ГЕНЕРАЦІЯ МУЗИКИ З УРАХУВАННЯМ ЕМОЦІЙНОЇСКЛАДОВОЇ КОРИСТУВАЦЬКОГО ЗАПИТУ

GENERATION OF MUSIC BASED ON THE EMOTIONAL COMPONENT OF THE USER REQUEST

ANNOTATION

The graduation research of the 4-year student A. Chebotarova (Oles Honchar Dnipro National University, Faculty of Applied Mathematics, Department of Mathematical Support of Computers) deals with software development for the music generation based on user`s emotion

The purpose of the work is to develop a system of neural networks and additional functions that classify the user request according to the class of emotion and generate music in the genre corresponding to the emotion.

In the process of work, a neural network classifier was developed that classifies a user request by emotion and a neural network based on generative-competitive neural networks. Algorithms for preparing data for training, visualization of the obtained results, methods for improving the quality of the neural network and methods for stabilizing training were used during the work. An analysis of the influence of various parameters and architectures of neural networks on the quality of learning was carried out.

The software for generating music based on user emotion is developed on Python in the Goggle Collaboratory environment, Tensorflow machine learning libraries, text, visual and audio data processing libraries on Windows 10.

Bibliography 23, pictures 81, tables 0.

ЗМІСТ

ВСТУП

В суспільстві, від його появлення і до сьогодення, мистецтво вважається складною для машинного розуміння областю, оскільки передає складно описувані стани: емоції, стан душі автора, ідею тощо. Завдяки механізму асоціацій, людський мозок прив'язує подію, колір, зображення чи музику до конкретної емоції чи групи емоцій.

У сучасних музичних генераторах, щоб згенерувати музику потрібно вводити музичні параметри, вплив яких розуміє музикант або людина, яка працює у музичній сфері. Більшість користувачів, яким потрібна музика для власних задач, не є мають знань у музичній сфері, генерація музики, що займає декілька секунд для алгоритми, може займати десятки хвилин для таких користувачів. Дана робота базується на механізмі асоціації та пропонує алгоритм, що може полегшити процес генерації музики для користувачів. (актуальність)

Завдяки розвитку Штучного Інтелекту людина може автоматизувати частину своєї роботи: починаючи з простих задач, на кшталт обробки зображень та закінчуючи життєвонеобхідними: діагностування раку та інших хвороб. ШІ показав високу якість не тільки у точних задачах, але й у задачах state - of -art: задачі генерації зображень, перетворення цих зображень, генерація музики чи написання текстів. Високу якість забезпечують не тільки точні алгоритми, але й людина, що коректно формулює запит до нейромережі. Актуальність музичних генераторів — програм або застосунків, що можуть генерувати музику, настільки ж висока, як і генераторів state - of -art зображень. Як і варіантів використання музики, варіантів використання розробленого програмного забезпечення багато, але основні області це — музичний супровід для інді - ігор, фонове музичне звучання для сайтів, презентацій, тощо. Використання музики у перелічених областях є другорядним аспектом, тому наймати професійну команду музикантів для створення музики не є розповсюдженою практикою.

Програма розроблена з використанням точних та оптимізованих алгоритмів машинного навчання. Основні функції програми полягають у визначенні емоції користувача та подальшій генерації музики, яка співвідноситься цій емоції.

Основною задачею, яке дане програмне забезпечення покликане розв’язати, це задача спрощення та максимізування якості користувацького запиту, що в свою чергу покращує результат генерації. Якість нейромереж, які на вхід отримують запит, та якість яких оцінює користувач, значною мірою залежить від того, наскільки точний запит ввів користувач. Оскільки більшість користувачів не мають музичної освіти та незнайомі з музичними термінами, які необхідно вводити в генераторах музики, дане програмне забезпечення використовує емоцію замість них.

Постановку задач для даної роботи можна розділити на такі групи

1) Класифікація користувацького запиту за емоцією.
2) Генерація музичних спектрограм у жанрі, що відповідає цій емоції.
3) Перетворення музичних спектрограм у музичну послідовність.

1. Задача класифікації користувацького запиту за емоцією

Мета першої частини — це розробка алгоритму машинного чи глибокого навчання (нейромережі) для класифікації користувацького запиту за однією з емоцій. Розроблений алгоритм повинен мати високу якість генерації та оптимізований для швидкого виконання класифікації.

Користувацький запит — це емоційне забарвлений текст, який може мати визначену довжину та складатися з одного або декількох речень.

Результатом класифікації користувацького запиту має бути клас (мітка) емоції та її точність у відсотках. Точність класифікації - числовий показник, який показує наскільки якісною є класифікація,

тобто наскільки сильно одна емоція домінує над іншими у користувацькому запиті.

2. Задача генерації музичних спектрограм у жанрі, що відповідають емоції

Для генерації музичних спектрограм повинна бути розроблена система, алгоритм машинного чи алгоритм глибокого навчання, результатом роботи якого буде згенеровані спектрограмі. Розроблений алгоритм повинен мати функцію зберігання прикладів згенерованих спектрограм параметрів, за якими визначається якість роботи алгоритма. Обраний алгоритм має бути якісний та оптимізований для середовища, в якому буде навчатися та працювати.

3. Перетворення музичних спектрограм у музичну послідовність

Після розробки першої та другої частини необхідно розробити функцію, яка буде перетворювати спектрограми обраного типу на музичну послідовність у визначеному форматі. Після перетворення функція має давати користувачу можливість прослухати та зберегти музику. Для успішної розробки наведений функції потрібно опиратися на математичні функції та параметри, з яких складається спектрограма наведеного типу,

(методи та інструменти)

Методи та інструменти, що використувалися для розробки ПЗ це:

1. Методології розробки та навчання алгоритмів машинного навчання.
2. Методи обробки тренувальних даних.
3. Інструментами розробки є: мова програмування Python, бібліотеки роботи з матрицями, бібліотеки роботи з зображеннями та музикою, бібліотеки машинного та глибокого навчання мови, бібліотека віджетів мови Python.
4. Середовище виконання Google Collaboratory.

Зміст роботи по розділах є наступним:

1. Розділ перший — опис предметної області та засоби розробки. Опис предметної області наводить теоретичні відомості про використані теоретичні знання. Засоби розробки містять детальний опис та обгрунтування вибору використаної мови, бібліотек та середовища.
2. Розділ другий — опис програмної реалізації та інтерфейсу користувача. Опис програмної реалізації містять детальні відомості про методи, які були використані для виконання кожного пункту завдання, також містить результати виконання цих методів.Інтерфейс користувача описує причину, чому інтерфейс виглядає наведеним чином, які засоби та чому використовувалися для проектування інтерфейсу, наводиться вигляд інтерфейсу користувача.
3. Розділ третій - аналіз впливу параметрів алгоритмів та тестування. Аналіз впливу параметрів алгоритмів представляє результати експериментів, які були проведені під час виконання роботи. Цей пункт представляє з себе розділені по задачах експерименти, їх результат та обґрунтування цих результатів. Пункт тестування містить в собі результати тестування ПЗ з реальними даними.

1 ПРЕДМЕТНА ОБЛАСТЬ ТА ЗАСОБИ РОЗРОБКИ

1.1 Використання нейромереж для вирішення поставленої задачі

Область машинного навчання — це область, що вивчає алгоритми, які можуть вчитися на основі інформації та виконувати визначені конкретним алгоритмом задачі. Область машинного навчання включає в себе область штучного інтелекта, яка містить в собі складні алгоритми машинного навчання, що можуть опрацьовувати більший спектр даних.

Нейромережі, на яких базується дана робота, представляють з себе математичні моделі, які складаються з таких елементів: вхідна інформація, активаційні функції, матриця вагових коефіцієнтів,зміщення (biases),додаткові компоненти для стабілізації навчання, коефіцієнт швидкості навчання та нейронии (аналог нейронів в мозку людини). Архітектура нейронних мереж частково копіює архітектуру людського мозку (рис. 1.1), використовуючи нейронну архітектуру та зв’язки між нейронами.

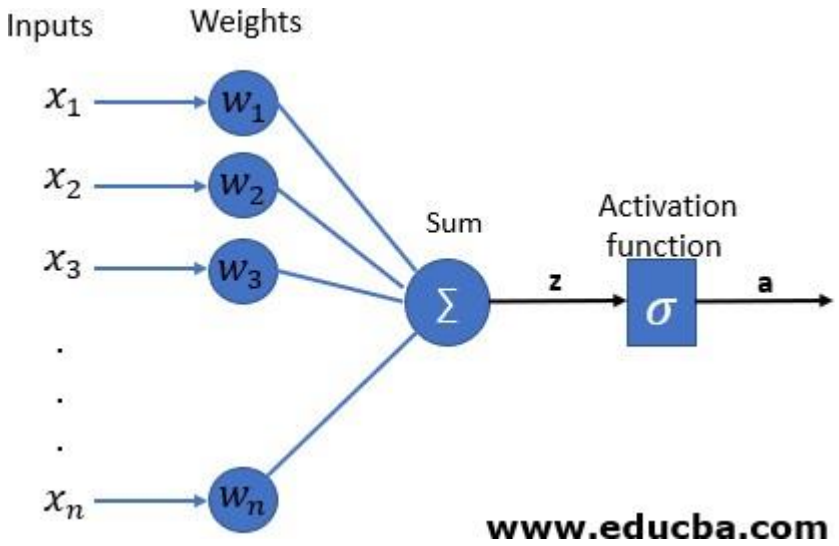

Рисунок 1.1 — Схема нейронної мережі

Нейромережа змінює вхідну інформації так, щоб результат роботи був максимально близький до правильного результата (істинного значення),

шляхом зміни параметрів нейромережі, зокрема вагових коефіцієнтів. Різницю між результатом роботи (передбаченням) нейромережі та правильним результатом вираховує функція втрат. Математична задача нейромережі — це мінімізація функції втрат (через зменшення різниці) (рис. 1.2) шляхом знаходження глобального мінімума функції та максимізація функції якості. Значення функції втрати показує, наскільки точно працює алгоритм, похідна функції визначає напрямок, в якому потрібно рухатися для знаходження глобального мінімума. Існує декілька типів функцій втрат, які використовуються для окремих задач: MSE (Середня квадратична помилка) обчислює суму квадратів різниці між передбаченням нейромережі та істинним значенням, крос - ентропійна використовується в задачах бінарної класифікації, де істині значення є мітками двох класів (0 чи 1), тоді як передбачення нейромережі є не цілим значенням (0.52, 0.74, тощо), функція обчислює суму істинних значень, множить на мінус одиницю та на логарифм від передбачених значень, підвидом крос - ентропії є категоріальна крос - ентропія використовується в задачах багатокласової класифікації, MAPE (Втрати середньої абсолютної відсоткової помилки) використовують для аналізу якості нейромережі для задач прогнозування попиту.

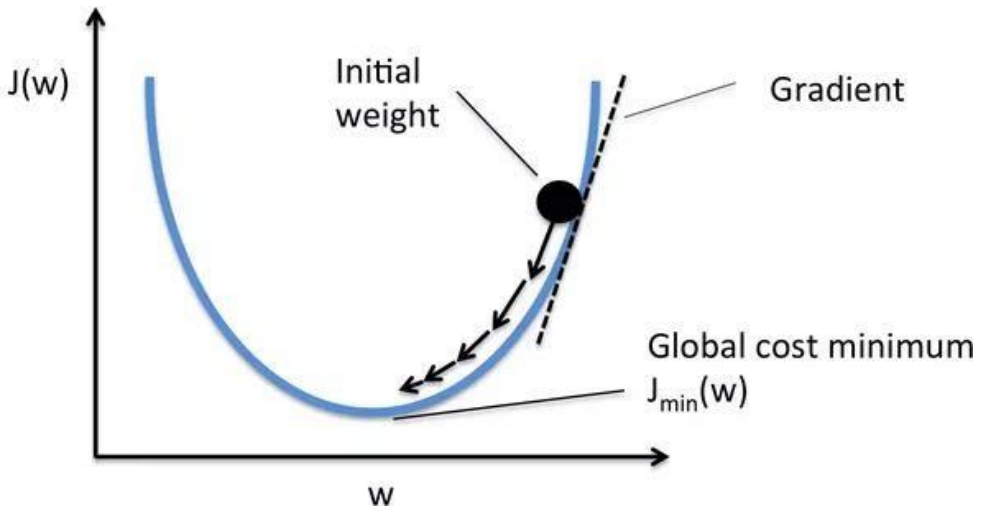

Рисунок 1.2 — Складові функції втрат

Нейромережа рухається у двох напрямках: прямого/проходження (англ. Forward Propagation) та зворотнього поширення/проходження (англ. Backward propagation). Під час прямого поширення обчислюються: сума вхідних параметрів, активаційна функція від суми, функція втрат, тазміщення (biases), додатково може обчислюватися нормалізація. При зворотньому проходженні обчислюються часткові похідні параметрів, що обчислюються під час прямого проходження та змінюються матриці вагових коефіцієнтів та коефіцієнтів зміщення (рис. 1.3).

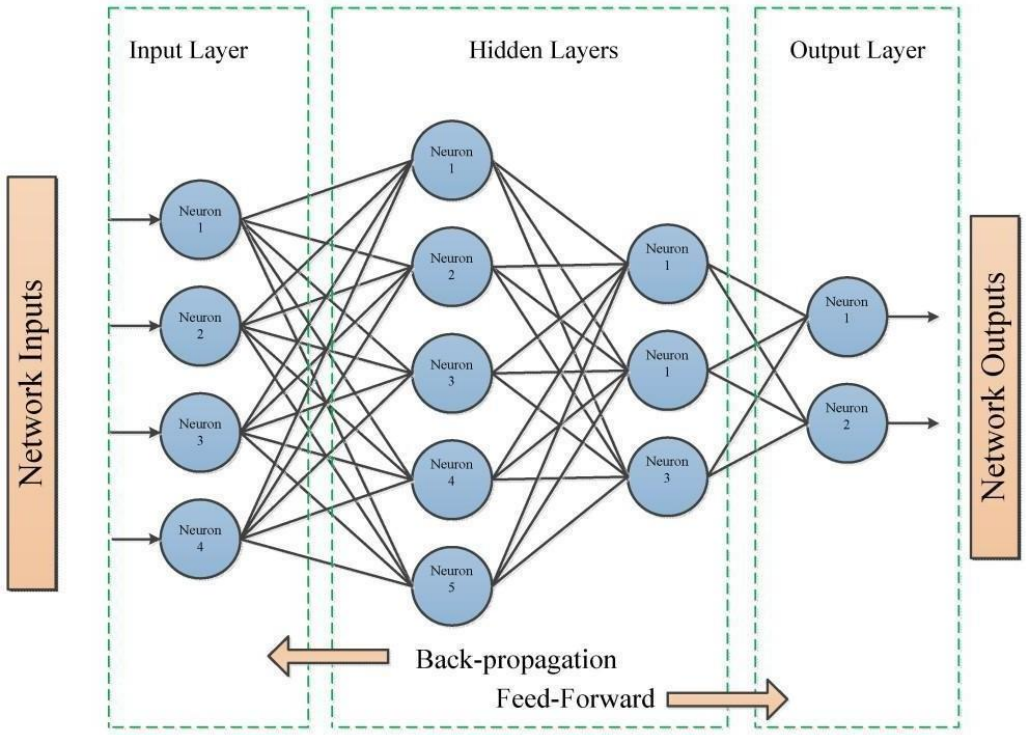

Рисунок 1.3 — Візуалізація напрямків руху проходжень у нейромережах

У цій роботі використовуються наступні алгоритми та практики:

- Алгоритми класифікації тексту за емоціями
- Двонаправлена рекурентна нейромережа
- Генеративно - змагальні нейромережі
- Алгоритми підготовки навчальних даних
- Практики подання даних до нейромереж та їх навчання
- Фізичні аспекти генерування та перетворення Mel спектрограм на музику.

1.2 Алгоритми класифікації тексту за емоціями

Задача класифікації тексту за емоціями — це задача з групи Sentiment Analysis області обробки людських мов (NLP). Ця область є областю штучного інтелекту через складність обробки та розуміння людської мови математичним алгоритмом. Складність обробки полягає в тому, що мовапередає параметри, які складно описати числовими методами, а саме: емоції, ідеї, форма, захований сенс, тощо. На даний момент розроблені багато методів, що дозволяють опрацьовувати текст алгоритмам машинного навчання. Одним з таких методів є методи токенізації та енкодування, що полягають у перетворенні тексту у числовий вектор шляхом присвоєння кожному слову окремого номера.

Метою задачі класифікації тексту є розробка алгоритму - класифікатора, що класифікуватиме текст по жанрам емоцій. Такі алгоритми використовуються в багатьох областях, починаючи від аналізу політичних ідей серед населення та закінчуючи аналізом купівельної спроможності та відгуків. Основними алгоритмами є алгоритми машинного: Naive Bayes. Support Vector Machines та глибокого навчання: RNN, Attention.

1.3 Двонаправлена рекурентна нейромережа

Рекурентні нейромережі (RNN) — це нейромережі, які обробляють послідовності, визначені у часі. Складаються з модулей (рис.1.4), які пов’язані один з одним, кількість входів та виходів залежить від задачі.

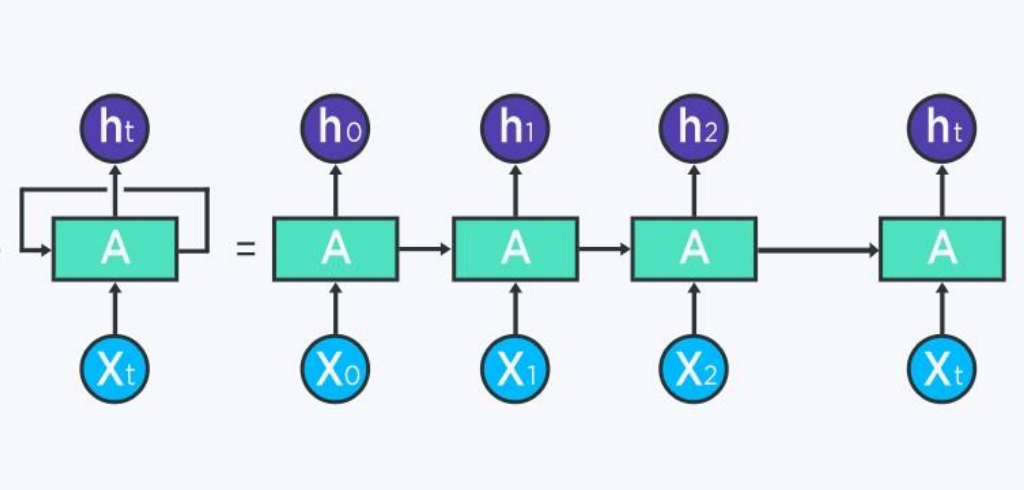

Рисунок 1.4 - Модуль рекурентної нейромережі

На відміну від нейромереж прямого проходження, рекурентні нейромережі мають «пам’ять» про інформацію, що до них надходить завдяки спеціальному механізму: інформація про проходження повертається до модуля (рис. 1.5).

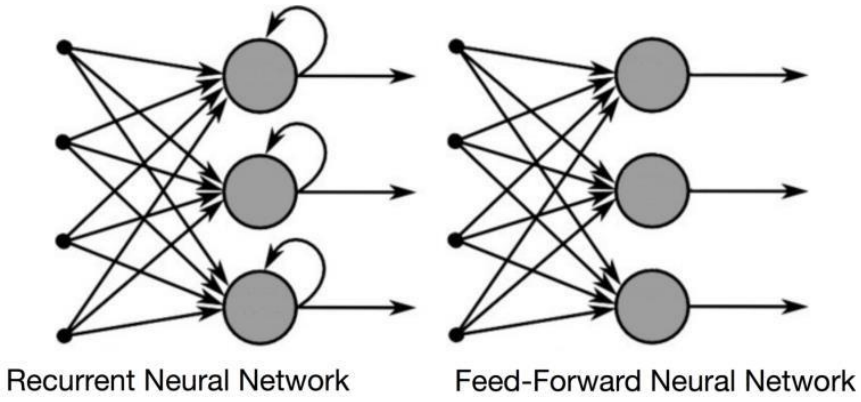

Рисунок 1.5 — Порівняння архітектур нейромереж прямого проходження та рекурентних

Поточна ітерація, змінюючи вагові коефіцієнти, враховує інформацію, що надійшла в цій ітерації та попередній. Функція збереження інформації (h_t) вираховується з попередньої матриці вагових коефіцієнтів (Wh_{t-1}) та поточної (Wx_t), для вирахування передбачення на поточній ітерації (y_t) найчастіше

використовують активаційну функцію softmax, функція втрат (J) вираховується як кросс - ентропійна.

$$1)\quad h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$$

$$2)\quad y_t = softmax(W^{(S)}h_t)$$

$$3)\quad J^{(t)}(\theta) = \sum_{i=1}^{|V|}(y'_{t_i} log\, y_{t_i})$$

Завдяки прив'язці даних до часу, в цій нейромережі є попередні, теперішні та майбутні ітерації (кроки). Оригінальні рекурентні нейромережі мають проблему затухаючого градієнту (англ. Vanishing gradient problem), яка виникає, коли значення часткових похідних функції втрат наближається до нуля або стає нулем (рис. 1.6).

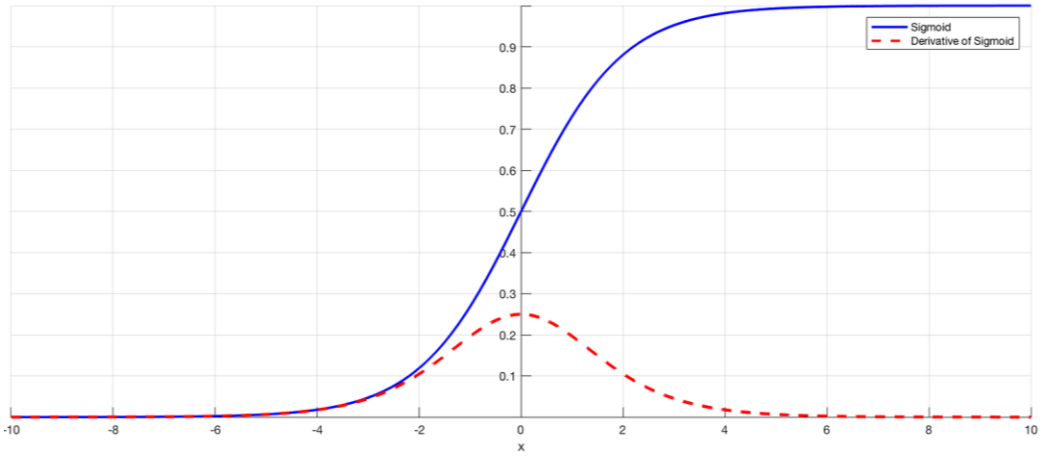

Рисунок 1.6 - Візуалізація проблеми затухаючого градієнту

Вирішення цієї проблеми реалізовано у LSTM [1] модуль — завдяки більш складній архітектурі кожного блоку, цей модуль може знаходити, зберігати та передавати далі тільку важливу інформацію: модуль має три типи «воріт» з функціями активації сигмоїд або тангенс. При навчанні параметри перших "воріт", які зберігають інформацію, змінюються, завдяки цьому параметри, що можуть призвести до затухання градієнта не обираються.

Двонаправленість — це розширення оригінальних рекурентних нейромеж, яке обходить послідовність у двох напрямках. Двонаправленість

у рекурентних нейромережах реалізується наступним чином: створюються два нейрони, замість одного для оригінальних рекурентних нейромереж, перший нейрон проходить послідовність прямо (позитивний напрямок), другий нейрон у зворотньому напрямку (негативний напрямок), в результаті інформація з попереднього та майбутнього проходження отримується одночасно (рис. 1.7). Це дозволяє знайти більш глибокі паттерни у даних.

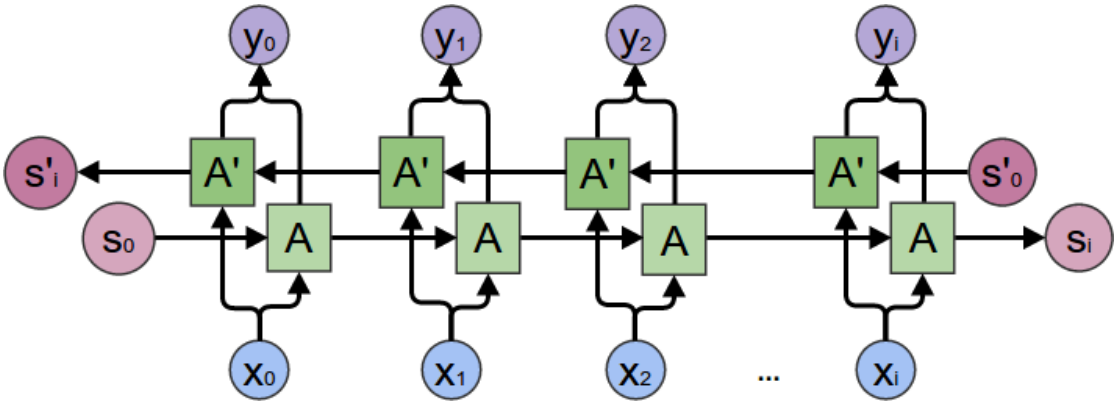

Рисунок 1.7 — Двонаправлена рекурентна нейромережа

1.4 Генеративно - змагальні нейромережі

Генеративно - змагальні нейромережі (GAN) є алгоритмом глибокого машинного навчання, що найчастіше використовують у задачах генерації зображень через високу якість на цих задачах. Ці нейромережі використовують навчання без вчителя — це вид машинного навчання, де нейромережа вчиться без втручання з боку експериментатора, завдяки власній архітектурі виконують дві задачі одночасно: генерація даних та їх оцінка (класифікація) якості згенерованих даних, завдяки чому нейромережа “самонавчається”. Складові — це генератор, що генерує дані тадискримінатор, що класифікує ці дані як справжні (генератор генерує якісні зображення) та фейкові (генератор не може згенерувати якісні дані) (рис. 1.8).

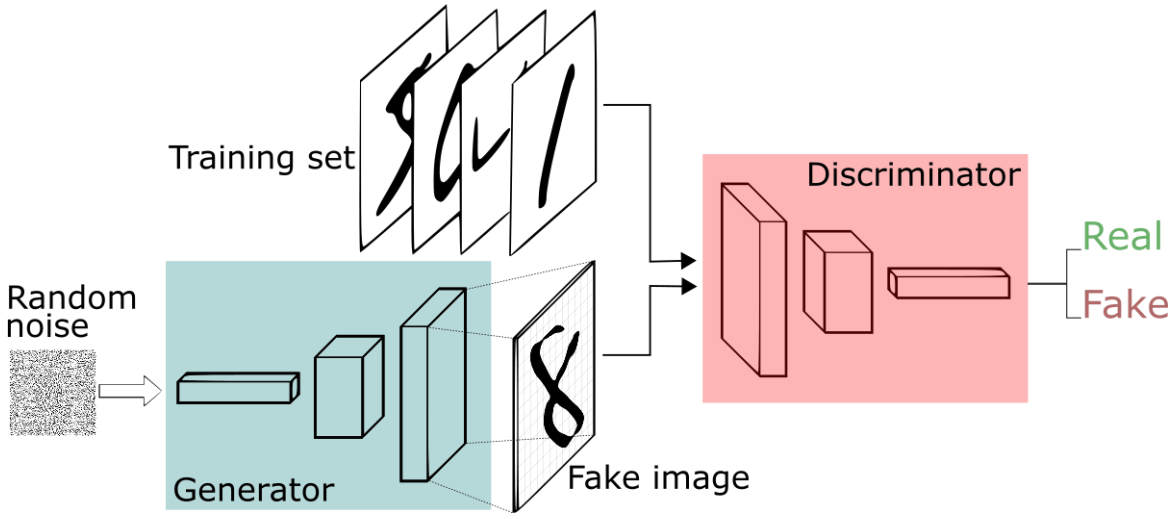

Рисунок 1.8 — Складові частини GAN нейромережі

Основою архітектури для генератора та дискримінатора є архітектура згорткової (конволюційної) нейромережі. Така архітектура передбачає використання математичної операції згортки для зміни зображень - зміни значень матриць зображення. Для операції згортки використовують фільтр або декілька фільтрів, які проходячи по зображенню, обчислюють нові значення пікселів. Матрицею вагових коефіцієнтів у згорткових нейромережах є матриця фільтру, яка змінюється під час навчання. Пряме проходження здійснюється через обчислення нового значення пікселів через операцію згортки. При зворотньому проходженні обчислюються похідні вихідної матриці з урахуванням вхідної матриці, фільтрів та зміщення.

GAN — нейромережі мають пряме та зворотне проходження, аналогічно зі згортковою нейромережею. Навчання генератора та дискримінатора проводяться окремо: спочатку навчається тільки дискримінатор (рис.1.9), після цього навчається GAN нейромережа, дискримінатор залишається фіксованим, тобто навчається тільки генератор (рис.1.10). На вхід генератор отримує латентний вектор (вектор шуму) - це матриця розміру кількість латентних точок на кількість зображень у батчі, яка містить в собі значення нормального розподілу. Входом до

дискримінатора є згенероване генератором зображення та зображення з навчального датасету.

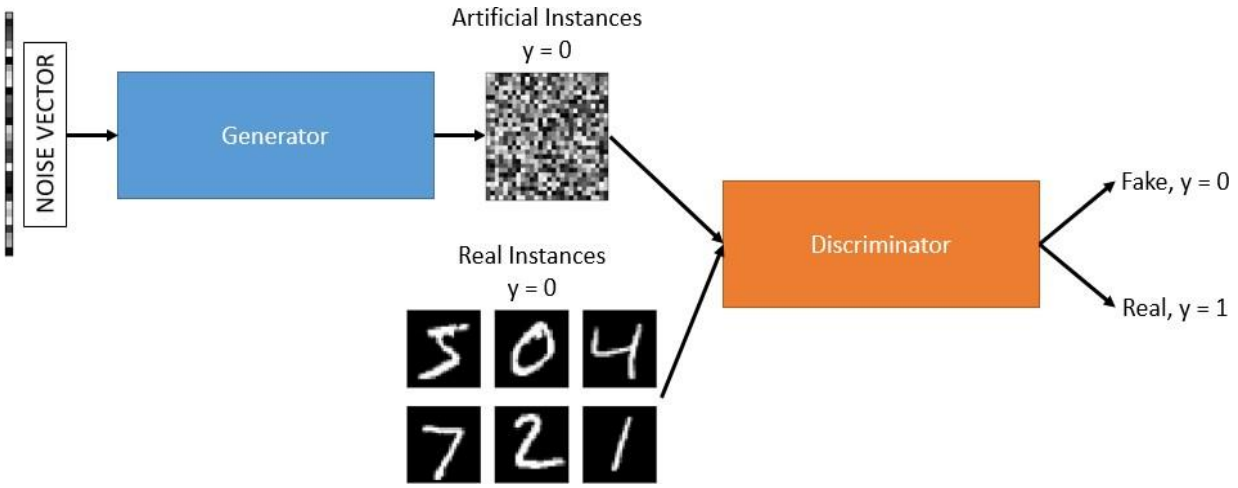

Рисунок 1.9 — Принцип навчання дискримінатора

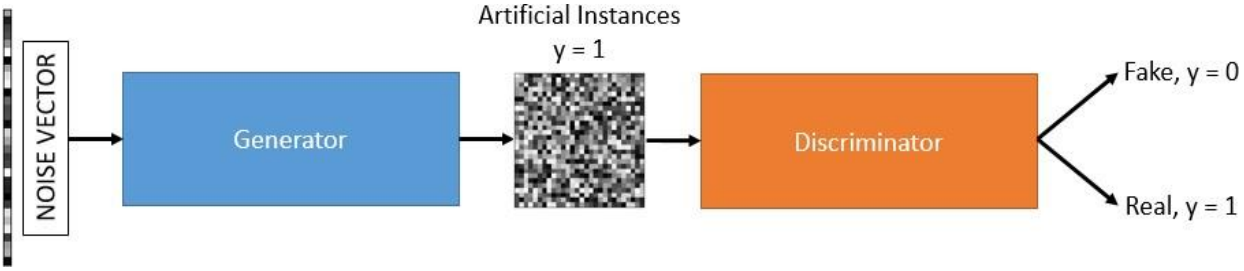

Рисунок 1.10 — Принцип навчання GAN нейромережі

GAN— нейромережі дуже складні у навчанні не тільки через складну архітектуру, але й через дві основні проблеми в період навчання. Перша проблема — Mode Collapse (укр. режим згортання або коллапсу), коли генератор генерує однакові або маленьку вибірку з зображень (рис.1.11).

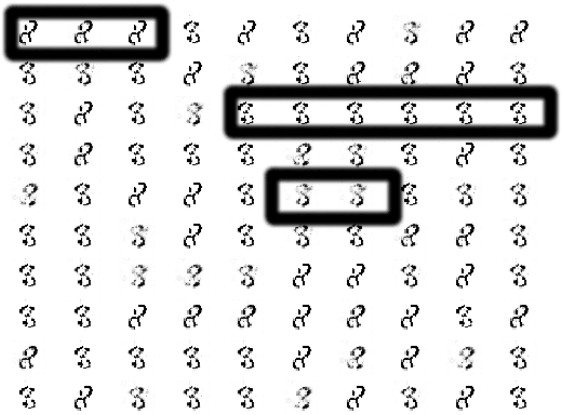

Рисунок 1.11 — Приклад Mode Collapse

Друга проблема - Convergence Failure (укр. помилка конвергенції) функція втрат є нестабільною, через що покращення якості генерації не відбувається (рис.1.12). При розробці вдалося вирішити ці проблеми.

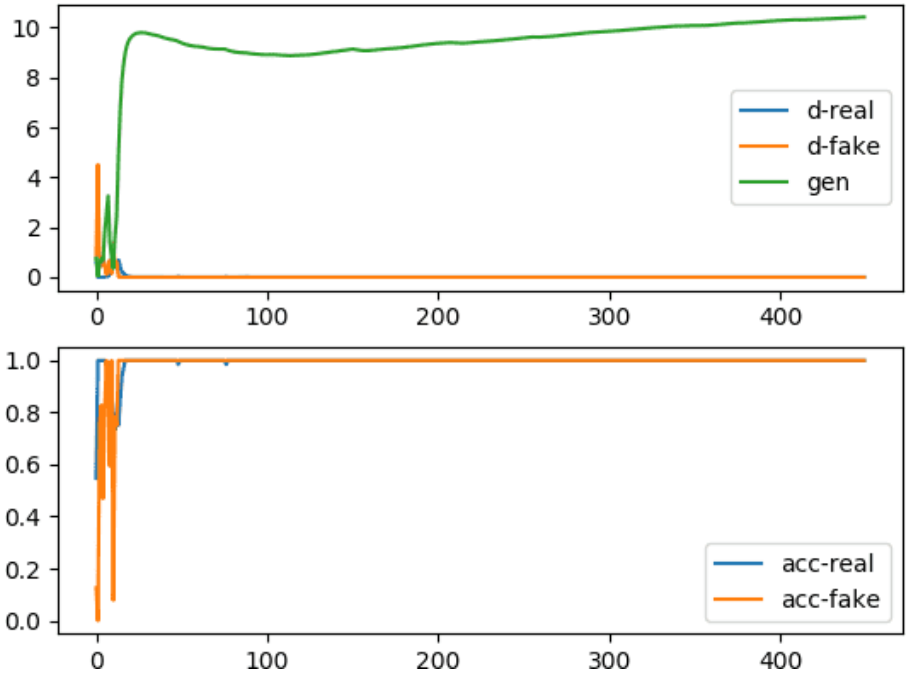

Рисунок 1.12 — Функція втрат (верхня) та функція точності (знизу)

1.5 Практики подання даних до нейромереж та їх навчання

Навчання нейромереж — це метод підвищення якості роботи нейромережі шляхом автоматичного налаштування параметрів нейромережі. Такими параметрами є вагові коефіцієнти. Під час навчання нейромережа змінює власні параметри доти, поки не знайде глобальний мінімум функції втрат. Часто нейромережі знаходять локальні мінімуми, але досягти глобального мінімума не можуть. Вирішенням цієї проблеми стали різні види оптимізаторів — алгоритмів розрахунку функції втрат, що дозволяють вийти з точки локального мінімума. Є кілька видів оптимізаторів: Gradient Descent, Stochastic Gradient Descent, Adagrad, Adadelta, RMSprop, Adam. Оптимізатор Adam вважається найкращими для більшості задач, де використовуються нейронні мережі. Він базується на двох попередніх оптимізаторах: Gradient Descent with Momentum, який заснований на методі моментів (рис.1.13), який визначає напрямок руху та RMSprop, який використовує експоненціальне зважене середнє щоб визначити швидкість навчання (швидкість руху) в момент часу для кожної ітерації.Зі спадом функції втрат, функція точності зростає, це пов’язано з тим, що нейромережа більш точно виконує своюзадачу.

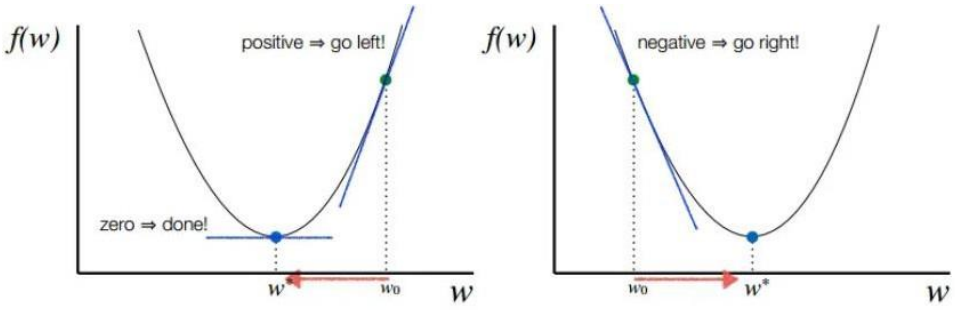

Рисунок 1.13 — Напрямок руху при позитивному та негативному значенні похідної

Нейромережу можна навчити виконувати операції над одним чи кількома видами даних, але з однієї області, неможливо навчити виконувати

однаково одне завдання, як і отримати повністю однакові результати. Наприклад, для задачі класифікації зображень тварин за класами “собака” та “кішка”, навчальні дані повинні містити зображення собак та кішок та мітки класів. В задачах бінарної класифікації, як задача класифікації собак та кішок, мітки класа можуть бути представлені двома значеннями: 0 — собака, 1 — кішка, або собака — собака, кішка — кішка. Вид типу міток залежить вибір функції втрат.

Одним з алгоритмом навчання нейромережі є навчанням не по всім даним одразу, а по батчам (групах) даних. Такий підхід дозволяє оновлювати вагові коефіцієнти після кожного батча, замість оновлення після проходу по всім даним. Це дозволяє пришвидшити навчання. Кількість даних у кожному батчі залежить від розміру тренувальної вибірки. Зазвичай розмір батчу обирають як двійку в степені: 16…256…1024.

1.6 Алгоритми підготовки навчальних даних

Один з кроків до покращення якості навчання нейромережі - це грамотна підготовка тренувальних (навчальних) даних, на яких нейромережа буде вчитися. У даній роботі використані два типи даних: текст тазображення.

1.6.1 Підготовка текстових даних

Текстові дані використовуються для навчання нейромережі для задачі класифікації. Перед подачею даних на навчання зазвичай використовують такий алгоритм:

1. Очищення даних : видалення тексту, що не несе смислової нагрузки для поточної задачі: цифри, посилання,

прийменників, сполучників, часток, якщо вони існують у мові тренувальних даних, знаків пунктуації

2. Лематизація — це заміна слова його лемою.
3. Токенізація — це розділення тексту на слова (токени), які переводяться у машинний (числовий) вектор.
4. Створення ембедінгу — це обчислення вектору відношення між словами. У роботі використаний Glove ембедінг — це різновид ембедінгу, що розраховує вірогідність спільного повторення слів. Завдяки ембедінгу вектор тексту, отриманий після токенізації, містить вектор відношень між словами: споріднені слова мають меншу різницю положення, ніж менш споріднені.

1.6.2 Підготовка зображень

Аналогічно з текстовими даними, зображення теж проходять обробку. Зображення у машинному вигляді є матрицями, у випадку кольорового зображення, або однією матрицею, у випадку чорно-білого зображення. Максимальне значення у обох випадках — 255, мінімальне — 0. Зображення у більшості випадках використовують у нейромережах, де присутня операція згортки — математична операція де за допомогою фільтра (матриці маленьких розмірів), у етапі проходки фільтром по зображенню вираховується нове значення пікселя. На швидкість обробки зображення впливає значення пікселів: чим менше різниця між пікселями, тим швидше працює згортка, розмірами зображення — чим більше зображення, тим повільніше працює згортка, кількість кольорових каналів (матриць) також впливає на швидкість, кольорові зображення обробляються повільніше заодноканальні (чорно - білі).

Для зображень алгоритм полягає у:

1. Нормалізація зображень — приведення значень всіх пікселів у діапазон [-1, 1], [0, 1].
2. (Вибіркова дія) Приведення до визначених розмірів.

1.7 Фізичні аспекти генерування та перетворення Mel спектрограм

Для перетворення спектрограм у музичну послідовність використовують алгоритми обробки звукових сигналів.

Звукові сигнали — це зміна тиску повітря, який змінюється впродовж часу. Формула трансформації Фур'є (FT) — це математична формула, що дозволяє розділити звуковий сигнал на окремі частоти (рис.1.14),перетворення площини сигналу з часової на частотну. Алгоритм, що виконує трансформацію Фур'є називають fast Fourier transform (FFT) (укр. Швидка трансформація Фур'є)

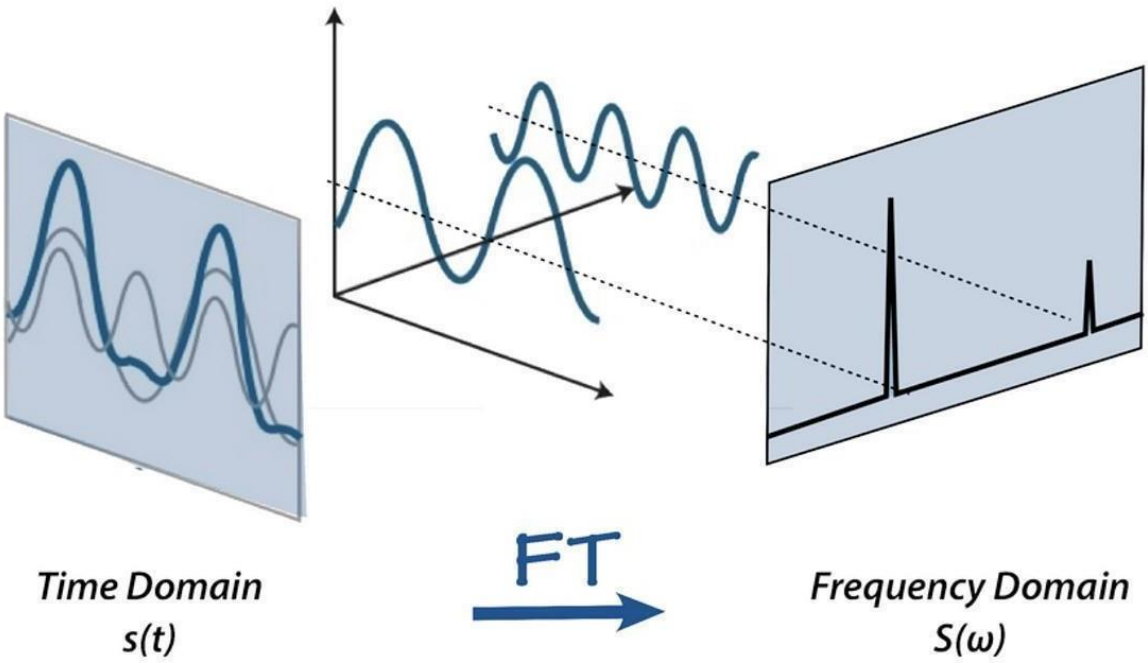

Рисунок 1.14 — Перетворення сигналу з часової у частотну площину

Спектрограми — це вид представлення звукових сигналів, їх частот або амплітуд, що змінюються з часом, на відміну від FFT алгоритму, який не має часової складової. Візуально спектрограми можна уявити як групу звукових сигналів, накладених одна на одну, оброблених за допомогою FFT, кожна з яких є окремою частотою, що змінюються впродовж часу. Такі спектрограми також називають магнітудними.

Mel спектрограм — різновид спектрограм, де ось частот (ось y) замінюється на мел шкалу. Мел шкала — це нелінійна математична формула, що виконує перетворення звукових висот таким чином, щоб відстань між різними висотами була однакова. Перевагою такої шкали є точніше сприймання людським вухом. Mel спектрограми частіше використовують у роботі з нейромережами, ніж магнітудні спектрограми. Порівняння видів представлення сигналів наведено на рис. 1.15.

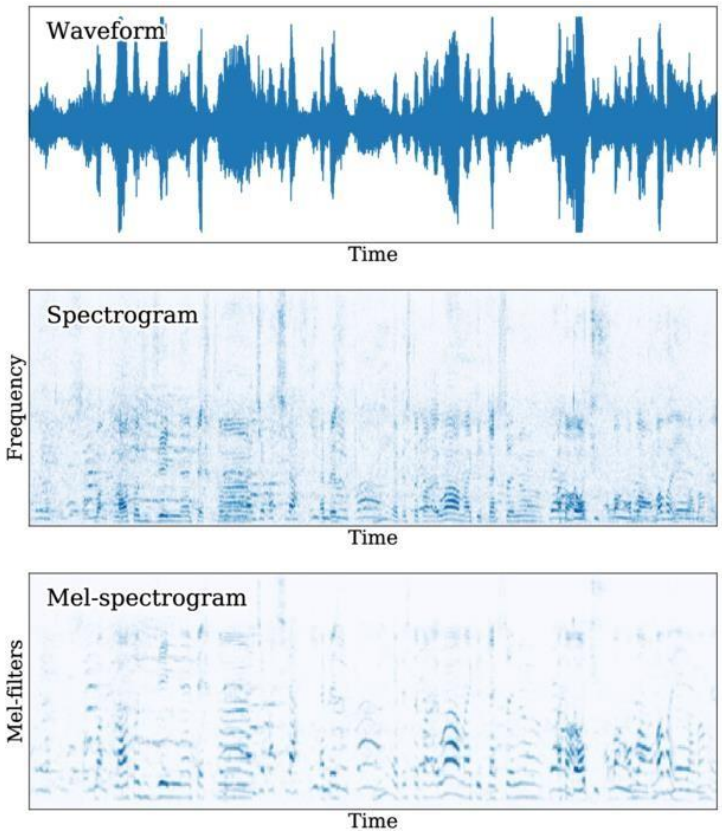

Рисунок 1.15 — Перетворення сигналу з часової у частотну площину

Задача перетворення спектрограм у музику виконується через алгоритм Гріффіна - Ліма — це метод відновлення звукового сигналу за спектрограмою. Алгоритм, на основі цього методу полягає у ітераційному підборі сигналу, створення спектроми цього сигналу до тих пір, поки створений (реконструйований) сигнал не буде максимально близький до оригінального (рис.1.16). Алгоритм можна використовувати в режимі реального часу через його ефективність.

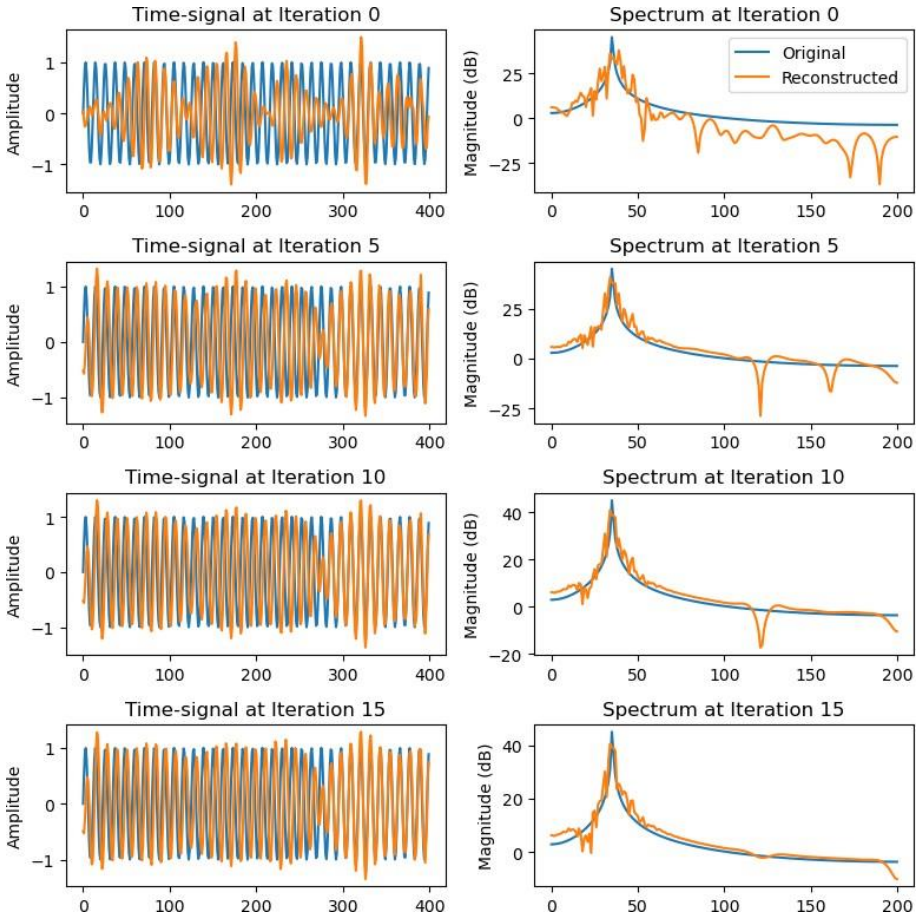

Рисунок 1.16 — Ефективність алгоритму впродовж ітерацій

Однією із особливостей алгоритма є те, що його можна використовувати в режимі реального часу через його ефективність.

Однією з задач при перетворенні спектрограми у музичну послідовність є вибір оптимального значення для частоти дискретизації —

кількість значень у одній секунді музичної послідовності. Чим більше значення, тим більше різних звуків (більший діапазон) буде у музиці.

1.8 Відповідність жанрів музики до емоції

Музика, як і людська мова, передає декілька параметрів, таких як емоція автора, ідея, емоція слухача. На емоцію людини впливають такі музичні показники як мелодія, ритм та гармонія. Для того, щоб розробити коректний алгоритм потрібно звернутися до психології, яка пояснює чому іяк конкретну емоцію викликає відповідний музичний жанр. Вивчення того, як музика впливає на емоцію людей розпочали відносно нещодавно, якщо порівнювати зі схожим вивченням впливу тексту. Попри це, вже знайдені основні емоції, що викликаються різними жанрами музики. Наприклад, джаз та диско викликають емоції ностальгії, тоді як метал співвідносять до емоції злості.

Використання знань з цієї області допоможуть розробити більш точніший та науково — грунтовний алгоритм для підбору жанру музики для відповідної емоції. Знання в цій області використовуються для створення функції, що буде підбирати жанр музики до емоції, отриманої з класифікатора.

1.9 Обгрунтування засобів розробки

1.9.1 Середовище виконання

Goggle Collaboratory — це браузерне середовище виконання Python проектів з завантаженими основними бібліотеками у вигляді записника, що дозволяє розміщувати не тільки код, але й текст, зображення, тощо. Оскільки

Goggle Collaboratory є браузерним середовищем виконання, в ньому присутня функція збереження файлу через визначений проміжок часу. В умовах розробки таких складних алгоритмів як нейромережі, ця функція убезпечує він ававійної зупинки виконання та незбереження файла врезультаті. Основною перевагою цього середовище виконання є безкоштовна обчислювальна потужність, код виконується на віртуальній машині, прив'язаній до акаунту користувача, окрім цього можна використовуватиTPU або GPU прискорювачі на Tesla K80 GPU та 12 ГБ RAM. Також Goggle Collaboratory підтримує різні версії мови Python , від версій мови залежить нетільки наявність функцій, але і їх оптимізація.

1.9.2 Мова виконання

Розробка нейромереж є циклічним процесом. Програміст може розробляти та покращувати одну або одночасно навчати декілька моделей. Сучасні процеси потребують швидкої та якісної розробки нейромереж. Пришвидшення процесу розробки, тестування та впровадження нейромереж є заслугою саме мови Python. У мові присутні бібліотеки, які не тільки пришвидшують підготовку даних, створення архітектури нейромереж та візуалізацію результатів, але є "мостом" між мовами.

Мова програмування Python є основною мовою для розробки алгоритмів та глибокого навчання, через функції, оптимізовані для роботи з матрицями великого розміру. Ця мова є об'єктно - орієнтованою мовою високого рівня з динамічною семантикою. Вона є синтаксично простою та має одну з найбільших колекцій бібліотек. Окрім оптимізації Pythonдозволяє швидко, порівняно з іншими мовами, візуалізовувати результати алгоритмів. що пришвидшує процес розробки, оскільки візуалізовані дані в більшості випадках швидше сприймаються людиною, ніж числові. На відміну від інших мов програмування, не має залежності від платформи, на якій виконується код.

1.9.3 Бібліотеки

Для виконання роботи використані такі бібліотеки: основні (Numpy, Pandas,Matplotlib), бібліотеки для роботи з текстом (NLTK), бібліотека машинних алгоритмів (Sklearn), бібліотеки глибокого машинного навчання (Tensorflow, Keras, OpenCV) та бібліотек обробки звукових сигналів Librosa. Основні бібліотеки:

- NumPy — Бібліотека з відкритим кодом, що підтримує складні математичні операції над багатовимірними масивами та включає в себе математичні функції. Numpy є найпоширенішою бібліотекою для роботи з матрицями через основні концепції, а саме: індексацію масивів через маски, скалярні координати або інші масиви, векторизація групи об’єктів, трансляція (англ. broadcasting) — оптимізоване перемноження матриць, скорочення розмірності матриць.Numpy має власні структури з даних, основна з яких — це масиви NumPy(NumPy arrays) є аналогом листів у Python, але більш швидкі та економніші. Дозволяє створювати масиви з одним значенням: нульові, одиничні, пусті, тощо — такі масиви часто використовують для зберігання та заповнення через більшу швидкість та можливість застосовувати операції лінійної алгебри.
- Pandas — Бібліотека з відкритим кодом для аналізу даних. За допомогою неї можна швидко експортувати дані табличних типів у структуру DataFrame, яка використовується для маніпуляції над даними: видалення рядків, дублікатів, вставлення, очищення пропущених даних та даних невизначеного типа, обчислення параметрів математичної статистики для подальшого аналізу, розділення даних за групами.
- Matplotlib — Бібліотеки для високоякісною візуалізації даних. Є основним інструментом через легку інсталяцію та синтаксис.Основним сабмодулем є plt модуль для створення та редагування

двовимірних графіків. Matplotlib дозволяє створювати матриці з графіків, що є необхідним інструментом при візуалізації багатовимірних даних.

Набір бібліотек для обробки природної (людської) мови NLTK (Natural Language Toolkit) — основний інструмент для символьної обробки завдяки зрозумілому синтаксису та потужним функціям. Складається з бібліотек для виконання основних операцій над текстовими даними, а саме: задачі класифікації, токенізації, формування основи, тегування, синтаксичного, семантичного аналізу, та інтерфейсів корпусів текстів, та лексичних ресурсів. У задачі класифікації використовуються інтерфейс для видалення стоп - слів та лематизації.

Sklearn — бібліотека алгоритмів машинного навчання з відкритим кодом. Містить такі основні алгоритми машинного навчання як регресія, кластеризація, класифікація. Окрім них бібліотек містить такі важливі функція для підвищення якості алгоритмів як передобробка даних та зменшення розмірності даних та алгоритми вилучення ознак з даних, алгоритми розділення даних на вибірки, тощо, алгоритми енкодування та векторизації. Sklearn легко імпортується та має зрозумілий синтаксис, розроблені алгоритми є універсальними та можуть використовуватися в більшості задач машинного навчання.

Бібліотеки глибокого машинного навчання містять набір інтерфейсів та функцій, за допомогою яких можна створювати, тестувати, та валідувати результат нейромереж. Основною бібліотекою при побудові нейромереж, залежно від задачі, яку буде вирішувати нейромережа може бути Tensorflow PyTorch. Для цієї роботи обрана Tensorflow, оскільки має більш ширший вибір потрібних функцій. Вона має більш зрозумілий синтаксис та більшу гнучкість, ніж PyTorch. Tensorflow має безліч функцій для обробки тренувальних даних, побудови, тестування та імплементації нейромереж до інших систем. Завдяки гнучкості бібліотеки, розбудова та змінення

архітектури займає мінімальну кількість часу, що доволяє приділити навчанню більше часу.

Бібліотека Keras може використовуватися як самостійна бібліотека, але частіше за все використовується як набудова до Tensorflow. Перевагою Keras є побудова синтаксису навколо такої практики, як зменшення когнітивного навантаження на людину, яке досягається шляхом використання людської мови як основного синтаксису, зменшення кроків, необхідних для розробки та простого API, логічної архітектури та повідомлень про помилки. Основними API є Functional та Sequential, вони є “скелетом” нейромереж — зв’язують шари нейромереж лінійно (послідовно) або нелінійно відповідно.

OpenCV — це бібліотека області комп’ютерного навчання. Містить алгоритми та функції для обробки, зміни та трансформації зображень. Бібліотека має багато складних функцій на кшталт тегування об’єктів на зображеннях чи відео, обробка зображень для генеративних нейромереж тощо. У цій роботі бібліотека використовувалася для обробки тренувальних даних та візуалізації результатів.

Бібліотека Librosa є найбільшою бібліотекою для обробки звукових сигналів. Обробка та аналіз звукових сигналів є складним процесом, завдяки бібліотеці, яка включає в себе функції для аналізу, створення та функціонування зміни музики та аудіо, можна їх прискорити. Для цієї роботи бібліотека використовується для візуалізації спектрограм та перетворення спектрограм у музику за допомогою визначеного у бібліотеці алгоритма Гріфіна - Ліма.

1.9.4 Датасети

Більша частина алгоритмів машинного навчання потребують проведення такої операції, як навчання. Навчання алгоритму машинного навчання проводиться для того, щоб алгоритм зрозумів патерни у даних, та в подальшому міг самостійно працювати з наданими даними: класифікувати,

обробляти, тощо.Датасети — це набір даних для навчання. Від обраного датасету залежить швидкість та якість обробки алгоритму. Часто розробники стикаються з тим, що алгоритм машинного навчання не може виконати задачу з обраним датасетом не через хибну архітектуру алгоритма, а саме через погано сконструйований датасет. Окрім якості датасету, важливим параметром є його розмір: недостатня кількість даних може призвести до неможливості навчання, тоді як завелика призводить до уповільнення алгоритму.

Для виконання кваліфікаційної роботи обрано датасети двох типів. Перший тип — це датасет текстових даних для задач класифікації. Він містить речення, зібрані з книг, статей та інших документів, кожне речення характеризує одна з шести емоцій. Другий тип — це датасет з музичних спектрограм, тип даних — зображення. Містить 1000 спектрограм, розділених по відповідним 10 жанрам музики, кожен жанр містить 100 спектрограм відповідно.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ

2.1 Опис програмної реалізації

2.1.1 Загальна схема ПЗ

Програмне забезпечення складається з трьох основних блоків: блок класифікації тексту, блок генерації музичних спектрограм, блок перетворення музичних спектрограм у музику. Кожен з блоків міститься у окремому файлі.

Загальний опис ПЗ (рис. 2.1) : введений користувачем запит надходить до першого блоку та класифікується за допомогою класифікатора за класом емоції. Результатом виконання першого блоку є лист, який складається з назви емоції та сили цієї емоції у числовому вигляді. Другий блок отримує результат першого блоку та через спеціальну функцію підбирає жанр музики до отриманої емоції, після цього запускається процес генерації спектрограми у цьому музичному жанрі. Згенерована спектрограма потрапляє до спеціальної функції, яка за допомогою алгоритма Гріффіна - Ліма перетворює спектрогаму у музику.

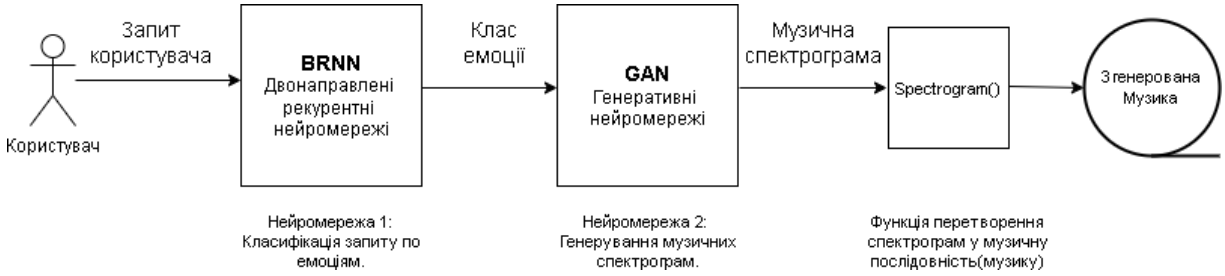

Рисунок 2.1 — Загальна схема ПЗ

Максимальна кількість символів у запиті не повинна перевищувати 229 символів. Запит користувача є емоційно забарвленим текстом або текстом, що описує емоцію, яку користувач бажає почути у згенерованій музиці. Користувач може ввести один або декілька запитів.

Згенеровану музику користувач може прослухати та зберегти у форматі wav. За бажанням користувач може також переглянути згенеровану спектрограму.

2.1.2 Програмна реалізація задачі класифікації.

Задача класифікація включає в себе наступні основні функції: normalize_text(), encoder, tokenizer, ембедінг, модель класифікатор (Classificator). Схема наведена на рис. 2.2.

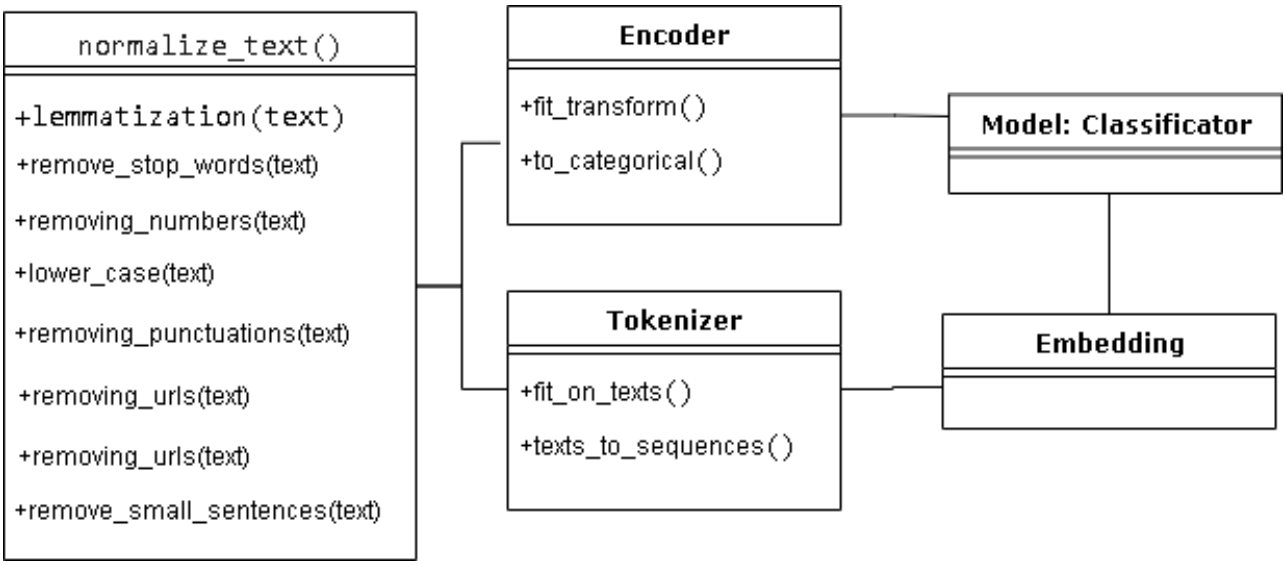

Рисунок 2.2 — Схема класів для задачі класифікації

Під час розробки архітектури класифікатора протестовано два підходи, серед яких обрано архітектуру з найвищою точністю. Такою архітектурою є двонаправлені рекурентні нейромереж з модулем довготривалої - короткотривалої пам’яті. Архітектура алгоритму з нижчою якістю та його результати наведені в розділі 3.

Для задачі були обрані та протестовані два датасети, обраний датасет, на якому була найвища якість навчання. При розробці стабільної архітектури були проведені тестування комбінацій різних параметрів, детальніше у розділі 3.

Обраний для цієї задачі датасет розділений на три вибірки: тренувальні, тестувальні та валідаційні дані. Загальна кількість становить 20000 речень, розділених на 6 класів емоцій (рис. 2.3). На діаграмі наведені відношення класів, числова інтерпретація відношень така: joy (укр. радість) — 6740 речень, sadness (укр. смуток) — 5793 речень, anger (укр. злість) — 2703 речень, fear (укр. страх) — 2369 речень, love (укр. любов) — 1630речень, surprise (укр. здивування) — 713 речень. Балансування класу не проводилося через малу кількість тренувальних даних.

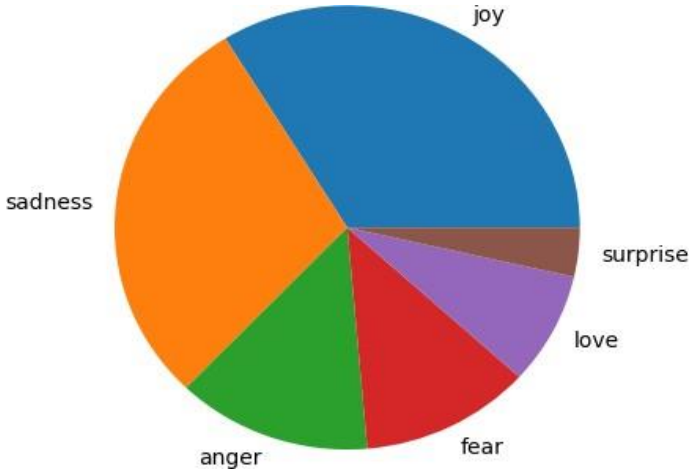

Рисунок 2.3 — Розподіл класів емоцій

Для зручності дані перерозподілені та тренувальну та тестові вибірки. Перед подачею до навчання дані були перевірені та оброблені, збережені у спеціальну структуру даних DataFrame (рис. 2.4).

	content	sentiment
0	i didnt feel humiliated	sadness
1	i can go from feeling so hopeless to so damned...	sadness
2	im grabbing a minute to post i feel greedy wrong	anger
3	i am ever feeling nostalgic about the fireplac...	love
4	i am feeling grouchy	anger

Рисунок 2.4 — DataFrame тренувальних даних

Перевірка даних складається з видалення дублікатів та нульових значень через функцію duplicated_all(). Процес обробки текстових даних складається з 4 частин: очистка даних, лематизація, токенізація (енкодування для назв емоцій) та створення ембедінгу.

Очистка даних включає в себе видалення стоп - слів, цифр, посилань, знаків пунктуації, приведення до нижнього регістру. Для цього розроблена група функцій, які викликаються через основну функцію, на вхід приймає DataFrame. Аналогом цієї функції є normalize_sentence() для презентації результатів очистки. Для заміни слів їх основою є два алгоритми — стемінг та лематизація, обрано другий алгоритм, оскільки він допомагає збільшити точність нейромережі. Очистка та лематизація виконуються у функції normalize_text(). Результатом є очищений текст та лематизований текст у структурі DataFrame, що складається з двох стовбців: тексту та емоції та 19948 рядків (варіантів тексту). Приклад роботи функції очистки та летатизації рис.

	content	sentiment
19946	truly feel passionate enough something stay tr...	joy
19947	feel like wanna buy cute make see online even one	joy

Рисунок 2.5 — Приклад даних після очищення та лематизації

Токенізація та енкодування проведені за допомогою токенізатора та енкодера з бібліотеки Keras.

Створення матриці ембедінгу за допомогою файлу 200 - вимірного ембедінгу типу Glove. Файл для створення матриці можна зробити самостійно, в такому разі час на виконання збільшиться на 2 - 3 години, тому для пришвидшення обрано вже готовий файл. Спеціальна функція створює матрицю, шляхом співставлення слів, що є у файлі та тренувальних даних, у випадку, якщо слово не знайдено, його помічають нульовим значенням.

Розробка та навчання нейромережі проводилося як ітераційний процес: розробка версії архітектури — тестування — збереження архітектури та результатів — розробка нової версії. Протестовано 20 наборів параметрів нейромережі та обрано найкращий. Він складається з 3 двонаправлених LSTM шарів та повного шару — tf.keras.layers.LSTM та tf.keras.layers.Dense відповідно. Архітектура класифікатора представлена на рис. 2.6. Тип моделі обрано Sequential, оскільки шари йдуть послідовно один за одним. Шар ембедінгу в процесі тренування нейромережі не тренується, оскільки матриця вже створена у попередньому пункті. Рекурентні нейромережі є повільними, тому для запобіганню перенавчання використана функціяранньої зупинки з бібліотеки Keras, вона зупиняє навчання як тільки функція точності для тестової вибірки починає спадати. Як оптимізатор обрано Адам оптимізатор, оскільки він допомагає запобігти обнуленню похідних, для функції втрат обрано категоріальну крос - ентропію.

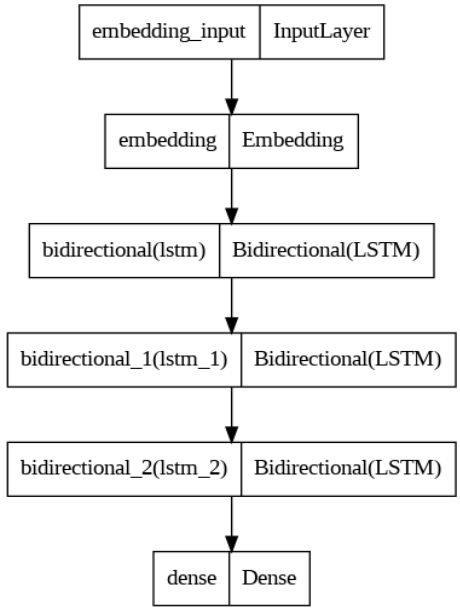

Рисунок 2.6 — Схема архітектури нейромережі - класифікатора

Така архітектура є оптимальною для пристрою, на якому навчалася, під час навчання використовувалося 12 ГБ пам’яті.

Для повного навчання необхідно близько шести годин - шість епох. Дані розділені по батчам, у кожному батчі по 256. Тренувальні дані татестувальні розділені у пропорції 20 на 80.

В результаті навчання досягнута якість на тренувальних даних становить 96%, на тренувальних — 93%. Модель нейромережі, енкодера та декодера збережені у вигляді файлів формату .h5 для використання в інших програмах. Результат навчання розміщено на рис. 2.7. Для повного навчання необхідно близько 6 годин, щоб уникнути перенавчання доданий метод EarlyStopping(), що зупиняє навчання коли функція втрат на валідаційних даних починає зростати.

```
Epoch 1/9
55/55 [==============================] - 1953s 35s/step - loss: 1.2109 - accuracy: 0.5412 - val_loss: 0.6437 - val_accuracy: 0.7838
Epoch 2/9
55/55 [==============================] - 1862s 34s/step - loss: 0.4976 - accuracy: 0.8244 - val_loss: 0.2914 - val_accuracy: 0.8922
Epoch 3/9
55/55 [==============================] - 1849s 34s/step - loss: 0.2509 - accuracy: 0.9029 - val_loss: 0.2170 - val_accuracy: 0.9058
Epoch 4/9
55/55 [==============================] - 1849s 34s/step - loss: 0.1712 - accuracy: 0.9246 - val_loss: 0.1649 - val_accuracy: 0.9252
Epoch 5/9
55/55 [==============================] - 1871s 34s/step - loss: 0.1188 - accuracy: 0.9448 - val_loss: 0.1524 - val_accuracy: 0.9280
Epoch 6/9
55/55 [==============================] - 2015s 37s/step - loss: 0.1049 - accuracy: 0.9487 - val_loss: 0.1561 - val_accuracy: 0.9313
Epoch 7/9
55/55 [==============================] - 2025s 37s/step - loss: 0.1044 - accuracy: 0.9491 - val_loss: 0.1566 - val_accuracy: 0.9303
Epoch 8/9
55/55 [==============================] - 2020s 37s/step - loss: 0.0935 - accuracy: 0.9546 - val_loss: 0.1402 - val_accuracy: 0.9322
Epoch 9/9
55/55 [==============================] - 2022s 37s/step - loss: 0.0819 - accuracy: 0.9614 - val_loss: 0.1472 - val_accuracy: 0.9318
```

Рисунок 2.7 - Результати навчання

На рис. 2.8 зображені функції втрати та функція точності нейромережі. По графікам видно невелике перенавчання, яке не впливає на якість класифікації.

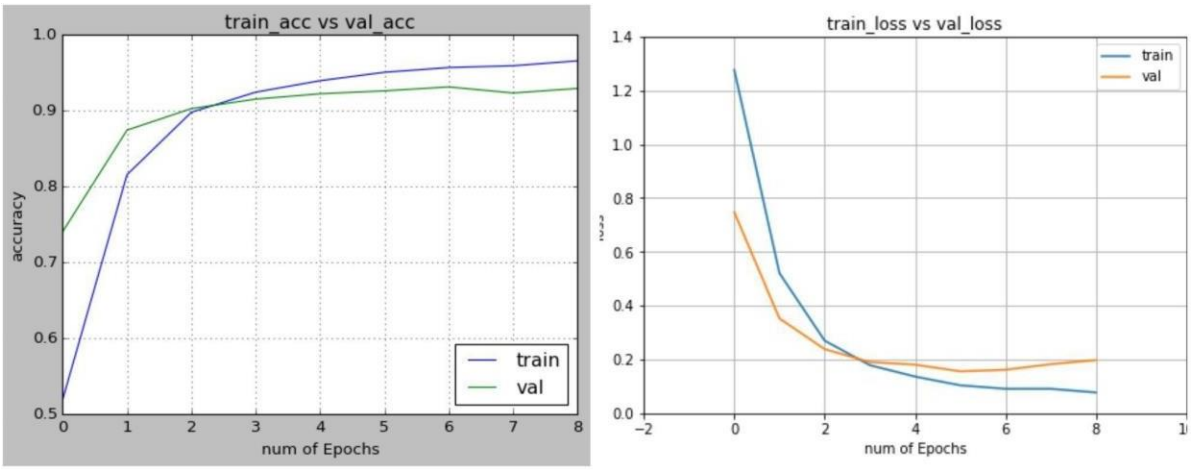

Рисунок 2.8 — Графік функції втрат (зліва) та точності (справа)

Вихідним даними є тип даних лист, що містить назву емоції та силу її впливу. Має наступний вигляд: (клас емоції: назва), (сила емоції: значення), межі значення від 0.4 до 0.99.

2.1.3 Програмна реалізація задачі генерації музичних спектограм

Під час розробки архітектури GAN - нейромережі використовувався підхід, запропонований командою GANSynth, запропонований командою Google AI в 2019 році.

Аналогічно з задачею класифікації, дані перед навчанням нейромережі — генератора потрібно підготувати. Обраний датасет складається зі спектрограм у десяти музичних жанрах. Схема основних функцій представлена на рис. 2.9.

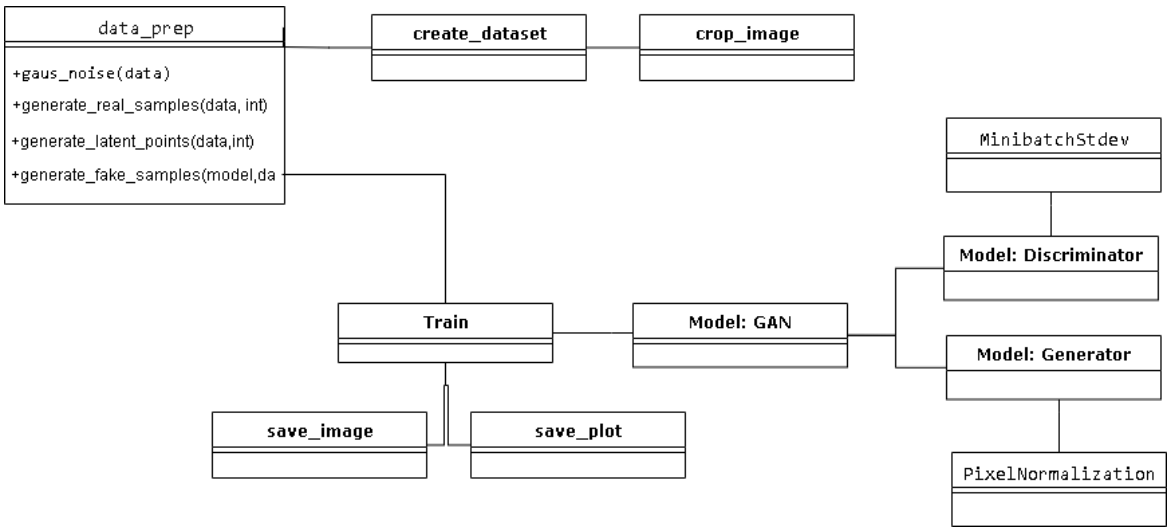

Рисунок 2.9 — Схема основних функцій

Підготовка складається зі зменшення розміру зображення та переформатуванню у одноканальне (чорно - біле) зображення. Оригінальний датасет містить кольорові зображення у розмірі 400 на 500 пікселів у білій рамці. Для видалення рамки з зображення створена функція crop_image(). Після видалення розмір зображення становить 300 на 400 пікселів. Для пришвидшення навчання зображення зменшені зі збереженням відносності

сторін до розмірі 128 на 192 пікселів та переформатовані з кольорових на чорно - білі зображення. Після тестування інших розмірів, такий розмір виявлено як оптимальний. Для отримання кращих результатів спектрограми нормалізовані у межах [-1, 1], вибір таких меж обумовлений функцією активації тангенс у генератора. Після обробки підготовлені спектрограми поміщені у відповідні масиви для кожного жанру, вибір масивів як структури даних обумовлений тим, що дані повинні бути розділені по батчам, тобто структура даних має бути індексованою, кожний масив збережений у форматі .npy.

Основою для частин нейромережі типу GAN — генератора та дискримінатора, є архітектура згорткових нейромереж. При розробці стабільної архітектури були проведені тестування комбінацій різних параметрів, детальніше у розділі 3. Стабільні архітектури генератора та дискримінатора, які показали найкращі результати представлені на рис. 2.10.

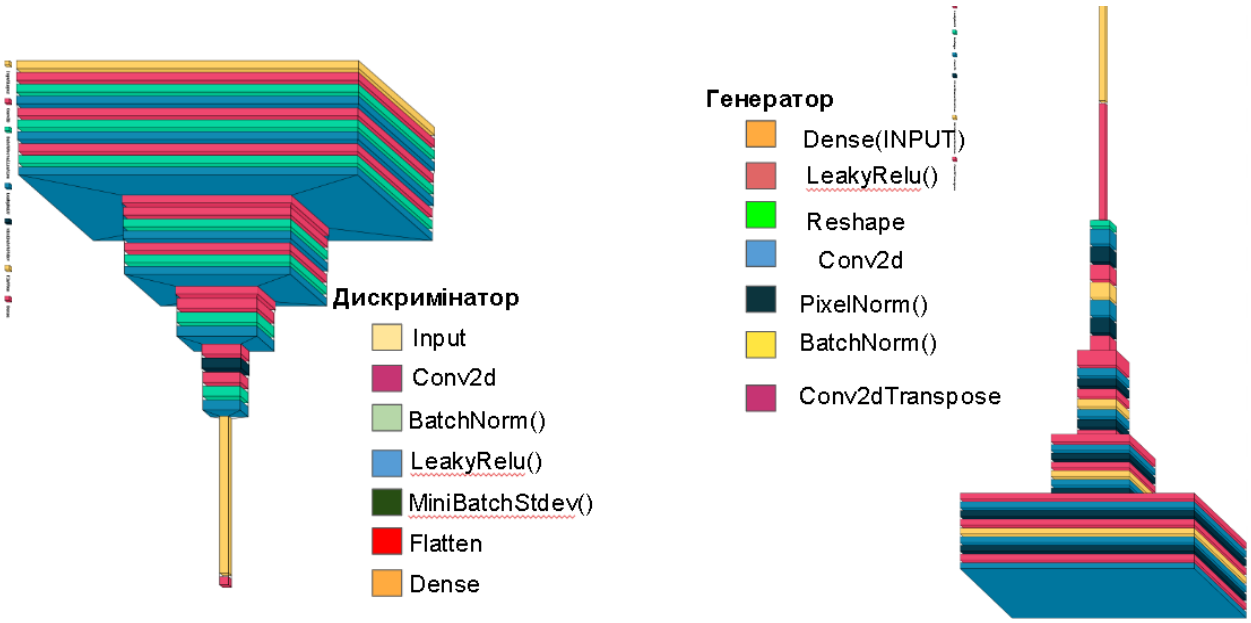

Рисунок 2.10 — Схема архітектури генератора та дискримінатора

Нейромережа GAN, аналогічно з іншими архітектурами навчання без вчителя, вчиться на датасеті, що містить спектрограму та мітку класу(0 — зображення згенероване генератором, 1 — справжнє) та створюється

безпосередньо під час навчання. У цій роботі функції для створення тренувального датасету визначені так: generate_real_samples() — розділяє датасет з справжніх спектрограм на батчі, розмір батчів вираховується з кількості навчальних епох, вибір номерів спектрограм, що потраплять у батч обирається рандомом, generate_latent_points() [12] — створює масив розміром кількість латентних точок на кількість даних у батчі з чисел у нормальному розподілі, generate_fake_samples() [12] — генерує фейкові спектрограми (спектрограми, згенеровані генератором) з латентного вектора попередньої функції, кількість зображень визначається аналогічно з кількістю даних у батчі. Функції generate_real_samples() та generate_fake_samples() повертають також мітки класів, для стабілізації навчання оригінальні мітки (0 та 1) замінені на такі: згенеровані спектрограми мають мітки 0 чи 0.1, що обираються рандомно, аналогічно справжні спектрограми рандомно обираються між 0.9 та 1.

Під час навчання виникли дві проблеми: нестабільне навчання та генерація однакових зображень. Для їх вирішення розроблені алгоритми, які застосовують для навчання ProgressiveGAN — нормалізація по пікселям та стандартне відхилення міні - батчів. Алгоритми розроблені через класи, PixelNormalization(Layer) [12] , який на вхід отримає шар нейромережі та MinibatchStdev(Layer) [12] , на вхід отримає також шар нейромережі. Результати генерації без цих алгоритмів розміщено у розділі 3.

Для з'єднання генератора та дискримінатора у одну нейромережу GAN розроблена функція define_gan(g_model, d_model), на вхід отримує моделі генератора та дискримінатора та поєднує у єдину нейромережу за допомогою Sequential API бібліотеки Keras.

Для спрощення архітектура кожна нейромережа навчається на власному жанрі музику, тобто цей пункт містить 10 різних нейромереж типу GAN.

Алгоритм навчання нейромережі розміщений у функції train(), під час розробки протестовано кілька варіантів та обраний алгоритм з найвищою

якістю. Функція має 2 цикли, перший цикл — це проходження навчання по епохам, другий — це проходження навчання по ітераціям. В кожній ітерації спершу генерується половина батчу (10 спектрограм) справжніх спектрограм, на яких навчається дискримінатор, потім генеруються генеровані спектрограми (10 спектрограм), на яких також навчається дискримінатор. Після навчання дискримінатора починає навчатися нейромережа GAN, під час якого, хоча GAN і складається з генератора та дискримінатора, навчається тільки генератор. Додатково розроблені функції для збереження згенерованих спектрограм, візуалізації графіків функцій втрат та точності. Через кожні дві епохи генеруються спектрограми для контролю їх якості, кожні п’ять епох візуалізуються графіки точності та втрат представлені на рис. 2.11. Оскільки дискримінатор навчався на двох датасетах, на функції втрат якість дискримінатора складається з двох частин - якості на справжніх та згенерованих спектрограмах.

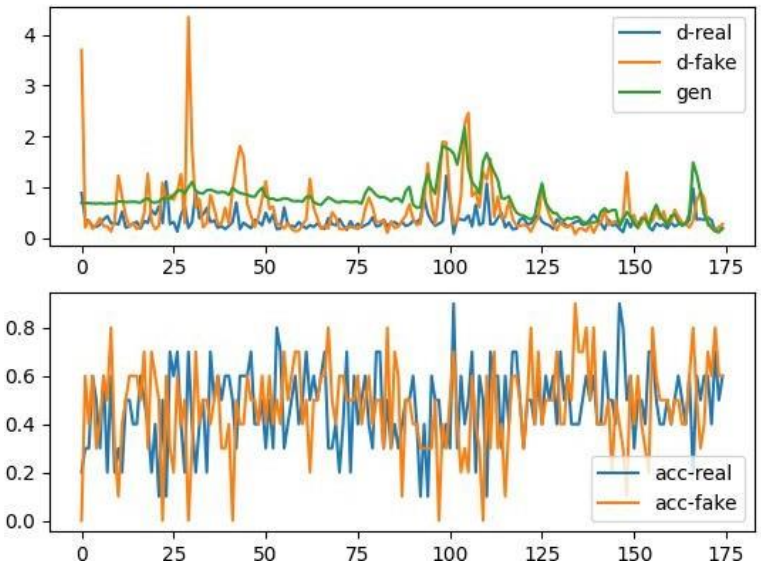

Рисунок 2.11 — Функції втрат (зверху) та функція точності (знизу)

2.1.4 Програмна реалізація процесу перетворення спектрограм на музику.

Реалізація розташована в окремому файлі. Перетворення Mel спектрограм у музику формату .wav розроблено на основі музичної бібліотеки Librosa. Перетворення здійснюється за допомогою алгоритма Гріффіна - Ліма.

Функція викликається одразу після отримання згенерованої спектрограми від GAN нейромережі. Отримана спектрограма зчитується у масив за допомогою бібліотеки OpenCV, після цього масив нормалізується до меж [-1,1], після чого викликається метод inverse.mel_to_audio() перетворення спектрограми у музику з бібліотеки Librosa. Перетворена музика зберігається у файлах поточної сесії у форматі .wav, який користувач може завантажити.

2.1.5. Функція відповідності жанру музики до емоції

Після того, як ПЗ класифікувало запит користувача за емоцією, викликається функція, що підбирає жанр музики, що найкраще описує отриману емоцію. ПЗ має 10 видів жанрів та 6 класів емоцій, тому деякі емоції описують один жанр. Переважаючим типом емоції негативні емоції (злість та страх), тому, щоб збалансувати відповідності емоцій до жанрів ці емоції мають по одному жанру відповідно, для емоції злість - метал, для емоції страх - класична музика. Інші емоції відповідають двом жанрам музики, вибір конкретного жанру залежить від сили емоції: якщо показник сили емоції є більшим за 0.6 обирається перший жанр, якщо менше — другий. Значенням обрано 0.6, а не 0.5, через дві причини: показник сили емоції починається з 0.4, “сильна” емоція має становити більше за 60% (0.6). Функція реалізовувала на основі match методу мови Python.

Діаграма розподілу приведена на рис. 2.12.

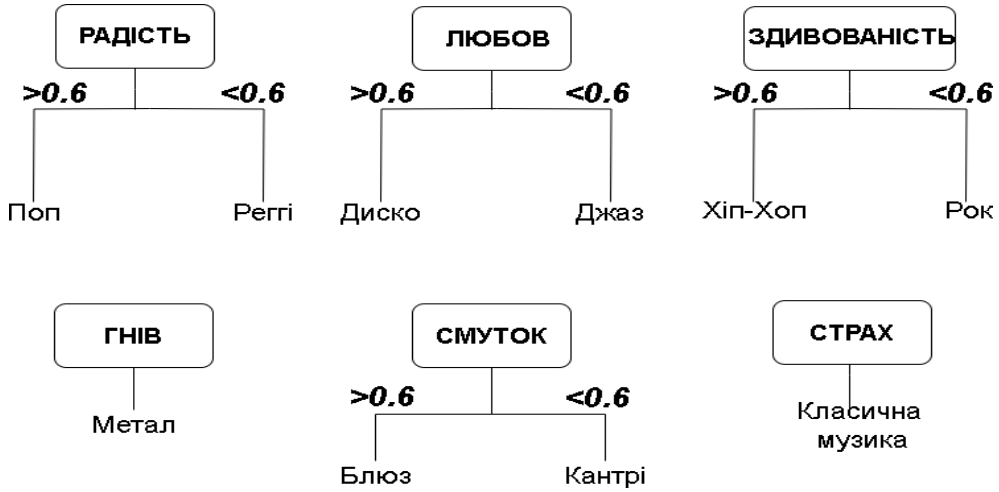

Рисунок 2.12 — Схема розподілу жанрів по емоціям

2.2 Інтерфейс користувача

Інтерфейс користувача розроблений на мові Python та бібліотеці Ipywidgets. Ця бібліотека дозволяє створювати інтерфейс користувача у середовищі ноутбуків, не витрачаючи час на розробку окремого інтерфейсу.

Користувач вводить запит, після чого ПЗ класифікує цей запит за класами емоцій та генерує музичну спектрограму, яку потім перетворює на музику.

Інтерфейс складається з наступних блоків.

Блок вводу тексту. У цей блок користувач вводить запит, за яким хоче згенерувати музику (рис. 2.13). Оскільки нейромережа тренувалася тільки на тренувальних даних на англійській мові, запит також повинен бути на англійській

Рисунок 2.13 — Форма введення запиту

Для класифікації запиту за класом емоції потрібно натиснути на кнопку “Класифікація”(рис. 2.14).

Рисунок 2.14 — Форма класифікації тексту

Після взаємодії з кнопкою розпочинається процес класифікації, користувач може бачити статус опрацювання запиту.

Щоб згенерувати музику потрібно натиснути на відповідну кнопку “Згенерувати музику.” (рис. 2.15).

Рисунок 2.15 — Форма класифікації тексту

Користувач може прослухати згенеровану музику, натиснувши на стрілку на кнопці прослуховування (рис. 2.16).

Рисунок 2.16 — Кнопка прослуховування

Для зберігання музики потрібно перейти до пункту “Файли” та натиснути кнопку “Зберегти” навпроти згенерованої музики — “result.wav”

Вибір використання такого інтерфейсу та розробку наведеними засобами полягає в тому, що ця дипломна робота акцентує увагу на методах машинного навчання. Наведений інтерфейс розділений на 2 секції: класифікація та генерація - це зроблено для того, щоб користувачі могли більш краще зрозуміти результати роботи алгоритмів та оцінити якість двох нейромереж окремо.

3 АНАЛІЗ ТА ТЕСТУВАННЯ ПЗ

Цей розділ описує параметри та архітектури нейромереж, які тестувалися в процесі навчання, щоб знайти оптимальний варіант за якістю.

3.1 Аналіз впливу різних параметрів та моделей у задачі класифікації

Цей підрозділ присвячений параметрам та моделям, що були протестовані для задачі класифікації тексту по класу емоцій, але не були обрані через низьку якість.

3.1.1 Модель RandomForestClassifier

Як описано в розділі 2, для задачі класифікації тестувалися два алгоритми навчання, RandomForestClassifier показує нижчу якість, ніжBRNN. Цей класифікатор є ансамблевим методом машинного навчання, який виконує задачу класифікації шляхом побудови численних дерев прийняття рішень, використовує усереднення для покращення результатів навчання. Аналогічно з навчанням нейромереж, дані перед подачею на навчання потрібно підготувати. Для навчання використовувалися ті ж дані, що для навчання нейромережі, попередньо очищені та лематизовані. На відміну від токенізації та ембедінгу даних для нейромережі, дані для RFC буливекторизовані через алгоритм TfidfVectorizer. TfidfVectorizer визначає міру оригінальності слова шляхом порівняння кількості разів, коли слово з’являється в документі, з кількістю документів, у яких це слово з’являється, використовуючи словник у пам’яті.

Результат навчання наведено на рис. 3.1.

```
accuracy 0.8477083333333333
              precision    recall  f1-score   support

       anger       0.87      0.78      0.82       607
        fear       0.86      0.76      0.80       566
         joy       0.78      0.95      0.86      1581
        love       0.89      0.64      0.75       434
     sadness       0.92      0.88      0.90      1434
    surprise       0.84      0.70      0.76       178

    accuracy                           0.85      4800
   macro avg       0.86      0.78      0.82      4800
weighted avg       0.85      0.85      0.85      4800
```

Рисунок 3.1 — Результати точності алгоритму на різних класах емоцій

Приклад класифікації речень наведено на рис. 3.2.

```
predict(sgd,"Sunny day in beatiful forest")
predict(sgd,"i'm very surprise to see you")
predict(sgd,"He is going throw the dark forest, he is so sad sad sad and afraid")
```

```
['joy']
['joy']
['fear']
```

Рисунок 3.2 — Класифікація речень

Цей алгоритм має гірші результати, ніж обрана архітектуранейромережі, оскільки дерева рішень схильні до перенавчання - побудови настільки складної архітектури, що алгоритм привчається тільки до одного патерну у даних.

3.1.2 Використання стемінга для обробки навчальних текстових даних

Процес стемінгу, на відміну від процесу лематизації, видаляє закінчення слів. Для стемінгу використан PorterStemmer з бібліотеки NLTK. Результат стемінгу представлений на рис. 3.3.

1998	truli feel passion enough someth stay true suc...	joy
1999	feel like wanna buy cute make see onlin even one	joy

Рисунок 3.3 — Приклад зміни речень після стемінгу

Результати навчання класифікатора RandomForestClassifier на стемінгових даних показано на рис. 3.4., точність на лематизованих даних становить 84%. Такий результат можна пояснити тим, що прибираючи закінчення, змінюється сенс деяких слів, таким чином слова перестають нести оригінальне значення.

accuracy 0.5133333333333333

	precision	recall	f1-score	support
anger	0.67	0.11	0.20	70
fear	0.74	0.39	0.51	64
joy	0.46	0.84	0.60	211
love	1.00	0.16	0.28	50
sadness	0.54	0.48	0.51	184
surprise	1.00	0.05	0.09	21
accuracy			0.51	600
macro avg	0.73	0.34	0.36	600
weighted avg	0.60	0.51	0.47	600

Рисунок 3.4 — Результати точності алгоритму на різних класах емоцій з операцією стемінг

3.1.3 Вплив часу навчання на результати класифікації

Як наведено в попередньому розділі, для повного навчання класифікатору потрібно 9 епох. Зупинка класифікатора відбувається, коли функція точності на валідаційних даних починає спадати. Між шостою та дев’ятою епохою навчання на валідаційних даних різниця становить 1, що є невеликою різницею, але результати класифікації на них дуже різні. На рис.3.5 наведений результат класифікації моделі, яка навчалася 6 епох, на

рис.3.6 представлений результат класифікації нейромережі, що навчалася 9 епох.

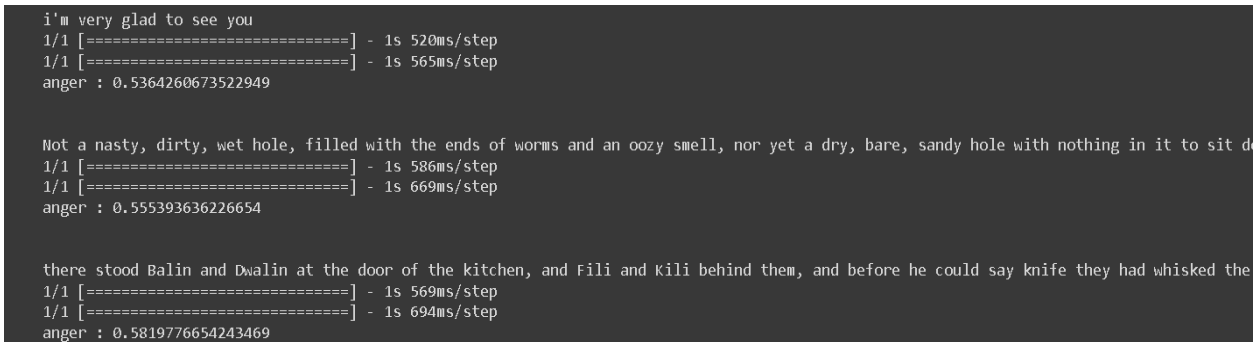

Рисунок 3.5 — Приклад класифікації моделі з меншим часом навчання

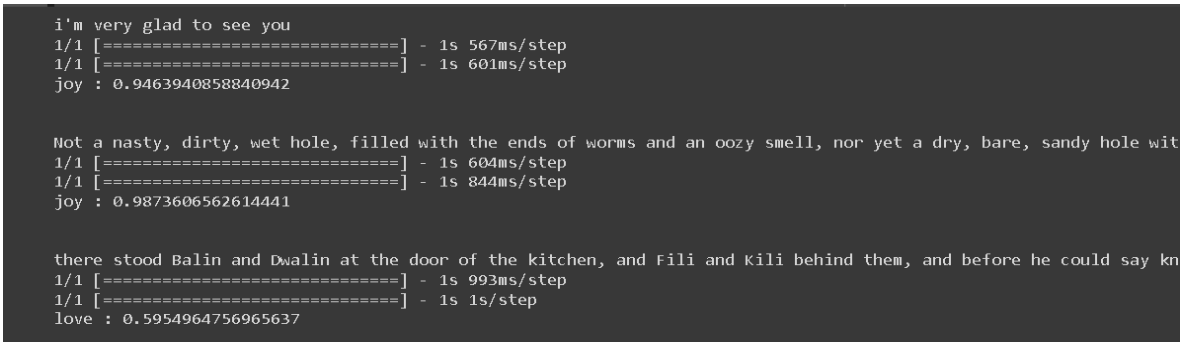

Рисунок 3.6 — Приклад класифікації моделі з більшим часом навчання

На наведених прикладах можна побачити, що у першому випадку майже всі прикладі класифікуються по переважаючій емоції “злість”, тоді як у другому прикладі, приклади класифікуються по більш відповідаючим запитам емоціям.

3.2 Аналіз параметрів, моделей для задачі генерації спектрограм

Як було описано в розділі 2, генеративно - змагальні нейромережі є типом нейромереж, що дуже складно навчати. Також варто зазначити, що генерація спектрограм є складною задачею в області генерації зображень, оскільки декілька пікселів можуть змінити звучання.

В цьому пункті описані архітектури нейромереж та параметри, що були протестовані під час роботи над задачею генерації спектрограм, але не були затверджені через низьку якість.

3.2.1 Вплив значень міток класу на якість згенерованих спектрограм

В фінальній версії ПЗ мітки класів є такими: для справжніхспектрограм (зображення з датасету) — вибір за допомогою рандомної функції між 0.9 та 1, для фейкових спектрограм (спектрограми, що згенеровані за допомогою GAN) — вибір за допомогою рандомної функції між 0 та 0.1. Окрім таких міток також тестувалися наступні: 1 для справжніх спектрограм, 0 для згенерованих. Такі значення обрані в оригінальній статті, на якій грунтується ця робота.

Для аналізу впливу тестування проводилися на фінальній архітектурі GAN, що показана в розділі 2 та використовує 3 види нормалізації

Результатом вибору таких міток є помилка конвергенції,яку можна побачити на графіках функції втрат та точності (рис. 3.7), також згенеровані спектрограми є майже однаковими, що є помилкою коллапсу (рис. 3.8).

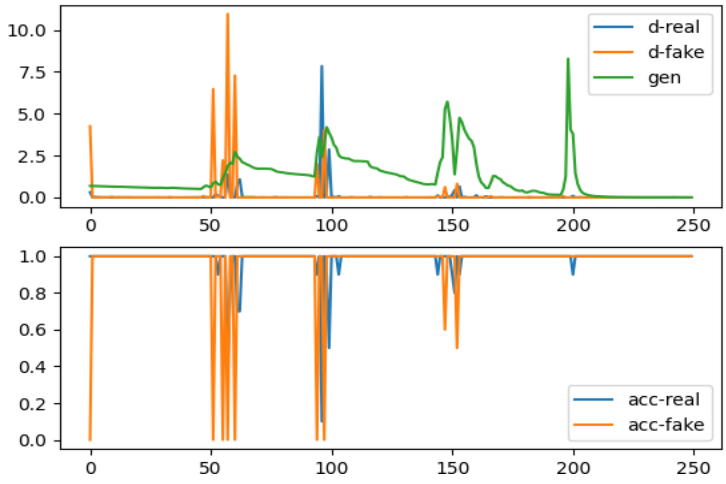

Рисунок 3.7 - Функція втрат (зверху) та функція точності (знизу)

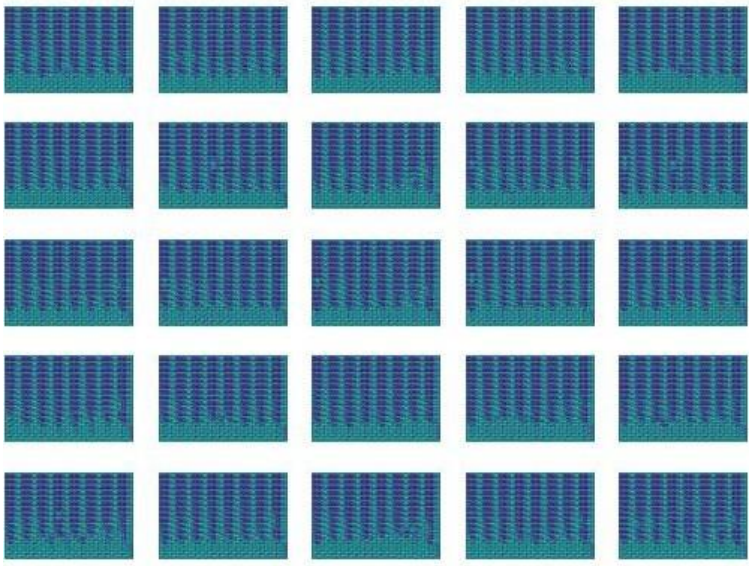

Рисунок 3.8 — Приклад згенерованих спектрограм

Причиною таких результатів є те, що генератор та дискримінатор не знайшли баланс, через що градієнт GAN затухає. Також на графіку видно, що функція точності з самого початку була максимальною і для справжніх, і для фейкових зображень. Таке відбувається, коли генератор не може згенерувати достатньо якісні зображення, через що дискримінатор з високою точністю може розпізнати справжню та фейкову спектрограму. Наведені зображення відображають результат за 50 епох навчання.

3.2.2 Вплив додавання шуму до латентного вектора

Однією з технік покращення якості генерації не додавання шуму (вектора гаусівського шуму) до зображення. Зазвичай її використовують у задачах генерації точних зображень: тварин, людей тощо. Для цієї роботи доданий шум допоміг покращити якість тільки для деяких жанрів музики, а саме: кантрі (рис. 3.9) та класична музика (рис. 3.10). Для жанру метал шум не вплинув на спектрограми.

Вплив шуму для всіх інших жанрів є негативним та представлений на рис. 3.12, на якому зображено спектрограми жанру Блюз, функцію втрат та точності (рис. 3.11).

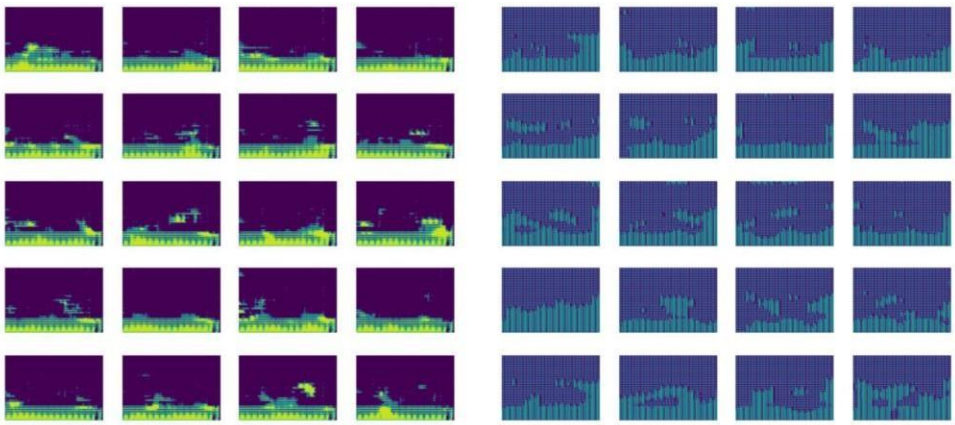

Рисунок 3.9 — Спектрограми без шума (зправа) та з шумом (зліва)

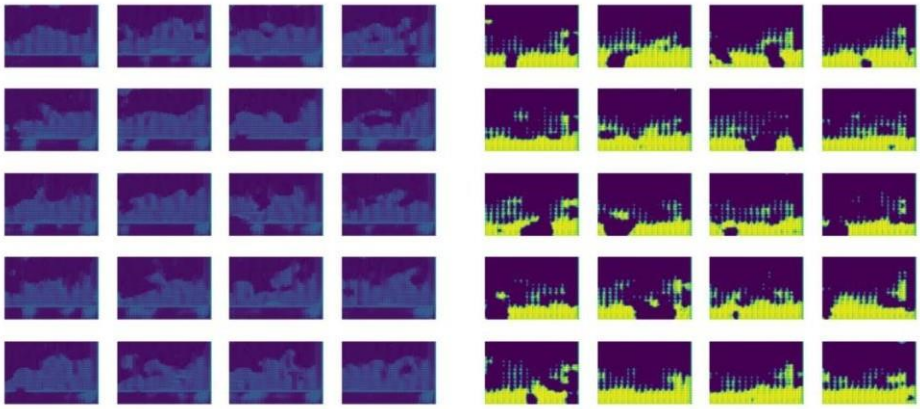

Рисунок 3.10 — Спектрограми без шума (зліва) та з шумом (зправа)

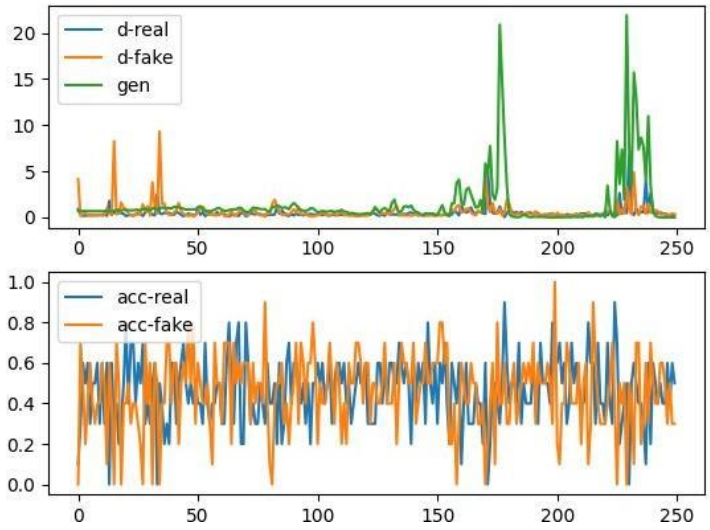

Рисунок 3.11 — Графік функції втрат (зверху) та точності (знизу)

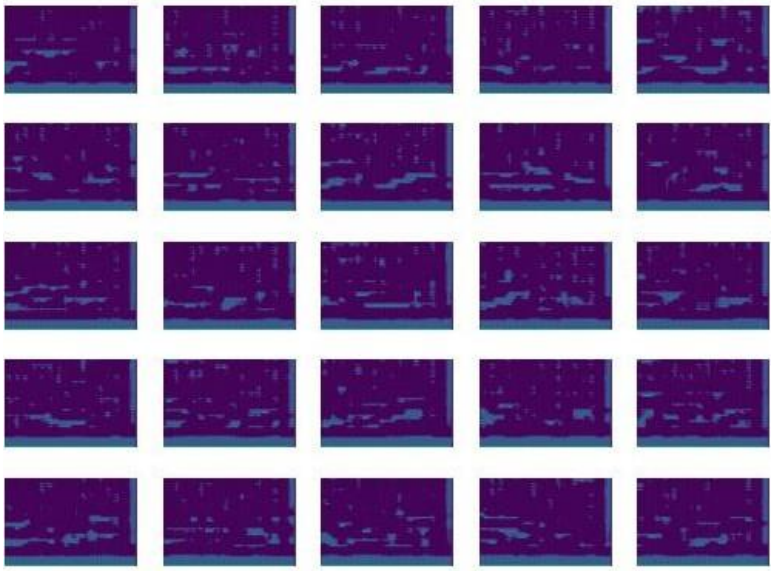

Рисунок 3.12 — Приклад згенерованих спектограм

3.2.3 Вплив різних конфігурацій архітектури GAN

У цьому пункті розглянуті протестовані архітектури генератора та дискримінатора GAN та їх результат генерації. Проведений аналіз, чому

наведені в цьому пункті архітектури мають гіршу якість ніж фінальна архітектура GAN - нейромережі. Моделі розділені по версіям. Версії з однаковими першими цифрами мають однакову архітектуру генератора та дискримінатора або якщо були змінені параметри тільки останнього шару згортки у генератора для збільшення/зменшення спектрограм. Архітектура GAN не змінювалася.

3.2.3.1 Версія 1.0.

Розмір генерованих спектрограм:112 на 112

Кількість кольорових каналів: 3

Мітки класів (справжні, фейкові): 1 та 0

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): разом

Кількість латентних точок: 10

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.00002, beta_1=0.5)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0002, beta_1=0.5)

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: нормалізація по батчам

Нормалізація зображень: від 0 до 1

Архітектура генератора представлена на рис.3.13, архітектура дискримінатора на рис. 3.14.

Приклад згенерованої спектрограми на рис. 3.15.

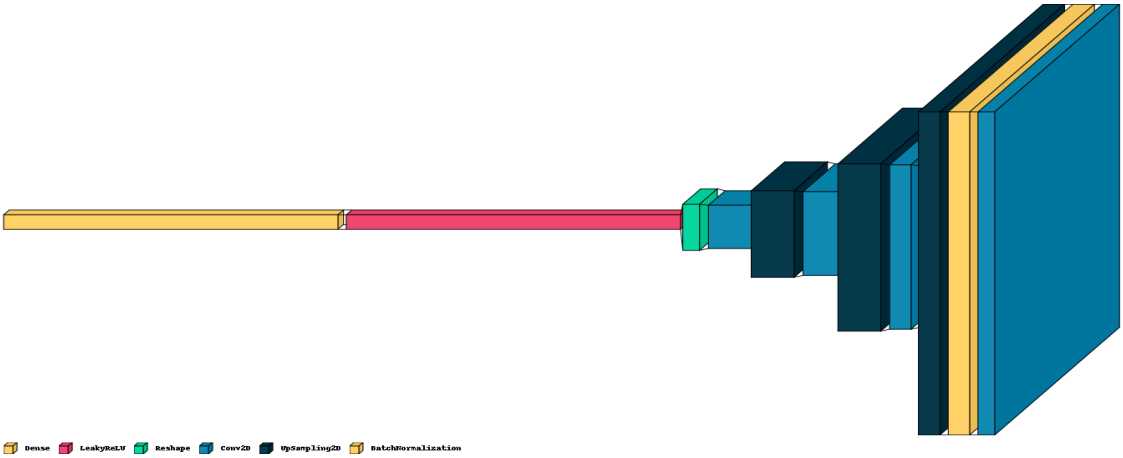

Рисунок 3.13 — Схема архітектури генератора

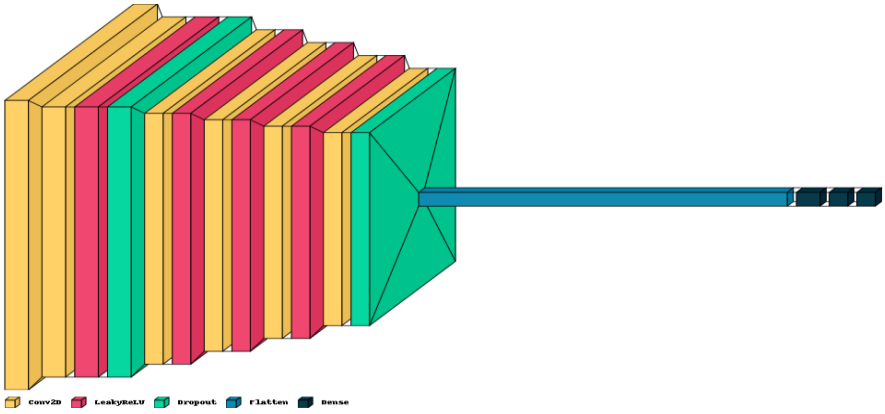

Рисунок 3.14 — Схема архітектури дискримінатора

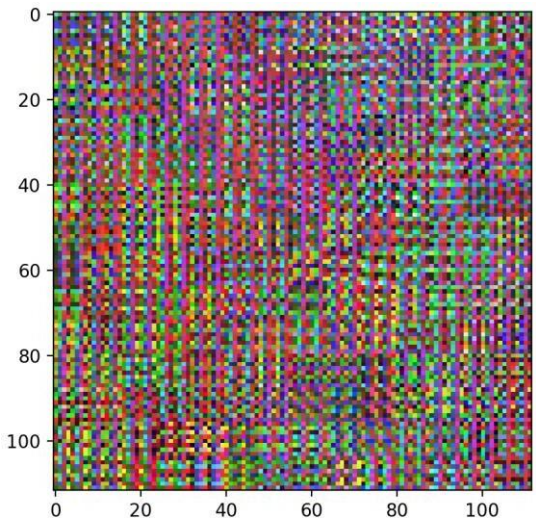

Рисунок 3.15- Приклад згенерованої спектрограми

Даній архітектурі не вистачає потужності (шарів) генератора, щоб вловити патерни у даних: колір фона та стовпчастий патерн спектрограм.

Версія 1.1.

Розмір генерованих спектрограм:112 на 112

Кількість кольорових каналів: 3

Мітки класів (справжні, фейкові): 1 та 0

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): разом

Кількість латентних точок: 10

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.0006, beta_1=0.8)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0006, beta_1=0.8

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: нормалізація по батчам

Навчання: 30 епох.

Приклад згенерованої спектрограми на рис. 3.16.

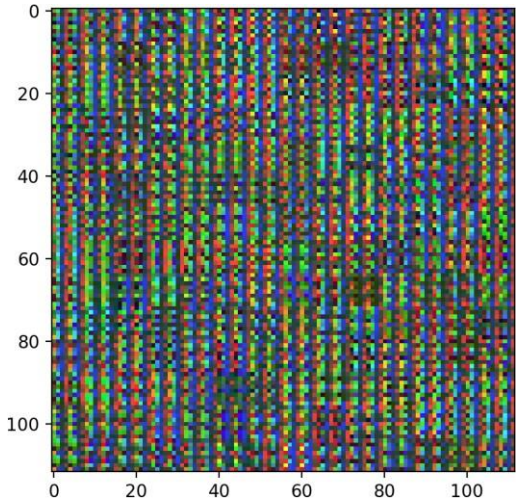

Рисунок 3.16- Приклад згенерованої спектрограми

Отримані результати свідчать про те, що генератор починає вловлювати основні кольори спектрограм, починають з’являтися вертикальні стовпці, але генератор має недостатньо згорткових шарів.

3.2.3.2 Версія 2.0

Розмір генерованих спектрограм:112 на 112

Кількість кольорових каналів: 3

Мітки класів (справжні, фейкові): 1 та 0

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): разом

Кількість латентних точок: 10

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.00002, beta_1=0.5)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0008,b1=0.8)

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: відсутня

Нормалізація зображень: від 0 до 1

Регуляризатор: L2 (0.03) на генераторі

Навчання: 30 епох.

Архітектура генератора представлена на рис.3.17, архітектура дискримінатора на рис. 3.18.

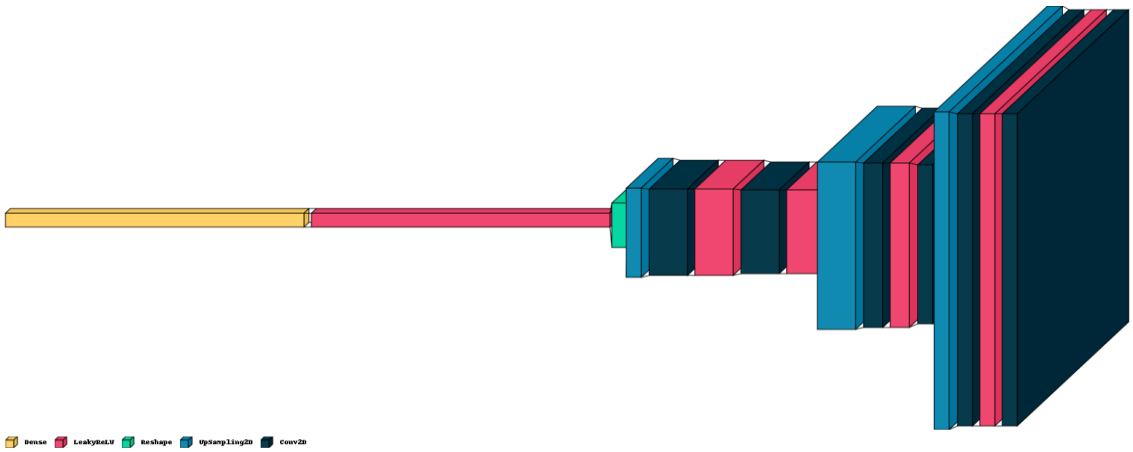

Рисунок 3.17 — Схема архітектури генератора

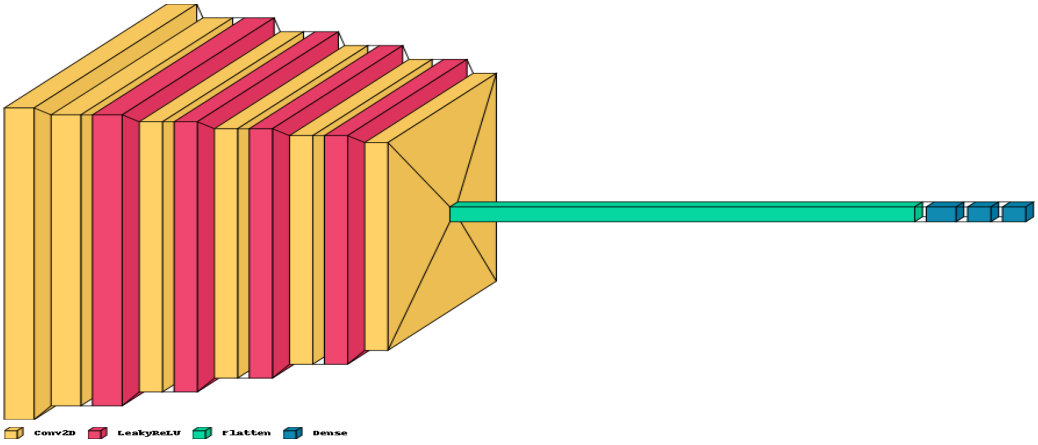

Рисунок 3.18 — Схема архітектури дискримінатора

Приклад згенерованої спектрограми на рис.3.19.

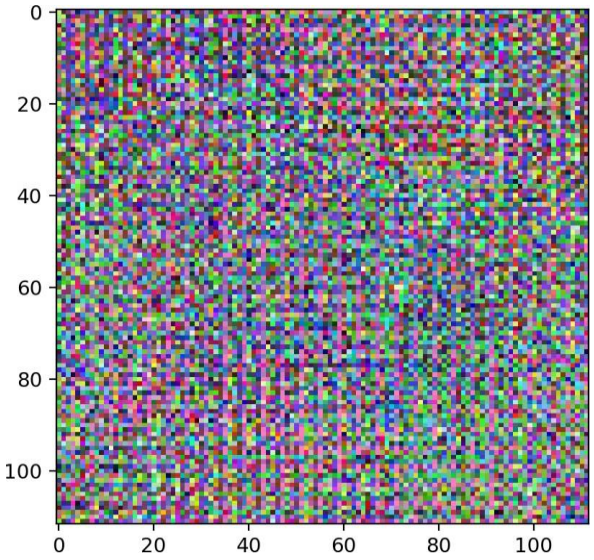

Рисунок 3.19 — згенерована спектрограма

Отримані результати, як і попередні, свідчать про недостачу згорткових шарів у генератора, але на відміну від попередньої версії, без нормалізації по батчам окремі пікселі є менш контрастними.

Версія 2.1.

Розмір генерованих спектрограм:128 на 128

Кількість кольорових каналів: 3

Мітки класів (справжні, фейкові): 1 та 0

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): окремо

Кількість латентних точок: 50

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.00002, beta_1=0.5)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0008,b1=0.9)

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: відсутня

Нормалізація зображень: від 0 до 1

Регуляризатор: L2 (0.03) на генераторі

Навчання: 30 епох.

Приклад згенерованої спектрограми на рис.3.20.

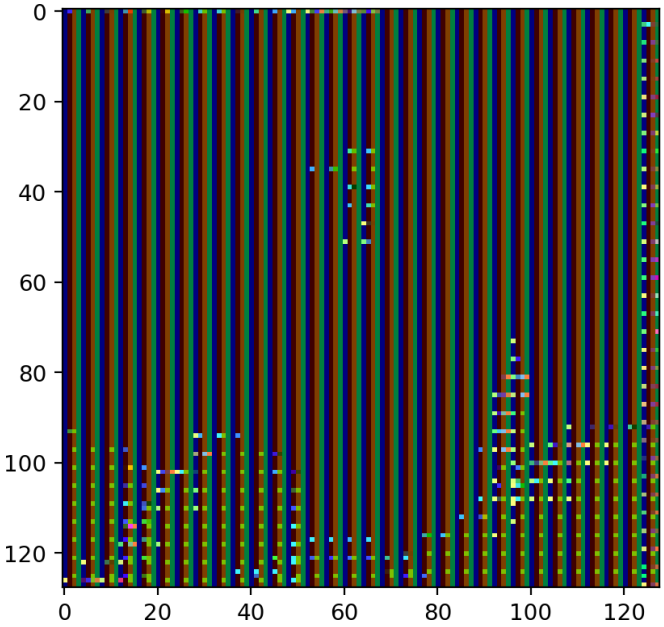

Рисунок 3.20 - згенерована спектрограма

Отримані результати, як і попередні, свідчать про недостачу згорткових шарів у генератора, але на відміну від попередньої версії, без нормалізації по батчам окремі пікселі є менш контрастними, але після збільшення розміру спектрограми, починають з’являтися перші вертикальні лінії та фон одного кольору.

Версія 2.2.

Розмір генерованих спектрограм:128 на 128

Кількість кольорових каналів: 3

Мітки класів (справжні, фейкові): 1 та 0

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): окремо

Кількість латентних точок: 70

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.00002, beta_1=0.5)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0006,b1=0.6)

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: відсутня

Нормалізація зображень: від 0 до 1

Регуляризатор: L2 (0.03) на генераторі

Навчання: 30 епох.

Приклад згенерованої спектрограми на рис.3.21.

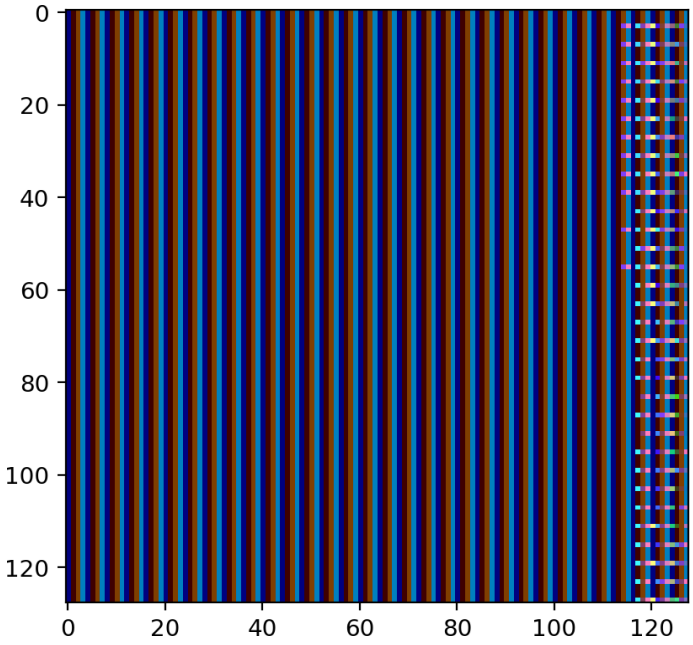

Рисунок 3.21 — згенерована спектрограма

Отримані результати, як і попередні, свідчать про недостачу згорткових шарів у генератора, після зменшення параметрів оптимізатора, генеруються вертикальні лінії синіх відтінків, але стобці будують не знизу, а на правому боці.

3.2.3.3 Версія 3.0.

Розмір генерованих спектрограм:128 на 128

Кількість кольорових каналів: 3

Мітки класів (справжні, фейкові): рандомний вибір між:(0.7,1.2) та 0:(0,0.3

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): окремо

Кількість латентних точок: 50

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.0002, beta_1=0.5)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0006,b1=0.6)

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: нормалізація по батчам

Нормалізація зображень: від -1 до 1

Регуляризатор: RandomNormal(stddev=0.02) на дискримінаторі та генераторі

Навчання: 82 епохи.

Архітектура генератора представлена на рис.3.22, архітектура дискримінатора на рис. 3.23.

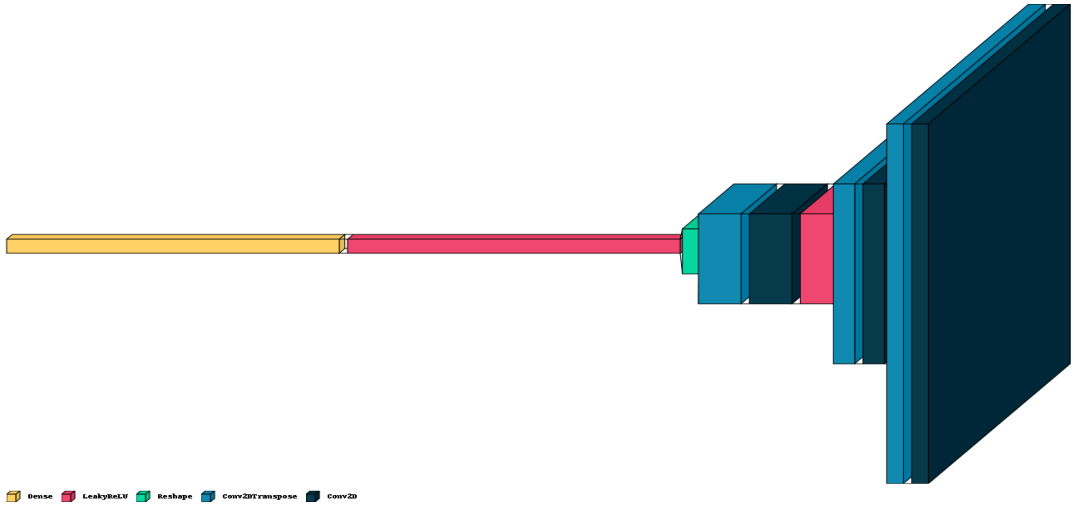

Рисунок 3.22 — Схема архітектури генератора

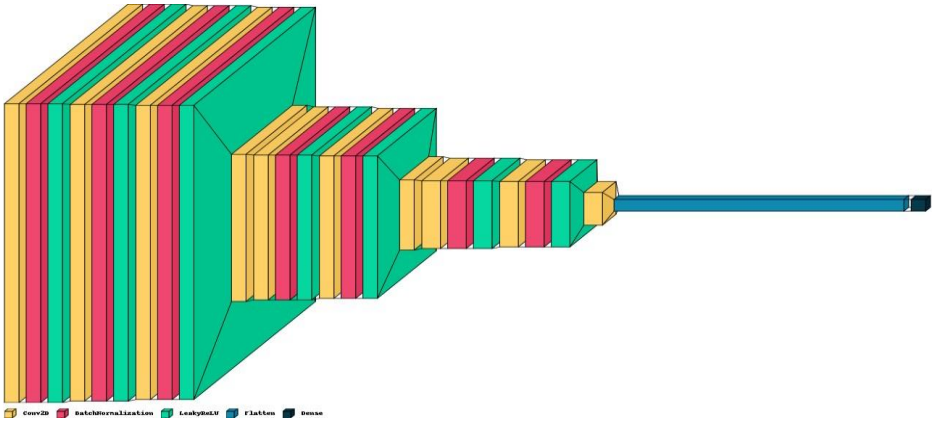

Рисунок 3.23 - Схема архітектури дискримінатора

Приклад згенерованої спектрограми на рис. 3.24. Графіки функцій втрат та точності на рис. 3.25.

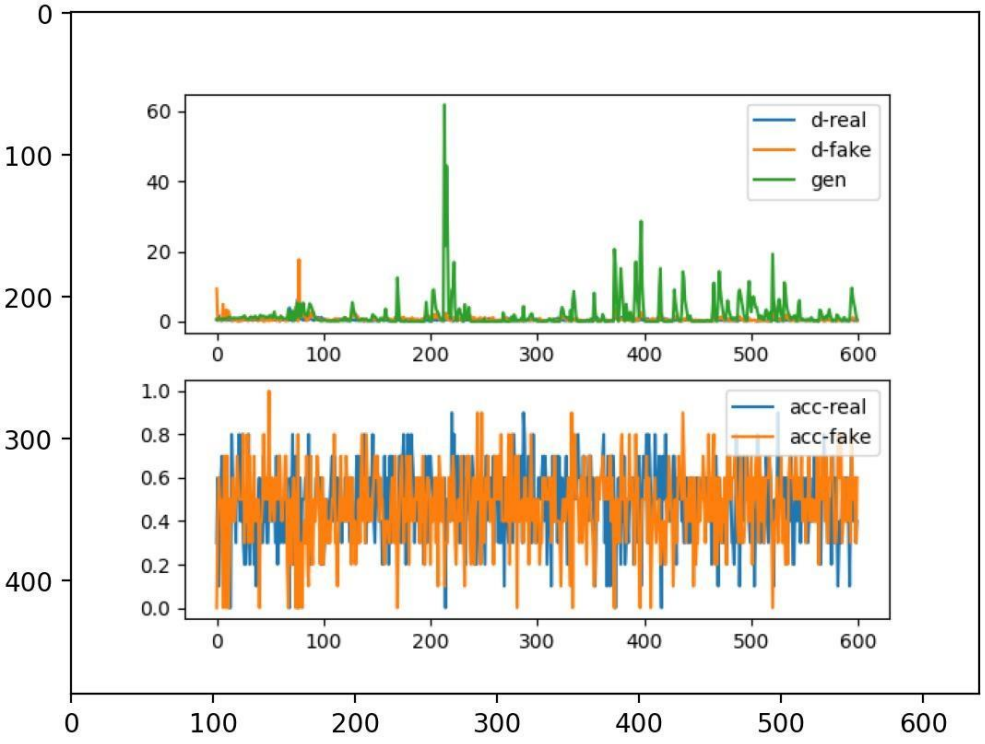

Рисунок 3.24 — Графік функції втрат (зверху) та функції точності (знизу)

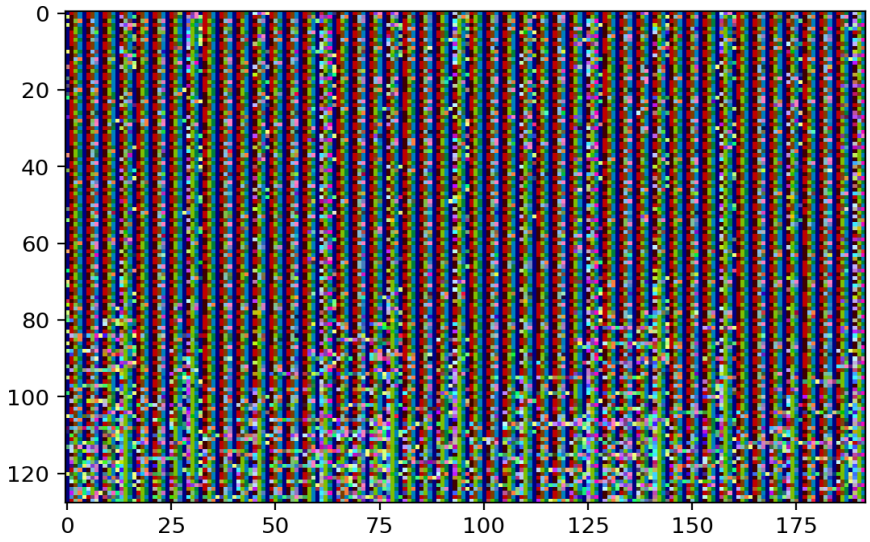

Рисунок 3.25 — Згенерована спектрограма

Отримані результати свідчать про те, що після зміни функції збільшення розміру з UpSampling на Conv2DTranspose, нейромережа починаєгенерувати переривисті вертикальні лінії, які збільшують знизу. Це сигналізує про те, що функцію Conv2DTranspose потрібно використовувати у подальших версіях.

Версія 3.1

Особливість цієї версії в тому, що активаційна функція LeakyRelu замінена на Relu. В результаті функція втрати GAN досягла максимального значення.

Розмір генерованих спектрограм:128 на 128

Кількість кольорових каналів: 3

Мітки класів (справжні, фейкові): рандомний вибір між:(0.7,1.2) та 0:(0,0.3)

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): окремо

Кількість латентних точок: 50

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.0002, beta_1=0.5)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0009,b1=0.7)

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: нормалізація по батчам

Нормалізація зображень: від -1 до 1

Регуляризатор: RandomNormal(stddev=0.02) на дискримінаторі та генераторі

Навчання: 44 епохи.

Функція активації замінена на Relu.

Згенерована спектрограмма представлена на рис. 3.26, графіки функції втрат та точності представлені на рис. 3.27.

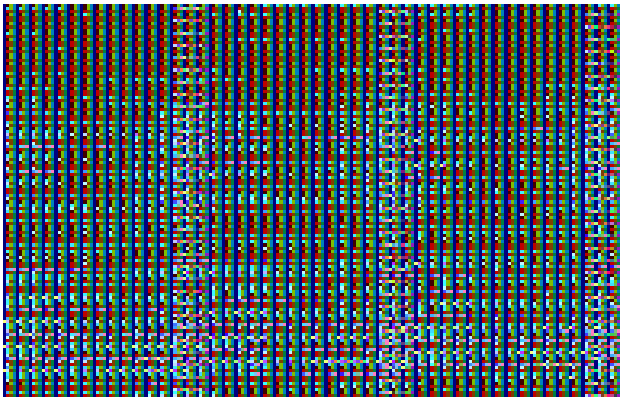

Рисунок 3.26 — Згенерована спектрограма

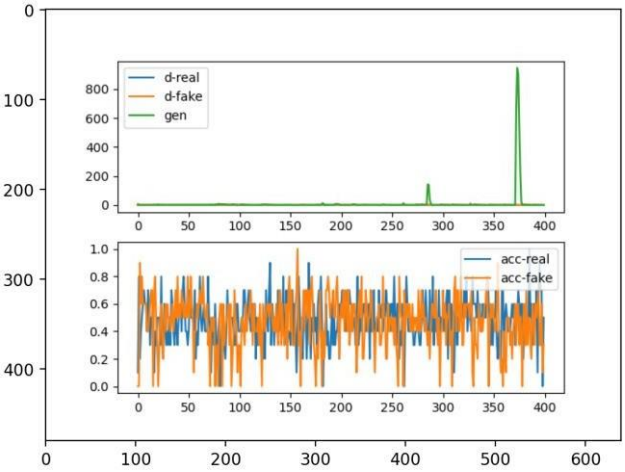

Рисунок 3.27 — Графік функції втрат (зверху) та функції точності (знизу)

Зміна функції активації генератора з LeakyRelu на Relu дестабілізує навчання, викликаючи швидке пікове зростання функції втрат, та більшу кількість нульових значень функції точності.

3.2.3.4 Версія 4.0

Розмір генерованих спектрограм:128 на 192

Кількість кольорових каналів: 1

Мітки класів (справжні, фейкові): рандомний вибір між:(0.8,1.1) та 0:(0,0.2)

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): окремо

Кількість латентних точок: 70

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.0002, beta_1=0.5)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0004,b1=0.6)

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: нормалізація по батчам

Нормалізація зображень: від -1 до 1

Регуляризатор: RandomNormal(stddev=0.02) на дискримінаторі та генераторі

Навчання: 82 епохи.

Архітектура генератора представлена на рис.3.28, архітектура дискримінатора рис. 3.29.

Зміна розміру справжніх та згенерованих спектрограм враховує оригінальне співвідношення сторін.

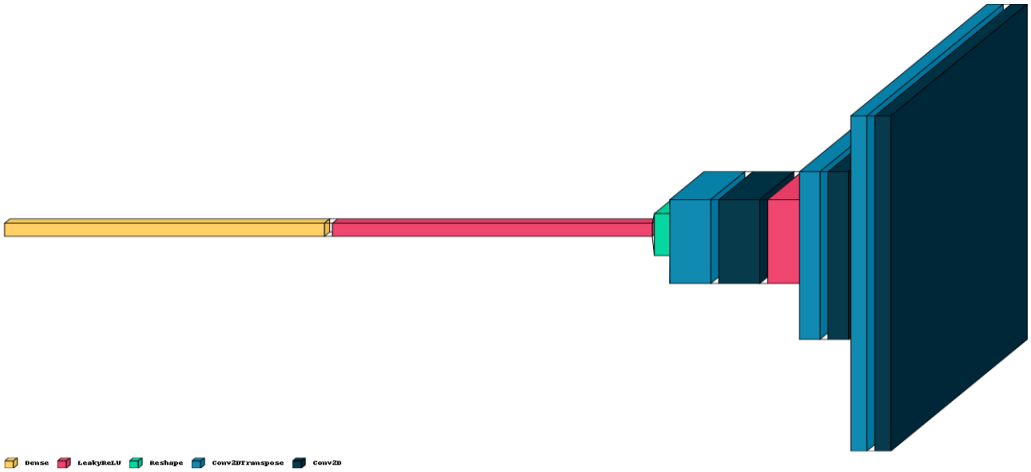

Рисунок 3.28 — Схема архітектури генератора

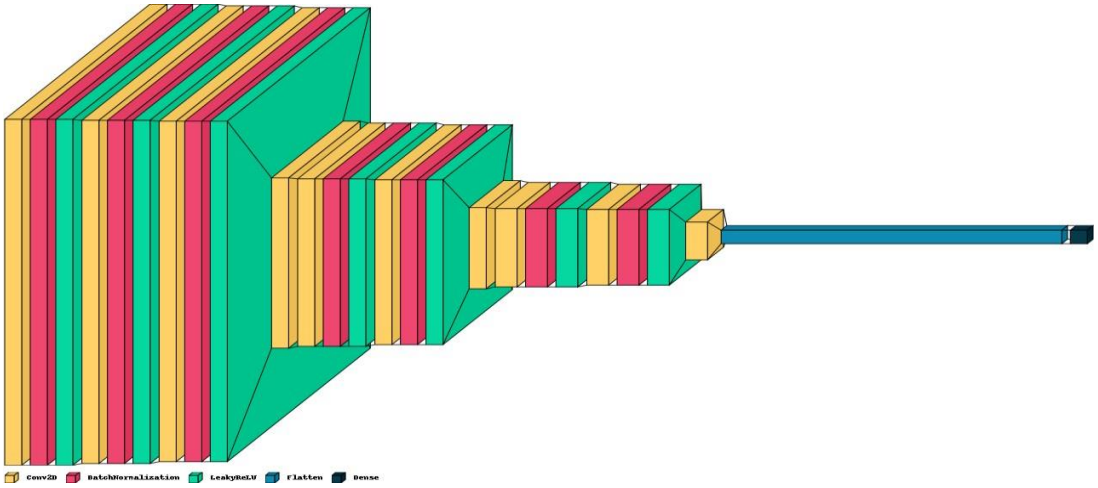

Рисунок 3.29- Схема архітектури дискримінатора

Графіки функцій втрат та точності на рис. 3.30. Приклад згенерованої спектрограми на рис. 3.31.

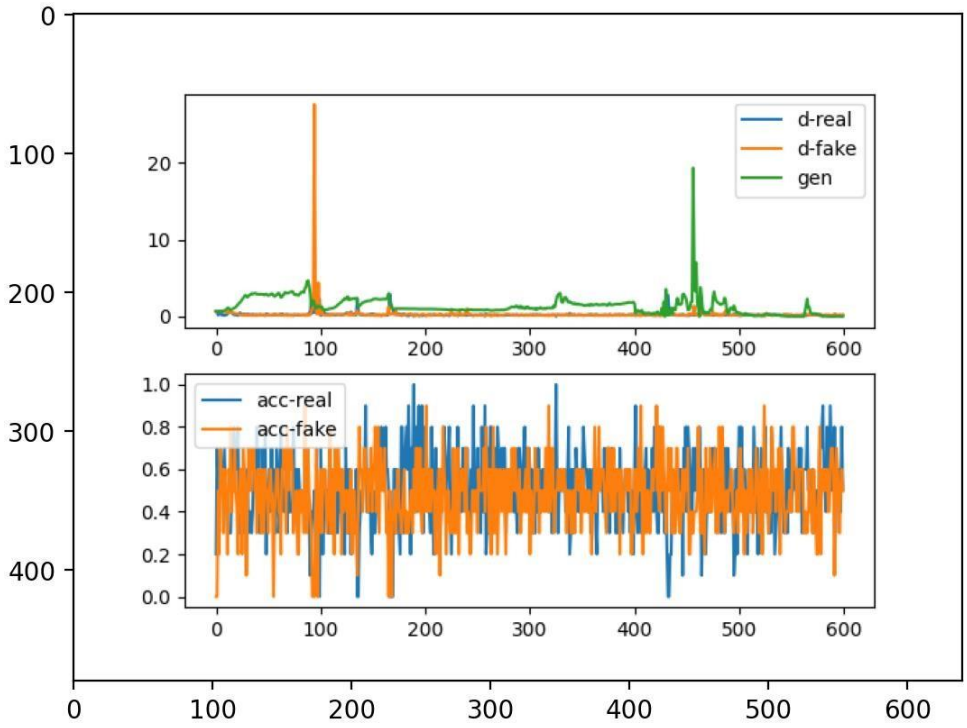

Рисунок 3.30 — Графік функції втрат (зверху) та функції точності (знизу)

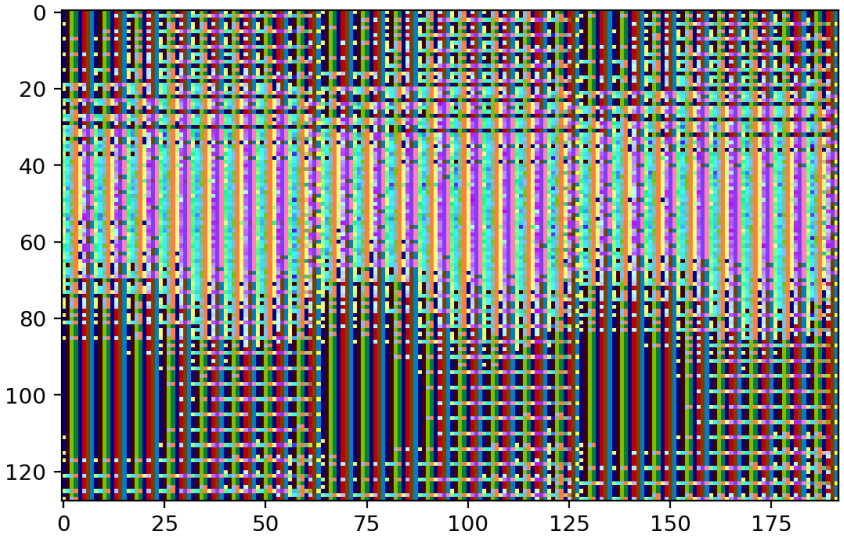

Рисунок 3.31 — Згенерована спектрограма

Врахування пропорцій оригінальних спектрограм та збільшення розміру латентного вектору генерує спектрограму, в якій розділені верхня та нижня частини та до менших значень функції втрати. Генеровані спектрограми містить повторювані вертикальні лінії однакових кольорів, ще не є гарним результатом генерації.

3.2.3.5 Версія 5.0

Розмір генерованих спектрограм:128 на 192

Кількість кольорових каналів: 1

Мітки класів (справжні, фейкові): рандомний вибір між:(0.9,1) та 0:(0,0.1)

Навчання дискримінатора на справжній та фейкових спектрограмах (разом/ окремо): окремо

Кількість латентних точок: 100

Оптимізатор дискримінатор (lr, beta_1): Адам (lr=0.0002, beta_1=0.5)

Оптимізатор GAN (lr, beta_1): Адам (lr=0.0002,b1=0.5)

Функція втрат дискримінатора: binary_crossentropy

Функція втрат GAN : binary_crossentropy

Нормалізації: нормалізація по батчам, нормалізація по пікселям, MiniBatchStdv.

Нормалізація зображень: від -1 до 1

Регуляризатор: RandomNormal(stddev=0.02) на дискримінаторі та генераторі

Навчання: 27 епох.

Архітектура генератора представлена на рис. 3.32, архітектура дискримінатора на рис. 3.33.

Приклад згенерованої спектрограми на рис. 3.34. Графіки функцій втрат та точності на рис. 3.35.

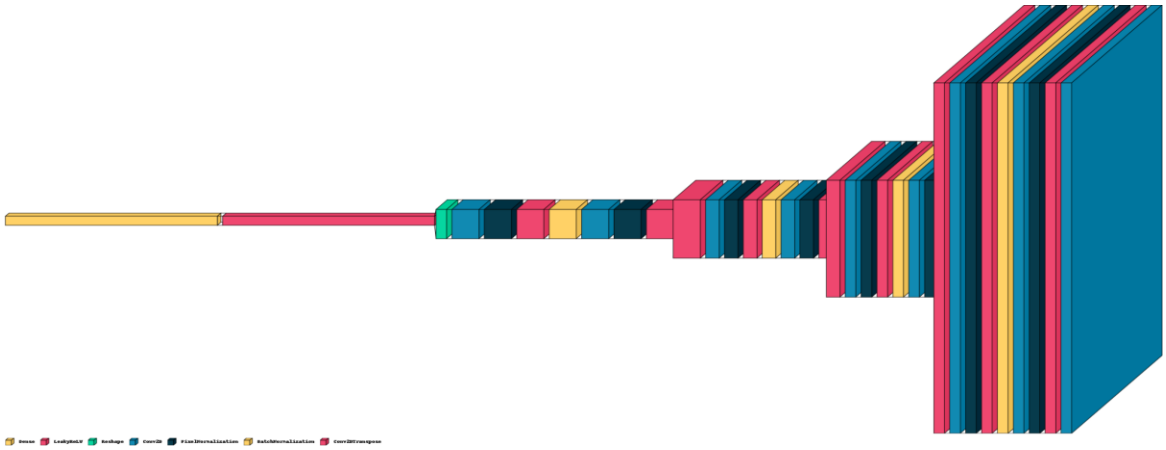

Рисунок 3.32 — Схема архітектури генератора

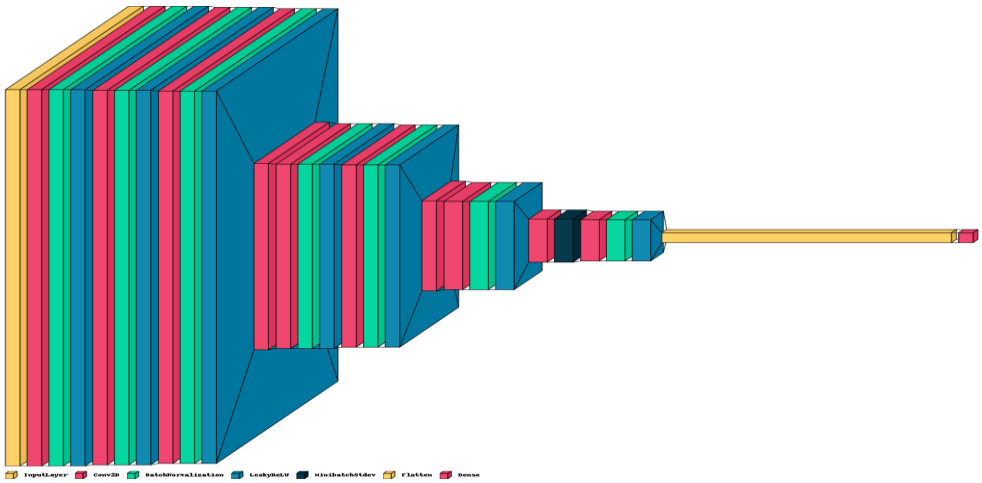

Рисунок 3.33 - Схема архітектури генератора

Використання одразу трьох видів нормалізації значно покращує результат, генератор вже генерує спектрограми, близькі до оригінальних.

Можна помітити використання двох основних кольорів, заповнення відстані між вертикальними стобцями фоновим кольором.

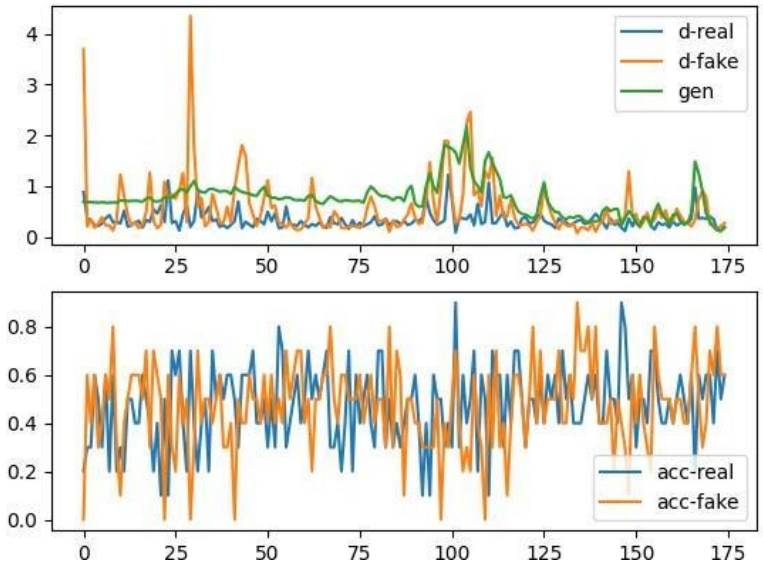

Рисунок 3.34 — Графік функції втрат (зверху) та функції точності (знизу)

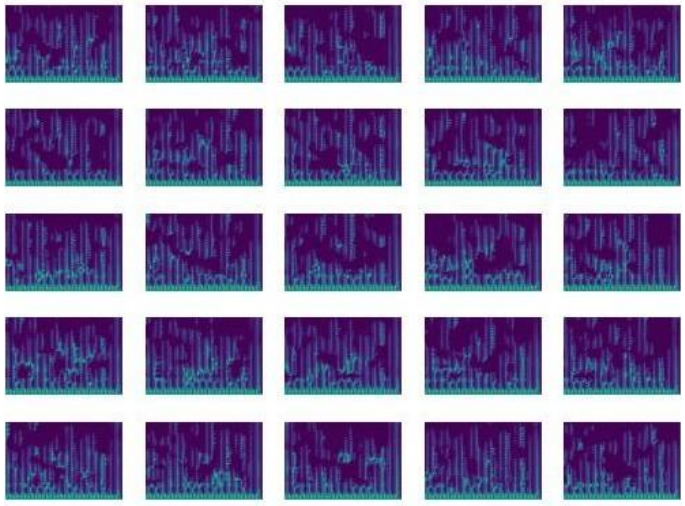

Рисунок 3.35 - Приклади згенерованих спектрограм

3.2.4 Аналіз впливу різних видів нормалізації на результати генерації

3.2.4.1 Нормалізація по батчам

Вплив нормалізації по батчам можна простежити у попередньому пункті. В цьому пункті приведені спектрограми, які були згенеровані з окремими нормалізаціями. Наведені спектрограми згенеровані на фінальній версії GAN, без накладання шуму на оригінальні спектрограми.

Від місця розташування нормалізації дуже сильно залежить результат генерації. Як наведено на рис.3.36, при розташуванні нормалізації тільки у генератора, згенеровані спектограми є однаковими та не знаходять патерни у оригінальних спектрограмах, але функція втрат стабільна та не спадає до нуля.

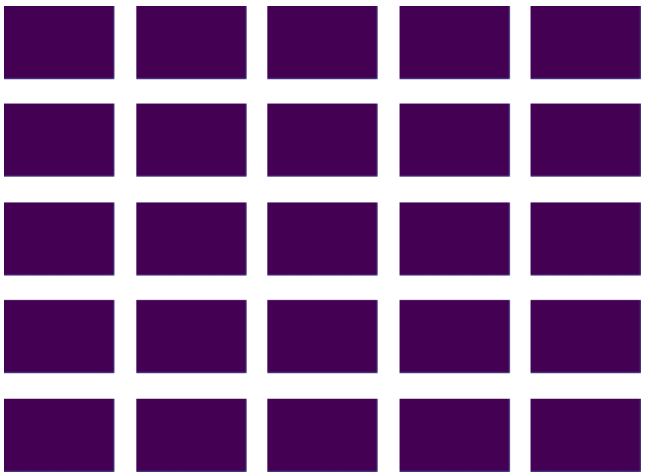

Рисунок 3.36 — Приклад згенерованих спектрограм

При розташуванні нормалізації тільки у дискримінатора, є схожа проблема. Відмінністю є поява горизонтальних ліній, майже ідентичного з фоном кольору. Представлені спектрограми згенеровані зі спектрограми жанру Блюз (рис. 3.37)

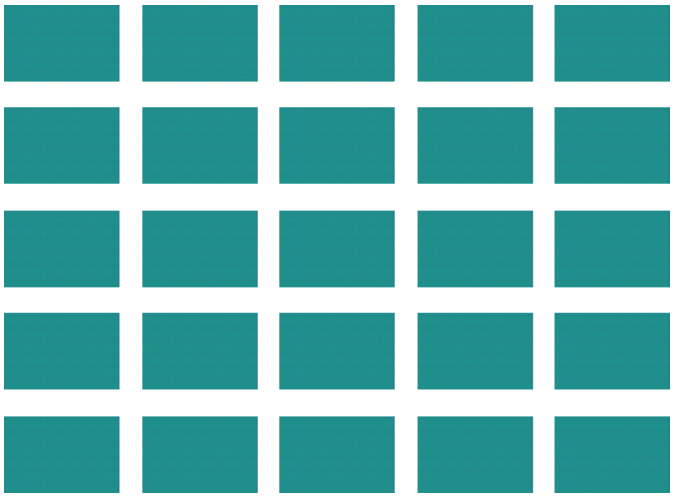

Рисунок 3.37 — Приклад згенерованих спектрограм

Використання нормалізації по батчам у генераторі та дискримінаторі, що є фінальної версією ахітектури нейромережі, представлена у наступному розділі.

3.2.4.2 Нормалізація MiniBatchDev

Як зазначалося в попередньому розділі, цей вид нормалізації дозволяє уникнути проблеми генерації однакових спектрограм. Представлені на рис. 3.38 спектрограми є результатом генерації на аналогічній з фінальною архітектурі, в якій прибрали нормалізацію у дискримінатора. Представлені спектрограми згенеровані зі спектрограми жанру Блюз.

Приклад використання цього виду нормалізації показан у наступному розділі.

Результатом є генерація однакових спектрограм.

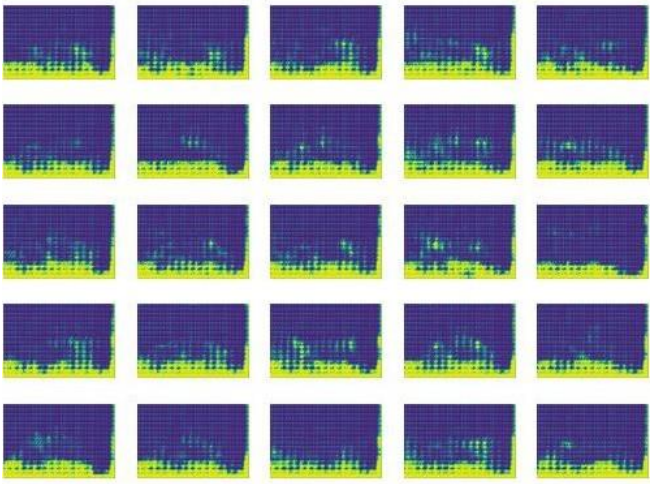

Рисунок 3.38 — Приклад згенерованих спектрограм

3.2.4.3. Нормалізація по пікселям

Як зазначалося в попередньому розділі, цей вид нормалізації єаналогом нормалізації по батчам для ProgressiveGAN. Зазвичай, використовують або нормалізацію по батчам, або нормалізацію по пікселям. Поєднання нормалізації по батчам та по пікселям дають кращі результати через архітектуру GAN: нормалізація по пікселям виконуються після кожного згорткового шару, тоді як нормалізація по батчам виконуються посередині двох згорткових шарів після активаційної функції. На рис.3.39 наведений приклад спектрограм, які будуть згенеровані без нормалізації по пікселям, всі інші параметри збережені, як у фінальній версії: нормалізаціяпо батчам та MiniBatchDev у дискримінатора, тільки нормалізація по батчаму генератора.

Приклад використання цього виду нормалізації показан у наступному розділі.

На наведеному нижче прикладі можна побачити, що основний колір згенерованих спектрограм — це фоновий колір, але на відміну від

спектрограм, що були згенеровані без нормалізації по батчам, ці спектрограми мають горизонтальну полосу зеленого кольору знизу.

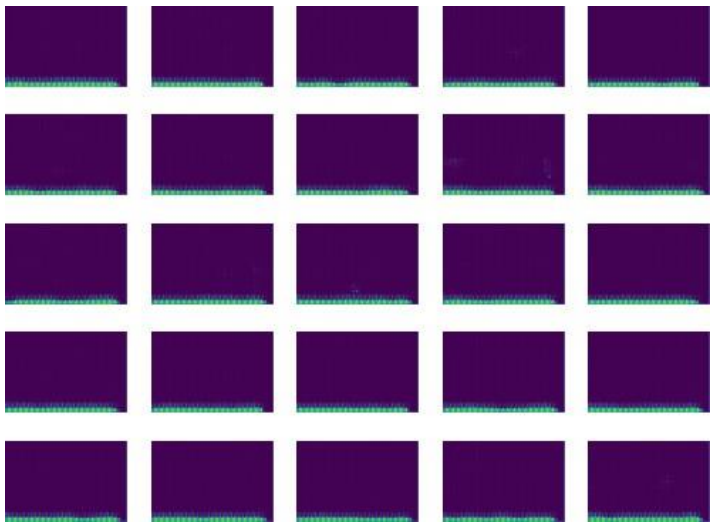

Рисунок 3.39 — Приклад згенерованих спектрограм

3.3 Тестування ПЗ

Тестування ПЗ дає змогу симулювати поводження алгоритмів при роботі з реальними даними. Основні юзер - кейси розділені на дві категорії. Перший юзер кейс як тестувальні дані використовує запити, які можуть використовувати опис дії або події, наприклад, ігровій студія потрібно згенерувати музику для фонового озвучення конкретної частини гри. Другий юзер - кейс — це генерація музики для особистих потреб користувача, наприклад, для прослуховування. Тестування показало хороші результати, приклади запитів для двох юзер - кейсів представлені нижче.

1. Тест 1

Сценарій: розробники нової гри про Дикій Захід вводять наступний запит для музичного озвучення частини гри.

Введений запит на англійській:

He ran like the wind across the desert prairies, but the chase cant caught up with him

Переклад:

Він біг, немов вітер по пустинним преріям, проте погоня не змогла наздогнати його

Результат класифікації (рис.3.40), згенерована спектрограма (рис.3.41)

Результат роботи ПЗ: клас емоції — щастя, музичний жанр — реггі

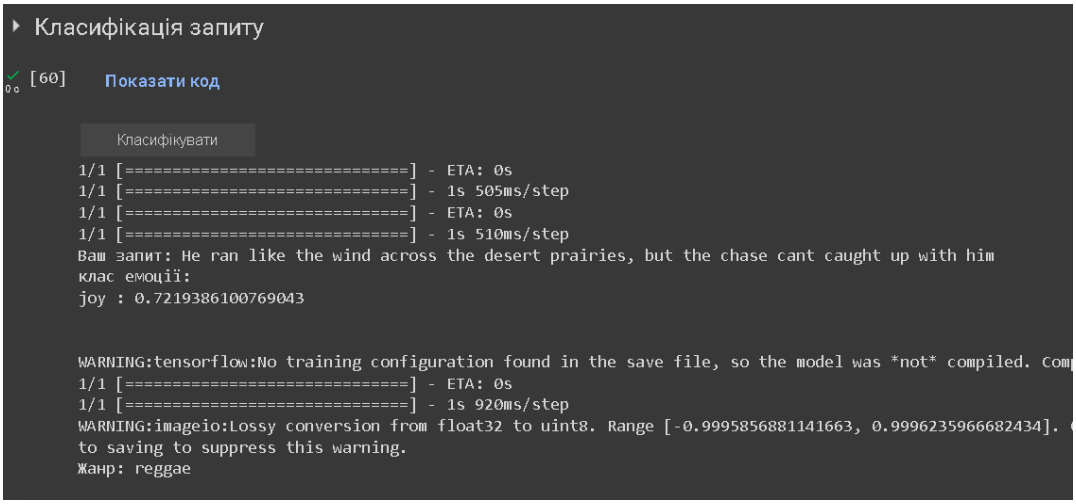

Рисунок 3.40 — Класифікація тесту 1.

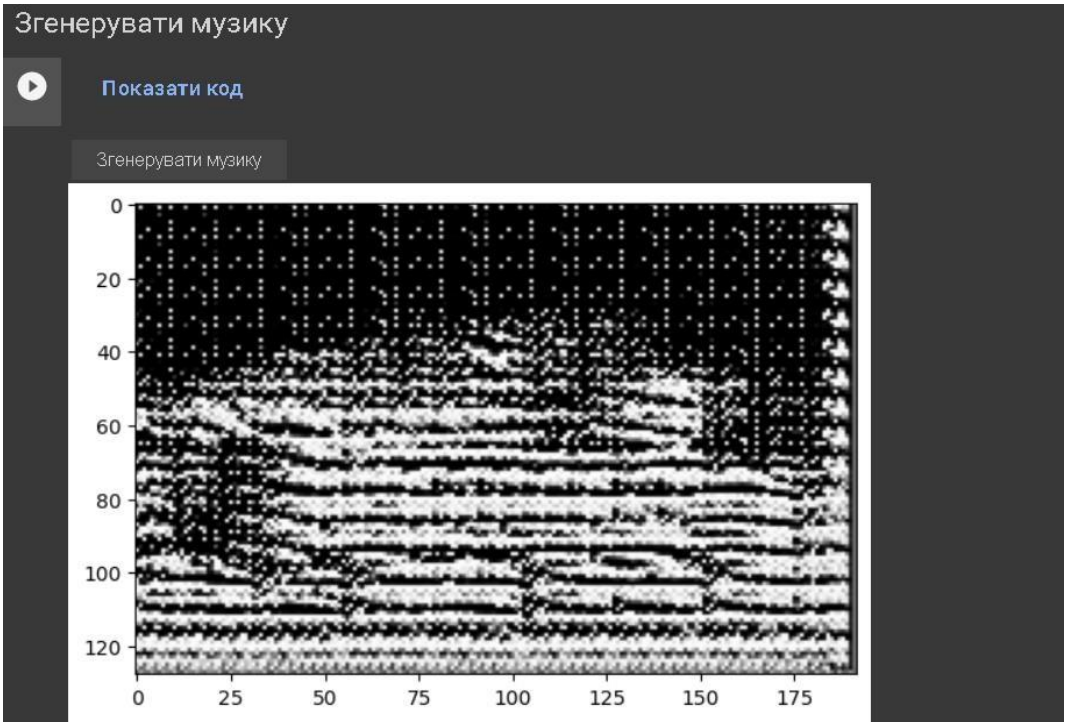

Рисунок 3.41 — Згенерована спектрограма тесту 1.

2. Тест 2

Сценарій: користувач хоче згенерувати музики для фонової музики під час виступу з презентацією про поїздку з друзями.

Введений запит на англійській: Presentation about vacation with friends feels like dream and i am in love with idea behind this

Переклад: Герой йде по похмурому лісу, чим далі він йде, вітки сплетаються над його головою, пропускаючи все менше місячного світла

Результат класифікації (рис.3.42), згенерована спектрограма (рис.3.43)

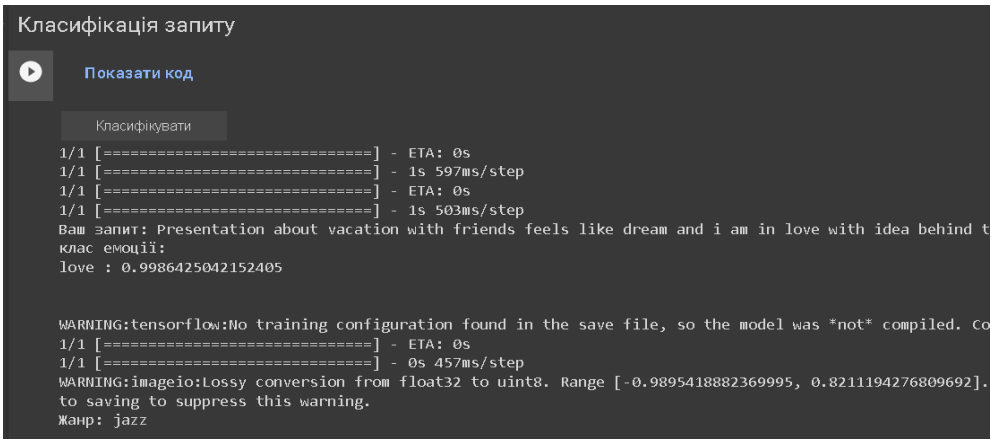

Рисунок 3.42 — Класифікація тесту 2.

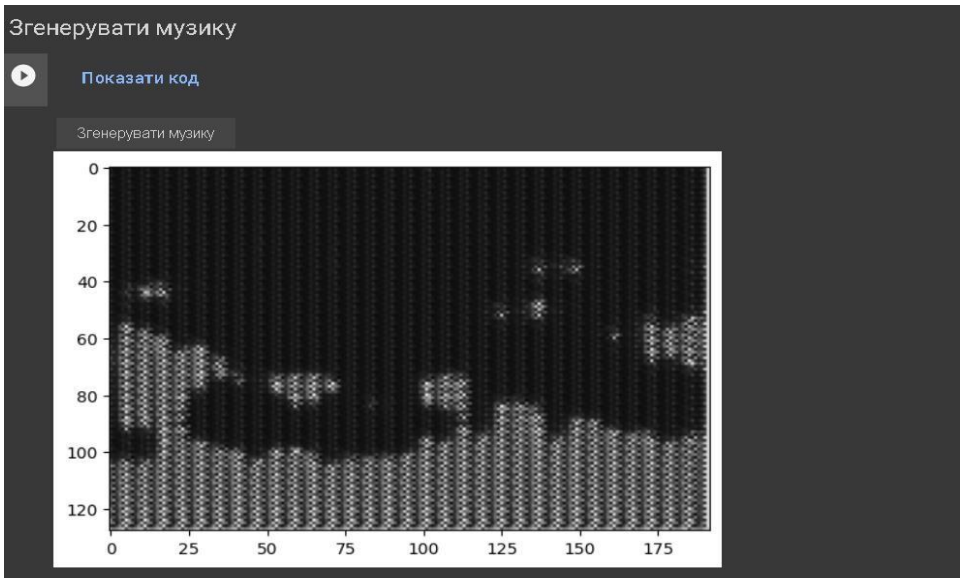

Рисунок 3.43 — Згенерована спектрограма тесту 2.

Результат роботи ПЗ: клас емоції — любов, музичний жанр — джаз

3. Тест 3

Сценарій: користувач хоче згенерувати музику під емоцію, що наразі відчуває, для особистого прослуховування.

Введений запит на англійській: I don`t want to hold this shocked effect forever

Переклад: Сьогодні в мене гарний настрій і я хочу веселу музику.

Результат класифікації (рис.3.44), згенерована спектрограма (рис.3.45)

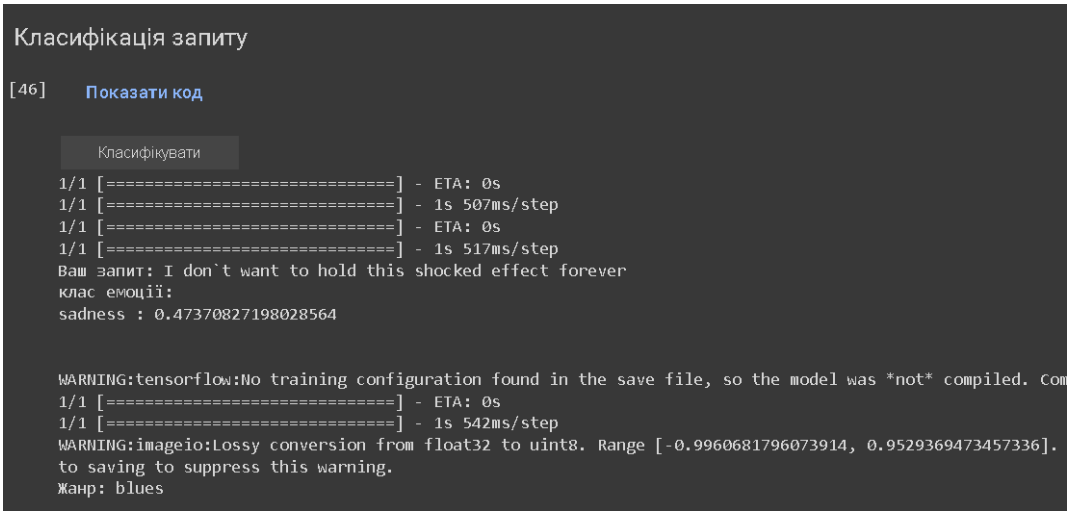

Рисунок 3.44 — Класифікація тесту 3.

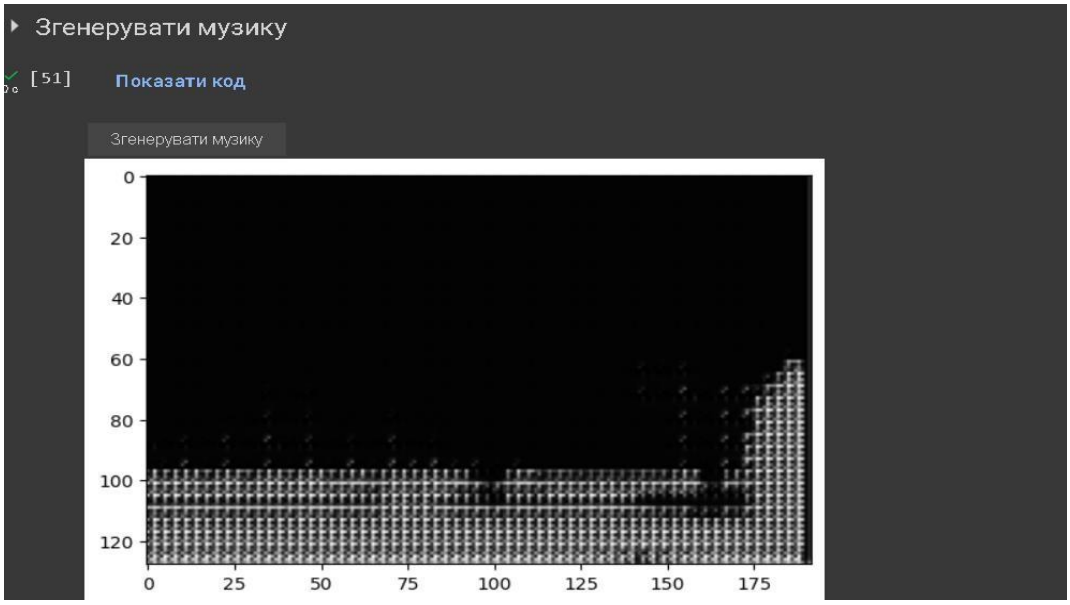

Рисунок 3.45 — Згенерована спектрограма тесту 3.

Результат роботи ПЗ: клас емоції — смуток , музичний жанр — блюз.

4. Тест 4

Сценарій: користувач хоче озвучити книгу “Гаррі Поттер та таємна кімната” Дж. Роулінг та додати фонове звучання під зачитаний текст.

Введений запит на англійській: Aunt Petunia’s masterpiece of a pudding, the mountain of cream and sugared violets, was floating up near the ceiling. On top of a cupboard in the corner crouched Dobby.

Переклад: Шедевральний пудингу тітки Петунії — гора вершків і цукрових фіалок — полетів під стелею. На шафі в кутку згорбився Добі.

Результат класифікації (рис.3.46), згенерована спектрограма (рис.3.47)

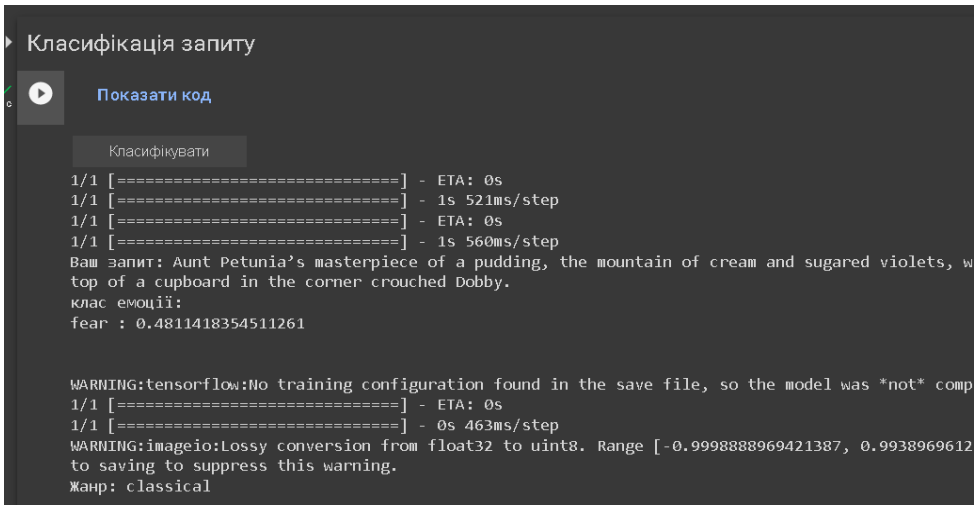

Рисунок 3.46 — Класифікація тесту 4.

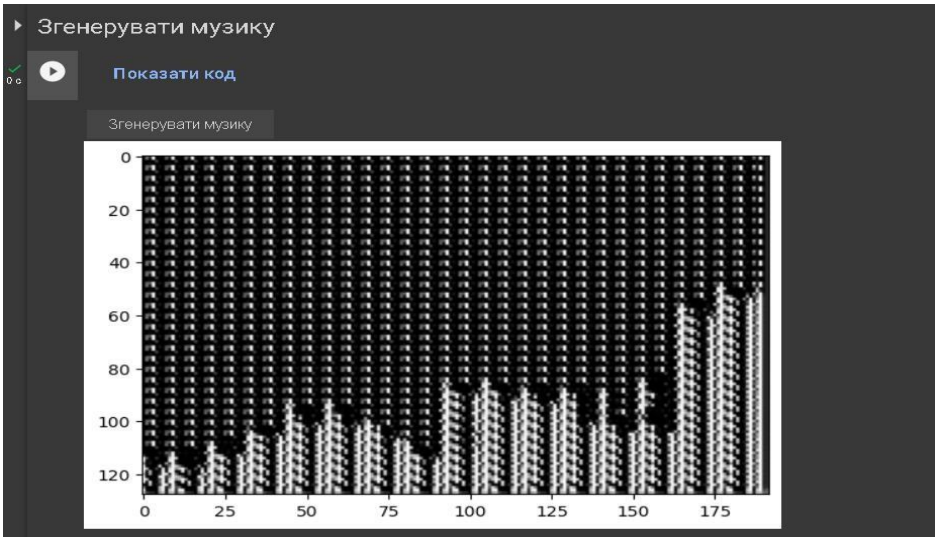

Рисунок 3.47 — Згенерована спектрограма тесту 4.

Результат роботи ПЗ: клас емоції — страх, музичний жанр — класична музика.

5. Тест 5

Сценарій: адміністратору онлайн - магазину дитячих іграшок потрібно згенерувати музику, яка б звертала увагу користувачів на акцію.

Введений запит на англійській: A site with children's toys is running a sale on Lego sets.

Переклад: На сайті дитячих іграшок проходить розпродаж наборів Lego.

Результат класифікації (рис.3.46), згенерована спектрограма (рис.3.47)

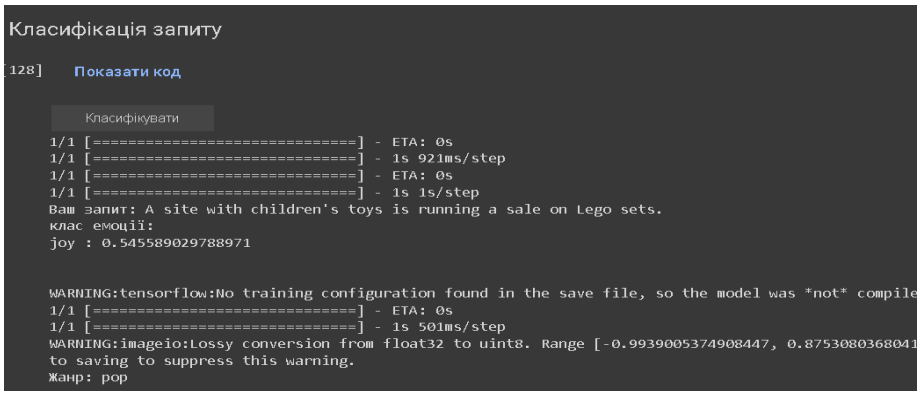

Рисунок 3.48 — Класифікація тесту 5.

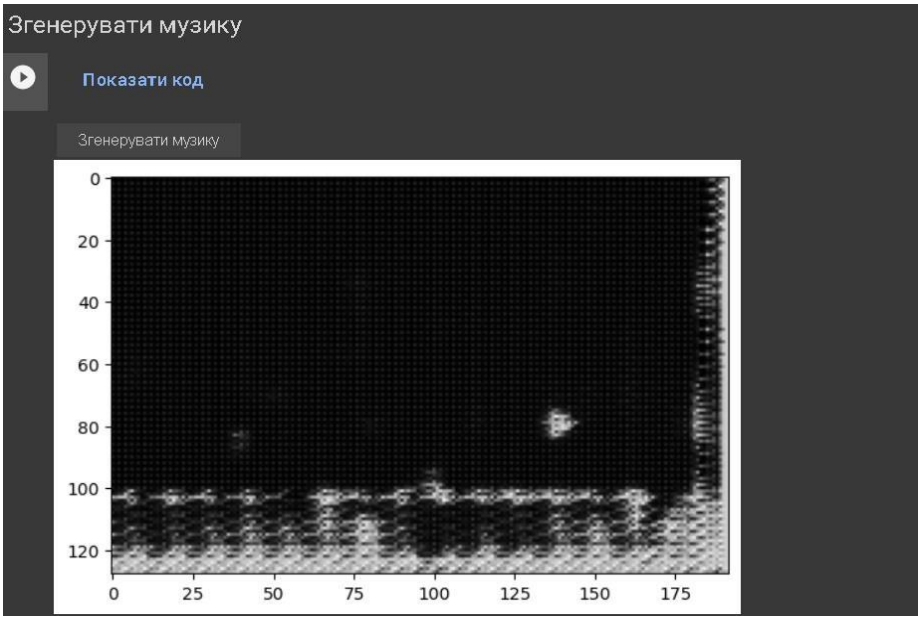

Рисунок 3.49 — Згенерована спектрограма тесту 5.

Результат роботи ПЗ: клас емоції — радість, музичний жанр — поп.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Xin-She Yang: Nature Inspired Computation and Swarm Intelligence
2. Alex Sherstinsky: Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. URL: https://arxiv.org/abs/1808.03314
3. Mike Schuster and Kuldip K. Paliwal: Bidirectional Recurrent Neural Networks. URL: https://deeplearning.cs.cmu.edu/F20/document/readings/Bidirectional%20Recurrent%20Neural%20Networks.pdf
4. Yelena Mejova: Sentiment Analysis: An Overview. навч.пос., 2009 р.
5. Review of Text Classification in Deep Learning / Qi Wang, Wenling Li, Zhezhi Jin // Open Access Library Journal. – Vol. 8 (03). – 2021. – P. 1-8. 5.
6. Kajal Kumari : Text Preprocessing techniques for Performing Sentiment Analysis! URL:https://www.analyticsvidhya.com/blog/2021/08/text-preprocessing-techniques-for-performing-sentiment-analysis/ (дата звернення: 27.08.2021).
7. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio: Generative Adversarial Networks. URL: https://arxiv.org/abs/1406.2661
8. Dalya Gartzman: Getting to Know the Mel Spectrogram. URL: https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0
9. Leland Roberts: Understanding the Mel Spectrogram. URL: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53
10. Jason Brownlee: How to Identify and Diagnose GAN Failure Modes. URL: https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/
11. Andrew Horner: The Correspondence of Music Emotion and Timbre in Sustained Musical Instrument Sounds.

URL:https://www.researchgate.net/publication/286187365_The_Correspondence_of_Music_Emotion_and_Timbre_in_Sustained_Musical_Instrument_Sounds (дата звернення: 06.10.2014).

12. Michael Nuzzolo: Music Mood Classification URL:https://sites.tufts.edu/eeseniordesignhandbook/2015/music-mood-classification/
13. Juan Sebastián Gómez-Cañón; Estefanía Cano; TuomasEerola; Perfecto Herrera; Xiao Hu; Yi-Hsuan Yang: Music Emotion Recognition: Toward new, robust standards in personalized and context-sensitive applications. URL: https://ieeexplore.ieee.org/document/9591555 (дата звернення: 27.10.2021).
14. How to Train a GAN? Tips and tricks to make GANs work / Soumith Chintala, Emily Denton, Martin Arjovsky, Michael Mathieu // NIPS 2016. URL: https://github.com/soumith/ganhac
15. Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, Thomas Hofmann: Stabilizing Training of Generative Adversarial Networks through Regularization. *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.* URL: https://proceedings.neurips.cc/paper/2017/file/7bccfde7714a1ebadf06c5f4cea752c1-Paper.pd
16. The Griffin-Lim algorithm: Signal estimation from modified short-time Fourier transform. URL: https://speechprocessingbook.aalto.fi/Modelling/griffinlim.html
17. Pranay Manocha, Zeyu Jin, Adam Finkelstein: Audio Similarity is Unreliable as a Proxy for Audio Quality. URL: https://arxiv.org/abs/2206.13411 (дата звернення: 08.06.2023).
18. Praveen: Emotions dataset for NLP. URL: https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp

19. Andara: GTZAN Dataset - Music Genre Classification. URL: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification
20. Anmol Kumar: Glove Embeddings. URL:https://www.kaggle.com/datasets/anmolkumar/glove-embeddings
21. GANSYNTH: ADVERSARIAL NEURAL AUDIO SYNTHESIS / J.Engel, K.K.Agrawal, S.Chen, I.Gulrajani, C.Donahue, A.Roberts // ICLR. – 2019. URL: https://arxiv.org/abs/1902.08710
22. Jason Brownlee: How to Train a Progressive Growing GAN in Keras for Synthesizing Faces. URL:https://machinelearningmastery.com/how-to-train-a-progressive-growing-gan-in-keras-for-synthesizing-faces/
23. Jupyter Widgets. URL:https://ipywidgets.readthedocs.io/en/stable/