Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розпізнавання мелодій з використанням нейромережевого підходу

Виконав:
студент 2 курсу, групи КІТм-21-1
Змійова Олена Валеріївна
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології
Керівник доц. Сердюк Н.М.

Допускається до захисту _____

Зав. кафедри _____ О.Г. Руденко

2022 р.

Харківський національний університет радіоелектроніки

Факультет	Комп’ютерної інженерії та управління
Кафедра	Комп’ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп’ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп’ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«_____»_____202_ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Змійовій Олені Валеріївні

1. Тема роботи (проекту) Розпізнавання мелодій з використанням нейромережевого підходу затверджена наказом університету від “7” листопада 2022р. № 1455 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 грудня 2022р.

3. Вихідні дані до роботи (проекту): Технічне завдання, шаблон дизайну застосування для задачі «Розпізнавання мелодій з використанням нейромережевого підходу»

4. Перелік питань, що потрібно опрацювати в роботі: аналіз структурних і функціональних особливостей об’єкта автоматизації; постановка задачі кваліфікаційної роботи; обґрунтування мети; вирішення поставленої задачі; аналіз інформаційних технологій, які використовуються при вирішенні задачі;математичний опис розроблюваної задачі; опис рішень з інформаційного забезпечення системи; розробка технічного забезпечення;вибір і розробка мовних засобів взаємодії користувачів системи; розробка програмногозабезпечення; аналіз структурних і функціональних особливостей об’єкта

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням кафедри

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд і аналіз сучасного стану розглянутої задачі	8.11.2022-11.11.2022	виконано
2	Змістовий опис та аналіз структурних і функціональних особливостей об'єкта дослідження	12.11.2022-14.11.2022	виконано
3	Постановка задачі кваліфікаційної роботи	15.11.2022-16.11.2022	виконано
4	Обґрунтування мети вирішення поставленої задачі	16.11.2022-17.11.2022	виконано
5	Змістовий опис і аналіз використаних інформаційних технологій	17.11.2022-18.11.2022	виконано
6	Обґрунтування вибору математичної схеми і розробка математичного опису розроблюваної задачі	18.11.2022-20.11.2022	виконано
7	Розробка й обґрунтування інформаційного забезпечення задачі	20.11.2022-21.11.2022	виконано
8	Розробка й обґрунтування технічного забезпечення задачі	21.11.2022-24.11.2022	виконано
9	Вибір і розробка мовних засобів взаємодії користувачів системи	24.11.2022-25.11.2022	виконано
10	Розробка програмного забезпечення	25.11.2022-28.11.2022	виконано
11	Тестування та оцінка надійності функціонування програмних і технічних рішень	28.11.2022-30.11.2022	виконано
12	Оформлення пояснювальної записки	30.11.2022-8.12.2022	виконано
13	Розробка презентації	10.12.2022-14.12.2022	виконано
14	Попередній захист	19.12.2022	виконано

Дата видачі завдання 8 листопада 2022 р.

Студент _____

Керівник роботи _____ доц. Сердюк Н. М.

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 55 сторінок, 11 рисунків, 1 додаток, 11 джерел.

Мета роботи – дослідження задачі автоматичної класифікації музичних творів за жанрами та створення прототипу програмної системи, що дозволяє виконувати автоматичну класифікацію з використанням згорткової нейронної мережі.

Предмет дослідження – автоматизація класифікації музичних творів з використанням нейромережевого підходу.

У роботі проведено огляд видів ознак, використовуваних для класифікації, а також огляд різних алгоритмів класифікації.

Для вирішення залач класифікації використано алгоритм згорткової нейронної мережі. програмної системи для автоматичної класифікації музичних творів за жанрами, описана цю систему, за допомогою засобів мови Python.

РОЗПІЗНАВАННЯ МУЗИКИ, ЦИФРОВІ АУДІОСИГНАЛИ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, ГЛИБИННІ НЕЙРОННІ МЕРЕЖІ.

ABSTRACT

Explanatory note of the qualification work: 55 pages, 11 figures, 1 appendix, 11 sources.

The purpose of the work is to investigate the task of automatic classification of musical works by genre and create a prototype of a software system that allows automatic classification using a convolutional neural network.

The subject of the research is the automation of the classification of musical tracks using a neural network approach.

This work provides an overview of the types of features used for classification, as well as an overview of various classification algorithms.

A convolutional neural network algorithm was used to solve classification problems. software system for automatic classification of musical works by genre, this system is described using Python language tools.

MUSIC RECOGNITION, DIGITAL AUDIO SIGNALS, CONVOLUTIONAL NEURAL NETWORKS, MACHINE LEARNING, DEEP NEURAL NETWORKS.

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Розпізнавання мелодій з використанням нейромережевого підходу

Виконав:
студент 2 курсу, групи КІТм-21-1
Змійова Олена Валеріївна
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології
Керівник доц. Сердюк Н.М.

2022 р.

АНОТАЦІЯ

Змійова О. В. Розпізнавання мелодій з використанням нейромережевого підходу. – Магістерська кваліфікаційна робота.

У магістерській кваліфікаційній роботі вирішено одне з актуальних питань — використання нейронних мереж у задачах класифікації наборів даних, а саме — музичних творів.

Предметом магістерської роботи є розпізнавання музики, цифрових аудіосигналів за допомогою згорткових нейронних мереж.

Метою даної роботи є дослідження задачі автоматичної класифікації музичних творів за жанрами та створення прототипу програмної системи, що дозволяє виконувати автоматичну класифікацію з використанням згорткової нейронної мережі.

Загальну схему рішення задачі класифікації музичних творів за жанрами можна умовно розділити на п'ять етапів:

1. Визначення набору ознак;
2. Виділення ознак з аудіозаписів;
3. Вибір алгоритму класифікації;
4. Вибір таксономії жанрів — ієрархічна множина категорій, відображуваного на музичну колекцію;
5. Класифікація за обраним алгоритмом.

Наявність надійних перевірочних даних також є ключовою вимогою для ефективного навчання класифікаторів музичних творів за жанрами. Так як між аннотаторами-людьми може бути досягнуто тільки обмежена згода, то це обмеження неминуче задає верхню межу ефективності автоматичної класифікації жанрів. Окремі люди можуть не тільки по-різному класифікувати окрему аудіозапис, а й мати уявлення про більше або меншому набір жанрів, з яких можна вибирати.

Дуже мала кількість жанрів має чітке визначення, а наявна інформація часто неоднозначна — деякі жанри значно перекриваються між собою, і окремі записи можуть по-різному одночасно належати до різних жанрів.

Дослідження в області автоматичної класифікації жанрів можуть допомогти в розумінні того, як люди створюють музичні жанри, які механізми вони використовують для класифікації музики, і яким чином усвідомлюються відмінності між жанрами. Техніки, що використовуються в автоматичній класифікації жанрів, можуть бути поширені й на інші види аналізу музики на основі тільки її змісту (клас задач музичного інформаційного пошуку), а також інших завдань класифікації, таких як вимір ступеня подібності. Такі системи можуть бути використані для класифікації музики, наприклад, на основі стилю виконання, географічного походження або історичного періоду; для пошуку нової музики, яка могла б сподобатися користувачеві, на основі вмісту ого колекції.

У роботі розглянуто основні аспекти тембру, частотного аналізу. Детально розглянуто перетворення мел-кепстральних частотних коефіцієнтів. Мел-частотні кепстральні коефіцієнти (MFCCs) широко використовуються в автоматичному розпізнаванні мови та динаміків. Звуковий сигнал постійно змінюється, тому припускається, що на коротких часових шкалах звуковий сигнал сильно не змінюється (статистично стаціонарні). Ось чому формується сигнал в 20-40 мс. Якщо кадр набагато коротший, тоді немає достатньо зразків, щоб отримати надійну спектральну оцінку, якщо вона довга, сигнал змінюється занадто сильно по всьому кадру.

Наступним кроком є розрахунок спектру потужності кожного кадру. Це мотивується вушною раковиною, яка вібрує в різних місцях залежно від частоти вхідних звуків. Залежно від місця розташування в раковині, різні нерви інформують мозок про наявність певних частот. Оцінка періодограми виконує подібну роботу до людей, визначаючи, які частоти присутні в кадрі.

Одним з існуючих рішень є Shazam який ідентифікує пісні за допомогою звукового відбитка на основі частотно-часового графіка — спектрограмою.

Він використовує вбудований мікрофон смартфона комп'ютера, щоб зібрати короткий фрагмент аудіо, що відтворюється. Shazam зберігає каталог аудіовідбитків у базі даних. Користувач позначає пісню тегами протягом 10 секунд, і програма створює аудіовідбиток. Shazam працює, аналізуючи записаний звук і шукаючи відповідність на основі акустичного відбитка в базі даних мільйонів пісень. Якщо він знаходить збіг, він надсилає користувачеві таку інформацію, як виконавець, назва пісні та альбом.

Роботу нейронної мережі можна описати наступним чином. На входи нейронів надходять сигнали, що в результаті обробляє суматор. На цьому етапі враховуються ваги, тобто значущість кожного із входів. На наступному кроці ці сигнали передаються на входи інших нейронів. Вага кожного з цих зв'язків може бути позитивною, або негативною. Позитивні зв'язки — збуджуючі, негативні — гальмівні. Саме завдяки ним визначається пам'ять та поведінка нейронної мережі. Таким чином, принцип дуже схожий на роботу мозку. Проте, щоб штучна нейронна мережа мала змогу виконувати складні операції та завдання, її треба навчити.

Ітоговий програмний продукт розроблено за допомогою мови програмування Python та програмного забезпечення JetBrains IntelliJ IDEA. Python пропонує широкі можливості для програмування нейромереж, зокерма, згорткових. Наступні бібліотеки та засоби необхідні для імпорту у програмний код: librosa library, glob, numpy, matplotlib, послідовна модель з Keras, шар щільної нейронної мережі.

Значення точності перевірки може бути достатньо неочікуваним після тестового набору. Ця ітогова точність залежить від даних тестування, які мережа ніколи раніше не розглядала. Тепер отримана точність буде близько 53%. Це здається відносно низьким, але треба брати до уваги, що це для 10 різних жанрів. Випадкове вгадування дасть лише 10% точності, тому отримані результати досить непогані.

Кінцевий продукт може бути вдосконалений у майбутньому — стати серверною частиною, при тому маючи графічний інтерфейс користувача (він може бути як і веб-сторінкою, так й мати десктопну або мобільну версію). З подальшим розвитком, мережу можливо буде навчити вгадувати конкретні музичні композиції, або підбирати композиції за заданим ритмом/настроєм/тощо, або сортувати їх при пошуку.

РОЗПІЗНАВАННЯ МУЗИКИ, ЦИФРОВІ АУДІОСИГНАЛИ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, ГЛИБИННІ НЕЙРОННІ МЕРЕЖІ.

ЗМІСТ

ВСТУП

Музичні жанри — це високорівневі описи, що були створені і що використовуються людьми для музичної категоризації. Вони виникають під час процесу складної взаємодії між різними культурами, виконацями і використовуються для того, аби описати подібності між музикантами або музичнними композиціями, а також для організації колекцій музики.

Для пошуку музики та задля отримання приблизного уявлення про те, чи подобається той чи інший твір ще перед його прослуховуванням, слухачі використовують жанри. У індустрії музики, жанри — це ключовий засіб для визначення цільової авдиторії. Наразі кількість музичних файлів, доступних в мережі Інтернет в цифровому вигляді, стрімко зростає — з'являються різні сервіси, які надають доступ до великого обсягу музичних творів за підпискою. Автоматичний аналіз музики може стати однією з послуг, за допомогою якої власники таких сервісів будуть залучати клієнтів. Така можливість може стати корисною і для адміністраторів мережевих архівів з інформацією про музику, що включає до себе також і її жанр. Подібні сервіси покладаються на мануальну класифікацію, а вона повільна і громіздка.

Мета роботи — дослідити питання щодо автоматичної класифікації музичних творів за жанрами, створити прототип програмної системи, яка дозволяє виконувати автоматичну класифікацію за допомогою згорткової нейронної мережі.

Для досягнення мети цієї роботи потрібно вирішити наступні моменти:

1. Огляд видів ознак, що використовуються для класифікації, і алгоритмів класифікації.

2. Обрати та описати спосіб вилучення значень ознак, визначити сам набір ознак і описати алгоритми, що використовуються для отримання ознак.

3. Описати алгоритм роботи згорткової нейронної мережі.

4. Обрати набір даних, на якому буде здійснюватися тестова автоматична класифікація, і провести тестування на ньому. Підібрати вхідні параметри, провести аналіз отриманих результатів.

5. Розробити програмну систему для класифікації музичних творів за жанрами. Реалізувати її за допомогою засобів мови Python.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Постановка задачі

Для вирішення задач класифікації треба розділити статистичні дані по певних класах. Найбільш поширеним способом подання даних є той, при якому зразок є величиною векторною. Компоненти цього вектору є характеристики, що впливають на прийняття рішення про клас, до якого відноситься даний зразок. Отже, необхідно визначити на підставі певної інформації про об’єкт, до якого класу його можна віднести. Таким способом класифікатор відносить об'єкт до одного з класів відповідно до визначеного розбиття N-мірного простору. Такий простір називається простором входів, а його розмірність відповідає кількості компонент вектору.

Задачу класифікації можна визначити таким чином: нехай X — це безліч описів об'єктів (прикладів), Y — це безліч номерів класів. Існує невідома цільова залежність — відображення $y^*: X \to Y$, та її значення відомі лише на об'єктах кінцевої навчальної вибірки $X_m = \{(x_1,\ y_1), \dots ,\ (x_m,\ y_m)\}$. Необхідно побудувати алгоритм $a: X \to Y$, який має здатність класифікувати довільний об'єкт $x\ \in X$.

Для автоматичної класифікації музичних творів за жанрами кожний приклад це є n-мірний вектор ознак, де ознака (дескриптор) — це порція інформації, яка може бути залучена з аудіозапису і потім використана для її опису або класифікації. У даній роботі під аудіозаписами будуть розумітися цифрові аудіо файли, які містять частину музичного твору чи його цілком.

Загальна схема вирішення задачі класифікації музичних творів за жанрами мже бути умовно розділена на такі етапи:

1. Визначення набору ознак;
2. Виділення ознак з аудіозаписів;
3. Вибір алгоритму класифікації;

4. Вибір таксономії жанрів — ієрархічної множинини категорій, відображуваної на музичну колекцію;
5. Класифікація за обраним алгоритмом.

Однією з головних вимог ефективного навчання класифікаторів музичних творів за жанрами є надійні перевірочні данні. Коли між аннотаторами-людьми може бути досягнуто тільки обмежена згода, то це обмеження задає верхню межу ефективності автоматичної класифікації. Різні люди можуть не тільки по-різному класифікувати один й той ж самий аудіозапис, а й також вони можуть мати зовсім різне уявлення про набір жанрів, з яких можна обирати. Достатньо мала кількість музичних жанрів має чітке визначення, а існуюча інформація частіше за все неоднозначна — бо деякі жанри перетинаються між собою, і окремі аудіозаписи можуть одночасно належати до різних жанрів. Між жанрами також часто є складні взаємовідносини, деякі жанри ширші, в той час як інші — вужчі.

Дослідження в галузі автоматичної класифікації можуть допомогти в розумінні процеса створенння музичних жанрів, які механізми вони використовують для класифікації музики, і як усвідомлюється різниця між жанрами. Механізми, які використовуються людьми для класифікації жанрів, слабо вивчені, і створення автоматичного класифікатора може надати цінну інформацію в цій області.

Техніки, які використовуються для автоматичної класифікації музичних жанрів, можуть також використатися й для інших видів аналізу музики на основі лише її змісту (музичний інформаційний пошук), й інших завдань класифікації, наприклад, вимір ступеня подібності. Такі системи можуть використовуватися для класифікації музики, наприклад, на основі стилю виконання, географічного походження або історичного періоду; а також задля пошуку нової музики, яка могла б сподобатися користувачеві, на основі вмісту його колекції, вподобаних пісень тощо.

Основне завдання аналізу звукових сигналів — це отримання ознак, які характеризують цей сигнал. У більшості випадків пропонуються три набори

ознак, що використовуються для подання висоти звуку, тембру, ритму. Один з наборів тембральних ознак збігається з ознаками для розпізнавання мови і звуків в цілому, проте два інших набора характеризують аспекти музики. Тембр — це така характеристика звуку, яка дозволяє звукам з однаковою висотою і гучністю, звучати по-різному. Ознаки тембру також часто називають низькорівневими, тому що зазвичай вони обчислюються на коротких відрізках звукового сигналу (від 10 до 60 мс).

Головна ідея — це використання функції, що описує розподіл висот на короткій ділянці композиції. Високорівневі ознаки, такі як тональність, головна частота, послідовність акордів тощо, не використовуються, замість цього на основі функції обчислюється набір значень ознак, таких як амплітуди і розташування основних піків, величини інтервалів між піками і будь-які інші статистичні описи розподілу функції висотного вмісту.

Ритм не має точного визначення. Більшість авторів розглядають ритм як певну регулярність. У загальному сенсі, слово «ритм» може використовуватися для визначення часових аспектів музичного твору. Інтуїтивно зрозуміло те, що ритм має також враховуватися при розрізненні жанрів, тому що ритмічне відображення часто є важливою особливістю музичного жанру. Такі системи можуть орієнтуватися на використання у різних застосунках: визначення темпу, відстеження ударів, визначення метра тощо. Сучасні системи для визначення ритму все ще мають певні недолікі, тому в системах автоматичної класифікації частіше за все використовується низькорівневий підхід. За використанням того ж підходу, що й для низькорівневих висотних ознак, можна витягнути ознаки за допомогою функції, що робить оцінк періодичності в діапазоні сприйманих темпів (від 40 до 200 ударів на хвилину).

Такі ознаки можуть вилучатися з усього звукового сигналу. Проте, у більшості задач класифікації музичних творів використовується невеликий звуковий сегмент, допускаючи, що він може мати необхідну кількість інформації для опису всього твору, тому що у більшості музичних жанрів спостерігаються повтори музичної структури. Також, при використанні

невеликої частини сигналу, можна помітно зменшити обчислювальні витрати. Часто використовується один звуковий сегмент на кожний твір — зазвичай це 30-секундний відрізок, який був вилучений через 30 секунд після початку композиції. Це необхідно для того, аби не розглядати різні вступи, що також можуть відрізнятися від твору. В даній роботі використовується класифікація за часовою шкалою.

Техніки вилучення високорівневих ознак з поліфонічних звукових игналів все ще мають недоліки. Саме тому, велика кількість підходів націлена на моделювання тембру на основі комбінацій ознак низького рівня. Тембр має достатню кількість інформації для грубого визначення музичних жанрів. Проводилося дослідження з учасниками, що мали невелику підготовку, або не мали її зовсім. Учасники були здатні виконати класифікацію музичних творів за жанрами з такими результатами — в 53% випадків після прослуховування всього лише 250 мс, та в 72% після прослуховування 3 секунд. Це дозволяє припустити, що для визначення жанрів, високорівневе розуміння музики не вимагається, бо, як 250 мс, так і 3 секунд замало для визначення музичної структури.

Проте, існує й більш песимістичний погляд на ситуацію. Використовуючи сучасний алгоритм визначення подібності тембру, було проведено дослідження на базі даних з 20000 творів в 18 жанрах. За результатами було отримано такий висновок — між тембром і жаном є лише невелика кореляція, тоді є припущення, що схеми класифікації, що засновані тільки на тембрі, — обмежені. Тоді, подібні схеми класифікації не масштабуються за кількістю творів та жанрів. Проте, було припущення, що звуковий сигнал має недостатньо інформації для опису жанру музичного твору, і тому необхідно брати до уваги культурні особливості, отримуючи релевантні ключові слова, пов'язані з музичними творами. Якщо хтось намагається визначити жанр лише на основі звукового запису, то він припускає, що жанр є таким ж внутрішнім атрибутом твору, як і його темп, що знаходиться під питанням.

1.2 Опис алгоритмів класифікації

Існують такі головні класи алгоритмів класифікації:

1. Експертні системи;
2. Алгоритми навчання без учителя;
3. Алгоритми навчання з учителем.

Експертні системи визначають набори ознак явно. Для завдання класифікації за жанрами це було б аналогічно перерахуванню набору правил, які б точно і однозначно могли описати жанр. За наявними даними, для музичних жанрів поки не було запропоновано моделей, заснованих на експертних системах. Такий підхід, якщо він взагалі можливий, не підходить для завдання класифікації за жанрами через складність як самого завдання, так і об'єктивного опису окремих піджанрів. Крім того, цей підхід передбачає отримання надійних високорівневих описів з аудіосигналу, а відповідні алгоритми є недосконалими. Експертні системи, хоч і містять глибокі знання про свою предметну область, є витратними в реалізації. Зі зростанням числа правил, які створюються мануально, можуть з’являтися непередбачувані побічні ефекти. Наразі все більший інтерес викликає підхід на основі машинного навчання (без учителя). Деякі підходи прагнуть до класифікації музики в заданій таксономії жанрів, інший погляд полягає в кластеризації даних неконтрольованим способом таким образом, що класифікація сама виникає з даних на основі об'єктивних заходів подібності. Перевага полягає в тому, що такий підхід дозволяє уникнути обмежень фіксованої таксономії, що призводять до неоднозначних результатів дослідження. Проте, деякі твори можуть не вписуватися в таксономію.

При машинному навчанні, звуковий запис відображається набором ознак, а для порівняння творів між собою використовується міра збіжності. Найпростішим способом вимірювання відстані між двома векторами є

евклідова відстань. Проте, такі метрики мають сенс, тільки коли вектори ознак не змінюються з часом. В таком випадку два суб'єктивно схожих твори можуть бути віддаленими один від одного за результатами вимірювань, коли схожі значення ознак за часом зрушені. Щоб побудувати стаціонарне уявлення часових рядів векторів ознак, спочатку необхідно побудувати статистичну модель розподілу ознак, потім використовується відстань для безпосереднього порівняння моделей.

Гаусові моделі і суміші гаусових моделей (GMMs) — типові такі моделі. Відстань Кульбака-Лейблера, або відносна ентропія, — це природній спосіб для оцінки віддаленості двох імовірнісних розподілі, однак для сумішей гаусовських моделей цей спосіб не підходить. Існують такі альтернативні заходи: семплінг (Sampling); EMD-відстань (Earth Mover distance); апроксимація асимптотичної правдоподібності (Asymptotic Likelihood Approximation). На відміну від більшості класичних задач розпізнавання образів, класифіковані дані можуть бути тимчасовими рядами. Тоді можна використовувати приховані марковські моделі (HMM) щоб описати відносини між ознаками з плином часу.

Для кластеризації використовуються наступні алгоритми:

1. Метод k-середніх.

Такий метод дозволяє розбити безліч векторів на K непересічних підмножин. Для цього методу є необхідність заздалегідь знати кількість кластерів.

2. Алгоритм агломеративного ієрархічної кластеризації.

Він починає з кількості елементів в базі даних кластерів-одинаків, і формує кластери за допомогою послідовного злиття.

3. Самоорганізовані карти.

Це нейромережі з машинним навчанням, які відображають багатовимірні вхідні дані на вихідні простору низької розмірності, при цьому зберігаючи топологічні відносини між елементами вхідних даних настільки, наскільки це можливо.

Навчання без вчителя підходить для реалізації таких систем, які визначають збіжність музики у цілому, не конкретно для завдання класифікації жанрів, тому що в цьому випадку одержані категорії можуть бути беззмістовними для користувача. Ці категорії можуть бути точнішими, чим жанри, що визначаються саме людьми, але система, яка використовує свій власний набір та визначення жанрових категорій, будуть мати не високу користь для користувачів, які хочуть використовувати знайомі і зрозумілі жанри.

Засоби навчання мережі з учителем вивчені детальніше і намагаються відобразити базу музичних творів на задану таксономию жанрів з використанням алгоритмів навчання без учителя. Спочатку система вчиться на даних, розмічених вручну, а потім використовується для класифікації нерозмичених даних. Під час навчання з учителем, на відміну від експертних систем, немає потреби у явному вигляді описувати музичні жанр — класифікатор самостійно спробує сформувати зв'язок між відповідними категоріями і ознаками навчальної множин.

Існують такі алгоритми навчання з учителем, що використовувалися в певних дослідженнях, що стосуються класифікації музики за жанрами:

1. Метод k найближчих сусідів.

Обирається k найближчих векторів з навчальної множини, і цільовим вектору привласнюється мітка класу з найбільшою кількістю входів з k сусідів. Це виконується для заданого вектора ознак з цільової безлічі (звідки необхідно розпізнати жанр);

2. Суміші гаусівських моделей.

Передбачається існування функції щільності ймовірності для кожного класу. Вона виражається у вигляді суміші набору багатовимірних нормальних розподілів. Використовується алгоритм ітеративної максимізації очікування для оцінки параметрів кожного компонента;

3. Приховані марківські моделі.

Двічі стохастичний процес, який складається з випадкових процесів: головного і прихованого. Часто використовуються для розпізнавання мови за допомогою обробки часових рядів даних;

4. Лінійний дискримінантний аналіз.

Обчислюється лінійне перетворення, що найкращим чином розділяє класи, і класифікація в трансформованому просторі на основі метрики, наприклад, евклідової відстані;

5. Метод опорних векторів.

Це набір схожих алгоритмів виду «навчання з учителем». Будуються паралельні гіперплощини по обидва боки гіперплощини, яка розділяє класи. Розділяючою гіперплощіною буде така, яка максимізує різницю у відстані між паралельними гіперплощинами. Алгоритм працює за таким припущенням — чим більше відстань або різниця між даними гіперплощинами, тим менша буде середня помилка класифікатора;

6. Штучна нейронна мережа (ШНМ).

ШНМ будується з великої кількості нейронів (обробних елементів), має здатність вирішувати певні завдання. ШНМ базується на поєднанні вузлів, які називаються штучними нейронами (за схожим принципом, що й нейрони у головному мозку). Кожне з'єднання (аналогічне до синапсів) між такими штучними нейронами дозволяє сигналу передаватися від одного нейрона до іншого. Той нейрон, який отримує сигнал, може його обробити, а потім сигналізувати іншим нейронам, суміжним з ним. У більшості ШНМ сигнал на з'єднанні між штучними нейронами є дійсним числом, а вихід кожного штучного нейрону обчислюється нелінійною функцією суми його входів. Штучні нейрони та з'єднання зазвичай мають вагу, яка підлаштовується в перебігу навчання. Вага збільшує або зменшує силу сигналу на з'єднанні. Штучні нейрони можуть мати такий поріг, що сигнал надсилається лише якщо сукупний сигнал перетинає цей поріг. Штучні нейрони зазвичай організовано в шари. Різні шари можуть виконувати різні види перетворень своїх входів.

Сигнали проходять від першого (вхідного) до останнього (вихідного) шару, можливо, після проходження шарами декілька разів.

На початку, мета підходу ШНМ — це було розв'язання задач таким же чином, як би це робив людський мозок. Згодом увага зконцентрувалася на відповідності до певних здібностей розуму, що призводило до відхилень від правил біології. ШНМ використовувалися для розв’язання певних задач, включаючи ті, що пов’язаі з комп'ютерним зором, машинним перекладом, розпізнаванням мовлення, соціально-мережевим фільтруванням, грою в настільні та відео- ігри, та медичним діагностуванням.

Для автоматизації процеса класифікації музичних творів застосовуються такі алгоритми, що базуються на деревах прийняття рішення — засобах подання правил в послідовній структурі, ієрархії, де кожному об'єкту відповідає єдиний вузол, що надає рішення. Задля класифікації тих об'єктів, які не увійшли в множину для навчання, відбувається пошук, що починається з коріння, і він триватиме до тих пір, доки не виявиться клас, що буде відповідний до об'єкта.

Для досягнення мети цієї роботи потрібно вирішити наступні моменти:

1. Огляд видів ознак, що використовуються для класифікації, і алгоритмів класифікації.

2. Обрати та описати спосіб вилучення значень ознак, визначити сам набір ознак і описати алгоритми, що використовуються для отримання ознак.

3. Описати алгоритм роботи згорткової нейронної мережі.

4. Обрати набір даних, на якому буде здійснюватися тестова автоматична класифікація, і провести тестування на ньому. Підібрати вхідні параметри, провести аналіз отриманих результатів.

5. Розробити програмну систему для класифікації музичних творів за жанрами. Реалізувати її за допомогою засобів мови Python.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

2.1 Shazam

Shazam — це програма, яка може ідентифікувати музику, фільми, рекламу та телевізійні шоу на основі короткого фрагменту, який відтворюється, та використовуючи мікрофон на пристрої. Його створила лондонська компанія Shazam Entertainment і належить Apple Inc. з 2018 року. Програмне забезпечення доступне для Android, macOS, iOS, Wear OS, watchOS і як розширення Google Chrome.

Shazam ідентифікує пісні за допомогою звукового відбитка на основі частотно-часового графіка, який називається спектрограмою. Він використовує вбудований мікрофон смартфона або комп'ютера, щоб зібрати короткий фрагмент аудіо, що відтворюється. Shazam зберігає каталог аудіовідбитків у базі даних. Користувач позначає пісню тегами протягом 10 секунд, і програма створює аудіовідбиток. Shazam працює, аналізуючи записаний звук і шукаючи відповідність на основі акустичного відбитка в базі даних мільйонів пісень. Якщо він знаходить збіг, він надсилає користувачеві таку інформацію, як виконавець, назва пісні та альбом. Деякі реалізації Shazam включають відповідні посилання на такі служби, як iTunes, Apple Music, Spotify, YouTube або Groove Music.

Shazam може ідентифікувати музику, яка відтворюється з будь-якого джерела, за умови, що рівень фонового шуму недостатньо високий, щоб запобігти зняттю акустичного відбитка, і що пісня присутня в базі даних програмного забезпечення.

Окрім безкоштовного додатку, компанія випустила платний додаток під назвою Shazam Encore. У вересні 2012 року послугу було розширено, щоб дозволити телевізійникам у США ідентифікувати пропоновану музику, отримувати доступ до інформації про акторів і отримувати посилання для

показу інформації в Інтернеті, а також додані можливості соціальних мереж. У 2014 році Shazam переробив свій додаток і додав додаткові функції.

2.2 Gracenote та MusicID

Це програмне забезпечення для розпізнавання музики, яке ідентифікує компакт-диски та передає метадані виконавця та обкладинки. База даних Gracenote містить відомості про музичний жанр і настрій, описи телешоу, інформацію про епізоди та список каналів, інформацію про акторський склад і знімальну групу, а також спортивну статистику та результати. Компанії, включаючи музичні сервіси, телепровайдери, виробники споживчої електроніки та автовиробники, використовують дані Gracenote для забезпечення свого контенту, універсального пошуку, навігації, посилань, пошуку та персоналізованих рекомендацій.

Технології розпізнавання музики Gracenote порівнюють цифрові музичні файли зі всесвітньою базою даних музичної інформації, що дозволяє цифровим аудіопристроям ідентифікувати пісні. Компанія надає ліцензії на свої технології розробникам споживчої електроніки та онлайнових медіаплеєрів, які інтегрують технології в медіаплеєри, домашні та автомобільні стереосистеми та цифрові музичні пристрої.

Gracenote надає підприємствам програмне забезпечення та метадані, що дозволяє їхнім клієнтам керувати цифровими медіафайлами та здійснювати пошук у них. Gracenote надає свою технологію керування медіа та глобальну медіа базу даних цифрової розважальної інформації для ринків мобільних, автомобільних, портативних, домашніх і ПК. Кілька програмних застосунків, які могли відтворювати компакт-диски (наприклад, Media Go та iTunes), використовували технологію CDDB Gracenote.

Рання лінійка продуктів Gracenote складалася з MusicID, Mobile MusicID, Music Enrichment, Discover, Playlist, Playlist Plus, Media VOCS, Classical Music

Initiative та Link. У квітні 2007 року Gracenote запустив першу легальну пропозицію текстів пісень у США, які були продані LyricFind у 2013 році.

Поточні музичні пропозиції Gracenote поділяються на три основні категорії: розпізнавання музики, музичні дані та відкриття музики. Продукт Gracenote розпізнавання музики під назвою MusicID спочатку був розроблений як система ідентифікації треків компакт-дисків. Gracenote також керує службою ідентифікації цифрових файлів, яка використовує технологію аудіо відбитків для ідентифікації цифрових музичних файлів, таких як MP3, і надання метаданих на рівні треків, обкладинок альбомів і посилань на додатковий вміст і служби. It's Music Data надає інформацію про жанр, настрій, епоху, походження та темп для десятків мільйонів пісень.

3 МАТЕМАТИЧНІ ОСНОВИ РОБОТИ

3.1 Використання нейронних мереж в задачах класифікації

Мозку людини неможливо швидко виконувати велику кількість математичних операцій водночас. Навіть найпростіші автоматизовані системи, такі як калькулятор, є більш ефективними, аніж людина. Але, людина швидко проходить адаптаці, на відміну від комп’ютерів, розуміє та аналізує зміни у навколишньому середовищі та нові умови. Людина може виділити знайоме обличчя в натовпі, сказати, що за дерево бачить перед собою, чи розуміти мову свого свпіврозмовника навіть в гучному приміщенні. Машини такі завдання виконують зі складнощами.

Людський мозок вміє аналізувати, мислити, виконувати складні операції за допомогою нейронів. Нейронами виконується функція передачі хімічних сигналів, також використовуючи електричні імпульси, один одному. Штучний інтелект та когнітивне моделювання базуються на роботі людського мозку. Вони імітують властивості «біологічних» нейромереж. Штучна нейромережа — це система, що поєднує штучні нейрони (прості процесори), які взаємодіють між собою. Ці нейрони у своїй більшості достатньо прості, якщо їх порівнювати з персональним комп’ютером. Кожна частина працює лише із сигналами, що до неї надходять, і що передає іншим частинам мережі, та така єдина мережа здатна виконувати складні завдання.

Роботу нейронної мережі можна описати наступним чином. На входи нейронів приходять сигнали, що надалі отримує та обробляє суматор. Враховуються ваги, тобто значущість кожного із входів. Подалі ці сигнали передаються на входи інших нейронів. Кожний з цих зв’язків може мати позитивну, або негативну вагу. Позитивні зв’язки — збуджуючі, негативні — гальмівні. За допомогою них визначається пам’ять та поведінка нейромережі.

Таким чином, принцип дуже схожий на роботу мозку. Проте, щоб штучна нейромережа змогла виконувати складні операції та завдання, її треба навчити.

3.1.1 Навчання штучної нейронної мережі

Штучні нейромережі навчаються, а не програмуються у звичайному розумінні цього слова. Завдяки натхненню від справжніх процесів у дюдському мозку, були створені штучні нейронні мережі (ANN). Не дивлячись на те, що подівбгі аналогії досить вільні, ШНМ мають мають багато відмінностей зі своїм біологічним «батьком». ШНМ складаються з певної кількості нейронів.

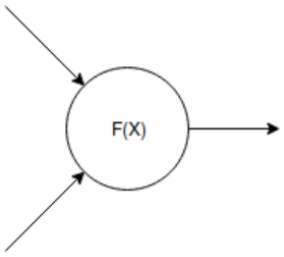

Рисунок 3.1 — Модель нейрона

На рисунку 3.1 представлено нейрон, що приймає два числа як вхідні дані. Кожний вхідний номер є таку позначку, як x_k, де k означає індекс вхідних даних. Для кожного входу x_k нейрон призначає інше число w_k. Вектор, що складається з цих чисел w_k, називається вектором ваг. Ці ваги роблять кожен нейрон унікальним, вони фіксуються під час тестування, але під час тренування це значення, що будуть змінюватися, щоб налаштувати мережу. Маємо лінійну комбінацію ваг і входів з певною нелінійною функцією над нею:

$$f(x_1, x_2) = w_1 x_1 + w_2 x_2, \tag{3.1}$$

де w_1, w_2 — вагові коефіцієнти;

x_1, x_2 — входи в нейромережу.

Наведена вище формула — це лінійна комбінація. Вхідні дані будуть помножені на відповідні ваги і підсумовані всі разом. Остання частина полягає

в тому, щоб на неї застосувати нелінійну функцію. Найбільш популярна нелінійність, яка використовується сьогодні — Rectified Linear Unit (ReLU), що обчислюється за формулою:

$$ReLU(x) = max(x, 0), \tag{3.2}$$

де x — це вхідне значення;

$max(x, 0)$ — функція активації.

Якщо номер перевищує нуль, тоді він залишається, а якщо він менше нуля, тоді замість цього використовується нуль.

Фнкція активації — це нелінійна функція, застосована на верхньому лінійному в нейроні. Підсумовуючи, маємо, що нейрон — це певна функція, що приймає певне фіксоване число входів і виводить одне число — його активацію. Тоді ітогова формула для нейрона буде мати вигляд:

$$f(x_1, x_2) = max(0, w_1x_1 + w_2x_2), \tag{3.3}$$

де w_1, w_2 — вагові коефіцієнти;

x_1, x_2 — входи в нейромережу;

$max(x, 0)$ — функція активації.

Нейронна мережа також є математичною функцією. Вона визначається зв'язкою нейронів, пов'язаних один з одним.

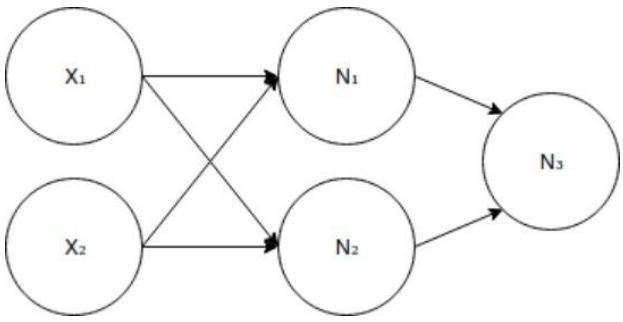

Рисунок 3.2 — Мережа з 5 нейронів

Необхідно взяти до уваги, що вихід з одного нейрона може використовуватися як вхід до інших, як на рисунку 2.2. Як можна побачити, ці

нейрони укладаються в 3 повністю пов'язаних шара, тобто кожний нейрон з одного шару з'єднаний з кожним нейроном з наступного шару. Кількість шарів в мережі та кількість нейронів в кожному шарі і принцип, за яким вони пов'язані — усе це визначає архітектуру мережі. Вхідний шар — це перший шар, який має 2 нейрона. Нейрони у цьому шарі не виконують ніяких обчислень. Вони визначають вхід в мережу. Коли в кожному нейроні не використається нелінійна функція, нейромережа буде лінійною функцією, тобто, не більш потужною, ніж єдиний нейрон.

Слід зазначити, що число, яке виходить з нейронної мережі, розглядається як ймовірність в діапазоні від 0 до 1.

Функція втрати — це функція, яка свідчить про те, наскільки добре виконує роботу нейромережа для деякого завдання. Взяти кожний приклад для навчання, пропустити його через мережу, аби отримати номер, відняти його від фактичного числа, яке бажано отримати, і порівняти його, — інтуїтивно зрозумілий спосіб зробити це:

$$L(y, \hat{y}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y})^2 \ , \qquad (3.4)$$

де y — це число, що бажано отримати від мережі;

ŷ — число, що було насправді отримано, пропустивши приклад через мережу.

На початку роботи мережі ваги випадково ініціалізовані. Очевидно, це не дасть дуже хороших результатів. У процесі навчання бажано почати з не дуже добре працюючої системи, і закінчити на мрежі з більш високою точністю. Дивлячись з боку функції втрати, необхідно, аби функція втрати була поміно нижчою наприкінці навчання. Удосконалення роботи нейромережі можливе, оскільки можливо змінити її функцію, регулюючи ваги. Ціль — знайти іншу функцію, що буде працювати краще, аніж перша.

Проблема навчання мережмає кореляцію з проблемою мінімізації функції втрат. Ціль мінімізації втрати замість максимізації полягає у наступному — набагато простіше для оптимізації функції буде саме втрата.

Існує багато алгоритмів, що домагають оптимізувати функції. Подібні алгоритми можуть полягатися на градієнті — вони застосовують не не лише інформацію, яка надається функцією, але і градієнтом функції:

$$\nabla L \approx \partial \frac{L}{\partial x_i} \nabla x_i \ , \tag{3.5}$$

де L — значення витрат в мережі;

x_i — вхід у мережу;

∇x_i — градієнт від x_i.

Це означає наступне — те, наскільки велика змінна функції, приблизно дорівнює похідній цієї функції відносно певної змінної x, помноженої на значення того, наскільки вона була змінена. Це наближення буде точним, коли буде зроблено нескінченно малий крок, і це є важливою концепцією похідної. Тоді маємо функцію f (x) = x², а її похідна дорівнює 2x. Легко перевірити, що зараз, починаючи з деякого значення x, і зробивши деякий крок epsilon, то наскільки змінюється функція не буде точно рівною формулі 3.5.

Тоді градієнт буде вектором часткових похідних, його елементи будутьь містити похідні відносно деякої змінної, від якої залежить функція. У випадку з простими функціями, цей вектор містить лише один елемент, оскільки використовується лише функція, яка приймає один вхід. З більш складними функціями, градієнт буде містити похідні по кожній змінній.

Як використовується інформація, яка надається похідними, щоб мінімізувати якусь функцію? Розглядаючи функцію f(x) = x², мінімум наданої функції знаходиться в точці, де x = 0. Маємо припущення, що початкове випадкове значення x дорівнює 2. Похідна функції в тому, що в x = 2 дорівнює 4. Маємо, що з кожним кроком у позитивному напрямку, функція буде змінюватися пропорційно до 4. Замість цього функція мінімізується, тому можливо зробити крок у протилежному напрямку, негативний, щоб бути впевненим, що функція зменшиться.

Наскільки великий може бути крок? Похідна буде гарантувати, що функція зменшиться, коли буде зроблено нескінченно малий крок, що неможливо зробити. Як правило, мають гіпер-параметр — швидкість навчання. Якщо початкова точка буде x = -2, то похідна тоді має значення -4, це позначає те, що якщо зробити невеликий крок у позитивний напрямок, функція буде змінюватися пропорційно до -4, таким чином, вона зменшиться. Коли x > 0, похідна більше нуля і необхідно йти в негативному напрямку. У випадку, якщо x < 0, й похідна менше нуля, необхідно йти в позитивному напрямку. Завжди необхідно робити крок у напрямку, протилежному похідній, таких підхід застосовується до градієнта.

Градієнт — це вектор, що вказує на певний напрямок у просторі. Це буде вказувати на напрям найшвидшого збільшення функції. Оскільки необхідно мінімізувати функцію, тоді робиться крок у протилежному напрямку градієнта. У нейронній мережі входи позначаються як x, а y — фіксовані числа. Змінна, щодо якої беруться похідні, є вагою w, оскільки ці цінності необхідно змінити, щоб покращити мережу.

Якщо обчислити градієнт функції втрат і зробити невеликі кроки в протилежному напрямку градієнта, то втрата буде поступово зменшуватись, до тих пір, доки вона не зійдеться до певних локальних мінімумів. Такий алгоритм має назву — спуск градієнта. Правило для оновлення ваг під час кожної ітерації спуску градієнта має наступний вигляд:

$$w_j = w_j - lr\partial \frac{L}{\partial w_j}, \tag{3.6}$$

де lr — швидкість навчанння;

L — значення витрат в мережі;

w_j — ваговий коефіцієнт.

Тут можна контролювати, наскільки великим кроком виконується кожна ітерація. Цей крок — це є найважливіший гіпер-параметр, щоб налаштувати нейронну мережу для навчання. Якщо вибрати надто велике навчання, кроки будуть занадто великими, і “перестрибнуть” мінімум. Якщо обрати швидкість

навчання, що має занадто мале значення, тоді може витратитися багато часу для наближення допенвих місцевих мінімумів. Люди розробили методиди для пошуку оптимальної швидкості навчання.

Після обчислень маємо, що сума похідних дорівнює похідної суми. Тому для обчислення похідної втрат, потрібно пройти через кожен приклад набору даних. Неефективною роботою було б виконання кожної ітерації градієнтного спуску, тому що кожна ітерація алгоритму тільки збільшує витрати на певному кроці. Задля розв’язання такої проблеми є такий алгоритм, що має назву міні-пакетний спуск градієнта. Міні-пакетний градієнтний спуск — це один з різновидів алгоритму градієнтного спуску, що розбиває тестувальний набір даних на невеликі партії, що застосовуються для підрахунку помилки моделі і оновленння її коефіцієнтів.

Імплементації можуть обирати підсумовування градієнта по міні-пакету або використовувати середнє значення градієнта, а це помітно зменшить дисперсію градієнта.

Алгоритм міні-градієнтного спуску прагне досягти балансу між надійністю стохастичного градієнтного спуску і ефективністю пакетного градієнтного спуску. Це є найбільш поширена реалізація градієнтного спуску, що застосовується в галузі глибокого навчання.

3.2 Згорткові нейронні мережі

Згорткова нейронна мережа — це клас штучних нейронних мереж, що фільтрує вхідні данні для отримання корисної інформації, застосовуючи згорткові шари. Операція згортки включає в себе об'єднання вхідних даних (карта об'єктів) з ядром згортки (фільтр) для формування перетвореної карти об'єктів. Фільтри в шарах змінюються на основі вивчених параметрів, для витягу більш корисної інформації для певного завдання. Такі мережі налагоджуються автоматично, аби знайти більш кращу функцію від залежності від завдання. Якщо є однотипні за формою об’єкти, тоді система

розпізнаватиме їх за кольором, а не за формою, тому що варіативність кольору в цьому випадку помітно вища, аніж форми.

Використання згорткових нейромереж включаються у різноманітні системи для розпізнавання зображень і мови — класифікація і обробка зображень, аналіз тексту і мови, розпізнавання, обробка природної мови, класифікація текст. Згортокі нейромережі також використовують у сучасних системи штучного інтелекту — роботах, віртуальних помічниках і автомобілях з автоматичним управлінням.

3.2.1 Компоненти згорткової нейронної мережі

Згорткові нейромережі складаються з таких шарів — вхідного, вихідного і одного або декількох прихованих. Згорткова нейронна мережа відрізняється від звичайної нейромережі тим, що нейрони в її шарах розташовані в трьох вимірах — ширина, висота і глибина. Таке розташування дозволяє CNN перетворювати вхідний обсяг в трьох вимірах у вихідні дані. Приховані шари поєднують у собі шари згортки, шари об'єднання, нормалізації і повністю пов'язаних шарів. CNN використовують кілька шарів повідомлень для фільтрації вхідних обсягів до більш високих рівнів абстракції.

На сьогодняшній день CNN використовують початкові модулі, що застосовують 1×1 згорткові ядра, з метою зменшення споживання пам'яті. Одночасно забезпечуючи більш ефективні обчислення (а як ітог і навчання). Тоді CNN будуть придатними для певних застосунків навчання без учителя.

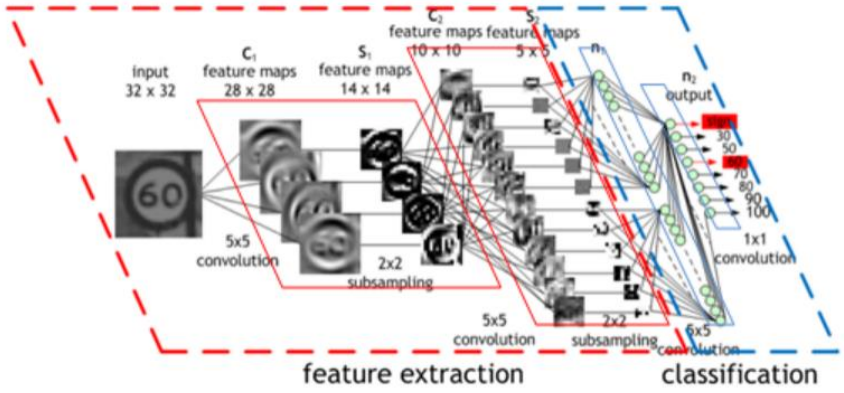

Рисунок 3.3 — Модель згорткової нейронної мережі

3.2.2 Функція активації

Функція активації в нейромережі має у собі нелінійне перетворення до зважених вхідних даних. Більш відомою функцією активації для CNN є ReLu — пряма лінійна функція, що зануляє негативні входи. Початкові модулі CNN можуть прискорити обчислення. Це робиться з використанням згорток 1×1 з невеликим розміром мапи об'єктів, наприклад, 192 мапи об'єктів розміром 28×28 можна зменшити до 64 карт об'єктів 28×28 за допомогою 64 згорток 1×1. Маючи зменшений розмір, такі згортки 1×1 можуть супроводжуватися більшими за розміром згортками 3×3 і 5×5. Додаючи до пакунку 1×1, максимальне об'єднання може також бути використане для зменшення розмірності згортки. У вихідних даних початкового модуля всі великі згортки об'єднуються у велику мапу об'єктів, що далі відправляється на наступний рівень, або на початковий.

Пулінг — це процес зменщення вхідних даних на певній області до одного значення. Ця процедура забезпечує базову інваріантність поворотів і перекладів, таким чином покращує можливість виявлення об'єктів у згорткових мережах. Наприклад, фігура на зображенні, що знаходиться не у його центрі, але трішки збоку, все ще може бути виявлена за допомогою згорткових фільтрів. Інформація про фігуру переноситься в потрібне місце за допомогою

процесу пулінга. Чим більший розмір області об'єднання, тим більше інформації ущільнюється, а це призводить до невеликих мереж, що підходять до пам'яті GPU. Але, у віпадку коли область об'єднання об'єктів завелика, тоді забагато інформації ігнорується, і зменшується прогнозована продуктивність.

Згорткові нейромережі також можуть бути підготовлені за допомогою методів оптимізації на основі градієнта, так саме, як і багатошарові персептрони або рекурентні нейронні мережі. Оптимізація параметрів нейромережі може бути досягнута за допомогою використаня алгоритмів стохастичного, пакетног, чи міні-серійного градієнтного спуску. Наприкінці проходження навчання CNN, вона може використовуватися для точного прогнозування виходів для необхідного входу.

3.3 Мел-частотні кепстральні коефіцієнти

За допомогою мел-кепстральних коефіцієнтів (MFCC), утворилася можливість відійти від несуттєвих компонентів для розпізнавання сигналу, значно знижуючи розмірність ознак.

Мел-частотні кепстральні коефіцієнти поширені у використанні для автоматичного розпізнаннян мови та динаміків. MFCC були запатентовані у 1980-х роках Девісом Мермелштейном, і, починаючи з тих часів, були найсучаснішими.

Аудіосигнал завжди змінюється, тому є припущення, що на коротких часових шкалах аудіосигнал помітно не буде змінюватися (статистично стаціонарні). Ось чому формується сигнал в 20-40 мс. У випадку, коли кадр набагато коротший, тоді немає достатньо зразків, щоб отримати надійну спектральну оцінку, якщо вона довга, сигнал змінюється дуже помітно упродовж всього кадру.

Після цього розраховується спектр потужності для кожного кадру. Мотивація у цього наступна — вушна раковина вібрує в різних місцях в залежності від частоти вхідних звуків. В залежності від місця розташування в

раковині, різні нерви надають інформацію мозку про наявність певних частот. Оцінка періодограми виконує подібну роботу до людей, визначаючи, які саме частоти присутні в кадрі.

Спектральна оцінка періодограми досі має у собі багато інформації, що не потрібна для автоматичного розпізнання мовлення ASR. Також, неможливо почути різницю між двомаблизькими за розташуванням частотами. Такий ефект має більшу вираженіть при збільшенні частот. Це є причини, за якими необхідно брати згустки періодограмних блоків і підсумувати їх, аби мати уявлення про те, яка кількість енергії існує в різних частотних областях. Це можна виконати за допомогою мел-фільтра — перший фільтр дуже вузький і вказує на кількість енергії, яка існує біля 0 Гц. Оскільки частоти стають вищими, відповідно, фільтри — ширшими. Шкала Mel дозволяє розмістити фільтрові блоки і зрозуміти, наскільки широкими вони будуть.

Блоки енергії фільтрів логарфимуються. Це мотивується людським слухом: людина не чує гучності в лінійному масштабі. Взагалі, щоб подвоїти обсяг звуку, необхідно вкласти в нього в 8 разів більше енергії.

Залишається лише обчислити DCT енергій блоків. Є дві основні причини — оскільки банки фільтрів перетинаютсья, енергії блоку фільтрів повністю корелють між собою. DCT декорелює енергії, тобто, матриці діагональної коваріації можуть використатися для моделювання ознак, таких як, HMM класифікатор.

Це називається періодограмною оцінкою потужності спектра. Береться абсолютне значення комплексного перетворення Фур'є і квадратикується результат. Зазвичай, виконується FFT на 512 пунктів і зберігаються тільки перші 256 коефіцієнтів.

Mal-spaced filermank — це набір з 20-40 трикутних фільтрів (26 стандартних), що застосовуються до спектральної оцінки енергії періодограми. Фільтр надходить у вигем для певної ділянки спектру. Для обчислення енергії блоку множиться кожен блок на потужність, а потім додаються коефіцієнти. Як тільки це буде зроблено, залишиться 26 номерів, які в коляді 26 векторів

довжиною 256. Кожний вектор в основному нульовий, але не вказують на кількість енергії, яка була в кожному блоці (див. приклад на рисунку 3.3).

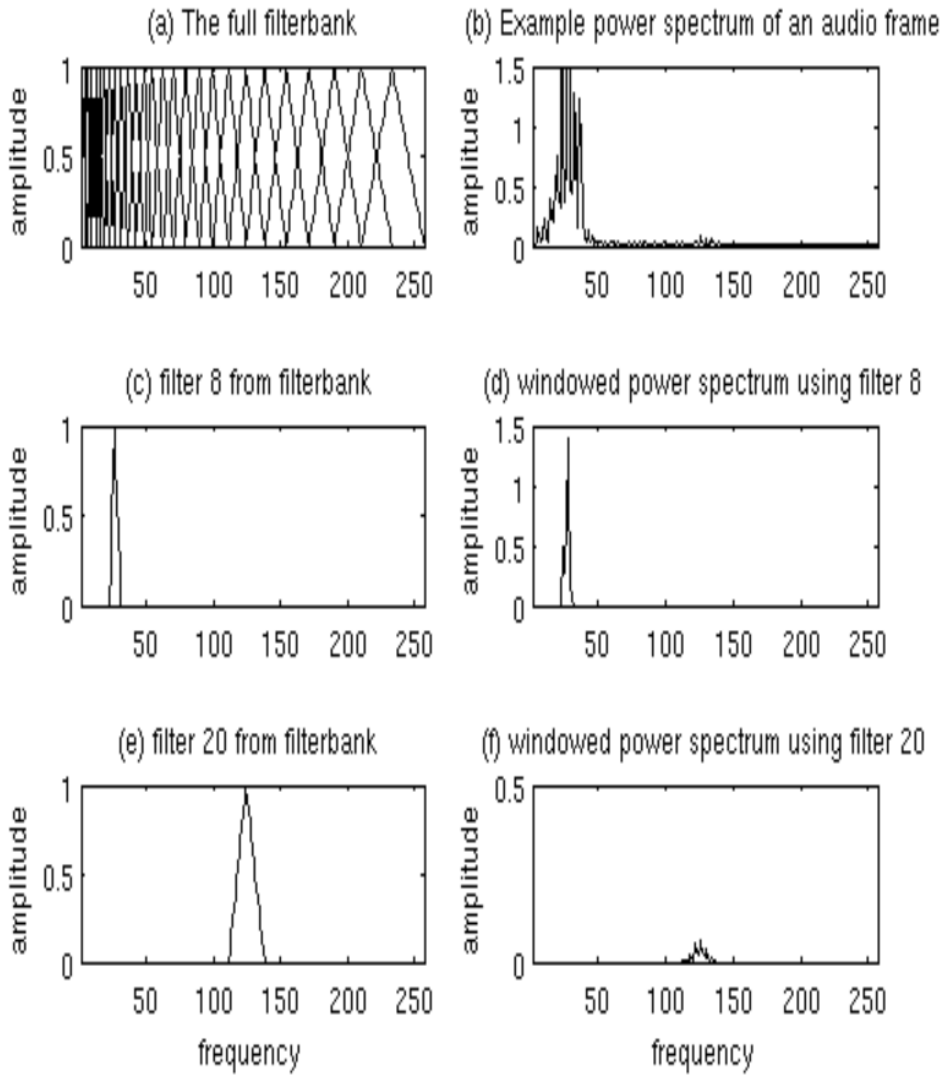

Рисунок 3.3 — MFCC

Візьмемо log кожної з 26 енергій, потім DCT з енергій 26 фільтрів, аби надати 26 кепстральних коефіцієнтів. Для ASR зберігаються лише нижні 12-13 з 26 коефіцієнтів. 12 номерів для кожного кадру й будуть мел-частотними кепстральними коефіцієнтами.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Огляд засобів розробки

4.1.1 Мова програмування Python

Python — це мова комп’ютерного програмування, яка часто використовується для створення веб-сайтів і програмного забезпечення, автоматизації завдань і аналізу даних. Python є мовою загального призначення, тобто її можна використовувати для створення різноманітних програм і не спеціалізуватися на конкретних проблемах. Ця універсальність, а також зручність для початківців зробили її однією з найбільш використовуваних мов програмування на сьогодні.

Python зазвичай використовується для розробки веб-сайтів і програмного забезпечення, автоматизації завдань, аналізу та візуалізації даних. Оскільки його відносно легко вивчити, Python був прийнятий багатьма непрограмістами, такими як бухгалтери та вчені, для різноманітних повсякденних завдань, як-от організація фінансів.

Python може використоуватися для таких завдань:

1. Веб- та Інтернет-розробка (наприклад, фреймворки Django та Pyramid, мікрофреймворки Flask та Bottle);
2. Наукові та числові обчислення (наприклад, SciPy – колекція пакетів для цілей математики, науки та інженерії; Ipython – інтерактивна оболонка, яка підтримує редагування та запис робочих сеансів);
3. Освіта (навчання програмуванню);
4. Графічні інтерфейси робочого столу (наприклад, wxWidgets, Kivy, Qt);
5. Розробка програмного забезпечення (контроль збирання, управління та тестування – Scons, Buildbot, Apache Gump, Roundup, Trac);
6. Бізнес-додатки (ERP та системи електронної комерції – Odoo, Tryton);

7. Ігри, веб-сайти та служби (наприклад, Dropbox, UBER, Pinterest, BuzzFeed).

4.1.2 JetBrains IntelliJ IDEA

IntelliJ IDEA — це інтегроване середовище розробки (IDE), написане на Java для розробки комп'ютерного програмного забезпечення, написаного на Java, Kotlin, Groovy та інших мовах на основі JVM. Він розроблений компанією JetBrains (раніше відомий як IntelliJ) і доступний як ліцензійне видання спільноти Apache 2, а також у комерційній версії. Обидва можуть бути використані для комерційного розвитку.

IDE надає такі можливості, як завершення коду шляхом аналізу контексту, навігація по коду, що дозволяє безпосередньо переходити до класу чи оголошення в коді, рефакторинг коду, налагодження коду, лінтування та параметри виправлення невідповідностей за допомогою пропозицій.

IDE забезпечує інтеграцію з такими інструментами збирання/пакування, як Grunt, bower, Gradle і sbt. Це середовище підтримує такі системи контролю версій, як Git, Mercurial, Perforce і SVN. Має доступ до таких баз даних, як Microsoft SQL Server, Oracle, PostgreSQL, SQLite та MySQL, можна отримати безпосередньо з IDE у версії Ultimate за допомогою вбудованої версії DataGrip, іншої IDE, розробленої JetBrains.

IntelliJ підтримує плагіни, за допомогою яких можна додати більше функції до IDE. Плагіни можна завантажити та встановити з веб-сайту сховища плагінів IntelliJ або за допомогою вбудованої функції пошуку та встановлення плагінів IDE. Кожне видання має окремі сховища плагінів, станом на 2019 рік як версії Community, так і Ultimate містять понад 3000 плагінів.

4.2 Огляд деталей реалізації

Наступні бібліотеки та засоби необхідні для імпорту у програмний код: librosa library, glob, numpy, matplotlib, послідовна модель з Keras, шар щільної нейронної мережі.

На відміну від згортки, наприклад, вона матиме двовимірне представлення. Тоді доведеться використовувати активацію імпорту, що дозволить надати кожному нейронному шару функцію активації, і to_categorical, яка дозволяє перетворювати назви класів на такі речі, як рок, диско тощо, що називається одноразовим кодуванням, наступним чином:

```
    def display_mfcc(song):
        y, _ = librosa.load(song)
        mfcc = librosa.feature.mfcc(y)
        plt.figure(figsize=(10, 4))
        librosa.display.specshow(mfcc,
x_axis='time',y_axis='mel')
        plt.colorbar()
        plt.title(song)
        plt.tight_layout()
        plt.show()
```

Спочатку завантажимо пісню, а потім витягнімо із неї значення MFCC. Потім користуючися specshow — спектрограмою з бібліотеки librosa, можемо отримати ударний барабан:

```
display_mfcc('kick-loop.wav')
```

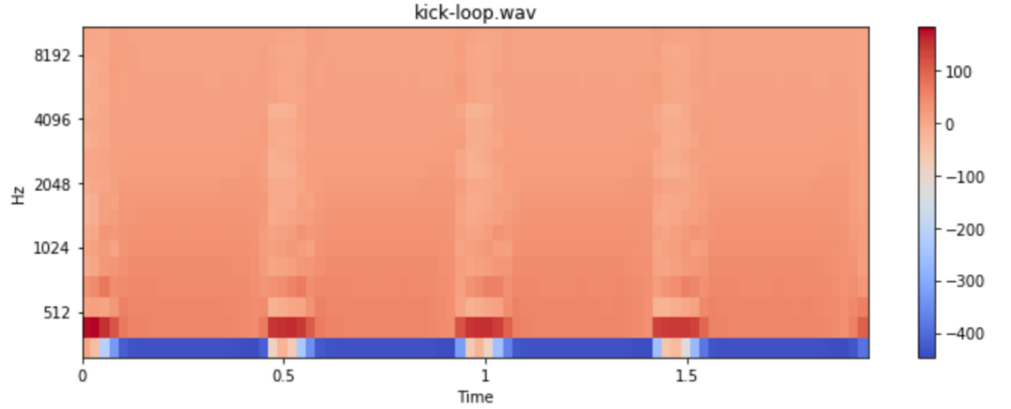

Рисунок 4.1 — Спектрограмма барабану

На низькій частоті бас дуже очевидний, а в решту часу він нагадує “промивання”. На рисунку представлено небагато інших частот. Однак, якщо порівнювати зі свистом, то цілком зрозуміло, що представлені вищі частоти:

```
display_mfcc('whistling.wav')
```

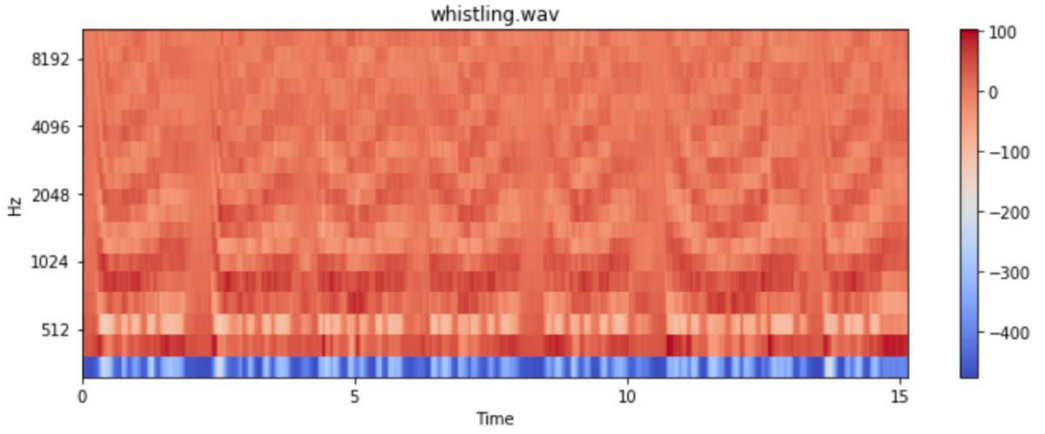

Рисунок 4.2 — Спектрограмма свисту

Чим темніший колір або ближчий до червоного, тим більше потужності в цьому частотному діапазоні в цей час. Навіть можливо побачити зміну частоти зі свистом. Нижче наведено частоту диско-пісень (рисунок 4.3).

```
display_mfcc('genres/disco/disco.00070.au')
```

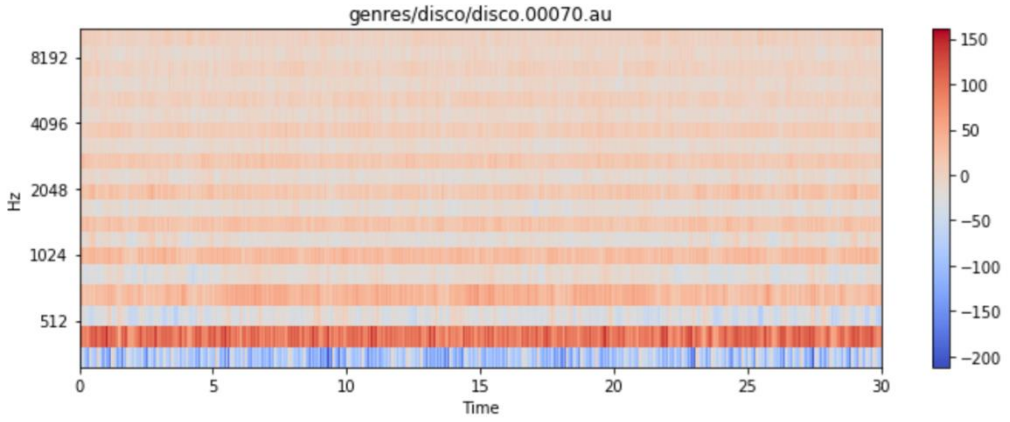

Рисунок 4.3 — Спектрограмма диско-пісень

Можливо побачити ритм в спектрограммі, але оскільки вони тривають лише 30 секунд, важко побачити окремі ритми. Порівняємо це з класичною музикою, де немає такої кількості ударів, як безперервна басова лінія, наприклад, така, як у віолончелі (рисунок 4.4).

```
display_mfcc('genres/classical/classical.00067.au')
```

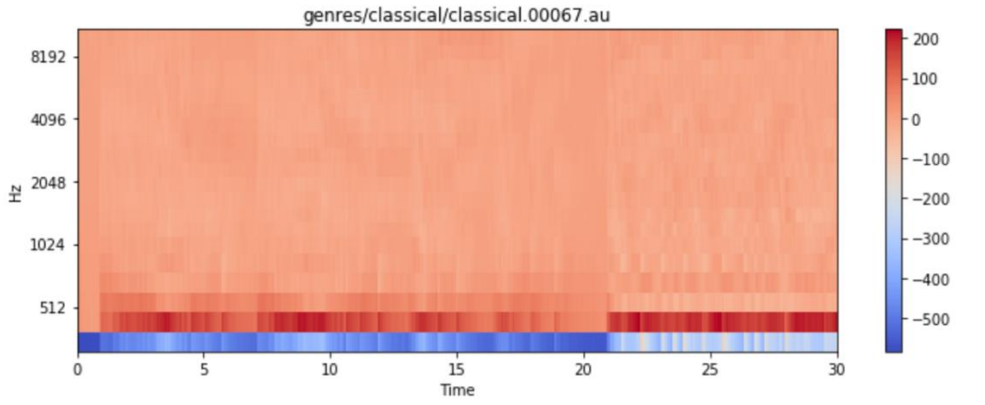

Рисунок 4.4 — Спектрограмма віолончелі

Спектрограмма хіп-хопу (рисунок 4.5) виглядає дещо схоже на стиль диско, але якби було потрібно побачити різницю власними очима, тоді насправді не знадобилася б нейронна мережа, тому що це, мабуть, була б

відносно проста проблема. Тож той факт, що неможливо відрізнити їх, є проблемою нейронної мережі.

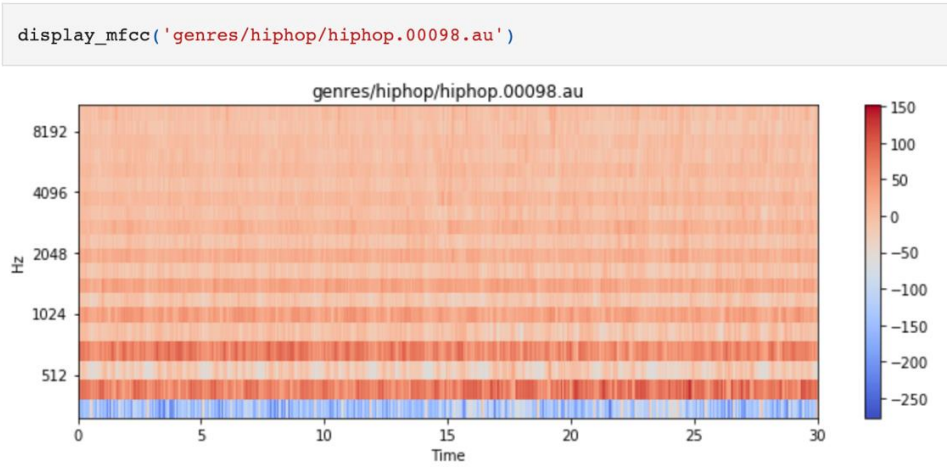

Рисунок 4.5 — Спектрограмма хіп-хопу

Є ще одна допоміжна функція, яка знову завантажує лише значення MFCC, але цього разу вона готується для нейронної мережі:

```
def extract_features_song(f):
    y, _ = librosa.load(f)
    mfcc = librosa.feature.mfcc(y)
    mfcc /= np.amax(np.absolute(mfcc))
    return np.ndarray.flatten(mfcc)[:25000]
```

Також завантажуються значення MFCC для пісні, але оскільки ці значення можуть бути від -250 до +150, вони не підходять для нейронної мережі. Тоді потрібно вводити значення, близькі до -1 і +1 або від 0 до 1.

Тому необхідно визначити максимальне та абсолютне значення для кожної пісні. Потім розділити усі значення на це максимальне значення. Крім того, пісні мають дещо іншу довжину, тому необхідно вибрати лише 25 000 значень MFCC. Те, що подається в нейронну мережу, завжди має однаковий розмір, тому що вхідних нейронів дуже багато, і неможливо це змінити, створивши мережу.

Далі є функція під назвою generate_features_and_labels, яка переглядатиме всі різні жанри та всі пісні в наборі даних і створюватиме ці значення MFCC та назви класів:

```
    def generate_features_and_labels():
        all_features = []
        all_labels = []
        genres = ['blues', 'classical', 'country', 'disco',
'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']
        for genre in genres:
            sound_files = glob.glob('genres/'+genre+'/*.au')
            print('Processing %d songs in %s genre...' %
(len(sound_files), genre))
            for f in sound_files:
                features = extract_features_song(f)
                all_features.append(features)
                all_labels.append(genre)
        label_uniq_ids, label_row_ids = np.unique(all_labels,
return_inverse=True)
        label_row_ids = label_row_ids.astype(np.int32,
copy=False)
     onehot_labels=to_categorical(label_row_ids,
len(label_uniq_ids))
        return np.stack(all_features), onehot_labels
```

Необхідно підготувати список усіх функцій і міток. Для кожного жанру необхідно переглянути файли в цій папці. Папка «generes/'+genre+'/*.au» показує, як організовано набір даних.

При обробці цієї папки в кожному файлі буде по 100 пісень; буде можливо витягнути функції та помістити їх у список all_features.append(features). Назву жанру цієї пісні також потрібно додати до списку. Отже, зрештою всі функції матимуть 1000 записів, а всі мітки матимуть

1000 записів. У випадку всіх функцій кожна з цих 1000 записів матиме 25 000 записів. Це буде матриця 1000 25000.

Тепер для всіх лейблів існує список із 1000 записів, у якому є такі слова, як блюз, класика, кантрі, диско, хіп-хоп, джаз, метал, поп, реггі та рок. Тепер це буде складністю, оскільки нейронна мережа не збирається передбачати слово чи навіть букви. Потрібно надати однократне кодування, що означає, що кожне слово тут буде представлено як десять двійкових чисел. У випадку з блюзом це буде одиниця, а потім дев'ять нулів. У класичному випадку це буде нуль, за яким іде одиниця, за якою слідують дев'ять нулів і так далі. По-перше, необхідно визначити всі унікальні імена за допомогою команди np.unique(all_labels, return_inverse=True), щоб повернути їх як цілі числа. Потім скористатися to_categorical, що перетворює ці цілі числа в одноразове кодування.

Отже, повертається розмір 1000 10. 1000, тому що є 1000 пісень, і кожна з них має десять двійкових чисел, які представляють одноразове кодування. Потім повернемо усі функції, зібрані разом, за допомогою команди return np.stack(all_features), onehot_labels в одну матрицю, а також матрицю one-hot. Отже, викличемо цю функцію та збережемо функції та мітки:

```
features, labels = generate_features_and_labels()
```

Щоб переконатися, надрукуємо форму елементів і міток. Отже, це 1000 на 25 000 для функцій і 1000 на 10 для міток. Тепер розділимо набір даних на потяг і тестовий поділ. Визначимо позначку 80% — training_split= 0,8, щоб виконати розщеплення:

```
print(np.shape(features))
print(np.shape(labels))
training_split = 0.8
alldata = np.column_stack((features, labels))
np.random.shuffle(alldata)
```

```
splitidx = int(len(alldata) * training_split)
train, test = alldata[:splitidx,:], alldata[splitidx:,:]
print(np.shape(train))
print(np.shape(test)
train_input = train[:,:-10]
train_labels = train[:,-10:]
test_input = test[:,:-10]
test_labels = test[:,-10:]
print(np.shape(train_input))
print(np.shape(train_labels))
```

Потрібно розмістити мітки з функціями, щоб вони не перемішувалися в різному порядку. Після виклику np.random.shuffle(alldata) і виконання перемішування, розділимо його, і тоді матимемо тестові набори.

Дивлячись на форму тестових наборів, потяг становить 800, тобто 80% із 1000 для рядків: 25 010 функцій. Однак це ще не всі функції. Насправді це 25 000 функцій плюс 10 для одноразового кодування, тому що вони були складені разом перед перетасуванням. Тому треба відняти це назад.

Можливо зробити це за допомогою train_input = train[:,:-10]. І для навчального введення, і для тестового введення візьмемо усе, крім останніх 10 стовпців, а для міток візьмемо 10 стовпців до кінця, і тоді будуть отримані форми навчального вводу та міток навчання. Отже, тепер є відповідні 800 на 25 000 і 800 на 10.

Отримаємо в результаті нейромережу:

```
model = Sequential([
    Dense(100, input_dim=np.shape(train_input)[1]),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
    ])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
```

```
                  metrics=['accuracy'])
    print(model.summary())
    model.fit(train_input,        train_labels,        epochs=10,
batch_size=32,
              validation_split=0.2)
    loss,    acc=    model.evaluate(test_input,    test_labels,
batch_size=32)
    print("Done!")
    print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
```

Матимемо послідовну нейронну мережу. Перший шар буде щільним шаром із 100 нейронів. Тепер, лише на першому шарі, важливо, щоб були вказані вхідні розміри, і це буде 25 000 у цьому випадку.

Це говорить про те, скільки вхідних значень надходить на приклад. Ці 25 000 підключаються до 100 на першому рівні.

Перший рівень виконує зважену суму своїх вхідних даних, своїх ваг і зміщення, а потім запустить функцію активації relu. Ця функція стверджує, що все, що менше 0, буде 0, все, що вище 0, буде цим значенням.

Потім ці 100 з’єднаються з ще 10, і це буде вихідний рівень. Це буде 10, тому що у цьому кодуванні 10 двійкових чисел.

Активація, яка використовується в коді, softmax, бере вихідні дані з 10 і нормалізуватиме їх так, щоб вони в сумі дорівнювали 1. Таким чином, вони в кінцевому підсумку стають ймовірностями. Тепер розглянемо найвищий бал або найвищу ймовірність із 10 як прогноз. Це безпосередньо відповідатиме позиції найвищого номера. Наприклад, якщо він знаходиться в положенні 4, це буде диско стиль.

Після компіляції моделі, за допомогою оптимізатора, наприклад Adam, необхідно визначити функцію витрат. Кожного разу, коли є кілька виходів, ймовірно, буде виконуватися перехресна ентропія за категоріями та точністю метрик, щоб побачити точність під час навчання та під час оцінки, на додаток до втрат, які завжди показуються. Однак точність має більше сенсу. Далі надрукуємо model.summary, що надає інформацію про шари, як на рисунку 4.6.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               2500100
_________________________________________________________________
activation_1 (Activation)    (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1010
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 2,501,110
Trainable params: 2,501,110
Non-trainable params: 0
_________________________________________________________________
None
```

Рисунок 4.6 — Інформація про шари нейромережі

Вихідна форма першого шару зі 100 нейронів однозначно має 100 значень, оскільки є 100 нейронів, а вихідна інформація щільного другого шару дорівнює 10, оскільки є 10 нейронів. Отже, чому в першому шарі 2,5 мільйона параметрів або ваг? Це тому, що є 25 000 входів. Кожен із цих входів спрямовується до кожного зі 100 щільних нейронів. Отже, це 2,5 мільйона, а потім додатково 100, тому що кожен із цих нейронів має свою власну вагу зміщення.

Загалом маємо близько 2,5 мільйонів параметрів або ваг. Далі запустимо підгонку. Вона приймає навчальні вхідні дані та навчальні мітки та приймає потрібну кількість жанрів. Необхідно 10 повторень над тестовими даними. Потрібен розмір партії, який повідомляє кількість, у цьому випадку, пісень, які потрібно пройти перед оновленням ваг; і validation_split 0,2 говорить про те, що потрібно взяти 20% тестових даних, розділити їх. Це потрібно, щоб оцінити, наскільки добре мережа працює після кожного жанру. Насправді мережа ніколи не тренується на верифікаційному спліті, але верифікаційний спліт дає змогу спостерігати за прогресом. Результати з навчання можна побачити на рисунку 4.7. Можливо, менш бажано мати значення точності близьке до 1, бо це буде свідчити про надмірність, але, якщо затримати виконання програми досить

довго, вона часто досягає точності 1,0 на навчальному наборі, оскільки він запам’ятовується.

```
Train on 640 samples, validate on 160 samples
Epoch 1/10
640/640 [==============================] - 2s 3ms/step - loss: 2.0585 - acc: 0.2906 - val_loss: 1.7780 - val_acc: 0.3187
Epoch 2/10
640/640 [==============================] - 0s 720us/step - loss: 1.4080 - acc: 0.5031 - val_loss: 1.5680 - val_acc: 0.4562
Epoch 3/10
640/640 [==============================] - 0s 723us/step - loss: 1.1128 - acc: 0.6281 - val_loss: 1.5202 - val_acc: 0.4625
Epoch 4/10
640/640 [==============================] - 0s 697us/step - loss: 0.8968 - acc: 0.7422 - val_loss: 1.4163 - val_acc: 0.5062
Epoch 5/10
640/640 [==============================] - 0s 707us/step - loss: 0.7990 - acc: 0.7734 - val_loss: 1.9091 - val_acc: 0.4625
Epoch 6/10
640/640 [==============================] - 0s 712us/step - loss: 0.6336 - acc: 0.8266 - val_loss: 1.4158 - val_acc: 0.5375
Epoch 7/10
640/640 [==============================] - 0s 690us/step - loss: 0.4935 - acc: 0.9031 - val_loss: 1.4425 - val_acc: 0.5188
Epoch 8/10
640/640 [==============================] - 0s 717us/step - loss: 0.3547 - acc: 0.9500 - val_loss: 1.4821 - val_acc: 0.4813
Epoch 9/10
640/640 [==============================] - 0s 710us/step - loss: 0.2855 - acc: 0.9672 - val_loss: 1.4005 - val_acc: 0.5312
Epoch 10/10
640/640 [==============================] - 0s 706us/step - loss: 0.2247 - acc: 0.9891 - val_loss: 1.4223 - val_acc: 0.4938
200/200 [==============================] - 0s 452us/step
Done!
Loss: 1.5206, accuracy: 0.5250
```

Рисунок 4.7 — Результати навчання мережі

Значення точності перевірки може бути достатньо неочікуваним після тестового набору. Ця ітогова точність залежить від даних тестування, які мережа ніколи раніше не розглядала. Тепер отримана точність буде близько 53%. Це здається відносно низьким, але треба брати до уваги, що це для 10 різних жанрів. Випадкове вгадування дасть лише 10% точності, тому отримані результати досить непогані.

ВИСНОВКИ

На підставі виконаної роботи було проведено дослідження автоматичної класифікації музичних творів за жанрами та створення програмної системи. Це дозволяє виконувати автоматичну класифікацію з використанням згорткової нейронної мережі.

У ході виконання даної роботи було проведено аналіз вимог до системи розпізнавання музичних жанрів. Було зроблено огляд вже існуючих систем розпізнавання музики, визначено їх основний функціонал. Розглянуто основні аспекти тембру, частотного аналізу. Детально розглянуто перетворення мел-кепстральних частотних коефіцієнтів.

Було зроблено огляд методів класифікації даних, а саме нейронних мереж (в тому числі і згорткових) та методу опорних векторів. Виконано порівняння існуючих систем, що використовують ці методи, на основі чого було обрано метод класифікації, який реалізується в цій роботі, а саме згорткова нейронна мережа.

Розроблено програмну систему для автоматичної класифікації музичних творів за жанрами. Ця система була описана за допомогою програмних засобів мови Python.

Кінцевий продукт може бути вдосконалений у майбутньому — стати серверною частиною, при тому маючи графічний інтерфейс користувача (він може бути як і веб-сторінкою, так й мати десктопну або мобільну версію). З подальшим розвитком, мережу можливо буде навчити вгадувати конкретні музичні композиції, або підбирати композиції за заданим ритмом/настроєм/тощо, або сортувати їх при пошуку.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zhang, Wei (1990). Parallel distributed processing model with local space-invariant interconnections and its optical architecture. Applied Optics 29 (32): 4790–7.
2. Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020). "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". Mathematics and Computers in Simulation. Elsevier BV. 177: 232–243. doi:10.1016/j.matcom.2020.04.031. ISSN 0378-4754. S2CID 218955622
3. Matusugu, Masakazu; Katsuhiko Mori; Yusuke Mitari; Yuji Kaneda (2003). "Subject independent facial expression recognition with robust face detection using a convolutional neural network" (PDF). *Neural Networks*. **16** (5): 555–559. doi:10.1016/S0893-6080(03)00115-1. PMID 12850007
4. Accelerated learning of the neural network ADALINE in the presence of stationary correlated. Oleksandr Bezsonov, Oleg Rudenko, Natalia Serdiuk, Oleg Lebediev / 10-та Міжнародна науково-технічна конференція «ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ ІСТ-2021» 13 − 19 вересня 2021 р
5. Wang, Avery Li-Chun (2003). "An Industrial-Strength Audio Search Algorithm". International Conference on Music Information Retrieval (ISMIR). CiteSeerX 10.1.1.217.8882
6. Sejdic, E.; Djurovic, I.; Stankovic, L. (August 2008). "Quantitative Performance Analysis of Scalogram as Instantaneous Frequency Estimator". *IEEE Transactions on Signal Processing*. **56** (8): 3837–3845. Bibcode:2008ITSP...56.3837S. doi:10.1109/TSP.2008.924856. ISSN 1053-587X. S2CID 16396084
7. Alexander Waibel et al., *Phoneme Recognition Using Time-Delay Neural Networks* IEEE Transactions on Acoustics, Speech and Signal Processing, Volume 37, No. 3, pp. 328. — 339 March 1989.

8. Matusugu, Masakazu; Katsuhiko Mori; Yusuke Mitari; Yuji Kaneda (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks* **16** (5): 555–559. doi:10.1016/S0893-6080(03)00115-1
9. Fukushima, Kunihiko (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics 36 (4): 193–202. PMID 7370364. doi:10.1007/BF00344251
10. Qiu Huang, Daniel Graupe, Yi Fang Huang, Ruey Wen Liu."Identification of firing patterns of neuronal signals." In Proc. 28th IEEE Decision and Control Conf., pp. 266—271, 1989.
11. Habibi,, Aghdam, Hamed. Guide to convolutional neural networks : a practical application to traffic-sign detection and classification. Heravi, Elnaz Jahani,. Cham, Switzerland. ISBN 9783319575490. OCLC 98779095