МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СИСТЕМА РОЗПІЗНАВАННЯ НАЯВНОСТІ МАСКИ НА ОБЛИЧЧІ ЛЮДИНИ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

Спеціальність 122 «Комп'ютерні наук»

122 – БКР – 401.21810121

Виконав студент 4-го курсу, групи 401
_____***М. О. Салютін***
« »_____ 2022 р.

Керівник: д-р техн. наук, професор
_____***Ю. П. Кондратенко***
« »_____ 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2021р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Салютіну Максиму Олександровичу.

1. Тема кваліфікаційної роботи «Система розпізнавання наявності маски на обличчі людини з використанням нейронних мереж».

Керівник роботи Кондратенко Юрій Пантелійович, д-р техн. наук, професор.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: вхідне (оригінальне) зображення з вебкамери, нейронні мережі для розпізнавання об'єктів на зображенні.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз поняття системи розпізнавання маски на обличчі людини та порівняльний аналіз наявних систем;

– огляд бібліотеки TensorFlow, Keras та OpenCV: історія виникнення, основні принципи роботи, необхідні інструменти бібліотек, опис модулів та компонентів;

– огляд архітектур нейронних мереж, методів навчання;

– демонстрація результатів обраної архітектури нейронної мережі на вхідних зображеннях з веб-камери для розв’язання поставленої задачі.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Аналіз вимог до організації заходів техніки безпеки та охорони праці під час епідемії covid-19 та роботи за комп’ютером»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	**Старший викладач Макарова О.В.**	

Керівник роботи __д-р техн. наук, проф. Кондратенко Ю. П._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Салютін М. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «__» _____ 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема : __Система розпізнавання наявності маски на обличчі людини з використанням нейронних мереж_____
_____

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	26.10.2021	26.10.2021	Виконано
2	Отримання завдання на виконання БКР	21.11.2021	21.11.2021	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	05.12.2021	09.12.2021	Виконано
4	Отримання завдання на переддипломну практику	20.05.2022	20.05.2022	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	23.05.2022	04.06.2022	Виконано
6	Розробка звіту з переддипломної практики	04.06.2022	06.06.2022	Виконано
7	Виконання БКР: аналіз сучасного стану задачі відслідковування очей, огляд існуючих технологій, розробка ПЗ	28.02.2022 та 06.06.2022	27.03.2022 та 19.06.2022	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	31.05.2022	31.05.2022	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2022	20.06.2022	Виконано
10	Подання БКР рецензенту	16.06.2022	18.06.2022	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	20.06.2022	22.06.2022	Виконано
12	Захист БКР перед екзаменаційною комісією	28.06.2022	28.06.2022	Виконано

Розробив студент _____Салютін М. О_____ _____
(прізвище та ініціали) (*підпис*)
Керівник роботи _д-р техн. наук, проф. Кондратенко Ю. П._ _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента 401 групи
ЧНУ ім. Петра Могили
Салютіна Максима Олександровича
Тема: “Система розпізнавання наявності маски на обличчі людини з використанням нейронних мереж”

Об’єктом дослідження є процеси навчання нейронної мережі для відстеження маски на обличчі людини.

Предметом дослідження є методи навчання нейронної мережі, які дозволять розпізнавати маску на обличчі людини.

Мета роботи - автоматизація розпізнавання наявності маски на обличчі людини мовою програмування Python з використанням бібліотек TensorFlow, Keras та OpenCV.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, п’яти розділів та висновків.

У першому розділі було проведено аналіз всесвітньої проблеми, викликану через вірус COVID-19 – її походження та аналіз наявних аналогів систем розпізнавання та постановка задач.

У другому розділі проведено аналіз нейронних мереж, бібліотек TensorFlow, Keras та OpenCV як інструменти для створення системи.

У третьому розділі було встановлено необхідні бібліотеки, обрано архітектуру нейронної мережі та на її основі навчена модель.

У четвертому розділі продемонстровано графічний інтерфейс для системи, застосування нейронних мереж для розв’язання поставленої задачі.

В результаті розроблено інтелектуальну систему мовою програмування Python з використанням бібліотек TensorFlow, Keras та OpenCV.

Бакалаврська кваліфікаційна робота містить __ сторінок, __ рисунків, __ таблиць, __ використаних джерел та __ додатків.

Ключові слова: COVID-19, Python, TensorFlow, OpenCV, Keras, ML.

ABSTRACT

To the bachelor's qualification work by student of group 401 of

Petro Mohyla Black Sea National University

Saliutin Maksym Olexandrovich

"A system for recognizing the presence of a mask on a person's face using neural networks"

The object of the research is to train a neural network to track the mask on a person's face.

The subject of the study is a system that allows you to recognize masks on a person's face.

The aim of this thesis is to development an intelligent system for recognizing the presence of a mask on a person's face in the Python programming language, using the TensorFlow and Keras libraries.

The work consists of a professional section and a special part on labor protection. The explanatory note consists of an introduction, five sections and conclusions to the thesis.

The first section analyzed the worldwide problem caused by the COVID-19 virus - its origins and analysis of the available analogues of recognition systems and problem statement.

The second section is analysis of neural networks, TensorFlow, Keras and OpenCV libraries as tools for creating an intelligent system.

The third section the necessary libraries were installed, the neural network architecture was selected and the model was trained on its basis.

The fourth section demonstrates a graphical interface for the system, the use of neural networks to solve the problem.

As a result, an intelligent system is developed in Python programming language using TensorFlow, Keras and OpenCV libraries.

Thesis contains: __ pages, __ figures, __ tables, __ formulas, __ references.

Keywords: COVID-19, Python, TensorFlow, OpenCV, Keras, ML.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВДТ – візуальний дисплейний термінал

ВООЗ – всесвітня організація охорони здоров’я

ГІК – графічний інтерфейс користувача

ЕОМ – електронна обчислювальна машина

НМ – нейронна мережа

КПО – коефіцієнт природної освітленості

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

COVID-19 – Coronavirus Disease 2019

DL – Deep learning

ML – Machine learning

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«СИСТЕМА РОЗПІЗНАВАННЯ НАЯВНОСТІ МАСКИ НА ОБЛИЧЧІ ЛЮДИНИ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401. 21810121

Виконав студент 4-го курсу, групи 401
_____М. О. Салютін
« »_____ 2022 р.

Керівник: д-р техн. наук, професор
_____Ю. П. Кондратенко
« »_____ 2022 р.

Миколаїв – 2022

ВСТУП

Актуальність теми. Після швидкого поширення нового випадку коронавірусної хвороби (COVID-19) в Ухані, Китай, у грудні 2019 року Всесвітня організація охорони здоров'я (ВООЗ) підтвердила, що це небезпечний вірус, який може передаватися від людини до людини повітряно-краплинним шляхом.

Що стосується профілактики, кожна людина повинна носити маску для обличчя, дотримуватися соціальної дистанції, уникати місць скупчення людей, а також завжди підтримувати імунну систему. Тому, щоб захистити один одного, кожна людина повинна правильно носити захисну медичну маску, коли вона знаходиться у громадському скупченні. Однак деякі безвідповідальні люди відмовляються носити маску незважаючи на усю небезпеку та наслідки вірусної інфекції. Тому розробка розпізнавання наявності маски для обличчя в цьому випадку є дуже важливою.

Пандемія COVID-19 значно впливає на економічний розвиток, спосіб життя населення та систему охорони здоров'я у всіх країнах. З кожним днем статистика захворювань, на жаль, росте через великі скупчення людей у метро, офісах, громадському транспорті та інших громадських місцях.

Відстежувати вручну наявність маски на обличчі людей є складною задачею і потребує великих зусиль та додаткового штату працівників. Тому є актуальним створення системи, що буде відстежувати наявність маски, робити статистику, що допоможе людям у подоланні цієї хвороби та вберегти їхнє життя.

Об’єкт: процеси навчання нейронної мережі для відстеження маски на обличчі людини.

Предмет: методи навчання нейронної мережі, які дозволять розпізнавати маску на обличчі людини.

Мета: автоматизація системи для розпізнавання наявності маски на обличчі людини на основі розробленої моделі нейронних мереж мовою програмування Python з використанням бібліотек TensorFlow та Keras.

Основними результатами даної роботи є розробка системи для розпізнавання наявності маски, яка буде відстежувати обличчя людей і видавати інформацію у реальному часі про її наявність або відсутність. При вимкненні програми буде записуватися статистика про наявність/відсутність маски у базу даних.

Для досягнення поставленої мети необхідно вирішити такі **завдання:**

- проаналізувати сучасний стан задачі розпізнавання наявності маски на обличчі людини;
- проаналізувати існуючі аналоги систем;
- обрати необхідну архітектуру нейронної мережі;
- розробити систему для розпізнавання наявності маски на обличчі за допомогою необхідних бібліотек;
- розробити користувацький графічний інтерфейс, модель бази даних для запису статистики та додатковий функціонал;
- провести тестування модулю;

Практичне значення: інтелектуальна система призначена для використання в метро, держустановах, навчальних закладах, офісах та інших громадських місцях. Розроблена система здійснює розпізнавання та запис статистики про наявність маски на обличчі людини використовуючи нейронні мережі.

1 ЗОВНІШНІ ПЕРЕДУМОВИ ДЛЯ ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ КОНТРОЛЮ ЗАХВОРЮВАНОСТІ COVID-19 З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ НЕЙРОННИХ МЕРЕЖ

1.1 Вірус COVID-19: особливості розповсюдження, вплив на організм, походження

Під час розвитку людської цивілізації людям доводилося стикатися з величезною кількістю природних катастроф, з яких можна перелічити наступні: чума, урагани, цунамі, землетруси та інші. Протягом кількох століть людство не могло знайти причин цих явищ, що призводило до жахливих наслідків.

З розвитком багатьох наукових сфер у людей прийшло розуміння та прорахування подібних катастроф, але розуміння проблеми не завжди може повністю допомогти знайти рішення, тому на даний момент розробляються різні системи, які допомагають завчасно визначити катастрофу та допомогти людям вберегти своє життя.

На сьогоднішній день населення Землі складає більше 7.753 мільярдів людей – що значно більше, ніж було у минулих віках. Зростання кількості та густини населення призводить до різних наслідків, один із яких – збільшується швидкість розповсюдження різних захворювань, які передаються повітряно-крапельним шляхом (грип, коронавірус). Ці захворювання в більшості випадків не призводять до тяжких наслідків, але можуть мутувати, що ускладнює їх виявлення та лікування і може привести до тяжких наслідків.

Гострі респіраторні вірусні інфекції (ГРВІ) - є найпоширенішими захворюваннями серед людей. Респіраторними вірусами викликається понад 500 мільйонів застуд на рік. Респіраторні вірусні інфекції також є найчастішою причиною госпіталізації як серед дітей, так і серед дорослих, а гострий респіраторний синдром, який може виникнути на тлі інфекції, є основною причиною смерті у країнах, що

розвиваються. Поширення вірусів залежить від багатьох факторів: рівня медицини у країні, вікового розподілення населення, густини населення.

У грудні 2019 року, в Китаї підняли ситуацію з пандемією під назвою COVID-19 [1]. Це захворювання, викликане вірусом SARS-CoV-2, і Всесвітня організація охорони здоров'я (ВООЗ) оголосила його глобальною пандемією 11 березня 2020 року. Протягом історії людство також спостерігало кілька пандемічних ситуацій, таких як африканська пандемія ожиріння, пандемічний грип, пандемія ВІЛ/СНІДу тощо. Зараз люди переживають вирішальний час для боротьби з потужним ворогом, а саме з коронавірусом (COVID-19). Через цю смертельну хворобу мільйони людей були інфіковані COVID-19 у всьому світі, і багато людей загинуло після тяжких симптомів даного вірусу. На початку січня 2020 року кілька пацієнтів повідомили медичні заклади про пневмонію в місті Ухань [2], причину якої пов'язували з оптовим ринком морепродуктів в регіоні країни. Швидке поширення епідемії COVID-19 створює серйозні проблеми для боротьби з майбутніми мутаціями вірусу. Згідно з новинами від компанії Google, з грудня 2019 року по теперішній час (по 18 травня 2022 р.), епідемія COVID-19 росла та розвивалася швидкими темпами, яка заразила понад 522 741 989 осіб, а 6 269 178 померли [3].

Таблиця 1.1 – Статистика COVID-19 з відкритих джерел

Worldometer[4]				Google Statistic	
Країни	Підтверджені випадки	Одужало	Загиблих	Підтверджені випадки	Загиблих
Світ	524,246,123	494,172,720	6,292,858	522,731,989	6,269,178
США	84,473,447	81,380,775	1,027,285	82,702,870	999,027
Індія	43,127,199	42,587,259	524,293	43,127,199	524,293
Бразилія	30,728,286	29,724,682	665,277	30,701,900	665,216
Франція	29,233,309	28,335,556	147,568	28,429,331	144,351
Німеччина	25,892,255	24,299,000	138,183	25,890,456	137,888
Великобританія	22,207,102	21,734,193	177,410	22,212,395	177,595

Закінчення таблиці 1.1

Worldometer				Google Statistic	
Країни	Підтверджені випадки	Одужало	Загиблих	Підтверджені випадки	Загиблих
Італія	17,116,550	15,983,655	165,494	17,116,550	165,494
Туреччина	15,057,184	14,955,005	98,911	15,057,184	98,911
Іспанія	12,179,234	11,588,576	105,642	12,179,234	105,642
В'єтнам	10,699,965	9,364,857	43,071	10,699,965	43,071

Декількома групами вчених в усьому світі на базі заснованих міжнародних інститутів та організації з питань дослідження коронавірусної інфекції та її розповсюдження були проведені досліди з метою дослідження як найбільшої кількості фактів та ефективних засобів боротьби з інфекцією. Дослідники намагаються дізнатися про вірус як можна більше і порекомендувати ефективні рекомендації. За даними Worldometer, загальна кількість підтверджених випадків коронавірусу в усьому світі становить 524 246 123, 6 292 858 загиблих , 494 172 720 одужалих. У таблиці 1.1 наведено інформацію, пов’язану з COVID-19, таку як підтверджені, вилікувані та летальні випадки, які були зібрані з ресурсів Worldometer та Google COVID-19 Statistic .

Симптоми COVID-19 можуть призвести до захворювання, яке може варіюватися від доброякісного до надзвичайно небезпечного. Симптоми від захворювання з’являються від 2 до 14 днів після інфікування, що називається інкубаційним періодом. Усіх, хто має симптоми COVID-19, утримують на карантині (в ізольованих від скупчення людей приміщеннях) під активним медичним оглядом. Основними симптомами захворювання є задишка, біль у горлі, кашель, прискорене серцебиття та підвищення температури. Зазвичай вірус може передаватися від однієї людини до іншої повітряно-крапельним шляхом, що утворюються під час кашлю, а також

може передаватися від дотику до нечистих поверхонь, а потім дотику до обличчя (Центри контролю та профілактики захворювань, 2020) [5]. Враховуючи усю небезпеку вірусу COVID-19, ВООЗ порадила кільком країнам підтвердити, що їхні громадяни носять маски в будь-яких громадських місцях, де є високий ризик заразитися цим вірусом [6].

До COVID-19 лише кілька людей носили маски для захисту здоров'я від забрудненого повітря, а також медичні працівники використовували їх під час огляданння пацієнтів з респіраторними хворобами. ВООЗ оголосила це глобальною пандемією через велику кількість зареєстрованих позитивних випадків зараження вірусом, де більшість випадків виявляється в людних місцях. Тому експерти рекомендували носити маску як у людних, так і в будь-яких громадських місцях для запобігання передачі захворювання.

Уряд Франції розпочав ініціативу виявлення пасажирів без маски на станції метро. Для цього алгоритми штучного інтелекту (ШІ) були розгорнуті на камерах відеоспостереження на станціях метро у Парижі [7]. Тестовий запуск зміг зібрати дані для майбутнього аналізу розробниками. Алгоритми ШІ, такі як глибоке навчання (DL) і машинне навчання (ML), можна використовувати кількома способами для запобігання передачі COVID-19.

Однак існує суттєва відмінність між відсутністю доказів і доказами відсутності. Доказів того, що маски для обличчя можуть забезпечити ефективний захист від респіраторних інфекцій у суспільстві мало, як визнається в дослідженнях Великобританії [8]. Проте медичні маски широко використовуються медичними працівниками для забезпечення безпеки при догляді за пацієнтом з респіраторною інфекцією. Було б розумно запропонувати вразливим особам із слабким організмом уникати місць скупчення людей і раціонально використовувати медичні маски, коли вони потрапляють у зони ризику (тобто у великі скупченнях людей). Оскільки дані свідчать про те, що COVID-19 може передаватися до появи симптомів, передача в

громаді може бути зменшена, якщо всі, включаючи людей, які були інфіковані, але без прояви симптомів носитимуть медичні маски.

Виявлення масок на обличчі за допомогою ML та DL досягло багатообіцяючих результатів. Однак, існуючі методи повинні бути більш надійними в середовищі реального часу. Крім того, на даний момент все ще триває відсутність відповідних наборів даних і моделей на основі DL для виявлення маски на обличчі, які могли б виявити людину без маски в режимі реального часу, щоб уникнути поширення COVID-19. З цією метою буде використано набір даних, який складається з двох класів, тобто містить у собі зображення обличчя у масці та без неї для навчання НМ.

1.2 Існуючі системи для розпізнавання наявності маски на обличчі людини

У цей складний час пандемії COVID-19 багато компаній та корпорацій змогли належним чином адаптуватися та розробити цілі системи для збереження здоров'я людей та забезпечення їх безпеки. Вчені та розробники створили автоматизовані системи розпізнавання масок у громадських місцях, щоб забезпечити використання масок у місцях загального скупчення. Після епідемії COVID-19 інші дослідники розробили власні методики відстеження масок для обличчя в місцях загального користування. Серед розробок і стартапів на даний момент можна знайти системи розпізнавання масок на обличчі, які поступово починають набирати популярності і стають затребуваними. На даний момент у країнах знову спостерігаються різкі спалахи цього вірусу, тому реалізація цих систем та методів для боротьби залишається актуальною.

Навіть у нових версіях телефонів компанії Apple, а саме iPhone, можна використовувати функцію Face ID, не знімаючи з обличчя медичну маску (рис. 1.1) [9]. У такий час, коли є високий ризик захворіти – така функція стає дуже корисною та затребуваною, якщо є необхідність скористатися телефоном у громадських місцях.

Компанія Apple розробила нову методику для розпізнавання людини, яка прихована за медичною маскою будь якого кольору. Дана технологія виконує сканування не усього обличчя, а тільки відкриті області навколо очей людини. Інженери компанії стверджують, що дані ділянки обличчя у кожної людини відрізняються, що забезпечує безпеку пристрою якщо він потрапить у чужі руки.

Рисунок 1.1 – Функція розпізнавання маски на телефонах компанії Apple модельного ряду iPhone, починаючи з 11 покоління

Також зараз поступово в місцях скупчень людей з'являються системи розпізнавання маски, для отримання доступу до громадського закладу. Такі системи після загального огляду мають свої недоліки та переваги. Із головних переваг цих систем варто виділити наступні: забезпечує більш безпечне середовище для персоналу та клієнтів, заохочує правильне носіння масок у громадських місцях, усуває конфронтацію, спонукаючи людей маскуватися.

Великий мінус усіх систем та застосунків у тому, що вони мають урізаний функціонал, який варто об’єднати у одну систему. Ці системи потребують багато

коштів для встановлення та налаштування. З цього можна зробити висновок, що на даний момент не існує системи, модулів або застосунків, які б могли окрім точного розпізнавання маски формувати статистику за день, сповіщати адміністрацію або охорону про відсутність маски та працювати без інтернет з'єднання.

Багато проектів отримали фінансування та зараз якщо за запитом у пошуковій системі Google ввести «Система для розпізнавання масок», результат видасть наступний список сайтів з інформацією про системи:

– https://getmaskcheck.com/ – модель планшету компанії Apple iPad 8 із встановленим застосунком MaskCheck на спеціальному кріпленні. Розміщується дана система на вході у заклад, для отримання доступу до громадського закладу. Дану систему можна перепрофілювати як клієнтський кіоск з іншими застосунками (рис. 1.2) [10].

Переваги:

- миттєві повідомлення про наявність або відсутність маски на особі людини;
- легко встановити та налаштувати застосунок на планшеті.

Недоліки:

- висока ціна за систему із-за додаткової покупки iPad 8;
- немає збору статистики за день;
- потребує постійне підключення до WI-FI мережі;
- відсутність конфіденційності, через закритий доступ до роботи програми є ризик несанкціонованого доступу (НСД) до інформації з боку розробника.

– https://safer.work/safe-space/ – застосунок, який може виявляти людські обличчя в масках, використовуючи систему на основі машинного навчання. Для

встановлення можна придбати будь який планшет або телефон із фронтальною камерою та встановити даний пристрій із додатком у громадському місці (рис. 1.3) [11].

Переваги:

- безкоштовний застосунок;
- надає налаштовані екранні повідомлення та звукові сповіщення;
- висока точність: 99,33% у виявленні масок для обличчя.

Недоліки:

- немає збору статистики;
- відсутність конфіденційності;
- тільки для Android та iOS пристроїв.

– https://felenasoft.com/ – програма для відеоспостереження та будь-яких завдань роботи з камерами. Дана система включає у себе великий набір різноманітних камер, які можна встановити в необхідних для контролю приміщеннях. Єдина система Xeoma дозволяє об’єднати усі пристрої до одного ПК, який буде вести запис на жорсткі диски та ідентифікувати людей у масках з високою точністю розпізнавання – 93.45%. Xeoma відрізняється простим зрозумілим інтерфейсом, гнучкими налаштуваннями «під ключ», «живою» підтримкою клієнтів і широким набором функцій, у тому числі відеоаналітики (рис. 1.4) [12].

Переваги:

- ведення статистики, підрахування кількості людей у масках та без неї;
- висока точність розпізнавання маски;
- не потребує постійного інтернет з’єднання.

Недоліки:

- потрібно покупати щомісячну підписку для постійного підтримання та роботи із хмарним сервісом для збереження статистики;
- немає звукового супроводу якщо у людини відсутня маска на обличчі.

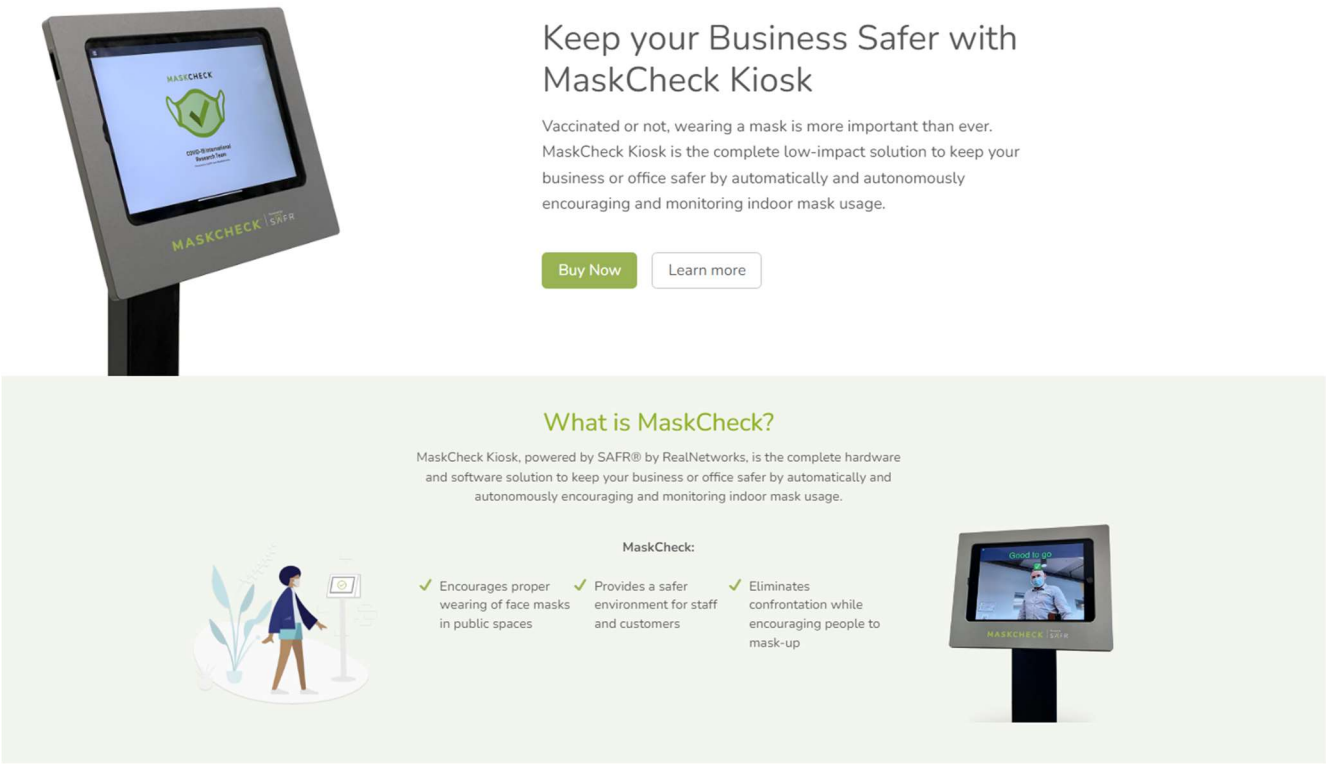

Рисунок 1.2 – Система розпізнавання MaskCheck

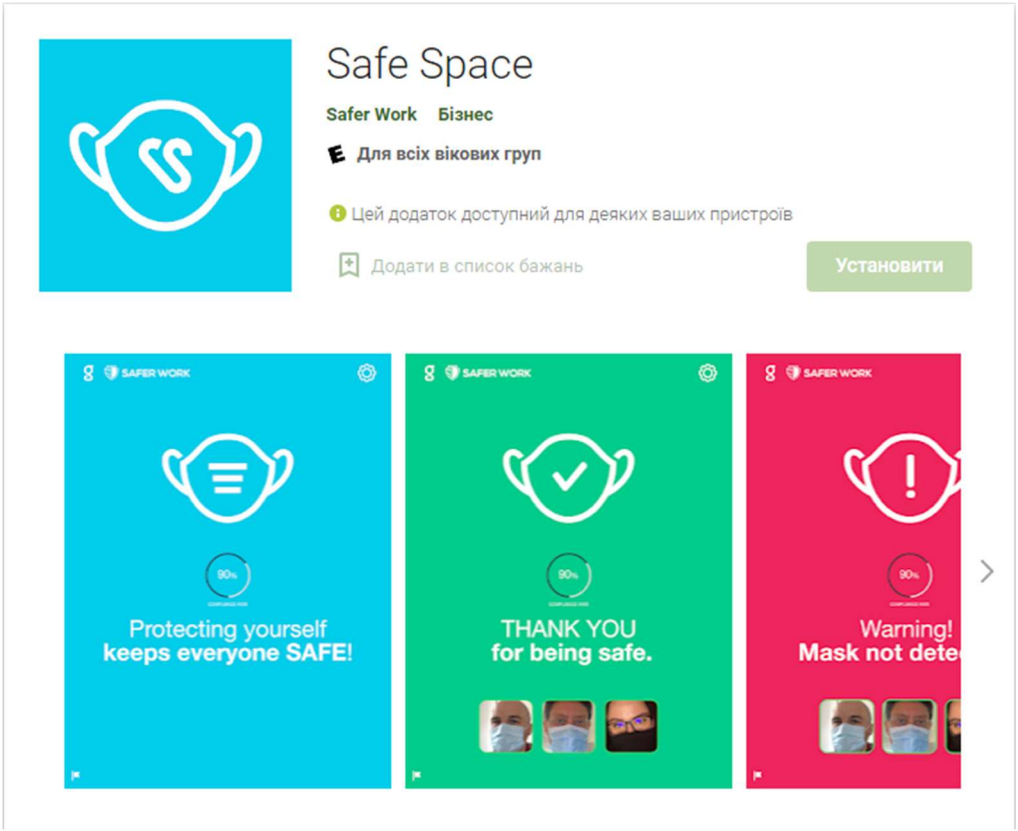

Рисунок 1.3 – Застосунок Safe Space

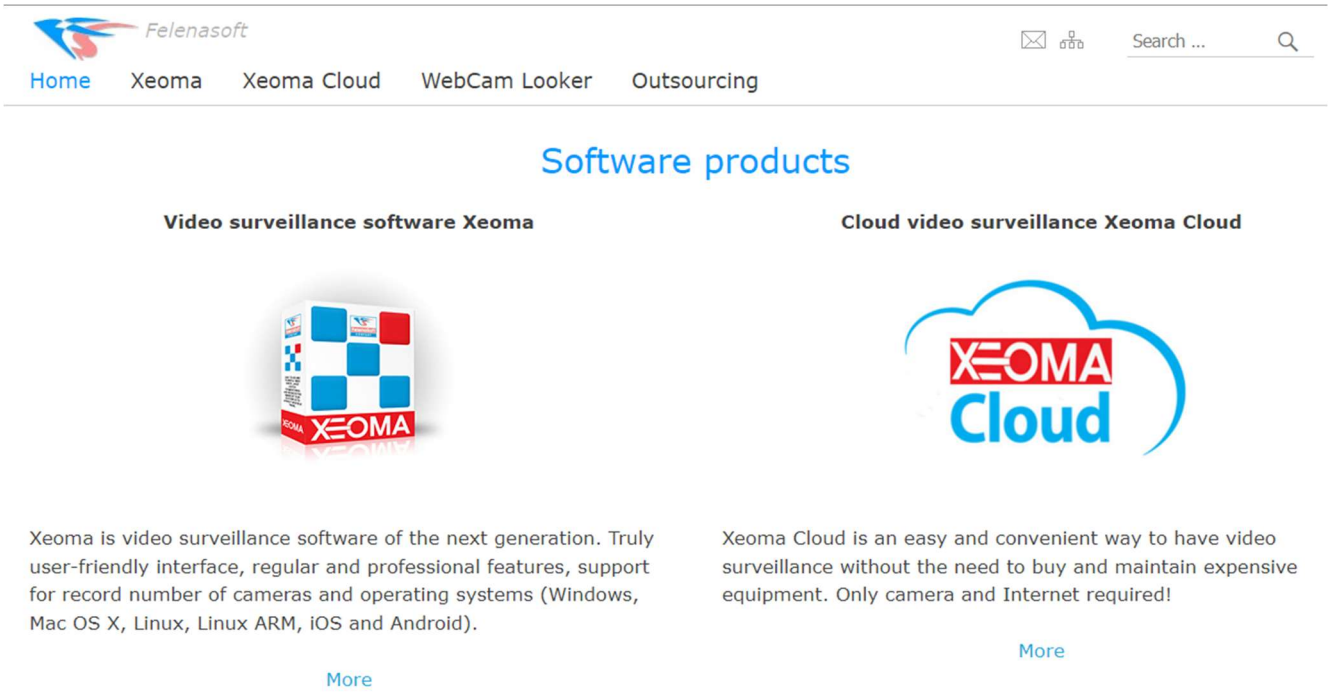

Рисунок 1.4 - Система розпізнавання Xeoma

Отже, передивившись усі вищенаведені системи можна виділити усі переваги та недоліки. Із переваг можна виділити наступне: високий відсоток розпізнавання та автономність систем. Із недоліків: відсутність у кожної системи та додатків певного функціоналу, із-за чого важко підібрати доступну систему для забезпечення безпеки громадян у місцях скупчення людей.

1.3 Вимоги до програмного забезпечення, їх специфікація

Метою роботи є розпізнавання наявності маски на обличчі людини та ведення статистики за рахунок розробки системи з використанням нейронних мереж. Для реалізації ПЗ необхідно ознайомитись та виокремити певні операції покадрової обробки вхідного відеопотоку з вебкамери, а також визначити межі задачі розпізнавання з використанням бібліотек для опрацювання відеопотоку OpenCV та роботи із нейронними мережами TensorFlow та Keras.

Для створення відповідного ПЗ необхідно вирішити наступні завдання:

– обрати необхідну архітектуру для подальшого створення нейронної мережі та забезпечення процесу навчання;

– розглянути функції для обробки відеопотоку з вебкамери, звукового сповіщення та відправлення повідомлень про розпізнавання порушника системою;

– розробити систему, яка містить у собі вищенаведені функції. Передбачається реалізація простого налаштування та підключення усіх необхідних для користувача функцій.

Система повинна відповідати наступним вимогам:

– під час роботи система може оповіщати адміністрацію о вияві людини без маски на обличчі надсилаючи повідомлення у месенджер;

– також система може оповістити людину без маски відповідним звуковим сигналом попередження;

– вести статистику про загальну кількість людей у масці та без неї у БД.

Система має зручний графічний інтерфейс користувача, призначений для людей, які мають базові навички володіння ПК. Процес інсталювання зазначеного ПЗ на кінцевий пристрій користувача, тобто ноутбук або ПК є оптимізованим з точки зору дизайну інтерфейсу та користувацького опиту, що надає можливість легко здійснити процес інсталювання для користувачів, які не мають досвіду у цьому. Параметри системи можуть бути легко налаштовані через спеціалізоване меню. При розробці даного ПЗ було застосовано алгоритми для побудови графічного меню, роботи із відеопотоком для подальшої обробки, навчання нейронної мережі для вирішення задачі розпізнавання. Також було використано технології оптимізації та багатопоточності задля покращення швидкодії ПЗ.

Для реалізації системи розпізнавання наявності маски на обличчі людини була обрана мова програмування Python. Дана мова програмування потребує невеликий поріг входження для вивчення. Завдяки тому що ядро має декілька реалізацій, зокрема: CPython, IronPython, RPython, Jython, PyPy . Мова відноситься до

класу інтерпретованих, строго та динамічно-типізованих – саме це надає можливість для швидкого опанування майбутніми спеціалістами у галузі і робить її підходящим вибором для швидкої розробки прототипу та реалізації програмних продуктів, які можуть бути запущені на різних платформах: починаючи від сімейства мікроконтролерів ARM та IoT та закінчуючи персональними комп'ютерами. Розробка мови програмування здійснюється через пропозиції від інших розробників, які використовують цю мову і хочуть її покращити або в якихось місцях спростити чи навіть виробити правила написання коду. Для цього програмісти придумали PEP (Python Enhancement Proposal), в якому описуються нові функції мови, правила оформлення і так само рекомендації [13].

Python став однією з найпопулярніших мов для використання в аналізі даних, машинному навчанні, DevOps, а також в інших сферах. З головних переваг мови можна виділити такі пункти:

- читабельність;
- простий синтаксис;
- відсутність необхідності в компіляції;
- містить технології для будь-якого завдання;
- легкий у проектуванні концептів та парадигм.

Має об'єктно-орієнтований підхід за бажанням розробника, якщо ні, то є можливість швидко написати скрипт-файл, який має waterfall-компіляцію, що спрощує його розуміння. Програми на Python працюють в будь-яких умовах та підтримують багатоплатформенність, що дозволяє при належній якості коду не переписувати логіку програми, змінюються лише спеціальні системозалежні частини, які особливо відрізняються від інших ОС, наприклад: версії технологій, апаратні частини і так далі.

Задля продуктивної оптимізації процесу написання коду та створення застосунку необхідно забезпечити інструментарій, який дозволить частково автоматизувати пошук синтаксичних помилок у програмному коді, полегшить сприйняття синтаксису мови програмування, допоможе у налагодженні коду та забезпечить загальну оптимізацію робочого процесу. Саме таким інструментом є IDE або інтегроване середовище розробки.

Найбільш популярним середовищем для розробки на мові програмування Python є IDE PyCharm від компанії JetBrains, яке має два типи ліцензування відповідно до потреб команди, яка розробляє відповідний продукт [14]. Дана IDE має всі базові речі необхідні для розробки. Також це інтегроване середовище розробки підходить для будь-якої операційної системи, а саме: Windows, Linux, macOS. Головна задача IDE це забезпечити програміста зручним інтерфейсом та підказками, щоб не зволікати його від таких важливих задач як написання функціоналу для системи.

Необхідні мінімальні характеристики для роботи системи:

- 2 Гб ОЗУ для 64-розрядної системи;
- 2 Гб вільного місця на жорсткому диску;
- Вебкамера;
- Інтернет-з’єднання.

Також для роботи системи необхідна встановлена будь яка із відомих на даний час операційна система, а саме: Windows 8.1-10, Linux або macOS.

Висновки до розділу 1

Незважаючи на всі труднощі, з якими стикається людство - рано чи пізно воно знаходить ефективне рішення для подолання загрози. Тому використання нових інформаційних технологій - це один з багатьох ключів, що може допомогти людям в боротьбі з багатьма проблемами, у тому числі і з новим невідомим вірусом COVID-19.

Система розпізнавання наявності маски на обличчі людини – це той самий інструмент, який можна використовувати для боротьби із вірусом. Оскільки люди з кожним роком дедалі все більше адаптують своє життя під нові правила та рекомендації при пандемії – то впровадження у громадські місця таких систем має великий сенс. Тому що, на жаль, деякі люди продовжують ігнорувати дотримання рекомендацій ВООЗ з приводу необхідності носити медичні маски, через що ставлять себе та інших людей під загрозу заразитись або заразити таким небезпечним захворюванням.

І щоб полегшити перевірку наявності в людини маски - дослідники вирішили створити систему для автоматизації даного процесу без залучення додаткового робочого персоналу.

У першому розділі було розглянуто всесвітню проблему, викликану через вірус COVID-19 – її походження, особливості розповсюдження та вплив на організм. Також було проаналізовано та порівняно схожі за функціоналом системи, включаючи платні та безкоштовні варіанти, а саме: стенд MaskCheck, застосунок SafeSpace та систему відеоспостереження для розпізнавання Xeoma. Деякі системи дуже дорогі, не містять у собі усього потрібного функціоналу для забезпечення ефективної перевірки.

Завдання розробки системи для розпізнавання маски на обличчі людини є дуже актуальним та необхідним у наш час. Тому як це допоможе знизити темп захворювань, забезпечить більш безпечне середовище для персоналу та клієнтів та заохочує правильне носіння масок у громадських місцях. Виходячи із поставлених задач для ПЗ, ця система не буде займати багато місця на серверах із камерою, ПК або мікроконтролерах, матиме прості налаштування та великий функціонал, що дозволить користувачу полегшити інсталювання.

2 ВИБІР АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ. ТЕХНОЛОГІЇ ДЛЯ РЕАЛІЗАЦІЇ ТА ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

Штучні нейромережі – це математична модель функціонування нейронних мереж, яка традиційна для живих організмів, а саме що містять мережі нервових клітин. Що й у біологічному оригіналі, що у штучних мережах основним елементом виступають ті самі нейрони, які з'єднані між собою синапсами. Ці нейрони об'єднуються у шари, число яких може змінюватись в залежності від складності нейромережі та її призначення, а точніше від завдання, яке їй потрібно вирішити.

2.1 Нейронні мережі, задача розпізнавання

Нейронна мережа (НМ) – це послідовність нейронів, які з’єднані між собою синапсами. Також це можна описати як серію алгоритмів, які намагаються розпізнати основні відносини у наборі даних за допомогою процесу, що імітує роботу людського мозку [15].

НМ зазвичай використовуються у досить широкому колі задач, виділимо з них основні:

1) класифікація – одна із найважливіших задач для інтелектуального аналізу даних. Вирішується вона за допомогою аналітичних моделей, які називають класифікаторами. Ця задача дуже поширена через простоту алгоритмів та методів для її реалізації, також вона має кращу інтерпретованість результатів порівняно з іншими технологіями аналізу даних, наприклад, класифікаторів для побудови яких використовуються як статистичні методи (дискримінантний аналіз, метод порівняння середніх та метод порівняння дисперсій). Використання НМ у якості класифікаторів має багато переваг: вони самонавчаються та їх робота практично не потребує втручання, універсальні апроксиматори, які можуть апроксимувати безпере-

рвну функцію з прийнятною точністю, можуть бути нелінійними моделями, що дозволяє швидко та ефективно вирішувати завдання класифікації за відсутності лінійної роздільності класів [16];

2) прогнозування – це передбачення майбутніх подій. Метою прогнозування є зменшення ризику під час прийняття рішень. Прогноз може бути помилковим, але ця помилка залежить від використовуваної системи для прогнозу.

Надаючи НМ більше ресурсів та матеріалу, можна збільшити точність прогнозу та зменшити збитки, пов'язані з невизначеністю при прийнятті рішень системою. Сфери, де найчастіше використовують прогнозування, це передбачення цін на фондовій біржі, прогноз споживання електроенергії, прогноз погоди, спортивні матчі тощо;

Компанії у багатьох своїх продуктах розглядають застосування НМ на вирішення завдання прогнозування часових рядів. Користувач вибирає довільний часовий ряд і розбиває його на 3 множини: навчальні, тестуючі та контрольні вибірки, які потім подаються на вхід мережі. Результатом прогнозування є значення часового ряду в потрібний час. Зазвичай прогнозування використовують у застосунках для передбачення відвідуваності громадських місць (пошти, магазини, державні установи) через пандемію COVID-19. Для підвищення якості прогнозу необхідно зробити попередню обробку інформації, і чим її буде більше - тим точнішим буде прогноз НМ.

При виборі архітектури на вирішення завдання пророкування зазвичай тестується кілька змін із різною кількістю інформації та елементів, що до неї входять. Виходячи з того, що завдання прогнозування є окремим випадком, то вона може бути вирішена такими архітектурами НМ: багатошаровим перцептроном (MLP), мережею Вольтеррі та мережею Ельмана [17].

3) Розпізнавання – одне з найпопулярніших завдань нейронних мереж на цей час. Використовується для розпізнавання візуальних образів, спершу навчаючи

нейронну мережу на зображеннях (зазвичай такий набір даних називають dataset), а потім можна її використовувати обробляючи вхідні відеопотоки з камер.

В наш час створюються нейронні мережі, які здатні розпізнавати символи та числа на папері, ідентифікувати особи, визначати швидкість руху об'єкту. Ці функції дозволяють полегшити працю людини та автоматизувати велику кількість процесів, а також підвищити надійність та точність уникаючи такої проблеми як людський фактор. Для того, щоб почати розпізнавати – НМ має спочатку вивчити всі особливості об'єкта та виявити закономірність.

Для побудови моделі фахівці з аналізу даних та їх обробки повинні визначитися з вхідними даними для подальшої їх обробки та критеріями, а саме які вхідні характеристики та модель враховуватиметься при прогнозуванні результату. Нейронні мережі вивчають ознаки прямо з отриманих даних, які вибрали фахівці для їхнього навчання, тому спеціалістам не потрібно вручну виділяти особливості та ознаки кожного елемента.

Кожне зображення класифікується для майбутньої обробки під час навчання, НМ визначає об'єкт у певну категорію. Мережа бере кожне зображення і пропускає його через алгоритми, вивчаючи візуальні характеристики кожної категорії , до якої він відносить об'єкт і зрештою вчиться розпізнавати кожен клас зображень, що надходить на вхід мережі.

Програмне забезпечення, засноване на моделях глибокого навчання, допомагає кардіологам справлятися з великим обсягом роботи з розпізнавання різних медичних зображень, які зберігаються у медичних закладах у великому обсязі. Нейронні мережі не можуть замінити спеціаліста, але можуть допомогти виявити аномалії на конкретних знімках і виділити щось, що може не помітити сама людина.

“Звичайні” НМ - звичайною найчастіше називають повнозв'язну нейронну мережу. Повнозв'язні мережі є штучними нейронними мережами, кожен нейрон якої передає свій вихідний сигнал іншим нейронам, у тому числі й самому собі (рис.

2.1). Усі вхідні сигнали подаються всім нейронам. Вихідними сигналами НМ можуть бути вихідні сигнали нейронів після кількох циклів відпрацювання мережі.

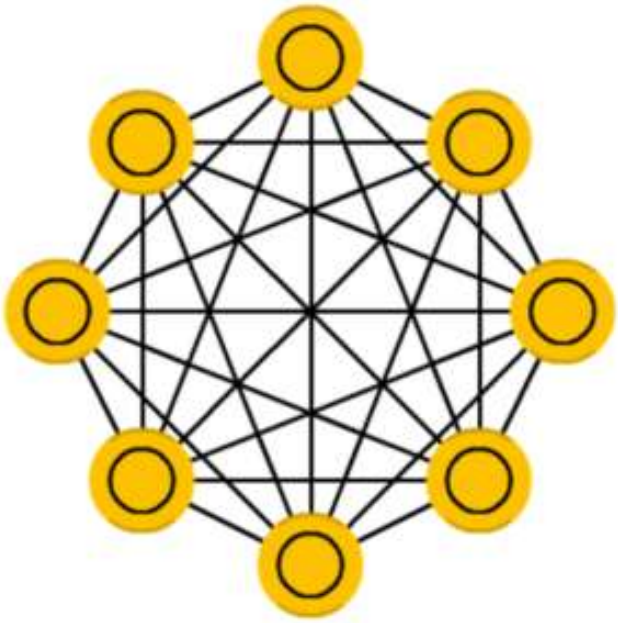

Рисунок 2.1 – Структура повнозв’зяної нейронної мережі

У багатозв'язних мережах нейрони поєднуються в шари. Шар містить кілька нейронів з єдиним типом вхідного сигналу. Число нейронів у кожному шарі може бути будь-яким і це ніяк не пов'язане з кількістю нейронів в інших шарах, все залежить від алгоритмів, що використовуються в мережі. У загальних випадках мережа складається з N-кількості шарів, які пронумеровані у порядку зліва направо. Зовнішні вхідні сигнали повинні подаватися на входи нейронів першого шару (вхідний шар прийнято нумерувати нульовим), а виходами є вихідні сигнали останнього шару. Входи нейронної мережі розглядають як вихід "нульового шару" нейронів, які служать лише як розподільні точки [18]. Перетворення та підсумовування сигналів тут не розраховуються. Крім вхідних та вихідних шарів у багатошаровій НМ існує ще один або кілька прихованих шарів.

Згорточні нейронні мережі – має особливу архітектуру, яка дозволяє їй ефективно розпізнавати образи. Дана нейронна мережа заснована на чергуванні згорточних та субдискретизуючих шарів, а структура є односпрямованою. Згортка НМ отримала свою назву від операції згортки, яка передбачає, що кожен фрагмент зображення буде помножений на ядро згортки по елементам, при цьому отриманий нею результат повинен підсумовуватися і записуватися в схожу позицію вихідного зображення.

На даний момент існує множина архітектур НМ, з них можна виділити наступні:

1) Багатошаровий перцептрон (MLP) – це НМ прямого розповсюдження. Вхідний сигнал даних мережах поширюється у напрямі, від шару до шару (рис. 2.2). Багатошаровий перцептрон у загальному вигляді складається з наступних елементів:

- множини вхідних вузлів, які утворюють вхідний шар;
- одного або кількох прихованих шарів обчислювальних нейронів;
- одного вихідного шару нейронів.

Багатошаровий перцептрон це узагальнення одношарового перцептрона Розенблатта. Кількість вхідних та вихідних даних у багатошаровому перцептроні визначається умовами поставленого перед НМ завданням. Одна зі складних умов – це вибір, які умови та дані враховувати, а які – ні.

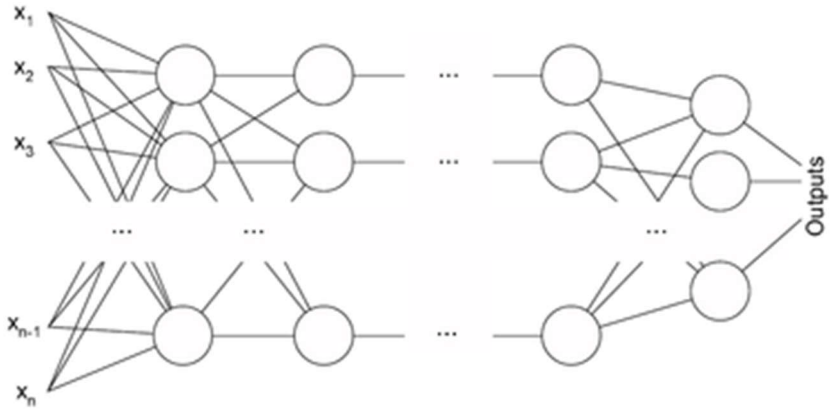

Рисунок 2.2 – Структура багатошарового перцептрона Румельхарта

Багатошарові перцептрони успішно застосовуються для вирішення різноманітних складних завдань та має три відмінні ознаки від інших архітектур. Перша ознака, це те що кожен нейрон має нелінійну функцію активації. Нелінійна функція має бути гладкою (повсюди диференційованою), на відміну від жорсткої порогової функції, яка використовується в перцептроні Розенблатта. Широко поширена функція, яка підходить під ці умови – сигмоїдальна. Наявність нелінійної функції відіграє дуже важливу роль у багатошаровому перцептроні, так як у гіршому випадку відображення входів та виходів мережі можна звести до звичайного одношарового перцептрону. Друга ознака - багатошаровий перцептрон містить один або кілька шарів прихованих нейронів, які є частиною входу/виходу мережі. Ці нейрони дозволяють мережі вчитися вирішувати дуже складні завдання, послідовно вивчаючи найважливіші ознаки з вхідного образу отриманих даних для більш точного аналізу. Третя ознака – багатошаровий перцептрон має високий ступінь зв'язності, за допомогою синаптичних з’єднань. При зміні рівня зв'язності мережі багатошаровий перцептрон потребує змін множини синаптичних з'єднань або вагових коефіцієнтів [20].

Перерахування всіх цих властивостей із здатністю до навчання на власному досвіді забезпечує високу обчислювальну потужність багатошарового перцептрона. Однак ці ознаки є причиною неповноти сучасних знань про цю архітектуру. Дані ознаки суттєво ускладнюють теоретичний аналіз багатошарового перцептрону.

2) Рекурсивні (RvNN) – це вид НМ, що працює з даними змінної довжини. Для навчання цих моделей використовують ієрархічні структури. Наприклад, можна навести набір зображень, складені з об'єктів, що об'єднують подібні об'єкти, що включають множини інших об'єктів. Виявлення структури об'єкта та її розбиття на об'єкти – нетривіальне завдання.

Структура у цієї нейронної мережі схожа з деревом аналізу. Оскільки дана структура НМ рекурсивна, то визначити значення вузлів можна за їхніми нащадками. Самі нащадки є вузлами і можуть бути визначені своїми нащадками рекурсивно. Тому для цієї архітектури характерно лише два вагові коефіцієнти - лівий, який показує, як він пов'язаний з вузлом, який пов'язаний з правим і так далі.

Головне завдання рекурсивної НМ ідентифікувати всю структуру об'єкта, і їх окремі об'єкти. Навчаючи послідовні структури у завданнях обробки природної мови, усі речення та слова моделюються через векторне уявлення слів (рис. 2.3).

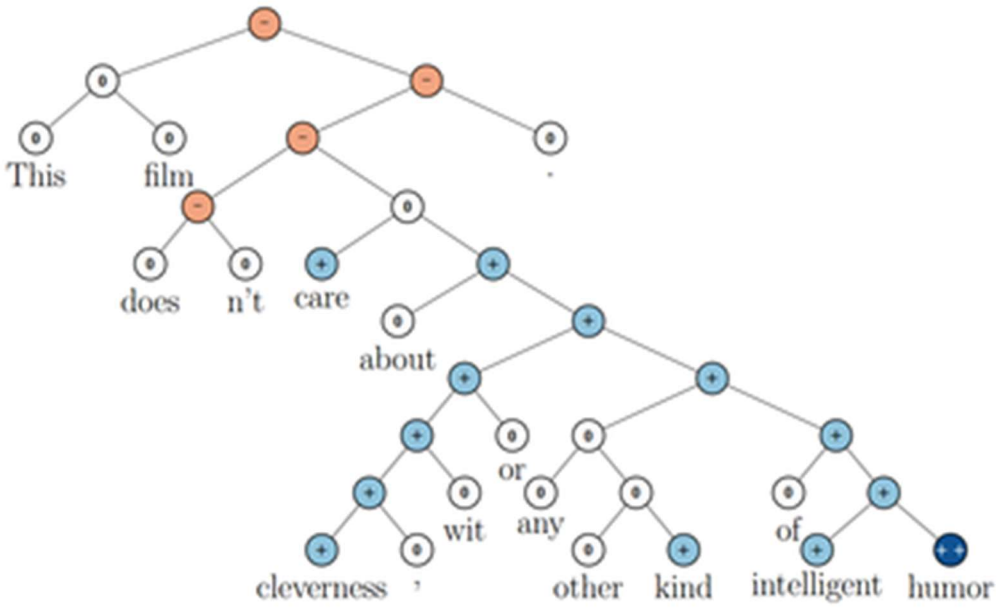

Рисунок 2.3 – Обробка природної мови використовуючи рекурсивні НМ

Базова структура рекурсивної мережі є бінарним деревом, що складається з кореня та його дочірніх компонентів. Компоненти ж уявляють собою набори нейронів, розмірність якого залежить від складності вхідних даних. Дані подаються на дочірні компоненти, а корінь використовується для класифікації даних, тобто визначає об'єкт певного класу.

3) Довга короткострокова пам’ять – один з різновидів архітектури рекурентних НМ, яка має властивість вчитися довгостроковим залежностям. Самі рекурентні НМ реалізують пам'ять до штучних НМ, ця пам'ять виходить короткою, тому

що на кожній епосі при навчанні отримана інформація поєднується між собою і в результаті повністю перезаписується. За допомогою даної архітектури розроблені модулі LSTM, щоб позбавитися проблеми довготривалої залежності. Модуль LSTM не використовує функцію активації всередині компонентів, що дозволяє запам'ятовувати вхідну інформацію як на короткі, так і на довгі проміжки часу. Виходячи з цього, значення, яке запам'ятовує НМ, не розмивається і не змішується з іншими даними при тренуванні мережі.

У даного модуля є стан комірки, сама комірка постає як пам'ять мережі, яка передає необхідну інформацію іншим модулям. Це забезпечує одержати більш ранню інформацію на пізніших етапах обробки без втрат, оминаючи проблеми короткочасної пам'яті.

Неглибокі – групи неглибоких двошарових моделей можна використовувати для представлення шарів векторів. Глибина це відмінна риса у глибоких нейронних мереж, але це, що для її роботи потрібно більше потужності, обчислень, що визиває затримку у роботі. Тому в практиці можуть використовуватися неглибокі НМ, деякі з них зберігають високу продуктивність і ефективність при накладенні одного шару на інший.

На даний момент робота із зображеннями – одна з важливих сфер для застосування DL. Зображення з усього світу являють собою величезне складання неструктурованих даних. Задіявши технології НМ, ML та ШІ можна класифікувати набори даних для виконання великої кількості задач: соціальних, аналітичних, побутових та забезпечення безпеки.

НМ для завдання розпізнавання зображень – один із відомих способів застосування мереж. Незалежно від типу завдання та мети, дана технологія працює за етапами. В якості образів, які потрібно розпізнати, можуть бути різні об'єкти, тобто класифікація чого завгодно: зображення, об'єкти, що входять до нього, рукописний текст, медичні захворювання і так далі. У момент навчання НМ надається набір

даних та позначки, на які потрібно звернути увагу при класифікації. У цьому використовується вектор значень ознак, а ця сукупність векторів має допомогти НМ визначити якого класу віднести об'єкт.

При навчанні НМ важливо не тільки визначати величезну кількість ознак для забезпечення точності при надходженні нових даних, а й щоб мережа не перенавчилася. Під перенавчанням мається на увазі можливість мережі "адаптуватися" під конкретний набір даних, що може завадити і знизити точність у майбутньому під час роботи з новими зображеннями (з тими самими класами) визначати образи. Так само важливо враховувати при навчанні, що вихідний набір даних повинен бути однозначним і не суперечити сам собі, щоб у процесі роботи НМ не видавалися хибні високі відсотки розпізнавання.

Створення НМ для розпізнавання включає у себе декілька етапів (рис. 2.4):

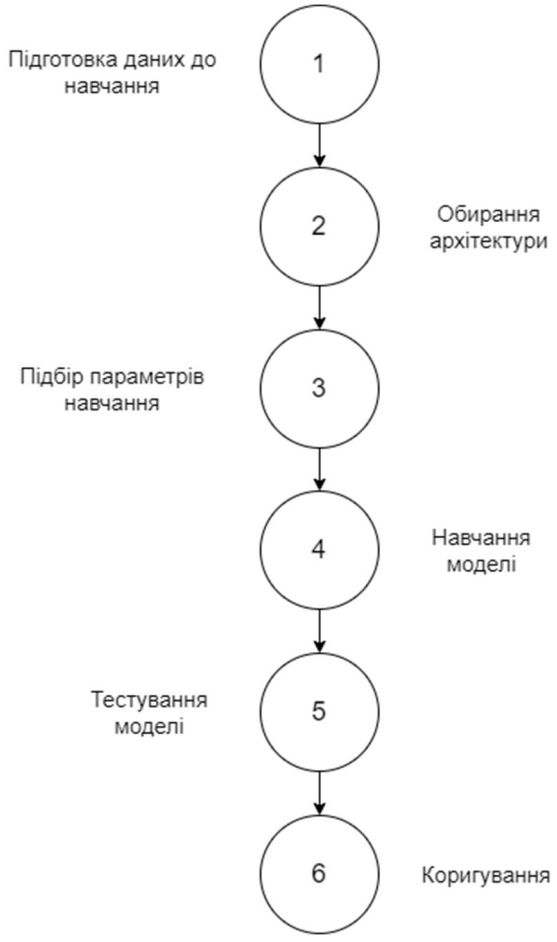

Рисунок 2.4 – Етапи навчання НМ

Навчити нейронну мережу можна різними способами: з вчителем, з частковим залученням вчителя, без вчителя, з підкріпленням. Спосіб навчання з вчителем має на увазі, що при навчанні НМ розпізнавати образи формується вибірка з класифікованими даними, мережа отримує на вхід зображення, після чого йдуть обчислення помилки, порівнюючи свої вихідні дані із класифікованими даними та ознаками. Залежно від ваги помилки та її типу - ваги НМ коригуються і вже за наступних зображеннях мінімізується помилка.

У разі навчання без вчителя НМ не отримує жодних ознак та класів заздалегідь, перед мережею стоїть завдання знайти невідомі їй відповіді та закономірності у поданих на вхід даних. З відомих завдань можна виділити кластеризацію - поширена задача для навчання моделі без участі вчителя, при кластеризації підбираються загальні ознаки для майбутнього угруповання даних на виході. З недоліків методу можна назвати, що у навчанні без вчителя важко обчислити точність використовуваного на навчання алгоритму, оскільки даних відсутні правильні рішення і заготовлені заздалегідь ознаки, що утрудняє і уповільнює процес навчання.

У навчанні з підкріпленням заготовлений набір даних містить як і підготовлені зображення з ознаками, і без них. Цей метод використовується у випадках, коли класифікувати заздалегідь усі об'єкти практично неможливо. Тому НМ може витягти з малої кількості даних закономірності і сама поліпшити точність своїх пророцтв і розпізнавань образів. Також це значно прискорює процес проти застосуванням методу навчання без вчителя.

Навчання з підкріпленням – це як у відеоіграх, все засноване на стимулах “гравця”. При правильних діях гравець отримує нагороди та рухається по грі далі. Даний метод діє за тією ж аналогією, популярна сфера застосування даного навчання для досліджень – відеоігри. НМ з кожною ітерацією намагаються знайти оптимальний спосіб досягнення поставленої йому мети та покращення продуктивності при виконанні завдання. Коли мережа приймає рішення про дію, то за правильного прийняття вона отримує нагороду. Найголовнішою метою даного навчання є

навчити НМ зумовлювати наступний правильний крок для отримання найкращого результату. Прийнявши рішення, мережа отримує зворотний зв'язок, виробляючи нові тактики і дії, які призводять її до більшого заробітку нагород. Даний підхід можна покращити використовуючи ітеративність – чим більше мережа отримує зворотний зв'язок, тим краще його вибори для наступного кроку.

2.2 Використання TensorFlow для машинного навчання

TensorFlow – це наскрізна платформа з відкритим вихідним кодом для машинного навчання. Дана платформа має комплексну та гнучку екосистему інструментів, бібліотек та ресурсів, яка відкриває програмістам та дослідникам можливість впроваджувати найсучасніші технології машинного навчання у різні сфери життя [21].

Тензор – це стандартний спосіб представлення даних у системах глибокого навчання. Тензори – це багатовимірні масиви, розширення двомірних матриць для представлення даних, що мають більш високі розмірності.

На даний момент TensorFlow підтримує відомі операційні системи, а саме: Linux, Windows, macOS та для мобільних обчислювальних платформ, включаючи Android та iOS.

TensorFlow має API-інтерфейси, доступні кількома мовами програмування як для побудови моделей, так і для їх використання. API для мови програмування Python в даний час є найбільш опрацьованим і простим у використанні, але API інших мов простіше інтегрувати в проекти і можуть досягати успіху за швидкістю навчання і виконання, так як у мови Python є глобальне блокування інтерпретаторів (GIL), що суттєво уповільнює час виконання програми. Окрім мови Python платформа TensorFlow підтримує наступні мови програмування: JavaScript, C++, Java, C#, Haskell, Ruby, Rust, MATLAB. Але частину із них підтримує суспільство, яке використовує цю платформу, тому для запобігання помилок краще використовувати мови програмування, які підтримують саме розробники платформи.

Використовуючи мову JavaScript можна навчати та використовувати моделі прямо у браузері завдяки розширенню TensorFlow.js, що допомагає розкрити повний потенціал веб-розробки та впровадження у продукти моделей глибокого навчання.

Історія платформи TensorFlow

Проект "ML DistBelief" розроблявся компанією Google для внутрішніх проектів з 2011 року для роботи з нейронними мережами глибокого навчання. Даний проект став використовуватись у багатьох дослідницьких та комерційних проектах фірми Alphabet. Після успіху системи ML DistBelief компанія Google вирішила вивести проект на новий рівень, і для оптимізації коду виділила групу з кількох розробників, до якої увійшов Джефф Дін. Метою розробників було повне перероблення коду для спрощення та оптимізації бібліотеки, а також збільшення надійності та зручності використання іншими розробниками. І так нова бібліотека отримала назву TensorFlow [22]. У 2013 році до проекту приєднався дослідник Джеффрі Хінтон, який у 2009 році створив метод узагальненого зворотного розповсюдження помилки. Ряд поліпшень, принесених у розробку допомогли поліпшити точність нейронних мереж.

9 листопада 2015 року код проекту було відкрито для всіх. Цей код під ліцензією Apache Open Source, що дозволяє його безкоштовно впроваджувати у комерційні проекти та доповнювати власними ідеями і функціоналом. Це допомогло розробникам-ентузіастам проявити себе і зробити внесок у ML, що значно прискорило процес розробки та підтримки проекту, а також відкрило нові можливості у цій сфері. TensorFlow постійно покращується, її постійно постачають чимось новим, оптимізують. Крім того, зростає і спільнота, яка сформована навколо цієї бібліотеки.

TensorFlow – це система машинного навчання фірми Google другого покоління. У той час як перша версія працює на декількох спеціалізованих пристроях,

Дана платформа може працювати на багатьох паралельних процесорах, використовуючи як CPU, так і GPU, спираючись на архітектуру з відкритим кодом CUDA для підтримки великих обчислень загального призначення на будь-яких графічних процесорах.

TensorFlow на даний момент має багатий спектр застосувань, а саме:

- моделі для розпізнавання образів, об’єктів;
- обробка замовлень;
- формування рекомендацій на основі вивченої вибірки даних;
- допомагає лікарям виявляти респіраторні хвороби та діагностувати ряд інших захворювань;
- на основі симптомів та протипоказань рекомендувати правильні ліки для подальшого лікування;
- охоронні системи;
- у багатьох підприємствах при контролі виготовлення продукції краще задіяти систему на базі машинного навчання, щоб уникнути помилок;
- аналізувати текст на його правдоподібність, щоб допомогти людині уникнути неправди;
- для аналізу багатої кількості даних та їхньої класифікації;
- онлайн-платформи для створення адаптивного плану навчання;
- у агрономії для виявлення захворювань рослин, цілих плантацій.

Серед переваг TensorFlow можна виділити наступне:

- швидкість роботи: навіть використовуючи мову програмування Python можна прискорити розробку та навчання моделі за допомогою модулів та типів даних на мові програмування C++, які додані у типи даних платформи TensorFlow. Це заощаджує багато часу, тому що змінні оточення C++ оброблюються та компілюються швидше, ніж у мові Python, тому розробники створили інтерпретатор CPython для прискорення виконання програми.

– інтеграції: ця платформа під відкритим вихідним кодом, що надає можливість легко інтегрувати у будь який застосунок, використовуючи інші бібліотеки, такі як Keras, numpy, matlpotlib, spark.

– простота: мова програмування Python так само відома своїм простим синтаксисом, що полегшує читання і розуміння коду, тому використовувати технологію TensorFlow разом із мовою програмування Python полегшить написання коду.

– готові моделі: якщо дуже складно створити будь-яку модель або зрозуміти принципи роботи бібліотеки, то можна звернутися до TensorFlow Hub, де можливо знайти готові моделі, набір даних або код під будь-яку популярну задачу.

Так як дана платформа поширюється з відкритим вихідним кодом, то їх використовує багато великих компаній (Google, IBM, Intel, DeepMind, AMD, Lenovo, LinkedIn), що приносить величезний внесок у розвиток цієї технології розробниками цих компаній. Наприклад, компанія Intel ввела такі оптимізації у свої технології:

– компанія оптимізувала свої процесори за допомогою технологій TensorFlow, що дозволила під час роботи на ЦП запускати Python-програми та отримати суттєвий приріст продуктивності без ручних змін конфігурацій та додаткових трудовитрат для налаштування НМ;

– позбулася дорогих перетворень макетів даних, об'єднавши всі операції під час навчання на своїх процесорах, повторно використовуючи кеш на ЦП.

– обробляє проміжні стани, що дозволяють прискорити алгоритм зворотного розповсюдження помилки для оновлення ваг багатошарового перцептрон.

Оптимізація процесорів компанії Intel та AMD за допомогою технологій TensorFlow допоможе програмам з використанням глибокого навчання працювати набагато швидше, що підвищує доступність, гнучкість та масштабованість майбу-

тніх проектів. Наприклад, процесор Intel Xeon Phi призначений для лінійного масштабування ядра, що скорочує час для навчання майбутньої моделі. Компанії все далі продовжують підвищувати продуктивність процесор обробки більш складних і великих навантажень НМ.

Один із найважливіших елементів вивчення технології розробником – це типи даних. У платформи TensorFlow є набір примітивів, який легко застосувати для необхідних розробнику цілей. Дані примітиви шаблонізовані, з основних шаблонів можна помітити всім відомі типи даних мови С та типи даних, для підтримки багатовимірних масивів за допомогою бібліотеки NumPy, яка використовує готові конструкції мовою С, Python та Fortran. Наступний тип даних – це набір готових рішень і класів для обробки звукових доріжок, зображень, великих обсягів даних, що спрощують і прискорюють процес розробки для глибокого навчання.

2.3 Бібліотека Keras для взаємодії із нейронними мережами, задача оптимізації

Keras – це бібліотека з відкритим вихідним кодом, яка написана на мові програмування Python, що забезпечує взаємодію з НМ. До версії 2.3 бібліотека Keras використовувалася як надбудова над бібліотекою Theano, яка використовувалася для великих чисельних обчислень мовою програмування Python. Після версії 2.3 Keras стали підтримувати розробники платформи TensorFlow, після чого підтримка розробки версій для Theano та інших нейромережевих бібліотек припинилася.

Ця бібліотека орієнтована працювати з мережами глибинного навчання, основною метою розробників була розробляти цей інструмент компактним з можливістю розширення під свої потреби. Перша версія була створена для допомоги в дослідженнях проекту ONEIROS [23]. Автор цієї бібліотеки – інженер компанії Google Франсуа Шолле.

У планах компанії Google була ідея впровадити цю бібліотеку прямо в основну платформу TensorFlow, але в результаті розробники вирішили відокремити

Keras, адже згідно з концепцією цієї бібліотеки вона є інтерфейсом для налаштування та адаптації НМ, а не системою машинного навчання.

Keras більш високорівневий набір абстракцій, який допомагає розробникам легко взаємодіяти та формувати НМ, незалежно від обраної ним обчислювальних бібліотек. В даний момент над Keras також працює компанія Microsoft з можливістю додати туди низькорівневу бібліотеку CNTK, що дозволить обробляти великий обсяг вхідних даних і використання внутрішньої пам'яті для подальших обробок.

Бібліотека Keras підходить для:

- будування моделей, за якими буде проходити навчання НМ;
- візуалізація готової моделі;
- підбір набору даних для навчання мережі;
- тестування готових моделей;
- збірка та запуск програми машинного навчання;
- перетворення даних, які подаються на вхід моделі, що навчається.

Усі вищенаведені пункти можна робити без бібліотеки Keras, але це потребує більше часу та затрат при формуванні моделі, так як Keras виступає у ролі програмного інтерфейсу, що спрощує усю роботу із НМ. Бібліотека Keras працює з моделями – це схеми, в яких поширюється та перетворюється вхідна інформація. Модель – структура мережі, зазвичай її у вигляді схеми, таблиці чи графа. Модель НМ визначає свою архітектуру та зміни, які були задані внаслідок навчання, а також зберігає у собі використовувані алгоритми навчання.

Програміст або дослідник може описати модель двома існуючими способами.

1. Із використанням готового функціонального API, яке надає Keras. Потрібно створити об'єкт, привласнити йому необхідні навчання параметри, які заздалегідь описані в готовому API [24].

2. Послідовний – описати модель набором команд, кожна яка додає до моделі параметри. Наприклад можна навести таку послідовність: описується щільність моделі, після формула, яка буде застосовуватися для розрахунків, а потім інші необхідні параметри для опису моделі.

Ці методи зручні для використання і залежать від цілей розробника, який спосіб йому більше підходить. Серед особливостей бібліотеки Keras можна виділити наступні:

– кросплатформенність: працює не тільки на відомих ОС Windows, Linux та macOS, а й на мікрокомп'ютерах, мобільних обчислювальних платформах Android та iOS, в хмарних сервісах та браузері;

– підтримує різні архітектури НМ: згорткові, багатошарові перцептрони, рекурентні та інші.

– легкий у розумінні: бібліотека написана на чистому Python, що дає можливість розробникам краще розуміти структуру та підтримувати проект у подальшому;

– сумісність: сумісна із мовою програмування Python з 2.7 до теперішніх версій;

– модульність: бібліотека складається із модулів або блоків, кожен з котрих має свій набір функцій. Вони представлені вже у готовому вигляді, розробнику потрібно тільки обрати ці модулі, поєднати друг з другом та налаштувати під свої потреби.

– оптимізований: для складних обчислень може використовувати CPU та GPU із звичайним або графічним процесором для прискорення швидкості навчання моделі НМ;

Додаткові компоненти бібліотеки Keras. Для кращої роботи Keras потребуються додаткові компоненти. Це інші інструменти, завдяки яким Keras стає затребуваною бібліотекою на тлі інших інструментів навчання моделей НМ.

1) h5py - бібліотека, яка дозволяє зберігати модель у форматі "model_name.h5";

2) cuDNN - бібліотека від компанії NVIDIA, дозволяє Keras використовувати без проблем графічний процесор для великих та довгих розрахунків. Найкращий варіант для ресурсномістких обчислень під час навчання моделі;

3) Pydot - бібліотека, яка потрібна для візуалізації графів, які зображують отриману в результаті навчання модель.

Використовуючи вищенаведені бібліотеки Keras показує себе як найкращий вибір для створення моделей НМ.

Оптимізатори в Keras. Одна із головних переваг у бібліотеки Keras – оптимізатори, які допомагають оптимізувати алгоритм градієнтного спуску у компіляції НМ TensorFlow. Однин із популярних вибором розробників – це алгоритм оптимізації Adam. Adam використовується замість стохастичного градієнтного спуску для циклічного оновлення ваги мережі на навчальних даних. У задачах оптимізації даний алгоритм простий у реалізації, потребує небагато пам’яті для роботи та ефективний для великих обчислювань.

Окрім даного алгоритму Keras має ще багато способів оптимізувати процес взаємодії та навчання НМ:

– SGD – стохастичний градієнтний алгоритм;

– RMSProp – алгоритм подібний до іншого під назвою Adagrad, але RMSProp бореться з надмірним накопиченням квадратів градієнтів, справляється з шумами даних, рахунок адаптації з урахуванням середнього значення ваг;

– Adam – суміш алгоритму з моментом та квадратів градієнтів;

– Adagrad – оптимізація на основі квадратів градієнтів, цей алгоритм покращує продуктивність при проблемах у роботі з комп'ютерним зором та природною мовою;

– Adamax - варіант оптимізації по Adam, але без обмежень за нормою;

– Nadam – комбінація алгоритму Adam із нестерівським (Nesterov Adam) моментом.

На практиці перед створенням моделі частіш за все починають з алгоритму по оптимізації Adam, але якщо вихідні дані не відповідають поставленим умовам, то розробники пробують інші алгоритми, які зазначено вище. Алгоритм Adam поєднує в собі найкращі властивості двох алгоритмів (Adagrad та RMSProp), що дозволяє забезпечити найкращий алгоритм оптимізації при створенні моделі для НМ.

2.4 OpenCV як технологія для обробки потокового зображення

Людський зір - це одна із здібностей сприйняття інформації людиною, що допомагає їй бачити та контролювати все те, що існує. За допомогою цього органу сприйняття людина легко розпізнає об'єкти, просто глянувши на нього, вона може визначити її форму, розмір і стан. Однак цей орган має властивість втомлюватися і вимагає відпочинку для подальшого функціонування, тому люди почали для вирішення та автоматизації проблеми спостереження писати інструменти та платформи для відеоспостереження із класифікацією об'єктів, що можуть потрапити у поле зору камер.

OpenCV (англ. Open Source Computer Vision Library) – це бібліотека з відкритим вихідним кодом, яка доступна для використання для багатьох мов програмування (Python, C++, Java, C#, Ruby). Перша реалізація була виконана на мові C/C++. Дана бібліотека розроблена під велику кількість поширених платформ (Microsoft Windows, Windows RT, Linux (GCC), macOS, Android та iOS). Дана можливість надає перевагу у використанні додатків із цією бібліотекою на багатьох пристроях, що робить її ефективніше у порівнянні з іншими існуючими аналогами інструментів для комп’ютерного зору (CUDA, PCL, MATLAB та ін.).

Стартував проект у компанії Intel в 1999 році дослідником Гері Брадськи. Головною метою на час розробки проекту – розробити платформи Intel більш ціка-

вими для розробників за рахунок апаратного прискорення OpenCV. Також в подальшому компанія Intel хотіла затвердити загальний стандартний інтерфейс для комп'ютерного зору у цій галузі, щоб сприяти розвиненню таких технологій у нових моделях ПК. Після багатьох доробок та оновлень у 2009 році було випущено другу версію даної бібліотеки, що принесла у собі серйозні зміни інтерфейсу. Розробники змінили напрямок на спрощення роботи з даним інструментом, покращив безпеку, продуктивність та переписавши старі функції на більш новий формат.

У версії 2.2 бібліотеки OpenCV уся структура була розподілена на блоки по функціональному використанню (таблиця 2.1).

Таблиця 2.1 – Блоки бібліотеки OpenCV 2.2

Блок	Функціонал
opencv_core	містить ядро, базові структури, обчислення (введення/виведення в JSON, XML, генерація псевдовипадкових чисел)
opencv_imgproc	обробка зображень (фільтри, накладання ефектів)
opencv_highgui	користувацький інтерфейс для завантаження та зберігання зображень або відео
opencv_ml	містить методи та моделі машинного навчання

Закінчення таблиці 2.1

Блок	Функціонал
opencv_features2d	містить різні дескриптори (напр. SURF)
opencv_video	методи для аналізу руху та відстеження об’єктів
opencv_objdetect	методи для детектування об’єктів на зображенні
opencv_calib3d	методи для обробки тривимірних даних
opencv_flann	бібліотека для пошуку ближчих сусідів (метод FLANN)
opencv_contrib	бібліотека для опробування нових функцій
opencv_legacy	застарілий код для зворотної сумісності
opencv_gpu	прискорення функціоналу за рахунок нових функцій бібліотеки CUDA (від компанії NVIDIA)

На процесорах Intel є можливість включити додаткове апаратне прискорення за рахунок Intel Performance Libraries, які підтримують багатоядерність і містить у собі набір функцій для обробки великих наборів відеоданих і алгоритми перетворення Фур'є.

Усі ці важливі кроки змогли розвивати таку технологію як комп’ютерний зір та впровадити її усюди надавши людям оптимізований ліцензований код BSD, який не потребує оплати при використані у комерційних проектах.

Найскладніша частина розробки програми з відкритим доступом - набрати швидке зростання популярності і привернути увагу ентузіастів для продовження

стабільного зростання проекту. На таку бібліотеку як OpenCV вплинули не тільки розробники Intel, але і розробники інших компаній, зробивши великий внесок у розвиток технології комп'ютерного зору. Якщо подивитися на тимчасову лінію розвитку даного проекту (рис. 2.5), то можна звернути, що з появою у світі технологій потужніших обчислювальних машин став зростати попит на такі технології як НМ, ШІ та комп'ютерний зір. З періоду 2012 року по поточний час можна помітити все частіший випуск стабільних версій (LST) бібліотеки OpenCV.

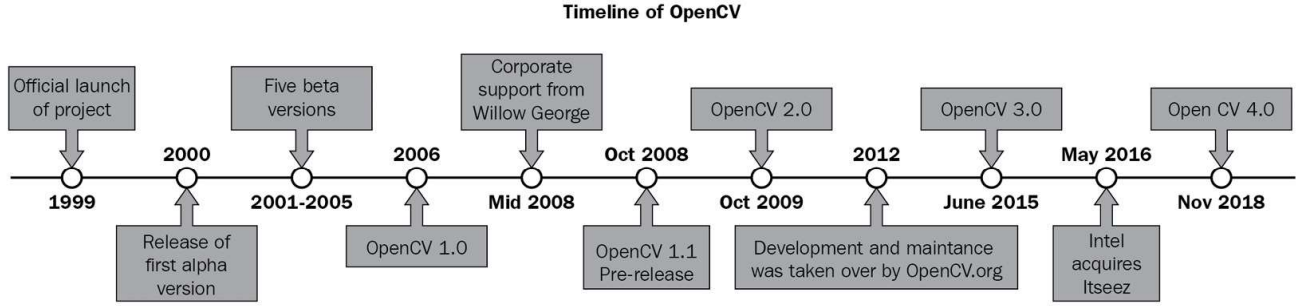

Рисунок 2.5 – Історія випуску бібліотеки OpenCV

Технологія комп'ютерного зору поєднується з НМ для вирішення великої кількості завдань, які воно може вирішити. Наприклад, якщо навчити модель НМ розпізнавати автомобільні номера, пішоходів та фіксувати швидкість руху об’єктів, то поєднавши цю модель із технологією комп’ютерного зору можливо розробити систему для фіксації порушень правил дорожнього руху (ПДР), яка буде обробляти зображення у режимі реального часу і виявляти порушників записуючи їх індивідуальний реєстраційний знак (номер) автомобілю у базу даних. Також такі технології впроваджують на практиці для роботомобілів: заснована на лідарах або спирається виключно на камери. Усі компанії по створенню ШІ для автомобілів використовують лідари для генерації 3D-простору, за допомогою чого авто орієнтується у просторі, але компанія Tesla вирішила використовувати виключно камери для своїх моделей Tesla Model S (рис. 2.6) [25].

Рисунок 2.6 – Як автомобіль компанії Tesla використовуючи камери із технологією комп’ютерного зору разом з НМ розпізнає об’єкти [26]

Також технології комп’ютерного зору використовують для ідентифікації особистості для отримання доступу до пристрою.

Група державних органів, яка провадить діяльність із захисту прав і свобод людини використовують технології комп’ютерного зору разом із НМ для відстеження та розпізнавання обличчя злочинця, щоб ідентифікувати особу та притягти її до відповідальності.

Передивившись публікацію про “Виявлення та підрахунок об'єктів на основі комп'ютерного зору для відповідності протоколу COVID-19: приклад з Джакарти” авторами Енді Ернесто, Хуан Інтан Канграван та іншими можна виділити усю актуальність використання комп’ютерного зору як допоміжної технології для вирішення проблеми епідемії COVID-19 [27]. Головна мета публікації – проаналізувати та забезпечити безпеку багатьох сфер людської діяльності, особливо діяльності, що

потребує фізичної взаємодії. Проблема в місті Джакарта та в інших містах світу полягає в мінімізації поширення вірусу COVID-19, для вирішення даної проблеми було прийнято оптимізувати та покращити камери виявлення та підрахунку транспортних засобів підключенням додаткової навченої моделі для виявлення та підрахунок скупчення людей з використанням технологій комп'ютерного зору OpenCV та НМ. Окрім бібліотеки OpenCV дослідники даної теми використовували бібліотеку Pandas для попередньої обробки вхідних даних. Алгоритм для обробки відео під назвою MobileNet SSD та YOLO у якості готового функціоналу для навчання НМ. Але із-за недостатньої кількості даних для навчання та складності визначати фігури у русі через розмиття вхідного зображення з камери відсоткову точність розпізнавання лише 30-40 відсотків.

Люди, які використовують або не використовують маску можуть бути виявлені за допомогою розробленого алгоритму ШІ. Подальший розвиток цієї методології потрібний для полегшення спостереження за порушеннями протоколів охорони здоров'я, коли в місті Джакарта застосовувався період “Нової норми”.

OpenCV так само служить як інструмент для попередньої обробки вибірки зображень перед навчанням НМ, для цього використовується зміна всіх зображень під один розмір методами згортки або пошуком найближчого сусіда.

Також було досліджено ще одну публікацію “COVID-19: автоматичне виявлення порушення правил соціального дистанціювання з використанням PP-Yolo та TensorFlow у OpenCV”, автори якого являються інженери коледжу Джабалпура Срішті Верма та Прашант Кумар Джайн [28]. Автори розглядають аналогічні проблеми щодо соціального або фізичного дистанціювання та носіння маски, за дотримання яких можна досягти меншого поширення вірусу COVID-19. Система включає комп'ютери з підключеними відеокамери в місцях поширення вірусу, таких як торгові центри, аеропорти, залізничні вокзали, школи та багато інших. Відповідно до постанови ВООЗ рекомендується дотримуватися дистанції не менше 2 метрів.

Даними інженерами були проведені різні дослідження для виявлення фізичної дистанції людей та одночасного виявлення маски на їхніх обличчях.

Мета цього дослідження - розробка системи з високим рівнем точності виявлення. Дана система розроблена на бібліотеці TensorFlow та OpenCV, що показує ідеальну роботу двох технологій разом.

Для вирішення задачі класифікації використовувався інструмент YOLO. Дана розробка двох інженерів у порівнянні з вищевказаною публікацією має більш точний відсоток розпізнавання (67,51 %), але цього все одно не відповідає бажаної точності.

Проблема, викликана низької точності визначення системою у тому, що натовп людей може мати невизначені структури, які важко визначати маски на обличчях людей і відстань з-поміж них. У майбутньому автори планують покращити роботу системи обчисленням правильного кута для вищого відсотка розпізнавання дистанції.

Вивчивши архітектуру використовуваної бібліотеки розробник або дослідник зможе краще зрозуміти усю роботу і процеси, які відбуваються поза його уваги, щоб зрозуміти усі алгоритми, які використовує система. CV опрацьовує усі алгоритми обробки зображень та візуалізації, що може використовуватися при створенні графічних редакторів. HighGUI використовується для створення вікон, відстеження та обробки користувацьких вікон, відстеження дій мишею та клавіатури, додання об’єктів на робочу зону, читання та запис зображень з диску користувача, також надає можливість записувати відео або зображення у файл відповідного формату (.mp4, .jpeg, .png тощо). CXCORE реалізує усі необхідні алгоритми, структуру та функції обробки зображень. MLL містить у собі набір інструментів для статистичної класифікації і обробки великого обсягу даних з використанням методів кластеризації. Архітектура бібліотеки OpenCV зображено на рис. 2.7.

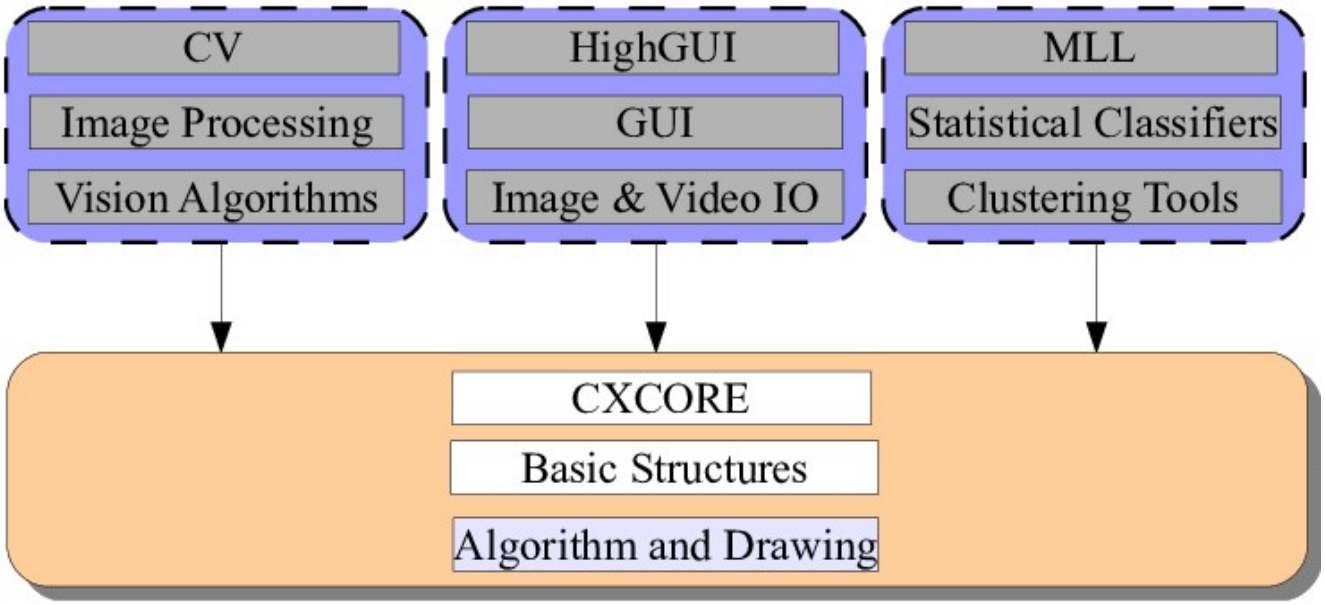

Рисунок 2.7 – Діаграма, що зображує архітектуру бібліотеки OpenCV

Якщо вивчати типи даних, бібліотека OpenCV має схожість з бібліотекою TensorFlow. Обидві ці бібліотеки використовують примітивні дані мови C++ та NumPy для оптимізації та більш швидкої роботи з векторами та двовимірними матрицями у процесі обробки потокового зображення з відеокамери.

Висновки до розділу 2

У наш час нейронні мережі вже інтегровані у повсякденне життя. Пошукові алгоритми Google та інших пошукових систем розроблені та підтримуються за допомогою НМ. Товарні послуги на основі переглянутих користувачів товаром аналізують та видають список рекомендацій теж при використанні НМ. Тому для вирішення більшості проблем вже прийнято використовувати НМ як основний інструмент у створенні. Бібліотека TensorFlow містить у собі велику кількість інструментів, алгоритмів та готових рішень, які допоможуть побудувати, навчити та інтегрувати НМ прямо у будь-який застосунок.

Бібліотека Keras допомагає сконструювати майбутню модель для нейронної мережі, легко та швидко створювати власні прототипи (завдяки масштабованості

та модульності бібліотеки). Дані моделі легко інтегрувати в середовище TensorFlow і без проблем використовувати їх, так як з останніми версіями бібліотеки розробники доклали всіх зусиль для їх спільної роботи, при цьому використовуючи чисту мову програмування Python.

Для вирішення завдань розпізнавання образів використовується бібліотека комп'ютерного зору OpenCV, що спрямована на обробку вхідних зображень та їх аналізу. Об'єднавши TensorFlow з Keras, можна створити екосистему, яку легко впровадити в методи OpenCV, що спрощує роботу розробникам при створенні програм для розпізнавання та аналізу відеопотоку з камери.

Усі вищенаведені бібліотеки є кросплатформеними, що дозволяє впровадити технології нового рівня на будь-який пристрій з мінімальними ресурсними витратами та втратою часу, адже написані модулі на швидкому C++ вирішують усі проблеми з продуктивністю та швидкодією коду.

Використання бібліотек TensorFlow, Keras та OpenCV допоможуть у проектуванні та створенні системи. Дані бібліотеки містять усі необхідні методи для навчання НМ та обробки вхідного відеопотоку з вебкамери.

3 МОДЕЛЮВАННЯ СИСТЕМИ. НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

3.1 Використання архітектури згорткової нейронної мережі у задачах розпізнавання

На даний момент існує багато архітектур НМ. Кожна підходить для великої кількості задач, але між ними є велика різниця у швидкості навчання, виконання та адаптації під різні платформи, а також продуктивності на більш слабкій техніці. Тому розробнику потрібно вивчити усі деталі кожної архітектури, щоб обрати найкращу для своєї задачі. Серед такої величезної кількості для розпізнавання образів була обрана архітектура згорткової НМ.

Після багаторазових досліджень та тестів серед дослідників та розробників найкращі результати при розв'язанні задачі розпізнавання осіб показала згорткова НМ (англ. Convolutional Neural Network). Ця НМ сприяла у майбутньому створити такі архітектури як когнітрон та неокогнітрон. Вся успішність даної архітектури завдання розпізнавання полягає у можливості враховувати двовимірну топологію зображення, на відміну архітектури багатошарового перцептрона [28].

Дана архітектура НМ може частково забезпечити стійкість до різного роду спотворення об'єкта, а саме: повороти, зміна ракурсу, масштабу та поворотів. Згорткові НМ поєднують у собі 3 архітектурні схеми, щоб забезпечити ту саму адаптацію до просторових спотворень об'єкта:

- зменшення вагових коефіцієнтів за рахунок спільних вагів (також забезпечує знаходження схожих рис у любому місці на зображенні);
- локальні рецепторні поля забезпечують двомірну зв’язність нейронів;
- просторові підвибірки за рахунок ієрархічної організації НМ.

Щороку серед дослідників проходить міжнародний конкурс із розпізнавання образів під назвою ImageNet [29]. Серед усіх архітектур згорткова НМ вважається

однією з найкращих за точністю та швидкістю знаходження об'єктів на вхідних даних. З 2012 року НМ почали посідати перші місця у міжнародному конкурсі ImageNet (рис. 3.1).

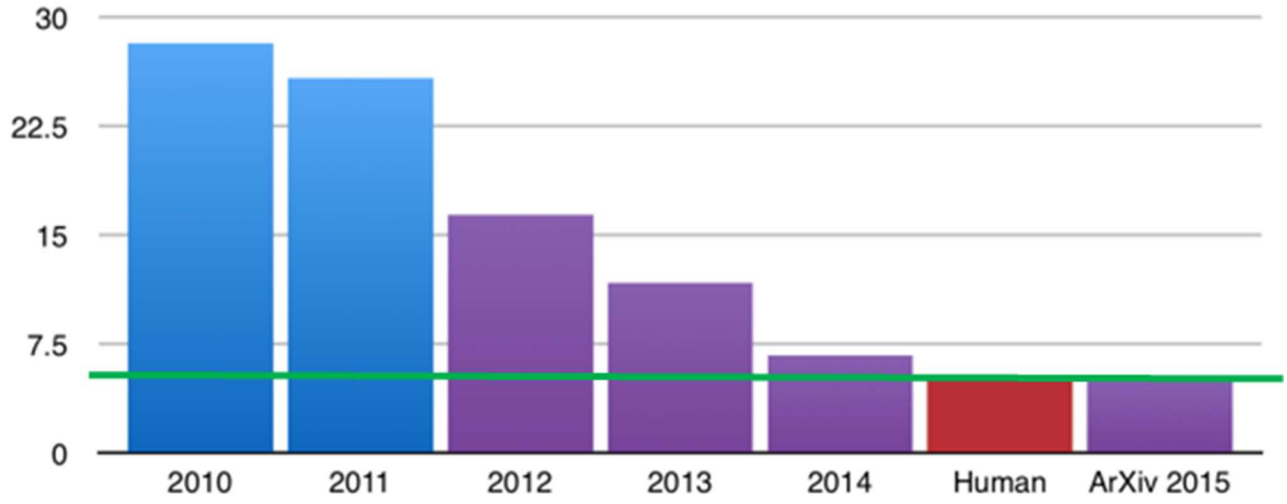

Рисунок 3.1 – Щорічні результати у задачі розпізнавання з використанням набору даних від ImageNet [30]

Структура згорткової НМ. Згорткова НМ складається з трьох видів шарів (рис. 3.2):

1) згортковий шар;
2) субдискретизуючий (підвибірковий) шар;
3) перцептрон.

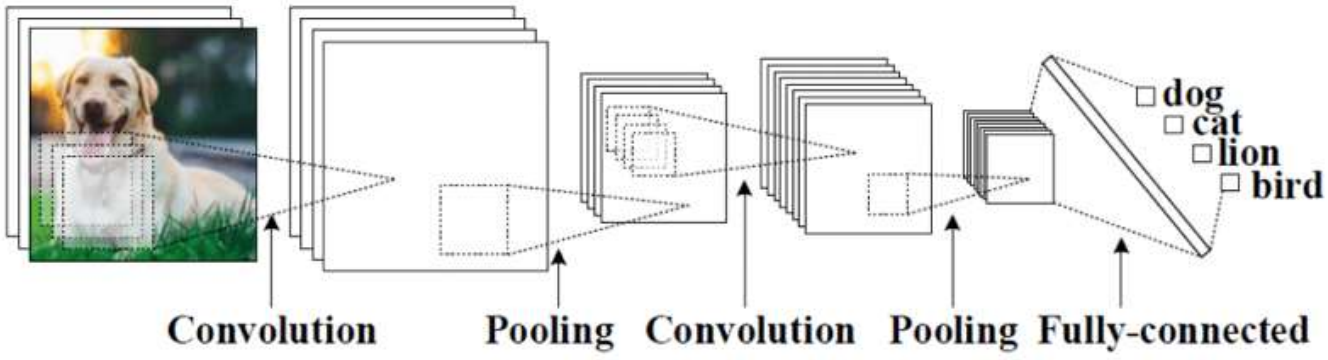

Рисунок 3.2 – Топологія згорткової НМ [31]

Два типи шарів формують вектор міняючись між собою карту ознак для майбутнього перцептрона, дані мережі за недавніми результатами перевищують звичайні НМ приблизно на 15%. Серед розробників згорткова НМ – ключова технологія DP. Концепція загальної ваги – та сама архітектура, яка стала причиною успіху в багатьох тестах з розпізнавання образів. Ця архітектура мережі має велику кількість налаштувань на відміну від архітектури НМ неокогнітроном. Також існує підвид згорткової мережі, яка частково відмовляється від зв'язаних ваг, але при цьому алгоритм залишається той же – зворотне поширення помилки. Згортка НМ може ефективно працювати на будь-яких пристроях та швидко навчатися через паралельні процеси згортки і так само швидко зробити операцію згортки при поширенні помилки при навчанні.

Щоб вибрати топологію для згорткової НМ – потрібно орієнтуватися на поставлене розробником завдання та доступні дані. Існують етапи, які впливають на потребу обирання необхідної для вирішення задачі топології НМ:

– обчислити обмеження у заданій задачі (точність розпізнавання, швидкість та практичність);

– визначити завдання для вирішення її за допомогою НМ;

– вивчити дані та визначити вхідні дані (зображення, звуки, їх розмір, формат) та вихідні дані (кількість класів).

Близько п'яти років тому НМ вважалася дуже ресурсно-витратним алгоритмом, який вимагав потужних обчислювальних машин для навчання та тестування мереж. Але в наш час НМ вже працюють прямо на мобільних телефонах з невисокими вимогами за характеристиками, вбудований у камеру ШІ аналізує довкілля та адаптує налаштування камери для кращої якості фото та визначає об'єкти за чисельні секунди. Тому для вирішення завдання розпізнавання маски на обличчі людини було використано технологію MobileNetV2, яка використовує сучасну архітектуру згорткової мережі.

Звичайна архітектура згорткової НМ є наступним фільтром:

$$D_k \cdot D_k \cdot C_{in}, \tag{3.1}$$

де D_k – розмір ядра згортки,

C_{in} – кількість каналів на вході.

Спільна розрахункова формула для згорткового шару:

$$D_k \cdot D_k \cdot C_{in} \cdot D_f \cdot C_{out}, \tag{3.2}$$

де D_f – висота, ширина вхідного шару НМ,

C_{out} – кількість каналів на виході.

Структура архітектури MobileNet. На наступному малюнку буде зображено структуру блоку звичайної згорткової НМ, а поруч із нею структуру блоку MobileNet без модифікацій (рис. 3.3).

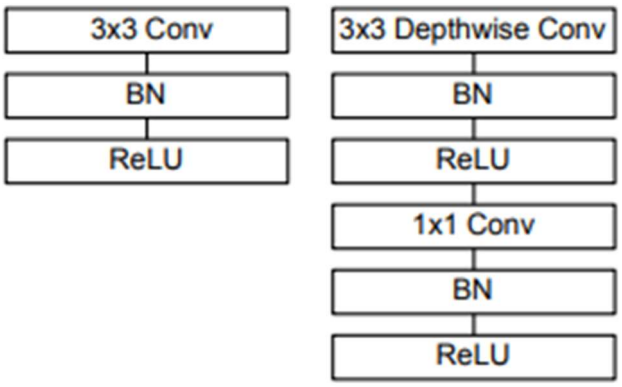

Рисунок 3.3 – Структура блоків звичайної згорткової НМ та MobileNet

Архітектура MobileNet має дві параметри – α (множник ширини) і p (множник глибини). Параметр α у даній задачі відповідає за певну кількість карт ознак на кожному шарі НМ, а параметр множника роздільної здатності – за просторові розміри вхідних даних.

Дані параметри дозволяють керувати розмірами НМ: зменшуючи вхідні параметри множника ширини та роздільної здатності знижується точність розпізнавання, але при цьому навчання нейронної мережі відбувається набагато швидше і зменшується витрата мережею пам'яті, що споживається під час навчання.

MobileNet має лише три версії, найстабільнішою є друга версія даної архітектури (MobileNetV2) – це наступне покоління згорткових НМ від компанії Google. Структура MobileNetV1 проста, але менш продуктивна і має меншу базу даних, на якій будувалася мережа [32].

Практичні результати.

Розробники провели дослідницькі роботи, де взяли кілька варіацій моделей MobileNetV1 та MobileNetV2 з різними параметрами ширини (alpha) та роздільною здатністю (rho) на звичайних мобільних пристроях [33].

Таблиця 3.1 – Порівняння архітектур НМ MobileNetV1 та MobileNetV2

Архітектура мережі	Кількість вхідних параметрів	Точність розпізнавання
MobileNetV1 (alpha=0.25, rho=0.57)	0.47 мільйонів	0.643
MobileNetV1 (alpha=0.75, rho=0.85)	2.59 мільйонів	0.865
MobileNetV1 (alpha=1, rho=1)	4.20 мільйонів	0.895
MobileNetV2 (alpha=0.35, rho=0.43)	1.66 мільйонів	0.706
MobileNetV2 (alpha=1, rho=1)	**3.47 мільйонів**	**0.912**
MobileNetV2 (alpha=1.4, rho=1)	6.06 мільйонів	0.900

Порівняв дані версії архітектури MobileNet можна помітити, що на даний момент мережі мають більш великий процент розпізнавання навіть на мобільних телефонах без великих затрат енергії та ресурсів пристрою. Автори даної архітектури також демонстрували, що їхня SSDLite-архітектура, яка поєднана із згортковою НМ має більш кращі результати, ніж популярна архітектура для розпізнавання об’єктів YOLOv2 (You Look Only Once) по точності на наборі даних MS COCO, демонструючи при цьому в 5 раз менший розмір та більш велику швидкість [34].

3.2 Встановлення необхідних бібліотек та налаштування віртуального оточення

Мова програмування Python має зручний функціонал та можливість використовувати сучасні технології, які прискорюють процес розробки ПЗ. Так само ця мова підтримує парадигму написання кросплатформових додатків.

Перед початком роботи над проектом потрібно встановити інтерпретатор Python, після чого розробник може почати створювати віртуальне оточення для роботи над проектом. Це необхідно для того, щоб при створенні нового проекту встановлені інструменти та фреймворки для одного проекту не перейшли на інший, що може спричинити купу помилок у процесі встановлення. Якщо використовувати одне віртуальне оточення, то можна зіткнутися з проблемою, що на якийсь із фреймворків може вийти оновлення, внаслідок проблем із оновленням може виникнути помилка зворотної сумісності з іншими пакетами. Рекомендується для кожного проекту створювати нове віртуальне середовище. Віртуальне оточення дозволяє створити необмежену кількість віртуальних середовищ на одній операційній системі.

Існує декілька видів віртуального оточення для проектів Python (virtualenv, pyenv, virtualenvwrapper). При створення оточення для системи розпізнавання маски на обличчі людини було обрано інструмент virtualenv, який є одним із популярнішим виборів серед розробників.

Перед створенням самого оточення потрібно встановити virtualenv з використанням пакетного менеджера PIP. Для встановлення необхідних інструментів використовується наступна команда у командному інтерпретаторі:

```
pip install  *package_name*
```

Команда приймає параметр package_name, що визначає ім’я пакету, який потрібно встановити у глобальну версію Python на пристрої.

Після встановлення virtualenv потрібно створити віртуальне оточення, де ми вказуємо версію інтерпретатора Python та назву директорії, де будуть зібрані усі необхідні інструменти [35]:

```
virtualenv --python=python3.9 mask_detector
```

Результат відпрацювання команди зображено на рис. 3.4.

```
E:\4_course(2_semester)\Диплом>virtualenv --python=python3.9 mask_detector
created virtual environment CPython3.9.8.final.0-64 in 3304ms
  creator CPython3Windows(dest=E:\4_course(2_semester)\Диплом\mask_detecto
lse)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=
AppData\Local\pypa\virtualenv)
    added seed packages: pip==21.3.1, setuptools==60.2.0, wheel==0.37.1
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,P
```

Рисунок 3.4 – Створення віртуального оточення virtualenv

Для активації бібліотеки та встановлення інших бібліотек потрібно проробити декілька етапів, які буде описано нижче.

1. Потрібно зайти до директорії, яку створила бібліотека virtualenv, а саме за шляхом “mask_detector\Scripts”, та активувати його за допомогою команди “activate” (рис. 3.5).

```
E:\4_course(2_semester)\Диплом>cd mask_detector\Scripts

E:\4_course(2_semester)\Диплом\mask_detector\Scripts>activate

(mask_detector) E:\4_course(2_semester)\Диплом\mask_detector\Scripts>
```

Рисунок 3.5 – Активація віртуального оточення mask_detector

2. Після активації віртуального оточення можна перейти до завантаження необхідних бібліотек для майбутньої системи (TensorFlow, Keras та OpenCV).

```
(mask_detector) E:\4_course(2_semester)\Диплом\mask_detector\Scripts>pip install
opencv
(mask_detector) E:\4_course(2_semester)\Диплом\mask_detector\Scripts>pip install
keras
(mask_detector) E:\4_course(2_semester)\Диплом\mask_detector\Scripts>pip install
tensorflow
```

Почекавши декілька хвилин пакетний менеджер завантажить усі необхідні пакети для майбутньої розробки. Після усіх завантажень потрібно у IDE PyCharm вказати шлях до необхідної Python версії, після чого можна починати розробку системи.

3.3 Навчання та тестування моделі для розпізнавання медичної маски

Для навчання НМ було написано скрипт “training.py”, який містить у собі наступний метод.

```
train_model(img_path: str, epochs: int, num_photo_per_epoch: int, name_model: str)
```

Даний метод приймає наступні параметри:

- img_path – параметр для шляху до набору даних, на яких НМ буде навчатися;
- epochs – кількість епох для навчання;
- num_photo_per_epoch – кількість фото за одну епоху;
- name_model – ім’я для навченої моделі.

Всередині даного методу також використовується об’єкт класу MobileNetV2, який містить у собі готову архітектуру згорткової НМ.

```
baseModel = MobileNetV2(weights="imagenet", include_top=False,
                        input_tensor=Input(shape=(224, 224, 3)))
```

Для навчання НМ було завантажено готовий набір даних, який складається із зображень людей «у масці» та «без маски». Даний набір даних було створено та розміщено на сайті Kaggle для організації конкурсів з дослідження великого обсягу даних, де збираються усі спеціалісти з машинного навчання [36].

Зібраний набір даних складається з 1920 зображень, розподілених на два потрібних для майбутнього навчання класів (рис. 3.6):

1. with_mask – 966 зображень людей з маскою на обличчі.
2. without_mask – 954 зображень людей без маски на обличчі.

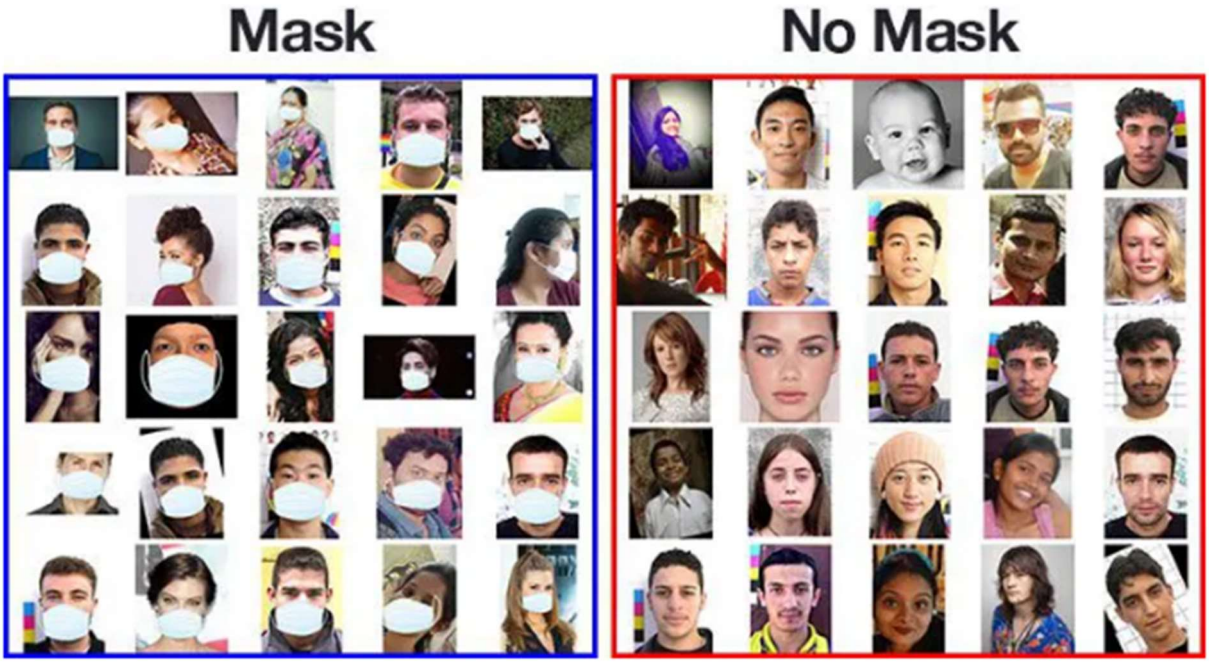

Рисунок 3.6 – Набір даних для навчання НМ [36]

Для початку навчання НМ потрібно у відкритому віртуальному оточенню запустити код Python, передав у нього необхідні параметри, що вказані нижче.

```
train_model(img_path="E:/4_course(2_semester)/Диплом/mask_detection/dataset",
epochs=20, num_photo_per_epoch=50, name_model="MaskDetector_20_epochs")
```

Після запуску коду починається збір даних та йде процес навчання НМ (рис. 3.7).

```
E:\4_course(2_semester)\Диплом\mask_detection\venv\lib\site-packages\keras\optimizer_v2\adam.py:105: UserWarning: The `lr` argument is deprec
  super(Adam, self).__init__(name, **kwargs)
[INFO] training head...
Epoch 1/20
2022-05-16 21:25:10.308298: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 154140672 exceeds 10% of free system memory.
2022-05-16 21:25:10.398672: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 156905472 exceeds 10% of free system memory.
 1/34 [..............................] - ETA: 7:33 - loss: 0.8340 - accuracy: 0.46882022-05-16 21:25:12.697215: W tensorflow/core/framework/c
2022-05-16 21:25:12.890774: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 156905472 exceeds 10% of free system memory.
 2/34 [>.............................] - ETA: 1:01 - loss: 0.9041 - accuracy: 0.40622022-05-16 21:25:14.742754: W tensorflow/core/framework/c
34/34 [==============================] - 77s 2s/step - loss: 0.5515 - accuracy: 0.7500 - val_loss: 0.2302 - val_accuracy: 0.9891
Epoch 2/20
34/34 [==============================] - 55s 2s/step - loss: 0.2159 - accuracy: 0.9625 - val_loss: 0.0982 - val_accuracy: 0.9891
Epoch 3/20
34/34 [==============================] - 55s 2s/step - loss: 0.1158 - accuracy: 0.9813 - val_loss: 0.0603 - val_accuracy: 0.9891
Epoch 4/20
34/34 [==============================] - 55s 2s/step - loss: 0.0796 - accuracy: 0.9841 - val_loss: 0.0451 - val_accuracy: 0.9964
Epoch 5/20
34/34 [==============================] - 57s 2s/step - loss: 0.0610 - accuracy: 0.9916 - val_loss: 0.0342 - val_accuracy: 1.0000
Epoch 6/20
```

Рисунок 3.7 – Процес навчання НМ на архітектурі MobileNetV2

На рис. 3.7 можна побачити наступну інформацію:

– кількість епох;
– час, який витрачається на одну епоху;
– кількість втрат під час навчання (в значенні від 0 до 1);
– точність розпізнавання (в значенні від 0 до 1).

Після усіх епох виводиться статистика навчання (рис. 3.8):

– precision –точність;
– recall – повнота, а саме як НМ знаходить усі правильні відповіді;
– f1-score – дане значення інтерпретується як гармонічне середнє між точністю та згадуванням образів

Потім створюється файл із назвою, яку було зазначено у методі при запуску навчання у розширенні“.h5” (hierarchical data format) для збереження бінарних даних. Даний формат файлів надає можливість читати/записувати навчені моделі.

```
[INFO] evaluating network...
                                                    precision    recall  f1-score

   E:/4_course(2_semester)/Диплом/mask_detection/dataset\with_mask       1.00      1.00      1.00
E:/4_course(2_semester)/Диплом/mask_detection/dataset\without_mask       1.00      1.00      1.00

                                                     accuracy                           1.00
                                                    macro avg       1.00      1.00      1.00
                                                 weighted avg       1.00      1.00      1.00

[INFO] saving mask detector model...
```

Рисунок 3.8 – Результат навченої моделі

Для якості прогнозування використовується метод F-score. У задачі розпізнавання маски є кілька кінцевих результатів. Для цього будуть розглянуті такі позначення (рис. 3.9) [37]:

– True Positive – справжнє позитивне рішення НМ. Маска була виявлена;
– True Negative – справжнє негативне рішення НМ. Мережа стала приймати руку, яка закривала рот і ніс людини, як маску;

– False Positive – помилкове позитивне рішення НМ. Мережа визначила будь-який інший об'єкт як маску.

– False Negative – хибне негативне рішення НМ. Мережа не стала розпізнавати об'єкт на обличчі людини, але це була маска;

– Precision – наскільки добре НМ знаходить об'єкт на вхідному зображенні з мінімальними втратами;

– Recall – як правильно НМ визначає маску на обличчі людини.

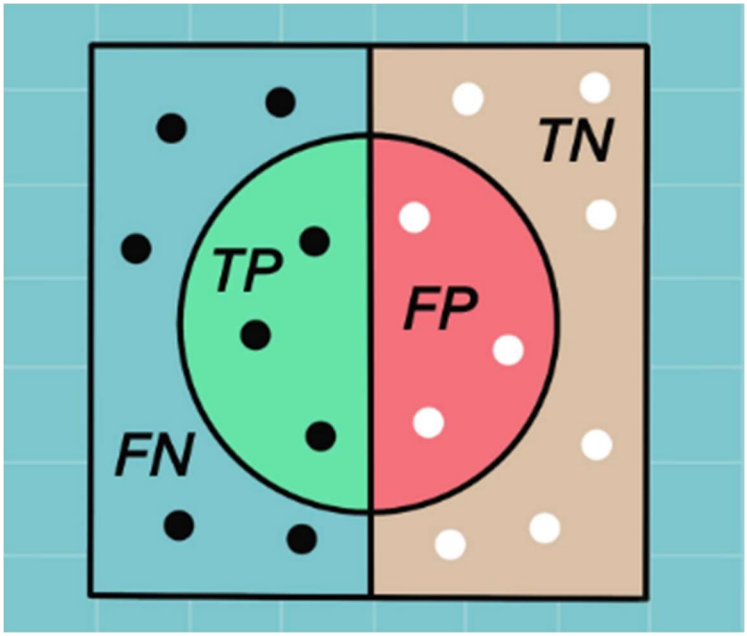

Рисунок 3.9 – Оцінка якості НМ F-score [37]

Мета при навчанні НМ – це мінімізація хибного негативного рішення, яке порушує точність розпізнавання. Значення precision, recall та F1-score (яке містить у собі точність роботи та повноту алгоритму) розраховуються за наступними формулами, які вказані нижче.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}, \tag{3.3}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}, \tag{3.4}$$

$$F1 - score = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}, \quad (3.5)$$

Змінна precision, яку зазначено у формулі 3.3 призначена для того, щоб НМ не брала зайві деталі під час навчання, а змінна recall у формулі 3.4 – щоб не впустити потрібні деталі.

Спосіб валідації навчальної вибірки. Для навчання НМ розпізнавати маски на особі людини вибралося 20% даних для вибірки валідації після успішного завершення навчання, а решта 80% - використовуються для навчання НМ (рис. 3.10). Цей метод краще ручного формування, тому що він бере автоматично відокремлює зазначений відсоток даних для валідації, з недоліків можна виділити тільки те, що у разі спостережень за навчанням на конкретних моментах може стати в нагоді інформація про зображення, які були відібрані для валідації.

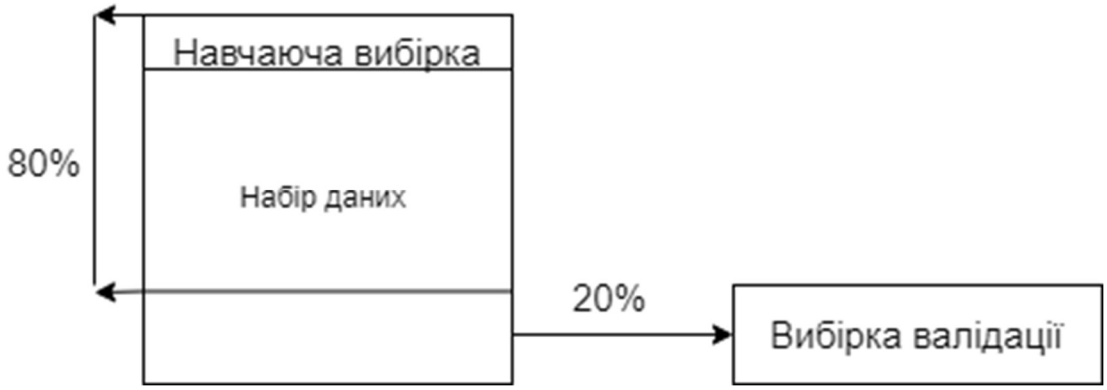

Рисунок 3.10 – Валідація навчальної вибірки

На вхідний шар подаються дані, саме зображення розширення ".jpeg" 248x248 пікселів. Для цього в скрипті перед навчанням зображення обробляються, а потім подаються на вхід НМ. Якщо розмір вхідних даних буде великим, то розрахунки вимагатимуть більше часу та ресурсів відповідно. Але якщо зменшити фото до маленького розміру, то НМ не зможе визначити ознаки маски на обличчі людини і відокремити її як ключовий елемент. При подачі на вхід зображення розбивається на 3 канали RBG (червоний, синій, зелений).

Вхідний шар нормалізує вхідні дані кожного пікселя у певний діапазон (від 0 до 1) за наступною формулою:

$$f(p, min, \max) = \frac{p - min}{\max - min}, \tag{3.6}$$

де f – функція нормалізації,

p – колір пікселя від 0 до 255;

min – мінімальне значення пікселя (тобто 0),

max – максимально значення пікселя (тобто 255).

Згортковий шар - це набір карт ознак, кожна карта містить у собі синаптичне ядро (або фільтр). Кількість карт залежить безпосередньо від завдання, якщо взяти велику кількість - підвищиться якість розпізнавання, але при цьому збільшиться час навчання. Частим вибором є співвідношення 1 до 2. Кожна карта попереднього згорткового шару пов'язана з двома картами наступного згорткового шару (рис. 3.11).

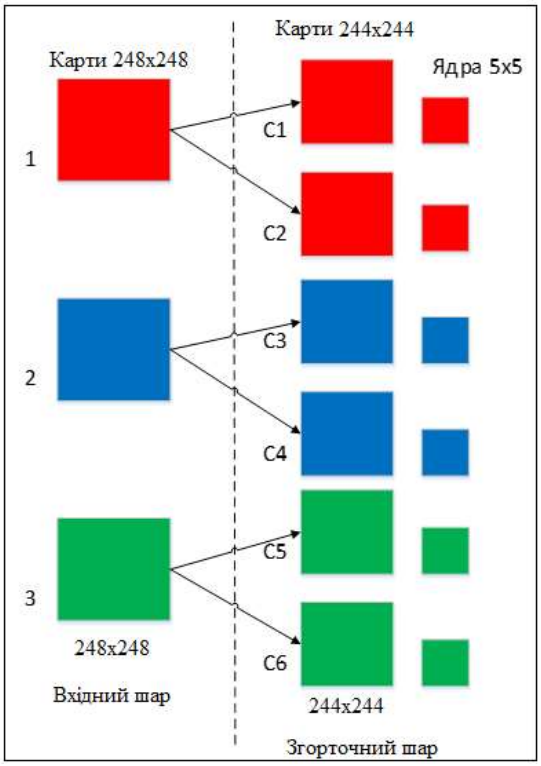

Рисунок 3.11 - Зв'язки карт згорткового та вхідного шару

Розмір згорточного шару визначається може бути обчислений за допомгою наступної формули:

$$(w, h) = (mW - kW + 1, mH - kH + 1), \quad (3.7)$$

де *(w, h)* – розмір згорткової карти,

mW – ширина попередньої карти,

kW – висота попередньої карти,

mH – ширина ядра,

kH – висота ядра.

Саме ядро є розділеною системою синапсів (вагів). Розмір ядра обирається таким чином, щоб розмір цих карт був парним. Ця конструкція дозволяє не втратити важливу інформацію при зменшенні розмірності вхідного зображення при пошуку ознак. За замовчуванням значення карти згорткового шару дорівнює нулю. Їхні ваги задаються випадково в області від -0.5 до 0.5.

Підвибірковий шар. Цей шар має карти, які за кількістю сходяться з розміром попереднього згорткового шару. Якщо після попередньої операції згортки мережею було виявлено потрібні ознаки для майбутнього розпізнавання - підвибірковий шар докладно не обробляє наступне зображення і ущільняється до менш докладного. Також це є великою перевагою, адже опускається фільтрація непотрібних деталей, що дозволяє не перевчитися НМ [38].

Пошукове ядро в картах ознак – це матриця 2x2, як надає можливість НМ зменшити попередні карти ознак вдвічі, що прискорює процес навчання НМ. З отриманих осередків вибирається максимальне значення. Основою роботи шару підвиборки – використання функції активації RelU. На рис. 3.12 зображено операцію підвиборки.

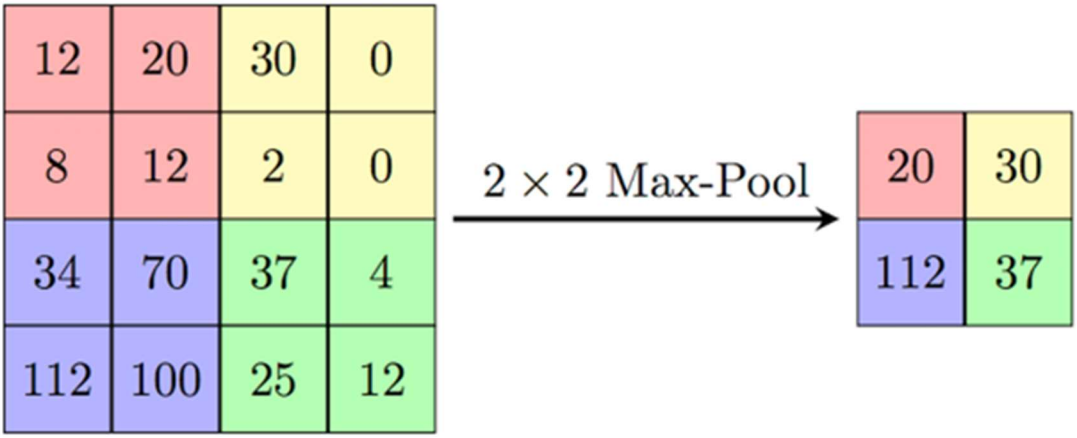

Рисунок 3.12 – Зменшення зображення у шарі підвиборки за рахунок попереднього шару з використанням операції підвиборки max-pooling

Вихідний шар НМ. Цей шар має зв'язок з усіма нейронами минулого шару, кількість нейронів на виході відповідають кількості класів, які були заздалегідь визначені, тобто 2 - людина в масці та людина без маски. Для зменшення кількості зв'язків згорткова НМ використовує один нейрон як функцію активації якщо нейрон на виході видає значення 1 - то людина перед камерою знаходиться в масці, а якщо значення -1 - людина перед камерою без маски відповідно.

Висновок до розділу 3

Згорткові НМ мають архітектуру, яка близька до будови мереж людського мозку. Мережа розділена на області, які не виділяються яскравою спеціалізацією, обробляючи однаково інформацію (звукову, зорову, логічну). Вивчивши дані нейронні мережі, можна зрозуміти їх всю перевагу в завданнях розпізнавання. Одна з переваг згорткових НМ - ознаки у вхідних зображеннях виявляються окремо. Дані мережі можуть вивчити основні ознаки в наборі даних знаходити головний об'єкт ігноруючи весь шум, що може стати на заваді роботі мережі. Також можна відзначити універсальність цих мереж у багатьох завданнях: модель може бути доповнена, розширена іншими алгоритмами або методами для завдань, які раніше могли бути не призначені для такого типу мережі.

Ядро у згорткової НМ виконує функцію фільтрації, тобто знаходить ознаки з усього зображення. Підвибірковий шар збільшує швидкість обчислень через зменшення розмірності карт попередніх шарів, фільтрує непотрібні деталі у разі успішного розпізнавання об'єктів.

Використовуючи передову архітектуру MobileNetV2, можна отримати необмежений функціонал у галузі навчання НМ та комп'ютерного зору. Крім простих моделей класифікації, що входять до цієї архітектури, MobileNetV2 можна використовувати на будь-яких пристроях (включаючи мобільні телефони) без втрати ефективності та трудомісткості. При цьому архітектура MobileNet підтримує використання таких технологій як TensorFlow та Keras, просто підібравши потрібні параметри для навчання мережі. Також дана архітектура за численними розмірами перевершує свого пращура MobileNet за швидкістю роботи та витратою меншої кількості ресурсів.

Налаштувавши віртуальне оточення та встановивши необхідні бібліотеки через пакетний менеджер PIP (TensorFlow, Keras, OpenCV та інші) можна приступати до розробки користувальницького графічного інтерфейсу та додаткового функціоналу для системи розпізнавання масок.

Віртуальне оточення забезпечить незалежність бібліотек, які можуть бути присутніми або відсутніми на пристрої кінцевого користувача та забезпечує кросплатформенність і безпеку за рахунок контейнера, в якому буде проходити розробка.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ МАСКИ НА ОБЛИЧЧІ ЛЮДИНИ

4.1 Tkiner як технологія для створення графічного застосунку

Графічний інтерфейс користувача (англ. Graphical User Interface) – це необхідна система взаємодії кінцевого користувача з комп'ютером як графічних компонентів. За допомогою цих компонентів користувач зможе запустити програму на рівні візуалізованої інформації, що допоможе людині швидше опанувати систему.

З переваг використання ГІК можна виділити такі:

– ГІК дуже добре приймається звичайними користувачами ПК, що тільки розпочали свою роботу;

– підтримується усіма видами ОС.

Тому для розробки графічного інтерфейсу було обрано бібліотеку Tkinter, що входить до стандартного набору при встановленні мови програмування Python. Використовуючи цю бібліотеку можна легко і швидко створити макет майбутнього інтерфейсу, прив'язати потрібний функціонал до кнопок і дана бібліотека забезпечує максимальну швидкодію при роботі з користувачем за рахунок деяких функцій, написаних на C++.

Tkinter має базовий набір шаблонів та компонентів, що схожий на PyQT. Ця властивість дає можливість розробнику створити інтерфейс для існуючої системи та приєднати весь функціонал, розроблений раніше без використання ГІК.

Для системи розпізнавання маски на обличчі людини було розроблено наступний інтерфейс, що складається з двох сторінок меню, перемикатися між якими можна за допомогою відповідної кнопки (рис. 4.1).

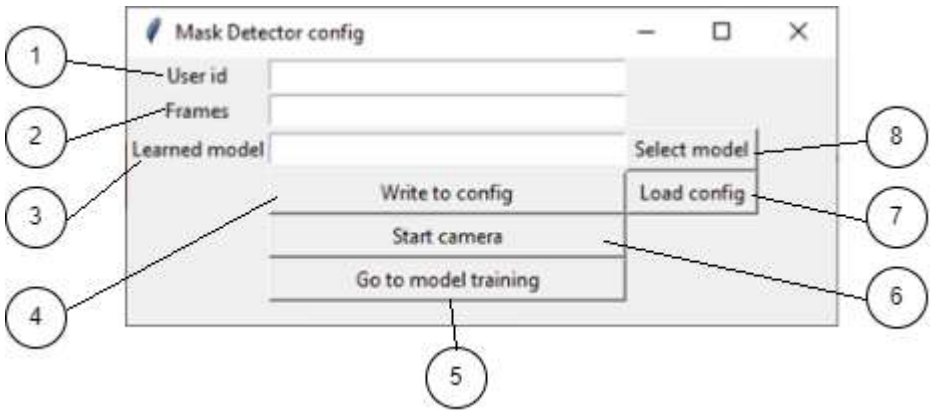

Рисунок 4.1 – Перша сторінка інтерфейсу системи

На рис. 4.1 можна виділити наступні елементи.

1) ID користувача у месенджері Telegram, куди при роботі системи будуть приходити сповіщення про людину без маски.

2) Кількість кадрів у секунду під час роботи камери для розпізнавання наявності маски на обличчі.

3) Навчена модель, яку буде використовувати НМ для розпізнавання маски на обличчі.

4) Кнопка для збереження вищенаведених даних, які можуть змінюватися.

5) Перехід до другої сторінки інтерфейсу.

6) Почати роботу камери для розпізнавання.

7) Завантажити останні збережені дані з файлу налаштування.

8) Обрати навчену модель НМ.

Друга сторінка включає елементи управління для навчання НМ з налаштуванням необхідних параметрів, для чого було розроблено наступний інтерфейс (рис. 4.2).

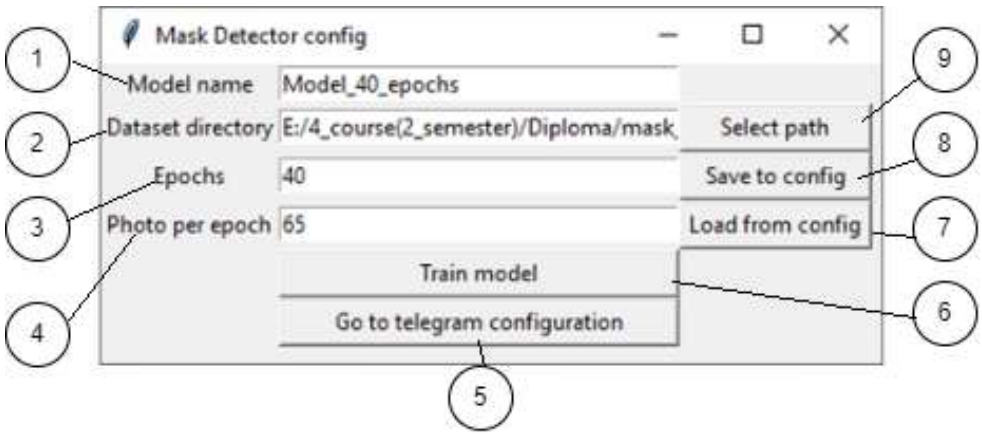

Рисунок 4.2 – Друга сторінка інтерфейсу

На рис. 4.2 зображені наступні елементи системи.

1) назва моделі НМ;
2) шлях до набору даних, на якому буде шукати ознаки та вчитись НМ;
3) кількість епох для навчання;
4) кількість фото на кожну епоху;
5) перехід до першої сторінки інтерфейсу;
6) почати навчання моделі;
7) загрузити дані з файлу конфігурації;
8) завантажити введенні дані;
9) обрати шлях через провідник.

Головна вимога до ГІК - реалізація однієї з найвідомішої концепції "Do what I mean". Дана концепція стверджує, що ГІК повинен функціонувати передбачувано для користувача, щоб він розумів, що відбудеться після введення певної інформації або дії (натискання на кнопку, свайпу, введення).

4.2 База даних для ведення статистики

При створенні системи кінцевому користувачеві може знадобитися збір інформації та формування необхідної статистики, щоб зрозуміти усю ситуацію і як діяти далі. Для зберігання можна використовувати реляційні бази даних. Дані бази

дозволяють асинхронно записувати велику кількість інформації, кодувати, видозмінювати та імпортувати в інші середовища, а також приєднувати їх до систем у процесі розробки. Тому для запису статистики кількості людей або маски була обрана компактна вбудована база даних SQLite. Для впровадження конектора у систему проект використовуватиметься стандартна вбудована бібліотека Python sqlite3. Серед переваг даної бази даних можна виділити таке:

– мінімалізм системи, який дозволяє легко та швидко створити потрібну таблицю без довгого вивчення системи;

– висока швидкість обробки запитів через використання асинхронності;

– кросплатформенність, що дає можливість працювати на різних версіях та видах ОС без змін;

– мінімальний розмір системи дозволить заощадити місце під час розгортання бази даних;

– доступність дає можливість об'єднати з багатьма мовами програмування (Python, C#, Java і т.д.);

– зберігання всіх записів в одному файлі дозволить легко мігрувати та переміщувати збережені дані без втрат.

Цю базу даних краще використовувати коли потрібно об'єднати запити та зберігання SQL запитів і мати звичайний доступ до системи файлів. Для запису наявності маски або її відсутності потрібен максимально швидкий та продуктивний відгук системи, а також запис у базу. Для цього з наведених вище переваг можна виділити продуктивність бази, що дозволить без витрат пам'яті не навантажуючи систему зробити швидкий запис. Таблиця, у якій писатиметься статистика матиме просту структуру.

Таблиця 4.1 – Структура бази даних для ведення статистики

Колонка	Тип даних
ID	integer

Закінчення таблиці 4.1

With mask	integer
Time	datetime
In mask	integer

За допомогою мови програмування Python можна легко створити конектор, який впроваджується прямо в код і в систему без особливих надбудов та змін.

4.3 Створення додаткового функціоналу для системи

Система розпізнавання масок на обличчі людини також оснащена додатковим функціоналом. Перша функція - оповіщувач адміністратора або охоронця про порушника. Реалізувати цю функцію можна за допомогою TelegramBotApi - вбудований інтерфейс на основі HTTP, створений спеціально для розробників від популярного месенджера Telegram, який призначений для написання ботів.

Цей інтерфейс має відкритий вихідний код, що дозволяє розробнику безпосередньо ознайомитися з проектом та використовувати його в комерційних цілях. Написати бота можна багатьма мовами програмування (Python, Java, C#, C++, JavaScript). На даний момент вийшла версія API 6.0, що несе у собі важливі оновлення, а саме:

– підтримка webapps, що дозволить розробникам, крім написання бота, прив'язувати HTML-сторінки разом з JavaScript кодом, що значно розширює можливості інтерфейсу;

– додано перелік класів для асинхронної обробки та запис запитів до баз даних;

– підтримка відеостікерів;

– підтримка захисту даних та самих роботів протоколами захисту;

– робота тільки з протоколом HTTPS для шифрування мережі, що забезпечить безпеку користувача під час роботи з ботом.

Також інтерфейс можна впровадити прямо в систему розпізнавання масок, що дозволить розробнику не створювати "міст" для спілкування системи з API Telegram. Інтерфейс дозволяє також розробляти наступних ботів з використанням інших відкритих API:

- бот Gmail для отримання листів;
- gif-бот для створення GIF-зображень;
- бот для музики;
- YouTube бот для завантаження відео;
- для створення ігор та інше.

Для створення робота використовується вже створений раніше розробниками Telegram бот "BotFather". Для цього потрібно через готові кнопки вигадати ім'я боту, опис, коротке ім'я для майбутнього пошуку в Telegram і API-token для майбутнього впровадження в систему (рис. 4.3).

Рисунок 4.3 – Створення боту у месенджері Telegram

Після створення боту даний API-token записується до файлу конфігурації системи та використовується далі для його контролю. Бот повідомляє адміністратора або охоронця в телеграмі (за вказівкою певного user_id в налаштуваннях системи). Так само через цього бота при потребі користувач може дізнатися загальну статистику будь-коли просто ввівши необхідну команду. Результат роботи робота зображено на рис. 4.4.

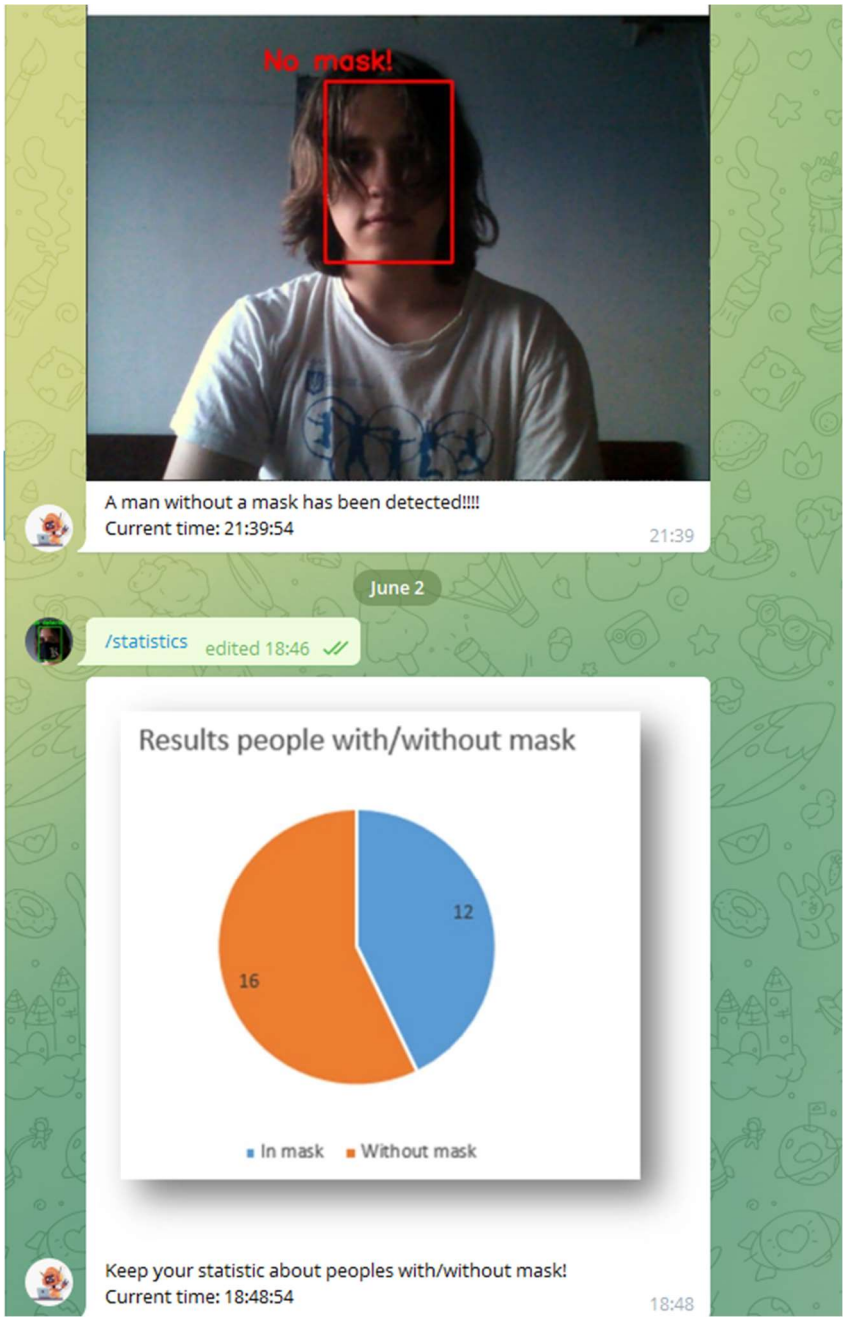

Рисунок 4.4 – Результат роботи бота

Наступна функція – звукове сповіщення. Ця функція необхідна для забезпечення засобів сповіщення людини про перепустку або заборону проходження до громадського закладу. Функція оповіщення може бути реалізована за допомогою доступних бібліотек для обробки звуку (PYO, PyAudio, Dejavu, PyDub, speech-dispatcher).

Для виклику звукового сповіщення потрібно створити два об'єкти класу winsound.

```
import winsound
duration = 2000
winsound.PlaySound(“Alert”, duration)
winsound.PlaySound(“Confirmation”, duration)
```

Дані об’єкти приймають наступні параметри:

– duration – тривалість звуку (де 1000 мс = 1 секунда);

– “Alert” – звук попередження людини про відсутність маски;

– “Confirmation” – звук дозволу проходу у приміщення.

Дані звукові сигнали будуть відтворюватися один раз для попередження про відсутність маски або дозволу на прохід.

4.4 Розгортання системи, її можливості

Для розгортання проекту на кінцевій машині користувача існує два шляхи:

– запустити ".exe" формат файлу для простого запуску;

– вивантажити код з GitHub та за інструкцією (Readme.txt файл) виконати етапи встановлення.

Не кожен користувач може запускати скрипти з віртуального оточення або робити складні запуски з множиною передачі параметрів. Мова програмування Python має можливість переробити код на бінарний файл (".exe", ".bat"), всередині якого знаходиться інтерпетований код, який може запускатися не залежно від того,

встановлена мова програмування або якісь бібліотеки. Для реалізації цієї можливості потрібно встановити бібліотеку pyinstaller, яка дозволить за допомогою наступної команди створити ".exe" формат файлу без додаткових операцій.

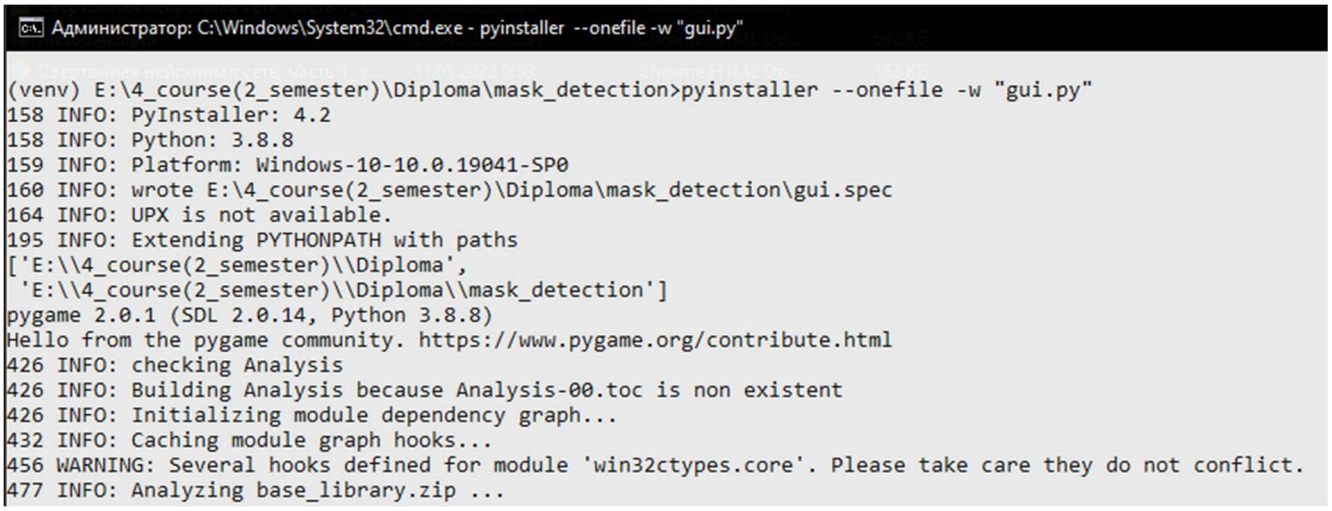

Рисунок 4.5 – Процес створення “gui.exe” файлу

Після завершення генерації “gui.exe” у директорії проекту створюється папка з відповідним файлом (рис. 4.6).

Рисунок 4.6 – Готовий “gui.exe” файл

Цей файл забезпечує повний функціонал і працює без обмежень, що дозволяє переносити його на різні пристрої без будь-яких додаткових налаштувань та коригувань у коді.

Другий варіант – завантаження вихідного коду з GitHub. Для цього потрібно з офіційного репозиторію проекту вивантажити всі необхідні файли та встановити інтерпетатор Python версії 3.9 (усі необхідні специфікації вказані у файлі Readme.txt) та PIP 20.04.13. Після завантаження потрібно створити віртуальне оточення з будь-якою назвою (наприклад “mask_detector_venv”) та активувавши його - встановити всі бібліотеки, які вказані у файлу requirements.txt.

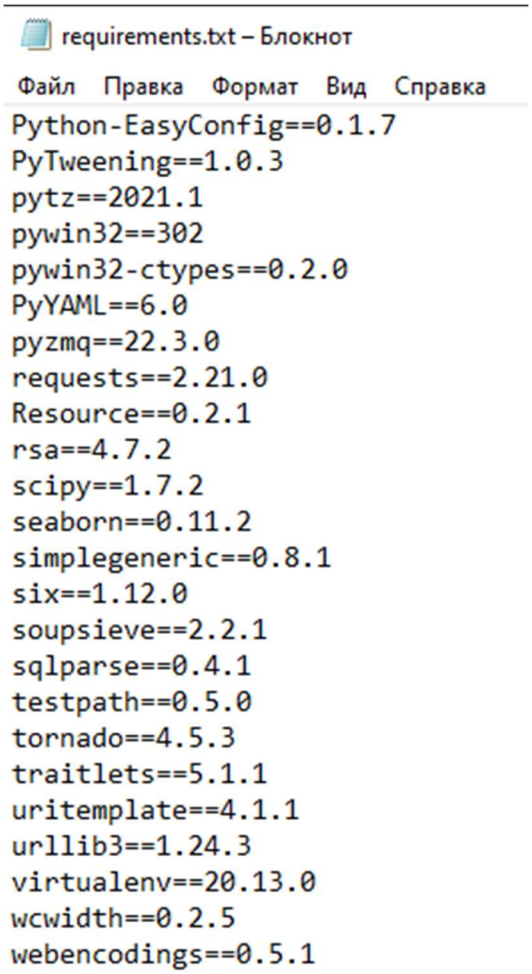

Рисунок 4.7 – Вміст файлу requirements.txt

Після всіх налаштувань користувачеві треба запустити через командний рядок скрипт, ввівши наступну команду.

```
python gui.py
```

Мова Python знаходить сама інтерпретатор із змінних системного оточення, а другим параметром передає ім’я скрипту, який потрібно запустити. Після запуску користувач бачить наступне меню, де можна вказати параметри та запустити камеру. Приклад параметрів, опис меню та їх використання з повним описом вказано у п.п. 4.1. Процес розпізнавання маски на обличчі буде показано використовуючи готові параметри з файлу конфігурацій (модель, яка навчена 40 епохами).

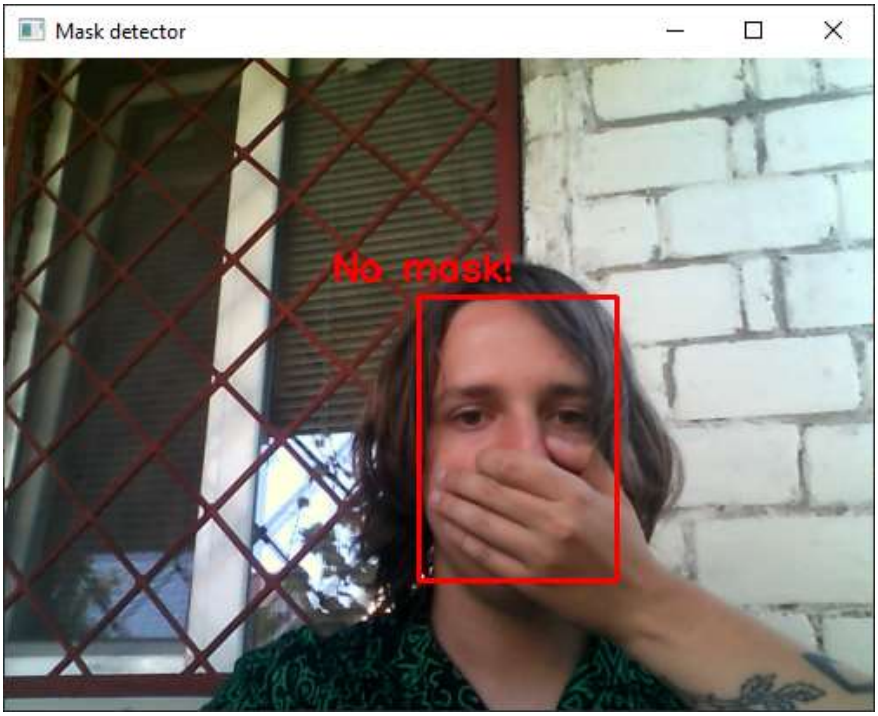

Рисунок 4.8 – Спроба обійти захисну систему розпізнавання маски на обличчі людини прикрив ніс та рот рукою

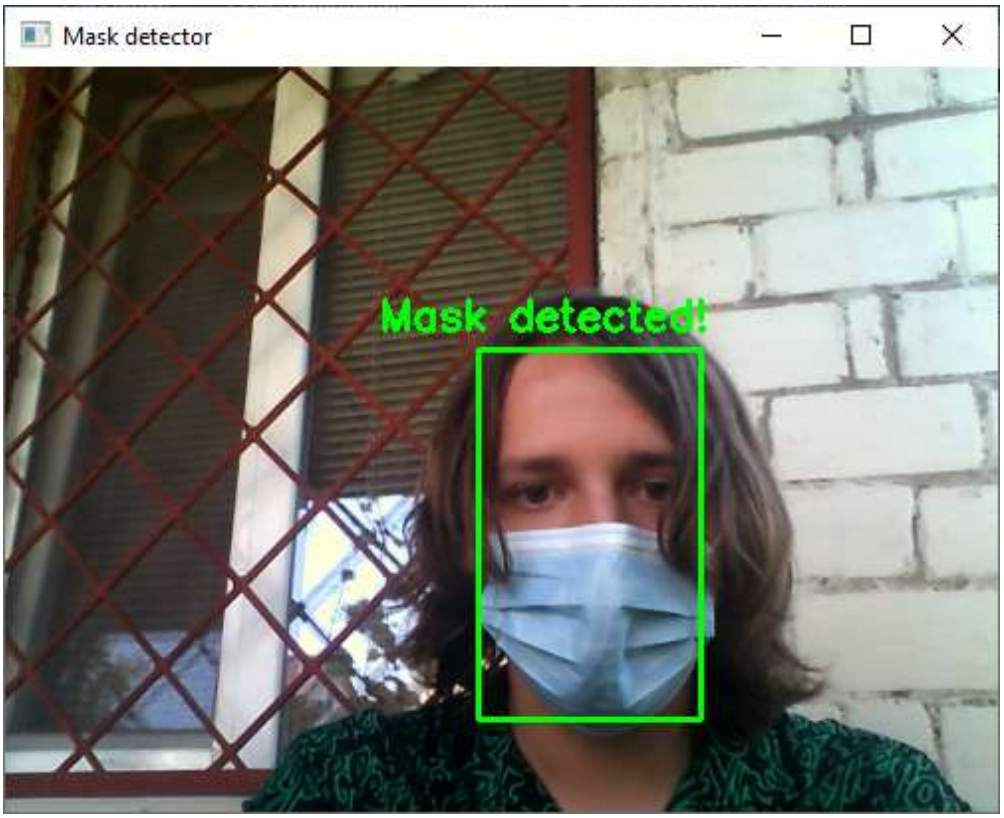

Рисунок 4.9 – Результат розпізнавання обличчя у масці

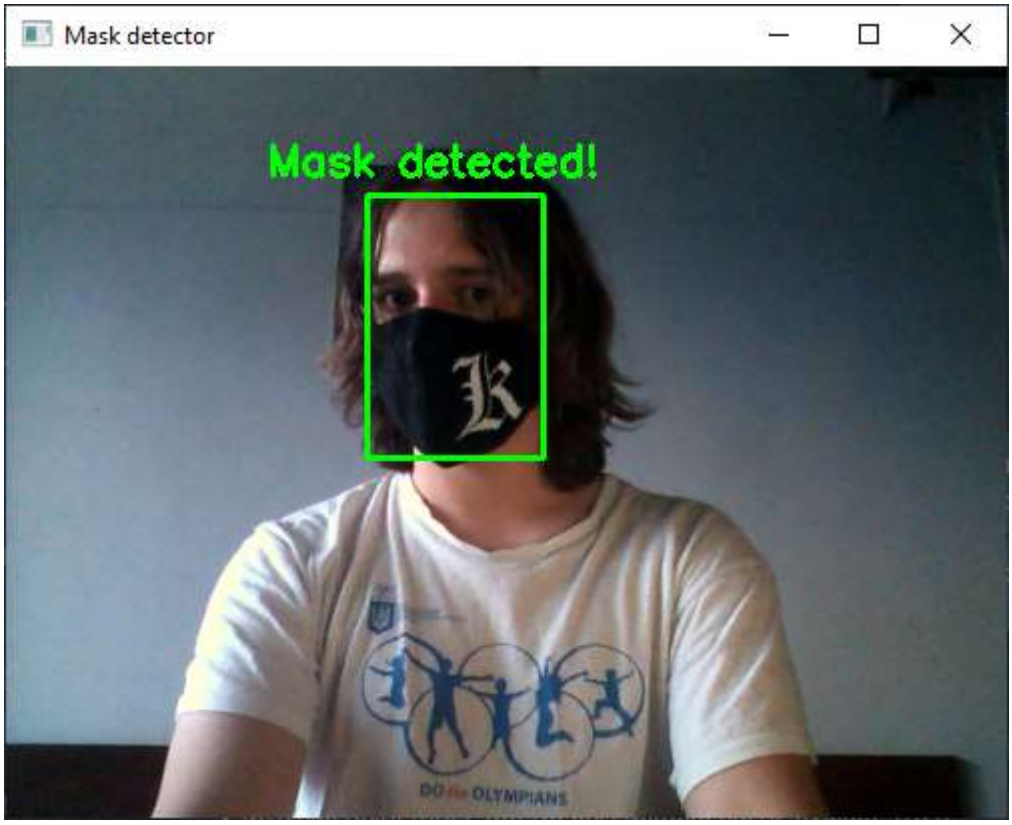

Рисунок 4.10 – Розпізнавання маски іншого кольору з декоративними елементами

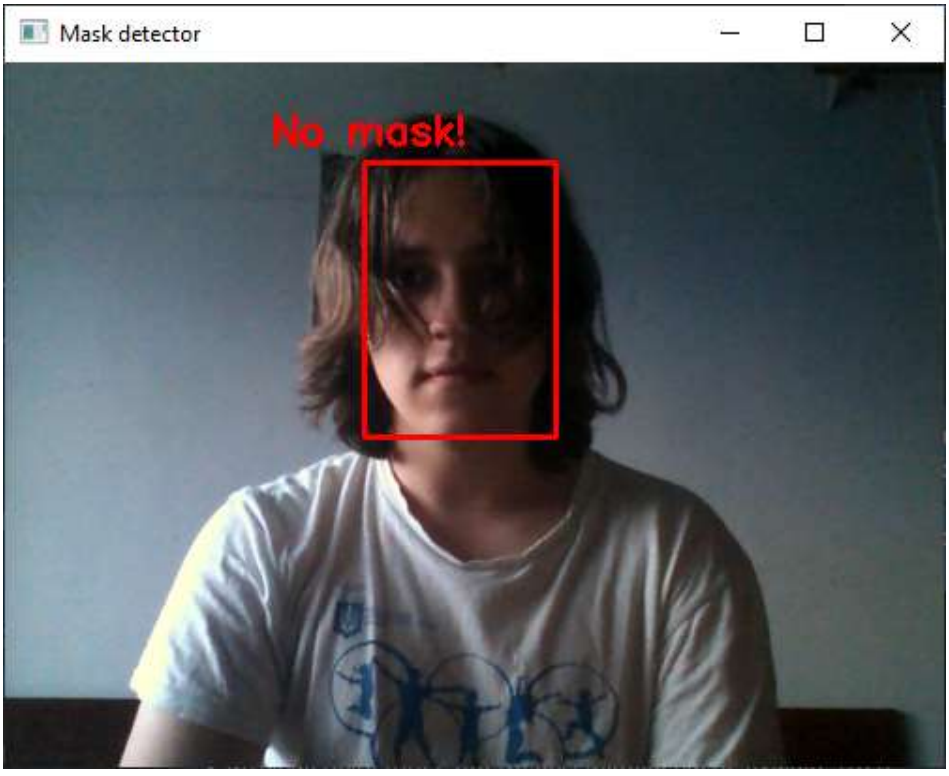

Рисунок 4.11 – Результат розпізнавання відсутності маски на обличчі

Також користувач може навчити НМ розпізнавати з більшою кількістю епох, іншою назвою або набором даних (рис. 4.12).

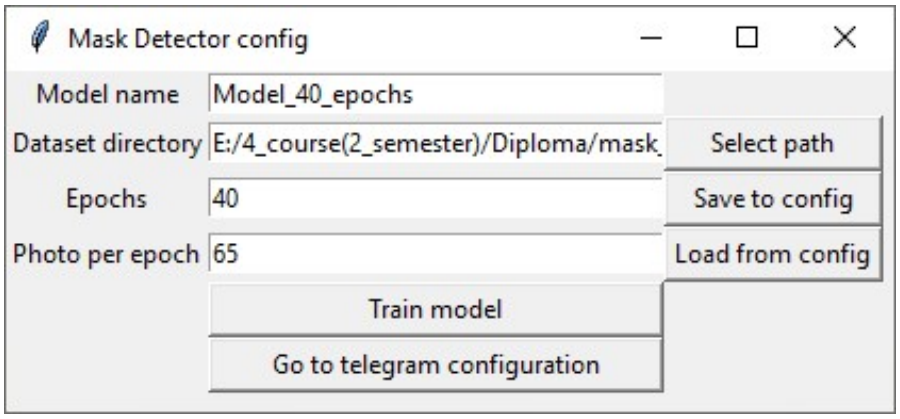

Рисунок 4.12 – Меню налаштування навчання НМ

Процес навчання НМ з вищевказаними параметрами вказано у п.п. 3.3.

Висновок до розділу 4

Tkinter – вбудована бібліотека в інтерпретатор Python, що дозволяє сконструювати ГІК для користувача використовуючи модульні компоненти та готові напрацювання системи. Tkinter реалізований мовою Python, що дозволяє розробляти ГІК для переліку ОС, що підтримуються цією мовою. Також дана система при розробці була забезпечена додатковим функціоналом, що була відсутня зовсім або частково в аналогічних системах.

ГІК має простий та зручний інтерфейс для користувача. Перша сторінка містить повністю інформацію про вибір навченої моделі НМ, налаштування камери та підключення телеграм-бота для оповіщення адміністрації або охорони людини без маски. Спеціально під систему була розроблена база даних для ведення статистики про кількість людей в масці або без неї, доступ до якої може отримати користувач через телеграм бота.

Друга сторінка ГІК призначена для навчання нейронної мережі, де можна задати ім'я, кількість епох, набір даних та кількість фото за одну епоху. Система була протестована на розпізнавання маски на обличчі та додатковий функціонал.

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«СИСТЕМА РОЗПІЗНАВАННЯ НАЯВНОСТІ МАСКИ НА ОБЛИЧЧІ ЛЮДИНИ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810121

Виконав студент 4-го курсу, групи 401
М. О. Салютін
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Консультант *старший викладач*
(наук. ступінь, вчене звання)
О. В. Макарова
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

5 АНАЛІЗ ВИМОГ ДО ОРГАНІЗАЦІЇ ЗАХОДІВ ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНИ ПРАЦІ ПІД ЧАС ЕПІДЕМІЇ COVID-19 ТА РОБОТИ ЗА КОМП'ЮТЕРОМ

Після швидкого поширення нового випадку коронавірусної хвороби (COVID-19) в Ухані, Китай, у грудні 2019 року Всесвітня організація охорони здоров'я (ВООЗ) підтвердила, що це небезпечний вірус, який може передаватися від людини до людини повітряно-краплинним шляхом.

Що стосується профілактики, кожна людина повинна носити маску для обличчя, дотримуватися соціальної дистанції, уникати місць скупчення людей, а також завжди підтримувати імунну систему. Тому, щоб захистити один одного, кожна людина повинна правильно носити захисну медичну маску, коли вона знаходиться у закритому просторі з дистанцією меншою ніж 2 метри між людьми. Однак деякі безвідповідальні люди відмовляються носити маску незважаючи на усю небезпеку вірусу.

Так як під час епідемії було запроваджено карантинні обмеження, серед яких є період самоізоляції та локдаун, то людині не залишається нічого окрім проведення часу перед комп'ютером. В результаті довгострокової роботи за комп'ютером може погіршуватися самопочуття, виникати головні болі та запаморочення, виникати оніміння спини та кисті, проблеми із зором та багато інших проблем

Метою даного розділу є створення вимог безпечних і здорових умов під час епідемії COVID-19 на робочому місці, для досягнення даної мети було виділено наступні **завдання**:

- забезпечення нормативних санітарно-гігієнічних умов праці;
- забезпечення працівників засобами індивідуального захисту (ЗІЗ);
- визначення правил поведінки при загрозі захворюванням вірусом COVID-19;
- аналіз рекомендацій ВООЗ задля забезпечення безпечних умов;

– формулювання вимог для роботи із ПК.

5.1 Нормативні правила та вимоги ВООЗ при епідемії COVID-19

Офіційні заяви щодо появи коронавірусу вперше прозвучали наприкінці 2019 року у зв'язку з випадками зараження на ринку морепродуктів Хуанань у китайському Ухані. Якщо припустити, що деякі проби або аналізи були зроблені в листопаді 2019 року і якщо вони були перевірені вже після того, як було розроблено методи діагностики нового коронавірусу (геном якого було розшифровано приблизно 9-10 січня, а 12 січня було вже надано медичній громадськості, про що повідомляла ВООЗ на своєму сайті), то, справді, за цими пробами можна діагностувати появу хвороби ще за місяць до офіційної появи про хворобу [39].

Вже у 2022 році людям вдалося виділити головні методології та заходи для боротьби із даним вірусом, але це не дало дуже сильного приросту у подоланні небезпечного вірусу COVID-19. Тому ВООЗ поновлює правила поведінки та заходи для забезпечення безпечних з мінімізацією шансу захворювання. Серед головних правил обмежень задля запобігання поширенню ГРВІ COVID-19 можна виділити наступні [40]:

– при перших симптомах респіраторних захворювань (підвищеній температурі, появі кашлю, нежитю або ускладненні дихання) негайно звертайтеся за медичною допомогою. Обов’язково зателефонуйте своєму сімейному лікарю;

– залишайтеся вдома. Самоізоляція - найдієвіший спосіб убезпечити себе від інфікування;

– амбулаторне (домашнє) лікування суворо заборонене у разі, якщо людина перебуває в групі ризику або має наступні симптоми: утруднене дихання, кровохаркання; нудоту, задуху, блювання, сплутаність свідомості;

– частіше провітрюйте приміщення. Доступ чистого повітря перешкоджає розповсюдженню вірусів;

– усі поверхні: дверні ручки, стільниці, клавіатуру тощо потрібно регулярно протирати дезінфікуючим засобом;

– регулярно мийте руки з милом протягом 30-40 секунд або обробляйте їх спиртовмісним засобом. Якщо на поверхні рук є вірус, миття рук або їх обробка спиртовмісним розчином вб’є його;

– в громадських місцях носіть маску або респіратор;

– по можливості не чіпайте руками очі, ніс, рот. Торкаючись руками очей, носа або рота, можна перенести вірус з поверхні рук до організму;

– дотримуйтесь правил респіраторної гігієни. При кашлі або чханні прикривайте рот і ніс серветкою або згином ліктя. Використану серветку відразу викидайте у контейнер для сміття;

– уникайте контактів з потенційно зараженими відходами або рідинам тваринного походження;

– дотримуйтесь звичайних правил гігієни в продуктових магазинах, де продаються м'ясо, риба, інші продукти тваринного походження;

– не вживайте в їжу сирі продукти тваринного походження (м'ясо, птицю, рибу, яйця) або ті продукти, які не пройшли належну термічну обробку;

– категорично уникайте будь – яких контактів з тваринами (бродячими котами, собаками, птахами, гризунами, кажанами);

– дотримуйтесь дистанції. Тримайтесь від людей на відстані мінімум 1,5 -2 метри, особливо, якщо у них кашель, нежить, підвищена температура. Кашляючи або чхаючи, людина, яка хворіє на респіраторну інфекцію, поширює навколо себе дрібні краплі, що містять вірус. Якщо ви перебуваєте занадто близько до такої людини, то можете заразитися при вдихання повітря;

– пройдіть лабораторне дослідження, якщо ви мали контакт з людиною хворою на COVID – 19.

Критерії за якими людина може вважатися контактною [41]**:**

– особа проживає в одному домі з хворим на COVID-19;

– особа мала прямий фізичний контакт з хворим на COVID-19 (наприклад, через рукостискання);

– особа мала контакт із слизовими виділеннями з дихальних шляхів хворого на COVID-19 та не використовувала засоби індивідуального захисту;

– особа контактувала із хворим на COVID-19 на відстані до одного метру протягом 15 хвилин і більше та не використовувала засоби індивідуального захисту;

– контакт в літаку в межах двох сидінь з хворим на COVID-19 (супутники подорожі).

Заходи безпеки в умовах карантину. Карантин – це адміністративні та медико-санітарні заходи, що застосовують для запобігання поширенню особливо небезпечної інфекційної хвороби (ст. 1 Закону України від 06.04.2000 №1645- III «Про захист населення від інфекційних хвороб») [42]. Карантин встановлюють на період, необхідний для ліквідації інфекційної хвороби. На цей період можуть змінюватися режими та умови роботи. Основними принципами профілактики інфекційних хвороб є дотримання громадянами санітарно – гігієнічних та санітарно – протиепідемічних правил і норм.

Основні поради щодо поведінки у період карантину на робочих місцях:

1) верхній одяг зберігайте в окремій закритій шафі, яка використовується тільки для верхнього одягу;

2) перед початком роботи ретельно помийте руки з милом а тоді обробляйте їх спиртовмісним розчином або антисептиком. На сьогоднішній день це обов’язкова процедура в разі будь – яких контактів із потенційно забрудненими об’єктами. Особливу увагу приділіть нігтям, там накопичується найбільше бруду;

3) у разі частого користування антисептиком, застосовуйте крем для рук, щоб уникнути підсушування та утворення тріщин шкіри через які можливе інфікування вірусом COVID – 19;

4) робоче місце, стіл клавіатуру, відкриті полиці, інші поверхні протріть разовою серветкою з дезінфікуючим засобом для обробки поверхонь на початку роботи та в кінці робочого дня;

5) провітрювати приміщення слід через кожні 2 – 3 години;

6) слід обмежити спілкування з колегами, робочі місця яких знаходяться в інших приміщеннях, виробничі питання вирішувати через засоби телефонного зв'язку;

7) поза межами свого кабінету при відвідуванні місць загального користування слід одягати маску;

8) зберігайте соціальну дистанцію 1- 1,5м. ;

9) правильно користуйтеся маскою. Знімати маску не доторкаючись до зовнішньої поверхні. Користуватися не більше 2 – 3 годин. Утилізувати маску в закриті ємності [43].

Для проведення дезінфекції користуйтесь засобами, дозволеними в Україні, що забезпечить ефективне знезараження щодо вірусної інфекції.

1) необхідно дотримуватися інструкції щодо кожного дезінфекційного засобу, в тому числі щодо отримання його ефективної концентрації та експозиції (дотримання часу перебування на оброблюваній поверхні);

2) антисептики для обробки шкіри рук - спиртовмісні препарати зі вмістом спирту вище 60%, діючі речовини – етиловий, ізопропіловий спирт чи їх комбінації, які мають пролонгований термін дії та певний час захищають шкіру рук від хвороботворних механізмів. Антисептики для обробки шкіри рук з мінімальним терміном експозиції (термін дії від 30 секунд до 2 хвилин) [44];

3) для обеззараження поверхонь меблів, обладнання необхідно обирати такі засоби, які забезпечували б якісну дезінфекцію при невисокій концентрації розчину, не мали неприємного запаху, не псували майно.

Цільовий інструктаж з безпеки життєдіяльності в умовах карантину.

1) при підвищеній температурі, появі кашлю, нежитю або ускладненні дихання негайно звертайтеся за медичною допомогою. Поява підвищеної температури, кашлю, ускладнення дихання можуть бути викликані респіраторною інфекцією або іншим захворюванням;

2) усі робочі поверхні (наприклад, столи, телефони, клавіатури, тощо) потрібно регулярно протирати дезінфікуючим засобом. Тому, що забруднення на поверхнях, які торкаються співробітники, є одним із основних джерел поширення інфекції;

3) частіше провітрюйте приміщення. Доступ чистого повітря на робочому місці перешкоджає розповсюдженню вірусів;

4) регулярно мийте руки з милом протягом 30- 40 секунд або обробляйте їх спиртовмісним засобом. Якщо на поверхні рук є вірус, миття рук або їх обробка спиртовмісним розчином вб’є його;

5) дотримуйтесь дистанції. Тримайтесь від людей на відстані мінімум 1 метра, особливо, якщо у них кашель, нежить, підвищена температура. Кашляючи або чхаючи, людина, яка хворіє на респіраторну інфекцію, поширює навколо себе дрібні краплі, що містять вірус. Якщо ви перебуваєте занадто близько до такої людини, то можете заразитися при вдихання повітря;

6) по можливості не чіпайте руками очі, ніс, рот. Торкаючись руками очей, носа або рота, можна перенести вірус з поверхні рук до організму;

7) дотримуйтесь правил респіраторної гігієни. При кашлі або чханні прикривайте рот і ніс серветкою або згином ліктя. Використану серветку відразу викидайте у контейнер для сміття. Це дозволяє запобігти поширенню вірусів. Якщо при

кашлі або чханні прикривати ніс або рот рукою, мікроби можуть потрапити на ваші руки, потім на предмети або людей, яких ви торкаєтесь;

8) уникайте контактів з потенційно зараженими відходами або рідинам тваринного походження;

9) дотримуйтесь звичайних правил гігієни в продуктових магазинах, де продаються м'ясо, риба, інші продукти тваринного походження;

10) не вживайте в їжу сирі продукти тваринного походження (м'ясо, птицю, рибу, яйця) або ті продукти, які не пройшли належну термічну обробку;

11) категорично уникайте будь – яких контактів з тваринами (бродячими котами, собаками, птахами, гризунами, кажанами);

12) в громадських місцях обов'язково носіть маску або респіратор;

13) перед тим, як одягати маску, вимийте руки з мильним розчином або обробіть їх спиртовмісним розчином;

14) одягніть маску так, щоб вона закривала рот і ніс без проміжків між обличчям і маскою;

15) не торкайтеся маски під час використання, в разі дотику, обробіть руки спиртовмісним розчином або вимийте їх з милом;

16) щойно маска матиме ознаки сирості, замініть її на нову, не використовуйте одноразової маски повторно;

17) безпечно знімайте маску, торкаючись її тільки ззаду (не торкайтеся передньої частини маски);

18) використану маску відразу викиньте в смітник.

5.2 Шкідливі фактори для організму при використанні ПК

Оцінку гігієнічних умов та характер праці на робочих місцях, які застосовуються на різних підприємствах, фірмах та інших форм власності надають Державні санітарні норми та правила «Гігієнічна класифікація праці за показниками шкідли-

вості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу», які затвердженими Наказом Міністерства охорони здоров'я України від 08.04.2014 №248 [45]. Але недотримання даних правил може значно знизити рівень імунної системи, внаслідок чого є високий ризик заразитися небезпечним вірусом COVID-19.

Відповідальність за виконання усіх цих правил покладається на посадових осіб, які займаються підприємницькою діяльністю та здійснюють застосування електронно-обчислювальних машин в промислових приміщеннях.

5.3 Нормативна база та вимоги безпеки під час роботи із екранними пристроями

Екранні пристрої — це електронні засоби, які відтворюють будь-яку графічну або алфавітно-цифрову інформацію (на основі електронно-променевої трубки, рідкокристалічні, плазмові, проекційні, органічні світлодіодні монітори й інші новітні розробки у сфері інформаційних технологій). Під час роботи з екранними пристроями на працівників впливають чинники важкості й напруженості трудового процесу, фізичні, хімічні та біологічні чинники.

Ризики від впливу основних видів небезпеки необхідно усунути або мінімізувати. Для цього вживають запобіжних заходів, щоб запобігти прогнозованим ризикам і забезпечити безпеку під час експлуатації екранних пристроїв. Мінімальні вимоги безпеки та захисту здоров'я під час роботи з екранними пристроями всіх типів і моделей установлюють Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями, затверджені наказом Мінсоцполітики від 14.02.2018 № 207 (далі — Вимоги) [46].

Вимоги поширюються на всіх суб'єктів господарювання незалежно від форм власності, організаційно-правової форми і видів діяльності. Вони встановлюють мінімальні вимоги безпеки та захисту здоров'я під час здійснення роботи, яка

пов’язана з використанням екранних пристроїв незалежно від їхнього типу та моделі. Водночас Вимоги не обмежують права роботодавця встановлювати більш жорсткі та/або спеціальні вимоги безпеки і захисту здоров’я та життя працівників під час роботи з екранними пристроями. Головне, щоб це не суперечило чинному законодавству.

Відповідно до вимог ст. 5 Закону України «Про охорону праці» працівників потрібно під розписку поінформувати про умови праці та наявність на їх робочих місцях небезпечних та шкідливих виробничих факторів (фізичних, хімічних, біологічних, психофізіологічних), які виникають під час роботи з екранними пристроями та ще не усунуто, а також про можливі наслідки їх впливу на здоров’я працівників [47]. Також роботодавець повинен забезпечити навчання і перевірку знань працівників з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними. Такий обов’язок передбачений і у випадках модифікації та організації роботи обладнання. Під час облаштування робочого місця працівника з екранними пристроями необхідно обирати таке устаткування, яке не створює зайвого шуму та не виділяє надлишкового тепла.

Роботодавець повинен за рахунок тривалості робочої зміни організувати внутрішні регламентовані перерви для відпочинку. Це передбачено Державними санітарними правилами і нормами роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 (затверджені постановою Головного державного санітарного лікаря України від 10.12.1998 № 7) [48].

Щоб не зашкодити здоров’ю і життю працівників, на підприємстві мають дотримуватися таких правил:

– не обслуговувати, ремонтувати, налагоджувати екранні пристрої безпосередньо на робочому місці працівника, коли він працює з ними;

– не вимикати захисні пристрої, не змінювати самочинно конструкцію та склад екранних пристроїв або їх технічне налагодження;

– не працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані й інші несправності.

5.4 Вимоги до приміщення під час використання екранних пристроїв

Серед основних вимог про правила охорони праці можна виділити наступні:

– розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено [49];

– площа на одне робоче місце становить не менше ніж 6,0 м, а об'єм - не менше ніж 20,0.м;

– приміщення для роботи з комп’ютерами повинні мати природне та штучне освітлення відповідно до СНиП II-4-79 ;

– природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід, і забезпечувати КПО не нижче, ніж 1,5%;

– покриття підлоги повинно бути матовим з коефіцієнтом відбиття 0.3-0.5;

– заборонено використовувати для оздоблення приміщень з ВДТ полімерні матеріали.

– у приміщеннях з електронно-обчислювальними машинами слід щоденно робити вологе прибирання;

Також на підприємствах, де є комп’ютери, мають бути приміщення для відпочинку від роботи, зазвичай її називають кімнатою психологічного розвантаження. В даній кімнати потрібно встановити пристрої для роздачі тонізуючих напоїв та наборами для занять фізичною культурою (від СНиП 2.09.04.-87) [50].

Перелік вимог для забезпечення безпечного користування екранними пристроями з наказу Міністерства соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [51]:

– робочі місця працівників з екранними пристроями повинні бути спроектовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення і рухів;

– всі випромінювання від екранних пристроїв повинне бути зведене до гранично допустимого рівня з точки зору безпеки і охорони здоров'я працівників;

– робочий стіл або робочу поверхню повинні бути достатнього розміру і мати поверхню з низькою відбивною здатністю, допускати гнучкість при розміщенні екрана, клавіатури, документів і відповідного обладнання;

– робоче крісло має бути стійким та дозволяти працівникові з екранними пристроями легко рухатися і займати зручне положення. Сидіння повинна регулюватися по висоті, спинка сидіння - як по висоті, так і по нахилу;

– слід передбачати підставку для тих, кому це необхідно для зручності;

– мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями повинен підтримуватися на постійному рівні і відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [52];

Крім тих вимог, які повинні дотримуватися усі робочі місця на підприємствах, є ще необхідні правила електронебезпеки. Далі буде наведено частину вимог з документу про правила охорони праці під час експлуатації електронно-обчислювальної машини,– Вимоги електронебезпеки [53]:

− під час монтажу необхідно унеможливити виникнення електричного джерела загоряння через коротке замикання та перевантаження проводів;

− обов’язково встановити аварійний резервний вимикач, якщо у приміщення працює понад п’ять комп’ютерів;

− електромережу штепсельних розеток живлення для комп’ютерів та іншого обладнання прокладати у каналах, при цьому не допускається використання деяких матеріалів, що швидко загоряються.

Норми клімату. У виробничих приміщеннях на усіх робочих місцях з персональними комп’ютерами мають обов’язково забезпечуватися оптимальні значення параметрів мікроклімату, а саме температури відносної вологості й рухливості повітря (СН 4088-86) (табл. 5.1) [54].

Таблиця 5.1 – Норми клімату для приміщень з екранними пристроями

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка - 1а	22-24	40-60	0,1
	Легка - 16	21-23	40-60	0,1
Тепла	Легка - 1а	23-25	40-60	0,1
	Легка - 16	22-24	40-60	0,2

Для вимірювання норми було використано гігрометр психрометричний — прилад для вимірювання температури та вологості повітря, простий тип гігрометра. Швидкість випаровування вологи збільшується в міру зменшення відносної вологості повітря. Випаровування вологи, в свою чергу, викликає охолодження конденсованої рідини. Таким чином, температура вологого об'єкта зменшується. За різницею температур повітря і вологого об'єкта можна визначити швидкість випаровування, а значить, і вологість повітря. При цьому треба враховувати той факт, що волога, яка випаровується, залишається в околицях вологого предмета, і, таким чи-

ном, локально збільшується вологість повітря. Для усунення цього ефекту, при вимірюванні вологості, застосовують аспірацію (створюється потік повітря над вологим об'єктом). В стаціонарних психрометрах термометри закріплені на спеціальному штативі в метеорологічній будці. Основний недолік станційних психрометрів — залежність показів зволоженого термометра від швидкості повітряного потоку в будці. Основний станційний психрометр — психрометр Августа. Зробивши наступні заміри у кімнаті, де проходила розробка системи для розпізнавання маски на обличчі було отримано наступні параметри:

1) Температура повітря – 21°C.
2) Вологість повітря – 44%.

Дані, які було порівняно із таблицею 5.1 повністю відповідають нормі для приміщення з екранними пристроями.

Рівні позитивних та негативних іонів в повітрі приміщень з ПК мають відповідати санітарно-гігієнічним нормам №2152-80 (табл. 5.2) [55].

Таблиця 5.2 – Допустимі норми рівня іонізації повітря приміщень

Рівні	Кількість іонів в 1 см повітря	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Штучне освітлення в приміщеннях з робочими місцями ВДТ ЕОМ та ПЕОМ має здійснюватися системою загального рівномірного освітлення, також допускаються додаткове встановлення світильників місцевого освітлення. Зазначення освітлення освітленості на поверхні робочого столу в зоні розміщення документів має становити в районі 300-500 лк [56].

Напруженість електромагнітного поля на відстані 0,5м від будь-якої поверхні відеомонітора не повинна перевищувати гранично допустимих рівнів (ГДР) які наведено у табл. 5.3.

Таблиця 5.3 – Гранично допустимі рівні при роботі з ПК

Діапазон частот	ГДР електричного поля, В/м	ГДР магнітного поля, нТл
5 Гц - 2 кГц	25	250
2 кГц - 400 кГц	2,5	25
3 МГц – 30 МГц	0,25	2,5

5.5 Вимоги роботи із комп’ютерами, рівень шуму

Вимоги щодо безпеки під час роботи з комп’ютерами (ВДТ ЕОМ, ПЕОМ) наведені у різних документах, зокрема у 4-й частині наказу Мінсоцполітики України. Згідно даних вимог, мінімальними вимогами є:

– робоче місце для працюючих з відео терміналами необхідно розташувати таким чином, щоб до поля зору працюючого не потрапляли вікна, освітлювальні прилади, поверхні які мають властивість віддзеркалювання. Поверхня робочого столу не повинна бути полірованою. Для попередження відблисків на екрані відео моніторів, особливо влітку та у сонячні дні, екран відео монітора слід розміщувати так, щоб світло від вікна падало збоку, бажано зліва;

– екран відео монітору ПК повинен знаходитись від очей користувача (надалі оператора) на відстані не менше 500 – 700 мм. Кут зору в межах 10-40 градусів. Найбільш раціональним є розташування екрану перпендикулярно до лінії зору оператора.;

– ПК повинен розташовуватися на відстані не ближче 1 метра від джерела тепла;

– клавіатура повинна розміщуватися на поверхні столу або спеціальній підставці на відстані 100-300 мм від краю, повернутого до користувача. Кут нахилу

панелі клавіатури до горизонтальної поверхні повинен бути в межах від 5 до 15 градусів;

– висота робочої поверхні стола повинна бути в межах 680-800 мм;

– крісло повинно забезпечувати операторові зручні умови праці та фізіологічну раціональну робочу позу в процесі праці. Крісло повинно забезпечувати можливість регулювання висоти поверхні сидіння, кут нахилу спинки та висоту спинки.

– для захисту від прямих сонячних променів, які створюють відблиски на екрані відео монітора на вікнах повинні бути встановлені сонцезахисні пристрої. Екран відео монітора повинен розміщуватись так, щоб світло від вікна падало на робоче місце збоку, бажано зліва.

– згідно ГОСТ 12 1 005-85 “ Повітря робочої зони. Загальні санітарно-гігієнічні вимоги ” температура навколишнього середовища повинна бути в межах 18-22 градусів С , відносна вологість повітря близько 55 % швидкість руху повітря – 0,1-0,2 м сек.;

– для захисту оператора від електромагнітних випромінювань і електростатичних полів, які створює відео монітор необхідно використовувати захисні екрани;

– у приміщеннях для роботи ПК необхідно проводити щоденне вологе прибирання та регулярне провітрювання протягом робочого дня. Видалення пилу з екрану необхідно проводити не рідше 1 разу на день.

Рівні шуму і вібрації. Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях мають відповідати вимогам СН 3223-85, ГОСТ 12.1.003-83, ГР 2411-81 (табл. 5.4) [57].

Таблиця 5.4 - Допустимі рівні звуку

Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних частотних смугах									
Вид трудової діяльності, робочі місця	Рівні звукового тиску в дБ								
	в октавних смугах із середньогеометричними частотами, Гц								
	31,5	**63**	**125**	**250**	**500**	**1000**	**2000**	**4000**	**8000**
Програмісти	86	71	61	54	49	45	42	40	38
Оператори в залах обробки інформації	96	83	74	68	63	60	57	55	54
В приміщеннях для розташування шумних агрегатів	103	91	83	77	73	70	68	66	64

Шумомір – прилад для об'єктивного вимірювання рівня звуку. Шумомір містить ненаправлений мікрофон, підсилювач, коригувальні фільтри, детектор, інтегратор (для інтегруючих шумомірів) та індикатор.

Фактично шумомір є мікрофоном, до якого підключений вольтметр, відградуйований в децибелах. Оскільки електричний сигнал на виході з мікрофона пропорційний вихідному звуковому сигналу, приріст рівня звукового тиску, що впливає на мембрану мікрофона, викликає відповідний приріст напруги електричного струму на вході у вольтметр, що відображається за допомогою індикаторного пристрою, що відгороджується в децибелах. Для вимірювання рівнів звукового тиску

контрольованих смугах частот, наприклад, 31,5; 63; 125 Гц і т. п., а також для вимірювання рівнів звуку (ДБА), коригованих за шкалою А з урахуванням особливостей сприйняття людським вухом звуків різних частот, сигнал після виходу з мікрофона, але до входу у вольтметр пропускають через відповідні електричні фільтри. Загальна схема шумоміра вибирається те щоб його властивості наближалися до властивостей людського вуха.

Метою акустичного розрахунку є визначення очікуваних рівнів звукового тиску (L, дБ) в октавних смугах частот. Акустичний розрахунок шуму виконують відповідно до СП52.13330.2011 «Захист від шуму».

Акустичний розрахунок повинен проводитись у наступній послідовності:

– виявлення джерел шуму та визначення їх шумових
– характеристик;
– вибір точок у приміщеннях та на територіях, для яких
– необхідно провести розрахунок (розрахункових точок);
– визначення шляхів поширення шуму з його джерела
– (джерел) до розрахункових точок та втрат звукової енергії по 7 кожному з шляхів (зниження за рахунок відстані, екранування, звукоізоляції огороджувальних конструкцій, звукопоглинання та ін.);
– визначення очікуваних рівнів шуму у розрахункових точках;
– визначення необхідного зниження рівнів шуму на основі
– зіставлення очікуваних рівнів шуму з допустимими рівнями
– шуму;
– перевірочний розрахунок достатності обраних заходів для
– забезпечення захисту об'єкта чи території від шуму.

Акустичний розрахунок слід проводити за рівнями звукового тиску Lp, дБ, в октавних смугах із середньогеометричними частотами 31,5, 63, 125, 250, 500, 1000,

2000, 4000 та 8000 Гц. Розрахункові точки у виробничих приміщеннях промислових підприємств вибирають на робочих місцях та у зонах постійного перебування людей на висоті 1,5 м від підлоги.

Вихідними даними для акустичного розрахунку є:

– план та розріз приміщення з розташуванням технологічного та

– інженерного обладнання та розрахункових точок;

– відомості про характеристики огороджувальних конструкцій приміщення (матеріал, товщина, щільність та ін.);

– шумові характеристики та геометричні розміри джерел шуму.

Розрахункові формули представлені для пропорційних приміщень (з відношенням найбільшого геометричного розміру до найменшого не більше 5) і для джерел шуму, найближчих до розрахункової точки (що знаходяться на відстані ri≤ 5 rмін, де rмін - відстань від розрахункової точки до акустичного центру найближчого джерела шуму). Октавні рівні звукового тиску L, дБ, у розрахункових точках при роботі кількох джерел шуму слід визначати за такою формулою:

$$L = 10 * \lg\left(\sum_{i=1}^{n} \frac{10^{0.1*L_{wi}*\chi_i*\Phi_i}}{\Omega*r_i^2} + \frac{4}{k*B} * \sum_{i=1}^{n} 10^{0.1*L_{wi}}\right) \quad (5.1)$$

де n – загальна кількість джерел шуму у приміщенні;

L_{wi} – октавний рівень звукової потужності і-го джерела, дБ;

χ – коефіцієнт, що враховує вплив ближнього поля. Визначається за табл. 6 залежно від співвідношення $r/l_{макс}$ ($l_{макс}$ – довжина обладнання);

Ф – фактор спрямованості джерела шуму (для джерел із рівномірним випромінюванням Ф = 1);

Ω – просторовий кут випромінювання джерела, рад. (Приймають за табл.2);

r_i – відстань від акустичного центру і-го джерела шуму до розрахункової точки м;

k – коефіцієнт, що враховує порушення дифузності звукового поля у приміщенні (приймають за табл. 3 залежно від коефіцієнта звукопоглинання?);

В – акустична постійна приміщення, м2, що визначається за формулою

$$\boldsymbol{B} = \frac{\boldsymbol{A}}{\mathbf{1} - \alpha_{ср}}, \tag{5.2}$$

де $\alpha_{ср}$ – середній коефіцієнт звукопоглинання,

A – еквівалентна площа звукопоглинання, м^2.

$$\boldsymbol{A} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{a_i} * \boldsymbol{S_i}, \tag{5.3}$$

де $\boldsymbol{a_i}$ – коефіцієнт звукопоглинання і-ї поверхні,

$\boldsymbol{S_i}$ – площа i-ої поверхні, м^2.

Таблиця 5.5 – Вихідні дані

№	Джерело шуму	Розмір приміщення, м			Відстань від приміщення до робочого місця,м		
		A	B	H	r_1	r_2	r_3
1	Принтер	12	6	5	4	6	8
2	Комп'ютер						
3	Каво-машина						

Коефіцієнти звукопоглинання:

α1- коефіцієнт звукопоглинання стелі та стін;

α2 - коефіцієнт звукопоглинання статі.

Розрахунки.

χ - коефіцієнт, що враховує вплив ближнього поля;

Принтер – r1/l1 = 4 / 1,2 = 3,33χ = 1,

Комп'ютер – r2/l2 = 6 / 0,6 = 10χ = 1,

Каво-машина – r3/l3 =8 / 1,5 = 5,33χ = 1,

Φ = 1- фактор спрямованості джерела шуму,

Принтер $\Omega 1 = \pi / 2$;

Комп'ютер $\Omega 2 = \pi$;

Каво-машина $\Omega 3 = 2\pi$;

Відстань від акустичного центру джерела шуму до розрахункової точки:

k =1,25 коефіцієнт, що враховує порушення дифузності звукового поля у приміщенні (табл. 3 для α<0,2);

Площа звукопоглинання:

- площа стелі S = 12*6 = 72 м^2;
- площа стін S = 5*(2*12 + 2*6) = 80 м^2;
- площа підлоги S = 12*6 = 72 м^2.

Сумарна площа огороджувальних конструкцій:

- $S_{огр.}$ = 72 + 80 + 72 = 224 м^2;

Середньогеометрична частота 63 Гц. Еквівалентна площа звукопоглинання:

$$\boldsymbol{A = \sum_{i=1}^{n} a_i * S_i = 0.08 * 72 + 0.08 * 80 + 0.15 * 72 = 23.0} \text{ м}^2 \qquad (5.4)$$

Середній коефіцієнт звукопоглинання:

$$\boldsymbol{\alpha_{ср} = \frac{A}{S_{огр}} = \frac{22.96}{224} = 0.103}, \qquad (5.5)$$

Акустична постійна приміщення:

$$B = \frac{A}{1 - \alpha_{ср}} = \frac{22.96}{1 - 0.103} = 25.6, \quad (5.6)$$

$$L = 10 * \lg\left(\sum_{i=1}^{3}\left(\frac{10^{0.1*84} * 1 * 1}{\frac{\pi}{2} * 16} + \frac{10^{0.1*81} * 1 * 1}{\pi * 36} + \frac{10^{0.1*88} * 1 * 1}{2\pi * 64}\right)\right. \quad (5.7)$$
$$\left. + \frac{4}{1.25 * 25.6} * \sum_{i=1}^{3}(10^{0.1*84} + 10^{0.1*81} + 10^{0.1*88})\right) = 78.5 \text{ дБ}$$

Виходячи із розрахунків можна зробити висновок, що рівні звукового тиску в розрахунковій точці відповідають допустимим значення в октавних смугах частот із середньогеометричними частотами від 250 Гц до 8000 Гц згідно таблиці 5.4.

5.6 Відпочинок на підприємствах під час праці з комп'ютерними пристроями

При організації праці, що пов'язана з використанням персональних комп'ютерів, для збереження здоров'я працюючих, запобігання професійним захворювання і підтримки працездатності слід передбачити внутрішньозмінні регламентовані перерви для відпочинку. Внутрішньозмінні режими праці і відпочинку мають передбачати додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення і зниження працездатності. За основну роботу з персональним комп'ютером слід вважати таку, що займає не менше 50% часу впродовж робочої зміни. Протягом дня мають передбачатися:

- перерви для відпочинку і вживання їжі (обідні перерви);
- перерви для відпочинку і особистих потреб (згідно з трудовими нормами);
- додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Тривалість обідньої перерви визначається чинним законодавством про працю і Правилами внутрішнього трудового розпорядку. Встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за персональним комп’ютером;

– для операторів персональних комп’ютерів слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп’ютерного набора слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за персональним комп’ютером.

Активний відпочинок має полягати у виконанні комплексу гімнастичних вправ, спрямованих на зняття нервового напруження, м’язове розслаблення, відновлення функцій фізіологічних систем, що порушуються протягом трудового процесу, зняття втоми очей, поліпшення мозкового кровообігу і працездатності. За умови високого рівня напруженості робіт з персональним комп’ютером показане психологічне розвантаження у спеціально обладнаних приміщеннях (в кімнатах психологічного розвантаження) під час регламентованих перерв або в кінці робочого дня.

Для зниження нервово-емоційного напруження, втомлення зорового аналізатору, поліпшення кровообігу тощо, рекомендується проведення та виконання комплексів вправ (таблиця 5.5 на наступних сторінках). Також, в окремих випадках – при хронічних скаргах працівників на втомлення та погане самопочуття – допускається індивідуальний підхід до обмеження часу робіт з ВДТ, зміна характеру праці, чергування з іншими видами діяльності тощо.

Окремі вимоги та показання також визначаються для жінок, так працівники жіночої статі, які працюють з комп'ютерним приладдям, обов'язково оглядаються лікарем акушером-гінекологом один раз на два роки. Також жінки можуть бути звільнені від обов'язків роботи з ВДТ ЕОМ під час вагітності та в період годування дитини грудьми.

Висновки до розділу 5

Під час виконання спеціальної частини з охорони праці було проведено аналіз проблеми епідемії вірусу COVID-19, аналіз діючих нормативних документів, законів, а також досліджено норми та рекомендації ВООЗ при епідемії вірусу, досліджено умови праці людини за комп'ютером на підприємствах, вплив негативних факторів на здоров'я працівників і як це може оказати вплив та повисити ризик захворіти небезпечним вірусом.

Правила, які вказані у нормативних документах є актуальними на сьогодні через те, що вірус COVID-19 передається повітряно-крапельним шляхом, тому дуже важливо дотримуватися рекомендацій. Під час роботи за персональним комп'ютером людина швидко втомлюється та від цього страждає опорно-руховий, зоровий апарат, нервова система.

У результаті розрахунків визначено, що призначене приміщення відповідає допустимим значення в октавних смугах частот. На сьогодні існує багато шкідливих умов, які погіршують процес трудової діяльності людини, такі як: шум, вібрація та інші умови, що можуть заважати розробнику під час роботи.

Вимоги та нормативи, щодо охорони праці мають дотримуватися на підприємствах під час трудової діяльності людини. Дотримання поставлених вимог до працівників та власників підприємств дозволить мінімізувати шкідливі наслідки, які мають вплив на здоров'я людини та допоможуть вберегти своє здоров'я від небезпечних хвороб.

ВИСНОВКИ

В наш час людство справляється з багатьма труднощами, використовуючи нові сучасні технології, за допомогою яких створюються інтелектуальні та практичні системи. При боротьбі з COVID-19 створюються унікальні системи для боротьби з інфекцією та запобігання її поширенню в громадських місцях. Існує безліч систем з різним функціоналом, крім головної - розпізнавання маски, серед них можна виділити одні з популярних: MaskCheck, Xeoma, SafeSpace. Всі ці програми мають ряд переваг, але так само недоліки через відсутність необхідних функцій в одній системі.

Розробка системи розпізнавання маски на обличчі людини є актуальною, оскільки без контролю громадських місць (метро, школи, держустанови, магазини, офіси тощо) є високий ризик заразитися небезпечним вірусом COVID-19. Носіння маски в обіцяних місцях значно зменшить темп захворюваності та допоможе врятувати багато життів.

Метою проекту була автоматизація розпізнавання наявності маски на обличчі людини на мові програмування Python з використанням бібліотек TensorFlow, Keras та OpenCV. У процесі створення системи було виконано такі завдання:

1) досліджено проблему вірусу COVID-19, методи боротьби та застереження;
2) проаналізовано аналоги систем та програмних продуктів;
3) розглянуто архітектури нейронних мереж на вирішення завдання розпізнавання;
4) вивчено принципи роботи бібліотеки TensorFlow, Keras та OpenCV, а саме:
   - інтеграція Keras у методи TensorFlow як надбудова;
   - навчання нейронної мережі, використовуючи оптимізатори;
   - використання архітектури згорткової нейронної мережі;
   - попередня обробка зображення;
   - обробка потокового зображення з вебкамери;

- використання карти ознак та фільтруючого ядра в процесі пошуку ознак на вхідних фото.

5) розглянуто прототипи функції перерахованих вище бібліотек, що реалізують даний функціонал.

Спроектовано ГІК, що дозволяє навчити модель нейронної мережі, запустити камеру з нейронною мережею для розпізнавання маски, а також підключити додатковий функціонал. ДІК організований так, щоб діалог із користувачем був простішим.

Розроблена система має можливість розпізнавати маски на обличчях людей, вести статистику до бази даних та видавати цю інформацію у вигляді кругової діаграми, сповіщати про людину без маски в месенджері Telegram та відповідним звуковим сигналом. Завдання розробити систему необхідно та актуальне через небезпечну пандемію.

Виходячи з поставлених завдань та підібраних технологій дана система займатиме небагато місця на ПК або мікроконтролерах, мати зрозумілий інтерфейс і великий функціонал, а так само дана система поширюватиметься з відкритим вихідним кодом безкоштовно.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1) Timeline of the COVID-19 in pandemic 2019 year. URL: https://en.wikipedia.org/wiki/Timeline_of_the_COVID-19_pandemic_in_2019 (дата звернення: 01.05.2022).
2) Leveraging Data Science to Combat COVID-19: A Comprehensive Review. URL: https://ieeexplore.ieee.org/document/9184922 (дата звернення: 01.05.2022).
3) Coronavirus worldwide. URL: https://news.google.com/covid19/map?hl=en-PK&gl=PK&ceid=PK%3Aen (дата звернення: 03.05.2022).
4) Worldometer COVID-19. URL: https://www.worldometers.info/coronavirus/ (дата звернення: 03.05.2022).
5) Посібник з профілактики та боротьби з COVID-19. URL: https://www.dec.gov.ua/materials/ecdc-posibnyk-z-profilaktyky-ta-borotby-z-covid-19-u-czentrah-tymchasovogo-rozmishhennya-bizhencziv-2/(дата звернення: 03.05.2022).
6) Quantification of trace elements in surgical and KN95 face masks widely used during the SARS-COVID-19 pandemic URL: https://www.sciencedirect.com/science/article/pii/S0048969721070005?via%3Dihub (дата звернення: 04.05.2022).
7) Rational use of face masks in the COVID-19 pandemic. URL: https://www.thelancet.com/journals/lanres/article/PIIS2213-2600(20)30134-X/fulltext (дата звернення: 04.05.2022).
8) Coronavirus (COVID-19). URL: https://www.nhs.uk/conditions/coronavirus-covid-19/ (дата звернення: 08.05.2022).
9) Use Face ID while wearing a mask with iPhone 12 and later. URL: https://support.apple.com/en-us/HT213062 (дата звернення: 08.05.2022).
10) MaskCheck Kiosk, powered by SAFR. URL: https://getmaskcheck.com/kiosk/ (дата звернення: 08.05.2022).

11) Safe-space app. URL: https://safer.work/safe-space/ (дата звернення: 09.05.2022).
12) Felenasoft cameras. URL: https://felenasoft.com/ (дата звернення: 09.05.2022).
13) Python. URL: https://ru.wikipedia.org/wiki/Python (дата звернення: 09.05.2022).
14) JetBrains products. URL: https://www.jetbrains.com/ (дата звернення: 09.05.2022).
15) Нейронні мережі для починаючих. Частина 1. URL: https://habr.com/post/312450/ (дата звернення: 10.05.2022).
16) Статистичні методи для аналізу даних. URL: https://www.vsavm.by/knigi/kniga3/780.html (дата звернення: 10.05.2022).
17) Simon S. Haykin Neural Networks: A Comprehensive Foundatio. Ontario: Prentice Hall, Subsequent edition 1998. 586 с.
18) Ian Goodfellow Deep Learning (Adaptive Computation and Machine Learning series). MIT Press, 2016. 245 с.
19) Multilayer perceptron. URL: https://www.ibm.com/docs/ru/spss-statistics/SaaS?topic=networks-multilayer-perceptron (дата звернення: 10.05.2022).
20) Charu C. Aggarwal Neural Networks and Deep Learning: A Textbook. Springer, 2019. 467 с.
21) About TensorFlow. URL: https://www.tensorflow.org/about (дата звернення: 11.05.2022).
22) Tensorflow Wikipedia. URL: https://en.wikipedia.org/wiki/TensorFlow (дата звернення: 11.05.2022).
23) Why this name Keras. URL: https://keras.io/#why-this-name-keras (дата звернення: 15.05.2022).
24) Keras documentation. URL: https://faroit.com/keras-docs/1.2.0/ (дата звернення: 16.05.2022).

25) Tesla AI. URL: https://www.tesla.com/AI (дата звернення: 16.05.2022).

26) Viral clip tesla car autopilot. URL: https://www.indiatimes.com/technology/news/viral-clip-tesla-car-autopilot-553430.html (дата звернення: 16.05.2022).

27) A computer vision-based object detection and counting for COVID-19 protocol compliance: a case study of Jakarta. URL: https://ieeexplore.ieee.org/document/9307594 (дата звернення: 17.05.2022).

28) Leon O. Chua Cellular Neural Networks and Visual Computing: Foundations and Applications. Cambridge University Press, 2002. 254 с.

29) ImageNet Large Scale Visual Recognition Challenge (ILSVRC). URL: https://image-net.org/challenges/LSVRC/ (дата звернення: 17.05.2022).

30) Deep Residual Learning for Image Recognition. URL: https://arxiv.org/abs/1512.03385 (дата звернення: 25.05.2022).

31) Robotic control with graph networks. URL: https://towardsdatascience.com/robotic-control-with-graph-networks-f1b8d22b8c86 (дата звернення: 31.05.2022).

32) MobileNet. URL: https://keras.io/api/applications/mobilenet/ (дата звернення: 01.06.2022).

33) MobileNetV2. URL: https://machinethink.net/blog/mobilenet-v2/ (дата звернення: 01.06.2022).

34) Common objects in context. URL: https://cocodataset.org/#home (дата звернення: 02.06.2022).

35) Venv library Python. URL: https://docs.python.org/3/library/venv.html (дата звернення: 03.06.2022).

36) Mask detection. URL: https://www.kaggle.com/datasets/andrewmvd/face-mask-detection (дата звернення: 03.06.2022).

37) F1-score in Python. URL: https://pythobyte.com/f1-score-in-python-932523cd/ (дата звернення: 03.06.2022).

38) George S. Moschytz Cellular Neural Networks: Analysis, Design and Optimization. Springer, 2013. 74 с.

39) COVID-19: витоки вірусу (1 частина). URL: https://habr.com/post/646795/ (дата звернення: 04.06.2022).

40) Рекомендації для населення щодо інфекції, викликаної новим коронавірусом (COVID-19). URL: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public (дата звернення: 04.06.2022).

41) Інструкція з БЖД на час карантинних періодів. URL: http://izyaslav.osv.org.ua/news/09-25-17-13-03-2020/ (дата звернення: 05.06.2022).

42) Про захист населення від інфекційних хвороб. URL: https://zakon.rada.gov.ua/laws/show/1645-14#Text (дата звернення: 05.06.2022).

43) Заходи безпеки при роботі в умовах карантину. URL: http://www.fssu.gov.ua/fse/control/kievobl/uk/publish/printable_article/113437;jsessionid=9C2B6551BA46603B1E0475F10FF46C97 (дата звернення: 06.06.2022).

44) Чому не варто зловживати антисептиками. URL: https://health.nv.ua/ukr/medicine/antiseptik-i-koronavirus-50142516.html (дата звернення: 07.06.2022).

45) МІНІСТЕРСТВО ОХОРОНИ ЗДОРОВ'Я УКРАЇНИ НАКАЗ № 248. URL: https://opcb.kpi.ua/wp-content/uploads/2011/09/gigienichna_klasufikacija_praci.pdf (дата звернення: 07.06.2022).

46) Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. – К.: Постанова Головного державного санітарного лікаря України, 1998.- № 7.

47) Правові аспекти охорони праці. URL: https://minjust.gov.ua/m/str_3107 (дата звернення: 09.06.2022).

48) Державні санітарні правила і норми роботи з електронно-обчислюваними машинами. (затверджені постановою Головного державного санітарного лікаря України від 10.12.1998 № 7) URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 10.06.2022).

49) Вимоги до приміщень, розміщення в них ВДТ, ЕОМ, ПЕОМ. URL: https://lektsii.com/2-84145.html (дата звернення: 10.06.2022).

50) СНиП 2.09.04-87. URL: http://online.budstandart.com/catalog/doc-page?id_doc=4170 (дата звернення: 10.06.2022).

51) Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. URL: https://zakon.rada.gov.ua/rada/show/va042282-99#Text (дата звернення: 10.06.2022).

52) Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин. URL: https://zakon.rada.gov.ua/laws/show/z0293-10#Text (дата звернення: 11.06.2022).

53) Санітарні норми мікроклімату виробничих приміщень. URL: http://consultant.parus.ua/?doc=03N7Q8B192 (дата звернення: 12.06.2022).

54) ГН 2152-80. Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень. URL: https://dnaop.com/html/2296/doc-%D0%93%D0%9D_2152-80 (дата звернення: 12.06.2022).

55) Які норми освітленості робочих місць і виробничих приміщень? URL: http://zprim.com.ua/yaki-normi-osvitlenosti-robochih-mists-i-virobnichih-primishhen/ (дата звернення: 13.06.2022).

56) Санітарні норми виробничого шуму, ультразвуку та інфразвуку. Державні санітарні норми. ДСН 3.3.6.037-99. – К., 1999.

57) Розрахунок норми шуму. URL: https://www.legalaid.gov.ua/publikatsiyi/dopustymyj-riven-shumu/ (дата звернення: 14.06.2022).

ДОДАТОК А

Код для навчання нейронної мережі на Python

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
from sys import argv
import numpy as np
import os

script, img_path, epochs, num_photo_per_epoch, name_model = argv


def train_model(image_path: str, epoch: str, photo_per_epoch: str, model_name: str):
    # initialize the initial learning rate, number of epochs to train for,
    # and batch size
    EPOCHS: int = int(epoch)
    PHOTO_PER_EPOCH: int = int(photo_per_epoch)
    INIT_LR: float = 1e-4
    print("[INFO] loading images...")

    data: list = []
    labels: list = []

    for directory in os.listdir(image_path):
        label = os.path.join(image_path, directory)
        if not os.path.isdir(label):
```

```
        continue

    for item in os.listdir(label):
        if item.startswith("."):
            continue
        image = load_img(os.path.join(label, item), target_size=(224, 224))
        image = img_to_array(image)
        image = preprocess_input(image)

        data.append(image)
        labels.append(label)

data = np.array(data, dtype="float32")
labels = np.array(labels)

lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels,
                                    test_size=0.20, stratify=labels, random_state=42)

aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

baseModel = MobileNetV2(weights="imagenet", include_top=False,
                    input_tensor=Input(shape=(224, 224, 3)))

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
```

```
    model = Model(inputs=baseModel.input, outputs=headModel)

    for layer in baseModel.layers:
        layer.trainable = False

    print("[INFO] compiling model...")
    opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
    model.compile(loss="binary_crossentropy", optimizer=opt,
                  metrics=["accuracy"])

    print("[INFO] training head...")
    H = model.fit(
        aug.flow(trainX, trainY, batch_size=PHOTO_PER_EPOCH),
        steps_per_epoch=len(trainX) // PHOTO_PER_EPOCH,
        validation_data=(testX, testY),
        validation_steps=len(testX) // PHOTO_PER_EPOCH,
        epochs=EPOCHS)

    print("[INFO] evaluating network...")
    predIdxs = model.predict(testX, batch_size=PHOTO_PER_EPOCH)

    predIdxs = np.argmax(predIdxs, axis=1)

    print(classification_report(testY.argmax(axis=1), predIdxs,
                                target_names=lb.classes_))

    print("[INFO] saving mask detector model...")
    model.save("models/"+model_name + ".h5")


if __name__ == "__main__":
    train_model(img_path, epochs, num_photo_per_epoch, name_model)
```

ДОДАТОК Б

Код графічного користувацького інтерфейсу

```
import os
import json
import tkinter as tk
from tkinter import filedialog
from training import train_model
import subprocess

def write_to_model_config():
    path = input_dataset_path.get()
    epoch = input_epochs.get()
    photo_per_ep = input_photo_per_epoch.get()
    model = input_name_model.get()
    with open("./configurations/config.json", "r") as file_config:
        json_config = json.load(file_config)
    json_config["train_model"][0]["name"] = photo_per_ep
    json_config["train_model"][0]["path_to_photo"] = path
    json_config["train_model"][0]["epochs"] = epoch
    json_config["train_model"][0]["photos_per_epoch"] = model
    with open("./configurations/config.json", 'w') as ready_file:
        json.dump(json_config, ready_file)

def load_from_model_config():
    global input_name_model, input_dataset_path, input_epochs, input_photo_per_epoch

    with open("./configurations/config.json", 'r') as file_config:
        json_config = json.load(file_config)

    list_of_entries = [input_name_model, input_dataset_path, input_epochs, in-
put_photo_per_epoch]
    for entry in list_of_entries:
        entry.delete(0, 'end')

    config_name = json_config["train_model"][0]["name"]
    config_path_to_photo = json_config["train_model"][0]["path_to_photo"]
    config_epochs = json_config["train_model"][0]["epochs"]
    config_photo_per_epoch = json_config["train_model"][0]["photos_per_epoch"]

    input_name_model.insert(0, config_name)
```

```
    input_dataset_path.insert(0, config_path_to_photo)
    input_epochs.insert(0, config_epochs)
    input_photo_per_epoch.insert(0, config_photo_per_epoch)

def load_from_telegram_config():
    global input_user_id, input_frames, input_learned_model

    with open("./configurations/config.json", 'r') as file_config:
        json_config = json.load(file_config)

    list_of_entries = [input_user_id, input_frames, input_learned_model]
    for entry in list_of_entries:
        entry.delete(0, 'end')

    config_user_id = json_config["telegram"][0]["telegram_id"]
    config_frames = json_config["telegram"][0]["frames"]
    config_learned_model = json_config["telegram"][0]["learned_model"]

    input_user_id.insert(0, config_user_id)
    input_frames.insert(0, config_frames)
    input_learned_model.insert(0, config_learned_model

def browse_path_model():
    global input_dataset_path
    filename = filedialog.askdirectory()
    input_dataset_path.insert(0, filename)

def browse_path_telegram():
    global input_learned_model
    filename = filedialog.askopenfilename()
    model_file = filename.split('/')[-1]
    input_learned_model.delete(0, 'end')
    input_learned_model.insert(0, model_file)

def show_frame(frame):
    frame.tkraise()

def button_write_to_telegram_config():
    usr_id = input_user_id.get()
    frame = input_frames.get()
    model = input_learned_model.get()
```

```
    with open("./configurations/config.json", "r") as file_config:
        json_config = json.load(file_config)
    json_config["telegram"][0]["telegram_id"] = usr_id
    json_config["telegram"][0]["frames"] = frame
    json_config["telegram"][0]["learned_model"] = model
    with open("./configurations/config.json", 'w') as ready_file:
        json.dump(json_config, ready_file)

def start_training():
    path = input_dataset_path.get()
    epoch = input_epochs.get()
    photo_per_ep = input_photo_per_epoch.get()
    model = input_name_model.get()
    print(os.system(f"venv\Scripts\python.exe training_args.py {path} {epoch}
{photo_per_ep} {model}"))

def start_mask_detector():
    model = input_learned_model.get()
    os.system(f"venv\Scripts\python.exe detect_mask.py {model}")

window = tk.Tk()

# Options of main window
window.geometry("425x160")
window.title("Mask Detector config")

# Frames
telegram_window = tk.Frame(window)
train_model_window = tk.Frame(window)

for windows in (telegram_window, train_model_window):
    windows.grid(row=0, column=0, sticky='nsew')

# Telegram_windows
user_id = tk.Label(telegram_window, text="User id")
user_id.grid(column=0, row=1)

frames = tk.Label(telegram_window, text="Frames")
frames.grid(column=0, row=2)

learned_model = tk.Label(telegram_window, text="Learned model")
```

```
learned_model.grid(column=0, row=3)

input_user_id = tk.Entry(telegram_window, width=35)
input_user_id.grid(column=1, row=1)

input_frames = tk.Entry(telegram_window, width=35)
input_frames.grid(column=1, row=2)

input_learned_model = tk.Entry(telegram_window, width=35)
input_learned_model.grid(column=1, row=3)

button_write_to_telegram_config = tk.Button(telegram_window, text="Write to con-
fig",
                                            command=button_write_to_telegram_config)
button_write_to_telegram_config.grid(column=1, row=4, sticky="ew")

button_choose_model = tk.Button(telegram_window, text="Select model", com-
mand=browse_path_telegram)
button_choose_model.grid(column=2, row=3)

button_mask_detector = tk.Button(telegram_window, text="Start camera", com-
mand=start_mask_detector)
button_mask_detector.grid(column=1, row=5, sticky="ew")

button_go_to_train_model = tk.Button(telegram_window, text='Go to model training',
                                     command=lambda: show_frame(train_model_window))
button_go_to_train_model.grid(column=1, row=6, sticky="ew")

button_load_model = tk.Button(telegram_window, text="Load config", com-
mand=load_from_telegram_config)
button_load_model.grid(column=2, row=4, sticky="ew")

# Train model window
name_model = tk.Label(train_model_window, text="Model name")
name_model.grid(column=0, row=1)

dataset = tk.Label(train_model_window, text="Dataset directory")
dataset.grid(column=0, row=2)

epochs = tk.Label(train_model_window, text="Epochs")
epochs.grid(column=0, row=3)
```

```
photo_per_epoch = tk.Label(train_model_window, text="Photo per epoch")
photo_per_epoch.grid(column=0, row=4)

input_name_model = tk.Entry(train_model_window, width=35)
input_name_model.grid(column=1, row=1)

input_dataset_path = tk.Entry(train_model_window, width=35)
input_dataset_path.grid(column=1, row=2)

input_epochs = tk.Entry(train_model_window, width=35)
input_epochs.grid(column=1, row=3)

input_photo_per_epoch = tk.Entry(train_model_window, width=35)
input_photo_per_epoch.grid(column=1, row=4)

button_train_model = tk.Button(train_model_window, text="Train model", com-
mand=start_training)
button_train_model.grid(column=1, row=5, sticky="ew")

button_go_to_telegram_config = tk.Button(train_model_window, text="Go to telegram
configuration",
                                         command=lambda: show_frame(telegram_window))
button_go_to_telegram_config.grid(column=1, row=7, sticky="ew")

button_select_path = tk.Button(train_model_window, text="Select path", com-
mand=browse_path_model)
button_select_path.grid(column=2, row=2, sticky="ew")

button_write_training_model_spec = tk.Button(train_model_window, text="Save to
config", command=write_to_model_config)
button_write_training_model_spec.grid(column=2, row=3, sticky="ew")

button_load_training_model_spec = tk.Button(train_model_window, text="Load from
config", command=load_from_model_config)
button_load_training_model_spec.grid(column=2, row=4, sticky="ew")

show_frame(telegram_window)
window.mainloop()
```