КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття ступеня магістра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

УПРАВЛІННЯ РОБОТИЗОВАНИМИ СИСТЕМАМИ З ВИКОРИСТАННЯМ ГОЛОСОВИХ КОМАНД НА ОСНОВІ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Виконав студент 2-го курсу магістратури
Терпіловський Єгор Олександрович _____
(підпис)

Науковий керівник:
доцент, кандидат фізико-математичних наук
Верес Максим Миколайович _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
протокол № 12 від 12 травня 2021 р.
Завідувач кафедри
О.І. Провотар _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи − 57 сторінки, містить 17 рисунків, 5 додатків, 15 джерел.

Кваліфікаційна робота присвячена розробці системи управління роботизованими системами з використанням голосових команд на основі генетичних алгоритмів. Метою цієї кваліфікаційної роботи є підвищення ефективності управління роботизованою системою розумного будинку за допомогою голосових команд на основі генетичних алгоритмів.

Об’єктом дослідження цієї кваліфікаційної роботи є роботизована система розумного будинку. Предметом дослідження даної кваліфікаційної роботи є види та методи управління роботизованою системою розумного будинку за допомогою голосових команд. Сфера використання даної кваліфікаційної роботи полягає у спрощенні способу досягнення розпізнавання голосових команд на основі вузькоспеціалізованих генетичних алгоритмів. Зв'язок цієї кваліфікаційної роботи з іншими роботами полягає у поєднанні нейромережевих технологій та генетичного алгоритму в моделі розпізнавання голосових команд.

Було порівняно та проаналізовано кілька методів навчання нейронних мереж розпізнаванню звуку. Виявлені переваги та недоліки розглянутих методів. На основі побудованої моделі нейронної мережі та обраного методу навчання було написано програмне забезпечення, яке повністю відповідає поставленому завданню. У цій роботі помітна швидкість і точність розпізнавання команд. Результати роботи генетичного алгоритму відповідають очікуванням швидкості навчання нейронної мережі, на відміну від алгоритму зворотного поширення помилок. Крім того, генетичний алгоритм набагато легше зрозуміти та записати, але вони також мають свої недоліки. Одна з них – це не проста операція масштабування створеної моделі, оскільки якщо вам потрібно додати кількість слів до однієї команди, функцію пристосування доведеться змінити, а також перевчити мережу.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ПЗ – програмне забезпечення.

ГА – генетичний алгоритм.

ПК – персональний комп’ютер.

РС – роботизована система.

РД – розумний дім.

ШНМ – штучна нейронна мережа.

РНМ – рекурентна нейронна мережа.

ВСТУП

Оцінка сучасного стану об’єкта розробки. У сучасному світі спостерігається бурхливий розвиток технологій.

Мобільні пристрої замінюють класичні настільні ПК та ноутбуки. Стандартні інтерфейси також втрачають популярність. Пристрої введення: клавіатура, миша, джойстики. Їх замінюють більш природні інтерфейси: сенсорний інтерфейс, знайомий нам з дитинства (діти починають досліджувати світ за допомогою дотику), голосове управління та жести, добре відомі і використовуються в повсякденному житті (спілкування).

У багатьох науково-фантастичних творах супутником героя є робот або комп’ютер. І він здатний не тільки виконувати роботу, команди, обчислення, а й відповідати на запитання. Він може відповісти на запитання людини, як на серйозні, що вимагають пошуку інформації, так і на легковажні, що вимагають почуття гумору в машині. Він також може виконувати запити, що відповідають запитам на розумні домашні технології, такі як: увімкнути світло, змінити температуру в приміщенні, приготувати вечерю, змінити курс корабля тощо.

З кожним днем ми наближаємося все ближче і ближче до фантастичного світу майбутнього. Сьогодні голосове управління стало реальністю, і фрази, адресовані телефону, нікого не здивують, наприклад, «Створити маршрут до Поштової площі», «Подзвони мамі», «Заплануй будильник на восьму ранку».

Мета й завдання роботи. Метою цієї кваліфікаційної роботи є підвищення ефективності управління роботизованою системою розумного будинку за допомогою голосових команд, що будуть розпізнані за допомогою нейронної мережі, навчання якої відбуватиметься на основі генетичних алгоритмів.

Об’єкт, методи й засоби розроблення. Об’єктом дослідження цієї кваліфікаційної роботи є роботизована система розумного будинку. Предметом дослідження даної кваліфікаційної роботи є види та методи

управління роботизованою системою розумного будинку за допомогою голосових команд. Засоби розроблення є нейронна мережа та генетичний алгоритм.

Можливі сфери застосування. Сфери застосування цієї кваліфікаційної роботи є у спрощенні способу досягнення розпізнавання голосових команд на основі вузькоспеціалізованих генетичних алгоритмів.

Взаємозв'язок з іншими роботами. Взаємозв'язок цієї кваліфікаційної роботи з іншими роботами є у поєднанні нейромережевих технологій та генетичного алгоритму в моделі розпізнавання голосових команд.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Для цієї кваліфікаційної роботи було обрано роботизовану систему розумного будинку.

1.1. Опис проблеми розумного дому

Щодня світ освоює новітні технології розумних пристроїв, які полегшують життя людям, економлять їхній час та ресурси. Однією з таких технологій є розумний дім. Ця технологія дозволяє контролювати різні роботизовані системи вдома за допомогою різних інтерфейсів взаємодії. Ці інтерфейси в різних системах можуть відрізнятися в базовій концепції управління.

Кожен із цих методів має свої недоліки. Сценарії – це дуже узагальнені випадки, в які наше життя не завжди вкладається. Проблема виникає, коли наше життя відхиляється від повсякденного сценарію. Приклад з апаратним пультом дистанційного керування обмежений тим фактом, що людині завжди потрібно носити з собою передавальний пристрій, який також має максимально допустиму відстань, на якій він працює, і залежність від електрики. Звукові команди, навпаки, мають лише недолік у складності вирішення проблеми розпізнавання бажаних звуків, оскільки система повинна мати можливість ізолювати мову від усіх оточуючих шумів, розпізнавати зміни тембру, зміни гучності голосу людини, мови та розділяти мову на окремі команди. Розумний дім – це складна система автоматизації управління різними пристроями, розташованими в приватному будинку або квартирі. Перші прояви такої системи були зафіксовані в середині XX століття, і лише зараз вона поширюється, все більше і більше споживачів обирають цей метод контролю технічного оснащення житла.

1.2. Опис використання роботизованих систем

Важливою особливістю та властивістю «Розумного будинку», що відрізняє його від інших методів організації житлового простору, є те, що це найбільш прогресивна концепція взаємодії людини з житловим простором, коли людина однією командою задає бажану ситуацію, а також автоматично у відповідності із зовнішніми та внутрішніми умовами контролює режими роботи всіх інженерних систем та електроприладів. У цьому випадку відпадає необхідність використання декількох пультів дистанційного керування під час перегляду телевізора, десятків вимикачів при контролі освітлення, окремих блоків при керуванні системами вентиляції та опалення, систем відеоспостереження та сигналізації, воріт тощо. У будинку, який обладнаний системою «Розумний дім», достатньо обрати один із сценаріїв одним клацанням клавіші на стіні (або пультом дистанційного керування, сенсорною панеллю тощо). І тоді сам будинок налаштує роботу всіх систем відповідно до ваших побажань, часу доби, вашого положення в будинку, погоди, навколишнього освітлення тощо, щоб забезпечити комфортний стан всередині будинку.

Що може зробити такий будинок? Почнемо з найпростішого – управління освітленням. За допомогою комбінації елементів управління стає можливим керувати – вмикати, вимикати, регулювати яскравість – будь-яке джерело світла у всіх кімнатах, незалежно від місця розташування лампи та її типу. Ви можете не встати з ліжка, щоб вимкнути світло і повертатися у темряві від перемикача на ліжко. Зі своєї спальні ви можете вимкнути світло в дитячій кімнаті або по всьому будинку. Або залиште нічне освітлення в коридорах на ніч. За допомогою цього набору сценарії або команд ви можете керувати своїм освітленням майже з будь-якого місця, навіть перебуваючи на великій відстані від дому. Якщо ви розмістите датчик руху, світло ввімкнеться, як тільки одна з дверей відкриється у кімнату. Світло можна вмикати та

вимикати у визначений час. А у котеджі за допомогою встановлених на вулиці датчиків про те, що вони знайшли рухомий об’єкт, включається зовнішнє освітлення, а також надсилати сигнал про виявлення об’єкта.

Розумний дім може імітувати присутність власників. Встановлена програма вмикає світло в різних кімнатах, а в пізній час доби вимикає світло, залишаючи аварійне освітлення. Вранці світло знову зникне, і сторонній спостерігач буде повністю впевнений, що в будинку живуть люди.

Є кілька способів керувати своїм розумним будинком. Простіше використовувати клавіатуру, де будь-яка кнопка реагує на пристрій, групу пристроїв або вказує послідовність необхідних дій. Для наочності на клавіатурі може бути невеликий дисплей, на якому відображається інформація про стани пристроїв. Найповажніші та ергономічні моделі застосовують керування сенсорним екраном. Таким пультом є невеликий монітор з кнопками, назвами команд, пояснювальними зображеннями. Ці моделі також можуть відображати зображення з камер. Керування домом через Інтернет є досить ергономічним та зручним. У цьому випадку ви можете дізнатись про стан будинку, про наявність у ньому людей, про погоду і т.д., навіть коли ви знаходитесь на великій відстані. Звичайно, ця інформація буде доступна лише власникам будинків.

Зараз «розумний дім» – одне з найсучасніших досягнень у галузі технологій. Найголовніше, що основною метою домашньої автоматизації є комфорт, адже запам'ятовування та реалізація купи дрібних і не зовсім домашніх завдань – від температури та підтримки вологості до поливу та годування риби в зимовому саду – для цього потрібен не лише час але і постійна увага власників, не кажучи вже про необхідне забезпечення витратними матеріалами. Виходячи з цього, агрегат розумного будинку є найбільш зручним агрегатом для управління приміщенням на сьогодні.

Роботизовані системи широко використовуються в наші дні. Всі рухомі частини автоматизованої системи є роботизованими системами. ПК

використовуються майже у всіх галузях промисловості, у будівництві, у гірництві, майже у всіх видах транспорту, у галузях безпеки, в медицині, у наукових дослідженнях та у повсякденному житті людей, наприклад, турнікети метро, банкомати, кавові машини, розважальні пристрої, ескалатори і більше. У системах розумного будинку також існує велика кількість комп'ютерів на базі мікроконтролерів, про які докладніше йдеться в наступному розділі.

1.3. Опис функціональної моделі роботизованих систем

Основна функціональність роботизованої системи розумного будинку полягає у реагуванні на людські команди або систему сценаріїв відповідно до очікуваного результату. Тобто ПК отримує команду закрити певні двері, отже це має бути не що інше, як закриття певних дверей.

Система розумного будинку, обрана для цієї кваліфікаційної роботи, має наступні функції: зачинити вікно, відчинити вікно, жалюзі вгору, жалюзі вниз, увімкнути вентиляцію, вимкнути вентиляцію, збільшити температуру повітря, зменшити температуру повітря, підняти ворота гаража, опустити ворота гаража.

1.4. Недоліки та переваги

Переваги роботизованих систем розумного будинку: автоматизація повторюваних багато разів в повсякденному житті дій, можливість віддаленого управління, можливість одночасного управління декількома роботизованими системами, економія часу, можливість управляти важкими роботизованими системами поодинці, навіть якщо людина не може цього зробити фізично.

До недоліків таких систем також можна віднести: дорожнеча, залежність від електрики, складність монтажу і ремонту в разі збою,

наявність помилок при виконанні команд, ризик перехоплення управління третіми особами, складність масштабування або зміни вмонтованої системи.

1.5. Постановка задачі

У даній кваліфікаційній роботі основне завдання – розробити програмне забезпечення, що дозволяє управляти роботизованою системою розумного будинку за допомогою голосових команд. Програмне забезпечення, що використовує нейронну мережу, навчену за допомогою генетичного алгоритму, має розпізнавати, розрізняти і розуміти голосові команди. Відправляти сигнали, відповідні голосовим командам, в роботизовану систему розумного будинку, яка, в свою чергу, повинна відпрацювати функції, відповідні сигналам.

Висновок до розділу

Таким чином, в розділі описані проблеми систем розумного будинку, описані використання роботизованих систем і функціональна модель роботизованої системи, що розглядається в системі розумного будинку, обраної для даної кваліфікаційної роботи. Перераховані переваги і недоліки систем розумного будинку, а також чітко сформульовані цілі цієї роботи. Це дозволяє розглянути існуючі аналоги, їх різновиди та способи управління ними.

2. ОГЛЯД НАЯВНИХ АНАЛОГІВ

Підбір існуючих аналогів заснований на певній послідовності дій. Він включає в себе опис існуючих ідентичних систем, за винятком систем, які не відповідають критеріям відбору.

Щоб вибрати серед існуючих аналогів технології розумного будинку, ми визначимо кілька конкретних критеріїв, за якими має бути рішення РД. Перший і найголовніший критерій – це управління роботизованою системою розумного будинку за допомогою голосових команд. Ми порівнюємо тільки ті рішення РД, в яких доступна ця технологія, щоб виявити недоліки існуючих систем і розробити більш ефективну. Наступний критерій – наявність роботизованої системи розумного будинку. Тобто в функціональності РД повинні бути рухливі елементи, які управляються певними сигналами. Ще один критерій – доступність. Ми дивимося на системи, які є або будуть доступні для населення в цілому в повсякденному житті, а не для підприємств або заможних домовласників, тому ми вибираємо тільки ті, які доступні більшості звичайних домовласників в середній країні, що розвивається.

Також необхідно буде визначити більш ефективну модель управління роботизованими системами розумного будинку за допомогою голосових команд за певними критеріями. По-перше, модель повинна бути стійкою до змін тембру або гучності голосу людини, який вимовляє команду. По-друге, модель повинна розпізнавати зі всіх записаних звуків саме ті голосові команди, які були виголошені. По-третє, швидкість розпізнавання голосових команд і відпрацювання їх РС повинна бути в межах середнього часу прийнятного для людини.

2.1. Існуючі системи розумного дому

Системи розумного будинку розрізняють за типами управління. Існують статичні і автоматизовані системи. Статичні – системи, які управляються

шляхом контролю кожної дії людиною. Автоматизовані системи – це системи, які виконують більшість дій незалежно, керівнику доводиться докладати менше зусиль для управління.

Amazon Alexa. «Алекса» – найпопулярніша і одна з найпоширеніших систем розумного будинку в світі. Сьогодні з нею можуть працювати кілька сотень пристроїв, і їх кількість постійно зростає. Чому? Та просто тому, що їх дуже легко створювати і управляти ними, вона прекрасно розуміє голосові функції і не змушує свого власника замислюватися про тонкощі всіх цих сучасних технологій. Загалом, можна просто із задоволенням користуватися.

Однак для українців це буде пов'язано з дещо обмеженими можливостями – наприклад, покупка продуктів та інших товарів, замовлення на послуги будуть обмежені. І вам доведеться вивчити англійську – «Алекса» не розуміє ні російської, ні української.

Google home. Ця система була представлена відносно недавно, але вже працює з багатьма пристроями – такими як Philips, TP-Link, Nest і багатьма іншими. Вона має ті ж переваги, що і «Алекса», за винятком того, що у вас буде трохи менше варіантів для підключення пристроїв, сервісів та послуг, щонайменше, до тієї пори, коли виробники не розширять цей список. На жаль, у цієї системи теж є обмеження – і знову ж таки, такі самі як у аналога Amazon.

Apple Home Kit. Не обійшлося і без розумного будинку від «яблучного» виробника. Ця система може мати не так багато пристроїв, як попередні, але в ній також є все необхідне – від розеток і вимикачів до охоронних систем безпеки і камер спостереження. До речі, незабаром виробник має намір випустити власний бездротової динамік для управління ним через Siri, тобто голосом.

Додатковим плюсом для нас є те, що Apple Home Kit входить в трійку кращих, які можуть працювати російською мовою, тому користуватися програмою набагато простіше і зручніше, ніж англійською для українців, але все ж таки українську ця система також не розумітиме.

Xiaomi Smart Home. Не обійшлося і без Xiaomi – схоже, жоден прибутковий сегмент цей виробник не упускає! Сьогодні ця екосистема не найбагатша на пристрої, але, з огляду на продуктивність Xiaomi, це тільки питання часу. Крім того, в Xiaomi Smart Home вже є все необхідне, включаючи купу датчиків і розумну техніку.

Додаток простий і зручний, але голосовий помічник нам поки не буде доступний – він розуміє тільки китайську мову. Таким чином, ця система призначена для тих, хто використовує її традиційним способом, тобто через екран смартфона.

Комплект безпеки Orvibo. Ще одна цікава бюджетна система розумного будинку. Його назва означає систему безпеки, але завдяки сучасним хмарних технологій він також може виступати в якості контролера для розумного будинку. І він настільки хороший, що в використанні мало відрізняється від традиційних аналогів.

У Orvibo є власна екосистема, яка включає навіть розумні дзеркала, але це ще не все. Контролер заснований на протоколі ZigBee, який використовують багато сторонніх виробники, тому ви не будете обмежені у виборі. Ну і як додатковий плюс – зручний інтерфейс з повністю російським інтерфейсом – шкода, що без голосового помічника.

Які бувають системи автоматизації «розумного будинку» за основними ознаками. Можна виділити три основні параметри:

- Централізовані або децентралізовані;
- З відкритим протоколом або з закритим протоколом;
- Дротові або бездротові.

Відповідно, ці параметри можуть бути комбінаціями, наприклад: «Розумний будинок з дротовим підключенням, децентралізований з відкритим протоколом» або «Розумний будинок з бездротовим підключенням, централізований з закритим протоколом». Але по порядку.

Централізовані системи автоматизації. Суть централізованої системи управління полягає в тому, що тут відбувається програмування одного центрального елементу (логічного модуля). Часто це контролер з безліччю виходів. Програма, що управляє приводами і системами, написана для конкретного об'єкта. Потім ця програма заливається в контролер. Ця схема дозволяє використовувати сценарії багатозадачності з використанням найрізноманітнішого обладнання. Централізовані системи автоматизації можуть бути як дротовими, так і бездротовими.

Децентралізовані системи автоматизації. У децентралізованій системі кожен вузол містить мікропроцесор, який, в свою чергу, містить незалежну пам'ять. Це дозволяє досягти досить високої надійності для цих систем.

Відповідно, ми розуміємо, що при руйнуванні одного пристрою вся система, за замовченням, працює правильно, за винятком тих пристроїв, які були підключені до пристрою, яке було пошкоджено. Прикладом децентралізованої системи є розумні будинки KNX.

Системи автоматизації з відкритим протоколом. Не бійтеся слова «протокол» – це не протокол, складений поліцією. У нашому випадку це мова, якою спілкуються всі пристрої у вашій системі. Якщо взяти улюблений і шанований кожним протокол KNX, він відкритий, а це значить, що величезна кількість виробників обладнання для розумного будинку роблять пристрої, які працюють на цій мові. Асоціація KNX перевіряє всі вироблені пристрою, і знак KNX на пристрої гарантує високу якість продукції. Хоча вартість його вище, ніж в системах на закритих протоколах. За гарантію якості доведеться платити. Та й блукаючі в своїх творчих задумах інженери при створенні нових пристроїв не керуються стандартами асоціації.

Системи автоматизації з закритим протоколом. Ці системи, реалізовані на основі власних закритих протоколів, існують в основному для спрощення процесу програмування. І фінансові витрати виробників таких систем будуть нижче – немає необхідності комусь платити за схвалення їхнього продукту.

Підкреслю, що обладнання, яке працює по закритому протоколу, співпрацює лише з рідним виробником. З одного боку, це дає необмежений політ фантазії виробника, з іншого – повна залежність споживача від одного і того ж виробника.

Сподіваюся, що я підняв завісу за цим «страшним звіром» під назвою «розумний дім». Сучасний світ систем домашньої автоматизації надзвичайно широкий і різноманітний. А головне – він стрімко розвивається і набирає популярність. І не за горами цей день, коли в житті з'являться дуже розумні будинки, які ми бачимо в голлівудських фільмах.

Дротові системи автоматизації. Як бачите, назва говорить сама за себе. Сенс дротової системи полягає в наступному: всі пристрої управління – перемикачі, панелі управління, різні датчики і пристрої – пов'язані єдиною шиною. Вони подають команди (сигнали, потоки даних) на виконавчі пристрої (реле, виконавчі механізми, двигуни, мікроконтролери), які знаходяться в щитовій (або у командному блоці) на цій шині. У ролі шини може виступати спеціальний кабель, а іноді і звичайна вита пара.

Системи бездротового автоматизації. Знову ж з назви ми розуміємо, що команди від керуючого пристрою до виконавчого механізму в такій системі приходять не по дротах, а по радіосигналу. Відсутність дротів економить місце, економить час на установку і заощаджує гроші за рахунок відсутності цих дротів та робіт з їх монтажу. Головний козир бездротових систем – можливість монтажу з готовим ремонтом, адже це не потребує прокладання дротів.

2.1.1. Статичні елементи системи розумного дому

Серед різних систем управління розумним будинком велика кількість статичних. Наприклад, термостати – системи контролю температури повітря, пульти дистанційного керування, вимикачі світла, вентиляції, розетки. В цілому у цих систем є кілька недоліків. По-перше, вони завжди вимагають,

щоб ними керувала людина, по-друге, всі ці елементи управління залежать від джерела живлення, по-третє, ці системи мають просторові обмеження (перемикачі і термостати закріплені в певних місцях, а дистанційне керування має обмежену дальність передачі сигналу).

2.1.2. Автоматизовані системи розумного дому

Автоматизовані системи були створені для усунення недоліків систем статичного контролю для розумних будинків. Слово «автоматизований» вже говорить сама за себе. Це системи, які можуть аналізувати ситуації, сценарії і приймати рішення на основі аналізу. До таких систем відносяться, наприклад, роботи-прибиральники, датчики руху для включення світла, голосові помічники для управління розумними будинками відомих компаній, таких як Alexa від Amazon, Google Assistant від Google, Apple Home Kit від Apple.

2.2. Управління системами розумного дому

Управління можна здійснювати декількома способами. Перший передбачає управління за допомогою спеціальної клавіатури. На ньому є клавіші, при натисканні яких відбувається певна дія. У деяких випадках такі клавіатури оснащуються моніторами, що відображають інформацію по кожному пристрою, включеному в систему. Також в цьому випадку можна використовувати голосове управління розумним будинком. Є ще один спосіб управління програмою за допомогою сенсорного екрану. На екрані такого пристрою відображається інформація про стан різних систем будинку і зображення з відеокамер. Останні розробки дозволяють дистанційно керувати через Інтернет.

Існує три основні методи управління вищевказаними системами:

- управління за допомогою сценарію, прописаного в програмному забезпеченні (soft);

- за допомогою обладнання, такого як пульт дистанційного керування або смартфон (hard);
- за допомогою голосових або інших звукових команд (звук).

На рис. 2.1 (додаток Д) показана приблизна схема управління роботизованими системами розумного будинку за допомогою голосових команд.

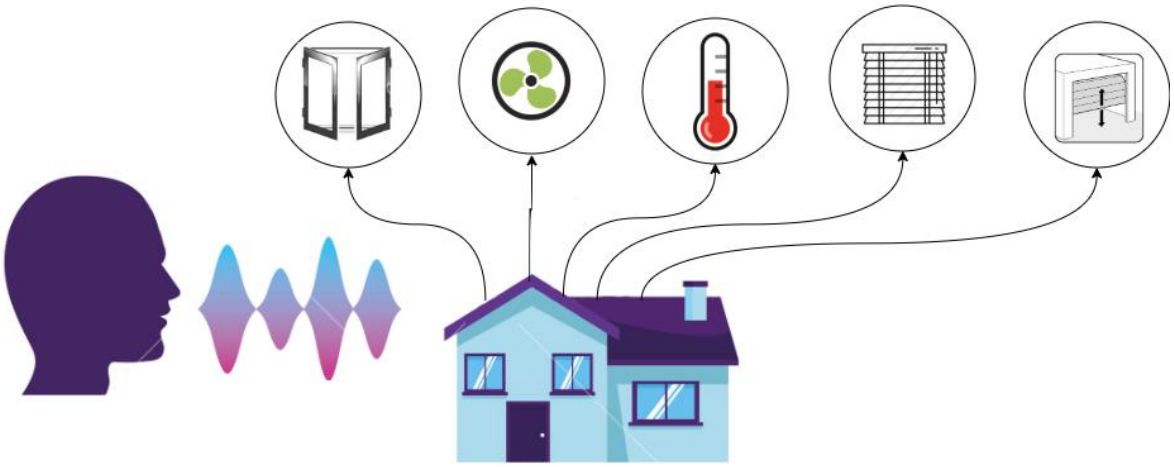


Рисунок 2.1 – Схема управління розумним будинком голосовими командами

2.3. Підсумки вимог до програмного забезпечення

Рекурентне програмне забезпечення нейронної мережі, яке регулює ваги за допомогою генетичного алгоритму, має розпізнавати, розрізняти і розуміти голосові команди. Відправляти сигнали, відповідні голосовим командам, в роботизовану систему розумного будинку, яка, в свою чергу, повинна відпрацювати функції, відповідні сигналам.

Таким чином, програма повинна отримувати вхідні дані у вигляді звуку, записаного моно-мікрофоном (один канал) або стерео-мікрофонами (два канали). Далі програма повинна відокремити голосову доріжку від шуму, відокремити всі команди одну від одної у отриманих вхідних даних і з

урахуванням різних тембрів і різної гучності голосів розпізнати їх. Після отримання чітких команд програмне забезпечення обробляє їх в роботизованій системі відповідно до заданої логікою.

Висновок до розділу

Таким чином, були розглянуті існуючі моделі роботизованих систем розумного будинку, відібрані серед усіх такі, які відповідають критеріям даної кваліфікаційної роботи, проаналізовані ці моделі і з'ясовано, які чинники з'явилися для підвищення ефективності:

- швидкість розпізнавання;
- шумостійкість;
- стійкість до змін частоти і довжини хвилі голосу.

Ефективність цих критеріїв буде підвищена за рахунок використання рекурентної нейронної мережі, ваги якої коригуються за допомогою генетичного алгоритму. Наступним кроком є вивчення аналітичних методів обробки вхідної інформації, моделювання архітектури рекурентної нейронної мережі і генетичного алгоритму, вивчення ефективності використання генетичних алгоритмів для налаштування ваг нейронної мережі.

3. АНАЛІТИЧНІ МЕТОДИ ОБРОБКИ ВХІДНОЇ ІНФОРМАЦІЇ

У системах розпізнавання голосових команд є дві основні підсистеми:

- система попередньої обробки голосових команд;
- система класифікації голосових команд.

Перша підсистема готує дані для подальшого використання нейронною мережею. Друга підсистема займається спочатку навчанням розпізнаванню голосових команд, а потім класифікацією. В якості другої підсистеми обрана рекурентна нейронна мережа.

3.1. Попередня обробка вхідних даних

На рис. 3.1 показана схема попередньої обробки голосових команд. Обробка розділена на п'ять частин:

- Введення голосової команди;
- Попередня фільтрація;
- Нарізка команди на кадри;
- Обробка кадрів у віконній функції;
- Нормування даних в кадрах.

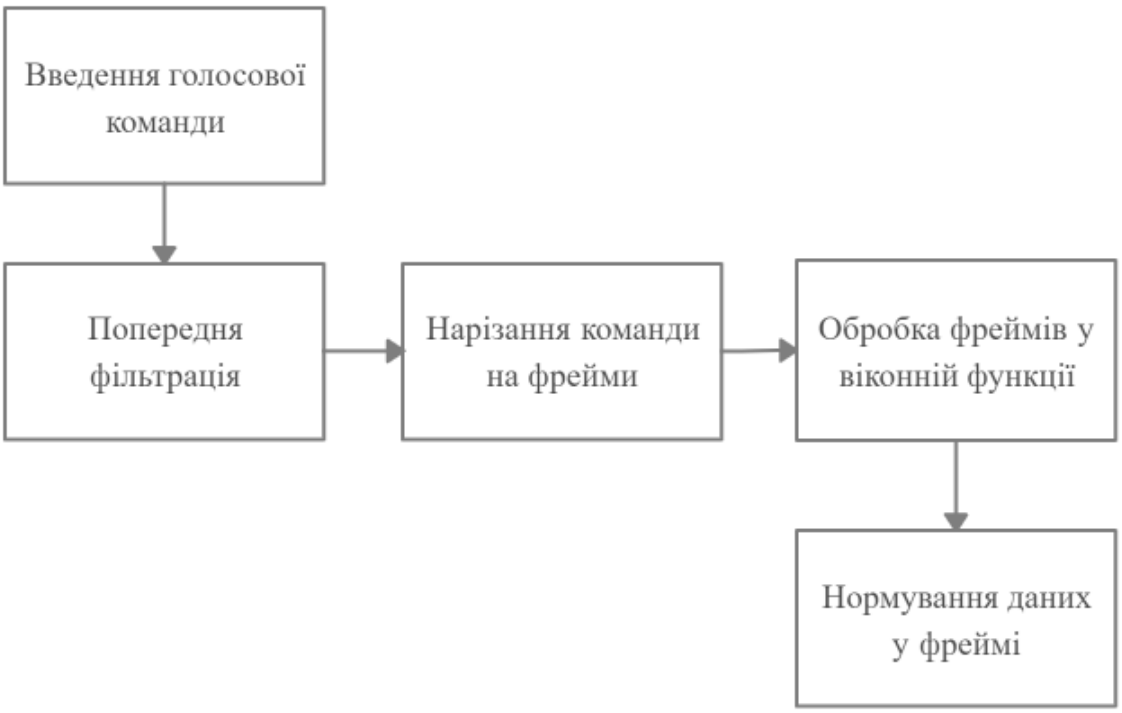


Рисунок 3.1 – Схема алгоритму попередньої обробки голосових команд

Докладніше про кожну частину обробки вхідної інформації.

По-перше, введення команд відбувається в одному потоці, тому вони відокремлюються одна від одної шляхом аналізу тиші за певний період часу до промовляння команди і після.

По-друге, крім необхідної системі команди, разом з нею мікрофони вловлюють різні шуми. Шум знижує точність системи розпізнавання, тому вхідні сигнали необхідно фільтрувати. У роботі використано смуговий фільтр для фільтрації сигналу, який являє собою комбінацію фільтрів низьких і високих частот. Цей фільтр пропускає тільки ті частоти, які вище, ніж так звана нижня частота проходу, і нижче, ніж частота верхнього проходу. На рис. 3.2 ви можете побачити приблизний розклад смугового фільтра.

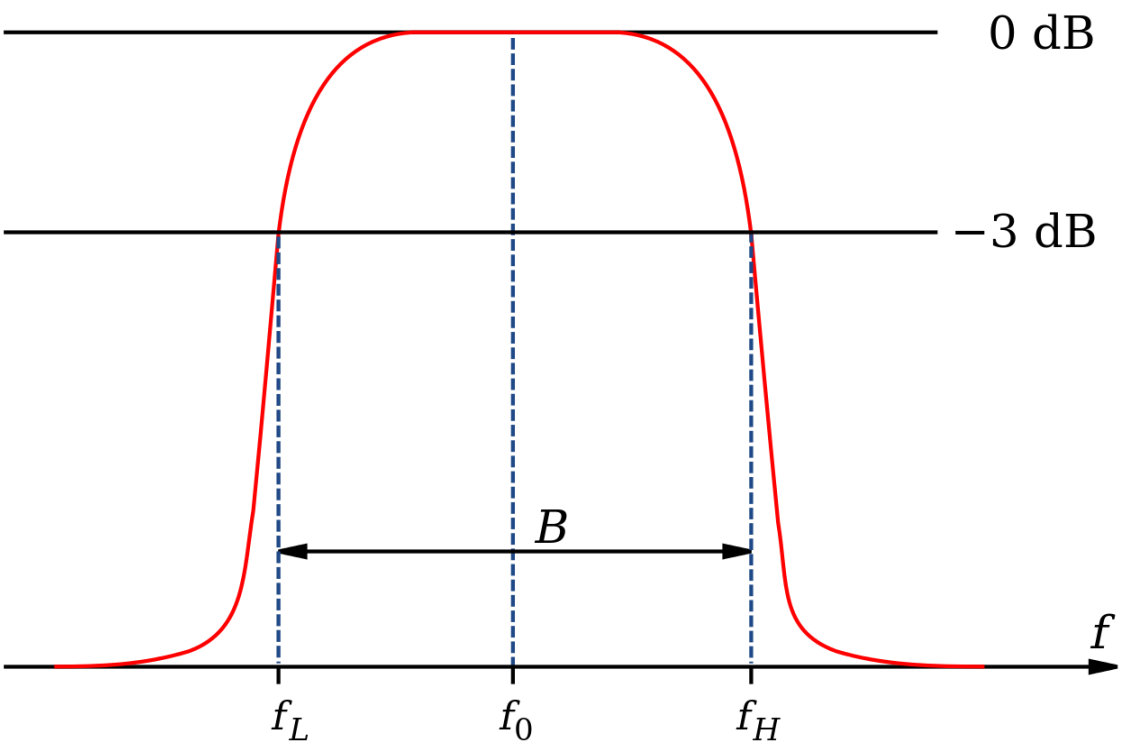


Рисунок 3.2 – Приклад розкладу смугового фільтра

По-третє, щоб надати нейронній мережі дані для обробки, вам потрібно розкласти одну команду на фіксовану кількість кадрів однакової довжини. Кількість кадрів, на які розділяється одна команда, дорівнює кількості нейронів у вхідному шарі нейронної мережі, в цій роботі їх десять. Розмір вектора даних в одному кадрі залежить від розміру вектора даних в самій команді і дорівнює одній десятій розміру повної команди.

По-четверте, щоб зменшити небажані ефекти відсічення, що виникають в результаті сегментації сигналу, необхідно помножити сигнал на віконну функцію. Це підвищить точність системи розпізнавання голосових команд. Віконних функцій існує багато, ось найвідоміші з них:

- Прямокутне вікно;
- Вікно Ханна;
- Вікно Хеммінга;
- Вікно Блекмана;
- Вікно Кайзера.

У цій роботі вікно Хеммінга було обрано в якості віконної функції. Функція підбирається емпіричним шляхом.

По-п'яте, всі обчислення в нейронних мережах здійснюються над числами з плаваючою комою, тому значення параметрів об'єктів, які будуть класифікуватися нейронною мережею, обмежені інтервалом [0; 1]. Для цього вектор цих кадрів нормалізується до 1.0, тобто кожен елемент вектора ділиться на максимальний елемент вектора.

3.2. Розпізнавання голосових команд за допомогою нейронної мережі

Існує велика кількість варіантів і підходів для реалізації нейромережевого розпізнавання голосових команд. Для цього завдання використовуються різні типи нейронних мереж, різні архітектури, ваги мережі налаштовуються по-різному, і існує кілька способів навчання мереж.

На сьогоднішній день найбільш поширеним методом розпізнавання голосових команд є варіант зі згортковими нейронними мережами, які отримують спектральне подання звуку (амплітудні спектри).

У цій роботі використано рекурентні нейронні мережі, у яких є аналог пам'яті. Мається на увазі, що нейрони можуть отримувати дані від попередніх ітерацій. Це дозволить мережі «запам'ятати» вже розраховані частини команди, щоб швидко обчислити частини команди, що залишилися. Такий підхід повинен прискорити роботу мережі і її навчання.

До речі про навчання, продиктована деяка база команд з різними варіаціями вимови команд. Ця база даних є навчальною вибіркою. Навчання відбувається шляхом поступового надання мережі навчальних вибірок з одночасним коригуванням ваги з використанням генетичного алгоритму до тих пір, поки функція помилки не досягне свого найнижчого рівня.

3.2.1. Топологія нейронної мережі

Як згадувалося вище, на вхідний шар нейронної мережі подаються кадри голосової команди, тому кількість вхідних нейронів відповідає кількості кадрів, на які ділиться голосова команда, нехай ця кількість буде I нейронів. Мережа повністю взаємозалежна, що означає, що кожен нейрон в одному шарі пов'язаний з кожним нейроном в сусідньому шарі. Також в мережі є прихований шар, який пропускає J нейронів.

Прихований шар складається з декількох шарів наступній послідовності:

- Повноз'єднаний шар з кількістю нейронів N_1;
- Dropout шар з ймовірністю 40%;
- Повноз'єднаний шар з кількістю нейронів N_2;
- Повноз'єднаний шар з кількістю нейронів N_3;
- Повноз'єднаний шар з кількістю нейронів N_4;
- Dropout шар з ймовірністю 50%;
- Повноз'єднаний шар з кількістю нейронів N_5;

Всі повнозв'язні шари, крім останнього, мають функцію активації – функцію ReLU, останній шар – має функцію активації – Softmax. За першим і п'ятим рівнями також слідують dropout шари, щоб запобігти перенавчання мережі. Dropout шари забезпечують ігнорування з певною ймовірністю деяких нейронів в шарі за одну ітерацію. Це дозволяє мережі обробляти помилки і не покладатися на існування конкретного нейрона або групи нейронів, а покладатися на консенсус нейронів в одному шарі. Це досить простий і ефективний метод, що дозволяє впоратися з проблемою перенавчання без додаткових регуляризаторів. На рис. 3.3 показано приклад dropout шару.

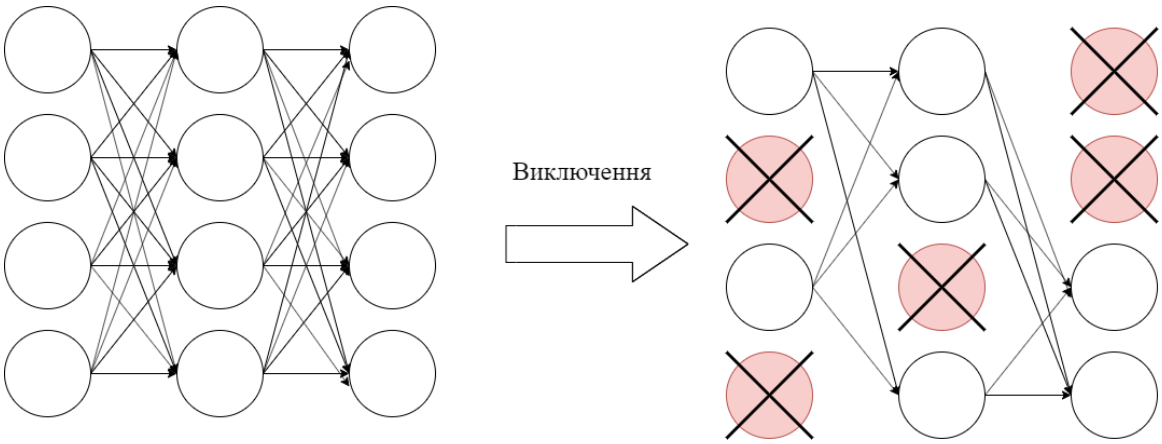


Рисунок 3.3 – Демонстрація роботи шару dropout в нейронної мережі

Після вхідного і прихованого шарів в мережі йде вихідний шар (кількість нейронів K), який повертає відсоток точності розпізнаної команди. На рис. 3.4 ви можете побачити структуру і топологію використовуваної рекурентної повноз'єднаної нейронної мережі.

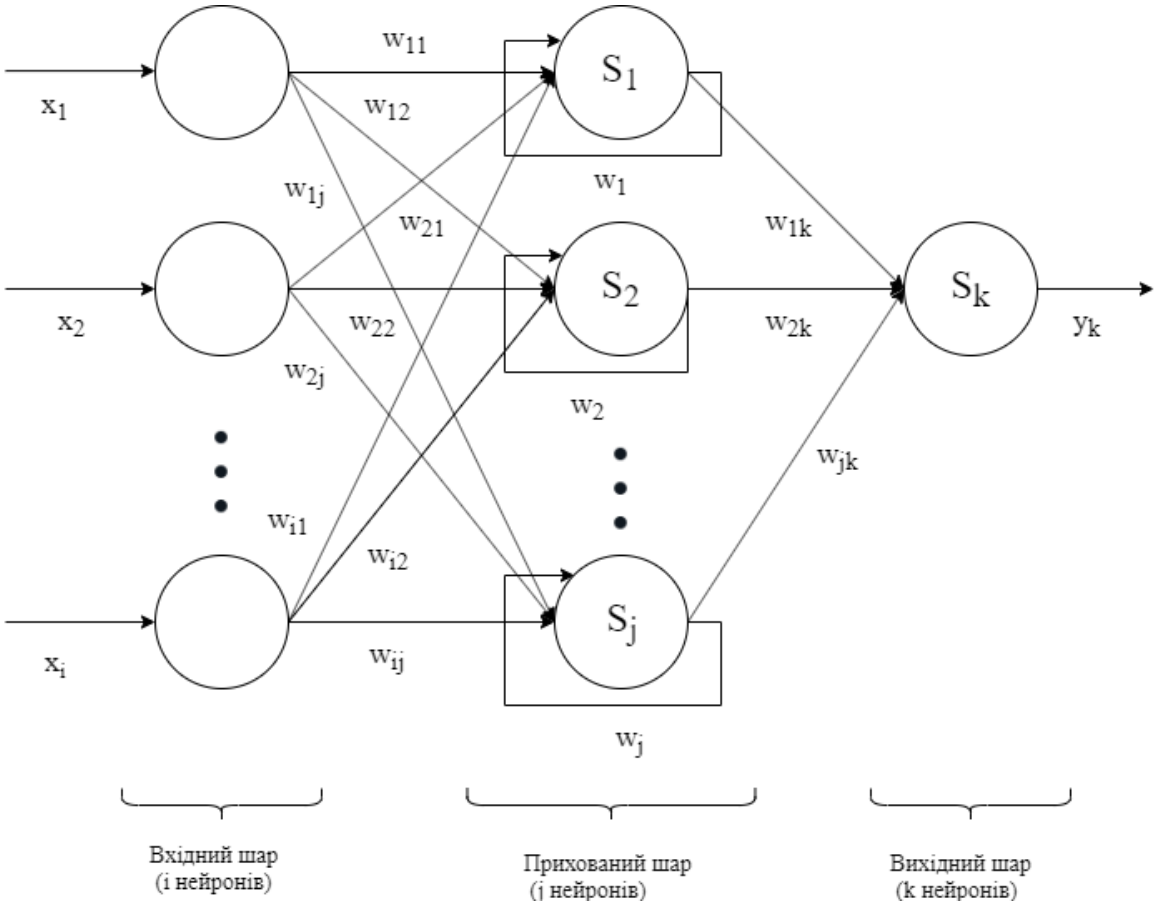
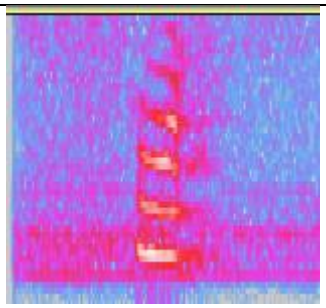
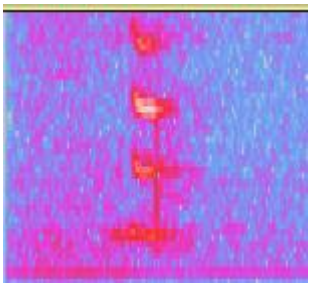
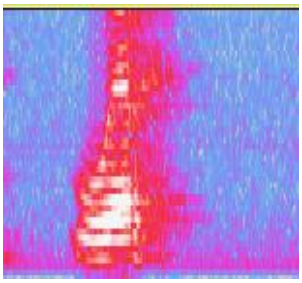
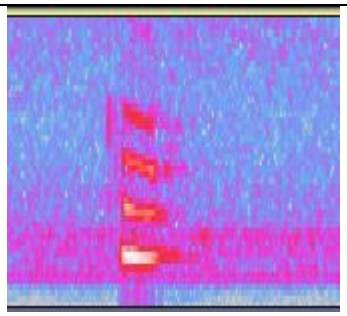
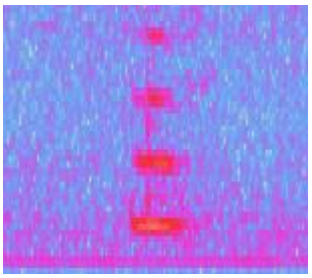
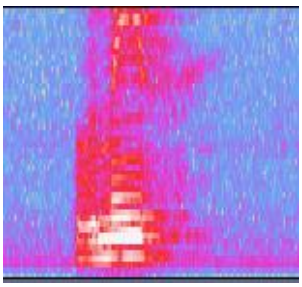
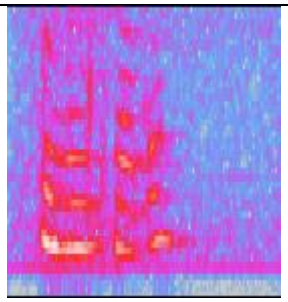
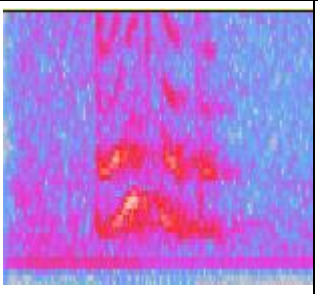
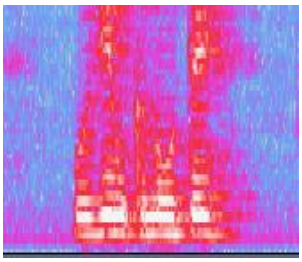
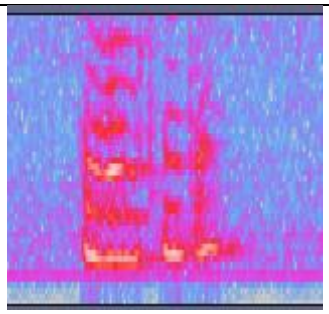
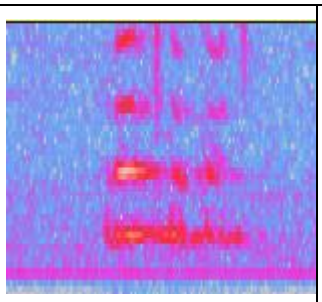
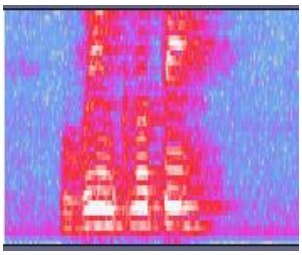
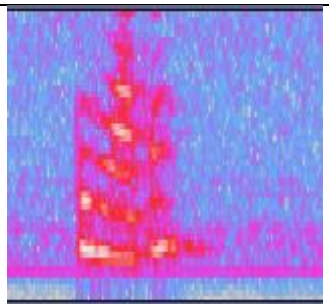
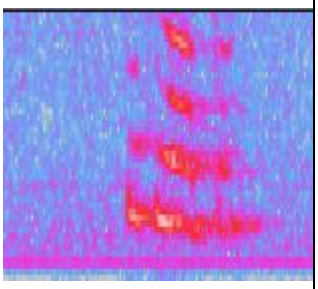
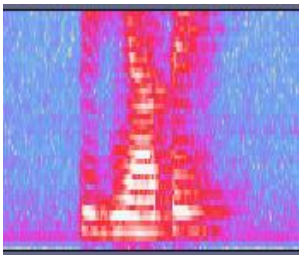
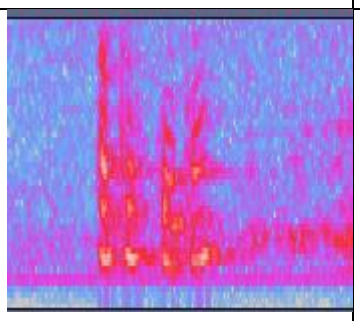
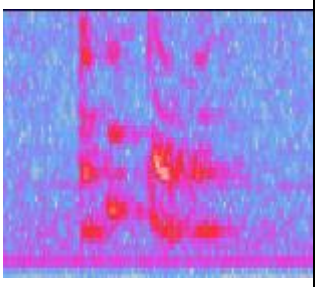
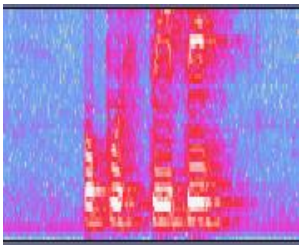
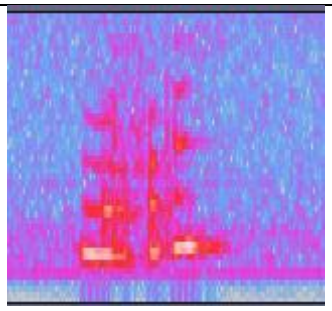
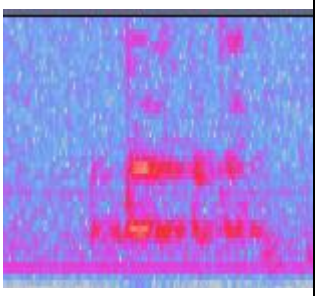
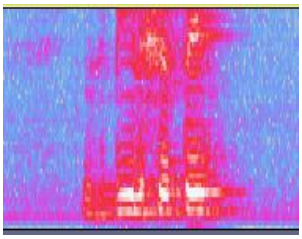
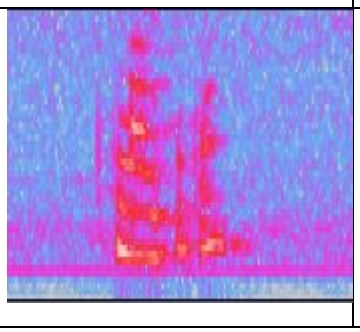
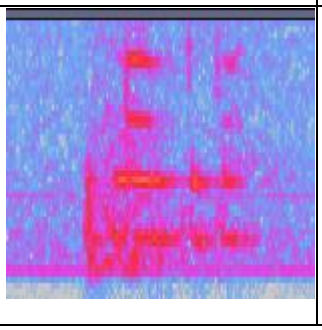
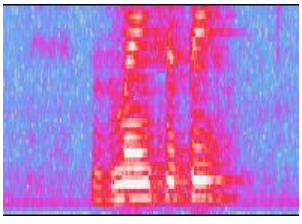


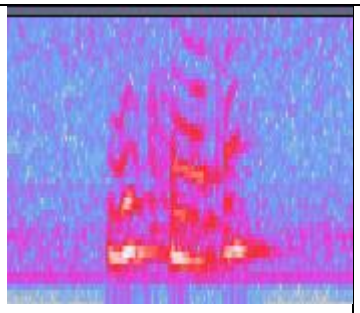
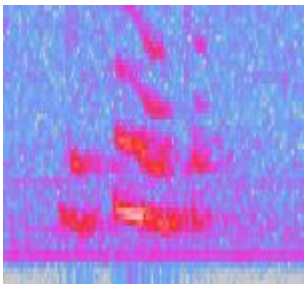
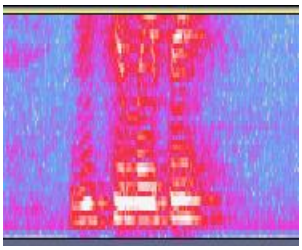
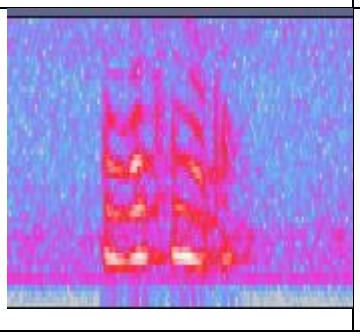
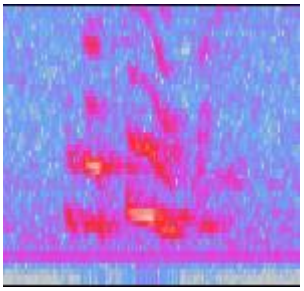
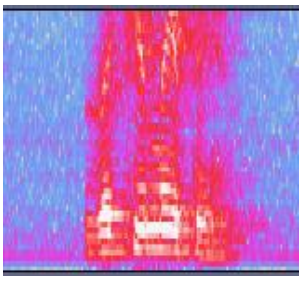
Рисунок 3.4 – Демонстрація топології і структури нейронної мережі

Навчання відбуватиметься з учителем, тобто мережа знатиме, яка команда була сказана, але коригування ваг буде відбуватися не за класичним алгоритмом зворотного поширення помилки, а за допомогою генетичного алгоритму. Дані для навчання мережі складатимуться з 30 аудіозаписів на кожну голосову команду. З цих 30 записів 10 вимовлятиму я, 10 моя сестра, 10 батько. Запис тренувальних команд відбуватиметься з різною інтонацією та гучністю. Усереднені спектрограми тестових голосових команд можна побачити в таблиці 3.1. Такий підхід дозволить скоротити час навчання і вирішити проблему застрягання функції помилки в локальних мінімумах.

Таблиця 3.1 – Усереднені спектрограми тестових даних

Спікер / Команда	Єгор	Дар'я	Андрій
«Вгору»			
«Вниз»			
«Увімкнути»			

«Вимкнути»			
«Підняти»			
«Опустити»			
«Збільшити»			
«Зменшити»			

«Відчинити»			
«Зачинити»			

3.3. Методи навчання нейронної мережі

Отже, перед нами стоїть завдання навчити нейронну мережу. Є два найбільш поширені методи навчання нейронних мереж:

- Зворотне поширення помилки (Backpropagation);
- Використання генетичних алгоритмів.

3.3.1. Метод зворотного поширення помилки

Однак у обох цих методів є свої плюси і мінуси. Недоліком методу зворотного поширення помилок є «застрягання» функції помилки в локальних мінімумах, з котрих вона не може вибратися, а також тривалий час навчання. Тобто, якщо ви подивитеся на рис. 3.5, ви побачите графік функції помилки при backpropogation.

Рисунок 3.5 – Графік функції помилки нейронної мережі при використанні методу зворотного поширення помилки

Графік показує, що функція помилки може потрапляти в локальний мінімум замість глобального. Після тривалого перебування на мінімумі функція намагається вийти за допомогою різкого збільшення ваги («стрибок»). Якщо різке збільшення ваги дало меншу функцію помилки, ніж теперішня, то функція продовжує поступово спадати. Таким чином, йде пошук глобального мінімуму, але цей метод навчання вимагає багато часу, іноді цей час може тривати кілька днів або навіть тижнів, залежно від топології мережі та обсягу даних для тренування.

Тобто у цього методу, цього алгоритму навчання, є два основних недоліки, яких необхідно уникнути при використанні генетичного алгоритму. Нижче наводиться порівняльний опис методів (генетичний алгоритм і алгоритм зворотного поширення помилки).

3.3.2. Генетичний алгоритм

Відомо, що генетичний алгоритм є ітеративним і дає тільки приблизне рішення, що компенсується його сферою застосування і швидкістю обчислень

при обмежених обчислювальних ресурсах. Він може використовуватися для підналаштування ваг прихованих і вихідних шарів з фіксованим набором зв'язків і широко використовується в задачах оптимізації та навчання нейронних мереж, обробки безлічі альтернативних рішень, при цьому пошук орієнтований на найбільш перспективні.

Генетичний алгоритм оперує такими визначеннями в даній роботі:

- ген – ваговий коефіцієнт нейронної мережі;
- хромосома або генотип – набір генів (тобто ваги нейронної мережі, що прочитуються в певному порядку зверху вниз, справа наліво). Кожна хромосома – можливе рішення (набір ваг, який краще підходить для розпізнавання голосових команд);
- фенотип – сукупність фізичних характеристик, притаманних індивіду;
- популяція – багато хромосом, варіанти наборів ваг;
- епоха – ітерація, відповідне створення нового покоління хромосом. Хромосоми – це основні сутності, над якими такі операції виконуються в певному порядку в межах однієї епохи;
- схрещування – створення з певним ступенем ймовірності (P_c) нової хромосоми з генів двох батьківських і додавання її в популяцію;
- мутація – зміна з певним ступенем ймовірності (P_m) значення довільного гена будь-якої хромосоми і додавання його в популяцію;
- адаптація – видалення з популяції хромосом (тобто наборів ваг), що показали найгірший результат в розпізнаванні (пристосування).

Мутація вирішує одну з проблем, що виникають при навчанні багатошарової нейронної мережі методом зворотного поширення помилок. Вона необхідна для виведення популяції з локального екстремуму і запобігає

передчасної збіжності. Оскільки генетичний алгоритм не є строго детермінованим, можна виконувати схрещування і мутації в довільному порядку в межах однієї епохи. В рамках цього завдання найбільш ефективним виявився порядок, в якому спочатку здійснюється схрещування, а потім мутація популяції. На рис. 3.6 зображено роботу генетичного алгоритму.

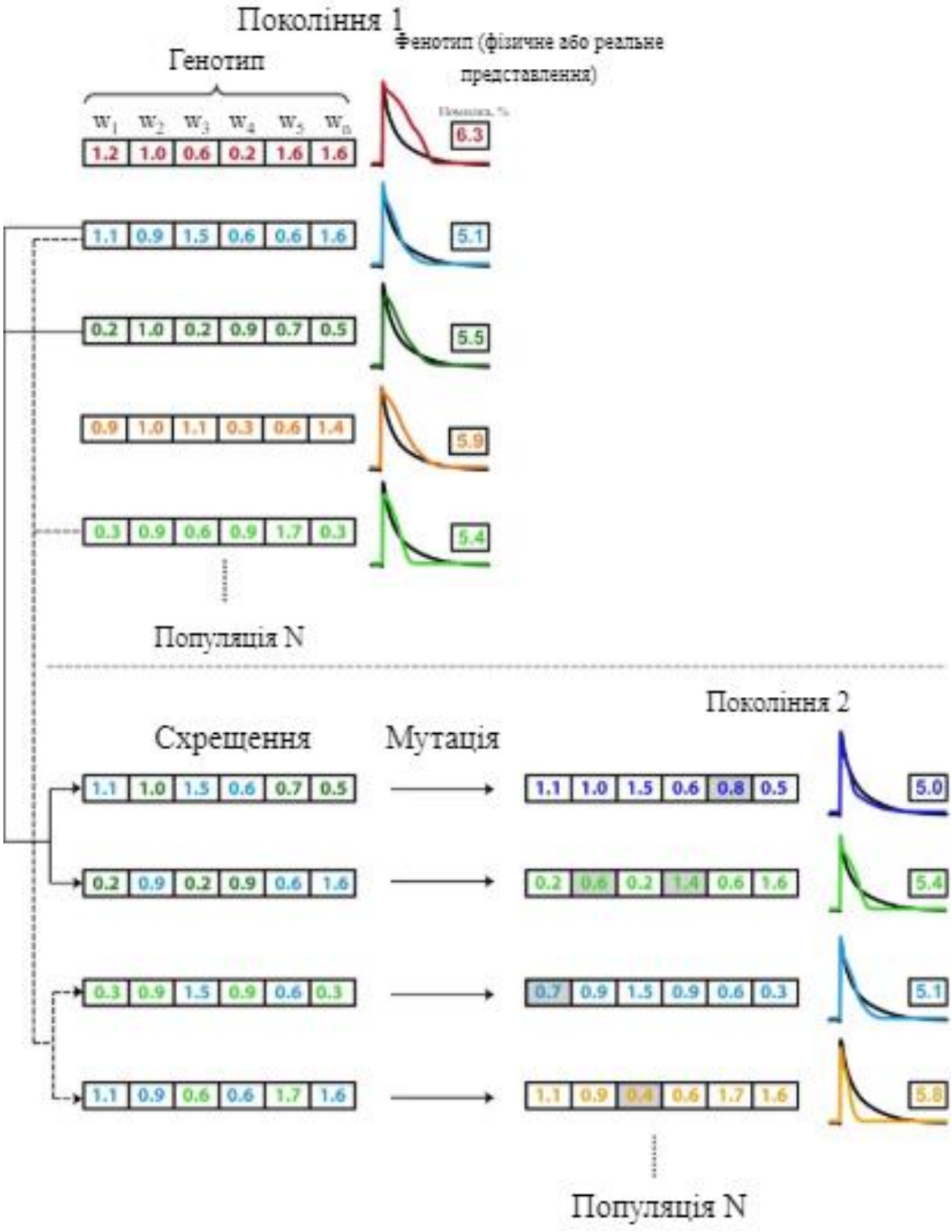


Рисунок 3.6 – Ілюстрація генетичного алгоритму

Оператор селекції (пристосування) специфічний для задачі розпізнавання голосових команд. Визначення адекватності та, як наслідок, вибір найкращої хромосоми ваг визначається помилкою розпізнавання голосових команд. Набори хромосом, які дають найменшу помилку розпізнавання, вибираються з популяції. Похибка визначається за формулою:

$$\varepsilon_i = 1 - \frac{y}{y_0}, \qquad (3.1)$$

де y_0 − еталонне значення вихідного сигналу (пронормоване за функцією softmax конкатеновані значення команди типу
int беручи коди символів); y − значення вихідного сигналу при розпізнаванні еталонної команди з навчальної вибірки з заданим набором ваг. На рис. 3.7 (додаток А) показана блок-схема логіки генетичного алгоритму.

При порівнянні роботи алгоритму зворотного поширення помилки і генетичного алгоритму на вхід нейронної мережі подається навчальний набір кадрів однакових команд; навчання закінчується, коли результуючий вихідний сигнал у всьому наборі кадрів є рівним необхідному еталонному вихідному сигналу. Для навчання нейронної мережі досить виконати 50 ітерацій з використанням генетичного алгоритму, в той час як для алгоритмів зворотного поширення потрібно більше 500 ітерацій (де одна ітерація – це повний перерахунок всіх навчальних даних: ваг, помилок, значень виходів нейронної мережі). У цьому випадку два згенерованих покоління в генетичному алгоритмі еквівалентні одній ітерації алгоритму зворотного поширення, оскільки зворотне поширення складається з двох частин – прямого проходу (обчислення виходу мережі і помилки) і зворотного проходу (зміна ваги). Генетичний алгоритм виконує тільки першу частину. Таким чином, одне згенероване покоління витрачає менше часу, ніж обчислення однієї ітерації алгоритму зворотного поширення. Загалом, генетичні алгоритми значно виграють у схем зворотного поширення в задачі розпізнавання голосових команд, дозволяючи отримувати кращі вектори ваг за значно менший час.

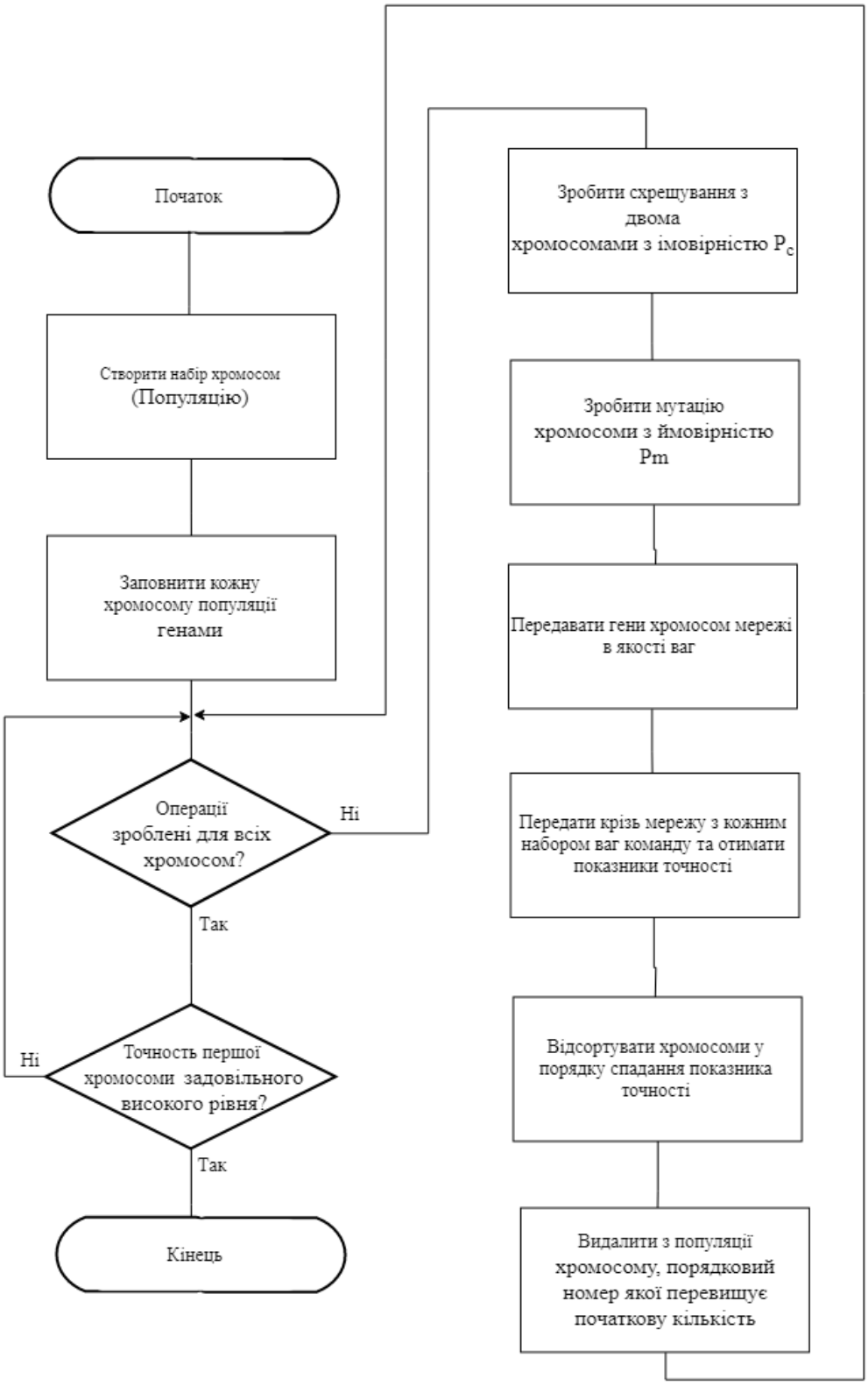


Рисунок 3.7 – Блок-схема логіки генетичного алгоритму

На кінець, в цьому розділі продемонстровано порівняльні характеристики на рис. 3.8 двох алгоритмів, побудовані у середовищі Mathcad.

Графік показує переваги методу генетичних алгоритмів в меншій кількості епох для досягнення низького рівня помилки.

Рисунок 3.8 – Порівняльна характеристика функцій помилки при різних алгоритмах навчання нейронної мережі

Висновок до розділу

Таким чином, проаналізувавши найбільш поширені методи навчання нейронних мереж і вибравши метод генетичного алгоритму, можна сказати, що це рішення дозволило уникнути проблем, що виникають при використанні більш поширеного методу зворотного поширення помилки. Генетичний алгоритм дозволяє не потрапляти в локальні мінімуми функції помилки, а також збільшувати швидкість навчання за рахунок проходження по мережі в одному напрямку. Розглянувши всю необхідну теорію і проаналізувавши всі прийняті рішення, можна переходити до написання програмного забезпечення.

4. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

Ідея розпізнавання мов завжди виглядала багатообіцяючою. Але вже на етапі розпізнавання чисел і найпростіших слів дослідники зіткнулися з проблемою. Суть розпізнавання зводилася до побудови акустичної моделі, коли вона представлялася у вигляді статистичної моделі, яку порівнювали з готовими шаблонами. Якщо модель відповідала шаблону, система вирішувала, що команда або номер розпізнані. Зростання словників, які система може розпізнавати, вимагає збільшення можливостей комп'ютерної системи.

Сьогодні алгоритми розпізнавання доповнюються мовними моделями, які описують структуру мови, наприклад, типову послідовність слів. Система навчається на реальному мовному матеріалі. Новим етапом розвитку технологій стало використання нейронних мереж. Система розпізнавання влаштована таким чином, що кожне нове розпізнавання впливає на точність розпізнавання в майбутньому. Тобто система вчиться.

4.1. Засоби розробки

Програмне забезпечення для даної кваліфікаційної роботи розроблено з використанням наступних засобів:

- Microsoft Visual Studio 2019 Professional;
- .NET 5.0;
- C# Language 9.0;
- NAudio.dll;
- IronPython.dll;
- Python 3.9
- TesnorFlow;
- Numpy;
- Keras;

- Databricks;
- Matcad;
- Audacity.

Для реалізації алгоритму в даній кваліфікаційній роботі я використовував базові класи платформи .NET 5.0 на C #. Сторонніми бібліотеками не користувався. Генетичний алгоритм реалізований універсально у вигляді бібліотеки класів (додаток В). Бібліотека містить узагальнені класи і методи для використання алгоритму з будь-яким типом даних. Приклад роботи алгоритму з текстом показаний на рис. 4.1.

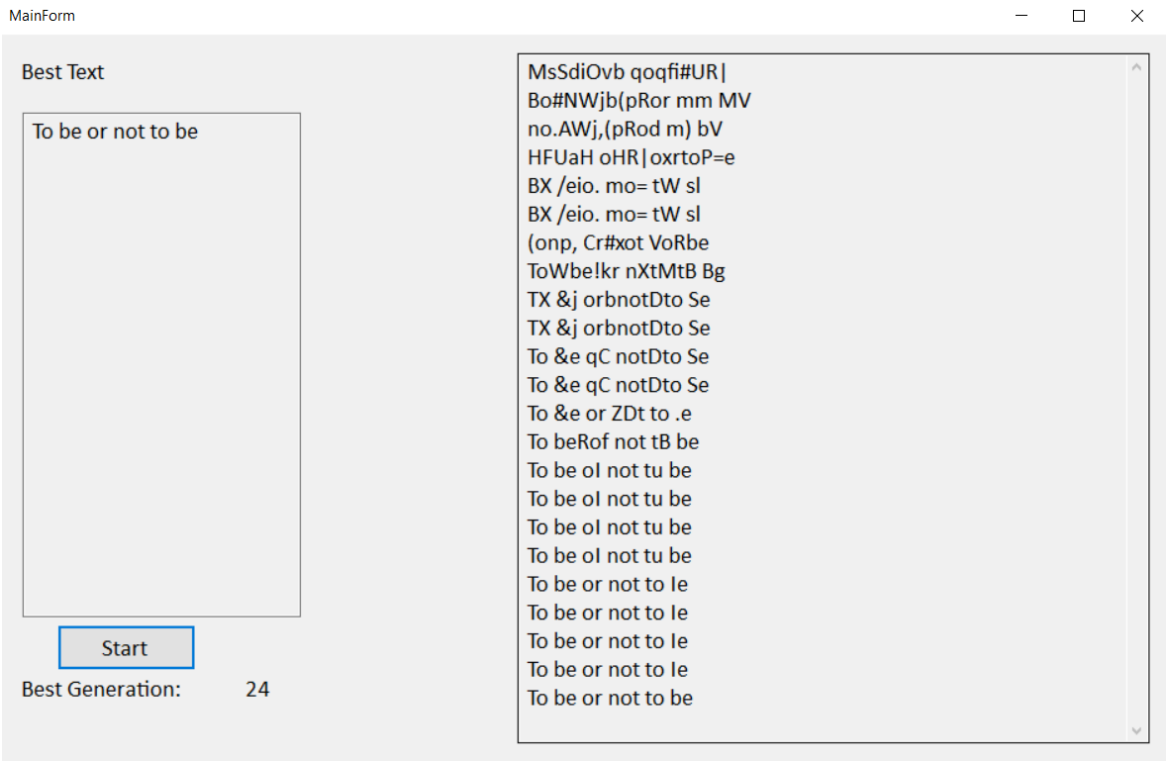


Рисунок 4.1 – Робота генетичного алгоритму з пошуку заданої текстової послідовності

Нейронна мережа була створена з використанням накладення поверх Tensorflow (пакет для Python) – keras. Мережа була навчена на сервісі Apache Spark – Databricks.

Статистика наближення результату до еталонного значення при розпізнаванні представлена на рис. 4.2.

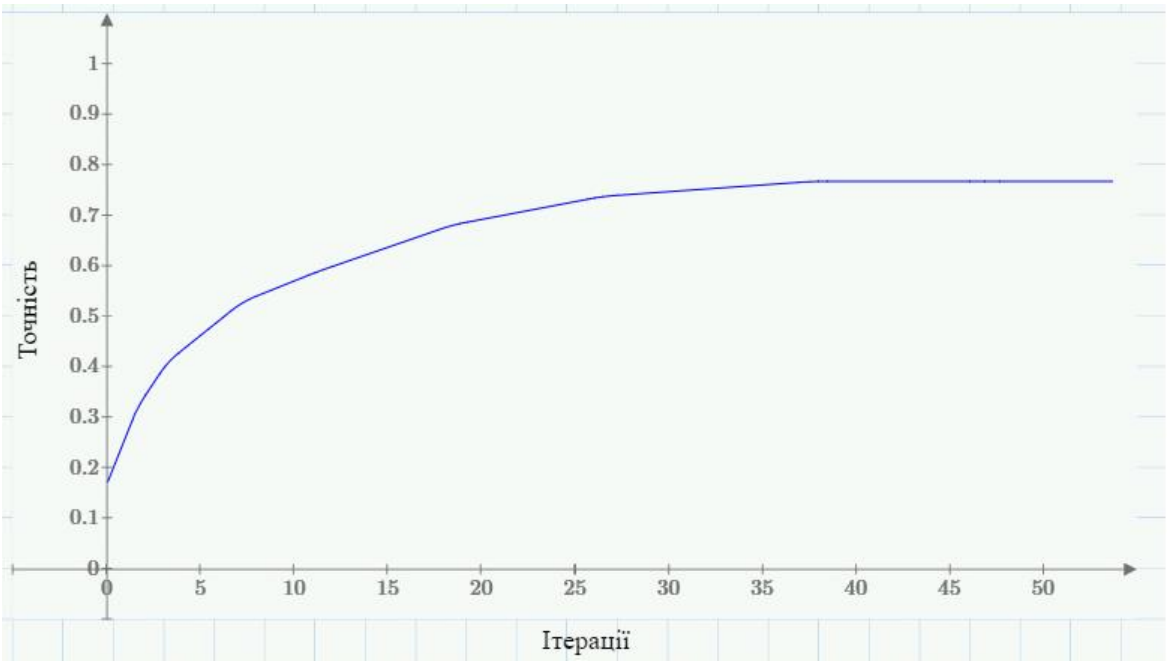


Рисунок 4.2 – Графік підвищення точності нейронної мережі при навчанні

Емпіричним шляхом було встановлено, що для досягнення найкращого результату розпізнавання з отриманим набором ваг ймовірність схрещування (P_c) становить 0,64, ймовірність мутації (P_m) – 0,01. На наступному рис. 4.3 показано графік зменшення помилки в залежності від ітерації.

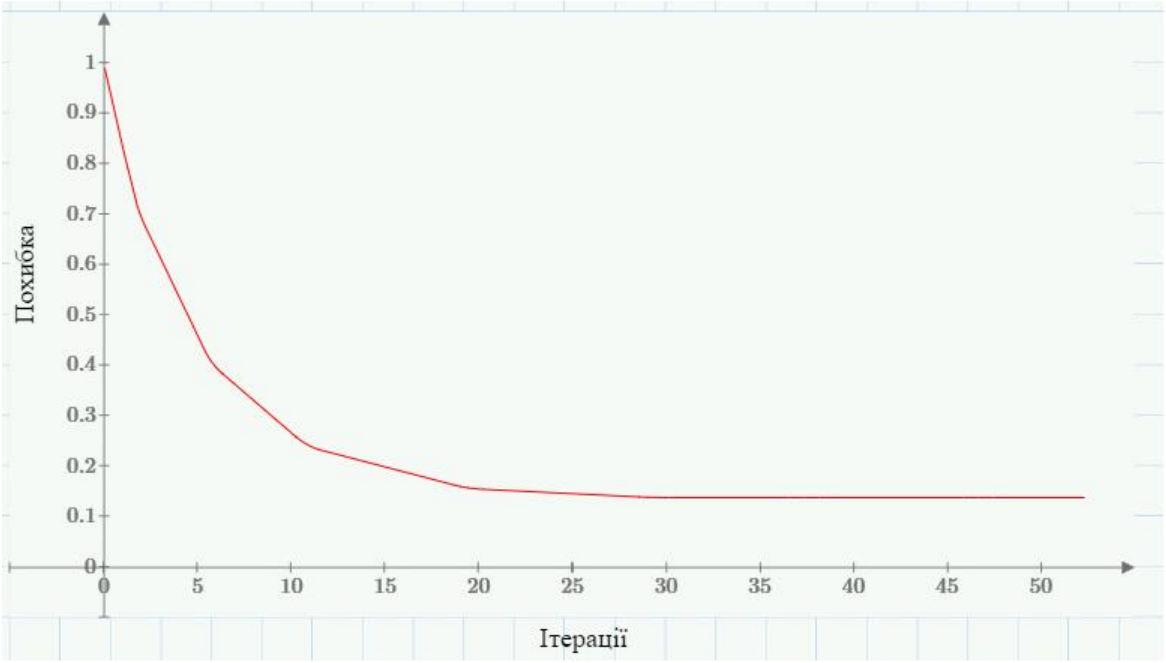


Рисунок 4.3 – Графік спадання помилки нейронної мережі при навчанні

На рис. 4.4 (додаток В) зображено блок-схему логіки роботи розробленого програмного додатку.

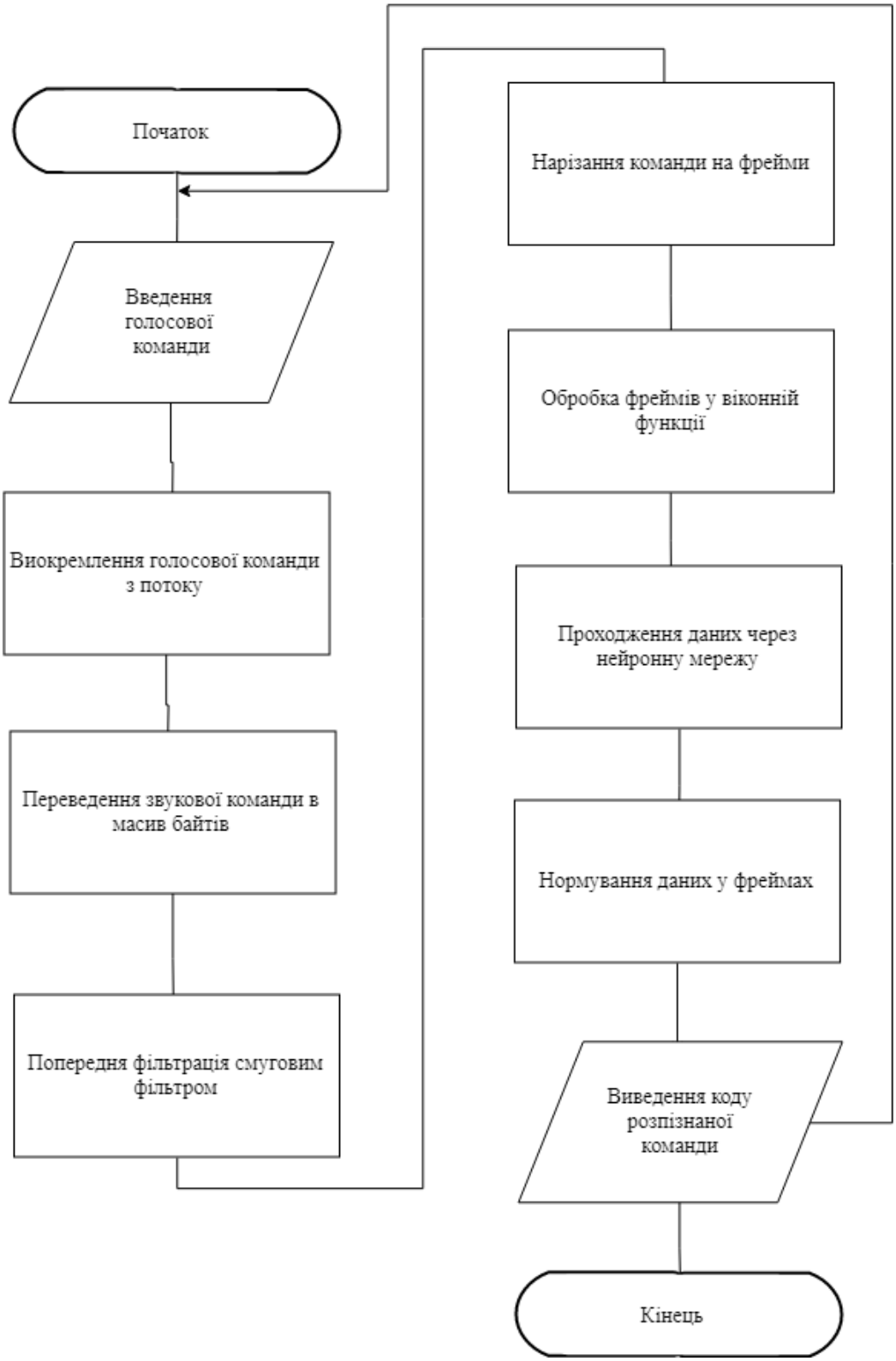


Рисунок 4.4 – Блок-схема програмного забезпечення для розпізнавання голосових команд

4.2. Архітектура програмного забезпечення

Для реалізації цього завдання були написані:

- Одна бібліотека класів генетичного алгоритму, що містить два класи;
- Один клас інструментів для обробки звуку;
- Проекти з тестування бібліотеки генетичних алгоритмів;
- Один клас графічного інтерфейсу Windows Form;
- Для кожного написаного класу був написаний один тестовий клас, який перевіряє кожен відповідний метод відповідного класу;
- Один скрипт Python з нейронною мережею.

Взаємозв'язок між класами розробленого програмного забезпечення можна побачити на рис. 4.5 (додаток Г).

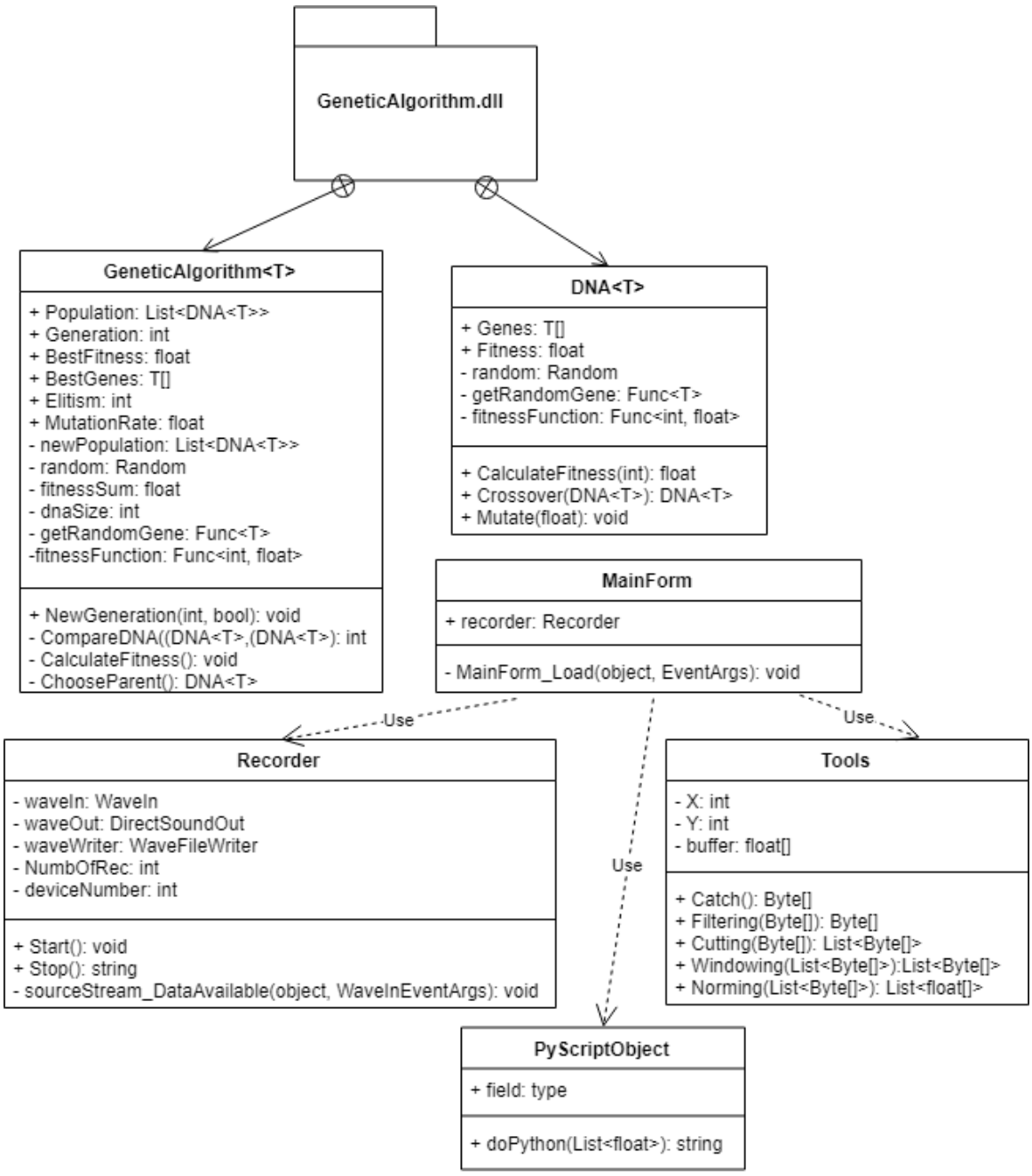


Рисунок 4.5 – UML діаграма класів системи розпізнавання голосових команд

Висновок до розділу

Таким чином, розробивши програмний додаток, що відповідає вимогам завдання, і протестувавши його, досліджено роботу нейронної мережі, навченої генетичним алгоритмом, і значення точності мережі збільшилося до максимально можливого з цією вибіркою навчальних даних, а саме до ~ 77%. Це досить хороший показник, враховуючи невеликий обсяг даних для

навчання. Систему, навіть з такою точністю, можна застосовувати в багатьох сферах використання, тому що в будь-якому випадку це всього лише базис, на який можна додавати різні функції.

5. ПРАКТИЧНЕ ЗАСТОСУВАННЯ

5.1. Керівництво користувача

Для використання програмного продукту, розробленого в даній кваліфікаційній роботі, користувач повинен володіти наступним:

- Розуміння роботи електронних пристроїв;
- Навички використання персонального комп'ютера для налаштування додаткових параметрів;
- Уміння говорити на мові, локалізація команд якої обрано;
- Моральна підготовка до вступу в епоху спілкування людей і машин.

Програмне забезпечення передбачає розпізнавання наступних десяти команд:

- «Вгору»;
- «Вниз»;
- «Відчинити»;
- «Зачинити»;
- «Увімкнути»;
- «Вимкнути»;
- «Підняти»;
- «Опустити»;
- «Збільшити»;
- «Зменшити»;

Але за бажанням власника софта ці команди можна перепрограмувати на будь-яку іншу кількість і будь-яке інше значення, але вони мають бути не

словосполученнями, а складатися з одного слова. І, звичайно ж, вам потрібно буде заново навчити мережу розпізнавати нові команди.

Мінімальні системні вимоги до ПЗ:

- Процесор 500 МГц
- ОЗУ 20 Мб
- Дисковий простір 1.2 Мб

5.2. Випробування програмного продукту

Всі написані елементи програмного продукту пройшли модульне тестування, а саме програмне забезпечення було перевірено декількома добровольцями, які вирішили взяти участь в дослідницькій роботі.

Практична апробація розробленого рішення задачі проходила на контролері роботизованої системи розумного будинку Arduino, на який комп'ютер передає код розпізнаної команди. Після отримання коду контролер відтворює задану логіку відповідно до отриманого коду команди. Приблизна схема цього процесу представлена на рис. 5.1 (додаток Д).

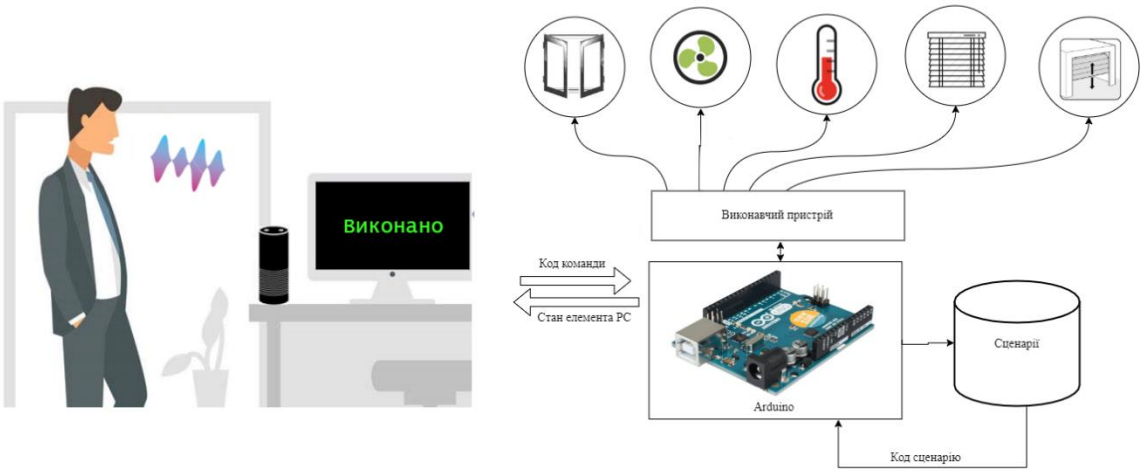


Рисунок 5.1 – Ілюстрація передачі коду команди контролеру роботизованої системи розумного будинку

5.3. Приклади застосування

Технологія розпізнавання голосу існує вже 50 років. Вчені вирішували цю проблему протягом півстоліття, і лише в останні десятиліття IT-компанії

знайшли її рішення. Результатом минулого року роботи став новий рівень точності і широкого використання голосових технологій в повсякденному і професійному житті. Ми користуємося пошуковими системами кожен день. Шукаємо, де пообідати, як дістатися до потрібного місця, або намагаємося знайти значення невідомого терміна. Технологія розпізнавання мови, яку використовують, наприклад, Google або Яндекс, дозволяє нам витрачати мінімум часу на пошук. Все просто і зручно.

У професійному середовищі технології допомагають спростити роботу в кілька разів. Наприклад, в медицині мова лікаря перетворюється в текст історії хвороби і рецепт відразу для прийому. Це економить час на введення інформації про пацієнта в документи. Бортовий комп'ютер автомобіля реагує на запити водія, наприклад, допомагає знайти найближчу заправку. Для людей з обмеженими можливостями життєво важливо реальне впровадження систем в програмне забезпечення побутової техніки, щоб ними можна було керувати за допомогою голосу.

Згідно з дослідженням Google, 55% американських підлітків і 41% дорослих використовували голосовий пошук хоча б раз в день. Більш того, голосовий пошук потрібний при приготуванні їжі, перегляді телевізора і в інших подібних ситуаціях, коли складно набирати текст звичайним способом або це вимагає додаткових зусиль.

Дослідження було проведено на замовлення Google міжнародною консалтинговою фірмою Northstar Research. Вона вивчила звички голосового пошуку на смартфонах у 1400 американців старше 13 років (400 підлітків у віці 13-17 років і 1000 дорослих старше 18 років).

На початку 2017 року маркетингове агентство Higher Visibility опублікувало статистику використання голосового пошуку. За його словами, більше половини опитаних користуються пошуком за кермом. Це говорить про те, що голосовий пошук в першу чергу вплине на місцеві запити, тобто на місцеві підприємства.

Наприклад, користувач запитує: «Окей, Google, де я можу з'їсти піцу?». У відповідь голос асистента розповість про найближчі установи громадського харчування. Очевидно, що якщо ви є власником бізнесу і не перебуваєте в Google My Business, вас не буде на карті, і система не буде рекомендувати вас.

У квітні 2018 року BrightLocal провела аналітичні дії, стосовно найпоширенішого способу пошуку кафе та ресторанів, продуктових магазинів, служб доставки їжі, магазинів одягу та готелів користувачами. За результатами: 58% споживачів використовували голосовий пошук для пошуку інформації, прив'язаної до певного місця. За останні 12 місяців 46% користувачів голосового пошуку щодня шукають місцеві компанії. Найчастіше запитують адресу, телефон, контакти, способи дістатися і час роботи.

5.3.1. Область практичного використання

Найбільш ранні версії додатків мовної технології включали інтерактивну голосову відповідь (IVR). IVR часто використовується в колл-центрах для збору інформації про клієнтів і маршрутизації дзвінків до відповідного агенту. Через попередньо записану поведінку IVR технологія обмежена в тому, наскільки вона може автоматизувати завдання. Мовна технологія може поліпшити IVR за допомогою голосових команд для сортування попередньо записаних відповідей при управлінні викликами.

Чат-боти – ще один спосіб допомогти в зборі даних і відправити дзвінок або запит відповідному одержувачу. Чат-боти пропонують додаткові переваги, такі як можливість пошуку і вилучення інформації за ключовими словами за допомогою голосових команд, що спрощує агентам перегляд великих обсягів інформації.

Поза колл-центрами персональні цифрові помічники – популярне використання технологій мовлення. Більшість з них працюють з простим форматом команд-ресурсів, що дасть голосовим командам виконувати такі

завдання, як оновлення календарів, ініціювання конференц-зв'язку і диктування електронної пошти.

Застосування програмного забезпечення, розробленого в даній кваліфікаційній роботі, можливо не тільки в сфері розумних будинків, а й у багатьох інших.

Наприклад:

- полегшити використання різних інтерфейсів для людей з обмеженими можливостями;
- збільшення швидкості переналагодження додаткових параметрів автомобіля (дзеркала, температури, сидіння, круїз-контроль, клімат-контроль, навігаційна система, радіостанції);
- аналіз розмов на предмет рекламних рекомендацій;
- контроль роботів, що вільно рухаються.

5.3.2. Демонстрація застосування

Схема застосування програмного забезпечення, розробленого в даній кваліфікаційній роботі, представлена на рис. 5.2

Рисунок 5.2 – Схема застосування програми розпізнавання голосових команд

Також далі на рис. 5.3 (додаток Д) наведено приклад користувацького інтерфейсу, що демонструє розпізнавання команд.

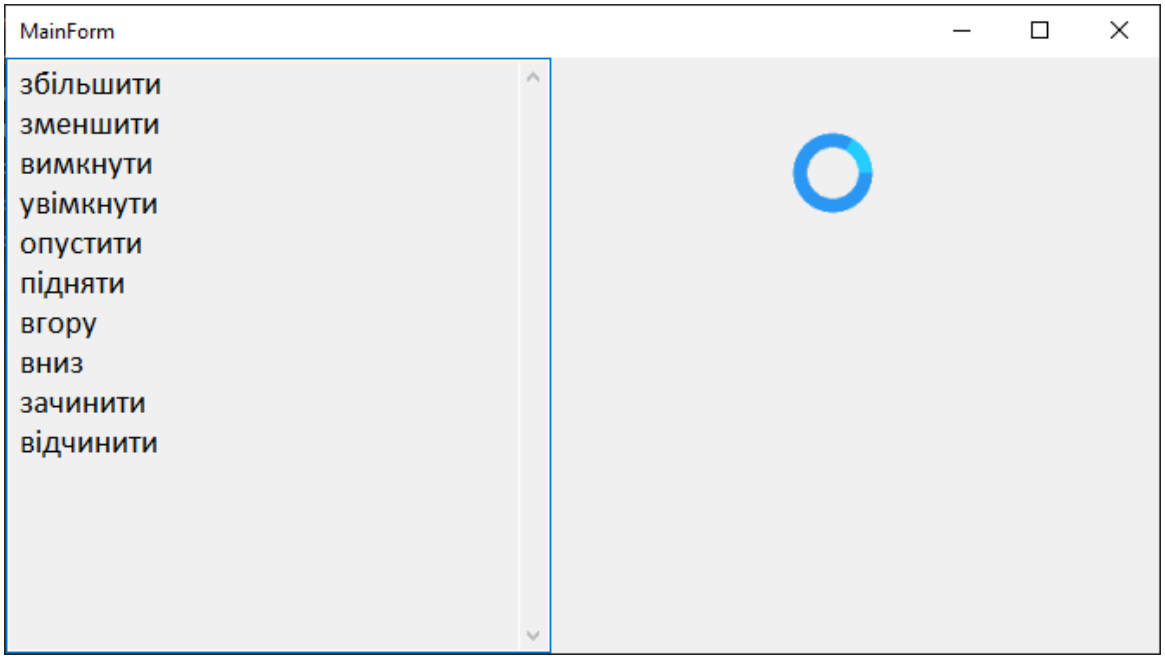


Рисунок 5.3 – Демонстрація розпізнаних команд у зразковому інтерфейсі

Висновок до розділу

Отже, було описано практичне застосування і можливі області використання цього програмного забезпечення. Описано мінімальні навички, які користувач повинен мати для успішного використання програми, і мінімальні системні вимоги для успішної роботи програмного додатку. Наведено приклад інтерфейсу користувача. Даний розділ є логічним завершенням цієї кваліфікаційної роботи, що дозволяє перейти до загального висновку.

ВИСНОВОК

В цій кваліфікаційній роботі були вивчені, досліджені, зіставлені і проаналізовані кілька методів навчання нейронних мереж розпізнаванню голосових команд. Виявлено переваги та недоліки розглянутих методів. На основі побудованої моделі нейронної мережі і обраного методу навчання було написано програмне забезпечення, що повністю відповідає поставленому завданню. Було розроблено механізм попередньої обробки вхідних голосових команд, а саме: попередня фільтрація команди, розділення команди на фрейми, обробка фреймів у віконній функції, нормалізація даних у фреймах. Було навчено нейронну мережу за допомогою генетичного алгоритму, на даних, які являють собою певну кількість записаних однакових голосових команд, вимовлених з різною інтонацією та гучністю, та попередньо оброблених для підвищення точності навчання нейронної мережі. Було розроблено бібліотеку класів генетичного алгоритму, яка надає API для користування її в якості алгоритму навчання нейронної мережі. В даній роботі помітна швидкість і точність розпізнавання команд. Генетичні алгоритми виправдали очікування швидкості навчання нейронної мережі та стійкість до застрягання функції помилки у локальному мінімумі на відміну від алгоритму зворотного поширення помилок. Крім того, генетичні алгоритми набагато легше зрозуміти і написати, ніж зворотне поширення, але у них також є свої недоліки. Одна з них – це не проста операція масштабування створеної моделі, тому що якщо вам необхідно додати більше слів в одну команду, функцію придатності (адаптації) доведеться змінити, а також перевчити мережу.

Результатом дослідження є те, що генетичні алгоритми можуть бути серед кандидатів на найефективніший алгоритм навчання нейронної мережі для розпізнавання людських голосових команд з метою управління роботизованою системою.

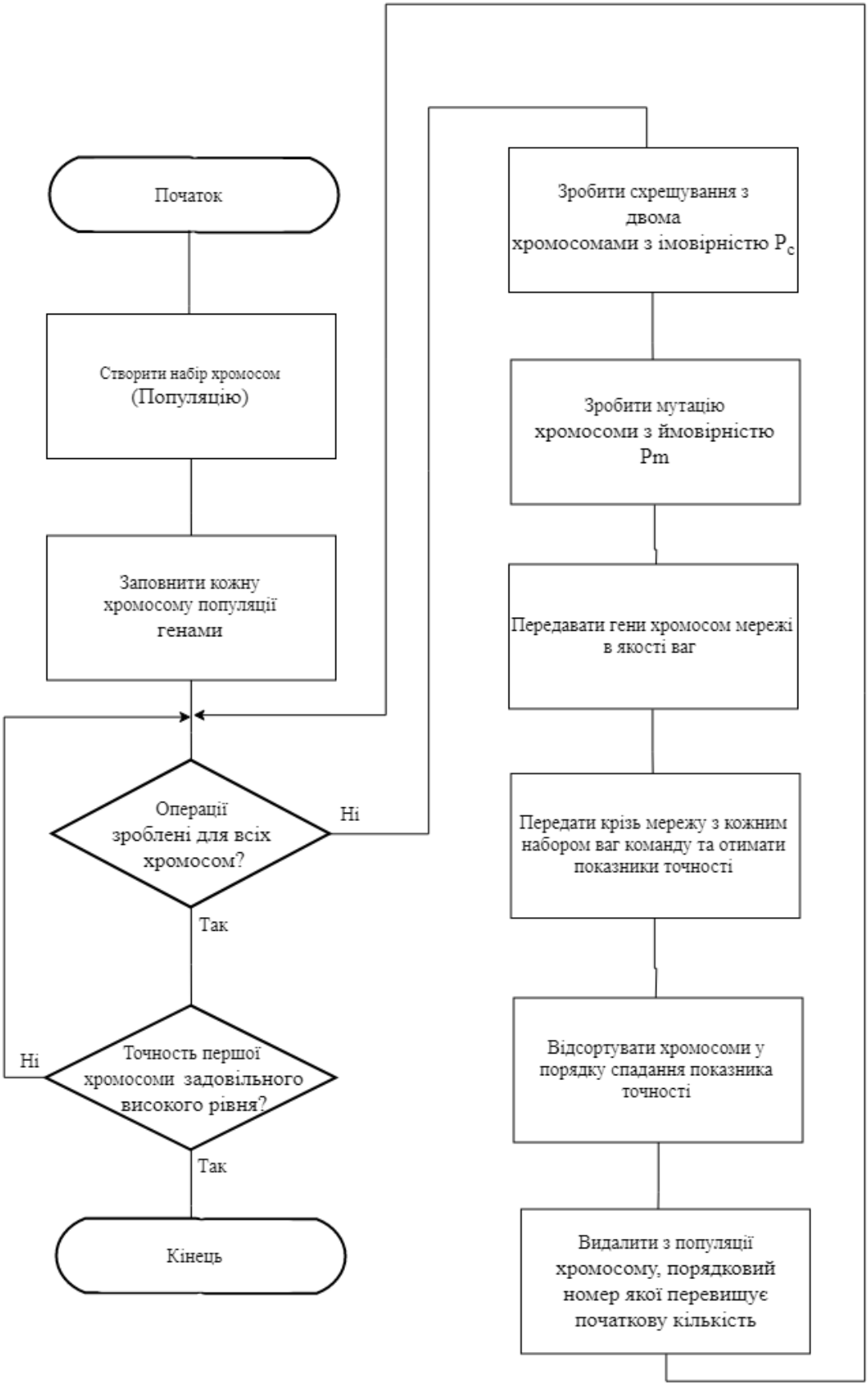
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Комп'ютерне розпізнавання і породження мовлення. [Електронний ресурс]. − Режим доступу: http://speech-text.narod.ru/chap3.html
2. Корицький, Д.В. Система розпізнавання мовних команд. [Електронний ресурс]. − Режим доступу: http://www.nsc.ru/ws/show_abstract.dhtml?ru+130+9365
3. Віконне перетворення Фур'є [Електронний ресурс]. − Режим доступу: http://ru.wikipedia.org/wiki/Оконное_преобразование_Фурье
4. Фролов, А.В. Синтез і розпізнавання мови. Сучасні рішення. / А.В. Фролов, Г.В. Фролов. − 186 с.
5. Маркова, В.А. Мережі Джордана і Елмана. [Електронний ресурс]. − Режим доступу: http://i-intellect.ru/articles-of-neural-networks/jordans-and-elmans-networks.html
6. Стариков, А. Генетичні алгоритми − математичний апарат. [Електронний ресурс]. − Режим доступу: http://www.basegroup.ru/library/optimization/ga_math/
7. Петро Радько, Глава 3. Основи ИНС [Електронний ресурс]. − Режим доступу: https://neuralnet.info/chapter/основы-инс/
8. Ле Н. В., Панченко Д. Попередня обробка мовних сигналів для системи розпізнавання мови // Молодий вчений. — 2011. — №5. Т.1. — С. 74−76. — [Електронний ресурс]. − Режим доступу: https://moluch.ru/archive/28/3171/ (дата звернення: 26.05.2019).
9. Шемановський. ReLU — [Електронний ресурс]. − Режим доступу: https://medium.com/@congyuzhou/relu-b1a6ba5455b (дата звернення: 26.05.2019).
10. Румбелхарт, Д.І. Вивчення внутрішніх уявлень шляхом поширення помилки / Румбелхарт, Д.І., Г.І. Хінтон, Р.Д. Уільямс // Паралельно розподілена обробка.− 1986.− V.1.− с. 318–362.

11. Монтана, Д.Д. Навчальні нейронні мережі з використанням генетичних алгоритмів / Д.Д. Монтана, Л. Девіс // Матеріали одинадцятої міжнародної спільної конференції з штучного інтелекту, Детройт, MI.− 1989.− с. 762–767.
12. Пруденсьо, Р.Б.К. Еволюційний дизайн нейрона Мережі: застосування до прогнозування річкового потоку / Р.Б.К. Пруденсьо, Т.Б. Лудемір // Праці Міжнародна конференція з штучної інтелігенції та застосування. − с. 56–66.
13. Пруденсьо, Р.Б.К. Дизайн нейронних мереж для Прогнозування часових рядів з використанням ініціалізації випадків Генетичні алгоритми / Р.Б.К. Пруденсьо, Т.Б. Людемір // Праці 8-го Інтернаціоналу Конференція з обробки нейронної інформації, ICONIP.− 2001.− с. 990–995.
14. А.М. Липанов, А.В. Тюриков, А.С. Суворов, Е.Ю. Шелковников, П.В. Гуляев. Застосування генетичного алгоритму для навчання нейронної мережі. − // 2001. – с. 221.
15. Елман, Дж. Л. Знахідка структури в часі. // Когнітивна наука. − 1990. − С. 179—211.

ДОДАТОК А

Блок-схема навчання нейромережі



ДОДАТОК Б

Програмний код бібліотеки генетичного алгоритму

```
    public class GeneticAlgorithm<T>
    {
       public class DNA<T>
    {
        public T[] Genes { get; private set; }
        public float Fitness { get; private set; }

        private Random random;
        private Func<T> getRandomGene;
        private Func<int, float> fitnessFunction;

        public DNA(int size, Random random, Func<T> getRandomGene,
            Func<int, float> fitnessFunction, bool shouldInitGenes = true)
        {
            Genes = new T[size];
            this.random = random;
            this.getRandomGene = getRandomGene;
            this.fitnessFunction = fitnessFunction;

            if (shouldInitGenes)
            {
                for (int i = 0; i < Genes.Length; i++)
                {
                    Genes[i] = getRandomGene();
                }
            }
        }

        public float CalculateFitness(int index)
        {
            Fitness = fitnessFunction(index);
            return Fitness;
        }

        public DNA<T> Crossover(DNA<T> otherParent)
        {
            DNA<T> child = new DNA<T>(Genes.Length, random, getRandomGene,
fitnessFunction, shouldInitGenes: false);

            for (int i = 0; i < Genes.Length; i++)
            {
                child.Genes[i] = random.NextDouble() < 0.5 ? Genes[i] :
otherParent.Genes[i];
            }

            return child;
        }

        public void Mutate(float mutationRate)
        {
            for (int i = 0; i < Genes.Length; i++)
            {
                if (random.NextDouble() < mutationRate)
                {
                    Genes[i] = getRandomGene();
                }
            }
```

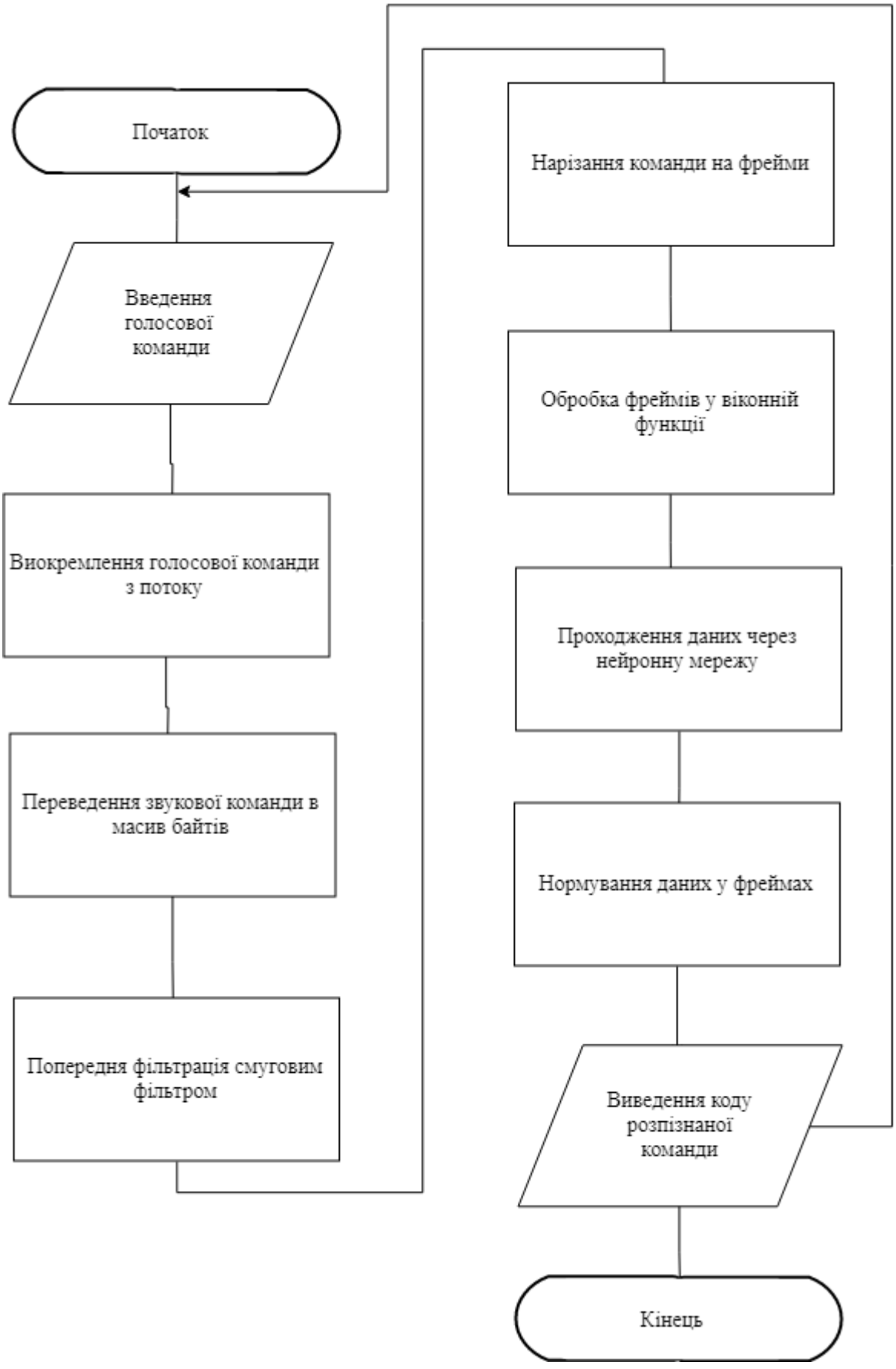
```
        }
    }
        public List<DNA<T>> Population { get; private set; }
        public int Generation { get; private set; }
        public float BestFitness { get; private set; }
        public T[] BestGenes { get; private set; }

        public int Elitism;
        public float MutationRate;

        private List<DNA<T>> newPopulation;
        private Random random;
        private float fitnessSum;
        private int dnaSize;
        private Func<T> getRandomGene;
        private Func<int, float> fitnessFunction;

        public GeneticAlgorithm(int populationSize, int dnaSize, Random random,
            Func<T> getRandomGene, Func<int, float> fitnessFunction,
            int elitism, float mutationRate = 0.01f)
        {
            Generation = 1;
            Elitism = elitism;
            MutationRate = mutationRate;
            Population = new List<DNA<T>>(populationSize);
            newPopulation = new List<DNA<T>>(populationSize);
            this.random = random;
            this.dnaSize = dnaSize;
            this.getRandomGene = getRandomGene;
            this.fitnessFunction = fitnessFunction;

            BestGenes = new T[dnaSize];

            for (int i = 0; i < populationSize; i++)
            {
                Population.Add(new DNA<T>(dnaSize, random, getRandomGene,
fitnessFunction, shouldInitGenes: true));
            }
        }

        public void NewGeneration(int numNewDNA = 0, bool crossoverNewDNA = false)
        {
            int finalCount = Population.Count + numNewDNA;

            if (finalCount <= 0)
            {
                return;
            }

            if (Population.Count > 0)
            {
                CalculateFitness();
                Population.Sort(CompareDNA);
            }
            newPopulation.Clear();

            for (int i = 0; i < Population.Count; i++)
            {
                if (i < Elitism && i < Population.Count)
                {
                    newPopulation.Add(Population[i]);
                }
```

```
                else if (i < Population.Count || crossoverNewDNA)
                {
                    DNA<T> parent1 = ChooseParent();
                    DNA<T> parent2 = ChooseParent();

                    DNA<T> child = parent1.Crossover(parent2);

                    child.Mutate(MutationRate);

                    newPopulation.Add(child);
                }
                else
                {
                    newPopulation.Add(new DNA<T>(dnaSize, random, getRandomGene,
fitnessFunction, shouldInitGenes: true));
                }
            }

            List<DNA<T>> tmpList = Population;
            Population = newPopulation;
            newPopulation = tmpList;

            Generation++;
        }

        private int CompareDNA(DNA<T> a, DNA<T> b)
        {
            if (a.Fitness > b.Fitness)
            {
                return -1;
            }
            else if (a.Fitness < b.Fitness)
            {
                return 1;
            }
            else
            {
                return 0;
            }
        }

        private void CalculateFitness()
        {
            fitnessSum = 0;
            DNA<T> best = Population[0];

            for (int i = 0; i < Population.Count; i++)
            {
                fitnessSum += Population[i].CalculateFitness(i);

                if (Population[i].Fitness > best.Fitness)
                {
                    best = Population[i];
                }
            }

            BestFitness = best.Fitness;
            best.Genes.CopyTo(BestGenes, 0);
        }

        private DNA<T> ChooseParent()
        {
```

```
            double randomNumber = random.NextDouble() * fitnessSum;

            for (int i = 0; i < Population.Count; i++)
            {
                if (randomNumber < Population[i].Fitness)
                {
                    return Population[i];
                }

                randomNumber -= Population[i].Fitness;
            }

            return Population[random.Next(1,Population.Count-1)];
        }
    }
}
```

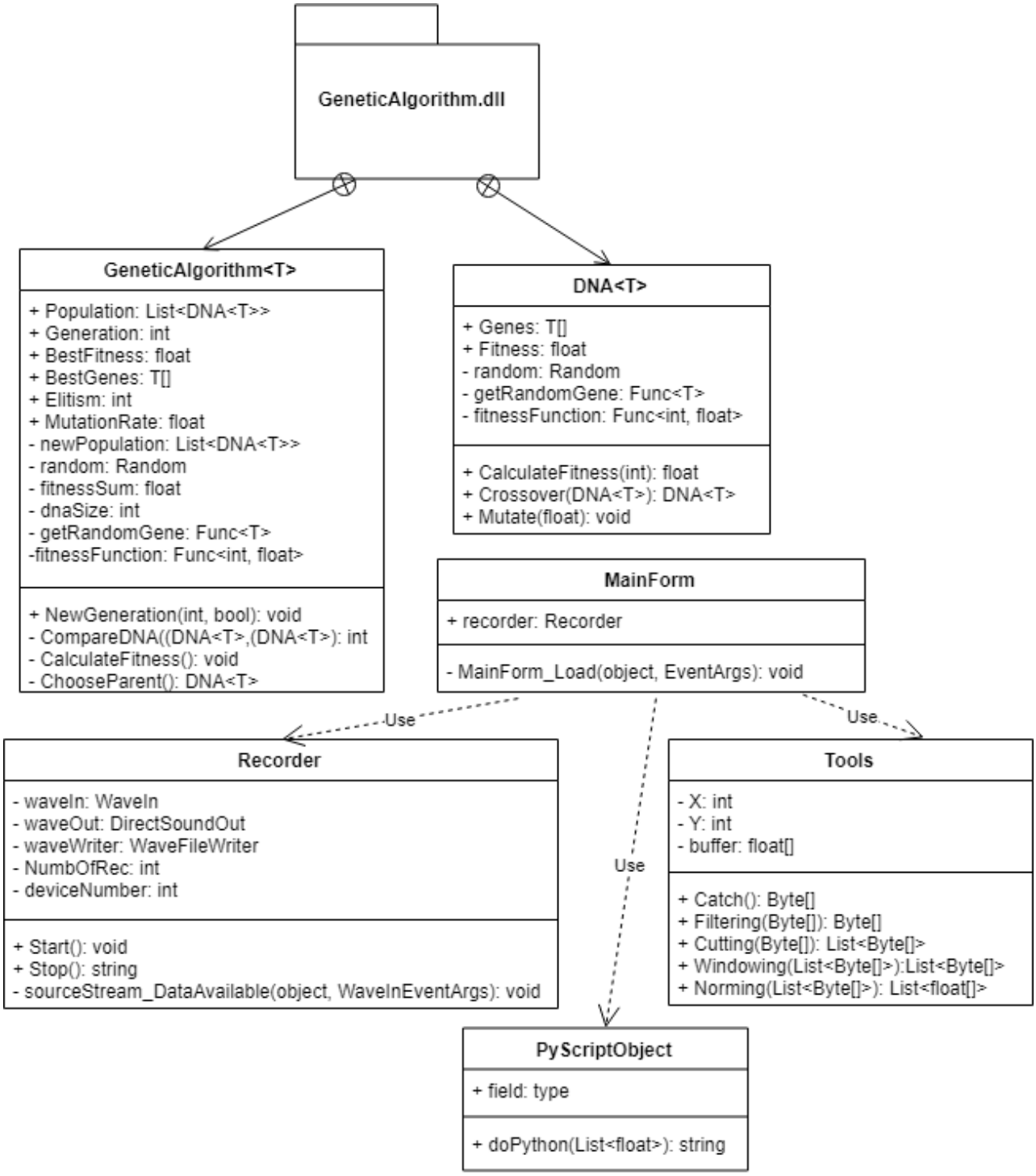
ДОДАТОК В

Блок-схема роботи програмного додатку



ДОДАТОК Г

UML Діаграма класів

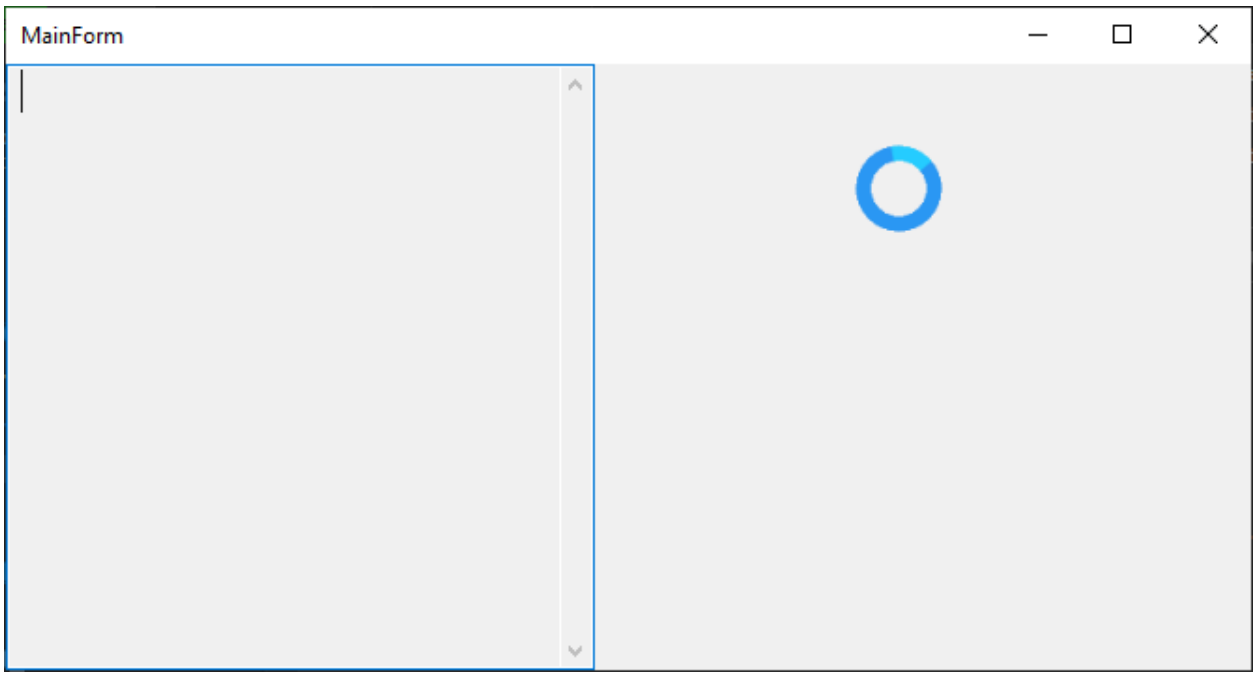


ДОДАТОК Д

Демонстрація роботи додатку

Ілюстрація управління роботизованою системою розумного дому за допомогою голосових команд

Ілюстрація передачі коду команди на контролер роботизованої системи розумного дому



Початковий вигляд вікна користувацького інтерфейсу розробленого додатку.

Вигляд вікна користувацького інтерфейсу розробленого додатку після розпізнавання команд.