::EmotionRecognizer::Impl::Impl()
    : translator(new ActReactTranslator), analyzer(new
    EmotionAnalyzer),
    sadVietnamFlashbacks(nullptr)
{
    translator->setAnalyzer(*analyzer);
}

void Emotion::EmotionRecognizer::Impl::initModel(const char* path)
{
    if((path != NULL) && (path[0] == '\0'))
    {
        Utils::faceCascade.load("..\\Model\\
    haarcascade_frontalface_default.xml");
        Utils::net = cv::dnn::readNetFromONNX("..\\Model\\emotion-
    ferplus-8.onnx");
    }
    else
    {
```

Додаток 5.5. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, .cpp

```
159         std::string name = std::string(path) + "
    haarcascade_frontalface_default.xml";
160         Utils::faceCascade.load(name);
161         Utils::net = cv::dnn::readNetFromONNX(std::string(path) + "
    emotion-ferplus-8.onnx");
162     }
163 }
164
165 std::vector<std::tuple<const char*, float>> Emotion::
    EmotionRecognizer::Impl::emotion(unsigned char* img, int height
    , int width, bool RGB, bool hasAlpha)
166 {
167 #ifdef _WIN32
168     Utils::imgCopy = Mat(height, width, CV_8UC3, img);
169 #else
170     Utils::imgCopy = Mat(height, width, CV_8UC4, img);
171 #endif
172
173     Utils::maxLoc = { -1, -1 };
174     std::vector<Rect> faces = Utils::haarDetect(Utils::imgCopy);
175     Mat forwardedRes, probability;
176     std::vector<std::tuple<const char*, float>> emotionResult;
177
178     std::for_each(faces.begin(), faces.end(), [&](Rect& face)
179         {
180             Utils::faceImage = Mat(Utils::imgCopy, face);
181             if(RGB)
182             {
183                 if(hasAlpha)
184                     cvtColor(Utils::imgCopy, Utils::gray,
    COLOR_RGBA2GRAY);
185                 else
186                     cvtColor(Utils::imgCopy, Utils::gray,
    COLOR_RGB2GRAY);
187             }
188             else cvtColor(Utils::imgCopy, Utils::gray,
    COLOR_BGR2GRAY);
189
190             Utils::reshaped = blobFromImage(Utils::gray, 1.0, Size
    (64, 64));
191             Utils::net.setInput(Utils::reshaped);
192             forwardedRes = Utils::net.forward();
193
194             //some hard calibration. i‘ve got a code snippet to
    automatize it but collegues had no time left to create UI so we
     left it to the better times:)
195 #ifdef _WIN32
196             forwardedRes.at<float>(0, 0) += 0;//neutral
197             forwardedRes.at<float>(0, 1) += 0;//happy
198             forwardedRes.at<float>(0, 3) += 0;//sad
199 #else
200             forwardedRes.at<float>(0, 0) += 1;
201             forwardedRes.at<float>(0, 1) += 3.5;
202             forwardedRes.at<float>(0, 3) += 1;
203 #endif
204
205             minMaxLoc(forwardedRes, NULL, NULL, &Utils::minLoc, &
```

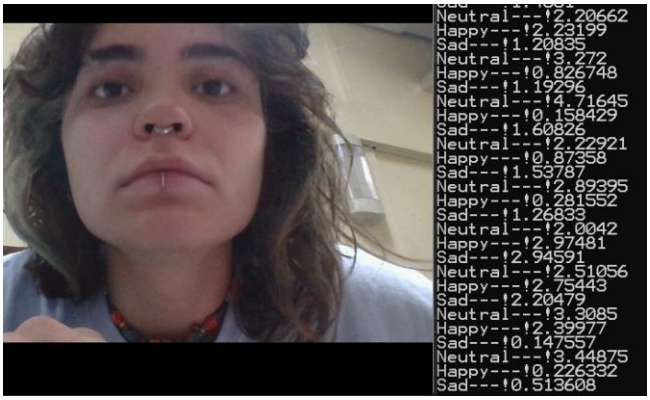
Додаток 5.6. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, .cpp

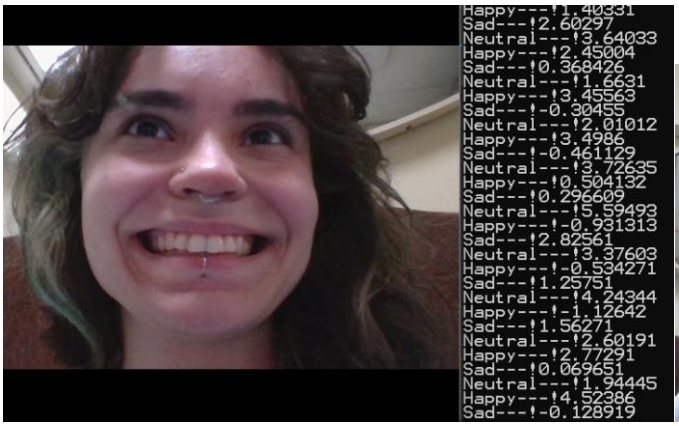
```
        Utils::maxLoc);
206             int counter = 0;
207             std::for_each(Utils::emotToStr.begin(), Utils::
        emotToStr.end(), [&](std::string& emotion)
208                 {
209                     if(counter == 2) ++counter;
210                     emotionResult.push_back(std::make_tuple(emotion
        .c_str(), forwardedRes.at<float>(0, counter)));
211                     ++counter;
212                 });
213         });
214 
215     analyzer->determineAction(getEmotion(Utils::maxLoc.x));
216 
217     return emotionResult;
218 }
219 
220 const char* Emotion::EmotionRecognizer::Impl::getAction()
221 {
222     return translator->resultedAction();
223 }
224 
225 void Emotion::EmotionRecognizer::Impl::setTiredStateTimeFrame(int
        seconds)
226 {
227     analyzer->setTiredStateTimeFrame(seconds);
228 }
229 
230 Emotions Emotion::EmotionRecognizer::Impl::getEmotion(int maxValue)
231 {
232     return maxValue == 3 ? Emotions::Sad : (maxValue == 1 ?
        Emotions::Happy : (maxValue == 0 ? Emotions::Neutral : Emotions
        ::userAway));
233 }
234 
235 //pointer to implementation
236 Emotion::EmotionRecognizer::EmotionRecognizer()
237 {
238     pImpl = std::make_unique<Impl>();
239 }
240 
241 void Emotion::EmotionRecognizer::initModel(const char* path)
242 {
243     pImpl->initModel(path);
244 }
245 
246 std::vector<std::tuple<const char*, float>> Emotion::
        EmotionRecognizer::emotion(unsigned char* img, int height, int
        width, bool RGB, bool hasAlpha)
247 {
248     return pImpl->emotion(img, height, width, RGB, hasAlpha);
249 }
250 
251 const char* Emotion::EmotionRecognizer::getAction()
252 {
253     return pImpl->getAction();
254 }


255 
256 void Emotion::EmotionRecognizer::setTiredStateTimeFrame(int seconds
        )
257 {
258     pImpl->setTiredStateTimeFrame(seconds);
259 }
```

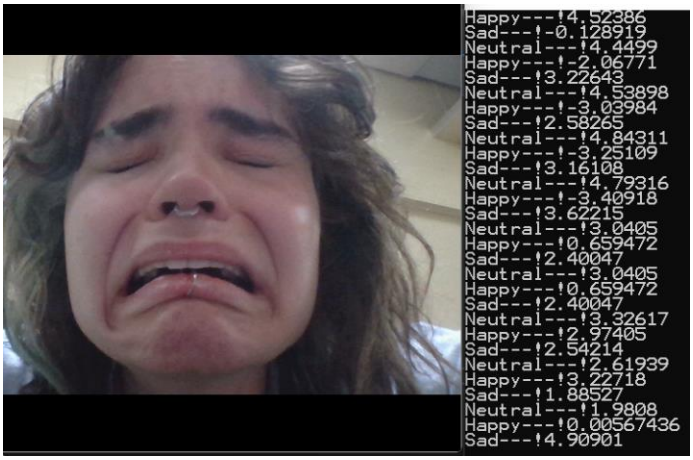
Додаток 5.7. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, .cpp

```cpp
#pragma once
#include "anotherMethod.h"
#include <opencv2/videoio.hpp>

int main()
{
  cv::VideoCapture cap;
  Emotion::EmotionRecognizer er;
  er.initModel("");

  int deviceID = 0;
  int apiID = cv::CAP_ANY;
  cv::Mat frame;
  cap.open(deviceID, apiID);

  if(!cap.isOpened()) {
    std::cerr << "ERROR! Unable to open camera\n";
    return -1;
  }

  for (;;)
    {
        cap.read(frame);

        if (frame.empty()) {
            std::cerr << "ERROR! blank frame grabbed\n";
            break;
        }
    auto a = er.emotion(frame.data, frame.rows, frame.cols);
    std::for_each(a.begin(), a.end(), [&](std::tuple<const char*,
    float> b) { std::cout << std::get<0>(b) << "---!" << std::get
    <1>(b) << "\n"; });
        // show live and wait for a key with timeout long enough to
     show images
        imshow("Live", frame);
    cv::waitKey(0);
  }
}
```

Додаток 5.8. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, main()







Додаток 5.8. Програмний код для обробки відео-потоку і застосування нової моделі глибокого навчання для визначення емоцій, main()

АНОТАЦІЯ

КУРСОВОЇ РОБОТИ

Тема .....

Студент .....

Факультет .....

Науковий керівник .....

(науковий ступінь, вчене звання, прізвище та ініціали)

Захищена «___» _____ 2023 р.

Короткий зміст роботи