КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

WEB-ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ ОБЛИЧЧЯ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

Кваліфікаційна робота бакалавра

студента 4 року навчання

Спеціальність: 123 «Комп'ютерна інженерія»

Олександра ГУЛІЦЬКОГО

_____(підпис)

Науковий керівник,

к.т.н. Олександр САМОЩЕНКО

доцент кафедри

комп'ютерної інженерії

_____(підпис)

Рецензент:

к. ф.-м. н Іван КОЛОМІЄЦЬ

асистент факультету радіофізики,

електроніки та комп'ютерних систем

_____(підпис)

До захисту допускаю

Завідувач кафедри Юрій БОЙКО

Ухвалено на засіданні кафедри "_____"_____202__р., протокол № _____

КИЇВ - 2022

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра містить 64 сторінки, 47 рисунків, 1 таблицю, 1 додаток, використано 18 інформаційних джерел.

Об‘єкт дослідження – способи детекції облич на статичних зображення та зображеннях у реальному часі.

Мета роботи – розглянути існуючі види нейронних мереж, їх особливості та відмінності, побудувати модель на основі згорткових нейронних мереж та інтегрувати її в застосунок.

В роботі представлені існуючі архітектури які основані на згорткових нейронних мережа. Проведений аналіз різних мов програмування, користувацьких фреймворків, засобів для використання нейронних мереж. Реалізований веб-застосунок, що дозволяє проводити аналіз зображень відповідно до мети роботи.

ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, CNN, WEB-ЗАСТОСУНОК, РОЗПІЗНАВАННЯ ОБЛИЧ, JAVASCRIPT, REACTJS, TENSORFLOW, BLAZEFACE.

ЗМІСТ

ВСТУП

Протягом останніх років розпізнаванню облич приділяється достатньо уваги і оцінюється як одна з найбільш перспективних напрямків у сфері аналізу зображень. Виявлення облич (face detection) може займати значну частину операцій розпізнавання облич (face recognation). Метод визначення обличчя на зображеннях є складною задачею через різноманітність людських облич, наприклад поза в якій зроблено знімок, вираз обличчя, колір шкіри, наявність окулярів чи волосся, відмінності в фільтрах накладених на знімок, умовах освітлення та роздільній здатності зображення.

Виявлення об‘єктів – це одна з комп‘ютерних технологій, яка пов’язана з обробкою зображень і комп’ютерним баченням і взаємодіє з виявленням таких об’єктів, як людські обличчя, будівля, дерево, автомобіль тощо. Основною метою алгоритмів виявлення обличчя є визначення чи є обличчя на зображенні чи ні.

Останнім часом зроблено багато дослідів у сфері розпізнавання та виявлення облич, щоб зробити цю технологію більш розвинутою та точнішою. Революція в даній галузі відбулася коли Віола-Джонс прийшов з своїм детектором, який здатний виявляти обличчя в режимі реального часу з високою точністю.

Виявлення облич (face detection) — це перший і важливий крок для розпізнавання облич, який використовується для виявлення облич на зображеннях. Він є частиною виявлення об’єктів і може використовуватися в багатьох сферах, таких як безпека, біометрика, правоохоронні органи, розваги, особиста безпека, тощо.

Він використовується для виявлення облич у режимі реального часу для спостереження та відстеження людини або об’єктів. Він широко

використовується в фотоапаратах для ідентифікації кількох появ у кадрі Ex-Mobile камер і DSLR. Facebook також використовує алгоритм розпізнавання облич, щоб виявляти обличчя на зображеннях і розпізнавати їх.

РОЗДІЛ 1
ОСНОВНІ ВІДОМОСТІ ПРО НЕЙРОННІ МЕРЕЖІ

1.1 Класифікація нейронних мереж

Доволі частіше для вирішення реальних проблем в штучному інтелекті використовуються нейронні мережі. Що ж таке нейронна мережа? В одному з визначень вказано, що це деяка послідовність нейронів, кожен з яких отримую певну кількість даних, обробляє її та передає наступному нейрону. Нейрони зв’язані між собою синапсами. Саме вони дозволяють отримати різний результат, шляхом посилення або послаблення сигналу. Можна представити що нейронна мережа це спрощена модель біологічного аналога, наприклад людського мозку. Зазвичай нейромережі поділяють на різні типи за декількома признаками, а саме:

1. В залежності від типу нейронів:
    - Однорідні
    - Гібридні
2. В залежності від методу навчання:
    - Навчання з вчителем
    - Без вчителя
    - З підкріпленням
3. По типу вхідної інформації:
    - Аналогові
    - Двійкові
    - Образні
4. По характеру налаштування синапів:
    - З фіксованим зв’язком
    - З динамічним зв’язком

На рисунку 1.1 зображено діаграму класифікації нейронних мереж.

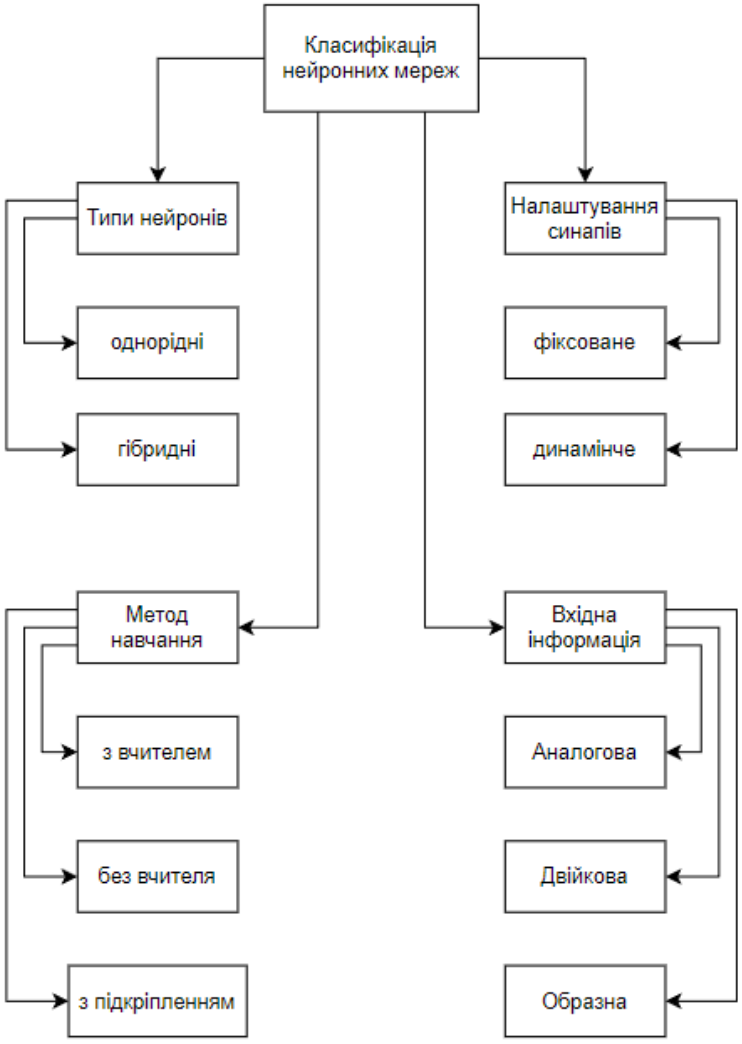

Рисунок 1.1 Діаграма класифікації нейронних мереж.

1.2 Види нейронних мереж

В даному розділі буде розглянуто основні відмінності найбільш популярних нейронних мереж.

Перцептрон (Perceptron) – це контрольований алгоритм, який класифікує вхідні дані всього на дві категорії, тобто можна вважати що це двійковий класифікатор. На вхід він приймає зважені вхідні дані(weighted inputs),

застосовує функцію та віддає кінцевий результат. Це одна з найстаріших і водночас найпростіших моделей, яка була запропонована Мінським-Папертом. В наш час, перцептрон вважають найменшою одиницею нейронної мережі, яка виконує деякі обчислення, щоб виявити особливість на основі вхідних даних. Також його називають пороговою логічною одиницею (TLU).

Переваги перцептрона:

- можна реалізувати прості логічні операції, такі як І, АБО, НІ

Недоліки:

- так як це двійковий класифікатор, він може використовуватися тільки для вирішення лінійних задач, тому його використання зустрічається доволі рідко

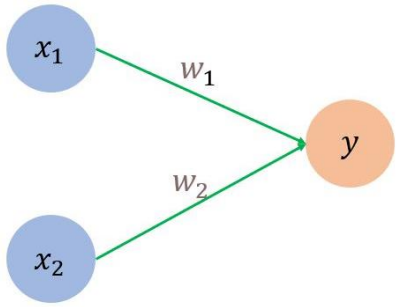

Рисунок 1.2 Перцептрон

Багатошаровий перцептрон (Multi-Layer Perceptron) - це набір перцептронів і тому вона вже є нелінійною і тим самим може виконувати більш комплексні задачі. Багатошаровий перцептрон може мати декілька вхідних шарів та один або декілька прихованих шарів, які поєднанні між собою. Ця мережа відноситься до алгоритмів прямого зв’язку, які будуть розглянуті пізніше. Кожен

вузол має зв’язок з всіма нейронами наступного шару, що робить його повноцінною нейронною мережею.

Принцип роботи полягає в тому, що початкові дані поєднуються з ваговими коефіцієнтами, після чого застосовується функція активації, тобто алгоритм роботи такий самий як перцептроні, але основна відмінність в тому, що ця комбінація поширюється далі на наступний шар і кожен шар обчислює і віддає свої результати. Але якби це був весь процес навчання, то точність результатів була низькою і тому сюди додається зворотне поширення (backpropagation). Це механізм навчання, який дозволяє ітеративно регулювати ваги в мережі, за для мінімізації функції вартості.

Переваги:

- використовуються для глибокого вивчення, завдяки зворотного поширення та зв’язку між шарами

Недоліки:

- доволі складні у проектуванні

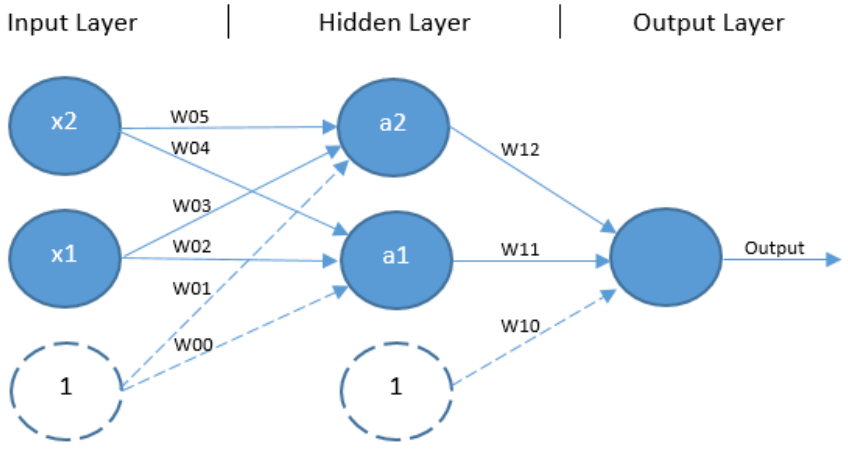

Рисунок 1.3 Багатошаровий перцептрон

Нейронна мережа прямого поширення (Feed Forward Neural Network) - найпростіша форма нейронних мереж. Принцип роботи – вхідні дані рухаються в одному напрямку через штучні нейронні вузли і виходять через вихідні вузли, звідси і взялася назва даної мережі. Прихованні шари не є обов'язковими в даній нейромережі і тому вона може бути як і одношарова так і багатошарова. Кількість шарів залежить від складності поставленої задачі. Якщо порівнювати мережу прямого поширення з багатошаровим перцептроном, то основна відмінність це відсутність зворотного поширення, тим самим ваги стають статичними і це призводить до більш повільного навчання.

Можливі області застосування:

- проста класифікація об'єктів
- розпізнавання обличчя
- розпізнавання мови

Переваги:

- менш складна, проста у проектуванні порівняно з багатошаровим перцептроном
- швидка
- чутлива до шумів

Недоліки:

- неможливість використання для глибокого навчання, через відсутність зворотного поширення

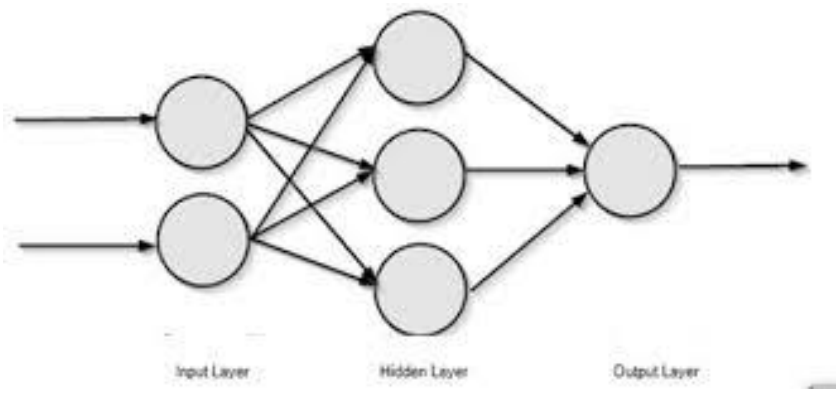

Рисунок 1.4 Нейронна мережа прямого поширення

Згорткова нейронна мережа (Convolutional Neural Network). На відміну раніше розглянутих мереж, згорткова використовує тривимірне розташування нейронів замість двовимірного розташування. Назва мережі походить від назви першого шару(згортковий). Нейрони в цьому шарі беруть та обробляють інформацію з невеликої частини вхідних даних. Принцип обробки полягає в перетворенні вхідного зображення з RGB в шкалу сірого. Подальші зміни у значеннях пікселів допоможуть виявити крайні точки і тим самим класифікувати вхідні дані за різними категоріями. Перед тим як передати результат далі, шар виконує згорнуту функцію і завдяки цьому можна проектувати більш глибоку мережу при меншій кількості вхідних параметрів чим в аналогах.

Поширення в мережі може бути як і однонаправленим так і двонаправленим. В однонаправленому варіанті мережа має один або декілька згорткових шарів, які в подальшому об’єднаються, а в двонаправлений – результат згорткового шару надходить напряму до нейронної мережі.

Так як CNN показує ефективні результати в обробці зображень та відео, то це стало основною областю використання даної мережі. Також її використовують для визначення парафраз.

Переваги:

- можливість проектувати глибоку мережу з малою кількості вхідних параметрів

Недоліки:

- може бути доволі повільною (залежить від кількості прихованих шарів)
- складна в проектуванні

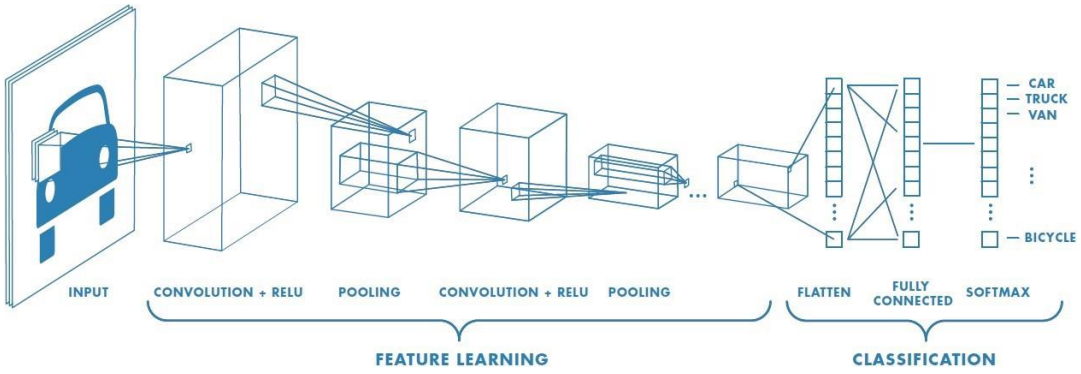

Рисунок 1.5 Згорткова нейронна мережа

Нейронна мережа радіальних базисних функцій (Radial Basis Function Neural Network) має принципово іншу архітектуру, чим інші нейронні мережі. В той час як більшість архітектур мереж складаються з багатьох шарів, то дана мережа складається всього з одного вхідного, одного прихованого та вихідного шарів. Роль вхідного шару доволі проста – прийняти вхідні дані та передати на прихований шар.

Всі обчислення відбуваються в прихованому шарі. Принцип обробки інформації відрізняється від того що ми розглядали раніше. Обчислення базується на векторах-прототипах, які є вектором з вхідних даних. Кожен нейрон в шарі має свій вектор-прототип та задану смугу пропускання і його основна

задача це знаходження подібності між вхідним вектором та вектором-прототипом.

Рекурентна нейронна мережа (Recurrent Neural Network). Основна ідея RNN полягає в послідовному використанні інформації. В більшості нейронних мереж входи та виходи є незалежні, що не є правильних для деяких задач, наприклад передбачення наступного слова в реченні. Відмінність даної мережі від інших – при виконанні однієї задачі для кожного окремого елемента враховується результат з попередніх обчислень. Рекурентні нейронні мережі часто використовуються в задачах обробки мови (NLP). На наступному рисунку зображено принцип роботи рекурентної мережі.

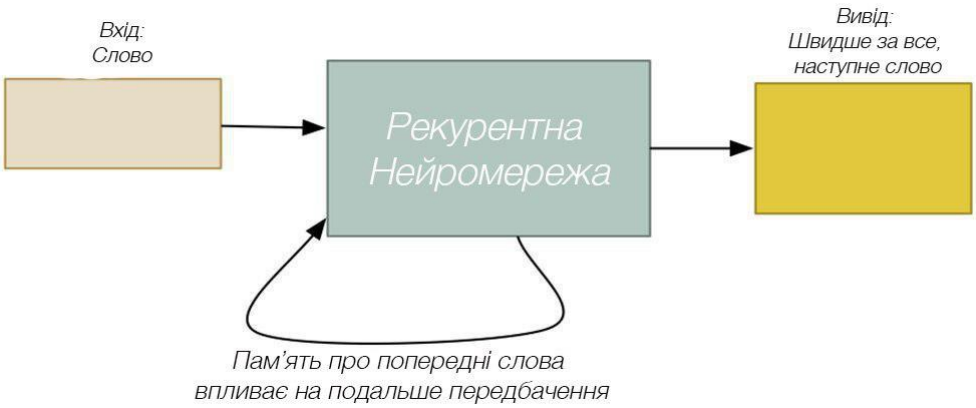

Рисунок 1.6 Рекурентна нейронна мережа

Модульна нейрона мережа (Modular Neural Network) складається з декількох моделей нейронних мереж, які пов'язані між собою за допомогою проміжного зв'язку. Насправді вони ніяк між собою не функціонують і виконують свою частину роботи. В мережі присутній інтегратора, який відповідає за поділ проблему на декілька модулів, а також за інтеграцію відповідей модулів для створення кінцевого результату.

Так як вирішенням різних проблем займаються різні модулі – це допомагає збільшити швидкість обчислення і тим самим зменшити час, потрібний для отримання результату. На рисунку 1.7 зображено модульну нейронну мережу.

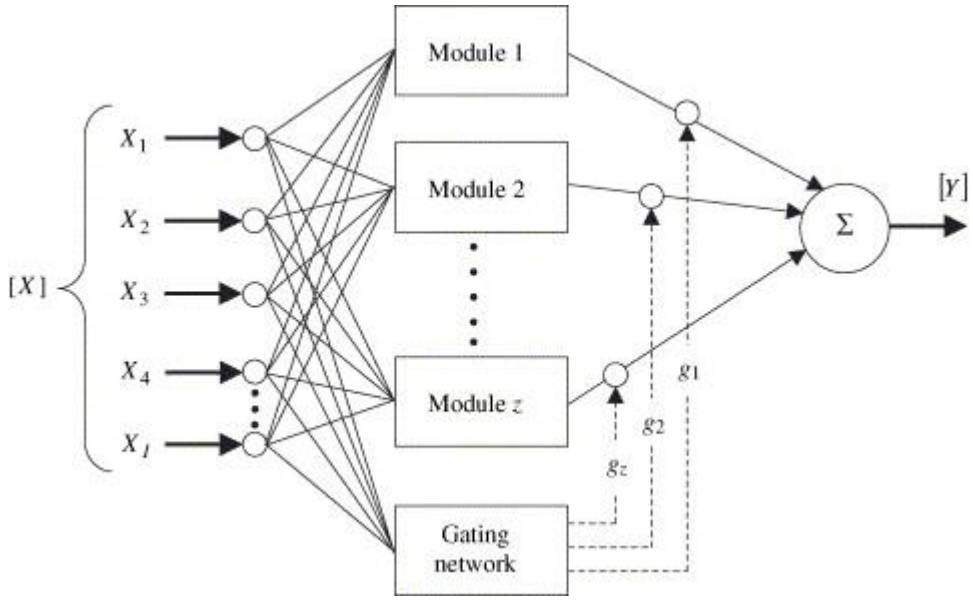

Рисунок 1.7 Модульна нейронна мережа

1.3 Архітектури на основі згорткових нейронних мереж

CNN — це тип алгоритму глибокого навчання, який використовується для обробки даних з топологією подібною сітці. CNN подібний до інших нейронних мереж, але вони мають один додатковий рівень через те, що використовують серію згорткових шарів. Ці шари є важливим компонентом мережі. На рисунку нижче зображено типову архітектуру згорткової нейронної мережі

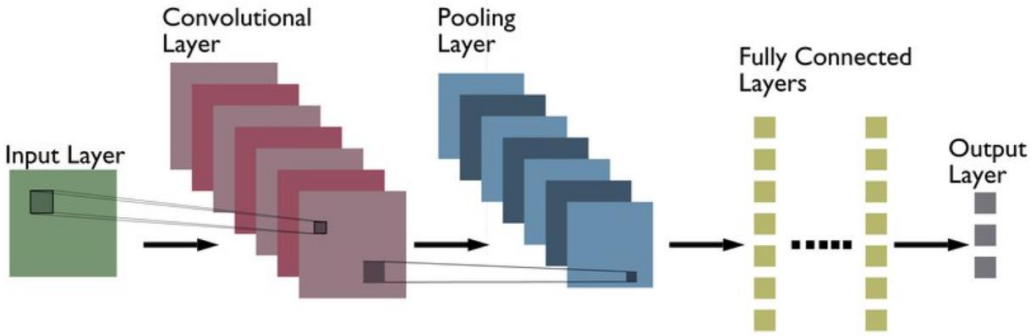

Рисунок 1.8 Типова архітектура CNN

Дивлячись на картинку, можна виділити наступні шари:

Згортковий шар (Convolutional layer) – складається з набору фільтрів, також відомих як ядрами, які застосовуються до вхідного зображення. Шари можна поєднувати між собою, що дозволяє створювати більш складні моделі, які зможуть детальніше обробляти вхідні дані. Результатом роботи цього шару є карта об‘єктів, яка є представленням вхідного зображення із застосованими фільтрами.

Рівень об‘єднання (Pooling layer) – тип згорткового шару, який використовується в навчанні. Він допомагає зменшити просторовий розмір вхідних даних, що дозволяє полегшити обробку та зменшити використання ресурсів. Рівень об‘єднання також допомагає зменшити кількість параметрів та прискорити навчання моделі. Можна виділити 2 типи цього шару: максимальний шар об‘єднання та усереднений шар. Відмінність полягає в тому, що максимальний приймає максимальне значення з кожної карти об‘єктів, в той час як усереднений – середнє. Даний шар розташовується після згорткового шару для того, щоб зменшити розмір вхідних даних, перш ніж вони подаються в повністю підключений шар.

Повністю підключений шар (Fully connected layer) – один з основних типів шарів у згортковій нейронній мережі. Як випливає з назви, кожен нейрон цього

шару повністю зв‘язаний з кожним іншим нейроном у попередньому шарі. Даний шар зазвичай використовується в кінці CNN, коли мета полягає в тому, щоб взяти особливості, засвоєні попередніми рівнями, і використовувати їх для прогнозування. Наприклад, якби ми використовували CNN для класифікації зображень тварин, остаточний шар «Повністю підключений» міг би взяти ознаки, засвоєні попередніми шарами, і використати їх, щоб класифікувати зображення як таке, що містить собаку, кішку, птаха тощо.

Далі будуть розглянуті основні архітектури на основі згорткових нейронних мереж.

LeNet-5 – одна з найпростіших архітектур. Число «5» не просто так присутнє в назві, архітектура має 2 згорткових та 3 повністю зв‘язаних шари. Присутній усереднений рівень об‘єднання, але в той час він носив назву шар вибірки і мав ваги, які можна навчати. Дана архітектура містить близько 60 тис. параметрів. Ця архітектура є одною з найстаріших і вона стала шаблоном для архітектур, які створювалися пізніше.

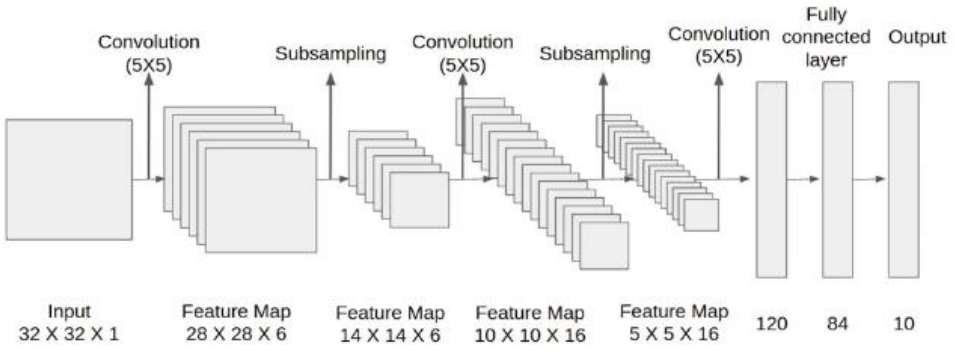

Рисунок 1.9 Архітектура LeNet-5

AlexNet має 8 шарів – 5 згорткових і 3 повністю підключених. По суті це LeNet-5, але з більшою кількістю шарів. Під час публікації даної архітектури (2012 рік), автори зазначили, що це «одна з найбільших згорткових нейронних мереж на сьогоднішній день у підмножинах ImageNet». Це була перша

архітектура, в якій реалізовано Rectified Linear Units (ReLUs) як функції активації.

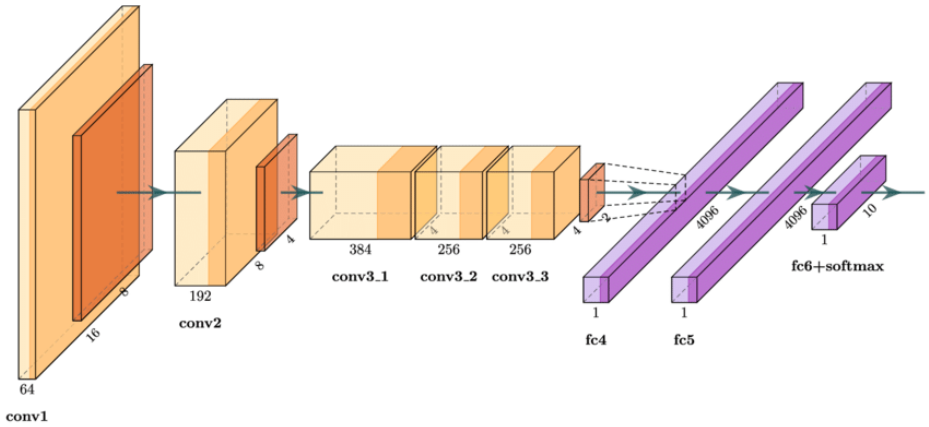

Рисунок 1.10 Архітектура AlexNet

Можна помітити, що найпростіший спосіб покращити продуктивність нейронних мереж – це збільшити їх розмір. Так виникла *VGG-16*. Складається з 13 згорткових та 3 повністю з‘єднаних шарів та використовує ReLU, яка була реалізована в попередньо розглянутій архітектурі. Ця мережа містить більше шарів ніж AlexNet і використовує фільтри менших розмірів (2×2 та 3×3). Вона складається з 138M параметрів та займає близько 500МБ пам‘яті. Також існує більш глибокий варіант даної архітектури - VGG-19.

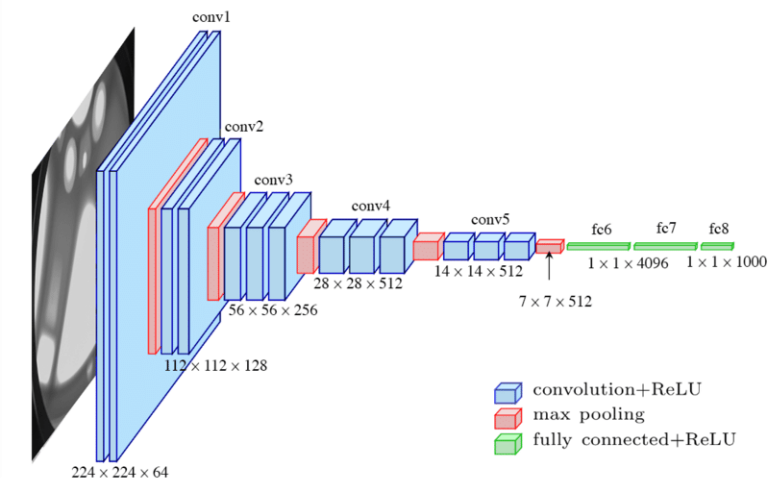

Рисунок 1.11 Архітектура VGG-16

22-рівнева архітектура з 5M параметрами носить назву *Inception-v1*. Тут широко використовується підхід «Мережа в мережі» Дизайн архітектури модуля Inception є продуктом досліджень наближення розріджених структур. Кожен модуль представляє 3 ідеї:

- Наявність паралельних башт згорток з різними фільтрами з наступною конкатенацією фіксує різні об’єкти в розмірах 1×1, 3×3 та 5×5, тим самим «групуючи» їх
- Згортки 1×1 використовуються для зменшення розмірності, щоб усунути вузькі місця обчислень
- Завдяки функції активації від згортки 1×1 її додавання також додає нелінійності

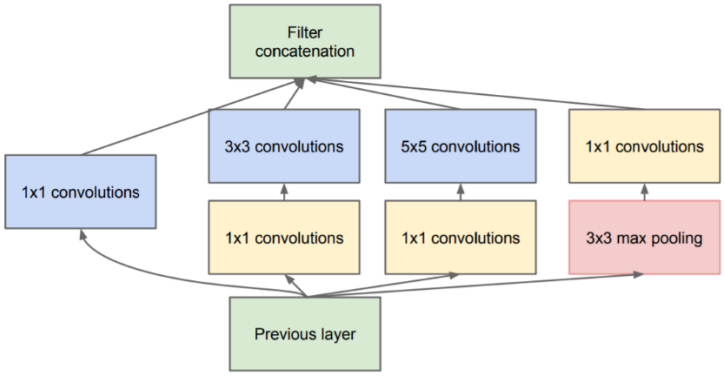

Рисунок 1.12 Архітектура модуля Inception

1.4 Класифікація зображень

В задачах object detection виділяють наступні типи задач:

Класифікація об’єкта – передбачення класу, до якого належить об’єкт на заданому зображенні. Множина класів є фіксованої і залежить від конкретної задачі. Наприклад, якщо створити класифікатор кішки-собаки, то в рамках даної задачі з великою вірогідністю можна визначити чи знаходиться на зображенні кішка або собака. Класифікація об’єкта є базовою задачею в машинному навчанні і призвана відповісти на запитання “Що за об’єкт знаходиться на зображенні?”.

Передбачення розташування об’єкта на вхідному зображенні називається локалізація об’єкта. Перш за все, дана задача відповідає на запитання “Де знаходиться об’єкт на зображенні?”. Зазвичай відповіддю на дане запитання будуть 4 точки, які визначають прямокутну область(далі bounding box), в якій знаходить об'єкт.

Object detection (виявлення об’єктів) – це задача локалізації всіх об’єктів з визначеної заздалегідь множини класів. Зазвичай в рамках цієї задачі виконується і задача класифікації об’єкта, так як множина класів є передвизначеною.

Instance segmentation (сегментація об’єктів) – схожа на задачу по виявленню об’єктів, але має 1 велику різницю. Замість знаходження bounding box, вона знаходить контур об’єкта, тим самим точніше визначає область в якій знаходиться об’єкт

На рисунку 1.13 наведена демонстрація вище перерахованих задач.

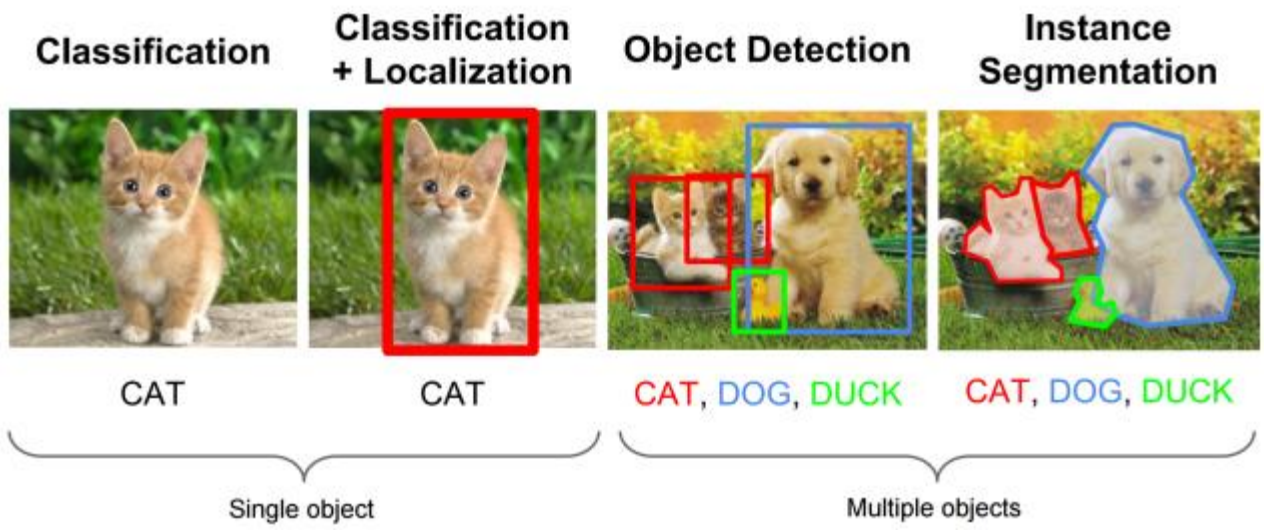

Рисунок 1.13 Демонстрація classification, localization, object detection, instance segmentation задач.

1.5 Висновки до розділу

В даному розділі розглянуто різні види нейронних мереж в залежності від типу нейронів, принципу навчання, типу вхідних даних та характеру сигналів. Виділено такі основні типи нейронних мереж:

- Нейронна мережа прямого поширення
- Згорткова нейронна мережа
- Мережа радіальних базисних функцій
- Рекурентна нейронна мережа
- Модульна нейрона мережа

На основі згорткової нейронної мережі розглянуто різні архітектури. В ході аналізу виявлено, що найпростіший спосіб створити складнішу нейрону мережу є збільшення кількості шарів. Але цей спосіб не з найкращим, так як таким чином ми збільшує розмір самої мережі, що може призвести до збільшення вузьких місць в ній.

РОЗДІЛ 2
ВИБІР АРХІТЕКТУРИ ТА ЗАСОБІВ РЕАЛІЗАЦІЇ WEB-ЗАСТОСУНКУ

2.1 Огляд мов програмування

Так як для реалізації даного проекту потрібна тільки клієнтська частина, то при виборі мови програмуванні враховувалося наявність та різноманіття інструментів для розробки клієнтської частини у відкритому доступі. Такими інструментами є open-source бібліотеки та фреймворки. На цьому етапі розглянуто наступні мови програмування – JavaScript, Python, PHP, Golang.

JavaScript – одна з найпопулярніших мов програмування в світі. Одна з причин його популярності – можливість використання для розробки клієнтської частини веб-застосунку, так і для серверної частини. Зазвичай використовується для створення інтерактивності веб-сторінок. Окрім веб-розробки може використовуватися для створення мобільних додатків та ігор.

Переваги:

- Простий синтаксис, легкий для вивчення
- Достатньо швидкий
- Інтеграція з іншими мовами програмування
- Велика різноманітність бібліотек та фреймворків

Недоліки:

- Немає підтримки множинного успадкування (multiple inheritance)
- Немає статичної типізації. Є можливість додати типізації за допомогою бібліотек
- Менш безпечний порівняно з іншими мовами

Рисунок 2.1 Логотип JavaScript

Python – мова з відкритим кодом, яка легка у вивченні, є однією з популярних мов, які використовуються для веб-розробки. Також на python автоматизують різні процеси та обробляють дані (data science).

Переваги:

- Легко масштабувати
- Простий у вивченні
- Багато open-source бібліотек
- Легко інтегрується

Недоліки:

- Інтерпретована мова програмування
- Неможливість створення мобільних додатків

Рисунок 2.2 Логотип Python

PHP – скриптова мова програмування, яка вже два десятиліття використовується у веб-розробці. PHP – це серверна мова, в яку вбудована підтримка HTML. Він використовується для керування динамічним змістом, базами даних та відстежуванням сеансів.

Переваги:

- Універсальна мова. PHP легко комбінується з іншими мовами, такі як C, які є швидше
- Багато фреймворків та бібліотек
- Не потребує великого бюджету для хостингу

Недоліки:

- Зниження популярності
- Проблеми з продуктивністю

Рисунок 2.3 Логотип PHP

Golang(Go) – це багатопарадигмальна, статично типізована та компільована мова програмування, розроблена Google. Go базується на класичному синтаксисі C/C++ з додатковими перевагами збирання сміття, безпеки пам’яті, автоматичного оголошення змінних, структурної типізації та

паралельності в стилі CSP. Це також остання мова програмування, яка зберігає багатопотоковість у своїй основі.

Переваги:

- Простий та швидко компілюється
- Програми можна редагувати та запускати безпосередньо в Інтернеті
- Підтримується компанією Google

Недоліки:

- Відносно нова мова з невеликою кількістю бібліотек та інформації
- Має дефекти в управлінні залежностями
- Мова високо рівня з низькорівневими функціями

Рисунок 2.4 Логотип Golang

2.2 Огляд фронтенд технологій

ReactJS - це широко використовувана бібліотека JavaScript з відкритим кодом. Він допомагає створювати веб-застосунки, які вимагають мінімальних зусиль та кодування. Основною метою ReactJS є розробка інтерфейсу користувача (UI), який покращує швидкість додатків. За допомогою нього можна створювати веб-застосунки, де вся взаємодія з користувачем відбувається на

стороні клієнта – так звані Client-Side applications. В сукупності з бібліотекою-менеджером стану Redux або Mobx можна реалізувати шаблон MVC, який в майбутньому буде просто масштабувати.

Переваги:

- Використання Virtual DOM. Virtual DOM набагато ефективніший, працює швидше і, зрештою, створює кращий інтерфейс користувача
- ReactJS ізолює свої компоненти. Це означає, що зміни одного екземпляра компонента не вплинуть на всі інші, що дозволяє розробникам ефективно повторно використовувати компоненти дизайну
- Односторонній потік даних. В такій структурі зміна дочірніх елементів не впливає на батьківські дані. Це створює більш стабільний код
- SEO Friendly. ReactJS пом’якшує проблему надмірно важких веб-сайтів JavaScript і дає можливість прочитати сайт Google та іншими пошуковими системами

Недоліки:

- Темпи розвитку. Реакт постійно розвивається та змінюється
- Зворотна сумісність. Зазвичай різні версії не мають зворотної сумісності, тому при переході на нову версію потрібно тестувати весь веб-застосунок
- Відсутність детальної документації. Ця проблема викликана з тим що бібліотека швидко розвивається і інколи доводитися залазити в вихідний код, щоб зрозуміти як працює та чи інша функція

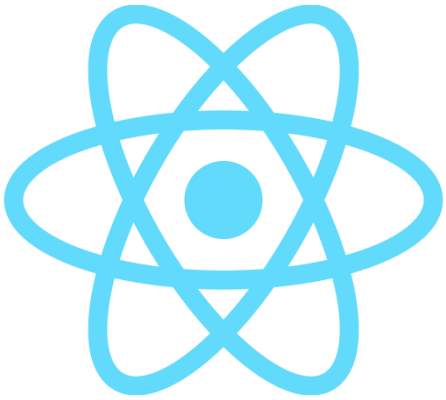

Рисунок 2.5 Логотип ReactJS

Vue.js - прогресивний фреймворк для створення односторінкових програм та веб-інтерфейсів на JavaScript. Він також використовується для розробки настільних та мобільних додатків за допомогою фреймворку Electron. Назва фреймворка – Vue – англійською збігається з фонетичною як view, і вона відповідає традиційній архітектурі Model-View-Controller (MVC). Але це не означає що його не можно використовувати з іншими архітектурними підходами, наприклад – Component Based Architecture (CBA).

Переваги:

- Малий розмір. Розмір стисненого архіву всього 18КБ. Це не тільки дозволяє швидко завантажувати застосунок, а й позитивно впливає на SEO
- Простота. Синтаксис фреймворку простий і легкий для розуміння. Його структура інтуїтивно зрозуміла, а компоненти, по суті, є сукупністю HTML і JavaScript
- Документація. Vue.js добре задокументований – відео-курси з документацією полегшують вам розуміння концепцій та взаємодію з ними

- Реактивність. Його інтеграція з HTML і JavaScript робить зв’язування даних між ними дуже простою. Також він здатний обробляти двосторонній потік даних

Недоліки:

- Мовні бар’єри та обмежена спільнота. Vue.js відносно новий фреймворк, який має китайське походження, тому деяку інформацію не можливо знайти на англійській мові
- Відсутність фінансової підтримки масштабних проектів. Через те що фреймворк молодий, він не отримує великої фінансової підтримки від підприємств, через це команда розробників не може швидко вирішувати проблеми, що виникають
- Обмежені ресурси. Хоча екосистема досить широка, і є всі необхідні інструменти для початку розробки за допомогою Vue, вона все ще не така велика, як React або Angular

Рисунок 2.6 Логотип Vue.js

Svelte – це component-based фреймворк. Можна сказати що він схожий з ReactJS та Vue.js, але є одна важлива відмінність: для них потрібен

декларативний код, який допомагає браузеру конвертувати стан застосунку в операції DOM, а для Svelte – цей прошарок не потрібен. Це не класичний фреймворк, а компілятор, який перетворює код у високоефективний імперативний код і безпосередньо маніпулює DOM.

Переваги:

- Мінімальна крива навчання. Оскільки ви пишете компоненти за допомогою звичайних HTML, CSS і Javascript, ви можете почати писати програми Svelte приблизно за 5 хвилин
- Швидкість виконання. Як згадувалося раніше, Svelte є компілятором, і тому під час процесу збірки ваші компоненти Svelte перетворюються на звичайний код JavaScript. Це допомагає уникнути зайвих витрат на завантаження або завантаження фреймворка перед запуском коду в браузері
- Розробка додатків на основі компонентів
- Багата екосистема. Керування станом, шаблони, рендеринг на стороні сервера, система плагінів та анімація — це деякі з багатьох інструментів, які поставляються зі Svelte прямо з коробки

Недоліки:

- Без серйозної підтримки. У Svelte зараз немає великої компанії, тому вона все ще не користується популярністю серед компаній і розробників
- Невелика спільнота. Оскільки Svelte є досить новим, у нього ще немає такої великої спільноти та шанувальників розробників, як інші фреймворки
- Підтримка інструментів і пакетів. Що стосується інструментів і пакетів для розробників, наразі розробникам Svelte доступні обмежені можливості на вибір. Але в міру того, як спільнота розростається і все

більше розробників починає використовувати цей фреймворк, ця проблема зникне

- Підтримка IDE. На сьогоднішній день існує декілька IDE, які частково підтримують синтаксис Svelte. Ця проблема ускладнює написання код і тим самим збільшую час, який потрібний для розробки застосунку на даному фреймворку

Рисунок 2.7 Логотип Svelte

Django – повнофункціональний фреймворк для веб-розробників, які працюють з Python. На офіційному сайті його називають «веб-фреймворк для перфекціоністів (з дедлайнами)». Його основна ціль, як і в інших фреймворках – скоротити загальний процес розробки при створенні веб-застосунків. Основні інструменти які пропонує Django: кешування, логування, аутентифікація, управління статичними сторінками то розбивання на сторінки. Фреймворк слідує принципу DRY. Ця концепція, скорочено від Don’t Repeat Yourself, призначена для того, щоб уникнути повторення шаблонного коду та допомагає розробникам зберегти велику кількість часу і дозволяє їм зосередитися на розробці самого застосунку.

Переваги:

- Швидкість розробки та масштабування. Django дозволяє швидко створювати та масштабувати проекти. І це одна з причин чому він популярний серед стартап проектів
- Підтримка спільноти. Оскільки фреймворк існує понад 15 років, є велика кількість документації та курсів
- Універсальність. За допомогою Django можна створити соціальну мережу так і систему управління контенту(CMS). Також його використовують для машинного навчання і аналізу даних

Недоліки:

- Не підходить для простих проектів. З самого початку, Django був створений для виконання непростих задач. Якщо вам потрібно створити невеликий та простий проект, то краще використати мікро-фреймворк Flask
- Повільність роботи. На Django можна швидко розробляти застосунки, але зі збільшенням продукту зменшується продуктивність
- Відсутність конвенцій. Хоч фреймворк існує доволі давно – досі не існує конвенції кодування

Рисунок 2.8 Логотип Django

Laravel – це PHP-фреймворк, побудований на концепції MVC. Laravel – модульний, інтуїтивно зрозумілий та простий в використанні. Він має багато інтеграцій багатофункціонального програмного забезпечення. Платформа наповнена зручними для розробників функціями, що дозволяє зосередитися на розробці застосунку, а не писати все з нуля.

Переваги:

- Вбудована функція контролю доступу. Laravel має метод аутентифікації, який поєднує всі необхідні кроки, такі як вхід, реєстрація та скидання пароля
- Спрощена система інтеграції електронної пошти. За допомогою фреймворку можна використовувати драйвера для Mandrill, Amazon SES, Sendmail та інших. Листи можна надсилати як через локальні сервіси, так і через хмару
- Відокремлення бізнес-логіки від логіки представлення. Laravel дотримується архітектури MVC, яка відповідає за поділ (біфуркацію) між бізнес-логікою та логікою представлення, що дозволяє легко змінювати інтерфейс користувача не змінюючи бізнес-логіку
- Спрощена конфігурація маршрутизації (зворотна маршрутизація). В структурі можна створювати іменовані маршрути. Це дозволяє спростити менеджмент існуючих сторінок

Недоліки:

- Оскільки це невеликий фреймворк, то він має обмежену вбудовану підтримку. Однак цю проблему вирішують підключення різних бібліотек

- Погана зворотна підтримка. При переході на нову версію Laravel може з‘явитися проблема, що деякі вбудовані функції не працюють, або працюють неналежним чином
- Непростий в засвоєнні. Багато вбудованих функцій є комплексними

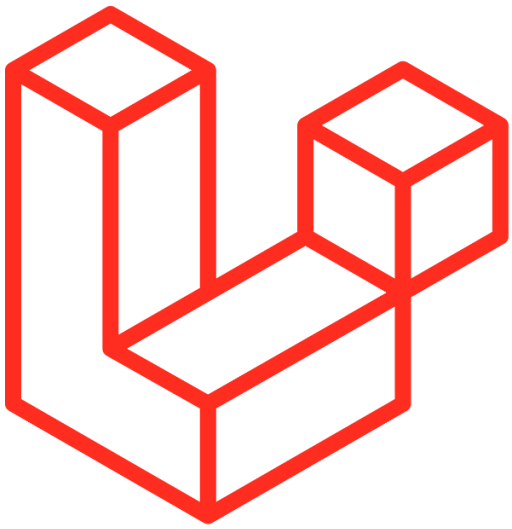

Рисунок 2.9 Логотип Laravel

Echo – це відносно новий веб-фреймворк Go. Echo дозволяє створювати API з нуля завдяки чітко сформульованій документації, якій дуже легко слідувати. Його функції включають можливість розробникам визначати власне проміжне програмне забезпечення (middleware) та підтримку HTTP/2

Особливості:

- Оптимізований маршрутизатор HTTP, який визначає пріоритети маршрутів
- Кастомізація
- Централізована обробка помилок HTTP
- Зручні функції для надсилання різноманітних відповідей HTTP

Рисунок 2.10 Логотип Echo

2.3 Огляд засобів реалізації аналізу обличчя

TensorFlow – це бібліотека машинного навчання з відкритим вихідним кодом, розроблена Google. Це математична бібліотека, яка використовує потік даних та диференційоване програмування для задач, зосереджених на навчанні нейронних мереж. Вона дозволяє створювати програти машинного навчання, використовуючи різні інструменти та бібліотеки з відкритим вихідним кодом.

Переваги:

- Графічне представлення. Бібліотека має хороші засоби для візуалізації даних. Це полегшує роботу з нейронними мережами
- Паралелізм. TensorFlow для функціювання системи використовує ресурси CPU та GPU. Розробник може самостійне визначити які ресурси та в якій кількості бібліотека буде використовувати
- Сумісність. Він сумісний з багатьма мовами програмування, такими як Python, C++, JavaScript тощо. Це дозволяє користувачам працювати в середовищі, в якому їм зручно
- Архітектурні переваги. Архітектура TensorFlow може використовувати TPU, що робить обчислення швидше, ніж CPU і GPU. Моделі, створені

на основі TPU, можна легко розгорнути в хмарах і працювати швидше в порівнянні з двома іншими

Недоліки:

- Швидкодія. Порівняно з конкуруючими фреймворками він повільний та менш зручний
- Підтримка графічного процесора. Tensorflow має лише підтримку NVIDIA для GPU
- Залежності. Хоч і TensorFlow зменшує розмір програми, все рівно залишається шаблонний код, який потрібен для роботи платформи
- Часті оновлення. Через те що продукт швидко розвивається, виходять оновлення, що може призвести до того що деякий функціонал перестає працювати

Рисунок 2.11 Логотип TensorFlow

PyTorch – це бібліотека Python, яка здійснює негайне виконання динамічних тензорних обчислень з автоматичним диференціюванням та прискоренням графічного процесора, зберігаючи продуктивність. На сьогоднішній день більшість функціонала ядра написано на C++, що дозволяє

досягти того ж самого результату при менших затратах. Він підтримує імперативний та Pythonic стиль програмування.

Переваги:

- PyTorch заснований на Python. Тобто це означає що бібліотека не є інтерфейсом до бібліотеки, написаної на іншій мові
- Налагодження (debug). Бібліотека підтримує режим налагодження за допомогою одного з багатьох інструментів налагодження Python
- Підтримка динамічних обчислюваних графіків. PyTorch підтримує динамічні обчислювальні графіки, що означає, що поведінку мережі можна змінити програмно під час виконання. Дана підтримка значно полегшує оптимізацію моделі і цим самим бібліотека має перевагу перед фреймворками машинного навчання, які розглядають нейронні мережі як статичні об’єкти.
- Спільнота. Бібліотека має активну спільноту та детальну документацію, тому бібліотека проста в освоєнні для початківців. Сама крива навчання також коротка

Недоліки:

- Обмеженні інтерфейси моніторингу та візуалізації. В PyTorch немає аналога TensorBoard в TensorFlow, таким чином розробниками потрібно підключати додаткові бібліотеки
- Не є наскрізним інструментом. Розробка реальних додатків вимагає перетворення коду PyTorch в іншу платформу для того щоб розгорнути серверах, робочих станціях і інших пристроях

Рисунок 2.12 Логотип PyTorch

OpenCV – це безкоштовна бібліотека з відкритим вихідним кодом, яку використовують для машинного навчання та комп‘ютерного зору, в якій основний інтерфейс написаний на мові C++ і має інтеграцію в такі мови як x Python, Java, JavaScript та інші. Ця платформа пропонує загальну інфраструктуру для швидкого використання машинного сприйняття. OpenCV пропонує доступ до понад 2500 алгоритмів, які можна використати для різноманітних цілей машинного навчання та комп‘ютерного зору, такі як ідентифікація об‘єктів та розпізнавання обличчя.

Переваги:

- Підтримка компаній. Великі компанії, такі як IBM, Google, Toyota і стартапи, такі як Zeitera і Applied Minds, використовують OpenCV для різноманітних завдань.
- Велика кількість різноманітних алгоритмів. Бібліотека надає доступ до більш ніж 2500 найсучасніших та класичних алгоритмів. Використовуючи цю бібліотеку, користувачі можуть виконувати різні завдання, такі як видалення червоних очей, вилучення 3D-моделей об’єктів, стеження за рухами очей, тощо
- Інтеграція з популярними мовами програмування

Недоліки:

- Високий поріг входження. Бібліотека не є простою в освоєнні, тому зазвичай її використовують більш досвідчені розробники
- Швидкодія. Не зважаю чи на те що основний інтерфейс написаний на мові C++, OpenCV має малу швидкодію в браузері

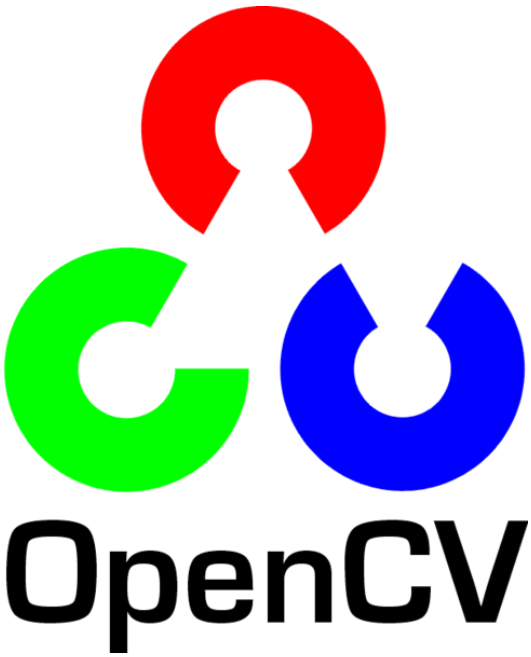

Рисунок 2.13 Логотип OpenCV

Keras – це високорівнева API нейронної мережі, написана мовою Python. Вона є надбудовою над фреймворками TensorFlow та Theano. Це одна з бібліотека, що використовується для навчання нейронних мереж, яка дотримується мінімалістичної філософії. Keras був створений для можливості проведення швидких дослідів та експериментів з нейронними мережами. Його архітектура розбита на малі модулі та підтримує можливість масштабування. Цю бібліотеку використовують як і IT-гіганти, так і малі стартапи

Переваги:

- Порівняно проста та зручна у використанні, так як більшість інструментів вже наявна і не потрібно встановлювати додаткові пакети
- Підтримка бекенда. Так як Keras є надбудовою над TensorFlow та Theano, він не працює з низькорівневими обчисленнями
- Попередньо навчені моделі. В бібліотеці існує достатня кількість попередньо навчених моделей, які можна використати для знаходження та ідентифікації об‘єктів
- Велика спільнота та детальна документація. Продукт має велику підтримку в суспільстві, що дозволяє швидко віднайти відповідь на своє питання

Недоліки:

- Keras не підтримує функцію створення динамічних діаграм
- Неефективні помилки. Бібліотека має бідний функціонал по виявленню помилок, що ускладнює процес налагодження
- Швидкодія. Має доволі низьку швидкодію, якщо використовувати як основний компонент великих проектів. Також те що бібліотека написана на мові Python впливає на це

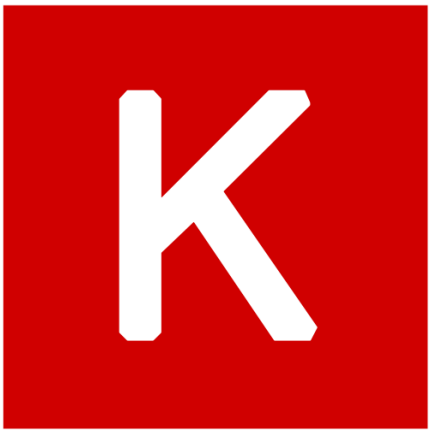

Рисунок 2.14 Логотип Keras

Face-api.js – це JavaSript бібліотека, яке застосовується для виявлення та розпізнавання обличь як в браузері, так і на серверній частині. Ця бібліотека реалізована поверх основного API tensorflow.js. Принцип його роботи базується на основі згорткових нейронних мережах (CNN), які оптимізовані для роботи в браузері та мобільних пристроях. Для виявлення обличчя face-api.js реалізує моделі SSD Mobilenet V1, Tiny Face Detector і експериментальний MTCNN.

Переваги:

- Оптимізація. Бібліотека оптимізована для роботи в браузері, що дає перевагу над іншими продуктами
- Наявність API. Це дозволяє зручно взаємодіяти з даною бібліотекою
- Відкритий вихідний код

Недоліки:

- Мала спільнота. Так як бібліотека пропонує вузький спектр можливостей, вона не має багато прихильників
- В документації описано тільки принцип роботи основних функцій, тому якщо потрібно розібратися в чомусь конкретно доведеться залазити в вихідний код

```
{
  "detection": {
    "_imageDims": {
      "_width": 1280,
      "_height": 720
    },
    "_score": 0.6889822483062744,
    "_classScore": 0.6889822483062744,
    "_className": "",
    "_box": {
      "_x": 121.50997161865234,
      "_y": 15.035667419433594,
      "_width": 507.80059814453125,
      "_height": 531.7609024047852
    }
  },
  "gender": "male",
  "genderProbability": 0.9683359265327454,
  "age": 30.109874725341797,
  "expressions": {
    "neutral": 0.9950351715087891,
    "happy": 0.000001711382424218754 7,
    "sad": 0.00000579671996092656 6,
    "angry": 0.00000466804613097338,
    "fearful": 1.3292748013427058e-9,
    "disgusted": 3.015825145169515e-9,
    "surprised": 0.004952521994709969
  }
}
```


Рисунок 2.15 Приклад відповіді API після обробки зображення

2.4 Висновки до розділу

В даному розділі проведено аналіз можливих мов програмування для застосунку та ряд бібліотек і фреймворків, їх переваги та недоліки, які б спростили розробку. Також розглянуто можливі бібліотеки для реалізації розпізнавання та виявляння обличчя на зображенні. Була отримана необхідна інформація щодо набору технологій які будуть використовуватися при розробці веб-застосунку для виявлення обличчя.

Мова програмування для даного застосунку – JavaScript. В додаток до JavaScript обрано бібліотеку React, для того щоб спростити та пришвидшити розробку клієнтського інтерфейсу.

Для реалізації функції аналізу обличчя на зображенні обрано бібліотеку TensorFlow з попередньо навченою моделлю Blazeface. Blazeface – це легка модель, яка розпізнає обличчя на зображеннях. Вона використовує модифіковану архітектуру Single Shot Detector зі спеціальним кодером.

РОЗДІЛ 3
РОЗРОБЛЕННЯ ВЕБ-ЗАСТОСУНКУ

3.1 Архітектура веб-застосунку

Початкова ціль – визначитися з архітектурою застосунку. Вирішено використовувати Component-Based Architecture – це архітектура, що складається з незалежних, модульних та багаторазових блоків, які називаються компонентами. Список розділів та їх призначення представлено у таблиці 1.

Таблиця 1

Структура застосунку

Назва папки	Призначення
assets	Сховище статичних файлів (картинки, іконки)
components	Набір багаторазових компонентів
config	Список наперед визначених значень
context	Спільне сховище додатку
hooks	Функції для управління станом застосунку
layouts	Макети застосунку
pages	Комбінація компонентів
utils	Спільні функції

3.2 Реалізація розпізнавання облич на зображенні

Функціональність з обробки зображень та розпізнавання на них облич була розроблена на основі фреймворку TensorFlow. За допомогою даного фреймворку можна навчити свою модель, так і використати вже попередньо навчені. Вирішено використати навчену модель Blazeface. Дана модель складається з 2 легких моделей (розміром 224 КБ і 308 КБ), які використовуються для визначення одного або декількох облич на зображенні. Перша модель використовується для обробки зображення зробленого на передню (селфі) камеру, а друга спрямована на зображення задньої камери.

Для навчання моделі використовувалася вибірка користувачів з 17 рівномірно представлених географічних субрегіонів:

1. Австралія та Нова Зеландія
2. Меланезія, Мікронезія, Полінезія
3. Європа
4. Центральна Азія
5. Східна Азія
6. Південно-Східна Азія
7. Південна Азія
8. Західна Азія
9. Карибський басейн
10. Центральна Америка
11. Південна Америка
12. Північна Америка
13. Північна Африка
14. Східна Африка
15. Центральна Африка
16. Південна Африка

17.Західна Африка

Рисунок 3.1 – Геосхема ООН

Для навчання моделі фронтальної камери використали 2 набори даних. Перший набір містив 720 зображень: 680 рівномірно розподілених по субрегіонам зображень, тобто по 40 зображень на кожен регіон та 40 зображень на яких не було людського обличчя. Другий набір даних складався з 1060 зразків: по 60 зображень кожен субрегіон та 40 зображень без облич. В першому та другому наборі використовувався однаковий набір зображень без облич. Модель задньої камери навчали на одному наборі: 1350 зображень, де 1275 – фото з обличчями, 75 – без обличчя.

Для кожного ідентифікованого обличчя модель повертає уніфікований набір даних:

- Координати обмежувальної рамки обличчя
- 6 координат ключових точок обличчя:
    - Ліве око
    - Праве око
    - Кінчик носа

- Рот
- Ліве вухо
- Праве вухо

- Оцінка впевненості виявлення

Представлених даних достатньо для реалізації основного функціоналу застосунку.

```
[
  {
    topLeft: [331.36, 29.89],
    bottomRight: [436.94, 88.39],
    landmarks: [
      [369.51, 48.69],
      [405.93, 47.37],
      [384.49, 60.28],
      [380.85, 70.74],
      [347.18, 52.12],
      [425.176, 51.80],
    ],
    probability: 0.939,
  },
];
```

Рисунок 3.2 Приклад відповіді від API моделі

Ідентифікація обличчя

Для реалізації розпізнавання облич на зображені потрібно створити міст між TensorFlow API та клієнтським інтерфейсом на ReactJS. Для цього розроблено наступний функціонал:

- Завантаження TensorFlow та моделі Blazeface
- Обробка статичного зображення

- Обробка відео з веб-камери
- Формування прямокутної рамки в межах якої знаходиться обличчя

На наступних рисунках зображено фрагменти коду, які відповідають за даний функціонал.

```
import '@tensorflow/tfjs';
import * as blazeface from '@tensorflow-models/blazeface';

const [isModelLoaded, setModelStatus] = useState(false);
const [model, setModel] = useState(undefined);

const requestModel = async () => {
  if (!isModelLoaded || !model) {
    const requiredModel = await blazeface.load();

    setModel(requiredModel);
    setModelStatus(true);
  }
};
```

Рисунок 3.3 Завантаження TensorFlow та моделі Blazeface

```
const detectFace = async () => {
  if (model && photoRef?.current) {
    const prediction = await model.estimateFaces(
      photoRef?.current,
      returnTensors,
    );

    setPredictions(prediction);
  }
};

useEffect(() => {
  if (isModelLoaded) {
    setTimeout(detectFace, 10);
  }
}, [isModelLoaded, photoRef]);
```

Рисунок 3.4 Обробка статичного зображення

```
const detectFace = async () => {
  if (model && webcamRef?.current?.video?.readyState === 4) {
    // Get video
    const { video } = webcamRef.current;

    const prediction = await model.estimateFaces(video, returnTensors);

    setPredictions(prediction);
  }
};

useEffect(() => {
  if (isModelLoaded && !interval) {
    interval = setInterval(() => {
      debounce(detectFace(), 100);
    }, 100);
  }
}, [isModelLoaded]);
```

Рисунок 3.5 Обробка відео

```
const [firstStartPoint, secondStartPoint] = topLeft;
const [firstEndPoint, secondEndPoint] = bottomRight;
const size = [
  firstEndPoint - firstStartPoint,
  secondEndPoint - secondStartPoint,
];
const probabilityInPercent = (probability * 100).toFixed(2);

ctx.beginPath();
ctx.lineWidth = '2';
ctx.lineJoin = 'round';
ctx.strokeStyle = '#FE9A22';
ctx.setLineDash([10, 10]);
ctx.font = '16px serif';

const { width: textWidth } = ctx.measureText(
  `${probabilityInPercent}%`,
);
ctx.fillStyle = 'white';
ctx.fillRect(firstEndPoint + 3, secondEndPoint, textWidth + 4, 16);
ctx.fillStyle = getProbabilityColor(probabilityInPercent);

ctx.fillText(
  `${probabilityInPercent}%`,
  firstEndPoint + 5,
  secondEndPoint + 14,
);

ctx.rect(firstStartPoint, secondStartPoint, ...size);
ctx.stroke();
```

Рисунок 3.6 Формування прямокутної рамки

3.3 Особливості застосунку

Програмний застосунок виконано у вигляді односторінкового веб-додатку (SPA), де вся взаємодія відбувається на стороні клієнта. Перевага такого способу полягає в тому що, завантаживши потрібний код, він кешується та буде використаний при наступних заходах на застосунок. Також, використано якого менша кількість сторонніх бібліотек, що дозволяє досягти невеликий розмір сайту в production режимі. При розробці застосунку дотримано рекомендації Google щодо створення юзер інтерфейсу і тому додаток приємний на вигляд та їм зручно користуватися як на комп‘ютері, так і на телефоні та на вигляд.

Можна виділити 4 основні сторінки:

- Сторінка попереднього завантаження
- Головна сторінка
- Сторінка з завантаженням зображень
- Сторінка з веб-камерою

Як зазначалося раніше, в основу архітектури закладено орієнтацію на перевикористання компонентів і тому деякі компоненти повторюються на різних сторінках

Хедер (шапка сайту) – компонент через виконується основна навігація по застосунку. За допомогою нього можна перейти на головну сторінку та сторінки, які обробляють зображення.

Нижче наведено приклади десктоп та мобайл хедера в різних станах.

Рисунок 3.7 Десктоп хедер на головній сторінці

Рисунок 3.8 Десктоп хедер на сторінці з завантаженням фото

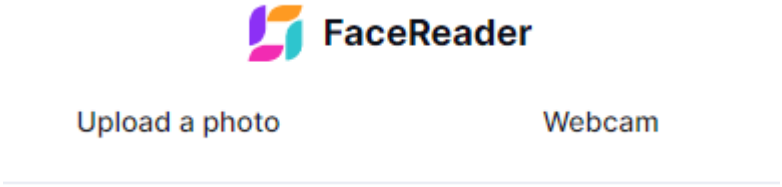

Рисунок 3.9 Мобайл хедер на головній сторінці

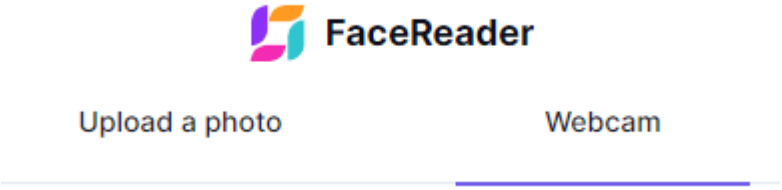

Рисунок 3.10 Мобайл хедер на сторінці веб-камери

Футер (підвал сайту) – в даному компоненті знаходяться основні контакти, корисні посилання, знак копірайту та дублюється навігація. На рисунках 3.11 та 3.12 проілюстровано вигляд в десктопному та мобільному вигляді відповідно.

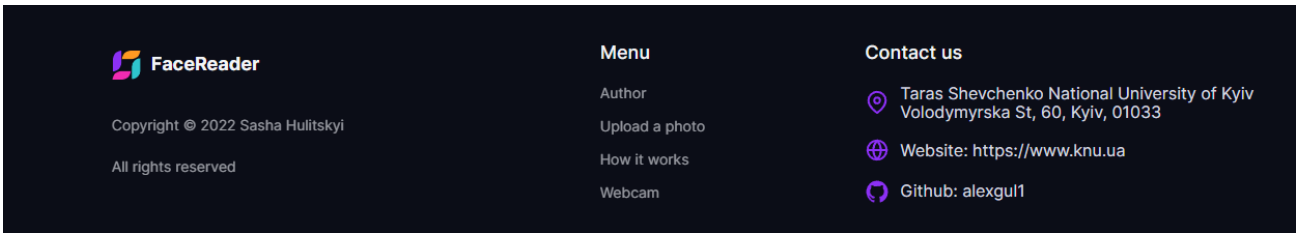

Рисунок 3.11 Футер в десктоп вигляді

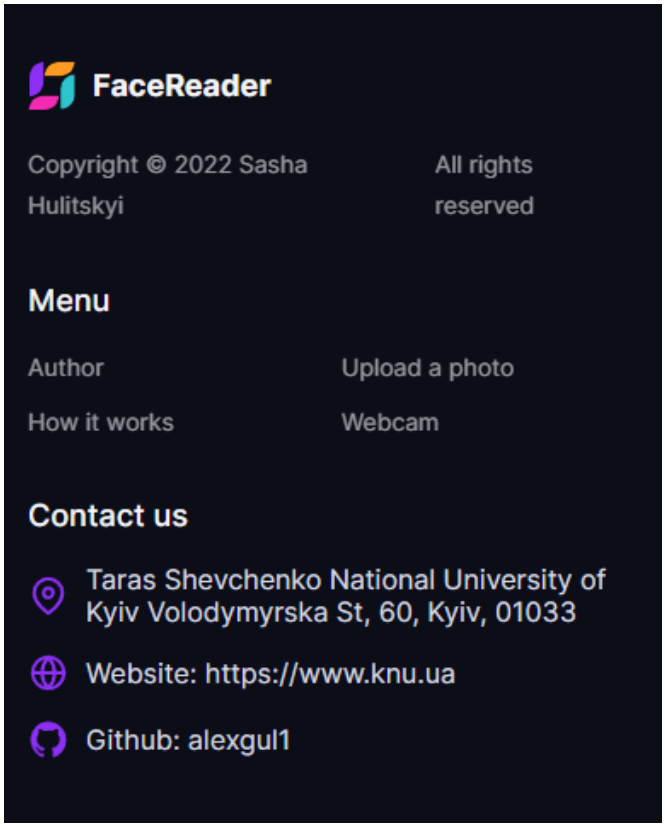

Рисунок 3.12 Футер в мобайл вигляді

Сторінка попереднього завантаження (preload page) – це сторінка, яка буде відображатися, до тих пір поки не будуть завантаженні необхідні ресурси для роботи застосунку (бібліотеки та модель). Дана сторінка допомагає користувачу зрозуміти, що додаток працює справно та зменшує процент людей, які покидають застосунок не дочекавшись завантаження. Вигляд мобільному та десктопному варіантах не відрізняється.

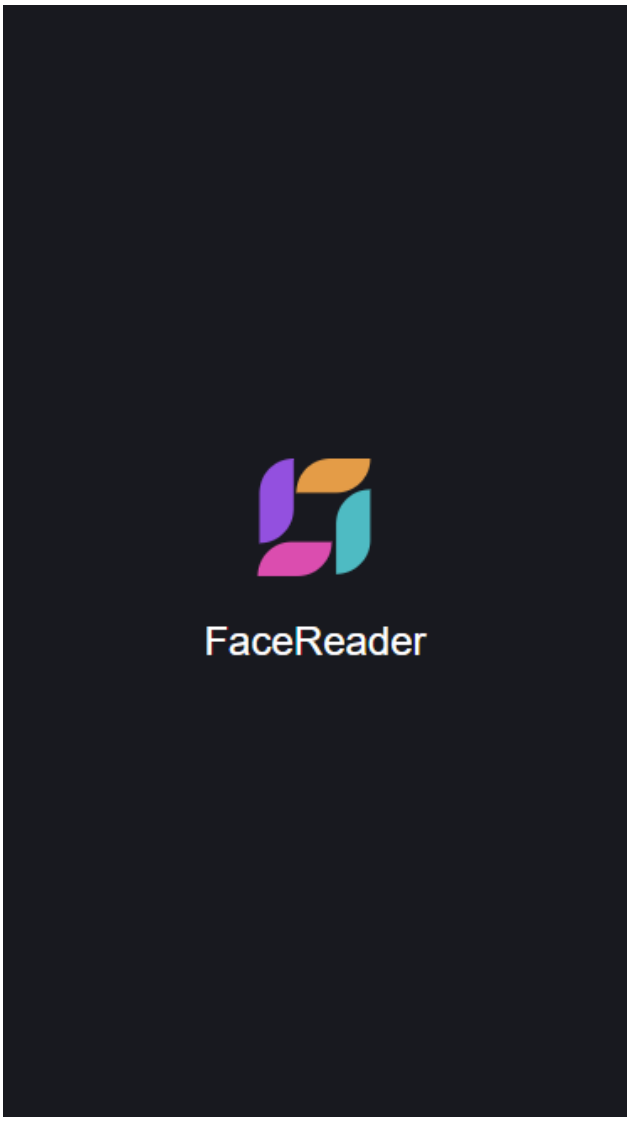

Рисунок 3.13 Сторінка попереднього завантаження

Головна сторінка – є інформаційною. Це перша сторінка, яку користувач бачить після завантаження. На ній міститься хедер, інформаційний банер, мету створення і принцип роботи застосунку. Головна мета цієї сторінки – допомогти користувачу розібратися з існуючим функціоналом та призвати до наступних дій.

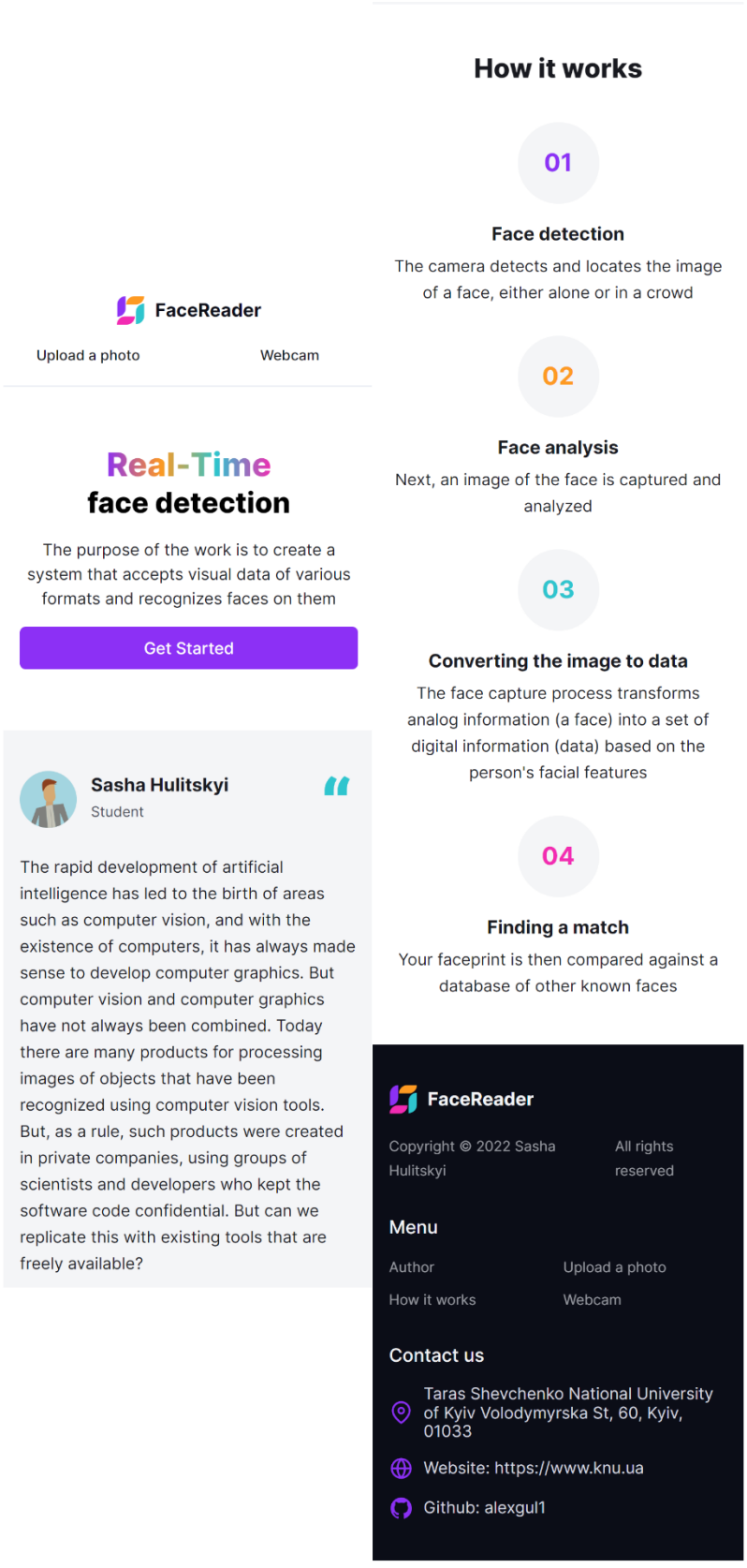

Рисунок 3.14 Головна сторінка в mobile view

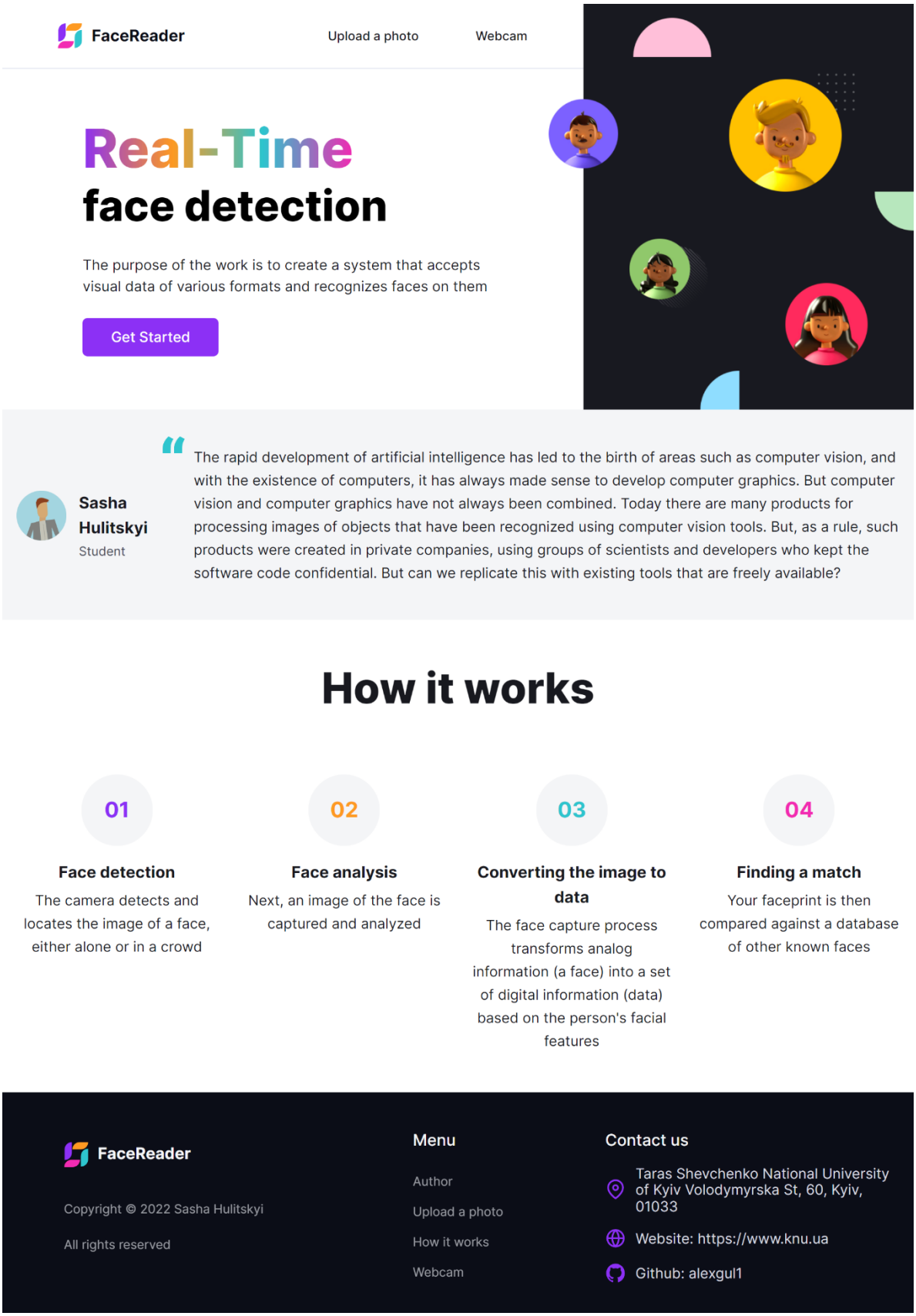

Рисунок 3.15 Головна в desktop view

Сторінка з завантаженням зображень – містить елемент керування завантаження фото за допомогою Drag-and-Drop API або звичайним способом – вибір через провідник. Є можливість завантажити один або декілька файлів одночасно, дозавантаження також реалізовано. Типи підтримуваних файлів обмежено – тільки файли з типом “image”. Після завантаження файлів, зображення будуть оброблятися почергово, один за одним.

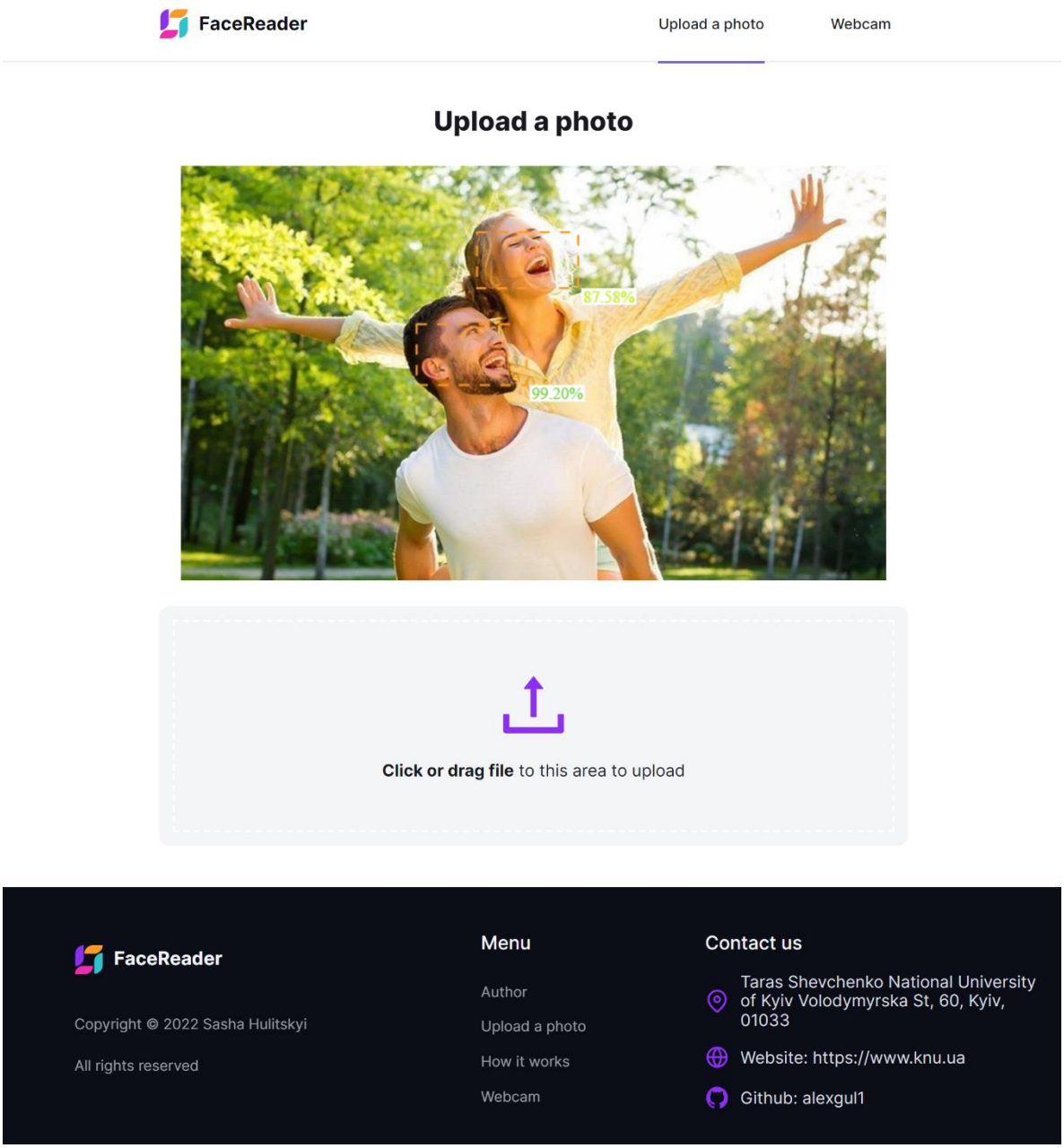

Рисунок 3.16 Приклад обробки зображення в desktop view

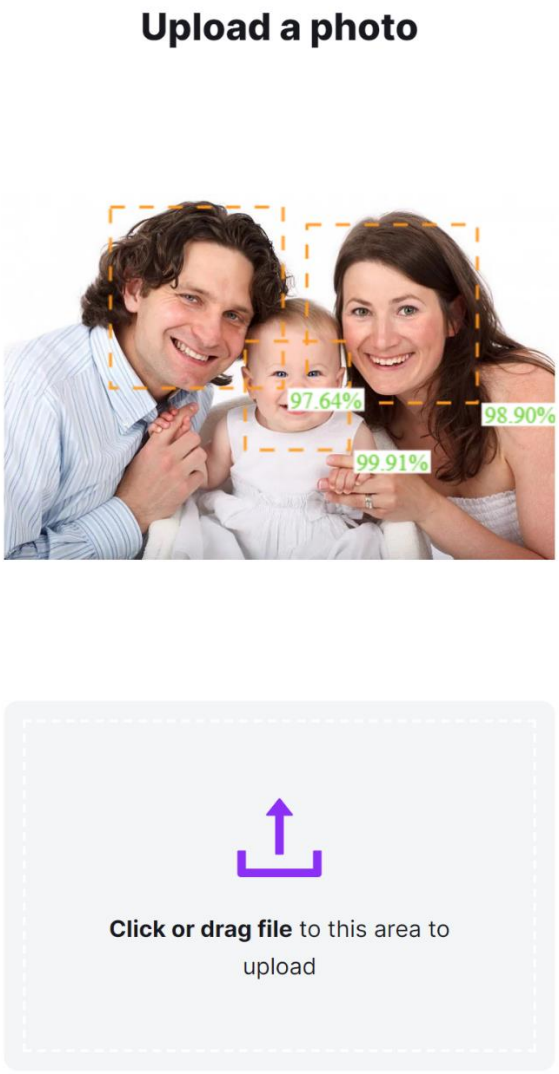

Рисунок 3.17 Приклад обробки зображення в mobile view

Сторінка з веб-камерою – це набір з компонентів, які відповідають за прийом та конвертування інформації з веб-камери, передачу даних до обробки засобами TensorFlow, що підвантажуються на етапі завантаження додатку та відображення області в якій виявлено обличчя. Приклади результатів обробки представлені на рисунках нижче.

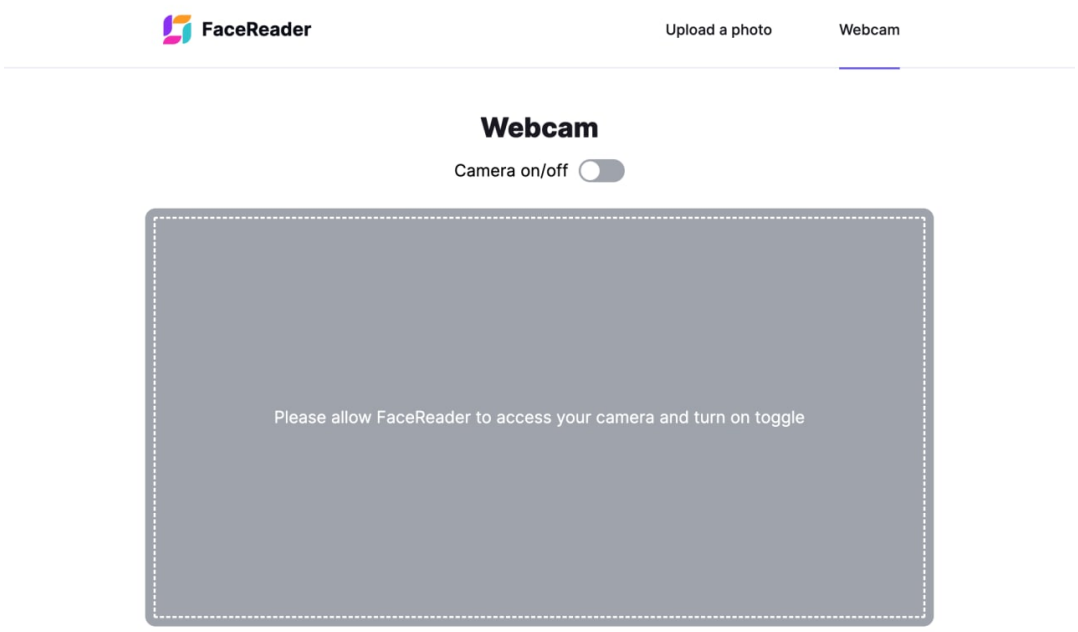

Рисунок 3.18 Сторінка до вмикання камери

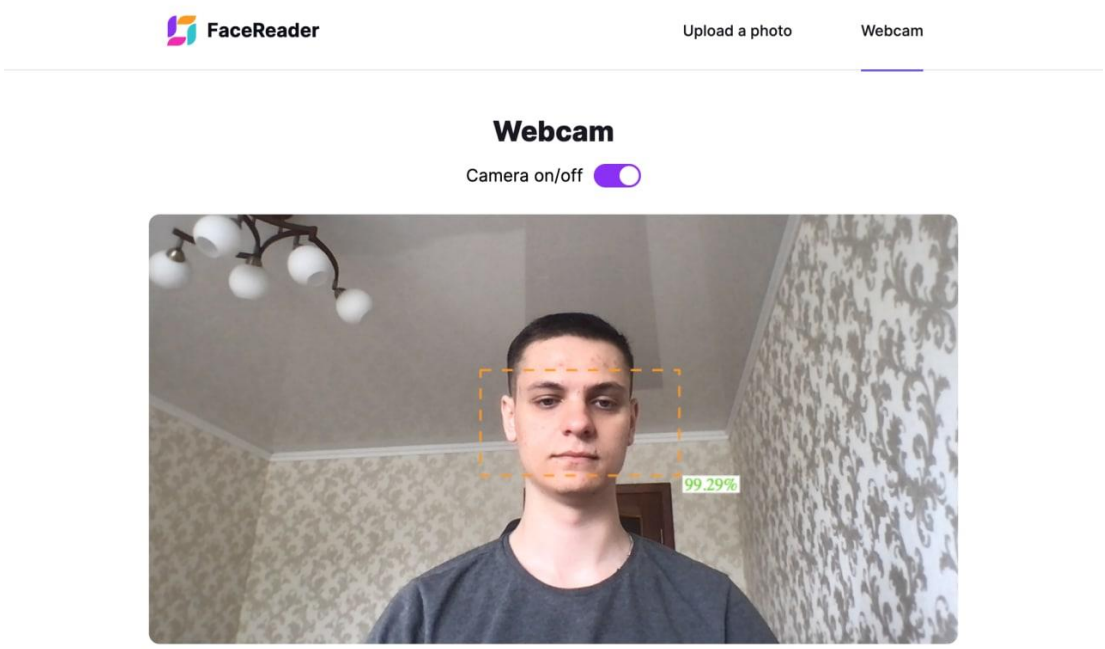

Рисунок 3.19 Приклад обробки зображення в реальному часі

3.4 Висновки до розділу

В даному розділі відображено архітектуру розробленого застосунку, вказано структуру програми та призначення різних папок. Описано принцип взаємодії TensorFlow з ReactJS та проілюстровано створений міст між TensorFlow та React.

Відображено створені сторінки: головна сторінка, сторінка з завантаженням зображень, сторінка з веб-камерою. Окрім цього описано підхід до реалізації основної частини – функціональність з розпізнаванням облич на зображеннях. Для цього у даному розділі описано принцип роботи моделі BlazeFace і її дата сету, що використовується для реалізації поставленої задачі. Описано основні компоненти, а саме: шапка сайту, підвал сайту, інформаційний банер, блок з принципом роботи застосунку, модуль веб-камери та модуль Drag-and-Drop файлі, та їх призначення.

ВИСНОВКИ

Метою даного дипломного проекту було розроблення веб-застосунку, який призначався для виявлення та відслідковування позиції обличчя. Приклад, де може допомогти даний застосунок, підрахування кількості людей, які відвідали певний заклад чи територія, для подальшого прогнозу прибутку. Це допоможе компаніям завчасно спрогнозувати, чи є дане місцерозташування – хороше.

В першому розділі розглянуто різні види нейронних мереж, більш детально розглянуто згорткові нейронні мережі, її складову та наявні архітектурні рішення.

Далі, на основі вимог цільового застосунку, проведено аналіз можливих засобів за допомогою яких можна реалізувати веб-застосунок. Було вирішено зробити у вигляді односторінкового застосунку з дотриманням архітектури Component-Based Architecture (CBA). Для цього було використано наступні інструменти:

- Мова програмування JavaScript
- Бібліотека ReactJS для створення клієнтських інтерфейсів
- Фреймворк TensorFlow для використання API моделі BlazeFace

В результаті створено веб-застосунок для виявлення обличчя за допомогою нейронних мереж, який має в собі наступну функціональність, що відповідає висунутим вимогам:

- Реалізація розпізнавання обличчя на статичних зображення
- Реалізація розпізнавання обличчя в реальному часі
- Трансформація масиву даних від TensorFlow та подавання інформації у зручному вигляді, в даному випадку – візуальному
- Доволі висока швидкодія сайту

- Сторінка з ознайомчою інформацією про застосунок
- Мінімалістичний та приємний UI
- Адаптивний дизайн – присутня мобільна та десктопна версія

Розробку програмного забезпечення виконано в повному обсязі. Програмний застосунок відповідає поставленим вимогам. Вихідний код застосунку лежить в відкритому доступі, що дозволяє використовувати його інших розробникам для своїх цілей.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. A Comprehensive Guide to Types of Neural Networks [Електронний ресурс]. Режим доступу: https://www.digitalvidya.com/blog/types-of-neural-networks.
2. TensorFlow [Електронний ресурс]. Режим доступу: https://en.wikipedia.org /wiki/TensorFlow
3. The Basics of Neural Networks [Електронний ресурс]. Режим доступу: https://medium.com/datadriveninvestor/the-basics-of-neural-networks-304364b712dc
4. Neural Network fundamentals [Електронний ресурс]. Режим доступу: https://medium.com/datadriveninvestor/neural-network-fundamentals-1956a3000c24
5. SSD: Single shot multibox detector / W. Liu, D. Anguelov, C. Szegedy et al [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1512.02325
6. Artificial intelligence. Object localization and detection / L. Araujo dos Santos [Електронний ресурс]. Режим доступу: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html
7. Tensorflow Object Detection API / J. Huang, V. Rathod, R. Votel et al. [Електронний ресурс]. Режим доступу: https://github.com/tensorflow/models/tree/master/research/object_detection
8. Tensorflow. Introduction, Architecture & Example: [Електронний ресурс]. Режим доступу: https://www.guru99.com/what-is-tensorflow.html
9. Django Framework: [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django
10. React vs Angular vs Vue.js — What to choose in 2021: [Електронний ресурс]. Режим доступу: https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d

11. A Comprehensive Guide to Convolutional Neural Networks: [Електронний ресурс]. Режим доступу: https://www.v7labs.com/blog/convolutional-neural-networks-guide
12. Common architectures in convolutional neural networks: [Електронний ресурс]. Режим доступу: https://www.jeremyjordan.me/convnet-architectures/
13. BlazeFace : A Machine Learning Model for Fast Detection of Face Positions and Key Points: [Електронний ресурс]. Режим доступу: https://medium.com/axinc-ai/blazeface-a-machine-learning-model-for-fast-detection-of-face-positions-and-key-points-5dcfb9429d72
14. What is facial recognition? How facial recognition works: Електронний ресурс]. Режим доступу: https://us.norton.com/internetsecurity-iot-how-facial-recognition-software-works.html
15. Svelte pros and cons, ecosystem overview and top resources: Електронний ресурс]. Режим доступу: https://daily.dev/blog/building-with-svelte-all-you-need-to-know-before-you-start
16. PyTorch vs. TensorFlow: Which Framework Is Best for Your Deep Learning Project?: Електронний ресурс]. Режим доступу: https://builtin.com/data-science/pytorch-vs-tensorflow
17. Golang Web Development – Everything You Need to Know: Електронний ресурс]. Режим доступу: https://distantjob.com/blog/golang-web-development/
18. 15 Best Programming Languages for Web Development in 2022: Електронний ресурс]. Режим доступу: https://www.globalmediainsight.com/blog/programming-languages-web-development

ДОДАТКИ

Додаток 1

Посилання на репозиторій

Посилання на GitHub репозиторій з вихідним кодом додатку:

https://github.com/alexgul1/react-tensorflow-face-detection