НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.852

До захисту допущено
В. о. завідувача кафедри ШІ
Олена Чумаченко
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 122 «Комп’ютерні науки»

на тему: «Система відстеження погляду в реальному часі для взаємодії з комп'ютером на основі методів глибокого навчання»

Виконав:
студент 2 курсу, групи КІ-11мп
Гончарук Олександр Петрович _____

Керівник:
доцент кафедри ММСА,
д.т.н., доц. Недашківська Н. І. _____

Рецензент:
доцент кафедри системного проектування
НН ІПСА КПІ ім. Ігоря Сікорського, к.т.н.
Богдан БУЛАХ _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ
2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти — другий (магістерський)

Спеціальність (ОПП) — 122 «Комп’ютерні науки» («Системи і методи штучного інтелекту»)

ЗАТВЕРДЖУЮ

В. о. завідувача кафедри ШІ

Олена Чумаченко

«___» _____ 2022 р.

ЗАВДАННЯ

на магістерську дисертацію студенту Гончаруку Олександру Петровичу

1. Тема дисертації:«Система відстеження погляду в реальному часі для взаємодії з комп'ютером на основі методів глибокого навчання», науковий керівник дисертації роботи Недашківська Надія Іванівна, доцент кафедри ММСА, д.т.н., доц., затверджені наказом по університету від «03» листопада 2022 р. № 4046-с

2. Термін подання студентом дисертації: 12.12.2022 р.

3. Об’єкт дослідження: Система відстеження погляду, що в реальному часі дозволяє передбачати напрямок погляду користувача та ідентифікувати його положення на дисплеї.

4. Предмет дослідження: Методи глибокого навчання, згорткові нейронні мережі, обробка зображень.

5. Перелік завдань, які потрібно розробити:

1) зробити огляд технічної література за темою;
2) дослідити актуальність роботи;
3) ознайомитись із існуючими методами розпізнавання погляду;
4) провести аналіз предметного середовища;

5) розробити архітектуру нейронної мережі та навчити модель;
6) провести аналіз точності розробленої моделі;
7) сконструювати та розробити архітектуру програмного забезпечення;
8) провести аналіз можливості запуску стартап проекту;
9) підготувати ілюстративні матеріали;
10) оформити пояснювальну записку.

8. Дата видачі завдання: 1 жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вивчення літератури за темою роботи.	28.10.2022 – 31.10.2022	Виконано
2.	Підготовка першого розділу.	31.10.2022 – 06.11.2022	Виконано
3.	Підготовка другого розділу.	06.11.2022 – 13.11.2022	Виконано
4.	Розробка програмного продукту.	13.11.2022 – 21.11.2022	Виконано
5.	Підготовка третього розділу	21.11.2022 – 01.12.2022	Виконано
6.	Підготовка частини стартап-проєкту	01.12.2022 – 12.12.2022	Виконано
8.	Оформлення пояснювальної записки	12.12.2022 – 18.12.2022	Виконано

Студент Олександр ГОНЧАРУК

Науковий керівник дисертації Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 19 рисунків, 30 таблиць, 1 додатку та 17 посилань — загалом 100 сторінок.

Об’єкт дослідження: система відстеження погляду, що в реальному часі дозволяє передбачати напрямок погляду користувача та ідентифікувати його положення на дисплеї.

Предмет дослідження: методи глибокого навчання, згорткові нейронні мережі, обробка зображень.

Мета дипломного проєкту: покращення наявних алгоритмів відстеження погляду завдяки використанню методів глибокого навчання.

У першому розділі відбувається знайомство з предметним середовищем, розроблення функціональних вимог та сформульовано постановку задачі.

У другому розділі описано процес навчання нейронної мережі, розглянуто архітектуру та особливості навчання моделі.

У третьому розділі описано особливості системи, розроблену архітектуру та процес взаємодії користувача з нею.

У четвертому розділі розглянуто можливість впровадження розробленої системи в якості стартап проєкту. Оцінено ризики, шляхи розвитку та проведено аналіз можливостей.

КЛЮЧОВІ СЛОВА: ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ЗОБРАЖЕНЬ, ГЛИБОКЕ НАВЧАННЯ, КОМП’ЮТЕРНИЙ ЗІР, ВІДСТЕЖЕННЯ ПОГЛЯДУ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ВЕБКАМЕРА

ABSTRACT

Structure and scope of the research work. The explanatory note of the diploma project consists of fourth sections, contains 19 images, 30 tables, 17 sources and 1 appendices - a total of 100 pages.

Object of research: deep learning methods, convolutional neural networks, image processing.

Subject of research: deep learning methods, convolutional neural networks, image processing.

The purpose of the diploma project: improvement of existing eye tracking algorithms using deep learning methods.

The first chapter introduces the subject environment, develops functional requirements and formulates the problem statement.

The second section describes the process of neural network training, architecture and features of model training.

The third section describes the features of the system, the developed architecture and the process of user interaction with it.

The fourth section considers the possibility of implementing the developed system as a startup project. Risks, ways of development and analysis of opportunities are assessed.

KEYWORDS: CONVOLUTIONAL NEURAL NETWORKS, IMAGE PROCESSING, DEEP LEARNING, COMPUTER VISION, GAZE TRACKING, CLIENT-SERVER ARCHITECTURE, WEBCAM

ЗМІСТ

ВСТУП

Людство винайшло велику кількість пристроїв вводу, що дозволяють нам впливати на процеси що відбуваються всередині комп’ютеру. Це надає можливість використовувати пристрій задля вирішення наших прикладних завдань. Таких як пошук інформації, перегляд відео, листування, написання текстів, створенню зображень, редагуванню відео і т.д. Саме користуючись пристроями вводу ми можемо взаємодіяти з апаратним забезпеченням обчислювального пристрою.

Зараз в нас є широка різноманітність таких пристроїв, наприклад: клавіатура, миша, сенсорний дисплей, сенсорна панель, трекпоінт, графічний планшет. Але постійно з’являються нові, які покликані проблеми та завдання що постають. Саме тому в сучасних смартфонах ми використовуємо сенсорний спосіб вводу, оскільки з часом зрозуміли що клавіатура не зовсім зручна для такого виду пристроїв. Також раніше не було потреби у мишах та трекпоінтах, оскільки більшість керування відбувалася за допомогою клавіатури. З часом з’явилися графічні користувацькі інтерфейси й для взаємодії з ними клавіатури виявилось недостатньо.

З розвитком технологій і їх активним впровадженням в повсякденне життя людини все частіше виникає питання про ефективність взаємодії із цифровими пристроями. Такі параметри як швидкість набору символів на клавіатурі, або швидкість переміщення курсора за допомогою миші мають чіткі обмеження. Тому, на наш погляд, управління пристроями за допомогою погляду є одним із перспективних та цікавих напрямків дослідження.

Причинами для використання такого способу можуть бути особливості будови пристрою, відмова наявних пристроїв, потреба у звільнених руках під час використання, фізіологічні особливості користувача та багато інших випадків використання.

Більшість технологій, що були створені для зчитування погляду використовують дорогі датчики. Вони надають достатньо точні виміри, але є

надмірно дорогими для впровадження на глобальному ринку. І звичайно це гальмує процес поширення цієї технології.

Ми ж поставили собі за мету надати можливість керувати пристроєм в реальному часі для будь-якої людини в якої присутня камера.

РОЗДІЛ 1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Постановка задачі

Метою роботи є вдосконалення наявних методів відстеження погляду користувача завдяки використанню методів глибокого навчання. Повна реалізація системи за допомогою використання мікросервісного архітектурного стилю.

В результаті необхідно працездатне програмне забезпечення, що буде в змозі виконувати всі висунуті вимоги, з можливістю масштабування, без непередбачуваних аварійних завершень та зручним інтерфейсом. Це дозволить комфортно користуватися системою як в повсякденному житті, так і у випадках настання критичних ситуацій.

1.2 Загальні положення та аналіз предметного середовища

Основним завданням роботи є створення системи відстеження погляду, яка зможе в реальному часі передбачати напрямок погляду користувача відносно камери пристрою, що в кінцевому результаті дозволить ідентифікувати положення погляду на дисплеї [7]. Також за допомогою

Розпочнемо з огляду бібліотек та фреймворків, що були використані при розробці програмного забезпечення.

OpenCV - це бібліотека програмних функцій та алгоритмів, що використовується для обробки зображень в основному в реальному часі. Ця бібліотека має відкритий вихідний код, повністю безплатна та багатоплатформова [1]. Також у разі наявності графічного процесора, зазначена бібліотека дозволяє виконувати обчислення на пристрої користувача.

За допомогою бібліотеки OpenCV, буде здійснюватися взаємодія з апаратним забезпеченням пристрою, зокрема камерою. Попередньо

здійснюється зчитування зображення з вказаної камери, а також попередня обробка.

PyTorch - це відкритий фреймворк машинного навчання, який використовують для комп’ютерного зору, обробки мовлення і т. д. Був спочатку розроблений Meta AI, а зараз є частиною некомерційного консорціуму розвитку Linux [15]. На основі PyTorch здійснювалась розробка та навчання моделі для визначення погляду користувача завдяки вхідному зображенню з камери.

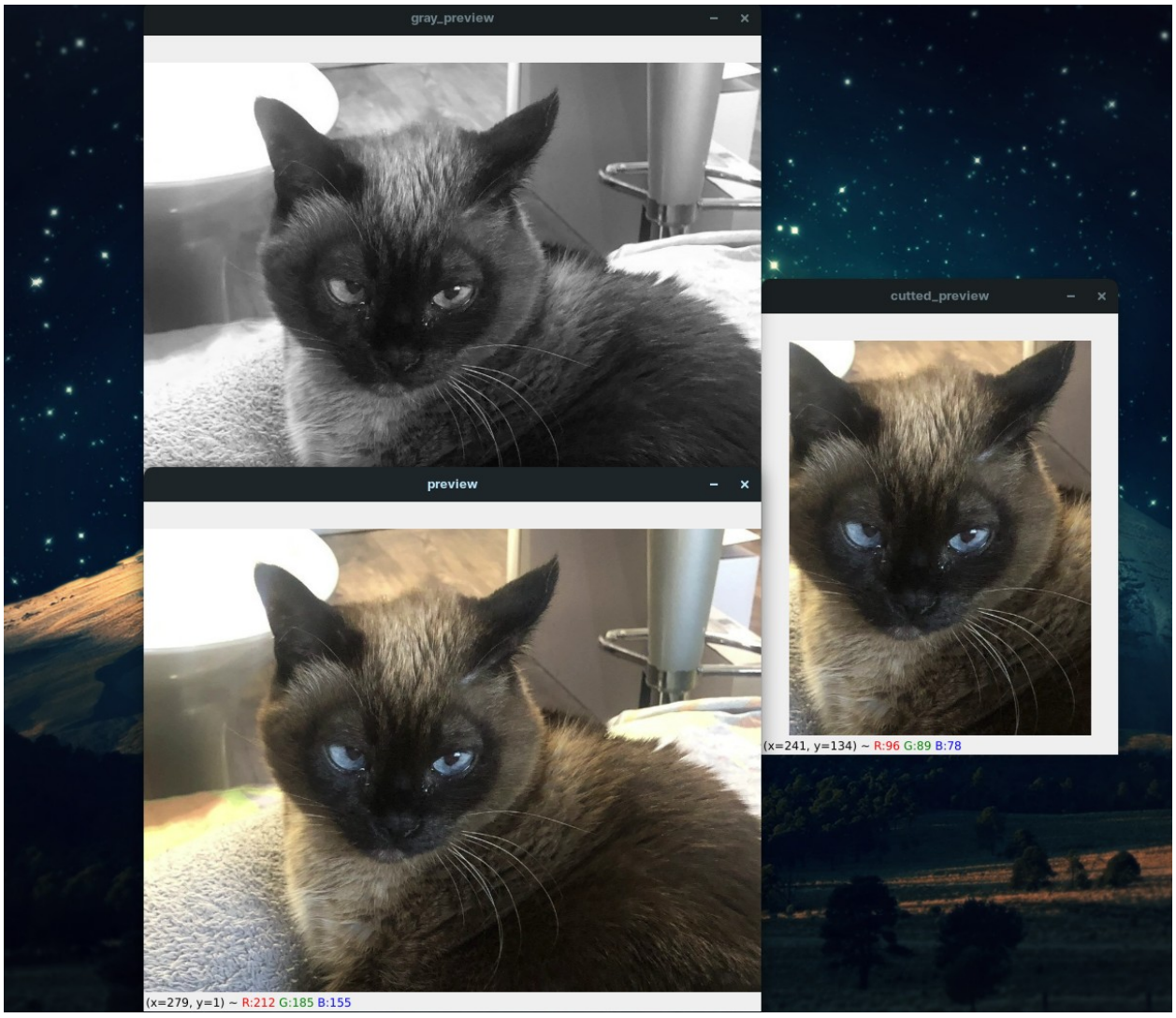

Рисунок 1.1 – Приклад читання камери та базового функціонала OpenCV

PyQT - оболонка написана на Python для бібліотеки Qt [2]. Створена для розробки багатоплатформового програмного забезпечення. В основному використовується для створення графічного інтерфейсу користувача.

Вище зазначена бібліотека використовується для зворотного зв’язку з користувачем, а саме відображення актуального стану програми та позиціювання поточного погляду користувача на дисплеї, для надання додаткових сповіщень.

Такі проблеми не раціонально вирішувати імперативним програмування. Оскільки це є малоефективним способом, бо розробка буде займати дуже багато часу та і точність передбачення буде неймовірно малою. Тому для цього ми будемо використовувати глибоке навчання.

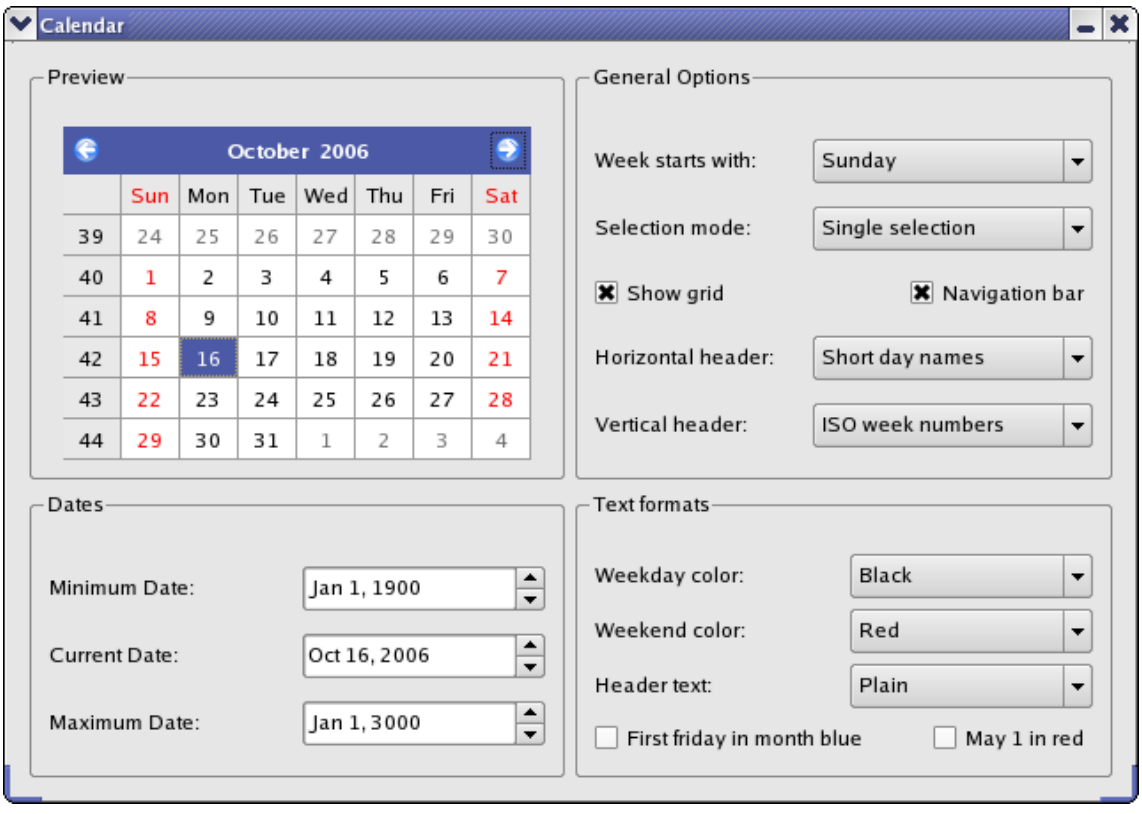

Рисунок 1.2 – Графічний застосунок створений за допомогою PyQT [11]

Глибоке навчання - це сімейство методів машинного навчання, що засновані на штучних нейронних мереж з навчанням на основі послідовних перетворень вхідного сигналу.

В даній роботі глибоке навчання розглядається в контексті окремого напряму – комп’ютерного зору, тому що поставлене завдання зводиться до обробки саме зображень [10]. Комп’ютерне бачення або комп’ютерний зір це набір підходів та методів, що дозволяють отримати високорівневе представлення цифрових зображень з подальшим вирішенням різного плану задач, наприклад класифікації. В нашому випадку буде здійснюватися перевірка наявності обличчя та очей на зображенні; визначення їх місце розташування, а також безпосереднє визначення позиціювання погляду не екрані.

Для того, щоб із зображення з обличчям людини отримати координати погляду необхідно створити власну згорткову нейронну мережу. Клас згорткових нейронних мереж був ідейно запозиченй із загальних принципів роботи зорової кори тварин [16].

1.3 Методи штучного інтелекту

Штучний інтелект – технологія створення інтелектуальних комп’ютерних програм яка відповідає за розв'язання задач і дій, що зазвичай виконує людина або інша жива істота.

Немає точної відповіді на те що таке штучний інтелект та чим саме він займається. У різних джерелах можна знайти свої власні визначення, але всі вони зводяться до того що це спосіб виконання образних або інтелектуальних функції живої істоти, вміння самостійно приймати рішення та розв’язувати проблеми на основі інформації із предметного середовища.

Штучний інтелект пройшов стрімкий ріст за останнє десятиліття та зараз користується неабиякою популярністю. Більшість найпопулярніших застосунків включають компоненти що використовують штучний інтелект.

Серед них Google Search, YouTube, Siri, Alexa, Google Assistant, Tesla, Amazon, Netflix і т.д. Також існує велика кількість вузькоспеціалізованих програм які не так відомі широкій публіці, але їх використання дуже поширене. Це можуть бути, наприклад, системи розпізнавання номерних знаків які фіксують правопорушення на дорозі, системи розпізнавання обличчя що встановлені на вулицях або підприємствах, більшість застосунків для перекладання тексту з мови на мову також використовують штучний інтелект.

Методи штучного інтелекту поєднують велику кількість інших методів, серед них методи представлення завдань та пошуку рішень, методи обчислювального інтелекту, методи аналізу та синтезу, моделювання та прогнозування та інші.

Серед найпопулярніших категорій штучного інтелекту є такі:

- Машинне навчання;
- Машинний зір;
- Автоматизація і робототехніка;
- Обробка природної мови.

У нашому випадку ми будемо використовувати саме машинний зір. Машинний зір це набір методів що дозволяють автоматично аналізувати передане зображення та виокремлювати необхідні ознаки, та автоматизовувати та аналіз і інспекцію на виробництві або промисловості.

Ми ж будемо використовувати ширше поняття, а саме комп’ютерний зір. Комп'ютерний зір — галузь яка допомагає програмам формувати високорівневе розуміння зображень. У нашому рішенні її використання необхідне для безпосереднього передбачення координат погляду користувача за допомогою зображення отриманого з вебкамери.

1.4 Методи навчання глибоких нейронних мереж

Розглянемо метод SGD – стохастичний градієнтний спуск. Градієнтні належать до загального класу методів безумовної оптимізації. В даній задачі градієнтний метод використовується для налаштування векторів ваг w шарів нейронної мережі. Нехай $y^*: X \to Y -$ цільова залежність, відома на об'єктах навчальної вибірки:

$$X^l = \left(x_l, y_i\right)_{i=1,}^{l} y_i = y^*\left(x_l\right)$$

Алгоритм f(x,w), який наближає залежність y* для лінійного класифікатора має наступний вигляд:

$$f\left(x, w\right) = \varphi\left(\sum_{i=1}^{n} w_j x^j - w_0\right)$$

де $\varphi\left(z\right)$ - функція активації.

Розглянемо сутність методу SGD [4].

На вхід подається:

- X^l – навчальна вибірка даних;
- η – темп навчання;
- λ – параметр, який забезпечує згладжування функціоналу якості навчання Q.

Результатом роботи методу є вектор ваг w.

Етапи методу SGD:

1. Ініціалізувати ваги w_j випадковим чином або сучасними методами Глоро або Хе, $j = 0, .., n$.

   Ініціалізувати поточну оцінку функціоналу якості навчання:

$$Q := \sum_{i=1}^{l} L\left(f\left(x_i, w\right), y_i\right)$$

2. Повторювати:

- Обрати об'єкт $x_i \in X^l$ з навчальної вибірки, наприклад, випадковим чином або одним з інших способів;

- Обчислити результуюче значення алгоритму $f(x_i, w)$ і помилку: $\xi_i := L(f(x_i, w), y_i)$;
- Виконати крок градієнтного спуску: $w := w - \eta L_a'(a(x_i, w), y_i)\varphi'(\langle w, x_i \rangle)x_i$;
- Знайти оцінку функціоналу якості: $Q := (1-\lambda)Q + \lambda_{\xi_i}$.

Етап 2 виконується поки значення Q не стабілізується або/та вектор ваг w не будуть змінюватися.

Модифікація методу градієнтного спуску із включенням імпульсів (моментів). Згідно з цим методом вектор ваг на кроці t+1 розраховується наступним чином:

$$w(t+1) = w(t) - \Delta w(t+1)$$

$$\Delta(t+1) = \gamma \Delta w(t) + \eta \frac{\partial E(w(t))}{\partial w(t)}$$

де $0 < \gamma < 1$ - параметр, який визначає яку частину попереднього градієнта беремо на поточному кроці, і яку частину нового градієнта використовуємо.

Метод Нестерова. Ідея в тому, щоб «дивитися вперед» за напрямом вектору оновлення параметрів.

Згідно з методом Нестерова вектор ваг на кроці t+1 розраховується [4]:

$$w(t+1) = w(t) - \left[\gamma \Delta w(t) - \eta \frac{\partial E[w(t) - \gamma \Delta w(t)]}{\partial w(t)}\right]$$

де $0 < \gamma < 1$ - параметр, який визначає яку частину попереднього градієнта беремо на поточному кроці, і яку частину нового градієнта використовуємо.

Метод Адам. Adam – метод оптимізації з адаптивною швидкістю навчання. Метод Adam вважається достатньо стійким до вибору гіперпараметрів, однак параметр «темп навчання» іноді потрібно брати відмінним від пропонованого за замовчуванням [9].

На вхід методу подаються наступні величини:

- коефіцієнти ρ1 і ρ2 є [0, 1) експоненціального затухання для оцінок моментів, за умовчанням дорівнюють 0.9 і 0.999 відповідно;
- коефіцієнт δ для забезпечення чисельної стійкості розв’язку, приймає невелике додатне значення;
- початкові значення параметрів θ.

Етапи Adam [9].

Крок 1. Ініціалізувати змінні для першого і другого моментів s = 0, r = 0. Ініціалізувати крок за часом t = 0.

Крок 2. while критерій зупинки не виконано do

Вибрати з навчального набору мініпакет m прикладів {x (1), ..., x (m)} і мітки y (i).

Обчислити градієнт: $g \leftarrow (1/m)\nabla_\Theta \sum_i L\left(f\left(x(i);\Theta\right), y(i)\right)$.

$t \leftarrow t+1$

Оновити зміщену оцінку першого моменту: $s \leftarrow \rho 1\, s+(1-\rho 1)\, g$

Оновити зміщену оцінку другого моменту: $r \leftarrow \rho 2\, r+(1-\rho 2)\, g \odot g$

Скорегувати зміщення першого моменту:

$$\hat{s} \leftarrow \frac{s}{1-p_1^t}$$

Скорегувати зміщення другого моменту:

$$\hat{r} \leftarrow \frac{r}{1-p_1^t}$$

Обчислити приріст ваги:

$$\Delta\theta = -\xi \frac{s}{\sqrt{\hat{r}} + \delta}$$

Оновити ваги: $\Theta \leftarrow \Theta + \Delta\Theta$

end while.

У методі Adam ваги налаштовуються наступним чином:

$$S_t := a * S_{t-1} + (1-a) * \nabla E_t^2; S_0 := 0$$

$$D_t := \beta * D_{t-1} + (1-\beta) * \nabla E_t; D_0 := 0$$

$$g_t := \frac{D_t}{1-\beta} * \sqrt{\frac{1-a}{S_t}}$$

$$\Delta W_t := \eta * (g_t + p * W_{t-1}) + \mu * \Delta W_{t-1}$$

де η – коефіцієнт швидкості навчання,

∇E – градієнт функції втрати,

μ – коефіцієнт моменту,

ΔWt-1 – зміна ваг на попередній ітерації,

ρ – коефіцієнт регуляризації,

Wt-1 – значення ваг на попередній ітерації,

α = 0.999, β = 0.9

1.5 Модель згорткової нейронної мережі

Найважливіша складова згорткової нейронної мережі (CNN) – це згортковий шар (рис. 1.3). Архітектура моделі CNN будується таким чином, щоб нейрони в першому згортковому шарі були пов’язані тільки з пікселями вхідного зображення у власних рецепторних полях (рис. 1.4). Кожний нейрон у другому згортковому шарі, своєю чергою, пов'язаний тільки з нейронами, які знаходяться всередині прямокутної області в першому згортковому шарі. Аналогічно будуються й інші згорткові шари глибокої мережі CNN. В результаті така архітектура дозволяє моделі CNN шукати низькорівневі ознаки за допомогою першого прихованого згорткового шару, далі поєднувати їх в ознаки вищого рівня в наступному прихованому згортковому шарі тощо. Подібна ієрархічна структура є в реальних зображеннях, тому мережі CNN показують дуже добрі результати при розпізнаванні зображень.

Розглянемо властивості моделі CNN.

Розріджена взаємодія. У традиційній нейронній мережі кожний вихідний блок взаємодіє з кожним вхідним. В згорткових мережах взаємодія розріджена (рис. 1.5) і тому можна зберігати менше параметрів порівняно з багатошаровим персептроном. Це пов’язано також із властивістю локальності мережі CNN: обробка частини зображення відбувається незалежно від того де ця частина розташована у вхідному зображенні.

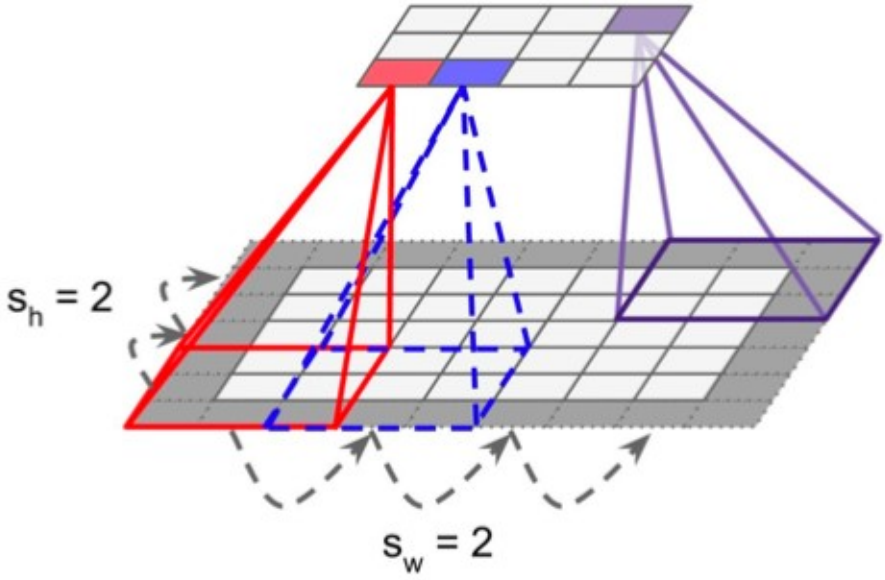

Рисунок 1.3 – Зниження розмірності із застосуванням згорткового шару з страйдом рівнем 2 за обома напрямками [4]

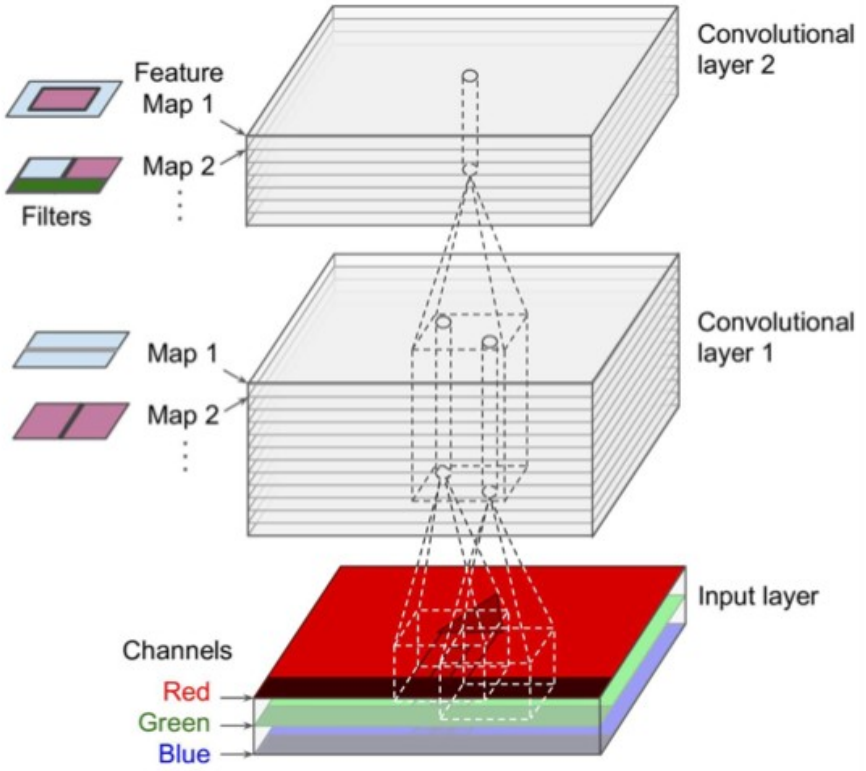

Рисунок 1.4 – Модель CNN і зображення з трьома каналами [4]

На рисунку 1.5 зверху зображено шар s, утворений згорткою з ядром розмірності 3, отже елемент x3 впливає лише на 3 виходи. Знизу на рисунку 1.5 показано традиційну модель – багатошаровий персептрон, де зв’язність не є розрідженою.

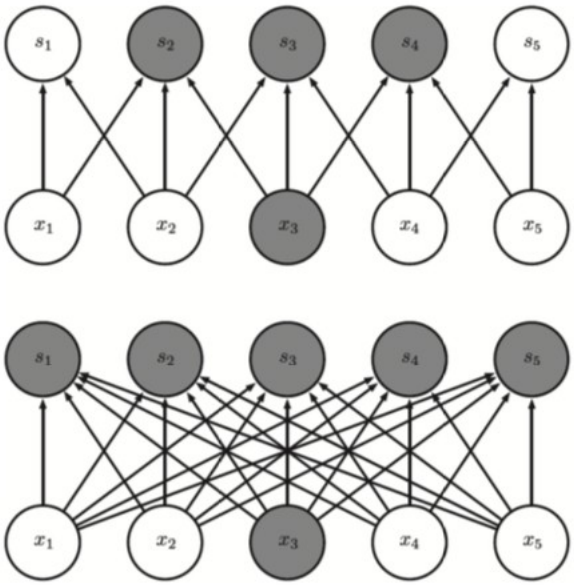

Рисунок 1.5 – Розріджена взаємодія в згортковому шарі (зверху) [9]

Розділення параметрів. В мережі CNN один і той самий параметр може використовуватись в кількох функціях згортки. В традиційній мережі – багатошаровому персептроні – кожний параметр використовується тільки один раз (розділення параметрів немає) під час обчислення скалярного добутку вихідного сигналу нейрона.

Ще одна властивість мережі CNN в тому, що вона може якісно описувати складні взаємодії між багатьма змінними за умови, що мережа складається з простих елементів, де кожний з них описує тільки розріджену взаємодію. На рисунку 1.6 наведено приклад опосередкованої взаємодії елемента g3 з усіма вхідними елементами (x1 – x5).

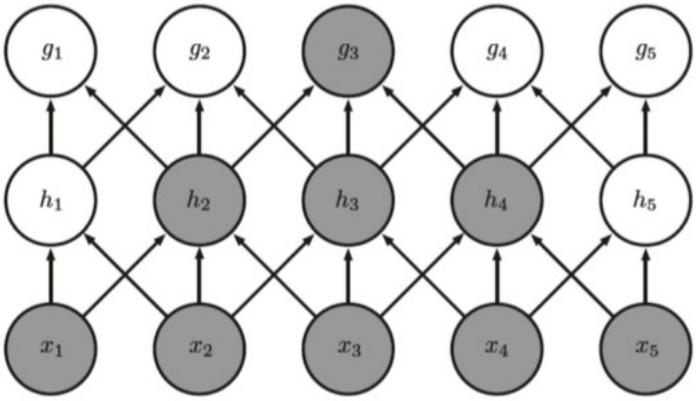

Рисунок 1.6 – Опосередкована взаємодія блоків в мережі CNN [9]

1.6 Ознаки Хаара

Ознаки Хаара — набір ознак цифрового зображення, що використовуються для розпізнавання образів.

Більшість алгоритмів що працюють із цифровими значеннями зображення мають велику обчислювальну складність. Кожен піксель складається із трьох цифрових значень інтенсивності червоного, синього та зеленого кольору. Для зменшення об’єму вхідних даних можна перетворити зображення з кольорового на чорно-біле, якщо це не шкодить подальшому аналізу зображення. Наприклад у нашому випадку ми саме так і робимо, оскільки для визначення напряму погляду нам можна знехтувати кольором зображення.

Папагеоргію [14] у своїй роботі розглянув множину ознак основаних на вейвлетах Харра. Ідею використання вейвлетів Хаара було адаптовано Віолою та Джонсом [17]. Вони розробили те що пізніше було названо ознаками Хаара. Ознаки Хаара складаються із суміжних прямокутних областей (рис. 1.7), які розміщаються на зображені та формують так зване вікно.

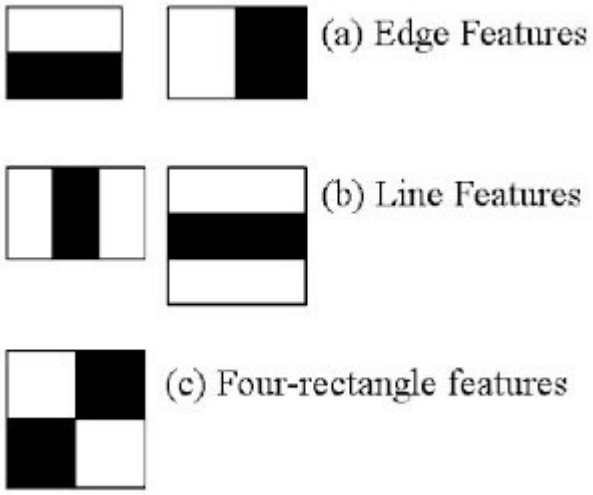

Рисунок 1.7 – Приклад ознак Хаара [6]

Після цього підраховується сума інтенсивності пікселів в кожній з областей та обчислюється різниця між ними (рис. 1.8).

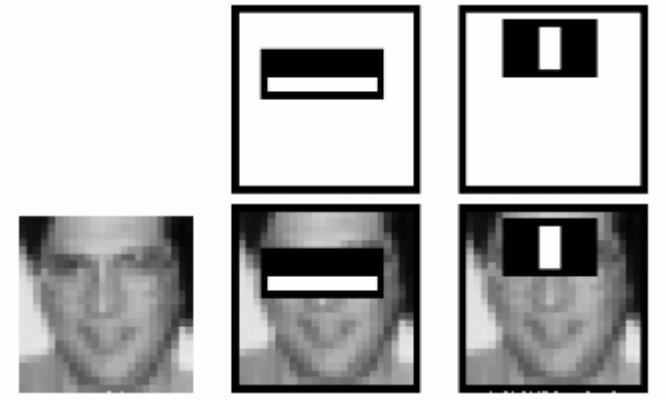

Рисунок 1.8 – Накладання ознак Хаара на зображення обличчя [6]

Отримана різниця і є значенням певної ознаки. Вікно ознак визначеного розміру рухається по зображенню і для кожної області розраховує ознаку Хаара. Для досягнення прийнятної точності не достатньо лише одного ядра ознак, тому набори ознак організовані в каскадні класифікатори.

Внаслідок невеликої кількості та складності обчислень досягається висока швидкодія. У системі що розробляється ми будемо використовувати

два набори ознак Хаара для розпізнавання обличчя та очей. Натреновані ознаки надаються бібліотекою OpenCV [12] яка використовується для отримання зображення з камери та їх попередньої обробки.

Виявлення обличчя на зображенні необхідне, щоб виконати вимоги щодо позиціювання обличчя в конкретній області зображення (прямо по середині відносно вебкамери та дисплею).

Розпізнавання очей необхідне для фіксування моменту кліпання правим або лівим оком. Це використовується для подальшої симуляції натискання відповідної кнопки миші, що дозволяє виконувати операції на робочому столі.

Шляхом того що алгоритм є швидким, цей процес не займає багато часу і дозволяє виконувати його для кожного отриманого кадру з вебкамери. Також це залишає достатньо обчислювальних можливостей для подальшого виконання розпізнавання напряму погляду.

1.7 Розроблення функціональних вимог

Для якісного виконання поставленої задачі необхідно сформувати необхідні функціональні вимоги яким повинна відповідати розроблена система. Це дозволить врахувати всі основні сценарії використання системи користувачами та забезпечення комфортного використання.

В таблиці 1.1 - 1.4, описані варіанти калібрування камери, розпізнавання погляду та дій які може виконувати користувач завдяки розробленому програмному забезпеченню.

Таблиця 1.1 – Варіант використання UC-1

Назва	Калібрування камери
Опис	Користувач вперше запускає програму на пристрої
Учасники	Користувач
Передумови	Користувацький клієнт успішно приєднався до сервера розпізнавання погляду. Обличчя користувача добре освітлене та відкрите.
Постумови	Погляд користувача розпізнається з похибкою не більше ніж 2 см в горизонтальній і вертикальній площині.
Основний сценарій	Користувач запускає програму. Після чого автоматично запускається скрипт який надає користувачу вказівки. А саме протягом однієї хвилини стежити за червоним вказівником на дисплеї. За цей час додаток зможе підлаштуватись під камеру що робить зображення та розміри дисплея.
Розширення сценаріїв	Під час калібрування користувач зникає з об'єктива камери: 1) Програма припиняє процес калібрування та виводить повідомлення про помилку з проханням повернути обличчя назад та розпочати калібрування спочатку.

Таблиця 1.2 – Варіант використання UC-2

Назва	Відстеження погляду користувача
Опис	Користувач запустив додаток та розглядає об’єкти розміщені на дисплеї пристрою. Обличчя користувача добре освітлене та відкрите.
Учасники	Користувач
Передумови	Користувацький клієнт успішно приєднався до сервера розпізнавання погляду. Користувач успішно виконав процедуру калібрування.
Постумови	На дисплеї пристрою з’являється червоний круг, який в реальному часі переміщається на місце погляду користувача.
Основний сценарій	Користувач запустив додаток та почав дивитись на дисплей. Додаток успішно розпізнає напрям погляду користувача і відображає червоне коло на дисплеї в передбаченому місці погляду.
Розширення сценаріїв	Обличчя користувача зникає з об’єктива або користувач дивиться не на дисплей пристрою: 1) Програма перестає відстежувати погляд користувача, та відображати червоне коло на дисплеї.

Таблиця 1.3 – Варіант використання UC-3

Назва	Симуляція натискання лівої кнопки миші
Опис	Користувач фокусується на елементі інтерфейсу та кліпає лівим оком (око закрите не менше ніж 0.3 секунди)
Учасники	Користувач
Передумови	Користувацький клієнт успішно приєднався до сервера розпізнавання погляду. Користувач успішно виконав процедуру калібрування.
Постумови	Відбулася симуляція подій натискання лівої кнопки миші, елемент на якому сфокусувався користувач отримав подію “натискання” та відреагував на неї.
Основний сценарій	Користувач поглянув на елемент інтерфейсу і кліпнув лівим оком. Після чого елемент інтерфейсу відреагував відповідним чином.
Розширення сценаріїв	Користувач сфокусувався на місці де немає активних елементів: 1) Операційна система отримала подію “лівий клік”, але на дисплеї нічого не змінилось, оскільки клік відбувався по місцю де немає активного елементу.

Таблиця 1.4 – Варіант використання UC-4

Назва	Симуляція натискання правої кнопки миші
Опис	Користувач фокусується на елементі інтерфейсу та кліпає правим оком (око закрите не менше ніж 0.3 секунди)
Учасники	Користувач
Передумови	Користувацький клієнт успішно приєднався до сервера розпізнавання погляду. Користувач успішно виконав процедуру калібрування.
Постумови	Відбулася симуляція подій натискання правої кнопки миші, елемент на якому сфокусувався користувач відкрив власне контекстне меню або виконав іншу запрограмовану подію.
Основний сценарій	Користувач поглянув на елемент інтерфейсу і кліпнув правим оком. Після чого елемент інтерфейсу відреагував відповідним чином.
Розширення сценаріїв	Користувач сфокусувався на місці де немає активних елементів: 1) Операційна система отримала подію “правий клік”, відкрилось контекстне меню активного застосунку. Якщо клік відбувся за межами застосунку, відкривається контекстне меню операційної системи.

Таблиця 1.5 – Варіант використання UC-5

Назва	Початок роботи з системою
Опис	Користувач розпочинає свою роботу із системою.
Учасники	Користувач
Передумови	Користувач має в наявності вебкамеру. Також необхідна відеокарта на якій будуть виконуватись обчислення, або як альтернатива доступ в інтернет. У випадку наявності інтернету система може встановити з’єднання з віддаленим сервером та виконувати на ньому необхідні обчислення.
Постумови	Система була успішно запущена без помилок та готова до використання.
Основний сценарій	Користувач запускає скрипт із розробленим застосунком, після чого спостерігає вікно для калібрування (якщо система запущена вперше) або повідомлення про те що система була успішно завантажена та готова до роботи.

Продовження таблиці 1.5

Розширення сценаріїв	Користувач запустив скрипт, але немає відеокарти: 1) Система відображає на екрані текст про те що на вашому пристрої відсутня відеокарта; 2) Система намагається встановити з’єднання з віддаленим сервером. Користувач запустив скрипт, але в нього немає відеокарти та доступу в інтернет: 1) Система відображає на екрані текст про те що на пристрої немає відеокарти та доступу до інтернету; 2) Система закінчує свою роботу.

Таблиця 1.6 – Варіант використання UC-6

Назва	Закінчення роботи системою
Опис	Користувач завершує свою роботу із системою.
Учасники	Користувач
Передумови	Перед цим система була успішно запущена та готова до роботи.
Постумови	Система завершила роботу та звільнила ресурси що використовувались на відеокарті. У разі використання віддаленого сервера, відеокарта на сервері вивільняє ресурси що використовувались.
Основний сценарій	Користувач припиняє роботу запущеного скрипту і спостерігає прощальне повідомлення з інформацією про успішне завершення.

Продовження таблиці 1.6

Розширення сценаріїв	Користувач зупиняє роботу скрипту, але система втратила зв’язок із відеокартою: 1) Система відображає на екрані текст про те що зв’язок із відеокартою було втрачено, та вона не може гарантувати успішне звільнення ресурсів; 2) Система аварійно завершує роботу. Користувач зупиняє роботу скрипту, але система немає доступу до інтернету: 1) Система відображає на екрані текст про те що доступ до інтернету було втрачено; 2) Віддалений сервер не отримує запити від клієнта протягом однієї хвилини після чого вивільняє ресурси. 3) Система аварійно завершує роботу.

1.8 Висновки до розділу

У даному розділі було розглянуто предметну область в контексті якої здійснюється дослідження та розробка системи. Зокрема проведено ознайомлення з бібліотеками та фреймворками що будуть використовуватись при розробці системи. Проведено ознайомлення із методами штучного інтелекту та ознаками Хаара. Формалізовано функціональні вимоги до майбутнього програмного продукту, що дозволило ідентифікувати учасників системи (лише користувач) та можливі сценарії використання програмного забезпечення.

РОЗДІЛ 2 НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

2.1 Опис навчального набору даних

Успіхи у сфері глибокого навчання, зокрема комп'ютерного зору активно використовуються в повсякденній людській діяльності. Однак залишаються сфери, які з певних причин використовують застарілий арсенал методів і підходів для розв'язання відповідних задач. Через відсутність великої кількості навчальних даних і погано спроєктований процес їх збору така проблема, як відстеження погляду не може бути вирішена повністю за допомогою глибокого навчання або інших методів та підходів розв'язання даної задачі.

Останні напрацювання, що безпосередньо стосуються збору та валідації навчальних даних, дозволяють використовувати повний арсенал методів комп'ютерного зору для розв'язання задачі відстеження погляду. Gaze Capture [5] дата сет включає 1450 різних людей і містить більше 2.5M зображень, які дозволяють ідентифікувати де на моніторі сконцентрований погляд, використовуючи лише зображення обличчя людини.

Мінливість даних дає загальну оцінку репрезентативності даних. Для цього в роботі було використано метод Dataset Characteristics [5], що оцінює позицію голову (h) та напрям погляду (g) для кожного зображення. Кінцевий розподіл зазначених двох величин в порівнянні з іншими наявними дата сетами: MPIIGaze [3] та TabletGaze [8] зображено на рисунку 2.1.

Зазначені розподіли підтверджують варіативність положення голови та погляду в порівнянні з іншими наявними дата сетами, що безпосередньо впливає на якість навчання згорткових нейронних мереж.

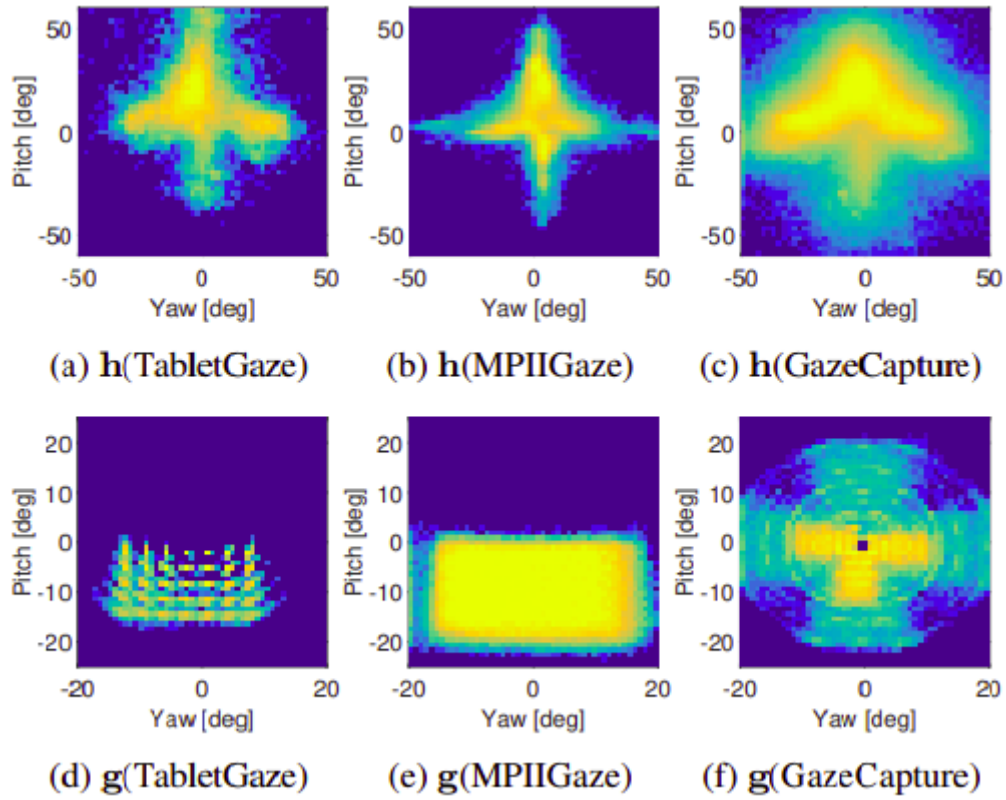

Рисунок 2.1 – Розподіл положення голови й напрямку погляду в порівнянні з наявними наборами даних [5]

2.2 Архітектура нейронної мережі

Мета даної роботи полягає в розробці підходу, що використовує інформацію представлену в вигляді зображення, щоб передбачати координати погляду на моніторі персонального комп'ютера. Було спроєктовано архітектуру нейронної мережі на основі згорткових нейронних мереж із додатковим використанням механізму уваги.

На вхід моделі передавався: 1 зображення лиця із відповідним його розміщенням на зображенні (іменується як маска лиця); і додатково 2 зображення лівого і правого ока відносно розміщення лиця (голови). Комбінуючи цю інформацію, модель здатна визначати координати погляду користувача. Загальна архітектура нейронної мережі із відповідними розмірностями зображена на рисунку 2.2.

Варто звернути увагу, що зображення лівого і правого ока передаються на вхід мережі, як окремі входи, хоча саме лице вже містить дану інформацію. Даний підхід було зроблено з метою збільшити роздільну здатність зображення, що спрощує процес ідентифікації напрямку погляду нейронною мережею.

Оскільки спостереження включають використання різних цифрових пристроїв: телефонів, планшетів та персональних комп'ютерів, необхідно було створити універсальну цільову зміну. В цьому випадку — це координати x, y відносно камери, які приймають значення в сантиметрах.

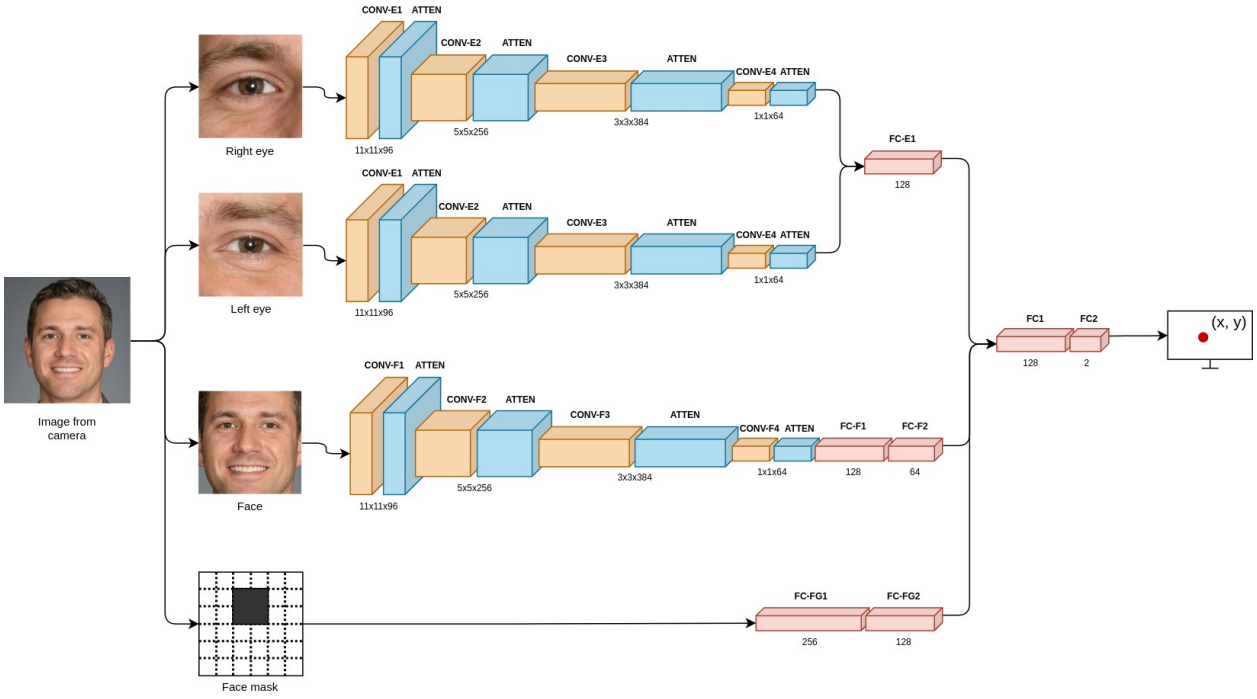

Рисунок 2.2 – Високорівнева архітектура нейронної мережі

Дата сет GazeCapture містить 2445504 зображень, серед яких лише 1490059 використовуються для навчання, валідації та тестування. Іншими словами, спостереження отримані від 1471 людини використовувались для навчання нейронної моделі, 50 людей для валідації, а також 150 людей для перевірки кінцевої якості моделі.

Враховуючи, що розміри кожного зображення мають різні розміри, було проведено їх трансформацію до єдиного розміру 224 на 224 пікселів за допомогою бібліотеки torchvision. Архітектура нейронної мережі було реалізовано за допомогою фреймворку Torch для глибокого навчання. Конкретна архітектура моделі розбита на функціональні блоки, які наведено в таблицях 2.1 та 2.2 відповідно.

Таблиця 2.1 – Модуль обробки зображень

Номер шару	Image Module
1	Conv2d(96/11/4/0) MaxPool2d(3/2) BatchNorm2d(96) MultiHead(96/3)
2	Conv2d(256/5/1/2) MaxPool2d(3/2) BatchNorm2d(256) MultiHead(96/3)
3	Conv2d(384/3/1/1) Conv2d(64/1/1/0)

де Conv2d(кількість фільтрів / розмір ядра згортки / крок зміщення / padding), MaxPool2d(розмір вікна / крок зміщення); BatchNorm2d(кількість фільтрів); MultiHead(кількість фільтрів / кількість шарів уваги). Після кожного згорткового шару використовується активаційна функція ReLU.

Таблиця 2.2 – Повнозв’язані шари для агрегації даних

Номер шару	Eyes Module	Face Module	Grid Module	Aggregate Module
1	Linear (18432, 128)	Linear (9216, 128)	Linear (784, 256)	Linear (320, 128)
2		Linear (9216, 128)	Linear (256, 128)	Linear (128, 2)

де Linear(кількість вхідних нейронів, кількість вихідних нейронів). Після кожного повнозв’язаного шару застосовується активаційна функція ReLU.

Таким чином, на виході нейронної мережі отримує два значення, які відповідають за координату погляду x та y відповідно.

2.3 Особливості навчання моделі

Оскільки дана задача відстеження погляду формулюється як регресійна задача, то в якості функції оптимізації було обрано функцію похибки Хубера. Зазначена функція об'єднує переваги Манхетенської метрики (L1), яка менш чутлива до викидів, а також Евклідової метрики (L2), яка забезпечує згладжування в околі нуля:

$$l_n = \begin{cases} 0.5\left(x_n - y_n\right)^2, if\ \left|x_n - y_n\right| < delta \\ delta\ *\ \left(\left|x_n - y_n\right| - 0.5\ *\ delta\right), otherwise \end{cases}$$

В якості оптимайзера було вибрано Adam із коефіцієнтом навчання (learning rate) рівним 0.0003, параметрами β1 = 0.9 та β2 = 0.999. Додатково використовувався weight_decay = 0.0001, як аналог L2 регулізації з адаптацією коефіцієнта навчання.

Процес навчання зображено на рисунку 2.3

Розмір батчу містить 128 зображень, а навчання проводиться протягом 100 епох.

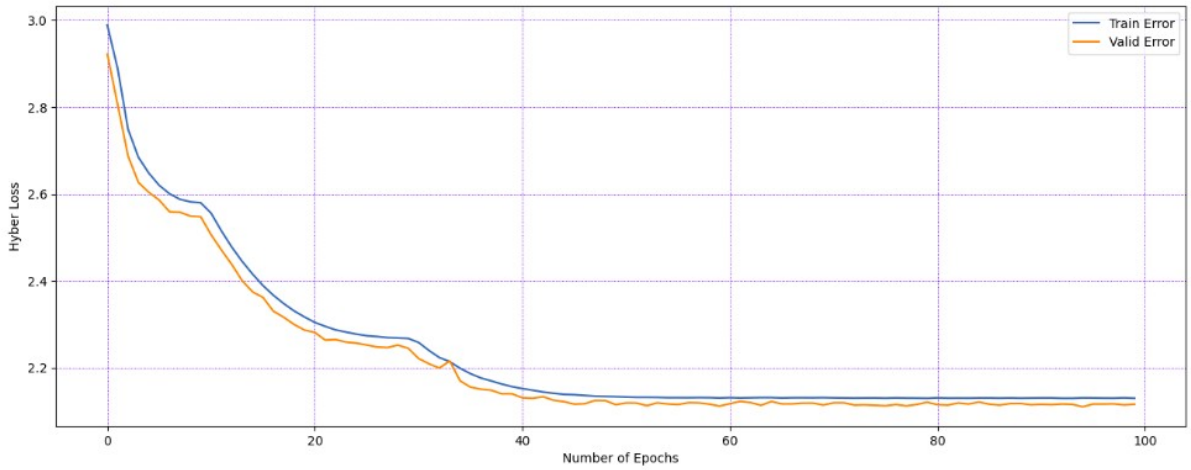

Рисунок 2.3 – Прогрес навчання нейронної мережі

Для покращення результатів розпізнавання можна використовувати інші набори даних. Наприклад лише той що був сформований за допомогою використання настільного персонального комп’ютера, лише ноутбуку, лише планшету, і т. д. Також можна використовувати набори що були отримані при поганому, або навпаки хорошому освітлені. Набори для людей з розладами зору (косоокість, короткозорість, далекозорість).

Отримані натреновані моделі будуть зберігатися в хмарі, а користувач зможе динамічно перемикатися на необхідну йому модель. Такий спосіб допоможе покращити якість роботи при використанні інших типів пристроїв, незвичних зовнішніх умовах або для окремих категорій людей.

Шляхом розробки архітектури що дозволить оперативно вносити зміни, таке доповнення функціонала не займе багато часу.

2.4 Висновки до розділу

В межах даного розділу було описано загальні характеристики навчального набору даних. Проведено порівняльну характеристику існуючих дата сетів. Спроєктовано та імплементовано модель глибоко навчання, що використовує згорткові шари та механізм уваги. Формалізували навчання

нейронної мережі, як оптимізаційну задачу регресії із використанням функції похибки Хубера.

РОЗДІЛ 3 ПОСТАНОВКА Й АНАЛІЗ ЕКСПЕРИМЕНТУ

3.1 Моделювання та аналіз програмного забезпечення

3.1.1 Особливості взаємодії частин системи

При плануванні архітектури програмного забезпечення було взято до уваги той факт, що не всі потенційні користувачі можуть мати можливість використовувати розроблену модель розпізнавання координат погляду.

У багатьох сучасних пристроях може бути відсутня відеокарта. Навіть якщо вона присутня, то це не може гарантувати нормальної роботи моделі, оскільки її продуктивність може бути недостатньою або в неї може не вистачати об’єму відеопам’яті для завантаження моделі.

Тому систему було розділено на дві основні частини: клієнтську та серверну. Це дозволить використовувати її на будь-якому пристрої, що має вихід в інтернет та вебкамеру.

Якщо ж у клієнта достатньо продуктивний пристрій або в нього немає доступу до інтернету то він все ще може розгорнути клієнтську та серверну частину локально. Це навіть може додати швидкодії, оскільки ми прибираємо затримку спілкування компонентів через інтернет.

3.1.2 Високорівневе представлення

Як вже було зауважено система поділяється на дві основні частини. Клієнтська яка запускається безпосередньо на пристрої користувача і споживає мало ресурсів. І серверна частина яка знаходиться в хмарі. На неї перенесено основні обчислювальні операції.

На рисунку 3.1 можна спостерігати високорівневу схему систему. Систему було розроблено стійкою до відмов.

Відмовостійкість була реалізована шляхом можливості регулювання кількості дублювальних компонентів. Саме тому на рисунку можна бачити

що між клієнтом та конкретним екземпляром сервера що надає відповідь розміщений LoadBalancer, який розподіляє навантаження між запущеними компонентами.

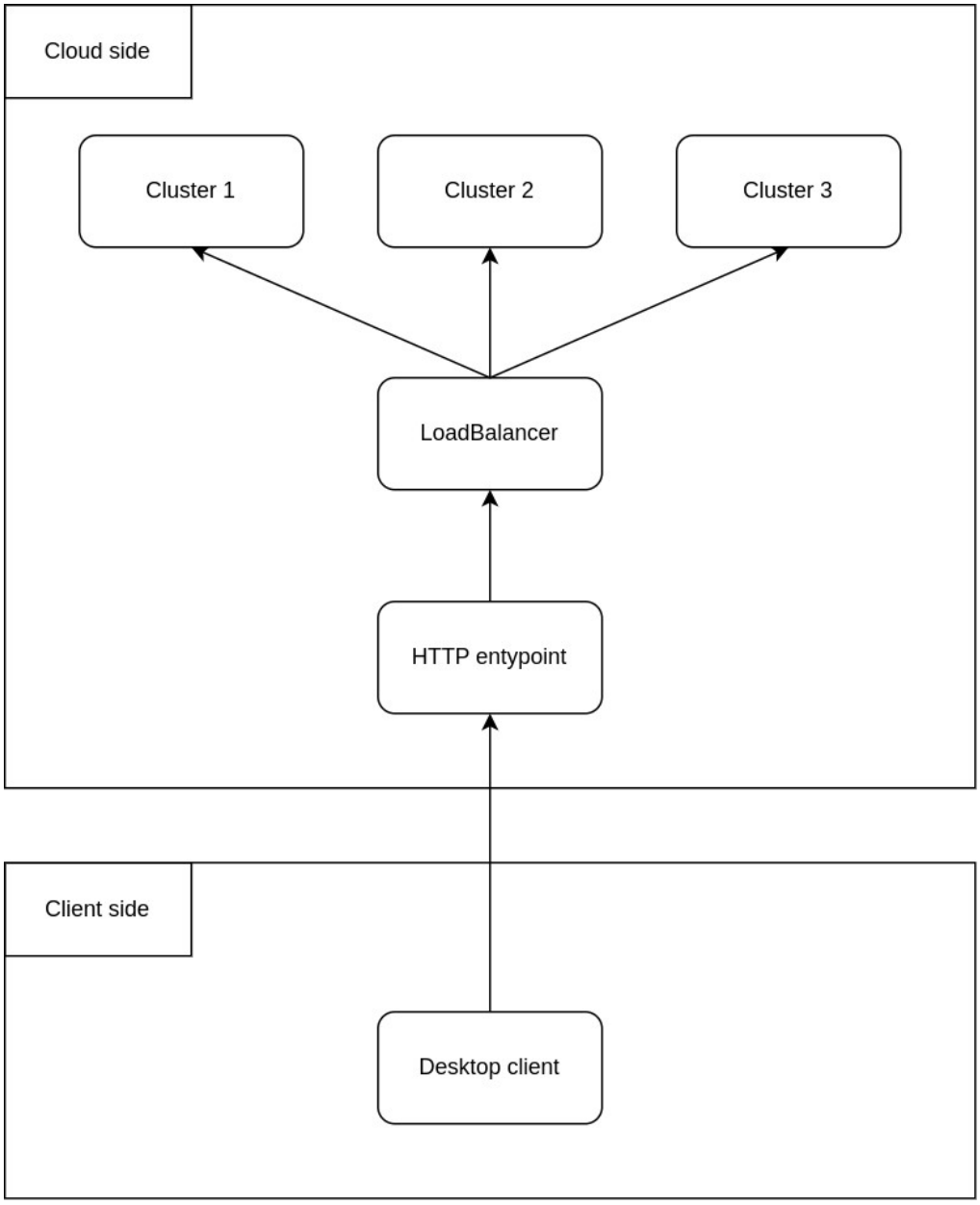

Рисунок 3.1 – Високорівневе представлення системи

Всі компоненти містять в собі абсолютно ідентичну та незалежну копію сервісів що займаються обробкою запитів.

3.1.3 Серверна частина

Серверна частина складається з 6 мікросервісів, бази даних та натренованих моделей. На рисунку 3.2 можна спостерігати її компоненти та те як вони взаємодіють один з одним.

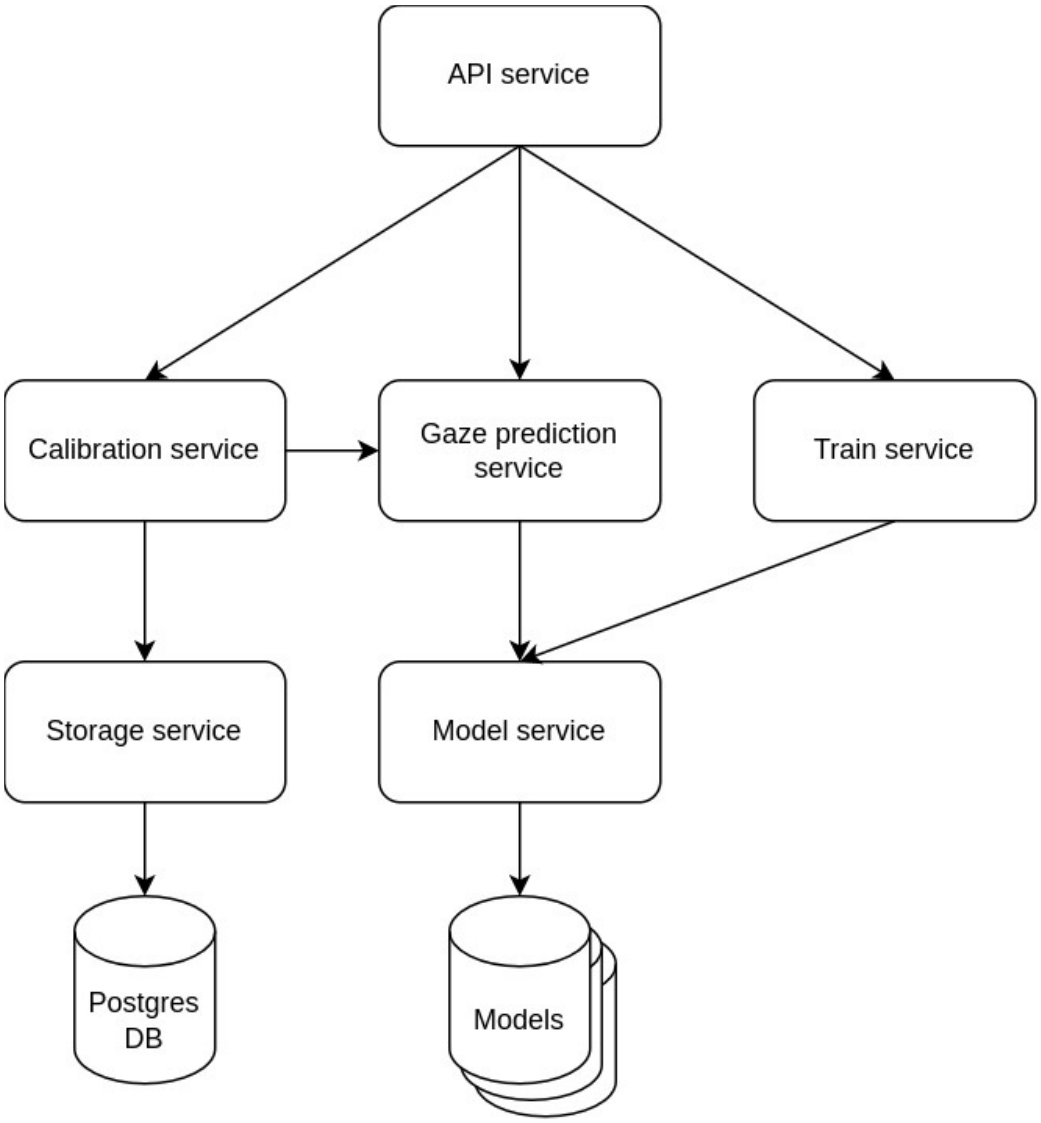

Рисунок 3.2 – Архітектура сервера

API service виступає в ролі вхідної точки. Він надає HTTP інтерфейс для взаємодії з сервером, тобто займається тим що збирає та компонує отримані дані з інших сервісів та повертає їх клієнту у JSON форматі.

Calibration service відповідальній за калібрування результатів передбачення моделі. Система використовує його при першому запуску

програми новою людиною або з нового пристрою. Вона враховує особливості погляду людини та параметрів дисплея. Погляд людей відрізняється один від одного, фізичні розміри дисплея також можуть бути різними та вони майже не мають кореляції з розширенням дисплея (реальний розмір пікселів відрізняється в залежності від матриці).

Gaze prediction service сервіс що безпосередньо займається передбачення координат погляду людини використовуючи отримане зображення з вебкамери. Цей сервіс запущений на фізичній машині з наявною відеокартою для досягнення максимальної швидкодії.

Train service дозволяє навчати нові моделі або донавчати вже існуючі. На вхід отримує розмічений дата сет із зображеннями людського обличчя, координатами їх погляду та іншою метаінформацією.

Storage service надає інтерфейс для отримання, запису та редагування інформації що знаходиться в базі даних.

Model service надає інтерфейс для отримання, додавання або зміни вже існуючих моделей створених для передбачення координат погляду людини із зображення.

Postgres DB екземпляр реляційної бази даних PostgreSQL, що зберігає налаштування застосунку для кожного користувача.

Models це сховище вже натренованих моделей. В якому можуть зберігатися різні моделі. Наприклад деякі можуть бути швидшими та менш точними, інші точнішими, але повільнішими. Також можна розділяти моделі під різні типи пристроїв, наприклад для ноутбуків, персональних комп'ютерів, мобільних пристроїв і т.д. Це також покращить точність роботи та досвід користування системою.

3.1.4 Клієнтська частина

Клієнтська частина складається з 5 модулів що разом займаються зчитуванням даних з камери, їх перетворенням в координати погляду, симуляцією роботи миші та відображенням результату на дисплеї.

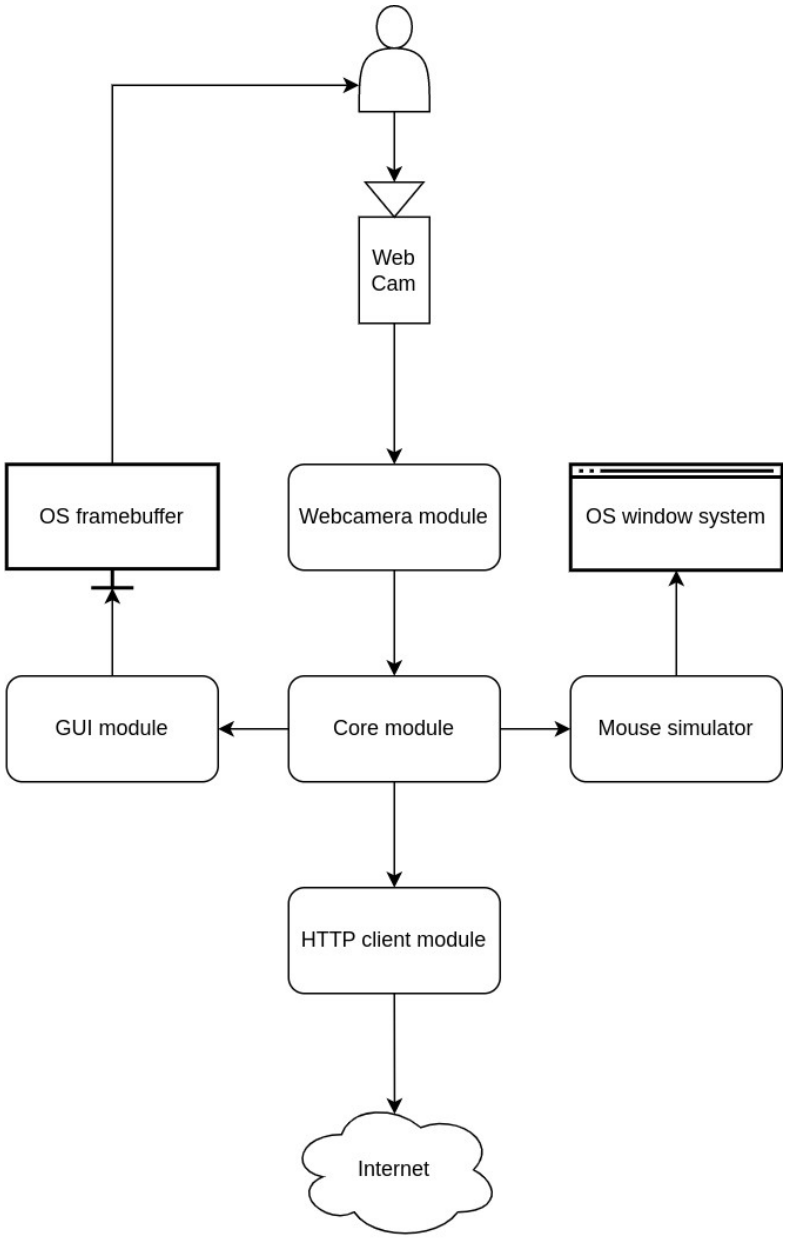

Рисунок 3.3 – Архітектура клієнту

На рисунку 3.3 можна спостерігати компоненти клієнту та процес їхньої взаємодії.

Core module основна частина програми, яка поєднує всі інші компоненти вона отримує зображення з камери, передає їх на обробку сервера, отримує результат та відображає його на дисплеї й симулює роботу миші.

GUI module взаємодіє з кадровим буфером операційної системи, що зберігає растрове зображення відеодисплея. Основним його завданням є відображення області дисплея на який зараз дивиться користувач. Це відбувається завдяки малюванню поверх всіх інших вікон що відкриті на пристрої.

Mouse module симулює роботу миші. А саме відповідає за її переміщення по дисплею та симуляції кліку правої та лівої кнопки.

HTTP client module реалізує взаємодію із серверною частиною системи. Готує необхідні дані для надсилання на сервер і отримуючи відповідь інтерпретує її необхідним чином.

3.2 Робота з системою

Перед початком використання системи потрібно щоб були виконані такі передумови:

- Обличчя видно в об’єктив вебкамери;
- У приміщенні якому знаходиться користувач хороше освітлення;
- За спиною користувача немає сильного світла;
- Обличчя користувача знаходиться по центру камери.

Для зручності користувача, програма надає можливість перевірити чи всі дотримані умови виконуються. Користувач може спостерігати вивід зображення зі своєї камери, необхідне положення голови, а також індикатор у верхньому правому куті який сигналізує про те чи всі необхідні умови дотримано.

На рисунках 3.4 та 3.5 можна спостерігати приклади успішного та невдалого розміщення обличчя перед об’єктивом камери відповідно. Також можна помітити напівпрозору маску обличчя, яка вказує де повинно бути обличчя користувача та індикатор зверху справа.

Рисунок 3.4 – Приклад невірного розміщення обличчя перед камерою

Рисунок 3.5 – Приклад вірного розміщення обличчя перед камерою

При першому запуску системи після дотримання передумови користувачу необхідно пройти першочергово калібрування. Для цього перед ним з’явиться білий екран із червоним колом за яким потрібно буде слідкувати протягом однієї хвилини. Коло буде рухатись по дисплею, приклад можна спостерігати на рисунках 3.6 і 3.7. Завдяки зіставленню розміщення кола на дисплеї та результатів передбачення моделі можна отримати необхідну інформацію про фізичні розміри екрана пристрою та особливості погляду людини.

Після завершення калібрування отримані результати передаються на сервер і зберігаються там. Тому при повторному запуску цю процедуру не потрібно буде проходити. Звичайно, якщо цим пристроєм буде

користуватись інша людина, або людина змінить пристрій то цей процес потрібно буде повторити.

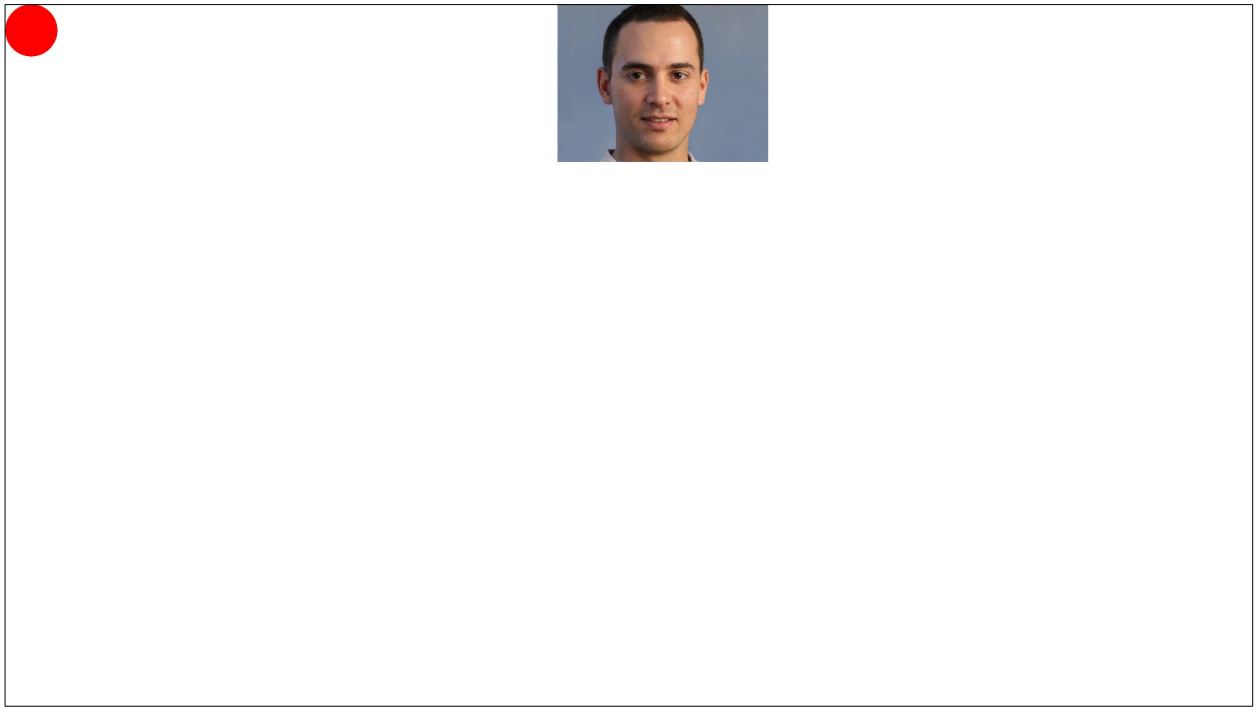

Рисунок 3.6 – Процес калібрування, коло у верхньому лівому куті

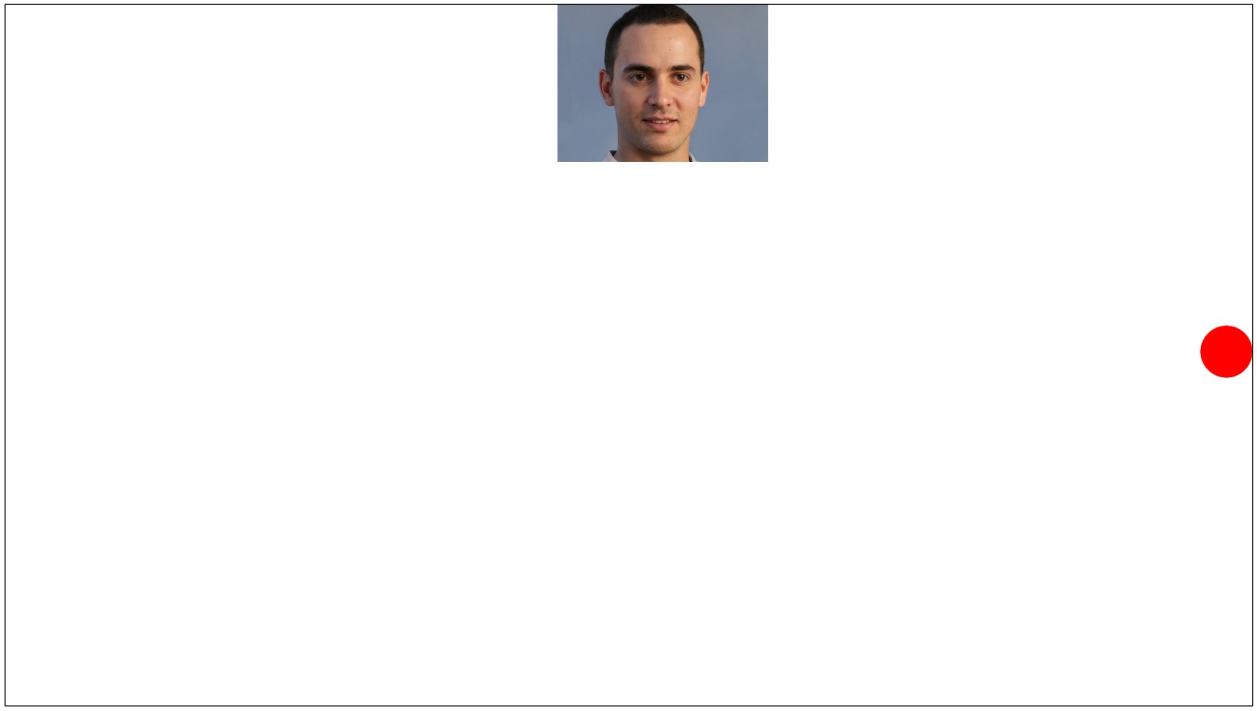

Рисунок 3.7 – Процес калібрування, коло посередині справа

Після проходження калібрування можна приступати до використання системи. Тепер користувач може керувати курсором своїм поглядом. Для того, щоб користувач розумів як розпізнається його погляд, він буде бачити коло червоного кольору на дисплеї яке вказує на розпізнане місце його погляду (рисунок 3.8).

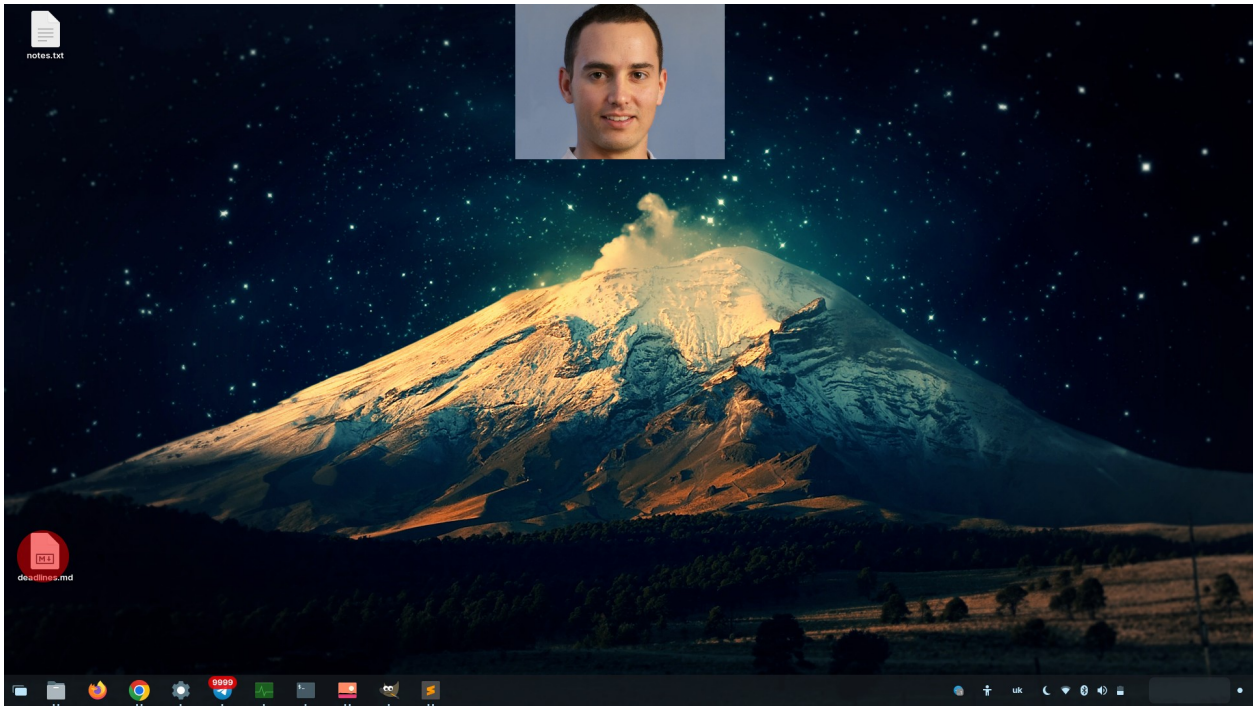

Рисунок 3.8 – Результат відстеження погляду

Для симуляції натискання кнопки миші, користувачу необхідно кліпнути відповідним оком. Око повинне бути закритим трішки довше ніж при звичайному морганні.

Для симуляції натискання правої кнопки миші потрібно кліпнути правим оком, а для симуляції натискання лівої кнопки миші потрібно кліпнути лівим оком.

3.3 Висновки до розділу

В даному розділі було описано особливості роботи з системою, розглянуто високорівневе представлення системи, а також детальну архітектуру клієнтської та серверної частини. Також описано порядок роботи та процес взаємодії користувача з системою.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЄКТУ

Останні роки довели, що стартапи сильніші, ніж більшість думала. Гнучкий та інноваційний підхід були ключовими, коли світ зіштовхнувся з такими проблемами, як віддалена робота, масштабні зміни в галузі та ринку, а також абсолютно нова реальність.

Здатність стартапів швидко змінюватися та адаптуватися і є ключовою властивістю даного виду бізнесу, не кажучи вже про те, що багато стартапів продовжували рости та розширювати свої команди.

Стартап — це бізнес-структура, що розвивається та працює на основі інновацій, створена для розв'язання проблеми шляхом надання нової пропозиції в умовах надзвичайної невизначеності.

Власне, стартап – це бізнес, який:

- швидко росте;
- порушує ринок або галузь (щось нове, що змушує конкурентів удосконалюватись);
- розв'язує проблему;
- працює в умовах надзвичайної невизначеності.

Багато підприємців і відомих бізнес-магнатів визначають стартап як культуру та менталітет побудови бізнесу на основі інноваційної ідеї для розв'язання критичних проблем.

Одне, що відрізняє стартапи від інших компаній — це зв’язок між їхнім продуктом та його попитом. Стартапи мають продукти, орієнтовані на невикористаний ринок. Підприємці-початківці знають ідеальну стратегію, щоб створити продукт, який хоче ринок, а також охопити й обслуговувати їх усіх. Це викликає швидке зростання [13].

4.1 Опис ідеї проєкту

В межах підпункту було проаналізовано і подано у вигляді таблиць:

- зміст ідеї;
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару;
- чим відрізняється від існуючих аналогів та замінників.

Перші три пункти подані у вигляді таблиці (таблиця 4.1) і дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів.

Таблиця 4.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Дана система дозволяє керувати комп’ютером за допомогою погляду.	Альтернативний спосіб керування комп’ютером	Можливість використання пристрою для користувачів з особливими потребами
	Резервний спосіб керування у разі виходу із ладу звичних пристроїв вводу	Збереження працездатності у випадках настання критичної ситуації

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї;

- визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проєкту та проєктів-конкурентів відповідно до визначеного вище переліку;
- проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають:
  - гірші значення (W, слабкі);
  - аналогічні (N, нейтральні) значення;
  - кращі значення (S, сильні) (табл. 4.2).

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Технікоекономічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W	N	S
		Мій проєкт	Skyle	Tobii Dynavox			
1.	Форма виконання	Надавання послуг	Надавання послуг, продаж пристрою	Надаванн я послуг, продаж пристрою			+
2.	Собівартість	Низька	Висока	Висока			+
3.	Функціонал	Широкий	Широкий	Широкий	+		

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит проєкту

В межах даного підрозділу було проведено аудит технології, за допомогою якої можна реалізувати ідею проєкту (технології створення товару). Визначення технологічної здійсненності ідеї проєкту передбачає аналіз таких складових (таблиця 4.3):

1. За якою технологією буде виготовлено товар згідно з ідеєю проєкту?
2. Чи існують такі технології, чи їх потрібно розробити/доробити?
3. Складність розробки (рангова оцінка від 1 до 3)?

Таблиця 4.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології реалізації	Наявність технологій	Складність розробки
1	Створення системи відстеження погляду в реальному часі	Використання мови програмування Python	Наявна	2
		Використання мови програмування C#	Відсутні	-
		Використання мови C++	Наявна	3
Обрана технологія реалізації ідеї проєкту: мова програмування Python.				

За результатами аналізу таблиці зроблено висновок щодо можливості технологічної реалізації проєкту. Технологічним шляхом реалізації проєкту було обрано такі технології, як Python через доступність, безплатність та складність розробки.

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проєкту, та ринкових загроз, які можуть перешкодити реалізації проєкту, дозволяє спланувати напрями розвитку проєкту з урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проєктів-конкурентів.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 4.4).

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум.од	10000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі, %	20%

За результатами аналізу таблиці 4.4 було зроблено висновок, що ринок є привабливим для входження.

Надалі були визначені потенційні групи клієнтів, їх характеристики та сформовано орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Швидкодія та точність роботи	Кінцевий користувач з особливими потребами	Повсякденне використання системи	Простота використання, точність роботи
Мінімальні фінансові витрати	Кінцевий користувач	Розглядають систему як резервний варіант	Швидкість налаштування, низька ціна

Після визначення потенційних груп клієнтів було проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проєкту, та факторів, що йому перешкоджають (табл. 4.6, 4.7).

Таблиця 4.6 – Фактори загроз

№ п/ п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок продуктів з кращими характеристиками	Передбачити додаткові переваги власного програмного продукту (ПП) для того, щоб повідомити про них саме після виходу на ринок конкурентів. Вдосконалення технічних моментів власного продукту. Обрати нову цільову аудиторію і зосередитися на ній: зниження цін.
2	Зміна потреб користувачів	Користувачам необхідний сервіс з більшим/новим функціоналом.	Розроблення гнучкої архітектури програмного забезпечення для легшого впровадження нового функціоналу

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Гнучкі ціни	Зменшення ціни товару задля збільшення попиту	Введення різних цін, в залежності від необхідного функціоналу
2	Поява новітніх методів розпізнавання погляду	З’являться нові методи, що будуть швидше та точніше розпізнавати напрям погляду	Замінити модель що використовується на нову, випустити нову версію продукту та оновити сервери

Надалі було проведено аналіз пропозиції: визначили загальні риси конкуренції на ринку (таблиця 4.8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції - чиста	Існує величезна кількість конкурентів на ринку.	Якісно провести рекламу.
2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти з інших країн	Створити архітектуру ПП таким чином, щоб можна було з легкістю локалізувати під різні країни
3. За галузевою ознакою - міжгалузева	Продукт може використовуватись для різних галузей	Підтримка продукту, впровадження нових можливостей та пришвидшення роботи
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між видами ПП, їх особливостями.	Створити ПП, враховуючи недоліки конкурентів

Продовження таблиці 4.8

5. За характером конкурентних переваг - нецінова	Вдосконалення технології створення ПП, щоб собівартість була нижчою	Удосконалення моделі. Використання дешевших технологій для розробки, ніж використовують конкуренти, але тільки якщо ці технології відповідають необхідним вимогам якості.
6. За інтенсивністю - не марочна	Бренд присутній, але його роль незначна	Реклама, участь у конференціях, семінарах.

Було проведено аналіз конкуренції у галузі за моделлю М. Портера (табл. 4.9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товариші замінники
	Навести перелік прямих конкурентів	Визначити бар’єри входження в ринок	Визначити фактори сили постачальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників

Продовження таблиці 4.9

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товариз амінники
	Skyle, Tobii Dynavox	Наявність вже існуючих рішень	-	Контроль якості продукту	Більша кількість функціонала, кращий дизайн, історія бренду.
Висновки:	Досить інтенсивна конкурентна боротьба з іншими гравцями	Є можливості виходу на ринок, але є і конкуренти.	-	Клієнти не повинні сумніватися у якості програмного продукту	Потрібно розробити ПЗ з приємним дизайном і широким функціоналом.

За результатами аналізу було зроблено висновок про можливість роботи на ринку з огляду на конкурентну ситуацію.

Цей висновок був врахований при формулюванні переліку факторів конкурентоспроможності у наступному пункті. На основі аналізу конкуренції, проведеного в таблиці, а також з урахуванням характеристик ідеї проєкту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів маркетингового середовища (табл. 4.6, 4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності.

Аналіз оформлено у (табл. 4.10).

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Ціна	Один із факторів для вибору продукту клієнтом.
2	Якість	Один із факторів для вибору продукту клієнтом.
3	Зручність роботи з програмою	Дозволяє користувачу легко працювати з програмою

За визначеними факторами конкурентоспроможності (табл. 4.10) проведено аналіз сильних та слабких сторін стартап-проєкту (табл. 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проєкту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	15					*		
2	Якість	10			*				
3	Зручність роботи з програмою	15					*		

Фінальним етапом ринкового аналізу можливостей впровадження проєкту є складання SWOT-аналізу (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 4.11).

Перелік ринкових загроз та ринкових можливостей було складено на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 4.12 – SWOT-аналіз стартап-проєкту

Сильні сторони: Ціна Простота використання Висока швидкодія	Слабкі сторони: Обмежений функціонал, менша точність в порівнянні з конкурентами.
Можливості: насичення ринку за рахунок мінімального цінового порогу входу, вдосконалення системи	Загрози: Конкуренція

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проєкти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів (таблиця 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	PR, просування бренду	50%	6 місяців
2	Перехід на безкоштовне розповсюдження	80%	3 місяців
3	Партнерство для об’єднання продукції	60%	2 місяці

Після аналізу було обрано альтернативу №2.

4.3 Аналіз ринкової стратегії проєкту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (таблиця 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Звичайний користувач	Середня: конкуренти, але з дорогим обладнанням	Середній	Сильна	Складно
2	Користувач з особливими потребами	Середня: велика конкуренція і можливість власних веб-відділів	Високий	Середня	Просто
Які цільові групи обрано:1,2					

За результатами аналізу потенційних груп споживачів було обрано цільову групу, для яких буде запропоновано даний товар, та визначено стратегію охоплення ринку - стратегію диференційованого маркетингу (компанія працює з декількома сегментами).

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Постійне оновлення і покращення продукту	Ринкове позиціювання на індивідуальних користувачів	Швидкодія, якість продукту, низька ціна, відсутність необхідності спеціальних пристроїв	Концентрований маркетинг

Наступним кроком обрано стратегію конкурентної поведінки (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки

Продовження таблиці 4.16

1	Ні.	Компанія буде шукати нових споживачів та забирати існуючих у конкурентів	Буде копіювати, удосконалювати та створювати свої унікальні пропозиції	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (табл. 4.5), а також в залежності від обраної базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (таблиця 4.16) розроблено стратегію позиціювання (таблиця 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проєкт.

Таблиця 4.17 – Визначення стратегії позиціювання

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Легкість налашту вання, інтуітивна взаємодія, точність та надійність ПП.	Стратегія диференціації	Позиція на основі порівняння з товарами конкурентів; Відмінні особливості споживача	Швидкодія; Економія фінансів; Зручність застосування

Результатом виконання підрозділу стала узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап- компанії на ринку.

4.4 Розроблення маркетингової програми стартап-проєкту

Сформовано маркетингову концепцію товару, який отримає споживач. Для цього підсумовано результати попереднього аналізу конкурентоспроможності товару (таблиця 4.18). Концепція товару – письмовий опис фізичних та інших характеристик товару, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/ п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидкодія	Система працює в реальному часі	Можливість використання віддалених обчислювальних ресурсів.
2	Дешевизна	Нема потреби у датчиків які дорого коштують.	Для роботи необхідна лише вебкамера.

Продовження таблиці 4.18

3	Простота налаштування	Для налаштування потрібно лише пройти короткий процес калібрування.	Для початку роботи потрібно інсталювати ПП та під'єднати вебкамеру. Нема потреби замовляти додаткову периферію.

Розроблено трирівневу маркетингову модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 4.19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби й / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов’язане з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ містить в собі якість, властивості, дизайн, пакування, ціну.

3-й рівень Товар з підкріпленням (супроводом) – додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості, доставлення, умови оплати та ін.).

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Розпізнавання напряму погляду на дисплеї та емуляція роботи миші.

Продовження таблиці 4.18

II. Товар реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	1. Індивідуальний підхід. 2. Низька ціна. 3. Простота у використанні.	1.Нм 2.Нм 3.Нм	1.Технологічна 2.Економічна 3.Технологічна
	Якість: точність розпізнавання погляду на дисплеї		
	Пакування: відсутнє		
	Марка: ROSO		

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проєкту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (таблиця 4.20). Аналіз проведено експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	1250$	2500$	У всіх трьох груп високий рівень доходів	500$--

Наступним кроком є визначення оптимальної системи збуту, в межах якого було прийняте рішення (таблиця 4.21).

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Канал нульового рівня	Продаж	0(напряму)	Власна

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціювання, визначену специфіку поведінки клієнтів (таблиця 4.22).

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/ п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціювання	Завдання рекламного повідомлення	Концепція рекламного звернення
	Встановлення на ПП на персональний комп’ютер	Інтернет	Низька ціна, простота використання, швидкий старт роботи	Показати переваги рішення над конкурентами, виділити ключові особливості	Створення сайту продукту, розповсюдження інформації про продукт на спеціалізованих ресурсах.

Було визначено, що придбання продукту буде проводитись через мережу Інтернет або при безпосередньому спілкуванні із представниками компанії. Розповсюдження інформації про продукт буде проводитись виключно через Інтернет, адже аудиторія даного продукту активно користується всесвітньою мережею.

Результатом підрозділу стала ринкова (маркетингова) програма, що містить в собі концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого впроваджено проєкт, та відповідну обрану альтернативу ринкової поведінки.

4.5 Висновки до розділу

В даному розділі проведено підготовчий аналіз для впровадження розробленої системи в якості стартап проєкту. Досліджено аналогічні конкурентні системи, встановлено сильні та слабкі сторони системи в порівнянні з ними. Також було досліджено можливі шляхи розповсюдження продукту та його ймовірну аудиторію, рівень доходів та ймовірну ціну продукту, що розробляється.

Було проведено аналіз потенційних ризиків і можливостей, а також розраховані основні фінансово-економічні показники проєкту. Отримані результати кажуть про те, що реалізація проєкту є доцільною. Було визначено сильні сторони проєкту: зручність у використанні, ціна, швидкодія та легкість налаштування. Серед слабких варто виділити гіршу точність у порівнянні з конкурентами, що використовують коштовні датчики. Варто зазначити можливість реклами продукту на спеціалізованих ресурсах із зазначенням сильних сторін проєкту.

ВИСНОВКИ

У цій дисертації був проведений аналіз уже існуючих рішень та пошук можливих варіантів вирішення поставленого завдання.

Також в ході роботи було досліджено і встановлено інструменти прогнозування погляду користувача на основі зображення обличчя. Після цього було підібрано архітектуру нейронної мережі яка достатньо точно передбачає необхідну величину, а саме координати погляду відносно камери.

Тобто запропоновано модифіковану модель глибокої нейронної мережі для відстеження погляду і взаємодії з комп’ютером в реальному часі на основі згорткових шарів із додаванням механізму уваги. Розроблено систему відстеження погляду у реальному часі з використанням фреймворку Torch Python.

В результаті порівняльного аналізу встановлено, що розроблена система за показником MAE (значення 0.98 см і 2.17 см для координат x і y), не поступається перед наявною системою iTracker, у якої MAE сягають для координат x та y значення 1.04 см та 2.31 см, відповідно.

Наступним кроком був процес проєктування архітектури системи з урахуванням висунутих вимог та планування принципу взаємодії користувача з системою.

Це було важливим пунктом, оскільки не всі потенційні користувачі можуть мати достатньо обчислювальних ресурсів для виконання обробки зображень локально. Саме тому було обрано мікросервісний стиль. Та спроєктовано систему яка може розгортатись як в хмарі, так і локально на персональному пристрої користувача, при необхідності.

ПЕРЕЛІК ПОСИЛАНЬ

1. About - OpenCV [Електронний ресурс] – Режим доступу до ресурсу: https://opencv.org/about/.

2. About PyQt [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.python.org/moin/PyQt.

3. Appearance-based gaze estimation in the wild. [Електронний ресурс] / X.Zhang, Y. Sugano, M. Fritz, A. Bulling. – 2015. – Режим доступу до ресурсу: https://openaccess.thecvf.com/content_cvpr_2015/papers/Zhang_Appearance-Based_Gaze_Estimation_2015_CVPR_paper.pdf.

4. Aurelien Geron. Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly Media Inc., Sebastopol, CA, 2017.

5. Eye Tracking for Everyone [Електронний ресурс] / [K. Krafka, A. Khosla, P. Kellnhofer та ін.]. – 2016. – Режим доступу до ресурсу: https://gazecapture.csail.mit.edu/.

6. Face Detection using Haar Cascades [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html.

7. Fairclough S. Eye Tracking and Eye-Based Human–Computer Interaction / S. Fairclough, K. Gilleade. – London, 2014. – (Springer).

8. Huang Q. TabletGaze: Unconstrained Appearance-based Gaze Estimation in Mobile Tablets [Електронний ресурс] / Q. Huang, A. Veeraraghavan, A. Sabharwal. – 2015. – Режим доступу до ресурсу: https://arxiv.org/abs/1508.01244.

9. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. The MIT Press Cambridge, Massachusetts London, England, 2017.

10. Krizhevsky A. ImageNet classification with deep convolutional neural networks [Електронний ресурс] / A. Krizhevsky, I. Sutskever, G. Hinton. – 2017. – Режим доступу до ресурсу: https://dl.acm.org/doi/abs/10.1145/3065386.

11. Learn Python PyQt [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://pythonpyqt.com/.

12. Open Source Computer Vision [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/4.6.0/.

13. Pahwa A. What Is A Startup? [Електронний ресурс] / Aashish Pahwa. – 2022. – Режим доступу до ресурсу: https://www.feedough.com/what-is-startup/.

14. Papageorgiou C. Sixth International Conference on Computer Vision / C. Papageorgiou, M. Oren, T. Poggio // A general framework for object detection / C. Papageorgiou, M. Oren, T. Poggio., 1998. – С. 555–562.

15. PyTorch Foundation [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/foundation.

16. Subject independent facial expression recognition with robust face detection using a convolutional neural network [Електронний ресурс] / M.Matsugu, K. Mori, Y. Mitari, Y. Kaneda. – 2003. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0893608003001151.

17. Viola P. Computer Society Conference on Computer Vision and Pattern Recognition / P. Viola, M. Jones // Rapid object detection using a boosted cascade of simple features / P. Viola, M. Jones., 2001.

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ

1. Файл web_camera.py

```
import cv2


class CameraNotFoundError(Exception):

    def __init__(self, message=None):
        if message is None:
            message = 'Camera not found, please check camera connection.'

        super().__init__(message)


class Camera:

    MAX_CAMERA_INDEX = 100
    MASK_PATH = '../blank_face1.png'
    MASK_SCALE_FACTOR = 0.6

    GREEN = (0, 255, 0)
    RED = (0, 0, 255)
    BLUE = (255, 0, 0)

    def __init__(self):
        cv2.namedWindow('Camera')
        self.capture = self.get_available_video_capture()
        self.capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
```

```
# self.capture.set(cv2.CAP_PROP_FRAME_WIDTH, 1280 // 2)
self.capture.set(cv2.CAP_PROP_FRAME_WIDTH, 720)

self.mask = self.scale_image(
        cv2.imread(self.MASK_PATH),
        self.MASK_SCALE_FACTOR
    )

self.frame_h, self.frame_w, _ = self.get_frame().shape
self.mask_h, self.mask_w, _ = self.mask.shape
print('Frame shape:', self.frame_h, self.frame_w)

# center x and y
self.c_y = int(self.frame_h/2)
self.c_x = int(self.frame_w/2)
print('Center:', self.c_y, self.c_x)

# mask boundaries
self.mask_t_y = self.c_y - int(self.mask_h/2)
self.mask_b_y = self.mask_t_y + self.mask_h
print('Mask Y bounds:', self.mask_t_y, self.mask_b_y)
assert self.mask_b_y > self.mask_t_y
self.mask_l_x = self.c_x - int(self.mask_w/2)
self.mask_r_x = self.mask_l_x + self.mask_w
print('Mask X bounds:', self.mask_l_x, self.mask_r_x)

assert self.mask_r_x > self.mask_l_x

self.frame = None
```

```
@classmethod
def scale_image(cls, img, scale_factor):
    width = int(img.shape[1] * scale_factor)
    height = int(img.shape[0] * scale_factor)

    return cv2.resize(img, (width, height), interpolation=cv2.INTER_AREA)

def get_available_video_capture(self):

    for camera_index in range(0, self.MAX_CAMERA_INDEX):
        capture = cv2.VideoCapture(camera_index)

        if not capture.read()[0]:
            continue

        return capture

    raise CameraNotFoundError()

def get_frame(self):
    self.frame = self.capture.read()[1]
    return self.frame

def show_frame(self):
    cv2.imshow('Camera', self.frame)

def get_mask_destination(self):
    return self.frame[self.mask_t_y:self.mask_b_y, self.mask_l_x:self.mask_r_x]
```

```
    def get_frame_by_rect(self, rect):
        return self.frame[rect[0][1]:rect[1][1], rect[0][0]:rect[1][0]]

    def add_mask(self):
        destination = self.get_mask_destination()
        result = cv2.addWeighted(destination, 1, self.mask, 0.5, 0)
            self.frame[self.mask_t_y:self.mask_b_y,  self.mask_l_x:self.mask_r_x]  =
result

    def draw_indicator(self, color, radius=30, margin=30):
        center = (self.frame_w - radius//2 - margin, radius//2 + margin)
        self.frame = cv2.circle(self.frame, center, radius, color, -1)

    def rectangle_relative_to_mask(self, recognition_result):
        x, y, w, h = recognition_result

        t_l = (self.mask_l_x + x, self.mask_t_y + y)
        b_r = (self.mask_l_x + x + w, self.mask_t_y + y + h)

        return t_l, b_r

    def rectangle_relative_to_face(self, recognition_result, face):
        rect = self.rectangle_relative_to_mask(face)

        x, y, w, h = recognition_result
        t_l = (rect[0][0] + x, rect[0][1] + y)
        b_r = (rect[0][0] + x + w, rect[0][1] + y + h)
```

```
        return t_l, b_r

    def draw_rectangle(self, t_l, b_r, color, thickness=2):
        self.frame = cv2.rectangle(self.frame, t_l, b_r, color, thickness)
```

2. Файл gui.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import Qt, QPoint, QTimer
from PyQt5.QtWidgets import QMainWindow, QApplication, QDesktopWidget, QWidget, QLabel, QGridLayout
from PyQt5.QtGui import QPainter, QPen, QBrush, QColor

from core import Core
from calibrator import Calibrator

import json
import random


class AnalyzerThread(QtCore.QThread):

    result_done = QtCore.pyqtSignal(object, object)

    def __init__(self):
        QtCore.QThread.__init__(self)
        self.core = Core()

    def run(self):
        while True:
```

```
            frame, prediction = self.core.analyze_frame()
            self.result_done.emit(frame, prediction)


class CalibrationWindow(QMainWindow):

    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setObjectName('Calibration')
        self.geometry = QtWidgets.QDesktopWidget().screenGeometry(0).size()
        self.resize(self.geometry)

        self.centralwidget = QtWidgets.QWidget(self)

        self.image_label = QLabel(self.centralwidget)
        self.image_label.setGeometry(
            QtCore.QRect(1920//2 - 160, 0, 320, 320))



        self.circle_label = QLabel(self.centralwidget)
        self.circle_label.resize(40, 40)
        self.move_circle(0,0)
        self.circle_label.setAlignment(QtCore.Qt.AlignCenter)
        self.circle_label.setStyleSheet(
            "background-color: red; border-radius: 20px;")
        self.circle_label.setHidden(True)
```

```
        self.setAttribute(Qt.WA_TranslucentBackground)
        self.setAttribute(Qt.WA_TransparentForMouseEvents)
        self.setAttribute(Qt.WA_X11DoNotAcceptFocus)
        self.showFullScreen()

        self.analyzer = AnalyzerThread()
        self.analyzer.result_done.connect(self.on_analyzer_result)
        self.analyzer.start()

        self.setCentralWidget(self.centralwidget)

        self.calib_size = (7, 5)
        self.calib_gen = self.calibration_index_gen()
        self.timer = QTimer(self)
        self.timer.setSingleShot(True)
        self.timer.timeout.connect(self.next_calibration_point)
        self.timer.setInterval(10000)
        self.timer.start()

        self.last_x, self.last_y = None, None

        self.calib_res = list()
        self.calibrator = None

        self.predictions = list()

    def move_circle(self, x, y):
        self.circle_label.move(
```

```
            x - self.circle_label.size().width()//2,
            y - self.circle_label.size().height()//2)

    def calibration_index_gen(self):
        for j in range(self.calib_size[1]):
            for i in range(self.calib_size[0]):
                yield i, j

    def next_calibration_point(self):
        self.circle_label.setHidden(False)
        try:
            i, j = next(self.calib_gen)
        except StopIteration:
            self.circle_label.setHidden(True)
            filename = f'calibration#{random.randint(100, 999)}.json'
            with open(filename, 'w') as f:
                json.dump(self.calib_res, f)

            self.calib_res.clear()
            print(f'Saved as {filename}')

            self.calibrator = Calibrator(f'./{filename}')
            self.calibrator.fit()
            return

        x = i * (self.geometry.width() // (self.calib_size[0] - 1))
        y = j * (self.geometry.height() // (self.calib_size[1] - 1))

        self.last_x, self.last_y = x, y
```

```
        self.move_circle(x, y)
        self.timer.setInterval(3000)
        self.timer.start()

    def on_analyzer_result(self, frame, prediction):
        if self.timer.isActive() and prediction is not None and self.last_x is not None:

            if 500 <= self.timer.remainingTime() <= 2500:
                self.calib_res.append(
                    (self.last_x, self.last_y, prediction[0], prediction[1]))

        if prediction is not None and self.calibrator:
            self.predictions.append(prediction)

            xs, ys = self.calibrator.predict_single(*prediction)
            print(int(xs), int(ys))

            self.circle_label.setHidden(False)
            self.move_circle(int(xs), int(ys))



        rgb_frame = QtGui.QImage(
            frame.data,
            frame.shape[1],
            frame.shape[0],
            QtGui.QImage.Format_RGB888).rgbSwapped()
         rgb_frame = rgb_frame.scaled(320, 320, Qt.KeepAspectRatio) # Only if you
want to scale image
```

```
        self.image_label.setPixmap(QtGui.QPixmap.fromImage(rgb_frame))


if __name__ == '__main__':
    import sys
    app = QtWidgets.QApplication(sys.argv)
    ui = CalibrationWindow()

    try:
        app.exec_()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemExit:
            os._exit(0)
```

3. Файл calibrator.py

```
import numpy as np
import pandas as pd
import sklearn

import matplotlib.pyplot as plt

from sklearn.preprocessing import QuantileTransformer
from sklearn.model_selection import train_test_split

import lightgbm as lgbm
```

```
from sklearn.metrics import r2_score
from sklearn.multioutput import MultiOutputRegressor
from sklearn.compose import TransformedTargetRegressor

from sklearn.linear_model import ElasticNet


class Calibrator:

    def __init__(self, calibration_data_path: str):

        self.data = pd.read_json(calibration_data_path)
        self.data.columns = ['rx', 'ry', 'px', 'py']

        self.clean_data = pd.DataFrame()
        coord = self.data[['rx', 'ry']].drop_duplicates()
        for x, y in zip(coord.rx.values, coord.ry.values):

            self.clean_data = pd.concat([self.clean_data,
                                         self.get_clean_data(x, y, self.data)])

        print(f'Origin data: {self.data.shape}, Cleaned data: {self.clean_data.shape}')
        self.regressor = None


    def get_clean_data(self, x: int, y: int, data: pd.DataFrame):

        cdata = data.loc[(data.rx == x) & (data.ry == y), :]
        xmean, xstd = np.mean(cdata.px), np.std(cdata.px)
```

```
    ymean, ystd = np.mean(cdata.py), np.std(cdata.py)

    up_x, bottom_x = xmean + 2 * xstd, xmean - 2 * xstd
    up_y, bottom_y = ymean + 2 * ystd, ymean - 2 * ystd

    cdata = cdata.loc[cdata.px.between(bottom_x, up_x) &\
                    cdata.py.between(bottom_y, up_y), :]

    return cdata

def split_dataset(self, data: pd.DataFrame):

    X, y = data.drop(columns = ['rx', 'ry']), data.loc[:, ['rx', 'ry']]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                        random_state=555, stratify=data.rx)

    return X_train, X_test, y_train, y_test


def train_model(self, X_train, X_test, y_train, y_test):

    target_transformer = QuantileTransformer(output_distribution = 'normal')
    transformed_target = target_transformer.fit_transform(y_train)

    wrapper = lgbm.LGBMRegressor()
    wrapper = MultiOutputRegressor(wrapper)
    self.regressor = wrapper

    self.regressor.fit(X_train, y_train)
```

```
        y_pred = self.regressor.predict(X_test)
        score = r2_score(y_test.values, y_pred, multioutput = 'raw_values')
        print('Valid R2 score: ', score)


    def fit(self):

        X_train, X_test, y_train, y_test = self.split_dataset(self.clean_data)
        self.train_model(X_train, X_test, y_train, y_test)


    def predict(self, calibration_data_path: str):

        data = pd.read_json(calibration_data_path)
        data.columns = ['rx', 'ry', 'px', 'py']

        clean_data = pd.DataFrame()
        coord = self.data[['rx', 'ry']].drop_duplicates()
        for x, y in zip(coord.rx.values, coord.ry.values):

            clean_data = pd.concat([self.clean_data,
                                    self.get_clean_data(x, y, self.data)])

        print(f'Origin data: {self.data.shape}, Cleaned data: {self.clean_data.shape}')
        print(clean_data[['px', 'py']].shape)
        ypred = self.regressor.predict(clean_data[['px', 'py']])
            score = r2_score(clean_data[['rx', 'ry']].values, ypred, multioutput =
'raw_values')
        print('Valid R2 score: ', score)
```

```
    def predict_single(self, x_cnn, y_cnn):
        return self.regressor.predict(((x_cnn, y_cnn),))[0]
```

4. Файл gaze.py

```
import cv2
import numpy as np
import torch
import torch.nn as nn
import torchvision.transforms as T


class Gaze:

    HAAR_FACE_PATH = '../haarcascade_frontalface_default.xml'
    HAAR_EYE_PATH =  '../haarcascade_eye_tree_eyeglasses.xml'
    WEIGHTS = '../epoch'
    DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'
    TRANSFORM = T.Compose([
        T.ToTensor(),
        T.Resize([224, 224]),
        T.ConvertImageDtype(torch.float32)
    ])

    def __init__(self):
        self.face_cascade = cv2.CascadeClassifier(self.HAAR_FACE_PATH)
        self.eye_cascade = cv2.CascadeClassifier(self.HAAR_EYE_PATH)
        self.model = EyeTracking().to(self.DEVICE)
        self.model.load_state_dict(torch.load(self.WEIGHTS))
```

```
    self.model.eval()

def to_gray(self, frame):
    return cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

def get_face(self, frame):
    frame = self.to_gray(frame)
    faces = self.face_cascade.detectMultiScale(frame)

    if len(faces) > 0:
        return faces[0]

    return None

def get_eyes(self, frame):
    frame = self.to_gray(frame)
    eyes = self.eye_cascade.detectMultiScale(
            frame,
            scaleFactor=1.1,
            minNeighbors=5,
            minSize=(30, 30)
        )

    res = {
        'left': None,
        'right': None
    }

    center_x = int(frame.shape[1]/2)
```

```
    for eye in eyes:
        x, y, w, h = eye
        eye_center_x = x + w/2

        if eye_center_x > center_x and res['left'] is None:
            res['left'] = eye
        elif eye_center_x < center_x and res['right'] is None:
            res['right'] = eye

    return res

@classmethod
def transform(cls, obj):
    return cls.TRANSFORM(obj)

@classmethod
def make_grid_mask(cls, frame_shape, face_rect, size=(25, 25)):
    mask = np.zeros(size)

    xd = frame_shape[1] / size[0]
    yd = frame_shape[0] / size[1]
    for i in range(size[0]):
        x = xd*i + xd/2
        for j in range(size[1]):
            y = yd*j + yd/2

            if face_rect[0][0] < x < face_rect[1][0] and \
                    face_rect[0][1] < y < face_rect[1][1]:
```

```
                mask[i][j] = 1

        return mask.astype(np.int8)

    def predict_gaze(self, frame, face, leye, reye, face_rect):
        facer = self.transform(face.copy())[None,:].to(self.DEVICE)
        leyer = self.transform(leye.copy())[None,:].to(self.DEVICE)
        reyer = self.transform(reye.copy())[None,:].to(self.DEVICE)

        grid = self.make_grid_mask(frame.shape, face_rect)
        grid = torch.from_numpy(grid).to(torch.float32)[None,:].to(self.DEVICE)

        ypred = self.model(facer, leyer, reyer, grid)

        return float(ypred[0][0]), float(ypred[0][1])


class AttentionModule(nn.Module):

    def __init__(self, num_channels, img_size):
        super().__init__()

        self.img_size = img_size
        self.num_channels = num_channels

        self.lnorm = nn.LayerNorm([num_channels])
        self.mhead = nn.MultiheadAttention(num_channels, 4, batch_first=True)

        self.fc = nn.Sequential(
```

```
            nn.LayerNorm([num_channels]),
            nn.Linear(num_channels, num_channels),
            nn.GELU(),
            nn.Linear(num_channels, num_channels),
        )

    def forward(self, x):
        num_samples = x.shape[0]
        x = x.view(num_samples, self.num_channels, -1).swapaxes(1, 2)
        out_norm = self.lnorm(x)
        attention_value, _ = self.mhead(out_norm, out_norm, out_norm)
        attention_value = attention_value + x

        attention_value = self.fc(attention_value) + attention_value
        return attention_value. \
                swapaxes(2, 1).\
                view(
                    num_samples,
                    self.num_channels,
                    self.img_size,
                    self.img_size)


class ImageModule(nn.Module):

    def __init__(self):
        super().__init__()

        self.conv_1 = nn.Sequential(
```

```
            nn.Conv2d(3, 96, kernel_size=11, stride=4, padding=0),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.CrossMapLRN2d(size=5, alpha=0.0001, beta=0.75, k=1.0)
        )

        self.conv_2 = nn.Sequential(
            nn.Conv2d(96, 256, kernel_size=5, stride=1, padding=2, groups=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.CrossMapLRN2d(size=5, alpha=0.0001, beta=0.75, k=1.0)
        )

        self.conv_3 = nn.Sequential(
            nn.Conv2d(256, 384, kernel_size=3, stride=1, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 64, kernel_size=1, stride=1, padding=0),
            nn.ReLU(inplace=True)
        )

        self.atten_1 = AttentionModule(96 , 26)
        self.atten_2 = AttentionModule(256, 12)
        self.atten_3 = AttentionModule(64 , 12)

    def forward(self, x: torch.Tensor):
        x = self.atten_1(self.conv_1(x))
        x = self.atten_2(self.conv_2(x))
        x = self.atten_3(self.conv_3(x))
        return x.reshape(x.shape[0], -1)
```

```
class FaceModule(nn.Module):

    def __init__(self):
        super().__init__()

        self.conv = ImageModule()
        self.fc = nn.Sequential(
            nn.Linear(12 *12*64, 128),
            nn.ReLU(inplace=True),
            nn.Linear(128, 64),
            nn.ReLU(inplace=True),
        )

    def forward(self, x):
        x = self.conv(x)
        return self.fc(x)


class GridModule(nn.Module):

    def __init__(self, grid_size=25):
        super().__init__()

        self.fc = nn.Sequential(
            nn.Linear(grid_size * grid_size, 256),
            nn.ReLU(inplace=True),
            nn.Linear(256, 128),
```

```
            nn.ReLU(inplace=True),
        )

    def forward(self, x):
        x = x.view(x.size(0), -1)
        return self.fc(x)


class EyeTracking(nn.Module):

    def __init__(self):
        super().__init__()
        self.eyes_module = ImageModule()
        self.face_module = FaceModule()
        self.grid_module = GridModule()

        self.eyes_fc = nn.Sequential(
            nn.Linear(2 * 12*12*64, 128),
            nn.ReLU(inplace=True)
        )

        self.fc = nn.Sequential(
            nn.Linear(128+64+128, 128),
            nn.ReLU(inplace=True),
            nn.Linear(128, 2)
        )

    def forward(self, face, eye_left, eye_right, grid):
```

```
        # Eye nets
        x_leye = self.eyes_module(eye_left )
        x_reye = self.eyes_module(eye_right)

        # Cat and FC
        x_eyes = torch.cat((x_leye, x_reye), 1)
        x_eyes = self.eyes_fc(x_eyes)

        # Face net
        x_face = self.face_module(face)
        x_grid = self.grid_module(grid)

        # Cat all
        x = torch.cat((x_eyes, x_face, x_grid), 1)
        return self.fc(x)
```

5. Файл core.py

```
from webcam import Camera
from gaze import Gaze
from calibrator import Calibrator
import cv2
import pickle
import random
import copy
from time import time
import numpy as np
```

```
class Core:

    camera = Camera()
    gaze = Gaze()

    def __init__(self):
        self.prev_eyes_set = set()
        self.predictions = list()
        self.start = time()

    def analyze_frame(self):
        curr_eyes_set = set()
        frame = self.camera.get_frame()
        frame_to_save = copy.deepcopy(frame)
        face_frame = None
        reye_frame = None
        leye_frame = None

        face = self.gaze.get_face(self.camera.get_mask_destination())
        if face is not None:

            face_rect = self.camera.rectangle_relative_to_mask(face)
            face_frame = copy.deepcopy(self.camera.get_frame_by_rect(face_rect))
                                    self.camera.draw_rectangle(face_rect[0],    face_rect[1],
self.camera.GREEN)

            eyes = self.gaze.get_eyes(face_frame)
```

```
            for eye_name, eye in eyes.items():
                if eye is not None:
                    curr_eyes_set.add(eye_name)

                    eye_rect = self.camera.rectangle_relative_to_face(eye, face)
                    eye_frame = self.camera.get_frame_by_rect(eye_rect)
                                        self.camera.draw_rectangle(eye_rect[0],  eye_rect[1],
self.camera.BLUE)

                    if eye_name == 'left':
                        leye_frame = eye_frame
                    else:
                        reye_frame = eye_frame

            eyes_diff = self.prev_eyes_set.difference(curr_eyes_set)
            if len(eyes_diff) == 2:
                print('Boths eyes blinked')
            elif len(eyes_diff) == 1:
                print(f'{eyes_diff.pop().capitalize()} eyes blinked')

            self.camera.draw_indicator(self.camera.GREEN)
        else:
            self.camera.draw_indicator(self.camera.RED)

        self.prev_eyes_set = curr_eyes_set
        self.camera.add_mask()
        self.camera.show_frame()
```

```
        key = cv2.waitKey(20)
        if key == 27: # exit on ESC
            exit(0)

        prediction = None
        if frame_to_save is not None and \
                face_frame is not None and \
                reye_frame is not None and \
                leye_frame is not None and \
                face_rect is not None:
                        prediction = self.gaze.predict_gaze(frame_to_save, face_frame,
reye_frame, leye_frame, face_rect)
            self.predictions.append(prediction)

            print(
                np.round(
                    np.asarray(self.predictions[-5:])[:,0].std(), 3),
                np.round(
                    np.asarray(self.predictions[-5:])[:,1].std(), 3))

        elif key == 32: # save on space
            print(type(frame_to_save))
            print(type(face_frame))
            print(type(reye_frame))
            print(type(leye_frame))

            series = random.randint(100, 999)
            prefix = './imgs/' + str(series)
            with open(prefix+'frame.pickle', 'wb') as f:
```

```
            pickle.dump(frame_to_save, f)

        with open(prefix+'face_frame.pickle', 'wb') as f:
            pickle.dump(face_frame, f)

        with open(prefix+'reye_frame.pickle', 'wb') as f:
            pickle.dump(reye_frame, f)

        with open(prefix+'leye_frame.pickle', 'wb') as f:
            pickle.dump(leye_frame, f)

        with open(prefix+'meta.pickle', 'wb') as f:
            meta = {
                'face_rect': face_rect
            }
            print(meta)
            pickle.dump(meta, f)

        print(f'Saved as SERIES: {series}')
    # predict on TAB
    elif key == 9 and \
            frame_to_save is not None and \
            face_frame is not None and \
            reye_frame is not None and \
            leye_frame is not None and \
            face_rect is not None:
        from time import time
        s = time()
```

```
                res = self.gaze.predict_gaze(frame_to_save, face_frame, reye_frame,
leye_frame, face_rect)
            elapsed = time() - s
            print(f'Prediction is: {res}, elapsed time {elapsed}')
        return self.camera.frame, prediction


if __name__ == '__main__':
    core = Core()
    while True:
        core.analyze_frame()[1]
```