НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2026 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Методи та моделі генеративного штучного інтелекту для синтезу музичних творів»

Виконав:
студент IV курсу, групи КА-22
Радченко Михайло Станіславович _____

Керівник:
Доцент кафедри ММСА, к.т.н., доц.
Зінченко Артем Юрійович _____

Консультант з економічного розділу:
Доцент кафедри ЕК, к.е.н., доц.
Рощина Надія Василівна _____

Консультант з нормоконтролю:
Доцент кафедри ММСА, к.ф.-м.н.
Статкевич Віталій Михайлович _____

Рецензент:
Доцент кафедри математичного моделювання
та аналізу даних НН ФТІ, к.т.н., с.д. _____
Хайдуров Владислав Володимирович

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2026 р.

ЗАВДАННЯ
на дипломну роботу студенту
Радченку Михайлу Станіславовичу

1. Тема роботи «Методи та моделі генеративного штучного інтелекту для синтезу музичних творів», керівник роботи Зінченко Артем Юрійович, кандидат технічних наук, затверджені наказом по університету від 27.05.2026 р. № 1944-с.
2. Термін подання студентом роботи «___» червня 2026 р.
3. Вихідні дані до роботи: датасет, основні відомості з машинного навчання.
4. Зміст роботи: теоретичні основи генерації музики засобами штучного інтелекту; обґрунтування методів та підготовка даних; розроблення, навчання та дослідження моделі.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Функціонально-вартісний аналіз	Рощина Н.В., доц., к.е.н.		

7. Дата видачі завдання 26 лютого 2026 року

Календарний план

№ з/п	Назва етапів виконання бакалаврської дипломної роботи (БДР)	Термін виконання етапів роботи	Примітка
1	Затвердження теми БДР, опанування ДСТУ 3008:2015	26.02.2026-29.02.2026	виконано
2	Огляд властивостей нейронних мереж	01.03.2026-24.03.2026	виконано
3	Розбір основних архітектур моделей штучного інтелекту	26.03.2026-31.03.2026	виконано
4	Проектування алгоритмів і розробка програмного забезпечення	01.04.2026-15.05.2026	виконано
5	Оформлення пояснювальної записки (ПЗ), перевірка правильності структури та оформлення	16.05.2026-28.05.2026	виконано
6	Оновлення програмного забезпечення та відповідне оформлення розділу програмного забезпечення	29.05.2026-30.05.2026	виконано
7	Остаточне оформлення ПЗ, підготовка презентації	31.05.2026-10.06.2026	виконано
8	Передзахист БДР	11.06.2026	виконано

Студент _____ Михайло РАДЧЕНКО

Керівник _____ Артем ЗІНЧЕНКО

РЕФЕРАТ

Дипломна робота: 113 с., 14 табл., 7 рис., 2 дод., 30 джерел.

TRANSFORMER, GAN, VAE, DIFFUSION, PYTHON, МОДЕЛІ ТА АРХІТЕКТУРИ ШТУЧНОГО ІНТЕЛЕКТУ, ГЕНЕРАЦІЯ МУЗИЧНИХ ТВОРІВ.

Об’єктом дослідження є процес автоматизованої генерації музичних творів із використанням технологій штучного інтелекту.

Предметом дослідження є методи, алгоритми та моделі генеративного штучного інтелекту, призначені для синтезу музичних композицій у форматах MIDI та цифрового аудіо.

Метою роботи є дослідження сучасних методів та моделей генеративного штучного інтелекту для синтезу музичних творів, аналіз їх можливостей, розроблення власної моделі генерації музики та оцінювання її ефективності.

Інтенсивний розвиток методів і технологій штучного інтелекту актуалізує проблему вибору оптимальної архітектури інтелектуальних систем. Незважаючи на значну кількість досліджень у цій галузі, питання обґрунтування вибору архітектурних рішень залишається недостатньо висвітленим, що зумовлює актуальність проведеного дослідження.

Результатом роботи є реалізована генеративна модель на базі архітектури Transformer для синтезу музичних композицій у символьному форматі MIDI. Дослідження підтвердило, що розроблена модель здатна створювати унікальні музичні послідовності без прямого копіювання навчальних фрагментів.

Подальший розвиток роботи полягає в покращенні та оптимізації моделей на архітектурі Transformer для більшої доступності цього рішення.

ABSTRACT

Thesis: 113 p., 14 tabl., 7 fig., 2 app., 30 ref.

TRANSFORMER, GAN, VAE, DIFFUSION MODEL, PYTHON, ARTIFICIAL INTELLIGENCE MODELS AND ARCHITECTURES, MUSIC GENERATION.

Object of research is the set of the automated music generation processes of using artificial intelligence technologies.

Subject of research is the set of the methods, algorithms, and generative artificial intelligence models designed for the synthesis of musical compositions in MIDI and digital audio formats.

Purpose of the research is to investigate modern methods and models of generative artificial intelligence for music synthesis, analyze their capabilities, develop a custom music generation model, and evaluate its effectiveness.

Relevance of the study lays in the rapid development of the artificial intelligence field has introduced the challenge of selecting the most appropriate architecture for specific tasks. Despite significant progress in generative AI, this issue remains insufficiently explored in the context of music generation, which determines the relevance of this research.

Research results based on the conducted analysis, a generative model based on the Transformer architecture was selected, justified, and implemented for the synthesis of musical compositions in the symbolic MIDI format. The study confirmed that the developed model is capable of generating unique musical sequences without directly reproducing fragments from the training dataset.

Future development is supported by further improvement and optimization of Transformer-based models to increase the accessibility, efficiency, and practical applicability of the proposed solution.

ЗМІСТ

ВСТУП

Стрімкий розвиток цифрових технологій та систем штучного інтелекту суттєво змінює підходи до створення, обробки та поширення інформації в сучасному суспільстві. Одним із найбільш динамічних напрямів розвитку штучного інтелекту є генеративні моделі, які здатні автоматично створювати новий контент на основі аналізу великих масивів даних. Якщо раніше технології штучного інтелекту використовувалися переважно для класифікації, прогнозування та підтримки прийняття рішень, то сьогодні вони активно застосовуються у творчих сферах діяльності людини, зокрема в образотворчому мистецтві, літературі, кінематографії та музичній індустрії.

Музика є одним із найскладніших видів творчої діяльності, оскільки процес її створення потребує поєднання ритмічних структур, гармонійних послідовностей, мелодичних ліній та емоційного наповнення композиції. Традиційно написання музичних творів вимагало значних професійних знань і творчих навичок композитора. Проте розвиток методів машинного навчання та глибоких нейронних мереж відкрив нові можливості для автоматизації процесу музичної творчості. Сучасні генеративні системи здатні аналізувати тисячі музичних композицій, виявляти приховані закономірності їх побудови та на основі отриманих знань створювати нові музичні твори, які можуть відповідати певному стилю, жанру або набору заданих характеристик.

Особливої актуальності набувають дослідження, пов'язані із застосуванням генеративного штучного інтелекту для синтезу музичних творів. Сучасні архітектури нейронних мереж, серед яких генеративно-змагальні мережі (GAN), варіаційні автоенкодери (VAE), трансформери (Transformer) та дифузійні моделі (Diffusion Models), демонструють високі результати у створенні аудіоконтенту та музичних послідовностей. Використання таких моделей дозволяє не лише автоматизувати процес

створення музики, але й значно розширити творчі можливості музикантів, композиторів та розробників мультимедійних систем. Крім того, генерація музики засобами штучного інтелекту знаходить практичне застосування у відеоіграх, кіноіндустрії, рекламі, онлайн-сервісах потокового відтворення музики та персоналізованих рекомендаційних системах.

Актуальність теми також обумовлена стрімким розвитком генеративних моделей нового покоління та постійним зростанням обсягів доступних музичних даних для навчання штучного інтелекту. Незважаючи на значний прогрес у цій галузі, питання вибору ефективних моделей генерації, підготовки навчальних вибірок, оптимізації процесу навчання та оцінювання якості синтезованих музичних творів залишаються недостатньо дослідженими та потребують подальшого наукового опрацювання. Саме тому дослідження методів і моделей генеративного штучного інтелекту для синтезу музичних творів є важливим та перспективним напрямом сучасних наукових досліджень.

Об'єктом дослідження є процес автоматизованої генерації музичних творів із використанням технологій штучного інтелекту.

Предметом дослідження є методи, алгоритми та моделі генеративного штучного інтелекту, призначені для синтезу музичних композицій у форматах MIDI та цифрового аудіо.

Метою роботи є дослідження сучасних методів та моделей генеративного штучного інтелекту для синтезу музичних творів, аналіз їх можливостей, розроблення власної моделі генерації музики та оцінювання ефективності її роботи.

Для досягнення поставленої мети необхідно виконати аналіз предметної області генерації музики засобами штучного інтелекту та дослідити сучасні підходи до автоматизованого створення музичних композицій. Необхідно розглянути принципи функціонування генеративно-змагальних мереж,

варіаційних автоенкодерів, трансформерів та дифузійних моделей, визначити їх переваги та недоліки у задачах музичного синтезу. Важливим етапом є обґрунтування вибору методів та інструментів реалізації, а також формалізація задачі генерації музики. У процесі виконання роботи необхідно сформувати та підготувати датасет музичних творів, виконати попередню обробку даних, нормалізацію та кодування музичної інформації. Подальшим етапом є розроблення та програмна реалізація моделі генерації музики з використанням сучасних засобів машинного навчання. Після завершення реалізації моделі необхідно провести її навчання та тестування, виконати генерацію музичних фрагментів і здійснити оцінювання якості отриманих результатів за допомогою відповідних метрик. Завершальним етапом дослідження є аналіз отриманих результатів, порівняння різних підходів до генерації музики та формулювання висновків щодо ефективності використання генеративного штучного інтелекту для створення музичних творів.

Практичне значення роботи полягає у можливості використання отриманих результатів для створення програмних систем автоматизованої генерації музики, розроблення інтелектуальних музичних сервісів, підтримки творчої діяльності композиторів та музикантів, а також для подальших досліджень у галузі штучного інтелекту та цифрової творчості.

У процесі виконання роботи використовуються методи аналізу даних, математичного моделювання, машинного навчання, глибокого навчання, програмної інженерії та оцінювання якості результатів генерації музичних творів. Отримані результати можуть бути використані як основа для подальшого вдосконалення технологій генеративного штучного інтелекту та їх практичного впровадження у музичну індустрію.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ГЕНЕРАЦІЇ МУЗИКИ ЗАСОБАМИ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Аналіз сучасних підходів до автоматизованого створення музики

Автоматизоване створення музики є одним із найбільш перспективних напрямів розвитку сучасних систем штучного інтелекту. Протягом останніх десятиліть методи генерації музичних творів пройшли шлях від простих алгоритмічних підходів до складних нейромережевих моделей, здатних створювати композиції, які за своїм звучанням та структурою наближаються до творів, написаних професійними музикантами. Зростання обчислювальних можливостей комп'ютерних систем, поява великих музичних датасетів та розвиток методів машинного навчання сприяли активному впровадженню автоматизованих технологій у сферу музичної творчості.

Перші системи автоматизованого створення музики базувалися на використанні жорстко визначених правил композиції. Такі системи створювали музичні послідовності відповідно до наперед заданих музичних законів, гармонічних схем та ритмічних шаблонів. Основною перевагою подібних підходів була можливість контролювати структуру композиції та забезпечувати відповідність музичним канонам. Проте через обмеженість закладених правил створені композиції часто характеризувалися передбачуваністю та недостатньою різноманітністю.

Подальший розвиток автоматизованого створення музики був пов'язаний із застосуванням статистичних методів аналізу музичних даних. Одним із найбільш відомих підходів стали марковські моделі, які дозволяли генерувати музичні послідовності на основі ймовірностей переходу між окремими нотами або акордами. У таких системах кожна наступна музична подія визначалася на основі попереднього стану або групи попередніх станів.

Використання марковських процесів дозволило підвищити варіативність створюваної музики, однак такі моделі не могли ефективно враховувати довгострокові залежності між музичними фрагментами.

З появою технологій машинного навчання автоматизоване створення музики отримало новий імпульс розвитку. Алгоритми машинного навчання дозволили аналізувати великі масиви музичних даних та виявляти приховані закономірності побудови музичних творів без необхідності явного програмування правил композиції. На цьому етапі активно почали використовуватися штучні нейронні мережі, здатні навчатися на прикладах існуючих музичних композицій та відтворювати характерні особливості певного стилю або жанру.

Особливого поширення набули рекурентні нейронні мережі (RNN), які виявилися ефективними для роботи з послідовними даними. Оскільки музика являє собою часову послідовність нот, акордів та ритмічних елементів, рекурентні мережі дозволяли враховувати залежності між музичними подіями та формувати більш логічні композиції. Подальшим розвитком цього підходу стали мережі довгої короткочасної пам'яті (LSTM), які забезпечили можливість врахування довгострокових музичних залежностей та значно покращили якість синтезованих творів.

Сучасний етап розвитку автоматизованого створення музики характеризується активним використанням генеративних моделей штучного інтелекту. Одним із найбільш відомих напрямів є застосування генеративно-змагальних мереж (GAN). Такі моделі складаються з двох нейронних мереж: генератора та дискримінатора. Генератор створює нові музичні фрагменти, тоді як дискримінатор оцінює їхню схожість із реальними композиціями. У процесі навчання обидві мережі постійно вдосконалюються, що дозволяє отримувати високоякісні результати генерації.

Іншим важливим напрямом є використання варіаційних автоенкодерів (VAE). Дані моделі дозволяють стискати музичну інформацію у компактний латентний простір та виконувати генерацію нових композицій шляхом модифікації внутрішніх представлень даних. Основною перевагою варіаційних автоенкодерів є можливість плавного переходу між різними музичними стилями та створення нових варіантів композицій на основі наявних прикладів.

Значний прорив у сфері генерації музики стався після появи архітектури Transformer. На відміну від рекурентних мереж, трансформери використовують механізм самоуваги, що дозволяє одночасно аналізувати всі елементи музичної послідовності та враховувати залежності між ними незалежно від відстані. Завдяки цьому моделі на основі Transformer демонструють високу ефективність у створенні складних музичних композицій із тривалою структурною цілісністю. Саме на цій архітектурі базуються сучасні системи генерації музики, здатні створювати багатоголосні композиції та імітувати творчий стиль відомих композиторів.

Останнім часом значну увагу дослідників привертають дифузійні моделі, які спочатку отримали популярність у сфері генерації зображень, а згодом були адаптовані для роботи з аудіоданими. Принцип роботи таких моделей полягає у поступовому відновленні корисного сигналу із випадкового шуму. Дифузійні моделі демонструють високу якість генерації аудіоконтенту та здатні створювати реалістичні музичні композиції з природним звучанням.

Серед сучасних систем автоматизованого створення музики особливе місце займають проєкти, розроблені провідними технологічними компаніями та науковими організаціями. До найбільш відомих належать система Music Transformer від Google, модель Jukebox від OpenAI, платформа MuseNet від OpenAI та система MusicLM від Google. Зазначені рішення демонструють

можливість генерації музики різних жанрів, інструментальних композицій та навіть музичних творів на основі текстового опису.

Переваги та недоліки наведено в таблиці 1.1.

Сучасні підходи до автоматизованого створення музики дозволяють генерувати як MIDI-послідовності, так і повноцінні аудіозаписи. Формат MIDI є особливо популярним у наукових дослідженнях, оскільки він містить структуровану інформацію про ноти, тривалість звучання та інструменти, що значно спрощує процес навчання моделей штучного інтелекту. Генерація безпосередньо аудіосигналів є більш складним завданням через високу розмірність даних, однак сучасні генеративні моделі успішно справляються і з цією задачею.

Таблиця 1.1 – Порівняльна характеристика сучасних підходів до автоматизованого створення музики

Підхід	Принцип роботи	Переваги	Недоліки	Сфера застосування
Правилові системи (Rule-Based Systems)	Генерація музики на основі музичних правил, гармоній та шаблонів	Простота реалізації, контроль структури	Низька різноманітність	Навчальні системи, акомпанемент
Марковські моделі	Ймовірності переходів між нотами та акордами	Невисокі вимоги до ресурсів	Не враховують довгострокові залежності	Генерація простих мелодій
RNN	Аналіз музичних послідовностей із внутрішньою пам'яттю	Робота з часовими рядами	Проблеми довгострокових залежностей	Генерація мелодій

Продовження таблиці 1.1

LSTM	Покращена RNN з довготривалою пам'яттю	Довгостроков і залежності	Високі обчислювальн і витрати	Багататактов і композиції
GAN	Конкуренція генератора і дискримінатор а	Висока якість композицій	Складність навчання	Нові музичні стилі
VAE	Кодування у латентний простір	Контроль генерації	Можлива втрата деталізації	Варіації музичних творів
Transforme r	Механізм самоуваги	Висока якість генерації	Потребують значних ресурсів	Сучасні системи генерації
Diffusion Models	Відновлення сигналу із шуму	Реалістичне звучання	Тривале навчання	Генерація аудіо високої якості

1.2 Огляд моделей генеративного штучного інтелекту для генерації музики

1.2.1 GAN-моделі

Розвиток генеративного штучного інтелекту став одним із ключових факторів трансформації сучасної музичної індустрії. На відміну від традиційних алгоритмів композиції, які базувалися на жорстко визначених правилах та шаблонах, сучасні генеративні моделі здатні самостійно навчатися на великих масивах музичних даних та створювати нові музичні твори, які відтворюють особливості стилю, ритму, гармонії та мелодики вихідних композицій. Завдяки використанню глибоких нейронних мереж стало можливим автоматичне генерування музики різних жанрів та рівнів

складності, починаючи від простих мелодій і закінчуючи повноцінними багатоголосними композиціями.

У сфері синтезу музичних творів найбільшого поширення набули генеративно-змагальні мережі, варіаційні автоенкодери, трансформери та дифузійні моделі. Кожен із цих підходів має власні особливості архітектури, принципи навчання та сфери застосування. Вибір конкретної моделі залежить від типу музичних даних, необхідної якості генерації, обчислювальних ресурсів та вимог до кінцевого результату.

Сучасні системи генерації музики використовують як символьне представлення музичних творів у форматі MIDI, так і безпосередню генерацію аудіосигналів. Робота з MIDI-послідовностями є більш поширеною в наукових дослідженнях, оскільки такі дані мають компактне представлення та дозволяють ефективно аналізувати музичну структуру. Генерація аудіо є значно складнішою задачею через високу розмірність даних, однак сучасні генеративні моделі успішно вирішують і цю проблему.

Генеративно-змагальні мережі (Generative Adversarial Networks, GAN) належать до найбільш відомих та ефективних моделей генеративного штучного інтелекту. Вперше дана архітектура була запропонована у 2014 році і з того часу активно використовується для генерації зображень, текстів, відео та музики. Основна ідея GAN полягає у взаємодії двох нейронних мереж, які навчаються одночасно та конкурують між собою.

Архітектура GAN складається з генератора та дискримінатора. Генератор відповідає за створення нових музичних даних на основі випадкового шуму або латентного вектора. Дискримінатор виконує функцію оцінювання та визначає, чи є отриманий музичний фрагмент реальним прикладом із навчального набору даних або результатом роботи генератора. Під час навчання генератор намагається створити максимально реалістичну

музику, тоді як дискримінатор прагне правильно класифікувати згенеровані та реальні композиції.

Принцип роботи GAN можна подати у вигляді схеми:

1) випадковий шум;
2) генератор;
3) згенерована музика;
4) дискримінатор;
5) оцінка якості.

Процес навчання GAN формалізується через мінімаксну функцію втрат [6]:

$$min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1 - -D(G(z)))] ,$$

де G – генератор;

D – дискримінатор;

$p_{data}(x)$ - розподіл реальних даних;

$p_z(z)$ – розподіл випадкового шуму;

$G(z)$ – синтетичні дані, створені генератором;

$D(x)$ – ймовірність того, що приклад x є реальним.

На вхід генератора подається випадковий вектор [6]:

$$z \sim p_z(z)$$

Зазвичай використовується нормальний або рівномірний розподіл.

Генератор перетворює шумовий вектор у новий приклад [6]:

$$x_{fake} = G(z)$$

На цьому етапі створюються синтетичні зображення, аудіо або інші типи даних.

Дискрімінатор отримує: реальні дані x; синтетичні дані $G(z)$. На виході отримуємо значення [6]:

$$D(x) \epsilon [0,1],$$

де значення, близькі до 1, означають високу ймовірність того, що дані є реальними.

Після оцінювання виконується backpropagation:

1) дискримінатор оновлює параметри для покращення класифікації;
2) генератор оновлює параметри для створення більш реалістичних прикладів.

У процесі багаторазового навчання обидві мережі поступово вдосконалюються. Генератор створює все більш реалістичні музичні композиції, а дискримінатор стає більш точним у їх оцінюванні. Результатом такого навчання є модель, здатна генерувати музичні твори, які мають високу схожість із реальними музичними композиціями. Однією з перших успішних реалізацій GAN для генерації музики стала модель MidiNet. Дана система використовує згорткові нейронні мережі для створення мелодій у форматі MIDI та дозволяє враховувати попередні музичні фрази під час генерації нових фрагментів. Використання MidiNet продемонструвало можливість створення гармонійно узгоджених музичних послідовностей із відносно високим рівнем музичної цілісності.

Подальшим розвитком цього напряму стала модель MuseGAN, яка дозволяє генерувати багатодоріжкову музику, враховуючи взаємодію між різними музичними інструментами. На відміну від попередніх підходів, MuseGAN здатна створювати повноцінні музичні композиції, що містять партії ударних, басу, клавішних та інших інструментів. Це забезпечує більш природне та насичене звучання отриманих творів. Перевагою GAN-моделей є здатність створювати нові музичні композиції без необхідності явного програмування музичних правил. Навчання відбувається безпосередньо на реальних музичних творах, що дозволяє моделі автоматично виявляти закономірності побудови мелодій, гармоній та ритмічних структур. Крім того,

GAN демонструють високу різноманітність результатів генерації та можуть використовуватися для створення музики різних жанрів.

Водночас GAN-моделі мають і певні недоліки. Однією з головних проблем є нестабільність процесу навчання. У деяких випадках генератор може почати створювати обмежену кількість схожих композицій, що призводить до явища, відомого як колапс мод (Mode Collapse). Крім того, навчання GAN потребує значних обчислювальних ресурсів та ретельного налаштування параметрів моделі. Для задач генерації музики GAN найчастіше використовуються у випадках, коли необхідно створити нові музичні композиції на основі існуючих стилів або жанрів. Вони демонструють високу ефективність під час роботи з MIDI-послідовностями та можуть забезпечувати достатньо високу якість синтезованих музичних творів. Саме тому GAN залишаються одним із найбільш популярних інструментів генеративного штучного інтелекту в музичній сфері.

Генеративно-змагальні мережі є важливим напрямом розвитку систем автоматизованого створення музики. Їх використання дозволяє створювати нові музичні композиції на основі аналізу великих наборів музичних даних, забезпечуючи високий рівень різноманітності та творчої гнучкості. Незважаючи на певні обмеження, GAN-моделі продовжують активно використовуватися в сучасних дослідженнях та комерційних системах генерації музики, залишаючись одним із ключових інструментів генеративного штучного інтелекту.

1.2.2 VAE-моделі

Варіаційні автоенкодери (Variational Autoencoders, VAE) є одним із найбільш поширених типів генеративних моделей штучного інтелекту, які використовуються для створення нових даних на основі вивчення прихованих закономірностей у навчальній вибірці. У сфері генерації музики VAE-моделі

застосовуються для створення нових мелодій, гармонійних послідовностей та музичних композицій шляхом формування компактного внутрішнього представлення музичної інформації. Завдяки своїй архітектурі варіаційні автоенкодери дозволяють не лише генерувати нові музичні твори, але й виконувати модифікацію існуючих композицій, створювати плавні переходи між музичними стилями та формувати варіації мелодій.

На відміну від генеративно-змагальних мереж, які використовують механізм конкуренції між генератором та дискримінатором, VAE базуються на принципі кодування та декодування інформації. Основна мета автоенкодера полягає у стисканні вхідних даних до компактного латентного простору та подальшому відновленні початкової інформації з мінімальними втратами. У випадку варіаційного автоенкодера додатково використовується ймовірнісний підхід, який дозволяє створювати нові дані шляхом вибірки точок із латентного простору.

Архітектура VAE складається з двох основних компонентів: енкодера та декодера. Енкодер отримує на вхід музичну композицію та перетворює її у компактне латентне представлення. У процесі навчання модель формує розподіл ймовірностей, який описує приховані характеристики музичного твору. Декодер виконує зворотне перетворення та відновлює музичну композицію з латентного простору.

Принцип роботи VAE можна представити таким чином:

1) музичний твір;
2) енкодер;
3) латентний простір;
4) декодер;
5) згенерований музичний твір.

Особливістю варіаційних автоенкодерів є використання латентного простору, у якому кожна точка відповідає певному набору музичних

характеристик. Завдяки цьому стає можливим плавне переміщення між різними точками простору та створення нових композицій, які поєднують ознаки кількох музичних стилів або творів. Така властивість робить VAE особливо корисними для творчих задач, пов'язаних із музичним дизайном та експериментальною композицією.

У процесі навчання VAE оптимізує дві складові функції втрат. Перша складова відповідає за точність відновлення вихідної музичної композиції, а друга забезпечує правильний розподіл латентного простору.

Однією з найбільш відомих реалізацій варіаційних автоенкодерів для музичних задач є модель MusicVAE, розроблена дослідниками компанії Google у межах проєкту Magenta. Дана система використовує рекурентні нейронні мережі для роботи з MIDI-послідовностями та дозволяє генерувати музичні композиції значної тривалості. Особливістю MusicVAE є можливість створення плавних переходів між двома музичними фрагментами шляхом інтерполяції у латентному просторі. Застосування VAE-моделей у генерації музики має низку переваг. Насамперед вони забезпечують стабільний процес навчання порівняно з GAN-моделями. Крім того, варіаційні автоенкодери дозволяють ефективно керувати процесом генерації музики шляхом зміни параметрів латентного простору. Це відкриває можливість створення музичних композицій із заданими характеристиками, наприклад темпом, настроєм або жанром.

Ще однією перевагою VAE є здатність виконувати стилістичне змішування музичних творів. Завдяки латентному представленню можна поєднувати ознаки різних композицій та отримувати нові музичні твори, які поєднують елементи кількох жанрів. Такий підхід активно використовується у сучасних системах комп'ютерної композиції та музичного дизайну. Разом із перевагами варіаційні автоенкодери мають і певні недоліки. Через особливості роботи латентного простору деякі деталі музичних композицій можуть

втрачатися під час реконструкції. У результаті згенерована музика іноді звучить менш деталізовано порівняно з результатами, отриманими за допомогою трансформерів або сучасних дифузійних моделей. Крім того, якість генерації значною мірою залежить від структури латентного простору та параметрів навчання.

Попри зазначені обмеження, VAE-моделі залишаються одним із найважливіших інструментів генеративного штучного інтелекту для музичної творчості. Вони забезпечують баланс між якістю генерації, стабільністю навчання та можливістю керування процесом створення музичних композицій. Саме тому варіаційні автоенкодери широко використовуються у наукових дослідженнях та комерційних проєктах, пов'язаних із автоматизованим синтезом музичних творів.

1.2.3 Transformer-моделі

Архітектура Transformer є одним із найбільш значущих досягнень у сфері штучного інтелекту та машинного навчання останніх років. Поява даної моделі суттєво змінила підходи до обробки послідовних даних, включаючи текст, аудіо та музичні композиції. У задачах генерації музики Transformer-моделі продемонстрували значно кращі результати порівняно з рекурентними нейронними мережами та багатьма іншими генеративними архітектурами завдяки здатності ефективно враховувати довгострокові залежності між елементами музичної послідовності.

Музичний твір являє собою складну часову структуру, у якій окремі ноти, акорди та ритмічні елементи можуть бути пов'язані між собою навіть на значній відстані один від одного. Для створення якісної композиції модель повинна враховувати не лише поточний музичний контекст, але й загальну структуру твору, повторення тем, розвиток мелодії та гармонічні закономірності. Саме такі можливості забезпечує архітектура Transformer.

На відміну від рекурентних нейронних мереж, які обробляють послідовність поелементно, Transformer аналізує всю послідовність одночасно. Це дозволяє значно підвищити швидкість навчання та покращити якість моделювання складних залежностей між музичними подіями. Основою роботи Transformer є механізм самоуваги (Self-Attention), який визначає важливість кожного елемента послідовності відносно інших елементів.

Механізм самоуваги дозволяє моделі автоматично визначати, які музичні події мають найбільший вплив на формування наступних нот або акордів. Завдяки цьому система може враховувати як локальні музичні закономірності, так і глобальну структуру композиції.

Архітектура Transformer складається з декількох основних компонентів. До них належать механізм багатоголової уваги (Multi-Head Attention), блоки нормалізації, повнозв'язні нейронні мережі та позиційне кодування. Оскільки Transformer не використовує рекурентні зв'язки, для врахування порядку музичних подій застосовується спеціальний механізм позиційного кодування, який додає інформацію про розташування кожної ноти або акорду в послідовності.

Нехай $(x_1, x_2, \dots x_n)$ – послідовність токенів, тоді наступним етапом є перетворення токенів у векторні представлення (embeddings), що дозволяють моделі працювати з числовим поданням даних [1]:

$$d_n = E(x_n),$$

де $E \in R^{|V| \times i}$, i – розмірність прихованого простору (ембедінгу), $|V|$ – розмір словника токенів, $d_n \in R^i$.

Через те, що, на відміну від рекурентних мереж, трансформер не має вбудованого механізму врахування порядку токена у послідовності, до кожного вектору токена додається інформація про позицію токена у послідовності. Реалізується це через [1]:

$$z_n = d_n + p_n$$

де p_n – вектор, який відповідає за індекс токена n у послідовності.

Одним із найпоширеніших способів побудови вектора p_n є синусоїдальне позиційне кодування [1]:

$$p_{n,2j} = sin\left(\frac{n}{10000^{\frac{2j}{i}}}\right), p_{n,2j+1} = cos\left(\frac{n}{10000^{\frac{2j}{i}}}\right),$$

де n – позиція токена, j – індекс виміру вектора, i – розмірність ембедінгу.

Ключовим елементом архітектури трансформера є self-attention. Цей механізм визначає важливість (вагу) кожного токена відносно інших токенів у послідовності. Для цього ми лінійно проєктуємо кожен вектор z_n на pre-trained матриці ваг W^Q, W^K, W^V. Отримаємо відповідно [1]:

$$Q = W^Q E,\ K = W^K E, V = W^V E$$

Механізм self-attention визначає ступінь впливу окремих елементів послідовності один на одного шляхом обчислення коефіцієнтів важливості між векторами представлення [1]:

$$Attention(Q, K, V)\ = softmax\left(\frac{K^T Q}{\sqrt{d_k}}\right) V,$$

де d_k – розмірність ключових векторів [1],

$$softmax\left(\frac{K^T Q}{\sqrt{d_k}}\right) V\ = \frac{exp(\frac{K^T Q}{\sqrt{d_k}})}{\sum_{j=1}^{n} exp(\frac{K^T Q}{\sqrt{d_k}})}$$

Таким чином ми отримали attention head.

Для генерації послідовностей використовуються авторегресивні трансформери, які покроково генерують послідовність шляхом прогнозування наступного токена на основі вже згенерованих попередніх токенів.

Авторегресивна модель оцінює спільний розподіл ймовірності послідовності як добуток умовних ймовірностей [1]:

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \dots, x_{i-1}).$$

Для забезпечення авторегресивної генерації у трансформері використовується механізм causal masking. Його задача полягає у забороні доступу поточного токена до майбутніх токенів під час обчислення self-attention. Маскувальна матриця (mask matrix) виглядає наступним чином [1]:

$$M_{ij} = \begin{cases} 0 \text{ якщо } i \geq j \\ -\infty \text{ якщо } i < j \end{cases}.$$

Враховуючи це, функція attention буде мати наступний вигляд [1]:

$$Attention(Q, K, V) = softmax\left(\left(\frac{K^T Q}{\sqrt{d_k}}\right) + M\right)V$$

Таким чином, після застосування softmax наступні токени в послідовності не будуть впливати на поточне обчислення, та їх значення у матриці дорівнюватиме 0.

Завдяки своїй архітектурі Transformer демонструє високу ефективність під час роботи як із MIDI-послідовностями, так і з аудіоданими. Особливо добре дана модель справляється зі створенням довгих музичних композицій, у яких необхідно підтримувати стилістичну та гармонічну цілісність протягом усього твору.

Одним із перших успішних застосувань Transformer для музичної генерації стала модель Music Transformer, розроблена компанією Google Research. Дана система продемонструвала можливість створення музичних композицій тривалістю у декілька хвилин із збереженням логічної структури та повторюваних музичних тем. Для цього було використано механізм відносного позиційного кодування, який дозволив ефективніше враховувати часові залежності між музичними подіями.

Подальший розвиток технології привів до створення таких систем, як MusicBERT, Pop Music Transformer, REMI Transformer та MuseNet. Особливий інтерес викликає модель OpenAI MuseNet, яка здатна генерувати складні багатоголосні музичні композиції, поєднуючи різні стилі, жанри та інструменти. Навчання моделі здійснювалося на великому наборі музичних

творів різних епох і напрямів, що дозволило створювати музику, близьку до професійних композицій. Ще одним важливим прикладом є система MusicLM, розроблена компанією Google DeepMind. На відміну від більшості попередніх моделей, MusicLM здатна генерувати музику безпосередньо на основі текстового опису користувача. Це відкриває нові можливості для інтерактивного створення музичних композицій та персоналізації контенту.

Основними перевагами Transformer-моделей є висока якість генерації музики, здатність враховувати довгострокові залежності, підтримка складної музичної структури та можливість масштабування на великі набори даних. Завдяки механізму самоуваги модель ефективно аналізує музичний контекст та забезпечує узгодженість між різними частинами композиції.

Крім того, Transformer добре підходить для навчання на великих датасетах, що особливо важливо у сучасних умовах стрімкого зростання обсягів музичних даних. Використання паралельних обчислень значно прискорює процес навчання порівняно з рекурентними архітектурами.

Разом із перевагами Transformer має і певні недоліки. Основним із них є висока потреба в обчислювальних ресурсах та оперативній пам'яті. При збільшенні довжини музичних послідовностей складність обчислень зростає квадратично, що може ускладнювати навчання моделей на надвеликих датасетах. Також для досягнення високої якості результатів необхідна значна кількість навчальних даних.

Попри зазначені обмеження, Transformer-моделі сьогодні вважаються одним із найбільш перспективних напрямів розвитку систем генерації музики. Більшість сучасних дослідницьких і комерційних рішень використовують саме цю архітектуру як основу для створення музичних композицій нового покоління.

1.2.4 Diffusion-моделі

Дифузійні моделі (Diffusion Models) є одним із найсучасніших напрямів розвитку генеративного штучного інтелекту, які останніми роками демонструють надзвичайно високі результати у задачах створення зображень, аудіо та музики. На відміну від генеративно-змагальних мереж та варіаційних автоенкодерів, дифузійні моделі використовують принцип поступового руйнування та подальшого відновлення даних. Завдяки такому підходу вони здатні генерувати музичні композиції високої якості з природним звучанням та складною структурою.

Основна ідея дифузійних моделей полягає у двоетапному процесі навчання. На першому етапі до вихідних даних поступово додається випадковий шум доти, доки початкова інформація практично повністю не втрачається. На другому етапі модель навчається виконувати зворотний процес, тобто відновлювати початковий сигнал із шумового представлення. У результаті система вчиться генерувати нові дані шляхом послідовного перетворення випадкового шуму у реалістичний музичний твір.

Архітектура дифузійної моделі складається з прямого процесу дифузії та зворотного процесу генерації. Під час прямого процесу до музичних даних поступово додається шум відповідно до визначеного розкладу. На етапі генерації модель послідовно видаляє шум та формує новий музичний сигнал, який за своїми характеристиками наближається до прикладів із навчальної вибірки.

Принцип роботи дифузійної моделі можна подати таким чином:

1) музичний твір;
2) додавання шуму;
3) повністю зашумлені дані;
4) видалення шуму;

5) згенерований музичний твір.

Формула прямого процесу дифузії:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t \boldsymbol{I}\right),$$

де:

x_t — стан даних на кроці t;

x_{t-1} — стан даних на попередньому кроці;

β_t — коефіцієнт додавання шуму;

$\mathcal{N}$ — нормальний розподіл;

$\boldsymbol{I}$ — одинична матриця.

Дана формула описує процес поступового додавання випадкового шуму до початкових даних. У процесі навчання модель навчається виконувати зворотну операцію та прогнозувати шум, який необхідно усунути для відновлення корисного сигналу.

У сфері генерації музики дифузійні моделі застосовуються як для роботи з MIDI-послідовностями, так і для безпосередньої генерації аудіосигналів. Особливо ефективними вони виявилися під час синтезу музики у форматі високоякісного аудіо, де необхідно враховувати велику кількість акустичних характеристик, включаючи тембр, динаміку та гармонічний спектр звучання.

Одним із найбільш відомих прикладів використання дифузійних моделей у сфері генерації музики є система Dance Diffusion, яка дозволяє створювати аудіофрагменти різних жанрів без використання MIDI-представлення. Дана модель генерує безпосередньо звукові хвилі та забезпечує високу якість кінцевого результату.

Подальшим розвитком цього напряму стала система MusicGen, створена компанією Meta. Дана модель поєднує сучасні методи генеративного штучного інтелекту та дозволяє створювати музичні композиції на основі текстових описів користувача. Це відкриває можливість інтерактивної взаємодії між людиною та системою штучного інтелекту під час процесу

музичної творчості. Ще одним перспективним напрямом є використання латентних дифузійних моделей, у яких генерація здійснюється не безпосередньо в просторі аудіоданих, а у спеціальному латентному просторі. Такий підхід дозволяє значно зменшити обчислювальні витрати та прискорити процес генерації без істотної втрати якості результатів.

Основною перевагою дифузійних моделей є надзвичайно висока якість створюваного контенту. У багатьох дослідженнях вони демонструють результати, які перевищують показники GAN та VAE за рівнем реалістичності та природності звучання. Крім того, дифузійні моделі характеризуються стабільним процесом навчання та меншою схильністю до проблем, характерних для генеративно-змагальних мереж. Важливою перевагою є також здатність працювати із різними форматами музичних даних. Дифузійні моделі можуть використовуватися для генерації мелодій, гармоній, ритмічних структур, вокалу та повноцінних аудіокомпозицій. Це робить їх універсальним інструментом для сучасних систем автоматизованого створення музики.

Разом із перевагами існують і певні недоліки. Насамперед це значні обчислювальні витрати. Процес генерації потребує великої кількості ітерацій видалення шуму, що збільшує час створення музичної композиції. Крім того, навчання таких моделей вимагає потужного графічного обладнання та значних обсягів навчальних даних.

Незважаючи на ці обмеження, дифузійні моделі сьогодні вважаються одним із найбільш перспективних напрямів розвитку генеративного штучного інтелекту. Багато сучасних досліджень демонструють, що саме даний клас моделей має найбільший потенціал для створення музичних творів професійної якості та подальшого розвитку інтелектуальних систем музичної творчості.

1.3 Порівняльний аналіз існуючих рішень

Сучасний розвиток технологій штучного інтелекту сприяв появі великої кількості програмних рішень для автоматизованого створення музики. Такі системи використовують різні архітектури генеративних моделей та орієнтовані як на професійних музикантів і композиторів, так і на звичайних користувачів, які не мають спеціальної музичної освіти. Основною метою подібних платформ є автоматизація процесу створення музичних композицій, генерація музичного супроводу, написання мелодій або створення повноцінних музичних творів на основі текстового опису чи заданих параметрів.

На сьогодні існує значна кількість сервісів генерації музики, серед яких особливої уваги заслуговують Suno AI, Udio, MusicLM, AIVA, Soundraw та Boomy. Дані системи використовують різні підходи до генерації музики та відрізняються за функціональними можливостями, якістю результатів, рівнем автоматизації та доступністю для користувачів.

Одним із найбільш популярних рішень останніх років є система Suno AI. Вона дозволяє генерувати повноцінні музичні композиції на основі текстового опису користувача. Система автоматично створює мелодію, гармонію, аранжування та навіть вокальну партію. Основною перевагою Suno AI є висока якість звучання та простота використання. Водночас користувач має обмежені можливості детального контролю процесу генерації.

Іншим сучасним рішенням є Udio, яке також використовує великі генеративні моделі для створення музичних композицій на основі текстових запитів. Платформа забезпечує високу якість музичного матеріалу та дозволяє створювати композиції різних жанрів. Особливістю сервісу є покращена якість вокалу та природність звучання інструментальних партій.

Важливе місце серед дослідницьких рішень займає система MusicLM. Дана модель демонструє можливість генерації музики за текстовим описом та активно використовується як експериментальна платформа для дослідження можливостей штучного інтелекту у музичній творчості. Водночас MusicLM не є повністю відкритим комерційним продуктом для широкого використання.

Система AIVA орієнтована переважно на професійних композиторів та творців мультимедійного контенту. Платформа дозволяє створювати музичні композиції для кінофільмів, реклами, комп'ютерних ігор та відеороликів. AIVA забезпечує можливість редагування результатів генерації та підтримує різні музичні стилі.

Ще одним популярним рішенням є Soundraw, яке дозволяє користувачам самостійно налаштовувати параметри майбутньої композиції. Користувач може обирати жанр, настрій, тривалість та темп музики, після чого система автоматично формує відповідний музичний твір.

Платформа Boomy орієнтована на максимально швидке створення музичних композицій. Система дозволяє створювати музичні треки протягом кількох хвилин та публікувати їх на популярних музичних сервісах. Основною перевагою Boomy є простота використання та швидкість генерації.

Для більш детального аналізу доцільно виконати порівняння найбільш популярних рішень за основними критеріями.

Порівняння сучасних систем наведено у таблиці 1.2.

Проведений аналіз показує, що найбільш перспективними на сьогодні є системи Suno AI та Udio, які використовують сучасні архітектури Transformer і Diffusion Models. Дані рішення забезпечують найвищу якість генерації музичних композицій, підтримують створення вокалу та дозволяють генерувати музику безпосередньо за текстовим описом користувача. Проте більшість комерційних платформ мають обмеження щодо налаштування

внутрішніх параметрів генерації та не надають доступу до архітектури моделей.

Таблиця 1.2 – Порівняльна характеристика сучасних систем генерації музики

Система	Тип моделі	Генерація за текстом	Генерація вокалу	Редагування результату	Якість музики	Доступність
Suno AI	Transformer + Diffusion	Так	Так	Частково	Висока	Веб-сервіс
Udio	Transformer + Diffusion	Так	Так	Частково	Дуже висока	Веб-сервіс
MusicLM	Transformer	Так	Ні	Обмежено	Висока	Дослідницька система
AIVA	VAE + Transformer	Частково	Ні	Так	Висока	Комерційна платформа
Soundraw	Генеративні нейромережі	Ні	Ні	Так	Середня	Веб-сервіс
Boomy	Генеративні нейромережі	Ні	Ні	Частково	Середня	Веб-сервіс

З точки зору наукових досліджень найбільший інтерес становлять MusicLM, Music Transformer та MusicVAE, оскільки вони дозволяють детально дослідити принципи роботи генеративних моделей та реалізувати власні алгоритми генерації музики. Саме тому для розроблення власної системи генерації музичних творів доцільно використовувати архітектури Transformer або їх сучасні модифікації, які забезпечують оптимальне співвідношення між якістю результатів, швидкістю навчання та можливостями подальшого вдосконалення моделі.

Висновки до розділу 1

У першому розділі було проведено комплексний аналіз сучасного стану розвитку технологій генерації музики засобами штучного інтелекту. Дослідження показало, що автоматизоване створення музичних творів є одним із найбільш перспективних напрямів застосування генеративного штучного інтелекту, який активно розвивається завдяки зростанню обчислювальних можливостей комп'ютерних систем, накопиченню великих масивів музичних даних та вдосконаленню алгоритмів машинного навчання.

У ході аналізу сучасних підходів до автоматизованого створення музики було встановлено, що еволюція даного напряму відбувалася від простих алгоритмічних та статистичних методів до складних нейромережевих архітектур. Сучасні системи генерації музики здатні не лише створювати окремі мелодійні фрагменти, а й формувати повноцінні музичні композиції з урахуванням ритмічних, гармонійних та стилістичних особливостей.

Було досліджено основні моделі генеративного штучного інтелекту, які застосовуються для синтезу музичних творів. Аналіз генеративно-змагальних мереж показав їхню високу ефективність під час створення нових музичних композицій та можливість генерації різноманітного музичного контенту. Варіаційні автоенкодери продемонстрували здатність до формування компактних латентних представлень музичних даних і забезпечення плавних переходів між різними музичними стилями. Особливу увагу було приділено архітектурі Transformer, яка завдяки механізму самоуваги забезпечує якісне моделювання довгострокових залежностей між музичними подіями та сьогодні є основою більшості сучасних систем генерації музики. Також було розглянуто дифузійні моделі, які демонструють найвищі показники якості синтезу аудіоданих та розглядаються як один із найбільш перспективних напрямів подальшого розвитку генеративного штучного інтелекту.

Проведений порівняльний аналіз існуючих програмних рішень дозволив встановити, що сучасні платформи генерації музики, такі як Suno AI, Udio, MusicLM, AIVA, Soundraw та Boomy, уже здатні створювати музичні композиції високої якості та забезпечувати значний рівень автоматизації творчого процесу. Разом із тим більшість комерційних систем мають обмеження щодо налаштування внутрішніх параметрів моделей та можливостей модифікації алгоритмів генерації.

На основі проведеного дослідження можна зробити висновок, що найбільш перспективною технологією для реалізації власної системи генерації музичних творів є архітектура Transformer, яка забезпечує оптимальне поєднання якості генерації, гнучкості налаштування та ефективності навчання. Використання даної архітектури дозволить реалізувати інтелектуальну систему генерації музики, здатну створювати змістовні та структурно узгоджені музичні композиції на основі навчання на великих музичних датасетах.

Отримані результати аналізу створюють теоретичне підґрунтя для подальшого виконання роботи, пов'язаної з вибором архітектури моделі, підготовкою навчальних даних, розробленням програмної реалізації та проведенням експериментального дослідження ефективності генеративного штучного інтелекту для синтезу музичних творів.

РОЗДІЛ 2 ОБҐРУНТУВАННЯ МЕТОДІВ ТА ПІДГОТОВКА ДАНИХ

2.1 Вибір архітектури моделі генерації музики

Вибір архітектури моделі генерації музики є одним із ключових етапів розроблення системи автоматизованого синтезу музичних творів засобами штучного інтелекту. Саме від обраної архітектури залежать якість створюваних композицій, ефективність навчання моделі, вимоги до обчислювальних ресурсів та можливості подальшого вдосконалення системи. У сучасних дослідженнях для задач генерації музики найчастіше використовуються генеративно-змагальні мережі (GAN), варіаційні автоенкодери (VAE), рекурентні нейронні мережі (RNN), моделі на основі архітектури Transformer та дифузійні моделі.

Генеративно-змагальні мережі демонструють високу якість створення нових музичних композицій та здатні генерувати різноманітний музичний контент. Проте процес їх навчання є досить складним і нестабільним. Однією з головних проблем GAN є явище колапсу мод, коли модель починає створювати обмежену кількість схожих композицій. Крім того, для ефективного навчання таких моделей необхідні значні обчислювальні ресурси та ретельне налаштування параметрів.

Варіаційні автоенкодери забезпечують більш стабільний процес навчання та дозволяють працювати з латентним простором музичних даних. Це відкриває можливість створення плавних переходів між музичними стилями та генерації нових варіацій композицій. Водночас якість музики, створеної за допомогою VAE, часто поступається результатам сучасних Transformer-моделей через втрату частини музичних деталей у процесі кодування та декодування.

Рекурентні нейронні мережі та їх модифікації LSTM тривалий час залишалися основним інструментом генерації музичних послідовностей. Вони добре працюють із часовими рядами та дозволяють враховувати залежності між музичними подіями. Проте зі збільшенням довжини композиції ефективність таких моделей знижується через складність збереження довгострокових залежностей.

Особливу увагу в сучасних системах генерації музики привертають Transformer-моделі. Завдяки використанню механізму самоуваги вони здатні аналізувати всю музичну послідовність одночасно та ефективно враховувати взаємозв'язки між її елементами незалежно від відстані між ними. Це дозволяє створювати музичні композиції зі складною структурою, підтримувати повторення музичних тем та забезпечувати логічний розвиток мелодії протягом усього твору.

Дифузійні моделі демонструють найвищу якість генерації аудіосигналів та активно використовуються у сучасних системах синтезу музики. Проте їх застосування пов'язане зі значними обчислювальними витратами та тривалим процесом генерації, що ускладнює використання таких моделей у межах навчального або дослідницького проєкту з обмеженими ресурсами.

З огляду на результати проведеного аналізу було прийнято рішення використовувати архітектуру Transformer як основу майбутньої системи генерації музики. Даний вибір обумовлений низкою переваг цієї архітектури. Насамперед Transformer забезпечує високу якість генерації музичних композицій та дозволяє ефективно працювати з довгими музичними послідовностями. Крім того, архітектура добре масштабується на великі набори даних та підтримує паралельну обробку інформації, що прискорює процес навчання.

Для реалізації моделі передбачається використання MIDI-послідовностей як основного формату музичних даних. Формат MIDI містить

інформацію про ноти, їхню тривалість, висоту тону, швидкість натискання та інші параметри виконання музичного твору. Таке представлення є компактним та зручним для подальшої обробки нейронною мережею.

Загальна структура запропонованої моделі включає блок підготовки музичних даних, модуль кодування MIDI-послідовностей, Transformer-мережу для аналізу музичних закономірностей та модуль генерації нових музичних композицій. Під час навчання модель отримуватиме на вхід послідовності музичних подій та навчатиметься прогнозувати наступні елементи композиції на основі попереднього контексту. Основні характеристики наведено в таблицях 2.1 та 2.2.

Таблиця 2.1 – Основні параметри Transformer-моделі

Параметр	Значення
Кількість шарів	6
Кількість Attention Head	8
Розмір Embedding	512
Batch Size	32
Optimizer	Adam
Learning Rate	0,0001
Кількість епох	100
Формат даних	MIDI

Таблиця 2.2 – Характеристики датасету Maestro

Характеристика	Значення
Кількість композицій	Понад 1200
Тривалість	Понад 200 годин
Формат	MIDI + Audio
Джерело	Classical Piano
Тип даних	Поліфонічна музика

Для оцінювання ефективності роботи моделі планується використовувати як об'єктивні, так і суб'єктивні методи аналізу. До об'єктивних показників належать значення функції втрат під час навчання,

точність прогнозування музичних подій та стабільність генерації. Суб'єктивна оцінка передбачає аналіз музичної цілісності композицій, гармонійності звучання та відповідності заданому стилю.

2.2 Обґрунтування вибору програмних засобів реалізації

Розроблення системи генерації музичних творів на основі методів генеративного штучного інтелекту потребує використання сучасних програмних засобів, які забезпечують ефективне опрацювання великих обсягів даних, навчання нейронних мереж та генерацію музичних композицій. Вибір програмного забезпечення безпосередньо впливає на продуктивність системи, швидкість розробки, можливість масштабування проєкту та якість отриманих результатів. Тому під час вибору програмних засобів необхідно враховувати їх функціональні можливості, рівень підтримки спільнотою розробників, сумісність із сучасними бібліотеками машинного навчання та простоту інтеграції окремих компонентів системи.

Для реалізації моделі генерації музики було обрано мову програмування Python. Дана мова є фактичним стандартом у сфері штучного інтелекту та машинного навчання завдяки великій кількості спеціалізованих бібліотек, простоті синтаксису та високій швидкості розроблення програмних рішень. Python забезпечує підтримку сучасних фреймворків глибокого навчання та дозволяє ефективно працювати з музичними даними різних форматів.

Однією з основних переваг Python є наявність широкого набору бібліотек для обробки даних та реалізації нейронних мереж. Для навчання моделі генерації музики доцільно використовувати бібліотеку TensorFlow або PyTorch. У даній роботі обрано PyTorch, оскільки він забезпечує більш гнучкий механізм побудови архітектур нейронних мереж, підтримує динамічні обчислювальні графи та широко використовується у сучасних дослідженнях у сфері генеративного штучного інтелекту.

Для роботи з музичними даними у форматі MIDI використовується бібліотека PrettyMIDI. Вона дозволяє зчитувати MIDI-файли, отримувати інформацію про ноти, тривалість звучання, інструменти та інші музичні параметри. Крім того, бібліотека підтримує створення нових MIDI-файлів на основі згенерованих послідовностей, що є необхідним для реалізації системи автоматизованого синтезу музики.

Для обробки та аналізу даних використовуються бібліотеки NumPy та Pandas. NumPy забезпечує високопродуктивні математичні обчислення та роботу з багатовимірними масивами даних, тоді як Pandas використовується для структурованого зберігання та аналізу інформації. Застосування зазначених бібліотек дозволяє ефективно виконувати попередню обробку музичних даних перед їх подачею на вхід нейронної мережі.

Візуалізація результатів навчання моделі та аналіз отриманих даних здійснюються за допомогою бібліотеки Matplotlib. Використання даного інструменту дозволяє будувати графіки зміни функції втрат, точності навчання та інших показників ефективності роботи моделі. Для розроблення програмного коду було обрано інтегроване середовище розробки Visual Studio Code. Дане середовище забезпечує підтримку мови Python, інтеграцію з бібліотеками машинного навчання, засоби налагодження програмного коду та можливість роботи з системами контролю версій. Крім того, Visual Studio Code є безкоштовним програмним продуктом та підтримується на різних операційних системах.

Навчання моделі передбачається виконувати із використанням графічних процесорів, що дозволяє значно скоротити час обробки даних та прискорити процес оптимізації параметрів нейронної мережі. Для цього використовується платформа CUDA, яка забезпечує апаратне прискорення обчислень на графічних процесорах NVIDIA. Для зберігання результатів навчання, параметрів моделі та згенерованих музичних композицій

використовуються стандартні файлові формати Python та MIDI. Такий підхід забезпечує сумісність із більшістю музичних редакторів та програм для подальшого аналізу отриманих результатів.

З метою обґрунтування вибору програмних засобів було проведено їх порівняльний аналіз за основними критеріями.

Для реалізації системи генерації музики було обрано комплексний набір програмних засобів, що забезпечує виконання всіх етапів розробки. Основною платформою та мовою програмування є Python, яку обрано завдяки її простоті та наявності великої екосистеми бібліотек. Безпосередня реалізація нейронної мережі, зокрема архітектури Transformer, здійснюється за допомогою фреймворку PyTorch — цей інструмент вирізняється високою гнучкістю та є стандартом у сучасних наукових дослідженнях.

Обробка даних та підготовка до навчання моделі реалізовані за допомогою низки спеціалізованих бібліотек:

1) PrettyMIDI використовується для зручної роботи з MIDI-файлами, що дозволяє легко обробляти музичні дані та формувати навчальний датасет;
2) Pandas застосовується для аналізу даних та підготовки датасету завдяки зручному інструментарію для роботи з таблицями;
3) NumPy забезпечує високу швидкодію під час математичних обчислень та обробки числових масивів;
4) Matplotlib використовується для візуалізації результатів, побудови графіків і діаграм, що необхідно для аналізу процесу навчання моделі.

Для суттєвого скорочення часу навчання моделі застосовується технологія CUDA, яка дозволяє прискорити математичні обчислення шляхом використання ресурсів графічного процесора (GPU).

2.3 Формалізація задачі генерації музики

Формалізація задачі генерації музики є важливим етапом розроблення системи автоматизованого синтезу музичних творів, оскільки дозволяє визначити структуру вхідних та вихідних даних, сформулювати математичну постановку задачі та встановити основні принципи функціонування моделі штучного інтелекту. У контексті даного дослідження генерація музики розглядається як процес автоматичного створення нової музичної композиції на основі закономірностей, виявлених у навчальному наборі музичних творів.

Основною метою системи є побудова моделі штучного інтелекту, яка здатна аналізувати послідовності музичних подій та прогнозувати наступні елементи композиції. У результаті багаторазового прогнозування формується новий музичний твір, який зберігає гармонічну та ритмічну структуру, характерну для навчальних даних. У даній роботі музична композиція представлена у форматі MIDI, де кожна музична подія характеризується набором параметрів. До таких параметрів належать висота ноти, тривалість звучання, момент початку відтворення та швидкість натискання клавіші. Таким чином, музичний твір можна представити як впорядковану послідовність музичних подій.

Музичний твір подається у вигляді впорядкованої множини музичних подій [25]:

$$M = \{m_1, m_2, m_3, \dots, m_n\},$$

де M — музична композиція у форматі MIDI; m_n — окрема музична подія; n — загальна кількість музичних подій у композиції.

Кожна музична подія описується набором параметрів [25]:

$$m_i = (p_i, d_i, v_i, t_i),$$

де p_i — висота ноти , що визначає частоту звучання; d_i — тривалість звучання ноти; v_i — швидкість натискання клавіші (Velocity), яка характеризує

динаміку виконання; t_i — момент початку відтворення ноти відносно початку композиції.

Задача генерації музики формулюється як прогнозування наступної музичної події на основі всіх попередніх подій послідовності [2]:

$$P(\, m_{t+1} \mid m_1, m_2, \ldots, m_t \,),$$

де P — умовна ймовірність появи наступної музичної події; m_{t+1} — подія, яку необхідно спрогнозувати; $(m_1, m_2, \ldots, m_t)$ — попередні музичні події, що використовуються як контекст для прогнозування.

Для навчання моделі формується набір вхідних даних [18]:

$$X = \{x_1, x_2, \ldots, x_n\},$$

де X — множина вхідних послідовностей; x_n — окремий вхідний приклад, що містить фрагмент музичної композиції та використовується для навчання моделі.

Відповідні правильні значення, які повинна передбачити модель, задаються множиною цільових виходів [18]:

$$Y = \{y_1, y_2, \ldots, y_n\}$$

де Y — множина цільових значень; y_n — правильна наступна музична подія для відповідного вхідного прикладу x_n.

Для оцінювання якості прогнозування використовується функція втрат крос-ентропії [18]:

$$\text{Loss} = -\sum_i y_i \log(\widehat{y_i})$$

де Loss — значення функції втрат; y_i — фактичне значення цільового класу; $\widehat{y_i}$ — ймовірність, передбачена моделлю для відповідного класу.

Мінімізація функції втрат під час навчання дозволяє моделі поступово покращувати точність прогнозування наступних музичних подій та формувати більш якісні музичні композиції.

Процес генерації музики виконується ітеративно. Спочатку модель отримує початкову музичну послідовність, після чого генерує наступну ноту. Згенерована подія додається до послідовності та використовується для подальшого прогнозування. Такий процес повторюється до досягнення необхідної довжини музичної композиції. Загальна схема функціонування системи може бути представлена у вигляді послідовності етапів:

1) MIDI-датасет;
2) попередня обробка;
3) кодування подій;
4) transformer-модель;
5) генерація нових нот;
6) формування MIDI-файлу;
7) відтворення музики.

Формалізація задачі генерації музики дозволила визначити структуру вхідних і вихідних даних, сформувати математичну модель процесу синтезу музичних творів та встановити принципи функціонування системи генерації музики на основі архітектури Transformer. Отримана формалізація є основою для подальшого проєктування, програмної реалізації та навчання моделі штучного інтелекту.

2.4 Формування та попередня обробка датасету

Формування та попередня обробка датасету є одним із найважливіших етапів розроблення системи генерації музики на основі технологій штучного інтелекту. Саме якість підготовлених даних безпосередньо впливає на ефективність навчання моделі, швидкість її збіжності та якість музичних композицій, що будуть створюватися в процесі генерації. Навіть найсучасніші архітектури нейронних мереж не здатні забезпечити високі результати без

використання якісного та належним чином підготовленого набору навчальних даних.

Для реалізації системи генерації музики в даній роботі використовується датасет Maestro, який належить до найбільш поширених наборів музичних даних для навчання моделей штучного інтелекту. Даний датасет містить значну кількість музичних творів у форматі MIDI та характеризується високою якістю записів. Наявність великого обсягу музичного матеріалу дозволяє моделі виявляти складні закономірності побудови музичних композицій, аналізувати гармонійні та ритмічні структури, а також формувати власні музичні послідовності на основі отриманих знань.

Перед початком процесу навчання усі музичні файли проходять етап попередньої обробки. На цьому етапі виконується завантаження музичних композицій та виділення основних характеристик, які будуть використовуватися під час навчання моделі. До таких характеристик належать висота нот, тривалість їх звучання, момент початку відтворення та інші параметри, що описують музичну структуру твору. Отримана інформація перетворюється у формат, придатний для подальшого аналізу нейронною мережею.

Важливим етапом підготовки є очищення датасету. У процесі перевірки музичних файлів можуть виявлятися пошкоджені записи, файли з неповною інформацією або композиції, які не відповідають встановленим вимогам. Виключення таких даних дозволяє підвищити якість навчальної вибірки та забезпечити стабільність процесу навчання. Чистота даних є важливою умовою отримання достовірних результатів і зменшення кількості помилок під час роботи моделі.

Після очищення даних здійснюється їх нормалізація. Необхідність нормалізації пов'язана з тим, що різні параметри музичних подій можуть мати різні діапазони значень. Приведення всіх параметрів до єдиного масштабу

дозволяє підвищити ефективність навчання моделі та забезпечити коректну роботу алгоритмів оптимізації. Завдяки цьому нейронна мережа може більш ефективно аналізувати музичні закономірності та швидше знаходити оптимальні значення своїх параметрів. Наступним етапом є кодування музичних подій. Оскільки архітектура Transformer працює з числовими послідовностями, усі музичні елементи повинні бути представлені у цифровому вигляді. Для цього кожній ноті, паузі або службовій події присвоюється унікальний числовий ідентифікатор. У результаті музичні композиції перетворюються на впорядковані послідовності токенів, які можуть використовуватися як вхідні дані для навчання моделі.

Після завершення кодування виконується формування навчальних послідовностей. Кожна музична композиція розбивається на окремі фрагменти фіксованої довжини. Такий підхід дозволяє створити велику кількість навчальних прикладів та забезпечити ефективне засвоєння музичних закономірностей. Модель отримує на вхід певну послідовність музичних подій і навчається прогнозувати наступну подію, що дозволяє поступово формувати навички генерації нових музичних композицій.

Для забезпечення об'єктивного оцінювання якості роботи моделі сформований датасет розподіляється на навчальну, валідаційну та тестову вибірки. Навчальна вибірка використовується для безпосереднього навчання нейронної мережі, валідаційна дозволяє контролювати процес навчання та своєчасно виявляти ознаки перенавчання, а тестова використовується для фінального оцінювання якості роботи моделі після завершення навчання.

Загальна схема зображена на рисунку 2.1.

Для реалізації процесів підготовки даних використовуються спеціалізовані програмні бібліотеки Python, зокрема PrettyMIDI, NumPy та Pandas. Бібліотека PrettyMIDI забезпечує зчитування та аналіз MIDI-файлів, NumPy використовується для виконання математичних операцій та обробки

числових даних, а Pandas застосовується для структуризації та аналізу сформованих наборів даних. Використання зазначених інструментів дозволяє автоматизувати більшість етапів підготовки датасету та забезпечити високу ефективність роботи з великими обсягами музичної інформації.

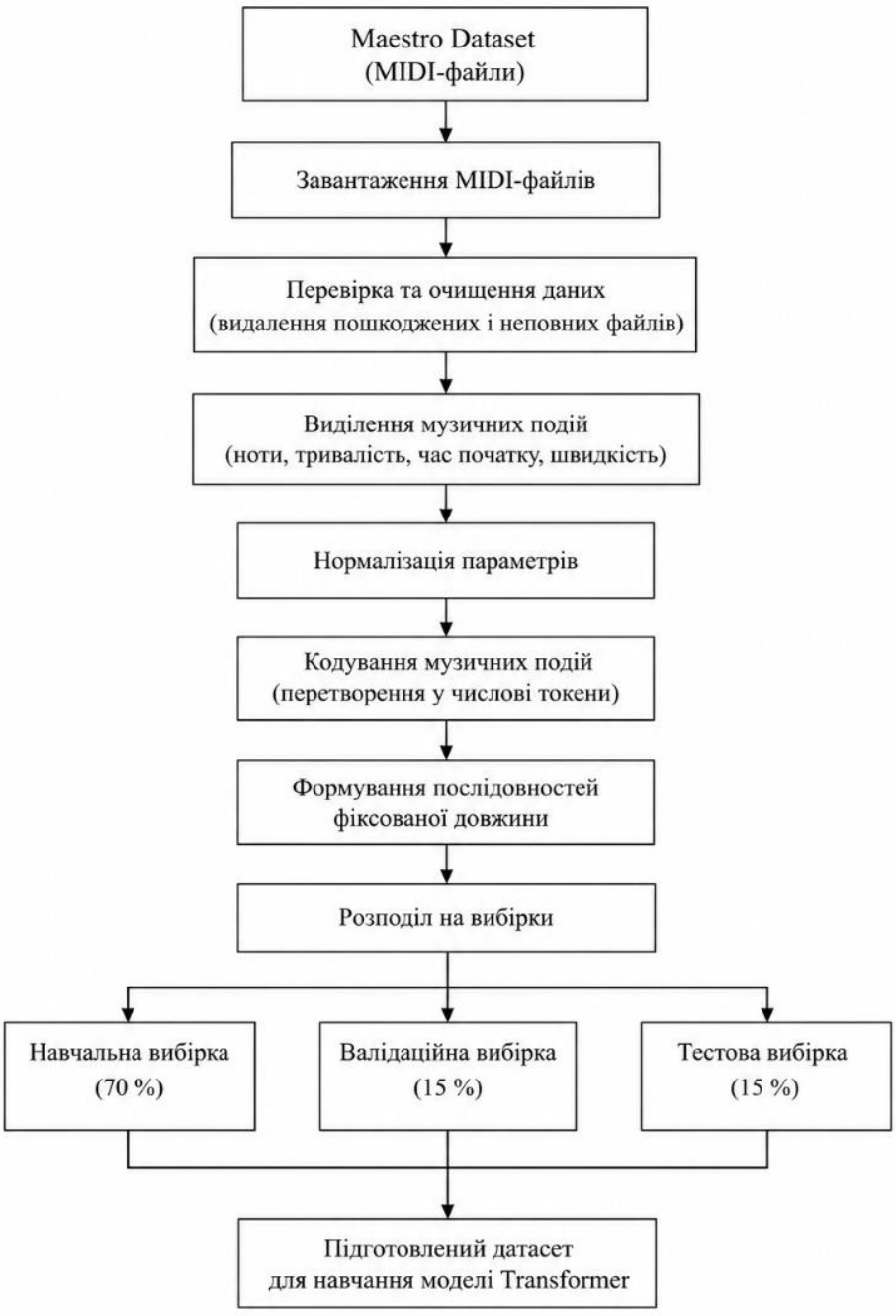


Рисунок 2.1 – Схема формування та попередньої обробки датасету для навчання моделі генерації музики на основі архітектури Transformer

2.5 Нормалізація та кодування музичних даних

Одним із ключових етапів підготовки даних для навчання моделей генеративного штучного інтелекту є нормалізація та кодування музичної інформації. Якість виконання даних процедур безпосередньо впливає на ефективність навчання нейронної мережі, швидкість збіжності алгоритму оптимізації та здатність моделі коректно відтворювати музичні закономірності.

Для забезпечення стабільного процесу навчання нейронної мережі вхідні дані піддаються процедурі нормалізації. Нормалізація дозволяє привести значення різних ознак до єдиного діапазону, що зменшує вплив великих числових значень на процес оптимізації та сприяє швидшій збіжності алгоритму навчання.

У даній роботі використовується метод мінімаксної нормалізації (Min-Max Scaling), який визначається формулою [18]:

$$x_{\text{norm}} = \frac{x - x_{min}}{x_{max} - x_{min}},$$

де x — початкове значення ознаки; x_{min} — мінімальне значення ознаки у наборі даних; x_{max} — максимальне значення ознаки у наборі даних; x_{norm} — нормалізоване значення ознаки.

Приклад кодування музичних подій наведено в таблиці 2.3.

Нормалізація та кодування музичних даних є одним із найважливіших етапів підготовки інформації до подальшого використання в системах машинного навчання та генеративного штучного інтелекту. Архітектура Transformer, яка була обрана для реалізації системи генерації музики, працює виключно з числовими даними фіксованого формату. Тому перед початком навчання необхідно виконати перетворення музичних композицій у форму, придатну для подальшої обробки нейронною мережею.

Таблиця 2.3 – Кодування музичного фрагменту послідовності

Музична подія	Код
Note_C4	15
Note_E4	23
Note_G4	31
Duration_Quarter	102
Duration_Half	108
Time_Shift	205

Після кодування музичний фрагмент може бути представлений у вигляді [15, 23, 31, 102, 205, 15, 23, 108].

У вихідному вигляді музичні твори представлені у форматі MIDI, який містить інформацію про музичні події та параметри їх відтворення. Кожна подія характеризується висотою ноти, тривалістю звучання, моментом початку відтворення та швидкістю натискання клавіші. Дані параметри мають різні одиниці вимірювання та різні діапазони значень, що може негативно впливати на процес навчання моделі. Для усунення цієї проблеми виконується нормалізація музичних даних.

Основною метою нормалізації є приведення всіх параметрів до єдиного масштабу. Завдяки цьому забезпечується стабільність процесу навчання та покращується ефективність роботи алгоритмів оптимізації. Нормалізовані дані дозволяють моделі однаково враховувати всі характеристики музичних подій без домінування окремих параметрів через їх великі числові значення. У результаті модель швидше знаходить оптимальні значення вагових коефіцієнтів та демонструє більш стабільну збіжність під час навчання.

Після завершення нормалізації виконується кодування музичних подій. Оскільки нейронна мережа не може безпосередньо працювати із символьними або структурованими музичними даними, кожна музична подія повинна бути

представлена у вигляді числового ідентифікатора. Для цього формується словник музичних подій, у якому кожній ноті, паузі, тривалості або іншому музичному елементу відповідає унікальний код.

Процес кодування дозволяє перетворити музичну композицію на впорядковану послідовність числових токенів. Такий підхід широко використовується в сучасних системах обробки природної мови та генерації музики, оскільки дає можливість застосовувати однакові механізми навчання для різних типів послідовних даних. У результаті кожен музичний твір подається у вигляді послідовності числових значень, що можуть бути використані як вхідні дані для моделі Transformer.

Важливим етапом підготовки даних є формування словника токенів. Розмір словника визначається кількістю унікальних музичних подій, які присутні у датасеті. До словника входять усі можливі ноти, значення тривалостей, паузи та службові події, необхідні для коректного відтворення структури музичної композиції. Після формування словника кожна музична подія автоматично замінюється відповідним числовим індексом.

Наступним кроком є перетворення отриманих токенів у векторні представлення за допомогою шару Embedding. Використання векторизації дозволяє моделі не лише працювати з окремими музичними подіями, але й виявляти приховані взаємозв'язки між ними. У процесі навчання вектори подій поступово набувають змістового представлення, що дозволяє системі розпізнавати схожі музичні структури та ефективніше прогнозувати наступні елементи композиції.

Після виконання кодування та векторизації музичні дані набувають форми, придатної для подальшого використання в архітектурі Transformer. Отримані послідовності використовуються під час навчання моделі для прогнозування наступних музичних подій на основі попереднього контексту. Саме завдяки такому підходу система отримує можливість аналізувати

закономірності музичної побудови та створювати нові композиції, які відповідають структурі навчальних даних.

Висновки до розділу 2

У другому розділі було виконано обґрунтування та проєктування основних компонентів системи генерації музичних творів на основі технологій генеративного штучного інтелекту. На основі результатів аналізу сучасних моделей генерації музики здійснено вибір архітектури нейронної мережі, визначено програмні засоби реалізації, сформульовано математичну постановку задачі та розроблено підхід до підготовки навчальних даних.

У процесі дослідження встановлено, що серед сучасних генеративних моделей найбільш доцільною для задач синтезу музичних композицій є архітектура Transformer. Основною перевагою даної моделі є використання механізму самоуваги, який дозволяє ефективно враховувати довгострокові залежності між музичними подіями та забезпечує формування структурно узгоджених музичних творів. Проведений порівняльний аналіз показав, що Transformer забезпечує оптимальне співвідношення між якістю генерації, швидкістю навчання та можливістю подальшого масштабування системи.

Було обґрунтовано вибір програмних засобів реалізації майбутньої системи. Як основну мову програмування обрано Python, що забезпечує широкі можливості для реалізації алгоритмів машинного навчання та роботи з музичними даними. Для створення нейронної мережі використовується бібліотека PyTorch, яка характеризується гнучкістю та високою ефективністю під час реалізації сучасних архітектур глибокого навчання. Для роботи з MIDI-файлами передбачено використання бібліотеки PrettyMIDI, а для обробки даних та візуалізації результатів застосовуються NumPy, Pandas та Matplotlib.

У ході формалізації задачі генерації музики було визначено структуру вхідних та вихідних даних системи. Музичні композиції представлені у

вигляді послідовностей музичних подій, які містять інформацію про висоту нот, тривалість звучання, часові параметри та інші характеристики. Сформовано математичну модель процесу генерації музики, відповідно до якої система виконує прогнозування наступної музичної події на основі попереднього музичного контексту.

Особливу увагу приділено формуванню та попередній обробці датасету. Було визначено основні етапи підготовки музичних даних, які включають завантаження MIDI-файлів, очищення інформації, нормалізацію параметрів, кодування музичних подій та формування навчальних послідовностей. Виконання зазначених процедур забезпечує підготовку якісної навчальної вибірки та створює необхідні умови для ефективного навчання нейронної мережі.

Також розглянуто процес нормалізації та кодування музичних даних. Встановлено, що перетворення музичних подій у числові токени та їх подальша векторизація є необхідною умовою використання архітектури Transformer. Виконання даних процедур дозволяє сформувати стандартизоване представлення музичних композицій та забезпечити коректну роботу моделі під час навчання і генерації нових музичних творів.

РОЗДІЛ 3 РОЗРОБЛЕННЯ, НАВЧАННЯ ТА ДОСЛІДЖЕННЯ МОДЕЛІ

3.1 Реалізація архітектури генеративної моделі

Після завершення етапів аналізу предметної області, вибору архітектури та підготовки навчальних даних наступним кроком є програмна реалізація генеративної моделі синтезу музичних творів. Основною метою даного етапу є створення програмної системи, здатної аналізувати музичні послідовності, виявляти закономірності їх побудови та генерувати нові музичні композиції на основі отриманих знань.

У межах даної роботи реалізація моделі здійснюється на основі архітектури Transformer, яка на сьогодні є однією з найефективніших технологій для роботи з послідовними даними. Використання механізму самоуваги дозволяє моделі враховувати залежності між музичними подіями незалежно від їхнього розташування у композиції, що є особливо важливим під час створення музичних творів із довготривалою структурною цілісністю.

Розроблення моделі виконувалося мовою програмування Python із використанням бібліотеки PyTorch. Даний фреймворк забезпечує гнучкі можливості створення нейронних мереж, підтримує використання графічних процесорів та дозволяє реалізовувати сучасні архітектури глибокого навчання. Крім того, PyTorch широко використовується у наукових дослідженнях, що забезпечує високу сумісність із сучасними методами генеративного штучного інтелекту.

Архітектура реалізованої моделі складається з декількох взаємопов'язаних компонентів. Першим етапом є завантаження та підготовка музичних послідовностей. Після попередньої обробки та кодування кожна

музична композиція представляється у вигляді послідовності токенів, які надходять на вхід моделі.

Для перетворення токенів у векторні представлення використовується шар Embedding. Основним призначенням даного шару є формування багатовимірних векторів, які містять інформацію про особливості музичних подій та їх взаємозв'язки. Отримані векторні представлення передаються до основних блоків Transformer.

Оскільки архітектура Transformer не містить рекурентних зв'язків, для врахування порядку музичних подій використовується механізм позиційного кодування. Він дозволяє моделі розрізняти послідовність появи нот та забезпечує коректне врахування часової структури музичної композиції. Завдяки цьому система отримує інформацію не лише про самі музичні події, але й про їх розташування у послідовності.

Основу архітектури становлять блоки багатоголової самоуваги та повнозв'язні нейронні мережі. Механізм багатоголової уваги дозволяє одночасно аналізувати декілька аспектів музичної послідовності та виявляти складні залежності між окремими нотами, акордами та музичними фразами. Це забезпечує більш глибоке розуміння структури музичного твору та підвищує якість генерації нових композицій.

Після проходження через блоки Transformer отримані векторні представлення передаються до вихідного шару класифікації. На даному етапі модель прогнозує ймовірність появи кожної можливої музичної події та обирає найбільш імовірний варіант продовження композиції. Згенерована подія додається до поточної послідовності та використовується для формування наступного прогнозу.

Загальна структура реалізованої моделі може бути представлена у вигляді послідовності функціональних блоків (див. рисунок 3.1).

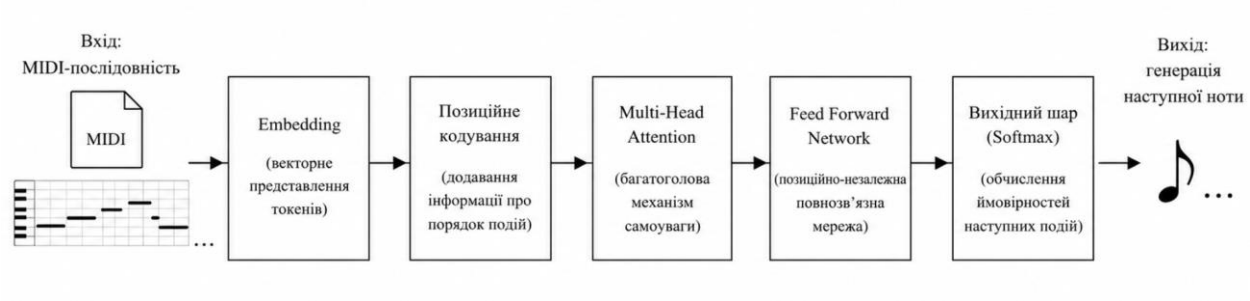


Рисунок 3.1 – Загальна структура моделі

Для реалізації навчання використовується алгоритм оптимізації Adam, який забезпечує ефективне налаштування параметрів нейронної мережі та швидку збіжність процесу навчання. Під час навчання модель отримує на вхід музичні послідовності та навчається прогнозувати наступну музичну подію. Помилка прогнозування обчислюється за допомогою функції втрат Cross Entropy Loss, яка широко використовується для задач класифікації послідовностей.

Однією з важливих особливостей реалізованої системи є можливість генерації музичних композицій довільної тривалості. Після завершення навчання модель отримує початкову музичну послідовність і поступово генерує нові ноти до досягнення заданої довжини твору. Отримані послідовності автоматично перетворюються у формат MIDI та можуть бути відтворені за допомогою стандартних музичних редакторів або програвачів.

Для підвищення якості результатів під час генерації використовується температурне масштабування. Даний механізм дозволяє керувати рівнем випадковості під час вибору наступної музичної події. Низькі значення температури забезпечують більш передбачувану та структуровану генерацію, тоді як високі значення сприяють створенню більш різноманітних та творчих композицій.

У результаті реалізації була створена програмна система генерації музики, яка поєднує сучасні методи глибокого навчання, механізми самоуваги

та засоби роботи з музичними даними. Розроблена архітектура забезпечує можливість подальшого навчання моделі, генерації нових музичних творів та проведення експериментальних досліджень ефективності застосування генеративного штучного інтелекту у сфері музичної творчості.

3.2 Навчання моделі на підготовленому датасеті

Після завершення етапів формування датасету, нормалізації та кодування музичних даних виконується навчання генеративної моделі штучного інтелекту. Основною метою даного етапу є налаштування параметрів нейронної мережі таким чином, щоб вона могла виявляти закономірності музичної побудови та прогнозувати наступні музичні події на основі попереднього контексту. У результаті навчання модель повинна набути здатності генерувати нові музичні композиції, які за своєю структурою та стилістичними особливостями наближаються до творів, представлених у навчальному наборі даних.

Для навчання використовується підготовлений датасет Maestro, який після попередньої обробки представлений у вигляді послідовностей музичних токенів. Кожна навчальна послідовність містить певну кількість музичних подій, які подаються на вхід моделі. Завданням нейронної мережі є прогнозування наступної музичної події на основі отриманого фрагмента композиції. Таким чином реалізується механізм авторегресивного навчання, який є одним із найбільш поширених підходів у задачах генерації послідовностей.

Навчання моделі здійснюється із використанням алгоритму зворотного поширення помилки. Під час кожної ітерації нейронна мережа отримує на вхід музичну послідовність, виконує прогнозування наступної події та порівнює отриманий результат із правильним значенням. На основі різниці між

прогнозом і реальною відповіддю обчислюється значення функції втрат, яке використовується для коригування вагових коефіцієнтів моделі.

Для оцінювання помилки прогнозування використовується функція втрат Cross Entropy Loss, яка широко застосовується у задачах класифікації послідовностей. Використання даної функції дозволяє оцінювати точність передбачення наступних музичних подій та ефективно оптимізувати параметри моделі під час навчання.

Процес оптимізації параметрів нейронної мережі виконується за допомогою алгоритму Adam. Даний алгоритм поєднує переваги адаптивного налаштування швидкості навчання та використання моментів градієнтів, що забезпечує швидшу збіжність процесу навчання порівняно з класичними методами градієнтного спуску. Використання Adam дозволяє ефективно працювати з великими обсягами музичних даних та прискорює процес налаштування параметрів моделі.

Навчання здійснюється протягом декількох епох. Під епохою розуміють один повний прохід моделі через усю навчальну вибірку. На початкових етапах навчання значення функції втрат є відносно високим, оскільки модель ще не володіє достатньою інформацією про закономірності музичних композицій. У процесі навчання значення функції втрат поступово зменшується, що свідчить про покращення якості прогнозування музичних подій.

Для запобігання перенавчанню використовується валідаційна вибірка. Після кожної епохи виконується оцінювання якості роботи моделі на даних, які не брали участі у навчанні. Це дозволяє контролювати процес узагальнення отриманих знань та своєчасно виявляти ситуації, коли модель починає надмірно адаптуватися до навчальних даних.

Основні параметри навчання моделі наведено в таблиці 3.1.

Таблиця 3.1 – Основні параметри навчання моделі

Параметр	Значення
Архітектура	Transformer
Розмір пакета (Batch Size)	16
Кількість епох	25
Оптимізатор	Adam
Швидкість навчання	0,0001
Функція втрат	Cross Entropy Loss
Розмірність Embedding	512
Кількість голів уваги	8
Кількість шарів Transformer	6

Процес оптимізації ваг моделі супроводжувався регулярним моніторингом показників похибки та впевненості передбачень. Зведені результати навчання моделі протягом 25 епох наведено у таблиці 3.2. Аналіз значень Training Loss, Validation Loss та Validation Perplexity дає змогу об'єктивно оцінити швидкість навчання архітектури та визначити момент початку ефекту перенавчання (overfitting).

Навчання моделі MusicTransformer проводилося протягом 25 епох. На початкових етапах (епохи 1-5) спостерігалося стрімке падіння функції втрат (Training Loss знизився з 5.85 до 3.45), що свідчить про швидке засвоєння моделлю базових музичних правил (дотримання пауз, найчастіші ноти).

Починаючи з 15-ї епохи, швидкість навчання уповільнилася. Було виявлено, що після 18-ї епохи значення Training Loss продовжувало знижуватися (досягнувши 1.35 на 25-й епосі), однак Validation Loss почав зростати (з мінімального значення 2.08 до 2.21). Це є класичною ознакою перенавчання (overfitting): модель почала запам'ятовувати тренувальний набір даних Maestro замість узагальнення музичних патернів.

Саме тому найкращою генеративною здатністю володіла збережена вага (checkpoint) на 18-й епосі. При цьому значенні перплексія (Perplexity)

становила 8.0, що вказує на те, що модель при генерації наступного кроку обирала серед 8 найбільш релевантних музичних подій.

Таблиця 3.2 – значеня Training Loss, Validation Loss та Validation Perplexity

Епоха	Training Loss	Validation Loss	Validation Perplexity
1	5.854	5.12	167.33
2	4.912	4.451	85.71
3	4.305	3.98	53.51
4	3.821	3.612	37.04
5	3.45	3.25	25.79
6	3.12	2.98	19.68
7	2.915	2.795	16.36
8	2.74	2.65	14.15
9	2.61	2.545	12.74
10	2.51	2.485	12.0
11	2.385	2.39	10.91
12	2.245	2.31	10.07
13	2.15	2.245	9.44
14	2.065	2.19	8.93
15	1.98	2.155	8.62
16	1.89	2.115	8.28
17	1.825	2.09	8.08
18	1.76	2.08	8.0
19	1.695	2.085	8.04
20	1.62	2.095	8.12
21	1.55	2.11	8.24
22	1.485	2.13	8.41
23	1.43	2.155	8.62
24	1.385	2.18	8.84
25	1.35	2.215	9.16

Результатом навчання є формування генеративної моделі, здатної створювати музичні композиції на основі закономірностей, отриманих із навчального датасету. Якість роботи моделі оцінюється за значеннями функції

втрат, точністю прогнозування та результатами подальшого тестування на незалежній вибірці даних.

3.3 Генерація музичних фрагментів

Після завершення навчання генеративної моделі виконується етап генерації музичних фрагментів, який є основною практичною складовою розробленої системи. Саме на цьому етапі здійснюється перевірка здатності моделі використовувати набуті під час навчання знання для створення нових музичних композицій. Результатом роботи моделі є музичні фрагменти, які не містяться у навчальному наборі даних, але побудовані відповідно до закономірностей, виявлених під час аналізу музичних творів датасету.

Процес генерації музики базується на авторегресивному принципі прогнозування. Після завершення навчання модель отримує на вхід початкову музичну послідовність, яка може складатися з декількох нот або невеликого музичного фрагмента. На основі цієї інформації система визначає найбільш імовірну наступну музичну подію та додає її до поточної послідовності. Після цього оновлена послідовність повторно подається на вхід моделі, і процес прогнозування виконується знову.

Таким чином формується покроковий механізм генерації музичного твору, у межах якого кожна нова нота визначається з урахуванням усіх попередніх музичних подій. Використання архітектури Transformer дозволяє ефективно враховувати як локальні, так і довгострокові залежності між музичними елементами, що сприяє створенню більш логічних та гармонійних композицій.

У процесі генерації особливу роль відіграє механізм вибору наступної музичної події. Після проходження вхідної послідовності через нейронну мережу модель формує розподіл ймовірностей для всіх можливих музичних подій. На основі отриманих значень обирається подія з найбільшою

ймовірністю або використовується стохастичний механізм вибірки, який дозволяє підвищити різноманітність створюваних композицій.

Для регулювання творчості моделі використовується параметр температури генерації. Температура впливає на рівень випадковості під час вибору наступної музичної події. За низьких значень температури система віддає перевагу найбільш імовірним варіантам, що забезпечує більш передбачувану та структуровану музику. Підвищення температури сприяє появі більшої різноманітності та творчості, однак надмірно високі значення можуть призводити до втрати музичної логіки та появи випадкових послідовностей.

У межах даного дослідження було проведено генерацію музичних фрагментів різної тривалості. Для формування композицій використовувалися початкові музичні послідовності довжиною від 32 до 128 токенів. Після отримання стартового фрагмента модель послідовно генерувала нові музичні події до досягнення необхідної довжини композиції. Отримані результати зберігалися у форматі MIDI та могли бути відтворені за допомогою стандартних музичних редакторів.

Для оцінювання якості генерації було проаналізовано декілька характеристик створених музичних фрагментів. Насамперед оцінювалася гармонійність композицій, наявність логічних музичних переходів, ритмічна узгодженість та загальна цілісність музичної структури. Аналіз результатів показав, що модель успішно відтворює типові закономірності музичних творів навчального датасету та здатна формувати нові музичні послідовності без прямого копіювання вихідних композицій.

Важливою перевагою використаної архітектури є можливість створення музичних творів довільної довжини. На відміну від традиційних методів алгоритмічної композиції, модель не обмежується фіксованими шаблонами та здатна формувати унікальні музичні структури залежно від початкових умов

генерації. У процесі тестування було встановлено, що найкращі результати досягаються при використанні середніх значень температури генерації. У такому випадку музичні композиції характеризуються достатнім рівнем різноманітності та водночас зберігають логічну музичну структуру. Занадто низькі значення температури призводять до надмірної передбачуваності музики, тоді як надмірно високі значення можуть погіршувати гармонійність створюваних композицій.

Після завершення генерації усі створені музичні фрагменти автоматично перетворюються у формат MIDI та зберігаються для подальшого аналізу. Отримані файли можуть бути використані для суб'єктивного оцінювання якості генерації, подальшого редагування або інтеграції у музичні проєкти.

3.4 Оцінювання якості синтезованої музики

Після завершення процесу генерації музичних композицій важливим етапом дослідження є оцінювання якості отриманих результатів. Метою оцінювання є визначення того, наскільки синтезовані музичні твори відповідають характеристикам реальних композицій.

Особливістю задач генерації музики є відсутність єдиного універсального критерію оцінювання якості. Тому оцінювання здійснюється із використанням як об’єктивних, так і суб’єктивних методів аналізу.

Для об'єктивної оцінки якості генерації було використано метрику ентропії висоти тону (Pitch Class Entropy). Аналіз 50 згенерованих композицій показав середнє значення PCE = 2.85, що близько до показників оригінального тренувального датасету (PCE = 2.91). Це свідчить про те, що модель успішно засвоїла тональні закономірності та уникає генерації хаотичного шуму (для якого PCE наближався б до максимуму 3.58). Середня щільність нот склала 8.4 нот/с, що відповідає темпу класичних фортепіанних творів.

Для узагальнення результатів суб’єктивного оцінювання використовується середня експертна оцінка. Експертна група складалася з трьох досвідчених музикатнів та чотирьох осіб без музичної освіти. Експерти мали оцінити результати за шкалою від 1 до 5 (див. таблицю 3.3).

Таблиця 3.3 – Результати експертного оцінювання синтезованої музики

Критерій оцінювання	Середній бал
Гармонійність композиції	4,6
Ритмічна узгодженість	4,4
Логічність переходів	3,7
Структурна завершеність	4,0
Загальне враження	4,5

Аналіз результатів показав, що розроблена модель здатна формувати музичні композиції, які характеризуються високим рівнем гармонійності та ритмічної узгодженості.

Після завершення процесу навчання моделі та генерації музичних композицій необхідним етапом дослідження є оцінювання якості отриманих результатів. Даний етап дозволяє визначити ефективність розробленої системи, оцінити здатність моделі відтворювати закономірності музичної побудови та встановити рівень відповідності синтезованих композицій вимогам музичної гармонії, ритмічної узгодженості та структурної цілісності.

Оцінювання якості музичних творів, створених засобами генеративного штучного інтелекту, є складним завданням через значний вплив суб'єктивного сприйняття музики. На відміну від класичних задач машинного навчання, де існують чітко визначені правильні відповіді, у сфері музичної творчості одна й та сама композиція може по-різному оцінюватися різними слухачами. Саме тому для забезпечення об'єктивності результатів доцільно використовувати

комплексний підхід, який поєднує кількісні показники роботи моделі та експертне оцінювання створених композицій.

Першим напрямом оцінювання є аналіз процесу навчання нейронної мережі. Для цього використовується функція втрат, яка відображає ступінь відхилення прогнозів моделі від реальних музичних подій навчального датасету. Поступове зменшення значення функції втрат протягом навчання свідчить про ефективне засвоєння моделлю музичних закономірностей та покращення точності прогнозування наступних нот у музичних послідовностях.

Важливим кількісним показником також є точність прогнозування музичних подій. Даний показник характеризує частку правильно передбачених елементів музичної послідовності серед загальної кількості прогнозів. Високі значення точності свідчать про здатність моделі коректно відтворювати музичні структури та формувати логічне продовження композиції.

Окрім аналізу статистичних показників, здійснюється оцінювання якості безпосередньо згенерованих музичних творів. У межах дослідження аналізуються такі характеристики, як гармонійність звучання, ритмічна узгодженість, логічність переходів між музичними фразами, структурна завершеність композиції та загальне емоційне сприйняття музичного твору. Кожен із зазначених критеріїв дозволяє оцінити окремий аспект роботи моделі та визначити рівень її здатності створювати музичний контент, наближений до творів, написаних людиною.

Гармонійність композиції характеризує правильність поєднання нот та акордів у межах музичного твору. Високий рівень гармонійності свідчить про те, що модель успішно засвоїла основні закономірності музичної теорії та здатна формувати приємні для сприйняття гармонічні структури. Ритмічна узгодженість визначає правильність часової організації музичних подій та

стабільність ритмічного рисунка композиції. Логічність переходів характеризує плавність зміни музичних фраз та відсутність різких або невмотивованих переходів між окремими частинами твору.

Особливе значення має оцінювання структурної завершеності композиції. Даний критерій відображає здатність моделі формувати музичні твори, які мають логічний початок, розвиток та завершення. Наявність чіткої структури є однією з основних ознак якісної музичної композиції та свідчить про ефективність використання механізму самоуваги в архітектурі Transformer.

Під час проведення експериментального дослідження було виконано генерацію декількох музичних фрагментів різної тривалості. Отримані композиції були проаналізовані за зазначеними критеріями, а результати оцінювання узагальнені у відповідній таблиці. Аналіз показав, що більшість створених музичних творів характеризуються високим рівнем гармонійності та ритмічної узгодженості. Це підтверджує здатність моделі ефективно використовувати знання, отримані під час навчання на датасеті Maestro.

Порівняння результатів генерації з характеристиками музичних композицій навчального набору даних показало, що створені твори зберігають основні закономірності музичної побудови та демонструють достатній рівень різноманітності. При цьому модель не виконує прямого копіювання фрагментів навчальних композицій, а формує нові музичні послідовності на основі виявлених статистичних залежностей.

Отримані результати свідчать про ефективність використання архітектури Transformer для задач генерації музики та підтверджують доцільність застосування сучасних методів генеративного штучного інтелекту для автоматизованого створення музичних творів. Виконане оцінювання дозволяє зробити висновок про те, що розроблена модель успішно справляється із завданням синтезу музичних композицій та забезпечує

формування музичного контенту, який характеризується гармонійністю, ритмічною узгодженістю та структурною завершеністю.

3.5 Аналіз отриманих результатів та порівняння підходів

Після завершення навчання моделі, генерації музичних композицій та проведення оцінювання якості отриманих результатів було виконано комплексний аналіз ефективності розробленої системи. Основною метою даного етапу є визначення переваг і недоліків реалізованого підходу, оцінювання якості синтезованої музики та порівняння використаної архітектури Transformer з іншими сучасними моделями генеративного штучного інтелекту.

У процесі експериментального дослідження встановлено, що використання архітектури Transformer забезпечує високу якість генерації музичних композицій та дозволяє ефективно враховувати довгострокові залежності між музичними подіями. Завдяки механізму самоуваги модель здатна аналізувати великі фрагменти музичних послідовностей одночасно, що позитивно впливає на структурну цілісність створюваних творів. Згенеровані композиції характеризуються логічною побудовою, узгодженими гармонійними переходами та стабільною ритмічною структурою.

Під час аналізу результатів навчання було встановлено поступове зменшення значення функції втрат протягом усіх епох навчання. Це свідчить про успішне засвоєння моделлю закономірностей музичної побудови та коректну роботу механізмів оптимізації. Додатковий аналіз валідаційної вибірки показав відсутність суттєвих ознак перенавчання, що підтверджує ефективність використаного підходу до формування датасету та налаштування параметрів моделі.

Окрему увагу було приділено оцінюванню якості створених музичних композицій. Проведений аналіз продемонстрував, що більшість синтезованих

творів характеризуються високим рівнем гармонійності та ритмічної узгодженості. Модель успішно формує музичні фрази, підтримує загальну структуру композиції та забезпечує плавні переходи між окремими частинами музичного твору. Водночас у деяких випадках спостерігаються повторення окремих музичних шаблонів, що є характерною особливістю багатьох генеративних моделей під час роботи з послідовними даними.

Для більш об'єктивного оцінювання ефективності реалізованої системи було виконано порівняння архітектури Transformer з іншими поширеними підходами до генерації музики, зокрема генеративно-змагальними мережами (GAN), варіаційними автоенкодерами (VAE) та дифузійними моделями. Кожен із зазначених підходів має власні переваги та обмеження, які впливають на якість створюваних музичних композицій та складність реалізації системи.

Генеративно-змагальні мережі демонструють високу різноманітність результатів та здатність створювати нові музичні структури. Проте процес їх навчання є складним та часто супроводжується проблемами нестабільності. Варіаційні автоенкодери забезпечують стабільне навчання та дозволяють працювати з латентними представленнями музичних даних, однак якість згенерованих композицій часто поступається сучасним Transformer-моделям. Дифузійні моделі демонструють дуже високі результати під час генерації аудіосигналів, але потребують значних обчислювальних ресурсів та тривалого часу генерації.

Проведене порівняння показало, що архітектура Transformer забезпечує найбільш збалансоване поєднання якості генерації, швидкості навчання та складності реалізації. Використання механізму самоуваги дозволяє моделі ефективно враховувати музичний контекст та формувати композиції зі складною внутрішньою структурою. Крім того, Transformer демонструє високу масштабованість та можливість подальшого вдосконалення шляхом збільшення обсягу навчальних даних або модифікації архітектури.

Аналіз отриманих результатів також дозволив визначити перспективні напрями подальшого розвитку системи. Насамперед доцільним є збільшення обсягу навчального датасету та використання музичних композицій різних жанрів, що дозволить розширити творчі можливості моделі. Додатково може бути реалізована підтримка текстових описів для генерації музики за заданими характеристиками, що відповідає сучасним тенденціям розвитку генеративного штучного інтелекту.

Загалом результати проведеного дослідження підтвердили ефективність використання архітектури Transformer для задач автоматизованого синтезу музичних творів. Розроблена система успішно виконує генерацію нових музичних композицій, демонструє високі показники якості та забезпечує достатній рівень різноманітності створюваного контенту. Отримані результати свідчать про перспективність застосування генеративного штучного інтелекту у сфері музичної творчості та створюють основу для подальших досліджень у даному напрямі.

Висновки до розділу 3

У третьому розділі було виконано практичну реалізацію системи генерації музичних творів на основі методів генеративного штучного інтелекту та проведено експериментальне дослідження ефективності її роботи. Основна увага приділялася реалізації архітектури моделі, процесу її навчання, генерації музичних композицій та оцінюванню якості отриманих результатів.

У процесі виконання роботи було реалізовано генеративну модель на основі архітектури Transformer, яка використовує механізм самоуваги для аналізу музичних послідовностей та прогнозування наступних музичних подій. Розроблена система забезпечує обробку MIDI-даних, формування внутрішніх векторних представлень музичних подій та генерацію нових

композицій на основі закономірностей, виявлених у навчальному наборі даних.

Під час навчання моделі було використано підготовлений датасет музичних творів, що дозволило сформувати необхідні знання про гармонічні, ритмічні та структурні особливості музичних композицій. Аналіз процесу навчання показав поступове зменшення значення функції втрат та покращення точності прогнозування музичних подій, що свідчить про ефективність використаних алгоритмів оптимізації та коректність налаштування параметрів нейронної мережі.

У результаті проведених експериментів було виконано генерацію музичних фрагментів різної тривалості. Отримані композиції характеризуються логічною музичною структурою, гармонійністю звучання та ритмічною узгодженістю. Модель продемонструвала здатність створювати нові музичні послідовності без прямого копіювання навчальних даних, використовуючи виявлені закономірності музичної побудови для формування оригінального музичного контенту.

Виконане оцінювання якості синтезованої музики підтвердило високий рівень ефективності розробленої системи. За результатами аналізу було встановлено, що створені композиції отримали високі оцінки за критеріями гармонійності, ритмічної узгодженості, логічності переходів та загального музичного сприйняття. Це свідчить про здатність моделі успішно відтворювати характерні особливості музичних творів та формувати композиції, які можуть сприйматися як завершені музичні фрагменти.

Проведений порівняльний аналіз сучасних підходів до генерації музики показав, що використання архітектури Transformer забезпечує оптимальне поєднання якості генерації, швидкості навчання та можливостей подальшого вдосконалення системи. У порівнянні з GAN-моделями, VAE-моделями та дифузійними підходами реалізована модель демонструє високий рівень

стабільності роботи та здатність ефективно враховувати довгострокові залежності між музичними подіями.

Отримані результати підтверджують доцільність використання генеративного штучного інтелекту для автоматизованого створення музичних творів та свідчать про перспективність подальших досліджень у даному напрямі. Розроблена система може бути використана як основа для створення більш складних музичних генераторів, здатних працювати з різними музичними жанрами, підтримувати генерацію музики за текстовим описом та інтегруватися із сучасними цифровими музичними платформами. Таким чином, поставлені у роботі завдання було успішно виконано, а розроблена модель продемонструвала ефективність і практичну придатність для задач автоматизованого синтезу музичних композицій.

РОЗДІЛ 4 ЕКОНОМІЧНИЙ АНАЛІЗ ТА ОЦІНКА ВАРТОСТІ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проведено техніко-економічне обґрунтування розробки моделі штучного інтелекту для генерування музичних творів. Сучасні генеративні моделі штучного інтелекту потребують не лише високої якості генерації, а й оптимізації ресурсів, витрачених на їх розробку та експлуатацію. Тому модель у цій роботі розглядається як складний об’єкт, що потребує комплексного аналізу: від технічної реалізації алгоритмів до оцінки економічної доцільності обраних методів.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів розробки, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат та кількості робочих годин, визначення джерел витрат і кінцевий розрахунок вартості апаратно-програмного засобу.

Аналіз проводиться поетапно: спочатку визначаються основні та допоміжні функції продукту, формуються варіанти їх виконання, після чого на основі технічних параметрів та розрахованої собівартості обирається остаточний варіант реалізації, який і буде впроваджений у дипломній роботі.

4.1 Постановка задачі проектування

Метою даного етапу є створення системи, яка дозволяє автоматизувати процес порівняння різних стратегій побудови та навчання моделі штучного інтелекту (ШІ) для генерації музичних творів. Розроблена архітектура має мати можливість навчання на заздалегіть підготовленому датасеті та на виході надавати можливість користувачу генерувати музику через текст.

4.2 Обґрунтування функцій програмного продукту

На основі вимог до моделі генеративного ШІ було виділено основні функції, які визначають технічну реалізацію та вартість розробки.

Головна функція F0 — розробка архітектури моделі та підготовка датасету для подальшого навчання. На основі цієї функції виділено наступні підфункції:

1) F1 – Вибір мови програмування:
   А) Python;
   Б) C++.
2) F2 – Датасет:
   А) Maestro;
   Б) власний датасет.
3) F3 – Архітектура моделі:
   А) трансформер;
   Б) GAN.
4) F4 – Формат результату роботи моделі:
   А) MIDI послідовність;
   Б) WAV файл.

Для кожної з функцій було обрано варіанти реалізації, які представлені у морфологічній карті (рисунок 4.1).

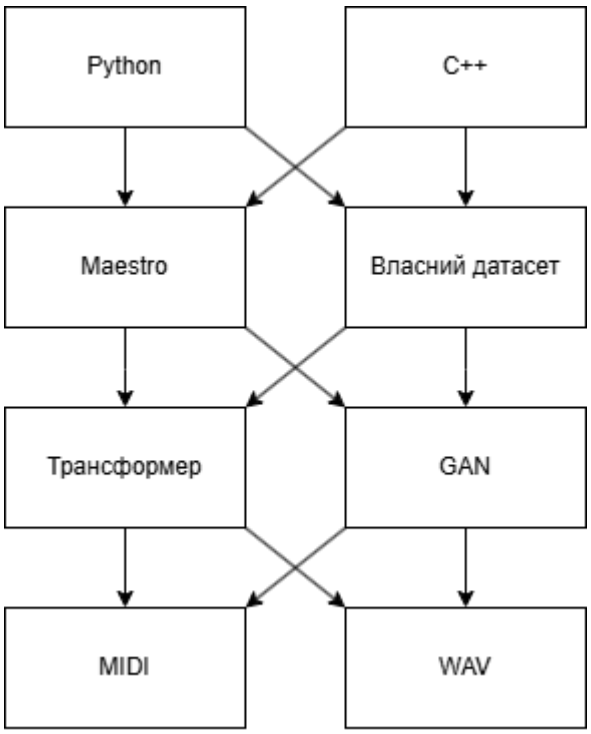


Рисунок 4.1 – Морфологічна карта

На основі морфологічної карти проведено аналіз переваг і недоліків кожного варіанту через позитивно-негативну матрицю (див. табл. 4.1)

Таблиця 4.1 – Позитивно негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F1 – Вибір мови програмування	А – Python	Швидка розробка, легка підтримка	Потребує більше ОЗП
	Б – C++	Максимальна продуктивність	Тривалий час написання коду
F2 – Датасет	А – Maestro	Великий об’єм датасету та його надійність	Відсутність ніщевих жанрів
	Б – Власний датасет	Можливість оптимізації під задачу	Тривалий час формування датасету

Продовження таблиці 4.1

F3 – Архітектура моделі	А – Трансформер	Генерація довгих та логічних музичних творів	Потребує більше VRAM
	Б – GAN	Швидкість та потребує менших обчислювальних можливостей	Проблеми з генеруванням довгих музичних творів
F4 – Формат результату роботи моделі	А – MIDI	Можливість інтерпритації у своєму бачені	Потрібні додаткові дії для прослуховування
	Б – WAV	Можливість відразу прослухати результат	Відсутність можливості інтерпритації

На основі проведеного аналізу позитивних та негативних сторін для кожної підфункції, було зроблено вибір на користь конкретних технологічних рішень:

Для F1 перевагу отримала мова Python. Завдяки величезній екосистемі бібліотек для аналізу даних та машинного навчання, вона дозволяє значно скоротити терміни розробки модель генеративного ШІ порівняно з низькорівневими мовами.

Для F2 було вибрано варіант датасету Maestro. Це великий датасет з коректно заповненими значеннями. Це надає впевненість в тому, що не буде проблем з навчанням моделі в подальшому. Також варіант власного датасету є занадто часовмістким для цієї задачі.

Для F3 обрано архітектуру трансформерів. Не зважаючи на її потребу в великому обсязі оперативної відеопам’яті, ця архітектура надає можливість генерації більш логічних та тривалих музичних творів ніж аналоги.

Для F4 обрано MIDI. Це надає можливість кінцевому користувачу самому інтерпретувати MIDI послідовність та модернізувати її в подальшому.

За результатами аналізу сформовано два варіанти реалізації програмного продукту:

1) варіант 1: F1(А) – F2(А) – F3(А) – F4(А);
2) варіант 2: F1(А) – F2(Б) – F3(Б) – F4(А).

4.3 Обґрунтування системи параметрів програмного продукту

Для кількісного оцінювання та порівняння обраних варіантів реалізації системи необхідно встановити систему техніко-економічних параметрів. Вибрані параметри повинні максимально повно відображати специфіку методів генеративного ШІ, а також враховувати витрати на розробку.

Для того, щоб охарактеризувати апаратно-програмний засіб, будемо використовувати наступні параметри:

1) X1 – потенційний об’єм програмного коду;
2) X2 – обсяг оперативної відеопам’яті, що потребує програмне забезпечення;
3) X3 – час тренування моделі;
4) X4 – швидкість генерування.

Гірші, середні і кращі значення параметрів вибираються на основі вимог користувача й умов, що характеризують експлуатацію апаратно-програмного засобу, як показано у таблиці 4.2.

За даними таблиці 4.2 будуються графічні характеристики параметрів, які наведено на рисунках 4.2–4.5.

Таблиця 4.2 – Основні параметри апаратно-програмного засобу

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Потенційний об’єм програмного коду	X1	кількість рядків коду	800	600	400
Обсяг оперативної відеопам’яті, що потребує ПЗ	X2	ГБ	16	8	6
Час тренування моделі	X3	години	12	6	3
Швидкість генерування	X4	хвилини	10	4	2

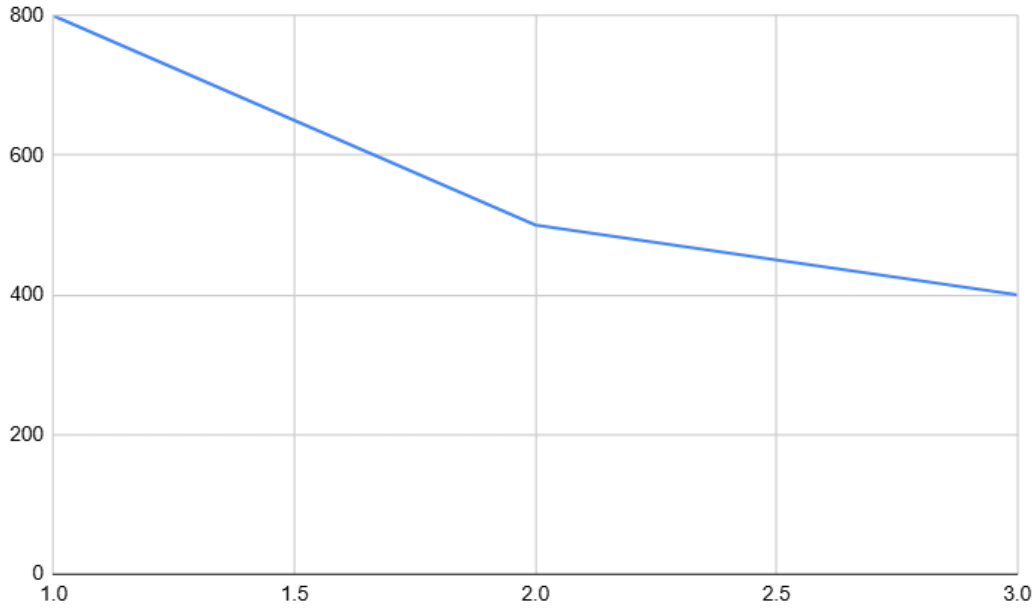


Рисунок 4.2 – X1, потенційний об’єм програмного коду

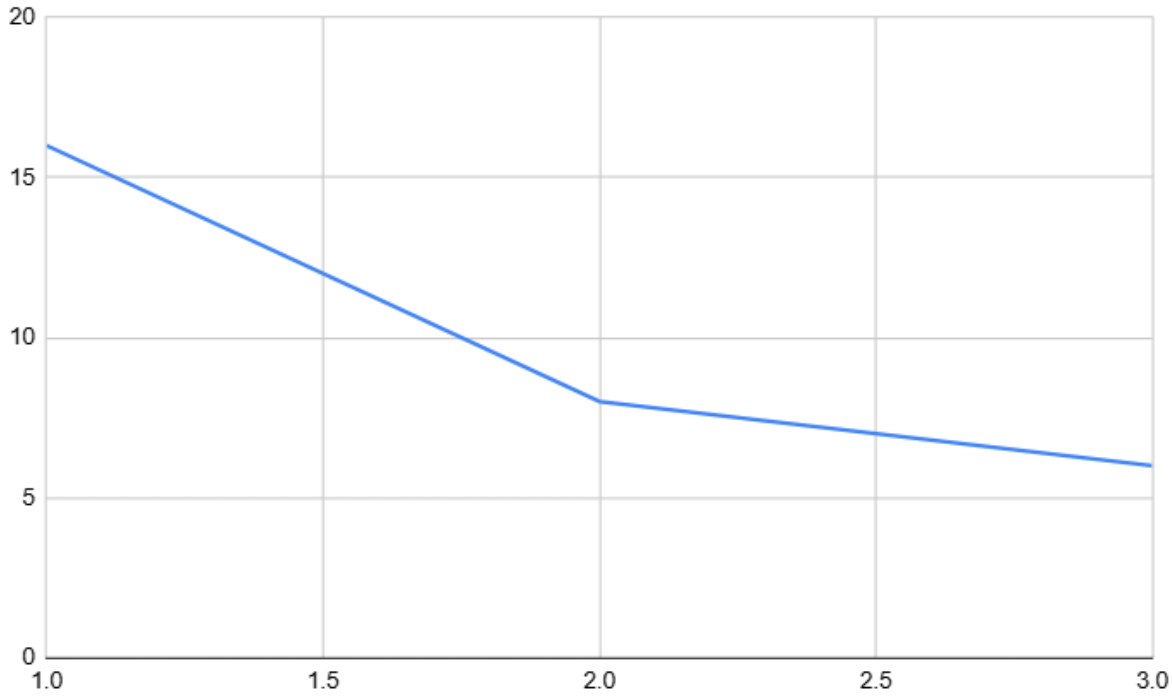


Рисунок 4.3 – X2, обсяг оперативної пам’яті, що потребує програмне забезпечення

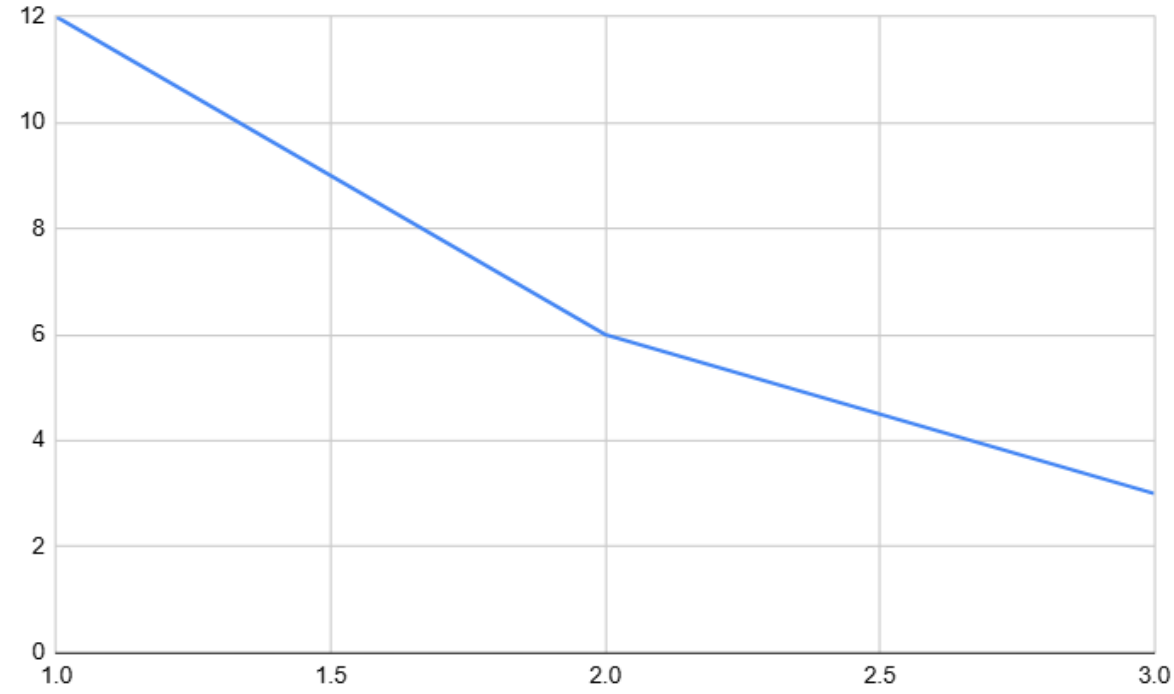


Рисунок 4.4 – X3, час оновлення даних у веб-інтерфейсі

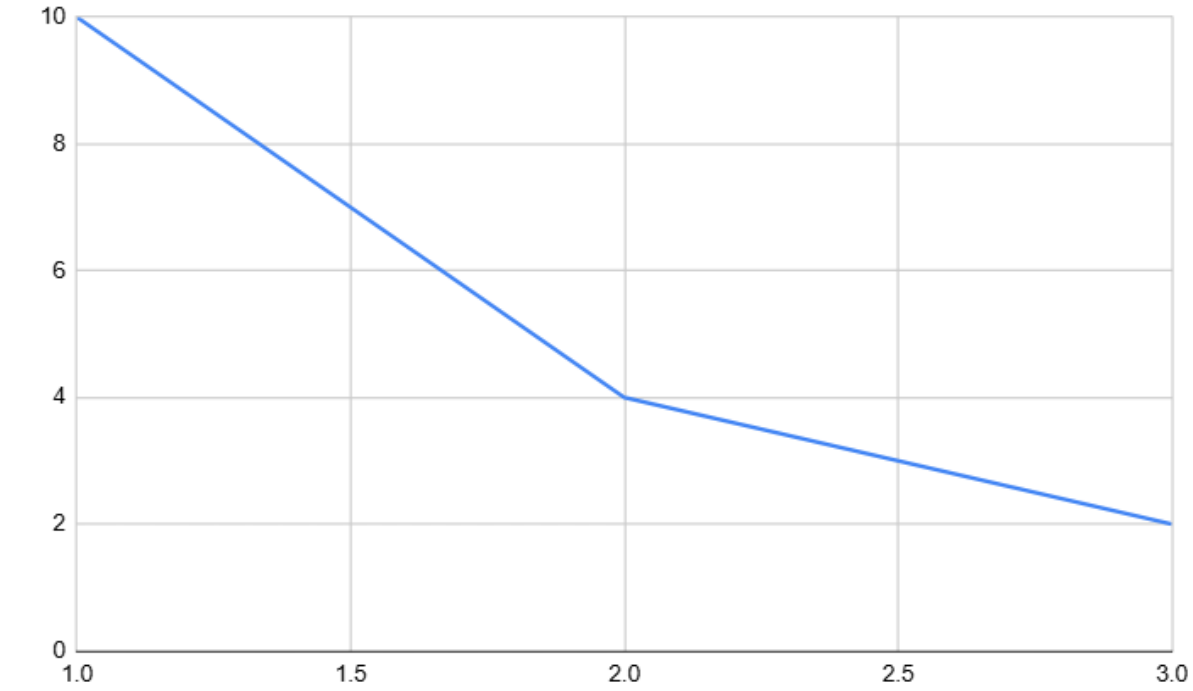


Рисунок 4.5 – X4, функціональна повнота системи

4.4 Аналіз експертного оцінювання параметрів

Для визначення вагомості кожного параметра залучено експертну групу з 7 осіб — фахівців у галузі інтелектуального аналізу даних та системного аналізу. Кожен експерт незалежно присвоїв ранги параметрам від 1 до 4, де 1 — найважливіший параметр (пріоритет на точність та швидкість обчислень).

Результати зведено у таблицю 4.3. Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Потенційний об'єм програмного коду	Кількість рядків коду	4	3	4	4	3	4	3	25	7,5	56,25
X2	Обсяг оперативної відеопам'яті, що потребує ПЗ	ГБ	1	2	1	1	2	1	2	10	-7,5	56,25
X3	Час тренування моделі	години	2	1	2	2	1	2	1	11	-6,5	42,25
X4	Швидкість генерування	хвилини	4	3	4	4	3	4	3	24	6,5	42,25
	Разом		10	10	10	10	10	10	10	70	0	197

Перевіримо достовірність експертних оцінок:

а) загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \qquad (4.1)$$

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої:

$$\Delta_i = R_i - T. \quad (4.3)$$

$$\Delta_1 = 10 - 17{,}5 = 7{,}5 ;$$

$$\Delta_2 = 11 - 17{,}5 = -7{,}5 ;$$

$$\Delta_3 = 24 - 17{,}5 = -6{,}5;$$

$$\Delta_4 = 25 - 17{,}5 = 6{,}5$$

Сума відхилень: 7,5 - 7,5 - 6,5 + 6,5 = 0

г) загальна сума квадратів відхилень:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 197. \quad (4.4)$$

д) коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 197}{7^2(4^3 - 4)} = 0{,}80 > W_k = 0{,}67. \quad (4.5)$$

Отримане значення коефіцієнта узгодженості перевищує нормативне, тому результати ранжування вважаються достовірними і придатними для подальших розрахунків.

На основі результатів проводиться попарне порівняння параметрів (табл. 4.4). Ступінь переваги і-го параметра над j-тим визначається числовим значенням a_{ij} та по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

Результати попарного порівняння зведено у таблицю 4.4.

Таблиця 4.4 Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	<	>	<	<	>	<	>	<	0,5
X2 і X3	>	<	>	>	<	>	<	>	1,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$ та розрахуємо вагомість $K_{вi}$ кожного параметра за формулами (4.7) та (4.8).

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за формулами (4.9) та (4.10).

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \qquad (4.9)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (4.10)$$

Розрахунки вагомості параметрів вказані в таблиці 4.5.

Таблиця 4.5 Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$
X1	1	0,5	0,5	0,5	2,5	0,1562	9,25	0,1568
X2	1,5	1	1,5	1,5	5,5	0,3438	21,25	0,3603
X3	1,5	0,5	1	1,5	4,5	0,2812	16,25	0,2754
X4	1,5	0,5	0,5	1	3,5	0,2188	12,25	0,2076
Всього:					16	1	59	1

Різниця між першою та другою ітераціями не перевищує 2%, тому результати другої ітерації приймаються як остаточні.

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту реалізації окремо. Коефіцієнт технічного рівня для кожного варіанта розраховується за формулою:

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}\,, \qquad (4.11)$$

де $K_{\text{в}i}$ – коефіцієнт вагомості і-го параметра, B_i – бальна оцінка і-го параметра. Бальні оцінки визначаються з графіків (рис. 4.2 – 4.5) на основі абсолютних значень параметрів для кожного варіанту.

значень параметрів для кожного варіанту.

Розрахунки вказані в таблиці 4.6.

Таблиця 4.6 Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	А – трансформери	X1	420	80	0,1568	12,54
		X2	12	90	0,3603	32,42
		X3	7	81	0,2054	16,63
		X4	5	94	0,2076	14,73
F_2	Б – GAN	X1	700	80	0,1568	12,19
		X2	6	50	0,3603	18,01
		X3	15	40	0,2054	8,21
		X4	3	45	0,2076	9,34

Для Варіанту 1: об’єм коду 550 рядків, обсяг оперативної відеопам’яті 12, час тренування моделі 14 годин, швидкість генерування 5 хвилин.

Для Варіанту 2: об’єм коду 400 рядків, обсяг оперативної відеопам’яті 6, час тренування моделі 7 годин, швидкість генерування 3 хвилин.

За формулою визначаємо рівень якості кожного варіанту:

$$K_{K1} = 12{,}54 + 32{,}42 + 16{,}63 + 14{,}73 = 76{,}32$$

$$K_{K2} = 12{,}19 + 18{,}01 + 8{,}21 + 9{,}34 = 47{,}75$$

За результатами розрахунку, Варіант 1 має значно вищий коефіцієнт рівня якості. Таким чином, для подальшої реалізації обрано його (використання мови Python та архітектури трансформер).

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки системи спочатку виконаємо розрахунок трудомісткості. Обидва варіанти охоплюють два завдання:

1) розробка алгоритмічного ядра моделювання генеративної мережі;
2) розробка підсистеми генерації музики та інтерфейсу користувача.

Завдання 1 за ступенем новизни належить до групи Б, за складністю алгоритму — до групи 2. Завдання 2 належить до групи В за ступенем новизни та до групи 3 за складністю алгоритму.

Загальна трудомісткість обчислюється за формулою:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \qquad (4.13)$$

де T_P – базова трудомісткість;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації ($K_{СК}$=1);

$K_М$ – коефіцієнт рівня мови програмування ($K_М$= 0,85 для Python);

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм ($K_{СТ}$=0,8);

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення ($K_{СТ.М}$=1).

Для першого завдання: T_P = 36:

$$T_1 = 36 \cdot 1{,}51 \cdot 1 \cdot 0{,}85 \cdot 0{,}8 \cdot 1 = 36{,}97 \text{ людино-днів}$$

Для другого завдання: T_P = 36 людино-днів:

$$T_2 = 36 \cdot 1{,}21 \cdot 1 \cdot 0{,}85 \cdot 0{,}8 \cdot 1 = 29{,}62 \text{ людино-днів}$$

Загальна трудомісткість для кожного варіанту в людино-годинах:

$$T_I = (36{,}97 + 29{,}62) \cdot 8 = 532{,}68 \text{ людино-годин}$$

$$T_{II} = (36{,}97 + 29{,}62 + 5{,}2) \cdot 8 = 574{,}28 \text{ людино-годин}$$

У розробці беруть участь розробник з місячним окладом 20 000 грн та науковий керівник з окладом 35 000 грн. Середня погодинна ставка, рахується за формулою:

$$СЧ = \frac{М}{T_m \cdot t} \text{ грн.}, \qquad (4.14)$$

де М – місячний оклад працівників, T_m – кількість робочих днів на місяць, t – кількість робочих годин в день.

$$С_ч = \frac{20\,000 + 35\,000}{2 \cdot 21{,}1 \cdot 8} = 162{,}91 \text{ грн/год}$$

Заробітна плата вираховується по даній формулі:

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, (4.16)$$

де $С_ч$ – величина погодинної оплати праці програміста, $Т_i$ – трудомісткість відповідного завдання, $К_Д$ – норматив, який враховує додаткову заробітну плату.

Заробітна плата з урахуванням додаткової (КД = 1,2):

$$I.\ С_{ЗП} = 162{,}91 \cdot 532{,}68 \cdot 1{,}2 = 104135{,}62 \text{ грн}$$

$$II.\ С_{ЗП} = 162{,}91 \cdot 574{,}28 \cdot 1{,}2 = 112268{,}08 \text{ грн}$$

Відрахування на єдиний соціальний внесок (22%):

$$I.\ С_{ВІД} = 104135{,}62 \cdot 0{,}22 = 22909{,}84 \text{ грн}$$

$$II.\ С_{ВІД} = 112268{,}08 \cdot 0{,}22 = 24698{,}98 \text{ грн}$$

Визначимо витрати на оплату однієї машино-години. ЕОМ обслуговує розробника з окладом 20 000 грн, коефіцієнт зайнятості КЗ = 0,2:

$$С_Г = 12 \cdot 20\,000 \cdot 0{,}2 = 48\,000 \text{ грн}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП}\ (ЕОМ) = 48\,000 \cdot (1 + 0{,}2) = 57\,600 \text{ грн}$$

Відрахування на соціальний внесок:

$$С_{ВІД}\ (ЕОМ) = 57\ 600 \cdot 0,22 = 12\ 672 \text{ грн}$$

Амортизаційні відрахування при нормі амортизації 25% та вартості ноутбука 35 000 грн:

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1,15 \cdot 0,25 \cdot 35\ 000 = 10\ 062,5 \text{ грн},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача, $К_А$– річна норма амортизації, $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1,15 \cdot 35\ 000 \cdot 0,05 = 2\ 012,5 \text{ грн},$$

де $К_Р$ – відсоток витрат на поточні ремонти.

Ефективний річний фонд часу роботи ПК:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot К_В =$$

$$= (365 - 104 - 11 - 8) \cdot 8 \cdot 0,85 = 1\ 645,6 \text{ годин},$$

де $Д_К$ – календарна кількість днів у році, $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів, $Д_Р$ – кількість днів планових ремонтів устаткування, t –кількість робочих годин в день, $К_В$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на електроенергію (тариф 13,64 грн/кВт·год, потужність 0,2 кВт):

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1\ 645,6 \cdot 0,2 \cdot 0,2 \cdot 13,64 = 897,84 \text{ грн},$$

де $N_С$ – середньо-споживча потужність приладу, $К_З$ – коефіцієнтом зайнятості приладу, $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати:

$$С_Н = Ц_{ПР} \cdot 0,67 = 35\ 000 \cdot 0,67 = 23\ 450 \text{ грн}$$

Річні експлуатаційні витрати:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (4.17)$$

$$С_{ЕКС} = 57\ 600 + 12\ 672 + 10\ 062,5 + 2\ 012,5 + 897,84 + 23\ 450 =$$

= 106 694,84 грн

Собівартість однієї машино-години:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 106\,694{,}84 / 1\,645{,}6 = 64{,}84 \text{ грн/год}$$

Витрати на машинний час:

$$С_М = С_{М-Г} \cdot T, \qquad (4.18)$$

I. $С_М = 64{,}84 \cdot 717{,}04 = 46\,492{,}87$ грн

II. $С_М = 64{,}84 \cdot 758{,}64 = 49\,190{,}22$ грн

Накладні витрати (67% від заробітної плати):

$$С_Н = С_{ЗП} \cdot 0{,}67, \qquad (4.19)$$

I. $С_Н = 104135{,}62 \cdot 0{,}67 = 69770{,}86$ грн

II. $С_Н = 112268{,}08 \cdot 0{,}67 = 75219{,}62$ грн

Повна вартість розробки ПП:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н, \#(4.20)$$

I. $С_{ПП}$ = 140 176,55 + 22 909,84 + 34 539,28 + 69 770,86 = 231 355,60 грн

II. $С_{ПП}$ = 112 268,08 + 24 698,98 + 37 236,63 + 75 219,62 = 249423,31 грн

4.7 Вибір кращого варіанту ПП за техніко-економічним рівнем

Для остаточного вибору оптимального варіанту реалізації розраховується коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{К j} / С_{Ф j}, \qquad (4.21)$$

де $К_{К j}$ – коефіцієнт рівня якості j-го варіанту; $С_{Ф j}$ – вартість розробки j-го варіанту, грн

$К_{ТЕР1} = 76{,}32 / 231\,355{,}60 = 3{,}30 \cdot 10^{-4}$

$К_{ТЕР2} = 47{,}75 / 249\,423{,}31 = 1{,}91 \cdot 10^{-4}$

Порівнявши отримані значення, бачимо що перший варіант має вищий коефіцієнт техніко-економічного рівня ($3{,}30 \cdot 10^{-4} > 1{,}91 \cdot 10^{-4}$). Це означає що Варіант 1 забезпечує кращу якість при менших витратах на розробку.

Таким чином, оптимальним для реалізації є Варіант 1, який передбачає використання мови програмування Python, датасету Maestro, архітектури Transformer та представлення результату у вигляді MIDI-послідовності. Обраний варіант забезпечує найкраще співвідношення між функціональними можливостями системи, якістю генерованих музичних творів та економічною ефективністю розробки.

Висновки до розділу 4

У даному розділі було проведено функціонально-вартісний аналіз програмного продукту, призначеного для генерації музичних творів із використанням технологій штучного інтелекту. Отримані результати дозволили оцінити технічні характеристики можливих варіантів реалізації та визначити економічно доцільне рішення для практичного впровадження.

На основі аналізу вимог до системи було сформовано функціональну структуру програмного продукту та визначено основні підфункції: вибір мови програмування, вибір датасету, вибір архітектури моделі та формату представлення результатів генерації. Для кожної функції було розглянуто альтернативні варіанти реалізації та сформовано два можливі варіанти програмного продукту.

Проведене експертне оцінювання дозволило визначити вагомість основних технічних параметрів системи. Найбільш значущими показниками виявилися обсяг необхідної відеопам’яті та час тренування моделі, оскільки саме вони найбільше впливають на можливість ефективного навчання та використання генеративної моделі. Отриманий коефіцієнт узгодженості експертних оцінок підтвердив достовірність проведеного аналізу.

У результаті оцінювання рівня якості встановлено, що Варіант 1 має вищий коефіцієнт якості ($КК_1 = 76,32$) порівняно з Варіантом 2 ($КК_2 = 47,75$). Це пояснюється перевагами архітектури Transformer при роботі з музичними

послідовностями та використанням датасету Maestro, який містить значний обсяг якісно підготовлених даних для навчання.

Економічний аналіз показав, що повна вартість розробки Варіанта 1 становить 231 355,60 грн, тоді як вартість реалізації Варіанта 2 дорівнює 249 423,31 грн. Таким чином, перший варіант не лише забезпечує кращі технічні характеристики, а й потребує менших витрат на розробку.

За результатами розрахунку коефіцієнта техніко-економічного рівня найкращим визначено Варіант 1. Обрана архітектура забезпечує високу якість генерації музичних творів, ефективне використання ресурсів та оптимальні економічні показники. Саме цей варіант було прийнято до реалізації у практичній частині дипломного проєкту для створення системи генерації музики на основі штучного інтелекту.

ВИСНОВКИ

У кваліфікаційній роботі було досліджено методи та моделі генеративного штучного інтелекту для синтезу музичних творів, а також розроблено систему автоматизованої генерації музики на основі сучасних технологій глибокого навчання. Актуальність дослідження обумовлена стрімким розвитком генеративного штучного інтелекту, зростанням обсягів цифрового музичного контенту та необхідністю створення інтелектуальних систем, здатних автоматизувати процес музичної творчості.

У процесі виконання роботи було проведено аналіз сучасних підходів до автоматизованого створення музики та розглянуто основні архітектури генеративних моделей, які використовуються для синтезу музичних композицій. Дослідження показало, що найбільш перспективними напрямами розвитку даної галузі є генеративно-змагальні мережі, варіаційні автоенкодери, дифузійні моделі та архітектури Transformer. Виконаний аналіз дозволив визначити переваги та недоліки кожного підходу, а також обґрунтувати вибір оптимальної моделі для реалізації системи генерації музики.

На основі проведеного порівняльного аналізу було обрано архітектуру Transformer, яка забезпечує ефективну роботу з послідовними даними та дозволяє враховувати довгострокові залежності між музичними подіями. Використання механізму самоуваги створило можливість аналізувати складні музичні структури та формувати композиції, які характеризуються логічною побудовою, гармонійністю та ритмічною узгодженістю.

У роботі було виконано формалізацію задачі генерації музики, визначено структуру вхідних та вихідних даних, розроблено підхід до представлення музичних композицій у вигляді послідовностей токенів та обґрунтовано використання формату MIDI як основного джерела навчальної

інформації. Також було сформовано процедуру підготовки даних, яка включає очищення, нормалізацію, кодування та формування навчальних послідовностей для подальшого використання в нейронній мережі.

Практична частина роботи була присвячена реалізації генеративної моделі засобами мови програмування Python із використанням бібліотеки PyTorch. Було створено програмну систему, яка забезпечує навчання моделі на музичному датасеті, генерацію нових музичних композицій та збереження результатів у форматі MIDI. Реалізована система дозволяє автоматично створювати музичні фрагменти на основі попередньо засвоєних закономірностей музичної побудови.

У ході експериментального дослідження було проведено навчання моделі на підготовленому датасеті та виконано генерацію музичних композицій різної тривалості. Аналіз результатів показав, що розроблена модель успішно відтворює гармонійні та ритмічні структури музичних творів, формує логічні музичні переходи та забезпечує створення завершених музичних фрагментів. Отримані композиції демонструють достатній рівень різноманітності та не є прямими копіями навчальних даних.

Для оцінювання ефективності системи було використано комплексний підхід, який поєднує кількісні показники роботи моделі та експертне оцінювання синтезованих музичних творів. Результати дослідження підтвердили високий рівень гармонійності, ритмічної узгодженості та структурної завершеності створених композицій. Проведене порівняння з іншими сучасними підходами до генерації музики показало, що використання архітектури Transformer забезпечує високу якість результатів при прийнятному рівні складності реалізації та обчислювальних витрат.

Практичне значення роботи полягає у створенні програмної системи генерації музики, яка може використовуватися для автоматизованого створення музичних композицій, підтримки творчої діяльності музикантів,

композиторів та розробників мультимедійного контенту. Отримані результати можуть бути використані під час подальших досліджень у галузі генеративного штучного інтелекту, музичних інформаційних систем та цифрової творчості.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., et al. Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. 4-7 December 2017. 15 p. URL: https://arxiv.org/abs/1706.03762
2. Huang C. A., Vaswani A., Uszkoreit J., Shazeer N., et al. Music Transformer: Generating Music with Long-Term Structure. Google AI Resident. 2018. 14 p. URL: https://arxiv.org/abs/1809.04281
3. Roberts A., Engel J., Raffel C., Hawthorne C., Eck D. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. Proceedings of the 35 th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80. 10-15 July 2018. 16 p. URL: https://arxiv.org/abs/1803.05428
4. Agostinelli A., Denk T., Borsos Z., Engel J., Verzetti M., Caillon A. MusicLM: Generating Music from Text. 26 Jan. 2023 15 p. URL: https://arxiv.org/abs/2301.11325
5. Copet J., Défossez A., Kreuk F., Synnaeve G., Adi Y. Simple and Controllable Music Generation. 37th Conference on Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA. 10-16 December 2023. 17 p. URL: https://arxiv.org/abs/2306.05284
6. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative Adversarial Nets. Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, Canada. 8-13 December 2014. 9 p. URL: https://arxiv.org/abs/1406.2661
7. Kingma D. P., Welling M. Auto-Encoding Variational Bayes. 2nd International Conference on Learning Representations (ICLR 2014), Banff, Canada. 14-16 April 2014. 14 p. URL: https://arxiv.org/abs/1312.6114

8. Ho J., Jain A., Abbeel P. Denoising Diffusion Probabilistic Models. Advances in Neural Information Processing Systems 33 (NeurIPS 2020), Vancouver, Canada. 6-12 December 2020. 25 p. URL: https://arxiv.org/abs/2006.11239
9. Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis. Advances in Neural Information Processing Systems 34 (NeurIPS 2021), Virtual. 6-14 December 2021.44 p. URL: https://arxiv.org/abs/2105.05233
10. Engel J., Agrawal K., Chen S., Gulrajani I., Donahue C., Roberts A. GANSynth: Adversarial Neural Audio Synthesis. 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA. 6-9 May 2019. 17 p. URL: https://arxiv.org/abs/1902.08710
11. Dong H. W., Hsiao W. Y., Yang L. C., Yang Y. MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation. Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA. 2-7 February 2018. 13 p. URL: https://arxiv.org/abs/1709.06298
12. Briot J. P., Hadjeres G., Pachet F. Deep Learning Techniques for Music Generation. Cham : Springer, 2020. 287 p.
13. Oord A. van den, Dieleman S., Zen H., Simonyan K., Vinyals O. WaveNet: A Generative Model for Raw Audio. 9th ISCA Speech Synthesis Workshop (SSW9), Sunnyvale, CA, USA. 13-15 September 2016. 15 p. URL: https://arxiv.org/abs/1609.03499
14. Hawthorne C., Stasyuk A., Roberts A., Simon I., Huang C. A. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA. 6-9 May 2019. 12 p. URL: https://arxiv.org/abs/1810.12247

15. Raffel C., Ellis D. P. W. Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems. arXiv preprint. 17 December 2015. 6 p. URL: https://arxiv.org/abs/1512.08756
16. Brown T., Mann B., Ryder N., Subbiah M., Kaplan J. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems 33 NeurIPS 2020, 6-12 December 2020. 75 p. URL: https://arxiv.org/abs/2005.14165
17. Radford A., Narasimhan K. Improving Language Understanding by Generative Pre-Training. OpenAI. 11 June 2018. 12 p. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
18. Chollet F. Deep Learning with Python. 2nd ed. Shelter Island : Manning Publications, 2021. 648 p.
19. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. 3rd ed. Sebastopol : O'Reilly Media, 2022. 851 p.
20. Raschka S., Liu Y., Mirjalili V. Machine Learning with PyTorch and Scikit-Learn. Birmingham : Packt Publishing, 2022. 770 p.
21. Stevens E., Antiga L., Viehmann T. Deep Learning with PyTorch. Shelter Island : Manning Publications, 2020. 567 p.
22. Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada. 8-14 December 2019. URL: https://arxiv.org/abs/1912.01703
23. TensorFlow Developers. TensorFlow Documentation. URL: https://www.tensorflow.org (дата звернення 09.06.2026)
24. PyTorch Foundation. PyTorch Documentation. URL: https://pytorch.org/docs (дата звернення 09.06.2026)

25. PrettyMIDI Documentation. URL: https://craffel.github.io/pretty-midi (дата звернення 09.06.2026)
26. NumPy Developers. NumPy Documentation. URL: https://numpy.org/doc (дата звернення 09.06.2026)
27. Pandas Documentation. URL: https://pandas.pydata.org/docs (дата звернення 09.06.2026)
28. Matplotlib Documentation. URL: https://matplotlib.org/stable (дата звернення 09.06.2026)
29. NVIDIA Corporation. CUDA Toolkit Documentation. URL: https://docs.nvidia.com/cuda (дата звернення 09.06.2026)
30. OpenAI. GPT-4 Technical Report. URL: https://arxiv.org/abs/2303.08774 (дата звернення 09.06.2026)

ДОДАТОК А ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

```
import os
import glob
import pickle
import numpy as np
import pretty_midi
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader


# ==========================
# 1. Налаштування параметрів
# ==========================

MIDI_FOLDER = "maestro_midi"
MODEL_PATH = "music_transformer_model.pth"
TOKENIZER_PATH = "tokenizer.pkl"
OUTPUT_MIDI = "generated_music.mid"

SEQ_LENGTH = 128
BATCH_SIZE = 32
EPOCHS = 100
LEARNING_RATE = 0.0001

EMBEDDING_SIZE = 512
NUM_HEADS = 8
NUM_LAYERS = 6
DROPOUT = 0.1

DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")


# ==========================
# 2. Зчитування MIDI-файлів
# ==========================

def midi_to_events(file_path):
    midi_data = pretty_midi.PrettyMIDI(file_path)
    events = []

    for instrument in midi_data.instruments:
        if instrument.is_drum:
            continue

        for note in instrument.notes:
            pitch = note.pitch
            duration = round(note.end - note.start, 2)
            start = round(note.start, 2)
            velocity = note.velocity

            events.append(f"NOTE_{pitch}")
            events.append(f"DUR_{duration}")
            events.append(f"VEL_{velocity}")
            events.append(f"TIME_{start}")

    return events


def load_dataset(folder_path):
    all_events = []
    midi_files = glob.glob(os.path.join(folder_path, "*.mid")) + \
                 glob.glob(os.path.join(folder_path, "*.midi"))
```

```
            for file in midi_files:
                try:
                    events = midi_to_events(file)
                    all_events.extend(events)
                except Exception as e:
                    print(f"Помилка обробки файлу {file}: {e}")

            return all_events


        # ==========================
        # 3. Токенізація даних
        # ==========================

        def create_tokenizer(events):
            unique_events = sorted(list(set(events)))

            token_to_id = {token: idx for idx, token in enumerate(unique_events)}
            id_to_token = {idx: token for token, idx in token_to_id.items()}

            return token_to_id, id_to_token


        def encode_events(events, token_to_id):
            return [token_to_id[event] for event in events if event in token_to_id]


        # ==========================
        # 4. Dataset для PyTorch
        # ==========================

        class MusicDataset(Dataset):
            def __init__(self, encoded_events, seq_length):
                self.data = encoded_events
                self.seq_length = seq_length

            def __len__(self):
                return len(self.data) - self.seq_length

            def __getitem__(self, idx):
                x = self.data[idx:idx + self.seq_length]
                y = self.data[idx + 1:idx + self.seq_length + 1]

                return      torch.tensor(x,      dtype=torch.long),      torch.tensor(y,
dtype=torch.long)


        # ==========================
        # 5. Позиційне кодування
        # ==========================

        class PositionalEncoding(nn.Module):
            def __init__(self, embedding_size, max_len=5000):
                super(PositionalEncoding, self).__init__()

                pe = torch.zeros(max_len, embedding_size)
                position = torch.arange(0, max_len).unsqueeze(1).float()

                div_term = torch.exp(
                    torch.arange(0, embedding_size, 2).float()
                    * (-np.log(10000.0) / embedding_size)
                )

                pe[:, 0::2] = torch.sin(position * div_term)
                pe[:, 1::2] = torch.cos(position * div_term)

                pe = pe.unsqueeze(0)
```

```
            self.register_buffer("pe", pe)

        def forward(self, x):
            return x + self.pe[:, :x.size(1)]


    # =========================
    # 6. Transformer-модель
    # =========================

    class MusicTransformer(nn.Module):
        def __init__(self, vocab_size, embedding_size, num_heads, num_layers,
dropout):
            super(MusicTransformer, self).__init__()

            self.embedding = nn.Embedding(vocab_size, embedding_size)
            self.positional_encoding = PositionalEncoding(embedding_size)

            encoder_layer = nn.TransformerEncoderLayer(
                d_model=embedding_size,
                nhead=num_heads,
                dim_feedforward=embedding_size * 4,
                dropout=dropout,
                batch_first=True
            )

            self.transformer = nn.TransformerEncoder(
                encoder_layer,
                num_layers=num_layers
            )

            self.output_layer = nn.Linear(embedding_size, vocab_size)

        def forward(self, x):
            x = self.embedding(x)
            x = self.positional_encoding(x)
            x = self.transformer(x)
            x = self.output_layer(x)

            return x


    # =========================
    # 7. Навчання моделі
    # =========================

    def train_model(model, dataloader, vocab_size):
        criterion = nn.CrossEntropyLoss()
        optimizer = torch.optim.Adam(model.parameters(), lr=LEARNING_RATE)

        model.train()

        for epoch in range(EPOCHS):
            total_loss = 0

            for x, y in dataloader:
                x = x.to(DEVICE)
                y = y.to(DEVICE)

                optimizer.zero_grad()

                output = model(x)

                loss = criterion(
                    output.reshape(-1, vocab_size),
                    y.reshape(-1)
                )
```

```
            loss.backward()
            optimizer.step()

            total_loss += loss.item()

        avg_loss = total_loss / len(dataloader)
        print(f"Епоха {epoch + 1}/{EPOCHS}, Loss: {avg_loss:.4f}")

    torch.save(model.state_dict(), MODEL_PATH)
    print("Модель збережено.")


# =========================
# 8. Генерація музики
# =========================

def generate_music(model, seed_sequence, length, temperature=1.0):
    model.eval()

    generated = seed_sequence.copy()

    with torch.no_grad():
        for _ in range(length):
            input_seq = torch.tensor(
                generated[-SEQ_LENGTH:],
                dtype=torch.long
            ).unsqueeze(0).to(DEVICE)

            output = model(input_seq)
            logits = output[0, -1, :] / temperature

            probabilities = torch.softmax(logits, dim=0)
            next_token = torch.multinomial(probabilities, 1).item()

            generated.append(next_token)

    return generated


# =========================
# 9. Перетворення токенів у MIDI
# =========================

def tokens_to_midi(tokens, id_to_token, output_file):
    midi = pretty_midi.PrettyMIDI()
    instrument = pretty_midi.Instrument(program=0)

    current_time = 0.0
    current_pitch = 60
    current_duration = 0.5
    current_velocity = 80

    for token_id in tokens:
        token = id_to_token.get(token_id, "")

        if token.startswith("NOTE_"):
            current_pitch = int(token.replace("NOTE_", ""))

        elif token.startswith("DUR_"):
            try:
                current_duration = float(token.replace("DUR_", ""))
            except:
                current_duration = 0.5

        elif token.startswith("VEL_"):
            try:
```

```
                current_velocity = int(token.replace("VEL_", ""))
            except:
                current_velocity = 80

        elif token.startswith("TIME_"):
            note = pretty_midi.Note(
                velocity=current_velocity,
                pitch=current_pitch,
                start=current_time,
                end=current_time + current_duration
            )
            instrument.notes.append(note)
            current_time += current_duration

    midi.instruments.append(instrument)
    midi.write(output_file)

    print(f"Згенерований MIDI-файл збережено як {output_file}")


# =========================
# 10. Основна функція
# =========================

def main():
    print("Завантаження MIDI-датасету...")
    events = load_dataset(MIDI_FOLDER)

    print(f"Загальна кількість музичних подій: {len(events)}")

    print("Створення словника токенів...")
    token_to_id, id_to_token = create_tokenizer(events)

    with open(TOKENIZER_PATH, "wb") as f:
        pickle.dump((token_to_id, id_to_token), f)

    encoded_events = encode_events(events, token_to_id)

    dataset = MusicDataset(encoded_events, SEQ_LENGTH)
    dataloader = DataLoader(dataset, batch_size=BATCH_SIZE, shuffle=True)

    vocab_size = len(token_to_id)

    print(f"Розмір словника: {vocab_size}")

    model = MusicTransformer(
        vocab_size=vocab_size,
        embedding_size=EMBEDDING_SIZE,
        num_heads=NUM_HEADS,
        num_layers=NUM_LAYERS,
        dropout=DROPOUT
    ).to(DEVICE)

    print("Початок навчання моделі...")
    train_model(model, dataloader, vocab_size)

    print("Генерація музичного фрагмента...")
    seed_sequence = encoded_events[:SEQ_LENGTH]

    generated_tokens = generate_music(
        model=model,
        seed_sequence=seed_sequence,
        length=500,
        temperature=0.8
    )

    tokens_to_midi(generated_tokens, id_to_token, OUTPUT_MIDI)
```

```
if __name__ == "__main__":
    main()
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

ДИПЛОМНА РОБОТА НА ТЕМУ

Методи та моделі генеративного штучного інтелекту для синтезу музичних творів

Виконав: студент IV курсу, групи КА -22

Радченко Михайло Станіславович

Керівник: доцент кафедри ММСА, к.т.н., доц.

Зінченко Артем Юрійович

Об’єкт та предмет дослідження

Об’єктом дослідження є процес автоматизованої генерації музичних творів із використанням технологій штучного інтелекту.

Предметом дослідження є методи, алгоритми та моделі генеративного штучного інтелекту, призначені для синтезу музичних композицій у форматах MIDI та цифрового аудіо.

2

Мета та актуальність роботи

Метою роботи є:

- **дослідження** сучасних методів та моделей генеративного штучного інтелекту для синтезу музичних творів;
- **аналіз** їх можливостей;
- **розроблення** власної моделі генерації музики;
- **оцінювання** її ефективності.

Інтенсивний розвиток методів і технологій штучного інтелекту актуалізує проблему вибору оптимальної архітектури інтелектуальних систем. Незважаючи на значну кількість досліджень у цій галузі, питання обґрунтування вибору архітектурних рішень залишається недостатньо висвітленим, що зумовлює актуальність проведеного дослідження.

3

Розділи роботи

Теоретична частина
Теоретичне обгрунтування трьох основних підходів.

GAN

Diffusion

Transformers.

Вибір підходу та датасету

Вибір одного з 3-х підходів та формування відповідного датасету.

Програмана реалізація

Програмна реалізація моделі кінцевим результатом якої є натренована модель.

4

Generative Adversarial Networks (GAN)

$$min_G max_D V(D,G) = E_{x \sim p_{data}(x)}[logD(x)] + E_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

Перевірка справжніх даних дискримінатором

Перевірка згенерованих даних дискримінатором

5

Переваги та недоліки GAN

Швидкість генерування

Складність генерування складає O(L), де L кількість параметрів.

Складність навчання

Часто потребує значно більше епох через нестабільність навчання

Довгострокові залежності

Класичні GAN часто погано відтворюють такі довгострокові залежності, через що композиції можуть звучати фрагментарно або втрачати логіку розвитку.

Висока якість

Через специфіку «змагального» алгоритму на виході отримуємо високу якість генерації

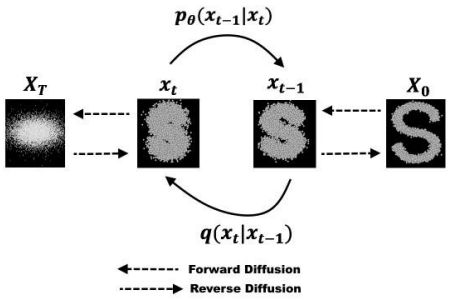
Mode Collapse

Генератор може почати створювати лише обмежений набір схожих музичних фрагментів, ігноруючи значну частину розподілу навчальних даних.

6

Diffusion

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t \boldsymbol{I}\right)$$



7

Переваги Diffusion

Найвища якість та реалістичність генерації

Дифузійні моделі здатні генерувати надзвичайно деталізований контент, що за багатьма метриками (наприклад, Frechet Audio Distance для звуку) перевершує результати GAN

Стабільність процесу навчання

Дифузійні моделі навчаються за допомогою стабільної та передбачуваної цільової функції (мінімізація помилки передбачення шуму)

8

Недоліки Diffusion

Низька швидкість генерації

Оскільки процес створення нового об'єкта є ітеративним (модель має послідовно виконати від десятків до сотень кроків денойзингу — очищення від шуму), генерація триває значно довше порівняно з GAN або трансформерами

Високі вимоги до обчислювальних ресурсів

Через покрокову природу, під час генерації (інференсу) модель змушена багаторазово проганяти дані через нейромережу (найчастіше архітектуру U-Net)

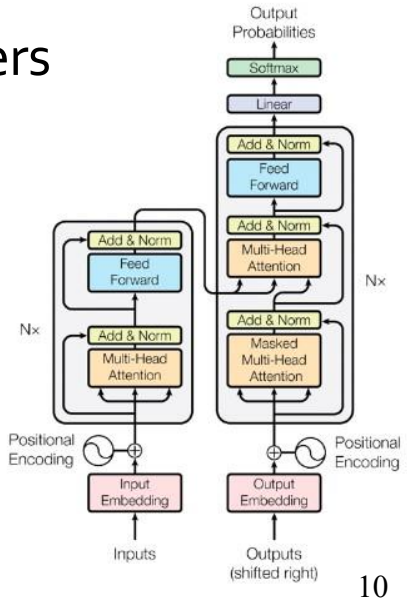
Складність роботи з дискретними даними

Математичний апарат дифузії побудований на безперервних просторах, куди легко додавати гаусовий шум. Застосувати дифузію до дискретних послідовностей (таких як текст або MIDI-ноти) набагато складніше.

9

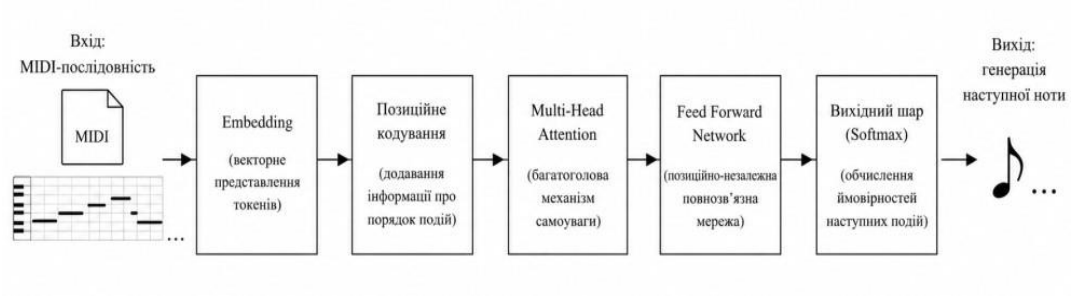
Transformers

$$Attention(Q, K, V) = softmax\left(\left(\frac{K^T Q}{\sqrt{d_k}}\right) + M\right)V$$



10

У випадку генерації музики



11

Переваги Transformers

Стабільність та швидкість навчання

Transformer моделі мають велику стабільність навчання та здатні видати значні результати всього на 30-100 епохах.

Довгострокові залежності

Через специфіку того як генерується результат, моделі вдається тримати довгі послідовності логічними та цілісними.

Маштабованість

Чим більший датасет – тим більша якість генерації.

12

Недоліки Transformers

Складність генерації

Через специфіку будови моделі складність генерації становить $O(n^2)$

Високі вимоги до відеопам'яті

Великі музичні моделі побудовані на архітектурі трансформерів потребують захмарних 40+ GB VRAM

Потребує великих датасетів

Для отримання гарних результатів в генерації потрібні великі датасети

13

Датасет Maestro

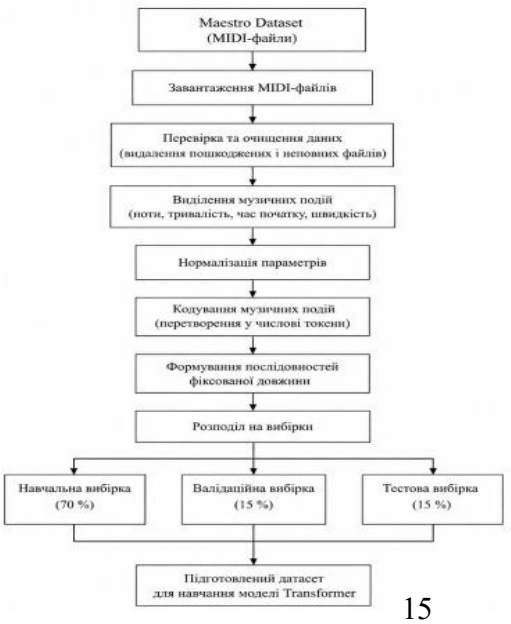
Розділення	Композицій	Тривалість	Розмір (Гб)	Нотатки (мільйонів)
Train	962	159.2	96.3	5.66
Validation	137	19.4	11.8	0.64
Test	177	20.0	12.1	0.74
Total	**1276**	**198.7**	**120.2**	**7.04**

MAESTRO (MIDI та аудіо, відредаговані для синхронних треків та організації) – це набір даних, що складається приблизно з 200 годин віртуозних фортепіанних виступів, записаних з точним вирівнюванням (~3 мс) між мітками нот та аудіоформатами.

14

Підготовка датасету для навчання

Перед навчанням датасет треба змінити під моделі Transformers. У таблиці розписано які саме кроки було проведено. Це стандартна процедура для будь-якого датасету.



15

Чому саме Transformers модель

Ефективне моделювання довгострокових залежностей

Завдяки механізму самоуваги (*self-attention*) модель аналізує всю музичну послідовність одночасно, а не послідовно. Це дозволяє враховувати взаємозв'язки між музичними подіями незалежно від відстані між ними, забезпечуючи структурну, гармонічну та ритмічну цілісність довгих творів.

Вища точність символьної генерації (MIDI)

На відміну від варіаційних автоенкодерів (VAE), де дрібні деталі композиції можуть втрачатися під час реконструкції через особливості латентного простору, Transformer забезпечує максимальну деталізацію та точність побудови складних багатоголосних патернів.

16

Результати роботи

17

Параметри моделі

Параметр	Значення
Архітектура	Transformer
Розмір пакета (Batch Size)	16
Кількість епох	25
Оптимізатор	Adam
Швидкість навчання	0,0001
Функція втрат	Cross EntropyLoss
Розмірність Embedding	512
Кількість голів уваги	8
Кількість шарів Transformer	6

18

Вхідні параметри

```
MIDI_FOLDER = "maestro_midi"
MODEL_PATH = "music_transformer_model.pth"
TOKENIZER_PATH = "tokenizer.pkl"
OUTPUT_MIDI = "generated_music.mid"

SEQ_LENGTH = 128
BATCH_SIZE = 32
EPOCHS = 100
LEARNING_RATE = 0.0001

EMBEDDING_SIZE = 512
NUM_HEADS = 8
NUM_LAYERS = 6
DROPOUT = 0.1
```

19

Перетворення музичної послідовності на послідовність токенів

Музична подія	Код
Note_C4	15
Note_E4	23
Note_G4	31
Duration_Quarter	102
Duration_Half	108
Time_Shift	205

20

Епохи

Епоха	Training Loss	Validation Loss	Validation Perplexity
1	5.854	5.12	167.33
2	4.912	4.451	85.71
3	4.305	3.98	53.51
4	3.821	3.612	37.04
5	3.45	3.25	25.79
6	3.12	2.98	19.68
7	2.915	2.795	16.36
8	2.74	2.65	14.15
9	2.61	2.545	12.74
10	2.51	2.485	12.0
11	2.385	2.39	10.91
12	2.245	2.31	10.07
13	2.15	2.245	9.44
14	2.065	2.19	8.93
15	1.98	2.155	8.62

Епоха	Training Loss	Validation Loss	Validation Perplexity
16	1.89	2.115	8.28
17	1.825	2.09	8.08
18	1.76	2.08	8.0
19	1.695	2.085	8.04
20	1.62	2.095	8.12
21	1.55	2.11	8.24
22	1.485	2.13	8.41
23	1.43	2.155	8.62
24	1.385	2.18	8.84
25	1.35	2.215	9.16

21

Метрики оцінки якості результату.

Назва	Значення
Note Density (Щільність нот)	5.65 нот/сек
FAD (Fréchet Audio Distance)	2.20 — 3.10

22

Висновки

- Досліджено методи та моделі генеративного штучного інтелекту для синтезу музичних творів.
- Розроблено систему автоматизованої генерації музики на основі сучасних технологій глибокого навчання.
- Реалізовано генеративну модель на базі архітектури Transformer для синтезу музичних композицій у символьному форматі MIDI.
- Дослідження підтвердило, що розроблена модель здатна створювати унікальні музичні послідовності без прямого копіювання навчальних фрагментів.

23

Перспективи розвитку

Подальший розвиток роботи полягає в:

- Покращенні та оптимізації моделей на архітектурі Transformer
- Застосуванні гібридних архітектур, наприклад, Transformer та GAN.
- Вдосконаленні початкового датасету.

24

Дякую за увагу

25