НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ Олег Чертов
«___» ___ 2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Наука про дані та математичне моделювання»
спеціальності 113 «Прикладна математика»
на тему: «Система створення музичних творів»

Виконала:
студентка IV курсу, групи КМ-92
Майструк Ніколь Олегівна _____

Керівник:
Старший викладач, канд. техн. наук
Ліскін Вячеслав Олегович _____

_____

Консультант з нормоконтролю:
Старший викладач,
Мальчиков Володимир Вікторович _____

Рецензент:
Доцент, канд. техн. наук, доцент кафедри СПСКС
Тарасенко-Клятченко О.В.

_____

Засвідчую, що в цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студентка _____

Київ — 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 113 «Прикладна математика»

Освітньо-професійна програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олег Чертов

«___» _____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студентці

Майструк Ніколь Олегівні

1. Тема роботи: «Система створення музичних творів», керівник роботи старший викладач, канд. техн. наук Ліскін В.О. затверджені наказом по університету від «31» травня 2023 р. № 2108-С.

2. Термін подання студентом роботи: «12» червня 2023 р.

3. Вихідні дані до роботи: розроблювана система має створювати музичні відрізки на основі попередньо заданої мелодії.

4. Зміст роботи: проаналізувати наявні системи автоматичного закінчення мелодій комп'ютерними засобами, дослідити математичні методи продовження мелодії, що можна реалізувати програмними засобами, спроектувати систему, що дописує мелодію та реалізувати її, провести випробування розробленої програми.

5. Перелік ілюстративного матеріалу: схеми можливих математичних рішень задачі, знімки екранних форм, алгоритм спроектованої системи

6. Консультанти розділів роботи:

7. Дата видачі завдання: «06» лютого 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою та збір даних	15.10.2022	
3	Проведення порівняльного аналізу математичних методів	24.12.2022	
4	Підготовка матеріалів першого розділу роботи	01.02.2023	
6	Підготовка матеріалів другого розділу роботи	15.03.2023	
7	Підготовка матеріалів третього розділу роботи	14.04.2023	
8	Розроблення програмного забезпечення	22.04.2023	
9	Підготовка матеріалів четвертого розділу роботи	20.05.2016	
10	Оформлення пояснювальної записки	08.06.2016	

Студентка _____ Ніколь МАЙСТРУК

Керівник роботи _____ Вячеслав ЛІСКІН

АНОТАЦІЯ

Дипломну роботу виконано на 51 аркуші. Вона містить два додатки та перелік посилань на використані джерела з 16 найменувань. В роботі містяться 27 рисунків та 2 таблиці.

Метою виконання даної дипломної роботи є створення математичного та програмного забезпечення системи завершення музичних творів.

В роботі проведено огляд існуючих програм та систем для автоматичного закінчення мелодій, досліджено методи закінчення мелодій шляхом застосування математичних алгоритмів. Зокрема було розглянуто методи з використанням ланцюгів Маркова, генеративних змагальних нейронних мереж, а також рекурентних нейронних мереж. На основі порівняння для реалізації обрано метод з застосуванням рекурентних нейронних мереж з довгою короткочасною пам’яттю.

Спроектовано та реалізовано систему закінчення мелодій на мові програмування Python. Проведено випробування розробленої системи.

Ключові слова: закінчення мелодій, LSTM.

ABSTRACT

The thesis is presented in 51 pages. It contains 2 appendixes and bibliography of 24 references. 16 figures and 2 tables are given in the thesis.

The purpose of this thesis is to create mathematical and software for a system for completing musical compositions.

The work reviews existing programs and systems for automatic melody completion, investigates methods of melody completion by applying mathematical algorithms. In particular, methods using Markov chains, generative adversarial neural networks, and recurrent neural networks were considered. Based on the comparison, the method using recurrent neural networks with long short-term memory was chosen for implementation.

A system for completing melodies in the Python programming language was designed and implemented. The developed system is tested.

Keywords: melody completion, LSTM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Forget gate — вентиль забування LSTM.

GAN — Генеративні змагальні мережі.

Imput gate — вхідній вентиль LSTM.

LSTM — рекурентна нейронна мережа з довгою короткочасною пам'яттю.

MIDI — Musical Instrument Digital Interface

Output gate — вихідний вентиль LSTM.

ВСТУП

Музика була невід'ємною частиною людського життя від початку часів. Її використовували для вираження емоцій, передачі інформації, створення відчуття спільності та розповіді історій. Існує припущення, що музика відіграла важливу роль в еволюції мови та комунікації.

Вважається, що перші люди використовували музику, щоб виражати свої емоції, святкувати і спілкуватися один з одним. Музика також використовувалася як інструмент для об'єднання групи. Наприклад, припускають, що в епоху палеоліту люди використовували барабанний дріб, щоб створити відчуття спільності та самовиразитися. Французький письменник Віктор Гюго сказав: "Музика виражає те, що не можна сказати і про що не можна промовчати"

З плином часу музика змінювалась. Різні композитори створювали музику за певними правилами, характерними для своїх епох. Композитори пристосовували свої стилі до мінливих смаків слухачів і використовували нові технології для створення інноваційних звуків і композицій.

У музиці існує популярний термін "Прокляття дев'ятої симфонії". Його суть полягає у тому, що композитори, які пишуть дев'яту симфонію, незабаром після цього помруть. Повір'я пов'язане з творчістю кількох видатних композиторів, зокрема Людвіга ван Бетховена, Антона Брукнера та Густава Малера. Ці композитори почали роботу на десятою симфонією, але так і не закінчили її.

«Ars longa, vita brevis» [12] латинська цитата, що походить з книги "Фауст" Йоганна Вольфганга фон Гете. Це лаконічний вислів, який спонукає розглянути поняття «мистецької спадщини»: Розробляючи систему, яка може завершувати твори, ми можемо потенційно втілити в життя мистецькі бачення композиторів, які залишилися незавершеними, зберігаючи та вшановуючи нематеріальну їх спадщину.

Широкого застосування система може набути в освітніх і наукових процесах. Вивчення незавершених творів, з новими створеними їх елементами, може надати

освітні можливості для студентів, науковців і дослідників музики. Вони дають уявлення про композиторські техніки та стилістичні нюанси різних епох, що дозволяє глибше зрозуміти музичну історію та еволюцію музичної композиції.

Важливою для сьогодення є проблема сприйняття людиною систем штучного інтелекту. Розробка даної системи може сприяти співпраці між людьми та системами штучного інтелекту. Композитори можуть працювати в тандемі зі штучним інтелектом, спираючись на наявний матеріал, і спільно створювати завершені композиції. Така співпраця може призвести до інноваційних і захопливих музичних результатів, які подолають розрив між людською творчістю та обчислювальною потужністю.

Проте потрібно визнати виклики, пов'язані із завершенням незавершених творів. Відсутність оригінального внеску композитора може поставити під сумнів автентичність і художню цілісність твору. Досягнення балансу між збереженням стилю композитора та впровадженням нових ідей може бути складним завданням. Крім того, дуже важливо підходити до цього завдання з повагою до первісних намірів митця і ретельно враховувати історичний та культурний контекст.

Нейронна мережа призначена для завершення незавершених музичних композицій шляхом створення нових сегментів, сумісних з існуючими. Потім згенеровані ноти будуть об'єднані з існуючими, щоб завершити незавершену музичну композицію. Вона аналізуватиме існуючу музичну композицію для визначення основної музичної структури, зможе розпізнавати шаблони на відрізку існуючої композиції , та створювати відрізки які гармонійно та мелодійно сумісні з існуючими.

1 ПОСТАНОВКА ЗАДАЧІ

Метою написання роботи є створення математичного та програмного забезпечення системи завершення музичних творів.

Для досягнення мети необхідно виконати наступні завдання:

а) проаналізувати наявні системи автоматичного закінчення мелодій комп’ютерними засобами;

б) дослідити математичні методи продовження мелодії, що можна реалізувати програмними засобами;

в) спроектувати систему, що дописує мелодію та реалізувати її;

г) провести випробування розробленої програми автоматичного продовження мелодії та зробити висновки про отримані результати.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ СТВОРЕННЯ МЕЛОДІЙ

2.1 Математичні методи для генерації музичних сегментів

2.1.1 Ланцюги Маркова

Ланцюги Маркова - це статистична модель, яка аналізує послідовності подій і прогнозує ймовірність переходу з одного стану в інший. У контексті музики ланцюги Маркова можуть фіксувати закономірності та взаємозв'язки між музичними елементами і генерувати новий музичний матеріал, який відповідає цим закономірностям.

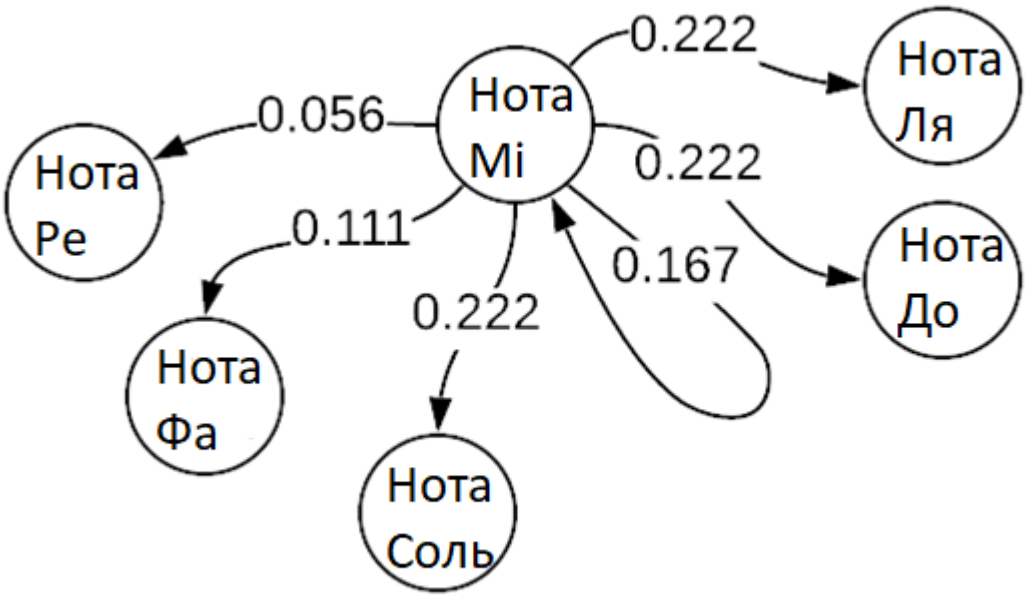


Рисунок 2.1 – Візуалізація ланцюга Маркова

Для використання ланцюгів Маркова система зазвичай аналізує набір даних існуючих музичних композицій. Цей набір даних може включати мелодії, акорди, ритми та інші музичні елементи. Після чого будується матриця переходів. Матриця переходів є фундаментальним компонентом ланцюга Маркова. Вона представляє

ймовірності переходу від одного музичного стану (наприклад ноти) до іншого. Кожен елемент матриці відповідає ймовірності переходу між двома станами. Значення в матриці визначаються на основі частоти або ймовірності спостереження цих переходів в аналізованому наборі даних.

Note	A	C#	E
A	0.1	0.6	0.3
C#	0.25	0.05	0.7
E	0.7	0.3	0

Рисунок 2.2 – Матриця переходів

Після побудови матриці переходів система може використовувати її для генерації нового музичного матеріалу. Починаючи з початкового стану або заданого музичного фрагмента, система використовує матрицю переходів для визначення ймовірностей переходу до різних музичних станів. Випадково вибираючи наступний стан на основі ймовірностей у матриці, система генерує послідовність музичних подій, формуючи цілісну музичну композицію.

Система може надавати регульовані параметри для впливу на вихід ланцюга Маркова. Наприклад, порядок ланцюга Маркова визначає, скільки попередніх станів враховується при визначенні наступного стану. Ланцюги вищого порядку можуть відображати більш довгострокові залежності, але можуть вимагати більших наборів даних для точного моделювання.

Хоча ланцюги Маркова забезпечують структурований підхід до створення нового музичного матеріалу, вони не враховують нюанси людської творчості та художнього судження. Тому дуже важливо, щоб система, яка завершує незавершені музичні твори, дозволяла втручатися людині

2.1.2 Генеративні змагальні мережі (GAN)

GAN – це тип моделі глибокого навчання, що складається з двох компонентів: генератора та дискримінатора. GAN успішно застосовуються в різних сферах, включаючи генерацію зображень та обробку природної мови. Компонент генератора GAN відповідає за створення нового музичного матеріалу. Він приймає випадковий шум або початковий музичний фрагмент на вході і генерує послідовність музичних подій на виході. Генератор навчається на навчальних даних і намагається створювати музичні композиції, які нагадують патерни та характеристики, що спостерігаються в наборі даних.

Дискримінатор виконує роль оцінювача в системі GAN. Він аналізує музичний матеріал, отриманий від генератора, і порівнює його з навчальними даними. Завдання дискримінатора - відрізнити згенеровану музику від реальної музики в наборі даних. Він забезпечує зворотний зв'язок з генератором, допомагаючи йому покращити свою здатність створювати більш автентичні та переконливі музичні композиції.

Компоненти генератора та дискримінатора навчаються разом у змагальному режимі. Генератор має на меті перемогти дискримінатор, створюючи музику, яку неможливо відрізнити від справжніх композицій. Одночасно дискримінатор прагне стати більш точним у розрізненні реальної та згенерованої музики. Цей змагальний процес навчання призводить до вдосконалення та покращення здатності генератора створювати реалістичні музичні композиції. Принцип роботи GAN зображено на рисунку 2.3.

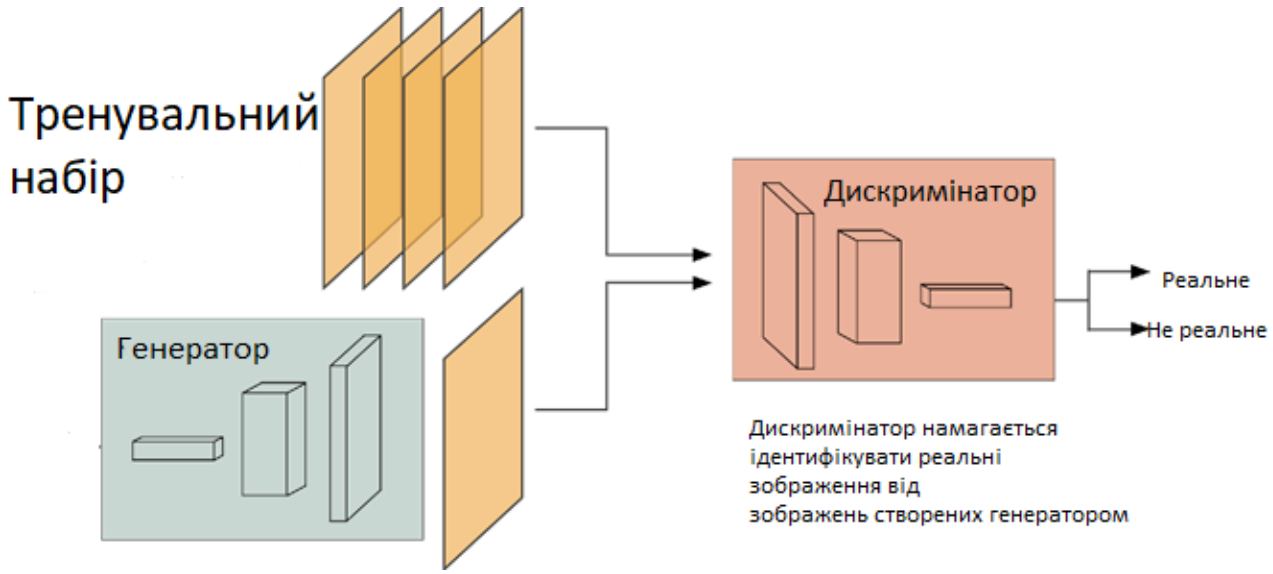


Рисунок 2.3 – Схема роботи GAN

Але такі системи більше підходять для генерації музичних творів, аніж для їх завершення.

2.1.3 LSTM

LSTM – це тип рекурентної нейронної мережі (RNN), яка може фіксувати послідовні залежності та довготривалі паттерни в даних. Щоб навчити LSTM-мережу завершувати музичні твори, потрібен набір даних існуючих музичних композицій. Цей набір даних слугує навчальними даними, забезпечуючи LSTM-мережу основою для навчання і створення нового музичного матеріалу. LSTM-мережі добре підходять для задач моделювання послідовностей, що робить їх придатними для фіксації часових залежностей у музиці. Після навчання мережу LSTM можна використовувати для створення нового музичного матеріалу.

Починаючи з початкового музичного фрагмента або випадково обраної початкової послідовності, мережа прогнозує наступну подію в послідовності на

основі вивчених шаблонів. Прогнозована подія додається до послідовності, і процес повторюється ітеративно, щоб згенерувати повну музичну композицію.

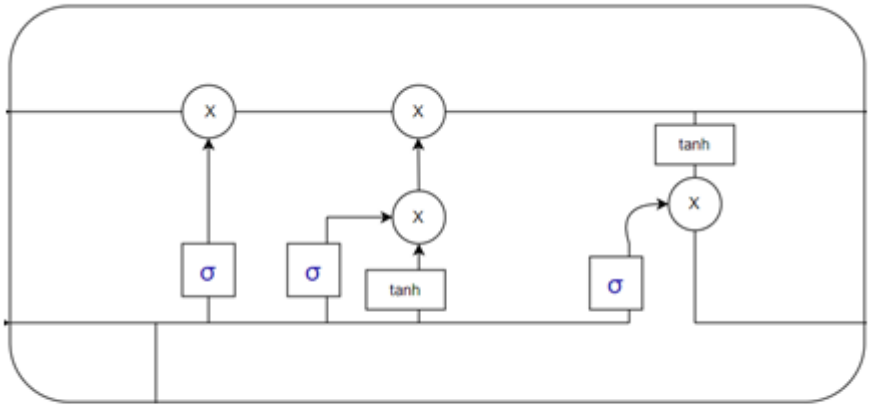


Рисунок 2.4 – Схема роботи LSTM [20]

LSTM-мережа може бути навчена за допомогою декількох ітерацій, з точним налаштуванням і коригуванням, зробленими на цьому шляху. Результати роботи мережі оцінюються, і її параметри можуть бути оновлені на основі бажаних музичних характеристик або поліпшень. Ітеративне навчання допомагає мережі вловлювати дрібніші деталі та вдосконалювати свою здатність генерувати завершені музичні твори.

2.1.4 Порівняння математичних методів розв’язання задачі

Результати аналізу математичних методів розв’язання задачі створення продовження музичних творів представлено в таблиці 2.1.

Таблиця 2.1 – Порівняння математичних методів

	Креативність системи	Фіксація довгострокових залежностей	Обчислювальні вимоги	Кількість даних необхідних для навчання
Ланцюги Маркова	Обмежена	Не має	Низькі	Невелика
GAN	Висока	Не має	Високі	Значна кількість різноманітних даних
LSTM	Висока	Має	Помірно-високі	Велика кількість

Отже з порівняння можна зробити висновки, що LSTM має можливість фіксувати довгострокові залежності, на відміну від двох інших методів, що добре підходить для розробки системи, що буде працювати з послідовностями. До недоліків системи можна віднести помірно-високі обчислювальні вимоги та велику кількість даних необхідних для навчання системи, проте ці вимоги менші ніж у системи розробленої з використанням GAN.

2.2 Огляд існуючих комерційних програмних рішень

2.2.1 Amper Music

Amper Music – стартап, створений Дрю Сільверштайном у 2014 році. Платформа Amper Music використовує передові алгоритми машинного навчання для аналізу та розуміння різних музичних стилів, жанрів і настроїв. Користувачі можуть взаємодіяти з платформою через зручний інтерфейс, де вони можуть вводити певні параметри, такі як темп, тривалість, інструментарій та настрій. На основі цих даних

система штучного інтелекту Amper Music генерує унікальну композицію в режимі реального часу, пристосовану до вподобань користувача.

Однією з визначних особливостей Amper Music є її здатність адаптувати та кастомізувати згенеровану музику. Користувачі можуть вносити корективи в композицію, змінюючи окремі елементи, такі як мелодія, гармонія, ритм та інструментарій. Такий рівень кастомізації дозволяє користувачам пристосовувати музику до конкретних вимог проекту, додаючи до згенерованих треків персональний підхід.

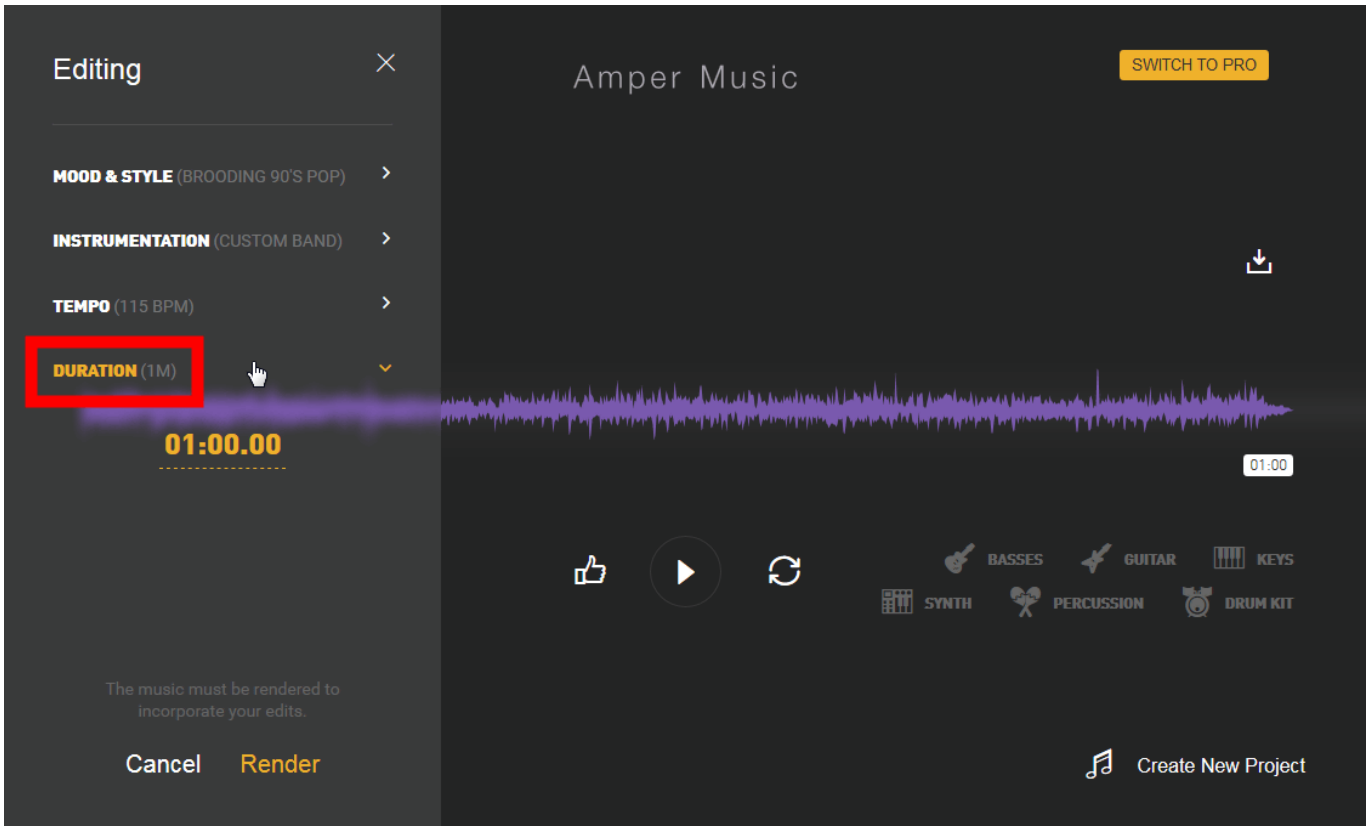


Рисунок 2.5 – Інтерфейс системи Amper Music [14]

На жаль, даний програмний продукт є більше генератором повноцінних музичних композицій, ніж системою, що буде завершувати музичні твори, хоча у програмі все ж таки можна задати певні параметри схожості з тим чи іншим твором певного композитора.

Використання програми не є безкоштовним, адже підписка з повним набором можливостей коштує 199$

2.2.2 AIVA (Artificial Intelligence Virtual Artist)

AIVA є системою, яку першою у світі визнали як віртуального композитора. AIVA використовує алгоритми глибокого навчання для створення оригінальної музики в різних жанрах і стилях. Вона була навчена на великому наборі даних класичних творів і сучасної музики, що дозволило їй створювати композиції, які імітують стиль композиторів-людей.

Користувачі можуть взаємодіяти з AIVA через зручний інтерфейс. Вони можуть надавати високорівневі інструкції, такі як бажаний настрій, темп, тривалість або інструментарій. На основі цих даних AIVA генерує оригінальні музичні композиції в режимі реального часу.

Унікальним аспектом AIVA є її здатність створювати складні оркестрові композиції. Вона може генерувати музику для різних інструментів та ансамблів, включаючи струнні, духові, дерев'яні, ударні тощо. Користувачі можуть вказати бажаний інструментарій, і AIVA створить композицію, яка відповідає їхнім уподобанням.

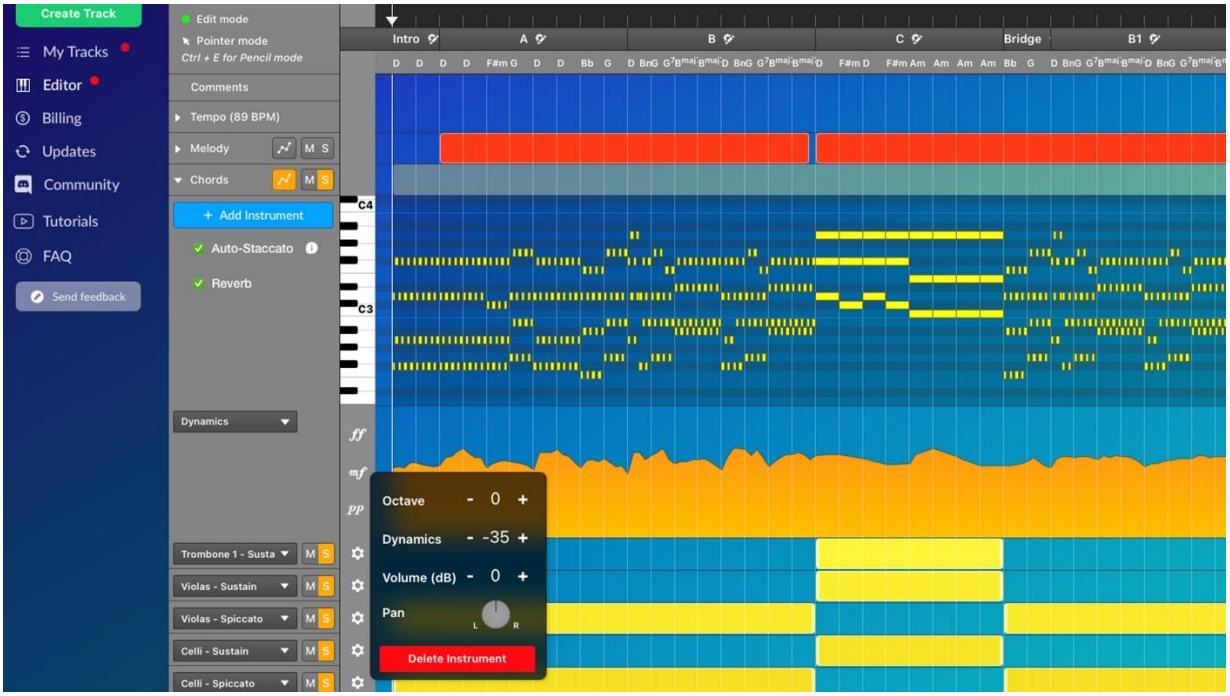


Рисунок 2.6 – Інтерфейс системи AIVA [15]

Тобто AIVA є чудовою систему для створення класичних мелодій, але все ж таки вона погано підходить саме до завершення творів.

AIVA також не є безкоштовним програмним забезпеченням. Підписка з всіма можливостями системи коштує 36$ на місяць.

2.2.3 Jukedeck

Jukedeck - система, яка спеціалізується на використанні штучного інтелекту для створення оригінальної музики. Створена у 2012 році Едом Ньютоном-Рексом, Jukedeck мала на меті революціонізувати музичну індустрію, надавши платформу для автоматизованого написання музики.

Основною технологією Jukedeck була система створення музики на основі штучного інтелекту. Використовуючи алгоритми глибокого навчання, система Jukedeck могла аналізувати закономірності в існуючих музичних даних і генерувати

оригінальні композиції в різних жанрах і стилях. Система мала можливість створювати мелодії, гармонії і навіть генерувати супровідні інструментальні треки, що робило її універсальним інструментом для творців контенту, відеопродюсерів та інших користувачів, які потребували музики для своїх проектів.

Однією з ключових особливостей Jukedeck був зручний інтерфейс, який дозволяв користувачам вводити певні параметри, такі як темп, настрій і тривалість, щоб налаштувати згенеровану музику відповідно до своїх потреб. Це дозволило користувачам легко створювати індивідуальні саундтреки для своїх відео, реклами чи інших творчих проектів, не вимагаючи жодних попередніх музичних знань чи навичок.

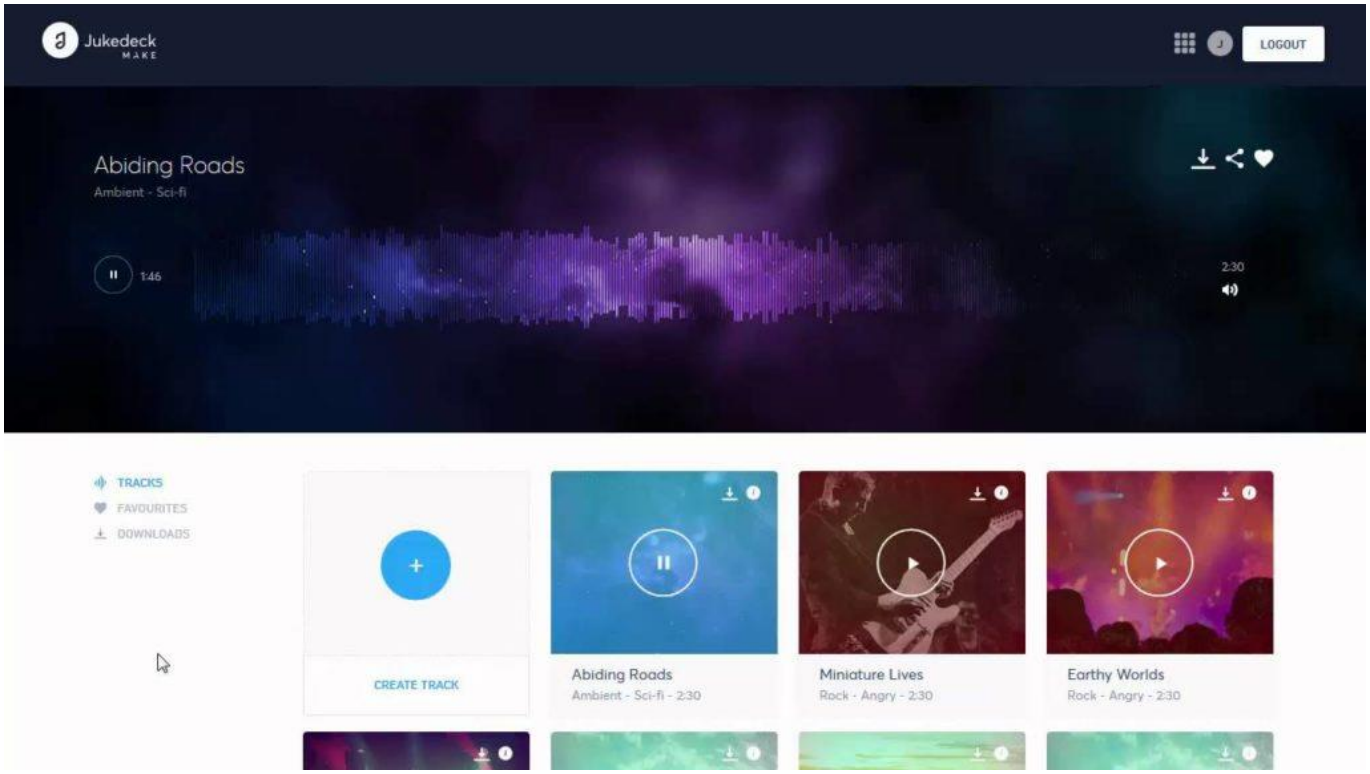


Рисунок 2.7 – Інтерфейс системи Jukedeck [16]

Проте саме завершити музичну композицію використовуючи дану програму не вийде. Можна лише згенерувати певні відривки, якщо задати параметри, які будуть задовольняти оригінальному твору. Використання системи також не є безкоштовним. Підписка на сервіс коштує 0.99$ на місяць.

2.2.4 Людина

Хоча некоректно розглядати людину як програмне рішення, людину можна розглядати як систему, що має здатність завершувати незавершені музичні твори. Як і будь-яка система, людина має різні компоненти і процеси, які сприяють її творчій діяльності.

Люди накопичують знання і досвід протягом багатьох років навчання і занять музикою. Це включає в себе розуміння музичної теорії, техніки композиції, історичного контексту та знайомство з різними музичними жанрами. Ці знання формують фундамент, на якому вони можуть будувати і розвивати незавершені твори, приймаючи обґрунтовані рішення на основі свого розуміння музичних принципів.

Творчий аспект людини як системи має вирішальне значення для завершення незавершених музичних творів. Завдяки своїй унікальній уяві та художній чутливості, які не є властивими «машинам» люди можуть досліджувати ідеї, мелодії, гармонії та структури. Вони можуть спиратися на свої творчі здібності, щоб заповнити прогалини і розвинути незавершений твір так, щоб він відповідав їхньому художньому баченню.

Музиканти та композитори володіють технічними навичками, розвиненими через практику та навчання. Ці навички включають вміння грати на музичних інструментах, розуміти нотну грамоту та користуватися інструментами для створення музики. Технічний досвід дозволяє людям ефективно передавати свої музичні ідеї, висловлювати свої наміри та втілювати незавершений твір у матеріальну форму.

Система людини охоплює її здатність інтерпретувати та виражати музичні концепції. Завершуючи незавершені твори, люди можуть проаналізувати наявний матеріал і заглибитися в стиль, наміри та емоційне вираження композитора. Вони можуть додати власну інтерпретацію та музичний голос до композиції, забезпечуючи унікальну перспективу, яка додає глибини та багатства кінцевому результату.

Таким чином людина може завершувати музичні твори, отримуючи більш живі композиції, але на закінчення одного втору підуть роки, аби людина змогла опанувати всі теоретичні та технічні навички

2.2.5 Порівняння програмних рішень для розв’язання задачі

Результати аналізу розглянутих комерційних програмних рішень наведено в таблиці 2.2.

Таблиця 2.2 – Порівняння комерційних програмних рішень

	Метод	Підходить для завершення мелодій	Вартість
Amper Music	Нейронна мережа	Частково	$199
AIVA	Нейронна мережа	Посередньо	$36
Jukedeck	Нейронна мережа	Частково	$0.99
Людина	Ручне написання музики	Так	Може змінюватись

Отже, з порівняння можна зробити висновки, що не існує доступного рішення спрямованого саме на завершення музичних творів. Доступні системи створення музики спрямовані скоріше на генерацію мелодій, адже вони є комерційними продуктами. Людина могла б стати рішенням для проблеми незакінчених мелодій, але на таке продовження можуть піти роки роботи.

2.3 Висновки до розділу

На основі порівняльного аналізу можливих алгоритмів для реалізації системи (таблиця 2.1) можна зробити висновок, що підхід із застосуванням рекурентної нейронної мережа з довгою короткочасною пам'яттю, або LSTM є найкращим для вирішення задачі закінчення музичних композицій, що може бути обумовлено здатністю системи до фіксації часових залежностей у послідовностях. Також плюсом системи є помірно-високі обчислювальні вимоги, на відміну від GAN. Ефективність методу буде залежати від якості даних для навчання, а також від правильного підбору параметрів для моделі.

На основі порівняльного аналізу (таблиця 2.2) серед існуючих комерційних програмних рішень не знайдено системи спеціалізованої для завершення незавершених музичних творів, адже у більшості випадків системи є генераторами.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Основні поняття LSTM

Як і інші нейронні мережі LSTM складаються з нейронів, які в даному випадку називають «комірками». Кожна комірка складається з 3 частин: Imput gate (вхідній вентиль), Forget gate (вентиль забування) Output gate (вихідний вентиль).

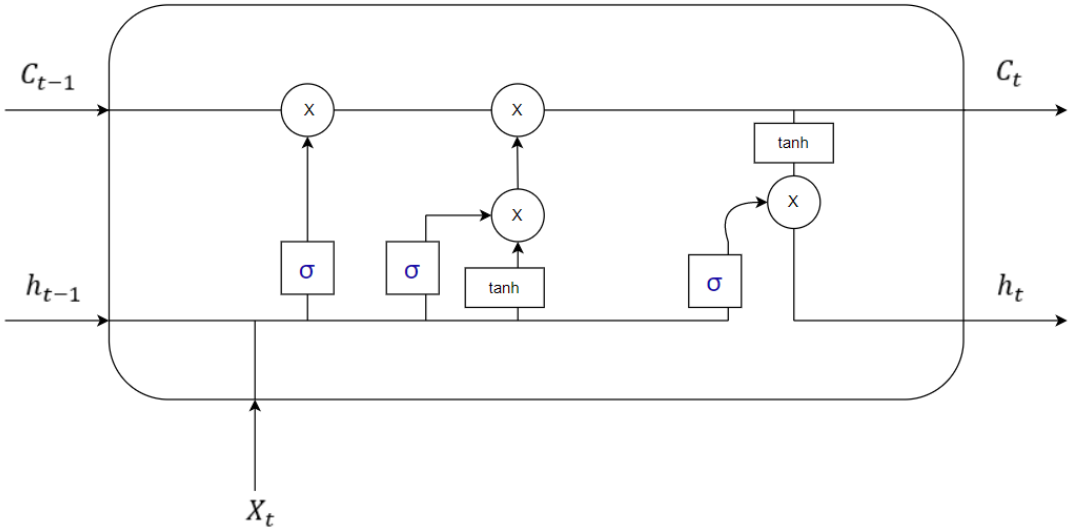


Рисунок 3.1 – Загальний вид комірки LSTM

- C_{t-1} – Стан попередньої комірки
- C_t – Стан поточної комірки
- h_{t-1} – Попередній прихований стан
- h_t – Прихований стан
- x_t – Вхід
- σ – Сигмоїдальна функція активації
- $tanh$ – Гіперболічний тангенс (функція активації)

3.2 Структура LSTM

3.2.1 Forget gate

Цей вентиль видаляє інформацію, що не потрібна перед тим, як об’єднати її зі станом комірки. На вхід приймаються: x_t, тобто «Вхід» і h_{t-1} – прихований стан попередньої комірки. Дані проходять крізь сигмоїдний вентиль та приймається рішення чи потрібно зберегти інформацію, чи забути її.

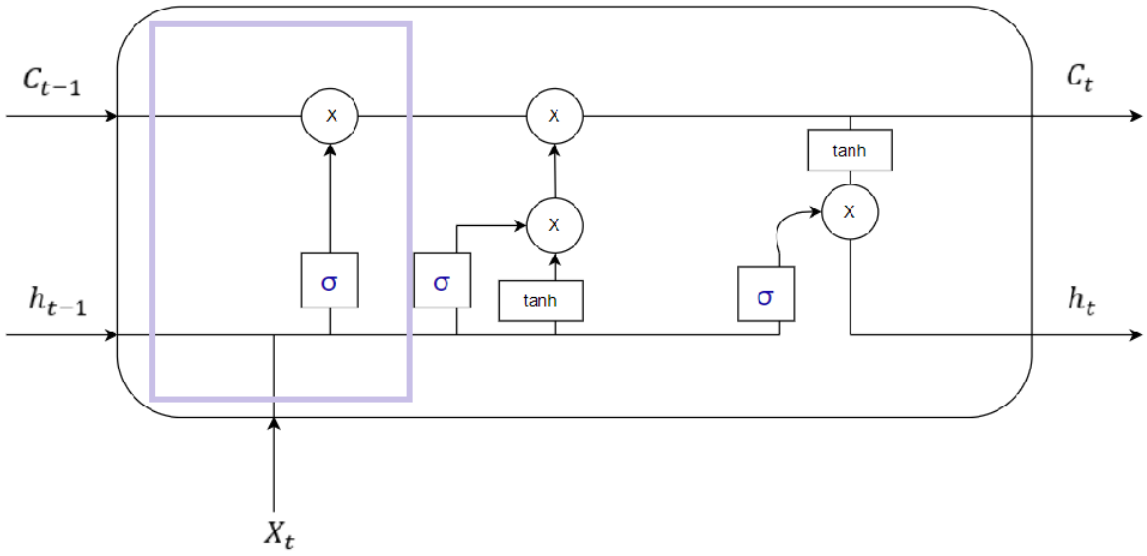


Рисунок 3.2 – Вентиль забуття LSTM

Рівняння:

$$f_t = \sigma(x_t * U_f + h_{t-1} * W_f) \tag{3.1}$$

U_f – ваги пов’язані зі входом

W_f – матриця вагів пов’язаних з попереднім виходом

Далі застосовується сигмоїдна функція, яка перетворює f_t на число від 0 до 1. Якщо значення близьке до 0 – інформація видаляється.

3.2.2 Imput gate

Вхідний вентиль визначає, скільки нової вхідної інформації має бути допущено до стану комірки.

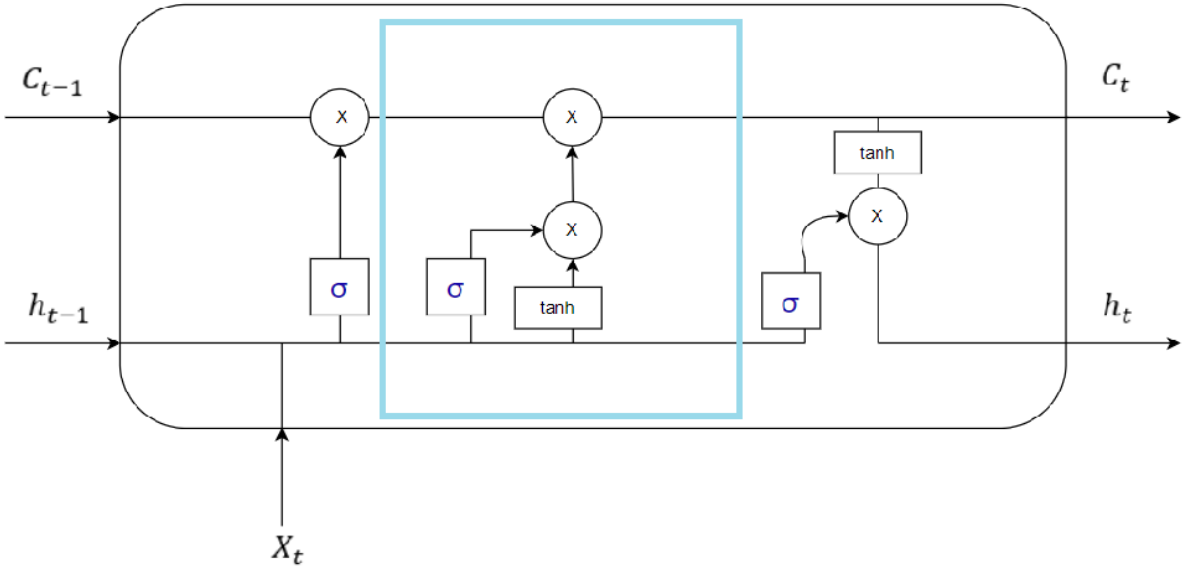


Рисунок 3.3 – Вхідний вентиль LSTM

Рівняння:

$$i_t = \sigma(x_t * U_i + h_{t-1} * W_i) \tag{3.2}$$

U_i – ваги пов’язані зі входом

W_i – матриця вагів пов’язаних з попереднім виходом

Далі знову застосовується сигмоїдна функція, яка перетворює i_t на число від 0 до 1.

Далі відбувається процес «оновлення кандидата». Тут LSTM вирішує, яку частини інформації зберегти, а яку відкинути.

Рівняння:

$$N_t = tanh(x_t * U_c + h_{t-1} * W_c) \quad (3.3)$$

Потім застосовується функція активації гіперболічний тангенс. Завдяки цій функції значення інформації буде знаходитись між 1 і -1. Якщо значення є від'ємним – інформація не додається, а якщо додатне - інформація додається до стану комірки.

Додавання до стану комірки відбувається за формулою:

$$C_t = f_t * C_{t-1} + i_t * N_t \quad (3.4)$$

3.2.3 Output gate

Вихідний вентиль дозволяє LSTM вибірково виводити інформацію зі стану комірки, відфільтровуючи нерелевантну або менш важливу інформацію.

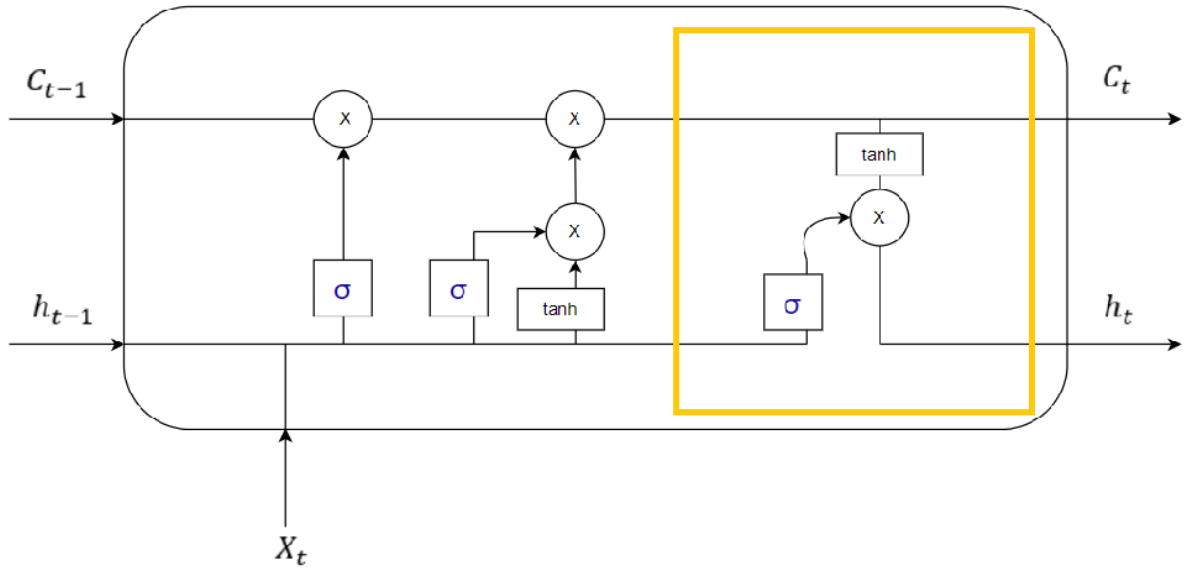


Рисунок 3.4 – Вихідний вентиль LSTM

Рівняння:

$$o_t = \sigma(x_t * U_o + h_{t-1} * W_o) \quad (3.5)$$

Застосовуючи сигмоїдну функцію отримуємо вихід між 0 і 1. Він визначає кількість інформації про стан комірки, яку слід вивести.

Тепер, щоб обчислити поточний прихований стан скористаємось рівнянням

$$H_t = o_t * tanh(C_t) \quad (3.6)$$

Отже ми можемо побачити, що прихований стан є функцією довгострокової пам'яті (C_t) і поточного виведення.

3.3 Застосування LSTM до продовження мелодій

«Оскільки LSTM ефективні для захоплення довгострокових часових залежностей, не страждаючи від перешкод оптимізації, які заважають простим рекурентним мережам (RNN), їх використовували для просування сучасного рівня техніки для розв'язання багатьох складних завдань. Це включає в себе розпізнавання і генерацію рукописного введення, мовне моделювання і переклад, акустичне моделювання мови, синтез мови, передбачення вторинної структури білка, аналіз аудіо- та відеоданих».[10]

Розглянемо принцип роботи алгоритму на прикладі, тобто ці процеси, які відбуваються в системі для створення нового відрізку музики. Для прикладу оберемо найпростішу музичну гаму – «До мажор».

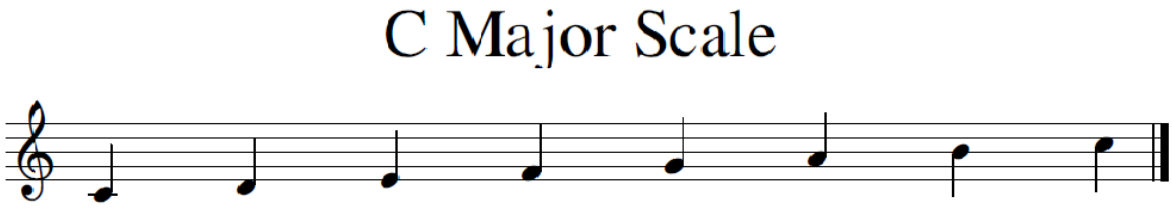


Рисунок 3.5 – Гама «До мажор»

У чисельному представленні гама виглядатиме так:

60 – 62 – 64 – 65 – 67– 69 – 71– 72. Якщо розглянути іншу гаму наприклад Соль мажор.

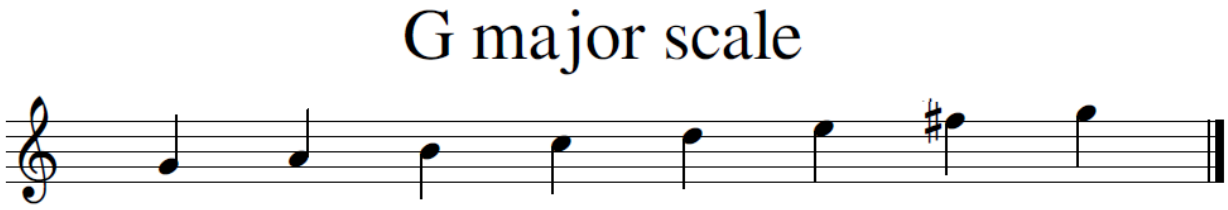


Рисунок 3.6 – Гама «Соль мажор»

У чисельному представлені вона виглядатиме: 67 – 69 – 71 – 72 – 74 – 76 – 78 – 79.

У чисельному представлені гам можна побачити закономірність їх побудови.

Крок між нотами чисельно можна описати: 2–2–1–2–2–2–1. Ця закономірність логічно пояснюється правилом побудови мажорної гами. Кожна мажорна гама будується по принципу: "тон – тон – полутон – тон – тон – тон – полутон". Ця схема означає, що між першою та другою ступенями розташовується один тон, між другою та третьою ступенями – один тон, і так далі за формулою. Ступінь - це порядковий номер ноти в гамі.

Цю закономірність побудови легко може вивчити і LSTM.

Розіб’ємо мелодію на послідовності. Наприклад нехай система прийматиме на вхід послідовність з 3 нот і доповнюватиме гаму 4 нотою.

На першій ітерації на вхід подаємо перші 3 ноти

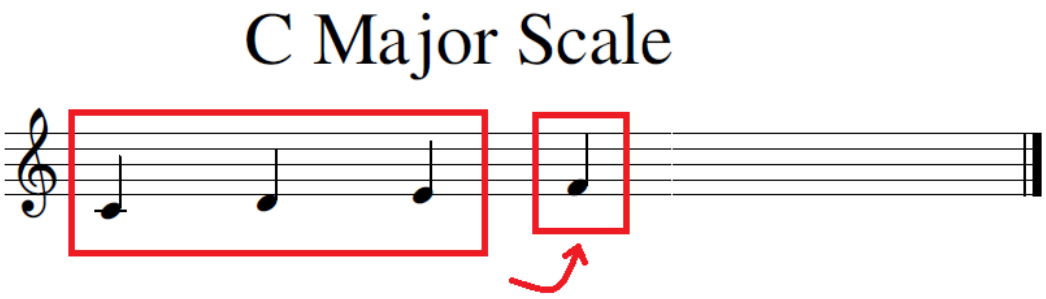


Рисунок 3.7 – Візуальна інтерпретація першої ітерації генерування

Далі на вхід подаємо вже три ноти, включаючи одну, згенеровану на минулому кроці

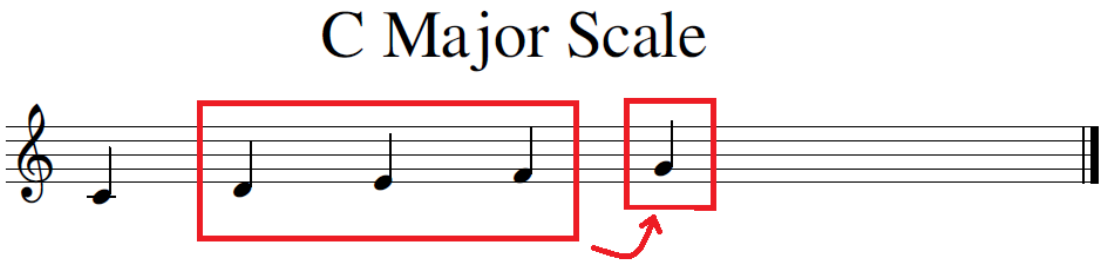


Рисунок 3.8 – Візуальна інтерпретація другої ітерації генерування

На останній ітерації отримаємо всю гаму

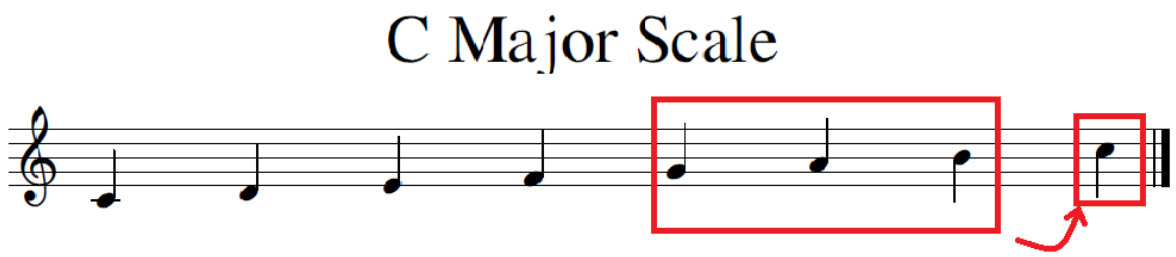


Рисунок 3.8 – Візуальна інтерпретація останньої ітерації генерування

В результаті, при достатній кількості даних для навчання LSTM легко зможе відтворити музику. Проте варто пам’ятати, що системи, що використовують LSTM-моделі спираються на закономірності, отримані з навчальних даних. Такій системі може бути складно зрозуміти ширший музичний контекст або емоційні аспекти мелодії.

3.4 Аналіз можливих проблем у роботі алгоритму

Оскільки закінчення мелодії відбувається без участі композитора можна поставити під сумнів автентичність і художню цілісність твору.

LSTM спирається на закономірності, отримані з навчальних даних, і як наслідок, згенерованим завершенням мелодій може бракувати музичності, яку міг би створити композитор. Також системам з LSTM може бути складно вловити складні музичні структури, такі як складні акордові прогресії, гармонійні модуляції або складні композиційні прийоми. Ці обмеження можуть призвести до неповного або непослідовного створення мелодії.

Також варто зазначити, що робота системи може потребувати додаткового коригування результату людиною.

3.5 Висновки до розділу

У розділі розглянуто математичне забезпечення системи завершення незавершених музичних творів. Система використовує нейромережеву архітектуру LSTM для моделювання музичної структури творів і генерації продовжень. LSTM є типом рекурентної нейромережі, яка може зберігати інформацію про попередні стани та використовувати її для прогнозування майбутніх станів. Вона добре підходить для моделювання послідовних даних, таких як музичні композиції.

Розглянуто структуру комірки LSTM та приклад роботи нейромережі з музичними послідовностями, а також виокремлено можливі недоліки в роботі математичного методу.

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Структура програми

Умовно програму можна поділити на два процеси, кожний з яких використовує декілька модулями.

Процес створення моделі:

– модуль зчитування творів з файлу;
– модуль декодування/кодування файлів;
– модуль ініціалізації моделі;
– модуль навчання моделі;
– модуль збереження моделі;

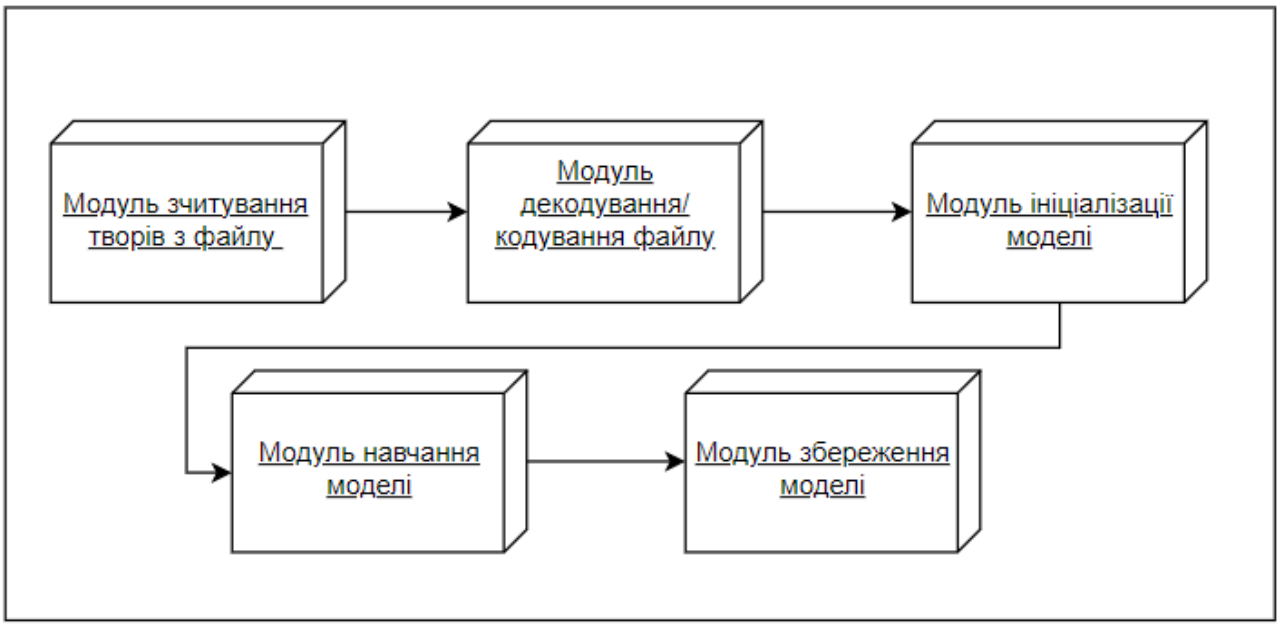


Рисунок 4.1 – Схематичне представлення процесу створення моделі

Процес генерації нот:

– модуль зчитування творів з файлу;

– модуль декодування/кодування файлів;
– модуль завантаження навченої моделі;
– модуль генерації мелодії;
– модуль відтворення мелодії;
– модуль збереження результуючого файлу;

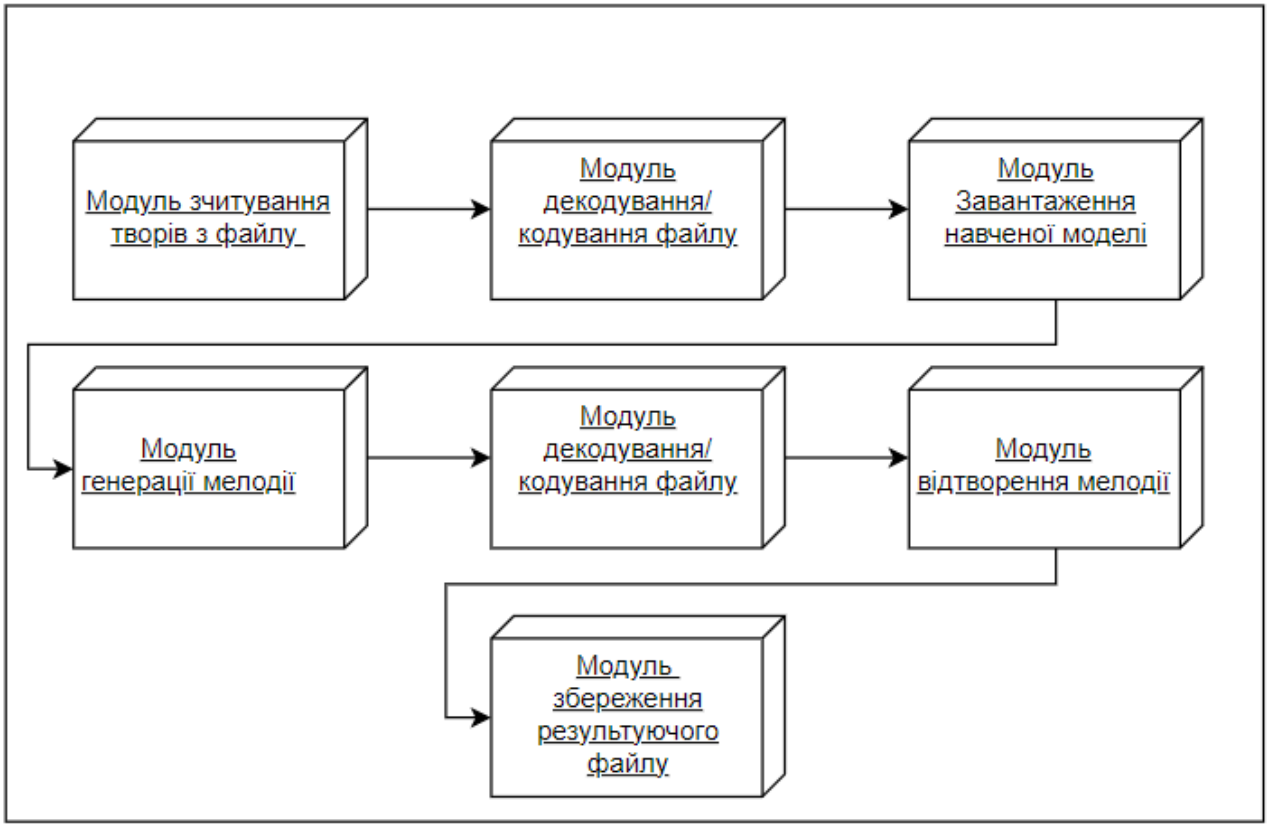


Рисунок 4.2 – Схематичне представлення процесу генерації мелодії

Тобто в програмі є два основних процеси, які мають повторювані модулі. Модулі, які зустрічаються у двох процесах одночасно є модулями обробки і підготовки вхідних даних.

4.2 Бібліотеки та засоби розробки

Програма розроблена на мові програмування Python. Дана мова має перевагу над іншими мовами програмування, завдяки великій кількості бібліотек для зручної роботи з файлами та розробки нейронних мереж.

fluidsynth – це бібліотека, яка надає програмний синтезатор для відтворення та синтезу MIDI-файлів. Вона дозволяє конвертувати MIDI-файли в аудіосигнали, імітуючи звучання різних музичних інструментів.

numpy – бібліотека для числових обчислень в мові програмування Python. Вона надає функції та інструменти для роботи з масивами і матрицями.

pandas – бібліотека для створення високопродуктивних структур даних, які називаються DataFrame, та дозволяють зручно та ефективно працювати з табличними даними.

pretty_midi – це інструмент для роботи з MIDI-файлами в мові програмування Python. Вона надає можливість зчитувати, створювати та редагувати MIDI-дані, що включають ноти, акорди, інструменти, контролери та інші елементи, які використовуються для представлення музичної інформації.

tensorflow – бібліотека для чисельних обчислень та машинного навчання. Вона слугує для визначення архітектури моделей, вибору функцій втрат та оптимізаторів, а також для тренування та оцінки моделей.

Систему було розроблено у середовищі розробки Google Colaboratory. Google Colaboratory – це безкоштовне середовище розробки, що надає можливість запускати та виконувати код Python у хмарному середовищі. Воно надає доступ до віртуальної машини з попередньо встановленими пакетами та бібліотеками. Також надається доступ до віддаленої машини з вбудованими центральним та графічним процесорами, що дозволяє прискорити обчислення моделей машинного навчання.

4.3 Реалізовані функції

У програмі реалізовано низку функцій. Основні функції для роботи системи:

display_audio – функція приймає на вхід midi файл та створює його аудіо інтерпретацію, використовуючи бібліотеку fluidsynth.

decode_midi_to_notes – функція приймає на вхід midi файл та створює таблицю з розкодованими нотами.

encode_notes_to_midi – функція приймає на вхід таблицю з нотами та створює з них midi файл.

create_sequences – функція приймає на вхід таблицю з нотами та створює послідовності нот для моделі LSTM.

predict_note – функція приймає на вхід послідовність нот, навчену модель та повертає нову ноту.

predict – функція приймає на вхід кількість нот, які необхідно згенерувати та використовує функцію predict_note для генерації закінчення мелодії.

4.4 Опис розроблених алгоритмів

4.4.1 Навчання моделі

Для навчання моделі з датасету оберемо всі файли композитора Франца Шуберта. Маємо для навчання моделі 29 файлів. Після імпортування файлів кожен файл декодується за допомогою функції decode_midi_to_notes. З кожної ноти добуваються три характеристики:

- нота – цифрове значення, яке вказує безпосередньо на ноту;
- тривалість – тривалість кожної ноти;

- крок – кількість секунд, що проходить від закінчення звучання попередньої ноти до початку поточної.

Далі ноти розбиваються на послідовності на яких буде навчатись модель завдяки функції create_sequences.

Для навчання моделі визначаємо функцію втрат (loss function). Функція втрат використовується для оцінки розбіжності між прогнозованим виходом і справжнім цільовим виходом. Тобто слугує мірою того, наскільки добре працює LSTM-модель.

Для створення моделі вказуються входи (послідовності нот), виходи (три характеристики ноти), функцію втрат(Categorical Cross-Entropy) та оптимізатор (Adam).

Блок-схему алгоритму проведення навчання наведено на рисунку 4.1.

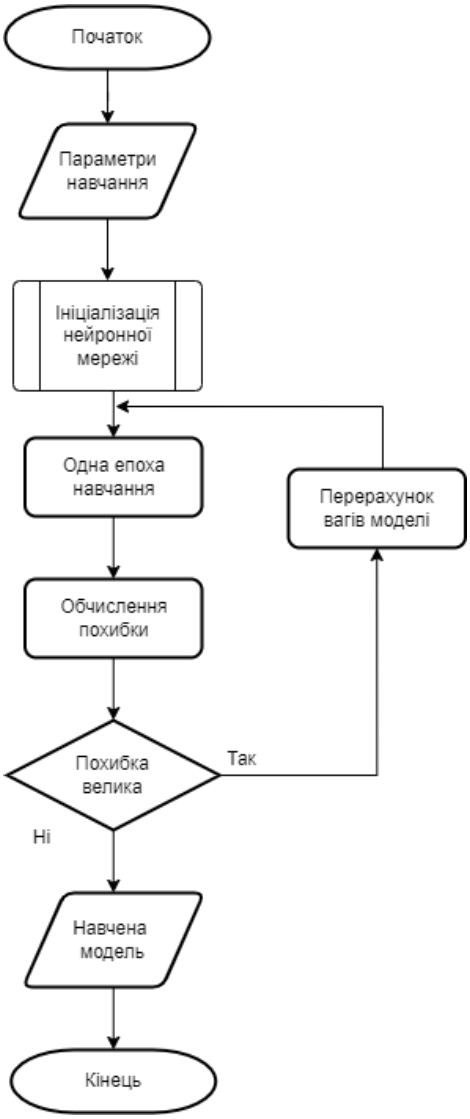


Рисунок 4.3 – Схема алгоритму навчання LSTM

Також визначається функція попередньої зупинки навчання моделі. Навчання завершиться, якщо буде досягнуто найкращого результату, і подальшого покращення не буде на протязі наступних епох.

4.4.2 Генерування продовження мелодії

Обирається мелодія для закінчення. Мелодія декодується так само, як декодувались мелодії для навчання, за допомогою функції decode_midi_to_notes. Ноти також діляться на послідовності за допомогою функцію create_sequences. Задається кількість нот, які потрібно згенерувати. Функція predict генерує закінчення мелодії. Коли ціль досягнуто, результат конвертується у midi файл функцією encode_notes_to_midi. Також створюється аудіо інтерпретування файлу функцією display_audio.

Рисунок 4.4 – Схема алгоритму генерування продовження мелодії

Користувач має змогу завантажити згенеровані файли завдяки методу files.download, який вбудовано в Google Colaboratory.

4.5 Формат вихідних даних

4.5.1 Вихідні дані для ініціалізації системи

Вимоги до вихідних файлів:

- Вхідні файли мають бути лише у форматі midi
- Вхідні мелодії мають бути написані для фортепіано

Варто зауважити, що система навчена на творах певного композитора, а саме Франца Шуберта, тому отримуючи продовження мелодії слід розуміти, що продовження матиме певні особливості притаманні саме творчості Шуберта.

4.5.2 Вихідні дані для навчання системи

Для навчання і роботи системи використовуються файли у форматі midi. MIDI (Musical Instrument Digital Interface, у перекладі озачає цифровий інтерфейс музичних інструментів) — стандарт передачі інформації між електронними музичними інструментами, розроблений 1983 року, що дає можливість електронним музичним інструментам взаємодіяти між собою чи комп'ютером та іншим MIDI-сумісним обладнанням, здійснювати з одного інструменту управління іншими [22].

MIDI-файли представляють музику як послідовність подій. Ці події можуть включати повідомлення про ввімкнення/вимкнення нот, зміни висоти тону, зміни

темпу тощо. Кожна подія асоціюється з певною часовою міткою, яка вказує, коли вона має відбутися під час відтворення.

Октави	Ноти											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127	–	–	–	–

Рисунок 4.5 – Принцип кодування нот у midi файлах

Основна увага в MIDI-файлах приділяється представленню музичних нот. Кожна нота представляється подією "note-on", яка вказує висоту (номер ноти) і гучність ноти. Відповідна подія "note-off" вказує, коли нота має бути відпущена.

Такі файли також містять інформацію про синхронізацію і темп для керування швидкістю і ритмом відтворення. Таймінг базується на поділі часу, який називається "tick", і може представляти музичні долі або частки долей. Темп задається в ударах на хвилину (BPM).

Для кодування різних музичних подій і керуючої інформації використовують MIDI-повідомлення.

MIDI-повідомлення - це короткі послідовності байтів, які зазвичай складаються з байта стану, за яким слідує один або декілька байтів даних. MIDI-повідомлення можна використовувати для вказівки подій модуляцій, зміни висоти тону, гучності тощо.

Для навчання системи використовувались твори отримані з датасету MAESTRO. MAESTRO (MIDI та Audio Edited for Synchronous Tracks and Organisation) — це набір даних, що складається з приблизно 200 годин віртуозних виступів на фортепіано, зафіксованих із точним вирівнюванням (~3 мс) між мітками нот і звуковими сигналами.[12]

Також пропонується роздільна конфігурація навчання/перевірки/тесту, щоб одна композиція, навіть якщо її виконують кілька учасників, не з’являлася в кількох підмножинах.

Набір даних має в собі твори 60 різних композиторів різних епох розвитку музики. Зокрема твори Йоганна Себастьяна Баха, епохи бароко, Вольфганга Амадея Моцарта, епохи класицизму, а також композиторів романтиків, та сучасних композиторів.

Оберемо для системи роботу з творами Франца Шуберта. Даний композитор відомий своїм значним внеском в розвиток фортепіанної музики, а також через ранню смерть у віці 31 року, залишив по собі багато незавершених творів.

Фортепіанні твори Шуберта охоплюють широкий спектр форм, включаючи сонати, експромти, музичні п'єси і танці. Його композиції часто мають розгалужену структуру, з кількома контрастними розділами, а також містять технічно складні пасажі, довгі мелодійні лінії, та акордових пасажі . Музика Шуберта характеризується емоційною глибиною та виразною силою. Його твори викликають широкий спектр емоцій, включаючи радість, смуток, тугу. Здатність Шуберта створювати пронизливі та проникливі мелодії є відмінною рисою його музичного стилю.

4.6 Формат результуючих даних

В результаті роботи програми користувач отримує дописаний фрагмент мелодії у вигляді аудіо-відтворення та файлу у форматі midi.

В результаті роботи програми дописується один з двох творів композитора Франца Шуберта. А саме, це фортепіанна соната в Мі мажорі, а також фортепіанна соната в До мажорі.

4.7 Результати випробування

4.7.1 Навчання LSTM

Під час навчання моделі було порівняно результати навчання у 20, 100 та 200 епох.

При навчанні моделі у 20 епох та подальшої генерації музичних сегментів – отримуємо негармонічну мелодію з часто повторюваними однаковими музичними фразами.

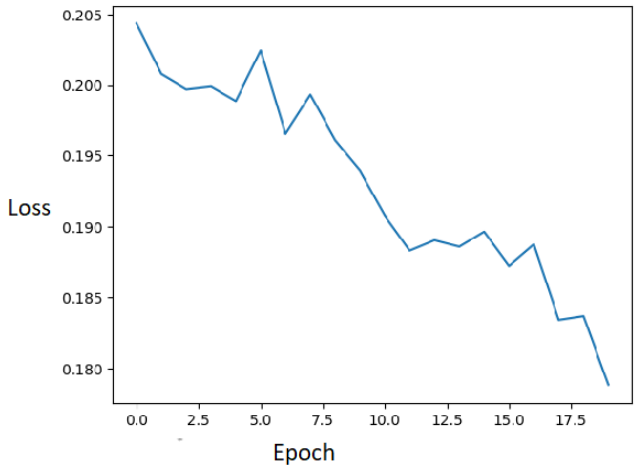


Рисунок 4.6 – Функція втрат при навчанні у 20 епох

При навчанні моделі у 200 епох спостерігається різке збільшення втрат, що свідчить про перенавчання моделі. Через це спрацьовує функція попередньої зупинки навчання.

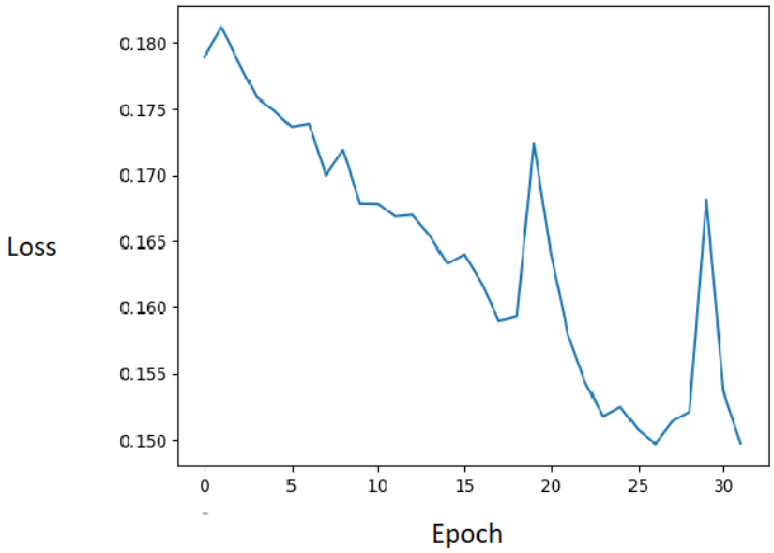


Рисунок 4.7 – Функція втрат при навчанні у 200 епох

При навчанні моделі у 100 епох. Модель показує найкращий результат у подальшій генерації мелодій. Згенерована мелодія часто має послідовну структуру, що складається з фраз, мотивів і музичних розділів.

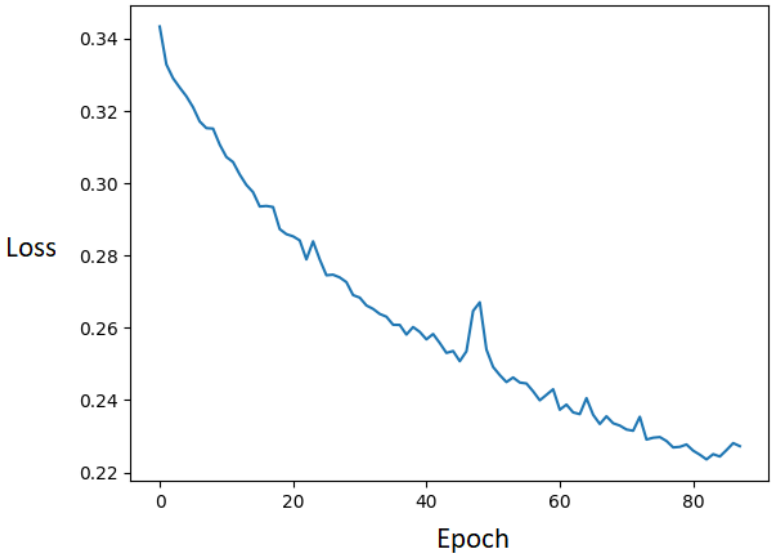


Рисунок 4.8 – Функція втрат при навчанні у 100 епох

Також під час навчання було порівняно результати навчання моделі використовуючи різну кількість нот у послідовностях. Дослідження проведено на послідовностях у 5, 15, 25 нот.

При навчанні на послідовності з 5 нот результуючий музичний сегмент не характеризується мелодичністю та послідовною структурою.

При навчанні на послідовності з 15 нот також відчувається недостатня мелодичність та загальна музична структуризація.

При навчанні на послідовності з 25 у мелодії відчувається наявність у мелодії мелодійних інтервалів, ритмічних мотивів. Це пояснюється тим, що 25 нот середня кількість нот у музичній фразі, яка є логічною музичною послідовністю.

4.7.2 Генерація

Під час генерації нових музичних сегментів порівняно результати генерації з різним параметром температури. Даний параметр вказує наскільки творчою має бути робота моделі. Розглянуто генерацію з показниками температури 1, 2, 3.

При показнику температури 1 мелодія має часто повторювані одні й ті самі ноти, повністю відсутня мелодичність та гармонічність.

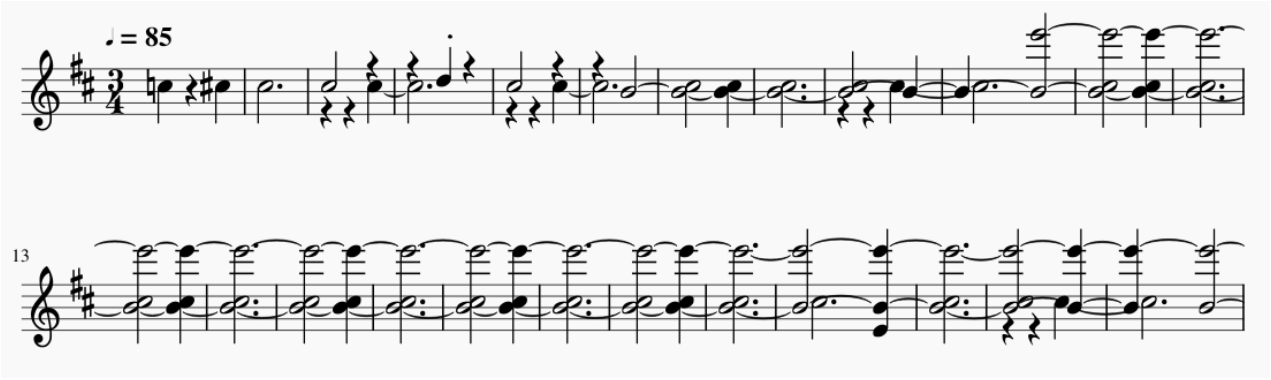


Рисунок 4.9 – Згенеровані ноти при температурі 1

При показнику температури 2 мелодія характеризується мелодичністю, проте все ще зустрічаються часті повторення однакових нот у мелодії.

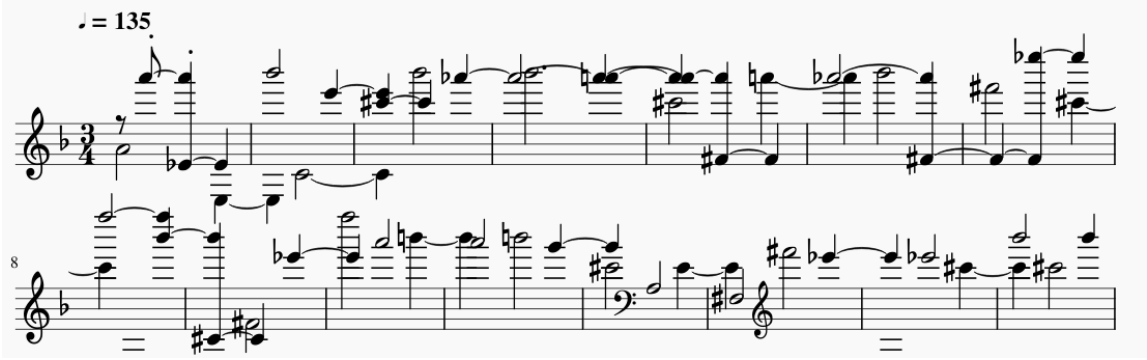


Рисунок 4.10 – Згенеровані ноти при температурі 2

При показнику температури 3 мелодія має послідовну структуру та гармонічне звучання з креативними елементами

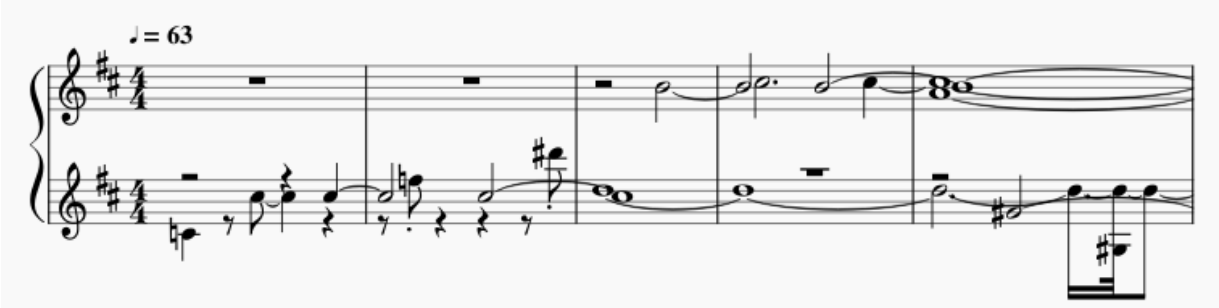


Рисунок 4.11 – Згенеровані ноти при температурі 3

Отже показник температури 3 є найкращим для генерації завершення мелодії.

4.8 Перспективи розробки

Серед подальших напрямків розвитку та вдосконалення системи слід зазначити:

– Розробка графічного інтерфейсу, для зручності користувача;
– Розширення системи, шляхом додавання можливості завершення творів інших композиторів;

– Розширення можливостей системи для роботи з іншими музичними інструментами
– Розширення можливостей системи для роботи з декількома музичними інструментами одночасно.

4.9 Висновки до розділу

У цьому розділі розроблено програмне забезпечення для автоматизованої системи створення продовження музичних композицій. Система створює продовження двох незавершених творів композитора Франца Шуберта. Реалізовану систему навчено на більш ніж 20 творах композитора. В ході реалізації моделі підібрано параметри системи для найкращого подальшого результату роботи системи.

ВИСНОВКИ

а) Проаналізовано наявні системи автоматичного закінчення мелодій комп’ютерними засобами. Серед існуючих можливих варіантів рішення поставленої задачі не знайдено системи, що задовольнятиме всім вимогам;

б) У роботі розглянуто основні методи, що можна використати при розробці системи створення музичних творів: ланцюги Маркова, генеративні змагальні мережі, рекурентні нейронні мережі з довгою короткочасною пам'яттю. У результаті проведеного порівняльного аналізу методів, для вирішення поставленої задачі обрано рекурентну нейронну мережу з довгою короткочасною пам'яттю, або LSTM. На детальному прикладі розглянуто принцип роботи LSTM, а також детально розібрано її структуру. А саме розглянуто принцип роботи трьох вентилів, вхідній вентиль, вентиль забування, вихідний вентиль, з яких скаладється LSTM;

в) Спроектовану модель реалізовано у програмі. Модель навчено на більш ніж 20 творах Франца Шуберта, що дозволило моделі освоїти стиль та характеристики цього композитора, та проведено випробування з підбору параметрів для моделі. Визначено, що найкращі результати навчання досягаються при навчанні в 100 епох, при довжині послідовності в 25 нот;

г) В результаті роботи системи було запропоновано завершення двох незавершених творів Франца Шуберта. Досягнуті результати підтверджують ефективність застосування рекурентних нейронних мереж з довгою короткочасною пам'яттю для створення музичних творів. LSTM здатна зберігати і використовувати довготривалу інформацію, що є важливим аспектом у музичній композиції, де попередні ноти можуть впливати на наступні.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hiller L. Experimental Music. Composition with an Electronic Computer / L. Hiller, J. Isaacson., 1959. – 306 с. – (Duke University Press).

2. Cope D. Experiments in Musical Intelligence / David Cope., 1996. – 263 с. – (A-R Editions).

3. Michael C. M. Neural network music composition by prediction: Exploring the benefits ofpsychoacoustic constraints and multiscale processing / Mozer Michael C.., 1994. – 280 с. – (Department of Computer Scienceand Institute of Cognitive ScienceUniversity of Colorado).

4. What is the Curse of the Ninth – and does it really exist? [Електронний ресурс] // classicfm. – 2019. – Режим доступу до ресурсу: https://www.classicfm.com/discover-music/curse-of-the-ninth/.

5. DK. The Classical Music Book: Big Ideas Simply Explained / DK., 2018. – 352 с.

6. DK. The Complete Classical Music Guide / DK., 2012. – 352 с.

7. Greff K. LSTM: A Search Space Odyssey / Klaus Greff., 2015.

8. Activation Function Definition [Електронний ресурс] – Режим доступу до ресурсу: https://deepai.org/machine-learning-glossary-and-terms/activation-function.

9. Гете Й. Фауст / Йоган Вольфганг Гете. – Київ: Андронум, 2021. – 329 с.

10. NumPy documentation [Електронний ресурс] – Mode of access: https://numpy.org/doc/.

11. pretty_midi [Електронний ресурс] – Режим доступу до ресурсу: https://craffel.github.io/pretty-midi/.

12. Goodfellow I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville., 2016.

13. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow / Aurélien Géron., 2019.

14. Aiva [Електронний ресурс] – Режим доступу до ресурсу: https://www.nvidia.com/en-us/research/ai-art-gallery/artists/aiva/.

15. Amper Music [Електронний ресурс] – Режим доступу до ресурсу: https://gigazine.net/news/20170406-amper-music/.

16. 4 Best AI Music Generators in 2023 [MuseNet, Aiva, Jukedeck & More] [Електронний ресурс] – Режим доступу до ресурсу: https://handlblogs.com/best-ai-music-generators/.

Додаток А

Лістинги програм

```
import collections
import datetime
import fluidsynth
import glob
import numpy as np
import pathlib
import pandas as pd
import pretty_midi
import seaborn as sns
import tensorflow as tf

from IPython import display
from matplotlib import pyplot as plt
from typing import Optional

seed = 42
tf.random.set_seed(seed)
np.random.seed(seed)

_SAMPLING_RATE = 16000

def display_audio(pm, seconds=30):
  waveform = pm.fluidsynth(fs=_SAMPLING_RATE)
  waveform_short = waveform[:seconds*_SAMPLING_RATE]
  return display.Audio(waveform_short, rate=_SAMPLING_RATE)

def display_audio_2(pm, seconds=30):
  waveform = pm.fluidsynth(fs=_SAMPLING_RATE)
  waveform_short = waveform
  return display.Audio(waveform_short, rate=_SAMPLING_RATE)

def decode_midi_to_notes(midi_file):
  pr_mid = pretty_midi.PrettyMIDI(midi_file)
  instrument = pr_mid.instruments[0]
```

```
    notes = {
        'pitch': [],
        'start': [],
        'end': [],
        'step': [],
        'duration': []
    }

    sorted_notes = sorted(instrument.notes, key=lambda note: note.start)
    prev_start = sorted_notes[0].start

    for note in sorted_notes:
        start = note.start
        end = note.end
        notes['pitch'].append(note.pitch)
        notes['start'].append(start)
        notes['end'].append(end)
        notes['step'].append(start - prev_start)
        notes['duration'].append(end - start)
        prev_start = start

    return pd.DataFrame(notes)

def encode_notes_to_midi(notes, out_file,instrument_name, velocity: int = 100):

    pm = pretty_midi.PrettyMIDI()
    instrument = pretty_midi.Instrument(program=pretty_midi.instrument_name_to_program(instrument_name))
    first_start = 0
    for i, note in notes.iterrows():
        start = float(first_start + note['step'])
        end = float(start + note['duration'])
        note = pretty_midi.Note(velocity=velocity,pitch=int(note['pitch']),start=start,end=end,)
        instrument.notes.append(note)
        first_start = start

    pm.instruments.append(instrument)
    pm.write(out_file)
    return pm

def create_sequences(dataset, length, vocab_size = 128,):
    length = length+1
```

```
    windows = dataset.window(length, shift=1, stride=1)
    flatten = lambda x: x.batch(length)
    sequences = windows.flat_map(flatten)
    def norm_pitch(x):
        x = x/[vocab_size,1.0,1.0]
        return x
    def split_labels(sequences):
        inputs = sequences[:-1]
        labels_dense = sequences[-1]
        labels = {key:labels_dense[i] for i,key in enumerate(keys)}

        return norm_pitch(inputs), labels

    return sequences.map(split_labels, num_parallel_calls=tf.data.AUTOTUNE)

def mse_with_positive_pressure(y_true, y_pred):
    mse = (y_true - y_pred) ** 2
    positive_pressure = 10 * tf.maximum(-y_pred, 0.0)
    return tf.reduce_mean(mse + positive_pressure)

def predict_note(notes, model, temperature):
  inputs = tf.expand_dims(notes, 0)
  predictions = model.predict(inputs)
  pitch_logits = predictions['pitch']
  step = predictions['step']
  duration = predictions['duration']
  pitch_logits /= temperature
  pitch = tf.random.categorical(pitch_logits, num_samples=1)
  pitch = tf.squeeze(pitch, axis=-1)
  duration = tf.squeeze(duration, axis=-1)
  step = tf.squeeze(step, axis=-1)
  step = tf.maximum(0, step)
  duration = tf.maximum(0, duration)
  return int(pitch), float(step), float(duration)

seq_length = 25
vocab_size = 128
batch_size = 64

filenames = glob.glob(str('drive/MyDrive/diploma/midi/chopin/*.midi*'))
```

```
sample_file = 'drive/MyDrive/diploma/midi/unfinished/MIDI-Unprocessed_17_R2_2008_01-04_ORIG_MID--AUDIO_17_R2_2008_wav--1.midi'
pm = pretty_midi.PrettyMIDI(sample_file)
display_audio(pm)

raw_notes = decode_midi_to_notes(sample_file)

notes_list = []
for f in filenames:
   notes = decode_midi_to_notes(f)
   notes_list.append(notes)
notes_list = pd.concat(notes_list)
keys = ['pitch', 'step', 'duration']
notes_for_train = np.stack([notes_list[key] for key in keys], axis=1)
notes_df = tf.data.Dataset.from_tensor_slices(notes_for_train)
seq_ds = create_sequences(notes_df, seq_length, vocab_size)

n_notes = len(notes_list)
buffer_size = n_notes - seq_length
train_ds = (seq_ds.shuffle(buffer_size).batch(batch_size, drop_remainder=True).cache().prefetch(tf.data.experimental.AUTOTUNE))

input_shape = (seq_length, 3)
learning_rate = 0.005

inputs = tf.keras.Input(input_shape)
x = tf.keras.layers.LSTM(128)(inputs)

outputs = {
  'pitch': tf.keras.layers.Dense(128, name='pitch')(x),
  'step': tf.keras.layers.Dense(1, name='step')(x),
  'duration': tf.keras.layers.Dense(1, name='duration')(x),
}

model = tf.keras.Model(inputs, outputs)

loss = {
     'pitch': tf.keras.losses.SparseCategoricalCrossentropy(
        from_logits=True),
     'step': mse_with_positive_pressure,
     'duration': mse_with_positive_pressure,
```

```
}

optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

model.compile(loss=loss, optimizer=optimizer)

model.summary()

losses = model.evaluate(train_ds, return_dict=True)

model.compile(
    loss=loss,
    loss_weights={'pitch': 0.05, 'step': 1.0,'duration':1.0,},
    optimizer=optimizer,
)

model.evaluate(train_ds, return_dict=True)

callbacks = [
    tf.keras.callbacks.ModelCheckpoint(
        filepath='./training_checkpoints/ckpt_{epoch}',
        save_weights_only=True),
    tf.keras.callbacks.EarlyStopping(
        monitor='loss',
        patience=5,
        verbose=1,
        restore_best_weights=True),
]

# Commented out IPython magic to ensure Python compatibility.
# %%time
# epochs = 100
#
# history = model.fit(train_ds,epochs=epochs,callbacks=callbacks)

plt.plot(history.epoch, history.history['loss'], label='total loss')
plt.show()

"""##Main

"""
```

```
model = tf.keras.models.load_model('drive/MyDrive/diploma/checpoint/saved_model/diploma_v1',custom_objects={'mse_with_positive_pressure': mse_with_positive_pressure})

sample_file = 'drive/MyDrive/diploma/midi/unfinished/MIDI-Unprocessed_17_R2_2008_01-04_ORIG_MID--AUDIO_17_R2_2008_wav--1.midi'

raw_notes = decode_midi_to_notes(sample_file)
temperature = 3.0
num_predictions = 200

sample_notes = np.stack([raw_notes[key] for key in keys], axis=1)

input_notes = (
    sample_notes[len(sample_notes)-seq_length*2:-seq_length] / np.array([vocab_size, 1, 1]))

notes_generated = []
prev_start = 0
for _ in range(num_predictions):
    pitch, step, duration = predict_note(input_notes, model, temperature)
    start = prev_start + step
    end = start + duration
    input_note = (pitch, step, duration)
    notes_generated.append((*input_note, start, end))
    input_notes = np.delete(input_notes, 0, axis=0)
    input_notes = np.append(input_notes, np.expand_dims(input_note, 0), axis=0)
    prev_start = start

notes_generated = pd.DataFrame(
    notes_generated, columns=(*keys, 'start', 'end'))

out_file = 'output.mid'
out_pm = encode_notes_to_midi(
    notes_generated, out_file=out_file, instrument_name=instrument_name)
display_audio_2(out_pm)
```

Додаток Б

Ілюстративний матеріал

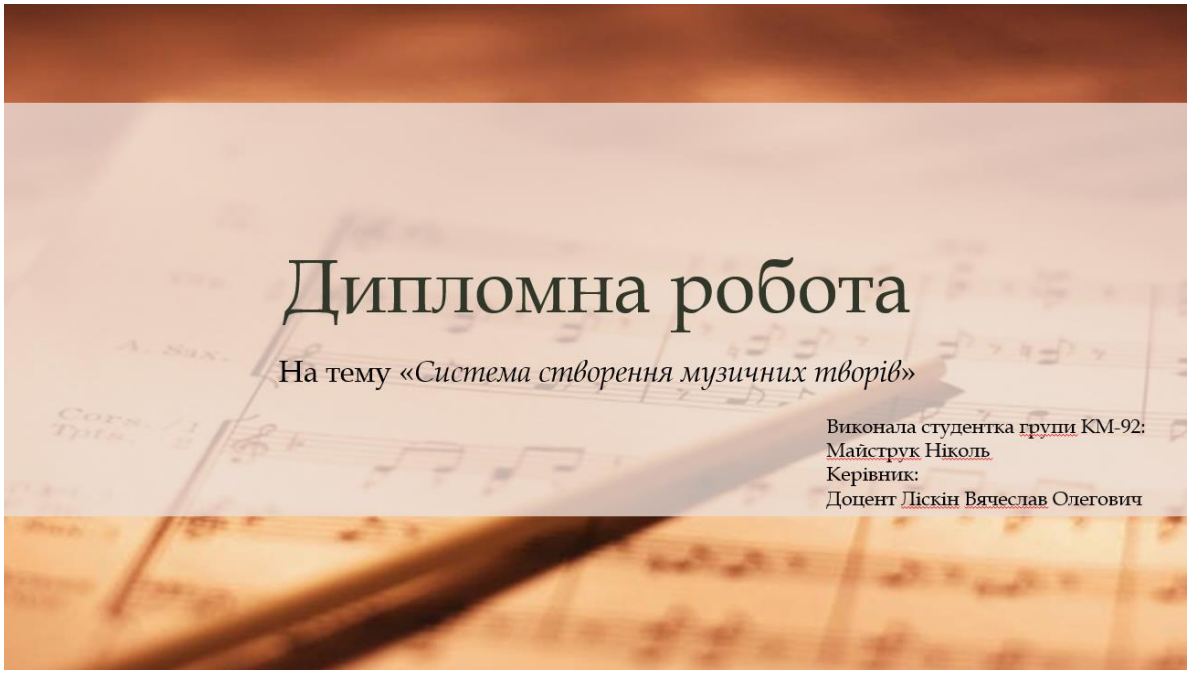


Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

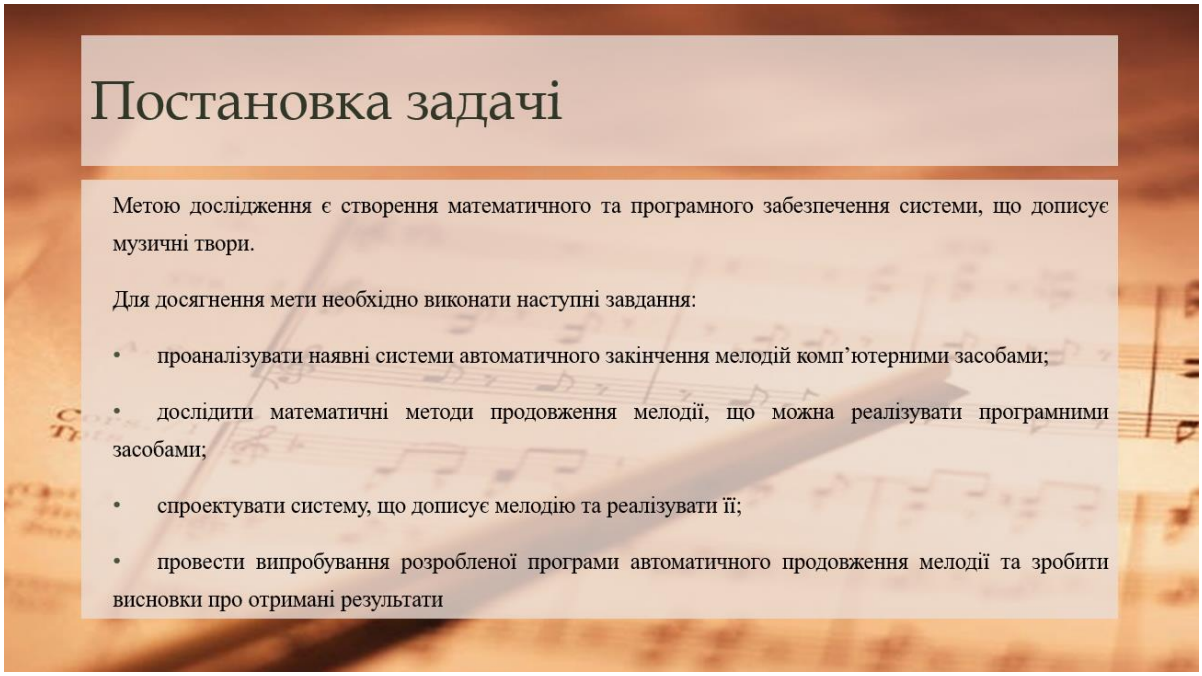


Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

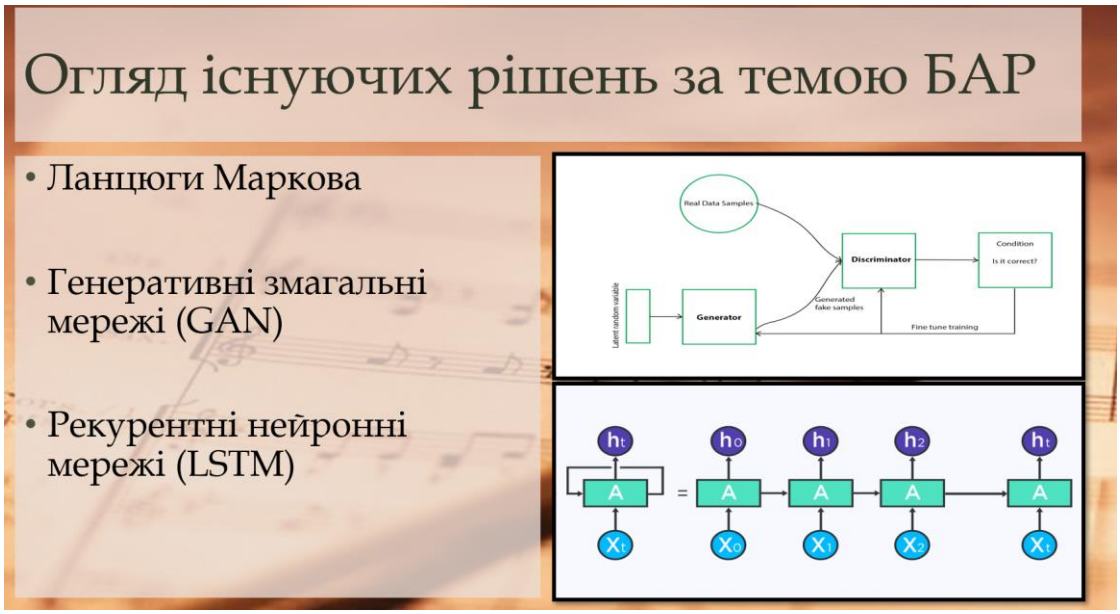


Рисунок Б.6 – Слайд 6

Рисунок Б.7 – Слайд 7

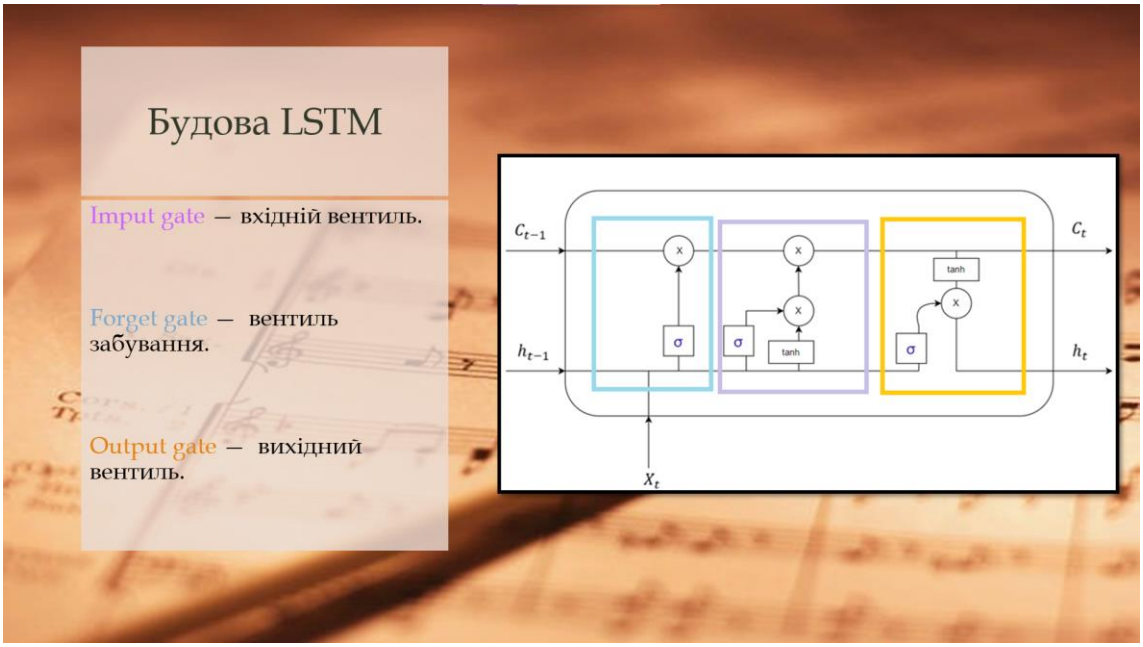


Рисунок Б.8 – Слайд 8

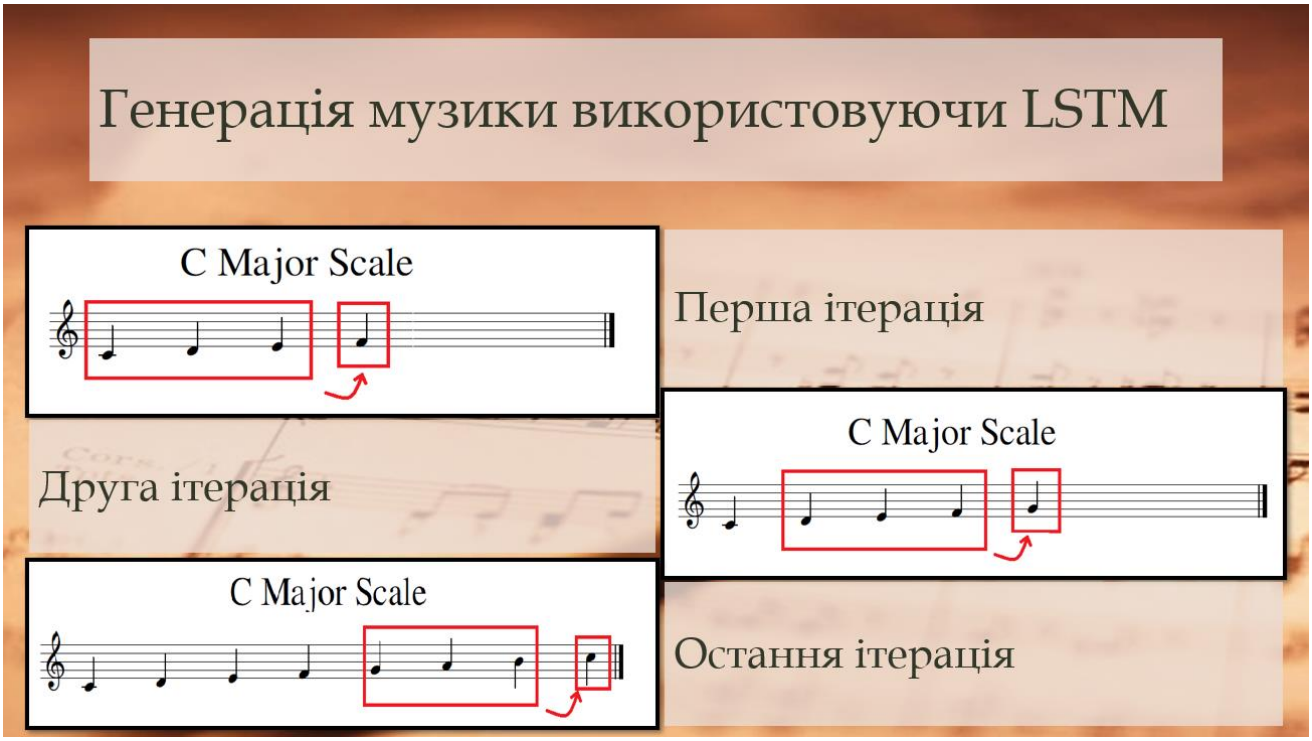


Рисунок Б.9 – Слайд 9

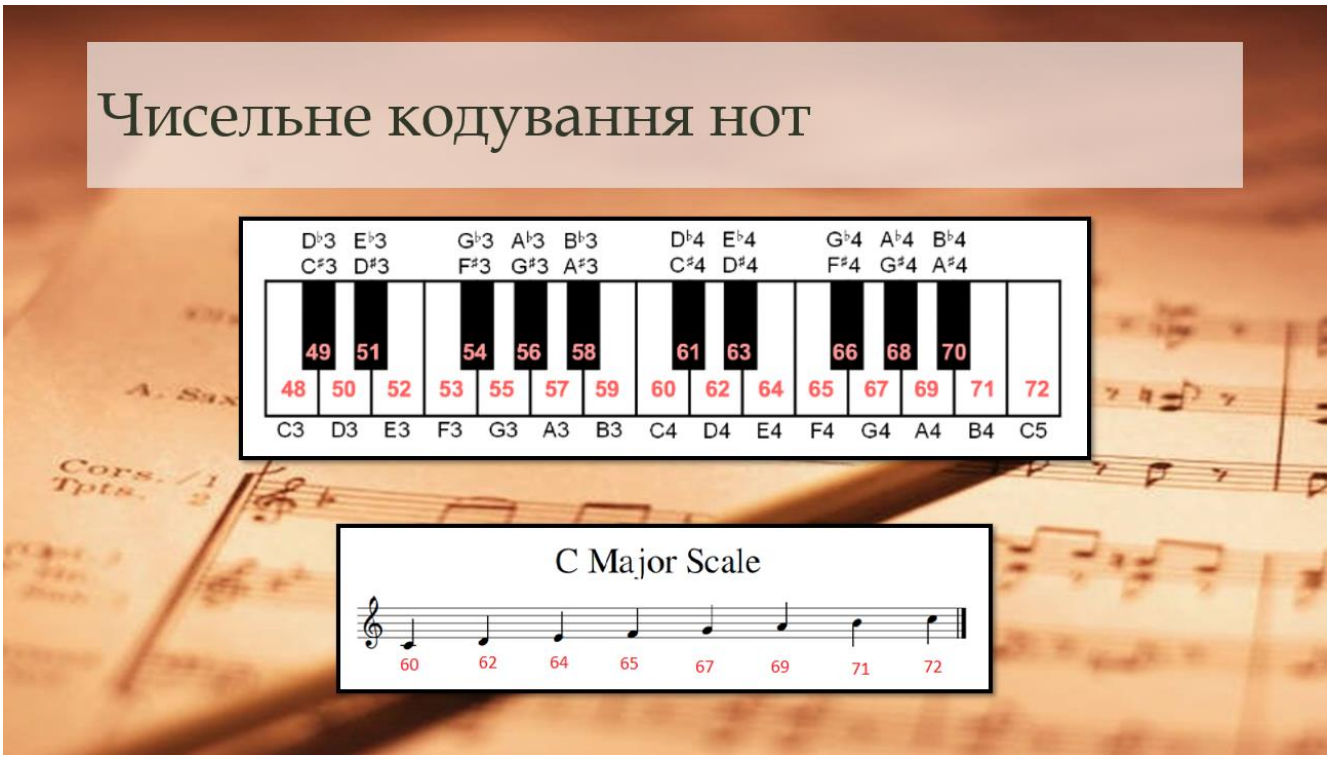
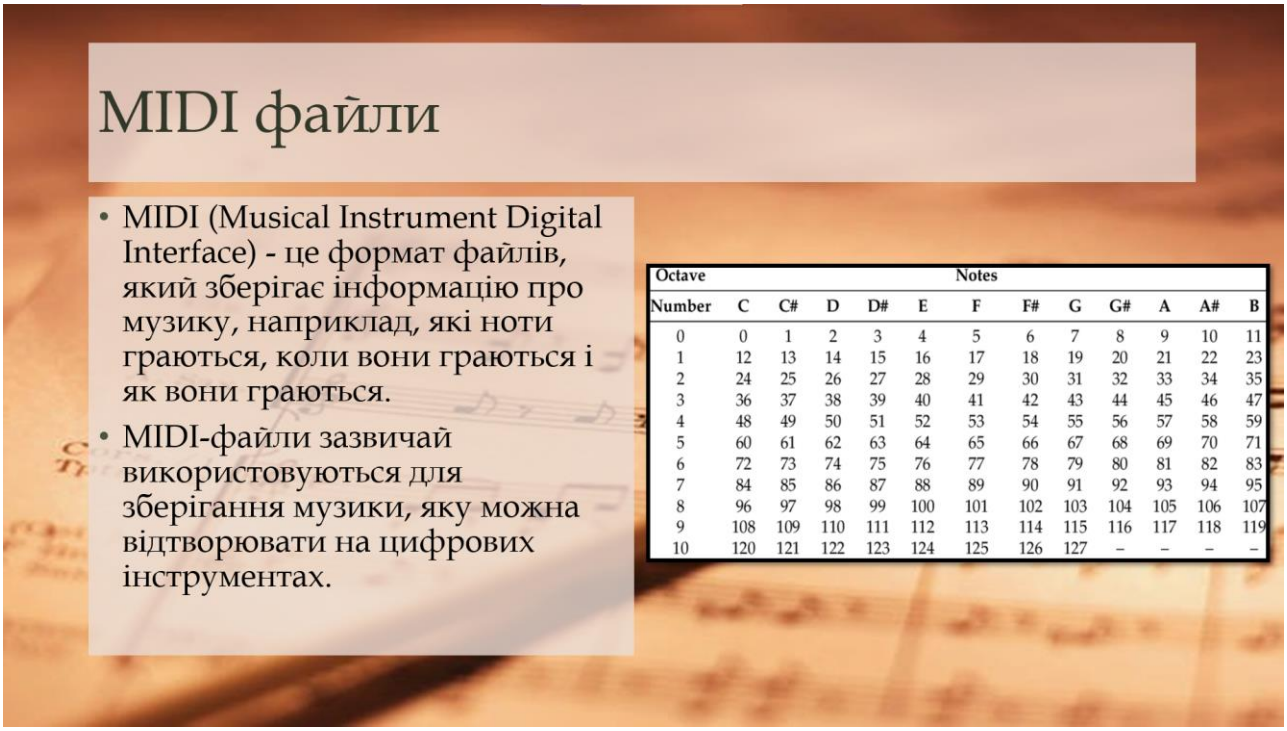


Рисунок Б.10 – Слайд 10



Octave Number	Notes											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127	–	–	–	–

Рисунок Б.11 – Слайд 11

Рисунок Б.12 – Слайд 12

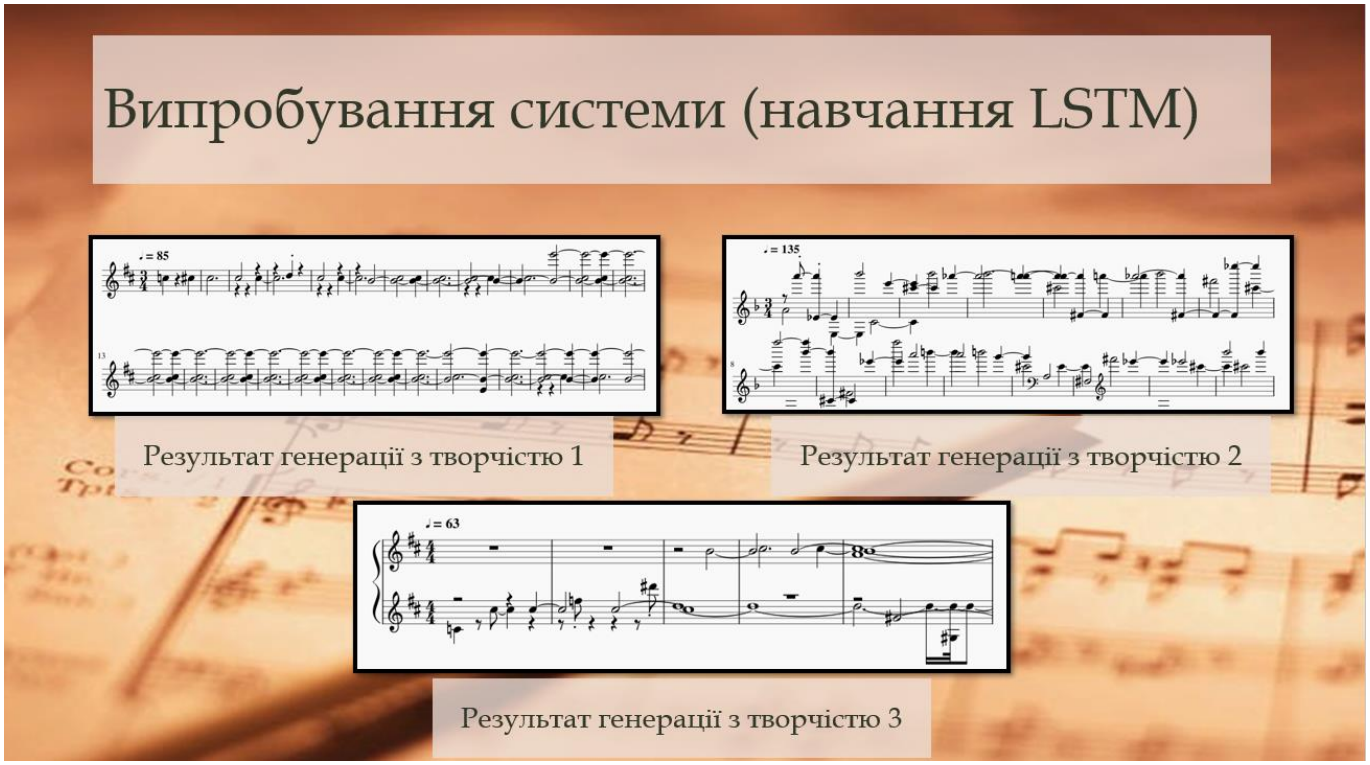


Рисунок Б.13 – Слайд 13

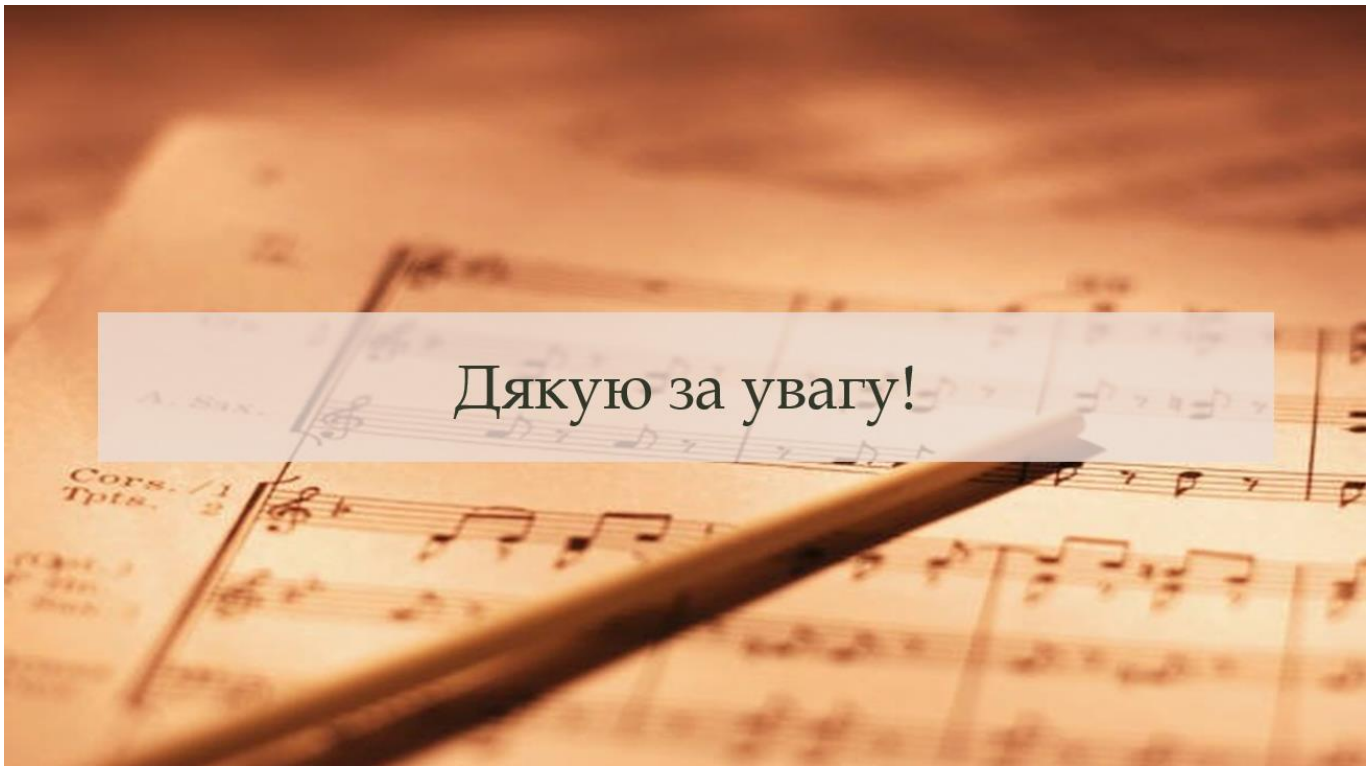


Рисунок Б.14 – Слайд 14