Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки
(повна назва інституту)

Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)
магістра
(рівень вищої освіти)
на тему
Маркування розпізнавання обличчя в умовах ковідної пандемії

Виконав: студент 2 курсу, групи 601-ТН
спеціальності
122 Комп'ютерні науки
(шифр і назва спеціальності)
Овчаренко В.О.
(прізвище та ініціали)
Керівник Ляхов О. Л.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Полтава – 2021 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

спеціальність 122 «Комп'ютерні науки»

на тему

«Маркування розпізнавання обличчя в умовах ковідної пандемії»

Студента групи 601-ТН Овчаренко Владислава Олександровича

Керівник роботи
доктор технічних наук,
професор Ляхов О. Л.

Завідуючого кафедри
кандидат технічних наук,
доцент Головко Г. В.

Полтава – 2021

РЕФЕРАТ

Кваліфікаційна робота магістра: 85 с., 22 малюнків, 2 додатки, 23 джерел.

Об’єкт дослідження: Люди на фото к масками та без.

Мета роботи:створити нейронну мережу для пошуку людей без масок та в масках на фото

Методи:створення нейронної мережі з допомогою TenserFlow, OpenCV, Python. Навчання моделі на датасеті із фотографій

Ключові слова:facemaskdetection, роспізнавання облич, пошук масок, Машинне навчання

ABSTRACT

Qualifying work of the master: 85 pages, 22 drawings, 2 appendix, 23 sources.

Object of research: People in the photo with masks and without.

Purpose: to create a neural network to search for people without masks and in masks in the photo.

Methods: creating a neural network using TenserFlow, OpenCV, Python. Model training on a photo dataset.

Keywords: network: face mask detection, face recognition, mask search, machine learning.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

TF – TenserFlow

МН – Машинне навчання

ML–Машинне навчання

Py–Pyton

Dataset – набір данних

ВСТУП

З тих пір, як інфекційна коронавірусна хвороба (COVID-19) була вперше зареєстрована в Ухані, вона стала проблемою громадського здоров'я в Китаї та навіть у всьому світі. Ця пандемія має руйнівні наслідки для суспільства та економіки в усьому світі. Збільшення кількості тестів на COVID-19 дає більше інформації про поширення епідемії, що може призвести до можливості оточувати її, щоб запобігти подальшому зараженню. Проте носіння маски для обличчя, яка запобігає передачі крапель у повітрі, і підтримка належної фізичної дистанції між людьми, а також зменшення тісного контакту один з одним все ще може бути корисним у боротьбі з цією пандемією. Тому ця дослідницька робота зосереджена на реалізації моделі виявлення маски для обличчя та соціального дистанціювання як вбудованої системи зору. Попередньо підготовлені моделі, такі як MobileNet, ResNet Classifier, і VGG використовуються в нашому контексті. Виявлено людей, які не носять маски. Після впровадження та розгортання моделей вибрана досягла 100% довіри. У цій статті також наведено порівняльне дослідження різних моделей визначення обличчя та класифікації масок. Продуктивність системи оцінюється з точки зору точності, запам'ятовування, F1-рахунку, підтримки, чутливості, специфічності та точності, які демонструють практичну застосовність. Система працює з показником F1 99%, чутливістю 99%, специфічністю 99% і точністю 100%. Таким чином, це рішення відстежує людей у масках або без них у режимі реального часу та забезпечує соціальне дистанціювання, генеруючи тривогу, якщо є порушення на місці події або в громадських місцях.

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Опис та постановка задачі

З кінця 2019 року в Ухані вперше повідомили про інфекційну коронавірусну хворобу (COVID-19), і вона стала проблемою суспільної шкоди здоров'ю в Китаї та навіть у всьому світі. Ця пандемія має руйнівні наслідки для суспільства та економіки в усьому світі, спричиняючи глобальну кризу охорони здоров'я [1]. Це нова респіраторна інфекційна хвороба, спричинена тяжким гострим респіраторним синдромом коронавірусу 2 (SARS-CoV-2). [2]. У всьому світі, особливо під час третьої хвилі, COVID-19 був серйозним викликом для охорони здоров'я [3]. Через це було спричинено багато зупинок у різних галузях

пандемія. Крім того, багато секторів, таких як проекти технічного обслуговування та будівництво інфраструктури, не були припинені через їх значний вплив на повсякденне життя людей.[4, 5].

Зараз вірус швидко поширився на більшість країн у всьому світі [2]. Останні статистичні дані (04.05.2021), надані Всесвітньою організацією охорони здоров'я (ВООЗ), показують 152 543 452 підтверджених випадків і 3 198528 смертей. За даними Центру контролю та профілактики захворювань (CDC), коронавірусна інфекція передається переважно дихально-крапельним шляхом, що утворюється, коли люди дихають, розмовляють, кашляють чи чхають.[3] із звичайними розмірами крапель 5–10 м, але викиди аерозолю збільшуються, коли люди говорять і голосно кричать [6].

Тому, щоб запобігти швидкому зараженню COVID-19, більшість світових урядів пропонує багато рішень, таких як ув'язнення та карантин. Однак цю неефективність управління COVID-19 можна додатково дослідити за допомогою теоретичних ігор сценаріїв за межами гри суспільних благ. Зокрема, деякі дослідники зосереджували увагу на ваганні урядів у впровадженні складних, але

необхідних заходів для стримування вірусів (наприклад, накази залишатися вдома та блокування), а також на відмову від співпраці з інших причин, ніж безкоштовна їзда. Наприклад, автори в[7] стверджував, що, оскільки суворі заходи щодо перебування вдома можуть значно вплинути на засоби до існування людей, вартість перебування вдома (у поєднанні з втомою від карантину) може в кінцевому підсумку перевищити ризик зараження від виходу. Оскільки рішення на індивідуальному рівні мають прямий вплив на ефективність наказів про перебування вдома на рівні суспільства, уряди можуть утримуватися від їх виконання через очікуваний низький рівень дотримання, особливо з боку соціально-економічно неблагополучних осіб, які не мають розкіш залишатися вдома[8]. Деякі уряди, можливо, також сподівалися, що колективний імунітет від одужання та вакцинації дозволить їм взагалі уникнути введення таких непопулярних заходів. [9].

Зі зростанням кількості випадків захворювання та розтягненням здоров'яміст, а також відсутність вакцини протягом 2020 року та труднощі, пов'язані з досягненням колективного імунітету проти COVID-19 [10], бездіяльність уряду ставала все більш нежиттєздатною. Отже, щоб посилити дотримання людьми суворих правил, автори в[7] запропонував використовувати соціальні програми, такі як фонди екстреної допомоги та страхування на випадок безробіття, щоб знизити витрати на дотримання вимог, особливо для низькооплачуваних працівників [11]. Оскільки вакцини стали доступними в кінці ст 2020, автори в [12] стверджував, що програми, які сприяють поширенню вакцинації, за важливістю перевершать інші аспекти, такі як ефективність вакцини та процедури ізоляції. Використовуючи EGT, аналіз соціальних мереж та моделювання на основі агентів, автори припустили, що на прийняття рішення щодо вакцинації індивідами впливатимуть «демографічні дані, фізичне місцезнаходження, рівень взаємодії, здоров'я вакцини, параметри епідемії та уявлення про вакцину, що впроваджується, і так само на прийняття урядом рішень

впливатимуть параметри епідемії, характер вакцини, що впроваджується, матеріально-технічне забезпечення, управління людськими ресурсами, необхідними для вакцинації, і кількість вакцин. доступні дози» [12]. Підсумовуючи, комплексне управління COVID-19 передбачало б оцінку багатьох факторів, які калібрують виплати, щоб як індивідуальні, так і урядові рішення зміщувалися в бік безпеки.

Це правда, що COVID-19 є глобальною пандемією і впливаєкілька доменів. Тим не менш, це створило шлях для дослідників у галузі інформатики. Ми бачили багато тем для досліджень, як-от створення нових методів автоматичного виявлення COVID-19 та виявлення людей у масках чи без них. Враховуючи, що в результатах ранніх лабораторних досліджень є деякі помилки та їх затримки, дослідники зосередилися на різних варіантах [13]. Тому застосування передових методів штучного інтелекту (ШІ) [14–17] у поєднанні з рентгенологічною томографією грудної клітки (комптомографія (КТ) і рентген) можуть привести до більш точного виявлення COVID-19 і можуть допомогти контролювати проблему втрати лікарів-спеціалістів в ізольованих селах [18]. У цьому контексті автори [13] запропонував новий метод на основі згорткової нейронної мережі (CNN) для виявлення COVID-19 з аналізом рентгенівських зображень грудної клітки (CXR). Цей метод дозволяє виявити пацієнтів з COVID-19 з точністю 91,34%. в [18], в автори представили нову модель автоматичного виявлення COVID-19 за допомогою CXR-зображень. Модель називається «DarkCovidNet». Для бінарних класів (COVID-19 VS без висновків) точність класифікації, вироблена цією моделлю, становить 98,08%, але для мультикласових випадків (COVID-19 VS пневмонія VS без висновків) точність становить 87,02%. Основною метою є використання таких моделей для діагностики додаткових захворювань грудної клітки, таких як туберкульоз та пневмонія. Для виявлення випадків COVID-19 на рентгенівських знімках глибока модель CNN запропонована в [19]. Ця модель під назвою COVID-Net є відкритим вихідним кодом і доступною для широкої

громадськості. Точність виявлення, досягнута цією моделлю (93,3%), доводить, що модель робить хороші прогнози в покращеному скринінгу. в[20],впроваджуються різні методи глибокого навчання, щоб диференціювати зображення на КТ-скануванні як COVID-19, так і не-COVID-19. З різних методик ми перерахуємо модель власної розробки (CTnet-10), VGG-16, ResNet-50, InceptionV3, VGG-19 і DenseNet-169, які мають, відповідно, значення точності близько 82,1%, 89 %, 60%, 53,4%, 94,52% і 93,15%. Точність ВГГ-19 найвища в порівнянні з іншимимоделі. Метод CTnet-10 – це добре організована модель, яка корисна для лікарів, особливо при масовому скринінгу.

У [21] пропонуються дві моделі глибокого навчання: CNN і згортка довготривала короткочасна пам'ять (ConvLSTM). Для їх моделювання припускаються два набори даних. Набір даних включає КТ-зображення, а інший — рентгенівські зображення. Моделі тестуються чотири рази. Коли вони досліджуються на зображеннях КТ, набір даних розбивається на 70% для навчального набору і 30% для тестового набору. Значення точності для моделі CNN і для ConvLSTM однакове, дорівнює 99%. При тестуванні на розширеному наборі даних А точність тестування CNN становить 99%, але це 100% для ConvLSTM. При тестуванні на розширеному наборі даних В точність тестування CNN становить 100%, але для ConvLSTM вона становить 99%. Коли обидві моделі тестуються на комбінованому наборі даних, що містить як рентгенівські, так і КТ-зображення, точність тестування становить 99% для CNN і 98% для ConvLSTM. нарешті, коли вони тестуються на наборі даних рентгенографії, точність тестування становить 95% і 88% для CNN і ConvLSTM відповідно. Цей сценарій ми можемо розглядати як складний, оскільки він покликаний розрізняти дві хвороби (COVID-19 і пневмонію) з високою близькістю за ознаками.

До коронавірусу деякі люди одягали маски для захистусебе від забруднення повітря, а інші люди одягають маски, щоб приховати своє обличчя та свої емоції від інших. За словами ВООЗ, захист від коронавірусу є обов’язковою

протидією.1]. Дійсно, носіння маски є ефективним методом блокування 80 усіх респіраторних інфекцій. [3]. Також ВООЗ рекомендує дотримуватися фізичного дистанціювання, щоб пом’якшити поширення вірусу. У всьому світі уряди борються з цим типом вірусу. Багато організацій дотримуються правил масок для обличчя для особистого захисту. Перевірити вручну, чи носять маски особи, які входять до організації, є громіздким і, можливо, суперечливим [1]. У цьому контексті автори в [6] запропонував модель на основі глибокого навчання під назвою MobileNet Mask, щоб запобігти передачі SARS-CoV-2 від людини до людини та виявити обличчя з маскою або без неї. Для навчання та тестування моделі використовуються два різних набори даних (IDS1 та IDS2) із понад 5200 зображеннями. Усі експериментальні випадки контролюються в Google Colab, який працює у хмарі. У IDS1 запропонована модель досягла точності тестування 93%. Проте в IDS2 досягнута точність становить майже 100%.

В [22], автори прагнуть виявити та розмежувати медичні маски для обличчя в реальних зображеннях. Запропонована модель складається з YOLO-V2 і ResNet-50. На етапі навчання автори використовували два оптимізатори: Adam і SGDM. Під час цього процесу SGDM краще, ніж Adam, щодо середньоквадратичної помилки перевірки (RMSE), часу та втрат при перевірці. Однак Адам кращий, ніж SGDM, у втратах і міні-партії RMSE. Середня точність (AP) оптимізатора Адама становить 0,81, що краще, ніж AP SGDM, що дорівнює 0,61. Більше того, середні логарифмічні показники промахів оптимізатора Адама становлять 0,4, що є кращими, ніж середні логарифмічні показники промахів оптимізатора SGDM (0,6), на всіх рівнях відкликання. З цих досліджень ми також виявили, в[23], система, яка запускає тривогу в операційній, коли медичний персонал не носить маски. Система поєднує в собі два детектори, для обличчя та для масок. На етапі тестування, використовуючи зображення з набору даних BAO і з власного набору даних зображень, досягається рівень істинно позитивних вище 95% і хибнопозитивних нижче 5%.

Автори в [1] запропонував гібридну модель з використанням глибоких навчання за допомогою класичного машинного навчання для виявлення облич у масках. Запропонована модель складається з двох компонентів: вилучення ознак за допомогою ResNet-50 та процесу класифікації. Три використовувані класифікатори – це дерева рішень (DT), машина опорних векторів (SVM) і алгоритм ансамблю. Набір даних із замаскованими обличчями реального світу (RMFD), набір даних імітованих масок обличчя (SMFD) та мічені обличчя в дикій природі (LFW) — це три набори даних із замаскованим обличчям, вибрані для перевірки. Класифікатор SVM є кращим за інші класифікатори. Він досяг 99,64%, 99,46% і 100% точності тестування відповідно в RMFD, SMFD і LFW.

Підсумовуючи і не забуваючи про юридичну сторону ШІ, методика глибокого навчання за своєю суттю зачіпає повний спектр правових сфер, починаючи від філософії права, прав людини, договірного права, права деліктів, трудового права, кримінального права, податкового права, процесуального права, і т. д. Хоча на практиці ШІ тільки починає реалізовуватись з точки зору його використання юристами, а в юридичній галузі науковці-юристи вже тривалий час займаються ШІ. Крім того, оскільки збір та аналіз даних поступово поширюється від компаній-розробників програмного забезпечення до виробничих компаній, які почали використовувати можливості, що виникають у результаті збору та використання потенційних даних, для створення додаткової вартості, цей інформаційний потік відкриває різні можливості. юридичні проблеми, які могли б стимулювати регуляторний

зворотна реакція. Враховуючи юридичні проблеми та переваги штучного інтелекту в боротьбі з пандемією COVID-19, рішення, засновані на техніках AI, все ще залишаються відкритим вікном для розробки та юридичного тлумачення[24].

Нагадування про цю роботу організовано таким чином: Розділ 2.1 підсумовує нещодавню пов’язану роботу в пропон контекст. Розділ 2.3

представляє запропоновану структуру. Після цього проводиться попереднє дослідження. Розділ 2.4 позначає набір даних. Після цього ми вводимо показники оцінки в Розділ 2.5, тоді як числовий результат обговорюється в Розділ 3.1 . Нарешті, ми завершуємо цю роботу в Розділі 3.1.

1.2 Огляд основи машинного навчання

З кінця 2019 року в Ухані вперше повідомили про інфекційну коронавірусну хворобу (COVID-19), і вона стала проблемою суспільної шкоди здоров'ю в Китаї та навіть у всьому світі. Ця пандемія має руйнівні наслідки для суспільства та економіки в усьому світі, спричиняючи глобальну кризу охорони здоров'я [1]. Це нова респіраторна інфекційна хвороба, спричинена тяжким гострим респіраторним синдромом коронавірусу 2 (SARS-CoV-2) [2]. У всьому світі, особливо під час третьої хвилі, COVID-19 був серйозним викликом для охорони здоров'я [3]. Через це було спричинено багато зупинок у різних галузях

пандемія. Крім того, багато галузей, таких як проекти технічного обслуговування та будівництво інфраструктури, не були припинені через їх значний вплив на повсякденне життя людей [4, 5].

На даний момент вірус швидко поширився на більшість країн світу [2]. Останні статистичні дані (04.05.2021), надані Всесвітньою організацією охорони здоров'я (ВООЗ), показують 152 543 452 підтверджених випадків і 319 8528 смертей. За даними Центру з контролю та профілактики захворювань (CDC), коронавірусна інфекція передається переважно повітряно-крапельним шляхом, що утворюється, коли люди дихають, розмовляють, кашляють або чхають [3], звичайні краплі розміром 5–10 м, але викиди аерозолю збільшуються. коли люди говорять і голосно кричать [6].

Таким чином, щоб запобігти швидкому зараженню COVID-19, більшість світових урядів пропонує багато рішень, таких як утримання та карантинні заходи.

Однак цю неефективність управління COVID-19 можна додатково дослідити за допомогою теоретичних ігор сценаріїв за межами гри суспільних благ. Зокрема, деякі дослідники зосереджували увагу на ваганні урядів у впровадженні складних, але необхідних заходів для стримування вірусів (наприклад, накази залишатися вдома та карантини), а також на відмову від співпраці з інших причин, ніж безкоштовна їзда. Наприклад, автори в [7] стверджували, що, оскільки суворі заходи щодо перебування вдома можуть сильно вплинути на засоби до існування людей, вартість перебування вдома (у поєднанні з втомою від карантину) може в кінцевому підсумку перевищувати ризик зараження від виходу. Оскільки рішення на індивідуальному рівні мають прямий вплив на ефективність наказів про перебування вдома на рівні суспільства, уряди можуть утримуватися від їх виконання через очікуваний низький рівень дотримання, особливо з боку соціально-економічно неблагополучних осіб, які не мають розкіш залишатися вдома [8]. Деякі уряди, можливо, також сподівалися, що колективний імунітет від одужання та вакцинації дозволить їм взагалі уникнути введення таких непопулярних заходів [9].

Зі збільшенням кількості випадків захворювання та розтягненням здоров’ямістах, а також відсутність вакцини протягом 2020 року та труднощі, пов’язані з досягненням колективного імунітету проти COVID-19 [10], бездіяльність уряду ставала все більш нежиттєздатною. Отже, щоб підвищити дотримання людьми суворих правил, автори в [7] запропонували використовувати соціальні програми, такі як фонди екстреної допомоги та страхування на випадок безробіття, щоб знизити витрати на дотримання, особливо для низькооплачуваних працівників [11]. Оскільки вакцини стали доступними наприкінці 2020 року, автори роботи [12] стверджували, що програми, що стимулюють запровадження вакцинації, перевершать за важливістю інші аспекти, такі як ефективність вакцини та процедури ізоляції. Використовуючи EGT, аналіз соціальних мереж та моделювання на основі агентів, автори припустили, що на прийняття рішення

щодо вакцинації індивідами впливатимуть «демографічні дані, фізичне місцезнаходження, рівень взаємодії, здоров'я вакцини, параметри епідемії та уявлення про вакцину, що впроваджується, і так само на прийняття урядом рішень впливатимуть параметри епідемії, характер вакцини, що впроваджується, матеріально-технічне забезпечення, управління людськими ресурсами, необхідними для вакцинації, і кількість вакцин. доступні дози» [12]. Підсумовуючи, комплексне управління COVID-19 передбачало б оцінку багатьох факторів, які калібрують виплати, щоб як індивідуальні, так і урядові рішення зміщувалися в бік безпеки.

Це правда, що COVID-19 є глобальною пандемією і впливаєкілька доменів. Тим не менш, це створило шлях для дослідників у галузі інформатики. Ми бачили кілька тем дослідження, як-от створення нових методів автоматичного виявлення COVID-19 та виявлення людей у масках чи без них. Враховуючи певні помилки в результатах ранніх лабораторних досліджень та їх затримки, дослідники зосередилися на різних варіантах [13]. Тому застосування передових методів штучного інтелекту (ШІ) [14–17] у поєднанні з рентгенологічною томографією грудної клітки (комп'ютерна томографія (КТ) і рентген) можуть привести до більш точного виявлення COVID-19 і можуть допомогти контролювати проблему втрати лікарів-спеціалістів в ізольованих селах [18]. У цьому контексті автори в [13] запропонували новий метод на основі згорткової нейронної мережі (CNN) для виявлення COVID-19 з аналізом рентгенівських зображень грудної клітки (CXR). Цей метод дозволяє виявити пацієнтів з COVID-19 з точністю 91,34%. У роботі [18] автори представили нову модель автоматичного виявлення COVID-19 за допомогою CXR-зображень. Модель називається «DarkCovidNet». Для бінарних класів (COVID-19 VS без висновків) точність класифікації, вироблена цією моделлю, становить 98,08%, але для мультикласових випадків (COVID-19 VS пневмонія VS без висновків) точність становить 87,02%. Основною метою є використання таких моделей для діагностики додаткових захворювань грудної

клітки, таких як туберкульоз та пневмонія. Для виявлення випадків COVID-19 за рентгенівськими зображеннями в [19] запропонована глибока модель CNN. Ця модель під назвою COVID-Net є відкритим вихідним кодом і доступною для широкої громадськості. Точність виявлення, досягнута цією моделлю (93,3%), доводить, що модель робить хороші прогнози в покращеному скринінгу. У [20] представлені різні методи глибокого навчання для диференціації зображень комп'ютерної томографії як COVID-19, так і не-COVID-19. З різних методик ми перерахуємо модель власної розробки (CTnet-10), VGG-16, ResNet-50, InceptionV3, VGG-19 і DenseNet-169, які мають, відповідно, значення точності близько 82,1%, 89 %, 60%, 53,4%, 94,52% і 93,15%. точність ВГГ-19 найвища в порівнянні з іншими моделі. Метод CTnet-10 – це добре організована модель, яка корисна для лікарів, особливо при масовому скринінгу. У [21] пропонуються дві моделі глибокого навчання: CNN і згортка довготривала короткочасна пам'ять (ConvLSTM). Для їх моделювання припускаються два набори даних. Набір даних включає КТ-зображення, а інший — рентгенівські зображення. Моделі тестуються чотири рази. Коли вони досліджуються на зображеннях КТ, набір даних розбивається на 70% для навчального набору і 30% для тестового набору. Значення точності для моделі CNN і для ConvLSTM однакове, дорівнює 99%. При тестуванні на розширеному наборі даних А точність тестування CNN становить 99%, але це 100% для ConvLSTM. При тестуванні на розширеному наборі даних В точність тестування CNN становить 100%, але для ConvLSTM вона становить 99%. Коли обидві моделі тестуються на комбінованому наборі даних, що містить як рентгенівські, так і КТ-зображення, точність тестування становить 99% для CNN і 98% для ConvLSTM. Нарешті, коли вони перевіряються на наборі даних рентгенографії, точність тестування становить 95% і 88% для CNN і ConvLSTM відповідно. Цей сценарій ми можемо розглядати як складний, оскільки він покликаний розрізняти дві хвороби (COVID-19 і пневмонію) з високою близькістю за ознаками.

До коронавірусу деякі люди одягали маски для захисту себе від забруднення повітря, а інші люди одягають маски, щоб приховати своє обличчя та свої емоції від інших. Захист від коронавірусу є обов'язковим контрзаходом, згідно з ВООЗ [1]. Дійсно, носіння маски є ефективним методом блокування 80 усіх респіраторних інфекцій [3]. Також ВООЗ рекомендує дотримуватися фізичного дистанціювання, щоб пом'якшити поширення вірусу. У всьому світі уряди борються з цим типом вірусу. Багато організацій дотримуються правил масок для обличчя для особистого захисту. Перевірити вручну, чи носять особи, які входять до організації, маски, є громіздким і, можливо, суперечливим [1]. У цьому контексті автори в [6] запропонували модель на основі глибокого навчання під назвою MobileNet Mask, щоб запобігти передачі SARS-CoV-2 від людини до людини та виявити обличчя з маскою або без неї. Для навчання та тестування моделі використовуються два різних набори даних (IDS1 та IDS2) із понад 5200 зображеннями. Усі експериментальні випадки контролюються в Google Colab, який працює у хмарі. У IDS1 запропонована модель досягла точності тестування 93%. Проте в IDS2 досягнута точність становить майже 100%.Це правда, що COVID-19 є глобальною пандемією і впливає кілька доменів. Тим не менш, це створило шлях для дослідників у галузі інформатики. Ми бачили кілька тем дослідження, як-от створення нових методів автоматичного виявлення COVID-19 та виявлення людей у масках чи без них. Враховуючи певні помилки в результатах ранніх лабораторних досліджень та їх затримки, дослідники зосередилися на різних варіантах [13]. Тому застосування передових методів штучного інтелекту (ШІ) [14–17] у поєднанні з рентгенологічною томографією грудної клітки (комп'ютерна томографія (КТ) і рентген) можуть привести до більш точного виявлення COVID-19 і можуть допомогти контролювати проблему втрати лікарів-спеціалістів в ізольованих селах [18]. У цьому контексті автори в [13] запропонували новий метод на основі згорткової нейронної мережі (CNN) для виявлення COVID-19 з аналізом рентгенівських зображень грудної клітки (CXR).

Цей метод дозволяє виявити пацієнтів з COVID-19 з точністю 91,34%. У роботі [18] автори представили нову модель автоматичного виявлення COVID-19 за допомогою CXR-зображень. Модель називається «DarkCovidNet». Для бінарних класів (COVID-19 VS без висновків) точність класифікації, вироблена цією моделлю, становить 98,08%, але для мультикласових випадків (COVID-19 VS пневмонія VS без висновків) точність становить 87,02%. Основною метою є використання таких моделей для діагностики додаткових захворювань грудної клітки, таких як туберкульоз та пневмонія. Для виявлення випадків COVID-19 за рентгенівськими зображеннями в [19] запропонована глибока модель CNN. Ця модель під назвою COVID-Net є відкритим вихідним кодом і доступною для широкої громадськості. Точність виявлення, досягнута цією моделлю (93,3%), доводить, що модель робить хороші прогнози в покращеному скринінгу. У [20] представлені різні методи глибокого навчання для диференціації зображень комп'ютерної томографії як COVID-19, так і не-COVID-19. З різних методик ми перерахуємо модель власної розробки (CTnet-10), VGG-16, ResNet-50, InceptionV3, VGG-19 і DenseNet-169, які мають, відповідно, значення точності близько 82,1%, 89 %, 60%, 53,4%, 94,52% і 93,15%. точність ВГГ-19 найвища в порівнянні з іншими моделі. Метод CTnet-10 – це добре організована модель, яка корисна для лікарів, особливо при масовому скринінгу.

У [21] пропонуються дві моделі глибокого навчання: CNN і згортка довготривала короткочасна пам'ять (ConvLSTM). Для їх моделювання припускаються два набори даних. Набір даних включає КТ-зображення, а інший — рентгенівські зображення. Моделі тестуються чотири рази. Коли вони досліджуються на зображеннях КТ, набір даних розбивається на 70% для навчального набору і 30% для тестового набору. Значення точності для моделі CNN і для ConvLSTM однакове, дорівнює 99%. При тестуванні на розширеному наборі даних A точність тестування CNN становить 99%, але це 100% для ConvLSTM. При тестуванні на розширеному наборі даних B точність тестування

CNN становить 100%, але для ConvLSTM вона становить 99%. Коли обидві моделі тестуються на комбінованому наборі даних, що містить як рентгенівські, так і КТ-зображення, точність тестування становить 99% для CNN і 98% для ConvLSTM. Нарешті, коли вони перевіряються на наборі даних рентгенографії, точність тестування становить 95% і 88% для CNN і ConvLSTM відповідно. Цей сценарій ми можемо розглядати як складний, оскільки він покликаний розрізняти дві хвороби (COVID-19 і пневмонію) з високою близькістю за ознаками.

До коронавірусу деякі люди одягали маски для захисту вірусу. Багато організацій дотримуються правил масок для обличчя для особистого захисту. Перевірити вручну, чи носять особи, які входять до організації, маски, є громіздким і, можливо, суперечливим [1]. У цьому контексті автори в [6] запропонували модель на основі глибокого навчання під назвою MobileNet Mask, щоб запобігти передачі SARS-CoV-2 від людини до людини та виявити обличчя з маскою або без неї. Для навчання та тестування моделі використовуються два різних набори даних (IDS1 та IDS2) із понад 5200 зображеннями. Усі експериментальні випадки контролюються в Google Colab, який працює у хмарі. У IDS1 запропонована модель досягла точності тестування 93%. Проте в IDS2 досягнута точність становить майже 100%.

У всьому світі уряди борються з цим типом вірусу. Багато організацій дотримуються правил масок для обличчя для особистого захисту. Перевірити вручну, чи носять особи, які входять до організації, маски, є громіздким і, можливо, суперечливим [1]. У цьому контексті автори в [6] запропонували модель на основі глибокого навчання під назвою MobileNet Mask, щоб запобігти передачі SARS-CoV-2 від людини до людини та виявити обличчя з маскою або без неї. Для навчання та тестування моделі використовуються два різних набори даних (IDS1 та IDS2) із понад 5200 зображеннями. Усі експериментальні випадки контролюються в Google Colab, який працює у хмарі. У IDS1 запропонована

модель досягла точності тестування 93%. Проте в IDS2 досягнута точність становить майже 100%.

У роботі [22] автори мають на меті виявити та розмежувати медичні маски для обличчя на реальних зображеннях. Запропонована модель складається з YOLO-V2 і ResNet-50. На етапі навчання автори використовували два оптимізатори: Adam і SGDM. Під час цього процесу SGDM краще, ніж Adam, щодо середньоквадратичної помилки перевірки (RMSE), часу та втрат при перевірці. Однак Адам кращий, ніж SGDM, у втратах і міні-партії RMSE. Середня точність (AP) оптимізатора Адама становить 0,81, що краще, ніж AP SGDM, що дорівнює 0,61. Більше того, середні логарифмічні показники промахів оптимізатора Адама становлять 0,4, що краще, ніж середні логарифмічні показники промахів оптимізатора SGDM (0,6), на всіх рівнях відкликання. З цих досліджень ми також знайшли в [23] систему, яка запускає тривогу в операційній, коли медичний персонал не носить маски. Система поєднує в собі два детектори, для обличчя та для масок. На етапі тестування, використовуючи зображення з набору даних BAO і з власного набору даних зображень, досягається рівень істинно позитивних вище 95% і хибнопозитивний менше 5%.

Автори в [1] запропонували гібридну модель із використанням глибини навчання за допомогою класичного машинного навчання для виявлення облич у масках. Запропонована модель складається з двох компонентів: вилучення ознак за допомогою ResNet-50 та процесу класифікації. Три використовувані класифікатори – це дерева рішень (DT), машина опорних векторів (SVM) і алгоритм ансамблю. Набір даних із замаскованими обличчями реального світу (RMFD), набір даних імітованих масок обличчя (SMFD) та мічені обличчя в дикій природі (LFW) — це три набори даних із замаскованим обличчям, вибрані для перевірки. Класифікатор SVM є кращим за інші класифікатори. Він досяг 99,64%, 99,46% і 100% точності тестування відповідно в RMFD, SMFD і LFW.

Підсумовуючи і не забуваючи про юридичну сторону ШІ, техніка глибокого навчання за своєю суттю зачіпає повний спектр правових сфер, починаючи від філософії права, прав людини, договірного права, права деліктів, трудового права, кримінального права, податкового права, процесуального права, і т. д. Хоча на практиці ШІ тільки починає реалізовуватись з точки зору його використання юристами, а в юридичній галузі науковці-юристи вже тривалий час займаються ШІ. Крім того, оскільки збір та аналіз даних поступово поширюється від компаній-розробників програмного забезпечення до виробничих компаній, які почали використовувати можливості, що виникають у результаті збору та використання потенційних даних, для створення додаткової вартості, цей інформаційний потік відкриває різні юридичні проблеми, які могли б стимулювати регуляторний зворотна реакція. Враховуючи юридичні проблеми та переваги штучного інтелекту в боротьбі з пандемією COVID-19, рішення, засновані на техніці штучного інтелекту, все ще залишаються відкритим вікном для розробки та юридичного тлумачення [24].

Нагадування про цю роботу організовано таким чином: Розділ 2 підсумовує нещодавню пов’язану роботу в запропонованому контексті. У розділі 3 представлено запропоновану структуру. Після цього в Розділі 4 наведено попереднє дослідження. Розділ 5 позначає збір даних. Після цього ми вводимо метрики оцінювання в Розділі 6, а числовий результат обговорюється в Розділі 7. Нарешті, ми завершуємо цю роботу в Розділі 8.

1.3 Аналіз зображень для машинного навчання

Перш ніж стане можливим використання ANN як моделі з використанням алгоритмів машинного навчання, необхідно належним чином опрацювати зображення, яке можна розділити на кілька етапів, як показано на рисунку 2.1.

Сегментація - це перша діяльність, що виконується в процесі машинного навчаннянавчання розпізнавання предметів [3, 14].

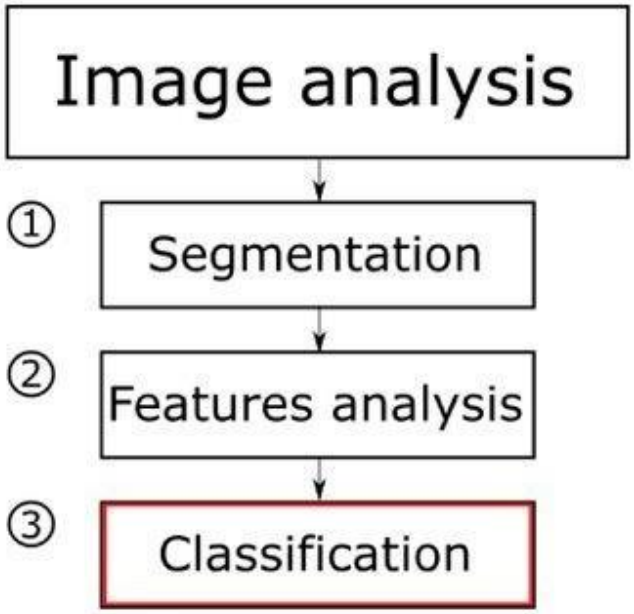

Рисунок 2.1 – Сегментація

Зображення тут поділено на частини, які якимось чином пов’язані між собою. Це для попередньої ізоляції областей, що належать даним об’єктам, її меж, формування або обмеження надсилання на наступний етап непотрібної інформації в пам’яті комп’ютера.

Наступний етап - це аналіз особливостей зображення, що дає змогу виявити та описати властивості об’єкта, які часто залишаються непоміченими лише за допомогою очей.

Об’єкти об’єктів можна об’єднати в кілька основних груп, наприклад геометричні, негеометричні, топологічні. Вибір того, які ознаки зображення слід аналізувати, є індивідуальною справою, залежно від кінцевого результату [4, 5, 10]. Однак для цілей цього алгоритму найбільш важливою є математична сторона обробки аналізованих рис зображення, в результаті чого виникають так звані підписи та скелети, тобто одновимірні функції, що представляють контуриоб’єкта.

Після виконання цих попередніх дій відбувається процес розпізнавання об’єктів за допомогою штучних нейронних мереж, в якому можна використовувати методи глибокого навчання.

1.4 Штучні нейронні мережі

Машинне навчання (ML) стало одним з найбільш широко використовуваних методів штучного інтелекту для кількох компаній, установ та приватних осіб, які займаються автоматизацією. Це пояснюється значним поліпшенням доступу до даних та збільшенням обчислювальної потужності, що дозволяє практикам досягати значущих результатів у кількох областях.

Сьогодні, коли справа доходить до зображень, алгоритми ML можуть інтерпретувати зображення так само, як і наш мозок. Вони використовуються практично скрізь, починаючи від розпізнавання облич, під час зйомки зображень на наших смартфонах, автоматизації стомлювальної роботи вручну, автомобілів, що керують автомобілем, і всього іншого.

У цьому блозі ми детально зануримось у основи машинного навчання зображень та обговоримо різні технології, які ми могли б використати для побудови найсучасніших алгоритмів на основі даних зображення.

Це одна з швидкозростаючих технологій, яка з роками широко розвивалася. Сьогодні кілька компаній та організацій різних секторів використовують обробку зображень для кількох застосувань, таких як візуалізація, вилучення інформації про зображення, розпізнавання образів, класифікація, сегментація та багато іншого!

Перш за все, існує два способи обробки зображень: аналоговий та цифровий. Метод аналогового IP застосовується до друкованих копій, таких як відскановані фотографії та роздруківки, і тут зазвичай виводяться зображення. Для порівняння, цифровий IP використовується для маніпулювання цифровими зображеннями за

допомогою комп'ютерів; тут зазвичай виводиться інформація,пов'язана з цим зображенням, наприклад дані про особливості, характеристики, обмежувальні рамки або маски.

Як обговорювалося з машинним навчанням та глибоким навчанням, методи обробки зображень можуть стати більш потужними.Самостійне керування автомобілями, стимуляція зображення Самостійні автомобілі, стимуляція зображення. Ось деякі звичні випадки використання, які використовують методи обробки зображень ML:

- медична візуалізація / візуалізація: Допоможіть медичним працівникам інтерпретувати медичні зображення та швидше діагностувати аномалії;
- правоохоронні органи та безпека: Допомога у спостереженні та біометричній автентифікації;
- технологія самокерування: допомагайте виявляти об’єкти та імітувати людські візуальні підказки та взаємодію;
- ігри: Покращення ігрового досвіду доповненої реальності та віртуальної реальності;
- відновлення та різкість зображення: Покращте якість зображень або додайте популярні фільтри тощо;
- розпізнавання візерунків: класифікуйте та розпізнайте об’єкти/візерунки на зображеннях та розумійте контекстну інформацію.Зображення зображень: розпізнає зображення для швидшого пошуку з великих наборів даних;
- у наступному розділі ми вивчимо основи роботи з машинним навчанням обробки зображень;
- робота з машинного навчання обробки зображень;
- як правило, алгоритми машинного навчання мають певний конвеєр або кроки для вивчення даних. Давайте візьмемо загальний приклад того ж

самого та змоделюємо робочий алгоритм для випадку використання обробки зображень.

По-перше, алгоритмам ML потрібна значна кількість високоякісних даних для вивчення та прогнозування високоточних результатів. Отже, нам доведеться переконатися, що зображення добре оброблені, коментовані та загальні для обробки зображень ML. Тут з'являється комп'ютерне бачення (CV); це область, яка стосується здатності машин розуміти дані зображення. Використовуючи резюме, ми можемо обробляти, завантажувати, трансформувати та обробляти зображення для створення ідеального набору даних для алгоритму машинного навчання.

Наприклад, скажімо, ми хочемо побудувати алгоритм, який передбачить, чи є на даному зображенні собака чи кішка. Для цього нам потрібно буде зібрати зображення собак і кішок і попередньо обробити їх за допомогою резюме. Етапи попередньої обробки включають:

- перетворення всіх зображень в один формат. Обрізання непотрібних областей на зображеннях.
- перетворення їх у числа, щоб алгоритми могли їх вивчити (масив чисел). Комп'ютери бачать вхідне зображення у вигляді масиву пікселів, і це залежить від роздільної здатності зображення. Виходячи з роздільної здатності зображення, він побачить розмір висоти * ширини *. Наприклад, зображення масиву 6 x 6 x 3 матриці RGB (3 відноситься до значень RGB) та зображення
- масиву 4 x 4 x 1 матриці зображення в градаціях сірого;

Після обчислення суми e, використовуючи вагові коефіцієнти wi та вхідні сигнали xi, результат множиться на функцію активації f, яка повинна відповідати таким умовам:

- безперервність зміни між вашим мінімальним і максимальним значеннями;

• безперервність похідних (розрахувати похідну також не складно).

Була використана функція активації ReLU, параметри якої зображені на рисунку 2.2. Цей тип активаційної функції в даний час є найбільш використовуваною функцією активації для навчання нейронних мереж, призначених для розпізнавання об’єктів на зображеннях. Це пов'язано з її нескінченно прагнучою реакцією на дійсний сигнал і обнуленням значення нейрона для негативного сигналу.

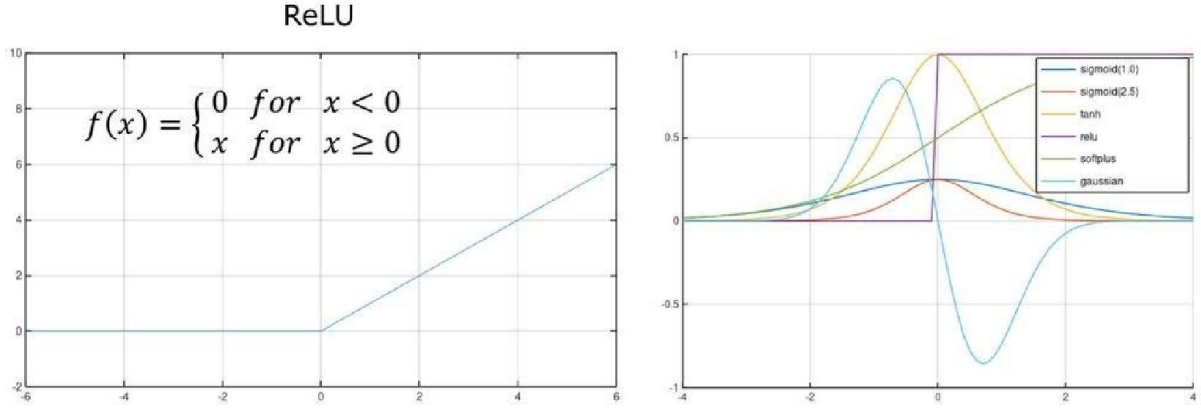

Рисунок 2.2 – Функція активації ReLU.

Такий підхід означає, що не всі нейрони використовуються в мережі, що захищає її від переобладнання та прискорює процес мережевого навчання. Крім того, ReLU дуже просто обчислює похідну, а її лінійність дозволяє використовувати метод поширення помилок назад для корекції вагових коефіцієнтів у мережі.

1.5 Моделі нейронних мереж

Створення програми для розпізнавання об'єктів на зображенні вимагає розробки математичної моделі та комплексного підходу через велику кількість даних, необхідних для обробки та класифікації. Одним із можливих рішень для

розпізнавання об’єктів на зображенні, завдяки ефективності роботи, євикористанняANN.

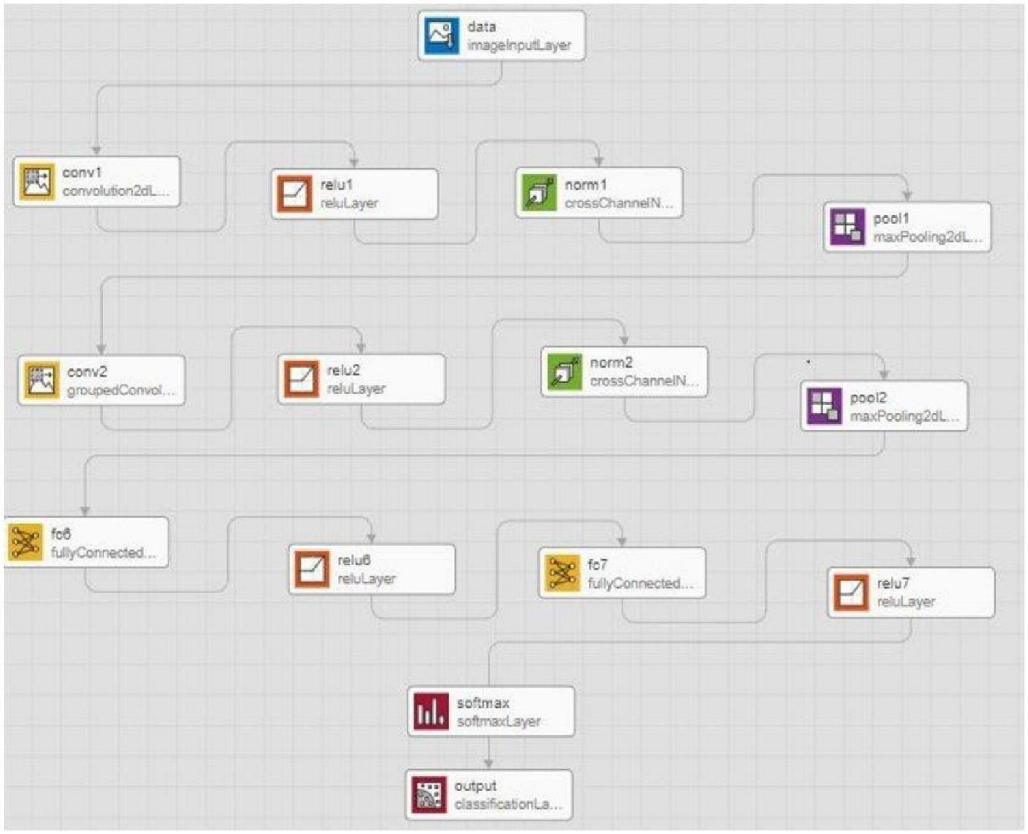

Рисунок 2.3 – Діаграма нейронної мережі, що використовується в цій програмі, зображена на рисунку.

Це вихідна мережа. Інформація в цьому типі мережевої архітектури рухається лише вперед, рис. 2.4 Мережа складається з декількох шарів, що є дуже поширеним рішенням [20]. В результаті утворюється складна структура з великою кількістю з'єднань, схильна до так званого переобладнання. Мережа складається з вхідного шару, прихованих шарів та вихідного шару, які за своєю природою можна розділити на дві групи:

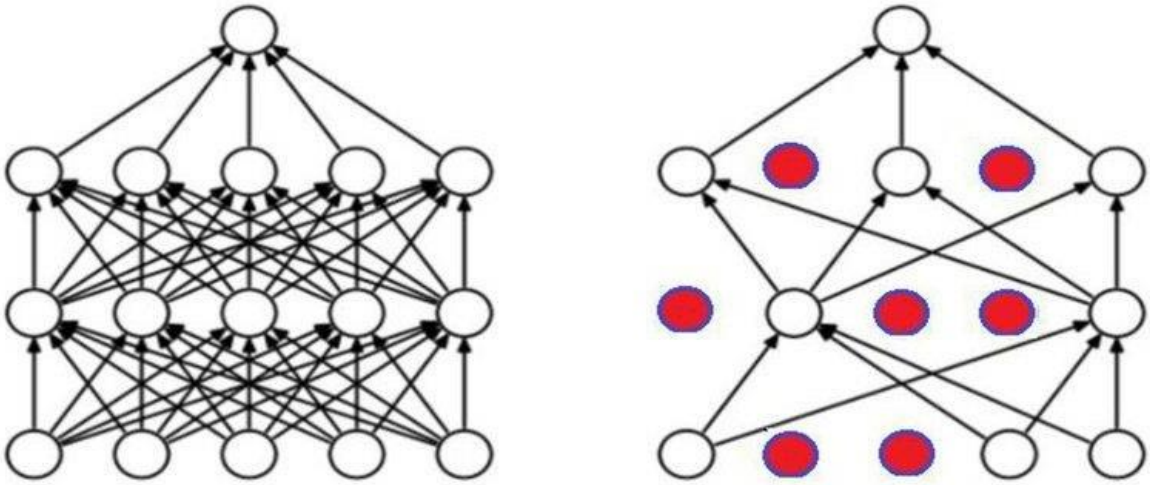

Рисунок 2.4 – Вихідна мережа.

- перші три шари мережі (включаючи вхідний шар), які завдяки використанню фільтрувальних масок, адаптованих до очікуваних характеристик, зменшують «розмірність» зображення, а потім використовують характерний ієрархічний рисунку, що міститься в дані (декілька характерних ознак відповідають за розпізнавання форми, а не всього зображення, оскільки деякі пікселі несуть більше інформації, ніж інші), дозволяють використовувати менше нейронів і зв'язків між ними для досягнення того самого ефекту. Це особливо важливо у випадку багатошарових мереж, схильних до перенапруження даних;
- шари з повним зв'язком нейронів один з одним (ця мережа все ще одностороння), які є останніми трьома шарами для обробки даних (включаючи вихідний шар, з кількістю нейронів, рівною кількості об'єктів, які можна класифікувати)) і два шари з 4096 нейронами, які відповідають за математичні розрахунки. На рисунку 4 показані відмінності між архітектурою шарів;

Використання цього алгоритму дозволяє прискорити роботу нейронної мережі, оскільки програміст діє як учитель, який знає, яких значень він очікує на виході нейронної мережі, визначаючи правильність результату алгоритму. Коли

розбіжність занадто велика, він дає запропоновану зміну значення, що дозволить правильно розпізнати об'єкт, і вказує алгоритму зробити наступну ітерацію - мережа дізнається, знаючи, який результат вона повинна отримати,тому спочатку випадкові значення для окремих нейронів будуть швидко встановлюється на рівні, що дозволяє передбачувану роботу мережі. Ідея методу зворотного розповсюдження показана на рисунку 2.5.

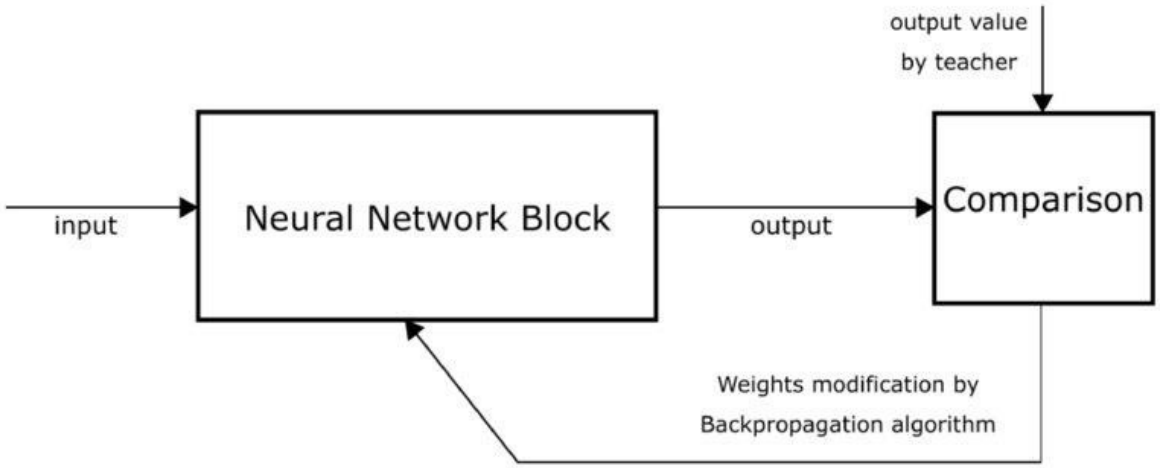

Рисунок 2.5 – методу зворотного розповсюдження

Цей метод використовується для розрахунку значень нейронів - він робить це за допомогою математичних формул. Основою його функціонування є використання знань про результат, який необхідно отримати на виході з мережі. Потім помилка обчислюється між запропонованим значенням і отриманим мережею, і помилка виправляється шляхом зміни значення нейронів [11]. Це просте завдання ускладнюється мережею, що складається з кількох шарів, що є основним типом, який використовується в більш просунутих алгоритмах штучного інтелекту. В цьому випадку алгоритм зворотного розповсюдження підсумовує помилки нейронів із прихованого шару, що передує останньому зміненому шару, і лише потім виправляє їх значення та ваги між ними.

Операції алгоритму можна представити так:

- випадкове визначення вагових значень і призначення їх нейрону;

- завантаження вхідних даних (вже в математичній формі);
- введення запропонованого вихідного значення та на його основі розрахунок значень на виходах нейронів та їх порівняння. Розрахунок помилки вихідного сигналу мережі;
- розрахунок похибок у прихованому шарі за вихідним шаром з урахуванням похибки вихідного шару та суми помилок прихованого шару (підтримання ваг між нейронами);
- повторення процедури для наступного прихованого шару з використанням підсумованих помилок та помилки на виході, що передує прихованому значенню, обчисленому на попередньому кроці;
- повторення процедури для наступного прихованого шару з використанням його загальних помилок та помилки виводу з попереднього прихованого шару, обчисленого на попередньому кроці;
- після обчислення похибок у всіх прихованих шарах алгоритм змінює значення нейронів на вхідному шарі однаково;
- усі ваги в мережі змінені;
- помилка зменшується з кожним повним проходженням алгоритму, поки вона не впаде до прийнятного для вчителя рівня;

РОЗДІЛ 2
РЕАЛІЗАЦІЯ МЕТОДУ ВИЯВЛЕННЯ ОБЛИЧА

2.1 Вибір метода виявлення обличчя

У методі виявлення обличчя обличчя виявляється на зображенні, яке має кілька атрибутів. Відповідно до [21], дослідження виявлення обличчя вимагає розпізнавання виразів, відстеження обличчя та оцінки пози. Враховуючи одиночне зображення, завдання полягає в тому, щоб визначити обличчя на зображенні. Розпізнавання облич є складним завданням, оскільки обличчя змінюються за розміром, формою, кольором тощо, і вони не є незмінними. Це стає трудомісткою роботою для непрозорого зображення, якому перешкоджає щось інше, що не зустрічається з камерою, і так далі. Автори роботи [22] вважають, що виявлення оклюзійних облич супроводжується двома основними проблемами: 1) недоступність значно об'ємних наборів даних, які містять як замасковані, так і немасковані обличчя, і 2) виключення виразу обличчя в охопленій області. Використовуючи алгоритм локального лінійного вбудовування (LLE) і словники, навчені на величезній кількості замаскованих облич, синтезованих повсякденних облич, можна відновити декілька втрачених виразів і значною мірою пом'якшити вплив ознак обличчя. Згідно з роботою, описаною в [11], згорткові нейронні мережі (CNN) в комп'ютерному баченні мають суворі обмеження щодо розміру вхідного зображення. Поширена практика змінює конфігурацію зображень, перш ніж вставляти їх у мережу, щоб подолати гальмування. згорткові нейронні мережі (CNN) в комп'ютерному баченні мають суворі обмеження щодо розміру вхідного зображення. Поширена практика змінює конфігурацію зображень, перш ніж вставляти їх у мережу, щоб подолати гальмування. згорткові нейронні мережі (CNN) в комп'ютерному баченні мають суворі обмеження щодо розміру вхідного

зображення. Поширена практика змінює конфігурацію зображень, перш ніж вставляти їх у мережу, щоб подолати гальмування.

Тут основним завданням завдання є правильно визначити обличчя на зображенні, а потім визначити, чи є на ньому маска чи ні. Для виконання завдань спостереження запропонований метод повинен також виявляти обличчя разом із маскою в русі.

У цьому розділі ви дізнаєтеся, чому можна очікувати різних результатів при використанні алгоритмів машинного навчання.

Після заповнення цього розділу ви дізнаєтеся:

- алгоритми машинного навчання навчатимуть різні моделі, якщо буде змінено навчальний набір даних;
- алгоритми стохастичного машинного навчання використовують випадковість під час навчання, забезпечуючи навчання іншої моделі кожного запуску.

Відмінності в середовищі розробки, такі як версії програмного забезпечення та тип ЦП, можуть спричинити відмінності помилок округлення в прогнозах та оцінках моделі.

Допоможіть, я отримую інші результати!?

- відмінності, спричинені даними навчання;
- відмінності, спричинені алгоритмом навчання;
- відмінності, спричинені процедурою оцінювання;
- відмінності, спричинені платформою.

Не панікуйте. Алгоритми чи моделі машинного навчання можуть давати різні результати.

Це не твоя помилка. Насправді це часто функція, а не помилка.

Ми чітко уточнимо та пояснимо проблему, яка у вас виникла.

Спочатку давайте розберемося з основами.

У прикладному машинному навчанні ми запускаємо «алгоритм» машинного навчання на наборі даних, щоб отримати «модель машинного навчання». Потім модель можна оцінити на основі даних, які не використовуються під час навчання, або використовувати для прогнозування нових даних, які також не можна побачити під час навчання.

Алгоритм. Процедура виконується на даних, що призводить до моделі (наприклад, навчання або навчання).

Модель: структура даних і коефіцієнти, які використовуються для прогнозування даних.

Контрольоване машинне навчання означає, що у нас є приклади (рядки) із вхідними та вихідними змінними (стовпцями). Ми не можемо написати код для прогнозування вихідних даних на вхідних даних, тому що це занадто складно, тому ми використовуємо алгоритми машинного навчання, щоб навчитися передбачати вихідні дані на основі вхідних даних на історичних прикладах.

Це називається апроксимацією функції, і ми вивчаємо або шукаємо функцію, яка відображає вхідні дані та вихідні дані для нашої конкретної задачі передбачення таким чином, щоб вона мала навички, що означає, що продуктивність відображення краща, ніж випадкова, і в ідеалі краще, ніж усі інші алгоритми та конфігурації алгоритмів, які ми спробували.

Навчання під керівництвом: автоматично вивчайте функцію відображення з прикладів вхідних даних до прикладів вихідних даних.

У цьому сенсі модель машинного навчання — це програма, яку ми маємо намір використовувати для якогось проекту чи програми; Так сталося, що програму вивчали на прикладах (з використанням алгоритму), а не в явному вигляді за допомогою операторів if тощо. Це різновид автоматичного програмування.

Модель машинного навчання: «програма», яка автоматично вивчається на основі історичних даних.

На відміну від програмування, до якого ми звикли, програми можуть бути не зовсім детермінованими.

Моделі машинного навчання можуть відрізнятися під час кожного навчання. У свою чергу, моделі можуть робити різні прогнози, а при оцінці можуть мати різний рівень помилки або точності.

Є принаймні чотири випадки, коли ви отримаєте різні результати; вони є:

- різні результати через відмінності в даних навчання;
- різні результати через стохастичні алгоритми навчання;
- різні результати через процедури стохастичного оцінювання;
- різні результати через різницю в платформі.

Розглянемо кожен по черзі докладніше.

Я пропустив можливу причину різниці в результатах? Дайте мені знати в коментарях нижче.

2. Відмінності, спричинені даними навчання

Ви отримаєте різні результати, якщо запустите один і той самий алгоритм на різних даних.

Це називається дисперсією алгоритму машинного навчання. Можливо, ви чули про це в контексті компромісу зміщення та дисперсії.

Дисперсія є мірою того, наскільки чутливий алгоритм до конкретних даних, які використовуються під час навчання.

Дисперсія: наскільки чутливий алгоритм до конкретних даних, які використовуються під час навчання.

Більш чутливий алгоритм має більшу дисперсію, що призведе до більшої різниці в моделі, і, в свою чергу, до прогнозів і оцінки моделі. І навпаки, менш чутливий алгоритм має меншу дисперсію і призведе до меншої різниці в отриманій моделі з різними навчальними даними і, у свою чергу, до меншої різниці в результатах прогнозів та оцінки моделі.

Висока дисперсія: Алгоритм більш чутливий до конкретних даних, які використовуються під час навчання.

Низька дисперсія: Алгоритм менш чутливий до конкретних даних, які використовуються під час навчання.

Усі корисні алгоритми машинного навчання матимуть певну дисперсію, а деякі з найбільш ефективних алгоритмів матимуть високу дисперсію.

Алгоритми з високою дисперсією часто вимагають більше навчальних даних, ніж алгоритми з меншою дисперсією. Це інтуїтивно зрозуміло, якщо ми розглянемо модель, яка апроксимує функцію відображення на входах і виходах і закон великих чисел.

Тим не менш, коли ви навчаєте алгоритм машинного навчання на різних навчальних даних, ви отримаєте іншу модель з різною поведінкою. Це означає, що різні навчальні дані дадуть моделі, які роблять різні прогнози та мають різну оцінку продуктивності (наприклад, помилка або точність).

Величина різниці в результатах буде пов’язана з тим, наскільки різні навчальні дані для кожної моделі, а також від дисперсії конкретної моделі та конфігурації моделі, які ви вибрали.

Що я повинен зробити? Часто можна зменшити дисперсію моделі, змінивши гіперпараметр алгоритму.

Наприклад, k в k-найближчих сусідах контролює дисперсію алгоритму, де малі значення, такі як k=1, призводять до високої дисперсії, а великі значення, наприклад k=21, призводять до низької дисперсії.

Ви можете зменшити дисперсію, змінивши алгоритм. Наприклад, простіші алгоритми, такі як лінійна регресія та логістична регресія, мають меншу дисперсію, ніж інші типи алгоритмів.

Ви також можете зменшити дисперсію за допомогою алгоритму високої дисперсії, збільшивши розмір навчального набору даних, тобто вам може знадобитися зібрати більше даних.

3. Відмінності, викликані алгоритмом навчання

Ви можете отримати різні результати, запустивши той самий алгоритм на тих самих даних через характер алгоритму навчання.

Це найімовірніша причина, чому ви читаєте цей розділ.

Ви запускаєте той самий код на одному наборі даних і отримуєте модель, яка робить різні прогнози або щоразу має різну продуктивність, і ви думаєте, що це помилка чи щось таке. Чи правий я?

Це не помилка, це особливість. Деякі алгоритми машинного навчання детерміновані. Так само, як програмування, до якого ви звикли. Це означає, що коли алгоритму дається той самий набір даних, він щоразу вивчає ту саму модель. Прикладом є лінійна регресія або алгоритм логістичної регресії.

Деякі алгоритми не є детермінованими; натомість вони стохастичні. Це означає, що їхня поведінка містить елементи випадковості.

Стохастичний не означає випадковий. Алгоритми стохастичного машинного навчання не вивчають випадкову модель. Вони вивчають модель на основі наданих вами історичних даних. Натомість конкретні невеликі рішення, прийняті алгоритмом під час процесу навчання, можуть змінюватися випадковим чином.

Вплив полягає в тому, що кожен раз, коли алгоритм стохастичного машинного навчання запускається на одних і тих же даних, він вивчає дещо іншу модель. У свою чергу, модель може робити дещо інші прогнози, а при оцінці з використанням помилки або точності може мати дещо іншу продуктивність.

Додавання випадковості до деяких рішень, прийнятих алгоритмом, може підвищити продуктивність у складних задачах. Вивчення функції відображення контролюваного навчання з обмеженою вибіркою даних з області є дуже важкою проблемою.

Пошук хорошої або найкращої функції відображення для набору даних є типом проблеми пошуку. Ми тестуємо різні алгоритми та конфігурації алгоритмів, які визначають форму простору пошуку та дають нам відправну точку в просторі

пошуку. Потім ми запускаємо алгоритми, які потім переміщують простір пошуку до однієї моделі.

Додавання випадковості може допомогти уникнути хороших рішень і допомогти знайти дійсно хороші та чудові рішення в просторі пошуку. Вони дозволяють моделі уникнути локального оптимуму або оманливого локального оптимуму, де алгоритм навчання може отримати таке, і допомагають знайти кращі рішення, навіть глобальний оптимум.

Прикладом алгоритму, який використовує випадковість під час навчання, є нейронна мережа. Він використовує випадковість двома способами:

Випадкові початкові ваги (коефіцієнти моделі).

Випадкове перемішування зразків кожної епохи.

Нейронні мережі (глибоке навчання) — це стохастичний алгоритм машинного навчання. Випадкові початкові вагові коефіцієнти дозволяють моделі спробувати навчатися з іншої початкової точки в просторі пошуку кожного алгоритму і дозволяють алгоритму навчання «порушити симетрію» під час навчання. Випадкове перемішування прикладів під час навчання гарантує, що кожна оцінка градієнта та оновлення ваги дещо відрізняються.

Іншим прикладом є стохастичні алгоритми машинного навчання ансамблю, наприклад, пакетування.

Випадковість використовується в процедурі вибірки навчального набору даних, що забезпечує підготовку іншого дерева рішень для кожного члена ансамблю. В ансамблевому навчанні це називається ансамблевим розмаїттям і є підходом до моделювання незалежних прогнозів з одного набору навчальних даних.

Що я повинен зробити? Випадковість, яку використовують алгоритми навчання, можна контролювати.

Наприклад, ви встановлюєте початкове значення, яке використовується генератором псевдовипадкових чисел, щоб гарантувати, що кожен раз під час виконання алгоритму він отримує ту саму випадковість.

Це може бути хорошим підходом до розділу, але не найкращим підходом на практиці. Це призводить до таких питань, як:

Яке найкраще насіння для генератора псевдовипадкових чисел?

Немає найкращого насіння для стохастичного алгоритму машинного навчання. Ви боретеся з природою алгоритму, змушуючи стохастичне навчання бути детермінованим.

Ви можете переконатися, що остаточна модель підходить, використовуючи фіксований початковий код, щоб забезпечити створення тієї ж моделі з тих самих даних перед використанням у виробництві перед будь-яким тестуванням системи перед розгортанням. Проте, як тільки набір навчальних даних зміниться, модель зміниться.

Кращий підхід — прийняти стохастичну природу алгоритмів машинного навчання.

Врахуйте, що для вашого набору даних не існує єдиної моделі. Натомість існує стохастичний процес (конвеєр алгоритму), який може генерувати моделі для вашої проблеми.

Використовуйте випадковість у машинному навчанні.

Потім ви можете підсумувати продуктивність цих моделей — конвеєра алгоритму — як розподіл із середньою очікуваною помилкою чи точністю та стандартним відхиленням.

Потім ви можете забезпечити досягнення середньої продуктивності моделей, встановивши кілька остаточних моделей у свій набір даних і усереднюючи їх прогнози, коли вам потрібно зробити прогноз на основі нових даних.

Як зменшити дисперсію в остаточній моделі машинного навчання

4. Відмінності, спричинені процедурою оцінювання

Завдяки процедурі оцінювання ви можете отримати різні результати під час виконання одного алгоритму з однаковими даними.

Дві найпоширеніші процедури оцінки — це розділення тесту на поїзд і перехресна перевірка k-кратів.

Розподіл тесту на поїздку включає випадкове призначення рядків, які будуть використовуватися для навчання моделі або оцінки моделі, щоб вона відповідала попередньо визначеному розміру набору або тестового набору.

Процедура перехресної перевірки k-кратів включає поділ набору даних на k розділів, що не перекриваються, і використання однієї складки як тестового набору, а всіх інших складок як навчального набору. Модель встановлюється на тренувальний набір і оцінюється на складці утримування, і цей процес повторюється k разів, надаючи кожному згину можливість використовувати його як складку утримання.

Обидві ці процедури оцінки моделі є стохастичними.

Знову ж таки, це не означає, що вони випадкові; це означає, що невеликі рішення, прийняті в процесі, передбачають випадковість. Зокрема, вибір рядків, які призначаються даній підмножині даних.

Таке використання випадковості є особливістю, а не помилкою.

Використання випадковості в цьому випадку дозволяє повторній вибірці наблизити оцінку продуктивності моделі, яка не залежить від конкретної вибірки даних, отриманої з області. Це наближення є упередженим, оскільки ми маємо лише невелику вибірку даних для роботи, а не повний набір можливих спостережень.

Оцінки продуктивності дають уявлення про очікувані або середні можливості моделі під час прогнозування в області даних, які не відображаються під час навчання. Незалежно від конкретних рядків даних, які використовуються для навчання або тестування моделі, принаймні в ідеалі.

Таким чином, кожна оцінка детермінованого алгоритму машинного навчання, як-от лінійна регресія або логістична регресія, може дати різну оцінку помилки або точності.

Що я повинен зробити?

Рішення в цьому випадку дуже схоже на випадок для алгоритмів стохастичного навчання.

Початок для генератора псевдовипадкових чисел може бути зафіксовано, або випадковість процедури може бути охоплена.

На відміну від алгоритмів стохастичного навчання, обидва рішення цілком розумні.

Якщо велика кількість алгоритмів машинного навчання та конфігурацій алгоритмів систематично оцінюється в задачі прогнозного моделювання, може бути гарною ідеєю виправити випадкове початкове значення процедури оцінки. Підійде будь-яке значення.

Ідея полягає в тому, що кожне рішення-кандидат (кожен алгоритм чи конфігурація) буде оцінюватися однаково. Це забезпечує порівняння яблук із яблуками. Це також дозволяє використовувати парні тести статистичних гіпотез пізніше, якщо необхідно, щоб перевірити, чи є відмінності між алгоритмами статистично значущими.

Використання випадковості також може бути доречним. Це передбачає багаторазове повторення процедури оцінювання та звітування підсумків розподілу показників ефективності, таких як середнє значення та стандартне відхилення.

Можливо, найменш упередженим підходом до повторного оцінювання було б використання багаторазової перехресної перевірки k-кратів, наприклад, три повтори з 10 згинами (3×10), що є звичайними, або п’ять повторів з двома згинами (5×2), що зазвичай використовується під час порівняння алгоритмів із перевірками статистичних гіпотез.

Щоб ознайомитися з використанням тестів статистичних гіпотез для порівняння алгоритмів, див. розділ:

Тести статистичної значущості для порівняння алгоритмів машинного навчання

Розділ про порівняння середньої продуктивності алгоритму з перевіркою гіпотези див. у розділі:

Перевірка гіпотези для порівняння алгоритмів машинного навчання

5. Відмінності, спричинені платформою

Ви можете отримати різні результати, запустивши один і той самий алгоритм на тих самих даних на різних комп'ютерах.

Це може статися, навіть якщо ви фіксуєте початкове значення випадкових чисел, щоб усунути стохастичний характер алгоритму навчання та процедури оцінювання.

Причиною в цьому випадку є платформа або середовище розробки, що використовується для запуску прикладу, і результати часто незначно відрізняються, але не завжди.

Це включає:

- відмінності в архітектурі системи, наприклад, CPU або GPU;
- відмінності в операційній системі, наприклад MacOS або Linux;
- відмінності в базових математичних бібліотеках, наприклад, LAPACK або BLAS.
- відмінності у версії Python, наприклад, 3.6 або 3.7;
- відмінності у версії бібліотеки, наприклад, scikit-learn 0,22 або 0,23.

Алгоритми машинного навчання є різновидом числових обчислень.

Це означає, що вони зазвичай включають багато математики зі значеннями з плаваючою комою. Відмінності в таких аспектах, як архітектура та операційна система, можуть призвести до відмінностей у круглих помилках, які можуть посилюватися з кількістю виконаних обчислень, щоб дати дуже різні результати.

Крім того, відмінності у версіях бібліотек можуть означати виправлення помилок і зміну функціональності, що також може призвести до різних результатів.

Крім того, це також пояснює, чому ви отримаєте різні результати для одного алгоритму на одній машині, реалізованого різними мовами, такими як R і Python. Невеликі відмінності в реалізації та/або відмінності в базових математичних бібліотеках, що використовуються, призведуть до відмінностей в результуючій моделі та прогнозах, зроблених цією моделлю.

Що я повинен зробити?

Це не означає, що саму платформу можна розглядати як гіперпараметр і налаштовувати на задачу прогнозного моделювання.

Натомість це означає, що платформа є важливим фактором під час оцінки алгоритмів машинного навчання і має бути зафіксованою або повністю описаною, щоб забезпечити повну відтворюваність під час переходу від розробки до виробництва або звітування про ефективність в академічних дослідженнях.

Одним із підходів може бути використання віртуалізації, наприклад, докер або екземпляр віртуальної машини, щоб забезпечити постійне середовище, якщо повна відтворюваність є критичною для проекту.

Чесно кажучи, на практиці ефект часто дуже незначний (принаймні, з огляду на мій обмежений досвід), якщо основні версії програмного забезпечення є гарними або досить близькими.

2.2 Набір даних (Dataset)

Для експериментування поточного методу було використано два набори даних. Набір даних 1 [16] складається з 1376 зображень, на яких 690 зображень із людьми в масках, а решта 686 зображень із людьми, які не носять масок. Рис. 1

переважно містить позу переднього обличчя з одним обличчям у кадрі та з однотипною маскою лише білого кольору.

Рисунок 2.1 - Зразки з набору даних 1, включаючи обличчя без масок і з масками

Набір даних 2 від Kaggle [17] складається з 853 зображень, і його обличчя уточнюються або за допомогою маски, або без маски. На рис. 2 деякі колекції облич включають поворот голови, нахил і нахил із кількома обличчями в кадрі, а також різними типами масок різних кольорів.

Рисунок 2.2 - Зразки з набору даних 2, включаючи обличчя без масок і з масками

2.3 Стек технологій

TensorFlow, інтерфейс для вираження алгоритмів машинного навчання, використовується для впровадження систем машинного навчання у виробництво в багатьох областях інформатики, включаючи аналіз настроїв, розпізнавання голосу, вилучення географічної інформації, комп'ютерний зір, узагальнення тексту, пошук інформації, обчислювальне виявлення ліків. та виявлення дефектів для продовження досліджень [18]. У запропонованій моделі вся архітектура Sequential CNN (складається з кількох шарів) використовує TensorFlow на сервері. Він також використовується для зміни форми даних (зображення) при обробці даних.

Keras дає фундаментальні відображення та будівельні блоки для створення та транспортування композицій ML з високою швидкістю ітерації. Він у повній

мірі використовує масштабованість і міжплатформні можливості TensorFlow. Основними структурами даних Keras є шари та моделі [19]. Усі шари, використані в моделі CNN, реалізовані за допомогою Keras. Поряд з перетворенням вектора класів у двійкову матрицю класів при обробці даних, це допомагає скласти загальну модель.

OpenCV (Open Source Computer Vision Library), бібліотека комп'ютерного бачення та програмного забезпечення ML з відкритим вихідним кодом, використовується для розрізнення та розпізнавання облич, розпізнавання об'єктів, групування рухів у записах, відстеження прогресивних модулів, слідування жестам очей, відстеження дій камери, видалення червоного очі на знімках, зроблених із використанням спалаху, знаходять порівняльні зображення з бази даних зображень, сприймають ландшафт і встановлюють маркери, щоб накласти його на підвищену реальність тощо [20]. Запропонований метод використовує ці особливості OpenCV для зміни розміру та перетворення кольору даних зображень.

Гарі Бредскі винайшов OpenCV у 1999 році, і незабаром у 2000 році вийшов перший випуск. Ця бібліотека заснована на оптимізованому C/C++ і підтримує Java та Python разом із C++ через інтерфейс. Бібліотека має понад 2500 оптимізованих алгоритмів, включаючи велику колекцію алгоритмів комп'ютерного зору та машинного навчання, як класичних, так і найсучасніших. За допомогою OpenCV стає легко виконувати складні завдання, такі як ідентифікувати та розпізнавати обличчя, ідентифікувати об'єкти. , класифікувати дії людини у відео, відстежувати рухи камери, відстежувати рухомі об'єкти, витягувати 3D-моделі об'єктів, генерувати тривимірні хмари точок із стереокамер, з'єднувати зображення, щоб створити цілу сцену із зображенням високої роздільної здатності та багато іншого.

Python є зручною мовою, з якою легко працювати, але ця перевага пов'язана з вартістю швидкості, оскільки Python повільніше для мов, таких як C або C++. Тому ми розширюємо Python за допомогою C/C++, що дозволяє нам писати

інтенсивний обчислювальний код. в C/C++ і створювати обгортки Python, які можна використовувати як модулі Python. Завдяки цьому код буде швидким, оскільки він написаний оригінальним кодом C/C++ (оскільки це фактичний код C++, який працює у фоновому режимі), а також його легше кодувати на Python, ніж на C/C++. OpenCV-Python — це обгортка Python для оригінальної реалізації OpenCV C++.

2.4. Метод виявлення обличчя в масці та без маски

Запропонований метод складається з каскадного класифікатора та попередньо навченої CNN, яка містить два шари 2D згортки, з'єднані з шарами щільних нейронів. Алгоритм визначення маски виглядає наступним чином:

Алгоритм виявлення маски для обличчя–додаток Б

Обробка даних – це перетворення даних у придатну для використання та бажану форму. Це перетворення або «обробка» виконується з використанням попередньо визначеної послідовності операцій вручну або автоматично. Більшість обробки здійснюється за допомогою комп'ютерів і, таким чином, відбувається автоматично. Вихідні або «оброблені» дані можна отримати в різних формах. Прикладами цих форм є зображення, графік, таблиця, векторний файл, аудіо, діаграми або будь-який інший бажаний формат. Отримана форма залежить від використовуваного програмного забезпечення або методу обробки даних. Коли це буде виконано, це називається автоматичною обробкою даних.

Обробка даних, по суті, синхронізує всі дані, введені в програмне забезпечення, щоб відфільтрувати з нього найбільш корисну інформацію. Це дуже важливе завдання для будь-якої компанії, оскільки воно допомагає їм отримати найбільш релевантний контент для подальшого використання. Кожен важливий сектор, будь то банки, школи, коледжі чи великі компанії, майже всі потребують

такої обробки даних. Ця обробка виконується для того, щоб зберегти найбільш витончену інформацію в їхніх системах для подальшого використання. Обробка вручну займає дуже багато часу і вимагає залучення занадто великої кількості людей для цього. Це справді неможливе завдання, коли у вас є масові дані. Сьогодні люди в галузі залежать від потужних та ефективних програмних інструментів, які допомагають обробляти всі ці дані. Це допомагає їм досягти більшої точності та підвищити ефективність. При належній обробці даних можна сортувати все більше і більше інформації. Це допомагає отримати більш чітке уявлення про матерію та краще її зрозуміти. Це може призвести до кращої продуктивності та більшого прибутку для різних сфер бізнесу.

Реальні програми обробки даних. За допомогою належних алгоритмів і протоколів безпеки можна гарантувати, що введені дані та оброблена інформація будуть безпечними та надійно збережені без несанкціонованого доступу або змін. З належним чином обробленими даними дослідники можуть писати наукові матеріали та використовувати їх у навчальних цілях. Те ж саме можна застосувати і для оцінки економічних і подібних сфер і факторів. У галузі охорони здоров’я оброблені дані можна використовувати для швидшого пошуку інформації та навіть для порятунку життя. Крім того, відомості про хворобу та записи методів лікування можуть зменшити витрати часу на пошук рішень і допомогти зменшити страждання пацієнтів.

Обробка даних, щоб упорядкувати їх за типом та інформацією, може заощадити багато місця, зайнятого даними, які не впорядковані й не зберігаються випадково. Оброблені дані також можуть допомогти всім співробітникам і працівникам легко їх зрозуміти. Вони можуть реалізувати це в роботі, яка в іншому випадку може зайняти більше часу і призвести до зниження продуктивності. Це може зашкодити інтересам підприємства чи організації.

Фокус обробки даних. Більшість компаній і сфер потребують даних для надання якісних послуг. Набір даних про зібрані дані та їх наслідки є дуже

важливим аспектом управління ними та забезпечення статистичної достовірності. Це особливо важливо для служб, пов’язаних із фінансовими технологіями. Це пов’язано з тим, що дані трансакцій та деталі платежу мають належним чином зберігатися для легкого доступу клієнтам, а також посадовим особам компанії в разі потреби. Обробка не обмежується комп’ютерами і може виконуватися вручну.

У той час як ручний варіант використовує потужність мозку та інтелект, електронні методи обробки даних можуть заощадити багато часу та забезпечити безперебійний робочий процес та забезпечити дотримання термінів. Точність також вища з електронною обробкою. Один із важливих аспектів цього полягає в тому, щоб переконатися, що сформовані ідеї зберігаються для майбутнього та спільного використання, щоб заощадити обчислювальну потужність та час.

Основи обробки даних та спосіб обробки даних. Обробка даних потрібна для будь-якої діяльності, яка вимагає збору даних. Ці зібрані дані необхідно зберігати, сортувати, обробляти, аналізувати та представляти. Цей повний процес можна розділити на 6 простих початкових етапів, які:

- збір даних;
- зберігання даних;
- сортування даних;
- обробка даних;
- аналіз даних.

Подання даних та висновки. Після збору даних виникає потреба у введенні даних для зберігання даних. Зберігання може здійснюватися у фізичній формі за допомогою паперів, у зошитах або в будь-якій іншій фізичній формі. З появою та зростаючим акцентом на комп’ютерних системах, великих даних і інтелекту даних збір даних є великим, і для змістовного аналізу та представлення необхідно виконати ряд операцій, дані зберігаються в цифровій формі. Переведення

необроблених і оброблених даних у цифрову форму дає змогу користувачеві виконувати велику кількість операцій за короткий час і дозволяє конвертувати їх у різні типи. Таким чином, користувач може вибрати вихід, який найкраще відповідає вимогам.

Це безперервне використання та обробка даних слідує за циклом, який називається циклом обробки даних і циклом обробки інформації. Ці цикли можуть забезпечити миттєвий результат або зайняти час, залежно від потреби обробки даних. Складність у цій галузі зростає, що створює потребу в передових техніках.

Зберігання даних супроводжується сортуванням і фільтрацією. На цей етап суттєво впливає формат, у якому зберігаються дані. Це також залежить від використовуваного програмного забезпечення. Загальні денні та нескладні дані можуть зберігатися у вигляді текстових файлів, таблиць або їх комбінації в Microsoft Excel або подібному програмному забезпеченні. Оскільки завдання стає складним, що вимагає виконання конкретних і спеціалізованих операцій. Вони вимагають різних інструментів обробки даних і програмного забезпечення, яке призначене для задоволення особливих потреб.

Зберігання, сортування, фільтрація та обробка даних може здійснюватися за допомогою одного програмного забезпечення або комбінації програмного забезпечення, залежно від можливостей і потреб. Така обробка, що виконується таким чином програмним забезпеченням, виконується відповідно до попередньо визначеного набору операцій. Більшість сучасного програмного забезпечення дозволяє користувачам виконувати різні дії на основі аналізу або дослідження, яке необхідно провести. Він забезпечує вихідний файл у різних форматах.

Різні типи вихідних файлів, отриманих як «оброблені» дані

Звичайний текстовий файл – це найпростіша форма або оброблені дані. Більшість із цих файлів є зручними для читання і легко сприйняті. Ці файли дуже незначні або не обробляються. Вони експортуються як файли блокнота або WordPad.

Таблиця/електронна таблиця – цей формат файлу найбільше підходить для числових даних. Наявність цифр у рядках і стовпцях дозволяє користувачеві виконувати різні операції. Наприклад, фільтрація та сортування в порядку зростання/спадання, щоб полегшити розуміння та використання. Під час використання цього файлу можна застосувати різні математичні операції.

Діаграми та графіки – можливість отримувати вихідні дані у вигляді діаграм і графіків зручна і тепер формує стандартні функції більшості програмного забезпечення. Ця опція вигідна при роботі з числовими значеннями, що відображають тенденції та зростання/зниження. Існує велика кількість діаграм і графіків, які відповідають різноманітним вимогам. Іноді буває ситуація, коли виникає потреба мати параметр, визначений користувачем. Якщо вбудовані діаграми чи графіки недоступні, то стане в нагоді можливість створити власні діаграми, тобто спеціальні діаграми/графіки.

Карти/векторні файли або файли зображень – при роботі з просторовими даними дуже корисна можливість експорту оброблених даних у карти, векторні та графічні файли. Наявність інформації на картах особливо корисна для містобудівників, які працюють з різними типами карт. Файли зображень отримуються під час роботи з графікою і не являють собою будь-який доступний для читання людиною.

Інші формати/необроблені файли – це специфічні для програмного забезпечення формати файлів, які можна використовувати та обробляти спеціалізованим програмним забезпеченням. Ці вихідні файли можуть бути неповноцінним продуктом і потребують подальшої обробки. Таким чином, потрібно буде виконувати дії кілька разів.

Обробка вручну: у цьому методі дані обробляються вручну без використання машини, інструменту або електронного пристрою. Дані обробляються вручну, а всі обчислення та логічні операції над даними виконуються вручну.

Механічна обробка – це здійснюється за допомогою механічного пристрою або дуже простих електронних пристроїв, таких як калькулятор і друкарська машинка. Коли потреба в обробці проста, можна використовувати цей метод.

Електронна обробка – це сучасна техніка обробки даних. Електронна обробка даних є найшвидшим і найкращим доступним методом з найвищою надійністю та точністю. Використовувана технологія є новітньою, оскільки цей метод використовує комп’ютери та використовується в більшості агентств. Використання програмного забезпечення є невід'ємною частиною цього типу. Дані обробляються через комп’ютер; Дані та набір інструкцій надаються комп’ютеру як вхідні дані, і комп’ютер автоматично обробляє дані відповідно до заданого набору інструкцій. Комп’ютер також відомий як електронна машина для обробки даних.

Типи обробки на основі виконаних процесів/кроківІснують різні види обробки даних, деякі з найпопулярніших типів:

- пакетна обробка;
- обробка в режимі реального часу;
- онлайн-обробка;
- багатопроцесорність;
- розподіл часу;
- пов’язано: Методи та типи обробки даних;
- чому це потрібно?.

Працювати з необробленими даними справді важко. Оскільки кожна надана інформація може бути не такою корисною для вас. Вам потрібно відфільтрувати відповідний вміст. Ви не можете кожен раз звертатися до цієї величезної купи необроблених даних і вибирати відповідну інформацію, яку шукаєте. Це зробить вашу роботу більш втомливою і громіздкою. Обробка даних допоможе вам

упорядкувати відфільтрований вміст у однорідну форму, щоб ви могли легко зіставляти ці великі цифри, коли вам це потрібно. Це полегшить вам пошук будь-якої відповідної інформації, а також полегшить вашу роботу.

Це навіть зробить всю цю процедуру більш економічно ефективною. Оскільки впорядкування цих великих цифр у добре структуровані таблиці позбавляє вас від ризику втрати важливої інформації. Крім того, частина інформації відфільтровується, таким чином, вартість збереження цієї невідповідної інформації також зберігається.

Це також полегшує вам зміну та редагування оброблених даних. Вам просто потрібно шукати схожі клітинки та застосовувати те саме правило для всіх клітинок, які ви хочете змінити.

Обробка даних дуже важлива перед тим, як ви почнете робити аналіз даних. Це зменшує ваші витрати на оформлення всіх документів, необхідних для обробки всієї інформації та відфільтрування всього відповідного вмісту вручну. Це підвищує загальну продуктивність будь-якої компанії, оскільки виключає непотрібні кроки, які можуть перешкодити всьому процесу обробки даних. Він автоматично видаляє всі ваші дублікати документів і, таким чином, допомагає звільнити місце для зберігання у вашій системі.

Що робить обробку даних важливою. Сьогодні все більше і більше даних збирається для академічних, наукових досліджень, приватного та особистого використання, інституційного використання, комерційного використання. Ці зібрані дані потрібно зберігати, сортувати, фільтрувати, аналізувати та представляти і навіть вимагати передачі даних, щоб вони були корисними. Цей процес може бути простим або складним залежно від масштабу, в якому здійснюється збір даних, і складності результатів, які необхідно отримати. Час, що витрачається на отримання бажаного результату, залежить від операцій, які необхідно виконати над зібраними даними, і від характеру вихідного файлу, який необхідно отримати. Ця проблема стає більш гострою при роботі з дуже великим

обсягом даних. Наприклад, дані, зібрані транснаціональними компаніями. Вони збирають дані про своїх користувачів, продажі, виробництво,

У таких випадках потреба в обробці стає все більш критичною. У таких випадках в дію вступають інтелектуальний аналіз даних і керування даними, без яких неможливо отримати оптимальні результати. Кожен етап, починаючи від збору даних і закінчуючи представленням, безпосередньо впливає на вихід і корисність оброблених даних. Надання набору даних третім особам має здійснюватися обережно та відповідно до письмової угоди та угоди про надання послуг. Це запобігає крадіжці даних, неправильному використанню та втраті даних.

Який тип даних необхідно обробити. Дані в будь-якій формі та будь-якого типу вимагають обробки більшу частину часу. Ці дані можна класифікувати як особисту інформацію, фінансові операції, податкові кредити, банківські реквізити, обчислювальні дані, зображення та майже все, що ви можете придумати. Обсяг необхідної обробки буде залежати від спеціальної обробки, якої потрібні дані. Згодом це буде залежати від виходу, який вам потрібен. Зі збільшенням попиту та потреби в таких послугах виник конкурентний ринок послуг даних.

Інструменти керування базами даних – ACCESS і BASE тощо – це інструменти, які допомагають нам керувати великою кількістю даних, які в іншому випадку стають надто втомливими, щоб доглядати або звертатися до них, коли нам це потрібно.

Це обліковий запис адміністратора Planning Tank. Під цим обліковим записом публікуються дипломній роботі, написані членами нашої команди, надані запрошеними авторами та інші випадкові матеріали. Щоб отримати будь-які деталі/запити, будь ласка, зв’яжіться з нами з вашим запитанням/запитом/відгуком.

Попередня обробка даних передбачає перетворення даних із заданого формату в набагато більш зручний, бажаний і значущий формат. Вона може бути в

будь-якій формі, як-от таблиці, зображення, відео, графіки тощо. Ця організована інформація вписується в інформаційну модель або композицію та фіксує відносини між різними об’єктами [6]. Запропонований метод має справу з даними зображення та відео за допомогою Numpy та OpenCV.

Візуалізація даних — це процес перетворення абстрактних даних у значущі уявлення з використанням передачі знань і виявлення уявлень за допомогою кодування. Корисно вивчити певну закономірність у наборі даних [7].

Загальна кількість зображень у наборі даних візуалізується в обох категоріях – «з маскою» та «без маски».

Оператор категорії=os.listdir(path_data) категоризує список каталогів у вказаному шляху даних. Категорії змінних тепер виглядають так: ['з маскою', 'без маски']

Потім, щоб знайти кількість міток, нам потрібно розрізнити ці категорії за допомогою labels=[i for i in range(len(categories))]. Він встановлює мітки як: [0, 1]

Тепер кожна категорія зіставляється зі своєю відповідною міткою за допомогою label_dict=dict(zip(categories,labels)), який спочатку повертає ітератор кортежів у вигляді об’єкта zip, де елементи в кожному переданому ітераторі об’єднуються разом. Відображена змінна label_dict виглядає так: {'з маскою': 0, 'без маски': 1}.

Перетворення зображення RGB на сіре зображення. Сучасні системи розпізнавання зображень на основі дескрипторів регулярно працюють над зображеннями у відтінках сірого, не розробляючи метод, який використовується для перетворення кольорів у відтінки сірого. Це пов’язано з тим, що метод від кольору до відтінків сірого не має великого значення при використанні надійних дескрипторів. Введення несуттєвої інформації може збільшити розмір навчальних даних, необхідних для досягнення хорошої продуктивності. Оскільки відтінки сірого раціоналізують алгоритм і зменшують обчислювальні реквізити, він

використовується для вилучення дескрипторів замість миттєвої роботи з кольоровими зображеннями [8].

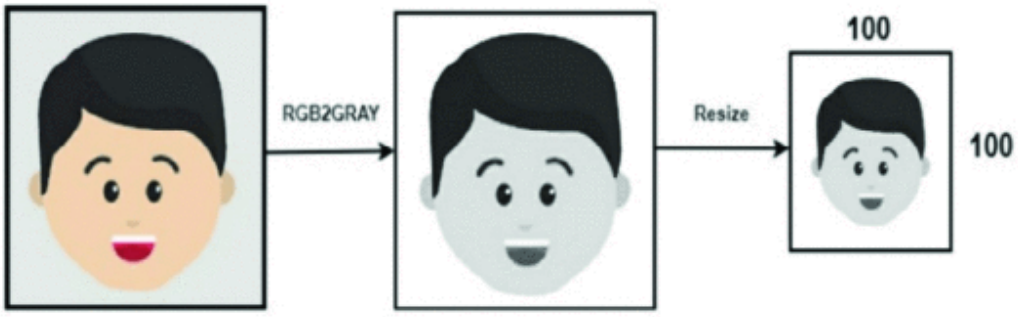

Рисунок. 2.3. Перетворення зображення RGB у зображення у шкалі сірого розміром 100x100

Ми використовуємо функцію cv2.cvtColor(input_image, flag) для зміни колірного простору. Тут прапорець визначає тип перетворення [9]. У цьому випадку для перетворення сірого використовується прапор cv2.COLOR_BGR2GRAY.

Глибокі CNN вимагають вхідного зображення фіксованого розміру. Тому нам потрібен фіксований загальний розмір для всіх зображень у наборі даних. За допомогою cv2.resize() розмір сірого зображення змінюється до 100 x 100.

Зміна зображення.Pviyf hjpvihe d Vanilla Python**.** Ми не будемо писати алгоритм зміни розміру зображень на Python. Чому? Це занадто багато роботи. Алгоритмів обробки зображень багато. Деякі люди присвячують цьому своє життя. Повторна дискретизація — використання одного пікселя в зменшеному зображенні для заміни багатьох навколо нього у вищій роздільній здатності — сама по собі є величезною темою. Якщо ви хочете переконатися в цьому самі, перегляньте Image.py у вихідному коді Pillow, коли ви встановлюєте його за адресою path/to/site-packages/PIL.

Потім є такі оптимізації, як згладжування та зменшення розривів… Це нескінченно. Тут ми будемо стояти на плечах гігантів, тобто довіряти блискучій роботі тих, хто працює в області комп'ютерного зору, і безсоромно закликати нас до вирішення наших проблем.

Якщо вам цікаво дізнатися більше про те, що відбувається за завісою під час обробки зображень, я рекомендую вам більше ознайомитися з темою «машинного зору»! Це, безумовно, процвітаюча сфера, і є більше ніш для вивчення, ніж у однієї людини.

Станьте достатньо хорошими, і є легіон компаній, які готові заплатити високі гроші за ваш досвід комп'ютерного зору. Автомобільне водіння, Інтернет речей, спостереження, ну і так; всі вони в основному покладаються на обробку зображень (зазвичай на Python або C++).

Відмінне місце для початку – перевірка scikit-image. Вхідним сигналом під час віддалення зображення є тривимірний тензор, де кожен канал має помітний унікальний піксель. Усі зображення повинні мати однаковий розмір, що відповідає тензору 3D-об'єктів. Однак ані зображення зазвичай не є співрозширеними, ані відповідні їм тензори ознак [10]. Більшість CNN можуть приймати лише тонко налаштовані зображення. Це породжує декілька проблем під час збору даних та реалізації моделі. Однак переналаштування вхідних зображень перед додаванням їх у мережу може допомогти подолати це обмеження. [11].

Зображення нормалізуються для зближення діапазону пікселів від 0 до 1. Потім вони перетворюються на 4-вимірні масиви за допомогою data=np.reshape(data,(data.shape[0], img_size,img_size,1)), де 1 вказує на Зображення у відтінках сірого. Оскільки кінцевий шар нейронної мережі має 2 виходи – з маскою та без маски, тобто має категоріальне представлення, дані перетворюються на категоріальні мітки.

Навчання моделі. Побудова моделі з використанням архітектури CNN. CNN став провідним місцем у різних задачах комп'ютерного зору [12]. Нинішній метод використовує послідовний CNN.

За шаром першої згортки слідують шари Rectified Linear Unit (ReLU) і MaxPooling. Рівень згортки вивчає 200 фільтрів. Розмір ядра встановлено на 3 x 3, що визначає висоту та ширину вікна 2D згортки. Оскільки модель повинна знати про форму очікуваного введення, перший шар моделі повинен бути наданий інформацією про форму вхідних даних. Наступні шари можуть виконувати інстинктивний розрахунок форми [13]. У цьому випадку input_shape вказується як data.shape[1:], який повертає розміри масиву даних з індексу 1. Заповнення за замовчуванням є «дійсним», де просторові розміри дозволено скорочувати, а вхідний об’єм доповнений відмінним від нуля . Параметр активації для класу Conv2D встановлюється як «relu». Він являє собою приблизно лінійну функцію, яка володіє всіма активами лінійних моделей, які можна легко оптимізувати за допомогою методів градієнтного спуску. З огляду на продуктивність і узагальнення в глибокому навчанні, воно краще порівняно з іншими функціями активації [14]. Max Pooling використовується для зменшення просторових розмірів вихідного обсягу. Pool_size встановлюється на 3 x 3, а отриманий результат має форму (кількість рядків або стовпців): shape_of_output = (input_shape – pool_size + 1) / strides), де strides має значення за замовчуванням (1,1) [15].

Як показано на рис, 4, другий шар згортки має 100 фільтрів, а розмір ядра встановлено на 3 x 3. За ним слідують шари ReLu і MaxPooling. Щоб вставити дані в CNN, довгий вектор введення передається через рівень Flatten, який перетворює матрицю ознак у вектор, який може бути поданий у повністю підключений класифікатор нейронної мережі. Щоб зменшити переобладнання, до моделі додано шар Dropout з 50% ймовірністю встановлення вхідних даних на нуль. Потім додається щільний шар із 64 нейронів з функцією активації ReLu. Останній шар

(Dense) з двома виходами для двох категорій використовує функцію активації Softmax.

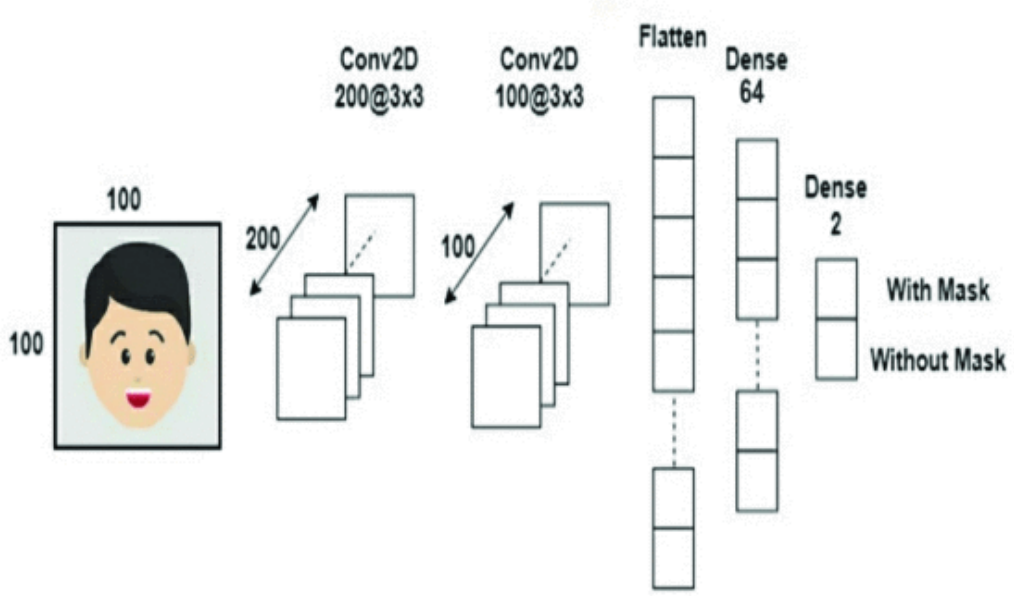

Рисунок. 2.4. Архітектура згорткової нейронної мережі

Процес навчання потрібно спочатку налаштувати за допомогою методу компіляції [13]. Тут використовується оптимізатор «адам». categorical_crossentropy, яка також відома як багатокласові втрати журналу, використовується як функція втрат (ціль, яку модель намагається мінімізувати). Оскільки проблема є проблемою класифікації, метрика встановлюється на «точність».

Розбиття даних і навчання моделі CNN. Після налаштування плану для аналізу даних модель потрібно навчити з використанням певного набору даних, а потім її протестувати з іншим набором даних. Правильна модель і оптимізований train_test_split допомагають отримати точні результати під час прогнозування. Test_size встановлено на 0,1, тобто 90% даних набору даних проходять навчання, а решта 10% йдуть на тестування. Втрата перевірки контролюється за допомогою ModelCheckpoint. Далі зображення в навчальному наборі та тестовому наборі підходять до послідовної моделі. Тут 20% даних навчання використовуються як

дані перевірки. Модель навчається протягом 20 епох (ітерацій), що підтримує компроміс між точністю та ймовірністю переобладнання. На рис. 5 зображено візуальне уявлення пропонованої моделі.

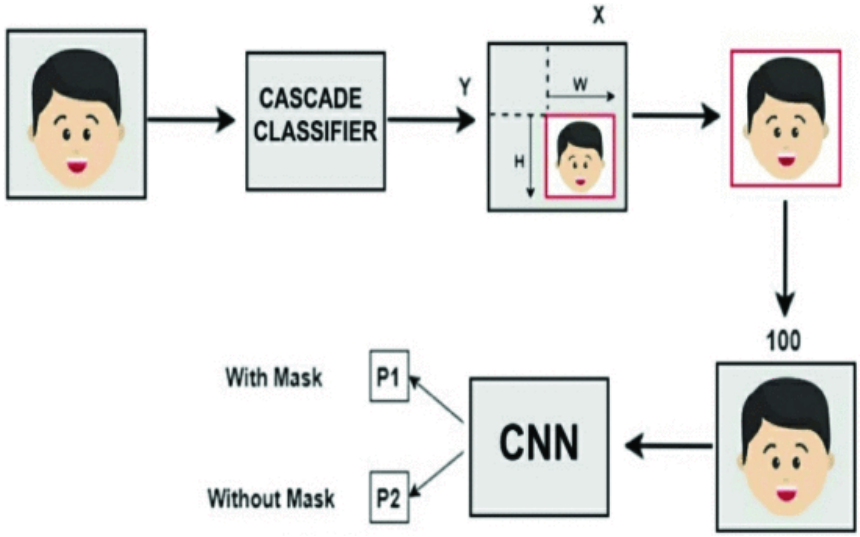

Рисунок 2.5 - Огляд моделі

РОЗДІЛ 3
АНАЛІЗ РЕЗУЛЬТАТІВ НЕЙРОННОЇ МЕРЕЖІ

3.1.Результат роботи нейронної мережі

Модель навчається, перевіряється та тестується на двох наборах даних. Відповідаючи набору даних 1, метод досягає точності до 95,77% (показано на рис. 7). На рис. 6 показано, як ця оптимізована точність зменшує вартість помилки. Набір даних 2 є більш універсальним, ніж набір даних 1, оскільки він має кілька граней у кадрі, а також різні типи масок, які мають різні кольори. Таким чином, модель досягає точності 94,58% у наборі даних 2, як показано на рис. 9. Рис. 8 зображує контраст між навчанням і втратою перевірки, що відповідає набору даних 2. Одна з основних причин досягнення цієї точності полягає в MaxPooling. Це забезпечує елементарну інваріантність трансляції внутрішньому представленню разом із зменшенням кількості параметрів, які модель має засвоїти. Цей процес дискретизації на основі вибірки зменшує вибірку вхідного представлення, що складається з зображення, шляхом зменшення його розмірності. Кількість нейронів має оптимізоване значення 64, що не є надто високим. Значно більша кількість нейронів і фільтрів може призвести до погіршення продуктивності. Оптимізовані значення фільтра та pool_size допомагають відфільтрувати основну частину (обличчя) зображення, щоб правильно виявити існування маски, не викликаючи надмірного підгонки.

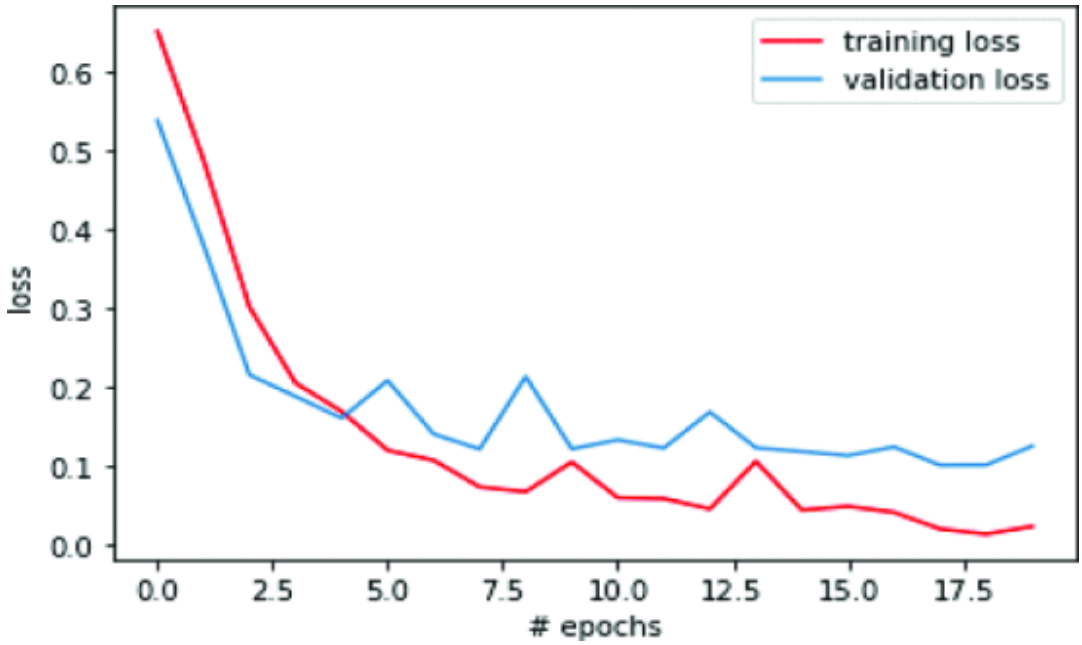

Рисунок2.6 - #епохи проти втрат, що відповідають набору даних 1

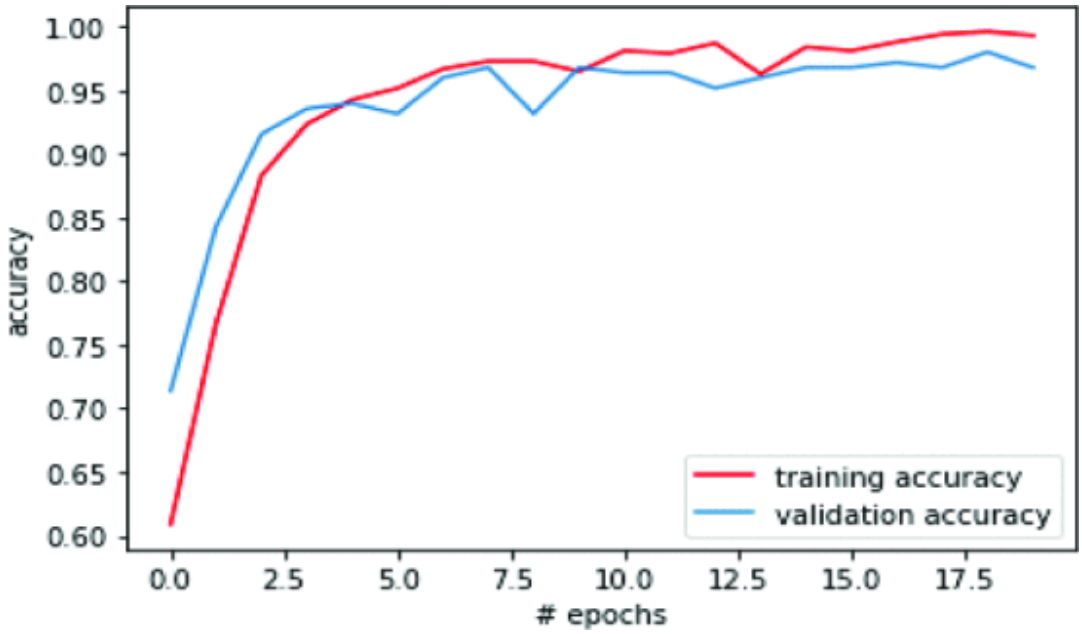

Рисунок 2.7 - #епохи проти точності, що відповідає набору даних 1

Система може ефективно виявляти частково закриті обличчя за допомогою маски, волосся або руки. Він враховує ступінь оклюзії чотирьох областей – носа,

рота, підборіддя та ока, щоб розрізняти анотовану маску чи обличчя, закрите рукою. Тому маска, яка повністю закриває обличчя, включаючи ніс і підборіддя, буде розглядатися моделлю лише як «з маскою».

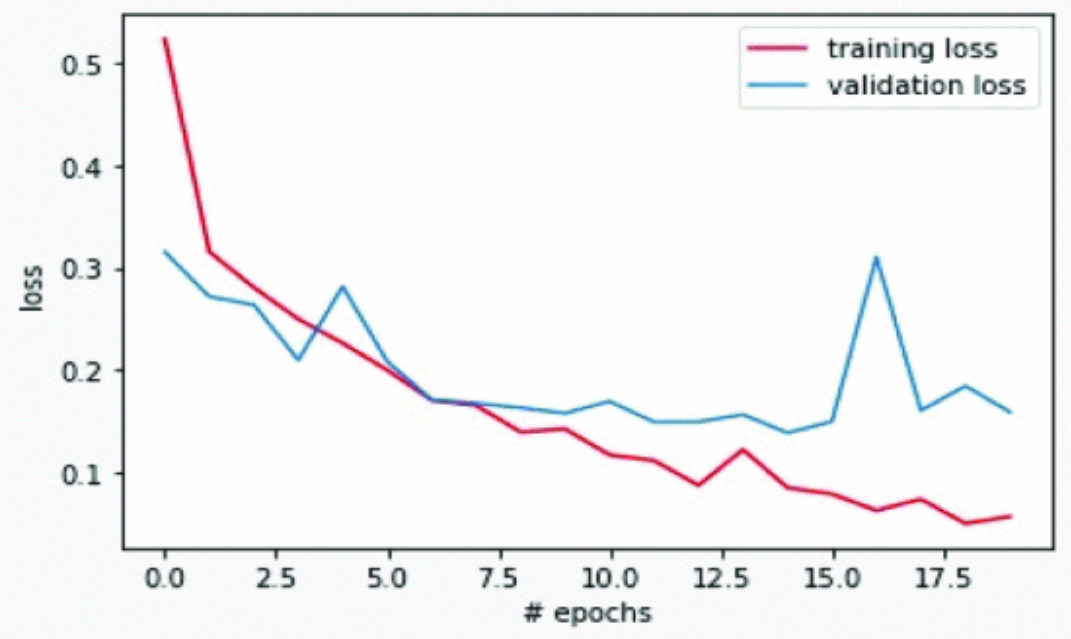

Рисунок 2.8 - #епохи проти втрат, що відповідають набору даних 2

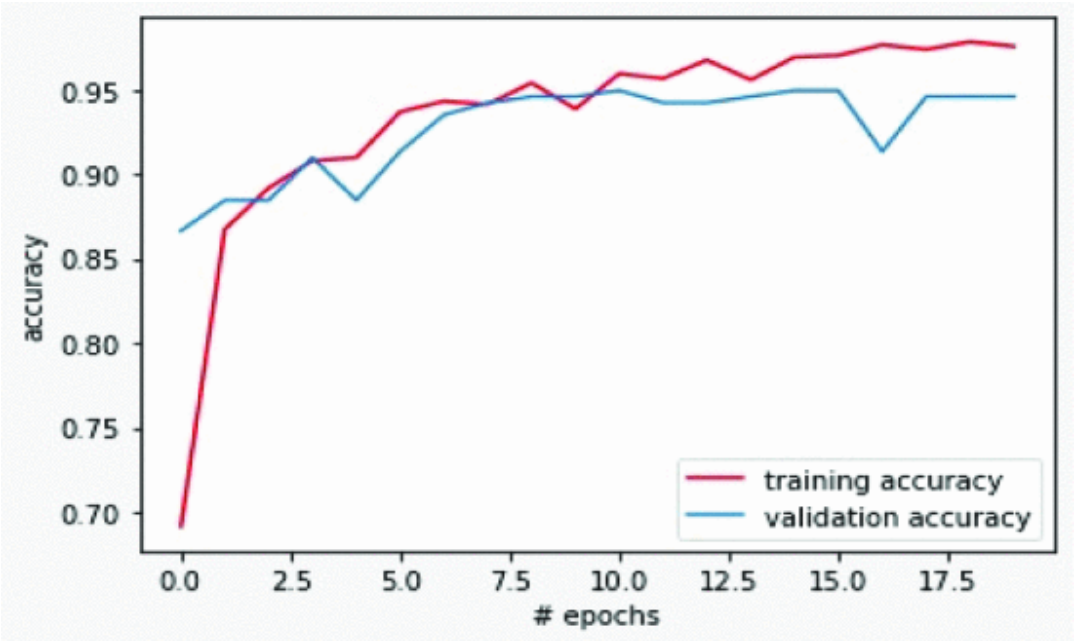

Рисунок 2.9 - #епохи проти точності, що відповідає набору даних 2

Основні проблеми, з якими стикається метод, в основному полягають у різному ракурсі та відсутності чіткості. Нечіткі рухомі обличчя у відеопотоку ускладнюють це. Однак слідування траєкторіям кількох кадрів відео допомагає прийняти краще рішення – «з маскою» чи «без маски».

3.2Тестування роботи нейронної мережі на фото сеті

Щоб протестувати реальну роботу мережі, треба запустити команду яка запускає якесь конкретне зображення. Наприклад розглянемо наступні випадки.

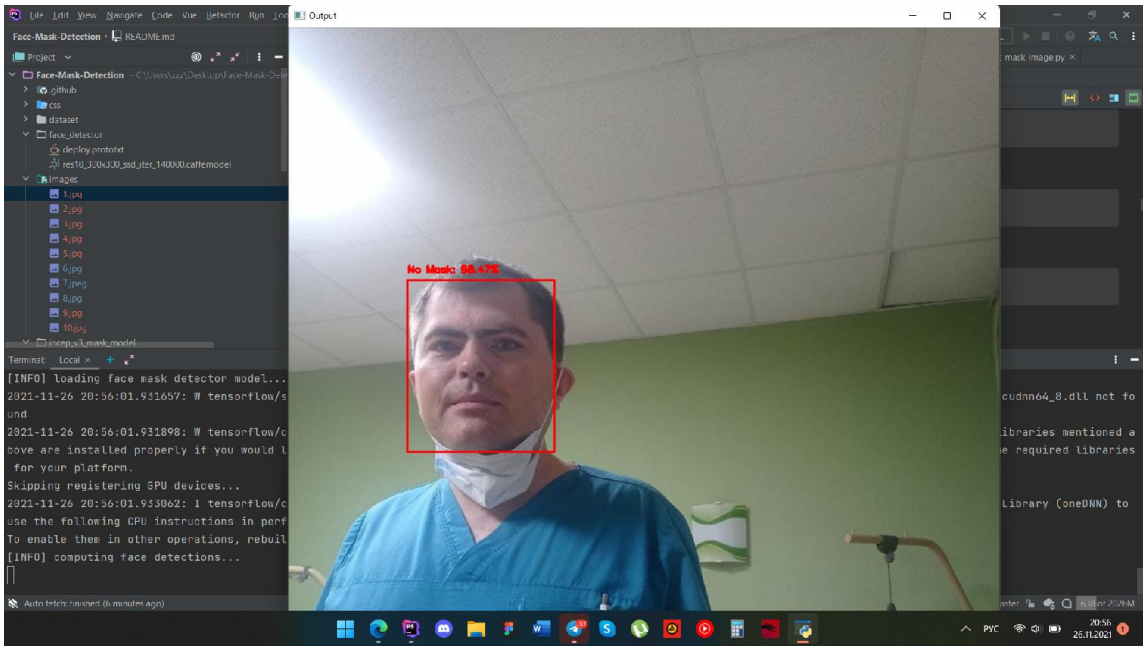

Рисунок 3.1 - Без маски.

На малюнку 3.1 зображений випадок коли на зобрженні є одно обличчя без маски. Звичайна ситуація

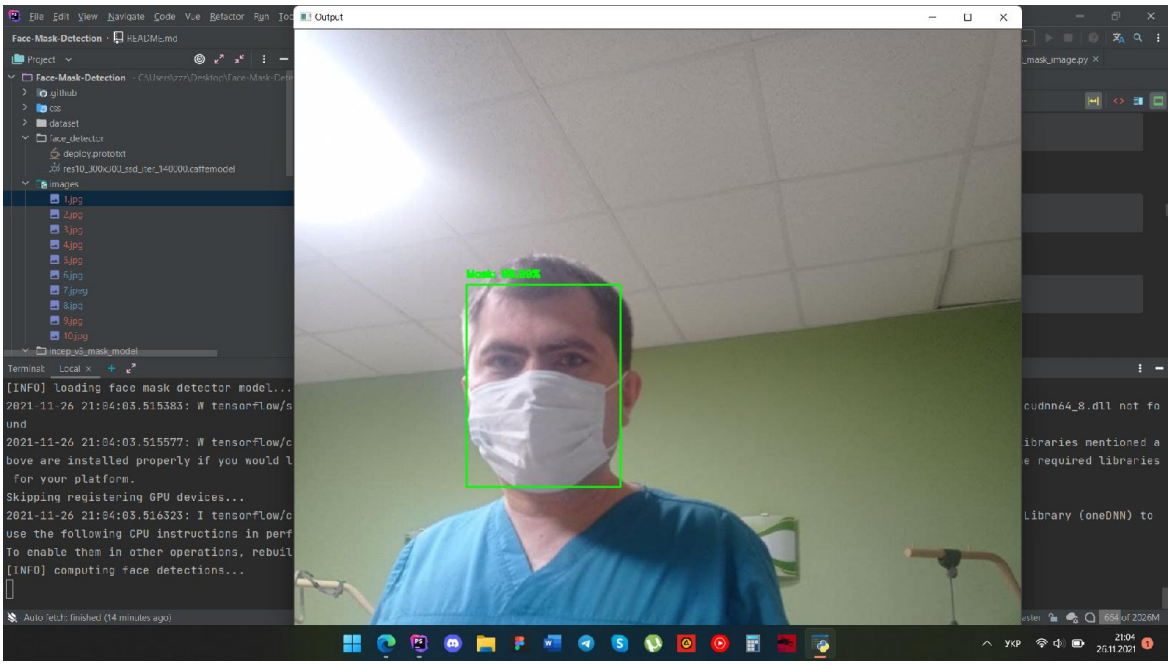

Рисунок 3.2 - Обличчя в масці.

На цьому прикладі зображено результат роботи алгоритму в масці повністю надягненою на ніс.

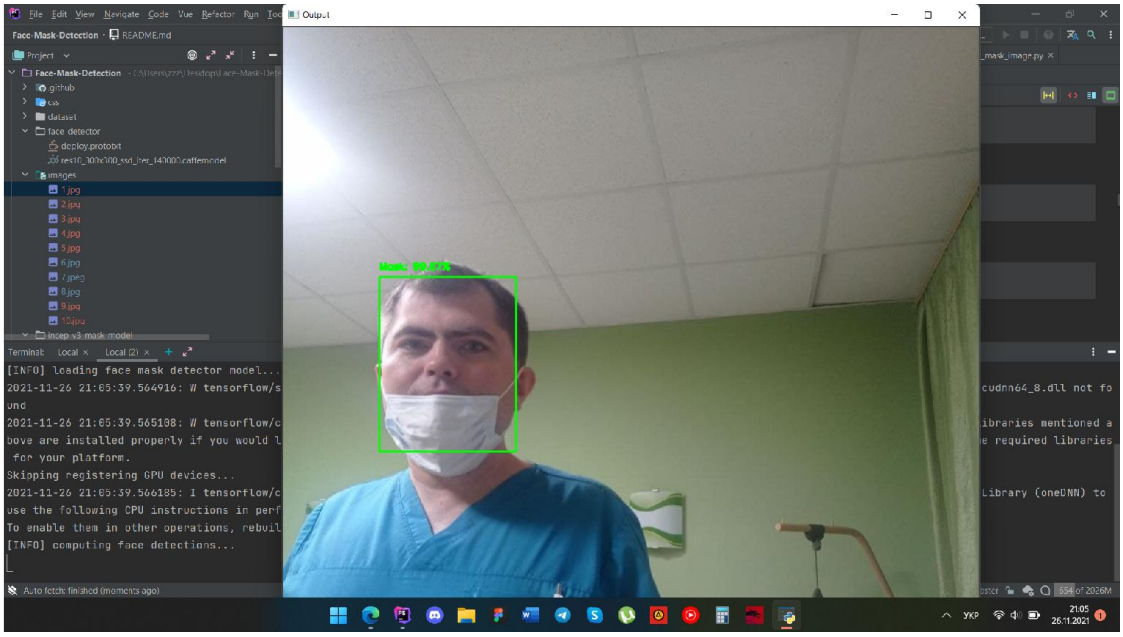

Рисунок 3.3 - Обличння в масці спущеної з носа.

Як видно на цьому прикладі. Маска яка не повнісью одягнута на обличчя, теж являється маскою

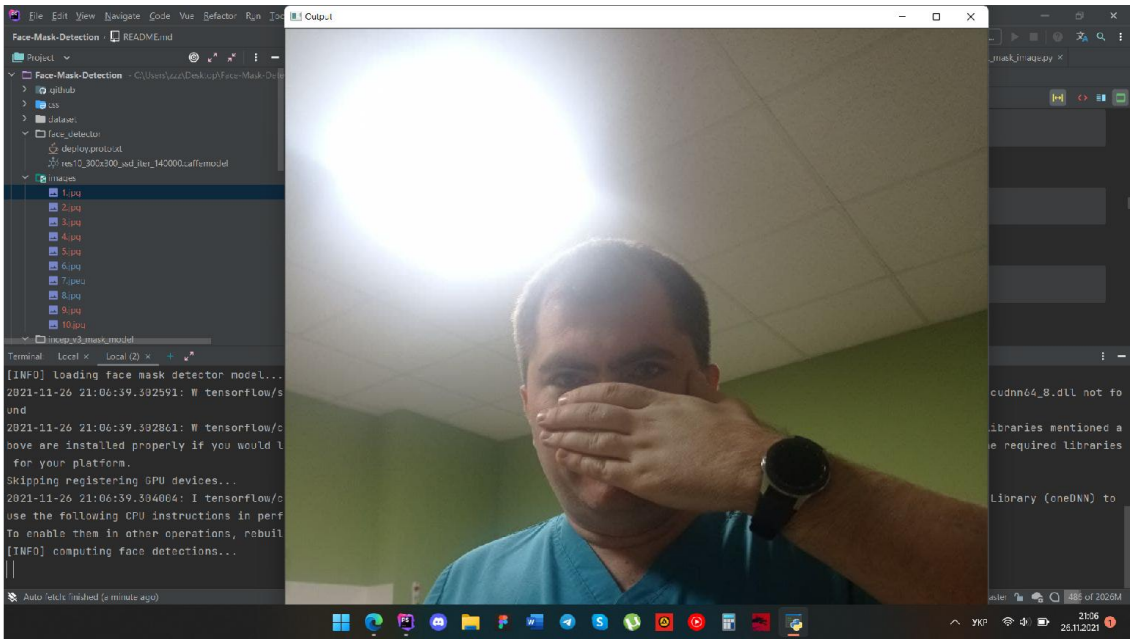

Рисунок 3.4 - Обличча без маски та закрите руккою

В цьому прикладі ми бачемо що алгоритм не може роспознати личе прикрите не маскою. Того що в нас не було задачі роспізнавати такі випадки

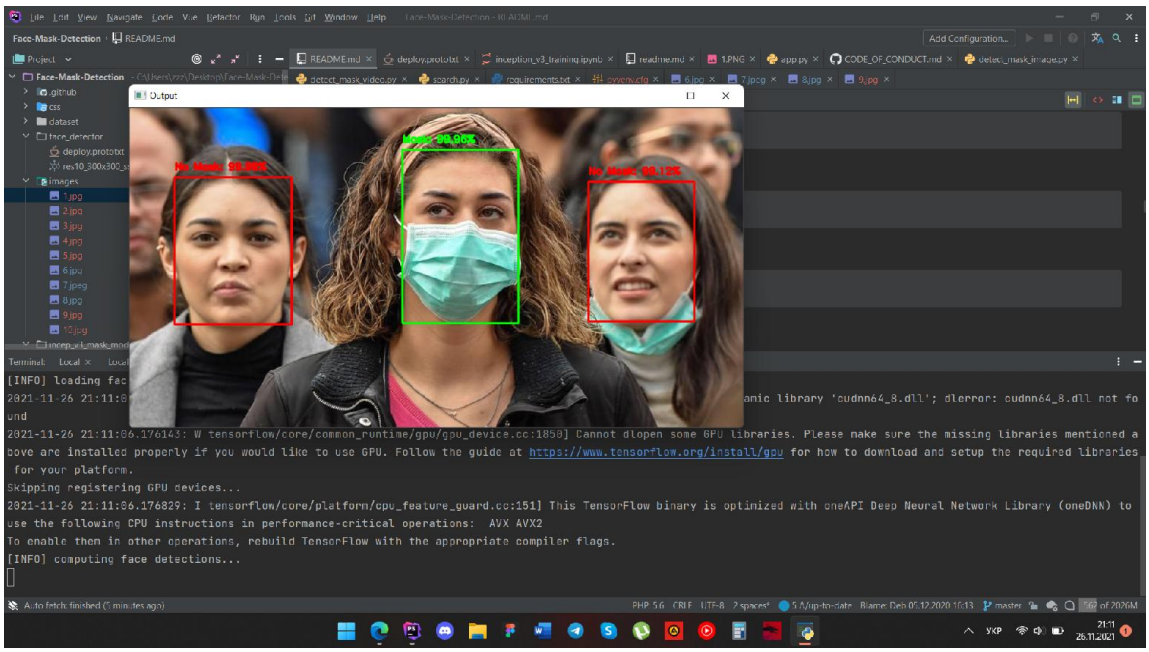

Рисунок 3.5 - Комбінований тип зображення.

На цьому зображенні ми бачимо що алгоритм може роспізнавати декілька облич одночасно та навідь баще що лице на которому повністю спущена маска – без маски

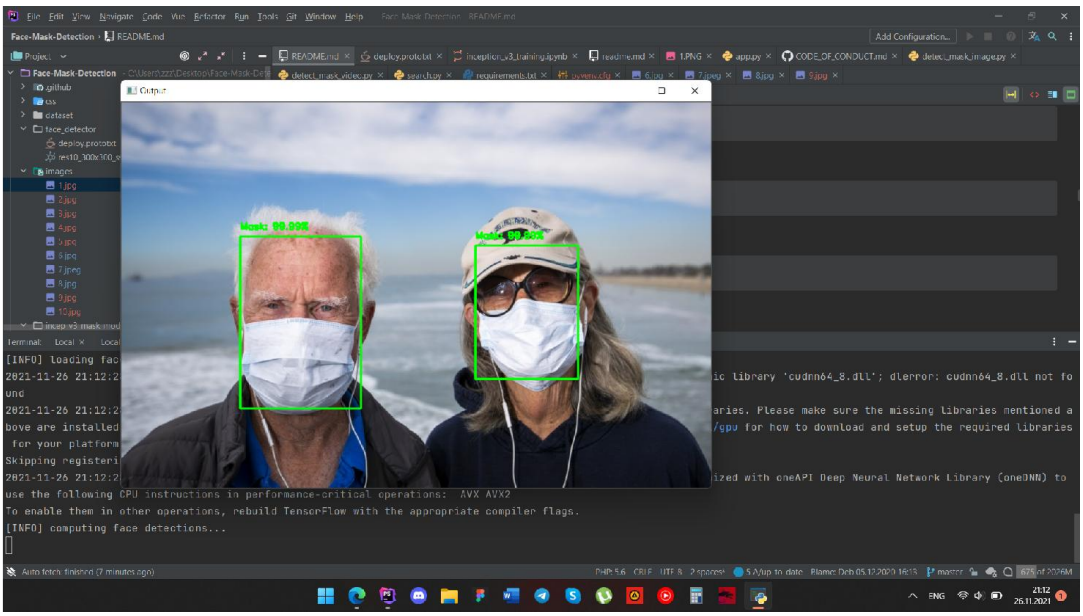

Рисунок 3.6 - Два обличча

Цей варіант інтересен нам тим що у жінки обличча майже закрито Капелюхом та окулярами. Але алгоритм все одно бачe що є обличча яке закрито маскою

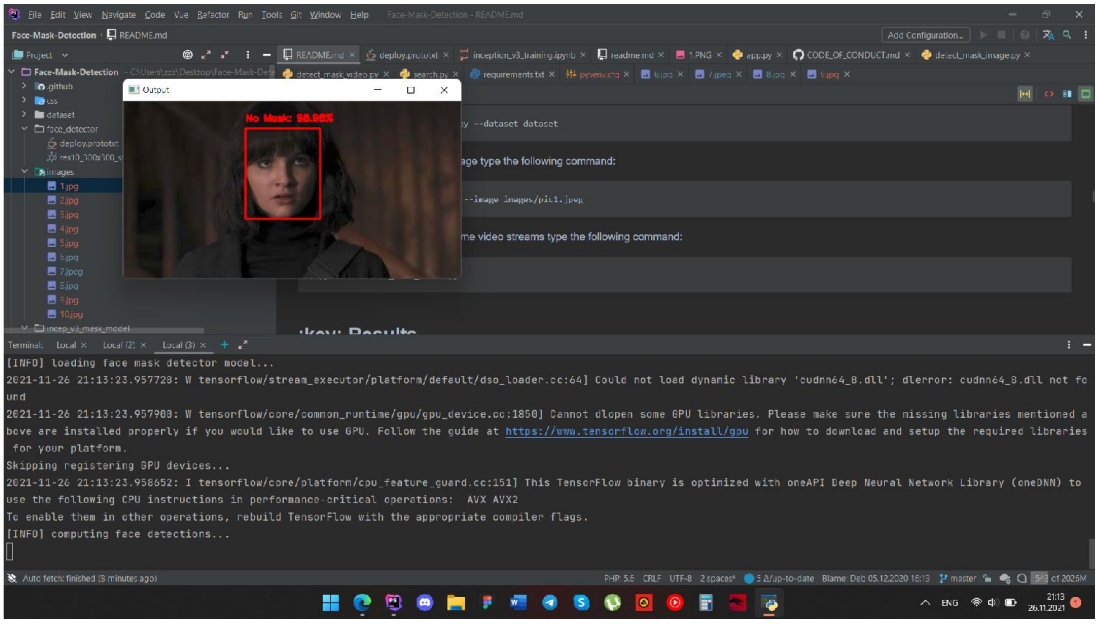

Рисунок 3.7 - Зображення з низкою вагою.

У цьому прикладі ми бачемо що алгоритм також нормально працює з зображеннями з низькою вагою і це добре

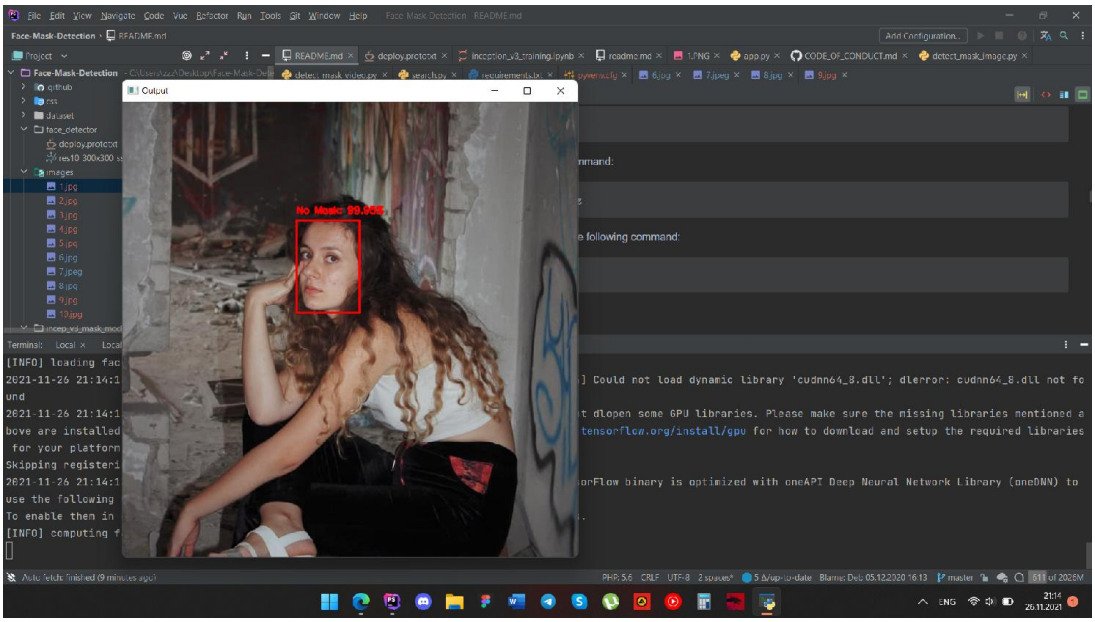

Рисунок 3.8 - Обличча яке ми бачемо під углом.

У цьому прикладі ми також бачимо що обличча під углом також нормально розпізнається алгоритмами.

ВИСНОВОК

Ми змогли досягнути нашої мети, а саме створите нейронну мережу яка може розпізнати облича без маски та в масці. TenserFlow дозволяє нам створювати різні нейронні мережі які можуть не тільки знаходити маски на фото, а ще й на відео у режимі реального часу. Цю мережу може використовувати держава для пошуку порушників санітарних норм, а на приклад вмикати попереджувальний аудіозапис чи викликати охорону. Також це можуть використовувати приватні підприємства для уникнення штрафів. Також, якщо вдосконалити нейронку, то ми можемо підщитувати кількість облич на зображенні, та фіксувати занадто велике скупчення людей.

На зараз ця нейронна мережа має деякі проблеми з розпізнаванням облич, яке плотно закрите чимось, але згодом це буду вирішено.

У цій дипломній роботі ми спочатку коротко пояснили мотивацію роботи. Потім ми проілюстрували завдання моделі на навчання та продуктивність. Використовуючи основні інструменти ML та спрощені методики, метод досяг досить високої точності. Його можна використовувати для різних застосувань. Носіння маски може стати обов’язковим у найближчому майбутньому, враховуючи кризу Covid-19. Багато постачальників державних послуг будуть просити клієнтів правильно носити маски, щоб скористатися їхніми послугами. Впроваджена модель зробить величезний внесок у державну систему охорони здоров’я. У майбутньому його можна буде розширити, щоб визначити, чи носить людина маску належним чином чи ні. Модель може бути додатково вдосконалена, щоб виявити, чи є маска вірусною чи ні, тобто тип маски хірургічний, N95 чи ні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. W.H.O., «Коронавірусна хвороба 2019 (covid-19): звіт про ситуацію, 205». 2020.

2. «Коронавірусна хвороба 2019 (COVID-19) – симптоми»[Електронний ресурс] Центри контролю та профілактики захворювань. - 2020. - Режим доступу до журн. :https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html. 2020 рік.

3. «Коронавірус — Типи коронавірусу людини — CDC»,.[Електронний ресурс]Cdc.gov. - 2020 Режим доступу до журн. :https://www.cdc.gov/coronavirus/types.html

4. W.H.O., «Поради щодо використання масок у контексті COVID-19: тимчасові рекомендації», 2020.

5., «RetinaMask: детектор маски для обличчя». [Електронний ресурс] / M. Цзян, X. Фан і X. Ян // arXiv.org. - 2020. Режим доступу до журн. :https://arxiv.org/abs/2005.03950

6. Б. Суварнамухі та М. Сешашаї, «Концепції та прийоми великих даних в обробці даних», Міжнародний журнал комп'ютерних наук та техніки, вип.

7., «Візуальна аналітика в глибокому навчанні: опитувальне опитування для наступних кордонів» у IEEETransactionsonVisualizationandComputerGraphics //F. Hohman, M. Kahng, R. Pienta та D. H. Chau2019. - С. 2674-2693

8. К. Кенен і Г. Коттрелл, «Від кольору до сірого: чи має значення метод у розпізнаванні зображень?»// К. Кенен і Г. Коттрелл. - 2012. Доступний: 10.1371/journal.pone.0029740.

9. Opencv-python-tutroals.readthedocs.io. Зміна колірних просторів[Електронний ресурс]Opencv-PythonTutorials 1 Документація -2020. Режим доступу до журн.: https://opencv-python-

tutroals.readthedocs.io/en/latest/pytutorials/
pyimgproc/pycolorspaces/pycolorspaces.html. 2020 рік.

10. М. Хашемі, «Збільшення менших зображень перед введенням у згорткову нейронну мережу: нульове заповнення проти інтерполяції» //JournalofBigData. - 2019.

11. С. Гош, Н. Дас та М. Насіпурі, “Переформування вхідних даних для згорткової нейронної мережі: деякі поширені та незвичайні методи”Розпізнавання образів, том. 93, С. 79-94, 2019

12. Р. Ямашіта, М. Нішіо, Р. До та К. Тогаші, “Згорткові нейронні мережі: огляд і застосування в радіології”, InsightsintoImaging, С. 611-629. - 2018.

13. «Керівництво до послідовної моделі – документація Keras», [Електронний ресурс] Faroit.com, 2020.Режим доступу до журн. :https://faroit.com/keras-docs/1.0.1/getting- start/sequential-model-guide/

14. C., Ijomah, W., Gachagan, A. and Marshall, S., 2020. Функції активації: порівняння тенденцій на практиці та дослідження для глибокого навчання. [Електронний ресурс]Nwankpa. -2020 Режим доступу до журн. :https://arxiv.org/abs/1811.03378

15. K. Team, «Документація Keras: шар MaxPooling2D[Електронний ресурс] »Keras.io. - 2020. Режимдоступудожурн. :https://keras.io/api/layers/poolinglayers/ maxpooling2d

16. «prajnasb/observations». [Електронний ресурс]// GitHub, - 2020 Режим доступу до журн. :https://github.com/prajnasb/observations/tree/master/experiements/data

17. «Виявлення маски для обличчя». [Електронний ресурс] \ Kaggle.com, - 2020. Режим доступу до журн. :https://www.kaggle.com/andrewmvd/face-mask-detection

18. “TensorFlow White Papers”, TensorFlow. [Електроннийресурс] – 2020. Режим доступу до журн. :https://www.tensorflow.org/about/bib

19.«Документація Keras: ПроKeras». [Електронний ресурс]Keras.io. - 2020 Режим доступу до журн. :https://keras.io/about

20. «OpenCV». [Електронний ресурс]Opencv.org- 2020 Режим доступу до журн. :https://opencv.org/

21. Д. Міна та Р. Шаран, «Підхід до виявлення та розпізнавання облич», 2016 Міжнародна конференція з останніх досягнень та інновацій в інженерії (ICRAIE)\ Джайпур, 2016. -С. 1-6

22. “Detecting Masked Faces in Wild with LLE-CNN”, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)\ S. Ge, J. Li, Q. Ye and Z. Luo\ Гонолулу, HI, 2017. - С. 426-434.

ДОДАТОК А
КОД ПРОГРАМИ

app.py

```
import streamlit as st
from PIL import Image, ImageEnhance
import numpy as np
import cv2
import os
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import detect_mask_image

# Setting custom Page Title and Icon with changed layout and sidebar state
st.set_page_config(page_title='Face Mask Detector', page_icon='😷', layout='centered',
initial_sidebar_state='expanded')


def local_css(file_name):
    """ Method for reading styles.css and applying necessary changes to HTML"""
    with open(file_name) as f:
        st.markdown(f'<style>{f.read()}</style>', unsafe_allow_html=True)


def mask_image():
    global RGB_img
    # load our serialized face detector model from disk
    print("[INFO] loading face detector model...")
    prototxtPath = os.path.sep.join(["face_detector", "deploy.prototxt"])
    weightsPath = os.path.sep.join(["face_detector",
                                    "res10_300x300_ssd_iter_140000.caffemodel"])
    net = cv2.dnn.readNet(prototxtPath, weightsPath)

    # load the face mask detector model from disk
    print("[INFO] loading face mask detector model...")
    model = load_model("mask_detector.model")

    # load the input image from disk and grab the image spatial
    # dimensions
    image = cv2.imread("./images/out.jpg")
    (h, w) = image.shape[:2]

    # construct a blob from the image
    blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
                                 (104.0, 177.0, 123.0))
```

```
# pass the blob through the network and obtain the face detections
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()

# loop over the detections
for i in range(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with
    # the detection
    confidence = detections[0, 0, i, 2]

    # filter out weak detections by ensuring the confidence is
    # greater than the minimum confidence
    if confidence > 0.5:
        # compute the (x, y)-coordinates of the bounding box for
        # the object
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        # ensure the bounding boxes fall within the dimensions of
        # the frame
        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

        # extract the face ROI, convert it from BGR to RGB channel
        # ordering, resize it to 224x224, and preprocess it
        face = image[startY:endY, startX:endX]
        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
        face = cv2.resize(face, (224, 224))
        face = img_to_array(face)
        face = preprocess_input(face)
        face = np.expand_dims(face, axis=0)

        # pass the face through the model to determine if the face
        # has a mask or not
        (mask, withoutMask) = model.predict(face)[0]

        # determine the class label and color we'll use to draw
        # the bounding box and text
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        # display the label and bounding box rectangle on the output
        # frame
        cv2.putText(image, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
```

```
            cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)
            RGB_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
mask_image()

def mask_detection():
    local_css("css/styles.css")
    st.markdown('<h1 align="center">😷 Face Mask Detection</h1>', unsafe_allow_html=True)
    activities = ["Image", "Webcam"]
    st.set_option('deprecation.showfileUploaderEncoding', False)
    st.sidebar.markdown("# Mask Detection on?")
    choice = st.sidebar.selectbox("Choose among the given options:", activities)

    if choice == 'Image':
        st.markdown('<h2 align="center">Detection on Image</h2>', unsafe_allow_html=True)
        st.markdown("### Upload your image here ⬇")
        image_file = st.file_uploader("", type=['jpg'])  # upload image
        if image_file is not None:
            our_image = Image.open(image_file)  # making compatible to PIL
            im = our_image.save('./images/out.jpg')
            saved_image = st.image(image_file, caption='', use_column_width=True)
            st.markdown('<h3 align="center">Image uploaded successfully!</h3>', unsafe_allow_html=True)
            if st.button('Process'):
                st.image(RGB_img, use_column_width=True)

    if choice == 'Webcam':
        st.markdown('<h2 align="center">Detection on Webcam</h2>', unsafe_allow_html=True)
        st.markdown('<h3 align="center">This feature will be available soon!</h3>',
unsafe_allow_html=True)
mask_detection()
```

detect_mask_image.py

```
# USAGE
# python detect_mask_image.py --image images/pic1.jpeg

# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import argparse
import cv2
import os
def mask_image():
        # construct the argument parser and parse the arguments
        ap = argparse.ArgumentParser()
        ap.add_argument("-i", "--image", required=True,
                help="path to input image")
        ap.add_argument("-f", "--face", type=str,
                default="face_detector",
```

```
        help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
        default="mask_detector.model",
        help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
        help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector model from disk
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
        "res10_300x300_ssd_iter_140000.caffemodel"])
net = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
model = load_model(args["model"])

# load the input image from disk, clone it, and grab the image spatial
# dimensions
image = cv2.imread(args["image"])
orig = image.copy()
(h, w) = image.shape[:2]

# construct a blob from the image
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
        (104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()

# loop over the detections
for i in range(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the detection
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the confidence is
        # greater than the minimum confidence
        if confidence > args["confidence"]:
                # compute the (x, y)-coordinates of the bounding box for
                # the object
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")

                # ensure the bounding boxes fall within the dimensions of
```

```
                        # the frame
                        (startX, startY) = (max(0, startX), max(0, startY))
                        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                        # extract the face ROI, convert it from BGR to RGB channel
                        # ordering, resize it to 224x224, and preprocess it
                        face = image[startY:endY, startX:endX]
                        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                        face = cv2.resize(face, (224, 224))
                        face = img_to_array(face)
                        face = preprocess_input(face)
                        face = np.expand_dims(face, axis=0)

                        # pass the face through the model to determine if the face
                        # has a mask or not
                        (mask, withoutMask) = model.predict(face)[0]

                        # determine the class label and color we'll use to draw
                        # the bounding box and text
                        label = "Mask" if mask > withoutMask else "No Mask"
                        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

                        # include the probability in the label
                        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

                        # display the label and bounding box rectangle on the output
                        # frame
                        cv2.putText(image, label, (startX, startY - 10),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
                        cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)

        # show the output image
        cv2.imshow("Output", image)
        cv2.waitKey(0)

if __name__ == "__main__":
        mask_image()
```

detect_mask_image.py

```
# USAGE
# python detect_mask_image.py --image images/pic1.jpeg

# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
```

```
from tensorflow.keras.models import load_model
import numpy as np
import argparse
import cv2
import os
def mask_image():
	# construct the argument parser and parse the arguments
	ap = argparse.ArgumentParser()
	ap.add_argument("-i", "--image", required=True,
		help="path to input image")
	ap.add_argument("-f", "--face", type=str,
		default="face_detector",
		help="path to face detector model directory")
	ap.add_argument("-m", "--model", type=str,
		default="mask_detector.model",
		help="path to trained face mask detector model")
	ap.add_argument("-c", "--confidence", type=float, default=0.5,
		help="minimum probability to filter weak detections")
	args = vars(ap.parse_args())

	# load our serialized face detector model from disk
	print("[INFO] loading face detector model...")
	prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
	weightsPath = os.path.sep.join([args["face"],
		"res10_300x300_ssd_iter_140000.caffemodel"])
	net = cv2.dnn.readNet(prototxtPath, weightsPath)

	# load the face mask detector model from disk
	print("[INFO] loading face mask detector model...")
	model = load_model(args["model"])

	# load the input image from disk, clone it, and grab the image spatial
	# dimensions
	image = cv2.imread(args["image"])
	orig = image.copy()
	(h, w) = image.shape[:2]

	# construct a blob from the image
	blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
```

```
	(104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()

# loop over the detections
for i in range(0, detections.shape[2]):
	# extract the confidence (i.e., probability) associated with
	# the detection
	confidence = detections[0, 0, i, 2]

	# filter out weak detections by ensuring the confidence is
	# greater than the minimum confidence
	if confidence > args["confidence"]:
		# compute the (x, y)-coordinates of the bounding box for
		# the object
		box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
		(startX, startY, endX, endY) = box.astype("int")

		# ensure the bounding boxes fall within the dimensions of
		# the frame
		(startX, startY) = (max(0, startX), max(0, startY))
		(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

		# extract the face ROI, convert it from BGR to RGB channel
		# ordering, resize it to 224x224, and preprocess it
		face = image[startY:endY, startX:endX]
		face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
		face = cv2.resize(face, (224, 224))
		face = img_to_array(face)
		face = preprocess_input(face)
		face = np.expand_dims(face, axis=0)

		# pass the face through the model to determine if the face
		# has a mask or not
		(mask, withoutMask) = model.predict(face)[0]
```

```
                    # determine the class label and color we'll use to draw
                    # the bounding box and text
                    label = "Mask" if mask > withoutMask else "No Mask"
                    color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

                    # include the probability in the label
                    label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

                    # display the label and bounding box rectangle on the output
                    # frame
                    cv2.putText(image, label, (startX, startY - 10),
                          cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
                    cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)

      # show the output image
      cv2.imshow("Output", image)
      cv2.waitKey(0)

if __name__ == "__main__":
      mask_image()
```

detect_mask_video.py

```
# USAGE
# python detect_mask_video.py

# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
```

```
# grab the dimensions of the frame and then construct a blob
# from it
(h, w) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
      (104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections
faceNet.setInput(blob)
detections = faceNet.forward()

# initialize our list of faces, their corresponding locations,
# and the list of predictions from our face mask network
faces = []
locs = []
preds = []

# loop over the detections
for i in range(0, detections.shape[2]):
      # extract the confidence (i.e., probability) associated with
      # the detection
      confidence = detections[0, 0, i, 2]

      # filter out weak detections by ensuring the confidence is
      # greater than the minimum confidence
      if confidence > args["confidence"]:
            # compute the (x, y)-coordinates of the bounding box for
            # the object
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # ensure the bounding boxes fall within the dimensions of
            # the frame
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            # extract the face ROI, convert it from BGR to RGB channel
            # ordering, resize it to 224x224, and preprocess it
            face = frame[startY:endY, startX:endX]
            if face.any():
```

```
                        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                        face = cv2.resize(face, (224, 224))
                        face = img_to_array(face)
                        face = preprocess_input(face)

                        # add the face and bounding boxes to their respective
                        # lists
                        faces.append(face)
                        locs.append((startX, startY, endX, endY))

      # only make a predictions if at least one face was detected
      if len(faces) > 0:
            # for faster inference we'll make batch predictions on *all*
            # faces at the same time rather than one-by-one predictions
            # in the above `for` loop
            faces = np.array(faces, dtype="float32")
            preds = maskNet.predict(faces, batch_size=32)

      # return a 2-tuple of the face locations and their corresponding
      # locations
      return (locs, preds)

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", type=str,
      default="face_detector",
      help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
      default="mask_detector.model",
      help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
      help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector model from disk
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
      "res10_300x300_ssd_iter_140000.caffemodel"])
```

```
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
maskNet = load_model(args["model"])

# initialize the video stream and allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# loop over the frames from the video stream
while True:
	# grab the frame from the threaded video stream and resize it
	# to have a maximum width of 400 pixels
	frame = vs.read()
	frame = imutils.resize(frame, width=400)

	# detect faces in the frame and determine if they are wearing a
	# face mask or not
	(locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

	# loop over the detected face locations and their corresponding
	# locations
	for (box, pred) in zip(locs, preds):
		# unpack the bounding box and predictions
		(startX, startY, endX, endY) = box
		(mask, withoutMask) = pred

		# determine the class label and color we'll use to draw
		# the bounding box and text
		label = "Mask" if mask > withoutMask else "No Mask"
		color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

		# include the probability in the label
		label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

		# display the label and bounding box rectangle on the output
		# frame
```

```
            cv2.putText(frame, label, (startX, startY - 10),
                  cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
            cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

      # show the output frame
      cv2.imshow("Frame", frame)
      key = cv2.waitKey(1) & 0xFF

      # if the `q` key was pressed, break from the loop
      if key == ord("q"):
            break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

search.py

```
from requests import exceptions
import argparse
import requests
import cv2
import os
ap = argparse.ArgumentParser()
ap.add_argument("-q", "--query", required=True,
      help="search query to search Bing Image API for")
ap.add_argument("-o", "--output", required=True,
      help="path to output directory of images")
args = vars(ap.parse_args())
API_KEY = "d8982f9e69a4437fa6e10715d1ed691d"
MAX_RESULTS = 500
GROUP_SIZE = 50
URL = "https://api.cognitive.microsoft.com/bing/v7.0/images/search"
EXCEPTIONS = set([IOError, FileNotFoundError,
      exceptions.RequestException, exceptions.HTTPError,
      exceptions.ConnectionError, exceptions.Timeout])
term = args["query"]
headers = {"Ocp-Apim-Subscription-Key" : API_KEY}
params = {"q": term, "offset": 0, "count": GROUP_SIZE}
print("[INFO] searching Bing API for '{}'".format(term))
```

```
search = requests.get(URL, headers=headers, params=params)
search.raise_for_status()
results = search.json()
estNumResults = min(results["totalEstimatedMatches"], MAX_RESULTS)
print("[INFO] {} total results for '{}'".format(estNumResults,
      term))
total = 0
for offset in range(0, estNumResults, GROUP_SIZE):
      print("[INFO] making request for group {}-{} of {}...".format(
            offset, offset + GROUP_SIZE, estNumResults))
      params["offset"] = offset
      search = requests.get(URL, headers=headers, params=params)
      search.raise_for_status()
      results = search.json()
      print("[INFO] saving images for group {}-{} of {}...".format(
            offset, offset + GROUP_SIZE, estNumResults))
      for v in results["value"]:
            try:
                  print("[INFO] fetching: {}".format(v["contentUrl"]))
                  r = requests.get(v["contentUrl"], timeout=30)
                  ext = v["contentUrl"][v["contentUrl"].rfind("."):]
                  p = os.path.sep.join([args["output"], "{}{}".format(
                        str(total).zfill(8), ext)])
                  f = open(p, "wb")
                  f.write(r.content)
                  f.close()
            except Exception as e:
                  if type(e) in EXCEPTIONS:
                        print("[INFO] skipping: {}".format(v["contentUrl"]))
                        continue
            image = cv2.imread(p)
            if image is None:
                  print("[INFO] deleting: {}".format(p))
                  os.remove(p)
                  continue
            total += 1
```

train_mask_detector.py

```
from requests import exceptions
import argparse
import requests
import cv2
import os
ap = argparse.ArgumentParser()
ap.add_argument("-q", "--query", required=True,
	help="search query to search Bing Image API for")
ap.add_argument("-o", "--output", required=True,
	help="path to output directory of images")
args = vars(ap.parse_args())
API_KEY = "d8982f9e69a4437fa6e10715d1ed691d"
MAX_RESULTS = 500
GROUP_SIZE = 50
URL = "https://api.cognitive.microsoft.com/bing/v7.0/images/search"
EXCEPTIONS = set([IOError, FileNotFoundError,
	exceptions.RequestException, exceptions.HTTPError,
	exceptions.ConnectionError, exceptions.Timeout])
term = args["query"]
headers = {"Ocp-Apim-Subscription-Key" : API_KEY}
params = {"q": term, "offset": 0, "count": GROUP_SIZE}
print("[INFO] searching Bing API for '{}'".format(term))
search = requests.get(URL, headers=headers, params=params)
search.raise_for_status()
results = search.json()
estNumResults = min(results["totalEstimatedMatches"], MAX_RESULTS)
print("[INFO] {} total results for '{}'".format(estNumResults,
	term))
total = 0
for offset in range(0, estNumResults, GROUP_SIZE):
	print("[INFO] making request for group {}-{} of {}...".format(
		offset, offset + GROUP_SIZE, estNumResults))
	params["offset"] = offset
	search = requests.get(URL, headers=headers, params=params)
	search.raise_for_status()
	results = search.json()
	print("[INFO] saving images for group {}-{} of {}...".format(
		offset, offset + GROUP_SIZE, estNumResults))
```

```
for v in results["value"]:
	try:
		print("[INFO] fetching: {}".format(v["contentUrl"]))
		r = requests.get(v["contentUrl"], timeout=30)
		ext = v["contentUrl"][v["contentUrl"].rfind("."):]
		p = os.path.sep.join([args["output"], "{}{}".format(
			str(total).zfill(8), ext)])
		f = open(p, "wb")
		f.write(r.content)
		f.close()
	except Exception as e:
		if type(e) in EXCEPTIONS:
			print("[INFO] skipping: {}".format(v["contentUrl"]))
			continue
	image = cv2.imread(p)
	if image is None:
		print("[INFO] deleting: {}".format(p))
		os.remove(p)
		continue
	total += 1
```

ДОДАТОК Б

Input: Dataset including faces with and without masks

Output: Categorized image depicting the presence of face mask

for each image in the dataset **do{**

Visualize the image in two categories and label them

Convert the RGB image to Gray-

Resize the gray-scale image into 100x100

Normalize the image and convert it into 4 dimensional array

}

for Buliding the CNN model **do {**

Add a convolution layer of 200 filters

Add the second Convolution layer of 100 filters

Insert a Flatten layer to network classifier

Add a Dense layer of 64 neurons

Add the final Dense layer with 2 outputs for 2 categories

}

Split the data and train the model