НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
(повне найменування вищого навчального закладу)

ІНЖЕНЕРНО - ХІМІЧНИЙ ФАКУЛЬТЕТ
(повне найменування інституту, назва факультету (відділення))

Кафедра автоматизації хімічних виробництв
(повна назва кафедри (предметної, циклової комісії))

«До захисту допущено»
Завідувач кафедри
_____ А.І.Жученко
«___» _____20_ р

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: Автоматизація контролю та управління доступом до виробничих приміщень в умовах санітарних обмежень

Виконав: студент 4 курсу, групи ЛА-72
(шифр групи)

Гайдук Кирило Германович _____
(прізвище, ім'я, по-батькові) (підпис)

Керівник *к. т. н ., доцент Сазонов А.Ю.* _____
(посада, науковий ступінь, вчене звання ,прізвище та ініціали) (підпис)

Консультант Охорона праці *асистент Ковтун А.І.* _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *к.т.н, доцент Бунке О. С.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що в цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань
Студент _____
(підпис)

Київ - 2021 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інженерно-хімічний факультет
(повна назва)

Кафедра автоматизації хімічних виробництв
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 151 Автоматизація та комп'ютерно-інтегровані технології
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Жученко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проект студенту

Гайдуку Кирилові Германовичу
(прізвище, ім'я, по батькові)

1. Тема проекту *Автоматизація контролю та управління доступом до виробничих приміщень в умовах санітарних обмежень*
_____
_____ ,
керівник проекту _____*к. т. н., доцент Сазонов А. Ю.*_____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «26» квітня 2021 р. № 1071

2. Термін подання студентом проекту 07.06.2020

3. Вихідні дані до проекту *фото та відеоматеріали, перелік санітарних обмежень для доступу до приміщень*

4. Зміст пояснювальної записки *аналіз існуючих автоматизованих систем контролю та управління доступом; аналіз методів детектування обличчя та розпізнавання особистості; алгоритмічне забезпечення системи; програмне забезпечення системи; охорона праці*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) *Блок-схема роботи алгоритму Telegram боту; блок-схема роботи алгоритму головної програми*

6. Консультанти розділів проекту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	асистент Ковтун А.І.		

7. Дата видачі завдання 10.04.2020

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Аналіз існуючих автоматизованих систем контролю та управління доступом.	10.04.2020	
2	Аналіз методів детектування обличчя та розпізнавання особистості.	15.04.2020	
3	Алгоритмічне забезпечення системи.	25.04.2020	
4	Реалізація програмного забезпечення системи.	10.05.2020	
5	Охорона праці.	20.05.2020	

Студент _____ Гайдук К. Г.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Сазонов А. Ю.
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту.

РЕФЕРАТ

Дипломний проект на здобуття ступеня бакалавра містить аналіз різних архітектур згорткових нейронних мереж та їх гіперпараметрів, аналіз способів підготовки зображень для коректної роботи нейронної мережі, вибір оптимальної архітектури та її налаштування, а також способу детектування обличчя та визначення особистості. Пояснювальна записка містить 22 рисунок та 25 літературних джерела.

Предметом дослідження дипломного проекту є обрана архітектура нейронної мережі та налаштування її гіперпараметрів.

У дипломному проекті досліджено найвідоміші типи архітектур згорткових нейронних мереж та перевірена їх робота на нашому наборі даних. За результатами досліджень була підібрана своя оптимальна архітектура та підібрані гіперпараметри до неї. Відповідно до складеного алгоритму реалізовано програмне забезпечення.

Ключові слова: Контроль та управління доступом, нейронна мережа, face detection, mask recognition, embedding.

ABSTRACT

The graduate bachelor's degree project includes an analysis of different convolutional neural network architectures and their hyperparameters, analysis of ways to complete image preprocessing, selection of optimal architecture and its settings, method of face detection, and person recognition. The explanatory note contains 22 drawings and 25 literary sources.

The subject of research of the diploma project is the chosen neural network architecture and defining of its hyperparameters.

In the diploma project, the most common convolutional neural network architectures were researched and tested on our dataset. Based on the results of the research the optimal self-made architecture was created and tuned. According to the compiled algorithm, the software was implemented.

Keywords: Physical Access Control System, neural network, face detection, mask recognition, embedding.

ЗМІСТ

ВСТУП

Тема дипломного проекту «Автоматизація контролю та управління доступом до виробничих приміщень в умовах санітарних обмежень».

Більшість сучасних підприємств, як виробничого так і невиробничого сектору, задля забезпечення безпеки та дотримання внутрішнього розпорядку працівниками вдаються до використання автоматизованих систем контролю управління доступом (АСКУД). АСКУД (англ. *Physical Access Control System*, *PACS*) – це сукупність програмно-апаратних технічних засобів контролю і засобів управління, завдання яких є обмеження і реєстрація входу-виходу персоналу на заданій території через «точки проходу»: двері, ворота, КПП [1]. При цьому, виклики, що ставить перед людством пандемія короновірусної інфекції вимагає підвищити рівень безпеки на виробництві, задля перешкоджання розповсюдження небезпечних факторів серед працівників.

Метою проекту є підвищення безпеки та забезпечення контролю за дотриманням карантинних обмежень шляхом розроблення автоматизованої системи контролю та управління доступом на основі візуального спостереження та інтелектуального аналізу зображень.

Для досягнення мети необхідним є вирішення наступних задач:

1. Провести аналіз проблем існуючих АСКУД.
2. Проаналізувати сучасні методи детектування обличчя, існуючі архітектури нейронних мереж, які можуть підійти для вирішення поставленої задачі, та методи розпізнавання особистості.
3. На основі проведеного аналізу sрозробити структурну схему АСКУД для контролю дотримання правил карантину.
4. На основі розробленої структурної схеми АСКУД розробити програмне забезпечення для контролю дотримання правил карантину.

1. АНАЛІЗ ПРОБЛЕМ ІСНУЮЧИХ АВТОМАТИЗОВАНИХ СИСТЕМ КОНТРОЛЮ УПРАВЛІННЯ ДОСТУПОМ.

1.1. Поняття АСКУД

В останні декілька десятиріч з розвитком технічних засобів забезпечення безпеки людей активно розвивались і розповсюджувались автоматичні системи контролю управління доступом (АСКУД).

АСКУД є системами, які здійснюють фізичний доступ або перешкоджання доступу до певних територій або будівель. Найчастіше АСКУД встановлюється з метою уникнення несанкціонованих проникнень особами, які не мають доступу до об’єкту, запобігання крадіжок та актів вандалізму. Особливо корисними такі системи є в тих місцях, які потребують охорони більш високого рівня, наприклад місця скупчення людей, які є привабливими для озброєних нападів або захоплення. На відміну від простих парканів та дверей, які також здійснюють фізичне перешкоджання, АСКУД можуть бути розроблені із майже будь-яким функціоналом, наприклад запам’ятовувати та контролювати хто та коли намагався здійснити або здійснив вхід до певного місця, скільки людина знаходилася там та навіть аналізувати її дії і реагувати на них, якщо є така необхідність. [2]

Найпростіші варіанти АСКУД вже давно присутні в нашому житті і кожен хоча б раз натрапляв на них. Більшість державних та приватних установ, навчальні заклади, торгівельно-розважальні центри, вокзали та аеропорти, тощо мають різного рівня обладнання для забезпечення контролю доступом. В Україні найбільш розповсюдженими АСКУД є турнікети, які, як правило, стоять на головних прохідних до будівель, хоча подекуди можна зустріти і більш серйозні системи, які можуть буди обладнані металошукачами або навіть біометричними контролерами для зчитування відбитків пальців. [3]

Різні АСКУД можуть мати різну складність, різне апаратне забезпечення та різну мету, проте всі вони працюють за приблизно одним алгоритмом та мають приблизно одну структуру [3]:

1. **Місце здійснення доступу.** Фізична перешкода, наприклад турнікет, брама або двері.
2. **Персональна ідентифікація.** Ключовий елемент в структурі АСКУД, який є унікальним для кожної особи і потрібен для того, щоб з'ясувати особистість. Може бути у вигляді магнітної картки, паролю, чіпу, біометричних даних (напр. відбитки пальців, сітківка ока), тощо.
3. **Зчитувач.** Стаціонарний пристрій, який стоїть в місці здійснення доступу, який приймає ідентифікаційні данні та передає їх на контролер.
4. **Контролер.** Пристрій (напр. комп'ютер), який отримує дані зі зчитувача, аналізує їх та приймає рішення, чи надавати доступ особі із цією персональною ідентифікацією. Якщо дані верифіковані, то контролер передає дозвіл на прохід на місце здійснення доступу, в противному випадку забороняє. Дані також можуть виводитися на комп'ютер.
5. **Комп'ютер.** В деяких системах комп'ютер застосовується лише для того, щоб уповноважений оператор міг слідкувати за станом цієї системи, а в інших комп'ютер відіграє головну роль, тобто сам є контролером і пристроєм спостереження за системою.
6. **Сервер**, який зв'язує між собою всю систему та зберігає необхідні дані.

1.2. Проблеми існуючих АСКУД

Усі існуючі наразі АСКУД можна класифікувати за способами їх реалізації за двома групами: контролерні та програмні. [4] Першими в світі почали розробляти саме контролерні системи, через те що на той час, коли вперше з'явилася потреба створення АСКУД, не існувало програмних засобів, які б давали можливість створення таких систем. Центральним органом контролерних АСКУД є саме контролер – окремий пристрій, який приймає дані зі зчитувачів, обробляє їх та приймає рішення про надання доступу чи відмову в ньому.

На заході, де вперше з'явилися контролерні АСКУД, поступово відмовляються від них, проте в Україні та решті світу системи такого типу все ще є переважними, тому, кажучи про недоліки АСКУД, маються на увазі саме системи побудовані із використанням окремих контролерів. [4]

Головними проблемами таких систем є відсутність можливості масштабування, недостатня швидкість роботи та обмежений спектр проблем, які можуть вирішувати такі системи, проте у сучасному світі питання безпеки є дуже гострим, а новітні стандарти потребують неперервного розвитку АСКУД в цих напрямках та багатьох інших, зокрема непомітність.

Для задоволення таких вимог на заміну класичним системам приходять технології XXI сторіччя із програмними засобами на чолі, які якісно відрізняються від попередників. Програмні АСКУД – сукупність програмно-технічних засобів автоматизації, головною відміною яких є відсутність спеціального контролера, а вся решта пристроїв системи підключена напряму до головного комп'ютера, що дозволяє миттєво без посередників обробляти інформацію, яка надходить зі зчитувачів, і тим самим підвищити швидкість роботи. Крім того в будь-який час з головного комп'ютера можна здійснювати налаштування системи, якщо з'являється потреба у тимчасових специфічних змінах у роботі пристроїв. При інтуїтивно-зрозумілій візуальній реалізації

таких програм, налаштування зможе зробити навіть співробітник, який не є. програмістом.

Головною перевагою програмних АСКУД є майже нескінченна масштабованість таких систем. В будь-який момент часу можна придбати та підключити будь-яку кількість нових пристроїв, та реалізувати нову логіку системи у вигляді програмного коду, інколи навіть із застосуванням штучного інтелекту.

Все більше застосовується і машинний зір, який дозволяє аналізувати візуальну інформацію, яка надходить з камер. Приклади таких систем вже можна побачити в деяких сучасних аеропортах, де відмовились від сканувальних рамок і перейшли на дистанційне спостереження через камери, тому що у часи пік стовпотворіння людей у черзі на вхід є привабливим для терористичних актів.

Інші проблеми, які також можуть буди вирішними за рахунок програмних АСКУД, виникли завдяки пандемії короновірусної інфекції, коли з’явилася потреба контролю людей на дотримання правил карантину та правильності носіння медичних масок. Такі системи можуть бути створені із застосуванням технологій машинного зору та штучного інтелекту і можуть стояти в будь-якому приміщенні, тому що все що потрібно для їх роботи це встановлена камера у необхідному місці та її підключення до головного комп’ютера. Таку систему можна масштабувати встановленням інфрачервоної камери, яка дозволить контролювати температуру співробітників у будь-який момент часу, що дозволить швидко реагувати на один із можливих симптомів прояву вірусу.

1.3. Постанова задачі

Під час пандемії короновірусної інфекції та у випадку схожих обставин у майбутньому, важливе збереження режиму роботи підприємств близького до штатного, проте також необхідне створення усіх умов для співробітників та вживання запобіжних заходів проти поширення вірусу. Одним із способів боротьби проти розповсюдження хвороби є носіння медичних масок [5] та перевірка температури [5]. Можливо реалізувати програму, яка дистанційно за допомогою камер, технологій комп'ютерного зору та штучного інтелекту зможе контролювати співробітників та сповіщати їх у разі порушення правил носіння маски або підвищеної температури.

Для доступу до підприємства та контролю працівників у виробничих приміщеннях необхідне наступне:

1. Отримати зображення з камери в реальному часі.
2. На отриманому зображенні за допомогою попередньо навченої штучної нейронної мережі (ШНМ) знайти обличчя людини, що є зоною інтересу (англ. *Region Of Interest, ROI*) для наступних кроків.
3. За допомогою іншої ШНМ проаналізувати виділене обличчя та класифікувати це зображення за показниками наявності маски:
    a. Маска вдягнена правильно.
    b. Маска вдягнена неправильно.
    c. Маска відсутня.
4. Якщо маска відсутня або вдягнена неправильно, виконується розпізнавання особистості, після чого цій людині надсилається повідомлення на особистий телефон, а саме в Telegram бот, який потрібен бути попередньо встановлений, із доказом порушення у вигляді зображення людини, часом, датою та проханням дотримуватися правил карантинних обмежень.

2. ІСНУЮЧІ МЕТОДИ ДЕТЕКТУВАННЯ ОБЛИЧЧЯ ТА РОЗПІЗНАВАННЯ ОСОБИСТОСТІ.

2.1. Поняття комп'ютерного зору

Візуальна інформація відіграє одну з ключових ролей в житті людства. Майже половина усіх нейронів людського мозку спрямована на обробку та аналіз того, що ми бачимо, та приблизно 80% всього інтернет-трафіку складає візуальний контент у вигляді відео та зображень. [6]

Комп'ютерний зір – одна із сфер комп'ютерних наук, головною метою якою є спроба відтворити можливості людської зорової системи за допомогою сучасних програмних та технічних можливостей. Завдяки технологіям комп'ютерного зору наразі є можливість збирати, обробляти, аналізувати та класифікувати візуальні дані, яких з розвитком інтернету стає тільки більше, так, наприклад, кожен день в мережу інтернет завантажується близько 3 мільярдів зображень. [7]

Історія розвитку сфери комп'ютерного зору починається задовго до появи перших потужних комп'ютерів з досліджень біологів. Перший крок до розуміння, як працює зорова система у людей та тварин, був зроблений у 1950-х роках. Тоді переважно дослідження проводили на кішках, тому що структурно саме їх мозок схожий на мозок людини. Завдяки цим дослідженням було з'ясовано, що головну роль при аналізі зображень відіграють ті нейрони, які реагують на зміщення контурів при русі предметів.

В 1960 – 1990 роках було задіяно велику кількість вчених зі всього світу та було запропоновано безліч алгоритмів та теорій для вирішення задач комп'ютерного зору, проте більшість з них не знайшли застосування ні тоді, ні в сучасному світу. Іншим фактором гальмування розвитку цієї сфери була недостатня обчислювальна потужність комп'ютерів того часу. [6]

В середині 1990 років через все ще малу потужність комп'ютерів, що не давало можливості класифікації об'єктів, у групи вчених та розробників

виникла ідея сегментації зображень. Сегментація – групування пікселів зі схожими властивостями в зони інтересу (англ. *Region Of Interest, ROI*), що дає можливість з'ясувати тип об'єкту, до якого може належати така зона інтересу.

В 2000 – 2007 роках вперше вдалося розпізнати обличчя людини на зображенні, що на той момент було задачею номер 1, проте головним проривом в цих роках стало створення алгоритму SIFT (scale-invariant feature transform). Алгоритм SIFT дозволяє знаходити локальні ознаки зображення та робити це в незалежності від кута нахилу зображення, освітлення, тощо. Ідея алгоритму полягає в пошуку ознак на зображеннях, які залишаються незмінними при будь-яких трансформаціях, та зіставлення їх. [6]

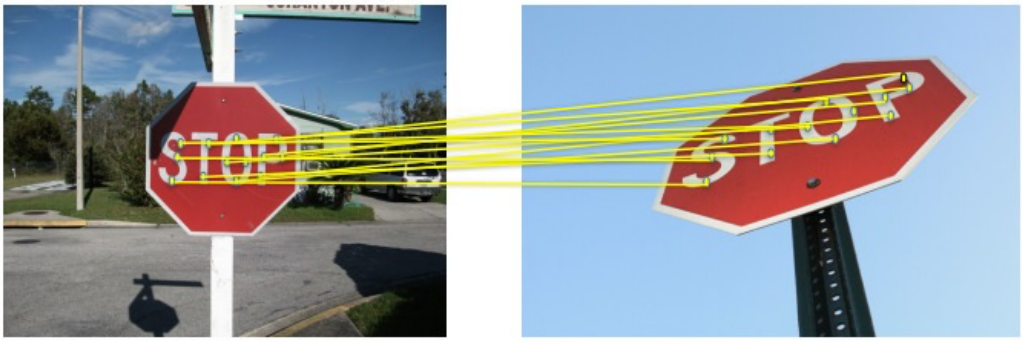

Рис. 2.1 Приклад роботи алгоритму SIFT.

Починаючи з 2007 року по наші дні сфера комп'ютерного зору розвивається дуже стрімко і причинами для цього є небувала кількість візуальної інформації, яка є в публічному доступі, розвиток сфери машинного навчання та стрімке зростання обчислювальної можливості у сучасних центральних (англ. central processing unit, *CPU) та графічних* (англ. graphics processing unit, *GPU*) процесорів.

Застосування технологій комп'ютерного зору набирає все більшої популярності і все частіше можна побачити автоматизовані системи, які працюють на базі цих технологій, наприклад:

1. **Безпілотні автомобілі та автопілоти.** Комп'ютерний зір дозволяє безпілотним автомобілям орієнтуватися у просторі. Камери з різних

сторін автомобіля в режимі реального часу передають зображення навколишнього середовища. Ці зображення поступають до програмного забезпечення, яке знаходить дорожню розмітку, знаки, світлофори, інші автомобілі, пішоходів, тощо. [7]

2. **Медицина.** Технології комп'ютерного зору вже не перший рік використовуються для аналізу родимок на шкірі та виявлення тих, які мають загрозу утворення раку, або аналіз рентгенівських знімків для пошуку хронічних захворювань. [7]
3. **Доповнена реальність та дизайн.** У цих сферах комп'ютерний зір дозволяє отримувати 3D модель приміщень за рахунок декількох десятків фотографій цього приміщення з різних ракурсів, або в реальному часі додавати на відео об'єкти, щоб подивитися, як вони будуть виглядати наживо. [7]

2.2. Методи детектування обличчя

Детектування обличчя на зображеннях є однією з перших задач, яку десятиліттями намагалися вирішити науковці у сфері комп'ютерного зору, і, хоча перше обличчя на фотографії було знайдено ще на початку 2000 років завдяки методу «AdaBoost», який був розроблений Паулем Віолою та Майклом Джонсоном [6], ця сфера комп'ютерного зору не перестає розвиватися, а навпаки привертає все більше уваги програмістів зі всього світу.

Рис. 2.2. Демонстрація роботи метода Віоли-Джонса.

Складність задачі розпізнавання обличчя полягає у різноманітності самих облич та умов, за яких здійснюється детектування. Різна орієнтація обличчя в просторі, колір шкіри, наявність бороди або окулярів, захисною маски або прикрас, освітлення або якість фотографії – все це суттєво впливає на точність розпізнавання обличчя, а в деяких випадках робить це неможливим. Таким чином, хоча системи і алгоритми для вирішення цієї проблеми безперервно вдосконалюються, назвати задачу детектування обличчя вирішеною наразі не можливо. [7]

Усі наразі існуючі методи детектування обличчя можна класифікувати за чотирма групами [8]:

1. Емпіричні методи.
2. Методи інваріантних ознак.
3. Детектування за шаблонами.
4. Методи зовнішніх ознак.

2.2.1. Емпіричний метод

Емпіричний метод полягає у створенні алгоритму, який буде мати набір правил, котрим повинен відповідати фрагмент зображення, для того щоб бути визнаним обличчям. Таким чином цей набір правил є спробою формалізувати алгоритм, яким керується зорова система та мозок людини, коли потрібно класифікувати обличчя. Деякими припустимими правилами можуть бути [8]:

1. Центральна частина обличчя має однорідну яскравість та колір.
2. Різниця в яскравості між центральною частиною та лобовою може буди дуже значною.
3. На обличчі повинні бути присутніми два симетрично розташовані ока, ніс, рот, котрі зазвичай значно відрізняються від інших частин обличчя яскравістю.

В сучасному світу з розвитком фотоапаратури зображення стають все більш якісними та чіткими, проте для більш коректної роботи емпіричного методу вдаються до сильного стиснення зображень, що дозволяє згладжувати шуми, а також зменшувати обчислювальне навантаження. Після стиснення зображення та перетворення його у відтінки сірого, легше помітити зону рівномірного розподілу яскравості, що потенційно і є обличчям, а потім перевірити наявність різких змін яскравості всередині цієї зони (очі, ніс та рот), що може підтвердити, що це саме обличчя.

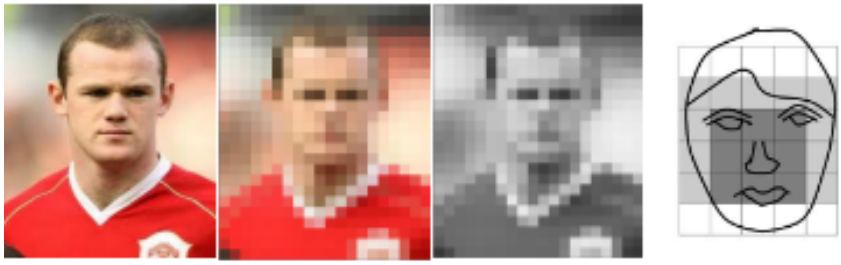

Рис. 2.3. Демонстрація роботи методу стиснення.

Іншим емпіричним методом знаходження обличчя є метод побудови гістограм, який полягає у побудові горизонтальної та вертикальної гістограм яскравості. Зони найбільш яскравих ділянок, що потенційно є обличчям, будуть виглядати як вершини гістограм, які можна відокремити прямими лініями. Перетин прямих вертикальною та горизонтальної гістограми і є потенційним шуканим обличчям.

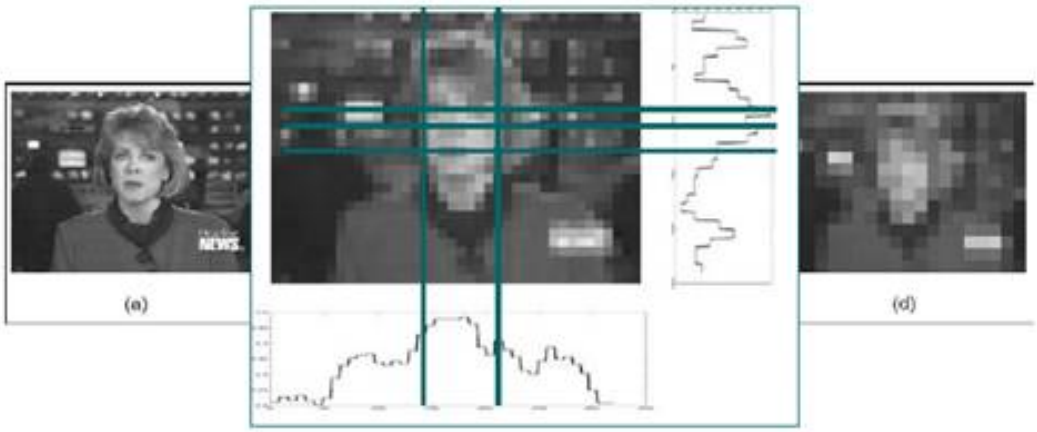

Рис. 2.4. Демонстрація роботи методу побудови гістограм.

Методи, які були описані вище, потребують досить малу обчислювальну потужність процесора для обробки зображення, проте вони мають гарні показники лише в тих випадках, коли зображення має однорідний фон. Хоча було дуже багато спроб вдосконалити ці алгоритми, всі вони непридатні при роботі із зображеннями із складними фоном, декількома обличчями та нестандартними кутами нахилу голови, тому всі вони використовувались лише на зорі розвитку комп'ютерного зору.

2.2.2. Метод інваріантних ознак

Група цих методів принципово відрізняється від сімейства емпіричних методів тим, що не намагаються наслідувати процеси зорової системи та мозку людини, а навпаки прагнуть знайти закономірності та інваріантні (незмінні) особливості саме облич. Основними етапами в цих методах є:

1. Пошук явних ознак обличчя: очі, ніс, рот.
2. Пошук форми, яскравості, кольору та меж потенційного обличчя.
3. Об’єднання всіх ознак та їх перевірка.

Перший метод із цієї групи – метод пошуку облич в складних сценах, який базується на пошуку геометрично правильно розташованих ознак обличчя. Для цього застосовується гаусовський фільтр із різним масштабом та орієнтацією, після чого виконується пошук ознак та аналізується їх взаємне розташування.

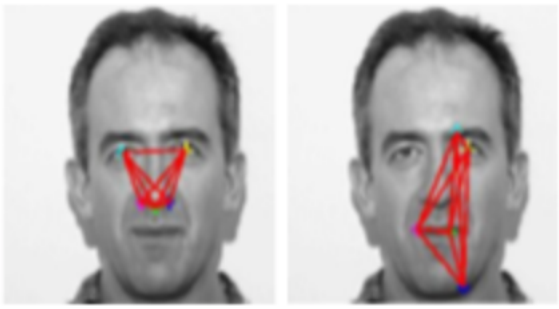

Рис. 2.5. Демонстрація роботи методу пошуку облич в складних сценах.

Наступний метод – це метод групування ознак, який також застосовує гаусовський фільтр для пошуку необхідних областей зображення. Після цього межі навколо кожної області групуються, а знайдені ознаки комбінуються за допомогою баєсовської сітки. Таким чином відбувається пошук рис обличчя.

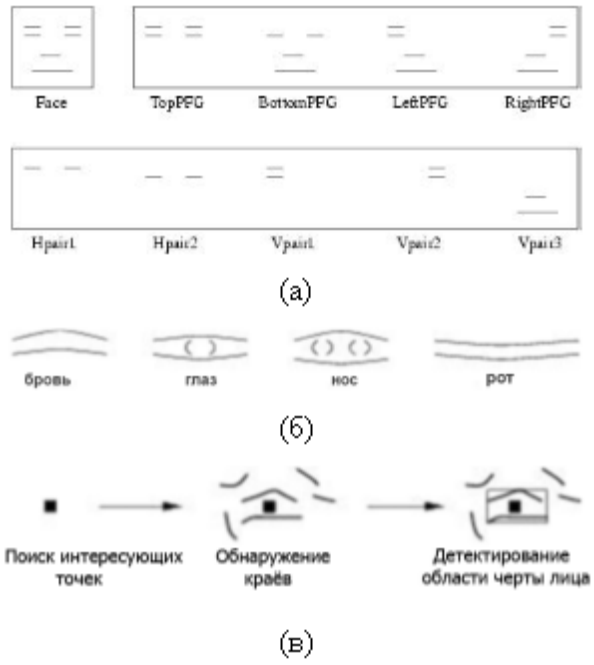

Рис. 2.6. Демонстрація роботи методу групування ознак
а) моделі обличчя та риси, б) парні межі рис обличчя, в) виділення областей обличчя.

Нажаль методи цієї групи, хоча і відрізняються від емпіричних своїм підходом, проте недоліки в них дуже схожі. Будь-який об’єкт, який хоча б трохи закриває обличчя, шуми, засвітлення або складний фон, будуть заважати коректній роботі алгоритму. Перевагами цих методів є можливість розпізнавання обличчя в різних ракурсах та кутах нахилу.

2.2.3. Детектування за шаблонами

Під шаблонами розуміють стандартне зображення обличчя, шляхом описую окремих його частин та їх взаємного розташування. Пошук обличчя за шаблоном передбачає перевірку окремих частин зображення на відповідність заданому шаблону. Виділяють два типи шаблонів:

1. Деформовані
2. Недеформовані

Першим методом в цьому сімействі є метод детектування обличчя із застосуванням тривимірних форм, який полягає у використанні шаблона у вигляді пар відношень яскравості в двох областях. Для того щоб знайти обличчя на зображенні потрібно перевірити усі можливі ділянки цього зображення за цим шаблоном та зробити це у різних масштабах.

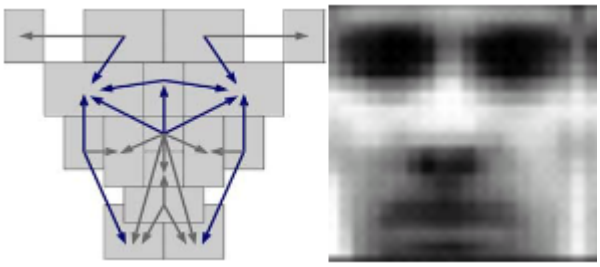

Рис. 2.7. Демонстрація роботи методу детектування
за допомогою тривимірних форм.

Іншим способом розпізнавання обличчя є метод із застосуванням моделі розподілу точок, яка є об’єктом, форма яких може змінюватись. Такий спосіб детектування може використовуватись для розробки ефективних систем класифікації.

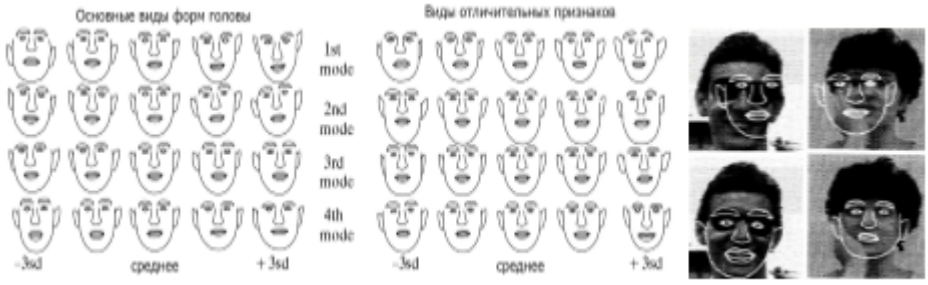

Рис. 2.8. Приклад моделей розподілу точок.

Перевагою методів розпізнавання із використанням шаблонів є непогані результати на зображеннях із нескладним фоном, проте головним недоліком є потрібність калібрування шаблона над областю обличчя, тому складність

використання цього методу на наборі зображень із різноманітним ракурсами не є доцільним.

2.2.4. Методи зовнішніх ознак

Зображення перевіряється із вектором ознак, обчисленим деяким шляхом, який використовується для класифікації зображень на два класи [8]:

1. Обличчя
2. Не обличчя

Зазвичай, методи, які базуються на математичних моделях, потребують перевірки усіх прямокутних фрагментів зображення на наявність обличчя, проте недоліком такого способу є велика обчислювальна складність, тому зараз частіше можна побачити розділення зображення на невелику кількість сегментів.

Рис. 2.9. Приклад розбиття зображення на фрагменти.

Для навчання алгоритмів треба підготувати базу фотографій із обличчями та без.

Рис. 2.10. Приклад бази зображень. [8]

Основною задачею є пошук сильних класифікаторів і відбракування тих, які були отримані через шуми. Існує дуже багато способів вирішення такої задачі:

1. Штучні нейронні мережі (англ. Neural Networks, NN)
2. Метод опорних векторів (англ. Support Vector Machines, CVM)
3. Факторний аналіз (англ. Factor Analysis)
4. Метод розподілення (англ. Distribution-based method)

та велика кількість інших. Наразі найпопулярнішим способом знаходження обличчя є саме метод нейронних мереж.

3. АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ЗАПРОПОНОВАНОЇ СИСТЕМИ КЕРУВАННЯ ТА УПРАВЛІННЯ ДОСТУПОМ В УМОВАХ САНІТАРНИХ ОБМЕЖЕНЬ

Проект запропонованої автоматизованої системи керування та управління доступом в умовах санітарних обмежень можна розділити на п’ять великих частин, де кожний наступний блок є логічним продовженням попереднього, а саме:

1. Створення бази фотографій із правильно вдягнутою, неправильно та відсутньою маскою та їх попередня обробка
2. Навчання нейронної мережі
3. Розпізнавання особистості
4. Створення Telegram Bot
5. Головна программа, яка використовує розпізнавання маски, особистості людини та надсилає повідомлення в Telegram Bot

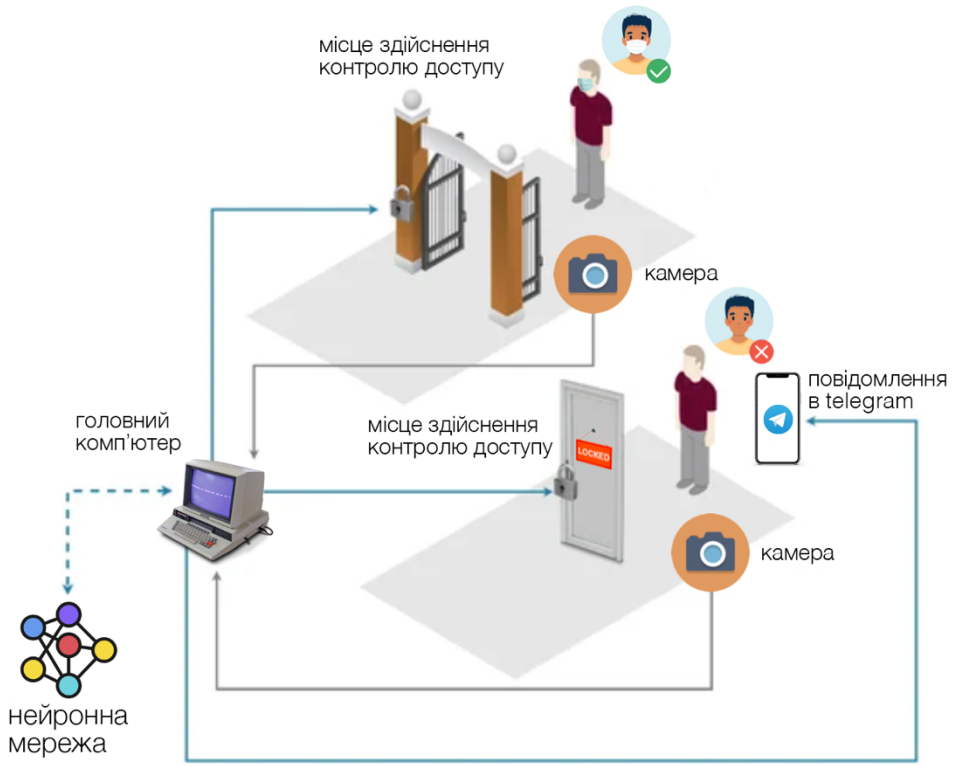

Рис 3.1. Графічна інтерпретація запропонованої АСКУД

На рис 3.1. графічно зображена схема запропонованої АСКУД та дві ситуації. У випадку, коли відвідувач дотримався правил карантинних обмежень, тобто маска вдягнена вірно, прохід дозволено. У противному випадку – заборонен.

Перші три блоки є підготовчими та виконуються лише один раз для того, щоб створити файли з необхідними даними для роботи головної програми, частиною якої є четвертий пункт.

3.1. Збір даних та їх попередня підготовка

Збір даних та їх попередня підготовка є одним із головних етапів при роботі із нейронними мережами та дуже впливовим на їх ефективність та точність.

3.1.1. Збір даних

Збір даних починається із визначення типів даних, які в майбутньому будуть використані для навчання нейронної мережі. В нашому випадку робота проводиться із візуальним контентом, тобто зображенням, яке надходить з камери, тому і навчання нейронної мережі буде відбуватися на великій кількості зображень. Метою нашою моделі ШНМ є коректна класифікація зображень за трьома показниками:

1. Маска вдягнена правильно
2. Маска відсутня
3. Маска вдягнена неправильно

тому було зібрано великий датасет із фотографіями, які відповідають цим трьома типам. Головними цілями при створенні бази зображень були якість фотографій та різноманітність: різні ракурси, освітлення, колір шкіри, наявність бороди або аксесуарів на обличчі.

Рис. 3.2. Приклади фотографій із зібраного датасету

Іншим важливим фактором при створенні датасету є кількість зображень та їх рівномірне розподілення. Кількість зображень впливає на якість навчання ШНМ: чим більше різноманітних зображень, тим до більшою кількості реальних ситуацій буде готова навчена модель. Під рівномірним розподіленням фотографій мається на увазі рівна кількість зображень в кожному класі. Робиться це для того, щоб майбутня модель не мала схильності до того чи іншого класу зображень.

Наступним кроком є розділення отриманої бази зображень на навчальну та випробувальну. Перша буде використовуватися безпосередньо для навчання ШНМ, а друга для перевірки якості роботи навченої моделі. Було прийнято рішення, що в навчальному датасеті буде по 3500 зображень в кожному класі, а в тестувальному по 350.

3.1.2. Підготовка зображень

Нейронні мережі є дуже чутливими при навчанні і майже будь-що може погіршити якість навченої моделі, тому обов’язково вхідні данні повинні бути попередньо оброблені різними методами.

Перше, що робить головна програма на початку кожного циклу – шукає обличчя, виділяє його і в наступних кроках працює виключно з ним, тому перше, що можна зробити в рамках підготовки зображень - обрізати всі

фотографії по обличчю, так як все інше на фотографії є зайвим. Зробити це можна за допомогою будь-якого методу пошуку обличчя, в нашому випадку використовується один із методів ШНМ.

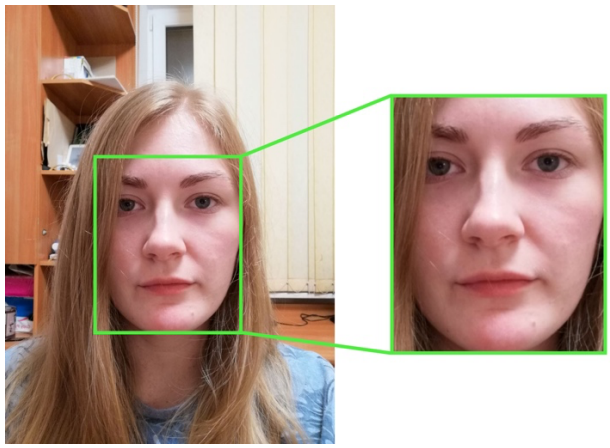

Рис. 3.3. Демонстрація етапу обрізання зображень

Наступним кроком є аугментація даних. Аугментація (англ. Augmentation) – популярний спосіб боротьби із перенавчанням ШНМ за допомогою штучних змін даних, наприклад: поворот, зміна кольору, деформація, наближення, тощо. Аугментація особливо корисна в тих випадках, коли потрібно штучно наповнити базу даних.

Рис. 3.4. Демонстрація процесу аугментації

Третім логічним кроком буде зміна моделі кольору з RGB (англ. Red, Green, Blue – Червоний, Зелений, Синій) на GRAYSCALE (англ. Відтінки сірого), тому що в нашому випадку немає сенсу аналізувати кольорову гамму зображень, а зайві альфа-канали додадуть лише обчислювального

навантаження та займуть більше місця. Існує два [9] головних способи переводу із RGB формату в GRAYSCALE:

1. Метод середнього:

$$\text{grayscale} = \frac{(\text{R} + \text{G} + \text{B})}{3} \tag{3.1}$$

2. Ваговий метод:

$$\text{grayscale} = 0.299\text{R} + 0.587\text{G} + 0.114\text{B} \tag{3.2}$$

Головним недоліком методу середнього є те, що він не враховує чутливість людського ока до різних кольорів, для чого і був виведений ваговий метод, з формули якого можна побачити, що наше око є найбільш сприйнятливе до зеленого кольору.

Рис. 3.5. RGB в GRAYSCALE

Наступним кроком є перетворення усіх зображень до одного розміру, при чому ширина та висота зображення повинні бути рівними. Розмір, до якого будуть приведені зображення не є принциповим, проте розмір 1000x1000 потребує більше обчислювальної потужності, пам’яті та часу, ніж розмір 150x150, який і використовується в нашому випадку.

Рис. 3.6. Зміна розміру зображення

П’ятим кроком є нормалізація зображення, що означає, що значення пікселів, які лежать в діапазоні від 0 до 255, діляться на 255. Таким чином нові значення пікселів будуть лежати в діапазоні від 0 до 1, що значно спрощує обрахунки.

Фінальними кроками є перемішування зображень між собою, щоб ШНМ не мала схильності до того чи іншого класу зображень, та зберігання двох файлів:

1. Із підготовленими та перемішеними зображеннями
2. Із мітками зображень.

В нашому випадку мітки зображення наступні: 0 – маска присутня, 1 – маска відсутня, 2 – маска вдягнена неправильно.

Увесь цей цикл робиться для двох наборів даних:

1. Тренувального
2. Тестувального

3.2. Навчання нейронної мережі

Навчання нейронної мережі – процес пошуку оптимальних значень вагових коефіцієнтів, яке відбувається в два етапи:

1. Пряме поширення помилки – передбачення відповіді [10]

2. Зворотне поширення помилки – мінімізація (оптимізація) помилки між правильною відповіддю та передбаченням [10]

У дипломному проекті для навчання нейронних мереж використовуються зображення, які попередньо вже були підготовлені і збережені у файли, тому першим етапом є завантаження з цих файлів зображень та міток тренувального та тестувального набору даних.

Другим етапом є визначення гіперпараметрів, які також можна назвати налаштуваннями нейронної мережі. В нашому проекті налаштовуються наступні гіперпараметри:

1. Epochs – кількість разів, які проходить датасет через нейронну мережу в прямому та зворотному напрямку.
2. Batch size – визначає кількість зображень, які за раз будуть надходити в нейронну мережу, тому що не можливо одразу пропустити увесь датасет.
3. Validation Split – визначається в діапазоні від 0 до 1 та показує, яка частина від тренувального набору даних буде використана як валідаційний набір, котрий слугує для того, щоб під час навчання перевіряти мережу на перенавчання, тобто суцільне запам’ятовування зображень.

В нашому проекті використовуються ще декілька гіперпараметрів, а саме: кількість фільтрів згорткового ядра, розмір ядра та розмір ядра підвиборки, проте вони зазвичай визначаються для кожного шару окремо.

Третій етап – вибір архітектури нейронної мережі та створення моделі. В нашому проекті взаємодія іде лише с зображеннями, тому працювати потрібно із згортковими нейронними мережами (англ. Convolutional Neural Network, CNN), так як цей тип ШНМ найкраще підходить для такого типу задач. Наразі існує велика кількість різних архітектур згорткових нейронних мереж, які гарно показують себе в різних ситуаціях, проте було прийнято рішення пошуку своєї архітектури замість використання універсального

варіанту. В ході експериментів вибір пав на архітектуру, яка має наступну структуру (Рис 3.7):

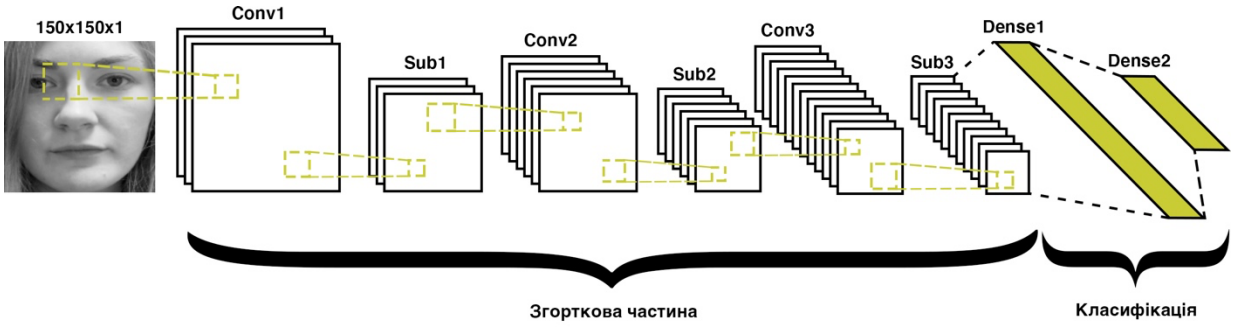

Рис. 3.7. Архітектура нейронної мережі

Архітектури згорткових нейронних мереж діляться на дві частини: згорткову та класифікацію. Перша відповідає за вилучення ознак з зображень, а друга відповідно вчиться відрізняти їх. Шари згорткової частини можна об'єднати по блокам згортка + підвиборка. В нашому випадку всі 3 згорткові блоки мають однакові налаштування:

- **Розмір ядра згортки (Kernel size)** – (5, 5)
- **Кількість фільтрів ядра згортки (Number of Kernels)** – 64
- **Активація (Activation)** – ReLU (англ. Rectified Linear Unit)
- **Розмір ядра підвиборки** – (2, 2)

Класифікаційна частина складається з двох повнозв'язних шарів. Перший (Dense1) має 512 нейронів та активацію ReLU, а другий (Dense2) має 3 нейрони, тому що на виході нам потрібно отримати один з трьох класів: в масці, без маски, погано вдягнена, та активацію SoftMax, тому що вона краще підходить для класифікації вихідних даних, а ReLU для проміжних.

Наступні два кроки після створення архітектури нейронної мережі – компіляція моделі та безпосередньо навчання. Компіляція – етап, на якому вказуються функції втрат (англ. Loss function) та метод оптимізації. В нашому

випадку використовується краща з нині існуючих функцій оптимізації «Adam», а функція втрат «Categorical Crossentropy», яка є рекомендованою, коли є більше ніж 2 вихідних класи.

Після навчання модель перевіряється на тестувальному наборі даних та у вигляді графіка зберігається інформація по точності, втратам, валідаційній точності та валідайним втратам в ході навчання.

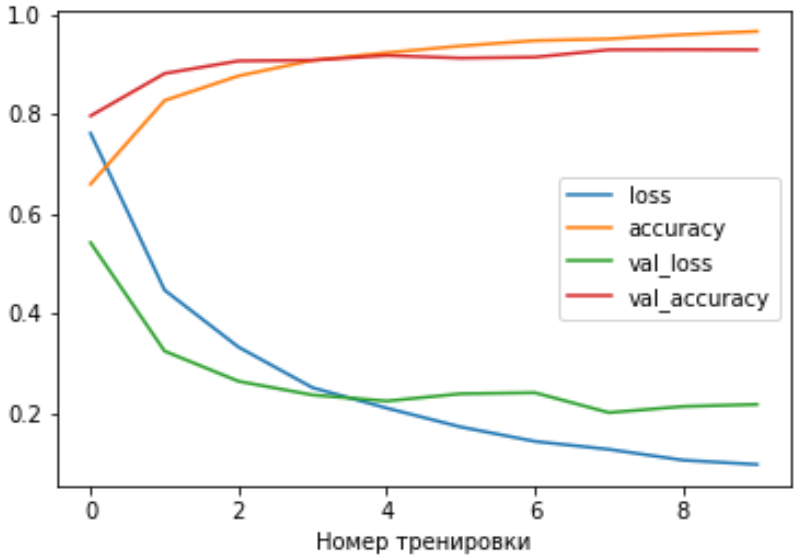

Рис. 3.8. Приклад візуалізації метрик

Останнім кроком є збереження отриманої моделі нейронної мережі та вагових коефіцієнтів в файл з розширенням «.model» для того, щоб не доводилося кожний раз знову навчати модель.

3.3. Розпізнавання особистості

Для того, щоб головна програма могла розпізнати ту чи іншу людину, спочатку треба створити базу фотографій людей, яких ми хочемо розпізнавати. В нашому випадку було зібрано по 30 фотографій кожної людини в різних ракурсах.

Наразі існує багато алгоритмів для розпізнавання особистостей, проте одним із найпопулярніших є метод, який запам’ятовує спеціальні ознаки на

зображеннях. Для цього використовується попередньо навчена ШНМ, яка для кожного обличчя людини створює масив із 128 ознаками.

Після того, як для кожної фотографії були створені ці масиви ознак, вони, а також ідентифікаційний номер людини записуються в файл для того, щоб їх можна було використовувати в майбутньому.

Розпізнавання люди в режимі реального часу відбувається за наступним алгоритмом:

1. На зображенні шукається обличчя людини
2. Це обличчя проходить через ШНМ, яка створює масив із 128 ознаками
3. Отриманні ознаки порівнюються із тими, які були збережені в файл, та відбувається голосування
4. Ідентифікаційний номер, який набрав більшість голосів, скоріш за все і відображає особистість, яка присутня на зображенні.

3.4. Алгоритм роботи Telegram Bot

Робота будь-якого Telegram Bot починається з команди «/start». Після введення цієї команди відбувається перевірка Telegram Id користувача на наявність в базі даних.

Якщо такого Telegram Id не знайдено, то бот надсилає повідомлення с проханням верифікуватись. Верифікація користувачів відбувається за рахунок надання номера телефону. Якщо наданий номер телефону існує в базі, то за цим номером закріплюється Telegram Id цього користувача, якщо номер не знайдено, то користувач отримує відповідне повідомлення та можливість ввести номер телефону знову.

Якщо при введені команди «/start» Telegram Id знайдено, тобто користувач вже пройшов верифікацію з цього пристрою, то бот просто вітає користувача та повідомляє його про те, що він вже верифікований.

Блок-схему алгоритму роботи Telegram Bot наведено на рис 3.9.

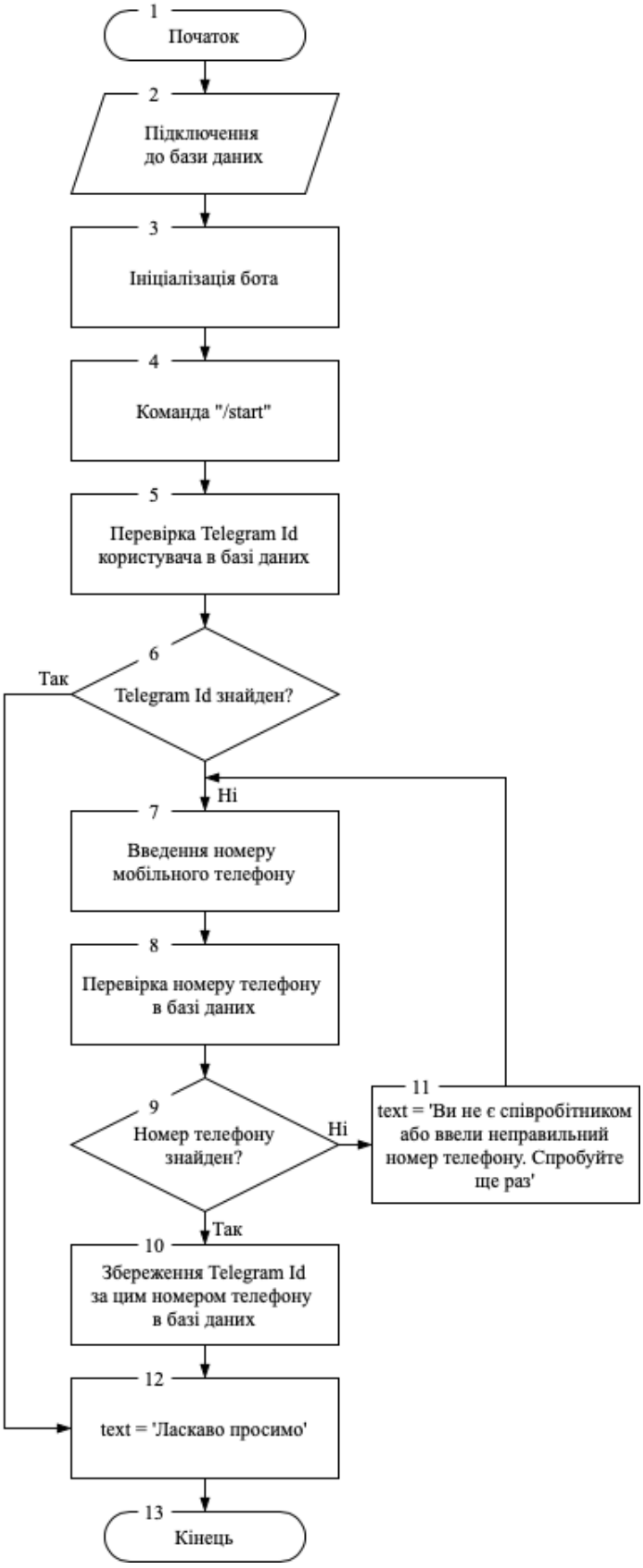

Рис. 3.9. Блок-схема роботи Telegram Bot

3.5. Головна програма

Цей блок проекту є головною програмою та логічним продовженням попередніх блоків.

Першим етапом є завантаження навченої моделі нейронної мережі з файлу, який був збережений на попередньому етапі проекту, після чого також завантажується файл, який містить масиви ознак людей, для розпізнавання особистості.

Для роботи із Telegram Bot використовується база даних, яка на даному етапі розробки проекту реалізована у вигляді файлу із розширенням «.json», тому другим кроком є завантаження цього файлу. Наразі не є доцільним використання більш складних типів баз даних, тому що проект не має остаточного вигляду.

Усі подальші кроки в цьому блоці будуть базуватися на зображенні, яке надходить з камери, тому на цьому етапі треба ініціалізувати камеру та почати отримувати відеопотік кадр за кадром.

На кожному кадрі виконується розпізнавання обличчя і, якщо обличчя або декілька облич знайдені, то для кожного з них виконується пошук медичної маски за допомогою попередньо навченої штучної нейронної мережі та відбувається класифікація стану носіння маски, де 0 – маска присутня, 1 – маска відсутня, 2 – маска вдягнена неправильно. Якщо стан маски не дорівнює 0, то відбувається процес розпізнавання особистості, який повертає ім’я людини та її унікальний ідентифікаційний номер або повертає «Unknown», що означає, що ця людина невідома.

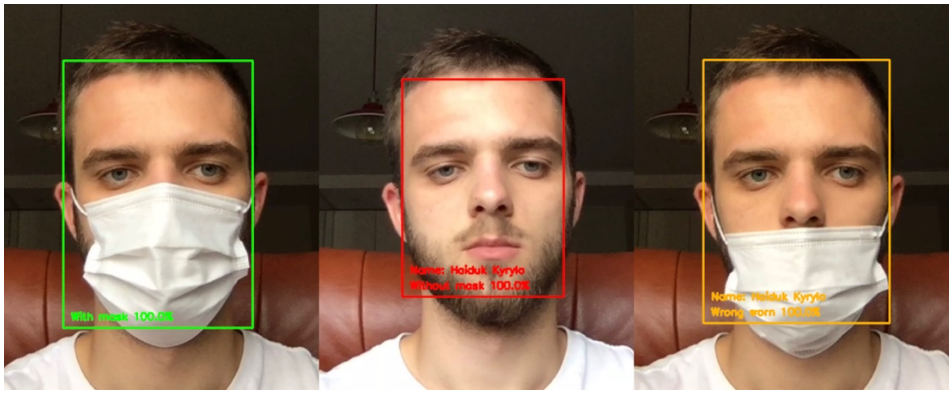

Рис. 3.10. Візуалізація роботи алгоритмів детектування маски
та розпізнавання особистості

Якщо все ж таки особистість людини відома для нашого алгоритму, то за її персональним ідентифікаційним номером з бази даних отримується її Telegram Id та виконується надсилання повідомлення, яке містить фотографію моменту порушення, час та прохання дотримуватись правил карантинних обмежень. Повідомлення буде надходити кожні 30 секунд до того моменту, поки людина не буде носити маску правильно.

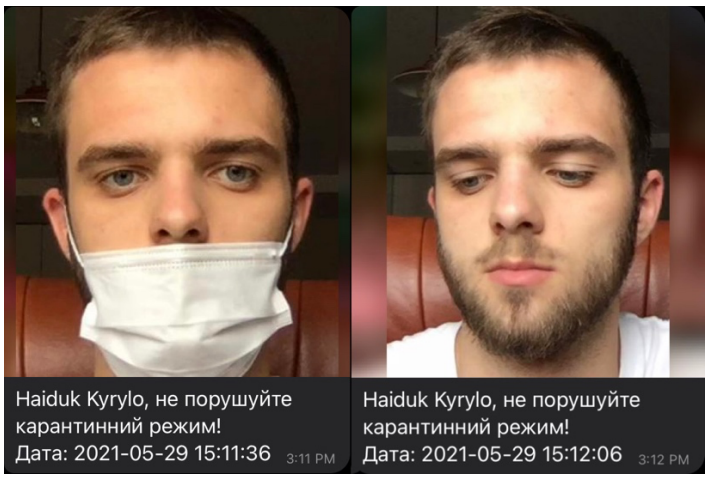

Рис. 3.11. Приклади повідомлень, які надійшли із затримкою в 30 секунд

Блок-схему головної програми наведено на рис 3.12.

Рис. 3.12. Блок-схема роботи головної програми

4. ПРОГРАМЕ ЗАБЕЗПЕЧЕННЯ ЗАПРОПОНОВАННОЇ СИСТЕМИ КЕРУВАННЯ ТА УПРАВЛІННЯ ДОСТУПОМ В УМОВАХ САНІТАРНИХ ОБМЕЖЕНЬ

Для реалізації запропонованої АСКУД була використана інтерпретована об’єктно-орієнтована мова програмування Python [11], яка за різними рейтингами наразі є однією з найпопулярніших мов програмування.

Така популярність зумовлена наступними перевагами цієї мови:

- **Синтаксис.** Програмування на Python дуже часто порівнюють із написанням тексту англійською мовою через легку читаність коду.
- **Широкий спектр застосування.** Python є однією з головних мов в WEB-розробці, Data Science, сфері комп’ютерного зору та машинного навчання.
- **Кросс-платформеність.** Python може бути використаним майже на всіх платформах, окрім мобільних, які найчастіше потребують специфічних мов програмування.
- **Розвиток мови.** Спільнота цієї мови є однією з найактивніших у сфері програмування, за рахунок чого з’являються все нові бібліотеки та фреймфорки, які дозволяють вирішувати все більше задач, використовуючи лише одну мову програмування.

Незважаючи на переваги цієї мови, список яких ще можна продовжити, треба миритися з рядом недоліків, а саме:

- **Продуктивність.** Python є достатньо повільною мовою програмування. Найчастіше причиною цього називають те, що ця мова є інтерпретованою, проте Python поступається по швидкості навіть іншим інтерпретованим мовам.

- **Динамічна типізація.** Можливість зміни типу змінної під час роботи програми потребує більше ресурсів та може призвести до збоїв.

Для нас головною причиною використання саме цієї мови програмування є те, що на цій мові написано безліч бібліотек, за допомогою яких можливо швидко та достатньо лаконічно реалізувати наш проект.

4.1. Використані бібліотеки

- **NumPy** – математична бібліотека, яка реалізована на мові програмування C, яка дозволяє виконувати складні маніпуляції із багатовимірними матрицями. [12, 13]
- **OpenCV (англ. Open-Source Computer Vision Library)** – бібліотека алгоритмів для обробки зображень, загальних чисельних алгоритмів та комп'ютерного зору з відкритим кодом. В нашому проекті OpenCV є однією з двох головних бібліотек, за допомогою якої відбувається отримання зображення з камери в режимі реального часу, підготовка зображень для навчання нейронної мережі та навіть створення рамок навколо обличчя та виведення тексту на відео. [14]
- **TensorFlow** – досить молода бібліотека, яка була створена командою Google в 2015, для машинного навчання. [15, 16] Ця бібліотека активно взаємодіє із іншою, не менш відомою в сфері машинного навчання та нейронних мереж, бібліотекою **Keras**, яка безпосередньо відповідає за зручну роботу із глибинними нейронними мережами. [17, 18] В нашому проекті ці дві бібліотеки в першу чергу використовуються для побудови архітектури нейронної мережі, її гіперпараметрів, самого навчання ШНМ, збереження моделі у файл з розширенням «.model» та подальшого використання навченої моделі.

- **Matplotlib**, а саме **PyPlot** для візуалізації результатів навчання ШНМ у вигляді графіків. [19, 20]
- **OS** – модуль для роботи із операційною системою та взаємодії із файловою системою комп'ютера. В проекті використовувався для роботи із папками всередині проекту та організації даних. [21]
- **Pickle** – модуль, який дозволяє швидко і зручно читати та створювати файли. [22]
- **Face Recognition** – модуль, завдяки якому створювались 128-розмірні матриці ознак облич для розпізнавання особистості. [23]
- **PyTelegramBotAPI** – модуль для створення та програмування Telegram Bots. [24, 25]

5. ОХОРОНА ПРАЦІ

Згідно з темою дипломного проекту «Автоматизація контролю та управління доступом до виробничих приміщень в умовах санітарних обмежень» будемо розглядати охорону праці для цього процесу. Відповідно до проекту оператор знаходиться в приміщені, яке обладнано одним робочим місцем та має розміри: загальна площа – 9 м^2 та об’єм 27м^3, що відповідає нормам ДСанПіН 3.3.2-007-98. На робочому місці оператора наявні наступні шкідливі та небезпечні виробничі фактори:

- пожежна небезпека
- освітлення приміщення
- електронебезпека
- шум та вібрації

6.1. Пожежна небезпека

Згідно ПЕУ-2017 кімната оператора відноситься до приміщень без підвищеної небезпеки.

Для забезпечення пожежної безпеки проектом передбачена система зв'язка й оповіщення, системи порошкового та пінного пожежогасіння, система протипожежного водопроводу, а також первинні засоби пожежогасіння.

До первинних засобів пожежогасіння відносяться: вогнегасники, пожежний інвентар (бочки з водою, пожежні відра, ящики з піском, совкові лопати, протипожежні покривала) та пожежний інструмент (гаки, ломи, сокири).

Основними причинами виникнення пожежі можуть бути:

– порушення елементарних правил пожежної безпеки;

– несправність електроустаткування, електромереж;

– порушення електротехнічних правил.

Площа приміщення, де проходить проектування робота становить 9 м^2. Для гасіння пожежі служать вуглекислотні вогнегасники типу ВП-3(3). За формулою, де 1 кг сухої суміші витрачається на площу у 25 м^2. У робочому приміщенні передбачено 1 вогнегасник, який розташований біля входу у приміщення.

Вогнегасники та пожежний інвентар мають червоне пофарбування, а бочки з водою та ящики з піском ще й відповідні написи білою фарбою.

Пожежний інструмент фарбується в чорний колір. Бочки для зберігання води з метою пожежогасіння встановлюються підвальному приміщенні. Такі бочки повинні бути укомплектовані пожежним відром місткістю не менше 8 л. Ящики з піском місткістю 0,5, 1,0 та 3,0 м3 та повинні бути укомплектовані совковою лопатою. Протипожежні покривала, виготовлені з негорючого теплоізоляційного полотна, грубо бавовняної тканини повинні мати не менш як 2х1 м та 2х2 м.

При пожежі в приміщенні знаходяться два евакуовиходи (двері) розміром 2,5х1,3 м. Протипожежний водопровід забезпечує роботу зрошувальних систем колонних апаратів і резервуарів з підключенням для пересувної пожежної техніки.

Для протипожежного захисту насосної використовується система порошкового пожежегасіння – модулі порошкового пожежегасіння у вибухозахищеному виконанні, згідно ДБН В.2.5-56:2014.

Зовнішнє пожежогасіння виробничого будинку здійснюється від пожежних гідрантів існуючої кільцевої протипожежної мережі.

Для гасіння невеликих вогнищ запалень при вимкненому електроустаткуванні застосовують вуглекислотні вогнегасники ОУ-5 та пінні ОХП-10. Для гасіння ввімкнених електромереж застосовують порошкові вогнегасники з речовинами ОПС-10 і ОППС-100. В приміщенні встановлений пожежний гідрант з рукавом 10 м.

6.2. Освітлення приміщення

Шкідливим для оператора є недостатня освітленість робочого місця, це проявляється в погіршенні зору та зниженні продуктивності праці самого оператора.

Для створення світлового комфорту на підприємствах хімічного виробництва використовують: природне освітлення, штучне освітлення, суміщене освітлення.

Для меншого навантаження очей оператора у робочому приміщені будуть використані лампи холодного світла. Загальна площина робочого приміщення оператора складає 9 $м^2$, для того щоб воно відповідало нормам освітленості були обрані світлодіодні лампи з потужністю 11 Вт. Оптимальна освітленість складає 200 Лк , яку забезпечує 2 лампи.

Згідно ДБН В.2.5-28:2018, робота з обслуговування обладнання відноситься до VI розділу підрозділу “а”, тобто загальне спостереження за технологічним процесом. При цьому робоче місце оператора повинно мати освітленість робочої зони Енор=200 лк.

6.3. Електронебезпека

Приміщення операторської відноситься до приміщень з підвищеною небезпекою по ступеню враження електричним струмом, відповідно до ПУЕ 2017 (глава 1.1, пункт 1.1.13.б). Не допускається включення електрообладнання при несправному заземленні.

У операторській встановлена мережа, що працює під напругою 220/380 В, частотою 50 Гц. Мережа з глухозаземленною нейтраллю. Основні причини нещасного випадку від впливу електричного струму наступні:

1) ушкодження струмопровідних ліній електрокабелів, порушення ізоляції і заземлення щитів, пультів і електроустаткування;

2) порушення правил електронебезпечності при експлуатації електричного устаткування і освітлення (спроби самовільного усунення несправностей, заміни світильників);

3) робота на несправному устаткуванні;

4) дотик до відкритих проводок струмоведучих частин;

5) пробій на установці (напруга дотику);

6) крокова напруга;

7) електрична дуга.

До заходів щодо захисту від поразки електричним струмом відносяться:

1. Ізоляція в електроустановках

Ізоляція - шар діелектрика, яким покривають поверхню струмоведучих елементів, або конструкція з непровідного матеріалу, за допомогою якого струмоведучі частини відокремлюються від інших частин електрообладнання. Встановлена ізоляція таких видів:

– робоча – електрична ізоляція струмоведучих частин електроустановки, що забезпечує її нормальну роботу і захист від поразки електричним струмом;

– додаткова – електрична ізоляція, передбачена додатково до робочої ізоляції для захисту від ураження електричним струмом в разі ушкодження робочої ізоляції;

– подвійна – ізоляція, яка складається з робочої і додаткової ізоляції;

– посилена – поліпшена робоча ізоляція, яка забезпечує такий же захист від ураження електричним струмом, як і подвійна ізоляція;

– опір ізоляції 0.5 МОм.

2. Блокування безпеки;

3. Малі напруги на переносні прилади U=42В;

4. Недосяжність проводів;

5. Орієнтація в електричних установках.

В аварійному режимі – занулення з автоматичним відключенням. Електронебезпека на виробництві відповідає ГОСТ 30331.3-95.

6.4. Шум та вібрації

Джерелом шуму та вібрацій під час роботи оператора є шум від робочого комп’ютера, який складає від 30 до 50 дБА. Фактичні показники шуму - L = 30…60 дБА, що відповідає ДСН 3.36.037-99.

Для зменшення виробничого шуму передбачено проведення наступних заходів:

- Встановлення захистних екранів, (ΔL = 15 дБА);
- Робота у захистних ізольованих навушниках;
- Своєчасний ремонт всіх механічних вузлів за регламентом(ΔL = 8 дБА).

ВИСНОВКИ

При виконанні дипломного проекту було проведено аналіз існуючих автоматизованих систем керування та управління доступом, методів детектування обличчя та розпізнавання особистості. На основі проведеного аналізу було зроблено:

1. Розроблено алгоритм для запропонованої АСКУД.
2. Зібрано базу зображень облич людей розміром більше ніж 11000 зображень.
3. Створено свою штучну нейронну мережу для розпізнавання маски на обличчі людини
4. Створено свою автоматизовану систему контролю та управління доступом та Telegram бот до неї за допомогою мови програмування Python
5. Визначені необхідні умови для забезпечення охорони праці під час експлуатації системи

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Поняття АСКУД [Електронний ресурс] https://deps.ua – 2021. – Режим доступу до ресурсу: https://deps.ua/knowegable-base-ru/spravochnaya-informatsiya/7819.html
2. Филиппов А. А. «СИСТЕМЫ КОНТРОЛЯ И УПРАВЛЕНИЯ ДОСТУПОМ» / А. А. Филиппов // – Хабаровск: ГОУ ВПО «Дальневосточный государственный университет путей сообщения», 2010. – 20 с.
3. Understanding physical access control systems [Електронний ресурс] https://www.openpath.com – 2021. – Режим доступу до ресурсу: https://www.openpath.com/blog-post/physical-access-control
4. Системы контроля и управления доступом [Електронний ресурс] https://studbooks.net – 2021. – Режим доступу до ресурсу: https://studbooks.net/1166180/informatika/sistemy_kontrolya_upravleniya_dostupom
5. Захист проти вірусів [Електронний ресурс] https://www.dw.com – 2021. – Режим доступу до ресурсу: https://www.dw.com/uk/як-захиститися-від-коронавірусу-практичні-поради/a-52328126
6. Комп’ютерний зір [Електронний ресурс] https://www.reg.ru – 2021. – Режим доступу до ресурсу: https://www.reg.ru/blog/stenfordskij-kurs-lekciya-1-vvedenie/
7. Computer vision [Електронний ресурс] https://towardsdatascience.com – 2021. – Режим доступу до ресурсу: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e
8. Татаренков Д. А. «АНАЛИЗ МЕТОДОВ ОБНАРУЖЕНИЯ ЛИЦ НА ИЗОБРАЖЕНИИ» / Д. А. Татаренков // Молодой ученый №4, 2015. – 106 с.

9. Color Space Conversion [Електронний ресурс] https://www.dynamsoft.com – 2021. – Режим доступу до ресурсу: https://www.dynamsoft.com/blog/insights/image-processing/image-processing-101-color-space-conversion/
10. Как работает нейронная сеть [Електронний ресурс] https://neurohive.io – 2021. – Режим доступу до ресурсу: https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri/
11. Мова програмування Python [Електронний ресурс] https://blog.ithillel.ua – 2021. – Режим доступу до ресурсу: https://blog.ithillel.ua/ua/articles/perevagi-i-nedoliki-movi-python
12. About NumPy [Електронний ресурс] https://numpy.org – 2021. – Режим доступу до ресурсу: https://numpy.org/about/
13. NumPy [Електронний ресурс] https://en.wikipedia.org – 2021. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/NumPy
14. OpenCV [Електронний ресурс] https://en.wikipedia.org – 2021. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/OpenCV
15. Введение в машинное обучение с TensorFlow [Електронний ресурс] https://habr.com – 2021. – Режим доступу до ресурсу: https://habr.com/ru/post/326650/
16. TensorFlow [Електронний ресурс] https://www.tensorflow.org– 2021. – Режим доступу до ресурсу: https://www.tensorflow.org/learn
17. Keras [Електронний ресурс] https://en.wikipedia.org – 2021. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Keras
18. Создание нейросети с помощью Keras [Електронний ресурс] https://neurohive.io – 2021. – Режим доступу до ресурсу: https://neurohive.io/ru/tutorial/nejronnaya-set-keras-python/

19. Matplotlib [Електронний ресурс] https://matplotlib.org – 2021. – Режим доступу до ресурсу: https://neurohive.io/ru/tutorial/nejronnaya-set-keras-python/
20. Data Visualization with Matplotlib [Електронний ресурс] https://towardsdatascience.com – 2021. – Режим доступу до ресурсу: https://towardsdatascience.com/data-visualization-using-matplotlib-16f1aae5ce70
21. Модуль OS [Електронний ресурс] https://pythonworld.ru – 2021. – Режим доступу до ресурсу: https://pythonworld.ru/moduli/modul-os.html
22. Модуль Pickle [Електронний ресурс] https://pythonworld.ru – 2021. – Режим доступу до ресурсу: https://pythonworld.ru/moduli/modul-pickle.html
23. Модуль Face Recognition [Електронний ресурс] https://github.com – 2021. – Режим доступу до ресурсу: https://github.com/ageitgey/face_recognition
24. Простой Telegram Bot [Електронний ресурс] https://habr.com – 2021. – Режим доступу до ресурсу: https://habr.com/ru/post/442800/
25. PyTelegramBotApi документація [Електронний ресурс] https://github.com – 2021. – Режим доступу до ресурсу: https://github.com/eternnoir/pyTelegramBotAPI

main.py:

```
import tensorflow as tf
from detect_mask import detect_mask
from face_detector.face_detector import detect_face,
borders_and_text
from person_recognition.recognize_person import recognize_person
from person_recognition.send_telegram_notification import
send_telegram_notification
from utils import load_from_database
import cv2 as cv
import os
import pickle
import time
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# Подгружаем модель
model = tf.keras.models.load_model('model.model')

# Интервал отправления сообщений
SENDING_DELAY = 30

# Определитель личности
recognizer = pickle.load(open('./person_recognition/dataset-
encodings.pickle', 'rb'))
# le = pickle.load(open('./person_recognition/le.pickle', 'rb'))

DATABASE = load_from_database('./database.json')

video = cv.VideoCapture(0)

PERSONS = [
    {
        'lastSent': 0
    },
    {
        'lastSent': 0
    },
    {
        'lastSent': 0
    }
]


while True:

    ret, frame = video.read()
```

```
    # Находим лицо
    faces = detect_face(image=frame)

    if len(faces) > 0:  # Если хотя бы одно лицо найдено
        for face in faces:
            start = time.time()
            name = 'Unknown'

            # Определение маски и вывод на экран
            (mask_condition,confidence,_)=
detect_mask(image=frame, model=model, face=face)

            if mask_condition != 0:  # Если человек носит маску
неправильно или снял
                person_id, name = recognize_person(recognizer,
image=frame, face=face)

                # Отправка сообщения
                if (time.time() -
float(DATABASE[str(person_id)]['lastSent'])) > SENDING_DELAY:
                    if person_id is not None and name !=
'Unknown':  # Если человек известен
                        print('sending message to telegram...')
                        DATABASE =
load_from_database('./database.json')
                        DATABASE[str(person_id)]['lastSent'] =
str(time.time())

send_telegram_notification(person_id=person_id,
                                                    photo=frame,
                                                    face=face,
                                                    name=name,

chat_id=DATABASE[str(person_id)]['telegram_id'])

            # Отрисовка рамки и текст
            borders_and_text(image=frame, face=face,
index=mask_condition, confidence=confidence, name=name)

    cv.imshow('superpooper', frame)

    if cv.waitKey(1) & 0xFF == ord('q'):
        break

video.release()
cv.destroyAllWindows()
```

detect_mask.py:

```
import numpy as np
import tensorflow as tf
IMAGE_SIZE = 150

CATEGORIES = ['With mask', 'Without mask', 'Wrong worn']


def detect_mask(image, model, face):

    (startX, startY, endX, endY) = face

    # Выделяем именно то место на кадре, где есть лицо
    gray = tf.image.rgb_to_grayscale(image)
    face = gray[startY:endY, startX:endX]
    face = tf.image.resize(face, [IMAGE_SIZE, IMAGE_SIZE])
    face = np.array([np.array(face)], np.float32)
    face = face / 255

    # Проверяем есть ли маска
    predictions = model.predict(face)
    prediction = np.max(predictions)  # Самое большое значение
    prediction = round(prediction * 100, 2)  # Привожу в формат от
(0-100)%
    index = np.argmax(predictions)  # Получаю индекс класса, где
самый большая вероятность

    # Чисто для консоли
    with_mask = round(predictions[0][0] * 100, 2)
    without_mask = round(predictions[0][1] * 100, 2)
    wrong_worn = round(predictions[0][2] * 100, 2)
    #       print(with_mask,       without_mask,       wrong_worn,
CATEGORIES[int(index)])

    return index, prediction, predictions[0]
```

image_preprocessing.py:

```
from random import shuffle
import pickle
import numpy as np
import os
import tensorflow as tf
import time

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

start = time.time()
```

```
TRAINING_DIRECTORY = r'../dataset/training'
TEST_DIRECTORY = r'../dataset/test'
CATEGORIES = ['with_mask', 'without_mask', 'wrong_worn']
CATEGORIES_RUS = ['в масках', 'без масок', 'неправильно
надетых']
IMAGE_SIZE = 150


def create_dataset(directory, categories, image_size, name):
    dataset = []  # Общая база
    data = []  # Массив для данных по изображениям
    labels = []  # Массив для данных по меткам

    for category in categories:
        path = os.path.join(directory, category)  # путь к
каждой из папок (/dataset/without_mask & /dataset/with_mask)
        label = categories.index(category)

        print(f'[INFO] Загрузка фото
{CATEGORIES_RUS[CATEGORIES.index(category)]}...')
        for image in os.listdir(path):
            try:
                img =
tf.keras.preprocessing.image.load_img(os.path.join(path, image))
# Подгружаем изображения
                img_array =
tf.keras.preprocessing.image.img_to_array(img)

                # Переводим в оттенки серого
                gray = tf.image.rgb_to_grayscale(img_array)

                # Приводим все изображения к одному размеру
                resized_image = tf.image.resize(gray,
[image_size, image_size])
                normalized = resized_image / 255.0
                dataset.append([normalized, label])  #
Записываем фото и метку в массив
            except Exception:
                pass

    # -----------------------------------------------------------
-----------------------------------------------------
    # -----------------------------------------------------------
-----------------------------------------------------
    # Перемешаем между собой изображения с масками и без, чтобы
не вырабатывалось "предрасположенности"
    print('[INFO] Перемешиваем между собой фото в масках и
без...')
    shuffle(dataset)
```

```
    # ------------------------------------------------------------
------------------------------------------------------
    # ------------------------------------------------------------
------------------------------------------------------
    # Отделим данные от меток
    for img, lab in dataset:
        data.append(img)
        labels.append(lab)

    # Перевод типа list в numpy.array
    data = np.array(data, np.float32)
    labels = np.array(labels)
    # ------------------------------------------------------------
------------------------------------------------------
    # ------------------------------------------------------------
------------------------------------------------------
    # НОРМАЛИЗАЦИЯ
    print('[INFO] Нормализируем изображения...')
    # Приводим данные внутри изображений от [0, 255] к [0, 1]
    # data = data / 255.0
    # ------------------------------------------------------------
------------------------------------------------------
    # ------------------------------------------------------------
------------------------------------------------------
    # Сохраняем всё по файлам
    print(f'Сохранение файлов с {"тренировочными" if name ==
"training" else "тестовыми"} фото и метками...' + '\n')
    pickle.dump(data, open(f'{name}-data.pickle', 'wb'))
    pickle.dump(labels, open(f'{name}-labels.pickle', 'wb'))
    # ------------------------------------------------------------
------------------------------------------------------
    # ------------------------------------------------------------
------------------------------------------------------


# Подготовка тренировочного набора
print('[INFO] Подготовка тренировочного набора...')
create_dataset(directory=TRAINING_DIRECTORY,
categories=CATEGORIES, image_size=IMAGE_SIZE, name='training')
'''
# Подготовка тестового набора
print('[INFO] Подготовка тестового набора...')
create_dataset(directory=TEST_DIRECTORY, categories=CATEGORIES,
image_size=IMAGE_SIZE, name='test')
'''
print(f'Время подготовки изображений - {time.time() - start}')
```

training.py:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Conv2D,
MaxPooling2D, Activation, Dropout
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import pickle
import os
import time
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

start = time.time()

MODEL_DIR_AND_NAME = '../model'
BATCH_SIZE = 32  # По сколько фото сразу будет подаваться на
тренировку
EPOCHS = 10  # Количество эпох
VAL_SPLIT = 0.2  # Какой процент выделяется на валиадцию (20%)
FILTERS = 32  # Количество фильтров в сверточном слое
CORE = (5, 5)  # Размер ядра в сверточном слое
POOL_SIZE = (2, 2)

# Пути к файлам
TRAINING_DATA = '../image_preprocessing/training-data.pickle'  #
Тренировчные фото
TRAINING_LABELS = '../image_preprocessing/training-
labels.pickle'  # Тренировчные метки
# TEST_DATA = '../image_preprocessing/test-data.pickle'  #
Тестовые фото
# TEST_LABELS = '../image_preprocessing/test-labels.pickle'  #
Тестовые метки

# Подгружаем тренировочные изображения и метки
print('[INFO] Загрузка тренировчных фото и меток...')
training_data = pickle.load(open(TRAINING_DATA, 'rb'))
training_labels = pickle.load(open(TRAINING_LABELS, 'rb'))

training_labels = to_categorical(training_labels, num_classes=3)

# Подгружаем тестовые изображения и метки
print('[INFO] Загрузка тестовых фото и меток...')
# test_data = pickle.load(open(TEST_DATA, 'rb'))
# test_labels = pickle.load(open(TEST_LABELS, 'rb'))

INPUT_SHAPE = training_data.shape[1:]  # Размер входного
изображения (150, 150, 1)

# Создаем модель
print('[INFO] Создание модели сети...')
```

```
model = Sequential()

model.add(Conv2D(64, CORE, input_shape=INPUT_SHAPE))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=POOL_SIZE))

model.add(Conv2D(64, CORE))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=POOL_SIZE))

model.add(Conv2D(64, CORE))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=POOL_SIZE))


model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(3))
model.add(Activation('softmax'))

# Компиляция модели
print('[INFO] Компиляция...')
model.compile(optimizer='adam',
              loss='categorical_crossentropy',  #
sparse_categorical_crossentropy
              metrics=['accuracy'])

# Тренировка модели
print('[INFO] Тренировка...')
history = model.fit(training_data,
                    training_labels,
                    batch_size=BATCH_SIZE,
                    epochs=EPOCHS,
                    validation_split=VAL_SPLIT)

# Проверка модели на тестовом наборе
# print('[INFO] Проверка на тестовом наборе...')
# AMOUNT_OF_TEST_PHOTOS = test_data.shape[0]  # Количество фото
в тестовом наборе
# test_loss, test_accuracy = model.evaluate(test_data,
test_labels, steps=math.ceil(AMOUNT_OF_TEST_PHOTOS /
BATCH_SIZE))
# print(f"Точность на тестовом наборе данных: ", test_accuracy)

# Отображение статистики тренировок
print('[INFO] Сохраняем данные по обучению...')
for key in history.history:
```

```
    plt.xlabel('Номер тренировки')
    plt.ylabel('')
    plt.plot(history.history[key], label=str(key))

plt.legend()
plt.savefig('history.png')
print(history.history)

# Сохраняем модель
print('[INFO] Сохраняем модель...')
model.save(f'{MODEL_DIR_AND_NAME}.model', save_format='h5')

print(f'Время подготовки изображений - {time.time() - start}')
```

face_detector.py:

```
import cv2 as cv
import numpy as np


def detect_face(image):

    # Подгружаем модель для определения лица
    proto_txt_path =
r"/Users/superpooper/Desktop/Python/diplom/face_detector/deploy.
prototxt"
    weights_path =
r"/Users/superpooper/Desktop/Python/diplom/face_detector/res10_3
00x300_ssd_iter_140000.caffemodel"
    detector = cv.dnn.readNetFromCaffe(proto_txt_path,
weights_path)

    (h, w) = image.shape[:2]  # Получаем размеры кадра
    blob = cv.dnn.blobFromImage(image, 1.0, (150, 150), (104.0,
177.0, 123.0))  # Преобразуем кадр в блоб

    # Находим лицо на кадре
    detector.setInput(blob)
    detections = detector.forward()

    faces_coordinates = []

    for i in range(0, detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence > 0.5:
            # Находим координаты прямоугольника, который обводит
лицо
```

```
            box = detections[0, 0, i, 3:7] * np.array([w, h, w,
h])
            (startX, startY, endX, endY) = box.astype("int")

            # Ограничиваем размер рамки, чтобы не выходила за
пределы кадра
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            faces_coordinates.append((startX, startY, endX,
endY))

    return faces_coordinates


def borders_and_text(image, face, index, confidence, name=None):

    categories = ['With mask', 'Without mask', 'Wrong worn']

    (startX, startY, endX, endY) = face

    # Colors
    colors = {
        'with_mask': (0, 255, 0),
        'without_mask': (0, 0, 255),
        'wrong_worn': (0, 179, 255)
    }

    # Отрисовка рамки
    color = colors['with_mask'] if index == 0 else
colors['without_mask'] if index == 1 else colors['wrong_worn']
    cv.rectangle(image, (startX, startY), (endX, endY), color,
2)  # Рамка для лица
    # cv.rectangle(image, (startX - 1, endY + 50), (endX + 1,
endY), color, -1)

    # Текст
    text = f'{categories[index]} {str(confidence)}%'
    y1 = endY - 10  # if startY - 10 > 10 else startY + 10
    y2 = endY - 30  # if startY - 30 > 30 else startY + 30
    cv.putText(image, text, (startX + 10, y1),
cv.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
    if index != 0:
        cv.putText(image, f'Name: {name}', (startX + 10, y2),
cv.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
```

create_dataset_encodings.py:

```
import cv2 as cv
import os
import pickle
import time
import face_recognition

start = time.time()

path_to_embedder =
'/Users/superpooper/Desktop/Python/diplom/person_recognition/ope
nface_nn4.small2.v1.t7'
embedder = cv.dnn.readNetFromTorch(path_to_embedder)  # Объект,
который будет создавать 128-d вектор
directory = './dataset/'  # Папка со всеми личностями

# Листы для записи в файл
data = {
        'ids': [],
        'encodings': [],
        'names': []
    }

for person in os.listdir(directory):
    if not person.startswith('.'):
        # Имя каждой папки = id-Surname Name
        person_id = int(person.split('-')[0])  # Получаем ID из
названия папки
        person_name = person.split('-')[1]  # Получаем ИМЯ из
названия папки
        path_to_person_folder = os.path.join(directory, person)
# Полный путь к папке

        for image in os.listdir(path_to_person_folder):
            if not image.startswith('.'):
                path_to_img =
os.path.join(path_to_person_folder, image)  # Полный путь к
фотографии
                img = cv.imread(path_to_img)  # Подгружаем
изображение
                img = cv.cvtColor(img, cv.COLOR_BGR2RGB)

                faces = face_recognition.face_locations(img,
model='hog')
                encodings = face_recognition.face_encodings(img,
faces)

                for encoding in encodings:
                    data['ids'].append(person_id)
```

```
                    data['encodings'].append(encoding)
                    data['names'].append(person_name)


# Сохраняем данные в файл с названием 'dataset-encodings.pickle'
pickle.dump(data, open(f'dataset-encodings.pickle', 'wb'))

print(f'Время: {round(time.time() - start, 2)} сек.')
```

recognize_person.py:

```
import cv2 as cv
import face_recognition
import imutils


def recognize_person(recognizer, image, face):

    roi = image[face[1]:face[3], face[0]:face[2]]
    rgb = cv.cvtColor(roi, cv.COLOR_BGR2RGB)
    resized = imutils.resize(rgb, width=250)

    faces = face_recognition.face_locations(resized,
model='hog')
    encodings = face_recognition.face_encodings(resized, faces)
    person_id = int()
    name = 'Unknown'

    for encoding in encodings:
        matches =
face_recognition.compare_faces(recognizer['encodings'],
encoding)

        if True in matches:

            matched_idxs = [i for (i, b) in enumerate(matches)
if b]
            counts = {}

            for i in matched_idxs:
                person_id = recognizer['ids'][i]
                name = recognizer["names"][i]
                counts[name] = counts.get(name, 0) + 1

            name = max(counts, key=counts.get)

    return person_id, name
```

send_telegram_notification.py:

```
import requests
from confing import API_LINK
import time


def send_telegram_notification(person_id, photo, face, name,
chat_id):
    try:
        image = cv.cvtColor(photo, cv.COLOR_BGR2RGB)  # BGR to
RGB

        # Resize frame to face with some offset
        offset = 50
        left = min(face[0], face[0] - offset)
        top = min(face[1], face[1] - offset)
        right = max(face[2], face[2] + offset)
        bottom = max(face[3], face[3] + offset)
        face_frame = image[top:bottom, left:right]

        # Create file from numpy array
        bio = BytesIO()
        bio.name = 'image.jpeg'
        Image.fromarray(face_frame).save(bio, 'JPEG')
        bio.seek(0)

        name = name  # PERSONS[person_id]['name']
        chat_id = chat_id  # PERSONS[person_id]['telegram_id']

        text = f'{name}, не порушуйте карантинний режим!\n' \
               f'Дата: {date.today()} {time.strftime("%H:%M:%S",
time.localtime())}'

        # bot.send_photo(chat_id=chat_id, photo=bio,
caption=text)
        params = {
            'chat_id': chat_id,
            'caption': text
        }
        files = {
            'photo': bio
        }
        requests.post(API_LINK + f'/sendPhoto', params,
files=files)
    except Exception as e:
        pass
```

main_bot.py:

```
from confing import BOT_TOKEN
import telebot
from telebot.types import Message, ReplyKeyboardMarkup,
KeyboardButton, ReplyKeyboardRemove
from utils import save_to_database, load_from_database


DATABASE = load_from_database('../database.json')

bot = telebot.TeleBot(BOT_TOKEN)


# Обработка запуска бота
@bot.message_handler(commands=['start'])
def start(message: Message):
    global DATABASE
    DATABASE = load_from_database('../database.json')

    user_found = False
    for person_id, person in DATABASE.items():
        if person['telegram_id'] == message.from_user.id:  #
Если человек найден
            user_found = True
            text = f'Ласкаво просимо, {person["name"]}'
            bot.send_message(chat_id=message.from_user.id,
text=text)
            break

    if user_found is False:
        text = f'Ласкаво просимо до боту
{bot.get_me().first_name}. ' \
               'Для верифікації потрібен ваш номер телефону.
Натисніть будь ласка на кнопку "Надати номер телефону" ' \
               '(Якщо вона зникла, відкрийте клавіатуру).'
        markup = ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
        markup.add(KeyboardButton(text='Надати номер телефону',
request_contact=True))
        bot.send_message(chat_id=message.from_user.id,
text=text, reply_markup=markup)
        bot.register_next_step_handler(message, register_user)


def register_user(message: Message):
    global DATABASE
    user_found = False
    if message.content_type == 'contact':
```

```
        phone = message.contact.phone_number
        phone = phone if phone.startswith('+') else ('+' +
phone)
        for person_id, person in DATABASE.items():
            if person['phone'] == phone:  # Если в базе есть
такой номер
                user_found = True
                person['telegram_id'] = message.from_user.id  #
Сохраняем chat_id в базу
                save_to_database('../database.json', DATABASE)
                DATABASE =
load_from_database('../database.json')
                text = f'Ласкаво просимо, {person["name"]}!'
                bot.send_message(chat_id=message.from_user.id,
text=text)
                break

        if user_found is False:
            text = 'Ви не є співробітником або ввели
неправильний номер! Спробуйте ще раз!'
            bot.send_message(chat_id=message.from_user.id,
text=text)
            bot.register_next_step_handler(message,
register_user)

    if message.content_type == 'text':
        markup = ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
        markup.add(KeyboardButton(text='Надати номер телефону',
request_contact=True))
        bot.send_message(chat_id=message.from_user.id,
                         text='Натисніть будь ласка на кнопку
"Надати номер телефону" '
                              '(Якщо вона зникла, відкрийте
клавіатуру)',
                         reply_markup=markup)
        bot.register_next_step_handler(message, register_user)


bot.polling(none_stop=True, interval=0)
```