МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інтелектуальна технологія моніторингу функціонального стану доріг та узбіч»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Коробов А.Г.**

Студента групи ІН.м-02 **Білоцерковець С. А.**

СУМИ 2021

Сумський державний університет

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«122 -Комп'ютерні науки»*

Затверджую:

зав.кафедрою _____

_____

"_____" _____

20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Білоцерковцю Сергію Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інтелектуальна технологія моніторингу функціонального стану доріг та узбіч*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____

_____

_____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналіз проблеми. Постановка задачі дослідження. 2) Аналіз існуючих рішень моніторингу 3) Вибір об'єктів дослідження. Детектування об'єктів. Інтелектуальна система в режимі екзамену 4) Розробка інформаційної технології моніторингу функціонального стану доріг та узбіч

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____

_____

_____

_____

_____

_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми. Постановка задачі дослідження*		
2.	*Аналіз існуючих рішень моніторингу*		
3.	*Вибір об'єктів дослідження. Детектування об'єктів. Інтелектуальна технологія в режимі екзамену*		
4.	*Розробка інформаційної технології моніторингу функціонального стану доріг та узбіч*		
5.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____

(підпис)

Керівник проекту _____

(підпис)

РЕФЕРАТ

Записка: 46 стор., 20 рис., 3 табл., 2 додатки, 17 джерел.

Об’єкт дослідження — процес моніторингу функціонального стану доріг та узбічь за умов інформаційних обмежень.

Мета роботи — підвищення ефективності функціонування технології моніторингу доріг та узбіч шляхом розробки інформаційної технології виявлення пошкоджень та засмічень.

Методи дослідження — методи технології згорткових нейронних мереж, методи теорії кодування та оптимізації при побудові інформаційної технології машинного навчання інформативного ознакового опису в умовах обмеженого обсягу розмічених зразків.

Результати — розроблено інтелектуальну технологію, що визначає дефекти дорожнього покриття та засміченість узбіч.

ТЕХНОЛОГІЯ ДЕТЕКТУВАННЯ, ВИБІРКА ЗОБРАЖЕНЬ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ІНФОРМАЦІЙНО-ЕКСТРЕМАЛЬНА ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ, КЛАСИФІКАЦІЯ ОБ’ЄКТІВ НА ЗОБРАЖЕННЯХ.

Зміст

Зміст 5

Вступ 6

1. Аналіз проблеми та постановка задачі 8

1.1 Сучасні технології моніторингу стану доріг та узбіч 8

1.2 Аналіз моделей та методів штучного інтелекту для прийняття рішень на основі візуальних даних 14

1.3 Постановка задачі 20

2. Інтелектуальна технологія моніторингу функціонального стану доріг та узбіч 22

2.1 Модель нейромережі для детектування пошкоджень дорожнього покриття та сміття на узбіччі 22

2.2 Навчання моделі нейронної мережі 24

2.3 Оцінка точності детектування 26

3. Реалізація інтелектуальної технології моніторингу функціонального стану 29

3.1 Створення вибірки навчальних та тестових даних 29

3.2 Результат навчання моделі 32

3.3 Програмна реалізація 35

Висновки 38

Список літератури 39

Додаток А 41

Додаток Б 45

Вступ

На сьогоднішній день процеси інформатизації стали невід'ємною частиною нашого життя. Вони активно розвиваються та знаходять застосування майже у всіх сферах нашого життя. Це створило потужний ґрунт для утворення нової інформаційної інфраструктури. Ця інфраструктура дуже тісно пов'язана з використанням нових інформаційних технологій у суспільних відносинах та процесах.

Однією з популярних тенденцій використання штучного інтелекту є його використання в інтелектуальних системах. Такі системи можуть виконувати функції, які вважаються інтелектуальними, тобто які базуються на принципах навчання, накопичення знань, їх використання та підведення логічного висновку.

Останні декілька років активно почали використовуватись системи розпізнавання образів. Сучасні технології дозволяють навчати інтелектуальні системи на великих об'ємах даних, що призводить до більш точного розпізнавання. Так званий машинний зір використовується у багатьох сферах життя: розважальної, промислової, інфраструктурної.

Проблеми з функціональним станом доріг та узбіч одна з основних інфраструктурних проблем нашої держави. Обслуговування та ремонт доріг та узбіч коштує не дешево. Постійний рух автомобілів і погодні умови знищують дорожнє покриття. Вода та технічні рідини потрапляють в тріщини на покритті і з часом призводять до більш серйозних проблем. Наслідками цього можуть бути затори та аварії, на ремонтах доріг відмиваються державні кошти, сміття на узбіччях забруднює природу.

Останнім часом все більше і більше проводяться дослідження щодо створення та вдосконалення систем моніторингу функціонального стану доріг. Деякі системи використовують візуальне спостереження, деякі використовують сенсорне рішення.

Оскільки моніторинг дорожнього покриття – ключовий фактор в забезпеченні безпечної дорожньої інфраструктури для учасників дорожнього

руху, а моніторинг стану узбіч – забезпеченні чистоти природи, то метою цієї роботи буде створення високоточної інтелектуальної системи автоматичного виявлення проблем з дорожнім покриттям та станом узбіч.

1. Аналіз проблеми та постановка задачі

1.1 Сучасні технології моніторингу стану доріг та узбіч

На сьогоднішній день дуже гостро стоїть питання про вирішення проблеми з дорожнім покриттям. Збільшується кількість приватних автомобілів, транспорту для пасажиро та товаро перевезень, будується все більша кількість нових доріг за сучасними стандартами, старі дороги та шосе ремонтуються. Дороги потребують постійної уваги, від етапу будування до постійного функціонування. Очевидно, що існує необхідність постійного спостереження за їх експлуатаційним станом.

У транспортній галузі існує спеціальна система, що забезпечує контроль та діагностику стану автомобільних доріг. Як правило, їх якість оцінюється інструментальними засобами, приладами, пристроями та системами. У світовій практиці для оцінки стану та діагностики автомобільних доріг розроблені потужні інформаційно-вимірювальні системи з високим рівнем автоматизації. В Україні функції оцінки стану доріг виконує науково-технічний центр «Дор'якість».

Слід зазначити, що на утримання доріг і так витрачаються великі кошти та людські ресурси, а будь-яка оцінка їх стану буде потребувати ще більше, особливо якщо врахувати, що такий нагляд за їх функціональним станом потрібно буде проводити постійно.

Не треба забувати і про узбіччя. Засмічення лісосмуг, придорожніх смуг та узбіч доріг твердими побутовими відходами є чи не найбільшою екологічною проблемою нашої держави. Для ліквідації таких звалищ потрібні неабиякі фінансові та людські ресурси. Проблема стоїть не в їх нестачі, а в тому, що на тих самих місцях утворюються нові звалища. За межами населених пунктів дороги обслуговуються Службою автомобільних доріг, і замість того, щоб ремонтувати автошляхи, служба змушена вимушена витрачати ресурси на покращення стану узбіч доріг. Санітарний стан доріг та

узбіч не тільки свідчить про культурний рівень людей, а ще й може впливати на безпеку дорожнього руху. Також не треба забувати про негативний вплив відходів на навколишнє середовище.

Поряд з детальним обстеженням автомобільних доріг за допомогою спеціальних інформаційно-вимірювальних систем необхідні порівняно просто методи, засоби та системи, які постійно оцінюють функціональний стан доріг без значних витрат.

На практиці приміняються різні методи та технологій оцінки стану дороги за значенням деяких параметрів, наприклад за зчепленням з дорожнім покриттям, за рівністю, за коефіцієнтом ковзання, та геометричними параметрами. Майже всі методи та технології передбачають використання інструментальних засобів. Сама технологія такої діагностики передбачає використання пересувних дорожніх лабораторій.

Для визначення функціонального стану автомобільних доріг у польових умовах часто використовуються пересувні лабораторії. Вони являють собою багатофункціональні вимірювальні комплекси, встановлені у звичайний автомобіль. Такі лабораторії дозволяють в автоматичному чи напівавтоматичному режимах отримувати геометричні параметри елементів доріг: коефіцієнти зчеплення, показники рівності, показники міцності, а також відеоінформацію про дороги.

Зараз існують пересувні дорожні лабораторії, обладнані сучасним обладнанням для оцінки рівності покриття, лазерними сканерами, системами GPS для визначення місцеположення.

Найцікавішим є створення мобільної дорожньої лабораторії «Мудрець» - мобільний вуличний дорожній рецептор (Рисунок 1).

Рисунок 1.1 - Мобільна дорожня лабораторія "Мудрець"

Ця лабораторія призначена для комплексного для комплексного інструментального діагностування шляхів сполучення, оцінки якості дорожнього покриття разом із реєстрацією кількісних характеристик та складу транспортних потоків [1].

Наступним витком еволюції моніторингу стану дорожнього покриття є інтелектуалізація інструментальних засобів. Саме вона покладена в основу подальшого розвитку інструментальних засобів дослідження за станом доріг.

Інтелектуалізація полягає в логічному поєднанні процесів, що базуються на основі трьох технологій: інформаційній, механічній, електронній [2]. Об'єднання цих технологій базується на застосуванні телекомунікаційних та комп'ютерних технологій, уніфікації та стандартизації. Телекомунікаційні та комп'ютерні технології забезпечують інформаційні послуги. Уніфікація – це забезпечення універсальних рішень. Стандартизація – використання загальних протоколів передачі даних.

Інтерактивний моніторинг є вдалим рішенням та логічним втіленням інтелектуалізації інструментальних засобів моніторингу доріг. Йому притаманні властивості режиму спостереження та діагностування, безпосередня оцінка функціонального стану доріг.

Моніторинг на основі датчиків смартфонів. В основу цього напрямку вирішення проблеми моніторингу стану доріг лягла концепція використання вбудованих датчиків смартфонів. Сучасні платформи смартфонів мають доступ до широкого кола вбудованих сенсорів. Деякі з датчиків є апаратними (фізичними), деякі – програмними (віртуальними). Фізичні датчики можуть вимірювати швидкість руху, орієнтацію і умови навколишнього середовища, освітлюваність, фізичне положення пристрою, наприклад акселерометр, гіроскоп, магнітометр тощо. Програмні ж датчики навпаки використовують дані від одного або декількох фізичних і на основі отриманих даних в режимі реального часу обчислюють необхідні значення, наприклад лінійне прискорення, силу тяжіння тощо.

Для методів, в основі яких лежить використання камери смартфону, використовується сам смартфон, закріплений відповідним чином для зйомки дорожнього покриття на шляху руху автомобіля. Далі отримані зображення аналізуються за допомогою алгоритмів аналізу. Наприклад, у статті [3] описано систему Video-based Pavement Distress Screening. Система аналізує відеоряд, а потім на основі цих даних знаходить проблемні ділянки на дорогах. Також системи на основі методу аналізу зображень та відео були описані у статті [4]. Для ефективного виявлення проблемних місць використовуються алгоритми глибокого навчання.

В методах, в основі яких лежить принцип вимірювання вібрацій, частіше всього використовують акселерометри та гіроскопи вбудовані у смартфон, оскільки вони чуттєві до скачків, викликаних аномаліями в дорожньому покритті, а також систему навігації GPS для визначення місцезнаходження. В статті [5] було описано системи на основі таких датчиків. Їх можна розділити на 2 основні категорії: методи на основі порогових значень, динамічне перетворення часу та машинне навчання разом з функцією проектування. Підходи основані на порогових значеннях виявляють аномалії коли амплітуда або інші значення даних перевищують задане значення. Для підходів з динамічним перетворенням часу

використовують алгоритми порівняння двох послідовностей, одна з яких отримана в результаті вимірювання, інші – еталонна. Методи машинного навчання здобули найбільшу популярність. В основі їх лежить алгоритм розпізнавання пошкоджень дорожнього покриття по декільком фільтрах (швидкість, верхні частоти, тощо). Пороги кожного фільтру використовуються в якості параметрів налаштування для оптимізації. Шляхом кластеризації результати таких методів досягли кінцевої точності виявлення аномалій у 92%.

Методи на основі лазерного сканування. Добре відомо, що характеристики дорожнього покриття, особливо його верхнього прошарку, є необхідними для задоволення ряду вимог, наприклад безпека користувачів та шумове забруднення. Дорожні виступи оцінюються і кваліфікуються шляхом прийняття параметрів, які впливають на різні характеристики дороги, наприклад текстура поверхності дороги, або стик колеса машини та дороги (Рисунок 1.2).

Рисунок 1.2 – Приклад параметрів кваліфікування дорожньої поверхні (текстура дороги та стик дорога-колесо)

Останні декілька років на ринку з’явився широкий спектр інструментів для 3D-цифрування [6]. Також ця технологія добралась і до завдання моніторингу стану дорожніх поверхонь. В цьому питанні найчастіше почала використовуватись система лазерного сканеру з триангуляцією [7]. Ці системи

в останні роки зарекомендували себе як надійна технологія для безконтактних вимірів. Лазерна триангуляція – це активна стереоскопічна техніка побудована на принципі топографічного перетину, здатна визначити положення точки в просторі за допомогою довідкової системи.

Відповідно до Рисунку 1.3, лазерний випромінювач створює промінь енергії, який посилається під деяким кутом, який попередньо відомий завдяки попередній калібровці дзеркала, яке може обертатися. Він потрапляє на поверхність об’єкту в точці та відбивається. Кут відбиття залежить від матеріалу цільової поверхні. Частина відбитого сигналу потрапляє на приємний датчик, який знаходиться на відомій відстані. Кут падіння відбитого променю невідомий, але його знаходять за допомогою тригонометричних формул, фокусній відстані та положенню лазерної плями на сканері. Повторюючи цю операцію для всіх точок, в яких можна дискретизувати поверхність об’єкту, можна визначити їх координати, а потім і дискретизувати поверхність через сукупність точок.

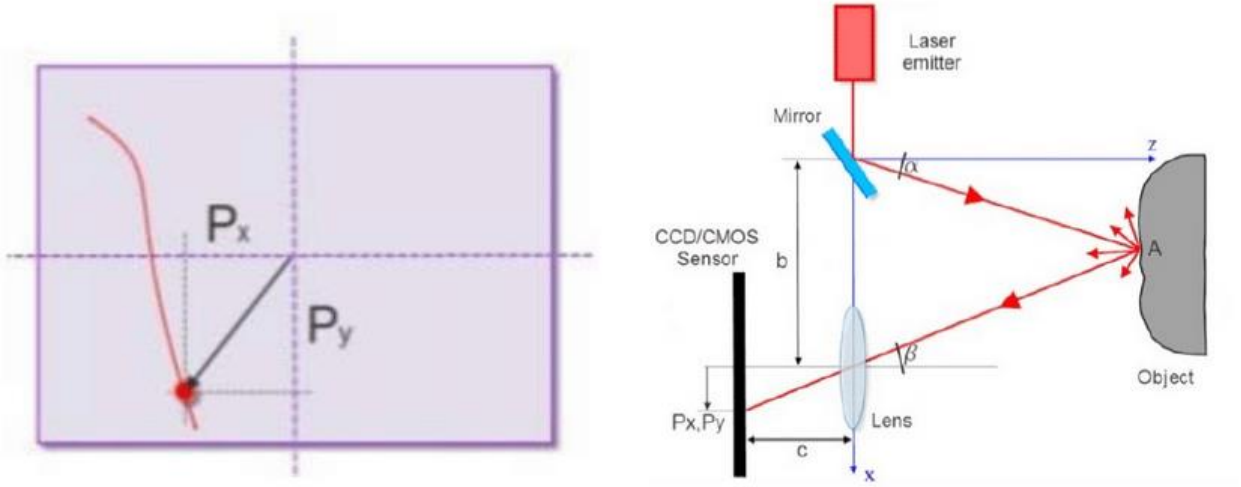

Рисунок 1.3 – Схема системи лазерної тріангуляції

Після процесу тріангуляції сукупність точок можна перетворити в сітку з трикутників, з яких складається 3D поверхня об’єкту.

Функціонування лазерних сканерів часто відхиляється від ідеальних умов, оскільки є певна кількість джерел похибок, які негативно впливають на продуктивність таких сканерів, викликаючи зниження точності, або навіть

повну відмову функціонування. До них відносяться неправильне освітлення, форму об’єкту, його колір, або навіть характеристика поверхні.

Спираючись на вищеописані дані, можна зробити висновок, що моніторинг функціонального стану доріг та узбіч досить актуальна та складна задача, оскільки було запропонована багато способів її вирішення, але результат залишає бажати кращого. Також вищеописані рішення потребують доволі багато матеріальних ресурсів та часу. Можна запропонувати альтернативу цим рішенням – використання технології глибоких нейронних мереж та технології машинного зору для вирішення цієї проблеми. Одним із рішень цієї проблеми може стати інтелектуальна інформаційна система.

Це рішення може спростити відповідні процедури діагностування стану дороги та процес обстеження, на порядок підвищити ефективність моніторингу дорожнього покриття. Інтерактивна оцінка та оперативна діагностика функціонального стану дороги, а також її узбіч, може зменшити в рази кількість затраченого часу і одночасно збільшити якість доріг у декілька разів.

1.2 Аналіз моделей та методів штучного інтелекту для прийняття рішень на основі візуальних даних

У сучасному розумінні термін «штучний інтелект» можна трактувати як науковий напрям, в рамках якого ставляться та розв’язуються завдання апаратного й програмного моделювання тих видів людської діяльності, які традиційно вважаються інтелектуальними, тобто потребують певних розумових зусиль.

Протягом процесу розвитку штучного інтелекту було розроблено багато додатків, які застосовуються в різних галузях науки та техніки. Найпопулярнішим типом систем, які в них використовуються, є нейронні мережі. Нейронна мережа – це надзвичайно спрощена модель нервової

системи людини, яка може імітувати такі здатності людини, як навчання, узагальнення та абстрагування.

Одним з основних завдань, які виконують нейронні мережі, є прийняття рішень на основі візуальних даних, наприклад розпізнавання об’єктів на зображені. Технології глибокого навчання стали популярними методами для розпізнавання об’єктів. Моделі глибокого навчання, такі як конволюційні нейронні мережі, або CNN, використовуються для автоматичного вивчення властивостей об’єкту для його ідентифікації.

Існує два підходи для розпізнавання об’єктів: статичний та структурний. Також є гібридні підходи, інколи вони називаються уніфікованими, які об’єднують методики від статичного та структурного підходів. Кардинальною різницею між цими двома підходами є опис. Опис сформований статистичним підходом є кількісним, тоді як опис створений структурним підходом складається з підшаблонів або будівельних блоків. Також перший підхід використовує чисельні відмінності між ознаками різних класів, другий – охоплює прийняті налаштування для кожної з груп.

Ознаками називаються будь-які відмінні аспекти, або характеристики, об’єкту. Шаблон – комбінація ознак, характерних для певної групи об’єктів (класу). Вектором властивостей називається вектор d – розмірного стовпця. Простір ознак – це D-мірний простір, визначений функцією вектора. Подання об’єктів у вигляді точок на цьому просторі називається графіком розсіювання. Якість вектора властивостей має зв’язок зі здатністю відрізняти об’єкти з різних класів. Об’єкти з одного класу мають подібне значення ознак, з різних – різне значення.

У системах розпізнавання об’єктів можна виділити три завдання, які потрібно виконати для правильного функціонування: попередня обробка, опис властивостей, класифікація [8].

Завданням класифікації полягає в тому, щоб присвоїти об’єкту певну категорію (мітку класу), використовуючи векторний функціонал. Також варто зазначити, що часто бувають випадки, при яких повна класифікація не

можлива, тому за таких обставин завдання перетворюється на визначення ймовірності кожної з можливих груп.

Завдання класифікації зображень – присвоєння конкретному образу мітки класу з обмеженого набору за конкретними ознаками. В інтелектуальних інформаційних системах завдання класифікації об’єкту часто стоїть поруч з завданням його локалізації. Завдання локалізації об’єкту – виявлення місцезнаходження об’єкту на зображенні. Завдання виявлення – це одночасна класифікація об’єкту міткою певного класу та його локалізація на зображенні контурною рамкою.

В якості класифікатора може виступати нейронна мережа. Нейронні мережі – це набір нейронів, пов’язаних у графі. Цикли в таких графах не дозволяються.

Моделі нейронних мереж організовуються в шари нейронів. У звичайних мережах найчастіше можна зустріти повністю з'єднаний шар, в

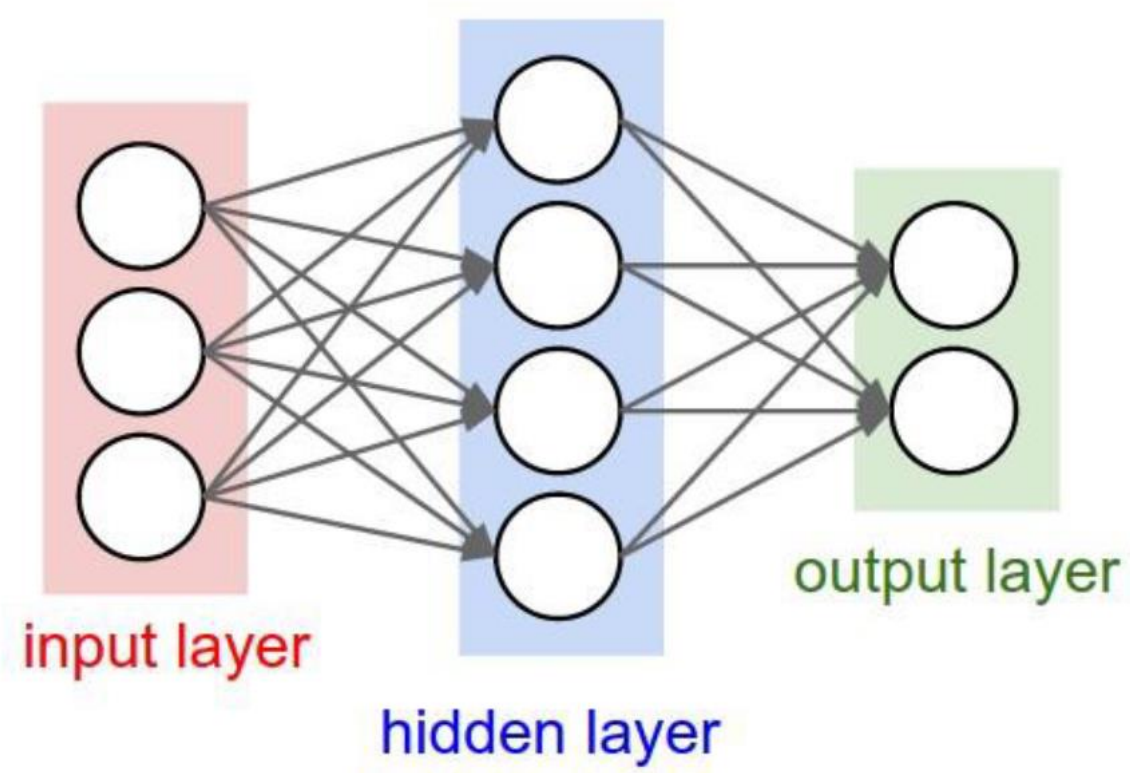

Рисунок 1.4 – Приклад шарової нейронної мережі

якому нейрони з сусідніх шарів попарно з’єднані, але в одному шарі ніяких зв’язків немає (Рисунок 1.4).

Багатошарові мережі можуть будуватися з шарів, які складаються каскадами. Виходом одного шару може бути вхід до наступного.

Також часто використовуються рекурентні нейронні мережі. Вони найчастіше приміняються в завданнях, які пов’язані з послідовностями, наприклад текстовими, голосовими, тощо.

Для завдань з обробки візуальної інформації найчастіше використовуються згорткові нейронні мережі. Від самого початку вони націлені на ефективне розпізнавання зображень. Вони схожі на звичайні шарові мережі, але їх основною відмінністю є декілька етапів виділення ознак. Типова архітектура таких мереж зазвичай складається з шарів згортки та об’єднання, які чергуються між собою, за ними слідує один або декілька повноз’єднаних шарів (Рисунку 1.5).

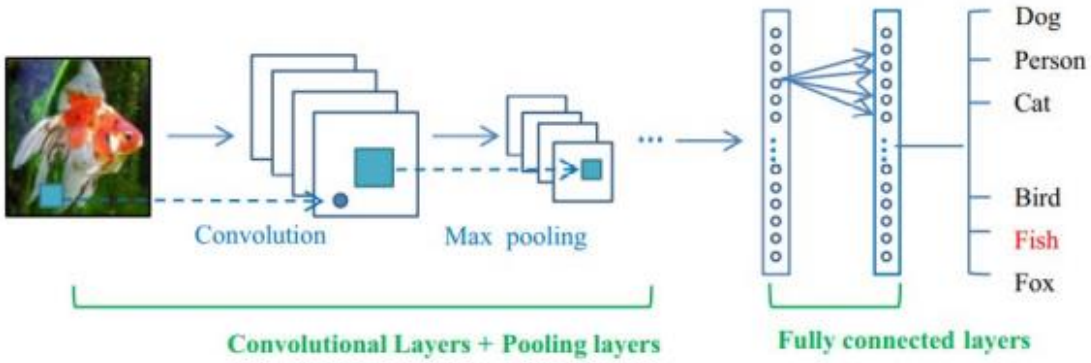

Рисунок 1.5 – Загальна архітектура згорткової нейронної мережі

Згортковий шар мереж такого типу складається з набору ядер, в якому кожен нейрон виконує фільтруючу роль, розділяючи зображення, яке аналізується, на частини, які називаються рецептивними полями. Розділення зображень на фрагменти має свою функцію – витягнення характерник ознак. Рецептивне поле ковзає по всій площі вхідного зображення. Розмір цього поля одночасно є вхідним розміром нейронів в згортковий шар. Виходом фільтрів в кожному шарі виступає карта ознак, яка передається в наступний шар в якості вхідного сигналу.

Наступним за згортковим йде об’єднуючий шар і використовується для зменшення розмірів карт ознак [9]. В кінці мережі знаходиться повноз’єднаний шар, він виконує задачу класифікації. Цей шар отримує вхідні дані з етапів витягнення ознак і аналізує вихід усіх попередніх шарів. Результатом його роботи є нелінійна комбінація ознак, на основі якої зображення класифікується.

До появи спеціальних нейронних мереж для детектування об’єктів заданих класів зазвичай використовувались прості згорткові нейронні мережі [10]. Але з активним розвитком технологій глибоких нейронних мереж на допомогу прийшли штучні згорткові нейронні мережі з новими революційними архітектурними підходами. Це зумовлено рядом причин :

- Прогрес у сфері розробки графічних процесорів
- Великий об’єм даних для навчання
- Кращі результати в порівнянні з класичними підходами

Нові архітектурні підходи можна розділити на дві основні групи: архітектури, які аналізують регіони на зображеннях (R-CNN), та архітектури, які аналізують ціле зображення (YOLO, SSD).

У роботах [11, 12] представлена архітектура детектора об’єктів YOLO (You Only Look Once). Ця архітектура початково розроблялась для задач реального часу. В цьому алгоритмі зображення, яке аналізується, розбивається на комірки з використанням сітки. Для кожної комірки оцінюється вірогідність наявності шуканого об’єкту, після цього будуються декілька найбільш вірогідних положень об’єкту у вигляді прямокутників з центром в даній комірці, потім для кожного з прямокутників виконується оцінка вірогідності наявності у ньому об’єктів для кожного класу. Оцінка вірогідності наявності об’єкту конкретного класу в конкретному прямокутнику – це добуток оцінки вірогідності знаходження об’єкту в комірці та оцінки вірогідності для конкретного класу.

В випадку модифікації YOLOv3 для виділення ознак використовується згорткова нейронна мережа, яка складається з 53 шарів. Також варто

зазначити, що в YOLOv3 детектування об’єктів відбувається на трьох масштабах, що дозволяє збільшити якість детектування невеликих об’єктів.

Відомо, що більшість алгоритмів розпізнавання роблять вихідні мітки взаємовиключними. В архітектурах YOLOv1 та YOLOv2 використовуються функції для перетворення оцінок у вірогідності класів, сума яких по всім класам дає одиницю [13].

Однією з архітектур, які найбільш часто використовуються для вирішення задач детектування, на основі глибокого навчання є Region-based CNN (R-CNN) [14]. Принцип роботи такої архітектури базується на трьох етапах:

1. Зображення розбивається на регіони, в яких можуть знаходитись шукані об’єкти.
2. Кожний регіон подається на вхід відповідно навченій згортковій нейронній мережі, яка витягує вектор ознак для свого регіону.
3. Вектор ознак подається на вхід класифікатор.

Додатковим кроком можна вважати подавлення немаксимумів для виключення зайвої кількості прямокутників, які охоплюються один і той самий об’єкт.

Такий архітектурний тип показав доволі високі показники точності детектування об’єктів, але були і певні недоліки: високі затрати пам’яті та часу на навчання та обробку зображень.

Щоб виправити недоліки цього підходу, було запропоновано модифікацію Fast R-CNN. Головною відмінністю цієї модифікації є те, що нейронна мережа використовується тільки один раз для аналізу всього зображення замість аналізу багатьох областей, які перетинаються. Новий підхід виграє у швидкості, але залишається недолік алгоритму пошуку регіонів-кандидатів. Виправлення цього недоліку потягло за собою створення нової модифікації Faster R-CNN.

На сьогодні архітектура Faster R-CNN дозволяє домогтися високої точності детектування і вважається відносно швидкою. При цьому було

збережено головну ідею: виділення на зображенні регіонів, в яких є вірогідність знаходження об’єктів, та класифікація їх вмісту.

Архітектура SSD (Single Shot MultiBox Detector) забезпечує значний приріст швидкості обробки відносно архітектури R-CNN [15]. Якщо остання виконує вибір регіонів-кандидатів та їх класифікацію в два окремі етапи, то SSD виконує ці дії одночасно при обробці всього зображення. Принцип роботи такої архітектури можна описати наступним чином:

1. Вхідне зображення проходить через ряд згорткових шарів, що в результаті дає набір карт ознак для різних масштабів.
2. В кожній точці кожної карти ознак використовується згортковий шар для отримання множини прямокутників, які описують зображення.
3. Для кожного прямокутника одночасно оцінюються просторове зміщення та вірогідність знаходження об’єкту.
4. В процесі навчання істинні прямокутники зіставляються з передбаченими для виключення хибних детектувань.

На відміну від R-CNN, де в регіонах-кандидатах є хоча б мінімальна вірогідність находження об’єкту, в SDD крок фільтрації регіонів відсутній. В результаті чого генерується набагато більша кількість прямокутників опису на різних масштабів відносно R-CNN, більша частина з яких не містить шуканих об’єктів. З метою вирішення даної проблеми в архітектурі цього типу використовується подавлення немаксимумів для об’єднання схожих один на одного прямокутників в один. Також використовується техніка hard negative mining, згідно з якою на кожній ітерації навчання використовується тільки частина негативних прикладів [16].

1.3 Постановка задачі

На основі вищеописаної інформації, що стосується сучасним технологій та методів моніторингу стану доріг та узбіч, а також методів обробки візуальних даних та сучасних моделей нейронних мереж, можна зробити

висновок, що процеси моніторингу можна значно полегшити за допомогою нейронних мереж.

У зв’язку з цим основною метою цієї роботи буде проектування та створення інтелектуальної технології, яка буде проводити моніторинг функціонального стану дорожнього покриття та дорожніх узбіч.

Технологія має отримувати на вхід дані у вигляді зображень дорожнього покриття в нормальному стані, а також з явними та неявними дефектами, а також зображення засмічених та чистих дорожніх узбіч.

Технологія має працювати в режимі екзамену. При роботі системи в режимі екзамену, тобто саме в робочому режимі, система приймає рішення про присвоєння аналізованого образу певного класу із заданого алфавіту на основі побудованих на етапі навчання правил.

Виконання задачі має включати в себе такі етапа:

1. Вибір моделі нейронної мережі для аналізу зображень.
2. Підготовка даних для навчання моделі.
3. Тестування роботи системи в режимі екзамену.

2. Інтелектуальна технологія моніторингу функціонального стану доріг та узбіч

2.1 Модель нейромережі для детектування пошкоджень дорожнього покриття та сміття на узбіччі

За останні декілька років нейромережі дістались до усіх галузь машинного навчання, але найбільший успіх вони зробили у галузі машинного зору. Свою ефективність згорткові нейронні мережі для детектування об’єктів на зображеннях довели на практиці, але їх використання для моніторингу стану інфраструктурних чи муніципальних об’єктів поки ще не набуло великої популярності.

Якщо декілька років тому нейромережі вважались «важкими» алгоритмами, то сьогодні нікого не здивувати такими алгоритмами, які працюються навіть на мобільних телефонах, як наприклад з архітектурним підходом MobileNet [17].

Основним принципом відмінності MobileNet від інших архітектурних підходів є розділення шару згортки на окремі шари. На рисунку 2.1 зображено відмінність шару згортки MobileNet від звичайної CNN.

Також особливістю такої архітектури являється відсутність max pooling шарів. Замість них для зниження просторової розмірності використовується згортка з параметром stride.

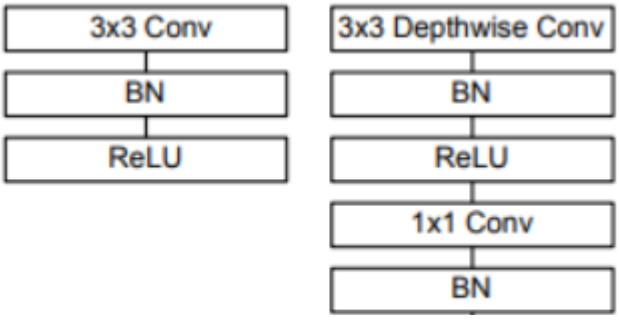

Рисунок 2.1 – Порівняння архітектури шару згортки звичайної CNN та MobileNet

Двома гіперпараметрами архітектури MobileNet являються α (множник ширини) та ρ (множник глибини або множник розширення). Множник ширини відповідає за кількість каналів в кожному шарі. Множник розширення відповідає за простірні розміри вхідних тензорів. Вони обидва дозволяють варіювати розміри мережі: зменшуючи α та ρ, знижується точність розпізнавання, але в той же час збільшується швидкість роботи і зменшується використовувана пам’ять.

Еволюцією архітектури MobileNet стала його модифікована версія – MobileNetV2. Як і свій попередник ця модифікація включає в себе згорткові блоки з кроками 1 та 2 (Рисунок 2.2).

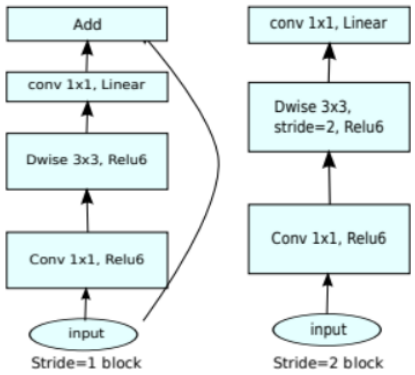

Рисунок 2.2 – Згорткові блоки MobileNetV2 з кроком 1 (зліва) та кроком 2 (справа)

Блок з кроком 2 призначений для зниження просторової розмірності тензору та, на відміну від блоку з кроком 1, не має residual connections. Сам блок MobileNetV2 називається розширювальним згортковим блоком і складається з трьох шарів:

1. Спочатку йде pointwise convolution з великою кількістю каналів, його ще називають expansion layer.

   На вході цей шар приймає тензор розмірності $D_f * D_f * C_{in}$ (D_f та D_f - розміри ядра згортки, C_{in} – кількість вхідних каналів), а на виході видає тензор $D_f * D_f * (t * C_{in})$, де t – новий гіперпараметр, який називається

рівнем розширення. Цей шар створює відображення вхідного тензору в просторі великої розмірності.

2. Далі йде depthwise convolution.

   На вході цей шар приймає тензор розмірністю $D_f * D_f * (t * C_{in})$, а на виході видає тензор $(D_f / s) * (D_f / s) * (t * C_{in})$, де s – це крок згортки.

3. В кінці йде 1x1 згортка з лінійною функцією активації, яка знижує число каналів. Приймає на вхід тензор з попереднього виходу, а на вихід видає тензор $(D_f / s) * (D_f / s) * C_{out}$, де C_{out} – кількість каналів на виході з блоку.

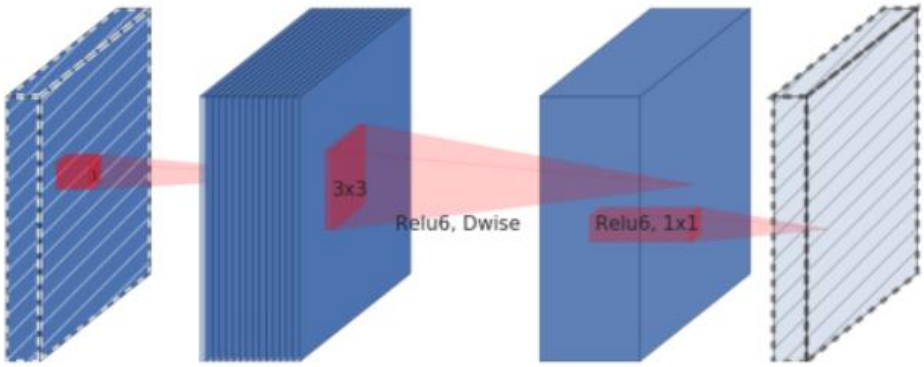

Рисунок 2.3 – Згортковий блок MobileNetV2

Дослідження показали, що вкладення відображення тензору з високою розмірністю в простір з меншою розмірністю не тільки можливе, але й не призводить до втрати інформції [17]. Ці ж дослідження показали високу ефективність взаємодії архітектурних підходів MobileNet, з її згортковими блоками, та SSD для задач класифікації та локалізації об’єктів на зображеннях.

Отже для ефективного вирішення проблеми моніторингу функціонального стану дорожнього покриття та узбіч можна обрати модель, яка поєднує в собі архітектури MobileNetV2 та SSD, як легку та ефективну модель нейромережі.

2.2 Навчання моделі нейронної мережі

Критерієм ефективного навчання нейромережі для детектування об’єктів є правильно підібрані дані. Часто для розширення набору даних використовується підхід аугментації даних. Також часто датасет розділяють

дві вибірки: навчальну та контрольну. Це робиться для того, щоб уникнути проблеми перенавчання, коли вже навчена модель ефективно працює лише на навчальній вибірці. Ці два набори даних повинні мати приблизно однаковий розподіл класів.

Також важливим критерієм ефективного навчання нейромережі є вимірювання її продуктивності. Для того щоб визначити наскільки передбачення відрізняється від оптимального рішення використовується функція втрат. Функція втрат являє собою звичайну функцію, параметрами якої є навчальні параметри моделі. Також ця функція надає моделі дані про те як продовжувати навчання та передбачувану точність. В задачах детектування обчислення втрат повинно включати в себе втрати локалізації для зміщення обмежуючої рамки та втрати класифікації. Для обчислення втрат найпопулярнішою є функція знаходження середньоквадратичної помилки (сума квадратів помилок):

$$L(x,t,\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta\left(x^i\right) - y^i\right)^2 \tag{2.1}$$

, де x – проноз моделі, y – очікуваний прогноз.

Для моделей з високою точністю передбачення функція втрат повинна бути мінімально, для моделей з низькою – високою.

Для кращого навчання модель потребує оптимізації цього процесу. Метод градієнтного спуску – найважливіший метод для мінімізації функції втрат. Оновлення параметрів навчання моделі за допомогою метода градієнтного спуску обчислюється за формулою:

$$\theta = \theta - \mu\nabla J(\theta) \tag{2.2}$$

, де μ – темп навчання, $\nabla J(\theta)$ – це градієнт функції втрат $J(\theta)$.

Метод градієнтного спуску є найпопулярнішим методом оптимізації нейронної мережі. В основу він використовується для оновлення ваг. Методика зворотного розповсюдження, в якій спершу розповсюджується добуток вхідних сигналів та відповідних ваг, а потім застосовується активаційна функція, яка перетворює сигналі, які в неї потрапляють, на

вихідні, дозволяє при навчанні засвоїти більшість довільних функцій відображення.

2.3 Оцінка точності детектування

Критично важливим є обраний підхід оцінювання результатів детектування, оскільки вхідні дані для навчання мережі не є збалансованими. Для оцінки якості роботі навченої моделі найчастіше використовуються такі параметри: точність, F-міра, повнота та кількість входжень класу в істинному наборі даних.

Також для оцінки точності класифікації використовується матриця невідповідностей (confusion matrix) (Рисунок 2.4).

Рисунок 2.4 – Приклад confusion matrix

В цій матриці для кожного класу показано кількість результатів прогнозування, віднесених до конкретного класу. Вона заповнюється такими показниками:

- Істинно позитивні випадки – кількість правильних класифікацій (головна діагональ).
- Хибно позитивні випадки – кількість класифікацій певного класу, які насправді відносяться до іншого.

- Хибно негативні випадки – кількість класифікацій певного класу, коли модель не змогла його розпізнати.

На основі вищеописаних показників розраховуються параметри оцінки якості роботи моделі. Влучність – здатність класифікатора виявляти тільки правильні об’єкти конкретного класу. Визначається за формулою:

$$precision = \frac{T_p}{T_p + F_p} \tag{2.3}$$

, де T – кількість коректних визначень, F – кількість всіх визначень цього конкретного класу.

Повнота – відображення відношення коректно визначених об’єктів конкретного класу до всієї кількості об’єктів цього класу.

$$recal = \frac{T_p}{T_p + F_n} \tag{2.4}$$

, де T – кількість коректних визначень, F – кількість всіх об’єктів цього класу.

Повнота та влучність розраховуються подібним чином, але ці дві міри оцінюють роботу моделі у різних відношеннях.

Точність – міра, яка показує відношення істнних позитивних передбачень до їх загального числа:

$$accuracy = \frac{T_p}{N} \tag{2.5}$$

, де T – кількість коректних визначень, N – кількість всіх визначень.

Бувають випадки, коли дані не є збалансованими. Тоді якщо один клас має кількісну перевагу і отримує точні визначення, а інший – погані показники якості, то обчислення точності не будуть враховувати незбалансованість даних. До того ж показники повноти та влучності не дають повного розуміння ефективності моделі. Для цього використовується їх поєднання – F1- міра.

$$F1 = 2 * \frac{precision * recal}{precision + recal} \tag{2.6}$$

Для оцінки точності визначення локалізації об’єкту використовується показник IoU. Ця міра показує ступінь перекриття між обмежуючою рамкою,

яка передбачена системою детектування, та між рамкою, поміченою на вхідному зображенні (Рисунок 2.5).

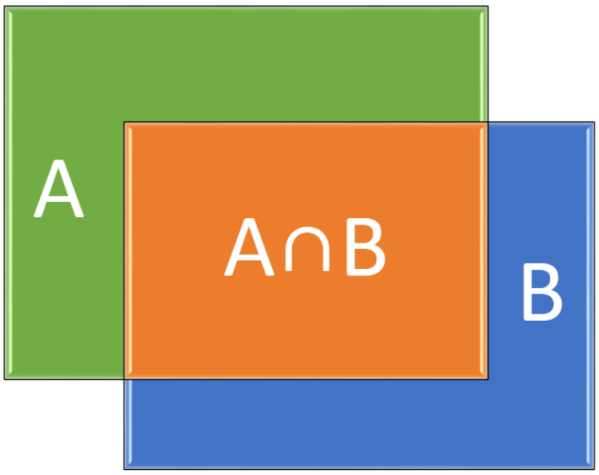

Рисунок 2.5 – Графічне представлення процесу визначення IoU. (Зелене поле – результат визначений системою, синє – помічений результат)

$$IoU = \frac{S(A \cap B)}{S(AUB)} \quad (2.6)$$

, де S(A∩B) – пересічення результатів детектування та розмічених даних, S(AUB) – об’єднання результатів детектування та розмічених даниї.

Отже серед багатьох оцінок якості роботи моделі найбільш інформативними є оцінка через повноту, точність та влучність. Для візуального зображення результатів прогнозування використовується confusion матриця.

3. Реалізація інтелектуальної технології моніторингу функціонального стану

3.1 Створення вибірки навчальних та тестових даних

Для проходження моделлю процесу навчання та для тестування її роботи необхідно сформувати набір візуальних даних, який складається з зображень дорожнього покриття з явними чи неявними дефектами, а також зображень засмічених узбіч. Також необхідно розподілити дані на навчальні та тестові у відсотковому відношенні 75% та 25% відповідно. Тестові дані будуть використані для оцінки якості детектування моделлю та будуть сховані на період її навчання.

Для навчання було сформовано вибірку даних, в яку входять 9053 зображення шуканої тематики з багатьох країн світу (Рисунок 3.1).

Рисунок 3.1 – Приклад зображень з вибірки

Також було сформовано алфавіт класів для опису всіх можливих пошкоджень дорожнього покриття та засміченості узбіч (Таблиця 3.1).

Таблиця 3-1 – Алфавіт класів розпізнавання

<table>
<tr><th colspan="3">Тип ушкодження</th><th>Опис</th><th>Назва класу</th></tr>
<tr><td rowspan="5">Тріщина</td><td rowspan="3">Лінійна тріщина</td><td rowspan="2">Продольна</td><td>Слід від колеса</td><td>D00</td></tr>
<tr><td>Подряпина</td><td>D01</td></tr>
<tr><td rowspan="2">Кругова</td><td>Рівний інтервал</td><td>D10</td></tr>
<tr><td>Подряпина</td><td>D11</td></tr>
<tr><td colspan="2">Нелінійна тріщина</td><td>Повне та неповне покриття</td><td>D20</td></tr>
<tr><td colspan="3" rowspan="2">Інші пошкодження</td><td>Нерівність, вибоїна, відрив</td><td>D40</td></tr>
<tr><td>Слід від розмітки</td><td>D44</td></tr>
<tr><td colspan="3">Сміття</td><td>Всі види сміття</td><td>Trash</td></tr>
</table>

Було сформовано розмітку та анотації до шуканих об’єктів. Анотації записані у форматі Pascal VOC XML. Оскільки завдання передбачає детектування всіх шуканих об’єктів на класі, то на одне зображення може мати декілька дефектів і, відповідно, анотацій до них. Приклад розмітки дефекту на зображенні представлено на рисунку 3.2.

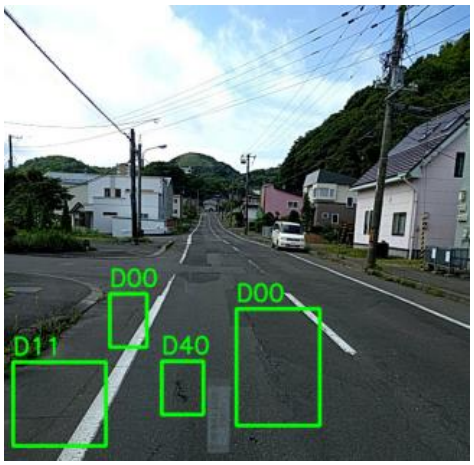

Рисунок 3.2 – Приклад розмітки зображення

В результаті було сформовано датасет з наступним балансом даних (Див. таблицю 3.2).

Таблиця 3.2 – Розподіл об’єктів класу

Клас	Кількість об’єктів
D00	2768
D01	3789
D10	742
D11	636
D20	2541
D40	409
D43	817
D33	3733
Trash	1746

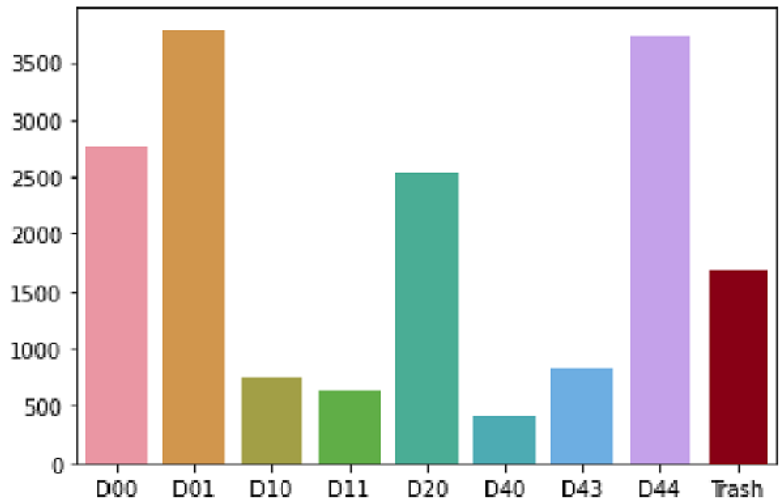

Рисунок 3.3 – Графічне представлення розподілу класів у вибірці

Оскільки проблема з нестачею зображень не має, то підхід з аугментацією даних використовуватись не буде.

3.2 Результат навчання моделі

Для детектування дефектів дорожнього покриття та звалищ сміття на узбіччі було обрано згорткову нейронну мережу, а саме удосконалений архітектурний підхід SSD MobileNetV2. У ході навчання використовувався фреймворк TensorFlow Object Detection Api. Для оцінки якості продуктивності навчання моделі використовувалась функція втрати локалізації та класифікації. Для оптимізації параметрів навчання використовувався метод градієнтного спуску.

Всього на етапі навчання було пройдено 20000 тисяч епох. Під епохою розуміється етап навчання, коли через вхід пройшов весь набір зображень. Після проходження усіх епох навчання, було отримано графіки змін втрат класифікації та її точності (Рисунок 3.4) і локалізації та її точності (Рисунок 3.5) відповідно на етапах навчання та тестування.

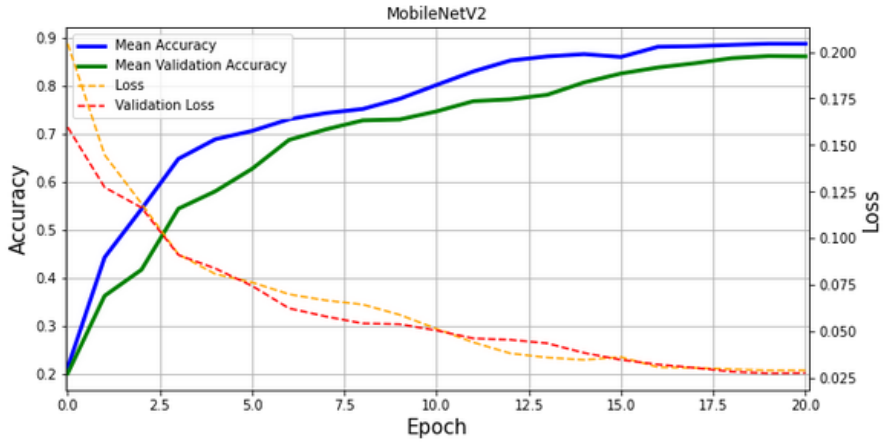

Рисунок 3.4 - Графік зміни втрат та точності класифікації

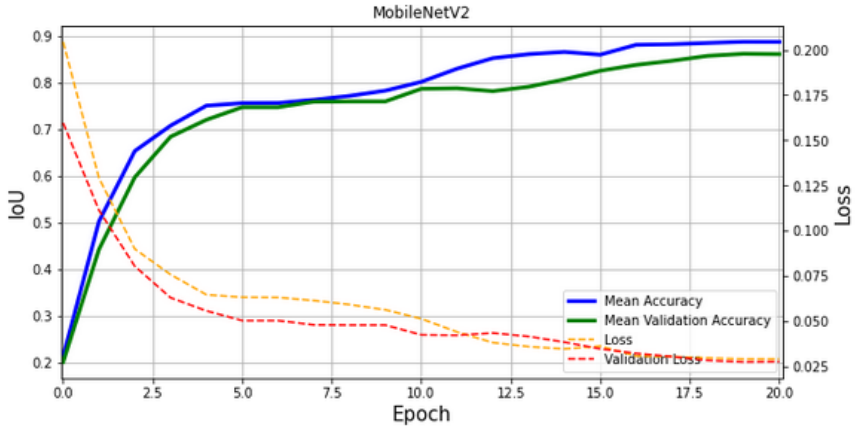

Рисунок 3.5 – Графік зміни втрат та точності локалізації

На основі даних, що подані на рисунках 3.4 – 3.5, можна зробити висновок, що точність навченої моделі сягає приблизно 90%.

Щоб довести ефективність обраної архітектури для моделі детектування було проведено порівняння з архітектурою іншого типу, а саме з модифікацією архітектури Yolo YoloV2 (Рисунки 3.6 – 3.7).

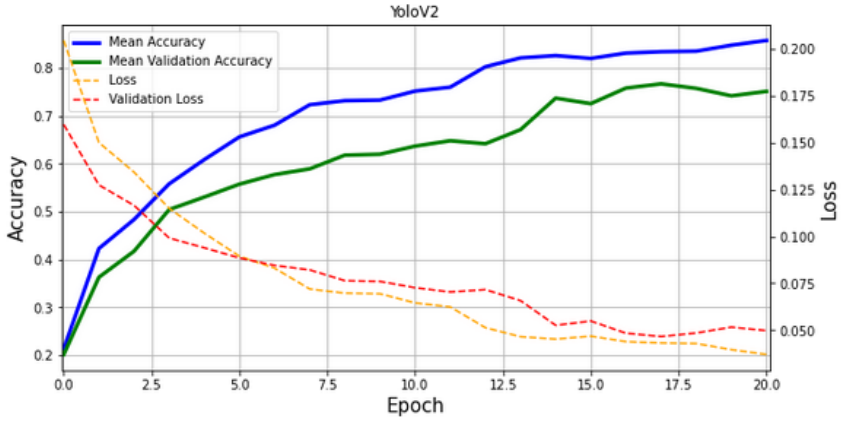

Рисунок 3.6 - Графік зміни втрат та точності класифікації для моделі YoloV2

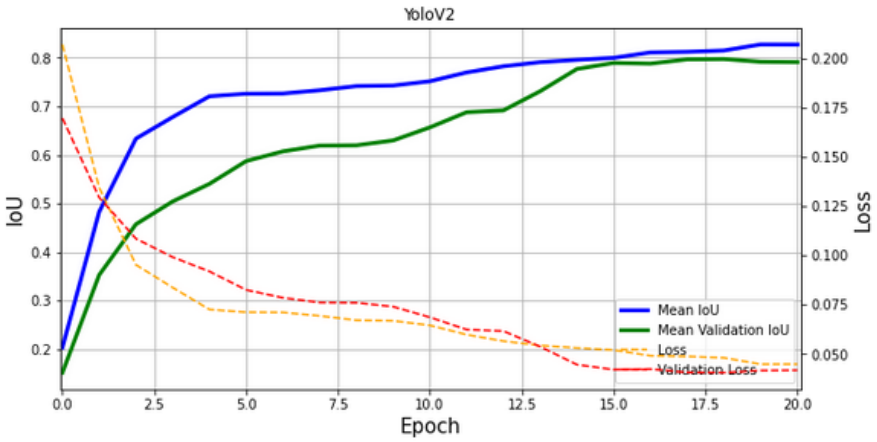

Рисунок 3.7 - Графік зміни втрат та точності локалізації для моделі YoloV2

Проаналізувавши дані, які подані в рисунка 3.6 – 3.7, можна зробити висновок що обрана архітектура моделі детектування SDD MobileNetV2 показує кращі показники в порівнянні з іншою популярною для задач детектування архітектурою YoloV2, точність якої за тієї ж самої вибірки зображень сягає приблизно 80%. Отже було обрано ефективну модель детектування.

Також для навченої моделі було сформовано матрицю невідповідностей, яка показує точність класифікації (Рисунок).

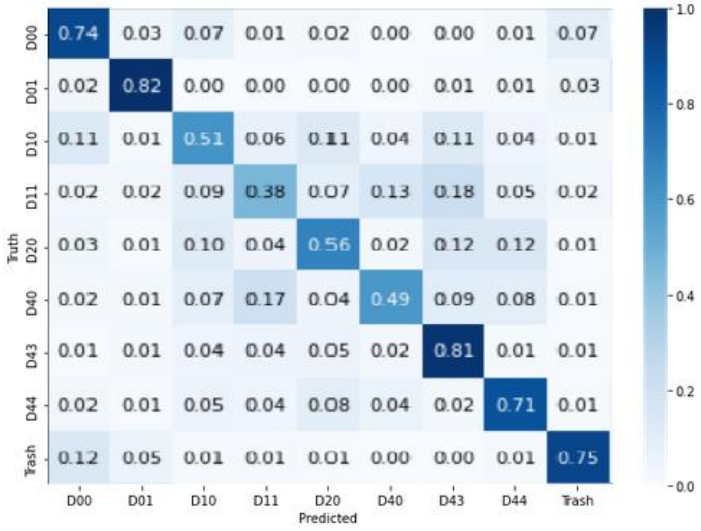

Рисунок 3.8 – Матриця невідповідностей обученої моделі

На рисунку 3.9 зображено приклади детектування дефектів дорожнього покриття на тестовому наборі.

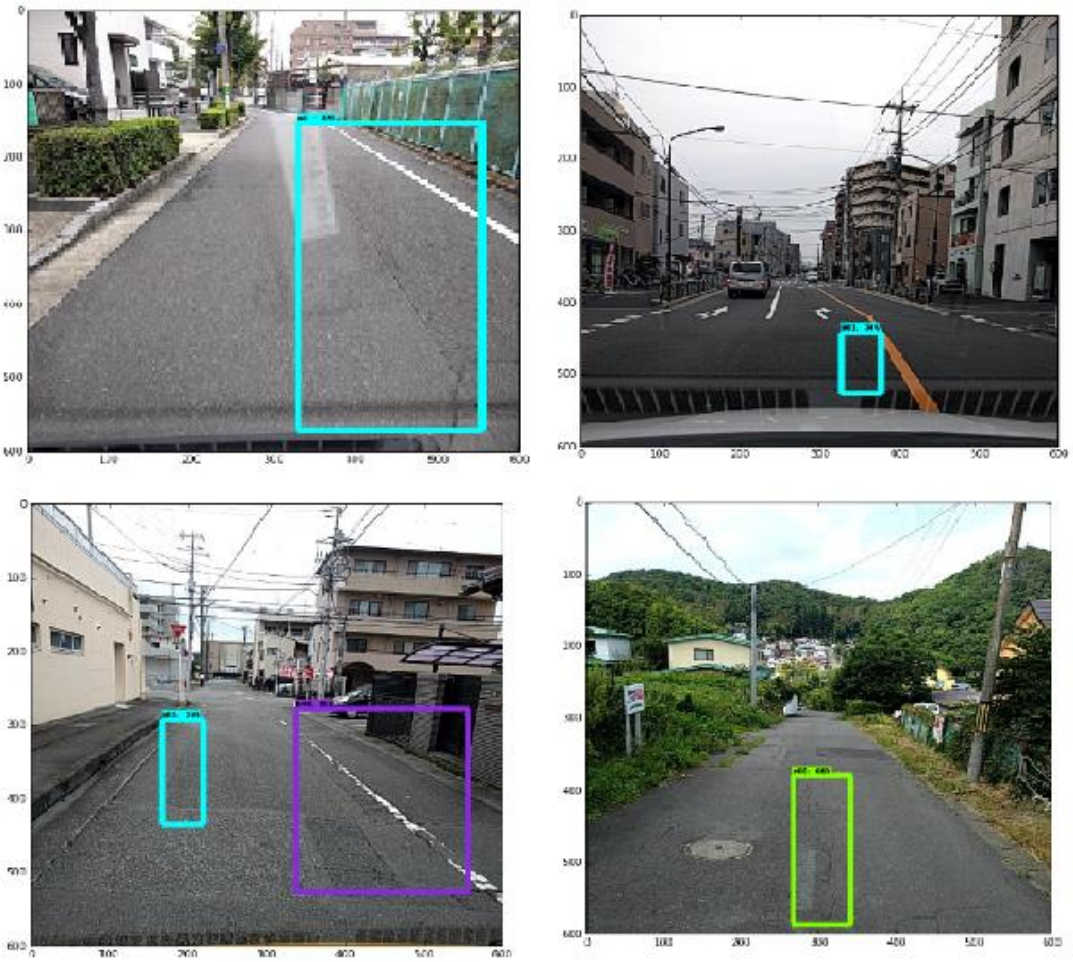

Рисунок 3.9 – Приклади детектування дорожнього покриття

Спираючи на вищеописані дані, можна зробити висновок, що побудована модель детектування з архітектурним підходом SSD

MobileNetV2 може бути використана як ефективна для задачі моніторингу функціонального стану доріг та узбіч.

3.3 Програмна реалізація

Для формування вихідних даних, а саме звіту, який містить в собі вхідне зображення з розміченою обмежуючою рамкою та міткою класифікованого дефекту, необхідно на вхід подати зображення, на якому необхідно детектувати дефект дорожнього покриття, або засмічення узбіч. На рисунку 3.10 зображено схему роботи технології.

Рисунок 3.10 - Схема роботи інтелектуальної технології

На першому етапі навчена модель обробляє вхідне зображення. Цей етап також включає в себе прогнозування моделі класифікації та локалізації шуканих об’єктів.

Далі оброблені дані (набір прогнозів) треба відфільтрувати. На наступному етапі відбувається фільтрація, при якій прогнози з близькими відстанями групуються та вважаються даними одного класу.

На останньому етапі роботи моделі формується звіт детектування. Звіт представлений у вигляді зображення, на якому обмежуючою рамкою виділено один або декілька дефектів дорожнього покриття чи засмічуваностей узбіч,

якщо вони наявні, та зображено мітку відповідного класу та відсоток входження до нього.

Для програмної реалізації технології використовувалась мова програмування Python 3.8.4 та сервіс Google Colab. Google Colaboratory – це безкоштовний хмарний сервіс на основі Jupyter Notebook. Цей сервіс надає все необхідне для машинного навчання прямо в браузері. Також він надає безкоштовний доступ до швидких GPU та TPU. Програмний код знаходиться в додатках А та Б. В таблиці 3.3 описано використані бібліотеки та фреймворки.

Таблиця 3.3 – Використані у роботі фреймворки та бібліотеки

Назва	Короткий опис
TensorFlow	Бібліотека з відкритим кодом для машинного навчання, особлива увага в якій приділяється саме глибоким нейронним мережам. Бібліотека може працювати на різних платформах, також повністю підтримується сервісом Google Colab
NumPy	Розширення функціоналу Python. Дозволяє працювати з різнорівневими масивами, матрицями та іншими структурами даних
OpenCV	Розширення функціоналу Python. Дозволяє працювати з методами машинного зору, а також оброблювати зображення, відео та деякі інші алгоритми
TensorFlow Object Detection API	Фреймворк призначений для спрощення роботи з моделями машинного навчання. Побудований на TensorFlow
TensorBoard	Інструментарій, який розширює функціонал роботи з машинним навчанням. Дозволяє

	представляти графічно показчики навчання моделі, виводити зображення, створювати гістограми, тощо.

Висновки

В ході виконання кваліфікаційної роботи магістра було проведено аналіз сучасних технологій моніторингу функціонального стану дорожнього покриття та чистоти узбіч. Аргументовано актуальність даної проблемної тематики та важливість імплементації інтелектуальних методів її вирішення. Літературний огляд існуючих рішень показав, що процеси автоматизації оцінки стану доріг та узбіч вже використовуються, але не дуже широко.

Було проведено огляд методів машинного навчання для обробки візуальних даних. Також був проведений огляд популярних архітектур та моделей для ефективного навчання та оброблення даних. Обрано оптимальну архітектуру згорткової нейронної мережі, яка поєднує в собі архітектурні підходи MobileNetV2 та SSD, для її використання у вирішенні проблеми моніторингу стану доріг та узбіч. Було відібрано вибірки зображень для навчання та тестування, на яких було проведено навчання та оцінку якості роботи технології відповідно.

Було проведено дослідження, суть якого полягала у порівнянні оцінок якості роботи обраної моделі з моделлю YoloV2. Дослідження показало вищу точність детектування у обраної моделі, отже її вибір можна назвати ефективним.

В майбутньому на основі цієї технології, при її подальшому вдосконаленні, можна створювати експертні систем для моніторингу стану інфраструктурних об’єктів. Оскільки архітектура MobilNet досить легка, а також показала себе ефективно при роботі на мобільних телефонах, такі системи можна буде використовувати саме на них, наприклад на телефонах з ОС Андроїд.

Список літератури

1. Васильев Ю. Э. Будущее диагностики за передвижными лабораториями: Ю. Э. Васильев, А. В. Беляков //Наука и техника в дорожной отрасли.- 2008. – No1 – С. 3–11.
2. Алексієв В. О. Управління розвитком транспортних систем :монографія / В. О. Алексієв. – Харків : ХНАДУ, 2008. – 268 с.
3. Yan, W.Y.; Yuan, X.X. A low-cost video-based pavement distress screening system for low-volume roads. J. Intell. Transp. Syst. 2018, 22, 376–389.
4. Maeda, H.; Sekimoto, Y.; Seto, T.; Kashiyama, T.; Omata, H. Road damage detection using deep neural networks with images captured through a smartphone. Comput. Aided Civ. Infrastruct. Eng. 2018, 33, 1127–1141.
5. Sattar, S.; Li, S.; Chapman, M. Road surface monitoring using smartphone sensors: A review. Sensors 2018, 18, 3845
6. Sansoni, G.; Trebeschi, M.; Docchio, F. State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation. Sensors 2009, 9, 568–601.
7. Boscaino, G.; Praticò, F.G.; Vaiana, R. Texture Indicators and Surface Performance Significance
8. Pattern Classification. (2nd ed.). / . Duda, R.O., Hart, P.E., and Stork D.G// New York: Wiley-Interscience Publication – 2001 – P. 12- 16.
9. A. Khan, A. Sohail, U. Zahoora, A. S. Qureshi, “A Survey, of the Recent Architectures of Deep Convolutional Neural Networks”, 2019, pp.
10. Gouk H.G.R., Blake A.M. Fast sliding window classification with convolutional neural networks // Proceedings of the 29th International Conference on Image and Vision Computing, New Zealand. – ACM, 2014. – pp. 114-118.
11. Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – pp. 779-788.

12. Redmon J., Farhadi A. YOLOv3: An incremental improvement Tech report, arXiv:1804.02767. – 2018. – 6 p.
13. Bishop C.M. Pattern Recognition and Machine Learning. – Springer-Verlag, New York, 2006. – 738 p.
14. Girshick R.B., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2014. – 21 p.
15. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu Ch.-Y., Berg A.C. SSD: Single Shot MultiBox Detector // European Conference on Computer Vision (ECCV),Springer, Cham. – 2016. – Vol. 9905. – pp. 21-37
16. Wan S., Chen Z., Zhang T., Zhang B., Wong K. Bootstrapping Face Detection with Hard Negative Examples arXiv:1608.02236. – 2016. – 7 p.
17. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, IEEE/CVF, Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT, USA: IEEE, 2018. pp.

Додаток А

```
model {
    sdd{
      num_classes: 9
      image_resizer {
        keep_aspect_ratio_resizer {
          min_dimension: 600
          max_dimension: 1024
        }
      }
      feature_extractor {
        type: 'mobile_net_v2'
        first_stage_features_stride: 8
      }
      first_stage_anchor_generator {
        grid_anchor_generator {
          scales: [0.25, 0.5, 1.0, 2.0]
          aspect_ratios: [0.5, 1.0, 2.0]
          height_stride: 8
          width_stride: 8
        }
      }
      first_stage_atrous_rate: 2
      first_stage_box_predictor_conv_hyperparams {
        op: CONV
        regularizer {
          l2_regularizer {
            weight: 0.0
          }
        }
        initializer {
          truncated_normal_initializer {
            stddev: 0.01
          }
        }
      }
      first_stage_nms_score_threshold: 0.0
      first_stage_nms_iou_threshold: 0.7
      first_stage_max_proposals: 300
      first_stage_localization_loss_weight: 2.0
      first_stage_objectness_loss_weight: 1.0
      initial_crop_size: 17
      maxpool_kernel_size: 1
      maxpool_stride: 1
```

```
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        use_dropout: false
        dropout_keep_probability: 1.0
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0
            }
          }
          initializer {
            variance_scaling_initializer {
              factor: 1.0
              uniform: true
              mode: FAN_AVG
            }
          }
        }
      }
    }
    second_stage_post_processing {
      batch_non_max_suppression {
        score_threshold: 0.0
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 100
      }
      score_converter: SOFTMAX
    }
    second_stage_localization_loss_weight: 2.0
    second_stage_classification_loss_weight: 1.0
  }
}
```

```
train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.0003
          schedule {
            step: 0
            learning_rate: .0003
          }
          schedule {
            step: 900000
            learning_rate: .00003
          }
          schedule {
            step: 1200000
            learning_rate: .000003
          }
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "voc/pretrained/model.ckpt"
  from_detection_checkpoint: true
  num_steps: 20000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}
```

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "voc/pascal_train.record
 }
  label_map_path: "voc/pascal_label_map.pbtxt"
}

eval_config: {
  num_examples: 5823

  max_evals: 10
}

eval_input_reader: {
 tf_record_input_reader {
    input_path: "voc/pascal_val.record"
 }
  label_map_path: "voc/pascal_label_map.pbtxt"
  shuffle: false
  num_readers: 1
}
```

Додаток Б

```
def __call__(self):
        logging.debug("Creating model...")

        inputs = Input(shape=self._input_shape)
        model_mobilenet = MobileNet(input_shape=self._input_shape, alpha=self.alpha,
        depth_multiplier=1, dropout=1e-3,
        include_top=False, weights=self.weights, input_tensor=None, pooling=None)

        x = model_mobilenet(inputs)

        feat_a = GlobalAveragePooling2D()(x)
        feat_a = Dropout(0.5)(feat_a)
        feat_a = Dense(self.FC_LAYER_SIZE, activation="relu")(feat_a)

        pred_g_softmax = Dense(2, activation='softmax', name='gender')(feat_a)
        pred_a_softmax = Dense(self.num_neu, activation='softmax', name='age')(feat_a)

        model = Model(inputs=inputs, outputs=[pred_g_softmax, pred_a_softmax])

        return model


def setUpClass(cls):
        print("******* Unit Test for Keras *******")
        cls.adapa_utility = AdapaUtility()
        cls.data_utility = DataUtility()
        model = applications.MobileNet(weights='imagenet',
        include_top=False,input_shape = (224, 224,3))
        activType='sigmoid'
        x = model.output
        x = Flatten()(x)
        x = Dense(1024, activation="relu")(x)
        predictions = Dense(2, activation=activType)(x)
        cls.model_final = Model(inputs =model.input, outputs = predictions,name='predictions')
def test_02_image_classifier__as_input(self):
        model = applications.MobileNet(weights='imagenet', include_top=False,input_shape = (80, 80,3))
        activType='sigmoid'
        x = model.output
        x = Flatten()(x)
        x = Dense(1024, activation="relu")(x)
        predictions = Dense(2, activation=activType)(x)
        model_final = Model(inputs =model.input, outputs = predictions,name='predictions')

        cnn_pmml = KerasToPmml(model_final,model_name="MobileNetBase64",description="Demo",\
            copyright="Internal User",dataSet='imageBase64',predictedClasses=['dogs','cats'])
        cnn_pmml.export(open('2classMBNetBase64.pmml', "w"), 0)

        img = image.load_img('nyoka/tests/resizedTiger.png')
        img = img_to_array(img)
        img = preprocess_input(img)
        imgtf = np.expand_dims(img, axis=0)

        base64string = "data:float32;base64," + FloatBase64.from_floatArray(img.flatten(),12)
        base64string = base64string.replace("\n", "")
        csvContent = "imageBase64\n\"" + base64string + "\""
        text_file = open("input.csv", "w")
        text_file.write(csvContent)
        text_file.close()

        model_pred=model_final.predict(imgtf)
        model_preds = {'dogs':model_pred[0][0],'cats':model_pred[0][1]}

        model_name  = self.adapa_utility.upload_to_zserver('2classMBNetBase64.pmml')

        predictions, probabilities = self.adapa_utility.score_in_zserver(model_name, 'input.csv','DN')

        self.assertEqual(abs(probabilities['cats'] - model_preds['cats']) < 0.00001, True)
        self.assertEqual(abs(probabilities['dogs'] - model_preds['dogs']) < 0.00001, True)
```

```
def test_validate_keras_mobilenet(self):
        input_tensor = Input(shape=(224, 224, 3))
        model = MobileNet(weights="imagenet", input_tensor=input_tensor)
        file_name = "keras"+model.name+".pmml"
        pmml_obj = KerasToPmml(model,dataSet="image",predictedClasses=[str(i) for i in range(1000)])
        pmml_obj.export(open(file_name,'w'),0)
        self.assertEqual(self.schema.is_valid(file_name), True)


def setUpClass(self):
        print("******* Unit Test for Keras *******")
        model = applications.MobileNet(weights='imagenet', include_top=False,input_shape = (224, 224,3))
        activType='sigmoid'
        x = model.output
        x = Flatten()(x)
        x = Dense(1024, activation="relu")(x)
        predictions = Dense(2, activation=activType)(x)
        self.model_final = Model(inputs =model.input, outputs = predictions,name='predictions')


def test_keras_01(self):

        cnn_pmml = KerasToPmml(self.model_final,model_name="MobileNet",description="Demo",\
            copyright="Internal User",dataSet='image',predictedClasses=['cats','dogs'])
        cnn_pmml.export(open('2classMBNet.pmml', "w"), 0)
        reconPmmlObj=ny.parse('2classMBNet.pmml',True)
        self.assertEqual(os.path.isfile("2classMBNet.pmml"),True)
        self.assertEqual(len(self.model_final.layers), len(reconPmmlObj.DeepNetwork[0].NetworkLayer))
```

```
def get_imagenet_architecture(architecture, variant, size, alpha, output_layer, include_top=False, weights='imagenet'):
    from keras import applications, Model

    if include_top:
        assert output_layer == 'last'

    if size == 'auto':
        size = get_image_size(architecture, variant, size)

    shape = (size, size, 3)

    if architecture == 'densenet':
        if variant == 'auto':
            variant = 'densenet-121'
        if variant == 'densenet-121':
            model = applications.DenseNet121(weights=weights, include_top=include_top, input_shape=shape)
        elif variant == 'densenet-169':
            model = applications.DenseNet169(weights=weights, include_top=include_top, input_shape=shape)
        elif variant == 'densenet-201':
            model = applications.DenseNet201(weights=weights, include_top=include_top, input_shape=shape)
    elif architecture == 'inception-resnet-v2':
        model = applications.InceptionResNetV2(weights=weights, include_top=include_top, input_shape=shape)
    elif architecture == 'mobilenet':
        model = applications.MobileNet(weights=weights, include_top=include_top, input_shape=shape, alpha=alpha)
    elif architecture == 'mobilenet-v2':
        model = applications.MobileNetV2(weights=weights, include_top=include_top, input_shape=shape, alpha=alpha)
    elif architecture == 'nasnet':
        if variant == 'auto':
            variant = 'large'
        if variant == 'large':
            model = applications.NASNetLarge(weights=weights, include_top=include_top, input_shape=shape)
        else:
            model = applications.NASNetMobile(weights=weights, include_top=include_top, input_shape=shape)
    elif architecture == 'resnet-50':
        model = applications.ResNet50(weights=weights, include_top=include_top, input_shape=shape)
    elif architecture == 'vgg-16':
        model = applications.VGG16(weights=weights, include_top=include_top, input_shape=shape)
    elif architecture == 'vgg-19':
        model = applications.VGG19(weights=weights, include_top=include_top, input_shape=shape)
    elif architecture == 'xception':
        model = applications.Xception(weights=weights, include_top=include_top, input_shape=shape)
    elif architecture == 'inception-v3':
        model = applications.InceptionV3(weights=weights, include_top=include_top, input_shape=shape)

    if output_layer != 'last':
        try:
            if isinstance(output_layer, int):
                layer = model.layers[output_layer]
            else:
                layer = model.get_layer(output_layer)
        except Exception:
            raise VergeMLError('layer not found: {}'.format(output_layer))
        model = Model(inputs=model.input, outputs=layer.output)

    return model
```