МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Кафедра радіоелектронних і комп'ютерних систем

Допустити до захисту
Завідувач кафедри
_____проф. Оленич Ігор Богданович
«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

Бакалавр
(освітній ступінь)

«РЕАЛІЗАЦІЯ РОЗГОРТАННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ЗАСОБАМИ MLOPS»

Виконав:
студент IV курсу групи ФеП-41
спеціальності 121 – Інженерія програмного забезпечення
_____Стефанків Максим Володимирович
Науковий керівник:
_____асист.. Сінькевич Олег Олександрович
«_____» _____ 2023 р.
Рецензент:
_____ _____
(підпис) (ПІБ)

Львів – 2023

Анотація

У межах виконання бакалаврської дипломної роботи була розроблена мобільна програма iOS для виявлення емоцій обличчя в реальному часі. Перший етап роботи полягав у виборі відповідної моделі машинного навчання для реалізації задачі. Після вибору моделі, сервер Flask був запущений локально для обробки запитів з мобільного додатка. Сам додаток був розроблений і запущений в середовищі Xcode, використовуючи з'єднання до реального пристрою для доступу до камери. Для оптимального управління процесом машинного навчання було використано MLFlow - відкрите програмне забезпечення від Databricks, що дозволяє слідкувати за експериментами, версіонувати моделі та розгортати їх. Застосування такого інструменту дозволяє систематично контролювати процес розробки та впровадження моделей машинного навчання, сприяючи зростанню продуктивності та відповідності високим стандартам якості.

Abstract

Within the framework of the bachelor's thesis, an iOS mobile application for real-time facial emotion detection was developed. The initial phase of the work involved the selection of an appropriate machine learning model to address the task at hand. Once the model was chosen, a Flask server was launched locally to handle requests from the mobile app. The app itself was developed and launched in the Xcode environment, using a connection to a real device to gain access to the camera. MLFlow, an open-source software from Databricks, was used for optimal management of the machine learning process, allowing experiment tracking, model versioning, and deployment. The application of such a tool allows systematic control over the development and implementation process of machine learning models, contributing to increased productivity and adherence to high-quality standards.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

- **ML** - машинне навчання
- **AI** - Штучний інтелект
- **DL** - Глибоке навчання
- **ANN** - Штучна нейронна мережа
- **CNN** - Згорткова нейронна мережа
- **RNN** - Рекурентна нейронна мережа
- **LSTM** - Довга короткочасна пам'ять
- **NLP** - Обробка природної мови
- **API** - Інтерфейс прикладного програмування
- **JSON** - JavaScript Object Notation
- **REST** - Передача представницького стану
- **HTTP** - протокол передачі гіпертексту
- **URL** - Уніфікований локатор ресурсів
- **SQL** - Мова структурованих запитів
- **IOS** - Операційна система iPhone
- **EDA** - Розвідувальний аналіз даних
- **ID** - Ідентифікатор
- **IoT** - Інтернет речей

ВСТУП

Початок 21-го століття приніс з собою безпрецедентний сплеск доступності даних та обчислювальних потужностей. Ці досягнення призвели до появи і швидкого розвитку машинного навчання (МН) - галузі комп'ютерних наук, яка використовує статистичні методи, щоб дати комп'ютерам можливість навчатися на основі даних. Сьогодні алгоритми машинного навчання відіграють невід'ємну роль у багатьох аспектах сучасного суспільства, включаючи охорону здоров'я, фінанси, транспорт та розваги, і це лише деякі з них.

Незважаючи на великий потенціал і широке застосування ML, розгортання цих моделей у виробництві залишається значною проблемою. Процес переходу від розробки моделі до її розгортання часто пов'язаний з проблемами, головним чином через значну різницю між середовищами, в яких моделі навчаються, і тими, де вони в кінцевому підсумку розгортаються. Подолання цього розриву вимагає комплексного підходу, що враховує не лише модель і дані, а й інфраструктуру та робочі процеси, пов'язані з переходом від розробки до виробництва.

Саме тут на допомогу приходить MLOps, або DevOps для машинного навчання. MLOps - це набір практик, що поєднує машинне навчання, DevOps та інженерію даних, який має на меті автоматизувати розгортання та моніторинг моделей машинного навчання у виробництві. Впровадження практик MLOps допомагає забезпечити надійність, масштабованість та надійність моделей, тим самим максимізуючи віддачу від інвестицій в ML.

Актуальність даної роботи полягає в необхідності спрощення та оптимізації процесу розгортання моделей машинного навчання. Впроваджуючи підхід MLOps, ми можемо забезпечити плавний перехід моделей ML від розробки до виробництва, тим самим скоротивши час на розгортання і збільшивши повернення інвестицій в проекти ML.

Метою цієї роботи є дослідження розгортання моделей машинного навчання з використанням практики MLOps для реального сценарію використання. Об'єктом дослідження є процес розгортання моделей машинного навчання, а предметом - застосування практик MLOps до цього процесу.

Для досягнення поставленої мети в дослідженні будуть використані як теоретичні, так і практичні методи. Теоретичний аспект включає в себе всебічний огляд літератури з MLOps та суміжних тем, тоді як практична частина передбачає реалізацію конвеєра MLOps для обраного варіанту використання з використанням таких інструментів, як Python, MLflow.

Новизна цієї роботи полягає в поєднанні методів машинного навчання та MLOps, що є новою галуззю, яка ще недостатньо вивчена в академічних дослідженнях. Результати цього дослідження можуть знайти застосування в широкому спектрі галузей, скрізь, де моделі машинного навчання розгортаються в широких масштабах.

Нарешті, очікується, що потреба в спрощеному розгортанні моделей машинного навчання в майбутньому тільки зростатиме, оскільки все більше компаній впроваджуватимуть практики ML. Можливі напрямки майбутніх досліджень можуть включати вивчення практики MLOps у конкретних галузях, розробку нових інструментів і методологій для MLOps або вивчення взаємозв'язку між MLOps та іншими галузями, що розвиваються, такими як AutoML або Explainable AI.

Таким чином, це дослідження має на меті зробити внесок у зростаючий обсяг знань про MLOps, надавши практичний приклад того, як реалізувати конвеєр MLOps для розгортання моделей машинного навчання. Ми сподіваємося, що ця робота допоможе подолати розрив між розробкою та розгортанням ML і, таким чином, сприятиме ширшому впровадженню практик ML в індустрії.

Стан проблемної області

Розпізнавання емоцій привертає значну увагу в останні роки завдяки своєму потенційному застосуванню в різних сферах, таких як взаємодія людини з комп'ютером, охорона здоров'я, маркетинг та соціальна робототехніка. З розвитком машинного навчання та штучного інтелекту дослідники вивчають різні методи і техніки для точного розпізнавання та інтерпретації людських емоцій. У цьому розділі подано огляд сучасних методів розпізнавання емоцій та критично проаналізовано існуючі роботи, пов'язані з цією проблемою.

Сучасні методи розпізнавання емоцій охоплюють широкий спектр підходів, включаючи традиційні алгоритми машинного навчання, моделі глибокого навчання та методи мультимодального злиття. Традиційні алгоритми машинного навчання, такі як машини опорних векторів (SVM) і випадкові ліси, широко використовуються для розпізнавання емоцій. Ці методи, як правило, передбачають вилучення ознак з виразу обличчя, мовних сигналів, фізіологічних сигналів і текстових даних. Однак ці підходи часто страждають від обмежень у захопленні складних емоційних нюансів та узагальненні різноманітних наборів даних.

В останні роки моделі глибокого навчання, зокрема, згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN), стали потужними інструментами для розпізнавання емоцій. CNNs добре розпізнають просторову інформацію на зображеннях, що робить їх придатними для аналізу виразу обличчя. З іншого боку, CNN ефективні в моделюванні послідовних даних, таких як мова і текстова інформація. Моделі глибокого навчання продемонстрували чудову продуктивність у розпізнаванні дрібнозернистих емоцій та обробці великих масивів даних. Однак їхній успіх значною мірою залежить від наявності маркованих даних, а моделі, як правило, вимагають значних обчислювальних затрат.

Методи мультимодального злиття набули популярності, оскільки вони використовують різні джерела інформації, такі як вираз обличчя, голос і фізіологічні сигнали, для підвищення точності розпізнавання емоцій. Поєднуючи

взаємодоповнюючі модальності, ці підходи мають на меті підвищити стійкість і надійність систем розпізнавання емоцій. Методи злиття можуть виконуватися на рівні ознак, на рівні рішень або за допомогою архітектур глибокого навчання, спеціально розроблених для роботи з мультимодальними даними. Однак залишаються проблеми з ефективною інтеграцією мультимодальних даних через варіації в якості даних, часові розбіжності та специфічні для кожної модальності вимоги до попередньої обробки.

Критичний аналіз існуючих робіт, пов'язаних з розпізнаванням емоцій, дає уявлення про сильні та слабкі сторони різних підходів. Дослідники вивчили різні набори даних, такі як Affectiva-MIT Facial Expression Dataset, Toronto Face Dataset і Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), щоб оцінити ефективність моделей розпізнавання емоцій. Вони також розробили інноваційні методи вилучення ознак, архітектури моделей та стратегії оптимізації.

Однак у сфері розпізнавання емоцій залишається кілька проблем. Однією з головних проблем є відсутність стандартизованих протоколів оцінювання та еталонних наборів даних, що перешкоджає об'єктивному порівнянню різних методів. Інша проблема - надійність моделей для різних демографічних груп, культурного походження та вікових діапазонів. Емоції можуть суттєво відрізнятися у різних людей і залежать від культурних та контекстуальних факторів, що робить необхідним розробку моделей, які добре узагальнюють різні групи населення.

Отже, аналіз стану проблемної області свідчить про досягнення в сучасних методах розпізнавання емоцій, включаючи традиційні алгоритми машинного навчання, моделі глибокого навчання та методи мультимодального злиття. Хоча ці підходи продемонстрували свою перспективність, проблеми, пов'язані зі стандартизацією наборів даних, узагальненням та демографічними упередженнями, потребують вирішення. Критичний аналіз існуючих робіт дає цінну інформацію для подальших досліджень і розробок у галузі розпізнавання емоцій.

Крім того, останні розробки в галузі розгортання моделей машинного навчання призвели до появи MLOPS (Machine Learning Operations), яка

фокусується на ефективному управлінні та розгортанні моделей машинного навчання в реальних додатках. MLOPS охоплює набір практик, інструментів та робочих процесів, які спрямовані на оптимізацію процесу розгортання та підтримки моделей машинного навчання, забезпечення їхньої надійності, масштабованості та відтворюваності.

Однією з ключових проблем при розгортанні моделей машинного навчання є необхідність подолання розриву між науковцями з даних та інженерами-програмістами. Data scientists зазвичай розробляють і навчають моделі за допомогою мов програмування, таких як Python, і спеціалізованих бібліотек, таких як TensorFlow або PyTorch. Однак розгортання цих моделей у виробничих середовищах часто вимагає інтеграції з існуючими програмними системами та фреймворками, які зазвичай розробляються з використанням різних мов програмування та технологій.

MLOPS вирішує цю проблему, надаючи фреймворки та інструменти, які полегшують співпрацю між аналітиками даних та інженерами-програмістами. Наприклад, MLFlow - це платформа з відкритим вихідним кодом, яка дозволяє відстежувати та керувати експериментами з машинного навчання, пакувати моделі та розгортати їх у різних середовищах. MLFlow дозволяє аналітикам даних реєструвати та порівнювати різні експерименти, відстежувати параметри та метрики моделей, а також легко відтворювати їхні результати. Він також надає функції розгортання, які дозволяють розгортати моделі як веб-сервіси або інтегрувати їх в існуючі додатки.

На додаток до MLFlow з'явилися інші інструменти і технології для підтримки розгортання моделей машинного навчання. Наприклад, Docker - це платформа контейнеризації, яка дозволяє створювати портативні та відтворювані середовища. Упаковуючи моделі та їх залежності в контейнери, фахівці з даних можуть гарантувати, що їх моделі будуть послідовно працювати в різних середовищах, зменшуючи ризик виникнення проблем сумісності.

Іншим важливим аспектом MLOPS є інтеграція моделей машинного навчання в робочий процес DevOps (розробка та експлуатація). Практики DevOps

наголошують на автоматизації та співпраці між командами розробників, тестувальників та операторів. Інтегруючи розгортання моделей машинного навчання в конвеєр DevOps, організації можуть забезпечити постійне тестування, моніторинг та оновлення моделей. Це гарантує, що розгорнуті моделі залишаються надійними та продуктивними, навіть коли з'являються нові дані або змінюється базова інфраструктура.

Загалом, впровадження розгортання моделей машинного навчання з використанням MLOPS є значним досягненням у сфері операцій машинного навчання. Надаючи стандартизовані практики, інструменти та робочі процеси, MLOPS дозволяє організаціям ефективно розгортати моделі машинного навчання та керувати ними в реальних додатках. Це допомагає подолати розрив між аналітиками даних та інженерами програмного забезпечення, забезпечуючи надійне та масштабоване розгортання моделей, одночасно використовуючи переваги методологій DevOps.

Крім того, розгортання моделі машинного навчання з використанням MLOPS дає організаціям кілька ключових переваг. Однією з головних переваг є підвищення продуктивності та надійності моделі. MLOPS забезпечує систематичне відстеження та моніторинг моделей, що дозволяє організаціям оперативно виявляти та вирішувати проблеми з продуктивністю. Завдяки включенню циклів зворотного зв'язку та автоматизованого тестування в конвеєр розгортання, моделі можна безперервно оцінювати та оптимізувати, гарантуючи, що вони залишатимуться точними та ефективними з часом.

Ще однією перевагою MLOPS є покращена масштабованість та ефективність. Використовуючи MLOPS, організації можуть розгортати моделі машинного навчання в різних середовищах і масштабувати їх за потреби. Така гнучкість дозволяє ефективно використовувати обчислювальні ресурси і гарантує, що моделі можуть ефективно обробляти різні робочі навантаження та обсяги даних.

MLOPS також сприяє відтворюваності та співпраці. Надаючи інструменти для контролю версій та пакування моделей, MLOPS дозволяє аналітикам даних

легко ділитися своїми моделями з іншими членами команди та зацікавленими сторонами. Це сприяє співпраці та обміну знаннями всередині організації, а також відтворюваності результатів, гарантуючи, що моделі можуть бути відтворені та перевірені за необхідності.

Крім того, MLOPS підтримує ефективне управління моделями та їх дотримання. За допомогою MLOPS організації можуть впроваджувати надійні процеси документування, аудиту та безпеки моделей. Це особливо важливо в регульованих галузях, де моделі можуть мати юридичні та етичні наслідки. Впроваджуючи належну версифікацію, документування та контроль доступу, організації можуть забезпечити відповідність нормативним вимогам, а також підтримувати прозорість і підзвітність у розгортанні машинного навчання.

Крім того, MLOPS сприяє швидшому виведенню на ринок додатків машинного навчання. Оптимізуючи процес розгортання моделей і автоматизуючи повторювані завдання, MLOPS скорочує час і зусилля, необхідні для запуску моделей у виробництво. Це дає змогу організаціям швидше впроваджувати нові функції та вдосконалення, отримуючи конкурентну перевагу на ринку.

Таким чином, впровадження моделей машинного навчання з використанням MLOPS приносить організаціям численні переваги. Вони включають покращену продуктивність і надійність моделі, підвищену масштабованість і ефективність, відтворюваність і спільну роботу, ефективне управління моделлю і дотримання нормативних вимог, а також швидший час виходу на ринок. Впроваджуючи практику MLOPS, організації можуть використовувати весь потенціал своїх моделей машинного навчання, стимулюючи інновації та досягаючи відчутних бізнес-результатів.

Крім того, впровадження MLOPS для розгортання моделей машинного навчання запроваджує систематичний і стандартизований підхід до управління всім життєвим циклом моделей машинного навчання. Це охоплює різні етапи, включаючи підготовку даних, навчання моделі, оцінку моделі, розгортання, моніторинг та обслуговування.

MLOPS сприяє ефективному управлінню даними, гарантуючи, що дані, які використовуються для навчання та оцінки моделей, належним чином збираються, попередньо обробляються та перевіряються. Це передбачає перевірку якості даних, обробку відсутніх значень, обробку відхилень і забезпечення узгодженості даних. Створюючи надійні конвеєри даних і робочі процеси, організації можуть гарантувати, що їхні моделі навчаються на високоякісних даних, що призводить до більш точних і надійних прогнозів.

На етапі навчання моделі MLOPS сприяє відтворюваності та експериментуванню. Системи контролю версій, такі як Git, дозволяють аналітикам даних відстежувати та керувати змінами в своїх моделях, коді та конфігураціях. Це дозволяє легко відтворювати експерименти і за потреби відкочуватися до попередніх версій. Використовуючи такі інструменти, як MLFlow, аналітики даних можуть реєструвати параметри експерименту, метрики та артефакти, що дозволяє краще відстежувати та порівнювати різні ітерації моделі.

MLOPS також підкреслює важливість оцінки та валідації моделей. Для оцінки продуктивності навчених моделей застосовуються належні методи валідації, такі як перехресна валідація або валідація на утримання, що застосовується для оцінки продуктивності моделей, які пройшли навчання. Це гарантує, що моделі добре узагальнюють невидимі дані та надійно працюють у реальних сценаріях. Завдяки автоматизованому тестуванню та практикам безперервної інтеграції організації можуть регулярно оцінювати продуктивність розгорнутих моделей і виявляти потенційні проблеми або відхилення від очікуваної поведінки.

Розгортання моделей машинного навчання в MLOPS зазвичай досягається за допомогою технологій контейнеризації, таких як Docker або Kubernetes. Контейнери забезпечують узгоджене і портативне середовище, що дозволяє розгортати моделі на різних платформах та інфраструктурах. Інкапсулюючи модель та її залежності в контейнер, організації можуть забезпечити узгоджену поведінку та спростити процес розгортання.

Після розгортання моделей MLOPS забезпечує постійний моніторинг та підтримку. Інструменти та методи моніторингу використовуються для відстеження продуктивності моделі, виявлення аномалій та отримання зворотного зв'язку від користувачів або середовища. Це дозволяє організаціям оперативно виявляти та вирішувати проблеми, такі як дрейф концепції або деградація моделі, гарантуючи, що розгорнуті моделі залишаються ефективними та надійними протягом тривалого часу.

Обслуговування розгорнутих моделей передбачає регулярне оновлення та вдосконалення. MLOPS спрощує процес перенавчання моделей на основі нових даних, включення оновлень функцій або точного налаштування моделей для адаптації до мінливих умов. Впроваджуючи автоматизовані конвеєри та механізми контролю версій, організації можуть ефективно управляти оновленнями моделей і забезпечувати плавний перехід між різними версіями моделей.

Отже, впровадження MLOPS для розгортання моделей машинного навчання надає організаціям комплексну основу для ефективного управління життєвим циклом моделей машинного навчання. Це сприяє ефективному управлінню даними, відтворюваності, експериментуванню, оцінці моделей, розгортанню, моніторингу та обслуговуванню. Впроваджуючи практики MLOPS, організації можуть підвищити надійність, масштабованість і ремонтопридатність своїх розгортань машинного навчання, розкриваючи весь потенціал своїх моделей і досягаючи успішних результатів.Крім того, MLOPS дозволяє організаціям створити цикл зворотного зв'язку, який постійно покращує продуктивність розгорнутих моделей машинного навчання. Цей цикл зворотного зв'язку необхідний для ітеративного підвищення точності, ефективності та зручності використання моделей у реальних додатках.

Одним з аспектів циклу зворотного зв'язку є збір та аналіз відгуків користувачів. Збираючи інформацію від кінцевих користувачів або зацікавлених сторін, організації можуть отримати цінне розуміння сильних і слабких сторін розгорнутих моделей. Відгуки користувачів можуть допомогти визначити сфери, в яких моделі досягають успіху, а також сфери, які потребують подальшого

вдосконалення або додаткових навчальних даних. Цей зворотній зв'язок може допомогти визначити пріоритетність майбутніх оновлень і вдосконалень моделей.

На додаток до зворотного зв'язку з користувачами, MLOPS полегшує збір та аналіз оперативних даних. Інструменти і методи моніторингу дозволяють організаціям відстежувати продуктивність і поведінку розгорнутих моделей у виробничих середовищах. Фіксуючи такі показники, як точність прогнозування, затримки, використання ресурсів і рівень помилок, організації можуть виявити вузькі місця або аномалії в роботі, які можуть вимагати розслідування або коригування моделі. Ці оперативні дані можуть бути використані для точного налаштування моделей, оптимізації їхньої продуктивності та забезпечення їхньої надійності в реальних сценаріях.

Крім того, MLOPS підтримує інтеграцію безперервного навчання в процес розгортання моделі. Безперервне навчання передбачає оновлення моделей новими даними в міру їх надходження, що дозволяє їм адаптуватися і вдосконалюватися з часом. Впроваджуючи механізми автоматизованого введення даних, перенавчання та оновлення моделей, організації можуть підтримувати свої моделі в актуальному стані та використовувати найновішу інформацію для покращення своїх можливостей прогнозування.

MLOPS також полегшує проведення A/B-тестування та експериментів для порівняння різних версій або варіантів розгорнутих моделей. Розгортаючи кілька версій моделі одночасно і перенаправляючи вхідні запити на різні варіанти, організації можуть збирати емпіричні дані для оцінки ефективності запропонованих змін. Такий підхід, заснований на експериментах, дозволяє організаціям приймати обґрунтовані рішення щодо оновлення та оптимізації моделі на основі конкретних показників ефективності та відгуків користувачів.

Крім того, MLOPS підтримує розгортання систем моніторингу та оповіщення моделей. Ці системи безперервно відстежують продуктивність і поведінку розгорнутих моделей і запускають попередження або сповіщення при виявленні аномалій або відхилень від очікуваної поведінки. Такий проактивний

моніторинг дозволяє організаціям швидко реагувати на проблеми, підтримувати високу продуктивність моделі та мінімізувати потенційні негативні наслідки.

Загалом, цикл зворотного зв'язку, який забезпечує MLOPS, відіграє вирішальну роль у безперервному вдосконаленні розгорнутих моделей машинного навчання. Збираючи та аналізуючи відгуки користувачів, операційні дані та показники ефективності, організації можуть ітеративно вдосконалювати свої моделі, усувати обмеження та оптимізувати їхню продуктивність. Такий підхід, заснований на зворотному зв'язку, гарантує, що розгорнуті моделі залишатимуться ефективними, надійними та відповідатимуть потребам користувачів, що змінюються, та характеристикам даних, з якими вони оперують, що змінюються.

Отже, впровадження MLOPS для розгортання моделей машинного навчання дозволяє організаціям створити цикл зворотного зв'язку, який сприяє постійному вдосконаленню. Використовуючи відгуки користувачів, операційні дані та показники ефективності, організації можуть вдосконалювати та покращувати розгорнуті моделі, забезпечуючи їхню точність, ефективність та зручність використання з плином часу. Цей ітеративний процес зворотного зв'язку та вдосконалення дозволяє організаціям максимізувати цінність і вплив розгортання машинного навчання в реальних додатках.

Розділ 1. АНАЛІЗ ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

1.1 Поняття та методики Machine Learning

1.1.1 Машинне навчання

Машинне навчання (Machine Learning) є однією з ключових галузей штучного інтелекту, що вивчає методи та алгоритми, які дозволяють комп'ютерним системам самостійно набувати знання та вдосконалювати свою продуктивність через аналіз даних і взаємодію з навколишнім середовищем. У сучасному світі машинне навчання має широке застосування у багатьох галузях, включаючи комп'ютерний зір, обробку природної мови, автономні системи, медицину та фінанси.

1.1.2 Основні поняття машинного навчання

У машинному навчанні використовуються поняття, що визначають основні елементи та процеси даної галузі. Один з основних термінів - модель (model) - представляє собою математичну або статистичну структуру, яка здатна взаємодіяти зі вхідними даними і здійснювати прогнози, класифікацію або розпізнавання.

Вхідні дані (input data) представляють собою набір спостережень, характеристик або ознак, що використовуються для навчання моделі. Вихідні дані (output data) є результатами прогнозування або класифікації, які модель намагається здійснити.

Навчання моделі (model training) є процесом, під час якого модель адаптується до вхідних даних, шляхом оптимізації своїх внутрішніх параметрів або ваг. Цей процес зазвичай використовує ітеративний алгоритм, що мінімізує помилку між прогнозованими і справжніми значеннями.

1.1.3 Методики машинного навчання

Машинне навчання включає різні методики, що використовуються для розв'язання різних типів завдань. Одна з найпоширеніших методик - навчання з

учителем (supervised learning). У навчанні з учителем модель навчається на наборі прикладів, для яких відомі правильні відповіді або мітки. Метою є створення моделі, яка здатна здійснювати прогнози для нових прикладів, що не мали міток.

Інша методика - навчання без учителя (unsupervised learning), в якій модель працює з набором даних без відомих міток або відповідей. Метою є виявлення прихованих структур, кластеризація або зведення розмірності даних.

Також існує навчання з підкріпленням (reinforcement learning), де модель навчається на основі взаємодії з навколишнім середовищем та отримання винагороди або покарання за свої дії. Цей підхід використовується для навчання автономних агентів, які вчаться приймати рішення в динамічному середовищі.

1.2 Python як мова програмування для машинного навчання

Python - це високорівнева, інтерпретована мова програмування з простим синтаксисом, яка здобула значну популярність у галузі машинного навчання. Вона має численні переваги, такі як легкість вивчення, зрозумілість коду, широкий вибір бібліотек та інструментів, що роблять Python привабливим вибором для розробників, дослідників та практиків у галузі машинного навчання.

1.2.1 Основні характеристики Python для машинного навчання

- **Простота та зрозумілість:**

  Python відомий своїм простим та зрозумілим синтаксисом, що дозволяє розробникам легко вивчати та розуміти код. Це зменшує час розробки та сприяє збереженню читабельності коду, що є важливим аспектом у машинному навчанні, де розробники працюють зі складними алгоритмами та моделями.

- **Багата екосистема бібліотек:**

  Python має широкий вибір бібліотек та інструментів, спеціалізованих для машинного навчання та аналізу даних. Наприклад, NumPy надає потужні можливості для роботи з числовими даними та векторизованими операціями. Pandas дозволяє легко та ефективно обробляти та аналізувати табличні дані.

Scikit-learn надає реалізації різних алгоритмів машинного навчання, таких як класифікація, регресія, кластеризація та інше. TensorFlow та PyTorch є популярними фреймворками глибокого навчання, що дозволяють реалізувати та навчати складні нейронні мережі.

- **Швидкість та продуктивність**

  Хоча Python є інтерпретованою мовою, вона має ряд оптимізацій, що дозволяють досягти прийнятної продуктивності. Крім того, Python здатний взаємодіяти з бібліотеками, написаними на мовах програмування з низькорівневим доступом до апаратного забезпечення, таких як C та C++, що дозволяє використовувати оптимізований код у критичних за швидкістю частинах алгоритмів.

1.2.2 Використання Python у машинному навчанні

Python широко використовується на всіх етапах процесу машинного навчання. На етапі підготовки даних розробники можуть використовувати Pandas для завантаження, очищення та обробки даних. На етапі побудови моделей використовуються бібліотеки, такі як Scikit-learn, TensorFlow, PyTorch або Keras, для реалізації та навчання різноманітних моделей машинного навчання, включаючи нейронні мережі. На етапі оцінки та тестування моделей розробники використовують різні метрики та методи оцінки, що надаються бібліотеками.

1.2.3 Інтеграція з іншими мовами та платформами

Python також має можливість інтеграції з іншими мовами програмування. Наприклад, ви можете використовувати бібліотеку Cython для написання оптимізованого коду на мові C, який можна викликати з Python. Крім того, Python може взаємодіяти з іншими мовами, такими як Java або R, що дозволяє використовувати їхні бібліотеки та інструменти разом з Python.

1.3 MLFlow як інструмент для відстежування параметрів моделей машинного навчання

MLFlow є відкритим фреймворком, розробленим для керування та відстежування параметрів, моделей та експериментів у процесі розробки моделей машинного навчання. Він надає зручні інструменти для організації роботи з моделями, збереження параметрів та результатів експериментів, а також для відновлення моделей та спільної роботи між командами.

1.3.1 Основні функції MLFlow

MLflow - це платформа відкритого коду, розроблена Databricks для управління процесом машинного навчання, який включає експерименти з моделями, відтворення результатів та розгортання моделей в продакшн. Вона створена таким чином, щоб працювати з будь-якою бібліотекою машинного навчання та мовою програмування, включаючи, але не обмежуючись, TensorFlow, PyTorch, Scikit-learn, H2O, и Python.

MLflow має чотири основних компонента:

- **MLflow Tracking:** це API та UI для реєстрації параметрів, версій коду, метрик та вихідних файлів під час роботи з машинним навчанням. Він дозволяє зберігати та зробити доступними для відстеження такі деталі, як гіперпараметри моделі, метрики оцінювання моделі, теги та артефакти моделі (наприклад, вихідні дані або графіки).
- **MLflow Projects:** це формат для упакування коду машинного навчання, який може бути використаний для відтворюваності та зручного розподілу. Це описує оточення та код для виконання проекту, так що він може бути перенесений та запущений на будь-якій платформі.
- **MLflow Models:** це формат для зберігання моделей машинного навчання, який дозволяє будь-якій бібліотеці машинного навчання зберігати моделі в одному форматі, що може бути зрозумілим різними системами розгортання.

- **MLflow Registry:** це централізоване сховище моделей машинного навчання, що допомагає управляти життєвим циклом моделі ML, включаючи етапи експерименту, тестування, виробництва та архівації.

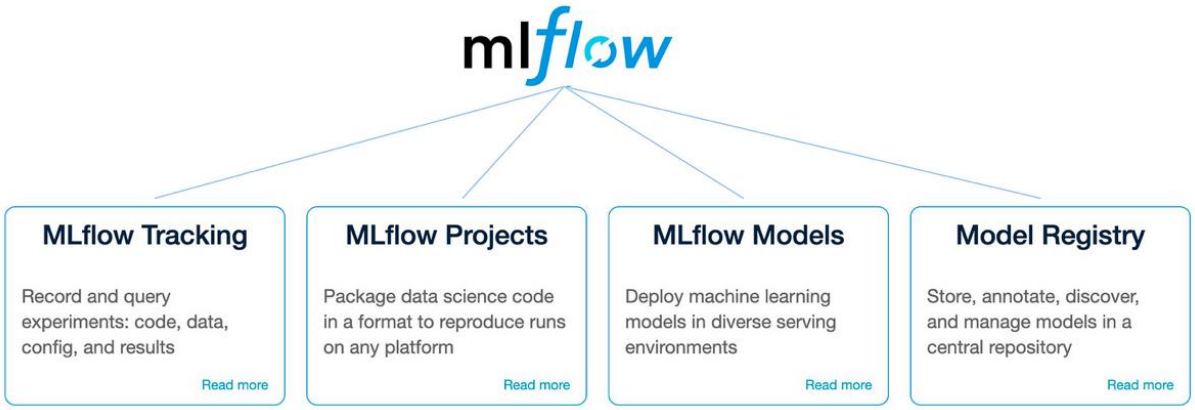

Рисунок 1.1 Основні компоненти MLFlow.

MLflow пропонує єдиний інтерфейс для всіх етапів роботи з машинним навчанням, від етапу розробки до виробництва, і допомагає керувати цим складним процесом.

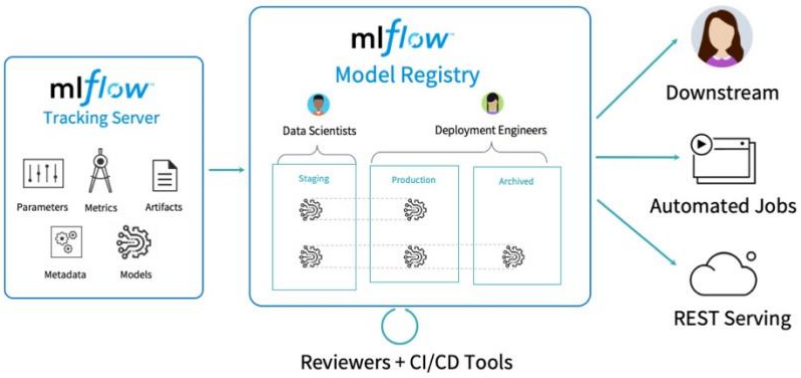

Рисунок 1.2 Приклад застосування MLFlow.

1.3.2 Характеристики MLFlow

MLFlow має декілька ключових характеристик, які роблять його потужним інструментом для розробників машинного навчання:

- **Простота використання**

  MLFlow має зрозумілий та легкий у використанні інтерфейс, що дозволяє швидко інтегрувати його у розробку моделей машинного навчання.

- **Багатомовність**

  MLFlow підтримує різні мови програмування, включаючи Python, R та Java, що робить його універсальним інструментом для розробки моделей на різних платформах.

- **Масштабованість**

  MLFlow має можливості масштабування, що дозволяють обробляти великі обсяги даних та моделей. Він може працювати з локальними обчислювальними ресурсами або інтегруватися з розподіленими системами.

- **Відкритий код**

  MLFlow є відкритим проектом, що означає, що ви можете переглядати та змінювати його джереловий код, а також вносити свої внески до спільноти розробників.

1.4 Flask як фреймворку для створення веб-додатків

Flask - це популярний і легкий веб-фреймворк для створення веб-додатків на Python. Він відомий своєю простотою, гнучкістю та легкістю у використанні, що робить його популярним вибором серед розробників для швидкого створення та розгортання веб-додатків. Flask дотримується мінімалістичного підходу, надаючи лише найнеобхідніші інструменти та функції, дозволяючи розробникам легко розширювати та налаштовувати його функціональність відповідно до їхніх конкретних вимог.

Flask надає вбудовану систему маршрутизації, яка дозволяє розробникам зіставляти URL-адреси з певними функціями або обробниками перегляду. Це дозволяє додатку реагувати на різні шаблони URL-адрес і обробляти HTTP-методи, такі як GET і POST. Маршрутизація у Flask проста та інтуїтивно зрозуміла, що дозволяє легко визначати маршрути та пов'язану з ними логіку.

Flask підтримує потужні механізми шаблонування, такі як Jinja2, що дозволяє розробникам відокремити рівень представлення від логіки програми. Шаблони дозволяють динамічно генерувати та рендерити контент, що полегшує створення динамічних веб-сторінок. Движки шаблонів надають такі функції, як заміна змінних, умови, цикли та успадкування шаблонів, що полегшує створення багаторазових і підтримуваних шаблонів HTML.

Flask надає об'єкт запиту, який інкапсулює вхідні HTTP-запити, полегшуючи доступ до даних запиту, таких як вхідні дані форми, заголовки, файли cookie та параметри запиту. Розробники можуть обробляти різні типи запитів і витягувати необхідні дані для обробки. Flask також спрощує завантаження файлів, надаючи зручні методи для обробки завантаження та доступу до завантажених файлів.

З Flask генерувати HTTP-відповіді дуже просто. Розробники можуть повертати у відповідь об'єкти, рядки, JSON-дані або навіть HTML-шаблони. Flask також підтримує узгодження вмісту, що дозволяє програмі відповідати у різних форматах (наприклад, JSON, XML) на основі вподобань клієнта або заголовків запитів.

Flask підтримує використання проміжного програмного забезпечення, яке дозволяє розробникам додавати додаткову функціональність або змінювати цикл запиту/відповіді. Проміжне програмне забезпечення можна використовувати для таких завдань, як обробка запитів, автентифікація, ведення журналів або обробка помилок. Flask також має багату екосистему розширень, які забезпечують додаткову функціональність, наприклад, інтеграцію з базами даних, автентифікацію, кешування та перевірку форм. Ці розширення можна легко інтегрувати в додатки Flask, щоб розширити їхні можливості.

Flask надає тестовий клієнт, який дозволяє розробникам імітувати HTTP-запити і тестувати функціональність свого додатку. Це полегшує написання модульних та інтеграційних тестів для додатків Flask. Крім того, Flask має вбудований режим налагодження, який надає детальні повідомлення про

помилки, трасування стеку та сервер розробки, який автоматично перезавантажує додаток при виявленні змін, прискорюючи процес розробки та налагодження.

Flask можна легко інтегрувати з іншими бібліотеками та фреймворками Python, що дозволяє розробникам використовувати їх функціональність у додатках на Flask. Наприклад, Flask можна використовувати з SQLAlchemy для інтеграції з базами даних, WTForms для обробки та валідації форм або Flask-Security для автентифікації та авторизації.

Простота і гнучкість Flask роблять його чудовим вибором для малих і середніх веб-додатків, API та прототипів. Його легка природа дозволяє швидко розробляти і розгортати, забезпечуючи при цьому достатню кількість функцій і розширюваність, щоб задовольнити різноманітні вимоги до веб-додатків. Незалежно від того, чи створюєте ви простий додаток, чи складний веб-проект, Flask надає розробникам міцну основу для створення ефективних, масштабованих і підтримуваних веб-додатків на Python.

Основні характеристики Flask:

- **Мікрофреймворк**
  Flask є мікрофреймворком, що означає, що він має мінімальний набір функцій "з коробки", але може бути легко розширений за допомогою різних плагінів.
- **Маршрутизація**
  Flask дозволяє легко визначати маршрути (або URL) для вашого веб-додатку, використовуючи декоратори Python.
- **Підтримка запитів і відповідей**
  Flask має вбудовані функції для обробки HTTP-запитів і відповідей.
- **Підтримка шаблонів**
  Flask підтримує Jinja2, потужну систему шаблонів Python, яка дозволяє генерувати динамічний HTML-код.
- **Підтримка сесій і кукі**
  Flask має вбудовану підтримку сесій і кукі, що є важливим для створення інтерактивних веб-додатків.

У моєму проекті Flask використовується для створення веб-додатку, який обробляє запити від IOS додатку. Flask був обраний через його легкість, гнучкість і сумісність з Python, мовою програмування, яку я використовую для розробки моделей машинного навчання в цьому проекті.

1.5 Розробки мобільних додатків на IOS

Розробка мобільних додатків для iOS є важливим аспектом у галузі технологій, оскільки продукти Apple, як-от iPhone, iPad та Apple Watch, є надзвичайно популярними по всьому світу. Для розробки додатків для iOS, Apple надає свій власний набір інструментів і технологій.

1.5.1 Swift

Swift - це мова програмування, розроблена Apple, яка використовується для розробки додатків для iOS, macOS, watchOS і tvOS. Swift відрізняється від більшості інших мов програмування тим, що вона була спеціально розроблена для забезпечення високої продуктивності та безпеки.

1.5.2 Xcode

Xcode - це інтегроване середовище розробки (IDE), яке використовується для створення додатків для продуктів Apple. Xcode надає розробникам доступ до бібліотек інструментів, засобів для налагодження, інтерфейсу для розробки, а також можливість тестування додатків на віртуальних або реальних пристроях.

1.5.3 SwiftUI

SwiftUI - це декларативний фреймворк для інтерфейсу, який дозволяє розробникам більш ефективно створювати UI для своїх додатків. Він надає засоби для створення гнучких, реактивних інтерфейсів для всіх продуктів Apple.

Swift характеризується особливою безпекою та продуктивністю. Вона була розроблена з акцентом на усунення цілих категорій поширених помилок

програмування, таких як зневажання нульових вказівників. Її синтаксис сприяє написанню чистого та послідовного коду. Swift також має відмінну продуктивність у реальному часі, як правило, перевершуючи Objective-C та Python.

Синтаксис Swift є виразним та чистим. Swift використовує чистий, виразний та легко зрозумілий синтаксис. Це робить мову більш доступною для новачків та пропонує досвідченим розробникам більш швидкий та ефективний досвід кодування.

Swift використовує виведення типів, що означає, що програміст не має анотувати змінні типами даних - компілятор може вивести тип на основі значення змінної.

Щодо управління пам'яттю, Swift використовує автоматичне підрахунок посилань (ARC), що допомагає збільшувати ефективність додатків, звільняючи ресурси пам'яті, коли вони більше не потрібні.

Swift може працювати поруч з Objective-C, що дозволяє розробникам створювати проект, який містить файли, написані на обох мовах.

iOS є мобільною операційною системою Apple, що використовується для запуску iPhone, iPad та iPod Touch.

iOS часто визнають за сильні заходи безпеки. Вона має закриту або контрольовану екосистему, де додатки походять тільки з App Store Apple, і кожен додаток ізольований, щоб запобігти його шкідливому впливу на інші частини системи.

iOS відома своїм високоякісним досвідом користувача. Її інтерфейс вважається чистим, елегантним та легким у використанні.

App Store для iOS встановлює суворі стандарти якості для всіх своїх додатків. Це призводить до колекції високоякісних додатків, що покращують досвід користувача.

З інструментами, такими як Swift та Xcode, розробка для iOS може бути більш структурованим досвідом. Більше того, стандартизовані пристрої та версії операційної системи Apple зменшують складність, яка виникає в середовищах, подібних до Android, що має широкий спектр пристроїв та версій ОС.

Незважаючи на те, що Android має більше глобальних користувачів, дослідження показують, що користувачі iOS, як правило, більше готові платити за додатки, що може зробити розробку додатків для iOS більш прибутковою для бізнесу.

AVFoundation - це фреймворк для iOS, який надає високорівневі API для роботи з аудіо- та відеоматеріалами. Він дозволяє розробникам захоплювати, відтворювати, редагувати та маніпулювати аудіо та відео. За допомогою AVFoundation розробники можуть отримати доступ до камери та мікрофона пристрою, виконувати обробку аудіо та відео в реальному часі, а також реалізовувати такі функції, як запис, відтворення та потокове передавання відео.

CoreMedia - це базовий фреймворк в iOS, який надає низькорівневі API для роботи з медіаформатами та перетворення медіа. Він включає такі функції, як робота з медіафайлами, потокове передавання медіа та синхронізація медіа. CoreMedia надає базову функціональність, на якій побудовані медіа-фреймворки вищого рівня, такі як AVFoundation.

CoreImage - це потужний фреймворк для обробки зображень для iOS. Він пропонує широкий спектр вбудованих фільтрів та ефектів, які можна застосовувати до зображень і відео в режимі реального часу. CoreImage дозволяє розробникам виконувати такі завдання, як покращення зображень, зменшення шуму, корекція кольору та розпізнавання облич.

CoreImage дозволяє розробникам виконувати такі завдання, як покращення зображення, шумозаглушення, кольорокорекція та розпізнавання облич. Він також надає можливість створювати власні фільтри для обробки зображень і відео.

UIKit - основний фреймворк для побудови користувацьких інтерфейсів в iOS-додатках. Він надає повний набір інструментів та API для створення візуально привабливих та інтерактивних користувацьких інтерфейсів. UIKit пропонує велику кількість готових компонентів інтерфейсу, таких як кнопки, мітки, текстові поля, таблиці та колекції. Він також включає підтримку анімації, розпізнавання жестів і управління розташуванням, що полегшує проектування і реалізацію користувацьких інтерфейсів.

Vision - це потужний фреймворк, який з'явився в iOS 11 і надає можливості комп'ютерного зору. З його допомогою розробники можуть виконувати завдання комп'ютерного зору, такі як розпізнавання зображень, відстеження об'єктів та розпізнавання облич. Vision дозволяє використовувати попередньо навчені моделі машинного навчання для виконання складних завдань комп'ютерного зору без необхідності розробляти моделі з нуля. Він надає прості у використанні API для інтеграції можливостей комп'ютерного зору в додатки для iOS.

Alamofire - популярна бібліотека сторонніх розробників для створення мереж у розробці iOS. Вона спрощує процес створення мережевих запитів та обробки відповідей, надаючи високорівневий інтерфейс, заснований на URLSession від Apple. Alamofire підтримує різноманітні методи HTTP, серіалізацію запитів/відповідей, аутентифікацію відповіді, автентифікацію та фонову обробку завдань. Це допомагає спростити мережеві операції і дозволяє розробникам зосередитися на основній функціональності своїх додатків.

Загалом, використання таких фреймворків і технологій, як AVFoundation, CoreMedia, CoreImage, UIKit, Vision та Alamofire, дозволяє створювати потужні та інноваційні додатки для iOS. Ці технології допомагають забезпечити функціональність для роботи з медіа, обробки зображень, користувацьких інтерфейсів, комп'ютерного зору та роботи в мережі.

Розділ 2. Програмна реалізація та аналіз отриманих результатів

2.1 Підготовка даних

У будь-якому процесі машинного навчання підготовка даних є критично важливим етапом попередньої обробки, який безпосередньо впливає на продуктивність моделей. У цьому проекті завдання розпізнавання виразу обличчя (FER) вимагало відповідного набору даних, який складався із зображень людських облич, позначених відповідними емоціями. З цією метою було використано набір даних fer2013, який було отримано з платформи Kaggle.

Набір даних fer2013 - це публічний набір даних, доступний на Kaggle, популярній онлайн-спільноті науковців з даних та практиків машинного навчання, яка надає створений користувачами контент, набори даних та змагання. Набір даних fer2013 спеціально розроблений для навчання моделей машинного навчання та глибокого навчання для задачі розпізнавання емоцій на обличчі.

Дані складаються з зображень облич у відтінках сірого розміром 48x48 пікселів. Обличчя були автоматично зареєстровані таким чином, щоб обличчя було більш-менш по центру і займало приблизно однакову кількість місця на кожному зображенні.

Набір даних fer2013 складається з сімома мітками емоцій: "Злість", "Огида", "Страх", "Щастя", "Сум", "Здивування" та "Нейтральний". Навчальна вибірка складається з 28 709 прикладів, а загальнодоступна тестова вибірка - з 3 589 прикладів. Приклад даних Рис.3.

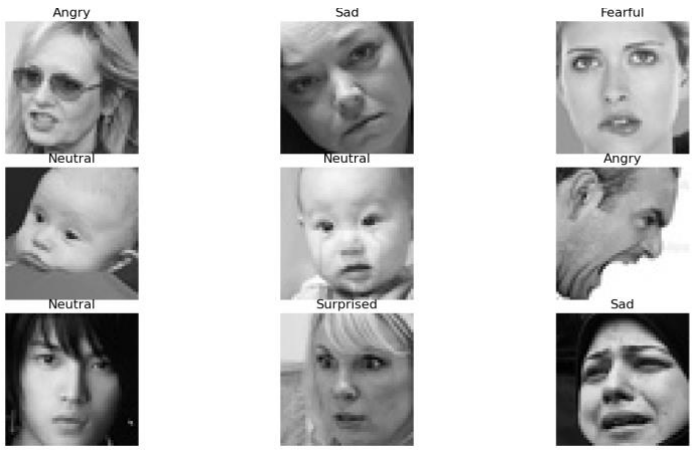

Рисунок 2.1 Набір даних з різними емоціями

Після того, як дані були зібрані, наступним кроком була обробка та очищення даних, щоб зробити їх придатними для використання в моделях машинного навчання. Враховуючи, що набір даних fer2013 містить зображення, загальноприйнятою практикою є нормалізація значень пікселів для кращої продуктивності моделей. У цьому випадку інтенсивність пікселів було масштабовано до діапазону від 0 до 1, замість початкового діапазону відтінків сірого від 0 до 255.

Іншим важливим аспектом підготовки даних була зміна форми даних відповідно до вхідних вимог моделі. Оскільки набір даних fer2013 містить зображення у сплющеному форматі, їх було перетворено у вихідну структуру 48x48 пікселів. Крім того, оскільки зображення мають відтінки сірого, було задано один колірний канал. Таким чином, оброблені зображення мали форму 48 пікселів на 48 пікселів на 1 колірний канал.

Крім того, мітки емоцій були закодовані в однократному кодуванні, щоб вони підходили для моделі. При однократному кодуванні кожна категорія емоцій представлена у вигляді унікального бінарного вектора в просторі міток. Наприклад, якщо є сім категорій емоцій, "Гнів" може бути представлений як [1, 0, 0, 0, 0, 0, 0], "Відраза" як [0, 1, 0, 0, 0, 0, 0] і так далі.

В рамках процесу очищення даних було перевірено будь-які невідповідності або пропущені значення в наборі даних. Втім, набір даних fer2013 загалом добре очищений, і таких проблем не було виявлено. Отже, після виконання цих кроків з підготовки даних, вони були готові до використання для навчання та перевірки моделі.

2.2 Моделі розпізнавання емоцій та їх параметри

2.1.1 Перша модель (див.додат.А)

Ця модель - це конволюційна нейронна мережа (CNN), яка включає в себе наступні шари:

- Чотири пари шарів Conv2D та MaxPooling2D. Conv2D - це шари конволюції, які використовуються для виявлення шаблонів на зображенні, а MaxPooling2D - це шари зменшення розміру, які використовуються для зменшення розміру вхідного зображення, зберігаючи при цьому найважливішу інформацію.
- Шар Flatten, який перетворює вхідні дані в одномірний вектор.
- Два Dense (або повністю з'єднані) шари з активацією ReLU, які використовуються для виконання класифікації.
- Dense шар для виходу з активацією Softmax, який використовується для прогнозування ймовірностей кожного класу.

Параметри:

- Розмір зображення: 48x48
- Кількість епох: 30
- Розмір пакету: 64
- Критерій ранньої зупинки: 5
- Оптимізатор: Adam
- Функція втрати: categorical_crossentropy
- Метрика: accuracy

Результати:

Результатом роботи цього коду буде навчена модель, яка може використовуватися для розпізнавання емоцій на нових зображеннях. Параметри моделі (гіперпараметри, точність навчання та валідації, втрати тощо) зберігаються в середовищі MLflow, яке є інструментом для моніторингу та управління машинним навчанням.

Виходом цього коду будуть також метрики втрати (loss) та точності (accuracy) для навчального та валідаційного наборів даних, а також серіалізована навчена модель, яка зберігається в MLflow. Ці результати можна використовувати для оцінки продуктивності моделі та її подальшого використання для прогнозування емоцій на нових зображеннях.

Архітектура моделі:

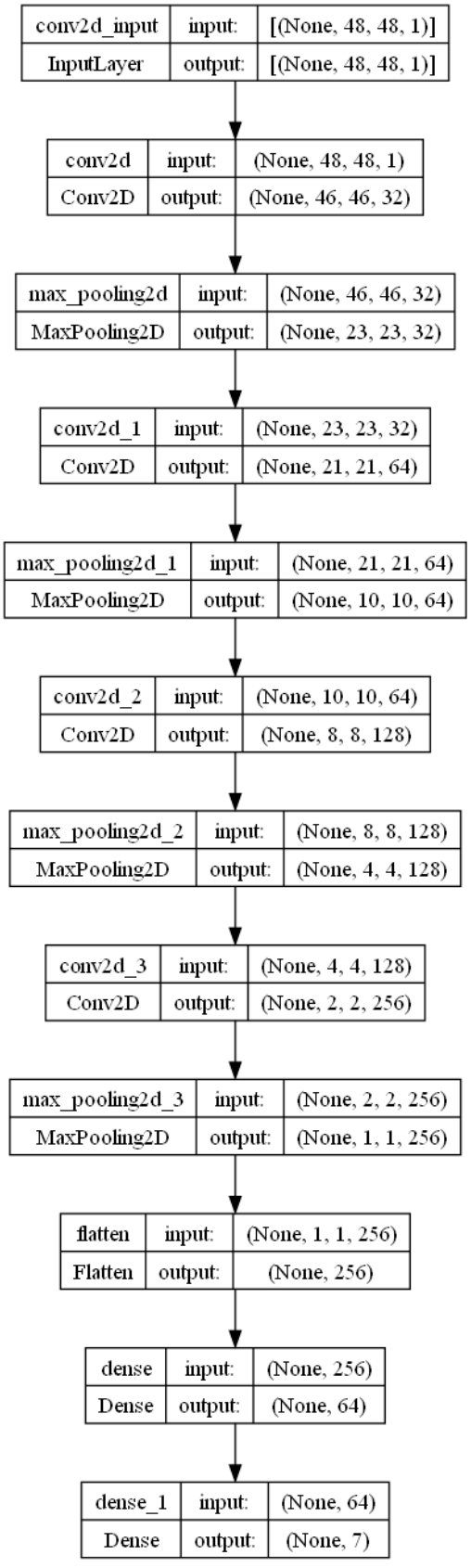

Рисунок 2.2 Архітектура першої моделі

2.1.2 Друга модель (див.додат.А)

Ця модель використовує архітектуру DenseNet169, попередньо навчену модель, доступну в Keras, в якості основи для виявлення особливостей. Після DenseNet169, використовуються декілька Dense та Dropout шарів для класифікації.

Параметри:

- Розмір зображення: 48x48
- Кількість епох: 30 (з можливістю додаткового тонкого налаштування)
- Розмір пакету: 32
- Критерій ранньої зупинки: 5
- Оптимізатор: Adam
- Функція втрати: categorical_crossentropy
- Метрика: accuracy

Модель DenseNet169 має приблизно 14 мільйонів параметрів. При додаванні додаткових шарів Dense та Dropout, кількість параметрів збільшується, але вона все одно буде значно нижчою, ніж в оригінальній моделі DenseNet169.

Опис моделі:

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 48, 48, 3)]       0
_________________________________________________________________
densenet169 (Functional)     (None, 1, 1, 1664)        12642880
_________________________________________________________________
global_average_pooling2d (Gl (None, 1664)              0
_________________________________________________________________
dense (Dense)                (None, 256)               426240
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              263168
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
classification (Dense)       (None, 7)                 3591
=================================================================
Total params: 13,860,679
Trainable params: 1,217,799
Non-trainable params: 12,642,880
_________________________________________________________________
```

Рисунок 2.3 Опис другої моделі

Результат моделі:

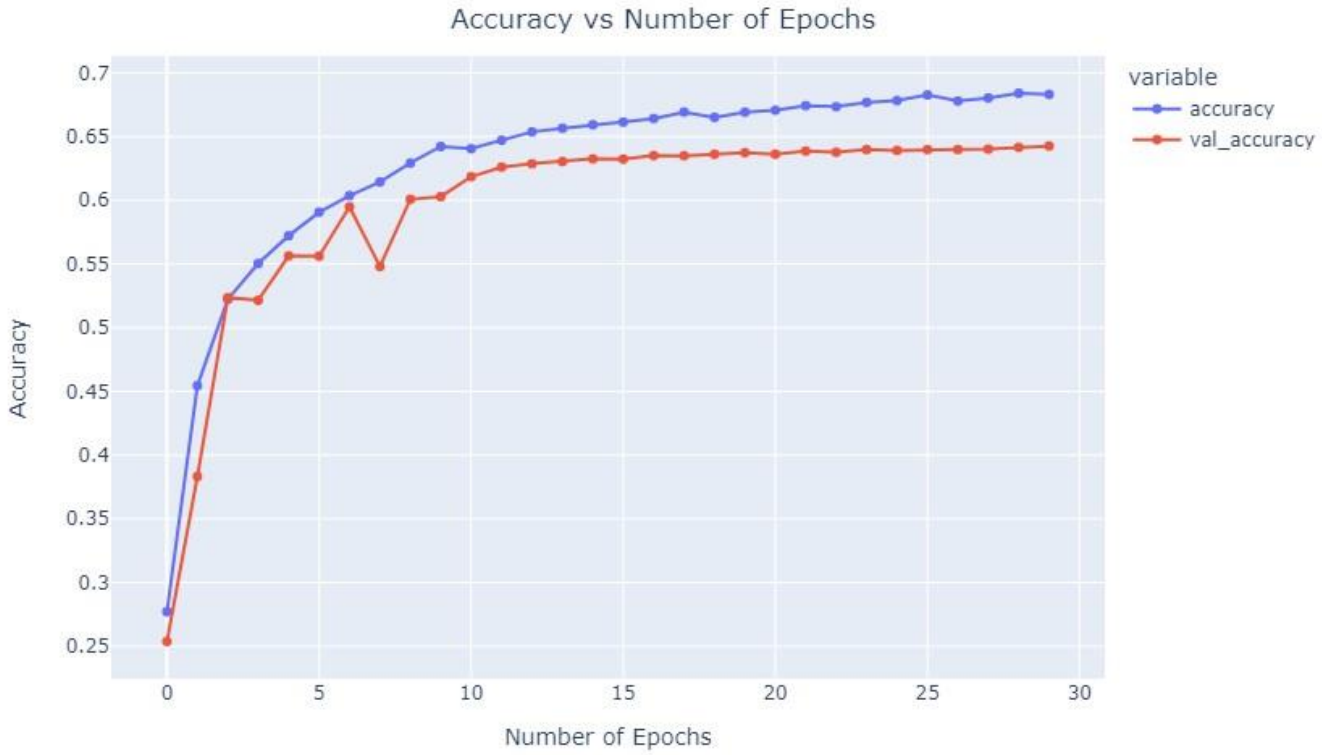

Рисунок 2.4 Точність та кількості епох

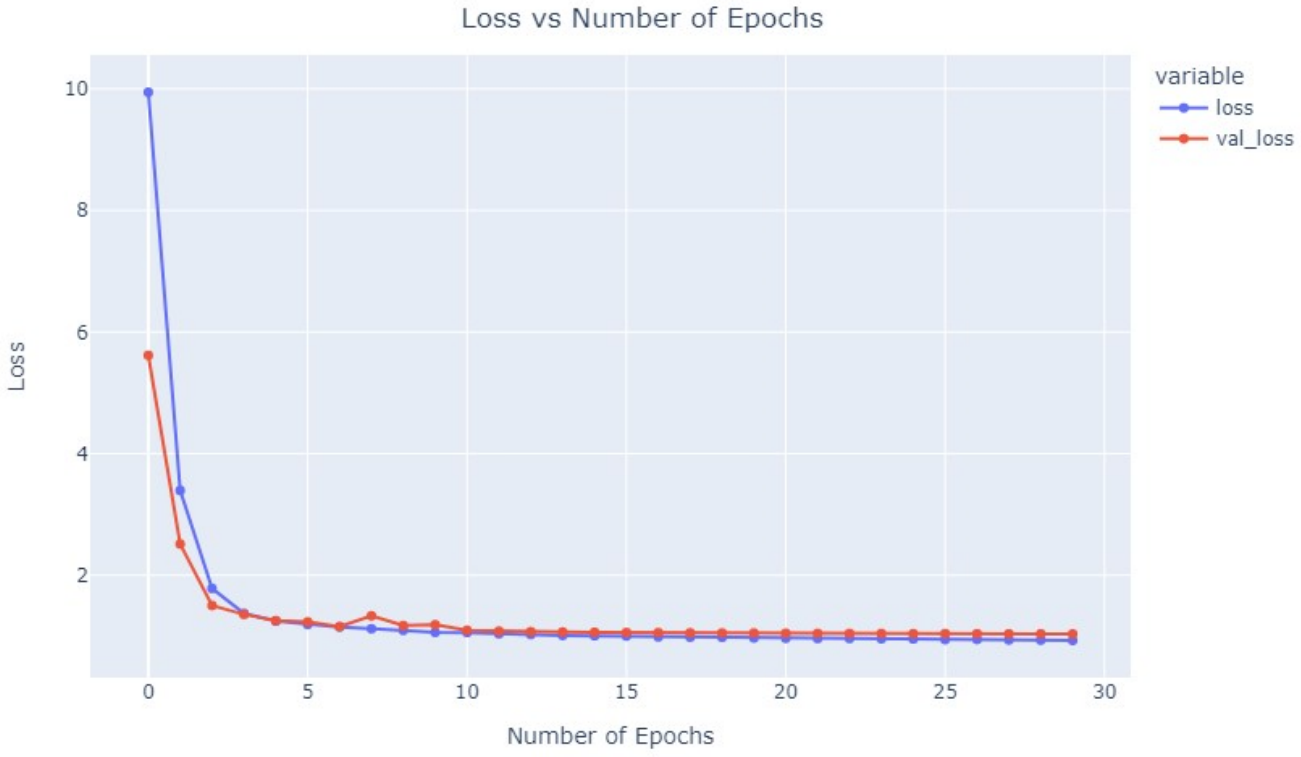

Рисунок 2.5 Втрати та кількість епох

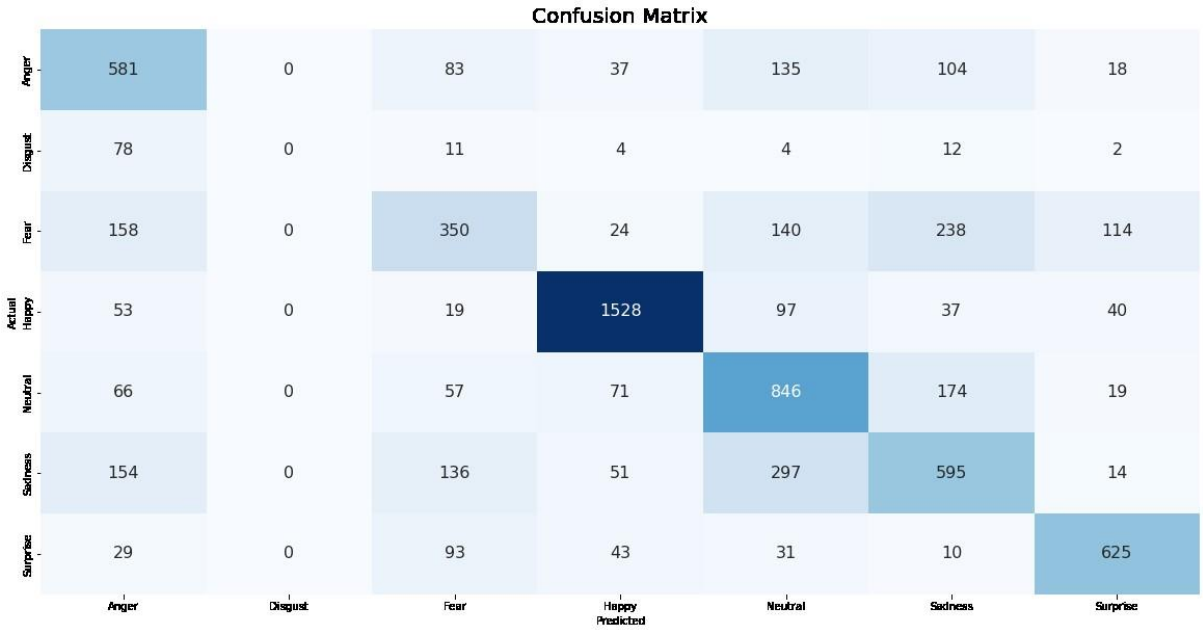

Рисунок 2.6 Матриця помилок

	precision	recall	f1-score	support
0	0.52	0.61	0.56	958
1	0.00	0.00	0.00	111
2	0.47	0.34	0.39	1024
3	0.87	0.86	0.87	1774
4	0.55	0.69	0.61	1233
5	0.51	0.48	0.49	1247
6	0.75	0.75	0.75	831
accuracy			0.63	7178
macro avg	0.52	0.53	0.52	7178
weighted avg	0.62	0.63	0.62	7178

Рисунок 2.7 Звіт класифікації

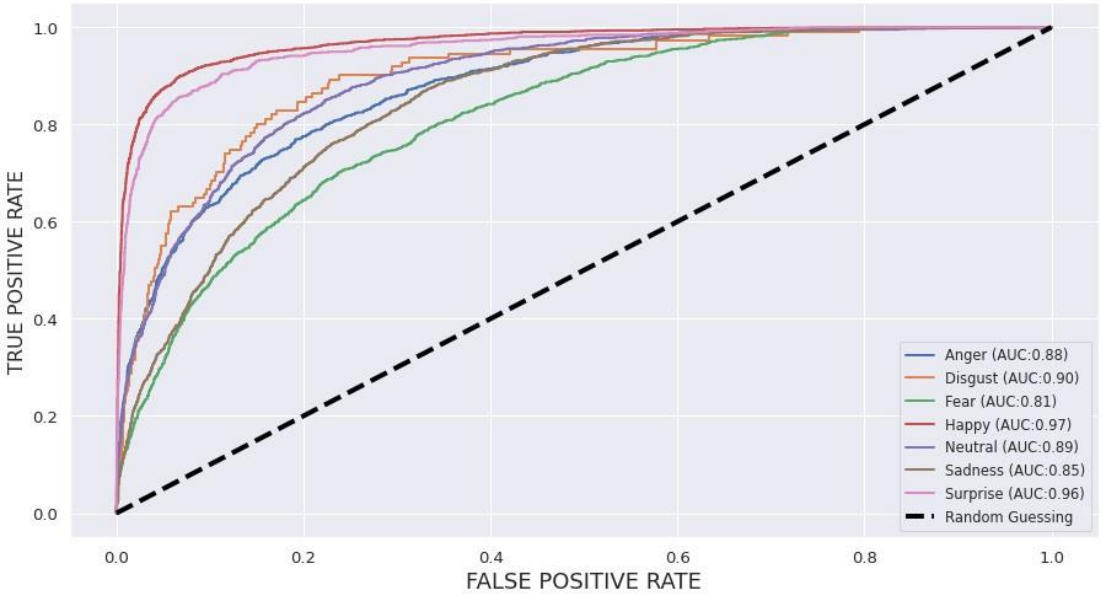

Рисунок 2.8 Мультикласова крива AUC.

2.2 MLFlow для відстеження параметрів та створення моделей

MLflow - це платформа з відкритим вихідним кодом для управління наскрізним життєвим циклом машинного навчання. Вона виконує три основні функції:

Використовуючи MLflow, її користувацький інтерфейс для візуалізації записаних експериментів. Команда `mlflow ui` запускає локальний сервер на порту 5000 (за замовчуванням), і тут можна взаємодіяти з ним через веб-браузер.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(venv) PS C:\Users\MaxSt\PycharmProjects\Diploma\HumanverifyTrainModel> mlflow ui
INFO:waitress:Serving on http://127.0.0.1:5000
```

Рисунок 2.9 Запуск локального сервера моделі

2.2.1 Відстеження експериментів

При реєстрації метрики, параметрів та артефактів у коді, MLflow записує їх у файли, щоб потім можна було відтворити результати і візуалізувати хід роботи. MLflow створює новий каталог для кожного прогону у визначеному каталозі експериментів і записує дані до цього каталогу.

Дані експерименту зберігаються у файлах у наступній ієрархії:

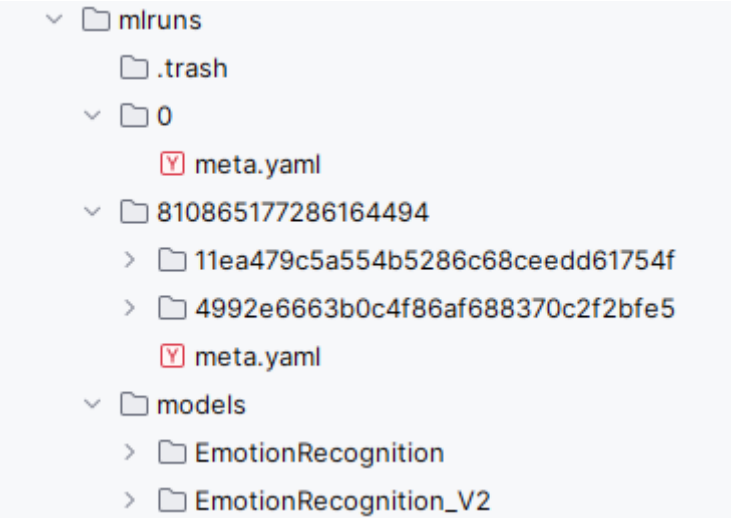

Рисунок 2.10 Ієрархія MLFlow каталогу

2.2.2 MLflow UI

Після запуску сервера можна отримати доступ до інтерфейсу MLflow, відкривши `localhost:5000` у веб-браузері. Нижче наведено загальний огляд:

- **Відстеження експерименту:** можна візуалізувати різні прогони для кожного експерименту, порівнювати їх і фільтрувати за параметрами і метриками. Для кожного прогону записуються параметри, метрики, джерело, версія, час початку і закінчення прогону. Приклад на власних моделях

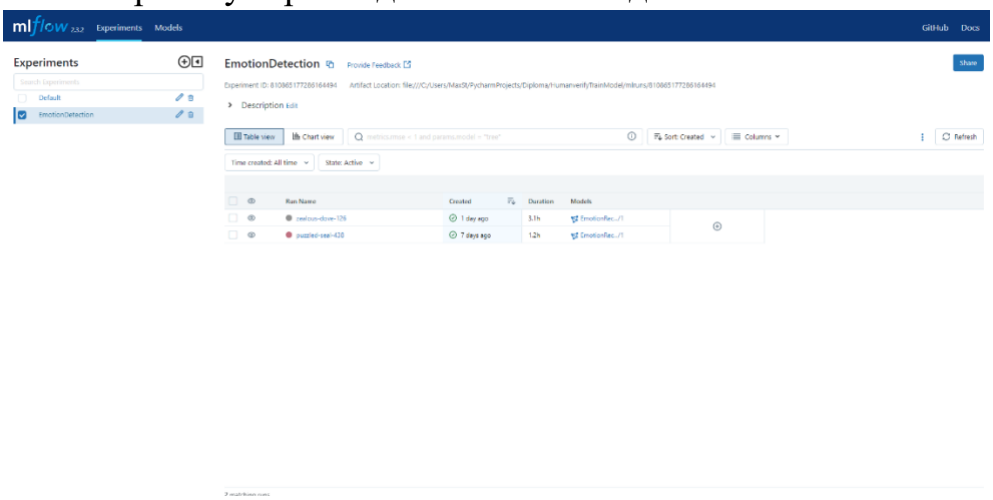

Рисунок 2.11 Відстеження експерименту

- **Деталі пробігу:** Можна натиснути на кожен цикл, щоб переглянути більш детальну інформацію про цикл. Вона включає візуалізації для зареєстрованих метрик, повний список зареєстрованих параметрів, список зареєстрованих артефактів і всі записані теги.

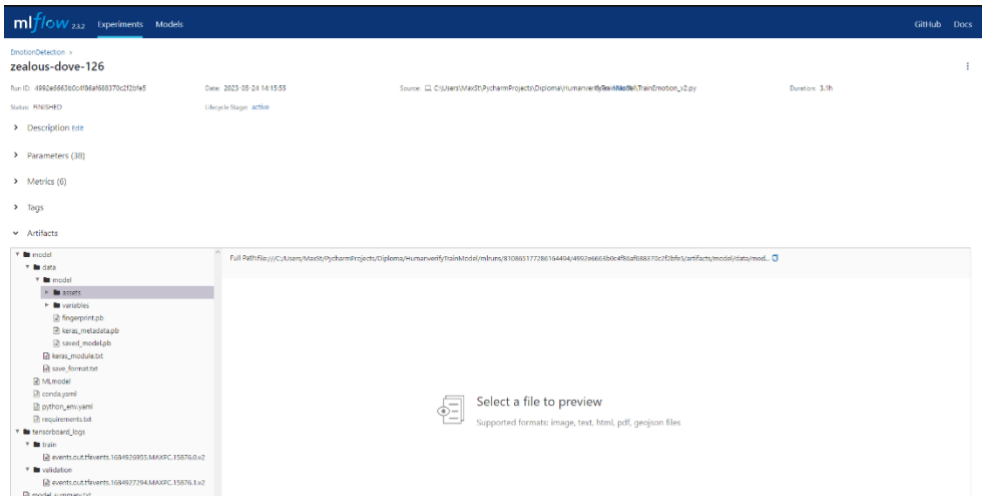

Рисунок 2.12 Деталі MLFlow моделі

- **Реєстр моделей**: Реєстр моделей MLflow надає централізоване сховище моделей, набір API та інтерфейс користувача для спільного керування повним життєвим циклом моделі MLflow.

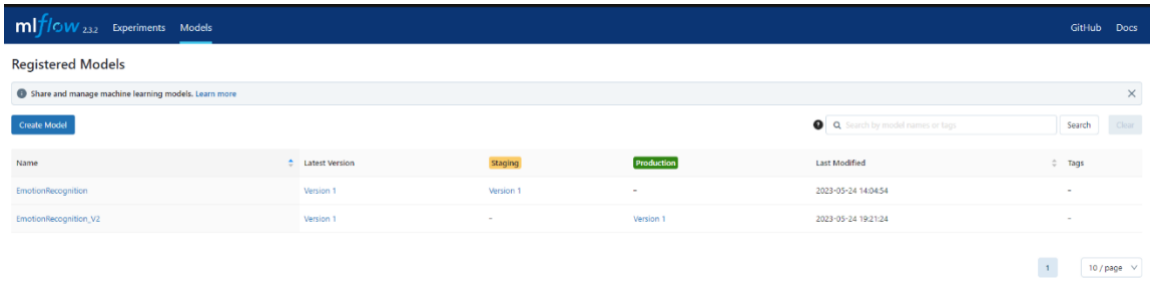

Рисунок 2.13 Зареєстровані MLFlow моделі

Реєстр моделей MLflow дозволяє командам керувати життєвим циклом своїх моделей машинного навчання, включаючи етапи. Кожна версія моделі може пройти кілька етапів протягом свого життєвого циклу. Ці етапи представляють життєвий цикл моделі, який може включати такі стадії, як "***None***", "***Staging***", "***Production***" та "***Archived***".

Ось короткий опис цих стадій:

- **None:** Типова версія перебуває на стадії `***None***`, коли її вперше зареєстровано. Це означає, що версія знаходиться на стадії чернетки або експериментальної версії.
- **Staging:** Коли типова версія знаходиться на стадії " ***Staging*** ", вона є кандидатом на запуск у виробництво. Ця стадія призначена для тестування і розробки.
- **Production:** Версія моделі на стадії "***Production***" вважається готовою до використання у виробництві. Зазвичай на стадії виробництва знаходиться лише одна версія моделі.
- **Archived:** Стадія `***Archived***` призначена для версій, які більше не використовуються. Версію моделі можна перемістити до ` ***Archived*** ` з будь-якої стадії. Версію ` *Archived* ` може бути розархівовано назад до попередньої стадії.

Стадія версії моделі є змінною і може бути переведена на іншу стадію відповідно до статусу її життєвого циклу. Перехід версій моделі з однієї стадії на

іншу можна здійснити вручну через інтерфейс Реєстру моделей або програмно, використовуючи API MLflow.

Ось приклад того, як перевести версію моделі на іншу стадію програмно:

```
# Import MLflow
import mlflow

# Set the model name and version
model_name = "MyModel"
model_version = 1

# Set the stage to "Production"
mlflow.register_model(
    name=f"models:/{model_name}/{model_version}",
    stage="Production",
)
```

Рисунок 2.14 Програмна реалізація стадіювання моделі

Переносячи версії моделі між етапами, можна ефективно керувати процесом розгортання моделі та оптимізувати його. Наприклад, можна спочатку зареєструвати нову версію моделі на стадії `***Staging*** `, а потім перенести її на стадію ` ***Production***` після тестування. Якщо ця версія моделі буде вже не потрібна то, її можна пізніше перенести на стадію ` ***Archived***`.

Реєстр моделей MLflow дозволяє командам керувати, відстежувати і навіть автоматизувати ці переходи, щоб забезпечити послідовний і надійний життєвий цикл машинного навчання.

Стадії версій моделі в процесі створення і налагодження їх:

Name	Latest Version	Staging	Production
EmotionRecognition	Version 1	Version 1	-
EmotionRecognition_V2	Version 1	-	Version 1

Рисунок 2.15 Версії MLFlow моделі і їх розміщення на етапах

2.3 Flask додаток з MLflow для IOS запитів

Створення програми Flask, яка здатна обробляти запити, що надходять від програми iOS яка передбачає емоції. Практичним прикладом використання в цьому сценарії буде додаток, який використовує машинне навчання для розпізнавання емоцій на основі виразу обличчя.

Файл `***app.py***` ініціює додаток. Цей файл використовує веб-фреймворк Flask та інші бібліотеки, зокрема MLflow для керування моделями, NumPy для числових операцій, OpenCV для обробки зображень та base64 для кодування і декодування даних.

```
from flask import Flask, request, jsonify, make_response
import mlflow.pyfunc
import numpy as np
import cv2
import os
import base64
```

Рисунок 2.16 Імпорти python модулів

Цей сегмент коду ініціалізує додаток Flask і налаштовує MLflow для відстеження моделі. Далі він завантажує модель EmotionRecognition з локального репозиторію моделей MLflow.

```
app = Flask(__name__)

emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"}

# Set MLflow Tracking URI
mlflow.set_tracking_uri("mlruns")

# Load MLflow model
model_name = "EmotionRecognition"
model_version = 1
model_path = f"models:/{model_name}/{model_version}"

model = mlflow.pyfunc.load_model(model_path)
```

Рисунок 2.17 Сегмент коду налаштувань MLFlow моделі

Функція ***`predict`,*** яка відображається на URL-адресу ***'/predict'***, є ядром цього додатка. Ця функція отримує дані зображення з POST-запиту, декодує їх і обробляє для виявлення емоцій. Якщо виявлено обличчя, функція обрізає його з зображення,

змінює розмір, нормалізує його, а потім завантажує в завантажену модель MLflow для прогнозування. Передбачена емоція повертається у відповіді.

```python
@app.route('/predict', methods=['POST'])
def predict():
    try:
        img_data = request.form['img_data']
        img_data = base64.b64decode(img_data)
        nparr = np.frombuffer(img_data, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # Find haar cascade to draw bounding box around face
        face_detector = cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_default.xml')
        faces = face_detector.detectMultiScale(gray_img, scaleFactor=1.3, minNeighbors=5)

        if len(faces) > 0:
            x, y, w, h = faces[0]
            roi_gray_img = gray_img[y:y + h, x:x + w]
            roi_gray_img = cv2.resize(roi_gray_img, (48, 48))
            roi_gray_img = roi_gray_img.reshape(1, 48, 48, 1)
            roi_gray_img = roi_gray_img.astype('float32') / 255

            with mlflow.start_run() as run:
                predictions = model.predict(roi_gray_img)
                maxindex = int(np.argmax(predictions))
                emotion = emotion_dict[maxindex]

                # Log model performance
                mlflow.log_param("emotion", emotion)

            return jsonify(x=int(x), y=int(y), w=int(w), h=int(h), emotion=emotion)
        else:
            return jsonify(error='No face detected')
    except Exception as e:
        print(f"Error: {e}")
        return make_response(jsonify(error='Error occurred during processing'), 500)
```

Рисунок 2.18 Функція ***predict***

Останній сегмент запускає програму, вказуючи їй прослуховувати вказаний порт для вхідних з'єднань.

```python
if __name__ == '__main__':
    port = int(os.environ.get("PORT", 5001))
    app.run(host="0.0.0.0", port=port)
```

Рисунок 2.19 Сегмент запуску Flask сервера

2.4 IOS додаток передбачення емоцій

IOS - додаток для розпізнавання емоцій користувача. Додаток використовує камеру пристрою для зйомки відеокадрів, а потім надсилає ці кадри на Flask сервер, який аналізує їх на наявність рис обличчя і повертає емоцію.

Дизайн програми використовує архітектурний патерн Model-View-ViewModel (MVVM), де SwiftUI використовується для розробки інтерфейсу користувача (View), `CameraViewModel` виступає в ролі ViewModel, а структура `EmotionResponse` представляє модель.

Структуризація теки файлів виглядає ось так:

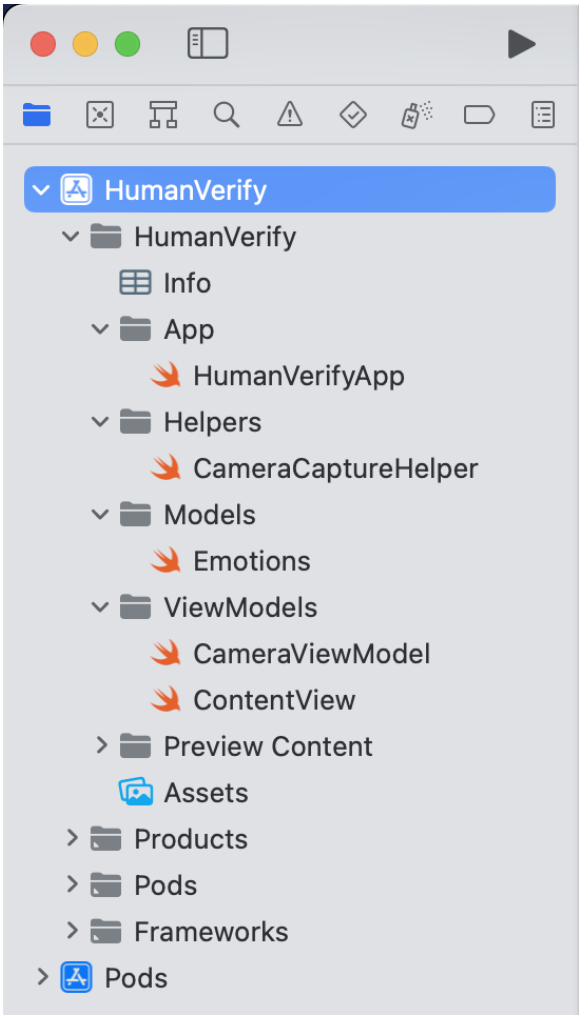

Рисунок 2.20 Тека файлів IOS додатка

Для більшого розуміння технологій і функцій розглянемо кожен файл окремо.

2.4.1 Точка входу в додаток HumanVerify для iOS

Файл `HumanVerifyApp.swift` є точкою входу в додаток. З нього починається виконання програми.

У цьому файлі оголошується нова структура *`HumanVerifyApp`*, яка відповідає протоколу *`App`*. SwiftUI викликає властивість *`body`*, коли він готовий відобразити інтерфейс користувача програми. **WindowGroup** - це сцена для програми, яка масштабується для підтримки будь-якої кількості вікон. Всередині *`WindowGroup`*, *`ContentView()`* встановлюється як кореневе подання.

У контексті програми, коли програма запускається, SwiftUI перегляне цей файл, побачить, що *`ContentView`* є кореневим поданням, і завантажить це подання та його підподання відповідно.

```
8  import SwiftUI
9
10 @main
11 struct HumanVerifyApp: App {
12     var body: some Scene {
13         WindowGroup {
14             ContentView()
15         }
16     }
17 }
```

Рисунок 2.21 Код HumanVerifyApp.swift

У цьому коді *`@main`* - це атрибут, який ідентифікує точку входу програми. Властивість *`body`* повертає *`Scene`*, яка представляє один екземпляр користувацького інтерфейсу програми. Властивість *`ContentView`* є основним представленням для програми, і вона розміщується всередині *`WindowGroup`* для обробки створення нових вікон для платформ, які їх підтримують.

2.4.2 Налаштування камери для додатка

Файл, *`CameraCaptureHelper.swift`*(див.додат.Б), відповідає за налаштування та керування камерою на пристрої. Він використовує *`AVFoundation`* для налаштування сесії захоплення камери, щоб отримати відеопотік від камери.

Основний клас у цьому файлі - *`CameraCaptureHelper`*. Він ініціалізує сесію захоплення *`AVCaptureSession`* та налаштовує вхід та вихід для камери.

Функція *`initialiseCaptureSession()`* налаштовує сесію захоплення. Спочатку вона налаштовує роздільну здатність для сесії за допомогою *`AVCaptureSession.Preset.hd1280x720`,* що встановлює високу роздільну здатність 1280x720.

Потім вона використовує *`AVCaptureDevice.DiscoverySession`* для отримання доступу до камери на пристрої.

Далі вхід з камери додається до сесії захоплення. У випадку виникнення помилки, програма завершує свою роботу з повідомленням про помилку *"Unable to access back camera"*.

Після цього налаштовується вихід відеоданих *`AVCaptureVideoDataOutput`* і додається до сесії.

Розширення *`CameraCaptureHelper*: *AVCaptureVideoDataOutputSampleBufferDelegate`* використовує метод *`captureOutput(_:didOutput:from:)`,* щоб обробляти кадри, отримані від камери.

Цей метод викликається кожного разу, коли камера знімає новий кадр. Він використовує *`CMSampleBufferGetImageBuffer(sampleBuffer)`,* щоб отримати відеокадр як `CVPixelBuffer`. Потім він конвертує цей відеокадр у *`CIImage`* та передає його делегату *`newCameraImage(_ cameraCaptureHelper: CameraCaptureHelper, image: CIImage)`*.

Останній протокол `CameraCaptureHelperDelegate` вимагає, щоб делегат міг обробляти нові зображення з камери, як це визначено в методі *`newCameraImage(_ cameraCaptureHelper: CameraCaptureHelper, image: CIImage)`*.

2.4.3 Запит на дані

Файл *`Emotion.swift`* визначає структуру *`EmotionResponse`*, яка представляє відгук на виявлення емоцій. Ця структура приймає JSON-відгук від Flask API розпізнавання емоцій та розбиває його на декілька полів.

Він використовує *`Decodable`* протокол, який дозволяє йому легко декодувати JSON відповідь в цю структуру. Кожне поле у відповіді представлене в структурі:

- *`x`*, *`y`*, *`w`*, *`h`* представляють координати та розмір області, де було виявлено обличчя на зображенні. `x` і `y` - це координати верхнього лівого кута області, а `w` і `h` - ширина та висота області відповідно.
- *`emotion`* містить виявлену емоцію.
- *`error`* містить повідомлення про помилку, якщо воно є.

За замовчуванням всі поля є опціональними (`?`), оскільки вони можуть бути відсутні в відповіді API.

```
8  import Foundation
9
10 struct EmotionResponse: Decodable {
11     let x: Int?
12     let y: Int?
13     let w: Int?
14     let h: Int?
15     let emotion: String?
16     let error: String?
17 }
```

Рисунок 2.22 Файл Emotion.swift

2.4.4 Модель камери для виявлення емоцій

Файл - *`CameraViewModel.swift`*(див.додат.Б) - виконує роль посередника між камерою та інтерфейсом користувача. Це він обробляє дані з камери, розпізнає обличчя, а потім передає дані на сервер для аналізу емоцій. Він також обробляє відповідь сервера та оновлює інтерфейс користувача.

Розглянемо основні фрагменти коду:

- **Оголошення класу та протоколів:**`*CameraViewModel*` клас є нащадком класу `*NSObject*` і він реалізує `*ObservableObject*`, `*CameraCaptureHelperDelegate*` та `*AVCaptureVideoDataOutputSampleBufferDelegate*`протоколи.
  `*ObservableObject*` використовується для відслідковування змін у даних, які потрібно відобразити в представленні (*View*). `*CameraCaptureHelperDelegate*` та `*AVCaptureVideoDataOutputSampleBufferDelegate*` використовуються для отримання вхідних даних з камери.
- **Оголошення змінних:** В класі оголошено декілька змінних та властивостей, включаючи `*session*`, `*detectedFace*`, `*emotionText*`, `*output*` та інші. Ці змінні використовуються для контролю сесії камери, зберігання виявлених даних обличчя та тексту емоцій, а також для інших задач, пов'язаних з роботою з камерою.
- **Методи для керування сесією:** Методи `*startSession*` та `*stopSession*` використовуються для запуску та зупинки сесії камери.
- **Налаштування сесії:** В методі `*configureSession*` перевіряється доступ до камери. Якщо доступ дозволено, то викликається метод `*setupSession*`, який ініціалізує камеру та вихідні дані відео.
- **Обробка зображення з камери:** В методі `*captureOutput*` зображення з камери конвертується в `*CIImage*`, а потім відправляється на сервер кожний 8-й кадр.
- **Отримання відповіді від сервера:** У методі `*newCameraImage*` зображення, отримане з камери, конвертується в `*UIImage*`, а потім в `*Data*`. Ці дані кодуються в Base64 і відправляються на сервер за допомогою POST-запиту. Відповідь від сервера обробляється, оновлюючи інтерфейс користувача з відповідною емоцією та областю обличчя.

Отже цей файл демонструє взаємодію з камерою, відправку зображень на сервер для аналізу емоцій та оновлення інтерфейсу користувача на основі відповіді сервера.

2.4.5 Головний інтерфейс додатку

Файл `*ContentView.swift*`(див.додат.Б) містить код для головного інтерфейсу користувача додатка HumanVerify. Інтерфейс реалізовано з використанням SwiftUI і включає в себе елементи керування для роботи з камерою та відображення результатів виявлення емоцій.

Імпорт модулів:

- `*SwiftUI*` для створення інтерфейсу користувача.
- `*AVFoundation*` для роботи з камерою.

Основна структура `ContentView`, яка є головним видом додатка. Вона містить:

- `*@StateObject private var cameraViewModel = CameraViewModel()*`: змінна для моделі представлення, яка управляє камерою та аналізом обличчя.
- `*@State private var showCamera = false*`: змінна стану, яка контролює видимість виду камери.
- Функцію `*body*`, яка визначає вид, який буде відображатися. У залежності від стану `*showCamera*` відображається або камера з нашаруванням для обличчя та відображенням емоцій, або основний вид з кнопкою для включення камери.

Структура `*CameraView*`, яка використовує `*UIViewControllerRepresentable*` для використання `*UIViewController*` у SwiftUI:

- Функцію `*makeUIViewController*`, яка створює `*UIViewController*` і додає шар передпоказу відео.
- Функцію `*updateUIViewController*`, яка оновлює шар передпоказу відео при зміні розмірів вигляду.

Структура `*OverlayShape*`, яка визначає вид для нашарування обличчя на відео з камери. Вона використовує параметр `*rect*`, щоб визначити положення і розмір рамки обличчя.

Визначення `*ContentView_Previews*` для попереднього перегляду вигляду `*ContentView*` у Xcode.

Розділ 6. Демонстрація та аналіз отриманих результатів

Вибрав модель і запустив Flask сервер:

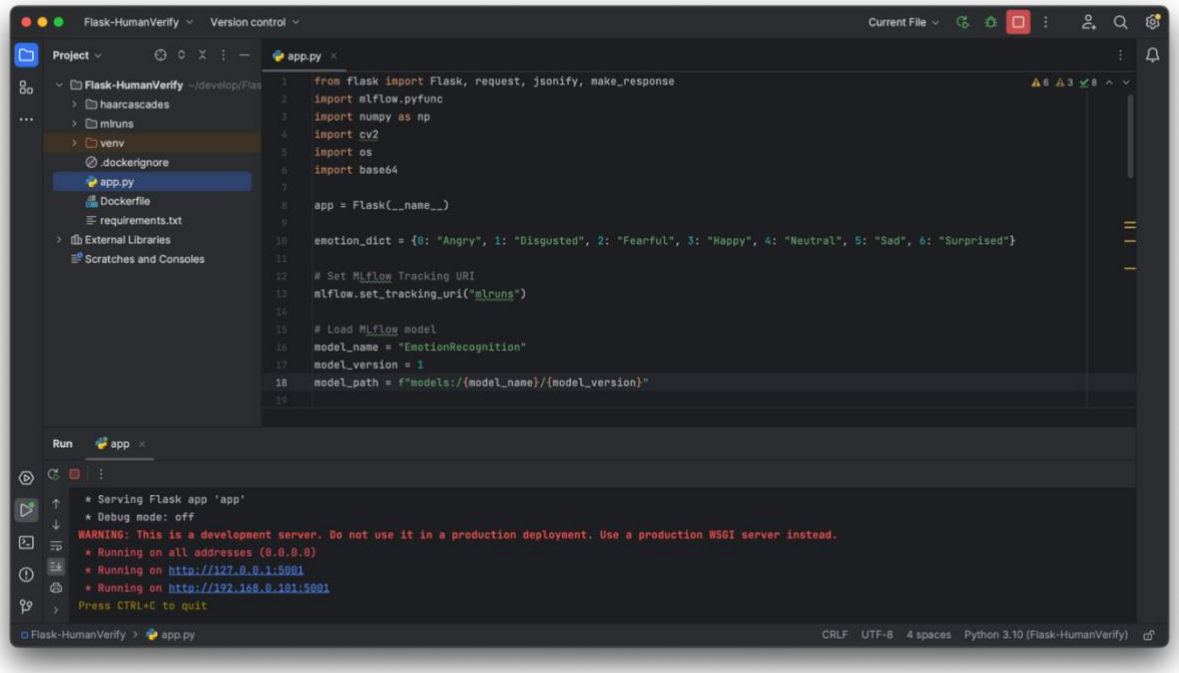

Рисунок 2.23 Запуск Flask сервера

Запустив клієнтську частину:

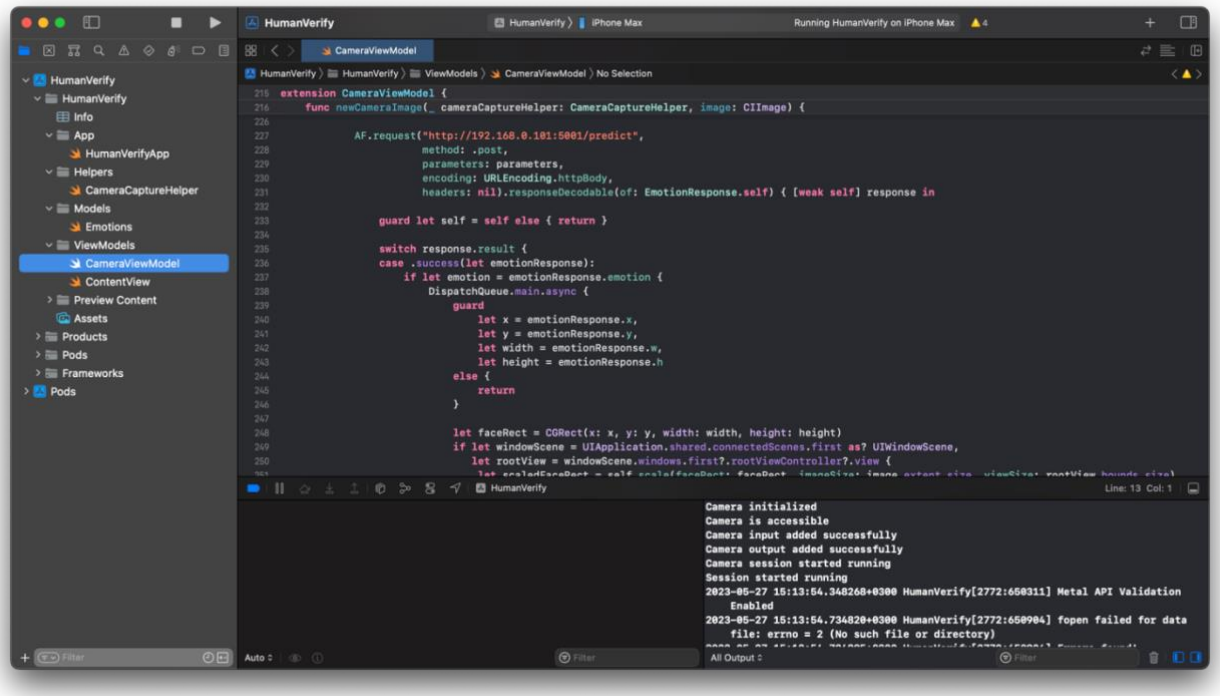

Рисунок 2.24 Запуск IOS клієнтської частини

Запустив IOS застосунок:

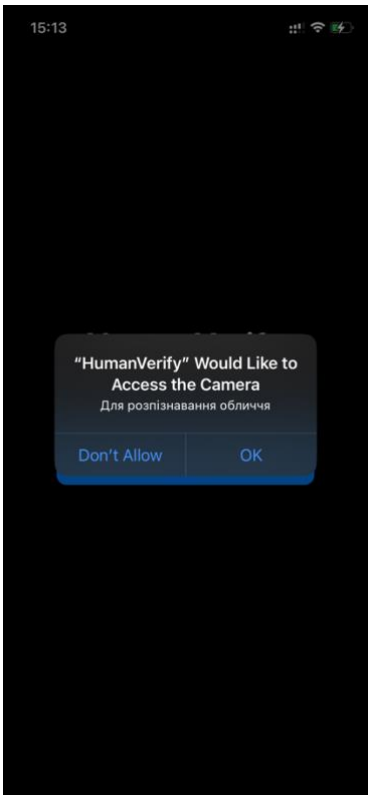

Рисунок 2.25 Дозвіл

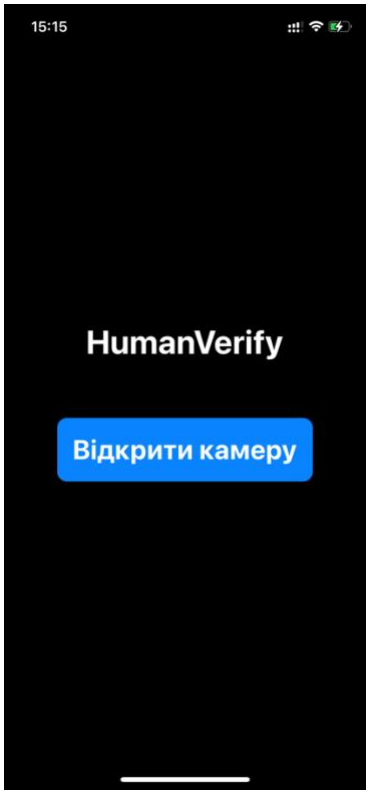

Рисунок 2.26 Головний екран

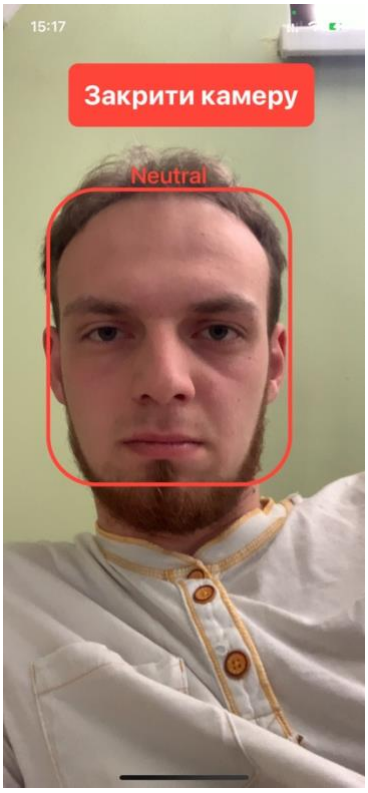

Рисунок 2.27 Екран передбачення

Отже, проаналізувавши дані, можна зробити висновок, що все працює правильно. Для більш автономної роботи в майбутньому, можна розгорнути Flask-сервер на віддаленому сервері.

ВИСНОВОК

Виконання цієї бакалаврської роботи призвело до успішної розробки додатку для iOS, здатного розпізнавати емоції обличчя в реальному часі, демонструючи потенціал та ефективність методів машинного навчання в практичних, реальних сценаріях.

Спочатку завдання було пов'язане з кількома проблемами, головним чином, з ефективним розгортанням моделей машинного навчання та обробкою даних у режимі реального часу. Однак завдяки використанню надійних інструментів, таких як Flask для обробки запитів, Xcode для розробки IOS додатку та MLFlow для управління процесами машинного навчання, ці проблеми були подолані.

Значення цієї роботи полягає в тому, що вона демонструє спрощений робочий процес розгортання моделей машинного навчання в мобільному додатку. Це особливо актуально з огляду на сучасну глобальну тенденцію до діджиталізації та все більш широкого використання штучного інтелекту в мобільних додатках.

Практичні результати цієї роботи є багатогранними. По-перше, розроблений додаток є реальним продуктом, що демонструє практичне застосування методів машинного навчання. По-друге, використання MLFlow у цьому проекті підкреслює важливість управління та моніторингу моделей машинного навчання - практики, яка стає все більш важливою, оскільки все більше організацій прагнуть розгортати ці моделі в масштабах.

У світлі висновків, отриманих в результаті цього проекту, можна зробити кілька рекомендацій щодо їх ефективного використання та подальшої роботи. По-перше, для організацій, які прагнуть розгорнути моделі машинного навчання, використання такого інструменту, як MLFlow, може значно спростити процес розгортання і підвищити відтворюваність результатів. По-друге, подальші дослідження можуть бути спрямовані на інтеграцію більш просунутих функцій в додаток, таких як здатність розпізнавати кілька емоцій одночасно або функціонувати в різних умовах освітлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introducing mlops: how to scale machine learning in the enterprise / M. Treveil et al. O'Reilly Media, Incorporated, 2020.
2. McMahon A. P. Machine learning engineering with python: manage the production life cycle of machine learning models using mlops with practical examples. Packt Publishing, Limited, 2021.
3. MLflow - A platform for the machine learning lifecycle. *MLflow*. URL: https://mlflow.org.
4. Munn M., Robinson S., Lakshmanan V. Machine learning design patterns: solutions to common challenges in data preparation, model building, and mlops. O'Reilly Media, Incorporated, 2020. 400 s.
5. New technologies WWDC22 | apple developer documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/New-Technologies-WWDC22.
6. Singh P. Deploy machine learning models to production. Berkeley, CA : Apress, 2021. URL: https://doi.org/10.1007/978-1-4842-6546-8.