МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження нейромережевого підходу для генерування музичних композицій
(тема)

Виконав:
Студент 2 курсу, групи ІПЗм-19-3

Шопинський М.В.
(прізвище, ініціали)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник доц. Голян Н.В.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ Дудар З.В.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність Інженерія програмного забезпечення
(код і повна назва спеціальності)

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

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. Кафедри _____
(підпис)
« 26 » 03 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Шопинського Максима Володимировича
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження нейромережевого підходу для генерування музичних композицій

затверджена наказом університету від 26.03.2021р. № 386Ст

2. Термін подання роботи до екзаменаційної комісії 11 05 2021р.

3. Вихідні дані до роботи Провести дослідження нейромережевого підходу для генерування музичних композицій, розробити нейронну мережу для генерування композицій на основі дослідженої моделі.

4. Перелік питань, що потрібно опрацювати в роботі мета роботи, постановка задачі, дослідження нейронних мереж для генерування музики, побудова моделі, розробка архітектури мережі, аналіз отриманих результатів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, слайдів, ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Титульний лист, Актуальність дослідження, Мета роботи, Постановка задачі, Огляд досліджень, Моделі нейронних мереж, Рекурентна нейронна мережа, Мережа довгої короткочасної пам'яті, Структура нейромережевої моделі, Внутрішня структура LSTM, Технології реалізації, Структура програмного коду, Код моделі для тренування, Аналіз результатів, Апробація результатів

дослідження, Напрямки подальшої роботи, Висновки

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Дослідження нейронних мереж для генерування музичних композицій	доц. Голян Н.В.		08.04.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вивчення наявної літератури	03-04-2021	виконано
2	Вивчення технології	09-04-2021	виконано
3	Збір вхідних даних	12-04-2021	виконано
4	Проведення експериментів	21-04-2021	виконано
5	Підготовка пояснювальної записки	03-05-2021	виконано
6	Підготовка презентації та доповіді	04-05-2021	виконано
7	Нормоконтроль, рецензування	12-05-2021	виконано
8	Занесення диплому в електронний архів	13-05-2021	виконано
9	Допуск до захисту у зав. кафедри	17-05-2021	виконано

Дата видачі завдання 26 березня 2021р.

Студент _____
(підпис)

Керівник роботи _____ доц. Голян Н.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 70 с., 30 рис., 1 табл., 17 джер.

ГЕНЕРУВАННЯ, НЕЙРОННА МЕРЕЖА, МАШИННЕ НАВЧАННЯ, МОДЕЛЬ, МУЗИКА, KERAS, LSTM, PYTHON, RNN, TENSORFLOW.

Об’єктом дослідження є моделі нейронних мереж та модель довгої короткотривалої пам’яті.

Метою роботи є проектування та розробка моделі нейронної мережі, здатної розбирати музичні композиції та створювати нові на основі попереднього досвіду.

Методи проектування нейронної мережі базуються на технологіях Python, Keras та TensorFlow.

У результаті роботи спроектовано та розроблено модель нейронної мережі для генерування музичних композицій, проаналізовано результати роботи моделі, сформовані загальні рекомендації щодо використання моделі.

GENERATION, NEURAL NETWORK, MACHINE LEARNING, MODEL, MUSIC, KERAS, LSTM, PYTHON, RNN, TENSORFLOW.

The object of research are the models of neural networks and a model of long-term short-term memory.

The aim of the work is to design a model of a neural network capable of disassembling musical compositions and creating new ones based on previous experience.

Neural network design methods are based on Python, Keras, and TensorFlow technologies.

As a result of the work, the model of a neural network for generation of musical compositions is designed and developed, results of work of model are analyzed, the general recommendations concerning use of model are formed.

Я, Шопинський Максим Володимирович, студент гр. ІПЗм-19-3, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження нейромережевого підходу для генерування музичних композицій», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Сьогодні, з розвитком інформаційних технологій, значно зросла потреба у засобах штучного інтелекту та машинного навчання. Крім уже звичних класифікацій та розпізнавання, моделі машинного навчання починають широко застосовуватися для емуляції роботи людського мозку в плані креативу та наслідування.

Генерування музики – найбільш абстрактний тип творчості. На відміну від книг, віршів, картин та відео, музика не прив’язана до контексту навколишнього світу і не вимагає знання таких понять, як, наприклад, значення слів або антропоморфної фігури. І хоча сприйняття світу індивіда, безсумнівно, впливає на те, як він розуміє той чи інший твір, музика залишається найбільш незалежним способом емоційної взаємодії між людьми.

Більше того, музику можна представити як дуже простий формат даних без втрати основного вмісту. Цей факт був відомий з давніх часів і призвів до появи нот. Музичні ноти представлені в комп’ютері як послідовність двійкових векторів, що спрощує їх алгоритмічну обробку.

На цій основі музика здається потенційно продуктивним напрямком досліджень творчої діяльності та її моделювання у штучному інтелекті. Однак досі немає обґрунтованих наукових теорій, чому одні послідовності нот сприймаються як музика, а інші – ні. Питання полягає також у тому, чи ці залежності є природними і не формуються в ході історичного процесу. Так чи інакше, музика містить певні закономірності, які інтуїтивно зрозумілі всім людям, які ще не були математично чітко сформульовані чи змодельовані.

Метою дослідження є розробка моделі нейронної мережі, яка повинна розбирати музичні композиції та створювати нові на основі попереднього досвіду. Така модель є емуляцією реального композитора, базою якого є його музичний досвід. Крім того, необхідно провести аналіз наявних моделей генерування музичних композицій.

Для досягнення цієї мети було поставлено та вирішено такі задачі:

– провести аналіз наявних моделей для генерування музичних композицій;

– на основі проведеного аналізу розробити модель нейронної мережі для обробки та генерування композицій;

– на основі отриманих результатів дослідження створити рекомендації щодо розробки моделі нейронної мережі для генерування музичних композицій різних жанрів та стилів.

Об’єктом дослідження є моделі нейронних мереж та модель довгої короткотривалої пам’яті зокрема.

Предметом дослідження в роботі є можливості вживання нейромережевих моделей типу довгої короткотривалої пам’яті для створення музичних творів на базі тестового набору даних.

Для досягнення мети дослідження необхідно розробити штучну нейронну мережу, здійснюючи аналіз ефективності генерування композицій цією мережею й поступово ускладнюючи її до досягнення необхідних результатів якості точності генерування.

Здійснена робота дозволила отримати нейронну мережу з використанням моделі довгої короткотривалої пам’яті для вирішення задачі генерації музичних композицій. Отриманий результат дозволив зробити висновки щодо ефективності нейронних мереж з довгою короткотривалою пам’яттю на даних музичної предметної області, зробити загальні рекомендації із проектування такої моделі для вирішення подібних задач. Крім того, було обрано технологію для програмної реалізації нейронної моделі, а також розроблено програмний код моделі.

За результатами дослідження було опубліковано статтю в рамках конференції PIC S&T`2020.

1 АНАЛІЗ СТАНУ ВИРІШЕННЯ ПРОБЛЕМИ

1.1 Огляд моделей для генерування композицій

За останнє десятиліття автоматизація створення музики була здійснена кількома різними способами. Одними з найдавніших методів були системи, засновані на правилах (граматики) та алгоритми, що використовують випадковість, як алгоритм «риффології» [1], представлений Ленгстоном (1988).

Однією з популярних форм граматик, що використовувалися для генерування музики, була L-Systems. Ці граматики переписування рядків могли створювати відносно прості послідовності нот. Зосередження уваги на музичній композиції та генерації музики в останній час перейшло до машинного навчання та методів штучного інтелекту. Це пояснюється здатністю нейронних мереж та глибокого навчання виробляти менш передбачувані та більш мелодійно складні мелодії, ніж їхні аналоги з випадковим алгоритмом та граматичними системами.

Більшість музичних генераторів за останні кілька років були створені за допомогою рекурентних нейронних мереж (Recurrent neural network, RNN) [2] та мереж довгої короткотривалої пам’яті (Long short-term memory, LSTM) [3].

RNN мають повторювані зв’язки всередині прихованих шарів між попереднім і поточним станами в нейронній мережі. Це означає, що модель зберігає інформацію у формі активації, тим самим пропонуючи своєрідну пам’ять. Ця можливість зберігання робить її дуже корисною для таких речей, як обробка мовлення та композиція музики. Основна проблема звичайної RNN полягає в тому, що вона зберігає інформацію лише про свій попередній стан. Це означає, що контекст поширюється лише на одне покоління назад. Така поведінка є не дуже корисною для музичної композиції, де початок пісні відіграє роль в середині та в кінці.

Мережі LSTM вирішують проблему відсутності довготривалої пам’яті в звичайних RNN. Вони додають спеціальний тип комірок з іменованою

пам’яттю. Ці комірки пам’яті складаються з трьох воріт: введення, виведення та забуття. Вхідний вентиль контролює, скільки даних вводиться в пам’ять, вихідний вентиль управляє даними, що передаються наступному шару, а вентиль забуття – втратою або розривом збереженої пам’яті. Цей вентиль містить функції сигмоїдної та гіперболічної дотичних з низькою обчислювальною складністю O(1), що означає, що вона не буде повільною під час тренування.

1.2 Дослідження у сфері автоматизованого створення музики

Досліджувані мережі відрізняються кодуванням вхідних даних із форматів MIDI та деякими аспектами структури нейронної мережі. Ек і Шмідхубер у 2002 році першими перейшли від звичайних рекурентних нейронних мереж до LSTM, щоб створити блюзову музику (жанр музики, подібний до джазу, що містить багато гітарних сольних нот, відомих як рифи). Мережа успішно створила музичні твори з належною структурою, що нагадує звичайну блюзову музику. Вони не зіставляли це з файлами MIDI, а натомість використовували власний формат для представлення послідовностей нот. Їх мережа LSTM виявилася здатною вивчати акордові послідовності та розпізнавати структуру акордів блюзу.

Скулі у 2017 році створив музичний генератор [4] із набором даних, що складається з треків відеоігри Final Fantasy. Він використовував Keras, (високорівневий API глибокого навчання Python, який працює поверх Tensorflow), що спрощує взаємодію з бібліотекою машинного навчання. Кодування файлів MIDI на входи для мережі здійснювалося за допомогою набору інструментів Python під назвою Music21.

Скулі ввів послідовності об’єктів нот і акордів використовував мережу для прогнозування наступного об’єкта (ноти або акорду) у межах

послідовності. Його результати були відносно хорошими, оскільки музика мала структуру і дуже мало атональних нот.

Інший підхід до використання LSTM для музики був зроблений Уолдером та Кімом у 2018 році. Вони з'ясували, що людська музика багато в чому схожа сама на себе, оскільки мотив повторюється і трансформується у всій пісні [5]. Мотив – це тема, виражена як коротка музична фраза (послідовність нот), яка повторюється протягом певної пісні. Метою їхньої мережі було розпізнавання та трансформації мотивів у пісенній послідовності, використовуючи концепції транспонування та зміщення. Цей підхід призвів до більш високого рівня розпізнавання закономірностей у музичній послідовності.

Більш успішним, але менш автоматизованим підходом до використання мереж LSTM для створення музики є початок із заздалегідь визначеного музичного ритму та використання мережі для вибору висоти нот у ній. Це було зроблено Уолдером у 2016 році під час дослідження імовірнісого моделювання музичних даних. Одним із методів, який він використав для розширення набору даних, було транспонування навчальних даних у всі 12 ключів, що він зробив, щоб уникнути нетривіальної проблеми розпізнавання музичних клавіш у мережі.

1.3 Постановка задачі

Задача роботи – створити модель, яка зможе вивчати різні музичні стилі, а потім генерувати нові композиції на основі вивчених раніше творів. Модель надає функцію з можливістю запам'ятовувати час та короткі фрагменти інформації, щоб використовувати їх для майбутніх прогнозів.

Нейронна мережа з глибоким навчанням використовує двоосьову модель LSTM, яка навчається із згортковим ядром, для створення поліморфного музичного твору.

Отримана мережева структура LSTM має спільні риси з нейронною мережею Скулі та Уолдера. Відмінності полягають у кодуванні даних, типах шарів та функціях активації.

Для підвищення глобальної послідовності музичних композицій використовується підхід глибокого підкріплення [6]. Цей підхід добре працює для кількісного та якісного аналізу при створенні поліфонічної музики.

Мережа повинна мати різну тривалість нот для збільшення музичної глибини та складності. Основною відмінністю підходу до кодування даних MIDI має бути інтерпретація акордів. Спеціальний метод кодування повинен збільшити складність моделі завдяки використанню вікна для захоплення заданих нот протягом інтервалу часу, не роблячи різниці між окремими нотами або акордами.

2 ОПИС МОДЕЛІ ГЕНЕРУВАННЯ КОМПОЗИЦІЙ

2.1 Основи музичної теорії для створення моделі

Розуміння того, як повинен працювати музичний генератор, вимагає певних знань музичної теорії [7]. Висота звуку використовується для опису високого чи низького звуку. Музика має сім нот – A, B, C, D, E, F, G (німецька система визначення). Кожен набір нот (від A до G) складає октаву. Для позначення висоти звуку, в якій розміщуються ноти, використовуються октави. Наприклад, нота A1 (A в октаві 1) знаходиться глибше та звучить нижче, ніж A7 (A в октаві 7), що відрізняється вищим тоном і чіткішим звуком. Звичайна клавіатура для фортепіано складається з 88 клавіш і семи цілих октав.

Більшість музичних творів має тон. Тон використовується для визначення діапазону нот, що складають гармонію музичного твору. Він визначає ноту, яка є основою композиції. Твір, що звучить у тоні E, означає, що нота E є основою музики в пісні.

Гама – це певний набір нот, побудований на основі ноти та гармонії. Це можна охарактеризувати як певний набір нот, за допомогою яких створюється композиція. Набір нот відрізняється один від одного тим, що деякі більше підходять для яскравішої музики (мажорні гами), а деякі краще для сумних композицій (мінорні гами).

Акорд – це група нот, що звучать одночасно, створюючи окремий звук як група. Наприклад, акорд до мажор складається з нот C, E і G.

Транспозицією у музиці називається процес переміщення мелодії, гармонійної прогресії або цілого музичного твору в іншу тональність, зберігаючи ту саму тональну структуру. Це означає, що структура пісні зберігається під час зміни тону.

Найпоширенішим комп’ютерним представленням музичних партитур є формат цифрового інтерфейсу музичних інструментів (MIDI). Файл складається з набору команд різного типу. Основний тип – це команда ноти,

параметрами якої є три значення: висота, гучність (дискретні числа від 0 до 127) та час, який повинен пройти з моменту виконання попередньої команди. Так само існує команда припинення звучання ноти. Важливо, щоб події, що відбуваються одночасно (наприклад, натискання клавіш одного акорду), могли йти у файлі в довільному порядку.

Час зазвичай вимірюється в мілісекундах і може бути заданий під час запису в режимі реальному часі у форматі MIDI. Однак у традиційних нотних записах тривалість нот необхідно вказувати як частку деякого заздалегідь визначеного інтервалу часу. При строгій домовленості в MIDI час між нотами кратний певному мінімальному періоду.

2.2 Огляд моделей нейронних мереж

Згорткова нейронна мережа (Convolutional neural network, CNN) – це клас глибокої нейронної мережі, який найчастіше застосовується для аналізу візуальних зображень. CNN – це регуляризована версія багатошарових перцептронів. Багатошарові персептрони зазвичай означають повністю зв’язані мережі, тобто кожен нейрон одного шару пов’язаний з усіма нейронами наступного шару. Повна зв’язність цих мереж робить їх схильними до перенавчання.

Згорткові мережі були натхненні біологічними процесами, оскільки модель зв’язку між нейронами нагадує організацію зорової кори тварини. Окремі коркові нейрони реагують на подразники лише в обмеженій області поля зору, відомій як рецептивне поле. Сприйнятливі поля різних нейронів частково перекриваються так, що покривають все поле зору.

CNN використовують порівняно мало попередньої обробки порівняно з іншими алгоритмами класифікації зображень [8]. Це означає, що мережа вчиться оптимізувати фільтри (або ядра) за допомогою автоматизованого

навчання, тоді як у традиційних алгоритмах ці фільтри виготовляються вручну. Ця незалежність від попередніх знань та втручання людини у видобуток ознак є головною перевагою.

Згорткова мережа може вивчати характеристики 2D-даних та виконувати вилучення та класифікацію ознак. CNN уникає явного вилучення характеристик, але неявно вивчає особливості з навчальних даних. Ваги нейронів на одній і тій же поверхні відображення об’єктів однакові, тому мережа може вчитися паралельно. Моделюючи музичну послідовність, CNN може ідентифікувати та виділити потенційні мелодійні особливості для вдосконалення моделі.

Рекурентна нейронна мережа (Recurrent neural network, RNN) – це штучна нейронна мережа, що використовує послідовну інформацію. ЇЇ називають рекурентною, оскільки вона виконує однакову функцію для кожного елемента послідовності, а результат залежить від попередніх обчислень.

RNN є повторюваною за своєю суттю, оскільки вона виконує однакову функцію для кожного введення даних, тоді як вихід поточного входу залежить від минулого обчислення. Після отримання вихідних даних вихід копіюється і відправляється назад у періодичну мережу. Для прийняття рішення мережа враховує поточний вхід і вихід, який було отримано з попереднього вводу.

На відміну від нейронних мереж прямого зв'язку, RNN можуть використовувати свій внутрішній стан (пам'ять) для обробки послідовностей входів. Це дозволяє застосовувати їх до таких задач, як несегментоване розпізнавання рукописного вводу чи мови.

Переваги рекурентних нейронних мереж:

– мережа може моделювати послідовність даних так, що можна вважати, що кожна вибірка залежить від попередніх;

– мережі можуть використовуватися із згортковими шарами.

Недоліки рекурентних нейронних мереж:

– градієнтні проблеми зникнення та вибуху;

– довгий час тренування та необхідність мати потужний комп’ютер;

– нездатність обробляти довгі послідовності при використанні функції активації tanh або relu.

Структуру RNN наведено на рисунку 2.1.

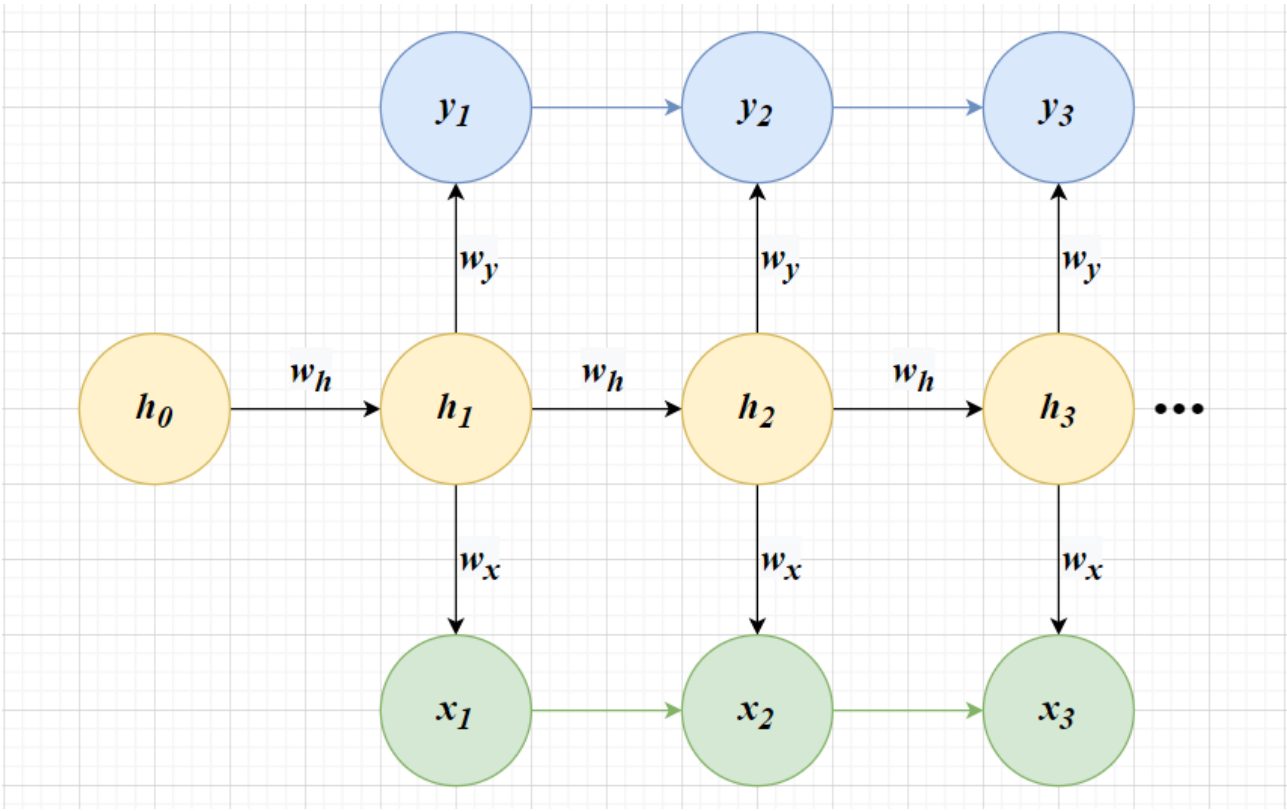

Рисунок 2.1 – Структура рекурентної нейронної мережі

Мережа довгої короткочасної пам’яті (Long short-term memory, LSTM) – це тип моделі RNN, яка може ефективно навчатись за допомогою градієнтного спуску. За допомогою механізму затвору LSTM здатні розпізнавати та кодувати довгострокові шаблони [9]. LSTM дуже корисні для вирішення проблем, коли мережа повинна запам’ятовувати інформацію протягом тривалого періоду часу, як це має місце при генеруванні музики та тексту.

LSTM – це модифікована версія періодичних нейронних мереж, що полегшує запам'ятовування минулих даних у пам'яті. Тут вирішена проблема зникаючого градієнта RNN. LSTM добре підходить для класифікації, обробки та прогнозування часових рядів з урахуванням часових лагів невідомої

тривалості. Мережа тренує модель за допомогою зворотного розповсюдження. У мережі LSTM присутні три типи вентилів (див. рис. 2.2):

– вхідний вентиль (Input Gate), що використовуються для модифікації пам'яті. Сигмоїдна функція вирішує, які значення пропускати 0,1. і функція tanh надає зважування значенням, які передаються, визначаючи рівень їх важливості в діапазоні від -1 до 1;

– забуваючий вентиль (Forget Gate), що визначають, які деталі слід викинути з блоку за допомогою сигмовидної функції. Вентиль розглядає попередній стан та вхідний вміст і виводять число від 0 до 1 для кожного числа у стані комірки;

– вихідний вентиль (Output Gate), що використовують вхід і пам'ять блоку для вирішення виводу. Сигмоїдна функція вирішує, які значення пропускати 0,1, а функція tanh надає ваги значенням, які передаються, визначаючи рівень їх важливості в діапазоні від -1 до 1.

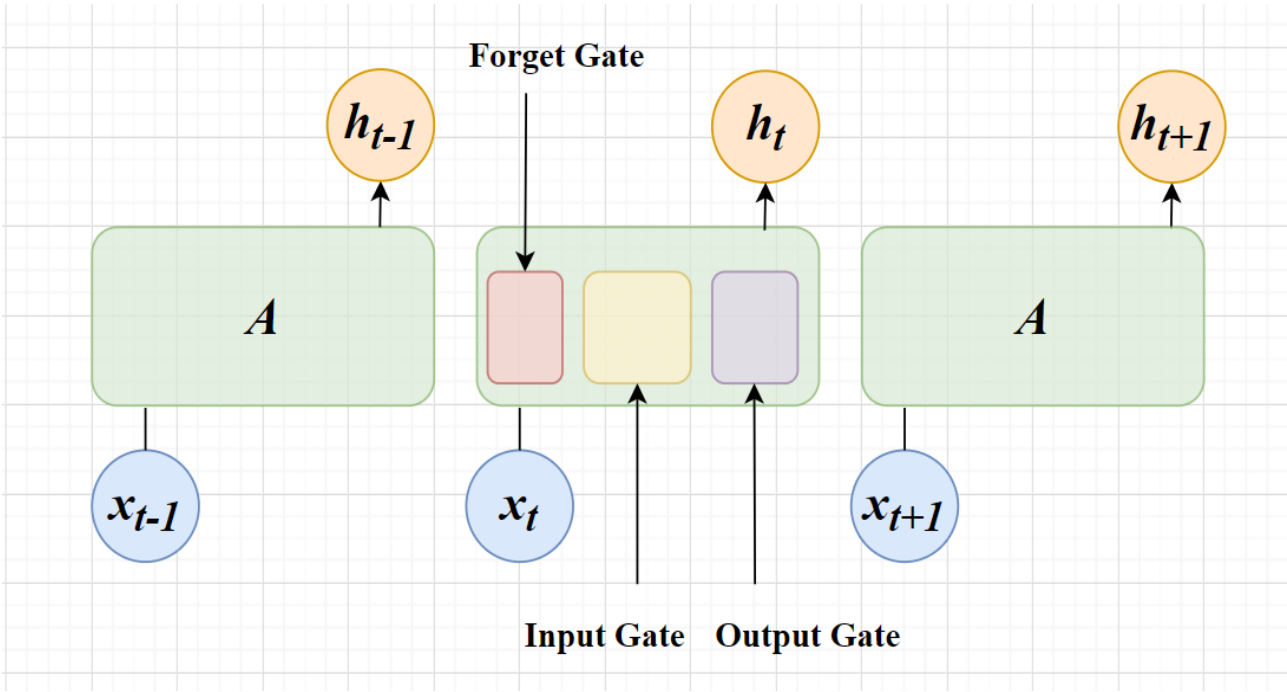

Рисунок 2.2 – Структура мережі LSTM

LSTM спеціально розроблені, щоб уникнути проблеми довготривалої залежності, зберігаючи значення як на короткий, так і на довгий проміжок часу.

Це відбувається за рахунок того, що LSTM не використовує функцію активації всередині своїх періодичних компонентів [10]. Таким чином, збережене значення не розмивається в часі і градієнт не зникає при використанні методу зворотного розповсюдження помилок у часі під час мережевого навчання.

Ключовими компонентами модуля LSTM є стан комірки та різні фільтри. Про стан комірки можна говорити як про мережеву пам’ять, яка передає відповідну інформацію по всьому ланцюжку модулів. Таким чином, навіть інформацію з ранніх етапів часу можна отримати на більш пізніх, нейтралізуючи ефект короткочасної пам’яті.

Таким чином, RNN добре підходять для обробки даних послідовностей для генерування композицій, але їх недоліком є короткочасна пам’ять. LSTM була створена як метод пом'якшення короткочасної пам’яті за допомогою механізмів, що називаються вентилями.

2.3 Модель для генерування музичних композицій

CNN має функцію розпізнавання та вилучення знаків, що дає змогу використовувати шар згортки для вилучення особливостей музичної мелодії. Після використання шару згортки для вилучення елемента матриці музичної партитури, витягнута карта об’єктів вводиться в LSTM для імітації часових рядів, а потім статус кожної ноти генерується наступного разу [11].

Слід зазначити, що рівень пулу, який зазвичай використовується в CNN, не може застосовуватися до цієї моделі, оскільки рівень пулу має інваріантний зсув, що може призвести до зміни тону або мелодійного циклу. Структура моделі згорткової LSTM (Convolution-LSTM, C-LSTM) показана на рисунку 2.3 (кількість квадратів на рисунку не відображає фактичну кількість даних).

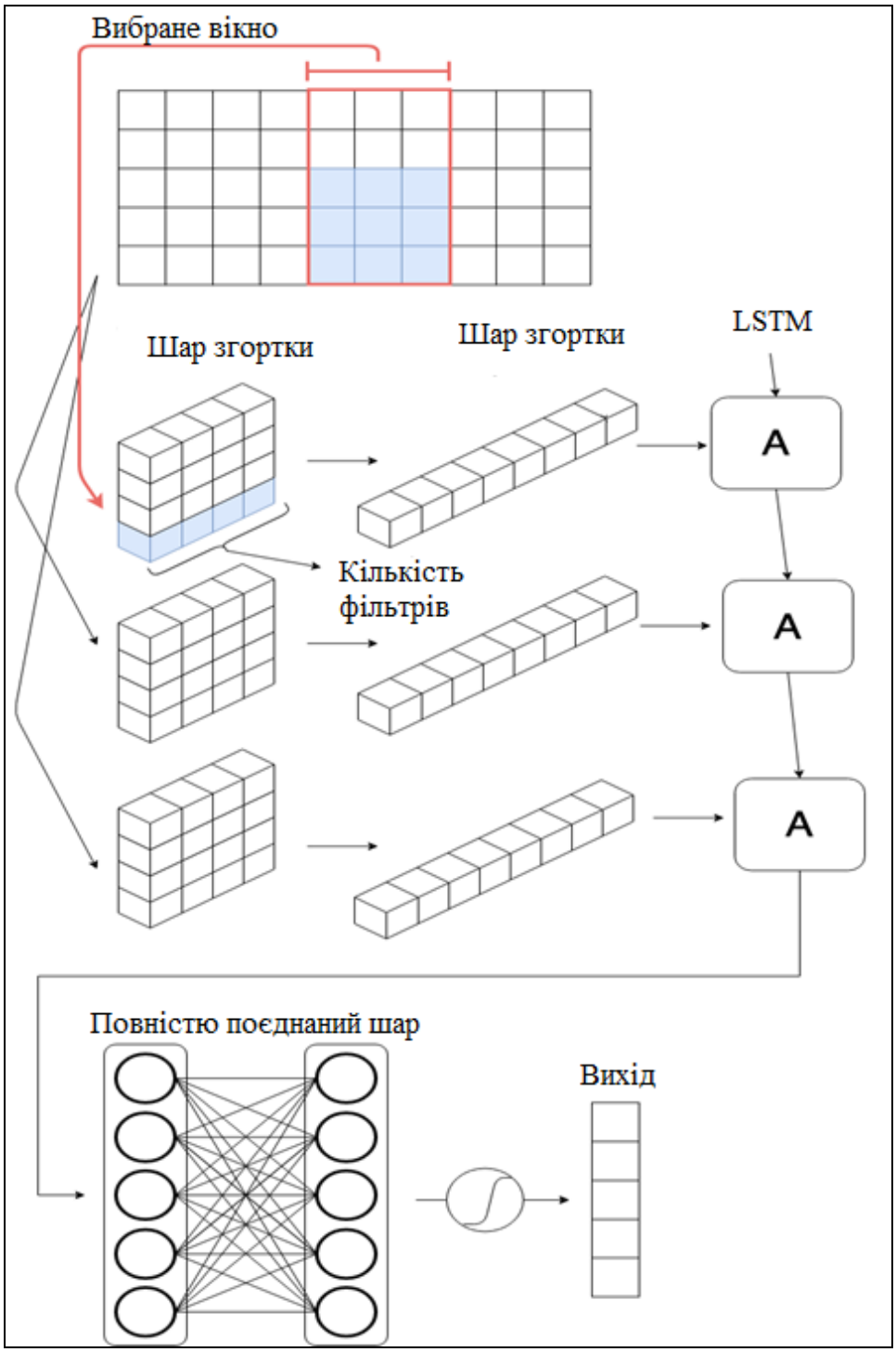

Рисунок 2.3 – Структура моделі згорткової LSTM

Вибираємо фіксовану довжину N у напрямку часу матриці, а потім вибираємо [t-N+1, t] як діапазон вікон у момент часу t. Вибране вікно буде повним входом для кожного тренування моделі.

Цей шар згортається з багатоядерним шаром у вибраному вікні. Нормальним результатом згортки має бути тривимірна матриця, її необхідно

розбити у напрямку осі часу та розділити на багато двовимірних матриць, щоб послідовно вводити до LSTM. Цей шар можна виразити формулою (2.1).

$$y_{i,j}^{(2l)'} = \theta(\sum_{m'}\sum_{n'} y_{i+m'-1,j+n'-1}^{(1l)} \cdot w_{m',n'}^{l'} + b^{l'}), \quad (2.1)$$

де θ – функція активації;

m, n – висота та ширина ядер згортки;

l – різні ядра згортки;

w – вага ядер згортки;

b^l – коефіцієнт зміщення.

У звичайній згортковій нейронній мережі згортковий шар може супроводжуватися шаром об’єднання, який може зменшити вилучення параметрів та ознак [12]. Однак пул має інваріантність зсуву, що дозволяє функції мати однаковий вихід у будь-якому положенні.

У музичному моделюванні однаковий вихід у будь-якому положенні безпосередньо призведе до модифікованого тону або циклічної мелодії, тому шар об’єднання тут використовувати не можна.

Другий згортковий шар використовується для зменшення параметрів та вилучення особливостей та розраховується за допомогою формули (2.2).

$$y_{i,j}^{(1)l} = \theta(\sum_{m}\sum_{n} x_{i+m-1,j+n-1} \cdot w_{m,n}^{l} + b^{l}), \quad (2.2)$$

де θ – функція активації;

m’, n’ – висота та ширина ядер згортки;

l’ – різні ядра згортки;

w – вага ядер згортки;

b^l – коефіцієнт зміщення.

LSTM отримує дані послідовно з попереднього рівня. Кількість вхідних даних кожного разу дорівнює кількості фільтрів у згортковому рівні 2. Після

отримання даних з попереднього рівня LSTM видасть результат. Вхідні дані LSTM можуть бути виражені за допомогою формули (2.3).

$$s_t = y_{i,t}^{(2)l'}, \tag{2.3}$$

де $t=j+n-1$ та $i=1$ (оскільки ядра шару згортки 2 покривають усі ноти по вертикальній осі).

Внутрішня структура LSTM та процес передачі інформації можуть бути виражені за допомогою формул (2.4-2.9).

$$f_t = \sigma(W_f \cdot [h_{t-1}, s_t] + b_f), \tag{2.4}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, s_t] + b_i), \tag{2.5}$$

$$\hat{C}_t = tanh(W_C \cdot [h_{t-1}, s_t] + b_C), \tag{2.6}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t), \tag{2.7}$$

$$i_t = \sigma(W_o \cdot [h_{t-1}, s_t] + b_o), \tag{2.8}$$

$$h_t = o_t \cdot tanh(C_t), \tag{2.9}$$

де f_t – забуваючий вентиль;

i_t – вхідний вентиль;

C_t – стан комірки в попередній момент;

$\hat{C}_t$ – стан поточної комірки;

o_t – вихідний вентиль;

h_t – вихід поточної комірки.

Кінцевий результат LSTM може бути виражений формулою (2.10).

$$Y^{(3)} = h_t \tag{2.10}$$

Повністю з’єднаний рівень отримує кінцевий вихід LSTM і виводить результат після одного повністю з’єднаного шару. Результатом є стан групи нот в момент часу t + 1, що виражений формулою (2.11).

$$y_v = \theta'(w_{u,v} \cdot y_u^{(3)} + b_{u,v}), \tag{2.11}$$

де y_v – кінцеві результати;

θ – функція активації;

w – вагові коефіцієнти;

y_u – виходи попереднього шару;

b – коефіцієнти зміщення.

Після завершення вищезазначеного процесу вибране вікно рухається на одну одиницю у напрямку до осі часу, і вищевказаний процес повторюється для навчальних або генераторських цілей.

Для мінімізації загальної похибки LSTM у всьому наборі послідовностей може бути використаний ітераційний спуск градієнта, такий як метод поширення зворотної помилки [13], розгорнутий з часом для зміни кожного з вагових коефіцієнтів пропорційно до його вихідного значення в залежності від значення помилки.

Основна проблема градієнтного спуску для стандартних рекурентних нейронних мереж полягає в тому, що градієнти помилок зменшуються з експоненціальною швидкістю із збільшенням затримки часу між важливими подіями. Однак у випадку блоків LSTM, коли значення помилок поширюються у напрямку, протилежному вихідному шару, помилка записується в пам’ять блоку. Це називається «каруселлю помилок», яка постійно «подає» помилку назад до кожного з клапанів, поки вони не навчаться відкидати значення. Таким чином, регулярне оброблення помилок дозволяє ефективно запам’ятовувати значення протягом дуже тривалого періоду часу для навчання модуля LSTM.

LSTM також можна навчити, використовуючи комбінацію еволюційного алгоритму для ваг у прихованих шарах та матриць псевдоповернення або методу опорних векторів для ваг у вихідному рівні [14].

3 АРХІТЕКТУРА НЕЙРОМЕРЕЖЕВОЇ МОДЕЛІ

3.1 Вибірка музичних даних

Для побудови нейромережевої моделі для генерування музичних композицій необхідно відібрати вхідні дані, необхідні для тренування.

Для тренування моделі було вибрано 20 музичних композицій сучасних українських авторів різних тональностей та темпу у форматі MIDI. Вибрані твори в основному мають мінорну гаму та ритм 4/4, що характерно для сучасної поп-музики.

Вибірку, із вказанням назви, темпу та тону кожної композиції наведено в таблиці 3.1.

Таблиця 3.1 Вибірка музичних даних для тренування моделі

Назва композиції	Темп	Тон	Ритм
ALEKSEEV – Как Ты Там	113	F мінор	4/4
KAZKA – Плакала	104	C мінор	4/4
MARUV – Siren Song	115	E мінор	4/4
MELOVIN – Oh No!	94	F# мінор	4/4
MELOVIN – That’s Your Role	120	F мінор	4/4
MELOVIN – Wonder	88	A мінор	4/4
MELOVIN – З Тобою, Зі Мною, І Годі	71	D мінор	4/4
MELOVIN – Не Одинокая	104	F мажор	4/4
MELOVIN – Світ В Полоні	62	A мінор	4/4
MELOVIN – Ты, Ты, Ты	76	D мінор	4/4
Pianобой – Ведьма	149	E мінор	4/4
Pianобой – Родина	75	E мінор	4/4
The Hardkiss – Кораблі	90	E мінор	4/4
Антитіла – TDME (Там Де Ми Є)	90	D# мінор	4/4

Кінець таблиці 3.1

Назва композиції	Темп	Тон	Ритм
Время и Стекло – Е,Бой	105	D# мінор	4/4
Один в каное – Човен	84	C# мінор	4/4
Океан Ельзи – Без Бою	82	B мінор	4/4
Океан Ельзи – Без Тебе	74	F мінор	4/4
Скрябін – Старі Фотографії	80	D мінор	4/4
Христина Соловій – Тримай	124	E мінор	3/4

3.2 Вибір технологій реалізації

У якості мови програмування для реалізації нейромережевої моделі було вибрано Python. Ця мова має обширний набір бібліотек для машинного навчання та побудови моделей нейронних мереж. Крім того, Python є легкою мовою для навчання, що пришвидшує процес розробки моделі.

У процесі роботи було використано Python-бібліотеки, такі як Music21 та Keras з TensorFlow.

Music21 – це бібліотека Python, що використовується для роботи з музикою, що підтримується комп’ютером. Набір інструментів надає простий інтерфейс для роботи з файлами MIDI. Крім того, інструментарій дозволяє нам створювати ноти та акорди для генерування нових файлів MIDI. У програмній системі Music21 використовується для обробки вмісту файлів MIDI, отримання вихідних даних для нейронної мережі та їх конвертації у нотну форму.

Keras – це високорівневий API нейронних мереж, який спрощує взаємодію з TensorFlow. Він був розроблений з акцентом на швидкі дослідження та тренування моделей. У програмній системі бібліотека Keras використовується для створення та навчання моделі LSTM. Після того, як

модель буде натренована, вона буде використана для створення нотних позначень згенерованих композицій.

TensorFlow – це бібліотека з відкритим кодом для чисельних обчислень та масштабного машинного навчання. TensorFlow поєднує в собі безліч моделей машинного навчання, глибокого навчання (нейронних мереж) та алгоритмів. TensorFlow використовує Python, щоб забезпечити зручний інтерфейсний API для побудови програм із фреймворком, виконуючи ці програми у високопродуктивному середовищі C++.

TensorFlow може навчати та запускати глибокі нейронні мережі для рукописної класифікації цифр, розпізнавання зображень, вбудовування слів, машинного перекладу, обробки природної мови та генерування послідовностей (таких як музичні композиції).

3.3 Структура коду нейромережевої моделі

Структура коду була розділена на два основні завдання: тренування нейромережевої моделі та використання навченої моделі для створення нових файлів MIDI з якісним оцінюванням [15]. Обидві функції використовують один і той же графік моделі в різних контекстах: задача тренування ітеративно вводить дані в модель, виконує модель для усіх кроків часу та нот, що наявні у даних, а потім виводить тензор, де зворотна сигмоїда показує ймовірність того, що конкретна нота на даному етапі часу буде відтворена.

Логарифмічна ймовірність вхідних даних – це здатність моделі отримувати як вхідні дані вектор ноти на даному кроці часу та передбачати набір нот на наступному кроці часу. Функція втрат обчислює перехресну ентропію між сформованими журналами та партією.

Під час задачі генерування музики модель ітеративно проходить одноразовий крок, кожного разу повертаючи згенеровані зразки як вхід для

наступного кроку часу. Ці зразки накопичуються, створюючи тензор сформованих зразків як партії випадкової тривалості. Після цього згенеровані партії перетворюються у MIDI-файли за допомогою функцій пост-обробки для якісної оцінки.

Тренування моделі починається з вхідного ядра. Він приймає музичну композицію у якості вхідних даних і для кожної пари нота-пауза створює розширений вектор, який складається з номера MIDI ноти, вектора висоти тону ноти, відтворення значення відносно ноти (ефективний аспект ядра моделі згортки), вектор суми всіх отриманих нот у кожному класі тонів та вектор із двійковими числами, що представляють 16-значне положення ноти в такті.

На другому етапі, що називається кроком часу LSTM, комірка LSTM проходить вздовж осі часу для визначення тривалості вимірювання часу композиції. Ця операція виконується над вектором розкриття для кожної ноти паралельно зв'язаними вагами. Ця частина моделі фіксує послідовні шаблони музики і, у поєднанні з вхідним ядром, підтримує постійну конвертацію завдяки відносному вікну введення нот та прив'язаним вагам LSTM для всіх нот. Через ці пов'язані ваги розрахунки можна робити паралельно між нотами та між зразками матриці стану нот як з окремими ефективними композиціями. Єдиний необхідний послідовний аспект знаходиться вздовж осі часу. Упродовж тренування можна запустити будь-яку кількість каскадних комірок LSTM. Після кожної комірки застосовується випадаюча маска.

Завершальним етапом у тренуванні моделі є потенційно один або кілька тимчасових шарів LSTM із відсівом після кожного шару. Однак замість того, щоб проходити послідовно вздовж осі часу, цей етап проходить послідовно вздовж осі ноти. Крім того, цей етап включає зворотні згенеровані вибірки у свої вхідні дані. Після кожного «кроку ноти» комірка LSTM створює пару записів, що представляють зворотну сигмоподібну ймовірність генерування відтворення для цієї ноти. Створена пара зразків у ноті n-1, що підключена до входу каскаду LSTM за часом у ноті n, повертається назад до входу LSTM для етапу n. Отриманий зворотний зв'язок розраховує умовну ймовірність для

кожної отриманої ноти на основі фактичних значень, створених для інших нот. Це допомагає запобігти одночасному звучанню дисонансних нот. Кінцевими вихідними тензорами графічної моделі є партія записів та відповідні згенеровані зразки, які використовуються для навчання та генерації музики відповідно.

3.4 Підготовка даних

Першим кроком для генерування композицій є підготовка вхідних даних. MIDI файли для тренування моделі складаються з партій одного музичного інструменту – фортепіано. Партія має два типи об’єктів: ноти та акорди. Об'єкти нот містять інформацію про висоту, октаву та зміщення ноти, об’єкти акордів є контейнером для набору нот, які відтворюються одночасно.

При читанні MIDI-файлу для подання партитури музики за допомогою матриці потрібні дискретизація та кількісна оцінка. Для різних нот встановлюється вертикальна вісь матриці та горизонтальна вісь часу. У якості одиниці часу для вибірки файлу MIDI використовується 1/8 секунди. Ноти мають ублеві позначення, де 1 означає, що музична нота була ввімкнена, а 0 – вимкнена.

Для точного створення композицій нейронна мережа повинна мати можливість передбачити, яка нота чи акорд буде наступною. Це означає, що масив передбачень повинен містити кожну ноту та акорд-об’єкт, які зустрічаються у тренувальному наборі.

Крім того, важливим компонентом обробки вхідних даних є паузи між нотами та акордами. Паузи роблять музичну композицію динамічною та усувають уривки монотонності. Для розрахунку інтервалу між кожною нотою та акордом обчислюється зміщення об’єкта відносно попереднього.

Дані кожної вхідної MIDI-композиції заносяться до масиву та обробляються за допомогою бібліотеки Music21. Використовуючи конвертер бібліотеки отримується список усіх нот і акордів у файлі. До цих даних додається висоту звуку кожного об’єкта. Акорди додаються за допомогою кодування кожної ноти у цьому акорді. Ці значення дозволяють декодувати вихідні дані, створені мережею, у правильні ноти та акорди.

Наступним кроком підготовки даних є створення послідовностей, які будуть вхідними даними для нейронної мережі. Спочатку необхідно створити функцію відображення для перетворення категоріальних даних на основі рядків у числові дані на основі цілих чисел. Це робиться тому, що нейронна мережа якісніше працює з цілочисельними, ніж категоріальним даними на основі рядків. Приклад категоріального чисельного перетворення можна побачити на рисунку 3.1.

Далі потрібно створити вхідні послідовності для мережі та їх відповідні виходи. Результатом для кожної вхідної послідовності буде перша нота або акорд, що йде після послідовності нот вхідної послідовності у списку нот.

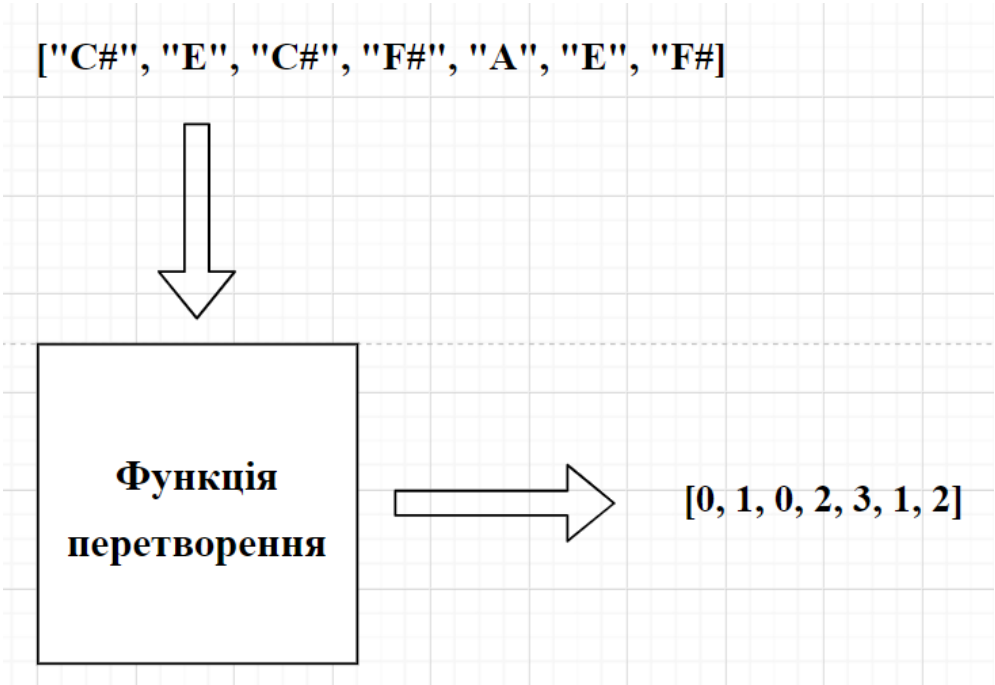

Рисунок 3.1 – Принцип роботи функції перетворення вхідних даних

Для тренування моделі було визначено довжину кожної послідовності рівною 100 нотам або акордам. Це означає, що для прогнозування наступної ноти в послідовності мережа обробляє попередні 100 нот, які допоможуть зробити прогноз.

Завершальним етапом підготовки даних для мережі є нормалізація вхідних даних та кодування виходу.

3.5 Тренування моделі

Для тренування моделі використовуються наступні типи нейромережевих модулів:

– модуль LSTM – це рекурентний шар, що приймає послідовність як вхідні дані та повертає послідовності об’єктів або матрицю;

– модуль Dropout – це техніка регуляризації, яка полягає у встановленні частки вхідних одиниць на 0 при кожному оновленні під час тренування для запобігання перенавчанню. Частка визначається параметром, що використовується з шаром;

– модуль Dense – це повністю зв’язані шари нейронної мережі, де кожен вхідний вузол підключений до кожного вихідного вузла;

– модуль Activation, що визначає, яку функцію активації використовуватиме нейромережа для обчислення виходу вузла.

Код створення моделі із зазначенням модулів наведено на рисунку 3.2.

Для кожного модулю LSTM, Dense та Activation перший параметр – це кількість вузлів, які повинен мати модуль.

Для модуля Dropout першим параметром є частка вхідних одиниць, яку слід скинути під час навчання.

```
# Створення моделі нейронної мережі
def create_network(model_input, pitches_range):
    network_model = Sequential()
    network_model.add(LSTM(
        512,
        input_shape=(model_input.shape[1], model_input.shape[2]),
        recurrent_dropout=0.3,
        return_sequences=True
    ))
    network_model.add(
        LSTM(512, return_sequences=True, recurrent_dropout=0.3,)
    )
    network_model.add(LSTM(512))

    network_model.add(BatchNorm())
    network_model.add(Dropout(0.3))
    network_model.add(Dense(256))
    network_model.add(Activation('relu'))

    network_model.add(BatchNorm())
    network_model.add(Dropout(0.3))
    network_model.add(Dense(pitches_range))
    network_model.add(Activation('softmax'))

    network_model.compile(
        loss='categorical_crossentropy',
        optimizer='rmsprop'
    )

    return network_model
```

Рисунок 3.2 – Код моделі генерування композицій

Для першого шару необхідно надати унікальний параметр, який інформує мережу про форму даних, які він буде навчати. Останній шар повинен завжди містити ту саму кількість вузлів, що і кількість різних виходів системи. Це гарантує, що результат мережі буде безпосередньо відображатись у класах.

Для програмної системи було використано просту мережу, що складається з трьох модулів LSTM, трьох модулів Dropout, двох модулів Dense та одного модулю Activation.

Для обчислення втрат кожної ітерації навчання було використано категоріальну перехресну ентропію, оскільки кожен з результатів належить лише одному класу та є більше двох класів для роботи.

Для оптимізації мережі було використовувати оптимізатор RMSprop, що підходить для періодичних нейронних мереж.

Код тренування моделі наведено на рисунку 3.3.

```
# Тренування моделі нейронної мережі
def train(model, model_input, output):
    file_path = "final-weights--{epoch:02d}-{loss:.4f}.hdf5"
    model_checkpoint = ModelCheckpoint(
        filepath=file_path,
        monitor='loss',
        verbose=0,
        save_best_only=True,
        mode='min'
    )

    callbacks = [model_checkpoint]
    model.fit(model_input, output, epochs=250, batch_size=128,
    callbacks=callbacks)
```

Рисунок 3.3 – Код тренування нейромережевої моделі

Після того, як було визначено архітектуру нейронної мережі, потрібно провести тренування моделі. Для цього використовується функція model.fit у Keras. Перший параметр функції – це список вхідних послідовностей, які було підготовано раніше, а другий – список відповідних результатів. Тренування моделі відбувається упродовж 200 епох (ітерацій), причому кожна партія, що розповсюджується через мережу, містить 64 об’єкти.

Для надання можливості припинення навчання в будь-який момент часу, не втрачаючи результатів роботи, було використано контрольно-пропускні пункти. Модельні контрольні точки забезпечують спосіб збереження ваг мережевих вузлів у файл після кожної епохи. Це дозволяє зупинити роботу нейронної мережі відразу після отримання достатньої точності результатів.

3.6 Генерування музичних композицій

Для того, щоб мати можливість використовувати нейронну мережу для генерування музичних композицій, необхідно перевести її в попередній стан. Для підготовки даних та налаштування мережевої моделі використовуються ті ж самі параметри, що й для навчання моделі за винятком того, що замість тренування мережі до моделі завантажуються ваги, що було збережено під час тренування.

Після завантаження вагів натреновану модель можна використовувати для створення нот. Оскільки модель має повний список послідовностей нот, необхідно вибирати випадковий індекс у списку як вихідну точку. Це дозволяє повторювати код генерації, не змінюючи нічого, і кожного разу отримувати різні результати. Для того, щоб контролювати початкову точку, необхідно замінити випадкову функцію аргументом командного рядка.

Крім того, потрібно створити функцію відображення для декодування вихідних даних мережі. Ця функція перетворить числові дані на категоріальні (цілі числа на ноти).

Мережа генерує 500 нот, що відповідає приблизно двом хвилинам музики і дає мережі достатньо місця для створення мелодії.

Код генерування нот на основі проведеного тренування нейромережевої моделі наведено на рисунку 3.4.

```
# Генерування нот на основі тренування
def generate_midi_notes(network_model, model_input, pitch_names,
pitch_range):
    start_index = numpy.random.randint(0, len(model_input)-1)
    converted_notes = dict((num, item) for num, item in enumerate
    (pitch_names))

    model_pattern = model_input[start_index]
    model_output = []

    for note_index in range(500):
        predict_input = numpy.reshape(model_pattern, (1, len
        (model_pattern), 1))
        predict_input = predict_input / float(pitch_range)

        model_prediction = network_model.predict(predict_input,
        verbose=0)
        prediction_index = numpy.argmax(model_prediction)

        prediction_result = converted_notes[prediction_index]
        model_output.append(prediction_result)

        model_pattern.append(prediction_index)
        model_pattern = model_pattern[1:len(model_pattern)]

    return model_output
```

Рисунок 3.4 – Код генерування нот на основі тренування моделі

Для кожної ноти, яку необхідно створити, необхідно подати послідовність в мережу. Перша послідовність, яку необхідно подати, – це послідовність нот з початкового індексу. Для кожної наступної послідовності, яка використовується як вхід, необхідно видаляти першу ноту послідовності та вставляти результат попередньої ітерації в кінець послідовності, як це зображено на рисунку 3.5.

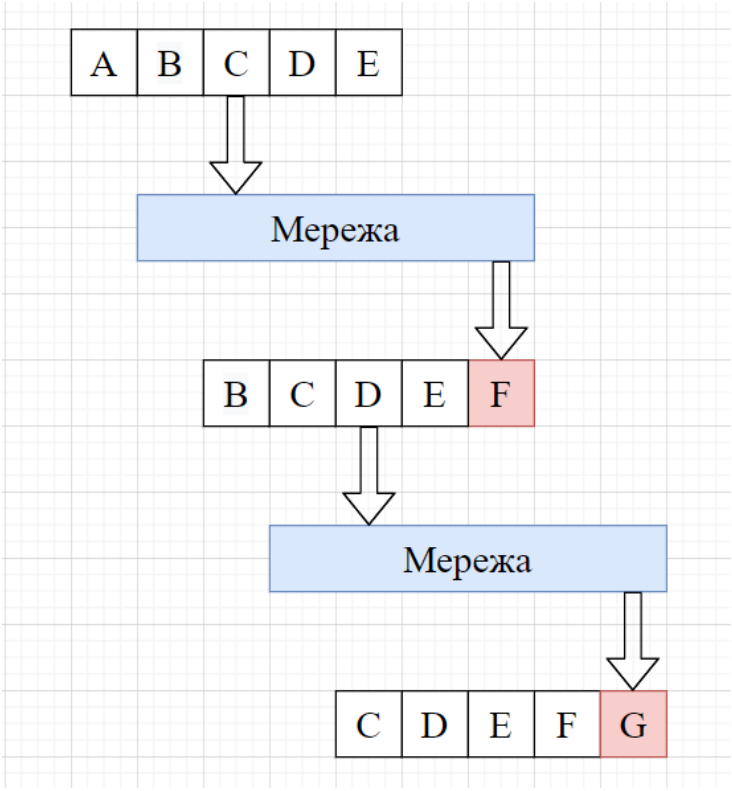

Рисунок 3.5 – Прицнип обробки вхідних нот

Щоб визначити найбільш вірогідне передбачення на виході з мережі, необхідно взяти індекс найвищого значення. Значення в індексі X у вихідному масиві відповідає ймовірності того, що X є наступною нотою. Схема передбачення зображено на рисунку 3.6.

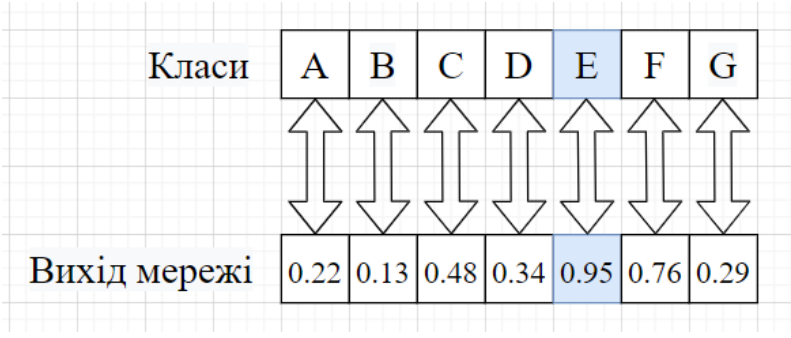

Рисунок 3.6 – Схема предбачення виходу мережі

Після цього всі виходи з мережі збираються до єдиного масиву. Після отримання всіх закодованих нот і акордів у масиві, можна розпочати їх декодування та створення результуючого масиву нотівта акордів.

Спочатку необхідно визначити, чи є вихід, який декодується, нотою чи акордом. Якщо об'єктом є акорд, то його доведеться розділити на масив нот. Після цього для кожного об'єкту акорду декодується відповідна нота. Масив отриманих нот формують реультуючий акорд. Якщо об'єктом є нота, то він декодується в ноту, використовуючи рядкове представлення висоти, що міститься в шаблоні.

У кінці кожної ітерації зміщення збільшується на 0,5, а створена нота або акорд додається до списку.

Код конвертації згенерованих нот до файлу наведено на рисунку 3.7.

```
# Конвертація згенерованих нот до файлу MIDI
def create_midi_file(output):
    notes_offset = 0
    result_notes = []

    for model_pattern in output:
        if ('.' in model_pattern) or model_pattern.isdigit():
            chord_notes = model_pattern.split('.')
            pattern_notes = []
            for cur_note in chord_notes:
                converted_note = note.Note(int(cur_note))
                converted_note.storedInstrument = instrument.Piano()
                pattern_notes.append(converted_note)
            converted_chord = chord.Chord(pattern_notes)
            converted_chord.offset = notes_offset
            result_notes.append(converted_chord)
        else:
            converted_note = note.Note(model_pattern)
            converted_note.offset = notes_offset
            converted_note.storedInstrument = instrument.Piano()
            result_notes.append(converted_note)

        notes_offset += 0.5

    output_stream = stream.Stream(result_notes)
    output_stream.write('midi', fp='generated_piece.mid')
```

Рисунок 3.7 – Код конвертації нот до файлу MIDI

Після отримання усього списку список нот і акордів, створених мережею, можна створити об’єкт Music21 Stream, використовуючи список як параметр. Щоб зберегти згенеровану музичну композицію необхідно створити файл MIDI, використовуючи функцію запису в наборі інструментів Music21 для запису потоку у файл.

У результаті на виході зберігається згенерований моделлю MIDI-файл, що містить задану кількість нот (у даному випадку 256) та складається з однієї фортепіанної партії.

Результати тренування моделі та набір нот для генерування композицій зберігаються на локальному диску та можуть бути використані повторно при наступному запуску програми.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Оцінка якості згенерованих композицій

На рисунку 4.1 представлено ноти музики, згенерованої за допомогою отриманої мережі LSTM. Деякі ноти в композиції розташовані неправильно. Це результат того, що нейронна мережа не може створити ідеальні мелодії. З поточною реалізацією завжди буде кілька помилкових нот. Для досягнення кращих результатів знадобиться більш натренована нейронна мережа.

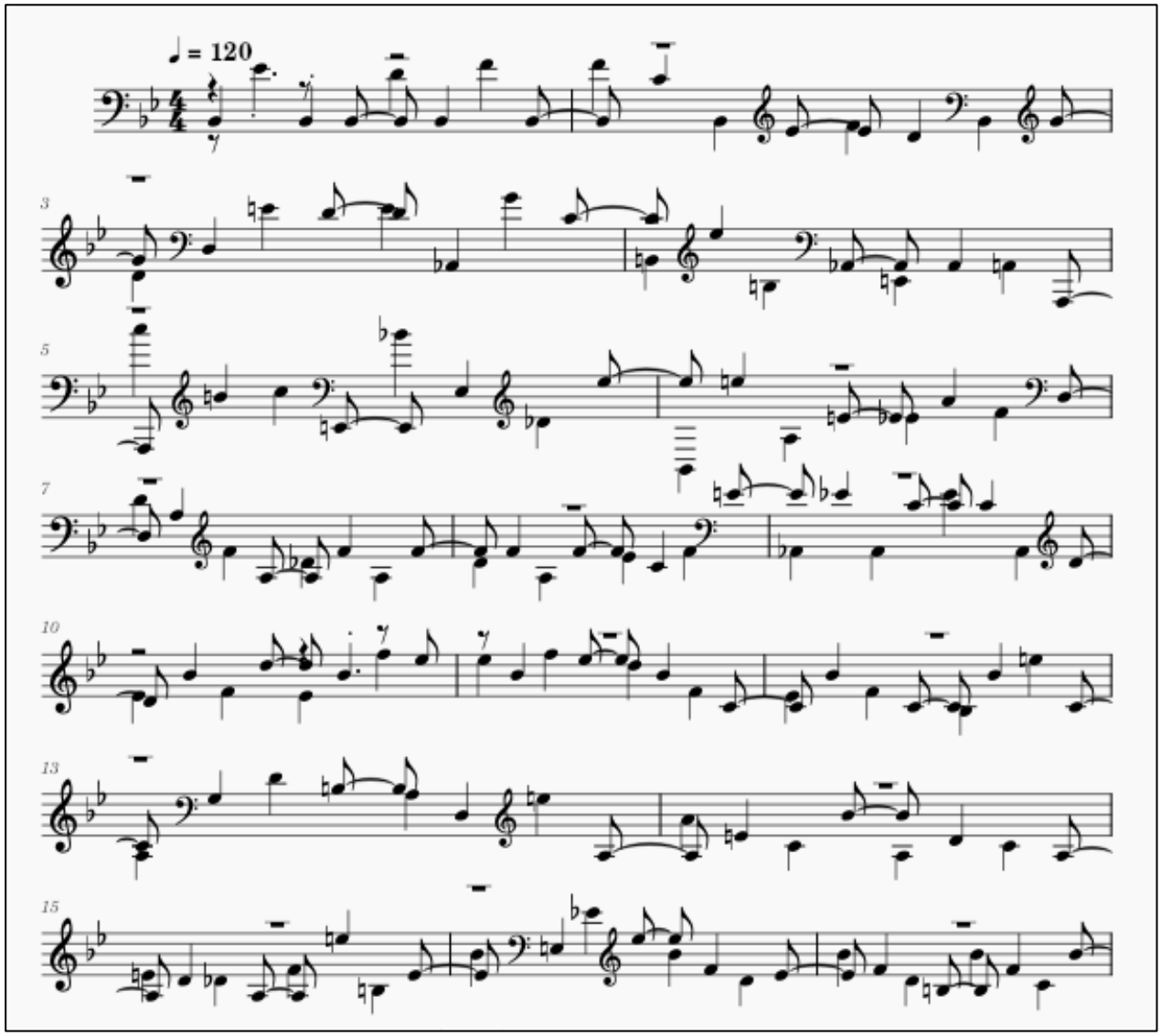

Рисунок 4.1 – Елемент композиції, згенерований мережею

Якісно згенеровані моделлю композиції мають гідний ритм і досягають успіху у створенні невеликих ділянок нот з мелодією, простою гармонією, а в деяких випадках навіть контрапунктом. Ці сегменти нот поліфонічні та має місце відчуття гармонії.

Недоліком моделі є недосконаліть музики у своїй здатності створювати чіткі переходи між великими ідеями у творі в цілому, більш глибокої структури не існує. Композиція також недостатньо використовує негативний простір, з невеликою кількістю пауз у роботі. Через відсутність глобальної структури музика має механічний сенс. Важливою особливістю є тривалість часу тренування. Коли модель тренується упродовж 30 хвилин, згенерована музика є набагато менш послідовною, ніж коли модель тренується 2 години. Простежується взаємозв'язок між створеною музикою та відповідними розробленими файлами для тренування.

Мінливість та складність музики, яку вивчала модель, досить сильно впливає на кінцевий результат. Вивчення нещодавно ініціалізованої моделі на великому наборі даних, що складається із значної мінливості музичних сегментів, прагнуло створити модель, яка спочатку здавалася заплутаною. Спроба опанувати такий набір функцій вимагає дуже складної функції, яка вимагає дуже тривалого періоду навчання.

З результатів дослідження випливає те, що на створення гармонійної музики потрібно дуже багато часу. Графіки тренувань не завжди дотримувались експоненціальної кривої. У багатьох випадках втрата тренувань врегульована на 1-2 години, а потім починає значно зменшуватися ще на декілька годин. Якість музики із збільшенням часу навчання відображає кількісний прогрес у навчанні.

Згенерована музика поступово вивчає ритм і структуру акордів. Це звучить так, ніби людина вчиться грати на фортепіано, але намагається грати пісні, які були занадто складними. Однією з можливих стратегій навчання може бути вивчення послідовності дедалі складніших пісень, тренування моделі вручну або автоматично після досягнення певної логарифмічної ймовірності.

Навчання може починатися з дуже коротких і збільшуватися до більш довгих інтервалів, щоб модель могла засвоїти базову структуру музичної форми.

4.2 Напрямки подальшої роботи

Нейромережева модель, реалізована на даний момент, не підтримує різну тривалість нот та різні паузи між нотами. Щоб досягти цього, необхідно додати більше класів для кожної тривалості та класи пауз, які представляють період часу між нотами.

Для досягнення задовільних результатів при додаванні більшої кількості класів також доведеться збільшити глибину мережі LSTM, для чого потрібен значно потужніший комп'ютер.

Далі необхідно додати вступ та закінчення до згенерованих партій, оскільки на даний момент мережа не враховує різниці між партіями, тобто не знає, де закінчується одна партія, а починається інша. Це дозволило б мережі генерувати фрагмент від початку до кінця, замість того, щоб генерувати фрагмент різко, як це відбувається у поточній мережі.

Ще олним покращенням може бути додавання методу обробки невідомих нот. Зараз мережа переходить у стан відмови, якщо зустрічає невідому ноту. Можливим методом вирішення цього питання могла б бути спроба знайти ноту або акорд, найбільш схожий на невідому ноту.

Також отриману систему можна покразити, додавши більше інструментів до набору даних. На даний момент мережа підтримує лише партії, які мають лише один інструмент.

Щодо подальшої роботи, було б корисно додати двовісний компонент LSTM, який фокусується лише на структурі. Така модель показує гарні результати, демонструючи переваги використання обмежених машин Больцмана для моделювання послідовностей акордів та інших форм

гармонічної та мелодійної структури. Крім того, ефективна модель може включати генетичні алгоритми [16]. Тренування моделі полягатиме у тому, щоб навчити її простій музиці та встановити міру придатності для новизни, тим самим дозволити алгоритму генерувати мутації, щоб ускладнювати фігуру з часом.

Іншим ефективним дизайном моделі є генеративні конкурентні мережі (Generative adversarial network, GAN) [17], які досягли значного прогресу у створенні фотореалістичних образів і як такі повинні забезпечити ефективне створення музики.

Ідея полягає у використанні навченого LSTM та налаштуванні гіперпараметрів, наприклад, за допомогою алгоритму навчання Q. Цей механізм працює, вивчаючи функцію вартості дії та дотримуючись найкращої політики. Потенційне вдосконалення може бути у композиції: досліджені шляхи або вжиті музичні заходи можуть мати різні переваги розподілу. Для ефективного вивчення та вивчення потенційних переваг може відбуватися баєсівське оновлення шляхом вибірки різних розподілів.

Дослідження або відбір проб може відбуватися через наївний жадібний механізм епсилону або верхню межу надійності, або найефективніший механізм вибірки Томпсона [18]. Використання парадигми додаткового навчання у поєднанні з технікою глибокого навчання ефективно змоделює основну музичну структуру та збільшить діапазон потенційних музичних виразів.

ВИСНОВКИ

У роботі була побудована модель нейронної мережі для генерування музичних композицій. Пройшовши певні етапи навчання, вона здатна будувати послідовності нот з акордами та невеликі музичні мелодії заданого характеру. Найбільш оптимальна мережа LSTM була створена таким чином, що наявність додаткової осі вздовж обчислювальних шарів гарантувала б гармонізацію побудованої мелодії.

Аналіз результатів показав відповідність побудованих композицій музичним канонам поєднання звуків, какофонічні або гармонійно неправильні фрази повністю відсутні. Додатковою перевагою є отримання досить чіткого звуку без спотворення частоти. До недоліків моделі можна віднести велику тривалість навчання запропонованої нейронної мережі і, відповідно, тривалу побудову музичної послідовності.

Розроблена модель може бути використана для побудови основних музичних композицій з метою їх подальшого використання при формуванні більш складних музичних творів, а також як демонстраційний матеріал, що ілюструє можливості нейронних мереж.

Крім того, отримана нейромережева модель може бути використана музикантами для аналізу музичних композицій певної категорії та створенні на їх основі нових композицій. Модель у даному випадку допоможе згенерувати унікальні партії для майбутнього твору.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Duncan A. Combinatorial music theory, 1991. – 448 с.

2. Donges N. Recurrent neural networks and LSTM – Towards data science, 2018. URL: https://towardsdatascience.com/ recurrent-neural-networks-and-lstm-4b601dd822a5 (дата звернення: 28.04.2021)

3. Moon T., Choi H., Lee H., Song I. RnnDrop: a novel dropout for RNNs in ASR, 2015. – 70 с.

4. Skuli, S. How to generate music using an LSTM neural network in Keras. 2017. URL: https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5 (дата звернення: 28.04.2021)

5. McCormack, J. Grammar-based music composition. 1996. – 16 с.

6. Zhang, G. Neural networks for classification: A survey, 2000. – 462 с.

7. Richer, M. Understand music theory: Teach yourself (3rd ed.) – London: Hachette UK, 2010. 224 с.

8. Arsenov A., Ruban I., Smelyakov K., Chupryna A. Evolution of Convolutional Neural Network Architecture in Image Classification Problems. – CEUR Workshop Processing, 2018.

9. Qi L., Wu Z., Zhu J., Meng, H. Modelling high-dimensional sequences with LSTM-RTRBM: application to polyphonic music generation –International Conference on Artificial Intelligence, 2015. С. 4138-4139.

10. Hochreiter S., Schmidhuber J. Long short-term memory – Neural computation 9, no. 8, 1997. С. 1735-1780.

11. Shopynskyi M., Golian N., Afanasieva I. Long short-term memory model appliance for generating music compositions. – PIC S&T'2020, 2020.

12. Agarwala, N., Inoue, Y., Sly, A. Music composition using recurrent neural networks, 2017. – 224 с.

13. Kline, D., Berardi M. Revisiting squared-error and cross-entropy functions for training neural network classifiers, 2005. С. 310-318

14. Briot, J., Hadjeres G., Pachet F. Deep Learning Techniques for Music Generation – A Survey, 2019. 312 c.

15. Johnson, D. Generating Polyphonic Music Using Tied Parallel Networks, 2017. C 128-143.

16. Иванов С. Нейросетевая Генерация Музыки, 2018. 57 с.

17. Chen, C. Creating melodies with evolving recurrent neural networks – Washington, DC: IEEE, 2001.

18. Bekuzarov M., Samantsov O., Mazurova O., Shirokopetleva M. Neural Network Architecture Editor With Code Generation. – Science and Technology (PIC S&T′2020), Kharkiv, Ukraine, 2020.