ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет Санкт-Петербургская школа физико-математических и компьютерных наук

Денисова Надежда

УЛУЧШЕНИЕ ГЕНЕРАЦИИ СИМВОЛЬНОЙ МУЗЫКИ С ПОМОЩЬЮ ИТЕРАЦИОННОГО АЛГОРИТМА С ДВУСТОРОННИМ КОНТЕКСТОМ НА БАЗЕ АРХИТЕКТУРЫ ТРАНСФОРМЕР

Выпускная квалификационная работа
по направлению подготовки 01.04.02 Прикладная математика и информатика
образовательная программа «Машинное обучение и анализ данных»

Рецензент
__О.А. Свидченко____
И.О. Фамилия

Научный руководитель
д-р ф-м. наук, проф., департамент информатики
____ Б. А. Новиков__
И.О. Фамилия

Консультант
аспирант, НИУ ВШЭ СПб
____И. Н. Лабутин__
И.О. Фамилия

Санкт-Петербург 2023

ОГЛАВЛЕНИЕ

СПИСОК ТЕРМИНОВ И АББРЕВИАТУР

Приведенные определения не являются точными дефинициями из музыкальной теории, а трактуются в соответствии с их использованием в данной работе.

Аккорд – одновременное звучание трех и более нот с разными интервалами между ними.

Долгосрочная структура композиции – связность последующих частей композиции с предыдущими посредством повторений отдельных сегментов и вариаций на них.

Интервал – расстояние между нотами, измеряющееся в полутонах.

Мажор – набор нот с заданными интервалами между ними, использующийся для создания музыкальной композиции; слушателями характеризуется как веселый и светлый.

Мелодия – одноголосная последовательность нот, выделяющаяся на фоне аккордов в музыкальной композиции.

Минор – то же, что и мажор, но характеризуется ощущениями грусти и печали.

Полифония – многоголосие; в данной работе используется для обозначения композиций или их частей, где одновременно звучат мелодия и аккорды.

Такт – отрезок музыкальной композиции, начинающийся с сильного удара и заканчивающийся перед следующим равным ему по силе.

Шестнадцатая – нота, по длине звучания равная 1/4 удара, или 1/16 такта, состоящего из четырех ударов, что является наиболее частой размерностью в популярной музыке.

MIDI – протокол записи, хранения и передачи музыкальной информации.

ВВЕДЕНИЕ

Задача генерации музыки с помощью компьютерных технологий зародилась еще в середине 20 века с публикацией первого алгоритма для ее решения с помощью марковских цепей. Однако в последние несколько лет число публикаций на тему генерации музыки уже нейросетевыми подходами особенно активно растет.

Генерация музыки делится на аудио и символьный подходы. Генерация аудио позволяет сохранить реалистичность исполнения композиций на основе записей в обучающей выборке, однако подвержена проявлению артефактов (шумов) и сложностям с уменьшением размерности пространства входных аудио-записей. Генерация символьной музыки, в свою очередь, хоть и уступает в “живости” звучания, но имеет свои преимущества в виде значительно меньших затрат памяти на данные, их меньшей размерности, а также меньшего времени на инференс.

Долгое время основной использовавшейся моделью оставалась LSTM, однако с появлением Трансформеров начала уступать им в популярности, так как Трансформеры способны принимать на вход всю композицию целиком, а также сохранять больше важной латентной информации о данных с помощью механизма самовнимания. Существует множество подходов, объединяющих Трансформеры с такими архитектурами, как GAN или VAE, что позволяет улучшать качество результатов генерации и добавлять в процесс дополнительные условия, например стиль или настроение желаемой композиции.

Однако ни одна из предложенных моделей в опубликованных на данный момент научных работах на тему генерации символьной музыки не позволяет итеративно изменять результат генерации модели, давая ей при этом возможность “видеть” не только префикс, но и суффикс данных. Такой подход интересен для исследования, так как согласно исследованиям на основе глубинного интервью (Bennet, S., 1976; Roels, H., 2016), в

процессе работы композиторов, как правило, присутствует этап создания черновика музыкальной композиции, а затем этап его исправления, которое можно охарактеризовать как итеративное. Другими словами, имея черновую композицию, автор видоизменяет определенные ее части, уже имея на руках начало (префикс) и конец (суффикс) его работы. Изменение отдельных кусочков композиции может также приводить к аугментации материала, идущего далее, с целью сохранения структуры и гармонии.

Таким образом, целью данной работы является создание такой итеративной модели с двусторонним контекстом на базе являющейся на данный момент state-of-the-art архитектуры Трансформер.

Задачи выделены следующие:

1. Разработка архитектуры модели
2. Имплементация модели и обучение с разными размерами окна
3. Проведение качественного анализа полученных результатов
4. Проведение слушательского теста для сравнения моделей между собой и с бейзлайновой архитектурой.

Данная работа имеет следующую структуру. В Главе 1 проводится обзор литературы по теме генерации музыки в целом и символьной музыки в частности, выделяются основные модели, их архитектуры и применение, а также наиболее популярные форматы представления музыкальных композиций в виде событий, описываемых дискретными целочисленными токенами. В Главе 2 описана методология работы: модель Music Transformer, взятая в качестве бейзлайна и базовой архитектуры, модель Transcormer, основанная на моделировании естественного языка с помощью скользящего окна, а также комбинация этих двух подходов сообразно цели работы. В Главе 3 представлены результаты: процесс обучения и метрика точности, качественный анализ визуализаций сэмплов моделей, а также результаты слушательского теста.

1. ОБЗОР ЛИТЕРАТУРЫ

1.1. Введение в генерацию музыки

Идея генерации музыки с помощью возможностей компьютера была задокументирована еще в 19 веке, когда известный английский математик Ада Лавлейс в 1848 оставила в своих заметках запись, в которой размышляла о том, как аналитическая машина Бэббиджа в теории могла бы создавать оригинальные музыкальные произведения, если бы законы гармонии и композиции в музыке стало возможным перевести в подходящий для обработки машиной формат (Menabrea & Lovelace, 1848). Однако первая модель генерации музыки появилась только середине 20 века, когда Hiller Jr & Isaacson (1957) представили свой алгоритм основанный на марковских цепях. С зарождением и, в настоящее время, активным развитием нейросетей и сферы глубокого обучения в сфере генерации музыки стали появляться все новые и новые исследования.

Civit et al. (2022) отмечают, что с 2017 по 2022 происходил заметный рост количества публикаций на тему генерации музыки с небольшим снижением в 2021 году и предсказывают, что в дальнейшие годы интерес к этой теме продолжит расти. Согласно их исследованию, публикации имеют широкое разнообразие в их географическом происхождении, а также пишутся как в академической сфере, так и в частной, включая такие компании как Google, Open AI и Amazon. Задачи, под которые обучаются модели варьируются от генерации монофонической мелодии, полифонического аккомпанемента, полифонической гармонизации и партии ударных до генерации полноценных композиций со всеми вышеперечисленными элементами или заполнения недостающего кусочка уже имеющейся композиции или каких-то двух композиции в разных жанрах, как, например JukeBox от OpenAI (Dhariwal et al., 2020). Кроме того, существуют исследования, авторы которых создали плагины для

Digital Audio Workstations (DAWs) – программного обеспечения, позволяющего создавать музыкальные произведения.

Генерация музыки делится на два вида в зависимости от формата входных и выходных данных: генерация аудио и генерация символьной музыки. Генерация аудио производится на основе входных данных в виде сырого аудио (в противовес спектрограммам), и основными представителем такого подхода являются модели WaveNet (Dieleman et al., 2016, Dieleman et al., 2018) и JukeBox (Dhariwal et al., 2020). Dieleman et al. (2018) отмечают, что преимущество такого подхода в сохранении отличительных особенностей исполнения (например, обертоны инструмента или миллисекундные различия в начале и окончании звучания отдельных нот), влияющих на восприятие людьми музыкальности композиции и ее естественности для слуха.

Аудио данные имеют высокую размерность, так как длина аудиофайла есть произведение его частоты дискретизации на длину, при этом первый множитель чаще всего равен 44.1kHz, а потому размерность четырехминутной композиции достигает длины около 10 миллионов значений, содержащих 16 бит информации (Dhariwal et al., 2020). Такая размерность входных данных приводит к высокой нагрузке на вычислительные ресурсы в процессе обучения моделей генерации аудио музыки. В силу этого, основной моделью, задействованной в таких генераторах является VQ-VAE (Razavi et al., 2019), способная проецировать входной сигнал в латентное пространство сильно меньшей размерности, сохраняя при этом максимум информации об объекте.

Однако генерация аудио все же имеет и значительные недостатки: так, во-первых, из-за перехода в пространство меньшей размерности невозможно обойтись без потерь информации, во-вторых в результирующих аудио часто встречаются артефакты (шумы), в-третьих, процесс генерации занимает длительное время (например, JukeBox

(Dhariwal et al., 2020) требует час на генерацию, а затем еще восемь часов на повышение размерности результата (upsampling)). Кроме того, при генерации музыки в аудио формате не представляется возможным сделать незначительные изменения в готовой композиции, такие как смена высоты или громкости отдельных нот, что потенциально делает этот подход сложным для использования в процессе работы композитора.

Символьная музыка, в отличие от аудиозаписей, представляет собой высокоуровневое представление нот, абстрагирующее детали отдельного исполнения каждой композиции. Так, данные изначально имеют низкую размерность, что позволяет избежать необходимости их проецирования в пространства меньшей размерности, тем самым уменьшая риск потери информации в процессе обучения. Самый распространенный формат представления символьной музыки в исследованиях основан на протоколе MIDI[1], который позволяет хранить длительность и высоту нот, громкость, темп, данные об инструменте, а также поддерживает композиции с несколькими дорожками для разных инструментов. Все это делает MIDI-файлы небольшими по весу, а также дает возможность легко аугментировать композицию посредством изменения одной или нескольких нот. Данные преимущества сделали символьную музыку значительно более популярным форматом для генерации музыки, чем аудио (Civit et al., 2022).

1.2. Существующие подходы к генерации символьной музыки

С выхода первой научной статьи по генерации музыки с помощью компьютерных технологий в 1957, фокусом которой были модели марковских цепей (Hiller Jr & Isaacson, 1957), произошло зарождение и бум нейросетей, которые на данный момент являются “state-of-the-art”

[1] Musical Instrument Digital Interface (https://midi.org/specifications)

подходом в задачах генерации. Ввиду этого, в данном разделе фокус будет именно на нейросетевых подходах.

Так как символьное представление музыки является последовательностью событий, представляемых дискретными значениями (токенами), наиболее очевидным и естественным подходом к задаче генерации в данном случае оказываются авторегрессионные модели, которые и являются наиболее популярным решением в литературе (Civit et al., 2022).

Модели, основанные на архитектуре Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) оказались наиболее часто применяемыми за 2017-2021 года. Помимо вектора скрытого состояния (hidden state), присутствующего в классическом представлении рекуррентной нейронной сети (RNN), в LSTM есть также вектор памяти, итеративно обновляющийся при подаче на вход каждого следующего токена. Такое архитектурное решение позволило модели “запоминать” информацию о входных данных для более длинных последовательностей, чем у классической RNN, подверженной быстрому “забыванию” информации при каждом обновлении вектора скрытого состояния. Это является важным достоинством модели при работе с генерацией музыки, в которой важно поддерживать общую структуру композиции, опираясь на уже имеющийся префикс.

Chu et al. (2016) применили LSTM к задаче генерации символьной музыки, использовав ее как основной блок для построения иерархической рекуррентной нейронной сети. Авторы обучили модель генерировать поп-музыку, разделив процедуру на несколько слоев: слой для высоты ноты, ее времени и удержания нажатия, аккордов и партии ударных, где первые два отвечают за мелодию, а вторые два, получая информацию от слоев мелодии, работают по принципу условной генерации. Так, данный подход по итогам человеческой оценки результатов генерации показал, что участники чаще выбирают музыку, созданную предложенным методом, по

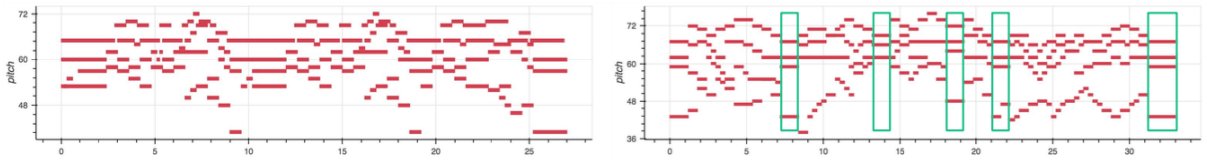
сравнению с музыкой, сгенерированной моделью в рамках проекта Google Magenta[2] (Waite et al.).

Однако LSTM имеет и недостатки, которые приводят к значительному сокращению ее использования в данной задаче и переходу на более современные модели. Так, из-за того что токены на вход подаются последовательно, модель время, требующееся на обработку одной последовательности становится ее слабым местом в случае длинных последовательностей токенов. Кроме того, хоть механизм памяти LSTM работает лучше классической RNN, в сравнении с более современными авторегрессионными архитектурами, например, Трансформерами (Vaswani et al., 2017), она проигрывает в сохранении структуры композиции из-за “забывания” входной информации (Huang et al., 2018).

Трансформеры, в свою очередь, активно набирают популярность в задаче генерации музыки, все чаще и чаще появляясь с научных публикациях в качестве основной модели или ее части (Civit et al., 2022). Их основные преимущества заключаются в том, что, во-первых, они обрабатывают всю последовательность за один проход вместо n (где n – длина последовательности), вместо этого получая информацию о позициях токенов черезе позиционные векторы (эмбеддинги). Во-вторых, они включают в себя блоки с механизмом самовнимания и остаточной связью, что позволяет модели сохранять намного больше важной для генерации информации о входной последовательности, чем в случае рекуррентных сетей. Так, например, в 2018 году была предложена модель Music Transformer (Huang et al., 2018), которая по результатам человеческой оценки сгенерированных примеров, превзошла LSTM, а также и базовый Трансформер. Отличие Music Transformer от базового Трансформера состоит в добавлении обучаемых векторов относительных позиций токенов в композиции (Shaw et al., 2018). Такой подход позволил модели

[2] https://magenta.tensorflow.org

выделять важность токенов на определенных позициях, в результате чего в сгенерированных моделью примерах начали прослеживаться повторяющиеся элементы и вариации на них через определенные отрезки времени, что присуще музыке, создаваемой человеком. Так, на Изображении 1 (Huang et al., 2018) представлены две случайным образом сгенерированные композиции: моделями Трансформер (слева) и Music Transformer (справа). Из визуализаций заметно, что, благодаря добавленным репрезентациям относительных позиций, в примере Music Transformer появляются повторяющиеся схожие сегменты (выделено зеленым) на протяжении композиции. Авторы отмечают, что такой подход делает музыку более похожей на создаваемую людьми, так как подобные повторяющиеся мотивы и вариации на них являются одним из основополагающих элементов общей структуры композиции.



Изображение 1: Примеры композиций, сгенерированных Трансформером (слева) и Music Transformer (справа) посредством безусловной генерации (в отсутствии подающегося на вход фиксированного кусочка композиции для ее продолжения моделью).

В отличие от классического Music Transformer, генерирующего только фортепианную партию, Donahue et al. (2019) предложили модель LakhNES, которая также строится на его архитектуре, но уже способна создавать мульти инструментальные композиции, где каждый отдельный инструмент все так же исполняет полифоническую музыку. Другой известной трансформерной моделью является MuseNet[3] от OpenAI (Payne, 2019), в которую авторы исследования добавили токены, содержащие информацию о композиторе и инструментах. Основанная на архитектуре GPT-2 (Radford et al., 2018) и имеющая больше миллиарда обучаемых параметров, модель

[3] https://openai.com/research/musenet

способна создавать музыкальные композиции в разных жанрах, а также в стиле известных композиторов.

Однако во многих публикациях Трансформер используется также и в комбинации с другими архитектурами. Так, Muhamed et al. (2021) предложили объединить Трансформер с архитектурой Generative Adversarial Network (GAN). GAN-модели строятся на основе двух отдельных нейросетей – генератора и дискриминатора, – совместное обучение которых с помощью минимаксной функции потерь заставляет дискриминатор учиться отличать реальные примеры от сгенерированных, а генератор – “обмануть” его создавая такие примеры, где дискриминатор будет ошибаться (Goodfellow I., 2014). Предложенная Трансформер-GAN модель состоит из Transformer XL в качестве генератора и предобученный SpanBERT (Joshi et al. 2020) в роли дискриминатора. По результатам исследования Трансформер-GAN показал улучшения в качестве генераций по сравнению с трансформенными моделями, обученными только путем оптимизации лог-правдоподобия (Music Transformer и Transformer-XL).

Помимо объединения с GAN-архитектурой, Трансформеры в литературе также часто представлены в композиции с вариационными автоэнкодерами (VAE). Так как задача энкодера VAE заключается в сжатии входных данных с целью их проекции в пространство меньшей размерности, он выучивает фундаментальные характеристики распределения обучающей выборки с целью максимально точного воссоздания входных данных через декодер. Кроме того, в латентном пространстве композиции с разными отличительными особенностями (наример, жанр или стиль) как правило образуют отдельные кластеры, которые впоследствии могут помочь при генерации композиций с желаемыми параметрами. Так, Wu & Huang (2021) предложили модель, в которой в качестве энкодера и декодера выступает архитектура Transformer-XL (в первом случае энкодер двустороннего контекста, во втором – авторегрессионный декодер), а обучение

производится с добавлением условий до, внутри и после механизма внимания и использованием функции потерь VAE в энкодере. Такой подход позволяет задавать параметры композиции такие как ритм и полифоничность вплоть до каждого отдельного такта.

Подход к обучению на основе вариационных автоэнкодеров используется также и с LSTM. Например, известная модель MusicVAE[4] от Google (Roberts et al., 2018), созданная в рамках проекта Magenta, использует в качестве энкодера и декодера иерархическую LSTM.

1.3. Представления MIDI-файлов в качестве входных данных

Как уже было отмечено ранее, наиболее распространенным стандартом записи символьной музыки является MIDI. Однако в своей исходной форме файлы .midi содержат команды: когда и как долго должна проигрываться каждая нота и какой инструмент используется. Такой формат не позволяет передавать .midi файлы напрямую в модель, из-за чего исследователями были разработаны разные способы кодирования этой информации в токены.

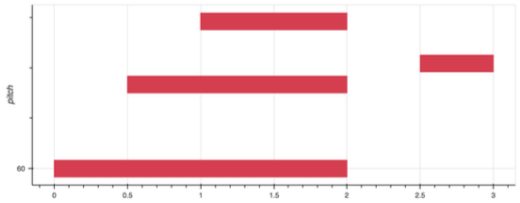
Oore et al. (2018) предложили MIDI-подобный способ перевода файлов в токены словаря нотных событий размера 413:

- 128 событий начала звучания ноты, описываемых их высотой
- 128 событий окончания звучания ноты, описываемых их высотой
- 125 событий сдвига по времени от последнего зарегистрированного момента времени; каждый последующий токен добавляет к сдвигу 8 мс, максимальное значение – 1 секунда.
- 32 события громкости, квантизирующие 128 событий громкости MIDI по 32-м отрезкам, где каждое событие распространяется на все последующие ноты до появления следующего события громкости.

[4] https://magenta.tensorflow.org/music-vae

Однако так как авторы используют RNN, каждое событие кодируется в виде one-hot вектора, из-за чего последовательность событий для 15 секунд музыкальной композиции занимает около 600 векторов длины 413.

Huang et al. (2018) адаптировали описанный подход под Music Transformer: теперь события представляют собой не one-hot векторы, положительные целочисленные значения на отрезке [0, 388]. События сдвига по времени авторы дискретизируют по 10 мс, поэтому размер словаря сокращается на 25 событий. На Изображении 2 (Huang et al., 2018) показан пример перевода MIDI событий в виде нот на визуализации в формат токенов.



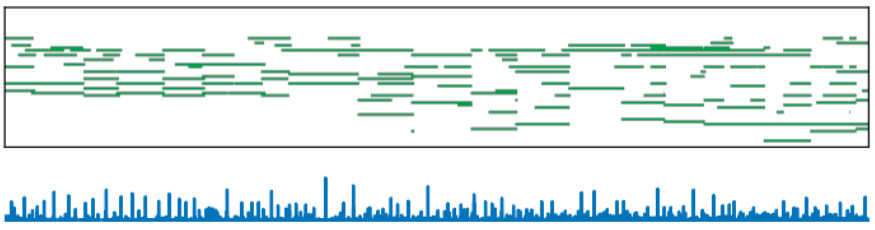
```
SET_VELOCITY<80>, NOTE_ON<60>
TIME_SHIFT<500>, NOTE_ON<64>
TIME_SHIFT<500>, NOTE_ON<67>
TIME_SHIFT<1000>, NOTE_OFF<60>, NOTE_OFF<64>,
NOTE_OFF<67>
TIME_SHIFT<500>, SET_VELOCITY<100>, NOTE_ON<65>
TIME_SHIFT<500>, NOTE_OFF<65>
```

Изображение 2: визуализация MIDI-событий в виде партии для фортепиано, где по оси *y* отложено время, а по оси *x* – высота нот (слева) и их представление в виде токенов (справа).

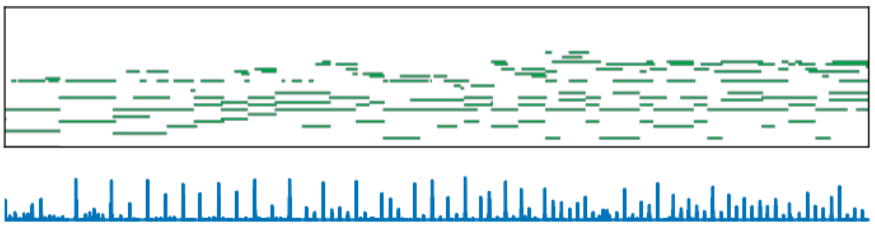
Еще одним способом создания токенов композиции является Revamped MIDI-Derived Events (REMI) (Huang & Yang, 2020). Мотивацией для предложенного подхода стало отсутствие в MIDI-подобных представлениях явной информации о ритме и тактах, в терминах которых мыслят и пишут музыку реальные композиторы. Авторы исследования заменяют события окончания звучания ноты на события длительности, представленные в виде количества шестнадцатых, так как в MIDI-подобных представлениях события начала и окончания звучания ноты часто разделены несколькими десятками событий, что приводит к генерации композиций с нотами, которым модель не сопоставила момент окончания. Кроме того, на замену сдвига по времени приходят события такта и позиции нот внутри него (в шестнадцатых), что позволяет избежать накапливающихся искажений из-за отдельных ошибок в событиях

временного сдвига. Наконец, добавляются события темпа и аккорда, состоящего из корневой ноты и его характеристики (мажор, минор и др.).

На Изображении 3 (Huang & Yang, 2020) показано, как Music Transformer “теряет” ритмическую структуру в процессе генерации композиции. Синим показаны кривые вероятностей начала нового такта. Видно, что использование REMI токенов помогает сделать эти вероятности более равномерно распределенными во времени.



(a) Transformer-XL × MIDI-like (‘Baseline 1’)



(b) Transformer-XL × REMI (with Tempo and Chord)

Изображение 3: пример результатов генерации классического Music Transformer с MIDI-подобными токенами (сверху) и той же модели с REMI-токенами (снизу). Кривые вероятностей начала нового такта выделены синим.

1.4. Выводы и результаты по главе

Обзор литературы показал, что наиболее частым подходом к генерации символьной музыки являются авторегрессионные модели, среди которых широко используются LSTM и Трансформеры, однако в последние годы активно растет популярность именно вторых, так как они превосходят LSTM в эффективности работы с длинными последовательностями. Авторегрессионные модели также часто используются как элемент

подходов GAN и VAE, где первые позволяют повысить качество генерируемой музыки и ее схожесть с работами реальных композиторов, а вторые чаще всего используются для добавления возможности задания различных параметров, например стиля или ритма. На момент начала написания данной работы модель Music Transformer все еще считается state-of-the-art архитектурой для генерации символьной музыки, а потому именно она берется в качестве бейзлайна и базовой архитектуры.

2. МЕТОДОЛОГИЯ

2.1. Music Transformer

В качестве основной архитектуры и бейзлайна была взята модель Music Transformer (Huang et al., 2018). Она представляет собой заданное количество авторегрессионных декодер-блоков классического Трансформера (Vaswani et al., 2017) с добавлением feedforward слоя, выдающего значения логитов для последующей классификации значения следующего предсказываемого токена по всему словарю. Отличительной особенностью модели являются обучаемые представления относительных позиций токенов входных последовательностей (RPR) в механизм самовнимания (Shaw et al., 2018).

В классическом представлении механизм внимания выглядит следующим образом. На вход подается L векторов скрытых состояний токенов размерности D, затем они проецируются в пространства *Query*, *Key* и *Value* посредством умножения на обучаемые матрицы W^Q, W^K и W^V соответственно, имеющие размерность $D \times D$:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V.$$

Затем полученные матрицы размерности $L \times D$ делятся на H частей размерности $L \times D_h$, где H – количество голов самовнимания, а $D_h = \frac{D}{H}$ – их размерность. Затем *Query*, *Key* и *Value* подают непосредственно в сам механизм внимания:

$$Z^h = Attention(Q^h, K^h, V^h) = \mathrm{softmax}\left(\frac{Q^h K^{h\top}}{\sqrt{D_h}}\right) V^h.$$

Функция софтмакса преобразует входные состояния в распределения плотности вероятностей, таким образом присваивая “значимость” каждому элементу из *Value*. Так, механизм самовнимания позволяет для каждого токена агрегировать информацию обо всех его предшествующих на основе обучаемых коэффициентов их важности для последующих предсказаний.

Наконец, выходные из самовнимания векторы снова объединяются в матрицу Z размерности $L \times D$, проходят через механизм остаточной связи, а затем попадают в feedforward слой, состоящий из двух линейных слоев и одной нелинейности между ними:

$$\mathrm{FF}(Z) = \mathrm{ReLU}(ZW_1 + b_1)W_2 + b_2,$$

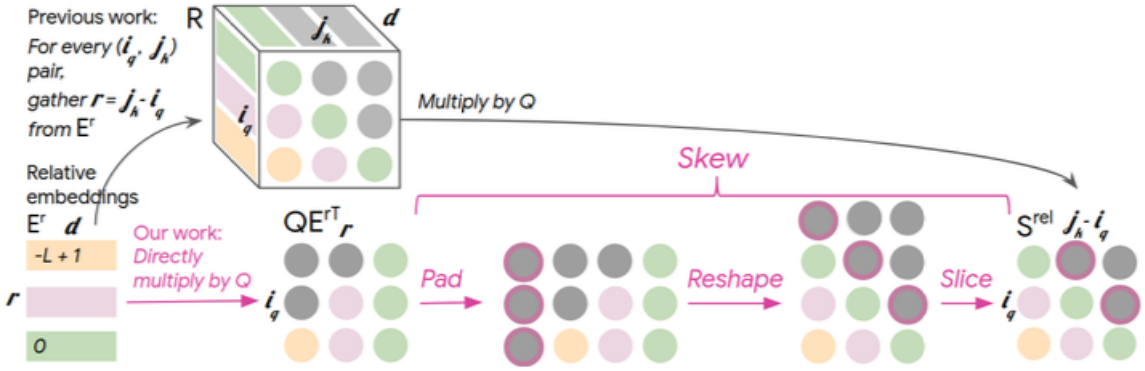
после которого находится еще один блок остаточной связи.

Репрезентации относительных позиций (RPR) представляют собой матрицу S^{rel}, прибавляемую к произведению внутри софтмакса:

$$RelativeAttention = \mathrm{softmax}\left(\frac{QK^\top + S^{rel}}{\sqrt{D}}\right)V.$$

В публикации Shaw et al. (2018) матрица S^{rel} выводится следующим образом. Сначала создается обучаемая матрица E^r размерности $H \times L \times D_h$, содержащая в себе эмбеддинги для всех попарных расстояний $r = j_k - i_q$ между векторами *query* и *key* на позициях i_q и j_k соответственно. Эти эмбеддинги упорядочены по соответствующим им расстояниям от $-L + 1$ до 0 и обучаются отдельно для каждой головы внимания. Затем для каждой головы внимания инициализируется промежуточная матрица R размерности $L \times L \times D_h$, содержащая эмбеддинги относительных позиций. После этого к матрице Q добавляется дополнительная третья размерность ($L \times 1 \times D_h$) и происходит умножение $S^{rel} = QR^T$, результат которого уже добавляется в дробь под софтмаксом.

Однако данный подход требует $O(L^2 D + L^2)$ затрат памяти и ограничивает длину последовательности, которую возможно вместить на ГПУ размером 16Гб при $D_h = 64$ до 650 токенов. С целью смягчения этих ограничений Huang et al. (2018) предложили алгоритм оптимизации по памяти при использовании эмбеддингов относительных позиций. Так, в модель была добавлена “процедура сдвига”, показанная на Изображении 4 (Huang et al., 2018), позволившая ограничиться только инициализацией матрицы E^r.



Изображение 4: “процедура сдвига” в модели Music Transformer с относительным глобальным вниманием. Благодаря этому методу пропадает необходимость создания матрицы R (сверху), требующей $O(L^2 D)$ памяти.

Таким образом, именно в вышеописанной форме модель Music Transformer используется в данной работе как бейзлайн и базовая архитектура для создания итеративного алгоритма с двусторонним контекстом.

2.2. Transcormer

Для создания выбранного алгоритма Music Transformer комбинируется с подходом, реализованным в модели Transcormer (Song et al., 2022). Transcormer – модель из сферы обработки естественного языка (NLP), разработанная для оценки правдоподобия сгенерированных какой-либо другой моделью предложений. Отличительная особенность данного алгоритма заключается в его способности обрабатывать двусторонний контекст за один шаг инференса. Так, предыдущие исследования используют преимущественно либо причинно-следственное языковое моделирование с помощью таких моделей, как GPT (Radford et al., 2018), или моделирование языка через накладывание масок, как, например, BERT (Devlin et al., 2018). Главный недостаток первого подхода заключается в доступности только одностороннего контекста (префиксов последовательностей) для оценки правдоподобия каждого следующего

токена, в то время как второй подход позволяет последовательно накладывать маску на каждый оцениваемый токен, учитывая не только префикс, но и суффикс предложения после него, но требует *n* итераций для получения оценки, где *n* – длина последовательности. Таким образом, целевая функция модели приобретает следующий вид:

$$\mathcal{L} = \sum_{i=1}^{|x|} \log P(x_i | \mathbf{x}_{<\mathbf{i}}, \mathbf{x}_{>\mathbf{i}}; \theta)$$

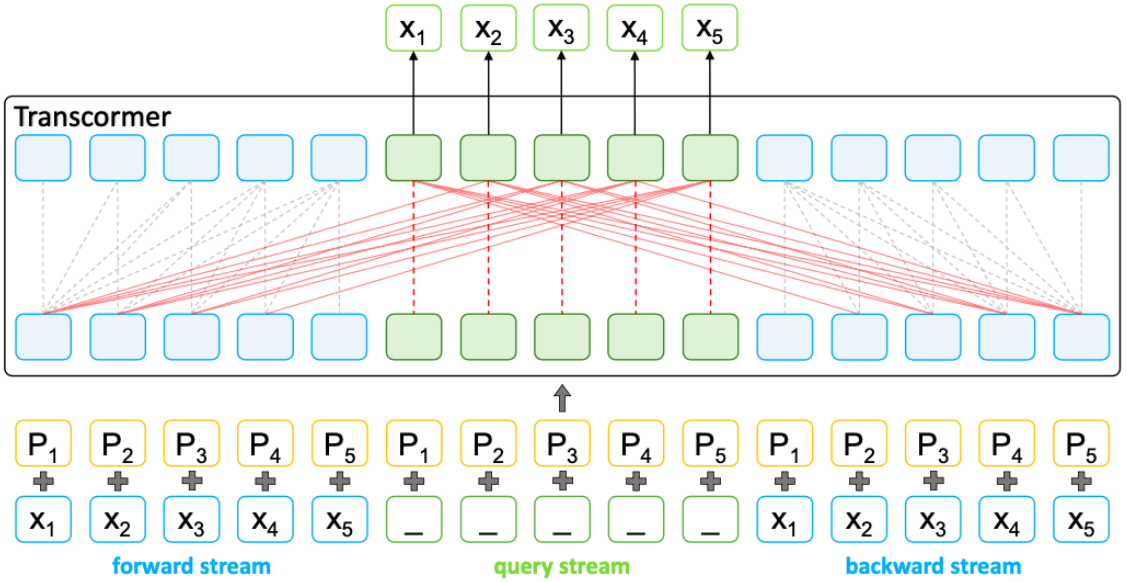
.

Предложенный Song et al. (2022) метод моделирования языка через скользящее окно заключается в модификации механизма самовнимания. Вместо одного входного тензора, состоящего из скрытых состояний каждого токена последовательности в модели появляются три разных стрима: forward, backward и query (Изображение 5, Song et al., 2022). Первый агрегирует в себе информацию о префиксах последовательности, используя стандартную верхне-треугольную мазку из декодера Трансформера. Второй стрим отвечает за суффиксы последовательности, а потому имеет нижне-треугольную маску. Query стрим создан для избежания утечки данных в модели, при которой оцениваемый токен получает информацию не только о его контексте, но и о самом себе. Так, в query агрегируется информация из forward и backward и добавляется дополнительная маска, закрывающая информацию об оцениваемом токене, полученную из обоих стримов (Изображение 6, Song et al., 2022).

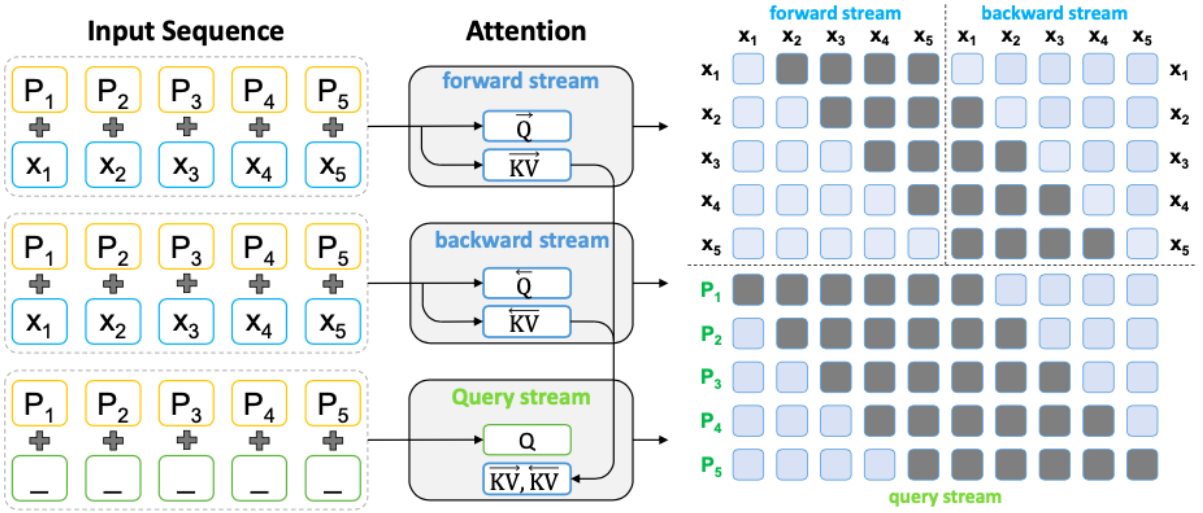
2.3. Итеративная модель с двусторонним контекстом

Модель Transcormer поддерживает двусторонний контекст и позволяет получать результат и вычислять функцию потерь по всем замаскированным токенам за один проход, но, так как она разрабатывалась для задачи

вычисления правдоподобия, а не генерации последовательности, в нее необходимо внести несколько модификаций.



Изображение 5: для последовательности длины 5 показано, какая информация поступает в query стрим из forward и backward стримов без утечки данных.



Изображение 6: на вход forward и backward стримам подаются суммы эмбедингов токенов и позиционных эмбеддингов, на вход query подаются только позиционные эмбеддинги, так как информация о токенах поступает в него из других стримов (слева); матрицы K и V в механизме внимания query получаются путем конкатенации соответствующих матриц из forward и backward стримов (по центру); маски forward и backward представляют собой верхне- и нижне-треугольную маску соответственно, а в query стриме добавляется также маска для каждого оцениваемого токена (справа).

На Изображении 7 показано, как были аугментированы маски стримов. Маска forward осталась неизменной, а на маске backward появилось дополнительное скользящее окно регулируемого размера. Так, задача

модели заключается в предсказании первого (самого левого) токена в окне, но вместо всего суффикса последовательности в “поле зрения” модели попадает только его часть, начинающаяся после маски окна.

В силу того, что задача заключается в предсказании следующего токена, а не оценки текущего, в forward части query стрима была убрана диагональная маска. В части, получаемой из backward стрима диагональная маска была оставлена для удобства визуальной интерпретации архитектуры, несмотря на то, что избавление от нее не привело бы к утечке данных. Маска скользящего окна может иметь минимальный размер в один токен и максимальный размер равный L – 1, где L – длина последовательности, что превратит модель в обычный декодер Трансформера. Целевая функция модифицированной модели выглядит следующим образом:

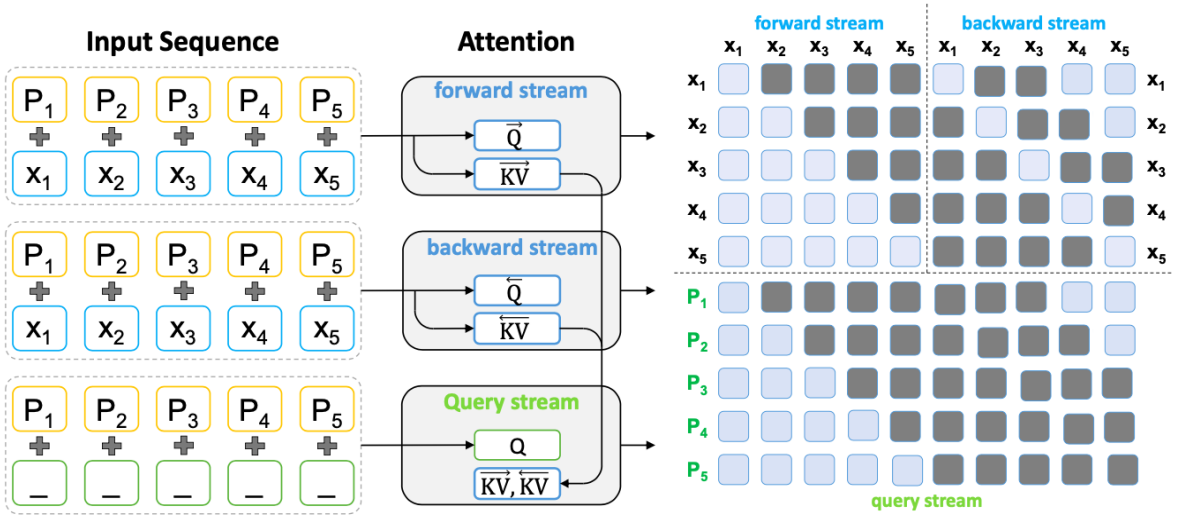
$$\mathcal{L} = \sum_{i=1}^{|x|} P(x_i | \mathbf{x}_{<\mathbf{i}}, \mathbf{x}_{>\mathbf{i+l}}; \theta),$$

где l – размер окна.

2.4. Данные и параметры обучения

Для обучения предложенной модели был взят датасет, состоящий из 21288 .midi файлов, содержащих партии электронного фортепиано. Они поделены на обучающую, валидационную и тестовую выборки в соотношении 80/10/10. В датасете также присутствует информация о жанрах композиций (поп, рок, джаз, блюз, кантри, фолк, реп, классика, соул, латина, электро и “старое”), которые впоследствии переводятся в one-hot векторы размерности 12 и конкатенируются с выходом эмбеддингового слоя Music Transformer перед первым блоком с вниманием. Однако в дальнейшей

работе эти векторы не проверялись на способность контролировать результаты генераций, поэтому глубже рассмотрены не будут.



Изображение 7: предложенная модель по своей структуре, отраженной слева и по центру, повторяет Transcormer, однако в маски стримов вносятся изменения; на изображении показаны маски для последовательности длины пять и скользящего окна длины два токена.

Формат входных данных для модели был выбран MIDI-like, в силу того, что он использовался в оригинальной статье Huang et al. (2018), а исследование влияния входных репрезентаций на процесс генерации музыки выходит за рамки целей и задач данной работы.

В процессе обучения из каждой композиции в датасете берется случайный отрезок длиной в 2048 токенов, который и подается модели. Структура модели следующая: 6 декодер-блоков Трансформера, 8 голов внимания, размерность эмбеддингов и скрытых состояний 512, размерность feedforward слоев – 1024. Словарь состоит из 391 токена: 128 событий начала ноты, 128 событий конца ноты, 100 событий сдвига по времени, 32 события громкости, а также токен конца композиции, паддинга и токен маски, пришедший в Music Transformer из модели Transcormer и заменяющий входные токены query стрима. В финальном классификационном слое токен маски отсутствует, а потом слой имеет размерность 1024×390.

В качестве функции потерь взята кросс-энтропия, а в качестве промежуточной метрики для оценки обучения выбрана точность (accuracy) выбора каждого следующего токена на основе префикса истинных композиций.

В качестве оптимизатора, как и у Huang et al. (2018) был взят Adam, и гиперпараметры обучения следующие: learning rate = 3e-4, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\varepsilon = 1e\text{-}6$. Размеры окна были взяты из множества {1, 4, 8, 16}. Количество эпох было выбрано 20 в силу ограниченности вычислительных ресурсов (длительность одной эпохи варьировалась от 1.5 до 3 часов).

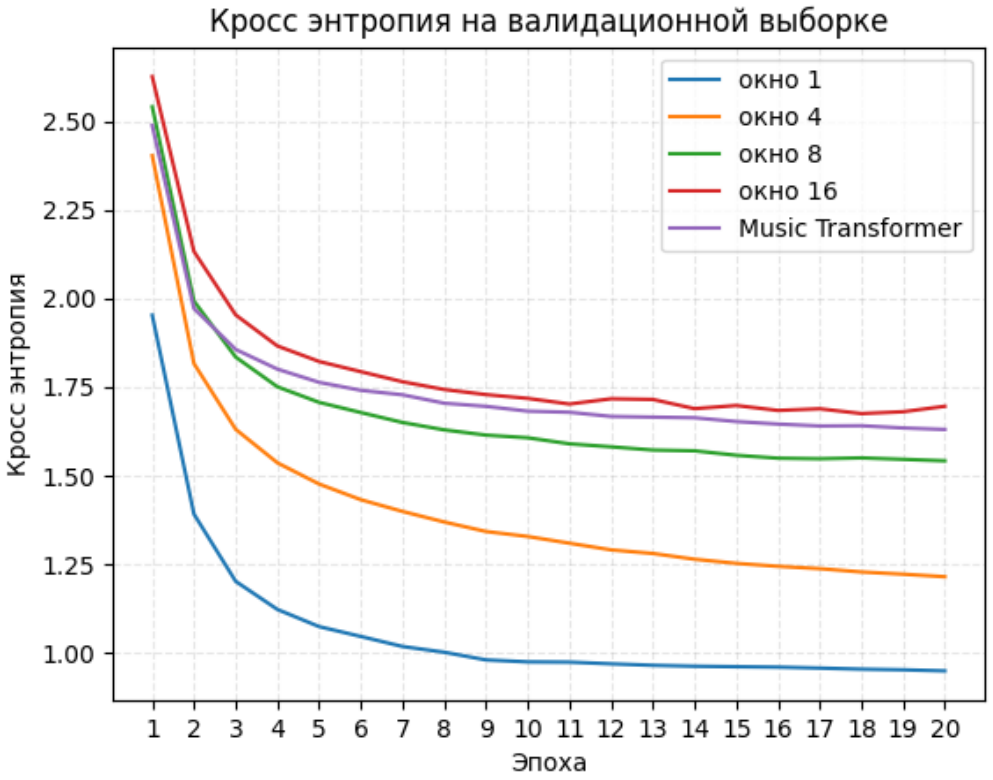
2.5. Выводы и результаты по главе

Таким образом, в данной главе были представлены две нейросетевые архитектуры, комбинация которых легла в основу предложенной модели, а также была описана модификация, производимая с моделью Transcormer для ее соответствия задаче генерации символьной музыки. Особенность предложенная модели заключается в наличии в ее “поле зрения” двустороннего контекста композиции вместо одного префикса, как присуще авторегрессионным моделям. Это достигается с помощью скользящего окна, размер которого задается гиперпараметром, так что модели доступна информация о токенах до и после замаскированного окна, в котором первым токеном является тот, который модель на данном этапе генерирует. Предполагается, что с уменьшением размера окна будет увеличиваться точность генерации, так как модель сможет “видеть” в суффиксе события окончания нот и предсказывать для них соответствующие события начала.

3. РЕЗУЛЬТАТЫ

3.1. Функция потерь и точность

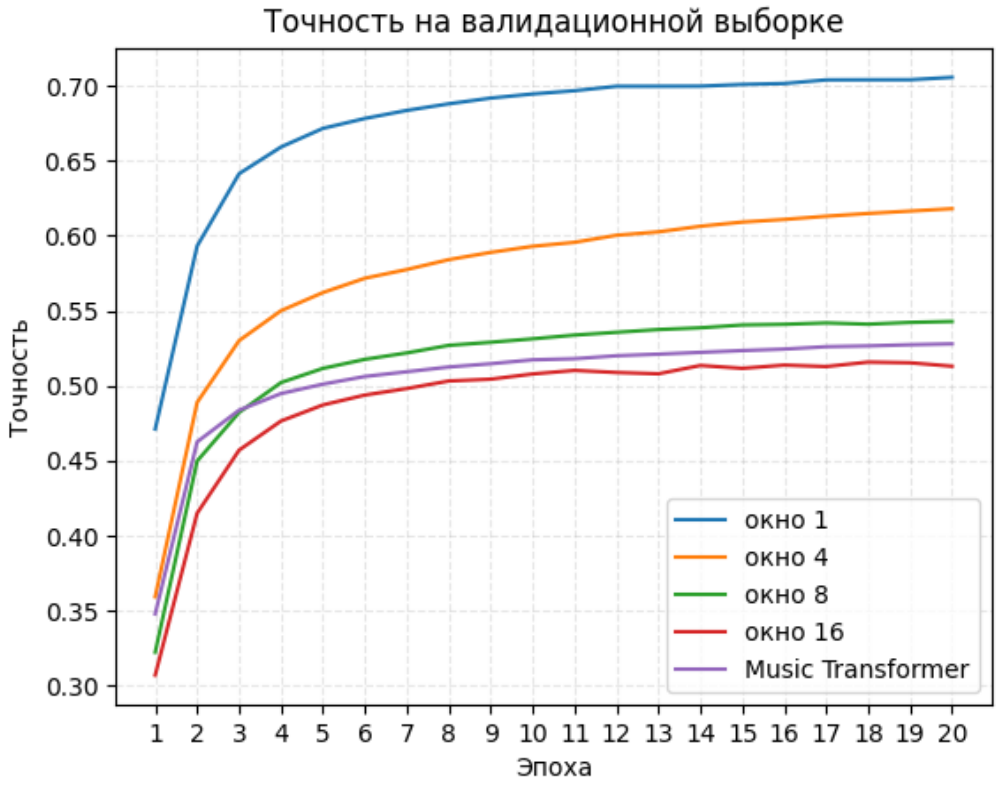
В процессе двадцати эпох значение функции потерь (Изображение 8) на валидационной выборке у большинства моделей стабильно снижалось (немного турбулентно показали себя значения у модели с окном 16), а значение точности (Изображение 9) росло. Хотя функция потерь не является метрикой и не содержит четко интерпретируемой информации о качестве работы модели, следуя по стопам Huang et al. (2018) было решено привести в данной работе эти значения для того, чтобы показать стабильность процесса обучения. В Таблице 1 представлены значения функции потерь и точности на валидационной выборке к концу двадцатой эпохи обучения.



Изображение 8: график зависимости функции потерь на валидационной выборке от эпохи обучении для Music Transformer и четырех моделей с предложенными окнами.

Как и ожидалось, модели с наименьшим размером окна показали наименьшее значение функции потерь и наибольшую точность. Однако интересным результатом оказалось то, что по этим значениям

бейзлайновый Music Transformer обошли все модели, кроме окна 16. Возможно, это обусловлено тем, что в пределы окна размером меньше 16-ти не попадает большинство событий конца звучания нот для тех нот, события начала которых были зарегистрированы до окна. В случае же окна 16 могло случиться так, что оно скрыло собой важные события конца нот, а из-за наличия в модели трех стримов внимания вместо одного обучение стало более длительным и нестабильным, поэтому результаты этой модели не сошлись близко к результатам базового Music Transformer.



Изображение 8: график зависимости точнности (accuracy) генерируемых композиций относительно соответствующих им входных эталонных композиций из датасета для Music Transformer и четырех моделей с предложенными окнами.

3.2. Качественная оценка

Как принято в исследованиях генерации музыки в целом и в статье Huang et al (2018) в частности, сделаем в первую очередь качественный анализ результатов, так как сгенерировать результаты нескольких итераций предложенной модели на всем тестовом множестве не представляется

возможным в силу долгого времени инференса и ограниченности вычислительных ресурсов.

Для проведения качественной оценки и выявления каких-либо закономерностей в работе моделей посмотрим на Изображение 9. Сверху представлена визуализация отрывка из композиции М.П. Мусоргского Intermezzo in modo classico длиной в 2048 токенов. По оси x отложено время, а по оси y – высота звучащих нот.

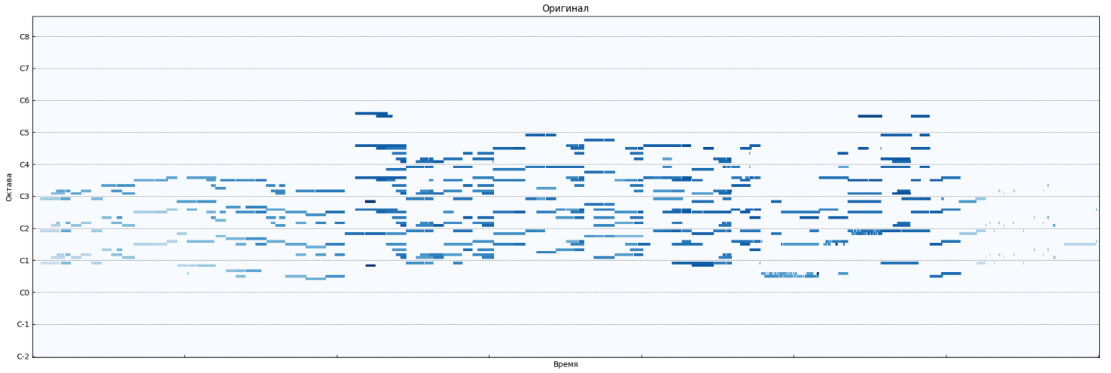
Далее сверху вниз расположены визуализации одной итерации предложенной в данной работе модели с окнами 1, 4, 8 и 16 соответственно.

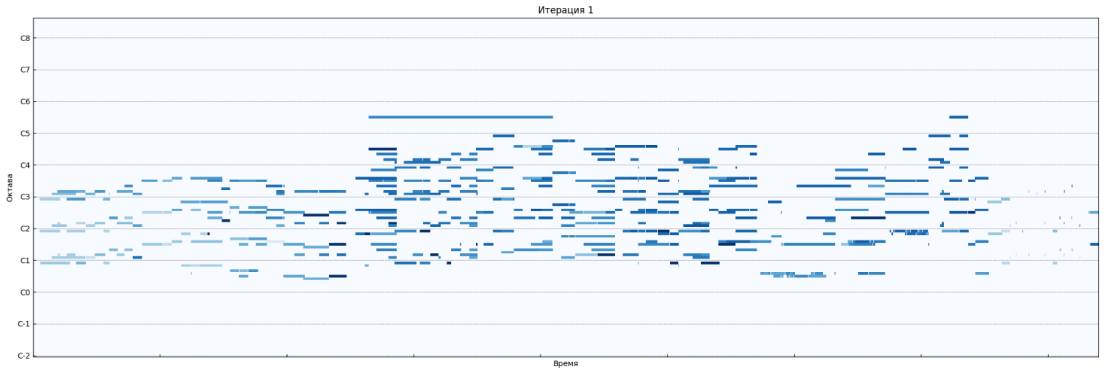
Модель	**CrossEntropyLoss**	**Точность**
Music Transformer	*1.630*	*0.528*
Окно 16	1.696	0.513
Окно 8	1.542	0.543
Окно 4	1.215	0.6181
Окно 1	0.950	0.706

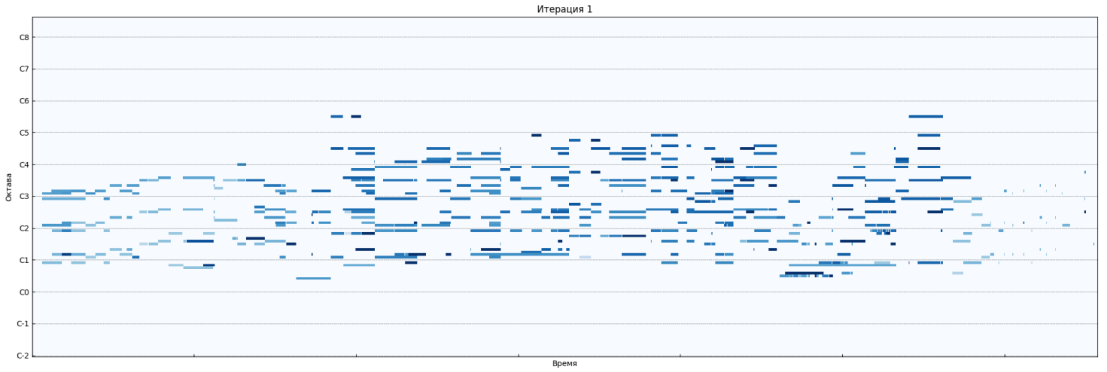
Таблица 1: результаты значений функции потерь и метрики точности на конец двадцатой эпохи для каждой из моделей.

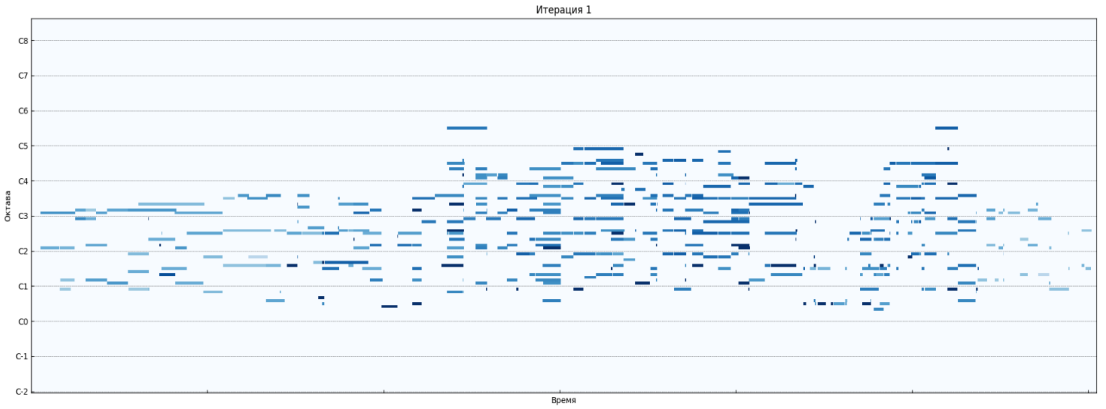
Сразу видно, как модель с окном размера 1 почти полностью сохраняет структуру оригинального отрывка, изменяя только некоторые отдельные ноты по длительности, высоте и громкости. Результат работы модели с окном 4 также зрительно похоже на оригинал, но в нем заметно уже намного больше аугментаций: ноты делятся на кусочки, какие-то убираются, где-то появляются новые. Модели с окнами 8 и 16 уже с первой итерации теряют зрительное сходство с оригиналом. Особенно это видно при использовании окна 16, где композиция становится местами затянутой

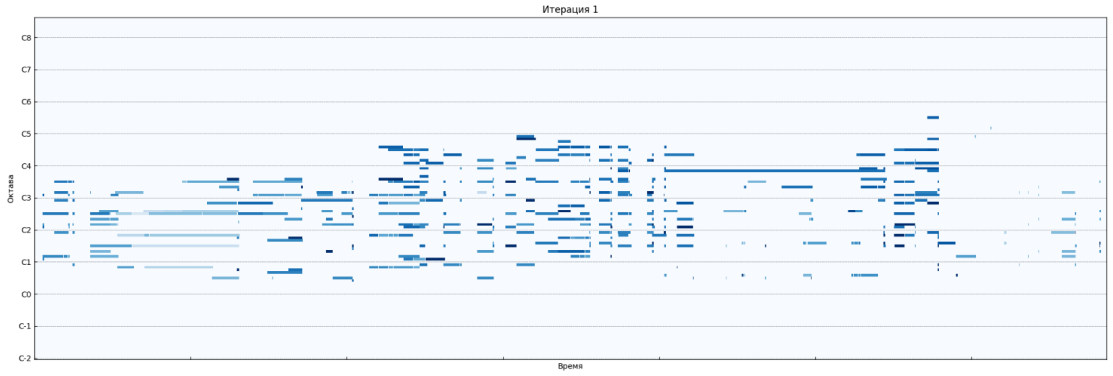
на одной высоте, местами отрывистой, а также начинает пропадать мелодия, а ей на замену приходят аккорды.





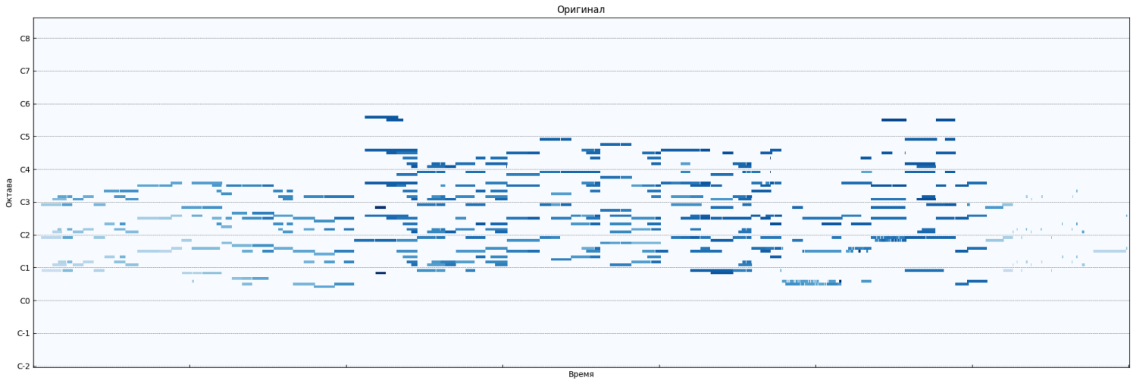


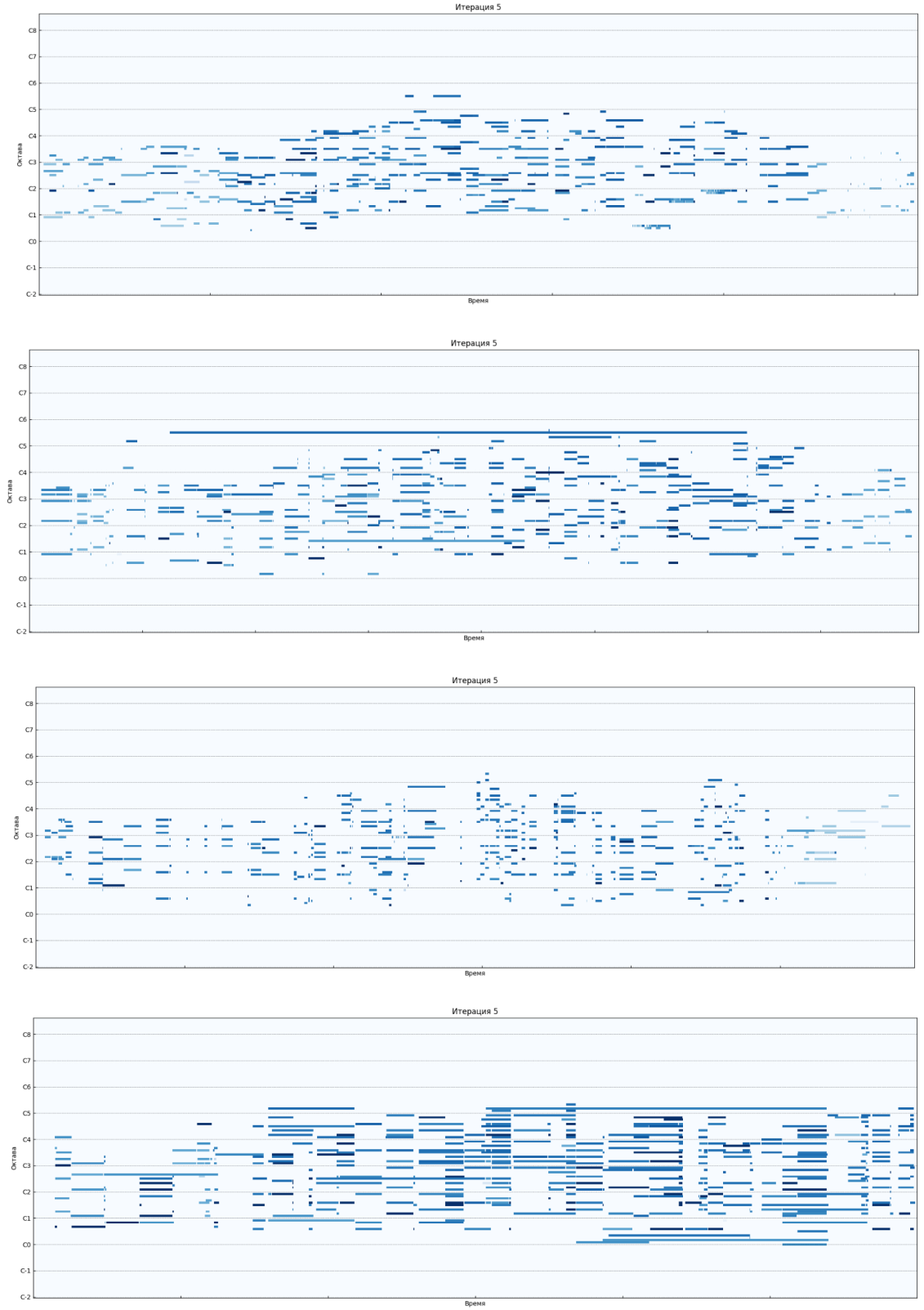




Изображение 9: визуализация результатов одной итерации моделей с окном 1 (вторая строка), 4 (третья строка), 8 (четвертая строка) и 16 (пятая строка) на основе единого отрывка из композиции М.П. Мусоргского (первая строка).

На Изображении 10 представлены в той же последовательности результаты уже пятой итерации всех моделей. Сверху все так же располагается визуализация оригинального отрывка. Видно, что у моделей с окнами 8 и 16 уже к пятой итерации потеряно близкое сходство с оригиналом. У сэмпла из модели с окном 8 присутствуют примерно все те же ноты, что и в оригинале, но они стали редкими и отрывистыми. При использовании окна 16 ноты, наоборот, увеличились во времени звучания, а композиция потеряла выраженную мелодию и превратилась в длительные аккорды. Результаты работы моделей с окнами 1 и 4 по структуре похожи на оригинал, но видно, что модель все же привнесла множество вариаций, а также все еще страдает от растянувшихся нот, “забывая” сопоставить им события окончания звучания, как и базовый Music Transformer.



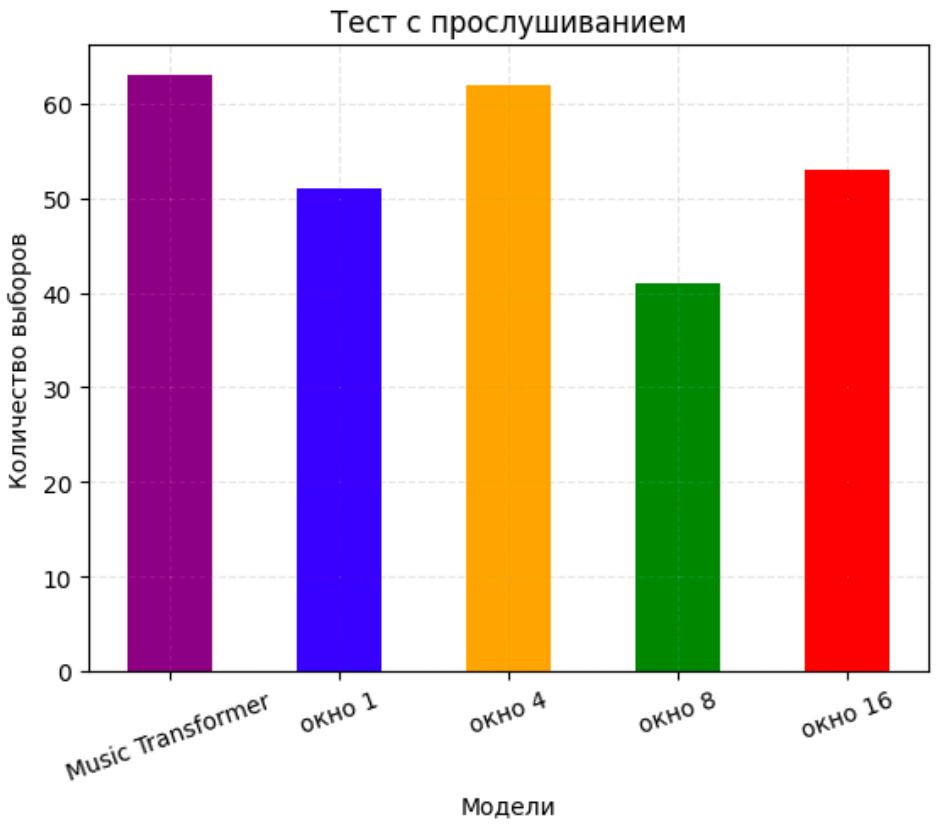


Изображение 10: визуализация результатов одной пяти моделей с окном 1 (вторая строка), 4 (третья строка), 8 (четвертая строка) и 16 (пятая строка) на основе единого отрывка из композиции М.П. Мусоргского (первая строка).

3.3. Слушательский тест

Моделью Music Transformer было сгенерировано 9 случайных композиций. К каждой из них была применена одна итерация модели с окном, где размеры окон брались из множества {1, 4, 8, 16}. Между собой сравнивались только те композиции, которые были сгенерированы на основе одного и того же сэмпла, полученного Music Transformer. Кроме того, результат каждой модели с окном сравнивался не только с моделями с другими размерами окон, но и с сэмплом Music Transformer, что дало по 10 попарных сравнений для каждого такого сэмпла и всего 90 сравнений, учитывая все различные сэмплы. Перекрытие составило 3 респондента на каждую пару. Таким образом, всего было сделано 270 попарных сравнений. Слушателям предлагалось выбрать из двух композиций ту, которую он считают наиболее музыкальной. Музыкальность определялась следующим образом: наличие четкой и связной мелодии, отсутствие многочисленных неуместных повторов отдельных кусков композиции, наличие четкого ритма, гармонически приятное звучание (отсутствие диссонанса) и распределение громкости нот, близкое к по звучанию к сочинениям реальных композиторов (отсутствие неожиданных и нелогичных очень громких или тихих частей).

На Изображении 11 показано количество сравнений, в которых сэмпл, полученный каждой из моделей, был выбран слушателями как наиболее музыкальный. Видно, что базовый MusicTransformer одержал больше всего побед, но в целом значения по каждой модели отличаются не очень разительно. Для проверки наличия статистически значимых отклонений в полученных результатах был проведен биномиальный тест внутри каждой пары сравниваемых моделей по данным их “побед” и “поражений” в проведенных сопоставлениях.



Изображение 11: количество “побед” каждой модели в попарных сравнениях.

В Таблице 2 представлены количества побед и поражений моделей в попарных сравнениях, а также значения p-value на основе одностороннего биномиального теста. Был выбран именно биномиальный тест, так как во время оценки слушателям предлагалось выбрать одну из двух композиций без каких-либо порядковых сравнений между ними. В каждом из них нулевая гипотеза заключается в отсутствии явных предпочтений внутри каждой пары (то есть вероятность выбора любой из двух композиций $p = (1 - p) = \frac{1}{2}$), а альтернативная гипотеза заключается в преимуществе модели в столбце “Модель 1”. Уровень значимости для проверки гипотез был выбран $\alpha = 0.05$.

По результатам тестов, нельзя утверждать о статистически значимых различиях в предпочтении слушателями сэмплов ни одной из модели из почти всех сравниваемых пар; единственное сравнение, показавшее статистически значимый результат с p-value = 0.026 < 0.05, оказалось в

паре между Music Transformer и моделью с окном 8. Так, в данной паре композиции, полученные Music Transformer, оказались наиболее предпочтительными для слушателей.

Модель 1	Модель 2	Число побед	Число поражений	p-value
Music Transformer	Окно 1	15	12	0.351
	Окно 4	14	13	0.5
	Окно 8	19	8	0.026*
	Окно 16	16	11	0.221
Окно 1	Окно 4	11	16	0.876
	Окно 8	17	12	0.124
	Окно 16	11	16	0.876
Окно 4	Окно 8	16	11	0.221
	Окно 16	16	11	0.221
Окно 8	Окно 16	15	12	0.351

Таблица 2: попарные сравнения результатов моделей после слушательского теста, где p-value показывает вероятность получения таких же или более экстремальных значений числа побед моделей столбца “Модель 1” при условии верности нулевой гипотезы.

3.4. Выводы и результаты по главе

В данной главе были представлены результаты проведенного исследования, а именно значения точности и функции потерь для всех выбранных моделей к концу финальной эпохи обучения, визуализация и качественный анализ результатов работы моделей на отрывке из оригинальной

композиции М.П. Мусоргского, а так приведены результаты слушательского теста и их анализ.

По результатам можно сделать вывод, что предложенная итеративная модель со скользящим окном не внесла заметных улучшений в качество работы базовой модели Music Transformer. Анализ визуализаций на первой и пятой итерациях моделей с окнами размера 1, 4, 8 и 16 показали, что меньший размер окна позволяет лучше сохранить схожесть со структурой исходной композиции, внося в нее незначительные вариации в разных местах, как и предполагалось для предложенной в работе модели. Результаты слушательского теста не показали статистически значимых различий в количестве выбора респондентами сэмплов из различных моделей, за исключением сравнения Music Transformer и модели с окном 8, где первый оказался более предпочтительным. Однако этот результат все равно показывает достаточно высокое значение p-value и вполне может оказаться статистически незначимым при увеличении числа респондентов теста. Таким образом, нельзя сказать о том, что предложенная модель работает лучше или хуже по сравнению с Music Transformer. Однако то, что она производит вариации на отдельные кусочки композиции или всю ее целиком, производя при этом примеры статистически не хуже бейзлайновой модели, может говорить о ее потенциальном применении как дополнительный инструмент в процессе работы реальных композиторов.

ЗАКЛЮЧЕНИЕ

По результатам проведенной работы можно сделать следующие выводы. Во-первых, предложенная модель не смогла избавиться от проблемы “бесконечных” нот, присущей Music Transformer даже в использования случае окна 4. Во-вторых, отсутствие статистически значимой разницы в предпочтениях результатов генерации той или иной модели при попарных сравнениях хоть и показывает, что предложенная модель не улучшает качество генерируемой музыки, но при этом и не говорит о том, что качество ухудшается. При этом по итогам качественного анализа видно, что предложенная модель создает вариации на отдельные кусочки или всю композицию целиком (в зависимости от выбранного размера окна), а потому в теории может использоваться композиторами для поиска новых идей по развитию черновика композиции. В-третьих, размер выборки респондентов и количество отобранных сэмплов для слушательского теста все же кажутся недостаточными для получения каких-либо точных результатов и данных о качестве работы моделей.

В качестве дальнейших путей развития исследования в области итеративной генерации с учетом двустороннего контекста композиции можно попробовать заменить MIDI-подобные представления токенов на REMI, так как они более приближены к реалиям сочинения музыки композиторами, а также по результатам исследования Huang & Yang, 2020 показывают улучшенные результаты по результатам оценки респондентов.

СПИСОК ЛИТЕРАТУРЫ

Chu, H., Urtasun, R., & Fidler, S. (2016). Song from PI: a musically plausible network for pop music generation. arXiv preprint arXiv:1611.03477.

Civit, M., Civit-Masot, J., Cuadrado, F. & Escalona, M. J. (2022). A systematic review of artificial intelligence-based music generation: Scope, applications and future trends. *Expert Systems with Applications, 209,* Article 118190. https://doi.org/10.1016/j.eswa.2022.118190

Devlin, J., Chang, M. W., Lee, K. & Toutanova K. (2018). BERT: pretraining of deep bidirectional Transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dieleman, S., van den Oord, A. & Simonyan, K. (2018). The challenge of realistic music generation: modelling raw audio at scale. *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Montréal, Canada.

Dhariwal, P., Heewoo, J., Payne, C., Kim, J. W., Radford, A. & Sutskever, I. (2020). JukeBox: a generative model for music. *arXiv preprint. arXiv:2005.00341v1*

Donahue, C., Mao, H. H., Li, Y. E., Cottrel, G. W. & McAuley., J (2019). LakhNES: Improving multi-instrumental music generation with cross-domain pretraining. *20th International Society for Music Information Retrieval Conference*, Delft, Netherlands

Goodfellow, I. J. Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 2672–2680.

Hochreiter, S. & Schmidhuber, J. (1997). Long short term memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M. & Eck, D. (2018). Music Transformer. *arXiv preprint*. https://doi.org/10.48550/arXiv.1809.04281

Huang, Y.-S., Yang, Y.-H. (2020). Pop-music Transformer: beat-based modeling and generation of expressive pop piano compositions. *Association for Computing Machinery*, 1180-1188. https://doi.org/10.1145/3394171.3413671

Joshi, M., Chen, D., Liu, Y., Weld, S. D., Zetttlemoyer, L. & Levy, O. (2020). SpanBERT: improving pre-training by representing and predicting spans. *arXiv preprint. arXiv:1907.10529v3*

Muhamed, A., Li, L., Shi, X., Yaddanapudi, S., Chi, W., Jackson, D., Suresh, R., Lipton, Z. C. & Smola, A. J. (2021). Symbolic music generation with transformer-GANs. In *Proceedings of the AAAI conference on artificial intelligence, 35*(1)*,* 408–417.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. & Kavukcuoglu, K. (2016). WaveNet: a generative model for raw audio. *arXiv preprint*. arXiv:1609.03499v2

Oore, S., Simon, I., Dieleman, S., Eck, D. & Simonyan, K. (2018). This time with feeling: learning expressive musical performance. *arXiv preprint.* arXiv:1808.03715

Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. (2018). Language models are unsupervised multitask learners.

Razavi, A., van den Oord, A., Vinyals, O. (2019). Generating diverse high-fidelity images with VQ-VAE-2. *33rd Conference on Neural Information Processing Systems*, Vancouver, Canada

Roberts, A., Engel, J., Raffel, C., Hawthorne, C. & Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music. *Proceedings of the 35th International Conference on Machine Learning, PMLR (80)*, 4364-4373. https://doi.org/10.48550/arXiv.1803.05428

Shaw, P., Uszkoreit, J. & Vaswani, A. (2018). Self-attention with relative position representations. *arXiv preprint.* arXiv:1803.02155

Song, K., Leng, Y., Tan, X., Zou, Y., Qin, T. & Li, D. (2022). Transcormer: Transformer for sentence scoring with sliding language modeling. *36th Conference on Neural Information Processing Systems* (NeurIPS 2022). *arXiv:2205.12986*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention is all you need. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems,* 6000-6010.

Wu, S.-L. & Yang, Y.-H. (2021). MuseMorphose: full-song and fine-grained piano music style transfer with One Transformer VAE. *arXiv preprint. arXiv:2105.04090*

Payne C., (2019) MuseNet. *OpenAI Blog.* https://openai.com/research/musenet