КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота
на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки

на тему:

АЛГОРИТМ ВИЗНАЧЕННЯ ЙМОВІРНОСТІ ТРАЄКТОРІЇ РУХУ АВТОМОБІЛЯ НА ПЕРЕХРЕСТІ

Виконав студент 4 курсу
Садок Микита Олександрович _____
(підпис)

Науковий керівник:
професор, доктор фіз.-мат. наук
Терещенко Василь Миколайович _____
(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри математичної інформатики
«____» _____ 202_ р.,
протокол № ___
Завідувач кафедри
В. М. Терещенко _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 38 сторінок, 14 ілюстрацій, 2 таблиці, 12 джерел посилань. ТРАЄКТОРІЯ РУХУ, ПЕРЕХРЕСТЯ, ПРЕДИКТИВНА МОДЕЛЬ, SMART CAR, БЕЗПЕКА, INTERACTION.

Об’єктом роботи є процес розв’язування задачі передбачення руху автомобіля на перехресті. Предметом роботи є програмний засіб для розрахунку ймовірності і генерації різних траєкторій транспортного засобу на перехресті.

Метою роботи є створення програмного забезпечення для розв’язування задачі передбачення руху автомобіля на перехресті.

Методи розробки: комп’ютерне моделювання, методи побудови і навчання нейронних мереж. Інструменти розробки: середовище розробки Jupyter Notebook, мова програмування Python, фреймворк TensorFlow, фреймворк Pandas.

Результати обробки: виконано огляд підходів, які використовуються для розрахунку ймовірності траєкторії, проаналізовано переваги кожного з цих методів, запропоновано власне рішення цієї задачі, розроблена програма, розв’язує та візуалізує рішення даної проблеми.

Програмний продукт може використовуватись у автомобілях з технологією Smart Car для підвищення безпеки водія та його автомобіля під час переїзду перехрестя.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об'єкта розробки. З самого зародження автомобілів та початку дорожнього руху однією з найактуальніших проблем були зіткнення різних масштабів: починаючи з колізій на невеликих швидкостях і закінчуючи летальними автокатастрофами. Розробка алгоритму та програмного забезпечення, що може убезпечити водія від зіткнень є однією з найважливіших проблем, яку мають подолати розробники "Smart Car" – розумного автомобіля. Також дана розробка буде корисною для машин на автопілоті: система зможе оцінювати ризики і обирати ту траєкторію руху, на яку в той самий момент часу не буде повертати інший учасник дорожнього руху.

Проблемами передбачення траєкторій руху транспортних засобів займалися: T.Streubel, H.Cui, V.Radosavljevic, S.Wang[1], та ін. У роботах цих вчених розглядаються проблеми передбачення траєкторій у різних дорожніх та погодних умовах, наявні вказівки для правильного збору даних, необхідних для створення застосунку.

Актуальність роботи та підстави для її виконання. При керуванні транспортних засобів людьми нерідко виникають ситуації, коли через людський фактор (недостатня кількість сну, погана видимість, поганий зір загалом, захворювання, незнання правил дорожнього руху, тощо) виникають аварійні ситуації, що загрожують життю людині, і часто не одній. Саме для такого випадку буде доцільним використати рішення за залучення математичного апарату, яке не буде давати збої через людей. Отже, необхідна

розробка допоміжного програмного забезпечення, яка допоможе якщо не нівелювати, то значно зменшувати вплив людських помилок.

Мета й завдання роботи. Метою роботи є створення робастної моделі машинного навчання, яка допоможе розв'язати задачу вибору оптимальної траєкторії шляхом виведення декількох найбільш вірогідних варіантів. Для досягнення цієї мети поставлені наступні завдання:

- Дослідити існуючі алгоритми та підходи до розв'язання такої чи подібної задачі
- Дослідити застосування різних архітектур нейронних мереж
- Розробити власний алгоритм розрахунку траєкторії
- Розробити варіації цього алгоритму та визначити найефективнішу
- Протестувати рішення

Об'єкт, методи й засоби розроблення. Об'єктом розроблення є модель машинного навчання, що буде навчена визначати можливі траєкторії руху транспортного засобу разом із ймовірностями кожної з запропонованих траєкторій.

Власне розробці алгоритму передувала генерація набору даних, потрібних для навчання моделі машинного навчання,задача якого полягала в першу чергу у виборі показників, які можливо зчитати під час руху автомобіля і які можуть стати у нагоді при визначенні траєкторії.

В якості інструментів для розробки даного програмного засобу було обрано програму Jupyter Notebook – інтегроване програмне середовище (IDE) мовами Julia, Python та R, яке є безкоштовним, поширюваним за вільною ліцензією "modified BSD license", та з відкритим вихідним кодом.

Python надає дуже широкий вибір бібліотек для машинного навчання, починаючи з найпростіших навчальних, і закінчуючи дуже просунутуми, готовими до використання у реальній індустрії. Ключовою рисою цих

бібліотек є висока швидкість виконання та детальна документація, підкріплена математичними формулами, що дає змогу ефективно програмувати математичні алгоритми.

Можливі сфери застосування. Розробка даного алгоритму може використовуватись для інтеграції в системи реального часу, які зможуть передбачати траєкторії інших учасників дорожнього руху. Подібний алгоритм також може використовуватися для генерації та подальшого аналізу трафіка при проектуванні доріг та перехресть.

РОЗДІЛ 1.
ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ

Проблема передбачення траєкторії полягає в визначенні короткострокових (1 -3 секунд) або довгострокових (3-5 секунд) координат транспортного засобу (автомобіля, автобуса, пішохода, рікші, тварини, тощо). Ці агенти можуть мати різну поведінку. яка може характеризуватися як консервативна (як наприклад та, що характеризується діями у рамках правил дорожнього руху), так і агресивна [2]. Проте, результатом передбачення траєкторії не обов'язково мають бути координати, це можуть бути будь-які показники руху транспортного засобу, які можна перетворити на координати. Ще одним популярним формулюванням даної задачі є так зване мультимодальне передбачення траєкторії, що полягає у передбаченні n можливих траєкторій разом з ймовірностями їх появи. Таке формулювання допомагає краще продумувати різноманітні можливі варіанти і вцілому вважається більш ефективним і правильним.

Проблема передбачення траєкторії транспортного засобу розв’язується різноманітними методами, включаючи як і традиційні методи (наприклад, використовуючи теорію графів), так і більш сучасні (такі як нейронні мережі). Автори [3] відзначають, що найбільш успішні та точні підходи якраз базуються на нейронних мережах та інших методах машинного навчання. Варто розглянути найголовніші з них:

1) **Використання згорткових нейромереж**

Багато робіт з цієї теми включають в себе обробку деяких графічних даних, що описують рух автомобілів. Так, автори роботи [1] пропонують на вхід до

моделі подавати 2 окремих масива вхідних даних: до першого входить послідовність растрових зображень перехресть під час руху автомобіля, на яких можна побачити розташування агентів дорожнього руху. Другий масив складається з послідовності певних станів, де записані числові дані про транспортний засіб як, наприклад, координати по осям Ox та Oy.

Дані з зображень пропускаються через згорткову нейронну мережу, в оригіналі MobileNetv2, яка відома низькою кількістю параметрів і як наслідок швидким навчанням. Після цього, ознаки, які були отримані згортковою мережею, конкатенуються до другого вхідного масиву (станів) і пропускаються через найпримітивніші повноз'єднані шари. На виході присутні декілька траєкторій автомобіля у вигляді координат транспортного засобу у наступні k моментів часу разом з ймовірностями вибору такої траєкторії водієм. Автори порівнюють цей метод з іншими state-of-the-art методами і переконують, що їх підхід є більш корисним у практичних задачах.

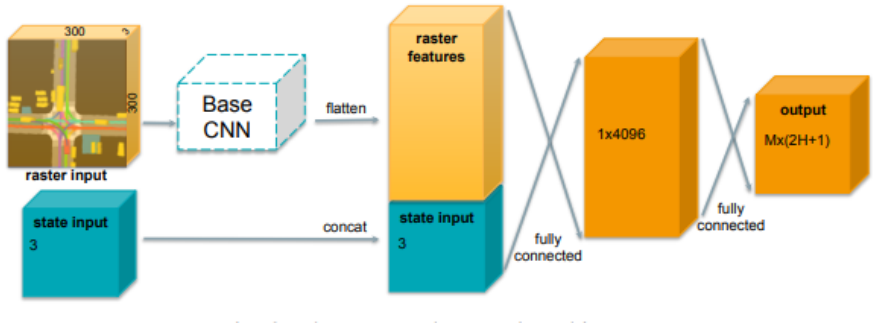

Рисунок 1 - модель, що використовується в статті [1]

2) Використання LSTM-нейромереж

LSTM нейромережі є підвидом рекурентних нейромереж, які відомі тим, що добре працюють з послідовними ланцюжками даних. Автори роботи [4]

розглядають випадок, що схожий на той, що розглянутий в цій роботі, і вони все ще дають передбачення траєкторії руху транспортного засобу, доповнену інформацією про стан. Подібні дані добре підходять під поняття ланцюгових. Одним із головних плюсів підходу LSTM-нейромереж виділяють майже повне розв'язання проблеми зникнення градієнту. Також, при правильно підібраному наборі даних ця архітектура дає дуже точні передбачення. Серед головних недоліків виділяють перенавчання моделі та дуже довгий та потребуючий багато ресурсів процес навчання нейронної мережі. Якщо перша проблема розв'язується викиданням частини входів та виходів моделі (т.зв. Dropout), то з другою проблемою можна боротися лише використовуючи більш ефективне обладнання.

Робота характерна тим, що до уваги береться не тільки інформація про стан автомобіля, а й дані про 9 сусідів, що рухаються поряд з транспортом, для якого ми передбачаємо траєкторію. Це суттєво збільшує кількість змінних та підвищує складність навчання моделі. Проте, модель, що показала найкращі показники є досить примітивною і містить небагато шарів (1 LSTM та 3 повноз'єднаних шари). Як показала ця робота, якісна обробка вхідних даних та виділення потрібних змінних можуть бути ефективнішими за складну модель. Схема використаної моделі на рис.

Іншою важливою особливістю є те, що деякі вхідні змінні дублюються. Так, інформація про швидкість та координати подається як на вхід LSTM шару, так і окремо на шар виводу. Це зроблено для того, щоб LSTM частина нейромережі вивчала більш складні патерни руху транспортних засобів, а те, що прокинуто на пряму - "розбиралося" з більш простими траєкторіями, як, наприклад, рух по прямій. Це дало змогу кожній окремій частині нейромережі сконцентруватись на вивченні окремої поведінки водія.

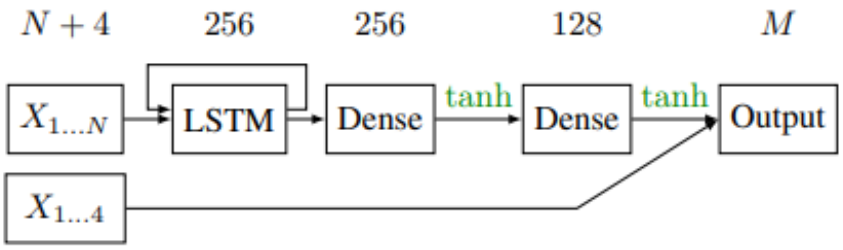

Рисунок 2 - схема моделі, запрононованої в [4]

Прихована марківська модель

Автори роботи [5] пропонують використання прихованих марківських моделей через те, що можливо розглянути переїзд автомобілем перехрестя як динамічний стохастичний процес з дуже високою частотою зміни стану. Серед плюсів виділяють високу швидкість та ефективність навчання, коректну роботу для різних розмірів вхідних даних, що є актуальним для невеликих наборів даних. Серед недоліків виділяють велику кількість параметрів, які зазвичай не структуровані. Також, модель впускає деякі залежності, що можуть бути корисними для передбачення траєкторії руху.

В якості вхідних даних використовується вже традиційний стан, що складається з координат, швидкості, прискорення, кута поворота та відстані до перехрестя. Вхідні дані, які складаються зі станів, зчитаних у певний момент часу, потім комбінуються у послідовності за допомогою кластеризації методом k середніх.

Як висновок з цього підходу, автори дійшли до того, що траєкторії з поворотами є досить очевидними для такого роду алгоритмів, проте він має великі проблеми з побудовою проїзду по прямій, адже, за авторами, таких варіантів такого проїзду може бути багато.

Баєсові мережі

Автори роботи [6] пропонують використовувати баєсові мережі для передбачення траєкторій. Серед переваг відмічають можливість легко підлаштовуватися під закорельовані дані та гарне навчання з як прямих (координати), так і непрямих (швидкість) ознак. Серед недоліків відмічають високу чутливість до якості вхідних даних та високу залежність від вибору початкових ймовірностей.

В цілому, тема є достатньо вивченою і існує велика кількість методів для розв’язання цієї задачі. Їхня ефективність та доцільність варіюється в залежності від набору даних та конкретної постановки задачі. Не існує єдиного оптимального підходу до розв’язання.

5

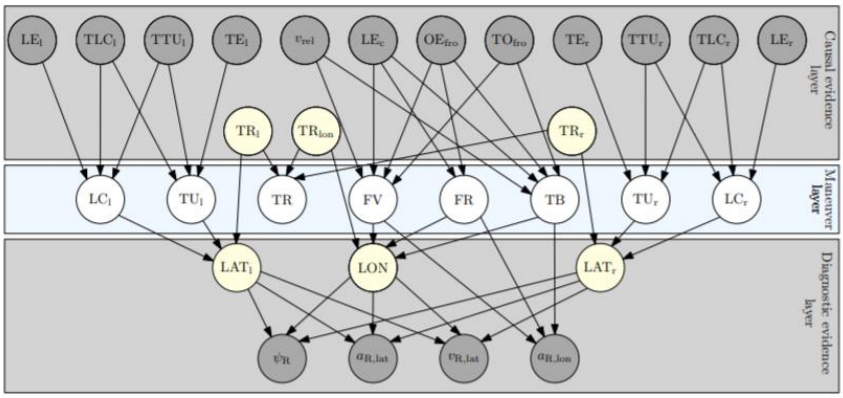

Fig. 3. Bayesian network for maneuver detection instantiated for each vehicle within the traffic scene at each time step k. Hidden maneuver nodes (blue) are inferred via Bayesian inference given causal and diagnostic evidence nodes (shaded). Helper nodes (yellow) are used to facilitate parametrization. Abbreviations are explained in Table I.

Рисунок 3 – Архітектура мережі, запропонованої в [6]

РОЗДІЛ 2.
ОГЛЯД РІШЕННЯ

2.1 Постановка задачі

Задача на власне передбачення траєкторії зазвичай являє собою передбачення комплексного набору ознак, які можуть так чи інакше допомогти у передбаченні поведінки автомобіля. Автори [1] формулюють задачу наступним чином:

Припустимо, що ми маємо дані з сенсорів деякого автомобіля. Нехай, дані певні інтервали часу $T = \{t_1, t_2, \dots, t_T\}$, де T – це кількість часових міток, а T – їх множина. Позначимо стан і-того автомобіля у час t_j як s_{ij}, де $i = \{1, \dots, N_j\}$, де N_j – кількість унікальних учасників дорожнього руху в момент t_j. Також в цій роботі використовуються растрові зображення ділянки дороги і траєкторії руху для і-того учасника дорожнього руху з великою кількістю деталей: дані про смуги руху, світлофори, перехрестя та інше.

За приведеним вище входом треба передбачити M можливих послідовностей станів $\{[\tilde{s}_{im(j+1)}, \dots, \tilde{s}_{im(j+H)}]\}_{m=1\dots M}$ разом з ймовірністю p_{im} появи кожного з M станів, таких ймовірностей, що $\sum_m \quad p_{im} = 1$, де m – індекс можливого варіанту, а H – кількість кроків, на які робиться передбачення.

Задача, розглянута в даній роботі є свого роду переформулюванням задачі, розглянутої в [1]. Основна відмінність в тих змінних, що входять до стану агенту дорожнього руху у певний момент часу. На вході маємо:

$\{s_{1,}s_2, \dots, s_{T_1}\}_{n=1\dots N}$, де $s_i = (x_i, y_i, v_{ix}, v_{iy}, \theta_i, l_i, w_i)$, $x_i, y_i -$ декартові координати транспортного засобу у момент часу i, v_{ix} – моментальне прискорення вздовж осі Ox, v_{iy} – моментальне прискорення вздовж осі Oy, θ_i - кут рискання (повороту) транспортного засобу, l_i, w_i - довжина і ширина транспортного засобу відповідно. N – кількість транспортних засобів. В даній постановці, необхідно передбачити наступні значення, які перегукуються з постановкою задачі з [1]. Вихід виглядає наступним чином: m можливих множин точок $\{c_1, c_2, \dots, c_{T_2}\}$, для кожного спостережуваного автомобіля a, $\forall a = 1 \dots N$, де c_i та N визначені аналогічно до відповідних позначок у вхідних даних, а T_2 – час на який розраховується можлива траєкторія руху автомобіля. Другою частиною виводу є $\{p_{1,}p_2, \dots, p_m\}$, де p_i – ймовірність $i -$ тої траєкторії для $\forall a = 1 \dots N$.

В реалізації, представленій у рамках цієї роботи, значення $T_1 = T_2 = 2$ дс $= 2$ с. Тобто, за 2 секундами руху транспортного засобу робиться передбачення траєкторії руху у наступні 2 секунди.

2.2 Інформація про датасет та попередня обробка даних

2.2.1 Інформація про датасет

Через відсутність змоги розташовувати сенсори на реальних автомобілях, було прийнято рішення використовувати існуючі набори даних, які були зібрані за допомогою збирання даних з сенсорів на реальних машинах. Вибір є досить широким, адже на тему передбачення траєкторії було проведено багато досліджень. Проте, більшість датасетів є графічними. При використанні такого датасета, задача значно ускладнюється і доповнюється складнощами пов'язаними з отриманням стану автомобіля з фото або відео матеріалів. Подібні проблеми також широко досліджуються, але в контексті даної роботи таке ускладнення задачі не є доцільним. Вирішено обрати датасет з вже готовими станами у вигляді csv або txt файлу. Набором даних, що чудово підходить під опис виявився INTERACTION (INTERnational, Adversarial and Cooperative moTION Dataset) датасет [7].

Датасет було зібрано протягом декількох років у чотирьох країнах (США, Німеччина, Китай та Болгарія) і опубліковано в 2019 році міжнародною групою вчених. Датасет являє собою один з найбільших наборів даних, що були зібрані з реальних автомобілів і є одним з найбільш вивчених у літературі. Цей набір містить дані з чотирьох типів транспортних розв'язок: пряма автострада, кругове перехрестя, звичайне перехрестя з та без світлофору. В цій роботі використовуються дані, що були зібрані на декількох сполучених перехрестях без сфітлофору. Ці дані містять у собі близько 9000 траєкторій автомобілів, що проїжджали транспортною розв'язкою з кількома перехрестями. Зі схемою ділянки дороги, де збиралися дані можна ознайомитися на рис. 2. Збір даних проходить з частотою 10 Гц, тобто кожні 0.1 с записуються дані про місцезнаходження та інша інформація про агентів дорожнього руху. Інформація про дорожній рух була записана в періоди дня, коли рух був найбільш інтенсивний, тобто в години пік.

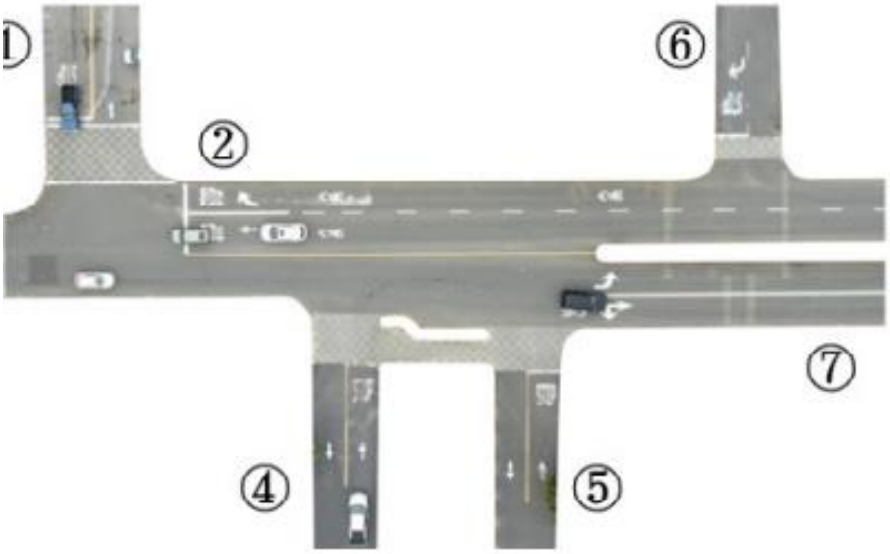

Рисунок 4 - схематичне зображення ділянки дороги, де проводився збір даних

Датасет INTERACTION надає дані про траєкторії у локальних координатах (X, Y), що відносними щодо ділянки дороги, на якій проходив збір даних. Датасет має деякі особливості, які роблять його особливо корисним для дослідження поведінки водіїв та побудови майбутніх траєкторій, а саме:

- присутність різних моделей поведінки (як водіння за правилами дорожнього руху, так і небезпечне кермування з порушеннями)
- різні рівні критичності порушень: є як незначні порушення, як зміна смуги руху там, де цього не можна робити, так і серйозні, що призводять до аварій (як виїзд на зустрічну смугу)
- Деякі аварії зафіксовані даним датасетом, то ж є змога проводити дослідження аварійності різних ситуацій

Єдиним суттєвим недоліком цього датасету може бути досить короткий час спостереження за ситуацією: всього 4 с. Проте, як показує дана робота, цього може бути достатньо для достойного прогнозу траєкторії.

2.2.2 Попередня обробка даних

Датасет NGSIM містить дані, що були зібрані декількома шляхами: статичними камерами, що були закріплені на дорожніх столбах, або ж дронами, що знімали ділянку дороги з висоти пташиного польоту. Дані, що були зібрані дронами, були переведені в координати та інші корисні змінні, були зчитані за допомогою аналізу відео. Так як ці дані є досить шумними, то до них було застосовано фільтр Калмана. Для плавності руху автомобілів було застосовано Rauch-Tung-Striebel smoother [8].

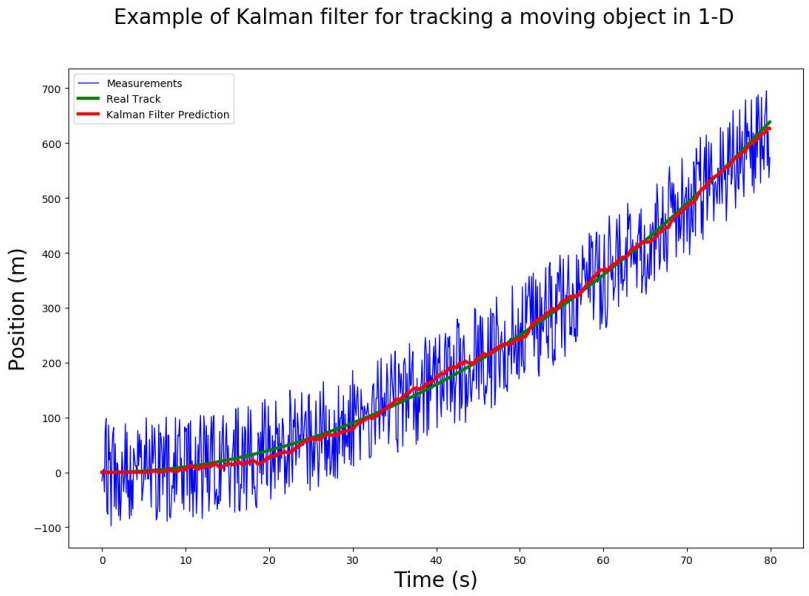

Рисунок 5 - графік величини до і після використання фільтра

Після згаданої вище обробки дані готові для використання та подальшої обробки. Загалом, датасет містить 12 колонок, а саме:

- case_id: ідентифікатор конкретного епізоду дорожнього руху довжиною 4 с.
- track_id: починається з id 1, є унікальним ідентифікатором агенда дорожнього руху в епізоді
- timestamp_ms: часова мітка, що показує коли були зняті дані про агента. Вимірюється в мс.
- agent_type: тип агента (транспортного засобу). Може бути car, truck або pedestrian
- x: координата x, що показує розташування агента в фреймі. Вимірюється в м.
- y: координата y, що показує розташування агента в фреймі. Вимірюється в м.
- v_x : прискорення агента вздовж осі Ox всередині фрейму. Вимірюється в м/с.
- v_y : прискорення агента вздовж осі Oy всередині фрейму. Вимірюється в м/с.
- psi_rad: кут рискання агента у кожному фреймі. Вимірюється в радіанах.
- length: довжина агента в метрах.
- width: ширина агента в метрах.

Обробка даних починається з викидання траєкторій, що пов’язані з пішоходами. Ці траєкторії не будуть корисними для передбачення траєкторій автомобілів. Також, необхідно брати до уваги траєкторії тільки тих автомобілів. які спостерігалися на ділянці руху протягом усього періоду спостереження, тобто всі 4 с. “Обірвані” траєкторії можуть нашкодити точності алгоритму.

З усіх вхідних даних, є сенс викинути декілька колонок. Наприклад, поля track_id та timestamp не є потрібними алгоритму: поля є просто зайвими і не несуть в собі ніякої інформативності, якщо дані правильно згрупувати та подати на вхід алгоритму. Всі інші змінні несуть в собі багато інформації і їх ігнорування може призвести до втрати в точності прогнозів.

З даних формуються вхідні вектори. Для кожного автомобіля у сцені формуються послідовності з 40 станів, зібраних за 4 секунди спостереження. З цих станів викидається непотрібна інформація. Саме ці стани і є основними вхідними одиницями алгоритму. Далі, це стани діляться навпіл - перша половина містить всі показники за перші 2 с руху, а друга містить лише 2 колонки - x та y. Задача: передбачити за допомогою змінних із першої половини дані із другої половини. Так як передбачається декілька траєкторій, то розмірність частини з координатами розширюється.

Далі необхідно провести певні махінації зі значеннями, що вже присутні в колонках. Першим пунктом буде нормалізація координат автомобіля відносно центру траєкторії. Тобто, $\forall x_i, y_i, i = 1 \dots T$, $x_i = x_i - x_{\lfloor \frac{T}{2} \rfloor}$, $y_i = y_i - y_{\lfloor \frac{T}{2} \rfloor}$. По суті, це еквівалентно перенесенню точки початку координат з точки $(0,0)$ в точку $\left(x_{\lfloor \frac{T}{2} \rfloor}, y_{\lfloor \frac{T}{2} \rfloor}\right)$.

Наступним кроком попередньої обробки буде традиційна нормалізація координат транспортного засобу, тобто $\forall x_i, y_i, i = 1 \dots T$, $x_i = \frac{x_i}{x_{MAX}}$, $y_i = \frac{y_i}{y_{MAX}}$, де x_{MAX}, y_{MAX} визначені як максимуми по відповідним координатам у вхідному масиві.

У процесі розробки та тренування моделі, було протестовано декілька підходів. Один із них містив у собі ще один етап додаткової обробки: видалення інформації про траєкторії, що мають значний коефіцієнт кореляції.

«Значна закорельованість» була визначена як така, що має коефіцієнт кореляції вищий за 0.8. Проте, цей підхід погано себе зарекомендував, так як модель мала поганий успіх у передбаченні проїзду по прямій.

2.3 Навчання моделей та їх архітектура

2.3.1 Архітектура нейронної мережі

Загальна архітектура нейронної мережі обиралась з огляду на ефективність, швидкість навчання та робастність моделі. Також бралася за мету можливість створити архітектуру, що буде добре діяти на незнайомих даних і буде добре узагальнювати. Було обрано дві основні архітектури нейромережі, робота яких оцінювалася у даній роботі. Перша архітектура - звичайна feed-forward нейромережа, яка базується на лінійних шарах і в якій дані передаються лише вперед. Нижче приведено опис архітектури нейронної мережі, розглянуто все на прикладі навчання для передбачення траєкторії одного автомобіля:

1) На вхід подаються конкатеновані дані про стан транспортного засобу. Розмірність: 20x7 (20 векторів, розмірності 7, що являють собою стани автомобіля зібрані з частотою 10 Гц).
2) Далі, за допомогою шару Flatten[9], ми стискаємо вхідні дані у один вектор довжини 20x7 = 140.
3) Додаємо 3 шари Dense[9], розмірностей 128, 256, 512, 1024 та 160. Ці шари власне і будуть брати основну участь у навчанні моделі. Більшість чисел є степенями 2, це зроблено для оптимізації обчислень.
4) Тепер стоїть задача дати якийсь вихід з нейромережі. Сформуємо перший вивід, за допомогою шару Reshape[9], який змінить розмірність даних до тривимірного вектору розмірності (4,20,2). Це і будуть 4 можливих траєкторії на наступні 20 дс, або 2 с.
5) Для відтворення ймовірностей траєкторій, використовуємо Dense шар розмірності 4.

6) Проробимо операцію, щоб отримати другий вивід, застосувавши Reshape(4,1,1). Таким чином отримуємо ймовірності відповідних траєкторій з п. 4.

У якості функцій активації нейронів, для шарів Dense було обрано “relu”, яка визначається наступним чином: $f(x) = x^+ = max(0, x)$.

Для останнього шару було обрано “linear”, звичайну лінійну функцію. Це було зроблено для коректного виводу можливої траєкторії, бо можливі і від’ємні значення. Використання, наприклад, “relu” не давало би таких можливих виходів. Для шару, що дає вивід ймовірностей траєкторій застосована функція “softmax”, що є узагальненням логістичної функції.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 20, 7)]	0	[]
flatten (Flatten)	(None, 140)	0	['input_1[0][0]']
dense (Dense)	(None, 128)	18048	['flatten[0][0]']
dense_1 (Dense)	(None, 256)	33024	['dense[0][0]']
dense_2 (Dense)	(None, 512)	131584	['dense_1[0][0]']
dense_3 (Dense)	(None, 1024)	525312	['dense_2[0][0]']
dense_4 (Dense)	(None, 1024)	1049600	['dense_3[0][0]']
dense_5 (Dense)	(None, 512)	524800	['dense_4[0][0]']
dense_6 (Dense)	(None, 160)	82080	['dense_5[0][0]']
dense_7 (Dense)	(None, 4)	644	['dense_6[0][0]']
input_2 (InputLayer)	[(None, 1, 20, 2)]	0	[]
reshape (Reshape)	(None, 4, 20, 2)	0	['dense_6[0][0]']
reshape_1 (Reshape)	(None, 4, 1, 1)	0	['dense_7[0][0]']
loss_func (LossFunc)	(None, 4, None)	0	['input_2[0][0]', 'reshape[0][0]', 'reshape_1[0][0]']
add_loss (AddLoss)	(None, 4, None)	0	['loss_func[0][0]']

Total params: 2,365,092
Trainable params: 2,365,092
Non-trainable params: 0

Рисунок 6 – Опис архітектури нейронної мережі

У якості альтернативної моделі було обрано модель, що базується на LSTM[10] шарах. Як вже згадувалося у цій роботі, ці нейромережі чудово підходять для виявлення якихось послідовних патернів, а саме тому використовуються у ряді задач машинного навчання, таких як обробка природної мови, аналіз часових рядів і також передбачення траєкторій. Так як LSTM cell є одним із основних елементів нейронної мережі і суттєво відрізняє її від аналогів, то варто зупинитися на принципі роботи LSTM шару.

Як вже зазначалося, LSTM - це окремий випадок рекурентних нейромереж. Порівняно зі звичайними рекурентними нейромережами, LSTM мережі є більш робастними для визначення довгих часових трендів (як от наприклад рух автомобіля протягом певного часу).

LSTM розшифровується як Long-Short-Term-Memory, а отже в такому шарі є так звана “пам’ять” (позначення на картинці). Базуючись на нових вхідних даних x_t, попередньому стані m_{t-1} та попередньому виході h_{t-1}, LSTM шар проводить операції, використовуючи так звані “гейти”. Вони бувають трьох типів:

- забуття (forget): використовує інпути для того, щоб вирішити які дані “забути” з попереднього стану m_{t-1}
- вхід (input) вирішує які нові дані зберегти, базуючись на x_t та h_{t-1}
- вивід (output) комбінує новий вивід шару, базуючись на попередніх станах та на виході input гейту.

Добавлено примечание ([1]): Compared to simpler vanilla RNN implementations, LSTMs are generally considered more robust for long time series [20]; future work will focus on comparing the performance of different RNN approaches on our particular dataset.
An interesting feature of LSTM cells is the presence of an internal state which serves as the cell’s memory, denoted by mt in Figure 4. Based on a new input xt, its previous state mt−1 and previous output ht−1, the cell performs different operations using so-called “gates”:
• forget: uses the inputs to decide how much to “forget” from the cell’s previous internal state mt−1;
• input: decides the amount of new information to be stored in memory based on xt and ht−1;
• output: computes the new cell output from a mix of the previous states and output of the input gate.
This particular feature of LSTMs allows a network to learn long-term relations between features, which makes them very powerful for time series prediction.
Due to their recurrent nature, even a single layer of LSTM nodes can be considered as a “deep” neural network.
Although such layers may theoretically be stacked in a

Добавлено примечание ([2R1]): fashion similar to convolutional neural networks to learn higher-level features, previous studies [11] and our own experiments (see Section V) seem to indicate that stacked layers of LSTM do not provide improvements over a single layer in our application

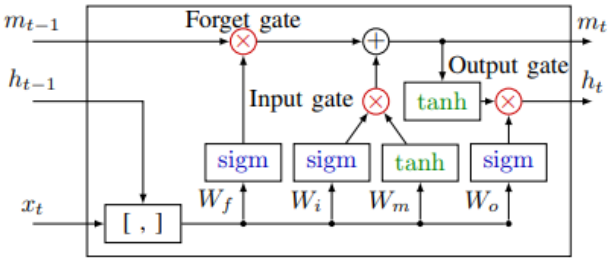

рисунок 7 - LSTM шар

Таким чином, нейромережа має можливість визначати довгострокові тренди у вхідних даних і робити це з адекватною обчислювальною ефективністю, адже шар не накопичує всі дані, а тримає в пам’яті тільки ті, які є актуальними. Через свою рекурсивну природу, один LSTM шар може бути розглянутий як окрема глибока нейронна мережа. Теоретично, такі шари можна нашаровувати один на одний, схоже з тим, як працюють згорткові нейронні мережі. проте попередні досліди показують, що в задачах, подібних на цю, нашарування цих шарів не має переваги над одним єдиним шаром нейромережі.

В цій задачі використовується модель, що комбінує у собі LSTM шар з звичайними лінійними шарами. Проте, цей підхід не є новим і схожу модель використовували в [11]. Структура моделі є наступною:

1) На вхід подається вектор розмірності (20, 7), що містить стани транспортного засобу за 2 с спостереження за ним.
2) Далі йде звичайний шар Dense розмірності 140, що співпадає з розмірністю вхідного вектору.
3) Далі йде шар, що є основним “мозком” моделі, що відповідає за захоплення часових трендів та аналізу вхідних даних як послідовності –

шар LSTM. Варто зазначити, що функція активації зі стандартної “tanh” змінена на “relu”. Це зроблено для консистентності функцій протягом проходження мережі.

4) Схема з виводом аналогічна моделі, представленій вище, яка працює за рахунок Dense шарів.

Вибір функції активації для Dense шарів співпадає із варіантом нейромережі, що має лінійні шари. Варто зазначити, що дана модель має набагато менше параметрів, які можна натренувати. Це має позитивний вплив на швидкість навчання моделі та потенційно може врятувати від перенавчання. Це допоможе моделі краще узагальнювати і реагувати на нові дані.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 20, 7)]	0	[]
dense (Dense)	(None, 20, 140)	1120	['input_1[0][0]']
lstm (LSTM)	(None, 256)	406528	['dense[0][0]']
dense_1 (Dense)	(None, 512)	131584	['lstm[0][0]']
dense_2 (Dense)	(None, 160)	82080	['dense_1[0][0]']
dense_3 (Dense)	(None, 4)	644	['dense_2[0][0]']
input_2 (InputLayer)	[(None, 1, 20, 2)]	0	[]
reshape (Reshape)	(None, 4, 20, 2)	0	['dense_2[0][0]']
reshape_1 (Reshape)	(None, 4, 1, 1)	0	['dense_3[0][0]']
loss_func (LossFunc)	(None, 4, None)	0	['input_2[0][0]', 'reshape[0][0]', 'reshape_1[0][0]']
add_loss (AddLoss)	(None, 4, None)	0	['loss_func[0][0]']

Total params: 621,956
Trainable params: 621,956
Non-trainable params: 0

Рисунок 8 - опис структури моделі

2.3.2 Навчання та функція втрат

Навчання відбувалося з використанням оптимізатора Adam[9], та з показником learning_rate = 10e-3. Вибір такого низького значення цього показника обумовлений тим, що проблема передбачення траєкторії може мати дуже велику кількість можливих рішень. Якби learning_rate був би обраний близьким до одиниці, то була б вкрай високою ймовірність того, що нейронна мережа швидко прийняла б неоптимальне рішення за ідеальне. Проте, в такого значення learning_rate[8] є і свої мінуси – необхідно проводити більшу кількість тренувальних епох.

У якості функції втрат було вирішено використовувати модифікацію до RMSE(Root Mean Squared Error), яка була зважена за ймовірностями.
Наступна формула визначає цю функцію.

$$\sqrt{\frac{\sum_{i=1}^{m} \quad (1 + (y_{gt} - y_{pred}) * (2 - p_i))^2}{N}} \rightarrow \ min$$

В цій формулі, m – кількість передбачених траєкторій, p_i – ймовірність вибору i-тої траєкторії. Визначимо оператор –, який використовується в операції з y_{gt} та y_{pred} як суму всіх різниць відповідних координат відповідних векторів.

Таке визначення функції втрат надає змогу мінімізувати функцію для найбільш ймовірної траєкторії. Неважко побачити з формули, що коефіцієнт $(2 - p_i)$ набуває найменшого значення при найбільшому значенні p_i. Також, це значення мінімізує і добуток, у розрахунку якого бере участь коефіцієнт. Під час розробки, одним із варіантів було використовувати значення 1 замість 2 у формулі. Проте, це було б фатальною помилкою – для найбільш ймовірної траєкторії коефіцієнт би занулявся і ми отримували б нульове значення функції втрат для найбільш ймовірної траєкторії незалежно від реального

значення координат на наступні 3 с. Аналогічна техніка використовується з $y_{gt} - y_{pred}$.

Під час навчання була використана техніка кросс-валідації(cross validation). Ця техніка полягає в випадковому розділенні вхідних даних на тренувальні та тестові у заданій пропорції (в конкретній реалізації в цій роботі було виділено 80% даних у якості тренувальних і 20% у якості тестових). Виконується декілька таких розділень і для кожного з них відбувається окремий процес навчання. Така техніка допомагає зрозуміти на яких даних модель дає неточні передбачення. Більше того, кросс-валідація допомагає уникнути проблеми перенавчання, коли модель занадто сильно «звикає» до навчальних даних і не може адекватно реагувати на появу тестових, яких вона не бачила до цього.

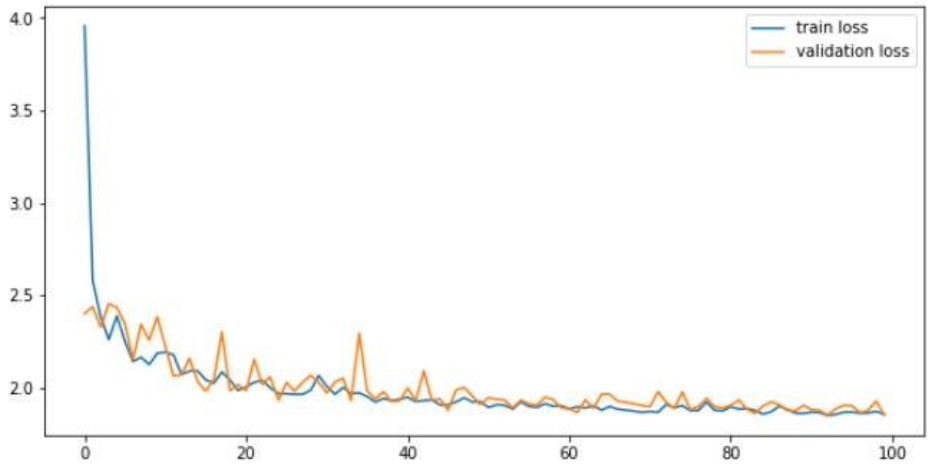

Рисунок 9 – графік, що описує навчання моделі

На рис.4, можемо побачити графік, де по осі Ox відмічені епохи – ітерації процесу навчання моделі, а по осі Oy – значення функції втрат. На

графіку видно дві лінії, блакитна відповідає за значення функції втрат на навчальному наборі даних, а жовта – на тестовому. Проаналізувавши графік, можемо зробити висновок, що архітектура підібрана правильно, так як з епохами знижуються як значення функції втрат на тестовому наборі, так і на наборі навчальному.

Модель з LSTM шаром навчалася за відмінною схемою. Незважаючи на те, що оптимізатором моделі також був Adam, використовувалися різні значення learning rate. Тестові навчання показали, що використання початкового learning rate 10e-3 є небезпечним, адже мережа зводиться до субоптимального рішення, яке є далеким від ідеального. Тож, після навчання протягом 100 епох на початковому learning rate було проведено донавчання з таким самим batch_size але lr=10e-4. Практика показала, що дана зміна була виправданою і цільова функція втрат була зменшена.

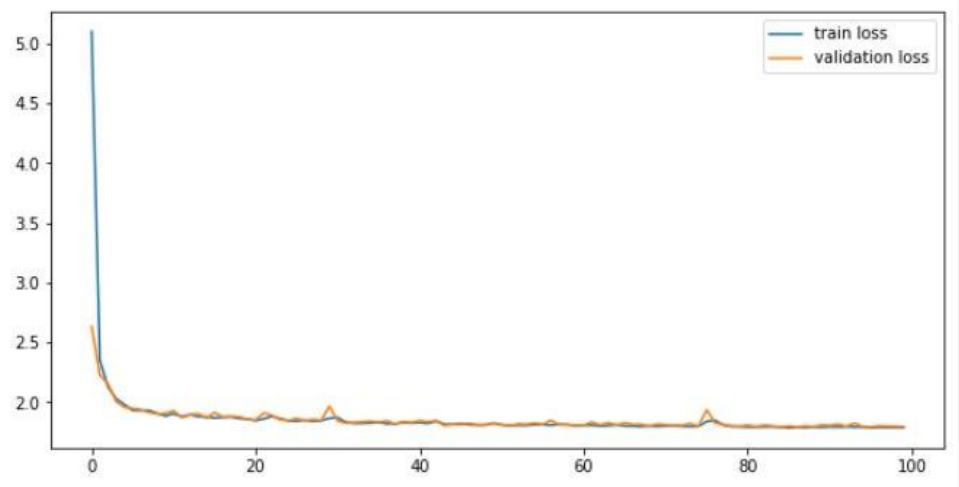

Рисунок 10 - графік навчання моделі 2

З графіка на рисунку видно, що навчання було поступовим і відповідає загальному розумінню того, як виглядає подібний графік. Проте, видно, що падіння функції, починаючи з приблизно 20 епохи було досить повільним і нестабільним, тобто були певні проблеми з пошуком мінімуму. Проте, подальше використання нижчого learning rate виправило цю проблему.

Протягом навчання також вносилися незначні зміни в архітектуру обидвох нейромереж. Так, в першу нейромережу додавався додатковий шар Dense розмірності 2048 між шарами розмірності 1024, а також навпаки викидався один із шарів розмірності 1024. Для другої нейромережі це було додавання ще одного шару LSTM, а також додавання ще одного Dense шару розмірності 1024 перед шаром з розмірністю 512. Ці зміни не мали якогось критичного ефекту і про їх ефективність можна буде судити виходячи з метрик, про які буде йти мова у наступному розділі.

РОЗДІЛ 3.
АНАЛІЗ РЕЗУЛЬТАТІВ

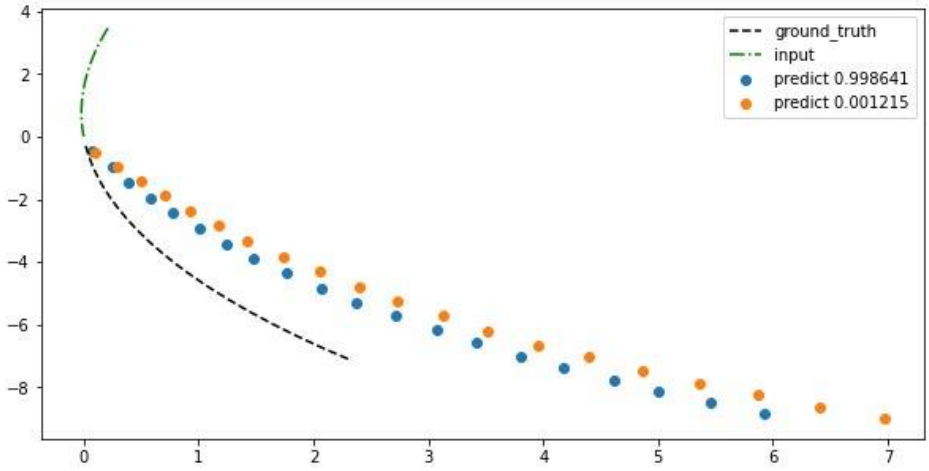

Рисунок 11 – приклад прогнозу нейронної мережі 1

На рис. 11 можна побачити приклад виводу нейронної мережі. Варто зазначити, що з чотирьох можливих траєкторій зображено лише 2, адже інші 2 мають занадто малі ймовірності, щоб вважати їх хоча б мінімально релевантними. Проте, одразу можна побачити недолік нейронної мережі: незважаючи на те, що найбільш ймовірна траєкторія є правдоподібною, в прогнозах мережі мало різноманіття: дві траєкторії проходять поруч одна з одною. З одного боку, це не є критичним, адже траєкторія є близькою до істинної. З іншого боку, це може грати негативну роль при більш неоднозначних сценаріях руху. Проте, як покажуть метрики, це не є проблемою і мережа достойно узагальнює і реагує на незнайомі дані.

На практиці, даний недолік буде згладжуватися тим фактом, що істинна траєкторія має другу найбільшу ймовірність і програмне забезпечення може брати до уваги не лише найбільш ймовірну траєкторію, а й другу за значимістю.

Введемо метрику, за допомогою якої матимемо змогу узагальнити середню точність моделі. Розрахуємо середню похибку найбільш ймовірної траєкторії. Для цього, порахуємо середньоквадратичну похибку, де братимемо різницю між реальною траєкторією на наступні 3 с та тією, яку нейронна мережа вважає найбільш ймовірною.

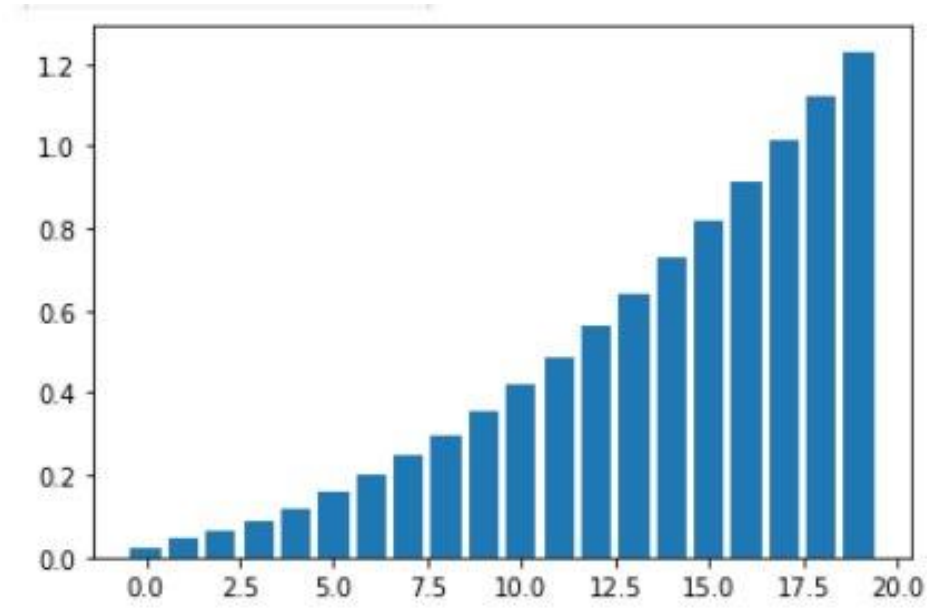

Рисунок 12 – графік похибки найбільш ймовірної траєкторії для моделі 1

На рис. 12 (по осі Ох часові кроки, по осі Оу похибка у метрах) можна побачити результати навчання моделі. Одразу можна виділити, що на перші 5-

10 кроків (0.5-1с) передбачення моделі є досить точним, проте для більш пізніх кроків характерна велика похибка. Це обумовлено досить логічним висновком про те, що траєкторія у час x все менше залежить від позиції у часі x –k , при зростаючому x.

Іншим поясненням такої, досить значної, похибки може бути той факт, що модель не завжди правильно визначає найбільш ймовірну траєкторію (Якщо бути точнішим, то у ~63% випадків модель зі 100% впевненістю визначає правильність траєкторії).

Проаналізуємо точність другої моделі.

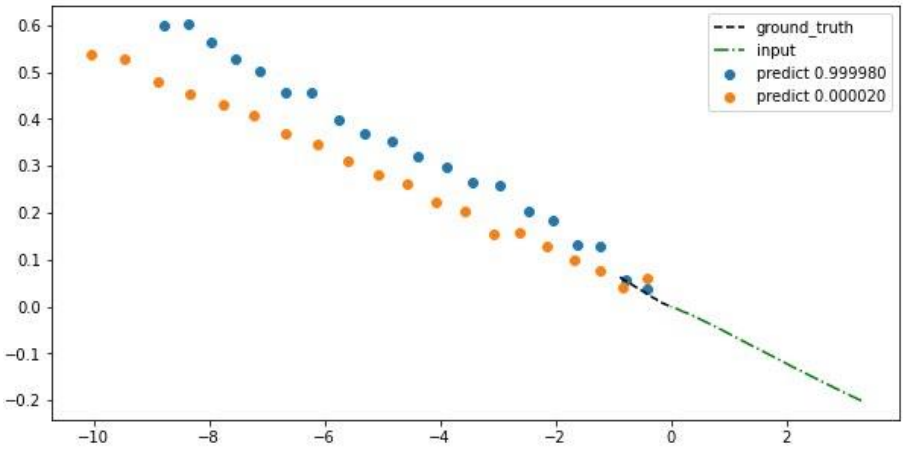

Рисунок 13 - приклад прогнозу нейронної мережі 1

З ймовірностей, можна побачити тенденцію, схожу на попередню модель - алгоритм обирає найбільш ймовірну траєкторію, мінімізує втрати для неї, а іншим надає дуже дуже малу ймовірність, яку зазвичай навіть не варто брати до уваги. Проте, дещо цей алгоритм виконує краще. Обидві траєкторії є реалістичними і є досить близькими до реальної.

Побудуємо графік змінення похибки для найбільш ймовірної траєкторії. Як можна побачити, зростання похибки є експоненційним: на дуже короткий проміжок часу, близько 1 с, проблем з побудовою траєкторії немає, після 1 с прослідковується великий стрибок вгору. Тенденція зберігається, порівняно з попередньою моделлю і є цілком логічною та обґрунтованою.

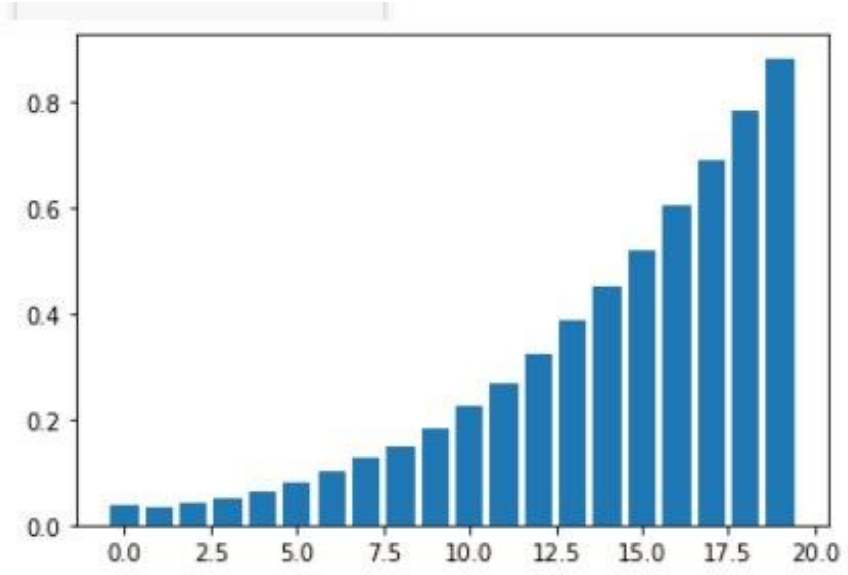

Рисунок 14 – графік похибки найбільш ймовірної траєкторії для моделі 8

Для кожної можливої моделі з якою проводилися дослідження, було розраховано 3 метрики: average RMSE (середнє від середньоквадратичної похибки найбільш ймовірної траєкторії) та відсоток тих передбачень, де найбільш імовірна траєкторія мала найменшу RMSE від істинної. Метрика дистанції була обрана за аналогією з функцією втрат. Ці метрики є Результати та метрики можна побачити у таб. 1

Останньою метрикою була метрика minFDE, яка є однією зі стандартних метрик для передбачення траєкторії. Її можна визначити наступним чином:

$$minFDE = min_{k \in \{1,\dots K\}} \sqrt{(x_T^{gt} - x_T^{pred})^2 + (y_T^{gt} - y_T^{pred})^2}$$

Суть цієї метрики така: знайти розмір мінімального відхилення серед усіх відхилень передбачуваних траєкторій у останній момент часу. Таким чином, метрика показує те, чи є хоча б одна достатньо точна траєкторія. В цій роботі, береться середнє по $minFDE$ для кожної передбаченої траєкторії. Ця метрика має обмежену репрезентативність для моделі в даній роботі, проте вона необхідна для чисельного порівняння моделі з іншими роботами, адже вона є стандартною для такого роду задач.

Мережа	**Середня RMSE**	**Точність пріоритезації траєкторії**	**Середня minFDE**
Повноз'єднана	0.4762	0.6312	0.7535
Повноз'єднана з дод. шаром	0.4953	0.6445	0.7674
Повноз'єднана без 1 шару	0.4899	0.6312	0.7854
LSTM мережа	0.2997	0.7974	0.6275
LSTM з дод. LSTM шаром	0.3031	0.7823	0.6299
LSTM з дод. Dense шаром	0.3154	0.7795	0.6329

Таблиця 1 - метрики навчання моделей

Для порівняння моделі з state-of-the-art рішеннями, візьмемо за приклад роботу [12]. В цій роботі також формулювалася задача побудови траєкторії у наступні моменти часу. У якості T береться значення 3 с. Це є відмінністю від даної роботи, де $T = 2$с. Проте, це єдиний приклад в літературі де на такому самому датасеті вирішувалася задача передбачення траєкторії та використовувалася метрика, яка є актуальною В таб.2 можна побачити порівняння результатів роботи обидвох алгоритмів.

Модель	**minFDE**
LaneGCN[12]	1.87
L-LaneLoss[12]	0.70
L-RouteLoss(ours)[12]	0.69
LSTM модель з даної роботи	0.63

Таблиця 2 - порівняння якості моделі з іншими рішеннями

З таблиці видно, що модель, що запропонована в цій роботі має потенціал. Незважаючи на те, що значення метрики нижче за ті, що представлені в роботі [12], треба розуміти, що горизонт передбачень більший на 1 с може негативно вплинути на значення метрики.

ВИСНОВКИ

В ході виконання проекту було розглянуто питання пов’язані з розробкою алгоритму машинного навчання, що зможе з високою точністю передбачати ймовірності траєкторій руху автомобілів.

Після постановки задачі, було визначено інструменти та математичну базу для вирішення даної проблеми.

Протягом виконання даної роботи було покращено навички роботи з мовою програмування Python та бібліотеками для машинного навчання Numpy та TensorFlow, було поглиблено знання лінійної алгебри та нейронних мереж.

Було розроблено програмний застосунок, що має змогу передбачати траєкторії автомобілів разом із їхніми ймовірностями.

Після виконання даної роботи була досягнута фінальна мета – модель машинного навчання, що може з достатньою точністю вирішувати проблему передбачення траєкторії.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. H. Cui *et al*., "Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks," *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2090-2096, doi: 10.1109/ICRA.2019.8793868.
2. Chandra, Rohan, et al. "Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms." *IEEE Robotics and Automation Letters* 5.3 (2020): 4882-4890.
3. Leon F, Gavrilescu M. A Review of Tracking and Trajectory Prediction Methods for Autonomous Driving. *Mathematics*. 2021; 9(6):660. https://doi.org/10.3390/math9060660
4. Altché, F., & de La Fortelle, A. (2017, October). An LSTM network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)* (pp. 353-359). IEEE.
5. Streubel, Thomas & Hoffmann, Karl. (2014). Prediction of driver intended path at intersections. 134-139. 10.1109/IVS.2014.6856508.
6. Schreier, Matthias & Willert, Volker & Adamy, Jurgen. (2016). An Integrated Approach to Maneuver-Based Trajectory Prediction and Criticality Assessment in Arbitrary Road Environments. IEEE Transactions on Intelligent Transportation Systems. 17. 2751-2766. 10.1109/TITS.2016.2522507.
7. Zhan, W., Sun, L., Wang, D., Shi, H., Clausse, A., Naumann, M., ... & Tomizuka, M. (2019). Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*.

8. Rauch, H.E.; Tung, F.; Striebel, C. T. (August 1965). "Maximum likelihood estimates of linear dynamic systems". *AIAA Journal*. **3** (8): 1445–1450.
9. TensorFlow Keras Documentation[Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/api_docs/python/tf/keras
10. Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.
11. D. J. Phillips, T. A. Wheeler, and M. J. Kochenderfer, “Generalizable Intention Prediction of Human Drivers at Intersections,” 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 1665–1670, 2017.
12. Zhang, Q., Gao, Y., Zhang, Y., Guo, Y., Ding, D., Wang, Y., ... & Zhao, D. (2022). TrajGen: Generating Realistic and Diverse Trajectories with Reactive and Feasible Agent Behaviors for Autonomous Driving. *arXiv preprint arXiv:2203.16792*.