**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

«На правах рукопису»
УДК ____ 004.8_____

До захисту допущено:
В. о. завідувача кафедри
_____ Олександр РОЛІК
«___»_____2021 р.

# Магістерська дисертація

**на здобуття ступеня магістра**
**за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»**
**зі спеціальності 126 «Інформаційні системи та технології»**
**на тему: «** Автоматизована система виявлення людей без засобів індивідуального захисту **»**

Виконав:
студент VI курсу, групи ІК-01мп
Рижий Андрій Русланович _____

Керівник:
доцент, к.т.н., доцент
Олійник Володимир Валентинович _____

Рецензент:
доцент кафедри ІПІ, к.т.н., доцент
Лісовиченко Олег Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ

В. о. завідувача кафедри

_____ Олександр РОЛІК

«___»_____2021 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**

**Рижий Андрій Русланович**

1. Тема дисертації « Автоматизована система виявлення людей без засобів індивідуального захисту», науковий керівник дисертації Олійник Володимир Валентинович доцент, к. т. н., затверджені наказом по університету від «27» 10 2021 р. № 3587-с

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – автоматизована система виявлення людей без ЗІЗ

4. Вихідні дані – методи детекції об'єктів

5. Перелік завдань, які потрібно розробити

1.Огляд та аналіз існуючих рішень;

2. Дослідити існуючі методи вирішення задачі для знаходження людей без ЗІЗ. Визначити їх переваги та недоліки;

3. Обрати та описати саму систему для знаходження людей без ЗІЗ;

4. Реалізувати систему навчання алгоритму детекції та автоматизовану систему знаходження людей без ЗІЗ на основі визначеного інформаційного забезпечення;

5. Провести експериментальне дослідження;

6. Проаналізувати результати;

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 8 плакатів

7. Орієнтовний перелік публікацій – Oliinyk V. An efficient face mask detection model for real-time applications / Oliinyk V., Ryzhiy A. // Adaptive systems of automatic control, 2021. Vol. 2, №39

8. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Огляд та аналіз існуючих рішень | 10.09.2021 | |
| 2 | Дослідити існуючі методи вирішення задачі для знаходження людей без ЗІЗ. Визначити їх переваги та недоліки | 25.09.2021 | |
| 3 | Обрати та описати саму систему для знаходження людей без ЗІЗ | 05.10.2021 | |
| 4 | Реалізувати систему навчання алгоритму детекції та автоматизовану систему знаходження людей без ЗІЗ на основі визначеного інформаційного забезпечення | 20.10.2021 | |
| 5 | Провести експериментальне дослідження | 01.11.2021 | |
| 6 | Проаналізувати результати | 15.11.2021 | |
| | | | |
| | | | |

Студент Андрій Рижий

Науковий керівник Володимир Олійник

# АНОТАЦІЯ

Дана робота присвячена дослідженню методів детекції об’єктів на задачі знаходження людей без засобів індивідуального захисту. В роботі реалізовано автоматизовану систему знаходження людей без ЗІЗ. Під час роботи розглянуто особливості роботи моделей для детектування облич, та реалізовано систему навчання такого алгоритму. Проведено серію експериментів, та обрано найкращий який в результаті працює на всіх складних прикладах.

Об’єкт дослідження – автоматизована система виявлення людей без ЗІЗ.

Мета магістерсьої дисетртації: підвищення ефективності розв’язку задачі виявлення людей без ЗІЗ.

За результатами досліджень теми магістерської дисертації опубліковано статтю у міжнародному науковому журналі.

Пояснювальна записка до магістерської дисертації має обсяг 100 сторінок та містить 55 рисунків, 26 таблиці, 26 літературних джерел.

Ключові слова: НЕЙРОННА МЕРЕЖА, ЕНКОДЕР, ЗНАХОДЖЕННЯ ОБЛИЧ, FEATURE ENCHANCE MODULE,  ЗАХИСНА МАСКА.

ABSTRACT

This project is devoted to the study of methods for detecting objects on the task of finding people without personal protective equipment. As a result, an automated system for finding people without PPE has been implemented. During the work the peculiarities of the work of models for face detection are considered, and the system of training of such algorithm is realized. A series of experiments was conducted, and the best one was chosen, which as a result works on all complex examples.

The object of study is an automated system for detecting people without PPE.

The aim of the diploma project is to increase the efficiency of solving the problem of identifying people without PPE.

According to the results of research on the topic of the master's dissertation, an article was published in an international scientific journal.

The explanatory note to the master's dissertation has a volume of 100 pages and contains 55 figures, 26 tables, 26 literary sources.

Keywords: NEURAL NETWORK, ENCODER, FACE DETECTION, FEATURE ENCHANCE MODULE, PROTECTIVE MASK.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

НМ – нейронна мережа

ЗІЗ – засоби індивідуального захисту

FEM – модуль покращення ознак

DSFD – модель dual shot face detector

API – програмний інтерфейс

# ВСТУП

Використання засобів індивідуального захисту є невід'ємною частиною багатьох сфер людського життя. Зокрема маски для обличчя широко використовуються для захисту від забрудненого повітря в будівництві, промисловості, сільському господарстві та для захисту від вірусів та патологічних бактерій. Крім того, в умовах пандемії Ковід19 з необхідністю носіння масок для обличчя зіткнулася фактично кожна доросла людини на планеті. Згідно рекомендацій Всесвітньої організації охорони здоров'я (ВООЗ) носіння масок є основним засобом переривання розповсюдження інфекції.

Очевидно це створює велику кількість задач, що потребує створення систем моніторингу та контролю використання ЗІЗ. Типовим прикладом є задача виявлення людей без маски для обличчя на основі відео або фото. Широке розповсюдження камер спостереження в громадських місцях дозволяє без зайвих витрат отримати дані для моніторингу та здійснення контролю. Тому основною задачею є створення високоефективних та доступних систем обробки відео-потоку для детектування відсутності на людині ЗІЗ.

Задача може ускладнюватися: якщо на зображенні є значна кількість людей, то потрібно знаходити кожного з них; на відео з камер спостереження люди часто не дивляться в камеру, а то і більше знаходиться в профіль; під час зйомки в нічний час бувають шуми та низькоякісне зображення; багато людей, заради полегшення носіння маски, роблять це не правильно, залишаючи ніс поза маскою тощо. Додатковими укладенням є наявність даних, пандемія триває недовго, а тому великої кількості даних не можна було зібрати, тому для моделювання носіння масок потрібно застосувати інструменти з комп'ютерної графіки.

Задачу детектування об'єктів на відео можна без втрати узагальнення розглядати як задачу обробки зображень, що є кадрами відео. Наявність послідовності зображень дозволяє додатково підвищити якість та надійність пошуку шляхом трекінгу об'єктів на послідовності кадрів, однак дослідження методів трекінгу не є об'єктом даної роботи. Тому розглядається задача знаходження людей без масок на зображенні.

Традиційним на сьогодні підходом інтегрованого штучного інтелекту для задачі знаходження об’єктів на зображені/відео є використання object detectors на основі варіацій згорткових нейронних мереж. Їх я і буду досліджувати в цій роботі в задачі знаходження людей без маски. Також в цій роботі буде представлена повністю автоматизована система виявлення людей без засобів індивідуального захисту, яка знаходить людей без масок за допомогою навченої нейронної мережі, та інформує користувача про наявність таких людей на потоці відео. Окремою частиною даної системи є власне навчання нейронної мережі та проведення експериментів, в рамках цієї роботи розроблено систему за допомогою якої можна зручно експериментувати та навчати різні варіації нейронних мереж.

# 1 ЗАДАЧА АВТОМАТИЗОВАНОЇ СИСТЕМА ВИЯВЛЕННЯ ЛЮДЕЙ БЕЗ ЗАСОБІВ ІНДИВІДУАЛЬНОГО ЗАХИСТУ

## 1.1 Постановка мети дослідження

Наразі світ перебуває під натиском COVID-19. COVID-19 – це інфекційне захворювання, спричинене важким гострим респіраторним синдромом (SARS-CoV-2). Люди можуть заразитися, вступаючи в тісний соціальний контакт із інфікованою людиною через дихальні краплі під час кашлю, чхання та/або розмови. Крім того, вірус також можна поширювати, торкаючись поверхні або предмета, на якому є вірус, а потім торкаючись рота, носа чи очей. Наразі ми можемо захистити себе, уникаючи контакту з вірусом. За даними Всесвітньої організація охорони здоров'я (ВООЗ), найкращий спосіб уникнути поширення хвороби або зараження – це дотримуватися соціальної дистанції та носити обличчя під час перебування в громадських місцях. Два основних підходи до профілактики – це уникнення непотрібного контакту та носіння маски. Виконання цих рекомендацій є основними для захисту від інфекції. Проте часто люди нехтують цими правилами, а для того щоб слідкувати за ними, і не давати вірусу розповсюджуватись витрачаються великі кошти, тому для полегшення цього завдання пропоную використовувати методи штучного інтелекту в цій задачі.

На сьогоднішній день системи штучного інтелекту вже дуже часто зустрічаються в нашому життя. Одне із найпопулярніших застосувань – це системи комп’ютерного зору, це задачі класифікації, сегментації, детекції зображень. Це пов’язано з тим що ця сфера вже достатньо добре розвинена, та задачі доволі добре вирішені. Підтвердженням цьому є багато прикладі: на результати роботи даних систем опираються системи автоматичного керування автомобілями, системи безпеки, які вміють достатньо добре розпізнавати людей, що свідчить про те що дані способи вже доволі точні, та на них можна опиратись в задачах які пов’язані з безпекою та життям людей.

Задача ж автоматизованої системи виявлення людей без засобів індивідуального захисту також відноситься до тих, від яких залежить здоров’я та життя людей. Оскільки ця тема ще не є добре дослідженою, то рішення які зараз

пропонуються є не найкращими, або не можуть справлятись із задачею знаходження людей без засобів особистого захисту взагалі. Тому метою даної роботи є підвищення ефективності розв’язку задачі виявлення людей без ЗІЗ. Критеріями оптимізації є точність знаходження таких людей а також швидкість роботи системи. Об’єктом дослідження є автоматизована система виявлення людей без ЗІЗ. А предметом дослідження у даній роботі є методи детекції об’єктів.

Для досягнення поставленої мети у процесі роботи необхідно виконати такі задачі:

- провести аналіз можливих технологій для детекції об’єктів;
- дослідити існуючі методи вирішення задачі для знаходження людей без ЗІЗ, визначити їх переваги та недоліки;
- обрати та описати саму систему для знаходження людей без ЗІЗ;
- реалізувати систему навчання алгоритму детекції та автоматизовану систему знаходження людей без ЗІЗ на основі визначеного інформаційного забезпечення;
- провести експериментальне дослідження алгоритму детекції людей без ЗІЗ;
- дослідити результати роботи реалізованої системи знаходження людей без ЗІЗ.

## 1.2 Опис задачі знаходження людей без ЗІЗ і особливостей задач детекції облич

Задачу автоматизованої системи виявлення людей без засобів індивідуального захисту можна сформулювати так: нехай у нас є постійний потік відео з якогось громадського місця у якому можуть бути люди без ЗІЗ, потрібно спеціальне програмне забезпечення яке зможе знаходити на цьому потоці людей без ЗІЗ і повідомляти про це користувача. Тут є декілька особливостей:

- оскільки відео потік це певна послідовність фото, то задачу детекції можна розглядати як задачу детекції на одному фото, що дещо спрощує задачу;
- якщо розглядати задачу як знаходження об’єкта на фото, то потрібно щоб швидкість 1 обробленої фотографії була близька до реального часу. Тобто

як і швидкість роботи самого детектора, так і інтерфейсу, і інших елементів має бути високою;

– дана система може запускатись на різних комп’ютерах, тому важливою буде кросплатформеність системи;
– якщо буде великий потік людей, користувачу важливо знати тільки про тих, хто не має ЗІЗ, тобто система повинна знаходити кожну окрему особину без ЗІЗ;
– важливою особливістю системи має бути сповіщення користувача про найдених людей, тому у програмі має бути це передбачено.

Оскільки головною задачею є знаходження об’єктів на фото, або відео, то формально ця задача вирішується відомими методами object detections.

Задача Object detection – на заданій картинці потрібно віднайти і обвести прямокутником об’єкти певного класу. Найкращі рішення  такої задачі – це підходи R-CNN сімейства (R-CNN[1], Fast R-CNN[2], Faster R-CNN[3], Mask R-CNN[4]), YOLO сімейства (YOLO[5], YOLOV3[6], YOLOV4[7], YOLOX[8]), SSD[9] та інші.

Ці всі підходи гарно працюють для знаходження об’єктів певних класів (дуже різних між собою) часто різної форми, кольору та розміру: стовп, автомобіль, кіт, собака та інші. Проте якщо подивитись на обличчя людей, вони дуже схожі між собою: в усіх по два ока, два вуха, дві брови, ніс та рот, це значно спрощує задачу object detection тому виокремили окремий клас задач: face detection. Це клас задач які спеціалізується для знаходжень саме облич людей, системи, архітектури мереж проектуються знаючи особливості облич. А оскільки задача знаходження людей без масок дуже схожа до задачі детекції облич, то для створення системи знаходження людей без масок, я буду досліджувати один з найкращих face detectors: Dual Shot Face Detector.

## 1.3 Існуючі підходи до вирішення задачі

Можна розглянути декілька основних підходів для вирішення задачі:

Підходи тільки класифікації зображень. До таких підходів відносяться доволі прості методи класифікації зображень, наприклад [10, 11]. Для коректної роботи

потрібно фотографію обрізувати до обличчя, а для цього потрібний окремий детектор. Проте якщо його вчити окремо, а потім вчити окремо класифікатор, то результат комбінації буде працювати гірше. А для тренування таких мереж потрібно буде більше часу та ресурсів, а ніж просто детектор який буде знаходити людей на два класи: люди в масці та люди без маски. Також важливим є те що методи які зразу детектують правильним класом, як правило, працюють краще, аніж окремі системи.

Використання штучних датасетів. До пандемії Ковід19 в світі не були поширені засоби ЗІЗ, тому очевидним недоліком для використання автоматизованого методу детектування облич була відсутність датасетів на початку пандемії. Тому багато дослідників використовували метод штучного накладання масок на обличчя людей. За допомогою різних фреймворків комп'ютерної графіки можна згенерувати, або намалювати самі маски різних форм та розміру. Також можна використати датасети в яких вже є розмітка ключових точок облич (ніс, рот, очі, брови та інші) які вирішують задачу face landmarks detection (приклад такої розмітки на рис 1.1). І якщо ж накласти маску так щоб вона закривала ніс, та рот, то таким чином датасет можна було перетворити із простого face detection в masked face detection [12, 13, 14].

Рисунок 1.1 Приклад датасету з face landmarks

Зараз такі методи можна використовувати в якості аугментації для існуючих датасетів.

Використання мереж SSD чи YOLO для швидкого але низькоякісного детектування. Такі підходи вже давно популярні в face detection за рахунок швидкості інференсу, їх часто застосовують в роботі з відео, та використовують на пристроях з малою потужністю, наприклад смартфонах. Основною їх перевагою є час роботи, проте якість цієї роботи не дуже висока. Такі детектори працюють за принципом single shot detection, та їх представники це різні генерації YOLO та SSD мереж. Детектори які працюють довго, але якісно працюють за принципом two shot, їхні представники це сімейство R-CNN мереж.

Основна відмінність між цими підходами в тому що в two shot детекторах використовується два етапи. Перший - це пропозиція регіону, це етап на якому за допомогою згорткової нейронної мережі по вхідному зображенні знаходять регіони на яких гіпотетично можуть бути якісь об’єкти. Другий – це класифікація цих регіонів та уточнення прогнозу за допомогою повнозв'язних нейронних мереж. Детектори single shot працюють без проміжного етапу пропозиції регіону, вони напряму знаходять об’єкти, це спрощення і є причиною швидкодії і гіршої якості.

Одним з прикладів такого підходу в задачі детектування маскованих облич є WearMask detector [15].

Dual Shot Face Detector [16]. Оскільки знаходження облич без засобів особистого захисту дуже схожа на задачу просто знаходження облич, то я думаю варто розглянути один з найкращих Face detectors. В цій роботі автори розробили новий спосіб знаходження облич за допомогою мережі яка використовує принципи які лежать десь між single shot і two shot детекторами. В цієї мережі є два кроки, проте перший це не створення пропозиції регіонів. Перший крок – це прогон вхідного зображення через вже відомі архітектури класифікаторів (Vgg, ResNet та інші) та створення певного внутрішнього представлення вхідного зображення в різних масштабах. Після чого ці представлення проходять через Feature Enhance Module (модуль покращення характеристик) який їх покращує для кращого детектування. FEM зображений на рис. 1.2. Він складається з операції згортки з розміром фільтру 1x1 та з різною кількістю фільтрів, для кожного з рівня знаходження ознак. Це модуль зв’язує ознаки в зворотному порядку і пропускає через себе їх

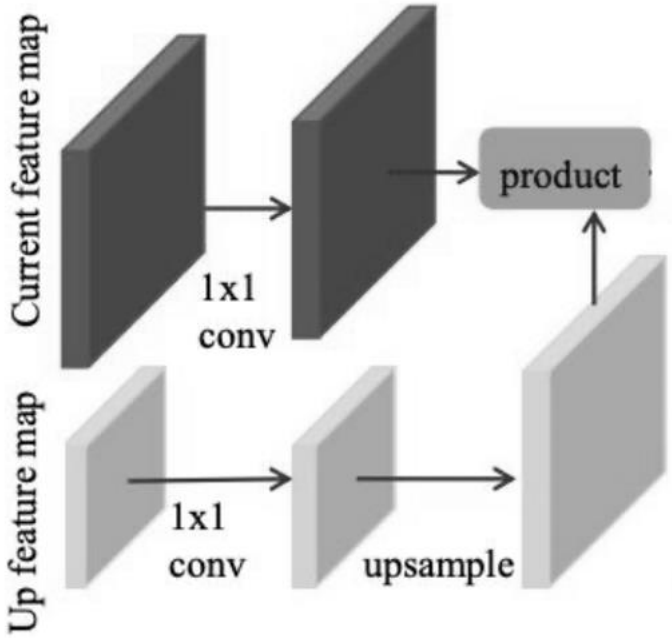


Рисунок 1.2 Feature Enhance Module

Другий крок схожий до першого, але за основу береться не зображення, а ознаки які ми отримали після FEM модуля. І вони об’єднуються в один шар. Заключний крок – це за допомогою ознак з першого та другого кроку зробити передбачення знаходження об’єкту.

Також до переваг цієї мережі можна додати використання Progressive Anchor Loss що обчислюється двома різними наборами якорів, щоб ефективно полегшити функції, та Improved Anchor Matching (IAM) (покращену відповідність якорів) шляхом інтеграції нової стратегії призначення якорів у збільшення даних для забезпечення кращої ініціалізації. У дослідженнях авторів, мережа показала себе краще за інші у задачі face detection, а отже є ймовірність того що вона покаже себе і краще на задачі masked face detection.

До переваг цієї мережі ще можна віднести використання різних енкодерів, за допомогою цього можна регулювати якість та швидкість мережі. Тобто для швидкого та менш якісного знаходження можна використовувати Mobilenet енкодер, а для більш якісного, проте повільнішого можна використовувати Vgg чи інші глибокі мережі. Саме цю мережу я і буду досліджувати в даній роботі.

## 1.4 Існуючі реалізації

Підходи для класифікації зображень. Реалізації цих підходів відсутні у відкритих джерелах, але їх не складно зробити самому. Задача класифікації це найпростіша задача у сфері комп'ютерного зору, та приклади її реалізації існують на всіх відомих бібліотеках машинного навчання. І щоб навчити таку систему, достатньо знайти підходящий датасет. Проте такі системи не підійдуть для моїх задач, оскільки знаходження окремих об'єктів у кадрі це не задача класифікації.

Використання штучних датасетів. Найкращий результат було знайдено в реалізації SMFD датасету а саме – SMFD датасет [14]. Він використовує алгоритм з [12]. Суть якого полягає у використанні бібліотеки dlib для детектування обличчя та визначення нахилу обличчя та шести ключових особливостей лиця необхідних для накладання маски. На основі нахилу обличчя вибирається відповідний шаблон маски з бібліотеки масок. Шаблон маски потім трансформується на основі шести ключових функцій, щоб ідеально прилягати до обличчя. Алгоритм пропонує близько 100 різних варіантів масок на вибір та його можна використовувати для будь-якого наявного набору даних з обличчями без маски в той що має вже маски. Він ідентифікує всі обличчя на зображенні та застосовує до них вибрані користувачем маски з урахуванням різних обмежень, таких як кут обличчя, посадка маски, умови освітлення тощо. Відповідно всі параметри задаються випадковим чином для випадкового створення масок. Нижче наведено особливості роботи системи:

- підтримка кількох типів масок: алгоритм підтримує 5 різних типів масок, крім того користувач може додавати власні;
- підтримка варіантів масок: алгоритм надає 24 шаблони які можна застосовувати для різних типів масок, крім того користувачі можуть додавати власні кольори та текстури дотримуючись наданих інструкцій;
- підтримка зображень як з одним так і з декількома обличчями. Можна застосувати маски для всіх облич на зображенні без будь-яких обмежень;
- кожна маска для обличчя має кілька шаблонів на основі кута, отже охоплює широкий діапазон нахилів обличчя;

– легко застосовувати для готового набору даних.

В [13] автори представили кращий алгоритм ніж у [12], він вирізняється більш реалістичним накладанням. Це зумовлено використанням не просто двовимірних ключових точок лиця, а використанням трьох вимірних. Алгоритм розпізнає тривимірне положення обличчя в майже 90% випадків. Також маючи 3Д точки обличчя, можна і перетворювати маску на обличчя в 3Д. Також в цьому підході маску не накладають бінарно, а використовують спеціальній альфаканал, який згладжує маску по краям. Це допомагає накладати маску реалістичніше. Автори проводили опитування в якому люди голосували за те де правильніше накладена маска в порівнянні з [12] і за результатами в [13] вийшла реалістичніша маска.

Автори [14] об’єднали переваги [12] і [13] та представили свою імплементацію даного алгоритму штучного накладання масок, та в своїй роботі я буду використовувати датасет який негенерований таким чином. Сама реалізація доволі просто написана, тому буду використовувати їхню імплементацію для аугментації або додавання нових зображень під час навчання. На рисунку 1.3 зображено приклад з датасету негенерованого таким чином.


Рисунок 1.3 Приклад із SMFD датасету

Використання мереж SSD чи YOLO для швидкого але низькоякісного детектування. Одним з прикладів такого підходу в задачі детектування маскованих облич є WearMask detector [15] В цій роботі автори використовують YOLO архітектуру для знаходження людей з масками і без. І пропонують метод знаходження масок в real time з використанням high-performance neural network inference computing framework (NCNN), and a stack-based virtual machine (WebAssembly). Також в цій роботі автори формують датасет для досліджень який я буду використовувати у своїх експериментах.

Серед інших реалізацій можна відмітити tensorflow object detection API [17] або mmdetection [18]. В них представлені реалізації різних систем знаходження об’єктів, серед яких є і найпопулярніші. Тут є реалізації YOLO, SSD, R-CNN мереж, та інших які можна використовувати для детекції об’єктів. Проте в них немає реалізації DSFD мережі, яку я планую використати. Тому такі системи мені не підходять.

DSFD мережа. Реалізація даної мережі є в [19]. Проте в даній реалізації є використання навчання тільки для знаходження одного, тобто немає можливості використовувати два класи під час навчання. Також автори не досліджували дану мережу конкретно для людей які не носять ЗІЗ. Я буду використовувати дану реалізацію в своєму дослідження, проте модифікую код для кращого експериментування (описано в 3.2), додам можливість використання двох класів та проведу експерименти з датасетом який націлений для знаходження людей без масок.

## 1.5 Формулювання конкретних вимог до розробки

Проаналізувавши існуючі підходи та їхні реалізації було розроблено вимоги до розробки моєї програмної реалізації:

– найближчим варіантом до мого дипломного проєкту є варіант з Dual shot face detector. Його і буду використовувати за основу. Це швидкий і якісний алгоритм для знаходження облич який можна буде адаптувати та використовувати як алгоритм знаходження людей без масок;

- для слабких комп’ютерів використовувати менші енкодери (такі як Mobilenet) для більш потужних, або тих які мають GPU в себе використовувати більш важкі мережі і точніші мережі, наприклад vgg чи resnet. Тобто потрібно реалізувати на навчити декілька варіантів нейронних мереж;
- для того щоб можна було робити експерименти з постановкою задачі та власне нейронною мережею потрібно створити інфраструктуру для створення мережі. Вона повинна включати в себе можливість вибору різних енкодерів та різних постановок задач, має бути зручна у використанні та гнучкою у заданні параметрів;
- для підвищення якості навчання використати штучно згенеровані набори даних або методи штучного накладання масок на обличчя для підвищення якості;
- використовувати відео як послідовність фотографій і робити знаходження по одній фотографії;
- система повинна працювати в реальному часі;
- система повинна бути кросплатформеною для можливості запуску на різних пристроях;
- система має мати можливість запуску на CPU та GPU;
- для зручної роботи з програмою, вона повинна мати графічний інтерфес;
- система повинна інформувати користувача про наявність в кадрі людей без ЗІЗ;
- система повинна відмічати окремих людей які не мають ЗІЗ (для чого і підходить object detector);

## 1.6 Висновки

В цьому розділі було визначено мету дослідження, об’єкт та предмет дослідження. Описано задачі які потрібно виконати під час даної роботи.

Проведено аналіз існуючих рішень та аналогів. Виконавши його було виявлено, що зараз є рішення які просто класифікують зображення, тобто не можуть підходити під вимоги системи.

Знайдено багато рішень які реалізовані на основі класичного детектування об’єктів, проте тільки деякі з них реалізовані для детектування облич без маски. Ті які реалізовані це або швидкі але не ефективні мережі. Вони працюють з доволі низькою якість, що не підходить для даної задачі.

Наближчою задачею до моєї було визначено face detection, тому для мого проекту я виростав одну з найкращих мереж в цій задачі: dual shot face detector. Оскільки в даному детекторі можливо змінювати енкодери, то потрібно провести експерименти з різними, для різної потужності обчислювальних ресурсів.

Також проаналізовано методи створення датасетів, а особливо методи штучного накладання масок на обличчя. Було прийнято рішення використовувати їх у своєму навчання для підвищення якості.

Також в цьому розділі було формалізовано основні вимоги до розробки програмного забезпечення. Самі ж моделі та експериментальне дослідження проведено у наступних розділах.

# 2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

## 2.1 Опис моделі машинного навчання для детекції облич

В своїх дослідженнях я буду використовувати DSFD мережу, яка описана в першому розділі. Суть її як і більшості детекторів в наступному. На вхід мережі подається картинка, з трьома каналами. Після чого ця картинка проходить ряд шарів нейронної мережі – це згортки, pooling шари, нормалізаційні шари, шари активації, повнозвязні шари, таким чином щоб на виході можна було отримати три масиви чисел:

- масив координат. В кожному елементі міститься 4 числа які характеризують позицію прямокутника в межах якого знаходить об’єкт, тобто в нашому випадку людина без маски. В моїй мережі такі 4 числа це дві координати цього прямокутника: верхній лівий та нижній правий. Ці числа всі в межах від 0 до 1, для отримання абсолютних значень їх достатньо помножити на висоту на ширину вхідного зображення;
- масив ймовірностей. Для того щоб можна було оцінити впевненість мережі використовується цей масив. Як і у більшості класифікаційних мереж це вихід після шару з активацією. Для кожного прямокутника з масиву координат визначається його ймовірність. Вона може бути в межах від 0 до 1, і чим ближче до 1 тим впевненіша мережа в передбачені певного прямокутника. На практиці використовується відкидання всіх прямокутників які менше певного значення, у моєму випадку це 0.5;
- масив класів. Оскільки детектор може одночасно знаходити об’єктів декількох класів, то потрібна інформація від мережі про відповідність кожного прямокутника до конкретного класу, за це і відповідає цей масив. В ньому знаходять числа від 0 до $n$, де $n$ – це максимальна кількість класів.

Структурна схема мережі зображена на рисунку 2.1.

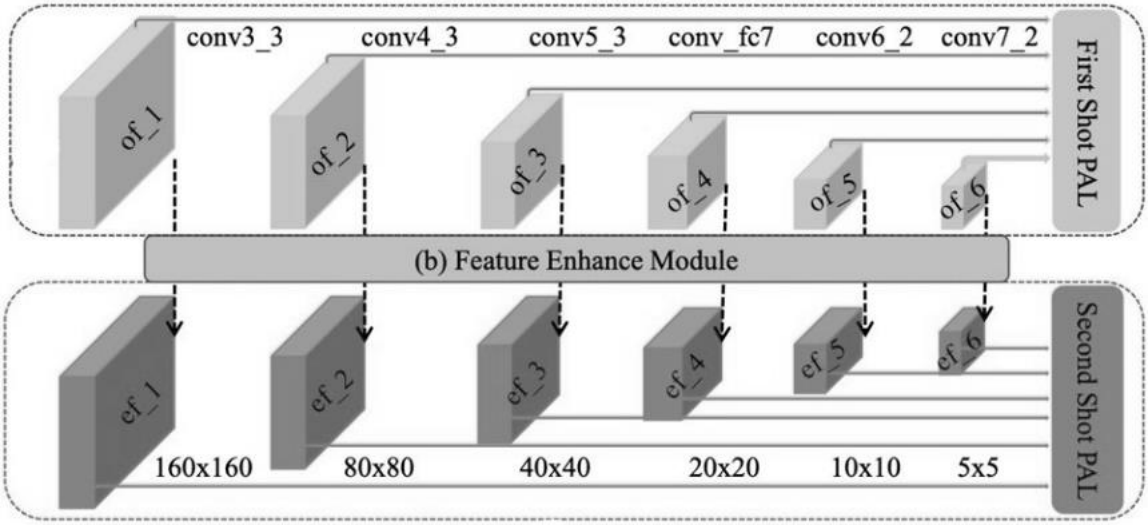


Рисунок 2.1 Архітектура DSFD мережі

2.1.1 Опис енкодерів які будуть використовуватись

Першим кроком, як і у більшості згорткових мереж, є проходження вхідного зображення через вже відому архітектуру класифікатора (енкодер). Цей крок потрібен для того щоб отримати ознаки, з якими потім можна буде працювати. Оскільки DSFD мережа дозволяє використати різні енкодери, я розглянув наступні: Vgg16, ResNet50, MobileNetV2. Розглянемо ці енкодери детальніше.

2.1.1.1 Vgg16[20]

Це проста і широко розповсюджена архітектура згорткової мережі. По сучасним міркам вже доволі застаріла модель, проте до сьогодні використовується в різних проектах. Вона була створена ще у 2014 році та проектувалась і використовувалась насамперед для задач класифікації зображень. Вхідне зображення проходить через набір згорток 3x3, в цій мережі вперше почали використовувати менші згорткові шари (в старших роботах часто можна було зустріти інші розміри: 5x5, 7x7 і тд, проте зараз використовуються тільки 3x3). А в кінці проходить через повно зв’язний шар. Архітектура Vgg16 представлена на рисунку 2.2.

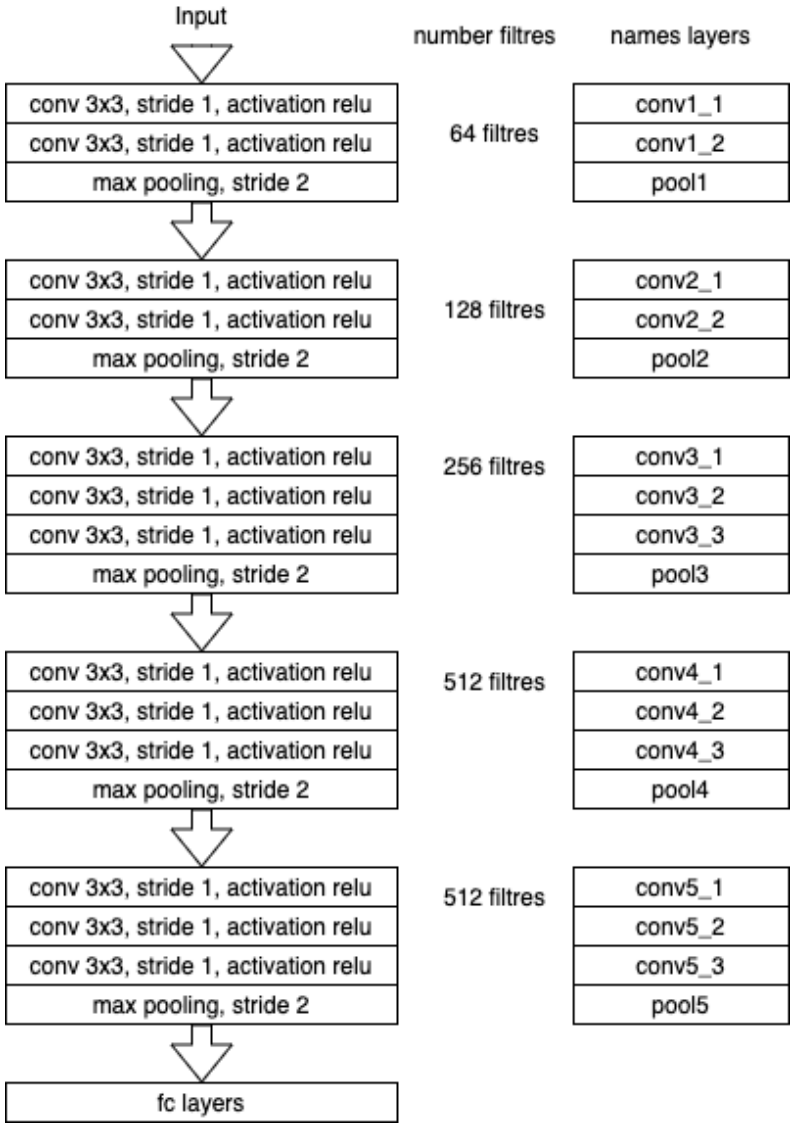


Рисунок 2.2. Архітектура Vgg16

Для мережі яку я використовую, було вибрано в у якості оригінальних ознак було обрано такі шари:

– для $of_1$ – шар conv3_3;
– для $of_2$ – шар conv4_3;
– для $of_3$ – шар conv5_3.

Але у моїй мережі використовувались 6 оригінальних ознак, тому у якості останніх трьох я додав додаткові шари. Для $of_4$ я використав дві згортки розміром 3x3 та 1x1 відповідно. З кількістю фільтрів 1024 і функцією активаціїї – relu. На вхід цим згорткам я додав шар з оригінальної моделі з назвою pool5.

Для $of_5$ та $of_6$ я застосував алгоритм додавання додаткових шарів на основі $of_4$. Він заключається у послідовному додаванні згорток з різною кількістю фільтрів та різним відступом, сам алгоритм зображений на рисунку 2.3.

```
def extra_feature(x):

    with tf.variable_scope('extra'):
        x=slim.conv2d(x, 256, [1, 1],stride=1,
                      activation_fn=tf.nn.relu,
                      normalizer_fn=None,
                      scope='extra_conv1')
        x = slim.conv2d(x, 512, [3, 3],stride=2,
                        activation_fn=tf.nn.relu,
                        normalizer_fn=None,
                        scope='extra_conv2')
        of5 = x

        x = slim.conv2d(x, 128, [1, 1], stride=1,
                        activation_fn=tf.nn.relu,
                        normalizer_fn=None,
                        scope='extra_conv3')
        x = slim.conv2d(x, 256, [3, 3], stride=2,
                        activation_fn=tf.nn.relu,
                        normalizer_fn=None,
                        scope='extra_conv4')
        of6 = x

    return of5, of6
```

Рисунок 2.3. Алгоритм додавання додаткових шарів до енкодера

2.1.1.2 ResNet50

Це варіант моделі ResNet, яка має 48 шарів згортки разом із 1 шаром максимального пулінгу, та 1 шаром середнього пулінгу. До представлення світу

ResNet моделей, надглибокі моделі не могли навчитись набагато краще ніж моделі з меншою кількістю шарів, оскільки виникала проблема вибуху градієнтів. І автори моделі вирішили це питання додавши зв’язки не тільки між послідовними шарами, а і між не послідовними шарами. Такий додатковий зв’язок називають залишковим. Він зображений на рисунку 2.4.

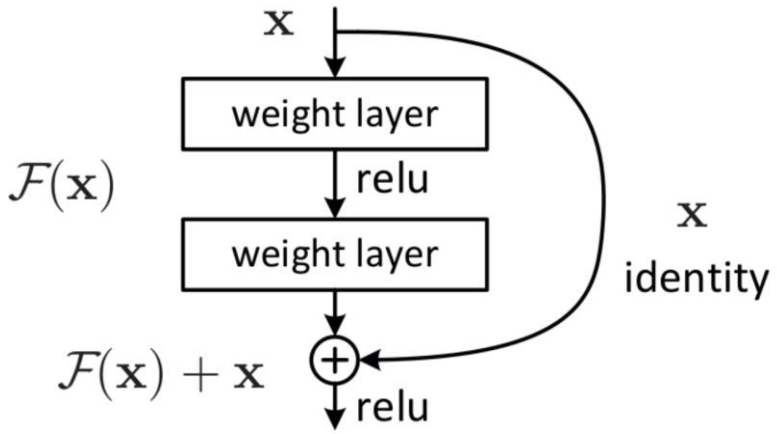


Рисунок 2.4. Залишковий зв’язок у ResNet

За допомогою цього зв’язка автори дозволили не тільки краще навчатись дуже глибоким мережам, а і в цілому змогли підвищити якість моделі, бо шар мережі в який входив цей зв’язок «бачив» інформацію не тільки безпосередньо від попередньго шару, а і дещо перед ним.

У моєму випадку я використовував архітектуру ResNet50, це не найбільш складна і не найбільш складна моделі сімейства ResNet. Її архітектура представлена на рисунку 2.5.

Для використання такої мережі у детекторі я використав наступні шари у якості оригінальних ознак:

- для $of_1$ – останній шар блоку 1;
- для $of_2$ – останній шар блоку 2;
- для $of_3$ – останній шар блоку 3;
- для $of_4$ – останній шар блоку 4.

Для створення додаткових шарів я використав такий же алгортим який використовував у мережі Vgg16.

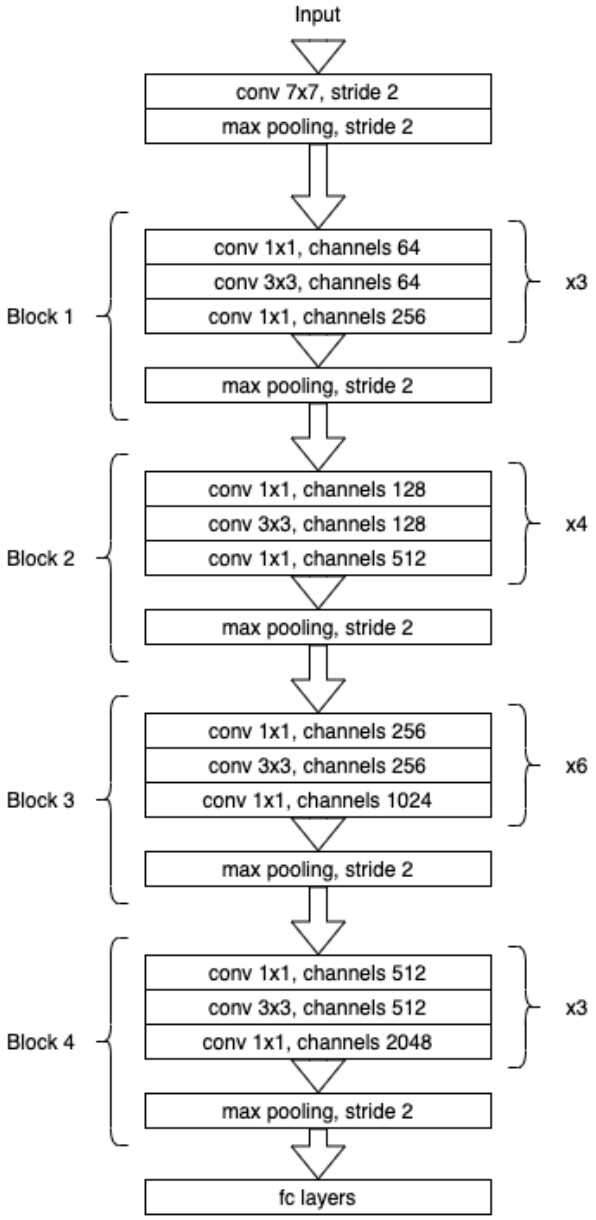


Рисунок 2.5. Архітектура ResNet50

2.1.1.3 MobileNetV2

MobileNetV2[23]. Ця мережа розроблялась спеціально для мобільних пристроїв, і основна її перевага перед іншими це швидкість роботи. Це досягається шляхом використання Relu6 активації та глибинною згорткою (depthwise convolution) яка вирізняється швидкодією під час запуску на процесорах, а математично

представляє собою звичайну згортку. Ця архітектура складається з блоків які називаються bottleneck та складаються з 1x1 згортки з Relu6, глибинної згортки 3x3 з Relu6 та 1x1 згортки з лінійною функцією активації. Детальніше архітектура представлена на рисунку 2.6:

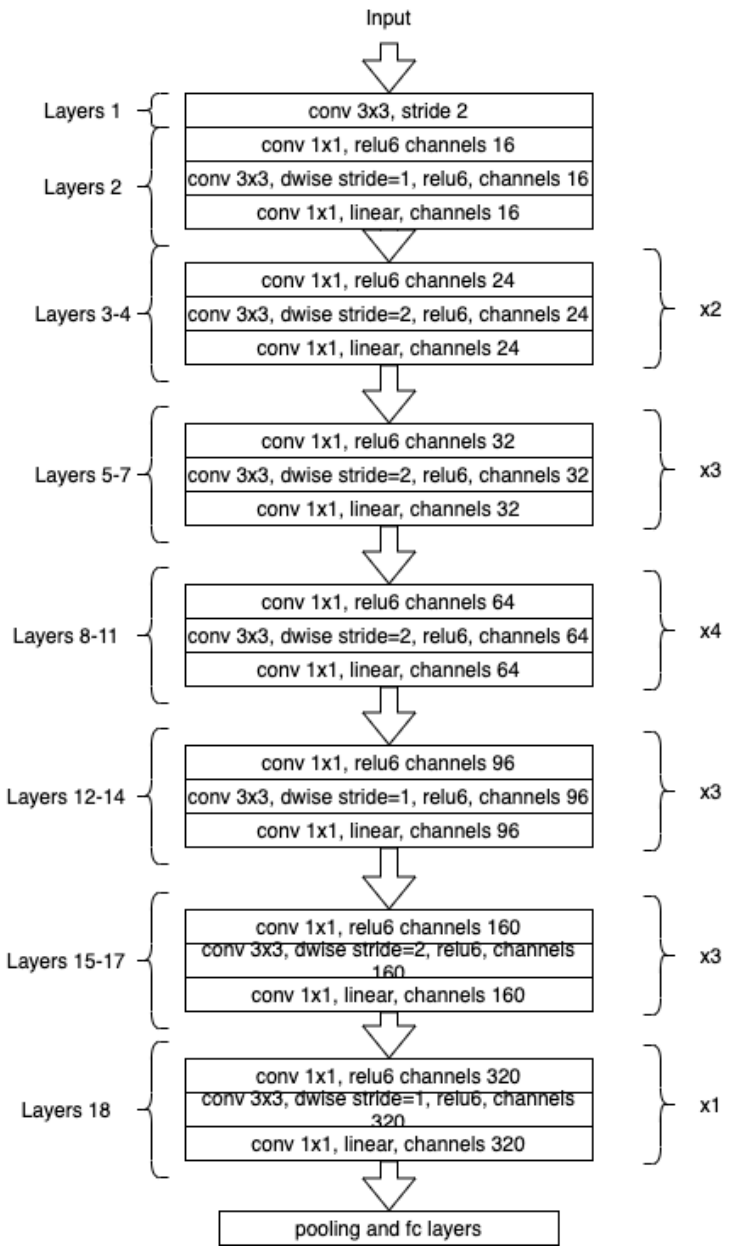


Рисунок 2.6. Архітектура MobileNetV2

В своєму детекторі я використовував виходи з MobileNetV2 в якості таких оригнальних ознак:

– для $of_1$ – останній шар Layers 5;

– для $of_2$ – останній шар Layers 8;
– для $of_3$ – останній шар Layers 15;
– для $of_4$ – останній шар Layers 18.

Для генерування додаткових ознак, як і у випадку з Vgg та ResNet я використав алгоритм додавання додаткових шарів до енкодера.

### 2.1.2 Feature Enhance Module

Після проходу енкодера є 6 проміжних виходів – $of_i$, кожен з яких має різний розмір та кількість каналів. Для вхідної картинки розміром 320x320 яку я використовую, їхні розміри будуть такі (табл. 2.1):

Таблиця 2.1 Розміри вхідних картинок

| Ознака | Розмір |
|---|---|
| $of_1$ | 80x80 |
| $of_2$ | 40x40 |
| $of_3$ | 20x20 |
| $of_4$ | 10x10 |
| $of_5$ | 5x5 |
| $of_6$ | 2x2 |

Після отримання оригінальних ознак з енкодера, використовується спеціальний шар який покращує ознаки – Feature Enhance Module. Алгоритм цього модуля наступний. Спочатку для перших трьох проміжних виходів з енкодера $of\_i$, де $i = 1,2,3$ виконується наступне:

1) беруться $of_3$ і $of_4$ ознаки. Для кожної з них проводиться згортка з розміром ядра 1x1 та з кількістю фільтрів 288, без функції нормалізації та без функції активації. Після чого вихід зі згортки з 4 ознаки збільшується у два рази та перемножується на вихід зі згортки для 3 ознаки. Таким чином отримується третя проміжна покращена ознака ($tef_3$);

2) береться $of_2$ і $tef_3$ ознаки. Для них проводиться згортка аналогічна попередній, але з кількістю фільтрів рівною 96. Також вихід з більшої ознаки збільшується у 2 рази та перемножується з першим. Після цього кроку отримується друга проміжна покращена ознака ($tef_2$);
3) аналогічно другій проміжній ознаці, отримується перша $tef_1$;
4) для ознак 4,5 та 6, проміжні покращені ознаки такі ж як і оригінальні ознаки, тобто $tef_i = of_i, i = 4,5,6$.

Реалізація цього алгоритму представлено на рисунку 2.7.

Після отримання проміжних покращених ознак, вони покращуються за допомогою розширених згорток (dilation convolution). Алгоритм полягає в тому що для кожної проміжної покращеної ознаки $tef_i, i = 1..6$ виконується ряд операцій, а саме створюються три паралельні виходи:

– перший – це згортка 3x3 з 128 фільтрами, без додатковий умов;
– другий – це комбінація з двох згорток. Перша складається з 3x3 та 128 фільтрів, і має коефіцієнт розширення 2, тобто згортка працює через 1 піксель. Друга отримує на вхід вихід першої, та складається з 3x3 та 64 фільтрів;
– третій є аналогом другої, проте використовує 64 фільтри для кожної згортки.

У кожній з цих згорток не використовується нормалізайна функція, та використовується функція активації relu. Отримавши три виходи, вони об’єднуються в один, та вважаються покращеною ознакою $ef_i$. Математично це представлено в 2.1.

$$ef_i = concat\left(dilation_{conv(tef_i)}\right) \tag{2.1}$$

Реалізація розширених згорток на фреймворку tensorflow представлена на рисунку 2.8. А повна схема мережі з енкодером MobilenetV2 та модуля FEM представлена в додатку А.

```
of0, of1, of2, of3, of4, of5 = blocks

lateral = slim.conv2d(of2, dims_list[2], [1, 1],
                      trainable=trainable, weights_initializer=initializer,
                      padding='SAME', normalizer_fn=None, activation_fn=None,
                      scope='lateral/res{}'.format(2))

upsample = slim.conv2d(of3, dims_list[2], [1, 1],
                       trainable=trainable, weights_initializer=initializer,
                       padding='SAME', normalizer_fn=None, activation_fn=None,
                       scope='merge/res{}'.format(2), data_format=data_format)
upsample = tf.keras.layers.UpSampling2D(data_format='channels_last' if data_format == 'NHWC' else 'channels_first')(
    upsample)

fem_2 = lateral * upsample

lateral = slim.conv2d(of1, dims_list[1], [1, 1],
                      trainable=trainable, weights_initializer=initializer,
                      padding='SAME', normalizer_fn=None, activation_fn=None,
                      scope='lateral/res{}'.format(1))

upsample = slim.conv2d(fem_2, dims_list[1], [1, 1],
                       trainable=trainable, weights_initializer=initializer,
                       padding='SAME', normalizer_fn=None, activation_fn=None,
                       scope='merge/res{}'.format(1), data_format=data_format)
upsample = tf.keras.layers.UpSampling2D(data_format='channels_last' if data_format == 'NHWC' else 'channels_first')(
    upsample)

fem_1 = lateral * upsample

lateral = slim.conv2d(of0, dims_list[0], [1, 1],
                      trainable=trainable, weights_initializer=initializer,
                      padding='SAME', normalizer_fn=None, activation_fn=None,
                      scope='lateral/res{}'.format(0))

upsample = slim.conv2d(fem_1, dims_list[0], [1, 1],
                       trainable=trainable, weights_initializer=initializer,
                       padding='SAME', normalizer_fn=None, activation_fn=None,
                       scope='merge/res{}'.format(0), data_format=data_format)
upsample = tf.keras.layers.UpSampling2D(data_format='channels_last' if data_format == 'NHWC' else 'channels_first')(
    upsample)

fem_0 = lateral * upsample

fpn_fms = [fem_0, fem_1, fem_2, of3, of4, of5]
```

Рисунок 2.7. Реалізація першої частини Feature Enhance Module

```
def cpm(product,scope):
    dim=256

    with tf.variable_scope(scope):
        eyes_1 = slim.conv2d(product, dim // 2, [3, 3], stride=1, rate=1, activation_fn=tf.nn.relu, normalizer_fn=None,
                             scope='eyes_1')

        eyes_2_1 = slim.conv2d(product, dim // 2, [3, 3], stride=1, rate=2, activation_fn=tf.nn.relu,
                               normalizer_fn=None, scope='eyes_2_1')
        eyes_2 = slim.conv2d(eyes_2_1, dim // 4, [3, 3], stride=1, rate=1, activation_fn=tf.nn.relu, normalizer_fn=None,
                             scope='eyes_2')

        eyes_3_1 = slim.conv2d(eyes_2_1, dim // 4, [3, 3], stride=1, rate=2, activation_fn=tf.nn.relu,
                               normalizer_fn=None, scope='eyes_3_1')
        eyes_3 = slim.conv2d(eyes_3_1, dim // 4, [3, 3], stride=1, rate=1, activation_fn=tf.nn.relu, normalizer_fn=None,
                             scope='eyes_3')

        fme_res = tf.concat([eyes_1, eyes_2,eyes_3], axis=3)

    return fme_res
```

Рисунок 2.8. Реалізація розширених згорток

Після отримання оригінальних ти покращених ознак, можна робити передбачення безпосередньо прямокутників, ймовірностей та класів.

2.1.3 Передбачення безпосередньо прямокутників та відношення їх до класу людини з маскою чи без

Для передбачення прямокутників використовується додаткова згортка 3x3, з 4 фільтрами, без функції активації, де на виході отримуються 4 координати прямокутників, які лежать в межах від 0 до 1. Оскільки кожна з ознак відповідає за свій масштаб вхідної картинки, то відповідно чим за більшу область відповідає ознака тим менші обличчя вона може знаходити, так на виході з перших ознак будуть знаходитись менші обличчя ніж на останніх.

Для отримання передбачень ймовірностей та класів використовується додаткова згортка розміру 3x3, з кількостями фільтрів рівною кількості класів. Після цієї згортки застосовується функція активації softmax. Після якої ймовірності – це самі значення після softmax, а для знаходження класів до яких відносяться прямокутники, потрібно вибрати той, в якого найбільша ймовірність, тобто виконати операцію argmax.

### 2.1.4 Non Maximum Suppression

Потім отримані прямокутники разом з ймовірностями обробляться алгоритмом NMS, суть якого в наступному:

Типовий детектор об’єктів як і мій на виході видає велику кількість прямокутників. Це пов’язано з тим що під час інференсу відбувається генерування тисячі вікон різних розмірів та форм. Це означає що на один і той же об’єкт може попадати багато прямокутників, що дає неоднозначність у передбаченні. Саме для того щоб мати однозначний результат детекції і використовується алгоритм Non Maximum Suppression.

Для правильного розуміння його роботи, потрібно розуміти метрику на яку він опирається – Interception Over Union. Вона по суті є методом який використовують для кількісної оцінки перекриття між істинним прямокутником і прямокутником передбачення. Математично це можна представити виразом (2.2).

$$Intercetion\ Over\ Union\ (IOU) = \frac{Target\ \cap Prediction}{Target\ \cup Prediction} \tag{2.2}$$

Але у випадку детекції, істинне значення і передбачуване є прямокутниками, тому для IOU в детекції потрібно знаходити перетин та об’єднання між прямокутниками. Оскільки перетин двох прямокутників не може бути більшим ніж об’єднання їх, то IOU буде в межах від 0 до 1.

На вхід даному алгоритму подається список прямокутників (P), клас для кожного з прямокутників (c), та ймовірність для кожного прямокутника (pr). Також для роботи алгоритму нам потрібно задати поріг перекриття (thresh_iou). У моєму випадку я використовував thresh_iou=0.4.

Алгоритм:

Крок 1. Вибрати передбачення S з найбільшою ймовірністю, та видалити його з P, проте додати до фінального передбачення (final_P, final_c, final_pr).

Крок 2. Порівняти передбачення S з усіма передбаченнями які є в P. Порівняти два передбачення - значить обчислити IOU між їхніми прямокутниками. Якщо

отримане IOU більше ніж порогове значення (thresh_iou) то для кожного такого передбачення T, видалити T з P.

Крок 3. Якщо в P ще залишились передбачення перейти до кроку 1, інакше повернути фінальні передбачення.

Після виконання такого алгоритму на виході з мережі залишаються найбільш впевнені передбачення які не повторюються. Візуально процес NMS можна побачити на рисунку 2.9:

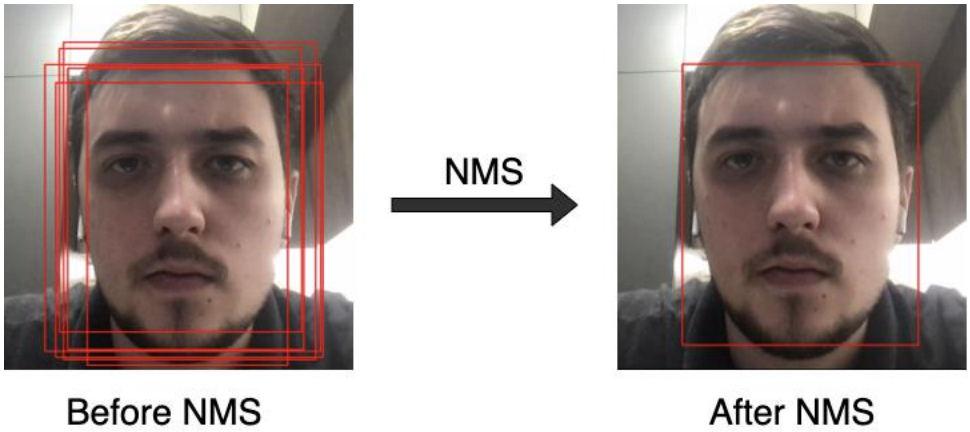


Рисунок 2.9 Приклад роботи NMS

2.1.5 Опис помилки для навчання моделі та алгоритм її знаходження

Даний алгоритм працює тільки для інференсу моделі, а для навчання він відключається, проте під час навчання працює алгоритм зворотного поширення помилки, а сама помилка рахується наступним чином:

В класичних детекторах функція втрат складається з двох частин: класифікаційної частини яка визначає до якого класу відноситься той чи інший прямокутник та з регресивної частини, яка відповідає за розміщення кожного прямокутника.

Для класифікаційної частини використовується функція втрат як для звичайного класифікатора – кросс ентропія. Її можна формалізувати формулою, представленою у (2.3,2.4):

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} \log\left(\hat{c}_j^p\right) - \sum_{i \in Neg} \log\left(\hat{c}_j^0\right), \tag{2.3}$$

$$\hat{c}_i^p = \frac{\exp\left(c_i^p\right)}{\sum_p \exp\left(c_i^p\right)} \tag{2.4}$$

де $N$ – це кількість прямокутників які відповідають істинним прямокутникам (процес відповідності описаний нижче). Pos – множина позитивних передбачень, тобто ті, які відповідають істинним передбачення, а Neg, навпаки, множина негативних передбачень, ті, яких немає в істинних передбаченнях. $\hat{c}_j^p$ – передбачена ймовірність класу $p$, для передбачення $i$, $x_{ij}^p$ – індикатор відповідності для $i$-го передбачення та $j$-го істинного прямокутника і класу p. В даній помилці використовуються також негативні передбачення, це зроблено для того щоб штрафувати модель за false positive передбачення.

В регресивній частині, зазвичай, використовується згладжена L1 норма, вона вираховується за формулою (2.5).

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{x1,y1,x2,y2\}} x_{ij}^{k} smoothL1\left(l_i^m - \hat{g}_j^m\right), \tag{2.5}$$

де $l$ – прямокутник який передбачила мережа, $g$ – істинні значення прямокутників, а згладжена L1 рахується так (2.6):

$$smoothL1(a) = \begin{cases} \frac{1}{2}a^2, |a| \leq 1 \\ |a| - \frac{1}{2} \end{cases} \tag{2.6}$$

Використання саме згладженої функції зумовлено тим, що така втрата є менш чутлива до викидів, а ніж звичайна L1. Не використання L2 зумовлено тим, що в

більшості випадків використання L2 вимагає більшого налаштування гіперпараметрів для уникнення вибуху градієнта [1].

І в класичному детекторі загальна помилка це сума класифікаційної та регресійної помилки. А у моєму випадку я використовую дещо модифіковану функцію. Оскільки в моєму випадку використовуються два кроки, і з виходів кожного кроку формується фінальне передбачення, то логічно використовувати і різні якорі (anhors) для різних кроків. Опираючись на твердження з [23] про те що для маленьких облич краще підходять для низько рівневі ознаки. Тому для першого кроку, де ознаки високо рівневі я буду використовувати менші якорі, а для другого кроку – більші. Таким чином я отримаю дві окремі функції втрат.

Для першого кроку (2.7):

$$L_{FS}(x, c, l, g, sa) = L_{conf}(x, c) + L_{loc}(x, l, g, sa), \tag{2.7}$$

де $sa$ означає що використовуються малі якорі для встановлення відповідності між передбачуваним та істинним обличчям. Якщо порівнювати вихід з другого кроку, то вихід з першого містить менше семантичної інформації, але містить більше інформації про розташування на більшому розширенні ніж другий крок, тому є підстави припускати, що вихід з першого кроку знаходить краще менші обличчя.

Для другого кроку (2.8):

$$L_{SS}(x, c, l, g, a) = L_{conf}(x, c) + L_{loc}(x, l, g, a), \tag{2.8}$$

де $a$ означає використання звичайних якорів для встановлення відповідності між передбачуваним та істинним обличчям. Тоді для фінальної помилки доцільно використати вираз (2.9):

$$L = L_{FS}(sa) + \lambda L_{SS}(a), \tag{2.9}$$

де $\lambda$ використовується як коефіцієнт яким можна корегувати для яких умов використовується такий детектор. Якщо $\lambda < 1$ то детектор краще буде працювати на малих обличчях, інакше на великих. У своїх експериментах я використовував $\lambda = 1$.

В додатку Б нарисована схема роботи мережі під час виконання інференсу та під час тренування.

### 2.2.5 Генерування прямокутників за замовчуванням для навчання моделі

Для того щоб використовувати для підрахунку загальної помилки правильні істинні значення потрібен процес відповідності (matching) між ними. У класичному варіанті цей процес відбувається так: для кожного істинного значення ($gt_i$) знаходиться IOU для кожного якоря. Потім з створюється відповідність для $gt_i$ кожного якоря, де IOU > певного порогу, у більшості випадків це 0.5. Після чого всі передбачення які «вийшли» з даного якоря відповідають цьому $gt_i$. І функція втрат вже рахується маючи істинне значення як $gt_i$. У моєму випадку відбувється схожий алгоритм, але для малих і великих облич шукаються і використовуються різні якорі.

Ключова ідея single shot детекторів це використання якорів, або як їх іще називають прямокутниками за замовчуванням. Вони представляють собою ретельно відібрані прямокутники які мають різні розміри, співвідношення сторін і розташування на зображенні. Основна ціль моделі що навчається полягає в тому, щоб вирішити які з прямокутників за замовчуванням використовувати для даного зображення, а потім передбачити зміщення від вибраних блоків для остаточного прогнозу. Тому ці прямокутники мають покривати всю фотографію. Оскільки, відомо що перші шари детектора через те що мають менше сприятливе поле краще знаходять менші об’єкти, а чим далі по мережі тим шари знаходять більші зображення. Тому потрібно генерувати прямокутники для різного кожного набору ознак які є або оригінальними або покращеними, при чому для більших за розміром ознак потрібно генерувати більші за розміром прямокутники. Розглянемо приклад із генеруванням для ознак розміром 5x5x256.

В якості центра кожного прямокутника доцільно обрати центр кожної із ознак та масштабувати їх до оригінального зображення. На рис 2.10 зображено центри прямокутників для генерації:


Рисунок 2.10. Центри прямокутників для генерації

Перед початком генерації задаються параметри для подальшої генерації. Це кількість прямокутників (я використовував 3) на один набір ознак; розмір основного прямокутника (для конкретно цієї карти ознак це 168); та співвідношення сторін для прямокутників (я використовував 2).

Потім відбувається сам процес генерування. Спочатку створюється основний прямокутник розміром 168x168, та ставиться на однин з центрів які ми отримали з набору ознак. Після чого генеруються ще два прямокутники які мають співвідношення сторін яке задане в параметрах, у моєму випадку це 2, тому розміри будуть $168 * \sqrt{2}$ та $168/\sqrt{2}$ . Приклад таких прямокутників на рисунку 2.11.

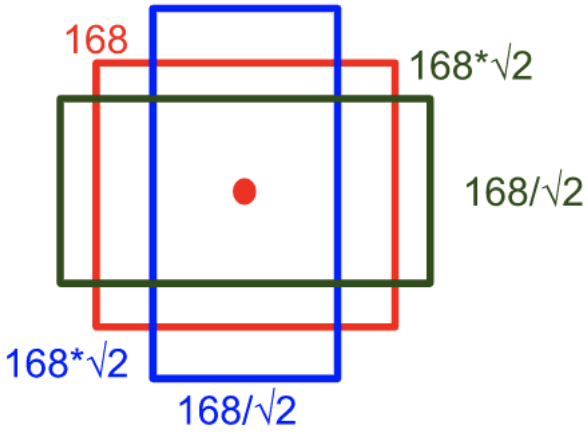


Рисунок 2.11. Приклад прямокутників за замовчуванням

Після чого така структура із прямокутників використовується для кожної точки які ми обрали і набору ознак.

Такий алгоритм виконується для кожного набору ознак. Із кількістю прямокутників заданій 3 та такою кількістю оригінальних ознак що є в моїй мережі, таких прямокутників буде 8732 для повної картинки. Проте я використовую ще покращений набір ознак, і для нього також буде ще 8732 прямокутника. Тому загальна кількість прямокутників за замовчуванням 8732*2=17464. Також слід замітити що параметр розмір прямокутника в оригінальних ознаках у два рази більший ніж у покращених.

### 2.1.6 IOU – основна метрика для порівняння експериментів

Для того щоб правильно порівнювати експерименти між собою, потрібно ввести метрику за допомогою якої можна визначати кращі експерименти. Для тренування детекторів існує багато різних метрик, я ж зупинюсь на mAP при IOU = 0.5. Для її розуміння, потрібно спочатку розуміти що таке точність та повнота.

Точність (precision) показує наскільки точні передбачення. Тобто який відсоток передбачень є правильним. Повнота (recall) показує наскільки добре знайдено всі позитивні прямокутники. Математично точність можна представити як (2.10) а повноту як (2.11).

$$Precision = \frac{TP}{TP + FP}, \tag{2.10}$$

$$Recall = \frac{TP}{TP + FN}, \tag{2.11}$$

де $TP$ – кількість істино позитивного, $FP$ – кількість помилок I роду, $FN$ – кількість помилок II роду. З цих формул можна побачити що точність та повнота можуть бути тільки в межах від 0 до 1.

Для розуміння AP (average precision) розглянемо простий приклад. Нехай я маю набір даних в якому є тільки 5 облич. А також маю детектор який їх знаходить. Нехай детектор після всіх операцій знайшов 10 облич, і ми порахували точність та повноту для цих передбачень, та відсортуємо ці передбачення по точності. Також будемо вважати що передбачення є коректним, якщо воно має IOU > 0.5 з істинним значенням. Результати представлені в таблиці 2.2.

Таблиця 2.2 Результати передбачень

| № | Чи коректне передбачення | Точність | Повнота |
|---|---|---|---|
| 1 | Так | 1.0 | 0.2 |
| 2 | Так | 1.0 | 0.4 |
| 3 | Ні | 0.67 | 0.4 |
| 4 | Ні | 0.5 | 0.4 |
| 5 | Ні | 0.4 | 0.4 |
| 6 | Так | 0.5 | 0.6 |
| 7 | Так | 0.57 | 0.8 |
| 8 | Ні | 0.5 | 0.8 |
| 9 | Ні | 0.44 | 0.8 |
| 10 | Так | 0.5 | 1.0 |

Можна побачити що повнота постійно зростає, а точність то падає (з однаковими значеннями повноти), то зростає, коли повнота зростає. Цей характер можна побачити на графіку точності та повноти (рис 2.12).

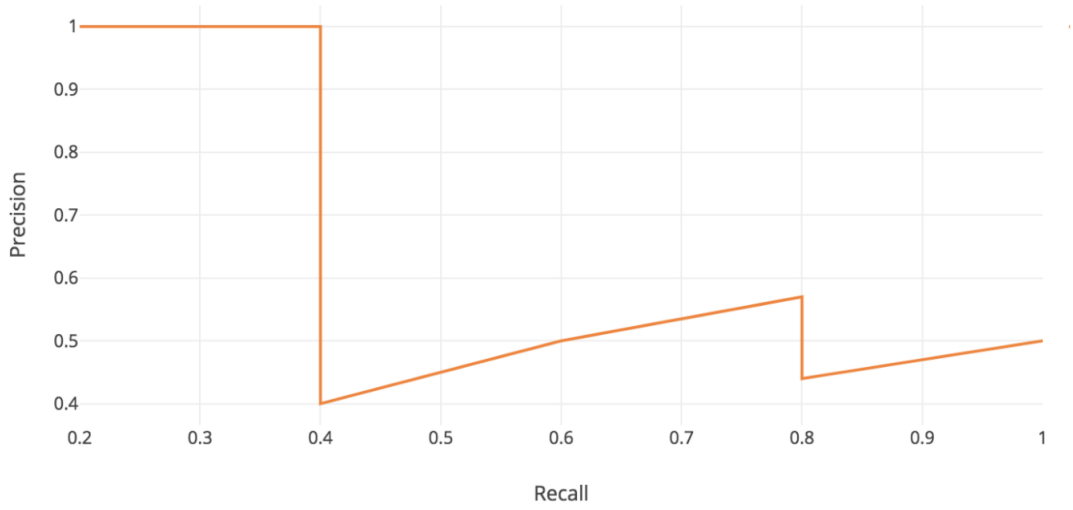


Рисунок 2.12. Графік точності та повноти

Основне визначення AP – це знайти площу під кривою точності та повноти, математично (2.12):

$$AP = \int_0^1 p(r)dr \tag{2.12}$$

Але для того щоб можна було швидко рахувати, використовують дещо інтерпольований графік, тобто згладжують його таким чином що для кожного рівня повноти замінюється точність на максимальне значення справа. Таким чином точність не зростає на будь якому значенні графіку. І новий графік виглядає так (рис 2.13).

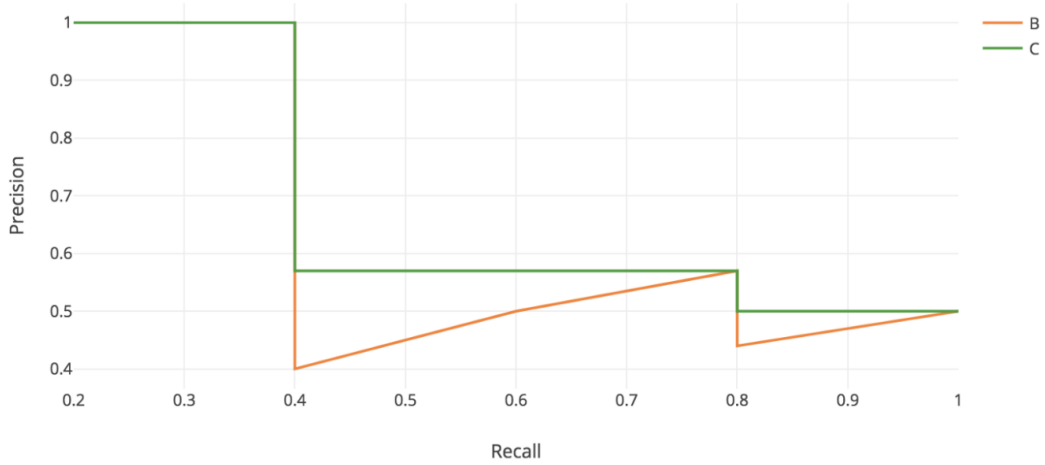


Рисунок 2.13. Інтерпольований графік точності та повноти

Після чого, як запропоновано в [24] рахується середнє з 11 точок на інтерпольованому графіку. Ці 11 рівномірно ділять проміжок повноти, тобто від 0 до 1. Таким чином ці точки – це 0, 0.1, 0.2, … 0.9, 1.0. Ці точки можна побачити на рисунку 2.14.

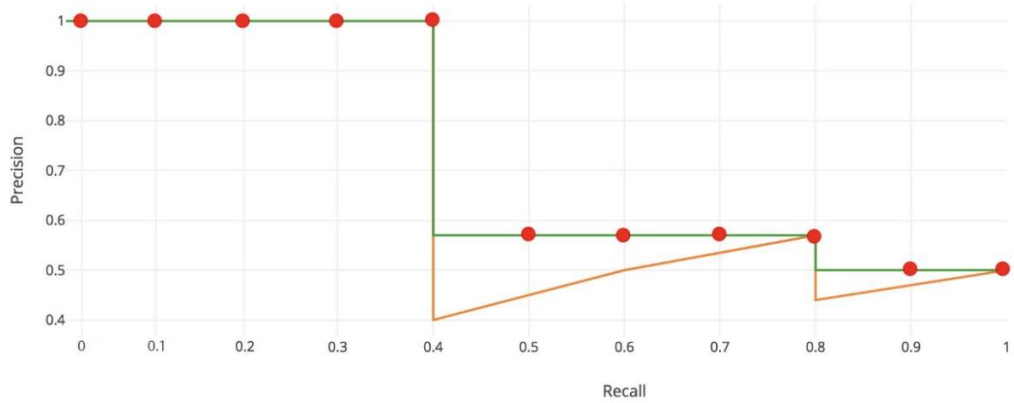


Рисунок 2.14. Знаходження 11 точок на кривій точності та повноти

Далі знаходиться середнє з оцих максимальних значень, і це і є AP (2.13)

$$AP = \frac{1}{11} * \left(AP\ AP_r(0) + AP_r(0.1) + \cdots + AP_r(1.0)\right) = \frac{1}{11} \sum_{r \in \{0.0,\ldots,1.0\}} AP_r \qquad (2.13)$$

Після чого таке ж AP рахується для всіх класів, та усереднюється, тому фінальна метрика називається mAP – mean AP. Її я і буду використовувати для порівняння своїх експериментів.

## 2.2 Обробка даних для навчання моделі детекції людей без масок

Для того щоб можна було навчати модель глибокого навчання потрібен певний набір даних з розміткою. Для своєї задачі я використав датасет з [15]. Він складається з двох типів даних: обличчя з масками та обличчя без масок. Нормальні дані, тобто обличчя без масок були взяті з WIDER FACE[25] датасету. А ті що мають маску, також можна поділити на дві категорії: ті які були отримані натуральним шляхом, тобто сфотографовано людину у масці, це датасети MAFA, FMFD, MMD та ті які отримані шляхом створення штучних масок (SMFD). На рисунках 2.15-2.17 зображено приклади з різних датасетів.

Рисунок 2.15 Приклад з WIDER FACE

Рисунок 2.16 Приклад з FMFD

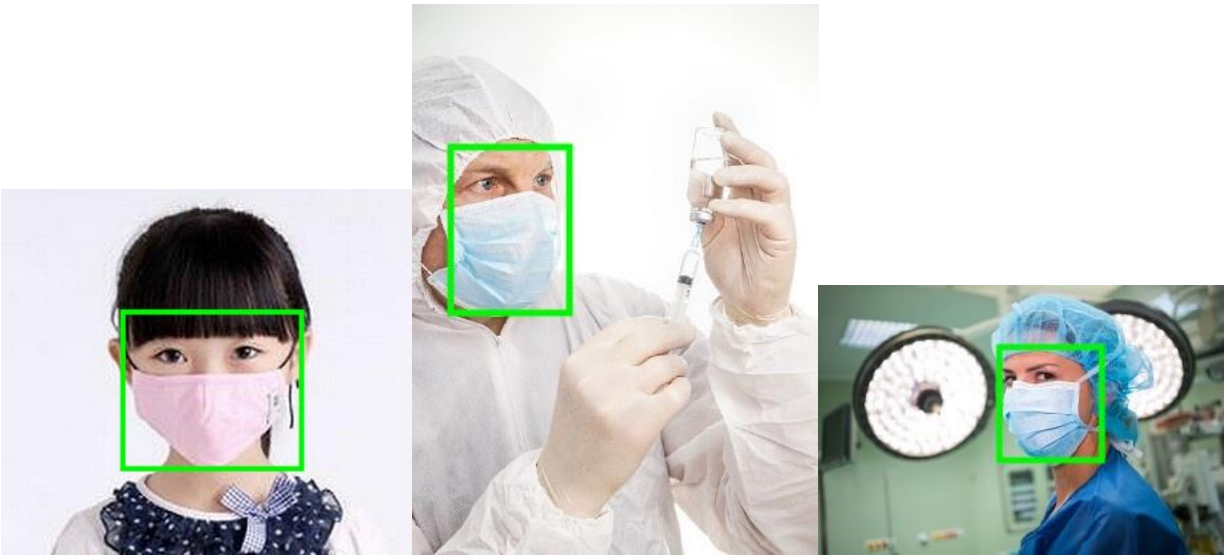

Рисунок 2.17 Приклад з MAFA

MAFA складається із зображень облич з різними обличчами, неоднорідними масками, фоном. RMFD складається із зображень тільки без фону. Оскільки зображення з фоном більш схожі на ті які будуть використовуватись в реальному житті, то було вирішено їх використовувати для навчання. Оскільки в MAFA знаходяться тільки регіони обличчя нижче брів, а у WIDER FACE розмічено повністю

обличчя, то, для того щоб розмітка була однакова прямокутники з WIDER FACE були дещо зменшені.

Як результат вдалось отримати з датасету MAFA 4065 картинок, з WIDER FACE – 3894 картинок, а також додано 1138 випадкових картинок з інтернету, таким чином отримали датасет що складається з 9097 картинок на яких розмічено 17532 прямокутника.

Для правильного тренування системи знаходження об’єктів датасет потрібно побити на дві частини та зафіксувати його. Це потрібно для того, щоб на одній частині тренувати мережу, а на іншій тільки перевіряти якість роботи. Ці дві частини називаються тренувальною та ваідаційною частиною. В моєму випадку було поділено з пропорцією 80% тренувальні дані, та 20% валідаційні дані.

Важливим етапом в роботі з даними є аугментація. Це спеціальний процес в якому дані деформуються різним чином перед навчанням. Це зроблено для того щоб урізноманітнити тренувальні дані. Серед популярний аугментації можна відмітити зміну кольору, додавання певного шуму на фото, обрізування чи перевернення фото та інші. У своїй роботі я використовував різні варіації афінних перетворень, поворотів, та зміни кольору. На рисунку 2.18 наведено приклад моїх аугментацій. Таким чином з однієї картинки можна отримати декілька, і під час навчання ця різноманітність приведе до кращого сходження мережі.

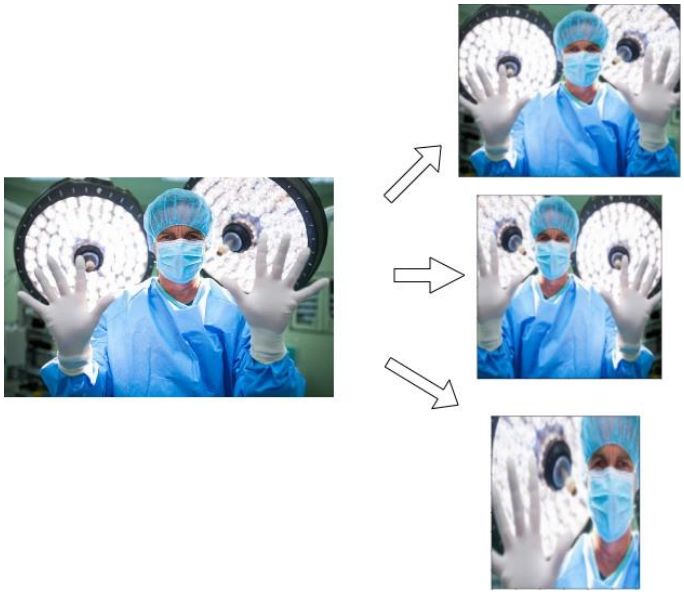

Рисунок 2.18. Приклад аугментацій

Ще одною особливістю роботи з даними для детектора об’єктів є те що для навчання та валідації треба подавати картинки однакового розміру. Це пов’язано з тим що під нас навчання при використання групування фото (batching) при проході фото потрібно групувати, а при різних розмірах фотографій у одному батчі це не можливо зробити. Є багато способів як можна досягти фотографій однакового розміру під час подачі на мережу. Я обрав такий: для збереження пропорційності фотографії (а це важливо для задачі саме знаходження облич, оскільки всі обличчя схожі за пропорціями один на одного) я змінював розмір таким чином, щоб більша сторона зображення дорівнювала 320, а інша рахувалась пропорційно до оригінального розміру. А рамки які утворились внаслідок такого перетворення я заповнював сірим кольором. Приклад такої картинки на рисунку 2.19.

Рисунок 2.19. Приклад подавання картинки на мережу

## 2.3 Висновки

В цьому розділі описано саму систему детектування людей без ЗІЗ та ключову частину цієї системи – модель детектування. В якості моделі було розглянуто один з найкращих методів детектування облич – Dual Shot Face Detecor, в пункті 2.1 було описано її структуру та особливості. Повну структуру мережі надано в додатках.

Було описано використання різних енкодерів з якими я планую експериментувати – це популярні мережі Vgg16, ResNet50, MobileNetV2 – вони

відрізняються якістю та швидкістю виконання. Розглянуто важливий модуль Features Enhance Module – за допомогою цього модуля відбувається покращення оригінальних ознак, і вони стають кращі для знаходження облич. І саме завдяки ньому ця мережа може знаходити малі обличчя. Описано алгоритм NMS який використовується тільки під час інференсу моделі, та допомагає покращити результат, за допомогою нього система не знаходить дублюючі передбачення.

Сформульовано те яким чином рахується помилка та чим вона відрізняється від класичного підходу, а саме те, що для знаходження помилки використовуються різні масштаби ознак, та різні згенеровані якорі. Також було зроблено опис алгоритму генерування якорів та їх відповідність до істинних значень. Наведено приклади реалізації деяких якорів. Важливим компонентом для проведення експериментом є правильний підрахунок метрики, та її кореляція з реальними результатами, тому в 2.1.6 Описано метрику за якою будуть порівнюватись експерименти – mAP при IOU = 0.5.

В пункті 2.2 розглянуто одну з найважливіших складових навчання – підготовку даних для навчання. Тут описано дані які використовуються, описано їхні особливості та наведено приклади з різних датасетів. Описано яким чином я буду використовувати аугментації в своїх дослідженнях, та яким чином реалізовано подавання картинки в мережу.

# 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

## 3.1 Вибір засобів розробки

Система автоматизованого виявлення людей без засобів ідивідуального захисту повністю написана на мові програмування Python.

У своїй роботі я використовував бібліотеку машинного навчання tensorflow першої версії. Вона, як і всі інші фреймворки для навчання, має набір реалізованих шарів з яких будується сама мережа, це шари згортки, пулінгу, нормалізації, функції активації та інші. Крім неї для коректної роботи я використовую ще бібліотеку матиматичних досліджень numpy, бібліотеку для роботи із зображеннями opencv. В якості системи логування було обрано сервіс Neptune, в нього можна з легкістю писати всі артефакти які породжуються під час тренування моделі. Його зручно використовувати, оскільки всі експерименти доступні в хмарі, і це полегшує роботу під час тренування на більше як одному сервері.

У якості сервера на якому я проводив експерименти було обрано компютер з Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz та NVidia GeForce RTX 2080 Ti. З встановленою операційною системою ubuntu 18.04.

## 3.2 Архітектура системи проведення експериментів та її модулі

Для того щоб можна було зручно та зрозуміло проводити експерименти, я створив спеціальну систему для проведення експериментів. Вона складається зі складових: модель, зчитувач даних, обчислення помилки, підрахунок метрики та тренер який об’єднує всі ці елементи в собі. За допомогою конфігураційного файлу я буду об’єднувати між собою всі елементи. Розглянемо кожен з них детальніше.

### 3.2.1 Модель

У своїй системі для тренувань я пропоную використовувати наслідування в створені моделей. Тому універсальні частини моделі винесу в окремий клас BaseModel а всі моделі буду унаслідувати від нього. У BaseModel я виніс такий функціонал:

- конструктор у який передаються всі змінні елементи які має знати модель: розмір входу картинки, сесія у якій проходить навчання, використання перетренованого енкодера та шлях до ваг, коефіцієнт з яким виконується регуляризація ваг (weight_decay);
- зберігання та зчитування моделі;
- реалізовано «заморожування» моделі в кінці тренування, це потрібно для того щоб вже готову моделі можна було використовувати в системі;
- створення placeholder для входів та виходів моделі. Для tensorflow 1 потрібно створювати спеціальний елемент який називається placeholder для правильного навчання. Окремим placeholder буде інформація про те чи зараз йде цикл тренування чи валідації (це важливо знати, оскільки під час валідації потрібно відключати додатковий програхунок у нормалізаційних шарах);
- абстрактні методи для створені самої моделі та для зчитування ваг енкодера;
- реалізовано алгоритм NMS, оскільки він є однаковим для всіх моделей.

Тому у конкретних моделей потрібно буде реалізувати тільки саму модель, тобто послідовність шарів яку проходить зображення. А саме в мене будуть дві реалізації моделей: модель для одного класу, та модель для двох.

Окремо хочу відзначити роботу з енкодерами. Вони реалізовані схожим принципом як і моделі, є базовий енкодер, в ньому реалізовано спільний метод – перевірка та зчитування ваг перед тренуванням. А в самих енкодерах додатково реалізовано:

- самі шари енкодера;
- додавання додаткових шарів для енкодера в якому не вистачає виходів для шести оригінальних ознак (це описано в 2.1.1);
- модуль FEM, який реалізований для кожної модель окремо, оскільки має різницю в кількості каналів залежно від енкодера.

В моєму випадку, згідно 2.1.1, я реалізував такі енкодери: mobilenetv2, resnet50, vgg16. Загальна ж діаграма класів модуля представлена в додатку В.

### 3.2.2 Зчитувач даних

Зчитувач даних реалізований схожим чином як і модель. Є клас BaseDataloader і в ньому відбуваються майже всі перетворення, оскільки під час обробки даних вони універсальні. Тому в цьому класі реалізований такий функціонал:

- в конструкторі задається інформація про дані: шлях, розмір батчу, розмір виходу картинки, наявність аугментацій, також задається кількість потоків, та розмір кешованого розміру. Два останніх параметри потрібні для того щоб можна було запускати дану систему на різних ресурсах: якщо буде дуже потужна GPU то потрібно більше потоків і більший розмір кешованого розміру ніж зазвичай (в системах з потужною відеокартою трапляються затримки в подачі даних через те що ітерація проходить занадто швидко), і навпаки для економії надлишкових ресурсів при запуску на слабшому комп’ютері;
- сам процесинг даних, а саме зчитування, аугментацію, зміна розміру, нормалізація даних. Оскільки я використовую однаковий формат з даними, то ці всі процеси є однаковими;
- окремо треба відмітити аугментацію. Вона реалізована як абстрактний метод, а його реалізація написана в нащадках класу.

Оскільки в моїх експериментах використовується два набори даних: один для навчання з одним класом, а інших для навчання з обома. То і нащадків класу BaseDataloader повинно бути два: для одного класу, та для двох. Але не варто забувати про те що існує два датасети: тренувальний та валідаційний. Для кожного з них буде використовуватись свій нащадок. Важливо зауважити що різниця між тренувальним та валідаційним є принципова: у валідаційному не можна робити аугментацію. Таким чином у моїй системі буде 4 нащадки: Dataloader_train_1_class, Dataloader_val_1_class, : Dataloader_train_2_class, : Dataloader_val_2_class.

Для того щоб було легко використовувати аугментації під час навчання, та комбінувати їх між собою, я використав принцип поліморфізму. Тобто в мене всі аугментації є наслідниками одного класу і мають спільний інтерфейс. Тому в конфігураційному файлі я задавав просто імена аугментації, а для їх виклику я просто

проходився по об’єктам цих класів та викликав функцію aug(). Реалізація такого представлена на рисунку 3.1.

```
def aug(self, image, box):

    for augmetation in self.list_aug:
        image, box = augmetation.aug(image, box)

    return image, box
```

Рисунок 3.1. Як працює виклик аугментації

Загальна діаграма класів представлена на додатку Г.

Важливою деталлю роботи самого зчитувача є правильний формат даних який йому подається. В мене це реалізовано так:

– для картинок зроблена окрема папка, де знаходяться картинки як тренувальної так і валідаційної частини;
– для розуміння які картинки до якої частини датасету відносяться створено два файла train.txt і val.txt. В ньому представлені шляхи до картинок та самі анотації;
– в цих же .txt файлах представлені прямокутники у вигляді координат верхнього лівого кута та нижнього правого. Також числами 1 або 2 позначено до якого класу відноситься той чи інший прямокутник. Де клас 1 це людина без маски, а кдас 2 – людина з маскою.

В загальному формат такий: шлях до картинки, знак |, а далі йде перелік по 5 чисел: x1,y1,x2,y2 та клас до якого відноситься прямокутник. Якщо на фото є більше одного прямокутника, то таких блоків з 5 чисел декілька, вони розділені пробілом. Детальніше схема анотаційного файлу представлена на рисунку 3.2.


Рисунок 3.2. Схема анотаційного файлу

### 3.2.3 Обчислення помилки та метрики

В моїй мережі існує два типи помилок: класифікаційна та регресійна. Проте залежно від кількості класів які будуть пі час тренування, буде відрізнятись класифікаційна помилку. Також слід зауважити, що для тренування і знаходження градієнта для зворотного проходу мене цікавить тільки загальна помилка (total_loss). Тому в своїй системі я реалізував окрему функцію знаходження загальної помилки. Вона створюється з реалізує такі складові:

- знаходження регресійної помилки для малих та великих облич. В мене регресійна помилка рахується за одним алгоритмом, тому логічно її реалізувати один раз в total_loss;
- ініціалізація та виклик класифікаційної помилки залежно від кількості класів;
- додавання всіх помилок з коефіцієнтами які взяти з конфігураційного файлу.

Знаходження ж класифікайної помилки реалізовано так, що існує два класи які унаслідуванні від спільного класу ClaLoss. Один з цих класів реалізує помилку для одного класу, а інший для двох. На рисунку 3.3 зображено діаграму класів для класифікаційної помилки. В додатку Е нарисована схема алгоритму знаходження помилки.

Для знаходження метрики потрібно рахувати метрику для одного класу, метрику для двох класів і середнє між ними. Тому для знаходження метрики реалізована схожа система, проте у якості спільного класу виступає IOU на першому класі, а IOU на двох класах використовує вже підрахований результат на одному,

рахує результат на другому, та усереднює результати. Діаграма цих класів представлена на рисунку 3.4.

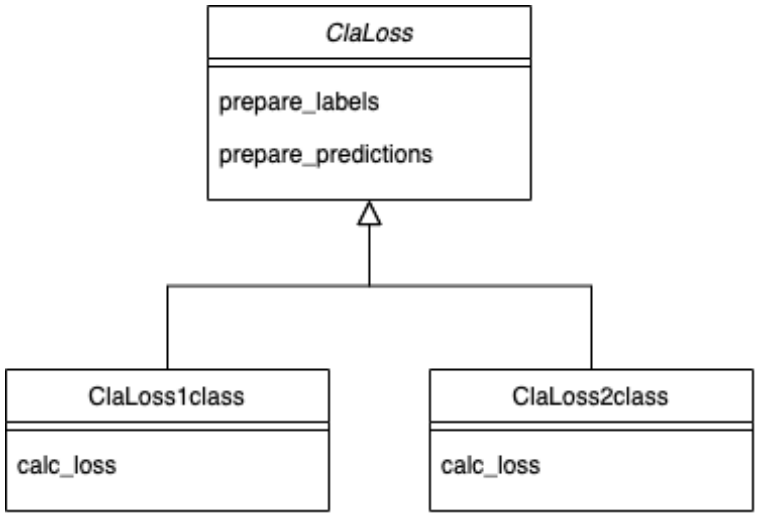


Рисунок 3.3. Діаграма класів для пошуку класифікаційної помилки

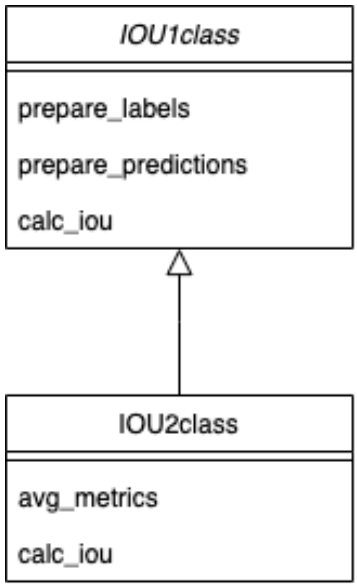


Рисунок 3.4. Діаграм класів для підрахунку метрик

### 3.2.4 Тренер

Ця частина системи об’єднує всі модулі які описані вище. Це є точкою входу в систему навчання, і тут відбувається саме навчання моделі. Сам алгоритм тренування наступний:

1) зчитується конфігураційний файл, який представлений як python файл з словниками які мають інформацію щодо тренування. В конфігураційному файлі крім аргументів для налаштування модулів, додатково описано кількість епох навчання, шлях для збереження моделей;
2) через використання tensorflow 1 потрібно завжди використовувати сесію для навчання, і в ній рахувати все. Тому сесія створюється саме тут;
3) маючи сесію, ініціалізуються всі модулі навчання: модель, зчитувач даних, обчислювач помилки та обчислювач метрики;
4) додатково створюються модулі які є невеликими але теж потрібні для навчання – це алгоритм оптимізації, алгоритм зміни кроку навчання та сам крок навчання, система логування;
5) після ініціалізація проводиться саме навчання. Воно складається з двох кроків які повторюються n разів, де n це кількість епох в конфігураційному файлі;
6) перший крок, це безпосередньо тренування. Воно проводиться k кроків, де $k = \left\lfloor \frac{dataset_{size}}{batch_{size}} \right\rfloor$. Під час кожного кроку:
   – через модель поганяється батч з картинками;
   – рахується загальна помилка для кроку;
   – рахується метрика для кроку;
   – відбувається розрахунок градієнтів по помилці;
   – обновляються ваги залежно від градієнтів;
7) після тренування, рахуються середні значення помилок та метрик, та відправляються в систему логування;
8) другий крок, це валідація. Вона проводиться так само як і тренування, проте відсутні кроки рахування градієнтів та оновлення ваг;
9) після валідації, рахуються середні значення помилок та метрик для валідації, та відправляються в систему логування;
10) якщо загальна помилка для валідації є меншою ніж всі інші кроки перед цим, то ця ітерації моделі вважається найкращою та зберігається;
11) після закінчення всього навчання. Відбувається замороження найкращої моделі.

Загальна схема тренування представлена в додатку Д.

Для тренування моделі я використовував розмір картинки 320x320 пікселів. Для навчання підрахував найбільший batch_size який помістився у відео пам’ять мого серверу – це 16. В якості optimizer я використовував momentum зі сталим кроком навчання $lr = 0.0001$.

## 3.3 Система знаходження людей без ЗІЗ

Як було описано в попередніх розділах для того щоб знайти людину без ЗІЗ буде використовуватись зображення. Воно отримуватиметься з потоку відео. Відео в свою чергу буде розбиватися на кадри. Після чого ці зображення будуть подаватись на вхід нейронній мережі і на виході отримуватимемо bounding boxes облич людей, які не користуються ЗІЗ. На одному фреймі може одночасно перебувати більше однієї людини, і детектор буде знаходити всіх людей без ЗІЗ, що є у кадрі. Для знаходження людей без масок можна як і знаходити, так і не знаходити людей які носять маску. На рисунку 3.5 зображена структурна схема використання такої системи.

Тобто спочатку відео б’ється на кадри, після чого кожен кадр послідовно обробляється нейронною мережею, нейронна мережа видає ймовірності та класи та баундіг бокси, після чого, при умові ймовірності більше 0.5 на класі людини без маски спрацьовує сигнал інформування про наявність людини без маски. Проте сама нейронна мережа може працювати не так швидко, як йде відео потік, тому будемо брати наступний кадр не зі всіх в черзі а той який в цей момент знімає камера. Тому другою програмою в цій роботі є сама програма яка і виконує знаходження людей без ЗІЗ.

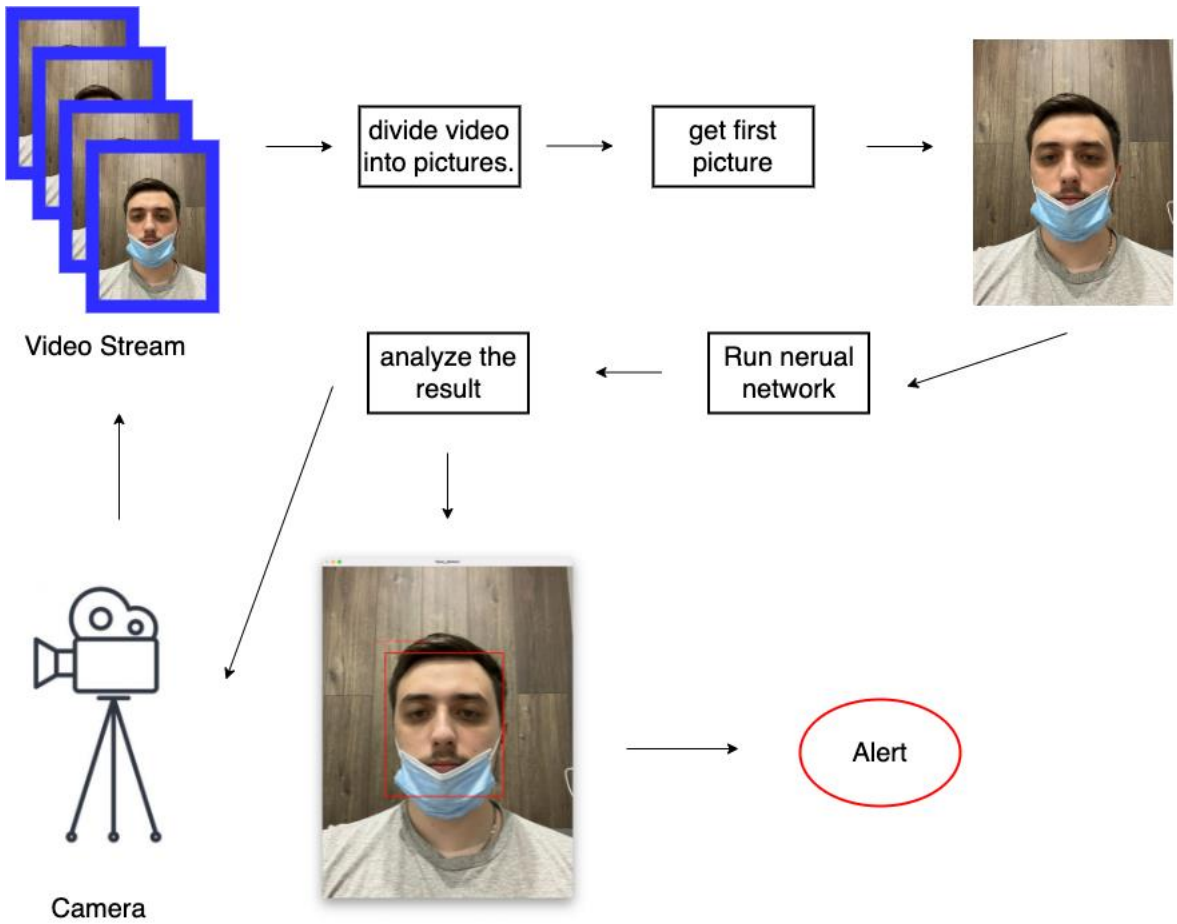


Рисунок 3.5  Структурна схема знаходження людей без масок

Друга програмна реалізація це безпосередньо використання навчених моделей безпосередньо для знаходження людей без маски.

```
def freeze_graph(graph, output_node_names, output_graph):
    sess = tf.Session(graph=graph)

    output_graph_def = tf.graph_util.convert_variables_to_constants(
        sess,  # The session is used to retrieve the weights
        g.as_graph_def(),  # The graph_def is used to retrieve the nodes
        output_node_names.split(","),
    )
    with tf.gfile.GFile(output_graph, "wb") as f:
        f.write(output_graph_def.SerializeToString())
    print("%d ops in the final graph." % len(output_graph_def.node))
```

Рисунок 3.6. Функція заморожування моделі

Для того щоб програма працювала, потрібно передати модель із першої програми в другу. Це робиться за допомогою заморожування моделі. Також в процес заморозки моделі включається перетворення шарів нормалізації (наприклад BatchNorm) у такий вигляд що є придатним тільки для інференсу. Тобто процес знаходження середнього та відхилення не знаходить в батчі а береться з ваг. Ще одною перевагою заморожених моделей є певна оптимізація, під час заморожування моделі, шари згортки нормалізації та пулінгу об'єднуються в один і працюють дещо швидше. За допомогою tensorflow заморозити модель нескладно, функція заморожування представлена на рисунку 3.6.

Після чого таку модель легко зчитати та використати. Це також відбувається завдяки стандартним операціям в tensorflow. Це та отримання входів та виходів представлено у функції на рисунку 3.7.

```
def load_graph(frozen_graph_filename):
    # We load the protobuf file from the disk and parse it to retrieve the
    # unserialized graph_def
    with tf.gfile.GFile(frozen_graph_filename, "rb") as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())

    # Then, we import the graph_def into a new Graph and returns it
    with tf.Graph().as_default() as graph:
        # The name var will prefix every op/nodes in your graph
        # Since we load everything in a new graph, this is not needed
        tf.import_graph_def(graph_def, name="prefix")

    x = graph.get_tensor_by_name('prefix/tower_0/images:0')
    training = graph.get_tensor_by_name('prefix/training_flag:0')
    y = graph.get_tensor_by_name('prefix/tower_0/boxes:0')
    y_sc = graph.get_tensor_by_name('prefix/tower_0/scores:0')
    y_lbl = graph.get_tensor_by_name('prefix/tower_0/labels:0')
    return x, training, y, y_sc, y_lbl
```

Рисунок 3.7. Функція зчитування та отримання входів та виходів моделі

У моїй програмній дана система знаходження людей без ЗІЗ працює тільки з веб камерою комп’ютера та запускається командою:

```
python freeze_run_camera.py --model <path> -d <device>
```

Тут `path` – це шлях до моделі яка буде використовуватись, а `device` – вказується девайс на якому запускати інференс моделі, може приймати значення CPU або GPU, CPU стоїть по замовчуванню. Для того щоб запускати на GPU потрібно мати встановлену бібліотеку CUDA.

Після запуску, з’являється вікно програми яке транслює все що є на відеокамері. Проте якщо в кадрі буде людина з маскою чи без, програма проінформує про це користувача. Інформування відбувається за рахунок обведення облич в прямокутники червоного або зеленого кольору. Червоний означає, що людина без маски, або маска одягнута не вірно, а зелений, це те що людина знаходиться в масці. Також додатково над прямокутником з’являється надпис Mask чи No Mask, що полегшує нативне сприйняття результатів. Біля надпису також з’являється число – це ймовірність передбачення. Воно потрібне для додаткового інформування користувача програми. Приклад роботи програми зображений на рисунку 3.8, більше прикладів та різних застосувань представлено в розділі 4.

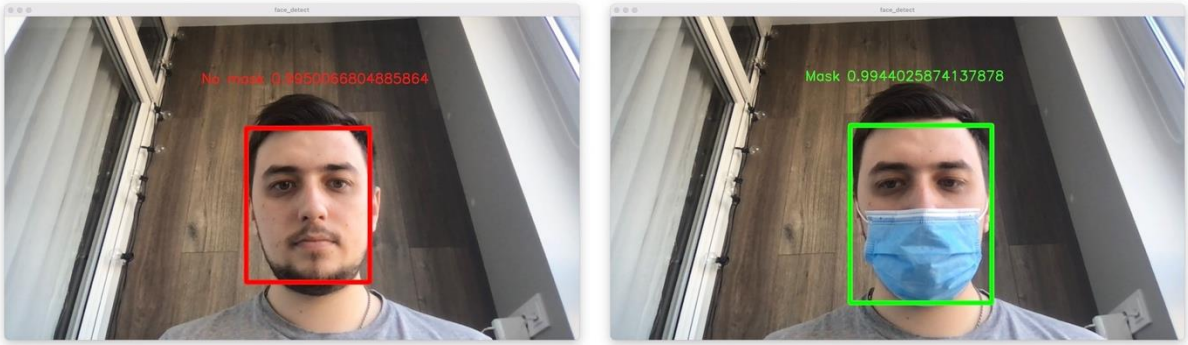


Рисунок 3.8 Приклад застосування програми

Загальний алгоритм роботи даної програми зображений в додатку Ж. Розглянемо його детальніше:

1) береться перший кадр з відеопотоку за допомогою cv2.VideoCapture(0).read();

2) відбувається препроцесинг картинки, тобто вона перетворюється в такий формат на якому вчилась мережа. Детальніше про нього в 2.4;
3) відбувається інференс моделі машинного навчання;
4) отримуються результати передбачення, вони відсіюються по критерію: ймовірність передбачення > 0.5;
5) постпроцесинг результатів. Тут прямокутники які були отримані на зображенні з рамками по боками, перетворюються в такі які можна накласти на реальний розмір картинки;
6) відбувається малювання прямокутника на обличчі та пишеться текст над прямокутником;
7) картинка в головному вікні замінюється на картинку вже з прямокутниками, та залишається поки не виконається наступна ітерація.

## 3.4 Висновок

В цьому розділі було розглянуто програмне забезпечення цієї дисертації. В 3.1 було розглянуто ресурси та бібліотеки які я використовував під роботи. Це стандартні бібліотеки машинного навчання та потужний комп’ютер на якому я проводив експерименти із навчання мереж.

В 3.2 розглянуто систему яку я використовував під час навчання своїх експериментів. Система доволі гнучка, та дозволяє з легкістю використовувати її і для інших експериментів такого ж типу. Вона складається з таких складових: модель (вона реалізована так, що можна легко використовувати модель як з одним класом, так і з двома, а також можна використовувати різні енкодери: Vgg16, Mobilenetv2, Resnet50); зчитувач даних (в якому реалізовано датасет як на один, так і на два класи, а також методи аугментації), обчислювач помилки та метрики, а також сам тренер який включає в себе всі ці елементи. Представлено формат даних для датасету. Для кожної складової представлено діаграму класів, а також представлено повний алгоритм навчання експерименту. За допомогою цих елементів можна гнучко експериментувати з різними параметрами моделі.

В 3.3 представлено метод заморожування моделі та подальшого зчитування моделі. Представлено реалізацію на tensorflow. Також в цьому підрозділі розглянуто другу програму – систему знаходження людей без ЗІЗ. Ця система працює на будь-якому комп’ютері на який можна встановити python з потрібними бібліотеками. Вона працює з веб камерою комп’ютера, а за потреби може бути модифікована для зовнішньої камери. Система працює у режимі реального часу та легка в користуванні. Для того щоб інформувати користувача про наявність обличчя без маки в кадрі, в системі передбачено графічне представлення результатів. Представлено алгоритм її роботи з детальною блок схемою алгоритму. Розглянуто інтерфейс та показано приклад використання.

# 4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ

## 4.1 Планування експериментального дослідження

В результаті базової моделі для своїх експериментів було обрано модель з енкодером MobileNetV2, оскільки це доволі швидка та якісна мережа, тому певний бейзлайн можна отримати не потративши багато ресурсів на обчислення. Оскільки я вирішую задачу знаходження людей без масок, то логічно буде навчати мережу детектувати саме людей без масок. Тому базовою моделлю буде модель, навчена на одному класі і з MobileNetV2.

Наступним експериментом є сенс спробувати метод transfer learning. Суть цього методу полягає в тому, щоб почати процес навчання не з просто випадкової ініціалізації ваг, а з певного шаблону, який краще за все підходить до задачі. У задачах комп'ютерного зору такий підхід застосовується у більшості випадків методом застосування вже переднавчених моделей. А, оскільки, майже у всіх задачах використовується якийсь відомий енкодер, який вже є навчений на відомому датасеті для задачі класифікації, наприклад ImageNet[26], то найпростішим способом використати transfer learning є використання вже натренованих ваг екнодера. В даному випадку енкодер мережі вже буде мати інформацію про навколишній світ, а саме де гіпотетично можуть знаходитись люди, адже клас людина в ImageNet присутній. Єдиною проблемою є те що під час розробки датасету ImageNet засоби індивідуального захисту не були поширені, тому є ймовірність того, що мережа покаже погані результати. Це і перевіриться під час цього експерименту.

Якщо ж в попередньому експерименті все вдасться, то я пропоную розглянути наступний. На мою думку просто навчання детектора на 1 класі «людина без маски» буде не таким продуктивним, як разом і з класом «людина в масці». Це пов'язано з тим, що під час навчання тільки на одному класі детектор не знатиме про існування впринципі людини з маскою. Це зумовлено тим що обличчя людини в масці та без неї дуже схожі: вони однакового розміру, мають очі, брови, вуха, знаходяться зверху класу людина, та багато інших спільних ознак. І єдиною різницею між ними це є наявність маски, що на фоні існування в природі дуже великої кількості класів в

принципі не є великою. Тому, на мою думку, мережа буде помилятися, і якщо показувати мережі людину в масці як окремий клас, то результат мав би бути кращий. Це і перевіряться шляхом другого експерименту.

Останнім ж експериментам я планую використати інші енкодери які я описував в 2.1.1. Це ResNet50 та Vgg16, оскільки такі енкодери використовувались і в інших системах детекції об’єктів. Ці енкодери мають показати кращий результат ніж MobileNet оскільки є складнішими і глибшими мережами. Проте вони будуть працювати довше, але на GPU така різниця буде не помітна. Тому останнім експериментом я спробую використати різні енкодери для свого детектора.

## 4.2 Результати експериментів з формалізацією задачі та transfer learning

Під час проводження експериментів я один раз поділив вибірку на дві частини: тренувальну та валідаційну. А також перед проведенням експериментів я підготував два окремих набори анотацій: один для експериментів з одним класом, інший – для експериментів на двох класах.

В результаті першої базової моделі вдалось отримати результати які були погані за якістю. Ця модель була з MobileNetv2 енкодером та працювала тільки на одному класі – людина без маски. В цьому експерименті я міряв IOU тільки на одному класі. Та отримав 0.6767 на валідаційному датасеті, а на тренувальному – 0.7108. Графіки навчання цієї моделі представлені на рисунках 4.1, 4.2, 4.3.

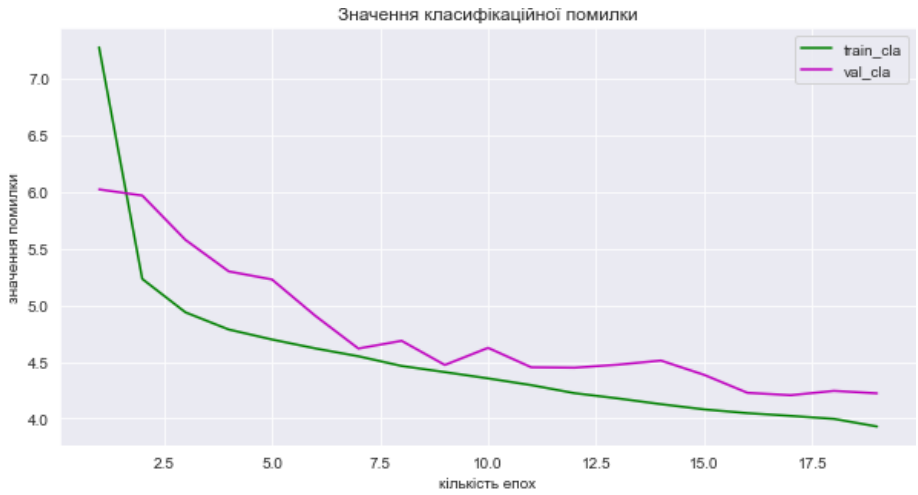


Рисунок 4.1 Графіки класифікаційної помилки навчання першого експерименту

Можна побачити, що значення класифікаційної помилки падає рівномірно на тренувальній частині, та не рівномірно на тестовій, одночасно з цим помилка падає не так стрімко. Це означає що мережа вчиться класифікувати, проте якість класифікації ще не найкраще.

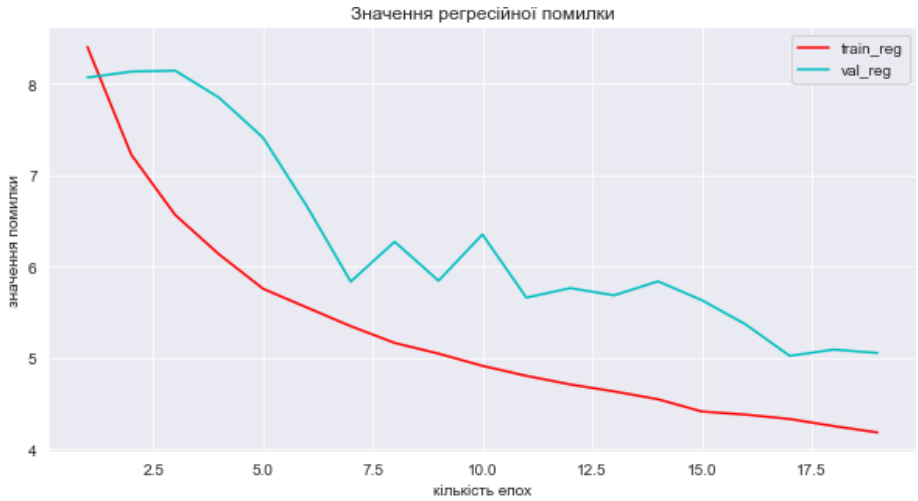


Рисунок 4.2 Графіки регресійної помилки навчання першого експерименту

Під час аналізу графіків регресійної помилки можна побачити, що тестовий графік падає дуже погано і не стабільно, це означає що знаходження безпосередньо прямокутника погане. Також якщо порівняти графіки 4.1 і 4.2 можна побачити, що піки тестової помилки на класифікації і регресії збігаються, тобто якщо мережа помиляється в правильній класифікації, то і помиляється в правильному розміщенні прямокутника. Це і підтверджується на графіку із загальною помилкою, яка включає в себе обидві помилки (рис 4.3).

Також з графіків можна побачити невелику різницю між тренувальними та валідаційними кривими, та те що ці криві падають повільно також означає що мережа не донавчилась, в цілому вона навчається проте є ще варіанти для покращення, які я перевірив в наступних експериментах.

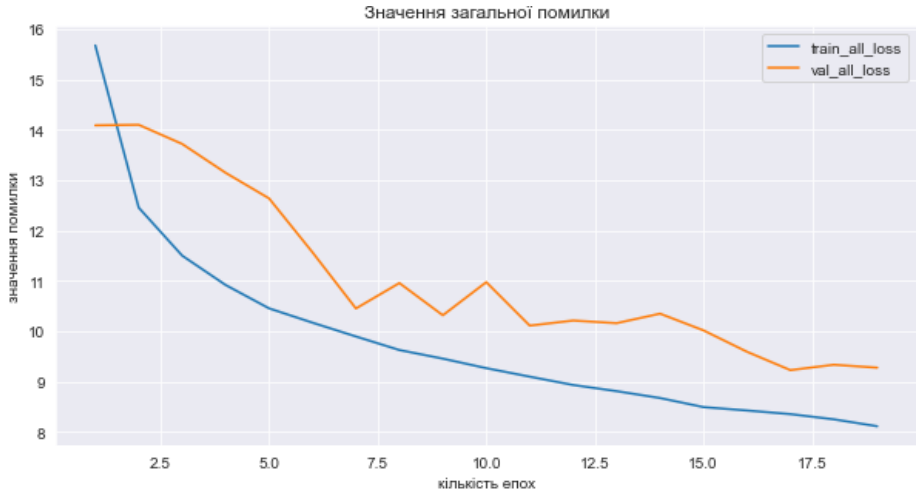


Рисунок 4.3 Графіки загальної помилки навчання першого експерименту

Наступним експериментом був експеримент в якому я застосував переднавчений на датасеті ImageNet енкодер. В результаті отримано наступні результати (рис 4.4-4.6).

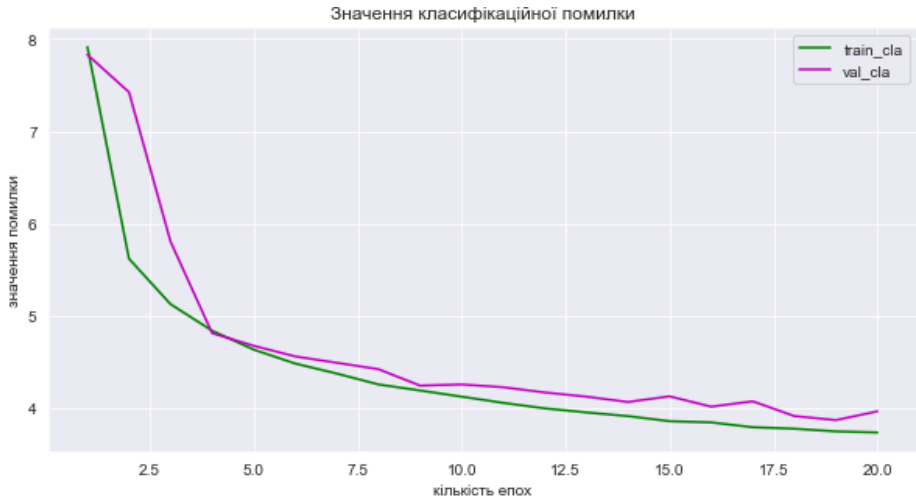


Рисунок 4.4 Графіки класифікаційної помилки навчання експерименту з передтернованими вагами

За графіками класифікаційної помилки можна сказати що мережа сходиться краще ніж у першому експерименті, проте можна помітити що початкові значення графіку більші, це зумовлено тим, що мережа на початку ініціалізації знала вже про існування багатьох класів, і спочатку не відрізняла їх між собою. Проте далі мережа зійшлась до кращогого локального мінімуму, що підтверджується нижчою

фінальною помилкою на валідаційній частині (блиько 4, а в першому експерименті близько 4,25).

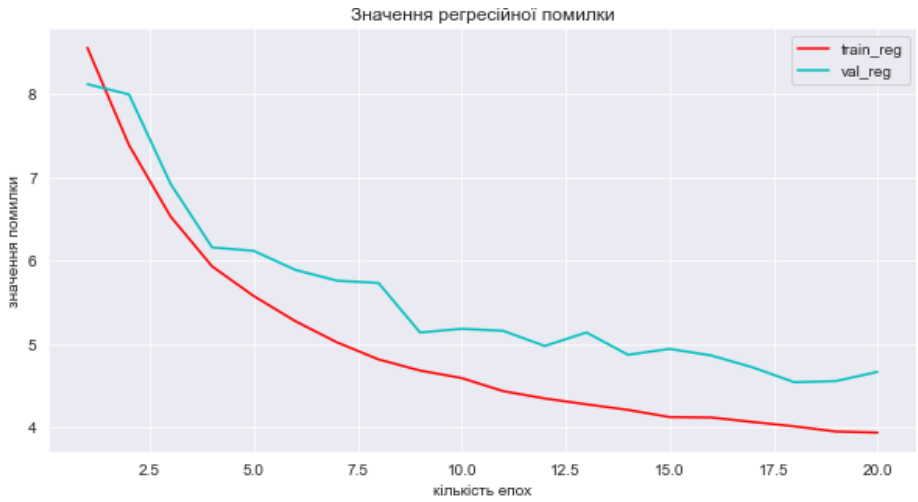


Рисунок 4.5 Графіки регресійної помилки навчання експерименту з передтернованими вагами

За графіком регресійної помилки теж можна помітити що мережа вчиться краще, і сходиться до меншого локального мінімуму. Можна також помітити кореляцію між регресійної і класифікаційною помилкою як і в першому експерименті, це ж можна помітити і на загальному графіку.

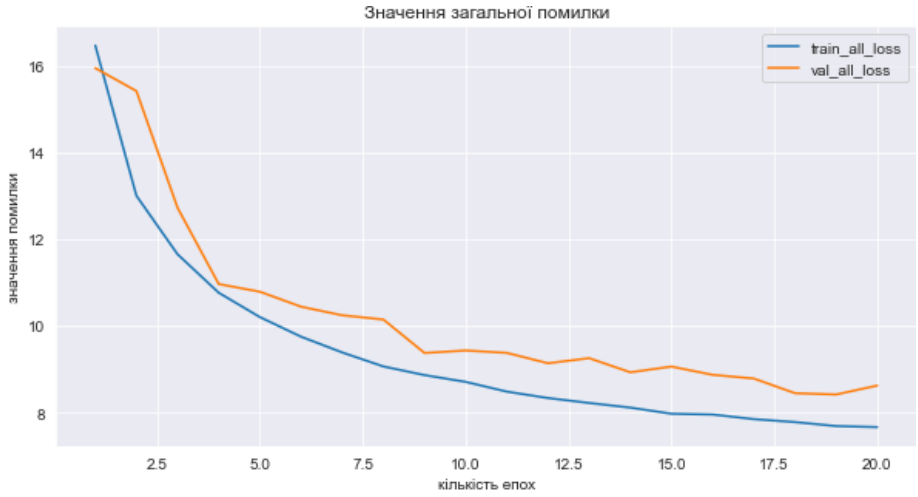


Рисунок 4.6 Графіки загальної помилки навчання експерименту з передтернованими вагами

Із загального графіка видно що модель вчилась значно краще. Інформація з ImageNet додала в якості передбачування. Фінальний результат на валідаційному датасеті – mAP = 0.7747, а на тренувальному – 0.8. Порівняно з попереднім виросло значення і тренувальної і валідаційної метрики. Тому в наступних експериментах я буду використовувати вже натренеровані ваги енкодера.

Наступним я екпериментував з постановкою задачі. Якщо ж вчити на декількох класах, а не на одному, то результат став кращий. Я використовував в цьому експерименті MobilenetV2 з натренованими вагами. Результати цього експерименту зображено на графіках (рис. 4.7-4.9)

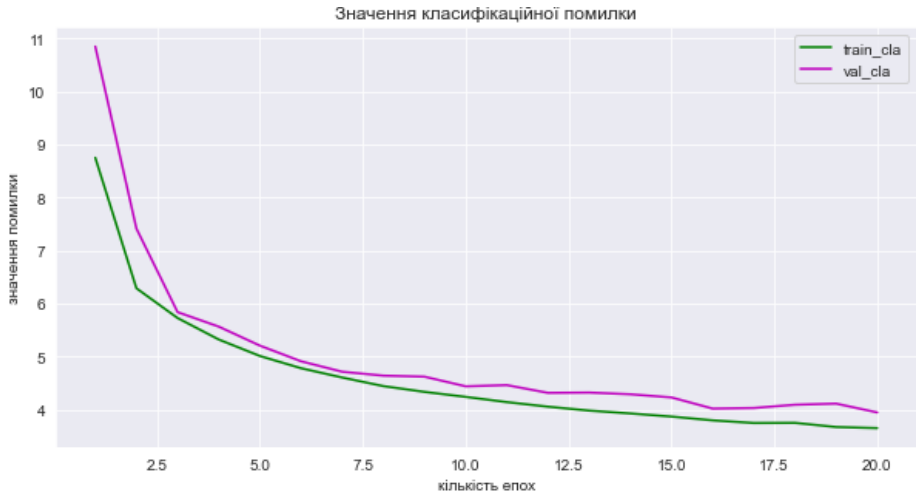


Рисунок 4.7 Графіки класифікаційної помилки навчання експерименту з використанням двох класів

Значення графіку дещо відрізняються від значень попередніх експериментів. Це пов’язано з тим що використовується два класи, і помилка для класифікації також рахується за двома класами. Особливо можна помітити стартові значення, що є більшими за навчання з одним класом, це пов’язано не тільки із збільшенням класів, а і з тим що загальна кількість об’єктів збільшилась. Проте сама помилка падає стрімкіше ніж у перших двох експериментах – це означає що модель навчається краще, і те що валідаційна помилка падає одночасно з тренувальною означає що модель не перенавчається.

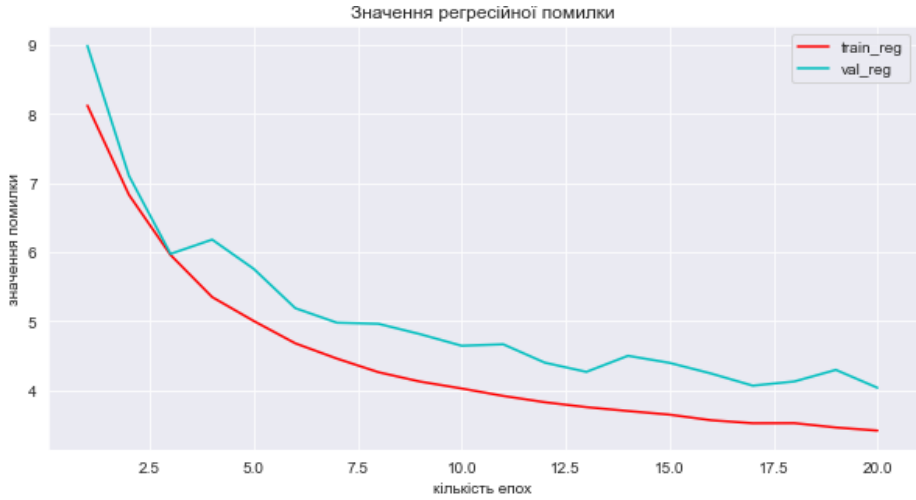


Рисунок 4.8 Графіки регресійної помилки навчання експерименту з використанням двох класів

На регресійній помилці також можна помітити вплив додавання ще одного класу. Загальна кількість об’єктів збільшилась, а отже і помилка також збільшилась, проте з графіку видно що помилка як на тренувальному так і на валідаційному датасеті падає.

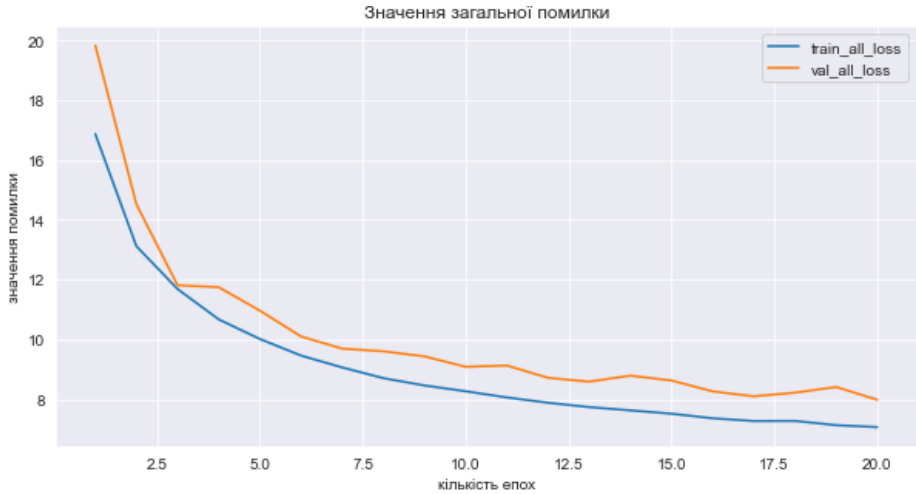


Рисунок 4.9 Графіки загальної помилки навчання експерименту з використанням двох класів

В цьому експерименті я міряв метрику окремо по двох класам, для того щоб можна було порівняти його з попередніми. В результаті, mAP на класі людина без маски на валідаційному датасеті – 0.7899, а на тренувальному – 0.8173. Значення

метрик більше ніж у другому експерименті, і сама різниця між тренувальною метрикою та валідаційною менша, що означає краще навчання даної мережі. Результат ж для людей в масці дуже добрий – 0.902 та 0.908 для валідаційних та тренувальних даних.

Рисунок 4.10 Візуалізація результатів перших трьох експериментів

На рисунку 4.10 представлено візуальні результати кожної з мереж. Тут в першому рядку знаходяться результати передбачення першого експерименту, у другому та третьому відповідно результати другого та третього експерименту. Всі фото які використовуються для показу роботи не брали участь у тренуванні, а були тільки на валідаціїї. Тут можна помітити кореляцію візуально кращих результатів з кращими метриками. Можна помітити що різниця між 1 і 2 експериментом більша ніж між 2 і 3, по метрикам така ж ситуація: різниця в метриці між 1 і 2 експериментом на валідаційній частині = 0.1, а між 2 і третім тільки 0.02. Це показує що метрика обрана правильна.

В цілому якщо порівняти ці три експерименти, то і по графікам навчання (рис 4.11), і по метрикам, і візуально (рис. 4.10) кращий третій експеримент.

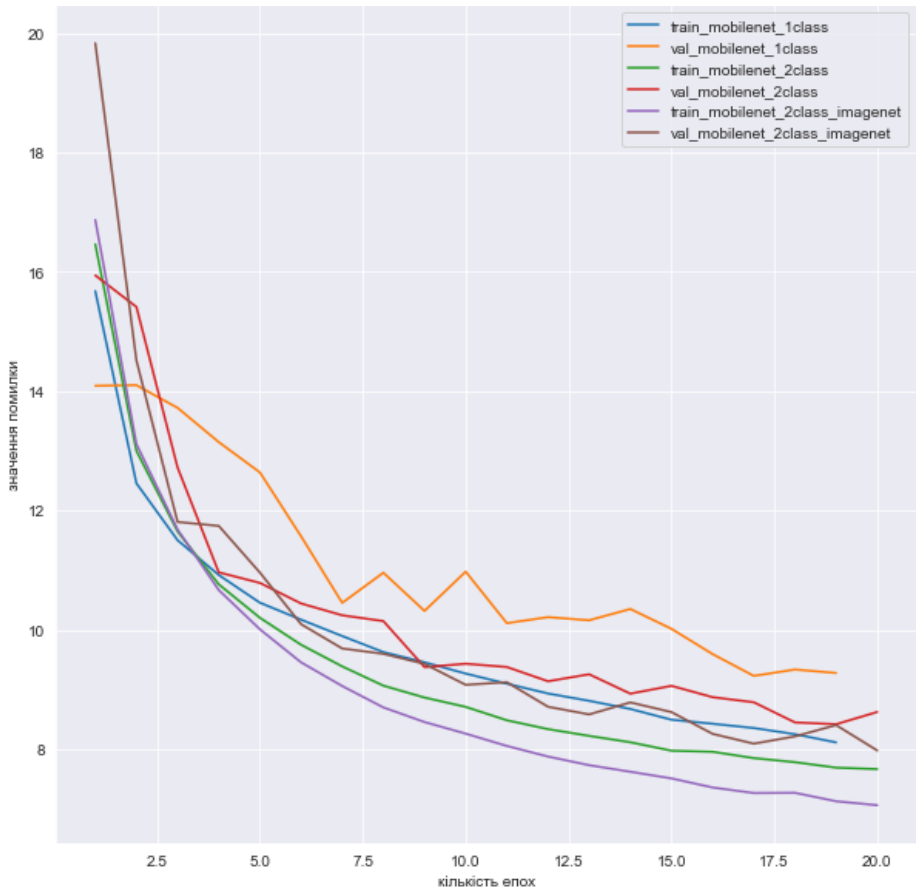


Рис 4.11 Графіки порівняння перших трьох експериментів

По рис 4.11 видно, що помилка на першому експерименті набагато більша ніж на другому та третьому, що підтверджує те що перша модель недонавчилась.

4.3 Результати експериментів з різними енкодерами

Оскільки швидкість та результативність мережі дуже сильно залежить від енкодера який використовується я вирішив їх порівняти між собою. Оскільки експеримент з двома класами дав найкращий результат, то я натренував експерименти з двома класами для таких популярних в задачі детекції енкодерів: mobilenetv2, resnet50, vgg16.

В рамках експериментів з різними експериментами я буду порівнювати результати з третім експериментом, оскільки в це найкращий варіант з тих що проводились раніше. Також порівнювати експерименти між собою я буду вже за загальним mAP а не тільки по одному класі. Для третього експерименту це будуть значення 0.8459 та 0.8630 для валідаційного та тренувального сету відповідно.

Спочатку я спробував використати ResNet50 в якості енкодера. Результати навчання представлені на рис. 4.12-4.14.

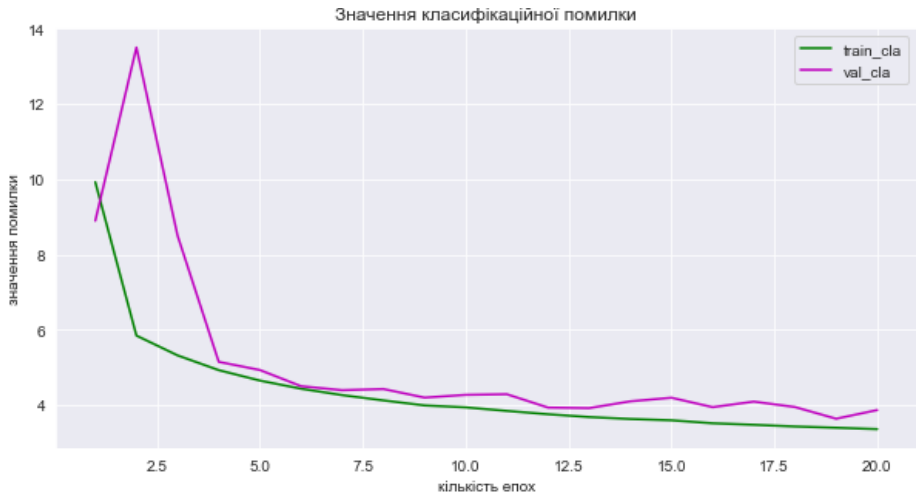


Рисунок 4.12 Графіки класифікаційної помилки навчання експерименту з resnet50

На графіку видно, що існує спочатку невеликий скачок на валідаційній помилці, оскільки це сталось на початку тренування, то це зумовлено початковою ініціалізацією мережі, з часом ця проблема зникає і мережа сходиться до правильних значень помилки. Якщо порівняти з Mobilenet то помилка трохи менша.

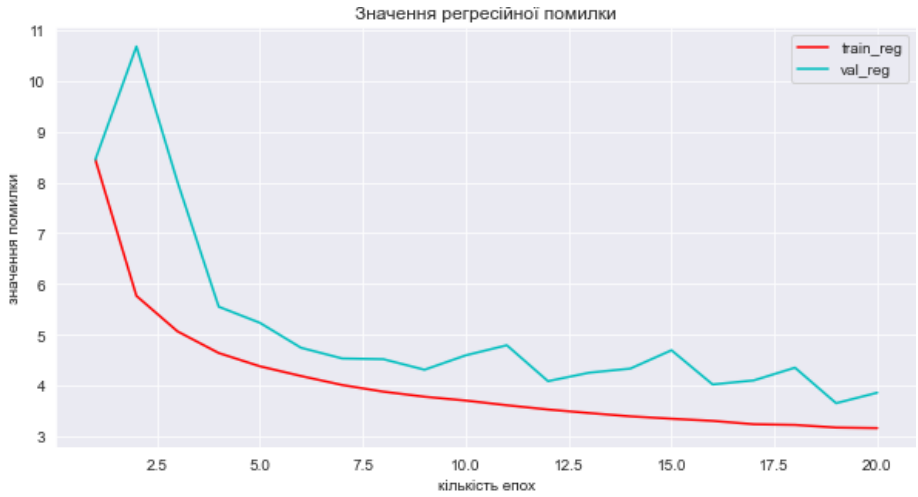


Рисунок 4.13 Графіки регресійної помилки навчання експерименту з resnet50

Схожа ситуація зі скачком помилки відбувається і на регресійній частині. Також при порівнянні з Mobilenet значення найкращої помилки трохи менші, як і у

випадку з класифікацією, різниця невелика, але вона є. Це і підтверджується метрикою, що є більшою, але не набагато.

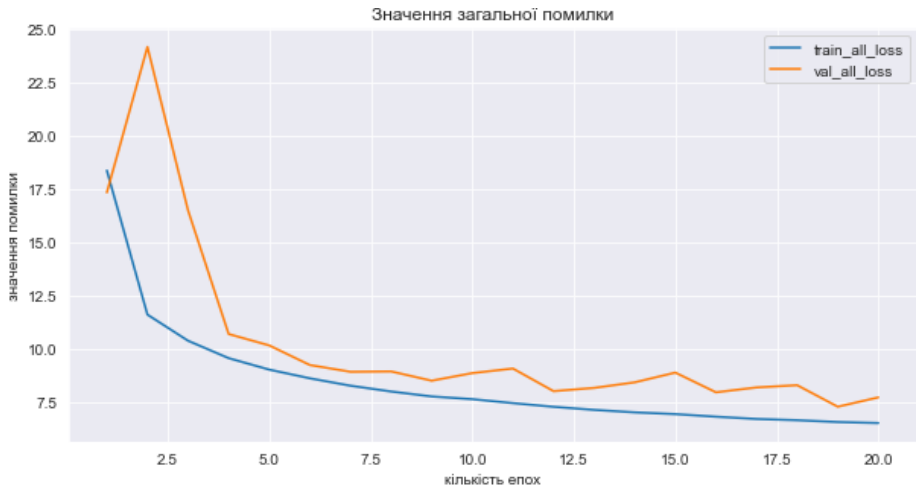


Рисунок 4.14 Графіки загальної помилки навчання експерименту з resnet50

Результати з resnet50 в метриках: на валідаційному датасеті середнє між двома класами - 0.8575, а на тренувальному – 0.8777. По метрикам і по графікам мережа з resnet вчиться краще, і дає кращий результат. Потім спробував використати vgg16 у якості енкодера. Результати навчання представлені на рис 4.15-4.17.

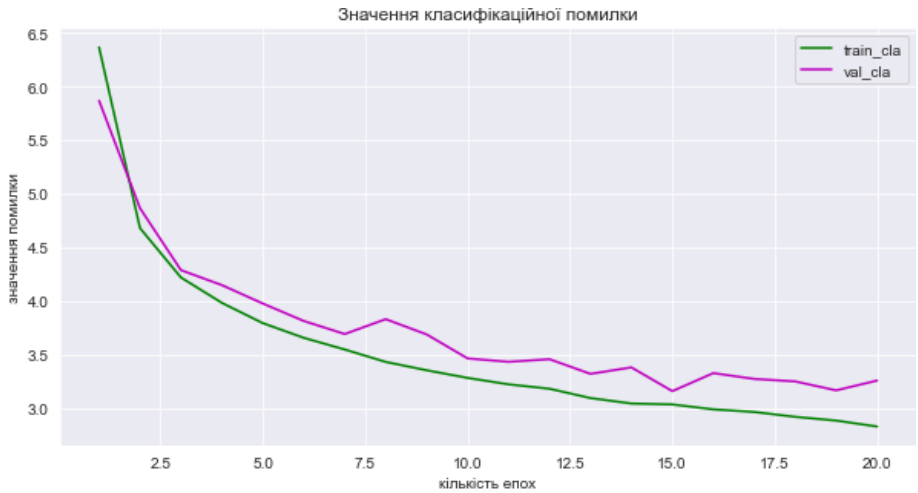


Рисунок 4.15 Графіки класифікаційної помилки навчання експерименту з vgg16

Під час навчання з vgg16 енкодером класифікаційна помилка падає як на тренуванні так і на валідації краще ніж всі попередні експерименти. Також помилка

сходиться до меншого локального мінімуму, що означає що мережа працює краще ніж всі інші. В попередніх експериментах найкраща класифікаційна помилка була близькою до 4, а тут менша ніж 3.5

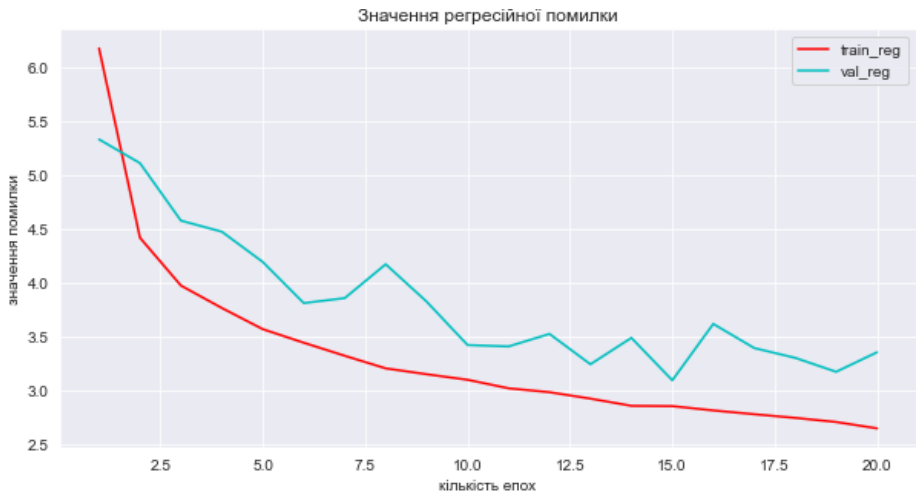


Рисунок 4.16 Графіки регресійної помилки навчання експерименту з vgg16

Регресійна помилка має схожу тенденцію як і класифікаційна, тобто також є найкращою ніж всі інші експерименти, така ж тенденція і на загальній помилці (рис. 4.17).

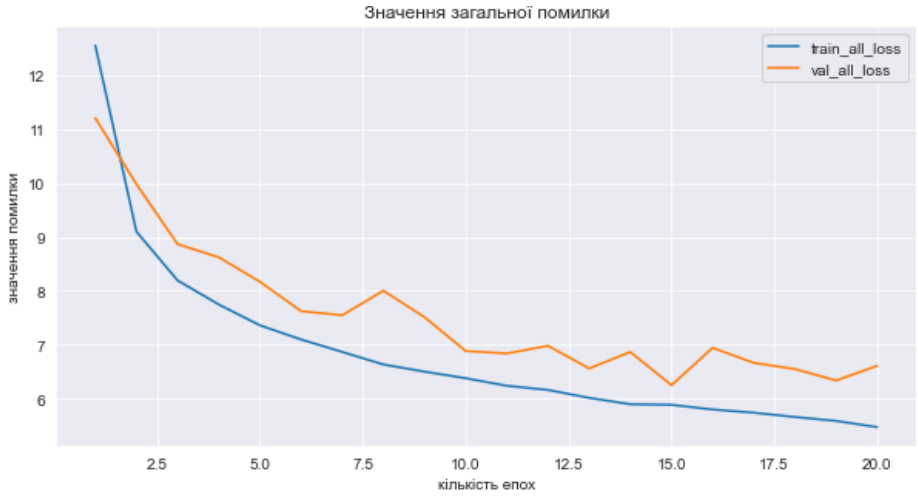


Рисунок 4.17 Графіки загальної помилки навчання експерименту з vgg16

За результатами по метрикам: на валідаційній вибірці середнє між двома класами – 0.9, а на тренуванні 0.9282. Що є найкращим результатом зі всіх мереж.

Розглянемо порівняльні графіки трьох останніх експериментів на рис. 4.18. По ньому також видно те що графіки помилки падали пропорційно до використовуваного енкодера: найкраще для vgg16, потім resnet50 і потім mobilenetv2.

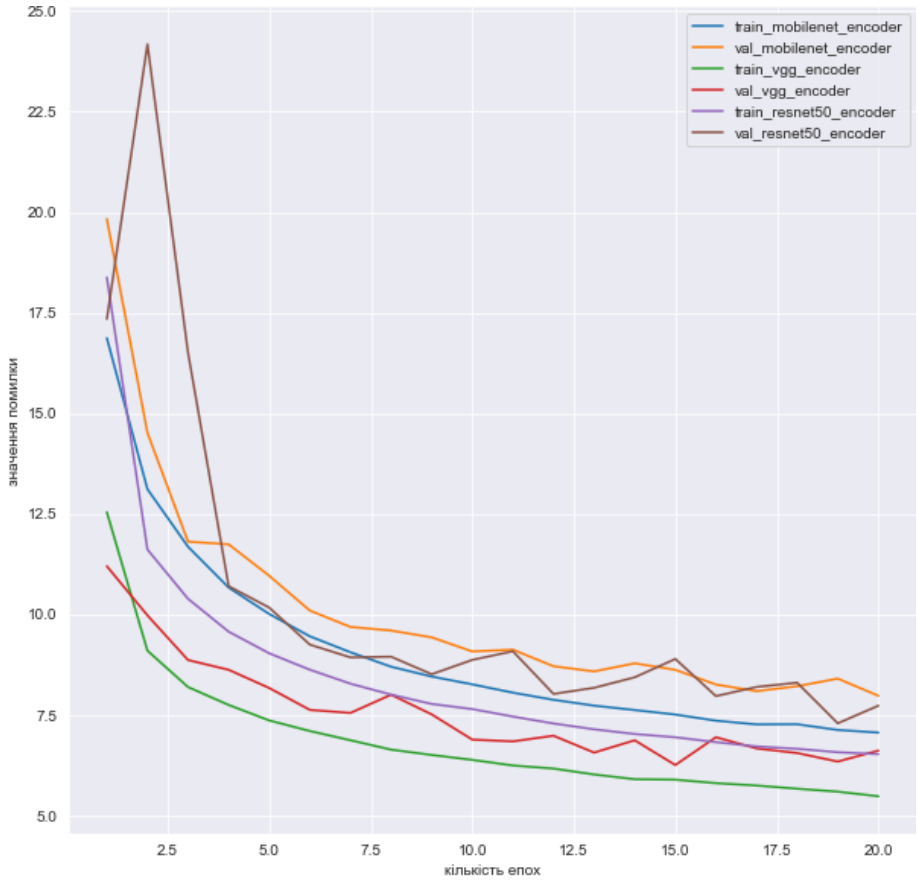


Рисунок 4.18 Графіки порівняння загальної помилки для експериментів з різними енкодерами

Також варто порівняти ці експерименти візуально. Це зроблено на рис. 4.19. В цьому порівнянні видно що vgg16 трохи краще знаходить людей без масок, проте ця різниця не значна, і можливо так співпало і тільки на моїх тестових фотографіях така залежність. На моїх тестових даних видно що на третій фотографії найшлась додаткова людина без маски. Це доволі складний випадок із людиною не у фокусі, тому не дивно що справилась тільки найкраща модель На рисунку зображені розташовані в наступному порядку: перший рядок – це результат третього експерименту, другий – це результат четвертого експерименту, третій – це результат п’ятого експерименту.

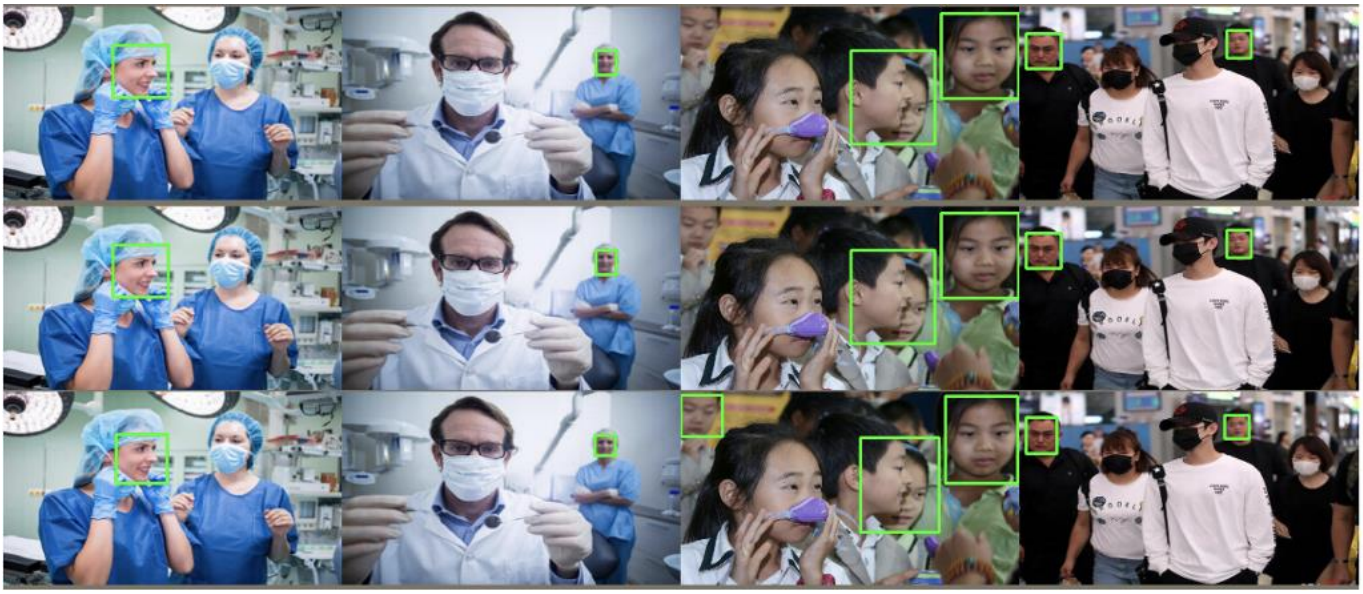

Рисунок 4.19. Порівняння роботи 3,4,5 експериментів

Якщо ж дивитись на метрики і графіки навчання різниця між трьома енкодерами є, проте не велика. Загальну картину всіх експериментів можна побачити в таблиці 4.1. А загальні графіки тренунувань по всіх експеиментах в додатку И. Примітка * позначено те, що для базової моделі та моделі яка претрейнена на імаджені (ті які тренувались тільки на одному класі) фінальний mAP не рахувався як середній.

Таблиця 4.1 Результати експериментів

| Екперимент | Валідаційна mAP на класі без маски | Валідаційна mAP на класі з маскою | Середнє між двома класами на валідаційному сеті | Середнє між двома класами на тренувальному сеті |
|---|---|---|---|---|
| Базова модель | 0.6767 | - | 0.6767* | 0.7108* |
| Претрейн на ImageNet | 0.7747 | - | 0.7747* | 0.8006* |

| | | | | |
|---|---|---|---|---|
| Детектор на два класи (Mobilenetv2 як енкодер) | 0.7899 | 0.9020 | 0.8459 | 0.8630 |
| Resnet50 як енкодер | 0.8126 | 0.9024 | 0.8575 | 0.8777 |
| VGG16 як енкодер | 0.8669 | 0.9361 | 0.9 | 0.9282 |

На перший погляд може здаватись, що найкраще використовувати Vgg16 в будь-якому випадку. Проте неправильно порівнювати різні енкодери у відриві від часу їх роботи. Ми працюємо з відео, а тому час роботи на одному кадрі має критичне значення. Я поміряв ці значення на своєму компютері з Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz. Та вимірював тільки час інференсу на CPU. Я використовував чистий tensorflow v1 без ніяких оптимізацій і результати часу роботи представлені в таблиці. 4.2

Таблиця 4.2 Результати часу роботи

| Енкодер | Середній час інференсу (в секундах на один інференс) |
|---|---|
| MobilenetV2 | 0.1 |
| Resnet50 | 0.16 |
| VGG16 | 0.23 |

Тут можна побачити, що відносно поганий MobilenetV2 компенсує свою точність прекрасною швидкодією, а vgg16 напваки. Отже найкращим в плані швидкодії є Mobilenetv2 енкодер, проте якщо ми використаємо більш потужніші ресурси (наприклад NVIDIA RTX) то результат інференсу буде приблизно однаковий і не буде впливати на роботу системи. Є певне граничне обмеження в тому що, при часу інференсу меншому ніж 0.1 секунди якість роботи системи не буде покращуватись. Це пов’язано з тим що люди не рухаються швидше, і велика кількість

оброблених кадрів не дасть більшого результату. Тому чим потужніші ресурси в нас є, тим більшу модель і відповідно кращу якість ми можемо використати. Мої рекомендації щодо використання моделей щодо їх швидкодії та точності:

- використовувати тільки моделі які були натреновані з переднавченим енкодером та ті що натреновані на двох класах. Вони мають велику перевагу в точності порівняно з іншими;
- якщо програма буде запускатись на смартфоні, чи слабкому процесорі, то найкращим вибором буде використання MobileNetV2 енкодера;
- якщо програма буде запускатись на потужному процесорі, чи слабкій відеокарті, то краще використовувати ResNet50 у якості енкодера. Перевага у точності яку надає ResNet50 буде більшою ніж втрата швидкості на таких ресурсах;
- якщо програма буде запускатись на потужній відеокарті (наприклад серія RTX від Nvidia) то є сенс використовувати найпотужнішу модель – з енкодером Vgg16.

## 4.4 Порівняння результатів з попередніми роботами

Багато попередніх робіт використовували дуже різні датасети та різні методи вимірювання. Так, наприклад, у задачах в яких просто класифікують картинки використовують метрики для класифікації: accuracy, precision, recall. В багатьох статтях де використовується детекція часто міряють не mAP а інші метрики, наприклад точність чи кількість не павильно найдених об’єктів. Проте, навіть статті в яких використовується детекція об’єктів, і така ж метрика як в мене не завжди можна порівняти, це пов’язано з тим що у різних роботах використовуються різні датасети, і найти деякі з них практично неможливо. А, коли річ йде про певні відсотки точності, порівняння моделей які вчились та тестувались на різних вибірках є некоректним.

Найближчою до моїх досліджень була робота [15]. У якій був наданий датасет для досліджень який був вже поділений на тренувальну та валідаційну частину та використовувалась та ж метрика для вимірювань що і в мене – mAP при IOU=0.5. Для

коректного порівняння я обрав датасет з цієї роботи для проведння своїх експериментів, використовуючи оригінальне розбиття на тренувальну і валідаційну частину. І для правильного порівняння в цих експериментах я міряв mAP на двох класах: люди без маски та люди з маскою. А фінальним результатом вважав середнє з цих двох вимірів. За результатами найкращий mAP на валідаційному датасеті виявився 0.9, що є на 0.01 краще ніж у [15]. У них - 89%

## 4.5 Використання системи для виявлення людей без ЗІЗ на реальних прикладах

Для демонстрування роботи програми, я використав модель з MobileNet енкодером, та запускав його на своєму комп'ютері. На рисунку 4.20 зображено примітивні приклади, де лице з маскою та без маски і дивиться прямо в камеру. На таких прикладах моя система справляється без проблем.

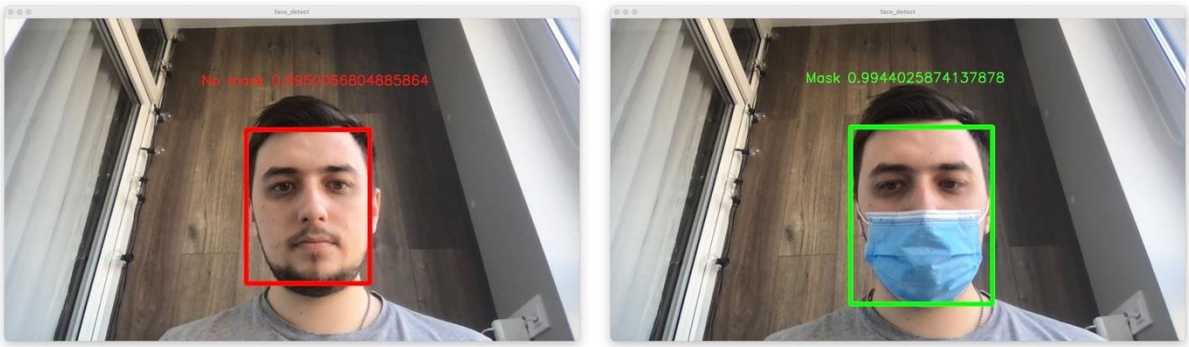


Рисунок 4.20 Приклади використання на примітивному прикладі

Також система працює без проблем для прикладів в яких присутні відхилення від прямого положення голови: вправо та вліво (рис. 4.21) та вверх і вниз (рис. 4.22). Це пов'язано з використання різних аугментації під час подачі даних до мережі, тобто повернення вправо чи вліво, використання афінних перетворень, віддзеркалення фото з права на ліво, та зверху вниз. Це допомогло мережі бачити більше різноманітних облич.


Рисунок 4.21 Приклад з відхиленням голови вправо та вліво

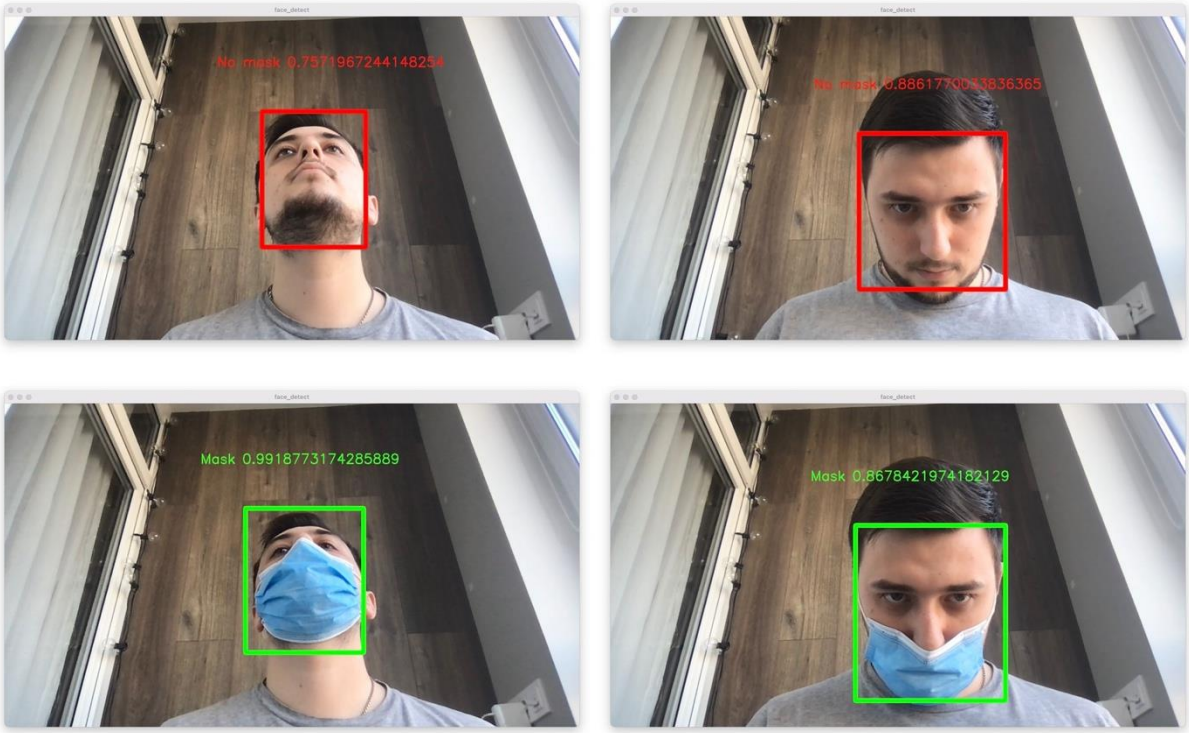


Рисунок 4.22 Приклад з відхиленням голови вверх та вниз

Також моя система працює добре з людьми які носять маску на підборідді. Натягнута маска на підборіддя не тільки заважає, а і не допомагає при

розповсюдженні хвороби. Проте з такими випадками моя система теж працює (рис. 4.23).

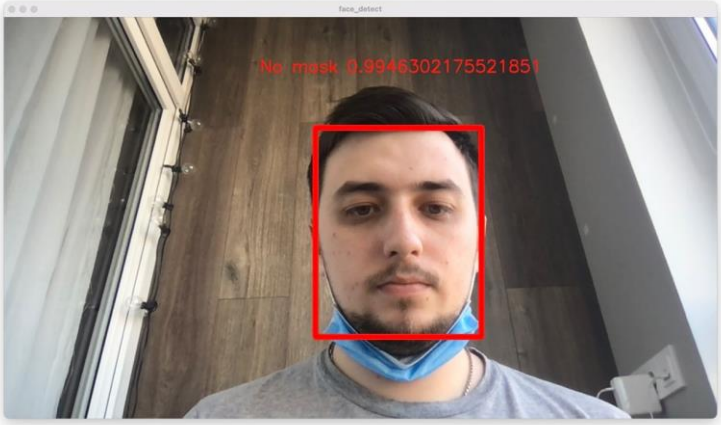


Рисунок 4.23 Приклад з носінням маски на підборідді

Моя система працює також і на фотографія в яких присутні не тільки носіння маски на підборідді, а і повороти голови (рис. 4.24), що означає що система не реагує на просто маску в кадрі, а знаходить семантичний зв'язок між обличчям та розплодженням маски відносно нього.

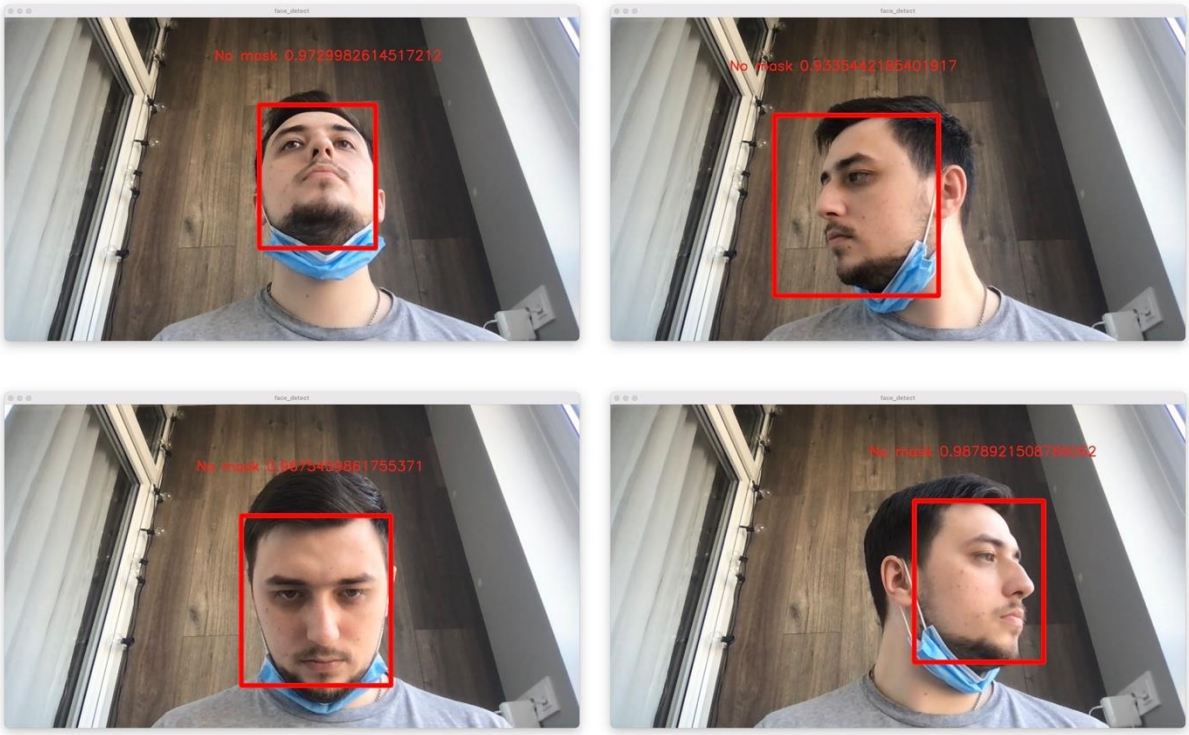


Рисунок 4.24 Приклад із використання маски на підборідді та поворотів голови

Ще одним таким підтвердженням знаходження семантичного зв’язку є те що система працює на обличчях в яких маска одягнена неправильно, навіть у випадку коли рот закритий частково, а «виглядає» з під маски тільки ніс (рис. 4.25)


Рисунок 4.25 Приклад із використанням неправильно одягнутої маски

Хочу зауважити що всі ці приклади актуальні як для відео потоку так і для звичайної картинки. Розглянуті приклади це скриншот із робочої програми, яка працювала з відеопотоком.

4.6 Висновок

В цьому розділі описані результати моїх експериментальних досліджень. Оскільки дослідження проводились двох типів, результати я представив у двох частинах: ті які проводились тільки під час пошуку кращої архітектури для навчання, та експерименти з різними енкодерами.

В пункті 4.1 описано планування тих експериментів які я хочу провести, це експреимент з додаванням перетренованого енкодера. Експеримент зі зміною задачі, тобто навчання детектора не на одному класі а на двох. Експерименти з викорстання різних енкодерів.

В 4.2 розглянуто результати експериментів 1-3. Для кожного експерименту представлені криві навчання, та результати роботи на тестових даних. Експерименти порівнювались за допомогою mAP метрики яка описана в другому розділі цієї роботи В результаті найкраще себе показав експеримент з використанням

передтренерованим енкодером та використання двох класів для навчання. Він краще від першого експеримента на 0.1 mAP, а від другого на 0.01.

Оскільки в 4.2 було визначено найкращий експеримент третім в 4.3 розглянуто різні енкодери які вчились так само як і третій. В цих дослідженнях я вже порівнював експерименти по двом класам, в результаті чого було визначено найкращий енокдер, ним виявився vgg16. Окремо наведено порівняння енкодерів між собою за характеристиками швидкодії – найшвидшим був MobileNetV2. В цьому пункті було наведено рекомендації користувачеві в залежності від його обчислювальних ресурсів.

В 4.4 відбулось порівняння моїх результатів з попередніми роботами. Оскільки таких як в мене експериментів проводилось небагато, то порівнювати було мало з чим. Проте в порівнянні з робою [15] в мене результати краще на 0.01 mAP.

В 4.5 було представлено приклади роботи системи автоматичного виявлення людей без ЗІЗ. Розглянуто приклади різної складності: від примітивних де людина дивилась прямо в камеру до таких де з під маски видно тільки ніс. На всіх прикладах моя система справилась.

# 5 РОЗРОБКА СТАРТАП ПРОЕКТУ

## 5.1 Опис ідеї стартап проекту

Зараз світ перебуває в стані пандемії COVID-19. Цей вірус може поширюватись через дихальну систему. Тому носіння масок зараз є найкращим способом захистити себе та оточуючих від зараження. Проте люди можуть хитрувати, та не носити маски, тому їх потрібно контролювати. В цьому зацікавлені не тільки ВООЗ чи МОЗ, а також і бізнес: якщо працівники не будуть дотримуватись правил носіння ЗІЗ під час пандемії, то вони частіше будуть хворіти, що відобразиться на фінансовій ситуації компанії. Мій проект допомагає боротись із тими хто нехтує правилами користування засобами індивідуального захисту.

Він заснований на алгоритмах машинного навчання, та працює автоматично. Він має зручний графічний інтерфейс та знаходить кожну окрему особистість. Також ця система знаходить не тільки людей без маски, а і з нею. Завдяки добре розвиненій сфері комп’ютерного зору вдалось досягти гарної якості детекції (більше > 90% в кращої моделі). Також система є кросплатфорента і має декілька готових до використання моделей які підходять для різних за обчислювальними можливостями пристроях. В таблиці 5.1 наведено основну ідею, напрямки застосування та вигоди для користувача мого проекту.

Таблиця 5.1. Опис ідеї стартап-проекту

| Зміст ідеї | Напрямки застосування | Вигоди для користувача |
|---|---|---|
| Знаходження людей без засобів індивідуального захисту | Використання під час пандемії у громадських місцях для запобігання розповсюдження вірусу | Має змогу знаходити людей без ЗІЗ у автоматичному режимі |

Проаналізувавши ринок, було визначено основних конкурентів – WearMask та RetinaFace. В таблиці 5.2 наведено порівняння мого проекту з конкурентами, визначено переваги та недоліки кожного.

Таблиця 5.2 Визначення сильних, слабких та нейтральних характеристик ідеї проекту

| № | Технікоекономічні характеристики ідеї | Продукція конкурентів | | | W (слабка сторона) | N (нейтральна сторона) | S (сильна сторона) |
|---|---|---|---|---|---|---|---|
| | | Мій проект | WearMask | RetinaFace | | | |
| 1 | Зручність графічного інтерфейсу | + | + | + | - | + | - |
| 2 | Простота у користуванні | + | + | - | - | - | + |
| 3 | Швидкість знаходження обличчя | + | + | - | - | - | + |
| 4 | Знаходження кожного обличчя окремо | + | + | + | - | + | - |
| 5 | Можливість запуску на GPU | + | - | + | - | + | - |
| 6 | Використання штучних датасетів, що поліпшує якість | + | + | - | - | - | + |
| 7 | Можливість запуску на будь якому пристрої | + | + | - | - | - | + |
| 8 | Наявність моделі під конкретні потужності | + | - | - | - | - | + |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | Робота з відеопотоком | + | + | - | - | + | - |
| 10 | Знаходження, окрім людей без ЗІЗ, людей з ЗІЗ | + | + | - | - | - | + |

В цій таблиці представлено основні сильні сторони мого проекту в порівнянні з конкурентами. Можна помітити що в обох конкурентах немає наявних моделей для конкретних потужностей. Також можна помітити що всі потрібні характеристики є тільки в моєму проекті, в інших є не всі, а тільки деякі представники. Всі ці характеристики є підставою для формування конкурентоспроможності мого проекту.

5.2. Технологічний аудит ідеї проекту

В цьому підрозділі визначено аудит та технології за допомогою яких можна реалізувати мій проект. В таблиці 5.3 описано технолігії за допомогою яких можна реалізувати мою ідею.

Таблиця 5.3 Технології здійснення ідеї проекту

| № | Ідея проекту | Технології її реалізації | Наявність технологій | Доступність технологій |
|---|---|---|---|---|
| 1 | Web-додаток | CSS, HTML, JS, TensorflowJS, MYSQL | Технології наявні в Open Source | Вільна |
| 2 | Декстопний додаток | Python, Opencv, Tensorlfow, CUDA | Технології наявні в Open Source | Вільна |
| 3 | Викоритсання серверного рішення з | Ubuntu, Python, Opencv, Tensorlfow, CUDA, MongoDB | Технології наявні в Open Source | Вільна |

| | | | | |
|---|---|---|---|---|
| | декстопним клієнтом | | | |

В результаті обрана технологічна реалізація продукту під номером 2, основні причини:

– система дешевша;
– проста в розроблені;
– простота інтеграції;
– не потребує додаткових засобів для коритсування.

5.3. Аналіз ринкових можливостей запуску стартап-проекту

В цьому підпункті проведено аналіз ринкових можливостей стартап-проекту. В таблиці 5.4 представлені характеристики ринку на якому буде конкурувати мій проект.

Таблиця 5.4. Попередня характеристика потенційного ринку стартап проекту

| № | Показники стану ринку | Характеристика |
|---|---|---|
| 1 | Кількість головних учасників, од | 5 |
| 2 | Загальний обсяг реалізації, грн./ум.од | 12000 в рік |
| 3 | Динаміка ринку | Стрімко зростає |
| 4 | Наявність обмежень | Немає |
| 5 | Специфічні вимоги | Немає |
| 6 | Середня рентабельності в галузі або по ринку, % | 90% |

З таблиці видно, що динаміка ринку стрімко зростає, це логічно, оскільки пандемія почалась недавно. Цим і зумовлена невелика кількість головних учасників ринку. Також очікується доволі великі продажі продукту у зв’язку із стрімким поширенням хвороби та впливом її на загальну економіку. Проте протягом декількох років ця ситуація буде змінюватись. Розглянемо характеристики потенційних клієнтів мого стартап-проекту (табл. 5.5).

Таблиця 5.5 Характеристика потенційних клієнтів стартап проекту

| № | Потреба, що формує ринок | Цільова аудиторів(цільові сегменти ринку) | Відмінності у поведінці різних потенційних цільових груп клієнтів | Вимоги споживачів до товару |
|---|---|---|---|---|
| 1 | Потреба знаходження людей без ЗІЗ для зниження розповсюдженості вірусу у час пандемії | Торгові центри, державні установи, навчальні заклади, приватні компанії | Дані групи користувачів зацікавлені у швидкому знаходженні людей без масок | Швидке знаходження людей без ЗІЗ |
| 2 | Потреба знаходження працівників без масок у медичних закладах | Медичні заклади | Зацікавлені у точному знаходженні працівників без масок | Точне знаходження людей без ЗІЗ |

Із таблиці видно, що клієнтів є доволі багато, тому буде багато зацікавлених учасників ринку в придбанні даного товару. Це різні організації які зацікавленні у зменшенні розповсюдження вірусу, які включають в себе як приватні так і державні організації. Всі вони зацікавлені як і у швидкому так і доволі точному детектування людей без ЗІЗ.

Як і будь-який проект, мій має певні фактори загроз. Розглянемо їх у таблиці 5.6.

Таблиця 5.6 Фактори загроз

| № | Фактор | Зміст загрози | Можлива реакція компанії |
|---|---|---|---|
| 1 | Конкуренти | Наявність конкурентів які надають такі ж послуги але за меншою ціною чи кращою якістю | Зменшення ціни, покращення якості послуги |
| 2 | Закінчення пандемії | Якщо пандемія закінчиться, основним клієнтам перестане бути потрібна послуга | Фокусування на інших ринках, створення точніших знаходжень для медичних закладів |
| 3 | Інвестиції | Закінчення коштів для розробки | Залучення нових інвесторів, пропонувати себе на фінансування ВООЗ |
| 4 | Кадрові проблеми | Нестача розробників потрібної кваліфікації | Залучення розробників з фрілансу |

Із таблиці можна виділи загрозу закінчення пандемії, проте її можна вирішити якщо розвинути проект в іншому напрямі. Цей та інші фактори росту можна побачити в таблиці 5.7.

Таблиця 5.7 Фактори можливостей

| № | Фактор | Зміст можливостей | Можлива реакція компанії |
|---|---|---|---|
| 1 | Створення нового продукту на основі старого | Можна використовувати можливості програми в інших цілях, наприклад підрахунок кількості людей в магазині | Перероблення основного продукту під нові потреби |

| | | | |
|---|---|---|---|
| 2 | Науково-технічний | Використання нових та інноваційних технологій | Зростання актуальності і потреби продукту |
| 3 | Поява схожих до моєї розробок у конкурентів | Конкуренти можуть зробити такі ж або дуже схожі технології | Патентування алгоритмів |

Протягом часу кількістю можливостей та загроз будуть збільшуватись, тому потрібно буде швидко реагувати на зміни в ринковому положенні. Проведемо ступеневий аналіз конкуренції на ринку (табл. 5.8).

Таблиця 5.8 Ступеневий аналіз конкуренції на ринку

| № | Особливості конкурентного середовища | В чому проявляється дана характеристика | Вплив на діяльність підприємства |
|---|---|---|---|
| 1 | Тип конкуренції: олігополія | На ринку налічується декілька конкурентів зі схожими продуктами | Покращення алгоритмів свого продукту, можливе об’єднання з конкурентами для створення монополічного продукту |
| 2 | За галузевою ознакою: внутрішньогалузева | Даний продукт може використовуватись тільки в одній сфері | Потрібно покращувати якість та швидкість детектування |
| 3 | За рівнем конкурентної боротьби: світовий | Всі продукти розроблялись інтернаціональними командами, і належать групі розробників а не державі | Пристосування продукту для різних регіонів |

| | | | |
|---|---|---|---|
| 4 | Конкуренція за видами товарів: товарно-видова | Кожнкуренція між схожими алгоритмами товарів | Покращення точності роботи алгоритму |
| 5 | За характером конкурентних переваг: цінова | Співвідношення якості і ціни | Продаж програмного забезпечення за економічно-зваженою ціною |
| 6 | За інтенсивністю: марочна | Наявність брендів та їхньої конкуренції | Робота над впізнаваностю бренду за допомогою реклами |

Після проведення аналізу можна визначити що на ринку є конкуренти, але потрібно зазначити що якість моєї моделі найкраща, тому потрібно позиціонувати себе зі сторони якості моделі, та постійно її покращувати. Розглянемо аналіз конкуренції в галузі (табл. 5.9).

Таблиця 5.9 Аналіз конкуренції в галузі за М. Портером

| | Прямі конкуренти в галузі | Потенційні конкуренти | Постачальники | Клієнти | Товаро замінники |
|---|---|---|---|---|---|
| Складові | WearMask та RetinaFace | Реалізація схожих рішень | Немає | Торгові центри, державні установи, навчальні заклади, приватні компанії, медичні заклади | Відсутні |

| | | | | | |
|---|---|---|---|---|---|
| Висновки | Для утримання на ринку потрібно покращувати продукт, а саме працювати над точністю та швидкістю роботи. Та впроваджувати нові методи детекції | Оскільки ринок новий, тому є ймовірність появи нових аналогів. | Немає | Нові потреби користувачів мають бути прийняті до уваги в найближчий час | Відсутні |

Із таблиці видно що є достатня кількість конкурентів. Тому потрібно покращувати продукт, щоб залишатись конкурентоспроможним. Розглянемо обґрунтування факторів конкурентоспроможності (табл. 5.10).

Таблиця 5.10 Обгрунтування факторів конкурентоспроможності

| № | Фактор конкурентоспроможності | Обґрунтування |
|---|---|---|
| 1 | Автоматизація знаходження людей без ЗІЗ | В період пандемії допомагає зменшувати розповсюдження хвороби через знаходження людей без ЗІЗ |
| 2 | Реалізація мультиплатформеного додатку | Можливість запуску на різних пристроях |
| 3 | Графічний інтерфейс | Реалізація інтерфейсу з використанням спеціальних позначень |

| | | |
|---|---|---|
| 4 | Швидкодія алгоритму | Реалізація запуску на хардверних прискорювачах |
| 5 | Ціноутворення | Ціна буде утворюватися для продажу більшої кількості екземплярів |
| 6 | Точність роботи | Використання передових технологій при розробці проекту |

Із таблиці можна помітити, що основних факторів конкурентноспрожоності 6. Розглянемо їх всі на своєму проекті (табл. 5.11).

Таблиця 5.11 Порівняльний аналіз сильних та слабких сторін системи

| № | Фактор конкурентоспроможності | Бали 1-20 | Рейтинг товарів-конкурентів у порівнянні з новою системою | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| 1 | Автоматизація знаходження людей без ЗІЗ | 16 | | | + | | | | |
| 2 | Реалізація мультиплатформеного додатку | 19 | + | | | | | | |
| 3 | Графічний інтерфейс | 10 | | | | + | | | |
| 4 | Швидкодія алгоритму | 14 | | | + | | | | |
| 5 | Ціноутворення | 11 | | | | | + | | |
| 6 | Точність роботи | 17 | | + | | | | | |

За результатами можна побачити, що мій проект має велику різниця в рейтингу відносно конкурентів в основних факторах конкурентоспроможності. Розглянемо

SWOT-аналіз проекту (табл. 5.12). Визначаємо сильні та слабкі сторони а також можливості та загрози проекту.

Таблиця 5.12. SWOT-аналіз стартап-проекту

| Сильні сторони (S) | Слабкі сторони (W) |
|---|---|
| – автоматизація знаходження людей без ЗІЗ;<br>– зрозумілий графічний інтерфейс;<br>– швидкодія алгоритму;<br>– точність роботи;<br>– кросплатформеність програмного забезпечення;<br>– зручність у використанні;<br>– цінова політика;<br>– застосування новітніх технологій. | – складність розробки;<br>– потреба у виборі потрібної моделі. |
| **Можливості (O)** | **Загрози (T)** |
| – удосконалення функціоналу новими функціями та методиками детекції;<br>– удосконалення роботи та швидкості запуску програмного забезпечення;<br>– покращення алгоритму обробки вхідної інформації. | – потреба у фінансуванні;<br>– Нестача працівників потрібної кваліфікації;<br>– поява нових конкурентів;<br>– особливості епідеміологічного стану. |

За результатами аналізу, видно, що кількість сильних сторін більша за кількість слабких. Також видно що можливостей більше, аніж загроз. Розробимо альтернативи ринкового впровадження стартап-проекту та проаналізуємо їхні ймовірності та терміни реалізації (табл. 5.13).

Таблиця 5.13 Альтернативи ринкового впровадження стартап-проекту

| № | Альтернатива (орієнтовний комплекс заходів) ринкової поведінки | Ймовірність отримання ресурсів | Терміни реалізації |
|---|---|---|---|
| 1 | Робота над впізнаваністю бренду, PR-реклама, контекстна реклама | Середня | 3-4 місяці |
| 2 | Налаштування команди спеціалізаваної на продаж на презентування продукту | Середня | 7 місяців |
| 3 | Показ програмного забезпечення на вузьконаправлених конференціях | Висока | 1 місяць |

Проаналізувавши таблицю, видно що найпростішим способом буде показ програмного забезпечення на конференціях.

5.4 Розроблення ринкової стратегії проекту

В цьому підрозділі розглянуто ринкову стратегію. В таблиці 5.14 представлено вибір цільових груп.

Таблиця 5.14 Вибір цільових груп потенційних споживачів

| № | Опис профілю цільової групи потенційних клієнтів | Готовність споживачів сприйняти продукт | Орієнтовний попит в межах цільової групи (сегменту) | Інтенсивність конкуренції в сегменті | Простота входу у сегмент |
|---|---|---|---|---|---|
| 1 | Державні заклади | Висока | Високий | Дуже низька | Висока |

| | | | | | |
|---|---|---|---|---|---|
| 2 | Тогівельні центри | Середня | Середній | Низька | Висока |
| 3 | Медичні заклади | Висока | Високий | Середня | Висока |

Проаналізувавши цільові групи клієнтів було обрано: державні заклади, торгівельні центри та медичні заклади. В таблиці 5.15 представлено базову стратегію розвитку. Також визначимо базову стратегію конкуретноспроміжної поведінки (табл. 5.16).

Таблиця 5.15 Визначення базової стратегії розвитку

| Обрана альтернатива розвитку проекту | Стратегія охоплення ринку | Ключові конкурентоспроможні позиції відповідно до обраної альтернативи | Базова стратегія розвитку |
|---|---|---|---|
| Впровадження новітніх технологій в програмне забезпечення, розширення напрямку застосування програмного забезпечення | Контекстна реклама на інтернет ресурсах, PR-реклама, побудова бонусної програми | Робота над унікальністю продукту; відомість бренду; | Стратегія диференціації |

Таблиця 5.16 Визначення базової стратегії конкурентної поведінки

| Чи є проект «першопрохідцем» на ринку | Чи буде компанія шукати нових споживачів, або | Чи буде компанія копіювати основні характеристики | Стратегія конкурентної поведінки |
|---|---|---|---|

| | | | |
|---|---|---|---|
| | забирати існуючих у конкурентів? | товару конкурента, які? | |
| Ні, так як на ринку вже є конкуренти | Ціль продукту це збільшення бази клієнтів різноманітними методами, це і долучення нових клієнтів, а також перетягнення клієнтів конкурентів | В компанії є можливості, які пересікають, хоча даний продукт має більш доповнений функціонал | Стратегія заняття конкурентної ніші |

В результаті було обрано стратегію заняття конкурентної ніші. В таблиці 5.17 представлено визначення стратегії позиціонування.

Таблиця 5.17 Визначення стратегії позиціонування

| № | Вимоги до товару цільової аудиторії | Базова стратегія розвитку | Ключові конкурентоспроможні позиції власного стартаппроекту | Вибір асоціацій, які мають сформувати комплексну позицію власного проекту |
|---|---|---|---|---|
| 1 | Зручність графічного інтерфейсу | Стратегія диференціації | Простота та зрозумілість графічного інтерфейсу | Зручність, Простота |

| | | | | |
|---|---|---|---|---|
| 2 | Ціноутворення | Стратегія конкуретної ніші | Ціно буде утворюватись з метою продажу більше екземпдярів, а не дорожчою ціною за один екземпляр | Доступність |
| 3 | Новітність | Стратегія диференціації | Застосування ноухау технологій при розробці даного продукту | Новизна, актуальність |

В результаті для даного проекту в якості базової стратегії для графічного інтерфейсу та новісноті було обрано стратегію диференціації, а для ціноутворення – стратегія конкурентної ніші.

5.5 Розроблення маркетингової програми стартап-проекту

Розглянемо визначення ключових переваг (табл. 5.18).

Таблиця 5.18 Визначення ключових переваг концепції потенційного товару

| № | Потреба | Вигода, яку пропонує товар | Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити) |
|---|---|---|---|
| 1 | Автоматизація знаходження людей без ЗІЗ | Запропонований товар пропонує швидкодію та високу точніть детекції | Товари-аналоги мають меншу швидкодію та точність |

| | | | |
|---|---|---|---|
| 2 | Використання новітніх технологій при розробці програмного забезпечення | Це надає користувачу тримати високу точність та бути впевненим в актуальності додатку | Конкуренти не надають такої можливості |

Можна помітити що основні переваги системи – це знахтодження людей без ЗІЗ та автоматизація новітніх технологій при розробці ПЗ. В таблиці 5.19 представлено опис трьох рівнів.

Таблиця 5.19. Опис трьох рівнів моделі товару

| Рівні товару | Сутність та складові | | |
|---|---|---|---|
| 1. Товар за задумом | Автоматичні розрахунки параметрів генератора аеродинамічної піднімальної сили | | |
| 2. Товар у реальному виконанні | Властивості/характеристики | М/Нм | Вр/Тх/Тл/Е/Ор |
| | Клієнтоорієнтованість та простота | М | Е |
| | Швидкість та точність алгоритму | М | Тл |
| | Ціна | М | Вр |
| | Якість: відповідна нормам програмного забезпечення | | |
| | Пакування: немає | | |
| | Марка: RAR Detection | | |
| 3. Товар із підкріпленням | До продажу: Програмне забезпечення для автоматизованої детекції людей без ЗІЗ | | |

| | |
|---|---|
| | Після продажу: прямий канал зв'язку клієнтів та покращення і впровадження нових технологій у продукт продукту |
| За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності | |

Після цього визначимо порядок цін для проекту (табл. 5. 20)

Таблиця 5.20 Визначення меж встановлення ціни

| Рівень цін на товари-замінники | Рівень цін на товари-аналоги | Рівень доходів цільової групи споживачів | Верхня та нижня межі встановлення ціни на товар/послугу |
|---|---|---|---|
| 100$-200$ | 150$-230$ | 10 000$ | 10$-100$ |

Маючи верхню та нижню межі встановлення ціни, можемо сформувати систему збуту товару (табл. 5.21).

Таблиця 5.21 Формування системи збуту

| Специфіка закупівельної поведінки цільових клієнтів | Функції збуту, які має виконувати постачальник товару | Глибина каналу збуту | Оптимальна система збуту |
|---|---|---|---|
| Потрібно придбати програмне забезпечення на сайті | Веб-сайт | Нульовий | Веб-сайт |

Отже, основною системою збуту буде продаж через веб сайт. В таблиці 5.22 представлено основні комунцікації.

Таблиця 5.22 Концепція маркетингових комунікацій

| № | Специфіка поведінки цільових клієнтів | Канали комунікацій, якими користуються цільові клієнти | Ключові позиції, обрані для позиціонування | Завдання рекламного повідомлення | Концепція рекламного звернення |
|---|---|---|---|---|---|
| 1 | Надання доступу до програмного забезпечення | Соціальні мережі, форма на сайті | Швикодія та точність | Донести до клієнта всю унікальність та новизну даного продукту | Легкодоступна і креативна |
| 2 | Регулярні оновлення додатку | | Клієнто-орієнтованість | Показати клієну що є можливість користуватись останніми оновленнями | |
| 3 | Служба підтримки | | Клієнто-орієнтованість | Служба яка допоможе клієнтам правильно запустити продукт, та вирішить проблеми з використанням проекту | |

В результаті комунікації будуть націлені на дві основні специіфки поведінки: безпосередньо надання доступу до програмного забезпечення та надання доступу до регулярних оновлень. Клієнти зможуть комікувати з розробниками через соціальні мережі та форму яка буде на сайті.

## 5.6 Висновок

В цьому розділі було проведено маркетинговий аналіз стартап проекту автоматизованої системи знаходження людей без ЗІЗ. В результаті було підтверджено можливість ринкової комерціалізації проекту. Зараз існує велика кількість клієнтів яким потрібен даних продукт.

Зараз присутньо небагато конкурентів, а отже є перспективи впровадження системи. Проте існує також основна загроза проекту – завершення пандемічого часу. В даному розділі надано альтернативні можливості роботи проекту – він може бути застосовани тільки у медичних закладах, або для моніторингу кількості відвідувачів певного місця.

Після проведеного аналізу конкурентів та ринкової ситуації доцільною є імплементація даної роботи, та подальше інвестування в даний проект.

# ВИСНОВКИ

В результаті роботи над магістерською дисертацією було розроблено автоматизовану систему виявлення людей без засобів індивідуального захисту. Для створення цієї системи було проаналізовано наявні способи вирішення даної проблеми та визначено, що зараз не існує рішень які підходять для вирішення задачі, проте одне з таких рішень було обрано як основу для досліджень. Була описана сама модель машинного навчання, її шари та особливості роботи. Описано повний алгоритм навчання. Також було проведено опис даних, та їхню обробку.

Ця система була реалізована у вигляді двох програм. Перша – це система для тренування. Вона потрібна для того щоб можна було створювати експерименти, та навчати моделі. Ця програма гнучко налаштовується за допомогою конфігураційного файлу, та після роботи першої програми можна отримати модель машинного навчання яка вже готова до роботи.

За допомогою першої програми було проведено експериментальне дослідження моделі машинного навчання. Дослідження складається з двох етапів: перший це дослідження моделі при зміні задачі машинного навчання та експериментів з transfer learning. Найкращою моделлю була модель з використанням перетренованого енкодера та з навчанням на два класи. У другій частині було проведено експерименти з різними енкодерами та визначено найкращий за якістю – Vgg16 та найкращий за швидкістю – MobileNet.

Ці моделі використовуються у другій програмі. Суть її – знаходити людей без ЗІЗ у реальному часі. Ця програма і є основним результатом роботи над проектом. Для коректної її роботи було наведено рекомендації користувачеві під час яких ресурсів які моделі використовувати. Також показано прилад роботи самої системи на прикладах різної складності, на всіх – результати позитивні.

Отриману систему можна використовувати в громадських місцях для знаходження людей без ЗІЗ. В п’ятому розділі проведено аналіз стартап проекту. У результаті роботи була досягнена мета, та розроблено автоматичну систему виявлення людей без ЗІЗ.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.
2. R. Girshick. Fast R-CNN. In ICCV, 2015.
3. S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.
4. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-Cnn. In: IEEE international conference on computer vision, pp 2961–2969
5. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In CVPR, 2016
6. Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.
7. 7. Alexey Bochkovskiy, Chien-Yao Wang, and HongYuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.
8. Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, Jian Sun. YOLOX: Exceeding YOLO Series in 2021. arXiv preprint arXiv:2107.08430, 2021.
9. Liu, Wei and Anguelov, Dragomir and Erhan, Dumitru and Szegedy, Christian and Reed, Scott, “SSD: Single Shot MultiBox Detector” in Computer Vision – ECCV 2016, publisher. Springer International Publishing, pp. 21–37, 2016.
10. Walid Hariri. Efficient Masked Face Recognition Method during the COVID-19 Pandemic. arXiv preprint arXiv:2105.03026, 2021.
11. Bishwas Mandal, Adaeze Okeukwu, Yihong Theis. Masked Face Recognition using ResNet-50. arXiv preprint arXiv:2104.08997, 2021.
12. A. Anwar and A. Raychowdhury. Masked face recognition for secure authentication. arXiv preprint arXiv:2008.11104, 2020.
13. Mare, T., Duta, G., Georgescu, M. I., Sandru, A., Alexe, B., Popescu, M., & Ionescu, R. T. (2021). A realistic approach to generate masked faces applied on two novel masked face recognition data sets. arXiv preprint arXiv:2109.01745.
14. Dataset, https://github.com/prajnasb/observations, online accessed May 25, 2020

15. Wang, Z., Wang, P., Louis, P. C., Wheless, L. E., & Huo, Y. (2021). Wearmask: Fast in-browser face mask detection with serverless edge computing for covid-19. arXiv preprint arXiv:2101.00784.
16. Li, J., Wang, Y., Wang, C., Tai, Y., Qian, J., Yang, J., ... & Huang, F. (2019). DSFD: dual shot face detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5060-5069).
17. Tensorflow object detection api, https://github.com/tensorflow/models/tree/master/research/object_detection, online accessed Jan 10, 2016.
18. Mmdetection, https://github.com/open-mmlab/mmdetection, online accessed Aug 19, 2018.
19. DSFD-tensorflow, https://github.com/610265158/DSFD-tensorflow, online accessed Aug 18, 2019.
20. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
21. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
22. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).
23. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of Advances in Neural Information Processing Systems (NIPS), 2015.
24. Hoiem, D., Divvala, S. K., & Hays, J. H. (2009). Pascal VOC 2008 challenge. *World Literature Today*.
25. S. Yang, P. Luo, C.-C. Loy, and X. Tang, “Wider face: A face detection benchmark,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, p

26. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.