Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

АЛЬТЕРНАТИВНИЙ МЕТОД ДЕКОДУВАННЯ МОВНОГО СИГНАЛУ ЗГОРТКОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ

Текстова частина

магістерської роботи

за спеціальністю „Інженерія програмного забезпечення” 121

Керівник магістерської роботи
професор, д.н. М. М. Глибовець

_____
(підпис)

“____” _____ 2021 р.

Виконав студент Є.В.Редчиць
“____” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики, к.ф.-м.н.
_____С. С. Гороховський
(підпис)
„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на магістерську роботу

студенту 2 р.н. магістерської програми Комп’ютерні науки
Редчицю Євгенію Володимировичу
Розробити Альтернативний метод декодування мовного сигналу згортковою нейронною мережею

Зміст текстової частини до магістерської роботи:

Зміст
Анотація
Вступ
1 Огляд згорткових нейроних мереж та систем кодування мовного сигналу
2 Розробка архітектури альтернативного методу декодування мовного сигналу згортковою нейронною мережею
3 Реалізація прототипу програмного застосунку на базі архітектури альтернативного методу декодування мовного сигналу згортковою нейронною мережею
Висновки
Список літератури
Додатки

Дата видачі „___” _____ 2020 р.

Керівник
М.М. Глибовець, професор, д.н. _____
(підпис)

Завдання отримав
Є.В.Редчиць _____
(підпис)

Тема: Альтернативний метод декодування мовного сигналу згортковою неронною мережею

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу	01.11.2020	
2.	Огляд технічної літератури за темою роботи	15.11.2020	
3.	Виконання аналізу сучасних рішень	29.11.2020	
3.	Розробка архітектури альтернативного методу декодування мовного сигналу згортковою нейронною мережею	27.12.2020	
4.	Реалізація прототипу програмного застосунку на базі розробленої архітектури	17.01.2021	
5.	Інтеграція прототипу програмного застосунку з зовнішніми системами	14.02.2021	
6.	Налаштування прототипу програмного застосунку на прикладі бітового потоку	27.03.2021	
7.	Написання пояснювальної записки	24.04.2021	
8.	Створення слайдів для доповіді та написання доповіді	27.04.2021	
9.	Аналіз отриманих результатів з керівником, написання доповіді та попередній захист магістерської роботи	30.04.2021	
10.	Корегування роботи за результатами попереднього захисту	5.05.2021	
11.	Остаточне оформлення пояснювальної записки та слайдів	10.05.2021	
12.	Захист магістерської роботи (проекту)	18.06.2021	

Студент _____

Керівник _____

“____”_____

Зміст

Анотація

В рамках даної роботи проведено огляд згорткових нейроних мереж для розпізнавання зображень та мовного сигналу. Також проаналізовано системи кодування та декодування мовного сигналу з лінійним передбаченням. Здійснено аналіз систем кодування та декодування мовного сигналу з використанням зображення спектрограми за допомогою згорткової нейронної мережі. Запропоновано архітектуру та прототип програмного застосунку альтернативного методу декодування мовного сигналу з використанням зображення бітового потоку згортковою нейронною мережею..

Ключові слова: згорткова нейронна мережа, спектрограма, аудіосигнал, бітовий потік, кодування, декодування.

Вступ

Актуальність. За умов всесвітньої пандемії коронавірусу (COVID-19) питання комунікації людей на відстані є запорукою їх здоров'я та життя. Найбільш розповсюдженими програмними продуктами віддаленої комунікації є Teams, Zoom, Skype, Messenger, FaceTime, Hangout тощо. На сьогоднішній день розроблена велика кількість алгоритмів кодування мовного сигналу, які є основою для більшості стандартів, таких як LPC-10, MELP, CELP, G.732 тощо. В залежності від сфери використання кожен з цих стандартів може бути адаптований до відповідних умов та задовольняти потреби користувачів. Але процес вдосконалення алгоритмів не припиняється й досі, так як в різних каналах зв'язку можуть утворюватися низка чинників, які впливають на якість. З моменту використання нейронної мережі для промислових потреб з'явилися нові можливості вдосконалення обробки мови. Це дозволяє позбутися проблем, які притаманні для класичних систем кодування, таких як випадання слів під час розмови, сторонні звуки, шум, нерозбірливість тощо. Велика кількість даних, які використовуються в телекомунікаційних системах, дозволяє досить якісно провести процес навчання нейронної мережі. Але в більшості випадків, таке навчання проводять великі компанії, які мають свої власні сервіси голосових помічників (Facebook, Google, Amazon тощо). Аудиторія в мільйони користувачів надає величезний об'єм даних для щоденного вдосконалення систем розпізнавання, але за відсутності такої кількості даних провести навчання системи досить важко. Як альтернатива, існує можливість використовувати успішно зарекомендований підхід розпізнавання зображень для розпізнавання мовного сигналу. Зображення мовного сигналу має систематичний характер (амплітуда, тон, спектральні характеристики тощо) як в спектральному вигляді після обробки вокодером, так і в бітовому вигляді перед обробкою вокодером. Розпізнавання мовного сигналу за його зображенням є напрямком, який досліджується, але вже є напрацювання, які доводять позитивний характер такого дослідження.

Мета дослідження. Розробити архітектуру альтернативного методу декодування мовного сигналу згортковою нейронною мережею та створити прототип програмного застосунку на базі зазначеної архітектури.

Завдання дослідження. Проаналізувати існуючі підходи до розробки архітектур декодування мовного сигналу згортковою нейронною мережею. Вивчити та дослідити програмні продукти, що можуть бути використані як компоненти в архітектурі альтернативного методу декодування мовного сигналу згортковою нейронною мережею. Перевірити можливості інтеграції досліджених компонентів, порівняти компоненти з близькими чи подібними характеристиками, обрати компоненти, що найкраще підходять для вирішення потрібних задач. Обрати підходи та інструменти для практичної реалізації розробленої архітектури.

Об´єкт дослідження. Системи розпізнавання мовного сигналу нейронною мережею, класичні системи декодування мовного сигналу, системи розпізнавання зображень нейронною мережею.

Предмет дослідження. Метод кодування та декодування мовного сигналу за зображенням спктрограми та можливість декодування мовного сигналу за зображенням його бітового потоку згортковою нейронною мережею.

Джерела дослідження. Електронні версії друкованої літератури, програмна документація, електронні ресурси, в тому числі спеціалізовані форуми та віртуальні конференції, вихідні коди програм та бібліотек, відео-інструкції.

Наукова новизна одержаних результатів дослідження полягає у використанні саме зображення спектральних характеристик мовного сигналу для аналізу архітектури кодування та декодування мовного сигналу та розробці альтернативної архітектури декодування мовного сигналу за зображенням бітового потоку нейронною загортковою мережею.

Практичне значення одержаних результатів. За рахунок використання розробленої архітектури декодування розширюються можливості покращення

якості обробки мовного сигналу в телекомунікаційних мережах. Розпізнавання за зображенням бітового потоку здійснюється на виході вокодера, що дозволяє або повністю замінити класичний декодер, або застосувати додаткові механізми у вигляді розробленої архітектури для підвищення завадостійкості в каналах зв’язку.

РОЗДІЛ 1 Огляд згорткових нейроних мереж та систем кодування мовного сигналу

1.1. Загальні поняття згорткової нейронної мережі

Згорткова нейронна мережа (ЗНМ, англ. convolutional neural network, CNN) – це клас глибинних штучних нейронних мереж прямого поширення, який застосовується до аналізу інформаційних даних, таких як зображення, мовний сигнал, аудіосигнал тощо [1].

ЗНМ будується з вхідних та вихідних шарів, а також із декількох прихованих шарів. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів (ЗШ), агрегувальних шарів (АШ), повноз'єднаних шарів (ПШ) та шарів нормалізації (ШН). Згорткові шари застосовують до входу операцію згортки, передаючи результат до наступного шару. Згортка імітує реакцію окремого нейрону на зоровий стимул [1]. Також ЗНМ характеризуються ваговими значеннями та вектором зсуву. Кожен нейрон приймає значення, отримує скалярний добуток та за необхідності визначається нелінійною системою. Вся мережа виражає єдину диференційовану функцію оцінок, від необроблених пікселів зображення на одному кінці до оцінок класу на іншому [2, 3, 4, 5].

Операція згортки дає змогу розв'язати проблему зникання або вибуху градієнтів при використанні поверхневої (протилежної до глибинної) архітектури де кожен піксель є відповідною змінною. Наприклад, повноз'єднаний шар для (маленького) зображення розміром 100 на 100 має значення вагового коефіцієнта – 10000. ЗНМ зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою за меншої кількості параметрів. [1].

На відміну від звичайної архітектури ЗНМ мають перевагу в тому, що вхід складається з зображень і вони описують архітектуру більш раціонально. Зокрема, на відміну від звичайної нейронної мережі, шари ЗНМ мають нейрони розташовані в 3 вимірах: ширина, висота, глибина. Наприклад, вхідні зображення в CIFAR-10 (Canadian Institute for Advanced Research) є вхідними значеннями активації і дані значення мають розміри 32 на 32 на 3 (ширина, висота, глибина

відповідно). Нейрони в шарі пов’язані лише з невеликою областю шару перед ним, замість усіх нейронів повністю пов’язаним чином. Кінцевий вихідний шар для CIFAR-10 мав би розміри 1 на 1 на 10, в результаті чого повне зображення буде зменшено в один векторний показник класу, розміщений вздовж глибинного виміру. Приклад стандартної ЗНМ зображено на рисунку 1.1 [2, 3, 4, 5].

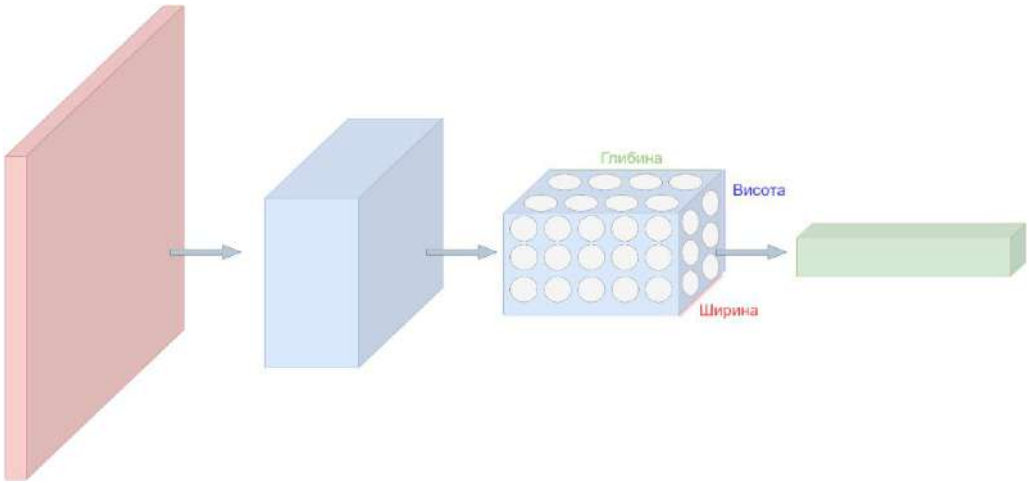


Рисунок 1.1 – Приклад стандартної ЗНМ [5]

Як зазначалося вище, для перетворення одного значення активацій в інше за допомогою диференційованої функції використовуються три основні типи шарів – ЗШ, АШ та ПШ. Дані шари складаються для того щоб сформувати повну архітектуру ЗНМ [2, 3, 4, 5].

Проста класифікація ЗНМ для CIFAR-10 може мати архітектуру зображену на рисунку 1.2 [3, 5].

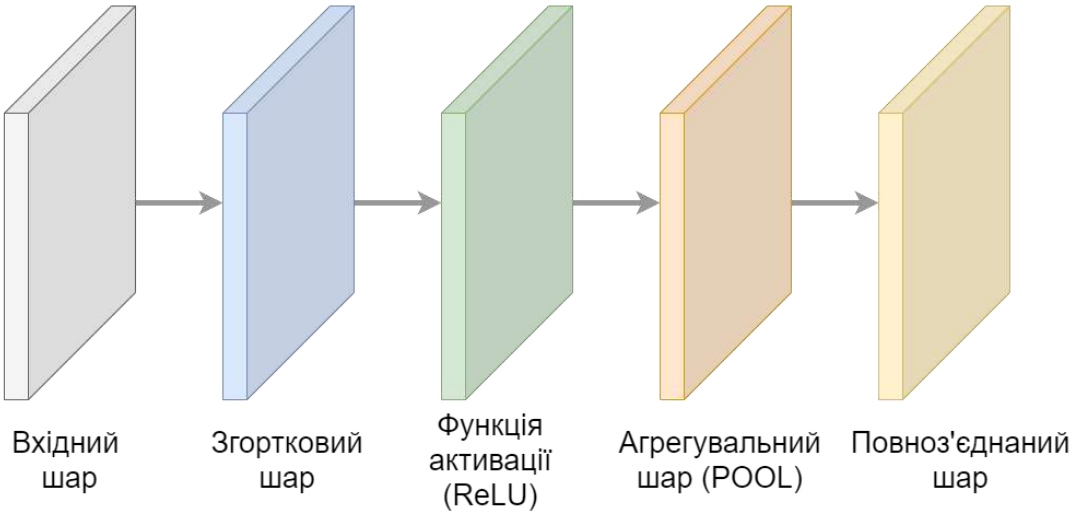


Рисунок 1.2 – Проста класифікація ЗНМ для CIFAR-10 [5]

Вхідний шар (32 на 32 на 3) буде містити значення необроблених пікселів зображення, у цьому випадку зображення має характеристики з шириною 32, висотою 32 та трьома кольоровими каналами R (Red), G (Green), B (Blue) [2, 3, 5].

ЗШ обчислює вихід нейронів, які з'єднані з локальними регіонами на вході, кожен з яких обчислює скалярний добуток між їх ваговими значеннями та невеликою областю до якої вони підключені у вхідному значенні, якщо буде прийнято рішення використовувати 12 фільтрів, результатом цього буде значення 32 на 32 на 12 [2, 3, 5].

Шар ReLU застосує функцію активації елементів, наприклад операція максимізації (*MAX(0, x))* з нульовим пороговим значенням. Таким чином, значення розміру залишається незмінним, тобто 32 на 32 на 12 [3, 5].

АШ буде виконувати операцію децимації вздовж просторових розмірів (ширина, висота), в результаті чого буде досягнуто значення 16 на 16 на 12 [3, 5].

ПШ обчислює оцінку класів, в результаті отримуючи значення розміру 1 на 1 на 10, де кожне з 10 чисел відповідає оцінці класу. Як і у звичайних нейронних мережах кожен нейрон у цьому шарі буде з'єднаний з усіма числами попереднього значення [2, 3, 5].

ЗНМ перетворює вихідний шар за шаром від початкових значень пікселів до кінцевих показників класу. Деякі шари містять параметри, а інші не містять. Зокрема, шари ЗШ/ПЗ виконують перетворення, які є функцією не тільки активації вхідного значення, але і параметрів (вагове значення та вектор зсуву). З іншого боку, шари ReLU/АШ будуть реалізовувати фіксовану функцію. Параметри шарів ЗШ/ПЗ будуть навчатися з градієнтним спуском так, щоб оцінка класу, що обчислює ЗНМ, відповідали міткам у навчальному наборі для кожного зображення [2, 3, 4, 5].

1.2 Використання згорткової нейронної мережі для розпізнавання мовного сигналу

Процес побудови ЗНМ для розпізнавання мовного сигналу можна розділити на наступні етапи: підбір данних та побудова датасету, визначення

характеристик сигналу, навчання та тестування ЗНМ, розгортання ЗНМ (рисунок 1.3) [6].

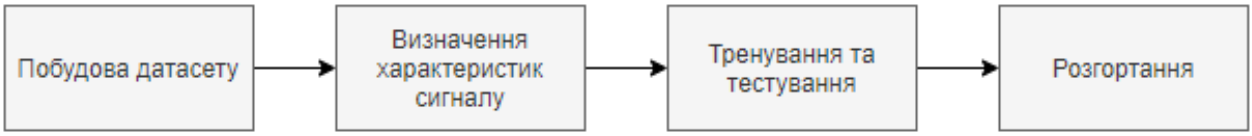


Рисунок 1.3 – Етапи побудови ЗНМ для розпізнавання мовного сигналу

На першому етапі необхідно визначити тип та структуру даних, які будуть використовуватись для навчання ЗНМ та її валідації. Правильно розроблений датасет може виконувати вирішальну функцію в ефективності ЗНМ. Кількість окремих слів, які використовуються для навчання, залежить від складності ЗНМ, і можуть сягати сотні тисяч [6].

На другому етапі необхідно визначити відповідні характеристики сигналу для його сумісності з ЗНМ. Такими характеристиками можуть бути спектральні та енергетичні складові сигналу отримані після перетворення Фур'є, кодовані на відповідні числа слова або кольорові канали зображення [6].

На наступному етапі виконується безпосереднє навчання ЗНМ за допомогою одного або декількох алгоритмів. Одним з таких алгоритмів може бути зворотнє розподілення для автоматичного оновлення параметрів моделі (вагові коефіцієнти та зміщення), це дозволить підвищити відповідність прогнозів до очікуваних результатів. Для досягнення відповідної точності ЗНМ можливе використання різних моделей та гіперпараметрів, наприклад, типи функцій, які використовуються, розмір та форма моделі тощо.

Далі здійснюється безпосереднє розгортання або використання моделі у відповідному середовищі. Розгортання може здійснюватися як локально, використовуючи обчислювальні ресурси персонального ком'ютера, так і глобально, використовуючи обчислювальні ресурси віддалених серверів у хмарах (Google Colab).

Для того щоб детектувати мовний сигнал на певному проміжку часу необхідно виконати для нього процедуру швидкого перетворення Фур'є (ШПФ).

Таким чином, буде визнано амплітудно-частотна характеристика (спектрограма) даного мовного сигналу.

Наступним кроком є фільтрація сигналу через відповідний фільтр, який має лінійну періодичність до 1 кГц та логарифмічну періодичність після 1 кГц. Це дозволить нормалізувати мовний сигнал у відповідності до того, як його сприймає людський слух.

Енергія кожного з цих чисел додається для того щоб створити вектор відповідних чисел. Далі визначається логарифм цих чисел та здійснюється дискретне перетворення по косинусу (ДПК). Як результат, ми отримаємо набір спектральних коефіцієнтів де нижні значення (наприклад 0 та 1) описують безпосередньо спектр мовного сигналу, а верхні значення (більше 2) ідентифікують другорядні значення (наприклад шум). На рисунку 1.4 зображено описаний вище процес трансформації сигналу.

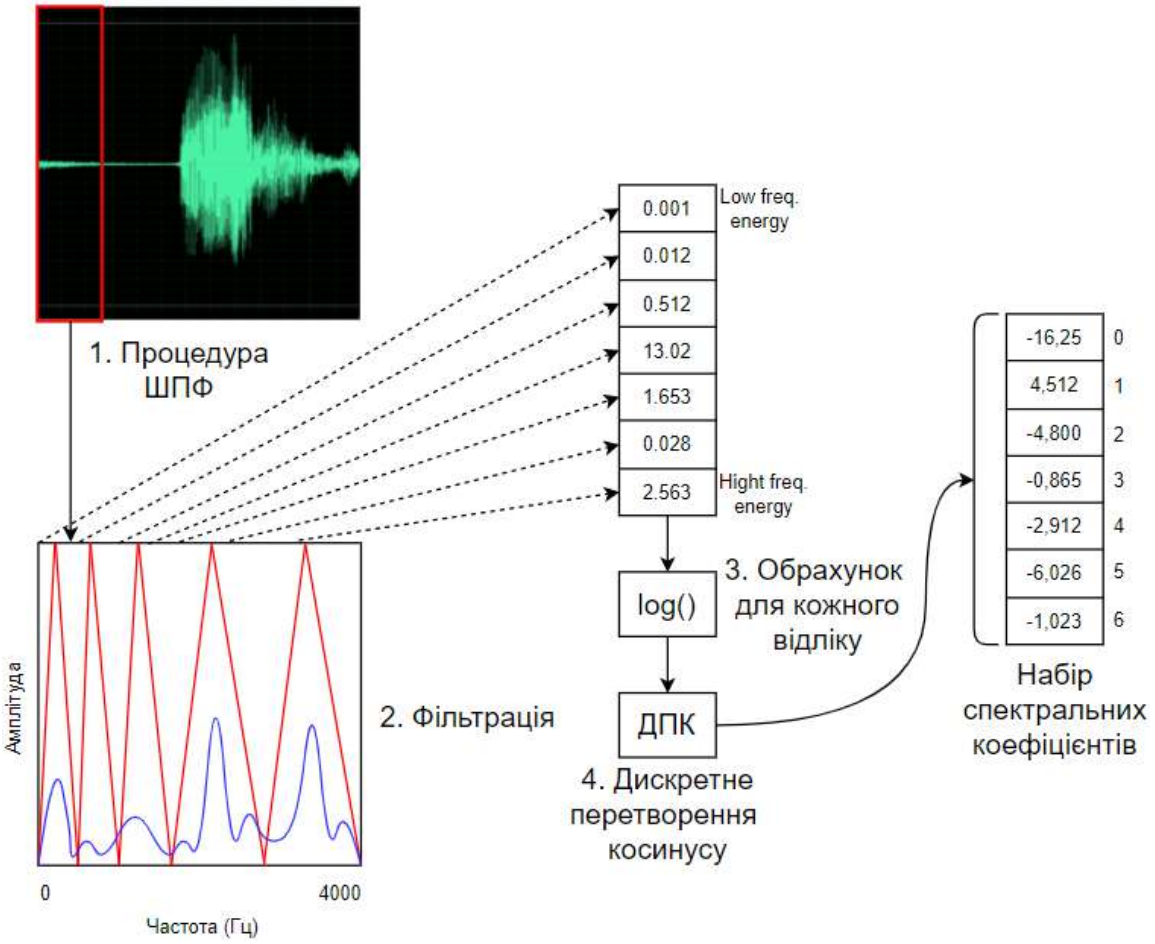


Рисунок 1.4 – Процес трансформації сигналу

Переважна більшість систем розпізнавання мовного сигналу використовує перші 13 коефіцієнтів. На рисунку 1.5 зображено процес трансформації сигналу з візуалізацією коефіцієнтів.

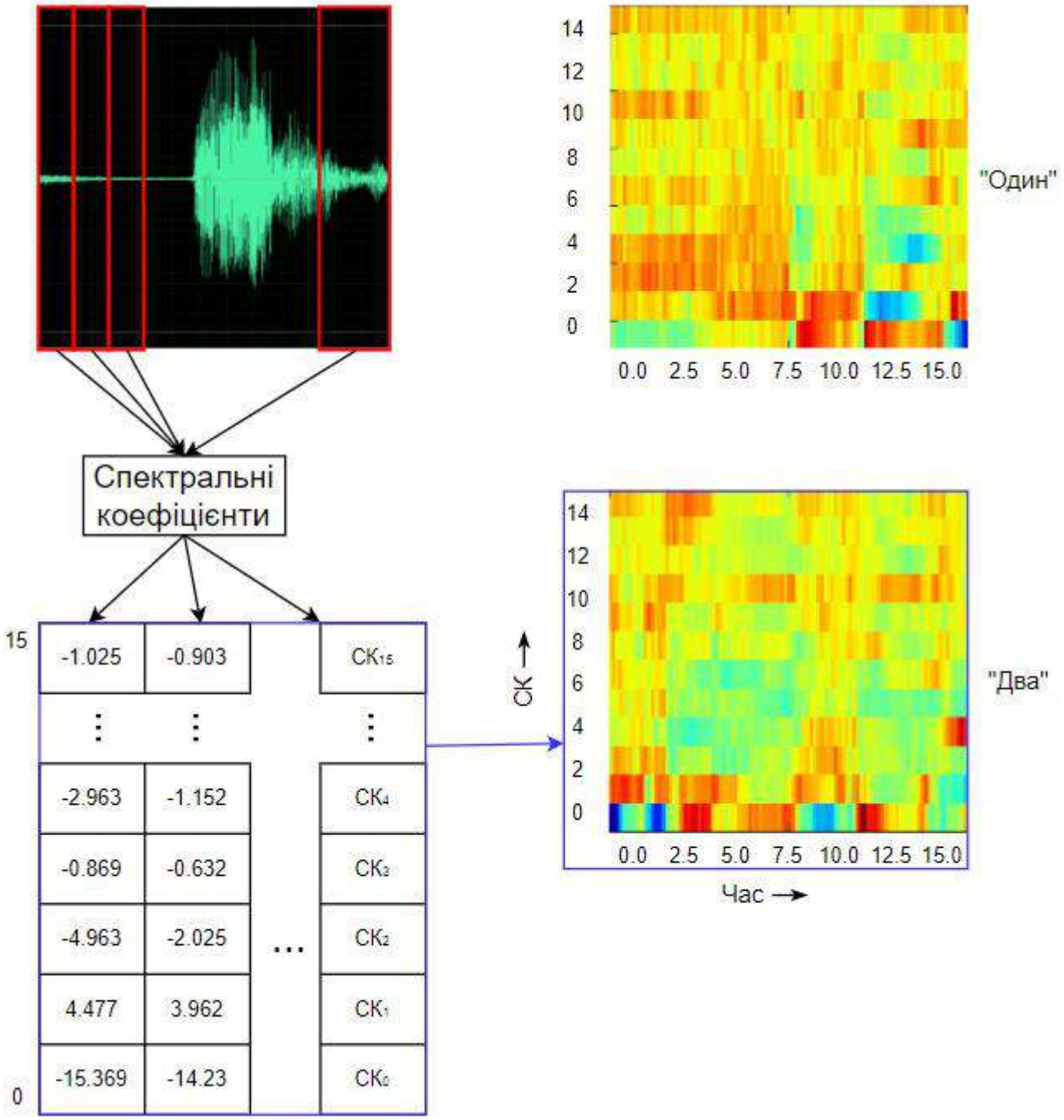


Рисунок 1.5 – Трансформація сигналу з візуалізацією коефіцієнтів

В даному випадку зображено процес наповнення умовного масиву (16 на 16) шляхом ітеративного руху вікна, обрахунку коефіцієнтів та відображенні результатів у масиві відповідно до часу.

В подальшому нейронна мережа навчається розпізнавати отримані масиви коефіцієнтів та робити висновки про відповідність їх істинному значенню.

Для навчання нейронної мережі готуються вхідні данні, які розподілені за приблизною схемою: 10 % - крос-валідація, 10 % - тестовий набір даних, 80 % - навчальний масив. Таке співвідношення не є обов'язковим, воно має тільки рекомендаційний характер.

В даному випадку навчальний масив використовується безпосередньо для навчання мережі, крос-валідація необхідна для відслідковування прогресу навчання мережі з показниками втрат, тестовий набір порівнюють з тренованою моделлю в самому кінці навчання, це необхідно для остаточного визначення показників втрат та успішності мережі [6].

1.3 Загальні відомості про кодування з лінійним передбаченням

Для формування спектру мовного сигналу передбачається, що всі голосові зв'язки є джерелом збудження, а голосовий тракт – нелінійним фільтром. Параметри джерела та фільтра в даному випадку можуть змінюватись динамічно. Вокодерами називають системи, які використовують методи стиснення параметрів мовного сигналу людини [7].

Робота вокодерів побудована на моделі джерела, з якого одержується інформація про параметри мовного сигналу (LPC, MBE, Channel, Formant, Phase, Sinusoidal), на відміну від деяких систем, які використовують метод стиснення форми мовного сигналу (PCM, DPCM, ADPCM, DM, ATC, SBC) [7].

Для вокодерів, результатом стиснення будуть не коди сигналів, а коди параметрів джерела цих сигналів. Основними параметрами, які використовує вокодер для стиснення є параметри фільтра створення мови, вказівник на голосний/приголосний звук, рівень збудження сигналу та період основного тону для голосних звуків [7].

Вокодери можливо розділити за принципами аналізу та синтезу мовних сигналів, а саме на смугові (канальні), формантні, фонемні, ортогональні, гомоморфні. Для канальних вокодерів сигнали з виходів фільтрів детектуються,

пропускаються через фільтри низької частоти, дискретизуються та піддаються двійковому стисненню. Як результат, визначаються параметри мовного тракту людини, а також період основного тону збудження та вказівник на голосний/приголосний звук. Формантні вокодери мають огинаючу спектру мови, яка описується комбінацією формант (резонансних частот голосового тракту) з такими основними параметри формант, як центральна частота, амплітуда та ширина. В ортогональних вокодерах огинаюча спектра розкладається в ряд по вибраній системі ортогональних базисних функцій, де обчислені коефіцієнти цього розкладання передаються на декодер [7, 8, 9]. Для гомоморфної обробки сигналів мовний сигнал можна представити як тимчасову згортку імпульсної перехідної характеристики мовного тракту з сигналом збудження [8]. В такому випадку для частотної області це відповідає добутку частотної характеристики голосового тракту та спектра сигналу збудження. [7].

Вокодери з алгоритмом лінійного передбачення (LPC – linear predictive coding), описують мовний тракт людини лінійним фільтром з безперервною імпульсною перехідною характеристикою. В зазначеному фільтрі кожне чергове значення сигналу може бути отримано як лінійна комбінація деякого числа його попередніх значень [8]. В LPC вокодері мовний сигнал ділиться на блоки, для кожного з яких визначаються коефіцієнти фільтра передбачення. Зазначені коефіцієнти квантуються та передаються декодеру. На наступному кроці мовний сигнал пропускається через фільтр у якого частотна характеристика зворотна частотній характеристиці мовного тракту. Вихідним результатом фільтра є помилка передбачення. Таким чином, набагато виразніше виявляється тривала кореляція в сигналі, і як наслідок, можливо точніше визначити частоту основного тону та виділити вказівник на голосний/приголосний звук. При задовільній якості мовлення виділяють наступні швидкості кодів для вокодерів, побудованих згідно з алгоритмом LPC – в межах від 1 до 2,4 Кб/с. Загальним недоліком алгоритмів стиснення мови, побудованих згідно з методом стиснення параметрів мовного тракту людини, є їх синтетична якість мовлення. Можливо зазначити, що збільшення швидкості коду практично не покращує якості мови. Дані

алгоритми стиснення в основному використовують у спеціалізованих системах зв'язку, де основним завданням є досягнення великого коефіцієнту стиснення.[7].

Вокодер типу LPC-10 є одним з найрозповсюдженіших реалізацій систем лінійного передбачення. Даний вокодер використовується для успішної оцінки основних мовних параметрів, таких як тон мовного сигналу, рівень збудження сигналу та коефіцієнти фільтра передбачення. Загальну структурну схему роботи вокодера LPC-10 можна побачити на рисунку 1.6 [5, 10].

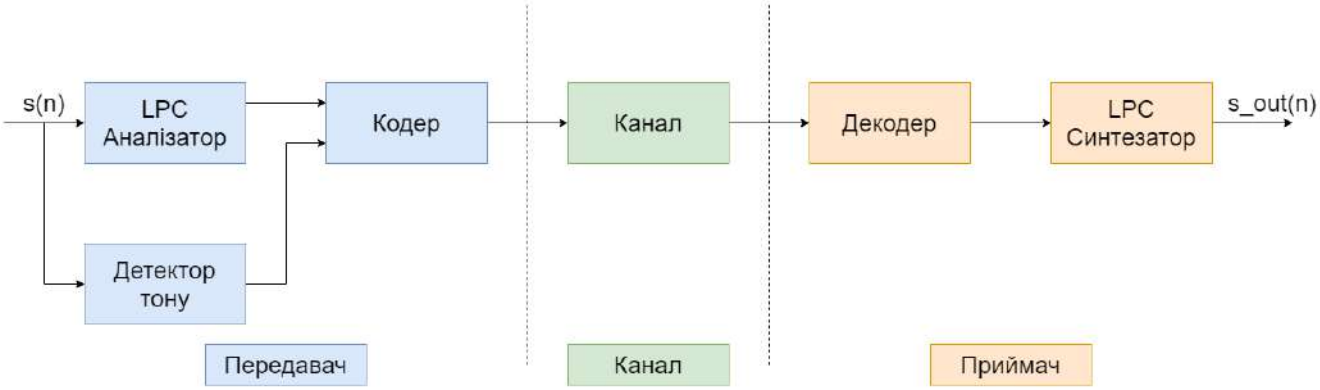


Рисунок 1.6 – Загальна структурна схема вокодера LPC-10 [5, 10]

В даному випадку вхідний мовний сигнал S(n) потрапляє на вхід передавача з подальшим аналізом відповідних параметрів (тон мовного сигналу, рівень збудження сигналу та коефіцієнти фільтра передбачення). Далі проаналізовані параметри кодуються та передаються в канал зв'язку (радіоканал, провідний або оптичний). Більш детально етап трансформації мовного сигналу зображено на рисунку 1.7.

Рисунок 1.7 – Трансформація мовного сигналу з боку передавача

Перед потраплянням мовного сигналу на аналогово-цифровий перетворювач (АЦП) відбувається його попередня фільтрація смуговим фільтром в діапазоні від 100 до 3600 Гц. Це необхідно для виділення більш високих частот, що в подальшому позитивно вплине на розрахунок коефіцієнтів фільтра передбачення. Після фільтрації та АЦП відбувається аналіз LPC для визначення коефіцієнтів передбачення. Також, паралельно розраховується рівень збудження сигналу. Додатково після АЦП аналізуються тон сигналу та вказівник на наявність сигналу (детектор мови) для визначення обробки коефіцієнтів передбачення (після аналізу LPC), як для мовного або немовного сигналу (шум або тиша наприклад). У випадку аналізу сигналу як мовного, визначається також його тон, в іншому випадку сигнал визначається, як немовний або як такий, що знаходиться в процесі переходу з немовного в мовний стан. Далі відбувається кодування визначених параметрів (тон мовного сигналу, рівень збудження сигналу та коефіцієнти фільтра передбачення), також додається біт синхронізації та можливий контроль помилок під час передачі мовного сигналу.

На рисунку 1.8 зображено трансформацію сигналу на боці приймача.

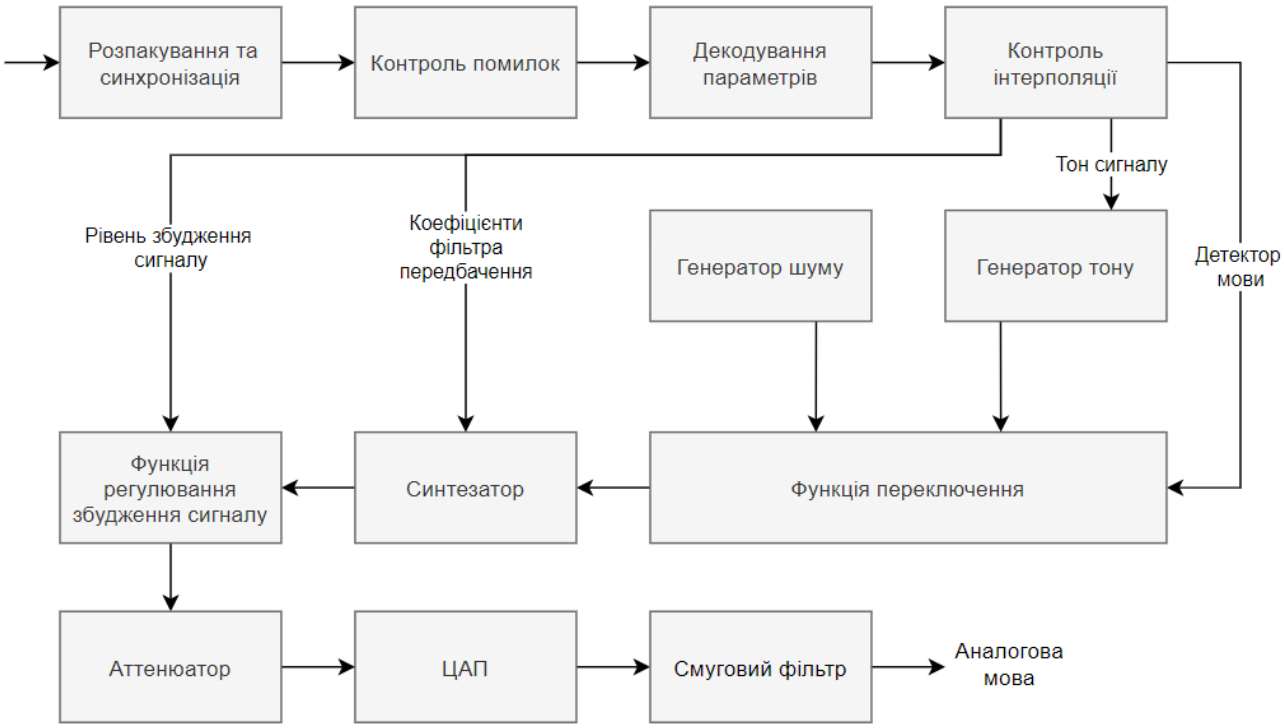


Рисунок 1.8 – Трансформація мовного сигналу з боку приймача

Першочергово відбувається розпакування вхідного сигналу та його синхронізація з виправленням помилок, якщо вони присутні в сигналі. Далі здійснюється декодування параметрів з інтерполяцією за визначеною схемою для забезпечення відповідного рішення щодо наявності мовного сигналу, його тону (для мовного сигналу), коефіцієнтів фільтру передбачення та рівня збудження. З визначених параметрів, рішення про наявність мовного сигналу передається на функцію переключення, яка обирає генератор випадкового шуму (для немовного сигналу) або тоновий генератор (для мовного сигналу), як вхід синтезатора. Визначений тон мовного сигналу передається до тонового генератора (за умови мовного сигналу), коефіцієнти фільтра передбачення передаються на вхід синтезатора і рівень збудження передається на функцію регулювання збудження сигналу. Далі сигнал вирівнюється відповідно до вхідного підсилення, перетворюється з цифрового в аналоговий вигляд та фільтрується [11, 12].

В таблиці 1 зазначено розподілення біт одного вокодерного фрейму відповідно до визначених параметрів.

Таблиця 1.1

	Мовний сигнал (біт)	Немовний сигнал (біт)
Тон/мова	7	7
Рівень збудження сигналу	5	5
К (1)	5	5
К (2)	5	5
К (3)	5	5
К (4)	5	5
К (5)	4	-
К (6)	4	-
К (7)	4	-
К (8)	4	-
К (9)	3	-
К (10)	2	-
Контроль помилок	-	20
Синхронізація	1	1
Не використовується	-	1
Загальна сума	54	54

Примітки

1 К – коефіцієнт фільтра передбачення.

2 Немовний сигнал – відсутня наявність мови або сигнал в процесі переходу з немовного в мовний стан.

РОЗДІЛ 2 Розробка архітектури альтернативного методу декодування мовного сигналу згортковою нейронною мережею

2.1. Аналіз архітектурних реалізацій розпізнавання мовного сигналу використовуючи зображення його параметрів

Важливою складовою у розпізнавані мовного сигналу як прямим методом, так і за допомогою зображення його параметрів є виділення спектральних коефіцієнтів. Дані коефіцієнти описуються значенням мел (mel) та частотою. Мел – це психофізична одиниця висоту звуку, отримана від сприйняття органами слуху людини. Сприйняття звуків людським слухом відбувається нелінійно [13]. На рисунку 2.1 зображено графік залежності мел-коефіцієнтів від частоти.

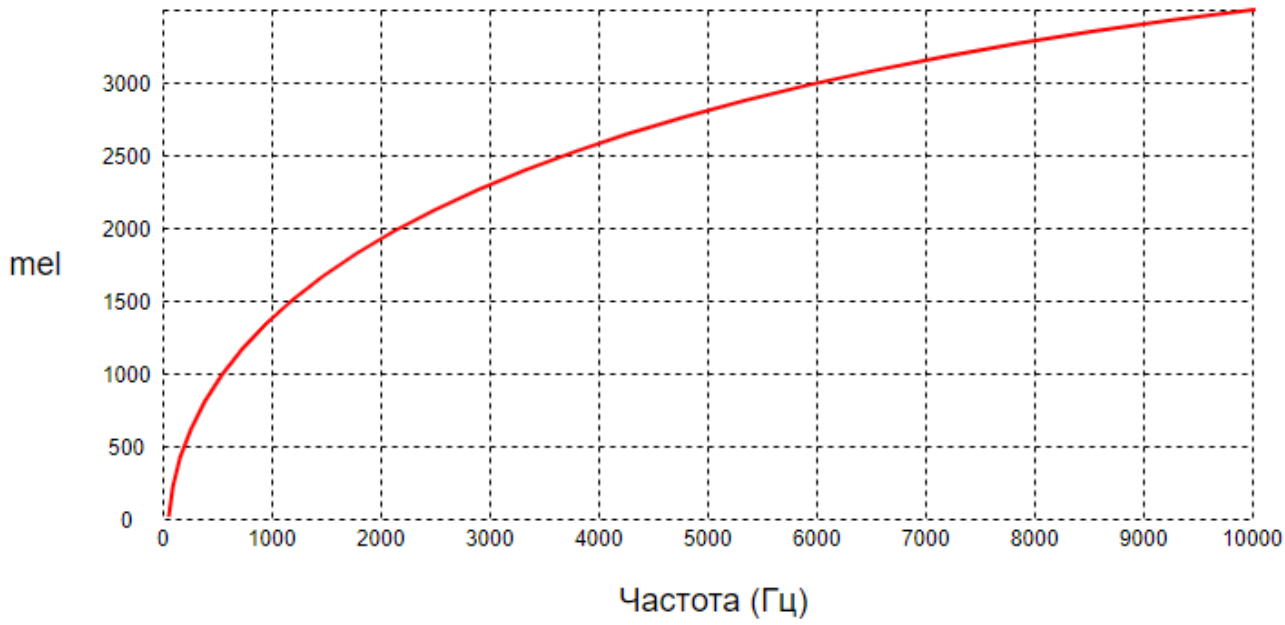


Рисунок 2.1 – Залежність значення мел від частоти

Значення мел-коефіцієнтів розраховується за формулою 2.1.

$$m = 1125 \ln\left(1 + \frac{f}{700}\right), \tag{2.1}$$

де f – це частота.

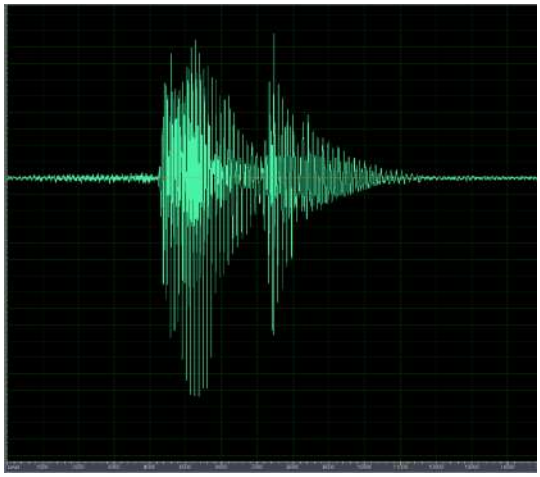
Іншою важливою складовою в процесі розпізнавання мовного сигналу різними системами є кепстр. Кепстр – це функція зворотного перетворення Фур'є від логарифму спектру потужності сигналу, яка визначається за формулою 2.2.

$$C_s(q) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \ln|S(\omega)|^2\, e^{i\omega q} d\omega, \qquad (2.2)$$

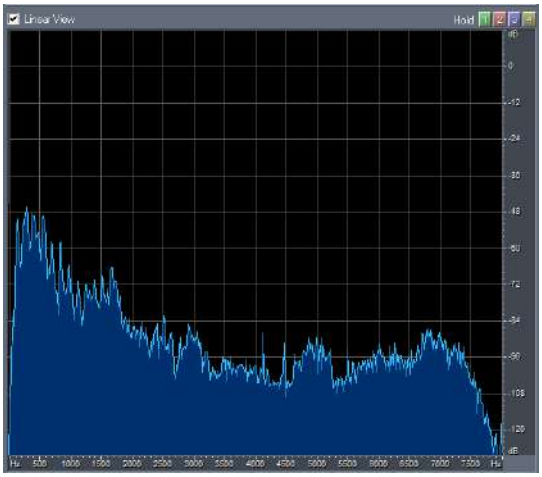
де q – значення кепстрального часу;

S(ω) – спектр сигналу.

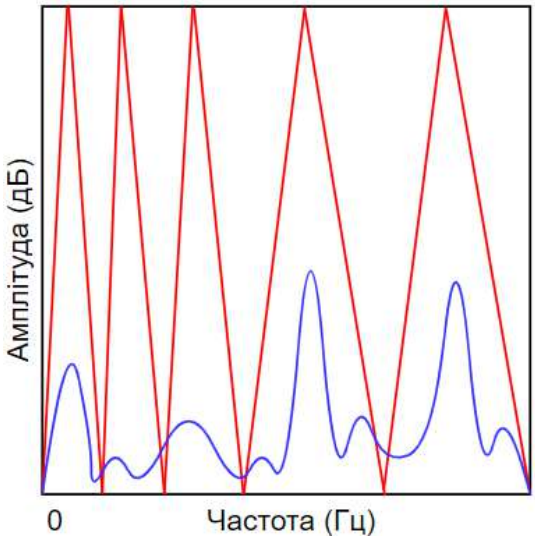
На рисунку 2.2 позначено набір зображень, що описують порядок отримання мел-коефіцієнтів.



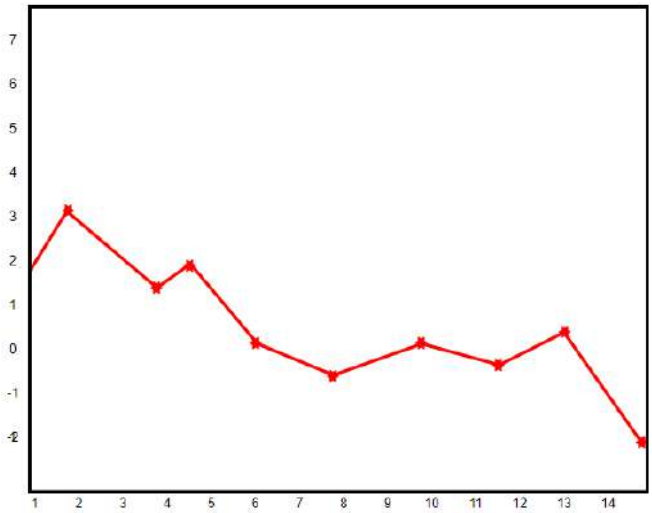
а)



б)



в)



г)

Рисунок 2.2 – а) представлення мовного сигналу на часовій спектрограмі; б) представлення сигналу на частотно-часовій спектрограмі; в) вікна фільтрації; г) представлення мел-коефіцієнтів після дискретного перетворення по косинусу

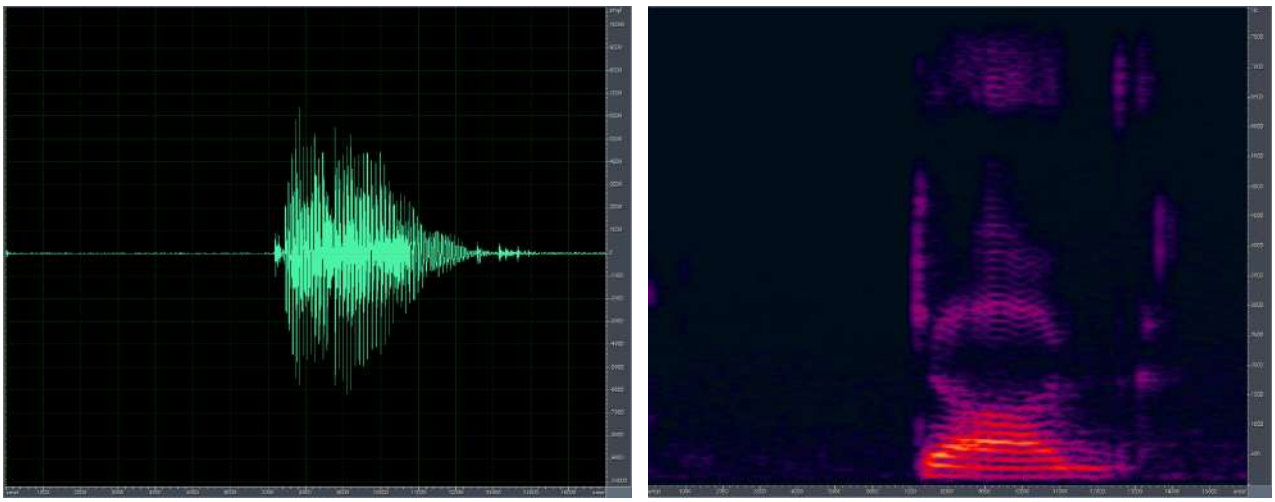
Для отримання зображення сигналу в частотно-часовій спектрограмі, необхідно виконати перетворення Фур'є сигналу в початковій формі – на часовій

спектрограмі. На наступному кроці здійснюється фільтрування сигналу відповідно до шкали мел та частоти. На останньому кроці, отримані коефіцієнти з кожного вікна фільтру зводяться до степені два та логарифмується. Для отримання кепстральних коефіцієнтів виконується дискретне перетворення по косинусу[13].

На сьогоднішній день існує велика кількість реалізацій систем розпізнавання мови, які починалися з розпізнавання окремих команд (окремих слів). На шляху до вдосконалення цих систем було запропоновано використовувати ЗНМ для розпізнавання зображень. В якості зображень пропонується використовувати зображення спектрограми сигналу [14].

Аналізуючи системи для розпізнавання зображень, зазначається, що переважна їх більшість використовує на вхід зображення з співвідношенням сторін 1 до N, де значення N становить від 1 до 2. Для того щоб адаптувати аудіо сигнал до вимог ЗНМ, застосовується виділення спектрограми сигналу. Зазначена спектрограма може бути отримана шляхом використання дискретного перетворення Фур'є, в такому випадку, одновимірне представлення сигналу в часі трансформується в двовимірне. В результаті трансформації зберігається вся інформація про сигнал, але зображення відповідає вимогам ЗНМ. В даному випадку, кожен стовпець представляє рівень збудження під час перетворення Фур'є в окремому часовому вікні на загальному часовому відрізку. В результаті, ми отримаємо двовимірне зображення співвідношення часу до частоти по рівню збудження у порівнянні з початковим виглядом одновимірного співвідношення часу до рівня збудження. На рисунку 2.3 зображено вигляд сигналу до перетворення Фур'є, на рисунку 2.2 – після перетворення. Як наслідок, після перетворення можна безпосередньо побачити, яка частотна складова змінюється та як спектр змінюється відповідно до часу [14].

Зазначається, що вигляд сигналу після перетворення Фур'є має бути більш придатним для аналізу ЗНМ.



а) б)

Рисунок 2.3 – а) вигляд сигналу до трансформації Фур’є; б) вигляд сигналу після перетворення Фур’є

Якщо порівнювати два зображення, на другому можна побачити контур тону мовного сигналу (горизонтальні лінії), що не характерно для першого зображення. На рисунку 2.3 горизонтально виділені різні значення тону відповідно до частот, як вони формуються під час мовлення людини. Якщо зазначений сигнал розповсюджується в просторі, то він сприймається людським слухом відповідно до кожної частоти. В даному випадку, при отриманні зображення мовного сигналу, функціонування ЗНМ частково повторює функціонування людського слуху [14].

Розглядається також можливість використання маштабнонезалежного перетворення ознак (SIFT – Scale-invariant feature transform) або використання гістограми напрямлених об’єктів (HOG – histogram of gradients) для виділення параметрів з зображення [15]. На рисунку 2.4 зображено процес виділення градієнтних параметрів зображення.

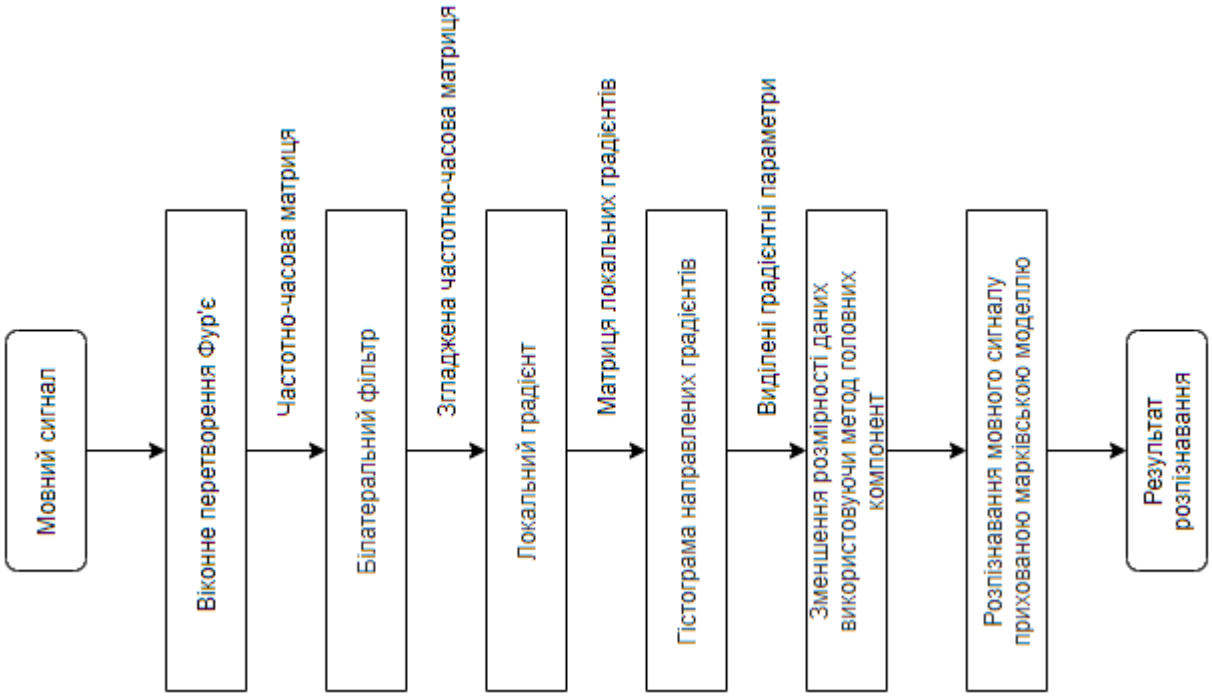
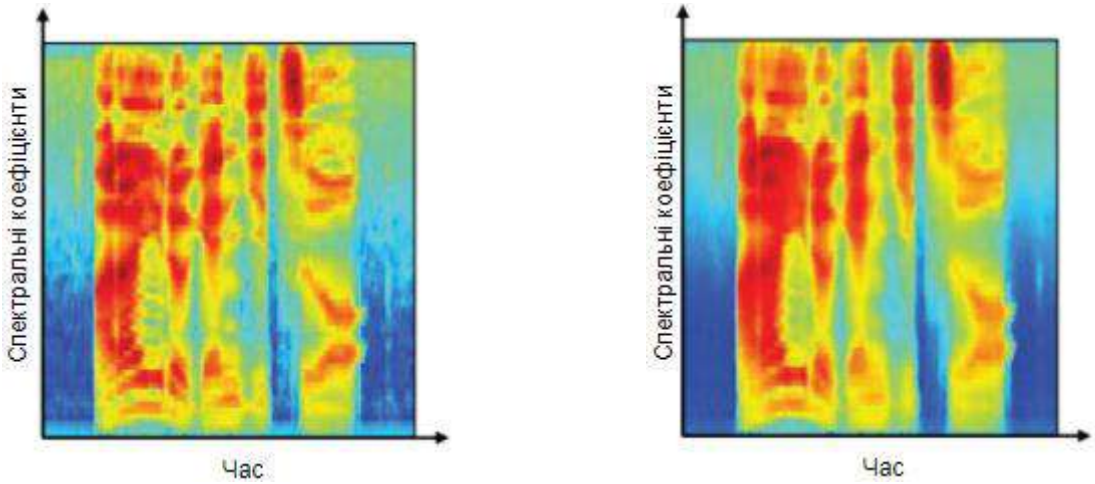


Рисунок 2.4 – Процес виділення параметрів на основі градієнту зображення

На першому кроці виконується віконне перетворення Фур'є для отримання частотно-часового представлення сигналу і як наслідок, частотно-часової матриці вікон (фреймів) спектру. Далі застосовується білатеральний фільтр для видалення завад та згладження спектру. На рисунку 2.5 зображено вид вихідного сигналу після віконного перетворення Фур'є до і після білатерального фільтру [15].



а) б)

Рисунок 2.5 – а) сигнал до білатерального фільтру; б) сигнал після білатерального фільтру [15];

На рисунку 2.6 зображено параметри локального градієнту в частотно-часовому представленні. Де значення рівня та вектори градієнту отримані навколо вихідної позиції використовуючи для цього згладжену частотно-часову матрицю [15].

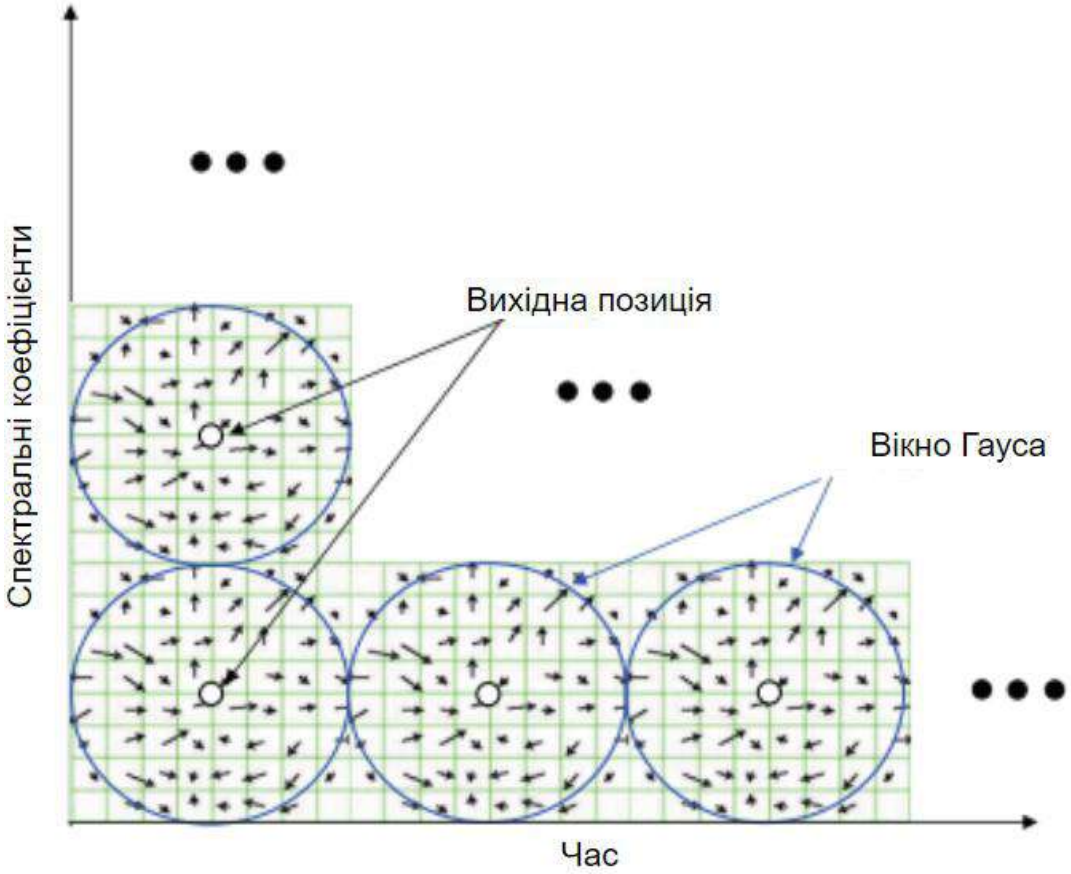


Рисунок 2.6 - Параметри локального градієнту в частотно-часовому представленні [15]

Гістограма направлених градієнтів (ГНГ) вирахувана з параметрів локального градієнту з площини 8 на 8 в площину 4 на 4 (рисунок 2.7). Кожна ГНГ визначена з рівня градієнту та його вектору. Параметр локального дескриптора сформований як вектор, який складається з чотирьох ГНГ. До даного параметру локального градієнту входить 8 напрямків помножених на 4 площини, що дорівнює 32 значенням для кожної вихідної позиції [15].

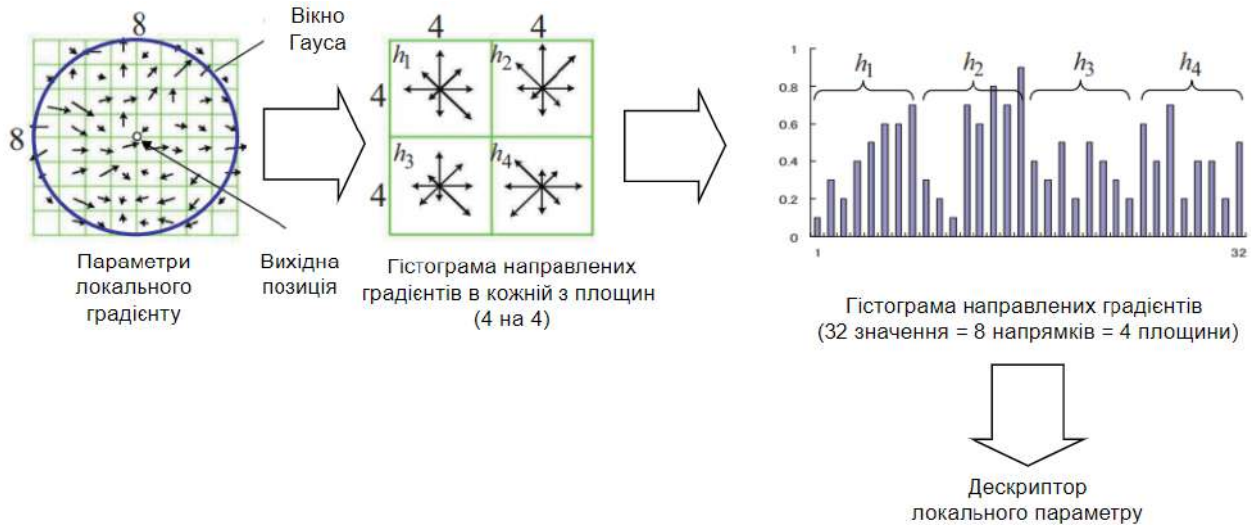


Рисунок 2.7 – Визначення локального дескриптора [15]

На завершальному етапі, вектор виділеного градієнтного параметру X визначений складанням локальних дескрипторів на кожному вікні (фреймі) (рисунок 2.8). В даному прикладі було визначено 64 спектральні коефіцієнти і 8 вихідних позицій. В такому випадку кількість значень вектору X дорівнює 256 (32 елементи на 8 дескрипторів). Дане значення є зависоким для запропонованої марківської моделі, тому використовується зменшення розмірності даних за методом головних компонент [15].

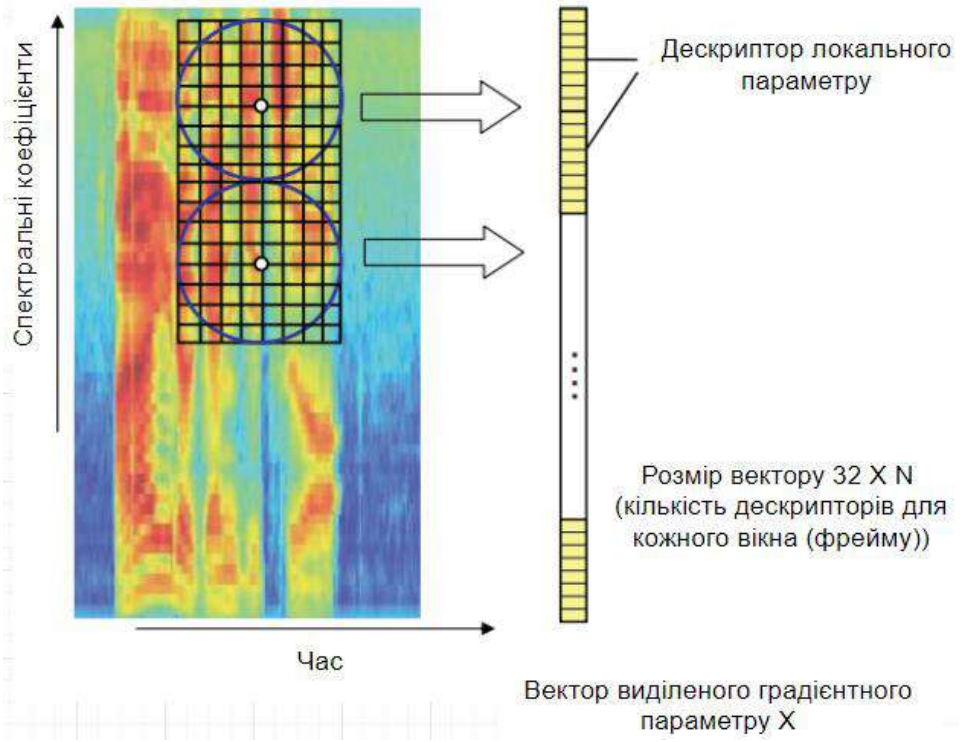


Рисунок 2.8 – Складання локальних дескрипторів[15]

2.2. Аналіз методу кодування та декодування мовного сигналу для розпізнавання згортковою нейронною мережею

В даному розділі буде розглянуто метод, який був запропонований групою науковців з Технологічного університету Ченстохова, Республіка Польща – професором Маріусом Кубанеком, професором Дженуш Бобульскі та Джоаною Кулавік.

Проаналізовано новий підхід в розпізнаванні мови, який побудований на кодуванні та декодуванні часових та частотних характеристик мовного сигналу. Для реалізації даного підходу пропонується застосовувати ЗНМ, так як, даний тип мережі показує високу стійкість до крос-спектральних спотворень та різниці між довжинами сигналу в мовному тракті. До опису даного підходу, зазвичай використовувалось застосування двох шарів згортки – часова та частотна. Новизною в даному методі є використання трьох окремих шарів – часової згортки та представлення частотної згортки, як згортки спектральних коефіцієнтів та спектральної згортки. Також застосовується підхід використання розпізнавання зображення за окремими кольоровими каналами (червоний, зелений та синій (RGB – Red, Green, Blue)). Метод враховує розпізнавання, як окремих слів, так і словосполучень [16].

Загальноприйняті методи використання ЗНМ для розпізнавання мовного сигналу будуються на виділенні частотних характеристик сигналу. В даному випадку пропонується застосовувати спектральні характеристики мовного сигналу для побудування мережі. Особливістю даного методу є використання трьох згорткових шарів (ЗШ). Кожен з ЗШ відповідає за відповідну характеристику сигналу, а саме часову згортку, згортку спектральних коефіцієнтів (ЗСК) та спектральну згортку загалом. Використання часової згортки допомагає позбавитись проблем пов’язаних з часовими розбіжностями. Проблеми з спектральним зсувами, пов’язані з різною довжиною вокальних пачок, вирішуються за допомогою використання частотної згортки. Використання згортки спектральних коефіцієнтів зменшує чутливість до сторонніх завад та шумів, а також дозволяє адаптувати частотний аналіз мовного

сигналу, який близький до сприйняття сигналу людським слухом. Спектр аналізує сигнал лінійно, тоді як ЗСК виконує лінійний аналіз до частоти 1 кГц, а на 1 кГц і вище аналіз характеристик відбувається експоненціальною. На рисунку 2.9 схематично зображено будова зазначеного методу [5, 16].

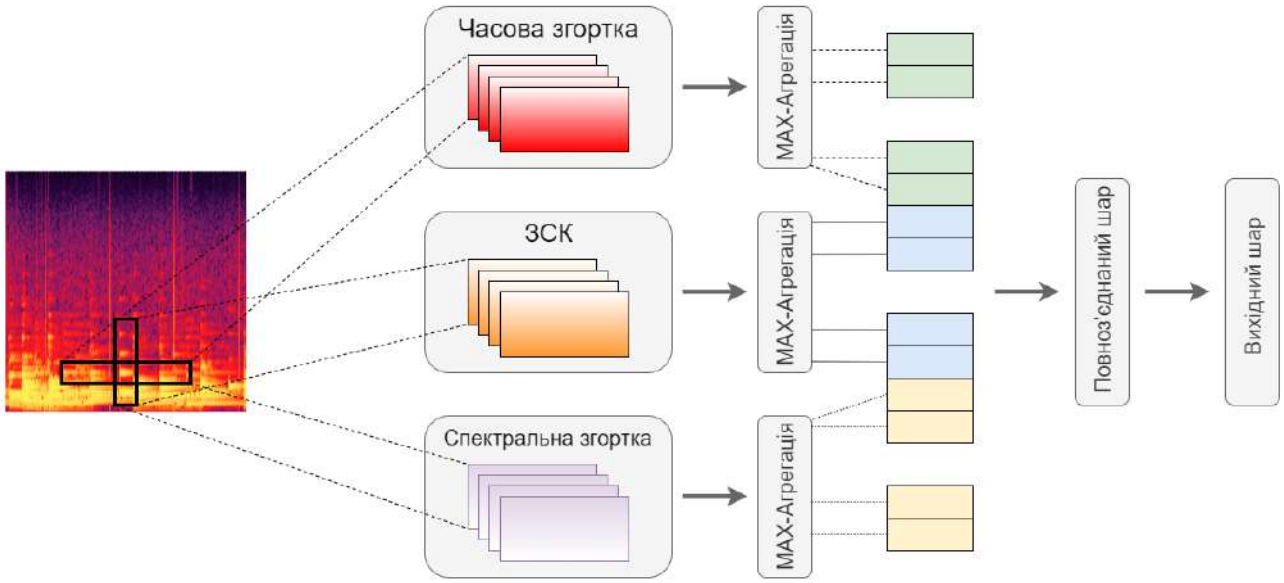


Рисунок 2.9 – Частотно-часова згортка для ЗНМ [5].

Для кодування та декодування мовного сигналу даний метод використовує підхід розпізнавання зображення за кольоровими каналами (RGB). Для кодування ЗСК було виділено R компонент, для кодування часових характеристик застосовано G компонент, і для кодування загального спектру G компонент відповідно. Згортка всіх трьох компонентів R, G, B відбувається паралельно. Карта активації, яка використовується для кодування, має різні розміри. При створенні кольорового зображення важливою частиною є масштабування RGB компонентів до єдиного розміру. Отже, було визначено розмір для окремих звуків 120 на 120 пікселів.

Запропонована ЗНМ складається з 15 шарів. Перший ЗШ включає в себе 64 фільтри розміром 9 на 9. Три ЗШ відповідають за кодування інформації, яка передається до повноз’єднаного шару (ПШ). Дана схема зображена на рисунку 2.10 [5, 16].

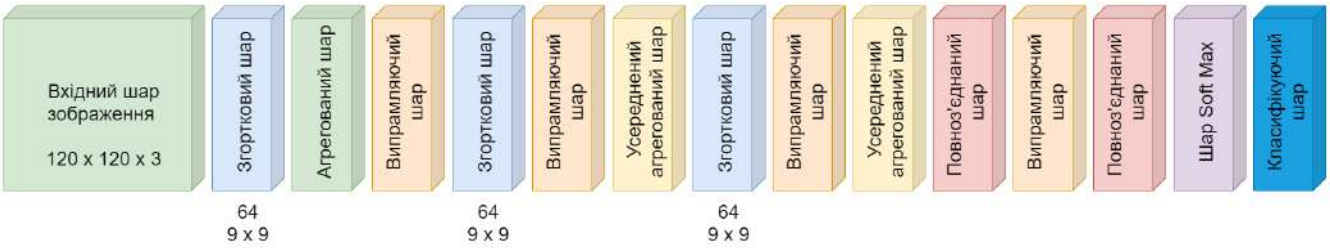


Рисунок 2.10 – Структура альтернативної ЗНМ [5, 16]

Останній ПШ та класифікуючий шар (КШ) мають X можливих вихідних варіантів [5, 16].

Дані X варіанти залежать від навчального набору даних ЗНМ. Вхідні дані можуть бути як цілі слова, так і склади. Для окремих слів X, це кількість визначених слів. Для складів X, це кількість різних складів визначених з навчального набору даних. Бувають випадки коли алгоритм розділення комбінує окремі звуки з тишею [5, 16].

Дослідження включали два важливі питання. Перший стосувався правильності роботи алгоритму, реалізуючи поділ мови на склади. Був застосований метод, який враховує енергію сигналу та частоту сигналу для розглянутого нерухомого фрагмента. Розділення на окремі склади проводили для експериментально вибраних порогових значень. Для кожного ізольованого складу (який також міг включати тишу) зображення створювалося, як графічний візерунок. Експеримент проводили для ізольованих слів і безперервного мовлення. Для безперервного мовлення прийнято різне число слів. Таблиця 2.1 показує результати експерименту щодо поділу на склади. Дослідження проводилось на базі даних університету та публічних аудіокниг [5, 16].

Таблиця 2.1 [5, 16]

Номер експерименту	Тип експерименту	Кількість слів (1) та речень (2 - 4)	Правильна вибірка, %
1	Окремі слова	70	98.1
2	безперервна мова (від 3 до 7 слів)	65	86.7
3	безперервна мова (від 6 до 12 слів)	65	80.5
4	безперервна мова (від 10 до 20 слів)	65	69.7

Отриманий результат важко порівняти з іншими існуючими методами, оскільки попередні підходи включали лише одну або дві складові (частотну та спектральну). Крім того, ці методи використовувались для зображень у масштабах сірого, в той час як даний метод використовує потрійний аналіз із застосуванням RGB каналів [5, 16].

Зазначається, що найкращі результати були отримані для відокремлених слів – 98%. Очевидно, що результати, отримані для безперервного мовлення, мають менший показник розпізнавання. Проте, результат 70 – 80% вважають задовільним [5, 16].

Другий тип досліджень стосувався оцінки ефективності розпізнавання мовлення для ізольованих слів та для постійного мовлення. Записані дані були поділені на навчальні дані та тестові дані у співвідношенні 70 до 30. Також було вивчено час навчання нейронної мережі та кількість часових проходів, необхідних для правильного навчання. Результати основного експерименту наведені в таблиці 2.2 [5, 16].

Таблиця 2.2 [5, 16]

Номер експерименту	Тип експерименту	Час навчання, с	Кількість часових проходів, %	Відношення кількості помилок, %
1	Окремі слова	1920	125	4.2
2	безперервна мова (від 3 до 7 слів)	2658	317	9.7
3	безперервна мова (від 6 до 12 слів)	8280	428	11.3
4	безперервна мова (від 10 до 20 слів)	10.065	641	14.8

2.3. Розробка альтернативного методу декодування мовного сигналу згортковою нейронною мережею

Запропонований та успішно реалізований метод кодування та декодування мовного сигналу для розпізнавання ЗНМ науковцями з Технологічного університету Ченстохова створює можливість для розширення його реалізації, а саме застосування підходу сприйняття мовного сигналу через розпізнавання зображень його параметрів. Даний метод частково повторює

роботу звичайного вокодеру типу LPC-10, але використовує для трансформації сигналу ЗНМ. На рисунку 2.11 зображено загальну схему розпізнавання мовного сигналу, який передбачає виділення зображень часової згортки, спектральної згортки та ЗСК.

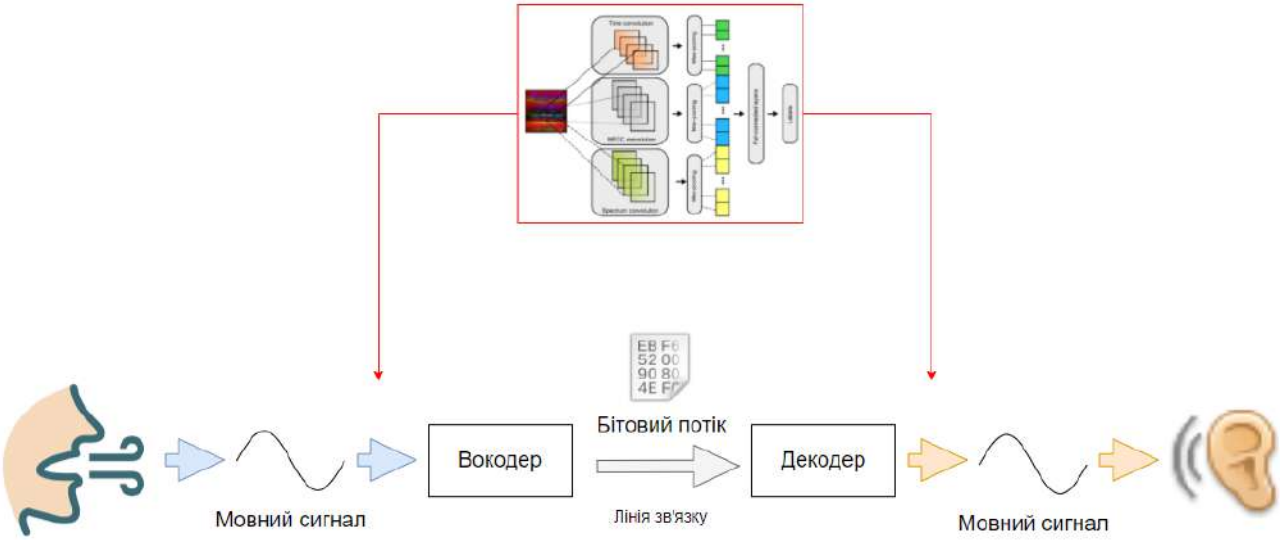


Рисунок 2.11 – Загальна схема методу кодування та декодування мовного сигналу

На зазначеній вище схемі спостерігається, що для реалізації запропонованого методу необхідно подавати мовний сигнал перед вокодером для кодування та після декодера для декодування. Такими діями, фактично, підмінюється роль вокодера (декодера), чим розширюється можливість обробки мовного сигналу для його кращого сприйняття.

На рисунку 2.12 зображено альтернативний метод декодування мовного сигналу. В основі цього методу лежить сприйняття системою інформації після вокодеру, а саме бітового потоку мовного сигналу. Подібний підхід можливо використовувати не тільки для вокодерної системи типу LPC-10, а також для будь-якої системи де порядок біт наслідує певну систематичність. Людина, аналізуючи подібні зображення, може прийняти рішення про інформаційне навантаження в каналі зв’язку за загальним зображенням бітового потоку, напряму не декодуючи кожен біт. Схожий принцип можливо імплементувати для систем розпізнавання зображень.

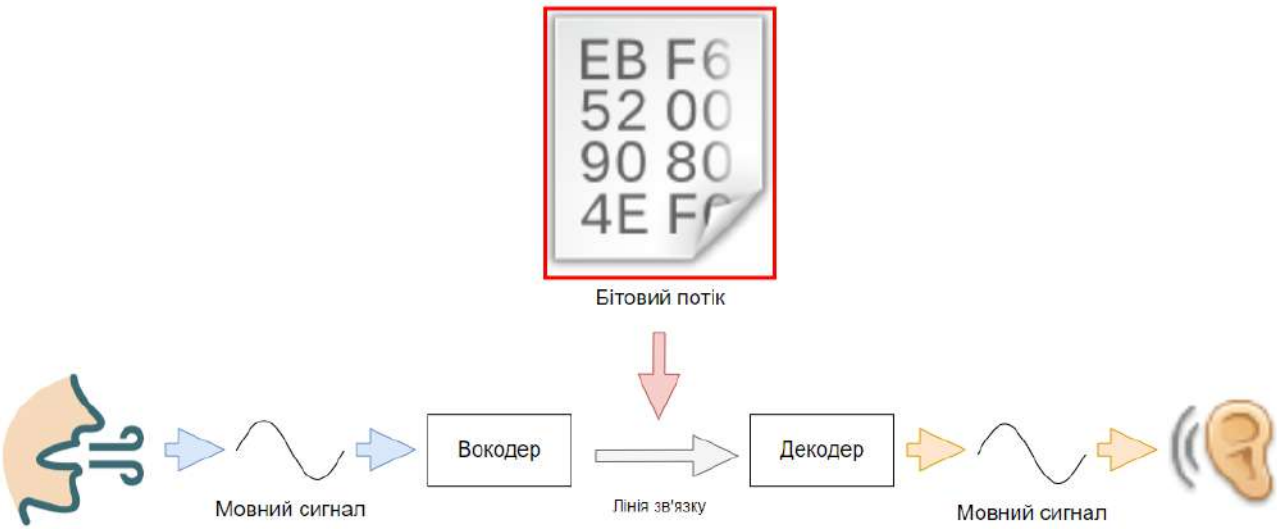


Рисунок 2.12 – Загальна схема альтернативного методу декодування мовного сигналу

На виході вокодеру ми отримуємо бітовий потік з фіксованим періодом. Кожен біт відповідає за відповідний кодований параметр (тон, рівень збудження, коефіцієнти фільтра передбачення, тощо) та знаходиться на фіксованій позиції, що передбачає певну систематичність зображення даного бітового потоку. Загальний вигляд бітового потоку та схема розташування відповідних параметрів на ньому зображено на рисунку 2.13.

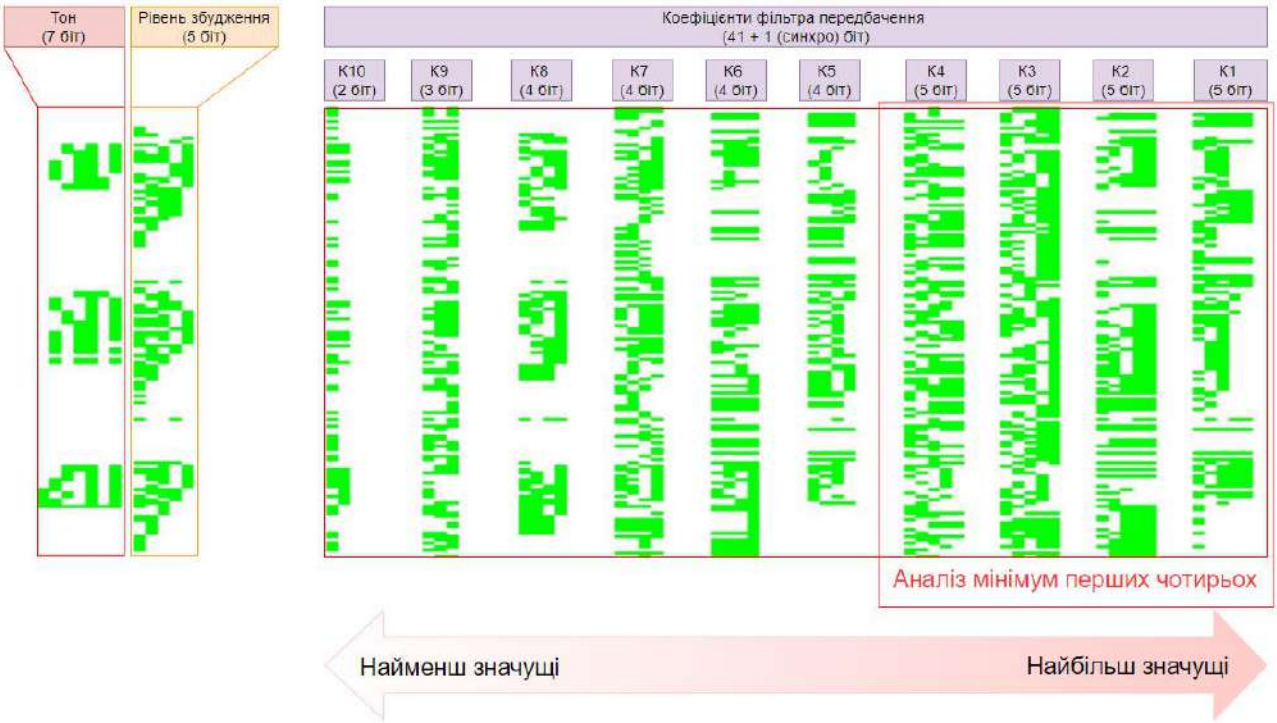


Рисунок 2.13 – Загальний вигляд бітового потоку та схема розташування відповідних параметрів мовного сигналу

Як видно з рисунку 2.13, параметри розташовуються в наступному порядку: 7 біт тону мовного сигналу, 5 біт рівня збудження, коефіцієнти фільтра передбачення – К1-К4 – 5 біт, К5-К8 – 4 біта, К9 – 3 біта, К10 – 2 біта. Загальна сума складає 53 біта, але в системах типу LPC-10 також передбачається один біт синхронізації, який не враховується в даній роботі. При аналізі вокодерного фрейму найбільш значущими є перші чотири коефіцієнти фільтру передбачення (К1-К4), таким чином, для того щоб отримати розбірливу мову, достатньо подати на вхід вокодера тон, рівень збудження та перші чотири коефіцієнти. Звісно, мова буде звучати неприродньо і по ній буде важко визначити кому вона належить, але зміст буде зрозумілим.

Для того щоб адаптувати метод розпізнавання мовного сигналу за бітовим потоком до методу розпізнавання за зображенням спектру було розроблено архітектуру, яка зображена на рисунку 2.14.

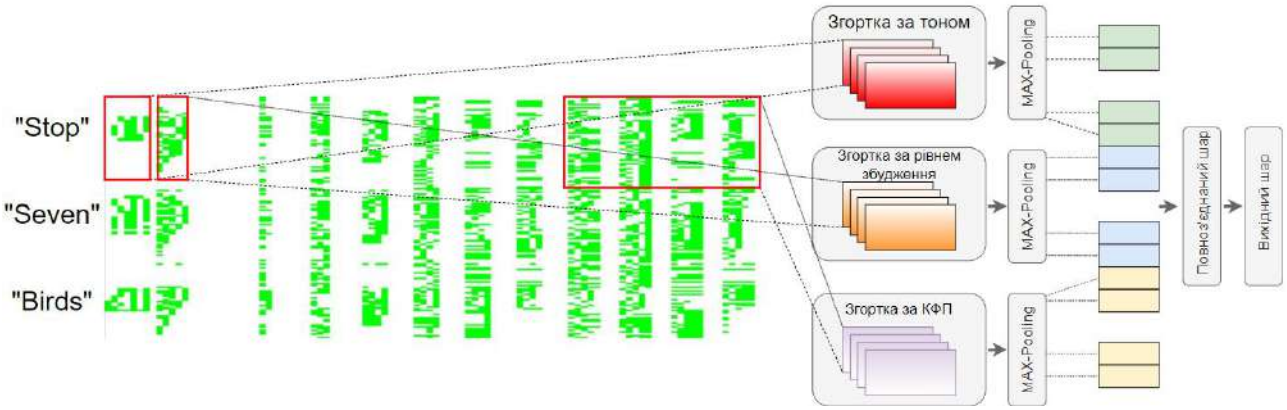


Рисунок 2.14 – Архітектура альтернативного методу декодування мовного сигналу ЗНМ

В основі даної архітектурі лежить описаний вище метод кодування та декодування мовного сигналу професора Кубанека. Але вхідними параметрами для ЗНМ є зображення бітового потоку. В даному альтернативному методі також можливе застосування кольорових каналів для розділення параметрів, що підвищує здатність ЗНМ диференціювати зображення.

РОЗДІЛ 3 Реалізація прототипу програмного застосунку на базі архітектури альтернативного методу декодування мовного сигналу згортковою нейронною мережею

3.1 Підготовка вхідних даних для прототипу програмного застосунку

Для проведення даної роботи було обрано набір аудіофайлів, де кожен файл представлено англійською мовою довжиною в 1 секунду та розширенням типу «.wav». Фали зібрані з усього світу та були записані в різних форматах з подальшим переформатуванням в єдиний формат (16-bit little-endian PCM-encoded WAVE) з частотою дискретизації 16000 кГц. Кожне слово отримано від різних людей при записі коротких мовних команд [17]. Загальна кількість окремих слів становить 35, для кожного з яких створена директорія з відповідною назвою («backward», «bed», «bird», «cat», «dog», «down», «eight», «five», «follow», «forward», «four», «go», «happy», «house», «learn», «left», «marvin», «nine», «no», «off», «on», «one», «right», «seven», «sheila», «six», «stop», «three», «tree», «two», «up», «visual», «wow», «yes», «zero»). Загальна кількість файлів для всіх слів становить 96099. Набір даних наданий в публічний доступ на платформі Tensorflow [18] з ліцензією Creative Commons BY 4.0 license.

Зазначений вище набір даних використовується для систем розпізнавання мови, подібних до тих, які зазначені в розділі 1.2. Вони отримують на вхід аудіосигнал і за допомогою перетворення Фур’є аналізують значення його параметрів, які лягають в основу розпізнавання. Для реалізації архітектури зазначеної в розділі 2.3 на рисунку 2.12, з мовного сигналу треба отримати зображення його параметрів у вигляді бітового потоку. На рисунку 3.1 зображено схему отримання зображення бітового потоку зі звичайного аудіофайлу.

В зазначеній схемі вхідний аудіофайл з розширення «.wav» подається на вхід вокодера LPC-10, відповідно до стандарту FED-SDT-1015 [11, 12].

Рисунок 3.1 – Схема отримання зображення бітового потоку зі звичайного аудіофайлу

На виході вокодера ми отримуємо бітовий потік, який використовується для передачі в різних лініях зв'язку та в різних протоколах в складі транспортного рівня моделі OSI (Open Systems Interconnection). За умови застосування кодованих даних приймаючою стороною відповідно до стандарту FED-SDT-1015, в лінії зв'язку не здійснюється ніяких додаткових дій щодо бітового потоку.

В нашому випадку, для реалізації запропонованої архітектури, стандартний вихід бітового потоку потребує попереднього деінтерлівінгу. Відповідно до стандарту FED-SDT-1015, вокодер, після складання вокодерного фрейму, здійснює його інтелівінг (перемішування), це необхідно для того, щоб підвищити завадостійкість сигналу в лініях зв'язку. Інтерлівінг здійснюється за відомою схемою для передавача і приймача, але проміжний вигляд сигналу не має візуального системного сприйняття, це означає, що на вокодерному фреймі важко виділити мовну та немовну частини. Схема інтерлівінгу зображена в таблиці 3.1.

Таблиця 3.1

Біт	Мовний	Немовний	Біт	Мовний	Немовний	Біт	Мовний	Немовний
1	К(1)-0	К(1)-0	19	К(3)-3	К(3)-3	37	К(8)-1	РЗ-6*
2	К(2)-0	К(2)-0	20	К(4)-2	К(4)-2	38	К(5)-1	К(1)-6*
3	К(3)-0	К(3)-0	21	РЗ-3	РЗ-3	39	К(6)-1	К(2)-6*
4	Т-0	Т-0	22	К(1)-4	К(1)-4	40	К(7)-2	К(3)-7*
5	РЗ-0	РЗ-0	23	К(2)-3	К(2)-3	41	К(9)-0	К(4)-6*
6	К(1)-1	К(1)-1	24	К(3)-4	К(3)-4	42	Т-5	Т-5
7	К(2)-1	К(2)-1	25	К(4)-3	К(4)-3	43	К(5)-2	К(1)-7*

8	К(3)-1	К(3)-1	26	РЗ-4	РЗ-4	44	К(6)-2	К(2)-7*
9	Т-1	Т-1	27	Т-3	Т-3	45	К(10)-1	Н/в
10	РЗ-1	РЗ-1	28	К(2)-4	К(2)-4	46	К(8)-2	РЗ-7*
11	К(1)-2	К(1)-2	29	К(7)-0	К(3)-5*	47	Т-6	Т-6
12	К(4)-0	К(4)-0	30	К(8)-0	РЗ-5*	48	К(9)-1	К(4)-7*
13	К(3)-2	К(3)-2	31	Т-4	Т-4	49	К(5)-3	К(1)-8*
14	РЗ-2	РЗ-2	32	К(4)-4	К(4)-4	50	К(6)-3	К(2)-8*
15	Т-2	Т-2	33	К(5)-0	К(1)-5*	51	К(7)-3	К(3)-8*
16	К(4)-1	К(4)-1	34	К(6)-0	К(2)-5*	52	К(9)-2	К(4)-8*
17	К(1)-3	К(1)-3	35	К(7)-1	К(3)-6*	53	К(8)-3	РЗ-8*
18	К(2)-2	К(2)-2	36	К(10)-0	К(4)-5*	54	Синхро	Синхро

Примітки

1 К – Коефіцієнт фільтра передбачення.

2 Т – тон.

3 РЗ – рівень збудження.

4 * – Біт контролю помилок.

5 Біт 0 – молодший біт параметру.

6 Біт 5 – молодший біт параметру контролю помилок.

7 Немовний – немовний або як такий, що знаходиться в процесі переходу з немовного в мовний стан.

8 Синхро – біт синхронізації, який чергується у фреймах між 0 та1.

Використовуючи таблицю інтерлівінгу ми формуємо вектор фрейму з наступним розташуванням параметрів: тон, рівень збудження, коефіцієнти фільтру передбачення (К10-К1). В реалізації відповідно до стандарту деінтерлівінг здійснюється на стороні декодера перед процедурою передбачення. В даному випадку, деінтерлівінг необхідно зробити до декодера, так як ми частково підмінюємо його функціонал роботою ЗНМ. На рисунку 3.2 зображено структуру фрейму в бінарному представленні після деінтерлівінгу. Згідно з даним рисунком, можливо зауважити, що кожен параметр, який був описаний вище, займає кількість біт меншу від розміру байту, тобто 8. Це означає, що візуально

створюються пробіли, які небажано використовувати для навчання та використання ЗНМ. Також, коефіцієнти фільтра передбачення розміщенні в зворотному порядку, з 10 по 1, де спочатку розташовані найменш значущі параметри після яких йдуть найбільш значущі параметри. Для побудови зображення, яке буде оптимально використовуватись ЗНМ було прийнято рішення зміни порядок розташування коефіцієнтів з 1 по 10.

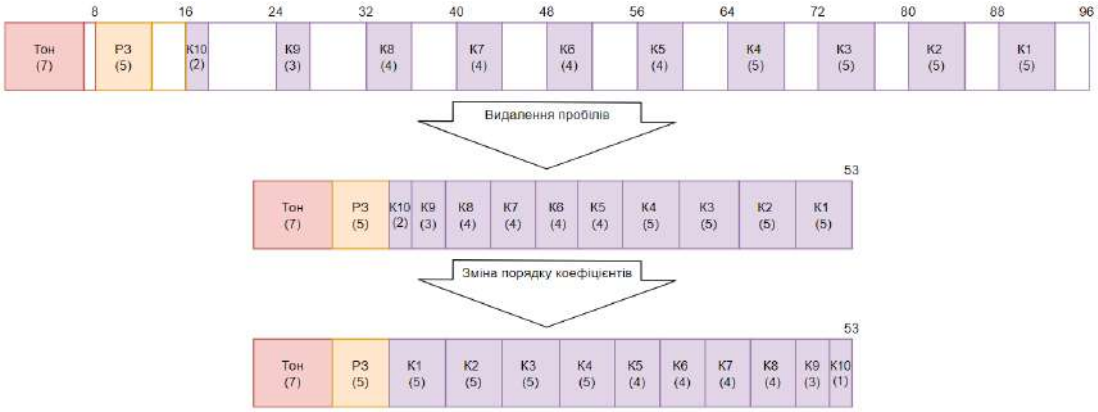


Рисунок 3.2 – Процес адаптації фрейму до ЗНМ

Враховуючи довжину кожного запису (~1 секунда), кількість фреймів становить від 89 до 91, таким чином, було зафіксовано значення для всіх згенерованих зображень, що становить 90. На рисунку 3.3 представлено приклад зображення, яке подається на вхід ЗНМ.

Рисунок 3.3 - Приклад зображення, яке подається на вхід ЗНМ

Приклад програмного застосунку, який генерує зображення для ЗНМ, відповідно до схеми наведеної на рисунку 3.4 надано в додатку А.

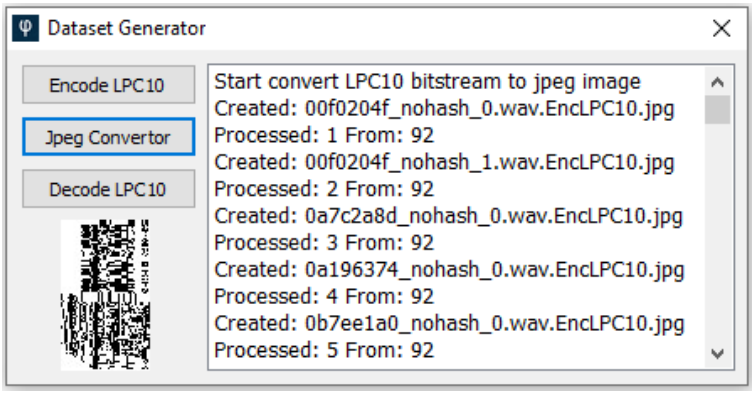


Рисунок 3.4 – Зовнішній вигляд програмного застосунку генерації зображень

3.2 Реалізація прототипу програмного застосунку

Для реалізації прототипу програмного застосунку на основі запропонованої архітектури альтернативного методу декодування мовного сигналу (рис. 2.14) було вирішено використовувати середу розробки PyCharm 2021.1.1 (Community Edition) від компанії JetBrains s.r.o., відповідно мову програмування Python 3.7, бібліотеку машинного навчання TensorFlow 2.4.1 від компанії Google та відкриту нейромережну бібліотеку Keras. Вихідний код прототипу програмного застосунку наведено в додатку Б.

Не першому етапі здійснюється підготовка датасету для навчання та валідації моделі. Для цього використовуються файли зображень, які були підготовлені відповідно до рисунку 3.2 зображеному в розділі 3.1. З бібліотеки Keras [19] застосовується метод image_dataset_from_directory [20], який генерує датасет з файлів визначеної директорії. Перед початком роботи файли мають бути розташовані відповідно структури зображеної на рисунку 3.5.

```
Загальна директория/
...клас_a/
......a_зображення_1.jpg/
......a_зображення_2.jpg/
...клас_b/
......b_зображення_1.jpg/
......b_зображення_2.jpg/
...клас_n/
......n_зображення_1.jpg/
......n_зображення_2.jpg/
```

Рисунок 3.5 – Структура організації файлів датасету

На вхід методу image_dataset_from_directory необхідно подати відповідні параметри для коректної роботи моделі, а саме: labels, label_mode, color_mode, batch_size, image_size, shuffle, seed, validation_split та subset.

Для параметру labels ми визначаємо значення inferred, це означає, що найменування класів (labels) буде підібрано відповідно до значень піддиректорій.

Параметр label_mode очікує на вхід тип, який описує найменування піддиректорій (int – кодування назв в цілочисельнуму значенні, binary – 0,1, categorical – кодування вектором значенння, тощо). В нашому випадку ми використали кодування вектором значення – categorical, для того щоб краще орієнтуватися в назвах підкласів в процесі роботи моделі.

color_mode – в даному випадку можливі варіанти застосування greyscale (чорно-біле зображення), rgb (формат RGB), rgba (формат RGBA). Ми застосовуємо greyscale, так як фактично конвертуємо значення нулів та одиниць бітового потоку в піксельне зображення, де 0 – білий колір (фон), 1 – чорний колір.

batch_size – кількість вхідних файлів, які використовуються за один навчальний прохід моделі. Враховуючи те, що кількість вхідних відносно невелика (105829), ми будемо використовувати загальноприйняте значення в 32 файли (batchs).

image_size – значення, що описує розмір вхідного зображення, в нашому випадку це 90 на 53. Якщо ці показники відрізняються від реального розміру зображення, необхідно здійснити конвертацію (reshape).

shuffle – перемішування в довільному порядку. Визначено як True, у випадку False, дані будуть відсортовані в алфавітному порядку.

seed – задається початкове значення для функції генерації випадкових чисел.

validation_split – відсоток даних, які виділені для безпосередньої валідації та навчання, в нашому випадку, це 10%.

subset – відповідність навчального датасету або датасету валідації. Для навчального датасету ми вказали trainin та для датасету валідації validation відповідно.

На рисунку 3.6 зображено результат формування датасету

```
Found 96099 files belonging to 35 classes.
Using 86490 files for training.
Found 96099 files belonging to 35 classes.
Using 9609 files for validation.
['backward', 'bed', 'bird', 'cat', 'dog', 'down', 'eight', 'five', 'follow', 'forward', 'four', 'go', 'happy',

'learn', 'left', 'marvin', 'nine', 'no', 'off', 'on', 'one', 'right', 'seven', 'sheila', 'six', 'stop',

'stop', 'three', 'tree', 'two', 'up', 'visual', 'wow', 'yes', 'zero']
```

Рисунок 3.6 - Результат формування датасету

На другому етапі необхідно підготувати тестову модель для декодування мовного сигналу. Для цього будемо використовувати послідовну тип моделі бібліотеки Keras – Sequential model [21]. До складу даної моделі входять наступні шари: вхідний шар (Input), згортковий шар (Conv2D), агрегувальний шар (MaxPooling2D), шар стиснення (Flatten), повнозєднаний шао (Dense). Схематично зазначена структура зображена на рисунку 3.7.

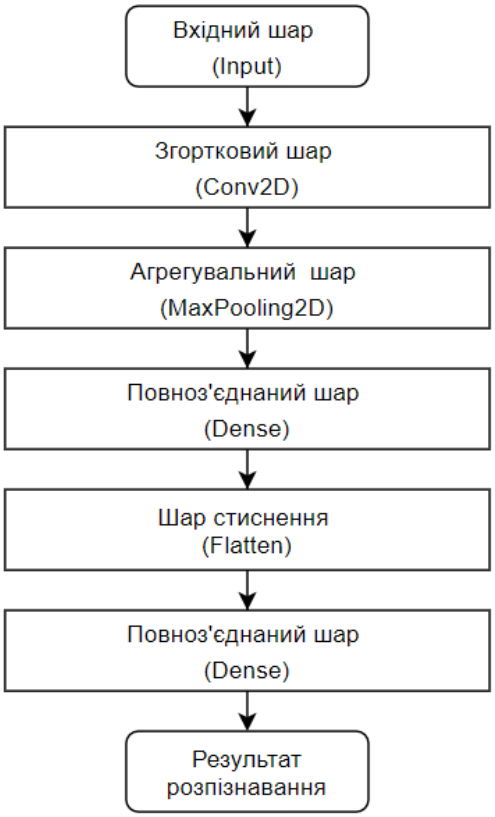


Рисунок 3.7 – Схематичне зображення згорткової нейронної моделі

Шар Input приймає на вхід зображення з відповідними параметрами shape. Як і зазначалось раніше, ми передаємо значення висоти, ширини та кількості кольорових каналів (90 на 53 на 1). Наступним йдуть два згорткових шари

Conv2D, де першим вхідним параметром є кількість вихідних каналів, для даної моделі, це 16 і 32 відповідно. Всі подальші параметри, як для першого згорткового шару, так і для другого є однаковими. Наступним параметром є kernel_size, що фактично описує висоту и ширину вікна, яке рухається по зображенню для здійснення згортки, значення даної моделі становлять 3 на 3. Параметр padding (відступ), застосовується для вирівнювання зображення по висоті або ширині. В нашому випадку ми залишаємо зображення без змін, відповідно передаємо по даному аргументу значення same. Далі ми використовуємо агрегувальний шар MaxPooling2D, який використовує максимальне значення з кожного з кластерів нейронів попереднього шару відповідно до висоти та ширини пулу. За замовченням в бібліотеці встановлені значення 2 на 2, що і використовується для даної моделі. На наступному кроці використовується шар стиснення (Flatten), цей шар використовується, як зв'язка між агрегувальним шаром та наступним повноз'єднаним шаром. Останнім в даному випадку шаром є повнозєднаний шар (Dense), який з'єднаний з усіма збудженнями попердного шару. На вхід даного шару передається значення вихідного шару, що дорівнює 10.

На третьому етапі здійснюється конфігурацію моделі. Для цього необхідно викликати метод compile [22] моделі отриманої на другому етапі і передати на вхід наступні параметри: optimizer, loss, metrics.

optimizer – потимізатар, параметр, що використовується для оновлення моделі враховуючи поточні дані та функцію втрат. В нашому випадку обраний оптимізатор Adam. Зазначений оптимізатор є алгоритмом стохастичних цільових функцій на основі градієнта першого порядку, заснований на адаптивних оцінках моментів нижчого порядку. Даний метод простий в реалізації, є обчислювально ефективним, не має особливих вимог до пам'яті, є інваріантним до діагонального масштабування градієнтів і добре підходить для роботи з великим масивом даних. Метод також підходить для градієнтів з високим показником шуму та розрідження. [23].

loss – функція втрат, яка допомагає визначити точність моделі під час навчання. Ціллю даної функції, є зведення її значення до мінімуму, що визначає

успішність навчання. Для нашої моделі ми використовуємо значення SparseCategoricalCrossentropy (розріджену крос-ентропію визначену за категоріями), вона вираховує крос-ентропію між заданими класами (labels) та передбаченнями.

metrics – значеня метрики, яке використовується для моніторингу процесу навчання та тестування. В нашому випадку передається загально прийняте значення accuracy (точність).

На четвертому етапі ми аналізуємо конфігурацію моделі, для цього використовується метод summary() [24]. На рисунку 3.8 зображено приклад конфігурації тестової моделі.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential (Sequential)      (None, 90, 53, 3)         0
_________________________________________________________________
rescaling (Rescaling)        (None, 90, 53, 3)         0
_________________________________________________________________
conv2d (Conv2D)              (None, 90, 53, 16)        448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 45, 26, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 45, 26, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 22, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 22, 13, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 6, 64)         0
_________________________________________________________________
dropout (Dropout)            (None, 11, 6, 64)         0
_________________________________________________________________
flatten (Flatten)            (None, 4224)              0
_________________________________________________________________
dense (Dense)                (None, 128)               540800
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 565,029
Trainable params: 565,029
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.8 – Приклад конфігурації тестової моделі

На п’ятому етапі необхідно безпосередньо здійснити навчання моделі підготовленим раніше датасетом. Для цього з зазначеної вище бібліотеки Keras викликається метод fit [25]. На вхід даного методу передається вказівник на датасет, кількість проходів моделі вздовж датасете (epoches) та рівень деталізації логування процесу навчання (verbose).

3.3 Аналіз результатів застосування прототипу програмного застосунку

Для перевірки доцільності використання запропонованої архітектури, першочергово булу заплановано порівняти результати розпізнавання відповідно до методу запропанованого професором Кубанеком [16] та результати альтернативного методу запропонованого нами, але вихідні дані на яких будувався датасет оригінального методу знаходиться на серверах Технологічного університету Ченстохова, до яких ми не маємо доступу. Тому було вирішено використати датасет побудований на основі вихідних даних, походження яких описано в розділі 3.1. Експеримент проводився в два етапи. На першому етапі визначалася оптимальна комбінація шарів та параметрів, які передаються на їх вхід. Для розрахунку оптимальної конфігурації мережі ми відслідковували значення точності (accuracy) та втрат (loss) протягом процесу навчання. Для отримання візуального представлення були побудовані графіки точності та втрат для навчальних (training) даних та даних валідації (validation). Відповідні графіки точності та втрат зображені на рисунку 3.9.

З наведених нижче графіків видно, що точність навчання моделі збільшується відповідно до збільшення кількості проходів, але ця залежність різна для різних класів слів. На рисунках 3.10 та 3.11, на який зображено графік залежності точності визначення типу класу від кількості проходів та графік середньої лінійної залежності типу класу від кількості проходів видно, що є класи для яких показник точності майже не змінюється в процесі збільшення проходів, це дозволяє оптимізувати ресурси необхідні в процесі навчання моделі. Також відсутні значні прогалини між навчальними значеннями та значеннями валідації, що говорить про те, що в більшості випадків модель не має ознак перенасичення

(overfitting). Для даного випадку значення точності досягає 80 %, що є прийнятним показником і дозволяє використовувати модель для подальшої класифікації даних. Показники втрат мають обернений характер, тобто їх значення зменшуються відповідно до кількості проходів та в даному випадку становлять менше 2 %.

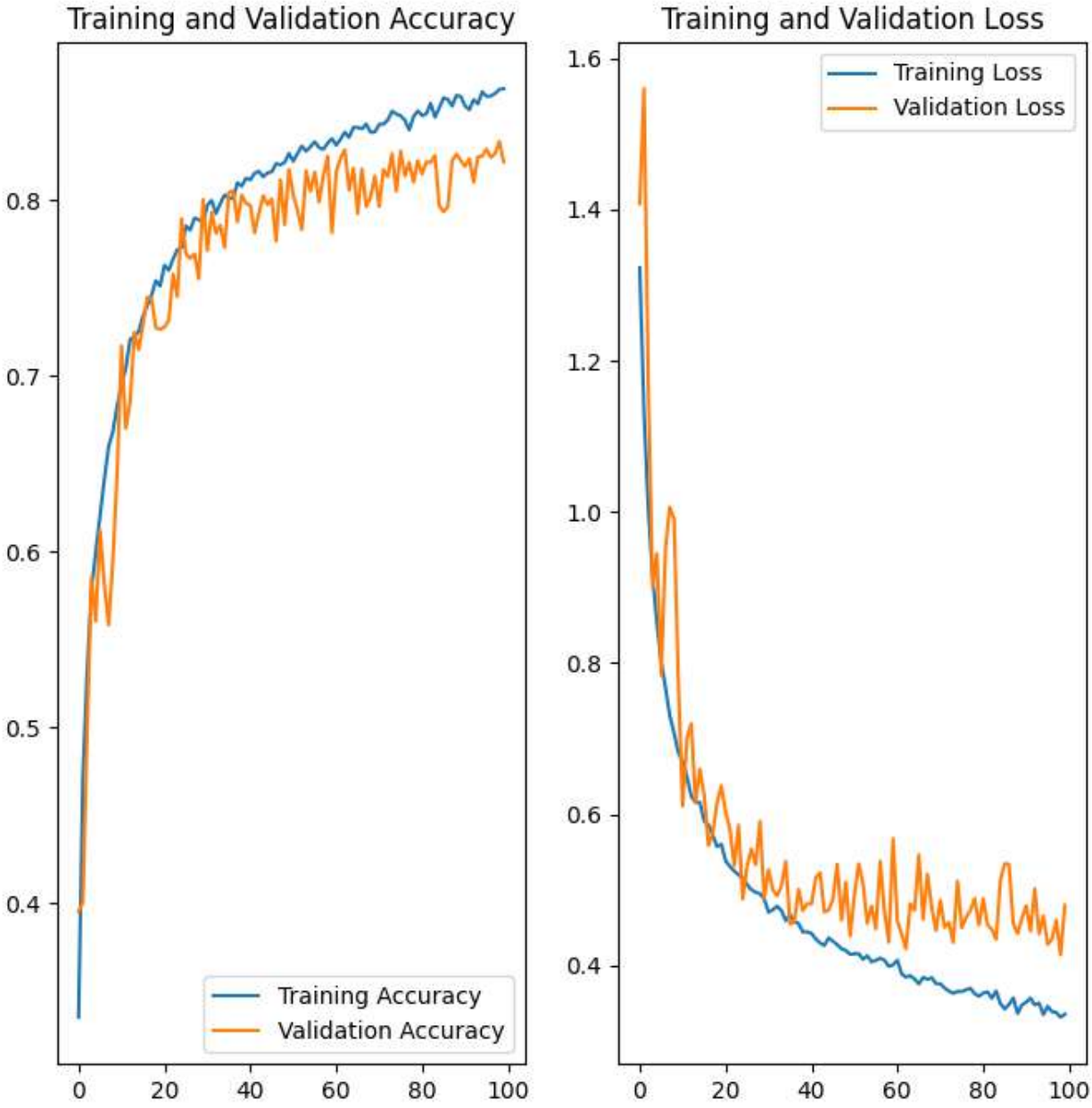


Рисунок 3.9 – Графіки точності та втрат програмного застосунку

На другому етапі було визначено точність визначення вхідного мовного сигналу для кожного класу. Також, додатково проаналізовано залежність точності визначення належності мовного сигналу до класу відповідно до кількості проходів моделі (таблиця 3.1).

Таблиця 3.1

Значення / Клас	Кількість	Точність	Кількість	Точність	Кількість	Точність
backward	50	95.44	100	89.64	150	84.18
bed	50	62.86	100	60.19	150	30.57
bird	50	30.79	100	47.86	150	31.73
cat	50	93.98	100	96.23	150	99.42
dog	50	85.12	100	61.51	150	86.75
down	50	45.23	100	72.12	150	75.16
eight	50	54.74	100	87.79	150	84.40
five	50	96.14	100	96.34	150	96.41
follow	50	34.45	100	45.96	150	62.37
forward	50	59.93	100	42.17	150	49.28
four	50	42.85	100	89.47	150	90.21
go	50	45.50	100	53.38	150	76.24
happy	50	99.91	100	98.89	150	100.00
house	50	95.64	100	98.58	150	99.99
learn	50	51.29	100	96.74	150	86.67
left	50	90.11	100	50.13	150	95.26
marvin	50	39.27	100	46.60	150	62.96
nine	50	59.05	100	89.72	150	50.61
no	50	69.05	100	79.22	150	94.77
off	50	93.66	100	91.50	150	83.09
on	50	71.22	100	86.33	150	78.29
one	50	63.32	100	71.36	150	86.73
right	50	89.12	100	99.17	150	57.79
seven	50	59.63	100	71.36	150	97.08
sheila	50	68.42	100	70.24	150	83.76
six	50	99.40	100	99.95	150	99.92
stop	50	35.21	100	57.73	150	59.87
three	50	56.92	100	54.41	150	44.41
tree	50	98.41	100	90.54	150	86.11
two	50	44.04	100	73.54	150	73.93
up	50	41.20	100	52.43	150	47.45
visual	50	88.13	100	91.53	150	97.92
wow	50	87.69	100	89.33	150	90.83
yes	50	89.65	100	90.36	150	99.89
zero	50	91.17	100	94.24	150	99.96

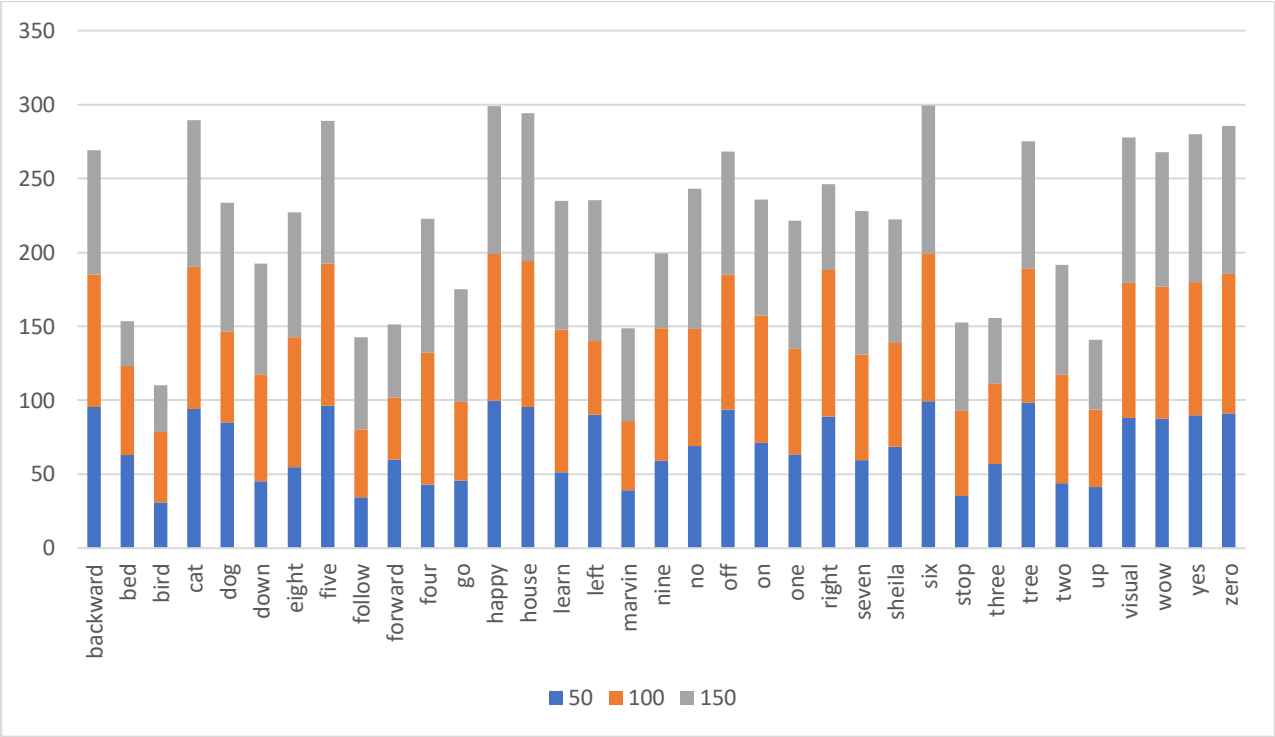


Рисунок 3.10 – Графік залежності точності визначення типу класу від кількості проходів

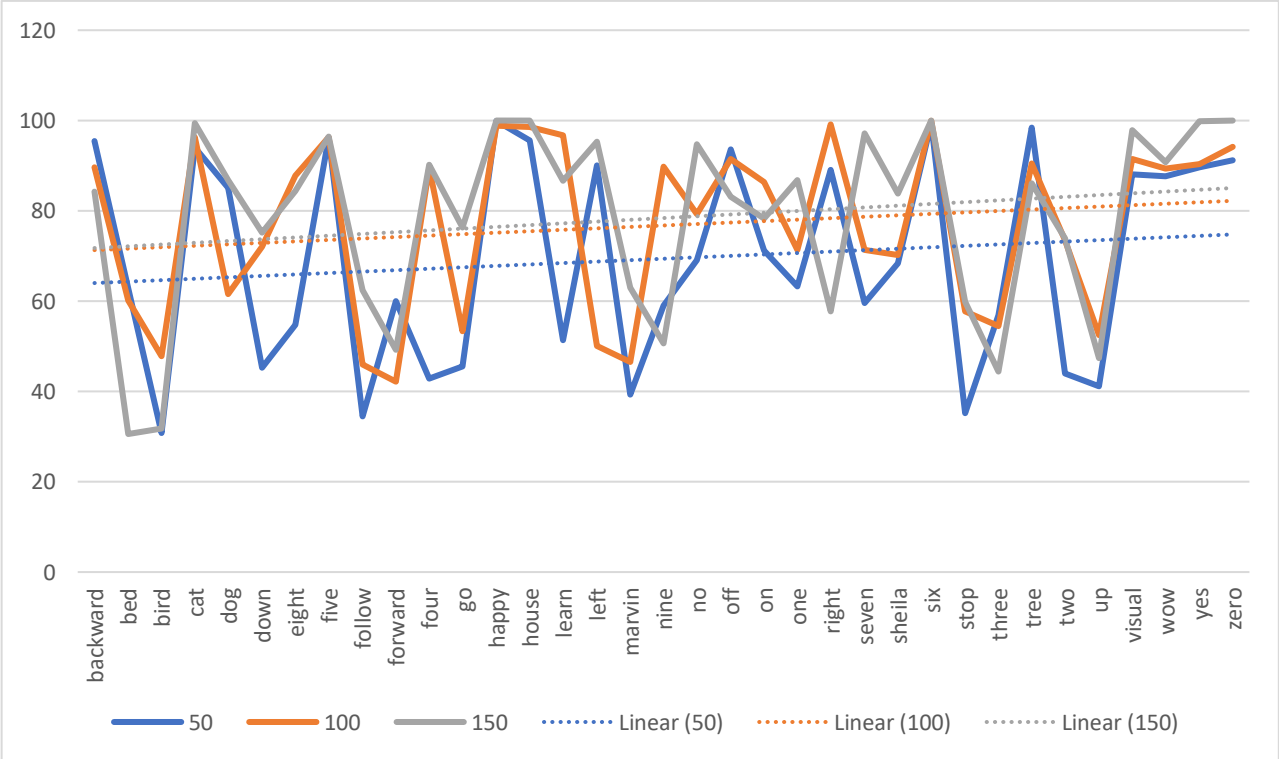


Рисунок 3.11 – Графік середньої лінійної залежності типу класу від кількості проходів

Так як обчислення проводились не на спеціалізованому сервері для обрахунку моделі, а в умовах персонального комп’ютера (ноутбуку) з параметрами операційної системи Windows 10 Pro 64 bit, процесором Intel Core i7-8750H CPU 2.2 ГГц оперативною пам’яттю 16 ГГц, кількість навчальних проходів становить 50, 100 та 150.

Аналізуючи отримані результати можна зазначити, що середній показник точності визначення належності мовного сигналу до відповідного класу досягає 80 %, що є прийнятним. В даному випадку модель використовувалась для розпізнавання мовного сигналу у вигляді одичного слова, це означає, що безперервна мовна послідовність та словосполучення є предметом подальших досліджень. Для поточного експерименту, на жаль, не було можливості отримати необхідну вибірку даних у вигляді безперервної мови або словосполучень від різних людей. Також, поточний результат можливо покращити за рахунок збільшення вибірки даних одиночних слів, що і відбулось для датасету, який використовується в даній роботі, де друга версія збірки даних має приблизно в два рази більше мовних записів. Дана модель побудована для розпізнавання всіх параметрів мовного сигналу у вигляді бітового потоку на одному зображенні. За умови реалізації розділення параметрів на різні потоки для розпізнавання, можливе підвищення значення точності розпізнавання. Перша ітерація з навчанням моделі, де параметри мовного сигналу знаходяться на одному зображені, показала прийнятні результат, що є підставою для подальших досліджень та реалізації запропонованої архітектури з розділенням параметрів, яка зазначення на рисунку 2.14. Якщо порівнювати отримані результати з результатами дослідження професера Кубанека, метод якого було обрано за основу дослідження, можна зазначити, що в частині розпізнавання окремих мовних слів, наш метод також має точність визначення для деяких класів вище 98 %, що є підставою для подальшого розширення дослідження та порівняння результатів для ширшої групи слів, словосполучень та безперевної мови.

Висновки по роботі та рекомендації для подальших досліджень

В першому розділі роботи було проаналізовано загальні поняття ЗНМ. Було надано визначення ЗНМ та розглянуто принципи її роботи, які в процесі розробки архітектури неодноразово використовувались. Також проаналізовано використання ЗНМ для розпізнавання мовного сигналу, що сформувало розуміння існуючих систем, а саме їх структуру, переваги та недоліки. Для реалізації основної архітектури було описано відомості про кодування з лінійним передбаченням. Дана частина важлива для подальшої роботи з вокодером типу LPC-10, який працює за принципом лінійного передбачення та обраний, як основний інструмент отримання бітового потоку кодованого мовного сигналу.

В другому розділі здійснено аналіз архітектурних реалізацій розпізнавання мовного сигналу використовуючи зображення його параметрів. Даний підхід є нестандартним для подібного типу розпізнавання, тому було важливо зрозуміти існуючі реалізації та визначити їх переваги та недоліки. Проаналізовано метод кодування та декодування мовного стгналу ЗНМ розроблений професором Кубанеком з Технологічного університету Ченстохова, який став основою для побудови альтернативного методу декодування мовного сигналу ЗНМ. В нашому випадку ми не використовуємо процес кодування, а тільки декодуємо (розпізнаємо) з використанням зображення бітового потоку на відміну від оригінального методу, де кодування та декодування здійснюється на основі спектрального зображення мовного сигналу. Як результат, було розроблено архітектуру зазначеного альтернативного методу декодування мовного сигналу ЗНМ, яка лягла в основу майбутнього прототипу програмного застосунку.

В третьому розділі було описано процес підготовки вхідних даних для прототипу програмного застосунку. Даний етап є однією з найважливіший частин дослідження, так як від нього залежить ефективність роботи моделі. Було визначено основні характеристики та параметри вокодеру LPC-10, який використовується, як основний інструмент кодування мовного сигналу та розроблено схему перетворення мовного сигналу в зображення бітового потоку.

Як результат, розроблено окремий програмний застосунок для реалізації зазначеної схеми та підготовки датасету, який відповідає вимогам розробленої архітектури декодування мовного сигналу ЗНМ.

Було реалізовано прототип програмного застосунку декодування мовного сигналу ЗНМ на основі попередньо розробленої архітектури. Визначено та описано основні складові частини моделі та необхідні параметри для роботи з нею. Також проаналізовано та порівняно результати роботи моделі в різних умовах.

Таким чином, в рамках даного дослідження було проаналізовано загальні поняття ЗНМ, визначено підходи до розпізнавання зображень та розпізнавання мовних сигналів ЗНМ, розглянуто основні характеристики та параметри вокодеру LPC-10. Проаналізовано існуючі реалізації розпізнавання мови за зображеннями її характеристик та розроблено власну архітектру декодування мовного сигналу ЗНМ. Також розроблено два прототипи програмних застосунків для перевірки роботи архітектури в частині підотовки вхідного датасету та безпосередньої роботи моделі ЗНМ.

В цілому, визначений план роботи було виконано, отримані результати можуть бути використанні у подальших дослідженнях, а саме для реалізації архітектури з розділенням зображень основних параметрів мовного сигналу на різні вхідні канали, адаптації моделі для розпізнавання ширшої групи слів, словосполучень та безперевної мови.

Результати роботи можуть бути використані в телекомунікаційних системах з низької завадостійкістю де необхідно підвищувати якість мовного сигналу за рахунок додаткового розпізнавання або повноцінної заміни декодуючої сторони відповідно до запропонованої архітектури. Даний підхід корисний за умови, якщо в одному каналі зв’язку передаються вокодерні системи декількох типів (LPC-10, MELP, CELP, G.732 тощо). За умови втрати ідентифікатора типу вокодерної системи запропонована архітектура може декодувати мовний сигнал або визначити його відношення до відповідної системи.

Список літератури

1. Невмержицький Р.В Згорткові нейронні мережі. URL: https://conf.ztu.edu.ua/wp-content/uploads/2019/02/45-1.pdf (дата звернення: 15.11.2020).
2. Rawat W., Wang Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. URL: https://www.mitpressjournals.org/doi/pdfplus/10.1162/neco_a_00990 (дата звернення: 15.11.2020).
3. CS231n Convolutional Neural Networks for Visual Recognition. URL: https://conf.ztu.edu.ua/wp-content/uploads/2019/02/45-1.pdf (дата звернення: 15.11.2020).
4. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. URL: https://arxiv.org/pdf/1512.03385.pdf (дата звернення: 28.03.2020).
5. Редчиць Є.В. Альтернативний метод кодування та декодування мови згортковою нейронною мережею. URL: http://ekmair.ukma.edu.ua/bitstream/handle/123456789/18228/Redchyts_Alternatyvnyi_metod_koduvannia_ta_dekoduvannia_movy_zghortkovoiu_neironnoiu_merezheiu.pdf?sequence=1&isAllowed=y (дата звернення: 20.11.2020).
6. TensorFlow Lite Tutorial Part 1: Wake Word Feature Extraction. URL: https://www.digikey.com/en/maker/projects/tensorflow-lite-tutorial-part-1-wake-word-feature-extraction/54e1ce8520154081a58feb301ef9d87a (дата звернення: 26.11.2020).
7. Мельник А.О., Шевчук Р.П. Порівняльний аналіз алгоритмів стиснення мовних сигналів. URL: http://science.lpnu.ua/sites/default/files/journal-paper/2017/dec/7464/18.pdf (дата звернення: 26.11.2020).
8. Speech coding: a tutorial review. URL: http://spanias.faculty.asu.edu/papers/spanias94tutorial.pdf (дата звернення: 16.12.2020).
9. Сапожков М.А., Михайлов В.Г. Вокодерная Связь: Навч. посіб. Москва: Радио и связь, 1983.

10. Rabiner L., Schafer R. Digital Processing of Speech Signals. URL: https://asa.scitation.org/doi/pdf/10.1121/1.384160 (дата звернення: 20.04.2020).
11. Analog to digital conversion of voice by 2,400 bit/seconds linear predictive coding. URL: https://ptabdata.blob.core.windows.net/files/2017/IPR2017-01077/v23_Exhibit%201022%20-%20FS1015-LPC10.pdf (дата звернення: 20.12.2020).
12. Alwan A., Ozgu Ozun O., Steurer P., Thell D., Wideband Speech Coding With Linear Predictive Coding (LPC). URL: http://www.seas.ucla.edu/spapl/projects/ee214aW2002/1/report.html (дата звернення: 05.01.2021).
13. Douglas O'Shaughnessy. Speech communication: human and machine: Addison-Wesley, 1987. 150 с.
14. Speech Recognition: Key Word Spotting through Image Recognition. URL: https://arxiv.org/pdf/1803.03759.pdf (дата звернення: 25.12.2020).
15. Gradient-Based Acoustic Features for Speech Recognition. URL: http://www.me.cs.scitec.kobe-u.ac.jp/~takigu/pdf/2009/TP2-D-2.pdf (дата звернення: 25.12.2020).
16. Kubanek M, Janusz Bobulski J, Kulawik J A Method of Speech Coding for Speech Recognition Using a Convolutional Neural Network. URL: https://www.mdpi.com/2073-8994/11/9/1185/htm#B18-symmetry-11-01185 (дата звернення: 22.03.2020).
17. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. URL: https://arxiv.org/abs/1804.03209 (дата звернення: 28.12.2020).
18. Speech_commands_v0.02. URL: http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz (дата звернення: 28.12.2020).
19. Module: tf.keras. URL: https://www.tensorflow.org/api_docs/python/tf/keras (дата звернення 10.01.2021).
20. tf.keras.preprocessing.image_dataset_from_directory. URL: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image_dataset_from_directory (дата звернення: 11.01.2021).

21. tf.keras.Sequential. URL: https://www.tensorflow.org/api_docs/python/tf/keras/Sequential (дата звернення: 11.01.2021).
22. compile method. URL: https://keras.io/api/models/model_training_apis/ (дата звернення: 11.01.2021).
23. Adam: A Method for Stochastic Optimization. URL: https://arxiv.org/abs/1412.6980 (дата звернення: 11.01.2021).
24. summary method. URL: https://keras.io/api/models/model/ (дата звернення: 11.01.2021).
25. fit method. URL: https://keras.io/api/models/model_training_apis/ (дата звернення: 11.01.2021).

Додаток А. Програмний код підсистеми генерації зображень

```
unit Unit_MainForm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, UnitVHF_Vocoder, UnitVHF_VocoderProc;

type
  TForm1 = class(TForm)
    OpenDialog1: TOpenDialog;
    ListBox1: TListBox;
    EncodeLpc10: TButton;
    JpegConvertor: TButton;
    procedure EncodeLpc10Click(Sender: TObject);
    procedure JpegConvertorClick(Sender: TObject);
  private
    FHeaderNum: Integer;
  public
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.EncodeLpc10Click(Sender: TObject);
```

```
var
  FName: string;
  VHF_Vocoder_Proc: TVHF_Vocoder_Proc;
begin
  ListBox1.Clear;
  ListBox1.Items.Add('Start encoding LPC10');
  if OpenDialog1.Execute then
  begin
    VHF_Vocoder_Proc := TVHF_Vocoder_Proc.Create;
    for FName in OpenDialog1.Files do
    begin
      VHF_Vocoder_Proc.ProcEncNoHdrLPC10(FName);
    end;
    VHF_Vocoder_Proc.Free;
  end;
  ListBox1.Items.Add('Finished!');
end;

procedure TForm1.JpegConvertorClick(Sender: TObject);
var
  FName: string;
  VHF_Vocoder_Proc: TVHF_Vocoder_Proc;
begin
  ListBox1.Clear;
  ListBox1.Items.Add('Start convert LPC10 bitstream to jpeg image');
  if OpenDialog1.Execute then
  begin
    VHF_Vocoder_Proc := TVHF_Vocoder_Proc.Create;
    for FName in OpenDialog1.Files do
    begin
      VHF_Vocoder_Proc.JpegFromInterleaving(FName);
```

```
    end;
    VHF_Vocoder_Proc.Free;
  end;
  ListBox1.Items.Add('Finished!');
end;

end.

unit UnitVHF_VocoderProc;

interface

uses
  Windows, classes, SysUtils, UnitVHF_Vocoder, UnitLPC_Routine, jpeg, Graphics;

type
  TVHF_Vocoder_Proc = class
  private
    FIn, FOut: TFileStream;
    FOutLpc10_E: array[0..6] of Byte;
    FOutLpc10_D: array[0..179] of Word;
    const
      __iblist: array[0..52] of Byte =( //
        13, 12, 11, 1, 2, // R1-0, R2-0,  R3-0,   P-0,  A-0,
        13, 12, 11, 1, 2, // R1-1, R2-1,  R3-1,   P-1,  A-1,
        13, 10, 11, 2, 1, 10, // R1-2, R4-0,  R3-2,   A-2,  P-2, R4-1,
        13, 12, 11, 10, 2, // R1-3, R2-2,  R3-3,  R4-2,  A-3,
        13, 12, 11, 10, 2, // R1-4, R2-3,  R3-4,  R4-3,  A-4,
        1, 12, 7, 6, 1, 10, // P-3, R2-4,  R7-0,  R8-0,  P-4, R4-4,
        9, 8, 7, 4, 6, // R5-0, R6-0,  R7-1, R10-0, R8-1,
        9, 8, 7, 5, 1, // R5-1, R6-1,  R7-2,  R9-0,  P-5,
```

```
      9, 8, 4, 6, 1, 5, // R5-2, R6-2, R10-1,  R8-2,  P-6, R9-1,
      9, 8, 7, 5, 6 // R5-3, R6-3,  R7-3,  R9-2, R8-3, SYNC
      );
  public
    constructor Create();
    destructor Destroy(); override;
  public
    procedure ProcEncNoHdrLPC10(aPath: string);
    procedure ProcDecNoHdrLPC10(aPath: string);
    procedure JpegFromInterleaving(aPath: string);
  end;

implementation

{ TVHF_Vocoder_Proc }

constructor TVHF_Vocoder_Proc.Create;
begin
  InitVocoderProc();
end;

destructor TVHF_Vocoder_Proc.Destroy;
begin
  FreeVocoderProc();
end;

procedure TVHF_Vocoder_Proc.ProcEncNoHdrLPC10(aPath: string);
var
  VocBuf: PByte;
  BufPos: Integer;
  CErr: Integer;
```

```
  Size: Integer;
begin
  FIn := TFileStream.Create(aPath, fmOpenRead);
  if not DirectoryExists(ExtractFilePath(aPath) + 'EncLPC10') then
    CreateDir(ExtractFilePath(aPath) + 'EncLPC10');
  FOut := TFileStream.Create(ExtractFilePath(aPath) + 'EncLPC10\' +
ExtractFileName(aPath) + '.EncLPC10', fmCreate);
  VocBuf := GetMemory(FIn.Size);
  FIn.Read(VocBuf^, FIn.Size);
  FIn.Seek(44, 0);
  BufPos := FIn.Position;

  VC_InitVocoderParams();

  while (BufPos <= FIn.Size) do
  begin
    Size := VC_EncodeLpc10(@VocBuf[BufPos], @FOutLpc10_E[0], @CErr);
    FOut.Write(FOutLpc10_E[0], Size);
    Inc(BufPos, 360);
  end;
  FreeMemory(VocBuf);
  FreeAndNil(FIn);
  FreeAndNil(FOut);
end;

procedure TVHF_Vocoder_Proc.ProcDecNoHdrLPC10(aPath: string);
var
  VocBuf: PByte;
  BufPos: Integer;
  CErr: Integer;
  Size: Integer;
```

```
begin
  FIn := TFileStream.Create(aPath, fmOpenRead);
  FOut := TFileStream.Create(aPath + '.DecLPC10', fmCreate);
  VocBuf := GetMemory(FIn.Size);
  FIn.Read(VocBuf^, FIn.Size);
  BufPos := 0;

  VC_InitVocoderParams();

  while (BufPos <= FIn.Size) do
  begin
    Size := VC_DecodeLpc10(@VocBuf[BufPos], @FOutLpc10_D[0], @CErr);
    FOut.Write(FOutLpc10_D[0], Size * 2);
    Inc(BufPos, 7);
  end;
  FreeMemory(VocBuf);
  FreeAndNil(FIn);
  FreeAndNil(FOut);
end;

procedure TVHF_Vocoder_Proc.JpegFromInterleaving(aPath: string);
const
  VektorOrder: array[0..11] of Byte = (0, 1, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3);
  BitsOrder: array[0..11] of Byte = (7, 5, 5, 5, 5, 5, 4, 4, 4, 4, 3, 2);
var
  InData: array[0..6] of Byte;
  BitPerByte: array[0..55] of Byte;
  Vectors: array[0..12] of ShortInt;
  i, J: Integer;
  Bitmap: TBitmap;
  JpegImg: TJpegImage;
```

```
  HeightCnt, VectorCnt: Integer;
  JpegRowString: string;
begin
  FIn := TFileStream.Create(aPath, fmOpenRead);
  if not DirectoryExists(ExtractFilePath(aPath) + 'JPEG') then
    CreateDir(ExtractFilePath(aPath) + 'JPEG');
  Bitmap := TBitmap.Create;
  HeightCnt := 0;
  try
    Bitmap.Width := 53;
    Bitmap.Height := 90;
    Bitmap.PixelFormat := pf1bit;

    while (FIn.Position < FIn.Size) do
    begin
      FillChar(Vectors, 13, 0);
      FillChar(BitPerByte, 56, 0);
      FIn.Read(InData, 7);
      LPC10_extract_bits(@InData, @BitPerByte);
      for i := 1 to 53 do
      begin
        Vectors[__iblist[54 - i - 1] - 1] := (Vectors[__iblist[54 - i - 1] - 1] shl 1) or
BitPerByte[53 - i];
      end;
      VectorCnt := 0;
      JpegRowString := '';
      for i := 0 to Pred(12) do
      begin
        for J := 0 to Pred(BitsOrder[i]) do
        begin
          if ((Vectors[VektorOrder[i]] shr J) and 1) > 0 then
```

```
          Bitmap.Canvas.Pixels[J + VectorCnt, HeightCnt] := clBlack
        else
          Bitmap.Canvas.Pixels[J + VectorCnt, HeightCnt] := clWhite;
        JpegRowString := JpegRowString + IntToStr(Integer(not
(Bitmap.Canvas.Pixels[J + VectorCnt, HeightCnt] > 0)));
      end;
      Inc(VectorCnt, BitsOrder[i]);
    end;
    Inc(HeightCnt);
  end;
  JpegImg := TJpegImage.Create;
  try
    JpegImg.Assign(Bitmap);
    JpegImg.SaveToFile(ExtractFilePath(aPath) + 'JPEG\' + ExtractFileName(aPath)
+ '.jpg');
  finally
    JpegImg.Free;
  end;
 finally
  Bitmap.Free;
  FreeAndNil(FIn);
 end;
end;
end.

unit UnitVHF_Vocoder;

interface

uses
 Windows, classes, SysUtils;
```

```
const
  __DEF_DLL_PATH = 'dll\New\VC_vhf.dll';

type
  PFR_Lpc10_D = ^TFR_Lpc10_D;

  TFR_Lpc10_D = array[0..6] of Byte;

  PFR_Lpc10_E = ^TFR_Lpc10_E;

  TFR_Lpc10_E = array[0..359] of Byte;

  TInitVocoderParams = procedure(); cdecl;

  TVC_DecodeLpc10 = function(aFR_Lpc10_D: PFR_Lpc10_D; aOutBuf: PWord;
aErr: PInteger): Integer; cdecl;

  TVC_EncodeLpc10 = function(aFR_Lpc10_E: PFR_Lpc10_E; aOutBuf: PByte;
aErr: PInteger): Integer; cdecl;

procedure InitVocoderProc();

procedure FreeVocoderProc();

var
  VC_Dll: THandle;
  VC_InitVocoderParams: TInitVocoderParams;
  VC_DecodeLpc10: TVC_DecodeLpc10;
  VC_EncodeLpc10: TVC_EncodeLpc10;
```

```
implementation

procedure InitVocoderProc();
begin
  VC_Dll := loadLibrary(__DEF_DLL_PATH);
  if VC_Dll <> 0 then
  begin
    @VC_InitVocoderParams := getProcAddress(VC_Dll, 'VC_InitVocoderParams');
    @VC_DecodeLpc10 := getProcAddress(VC_Dll, 'VC_DecodeLpc10');
    @VC_EncodeLpc10 := getProcAddress(VC_Dll, 'VC_EncodeLpc10');

    if addr(VC_InitVocoderParams) = nil then
      raise Exception.Create('Vocoder Dll procedure not found!');
  end
  else
    raise Exception.Create('Vocoder Dll not found!');
end;

procedure FreeVocoderProc();
begin
  FreeLibrary(VC_Dll);
end;

end.
```

Додаток Б. Програмний код підсистеми згорткової нейронної мережі

```
import matplotlib.pyplot as plt
import numpy as np
import os
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
path = ‘SpeechImages’



if __name__ == ‘__main__’:
    batch_size = 32
    img_height = 90
    img_width = 53

    train_ds = tf.keras.preprocessing.image_dataset_from_directory(
        path,
        validation_split=0.2,
        subset="training",
        seed=123,
        image_size=(img_height, img_width),
        batch_size=batch_size)

    val_ds = tf.keras.preprocessing.image_dataset_from_directory(
        path,
        validation_split=0.2,
        subset="validation",
        seed=123,
```

```
    image_size=(img_height, img_width),
    batch_size=batch_size)

class_names = train_ds.class_names

print(class_names)

for image_batch, labels_batch in train_ds:
    print(image_batch.shape)
    print(labels_batch.shape)
    break

AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)


def fcount(path, map={}):
    count = 0
    for f in os.listdir(path):
        child = os.path.join(path, f)
        if os.path.isdir(child):
            child_count = fcount(child, map)
            count += child_count + 1  # unless include self
    map[path] = count
    return count

num_classes = fcount(path)

data_augmentation = keras.Sequential(
```

```
    [
        layers.experimental.preprocessing.RandomFlip("horizontal",
                                                     input_shape=(img_height,
                                                                  img_width,
                                                                  3)),
        layers.experimental.preprocessing.RandomRotation(0.1),
        layers.experimental.preprocessing.RandomZoom(0.1),
    ]
)

model = Sequential([
    data_augmentation,
    layers.experimental.preprocessing.Rescaling(1. / 255),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes)
])

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

model.summary()
```

```
epochs = 100
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs
)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
# plt.savefig()

from keras.preprocessing.image import load_img
```

```
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import array_to_img
import glob

ImagesList = glob.glob('CheckForSpeechImages/*.jpg')

for image in ImagesList:
    # load the image
    img = load_img(image)
    print(type(img))
    # convert to numpy array
    img_array = img_to_array(img)
    # img_array = keras.preprocessing.image.img_to_array(path +
'/Seven/Test.jpg')
    img_array = tf.expand_dims(img_array, 0)  # Create a batch

    predictions = model.predict(img_array)
    score = tf.nn.softmax(predictions[0])

    print(
        "{} most likely belongs to {} with a {:.2f} percent confidence."
            .format(image, class_names[np.argmax(score)], 100 * np.max(score))
    )
```