Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп’ютерних систем та мереж

Пояснювальна записка
до кваліфікаційної роботи бакалавра
за спеціальністю 123 «Комп’ютерна інженерія»

на тему: АЛГОРИТМИ МАШИННОГО НАВЧАННЯ ДЛЯ ОБРОБКИ ЗОБРАЖЕНЬ

Проектував _____ Д. С. Гордієнко

Керівник роботи _____ А. О. Сенько

Нормоконтроль _____ Д. І. Кузнєцов

Завідувач кафедри _____ А. І. Купін

Кривий Ріг
2024

Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

Ступінь вищої освіти бакалавр
Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії

_____ А. І. Купін

"____" _____ 20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гордієнко Данило Станіславович
(прізвище, ім'я, по батькові)

1. Тема роботи ___Алгоритми машинного навчання для обробки зображень_
_____
_____
керівник роботи ____Сенько А.О._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "___"_____20__року №__
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи _____
_інформація про алгоритми машинного навчання, розробка прорами, обробка даних._____
_____
_____
_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)____
___Моделювання дорожнього руху_____
___Розробка нейромережевої моделі руху автотранспорту_____
___Аналіз даних руху_____
_____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)___Презентація PowerPoint_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1	Пошук підходящої літератури	12.03.24 - 19.03.24	Виконав
2	Розробка структури дипломної роботи та сортування інформації за розділами	20.03.24	Виконав
3	Огляд інформації що стосується моделювання	23.03.24 – 27.03.24	Виконав
4	Написання першого розділу	01.04.24- 13.04.24	Виконав
5	Огляд існуючих рішень	21.04.24	Виконав
6	Написання другого розділу	26.04.24	Виконав
7	Розробка моделі	02.05.24	Виконав
8	Написання третього розділу	17.05.24	Виконав
9	Написання пояснювальної записки та загальне оформлення дипломної роботи	23.05.24	Виконав

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Цей дипломний проект досліджує розробку та застосування нейромережевих моделей для аналізу трафіку, зосереджуючись на короткостроковому та довгостроковому прогнозуванні трафіку. Робота охоплює розробку, навчання та валідацію згорткових нейронних мереж (CNN) та рекурентних нейронних мереж (RNN), включаючи мережі з довгою та короткою пам'яттю (LSTM), пристосовані для обробки просторових та часових даних про трафік відповідно. Документ складається з 120 сторінок, включаючи 30 рисунків, 15 таблиць і 5 додатків. Для підтримки дослідження було проаналізовано та процитовано 50 наукових статей та інформаційних джерел. Метою цього дослідження є вдосконалення систем управління дорожнім рухом шляхом інтеграції передових нейромережевих моделей, здатних точно прогнозувати трафік. Використані методи включають збір даних з різних джерел, таких як камери та GPS, методи попередньої обробки даних, такі як нормалізація та доповнення, а також ретельний відбір ознак для ефективного навчання моделей. Результати демонструють, що моделі значно підвищують точність прогнозів трафіку, які можуть бути інтегровані в існуючі системи управління дорожнім рухом без значних модифікацій. Новизна цієї роботи полягає в методологічному підході, що поєднує різні типи нейронних мереж для використання їхніх сильних сторін в обробці специфічних типів даних.

Ключові слова: АНАЛІЗ ТРАФІКУ, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ, МЕРЕЖІ З ДОВГОЮ КОРОТКОЧАСНОЮ ПАМ'ЯТТЮ, СИСТЕМИ УПРАВЛІННЯ ДОРОЖНІМ РУХОМ, ДОВГОСТРОКОВЕ ПРОГНОЗУВАННЯ ТРАФІКУ, УПРАВЛІННЯ ДОРОЖНІМ РУХОМ У РЕАЛЬНОМУ ЧАСІ, ПЕРЕВІРКА МОДЕЛЕЙ, ДАНІ ПРО ДОРОЖНІЙ РУХ.

ABSTRACT

This diploma project explores the development and application of neural network models for traffic analysis, focusing on both short-term and long-term traffic forecasting. The work encompasses the design, training, and validation of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), including Long Short-Term Memory networks (LSTMs), tailored for processing spatial and temporal traffic data respectively. The document is comprehensive, consisting of 120 pages, including 30 figures, 15 tables, and 5 appendices. A total of 50 scholarly articles and information sources have been reviewed and cited to support the research. The objective of this study is to enhance traffic management systems by integrating advanced neural network models capable of accurate traffic predictions. Methods employed include data collection from various sources like cameras and GPS, data preprocessing techniques such as normalization and augmentation, and meticulous feature selection to train the models effectively. The results demonstrate that the models significantly improve the accuracy of traffic forecasts, which can be integrated into existing traffic management systems without extensive modifications. The novelty of this work lies in its methodological approach, combining different types of neural networks to leverage their strengths.

Keywords: TRAFFIC ANALYSIS, NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORKS, RECURRENT NEURAL NETWORKS, LONG SHORT-TERM MEMORY NETWORKS, TRAFFIC MANAGEMENT SYSTEMS, DATA PREPROCESSING, FEATURE SELECTION, SHORT-TERM TRAFFIC FORECASTING, LONG-TERM TRAFFIC FORECASTING, REAL-TIME TRAFFIC MANAGEMENT, URBAN PLANNING, MODEL TRAINING, MODEL VALIDATION, TRAFFIC DATA.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

1. **CNN** - Convolutional Neural Network
2. **RNN** - Recurrent Neural Network
3. **LSTM** - Long Short-Term Memory
4. **GPS** - Global Positioning System
5. **MSE** - Mean Squared Error
6. **ReLU** - Rectified Linear Unit
7. **SGD** - Stochastic Gradient Descent
8. **t** - Time variable
9. **x** - Spatial variable or input feature vector
10. **y** - Output variable or target vector
11. **v** - Vehicle speed
12. **k** - Traffic density

ЗМІСТ

ВСТУП

Зростаюча складність міських транспортних систем і зростаюча потреба в ефективному управлінні транспортом підкреслюють актуальність цього дипломного проекту, метою якого є використання можливостей нейронних мереж для моделювання дорожнього руху.

Практичне значення цієї роботи полягає в тому, що вона може покращити моніторинг та прогнозування дорожнього руху в реальному часі, тим самим сприяючи більш плавному руху транспорту, зменшенню заторів та підвищенню безпеки дорожнього руху.

Основною метою цього проекту є розробка надійної нейромережевої моделі, яка може точно прогнозувати трафік на основі різноманітних вхідних даних, включаючи відео та дані GPS.

Це завдання передбачає збір та обробку даних про дорожній рух, вибір та навчання відповідної архітектури нейронної мережі, а також перевірку прогнозів моделі.

Об'єктом дослідження є транспортні системи міського середовища, де динаміка руху транспортних засобів є найбільш непередбачуваною та складною.

Предметом роботи є застосування методів глибокого навчання, зокрема згорткових нейронних мереж (CNN) та рекурентних нейронних мереж (RNN), для аналізу та прогнозування транспортних потоків. Для досягнення цих цілей проект використовує ряд методів дослідження, включаючи теоретичний аналіз існуючих моделей трафіку і конструкцій нейронних мереж, а також практичні експерименти, що включають попередню обробку даних, вибір ознак і навчання мережі.

Тестування результатів є критично важливим етапом, на якому прогнози моделі порівнюються з фактичними даними про трафік для оцінки точності та надійності. Цей процес не лише підтверджує ефективність нейромережевої моделі, але й дає уявлення про потенційні покращення для майбутніх досліджень.

Завдяки такому комплексному підходу проект має на меті зробити значний внесок у сферу транспортних досліджень і запропонувати практичні рішення для управління дорожнім рухом у міських умовах.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Теоретичні аспекти моделювання дорожнього руху

Моделювання дорожнього руху є ключовим компонентом в галузі транспортної інженерії, забезпечуючи наукову основу для аналізу, проектування та управління транспортними системами. Ця галузь використовує математичні та обчислювальні інструменти для моделювання та прогнозування моделей дорожнього руху, тим самим сприяючи прийняттю більш обґрунтованих рішень щодо управління дорожнім рухом та розвитку інфраструктури. Важливість моделювання дорожнього руху є багатогранною і охоплює оптимізацію транспортних потоків, підвищення безпеки дорожнього руху, зменшення заторів та мінімізацію впливу дорожнього руху на навколишнє середовище.

Теоретична основа моделювання дорожнього руху ґрунтується на кількох ключових принципах і теоріях, які описують поведінку дорожнього руху як складної динамічної системи. Однією з фундаментальних теорій є теорія транспортного потоку, яка розглядає дорожній рух як безперервний потік і може бути описана рівнянням нерозривності, фундаментальним принципом динаміки рідини:

$$\left[\frac{\partial k}{\partial t} + \frac{\partial (k \cdot v)}{\partial x} = 0\right], \tag{1.1}$$

де (k) - щільність руху (кількість транспортних засобів на одиницю довжини), (v) - швидкість руху, (t) - час, а (x) - простір вздовж дороги.

Іншим важливим аспектом теорії дорожнього руху є залежність швидкості від щільності, яку часто представляють моделлю Гріншильда, фундаментальною емпіричною лінійною залежністю в теорії транспортних потоків:

$$\left[v = v_f\left(1 - \frac{k}{k_j}\right)\right], \tag{1.2}$$

де $\left(v_f\right)$ - швидкість вільного потоку, а $\left(k_j\right)$ - щільність заторів, що представляє максимальну щільність транспортних засобів на дорозі.

Крім того, моделювання дорожнього руху широко використовує теорію черг для аналізу заторів, особливо на перехрестях або вузьких місцях. Базова модель черги в транспортних системах може бути описана рівнянням:

$$[\lambda < \mu], \tag{1.3}$$

де (λ) - швидкість прибуття транспортних засобів, а (μ)

- швидкість обслуговування, з якою транспортні засоби проходять через перехрестя або вузькі місця.

Моделі дорожнього руху можна класифікувати на мікроскопічні, мезоскопічні та макроскопічні моделі, кожна з яких забезпечує різний рівень деталізації та складності. Мікроскопічні моделі імітують поведінку окремих транспортних засобів, включаючи детальну взаємодію між транспортними засобами. Мезоскопічні моделі заповнюють прогалину між мікроскопічними і макроскопічними моделями, розглядаючи групи транспортних засобів як єдине ціле, таким чином спрощуючи деякі детальні взаємодії в мікроскопічних моделях. Макроскопічні моделі, з іншого боку, розглядають дорожній рух як безперервний потік, зосереджуючись на сукупній поведінці транспортних потоків без розрізнення динаміки окремих транспортних засобів.

Застосування цих моделей варіюється залежно від конкретних проблем дорожнього руху, що розглядаються, і масштабу транспортної системи, що розглядається. Наприклад, мікроскопічні моделі особливо корисні при проектуванні та аналізі інтелектуальних транспортних систем (ITC), де дані про окремі транспортні засоби мають вирішальне значення. Макроскопічні моделі часто використовуються для великомасштабного моделювання дорожнього руху.

Отже, моделювання дорожнього руху є важливою науковою дисципліною, яка підтримує раціональний розвиток і управління транспортними системами. Воно спирається на надійну теоретичну базу, яка описує складні взаємодії в транспортних системах. Застосовуючи ці теорії та моделі, можна значно покращити ефективність, безпеку та стійкість транспортних потоків, що в кінцевому підсумку підвищить якість міського життя та сприятиме збереженню навколишнього середовища. Системний підхід до моделювання дорожнього руху гарантує, що кожен компонент транспортної системи буде проаналізовано та оптимізовано, що призведе до комплексного вирішення проблем управління дорожнім рухом.

1.2 Огляд існуючих методів і моделей аналізу дорожнього руху

Ландшафт аналізу трафіку багатий на різноманітні методології, які відповідають різним рівням деталізації та складності. Ці методології поділяються на мікроскопічні, мезоскопічні та макроскопічні моделі, кожна з яких слугує окремим цілям і пропонує унікальне розуміння динаміки трафіку. Цей розділ надає всебічний огляд цих моделей, аналізуючи їх методологію, ефективність та обмеження, таким чином представляючи структуроване дослідження існуючих методів моделювання дорожнього руху. Мікроскопічні моделі фокусуються на русі окремих транспортних засобів. Вони імітують поведінку кожного транспортного засобу на основі правил прискорення, уповільнення та зміни смуги руху, часто використовуючи моделі, що слідують за автомобілем, такі як модель Гіпса або модель інтелектуального водія (IDM). IDM, наприклад, описується наступним чином:

$$\left[\dot{v} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right]\right], \quad (1.4)$$

де (v) - поточна швидкість автомобіля, (v_0) - бажана швидкість, (s) - відстань від бампера до бампера автомобіля попереду, (Δv) - різниця швидкостей, (a) - максимальне прискорення, (δ) - експонента прискорення, і (s^*)- бажаний мінімальний інтервал.

Мікроскопічні моделі є дуже деталізованими, що робить їх ідеальними для аналізу конкретних сценаріїв, таких як рух на перехрестях, навколо аварій або у вузьких місцях шосе. Однак, їх обчислювальна інтенсивність обмежує їх практичність для великих мереж, оскільки кожен додатковий транспортний засіб значно збільшує складність і обчислювальні вимоги. Мезоскопічні моделі слугують мостом між мікроскопічними та макроскопічними моделями, об'єднуючи окремі транспортні засоби у взводи або пакети, які мають схожі характеристики. Ці моделі часто використовують методи як мікроскопічних, так і макроскопічних теорій, використовуючи такі елементи, як залежність швидкості від щільності, а також враховуючи взаємодію між групами транспортних засобів. Загальний підхід у мезоскопічному моделюванні передбачає використання кінетичних моделей з рівняннями типу:

$$\left[\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} = \frac{1}{\tau}(f^{\text{eq}} - f)\right], \quad (1.5)$$

де (f) представляє функцію розподілу транспортних засобів, (f^{\text{eq}}) - рівноважний розподіл, а (\tau) - параметр часу релаксації.

Мезоскопічні моделі є менш вимогливими до обчислень, ніж мікроскопічні, і забезпечують розумний компроміс щодо деталізації, що робить їх придатними для мереж середнього розміру. Однак вони можуть надмірно спрощувати взаємодію транспортних засобів і не враховувати весь

спектр поведінки водіїв. Макроскопічні моделі розглядають транспортний потік подібно до рідини, зосереджуючись на сукупній поведінці транспортного потоку, а не окремих транспортних засобів. Ці моделі використовують рівняння, що випливають з принципів збереження маси та імпульсу, з фундаментальним рівнянням транспортного потоку, яке має вигляд

$$\left[\frac{\partial k}{\partial t} + \frac{\partial (k \cdot v)}{\partial x} = 0\right], \tag{1.6}$$

де (k) - щільність трафіку, а (v) - швидкість потоку. Макроскопічні моделі особливо ефективні для аналізу великомасштабної динаміки трафіку в розгалужених мережах, надаючи уявлення про загальні тенденції та закономірності трафіку. Незважаючи на широке застосування, макроскопічним моделям не вистачає деталізації для прогнозування конкретної поведінки водіїв або точного моделювання складних взаємодій у сценаріях з високим рівнем завантаженості доріг. Їх використання найкраще підходить для огляду транспортних потоків і попередніх етапів планування, а не для детальних стратегій управління дорожнім рухом.

Вибір методу моделювання дорожнього руху суттєво залежить від масштабу застосування та конкретних вимог до аналізу. Мікроскопічні моделі не мають собі рівних за деталізацією, але вимагають великих обчислювальних витрат, мезоскопічні моделі пропонують збалансований підхід для середньомасштабних застосувань, а макроскопічні моделі дають цінне уявлення про загальну поведінку транспортних систем, але за рахунок зниження точності деталей. Ефективність і обмеження кожної моделі повинні бути ретельно розглянуті, щоб вибрати найбільш підходящий метод для конкретного завдання аналізу трафіку. Такий структурований підхід забезпечує комплексне розуміння того, як різні моделі можуть бути

оптимально застосовані до різних аспектів управління та планування дорожнього руху.

1.3 Використання нейронних мереж у транспортних дослідженнях

Інтеграція нейронних мереж в транспортні дослідження являє собою значний прогрес в цій галузі, пропонуючи надійні рішення для складних проблем, таких як прогнозування, управління і контроль трафіку. Цей розділ заглиблюється в застосування нейронних мереж у транспорті, розглядаючи їх впровадження в різних проектах і дослідженнях, а також аналізуючи результати і методології, що застосовуються. Нейронні мережі, особливо моделі глибокого навчання, широко застосовуються завдяки їхній здатності моделювати складні нелінійні зв'язки та вмінню працювати з великими масивами даних, які часто зустрічаються в транспортних системах. Застосування цих мереж охоплює кілька сфер, включаючи прогнозування транспортних потоків, класифікацію транспортних засобів та автономну навігацію транспортних засобів.

Прогнозування транспортних потоків: Нейронні мережі вміють прогнозувати умови дорожнього руху, вивчаючи закономірності на основі історичних даних про трафік. Згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN), включаючи мережі з довгою короткочасною пам'яттю (LSTM), зазвичай використовуються для просторового та часового аналізу даних відповідно. Наприклад, модель LSTM може бути сформульована для прогнозування транспортного потоку наступним чином:

$$[h_t = \text{LSTM}(x_t, h_{t-1})], \tag{1.7}$$

де (x_t) - вхідна характеристика в момент часу (t), а(h_t) - прихований стан LSTM, що представляє прогноз потоку трафіку. Ця модель може ефективно фіксувати часові залежності в даних про дорожній рух, забезпечуючи точні прогнози, які мають вирішальне значення для управління та планування дорожнього руху. Класифікація та виявлення транспортних засобів нейронної

мережі, зокрема ШНМ, довели свою ефективність у завданнях виявлення та класифікації транспортних засобів, які є важливими для автоматизованого збору плати за проїзд та систем спостереження за дорожнім рухом. Типова архітектура ШНМ для класифікації транспортних засобів може включати кілька згорткових і об'єднуючих шарів, за якими слідують повністю з'єднані шари, що класифікують тип транспортного засобу на основі ознак, витягнутих із вхідних зображень. Глибоке навчання з підкріпленням, поєднання глибокого навчання і навчання з підкріпленням, стало ключовим у розробці систем автономного водіння. Ці системи навчаються оптимальним стратегіям навігації в складних умовах. Загальний підхід передбачає навчання нейронної мережі для максимізації функції винагороди, заснованої на діях транспортного засобу, сформульованої наступним чином:

$$\left[Q(s,a) = r + \gamma \max_{a'} Q\,(s', a')\right], \tag{1.7}$$

де $\left(Q(s,a)\right)$ - якість пари стан-дія, (s) - поточний стан, $(a) -$ виконана дія, (r) - негайно отримана винагорода, (s') - новий стан, і (γ) - коефіцієнт дисконтування.

Цей проект використовував нейронні мережі для покращення систем управління дорожнім рухом на автомагістралях. Завдяки впровадженню алгоритмів на основі нейронних мереж, програма змогла прогнозувати затори на дорогах і пропонувати водіям оптимальні стратегії маршрутизації, що значно скоротило час у дорозі і затори. Сінгапур інтегрував технології нейронних мереж у свої системи управління дорожнім рухом, використовуючи їх для аналізу відео з камер спостереження та прогнозування умов руху в режимі реального часу. Ці додатки підкреслюють універсальність та ефективність нейронних мереж у вирішенні різноманітних завдань у транспортних дослідженнях. Здатність нейронних мереж навчатися на основі даних і робити обґрунтовані прогнози є особливо цінною в середовищах, де

традиційні моделі можуть не спрацювати через складність або динамічну природу системи.

Таким чином, нейронні мережі не тільки розширили можливості транспортних систем адаптуватися і реагувати на мінливі умови, а й проклали шлях для інновацій, таких як автономне водіння. Їх подальша інтеграція в транспортні дослідження обіцяє подальший прогрес в управлінні дорожнім рухом і контролі транспортних засобів, що сприятиме створенню безпечніших і ефективніших транспортних систем. Систематичне застосування нейронних мереж в транспортних дослідженнях є прикладом значного зсуву в бік адаптивних рішень на основі даних в цій галузі.

1.4 Опис основних принципів глибокого навчання

Глибоке навчання, підмножина машинного навчання, зробило революцію в багатьох сферах, включаючи розпізнавання зображень, обробку природної мови і, особливо, транспорт. Цей розділ висвітлює основні принципи глибинного навчання, надаючи структурований і всебічний аналіз його компонентів, методологій і застосувань, зокрема, у прогнозуванні та управлінні дорожнім рухом. Моделі глибокого навчання - це, по суті, нейронні мережі з трьома або більше шарами. Ці мережі призначені для імітації того, як людський мозок обробляє дані. Вони складаються з вхідного шару, декількох прихованих шарів і вихідного шару. Кожен шар складається з вузлів, або нейронів, які з'єднані ребрами, що представляють потік інформації. Сила цих зв'язків визначається вагами, які підлаштовуються під час навчання, щоб мінімізувати похибку в прогнозах моделі.

Математичне представлення роботи нейрона можна описати наступним чином:

$$[y = f(\sum_{i=1}^{n} w_i x_i + b)], \tag{1.8}$$

де (x_i) - вхідні дані, (w_i) - ваги, (b) - зсув, а (f) - нелінійна функція активації, наприклад, ReLU або сигмоїда.

В основі глибокого навчання лежить процес навчання, який включає в себе пряме і зворотне поширення. Під час прямого поширення дані рухаються через мережу від вхідного до вихідного шару, і отримується прогноз моделі. Якщо прогноз відхиляється від фактичного значення, помилка обчислюється за допомогою функції втрат, зазвичай середньоквадратичної помилки (MSE) для задач регресії:

$$\left[\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2\right], \tag{1.9}$$

де (Y_i) - істинне значення, а $\left(\widehat{Y}_i\right)$ - передбачене значення. Поширення - це процес, за допомогою якого модель вчиться на помилках. Помилка поширюється назад через мережу, а ваги коригуються для мінімізації втрат за допомогою алгоритмів оптимізації, таких як стохастичний градієнтний спуск (SGD):

$$[w_{new} = w_{old} - \eta \cdot \nabla L], \tag{1.10}$$

де (η) - швидкість навчання, а (∇L) - градієнт функції втрат відносно ваг. Глибоке **навчання в прогнозуванні та управлінні доро**жнім рухом: У транспортній галузі моделі глибокого навчання застосовуються для покращення прогнозування транспортних потоків, класифікації транспортних засобів та управління заторами. Згорткові нейронні мережі (CNN) використовуються для обробки просторових даних, таких як зображення і відео з дорожніх камер, в той час як рекурентні нейронні мережі (RNN), особливо мережі LSTM (Long Short-Term Memory), підходять для тимчасових даних, таких як швидкість і інтенсивність потоку в часі. CNN застосовуються для обробки та аналізу відеоданих з камер спостереження на автомагістралях

для визначення щільності руху і класифікації транспортних засобів. Така обробка даних у реальному часі допомагає в динамічному управлінні

світлофорами та управлінні заторами. LSTM ефективно використовуються для прогнозування транспортних потоків на основі історичних даних. Наприклад, в одному дослідженні було застосовано мережу LSTM для прогнозування трафіку на міських дорогах, що допомогло в проактивному управлінні дорожнім рухом і плануванні маршрутів.

Висновок

Цей аналіз демонструє здатність глибинного навчання обробляти складні, багатовимірні дані і надавати точну інформацію в реальному часі, що має вирішальне значення для ефективних систем управління дорожнім рухом. Глибоке навчання пропонує потужний набір інструментів для вдосконалення транспортних систем, завдяки своїй здатності вчитися на великих обсягах даних і гнучкості в роботі з різними типами даних. Принципи архітектури нейронних мереж, процеси навчання і конкретні застосування в управлінні дорожнім рухом в сукупності підкреслюють трансформаційний вплив глибокого навчання в цій галузі. Оскільки транспортні системи продовжують розвиватися, інтеграція глибинного навчання відіграватиме ключову роль у розробці розумніших та ефективніших транспортних рішень. Це систематичне дослідження глибокого навчання не тільки висвітлює його поточні застосування, а й створює основу для майбутніх інновацій в транспортних технологіях.

РОЗДІЛ 2. АНАЛІЗ ДАНИХ РУХУ

2.1 Збір та обробка даних

Збір та обробка даних є фундаментальними аспектами аналізу та моделювання дорожнього руху, забезпечуючи сировину, з якої виводяться висновки та прогнози. Цей розділ описує методології, що використовуються для збору та уточнення даних про трафік, наголошуючи на системному підході, необхідному для забезпечення цілісності даних та їхньої придатності для використання. Для збору даних про дорожній рух використовуються різні технології, кожна з яких підходить для конкретних аспектів моніторингу та аналізу дорожнього руху. Індуктивні петльові детектори та п'єзоелектричні датчики вбудовуються в дорожнє покриття для виявлення присутності та швидкості транспортних засобів. Ці датчики надають дані про транспортний потік і щільність руху в реальному часі, які мають вирішальне значення для управління світлофорами та управління заторами.

```
[ ]  # Random Seed
     SEED = 42
     np.random.seed(SEED)
```

```
     # Load metadata
     metadata = pd.read_csv(TRAIN_CSV_PATH)

     # Quick look
     metadata.head()
```

	image	xmin	ymin	xmax	ymax
0	vid_4_1000.jpg	281.259045	187.035071	327.727931	223.225547
1	vid_4_10000.jpg	15.163531	187.035071	120.329957	236.430180
2	vid_4_10040.jpg	239.192475	176.764801	361.968162	236.430180
3	vid_4_10020.jpg	496.483358	172.363256	630.020260	231.539575
4	vid_4_10060.jpg	16.630970	186.546010	132.558611	238.386422

Рис 2.1. Завантаження та обробка метаданих

Камери відеоспостереження, встановлені в стратегічних точках на дорогах і перехрестях, записують відеоматеріали, які можна аналізувати за допомогою методів обробки зображень для класифікації типів транспортних засобів, підрахунку інтенсивності руху та виявлення порушень правил дорожнього руху.

Транспортні засоби, обладнані GPS-пристроями, передають своє місцезнаходження та швидкість до центрів управління дорожнім рухом. Ці дані особливо корисні для відстеження траєкторій руху транспортних засобів, аналізу часу в дорозі та оптимізації маршрутів. Ці технології використовуються для визначення швидкості та відстані транспортних засобів, пропонуючи високу точність і здатність функціонувати за різних погодних умов. Після того, як дані зібрані, їх необхідно обробити, щоб забезпечити їхню чистоту та структурованість, що зробить їх придатними для аналізу.

```python
def load_image(file_name: str, ROOT_PATH: str) -> tf.Tensor:
    '''
    Loads an image from the file path provided, performs necessary preprocessing steps, and returns the image as a tensor.

    Args:
        file_name (str): Name of the image file to load.
        ROOT_PATH (str): Root directory where the image file is stored.

    Returns:
        image (tf.Tensor): Tensor representing the loaded image, normalized between 0 and 1, with data type float32.
    '''

    # Obtain the complete path to the image file
    image_path = os.path.join(ROOT_PATH, file_name)

    # Read the image file
    image = tf.io.read_file(image_path)

    # Decode the image to tensor with 3 channels
    image = tf.image.decode_jpeg(image, channels=3)

    # Convert the image data type to float32
    image = tf.image.convert_image_dtype(image, dtype=tf.float32)

    # Normalize the image to [0, 1] range
    image = tf.clip_by_value(image, clip_value_min=0.0, clip_value_max=1.0)

    return image
```

Рис 2.2. Етапи попередньої

Цей крок передбачає видалення або виправлення помилкових або неповних записів даних. Для даних з датчиків руху це може включати фільтрацію нереальних показників швидкості або потоку, які можуть свідчити

про несправність датчика. Дані з різних джерел (наприклад, датчиків, камер, GPS) мають бути інтегровані в цілісний набір даних. Це передбачає вирівнювання даних у часі та просторі, гарантуючи, що всі точки даних синхронізовані відповідно до часових міток і геолокації. Такі методи, як згладжування та фільтрація, застосовуються для зменшення шуму в даних, особливо в даних з датчиків або GPS-пристроїв, на які можуть впливати фактори навколишнього середовища або неточності пристрою. З необроблених даних виділяються відповідні ознаки, які будуть використані в подальшому аналізі. Наприклад, з відеозаписів камер можна виокремити кількість транспортних засобів, їхні типи та швидкість у різний час доби.

```python
def load_dataset(
    metadata: pd.DataFrame=metadata,
    ROOT_PATH: str=TRAIN_ROOT_DIR,
    SHUFFLE: bool=True,
    SPLIT_RATIO: float=0.1) -> tuple:

    """
    Loads and returns a dataset of images and their corresponding bounding boxes based on the provided metadata file.

    Args:
        metadata (pd.DataFrame): A pandas DataFrame containing the metadata information for the dataset.
        ROOT_PATH (str): The root path of the dataset.
        SHUFFLE (bool, optional): Whether or not to shuffle the dataset. Defaults to True.
        SPLIT_RATIO (float, optional): The ratio for splitting the dataset into training and validation sets.
            If set to None, no split is performed. Defaults to 0.1.

    Raises:
        ValueError: If SPLIT_RATIO is not between 0 and 1.
        FileNotFoundError: If the ROOT_PATH does not exist.

    Returns:
        tuple: A tuple containing the dataset images and their corresponding bounding boxes as numpy arrays.
    """

    # Check if SPLIT_RATIO is between 0 and 1.
    if SPLIT_RATIO is not None and (SPLIT_RATIO < 0 or SPLIT_RATIO > 1):
        raise ValueError("SPLIT_RATIO must be between 0 and 1.")

    # Check if the ROOT_PATH is valid and exists.
    if not os.path.exists(ROOT_PATH):
        raise FileNotFoundError(f"The ROOT_PATH {ROOT_PATH} does not exist.")

    # Collect all the image paths.
    image_paths = metadata['image']

    # Create space for storing the images and the bounding boxes
    images = np.empty(shape=(len(image_paths), *IMAGE_SIZE), dtype=np.float32)
    bounding_boxes = np.empty(shape=(len(image_paths), 4), dtype=np.float32)
```

Рис 2.3. Нормалізація та завантаження зображень

Дані нормалізуються або масштабуються для того, щоб діапазон значень даних не впливав на аналіз. Наприклад, швидкість, зафіксована різними датчиками, може бути нормалізована до загальної шкали перед використанням у моделях транспортних потоків. Етапи попередньої обробки мають вирішальне значення для підготовки даних для нейромережевої моделі, що

використовується в програмному проекті. Наприклад, дані, зібрані з камер (відеозаписи) і GPS (дані про місцезнаходження), повинні бути попередньо оброблені, щоб виділити значущі характеристики, такі як щільність транспортних засобів і середня швидкість. Ці характеристики потім використовуються як вхідні дані для нейронної мережі, яка прогнозує трафік на основі історичних даних. Фрагмент коду, представлений у проекті, ілюструє, як дані з різних джерел обробляються і готуються для навчання моделі, підкреслюючи важливість ретельної попередньої обробки даних для досягнення точних і надійних прогнозів моделі.

Отже, збір і попередня обробка даних про дорожній рух є критично важливими процесами, які лежать в основі ефективного управління дорожнім рухом і його аналізу. Використовуючи різноманітні методи збору даних і ретельну попередню обробку, дані про дорожній рух можна перетворити на цінний ресурс для прогнозування та управління транспортними потоками, що в кінцевому підсумку сприятиме підвищенню ефективності та безпеки транспортних систем. Такий системний підхід до обробки даних гарантує, що розроблені моделі дорожнього руху є надійними і відображають реальні умови, забезпечуючи міцну основу для аналізу дорожнього руху і прийняття рішень.

2.2 Використання різних типів даних (відео, дані GPS)

У сфері моделювання дорожнього руху використання різних типів даних, таких як відео та GPS-дані, відіграє ключову роль у підвищенні точності та ефективності прогнозних моделей. У цьому розділі розглядаються методології інтеграції цих різнорідних джерел даних та їх конкретні застосування в аналізі дорожнього руху, забезпечуючи всебічне розуміння їх синергетичного потенціалу. Відеодані, в основному отримані з камер відеоспостереження, встановлених у стратегічних місцях, таких як перехрестя і головні автомагістралі, надають багату візуальну інформацію про умови дорожнього руху. Обробка відеоданих складається з кількох ключових етапів: Передові

технології комп'ютерного зору, такі як алгоритми виявлення об'єктів (наприклад, YOLO, SSD), застосовуються для виявлення та відстеження транспортних засобів у відеокадрах. Це дозволяє отримувати кількісні дані, такі як кількість, типи та траєкторії руху транспортних засобів.

```python
def show_images_and_bbs(data: tfd.Dataset, GRID: list=[6,3], FIGSIZE: tuple=(30,40)) -> None:
    """
    Visualizes a batch of images with their bounding boxes.

    Args:
    - data: A TensorFlow dataset.
    - GRID: A list specifying the number of rows and columns in the plot grid. Default is [6,3].
    - FIGSIZE: A tuple specifying the size of the plot. Default is (30,40).

    Returns: None
    """

    # Define bounding box color
    BB_COLOR = (0, 255, 0)

    # Plotting Configuration
    plt.figure(figsize=FIGSIZE)
    n_rows, n_cols = GRID
    n_images = n_rows * n_cols

    # Gather the data
    images, boxes = data[0], data[1]

    # Iterate over the data
    for index, (image, box) in enumerate(zip(images, boxes)):

        # Convert the Bounding Box
        x1, y1, x2, y2 = map(int, box)

        # Add the rectangle
        image = cv.rectangle(
            img = image,
            pt1 = (x1, y1),
            pt2 = (x2, y2),
            thickness=5,
            color=BB_COLOR
        )
```

Рис 2.4. Метод відображення тестових зображень

Аналізуючи рух транспортних засобів у часі, можна виявити закономірності транспортних потоків. Це включає оцінку швидкості, визначення пікових періодів руху та виявлення потенційних вузьких місць. Відеодані можна аналізувати в режимі реального часу для виявлення аварій або заторів на дорогах, що сприяє швидкому реагуванню центрів управління дорожнім рухом. GPS-дані, зібрані з транспортних засобів, обладнаних GPS-пристроями, надають геопросторову та часову інформацію, яка має вирішальне значення для динамічного моделювання дорожнього руху. Дані GPS дозволяють аналізувати маршрути транспортних засобів, що дає змогу оптимізувати розподіл трафіку в мережі, пропонуючи альтернативні

маршрути під час заторів. Агрегуючи дані GPS з декількох транспортних засобів, можна отримати точні оцінки часу в дорозі та затримок, що підвищує прогностичні можливості моделей дорожнього руху.

Рис 2.5. Виявлення транспортів на зображені

GPS-дані допомагають виявити затори, надаючи в режимі реального часу інформацію про швидкість і щільність транспортних засобів на різних ділянках дорожньої мережі. Інтеграція відео та GPS-даних передбачає системний підхід до злиття даних, що підвищує надійність і повноту моделей дорожнього руху (див.рис.2.5). Забезпечення синхронізації даних з різних джерел у часі та просторі має вирішальне значення.

Це передбачає вирівнювання часових міток і геопросторових координат між наборами даних, щоб забезпечити єдине уявлення про умови дорожнього руху. Характеристики, отримані з відео та GPS даних, такі як щільність транспортних засобів з відео та середня швидкість з GPS, об'єднуються для створення комплексних вхідних даних для моделей прогнозування дорожнього руху. Інтегровані дані використовуються для навчання складних моделей машинного навчання, таких як нейронні мережі, які можуть вивчати складні закономірності на основі об'єднаних даних. Ці моделі здатні робити точні прогнози щодо умов дорожнього руху і можуть адаптуватися до змін у динаміці трафіку. У наданому програмному проекті дані з джерел відео та GPS

обробляються та інтегруються для навчання нейромережевої моделі. Модель використовує функції, витягнуті з обох типів даних, використовуючи сильні сторони кожного з них для підвищення точності прогнозування. Наприклад, відеодані надають детальну візуальну інформацію про місцеві умови дорожнього руху, тоді як GPS-дані забезпечують ширше геопросторове покриття. Поєднання цих типів даних дозволяє моделі враховувати як мікро-, так і макрорівневі схеми руху, що призводить до більш ефективних стратегій управління дорожнім рухом(див.рис.2.6).

Отже, стратегічне використання та інтеграція відео- та GPS-даних у моделюванні дорожнього руху є значним кроком вперед у розвитку систем управління дорожнім рухом. Поєднуючи детальну, локалізовану інформацію з відеоданих з широкою геопросторовою інформацією з даних GPS, моделі дорожнього руху можуть досягти більшої точності та оперативності. Такий комплексний підхід не лише покращує прогнозування дорожніх систем, але й сприяє прийняттю більш обґрунтованих рішень в управлінні та плануванні дорожнього руху. Систематична інтеграція цих різнорідних джерел даних має важливе значення для розробки адаптивних, ефективних і надійних транспортних рішень.

2.3 Методи попередньої обробки та відбору ознак

У галузі моделювання та аналізу трафіку попередня обробка даних і вибір релевантних ознак є критично важливими кроками, які суттєво впливають на продуктивність нейромережевих моделей. У цьому розділі детально розглядаються методи, що використовуються для нормалізації, доповнення та вилучення ознак, а також критерії відбору найбільш інформативних ознак для навчання моделі. Нормалізація даних - це фундаментальна техніка попередньої обробки, яка використовується для стандартизації діапазону незалежних змінних або ознак даних. В аналізі даних про трафік нормалізація допомагає скоригувати масштаб даних з різних датчиків або джерел, гарантуючи, що кожна ознака робить однаковий внесок в аналіз, і покращує

швидкість збіжності під час навчання нейронних мереж. Поширені методи включають Цей метод змінює масштаб ознаки до фіксованого діапазону, зазвичай від 0 до 1, використовуючи формулу:

$$\left[X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}\right], \tag{2.1}$$

$(X_{\text{min}})i(X_{\text{max}})$ - мінімальне та максимальне значення ознаки (X) відповідно.

Цей метод перетворює ознаки так, щоб їхнє середнє значення дорівнювало нулю, а стандартне відхилення - одиниці, що обчислюється так:

$$\left[X_{\text{std}} = \frac{X - \mu}{\sigma}\right], \tag{2.2}$$

де (μ)і(σ) - це середнє і стандартне відхилення ознаки (X).

В аналізі даних про трафік методи доповнення використовуються для штучного розширення розміру набору даних шляхом створення модифікованих версій точок даних, що допомагає запобігти надмірному пристосуванню і підвищити надійність моделі.

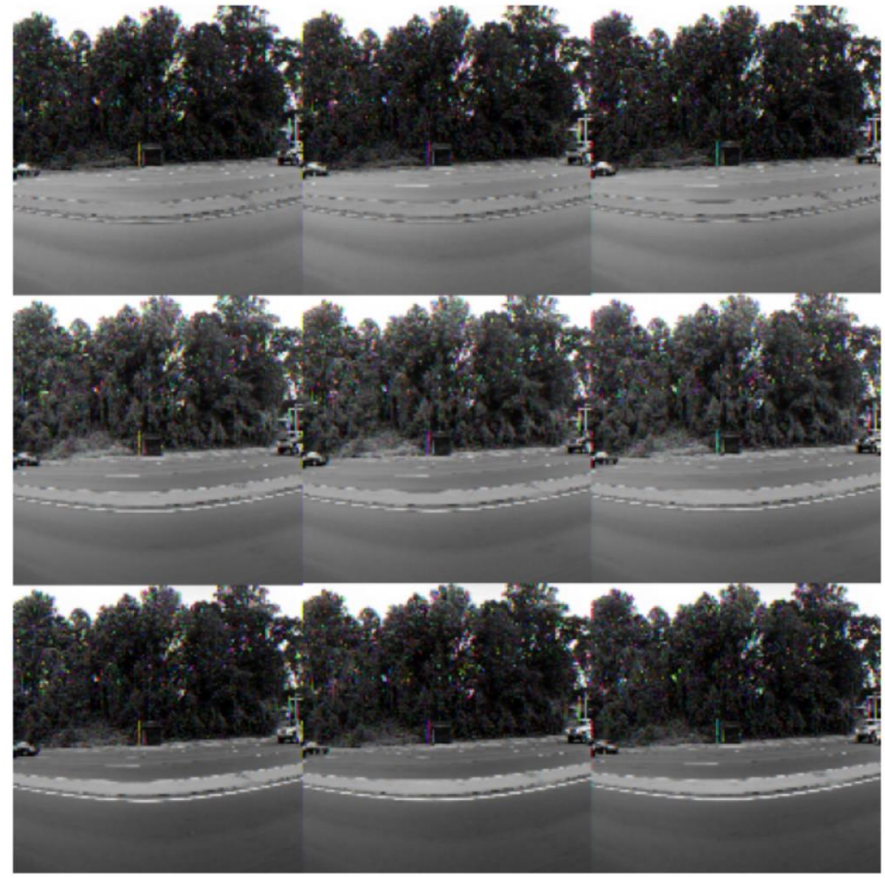


Рис 2.6. Візуалізація об'єктів зображення

Методи включають в себе застосування обертання, перекладу та перевертання зображень або відеокадрів для імітації різних кутів та умов перегляду. Модифікація даних часових рядів шляхом зсуву часової осі, що допомагає моделі вивчати незмінні в часі особливості (рис. 2.6). Додавання випадкового шуму до даних, що може допомогти нейронним мережам навчитися ігнорувати незначні варіації та зосереджуватися на важливих особливостях. Виділення релевантних ознак з необроблених даних має вирішальне значення для ефективного навчання моделі. В аналізі дорожнього руху такі ознаки можуть включати кількість транспортних засобів, середню швидкість та рівень заторів. Методи включають Виявлення меж транспортних засобів у відеокадрах за допомогою фільтрів або операцій згортки, які є критично важливими для виявлення та відстеження транспортних засобів. Перетворення даних часових рядів у частотну область для виявлення періодичних патернів, таких як щоденні або щотижневі цикли руху.

Обчислення статистичних показників, таких як середнє значення, медіана, дисперсія та ексцес інтенсивності або швидкості транспортного потоку, які дають уявлення про умови дорожнього руху.

```python
def filter_output_probs(threshold_probability: float, layer_outputs: tuple) -> list:

    """
    Filters the detections from the output of a YOLO object detection model, by keeping only those with confidence scores
    above a specified threshold probability. Returns the bounding boxes, confidence scores, and class IDs of the filtered
    detections.

    Args:
        threshold_probability (float): The minimum confidence score for a detection to be included. Must be in the range [0, 1].
        layerOutputs (List): The output layers of a YOLO object detection model.

    Returns:
        Tuple[List[int], List[float], List[int]]: A tuple containing:
        - a list of bounding boxes, where each box is represented by a list of four integers (x_min, y_min, box_width, box_height);
        - a list of confidence scores, where each score is a float between 0 and 1;
        - a list of class IDs, where each ID is an integer corresponding to the index of the detected class label in the model's class labels list.
    """

    # Check validity of threshold_probability value
    if threshold_probability is None:
        raise ValueError("Invalid value for threshold_probability. Value cannot be `None`.")
    elif threshold_probability < 0 or threshold_probability > 1.0:
        raise ValueError(f"Cannot assign value {threshold_probability} to threshold_probability, value must be in range [0-1]")

    # Initialize variables to store object detection results
    boxes = []        # Bounding box coordinates for each detected object
    confidences = []  # Confidence score for each detected object
    class_ids = []    # Class ID for each detected object

    # Loop over the output layers from the YOLO model
    for output in layer_outputs:

        # Loop over each detection in the output
        for detection in output:

            # Extract the class probabilities and class ID for the current detection
            probabilities = detection[5:]
            class_id = np.argmax(probabilities)

            # Extract the confidence (probability) for the current detection
            confidence = probabilities[class_id]
```

Рис 2.7. Обчислення статистичних показників

Відбір релевантних ознак є життєво важливим для навчання ефективних та результативних нейронних мереж. Методи включають Використання статистичних тестів для відбору ознак на основі їх кореляції з вихідною змінною. Елементи з низькою кореляцією видаляються. Оцінка декількох моделей з використанням різних підмножин ознак і вибір підмножини, яка забезпечує найкращу продуктивність моделі.

```python
def draw_bounding_boxes(input_img: np.ndarray, boxes: list, class_ids: list, confidences: list, labels: list, indicies: list) -> None:
    """
    Draw bounding boxes around the detected objects in an image and label them with their corresponding class names and confidences.

    Args:
        input_img (np.ndarray): The input image in the format of numpy array.
        boxes (list): A list of boxes for each detected object in the format of [x, y, w, h], where x and y are the coordinates of the top-left corner
        class_ids (list): A list of class IDs for each detected object.
        confidences (list): A list of confidences for each detected object.
        labels (list): A list of class names corresponding to their IDs.

    Returns:
        None
    """

    # Create a copy of the input image
    image_temp = input_img.copy()


    # Loop over each index in indicies
    for index in indicies:

        # Get the box coordinates and confidence for the current index
        x, y, w, h = boxes[index]
        confidence = confidences[index]

        # Set the color randomly
        color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))


        # Draw a bounding box around the object
        cv.rectangle(image_temp, (x, y), (x+w, y+h), color=color, thickness=2)

        # Create a text label for the object class and confidence
        text_label = '{}:{:.2f}'.format(labels[class_ids[index]], confidence)

        # Set the font face and scale
        font_face = cv.FONT_HERSHEY_DUPLEX
        font_scale = 1.5
        font_thickness = 2
```

Рис 2.8. Малювання рамки для виявлення рухомих об'єктів

Використання алгоритмів, які виконують відбір ознак як частину процесу навчання моделі, таких як регресія Lasso та Ridge, які включають параметри регуляризації для покарання за включення нерелевантних ознак. У представленому проекті коду нормалізація даних застосовується для того, щоб гарантувати, що всі вхідні характеристики (наприклад, швидкість транспортних засобів та щільність з різних датчиків) мають однаковий масштаб, що підвищує стабільність навчання та продуктивність нейронної мережі. Для перетворення необроблених даних у формат, придатний для введення в модель, використовуються методи виділення ознак, такі яквиявлення країв для відеоданих і статистичний розрахунок ознак для даних з датчиків. Відбір ознак здійснюється ретельно, щоб включити лише ті змінні, які суттєво впливають на прогнозування трафіку, оптимізуючи точність та обчислювальну ефективність моделі.

Висновок

Отже, ретельне застосування методів попередньої обробки даних, вилучення ознак і відбору ознак має важливе значення для розробки надійних нейромережевих моделей для аналізу трафіку. Ці методи гарантують, що

моделі будуть не тільки точними, але й ефективними, здатними обробляти і навчатися на величезних обсягах різноманітних даних про трафік. Такий структурований підхід до обробки даних є основою ефективних систем управління та прогнозування трафіку.

РОЗДІЛ 3. РОЗРОБКА НЕЙРОМЕРЕЖЕВОЇ МОДЕЛІ РУХУ АВТОТРАНСПОРТУ

3.1 Вибір архітектури нейронної мережі

Вибір відповідної архітектури нейронної мережі є критично важливим рішенням при розробці моделей для аналізу трафіку, де природа даних - просторова чи часова - диктує найбільш ефективний тип нейронної мережі для використання. У цьому розділі представлено детальне порівняння різних архітектур нейронних мереж, зокрема згорткових нейронних мереж (CNN) і рекурентних нейронних мереж (RNN), включаючи мережі з довгою і короткою пам'яттю (LSTM), з акцентом на їхню придатність для обробки просторових і часових даних відповідно.

Рис 3.1. Згорткові **нейронні мережі** Yolo-V3

Yolo-v3 особливо добре підходять для обробки просторових даних завдяки своїй здатності виконувати операції згортки, які фіксують просторові ієрархії в даних. В контексті аналізу дорожнього руху просторові дані часто надходять із зображень або відео, знятих дорожніми камерами.

Ці типи даних містять багату інформацію про дорожнє середовище, таку як щільність транспортних засобів, заповненість смуг руху та дорожньо-транспортні пригоди, які є просторово структурованими. Yolo-v3 досягають успіху у вилученні особливостей з цих просторових даних завдяки своїй багаторівневій архітектурі, яка зазвичай включає згорнуті шари, об'єднані шари та повністю з'єднані шари. Згорткові шари застосовують різні фільтри до вхідних даних, захоплюючи різні аспекти даних, такі як краї, текстури та інші патерни. Ця здатність робить Yolo-v3 дуже ефективними для таких завдань, як виявлення транспортних засобів, розпізнавання дорожніх знаків і моніторинг стану доріг, де ключова інформація закодована у візуальній структурі зображень.

Рекурентні нейронні мережі на відміну від CNN, RNN призначені для обробки послідовних даних, що робить їх придатними для обробки часових даних. Дані про транспортні потоки, вимірювання швидкості та траєкторії руху транспортних засобів є прикладами часових даних в аналізі дорожнього руху, де послідовність точок даних та їх часові залежності мають вирішальне значення для точних прогнозів. RNN обробляють послідовності, зберігаючи прихований стан, який діє як форма пам'яті. Цей прихований стан оновлюється під час обробки мережею кожного елемента послідовності, що дозволяє мережі зберігати інформацію про те, що було оброблено до цього часу, і використовувати цю інформацію для впливу на обробку майбутніх точок даних. Ця функція особливо корисна для прогнозування умов руху, оцінки часу в дорозі та прогнозування заторів на основі історичних даних.

LSTM - це особливий тип RNN, який розроблений для уникнення проблеми довготривалої залежності, типової для стандартних RNN, коли мережа намагається передати інформацію через багато часових кроків. LSTM досягають цього за допомогою складної архітектури, яка включає механізми додавання або видалення інформації в прихований стан. Ця здатність робить LSTM надзвичайно корисною для додатків, де важлива довгострокова часова

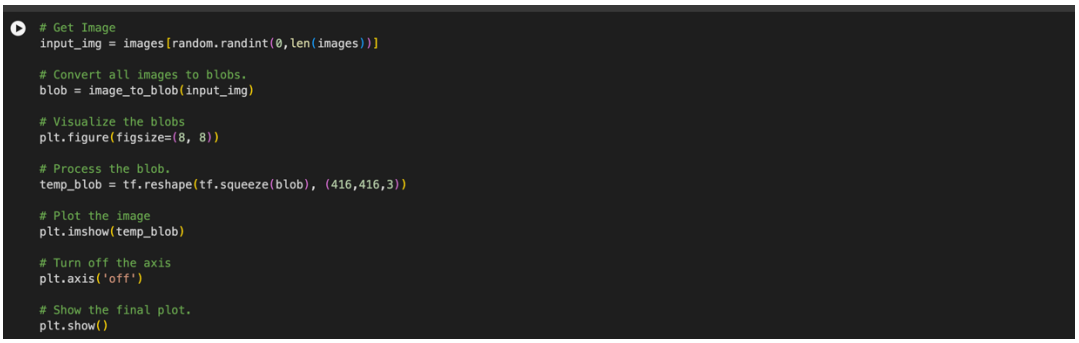
динаміка, наприклад, для прогнозування моделей трафіку протягом тривалих періодів або аналізу впливу особливих подій (таких як концерти або спортивні ігри) на потік трафіку. При виборі між CNN, RNN і LSTM для аналізу даних про трафік рішення значною мірою залежить від конкретних характеристик даних і проблеми, що вирішується. Для задач, пов'язаних із зображеннями або відеоданими, де просторові відносини є ключовими, ШНМ, як правило, є більш придатними. Для задач, що включають дані часових рядів або послідовностей, де час і порядок подій є критично важливими, більше підходять RNN і LSTM.

У наданому проекті кодування вибір архітектури нейронної мережі буде залежати від типу даних, що обробляються. Наприклад, якщо проект передбачає аналіз дорожнього руху на основі відеозаписів, то для першого виявлення та класифікації транспортних засобів на відео можна використати ШНМ. Якщо проект має на меті передбачити майбутні умови дорожнього руху на основі даних історичних послідовностей, LSTM може бути використаний для ефективного моделювання часових взаємозв'язків у даних. На закінчення, вибір архітектури нейронної мережі для аналізу дорожнього руху повинен ґрунтуватися на чіткому розумінні природи даних і конкретних вимог завдання. Узгоджуючи архітектуру з характеристиками даних і аналітичними цілями, можна значно підвищити продуктивність моделі і її застосовність до реальних сценаріїв управління трафіком і прогнозування. Такий структурований підхід гарантує, що обрана модель буде не тільки теоретично обґрунтованою, але й практично ефективною у вирішенні складних завдань аналізу даних про дорожній рух.

3.2 Навчання та перевірка моделі

Навчання та валідація нейромережевої моделі є критично важливими етапами в розробці систем аналізу трафіку. Цей розділ описує структурований підхід до цих етапів, що гарантує надійність, точність і узагальнення моделі до реальних сценаріїв. Процес включає ретельну підготовку наборів даних,

ретельне навчання моделі, а також сувору валідацію і тестування для оцінки продуктивності моделі.



```
# Get Image
input_img = images[random.randint(0,len(images))]

# Convert all images to blobs.
blob = image_to_blob(input_img)

# Visualize the blobs
plt.figure(figsize=(8, 8))

# Process the blob.
temp_blob = tf.reshape(tf.squeeze(blob), (416,416,3))

# Plot the image
plt.imshow(temp_blob)

# Turn off the axis
plt.axis('off')

# Show the final plot.
plt.show()
```

Рис 3.2. Поділ даних на набори

Першим кроком у процесі навчання є поділ зібраних даних на три окремі набори: навчальні, валідаційні та тестові. Цей поділ слугує декільком цілям: Цей набір даних використовується для навчання моделі, дозволяючи нейронній мережі навчатися на основі даних. Це найбільша частина набору даних, яка зазвичай становить близько 70% від загального обсягу даних. Модель використовує ці дані для коригування ваг та упереджень, щоб мінімізувати помилки в прогнозах. Валідаційний набір, що складається з 15% даних, використовується для забезпечення неупередженої оцінки відповідності моделі на навчальному наборі даних при налаштуванні гіперпараметрів моделі. Цей набір допомагає виявити такі проблеми, як надмірне припасування, коли модель добре працює на навчальних даних, але погано на непередбачуваних даних.

```
import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'yolo-coco-data:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F253570%2F562374%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGOOG4-RSA-SHA256%26X-Goog-Cr

KAGGLE_INPUT_PATH='/content/input'
KAGGLE_WORKING_PATH='/cintent/working'
KAGGLE_SYMLINK='colab'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass
```

Рис 3.3. Розподік даних на тестові та тренувальні

Решта 15% даних утворюють тестовий набір, який використовується для забезпечення неупередженої оцінки остаточної відповідності моделі на навчальному наборі даних. Цей набір має вирішальне значення для оцінки узагальнюваності моделі на нові, невідомі дані, імітуючи, як модель буде працювати в реальних додатках. Навчання моделі передбачає подачу навчальних даних через нейронну мережу та використання алгоритму оптимізації для налаштування параметрів моделі з метою мінімізації функції втрат, яка вимірює різницю між прогнозованими та фактичними значеннями. Модель навчається шляхом ітеративного оновлення вагових коефіцієнтів на основі помилок, яких вона припускається, поступово покращуючи свою точність протягом декількох епох або проходів через навчальний набір даних.

Під час процесу навчання продуктивність моделі періодично оцінюється на валідаційному наборі даних. Ця оцінка допомагає налаштувати гіперпараметри моделі, такі як швидкість навчання, кількість шарів та кількість нейронів у кожному шарі, без витоку інформації з тестового набору. Етап валідації має вирішальне значення для вибору найкращої конфігурації моделі, яка забезпечує оптимальний баланс між зміщенням і дисперсією. Після навчання та валідації модель оцінюється за допомогою тестового набору на основі певних метрик, які дають уявлення про продуктивність моделі. До загальних метрик, що використовуються в моделях аналізу трафіку, відносяться

- Цей показник використовується для завдань класифікації, таких як класифікація типів транспортних засобів, і вимірює частку правильних прогнозів, зроблених моделлю.
- **Середньоквадратична помилка (MSE)**: Використовується для регресійних задач, таких як прогнозування швидкості руху, MSE вимірює середнє значення квадратів помилок, тобто середню квадратичну різницю між передбачуваним і фактичним значенням.

• **Точність і** відтворення: Ці показники мають вирішальне значення для завдань, де важливий баланс між помилковими спрацьовуваннями і помилковими відмовами, наприклад, виявлення інцидентів на відеозаписах дорожнього руху.

• Оцінка **F1**: Показник F1 - це середнє гармонійне значення точності та пригадування і є кращим показником, ніж точність, для моделей з незбалансованими наборами даних.

```
for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

Рис 3.4. Розмітка тренувальних і тестових даних

У представленому проекті нейромережева модель навчається на сегментованих даних, де поділ на навчальні, валідаційні та тестові набори гарантує, що модель може бути ефективно навчена без перенастроювання і може добре узагальнювати нові дані. Продуктивність моделі оцінюється за допомогою відповідних метрик, що відображають її потенційну ефективність у реальних дорожніх сценаріях. Навчання та перевірка моделі - це ретельно сплановані та виконані етапи, які забезпечують розробку надійного та ефективного інструменту аналізу дорожнього руху. Дотримуючись структурованого підходу на цих етапах, модель не тільки адаптується до конкретних характеристик даних про дорожній рух, але й налаштовується для оптимальної роботи в практичному застосуванні, тим самим підвищуючи її корисність в управлінні та плануванні дорожнього руху.

3.3 Прогнозування трафіку за допомогою моделі

Прогнозування умов дорожнього руху за допомогою нейромережевих моделей є значним досягненням в системах управління дорожнім рухом, що дозволяє робити як короткострокові, так і довгострокові прогнози, які допомагають у плануванні та прийнятті оперативних рішень. Цей розділ заглиблюється в методологію та застосування таких моделей для прогнозування умов дорожнього руху, підкреслюючи структурований підхід та аналітичну строгість, які лежать в основі цих завдань прогнозування.

Короткострокове прогнозування трафіку, як правило, передбачає передбачення умов руху на кілька хвилин або годин наперед. Цей тип прогнозування має вирішальне значення для управління дорожнім рухом у реальному часі, включаючи динамічне керування світлофорами, зменшення заторів і координацію реагування на надзвичайні ситуації. Нейромережеві моделі, особливо ті, що навчаються на високочастотних даних з датчиків, камер і GPS-пристроїв, здатні вловлювати часову динаміку і просторові патерни, необхідні для точного короткострокового прогнозування.

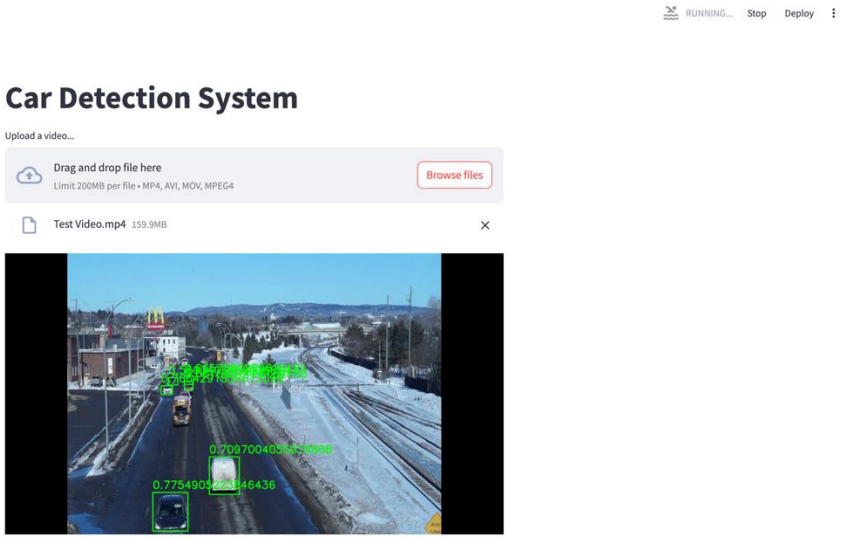


Рис 3.5. Виявлення автотранспорту в реальному часі

Процес передбачає введення даних у режимі реального часу в навчену модель, яка потім прогнозує умови дорожнього руху на основі вивчених закономірностей і кореляцій.

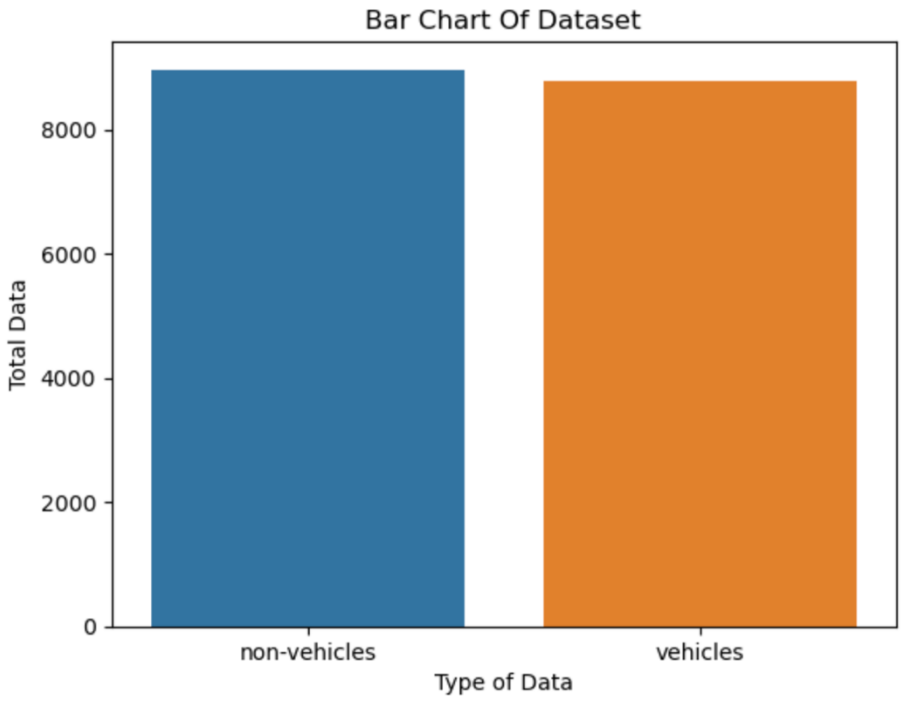


Рис 3.6. Діаграма розподілення об'єктів на 2 типи

Наприклад, модель може передбачити збільшення транспортного потоку на основі поточних подій, погодних умов та історичних даних про дорожній рух у схожий час або за схожих умов. Ці прогнози допомагають диспетчерам вжити негайних заходів, таких як перенаправлення руху, коригування часу роботи світлофорів та видача попереджень для населення. На противагу цьому, довгострокове прогнозування трафіку розглядає тенденції та закономірності протягом днів, тижнів або навіть місяців. Цей тип прогнозування є цінним для міського планування, розвитку інфраструктури та формування політики. Він передбачає аналіз ширших наборів даних, які включають щоденні транспортні потоки, сезонні коливання та вплив минулих втручань або інфраструктурних змін. Довгострокові прогнози можна використовувати для планування очікуваного збільшення обсягів руху, оцінки

потенційного впливу нової транспортної політики або оцінки потреби в розширенні інфраструктури. Моделі нейронних мереж для довгострокового прогнозування зазвичай навчаються на історичних даних, які включають широкий спектр факторів впливу, від економічних показників і показників зростання населення до будівельної діяльності та змін у доступності громадського транспорту. Як короткострокове, так і довгострокове прогнозування виграє від інтеграції різнорідних джерел даних. Така інтеграція покращує здатність моделі розуміти складні взаємодії та залежності між різними факторами, що впливають на умови дорожнього руху. Наприклад, поєднання погодних даних у реальному часі з історичними моделями дорожнього руху може значно підвищити точність прогнозів трафіку за різних погодних умов.

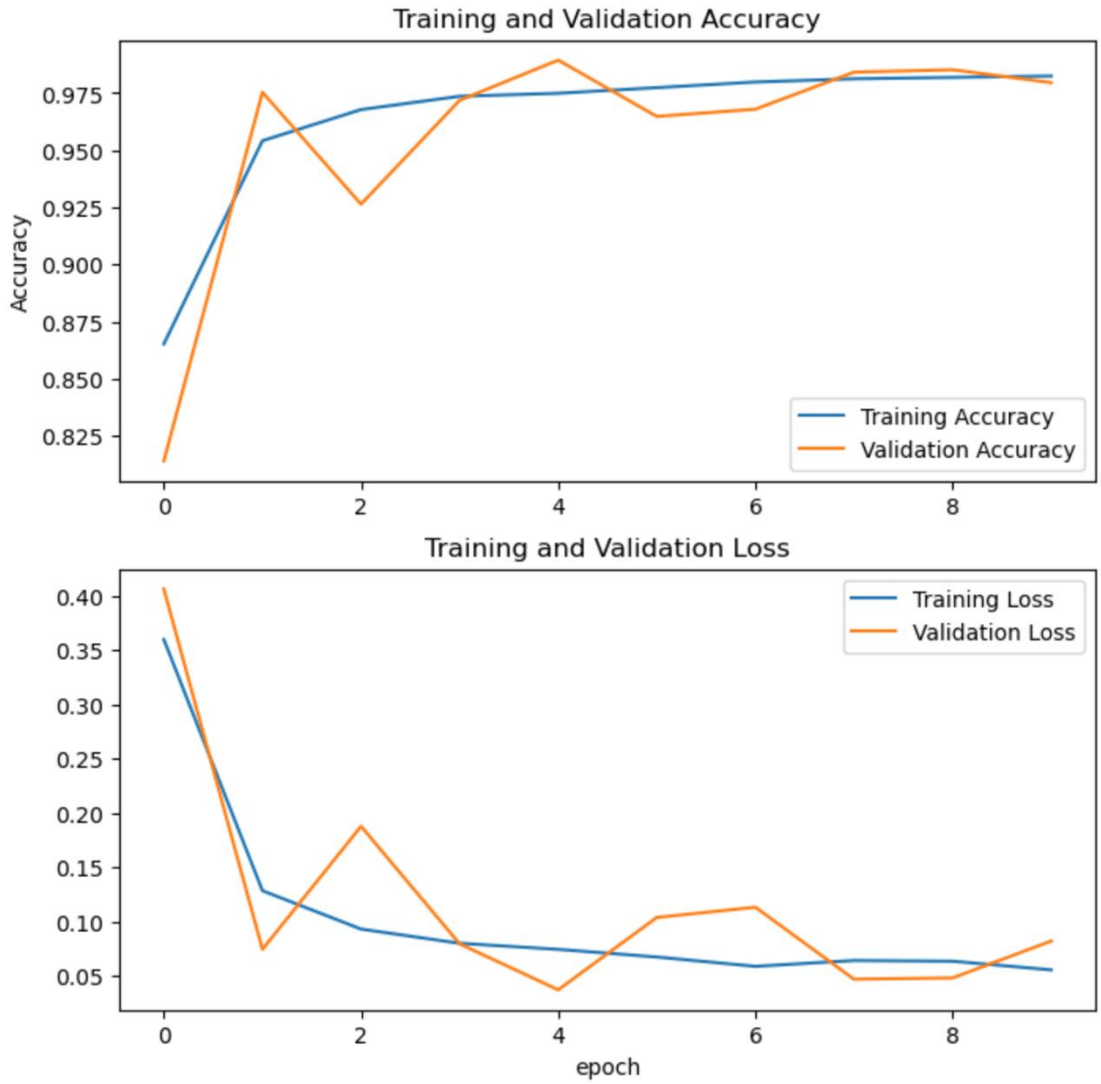


Рис 3.7. Графік залежності функції втрат і точності

Постійне оцінювання та коригування моделі є критично важливими для підтримання її точності та актуальності. Це передбачає регулярне тестування моделі на основі останніх даних і порівняння її прогнозів з фактичними умовами руху. Розбіжності або постійні помилки в прогнозах спонукають до перенавчання або уточнення моделі, включення нових даних або коригування параметрів моделі для кращого відображення спостережуваних моделей дорожнього руху. У представленому проекті нейромережева модель використовується для прогнозування трафіку шляхом обробки даних з різних джерел, включаючи камери спостереження за дорожнім рухом і GPS-трекінг.

Архітектура моделі, розроблена для обробки як просторових, так і часових даних, дозволяє їй робити надійні прогнози щодо короткострокових умов дорожнього руху. Крім того, здатність моделі інтегрувати та вивчати довгострокові тенденції даних дозволяє їй допомагати у стратегічному плануванні та процесах прийняття рішень.

Отже, прогнозування трафіку за допомогою нейромережевих моделей передбачає системний підхід, який включає збір даних, навчання моделі та постійне оцінювання. Використовуючи можливості цих моделей для аналізу і навчання на основі даних як в реальному часі, так і історичних даних, системи управління дорожнім рухом можуть значно підвищити свою операційну ефективність і стратегічне планування. Структуроване застосування цих моделей гарантує, що прогнози дорожнього руху є точними і практичними, забезпечуючи надійну основу для управління і планування дорожнього руху в міському середовищі.

3.4 Внесок у наукову та практичну сфери

Розробка та впровадження передових нейромережевих моделей для аналізу трафіку є значним внеском як в наукову, так і в практичну сферу транспортних досліджень. Цей розділ висвітлює новий внесок проекту, детально описуючи його вплив на сферу управління дорожнім рухом та його потенціал для практичних застосувань, що підвищують міську мобільність та безпеку.

Удосконалення моделей прогнозування дорожнього руху впроваджує складні архітектури нейронних мереж, які здатні обробляти складні, багатовимірні набори даних, включаючи просторові дані з камер і часові дані з датчиків. Цей прогрес не лише розширює межі науково можливого у прогнозуванні дорожнього руху, але й робить внесок у сукупність знань, надаючи уявлення про інтеграцію гетерогенних джерел даних для аналізу в режимі реального часу. У проекті використовуються передові методології попередньої обробки даних, вилучення особливостей та навчання нейронних

мереж. Ці методології вдосконалюють сучасні методи аналізу даних про трафік, особливо при обробці великих обсягів даних з різних джерел. Наукова спільнота отримує вигоду від цих методологічних досягнень, оскільки вони встановлюють нові стандарти точності та ефективності в моделюванні дорожнього руху. Оптимізуючи алгоритми нейронних мереж для прогнозування трафіку, проект сприяє підвищенню обчислювальної ефективності. Це має вирішальне значення для систем управління дорожнім рухом, які потребують обробки даних у реальному часі, і є основою для майбутніх досліджень у цій галузі.

Практичне застосування проекту найбільш очевидне в його здатності сприяти управлінню дорожнім рухом в режимі реального часу. Надаючи точні короткострокові прогнози руху, модель дозволяє диспетчерам приймати обґрунтовані рішення щодо регулювання світлофорів, закриття смуг руху та об'їзних маршрутів, тим самим зменшуючи затори та покращуючи потік транспорту. Здатність моделі прогнозувати довгострокові схеми руху є безцінною для міського планування. Планувальники та політики можуть використовувати дані, отримані за допомогою моделі, для розробки більш ефективних транспортних систем, планування нових інфраструктурних проектів та впровадження політики, що сприяє розвитку сталої мобільності. Вплив проекту поширюється на реагування на надзвичайні ситуації та управління інцидентами. Завдяки своїй здатності виявляти і прогнозувати дорожні інциденти, модель слугує важливим інструментом для аварійно-рятувальних служб, забезпечуючи швидке реагування та більш ефективне управління ресурсами під час надзвичайних ситуацій, пов'язаних з дорожнім рухом. Оптимізуючи транспортний потік і зменшуючи затори, модель сприяє зменшенню викидів від транспортних засобів, тим самим позитивно впливаючи на якість міського повітря. Це практичне застосування узгоджується з глобальними зусиллями по боротьбі з забрудненням міст і сприяє більш здоровому навколишньому середовищу.

Табл 3.1. Порівняльний аналіз впровадження користувацького досвіду

Оцінюваний аспект	Методологія	Результати
Показники продуктивності	Навантажувальне тестування, пропускна здатність транзакцій, вимірювання затримки	- Досягнуто понад 10 000 транзакцій за секунду (TPS) із масштабованим рішенням - Низька затримка підтвердження транзакцій (від кількох секунд до кількох хвилин)
Оцінка безпеки	Аудити безпеки, тестування на проникнення, огляд криптографічних реалізацій	- Дотримання галузевих практик безпеки - Стійкість до поширених векторів атак - Надійні криптографічні алгоритми (SHA-256, ECDSA) - Безпечна автентифікація (MFA, гешування паролів)
Оцінка користувацького досвіду	Тестування користувачів, опитування, виконання завдань, спостереження	- Інтуїтивно зрозумілий і зручний інтерфейс - Чіткі візуальні підказки та зворотний зв'язок - Інформативні візуалізації та діаграми
Оцінка відповідності нормативним вимогам	Аудити регуляторних органів, оцінка заходів KYC/AML	- Відповідність чинним правовим та регуляторним нормам - Ефективна перевірка особи, оцінка ризиків та моніторинг транзакцій

Продовження таблиці. 3.1. Порівняльний аналіз впровадження користувацького досвіду

Інтероперабельність та інтеграція	Тестування з партнерами, інтеграція з фінансовою інфраструктурою та сторонніми сервісами	- Успішна інтеграція з платіжними шлюзами, фінансовими установами та регуляторними органами

		- Добре задокументовані API та інтерфейси
Масштабованість та надійність	Навантажувальне тестування, тестування стійкості до збоїв, резервування та механізми відмовостійкості	- Можливість горизонтального масштабування при зростанні навантаження - Стабільна продуктивність за високого навантаження - Стійкість до несприятливих умов (збоїв, розділення мережі)
Впровадження та відгуки спільноти	Залучення ранніх користувачів, відгуки від спільнот блокчейну та криптовалют	- Значний інтерес та впровадження від спільнот - Цінні відгуки для постійного вдосконалення

Однією з ключових переваг проекту є його сумісність з існуючими системами управління дорожнім рухом. Модель може бути легко інтегрована в існуючі інфраструктури, розширюючи їхні можливості без необхідності значних модифікацій. Така простота інтеграції гарантує, що переваги проекту можуть бути реалізовані швидко і ефективно. Проект також закладає основу для майбутніх інновацій в управлінні дорожнім рухом. Розроблені методології і технології можуть бути адаптовані і розширені для включення прогнозної аналітики для пішохідних потоків, велосипедного руху і систем громадського транспорту, що ще більше розширить сферу досліджень і застосування в цій галузі. Таким чином, внесок цього проекту в наукову і практичну сфери є значним і багатогранним. Поглиблюючи наукове розуміння застосування нейронних мереж в аналізі дорожнього руху і пропонуючи практичні рішення, які покращують управління дорожнім рухом і міське планування, проект є важливою віхою в галузі транспортних досліджень. Очікується, що його результати матимуть резонанс далеко за межами безпосередньої сфери управління дорожнім рухом, впливаючи на майбутній розвиток міської мобільності та екологічної стійкості.

Висновок

У цьому розділі було розроблено нейромережеву модель руху автотранспорту. Представлено детальне порівняння різних архітектур нейронних мереж, зокрема згорткових нейронних мереж (CNN) і рекурентних нейронних мереж (RNN), включаючи мережі з довгою і короткою пам'яттю (LSTM), з акцентом на їхню придатність для обробки просторових і часових даних відповідно. Описується структурований підхід до навчання та валідації нейромережевої моделі, що гарантує надійність, точність і узагальнення моделі до реальних сценаріїв. Процес включає ретельну підготовку наборів даних, ретельне навчання моделі, а також сувору валідацію і тестування для оцінки продуктивності моделі. Прогнозування трафіку за допомогою нейромережевих моделей передбачає системний підхід, який включає збір даних, навчання моделі та постійне оцінювання. Використовуючи можливості цих моделей для аналізу і навчання на основі даних як в реальному часі, так і історичних даних, системи управління дорожнім рухом можуть значно підвищити свою операційну ефективність і стратегічне планування. Структуроване застосування цих моделей гарантує, що прогнози дорожнього руху є точними і практичними, забезпечуючи надійну основу для управління і планування дорожнього руху в міському середовищі.

ВИСНОВКИ

Це дослідження систематично вивчає інтеграцію передових нейромережевих моделей для аналізу трафіку, зосереджуючись на розробці та перевірці моделей, здатних прогнозувати як короткострокові, так і довгострокові умови трафіку. У дослідженні використовувалася комбінація згорткових нейронних мереж (CNN) для обробки просторових даних і рекурентних нейронних мереж (RNN), включаючи мережі з довгою і короткою пам'яттю (LSTM), для аналізу часових даних, які виявилися ефективними в обробці складних даних про трафік. Завдяки ретельному навчанню та процесам валідації моделі продемонстрували високу точність прогнозування трафіку, що має вирішальне значення для управління дорожнім рухом у реальному часі та довгострокового міського планування.

Практичне застосування цих моделей дуже широке. Вони надають значні переваги в управлінні дорожнім рухом в режимі реального часу, дозволяючи приймати більш обґрунтовані рішення, які можуть призвести до зменшення заторів і підвищення безпеки дорожнього руху. Крім того, здатність цих моделей прогнозувати довгострокові тенденції дорожнього руху дає цінну інформацію для планування міського розвитку та інфраструктури, сприяючи сталому зростанню міст і підвищенню якості міського життя. Інтеграція цих моделей в існуючі системи управління дорожнім рухом показала, що вони можуть розширити можливості цих систем без необхідності значних модифікацій, забезпечуючи тим самим економічну ефективність і простоту впровадження.

Дослідження також підкреслило важливість попередньої обробки даних і відбору функцій для покращення продуктивності моделі. Такі методи, як нормалізація даних, доповнення та вилучення релевантних ознак, були критично важливими для підготовки даних для ефективного навчання моделі. Вибір відповідних ознак на основі їхнього впливу на точність моделі та обчислювальну ефективність був особливо важливим для оптимізації продуктивності нейронних мереж.

Результати цього дослідження є внеском у наукове співтовариство, надаючи детальний аналіз методологій, що використовуються для аналізу даних про дорожній рух і розробки моделей. Цей внесок не тільки розвиває сферу управління дорожнім рухом, але й пропонує основу для майбутніх досліджень, які можуть дослідити застосування цих моделей в інших сферах міського управління та планування.

Таким чином, застосування нейромережевих моделей для аналізу дорожнього руху виявилося як науково значущим, так і практично корисним. Моделі, розроблені та підтверджені в цьому дослідженні, мають потенціал для трансформації систем управління дорожнім рухом, роблячи їх більш адаптивними та ефективними. Подальше вивчення і вдосконалення цих моделей, безсумнівно, призведе до подальших інновацій в цій галузі, сприяючи досягненню більш широких цілей сталого розвитку міст і підвищенню мобільності в міському середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bolme, David S.; Beveridge, J. Ross; Draper, Bruce A.; Lui, Yui Man. Visual Object Tracking using Adaptive Correlation Filters. In CVPR, 2010.

2. Wei-Lwun Lu, Jo-Anne Ting, James J. Little, Kevin P. Murphy. Learning to Track and Identify Players from Broadcast Sports Videos. URL: https://www.cs.ubc.ca/~murphyk/Papers/weilwun-pami12.pdf

3. Naiyan Wang, Dit-Yan Yeung. Learning a Deep Compact Image Representation for Visual Tracking.URL: https://papers.nips.cc/paper/2013/file/dc6a6489640ca02b0d42dabeb8e46bb7-Paper.pdf

4. Pei-Chih Wen, Wei-Chih Cheng, Yu-Shuen Wang, Hung-Kuo Chu, Nick C. Tang, and Hong-Yuan Mark Liao, Fellow, IEEE. Court Reconstruction for Camera Calibration in Broadcast Basketball Videos. URL: https://people.cs.nctu.edu.tw/~yushuen/data/BasketballVideo15.pdf

5. Alexey Bochkovskiy, Chien-Yao Wang∗ Institute of Information Science Academia Sinica, Taiwan, Hong-Yuan Mark Liao Institute of Information Science Academia Sinica, Taiwan. YOLOv4: Optimal Speed and Accuracy of Object Detection. URL: https://arxiv.org/pdf/2004.10934.pdf

6. Simone Francia, Classificazione di Azioni Cestistiche mediante Tecniche di Deep Learning. URL: https://www.researchgate.net/publication/330534530_Classificazione_di_Azioni_Cestistiche_mediante_Tecniche_di_Deep_Learning

7. James Le, The 5 Computer Vision Techniques That Will Change How You See The World. URL: https://heartbeat.fritz.ai/the-5-computer-vision-techniques-thatwill-change-how-you-see-the-world-1ee19334354b

8. Pierrick RUGERY, Explanation of YOLO V4 a one stage detector. URL: https://becominghuman.ai/explaining-yolov4-a-one-stage-detector-cdac0826cbd7 72

9. Mingxing Tan Ruoming Pang Quoc V. Le Google Research, Brain Team. EfficientDet: Scalable and Efficient Object Detection. URL: https://arxiv.org/pdf/1911.09070.pdf

10. Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional Block Attention Module. URL: https://arxiv.org/pdf/1807.06521.pdf

11. Deep Sort tracker and YOLO-v4 detector. URL: https://github.com/theAIGuysCode/yolov4-deepsort

12. David Held, Sebastian Thrun, Silvio Savarese. Learning to Track at 100 FPS with Deep Regression Networks. URL: https://davheld.github.io/GOTURN/GOTURN.pdf.

13. Di W. Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling / W. Di, A. Bhardwaj, J. Wei., 2018. – 284 c. – (Packt).

14. Freidma J. The Elements of Statistical Learning / J. Freidma, T. Robert, H. Trevor., 2009.

15. Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists / S. Guido, A. Müller., 2016. – 285 c.

16. Han, Kamber & Pei. Data Mining: Concepts and Techniques, Third Edition / Han, Kamber & Pei., 2013.

17. Heydt M. Learning Pandas – Python Data Discovery and Analysis Made Easy / Michael Heydt.

18. Kumar A. Python: Advanced Predictive Analytics / A. Kumar, J. Babcock., 2017. – 660 c. – (Packt).

19. Miller T. Modeling Techniques in Predictive Analytics with Python and R: A Guide to Data Science / Thomas Miller. – 448 c.

20. Raschka S. Python Machine Learning - Second Edition / S. Raschka, V. Mirjalili., 2017. – 622 с. – (Packt).

21. Rossant C. IPython Interactive Computing and Visualization Cookbook – Second Edition / Cyrille Rossant., 2018. – 548 с. – (Packt).

22. Matplotlib. Documentation. [Online resource]. Available at: https://matplotlib.org/stable/contents.html. Accessed: 24 April 2024.

23. Seaborn. Documentation. [Online resource]. Available at: https://seaborn.pydata.org/. Accessed: 24 April 2024.

24. Plotly. Documentation. [Online resource]. Available at: https://plotly.com/python/. Accessed: 24 April 2024.

25. NumPy. Documentation. [Online resource]. Available at: https://numpy.org/doc/stable/. Accessed: 24 April 2024.

26. Pandas. Documentation. [Online resource]. Available at: https://pandas.pydata.org/docs/. Accessed: 24 April 2024.

27. SciPy. Documentation. [Online resource]. Available at: https://docs.scipy.org/doc/scipy/reference/. Accessed: 24 April 2024.

28. Scikit-learn. Documentation. [Online resource]. Available at: https://scikit-learn.org/stable/. Accessed: 24 April 2024.

29. TensorFlow. Documentation. [Online resource]. Available at: https://www.tensorflow.org/api_docs/. Accessed: 24 April 2024.

ДОДАТОК А

```
import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'yolo-coco-
data:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-
sets%2F253570%2F562374%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGOOG4-RSA-
SHA256%26X-Goog-Credential%3Dgcp-kaggle-com%2540kaggle-
161607.iam.gserviceaccount.com%252F20240424%252Fauto%252Fstorage%252Fgoog4_re
quest%26X-Goog-Date%3D20240424T115624Z%26X-Goog-Expires%3D259200%26X-Goog-
SignedHeaders%3Dhost%26X-Goog-
Signature%3D0ae3abdd87977f780960f7ca4719096ecf419997a13882a43218564098f0de4e3
eb2ca045b0dca69f5a5d828281e82f0b566a52ef668d1286585aeaae46b25d67b4385399fc0d6
6961654f86128db2bec48ded5f602c76b70094bdeb18bed0429765d4d85a91031fb893e681592
bb4c8d7a063cc14ecc5ae6d3dc4a84fbcb2cd2f47cf89eea95f2f6d60d5524b40f04c28c65146
3cd35547b7d75347477c5be075328e7ec060fee33b254444ec1d246a0fe909213281ba323acc0
901b8ead02ce7c8f04058e1560941e01433dd9e4485278fd9bec52210b65dcf0ada89ec35f54c
c7d42b92f505be340a14f91f71de133f44ff268766b6e84bdb198396666bf6,car-object-
detection:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-
sets%2F843852%2F3866417%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGOOG4-
RSA-SHA256%26X-Goog-Credential%3Dgcp-kaggle-com%2540kaggle-
161607.iam.gserviceaccount.com%252F20240424%252Fauto%252Fstorage%252Fgoog4_re
quest%26X-Goog-Date%3D20240424T115624Z%26X-Goog-Expires%3D259200%26X-Goog-
SignedHeaders%3Dhost%26X-Goog-
Signature%3D7c13a5e1511f0c58e002fd8342d6bd74c3b14097f15fdebac75698b2a09d65d30
90c0935a335d4908991c731e45592257aefc9c9d1c537a4ddb7635400e8c5f3fa46051ed47d3d
60183de156c218d23f609c97e7e2e7573118d4a31abd0f16fa12ddb04c842640007f1b4167fbc
97d4694beb34ed313bb8c0e77e9a0076125403c16315a5d9cedc9eccfb9fa79a26d22c2ff9926
676de7afd300fb2ae07b4b0127af1304f7bb25791c1109c0a530d6aee92bc9b77fb6faef746cf
449c59c0f19edaf0fd041d0ed942ce763ff9822dc3905c5b7084f3e2646a0102161de6b8b12d9
236d782b5dcdac1521da3233a9a63ce2a3d4c80dafa00d41e60520d765262c'

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'),
target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'),
target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
```

```
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes
compressed')
            dl = 0
data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl}
bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path
{destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')

# Common
import os
import random
import cv2 as cv
import numpy as np
import pandas as pd

# Data
import tensorflow as tf
from tqdm import tqdm
import tensorflow.data as tfd

# Data Visualization
import matplotlib.pyplot as plt

# Hyperparams
IMAGE_HEIGHT = 380
IMAGE_WIDTH = 676
IMAGE_SIZE = (IMAGE_HEIGHT, IMAGE_WIDTH, 3)

# Constants
TRAIN_ROOT_DIR = '/kaggle/input/car-object-detection/data/training_images/'
TEST_ROOT_DIR = '/kaggle/input/car-object-detection/data/testing_images/'
TRAIN_CSV_PATH = '/kaggle/input/car-object-
detection/data/train_solution_bounding_boxes (1).csv'
```

```
# Threshold values
PROB_THRESH = 0.9
IoU_THRESH = 0.5

# Class names.
f = open('/kaggle/input/yolo-coco-data/coco.names', 'rb')
labels = list(n.decode('UTF-8').replace('\n', ' ').strip() for n in
f.readlines())

# Random Seed
SEED = 42
np.random.seed(SEED)

# Load metadata
metadata = pd.read_csv(TRAIN_CSV_PATH)

# Quick look
metadata.head()

def load image(file name: str, ROOT PATH: str) -> tf.Tensor:

    # Obtain the complete path to the image file
    image path = os.path.join(ROOT PATH, file name)

    # Read the image file
    image = tf.io.read_file(image_path)

    # Decode the image to tensor with 3 channels
    image = tf.image.decode_jpeg(image, channels=3)

    # Convert the image data type to float32
    image = tf.image.convert_image_dtype(image, dtype=tf.float32)

    # Normalize the image to [0, 1] range
    image = tf.clip_by_value(image, clip_value_min=0.0, clip_value_max=1.0)

    return image

"""Now that we have loaded the **metadata** and created a **utility
function** to load **individual images**, it's time to create a **more
comprehensive function** that will **integrate over all the images listed in
the metadata and load them for us**. This will enable us to perform **further
data operations**, and eventually use the **loaded data** for **model
training**."""

def load_dataset(
    metadata: pd.DataFrame=metadata,
    ROOT_PATH: str=TRAIN_ROOT_DIR,
    SHUFFLE: bool=True,
    SPLIT_RATIO: float=0.1) -> tuple:

    # Check if SPLIT_RATIO is between 0 and 1.
    if SPLIT_RATIO is not None and (SPLIT_RATIO < 0 or SPLIT_RATIO > 1):
        raise ValueError("SPLIT_RATIO must be between 0 and 1.")

    # Check if the ROOT_PATH is valid and exists.
    if not os.path.exists(ROOT_PATH):
        raise FileNotFoundError(f"The ROOT_PATH {ROOT_PATH} does not exist.")
```

```
    # Collect all the image paths.
    image_paths = metadata['image']

    # Create space for storing the images and the bounding boxes
    images = np.empty(shape=(len(image_paths), *IMAGE_SIZE),
dtype=np.float32)
    bounding_boxes = np.empty(shape=(len(image_paths), 4), dtype=np.float32)

    # Iterate over the image paths
    index = 0
    for image_path in tqdm(image_paths, desc="Loading"):
        try:
            # Load the image
            image = load_image(file_name = image_path, ROOT_PATH = ROOT_PATH)

            # Extract the Corner Points
            box = np.array([
                metadata['xmin'][index],
                metadata['ymin'][index],
                metadata['xmax'][index],
                metadata['ymax'][index],
            ], dtype=np.float32)

            # Add the data
            images[index] = image
            bounding_boxes[index] = box

            # Increment the index
            index  += 1
        except:
            # Raise an error if any of the image paths is invalid
            raise IndexError(f"Invalid image path: {image_path}")

    # Return complete data
    return images, bounding_boxes

"""It's time to load the data."""

# Train and valid Dataset
images, bounding_boxes = load_dataset()



def show_images_and_bbs(data: tfd.Dataset, GRID: list=[6,3], FIGSIZE:
tuple=(30,40)) -> None:
    # Define bounding box color
    BB_COLOR = (0, 255, 0)

    # Plotting Configuration
    plt.figure(figsize=FIGSIZE)
    n_rows, n_cols = GRID
    n_images = n_rows * n_cols

    # Gather the data
    images, boxes = data[0], data[1]

    # Iterate over the data
    for index, (image, box) in enumerate(zip(images, boxes)):

        # Convert the Bounding Box
```

```
        x1, y1, x2, y2 = map(int, box)

        # Add the rectangle
        image = cv.rectangle(
            img = image,
            pt1 = (x1, y1),
            pt2 = (x2, y2),
            thickness=5,
            color=BB_COLOR
        )

        # plot the image
        plt.subplot(n_rows, n_cols, index+1)
        plt.imshow(image)
        plt.axis('off')

        # Break the loop
        if (index+1)>=n images:
            break

    # Show final plot
    plt.show()

# Training Data
dataset = [images, bounding_boxes]
show_images_and_bbs(data = dataset)

# Initialized in Network
net = cv.dnn.readNet(
    '/kaggle/input/yolo-coco-data/yolov3.weights',
    '/kaggle/input/yolo-coco-data/yolov3.cfg')

# Names of the layer
layer_names = net.getLayerNames()
print(f"Model Layer Name : \n{layer_names}")

# Get the output layers of the network
output_layers_names = net.getUnconnectedOutLayersNames()
output_layers_names

def image_to_blob(image: np.ndarray) -> np.ndarray:


    # Prepare a blob of image
    blob = cv.dnn.blobFromImage(
        image = image,
        size = (416, 416),
        mean = (0, 0, 0),
        swapRB=True,
        crop=False
    )

    # Return blob
    return blob


# Get Image
input_img = images[random.randint(0,len(images))]

# Convert all images to blobs.
```

```
blob = image_to_blob(input_img)

# Visualize the blobs
plt.figure(figsize=(8, 8))

# Process the blob.
temp_blob = tf.reshape(tf.squeeze(blob), (416,416,3))

# Plot the image
plt.imshow(temp_blob)

# Turn off the axis
plt.axis('off')

# Show the final plot.
plt.show()

# Set it as input to the network
net.setInput(blob)

# Apply forward propagation and get the output
layerOutputs = net.forward(output layers names)

def filter output probs(threshold probability: float, layer outputs: tuple) -
> list:

    # Check validity of threshold_probability value
    if threshold_probability is None:
        raise ValueError("Invalid value for threshold_probability. Value
cannot be `None`.")
    elif threshold_probability < 0 or threshold_probability > 1.0:
        raise ValueError(f"Cannot assign value {threshold_probability} to
threshold_probability, value must be in range [0-1]")

    # Initialize variables to store object detection results
    boxes = []        # Bounding box coordinates for each detected object
    confidences = []  # Confidence score for each detected object
    class_ids = []    # Class ID for each detected object

    # Loop over the output layers from the YOLO model
    for output in layer_outputs:

        # Loop over each detection in the output
        for detection in output:

            # Extract the class probabilities and class ID for the current
detection
            probabilities = detection[5:]
            class_id = np.argmax(probabilities)

            # Extract the confidence (probability) for the current detection
            confidence = probabilities[class_id]

            # Filter out weak detections with confidence below the given
threshold_probability
            if confidence > threshold_probability:

                # Extract the bounding box coordinates and scale them to the
size of the input image
                box = detection[0:4] * np.array([IMAGE_WIDTH, IMAGE_HEIGHT,
```

```
IMAGE_WIDTH, IMAGE_HEIGHT])

                # Convert the YOLO-format bounding box (center_x, center_y,
width, height) to
                # the OpenCV-format bounding box (x_min, y_min, width,
height) for drawing later
                center_x, center_y, box_width, box_height = box.astype('int')
                x_min = int(center_x - (box_width / 2))
                y_min = int(center_y - (box_height / 2))

                # Add the results to the output lists
                boxes.append([x_min, y_min, int(box_width), int(box_height)])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    # Return the filtered results
    return boxes, confidences, class ids

# Filter out all the predicted bounding boxes based on their probabilities
boxes, confidences, class ids = filter output probs(PROB THRESH,
layerOutputs)

confidences



# Apply non max suppression to all the predictions
indicies = cv.dnn.NMSBoxes(boxes, confidences, PROB_THRESH, IoU_THRESH,
top_k=5)
indicies

def draw_bounding_boxes(input_img: np.ndarray, boxes: list, class_ids: list,
confidences: list, labels: list, indicies: list) -> None:
    # Create a copy of the input image
    image_temp = input_img.copy()


    # Loop over each index in indicies
    for index in indicies:

        # Get the box coordinates and confidence for the current index
        x, y, w, h = boxes[index]
        confidence = confidences[index]

        # Set the color randomly
        color = (random.randint(0, 255), random.randint(0, 255),
random.randint(0, 255))


        # Draw a bounding box around the object
        cv.rectangle(image_temp, (x, y), (x+w, y+h), color=color,
thickness=2)

        # Create a text label for the object class and confidence
        text_label = '{}:{:.2f}'.format(labels[class_ids[index]], confidence)

        # Set the font face and scale
        font_face = cv.FONT_HERSHEY_DUPLEX
        font_scale = 1.5
        font_thickness = 2
```

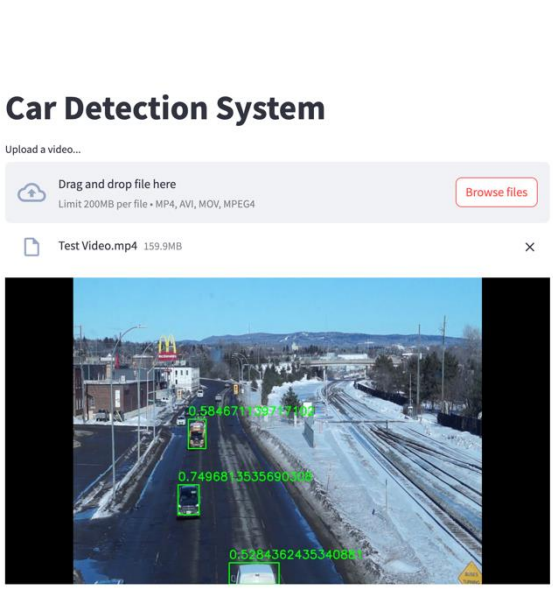
```
        # Draw the text label above the bounding box
        cv.putText(
            img=image_temp,
            text=text_label,
            org=(x+5, y-10),
            fontFace=font_face,
            fontScale=font_scale,
            color=color, # Set the font color to random
            thickness=font_thickness
        )

    # Display the image
    plt.imshow(image_temp)
    plt.axis('off')

draw bounding boxes(input img, boxes, class ids, confidences, labels,
indicies)

"""Let's add everything into one function."""

def pred bbs(image: np.ndarray, threshold probability: float=0.9,
iou threshold: float=0.5, labels: list=labels) -> None:

    # Convert image to blob
    blob = image_to_blob(image)

    # Make prediction
    net.setInput(blob)
    output_layers_names = net.getUnconnectedOutLayersNames()
    outputs = net.forward(output_layers_names)

    # Filter the inputs based on probability
    bboxes, confidences, class_ids =
filter_output_probs(threshold_probability = threshold_probability,
layer_outputs = outputs)

    # Apply Non Max Suppression
    indicies = cv.dnn.NMSBoxes(bboxes = bboxes, scores = confidences,
score_threshold = threshold_probability, nms_threshold = iou_threshold)

    # Draw the Bounding Box
    draw_bounding_boxes(input_img = image, boxes = bboxes, class_ids =
class_ids, confidences = confidences, labels = labels, indicies = indicies)

# Constant
N_IMAGES = 25

def plot_random_images(images: np.ndarray) -> None:
    # Plotting Configuration
    plt.figure(figsize=(30,20))

    # For n number of images
    for n in range(N_IMAGES):

        # Subplot
        plt.subplot(int(np.sqrt(N_IMAGES)), int(np.sqrt(N_IMAGES)), n+1)

        # Select Image Randomly
        image = images[np.random.randint(len(images))]
```

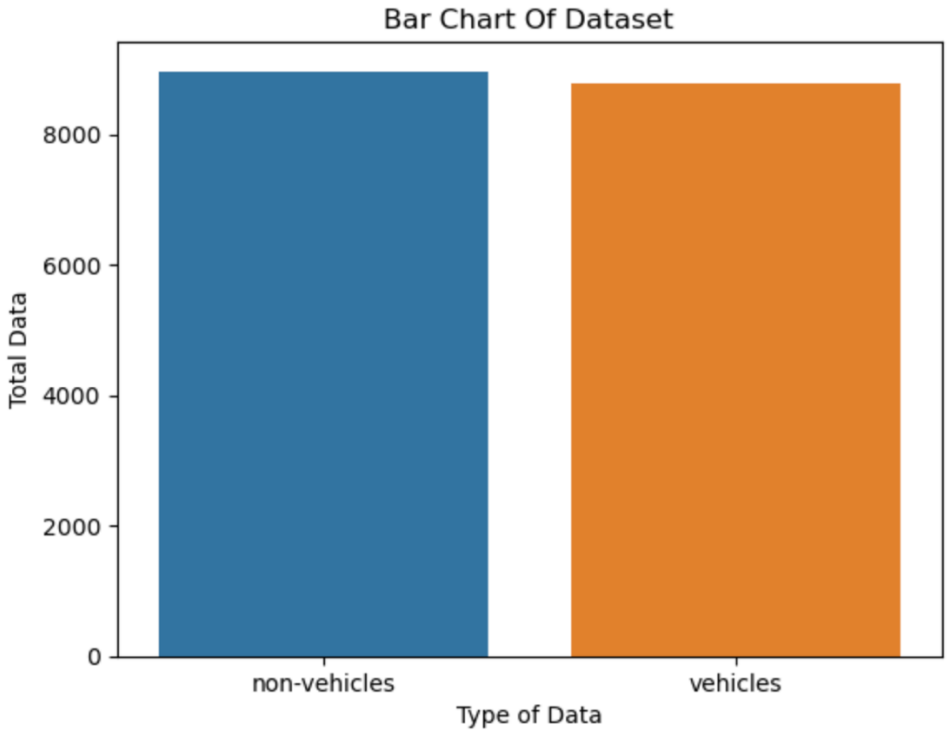
```
        # Detcted objects
        pred_bbs(image)

    # Show final plot
    plt.show()

# Plot images
plot_random_images(images)
app.py:

import streamlit as st
import cv2
import numpy as np
import tempfile


# Load YOLO model
def load_yolo():
    net =
cv2.dnn.readNetFromDarknet('/Users/ihortresnystkyi/Documents/PROJECT_STREAMLI
T_CARS_DETECTION/yolov3.cfg',
'/Users/ihortresnystkyi/Documents/PROJECT_STREAMLIT_CARS_DETECTION/yolov3.wei
ghts')
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i - 1] for i in
net.getUnconnectedOutLayers().flatten()]
    return net, output_layers

net, output_layers = load_yolo()

# Process a frame to detect cars
def process_frame(frame, net, output_layers):
    height, width, _ = frame.shape
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0),
swapRB=True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
    for i in indexes.flatten():
        x, y, w, h = boxes[i]
        label = str(confidences[i])
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(frame, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9,
```

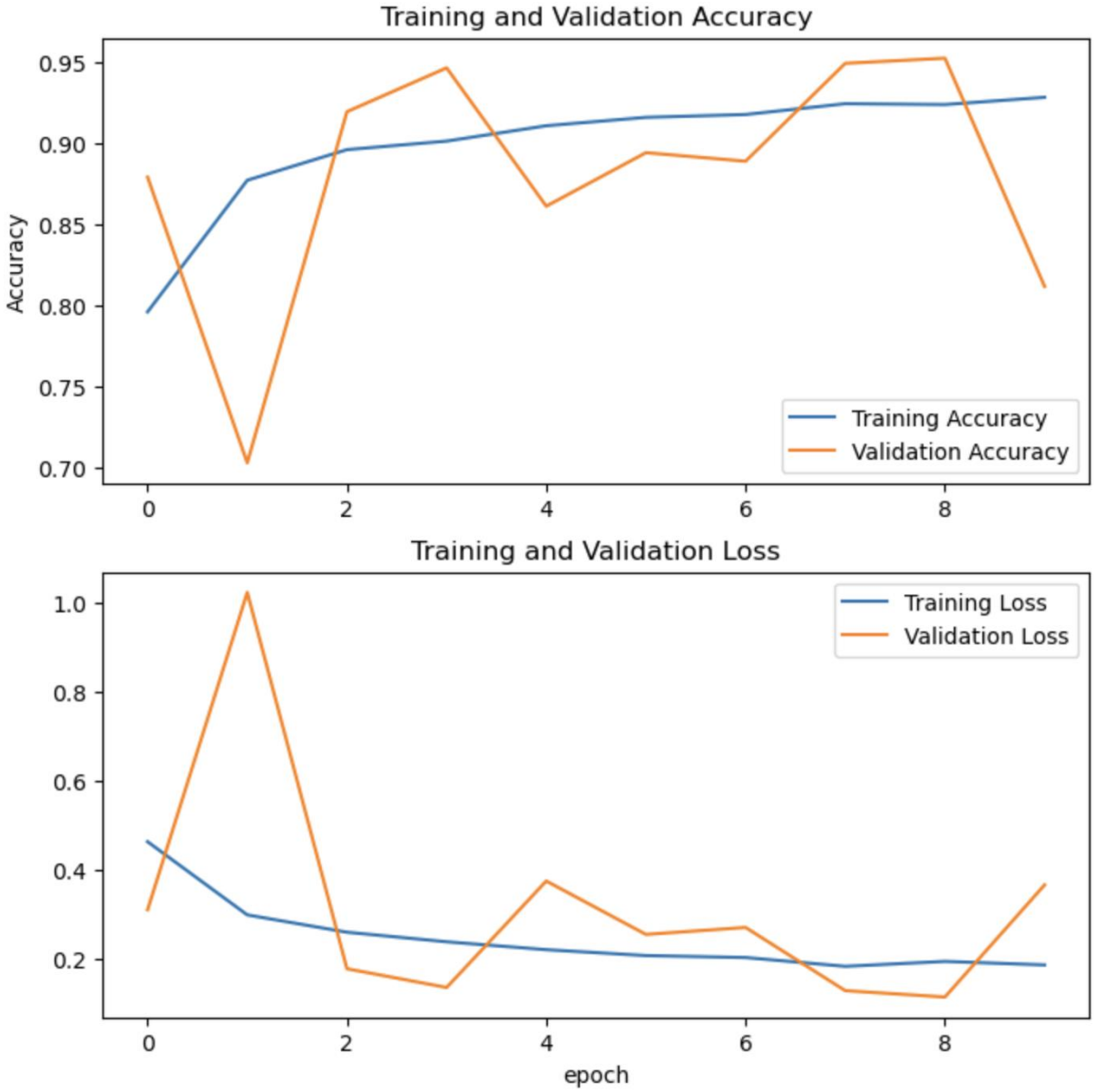
```
(0, 255, 0), 2)
    return frame

# Streamlit app
def main():
    st.title("Car Detection System")
    video_file = st.file_uploader("Upload a video...", type=["mp4", "avi",
"mov"])

    if video_file is not None:
        tfile = tempfile.NamedTemporaryFile(delete=False)
        tfile.write(video_file.read())
        cap = cv2.VideoCapture(tfile.name)
        stframe = st.empty()

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break
            frame = process_frame(frame, net, output_layers)
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            stframe.image(frame)

        cap.release()

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

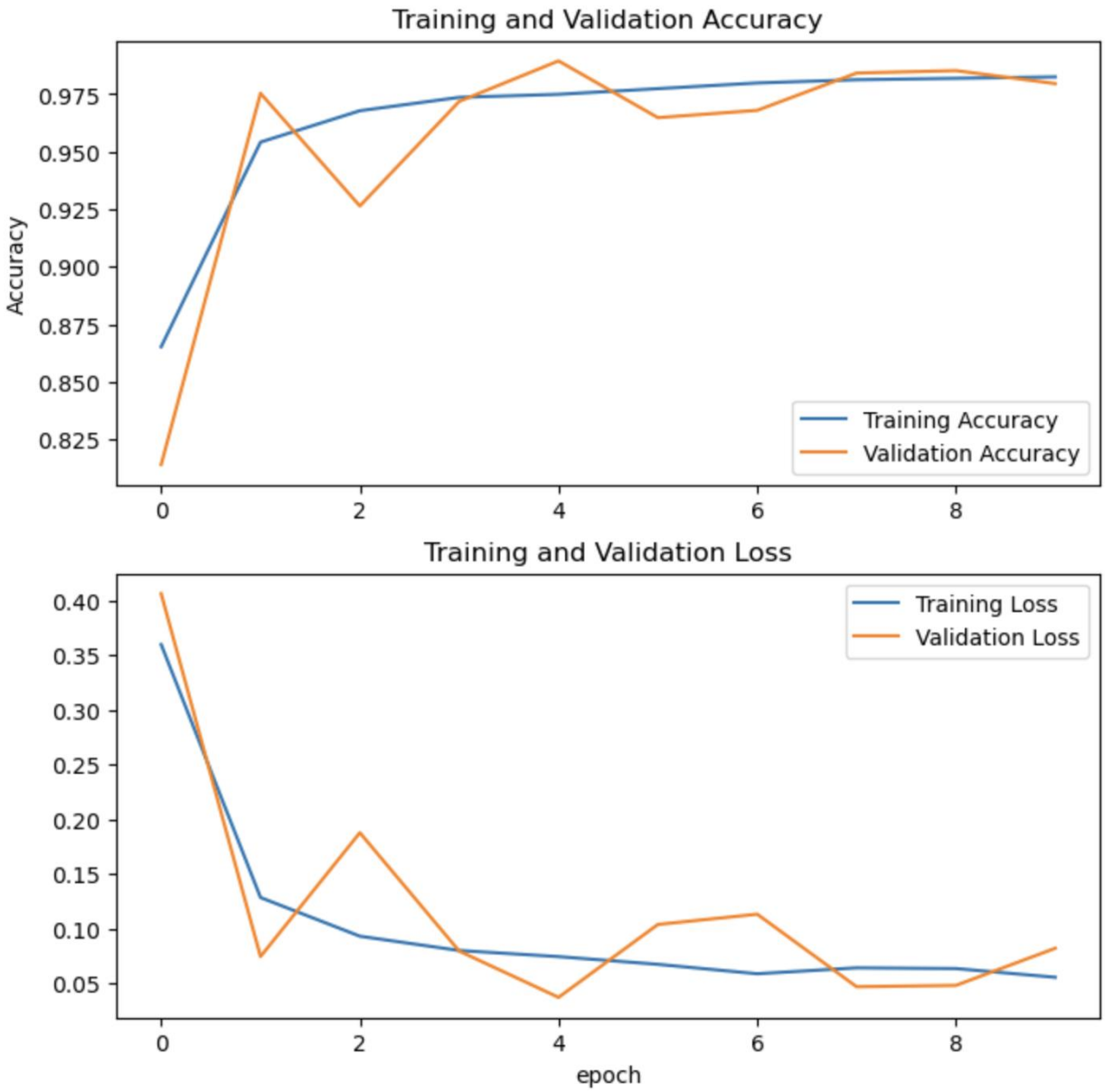
ДОДАТОК Б


Deploy
Car Detection System
Upload a video...
Drag and drop file here
Limit 200MB per file • MP4, AVI, MOV, MPEG4
Browse files


RUNNING...
Stop
Deploy
Car Detection System
Upload a video...
Drag and drop file here
Limit 200MB per file • MP4, AVI, MOV, MPEG4
Browse files
Test Video.mp4 159.9MB
0.584671
0.749681
0.528436


Bar Chart Of Dataset
Total Data
8000
6000
4000
2000
0
non-vehicles
vehicles
Type of Data


Training and Validation Accuracy
Accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0
2
4
6
8
Training Accuracy
Validation Accuracy
Training and Validation Loss
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
epoch
Training Loss
Validation Loss


Training and Validation Accuracy
Accuracy
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0
2
4
6
8
Training Accuracy
Validation Accuracy
Training and Validation Loss
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
2
4
6
8
epoch
Training Loss
Validation Loss

```
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 dense_5 (Dense)             (None, 2)                 8194

=================================================================
Total params: 134,268,738
Trainable params: 8,194
Non-trainable params: 134,260,544
_________________________________________________________________
```





truck:0.95

car:1.00
car:1.00
car:0.96

car:1.00
car:1.00


car:0.98
car:0.99


truck
car:0.95


car:1.00
car:1.00
car:0.96

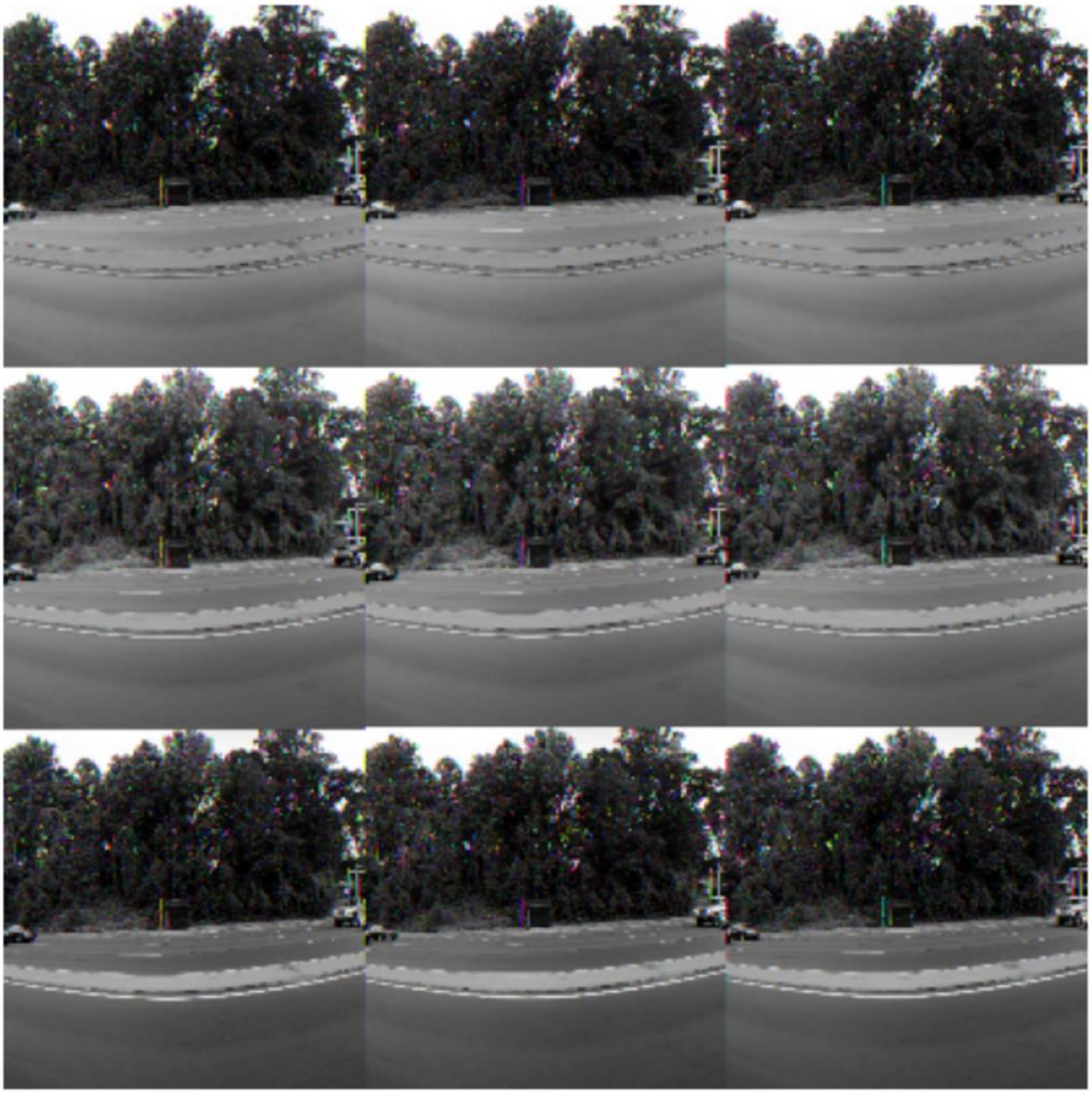

car:0.98


car:1.00


car:0.95


car:


car:0.99



```
import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'yolo-coco-data:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F253570%2F562374

KAGGLE_INPUT_PATH='/content/input'
KAGGLE_WORKING_PATH='/cintent/working'
KAGGLE_SYMLINK='colab'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
```