Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра оптимального керування та економічної кібернетики

Дипломна робота

бакалавра

на тему: **«Порівняльний аналіз методів розв'язування проблеми розділення сигналів за допомогою CNN»**

«Comparative analysis of methods for solving the problem of signal separation using CNN»

Виконала: студентка денної форми навчання
спеціальності 113 Прикладна математика
Ларікова Валерія Володимирівна

Керівник: к. т. н., доц. Мороз В. В.
Рецензент: к. т. н., доц. Мазурок І. Є.

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від «_____» _____ р.
Завідувач кафедри
_____ _____

Захищено на засіданні ЕК № _____
Протокол № ____ від «____» ____ р.
Оцінка _____ / _____ / _____
Голова ЕК
_____ _____

Одеса — 2021 р.

ЗМІСТ

ВСТУП

Актуальність теми. В даний час — століття інтернету, коли величезна кількість інформації доступна всім користувачам глобальної мережі, на перший план виходить проблема фільтрації даних — виділення цікавої для користувача інформації серед усього її обсягу. Ця проблема особливо актуальна в сфері розваг (фільми, музика, книги, ігри).

Аналізуючи існуючі підходи для розв'язання проблеми розділення сигналів, можна зробити висновок, що в якості системи поділу слід використовувати нейронну мережу. Це зроблено через те, що, як широко підтверджено на практиці, нейронні мережі часто добре працюють там, де немає строгого алгоритмічного рішення. Покладаючи на нейронну мережу завдання поділу, нам необхідно вирішити, що буде входом даної мережі. Звук в комп'ютері представляється у вигляді кінцевого набору амплітуд, знятих через рівні проміжки часу. Така форма подання звуку погано підходить для аналізу через її не універсальність.

Якщо взяти два фрагмента такого подання звуку, які різняться на половину секунди, то визначити, що вони практично однакові буде дуже складно. Необхідним є інша, більш зручна модель подання, якою є мел-частотна спектрограма. Вона використовується в задачах розпізнавання мови і полягає в наступному: сигнал розбивається на послідовні фрагменти рівної тривалості, для кожного з них будується спектр, а потім цей спектр перекладається в мел-шкалу для обліку сприйняття звуку людським вухом.

З огляду на ресурсомісткість навчання нейронних мереж, основним недоліком одержуваної системи є тривалість навчання мережі поділу. Прискорення навчання можна досягти зменшенням глибини мережі та кількості нейронів на кожному з шарів. Але при зменшенні мережі без зміни кількості вхідних параметрів знижується і якість її роботи. Для вирішення данного завдання, можна розбити мережу поділу на дві: перша — попередня обробка даних, друга — безпосередньо поділ. Це дозволить зменшити кількість параметрів мережі розподілу, тим самим дасть можливість зменшити її обсяг.

Як відомо, переважна більшість музичних творів є багатоголосими,

тобто складаються зі звуків, що створюються різними джерелами. При одночасному звучанні декількох джерел відбувається накладення, в результаті якого в повітрі виникають звукові коливання, що представляють собою суперпозицію безлічі синусоїдальних хвиль різних амплітуд і частот. Це ставить слухову систему людини перед проблемою поділу джерел звуку або, іншими словами, виділення окремих сигналів з суміші. Слухова система здатна успішно формувати різні частини звуку таким чином, щоб людина чула реальні звуки, вироблювані джерелами, а не просто шум.

Поділ джерел звуку, так легко вироблене мозком людини, є досить складним завданням для комп'ютера. Формально ця проблема носить назву „сліпий поділ джерел звуку" (blind audio source separation) [20, с. 43], а в сфері звукорежисури вона відома як „демікшировання" (demixing, unmixing). Термін „сліпе" в даному випадку означає, що поділ відбувається без допомоги інформації (або з дуже невеликою кількістю інформації) про специфічні властивості конкретних сигналів і процесів їх змішування.

В даний час актуальною є задача ізоляції звуків від окремих інструментів (у тому числі вокалу), які звучать в музичній композиції. Дане завдання є найбільш складним додатком сліпого поділу джерел звуку [20]. Серед областей застосування відповідної технології можна виділити наступні:

- Ремастеринг аудіозаписів, у тому числі перетворення монофонічних записів в стереофонічні (upmixing) [10];
- Постпродакшн аудіозаписів;
- Отримання мінусових фонограм (караоке) для розважальних і образотворчих цілей;
- Дубляж і реставрація старих кінофільмів;
- Створення електронної музики;
- Автоматична транскрипція музики;
- Розпізнавання текстів пісень;
- Автоматична генерація субтитрів;
- Візуалізація музики.

Варто відзначити, що методи, використовувані для поділу джерел в музичному аудіосигналі, також можуть бути застосовані для поліпшення роботи слухових апаратів [40].

Сліпий поділ джерел звуку є відносно новою технологією цифрової обробки сигналів, перші дослідження датуються початком 90-х років [20]. Однак, на сьогоднішній день вдалося домогтися значних досягнень в цій області. Так, якщо говорити про завдання, орієнтовані на якість звуку, то проблема отримання мінусових фонограм в даний час вважається вирішеною. При цьому проблема виділення сигналів від окремих музичних інструментів або вокалу представляє можливості для поліпшення [44].

Серед методів сліпого поділу джерел звуку в останні роки найкращі результати показують підходи, засновані на штучних нейронних мережах (CNN) [44].

Таким чином, в даній роботі буде побудована система розділення музики на основі методу фільтрації вмісту з використанням нейронної мережі.

Об'єктом дослідження цієї роботи є задача сліпого розділення аудіосигналу.

Предметом дослідження — ефективність алгоритмів розділення джерел аудіосигналу на основі спектральних методів та нейронних мереж.

Метою роботи є розробка алгоритму сліпого поділу джерел звуку в монофонічному музичному аудіосигналі з використанням штучних нейронних мереж.

Для досягнення поставленої мети визначені наступні завдання:

- Огляд методів поділу джерел звуку, зокрема в музичному аудіосигналі;
- Огляд архітектур CNN, які застосовуються в задачах розподілу джерел звуку;
- Розробка архітектури CNN, опис та аналіз алгоритму навчання;
- Підготовка навчального, валідаційного і тестового наборів;
- Оцінка ефективності за об'єктивними критеріями;
- Методика перетворення музичної композиції з музичного файлу в мел-спектрограму;
- Реалізація згорткової нейронної мережі розділення сигналів;
- Тестування якості розділення сигналів.

Структура роботи складається з: вступу, п'яти розділів, які об'єднують

підрозділи, висновків, списку використаних джерел і додатку.

У 2020 році дана робота пройшла апробацію на двадцятій міжнародній науково-технічній конференції, яка проводилася на базі національного технічного університету «Харківський політехнічний інститут», та має назву «Проблеми інформатики та моделювання». Тези були опубліковані у збірнику [34]. Цього року дана робота пройшла апробацію результатів дослідження на 77-й звітній студентській науковій конференції Одеського національного університету імені І. І. Мечникова. Тези були опубліковані у збірнику [35].

РОЗДІЛ 1

ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Сліпе розділення джерел

Сліпий поділ джерел звуку (blind audio source separation) є окремим видом сліпого поділу джерел (blind source separation), яке в свою чергу є технологією цифрової обробки сигналів і полягає у відновленні сигналу від одного або декількох джерел з наявної суміші.

Методи сліпого поділу джерел застосовуються в найрізноманітніших сферах: обробка зображень [7], сейсмічний моніторинг, обробка біомедичних сигналів (електроенцефалографія, магнітоенцефалографія: електрокардіографія) [5]. Сліпий поділ джерел було вперше використано в сфері комунікацій в задачах шумопониження і поліпшення мови.

Класичним завданням, що ілюструє проблематику сліпого поділу джерел, є проблема коктейльної вечірки (the cocktail party problem), вперше описана Коліном Черрі в 1953 році [8]. Вона полягає в розпізнаванні мови однієї людини на тлі мови інших людей, які говорять в той же час.

1.2. Види поділу джерел звуку

В даний час виділяють різні види завдань поділу джерел звуку. В даному розділі наводиться опис існуючих видів поділу, при цьому в процесі викладу проводиться конкретизація завдання, які вирішуються в рамках цієї роботи.

Залежно від повноти вихідної інформації виділяють сліпий і інформований поділ джерел звуку. Поділ вважається інформованим, коли доступна інформація про специфічні властивості поділюваних у деяких джерелах на ряду з терміном „сліпий поділ джерел" використовується термін „сліпий поділ сигналів" (blind signal separation).

Термін „сліпий" означає відсутність подібної інформації. Варто зазна-

чити, що не маючи інформації про конкретні джерела, ми можемо використовувати знання і гіпотези про природу поділюваних сигналів. Відповідно до сказаного у вступі, робота присвячена сліпому поділу джерел звуку.

За доступності змішаного сигналу поділ джерел звуку може бути пакетним (batch source separation), коли змішаний сигнал доступний цілком на момент поділу, і в реальному часі (online source separation), коли поділ має виконуватися в міру надходження сигналу [20]. В рамках цієї роботи проводиться розробка пакетного методу поділу.

За характером додатки завдання поділу джерел звуку можуть: бути орієнтованими на якість звуку (audio quality-oriented) і націленими на вилучення інформації (significance oriented). Дана типологія вперше була запропонована Е. Вінсентом і співавторами [30]. Робота присвячена розробці методу поділу, орієнтованого на якість звуку.

Р.М. Піньон в роботі [20] запропонував класифікацію методів поділу джерел звуку в залежності від часу затримки. Дана класифікація очевидним чином перегукується з описаними вище. Так, Піньон виділяє методи з низькою затримкою (low-latency), які застосовуються для поділу в режимі реального часу і для вбудованих додатків, і методи з високою затримкою (high-latency): орієнтовані на якість отриманого сигналу.

1.3. Поділ джерел звуку з точки зору фільтрації

Процес поділу джерел звуку по суті є фільтрацією змішаного сигналу. На рис. 1.1 наведена типізація методів фільтрації, що застосовуються в поділі джерел звуку.

Частотні фільтри представляють традиційний підхід до фільтрації сигналу. Використання інваріантних за часом фільтрів є найбільш простим способом фільтрації, але не підходить для поділу джерел звуку в музичному аудіосигналі, так як при виділенні сигналу, відповідного певному джерелу, „шум", створюваний іншими джерелами, змінюється в часі. Адаптивні фільтри враховують динамічний характер фонового „шуму", але вимагають на

вході крім змішаного сигналу сигнал, що створює перешкоди („шум”). Таким чином, адаптивні фільтри також не можуть бути застосовані у вирішенні поставленого завдання.

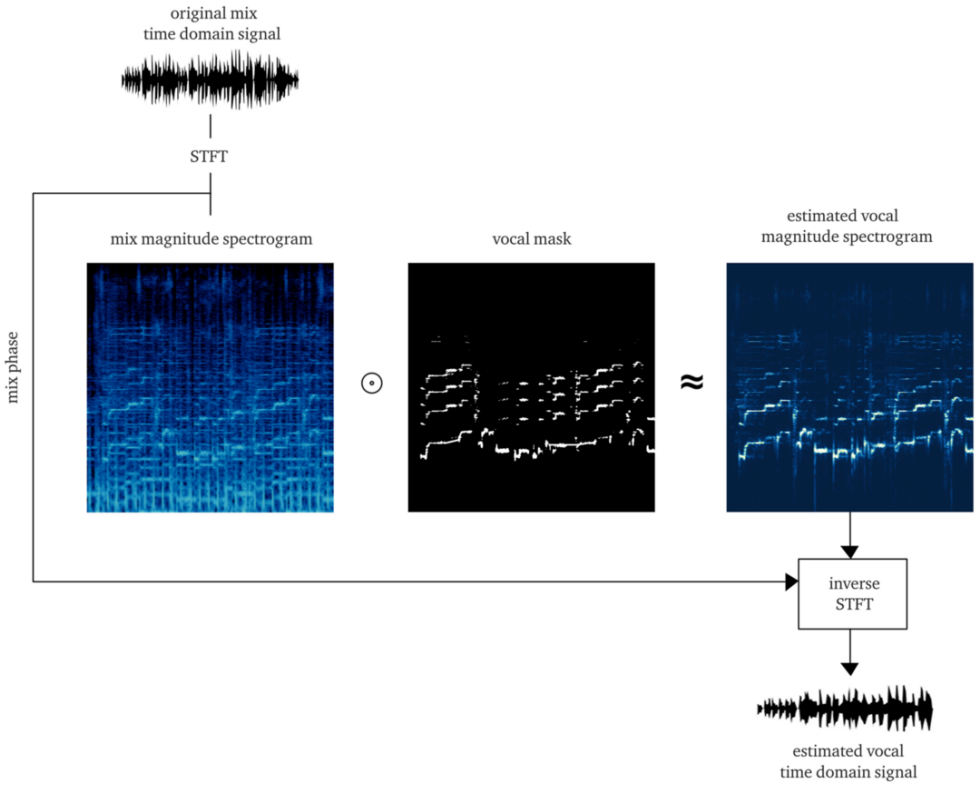

Рис. 1.1. Методи фільтрації, що застосовуються при сліпому поділі джерел звуку

Використання частотно-тимчасових масок є найбільш популярним сьогодні методом фільтрації аудіосигналу в задачах сліпого поділу джерел звуку. Суть даного методу відображена на рис. 1.2. Шляхом віконного перетворення Фур’є (short-time Fourier transform) з цифрового аудіосігналу отримують спектрограму. Потім певним чином, залежно від конкретного рішення, створюється частотно-тимчасова маска, застосування якої до спектрограми змішаного сигналу дозволяє отримати спектрограму [7] сигналу ізольованого джерела. Далі шляхом зворотного перетворення Фур’є зі спектрограми окремого джерела отримують відповідний аудіосигнал.

Спектрограма — зображення, що показує залежність спектральної щільності потужності сигналу від часу. В рамках цієї роботи під спектрограмою розуміється двовимірна діаграма, де вісь абсцис представляє час, вісь ординат — частоту, третій вимір, що представляє амплітуду на певній частоті в конкретний момент часу, представляється інтенсивністю або кольором

кожної точки зображення.

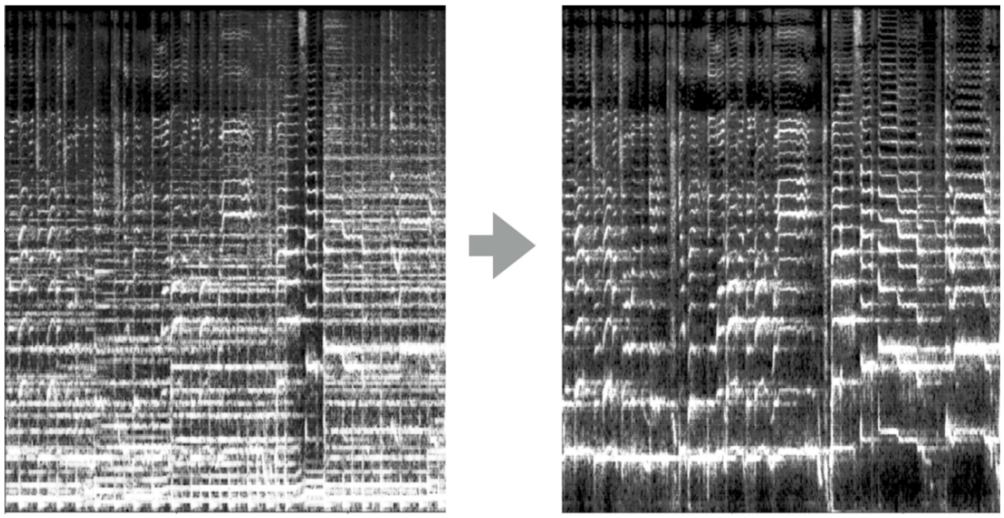

Рис. 1.2. Застосування бінарної маски для виділення вокалу з музичного аудіосигналу

У літературі описано два види частотно тимчасових масок: бінарні (жорсткі) і м'які. При застосуванні бінарних масок кожна точка (час; частота) спектрограми змішаного сигналу приписується строго одному з джерел. Важливо зауважити, що джерела звуку (інструменти) в музичному аудіосигналі мають пересічні частотні діапазони і часто звучать одночасно. Таким чином, іноді виникають ситуації накладення, коли амплітуда на певній частоті в конкретний момент часу складається з декількох джерел. Дана обставина може позначатися на якості результатів поділу. Незважаючи на це, в силу низької вартості обчислень, бінарні маски знаходять широке застосування, особливо в системах реального часу [20].

Якщо бінарна маска по суті являє собою матрицю [7], розмірність якої дорівнює розмірності спектрограми, то м'яка маска являє собою матрицю, елементи якої лежать в діапазоні $[0; 1]$. При використанні м'яких масок амплітуди, що відповідають кожній з точок спектрограми змішаного сигналу, розподіляються між джерелами пропорційно їхньому внеску. Дозволяючи домогтися кращої якості в порівнянні з бінарними, м'які маски збільшують значення обчислень.

При роботі з багатоканальним аудіосигналом можливі два способи генерації частотно тимчасових масок. У найпростішому випадку кожного

з каналів незалежно від інших може створюватися своя маска. Іншими словами, кожен з каналів обробляється як незалежний монофонічний сигнал. Інший спосіб полягає у використанні інформації про панорамування джерел. Варто зазначити, що використання інформації про панорамування не завжди представляється можливим, так як, по-перше, деякі музичні інструменти звичайно не панорамуються у стереопросторі (вокал, бас, ударні), а по-друге, велика кількість музичних записів існує тільки в форматі моно.

Рішення, що базуються на розглянутих методах фільтрації, мають природну верхню межу якості поділу. У разі, коли, крім змішаного сигналу заздалегідь відомі сигнали, які розділяються джерелом на можливі метрики, що застосовуються для оцінки якості розбиття, описуються нижче.

Панорама — це розподіл джерел звуку в стереопросторі. У найпростішому випадку в процесі панорамування відбувається пошук ідеального з точки зору звукорежисера балансу гучності в каналах для кожного з джерел. Також під час панорамування можуть застосовуватися ефекти затримки і динамічна обробка звуку.

Оперуючи оракулами, можна, по-перше, об'єктивно оцінити ефективність того чи іншого алгоритму поділу, а, по-друге, шляхом порівняння самих оракулів підібрати стратегію фільтрації найбільш оптимальну для вирішення поставленого завдання.

Результати визначення оракулів на різних наборах даних описані Е. Вінсентом і ін. [28], а також в звіті Міжнародної кампанії за оцінкою якості поділу джерел (Signal Separation Evaluation Campaign (SiSEC)) [44]. Згідно з представленими даними найкращі результати серед частотно-часових масок показують м'які маски, які не враховують інформацію про панорамування джерел.

Таким чином, теоретично кращих результатів можна досягти при використанні м'яких масок. Однак, на практиці рішення, засновані на нейронних мережах, що використовують бінарні маски, іноді показують кращі результати в порівнянні з аналогічними рішеннями, що використовують м'які маски [44].

1.4. Ступінь визначеності завдань поділу джерел

Співвідношення між кількістю джерел і числом сумішей (сенсорів, каналів аудіосигналу) є важливим і визначальним фактором при виборі методу поділу джерел. Можливі три принципові ситуації.

Коли кількість джерел збігається з кількістю вихідних сумішей, завдання поділу є певним (determined). У разі, коли число джерел перевищує число сумішей, завдання є невизначеним (undetermined). І відповідно, якщо число вихідних сумішей більше числа поділюваних джерел, завдання — перевизначене (overdetermined).

Якщо говорити про музичні аудіосигнали, то число джерел звуку не завжди є відомим і сильно варіюється, на відміну, наприклад, від завдань поліпшення мови, де кількість джерел дорівнює двум: голос і шум. При цьому в рамках цієї роботи проводиться розробка методу поділу для монофонічного аудіосигналу. Таким чином, можна зробити висновок, що розглянута задача є невизначеною.

1.5. Огляд методів поділу джерел звуку

На сьогоднішній день відомо безліч підходів до поділу джерел звуку. Їх можна розділити на три групи: моделеорієнтовані, засновані на декомпозиції сигналів і підходи на базі штучних нейронних мереж, (див. рис. 1.3)

Рис. 1.3. Методи поділу джерел звуку

1.5.1. Моделеорієнтовані методи

Моделеорієнтовані підходи до поділу аудіосигналу представлені в свою чергу двома групами методів. До методів першої групи відносяться декореляційні техніки, а також вважаються традиційними для сліпого поділу джерел аналіз незалежних компонент (independent component analysis) і метод головних компонент (principal component analysis). В основі перерахованих методів лежить використання знань і гіпотез про статистику поділюваних джерел.

Друга група моделеорієнтованних методів заснована на так званій технології формування променя (beamforming). В їх основі лежить використання інформації про просторове положення джерел звуку і сенсорів (мікрофонів). Наочною ілюстрацією цієї групи методів є мікрофонна решітка (мікрофонний масив) [41].

Важливо відзначити, що якість результатів поділу розглянутих методів сильно залежить від прийнятих моделей. Будь-яка невідповідність між розділяючим сигналом і прийнятою моделлю стає джерелом артефактів і шуму. При цьому моделеорієнтовані методи поділу джерел звуку застосовні тільки для вирішення певних та перевизначених завдань. Таким чином, описані моделеорієнтовані підходи недоречні для реалізації в рамках цієї роботи.

1.5.2. Декомпозиція сигналів

Методи, що ґрунтуються на декомпозиції сигналу, є більш сучасними підходами до поділу джерел звуку. Їх суть полягає в знаходженні компонентів, які формують вихідний змішаний сигнал, і обчисленні за допомогою їх угруповання сигналів окремих джерел.

До останнього часу найбільш популярні методи поділу джерел звуку базувалися на матричному розкладанні (НМР) (non-negative matrix factorization). В основі даного підходу лежить припущення про те, що спектр змішаного сигналу може бути змодельований як лінійна комбінація елементарних невід'ємних спектрів (базисних компонентів). Відомо, що рішення

на базі НМР показують задовільні результати. [43, 11]

У 2013 році Р.М. Піньон [20] запропонував в якості доповнення до НМР використовувати метод регуляризації Тихонова (Tikhonov Regularization). Використання даного підходу дозволяє знизити затримки і значення обчислень в порівнянні з НМР, відповідно відмінно підходить для поділу джерел звуку в реальному часі.

1.5.3. Глибоке навчання

Згідно звіту Міжнародної кампанії за оцінкою якості поділу джерел (Signal Separation Evaluation Campaign (SiSEC)) [44] за останні два роки відбувся різкий сплеск інтересу до проблеми поділу джерел звуку, при цьому більшість рішень, розроблених учасниками спільноти засновані на глибокому навчанні. Порівняння близько 30 рішень, надісланих у 2018 році для участі в кампанії SiSEC, показало, що методи, які базуються на нейронних мережах, показують найкращі результати. Важливо зауважити, що деякі з рішень на основі нейронних мереж в 50% випадків показують результати зіставні з оракулами. Таким чином, глибокі нейронні мережі є очевидним вибором для вирішення завдання, поставленого в цій роботі.

РОЗДІЛ 2

ОГЛЯД АРХІТЕКТУРИ CNN, ЯКА ЗАСТОСОВУЄТЬСЯ В ЗАДАЧАХ РОЗПОДІЛУ ДЖЕРЕЛ ЗВУКУ

Основні види архітектур нейронних мереж, включаючи згорткові мережі, автокодувальники і рекурентні мережі, відомі з кінця вісімдесятих років минулого століття. Однак справжню популярність CNN знайшли лише в результаті революції в машинному навчанні середини 2000-х, основними причинами якої послужили роботи Джеффрі Хінтона [15], Іошуа Бенджі [6], а також прогрес в обчислювальній техніці і поява наборів даних, що мають достатні розміри.

Перший промисловий успіх нейронних мереж був пов'язаний із застосуванням їх в сфері розпізнавання мови. Сьогодні індустріальні додатки на основі глибокого навчання знаходять застосування в найрізноманітніших сферах: комп'ютерному зорі, розпізнаванні мови і аудіозаписів, обробці природних мов, робототехніці, біоінформатиці і хімії, відеоіграх, пошукових системах, інтернет-рекламі і фінансах. Досягнення систем на базі CNN пробудили цікавість до їх застосування і в області розділення джерел звуку.

У цьому розділі наводиться огляд архітектур нейронних мереж, що застосовуються в задачах розподілу джерел звуку. Загальний аналіз основних архітектур CNN відомих на сьогоднішній день широко висвітлений в книгах І. Гудфеллоу, І. Бенджі, А. Курвілля [2], а також С. Ніколенко, А. Кадуріна, Е. Архангельскої [4] і виходить за рамки цієї роботи.

2.1. Повнозв'язні мережі

Повнозв'язні нейронні мережі (dense neural networks, DNN) не застосовуються для вирішення завдань поділу джерел звуку в силу неефективності в порівнянні з іншими типами архітектур. Справа в тому, що ці мережі не враховують топологію вхідних даних. Так, наприклад, при роботі з зобра-

женнями (спектрограми аудіосигналу), сусідні пікселі подаються на вхід повно-мережі як незалежні компоненти. Це означає, що мережа в процесі навчання повинна спершу зрозуміти, що деякі компоненти вхідного вектора сильно скорельовані [4].

В роботі [21] наведено порівняння повних і згорткових мереж стосовно задачі поділу джерел звуку. Автори відзначають, що згорткова мережа з меншою кількістю параметрів ніж у повнозв'язного аналога показує кращі результати.

2.2. Згорткові мережі

Згорткові нейронні мережі (convolutional neural networks) є однією з найбільш популярних архітектур ШНМ і призначені для обробки даних з гратковою топологією. Основним додатком, заради якого були розроблені згорткові мережі, є обробка зображень. Шляхом перетворення Фур'є аудіосигнал може бути перетворений в двовимірний тензор, рядки якого відповідають частотам, а стовпці - моментам часу. Отриманий тензор (спектрограма) по суті є поданням аудіосигналу у вигляді зображення і може бути поданий на вхід згорткової мережі.

На сьогоднішній день відомо кілька архітектур згорткових мереж, що застосовуються для вирішення завдання поділу джерел звуку в музичному аудіосигналі.

2.2.1. Класичні згорткові мережі

Під класичними згортковими мережами в рамках цієї роботи будемо розуміти мережі з декількох пересічних згорткових шарів, що складаються в загальному випадку з блоків згортки, нелінійної функції активації і пулінгу (pooling); декількох повнозв'язних шарів на виході мережі. Прикладом класичної архітектури згорткової мережі є відома архітектура VGG [24].

Як приклад CNN з класичною архітектурою, застосовуваних для поділу джерел звуку, можна навести архітектуру Ж. Рома, О. Гріна і П.А. Трамбле [21], а також архітектуру Е. Корецького [36].

CNN Рома, Гріна і Трамбле містить три послідовних згорткових шари, що складаються з блоків згортки з ядром 5×5, активації ReLU і пулінгу з ядром 2×2, і завершується двома повнозв'язними шарами. Число карт ознак (каналів) поступово зростає на більш глибоких рівнях мережі від 4 до 32. На вхід мережі подається відрізок спектрограми, відповідний 11 крокам перетворення Фур'є ($\sim$ 130 мс). На виході - вектор, що представляє собою одиничний інтервал маски. В якості опції помилки використовується сума квадратів відхилень (mean squared error (MSE)), в якості оптимізатора — адаптивний варіант градієнтного спуску ADAM. У ряді джерел використовується термін „субдискретизація".

До недоліків описаної архітектури можна віднести використання згорткових блоків з ядром 5×5. Відомо [4], що два підряд згорткових блоки з ядром 3×3 мають рецептивне поле розміром 5×5, кількість ваг при цьому у них буде 18 проти 25. Таким чином, мережа може стати глибшою, зменшуючи при цьому загальну кількість ваг. Наявність додаткової нелінійності між шарами дозволяє збільшити „роздільну здатність" в порівнянні з єдиним шаром з великою згорткою.

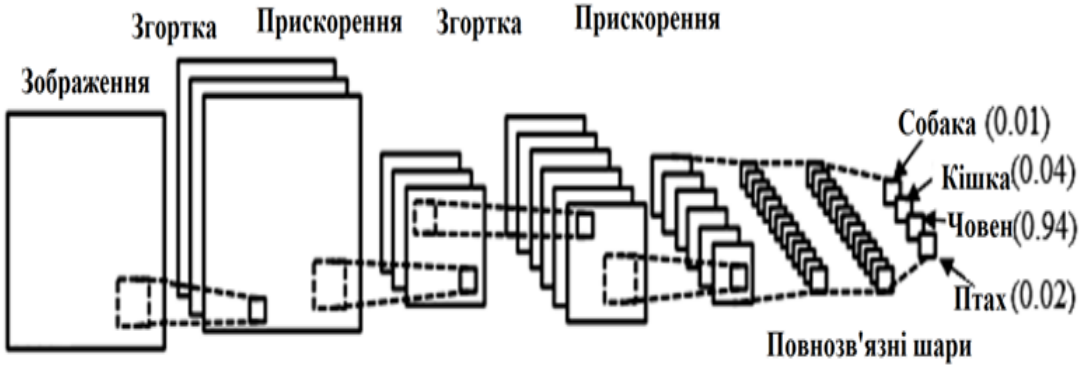

Рис. 2.1. Схема архітектури згорткової мережі Ж. Рома, О. Гріна і П.А. Трамбле [21]

ШНМ Е. Корецького містить чотири згорткових шари, які складаються з блоків згортки з ядром 3×3 і активації LeakyReLU. Через кожні два шари передбачений блок пулінгу з ядром 3×3. Завершується мережа двома повнозв'язними шарами. Число карт ознак (каналів) коливається від шару до шару. На вхід мережі подається відрізок спектрограми, відповідний 25 крокам перетворення Фур'є ($\sim$ 290 мс). На виході — вектор, що представляє

собою одиничний відрізок бінарної маски. В якості опції помилки використовується сума квадратів відхилень, хоча по суті отримання бінарної маски швидше є завданням класифікації (див. п. 3.2). В якості оптимізатора використовується стохастичний градієнтний спуск.

До недоліків описаної моделі можна віднести використання на вихідному шарі мережі лінійної функції активації. Очевидно, що застосування функції сигмоїда, що має область значень $(0; 1)$, може полегшити задачу мережі при навчанні на частотно-тимчасових масках, значення елементів яких знаходяться в діапазоні $[0; 1]$.

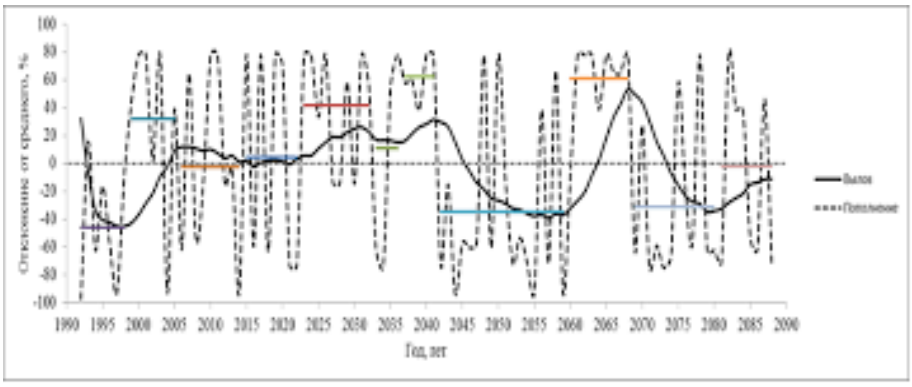

Рис. 2.2. Схема архітектури згорткової мережі Е. Корецького [36]

2.2.2. U-Net мережі

Архітектура U-Net була розроблена для сегментації біомедичних зображень [22] і по суті являє собою згорткову архітектуру типу кодувальник-декодувальник. Як кодувальник виступає класична згорткова мережа, кожен з шарів якої зменшує вхідне зображення, збільшуючи при цьому кількість каналів. Декодувальник є нейронною мережею (deconvolutional neural network) [19], кожен з шарів якої відповідно збільшує вхідне зображення, зменшуючи кількість каналів. Важливо відзначити, що архітектура U-Net передбачає з'єднання відповідних верств кодувальника і декодувальника. Початкове зображення, проходячи через мережу, стискається, минає вузьке місце на стику кодувальника і декодувальника і розтискається до початкового розміру.

Сегментація зображення — це процес його розбиття на сегменти шляхом присвоєння міток кожному пікселю. Створення маски для джерела

звуку на основі спектрограми змішаного сигналу по суті є її сегментацією. Дана ідея спонукала дослідників до адаптації U-Net мереж для вирішення завдання поділу джерел звуку. Як приклади відзначимо дві реалізації.

Архітектура А. Янссона, Е. Хамфрі, Н. Монтеккі, Р. Біттнер, А. Кумар, Т. Вейда [17] є варіацією класичної U-Net архітектури (див. рис. 2.3). Кодувальник і декодувальник мають глибину в 6 шарів. Ядро згортки різних верств має розмір 5×5 і крок рівний 2. В якості оцінки помилки прийнята сума модулів відхилень (mean absolute error). Оптимізатор — ADAM.

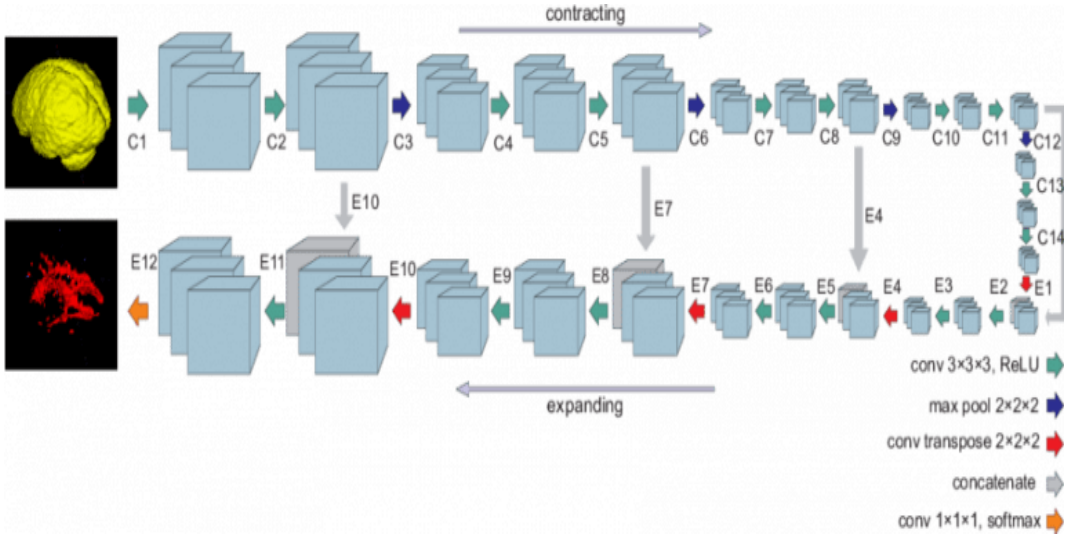

Рис. 2.3. Схема архітектури U-Net запропонована А. Янссон, Е. Хамфрі, Н. Монтеккі, Р. Біттнер, А. Кумар, Т. Вейд [17]

Архітектура Д. Столлера, С. Еверта, С. Діксона [25] представляє одновимірну адаптацію U-Net архітектури. Автори називають запропоновану ними специфічну архітектуру Wave-U-Net, підкреслюючи, що вона розроблена спеціально для роботи з аудіосигналом. Замість тимчасового відрізка спектрограми на вхід мережі подається відрізок форми хвилі змішаного аудіосигналу, на виході — відповідний відрізок форми хвилі джерела.

2.3. Рекурентні мережі

Рекурентні нейронні мережі — це сімейство нейронних мереж, призначених спеціально для обробки послідовних даних, до яких, можна віднести аудіосигнали. Архітектури рекурентних мереж діляться на три групи: звичайні рекурентні мережі, мережі з довгою короткостроковою пам'яттю (long

short term memory (LSTM)) і вентильні мережі (gated recurrent unit (GRU)).

Детальний огляд перерахованих видів архітектур наведено в літературі [4] і виходить за рамки цієї роботи. Нижче наведено огляд відомих архітектур рекурентних мереж, що застосовуються в задачах розпізнавання.

2.3.1. Мережі з довгою короткостроковою пам'яттю (LSTM)

Основною ідеєю, що лежить в основі архітектури рекурентних мереж є наявність в рекурентних шарах прихованих станів, запам'ятовуючих історію даних, що використовуються для передбачення майбутніх елементів послідовності. Як відомо, в звичайних рекурентних мережах вплив прихованого стану на наступні стани рекурентної мережі експоненціально спадає. Для подолання зазначеної проблеми були запропоновані мережі з довгою короткостроковою пам'яттю, прихований стан в яких представляє не єдине число, а спеціальний вид, в якому явно змодельовані „довга пам'ять", процеси запису і читання з цього „осередка пам'яті" [4].

Як приклад LSTM мережі, розробленої для вирішення завдання поділу джерел звуку, можна навести архітектуру запропоновану С. Уличем, М. Порку, Ф. Хироном, М. Ененклом, Т. Кемпом, Н. Такахаші та Ю. Міцуфуджі [27], що представляє собою двосторонню мережу з довгою короткостроковою пам'яттю, що складається з трьох шарів, кожен з яких містить 500 осередків.

2.3.2. Вентильні мережі (GRU)

Основним стимулом для розробки вентильних мереж було бажання знизити кількість параметрів в LSTM мережах, зберігши при цьому ефект довгострокової пам'яті.

Як приклад вентильної мережі, що використовується для поділу джерел звуку, можна вказати модель, розроблену Д. Уордом і Ц. Ц. Конгом [31] і представлену на Кампанії по оцінці якості методів поділу сигналу 2018 року [4]. Дана мережа складається з трьох двонапрямлених GRU шарів, кожен з яких містить по 521 осередку.

2.4. Мережі змішаного типу

Іноді на практиці застосовуються досить екзотичні моделі ШНМ, що поєднують в собі елементи різних видів архітектур. Так, серед моделей ШНМ, які застосовуються для вирішення поділу джерел звуку, зустрічаються архітектури, що поєднують в собі елементи згорткових U-Net і рекурентних мереж.

До архітектур даної групи відносять мережу Н. Такахасі, Н. Госвамі, Ю. Міцуфудзі [26], К. Дура [9], а також мережу, розроблену С.І. Мімілакісом, К. Дроссос, Ж. Ф. Сантосом, Д. Шуллер, Т. Віртаненом, І. Бенджі [18].

2.5. Штучні нейронні мережі

2.5.1. Мережі прямого поширення

Мережа прямого поширення (рис. 2.4) є найпростішою штучною нейронною мережею. У такій мережі сигнал поширюється тільки в одному напрямку, від вхідних вузлів до прихованих вузлів, якщо такі є, а потім на вихідні нейрони. У мережах прямого поширення немає циклічних зв'язків.

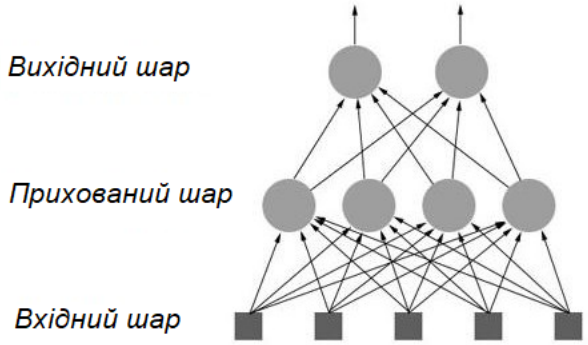

Рис. 2.4. Мережа прямого поширення

Часто мережу прямого поширення називають — перцептроном. Перцептрон може бути одношаровим, в ньому немає прихованого шару, сигнал з входу передається на вихід через систему ваг, і багатошаровим, в ньому сигнал проходить через приховані шари, а потім потрапляє на вихідний шар [4].

2.5.2. Навчання нейронної мережі

Створення нейронної мережі передбачає дві фази: перша — побудова структури мережі, друга — навчання мережі. Навчання мережі — це процес налаштування ваг зв'язків в мережі таким чином, щоб мережа за вхідними даними видавала необхідні вихідні.

Серед існуючих методів навчання мережі можна виділити 2 класи:

- Детерміновані методи, які ітеративно змінюють параметри мережі, в залежності від фактичних і бажаних виходів.
- Стохастичні методи, які випадковим чином змінюють параметри мережі, але зберігають тільки ті зміни, які привели до збільшення якості роботи мережі.

Також на основі іншого принципу класифікації можна виділити методи навчання з учителем і без вчителя.

Метод навчання з учителем використовує навчальну вибірку, в якій для кожного набору вхідних даних вказано вірний набір вихідних даних.

Завдання навчання мережі з учителем полягає в пошуку відображення, де X — простір вхідних даних; Y — простір вихідних даних на основі навчальної вибірки - кінцевого набору векторів X з X, для кожного з яких зазначено вірний вихід — Y з Y. Розмір навчальної вибірки повинен бути достатній для побудови відображення f.

Метод навчання без вчителя може застосовуватися, коли відомі тільки вхідні сигнали. На їх основі мережу навчають формувати на виході найкращі значення — визначається алгоритмом навчання, але зазвичай потрібно для близьких вхідних значень видавати близькі вихідні.

2.5.3. Метод зворотного поширення помилки

Метод зворотного поширення помилки є детермінованим методом навчання з учителем для багатошарового перцептрона.

Алгоритм зворотного поширення можна розділити на дві фази. Фаза поширення:

1) Пряме поширення навчального прикладу через мережу для формування виходу мережі.
2) Зворотне поширення, отриманого виходу з мережі для обчислення дельт-різниці між поточним і цільовим виходом для всіх нейронів мережі.

Фаза поновлення ваг:

1) Перемноження обчислених дельт і сигналів нейронів для обчислення градієнта ваг.
2) Віднімання частки значення градієнта з ваг.

Величина частки градієнта впливає на швидкість навчання і якість навчання, вона називається learning rate. Більше значення learning rate призводить до більш швидкого навчання, менші значення — до більш точного навчання.

Дві дані фази повторюються, поки мережа не досягне бажаної якості розпізнавання [5].

Алгоритм зворотного поширення помилки широко використовується, але не є універсальним, основна проблема — невизначений час навчання. У деяких задачах для навчання потрібно кілька днів або навіть тижнів, а також можливий випадок, коли мережа взагалі не навчиться. Це може статися з таких причин: випадок, коли в процесі навчання ваги зв'язків стають дуже великими і де значення похідної активаційної функції дуже малі. При цьому крок навчання стає дуже маленьким і навчання практично завмирає. Зазвичай дану проблему уникають шляхом зменшення розміру кроку навчання, але це збільшує час навчання.

Метод зворотного поширення помилки є різновидом градієнтного спуску, який в процесі навчання коригує ваги у напрямку до мінімуму. При цьому багатовимірний простір функції має безліч локальних мінімумів і мережа може потрапити в один з них, коли поруч є набагато більш глибокий мінімум. У точці локального мінімуму напрямок градієнта веде в одну точку і мережа не здатна вибратися з нього. Величина кроку впливає на швидкість навчання, і вибір конкретного значення змушує шукати компроміс між повільною швидкістю навчання і низькою точністю.

2.5.4. Згорткові нейронні мережі

Одним з класів нейронних мереж є згорткові нейронні мережі (CNN). Згорткові мережі стали проривом в області глибокого навчання. У дев'яностих CNN використовувалися для розпізнавання символів, але більш широке поширення вони отримали після роботи, в якій CNN використовувалися для класифікації зображень. На данний момент згорткові мережі є дуже зручним інструментом для тих, хто займається машинним навчанням.

Математичний аналіз CNN був ініційований Маллатом. Зокрема Маллат розглянув так звані розсіюючі мережі, засновані на частково-дискретних інваріантних до зсуву фреймах і модулях нелінійності в кожному шарі мережі, і які забезпечують незалежність від перенесення і стабільності до спотворень відповідних примітивів. Згорткові нейронні мережі охоплюють велику область, що включає згорткові ядра, різні активаційні функції, такі як, relu, logistic sigmoid, hyperbolic tangents, різні функції втрат mse, binary crossentrophy, categorical crossentrophy, softmax, шари об'єднання і субдескритезації. Глибокі згорткові нейронні мережі засновані на отриманні примітивів зазвичай розділених на основі навчених або заздалегідь визначених (апріорно вибраних) фільтрів. Фільтри використовують розмічені набори даних і є наслідком хорошої якості класифікації для великих наборів даних, на невеликих може проявитися перенавчання. Навчання фільтрів на немаркованих даних так само іноді є життєздатним. Заздалегідь певні фільтри, включаючи вейвлет фільтри, і неструкторизовані випадкові фільтри добре працюють на наборах даних довільних розмірів.

Архітектура згорткових нейронних мереж формується з послідовності шарів по-різному перетворених вхідних сигналів у вихідний. Загалом, використовується кілька певних типів шарів [6].

Основою CNN є афінне перетворення. Вектор, поданий на вхід, перемножується з матрицею, формуючи вихідний сигнал. Таке перетворення може бути застосовано для будь-яких типів даних — зображення, звукова доріжка або набір невизначених характеристик, тому що сигнал будь-якої розмірності може бути розгорнуто в вектор перед застосуванням перетворення.

Зображення, звукові записи, і інші типи даних схожі з ними мають, властиву їм, структуру:

- Дані представляються у вигляді багатовимірних масивів;
- Є осі, на яких важливий порядок (висота і ширина для зображень, час і частота для аудіосигналів);
- Одна вісь використовується для різного представлення даних, названа каналом (червоний, зелений і синій канали для кольорових зображень, лівий і правий канали для аудіосигналів).

При застосуванні афінних перетворень дана інформація не враховується, всі осі обробляються однаково, не беручи до уваги топологічну інформацію. Але знання про топологію даних може бути дуже корисним при вирішенні завдань комп'ютерного зору або розпізнавання мови. У таких випадках використовується дискретна згортка.

Дискретна згортка — це таке лінійне перетворення, яке зберігає інформацію про порядок. Одне перетворення застосовується до декількох частин вхідного блоку, формуючи вихід. Параметрами згорткового шару є набір учнів фільтрів (ядер). Кожне ядро має невелику зону чутливості, тобто з'єднується тільки з невеликою частиною попереднього шару, але кожне ядро поширюється вздовж усього вхідного шару. Кожне ядро переміщується вздовж вхідного сигналу, обчислюється скалярний добуток ядра і частини вхідного сигналу і формується карта ознак для кожного ядра, крім застосування афінного перетворення до його результату застосовується функція активації, функцій активації досить багато і вони будуть розглянуті докладніше далі в цьому розділі.

Згортковий шар описується декількома характеристиками:

- N — розмірність вхідних даних;
- n — кількість вхідних карт ознак;
- m — кількість ядер згортки;
- I_k — розмір вхідних даних уздовж кожної осі,$k = \overline{1,N}$;
- J_k — розмір ядра згортки уздовж кожної з осей, $k = \overline{1,N}$;
- S_k — stride — відстань між найближчими позиціями застосування ядра згортки уздовж кожної з осей, $k = \overline{1,N}$;
- P_k — zero padding — кількість нулів, що додаються до вхідних карт

ознак, на початку і наприкінці кожної з осей, $k = \overline{1,N}$.

Приклад двовимірної дискретної згортки для ядра розміром 3×3 (рис. 2.5); на карті вхідних ознак 5×5, с кроком 1×1 і zero padding 0×0, що наведено на рис. 2.6.

Рис. 2.5. Двовимірна дискретна згортка для ядра розміром 3×3

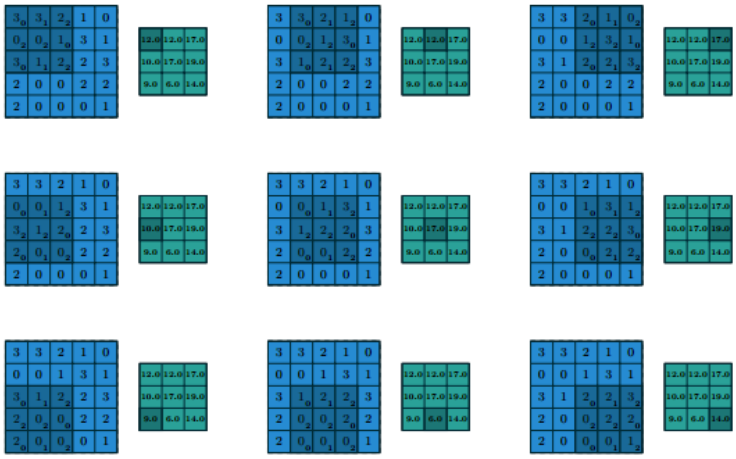

Рис. 2.6. Карта вхідних ознак 5×5, с кроком 1×1 і zero padding 0×0

Найчастіше згортка проводиться не з одним ядром, а з декількома ядрами і не на одній карті ознак, а на кількох. У таких випадках кожне ядро згортки застосовується до всіх карт і формує безліч вихідних карт ознак, які потім об'єднуються в одну шляхом підсумовування. Після дискретної згортки кількість вихідних карт ознак дорівнює кількості ядер згортки, застосованих на даному шарі.

Розмір виходу згорткового шару залежить від всіх вище перерахованих його характеристик [6].

Інтерпретувати роботу згорткового шару можна виділенням примітивів більш високого рівня, ніж на вході, не залежно від їх економічного становища у вхідному блоці. Наприклад, для зображень, згортковий шар нейронної мережі може виділяти відрізки, розташовані під різними кутами в будь-якій частині вихідного зображення. Кожен наступний згортковий шар

CNN виділяє примітиви все більш високого рівня. Продовжуючи приклад із зображеннями, наступний згортковий шар може виділяти, сформовані з відрізків на попередньому шарі, фігури: трикутники, кола та ін.

Другим типом шарів, з яких будуються CNN, є pooling layer. Даний шар використовується для зменшення розміру карти ознак, об'єднуючи його частини, використовуючи якусь функцію об'єднання, такі як середнє або максимальне значення.

Об'єднання відбувається шляхом переміщення вікна уздовж карти ознак та застосування до поточного вікна деякої функції об'єднання. Це схоже на роботу дискретної згортки, але тільки замість ядра згортки використовується інша функція.

Операція об'єднання має такі характеристики:

- I_k — розмір вхідних даних уздовж кожної осі, $k = \overline{1,N}$;
- J_k — розмір pooling-вікна уздовж кожної з осей, $k = \overline{1,N}$;
- S_k — stride — відстань між найближчими позиціями застосування вікна уздовж кожної з осей, $k = \overline{1,N}$.

У завданнях пов'язаних з обробкою зображень, зазвичай використовується вікно розміром 2×2, яке ковзає уздовж вхідної карти ознак без перекриття [6].

Крім згорткового шару і pooling шару в CNN застосовуються звичайні повнозв'язні шари, в яких кожен нейрон з'єднаний з усіма нейронами попереднього шару. Перед виконанням з'єднання відбувається розгортання N – мірної карти ознак попереднього шару в одновимірний вектор.

Цей тип шару зазвичай останній в нейронній мережі, він використовується для її навчання. Даний шар характеризує спосіб формування „штрафів" за різницю між отриманим і потрібним виходом мережі. Тут можуть бути використані різні функції втрат, вибір конкретної функції залежить від розв'язуваної задачі.

Найбільш загальною архітектурою CNN є архітектура виду, коли спочатку дані проходять через згортковий шар, після згорткового шару йде pooling шар, дана пара шарів повторюється кілька разів, а потім вихідний сигнал надходить на вхід повнозв'язного шару або послідовності

повнозв'язних шарів.

2.5.5. Функції активації

Існує безліч різних функцій активації [7], розглянемо найуживаніші функції в даному розділі.

Порогова функція (2.1), грубо апроксимуюча функція активації, яка використовується в біологічному нейроні.

$$f(x) = \begin{cases} a_1, & x > T \\ a_2, & x \leq T \end{cases} \tag{2.1}$$

Даний вид функції корисний в схемі бінарної класифікації, коли необхідно віднести сигнал, поданий на вхід до одного з двох класів, або коли необхідно створити набір який сигналізує про наявність якоїсь характеристики у вхідних даних. Кожен сигналізатор буде невеликою мережею, яка буде видавати на виході — 1, якщо особливість присутня у вхідних даних, 0 — в іншому випадку. Комбінація таких сигналізаторів може бути рішенням завдання класифікації.

Лінійна функція активації (2.2). Значення цієї функції залежить від величини вхідного сигналу.

$$f(x) = ax + b \tag{2.2}$$

Тут a і b — є параметрами функції.

Rectifield Linear Unit (ReLU)

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2.3}$$

Parametric Rectifield Linear Unit (ReLU)

$$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2.4}$$

Log-sigmoid — функція, яка більш точно наближає активаційну функцію біологічного нейрона, виражається співвідношенням (2.5).

$$f(x) = \frac{1}{1 + e^{-\beta x}} \tag{2.5}$$

Де β — параметр нахилу функції. Ця функція активації широко застосовується по причині простоти її похідної, що використовується при навчанні мережі.

Активація softmax переважно використовується в вихідному шарі системи кластеризації. Вона перетворює „сире" значення виходу до апостеріорної ймовірності. Значення функції виражається наступною формулою (2.6).

$$y_i = \frac{x_i}{\sum_{j=1}^{N} x_j} \tag{2.6}$$

Де N — кількість нейронів вихідного шару, x_i — зважена сума входів j-того нейрона, y_i — вихід i-го нейрона.

2.5.6. Функції втрат

Функції втрат використовуються для обчислення заходів схожості між обчисленим виходом мережі і правильним виходом. Інформація про помилку на виході мережі використовується при навчанні мережі [8].

Середньоквадратична помилка (MSE) виражається формулою (2.7)

$$V\left(f(x), y\right) = \frac{1}{N} \sum_{i=1}^{N} \left(f(x)_i - y_i\right)^2 \tag{2.7}$$

Тут y_i — очікуване значення i-того компоненту вихідного сигналу,

$f(x)_i$ — реальне значення, отримане на виході мережі. MSE часто використовується в задачах регресійного аналізу.

Функції втрат використовувані в класифікації. Даний тип функцій втрат обчислює ціну „витрат" за неточність класифікації. Дано векторний простір всіх можливих входів X і простір виходів $Y = \{-1; 1\}$ наше завдання знайти функцію, яка найкращим чином ставить кожному x з X вектор y з Y.

Функція квадратичних втрат. Зазвичай використовується в задачах регресії функція квадратичної помилки може бути переписана як (2.8) для задач класифікації.

$$V\left(f(x), y\right) = \left(1 - yf(x)\right)^2 \tag{2.8}$$

Квадратична функція втрат надмірно штрафує викиди, що веде до більш повільної швидкості збіжності, ніж у логістичній функції втрат або hinge loss function.

Hinge loss визначається співвідношенням (2.9).

$$V\left(f(x), y\right) = max\left(0.1 - yf(x)\right) = \left|1 - yf(x)\right| \tag{2.9}$$

Логістична функція втрат визначається виразом (2.10).

$$V\left(f(x), y\right) = \frac{1}{\ln 2} \ln\left(1 + e^{-yf(x)}\right) \tag{2.10}$$

Швидкість збіжності даної функції близька до швидкості збіжності hinge loss, але оскільки дана функція неперервна, то при навчанні може бути застосований метод градієнтного спуску. Структура логістичної функції дозволяє їй бути стійкою до викидів в даних.

Crossentropy loss. Використовуючи функцію (2.11), функція crossentropy loss визначається виразом (2.12).

$$t = \frac{1 + y}{2} \tag{2.11}$$

$$V\left(f(x), y\right) = -t \cdot \ln\left(f(x)\right) - (1 - t) \ln\left(1 - f(x)\right) \tag{2.12}$$

Ця функція широко застосовується в області машинного навчання.

Categorical crossentropy використовується для багатокласової класифікації.

$$V\left(f(x), y\right) = -\sum_{i=1}^{N} f(x)_i \log(y_i) \tag{2.13}$$

Тут $f(x)$ і y — одномірні вектори довжини N. Цю функцію можна використовувати тільки з активаційною функцією softmax.

2.5.7. Метод градієнтного спуску

Метод градієнтного спуску є оптимізаційним алгоритмом першого порядку. Під час пошуку локального мінімуму функції з використанням методу градієнтного спуску, кожен крок пропорційний градієнту функції в конкретній точці. Метод градієнтного спуску базується на спостереженні про те, що якщо функція декількох змінних, де X — вектор, визначений і диференційований в точці A, то вона зменшується максимально швидко в напрямку негативного градієнта функції f в точці A.

Використання для навчання мережі методу зворотного поширення помилки, через її недоліки, є не дуже раціональним. Були розроблені більш оптимальні методи навчання, яким не притаманні недоліки цього методу. Одним з них є метод стохастичного градієнтного спуску [9].

Даний метод слідує уздовж негативного градієнта після перегляду однієї або деякої частини прикладів з навчальної вибірки. Використання методу SGD в налаштуванні нейронних мереж мотивовано високою затратністю виконання зворотного поширення на всій навчальній вибірці. SGD дозволяє зменшити цю вартість і все ще зберігає високу швидкість збіжності.

Стандартний метод градієнтного спуску оновлює параметри функції обчислення помилки за формулою (2.14):

$$\Theta = \Theta - \alpha \nabla_{\Theta} E\left[J(\Theta)\right]. \tag{2.14}$$

Цей вираз аппроксимує градієнт і величину втрат на всій навчальній

вибірці. α — параметр швидкості навчання (learning rate). Стохастичний метод градієнтного спуску оновлює параметри відповідно до виразу (2.15), використовуючи тільки один тренувальний запис або кілька.

$$\Theta = \Theta - \alpha \nabla_{\Theta} J(\Theta;\ x^{(i)}, y^{(i)}), \tag{2.15}$$

де $x^{(i)}, y^{(i)}$ — пара з тренувального набору.

Зазвичай SGD оновлює параметри використовуючи відразу декілька тренувальних прикладів, проти того, щоб робити це для кожного запису. Так робиться з двох причин: по-перше, щоб зменшити дисперсію в оновленні параметрів, що може привести до більш стабільної збіжності, по-друге, це дозволяє використовувати оптимізовані матричні обчислення які можуть використовуватися при обчисленні величини помилки і градієнта.

У SGD learning rate — зазвичай набагато менше, ніж в методі стандартного градієнтного спуску, тому що дисперсія зміни більш висока. Вибір оптимального параметра навчання і закону його зміни в процесі навчання досить складне завдання. Найбільш стандартний метод, який добре працює на практиці, це використовувати досить маленьке константне значення learning rate на початковому етапі навчання, а потім зменшувати його в два рази в міру зменшення швидкості збіжності. Ще більш гарним способом є змінювати learning rate, коли зміна величини помилки між двома етапами нижче деякого порогу. Це хороший спосіб для швидкої збіжності до локального мінімуму. Також часто використовується для обчислення learning rate вираз (2.16):

$$lr_{i+1} = \frac{lr_0}{lr_{i+1} + 1}, \tag{2.16}$$

де lr_i — learning rate на і-му етапі навчання.

Для SGD є важливий порядок надходження даних при навчанні. Якщо дані подаються в певному порядку, це може змістити градієнт і погіршити збіжність. Зазвичай для того щоб цього уникнути дані перемішуються на кожному навчанні.

Якщо функція втрат має вигляд довгих крутих ярів з різкими крутими сторонами, то збіжність навчання може бути досить низькою через те, що рух буде відбуватися з боку в бік, замість того, щоб рухатися вздовж яру

до оптимуму. Для подолання даного ефекту використовується метод інерції (2.17):

$$\begin{gathered} v = \gamma v - \alpha \nabla_{\Theta} J(\Theta;\ x^{(i)}, y^{(i)}) \\ \Theta = \Theta - v \end{gathered} \tag{2.17}$$

Алгоритм на кожному кроці намагається рухатися проти градієнту, але при цьому, за інерцією, частково рухається в тому ж напрямку, що і на попередній ітерації. Такий метод має дві важливі властивості:

- Він практично не ускладнює звичайний градієнтний спуск в обчислювальному плані;
- При правильному виборі параметрів, такий метод швидше працює, ніж звичайний градієнтний спуск, навіть з оптимально підібранним кроком.

2.5.8. Dropout (виняток)

Глибокі нейронні мережі є потужним засобом, що дозволяє описувати складні нелінійні залежності між вхідними даними і вихідними. Однак, при обмеженому навчальному наборі даних, зв'язок може виникати через наявність певних „шумів" в навчальній вибірці, не характерних для реального набору, на якому мережа буде працювати, навіть якщо ці дані з одного розподілу. Це призводить до перенавчання мережі, для запобігання якого розроблено багато методів. Вони включають зупинку навчання, як тільки на тестовому наборі даних продуктивність мережі починає зменшуватися, включаючи різного роду штрафи на ваги, такі як L_1 і L_2 регуляризацію і м'який обмін ваг.

При необмежених обчислювальних ресурсах найкращим способом регуляризації моделей фіксованого розміру є усереднення передбачення за всіма можливими налаштуваннями параметрів. Комбінація моделей майже завжди збільшує якість роботи, але в великих мережах використання комбінації декількох мереж є дуже затратним. Комбінування декількох мереж є найбільш корисним, коли мережі мають різну архітектуру або навчаються на різних тренувальних даних. Навчання відразу декількох мереж

досить дорога і довга процедура через дуже велику кількість параметрів мережі, що вимагає великої кількості обчислювальних ресурсів. Більше того, навчання великих мереж вимагає великої кількості навчальних даних, яких ніколи не бракує. Але навіть якщо натренувати великі мережі, то їх застосування буде також вимагати великих обчислювальних ресурсів.

Dropout — технологія, що дозволяє вирішити обидві ці задачі. Вона запобігає перенавчанню і дозволяє ефективно комбінувати мережі з різними архітектурами. Термін „dropout" означає видалення деяких вузлів в нейронних мережах.

Під видаленням вузлів розуміється тимчасове виключення вхідних і вихідних зв'язків нейрона. Вибір нейрона, який буде видалений, випадковий. У найпростішому випадку ймовірність виключення будь-якого вузла однакова і вибирається, виходячи з конкретного завдання.

Застосування dropout до нейронних мереж створює кілька „розріджених" мереж всередині однієї. Кожна розріджена мережа містить вузли. Мережа, що складається з n нейронів, може утворити розріджені мережі. Ці мережі мають загальні ваги, тому загальне число параметрів буде менше. Для обробки кожного тренувального випадку, вибирається розріджена мережа і навчається. Таким чином, навчання нейронної мережі з використанням dropout можна звести до навчання набору з виряджених мереж із загальними вагами, де кожна мережа навчається набагато рідше, ніж всі в сукупності. При тестуванні мережі неможливо усереднити вихід кожної мережі через їх експоненціальну кількість. Але на практиці добре працює інший метод усереднення. Ідея в тому, щоб використовувати єдину мережу без dropout. Ваги даної мережі є стислими вагами тренованих мереж. Якщо якийсь нейрон мав можливість включення в мережу при навчанні то, під час тестування вага буде помножена на дану ймовірність.

Використання dropout як усереднення веде до поліпшення якості роботи мережі, зменшення помилки на широкому спектрі завдань класифікації в порівнянні з іншими методами регуляризації.

2.5.9. Подання звуку в цифровому вигляді

Звукові сигнали, які чує людина можуть бути представлені як результат підсумовування сигналів різної частоти і амплітуди. Результатом уявлення аналогового звукового сигналу в комп'ютері є цифровий звук. Найпростішим методом перетворення звуку в цифровий вигляд є імпульсно-кодова модуляція (ІКМ). ІКМ полягає в поданні послідовності миттєвих значень амплітуди звукових коливань вимірюваних аналого-цифровим перетворювачем, через рівні проміжки часу. Кількість вимірювань, які виробляються за одиницю часу називають частотою дискретизації. Кількість біт, які відводяться для представлення одного значення сигналу, називають розрядністю. Чим вище розрядність аудіофайлу, тим більше різних значень амплітуд можна описати в кожному фреймі. Якість цифрового звуку тим вища, чим вища розрядність, і чим більша їх кількість.

Для завдання музики дуже важливо вибрати спосіб представлення інформації, він повинен досить точно описувати звуковий сигнал і в той же час мати структуру, яка сприяє виділенню патернів сигналу. Звичайне представлення звуку у вигляді набору значень амплітуд сигналу з максимальною точністю описує звуковий сигнал.

Але структура даного подання є такою, що виділяти з неї якісь шаблони є дуже складним завданням, через високе різноманіття представлень різних сигналів. Наприклад, навіть елементарний зсув по фазі повністю змінить представлення одного і того ж сигналу.

Більш правильним способом подання сигналу є частотна діаграма — таке представлення, в якому звуковий сигнал розбивається на фрейми. Кожен фрейм представляється кінцевим набором різних частот. Для кожної частоти вказується її амплітуда. Тобто сигнал на деякому інтервалі часу апроксимується сумою гармонійних коливань різної частоти і амплітуди. Таке уявлення зменшує точність представлення звуку, так-як кінцевий набір гармонійних сигналів, причому в даних сигналах не враховується зсув сигналу по фазі.

Це подання може бути отримано методом дискретного перетворення Фур'є [12]. Дане перетворення розкладає дискретну функцію на суму

синусоїдальних сигналів різної частоти.

Наведемо формули дискретного перетворення Фур'є.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn}, \quad k = \overline{0,N-1} \tag{2.18}$$

- N — кількість дискретних значень функції на обраному для розкладання інтервалі часу;
- $x_0, x_1, ..., x_n$ — значення амплітуди сигналу на інтервалі;
- $X_0, X_1, ..., X_n$ — комплексні значення амплітуд синусоїдальних сигналів;
- k — індекс частоти. Значення частоти k-го сигналу дорівнює $\frac{k}{T}$, де T — тривалість інтервалу часу на якому проводиться перетворення.

Дискретне перетворення Фур'є розкладає дискретну функцію на суму синусоїдальних сигналів частотами від 1 коливання за період до N коливань за період. Властивістю дискретного перетворення є те, що високочастотні коливання не можуть бути коректно представлені. Тому друга половина з N комплексних амплітуд є дзеркальним відображенням першої і не несе додаткової інформації.

Властивість людського слуху така, що він є більш чутливим до звуків низької частоти, і менш чутливим до звуків в області високих частот. З огляду на дану властивість, стає очевидним недоліком подання звуку у вигляді частотної спектрограми, а саме використання герц в якості одиниці вимірювання частоти, щільність розподілу яких неефективна, якщо розглядати її по відношенню до людського слуху. Зазначене подання має надмірний дозвіл в області високих частот.

Як інше подання сигналу, яке не має вищевказаний недолік, є спектрограма.

Мел — психофізична величина вимірювання висоти звуку — кількісна оцінка висоти, яка заснована на статистичній обробці великого числа даних про суб'єктивне сприйняття висоти звукових тонів. Дана залежність

описується формулою (2.19).

$$m = 1127.01048 \ln\left(1 + \frac{f}{700}\right), \tag{2.19}$$

де m — висота звуку в Гаммелах, f — висота звуку в герцах.

Використовуючи мел-частотну спектрограму, отримуємо спектрограму, в якій, в області високих частот, амплітуди близьких частот як-би складаються, тим самим посилюючи сигнал, але стискаючи дозвіл по частоті [13]. Залежність мел від герц приведена на рис. 2.7.

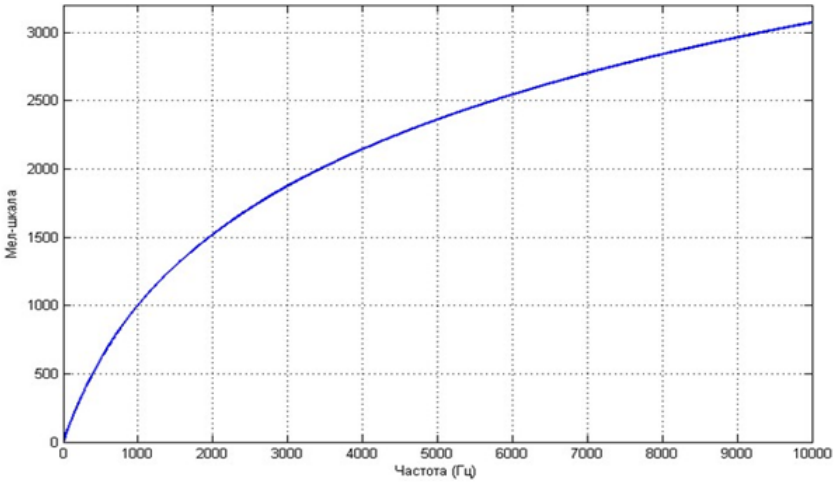

Рис. 2.7. Залежність мел від герц

2.5.10. Висновки до розділу

В даному розділі були розглянуті теоретичні основи необхідні для даної роботи. А саме: системи розділення, принципи їх роботи і принципи формування оцінок для об'єктів; штучні нейронні мережі, типи і принципи функціонування різних шарів, що застосовуються в згорткових нейронних мережах, функції активації і функції втрат а також специфіка їх застосування в різних завданнях машинного навчання, методи навчання нейронних мереж і метод запобігання перенавчання мережі; цифрове представлення звуку і особливість сприйняття його людиною.

РОЗДІЛ 3

МЕТОДИ ПЕРЕТВОРЕННЯ МУЗИЧНОЇ КОМПОЗИЦІЇ З МУЗИЧНОГО ФАЙЛУ В МЕЛ-СПЕКТРОГРАМУ

3.1. Конвертація MP3 файлу в WAV

У цьому розділі буде розглянута реалізація методу отримання мел-спектрограми зі звукового файлу. Найпопулярнішим, на сьогоднішній день, форматом зберігання музичних файлів є формат MP3. MP3 — формат кодування звукової інформації з втратами. Але працюючи безпосередньо зі стисненим MP3-файлом неможливо отримати уявлення про аудіосигнали, тому його потрібно переводити в формат без стиснення. Зручним для роботи є формат WAV — формат-контейнер для зберігання звукового оцифрованого потоку. З WAV-файлу можна витягти дискретне уявлення сигналу у вигляді імпульсно кодової модуляції, до якої можна застосувати дискретне перетворення Фур'є і отримати частотну спектрограму. До отриманої спектрограми, в свою чергу, застосувати деякі перетворення, для отримання з неї мел-спектрограми, яка і буде вхідними даними для CNN.

Дане перетворення виконується за допомогою програми, наданої самими розробниками формату MP3, ffmpeg [14]. FFMPEG — набір вільних бібліотек з відкритим вихідним кодом, які дозволяють записувати, конвертувати і передавати аудіо- та відеозаписи в різних форматах. FFMPEG складається з набору утиліт, одна з яких дозволяє конвертувати MP3 файли в WAV за допомогою консольного інтерфейсу. При конвертації потрібно вказати вхідний файл, вихідний файл, кодек, який буде використаний у вихідному файлі, а також ряд додаткових параметрів, такі як кількість каналів і частота семплірування.

Для данної задачі можна використовувати одноканальні WAV-файли з частотою семплірування — 44100 Гц. Кодек, який використовується для кодування вихідного файлу, — pcm_s16le, даний кодек використовує для

подання імульсно кодової модуляції звуку, і для кожної семпли сигналу зберігає його амплітуду у вигляді 16-ти бітового знакового числа [15].

Після конвертації файлу даним способом, буде отримано одноканальний WAV-файл, з частотою семплювання 44100 ГЦ і 16-ти бітовим знаковим числом, що представляють амплітуду сигналу для кожної семпли. Бітрейт такого файлу дорівнює 88200 байт/с. Після перетворення файлу з формату mp3 в wav, необхідно з отриманого файлу витягти інформацію про сигнал, для цього необхідно прочитати даний файл.WAV - файл має фіксовану структуру [16], вона наведена в таблиці 3.1.

Зміщення від початку файлу (байт)	Ім'я поля	Розмір поля (байт)
0	ChunkID	4
4	ChunkSize	4
8	Format	4
12	Subchunk1ID	4
16	Subchunk1Size	4
20	AudioFormat	2
22	NumChannels	2
24	SampleRate	4
28	ByteRate	4
32	BlockAlign	2
34	BitsPerSample	4
36	Subchink2ID	4
40	Subchunk2Size	4
44	data	Subchunk2Size

Табл. 3.1. Структура WAV-файлу

З усіх полів представлених в цій таблиці для нас є цікавим лише поле data, яке зберігає безпосередньо сам сигнал.

Для читання wav-файлу існують розроблені модулі на різних мовах

програмування, що дозволяє читати довільний wav-файл, і отримувати дані з будь-якого поля файлу. Читання блоку data здійснюється в цілочисельний масив. Надалі до цього масиву можна застосувати дискретне перетворення Фур'є.

3.2. Застосування дискретного перетворення Фур'є до сигналу

Після отримання сигналу з wav-файлу у вигляді набору значень амплітуд, взятих через рівні інтервали часу, можна побудувати частотну спектрограму, застосовуючи до сигналу дискретне перетворення Фур'є.

Якщо застосувати ДПФ до всього треку, то отримана спектрограма буде характеризувати трек вцілому і не буде нести якоїсь корисної інформації для даної задачі, так як спектрограма матиме високу роздільну здатність по частоті, і низький дозвіл за часом.

Тому потрібно знайти компроміс, між тим, щоб зберегти достатній дозвіл по частоті, але в той же час добитися прийнятного дозволу за часом. Для цього розіб'ємо вихідний сигнал на вікна однакового розміру, і будемо застосовувати ДПФ до кожного вікна. Залишається тільки вибрати розмір вікна. У даній роботі було вибрано вікно тривалістю $\sim 1/8$ секунди. При такій тривалості вікно буде містити 5512 семплів вихідного сигналу.

За властивостями ДПФ спектрограма буде містити 5512 фіксованих наборів частот, діапазону від 0 Гц до 44088 Гц, але через муаровий ефект, верхня половина даних складових буде дзеркальним відображенням нижньої, і тим самим спектр буде містити 2756 фіксованих частот в діапазоні від 0 Гц до 22048 Гц. Даний спектр охоплює весь, чутний людиною, діапазон частот. Але для даного завдання хотілося б зменшити обсяг даних, тому що чим більше даних, тим довше буде навчатися нейронна мережа.

Діапазони більшості музичних інструментів лежать в інтервалі від 40 Гц до 5000 Гц, людський вокал в діапазоні $80 - 10000$ Гц. Крім даних частот також породжуються обертони, верхні частоти яких, для деяких інструментів, доходять до 18000 Гц. Але в нашому випадку ми можемо

дозволити собі знехтувати частиною даного діапазону, так як обертони надають вишуканість звучанням, але ми націлені виявити більш змістовну частину музичної композиції, ніж відстежити всі її найтонші моменти. Тому було вирішено урізати діапазон, що представляється в спектрограмі, до верхньої частоти 11024 Гц, тим самим, кількість різних частот знизилася до 1378.

Підводячи підсумок, на виході після застосування ДПФ і обрізання частини частотного спектра ми отримали спектрограму аудіотрека, розбитого на вікна тривалістю $\sim 1/8$ секунди з 1378 фіксованими частотами в діапазоні від 0 Гц до 11024 Гц. Значення потужності конкретної частоти представляється 4-х байтовим числом з плаваючою точкою, в такому випадку бітрейт отриманої спектрограми складе 44096 байт/с.

3.3. Перетворення частотної спектрограми в мел-спектрограму

Вихідними даними перетворення є частотна спектрограма, отримана дискретним перетворенням Фур'є, де N — кількість сигналів різної частоти, що формують спектрограму, результатом перетворення є мел-спектрограма. Перетворення звичайних частот в мел-шкалу проводиться за допомогою набору трикутних вікон [17] зображених на рис. 3.1.

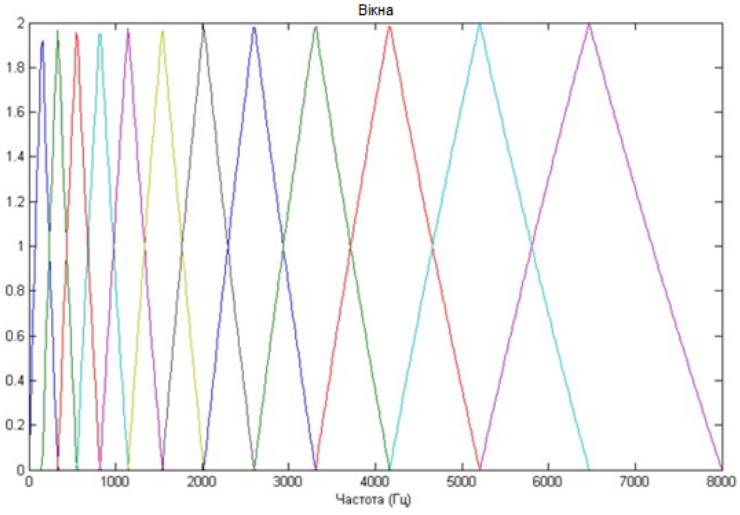

Рис. 3.1. Гребінка трикутних вікон, що описує функцію (3.1)

$$H_m = \begin{cases} 0, & k < f[m-1] \\ \dfrac{k - f[m-1]}{f[m] - f[m-1]}, & f[m-1] \le k < f[m] \\ \dfrac{f[m+1] - k}{f[m+1] - f[m]}, & f[m] \le k \le f[m-1] \\ 0, & k > f[m+1] \end{cases} \tag{3.1}$$

$$f[m] = \frac{N}{F_s} B^{-1}\left(B(f_1) + \frac{m\,(B(f_h) - B(f_1))}{M+1}\right) \tag{3.2}$$

де f_1 — нижня межа значень частоти герцах, а f_h — верхня межа значення частоти в герцах.

$$B(f) = 1127.01048 \ln\left(1 + \frac{f}{700}\right) \tag{3.3}$$

$$B^{-1}(b) = 700 e^{\frac{b}{1127.01048}} - 1 \tag{3.4}$$

(3.3) — перетворення частоти в мел-шкалу. (3.4) — зворотне перетворення частоти з мел-шкали в частоту в герцах.

Параметром даного перетворення є кількість різних значень мел-частот. Тут знову постає питання вибору оптимального значення між тим, щоб надати високий частотний дозвіл або забезпечити невеликий обсяг даних, необхідних для опису треку. Визначити теоретичними дослідженнями необхідний дозвіл по частоті для подібних завдань поки не вдалося. Але з наявних практичних напрацювань в цій сфері було виявлено, що 128 фіксованих частот цілком достатньо [18]. Якщо спробувати знайти теоретичне підґрунтя, то даний факт можна пояснити наступним чином: у діапазоні від 0 Гц до 11000 Гц лежить приблизно 10 октав, кожна з яких містить по 12 різних нот, отримуємо 120 різних нот на інтервалі, тому можна припустити, що дане подання має достатній дозвіл по частоті. Бітрейт отриманого подання дорівнює 4096 байт/с.

3.4. Аналіз отриманої мел-спектрограми

Отримана мел-спектрограма має 128 різних значень уздовж осі частот, а вздовж осі вікна часу фрейми тривалістю 1/8 секунди. Необхідно вирішити яку частину треку взяти в якості вхідних даних нейронної мережі. Якщо проводити аналогію зі згортковими нейронними мережами, що застосовуються для класифікації зображень, то можна припустити, що розміри вхідних даних цих мереж і мережі, яка застосовується в даній роботі, повинні приблизно збігатися. Більшість ШНМ використовують квадратні зображення розміром близько 100 пікселів. Опираючись на цей факт, можна зробити висновок, що слід використовувати в якості вхідних даних спектрограму, що має 128 фреймів уздовж осі часу, але з огляду на обчислювальні можливості технічних засобів, на яких буде проводитися навчання мережі, було вирішено взяти не 128 фреймів, а 64, таким чином, вхідні дані будуть представляти 8 секундній аудіотрек.

Крім розміру вхідних даних потрібно звернути увагу на діапазон значень вхідних спектрограм. Звертаючись до інших прикладів ШНМ, для зображень найчастіше використовуються кілька каналів зі значеннями інтенсивності від 0 до 255. Для інших завдань використовуються безперервний інтервал $[-1; 1]$. У данній роботі також будемо використовувати значення спектрограми в інтервалі $[-1; 1]$.

3.5. Висновки по розділу

У даному розділі було розглянуто метод отримання мел-спектрограми з MP3 файлу. Обрані конкретні параметри спектрограми на основі [18] і міркувань обчислювальної складності навчання. Був описаний модуль, який можна використовувати для побудови спектрограми, що дозволяє гнучко налаштовувати параметри спектрограми, такі як, кількість різних мел-частот, тривалість одного фрейму, верхній рівень частоти відображений в спектрограмі.

РОЗДІЛ 4

РЕАЛІЗАЦІЯ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ РОЗДІЛЕННЯ СИГНАЛІВ

4.1. Вхідні дані мережі

Для вирішення завдання розділення сигналів в даній роботі використовується згорткова нейронна мережа, яка буде апроксимувати розділення [23]. Вхід нейронної мережі — мел-спектрограма, яка представляється двовимірною матрицею дійсних чисел, в попередніх позначеннях кожна мел-спектрограма це елемент для розділення сигналів, при такому підході, вибір значення параметра, функції [21], є не дуже простим завданням. Тому замість фіксованого значення параметра, його можна визначати динамічно наступним чином. Спочатку обчислити значення функції, для всіх треків $d \epsilon D$ — треки, що належать до вибірки, а потім відсортувати їх за отриманим значенням ймовірності, і вибрати з отриманого рейтингу, його деяку частку для розділення.

Загальна ідея побудови згорткової нейронної мережі така, щоб на перших шарах виділити деякі характеристики звуку використовуючи згорткові шари, а потім, за допомогою повнозв'язних шарів, сформувати з набору примітивів розділення сигналів.

Недоліком даного способу є необхідність тривалого навчання нейронної мережі для кожного користувача через велику кількість нейронів в ній. Найдієвішим способом зменшення часу навчання мережі є зменшення її розміру, і тут два варіанти, або зменшувати кількість нейронів в кожному шарі, але тоді, можливо, мережа не зможе описати необхідну складну багатовимірну залежність, або зменшення кількості шарів, що буде причиною зменшення рівня ознак, що виділяються мережею. Виходом з даної ситуації може бути розбиття нейронної мережі на дві окремі незалежні частини. Перша мережа буде попередньо обробляти дані, тобто з вихідної мел-спектрограми формуватиме набір більш високорівневих ознак, ніж сама мел-спектрограма. Друга мережа буде формувати з високорівневих

ознак, вироблених на виході першої мережі, оцінку ймовірності, що трек подобається користувачеві.

Вхідними даними мережі є мел-спектрограма, отримана з частини музичного сигналу, тривалістю 8 секунд. Для розділення цього може виявитися недостатньо. Дійсно, навіть програмі, яка розділяє сигнали, складно визначити чи розділений звуковий сигнал чи ні, обробивши всього 8 секунд треку.

Тому для розділення будемо розбивати звуковий сигнал на частини по 8 секунд, і результатом розділення буде середнє арифметичне значення видане мережею для всіх частин.

4.2. Мережа попередньої обробки

Розглянемо докладніше мережу попередньої обробки даних, яка буде використовуватися для формування вхідних даних розділення сигналу. Основним завданням даної мережі є зменшення розмірності вхідних даних, і збільшення рівня абстракції ознак, тобто формування з мел-частотної діаграми деякого набору характеристик треку. Основне питання, що виникає тут: які характеристики повинні формуватися на виході мережі? Наприклад, це можуть бути характеристики дуже високого рівня, наявність в ньому будь-яких інструментів і їх специфічних партій (гітарне соло, духові інструменти), характеристика вокалу виконавця і т.д. Але такі характеристики можуть дуже загально описувати трек, і таких даних недостатньо для розділення. Або характеристики будуть менш високого рівня, що дозволить описати вхідний сигнал досить точно, але для виділення таких характеристик потрібна база для навчання мережі, яку дуже складно сформувати, так як виділити такі характеристики непросте завдання для людини. Вирішення проблеми, на перший погляд, здається неможливим, ми хочемо виділити певний набір характеристик з мел-спектрограми за допомогою нейронної мережі, але для її навчання потрібна розмічена база, якої не існує і сформувати яку при наявних ресурсах неможливо.

Для вирішення даної проблеми скористаємося властивостями згорткової нейронної мережі. Як відомо, згорткові шари виділяють примітиви

з вхідного сигналу, причому кожен наступний шар виділяє примітиви все більш високого рівня. Це наводить на думку, що якщо у нас є згорткова мережа, у якій на вході мел-спектрограма, а вихід належить простору не дуже високої розмірності, то на прихованих шарах даної мережі будуть формуватися ознаки, які менш точно описують трек, ніж вихідні дані, але вже більш високорівневі, ніж сигнал на вході мережі. Дане припущення, хоча і не має хорошого теоретичного підтвердження, але доводиться на практиці. Таким чином, рішення буде виглядати так: побудувати нейронну мережу, вхід якої мел-спектрограма, а вихід — деякий вектор невисокої розмірності, такий, щоб можна було скласти розмічену базу, придатну для навчання. Навчити дану мережу, а потім „відрізати" кілька останніх шарів, і використовувати як попередню обробку даних її перші шари. Найпростішою мережею, яка підходить під даний опис є мережа класифікації жанрів. Її головною перевагою є простота формування навчальної бази, так як існує величезна кількість музичних треків, для яких вказано жанр.

4.3. Мережа розділення

Мережа, вирішальним завданням якої є розділення, буде приймати на вхід дані вже попередньо оброблені за допомогою мережі, описаної вище, а на виході у даній мережі буде оцінка ймовірності того, що конкретний трек розділений на сигнали.

З входом і виходом даної мережі всі питання вирішені, але залишається визначити, на яких даних навчати дану мережу. Очевидно, що навчальною вибіркою повинні бути треки оцінені користувачем. Але проблема в тому, що знайти треки, не складає труднощів — це в основному ті треки, які він додає або явно оцінює, а ось треки які користувач чути б не хотів знайти досить складно, тому навчальна вибірка, швидше за все, буде зміщена в бік позитивних оцінок, і не буде придатна для навчання. Таким чином, необхідно знайти альтернативний спосіб формування навчальної вибірки. Тут знову рішенням є підхід, який не має строгого теоретичного обгрунтування, але перевірений на практиці. Він полягає в тому, щоб в якості негативних прикладів взяти великий набір треків, які не перетинаються з позитивними,

і на кожній ітерації вибирати різні набори негативних прикладів, а позитивні приклади залишати фіксованими [19].

4.4. Реалізація мережі попередньої обробки даних

Було вирішено, що мережа попередньої обробки даних буде формуватися в два етапи, спочатку побудова мережі для класифікації жанрів, потім взяти перші кілька шарів вже навченої мережі, які будуть мережею попередньої обробки. Вхід мережі — мел-частотна спектрограма. Виходом мережі попередньої обробки буде набір деяких ознак, що мають більш високий рівень, ніж мел-частотна спектрограма, але не мають „людського" пояснення їх значень.

Архітектура згорткової нейронної мережі класифікації музичних жанрів наведена в таблиці 4.1. У таблиці 4.1 наведені типи шарів, що використовуються в мережі: convolution (згортковий), maxpooling (об'єднання), dense (повнозв'язні). Дані типи шарів вже були розглянуті в попередніх розділах даної роботи. Крім них в мережі використані шари Reshape (зміна розміру), Flatten (розгортки), Dropout (виключення). Вони є допоміжними шарами, які реалізують деякі функції перетворень. Розглянемо їх трохи докладніше.

- Reshape — шар, що приводить вхідні дані до необхідної форми, єдине обмеження — кількість елементів поданих на вхід, має дорівнювати кількості елементів вихідної фігури.
- Flatten — шар, що є окремим випадком шару Reshape, розгортає подану на вхід фігуру у вектор.
- Dropout — шар реалізує метод виключення нейронів з мережі при навчанні. Параметром шару є частка виключених нейронів.

№	Тип шару	Параметри шару
1	Convolution layer	Вхід — матриця 64×128. 256 ядер згортки розміром 4×128. Функція активації — ReLu.
2	MaxPooling layer	Розмір пулу — 2×1.
3	Reshape layer	Вихід шару — $1 \times 256 \times 30$.
4	Dropout layer	Коефіцієнт відсікання — 0.25.
5	Convolution layer	128 ядер згортки розміром 256×2. Функція активації — ReLu.
6	MaxPooling layer	Розмір пулу — 1×2.
7	Flatten layer	
8	Dropout layer	Коефіцієнт відсікання — 0.25.
9	Dense layer	Кількість нейронів — 1024. Функція активації — ReLu.
10	Dropout layer	Коефіцієнт відсікання — 0.5.
11	Dense layer	Кількість нейронів — 128. Функція активації — ReLu.
12	Dropout layer	Коефіцієнт відсікання — 0.5.
13	Dense layer	Кількість нейронів — 9. Функція активації — softmax. Функція втрат — categoricalcrossentropy.

Табл. 4.1. Архітектура згорткової нейронної мережі класифікації музичних жанрів

Розглянемо докладніше всі шари наявні в нейронній мережі розділення сигналів.

Перший згортковий шар, входом якого є мел-частотна спектрограма розміром $1 \times 64 \times 128$, містить 256 ядер згортки розміром 4×128. Кожне ядро в нашому випадку буде представляти мел-частотну спектрограму

тривалістю в половину секунди. Вихід шару має розмірність $256 \times 61 \times 1$. Час змінюється уздовж другої осі.

Другий шар — MaxPooling, розмір ядра пулінгу 2×1, згортка проводиться тільки уздовж осі часу. Розмір виходу шару — $256 \times 30 \times 1$.

Третій шар — Reshape використовується для трансформації фігури 256×301 в фігуру $1 \times 256 \times 30$. Суть цієї трансформації — використовувати значення, отримані після згортки ядром, як частоти. Так як згортка відбувається тільки уздовж другої і третьої осі, а вздовж першої осі значення згортки складаються. Для наочності, результат, який ми отримуємо, проілюстрований на рис. 4.1.

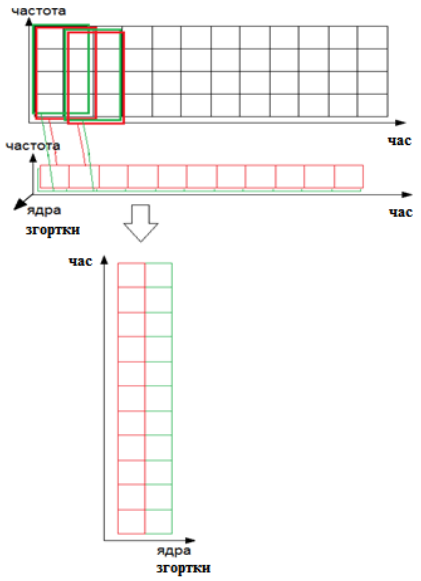

Рис. 4.1. Шари мережі розділення сигналів

Одержанний результат можна пояснити як спектрограму, складену не за частотам, а за певними більш високорівневими ознаками, і для кожного тимчасового вікна вказана вираженість ознак в цьому вікні.

Четвертий шар — Dropout з коефіцієнтом виключення 0.25.

П'ятий шар — Convolution. Вхід $1 \times 256 \times 30$, 128 ядер згортки розміром 256×2.

Активаційна функція — ReLu. Згортка знову ведеться тільки уздовж осі часу.

Шостий шар — MaxPooling, вікно пудлінга 1×2. Стиснення тільки по осі часу. Вихідна фігура має розмір $128 \times 1 \times 15$.

Сьомий шар — Flatten розгортає вхідні фігури у вектор 1×1920.

Восьмий шар — Dropout з коефіцієнтом виключення 0.25.

Шари з дев'ятого по дванадцятий чергують повнозв'язні шари і шари dropout. Тут відбувається формування ознак, отриманих на попередніх шарах.

Останній — тринадцятий повнозв'язний шар має всього 9 нейронів, що відповідає кількості різних жанрів. Функція активації даного шару — softmax. На виході даного шару — ймовірності приналежності поданого на вхід треку кожному з жанрів.

У даній архітектурі використовується categorical crossentropy функція втрат, найбільш часто застосовується в задачах багатокласової класифікації спільно з активацією softmax.

Навчання мережі проводилося на базі з 1000 восьмисекундних фрагментів з треків, які належать 9 різним жанрів: rock, hip-hop, pop, metal, reggae, drum and base, dubstep, jazz, classical. При навчанні головною метою було формування ядер згортки, а не досягнення великої точності розділення. На першому шарі були отримані ядра згортки наведені на рис. 4.2. Ядра згортки другого шару наведені на рис. 4.3.

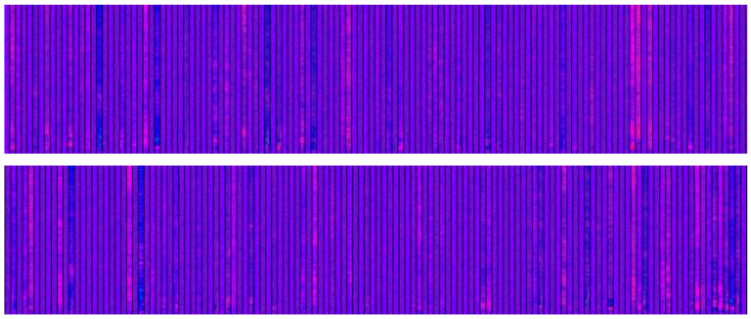

Рис. 4.2. Ядра згортки першого шару

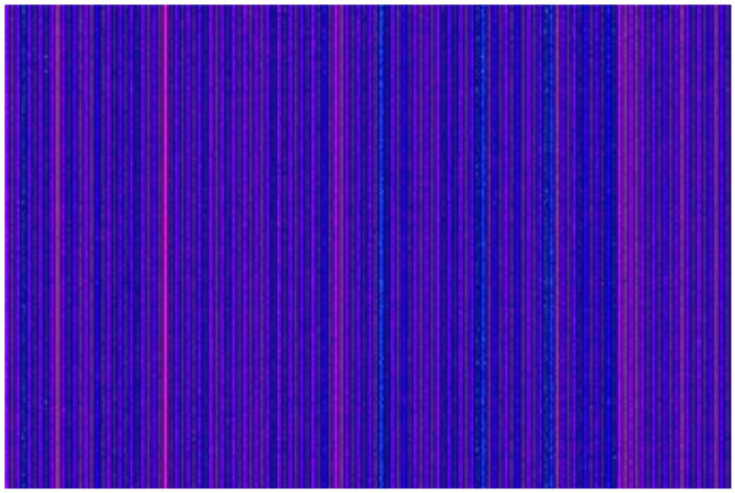

Рис. 4.3. Ядра згортки другого шару

Була побудована і навчена мережа розділення сигналів, це тільки перший етап формування мережі попередньої обробки. Необхідно вибрати частину даної мережі, яка буде обробляти мел-спектрограму. Логічним рішенням є вибір в якості такої частини тільки згорткових шарів, так як на наступних, повнозв'язних шарах, слідуючи логіці побудови мережі, починає формуватися „думка" про розподіл. Таким чином в якості мережі попередньої обробки були обрані два перших згорткових шари.

4.5. Реалізація мережі розділення сигналу

Мережа розділення сигналу повинна базуватися на основі оброблених, за допомогою мережі попередньої обробки мел-частотних спектрограм і видавати якісне розділення сигналу. Вхідні дані мережі мають розмір 1×1920. Для реалізації мережі були обрані звичайні повнозв'язні шари і dropout шари, так як навчальна вибірка є досить маленького розміру, і мережа буде дуже схильна до перенавчання. Наведемо конкретний опис кожного шару.

- Перший шар — flatten розгортає вихідну матрицю препроцесора у вектор.
- Другий шар — повнозв'язний шар з 1024 нейронами і сигмоїдальною функцією активації.
- Третій шар — dropout з коефіцієнтом виключення 0.5.

- Четвертий — повнозв'язний шар з 128 нейронами і сигмоїдальною функцією активації.
- П'ятий шар — dropout з коефіцієнтом виключення 0.5.
- Шостий шар — вихідний повнозв'язний шар з одним нейроном і сигмоїдальною функцією активації.

Як вже говорилося раніше в цьому розділі, навчальна вибірка формується з треків, позитивно оцінених користувачем, і випадково вибраних треків, які не перетинаються з безліччю позитивних прикладів, які виступають в якості негативних, причому на кожному етапі навчання безліч негативних прикладів змінюється.

У мережі використовується середньоквадратична функція втрат, яка найбільш часто застосовується в задачах регресії, якою, по суті, є дана задача. Так як вхід даної мережі формується всього з 8-ми секунд треку, то очевидним є недостатня кількість даних для опису його цілком. Логічним є взяти кілька 8-ми секундних фрагментів з кожного треку, для більш точного опису кожного з них. Зробити це можна двома способами. Перший — взяти кілька поспіль фрагментів, і згрупувати результати обробки цих фрагментів в один вектор, на якому проводиться навчання. Другий — брати кілька випадкових частин треку по 8 секунд і розділяти їх окремо. Обчислювати вихід мережі в першому випадку потрібно від фрагмента треку, тривалістю як при навчанні. У другому — розділяти весь трек на безліч 8-ми секундних фрагментів і обчислювати розділення треку — як середнє від оцінки кожної з цих частин. Був реалізований другий підхід, основним критерієм вибору була обчислювальна складність. Перший спосіб має набагато вищу обчислювальну складність через велику кількість вхідних даних, другий же, вимагає набагато менше обчислювальних ресурсів, а, отже, і часу на навчання.

4.6. Висновки по розділу

У результаті проведеної в цьому розділі роботи побудовані дві нейронні мережі: попередньої обробки і розділення. Мережа попередньої обробки була навчена для виділення більш високорівневих характеристик розділе-

ння, ніж мел-частотна спектрограма. Використання мережі попередньої обробки даних дозволяє знизити час навчання мережі розділення, за рахунок зменшення розмірності ознак і збільшення їх рівня. Мережа розділення буде навчатися для кожного випадку окремо. Даними для навчання мережі розділення є треки користувача, які він додав до свого списку. Негативні приклади вибираються випадково з більшої безлічі сигналів.

РОЗДІЛ 5

ТЕСТУВАННЯ ЯКОСТІ РОЗДІЛЕННЯ СИГНАЛІВ

5.1. Експеримент з розділенням сигналів

Мета даного експерименту — оцінити „знизу” якість розділення сигналу, тобто перевірити здатність видавати результат, де з людської точки зору це легко зробити. В якості навчальної вибірки мережі були обрані дуже близькі треки, в яких людина може на слух розділити сигнал. Наведемо список обраних треків:

- Drowning Pool „Let The Bodies Hit The Floor”
- Drowning Pool „One Finger And A Fist”
- Drowning Pool „Hate”
- Godsmack „I Stand Alone”
- Northlane „Set In Stone”
- Otep „I alone”
- Otep „Rise Rebel Resist”
- Otep „Warhead”
- POD „Boom”
- Rancid „Out of control”
- Sarah Where Is My Tea „This Is Not Twilight”
- Static-X „The Only”
- Symphony X „Set The World On Fire The Lie Of Lies”
- Varg „Apokalypse”
- WWE „CM Punk”
- Adept „Shark”
- Nirvana „Smells like teen spirit”
- Asking Alexandria „Just A Slave To Rock and Roll”
- Cancer Bats „Grand Canyon”
- Cerebral Bore „Maniacal Miscreation”
- Divine Heresy „Monolithic Doomsday Devices”

Для навчання з кожного треку випадковим чином були обрані 3 восьмисекундних фрагмента. Для кожного з них була побудована мел-частотна спектрограма, яка потім оброблена мережею попередньої обробки даних. Отриманому числу позитивних прикладів з 63 об’єктів в якості мітки було присвоєно число 1.

В якості негативних прикладів були обрані фрагменти треків, що

використовуються для навчання мережі попередньої обробки. На цій безлічі на кожному етапі навчання вибиралися випадковим чином 63 фрагмента, яким призначалася оцінка -1.

Тестування результатів навчання проводилося на вибірці з 18 треків, кожен з них був розбитий на 8-секундні фрагменти. Оцінка для треку обчислювалася як середнє серед оцінок всіх фрагментів одного трека. В результаті експерименту були отримані наступні оцінки для треків:

- Nirvana „Smells like teen spirit”: 0.6961
- System Of A Down „Sugar”: 0.6770
- Disturbed „Run”: 0.6595
- Rammstein „Mann Gegen Mann”: 0.5461
- Machinae Supremacy „Edge And Pearl”: 0.5147
- Bring Me The Horizon „Happy Song”: 0.5074
- Mushroomhead „Kill Tomorrow”: 0.4037
- Static-X „The Only”: 0.3836
- Rob Zombie „Feel So Numb”: 0.3677
- Heart Of A Coward „Miscreation”: 0.3467
- Silverstein „Face of the Earth”: 0.3384
- Zebrahead „Call Your Friends”: 0.3132
- Cervello „Carry Me Home”: 0.3091
- Girl on Fire „The Takedown”: 0.2679
- Slipknot „Skeptic”: 0.2485
- Starset „Point of No Return”: 0.0619
- Noize MC „Выдыхай”: 0.0295
- Pendulum „Blood sugar”: 0.0034

Проаналізувавши отримані результати, можна зробити висновок, що середня оцінка на навчальній вибірці дорівнює 0.7207. Серед протестованих треків є кілька треків, оцінки яких лежать дуже близько до максимальної. Побудуємо графік залежності між шириною інтервалу і кількістю треків, що потрапили в нього, рис. 5.1.

З графіка видно, що лише 4 треки потрапляють в 20%-й інтервал, за суб’єктивними оцінками було з’ясовано, що це саме ці треки вдалося розділити. В інтервалі від 20 до 80% лежать треки, які розділити вдалося,

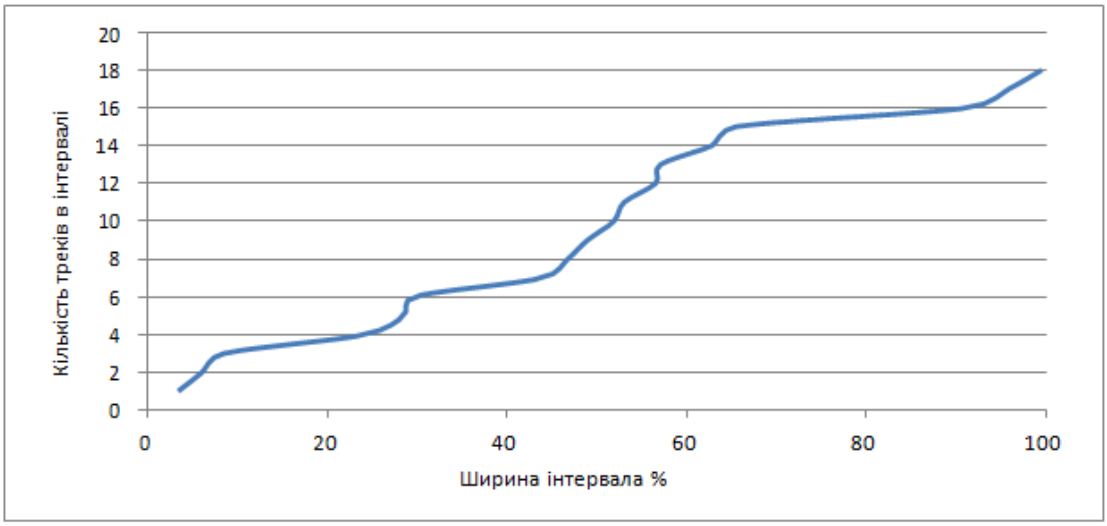

Рис. 5.1. Графік залежності між шириною інтервалу і кількістю треків

але якість розділення має середнє значення, і нарешті, в інтервалі від 80 до 100% лежать ті, які абсолютно не вдалося розділити. Отримані результати показують працездатність побудованої системи, вона здатна розділити треки, але це лише слабкий тест, який оцінює якість отриманої системи знизу. Також необхідно провести експеримент, де в якості розділених треків будуть виступати не тільки близькі за звучанням треки.

5.2. Експеримент з розрізненими треками

Мета даного експерименту — оцінити якість розділення, які видаються мережею для користувача, в списку якого знаходяться не тільки треки одного з жанрів, а розрізнені по стилю і жанру треки.

В якості навчальних даних було взято 31 трек. Для оцінки якості з цієї множини обрані 6 треків — безліч валідації. Кожен трек був розбитий на 8-ми секундні фрагменти. Отримана кількість, за винятком безлічі валідації, використовувалася як позитивний приклад, як негативний був узятий набір з 800 треків, які також були розбиті на 8-ми секундні фрагменти. З отриманої безлічі на кожному етапі навчання, які випадковим чином вибиралися з безлічі, по потужності рівня безлічі позитивних прикладів. По закінченню навчання на вхід мережі були подані треки з безлічю валідації і позитивно оцінені треки з навчальної вибірки. Для безлічі валідації були отримані наступні оцінки:

- + Pendulum „Slam”: 0.00118992977687
- + Green Day „Brain stew”: 0.00119383690374
- + Nirvana „Heart-Shaped Box”: 0.00124476053615
- + Sum 41 „Fat Lip”: 0.417011744501
- − Florence and The Machine „What Kind Of Man”: 0.00867434248866
- − A-Ha „Forever Not Yours”: 0.00897150128601
- − Katy Perry „Last Friday Night”: 0.0152586640373
- − Maroon 5 „Misery”: 0.234964552129

Тут ,,+″ відзначені позитивні приклади і ,,−″ — негативні. Для треків з навчальної вибірки середня оцінка: 0.511999682439.

На рис. 5.2 наведено графічне представлення отриманих результатів.

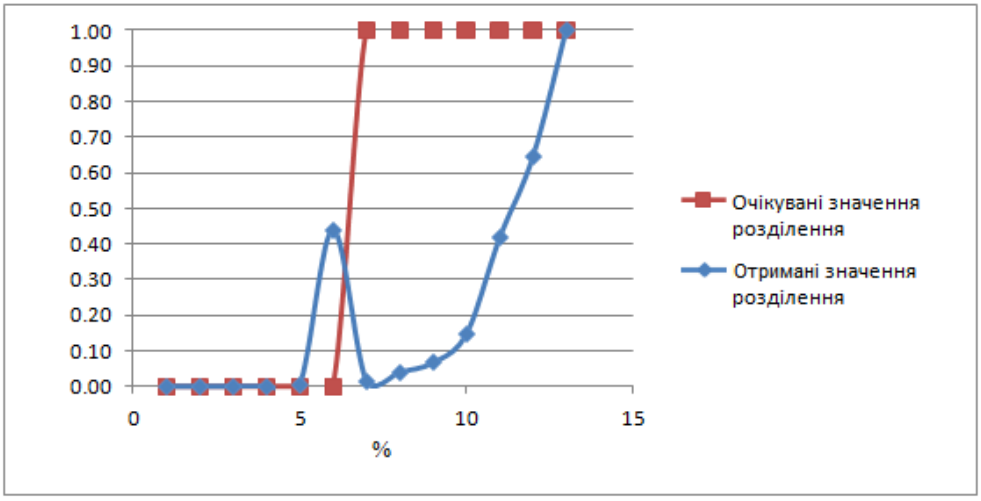

Рис. 5.2. Графічне представлення отриманих результатів

З одного боку максимальну ймовірність до розділення має трек, оцінений користувачем позитивно, з іншого — інші негативно оцінені треки мають оцінку вище, ніж позитивні, хоча їх оцінки близькі до нуля, і немає принципової різниці, який з треків отримав велику оцінку.

Для більш точної оцінки роботи мережі було проведено експеримент схожий за змістом, але на іншій вибірці. Для даного експерименту на валідаційні безлічі були отримані наступні оцінки, (позначення ,,+″ і ,,−″ зберігаються).

Середня оцінка для позитивних треків: 0.490256355655. Графічне представлення отриманих результатів наведено на рис. 5.3.

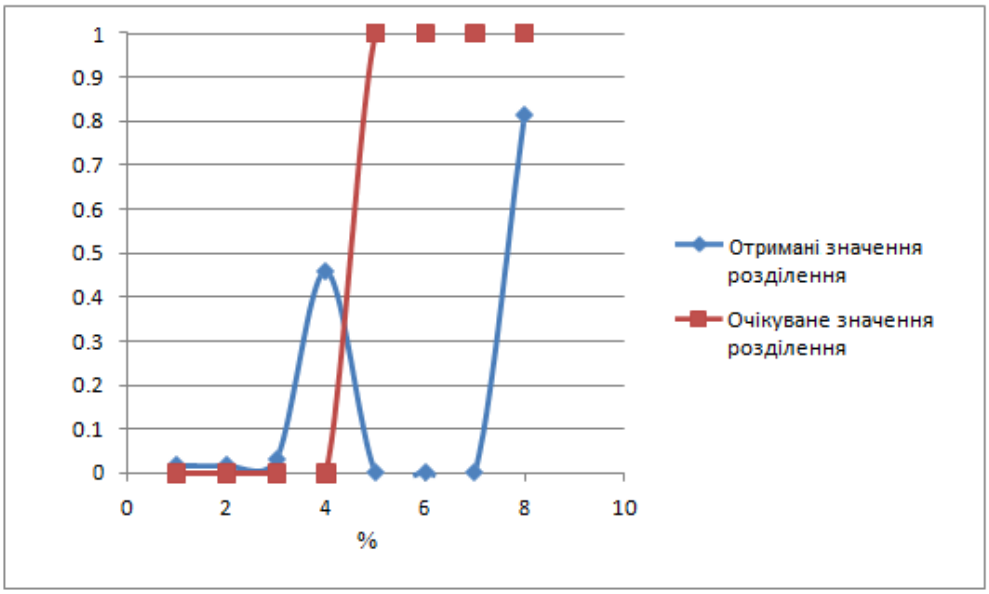

Рис. 5.3. Графічне представлення отриманих результатів

Наведемо отримані оцінки для данного експерименту:

- + Баста „Забытые”: 0.00750642064509
- + Макс Корж „Горы по колено”: 0.0193998943406
- + Noize MC „Антены”: 0.0320982784903
- + Oxxxymiron „Переплетенные”: 0.0712569377295
- + Anacondaz „Фотошоп”: 0.203862335311
- + Баста „Раз и навсегда”: 0.80903792722
- + Noize MC „Ты не считаешь”: 0.315852694516
- - Artik and Asti „Тебе все можна”: 0.215541947259
- - Amaranthe „Razorblade”: 0.00242544749957
- - AMATORY „Остановить время”: 0.000377503782505
- - Billboard, Britney Spears „Till The World Ends”: 7.62535008189e-05
- - Beth Hart „Thru The Window Of My Mind”: 1.39174066103e-05
- - Billie Holiday „Moonglow”: 4.17873211104e-06

Можна зробити висновок, що отримані результати показують досить високу якість розділення. Серед чотирьох високо оцінених треків, тільки один трек є погано розділений.

5.3. Висновки по розділу

Отримані, в ході експериментів, розділення показують працездатність системи на основі нейронної мережі. Хоча якість одержуваних розділень не завжди добра, але в більшості випадків результати є адекватними і передбачуваними. Дана система здатна, як мінімум, здійснити середнє розділення звуку.

ВИСНОВКИ

В ході цієї роботи мета, поставлена у вступі, була досягнута. А саме, було розроблено алгоритм сліпого поділу джерел звуку в монофонічному музичному аудіосигналі з використанням штучних нейронних мереж.

Були вирішені наступні завдання:

- Зроблено огляд методів поділу джерел звуку, зокрема в музичному аудіосигналі;
- Зроблено огляд архітектур CNN, які застосовуються в задачах розділення джерел звуку;
- Розроблено архітектуру CNN: зроблено опис алгоритму навчання;
- Здійснено підготовку навчального, валідаційного і тестового наборів;
- Здійснено реалізацію розробленої архітектури;
- Зроблено оцінку ефективності за об'єктивними критеріями.

В ході проведених досліджень було зроблено ряд наступних висновків. Навчання CNN на бінарних частотно-тимчасових масках, в порівнянні з м'якими масками, істотно полегшує завдання мережі. Збільшення довжини відрізків спектрограм, що подаються на вхід мережі, робить позитивний вплив на якість рішення. Збільшення числа карт ознак на більш глибоких рівнях CNN є оптимальним для CNN, які використовуються для вирішення завдання поділу джерел звуку.

Також варто відзначити, що існує ряд способів підвищення ефективності розробленого алгоритму. По-перше, навчену на першому етапі на бінарних масках мережу можна довчити на м'яких масках. По-друге. можуть бути застосовані різні способи обробки виходів CNN.

В результаті виконання даної роботи була побудована нейронна мережа для розділення музичних сигналів. Було проведено тестування якості розділення сигналів та оцінка ефективності мережі. Якість розділення можна оцінити як середню. Плюсом даної системи є те, що вона може створювати розділення для сигналів, які прослуховуються малою кількістю. Мінусом системи є висока обчислювальна складність, навчання мережі вимагає більших обчислювальних ресурсів, ніж алгоритми, що використовуються в коллаборативних системах. Для поліпшення даної системи можна

спробувати змінити мережу попередньої обробки даних, або використовувати для її навчання навчальну вибірку більшого обсягу. Більш якісні вхідні дані мережі розділення повинні сприяти отриманню більш якісних показників.

В данній роботі також був описаний модуль перетворення MP3-файлів в мел-частотну спектрограму, що дозволяє гнучко налаштовувати параметри вихідної спектрограми.

СПИСОК ЛІТЕРАТУРИ

1. Горячкін О. В. (2003). *Методи сліпої обробки сигналів та їх застосування в системах радіотехніки та зв'язку.* Москва: Радіо та зв'язок.
2. Гудфелоу Я., Бенджио И., Курвіль А. (2018). *Глибоке навчання.* Москва: ДМК-Пресс.
3. Ковальова. А. А. (2018). Розробка ефективного методу ідентифікації людини за голосом. *ВКР магістра,* ст. 1 — 89.
4. Ніколенко С., Кадурін А., Архангельська А. (2018) *Глибоке навчання.* Санк-Петербург: „Питер".
5. Шульгін В. І.. Морозов А. В.. Волосюк Є. В. (2005). Використання технології „сліпого поділу джерел" під час обробки біомедичних сигналів. *Клінічна інформатика та телемедицина,* ст.42-50.
6. Bengio Y. (2009). *Learning Deep Architectures for AI Foundations and Trends in Machine Learning.* `https://www.nowpublishers.com/article/Details/MAL-006`
7. Carmo F., Assis J., Estrela V. V., Coelho A. (2009). *Blind signal separation and identification of mixtures of images.* `https://arxiv.org/ftp/arxiv/papers/1603/1603.08095.pdf`
8. Cherry E. C. (1954, July). Some Further Experiments upon the Recognition of Speech, with One and with Two Ears. *The Journal of the Acoustical Society of America,* pp. 554 — 559.
9. Doire C. S. J. (2019). Online singing voice separation using a recurrent one-dimensional u-net trained with deep feature losses. *International Conference on Acoustics, Speech and Signal Processing (ICASSP).* `https://ieeexplore.ieee.org/document/8683251`
10. FitzGerald. D. (2013, May 4 — 7). *The Good Vibrations Problem.* `https://www.researchgate.net/publication/263807651_The_Good_Vibrations_Problem`
11. Fitzgerald, D., Cranitch, M., Coyle, E. (2005). *Non-negative Tensor Factorisation for Sound Source Separation.* `https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1072&context=argcon`

12. Glorot X., Bengio Y. (2010). *Understanding the Difficulty of Training Deep Feedforward Neural Networks.* http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

13. Kaiming H., Xiangyu Z., Shaoqing R., Sun J. (2015, December). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.* https://ieeexplore.ieee.org/document/7410480

14. Heinzei G., Rudiger A., Schilling R. (2002). *Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new flat-top windows Max-Planck-Institut fur Gravita-tionsphysik.* https://holometer.fnal.gov/GH_FFT.pdf

15. Hinton G. E. (2007). Learning Multiple Layers of Representation. *Trends in Cognitive Science,* pp. 428-434.

16. Hinton G. E., Osindero S., Teh Y.-W. (2006, July). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation,* pp.1527 — 1554.

17. Jansson A., Humphrey. E., Montecchio N., Bittner R., Kumar A., Weyde T. (2017). *Singing voice separation with deep U-Net convolutional networks.* https://ejhumphrey.com/assets/pdf/jansson2017singing.pdf

18. Mimilakis S.I., Drossosy K., Santos J.F., Schuller G., Virtaneny T., Bengio Y., (2018, April). Monaural Singing Voice Separation with Skip-Filtering Connections and Recurrent Inference of Time-Frequency Mask. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* https://arxiv.org/abs/1711.01437

19. Noh H., Hong S., Han B. (2015). Learning deconvolution network for semantic segmentation. *IEEE International Conference on Computer Vision,* pp. 1520-1528.

20. Pinon, R.M. (2013). *Audio Source Separation for Music in Low-latency and High-latency Scenarios.* https://www.tdx.cat/bitstream/handle/10803/123808/trmp.pdf

21. Roma G., Green O., Tremblay P.A. (2018). Improving single-network singlechannel separation of musical audio with convolutional layers. *Latent Variable Analysis and Signal Separation Lecture Notes in Computer Science,* pp. 306 — 315.

22. Ronneberger O., Fischer P., Brox T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation.* https://arxiv.org/pdf/1505.04597.pdf

23. Schobben D., Torkkola K. (1999). Evaluation of blind signal separation methods. *Eindhoven University of Technology,* pp. 1 — 7.

24. Srivastava R.K., Greff K., Schmidhuber J. (2015). *Training Very Deep Networks.* https://arxiv.org/pdf/1507.06228.pdf

25. Stoller D., Ewert S., Dixon S. (2018). *Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation.* https://arxiv.org/abs/1806.03185

26. Takahashi N., Mitsufuji Y. (2017, November).Multi-scale multi-band DenseNets for audio source separation. *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA).* https://www.researchgate.net/publication/321792723_Multi-Scale_multi-band_densenets_for_audio_source_separation

27. Uhlich S., Porcu M., Giron F., Enenkl M., Kemp T., Takahashi N., Mitsufuji Y. (2017). Improving music source separation based on deep neural networks through data augmentation and network blending. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* https://www.researchgate.net/profile/Yuki_Mitsufuji/publication/315100151_Improving_music_source_separation_based_on_deep_neural_networks_through_data_augmentation_and_network_blending/links/59ed4f844585151983ccdcba/Improving-music-source-separation-based-on-deep-neural-networks-through-data-augmentation-and-network-blending.pdf

28. Vincent E., Gribonval R., Plumbley M. (2007). Oracle estimators for the benchmarking of source separation algorithms. *Signal Processing,* pp. 1933 — 1950.

29. Vincent E., Gribonval R., Fevotte C. (2005). *Performance measurement in blind audio source separation.* https://www.researchgate.net/publication/3457620_Performance_measurement_in_blind_audio_source_separation

30. Vincent E., Fevotte C., Gribonval R., Benaroya L., Rodet X., Robel

A., Carpentier E.L., Bimbot F. (2003). A tentative typology of audio source separation tasks. *Independent Component Analysis and Blind Signal Separation (ICA),* pp. 715-720.

31. Ward D., Takahashi Q.K.N., Goswami N., Mitsufuji Y. (2018). *MMDeseLSTM: an efficient combination of convolutional and recurrent neural networks for audio source separation.* https://arxiv.org/pdf/1805.02410.pdf
32. *Простими словами про перетворення Фур'є.* (n. d.). Retrieved October 4, 2013, from https://liabr.com/ru/post/196374/
33. *Сегментація зображень за допомогою нейронної мережі U-net.* (n. d.). Retrieved July 7, 2017, from http://robocraft.ru.blog.machinelearning.3671.html
34. НТУ «Харківський політехнічний інститут», „Проблеми інформатики та моделювання," Двадцята міжнародна науково-технічна конференція, Кароліно-Бугаз, 2020, с. 43.
35. ОНУ імені І. І. Мечникова, 77-ма звітна студентська наукова конференція Одеського національного університету імені І. І. Мечникова, Одеса, 2021, с. 104.
36. *Audio AI: isolating vocals from stereo music using Convolutional Neural Networks.* (n. d.). Retrieved October 21, 2019, from https://towardsdatascience.com/audio-ai-isolating-vocals-from-stereo-music-using-convolutional-neural-networks-210532383785
37. *Conda documentation.* (n. d.). Retrieved November 1, 2020, from https://docs.conda.io/en/latest/
38. *Google Trends.* (n. d.). Retrieved November 1, 2020, from https:,trends.google.comtrendsexplore?date=today%205-y&q=P%20Machine%20Leaming:R%20Machine%20Leaming.Java%20Ma-chine%20Leaming;Scala%20Machine%20Leaming.Julia%20Machine%20Leaming
39. *How to Choose Loss Functions When Training Deep Learning Neural Networks.* (n. d.). Retrieved November 1, 2020, from https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-

neural-networks/

40. *LibROSA librosa 0.7.1 documentation.* (n. d.). Retrieved November 1, 2020, from https://librosa.org/doc/0.7.1/index.html
41. *LOUD: Large acOUstic Data Array Project.* (n. d.). Retrieved November 1, 2020, from http://groups.csail.mit.edu/cag/mic-array/
42. *MUSDB18.* (n. d.). Retrieved November 1, 2020, from https://sigsep.github.io/datasets/musdb.html
43. Bryan N., Sun D., Cho E. (2019). *Single-Channel Source Separation Tutorial Mini-Series,* from https://ccrma.stanford.edu/~njb/teaching/sstutorial/
44. *The 2018 Signal Separation Evaluation Campaign.* (n. d.). Retrieved October 10, 2019, from https://arxiv.org/abs/1804.06267
45. *TIOBE Index for January 2020.* (n. d.). Retrieved November 1, 2020, from https://www.tiobe.com/tiobe-index
46. Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan. (2010). *Collaborative Filtering Recommender Systems,* from http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf
47. Michael J. Pazzani, Daniel Billsus. (2007, January). *Content-Based Recommendation Systems* from https://www.researchgate.net/publication/280113634_Content-Based_Recommendation_Systems
48. *Методи збору оцінок.* Retrieved March 20, 2015, from https://yandex.ru/blog/company/92883
49. Dr. Roman V Belavkin. *Lecture 11: Feed-Forward Neural Networks.* (2014, July 22), from http://www.eis.mdx.ac.uk/staffpages/rvb/teaching/BIS3226/hand11.pdf
50. *Метод зворотного поширення помилки.* Retrieved March 12, 2011, from https://en.wikipedia.org/wiki/Backpropagation
51. Vincent Dumoulin, Francesco Visin. *A guide to convolution arithmetic for deep learning.* (2016, March 23), from https://arxiv.org/abs/1603.07285
52. *Функції активації.* Retrieved January 1, 2010, from https://en.wikipedia.org/wiki/Activation_function

53. *Функції втрат.* Retrieved November 11, 2014, from https://en.wikipedia.org/wiki/Loss_functions_for_classification

54. Leon Bottou. *Stochastic Gradient Descent Tricks.* (2015, October 2), from https://cilvr.cs.nyu.edu/diglib/lsml/bottou-sgd-tricks-2012.pdf

55. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* (2014, June), from https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

56. Алексей Лукин. *Введение в цифровую обработку сигналов.* (2019, June 5), from http://audio.rightmark.org/lukin/dspcourse/dspcourse.pdf

57. Douglas Lyon, „The discrete fourier transform, Part 2: Radix 2 FFT", Journal of object technology, vol. 5, pp. 21 — 23, 2009

58. S. Molau, M. Pitz, R. Schluter, H. Ney. „Computing Mel-frequency cepstral coefficients on the power spectrum," in Conf. 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, UT, USA, pp. 1 — 4.

59. *Утиліта FFMPEG.* (n. d.). Retrieved January 31, 2016, from http://ffmpeg.org/ffmpeg.html

60. *Кодеки аудіо фалів.* (n. d.). Retrieved June 24, 2015, from https://trac.ffmpeg.org/wiki/audio%20types

61. *Опис формату wav-файлу.* (n. d.). Retrieved March 22, 2012, from http://tiny.systems/software/soundProgrammer/WavFormatDocs.pdf

62. *Мел-кепстральних коефіцієнти.* (n. d.). Retrieved March 28, 2012, from https://habrahabr.ru/post/140828/

63. *Recommending music on Spotify with deep learning.* (n. d.). Retrieved Augest 05, 2014, from http://benanne.github.io/2014/08/05/spotify-cnns.html

64. Goldberg, Y. *word2vec Explained: Deriving Mikolov et al.'S Negative-Sampling Word-Embedding Method .* (2012, September 5), from https://arxiv.org/pdf/1402.3722.pdf

ДОДАТОК

У даній роботі на мові програмування Python 3.6 була реалізована згорткова нейронна мережа для сліпого поділу джерел звуку в монофонічному музичному аудіосигналі. Навчання мережі проводилося на сформованій корисувачем вибірці з треків, які раніше були наведені в дані роботі.

Щоб використовувати даний проект, потрібно встановити наступні бібліотеки та пакети:

- tensorflow (бібліотека)
- keras (бібліотека)
- librosa (бібліотека)
- h5py (бібліотека)
- numpy (бібліотека)
- pydot (пакет)
- graphviz (пакет)
- tensorboard (бібліотека)
- python3-tk (пакет)

Наведемо характеристику кожного з файлів проекту та код програм.

- analysis.py — запускає різні функціональні можливості аналізу;
- batch.py — різні генератори партій для навчання;
- checkpointer.py — контрольні точки для keras;
- chopper.py — різні функції нарізки для створення зразків;
- config.py — змінні середовища читання об'єкта конфігурації;
- console.py — клас для ведення журналу;
- conversion.py — утиліта для перетворення аудіофайлів у спектрограми та назад;
- data.py — дані для навчання;
- grid_search.py — виконує пошук сітки;
- loss.py — різні функції втрат для тренувань;
- metrics.py — різні метрики, що використовуються для навчання;
- modeler.py — різні моделі, які будуть використовуватися для навчання;
- normalizer.py — різні стратегії нормалізації для підготовки даних;
- optimizer.py — оптимізатори, які будуть використані для навчання;
- vocal_isolation.py — запускає проект.

Нижче наведено код основного файлу vocal_isolation.py, який відповідає за запуск усього проекту та розділення сигналу.

```
import random
import string
import os
import sys
import signal

import numpy as np
from keras.utils import plot_model

import console
import conversion

from data import Data, remove_track_boundaries
from config import config
from metrics import Metrics
from checkpointer import Checkpointer
from modeler import Modeler
from loss import Loss
from optimizer import Optimizer
from chopper import Chopper
from normalizer import Normalizer
from batch import Batch


class VocalIsolation:
    def __init__(self, config):
        self.config = config
        metrics = Metrics().get()
        m = Modeler().get()
        loss = Loss().get()
        optimizer = Optimizer().get()
        console.log("Model has", m.count_params(), "params")
        m.compile(loss=loss, optimizer=optimizer, metrics=metrics)
        m.summary(line_length=150)
        self.model = m
        # need to know so that we can avoid rounding errors with spectrogram
        # this should represent how much the input gets downscaled
        # in the middle of the network
        self.peakDownscaleFactor = 4

    def train(self, data, epochs, batch=8, start_epoch=0):
        x_train, y_train = data.train()
        x_valid, y_valid = data.valid()
        self.x_valid, self.y_valid = x_valid, y_valid
        checkpointer = Checkpointer(self)
        checkpoints = checkpointer.get()
        if self.config.batch_generator != "keras":
            batch_generator = Batch().get()
```

```
if self.config.epoch_steps:
    epoch_steps = self.config.epoch_steps
else:
    epoch_steps = remove_track_boundaries(x_train).shape[0]
epoch_steps = epoch_steps // batch
while epochs > 0:
    end_epoch = start_epoch + epochs
    console.log("Training for", epochs, "epochs on",
                epoch_steps * batch, "examples")
    console.log("Validate on", len(x_valid), "examples")
    if self.config.batch_generator == "keras":
        x_train = remove_track_boundaries(x_train)
        y_train = remove_track_boundaries(y_train)
        history = self.model.fit(
            x_train, y_train, batch_size=batch,
            initial_epoch=start_epoch, epochs=end_epoch,
            validation_data=(x_valid, y_valid),
            callbacks=checkpoints)
    else:
        history = self.model.fit_generator(
            batch_generator(x_train, y_train, batch_size=batch),
            initial_epoch=start_epoch, epochs=end_epoch,
            steps_per_epoch=epoch_steps,
            validation_data=(x_valid, y_valid),
            callbacks=checkpoints)
    console.notify(str(epochs) + " Epochs Complete!",
                   "Training on", data.in_path, "with size", batch)

    start_epoch += epochs
    if self.config.quit:
        break
    else:
        while True:
            try:
                epochs = int(
                    input("How many more epochs should we train for?"))
                break
            except ValueError:
                console.warn(
                    "Oops, number parse failed. Try again, I guess?")
        if epochs > 0:
            save = input("Should we save intermediate weights [y/n]? ")
            if not save.lower().startswith("n"):
                weight_path = ''.join(random.choice(string.digits)
                                      for _ in range(16)) + ".h5"
                os.path.join(os.path.dirname(config.weights),
                             weight_path)
                console.log("Saving intermediate weights to",
```

```
                                    weight_path)
                        self.save_weights(weight_path)
        return history

    def run(self, data):
        self.config.create_logdir()
        # save current environment for later usage
        last_env = os.path.join(self.config.logs, "env")
        config_str = str(self.config)
        with open(last_env, "w") as f:
            f.write(config_str)

        plot_model(self.model, show_shapes=True,
                   to_file=os.path.join(self.config.logs, 'model.png'))

        history = self.train(data, self.config.epochs,
                             self.config.batch, self.config.start_epoch)

        self.save_weights(self.config.weights)
        metrics_path = os.path.join(self.config.logs, "metrics")
        with open(metrics_path, "w") as f:
            metric_names = list(history.history.keys())
            for name in metric_names:
                f.write("%s %s\n" % (name,
                                     history.history[name][-1]))
        return history

    def save_weights(self, path):
        if not os.path.isabs(path):
            path = os.path.join(self.config.logs, path)
        self.model.save_weights(path, overwrite=True)

    def load_weights(self, path):
        if not os.path.isabs(path):
            path = os.path.join(self.config.logs, path)
        self.model.load_weights(path)

    def process_spectrogram(self, spectrogram, channels=1):
        chopper = Chopper()
        chopper.name = "infer"
        chopper.params = "{'scale': %d}" % self.config.inference_slice
        chop = chopper.get(both=False)

        slices = chop(spectrogram)

        normalizer = Normalizer()
        normalize = normalizer.get(both=False)
        denormalize = normalizer.get_reverse()
```

```
        new_spectrogram = np.zeros((spectrogram.shape[0], 0, channels))
        for slice in slices:
            # normalize
            slice, norm = normalize(slice)

            epanded_spectrogram = conversion.expand_to_grid(
                slice, self.peakDownscaleFactor, channels)
            epanded_spectrogram_with_batch_and_channels = \
                epanded_spectrogram[np.newaxis, :, :]

            predicted_spectrogram_with_batch_and_channels = self.model.predict(
                epanded_spectrogram_with_batch_and_channels)
            # o /// o
            predicted_spectrogram = \
                predicted_spectrogram_with_batch_and_channels[0, :, :, :]
            local_spectrogram = predicted_spectrogram[:slice.shape[0],
                                                      :slice.shape[1], :]

            # de-normalize
            local_spectrogram = denormalize(local_spectrogram, norm)

            new_spectrogram = np.concatenate(
                    (new_spectrogram, local_spectrogram), axis=1)
        console.log("Processed spectrogram")
        return spectrogram, new_spectrogram

    def infer(self, path, fft_window_size, phase_iterations=10,
              learn_phase=False, channels=1):
        console.log("Attempting to isolate vocals from", path)
        audio, sample_rate = conversion.load_audio_file(path)
        spectrogram = conversion.audio_file_to_spectrogram(
            audio, fft_window_size=fft_window_size,
            learn_phase=self.config.learn_phase)
        console.log("Retrieved spectrogram; processing...")

        info = self.process_spectrogram(spectrogram, channels)
        spectrogram, new_spectrogram = info

        console.log("reconverting to audio")

        # save original spectrogram as image
        path_parts = os.path.split(path)
        filename_parts = os.path.splitext(path_parts[1])

        conversion.save_spectrogram(spectrogram, os.path.join(
            path_parts[0], filename_parts[0]) + ".png")
```

```
        # save network output
        self.save_audio(new_spectrogram,
                        fft_window_size,
                        phase_iterations,
                        sample_rate,
                        path,
                        vocal=not self.config.instrumental,
                        learn_phase=learn_phase)

        # save difference
        self.save_audio(spectrogram - new_spectrogram,
                        fft_window_size,
                        phase_iterations,
                        sample_rate,
                        path,
                        vocal=self.config.instrumental,
                        learn_phase=learn_phase)

        console.log("Vocal isolation complete")

    def save_audio(self, spectrogram, fft_window_size,
                   phase_iterations, sample_rate,
                   path, vocal=True, learn_phase=False):
        part = "_vocal" if vocal else "_instrumental"
        new_audio = conversion.spectrogram_to_audio_file(
                spectrogram,
                fft_window_size=fft_window_size,
                phase_iterations=phase_iterations,
                learn_phase=learn_phase)
        path_parts = os.path.split(path)
        filename_parts = os.path.splitext(path_parts[1])
        output_filename_base = os.path.join(
            path_parts[0], filename_parts[0] + part)
        console.log("Converted to audio; writing to",
                    output_filename_base + ".wav")

        conversion.save_audio_file(
            new_audio, output_filename_base + ".wav", sample_rate)
        conversion.save_spectrogram(spectrogram, output_filename_base + ".png")


def get_signal_handler(vocal_isolation):
    def signal_handler(signal, frame):
        save = input("Should we save intermediate weights [y/n]? ")
        if not save.lower().startswith("n"):
            vocal_isolation.save_weights(vocal_isolation.config.weights)
        sys.exit(0)
    return signal_handler
```

```
if __name__ == "__main__":
    files = sys.argv[1:]
    config_str = str(config)
    print(config_str)

    vocal_isolation = VocalIsolation(config)

    if len(files) == 0 and config.data:
        console.log("No files provided; attempting to train on " +
                    config.data + "...")
        if config.batch_generator.startswith("random") \
                and config.epoch_steps == 0:
            console.error("EPOCH_STEPS is not set,"
                          " but cannot be determined from data.")
            exit(1)
        if config.load:
            console.h1("Loading Weights")
            vocal_isolation.load_weights(config.weights)
        console.h1("Loading Data")
        data = Data()
        console.h1("Training Model")
        signal.signal(signal.SIGINT, get_signal_handler(vocal_isolation))
        vocal_isolation.run(data)
    elif len(files) > 0:
        console.log("Weights provided; performing inference on " +
                    str(files) + "...")
        console.h1("Loading weights")
        vocal_isolation.load_weights(config.weights)
        for f in files:
            vocal_isolation.infer(f, config.fft,
                                  config.phase_iterations,
                                  config.learn_phase,
                                  config.get_channels())
    else:
        console.error(
            "Please provide data to train on (--data) or files to infer on")
```

Інші файли проекту можна переглянути, прейшовши за посиланням:
https://drive.google.com/drive/folders/11xjOt9TefqAea8xY2VegFqaNexS0we0q?usp=sharing.