Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи пригнічення шуму в комп’ютерній системі обробки аудіоряду

(тема)

Виконав:

студент II курсу, групи СПм-20-1

Сітніков В.І.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Барковська О.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Сітнікову Віталію Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи пригнічення шуму в комп’ютерній системі обробки аудіоряду

затверджена наказом по університету від “ 05 ” листопада 2021 р. № 1657 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2021 р.

3. Вхідні дані до роботи
Навчальна вибірка аудіофайлів, тестова вибірка аудіофайлів, графічна карта ….

4. Перелік питань, що потрібно опрацювати у роботі
Розробка системи розпізнавання акустичних подій
Адаптація запропонованих методів шумопригнічення під обчислювальні системи із різними типами організації пам’яті
Аналіз якості розпізнавання розробленої системи в залежності від методів пригнічення шуму
Аналіз часу роботи системи для різних методів пригнічення шуму

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

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	Розробка системи розпізнавання акустичних		
	подій	9 листопада 2021	
2	Адаптація методів пригнічення шуму під		
	різні типи обчислювачів	15 листопада 2021	
3	Оцінка таймінгу роботи розробленої системи	25 листопада 2021	
4	Оцінка якості розпізнавання акустичних подій	30 листопада 2021	
5	Підготовка пояснювальної записки	7 грудня 2021	
6	Подача роботи до екзаменаційної комісії	13 грудня 2021	

Дата видачі завдання 8 листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Барковська О.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 81 с., 29 рис., 5 табл., 2 дод., 29 джерел.

ШУМОПРИГНІЧЕННЯ, АУДІОАНАЛІЗ, ШУМ, ФІЛЬТР, СПЕКТРОГРАМА, ЧАСТОТА, НЕЙРОМЕРЕЖА.

Метою кваліфікаційної роботи є дослідження впливу методів пригнічення шуму на роботу системи розпізнавання акустичних подій.

У ході виконання кваліфікаційної роботи розглянуто існуючи методи шумопрінічення, розроблена та протестована система розпізнавання аудіоподій як у першопочатковому вигляді, так і з інтегрованими методами шумопригнічення.

ABSTRACT

Master’s thesis: 81 pages, 29 figures, 5 tables, 2 appendices, 29 sources.

NOISE SUPPRESSION, AUDIOANALYSIS, NOISE, FILTER, SPECTROGRAM, FREQUENCY, NEURAL NETWORK.

The major goal of this thesis is the investigation of the noise reduction methods impact on the acoustic event recognition system activity.

In order to achive these goals were analised existing noise reduction methods, an audio event recognition system developed and the system tested both in its initial state and with integrated noise reduction methods.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АШП, ANC – активне шумопригнічення (англ., Active Noise Cancellation)

Гц – герц

ШПФ, FFT – швидке перетворення Фур’є (англ, Fast Fourier Transform)

CNN – згорткова нейронна мережа (англ., Convolutional neural network)

GEMM – операції загального множення матриці на матрицю (англ., General Matrix Multiplications)

LSTM – довга короткострокова пам’ять (англ., Long short-term memory)

RAAM – рекурсивна автоасоціативна пам’ять (англ., Recursive Auto Associative Memory)

RvNN, RNN – рекурсивні нейронні мережі (англ., Recursive Neural Networks)

ВСТУП

Сьогодні фраза «work from home» – є звичайною річчю. Після виявлення COVID-19 багато людей перейшли на роботу з дому. Сучасні технології дозволяють нам проводити зустрічі, лекції, тренінги, онлайн конференції і т. д. не відходячи від домашнього комп’ютера. У кожного з нас є вдома провідний інтернет з достатньою швидкістю задля забезпечення гарної якості з’єднання. Проте, якість аудіосингалу залежить не лише від якості з’єднання інтернету.

Нерідко, при розмові, можуть виникати сторонні звуки, такі як: шум з вулиці, кулер ноутбука (якщо людина працює з нього), шум побутової техніки та інші. Навіть коли наявних шумів немає, погана якість мікрофону може генерувати «білий шум», який також може впливати на якість сигналу. Тож другим та ключовим аспектом якості бесіди є обробка аудіосигналу до його передачі співбесіднику.

Задач аудіоаналізу декілька: розпізнавання мови, ідентифікація мовця, пошук музичної інформації, виявлення подій, і т. д. Проте, найбільш популярною є шумопригнічення в аудіо. Усі вони припускають використання тих чи інших алгоритмів.

У цій атестаційній роботі будуть розглянуті як прості алгоритми для шумопригнічення (перетворення Фур’є), так і більш складні та сучасні (використання нейронних мереж).

Метою даної кваліфікаційної роботи є аналіз алгоритмів шумопригнічення у аудіосигналах, їх інтеграція у систему розпізнавання аудіо подій, а також оцінка їх впливу на систему по таким критеріям як точність і повнота.

1 АНАЛІЗ АКТУАЛЬНОСТІ ЗАДАЧІ ОБРОБКИ АУДІОРЯДУ У КОМП’ЮТЕРНИХ СИСТЕМАХ

1.1 Сучасні області застосування систем аналізу і синтезу аудіоряду

Наше повсякденне життя сильно залежить від інформації, наприклад у форматах у вигляді тексту та мультимедіа. Нам потрібна інформація для загальних процедур, таких як перегляд/читання новини, слухання радіо, перегляд відео тощо. Однак ми часто стикаємося з проблемами, коли потрібен певний тип інформації. Величезна нестача кількісті інформації ускладнює пошук того, що потрібно.

Швидке збільшення інформації накладає нові вимоги до управління контентом, оскільки медіа архіви та споживчі товари починають бути дуже складними і з ними важко впоратися. В даний час люди здійснюють пошук у базах даних за різними мета-тегами (наприклад назва, опис, автор і т. д.), які лише коротко описують цілі шматки інформації текстів великого обсягу. Мета-теги створюються різними особами і, зрозуміло, що тлумачення мета-тегів може відрізнятися один від одного для кожної особи. Автоматична система, яка описує звук, систематизує контент, а також дозволить здійснювати пошук за фактичними даними, а не лише за метаданими. Покращення управління контентом – мета автоматичних систем опису звуку. Деякі програми для управління комерційним контентом уже вийшли, але багато можливі програми ще не розроблені.

Наприклад, людина слухає радіо і хоче слухати джаз. На жаль, усі радіостанції грають поп-музику, змішану з рекламою. Слухач створює пошуки джазу і отримує лише поп-музику.

Наведений вище приклад можна вирішити за допомогою автоматичної системи опису звуку. Тоді сценарій може змінитися на інший. Людина, яка хоче слухати джаз може знайти поп-музику лише на налаштованих

радіостанціях цього жанру. Тоді слухач натискає кнопку «пошук джазу» на приймачі і через пару секунд приймач змінює радіостанцію та лунає джазовий звук з динаміків. Цей приклад показує, як управління вмістом може бути ефективним інструментом, який щодня спрощує роботу процедури, що включають управління інформацією.

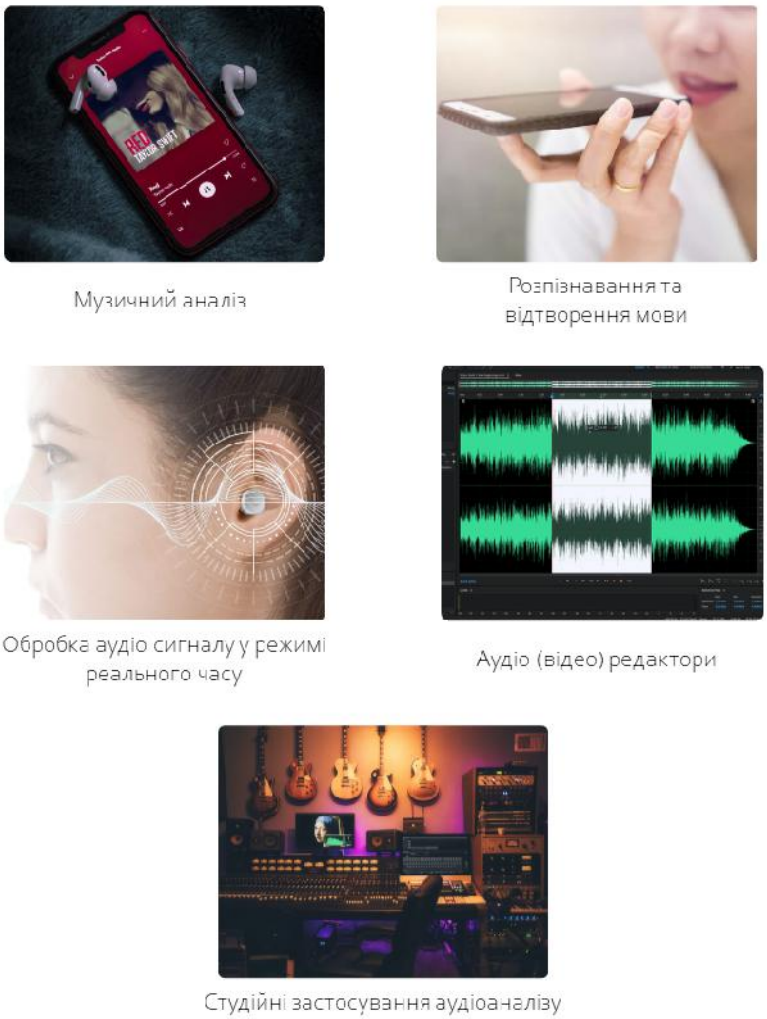


Рисунок 1.1 – Задачі аудіоаналітики

Насамперед, сама аудіо інформація відіграє досить важливу роль у збільшенні цифрового вмісту, доступного сьогодні, що спричиняє необхідність у методологіях, які автоматично аналізують такий вміст:

розпізнавання аудіоподій (наприклад, для детектування агресії – розмова на підвищених тонах, крик; сигналізація автомобілів; пошук ключових слів “поліція”, “допоможіть” і т. д.; постріли, вибухи; крик/плач дитини), розпізнавання мови задля виконання голосових команд або написання тексту, визначення відстані до джерела звуку, пошук музичної інформації, мультимодальний аналіз (напр. аудіовізуальний аналіз онлайн-відео для рекомендацій на основі вмісту) тощо.

Для будь-якої аудіо системи основною метою є якісне звучання. Однак, що саме визначає його якість? Існує три основні міри хорошого звуку:

- чутність;
- зрозумілість;
- точність.

Багато факторів сприяють якості звук, у тому числі якість звуку джерела, звукова система, і акустика приміщення.

Чутність промови або музики має бути достатньою для досягнення бажаного ефекту: зазвичай комфортний рівень для прослуховування промови та більш потужні рівні для певних видів музики. Ці рівні повинні бути досяжні без спотворення або зворотного зв’язку. Зрозумілість визначається відношенням сигнал / шум та відношенням прямого сигналу до реверберації. «Сигнал» – це джерело бажаного звук (мова, музичні інструменти тощо). На рисунку 1.2 зображена спектрограма запису читання аудіо книги. Тут можна спостерігати чіткі накопичення енергії приблизно від 100 Гц до 10 кГц, з чого можна зробити висновок, що це чоловіча промова [1] (рисунок 1.4). Тоді як «шум» – це навколишній звук у кімнаті, такий як електричний шум, що виробляє звукова система. На рисунку 1.3 представлений приклад спектрограми шуму натовпу. У цьому випадку, в порівнянні з попереднім, можна спостерігати велике накопичення енергії у діапазонах нижніх частот, що і свідчить про зашумлений запис.

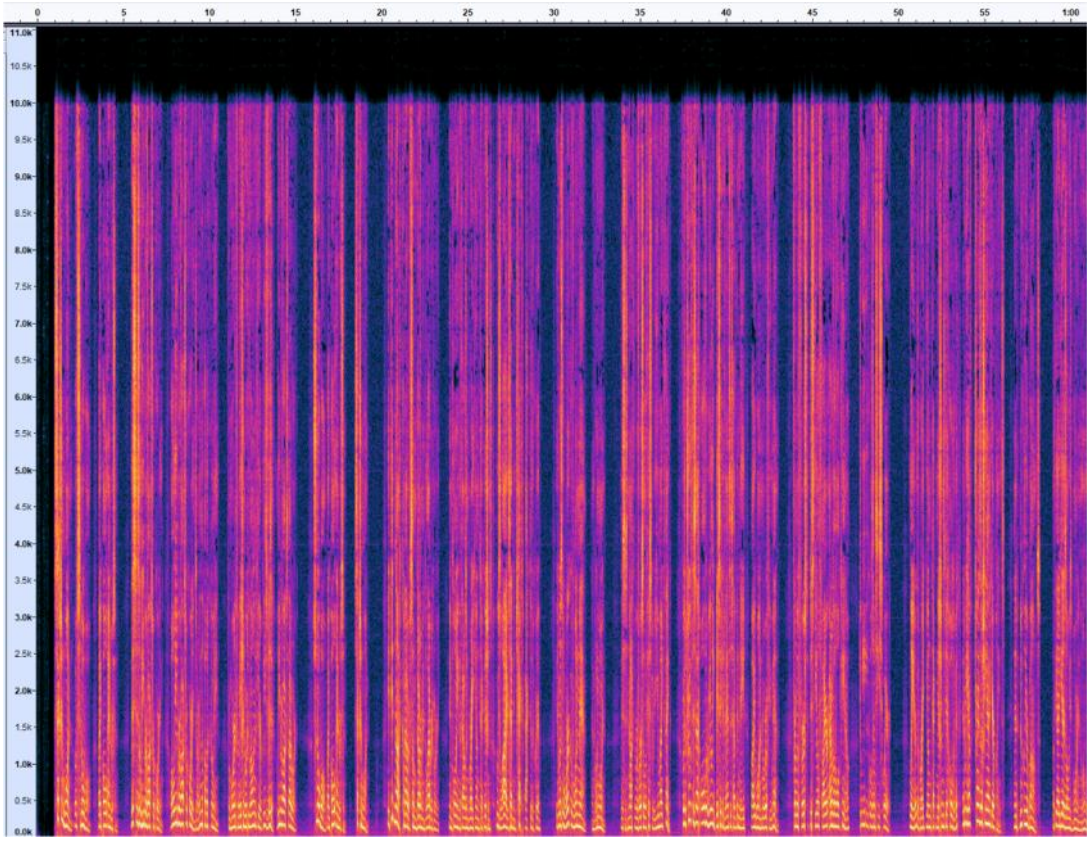


Рисунок 1.2 – Спектрограма запису читання аудіо книги

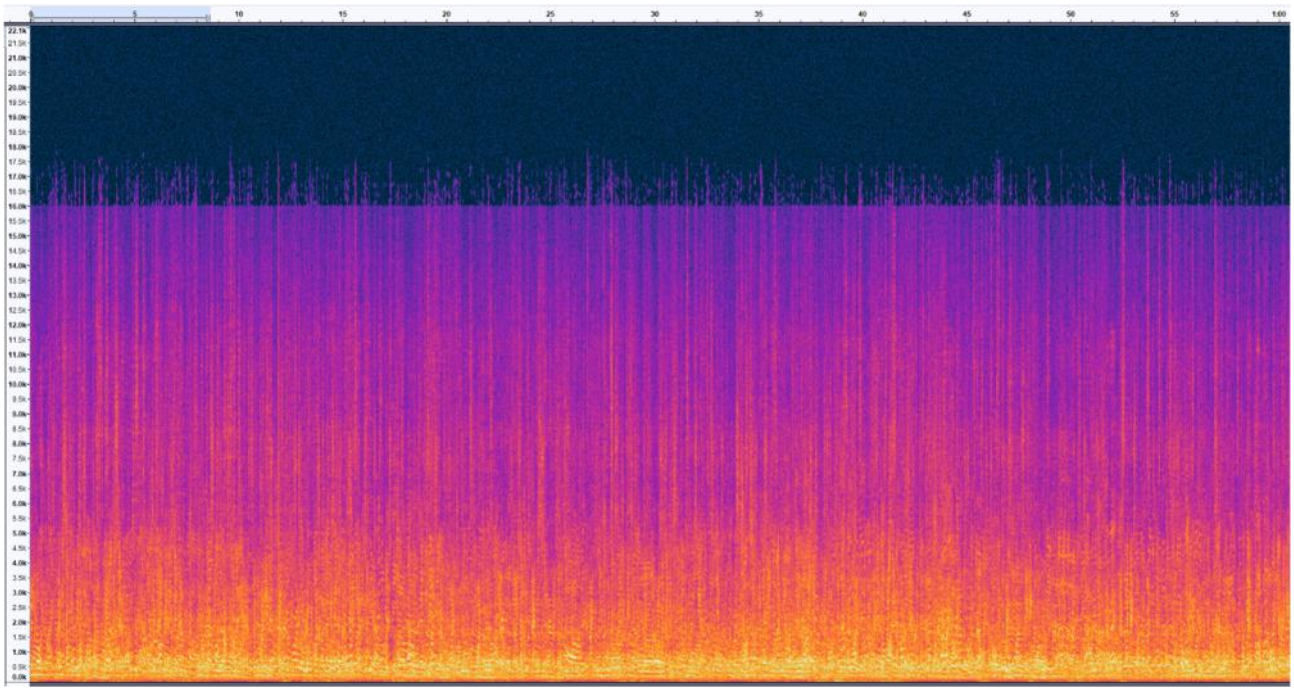


Рисунок 1.3 – Спектрограма запису шумного гулу натовпу

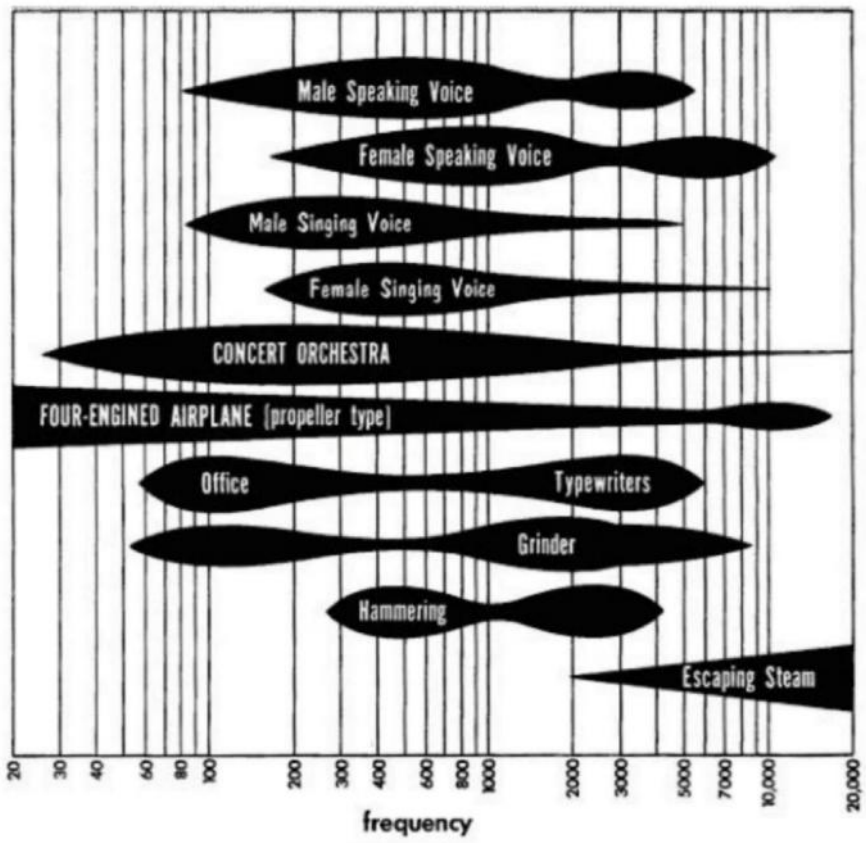


Рисунок 1.4 – Розподіл різних звуків у чутному діапазоні частот [1]

Максимальна розбірливість мови вимагає рівня мовлення принаймні на 20 дБ віще рівня шуму. Співвідношення прямого сигналу до реверберацийного визначається спрямованістю гучномовців та реверберацийними характеристиками приміщення. Високий рівень реверберації може серйозно погіршити розбірливість, ускладнивши розбірливості кінця одного слова та початку наступного. І, нарешті, точність звуку, насамперед, визначається загальною частотною характеристикою звуку, що надходить на вухо слухача. Діапазон частот повинен бути достатнім широкий та відносно однорідний, щоб забезпечити реалістичність і точне підкріплення мови та музики. Кожен компонент ланцюга сигналів сприяє на звук, а їх обмеження в будь-який момент може вплинути на точність всієї системи. [2]

Збільшення доступності аудіо контенту через величезний розподіл каналів призвело до необхідності створення систем, здатних автоматично

аналізувати його вміст. Залежно від окремих типів каналів розповсюдження, типів аудіо (мовлення, музики тощо), наявності інших засобів масової інформації (наприклад, візуальної інформації) та вимог, що стосуються конкретних програм, за останній час з’явився широкий спектр різних застосувань: пошук музичної інформації, виявлення аудіоподій, аналіз мовлення та динаміків, розпізнавання речових емоцій, мультимодальний аналіз тощо.

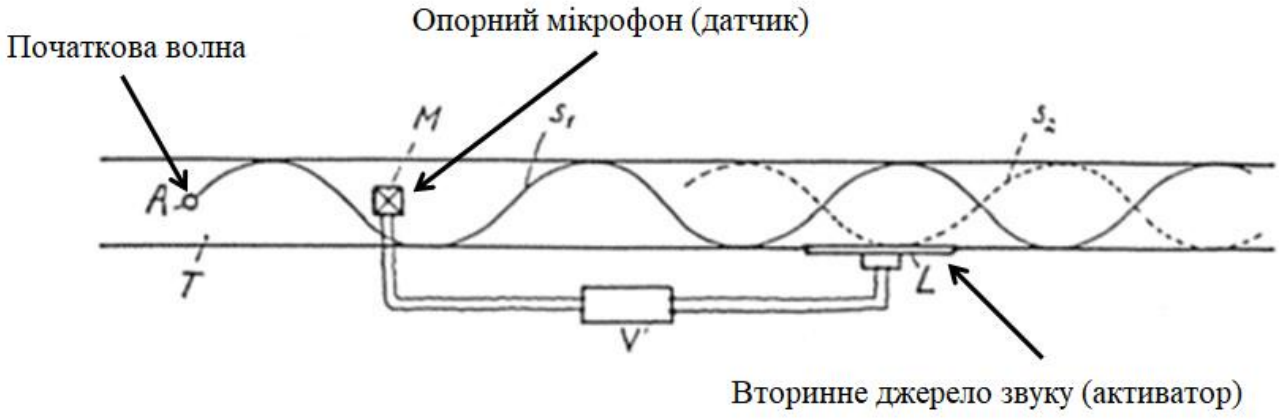
1.2 Еволюція методів шумопригнічення в аудіоситемах: від аналогових до цифрових

Все почалося в Німеччині в 1933 р., коли доктор філософії та медицини Пауль Люг подав заявку на патент щодо використання фазових сигналів для усунення синусоїдальних тонів у лампах та зміни полярності для усунення звуків навколо гучномовця.

Підхід передбачав зміщення переходів з витоками звукових коливань (наприклад, вібрації частинок повітря, паралельних напрямку хвилі) таким чином, що їх суперпозиція (наприклад, принцип суперпозиції, включаючи чисту реакцію двох або більше подразників у відповідь узагальнює суму індуковану кожним стимулом для всіх лінійних систем) є механічною.

Далі історія рухається у напрямку патенту, затвердженого в 1936 році. Теоретично, це було практично, але не було відповідного обладнання для зондування, обробки та виробництва звуку. Це не могло бути розширено Полом Люгом на новий рівень.

Доктор Лоуренс Джером Фогель продовжив розвиток пізніше в 1950-х роках, і він був одним із засновників дослідження еволюційних обчислень та аналізу людського фактора. Фогель також створив і запатентував пристрої активного шумопригнічення в галузі аерокосмічної промисловості (гелікоптери). Цей пристрій був призначений для зменшення шуму пілота в кабіні, щоб сприяти контактам.



Оригінальний патент Пола Люга (1993)

Рисунок 1.5 – Принцип роботи інтерференції при приглушенні синусоїдальних і інших (довільних) звукових сигналів в області гучномовця шляхом інверсії полярності запатентований Полом Люгом

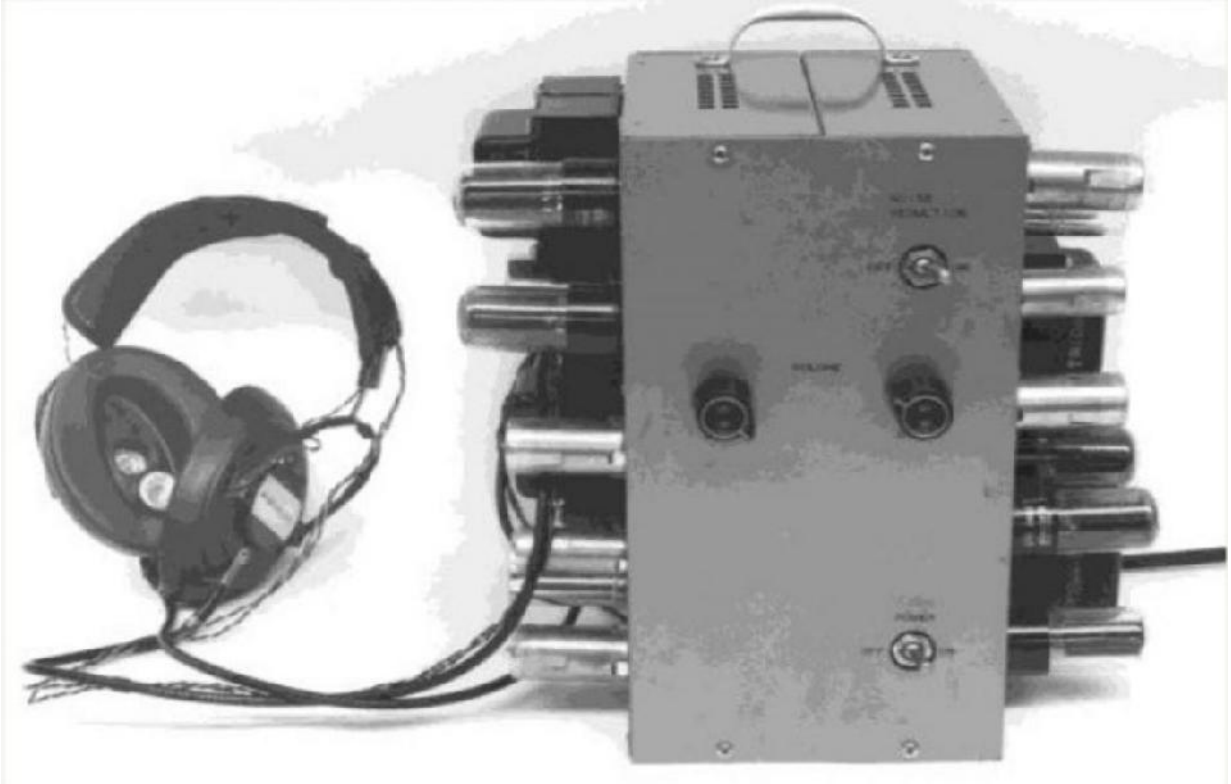


Рисунок 1.6 – Гарнітура з активним зменшенням шуму та електронна трубка [3]

Фогель був відомий як винахідник активного шумопоглинання, і він розробив одне з перших пристроїв для навушників для шумопригнічення. Приблизно в той же період Дослідницька лабораторія ВПС (AFRL) мала програму досліджень приладів, яка зосереджувалася на безпеці та захисті слуху, голосовому зв’язку, аналізі шуму та моделюванні шуму громади.

У 1986 році Дік Рутан та Джина Джигер провели справжнє випробування за допомогою навушників Bose. Протягом 9 днів, 3 хвилин та 44 секунд вони керували двомісним літаком (визнаним як модель Rutan 76 Voyager Rutan), який встановив новий рекорд витривалості польоту [4].

Рисунок 1.7 – Перші комерційні гарнітури з шумопоглинанням представлені компанією Bose у 1989 році

Bose не була єдиною організацією, що виробляє навушники, щоб зосередитися на навушниках з шумопоглинанням. Sennheiser, інша велика німецька корпорація, що виробляє навушники, зосереджена на агресивному контролі шуму в навушниках.

Вони мали на меті допомогти пілотам найбільшої німецької авіакомпанії Lufthansa захиститися від низькочастотного шуму. У 1987 році вони представили першу дозволену пілот-гарнітуру з активним шумопоглинанням (FAA-TSO) з тегом LHM 45 NoiseGard.

Озираючись на перші проекти навушників із шумозаглушенням у 1978 році, технологія шумопоглинання, яка використовується сьогодні, не так вже й відрізняється від попередньої. Однак, завдяки природній еволюції цифрових технологій, були досягнуті значні вдосконалення та еволюційні досягнення, такі як технологія бездротового та цифрового шумопоглинання. Сьогодні можливо побачити навіть побачити новаторські функції в навушниках, такі як адаптивна технологія шумопригнічення та використання штучного інтелекту для шумопоглинання в навушниках.

1.3 Загальний принцип роботи аналогового та цифрового шумопригнічення

Активне шумопригнічення (АШП) засноване на принципі протифазної компенсації. Видається сигнал пригнічення, який замінює та анулює наявний шум. В ідеалі компенсаційний сигнал має таку ж амплітуду, як і сигнал шуму, але з інвертованою фазою.

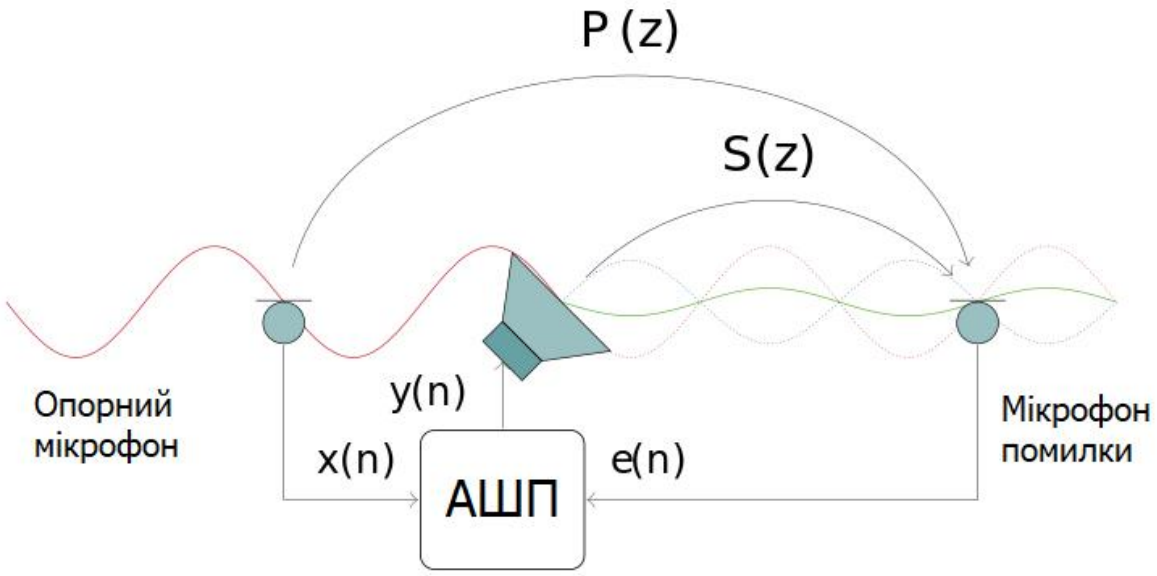


Рисунок 1.8 – Візуалізація активного шумопригнічення

Принцип АШП показаний на рисунку 1.8. Один мікрофон вловлює

сигнал зовнішнього шуму $x(n)$, а інший мікрофон вимірює внутрішній сигнал помилки $e(n)$. Акустична передача від зовнішнього опорного сигналу до внутрішнього сигналу помилки описується первинним шляхом $P(z)$. Алгоритм АШП визначає сигнал скасування $y(n)$, який відтворюється через внутрішній гучномовець. Передача від гучномовця до внутрішнього мікрофону здійснюється вторинним шляхом $S(z)$.

Завдання полягає у створенні відповідного сигналу пригнічення, оскільки незначні відхилення в амплітуді та фазі вже призводять до серйозного погіршення продуктивності системи. Обробка сигналу повинна мати якомога меншу затримку. Вся затримка, зібрана між отриманням опорного сигналу та випромінюванням сигналу пригнічення, призводить до погіршення продуктивності. Таким чином, вимоги до обладнання в режимі реального часу є дуже складними. Крім того, створення відповідного сигналу вимагає хорошого знання основної акустичної системи. Розташування мікрофонів та гучномовців у акустичній частині має вирішальне значення. [5]

Аналогові спроби зменшення шуму були обмежені технологічними обмеженнями часу. Схеми, як правило, реалізовувалися в одному каналі, зменшення коефіцієнта посилення обмежувалося можливостями використовуваних аналогових фільтрів, а зменшення коефіцієнта посилення (часто все або жодне) зазвичай базувалося тільки на вхідному рівні. З появою цифрового пригнічення шуму почалася еволюція все більш складних алгоритмів, які використовують правила прийняття рішень, здатні визначати, що таке шум, наскільки відповідним є зменшення коефіцієнта посилення, і в яких діапазонах частот слід реалізувати зменшення посилення.

Загалом, тема АШП – це поєднання різних галузей, і побудова цілісної системи, що вимагає знань з обробки сигналів, техніки управління, акустичних вимірювань, дизайну акустичного інтерфейсу та систем реального часу, включаючи апаратне забезпечення.

1.4 Існуючі рішення в області аудіоаналітики

Хоча відеоаналіз став стандартною функцією багатьох камер безпеки, вбудована аудіоаналітика залишається досить рідкісним явищем, незважаючи на наявність як аудіоканалу в пристроях, так і наявних обчислювальних можливостей для обробки аудіоданих. Тим не менш, аудіоаналітика має деякі переваги перед відеоаналітикою: вартість мікрофонів та їх обслуговування набагато дешевше, ніж відеокамер; коли система працює в режимі реального часу, обсяг аудіоінформаційних даних значно менший за обсяг потоку даних з відеокамер, що висуває більш лояльні вимоги до пропускної здатності каналу передачі даних. Аудіоаналітичні системи можуть бути особливо актуальними для міського спостереження, де можливо автоматично розпочати трансляцію відео в прямому ефірі на поліцейську консоль з місця вибуху та стрілянини. Аудіоаналітичні технології також можна використовувати для вивчення відеозаписів та виявлення подій. Зі зростанням обізнаності про можливості цих систем використання аудіоаналітики буде тільки розширюватися [5].

Існує багато готових рішень в області аудіоаналітики. Нижче представлені декілька серед них:

ShotSpotter.

Лідер США у виявленні пострілів з вогнепальної зброї в міських умовах. У найбільш небезпечних районах встановлені спрямовані мікрофони (на стовпах, будинках, інших високих будівлях), які вловлюють звуки міського фону. У разі подтвердження ідентифікації пострілу інформація з GPS-координатами конкретного мікрофона передається на центральний комп’ютер, де проводиться додатковий звуковий аналіз для відфільтровування можливих помилкових спрацьовувань, таких як літаючий вертоліт або, наприклад, вибухає петарда. Якщо постріл підтверджується, патруль вирушає на місце. На рисунку 1.10 представлений повна схема тренування алгоритму.

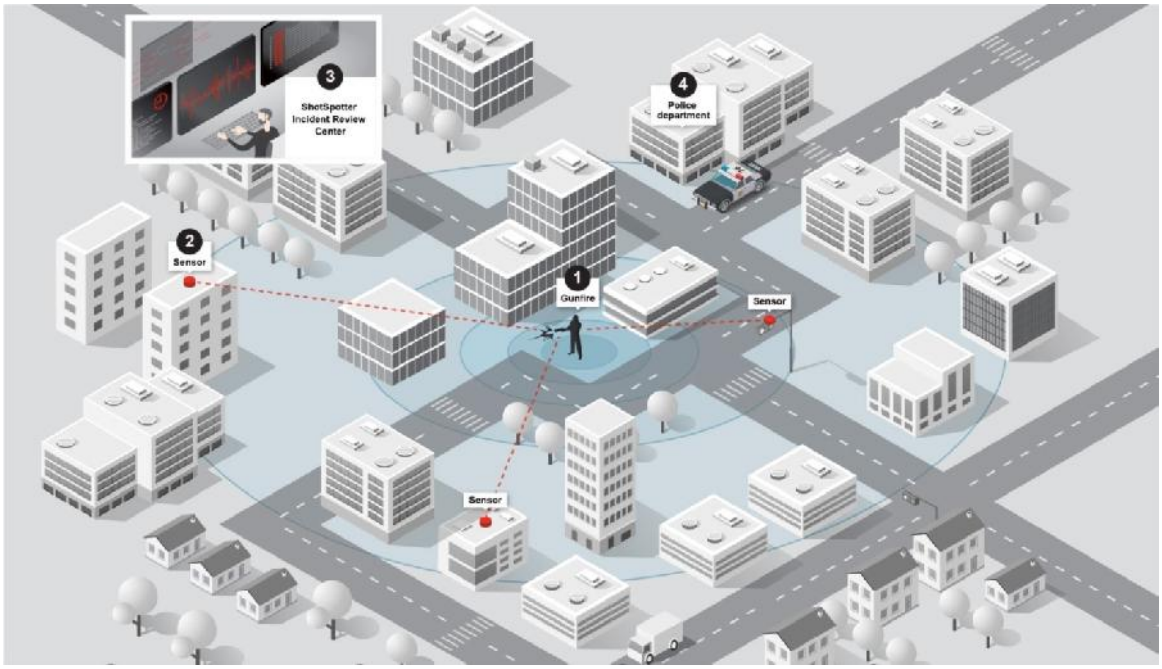


Рисунок 1.9 – Демонстрація роботи системи ShotSpotter

Рисунок 1.10 – Схема тренування алгоритму детектування [7]

TrueVoice.

Це рішення для голосової аналітики, яке дозволяє керівникам підприємств отримати уявлення про кожну взаємодію з клієнтами, щоб

зменшити ризик, покращити досвід роботи з клієнтами, ефективність роботи агента та загальну операційну ефективність.

TrueVoice — це масштабована платформа штучного інтелекту, яка пропонує такі можливості:

- аналізує як мову, так і поведінку – і «те, що говорять», і «як це говорять»;

- пов’язує результати розмов з подорожжю клієнтів, де це доречно, допомагаючи ресурсам контакт-центру побачити загальну картину в різних взаємодіях;

- використовує моделі машинного навчання для прогнозування результатів, а не спирається на стандартне визначення ключових слів;

- побудована на відкритій, гнучкій, масштабованій хмарній архітектурі, що забезпечує легку інтеграцію з існуючими платформами та рішеннями для запису дзвінків [8].

Krisp.

Krisp – найкраще програмне забезпечення для шумопоглинаня 2021 року. Воно побудоване на базі штучного інтелекту, яке пригнічує шум у режимі реального часу. Доступний для macOS і Windows, Krisp працює з більш ніж 600 програмами.

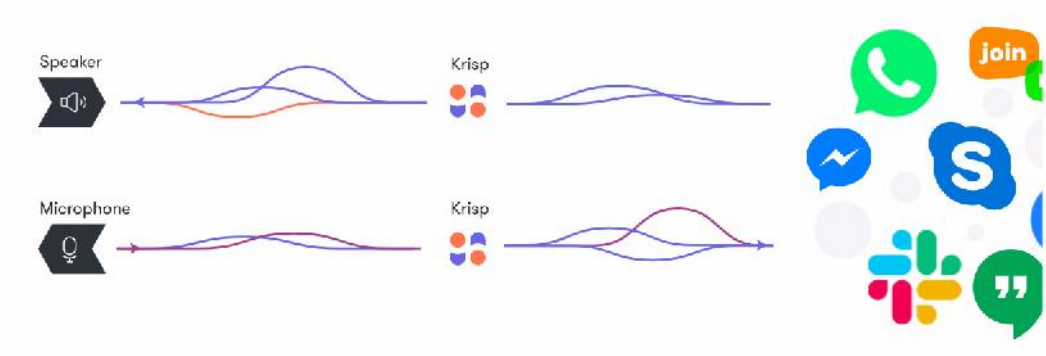


Рисунок 1.11 – Демонстрація програми Krisp (пригнічує шум як мікрофону так і динаміків)

Krisp приймає будь-який шум, який передається на мікрофон, і усуває фоновий шум за допомогою своїх алгоритмів штучного інтелекту, перш ніж він передається в додаток для відеоконференцій.

Велика бібліотека Krisp, що налічує понад 20000 шумів, 50000 окремих динаміків та 2500 годин аудіо допомагає ідентифікувати фоновий шум, що допомагає усунути його [9].

1.5 Класифікація методів шумозаглушення

Існує безліч різних класифікацій шумів, наприклад, за характером спектра або частотою хвиль. Однак, якщо є потреба позбавитися шумів у записі, варто в першу чергу враховувати категоризацію шумів за часовими характеристиками.

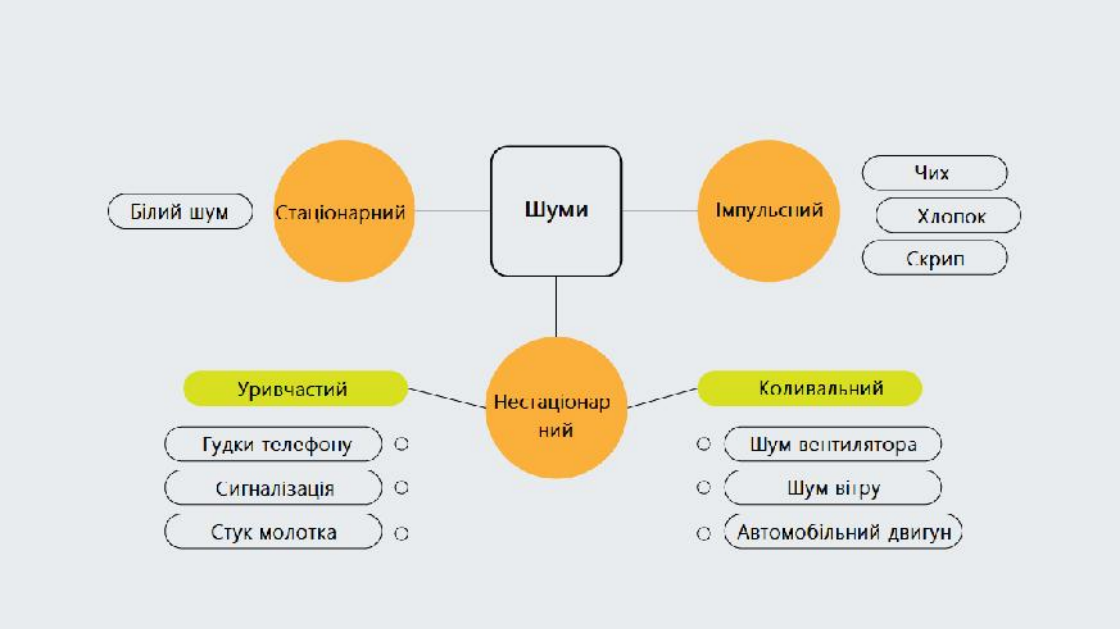


Рисунок 1.12 – Класифікація шумів [10]

Методи шумопргнічення можна розбити по таким групам:

- фільтри (як аналогові так і цифрові);
- комбінації фільтрів;

- нейромережі.

Фільтри є основою для аналізу та обробки аудіосигналів. Багато з них використовуються у тому числі для обробки зображень, або взагалі були розроблені для цього (якщо представити аудіосигнал у вигляді спектрограми – отримаємо те саме зображення). Багато фільтрів перейшли з аналогового виду у цифровий і використовуються по цей день. Їх комбінація дає змогу покращити обробку сигналу у рази.

Використання нейромереж – більш новий підхід обробки аудіосигналу.

1.6 Постановка задачі

Метою кваліфікаційної роботи є дослідження впливу методів пригнічення шуму на роботу системи розпізнавання акустичних подій.

Для досягнення поставленої мети мають бути вирішені наступні задачі:

- дослідження загальних параметрів акустичних систем;
- дослідження існуючих методів шумопригнічення в аудіоряді;
- розробка системи розпізнавання акустичних подій;
- адаптація методів пригнічення шуму під різні типи обчислювачів;
- оцінка таймінгу роботи розробленої системи;
- оцінка якості розпізнавання акустичних подій;
- аналіз отриманих результатів.

2 АНАЛІЗ ВИМОГ СИСТЕМИ

2.1 Аналіз обчислювальних систем з загальною пам’яттю та масовим паралелізмом

З часу винайдення першого у світі графічного процесора у 1999 році, графічні процесори NVIDIA стали першовідкривачами 3D-графіки та обчислень з графічним процесором. Кожна архітектура графічного процесора NVIDIA ретельно розроблена, щоб забезпечити проривні рівні продуктивності та ефективності.

Сімейство нових графічних процесорів з архітектурою NVIDIA Ampere розроблено для прискорення різноманітних типів обчислювальних програм та робочих навантажень. Перша архітектура NVIDIA Ampere графічного процесора, A100, була випущена в травні 2020 року і забезпечує величезні прискорення для навчання штучному інтелекту та передбачень, навантаження HPC та програм аналізу даних.

Найновішими представниками сімейства графічних процесорів NVIDIA Ampere є GA102 та GA104. Вони є частиною нового класу NVIDIA «GA10x» графічних процесорів амперної архітектури. Графічні процесори GA10x базуються на революційному графічному процесорі архітектури NVIDIA Turing.

Графічний процесор NVIDIA GA100 містить у собі тензорні ядра – спеціалізовані високопродуктивні обчислювальні ядра для матричних математичних операцій, які забезпечують новаторські показники для програм штучного інтелекту та HPC. Вони виконують матричні розрахунки множення та накопичення (MMA). Сотні тензорних ядер працюють паралельно в одному графічному процесорі NVIDIA, що дозволяє значно збільшити пропускну здатність та ефективність. Ці ядра були вперше представлені в графічному процесорі NVIDIA Tesla V100, і ще більше вдосконалені в

останніх графічних процесорах Тьюринга компанії NVIDIA.

Операції загального множення матриці на матрицю (GEMM) лежать в основі навчання нейронної мережі та припущень, а також використовуються для множення великих матриць вхідних даних та ваг у різних шарах. Операція GEMM обчислює матричний добуток $D = A * B + C$, де C і D – матриці m-до-n, A – матриця m-до-k, а B – матриця k-до-n. Розмір проблеми таких операцій GEMM, що виконуються на тензорних ядрах визначаються розмірами матриці, і зазвичай позначається як m-до-n-до-k.

Використовуючи, наприклад, операції з тензорним ядром змішаної точності FP16/FP32 на апаратному забезпеченні, рівень кожного тензорного ядра в архітектурі Volta може виконувати 64 FP16 операцій комбінованого множення-додавання (FMA) з накопиченням FP32 за такт, що дозволяє йому обчислювати змішану точність множення матриці 4x4x4 на такт. Оскільки кожен потоковий мультипроцесор Volta містить вісім тензорних ядер, то єдиний потоковий мультипроцесор забезпечує 512 FP16 FMA операцій на такт або 1024 окремих FP16 з плаваючою крапкою операцій за такт. Кожне з тензорних ядер A100 може виконувати 256 FP16 FMA операцій за такт, що дозволяє йому обчислити результати для матричного змішування з точністю 8x4x8 зі змішаною точністю за такт. Кожен потоковий мультипроцесор в графічному процесорі A100 включає чотири з нових перероблених тензорних ядер і тому кожен потоковий мультипроцесор в A100 забезпечує 1024 FP16 FMA операцій за такт (або 2048 FP16 індивідуальні операції з плаваючою комою на такт).

Порівнюючи загальну продуктивність графічного процесора, а не лише продуктивність на рівні SM, графічний процесор NVIDIA A100 оснований на тензорних ядрах зі своїми 108 SM містить у собі загалом 432 тензорних ядра, які забезпечують до 312 TFLOPS щільної продуктивності змішаної точності FP16/FP32. Це дорівнює 2,5-кратній змішаній точності продуктивності ядра Tensor для всього графічного процесора Tesla V100 та стандартних 20-ти FP32 V100 пропускної можливості (FMA операції, що виконуються на

традиційних ядрах FP32 CUDA). [11]

Графічний процесор NVIDIA GA100 складається з декількох кластерів обробки GPU (GPC), текстури обробки кластерів (TPC), потокових мультипроцесорів (SM) та контролерів пам’яті HBM2.

Повна реалізація графічного процесора GA100 містить у собі наступні характеристики:

- 8 GPC, 8 TPC/GPC, 2 SM/TPC, 16 SM/GPC, 128 SM на весь графічний процесор;
- 64 ядра FP32 CUDA/SM, 8192 ядра FP32 CUDA на весь графічний процесор;
- 4 тензорних ядра третього покоління/SM, 512 тензорних ядер третього покоління на повний графічний процесор;
- 6 стеків HBM2, 12 512-розрядних контролерів пам’яті.

Реалізація графічного процесора NVIDIA A100 з графічним процесором GA100 містить у собі наступні характеристики:

- 7 GPC, 7 або 8 TPC/GPC, 2 SM/TPC, до 16 SM/GPC, 108 SM;
- 64 ядра FP32 CUDA/SM, 6912 ядер FP32 CUDA на графічний процесор;
- 4 тензорні ядра третього покоління/SM, 432 Тензорні ядра третього покоління на графічний процесор;
- 5 стеків HBM2, 10 512-розрядних контролерів пам’яті [12].

У додатку Б на рисунку Б.1 показано повний графічний процесор GA100 із 128 SM. A100 базується на GA100 і має 108 потокових мультипроцесорів.

2.2 Аналіз нейромереж, які використовуються для аудіоаналізу та шумопригнічення

Існують три найбільш відомі типи мереж глибокого навчання. До них

відносяться рекурсивні нейронні мережі (RvNN), рекурентні нейронні мережі (RNN) та згорткові нейронні мережі (CNN). Усі вони можуть бути використані для задач аудіоаналітики.

RvNN може досягати передбачення в ієрархічній структурі, а також класифікувати вихідні дані, використовуючи композиційні вектори. Рекурсивна автоасоціативна пам’ять (RAAM) є основним джерелом натхнення для розробки RvNN. Архітектура RvNN генерується для обробки об’єктів, які мають структури випадкової форми, такі як графіки або дерева. Цей підхід створює розподілене представлення фіксованої ширини з рекурсивної структури даних змінного розміру. Мережа навчається за допомогою введеної системи навчання з зворотним поширенням через структуру (BTS). Система BTS відстежує ту саму техніку, що й загальний алгоритм поширення назад, і має можливість підтримувати деревоподібну структуру. Автоматична асоціація навчає мережу відновлювати шаблон вхідного рівня на вихідному рівні. RvNN є високоефективним у контексті обробки натуральної мови (NLP). Сохер представив архітектуру RvNN, призначену для обробки вхідних даних з різних модальностей. Ці автори демонструють два застосування для класифікації речень природної мови: випадки, коли кожне речення розбивається на слова та образи природи, і випадки, коли кожне зображення поділяється на різні сегменти, що цікавлять. RvNN обчислює ймовірну пару балів для злиття та створює синтаксичне дерево. Крім того, RvNN обчислює оцінку, пов’язану з правдоподібністю злиття для кожної пари одиниць. Далі пара з найбільшим балом об’єднується у вектор композиції. Після кожного злиття RvNN генерує (a) більшу площу численних одиниць, (b) композиційний вектор області та (c) мітку для класу (наприклад, іменна фраза стане міткою класу для нового площа, якщо дві одиниці є словами-іменниками). Композиційний вектор для всієї області є коренем деревоподібної структури RvNN [13].

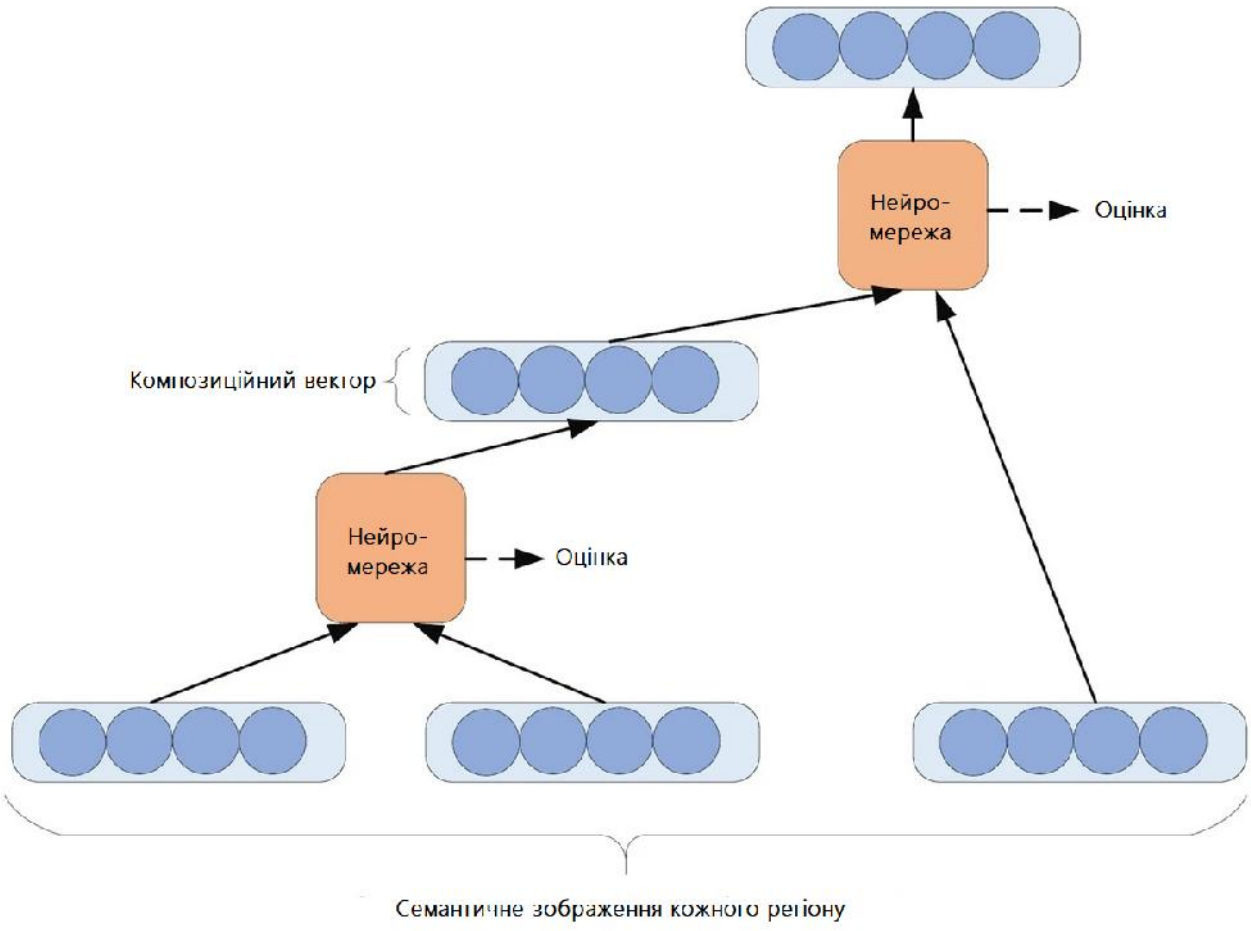


Рисунок 2.1 – Приклад дерева RvNN

RNN є широко використовуваним і знайомим алгоритмом у дисципліні глибокого навчання. RNN в основному застосовують у сфері обробки мовлення та контекстах NLP. На відміну від звичайних мереж, RNN використовує послідовні дані в мережі. Оскільки вбудована структура в послідовності даних надає цінну інформацію, ця функція є фундаментальною для ряду різних додатків. Наприклад, важливо зрозуміти контекст речення, щоб визначити значення конкретного слова в ньому. Таким чином, можна розглядати RNN як одиницю короткочасної пам’яті, де x являє собою вхідний шар, y – вихідний, а s – рівень стану (прихований). Для заданої вхідної послідовності типова розгорнута діаграма RNN проілюстрована на рисунку 2.2. Паскану представив три різні типи глибоких методів RNN, а саме «Прихований до прихованого», «Прихований до виведення» та «Вхід до

прихованого». Було запроваджено глибоку RNN, яка зменшує труднощі навчання в глибокій мережі та приносить переваги більш глибокої RNN на основі цих трьох методів.

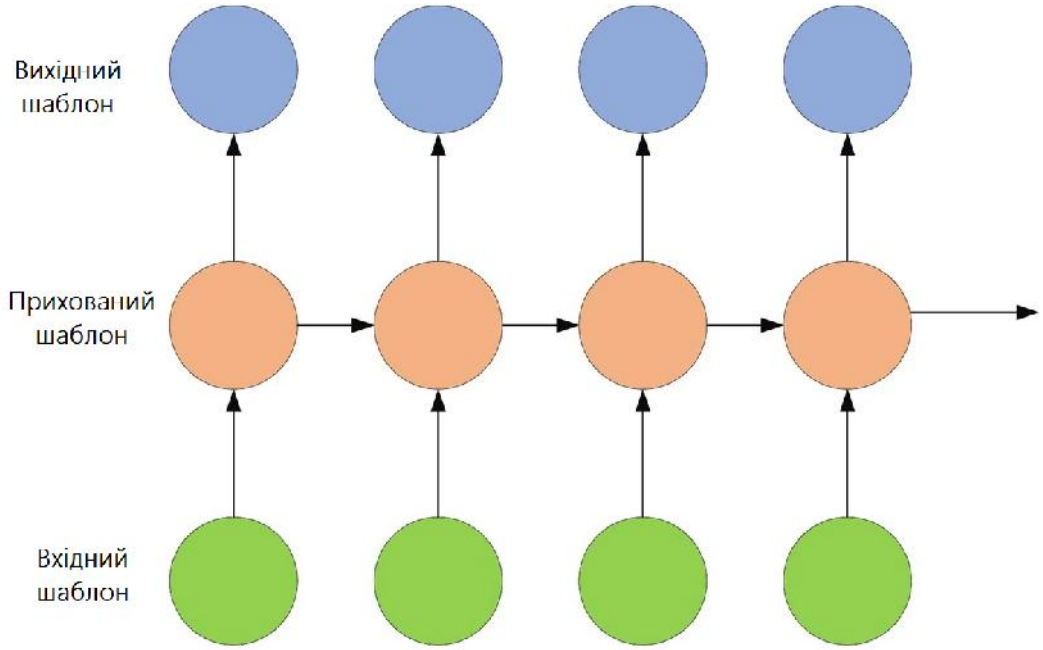


Рисунок 2.2 – Діаграма RNN

Однак чутливість RNN до вибухового градієнта та проблем зникнення є однією з головних проблем цього підходу. Більш конкретно, під час навчального процесу скорочення кількох великих або малих похідних може призвести до того, що градієнти експоненціально вибухають або розпадаються. З введенням нових входів мережа перестає думати про початкові; тому ця чутливість з часом зменшується. Крім того, цю проблему можна вирішити за допомогою довгої короткострокової пам’яті (LSTM). Цей підхід пропонує повторювані з’єднання з блоками пам’яті в мережі. Кожен блок пам’яті містить ряд комірок пам’яті, які мають можливість зберігати тимчасові стани мережі. Крім того, він містить блоковані блоки для управління потоком інформації. У дуже глибоких мережах залишкові з’єднання також мають здатність значно зменшити вплив проблеми зникаючого градієнта. CNN вважається потужнішим за RNN. RNN має

меншу сумісність функцій у порівнянні з CNN [13].

У сфері глибокого навчання, CNN є найбільш відомим і широко використовуваним алгоритмом. Основна перевага CNN у порівнянні з попередниками полягає в тому, що він автоматично визначає відповідні функції без будь-якого контролю людини. CNN широко застосовуються в різних областях, включаючи комп’ютерний зір, обробку мовлення, розпізнавання обличь тощо. Структура CNN була створена на зразок нейронів в мозку людини і тварин, подібними до звичайної нейронної мережі. Більш конкретно, в котячому мозку складна послідовність клітин утворює зорову кору; ця послідовність моделюється CNN. Гудфеллоу визначив три ключові переваги CNN: еквівалентні представлення, розріджені взаємодії та спільне використання параметрів.

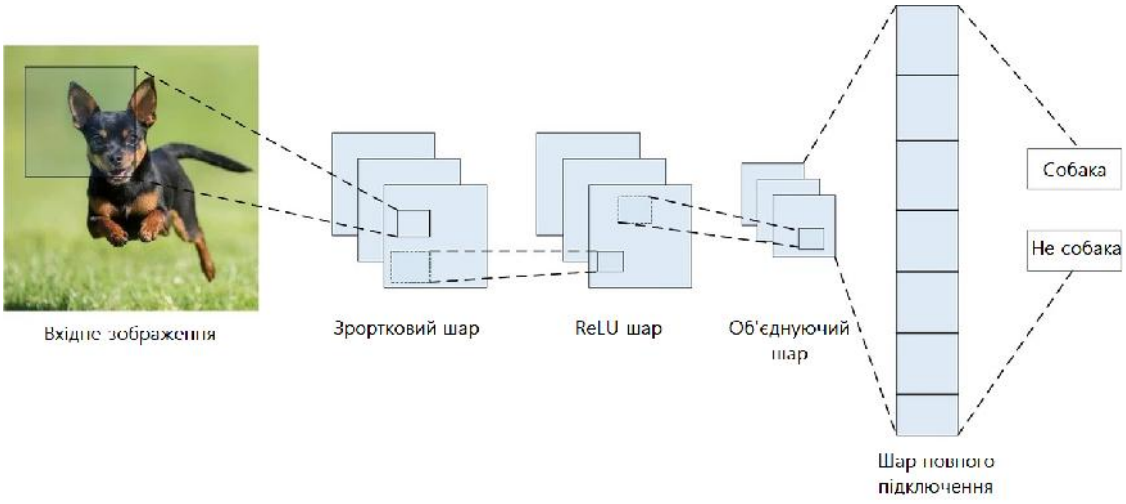


Рисунок 2.3 – Приклад архітектури CNN для класифікації зображень

На відміну від звичайних повністю з’єднаних мереж (FC), спільна вага та локальні з’єднання в CNN використовуються для повного використання 2D-структур вхідних даних, таких як сигнали зображення або спектрограмного представлення аудіо. Ця операція використовує надзвичайно малу кількість параметрів, що спрощує процес навчання та прискорює роботу мережі. Це те саме, що і в клітинах зорової кори.

Примітно, що ці клітинки сприймають лише невеликі області сцени, а не всю сцену (тобто ці клітини просторово витягують локальну кореляцію, наявну на вході, наприклад локальні фільтри над входом).

Зазвичай використовуваний тип CNN, схожий на багатошаровий персептрон (MLP), складається з численних шарів згортки, що передують шарам підвибірки (об’єднання), тоді як кінцеві шари є шарами FC [13]. Приклад архітектури CNN для класифікації зображень проілюстрований на рисунку 2.3.

2.3 Аналіз цифрових методів пригнічення шуму

Швидке перетворення Фур’є. Швидке перетворення Фур’є (ШПФ) є одним з найважливіших алгоритмів обробки сигналів та аналізу даних.

ШПФ — це швидкий алгоритм $\vartheta[N \log N]$ для обчислення дискретного перетворення Фур’є (DFT), яке наївно є обчисленням $\vartheta[N^2]$. DFT, як і більш знайома безперервна версія перетворення Фур’є, має пряму та зворотну форми, які визначаються таким чином:

Пряме дискретне перетворення Фур’є (DFT):

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-i2\pi \quad /N}$$

Обернене дискретне перетворення Фур’є (IDFT):

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i2\pi \quad /N}$$

Перетворення з $x_n \rightarrow X_k$ є трансляцією з конфігураційного простору в частотний простір і може бути дуже корисним як для дослідження спектру потужності сигналу, так і для перетворення певних задач для більш

ефективного обчислення [14].

Аналіз Фур’є заснований на ідеї, що будь-який часовий ряд можна розкласти на суму інтеграла гармонійних хвиль різної частоти. Отже, теоретично можливо використовувати ряд гармонійних хвиль для створення будь-якого сигналу.

Часова складність FFT $O(n \log n)$, що є хорошим показником. Проте, слід розглянути можливість розпаралелення, задля покращення результату.

Під час розпаралелювання алгоритму слід враховувати, який алгоритм підходить для реалізації FFT. Рекурсивний спосіб для алгоритму FFT є простим для впровадження. Однак є дві причини використання ітераційного способу для алгоритму FFT. По-перше, ітеративна версія алгоритму FFT може виконувати менше індексів обчислення. По-друге, легше отримати паралельний алгоритм FFT, коли послідовний алгоритм в ітераційній формі.

Є три фрази для паралельного алгоритму. Припустимо, що n – кількість елементів, а p – кількість процесів (потоків). Спочатку процеси змінюють вхідну послідовність a, переставляють індекси. У другій фазі процеси виконують перші $\log n - \log p$ ітерації FFT, виконуючи необхідні множення, додавання та віднімання комплексних чисел. На третій фазі процеси виконують остаточні логарифмічні ітерації FFT і обмінюють значення з процесами, що беруть участь у вимірі гіперкуба. Отже, кожен процес контролює $\frac{n}{p}$ елементів вхідної послідовності a. Є $\log p$ ітерацій, в яких кожен процес обмінюється місцями приблизно $\frac{n}{p}$ значень з процесами, що беруть участь у обробці. Загальна часова складність – $O(\frac{n}{p} \log p)$, а обчислювальна складність паралельного алгоритму $O(n \log \frac{n}{p})$ [16].

Фільтр Вінера. Фільтрація Вінера є промисловим стандартом для динамічної обробки сигналів і широко використовується в слухових апаратах та інших зовнішніх пристроях, таких як телефони та пристрої зв’язку. Адаптивний фільтр найкраще працює за наявності двох аудіосигналів: один з мовним і фоновим шумом, а інший вимірює лише фоновий шум. Сучасні

дизайнери смартфонів часто розміщують два мікрофони на відстані один від одного так, щоб один був розміщений біля рота мовця для запису шумної мови, а інший міг виміряти навколишній шум, щоб відфільтрувати шум.

Фільтр Вінера використовує властивості цих двох сигналів для отримання оцінок чистого мовлення. Потім обчислюється й мінімізується помилка, відома як середньоквадратична помилка, щоб отримати найкращу оцінку чистого мовлення.

Фільтрація Вінера є оптимальною з точки зору середньоквадратичної помилки. Іншими словами, це мінімізує загальну середньоквадратичну помилку в процесі зворотної фільтрації та згладжування шуму. Підхід базується на стохастичній структурі. Принцип ортогональності передбачає, що фільтр Вінера в області Фур'є можна виразити так:

$$W(f_1, f_2) = \frac{H*(f_1,f_2)S_x\ (f_1,f_2)}{|H(f_1,f_2)|^2 S_x\ (f_1,f_2)+S_{¦¦}(f_1,f_2)},$$

де $S_x\ (f_1, f_2), S_{¦¦}(f_1, f_2)$ – відповідно спектри потужності вихідного сигналу та адитивного шуму, а $H(f_1, f_2)$ – фільтр розмиття. Легко помітити, що фільтр Вінера має дві окремі частини, частину інверсної фільтрації та частину згладжування шуму. Він не тільки виконує деконволюцію шляхом зворотної фільтрації (фільтрація високих частот), але також усуває шум за допомогою операції стиснення (фільтрація низьких частот) [15].

Метод спектрального віднімання. Метод спектрального віднімання широко використовується і є досить поширеним. Адитивні стаціонарні шуми – породжувані навколишнім середовищем, звукозаписною апаратурою тощо. Стаціонарність означає, що властивості шуму (потужність, спектральний склад) змінюються у часі. Адитивність означає, що шум сумується з «чистим» сигналом $y(t)$ і не залежить від нього:

$$x(t) = y(t) + n \qquad (t)$$

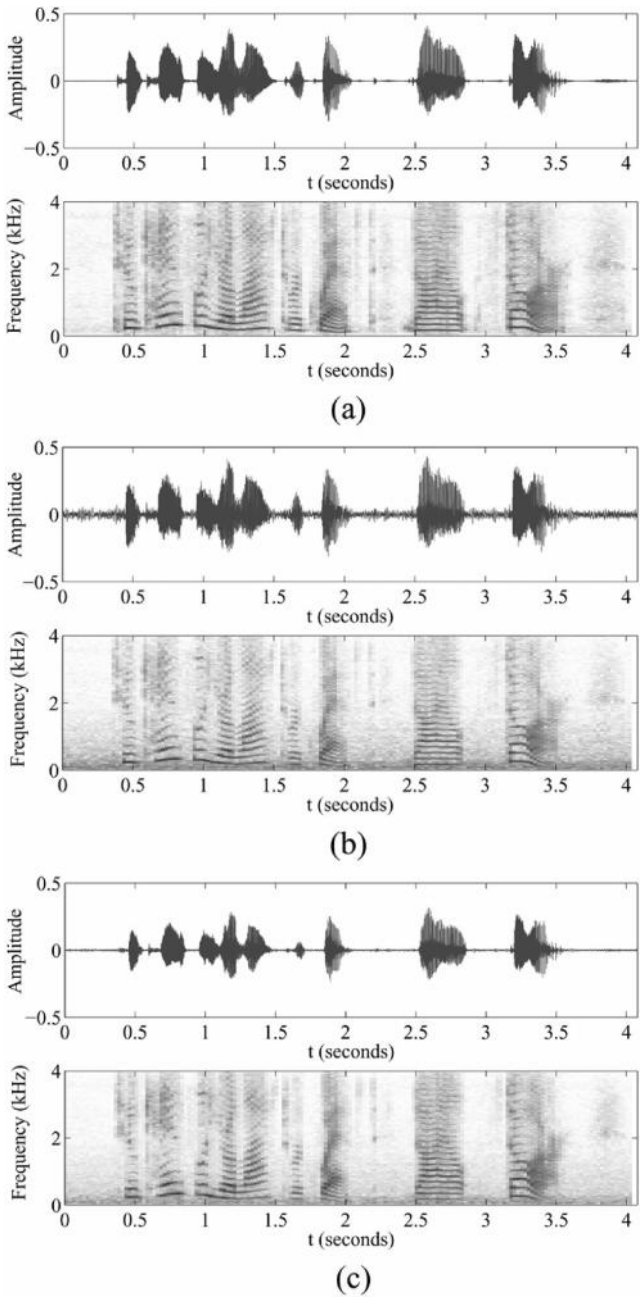


Рисунок 2.4 – Приклад використання фільтру Вінера: (a) чиста промова та її спектрограма; (b) зашумлена мова та її спектрограма; (c) відфільтрована мова та її спектрограма [16]

Для придушення адитивних стаціонарних шумів реалізовано алгоритм спектрального віднімання. Він складається з наступних стадій:

1. Розкладання сигналу за допомогою короткочасного перетворення Фур’є (STFT), що компактно локалізує енергію сигналу.
2. Складання віднімається амплітудного діапазону – noise footprint.

3. «Віднімання» амплітудного спектра шуму з амплітудного спектра сигналу.

4. Зворотне перетворення STFT – синтез результуючого сигналу.

Як банк фільтрів задіяно STFT з вікном Ханна довжини (FFT Size) і ступенем перекриття (Overlapping Level). Складання noise foorprint відбувається усередненням частот часу, взятих із спеціально підготовленого файлу з шумом, присутнім на зашумленій фонограмі. Віднімання амплітудних спектрів здійснюється за формулою, що еквівалентно наступній функції пригнічення:

$$Y(f,t) = \max\{X(f,t) - k * W(f,t), 0\}$$

Тут $X(f,t)$ і $W(f,t)$ – амплітудні спектри сигналу та шуму відповідно, $Y(f,t)$ – амплітудний спектр результуючого очищеного сигналу, а k – коефіцієнт придушення. Фазовий спектр очищеного сигналу зводиться до рівного фазовому спектру зашумленого сигналу.

Одна з проблем методу спектрального віднімання - «музичний шум». Він утворюється внаслідок того, що коефіцієнти STFT шумових сигналів статистично випадкові. Це призводить до їхнього нерівномірного придушення. В результаті, очищений сигнал містить короткочасні та обмежені по частоті сплески енергії, які на слух сприймаються як «дзвіночки» або «вода, що ллється». Для придушення цього артефакту застосовано метод рекурсивного згладжування першого порядку за часом оцінок спектра $X(f,t)$.

$$s[i] = s[i] + \alpha(f[i] - s[i]),$$

де $s(i)$ – згладжений спектр сигналу, а $f(i)$ – спектр вікна FFT, що обробляється, α – коефіцієнт усереднення.

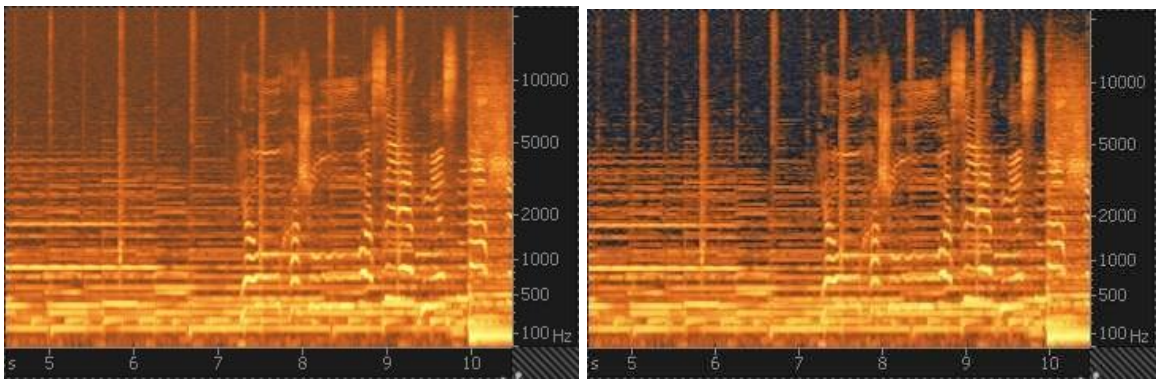


Рисунок 2.5 – Спектрограма зашумленого сигналу (ліворуч) та очищеного сигналу (праворуч)

Далі з усередненого спектру віднімається noise footprint і здійснюється зворотне перетворення Фур’є (ISTFT), з урахуванням перекриття вікон.

Швидкість алгоритму: 1 мс обробляється за 0.22 мс. Це дозволить використовувати алгоритм для потокових завдань [17].

Адаптивний фільтр. Адаптивний фільтр – це система з лінійним фільтром, яка складається з передатної функції, обмеженої змінними параметрами, і засобу для налаштування цих параметрів відповідно до алгоритму оптимізації. Адаптивні лінійні фільтри є лінійною динамічною системою зі змінною або адаптивною структурою і параметрами, і мають властивість змінювати значення своїх параметрів, тобто їх передатну функцію, під час обробки вхідного сигналу, щоб генерувати сигнал на виході, який без небажаних компонентів, шумів і погіршення, а також сигналів перешкод.

На рисунку показано основну концепцію адаптивного фільтра, основною метою якого є фільтрація вхідного сигналу $x(n)$ за допомогою адаптивного фільтра таким чином, щоб він відповідав бажаному сигналу $d(n)$. Бажаний сигнал $d(n)$ віднімається від відфільтрованого сигналу $y(n)$, щоб отримати сигнал помилки, який, у свою чергу, керує адаптивним алгоритмом, який генерує коефіцієнти фільтра таким чином, що мінімізує сигнал помилки. Адаптація коригує характеристики фільтра через взаємодію

з навколишнім середовищем, щоб досягти бажаних значень. На відміну від звичайних методів проектування фільтрів, адаптивні фільтри не мають постійних коефіцієнтів фільтра і невідома апріорна інформація, такі фільтри з регульованими параметрами називають адаптивним фільтром. Адаптивний фільтр коригує свої коефіцієнти, щоб мінімізувати сигнал помилки, і його можна назвати скінченною імпульсною характеристикою (FIR), нескінченною імпульсною характеристикою (IIR), фільтром решітки та області перетворення. Зазвичай адаптивні цифрові фільтри складаються з двох окремих блоків: цифрового фільтра зі структурою, визначеною для досягнення бажаної обробки (яка відома з точністю до невідомого вектору параметрів) і адаптивного алгоритму оновлення параметрів фільтра з метою гарантувати якнайшвидшу збіжність до оптимальних параметрів з точки зору прийнятого критерію. Більшість адаптивних алгоритмів означають модифікації стандартних ітераційних процедур для вирішення задачі мінімізації критеріальної функції в реальному часі. Найпоширенішою формою адаптивних фільтрів є поперечний фільтр із використанням алгоритму найменших середніх квадратів (LMS) та рекурсивного алгоритму найменших квадратів (RLS) [18].

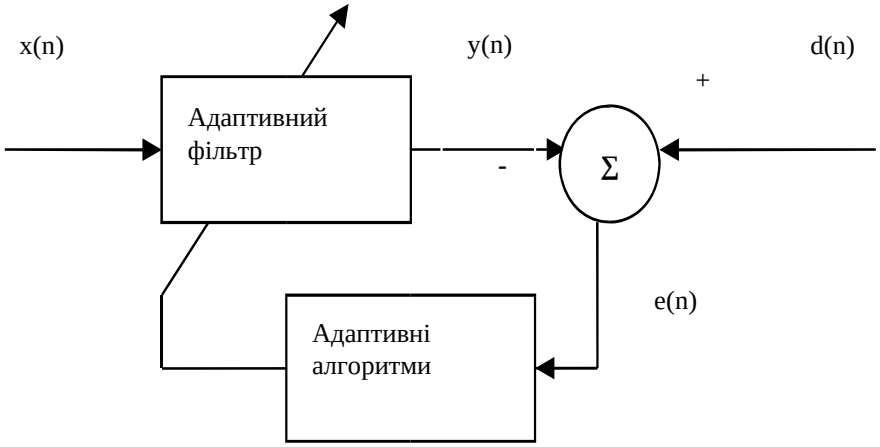


Рисунок 2.6 – Концепція адаптивного фільтру

Обчислювальна складність алгоритму фільтрації виражається залежністю N^2. У таких алгоритмах обчислення, обумовлені вхідними сигналами адаптивного фільтра, та обчислення, обумовлені регуляризацією, що не залежать один від одного, а отже, можуть виконуватися паралельно [19].

Метод активного шумопригнічення (ANC). Цей метод зазвичай використовується у навушниках. Активне шумозаглушення використовує мікрофони та динаміки для зменшення фонового та навколишнього шуму. Це найвідоміший вид, і в основному він використовується у повнорозмірних навушниках. Технологія стала настільки компактною і мінімально енерговитратною, що тепер може використовуватися в бездротових внутрішньоканальних навушниках.

Існує два його підвиди:

- адаптивне активне шумозаглушення використовує мікрофони та динаміки для автоматичного налаштування під навколишні шуми. Це складніший вид ANC, у якому рівень шумозаглушення у цифровій формі адаптується до довкілля;

- регульоване активне шумозаглушення дозволяє змінювати рівень фонового шуму, вручну регулюючи рівні шумозаглушення. Це зручно тоді, коли потрібний повний контроль.

Сам принцип технології дуже простий. Але на практиці досягти нормальних результатів надзвичайно складно через цілу низку технічних складностей.

По-перше, навушники власними силами забезпечують пасивну звукоізоляцію. Тому, щоб грамотно заміряти рівень шуму потрібно щонайменше два мікрофони: один зовні, що вловлює зовнішні шуми, і один усередині, щоб зрозуміти який відсоток зовнішніх шумів проникає всередину.

По-друге, якщо раптом інвертований сигнал відстане хоча б на 5 мілісекунд від реального звуку, шумозаглушення перестане працювати. Тому

в сучасних навушниках повинен бути потужний і дуже тонко налаштований цифровий процесор, який постійно адаптуватиметься під мінливу ситуацію і працюватиме з мінімальними затримками.

Особливо затримки негативно впливають на придушення високочастотних звуків, у яких коливання відбуваються тисячі разів на секунду, тому навіть найменша затримка в цьому випадку може навіть збільшити рівень шуму через накладання піків один на одного. Тому більш-менш непогано справляються з шумозаглушенням високих частот тільки найкращі навушники, але багато навушників не здатні придушити навіть людський голос.

Ну, а низькі частоти до 100 Гц взагалі не здатні погасити, адже маленьким динамікам просто не вистачить потужності. Це видно на графіках рисунку 2.7.

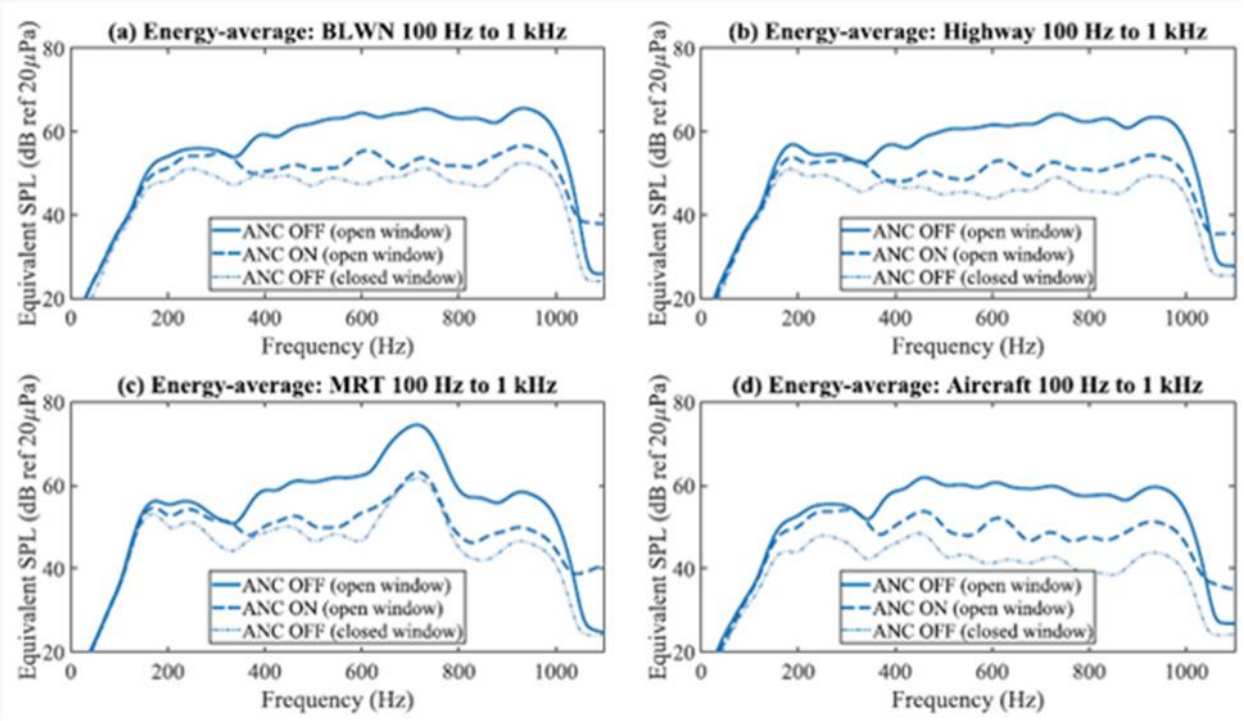


Рисунок 2.7 – Графіки прикладів роботи ANC з реальними шумами

Ну і по-третє, на фінальному етапі шумопригнічення потрібно

об’єднати дві хвилі: анти-шум та сигнал. Якщо це зробити бездумно, оригінальне аудіо може досить сильно спотворитись і якість звуку впаде. Тому знову ж таки потрібні розумні алгоритми, швидкий процесор і багато годин тестування та тонкого підстроювання [20].

Використання нейромереж. Фундаментальний документ щодо застосування глибокого навчання для придушення шуму, здається, був написаний Йонг Сю в 2015 році.

Йонг запропонував метод регресії, який навчається створювати маску співвідношення для кожної звукової частоти. Створена пропорційна маска нібито залишає людський голос недоторканим і видаляє сторонні шуми. Хоча це був далекий від досконалості, це був хороший ранній підхід.

У наступні роки було реалізовано багато різних запропонованих методів; підхід високого рівня майже завжди однаковий, що складається з трьох кроків, зображених на рисунку 2.8:

- збір даних: генерується великий набір даних синтетичного шумного мовлення, змішуючи чисте мовлення з шумом;
- навчання: передається цей набір даних DNN на вхід і чисту мову на виході;
- висновок: створюється модель (двійкова, пропорційна або комплексна), яка залишить людський голос і відфільтрує шум.

Використання глибокого навчання – один з провідних підходів видалення шуму з аудіо. Піонерами в області шумопригнічення за допомогою глибоких нейромереж, або DNN-шумопригнічення, стала компанія BubbleLabs.

Найбільш очевидними перевагами, в порівнянні з іншими фільтрами, є:

- видалення будь-якого роду шумів (усе залежить від кількості виборки для навчання);
- простота у реалізації (можливість використовування готових потужних бібліотек);
- можливість використовувати у режимі реального часу.

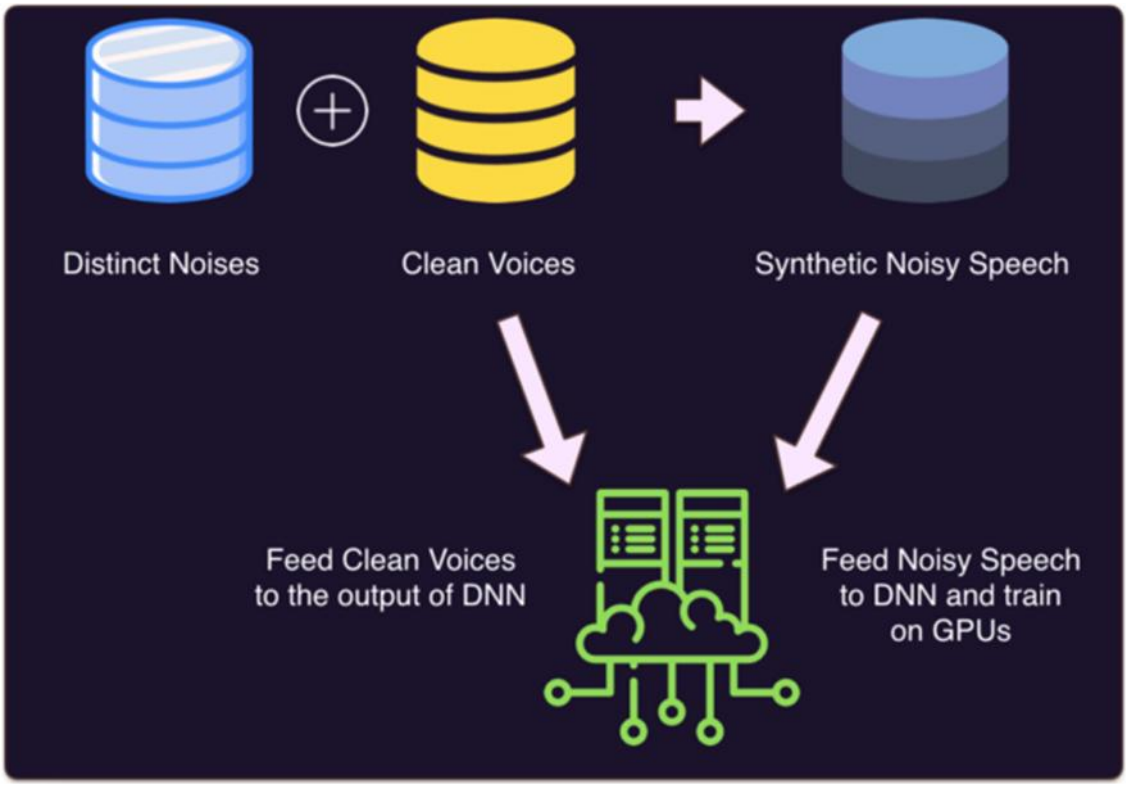


Рисунок 2.8 – Загальна концепція глибокого навчання для видалення шуму [21]

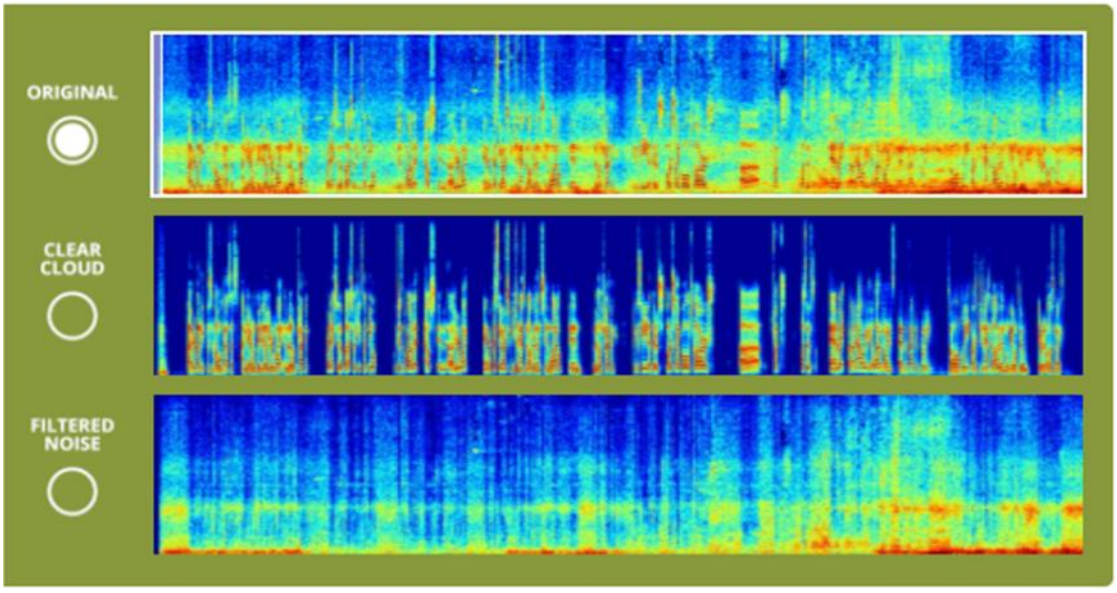


Рисунок 2.9 – Приклад видалення шуму представлений компанією Gabby’s Lab [22]

Можливості обчислити часову складність обробки сигналу немає, адже глибоке навчання припускає, що при навчанні буде сформована складна математична функція, яка б передбачила потрібну мету.

2.4 Вибір технології та мови програмування

2.4.1 Технології проектування інтелектуальних систем обробки та аналізу даних великого обсягу

На сьогоднішній день існують декілька мов програмування для вирішення задач нейронних мереж. Нижче представлені 5 найбільш популярних:

- Python вважається на першому місці в списку всіх мов розробки ШІ через простоту. Синтаксиси, що належать до python, дуже прості і їх можна легко вивчити. Тому в ньому можна легко реалізувати багато алгоритмів ШІ. Python займає короткий час на розробку в порівнянні з іншими мовами, такими як Java, C++ або Ruby. Python підтримує об’єктно-орієнтований, функціональний, а також процедурно-орієнтований стилі програмування. У Python є багато бібліотек, які полегшують завдання реалізації. Наприклад: Numpy – це бібліотека для Python, яка допомагає вирішувати багато наукових обчислень. Крім того, у існує Pybrain, який призначений для використання машинного навчання в Python;

- R є однією з найефективніших мов і середовища для аналізу та маніпулювання даними для статистичних цілей. Використовуючи R, можливо легко створити добре розроблений сюжет якості публікації, включаючи математичні символи та формули, де це необхідно. Окрім мови загального призначення, R має безліч пакетів, таких як RODBC, Gmodels, Class і Tm, які використовуються в області машинного навчання. Ці пакети полегшують реалізацію алгоритмів машинного навчання для вирішення проблем, пов’язаних з бізнесом;

- Lisp – одна з найстаріших і найбільш придатних мов для розробки AI. Його винайшов Джон Маккарті, батько штучного інтелекту в 1958 році. Він має здатність ефективно обробляти символічну інформацію. Він також відомий своїми чудовими можливостями створення прототипів і легким динамічним створенням нових об’єктів з автоматичним збором сміття. Його цикл розробки дозволяє проводити інтерактивну оцінку виразів і перекомпіляцію функцій або файлів під час роботи програми;

- Ця мова залишається поряд з Lisp, коли мова йде про розвиток у галузі ШІ. Функції, надані ним, включають ефективне зіставлення шаблонів, структурування даних на основі дерева та автоматичне повернення назад. Усі ці функції забезпечують потужну та гнучку основу програмування. Prolog широко використовується для роботи над медичними проектами, а також для проектування експертних систем штучного інтелекту;

- Java також можна вважати хорошим вибором для розробки AI. Штучний інтелект має багато спільного з алгоритмами пошуку, штучними нейронними мережами та генетичним програмуванням. Java надає багато переваг: простота використання, простота налагодження, пакетні послуги, спрощена робота з великомасштабними проектами, графічне представлення даних і краща взаємодія з користувачем. Він також включає в себе Swing і SWT (стандартний набір інструментів віджетів). Ці інструменти роблять графіку та інтерфейси привабливими та витонченими [23].

У таблиці 2.1 представлений список переваг та недоліків кожної з мов програмування для задач ШІ.

Співставивши усі фактори, тенденції та характеристики мов програмування, був обраний саме Python для реалізації та оптимізації алгоритму.

Таблиця 2.1 – Переваги та недоліки мов програмування для вирішення задач штучного інтелекту

Мова програмування	**Переваги**	**Недоліки**
Python	+ великий набір готових рішень, що продовжує зростати (у вигляді бібліотек, що досі розвиваються і підтримуються) + короткий час на розробку + Open Source + простота синтаксису + велика продуктивність	– є нетипізованою мовою
R	+ великий набір готових рішень (менший в порівнянні з Python) + Open Source	– важкий у вивченні – при неправильному використанні може бути повільним
Lisp	+ легкий у вивченні + легкий у налагодженні	– важко взаємодіяти з основними середовищами (.NET, Java) – є досить старою мовою програмування
Prolog	+ зрозумілий	– мала продуктивність – не підходить для рішення комплексних задач – повільно розвивається
Java	+ є типізованою мовою + великий вибір готових рішень	– нижча продуктивність (у порівнянні з Python)

2.4.2 Бібліотеки для процесінгу звуку для системи з масовим паралелізмом

Паралелізм та узгодженість. «Багатопроцесорний» модуль у Python дозволяє програмісту повністю використовувати кілька процесорів на даній машині. Використовуваний API подібний до класичного модуля потоків. Він пропонує як локальний, так і віддалений паралельний режим.

Багатопроцесорний модуль уникає обмежень Глобального блокування перекладача (GIL), використовуючи підпроцеси замість потоків. Багатопроцесорний код не виконується в тому ж порядку, що і послідовний код. Не існує гарантії, що перший створений процес буде завершений першим. GIL – це механізм, що використовується в інтерпретаторі Python для синхронізації виконання потоків так, що одночасно може виконуватись лише один нативний потік, навіть якщо він працює на багатоядерному процесорі.

Розширення C, такі як numpy, можуть вручну звільнити GIL для прискорення обчислень. GIL було випущено перед тим, як потенційно блокуючі операції введення-виводу. Jython, і IronPython не мають GIL.

Узгодженість означає, що два або більше обчислень виконуються в межах одного проміжку часу. Паралелізм означає, що два або більше обчислень виконуються в один і той же момент. Таким чином, паралелізм є специфічним випадком узгодженості. Для цього потрібно кілька процесорних блоків або ядер.

Справжній паралелізм у Python досягається шляхом створення кількох процесів, кожен з яких має інтерпретатор Python із власним GIL [24].

Python має три модулі для паралельності: багатопроцесорний, багатопотоковий та асинхронний. Коли завдання інтенсивні для ЦП, варто розглянути багатопроцесорний модуль. Якщо завдання пов’язані з введенням-виводом і вимагають великої кількості з’єднань, рекомендується використовувати модуль asyncio. Для інших типів завдань і коли бібліотеки не можуть співпрацювати з asyncio, можна розглянути модуль потоків.

Термін «eberrassinbly parallel» використовується для опису проблеми або робочого навантаження, які можна легко виконувати паралельно. Важливо розуміти, що не всі робочі навантаження можна розділити на підзадачі і виконувати їх паралельно. Наприклад, ті, які вимагають багато спілкування між підзадачами.

Приклади ідеально паралельних обчислень включають:

- аналіз Монте-Карло;
- числове інтегрування;
- рендеринг комп’ютерної графіки;
- пошуки грубою силою в криптографії;
- генетичні алгоритми.

Інша ситуація, коли можна застосувати паралельні обчислення, – це коли виконується кілька різних обчислень, тобто проблема не ділиться на підзадачі. Наприклад, можливо б виконувати обчислення π за допомогою різних алгоритмів паралельно.

Об’єкт Process представляє діяльність, яка виконується в окремому процесі. Клас multiprocessing.Process має еквіваленти всіх методів threading.Thread. Конструктор Process завжди слід викликати з аргументами ключового слова.

Аргумент цільового конструктора – це об’єкт, що викликається, який викликається методом run. Ім’я – це ім’я процесу. Метод start запускає процес. Метод приєднання блокується, доки не завершиться процес, чий метод об’єднання називається. Якщо передбачено параметр тайм-аут, він блокує процес після вказаних максимум секунд очікування. Метод is_alive повертає логічне значення, яке вказує, чи є процес живим. Метод terminate завершує процес [25].

І процеси, і потоки є незалежними послідовністю виконання. У таблиці 2.2 підсумовано відмінності між процесом і потоком.

Таблиця 2.2 – Порівняння процесу та потоку

Process	**Thread**
процеси виконуються в окремій пам’яті (ізоляція процесів)	потоки поділяють загальну пам’ять
використовує більше пам’яті	використовує менше пам’яті
дочірні можуть стати зомбі	зомбі не є можливими
більше накладних витрат	менше накладних витрат
повільніше створюються і руйнуються	швидше створювати і руйнувати
легше кодувати та налагоджувати	кодування та налагодження може стати складнішим

Навіть не зважаючи на порівняльну характеристику Процесу та Потока, Процес не є відповідним для вирішення задач ШІ. Тож, потоки є ідеальними для реалізації та оптимізації подібних задач.

3 ІНТЕГРАЦІЯ МЕТОДІВ ПРИГНІЧЕННЯ ШУМУ В КОМП’ЮТЕРНІЙ СИСТЕМІ РОЗПІЗНАВАННЯ АКУСТИЧНИХ ПОДІЙ

3.1 Система розпізнавання акустичних подій

Для дослідження методів пригнічення шуму в аудіоряді було розроблено комп’ютерну систему для розпізнавання акустичних подій, а саме: музики та її жанрів, звуків тварин з розподіленням на класи, транспорту та його типу, випадкових подій (стук, вибух, постріл та ін.), людської промови, природних явищ і т. д.

Система призначена для ідентифікації типу події на заданому аудіоряді будь-якого розміру та може бути використана як аналог ShotSpotter (розділ 1.4) чи для усунення «небажаних» артефактів при використанні програмного забезпечення для аудіо (відео-) конференцій.

Система вміє працювати за трьома наступними напрямками:

- попередньо записаний аудіо файл;
- захоплення та обробка даних з мікрофона;
- веб-інтерфейс.

Система створена на мові Python, що дозволяє її легко адаптувати та оптимізувати. Вона використовує попередньо задану модель нейромережі типу CNN. Модель побудована на основі набору даних Google AudioSet (розмір навчальної вибірки становить приблизно 2 мільйони сегментів з різних відео файлів, розмір тестової вибірки становить 20 тисяч сегментів з різних відео файлів), що є у відкритому доступі [26]. Загалом система використовує наступні зовнішні Python бібліотеки:

- numpy;
- scipy;
- resampy;

- PyAudio;
- Tensorflow;
- Six;
- Devicehive.

Окрім цього, система (зокрема бібліотека Tensorflow) потребує наступне програмне забезпечення [27]:

- NVIDIA® GPU drivers;
- CUDA® Toolkit;
- CUPTI;
- cuDNN SDK.

Окрім цього, слід зазначити, що нейромережа навчена на основі моделі VGGish платформи Tensorflow.

Рисунок 3.1 – Узагальнена схема роботи системи розпізнавання акустичних артефактів

Через те, що графічні процесори більше підходять для навчання нейромереж, а платформа Tensorflow надає можливість використання графічного процесору (при наявності вище зазначеного програмного забезпечення), було використане наступне апаратне забезпечення: NVidia GeForce RTX 2070 Super 8 GB.

На рисунку 3.1 представлена загальна IDEF0 схема роботи системи.

На схемі зліва зображено вхідні дані у систему, а саме аудіоряд (аудіофайл), який має бути ідентифікований до одного чи декілька класів. Зверху зображені інструменти, якими користується система, а саме:

- платформа TensorFlow, потрібна для роботи моделі нейромережі;
- колекція класів, потрібна для відношення вхідного аудіоряду до того чи іншого класу;
- YouTube-8M модель, інтерфейс для роботи з моделлю нейромережі;
- модулі Python (бібліотеки), потрібні для роботи кожної з частин системи (наприклад os використовується для роботи системи з локальними файлами).

Нижні входи, це інструменти та забезпечення, яким система користується. CPU та навчена VGGish модель нейромережі є обов’язковими складовими системи, у той час, як GPU – необов’язковою, проте бажаною. VGGish – це попередньо навчена згорткова нейронна мережа від Google [28]. Присутність GPU у системі покращує швидкість роботи.

На рисунку 3.2 представлена функціональна модель системи.

Запропонована система складається з модулів:

- перетворення файлу у масив бітів;
- перетворення масиву бітів у мел-спектрограму;
- перетворення мел-спектрограми у фрейми;
- отримання ключових характеристик;
- обробки ключових характеристик;
- фільтрації передбачень.

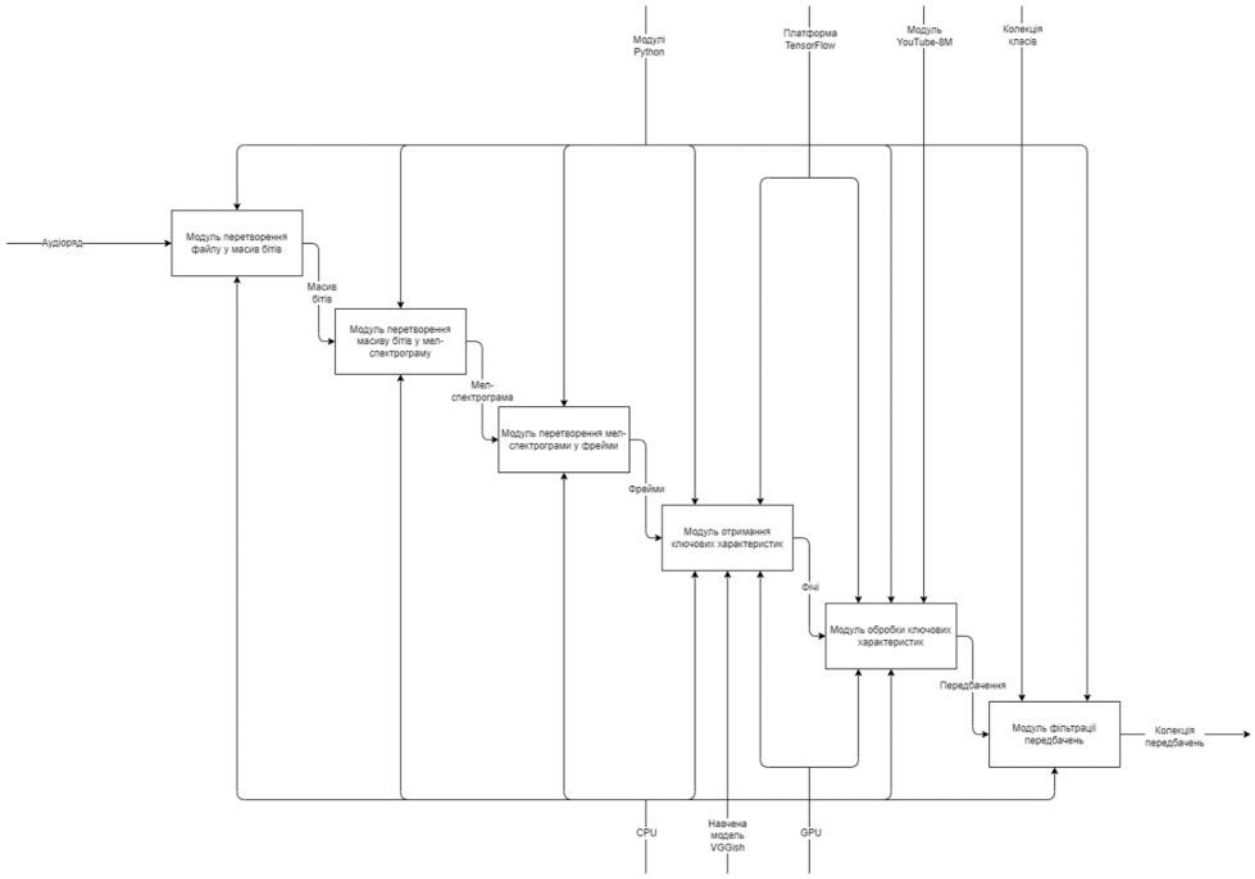


Рисунок 3.2 – Функціональна модель системи

До задач модулю перетворення файлу у масив бітів відносяться:

- зчитування файлу у потік;
- перетворення потоку у масив бітів;
- перевірка типу масиву.

Модуль приймає на вхід шлях до файлу та видає на виході масив із бітів. Модуль представлений у лістингу 3.1.

Лістинг 3.1 – Модуль перетворення файлу у масив бітів

```
sr, data = wavfile.read(wav_file)
if data.dtype != np.int16:
    raise TypeError('Bad sample type: %r' % data.dtype)
```

Наступним йде модуль перетворення масиву бітів у мел-спектрограму.

До його задач входять:

- перетворення 2 та більше канального сигналу у моно;
- короткочасне (віконне) перетворення Фур’є;
- отримання логарифмічної амплітудної мел-частотної спектрограми.

Короткочасне перетворення Фур’є виконується для можливості отримання стабілізованної спектрограми. У лістингу 3.2 представлений модуль.

Лістинг 3.2 – Модуль перетворення масиву бітів у мел-спектрограму

```
    if len(data.shape) > 1:
        data = np.mean(data, axis=1)
    if sample_rate != params.SAMPLE_RATE:
        data = resampy.resample(data, sample_rate,
params.SAMPLE_RATE)

    log_mel = mel_features.log_mel_spectrogram(
        data,
        audio_sample_rate=params.SAMPLE_RATE,
        log_offset=params.LOG_OFFSET,
        window_length_secs=params.STFT_WINDOW_LENGTH_SECONDS,
        hop_length_secs=params.STFT_HOP_LENGTH_SECONDS,
        num_mel_bins=params.NUM_MEL_BINS,
        lower_edge_hertz=params.MEL_MIN_HZ,
        upper_edge_hertz=params.MEL_MAX_HZ)

def log_mel_spectrogram(data,
                        audio_sample_rate=8000,
                        log_offset=0.0,
                        window_length_secs=0.025,
                        hop_length_secs=0.010,
                        **kwargs):
    window_length_samples = int(round(audio_sample_rate *
window_length_secs))
    hop_length_samples = int(round(audio_sample_rate *
hop_length_secs))
    fft_length = 2 ** int(np.ceil(np.log(window_length_samples)
/ np.log(2.0)))
    spectrogram = stft_magnitude(
        data,
        fft_length=fft_length,
        hop_length=hop_length_samples,
        window_length=window_length_samples)
    mel_spectrogram = np.dot(spectrogram,
spectrogram_to_mel_matrix(
```

Продовження лістингу 3.2

```
        num_spectrogram_bins=spectrogram.shape[1],
        audio_sample_rate=audio_sample_rate, **kwargs))
    return np.log(mel_spectrogram + log_offset)
```

Далі отримана спектрограма подається на модуль перетворення мел-спектрограми у фрейми. Його задача лише одна – розбиття спектрограми на сегменти по 10 мілісекунд. На виході отримується масив із фреймів (сегментів). Модуль представлений у лістингу 3.3.

Лістинг 3.3 – Модуль перетворення мел-спектрограми у фрейми

```
    features_sample_rate = 1.0 / params.STFT_HOP_LENGTH_SECONDS
    example_window_length = int(round(
        params.EXAMPLE_WINDOW_SECONDS * features_sample_rate))
    example_hop_length = int(round(
        params.EXAMPLE_HOP_SECONDS * features_sample_rate))
    log_mel_examples = mel_features.frame(
        log_mel,
        window_length=example_window_length,
        hop_length=example_hop_length)
    return log_mel_examples
def frame(data, window_length, hop_length):
    num_samples = data.shape[0]
    num_frames = 1 + int(np.floor((num_samples - window_length)
/ hop_length))
    shape = (num_frames, window_length) + data.shape[1:]
    strides = (data.strides[0] * hop_length,) + data.strides
    return np.lib.stride_tricks.as_strided(data, shape=shape,
strides=strides)
```

Наступний крок – модуль отримання ключових характеристик. Суть цього модуля це отримання тензорів та ключових характеристик з набору фреймів (лістинг 3.4).

Лістинг 3.4 – Модуль отримання ключових характеристик

```
    def _get_features(self, examples_batch):
        sess = self._vggish_sess
        features_tensor = sess.graph.get_tensor_by_name(
            params.VGGISH_INPUT_TENSOR_NAME)
```

Продовження лістингу 3.4

```
        embedding_tensor = sess.graph.get_tensor_by_name(
            params.VGGISH_OUTPUT_TENSOR_NAME)

        [embedding_batch] = sess.run(
            [embedding_tensor],
            feed_dict={features_tensor: examples_batch}
        )

        postprocessed_batch = np.dot(
            self._pca_matrix, (embedding_batch.T -
self._pca_means)
        ).T

        return postprocessed_batch
```

Задачами передостаннього модулю обробки ключових характеристик є:

- попередня обробка ключових характеристик модулем YouTube-8M;
- отримання передбачень із ключових характеристик.

Його реалізація представлена у лістингу 3.5.

Лістинг 3.5 – Модуль обробки ключових характеристик

```
    def _process_features(self, features):
        sess = self._youtube_sess
        num_frames = np.minimum(features.shape[0],
params.MAX_FRAMES)
        data = youtube8m.input.resize(features, 0,
params.MAX_FRAMES)
        data = np.expand_dims(data, 0)
        num_frames = np.expand_dims(num_frames, 0)

        input_tensor =
sess.graph.get_collection(«input_batch_raw»)[0]
        num_frames_tensor =
sess.graph.get_collection(«num_frames»)[0]
        predictions_tensor =
sess.graph.get_collection(«predictions»)[0]

        predictions_val, = sess.run(
            [predictions_tensor],
            feed_dict={
                input_tensor: data,
                num_frames_tensor: num_frames
            })
        return predictions_val
```

І, нарешті, останній модуль – фільтрація передбачень. У даному випадку під фільтрацією мається на увазі відкидання передбачень, які не перейшли заданий поріг, а також взяття заданої максимально допустимої кількості одиниць результатів, відсортованих по спаданню (лістинг 3.6).

Лістинг 3.6 – Модуль фільтрації передбачень

```
    def _filter_predictions(self, predictions):
        count = params.PREDICTIONS_COUNT_LIMIT
        hit = params.PREDICTIONS_HIT_LIMIT

        top_indices = np.argpartition(predictions[0], -count)[-
count:]
        line = ((self._class_map[i], float(predictions[0][i])) 
for
                i in top_indices if predictions[0][i] > hit)
    return sorted(line, key=lambda p: -p[1])
```

Результат отримується у вигляді відносності співпадіння до того чи іншого класу. Тобто, у вигляді назви класу та коефіцієнта величини співпадіння від 0 до 1.

3.2 Адаптація запропонованих методів шумопригнічення під обчислювальні системи із різними типами організації пам’яті

Основою даної адаптації є інтеграція різних методів шумопригнічення у систему розпізнавання акустичних подій задля покращення результатів розпізнавання. На рисунку 3.3 зображена функціональна модель системи з інтегрованим модулем шумопригнічення.

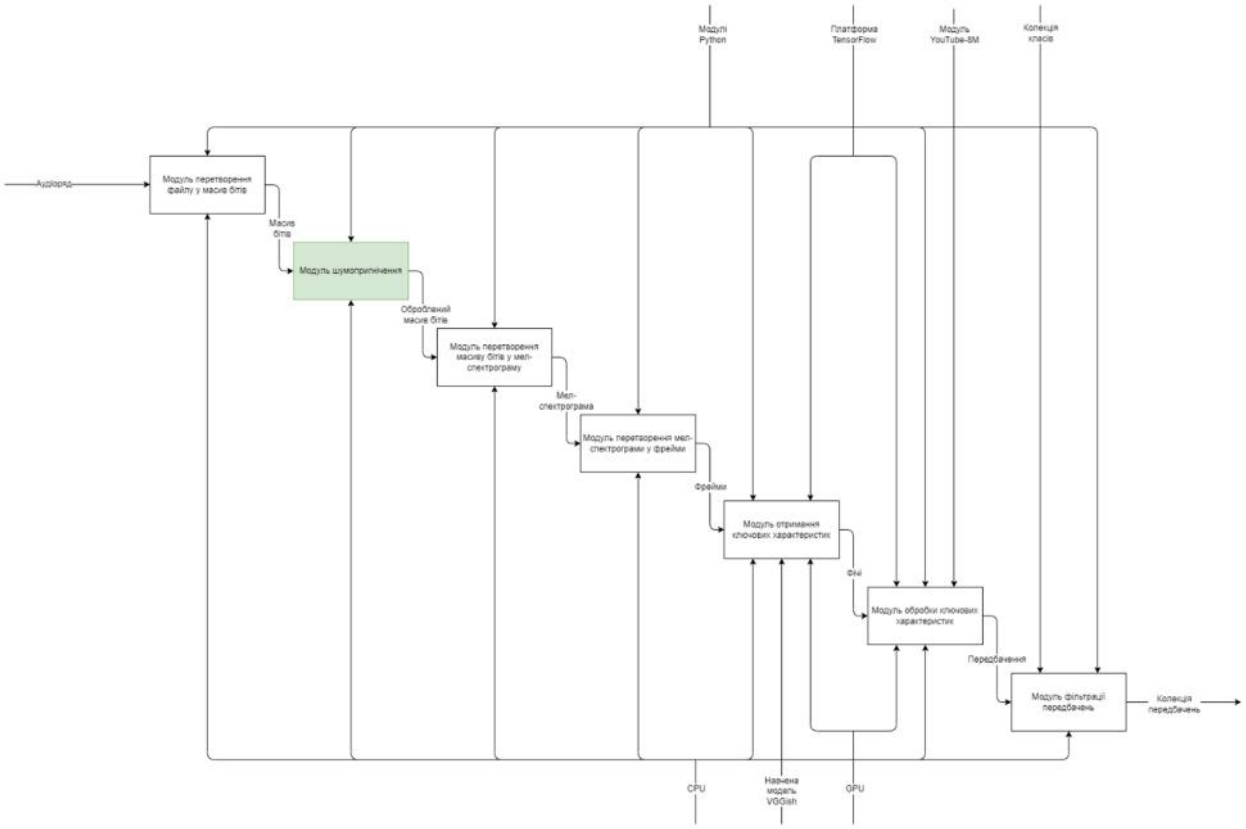


Рисунок 3.3 – Функціональна модель системи з інтегрованим алгоритмом шумопригнічення

Задача у цього модуля лише одна – попередня обробка аудіоряду одним із методів шумопригнічення. На вхід подається масив бітів, на виході отримується масив того ж типу, але оброблений. Це дозволяє не змінювати інші компоненти системи.

Для інтеграції було розглянуто 3 методи шумопригнічення:

- метод спектрального віднімання;
- швидке перетворення Фур’є;
- фільтр Вінера.

Кожен з них представляє свою версію модулю шумопригнічення, який може бути інтергований у систему (рисунок 3.4).

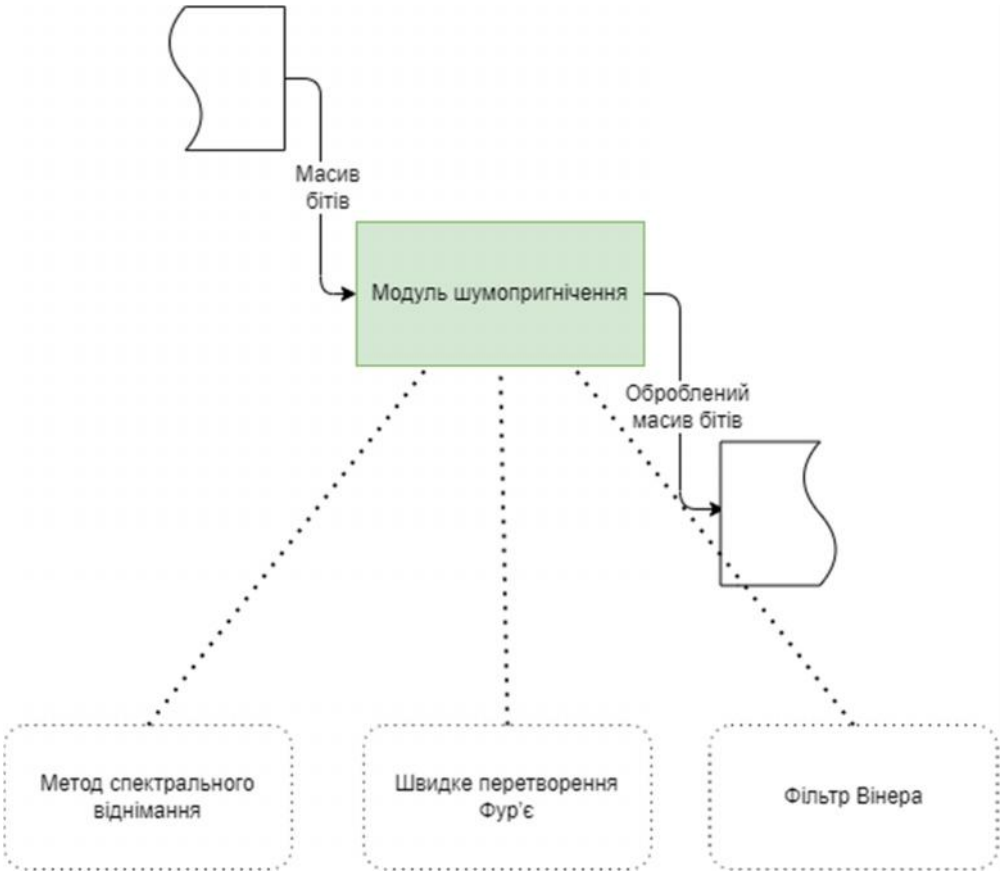


Рисунок 3.4 – Модуль шумопригнічення різних варіацій

Реалізація кожного з методів потребує розуміння області використання системи. Тобто, потрібно розуміти, яку саме подію системі потрібно ідентифікувати. Наприклад, системі потрібно розпізнати гавкіт собаки, то «корисний сигнал» знаходиться у межах від 300 до 3500 Гц, тому усе інше можна вважати «шумом».

Реалізація методу спектрального віднімання існує у бібліотеці noisereduce, тому модуль виглядає наступним чином (лістинг 3.7):

Лістинг 3.7 – Модуль шумопригнічення з використанням методу спектрального віднімання

```
    def spectral_subtraction(data, rate):
        data = nr.reduce_noise(y = data, sr = rate,
freq_mask_smooth_hz=3000, n_jobs=-1)
        return data
```

Той самий модуль, але з інтеграцією Швидкого перетворення Фур’є представлений у лістингу 3.8.

Лістинг 3.8 – Модуль шумопригнічення з використанням швидкого перетворення Фур’є

```
def fft_transform(data, rate):
    duration = data.shape[0] / rate
    N = rate * duration
    yf = rfft(data)
    xf = rfftfreq(int(N), 1 / rate)

    points_per_freq = len(xf) / (rate / 2)

    target_idx = int(points_per_freq * 3000)

    while(target_idx < len(yf)):
        yf[target_idx] = yf[target_idx]/2
        target_idx += 1

    data = irfft(yf).astype(np.int16)
    return data
```

І останній варіант модуля – з використанням фільтра Вінера. Бібліотека sciру має готову реалізацію фільтру, тому модуль буде мати наступний вигляд (лістинг 3.9):

Лістинг 3.9 – Модуль шумопригнічення з використанням швидкого перетворення Фур’є

```
def wiener(data, rate):
    data = signaltools.wiener(data, mysize=10, noise=0.5)
    return data
```

Задля задачі аналізу системи, кожен з цих модулів має бути інтегрованим окремо.

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Аналіз якості розпізнавання розробленої системи в залежності від методів пригнічення шуму

Аналіз системи був проведений без інтегрованого модулю шумопригнічення та з трьома методами шумопригнічення. Поріг ідентифікації – 0.8. Тип шуму – синтетично згенерований білий шум різного ступеню сили. У таблиці 4.1 представлений приклад порівняння результатів ідентифікації події використовуючи різні методи шумопринічення. Значення у таблиці – коефіцієнт ідентифікації, за яким система розпізнала подію. За приклад був взятий аудіоряд з «гавкіт собаки».

Таблиця 4.1 – Порівняння методів шумопригнічення в залежності від рівня білого шуму

Використаний метод шумопригнічення	Сила шуму		
	0.1	0.5	1.0
Метод шумопригнічення відсутній	0,93	0,80	-
Метод спектрального віднімання	0,96	0,96	0,90
Швидке перетворення Фур’є	0,95	0,94	0,92
Фільтр Вінера	0,97	1,00	1,00

Тестова вибірка складається з 200 аудіофалів, 50 з яких «гавкіт собаки». Загальний набір результатів представлений на рисунках 4.1-4.3.

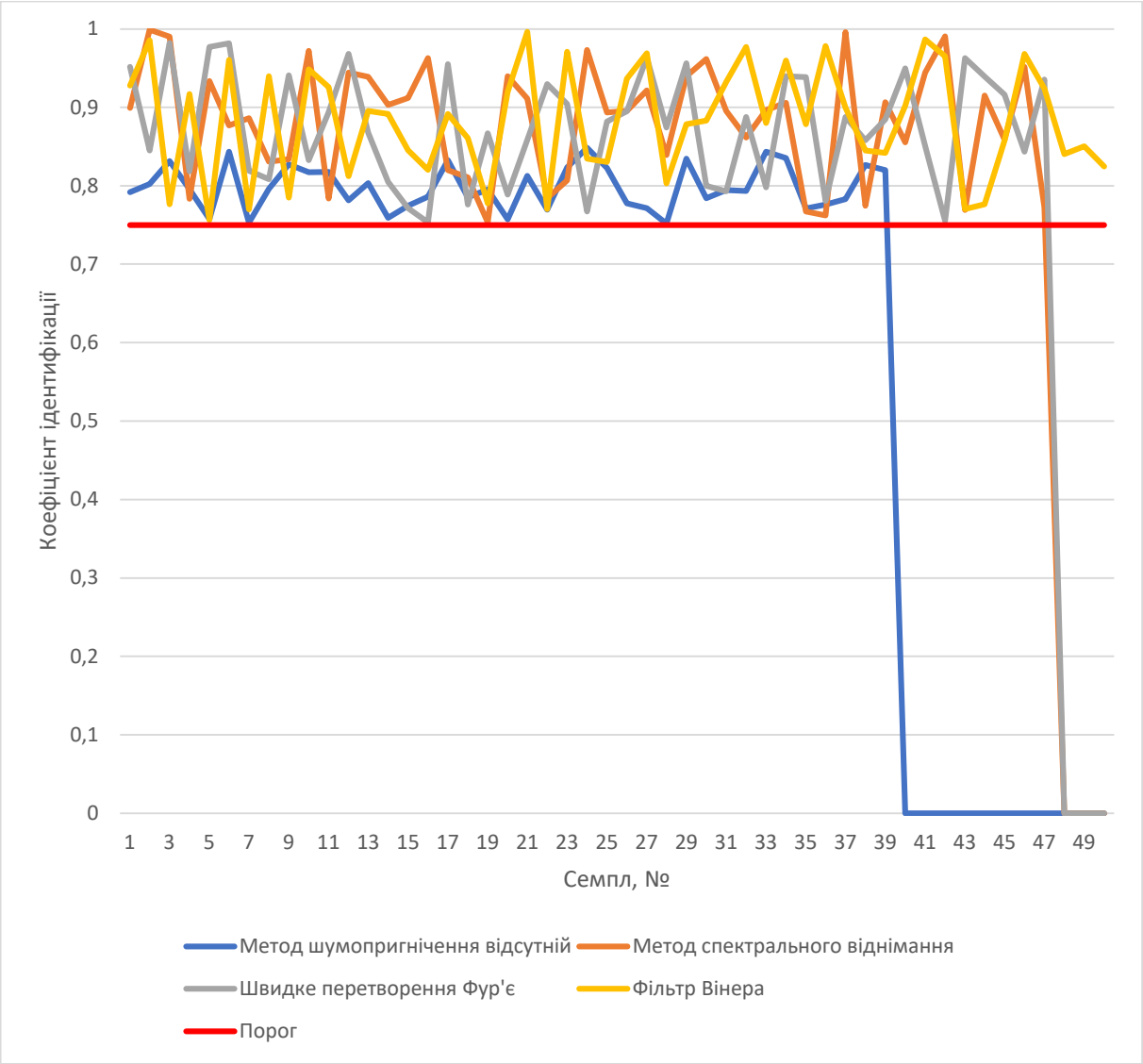


Рисунок 4.1 – Результати ідентифікації аудіоподій з рівнем шуму 0.1

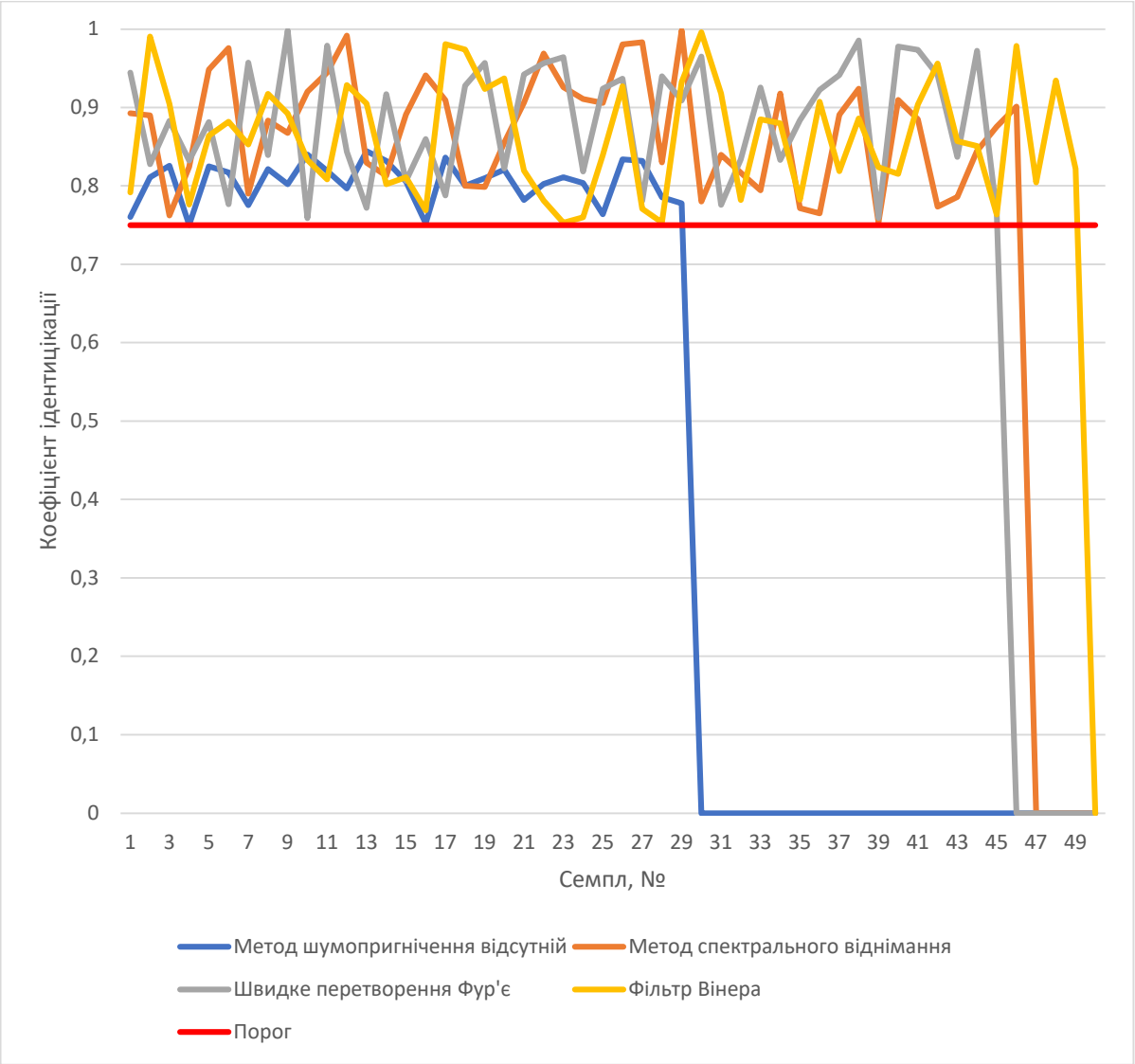


Рисунок 4.2 – Результати ідентифікації аудіоподій з рівнем шуму 0.5

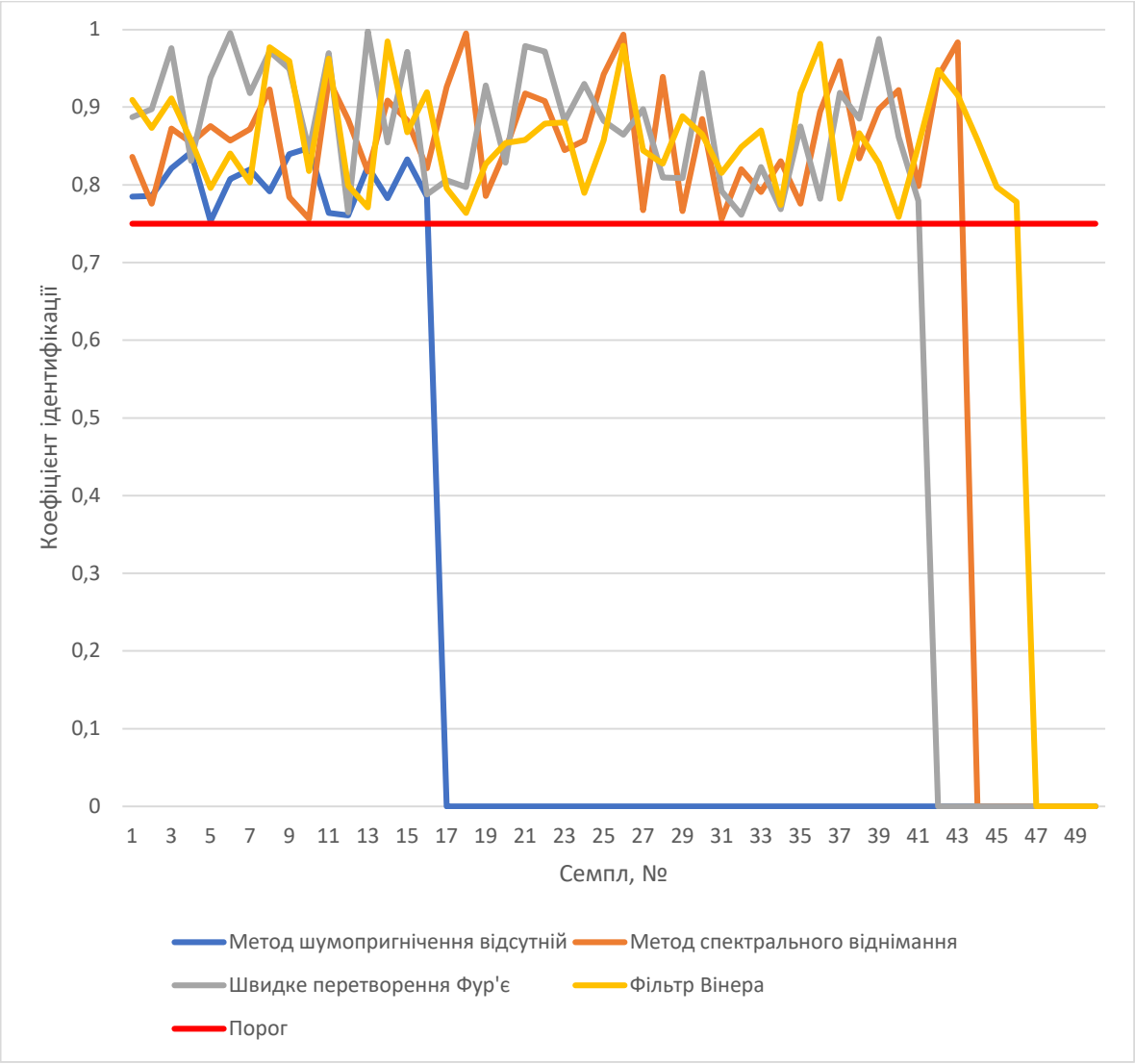


Рисунок 4.3 – Результати ідентифікації аудіоподій з рівнем шуму 1.0

Загальна оцінка системи була проведена за допомогою метрик точності (precision) та повноти (recall) з рівнем шуму 0.5. Точність системи в межах класу – це частка документів, що дійсно належать даному класу щодо всіх документів, які система віднесла до цього класу. Повнота системи – це частка знайдених класфікатором документів, що належать класу, щодо всіх документів цього класу в тестовій вибірці [29].

Ці значення легко розрахувати виходячи з таблиці контингентності, що складається кожного класу окремо.

Таблиця 4.2 – Таблиця контингентності

Категорія і		Експертна оцінка	
		Позитивна	Негативна
Оцінка системи	Позитивна	TP	FP
	Негативна	FN	TN

У таблиці міститься інформація скільки разів система прийняла правильне і скільки разів неправильне рішення щодо документів заданого класу. А саме:

- TP – істино-позитивне рішення;
- TN – істино-негативне рішення;
- FP – хибно-позитивне рішення;
- FN – хибно-негативне рішення.

Тоді, точність та повнота визначаються таким чином:

$$P = \frac{T}{T + F}$$

$$R = \frac{T}{T + F}$$

При використанні системи без шумопригнічення, вона помітила 32 семпли як «гавкіт собаки», серед яких 29 правильно розпізнаних і 3 не відносяться до цього класу.

Тобто маємо: TP = 29, FN = 21, FP = 3. Тоді, точність буде 0,90625, а повнота – 0,58.

При під’єднанні модуля з шумопригніченням метода спектрального віднімання було отримано TP = 46, FN = 4, FP = 1. Тому точність 0,97872, а повнота – 0,92.

При використанні швидкого перетворення Фур’є TP = 45, FN = 5, FP =

3, точність – 0,9375, повнота – 0,9.

І при використанні фільтра Вінера було отримано наступні результати: TP = 49, FN = 1, FP = 0. Тому точність буде 1, а повнота – 0,98.

У таблиці 4.3 представлені узагальнені результати тестування.

Таблиця 4.3 – Узагальнені результати оцінки системи ідентифікації подій

Метод шумопригнічення	Оцінка системи			Метрики		Відносна оцінка оптимізації
	TP	FN	FP	Precision	Recall	
Метод шумопригнічення відсутній	29	21	3	0,90625	0,58	-
Метод спектрального віднімання	46	4	1	0,97872	0,92	21%
Швидке перетворення Фур’є	45	5	3	0,9375	0,9	18%
Фільтр Вінера	49	1	0	1	0,98	25%

Із таблиці 4.3 можна зробити висновок, що фільтр Вінера найкраще справляється з задачею шумопригнічення та покращує результати ідентифікації подій.

4.2 Аналіз часу роботи системи для різних методів пригнічення шуму

Аналіз часу роботи системи проводився паралельно з аналізом ідентифікації. Тому, вибірка для тестування – ті самі 200 аудіофайлів, однакової тривалості. Відлік часу розпізнавання починається до початку роботи першого модуля – модуль перетворення файлу у масив бітів, а кінцева точка розрахунку після роботи останнього модулю – модуль фільтрації

передбачень.

На рисунку 4.2 представлений сумарний графік часу ідентифікації кожного семплу з кожним методом шумопригнічення.

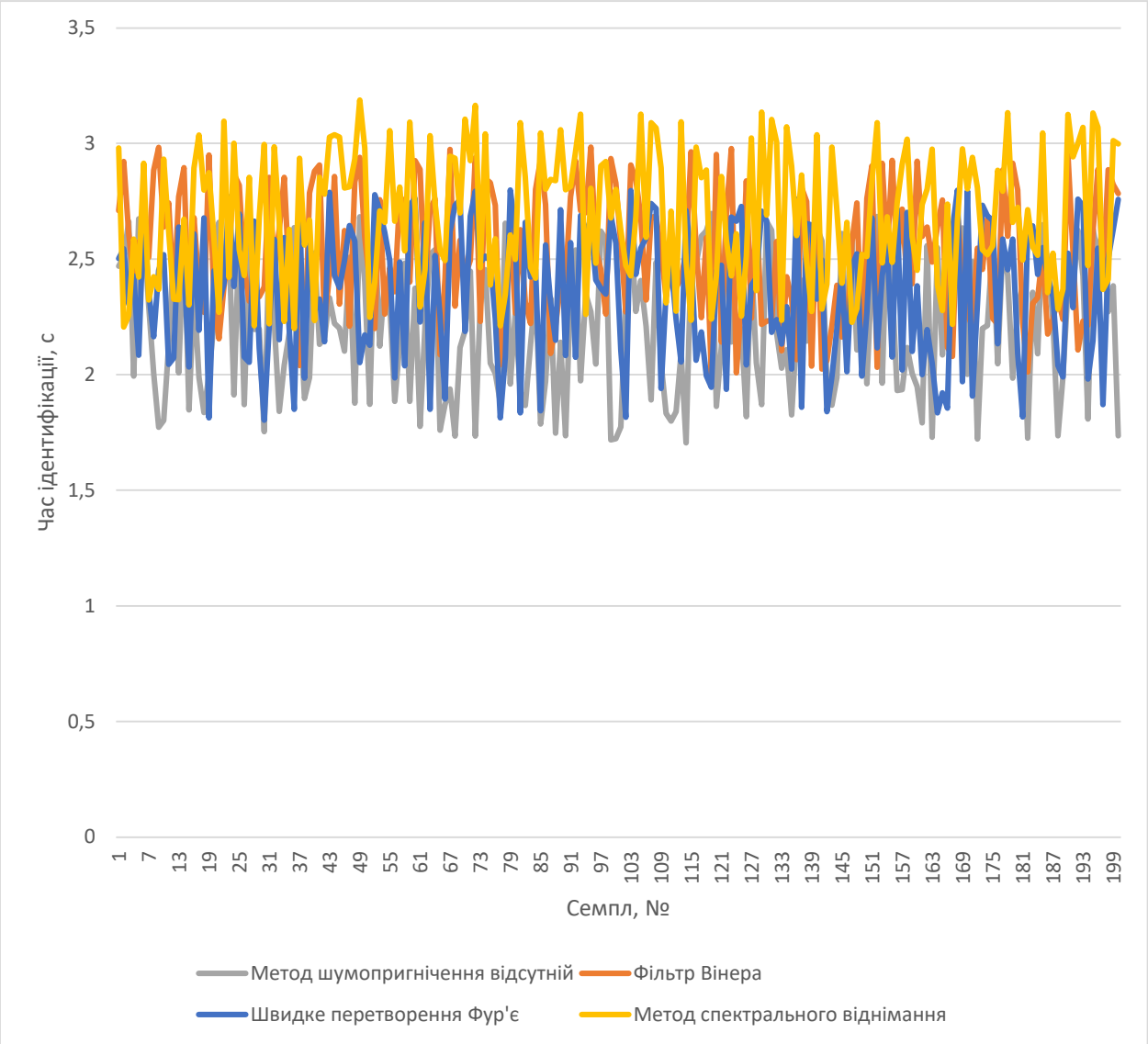


Рисунок 4.2 – Порівняння часу роботи системи з різними методами шумопригнічення

З графіка та набору метрик часу було отримано середній час кожного з методів:

- метод шумопригнічення відсутній: 2,207 секунд;
- швидке перетворення Фур’є: 2,288 секунд;

- фільтр Вінера: 2,437 секунд;
- метод спектрального віднімання: 2,663 секунд.

З середніх значень часу ідентифікації аудіоподій можна зробити висновок, що швидке перетворення Фур’є є найбільш швидким.

Обираючи метод шумопригнічення для даної системи слід ураховувати як час обробки алгоритму, так і якість ідентифікації. Наприклад, якщо система на певному апаратному забезпеченні достатньо швидко обробляє аудіоряд і затримка зневажливо мала, а користувачу важлива саме якість розпізнавання, слід обрати фільтр Вінера як найкращий за результатами тестування якості. Якщо ж, наприклад рівень шуму невеликий, а апаратне забезпечення не є досить продуктивним (наприклад, відсутність відеопроцесора), слід обрати більш простий та найшвидший метод – швидке перетворення Фур’є.

ВИСНОВКИ

У даній кваліфікаційній роботі було розглянуто основні застосування аудіоаналітики, такі як музичний аналіз, розпізнавання мови, використання аудіоредакторів тощо. Багато з них використовують шумопригнічення як основу задля покращення рівня сигналу, що було продемонстровано у роботі.

Було розглянуто основні методи та алгоритми шумопригнічення з представленням прикладів результатів обробки, їх обчислювальною складністю та можливістю паралелізації. Були розглянуті саме такі методи та алгоритми:

- швидке перетворення Фур’є;
- фільтр Вінера;
- метод спектрального віднімання;
- адаптивний фільтр;
- метод активного шумопригнічення;
- використання нейромереж.

Окрім цього, у ході роботи над даною кваліфікаційною роботою, було розроблено систему ідентифікації акустичних подій. Система була протестована у наявному вигляді. Також, відбулася інтеграція у систему 3-х модулів шумопригнічення:

- швидке перетворення Фур’є;
- фільтр Вінера;
- метод спектрального віднімання;

Після тестування системи з інтеграцією кожного з видів модулів шумопригнічення було зроблено висновок, що модуль з використанням фільтра Вінера покращує результати ідентифікації на 25%, що є найбільшим показником. Проте, тестування продуктивності показало, що швидке перетворення Фур’є є найбільш швидким методом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Crowhurst N. H. Frequency and Pitch [Електронний ресурс] / Crowhurst // Virtual Institute of Applied Sound. – 2011. – Режим доступу до ресурсу: http://www.vias.org/crowhurstba/crowhurst_basic_audio_vol1_006.html.

2. Sigismondi G. Audio Signal Processors [Електронний ресурс] / Gino Sigismondi // Shure. – 2003. – Режим доступу до ресурсу: https://engineering.purdue.edu/ece40020/References/audio_signal_processors.pdf.

3. Pascua D. The Fascinating History of Noise-Cancelling Headphones [Електронний ресурс] / Dionne Pascua // HEADOHONESTY. – 2021. – Режим доступу до ресурсу: https://www.headphonesty.com/2020/10/history-of-noise-cancelling-headphones/.

4. Active Noise Control [Електронний ресурс] // Institute of Communication Systems. – 2016. – Режим доступу до ресурсу: https://www.iks.rwth-aachen.de/en/research/audio/active-noise-control/.

5. Antonenko M. Analysis of audio analytics algorithms [Електронний ресурс] / Mikhail Antonenko – Режим доступу до ресурсу: https://sudonull.com/post/101170-Analysis-of-audio-analytics-algorithms.

6. Precision Policing Platform [Електронний ресурс] // ShotSpotter – Режим доступу до ресурсу: https://www.shotspotter.com/platform/.

7. TrueVoice™: Call center AI voice analytics [Електронний ресурс] // Deolitte – Режим доступу до ресурсу: https://www2.deloitte.com/us/en/pages/consulting/solutions/truevoice-call-center-voice-analytics.html.

8. Обзор методов улучшения речи и шумоподавления: от классики к SotA [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://habr.com/ru/company/ru_mts/blog/584308/.

9. Abdulgafar T. 6 Best Audio Noise Cancelling Software to Remove Noise [Електронний ресурс] / Tobi Abdulgafar // Krisp Blog. – 2019. – Режим

доступу до ресурсу: https://krisp.ai/blog/krisp-detailed-guide/.

10. Understanding the FFT Algorithm [Електронний ресурс] // Pythonic Perambulations. – 2013. – Режим доступу до ресурсу: https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/.

11. Wiener Filtering [Електронний ресурс] // rice.edu – Режим доступу до ресурсу: https://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/wiener.html.

12. New Insights Into the Noise Reduction Wiener Filter [Електронний ресурс] / J.Chen, J. Benesty, Y. Huang, D. Simon // IEEE. – 2006. – Режим доступу до ресурсу: https://uol.de/f/6/dept/mediphysik/ag/sigproc/download/papers/doclo/journal_wiener.pdf.

13. Dixit S. LMS Adaptive Filters for Noise Cancellation: A Review [Електронний ресурс] / Shubhra Dixit // International Journal of Electrical and Computer Engineering (IJECE). – 2017. – Режим доступу до ресурсу: https://www.researchgate.net/publication/320248881_LMS_Adaptive_Filters_for_Noise_Cancellation_A_Review.

14. Джиган В. Параллельные регуляризированные RLS-алгоритмы многоканальной адаптивной фильтрации [Електронний ресурс] / В. Джиган // dspa.ru. – 2004. – Режим доступу до ресурсу: http://www.dspa.ru/articles/year2004/jour04_2/art04_2_2.pdf.

15. Baghdasaryan D. Real-Time Noise Suppression Using Deep Learning [Електронний ресурс] / Davit Baghdasaryan // DEVELOPER BLOG. – 2018. – Режим доступу до ресурсу: https://developer.nvidia.com/blog/nvidia-real-time-noise-suppression-deep-learning/.

16. NVIDIA AMPERE GA102 GPU ARCHITECTURE. THE ULTIMATE PLAY. v 1.0 [Електронний ресурс] // NVIDIA Corporation. – 2020.

17. NVIDIA A100 Tensor Core GPU Architecture UNPRECEDENTED ACCELERATION AT EVERY SCALE. v 1.0 [Електронний ресурс] // NVIDIA Corporation. – 2020.

18. Sainburg T. Noise reduction using spectral gating in python [Електронний ресурс] / Tim Sainburg // timsainburg.com. – 2018. – Режим доступу до ресурсу: https://timsainburg.com/noise-reduction-python.html.

19. Parallel Fast Fourier Transform [Електронний ресурс] – Режим доступу до ресурсу: https://cs.wmich.edu/gupta/teaching/cs5260/5260Sp15web/studentProjects/tiba&hussein/03278999.pdf.

20. Ткаченко М. Система шумоподавления для аудио, основанная на методе спектрального вычитания [Електронний ресурс] / М. Ткаченко. – 2008. – Режим доступу до ресурсу: http://tka4.org/tka4/soft/Simple%20Noise%20Reduction/Simple%20Noise%20Reduction.pdf.

21. Істішев В. КАК РАБОТАЕТ ШУМОПОДАВЛЕНИЕ? РАЗБОР [Електронний ресурс] / Валерій Істішев // Droider. – 2021. – Режим доступу до ресурсу: https://droider.ru/post/kak-rabotaet-shumopodavlenie-razbor-24-05-2021/.

22. Gabby’s Lab: Play and learn [Електронний ресурс] // Gabby’s Lab – Режим доступу до ресурсу: https://babblelabs.com/clear-cloud/gallery/.

23. Nautiyal D. Top 5 best Programming Languages for Artificial Intelligence field [Електронний ресурс] / Dewang Nautiyal // GeeksForGeeks. – 2018. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/top-5-best-programming-languages-for-artificial-intelligence-field/.

24. Multiprocessing in Python [Електронний ресурс] // GeeksForGeeks. – 2018. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/multiprocessing-python-set-1/.

25. How to use parallelism in Python [Електронний ресурс] // NERSC – Режим доступу до ресурсу: https://docs.nersc.gov/development/languages/python/parallel-python/.

26. AudioSet [Електронний ресурс] // Google Research. – 2021. – Режим доступу до ресурсу:

https://research.google.com/audioset/download.html#split.

27. GPU support [Електронний ресурс] // TensorFlow. – 2021. – Режим доступу до ресурсу: https://www.tensorflow.org/install/gpu#software_requirements.

28. CNN Architectures for Large-Scale Audio Classification [Електронний ресурс] / [S. Hershey, S. Chaudhuri, D. Ellis та ін.] // International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. – 2017. – Режим доступу до ресурсу: https://research.google/pubs/pub45611/.

29. Оценка классификатора (точность, полнота, F-мера) [Електронний ресурс] // Bazhenov. – 2012. – Режим доступу до ресурсу: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html.