МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

кафедра інформаційних систем
(повна назва кафедри)

ДИПЛОМНА РОБОТА

КЛАСИФІКАЦІЯ 3-D МОДЕЛЕЙ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

Виконав: студент групи ПМі-42
спеціальності 122 – комп'ютерні науки
(шифр і назва спеціальності)

Полюга П. В.
(підпис) (прізвище та ініціали)

Керівник доц. Бернакевич І. Є.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

2023

ЗМІСТ

ВСТУП

У сучасному світі, де розвиток комп'ютерного бачення, інженерії, промисловості та робототехніки набуває все більшого значення, ефективна класифікація різноманітних віртуальних об'єктів є невід'ємною складовою процесу. Від швидкості та точності класифікації залежить успішність вирішення широкого спектру завдань в цих галузях. Саме тут машинне навчання, з його потужними алгоритмами та функціоналом, виявляється незамінним інструментом, який здатен ефективно справлятися зі складністю цієї задачі та знаходити оптимальні рішення для узагальнених сценаріїв.

Проблематика, пов'язана з класифікацією 3-D об'єктів, полягає в складності та розмаїтості цих моделей. Вони можуть мати різні форми, розміри, комплексні структури та різні характеристики. Такі особливості роблять завдання класифікації складним і вимагають спеціалізованих підходів.

Одним із потенційних вирішень цієї проблеми є застосування методів машинного навчання для класифікації 3-D об'єктів. Машинне навчання дозволяє побудувати моделі, які вчаться на основі великого обсягу даних та самостійно знаходять закономірності та шаблони в цих даних. Застосування машинного навчання для класифікації 3-D об'єктів може значно полегшити процес та забезпечити високу точність класифікації навіть у випадках складних структур та об'єктів різних розмірів.

Також, важливим аспектом є розробка застосунків, які дозволяють не лише класифікувати 3-D об'єкти, але й візуалізувати їх для зручного аналізу та сприйняття результатів. Візуалізація дозволяє користувачам легко розпізнавати та інтерпретувати класифіковані об'єкти, що зробить процес роботи з ними більш зручним та ефективним.

Отже, метою дипломної роботи є аналіз, порівняння та реалізація ефективних алгоритмів машинного навчання для класифікації 3-D об'єктів. Дослідження спрямоване на знаходження оптимальних рішень для проблем,

пов'язаних з класифікацією 3-D об'єктів, та розробку застосунку, що поєднує класифікацію та візуалізацію для полегшення роботи з ними.

Звіт з дипломної роботи містить сім розділів з детальним описом вирішення завдання, 9 скріншотів демонстрації системи десктопного застосунку, вступ, висновки та 25 використаних джерел.

1 ПОСТАНОВКА ЗАДАЧІ

Взявши до уваги тематику класифікації 3-D моделей було розроблено наступний план реалізації проекту машинного навчання, який включає наступні етапи:

а) ознайомлення з предметною областю і визначення цілей проекту;

б) вибір відповідних наборів даних, які відповідають постановці проблеми та містять достатньо даних для навчання та тестування. Розподіл даних між тренувальною та тестувальною підмножинами;

в) порівняння та вибір відповідних алгоритмів або моделей машинного навчання, які добре підходять для вирішення завдання та ефективно працюють із запропонованими наборами даних;

г) селекція відповідних технологій та інструментів для імплементації моделей машинного навчання, враховуючи такі фактори, як мови програмування, фреймворки та бібліотеки;

д) реалізація вибраних моделей із забезпеченням належної інтеграції та функціональності;

е) навчання моделей за допомогою тренувального набору даних і оцінка їхньої продуктивності за допомогою тестової вибірки, повторюючи процес для оптимізації налаштувань;

є) порівняння результатів, отриманих від різних моделей, щоб визначити найефективніший підхід, враховуючи такі показники, як точність, правильність і неправильність прогнозів на кожну категорію вибірки;

ж) для зручного подання виконаної роботи, створення десктопного застосунку, який демонструє вибраний об’єкт і класифікує його, надаючи інтуїтивно зрозумілий і зручний інтерфейс для взаємодії та візуалізації.

Дотримуючись цих кроків, було ефективно вирішено проблему та по кроках побудовано необхідний функціонал, який є підсумований програмою для класифікації та відображення 3-D об’єктів.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

Машинне навчання [1] широко використовується в різних типах застосувань для вирішення завдань, які можуть бути складними для точного формулювання або для проблем, де створення алгоритму явно вимагає багато зусиль. За допомогою цього підходу певна робота може бути виконана самонавчальним алгоритмом, особливо коли потрібно обробити велику кількість даних.

Метою навчального алгоритму є знаходження функції $f: X \to Y$ з найнижчою ймовірністю помилки з заданого класу функцій, яка відображає вхідні дані X на прогноз Y.

2.1 Види машинного навчання

Існують 3 основні підходи до машинного навчання:

• навчання з учителем - для навчання алгоритмів використовуються лише позначені дані (x, y), де x - це певний елемент даних, а y - це мітка, що описує x. Завдання функції f полягає у прогнозуванні мітки вхідних даних, мінімізуючи помилку на основі прикладів, які були побачені функцією раніше. Основними областями застосування є класифікація (визначення категорії для вхідного елемента) і регресія (прогнозування невідомого значення на основі реальних даних);

• навчання без учителя - з урахуванням схожості між об'єктами без будь-яких міток, оцінюється форма даних і на основі спільних ознак робиться прогноз. Часто використовується для кластеризації (групування схожих даних) і асоціації (виявлення правил асоціації, що описують деякі частини даних);

• навчання з підкріпленням - коли лише невелика частина вхідних даних позначена. Проблема перебуває на межі між навчанням з учителем та навчанням без учителя. Цей підхід актуальний у випадку, коли створення позначок для всіх даних є складною або дорогою задачею. Структура даних може бути вивчена за допомогою підходу навчання без учителя, а потім, якщо деякі дані з

новостворених кластерів мають позначки, використовуються ці мітки для навчання за допомогою методу навчання з учителем, і в кінці передбачають вихід для невідомих прикладів.

Всі ці підходи машинного навчання використовуються майже у всіх галузях технологій, і їх вибір залежить від доступних вхідних даних та типу проблеми, що потребує вирішення.

2.2 Нейронні мережі

Штучні нейронні мережі (ШНМ) [2] є напрочуд схожими до людського мозку, де набір нейронів працює разом для сприйняття інформації. Нейронні мережі працюють шляхом перетворення вхідної інформації на набір вихідних сигналів. Це перетворення відбувається за допомогою зміни внутрішнього стану мережі. Прямі зв'язки передають інформацію від входу до виходу, тоді як зворотні зв'язки передають інформацію від виходу до входу.

Нейрони, що утворюють нейронну мережу, є простими процесорами. Обчислювальні параметри цих нейронів обмежені правилами комбінування вхідних сигналів і функцією активації, яка визначає вихідний сигнал на основі вхідних сигналів. Інші нейронні мережі можуть отримувати вихідний сигнал нейрону через синаптичні (зважені) зв'язки. Кожному зв'язку відповідає ваговий коефіцієнт, який має назву – вага зв'язку.

Вхідні нейрони у нейронній мережі відповідають за прийом інформації з зовнішнього середовища. Вони приймають вхідні сигнали та передають їх усередину мережі для подальшої обробки. Зворотнім процесом є передача інформації від вихідних нейронів до зовнішнього середовища. Вихідні нейрони збирають сигнали від інших нейронів у мережі та генерують вихідні сигнали, які передаються зовнішньому середовищу для виконання певних дій або подальшого використання.

Перші дослідження були проведені Ворреном МакКаллохом і Волтером Піттсом у 1943 році, а створення персептрона Френком Розенблаттом в 1957 році.

Розглянемо ретельніше модель нейрона МакКаллоха-Піттса [3] (рисунок 2.1) та опишемо принцип його роботи.

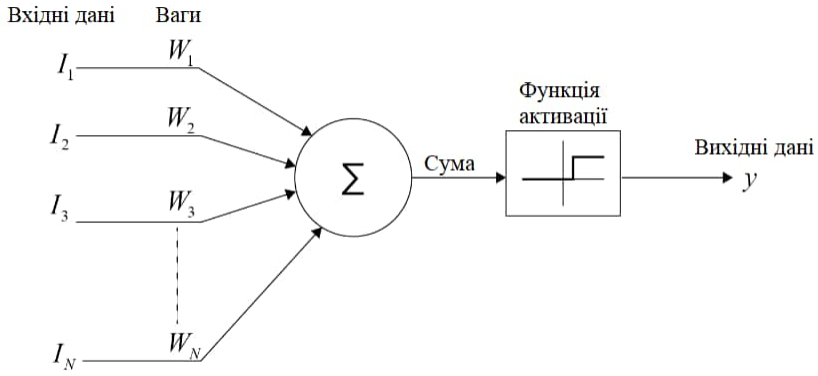


Рис. 2.1 Нейронна модель МакКаллоха-Піттса

Модель приймає N вхідних сигналів, кожен з яких має відповідну вагу. Лінійна комбінація всіх цих вхідних сигналів разом з їхніми вагами обчислюється та агрегується, після чого результат передається до лінійної функції активації ϕ, яка повертає вихідні дані мережі. Рівняння (2.1), показує, як вони обчислюються.

Відомо, що

$$y = \phi(\sum_{n=0}^{N} W_n \times I_n) \quad (2.1)$$

де ϕ – функція активації, N – кількість вхідних сигналів, W_n – матриця ваг зв’язку, I_n – матриця значень елементів.

Ваги мережі підлаштовуються та вибираються з метою отримання кращого виходу від моделі.

2.3 Багатошаровий персептрон

Багатошаровий персептрон [4] є ациклічною нейронною мережею, де нейрони розташовані в послідовних шарах, починаючи з вхідного шару, прихованими шарами посередині і закінчуючи вихідним шаром.

Цей алгоритм навчання з учителем розглядає функцію $f: R^m \rightarrow R^o$, де m - розмір вхідного вектора, а o - розмір вихідного вектора. Він може бути використаний для задач класифікації або регресії. На рисунку 2.2 показано багатошаровий персептрон, де вхідний і приховані шари повністю з'єднані, і кожне з'єднання має свою вагу. Вхідний шар просто отримує дані, а обчислення

виконуються в прихованому шарі, де дані множаться на вагу ребра, яке використовується. Потім вихідний шар виконує активацію і дає на виході результат.

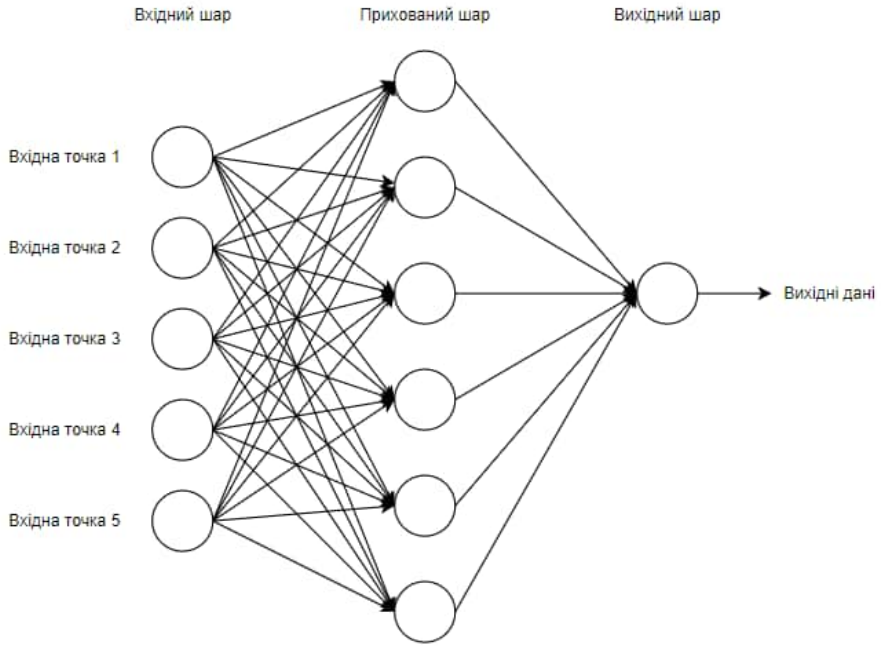


Рис. 2.2 Будова багатошарового персептрона

До переваг багатошарового персептрона можна віднести:

• можливість працювати з нелінійними задачами;

• навчання в реальному часі.

З недоліків цієї моделі можна виділити:

• необхідність налаштування гіперпараметрів (кількість нейронів, розміри, шари);

• зміни в трансформації ознак можуть призвести до погіршення якості прогнозування.

2.4 Зворотне поширення

Мережа багатошарового персептрона може мати складну структуру з кількома прихованими шарами, і може бути складним вибрати правильні ваги для моделі, щоб отримати розумний вихід. В такому випадку алгоритм зворотного поширення помилок [5] може вирішити проблему. Ваги обчислюються і

оновлюються протягом кількох ітерацій, шукаючи глобальний мінімум помилки за допомогою алгоритму градієнтного спуску.

Алгоритм зворотного поширення помилок включає кілька кроків:

• При призначенні ваг з'єднань певними початковими значеннями, виконується прямий прохід по мережі. З використанням функції втрат, отриманий вихід порівнюється з міткою вхідного об'єкта. Виконується розрахунок градієнта функції втрат.

• Градієнт відправляється назад через мережу для обчислення помилок для кожного прихованого шару.

• Після отримання всіх значень ваги оновлюються за допомогою рівняння (2.2), де W_x^* - оновлена вага, W_x- старе значення ваги, $\partial\, error$ - градієнт помилки, а α - швидкість навчання. Швидкість навчання може бути статичною протягом усього процесу або адаптивною (змінюватися між ітераціями / епохами).

• Процес повторюється, поки не досягнута збіжність ваг.

$$W_x^* = W_x - \alpha(\frac{\partial\, error}{W_x}) \qquad (2.2)$$

Існують 3 підходи до виконання описаного алгоритму. Перший підхід називається онлайн-навчанням, де ваги змінюються після кожного вхідного сигналу до мережі, і ми вже бачили цей метод у попередньому прикладі. При розгляді одного вхідного сигналу одночасно обчислений градієнт може змінюватися через деякі особливості даних, і мінімум не буде досягнутий безпосередньо. Другий підхід полягає в використанні повного набору даних об'єктів для подачі їх на мережу, і ваги оновлюються після того, як всі вхідні дані пройшли через мережу. Цей метод має деякі недоліки, оскільки набір даних може бути дуже великим, а також має високу складність обчислень. Третій підхід, який є більш поширеним, полягає в розбитті даних на невеликі пакети (наприклад,

16/32/64 об'єкти на пакет) і коригуванні ваг після проходження кожного пакета через мережу.

2.5 Згорткові нейронні мережі (CNN)

Згорткові нейронні мережі (CNN) [6] — це тип моделей глибокого навчання, які довели свою високу ефективність у різних завданнях комп’ютерного зору, включаючи класифікацію 3D-моделей. CNN особливо добре підходять для обробки структурованих сітчастих даних, таких як зображення, завдяки їхній здатності фіксувати локальні моделі та ієрархічні особливості.

При застосуванні CNN до класифікації 3D-моделей моделі зазвичай представлені як об’ємні дані або хмари точок. Об’ємні дані представляють тривимірну форму як звичайну сітку вокселів, де кожен воксель представляє невеликий об’ємний елемент. У той час, як хмари точок складаються з набору тривимірних точок, які представляють поверхню об’єкта.

CNN складаються з кількох рівнів (рис. 2.3), у тому числі згорткових. Згорткові шари застосовують набір фільтрів до вхідних даних і обчислюють скалярний добуток для захоплення локальних моделей.

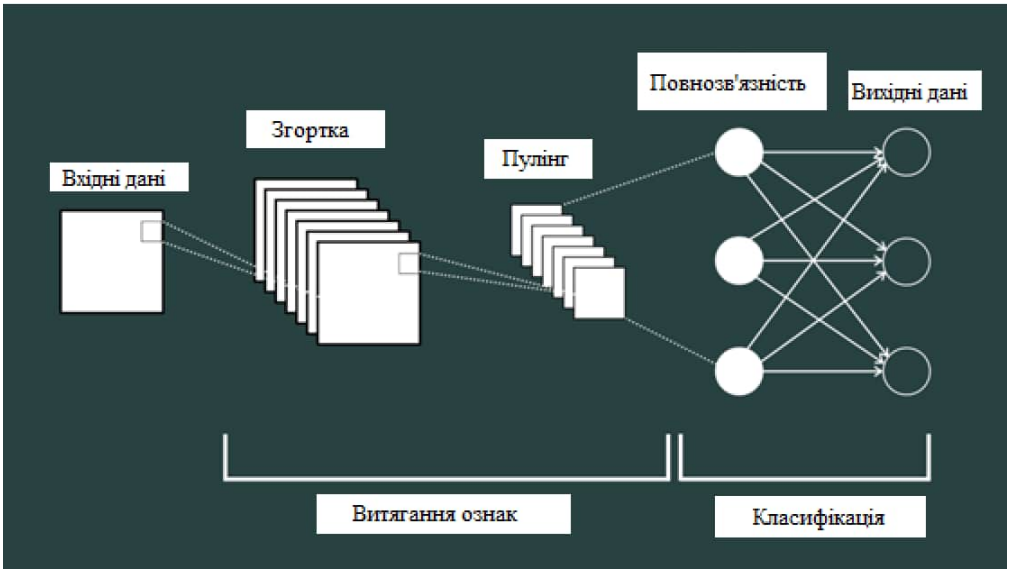


Рис. 2.3 Принцип згорткової нейронної мережі [6]

Після згорткових шарів часто розміщується об’єднання шарів, щоб зменшити просторову розмірність об’єктів. Операції об’єднання, наприклад макс

пулінг, зменшують дискретизацію карт функцій, зберігаючи найважливішу інформацію, одночасно понижуючи обчислювальну складність.

Далі отримані об’єкти вирівнюються та подаються в один або кілька повністю з’єднаних шарів. Ці шари подібні до тих, які є в традиційних штучних нейронних мережах, і відповідають за прогнозування на основі вилучених ознак.

Останнім рівнем CNN є вихідний рівень, який зазвичай використовує активацію softmax для багатокласової класифікації. Він створює розподіл ймовірностей за класами, вказуючи ймовірність належності 3D-моделі до кожного класу.

2.6 Файлові формати 3-D об’єктів

Існують різноманітні формати для зберігання і представлення тривимірних об'єктів, які можуть бути специфічними для різних галузей (3D-друк, ігри, кіно, архітектура, медична допомога, інженерія, моделювання). Об'єкт може бути збережений у вигляді звичайного тексту або у бінарному представленні і може містити інформацію про колір, текстуру, анімацію. Деякі формати можуть бути розроблені і використовуватися специфічно для певного програмного забезпечення.

У роботі було взято до уваги розширення файлу ".off", яке часто використовується в програмах комп’ютерної графіки та 3D-моделювання. Цей формат застосовується для зберігання даних 3D-об’єктів, включаючи позиції вершин і з’єднання граней.

Структура файлу ".off" [7] відповідає особливому формату. Зазвичай файл складається з рядка-заголовка, за яким слідують рядки даних. Рядок-заголовка вказує формат файлу та надає інформацію про кількість вершин, граней і ребер в об’єкті. Потім рядки даних містять координати кожної вершини та інформацію про зв’язок для кожної грані.

Файли у форматі OFF не містять жодної інформації про колір чи текстуру, а просто представляють геометрію об'єкта. Його можна легко проаналізувати та обробити різними програмними бібліотеками та інструментами. Крім того,

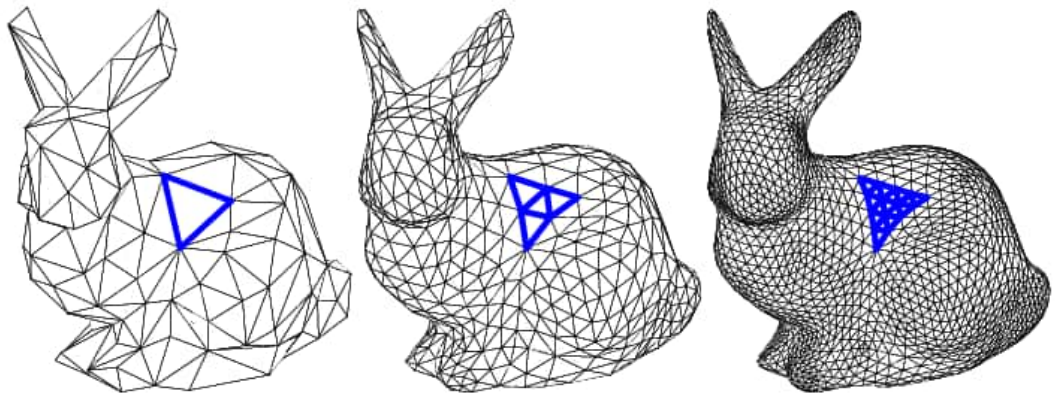
оскільки це формат звичайного тексту, його можна редагувати за допомогою простого текстового редактора, якщо потрібно.

3 НАЯВНІ ПІДХОДИ КЛАСИФІКАЦІЇ 3-D ОБ’ЄКТІВ

3.1 Хмари точок

Одним зі способів створення 3D-моделі є використання 3D-сіток [8] (складених з полігонів) як структурних елементів. Висота, ширина та глибина всередині сітки описуються осями X, Y, Z. Залежно від складності 3D-об'єкта, для забезпечення гладкості та реалістичного вигляду використовується велика кількість сіток. Зазвичай полігони (грані) мають форму трикутників або чотирикутників, які з'єднані лініями (ребрами) і де кожна вершина має свої координати (X, Y, Z). Кілька сіток можуть мати одну або більше спільних вершин та ліній, якщо вони є сусідніми у 3D-моделі.

На рисунку 3.1 показано, як сітки структуровані для утворення об'єкта, тому для отримання більш реалістичного та деталізованого вигляду моделі потрібно використовувати більше сіток.



а) 280 граней б) 1100 граней в) 4480 граней

Рис. 3.1 3-D представлення об'єкта трикутними сітками на прикладі кролика.

Точки, отримані з цих сіток, утворюють хмару точок, як на рисунку 3.2. Тут, залежно від щільності та кількості точок, що містяться у об'єкті, навіть без ребер, що з'єднують точки, ми легко можемо сказати, який об'єкт ми спостерігаємо.

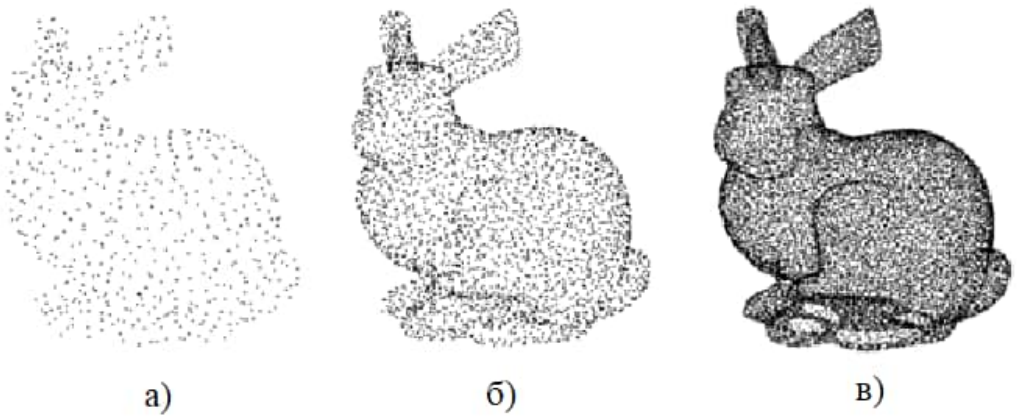


Рис. 3.2 3-D представлення об'єкта хмарами точок на прикладі кролика

Хмари точок зберігають певні властивості, які важливо розуміти під час їхнього використання. Точки не мають визначеного порядку у своєму представленні, і, якщо точки походять від реальних об'єктів, вони знаходяться в евклідовому просторі та повинні зберігати деяку відстаневу метрику, яка може розкодувати локальні або глобальні особливості об'єкта. Важливою властивістю хмар точок є те, що виконання таких перетворень, як зсув, обертання, масштабування для всіх точок разом не змінить глобального вигляду або класу об'єкта.

3.2 Об’ємні техніки

Об'ємні підходи [9] базуються на сітці зайнятості, яка представляє форми у тривимірному просторі. На рисунку 3.3 показано можливості представлення об'єкта (кролика) у оригінальному форматі та різних роздільних здатностях вокселів (об'ємних пікселів) від загального перегляду та менш включених частин до деталізованого об'єкта з великою кількістю вокселів.

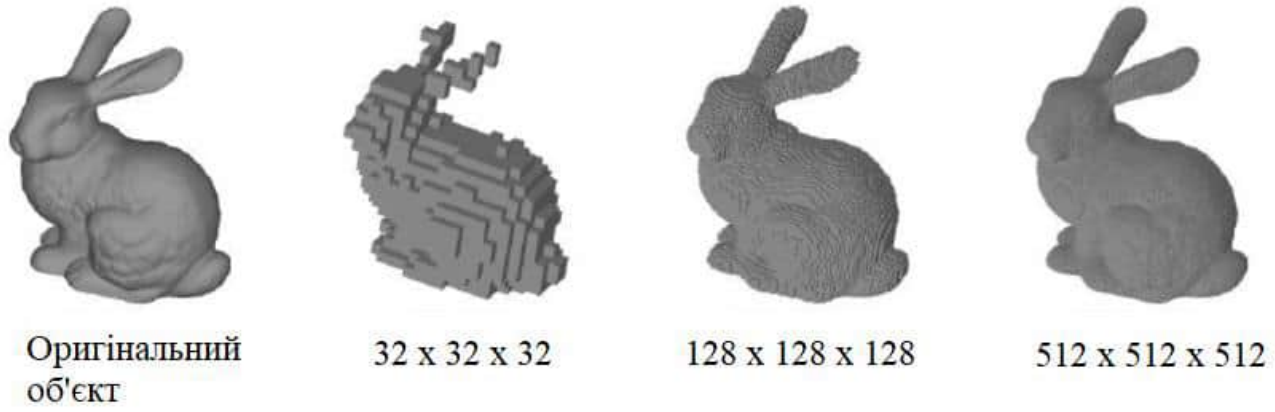


Рис. 3.3 Відображення кролика різними роздільними здатностями вокселів.

Усі точки 3D-об'єкта з декартових координат відображаються у воксельних координатах (i, j, k) в сітці зайнятості, і результат залежить від трьох властивостей:

• орієнтація - об'єкти повинні бути вирівняні навколо певної осі. У деяких випадках потрібно знайти правильну орієнтацію за допомогою додаткових технік, а процес може включати нетривіальні ситуації та складні алгоритми;

• походження об'єкта - його можна взяти з хмар точок або зі сканування реального об'єкта;

• роздільна здатність - розмір вокселя.

Входом згорткової нейронної мережі (CNN), що складається з кількох шарів, є сітка зайнятості розміром 32x32x32. Після шару пулінгу відбувається заміна накладних блоків вокселів їхнім максимумом. Нейрони повнозв'язаних шарів формують лінійні комбінації даних з попередніх шарів, і потім мережа надає k оцінок, що відповідають k класам об'єктів. Увесь архітектурний план показано на рисунку 3.4.

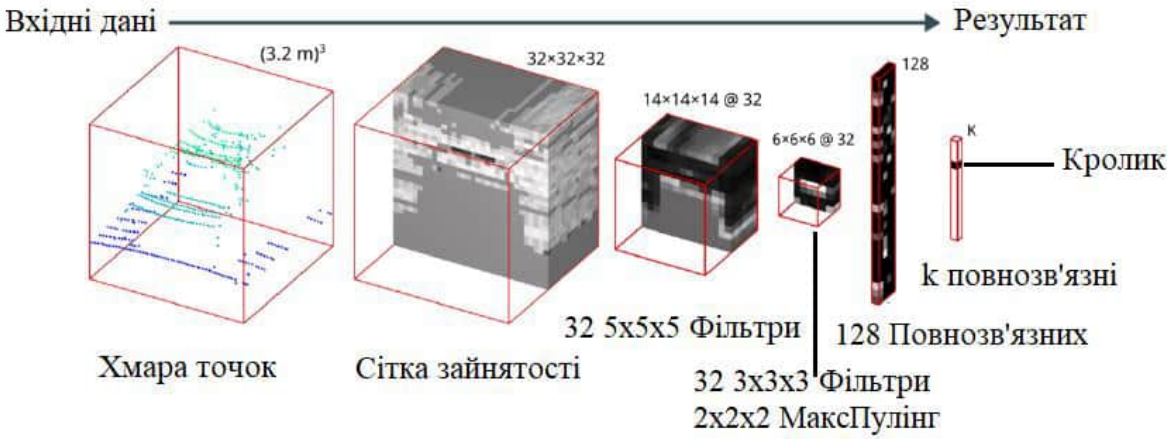


Рис. 3.4 Класифікація воксельної фігури за допомогою CNN [9]

Об'ємний підхід класифікації 3D-об'єктів за допомогою нейронних мереж вважається відносно старим. Це представлення є обчислювально витратним і вимагає великої пам'яті для здійснення 3D-згортки через розрідженість даних, оскільки об'єкт у сітці зайнятості повинен бути представлений разом із порожнім простором.

3.3 Обертові методи двовимірної перспективи

3D-об'єкти можуть бути перетворені на 2D зображення, які представляють перегляд з різних кутів, і потім використовуватись як вхід для класифікації згорткових нейронних мереж (CNN).

На рисунку 3.5 зображено типовий робочий процес підходу 2D перспективи [10], де використовується кілька згорткових нейронних мереж (окрема мережа для кожного відтвореного зображення), а результат цієї стадії агрегується за допомогою вигляду-пулінгу.

Під час вигляду-пулінгу з кількох згорткових нейронних мереж виникає стадія обмеження пропускної здатності, яка є незручною та неефективною через потребу обчислення міри подібності у формі відстані між кожною проекцією всіх вхідних об'єктів. Іншими словами, відбувається обчислення взаємної подібності для двох 3D-об'єктів.

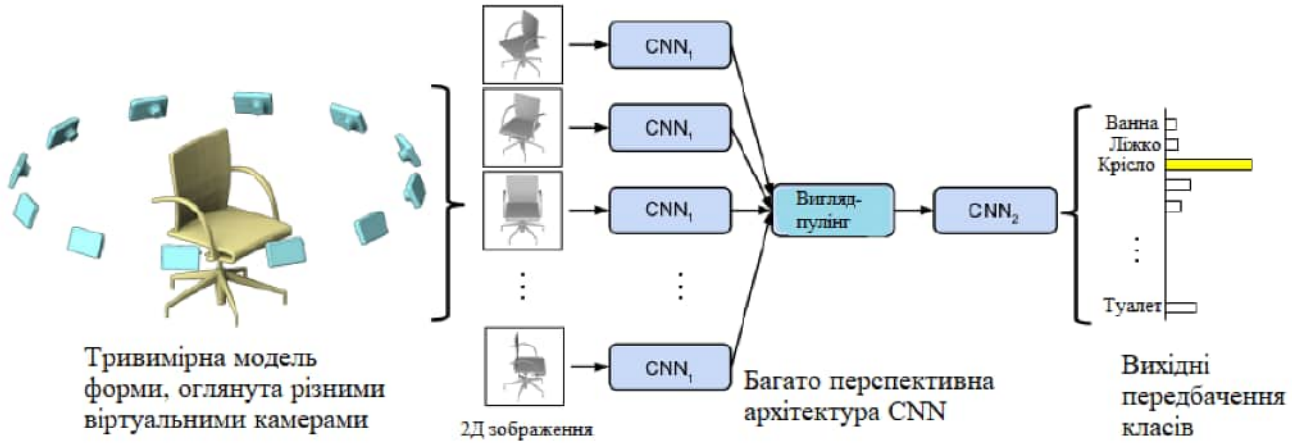


Рис. 3.5 Двовимірна багато перспективна архітектура CNN [10]

Важлива робота повинна бути проведена для оптимізації виконання. Потім використовується інша згорткова нейронна мережа, в якій обчислюється прогнозоване оцінювання для кожної категорії об'єктів. Підхід може мати дуже перспективні результати та продуктивність для класифікації форм. Однак він може включати в собі повільне відтворення вигляду, велику витрату пам'яті та простору, а також складність розширення класифікатора для нових класів. Зрештою, чим більше зображень беруться, тим краща якість прогнозування. Проте потрібно знайти компроміс між бажаним результатом та складністю прогнозування.

3.4 Вибір алгоритмів машинного навчання для імплементації

Сучасні методи класифікації 3D-форм часто використовують дані хмар точок, і одним таких методів є модель PointNet. PointNet є підходом, який спочатку ідентично та незалежно обробляє точки вхідної хмари, що описує 3D-об'єкт. Він використовує згорткові шари, щоб мережа могла виявити цікаві особливості та значущі точки об'єкта. Крім PointNet, варто також згадати про PointNet++, ще один прогресивний метод, який покращує точність класифікації та семантичного розпізнавання об'єктів у 3D вигляді.

3.4.1 PointNet

PointNet [11] є архітектурою (рис 3.6), яка приймає хмари точок безпосередньо на вхід і повертає класові мітки для всього вводу. Базова архітектура мережі вкрай проста, оскільки на початкових етапах кожна точка обробляється однаково і незалежно. Кожна точка 3D-об'єкта представлена лише своїми трьома координатами (x, y, z).

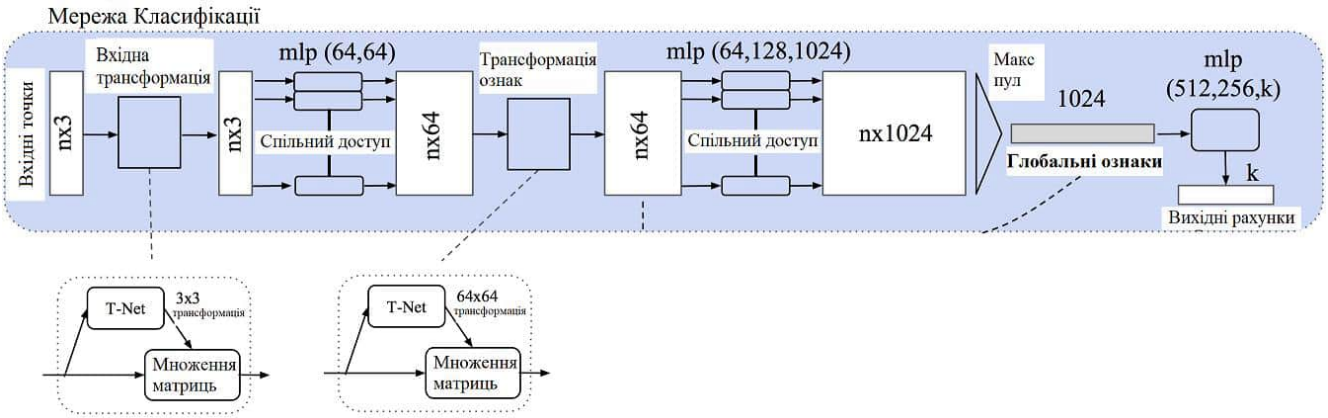


Рис. 3.6 Архітектура мережі PointNet [11]

Мережа класифікації приймає n точок на вході, застосовує функції перетворення для вхідних рис і потім агрегує точки за допомогою максимального пулінгу. Результатом є класифікаційні оцінки для k класів.

Основним елементом цього підходу є використання симетричної функції максимального пулінгу. Мережа ефективно навчається за допомогою оптимізаційних функцій/критеріїв [12], які вибирають інформативні точки у хмарі точок та кодують причини їх вибору. Завершальні повнозв'язні шари мережі агрегують ці оптимальні значення, що були навчені для всієї форми, у глобальний опис.

3.4.2 PointNet++

PointNet++ [13] є розширенням архітектури PointNet, яка призначена для обробки хмар точок, типу представлення даних, що використовується в задачах тривимірного сприйняття.

Оскільки технологія PointNet мала обмеження в обробці локальних геометричних візерунків і варіації масштабу через свою глобальну операцію об’єднання, було запропоновано його оновлену версію PointNet++. Ця модель представляє ієрархічну архітектуру нейронної мережі, яка охоплює як локальні, так і глобальні характеристики хмар точок. Вона організовує точки в ієрархічну структуру за допомогою стратегії просторового поділу, такої як k-d дерево.

Архітектура PointNet++, зображена на рисунку 3.7, складається з кількох рівнів, які обробляють інші рівні ієрархії хмари точок. PointNet++ виконує вилучення функцій на кожному рівні, використовуючи спільні багаторівневі персептрони (MLP) і локальну та глобальну контекстну інформацію. Він використовує пропуск з’єднань і операції агрегації, щоб об’єднати функції з різних рівнів, дозволяючи моделі ефективно отримувати локальну та глобальну інформацію.

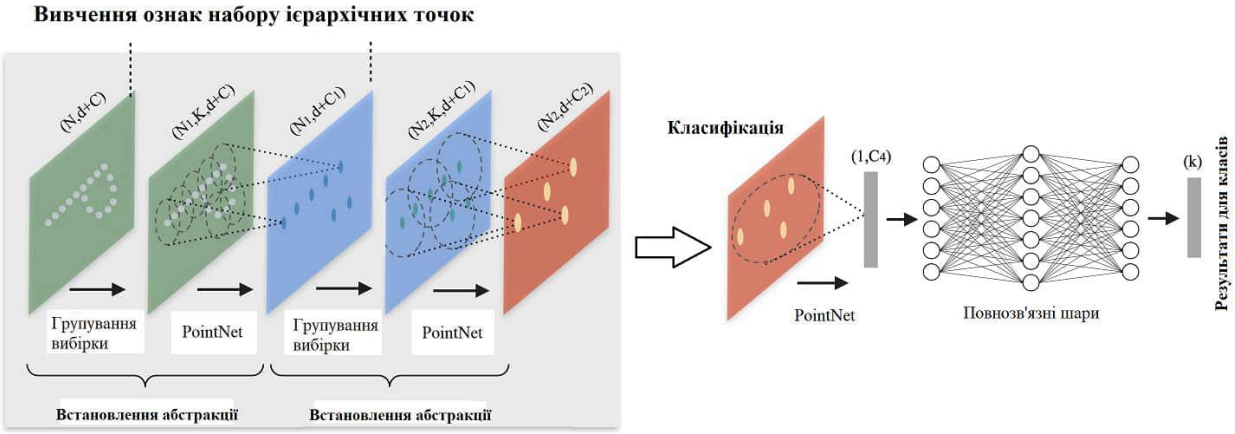


Рис. 3.7 Архітектура мережі PointNet++ [13]

Порівняно з PointNet, PointNet++ продемонстрував покращену продуктивність у різних завданнях тривимірного сприйняття, таких як класифікація об’єктів. Використовуючи ієрархічну архітектуру та контекстну інформацію, PointNet++ краще оснащений для роботи зі складними 3D-структурами, локальними шаблонами та варіаціями масштабу, що робить його більш потужним інструментом для аналізу та розуміння хмари точок.

4 ВХІДНІ НАБОРИ ДАНИХ

4.1 Набір даних ModelNet10

Метою проекту Princeton [14] ModelNet є надання дослідникам у галузі комп'ютерного зору, комп'ютерної графіки, робототехніки та когнітивної науки, всеосяжної та чистої колекції тривимірних CAD-моделей об'єктів.

Для створення основи набору даних було складено список найпоширеніших категорій об'єктів у світі, використовуючи статистику, отриману з бази даних SUN [15]. Після встановлення словника об'єктів було зібрано 3D CAD-моделі, що належать до кожної категорії об'єктів, використовуючи онлайн-пошукові системи та запити для кожної категорії об'єктів. Потім були найняті людські робітники на платформі Amazon Mechanical Turk для ручного визначення того, чи належить кожна CAD-модель до визначених категорій, використовуючи внутрішньо розроблене інструментарій з контролем якості. Для отримання дуже чистого набору даних було вибрано 10 популярних категорій об'єктів (рис 4.1), а неможливі моделі, що не належали до цих категорій, були видалені вручну.

Рис. 4.1 Зображення об’єктів ModelNet10

Набір даних ModelNet10 є підмножиною ModelNet40 і містить 4 899 3-D форм з 10 категорій. Окрім цього, є присутні 3 991 (80%) форми для навчання та

908 (20%) форм для тестування. CAD-моделі знаходяться у форматі Object File Format (OFF), про який згадується в розділі 2.

3.2 Набір даних TraceParts

Для порівняння роботи моделей на інших наборах даних, що відрізняється від описаного в розділі 4.1, було прийнято рішення знайти 3-D об’єкти, які значно відрізняються від ModelNet10, і сферою застосування, і розмірами, і виглядом.

На веб-платформі TraceParts [16], яка є одним з провідних постачальників 3D-цифрового контенту для інженерії, було підібрано каталог з 6 класів стандартно сертифікованих об'єктів для тренування моделі. Серед цих класів були гайки, о-кільця, шпильки, гвинти, квадратні ключі та прокладки, загальна кількість різних об'єктів становила 11150 з різними розмірами. На рисунку 4.1 зображено типові приклади кожного класу.

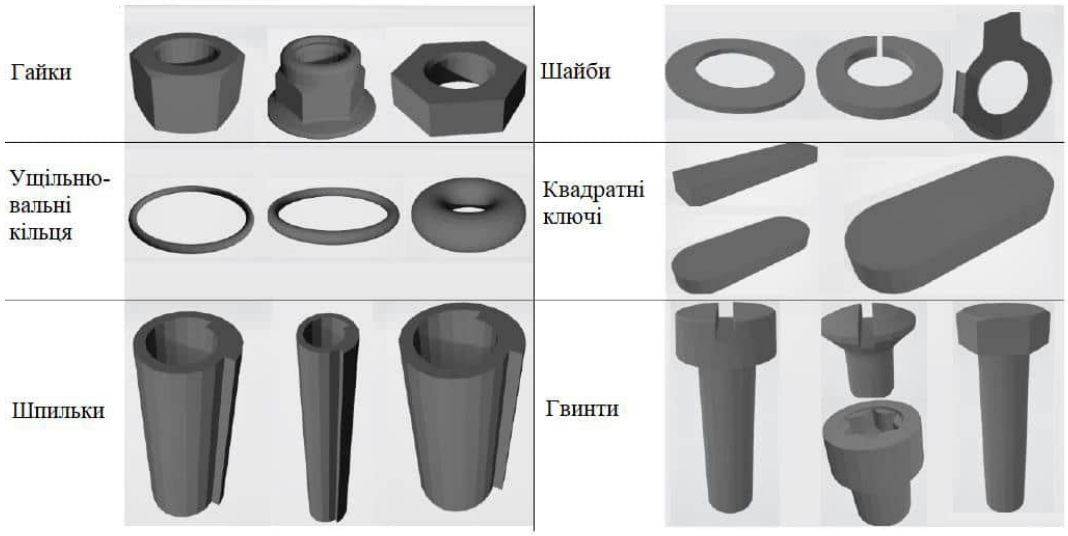


Рис. 4.2 Представлення 6-ти класів набору даних TraceParts

Елементи набору даних зберігаються у файлах з розширенням OFF. У таблиці 4.1 наведена кількість об’єктів по кожному з класів.

Таблиця 4.1 Кількість об’єктів кожної з категорії набору даних TraceParts

Назва предмету	Кількість
Гайки	414
Ущільнювальні кільця	383
Шпильки	4730
Шайби	820
Квадратні ключі	580
Гвинти	4215

По таблиці можна сказати, що найбільшу кількість об’єктів представляють категорії “Шпильок” та “Гвинтів”, а найменшу демонструють “Гайки” та “Ущільнювальні кільця”.

5 ОСОБЛИВОСТІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В цьому розділі буде описано за допомогою якого стеку технологій та як розроблялись початкові етапи машинного навчання вибраних моделей.

5.1 Вибір технологій

Перед тим як починати програмну реалізацію, було ретельно вивчено область необхідних інструментів і вибрано найкращі з них.

5.1.1 Jupyter Notebook

Jupyter Notebook [17] дозволяє писати та виконувати код в окремих клітинках. Ця модульність дозволяє легко повторно використовувати та змінювати фрагменти коду, що особливо корисно під час розробки складних моделей машинного навчання з кількома компонентами та ітераціями.

Технологія полегшує розробку та ітераційне вдосконалення моделей машинного навчання. Існує можливість написання коду для навчання та оцінки моделей, налаштування гіперпараметрів і порівняння різних підходів — і все це, маючи миттєвий доступ до показників ефективності моделі та візуалізацій.

Jupyter Notebook надає інтерактивне та гнучке середовище для розробки моделей машинного навчання. Його інтеграція з різними бібліотеками, підтримка документації, можливості візуалізації даних і функції спільної роботи роблять його цінним інструментом для дослідження даних, розробки моделей і експериментів у сфері машинного навчання.

5.1.2 PyTorch

PyTorch [18] - це відкрита бібліотека глибокого навчання, відома своєю гнучкістю та легкістю використання. Вона популярна серед розробників мови програмування Python, оскільки написана саме на Python і використовує його імперативний режим виконання. PyTorch підтримує використання процесора, графічного прискорювача (GPU) і паралельну обробку даних, що дозволяє

розподіляти обчислювальну роботу між кількома ядрами процесора та графічними прискорювачами. Він також підтримує динамічні обчислювальні графи, що надає гнучкість у зміні поведінки мережі під час виконання. PyTorch має широкий набір інструментів і бібліотек для різних областей, а його сильні сторони полягають у швидкому прототипуванні та реалізації менших проектів, а також у зручності використання та гнучкості.

5.1.3 Azure ML

Під час етапу тренування з великою кількістю даних, на локальному комп’ютері виникли складнощі з недостатньою кількістю оперативної пам’яті та потужністю центрального процесора. Тому було взято до уваги інструмент Azure Machine Learning Studio [19], який є центральною точкою зв'язку Microsoft для обчислень машинного навчання в хмарі Azure.

Розділ "Compute" надає можливість вибрати необхідні обчислювальні ресурси для моделі або сервісу. Microsoft пропонує різноманітні обчислювальні ресурси, починаючи від невеликих обчислювальних машин з кількома гігабайтами оперативної пам'яті до потужних машин, здатних керувати практично будь-якою моделлю. Також можна вибрати між процесорами (CPU) та графічними процесорами (GPU). Варіанти в розділі "Compute" включають обчислювальні екземпляри, кластери навчання, кластери інференсу та під'єднані обчислювальні ресурси.

Опісля, в розділі "Notebooks" можна обрати обчислювальний ресурс, який був створений в розділі "Compute", щоб запустити блокнот написаної програми на ньому.

Скориставшись доступними квотами, як обчислювальний ресурс було взято на виконання центральний процесор з розміром віртуальної машини Standard_E4ds_v4 (4 cores, 32 GB RAM, 150 GB disk).

5.1.4 Бібліотека customtkinter

Для розробки користувацького інтерфейсу десктопного застосунка було використано спеціалізовану бібліотеку customtkinter [20].

Бібліотека customtkinter — це бібліотека Python, розроблена для створення власних і розширених графічних інтерфейсів користувача (GUI) за допомогою фреймворку Tkinter. Вона має на меті надати додаткові можливості, функції та параметри налаштування, які виходять за рамки базового функціоналу стандартної бібліотеки Tkinter.

Бібліотека містить різноманітні класи, функції та методи, які спрощують розробку GUI та створюють більш динамічний та візуально привабливий інтерфейс користувача.

5.2 Попередня обробка даних

Першим кроком у імплементації алгоритмів за допомогою вищезгаданих технологій була робота з обробітком вхідних даних. Для цього були реалізовані три базових класи, такі як “Select_Points”, “Normalize”, і “ToTensor”. Ці класи відповідають етапам попередньої обробки даних, які застосовуються до деякого тривимірного об’єкта сітки.

Клас “Select_Points” приймає як вхідний параметр загальну кількість точок 3-D предметів, що будуть розглядатися, і ініціалізується цим значенням. При виклику з сіткою він витягує вершини та грані з об’єкта сітки. Потім обчислює площі кожної грані, використовуючи координати вершин. На основі областей клас вибирає підмножину граней за допомогою підходу зваженого випадкового вибору. Для кожної вибраної грані він генерує випадкову точку в межах трикутника, визначеного його вершинами, і зберігає ці вибіркові точки.

Щоб нормалізувати вхідну хмару точок був впроваджений клас “Normalize”. Він приймає хмару точок як вхідний аргумент та перевіряє, щоб її форма була двовимірною. Під час виклику класу, віднімається середнє значення хмари точок уздовж кожного виміру, щоб центрувати дані навколо початку координат. Наступним кроком він ділить хмару точок на максимальну норму

(евклідову відстань) будь-якої точки в хмарі, гарантуючи, що всі точки потрапляють в одиничну сферу.

Клас “ToTensor” перетворює дані хмари точок у тензор PyTorch. Подібно до попереднього класу, він очікує хмару точок як вхідний параметр та перевіряє її форму. Вкінці, функціонал класу конвертує масив бібліотеки “Numpy” у тензор PyTorch та повертає його як результат. Цей клас виконує важливу функцію у поданні даних в архітектуру моделей машинного навчання.

5.3 Зчитування набору даних

Для основних процесів роботи з нейронними мережами необхідно правильно доступитися і подати вхідні дані. Тому був імплементований клас під назвою “ModelRead”, який дає змогу розпакувати вхідний набір даних і повернути всі файли окремо у зручному для опрацювання вигляді. Цей клас успадковується від класу Dataset, що вказує на його сумісність із обробкою набору даних PyTorch [21].

На ранньому етапі розглядаються шлях кореневого каталогу, додатковий параметр папки та параметр перетворення як вхідні аргументи класу. Кореневий каталог містить підкаталоги, що представляють різні категорії об’єктів. Далі класи організовуються у словнику, де ключами є імена папок, а значеннями є відповідні індекси класів.

Один із методів класу, який працює з окремим файлом, доступається за допомогою індексу до шляху предмета та мітки класу. Згодом він зчитує об’єктний файл (у форматі .off), використовуючи отриманий шлях. Файл читається рядок за рядком, вилучаючи дані об'єкта, такі як кількість граней і координати вершин. Далі до даних застосовуються операції перетворення, визначені у розділі 5.2, і повертається словник з перетвореними даними та відповідною категорією об’єкта.

Зрештою, клас “ModelRead” забезпечує зручний інтерфейс для читання та доступу до зразків даних із поданого набору даних, а також підтримує додаткові перетворення, що застосовуються до об’єктів.

5.4 Впровадження архітектури моделей

Основним кроком у визначенні моделей машинного навчання є задання внутрішньої системи архітектури з усіма необхідними шарами, через які під час роботи системи проходять вхідні точки 3-D об’єкта. Правильне формування мережі призводить до ефективних і точних результатів на виході. Тому були реалізовані класи двох алгоритмів нейронних мереж “PointNet” та “PointNet++” разом із додатково важливими складовими цих моделей.

5.4.1 PointNet та її складові

Клас “T-Net” представляє однойменну модель. Вхідний параметр представлений розмірністю вхідної хмари точок. Окрім цього визначаються послідовність згорткових шарів (ConvBlock) і повнозв’язних шарів (FC_Block). Згорткові шари витягують ієрархічні об’єкти з вхідних даних, а повністю зв’язані шари відображають витягнуті об’єкти на виході.

Для визначення моделі було реалізовано клас “PointNet”. У якості вхідних даних приймається кількість класів набору даних. Він ініціалізує екземпляри вищезгаданого “T-Net” з різними вхідними розмірами. Клас визначає два набори згорткових шарів (ConvSet_1 і ConvSet_2) і повністю пов’язаних шарів (FC_Layers). Екземпляри “T-Net” використовуються для перетворення вхідних даних хмари точок перед проходженням їх через згорткові шари. Перетворені дані потім обробляються наступними згортковими та повністю зв’язаними шарами для отримання остаточного результату, який представляє ймовірності класу.

У кожному з класів є зроблений метод “forward”, який призначений для прямого проходу моделей. У класі “PointNet” прямий перехід передбачає застосування екземплярів “T-Net” для перетворення вхідних даних, а потім згорткових і повністю зв’язаних шарів. Вихід останнього повністю підключеного рівня представляє прогнозовані ймовірності класу.

5.4.2 Архітектура PointNet++

Модель PointNet++ складається з кількох модулів, кожен з яких відповідає за певні операції.

Клас PointNetSetAbstractionMsg реалізує операцію абстрагування набору. Він приймає за вхідні дані позиції вхідних точок (xyz) і об’єкти (точки). Клас виконує операції групування та згортання над вхідними точками, створюючи нові точки зі зменшеною розмірністю.

У класі PointNet++ визначена архітектура моделі. Він включає три екземпляри класу PointNetSetAbstractionMsg, які виконують операції абстрагування набору над вхідними точками. Вихідні дані кожного модуля абстракції набору далі обробляються з використанням повністю зв’язаних шарів із пакетною нормалізацією і вилученням. Остаточний результат передається через функцію “softmax”, який вказує на передбачення категорії предмету.

5.5 Підготовка до тренування

Опісля реалізації всіх вищеописаних структур доходить черга до ініціалізації та оголошення відповідних змінних. Перш за все, відбувається завантаження робочих наборів даних, до яких застосовується перетворення, і розподілення їх по тренувальних та тестувальних частинах. Елементи цих множин даних повинні бути між собою випадково помішані та поділені на певне число “батчів”, які контролюють точність оцінки помилки при навчанні нейронних мереж.

Наступним завданням є перевірка сумісності комп’ютерного пристрою, на якому буде відбуватися процес тренування, з доступними процесорами. Якщо в приладі є присутній графічний процес, що значно пришвидшує навчання і оголошення результатів, то він береться до уваги і застосовується до визначених моделей. В іншому випадку використовується центральний процесор.

Згодом створюється екземпляр класу імплементованих нейронних мереж, в які передається загальне число класів деякого набору даних. Тоді до моделі призначається вибраний процесор.

Далі відбувається оголошення оптимізаційних налаштувань та додаткових гіперпараметрів. Таким чином, визначається:

- темп навчання, який задає розмір кроку на кожній ітерації пошуку мінімуму функції втрат;
- критерій, який обчислює перехресну втрату ентропії між вхідними логітами та ціллю;
- оптимізатор, який мінімізує попередньо визначену функцію втрат на заданих тестових даних.

5.6 Тренування моделі

Перед навчанням моделі визначається число епох, протягом яких навчальні дані проходять через алгоритм. У межах кожної епохи весь навчальний набір даних тренується партіями. Для кожної партії витягуються точки зображень та відповідні мітки, які передаються на призначений пристрій (процесор). Визначена модель машинного навчання застосовується до зображень для отримання прогнозованого результату. Далі результат порівнюється з міткою за допомогою визначеного критерію для розрахунку втрат.

Згодом, виконується зворотне поширення для обчислення градієнтів, а оптимізатор оновлює параметри моделі. Значення втрат накопичується за епоху. Цей процес повторюється для всіх пакетів у навчальному наборі даних, і наприкінці кожної епохи накопичена втрата представляє загальну втрату для цієї епохи.

Після завершення тренування модель з останньої епохи зберігається у файлі з розширенням .pt. Цей крок є необхідним для подальшого використання навчаної моделі.

6 РЕЗУЛЬТАТИ КЛАСИФІКАЦІЇ ТА ОЦІНКА ЕФЕКТИВНОСТІ

Наступним і одним із завершальних кроків імплементування моделей є тестування та застосування результатів на практиці. Тому в цьому розділі буде детально описано процес тестування та ознайомлення з візуалізацією результатів.

6.1 Тестування навчених моделей

Тестування проводиться майже таким самим чином як і навчання, тільки в цьому випадку застосовується вже натренована та оцінена модель машинного навчання. Метою тестування є визначення точності алгоритму на нових, випадкових даних, без попереднього оголошення категорії 3-D об’єкта.

Перші етапи по суті не відрізняються від процесу тренування і вже завантажена модель застосовується до набору точок кожного з файлів тестувальної вибірки даних, розподіленої по партіях. Прогнозовані результати порівнюються з назвами класів, щоб визначити кількість правильних передбачень. Цей процес повторюється для всіх партій у тестовому наборі даних. Наприкінці розраховується точність моделі на тестових зображеннях шляхом ділення кількості правильних прогнозів на загальну кількість елементів.

По завершенню всіх етапів машинного навчання було проведено тестування двох наборів даних ModelNet10 та TraceParts на двох моделях PointNet і PointNet++. У таблиці 6.1 наведено порівняння точностей двох алгоритмів з двома різними тестовими наборами даних.

З результатів видно, що PointNet++ демонструє кращу точність на 3.9% в порівнянні з PointNet з тестовими даними ModelNet10.

Таблиця 6.1 – Наведення точностей алгоритмів на наборах даних

Набір даних	Точність моделі	
	PointNet	PointNet++
ModelNet10	87.8%	91.7%
TraceParts	97.6%	98.8%

Розглядаючи набір даних TraceParts, то можна зробити висновок, що через малу кількість класів і належне число об’єктів у множині всіх файлів, точність передбачення міток є практично однаковою і надзвичайно високою у двох моделях.

6.2 Порівняння результатів

Після проведення експериментів було побудовано матрицю невідповідностей [22] для кращої візуалізації виконання класифікації. Кожен елемент рядка матриці є очікуваним класом, а кожен елемент у стовпці є справжньою міткою на вході. По діагоналі розташовано кількість правильних класифікацій. На рисунку 6.1 проілюстровані результати виконання матриці невідповідностей для моделей PointNet та PointNet++ на наборі даних ModelNet10.

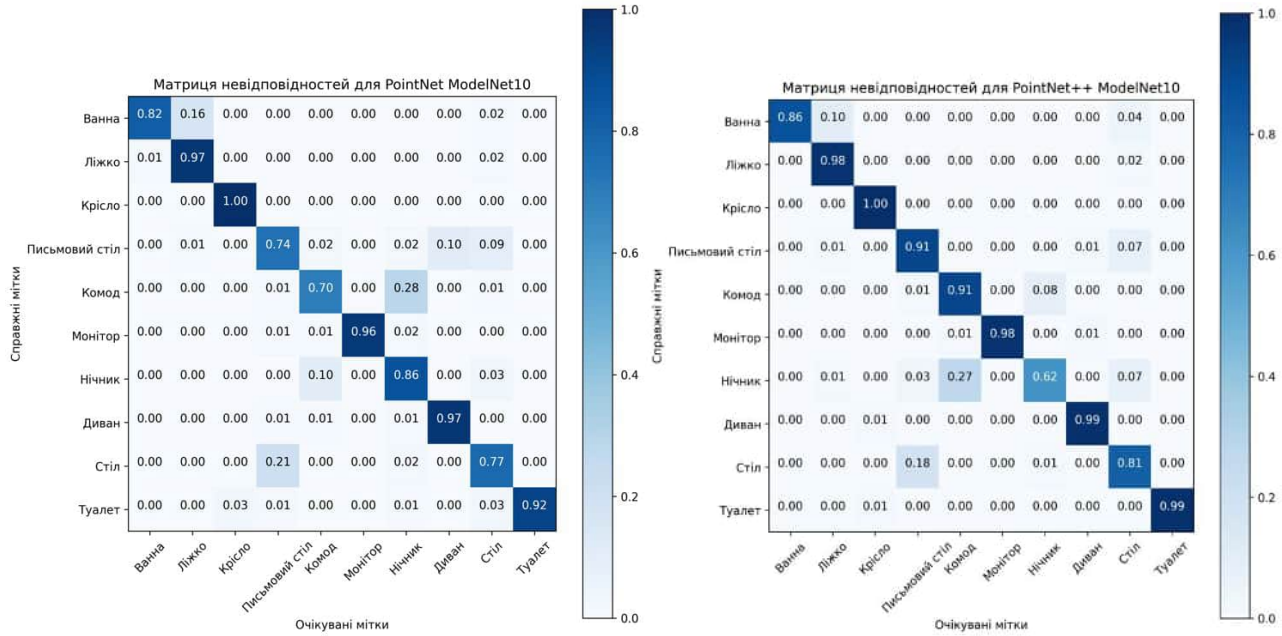


Рис. 6.1 Матриця невідповідностей для набору даних ModelNet10 для моделей PointNet та PointNet++

По кількості правильних передбачень видно, що модель PointNet визначає майже з впевненістю передбачення для 5-ти класів (“Ліжко”, “Крісло”, “Монітор”, “Диван”, та “Туалет”). Найгіршою точністю виділяється мітка під назвою “Комод” з 70-ма відсотками. Цей клас часто помилково сприймається мережею як “Нічник”.

PointNet++ в свою чергу демонструє гарні результати практично для всіх категорій. Однак, клас з назвою “Нічник” класифікується найгірше, якщо брати до уваги дві моделі. Мережа певною мірою помиляється з визначенням цієї категорії на користь “Комоду”.

Взявши до уваги рис. 6.2, то можна підсумувати, що обидві моделі практично беззаперечно визначають передбачувані мітки.

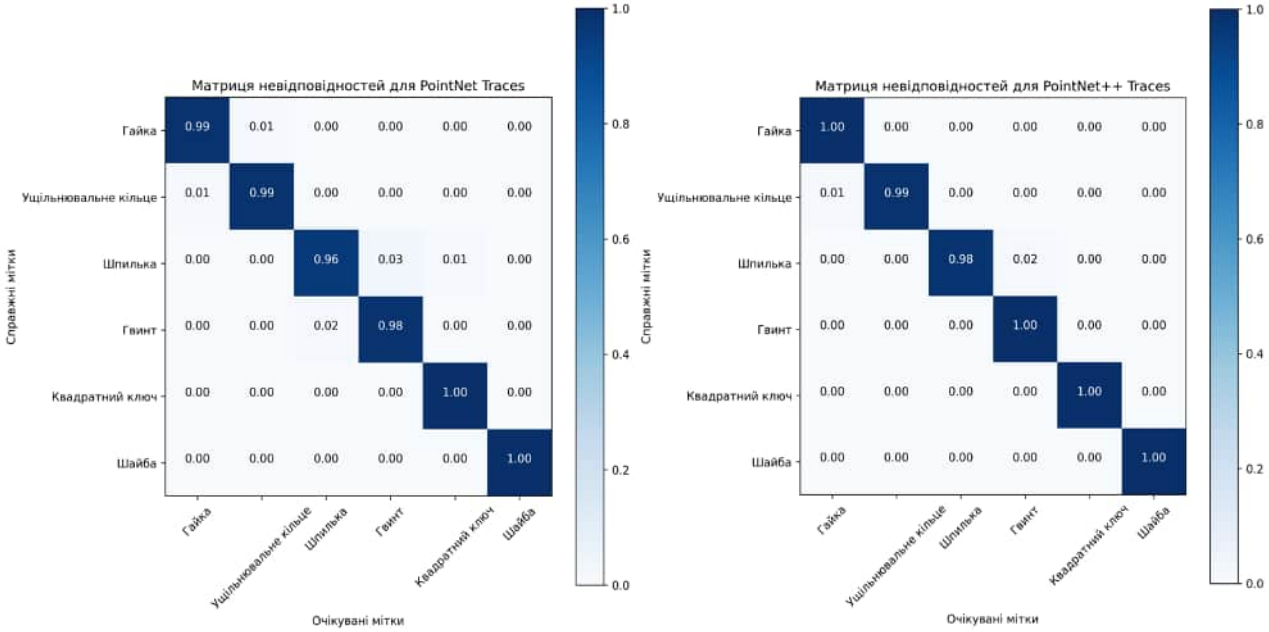


Рис. 6.2 Матриця невідповідностей для набору даних TraceParts для моделей PointNet та PointNet++

Оскільки точності продемонстрованих навчених алгоритмів є максимально близькими до одиниці, то очевидно, що певні категорії будуть класифікуватися з мінімальною похибкою.

7 РОБОТА ЗАСТОСУНКУ

В цьому розділі буде продемонстровано інтерфейс користувача реалізованої десктопної аплікації для прогнозування міток 3-D об’єктів під назвою “PPScanNet+”.

7.1 Панель класифікації

Запустивши програму, перед користувачем появляється основна вкладка системи під назвою “Класифікація”, яка дає можливість на основі вже натренованих моделей машинного навчання класифікувати наперед вибраний об’єкт тривимірного простору і ознайомитися з результатами передбачення.

Для більш зручнішого використання, застосунок забезпечує вибір файлів з різними розширеннями 3-D моделей. Як було описано у попередніх розділах, моделі PointNet та PointNet++ працюють з наборами даних, які містять в собі файли з розширеннями .off. Тому додатково було додано функціонал автоматичного конвертування файлів 3-ох типів, а саме .obj, .ply та .stl, у попередньо згаданий формат.

Після вибору файлу з файлового провідника (рис. 7.1), користувачу стає доступним перегляд та маніпулювання об’єктом, який за замовчуванням відображений у суцільно-каркасному вигляді, за допомогою відповідних рухів мишкою.

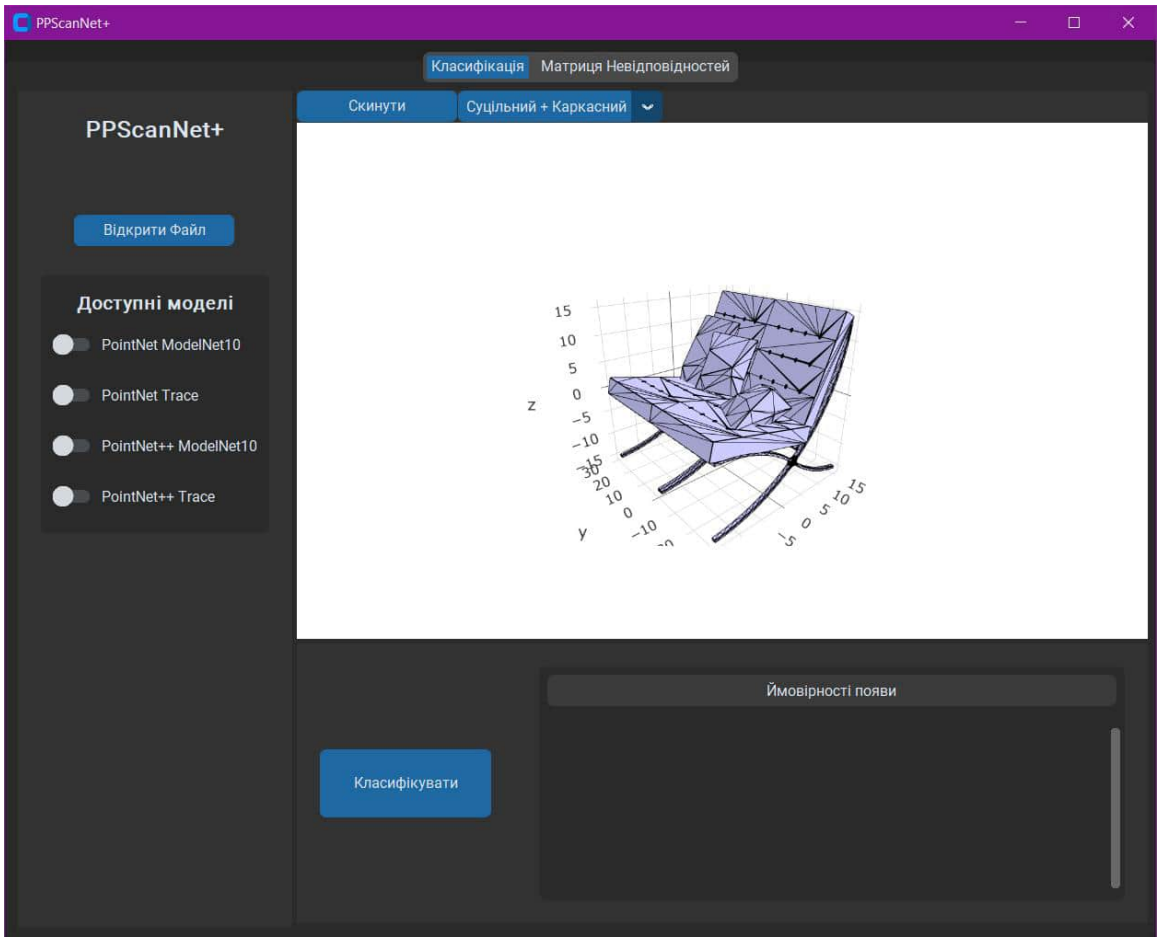


Рис. 7.1 Основна панель застосунку

Окрім цього, як зображено на рис. 7.2, імплементовані інші функції для візуалізації предметів з іншої перспективи.

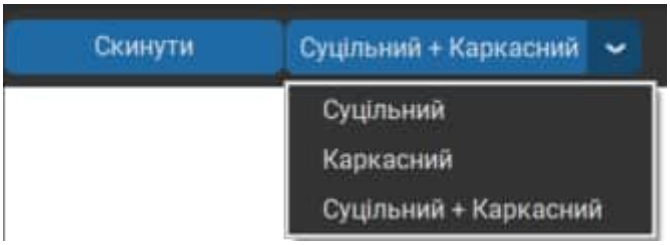


Рис. 7.2 Вибір перспективи зображення

За допомогою кнопки “Скинути” можна повернутися до стану, який демонструє об’єкт на початковому етапі після його відкриття. У випадному списку постає можливість змінити загальний образ предмету на просто “Суцільний” (рис. 7.3а) або лише “Каркасний” (рис. 7.3б) вигляд. Ці характеристики дають змогу передати більш детальну інформацію про одиницю набору даних за потреби.

(а) (б)

Рис. 7.3 Суцільна та каркасна перспективи зображення

Повертаючись до основної мети застосунку, до бокової панелі (рис. 7.4) було добавлено натреновані моделі з найвищою точністю прогнозу з усіх експериментів. Користувач аплікації перед самою класифікацією повинен обрати на вибір одну або дві бажані моделі, файли яких є автоматично ініціалізовані та викликані після запуску програми. Моделі наразі працюють з двома наборами даних, такими як ModelNet10 та TraceParts.

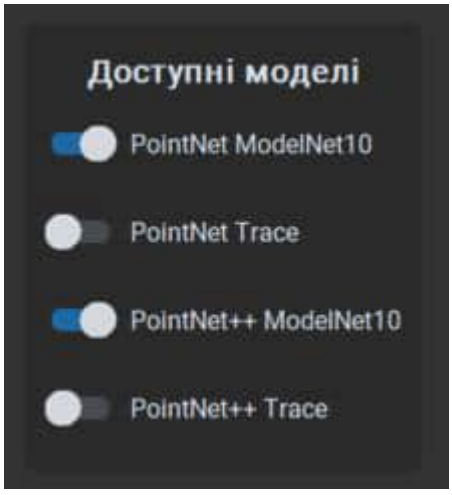


Рис. 7.4 Вибір доступних моделей

Обравши файл з бажаним тривимірним предметом та модель машинного навчання, можна перейти до самого кроку класифікації. У нижній частині

основної панелі проілюстрований функціонал визначення назви об’єкта (рис. 7.5), а саме:

- кнопка “Класифікувати”, яка на основі визначених моделей обраховує тестувальну частину завдання;
- таблиця “Ймовірності появи”, яка зображає ймовірності передбачення для кожного з класів набору даних.

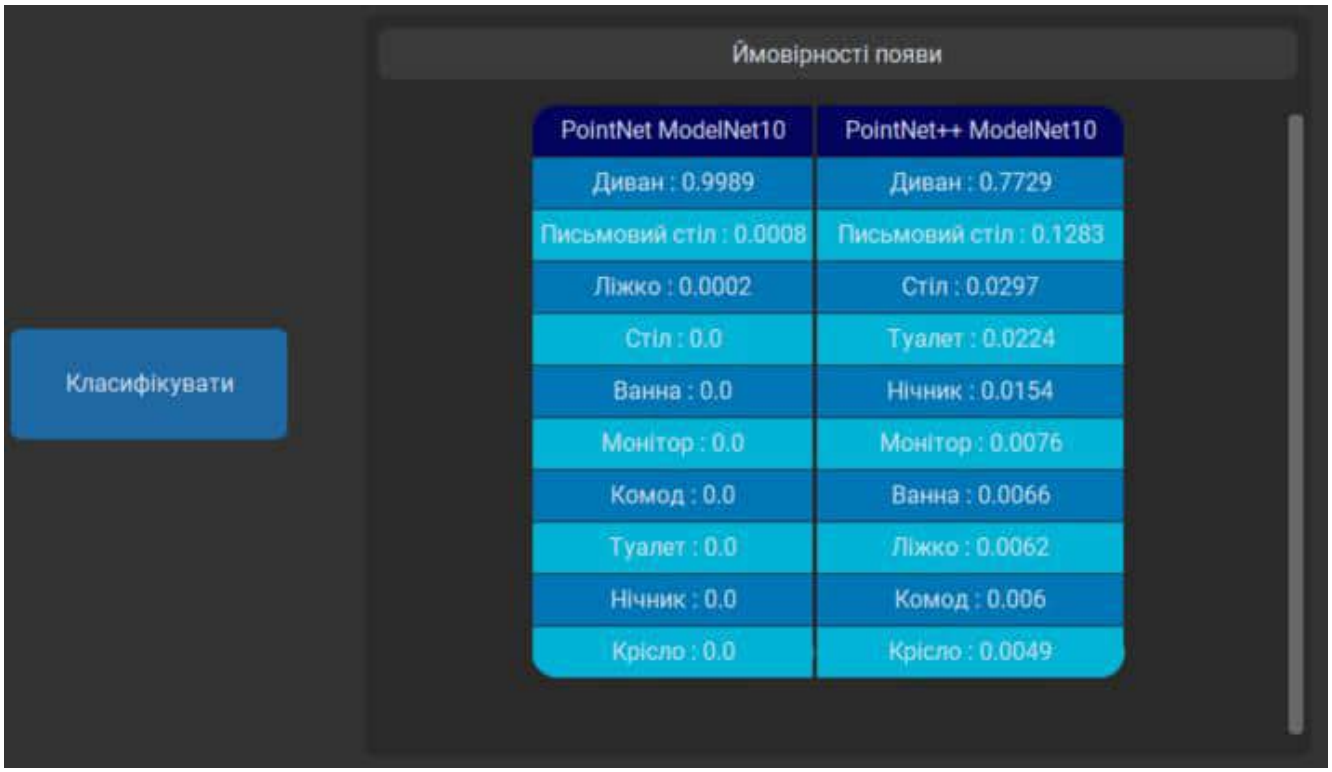


Рис. 7.5 Відображення результатів класифікації

На рис. 7.5 можна побачити, що обидві моделі визначили правильні мітки для початкового вибраного 3-D предмету, хоча модель PointNet++ мала більші сумніви у результаті. Зауважимо, що сума ймовірностей для всіх класів дорівнює одиниці для кращого і правильного сприйняття результатів.

Наступним кроком, показаним на рис. 7.6, є селекція будь-якої фігури з іншого набору даних під назвою “TraceParts”.

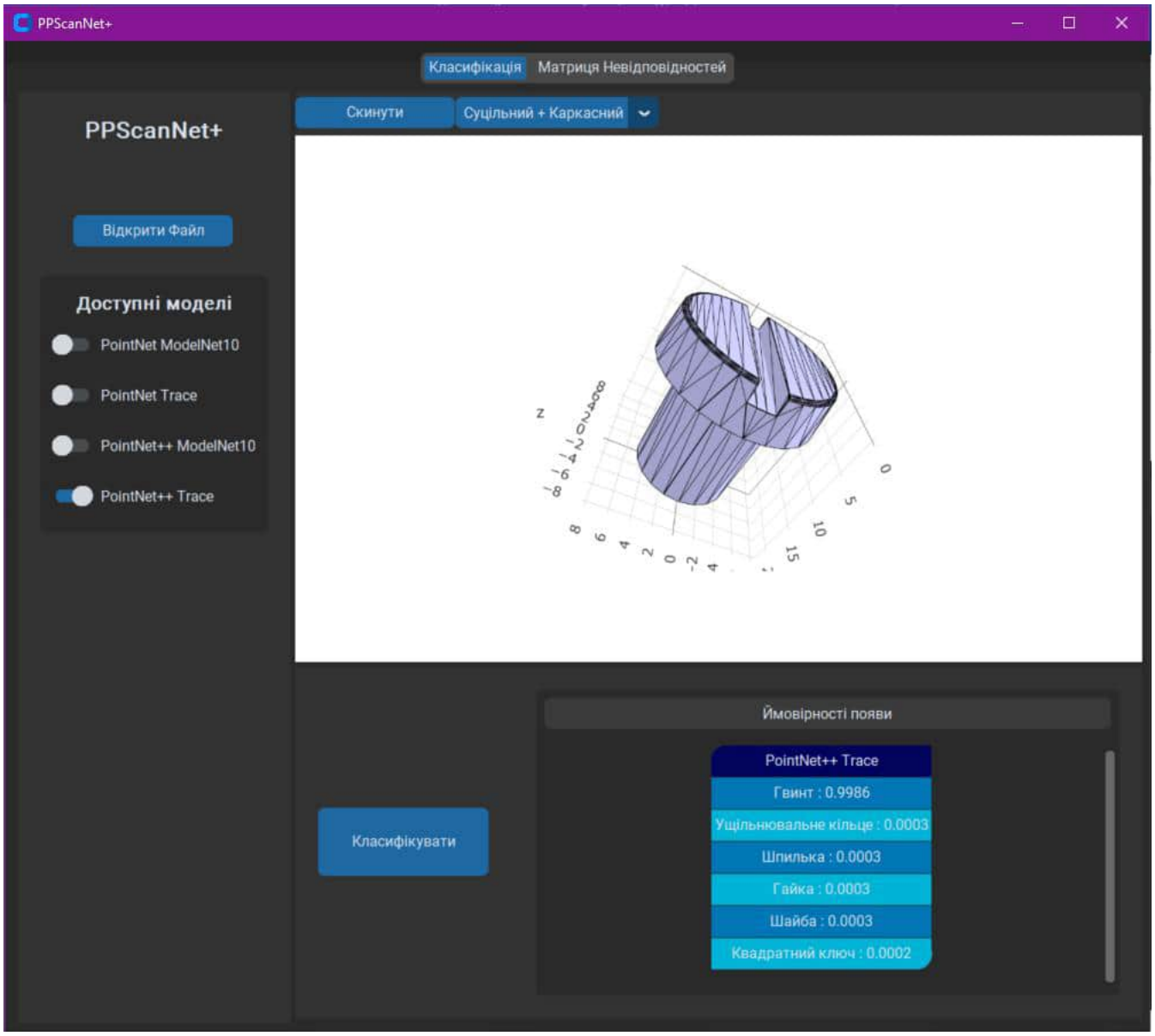


Рис. 7.6 Результати для іншого набору даних

Після вибору лише однієї моделі з усіх доступних та класифікації, наведено загальну картину застосунку з показаними результатами точного прогнозування для 6-ти класів.

7.2 Панель матриці невідповідностей

Щоб розширити функціональність застосунку, було впроваджена вкладка візуалізації матриць невідповідностей. При переході цю панель, користувач програми повинен вибрати одну з опцій моделей, для обрахування матриці (рис 7.7).

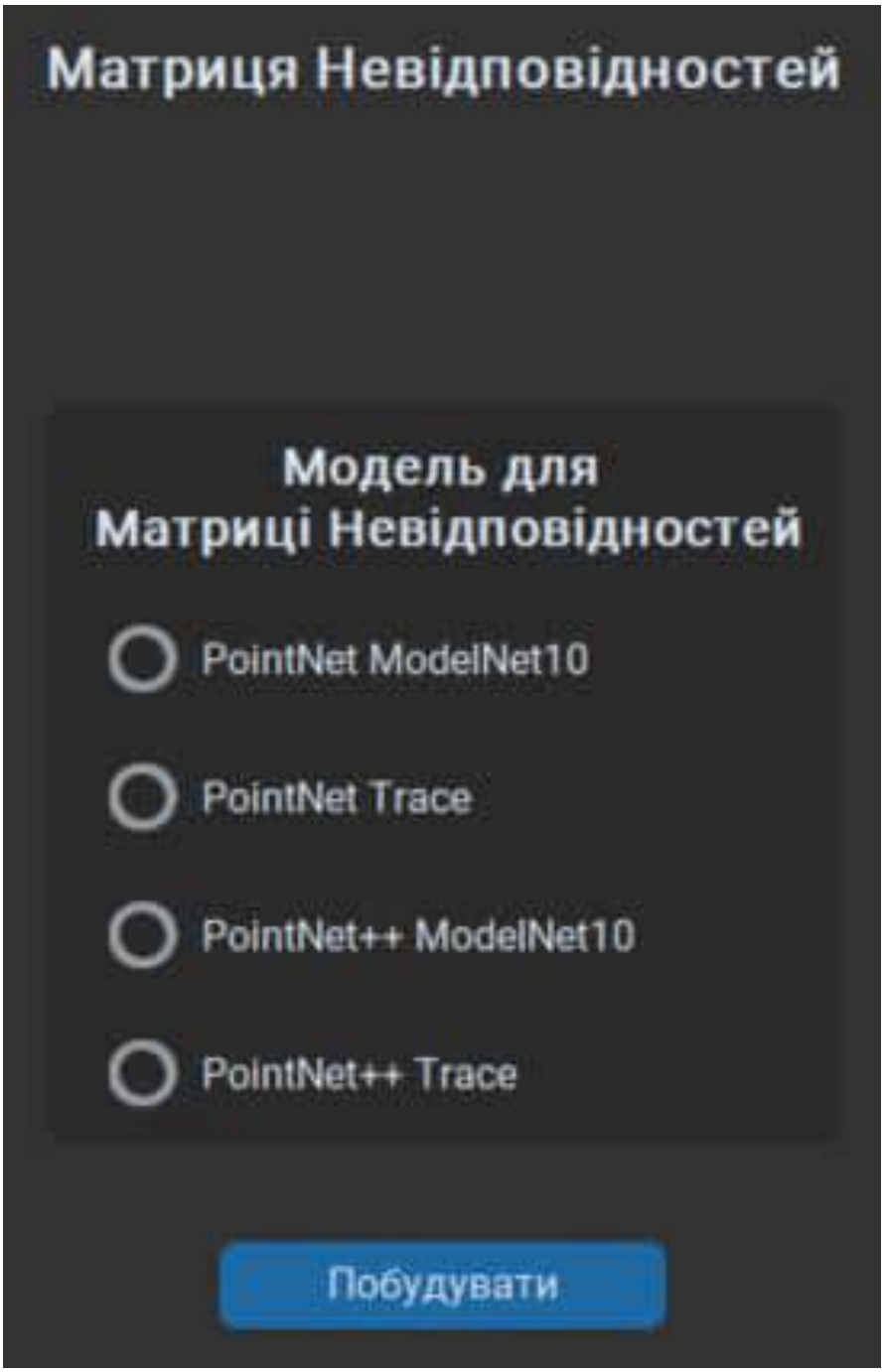


Рис. 7.7 Вибір побудови матриці невідповідностей

Натиснувши на кнопку “Побудувати”, в центральній частині програми (рис. 7.8) з’являється матриця невідповідностей, яка обраховується на тестовому наборі даних вибраної моделі. Деталі та принцип самої матриці були описані в розділі 5.

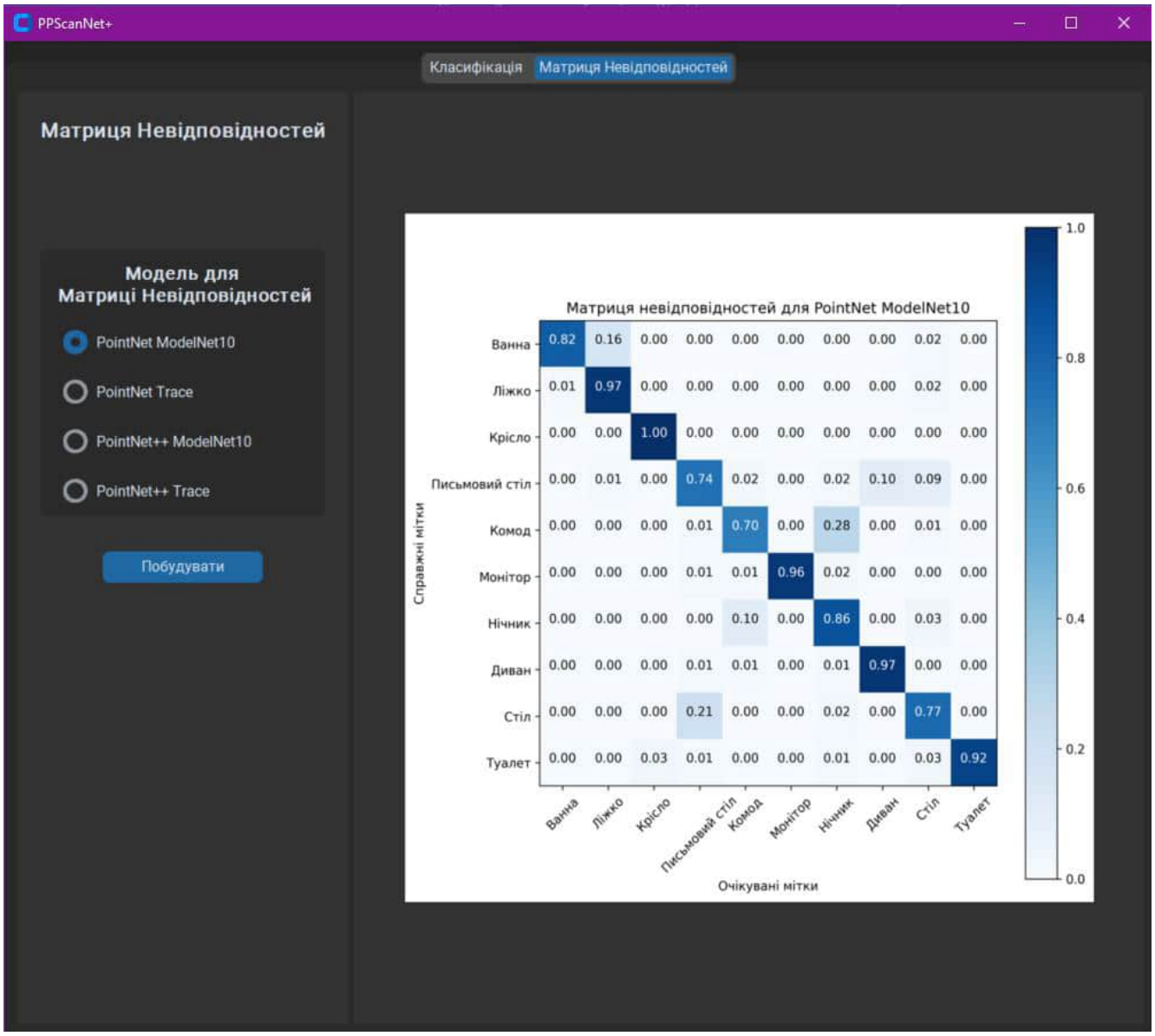


Рис. 7.8 Візуалізація матриці невідповідностей

У цьому розділі було показано теперішню версію застосунку. У майбутньому планується вдосконалення користувацького інтерфейсу, тренування та тестування моделей, задаючи власний набір даних, додавання більшої кількості моделей, можливість демонстрації 3-D фігур у вигляді регульованого числа точок та побудова графіків точності алгоритму.

ВИСНОВКИ

У даній дипломній роботі було поставлено задачу розробки системи класифікації 3-D моделей за допомогою методів машинного навчання. Для досягнення цієї мети був розроблений план реалізації проекту, який включав такі етапи, як ознайомлення з предметною областю, вибір відповідних наборів даних, порівняння та обрання алгоритмів машинного навчання, вибір технологій та інструментів для реалізації. Окрім цього, була впроваджена реалізація моделей, навчання та оцінка їх продуктивності, порівняння результатів та створення десктопного застосунку для візуалізації та класифікації.

У ході дослідження, вдалося ефективно вирішити проблему класифікації 3-D об'єктів. Зроблено значний прогрес у розумінні та виборі найефективніших алгоритмів машинного навчання для класифікації таких об'єктів. Отримані результати демонструють високу точність та правильність прогнозів для кожної категорії вибірки.

Крім того, розроблено десктопний застосунок, який дозволяє зручно класифікувати та візуалізувати 3-D об'єкти. За допомогою програми та її функціоналу можна переглядати відкритий файл 3-D об’єкта з різних ракурсів та стилів подання, вибирати бажані натреновані моделі, проводити класифікацію та порівнювати результати кількох алгоритмів одночасно за допомогою таблиці. Окрім цього, реалізовано панель з ілюстраціями матриці невідповідностей для кожної з моделей для візуального порівняння справжніх міток з очікуваними. Зрештою, цей застосунок має інтуїтивно зрозумілий та зручний інтерфейс, що сприяє зручній взаємодії з системою та полегшує сприйняття результатів.

У майбутньому, планується розширення системи шляхом використання нових наборів даних та алгоритмів машинного навчання, таких як ShufflePointNet [23], DGCNN [24] та PointCNN [25]. Також, планується запустити систему на віддаленому сервері, що дозволить забезпечити доступ до неї з будь-якого місця та сприяти розповсюдженню і використанню даної технології у широкому колі дослідників та спеціалістів.

Отже, дана дипломна робота вирішує (розглядає) важливу (актуальну) проблему класифікації 3-D об'єктів за допомогою машинного навчання. Результати дослідження є об'єктивним підтвердженням ефективності обраних підходів та створеного застосунку. Завдяки цим досягненням, можна сподіватись на подальший прогрес у розробці та застосуванні систем класифікації 3-D об'єктів за допомогою машинного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Платформа Prometheus: Курс Машинне Навчання [Prometheus] – Режим доступу: https://prometheus.org.ua/course/course-v1:IRF+ML101+2016_T3
2. І.А. Терейковський, Д.А. Бушуєв, Л.О. Терейковська Штучні нейронні мережі: базові положення [Навчальний посібник] – Режим доступу: https://ela.kpi.ua/bitstream/123456789/50135/1/ANN.pdf
3. McCulloch-Pitts Neuron — Mankind's First Mathematical Model Of A Biological Neuron [Towardsdatascience] – Режим доступу: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1
4. Багатошаровий персептрон. [Sciencedirect] – Режим доступу: https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron
5. Backpropagation Process in Deep Neural Network. [Javatpoint] – Режим доступу: https://www.javatpoint.com/pytorch-backpropagation-process-in-deep-neural-network
6. Convolutional Neural Networks, Explained [Towardsdatascience] – Режим доступу: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
7. Файловий формат OFF [Wikipedia] – Режим доступу: https://en.wikipedia.org/wiki/OFF_(file_format)
8. PointCloud guide [Gigabyte] – Режим доступу: https://www.gigabyte.com/Glossary/point-cloud
9. Point-Voxel CNN for Efficient 3D Deep Learning [Pvcnn] – Режим доступу: https://pvcnn.mit.edu/
10. H. Su, S. Maji, E. Kalogerakis. Multi-view Convolutional Neural Networks for 3D Shape Recognition [Thecvf] – Режим доступу: https://openaccess.thecvf.com/content_cvpr_2016/papers/Qi_Volumetric_and_Multi-View_CVPR_2016_paper.pdf

11. Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation [Arxiv] – Режим доступу: https://arxiv.org/pdf/1612.00593v2.pdf
12. An In-Depth Look at PointNet [Medium] – Режим доступу: https://medium.com/@luis_gonzales/an-in-depth-look-at-pointnet-111d7efdaa1a
13. Charles R. Qi, Hao Su, Kaichun Mo. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space [Neurips] – Режим доступу: https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf
14. ModelNet dataset description [ModelNet] – Режим доступу: https://modelnet.cs.princeton.edu/
15. SUN database [Csail] – Режим доступу: https://groups.csail.mit.edu/vision/SUN/hierarchy.html
16. Traceparts platform [Traceparts] – Режим доступу: https://www.traceparts.com/fr
17. Jupyter Notebook Documentation [Jupyter] – Режим доступу: https://docs.jupyter.org/en/latest/
18. PyTotch usage [Nvidia] – Режим доступу: https://www.nvidia.com/en-us/glossary/data-science/pytorch/
19. Azure Machine Learning Studio [Azure] – Режим доступу: https://ml.azure.com/
20. CustomTkinter Documentation [CustomTkinter] – Режим доступу: https://customtkinter.tomschimansky.com/documentation/
21. Документація PyTotch [Pytorch] – Режим доступу: https://pytorch.org/get-started/locally/
22. Матриця невідповідностей [Wikipedia] – Режим доступу: https://en.wikipedia.org/wiki/Confusion_matrix
23. Can Chen, Luca Zanotti Fragonara, Antonios Tsourdos. Go Wider: An Efficient Neural Network for Point Cloud Analysis via Group Convolutions [Arxiv] – Режим доступу: https://arxiv.org/abs/1909.10431

24. Yue Wang, Yongbin Sun, Ziwei Liu. Dynamic Graph CNN for Learning on Point Clouds [Arxiv] – Режим доступу: https://arxiv.org/abs/1801.07829
25. Yangyan Li, Rui Bu, Mingchao Sun. PointCNN: Convolution On X-Transformed Points [Arxiv] – Режим доступу: https://arxiv.org/abs/1801.07791