НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.852

До захисту допущено
В.о. зав. кафедри ШІ
_____ О.І. Чумаченко
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 122 «Комп'ютерні науки»

на тему: «Прогноз траєкторії учасників дорожнього руху з використанням кількох камер»

Виконав:
студент ІІ курсу, групи КІ-11мп
Загній Єгор Васильович _____

Керівник:
професор кафедри ММСА,
д.т.н., проф. Данилов В. Я. _____

Рецензент:
член-кореспондент НАН України,
директор НН ФТІ КПІ ім. Ігоря Сікорського,
д.т.н., проф. Новіков О. М. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ
2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 «Комп'ютерні науки »

ЗАТВЕРДЖУЮ
В.о. зав. кафедри
_____ О.І. Чумаченко
«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Загній Єгору Васильовичу

1. Тема дисертації: «Прогноз траєкторії учасників дорожнього руху з використанням кількох камер», науковий керівник роботи Данилов Валерій Якович, професор кафедри ММСА, д.т.н., проф. затверджено наказом по університету від «3» листопада 2022 р. № 4046-с.
2. Термін подання студентом дисертації 15.12.2022
3. Об'єкт дослідження: Задача прогнозу траєкторії учасників дорожнього руху.
4. Предмет дослідження: Алгоритми та моделі прогнозу траєкторії учасників дорожнього руху.
5. Перелік завдань, які потрібно зробити:
    1) здійснити огляд технічної літератури за темою роботи;
    2) дослідити актуальність обраної теми;

3) ознайомитись із існуючими методами та моделями прогнозу траєкторій

4) здійснити порівняльний аналіз наявних методів, виявити їх переваги та недоліки;

5) розробити та реалізувати систему, що використовує апарат нейрониих мереж, та вирішує задачу прогнозування траєкторій;

6) провести експеримент, що засвідчує працеспроможність запропонованої моделі, виконати аналіз результатів;

7) провести аналіз ринкових можливостей запуску стартап проекту;

8) розробити концептуальні висновки;

9) підготувати ілюстративний матеріал;

10) оформити пояснювальну записку.

6. Перелік ілюстративного матеріалу.

7. Дата видачі завдання: 31 січня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	01.08.2022 – 14.08.2022	Виконано
2.	Підготовка першого розділу.	14.08.2022 – 16.08.2022	Виконано
3.	Підготовка другого розділу.	16.08.2022 – 23.08.2022	Виконано
4.	Розробка програмного продукту.	01.10.2022 – 01.11.2022	Виконано
5.	Підготовка третього розділу	01.11.2022 – 10.11.2022	Виконано
6.	Підготовка частини стартап-проєкту	11.11.2022 – 13.11.2022	Виконано
9.	Концептуальні висновки. Перспективи розвитку отриманих рішень	13.11.2022 – 15.11.2022	Виконано
10.	Оформлення пояснювальної записки	15.11.2022 – 21.11.2022	Виконано

Студент Єгор ЗАГНІЙ

Керівник Валерій ДАНИЛОВ

РЕФЕРАТ

Магістерська дисертація: 114 с., 25 табл., 10 рис., 19 джерел, 1 додаток.

МАШИННЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ОБ'ЄКТІВ, ОЦІНКА ВІДСТАНЕЙ, ВІДСЛІДКОВУВАННЯ ОБ'ЄКТІВ, ПРОГНОЗ ТРАЄКТОРІЙ

Об’єктом дослідження є задача прогнозування траєкторій.

Предмет дослідження – алгоритми та моделі прогнозування траєкторій.

За останні 10 років людство досягло значного успіху у галузі штучного інтелекту. Це стало можливим завдяки значному зростанню обчислюваних потужностей комп’ютерів, завдяки чому з’явилася можливість тренувати великі нейронні мережі. Штучний інтелект вже є частиною звичного життя людей. Одним з напрямком штучного інтелекту є створення автопілотів для машин. Створення автопілоту неможливо без розв’язання задачі розпізнавання машин і пішоходів навколо машини, знаходження відстаней до них, відслідковування переміщення цих об’єктів, побудови карти з висоти пташиного польоту з розміщенням на ній цих об’єктів, та прогнозуванням траєкторій сусідніх машин.

Метою роботи є дослідження й порівняння різних методів, моделей, структур нейронних мереж для вирішення кожної з підзадач: розпізнавання об’єктів навколо машини, знаходження відстаней до них, побудови виду з висоти пташиного польоту навколо машини, відслідковування переміщення кожного об’єкта та прогнозування траєкторій об’єктів.

ABSTRACT

Master’s thesis: 114 p., 25 tab., 10 fig., 19 references, 1 appendix.

MACHINE LEARNING, COMPUTER VISION, OBJECT RECOGNITION, DISTANCE ESTIMATION, OBJECT TRACKING, TRAJECTORY PREDICTION

The object of research is the task of forecasting trajectories.

The subject of research is algorithms and models of trajectory forecasting.

Over the past 10 years, humanity has made significant progress in the field of artificial intelligence. This became possible due to the significant increase in computing power of computers, which made it possible to train large neural networks. Artificial intelligence is already a part of people's everyday life. One of the directions of artificial intelligence is the creation of autopilots for cars. Creating an autopilot is impossible without solving the problem of recognizing cars and pedestrians around the car, finding the distances to them, tracking the movement of these objects, building a map from a bird's eye view with the location of these objects on it, and predicting the trajectories of nearby cars.

The purpose of the work is to research and compare different methods, models, structures of neural networks for solving each of the sub-tasks: recognizing objects around the car, finding distances to them, building a bird's-eye view around the car, tracking the movement of each object and predicting trajectories objects.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9
1.1 Комп’ютерний зір 9
1.2 Нейронні мережі 9
1.2.1 Згорткові нейронні мережі 11
1.2.2 Рекуретні нейронні мережі 12
1.2.3 LSTM 13
1.3 Висновки 14
РОЗДІЛ 2 ПІДЗАДІ ПРОГНОЗУВАННЯ ТРАЄКТОРІЙ УЧАСНИКІВ ДОРОЖНЬОГО РУХУ 15
2.1 Розпізнавання об’єктів 15
2.1.1. CenterNet 15
2.2 Знаходження відстаней до об’єкта 18
2.3 Відслідковування об’єктів 19
2.3.1. CenterTrack 19
2.4 Прогнозування траєкторій 22
2.5 Висновки 26
РОЗДІЛ 3 ОПИС РЕАЛІЗАЦІЇ ТА ТРЕНУВАЛЬНИХ ДАНИХ 28
3.1 Реалізація 28
3.2 Датасети 29
3.3 Висновки 30
РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ 32
4.1 Опис ідеї проекту 33

4.2 Технологічний аудит ідеї проекту.....35
4.3 Аналіз ринкової стратегії проекту.....43
4.4 Розроблення маркетингової програми стартап-проекту.....47
4.5 Висновки.....51
ВИСНОВКИ.....52
ПЕРЕЛІК ПОСИЛАНЬ.....53
ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ.....54

ВСТУП

За останні 10 років людство досягло значного успіху у галузі штучного інтелекту. Це стало можливим завдяки значному зростанню обчислюваних потужностей комп’ютерів, завдяки чому з’явилася можливість тренувати великі нейронні мережі. Штучний інтелект вже є частиною звичного життя людей. Одним з напрямком штучного інтелекту є створення автопілоту для машин. Створення автопілоту неможливо без рішення задачі розпізнавання машин і пішоходів навколо машини, знаходження відстаней до них, відслідковування переміщення цих об’єктів, побудови карти з висоти пташиного польоту з розміщенням на ній цих об’єктів, та прогнозуванням траєкторій сусідніх машин.

З огляду на практичність цієї задачі та те що я працюю в цій галузі на роботі, я вирішив дослідити дану задачу, створити на навчити нейронні мережі для кожної з підзадач, та зробити демонстрацію їх роботи.

Метою роботи є дослідження й порівняння різних методів, моделей, структур нейронних мереж для вирішення кожної з підзадач: розпізнавання об’єктів навколо машини, знаходження відстаней до них, побудови виду з висоти пташиного польоту навколо машини, відслідковування переміщення кожного об’єкта та прогнозування траєкторій об’єктів.

Мова програмування, яка використовується в даній роботі: Python. Основні бібліотеки: Pytorch, Numpy, OpenCV.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Комп’ютерний зір

Комп’ютерний зір – це галузь коп’ютерних наук, яка вивчає, те як комп’ютери можуть отримати інформацію з зображень. Тобто знати не тільки колір пікселів зображення, а й те які об’єкти на ньому присутні, в якій частині зображення вони присутні. Задач для комп’ютерного зору багато: розпізнавання об’єктів, сегментація зображення, розпізнавання пози людини, ре ідентифікація обличчя людини… Галузь існує давно, спочатку задачі намагалися вирішувати різними алгоритмами аналізу зображень, дескрипторами, було написано багато наукових робіт, проте якийсь серйозні успіхи в галузі комп’ютерного зору з’явились тільки тоді, коли з’явилися комп’ютери з достойною обчислюваною потужністю, і в гру вступило машинне навчання й з’явилася можливість тренувати нейронні мережі. Особливим проривом в галузі комп’ютерного зору стала поява згорткових нейронних мереж.

1.2 Нейронні мережі

Нейронні мережі були натхненні реальними біологічними структурами нервових клітин нейронів, які складаються з клітин нейронів та синапсів, по яким клітини посилають електронні сигнали. Нейронні мережі є підмножиною машинного навчання та є основою алгоритмів глибокого навчання. Їх назва та структура навіяні людським мозком, імітуючи спосіб, яким біологічні нейрони передають сигнали один одному.

Штучні нейронні мережі складаються з вузлових шарів, що містять вхідний рівень, один або більше прихованих шарів і вихідний рівень. Кожен

вузол, або штучний нейрон, з’єднується з іншим і має відповідну вагу та поріг. Якщо вихід будь-якого окремого вузла перевищує вказане порогове значення, цей вузол активується, надсилаючи дані на наступний рівень мережі. В іншому випадку дані не передаються на наступний рівень мережі.

Нейронні мережі покладаються на навчальні дані, щоб навчатися та підвищувати свою точність з часом. Однак, коли ці алгоритми навчання точно налаштовані на точність, вони стають потужними інструментами в інформатиці та штучному інтелекті, що дозволяє класифікувати та кластеризувати дані з високою швидкістю. Завдання з розпізнавання мовлення або розпізнавання зображень можуть тривати хвилини чи години порівняно з ручною ідентифікацією експертів-людей. Однією з найвідоміших нейронних мереж є пошуковий алгоритм Google.

Кожен окремий вузол як про власну модель лінійної регресії, що складається з вхідних даних, вагових коефіцієнтів, зміщення (або порогу) і вихідних даних. Формула виглядатиме приблизно так:

Первинною метою підходу ШНМ було розв'язання задач таким же способом, як це робив би людський мозок. З часом увага зосередилася на відповідності певним розумовим здібностям, ведучи до відхилень від біології. ШНМ використовували в ряді різноманітних задач, включно з комп'ютерним баченням, розпізнаванням мовлення, машинним перекладом, соціально-мережевим фільтруванням, грою в настільні та відеоігри, та медичним діагностуванням.

Після визначення вхідного рівня призначаються ваги. Ці вагові коефіцієнти допомагають визначити важливість будь-якої заданої змінної, причому більші з них мають більший внесок у результат порівняно з іншими вхідними параметрами. Потім усі вхідні дані множаться на відповідні ваги, а потім підсумовуються. Після цього вихідні дані проходять через функцію активації, яка визначає вихідні дані. Якщо цей вихід перевищує заданий поріг, він «спрацьовує» (або активує) вузол, передаючи дані на наступний рівень у

мережі. Це призводить до того, що вихід одного вузла стає входом наступного вузла. Цей процес передачі даних з одного рівня на наступний визначає цю нейронну мережу як мережу прямого зв’язку.

1.2.1 Згорткові нейронні мережі

У глибокому навчанні згорточна нейронна мережа (CNN/ConvNet) — це клас глибоких нейронних мереж, які найчастіше застосовуються для аналізу візуальних зображень. Тепер, коли ми думаємо про нейронну мережу, ми думаємо про множення матриці, але це не стосується ConvNet. Він використовує спеціальну техніку під назвою Convolution. Тепер у математиці згортка — це математична операція над двома функціями, яка створює третю функцію, яка виражає, як форма однієї змінюється іншою.

Згорткові нейронні мережі складаються з кількох шарів штучних нейронів. Штучні нейрони, груба імітація своїх біологічних аналогів, є математичними функціями, які обчислюють зважену суму кількох вхідних даних і виводять значення активації. Коли ви вводите зображення в ConvNet, кожен рівень генерує кілька функцій активації, які передаються на наступний рівень.

Перший шар зазвичай виділяє основні елементи, такі як горизонтальні або діагональні краї. Цей результат передається на наступний рівень, який виявляє більш складні елементи, такі як кути або комбіновані краї. Коли ми просуваємося глибше в мережу, він може ідентифікувати навіть більш складні характеристики, такі як об’єкти, обличчя тощо.

На основі карти активації останнього шару згортки класифікаційний рівень виводить набір оцінок достовірності (значення від 0 до 1), які визначають, наскільки ймовірно, що зображення належить до «класу». Наприклад, якщо у вас є ConvNet, який виявляє котів, собак і коней, результат

останнього шару – це можливість того, що вхідне зображення містить будь-яку з цих тварин.

Подібно до згорткового рівня, рівень об’єднання відповідає за зменшення просторового розміру згорнутого об’єкта. Це робиться для зменшення обчислювальної потужності, необхідної для обробки даних, шляхом зменшення розмірів. Існує два типи об’єднання: середнє об’єднання та максимальне об’єднання. У мене був досвід роботи лише з MaxPooling, поки я не зіткнувся з жодними труднощами.

Отже, що ми робимо в Max Pooling, ми знаходимо максимальне значення пікселя з частини зображення, охопленої ядром. Max Pooling також працює як шумопоглинач. Він повністю відкидає шумні активації, а також виконує усунення шумів разом із зменшенням розмірності.

З іншого боку, Average Pooling повертає середнє значення всіх значень із частини зображення, охопленої ядром. Average Pooling просто виконує зменшення розмірності як механізм придушення шуму. Отже, ми можемо сказати, що Max Pooling працює набагато краще, ніж Average Pooling.

1.2.2 Рекуретні нейронні мережі

Рекурентні нейронні мережі — це клас штучних нейронних мереж, які можуть працьювати з даними різної довжини (текст, аудіо, відео). Особливість цих мереж полягає втому, що я к вхідні дані вони приймають не тільки самі вхідні дані, а й результат минулої ітерації. Це створює внутрішній стан мережі, що дозволяє їй проявляти динамічну поведінку в часі. На відміну від нейронних мереж прямого поширення, РНМ можуть використовувати свою внутрішню пам'ять для обробки довільних послідовностей входів. Це робить їх застосовними до таких задач, як розпізнавання несегментованого неперервного рукописного тексту та розпізнавання мовлення.

1.2.3 LSTM

Довга короткочасна пам’ять (LSTM) — це штучна нейронна мережа, яка використовується в галузі штучного інтелекту та глибокого навчання. На відміну від стандартних нейронних мереж прямого зв’язку, LSTM має зворотні зв’язки. Така рекурентна нейронна мережа (RNN) може обробляти не лише окремі точки даних (наприклад, зображення), але й цілі послідовності даних (наприклад, мову чи відео). Наприклад, LSTM можна застосувати до таких завдань, як несегментоване розпізнавання пов’язаного рукописного тексту, розпізнавання мови, машинний переклад, керування роботами, відеоігри, та охорона здоров’я.

Назва LSTM відноситься до аналогії, що стандартний RNN має як «довготривалу», так і «короткочасну пам’ять». Вага зв’язку та зміщення в мережі змінюються один раз за епізод тренування, аналогічно тому, як фізіологічні зміни синаптичної сили зберігають довгострокові спогади; патерни активації в мережі змінюються один раз за часовий крок, аналогічно тому, як миттєва зміна схем електричного запалювання в мозку зберігає короткочасні спогади. Архітектура LSTM має на меті забезпечити короткочасну пам’ять для RNN, яка може тривати тисячі кроків у часі, отже, «довгу короткочасну пам’ять».

Звичайний блок LSTM складається з комірки, вхідного вентиля, вихідного вентиля і пропускного вентиля. Комірка запам'ятовує значення протягом довільних інтервалів часу, а три ворота регулюють потік інформації в комірку та з неї.

Мережі LSTM добре підходять для класифікації, обробки та прогнозування на основі даних часових рядів, оскільки між важливими подіями в часовому ряді можуть бути затримки невідомої тривалості. LSTM

були розроблені для вирішення проблеми зникнення градієнта, яка може виникнути під час навчання традиційних RNN. Відносна нечутливість до довжини проміжків є перевагою LSTM перед RNN, прихованими моделями Маркова та іншими методами навчання послідовності в численних додатках.

1.3 Висновки

У даному розділі були дані визначення поняттям «комп'ютерний зір», «нейронні мережі», «згорткові нейронні мережі», «рекуренті нейронні мережі», «LSTM» .Задачу з прогнозування траєкторії учасників дорожнього руху було розкладено на декілька підзадач: розпізнавання об'єктів, знаходження відстаней до них, відслідковування об'єктів у відеопотоці, нанесення об'єктів на карту навколо машини, прогнозування траєкторій об'єктів. По кожній з задач проаналізовано літературу, розглянуто декілька методів рішення кожної з підзадач.

В розробленому рішенні задачі з розпізнавання об'єктів, оцінки відстаней та відслідковування об'єктів поєднано в один етап, тобто вирішується однією багатозадачною нейронною мережею, що дозволяє скоротити час робити, це стало можливим завдяки використанню центернетівського підходу. Також розроблені моделі працюють з відеопотоком а не з окремими кадрами що дозволяє трохи збільшити точність моделі. В роботі було продемонстровано і візуалізовано результати роботи натренованих моделей.

Розроблені моделі можна використовувати на бортових комп'ютерах автомобілях задля збору інформації щоб допомагати водію з донесенням ситуації що складається навколо машини.

РОЗДІЛ 2 ПІДЗАДІ ПРОГНОЗУВАННЯ ТРАЄКТОРІЙ УЧАСНИКІВ ДОРОЖНЬОГО РУХУ

2.1 Розпізнавання об’єктів

Задача розпізнавання об’єктів полягає у тому щоб знайти на зображенні об’єкти певного класу. Зазвичай кожен об’єкт буде позначатися прямокутником з координатами його розміщення на зображенні, а також ід класу об’єкта. Усі адекватні методи розпізнавання об’єктів використовують згорткові нейронні мережі. Отже по черзі розберемо кожен з методів.

2.1.1. CenterNet

Суть центернетівської архітектури полягає в тому, що нейронна мережа повертає для кожного класу об'єкта матрицю в якій вказані ймовірності того що в цій частині зображення є даний об’єкт. Всі локальні максимуми які перевищують певне значення й будуть центрами цих об’єктів. В додаткових матрицях буде ширина й висота об’єктів.

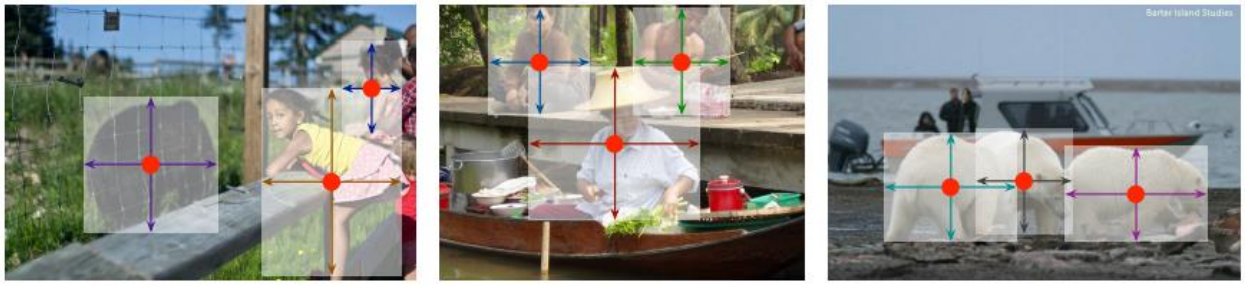

Рисунок 2.1 – Об’єкти в CenterNet

Нехай $I \in R^{W \times H \times 3}$ буде вхідним зображенням ширини W і висоти H. Наша мета – створити теплову карту ключових точок $\tilde{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$, де R – це у скільки разів вихідна матриця менше за вхідне зображення, а C – кількість типів об’єктів. Зазвичай R = 4. Прогноз $\hat{Y}_{x,y,c} = 1$ відповідає виявленому

об’єкту, тоді як $\hat{Y}_{x,y,c} = 0$ є фоном. Використовується повністю згорткові мережі, щоб передбачити $\hat{Y}$ на основі зображення I.

Для кожного об’єкта $p \in R^2$ класу c ми обчислюємо еквівалент низької роздільної здатності $\tilde{p} = \left\lfloor \frac{p}{R} \right\rfloor$. Потім ми наносимо всі основні ключові точки істинності на теплову карту $Y \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ за допомогою ядра Гауса

$$Y_{xyc} = exp\left(-\frac{(x - \hat{p}_x)^2 + (y - \hat{p}_y)^2}{2\sigma_p^2}\right) \tag{2.1}$$

де σ_p – стандартне відхилення, адаптоване до розміру об’єкта. Якщо два гаусівці одного класу перекриваються, ми беремо поелементний максимум. Метою навчання є попіксельна логістична регресія зі зменшенням штрафу з фокальними втратами:

$$L_k = \frac{-1}{N} \sum_{xyz} \begin{cases} (1 - \hat{Y}_{xyc})^a \log(\hat{Y}_{xyc}) & \text{якщо } Y_{xyc} = 1 \\ (1 - Y_{xyc})^\beta (\hat{Y}_{xyc})^a \log(1 - \hat{Y}_{xyc}) & \text{інакше} \end{cases} \tag{2.2}$$

де α і β — гіперпараметри фокальних втрат, а N — кількість ключових точок на зображенні I. Нормалізація за N вибирається так, щоб нормалізувати всі позитивні випадки фокусних втрат до 1. Ми використовуємо α = 2 і β = 4 у всіх наших дослідах

Щоб відновити помилку дискретизації, викликану зменшенням входу відносно виходу, ми додатково прогнозуємо локальне зміщення $\hat{O} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times C}$ для кожної центральної точки. Усі класи c мають однакове передбачення зсуву. Зміщення тренується з функцією втратою L1

$$L_{off} = \frac{1}{N} \sum_p \left| \hat{O}_{\hat{p}} - \left(\frac{\hat{p}}{R} - p \right) \right| \tag{2.3}$$

Це діє лише в ключових точках p˜, усі інші місця ігноруються.

Отже нехай $({x_1}^{(k)}, {y_1}^{(k)}, {x_2}^{(k)}, {y_2}^{(k)})$ це прямокутник який позначає розміщення об'єкта к-тої категорії на зображенні. Його центр знаходиться в $pk = \left(\frac{x_1^{(k)}+x_2^{(k)}}{2}, \frac{y_1^{(k)}+y_2^{(k)}}{2}\right)$ Використовуємо наш оцінювач ключових точок Yˆ, щоб передбачити всі центральні точки. Крім того, ми регресуємо до розміру об'єкта $s_k = (x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - y_1^{(k)})$ для кожного k. Щоб обмежити обчислювальне навантаження, Використовується єдиний прогноз розміру $S\hat{} \in \mathbb{R}^{\frac{W}{R}\times\frac{H}{R}\times C}$ для всіх категорій об'єктів. Використовується функцію втрату L1 у центральній точці.

$$L_{size} = \frac{1}{N}\sum_{k=1}^{N}\left|\hat{S}_{p_k} - s_k\right| \tag{2.3}$$

Ми не нормалізуємо масштаб і безпосередньо використовуємо необроблені координати пікселів. Натомість масштабується втрати за допомогою постійного λ_{size}. Загальна функція втрат навчання така

$$L_{det} = L_k + \lambda_{size}L_{size} + \lambda_{off}L_{off} \tag{2.4}$$

Встановлюється $\lambda_{size} = 0{,}1$ і $\lambda_{off} = 1$ у всіх експериментах. Використовуємо одну і ту ж саму нейронну згорткову мережу для прогнозування ключових точок Yˆ, зміщення Oˆ та розміру Sˆ. Мережа передбачає загальну кількість виходів C + 4 у кожному місці. Усі виходи спільно використовують загальну повністю згортову магістральну мережу. Для кожної модальності характеристики хребта потім пропускаються через окрему згортку 3 × 3, ReLU та іншу згортку 1 × 1. На рисунку 1.2 показано огляд виходу нейронної мережі.

Спочатку знаходимо локальні максимуми на тепловій карті для окремо кожної категорі. Виявляємо всі результати, значення яких більше або дорівнює його 8 зв’язним сусідам, і зберігаємо 100 найбільших максимумів. Нехай $\hat{P}_c$ — набір із n виявлених центральних точок $\hat{P} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{n}$ класу c. Розташування кожної ключової точки задається цілими координатами (x_i, y_i). Ми використовуємо значення ключових точок $\hat{Y}_{x_i\, y_i\, c}$ як міру впевненості виявлення та створюємо прямокутник який позначає розташування об'єкта в місці $(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i/2, \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2, \hat{x}_i + \delta\hat{x}_i + \hat{w}_i/2, \hat{y}_i + \delta\hat{y}_i + \hat{h}_i)$ де $(\delta\hat{x}_i, \delta\hat{y}_i) = \hat{O}_{\hat{x}_i, \hat{y}_i}$ є прогнозом зміщення, а $(\hat{w}_i, \hat{h}_i) = \hat{S}_{\hat{x}_i, \hat{y}_i}$ є прогнозом розміру. Усі результати створюються безпосередньо з оцінки ключових точок без необхідності придушення немаксимумів (NMS) на основі IoU або іншої постобробки. Вилучення пікових ключових точок служить достатньою альтернативою NMS і може бути ефективно реалізовано на пристрої за допомогою операції об’єднання 3 × 3 максимум.

Рисунок 2.2 – Вихід CenterNet

Тому постає задача переведення індексу товару, або користувача у стиснутий числовий вектор, яким краще оперувати моделям глибинного навчання.

2.2 Знаходження відстаней до об’єкта

Щоб знайти відстань до об’єктів можна спробувати декілька варіантів. Наприклад знаючи середню висоту машини, можна знаходити відстань за висотою машин, а бо якщо ми знаємо розміщення камери можна пробувати знаходити відстань за розміщенням об’єкта у кадрі. Проте такі методи можуть видавати значні похибки, тому якщо є датасет в якому розмічені відстані до об’єктів можна просто додати їх до виходу CenterNet ще один канал з відстанями до об’єктів, що буде працювати аналогічно визначенню розміру об’єкта.

2.3 Відслідковування об’єктів

Задача відслідковування об’єктів полягає в тому щоб у відеопотоці зрозуміти що об’єкт, який присутній на декількох кадрах є одним і тим самим об’єктом. Відслідкувати переміщення об’єкта у відео.

Для гарантування якомога більшого стиснення даних при отриманні вектору $z = E_\theta(x)$ пропонується скористатися підходом, запропонованим авторами алгоритму неявних ранг-мінімізуючих автокодувальників (implicit rank-minimizing autoencoders, IRMAE) [19].

2.3.1. CenterTrack

А саме: пропонується ввести l випадкових матриць $W_1, W_2, \dots, W_l,\ W_i \in \mathbb{R}^{m\times m}$. Тобто (2.4) перетворюється на:

$$\min_\theta -x^T \log\left(D_\theta\left(\prod_i W_i * E_\theta(x)\right)\right) \tag{2.5}$$

CenterTrack має підхід до відслідковування об’єктів з локальної точки зору. Коли об’єкт залишає кадр або закривається та з’являється знову, йому призначається новий ідентифікатор. Таким чином, розглядається відстеження як розповсюдження ідентифікаторів об’єктів між послідовними кадрами без відновлення асоціацій через часові проміжки.

У момент часу t ми отримуємо зображення поточного кадру $I(t) \in \mathbb{R}^{W\times H\times 3}$ і попереднього кадру $I(t-1) \in \mathbb{R}^{W\times H\times 3}$, а також відстежуваних об’єктів у попередньому кадрі $T^{(t-1)} = \{b_0^{(t-1)}, b_1^{(t-1)}, \dots\}_i$. Кожен об’єкт b = (p,s,w,id) описується своїм центральним розташуванням $p \in \mathbb{R}^2$, розміром $s \in \mathbb{R}^2$, впевненістю виявлення $w \in [0,1]$ та унікальним ідентифікатором $id \in I$. Наша мета — виявити і відстежувати об'єкти $T^{(t)} = \{b_0^{(t)}, b_1^{(t)}, \dots\}_i$ в поточному кадрі t, і призначати об'єкти, які з'являються в обох кадрах, ідентифікатором.

Тут є дві основні проблеми. По-перше, це пошук усіх об’єктів у кожному кадрі, включаючи закриті. Друга проблема полягає в асоціації цих об’єктів у часі. Ми звертаємося до обох через єдину глибоку згорткову нейронну мережу.

Щоб пов’язати виявлення в часі, CenterTrack передбачає двовимірне зміщення як два додаткових вихідних канали $\widehat{D}^{(t)} \in \mathbb{R}^{\frac{W}{R}*\frac{H}{R}*2}$. Для кожного виявленого об’єкта в місці розташування $\hat{p}^{(t)}$ зміщення фіксує різницю в місці розташування $\hat{d}^{(t)} = \widehat{D}^{(t)}_{\hat{p}^{(t)}}$ об’єкта в поточному кадрі $\hat{p}^{(t)}$ і попередньому кадрі $\hat{p}^{(t-1)}$: $\hat{d}^{(t)} = \hat{p}^{(t)} - \hat{p}^{(t-1)}$. Ми вивчаємо це зміщення, використовуючи ту саму ціль регресії, що й уточнення розміру або розташування:

$$L_{off} = \frac{1}{N}\sum_{i=1}^{N}\left|\widehat{D}^{(t)}_{\hat{p}_i^{(t)}} - (p_i^{(t-1)} - p_i^{(t)})\right| \qquad (2.6)$$

де $p_i^{(t-1)}$ і $p_i^{(t)}$ – відстежувані наземні об’єкти. На рисунку 2 показано приклад цього прогнозу зміщення.

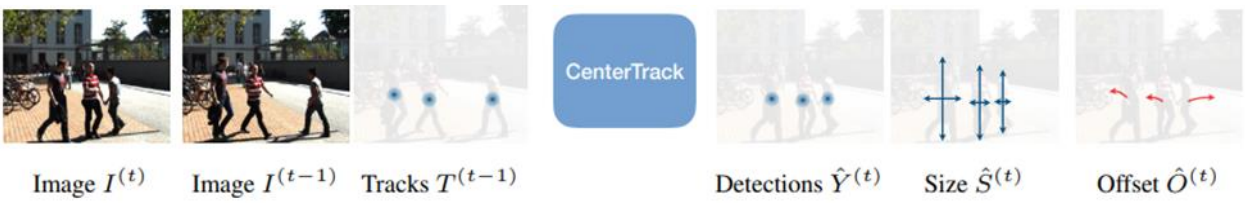

Рисунок 2.3 – Вхід і вихід CenterTrack

З достатньо хорошим прогнозуванням зсуву простий жадібний алгоритм зіставлення може асоціювати об'єкти в часі. Для кожного виявлення в позиції $p\hat{}$ ми асоціюємо його з найближчим попереднім виявленням у позиції $p\hat{} - D\hat{}_{p\hat{}}$, у порядку спадання достовірності $w\hat{}$. Якщо в радіусі к немає незрівнянного попереднього виявлення, ми створюємо новий треклет. Ми визначаємо к як середнє геометричне ширини та висоти прямокутника, що позначає об'єкт для кожного треку. Простота цього жадібного алгоритму зіставлення знову підкреслює переваги відстеження об'єктів як точок. Простого передбачення переміщення достатньо, щоб зв'язати об'єкти в часі. Немає потреби у складній метриці відстані чи зіставленні графіків.

CenterTrack — це, перш за все, детектор об'єктів. Архітектурні зміни від CenterNet до CenterTrack незначні: чотири додаткові вхідні канали та два вихідних канали. Це дозволяє нам точно налаштувати CenterTrack безпосередньо з попередньо підготовленого детектора CenterNet. Ми копіюємо всі ваги, пов'язані з поточним конвеєром виявлення. Усі ваги, що відповідають додатковим входам або виходам, ініціалізуються випадковим чином. Ми дотримуємося протоколу навчання CenterNet і тренуємо всі передбачення як багатозадачне навчання. Ми використовуємо ту саму навчальну ціль із додаванням регресії зміщення L_{off}.

Основна проблема під час навчання CenterTrack полягає у створенні реалістичної теплової карти зміщення об'єктів відносно попереднього кадру $H^{(t-1)}$. Під час висновку ця теплова карта треклетів може містити довільну кількість відсутніх треклетів, неправильно локалізованих об'єктів або навіть помилкових спрацьовувань. Ці помилки відсутні в треклетів $\{p_0^{(t-1)}, p_{i1}^{(t-1)}, \dots\}$,

наданих під час навчання. Натомість ми симулюємо цю помилку під час тестування під час навчання. Зокрема, ми моделюємо три типи помилок. По-перше, ми трішки зміщуємо кожен треклет $p^{(t-1)}$ від попереднього кадру, додаючи гаусівський шум до кожного центру. Тобто ми виводимо $p'_i = (x_i + r \times \lambda_{jt} \times w_i, y_i + r \times \lambda_{jt} \times h_i)$, де r вибирається з розподілу Гауса. Ми використовуємо $\lambda_{jt} = 0{,}05$ у всіх експериментах. По-друге, ми випадково додаємо хибні об'єкти поблизу місць розташування правильних об'єктів, відтворюючи помилковий шумовий пік p'_i з імовірністю λ_{fp}. По-третє, ми моделюємо хибні негативи шляхом випадкового видалення об'єктів з імовірністю λ_{fn}. Цих трьох доповнень достатньо для навчання надійного детектора об'єктів з умовами відстеження.

На практиці $I^{(t-1)}$ не обов'язково має бути кадром, що безпосередньо передує з часу t−1. Це може бути інший кадр із того самого відеоряду. Випадково відбираємо кадри поблизу t, щоб уникнути перепідгонки до частоти кадрів. Зокрема, Виконуємо вибірку з усіх кадрів k, де $|k - t| < M_f$, де $M_f = 3$ — гіперпараметр.

Без розмічених відеоданих CenterTrack не має доступу до попереднього кадру $I^{(t-1)}$ або відслідковуваних об'єктів $\{p_0^{(t-1)}, p_1^{(t-1)}, \dots\}$. Однак ми можемо симулювати відстеження на стандартних датасетах, маючи окремі зображення $I^{(t)}$ і об'єкти $\{p_0^{(t)}, p_1^{(t)}, \dots\}$. Ідея проста: ми моделюємо попередній кадр шляхом випадкового масштабування та зміщення поточного кадру.

2.4 Прогнозування траєкторій

Розуміння ситуацій водіння є важливою характеристикою автономного транспортного засобу для прийняття рішень високого рівня, планування траєкторій і здійснення контролю. Зокрема, для безпечного та комфортного

водіння автономний транспортний засіб має бути здатним виявляти не лише поточне середовище, але й передбачати майбутнє середовище. Транспортний засіб може бути не в змозі безпечно рухатися по траєкторії, щоб уникнути об’єктів, виявлених у реальному часі, через його інерцію та неголономні характеристики. Крім того, раптові зміни швидкості та прискорення автомобіля можуть порушити рух і викликати дискомфорт у пасажирів. Тому прогнозування траєкторії руху транспортних засобів, таких як автомобілі, вантажівки та велосипеди, є важливим для автономного транспортного засобу. В останні роки в численних дослідженнях намагалися використати прогнозовані траєкторії для прийняття рішень на високому рівні і планування траєкторії автономних транспортних засобів.

Прогнозування траєкторій транспортних засобів є актуальною проблемою з наступних причин. По-перше, подальший маневр транспортного засобу має певну невизначеність, оскільки місце призначення водія невідоме. Як показано на малюнку 1a, навіть якщо транспортний засіб слідує по прямій траєкторії, він може залишатися в смузі або змінити смугу залежно від місця призначення водія. Таким чином, алгоритм прогнозування траєкторії повинен бути здатний передбачати всі можливі маневри. По-друге, рухи транспортних засобів взаємозалежні; таким чином, рухи транспортного засобу та оточуючих транспортних засобів впливають один на одного. Зокрема, кожен транспортний засіб зазнає різного впливу залежно від маневрів транспортних засобів, які його оточують. Наприклад, як показано на рисунку 2.4, коли два транспортні засоби, що рухаються поруч, рухаються по прямій траєкторії, кожен транспортний засіб має декілька маневрів (залишатися на смузі руху або змінитися на ліву чи праву смугу). Якщо обидва транспортні засоби

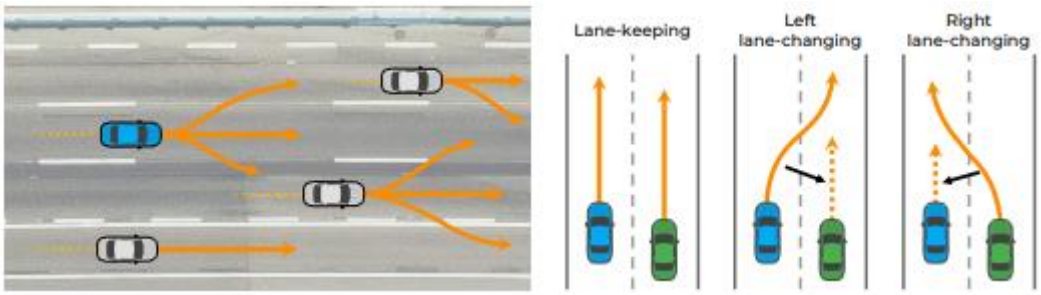

Рисунок 2.4 –Демонстрація необхідності взаємодії між транспортними засобами

Щоб вирішити ці проблеми, нещодавно були зроблені різні дослідження, які намагалися передбачити траєкторії, використовуючи підходи, засновані на глибокому навчанні, здатні вивчати дані; отже, немає необхідності математично виражати взаємодію маневру. Для ефективного розгляду взаємодій попередні дослідження намагалися змоделювати їх, представляючи інформацію про транспортні засоби у вигляді зображень у формі сітки або графічних структур, а не використовувати необроблені дані датчиків. Взаємодія представленої інформації розпізнається за допомогою згорткової нейронної мережі (CNN) або графової нейронної мережі (GNN). Крім того, щоб вирішити мультимодальність маневрування, певні підходи передбачають траєкторію відповідно до маневрів та їх імовірностей.

2.4.1. Прогнозування траєкторії транспортного засобу з використанням ієрархічної графової нейронної мережі для врахування взаємодії між мультимодальними маневрами

Основна мета полягає в тому, щоб зробити точний прогноз траєкторії з урахуванням взаємодії між мультимодальними маневрами в дуже інтерактивних ситуаціях. Для досягнення цієї мети пропонується ієрархічна GNN для прогнозування траєкторії. Як показано на рисунку 2.5, запропонований метод складається з двох етапів: мультимодальна мережа

прогнозування траєкторії на основі маневру та мережа прогнозування траєкторії з урахуванням взаємодії.

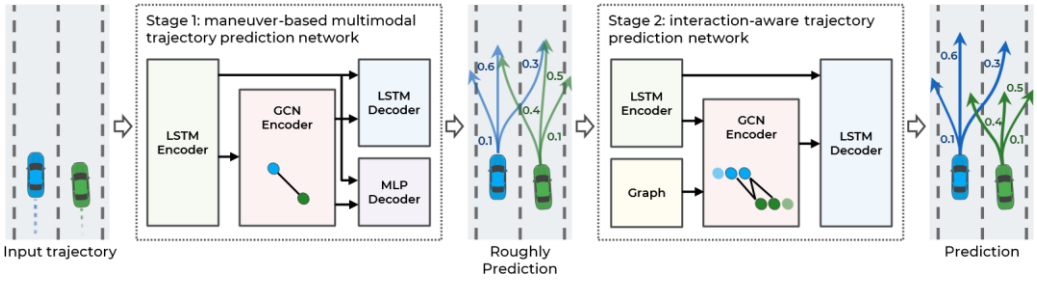

Рисунок 2.5 –Загальна архітектура системи запропонованої ієрархічної моделі прогнозування траєкторії.

Перший етап приблизно прогнозує кілька можливих траєкторій відповідно до мультимодальних маневрів навколишніх транспортних засобів і прогнозує ймовірність кожного маневру. Коли вводиться кожна спостережувана траєкторія, динамічні характеристики транспортного засобу витягуються з кодера LSTM. LSTM може кодувати послідовні дані, такі як спостережувані траєкторії, але він має труднощі з кодуванням просторових відносин, включаючи взаємодію транспортних засобів. Відповідно, щоб зафіксувати особливості взаємодії між спостережуваними траєкторіями навколишніх транспортних засобів, введено кодер графової згорткової мережі (GCN). Вузли графіка представлені як динамічні особливості кодера LSTM, а краю графіка як просторова відстань поточного положення автомобіля. З динамічними характеристиками та характеристиками взаємодії в якості вхідних даних траєкторії прогнозуються відповідно до маневрів за допомогою декодера LSTM та ймовірності маневрів за допомогою декодера багатошарового персептрона (MLP).

На другому етапі прогнозуються траєкторії, враховуючи взаємодію між мультимодальними маневрами транспортних засобів. На цьому етапі, використовуючи мультимодальні траєкторії, передбачені на першому етапі, як вхідні дані, динамічні характеристики кожної траєкторії кодуються за допомогою кодера LSTM. Потім, щоб розглянути взаємодію між

прогнозованими траєкторіями, використовується кодер GCN. На відміну від першого етапу, ребра графа будуються шляхом визначення зіткнень між прогнозованими траєкторіями для ефективного розгляду взаємозв’язків. Нарешті, витягнувши динамічні характеристики та характеристики взаємодії на другому етапі як вхідні дані, траєкторії прогнозуються за допомогою декодера LSTM. Запропонований метод розроблений ієрархічно для розпізнавання взаємодії між мультимодальними маневрами. Отже, більш точні траєкторії прогнозуються в дуже інтерактивних ситуаціях. Детальний опис кожного етапу наведено в наступних розділах.

2.5 Висновки

У поточному розділі було розділено задачу прогнозування траєкторій учасників дорожнього руху на декілька підзадач. А саме розпізнавання об’єктів, знаходження відстаней до них, відслідковування об’єктів у відеопотоці, нанесення об’єктів на карту навколо машини, прогнозування траєкторій об’єктів. По кожній з задач проаналізовано літературу, розглянуто декілька методів рішення кожної з підзадач. Були розглянуті такі архітектури, як CenterNet, CenterTrack, а також ієрархічну графову нейронну мережу для прогнозування траєкторій

В розробленому рішенні задачі з розпізнавання об’єктів, оцінки відстаней та відслідковування об’єктів поєднано в один етап, тобто вирішується однією багатозадачною нейронною мережею, що дозволяє скоротити час робити, це стало можливим завдяки використанню центернетівського підходу. Також розроблені моделі працюють з відеопотоком а не з окремими кадрами що дозволяє трохи збільшити точність моделі. В роботі було продемонстровано і візуалізовано результати роботи натренованих моделей.

Розроблені моделі можна використовувати на бортових комп’ютерах автомобілях задля збору інформації щоб допомагати водію з донесенням ситуації що складається навколо машини.

РОЗДІЛ 3 ОПИС РЕАЛІЗАЦІЇ ТА ТРЕНУВАЛЬНИХ ДАНИХ

1.1 Реалізація

Задача складається з декількох етапів. Спочатку розпізнавання об’єктів, їх глибини та відслідковування їх переміщення по кадрах відеопотоку. Потім розміщення розпізнаних об’єктів на карті навколо нашої машини, у якій знаходиться камери, які і знімають відео. Розпізнавання об’єктів, відстаней до них можна робити паралельно, тому будемо це робити однією і тією ж самою нейронною мережею, тобто до одного і того тіла нейронної мережі будемо додавати декілька голів. Функцією втрат в такому випадку буде сума функцій втрат з кожної голови помножених на коефіцієнти. Щодо розпізнавання об’єктів в основному будемо експериментувати з центернетівською головою оскільки вона легше в реалізації, її легше змінювати ніж Yolo, чи SSD. При цьому будемо використовувати не оригінальний CenterNet, а його модифікацію TTF. Де замість того що видавати ширину, висоту і зміщення центра об’єкта, буде відстать лівого, правого, нижнього і верхнього країв об’єкту. На точність це не вплине, а якщо і вплине то в кращу сторону проте в межах похибки, але при цьому це інтуїтивно більш зрозуміла голова. Для відслідковування об’єктів або як це ще називають трекінгом об’єктів можна використовувати CenetTrack, що дозволить нам об’єднати задачі розпізнавання об’єктів, знаходження відстаней до них та трекінг в одну мережу з трьома головами. Розпізнавати будемо 3 класи об’єктів людей, мотоцикли/велосипеди, люди. Отже голова з розпізнавання об’єктів буде складатися з 3+4=7 каналів, відстані 1 канал та трекінг 2 канали, тобто всього 10 каналів.

Усі моделі схематично представлені на рисунках 3.1, 3.2 та 3.3 відповідно.

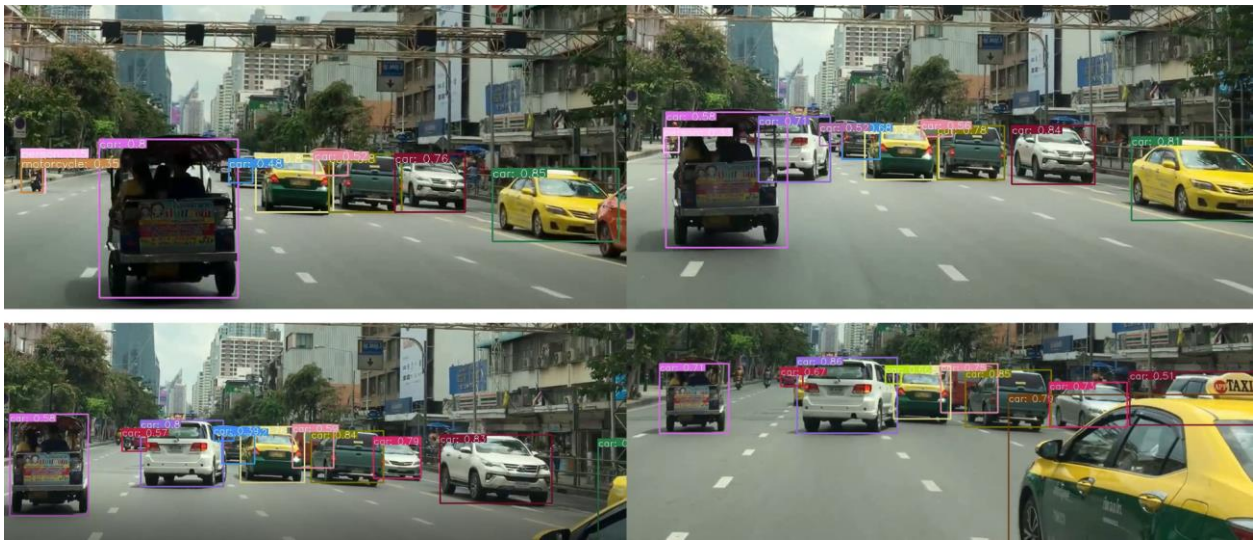

Рисунок 3.1 – Розпізнавання об’єктів разом з відстеженням

1.2 Датасети

Датасет BDD100K складається з багатьох відео знятих з камер на автомобілях, де на кожному кадрі розмічені об’єкти у вигляді прямокутників, кожен з яких має свій ідентифікатор, який однаковий у одного і того ж об’єкта на двох різних кадрах. В датасеті є такі класи об’єктів: автомобіль, вантажний автомобіль, автобус, мотоцикл, велосипед, людина, потяг,

дорожні знаки. Також в датасеті присутня сегментація об’єктів, включаючи сегментацію дорожньої розмітки.

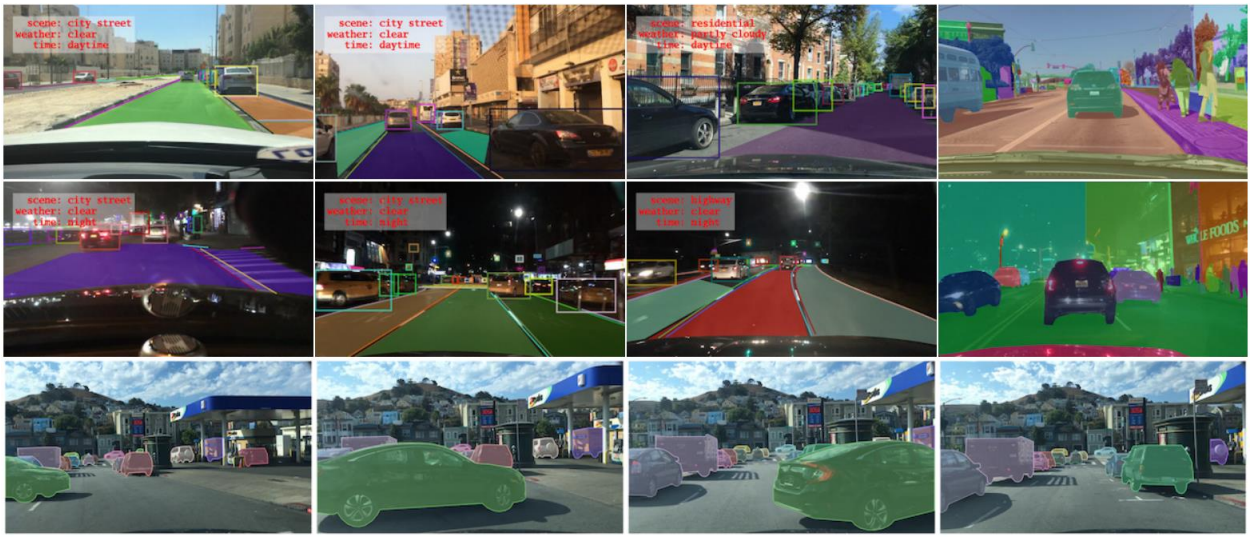

Рисунок 3.2 – Датасет BDD100K

Як видно, можна зробити висновок, що покупці планшетів Xiaomi навряд чи змінюють свої вподобань, на відміну від покупців смартфонів даного бренду, що можуть схилятися до покупки гаджетів фірми-конкурента.

1.3 Висновки

У даному розділі була описана кінцева структура моделей та рішення кожного з етапів задачі: розпізнавання об’єктів, знаходження відстаней до них, відслідковування об’єктів у відеопотоці, нанесення об’єктів на карту навколо машини, прогнозування траєкторій об’єктів.

В розробленому рішенні задачі з розпізнавання об’єктів, оцінки відстаней та відслідковування об’єктів поєднано в один етап, тобто вирішується однією багатозадачною нейронною мережею, що дозволяє скоротити час робити, це стало можливим завдяки використанню центернетівського підходу. Також розроблені моделі працюють з відеопотоком а не з окремими кадрами що дозволяє трохи збільшити точність

моделі. В роботі було продемонстровано і візуалізовано результати роботи натренованих моделей.

Розроблені моделі можна використовувати на бортових комп’ютерах автомобілях задля збору інформації щоб допомагати водію з донесенням ситуації що складається навколо машини.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Останні роки довели, що стартапи сильніші, ніж більшість думала. Гнучкий та інноваційний підхід були ключовими, коли світ зіштовхнувся з такими проблемами, як віддалена робота, масштабні зміни в галузі та ринку, а також абсолютно нова реальність.

Здатність стартапів швидко змінюватися та адаптуватися і є ключовою властивістю даного виду бізнесу, не кажучи вже про те, що багато стартапів продовжували рости та розширювати свої команди.

Стартап — це бізнес-структура, що розвивається та працює на основі інновацій, створена для вирішення проблеми шляхом надання нової пропозиції в умовах надзвичайної невизначеності.

Власне, стартап – це бізнес, який:

- швидко росте;
- порушує ринок або галузь (щось нове, що змушує конкурентів удосконалюватись);
- вирішує проблему;
- працює в умовах надзвичайної невизначеності.

Багато підприємців і відомих бізнес-магнатів визначають стартап як культуру та менталітет побудови бізнесу на основі інноваційної ідеї для вирішення критичних проблем.

Одне, що відрізняє стартапи від інших компаній — це зв’язок між їхнім продуктом та його попитом. Стартапи мають продукти, орієнтовані на невикористаний ринок. Підприємці-початківці знають ідеальну стратегію, щоб створити продукт, який хоче ринок, а також охопити й обслуговувати їх усіх. Це викликає швидке зростання. [19].

1.1 Опис ідеї проекту

В межах підпункту було проаналізовано і подано у вигляді таблиць:

- зміст ідеї;
- можливі напрямки застосування;
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару;
- чим відрізняється від існуючих аналогів та замінників.

Перші три пункти подані у вигляді таблиці (таблиця 4.1) і дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів.

Таблиця 4.1 — Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Дана комплексна система дозволяє прогнозувати траєкторії учаснивік дорожнього руху.	Прогнозування траєкторії об’єктів	Попередження про небезпеку під час водіння
	Розпізнавання об’єктів	Додаткова інформація для аналізу

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї;
- визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних

показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

- проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають:
  - гірші значення (W, слабкі);
  - аналогічні (N, нейтральні) значення;
  - кращі значення (S, сильні) (табл. 4.2).

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Технікоекономічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Adobe Target	Amazon Personalize			
1.	Форма виконання	Надавання послуг	Надавання послуг	Надавання послуг		+	
2.	Собівартість	Низька	Висока	Висока			+
3.	Функціонал	Широкий	Широкий	Широкий	+		

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

1.2 Технологічний аудит ідеї проекту

В межах даного підрозділу було проведено аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 4.3):

1. За якою технологією буде виготовлено товар згідно ідеї проекту?
2. Чи існують такі технології, чи їх потрібно розробити/доробити?
3. Чи доступні такі технології авторам проекту?

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Створення системи генерування рекомендацій користувачу за даними покупок користувачів	Використання мови програмування Python	Наявна	Доступна
		Використання мови програмування C#	Відсутні	Недоступні
		Використання мови C++	Відсутні	Недоступні
Обрана технологія реалізації ідеї проекту: мова програмування Python.				

За результатами аналізу таблиці зроблено висновок щодо можливості технологічної реалізації проекту. Технологічним шляхом реалізації проекту було обрано такі технології, як Python через доступність та безкоштовність.

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 4.4).

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	50
2	Загальний обсяг продаж, грн/ум.од	10000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі, %	20%

За результатами аналізу таблиці 4.4 було зроблено висновок, що ринок є привабливим для входження.

Надалі були визначені потенційні групи клієнтів, їх характеристики та сформовано орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Прогноз траєкторій об’єктів	Власник бізнесу	Велика кількість даних	Простота використання, висока точність
Розпізнавання та відслідковування об’єктів	Кінцевий користувач	Цікавить простота у використанні, низька ціна підтримки системи	Швидкість створення, низька ціна

Після визначення потенційних груп клієнтів було проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. 4.6, 4.7).

Таблиця 4.6 – Фактори загроз

№ п/ п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок продуктів з кращими характеристиками	Передбачити додаткові переваги власного програмного продукту (ПП) для того, щоб повідомити про них саме після виходу на ринок конкурентів. Вдосконалення технічних моментів власного продукту. Обрати нову цільову аудиторію і зосередитися на ній: зниження цін.
2	Зміна потреб користувачів	Користувачам необхідний сервіс з більшим/новим функціоналом.	Розроблення гнучкої архітектури програмного забезпечення для легшого впровадження нового функціоналу

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Гнучкі ціни	Зменшення ціни товару задля збільшення попиту	Введення власних гнучких цін
2	Поява нових методів квантування зображення	З’являться нові методи, що будуть швидше, та більш точно квантувати зображення	Покращити ПП додаванням нового функціоналу, розширення можливостей

Надалі було проведено аналіз пропозиції: визначили загальні риси конкуренції на ринку (таблиця 4.8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції- чиста	Існує величезна кількість конкурентів на ринку.	Якісно провести рекламу.
2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти з інших країн	Створити основу ПП таким чином, щоб можна було легко переробити даний ПП для використання у галузях інших країн.
3. За галузевою ознакою - міжгалузева	Продукт може використовуватись для різних галузей	Постійне вдосконалення продукту, що не має прив’язки до сфери
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між видами ПП, їх особливостями.	Створити ПП, враховуючи недозображення конкурентів
5. За характером конкурентних переваг - неціновa	Вдосконалення технології створення ПП, щоб собівартість була нижчою	Удосконалення моделі. Використання більш дешевих технологій для розробки, ніж використовують конкуренти, але тільки якщо ці технології відповідають необхідним вимогам якості.
6. За інтенсивністю - не марочна	Бренд присутній, але його роль незначна	Реклама, участь у конференціях, семінарах.

Було проведено аналіз конкуренції у галузі за моделлю М. Портера (табл. 4.9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаризамінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визначити фактори сили постахальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
	Amazon Personalize	Наявність вже існуючих рішень	-	Контроль якості продукту	Наявність більш широкого функціоналу, зручнішого інтерфейсу та авторитет
Висновки:	Досить інтенсивна конкурентна боротьба з іншими гравцями	Є можливості і виходу на ринок, але є і конкуренти.	-	Клієнти диктують умови роботи на ринку: зручний інтерфейс	Необхідно випускати ПЗ не гірше, ніж у конкурентів та розширяти функціонал.

За результатами аналізу було зроблено висновок про можливість роботи на ринку з огляду на конкурентну ситуацію.

Цей висновок був врахований при формулюванні переліку факторів конкурентоспроможності у наступному пункті. На основі аналізу конкуренції, проведеного в таблиці, а також із урахуванням характеристик ідеї проекту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів маркетингового середовища (табл. 4.6, 4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності.

Аналіз оформлено у (табл. 4.10).

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Ціна	Один із факторів для вибору продукту клієнтом.
2	Якість	Один із факторів для вибору продукту клієнтом.
3	Зручність роботи з програмою	Дозволяє користувачу легко працювати з програмою

За визначеними факторами конкурентоспроможності (табл. 4.10) проведено аналіз сильних та слабких сторін стартап-проекту (табл. 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	15					*		
2	Якість	10			*				
3	Зручність роботи з програмою	15					*		

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 4.11).

Перелік ринкових загроз та ринкових можливостей було складено на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Якість Простота використання Висока швидкодія	Слабкі сторони: Дуже насичений ринок, мала кількість функціоналу, відсутня кросплатформеність.
Можливості: насичення ринку новим підходом до прогнозування; різноманітна клієнтура, вдосконалення системи	Загрози: Конкуренція

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів (таблиця 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	PR, просування бренду	50%	6 місяців
2	Перехід на безкоштовне розповсюдження	75%	3 місяців
3	Партнерство для об’єднання продукції	65%	2 місяці

Після аналізу було обрано альтернативу №2.

1.3 Аналіз ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (таблиця 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Підприємці	Висока	Високий	Сильна	Просто
2	Великі компанії	Середня: велика конкуренція і можливість власних веб-відділів	Високий	Сильна	Складно
3	Маленькі компанії.	Низька	Низький	Слабка	Середня
Які цільові групи обрано:1,2,3					

За результатами аналізу потенційних груп споживачів було обрано цільові групу, для яких буде запропоновано даний товар, та визначено стратегію охоплення ринку - стратегію диференційованого маркетингу (компанія працює з декількома сегментами).

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Постійне оновлення і покращення продукту	Ринкове позиціонування на індивіуальних користувачів	Швидкодія, якість продукту	Концентрований маркетинг

Наступним кроком обрано стратегію конкурентної поведінки (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні.	Компанія буде шукати нових споживачів та забирати існуючих у конкурентів	Буде копіювати, удосконалювати та створювати свої унікальні пропозиції	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (табл. 4.5), а також в залежності від обраної

базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (таблиця 4.16) розроблено стратегію позиціонування (таблиця 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартаппроекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Легкість розуміння, зручний інтерфейс, надійний, швидкий, точний та достовірни ПП для генерації рекомендацій.	Стратегія диферен нціації	Позиція на основі порівняння фірми з товарами конкурентів; Відмінні особливості споживача	Економія часу; Зручність застосування; Практичність та точність результату

Результатом виконання підрозділу стала узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап- компанії на ринку.

1.4 Розроблення маркетингової програми стартап-проекту

Сформовано маркетингову концепцію товару, який отримає споживач. Для цього підсумовано результати попереднього аналізу конкурентоспроможності товару (таблиця 4.18). Концепція товару – письмовий опис фізичних та інших характеристик товару, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидкість отримання результату	Швидка видача рекомендацій наступної пропозиці	Необхідно покращити швидкість навчання моделі для видачі рекомендацій наступної пропозиці
2	Зручність застосування	Нативна підтримка мови програмування Python	Розробка зручного прикладного програмного інтерфейсу
3	Практичність та точність результату	Користувач отримує точні (з малою похибкою розбіжності) результати рекомендацій.	Користувач на виході роботи ПП отримує модель та прогноз, котрі відповідають необхідним показникам варіативності та точності.

Розроблено трирівневу маркетингову модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 4.19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов’язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень Товар з підкріпленням (супроводом) – додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін).

Таблиця 4.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Генерація палітри кольорів зображення за допомогою інтелектуальних систем</td></tr>
<tr><td rowspan="5">II. Товар реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Індивідуальний підхід.
2. Низька ціна.
3. Простота у використанні.</td><td>1.Нм
2.Нм
3.Нм</td><td>1.Технологічна
2.Економічна
3.Технологічна</td></tr>
<tr><td colspan="3">Якість: тестування фірмами аудиторами</td></tr>
<tr><td colspan="3">Пакування: відсутнє</td></tr>
<tr><td colspan="3">Марка: ROSO</td></tr>
</table>

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути,

а також аналіз рівня доходів цільової групи споживачів (таблиця 4.20). Аналіз проведено експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	2000$	3700$	У всіх трьох груп високий рівень доходів	900$--

Наступним кроком є визначення оптимальної системи збуту, в межах якого було прийняте рішення (таблиця 4.21).

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Канал нульового рівня	Продаж	0(напряму)	Власна

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 4.22).

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/ п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Інтеграція API у клієнтській системі	Інтернет	Низька ціна, простота використання, універсальність	Показати переваги рішення над конкурентами, виділити ключові особливості	Створення сайту продукту, розповсюдження інформації про продукт на спеціалізованих ресурсах.

Було визначено, що придбання продукту буде проводитись через мережу Інтернет або при безпосередньому спілкуванні із представниками компанії. Розповсюдження інформації про продукт буде проводитись виключно через Інтернет, адже аудиторія даного продукту активно користується всесвітньою мережею.

Результатом підрозділу стала ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

1.5 Висновки

В даному розділі проведено підготовчий аналіз для впровадження розробленої системи в якості стартап проекту. Досліджено аналогічні конкурентні системи, встановлено сильні та слабкі сторони системи в порівнянні з ними. Також було досліджено можливі шляхи розповсюдження продукту та його ймовірну аудиторію, рівень доходів та ймовірну ціну продукту, що розробляється.

Було проведено аналіз потенційних ризиків і можливостей, а також розраховані основні фінансово-економічні показники проекту. Отримані результати кажуть про те, що реалізація проекту є доцільною. Було визначено сильні сторони проекту: зручність у використанні, ціна, якість та широкий функціонал. Серед слабких варто виділити повільне навчання.

ВИСНОВКИ

Задачу з прогнозування траєкторії учасників дорожнього руху було розкладено на декілька підзадач: розпізнавання об’єктів, знаходження відстаней до них, відслідковування об’єктів у відеопотоці, нанесення об’єктів на карту навколо машини, прогнозування траєкторій об’єктів. По кожній з задач проаналізовано літературу, розглянуто декілька методів рішення кожної з підзадач.

В розробленому рішенні задачі з розпізнавання об’єктів, оцінки відстаней та відслідковування об’єктів поєднано в один етап, тобто вирішується однією багатозадачною нейронною мережею, що дозволяє скоротити час робити, це стало можливим завдяки використанню центернетівського підходу. Також розроблені моделі працюють з відеопотоком а не з окремими кадрами що дозволяє трохи збільшити точність моделі. В роботі було продемонстровано і візуалізовано результати роботи натренованих моделей.

Розроблені моделі можна використовувати на бортових комп’ютерах автомобілях задля збору інформації щоб допомагати водію з донесенням ситуації що складається навколо машини

ПЕРЕЛІК ПОСИЛАНЬ

1. Zhou X., Wang D, Krähenbühl K. *Objects as Points* URL: https://arxiv.org/abs/1904.07850

2. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C., Berg A. *SSD: Single Shot MultiBox Detector* URL: https://arxiv.org/abs/1512.02325

3. Zhou X., Koltun V., Krähenbühl K.. *Tracking Objects as Points* URL: https://arxiv.org/pdf/2004.01177.pdf

4. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. URL: https://arxiv.org/pdf/1805.04687.pdf

5. Jo E., Sunwoo M., Lee M. *Vehicle Trajectory Prediction Using Hierarchical Graph Neural Network for Considering Interaction among Multimodal Maneuvers* URL: https://www.mdpi.com/1424-8220/21/16/5354

6. Yu D., Lee H., Hwang S. *Vehicle Trajectory Prediction with Lane Stream Attention-Based LSTMs and Road Geometry Linearization* URL: https://www.mdpi.com/1424-8220/21/23/8152

7. What is A Startup? URL: https://www.feedough.com/what-is-startup/

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ

1. Файл layers.py

```
import math
import typing as tp
from collections import defaultdict

import tensorflow as tf
from tensorflow.keras import backend
from tensorflow.keras.initializers import Initializer
from tensorflow.python.ops import embedding_ops, math_ops


class NeighborEmbedding(tf.keras.layers.Embedding):
    """
    Neighbor Embedding
    ------------------
    Arguments:
    ----------
        input_dim: Size of sequence.
        output_dim: Size of the hidden layer.
    """
    def __init__(self, input_dim: int, output_dim: int, *,
known_items_relations: tf.SparseTensor, embeddings_initializer:
tp.Union[str, Initializer] = 'uniform', **kwargs):
        if 'decay_rate' in kwargs:
            self.decay_rate = kwargs.pop('decay_rate')
        self.decay_rate = 0.5
        self._A = known_items_relations
        super().__init__(input_dim, output_dim,
embeddings_initializer=embeddings_initializer, **kwargs)

    def get_config(self) -> tp.Dict[str, tp.Any]:
        config = {
            "decay_rate": self.decay_rate,
        }
        base_config = super(RelativePositionEmbedding,
self).get_config()
        return dict(list(base_config.items()) +
list(config.items()))

    def _select_neighbors_for_index(self, token: int):
        neighbor_index = tf.squeeze(
                    tf.gather(self._A.indices,
tf.where(self._A.indices[:,0] == token))[:,:,1], # second index
is neighbor index
```

```
                    axis=[-1])
        neighbor_embeddings =
embedding_ops.embedding_lookup_v2(self.embeddings,
neighbor_index)
        return tf.reduce_sum(neighbor_embeddings, axis=0)

    def _get_aggregated_neighbor_embeddings(self, token:
tf.Tensor) -> tf.Tensor:
        neighbors = tf.map_fn(self._select_neighbors_for_index,
token, fn_output_signature=tf.float32)
        return neighbors

    def call(self, inputs: tf.Tensor) -> tf.Tensor:
        """
        Arguments:
        ----------
            inputs: A tensor tuple of sequences `inputs` of size
`(batch_size, sequence_length)`
            and adjacency matrix `A` of size `(known_vocab_size,
known_vocab_size)`
        Returns:
        --------
            A tensor in shape of `(length, output_dim)`.
        """
        dtype = backend.dtype(inputs)
        if dtype != 'int32' and dtype != 'int64':
            inputs = math_ops.cast(inputs, 'int32')
        out = embedding_ops.embedding_lookup_v2(self.embeddings,
inputs)
        out = tf.multiply(out, 1 - self.decay_rate)
        out_neighbors =
tf.map_fn(self._get_aggregated_neighbor_embeddings, inputs,
fn_output_signature=tf.float32)
        out_neighbors = tf.multiply(out_neighbors,
self.decay_rate)
        out = out + out_neighbors
        if self._dtype_policy.compute_dtype !=
self._dtype_policy.variable_dtype:
            out = math_ops.cast(out,
self._dtype_policy.compute_dtype)
        return out

class PositionEmbedding(tf.keras.layers.Embedding):
    """
    Position Embedding
    ------------------
    Arguments:
    ----------
        input_dim: Size of sequence (maximm length).
        output_dim: Size of the hidden layer.
    """
```

```
    def __init__(self, input_dim: int, output_dim: int, *,
embeddings_initializer: tp.Union[str, Initializer] = 'uniform',
**kwargs):
        super().__init__(input_dim, output_dim,
embeddings_initializer, **kwargs)

    def call(self, inputs: tf.Tensor) -> tf.Tensor:
        """
        Arguments:
        ----------
            inputs: A tensor of size `(batch_size,
sequence_length)`
        Returns:
        --------
            A tensor in shape of `(length, output_dim)`.
        """
        input_shape = tf.shape(inputs[..., tf.newaxis])
        actual_seq_len = input_shape[1]
        position_embeddings = self.embeddings[tf.newaxis,
:actual_seq_len, :]
        new_shape = tf.where([True, True, False], input_shape,
tf.shape(position_embeddings))
        return tf.broadcast_to(position_embeddings, new_shape)

class RelativePositionEmbedding(tf.keras.layers.Layer):
    """
    Relative Position Embedding
    ---------------------------
    This layer calculates the position encoding as a mix of sine
and cosine
    functions with geometrically increasing wavelengths. Defined
and formulized in
    "Attention is All You Need", section 3.5.
    (https://arxiv.org/abs/1706.03762).
    Arguments:
    ----------
        output_dim: Size of the hidden layer.
        min_timescale: Minimum scale that will be applied at
each position
        max_timescale: Maximum scale that will be applied at
each position.
    """
    def __init__(self,
                 output_dim: int,
                 *,
                 min_timescale: float = 1.0,
                 max_timescale: float = 1.0e4,
                 **kwargs):
        # We need to have a default dtype of float32, since the
inputs (which Keras
        # usually uses to infer the dtype) will always be int32.
        if "dtype" not in kwargs:
```

```
            kwargs["dtype"] = "float32"
        super().__init__(**kwargs)
        self.output_dim = output_dim
        self.min_timescale = min_timescale
        self.max_timescale = max_timescale

    def get_config(self) -> tp.Dict[str, tp.Any]:
        config = {
            "output_dim": self.output_dim,
            "min_timescale": self.min_timescale,
            "max_timescale": self.max_timescale,
        }
        base_config = super(RelativePositionEmbedding,
self).get_config()
        return dict(list(base_config.items()) +
list(config.items()))

    def call(self, inputs: tf.Tensor, length: int = None) ->
tf.Tensor:
        """
        Arguments:
        ----------
            inputs: A tensor whose second dimension will be used
as `length`. If
                `None`, the other `length` argument must be
specified.
            length: An optional integer specifying the number of
positions. If both
                `inputs` and `length` are spcified, `length`
must be equal to the second
                dimension of `inputs`.
        Returns:
        --------
            A tensor in shape of `(length, output_dim)`.
        """
        if inputs is None and length is None:
            raise ValueError("If inputs is None, `length` must
be set in "
                             "RelativePositionEmbedding().")
        if inputs is not None:
            input_shape = backend.shape(inputs) # (batch_size,
sequence_length)
            if length is not None and length != input_shape[1]:
                    raise ValueError(
                        "If inputs is not None, `length` must
equal to input_shape[1].")
            length = input_shape[1]
        position = tf.cast(tf.range(length), tf.float32)
        num_timescales = self._output_dim // 2
        min_timescale, max_timescale = self._min_timescale,
self._max_timescale
        log_timescale_increment = (
```

```
            math.log(float(max_timescale) /
float(min_timescale)) /
                (tf.cast(num_timescales, tf.float32) - 1))
        inv_timescales = min_timescale * tf.exp(
            tf.cast(tf.range(num_timescales), tf.float32) *
            -log_timescale_increment)
        scaled_time = tf.expand_dims(position, 1) *
tf.expand_dims(
            inv_timescales, 0)
        position_embeddings = tf.concat(
            [tf.sin(scaled_time), tf.cos(scaled_time)], axis=1)
        return position_embeddings
```

2. Файл models.py

```
import typing as tp
from collections import defaultdict

import numpy as np
import tensorflow as tf
import tensorflow_ranking as tfrk
import tensorflow_recommenders as tfrs

from .constants import INTEGER, PLAIN, RELATIVE
from .layers import (NeighborEmbedding, PositionEmbedding,
                     RelativePositionEmbedding)
from .model_utils import (check_embedding_type,
check_feature_type,
                          check_position_embedding_type)


class Node2Vec(tf.keras.models.Model):
    """
    Node2Vec
    --------
    """
    def __init__(self,
                 embedding_dim: int,
                 target_feature: str,
                 context_feature: str,
                 feature_type: str,
                 feature_vocab: np.ndarray):
        check_feature_type(feature_type)
        super().__init__()
        self._target_feature = target_feature
        self._context_feature = context_feature
```

```
        feature_size = feature_vocab.shape[0] + 1

        if feature_type == INTEGER:
            self.lookup_layer =
tf.keras.layers.IntegerLookup(max_tokens=feature_size,
vocabulary=feature_vocab, oov_token=0)
        else:
            self.lookup_layer =
tf.keras.layers.StringLookup(max_tokens=feature_size,
vocabulary=feature_vocab, oov_token="_PAD_")

        self.embedding_layer = tf.keras.layers.Embedding(
            feature_size,
            embedding_dim,
            mask_zero=True,
            embeddings_initializer='he_normal',
            embeddings_regularizer= tf.keras.regularizers.l2(1e-
6),
            name="embeddings",
            )
        self.dot = tf.keras.layers.Dot(axes=1, normalize=False)

    def call(self, inputs: tp.Dict[str, tf.Tensor]) ->
tf.Tensor:
        target_lookup =
self.lookup_layer(inputs[self._target_feature])
        context_lookup =
self.lookup_layer(inputs[self._context_feature])
        target_embeddings = self.embedding_layer(target_lookup)
        context_embeddings =
self.embedding_layer(context_lookup)
        logits = self.dot([target_embeddings,
context_embeddings])
        return logits

class IRMAE(tf.keras.models.Model):
    """
    Implicit Rank-Minimizing Autoencoder
    ------------------------------------
    """
    def __init__(self, encoder: tf.keras.models.Model, decoder:
tf.keras.models.Model, num_penalty_layers: int = 2):
        super().__init__()
        self.encoder = encoder
        self.decoder = decoder
        encoder_output = self.encoder.layers[-1].output_shape[-
1]
        self.penalty_model = tf.keras.models.Sequential([
            tf.keras.layers.Dense(units=encoder_output)
            for _ in range(num_penalty_layers)
        ])
```

```
    def call(self, inputs) -> tf.Tensor:
        encoder_output = self.encoder(inputs)
        encoder_output = self.penalty_model(encoder_output)
        decoder_output = self.decoder(encoder_output)
        return decoder_output

class PlainEmbeddingModel(tf.keras.models.Model):
    """
    Basic Embedding Model
    ---------------------
    """

    def __init__(self, feature_name: str, feature_type: str,
feature_vocab: np.ndarray, embedding_dim: int,
**embedding_kwargs):
        check_feature_type(feature_type)

        super().__init__()
        self.feature_name = feature_name
        feature_size = feature_vocab.shape[0] + 1
        if feature_type == INTEGER:
            self.lookup_layer =
tf.keras.layers.IntegerLookup(max_tokens=feature_size,
vocabulary=feature_vocab, oov_token=0)
        else:
            self.lookup_layer =
tf.keras.layers.StringLookup(max_tokens=feature_size,
vocabulary=feature_vocab, oov_token="_PAD_")
        self.embedding_layer =
tf.keras.layers.Embedding(feature_size, embedding_dim,
mask_zero=True, **embedding_kwargs)

    def call(self, inputs: tp.Dict[str, tf.Tensor]):
        feature_lookup =
self.lookup_layer(inputs[self.feature_name])
        return self.embedding_layer(feature_lookup) # output
shape (batch_size, embedding_dim)

class LSTMEmbeddingModel(tf.keras.models.Model):
    """
    LSTM Embedding Model
    --------------------
    """
    def __init__(self, feature_name: str, feature_vocab:
tp.List[tp.Any], embedding_dim: int, num_recurrent_units: int,
feature_type: str = "str"):
        check_feature_type(feature_type)

        super().__init__()
        self.feature_name = feature_name
        feature_size = feature_vocab.shape[0] + 1
        if feature_type == INTEGER:
```

```
            self.lookup_layer =
tf.keras.layers.IntegerLookup(max_tokens=feature_size,
vocabulary=feature_vocab, oov_token=0)
        else:
            self.lookup_layer =
tf.keras.layers.StringLookup(max_tokens=feature_size,
vocabulary=feature_vocab, oov_token="_PAD_")
        self.embedding_layer =
tf.keras.layers.Embedding(feature_size, embedding_dim,
mask_zero=True)
        self.lstm = tf.keras.layers.LSTM(num_recurrent_units)

    def call(self, inputs: tp.Dict[str, tf.Tensor]):
        feature_lookup =
self.lookup_layer(inputs[self.feature_name])
        feature_embedding = self.embedding_layer(feature_lookup)
        return self.lstm(feature_embedding)

class AttentionDCN(tf.keras.models.Model):
    """
    Attention Deep Cross Network
    ----------------------------
    """
    def __init__(self, feature_name: str, feature_type: str,
feature_vocab: np.ndarray, embedding_dim: int,
deep_model_layer_sizes: int, use_cross_layer: bool,
sequence_length: int, *, position_embeddings: str = None,
embedding_type: str = 'plain', projection_dim: int = 32,
conv_filters: int = 4, **embedding_kwargs):
        check_feature_type(feature_type)
        check_embedding_type(embedding_type)

        super().__init__()
        self.feature_name = feature_name
        feature_size = feature_vocab.shape[0] + 1

        if feature_type == INTEGER:
            self.lookup_layer =
tf.keras.layers.IntegerLookup(max_tokens=feature_size,
vocabulary=feature_vocab, oov_token=0)
        else:
            self.lookup_layer =
tf.keras.layers.StringLookup(max_tokens=feature_size,
vocabulary=feature_vocab, oov_token="_PAD_")

        self.query_layer_feature =
tf.keras.layers.Conv1D(filters=embedding_dim, padding="same",
kernel_size=conv_filters, input_shape=(sequence_length
,embedding_dim))
        self.key_layer_feature =
tf.keras.layers.Conv1D(filters=embedding_dim, padding="same",
```

```
kernel_size=conv_filters, input_shape=(sequence_length
,embedding_dim))

        self.attention =
tf.keras.layers.Attention(use_scale=True, causal=True)
        self.average_pooling =
tf.keras.layers.GlobalAveragePooling1D()

        self.position_embedding = None
        if position_embeddings:
            check_position_embedding_type(position_embeddings)
            self.query_layer_position =
tf.keras.layers.Conv1D(filters=embedding_dim, padding="same",
kernel_size=conv_filters, input_shape=(sequence_length
,embedding_dim))
            self.key_layer_position =
tf.keras.layers.Conv1D(filters=embedding_dim, padding="same",
kernel_size=conv_filters, input_shape=(sequence_length
,embedding_dim))

            if position_embeddings == RELATIVE:
                self.position_embedding =
RelativePositionEmbedding(output_dim=embedding_dim)
            else:
                self.position_embedding =
PositionEmbedding(input_dim=feature_size,
output_dim=embedding_dim)

        if embedding_type == PLAIN:
            self.embedding_layer =
tf.keras.layers.Embedding(input_dim=feature_size,
output_dim=embedding_dim, mask_zero=True)
        else:
            if "known_items_relations" not in embedding_kwargs:
                raise TypeError("`known_items_relations`
parameter must be specified if using neigbor embeddings")
            known_items_relations =
embedding_kwargs.pop("known_items_relations")
            self.embedding_layer =
NeighborEmbedding(input_dim=feature_size,
output_dim=embedding_dim, mask_zero=True,
known_items_relations=known_items_relations, **embedding_kwargs)

        self.cross_layer = None
        if use_cross_layer:
            self.cross_layer =
tfrs.layers.dcn.Cross(projection_dim,
kernel_initializer="glorot_uniform")
        self.deep_model = [tf.keras.layers.Dense(num_units,
activation='relu') for num_units in deep_model_layer_sizes]
        self.candidate_layer =
tf.keras.layers.Dense(embedding_dim)
```

```
    def call(self, input: tp.Dict[str, tf.Tensor]):
        feature_lookups =
self.lookup_layer(input[self.feature_name])
        item_embedding = self.embedding_layer(feature_lookups)

        item_query = self.query_layer_feature(item_embedding)
        item_key = self.key_layer_feature(item_embedding)

        if self.position_embedding is not None:
            position_embedding =
self.position_embedding(input[self.feature_name])
            position_query =
self.query_layer_position(position_embedding)
            position_key =
self.key_layer_position(position_embedding)

            q = tf.add(item_query, position_query)
            k = tf.add(item_key, position_key)

            attn = self.attention([q, k])
        else:
            attn = self.attention([item_query, item_key])

        attn = self.average_pooling(attn)

        if self.cross_layer is not None:
            attn = self.cross_layer(attn)

        for layer in self.deep_model:
            attn = layer(attn)

        out = self.candidate_layer(attn)
        return out

class RetrievalModel(tfrs.Model):
    """
    Baseline for Retrieval Models
    -----------------------------
    """
    def __init__(self,
                 query_model: tf.keras.models.Model,
                 candidate_model: tf.keras.models.Model,
                 candidate_pool: tf.data.Dataset
                 ) -> None:
        super().__init__()
        self.query_model = query_model
        self.candidate_model = candidate_model

        self.task = tfrs.tasks.Retrieval(
            metrics=tfrs.metrics.FactorizedTopK(
```

```
candidates=candidate_pool.batch(1024).map(self.candidate_model),
                k=100,
                metrics=[

tfrk.keras.metrics.MeanAveragePrecisionMetric(topn=1,
name='map_1'),

tfrk.keras.metrics.MeanAveragePrecisionMetric(topn=10,
name='map_10'),

tfrk.keras.metrics.MeanAveragePrecisionMetric(topn=100,
name='map_100'),
                        tfrk.keras.metrics.NDCGMetric(topn=1,
name='ndcg_1'),
                        tfrk.keras.metrics.NDCGMetric(topn=10,
name='ndcg_10'),
                        tfrk.keras.metrics.NDCGMetric(topn=100,
name='ndcg_100'),
                        ]
            ),
        )

    def compute_loss(self, features, training=False):
        # We only pass the user id and timestamp features into
the query model. This
        # is to ensure that the training inputs would have the
same keys as the
        # query inputs. Otherwise the discrepancy in input
structure would cause an
        # error when loading the query model after saving it.
        query_embeddings = self.query_model(features)
        embedding_type = self.candidate_model(features)

        return self.task(
            query_embeddings, embedding_type,
compute_metrics=not training)
```

3. Файл process.py

```
import logging
import typing as tp
from collections import defaultdict

import hydra
import networkx as nx
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from hydra.utils import to_absolute_path as abspath
from omegaconf import DictConfig
from tqdm import tqdm


def get_aggregated_sessions_data(events: pd.DataFrame,
session_column: str, user_column: str, item_column: str,
transaction_type_column: str) -> pd.DataFrame:
    sessions = events.groupby(session_column).agg(
        {user_column: 'first', item_column: list,
transaction_type_column: list}
        )
    return sessions

def prepare_dataset_for_sequential_model(sessions: pd.DataFrame,
item_column: str, transaction_type_column: str, target_column:
str) -> pd.DataFrame:
    sessions[target_column] = sessions[item_column].map(lambda
x: x[-1:])
    sessions[item_column] = sessions[item_column].map(lambda x:
x[-11:-1])
    sessions[transaction_type_column] =
sessions[transaction_type_column].map(lambda x: x[-11:-1])
    return sessions[sessions[item_column].map(len)>1]

def calculate_frequencies(sessions: pd.DataFrame, item_column:
str) -> tp.Dict[str, int]:
    pair_frequency = defaultdict(int)
    item_frequency = defaultdict(int)

    for group in tqdm(sessions[item_column],
                      position=0,
                      leave=True,
                      desc="Compute item rating frequency"):
    # Get a list of movies rated by the user.
        current_movies = group
        for i in range(len(current_movies)):
            item_frequency[current_movies[i]] += 1
            for j in range(i + 1, len(current_movies)):
                x = min(current_movies[i], current_movies[j])
                y = max(current_movies[i], current_movies[j])
                pair_frequency[(x, y)] += 1

    return item_frequency, pair_frequency

def construct_knowledge_graph(item_frequency: tp.Dict[str, int],
pair_frequency: tp.Dict[str, int], weight_threshold: int = 10) -
> nx.Graph:
    D = np.log(sum(item_frequency.values()))
```

```
    # Create the movies undirected graph.
    item_graph = nx.Graph()
    # Add weighted edges between movies.
    # This automatically adds the movie nodes to the graph.
    for pair in tqdm(
        pair_frequency, position=0, leave=True, desc="Creating
the item graph"
    ):
        x, y = pair
        xy_frequency = pair_frequency[pair]
        x_frequency = item_frequency[x]
        y_frequency = item_frequency[y]
        pmi = np.log(xy_frequency) - np.log(x_frequency) -
np.log(y_frequency) + D
        weight = pmi * xy_frequency
        if weight >= weight_threshold:
            item_graph.add_edge(x, y, weight=weight)

    return item_graph

def random_walk_next_step(graph: nx.Graph, previous: int,
current: int, p: float, q: float) -> int:
    neighbors = list(graph.neighbors(current))

    weights = []
    # Adjust the weights of the edges to the neighbors with
respect to p and q.
    for neighbor in neighbors:
        if neighbor == previous:
            # Control the probability to return to the previous
node.
            weights.append(graph[current][neighbor]["weight"] /
p)
        elif graph.has_edge(neighbor, previous):
            # The probability of visiting a local node.
            weights.append(graph[current][neighbor]["weight"])
        else:
            # Control the probability to move forward.
            weights.append(graph[current][neighbor]["weight"] /
q)

    # Compute the probabilities of visiting each neighbor.
    weight_sum = sum(weights)
    probabilities = [weight / weight_sum for weight in weights]
    # Probabilistically select a neighbor to visit.
    next = np.random.choice(neighbors, size=1,
p=probabilities)[0]
    return next
```

```
def construct_random_walks(graph: nx.Graph, nodes_ordered:
tp.List[int], num_walks: int, num_steps: int, p: float, q:
float) -> tp.List[int]:
    walks = []
    # Perform multiple iterations of the random walk.
    for walk_iteration in range(num_walks):
        np.random.default_rng(0).shuffle(nodes_ordered)

        for node in tqdm(
            nodes_ordered,
            position=0,
            leave=True,
            desc=f"Random walks iteration {walk_iteration + 1}
of {num_walks}",
        ):
            # Start the walk with a random node from the graph.
            walk = [node]
            # Randomly walk for num_steps.
            while len(walk) < num_steps:
                current = walk[-1]
                previous = walk[-2] if len(walk) > 1 else None
                # Compute the next node to visit.
                next = random_walk_next_step(graph, previous,
current, p, q)
                walk.append(next)
            # Add the walk to the generated sequence.
            walks.append(walk)
    return walks

def generate_dataset_for_node2vec_model(sequences: tp.List[int],
window_size: int, num_negative_samples: int, vocabulary_size:
int) -> tf.data.Dataset:
    example_weights = defaultdict(int)
    # Iterate over all sequences (walks).
    for sequence in tqdm(
        sequences,
        position=0,
        leave=True,
        desc=f"Generating postive and negative examples",
    ):
        # Generate positive and negative skip-gram pairs for a
sequence (walk).
        pairs, labels =
tf.keras.preprocessing.sequence.skipgrams(
            sequence,
            vocabulary_size=vocabulary_size,
            window_size=window_size,
            negative_samples=num_negative_samples,
        )
        for idx in range(len(pairs)):
            pair = pairs[idx]
            label = labels[idx]
```

```
            target, context = min(pair[0], pair[1]),
max(pair[0], pair[1])
            if target == context:
                continue
            entry = (target, context, label)
            example_weights[entry] += 1

    targets, contexts, labels, weights = [], [], [], []
    for entry in tqdm(
        example_weights,
        position=0,
        leave=True,
        desc=f"Populating the dataset"
    ):
        weight = example_weights[entry]
        target, context, label = entry
        targets.append(target)
        contexts.append(context)
        labels.append(label)
        weights.append(weight)

    inputs = {
        "target": np.array(targets),
        "context": np.array(contexts),
    }

    dataset = tf.data.Dataset.from_tensor_slices((inputs,
np.array(labels), np.array(weights)))

    return dataset

@hydra.main(config_path="../config", config_name='main')
def process_data(config: DictConfig):
    """Function to process the data"""
    logging.basicConfig(level=logging.DEBUG,
        format='%(asctime)s %(levelname)s %(message)s',
            filename=config.logging.path + 'process_data.txt',
            filemode='w'
        )

    raw_path = abspath(config.raw.path)
    savefile_path = abspath(config.processed.path) + "/"
    logging.info(f"Process data using {raw_path}")

    timestamp_column = config.process.timestamp_column
    user_column = config.process.user_column
    session_column = config.process.session_column
    transaction_type_column =
config.process.transaction_type_column
    item_column = config.process.item_column
    target_column = config.process.target_column
    train_ratio = config.process.train_test_ratio
```

```
    events = pd.read_csv(raw_path)
    logging.info(f"Constructing KG, train and test datasets
{raw_path}")
    events[timestamp_column] =
pd.to_datetime(events[timestamp_column])
    datetime_for_knowledge_graph =
events[timestamp_column].quantile(train_ratio/2).date()
    datetime_for_train_test =
events[timestamp_column].quantile(train_ratio).date()
    events_for_kg =
events[events[timestamp_column].dt.date<=datetime_for_knowledge_
graph]
    events_for_train =
events[events[timestamp_column].dt.date<=datetime_for_train_test
]
    events_for_test =
events[events[timestamp_column].dt.date>datetime_for_train_test]

    user_vocabulary = events_for_train[user_column].unique()
    np.save(savefile_path + user_column, user_vocabulary)
    logging.info(f"User vocabulary saved to {savefile_path +
user_column}")

    transaction_type_vocabulary =
events_for_train[transaction_type_column].unique()
    np.save(savefile_path + transaction_type_column,
transaction_type_vocabulary)
    logging.info(f"Transaction vocabulary saved to
{savefile_path + transaction_type_column}")


    sessions_for_kg =
get_aggregated_sessions_data(events_for_kg, session_column,
user_column, item_column, transaction_type_column)
    sessions_for_train =
get_aggregated_sessions_data(events_for_train, session_column,
user_column, item_column, transaction_type_column)
    sessions_for_train =
prepare_dataset_for_sequential_model(sessions_for_train,
item_column, transaction_type_column, target_column)
    sessions_for_test =
get_aggregated_sessions_data(events_for_test, session_column,
user_column, item_column, transaction_type_column)
    sessions_for_test =
prepare_dataset_for_sequential_model(sessions_for_test,
item_column, transaction_type_column, target_column)

    item_frequency, pair_frequency =
calculate_frequencies(sessions_for_kg, item_column)
    knowledge_graph = construct_knowledge_graph(item_frequency,
pair_frequency)
```

```
    logging.info(f"Total number of graph nodes:
{knowledge_graph.number_of_nodes()}")
    logging.info(f"Total number of graph edges:
{knowledge_graph.number_of_edges()}")

    kg_nodes = list(knowledge_graph.nodes)

    items_not_in_kg =
set(events_for_train[item_column].unique()) - set(kg_nodes)

    item_vocabulary = np.array(kg_nodes + list(items_not_in_kg))
    np.save(savefile_path + item_column, item_vocabulary)
    logging.info(f"Item vocabulary saved to {savefile_path +
item_column}")
    walks = construct_random_walks(knowledge_graph, kg_nodes,
config.process.num_walks, config.process.num_steps,
config.process.p, config.process.q)

    node2vec_dataset =
generate_dataset_for_node2vec_model(walks,
config.process.window_size, config.process.num_negative_samples,
len(item_vocabulary))
    tf.data.experimental.save(node2vec_dataset, savefile_path +
"node2vec_dataset")
    logging.info(f"Node2Vec dataset saved to {savefile_path +
'node2vec_dataset'}")

    adjacency_matrix =
nx.adjacency_matrix(knowledge_graph).tocoo()
    indices = np.mat([adjacency_matrix.row + 1,
adjacency_matrix.col + 1]).transpose()
    adj_matrix_sparse = tf.SparseTensor(indices,
adjacency_matrix.data, dense_shape=(len(item_vocabulary) + 1, )
* 2)
    serialized_adj_matrix =
tf.io.serialize_sparse(adj_matrix_sparse).numpy()[0]
    tf.io.write_file(savefile_path + "adjacency_matrix",
serialized_adj_matrix)
    logging.info(f"Adjacency matrix of KG saved to
{savefile_path + 'adjacency_matrix'}")


    sessions_for_train_ds = tf.data.Dataset.from_tensor_slices(
        {
            user_column:
sessions_for_train[user_column].values.tolist(),
            transaction_type_column:
tf.keras.preprocessing.sequence.pad_sequences(sessions_for_train
[transaction_type_column].values.tolist(), dtype=object,
value='_PAD_', padding='post'),
```

```
            item_column:
tf.keras.preprocessing.sequence.pad_sequences(sessions_for_train
[item_column].values.tolist(), value=0, padding='post'),
            target_column:
sessions_for_train[target_column].values.tolist(),
        }
    )
    tf.data.experimental.save(sessions_for_train_ds,
savefile_path + "train")
    logging.info(f"Train dataset of KG saved to {savefile_path +
'train'}")

    sessions_for_test_ds = tf.data.Dataset.from_tensor_slices(
        {
            user_column:
sessions_for_test[user_column].values.tolist(),
            transaction_type_column:
tf.keras.preprocessing.sequence.pad_sequences(sessions_for_test[
transaction_type_column].values.tolist(), dtype=object,
value='_PAD_', padding='post'),
            item_column:
tf.keras.preprocessing.sequence.pad_sequences(sessions_for_test[
item_column].values.tolist(), value=0, padding='post'),
            target_column:
sessions_for_test[target_column].values.tolist(),
        }
    )
    tf.data.experimental.save(sessions_for_test_ds,
savefile_path + "test")
    logging.info(f"Test dataset of KG saved to {savefile_path +
'test'}")

if __name__ == '__main__':
    process_data()
```

4. Файл NodeAttentionNetworks.ipynb

```
# %%
! pip install --quiet tensorflow-recommenders tensorflow-ranking
tensorflow-addons kaggle hydra hydra-core

# %%
! git clone https://github.com/tupoylogin/dge-ann

# %%
```

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! kaggle datasets download -d mkechinov/ecommerce-events-
history-in-electronics-store
! unzip /content/ecommerce-events-history-in-electronics-
store.zip
! cp events.csv dge-ann/data/raw/events.csv

# %%
import os
os.chdir('dge-ann')

import tensorflow as tf
import numpy as np
import pandas as pd

from src.models import Node2Vec, PlainEmbeddingModel,
LSTMEmbeddingModel, AttentionDCN, RetrievalModel

# %%
! python src/process.py

# %%
adjacency_matrix_path = 'data/processed/adjacency_matrix'
node2vec_dataset_path = 'data/processed/node2vec_dataset'
train_dataset_path = 'data/processed/train'
test_dataset_path = 'data/processed/test'

users_vocab_path = 'data/processed/user_id.npy'
users_feature_type = "int"

items_vocab_path = 'data/processed/product_id.npy'
items_feature_type = "int"

events_vocab_path = 'data/processed/event_type.npy'
events_feature_type = "str"

learning_rate = 0.001
embedding_dim = 64
batch_size = 1024

# %%
def construct_adjacency_matrix(path: str, shape: int) ->
tf.SparseTensor:
    adjacency_matrix = tf.io.read_file(path)
    adjacency_matrix = tf.io.parse_tensor(adjacency_matrix,
out_type=tf.int64)
    len_of_adj_matrix = adjacency_matrix.shape[0]
    adjacency_matrix = tf.sparse.SparseTensor(
        indices=np.append([[0, 0]], adjacency_matrix.numpy(),
axis=0),
```

```
        values=[1 for _ in range(len_of_adj_matrix+1)],
        dense_shape=(shape, shape)
    )
    return adjacency_matrix

# %%
users_vocab = np.load(users_vocab_path)
events_vocab = np.load(events_vocab_path, allow_pickle=True)
items_vocab = np.load(items_vocab_path)

adjacency_matrix =
construct_adjacency_matrix(adjacency_matrix_path,
shape=len(items_vocab) + 1)

# %%
node2vec_dataset =
tf.data.experimental.load(node2vec_dataset_path)
node2vec_dataset = node2vec_dataset.batch(batch_size,
drop_remainder=True)
node2vec_dataset = node2vec_dataset.prefetch(tf.data.AUTOTUNE)

# %%
node2vec_model = Node2Vec(
    embedding_dim=embedding_dim,
    target_feature='target',
    context_feature='context',
    feature_type='int',
    feature_vocab=items_vocab)

node2vec_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate),
    loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
)

# %%
node2vec_model.fit(node2vec_dataset, epochs=10)

# %%
item_embeddings =
node2vec_model.get_layer("embeddings").get_weights()[0]
print("Embeddings shape:", item_embeddings.shape)

# %%
np.save('data/processed/node2vec_embeddings', item_embeddings)

# %%
item_embeddings =
np.load('data/processed/node2vec_embeddings.npy')

# %%
import typing as tp
```

```
import tensorflow_recommenders as tfrs
import tensorflow_ranking as tfrk

class QueryModel(tf.keras.Model):
  """Model for encoding user queries."""

  def __init__(self,
               deep_model_layer_sizes: tp.List[str],
               embedding_dim: int,
               use_cross_layer: bool,
               position_embeddings: str = None,
               embedding_type: str = None,
               **item_embedding_kwargs):
    """Model for encoding user queries.

    Args:
      deep_model_layer_sizes:
        A list of integers where the i-th entry represents the
number of units
        the i-th layer contains.
    """
    super().__init__()

    # We first use the user model for generating embeddings.
    self.user_model = PlainEmbeddingModel(
        feature_name="user_id",
        feature_type=users_feature_type,
        feature_vocab=users_vocab,
        embedding_dim=embedding_dim)
    self.event_type_model = tf.keras.models.Sequential(
        [
         PlainEmbeddingModel(
          feature_name="event_type",
          feature_type=events_feature_type,
          feature_vocab=events_vocab,
          embedding_dim=embedding_dim,
          ),
         tf.keras.layers.GlobalAveragePooling1D()
        ]
        )
    self.item_history = AttentionDCN(
        feature_name="product_id",
        feature_type=items_feature_type,
        feature_vocab=items_vocab,
        embedding_dim=embedding_dim,
        deep_model_layer_sizes=deep_model_layer_sizes,
        use_cross_layer=use_cross_layer,
        position_embeddings=position_embeddings,
        embedding_type=embedding_type,
        **item_embedding_kwargs)
    # Then construct the layers.
    self.dense_layers = tf.keras.Sequential()
```

```
    # Use the ReLU activation for all but the last layer.
    for layer_size in deep_model_layer_sizes[:-1]:
      self.dense_layers.add(tf.keras.layers.Dense(layer_size,
activation="relu"))

    # No activation for the last layer.
    for layer_size in deep_model_layer_sizes[-1:]:
      self.dense_layers.add(tf.keras.layers.Dense(layer_size))

  def call(self, inputs):
    user_embedding = self.user_model(inputs)
    event_type_embedding = self.event_type_model(inputs)
    item_history_embeddings = self.item_history(inputs)
    feature_embedding = tf.concat([
        user_embedding,
        event_type_embedding,
        item_history_embeddings
    ], axis=1)
    return self.dense_layers(feature_embedding)

# %%
deep_layer_sizes = [128, 64]

query_model = QueryModel(
    deep_model_layer_sizes=deep_layer_sizes,
    use_cross_layer=False,
    position_embeddings='absolute',
    embedding_type='neighbor',
    embedding_dim=embedding_dim,

embeddings_initializer=tf.keras.initializers.Constant(item_embed
dings),
    known_items_relations=adjacency_matrix,
    sequence_length=10)

candidate_model = tf.keras.models.Sequential([
    PlainEmbeddingModel(
      feature_name="target_product_id",
      embedding_dim=embedding_dim,
      feature_type=items_feature_type,
      feature_vocab=items_vocab,
    ),
    tf.keras.layers.GlobalAveragePooling1D()
])

# %%
train_dataset =
tf.data.experimental.load(train_dataset_path).batch(batch_size).
cache()
```

```
test_dataset =
tf.data.experimental.load(test_dataset_path).batch(batch_size).c
ache()

# %%
dge_ann_model = RetrievalModel(
      query_model=query_model,
      candidate_model=candidate_model,

candidate_pool=tf.data.Dataset.from_tensor_slices({"target_produ
ct_id": items_vocab[..., np.newaxis]})
    )

# %%
dge_ann_model.compile(optimizer=tf.keras.optimizers.Adagrad(0.01
))

dge_ann_history = dge_ann_model.fit(
    train_dataset,
    validation_data=test_dataset,
    validation_freq=5,
    epochs=50,
    verbose=1)

# %%
import matplotlib.pyplot as plt

num_validation_runs =
len(dge_ann_history.history["val_map_100"])
epochs = [(x + 1)* 5 for x in range(num_validation_runs)]

plt.plot(epochs, dge_ann_history.history["val_map_100"],
label="MAP@100")
plt.plot(epochs, dge_ann_history.history["val_map_10"],
label="MAP@10")
#plt.plot(epochs,
two_layer_history.history["val_factorized_top_k/top_100_categori
cal_accuracy"], label="2 layers")
plt.title("MAP vs epoch")
plt.xlabel("epoch")
plt.ylabel("MAP");
plt.legend()

# %%
query_dcn_model = QueryModel(
    deep_model_layer_sizes=deep_layer_sizes,
    use_cross_layer=True,
    position_embeddings='absolute',
    embedding_type='neighbor',
    embedding_dim=embedding_dim,
```

```
embeddings_initializer=tf.keras.initializers.Constant(item_embed
dings),
    known_items_relations=adjacency_matrix,
    sequence_length=10)

candidate_model = tf.keras.models.Sequential([
    PlainEmbeddingModel(
      feature_name="target_product_id",
      embedding_dim=embedding_dim,
      feature_type=items_feature_type,
      feature_vocab=items_vocab,
    ),
    tf.keras.layers.GlobalAveragePooling1D()
])

dge_ann_dcn_model = RetrievalModel(
      query_model=query_model,
      candidate_model=candidate_model,

candidate_pool=tf.data.Dataset.from_tensor_slices({"target_produ
ct_id": items_vocab[..., np.newaxis]})
    )

# %%
dge_ann_dcn_model.compile(optimizer=tf.keras.optimizers.Adagrad(
0.01))

dge_ann_dcn_history = dge_ann_dcn_model.fit(
    train_dataset,
    validation_data=test_dataset,
    validation_freq=5,
    epochs=50,
    verbose=1)

# %%
import matplotlib.pyplot as plt

num_validation_runs =
len(dge_ann_dcn_history.history["val_map_100"])
epochs = [(x + 1)* 5 for x in range(num_validation_runs)]

plt.plot(epochs, dge_ann_dcn_history.history["val_map_100"],
label="MAP@100")
plt.plot(epochs, dge_ann_dcn_history.history["val_map_10"],
label="MAP@10")
#plt.plot(epochs,
two_layer_history.history["val_factorized_top_k/top_100_categori
cal_accuracy"], label="2 layers")
plt.title("MAP vs epoch")
plt.xlabel("epoch")
plt.ylabel("MAP");
```

```
plt.legend()

# %%
class LSTMQueryModel(tf.keras.Model):
  """Model for encoding user queries."""

  def __init__(self,
               deep_model_layer_sizes: tp.List[str],
               embedding_dim: int,
               **embedding_kwargs,
               ):
    """Model for encoding user queries.

    Args:
      deep_model_layer_sizes:
        A list of integers where the i-th entry represents the
number of units
        the i-th layer contains.
    """
    super().__init__()
    # We first use the user model for generating embeddings.
    self.user_model = PlainEmbeddingModel(
        feature_name="user_id",
        feature_type=users_feature_type,
        feature_vocab=users_vocab,
        embedding_dim=embedding_dim)
    self.event_type_model = tf.keras.models.Sequential(
        [
         PlainEmbeddingModel(
          feature_name="event_type",
          feature_type=events_feature_type,
          feature_vocab=events_vocab,
          embedding_dim=embedding_dim,
          ),
         tf.keras.layers.GlobalAveragePooling1D()
        ]
        )
    self.item_history = LSTMEmbeddingModel(
        feature_name="product_id",
        feature_type=items_feature_type,
        feature_vocab=items_vocab,
        embedding_dim=embedding_dim,
        num_recurrent_units=embedding_dim,
        **embedding_kwargs)
    # Then construct the layers.
    self.dense_layers = tf.keras.Sequential()

    # Use the ReLU activation for all but the last layer.
    for layer_size in deep_model_layer_sizes[:-1]:
      self.dense_layers.add(tf.keras.layers.Dense(layer_size,
activation="relu"))
```

```
    # No activation for the last layer.
    for layer_size in deep_model_layer_sizes[-1:]:
      self.dense_layers.add(tf.keras.layers.Dense(layer_size))

  def call(self, inputs):
    user_embedding = self.user_model(inputs)
    event_type_embedding = self.event_type_model(inputs)
    user_history_embeddings = self.item_history(inputs)
    feature_embedding = tf.concat([
        user_embedding,
        event_type_embedding,
        user_history_embeddings
    ], axis=1)
    return self.dense_layers(feature_embedding)

# %%
query_lstm_model = LSTMQueryModel(
    deep_model_layer_sizes=deep_layer_sizes,
    embedding_dim=embedding_dim,
    )
candidate_model = tf.keras.models.Sequential([
    PlainEmbeddingModel(
      feature_name="target_product_id",
      embedding_dim=embedding_dim,
      feature_type=items_feature_type,
      feature_vocab=items_vocab,
    ),
    tf.keras.layers.GlobalAveragePooling1D()
])

lstm_model = RetrievalModel(
      query_model=query_lstm_model,
      candidate_model=candidate_model,

candidate_pool=tf.data.Dataset.from_tensor_slices({"target_produ
ct_id": items_vocab[..., np.newaxis]})
    )

# %%
query_lstm_model.item_history.embedding_layer.embeddings_initial
izer = tf.keras.initializers.Constant(item_embeddings)

# %%
lstm_model.compile(optimizer=tf.keras.optimizers.Adagrad(0.001))

lstm_history = lstm_model.fit(
    train_dataset,
    validation_data=test_dataset,
    validation_freq=5,
    epochs=50,
    verbose=1)
```

```
# %%
num_validation_runs = len(lstm_history.history["val_map_100"])
epochs = [(x + 1)* 5 for x in range(num_validation_runs)]

plt.plot(epochs, lstm_history.history["val_map_100"],
label="MAP@100")
plt.plot(epochs, lstm_history.history["val_map_10"],
label="MAP@10")
#plt.plot(epochs,
two_layer_history.history["val_factorized_top_k/top_100_categori
cal_accuracy"], label="2 layers")
plt.title("MAP vs epoch")
plt.xlabel("epoch")
plt.ylabel("MAP");
plt.legend()

# %%
import matplotlib.pyplot as plt

# %%
plt.figure(figsize=(10, 5))
plt.plot(epochs, lstm_history.history["val_map_10"], label="LSTM
MAP@10")
plt.plot(epochs, dge_ann_history.history["val_map_10"],
label="DGE-ANN MAP@10")
plt.plot(epochs, dge_ann_dcn_history.history["val_map_10"],
label="DGE-ANN DCN MAP@10")

plt.title("MAP@10 vs epoch")
plt.xlabel("epoch")
plt.ylabel("MAP");
plt.legend()
plt.show()

# %%
plt.figure(figsize=(10, 5))
plt.plot(epochs, lstm_history.history["val_ndcg_10"],
label="LSTM NDCG@10")
plt.plot(epochs, dge_ann_history.history["val_ndcg_10"],
label="DGE-ANN NDCG@10")
plt.plot(epochs, dge_ann_dcn_history.history["val_ndcg_10"],
label="DGE-ANN DCN NDCG@10")

plt.title("NDCG@10 vs epoch")
plt.xlabel("epoch")
plt.ylabel("NDCG");
plt.legend()
plt.show()

# %%
```