Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Другий (магістерський)
(рівень вищої освіти)

Розроблення системи ідентифікацій команд оператора для керування мобільним роботом на базі нейронних мереж
(тема)

Виконала:
студентка 2 курсу, групи КІТПВм-21-1

Пилипенко В. М.
(прізвище, ініціали)

Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)

Тип програми Освітньо-професійна

Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва
(повна назва освітньої програми)

Керівник проф. В.В. Євсєєв
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАМ _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2022р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«26» грудня 2022 р. *Пилипенко В. М.*

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
Рівень вищої освіти другий (магістерський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми Освітньо-професійна
Освітня програма Комп'ютерно-інтегровані технологічні процеси виробництва
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАМ _____
(підпис)
«_____» _____2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Пилипенко Владиславі Михайлівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи ідентифікацій команд оператора для керування мобільним роботом на базі нейронних мереж
Затверджена наказом по університету від 07.11.2022 р. № 1464 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 26.12.2022 р.
3. Вихідні дані до роботи
3.1 Рекурента нейрона мережа
3.2 Топологія нейронної мережі
3.3 Команди для ідентифікації
3.4 Бібліотека Tensorflow
3.5 Бібліотека Keras
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Дослідження сучасних методів керування мобільними роботами
4.3 Розробка топології нейронної мережі для голосового керування роботом
4.4 Розроблення системи ідентифікації команд оператора для керування мобільним роботом на базі нейронних мереж
4.5 Експериментальні дослідження з розробленим програмним додатком
4.6 Висновки
4.7 Додатки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій _____
Демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) – стор. 17 Ф. А4

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Дослідження сучасних методів		
	керування мобільними роботами	01.09.22-02.10.22	виконано
2	Розробка топології нейронної мережі		
	для голосового керування роботом	03.10.22-01.11.22	виконано
3	Розроблення системи ідентифікації		
	команд оператора для керування		
	мобільним роботом на базі нейронних		
	мереж	02.11.22-03.12.22	виконано
4	Експериментальні дослідження з		
	розробленим програмним додатком	04.12.22-19.12.22	виконано
5	Висновки	20.12.22	виконано
6	Оформлення пояснювальної записки	21.12.22	виконано
7	Подання роботи на перевірку		
	Інтернет-сервісом Unicheck	22.12.21	виконано
8	Подання роботи на рецензію	23.12.21	виконано
9	Подання роботи на підпис		
	зав. кафедри	25.12.21	виконано
10	Подання кваліфікаційної роботи в ЕК	26.12.21	виконано

Дата видачі завдання 01.09.2022 р.

Студент _____ Пилипенко В.М.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. В.В. Євсєєв
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 87 с., 1 табл., 51 рис., 2 дод., 23 джерел.

ГОЛОСОВЕ КЕРУВАННЯ, ІДЕНТИФІКАЦІЯ КОМАНД, НЕЙРОННІ МЕРЕЖІ, МОБІЛЬНИЙ РОБОТ, ПРОГРАМНИЙ ЗАСІБ, PYTHON.

Мета кваліфікаційної роботи – розробка системи ідентифікацій команд для керування мобільним роботом на базі нейронних мереж.

Об’єктом дослідження є процес керування мобільним роботом.

Предметом дослідження є моделі та методи голосового керування.

Методи дослідження – метод ідентифікації голосових команд, метод математичного розрахунку нейронної мережі.

У кваліфікаційній роботі досліджено сучасні методи керування мобільним роботом. Наведено аналіз методів ідентифікації голосових команд. Досліджено топології нейронних мереж для застосування в системах управління.

На основі проведених досліджень було обрано тип нейронної мережі, а саме рекурентний, розроблено топологію та виконано математичний розрахунок елементів нейронної мережі для голосового керування роботом.

Для досягнення поставленої мети було розроблено програмний продукт для керування мобільним роботом на базі нейронних мереж та проведено експериментальні дослідження розробленого програмного засобу.

ABSTRACT

Explanatory note: 86 pages, 1 table, 51 pictures, 2 applications, 23 sources.

VOICE CONTROL, COMMAND IDENTIFICATION, NEURAL NETWORKS, MOBILE ROBOT, SOFTWARE, PYTHON.

The purpose of this work is to develop of a command identification system for controlling a mobile robot based on neural networks.

The object of research is voice control of a mobile robot.

The subject of research are models and methods of voice control.

Research methods – voice command identification method, neural network mathematical calculation method.

Modern methods of controlling a mobile robot are investigated in the qualification work. The analysis of voice command identification methods is considered. Topologies of neural networks for use in control systems has been investigated.

On the basis of the conducted research, the type of neural network was chosen, namely the recurrent one, the topology was developed and the mathematical calculation of the neural network elements was made for voice control of the robot.

To achieve the goal, a software product for controlling a mobile robot based on neural networks was developed and experimental studies of the developed software were conducted.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ГС – градієнтний спуск;

ЕОМ – електронно-обчислювальна машина;

МР – мобільний робот;

ПММ – прихована марковська модель;

РНМ – рекурентні нейронні мережі;

СКП – середню квадратична помилка;

ЧПУ – числове програмне управління;

ШНМ – штучні нейронні мережі;

DTW (Dynamic Time Warping) – динамічна деформація часу.

ВСТУП

Область застосування мобільних роботів (МР) дуже широка. Це пошук та знешкодження небезпечних об'єктів, завдання радіаційної та хімічної розвідки, робота в зоні техногенних і природних катастроф. Такі робото технічні системи знаходять застосування і в цивільній сфері в якості сервісної робототехніки. Проте, в більшості завдань, які виконуються в заздалегідь не визначених умовах і пов'язаних з високою «ціною» помилки при невірних діях, як і раніше передбачається участь людини-оператора в управлінні роботом. Саме тому застосування робототехніки в різних додатках, пов'язаних з вирішенням спеціальних завдань, вимагає максимальної спрощення способів взаємодії людини і робота.

Одним з актуальних способів вирішення цього питання може бути використання системи голосового керування на основі нейронних мереж. Даний спосіб має можливість швидкого навчання, тому навіть при умові виходу зі строю усі інших способів керування (припустимо, що робота укомплектовано одразу декількома способами) управління системою можна буде відновити.

Таким чином метою кваліфікаційної роботи є розробка системи ідентифікацій команд для керування МР на базі нейронних мереж

Об’єктом дослідження є процес керування МР.

Предметом дослідження є моделі та методи голосового керування.

Методами дослідження є ідентифікації голосових команд та математичний розрахунку нейронної мережі.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– дослідити сучасні методи керування МР;

– обрати тип нейронної мережі;

– розробити топологію нейронної мережі;

– виконати математичний розрахунок елементів нейронної мережі;

– розробити алгоритму роботи системи;

– розробити програмний продукт системи;

– провести експериментальні дослідження;

– розглянути питання безпеки життєдіяльності;

– оформити звіт згідно з ДСТУ 3008:2015 [1].

Робота виконана відповідно рекомендаціям [2-3]. Результати дослідження опубліковані у [4].

1 ДОСЛІДЖЕННЯ СУЧАСНИХ МЕТОДІВ КЕРУВАННЯ МОБІЛЬНИМИ РОБОТАМИ

1.1 Аналіз сучасних методів управління МР

Управління роботами здійснює його прилад управління. В сукупності з сенсорною і виконавськими системами воно утворює систему автоматичного управління робота. Крім того, через прилад управління роботом може управляти людина-оператор.

В роботах застосовуються три способи управління:

– програмне;

– адаптивне;

– інтелектуальне.

Практично тільки програмне управління знайшло застосування в чистому вигляді, да і то часто і до нього додають елементи адаптації. В цілому ж усі ці три способи управління застосовуються комплексно. Адаптивне управління зазвичай будується на базі програмного як наступний рівень управління. Інтелектуальне управління у свою чергу реалізується як надбудова над першими двома рівнями. Назви систем управління конкретних роботів зазвичай визначається основним використаним в ній способом управління.

По мірі участі людини-оператора в процесі управління розрізняють:

– системи автоматичного автоматизованого управління;

– ручного управління.

За типом руху виконавських систем існують системи управління:

– безперервні дискретні позиційні;

– дискретні циклові.

По керованих змінних розрізняють системи управління положенням швидкістю силою.

Часто ці способи управління застосовують в комбінації або різні способи по різних координатах, або з послідовним переходом від одного до іншого, або у вигляді функціональної залежності керованої змінної від іншої [4].

1.1.1 Системи програмного управління

1.1.1.1 Системи дискретного циклового управління

Даний тип управління мають практично усі пневматичні роботи. Процес управління окремими приводами зводиться до одноразового розгону, руху з постійною швидкістю і гальмування досягши упору. Програмування робота полягає в установці на кожному приводі цих упорів, які визначають величину переміщення по відповідній мірі рухливості, швидкості цих переміщень, послідовності включень приводів і можливих затримок часу між цими включеннями. Усі ці операції окрім установки упорів проводяться за допомогою перемикачів або інших органів на пульті облаштування управління. Внаслідок простоти циклового управління для роботів з таким управлінням, як правило, застосовують облаштування групового управління.

1.1.1.2 Системи дискретного позиційного управління.

Типові роботи з таким управлінням – це роботи для точкового зварювання, складання і обслуговування різного технологічного устаткування.

Ці роботи мають велике число точок позиціонування робочого органу маніпулятора. На відміну від систем циклового управління тут точність позиціонування забезпечується не упорами, а точністю відробітку приводами із зворотним зв'язком по положенню заданих програмою, що управляє, точок позиціонування.

Системи програмного управління роботів спочатку були запозичені з систем числовим програмним управлінням (ЧПУ) технологічного устаткування, але вони істотно складніше за останніх, передусім із-за більшого числа ступенів рухливості і їх взаємозв'язаної. Процес дискретного

позиційного програмного управління маніпулятором виглядає таким чином. У облаштуванні управління зазвичай на магнітному носії зберігається програма, що управляє, яка складається із занесених на окремі паралельні доріжки програм для окремих приводів. Ці програми є послідовністю чисельних значень кроків позиціонування приводу цієї ступені рухливості. Відробіток програми, що управляє, полягає в одночасному поданні на усі приводи значень чергового кроку і відробітку приводами цього завдання. Після того, як усі приводи зупиняться, робочий орган маніпулятора займе відповідну чергову позицію в просторі і орієнтацію. Після цього програма, що керує, видасть команду на виконання приводами наступного кроку і так далі. В результаті робочий орган маніпулятора переміщатиметься кроками по запланованій дискретній траєкторії, зупиняючись після кожного кроку.

Програмування, тобто синтез програми, що керує здійснюється методом навчання на самому роботові або аналітично на електронно-обчислювальній машині (ЕОМ). Перший спосіб програмування, так само свого часу запозичений у систем ЧПУ технологічного устаткування, стосовно маніпуляторів має два варіанти.

У першому варіанті оператор в режимі ручного управління окремими приводами послідовно встановлює робочий орган маніпулятора в заздалегідь вибрані точки заданої програмної траєкторії. При цьому в кожній такій точці в пам'ять облаштування управління заносяться значення сигналів з датчиків положення усіх приводів. В результаті проходження таким чином усієї траєкторії в облаштуванні управління опиняється записаною управляє програма, що відповідає їй. Після пробного її відтворення і при необхідності коригування в окремих точках програма готова до роботи. Управління маніпулятором здійснюється при цьому з допомоги миші або інших аналогічних засобів шляхом одночасної скоординованої дії на приводи маніпулятора. Останні обчислюються комп'ютером відповідно до завдання від оператора. Гідність цього варіанту програмування в істотному прискоренні цього процесу.

Другий варіант програмування методом навчання полягає в переміщенні робочого органу маніпулятора рукою оператора і запису при цьому показників датчиків положення приводів як в попередньому варіанті. Для виконання такої операції на робочому органі передбачаються спеціальні ручки, а в конструкції самого маніпулятора – можливість від'єднання приводів від його механічної частини, щоб дати можливість операторові безперешкодно її переміщати. Таким чином цей варіант програмування вимагає відповідної зміни конструкції маніпулятора. Цей варіант програмування застосований до усіх маніпуляторів, не вимагаючи від'єднання приводів як в початковому варіанті.

Аналітичний спосіб програмування дозволяє синтезувати програми, що управляють, на ЕОМ, не задіюючи робота. По суті, в цьому випадку замість робота використовується його математична модель, за допомогою якої і здійснюється процес програмування подібно до того, як це робиться на реальному роботові. При цьому для отримання математичної моделі необхідної точності з робота необхідно регулярно знімати відповідні характеристики. Такий спосіб програмування не вимагає відключення робота на час програмування від технологічного процесу, в якому він задіяний. Тому майбутнє за аналітичним способом програмування.

1.1.1.3 Системи безперервного управління

Типові роботи з безперервним управлінням – це роботи для дугового зварювання і різання, для нанесення покриттів. Головна відмінність цих роботів від роботів з розглянутим вище дискретним позиційним управлінням полягає в тому, що рух по програмній траєкторії здійснюється без зупинок. Це вимагає від приводів більшої швидкодії і призводить до принципової відмінності їх програмування. Якщо, наприклад, записати програму, що управляє, для маніпулятора з безперервним управлінням методом навчання, переміщаючи його робочий орган по необхідній програмній траєкторії на невеликій швидкості, а потім відтворити цю програму на істотно більшій швидкості, яку потрібно за технологією, то із-за неминучого динамічного

запізнювання робочий орган на усіх вигинах траєкторії сходитиме з неї. Це динамічна помилка зростатиме зі збільшенням швидкості руху. Тому програми, що управляють, при такому методі програмування навчанням необхідно коригувати і відпрацьовувати на заданій реальній швидкості, з якою програмна траєкторія повинна відтворюватися. Те ж відноситься, зрозуміло, і до аналітичного програмування: тут потрібна динамічна математична модель робота, тоді як при дискретному позиційному управлінні потрібно кінематичну модель.

При програмуванні систем безперервного управління методом навчання окрім запам'ятовування безперервного переміщення приводів знайшов застосування істотно простіший спосіб, коли запам'ятовується тільки ряд дискретних позицій на програмній траєкторії, а ділянки траєкторії між ними формуються при відтворенні програми за допомогою інтерполятора у вигляді стандартних математичних функцій. Вибір точок на програмній траєкторії робиться з урахуванням кривизни траєкторії: чим вона більша, тим менше береться відстань між точками. Гідність такого способу програмування в різкому скороченні необхідного об'єму пам'яті облаштування управління, а недолік – в меншій точності відтворення програмної траєкторії. Апаратно відповідне облаштування управління є облаштуванням дискретного позиційного управління, доповненим інтерполятором, тобто воно придатне для роботів з будь-яким з цих способів управління.

1.1.1.4 Системи управління по силі

Разом з управлінням переміщенням в маніпуляторах часто потрібно управління силою, з якою робочий орган маніпулятора впливає на об'єкти зовнішнього середовища. Для здійснення управління силою робочий орган маніпулятора забезпечується сенсорним облаштуванням виміру вектору сили, яке зазвичай встановлюється безпосередньо перед робочим органом. Програма управління величиною сили зазвичай полягає в підтримці її постійного значення або в зміні у функції від переміщення. Можливий і

зворотний варіант управління переміщенням у функції від сили дії, що розвивається, на середовище. Останні варіанти називаються позиційно-силовим управлінням.

Строго кажучи, управління з використанням інформації про силу повинне відноситися вже до адаптивного управління, оскільки ця інформація відноситься до зовнішнього середовища, хоча в його основі і лежить програмне управління.

1.1.2 Системи адаптивного управління

Розглянуті вище системи програмного управління роботами ґрунтовані на найбільш простому способі автоматичного управління без зворотного зв'язку по фактичному стану зовнішнього середовища, з яким взаємодіє робот. У зв'язку з цим такі системи застосовні тільки при повністю детермінованих і незмінних упродовж усього процесу управління зовнішніх умовах роботи, а також цілях управління і параметрах самого робота.

Адаптивне управління здійснюється у функції від параметрів зовнішнього середовища і тому дозволяє забезпечити досягнення мети управління при непостійності або неповній апріорній інформації про ці параметри. Для здійснення такого управління робот має бути забезпечений сенсорними пристроями.

При адаптивному управлінні, зрозуміло, максимально використовують і заздалегідь складені програми для виконання тих частин завдання, які можуть бути реалізовані цим простим способом.

На рис. 1.1 показана узагальнена структура системи управління сенсорних роботів, до яких відносяться і дані роботи з адаптивним управлінням. Вона включає п'ять рівнів управління Р1-Р5.

Зв'язок людини-оператора з роботом здійснюється через пульт. Оператор видає роботові завдання, контролює їх виконання і проводить загальний контроль за процесом функціонування робота в цілому.

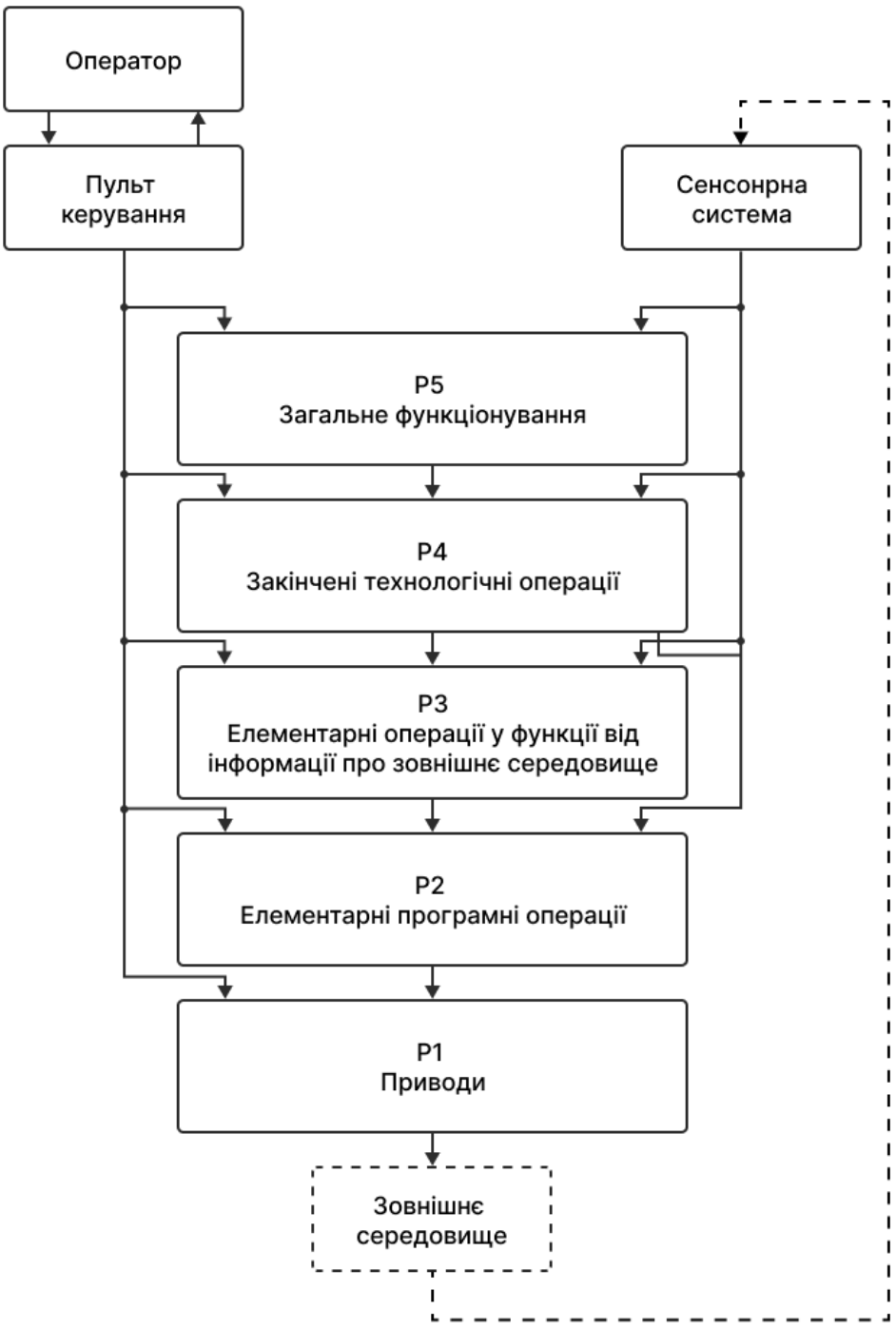


Рисунок 1.1 – Узагальнена структура системи управління сенсорним роботом [5]

П'ятий (верхній) рівень автоматичного управління P5 аналізує завдання, що поступають від людини-оператора, і визначає послідовність дій робота відповідно до завдання, тобто планує дії робота. На цьому рівні аналізується інформація про зовнішнє середовище, що отримується від сенсорної системи, і синтезуються моделі, на базі яких виконується планування дії робота. У

загальному випадку моделі зовнішнього середовища утворюють ієрархічну послідовність від первинної, найбільш конкретної моделі, яка описується за допомогою параметрів середовища, безпосередньо визначуваних сенсорними пристроями, і далі до усе більш абстрактних моделей, що використовують відповідно більше узагальнені поняття для опису зовнішнього середовища. В процесі функціонування робота моделі зовнішнього середовища коригуються і удосконалюються.

П'ятий рівень управління відповідає за функціонування робота як єдиної системи, забезпечуючи реалізацію не лише основних «професійних» функцій робота, але і службових загальносистемних завдань, які визначаються вимогами до умов функціонування робота. Рівень Р5 визначає в цілому інтелектуальні можливості робота і круг вирішуваних ним завдань.

Четвертий рівень управління Р4 – це рівень синтезу функціонально закінчених складних дій, в результаті яких вирішується конкретне завдання. Відповідно до плану, виробленого для цього на вище стоячому рівні Р5, на рівні Р4 робиться його розбиття на послідовність елементарних типових операцій, які реалізуються нижніми рівнями управління. Результатом дії Р4 є видача управлінь на подальші рівні Р3 і Р2. Рівень Р4 використовує також поточну інформацію від сенсорних пристроїв для оперативної корекції планів, що отримуються з рівня Р5.

Третій і другий рівні управління Р3 і Р2 – це рівні виконання елементарних операцій, на які можуть бути розбиті закінчені дії робота. Відмінність між цими рівнями полягає в тому, що на рівні Р3 синтезуються адаптивні управління у функції від інформації про зовнішнє середовище, а на рівні Р2 – простіші управління за програмою. У зв'язку з цим при синтезі управлінь на рівні Р3 використовуються разом з типовими програмами рівня Р2 команди на вхід рівня Р1 паралельно з діями, що управляють, з виходу рівня Р2. Що в результаті поступило на вхід третього рівня завдання реалізується, по-перше, у вигляді послідовності типових програм другого рівня, і по-друге, у вигляді сукупності дій, що управляють, безпосередньо на окремі приводи

рівня Р1. Усі ці дії в цілому задаються і координуються рівнем Р3 залежно від поточної інформації про зовнішнє середовище і стан самого робота. На рівні Р2 розраховуються дії, що управляють, які потім поступають на рівень Р1, що реалізовує програмне управління приводами.

Нижній рівень управління Р1 реалізує управління по окремих мірах рухливості робота і є системою управління приводами.

Схема системи управління роботом, зображена на рис. 2.1, є спрощеною. На ній не показані усі прямі зв'язки виходів окремих рівнів управління з входами нижніх рівнів, окрім найближчого, а також зворотні зв'язки виходів нижніх рівнів з входами верхніх. На схемі не відбиті інформаційні зв'язки окремих рівнів з пультом управління, які забезпечують передачу інформації про функціонування робота людині-операторові.

Людина-оператор принципово може взаємодіяти з роботом на будь-якому рівні ієрархії управління. Людина-оператор може видавати завдання роботові безпосередньо на рівень Р1 шляхом командного управління кожним приводом окремо. Таке управління є дуже трудомістким і вимагає великої навички. Тимчасове запізнювання в каналі зв'язку ще більше ускладнює роботу в цьому режимі. У зв'язку з цим до нього прибігають тільки в тих випадках, коли з яких-небудь причин інші способи управління виявляються неприйнятними.

При управлінні роботом через рівні Р2 і Р3 людина-оператор замінює рівень Р4, задаючи на їх входи найменування що підлягають виконанню програмно (на Р2) або адаптивно (на Р3) елементарних операцій, після чого стежить за їх автоматичним виконанням. Таке управління називається супервізорним. Аналогічним чином людина-оператор може управляти і через рівні Р4 і Р5, задаючи вже не елементарні операції, а складніші закінчені технологічні процеси.

Розвитком супервізорного способу управління є інтерактивне управління, яке включає двосторонній обмін інформацією між людиною і роботом у вигляді діалогу. Робот, отримавши чергове завдання від людини, у

свою чергу просить його про необхідні уточнення або інформує про необхідність відкоригувати завдання, щоб зробити його здійснимим. Цей режим управління, таким чином, максимально спрощує функції і рівень уміння людини-оператора за рахунок відповідного алгоритмічного ускладнення системи управління робота аж до наділу його штучним інтелектом.

Згідно з узагальненою схемою на рис. 2.1 така система повинна включати не менше трьох рівнів управління – Р1, Р2 і Р3. Власне адаптивне управління реалізується рівнем Р3 через рівень програмного управління Р2 або безпосередньо впливаючи на рівень системи приводів Р1. Залежно від міри складності технологічних операцій, що виконуються роботом в адаптивному режимі, тобто з використанням сенсорної інформації, система адаптивного управління може включати і інші верхні рівні управління Р4 і Р5. Проте обов'язковою приналежністю ці рівні є для системи інтелектуального управління.

1.1.3 Системи інтелектуального управління

Інтелектуальне управління – це наступний після адаптивного управління найвищий відносно алгоритмічних можливостей тип управління.

Інтелектуальні системи будуються як біотехнічні системи, включаючи системи інтелектуального управління, поки мають дуже обмежені інтелектуальні можливостями. У них використовують теорію нечітких великих кількостей і нечіткої логіки, різні евристичні алгоритми і технології експертних систем, асоціативної пам'яті і технічних нейронних мереж. Ведуться роботи із створення так званих нечітких комп'ютерів, які оперують нечіткими даними і виведеннями. Це вимагає створення нової нечіткої елементної бази і відповідного програмного забезпечення.

Головна сфера застосування інтелектуального управління – це передусім складні і великі об'єкти і системи, для яких доступний опис тільки на семіотичному рівні. До них передусім відносяться біотехнічні системи що

включають людину. Оскільки такі системи мають природний інтелект, управління ними може бути так само тільки інтелектуальним. В той же час інтелектуальне управління може знадобиться і для досить простих об'єктів, якщо з їх допомогою вирішуються інтелектуальні завдання або якщо саме завдання управління ними вимагає інтелектуального підходу в силу, наприклад, складнощі зовнішніх умов. У робототехніці штучний інтелект може знадобитися передусім для вирішення наступних завдань [6]:

– обробка сенсорної інформації(фільтрація, стискування інформації, розпізнавання образів);

– створення моделей зовнішнього середовища; планування поведінки;

– управління рухом;

– створення інтелектуального інтерфейсу між людиною-оператором і роботом.

Підвищення рівня штучного інтелекту пов'язане передусім з розвитком ієрархічної структури моделей середовища шляхом формування усе більш узагальнених, абстрактніших рівнів її представлення. Відповідно розвиватиметься і ієрархія в системах, вирішальних перелічені вище завдання шляхом переходу від образів зовнішнього середовища, що безпосередньо сприймаються сенсорами системи, до усе більш абстрактних образів. Наслідком цього буде розширення функціональних можливостей робота завдяки можливості автономного рішення усе більш складних неалгоритмуючих інтелектуальних завдань, включаючи те, що самовдосконалило в процесі активної взаємодії із зовнішнім середовищем при рішенні конкретних завдань.

Одним з найбільш узагальнених типів моделей середовища є логіколінгвістичні моделі. Вони застосовуються для найбільш складних об'єктів з неоднозначною реакцією на одні і ті ж ситуації, які не можуть бути описані формально математично і тому описується евристично на основі експертних оцінок на мові близькому природному. Приклади таких об'єктів управління – це передусім системи, що включають людей.

На рис. 1.2 показана узагальнена схема системи інтелектуального управління роботом, яка є конкретизацією загальної схеми управління сенсорного робота на рис. 1.1 в частині застосування штучного інтелекту при рішенні перелічених вище за п'ять завдань.

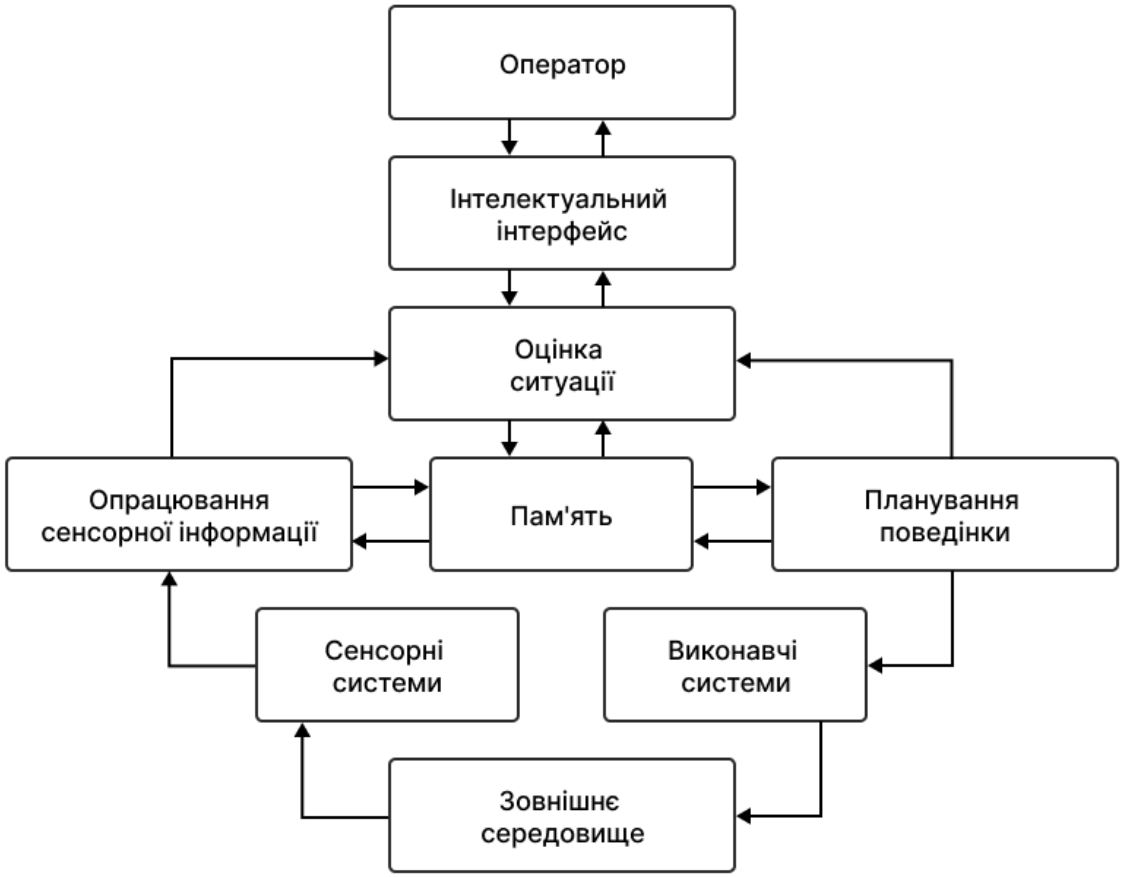


Рисунок 1.2 – Схема системи інтелектуального управління роботом [7]

У центрі схеми знаходиться блок пам'яті, двосторонньо пов'язаний з іншими системами, що переробляють інформацію. У цей блок входить база знань про зовнішнє середовище – ієрархічна модель зовнішнього середовища і база даних як про зовнішнє середовище, так і про самого робота і про операції, які він повинен виконувати. Крім того, спеціалізовані оперативні бази знань і даних, пов'язаних з цією центральною пам'яттю, можуть бути і в окремих системах робота.

База знань про зовнішнє середовище містить як апріорну інформацію, що вводиться до початку роботи, так і оперативну сенсорну, яка отримується

в процесі сприйняття довкілля при виконанні роботом заданих дій, а так само в процесі його спеціальних пізнавальних дій для вивчення цього середовища.

Усі інші блоки схеми так само мають ієрархічну структуру, рівні якої сполучені один з одним по вертикалі від низу до верху. У свою чергу показані на схемі з'єднання блоків здійснюються багатоканальний між однойменними рівнями по горизонталі.

Блок обробки сенсорної інформації отримує з блоку пам'яті екстраполяцію зміни стану зовнішнього середовища, а передає в нього корекцію цього стану на рівні безпосередньої сенсорної картини середовища.

Блок оцінки ситуації і блок планування поведінки отримують з блоку пам'яті поточну модель зовнішнього середовища, а передають в нього відповідно до її оцінку за певними критеріями і синтезований план управління рухом робота. При синтезі цього плану застосовується різні способи рішення завдань, розроблені у рамках штучного інтелекту, у тому числі [8]:

– пошук рішення в просторі станів (шляхом знаходження послідовності перетворення початкового стану в кінцеве цільове);

– зведенням завдання до підзадач (шляхом послідовного розбиття завдання на підзадачі аж до елементарних, рішення яких відоме);

– пошук у формі рішення теореми (шляхом формулювання завдання як теореми і її рішення на базі системи аксіом).

Блок інтелектуального інтерфейсу в загальному випадку має бути двосторонньо-пов'язаний з усіма переліченими вище функціональними блоками.

Особливістю розглянутої узагальненої схеми системи інтелектуального управління, як і загальної схеми на рис. 1.2, є те що в ній відсутній в явному виді блок, відповідальний за реалізацію способу інтелектуального управління, як це має місце для адаптивного і програмного управління. Пояснюється це тим, що штучний інтелект розподілений по усіх функціональних блоках схеми відповідно до перелічених вище функцій, при реалізації яких він може

вимагатися. У конкретних системах він може бути присутнім у будь-якому з цих блоків.

Нині застосування штучного інтелекту в системах управління роботами, як вже відзначалося, розпочинається з введення елементів штучного інтелекту в системи адаптивного управління на основі застосування перелічених вище інтелектуальних технологій.

1.2 Аналіз методів голосового управління МР

Існує декілька методів голосового управління МР. Одні різняться за типом з'єднання, а інші функціонально. Однак, усі вони є пристроями розпізнавання мови.

За типом з'єднання класифікація наступна: з'єднання по кабелю, Bluetooth, Wi-Fi з'єднання. У одному із прикладів такої системи, для передачі голосу та мовлення може використовуватися окремий модуль для прийому команд через звук, у свою чергу, він поєднаний з модулем обробки і перетворення голосових команд у сигнал за допомогою кабелю, або може знаходитись у його складі у якості єдиного модуля.

Найбільш поширено передача голосу для обробки здійснюється за допомогою Bluetooth або Internet по Wi-Fi. З'єднання по Bluetooth не потребує підключення до мережі Internet та є простішим у використанні, аніж Wi-Fi, хоча мінусом такого підходу є відносно невелика дальність передачі сигналу.

Передача даних через Internet по Wi-Fi потребує наявності підключення, але суттєвою перевагою такого методу є те, що управління системою можливе з будь-якої точки земної кулі.

1.2.1 Голосове керування МР на основі когнітивної моделі FCAS

Дана модель є відносно легковагою і не вимагає тренування на вимову конкретного диктора. Її зручність полягає також у можливості налаштування вільних параметрів для конкретних ситуацій та додатків.

Головним блоком моделі FCAS є ядро, в якому обробляється нечітка інформація, яка отримується від двох блоків – блоку розрахунку ваг мовних сегментів та ознакового (фонологічного) блоку (рис. 1.3). У загальному випадку модель також містить блок фонемного аналізу, однак у розпізнавання обмеженого набору голосових команд його можна виключити із загальної схеми з метою підтримки легковажності системи; при цьому функції фонемного блоку приймає на себе ознаковий блок, про що буде сказано далі.

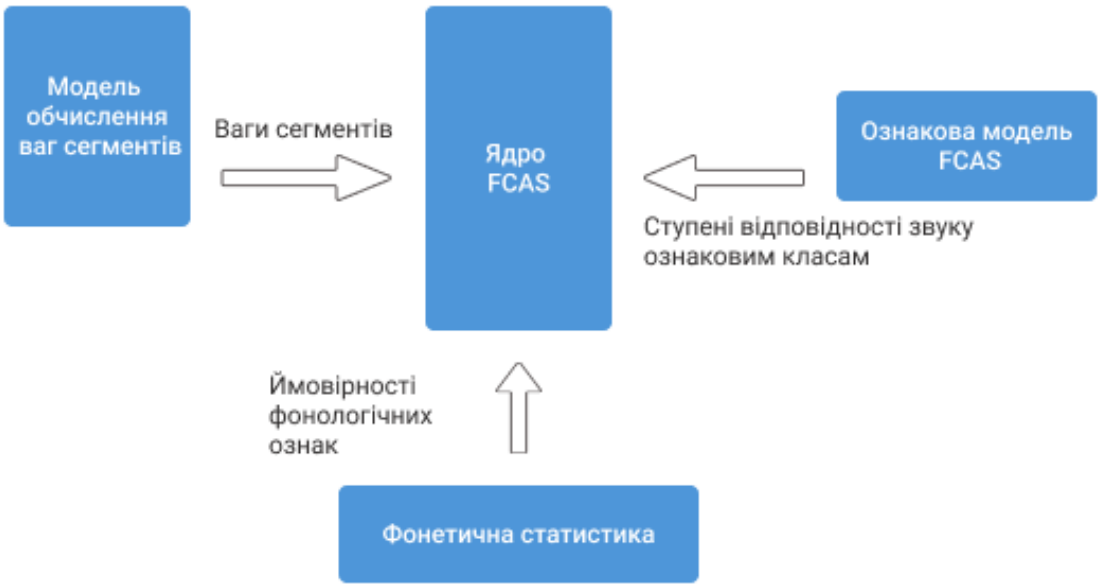


Рисунок 1.3 – Схема моделі FCAS [9]

Основною відмінністю запропонованої моделі від традиційно використовуваних у сучасних мовних технологіях СММ є багаторівневий облік акустикофонетичної інформації на етапі постобробки сигналу. У моделі FCAS не лише закладається статистика звуків мови та слів будь-якої мови з їх спектральними та кепстральними прототипами, а й здійснюється прийняття рішень на основі такої інформації, як: вага сегмента (модель обчислення ваг FCAS), фонологічний клас сегмента (ознакова модель FCAS) та послідовність сегментів звуків мови, що змінюються (ядро FCAS).

1.3 Дослідження методів ідентифікацій голосових команд

1.3.1 Приховані марковскі моделі

Прихована марковська модель (ПММ) – статистична модель, яка імітує роботу процесу, що схожий на марковський процес з невідомими параметрами, задачею якого ставиться розгадування невідомих параметрів на основі тих, котрі спостерігаються. ПММ може бути розглянута як найпростіша Байєсівська сітка довіри. Статистичні методи, основані на понятті марковського джерела, і ПММ були вперше застосовані для обробки мови Бейкером з CMU, а також Джелінеком і його колегами з IBM.

Ці моделі достатньо змістовні за своєю математичною структурою і, як наслідок, можуть скласти теоретичний фундамент для широкого кола додатків. Окрім того, правильне застосування цих моделей для рішення деяких важливих прикладних задач призводить до дуже гарних результатів. У додатку до мовного сигналу ланцюг Маркова будується як односпрямований процес переходу між станами в дискретні моменти часу, при цьому ймовірність переходу в наступний стан залежить тільки від поточного стану і не залежить від того, в яких станах перебував процес в попередні моменти часу . Ця вимога, однак, призводить до неправильних гістограм часу життя станів, і від неї в сучасних системах відмовилися. Моделі, в яких імовірність переходу в наступний стан залежить від часу перебування в поточному стані, називаються неоднорідними марковськими або напівмарковськими.

На даний час існує кілька типів ПММ, які різняться за топологією. На рис. 1.4 представлена топологія системи з трьома станами. Система є кінцевим автоматом, змінює стан у кожен дискретний момент. Перехід зі стану S_i в стан S_j з імовірністю a_j здійснюється випадковим чином. Модель породжує вектор спостережень O_n з ймовірністю $b_j(O_n)$ кожного дискретного моменту часу.

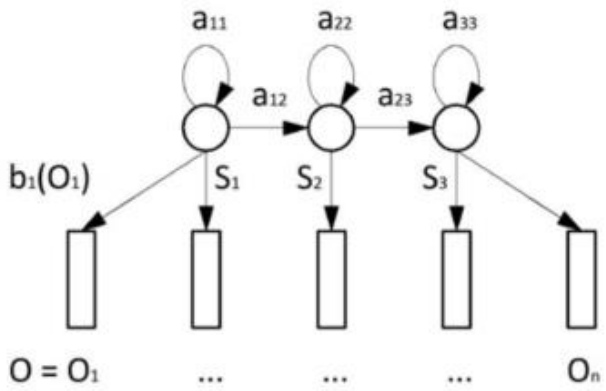


Рисунок 1.4 – Компоненти систем розпізнавання мови [10]

Таким чином, марковська модель деякого звуку або слова являє собою одне або декілька послідовних станів, для яких визначені функції щільності ймовірності в просторі ознак і ймовірності переходів. Ці функції можуть бути представлені в дискретному вигляді для проквантованого простору ознак або в безперервному вигляді. У другому випадку їх зазвичай апроксимують сумою гаусових функцій з діагональними матрицями коваріації. Діагональність матриць коваріації зменшує кількість навчених параметрів і спрощує деякі алгоритми, забезпечуючи аналітичні рішення деяких проблем, які для повних матриць коваріації допускають тільки чисельні рішення.

1.3.2 Динамічна деформація часу (DTW)

Важлива роль у вирішенні завдань розпізнавання мови належить методу Dynamic Time Warping (DTW), який інтерпретується як нелінійне розтягування-стиснення осі часу. DTW є алгоритмом для вимірювання подібності між двома послідовностями, які можуть змінюватися в часі або швидкості. Наприклад, схожість в рухах при ходьбі буде виявлено навіть якщо в одному відео людина йшла повільно, а в іншому швидше, і навіть якщо будуть прискорення і уповільнення в ході одного спостереження.

Він був домінуючою парадигмою для розпізнавання ізольованих слів із малим словником розпізнавання. DTW у своїй галузі застосування давав дуже

хороші результати і фактично перевершував ПММ. Від DTW сутнісно відмовилися через наступні проблеми [11]:

− запровадження моделі мови не було природним;

− завдання побудови синтетичних еталонів залишилося невирішеним;

− не знайдено єдиного статистичного формулювання розпізнавання, що включає всі модулі розпізнавання мовлення.

Відомо також про інші проблеми методу розпізнавання алгоритму DTW. Проблема великого розкиду довжин стандартів полягає в наступному: якщо довжина одного з еталонних сигналів значно менше довжин інших, міра розходження від нього до сигналу, що розпізнається, буде мінімальною. Інша проблема методу формулюється як «коректне тимчасове вирівнювання двох вимов різних слів не є чітко визначеною лінгвістичною концепцією».

Підхід DTW раніше використовувався повсюдно, але зараз його витіснили ПММ. DTW може бути застосована до відео, аудіо та графіку. Насправді, будь-які дані, які можуть бути перетворені в лінійне уявлення можуть бути проаналізовані за допомогою DTW.

1.3.3 Штучні нейронні мережі

Штучні нейронні мережі (ШНМ) – математичні моделі, а також їх програмні або апаратні реалізації, побудовані за принципом організації та функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, що протікають в мозку, і при спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі Маккалока і Піттса. Згодом, після розробки алгоритмів навчання, створювані моделі стали використовувати в практичних цілях таких як задачі прогнозування, для розпізнавання образів, та задачі управління. ШНМ є системою з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів). Такі процесори зазвичай досить прості, особливо в порівнянні з процесорами, використовуваними в персональних комп'ютерах. Кожен процесор подібної мережі має справу тільки з сигналами, які він

періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І тим не менш, будучи з'єднаними в досить велику мережу з керованим взаємодією, такі локально прості процесори разом здатні виконувати досить складні завдання.

З точки зору машинного навчання, нейронна мережа являє собою окремий випадок методів розпізнавання образів. З математичної точки зору, навчання нейронних мереж – це багато параметричне завдання нелінійної оптимізації. З точки зору кібернетики, нейронна мережа використовується в задачах адаптивного управління і як основа алгоритмів для робототехніки. З точки зору обчислювальної техніки та програмування, нейронна мережа – спосіб ефективного розпаралелювання завдань. А з точки зору штучного інтелекту, ШНМ є основою філософської течії конективізму і основним напрямком в структурному підході з вивчення можливості побудови (моделювання) природного інтелекту за допомогою комп'ютерних алгоритмів.

Нейронні мережі не програмуються в звичайному сенсі цього слова, вони навчаються. Можливість навчання – одне з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в навчальній вибірці, а також неповних і / або «зашумлених», частково перекручених даних.

Розглядаючи більш детально: штучна нейронна мережа – це математична модель біологічного нейрон. На сьогоднішній день найбільшу популярність в задачах розпізнавання мови отримали багатошарові нейронні мережі, які покликані заповнити недоліки стандартних сумішей нормальних розподілів.

Багатошарова нейронна мережа – це односпрямована штучна нейронна мережа з одним і більше рівнями (шарами) з прихованими нейронами між

вхідним і вихідним шарами. На вході нейронна мережа отримує акустичні параметри фіксованою розмірності, витягнуті з мовного сигналу. Кожен прихований нейрон j використовує логістичну функцію для відображення вхідного сигналу xj з попереднього рівня в вихідний сигнал yj для наступного рівня:

$$y_j = \frac{1}{1+e^{-x_j}},\ x_j = b_j + \sum_i y_i w_{ij}, \tag{1.1}$$

де b_j – коефіцієнт відхилення нейрона j;

i – індекс нейронів попереднього шару;

w_{ij} – вага зв'язку між i-им нейроном попереднього шару з даними j-им нейроном.

При класифікації, вихідний нейрон j перетворює вхідний сигнал x_j в ймовірність класу p_j як:

$$p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}, \tag{1.2}$$

де k – індекс, що пробігає по всіх класах.

Навчання нейронної мережі полягає в підборі оптимального набору ваг і відбувається за допомогою алгоритму зворотнього поширення помилки між істинним результатом і фактичним результатом, отриманим на виході нейронної мережі. Багатошарова нейронна мережа з безліччю шарів і нейронів у кожному шарі вимагає дуже великих обчислювальних ресурсів, даних і часу для проведення повноцінного навчання. Методи градієнтного спуску, які використовуються при навчанні, можуть знайти лише локальні оптимуми, якщо не правильно задати початкові значення ваг, тим самим зводячи мережу до перенавчання. Щоб уникнути такої ситуації і перенавчання мережі, був

запропонований принципово новий підхід навчання мережі, названий генеративним щодо попереднього навчання, ідея якого полягає в наступному. Кожен шар навчається окремо, причому вихідні дані попереднього шару, після його навчання, є вхідними даними для навчання наступного шару. Ваги, отримані в результаті такого навчання, є набагато кращими початковими умовами для проведення підсумкового дискримінативного навчання нейронної мережі, при якому дані ваги лише трохи зміняться. Багатошарові нейронні мережі через здатність апроксимувати будь-яку (статичну) нелінійну функцію можуть повністю замінити суміші нормальних розподілів, але до сих пір не знайдені підходи повної заміни ПММ, які показали перевагу в моделюванні часових (динамічних) залежностей.

Сьогоднішні системи автоматичного розпізнавання мови зробили крок вперед великими темпами за останні десятки років, починаючи з простих дикторозалежних додатків до диктороне залежних систем автоматичної транскрипції новин, телефонних розмов, лекцій. Незважаючи на повсюдне застосування таких систем, завдання розпізнавання мови далеко не вирішене в проблемах, пов'язаних з шумами, спотвореннями на лінії, іноземним акцентом, швидкістю і манерою мови та інше.

Проте, чимало досліджень по розпізнаванню мови активно ведуться в світлі останніх досягнень, і існує величезна кількість літератури на цю тему.

На рис. 1.5 показана загальна структура сучасних систем розпізнавання мови, що включають такі її компоненти як попередня обробка сигналу, акустична модель, мовна модель і пошук гіпотез. Первинна обробка сигналу включає такі процедури як витяг і трансформація акустичних характеристик сигналу, адаптація до шумових ефектів або до варіацій між різними дикторами. Стандартним підходом до вилучення акустичних характеристик є обчислення вхідних векторів з кепстральних коефіцієнтів MFCC або коефіцієнтів перцептивних лінійних пророкувань PLP. Далі вектори трансформуються лінійною проекційною матрицею в простір меншої розмірності так, що фонетичні класи поділяються якомога більше. Зазвичай це

здійснюється за допомогою лінійного дискримінаційного аналізу або його розширення HLDA.

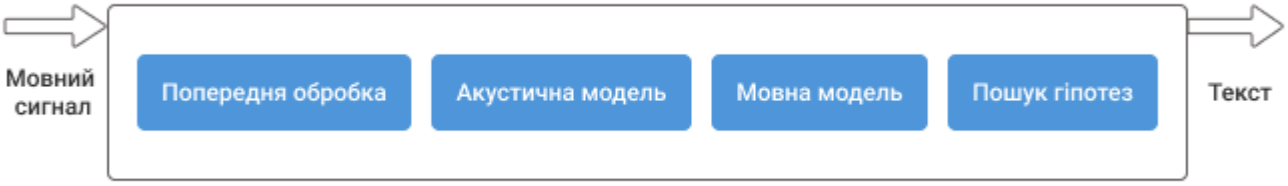


Рисунок 1.5 – Компоненти систем розпізнавання мови [12]

Для забезпечення більшої стійкості від шумів застосовуються алгоритми SPLICE або QE. Адаптація до різних дикторів проводиться шляхом нормалізації довжини голосового тракту до еталонного диктору, застосування лінійної регресії максимальної правдоподібності в просторі параметрів або мінімальної фонетичної помилки в просторі параметрів.

Акустична модель відображає акустичні характеристики звуків (фонем, діфонів, трифонів) мови, для якої будується система розпізнавання мови. Найбільш популярними тут є підходи, засновані на ПММ в зв'язці з сумішшю нормальних розподілів або ж на багатошарових нейронних мережах.

Успішними методами побудови мовних моделей є мовна модель KneyserNey, ієрархічна модель Pitman-Yor, модель максимальної ентропії або, так звана модель «М». Дані методи в засновані на статистичному підході. Однак існують і інші методи, які використовують синтаксичну структуру мови. Мета декодера обчислити найбільш ймовірну послідовність слів, маючи послідовність акустичних векторів, використовуючи при цьому інформацію акустичної та мовної моделей.

Однак застосовуються також методи на базі зважених кінцевих трансдюсерів. Експерименти, які використовують дані моделі, досягли високих показників продуктивності. Наприклад, дикторонезалежні системи розпізнавання телефонних розмов англійською мовою досягли рівня помилки

слів порядку 15,2%. Як приклади сучасних систем розпізнавання мови можна виділити системи з відкритим кодом HTK, CMU Sphinx, Kaldi, Julius, а також комерційні продукти Dragon NaturallySpeaking від Nuance, ViaVoice від IBM, Windows Speech Recognition від Microsoft, SIRI від Apple, Google Voice Search від Google [13].

Незважаючи на дуже значний прогрес в автоматичному розпізнаванні мови, досягнутому в останні 3-4 роки на тлі більш ніж тридцятирічного застою, можливості систем розпізнавання ще дуже обмежені в порівнянні з людиною. В основному переваги слухової системи визначаються надзвичайно потужними можливостями по адаптації і, головне, «кінцевим пристроєм» слухової системи, що здатен розуміти сказане, завдяки яким людина не відчуває труднощів при розпізнаванні віддаленої, ревербованної, акцентної мови, мови в поганих каналах зв'язку, а також може виділяти мову одного диктора з багатоголосся і розпізнавати спонтанну мову.

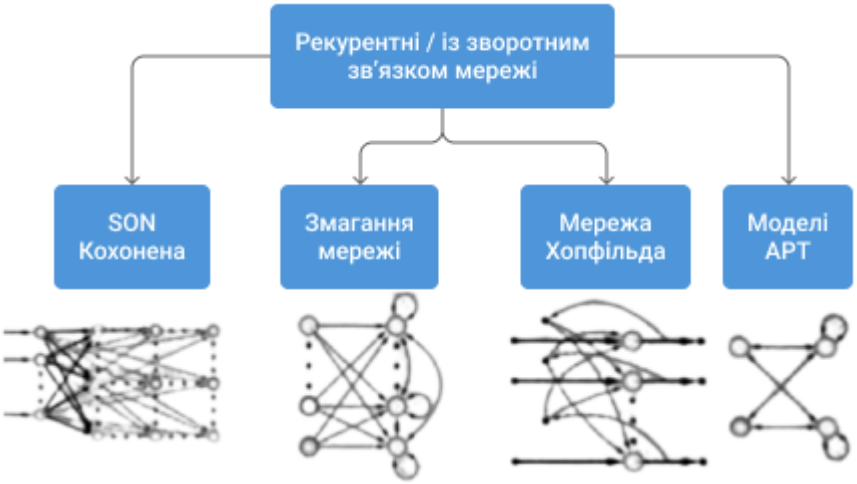
Всі ці завдання, а особливо останні два, представляють великі труднощі для сучасних систем розпізнавання. Автоматичні системи розпізнавання мови поки що перевершують людину тільки в задачах, де розуміння і модель мови не грають ролі, наприклад, при розпізнаванні ізольованих команд або чисел. Розвиток систем розпізнавання мови буде пов'язано з удосконаленням структури нейронних мереж, обов'язковою наявністю зворотних зв'язків на різних рівнях і розробкою нових методів навчання таких нейронних мереж з використанням алгоритму динамічного програмування.

Структура нейронних мереж повинна буде мати механізми адаптації і повернення для корекцій. Буде недивним, якщо в архітектурі нейронних мереж з'являться деякі елементи слухової системи, а методи навчання запозичать від навчання дитини, наприклад, пред'явлення на першому етапі найпростіших звуків – модульованих голосних і сполучень приголосна-голосна.

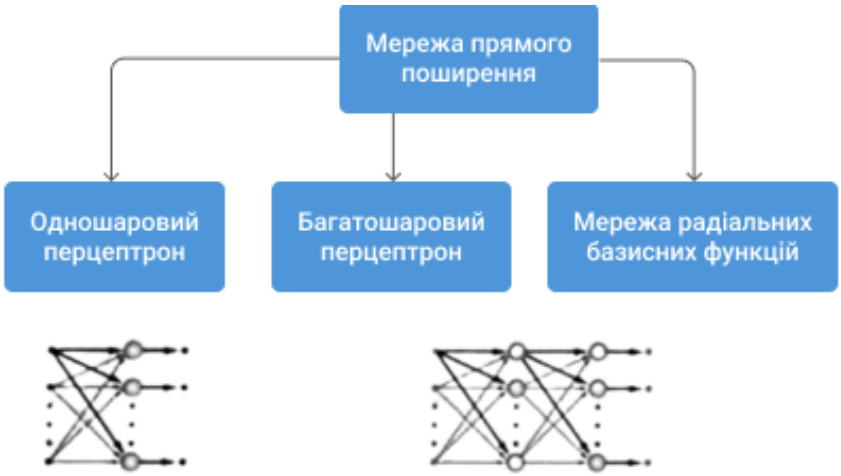
1.4 Дослідження топологій нейронних мереж для застосування в системах управління

ШНМ мають багато варіантів різних конфігурацій. Незважаючи на це, мережі мають багато спільного між собою.

Нейронні мережі класифікують за топологічними типами у відповідності до виду зв'язків між нейронами мережі, а також за видами використаних формальних нейронів. По виду зв'язків нейронні мережі можуть бути розподілені в два класи: мережі прямого поширення, у яких немає петель (рис. 1.6, а) і мережі зі зворотними зв'язками (рис. 1.6, б).



(а)



(б)

Рисунок 1.6 – Типи нейронних мереж [14]

Їх також класифікують за способом подачі інформації на вхід:

− подачу на синапси вхідних нейронів;

− подачу до виходів вхідних нейронів;

− подачу у вигляді синапсичних вагових коефіцієнтів вхідних нейронів;

− адитивну подачу на синапси вхідних нейронів.

За способом зняття інформації з виходів нейронної мережі розрізняють:

− зняття сигналів з виходів вихідних нейронів;

− зняття сигналів з синапсів вихідних нейронів;

− зняття сигналів у вигляді значень ваги синапсів вихідних нейронів;

− адитивний зйом сигналів із синапсів вихідних нейронів.

За організацією навчання розрізняють:

− навчання нейронних мереж з вчителем;

− без вчителя.

За способом навчання:

− навчання за входами (навчальний приклад є вектором вхідних сигналів);

− навчанні за виходами використовують вектор вихідних сигналів, який точно відповідає вхідному вектору.

1.5 Постановка завдання дослідження

Сучасні МР можуть самостійно переміщатися в навколишньому просторі і виконувати необхідні дії за допомогою маніпуляторів. Робот оснащений системою технічного зору і комплексом інформаційних датчиків, здатних сформувати комплексне уявлення про поточну ситуацію. База знань робота, дозволяє йому самостійно орієнтуватися в навколишньому середовищі і приймати рішення про дії, необхідні для вирішення поставленого завдання. Таким чином, маніпуляційний МР являє собою «інтелектуальну» технічну систему, здатну до автономного поведінки.

Одним із рішень проблеми обміну інформацією між роботом та людиною-оператором вважається створення системи мовного управління. Завдання управління роботом з боку оператора в цьому випадку включає діалог проблемно-орієнтованою мовою, близькою до природної, і спостереження за діями робота. Це дозволить людям, які не мають можливість використовувати традиційні засоби інтерфейсу взаємодіяти з роботом, і навіть знизить стомлюваність під час роботи і зробить її цікавішою.

Отже, об’єктом дослідження є процес керування МР, а предметом дослідження є моделі та методи голосового керування.

Результатом виконання кваліфікаційної роботи повинна бути розроблена система ідентифікацій команд для керування МР на базі нейронних мереж.

2 РОЗРОБКА ТОПОЛОГІЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ГОЛОСОВОГО КЕРУВАННЯ

2.1 Обґрунтування вибору рекурентної нейронної мережі

Нейронна мережа з кількістю прихованих шарів є універсальним апроксиматором, тобто навіть мережі з одним прихованим шаром, що використовувалися до цього етапу, можуть апроксимувати будь-яку поверхню у просторі ознак. Проте успіх у розпізнаванні мови прийшов лише з використанням багатошарових мереж. Це пояснюється неможливістю або крайньою складністю створення розумної методики ініціалізації ваг для мереж з одним прихованим шаром, що призводить до далекого від оптимуму набору ваг під час навчання.

Одним із методів є ініціалізація за допомогою пошарового навчання, починаючи з нижніх шарів. Як цільову функцію для першого прихованого шару розглядається вхідний вектор ознак. Щоб уникнути тотожного перетворення, вхідний вектор зашумлюють. Наступний шар нейронної мережі таким чином навчають відтворювати вихідні сигнали попереднього шару.

Усього в такий спосіб навчають до 5-7 шарів. Після того, як ініціалізація перших шарів проведена, включають стандартний алгоритм зворотного розповсюдження помилки для всієї мережі з цільовою функцією, що відображає належність вхідного сигналу до трифону.

Оскільки нейронні мережі не можуть ідентифікувати динамічні об'єкти, для порівняння моделей із сигналом, то використовується формалізм марківських моделей, проте тепер як вектор ознак використовується набір апостеріорних ймовірностей трифонів, отриманий на виході нейронної мережі. Більш істотним недоліком, властивим даному методу, є те, що глибокі нейронні мережі не можуть розпізнавати динамічні об'єкти, через що й доводиться використовувати алгоритми марківської моделі. Недоліки

марковської моделі досить очевидні: дискретність, чи незалежність послідовних станів друг від друга; відсутність глибоких тимчасових зв'язків, тобто нездатність розпізнавати траєкторії у просторі ознак як інформативні об'єкти. Можна припустити, що обидва зазначені недоліки можна подолати, використовуючи рекурентних нейронні мережі (РНМ).

РНМ містять нейрони, які об'єднані в спрямований круговий процес. Це наділяє нейронну мережу пам'яттю і, отже, здатністю розпізнавати процеси, а чи не лише статичні об'єкти, як розглянуті вище глибокі нейронні мережі. РНМ відрізняються від багатошарових тим, що з обробці чергового вектора ознак система враховує також внутрішні стани нейронів, які, своєю чергою, формуються попередніми векторами ознак і станами попередні моменти часу.

У цьому сенсі одинична рекурентна нейронна мережа є більш потужною освітою, ніж глибока нейронна мережа. Проте, розглядаються ієрархічні комбінації РНМ та комбінації рекурентних та багатошарових мереж. Це можна пояснити, як і для багатошарових мереж, бажанням структурувати систему, наблизити її до принципів функціонування нервової системи, спростити процедуру ініціалізації та навчання.

РНМ є основною моделлю розпізнавання мови, що використовується для прогнозування. На рис. 2.1 показано розпізнавання мовлення з використанням РНМ. Тепер звук, який подається на вхід, легко обробляти, він подаватиметься в глибоку нейронну мережу. Після подачі в мережу невеликих звукових фрагментів він визначить букву, що вимовляється.

РНМ використовує ідею послідовної інформації, так як ця нейронна мережа має пам'ять, що впливає на майбутні прогнози. Для прогнозів використовується послідовна інформація, що зберігається у пам'яті РНМ. Ідея використовувати РНМ полягає в тому, що в традиційній нейронній мережі передбачається, що кожен вхід і кожен результат залежать один від одного.

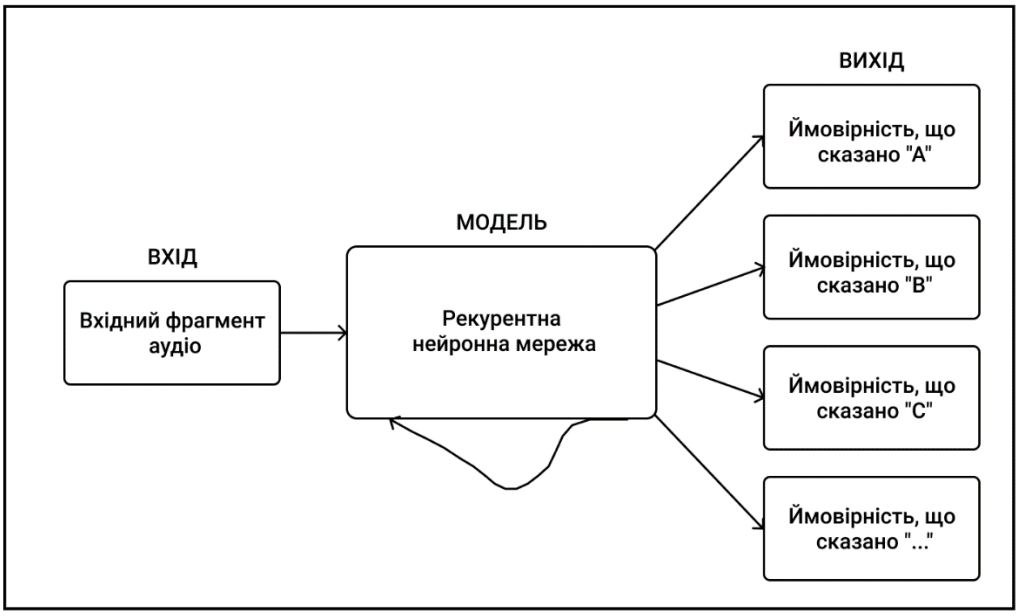


Рисунок 2.1 – Розпізнавання мовлення з використанням РНМ [15]

Отже, використання традиційної нейронної мережі – погана ідея при обробці мови. Передбачення будь-яких слів у реченні вимагає інформації про слово, що використовується до того, як минуле слово обробляється. Наявність пам'яті – одне з особливостей РНМ, що робить її унікальною проти іншими мережами. Отже, РНМ є найефективнішою для розпізнавання мови.

2.2 Розробка топології нейронної мережі

Рекурентні нейронні мережі – вид нейронних мереж, архітектура яких складається з трьох шарів: вхідного, прихованого та вихідного. При цьому прихований шар має зворотній зв’язок сам на себе.

На відміну від нейронних мереж з прямим зв’язком, рекурентна нейронна мережа може приймати на вхід послідовність векторів $x = (x_1, x_2, \dots, x_T)$ в заданому порядку. Робота рекурентної нейронної мережі описується співвідношеннями (2.1) і (2.2).

$$h_t = \varphi(W h_{t-1} + W_x x_t + b_h), \tag{2.1}$$

$$y_t = \varphi(W_y h_t + b_{y)}, \tag{2.2}$$

де φ – нелінійна функція;

x_t – вхідний вектор за номером t;

h_t – стан прихованого шару для входу x_t;

y_t – вихід мережі для входу x_t;

W, W_x, W_y – вагові коефіцієнти матриці нейронної мережі;

b_h, b_y – вектори здвигу.

За рекурентним співвідношенням (2.1) є можливість урахування результатів перетворення нейронної мережею інформації на попередніх етапах для обробки вхідного вектора на наступному етапі функціонування мережі. На рис. 2.2 представлена схема роботи рекурентної нейронної мережі.

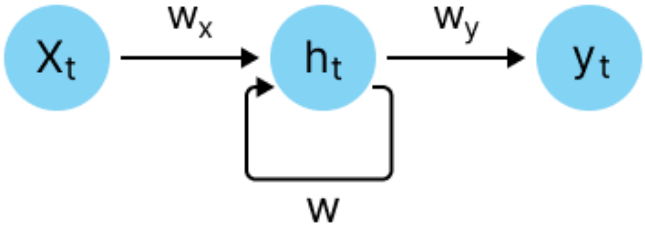


Рисунок 2.2 – Схема рекурентної нейронної мережі [16]

Гнучкість рекурентних нейронних мереж дозволяє навчати їх генеративно. При генеративному навчанні, застосовуючи до виходу y_t нормувальну функцію g, отримаємо, що $g(y_t)$ – це розподіл на наступний елемент послідовності, що враховує знання про попередні елементи послідовності. Використання правила перемноження за формулами (2.3) і (2.4), дає можливість оцінити ймовірність всієї послідовності.

$$g(y_t) = p(x_{t+1}|x_1, \dots, x_t), \tag{2.3}$$

$$p(x_1, \dots, x_\mathrm{T}) = p(x_1)p(x_1|x_2)p(x_3|x_1, x_2) \dots p(x_\mathrm{T}|x_1, \dots, x_{\mathrm{T}-1}). \qquad (2.4)$$

Можна стверджувати, що використання глибоких нейронних мереж РНМ є однією з найбільш сучасних технологій автоматичного розпізнавання мовлення.

2.3 Математичний розрахунок елементів нейронної мережі

Нейрони є найважливішими компонентами нейронної мережі. Точніше, нейронні мережі формуються шляхом з'єднання одного нейрона з кожним іншим нейроном. На відміну від нервової системи людини, нейрони в нейронних мережах пов'язані один з одним через шари. Час обробки буде надзвичайно великим, враховуючи, що кожен нейрон буде пов'язаний з кожним іншим нейроном. Щоб скоротити час обробки та заощадити обчислювальну потужність комп'ютера, у нейронній мережі використовуються шари. Завдяки шарам кожен нейрон у шарі пов'язаний з нейронами у наступному шарі. На рис 2.3 нижче представлено базову структуру нейронної мережі з трьома шарами [17].

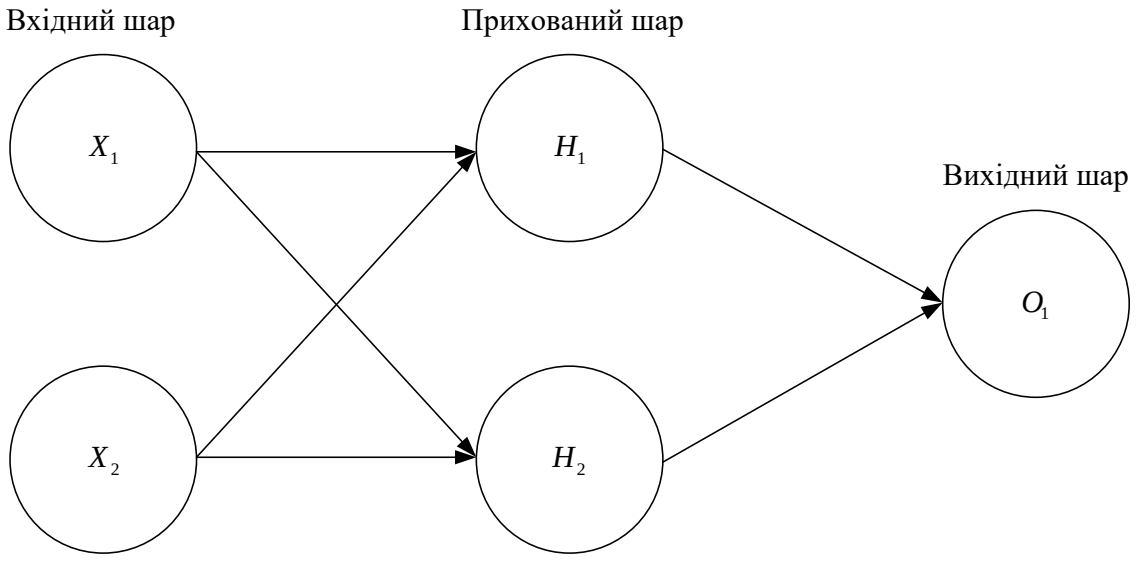


Рисунок 2.3 – Загальний вигляд нейрону з одним прихованим шаром

Продуктивність нейронних мереж залежить від кількості шарів та кількості нейронів в них. Ефективність прогнозування між мережею з трьома прихованими шарами та трьома нейронами в кожному шарі та мережею з одним шаром та одним нейроном може бути різною (рис. 2.4).

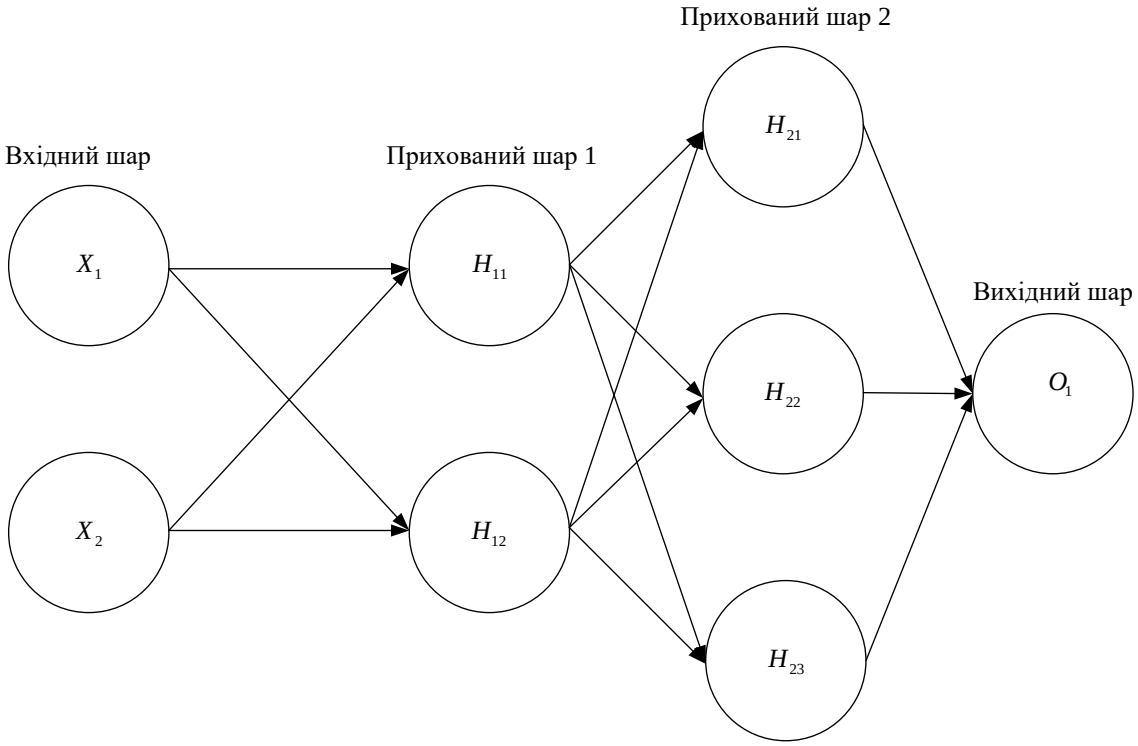


Рисунок 2.4 – Загальний вигляд нейрону з двома прихованими шарами

Нейронні мережі працюють протягом ітерацій, і кожна ітерація навчає модель отримання кращого прогнозу. Отже, живлення нейронів – це основний рух, який тренує нашу мережу та називається прямим зв'язком у нейронних мережах. Це означає отримання даних від попередніх підключених нейронів, виконання обчислень з цими даними та надсилання результату наступним підключеним нейронам. Коли у вихідному нейроні виконано остаточний розрахунок, із набору даних береться ще одне спостереження і знову вирушає на обробку. Цей процес триває доти, доки прогноз нашої мережі не стане справді близьким до фактичного значення, яке було передбачено. Найголовнішими є обчислення у нейронах на рис. 2.5.

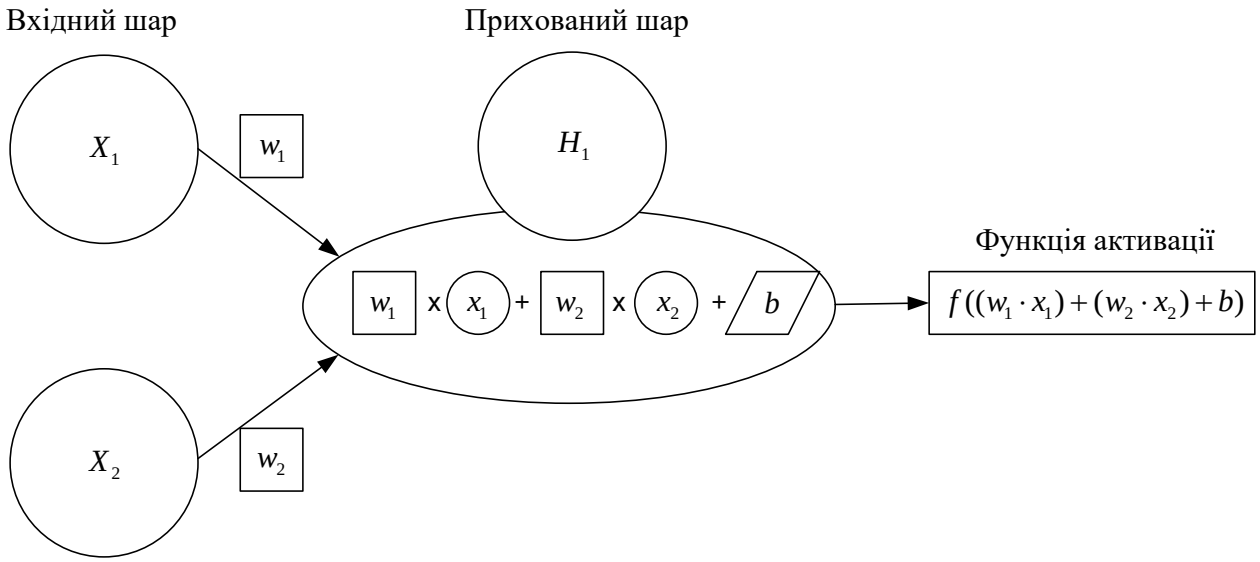


Рисунок 2.5 – Отримання функції активації

Навчимо мережу прогнозувати та розпізнавати аудіофайли, представлені у вигляді цифр, використовуючи довжину та ширину. Це означає, що будуть використовуватися дві змінні як предиктори і одна змінна як передбачення. Набір даних, який використовуватимуться для навчання, показано у табл. 2.1.

Таблиця 2.1 – Дані для навчання

Дані	Ширина	Довжина
5	5	7
8	4	9
9	6	12
4	3	6
3	2	3

Підставивши дані до формул на рис. 2.5, отримаємо наступні дані на рис. 2.6.

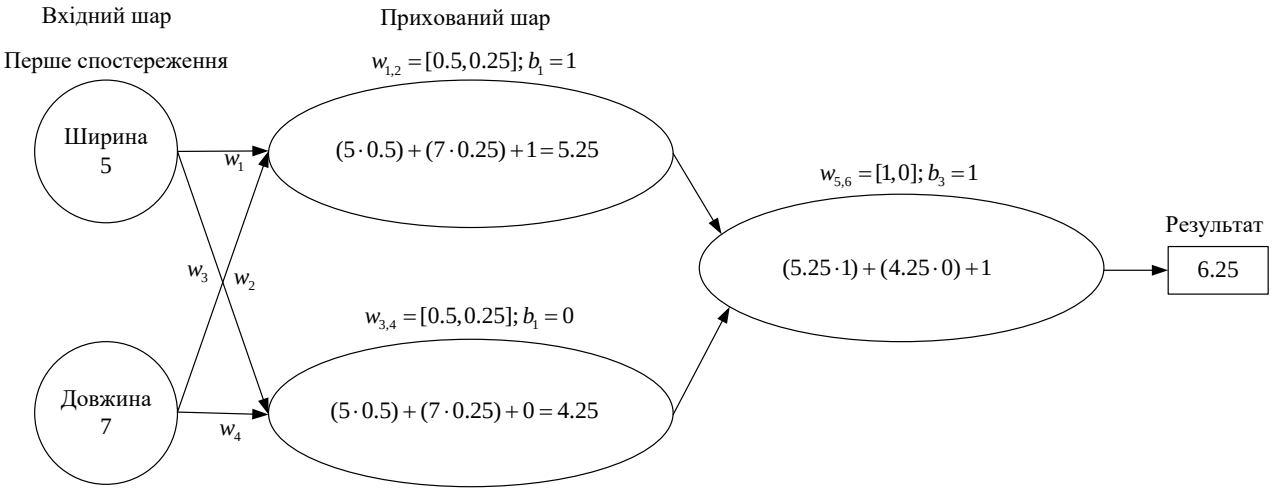


Рисунок 2.6 – Розрахунок функції активації

Після обчислень у нейронах як кінцевий результат на першій ітерації було знайдено 6,25. Можна сказати, що це перше передбачене значення. Далі потрібно порівняти цей результат із фактичним віком для цього спостереження. Вік дерева для першого спостереження дорівнює 5. Це означає, що відстань від фактичного значення становить 1,5 віку, і потрібно оновити ваги нейронів відповідно до цієї відстані. Цей процес називається зворотним поширенням. Далі буде використано 2 приховані шари по 2 нейрони в кожному в цій штучній нейронної мережі (рис. 2.7).

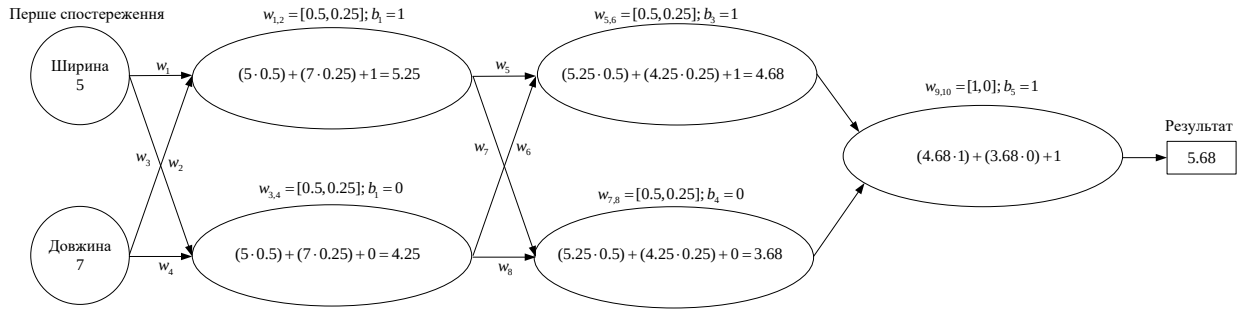


Рисунок 2.7 – Розрахунок функції активації для двох прихованих шарів

Розрахуємо середню квадратичну помилку (СКП) за форм. (2.5):

$$СКП = \frac{1}{n}\sum_{i=1}^{n}(Y_{істине} - Y_{прогнозування})^2 . \quad (2.5)$$

Тепер можна обчислити різницю між фактичним значенням та прогнозованим значенням у попередньому прикладі, використовуючи MSE. Прогнозований вік дорівнював 6,25 для першого спостереження, а фактичний вік дорівнював 5. Оскільки потрібно обрати одну вибірку для однієї ітерації, n дорівнюватиме 1:

$$СКП = \frac{1}{n}\sum_{i=1}^{n}(Y_{істине} - Y_{прогнозування})^2 == \frac{1}{1}\sum_{i=1}^{1}(5-6,25)^2 = 1,5625 .$$

Згідно з наведеними вище математичними операціями СКП знайдено рівним 1,5625. Можна використовувати термін втрати цього значення.

Далі необхідно змінити ваги та усунення, враховуючи їх вплив на втрати. Тому потрібно використати метод приватної похідної, адже часткові похідні функції втрат L з вагами та зсувами можуть надати інформацію про те, як ваги та зсуви впливають на втрати. Отже, потрібно виконати наступну операцію із частковою похідною:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial H_1} \cdot \frac{\partial H_1}{\partial w_1} , \quad (2.6)$$

$$Y_{прогнозування} = O_1 = (w_5 \cdot H_1) + (w_6 \cdot H_2) + b_3 , \quad (2.7)$$

$$H_1 = (w_1 \cdot x_1) + (w_2 \cdot x_2) + b_1 . \quad (2.8)$$

Далі необхідно використати формулу функції втрат СКП.

Відповідно до того, що на кожному кроці здійснюється лише одне спостереження. Отже, значення n дорівнюватиме 1:

$$СКП = L = \sum_{i=1}^{1} (Y_{істине_i} - Y_{прогнозування_i})^2, \tag{2.9}$$

$$L = (Y_{істине_i} - Y_{прогнозування_i})^2, \tag{2.10}$$

$$L = (Y_{істине})^2 - 2 \cdot Y_{істине} \cdot Y_{прогнозування} + (Y_{прогнозування})^2 \ . \tag{2.11}$$

Підставивши отримаємо:

$$\frac{\partial(L)}{\partial(Y_{прогнозування})} = \frac{\partial((Y_{істине})^2 - 2 \cdot Y_{істине} \cdot Y_{прогнозування} + (Y_{прогнозування})^2)}{\partial(Y_{прогнозування})} = \\ = -2 \cdot Y_{істине} + 2 \cdot Y_{прогнозування} = -2(Y_{істине} - Y_{прогнозування}), \tag{2.12}$$

$$\frac{\partial(L)}{\partial(Y_{прогнозування})} = -2(Y_{істине} - Y_{прогнозування}). \tag{2.13}$$

Основна ідея цієї операції полягає в тому, щоб з'ясувати, як прогноз впливає на втрати:

$$\frac{\partial(Y_{прогнозування})}{\partial(H_1)} = w_5. \tag{2.14}$$

Остання похідна:

$$\frac{\partial H_1}{\partial w_1} = x_1. \tag{2.15}$$

Отримаємо:

$$\frac{\partial L}{\partial w_1} = -2(Y_{істине} - Y_{прогнозування}) \cdot w_5 \cdot x_1. \qquad (2.16)$$

Далі підставимо:

$$Y_{істине} = 5,$$

$$Y_{прогнозування} = 6,25,$$

$$w_5 = 1,$$

$$x_1 = 5,$$

$$\frac{\partial L}{\partial w_1} = -2(5-6) \cdot 1 \cdot 5 = 12,25.$$

Проведемо розрахунок градієнтного спуску (ГС). У ГС, після того, як буде знайдено похідні для всіх ваг і точок перетину, потрібно помножити їх на швидкість навчання і відняти їх старих ваг і точок перетину. Отже, можна виконати такі операції, щоб дізнатися про нове оновлене значення w_1, де η – швидкість навчання, w_1 – вага:

$$w_1 = 0,5,$$

$$\eta = 0,05,$$

$$(\frac{\partial L}{\partial w_1}) = 12,25.$$

$$w_{1_{нове}} \leftarrow w_1 - \eta \cdot \left(\frac{\partial L}{\partial w_1} \right), \qquad (2.17)$$

$$w_{1_{нове}} = 0,5 - 0,05 \cdot 12,25 = -0,1125.$$

Як можна побачити з вище, нове значення w_1 було знайдено як -0,1125. Тому потрібно замінити старе значення на -0,1125 і використовувати це значення в наступній ітерації. Отже, відоме лише нове значення w_1, та потрібно знайти нові значення інших значень та точок перетину.

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial H_1} \cdot \frac{\partial H_1}{\partial w_2} = -2(Y_{істине} - Y_{прогнозування}) \cdot w_5 \cdot x_2, \quad (2.18)$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial H_1} \cdot \frac{\partial H_1}{\partial b_1} = -2(Y_{істине} - Y_{прогнозування}) \cdot w_5 \cdot 1, \quad (2.19)$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial H_2} \cdot \frac{\partial H_2}{\partial w_3} = -2(Y_{істине} - Y_{прогнозування}) \cdot w_6 \cdot x_1, \quad (2.20)$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial H_2} \cdot \frac{\partial H_2}{\partial w_4} = -2(Y_{істине} - Y_{прогнозування}) \cdot w_6 \cdot x_2, \quad (2.21)$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial H_2} \cdot \frac{\partial H_2}{\partial b_2} = -2(Y_{істине} - Y_{прогнозування}) \cdot w_6 \cdot 1, \quad (2.22)$$

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial w_5} = -2(Y_{істине} - Y_{прогнозування}) \cdot H_1, \quad (2.23)$$

$$\frac{\partial L}{\partial w_6} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial w_6} = -2(Y_{істине} - Y_{прогнозування}) \cdot H_2, \quad (2.24)$$

$$\frac{\partial L}{\partial b_3} = \frac{\partial L}{\partial Y_{прогнозування}} \cdot \frac{\partial Y_{прогнозування}}{\partial b_3} = -2(Y_{істине} - Y_{прогнозування}) \cdot 1. \quad (2.25)$$

Підставивши наступні значення, отримаємо:

$$w_2 = 0,25,\ w_3 = 0,5,\ w_4 = 0,25,\ w_5 = 1,\ w_6 = 0,\ b_1 = 1,\ b_2 = 0,\ b_3 = 1,$$

$$x_1 = 5, x_2 = 7, Y_{прогнозування} = 6,25, H_1 = 5,25, H_2 = 4,25,$$

$$w_{2_{нове}} = 0,25 - 0,05 \cdot (-2 \cdot (5 - 6,25)) \cdot 1 \cdot 7 = -0,65,$$

$$b_{1_{нове}} = 1 - 0,05 \cdot (-2 \cdot (5 - 6,25)) \cdot 1 \cdot 1 = 0,87,$$

$$w_{3_{нове}} = 0,5 - 0,05 \cdot (-2 \cdot (5 - 6,25)) \cdot 0 \cdot 5 = 0,5,$$

$$w_{4_{нове}} = 0,25 - 0,05 \cdot (-2 \cdot (5 - 6,25)) \cdot 0 \cdot 7 = 0,25,$$

$$b_{2_{нове}} = 0 - 0,05 \cdot (-2 \cdot (5 - 6,25)) \cdot 0 \cdot 1 = 0,$$

$$w_{5_{нове}} = 1 - 0,05 \cdot (-2 \cdot (5 - 6,25)) \cdot 5,25 = 0,34,$$

$$w_{6_{нове}} = 0 - 0,05 \cdot (-2 \cdot (5 - 6,25)) \cdot 4,25 = -0,53,$$

$$b_{3_{нове}} = 1 - 0,05 \cdot (-2 \cdot (5 - 6,25)) \cdot 1 = 0,87.$$

Підставивши до схеми навчання, отримали наступне (рис. 2.8).

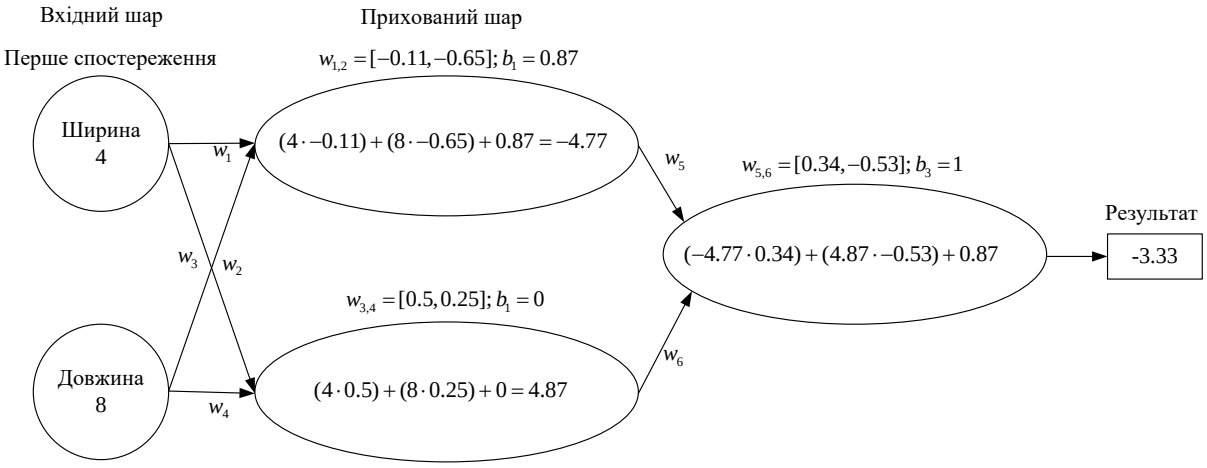


Рисунок 2.8 – Перша ітерація з першим зразком набору даних

Таким чином, було завершено першу ітерацію з першим зразком даних.

2.4 Висновки до другого розділу

В ході написання другого розділу було обрано тип нейронної мережі для голосового керування та розроблено її топологію. Також було виконано математичний розрахунок елементів нейронної мережі у результаті якого було навчено мережу прогнозувати та розпізнавати аудіофайли, представлені у вигляді цифр, використовуючи довжину та ширину.

3 РОЗРОБЛЕННЯ СИСТЕМИ ІДЕНТИФІКАЦІЇ КОМАНД ОПЕРАТОРА ДЛЯ КЕРУВАННЯ МОБІЛЬНИМ РОБОТОМ НА БАЗІ НЕЙРОННИХ МЕРЕЖ

3.1 Розробка алгоритму роботи системи

Алгоритм роботи програми представлено на рис. 3.1.

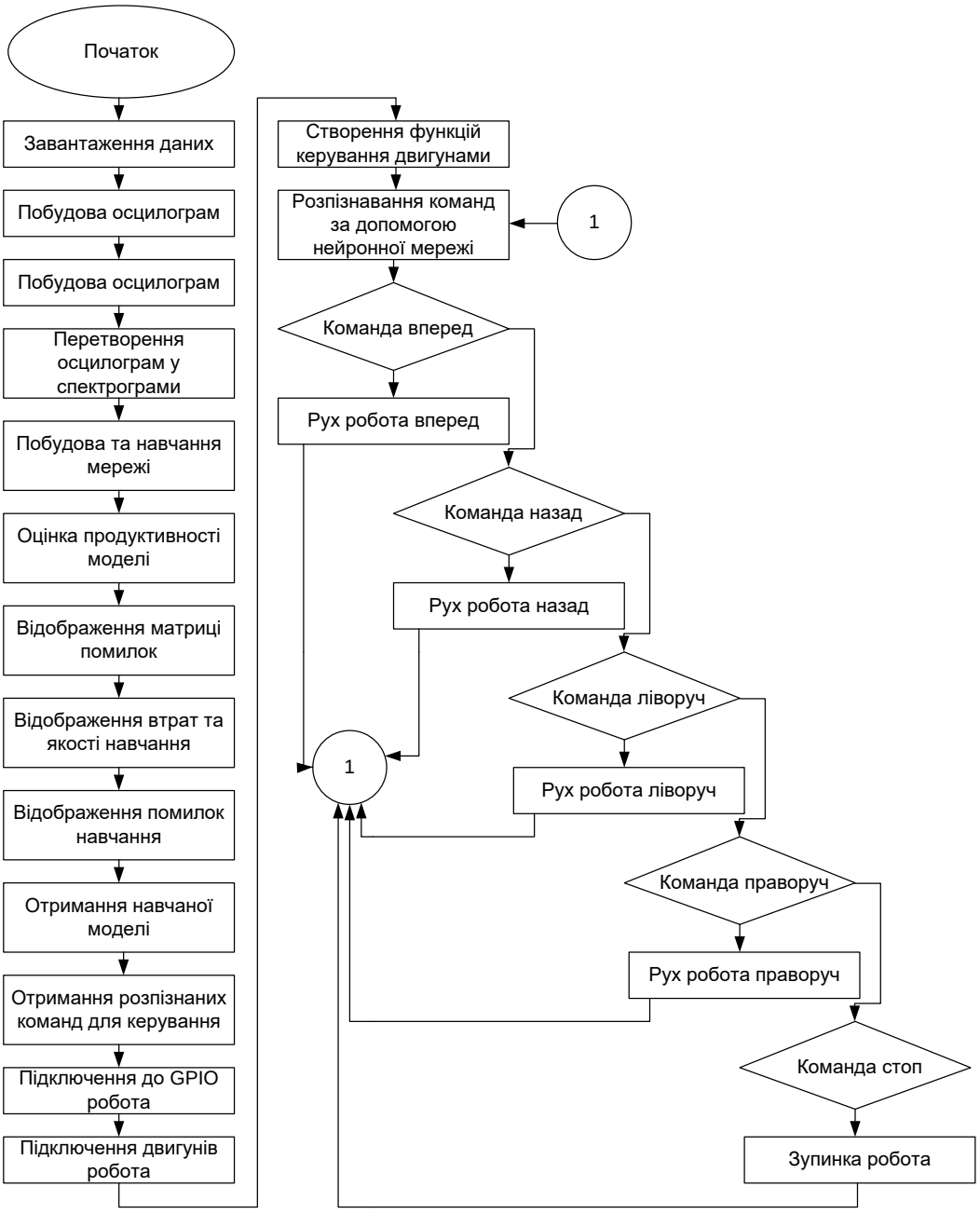


Рисунок 3.1 – Блок-схема алгоритму роботи програми

3.2 Реалізація розробленого програмного продукту

Для розроблення програмного продукту було використано наступні бібліотеки:

– os – бібліотека для роботи з операційною системою;

– pathlib – бібліотека пропонує класи, що представляють шляхи файлової системи з семантикою, що підходить для різних операційних систем;

– matplotlib – бібліотека для побудови графіків;

– numpy – бібліотека для математичних розрахунків;

– seaborn – бібліотека візуалізації даних на основі matplotlib, забезпечує інтерфейс високого рівня для малювання привабливої та інформативної статистичної графіки;

– tensorflow – відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявляння та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди;

– keras – відкрита нейромережна бібліотека, написана мовою Python;

– Ipython – інтерактивна оболонка мови програмування Python.

Програмно це має такий вигляд:

```
import os
import pathlib
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import tensorflow as tf
from keras import layers
from keras import models
from IPython import display.
```

Щоб заощадити час із завантаженням даних, їх було обрано з набору даних Speech Commands. Оригінальний набір даних складається з понад 105 000 аудіофайлів у форматі аудіофайлів WAV, де люди вимовляють 35 різних слів [18].

Було завантажено та розпаковано архів mini_speech_commands.zip, що містить менші набори даних Speech Commands за допомогою tf.keras.utils.get_file:

```
DATASET_PATH = 'data/mini_speech_commands'

data_dir = pathlib.Path(DATASET_PATH)
if not data_dir.exists():
    tf.keras.utils.get_file(
    'mini_speech_commands.zip',
    origin="http://storage.googleapis.com/download.tensorflow.org/data/
mini_speech_commands.zip",
    extract=True,
    cache_dir='.', cache_subdir='data').
```

Аудіозаписи набору даних зберігаються у восьми папках, які відповідають кожній команді: no, yes, down, go, left, up, right, and stop:

```
commands = np.array(tf.io.gfile.listdir(str(data_dir)))
commands = commands[commands != 'README.md']
print('Commands:', commands).
```

Аудіофайли тривають 1 секунду або менше на 16 кГц. Output_sequence_length=16000 розширює короткі до 1 секунди (і обрізає довші), щоб їх можна було легко групувати. На рис. 3.2 показано результат завантаження файлів для тренування:

```
Commands: ['down' 'go' 'left' 'no' 'right' 'stop' 'up' 'yes']
Found 8000 files belonging to 8 classes.
Using 6400 files for training.
Using 1600 files for validation.
```

Рисунок 3.2 – Завантаження файлів для тренування

```
train_ds, val_ds = tf.keras.utils.audio_dataset_from_directory(
    directory=data_dir,
    batch_size=64,
    validation_split=0.2,
    seed=0,
    output_sequence_length=16000,
    subset='both')

label_names = np.array(train_ds.class_names)
print()
print("label names:", label_names).
```

Цей набір даних містить лише одноканальне аудіо, тому використовуйте функцію tf.squeeze, щоб видалити додаткову вісь:

```
def squeeze(audio, labels):
  audio = tf.squeeze(audio, axis=-1)
  return audio, labels

train_ds = train_ds.map(squeeze, tf.data.AUTOTUNE)
val_ds = val_ds.map(squeeze, tf.data.AUTOTUNE).
```

Функція utils.audio_dataset_from_directory повертає лише до двох розділень. Доцільно зберігати тестовий набір окремо від набору перевірки, наприклад, в окремому каталозі, але з метою розділення набору перевірки на дві половини, було використано Dataset.shard. Ітерація по будь-якому сегменту завантажить усі дані та збереже лише їхню частину:

```
test_ds = val_ds.shard(num_shards=2, index=0)
val_ds = val_ds.shard(num_shards=2, index=1).
```

Для побудови кількох звукових сигналів використано наступний код:

```
rows = 3
cols = 3
n = rows * cols
fig, axes = plt.subplots(rows, cols, figsize=(16, 9))
for i in range(n):
  if i>=n:
    break
  r = i // cols
  c = i % cols
  ax = axes[r][c]
  ax.plot(example_audio[i].numpy())
  ax.set_yticks(np.arange(-1.2, 1.2, 0.2))
  label = label_names[example_labels[i]]
  ax.set_title(label)
  ax.set_ylim([-1.1,1.1])
plt.show()
```

Результат виконання побудови графіків звукового коду можна побачити на рис. 3.3-3.9.

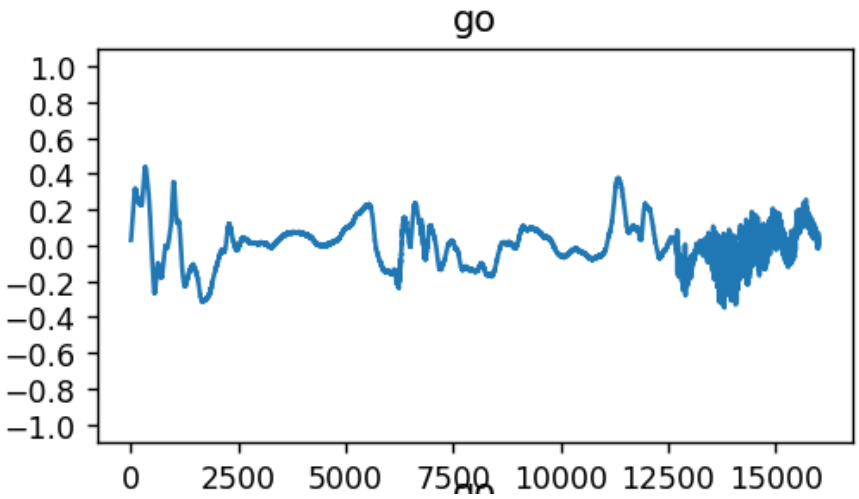


Рисунок 3.3 – Графік звукового коду для команди go, варіант 1

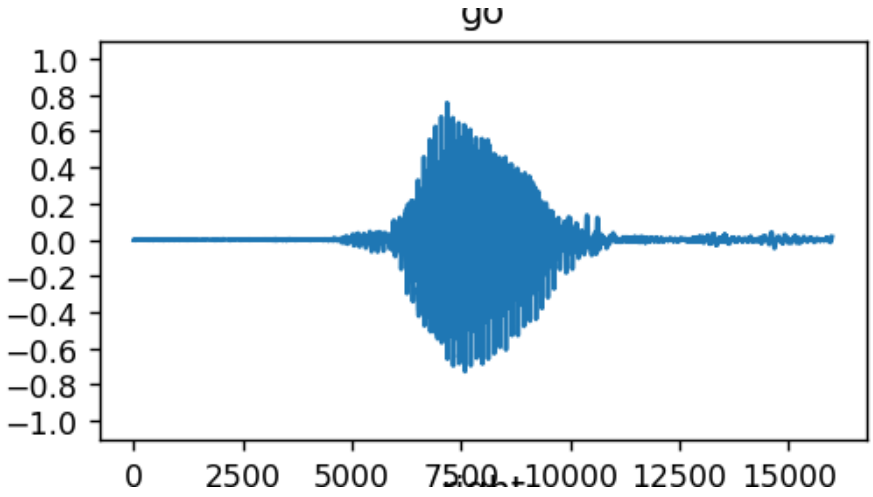


Рисунок 3.4 – Графік звукового коду для команди go, варіант 2

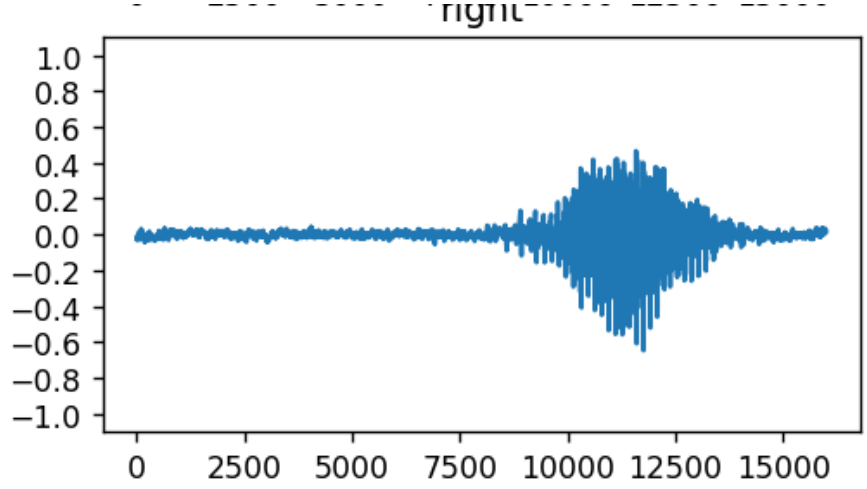


Рисунок 3.5 – Графік звукового коду для команди right

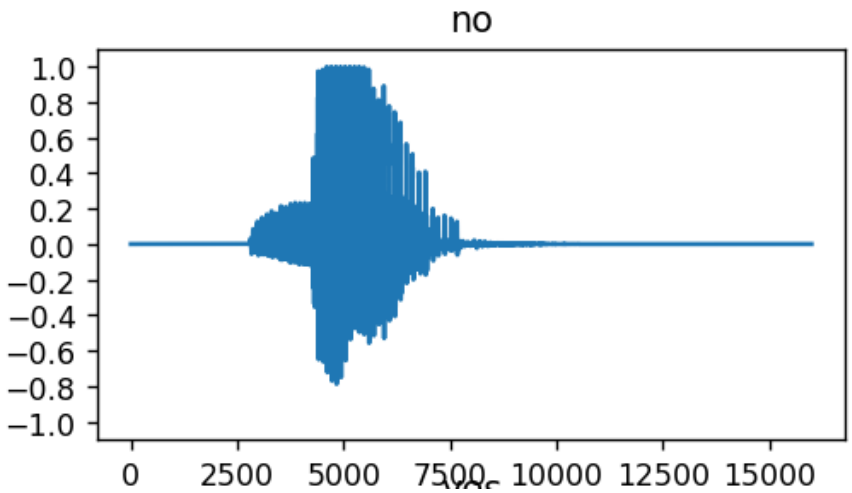


Рисунок 3.6 – Графік звукового коду для команди no, варіант 1

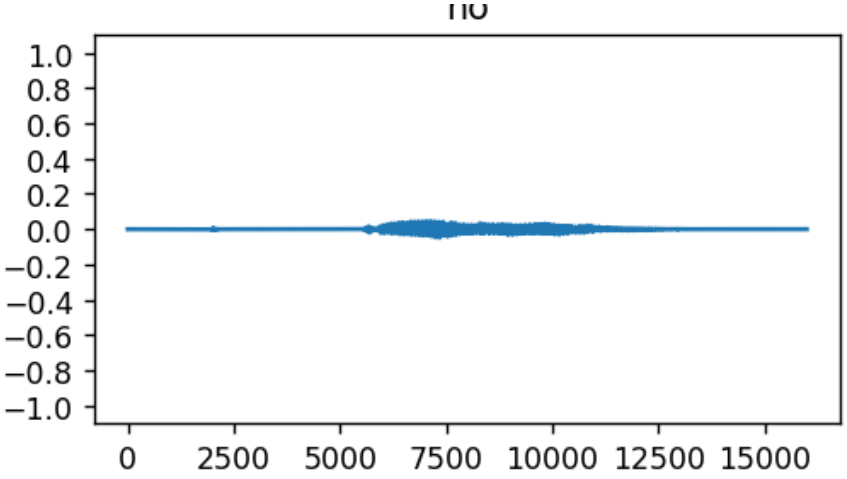


Рисунок 3.7 – Графік звукового коду для команди no, варіант 2

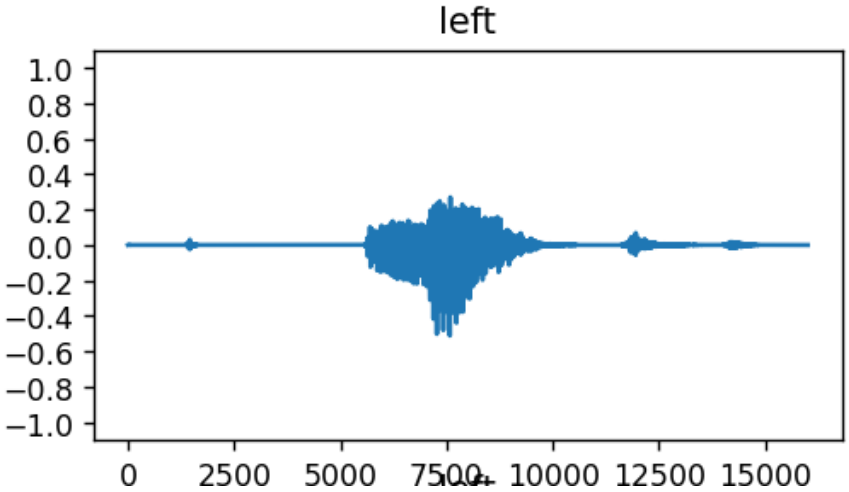


Рисунок 3.8 – Графік звукового коду для команди left

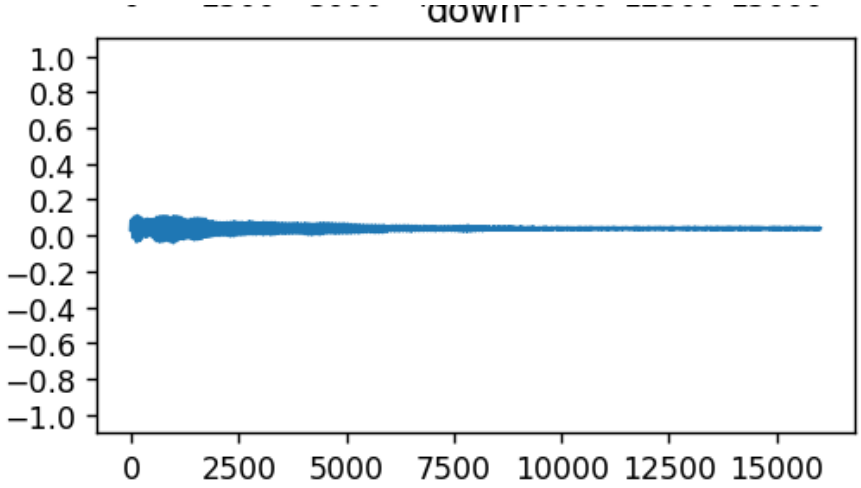


Рисунок 3.9 – Графік звукового коду для команди down

Осцилограми в наборі даних представлені в часовій області. Далі перетворимо хвилі сигналів у часовій області на сигнали у частотно-часовій області, обчисливши короткочасне перетворення Фур'є (STFT), щоб перетворити хвилі на спектрограми, які показують зміни частоти з часом і можуть бути представлені у вигляді 2D зображень. Подамо зображення спектрограми у свою нейронну мережу, щоб навчити модель.

Перетворення Фур'є (tf.signal.fft) перетворює сигнал на його складові частоти, але втрачає всю інформацію про час. Для порівняння STFT (tf.signal.stft) розбиває сигнал на вікна часу та виконує перетворення Фур'є для кожного вікна, зберігаючи деяку інформацію про час і повертаючи 2D-тензор, на якому можна виконувати стандартні згортки [19]:

```
def get_spectrogram(waveform):
  spectrogram = tf.signal.stft(
      waveform, frame_length=255, frame_step=128)
  spectrogram = tf.abs(spectrogram)
  spectrogram = spectrogram[..., tf.newaxis]
  return spectrogram.
```

Далі потрібно досліджувати дані. Надрукуємо форми тензоризованої хвилі одного прикладу та відповідну спектрограму та відтворимо оригінальний звук:

```
for i in range(3):
  label = label_names[example_labels[i]]
  waveform = example_audio[i]
  spectrogram = get_spectrogram(waveform)
  print('Label:', label)
  print('Waveform shape:', waveform.shape)
  print('Spectrogram shape:', spectrogram.shape)
  print('Audio playback')
  display.display(display.Audio(waveform, rate=16000)).
```

Визначимо функцію для відображення спектрограми:

```
def plot_spectrogram(spectrogram, ax):
  if len(spectrogram.shape) > 2:
    assert len(spectrogram.shape) == 3
    spectrogram = np.squeeze(spectrogram, axis=-1)
  log_spec = np.log(spectrogram.T + np.finfo(float).eps)
  height = log_spec.shape[0]
  width = log_spec.shape[1]
  X = np.linspace(0, np.size(spectrogram), num=width, dtype=int)
  Y = range(height)
  ax.pcolormesh(X, Y, log_spec).
```

Побудуємо осцилограму сигналу для слова left (рис. 3.10) в часі та відповідну спектрограму (частоти від часу):

```
fig, axes = plt.subplots(2, figsize=(12, 8))
timescale = np.arange(waveform.shape[0])
axes[0].plot(timescale, waveform.numpy())
axes[0].set_title('Waveform')
axes[0].set_xlim([0, 16000])
plot_spectrogram(spectrogram.numpy(), axes[1])
axes[1].set_title('Spectrogram')
plt.suptitle(label.title())
plt.show().
```

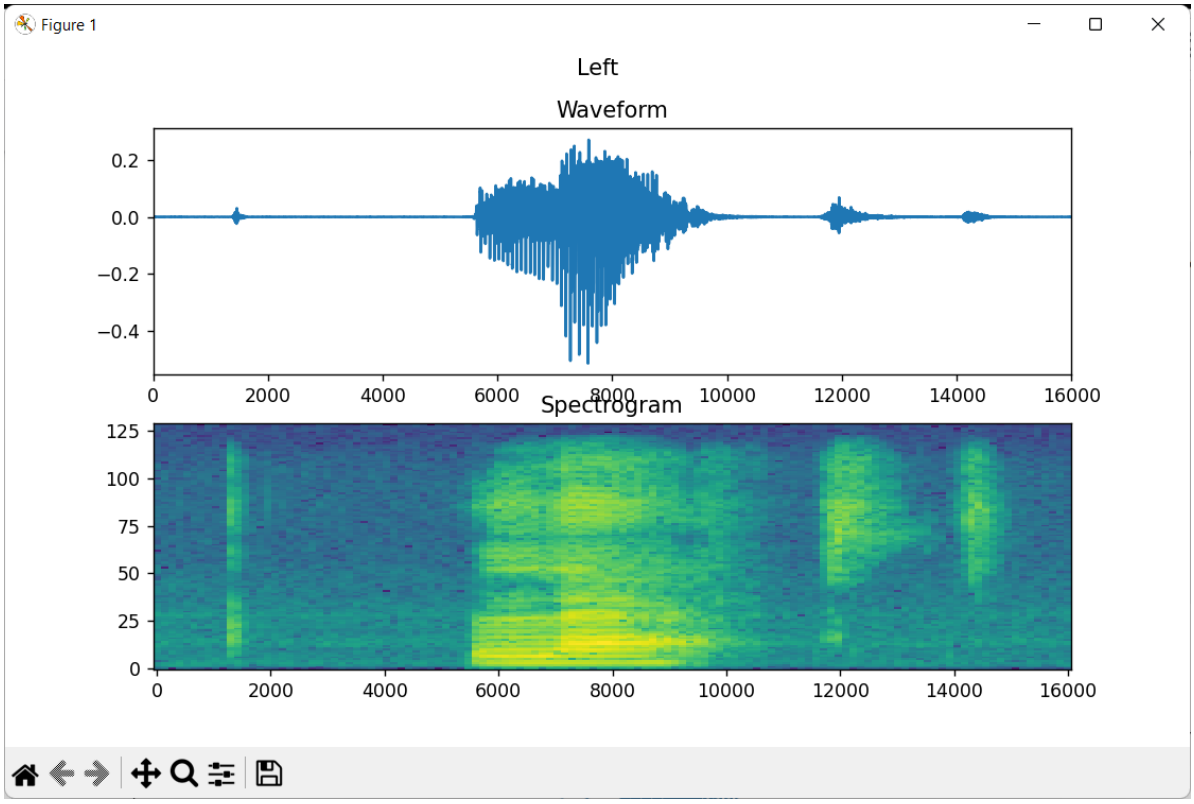


Рисунок 3.10 – Осцилограма та Спектрограма сигналу для слова left

Створимо набори даних спектрограми з наборів аудіо даних:

```
def make_spec_ds(ds):
  return ds.map(
```

```
        map_func=lambda audio,label: (get_spectrogram(audio), label),
        num_parallel_calls=tf.data.AUTOTUNE)
    train_spectrogram_ds = make_spec_ds(train_ds)
    val_spectrogram_ds = make_spec_ds(val_ds)
    test_spectrogram_ds = make_spec_ds(test_ds).
```

Переглянемо спектрограми для різних прикладів набору даних, що представлені на рис. 3.11-3.19:

```
    rows = 3
    cols = 3
    n = rows*cols
    fig, axes = plt.subplots(rows, cols, figsize=(16, 9))
    for i in range(n):
       r = i // cols
       c = i % cols
       ax = axes[r][c]
       plot_spectrogram(example_spectrograms[i].numpy(), ax)
       ax.set_title(commands[example_spect_labels[i].numpy()])
    plt.show().
```

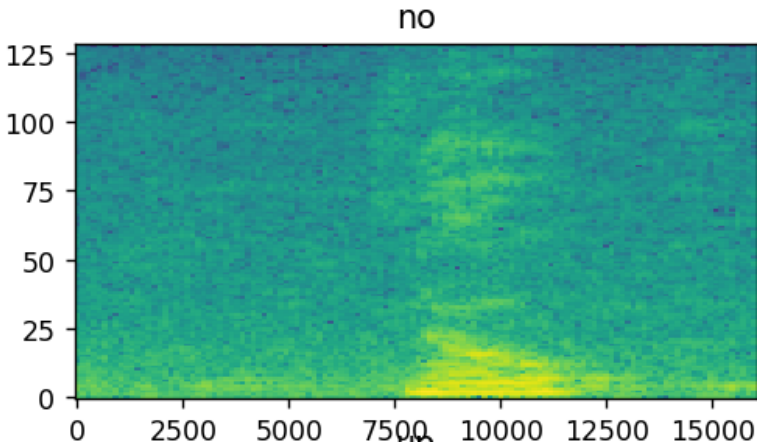


Рисунок 3.11 – Спектрограма сигналу для слова no

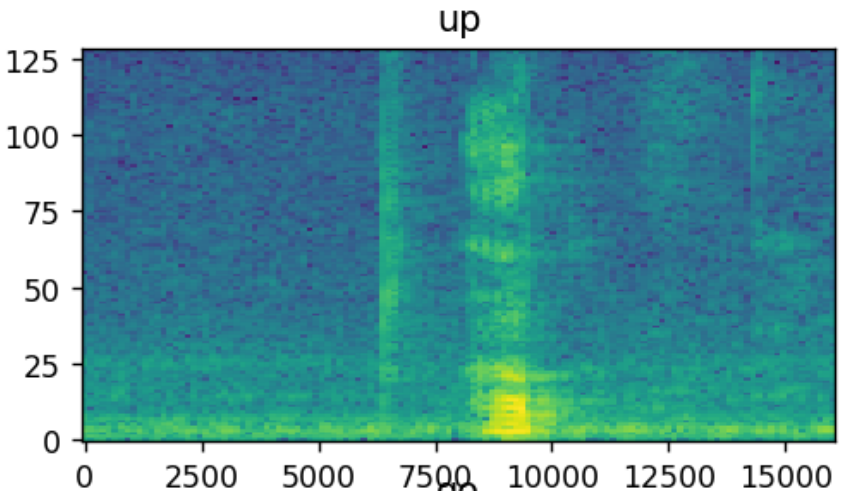


Рисунок 3.12 – Спектрограма сигналу для слова up

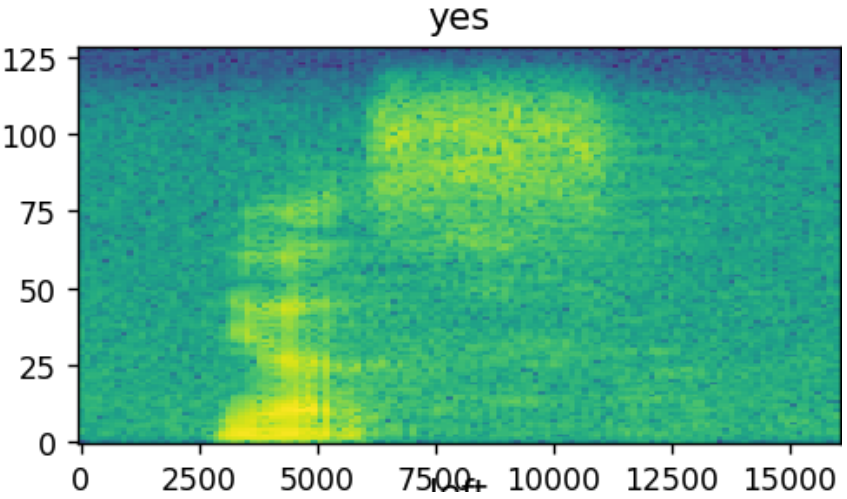


Рисунок 3.13 – Спектрограма сигналу для слова yes

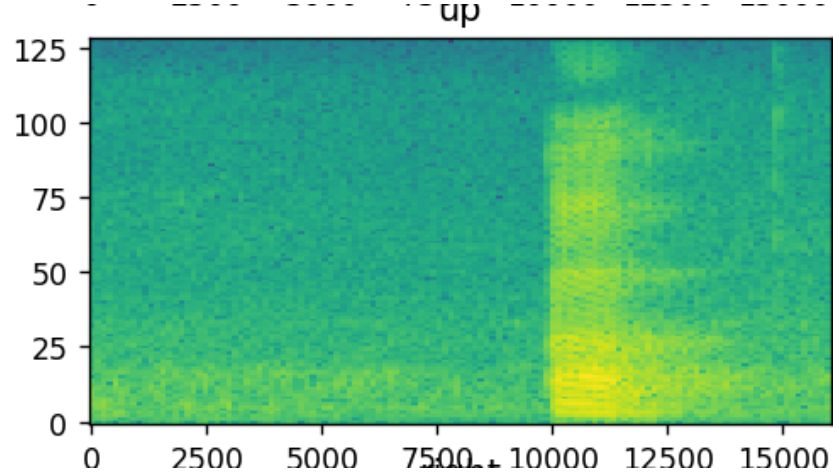


Рисунок 3.14 – Спектрограма сигналу для слова up

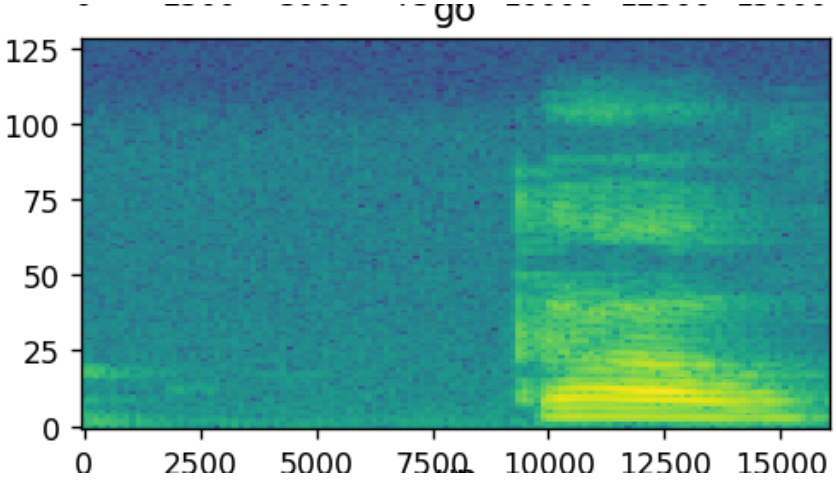


Рисунок 3.15 – Спектрограма сигналу для слова go

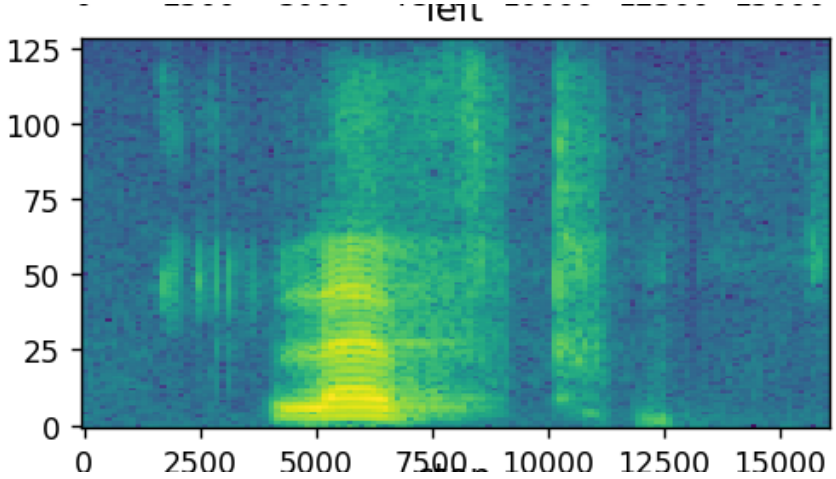


Рисунок 3.16 – Спектрограма сигналу для слова left

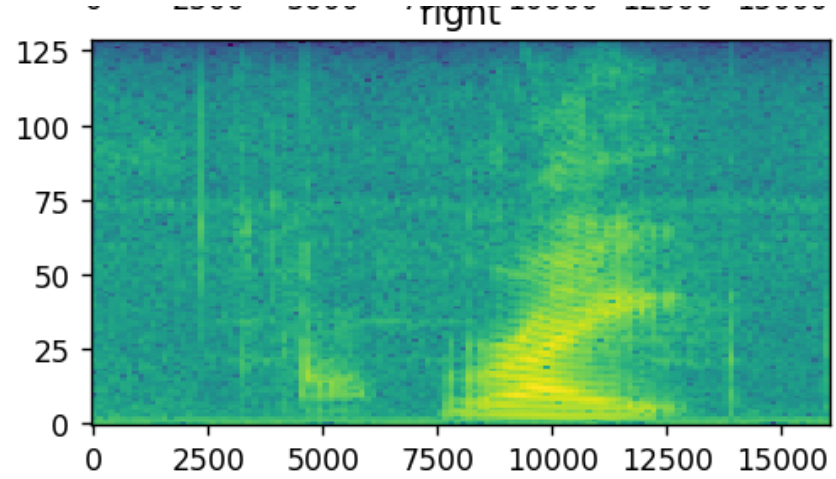


Рисунок 3.17 – Спектрограма сигналу для слова right

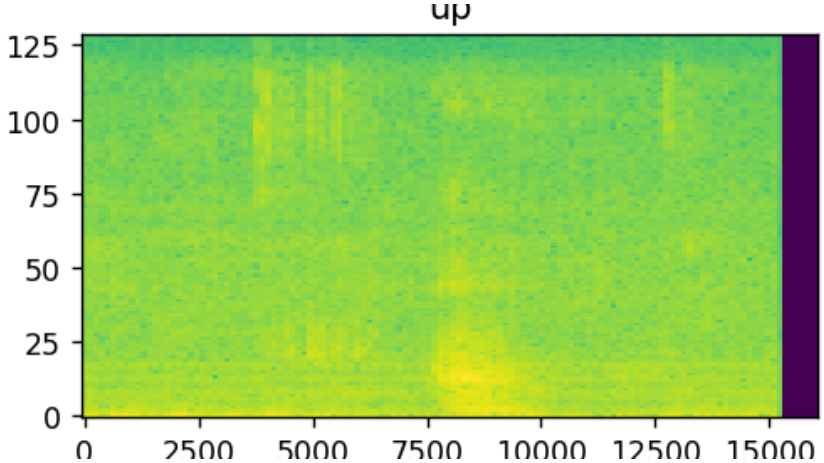


Рисунок 3.18 – Спектрограма сигналу для слова up

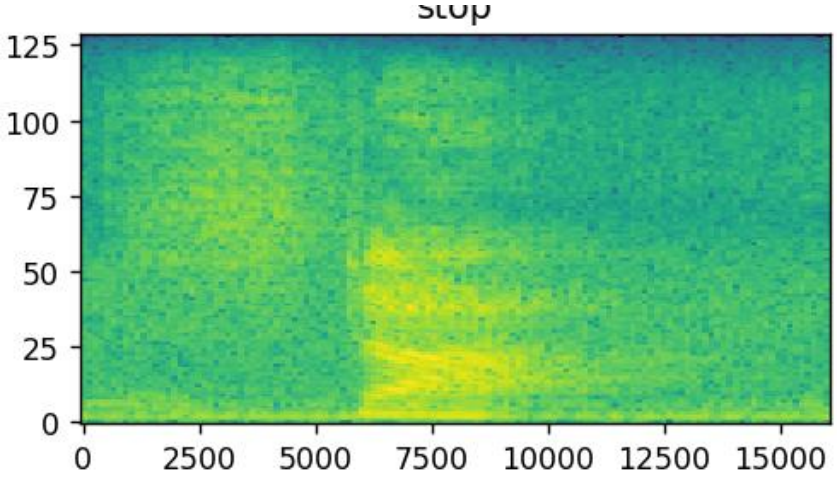


Рисунок 3.19 – Спектрограма сигналу для слова stop

Додамо операції Dataset.cache і Dataset.prefetch, щоб зменшити затримку читання під час навчання моделі:

train_spectrogram_ds = train_spectrogram_ds.cache().shuffle(10000).prefetch(tf.data.AUTOTUNE)

val_spectrogram_ds = val_spectrogram_ds.cache().prefetch(tf.data.AUTOTUNE)

test_spectrogram_ds = test_spectrogram_ds.cache().prefetch(tf.data.AUTOTUNE).

Для моделі використано PHM, оскільки було перетворено аудіофайли на зображення спектрограм на рис. 3.20-3.21:

```
input_shape = example_spectrograms.shape[1:]
print('Input shape:', input_shape)
num_labels = len(commands)
norm_layer = layers.Normalization()
norm_layer.adapt(data=train_spectrogram_ds.map(map_func=lambda spec, label: spec))
model = models.Sequential([
    layers.Input(shape=input_shape),
    layers.Resizing(32, 32),
    norm_layer,
    layers.Conv2D(32, 3, activation='relu'),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),
    layers.Flatten(), layers.Dense(128, activation='relu'),
    layers.Dropout(0.5), layers.Dense(num_labels),
])
model.summary().
```

Налаштуємо модель Keras за допомогою оптимізатора Adam і крос-ентропійних втрат [20]:

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'], ).
```

```
Input shape: (124, 129, 1)
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing (Resizing)         (None, 32, 32, 1)         0

 normalization (Normalizatio  (None, 32, 32, 1)        3
 n)

 conv2d (Conv2D)             (None, 30, 30, 32)        320

 conv2d_1 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d (MaxPooling2D  (None, 14, 14, 64)       0
 )

 dropout (Dropout)           (None, 14, 14, 64)        0
```

Рисунок 3.20 – Налаштування мережі, частина 1

```
 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 128)               1605760

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 8)                 1032
```

Рисунок 3.21 – Налаштування мережі, частина 2

Тренуймо модель протягом 10 епох для демонстрації (рис. 3.22):

```
EPOCHS = 10
history = model.fit(
    train_spectrogram_ds,
    validation_data=val_spectrogram_ds,
    epochs=EPOCHS,
    callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=2),).
```

```
Total params: 1,625,611
Trainable params: 1,625,608
Non-trainable params: 3
_________________________________________________________________
Epoch 1/10
100/100 [==============================] - 11s 94ms/step - loss: 1.7682 - accuracy: 0.3653 - val_loss: 1.3279 - val_accuracy: 0.5911
Epoch 2/10
100/100 [==============================] - 9s 86ms/step - loss: 1.1913 - accuracy: 0.5905 - val_loss: 0.9345 - val_accuracy: 0.7214
Epoch 3/10
100/100 [==============================] - 9s 87ms/step - loss: 0.8941 - accuracy: 0.6894 - val_loss: 0.7236 - val_accuracy: 0.7839
Epoch 4/10
100/100 [==============================] - 9s 89ms/step - loss: 0.7138 - accuracy: 0.7522 - val_loss: 0.6402 - val_accuracy: 0.8073
Epoch 5/10
100/100 [==============================] - 9s 86ms/step - loss: 0.5937 - accuracy: 0.7900 - val_loss: 0.5549 - val_accuracy: 0.8190
Epoch 6/10
100/100 [==============================] - 9s 87ms/step - loss: 0.5209 - accuracy: 0.8122 - val_loss: 0.5432 - val_accuracy: 0.8255
Epoch 7/10
100/100 [==============================] - 9s 88ms/step - loss: 0.4680 - accuracy: 0.8297 - val_loss: 0.5190 - val_accuracy: 0.8411
Epoch 8/10
100/100 [==============================] - 9s 90ms/step - loss: 0.4127 - accuracy: 0.8530 - val_loss: 0.5254 - val_accuracy: 0.8333
Epoch 9/10
100/100 [==============================] - 9s 87ms/step - loss: 0.4039 - accuracy: 0.8542 - val_loss: 0.5446 - val_accuracy: 0.8372
Epoch 9: early stopping
13/13 [==============================] - 1s 46ms/step - loss: 0.5347 - accuracy: 0.8353
13/13 [==============================] - 0s 19ms/step
```

Рисунок 3.22 – Навчання нейронної мережі

Побудуємо криві втрат під час навчання та перевірки, щоб перевірити, як модель покращилася під час навчання (рис. 3.23).

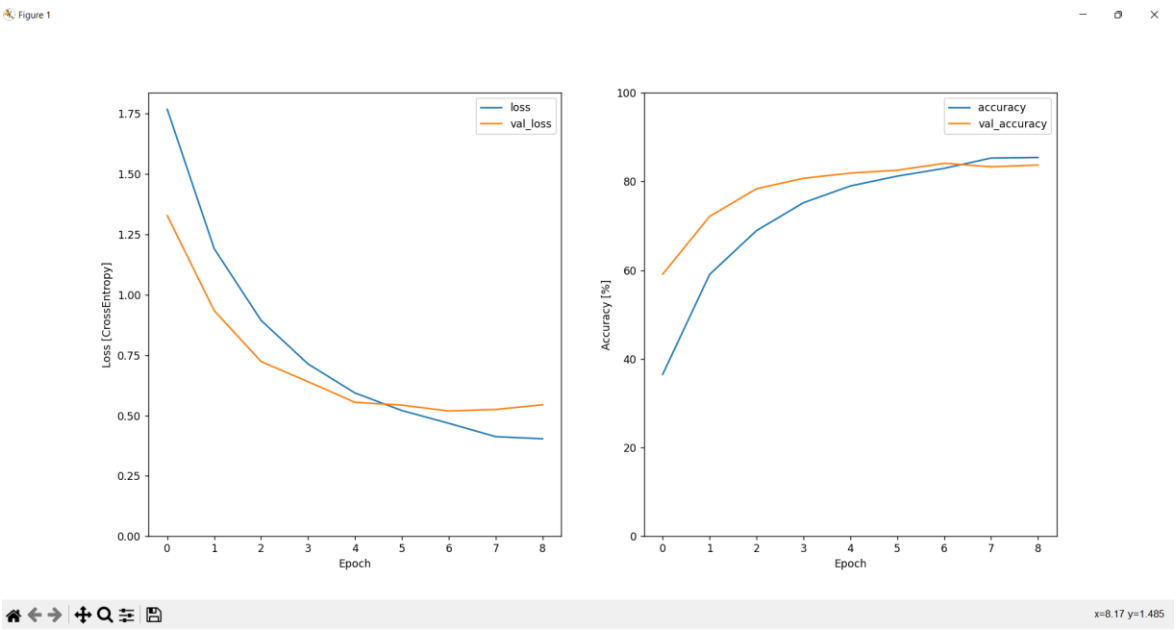


Рисунок 3.23 – Криві втрат під час навчання та перевірки

```
metrics = history.history
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
```

```
plt.plot(history.epoch, metrics['loss'], metrics['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.ylim([0, max(plt.ylim())])
plt.xlabel('Epoch')
plt.ylabel('Loss [CrossEntropy]')
plt.subplot(1,2,2)
plt.plot(history.epoch,
100*np.array(metrics['accuracy']), 100*np.array(metrics['val_accuracy']))
plt.legend(['accuracy', 'val_accuracy'])
plt.ylim([0, 100])
plt.xlabel('Epoch')
plt.ylabel('Accuracy [%]').
```

де accuracy – точність навчання;

loss – втрати;

Epoch – епохи;

val_accuracy – точність під час перевірки;

val_loss – втрати під час перевірки.

Запускаємо модель на тестовому наборі та перевіяємо продуктивність моделі:

```
model.evaluate(test_spectrogram_ds, return_dict=True).
```

Використаємо матрицю помилок, щоб перевірити, наскільки добре модель класифікувала кожну з команд у тестовому наборі і отримаємо наступний графік з мітками (label) та прогнозом (prediction) (рис. 3.24):

```
y_pred = model.predict(test_spectrogram_ds)
y_pred = tf.argmax(y_pred, axis=1)
```

```
y_true = tf.concat(list(test_spectrogram_ds.map(lambda s,lab: lab)), axis=0)
confusion_mtx = tf.math.confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx,
            xticklabels=commands,
            yticklabels=commands,
            annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
```

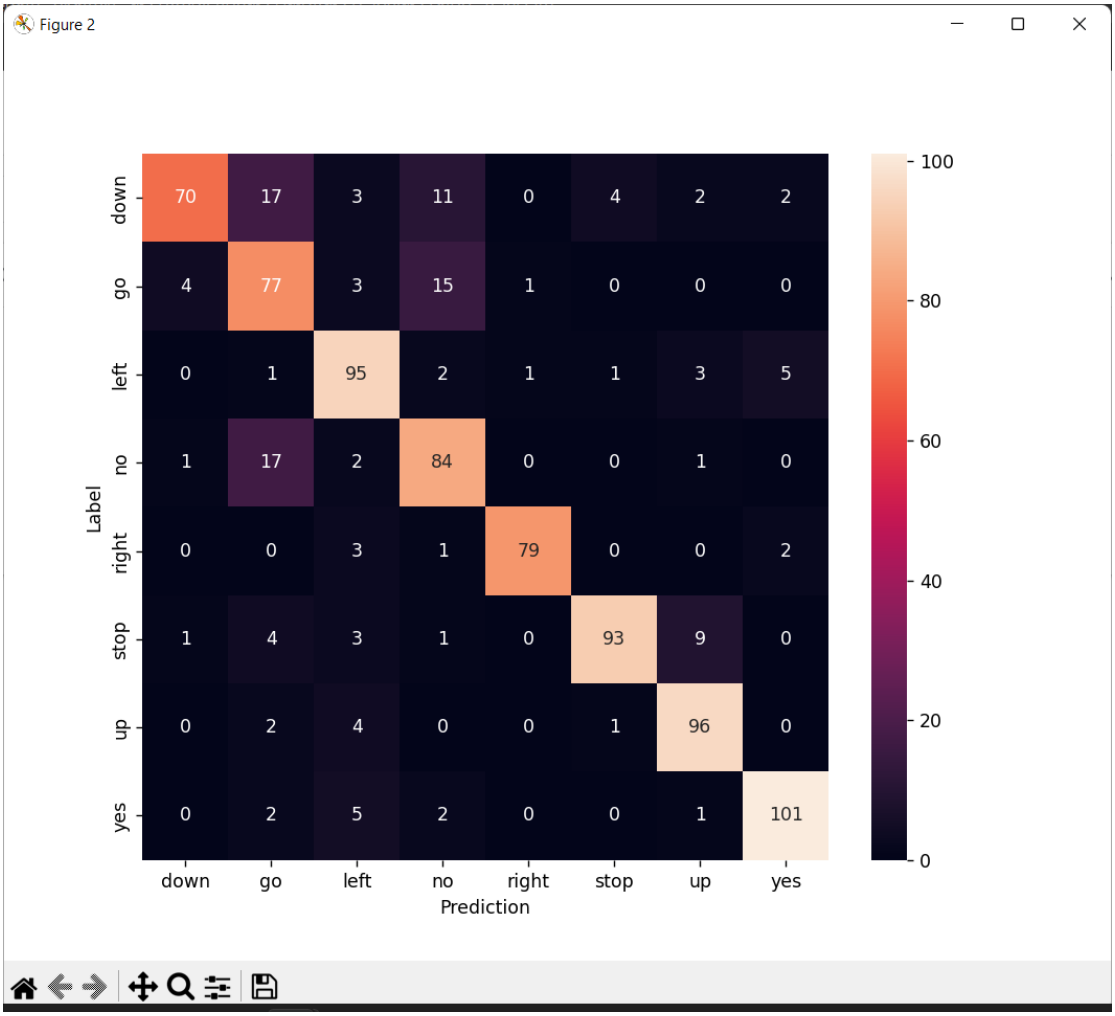


Рисунок 3.24 – Графік з мітками (label) та прогнозом (prediction) розпізнавання

3.3 Розпізнавання голосу та керування роботом

Розроблене програмне забезпечення було протестовано на платі Raspberry Pi, що може бути використана для керування МР [21]. На жаль, реальної моделі робота не було, але запропоновано команди керування ним.

Для цього потрібно підключити бібліотеку RPi.GPIO:

```
import RPi.GPIO as GPIO
import time
```

Далі підключити двигуни:

```
MOTOR1B=24
MOTOR1E=7
MOTOR2B=22
MOTOR2E=14
```

Провести налаштування двигунів:

```
GPIO.setup(MOTOR1B, GPIO.OUT)
GPIO.setup(MOTOR1E, GPIO.OUT)

GPIO.setup(MOTOR2B, GPIO.OUT)
GPIO.setup(MOTOR2E, GPIO.OUT)
```

Далі створити функції для переміщення. Першою з них є функція руху вперед:

```
def forward():
    GPIO.output(MOTOR1B, GPIO.HIGH)
```

```
    GPIO.output(MOTOR1E, GPIO.LOW)
    GPIO.output(MOTOR2B, GPIO.HIGH)
    GPIO.output(MOTOR2E, GPIO.LOW)
```

Другою є функція переміщення назад:

```
def reverse():
    GPIO.output(MOTOR1B, GPIO.LOW)
    GPIO.output(MOTOR1E, GPIO.HIGH)
    GPIO.output(MOTOR2B, GPIO.LOW)
    GPIO.output(MOTOR2E, GPIO.HIGH)
```

Наступною є функція правого повороту:

```
def rightturn():
    GPIO.output(MOTOR1B,GPIO.LOW)
    GPIO.output(MOTOR1E,GPIO.HIGH)
    GPIO.output(MOTOR2B,GPIO.HIGH)
    GPIO.output(MOTOR2E,GPIO.LOW)
```

А також лівого повороту:

```
def leftturn():
    GPIO.output(MOTOR1B,GPIO.HIGH)
    GPIO.output(MOTOR1E,GPIO.LOW)
    GPIO.output(MOTOR2B,GPIO.LOW)
    GPIO.output(MOTOR2E,GPIO.HIGH)
```

Останньою є функція зупинки руху:

```
def stop():
    GPIO.output(MOTOR1E,GPIO.LOW)
    GPIO.output(MOTOR1B,GPIO.LOW)
    GPIO.output(MOTOR2E,GPIO.LOW)
    GPIO.output(MOTOR2B,GPIO.LOW)
```

Після розпізнавання команд потрібно використовувати відповідні функції.

3.4 Висновки до третього розділу

В ході написання третього розділу було розроблено нейронну мережу для розпізнавання команд керування роботом з використанням бібліотек TensorFlow та Keras.

Розроблений додаток має високий відсоток розпізнавання.

За допомогою розпізнаних команд нейронною мережею було запропоновано основні команди для керування МР:

– рух вперед;

– рух назад;

– правий поворот;

– лівий поворот;

– зупинка руху.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ З РОЗРОБЛЕНИМ ПРОГРАМНИМ ДОДАТКОМ

4.1 Постановка задачі експерименту

Основним завданням експерименту є розпізнавання голосових команд нейронною мережею. Ці команди у вигляді аудіофайлів формату wav представлені на рис. 4.1.

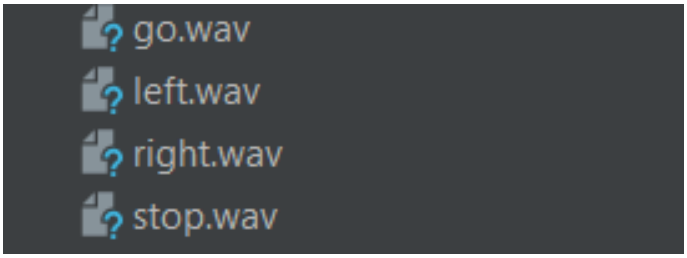


Рисунок 4.1 – Аудіофайли з командами та їх розпізнавання

Отже, потрібно зробити висновок щодо аудіофайлу, для чого використаємо наступний код:

```
x = 'шлях до файлу'
x = tf.io.read_file(str(x))
x, sample_rate = tf.audio.decode_wav(x, desired_channels=1, desired_samples=16000,)
x = tf.squeeze(x, axis=-1)
waveform = x
x = get_spectrogram(x)
x = x[tf.newaxis,...]
prediction = model(x)
plt.bar(commands, tf.nn.softmax(prediction[0]))
plt.title('Назва команди')
```

```
plt.show()
display.display(display.Audio(waveform, rate=16000))
```

В результаті цього отримаємо графіки з прогнозами розпізнавання, кривими втрат під час навчання та перевірки, а також графік з величиною помилки (чим вона менше, тим краще розпізнавання).

4.2 Експериментальні дослідження з розробленим програмним засобом

Перший експеримент було проведено з командою go. Її графіки представлено на рис. 4.2-4.4.

go

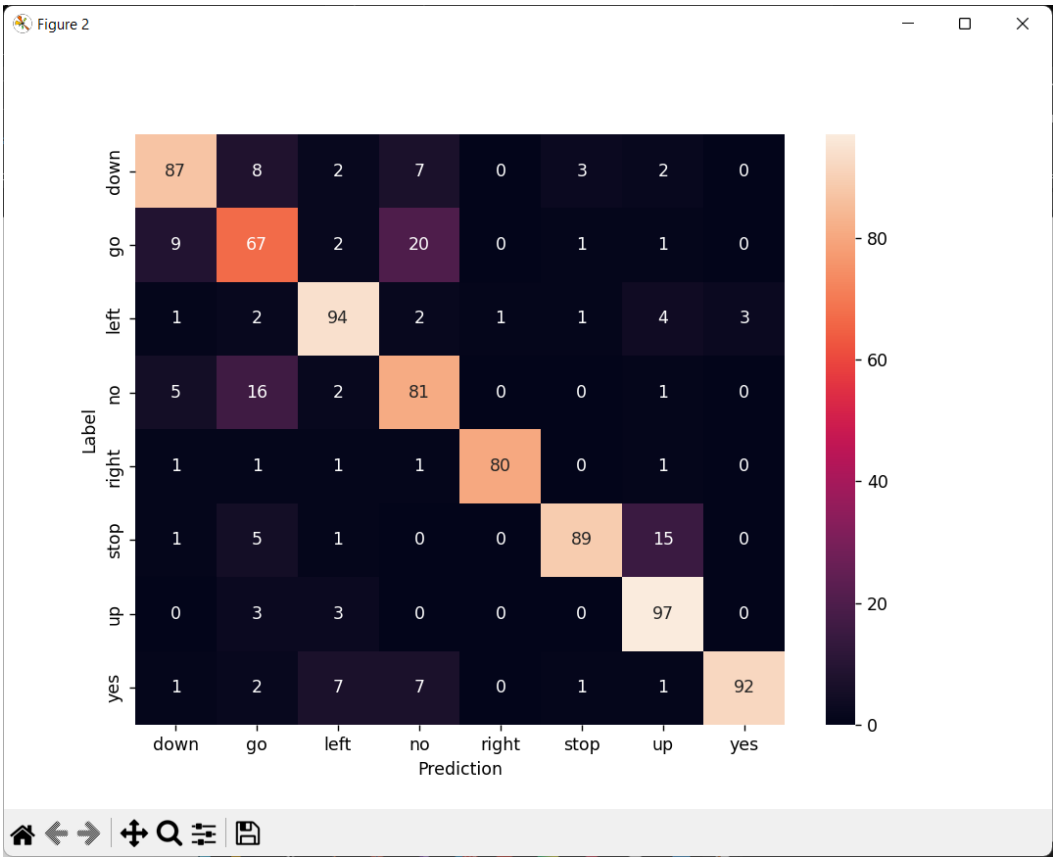


Рисунок 4.2 – Прогноз розпізнавання команди go

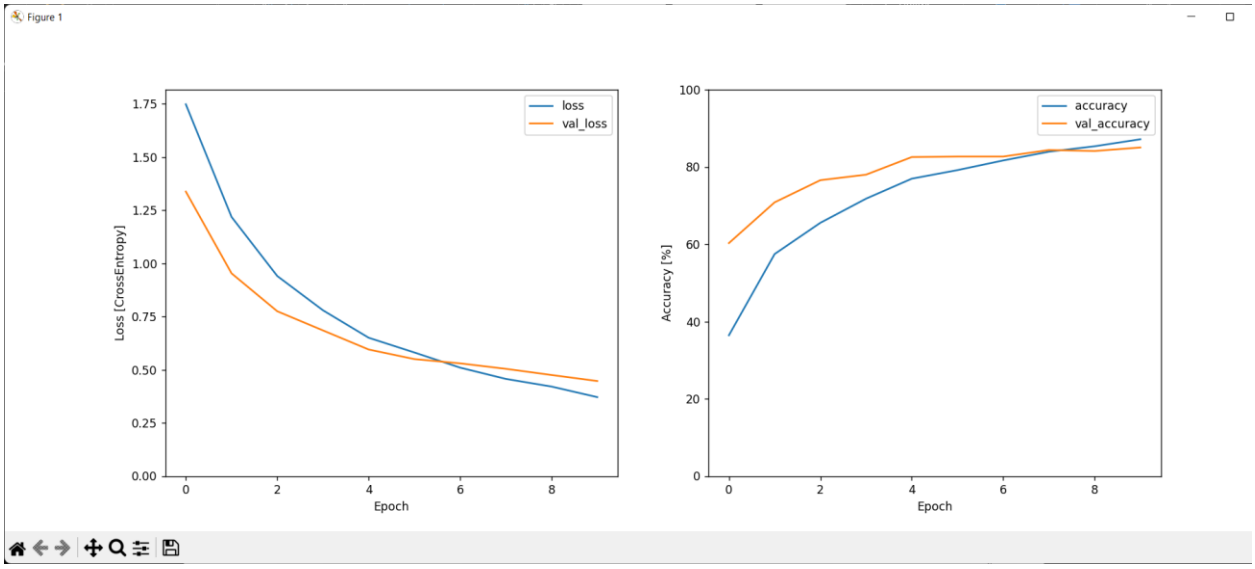


Рисунок 4.3 – Криві втрат під час навчання та перевірки команди go

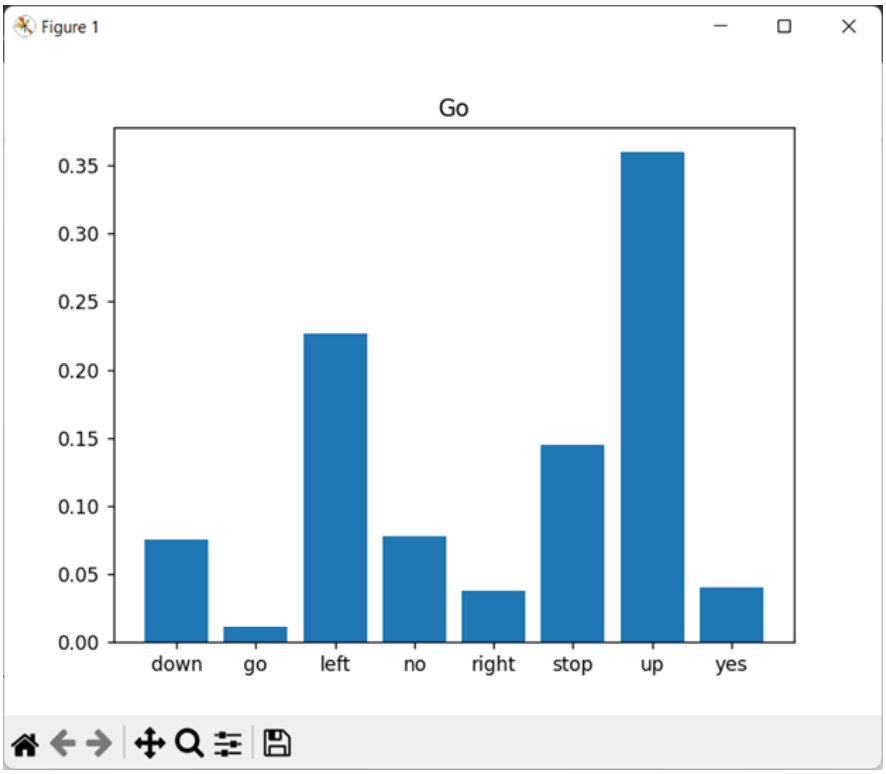


Рисунок 4.4 – Графік величини помилки команди go

Другий експеримент було зроблено для команди left. З результатами розпізнавання можна ознайомитися на рис. 4.5-4.7.

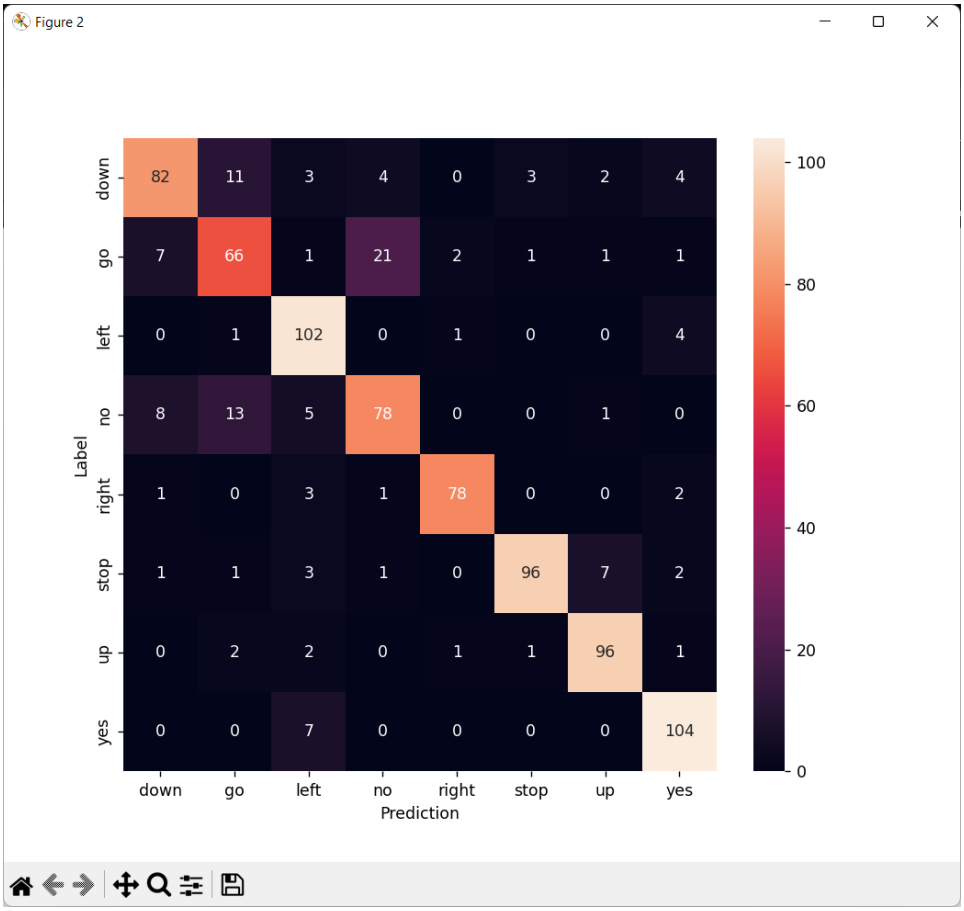


Рисунок 4.5 – Прогноз розпізнавання команди left

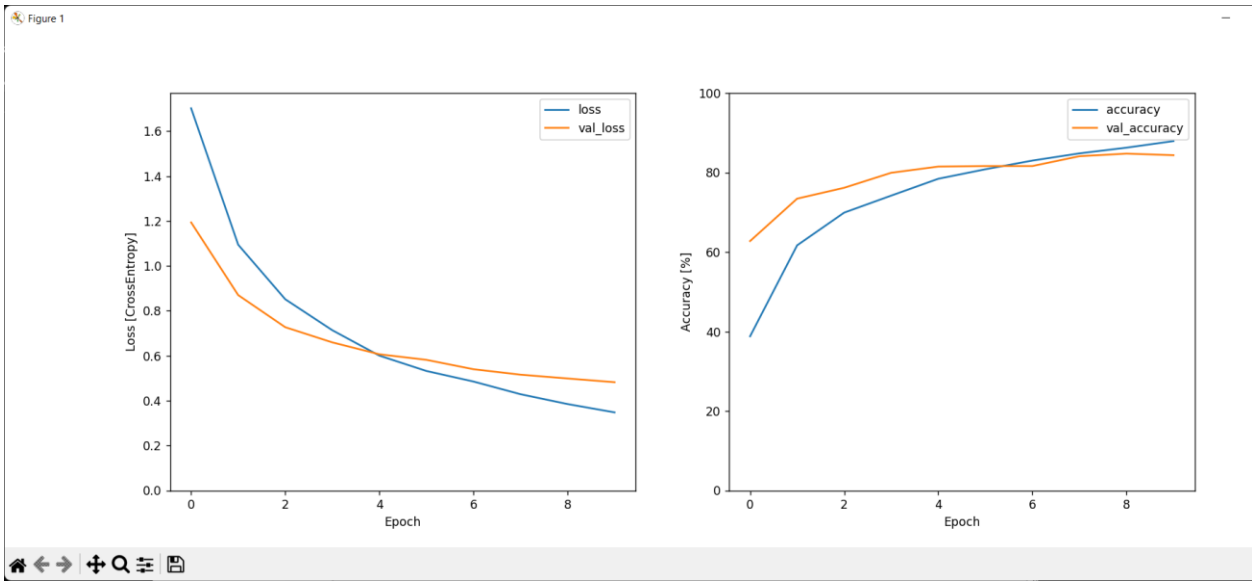


Рисунок 4.6 – Криві втрат під час навчання та перевірки команди left

Третій експеримент було зроблено для команди right. З результатами розпізнавання можна ознайомитися на рис. 4.8-4.10.

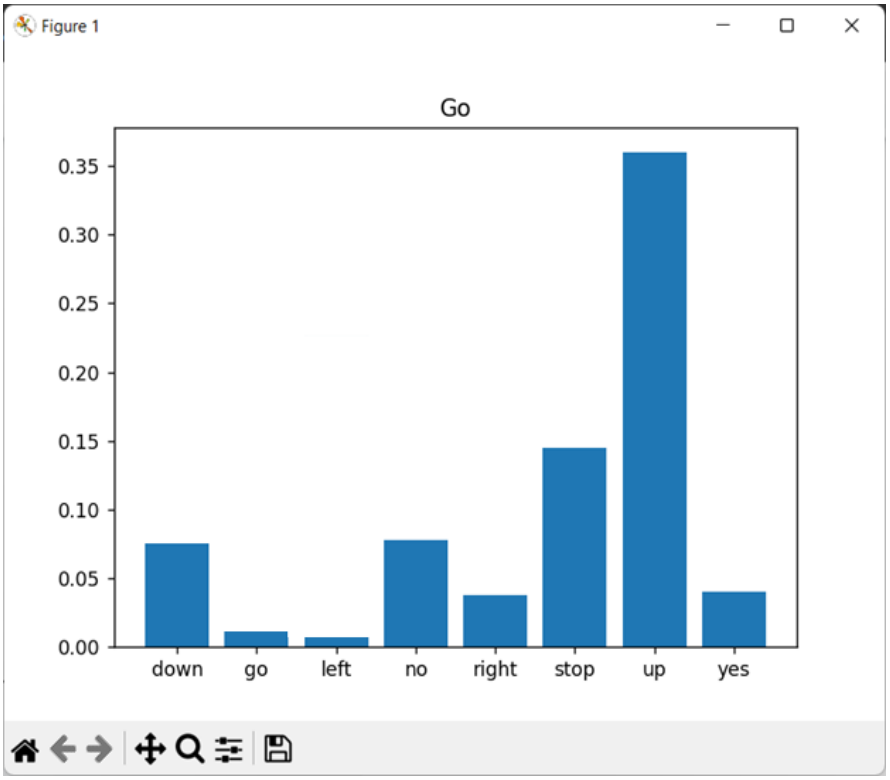


Рисунок 4.7 – Графік величини помилки команди left

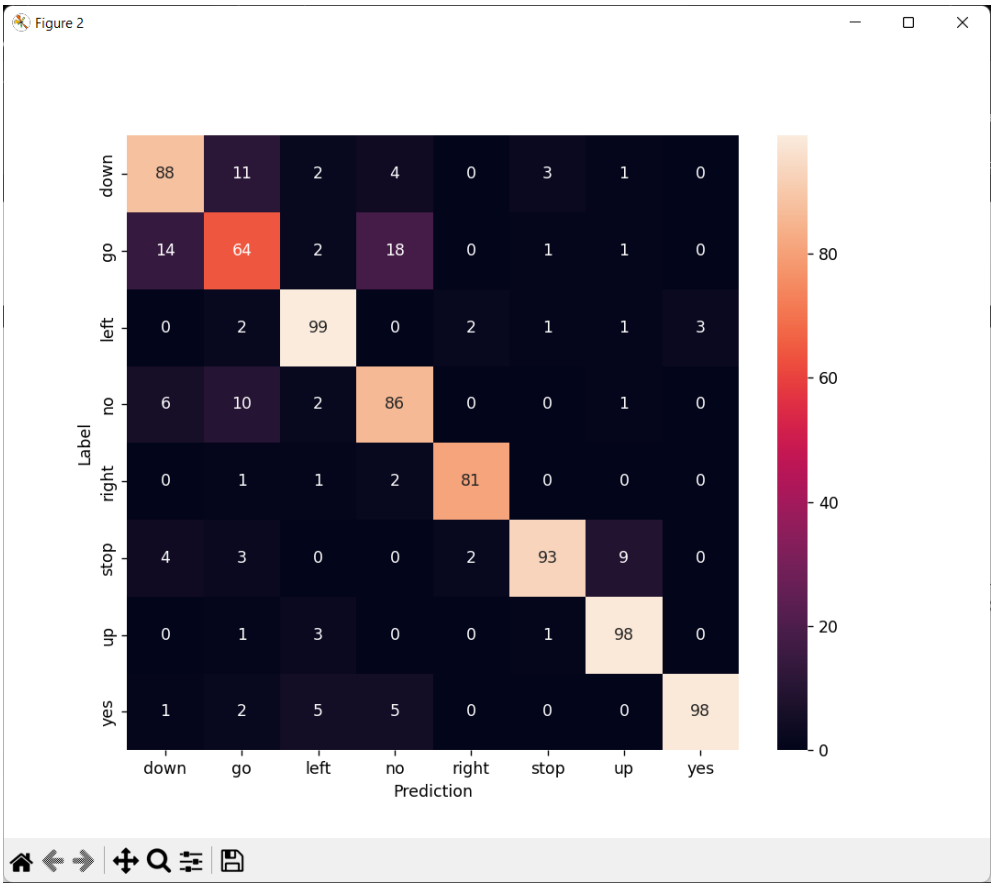


Рисунок 4.8 – Прогноз розпізнавання команди right

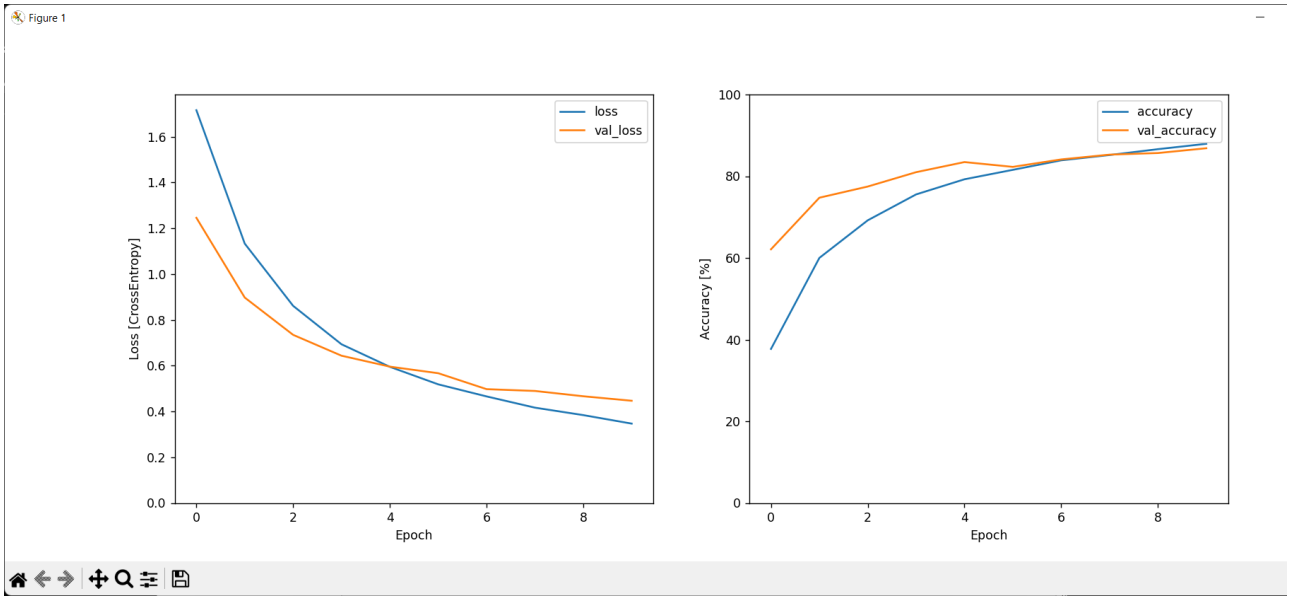


Рисунок 4.9 – Криві втрат під час навчання та перевірки команди right

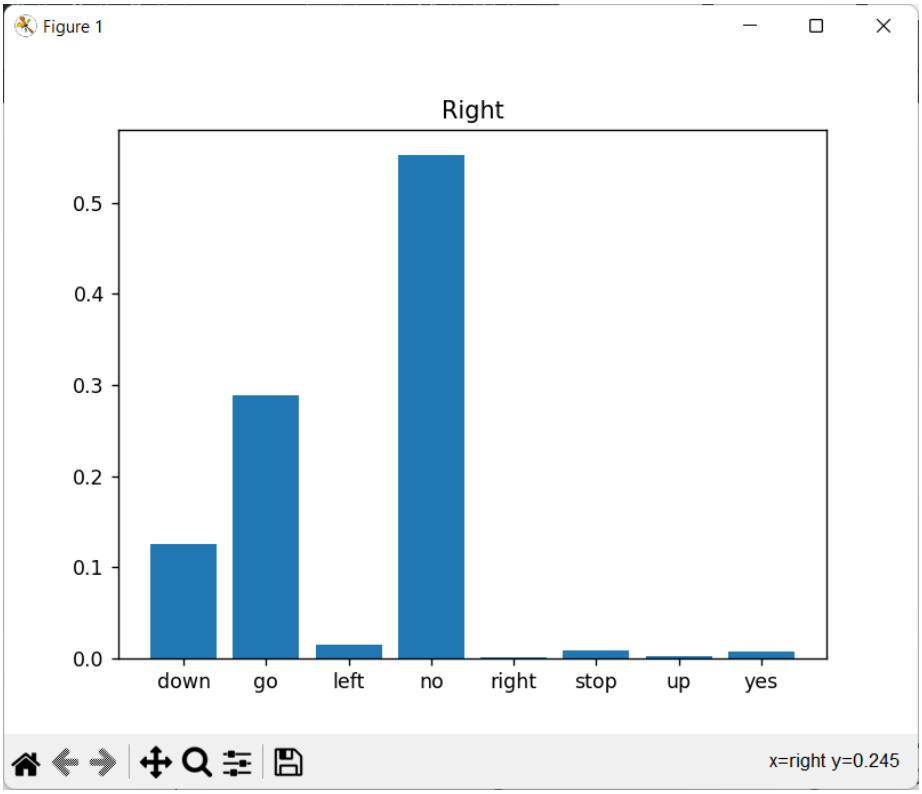


Рисунок 4.10 – Графік величини помилки команди right

Четвертий експеримент було проведено для команди stop. З результатами розпізнавання можна ознайомитися на рис. 4.11-4.13.

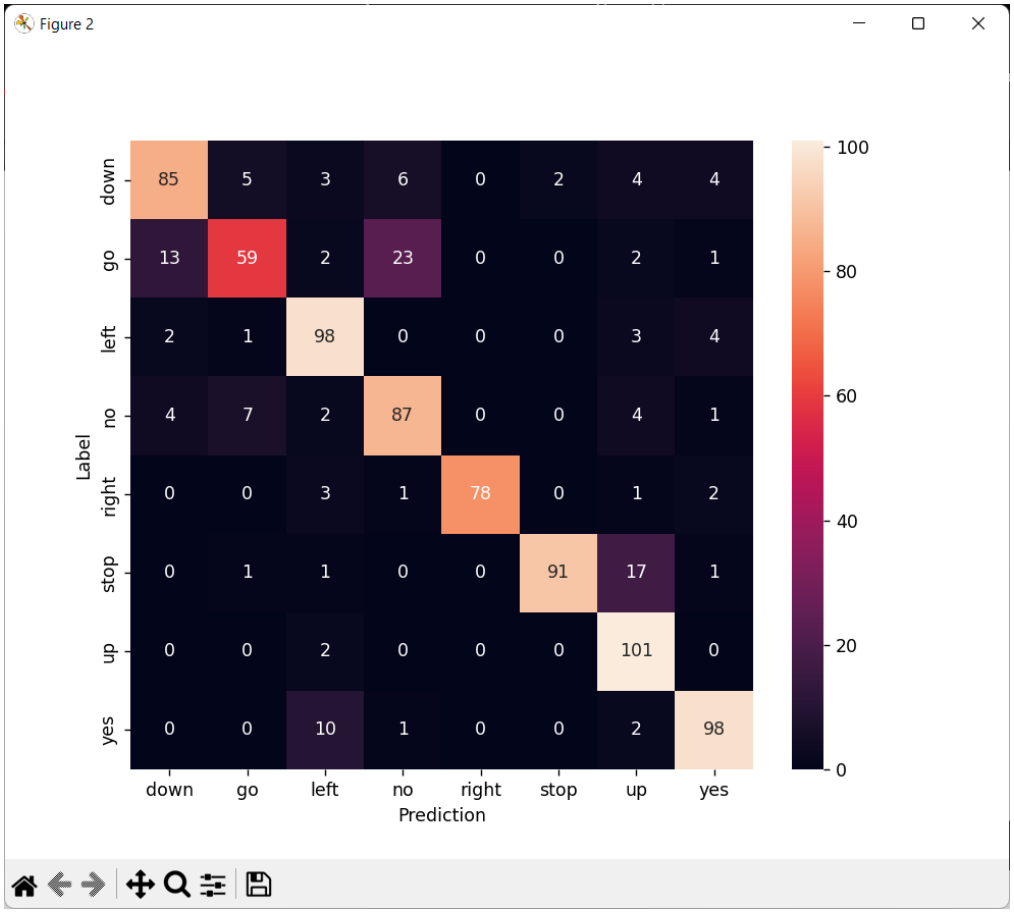


Рисунок 4.11 – Прогноз розпізнавання команди stop

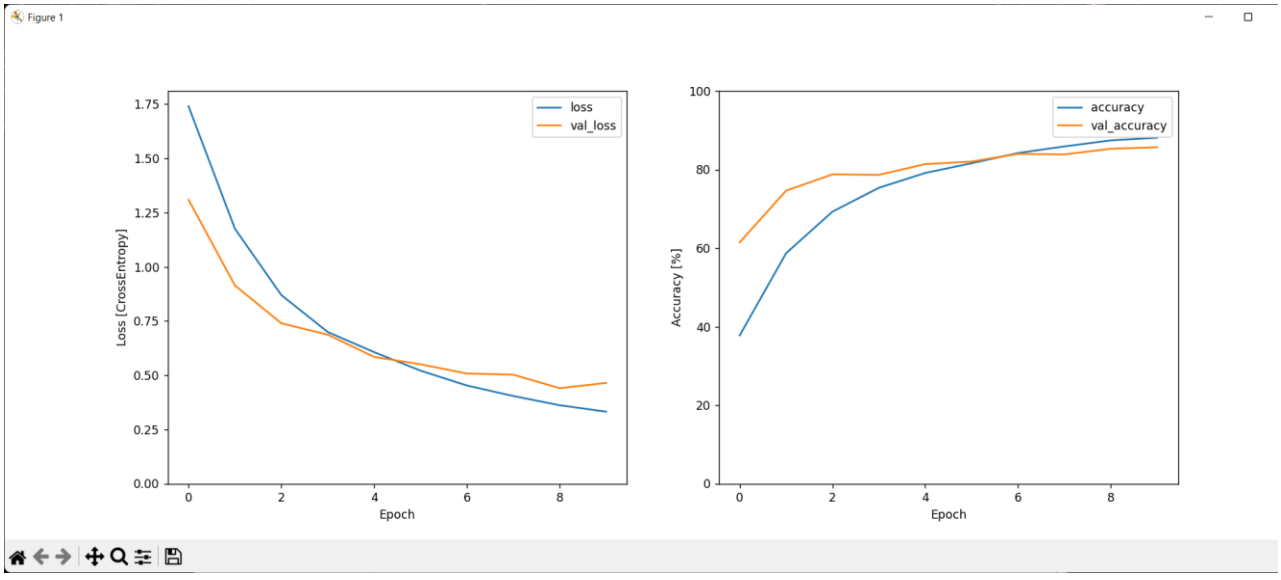


Рисунок 4.12 – Криві втрат під час навчання та перевірки команди stop

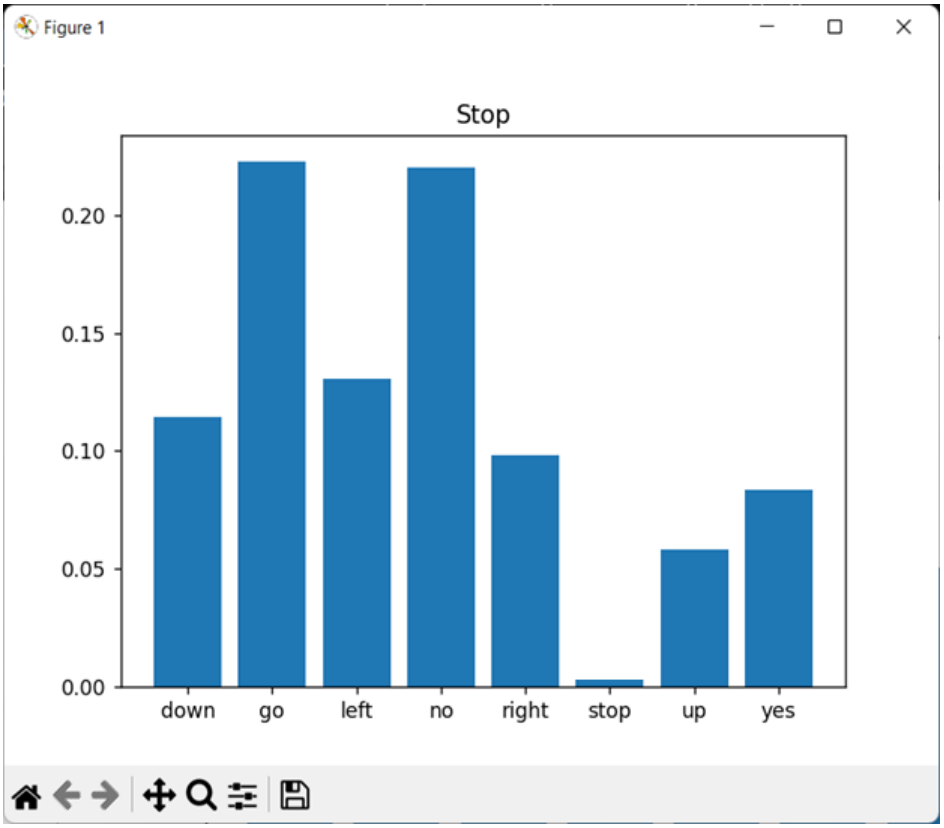


Рисунок 4.13 – Графік величини помилки команди stop

4.3 Охорона праці

Згідно з ГОСТ 12.1.013–78 ССБТ, приміщення належить до класу приміщень без підвищеної небезпеки, оскільки воно сухе, вологість повітря до 65%, струмопровідний пил відсутній, а також відсутні підвищена температура та можливість одночасного дотику людини до з'єднаних з землею металоконструкцій. до металевих корпусів електроустаткування.

Енергоживлення здійснюється від трифазної чотирипровідної мережі з глухозаземленою нейтраллю зі змінним струмом частотою 50 Гц і напругою 380/220В. У приміщенні комп'ютерного залу необхідно прокласти шину занулення, виконану відповідно до вимог НПАОП 40.1-1.32-01, яка гальванічно з'єднуватиметься із заземленою нейтраллю мережі [22].

Занулення також підлягають металеві частини електроустановок, які доступні для дотику людини і які не мають інших видів захисту, що забезпечують електробезпеку.

Відповідно до НПАОП 0.00-4.12-05 проводиться вступний інструктаж на робочому місці перед початком робіт. Програма розробляється керівником відділу адміністрування, узгоджується із службою охорони праці та затверджується керівником підприємства. Інструктаж фіксується у журналі реєстрації первинних інструктажів. Повторний іструктаж проводиться на робочому місці з усіма робітниками раз на півроку. Якщо порушення техніки безпеки призвели до травм, повторний інструктаж проводиться одразу. Програма дублює первинний інструктаж. Запис про повторний інструктаж також зазначається в журналі первинного інструктажу.

Згідно з нормами НАПБ Б.03.002-2007 за ступенем пожежо-вибухонебезпечності дане виробництво відноситься до категорії В. Відповідно до ДБН В.1.1.7-2002 будівля відноситься до 1-го ступеня вогнестійкості, (будівлі з несучими та огороджувальними конструкціями з природних або штучних кам'яних матеріалів, бетону або залізобетону із застосуванням листових та плитних негорючих матеріалів). Згідно з НПАОП 40.1-1.01-97 приміщення з пожежонебезпечності належить до класу П-IIа.

У приміщенні розміщено 3 ручні вуглекислотні вогнегасники типу ВВК-2. Їх використання обумовлене необхідністю гасіння електроустановок, які перебувають під напругою трохи більше 1000В, розташованих у приміщенні. З розрахунку один вогнегасник на 3 ПК, але в одному приміщенні їх має бути не менше ніж два (НАПБ Б.03.001-2004) [23].

4.4 Висновки до четвертого розділу

В ході проведення експериментального дослідження з розробленим програмним засобом розпізнавання за допомогою нейронної мережі було виявлено високий рівень розпізнавання слів, записаних з мікрофону, що представлені на графіках величин помилок відповідних команд – чим менше, тим краще.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проаналізовано сучасні методи управління МР на основі систем програмного, адаптивного та інтелектуального управління. Було проведено аналіз методів голосового управління МР та розглянуто приклад на основі когнітивної моделі FCAS.

Далі було проведено детальний аналіз таких методів ідентифікації голосових команд, як приховані марковські моделі, динамічна деформація часу та штучні нейронні мережі. Також було досліджено топології нейронних мереж для застосування в системах управління.

Після проведення усіх необхідних досліджень було виконано наступні завдання:

– обрано тип нейронної мережі;

– розроблено топологію нейронної мережі;

– виконано математичний розрахунок елементів нейронної мережі;

– розроблено алгоритму роботи системи;

– розроблено програмний продукт системи;

– проведено експериментальні дослідження.

У результаті була досягнута основна мета роботи – розроблено систему ідентифікацій команд для керування МР на базі нейронних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. – Введ. 2015-06-22. – К. Держстандарт України, 2017. – 29 с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2021. 55 с.

3. Пилипенко В. М. Дослідження методів розпізнавання голосу / В. М. Пилипенко // збірник студентських наукових статей «Автоматизація та приладобудування» ADED-2021 (Випуск 2) [електронне видання] / Пилипенко В. М. – Харків, 2021. – С. 183–187 (дата звернення: 23.10.2022).

4. Лисенко О. С. Навчальний комплекс для дослідження інтелектуальних систем керування автономними роботами. – 2019. – Режим доступу до ресурсу: http://ir.stu.cn.ua/handle/123456789/19520 (дата звернення: 18.09.2022).

5. Zhi, Li, and Mei Xuesong. "Navigation and control system of mobile robot based on ROS." 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). IEEE, 2018 – Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/8577901 (дата звернення: 18.09.2022).

6. Tzafestas, Spyros G. "Mobile robot control and navigation: A global overview." Journal of Intelligent & Robotic Systems 91.1 (2018): 35-58 – Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s10846-018-0805-9 (дата звернення: 18.09.2022).

7. Delic, Vlado, et al. "Speech technology progress based on new machine learning paradigm." Computational intelligence and neuroscience (2019) – Режим доступу до ресурсу: https://psycnet.apa.org/record/2019-38024-001 (дата звернення: 20.09.2022).

8. Прихована марковська модель – Вікіпедія. [Електронне джерело] // wikipedia.org – Режим доступу до ресурсу: http://uk.wikipedia.org/wiki/ Прихована _ марковська_модель (дата звернення: 20.09.2022).

9. Fedorov, A. A., et al. "Development of a control system for a mobile robot based on Markov processes." IOP Conference Series: Materials Science and Engineering. Vol. 1129. No. 1. IOP Publishing, 2021 – Режим доступу до ресурсу: https://iopscience.iop.org/article/10.1088/1757-899X/1129/1/012057/pdf (дата звернення: 20.09.2022).

10. A. Krug, R. Knaebel and S. Stober, "Neuron activation profiles for interpreting convolutional speech recognition models", Proc. NeurIPS Workshop Interpretability Robustness Audio Speech Lang., pp. 1-13, 2018 – Режим доступу до ресурсу: https://openreview.net/pdf?id=Bylpgfjen7 (дата звернення: 22.09.2022).

11. Song, Zhaojuan. "English speech recognition based on deep learning with multiple features." Computing 102.3 (2020): 663-682. – Режим доступу до ресурсу: https://arxiv.org/pdf/1508.01211.pdf (дата звернення: 27.09.2022).

12. An All-Neural On-Device Speech Recognizer [Електронне джерело]// ai.googleblog.com – Режим доступу до ресурсу: https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html (дата звернення: 02.10.2022).

13. Озеранець, О. П. "Пристрій розпізнавання голосової інформації з використанням глибинних нейронних мереж" Матеріали VIII науково-технічної конференції "Інформаційні моделі, системи та технології" (2020): 49-49. – Режим доступу до ресурсу: http://elartu.tntu.edu.ua/bitstream/lib/34379/2/VIII_NTK_2020_Ozeranets_O_P-Voice_information_recognition_49.pdf (дата звернення: 13.10.2022).

14. Мартинюк, Володимир Любомирович. Побудова нейронної мережі для розпізнавання мови. – Режим доступу до ресурсу: http://elartu.tntu.edu.ua/bitstream/lib/38283/1/2022_KRB_SNs-42_Martyniuk_VL_v1.1.pdf (дата звернення: 13.10.2022).

15. Ткаченко, Костянтин, Владислав Брусєнцев. Використання нейронних мереж під час розпізнавання голосових команд. Цифрова платформа: інформаційні технології в соціокультурній сфері 5.1 (2022): 130-143. – Режим доступу до ресурсу: http://infotech-soccult.knukim.edu.ua/article/view/261297 (дата звернення: 23.10.2022).

16. Федоренко М. О., Дудник О. В. Використання рекурентної нейронної мережі для ідентифікawії параметрів об'єкту 2-го та 3-го порядку //автоматизація, електроніка, інформаційно-вимірювальні технології: освіта, наука, практика. – 2022. – С. 21. – Режим доступу до ресурсу: http://repository.kpi.kharkov.ua/bitstream/KhPIPress/60123/1/Conference_NTU_KhPI_2022_AEIVT.pdf (дата звернення: 25.11.2022).

17. Мережа Н., Ергографом К. З. УДК004. 032.26 //Перспективні напрямки інформаційних і комп‘ютерних систем та мереж, комп‘ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті: Матеріали всеукраїнської науково-практичної інтернет конференції.-Кропивницький: ЦНТУ, 2019.–162 с. – 2019. – С. 51. – Режим доступу до ресурсу: http://www.kntu.kr.ua/doc/science/zahody/zdob/2019/26.pdf (дата звернення: 27.11.2022).

18. Шуліка Я. П., Мелешко Є. В. Нейронна мережа ALPHASTAR //Тези доповідей. – 2020. – С. 59. – Режим доступу до ресурсу: http://www.itconf.hneu.edu.ua/wp-content/uploads (дата звернення: 30.11.2022).

19. Тарасенко-Клятченко О. В., Буц В. В. Организация многоэтапного метода обучения сверточной нейронной сети //Міжнародний науковий журнал Інтернаука. – 2018. – Т. 1. – №. 6. – С. 44-46. – Режим доступу до ресурсу: https://www.inter-nauka.com/uploads/public/15223216798565.pdf (дата звернення: 02.12.2022).

20. Офіційний сайт бібліотеки Tensorflow [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу : https://www.tensorflow.org/api_docs/python/tf/keras. – 05.12.2018.

21. Власенко С. С. Нейромережева система управління мобільним роботом на базі Raspberry Pi. – 2019. – Режим доступу до ресурсу: https://openarchive.nure.ua/handle/document/10779 (дата звернення: 07.12.2022).

22. Репін Ю. В. Безпека та захист людини в надзвичайниї ситуаціях: навч. посіб. Миколаїв, 2005. 192 с.

23. Єфремова О. С. Збірник інструкцій з охорони праці: навч.посіб. 2008. 384 с