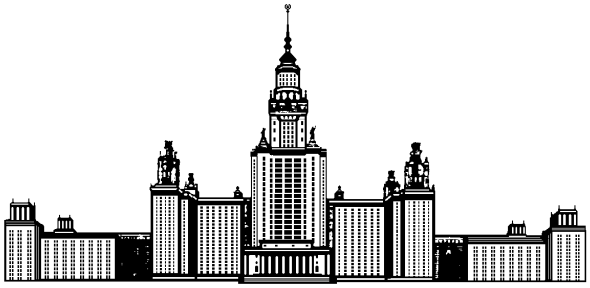


Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Иванов Сергей Максимович

Нейросетевая Генерация Музыки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
д.ф.-м.н., профессор
А. Г. Дьяконов

Москва, 2018

Содержание

1 Введение **4**

2 Задача **5**
2.1 Виды постановок 5
2.2 Представление данных 5

3 Открытые проблемы **7**
3.1 Моделирование принятия творческих решений 7
3.2 Проблема глобальной структуры 10
3.3 Проблема представления 11
3.4 Общие препятствия 12

4 Используемые инструменты **13**
4.1 Нейросетевые генеративные модели 13
4.2 Рекуррентные нейронные сети (RNN) 13
4.3 Long Short-Term Memory (LSTM) 14
4.4 Ограниченная машина Больцмана (RBM) 15
4.5 Generative Adversarial Networks (GAN) 16
4.6 Variational AutoEncoder (VAE) 18

5 Обзор существующих моделей **21**
5.1 Обзор методов генераций мелодии 21
5.2 Обобщения для полифонии при ограничениях 22
5.3 RNN-RBM 25
5.4 Classifying Variational Autoencoder 27
5.5 MuseGAN 31
5.6 Tied Parallel Networks 32
5.7 Performance RNN 33
5.8 BachProp 34

6 Предлагаемая модель **37**
6.1 Общая схема 37
6.2 Сжатие в октаву 37
6.3 Механизм внимания как дополнительная модель памяти 39
6.4 Поголосовое полифоническое сэмплирование 40

7 Эксперименты **44**
7.1 Данные и их предобработка 44
7.2 Параметры модели и реализация 44
7.3 Результаты 45
7.4 Обсуждение 46

8 Заключение **51**

Список литературы **52**

Аннотация

В работе представлен обзор и анализ существующих подходов к генерации полифонических[1] музыкальных произведений для фортепиано. Рассматриваются открытые проблемы в задаче и потенциальные способы их решения.

Также предлагается оригинальная генеративная модель на основе LSTM-нейросетей с использованием аналога механизма внимания и последовательным сэмплированием голосов в качестве модели полифонии.

[1] допускается звучание нескольких нот одновременно

1 Введение

Генерация музыки является самым абстрактным видом творческой деятельности. В отличии от книг, стихов, картин и видео, музыка не содержит привязки к контексту окружающего нас мира и не требует знания таких концепций как, например, смысл слов или антропоморфная фигура. И хотя, конечно, восприятие мира отдельного человека несомненно влияет на то, как он понимает то или иное произведение, музыка остаётся наиболее самостоятельным способом эмоционального взаимодействия между людьми.

Более того, музыка может быть представлена в виде очень простого формата данных без потери основного содержания. Этот факт был известен издревле и привёл к появлению нот; как будет показано далее, нотные партитуры представимы в компьютере как последовательности бинарных векторов, что упрощает их алгоритмическую обработку.

Исходя из этого, музыка кажется потенциально плодоносной областью исследований творческой деятельности и её моделирования в искусственном интеллекте.

Однако, до сих пор нету разумных научных теорий, почему одни последовательности нот воспринимаются как музыка, а другие нет. Философским вопросом является и то, являются ли эти зависимости естественными, а не сформировавшимися в ходе исторического процесса. Так или иначе, в музыке содержатся некие закономерности, интуитивно понятные всем людям, математически строго сформулировать или смоделировать которые пока не удалось [31].

Глубинное обучение сегодня является инструментом построения наиболее сложных моделей, ценой за которые является их интерпретируемость [8]. В последние годы нейросети активно используются для построения генеративных моделей, пытающихся аппроксимировать неизвестное распределение на данных, а рекуррентные сети оказались удобным инструментом для работы с последовательностями. На сегодняшний день, передовые методы генерации музыки основаны на нейросетевом подходе, однако и для них задача представляет собой вызов, подробнее о чём см. раздел 3.

В разделе 4 приводится описание рекуррентных сетей и генеративных моделей как основ для построения моделей генерации музыки. Далее в разделе 5 приводится обзор основных существующих подходов. В разделе 6 описана предлагаемая модель.

2 Задача

2.1 Виды постановок

Конкретизированные формулировки задачи генерации музыки могут принимать различный вид [5]:

- Генерация аудиофайла с записью музыкальных произведений определённого вида.
- Генерация ритма или партии для перкуссии.
- Генерация аккомпанемента под мелодию, обычно в формате выбора одного из заранее фиксированного перечня аккордов.
- Генерация вариаций мелодии, то есть внесение модификаций в заданный пример.
- Генерация мелодии, то есть партии, в которой в каждой момент звучит не более одной ноты, в рамках некоторого символьного представления.
- Генерация полифонической музыки, в рамках некоторого символьного представления.

Считается, что любое музыкальное произведение может быть перенесено на фортепиано, а значит, представимо в виде полифонического произведения. Далее обсуждается задача в последней постановке, однако задача генерации мелодии является важным частным случаем.

2.2 Представление данных

Наиболее распространённым компьютерным представлением нотных партитур является формат MIDI[2] (Music Instrument Digital Interface). Файл состоит из набора команд различного типа; основным типом является команда нажатия ноты, параметрами которой являются три числа: высота звука и громкость, дискретные числа от 0 до 127, и время, которое должно пройти с момента выполнения предыдущей команды. Аналогично есть команда прекращения звучания ноты. Существенно, что события, происходящие одновременно, например, нажатия клавиш одного аккорда, могут идти в файле в произвольном порядке.

Время обычно измеряется в миллисекундах, и в записях живого исполнения в MIDI-формате может быть зашумлено. Однако, в традиционной нотной записи положено указывать продолжительность нот как долю от некоторого заранее фиксированного промежутка времени; при строгом переложении в MIDI время между нотами кратно некоторому минимальному периоду, который образует

[2] https://www.midi.org/

на оси времени дискретную сетку. При этом в свободном доступе есть датасеты MIDI-данных как живых выступлений, то есть без привязки нот к сетке, так и данных с соблюдением формальной длительности нот. Последнее является важным требованием для обучения большинства моделей, поскольку для преобразования таких данных в последовательности требуется дискретизация.

Простейшим примером такой дискретезации является представление Piano Roll [31, с.9-12]. В каждый дискретный момент времени для каждой ноты указывается, нажимается ли она (1) или нет (0).

В таком представлении Piano Roll происходит потеря следующей информации:

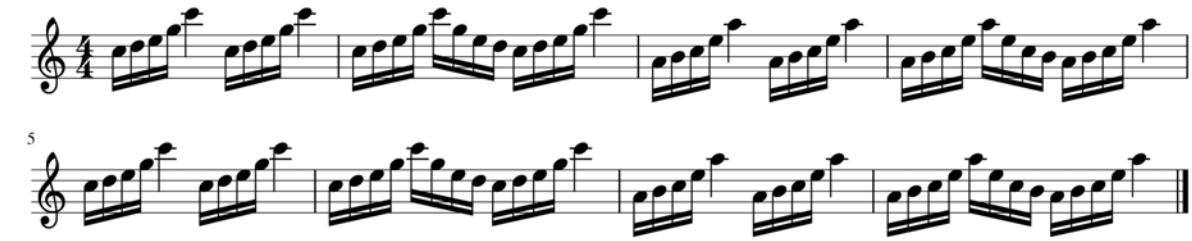
- Громкости звучания ноты. Модели могут иметь дополнительный независимый модуль, определяющий громкость нажатия выбранных нот; тем не менее, ключевые музыкальные закономерности заложены в высоте ноты, и далее генерация громкости рассматриваться не будет.

- Продолжительности звучания ноты. Предположение, что одновременно нажатые ноты звучат до тех пор, пока не будут нажаты следующие ноты, является упрощением. Во избежание этого, во многих моделях 1 в бинарном представлении означает, что нота зажата или продолжает звучать, что позволяет генерировать продолжительность; однако, это исключает возможность нажатия одной и той же ноты два момента времени подряд, и вызывает зависимость от чёткой разметки продолжительности нот в датасете.

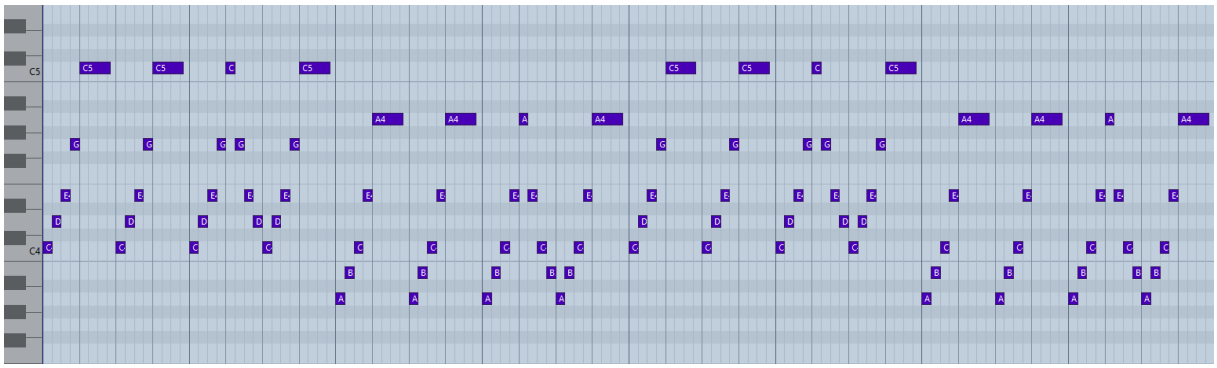
Пример данных в этих трёх представлениях приведён на рис.1.

Традиционно количество клавиш фортепиано равно 88, что соответствует нотам под номерами от 21 до 102 в формате MIDI. Соответственно, музыкальные композиции в таком виде имеют вид 88-мерной бинарной последовательности.

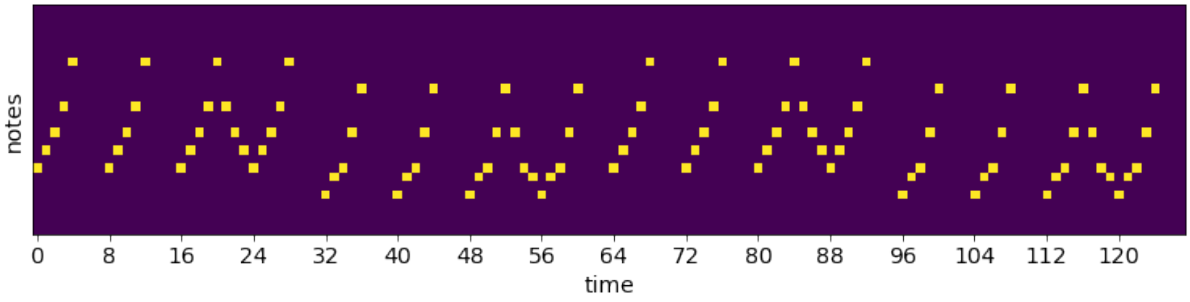
Большинство моделей генерации полифонической музыки работают с представлением Piano Roll, однако оно не является единственно возможным. Более того, некоторые недавние модели, например, PerformanceRNN (см. раздел 5.7) и BachProp (см. раздел 5.8) достигают ощутимых результатов именно за счёт оригинального представления, более близкого к «покомандному» представлению MIDI.



(a)



(b)



(c)

Рис. 1: Различные представления одного и того же фрагмента: a) нотное b) визуализация MIDI-формата в MIDI-редакторе c) Piano Roll

3 Открытые проблемы

3.1 Моделирование принятия творческих решений

Творческая деятельность человека сопряжена с воображением, которое математически можно промоделировать сэмплированием объектов из построенной модели окружающего мира. Рассмотрим для примера мир с дискретным временем $(x_1, y_1), (x_2, y_2), \ldots$, т. ч. $x_t \in \{0, 1\}, y_t \in \{0, 1\}$, и $\forall t\colon p(x_t = 1, y_t = 1) = 0$. Некоторый интеллект, способный строить вероятностное приближение $\hat{p}$ такого мира со способностью сэмплировать из него, имеет возможность получать возможные реализации будущего. В том числе, некоторые из них на самом деле будут невозможными, если для некоторого $t\colon \hat{p}(x_t = 1, y_t = 1) \neq 0$, что сходится с интуитивным понятием человеческого воображения.

Если пространство возможных объектов достаточно сложно, и вероятность независимо засэмплировать один и тот же объект повторно ничтожно мала, сам результат сэмплирования начинает ассоциироваться с построенной интеллектом моделью, а значит, и с самим этим интеллектом (индивидуумом). Тогда результат сэмплирования можно назвать произведением, а индивидуума — автором.

При этом, если для другого индивидуума правдоподобие произведения в рамках его собственной модели мира будет достаточно высоко, ассоциирование бессмысленно, и произведение не будет представлять для него интереса; на примере музыки такими случаями являются примитивные последовательности вроде повторения одной и той же ноты. Противоположно, если правдоподобие слишком низко или равно нулю, произведение «не соотносится» с моделью другого индивидуума, остаётся «не понятым»; в музыке таковым может быть хаотичная последовательность нот, или последовательность, закономерность в которой человек не сможет уловить. Поэтому композиторы часто обращают внимание на чередование ожидаемых нот с появлением менее ожидаемых, избегая абсолютно неожиданных вариантов, которые воспримутся как очевидная фальшь [31, с.307-323].

Для того, чтобы сэмплировать из распределений на достаточно сложных пространствах, процесс генерации может разлагаться на последовательное принятие более простых решений. Например:

$$\begin{gathered} p(Z) = \{Z = \{z_1, z_2, \dots, z_k\}\} = p(z_1)p(z_2 \mid z_1)p(z_3 \mid z_1, z_2) \dots p(z_k \mid z_1, z_2, \dots, z_{k-1}) \\ z \sim p(Z) \quad \Leftrightarrow \quad z = \{z_1^* \dots z_k^*\} : z_1^* \sim p(z_1), z_2^* \sim p(z_2 \mid z_1^*) \dots \end{gathered} \tag{1}$$

Однако, уже в данном примере сделано несколько существенных допущений. Фиксирован и не меняется в процессе сэмплирования набор $\{z_1, z_2, \dots z_k\}$; фиксирован однозначно заданный порядок нумерации, и, как следствие, порядок сэмплирований z_i. Уже сама эта факторизация, как неоднозначная, может сэмплироваться как часть процесса генерации. Однако, она является хорошим приближением для генерации музыки, в которой важную роль играет ось времени, а данные представлены последовательностями $\{n_t\}$, что, казалось бы, задаёт естественную факторизацию; тем менее, такая факторизация является явно недостаточной.

Причиной является то, что произведение $\{n_t\}, n_t \in \{0, 1\}^{88}$, задаёт, аналогично первому примеру, собственный «мир», который можно моделировать и прогнозировать [31, с. 291-304]. Иначе говоря, ставится задача предсказания следующих нот одного конкретного музыкального произведения по обучающей выборке в виде ранее прозвучавших нот. Считается, что слушатель (вне зависимости от наличия музыкального образования) подсознательно строит достаточно точный прогноз на следующие ноты конкретно данного, прослушиваемого произведения. Можно предположить, что из данного прогноза композиторы и сэмплируют следующие ноты произведения; в таком случае, после сэмплирования n_t происходит этап уточнения модели конкретного произведения, «дообучения».

В музыке каждое произведение часто основано на индивидуальных «паттернах», то есть, сильно упрощая, определённые комбинации нот встречаются в нём чаще, чем в других произведениях. Говоря другими словами, каждое музыкальное произведение отличимо от других своей собственной, индивидуальной закономерностью, а значит, от полноценного генератора музыки требуется генерировать не столько комбинации нот, сколько эту самую закономерность. С учётом этого нюанса, задача на самом деле является задачей метаобучения, то есть задачей построения модели, способной обучаться т. н. «one-shot»-задачам [25], в данном случае — отдельным произведениям. Однако, генеративных моделей в подобной парадигме пока построено не было.

В предположении детерминированности процесса обучения можно полагать, что влияние этого эффекта на процесс генерации будет учтено в аппроксимации $p(n_t \mid n_1, n_2, \dots n_{t-1})$, однако тогда данная функция будет существенно зависеть от всех $t-1$ входов, и дополнительные предположения о её виде будут нерелевантны. Например, простейшее предположение марковости, $p(n_t \mid n_1, n_2, \dots n_{t-1}) \approx p(n_t \mid n_{t-1})$, для музыки неприемлемо. Иначе говоря, пространство «текущего состояния» музыки не ограничивается звучавшими в предыдущий момент нотами.

Для расширения пространства «текущего состояния» можно использовать латентные переменные. Например, в рамках общих обозначений (1), положить $p(z_t, s_t \mid z_{t-1}, s_{t-1})$, надеясь, что в s_t будет заложена вся необходимая информация о предыдущих моментах времени. Такое предположение даёт в (1) альтернативную факторизацию:

$$\begin{aligned} p(Z) &= \{Z = \{z_1, s_1, z_2, s_2 \dots, z_k, s_k\}\} = \\ &= p(s_1)p(z_1 \mid s_1)p(s_2 \mid z_1, s_1) \dots p(s_t \mid z_{t-1}, s_{t-1})p(z_t \mid s_t) \end{aligned}$$

Здесь существенно то, что, раз латентная переменная состояния вводится для описания «текущего состояния» музыки, как и на звучащие ноты, их следует генерировать из некоторого аппроксимирующего распределения, а не полагать детерминированной функцией от истории. Иначе говоря, определение следующего «текущего состояния» представляет собой такой же выбор, как и выбор последующих нот, который в такой парадигме представляет собой стохастическое сэмплирование из аппроксимирующего распределения.

К сожалению, такая модель представляет собой вычислительный граф со стохастическими узлами. Аппарат обучения моделей с таким числом латентных переменных требует введения серьёзных дополнительных ограничений на их структуру, пока не позволяющую им играть роль полноценной «памяти», аккумулирующих всю информацию о предыдущих событиях.

Вынужденным решением является предположение о детерминированной зависимости s_t от s_{t-1}, z_{t-1}. В таком виде факторизация примет вид:

$$p(Z) = \{Z = \{z_1, z_2, s_2 \dots, z_k, s_k\}\} = p(z_1)p(z_2 \mid s_2(z_1)) \dots p(z_t \mid s_t(z_{t-1}, s_{t-1}))$$

В таком виде s_t могут быть представлены функциональными моделями памяти (например, LSTM (см. раздел 4.3)). При обучении такой модели отсутствует возможность предполагать какую-либо иную факторизацию (1), помимо факторизации по наблюдаемым переменным; критически важным это обстоятельство становится при переходе от задачи генерации мелодии к полифонии (см. раздел 3.3).

3.2 Проблема глобальной структуры

Считается, что практически произвольные комбинации нот могут быть первыми секундами композиции. Одним из немногих эмпирических ограничений является то, что из 12 нот[3] не может звучать одновременно более 5-7 различных[4]. Продолжение первых комбинаций обязано быть очень похожим (в некотором смысле) на начало, в противном случае продолжение будет воспринято как внезапное начало нового произведения. В музыке это часто называют «установкой паттерна», что интуитивно является повтором первых комбинаций, возможно, с некоторыми изменениями.

Дальнейшее развитие композиции, сильно упрощая, построено на повторах и преобразованиях целых фрагментов, причём размеры этих фрагментов разнятся от нескольких нот до 16 и более тактов, и зачастую образует этакую иерархическую структуру [31, с. 55-83]. Например, на рис. 1 можно заметить, что вторые 8 элементов последовательности являются копией первых 8; вторые 16 элементов копией первых 16 с добавлением трёх нот; вторые 32 очень похожи на циклически сдвинутые первые 32; а вся вторая половина фрагмента из 64 элементов является точной копией первой.

Это наблюдение не является законом или обязательным свойством музыкальных композиций, однако накладывает явное требование на модель: умение считать и копировать. В глубинном обучении такие модели обычно называют моделями с памятью [8, с. 235-245], самой распространённой из которых являются Long Short Term Memory (LSTM) сети, подробнее о которых в разделе 4.3. Однако, эксперименты показывают, что LSTM не справляется с задачей копирования более, чем на 16-20 фрагментов [14], когда для многих произведений (при довольно грубой частотой дискретизации 16 моментов времени на такт) размер копируемых фрагментов может достигать 128, 256 и более моментов времени.

Усложняет воспроизведение повторов и типично используемый способ генерации, при котором на каждом шаге модель выдаёт распределение на следующие ноты $p_t(n)$, из которого происходит сэмплирование. Даже если допустить, что модель с вероятностью 99% будет справедливо уверена в необходимости повтора и

[3] то, что нот 7, является распространённым заблуждением; 7 из 12 нот получили название, однако все 12 нот в традиционном равномерном строе абсолютно равноправны.

[4] иначе человеческое ухо перестанет различать их высоты, а значит, воспринимать последовательности.

сможет при помощи памяти указать нужные ноты, вероятность, что повтор будет засэмплирован 32 итерации подряд не превышает 73%.

Существуют генеративные модели мелодий, которые «локально» (т.е. в пределах нескольких тактов) проходят тест Тьюринга. Однако, даже для мелодий, при субъективном прослушивании результата ощутимо, что каждые 5-10 секунд произведение изменяется, при этом оставаясь музыкой. Это называют проблемой глобальной структуры, которая остаётся нерешённой.

3.3 Проблема представления

Аналогичный «локальный» тест Тьюринга для полифонической генерации пока не пройден. Основным препятствием по сравнению с генерацией мелодии является необходимость промоделировать сэмплирование нескольких нот в каждый момент времени. При этом какие-либо предположения о факторизации распределения приводят к потере т.н. гармонических зависимостей, заключающихся в том, что при определённых условиях тенденцию звучать одновременно имеют определённые подмножества нот. Существуют теории, пытающиеся описать комбинаторные связи между подмножествами нот и формализовать понятие гармонических зависимостей [10], но строгих закономерностей уловить не удаётся.

Формально, распределение на множестве всевозможных комбинация звучащих нот есть распределение на множестве всевозможных бинарных D-мерных векторов. Существуют модели, позволяющие обучиться подобному распределению по выборке (например, RBM, см. раздел 4.4), однако в основе таких моделей часто лежит предположение если не о независимости компонент распределения (то есть, в рамках задачи генерации музыки, нот), то, по крайней мере, их условной независимости. Интуитивно кажется, что такие предположения неверны, так как принятие решения о том, звучит ли некоторая нота, неизбежно влечёт изменение вероятностей звучания других нот вне зависимости от «текущего состояния» музыки. Однако, «текущее состояние» влияет на то, какие именно изменения произойдут.

Разумной альтернативой является предположение некоторого порядка на нотах и сэмплирование каждой последующей на основании того, засэмплировались ли предыдущие. Этот способ довольно громоздкий, но соответствует общему виду распределения на взаимозависимые бинарные переменные $n^1, n^2 \dots n^D$:

$$p(n^1, n^2 \dots n^D) = p(n^1)p(n^2 \mid n^1) \dots p(n^D \mid n^1 \dots n^{D-1})$$

В более общей форме, идея заключается в том, чтобы каким-то образом отойти от сэмплирования из тяжело моделируемого распределения на взаимозависимых бинарных переменных к сэмплированию из категориальных. Эта идея также соотносится с существующими представлениями музыкальной теории о полифонии [31, с. 76-114], где, при переходе от одной комбинации из, например, трёх нот

в другую комбинацию из трёх нот, полагается, что один «голос» отвечает за верхние ноты, второй — за средние, и третий — за нижние. Отчасти такое представление появилось за счёт существования монофонических инструментов, в первую очередь, человеческого голоса. Однако, в такой концепции существуют модели только в рамках некоторых упрощающих предположений (см. раздел 5.2).

Ещё одной важной особенностью музыкальных данных, является тот факт, что все 12 нот являются равноправными, и любое произведение может быть сыграно от любой клавиши фортепиано. Математически это означает инвариантность относительно циклического сдвига векторов последовательности.

При этом очевидных способов учесть эти особенности нот нет. Подобная особенность в задачах нейросетевой обработки изображений обычно учитывается в архитектуре сети использованием свёрточных слоёв [8, с. 199-216], предполагающих, что соседние пиксели изображения взаимозависимы, а далеко расположенные друг от друга — скорее всего, нет. Представленные в формате Piano Roll музыкальные произведения можно обрабатывать свёрточными сетями как «изображения», но важно учитывать, что, во-первых, зависимости по вертикальной оси (гармонические зависимости) и горизонтальной (временные зависимости) имеют совершенно разную природу, а во-вторых, многие «далеко расположенные» относительно друг друга ноты на самом деле взаимозависимы, что, опять же, связано с иерархической глобальной структурой.

Существует попытка построить архитектуру сети с учётом инвариантности относительно циклического сдвига (см. раздел 5.6), но наиболее распространённым решением является расширение выборки инвариантами.

3.4 Общие препятствия

На сегодняшний день целью является построить модель, для которой правдоподобие на имеющихся примерах настоящих музыкальных произведений достаточно высоко. При использовании относительно больших датасетов[5] этап, на котором потенциально возникает переобучение, достигнут не был; авторы генеративных моделей не сталкивались с проблемой, что их модель запоминала всю выборку целиком и не выдавала новизны. Причиной является недостаточно хорошее моделирование памяти и процесса генерации полифонии.

Усложняет положение и отсутствие каких-либо объективных критериев качества генеративных моделей. При условии, что обучение проходило на в точности одном и том же датасете в одном и том же представлении, возможно сравнение моделей через значение её правдоподобия; иначе, единственным способом както сравнить две модели является независимая человеческая оценка. Как следствие, в рамках данной задачи пытаются применять самые различные подходы, проверяя применимость имеющихся генеративных моделей к музыке эмпирическим путём.

[5] 100-300 музыкальных произведений

4 Используемые инструменты

4.1 Нейросетевые генеративные модели

Нейросети [8] как семейства параметрических функций позволяют приближать вероятность одних величин y при условии других величин x: $p(y \mid x) \approx f(x, \theta)$. Для этого необходимо гарантировать, что выход нейросети является распределением на множестве возможных значений y, и предоставить стратегию сэмплирования; например, если y представляет собой бинарный вектор с единственной единицей, возможно в последнем слое заменить функцию активации от его выходов $h \in \mathbb{R}^D$ на софтмакс:

$$p(y = i) = \operatorname{softmax}(h)_i = \frac{e^{h_i}}{\sum_j^D e^{h_j}} \tag{2}$$

Это гарантирует, что выход сети будет категориальным распределением, из которого можно сэмплировать x_t. При этом за счёт дифференцируемости параметры θ можно настроить методом максимального правдоподобия:

$$\prod_{(x_i, y_i) \in \text{data}} p(y_i \mid x_i, \theta) \to \max_\theta$$

Такой метод позволяет простейшим образом генерировать данные, представленные в виде некоторой последовательности $x_1, x_2, \ldots, x_T$, обучая нейросеть $p(x_t \mid x_{t-1}, x_{t-2}, \ldots x_{t-k}, \theta)$ для некоторого выбранного k. Для музыки это применимо в рамках генерации мелодии, где в каждый момент времени может звучать не более одной ноты; при этом для паузы выделяется дополнительная компонента вектора выхода сети, позволяя пользоваться формулой (2).

Для полифонической музыки простейшим вариантом является выдача сетью независимых вероятностей $p_i(n_t) \in [0, 1]$ для каждой ноты в отдельности; сэмплирование далее происходит для каждой ноты независимо в предположении $p(n_t) = \prod_i p_i(n_t)$, что приводит к неудовлетворительным результатам.

4.2 Рекуррентные нейронные сети (RNN)

Простейшей моделью памяти для нейронных сетей является передача во времени информации о выходах нейронов в её скрытых слоях [8, с. 219-227]. В отличии от обычных сетей прямого распространения, нейроны рекуррентной сети получают на вход не только выходы предыдущего слоя, но и собственный выход в предыдущий момент времени:

$$h_t = \sigma(W x_t + U h_{t-1} + b),$$

где $x_t \in \mathbb{R}^D$ — вход слоя в момент времени t, обычно соответствующий выходу предыдущего слоя сети; $h_t \in \mathbb{R}^d$ — выход слоя в момент времени t; σ — некоторая

явно заданная функция активации, например, поэлементно применяемая к компонентам вектора сигмоида, $W \in \mathbb{R}^{d \times D}$, $U \in \mathbb{R}^{d \times d}$, $b \in \mathbb{R}^d$ — параметры модели.

Для обучения параметров применяется алгоритм Back Propagation Through Time (BPTT) [34], при котором при расчёте производной учитывается зависимость $h_t(h_{t-1})$. Обычно последовательности делят на фрагменты некоторой фиксированной продолжительности, и на очередном этапе обучения полагают, что в начале этого фрагмента скрытое состояние равнялось 0; градиентами за пределы фрагмента пренебрегают.

4.3 Long Short-Term Memory (LSTM)

Модель LSTM (Long Short-Term Memory) [20] — одна из самых распространённых моделей памяти. На вход слой из таких нейронов получает, как и для обычного рекуррентного слоя, x_t, h_{t-1}, а также дополнительно «внутреннее состояние» c_{t-1}, соответствующее памяти слоя. Выходом ячейки является не только h_t, распространяющееся дальше по сети, но и c_t, с которым сеть до следующего момента времени не манипулирует.

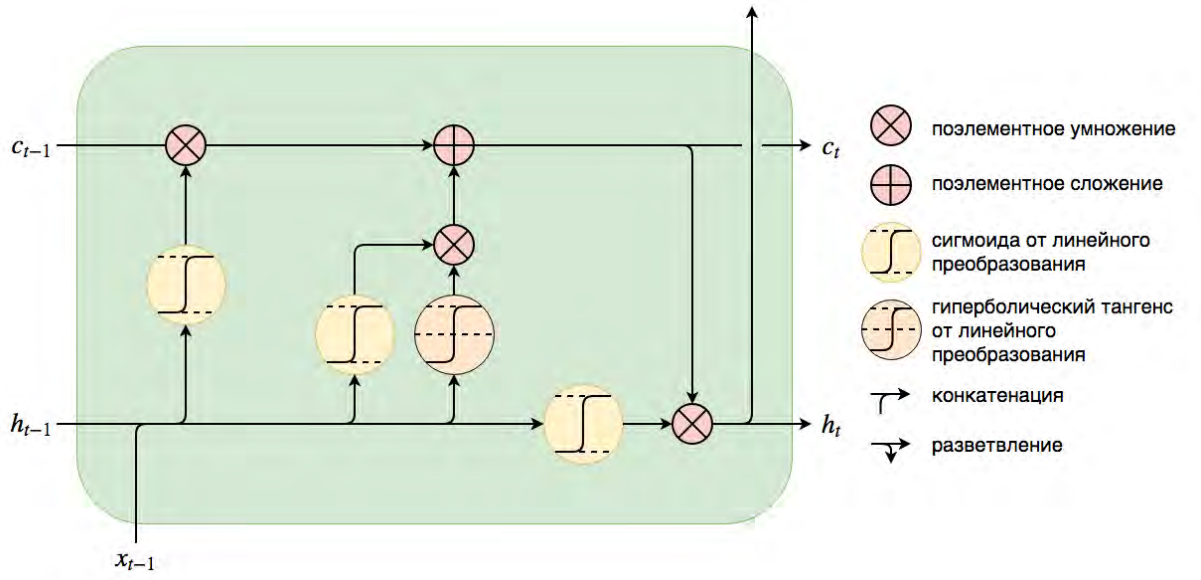


Рис. 2: LSTM-слой

Схема модели (рис. 2) выглядят следующим образом: вычисляется значение-кандидат на заполнение памяти:

$$c'_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

Само заполнение памяти при этом происходит линейно относительно предыдущего значения памяти для предотвращения проблемы затухающих градиентов:

$$c_t = f_t \circ c_{t-1} + i_t \circ c'_t$$

Здесь $\circ$ обозначает операцию поэлементного умножения. Значения f_t и i_t называются «вратами» (gate) и интуитивно должны бы быть пороговыми функциями от входов, принимающими значения 0 или 1 («записывать или не записывать»). Для дифференцируемости модели пороговая функция аппроксимируется сигмоидой:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

Третьи врата отвечают за считывание из памяти, определяющее выход нейрона:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$h_t = o_t \circ \tanh(c_t)$$

Экспериментально показано, что модель значительно успешнее справляется с задачами счёта, что существенно в поиске музыкальных зависимостей. На сегодняшний день, практически все модели для генерации музыки используют LSTM-сети.

Существуют и более сложные модели памяти [14], которые, возможно, смогут приблизиться к улавливанию глобальных зависимостей в музыкальных произведениях. Однако, на текущий момент, моделей с их использованием построено не было, видимо, в силу вычислительной сложности процесса их обучения.

4.4 Ограниченная машина Больцмана (RBM)

RBM (Restricted Boltzman Machine) [18] — способ обучения распределению на взаимозависимых бинарных переменных. Рассматривается двудольная графическая модель из наблюдаемых переменных $x \in \{0,1\}^D$ и скрытых бинарных переменных $h \in \{0,1\}^d$.

Распределение задаётся следующим образом:

$$p(x,h) = \frac{1}{Z}\exp\{b^T x + c^T h + hWx\} \tag{3}$$

Здесь $b \in \mathbb{R}^D, c \in \mathbb{R}^d, W \in \mathbb{R}^{d\times D}$ — параметры модели, Z — неизвестная константа, зависящая от параметров.

Из (3) можно найти условные распределения:

$$\begin{aligned} p(h \mid x) &= \sigma(Wx + c) \\ p(x \mid h) &= \sigma(W^T h + b) \end{aligned} \tag{4}$$

Знание условных распределений позволяет сэмплировать из распределения $p(x,h)$ при помощи стандартного метода Гиббса: для произвольного x_0 последо-

вательно генерируются

$$\begin{aligned} h_0 &\sim p(h \mid x_0) \\ x_1 &\sim p(x \mid h_0) \\ h_1 &\sim p(h \mid x_1) \\ x_2 &\sim p(x \mid h_1) \\ &\vdots \end{aligned}$$

В пределе, то есть после бесконечного числа итераций, $x, h \sim p(x, h)$; на практике ограничиваются k шагами такого алгоритма, причём эмпирически показано, что уже $k = 1$ показывает достаточный результат.

Для обучения параметров модели применяется алгоритм контрастивной дивергенции [18]. Из (3) можно найти маргинальные распределения:

$$p(x \mid \theta) = \frac{e^{-F(x,\theta)}}{Z(\theta)}$$

где θ — параметры модели, а функция $F(x, \theta)$ (т.н. «свободная энергия») имеет следующий вид:

$$F(x, \theta) = -b^T x - \sum_i \log\left(1 + e^{(c+Wx)_i}\right)$$

В данных обозначениях производная логарифма правдоподобия вектора x равна:

$$\frac{\partial\left(-\log p(x \mid \theta)\right)}{\partial\theta} = \frac{\partial F(x,\theta)}{\partial\theta} + \frac{\partial\left(\log Z(\theta)\right)}{\partial\theta} \approx \frac{\partial F(x,\theta)}{\partial\theta} - \frac{\partial F(x^*,\theta)}{\partial\theta} \tag{5}$$

Здесь $x^* \sim p(x \mid \theta)$, где x^* обычно считают методом Гиббса при $k = 1, x_0 = x$. Формула (5) позволяет обучать параметры модели методами стохастической оптимизации.

При генерации из RBM-а сначала принимается стохастически решение по значению некоторого скрытого состояния, а затем на основании значения этого состояния сэмплируются компоненты вектора x в предположении их условной независимости. При этом из-за вида формул условных распределений (4) RBM часто отображается как однослойный персептрон; однако, за одну итерацию по слою будет совершено два прохода, сначала от x к h, а затем от h к x, причём, между этими двумя проходами в вычислительном графе содержится стохастический узел.

4.5 Generative Adversarial Networks (GAN)

Генеративно-состязательная сеть [12] состоит из двух произвольных дифференцируемых функций — $G_\theta(z)$, генератора, параметризованного θ и преобразующему случайный вектор $z \sim p(z)$ в синтезированный пример x, и $D_\phi(x)$, дискриминатора, решающего задачу классификации: относится ли данный ему на вход

пример x синтезированным генератором, или взятым из настоящего датасета. Соответственно, дискриминатор максимизирует правдоподобие:

$$L(\theta, \phi) = \mathbb{E}_{x \sim p_{\text{data}}} \left[\log D_\phi(x)\right] + \mathbb{E}_{z \sim p(z)} \left[\log \left(1 - D_\phi \left(G_\theta(z)\right)\right)\right] \to \max_\phi \tag{6}$$

В силу дифференцируемости оптимизируемой функции по ϕ, дискриминатор можно задавать произвольной нейросетью и обучать. В предположении, что параметры генератора зафиксированы, а дискриминатор оптимален, а то есть выдаёт

$$D_\phi(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{synth}}(x)},$$

где $p_{\text{synth}}(x)$ — вероятность генератора засэмплировать x, значение максимума оптимизируемой функции равно:

$$\max_\phi L(\theta, \phi) = \mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{synth}}(x)}\right] + \mathbb{E}_{x \sim p_{\text{synth}}} \left[\log \frac{p_{\text{synth}}(x)}{p_{\text{data}}(x) + p_{\text{synth}}(x)}\right] =$$

$$= \text{JS}(p_{\text{data}} \parallel p_{\text{synth}}) - \log 4$$

Здесь $\text{JS}(p_{\text{data}} \parallel p_{\text{synth}})$ — дивергенция Йенсена–Шеннона, одна из возможных метрик на множестве вероятностных распределений. Таким образом, логично положить, что для генератора оптимизируемая функция (6) полагается штрафом как оценка различия между синтезированными примерами и настоящими данными, и минимизировать по θ. В результате, возникает антагонистическая минимаксная игра:

$$\min_\theta \max_\phi \mathbb{E}_{x \sim p_{\text{data}}} \left[\log D_\phi(x)\right] + \mathbb{E}_{z \sim p(z)} \left[\log \left(1 - D_\phi \left(G_\theta(z)\right)\right)\right]$$

Показано, что ситуация, при которой генератор производит сэмплы из того же распределения, из которого приходят настоящие данные, а дискриминатор не может их различить, является равновесием Нэша в этой игре.

Модель добилась успеха [27] в генерации изображений. На практике обучение состязательных сетей сопряжено с определёнными трудностями, для избежания которых вносятся эвристические поправки в процедуру обучения. Так, вместо минимизации

$$\mathbb{E}_{z \sim p(z)} \left[\log \left(1 - D_\phi \left(G_\theta(z)\right)\right)\right]$$

по параметрам генератора θ минимизируют другую функцию, обладающую лучшими характеристиками градиента:

$$-\mathbb{E}_{z \sim p(z)} \left[\log \left(D_\phi \left(G_\theta(z)\right)\right)\right]$$

На генератор, помимо дифференцируемости, также накладывается техническое ограничение в виде необходимости выдавать x в явном виде, так как иначе величина $D(G(z, \theta_G))$ не будет дифференцируема по θ_G. Это препятствует использованию модели для генерации дискретных величин. Простейшим способом борьбы является эвристика: пространство объектов $\{0, 1\}^D$ подменяется $[0, 1]^D$, а

реальные данные зашумляются, мешая дискриминатору строить решающее правило на основе близости компонент вектора x к крайним значениям 0 и 1.

Обучение такой модели на музыке усложняется и тем, что в качестве генератора и дискриминатора требуется использовать сложные функции вроде многослойных LSTM-ов; в противном случае, дискриминатор не будет уметь различать синтезированные и настоящие примеры по наличию долгосрочных зависимостей, а генератор — имитировать их.

4.6 Variational AutoEncoder (VAE)

В модели Variational AutoEncoder [23] предполагается, что каждому объекту $x_i \in \mathbb{R}^D$ соответствует некое внутреннее, латентное представление $z_i \in \mathbb{R}^d$, на которое известно априорное распределение $p(z)$. Таким образом, задана вероятностная модель $p(x, z) = p(x \mid z)p(z)$, сэмплирование из которой происходит следующим образом:

- Сэмплируется $z \sim p(z)$ — латентное представление нового сэмпла.
- Вычисляется $p(x \mid z)$ — распределение на x с учётом латентного представления.
- Сэмплируется $x \sim p(x \mid z)$.

Вероятность $p(x \mid z)$ аппроксимируется нейросетью с параметрами θ, обучение которой методом максимального правдоподобия представляет собой нетривиальную задачу:

$$p(x) = \int_z p(x, z)dz = \int_z p(x \mid z, \theta)p(z)dz \to \max_\theta \tag{7}$$

Для решения этой задачи используется техника вариационного байесовского вывода. Для любого распределения $q(z \mid x)$ справедливо следующее разложение:

$$\log p(x) = \int_z q(z \mid x) \log p(x)dz = \int_z q(z \mid x) \log \frac{p(x, z)}{p(z \mid x)}dz =$$

$$= \int_z q(z \mid x) \log \frac{p(x, z)q(z \mid x)}{p(z \mid x)q(z \mid x)}dz = \int_z q(z \mid x) \log \frac{p(x, z)}{q(z \mid x)}dz + \int_z q(z \mid x) \log \frac{q(z \mid x)}{p(z \mid x)}dz =$$

$$= \int_z q(z \mid x) \log \frac{p(x \mid z)p(z)}{q(z \mid x)}dz + \mathrm{KL}(q(z \mid x) \parallel p(z \mid x)) \tag{8}$$

В силу неотрицательности KL-дивергенции, выполняется неравенство:

$$\log p(x \mid \theta) \geq \mathrm{ELBO}(q, \theta) = \int_z q(z \mid x) \log \frac{p(x \mid z)p(z)}{q(z \mid x)}dz$$

При этом равенство достигается при $\mathrm{KL}(q(z \mid x) \parallel p(z \mid x)) = 0$, т.е. в случае, когда $q(z \mid x) = p(z \mid x)$ почти всюду. Следовательно, $\mathrm{ELBO}(q, \theta)$ есть нижняя вариационная оценка на $\log p(x)$, и задача (7) эквивалентна задаче

$$\mathrm{ELBO}(q, \theta) \to \max_{q,\theta}$$

Задача обычно записывается в эквивалентном виде:

$$\mathrm{ELBO}(q, \theta) = \mathbb{E}_q \log p(x \mid z) - \mathrm{KL}(q \parallel p(z)) \to \max_{q,\theta} \tag{9}$$

Если при определённом выборе семейства q, в рамках которого данная задача будет приближённо решаться, второе слагаемое может быть рассчитано аналитически, первое слагаемое представляет собой вычислительную катастрофу. Оптимизируемая функция является мат. ожиданием от параметрической функции по параметрическому семейству, отчего попытки его приближённого вычисления на практике непригодны из-за колоссальной дисперсии.

Ключевой идеей вариационного автокодировщика является Reparametrization Trick. Задача решается в таком параметрическом семействе $q(z \mid x, \phi)$, что существует распределение $\gamma(\xi)$, не зависящее от x и ϕ, такое что величина $z = f(\xi, x, \phi), \xi \sim \gamma(\xi)$ имеет распределение $q(z \mid x, \phi)$ для всех ϕ и x. Здесь f — произвольная дифференцируемая функция с параметрами ϕ, например, нейросеть.

При такой репараметризации в задаче (9) оптимизируется мат.ожидание параметризованной функции не по параметризованному семейству, а по зафиксированному распределению $\gamma(\xi)$:

$$\mathbb{E}_q \log p(x \mid z)p(z) = \mathbb{E}_\xi \log p(x \mid f(\xi, x))$$

Для данного слагаемого возможна приблизительная несмещённая Монте-Карло оценка:

$$\mathbb{E}_\xi \log p(x \mid f(\xi, x)) \approx \sum_{i=0}^{M} \log p(x \mid f(\hat{\xi}_i, x)), \quad \hat{\xi}_i \sim \gamma(\xi) \tag{10}$$

В таком виде, при условии, что второе слагаемое оптимизируемой функции (9) $\mathrm{KL}(q \parallel p(z))$ рассчитывается аналитически, возможен расчёт градиентов как по θ, так и по ϕ, параметрам распределения q. Следовательно, возможен поиск решения задачи методами стохастической оптимизации [22].

Модели $p(x \mid z)$ и $q(z \mid x) \approx p(z \mid x)$ соответственно называются вариационным декодировщиком и вариационным кодировщиком по аналогии с детерминированной моделью нейросетевого автокодировщика.

Данный инструментарий является одним из основных текущих байесовских методов работы с латентными переменными. Однако, ограничения, которые на них накладываются, довольно сильны. Во-первых, Reparametrization Trick неприменим к дискретным переменным. Типичным выбором $q(z \mid x)$ и априорного $p(z)$ является

$$p(z) = \mathcal{N}(0, I)$$

$$q(z \mid x) = \mathcal{N}(\mu(\phi), \Sigma(\phi)) = \mu(\phi) + \Sigma(\phi)\xi, \quad \xi \sim \mathcal{N}(0, I)$$

Во-вторых, использование нейросетей в качестве моделей условных распределений препятствует использованию на практике Монте-Карло оценки с $M > 1$, что делает аппроксимацию относительно грубой, и утяжеляет обучение на практике.

5 Обзор существующих моделей

5.1 Обзор методов генераций мелодии

Хотя практически все современные модели для генерации музыки основаны на нейросетевом подходе, возможно использование других подходов, например, скрытых марковских моделей.

Основное преимущество нейросетей перед марковскими цепями — возможность генерировать новые комбинации из нотных последовательностей, не встречавшиеся в исходных данных. Впервые рекуррентные сети были применены к задаче генерации мелодий в [32]. В [11] при помощи LSTM-сети построили модель, генерирующую не только мелодию, но и аккомпанемент к ней в формате выбора аккорда из зафиксированного перечня, и, отчасти, впервые добились благозвучного результата.

Современных моделей для генерации мелодий или мелодий с аккордовым аккомпанементом из перечня достаточно много. Практически все из них используют представление Piano Roll или т. н. ABC-нотацию, в которой мелодия с гармониями записывается в текстовом виде (например, «| A E A E | A G# G#2 |»), сводя таким образом задачу к задачи генерации текста (аналогично решаемой при помощи LSTM-сетей). Основные различия заключаются в выборе различных стратегий предобработки данных, разных датасетов, архитектуры LSTM-сети, последовательности генерации аккордов и мелодии.

Помимо аккордов и мелодии, возможна предварительная генерация ритма, или, в более сложном варианте, генерация партии перкуссии. Часто эти генераторы представляют как общую сеть, как, например, в [6], где каждый слой представляет собой один генератор. На стадии обучения такого «стэка» генераторов в последующие слои подаются истинные сэмплы из выборки (например, в генератор аккордов, следующий за генератором мелодии, подаётся мелодия из датасета), а на стадии генерации «слои» работают с выходом предыдущих.

Помимо значения предыдущей ноты n_t, в генератор может быть подана дополнительная информация [26]: так обычно подаётся время в дискретном представлении (фиксированной размерности) для облегчения обнаружения ритмических зависимостей.

С целью попытаться уловить глобальную структуру, можно подавать вектора из истории, соответствующие потенциальным периодам мелодии, например n_{t-15}, n_{t-31} при прогнозирования n_{t+1}, или аналогичным образом встраивать skip connections в архитектуру сети. Также можно подавать индикаторы копирования, например, $\mathbb{I}[n_{t-16} = n_t]$, $\mathbb{I}[n_{t-16} = n_t]$. Подобные трюки теоретически могут позволить добиться от генератора копирования фрагментов, но могут привести к переобучению и не способны позволить имитировать модификации уже созданных фрагментов.

Возможно подавать и статистическую информацию по истории. Например, в [6] для каждого момента времени считается статистика по недавно прозвучав-

шим нотам, которая далее кластеризуется по 10 классам. Индексы кластеров сглаживаются скользящим окном и преобразуются в 10-мерное векторное представление, «описывающее» недавнюю историю.

В улавливании глобальной структуры также может помочь использование иерархических архитектур сети, например, Multi Time Scale-архитектур [17]. Проход по первому слою сети в таких архитектурах производится только раз в T_1 моментов времени, а выход слоя считается постоянным в течение этих промежутков. Аналогично, проход по второму слою производится раз в T_2 момента времени, и так далее. Только последние слои представляю собой обычную рекуррентную сеть, производящую вычисления на каждом временном шаге. На рис. (3) представлена такая трёхуровневая модель.

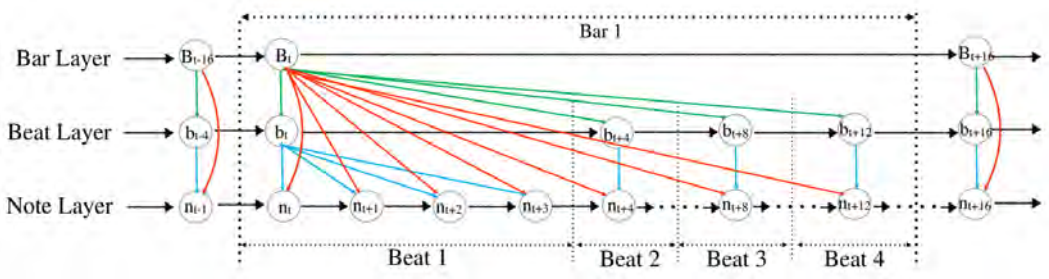


Рис. 3: Трёхслойная иерархическая Multi Time Scale-архитектура [17]

В идеале, генератор, способный сэмплировать произведения с глобальной структурой, должен уметь имитировать такие действия, как модификация уже созданных фрагментов и склеивание этих фрагментов между собой. Первая задача является задачей построения вариаций по заданной мелодии. Интересные результаты в этой задаче были получены в модели [28], в которой использовалась архитектура вариационного автокодировщика с иерархической организацией декодировщика (рис. 4). Благодаря интерполяции в пространстве латентных переменных двух фрагментов мелодии, возможно получать благозвучные промежуточные мелодии «между ними».

Задача «сбора» произведения из небольших фрагментов также является нетривиальной в силу неочевидности того, какие фрагменты могут следовать друг за другом. Задача генерации мелодий на основе выборе фрагментов из произвольного перечня [4] может быть решена при помощи выделения признаков предыдущего фрагмента и генерации желаемых признаков последующего, которые сравниваются с признаковым описанием вариантов.

5.2 Обобщения для полифонии при ограничениях

В связи с успехами в задаче генерации мелодий, логичным способом обобщить модели на случай полифонии является «параллельный» запуск нескольких монофонических генераторов («голосов») с какой-либо организацией их взаимозависимости, например, за счёт общего внутреннего состояния или общих высо-

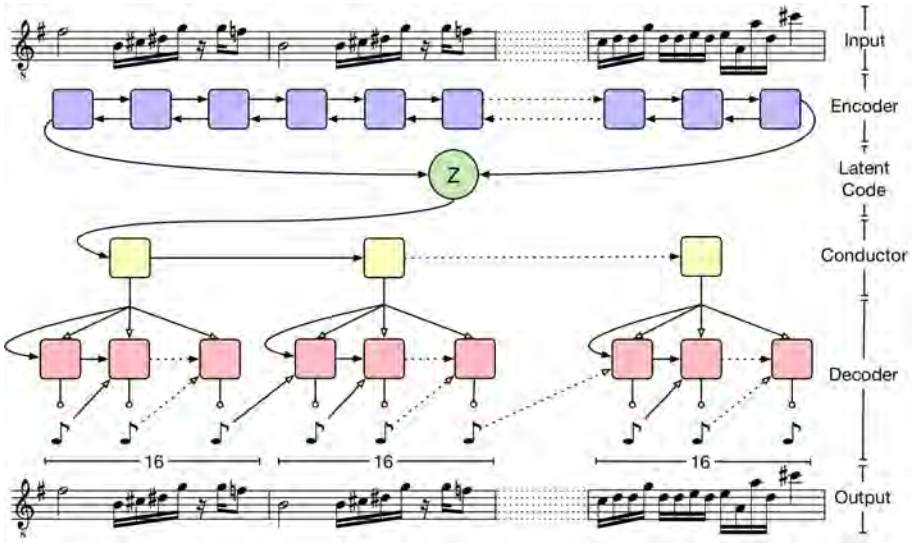


Рис. 4: Модель вариационного автокодировщика с иерархическим декодировщиком [28]

коуровневых признаков, полученных из общего модуля. Количество голосов при этом можно зафиксировать, ограничив, например, максимальным количеством одновременно звучащих нот в реальных данных, V.

Иначе говоря, сэмплирование $n \in \{0,1\}^{88}$ из нетривиального распределения на взаимозависимых бинарных переменных подменяется сэмплированием из V категориальных распределений $n^i, i \in \{1 \dots V\}$, например, в предположении «нота звучит, если её выдал хотя бы один голос»:

$$n = \bigcup_i^V n^i$$

Однако, чтобы трактовать выход нейросети как категориальное распределение $p(n^V \mid n^1, n^2, \dots, n^{V-1})$, необходимо[6] иметь выходной слой размером 88^V. Вместо этого разумнее непосредственно моделировать каждое $p(n^k \mid n^1, n^2, \dots, n^{k-1})$ нейросетью с $n^1, n^2, \dots, n^{k-1}$ в качестве входа.

Можно заменить $p(n^k \mid n^1, n^2, \dots, n^{k-1})$ на $p(n^k \mid \bigcup_i^{k-1} n^i)$, поскольку это приводит лишь к потере информации о том, какой голос какую ноту засэмплировал — в предположении равноценности голосов эта информация бессодержательна.

При обучении такой модели для вычисления функции потерь необходимо рассчитать вероятность звучания каждой из нот.

$$p(n_m) = p(n^1 = m) + p(n^1 \neq m)p(n^2 = m) + \dots + p\left(m \notin \bigcup_{i=0}^{V-1} n^i\right) p(n^V = m) \qquad (11)$$

[6] Альтернативные упрощения вроде $p(n^1 \dots n^V) \approx \prod_i p(n^i)$ здесь нерелевантны

При этом составляющие этого выражения могут быть получены маргинализацией. Так, для последнего голоса:

$$p(n^V) = \int_{n^1 \dots n^{V-1}} p(n^V \mid n^1, n^2, \dots, n^{V-1}) p(n^1, n^2, \dots, n^{V-1}) dn^1 \dots dn^{V-1}$$

Данный интеграл не упрощается и требует 88^{V-1} запусков нейросети для расчёта одной вероятности. Перебор всех возможных значений $\bigcup_i^{V-1} n^i$ не сильно упрощает положение.

Поголосовое сэмплирование может стать возможным при введении дополнительных ограничений на данные. В модели DeepBach [15] (2016) обучение проводилось на датасете BachChorales, более 300-ах четырёхголосых произведениях. Особенностью датасета является то, что для каждой ноты указано, какому голосу он принадлежит, или, иначе говоря, разделение на голоса размечено. Это позволяет непосредственно обучать $p(n^V \mid n^1, n^2, \dots, n^{V-1})$.

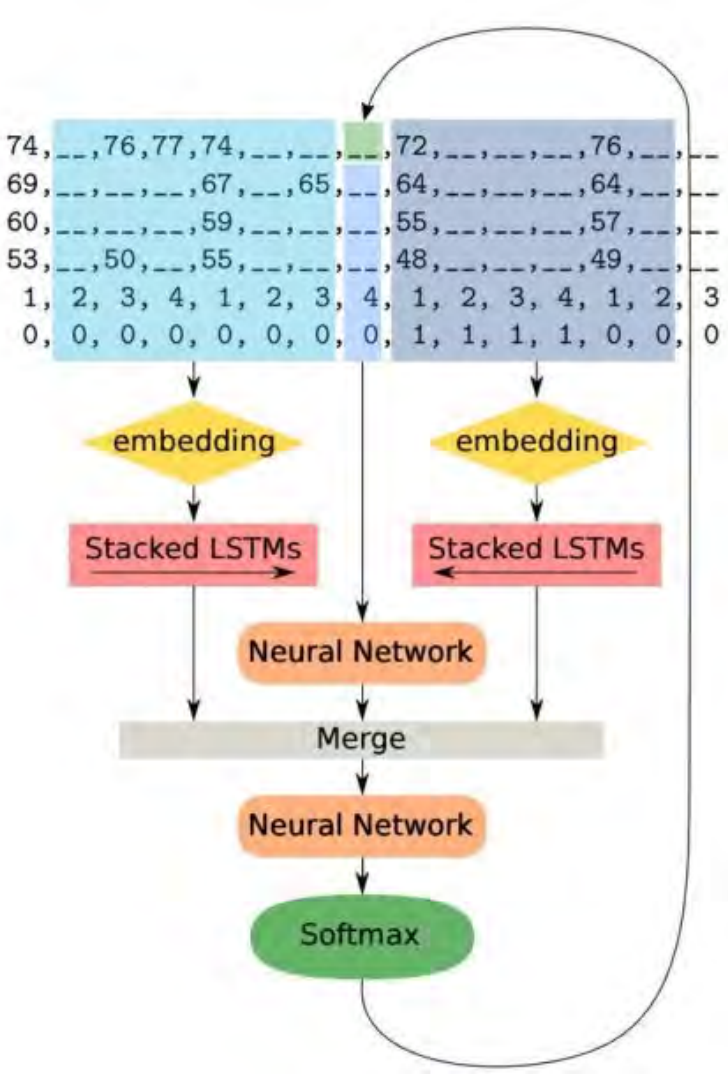


Рис. 5: Модель DeepBach [15]

Общая схема модели DeepBach представлена на рис. 5. Предлагается генерировать i-ый голос в t-ый момент времени на основе значений других голосов, а также всех остальных (как предыдущих, так и последующих) моментов времени. Один LSTM проходит с начала последовательности до $t-1$ времени по значениям всех четырёх голосов; второй, аналогично, проходит с конца произведения T до $t+1$. Только выходы LSTM-ов в последние моменты времени используются далее: они конкатенируются с обработанными обычной сетью данными о текущем моменте времени и подаются в полносвязные слои. Выходом сети является обычное категориальное распределение.

Такая схема задаёт $p(n_t^i \mid n_{\setminus(i,t)})$ и позволяет сэмплировать само произведение методом Гиббса с факторизацией по всем n_t^i.

В общем случае, для использования подобных поголосовых схем требуется свести задачу к индивидуальному обучению каждого голоса. Возможным способом сделать это является «неявная» разметка: тем или иным способом указать, от какого голоса какую ноту требуется засэмплировать. Подходов может быть несколько:

- воспользоваться эвристическими способами разделить голоса по принципу близости («если три ноты переходят в три ноты, то верхняя нота переходит в верхнюю, средняя в среднюю, нижняя в нижнюю» и подобное)
- построить графическую модель для поиска оптимальных с точки зрения той же, например, близости разделений. Критерий всё равно потребуется задать эвристически.
- Потребовать от модели сэмплировать одновременно звучащие ноты именно в заданном порядке, например, от нижних к верхним.

Последний подход по сути является явным переходом от полифонии к последовательному сэмплированию из категориальных распределений: модель либо сэмплирует следующую ноту, либо спецсимвол, указывающий на переход к следующему моменту времени. В таком формате модель может засэмплировать любое количество одновременно звучащих нот, снимая ограничение на число голосов. Однако, поскольку каждый момент времени моделируется более чем одним проходом по сети, улавливание ритмических зависимостей заведомо усложняется, как и утяжеляется задача для памяти. В 2016-ом году на основе этого подхода удалось построить модель [33] в предположении заданного ритма.

5.3 RNN-RBM

Поскольку рекуррентные сети справляются с временными («горизонтальными») зависимостями, а RBM позволяет сэмплировать аккорды, т.е. улавливать «вертикальную» зависимость, логично эти подходы объединить. Реализацией этой идеи стала модель RNN-RBM [3].

Нестрого говоря, некоторые параметры RBM должны зависеть от времени; эту зависимость и аппроксимировать рекуррентной сетью. В модели из трёх параметров распределения (3), W, b, c, от времени зависят только два последних. Таким образом, выходом рекуррентной сети в каждый момент времени являются b и c для машины Больцмана, регулируя распределение, из которого далее будет производиться сэмплирование (рис. 6).

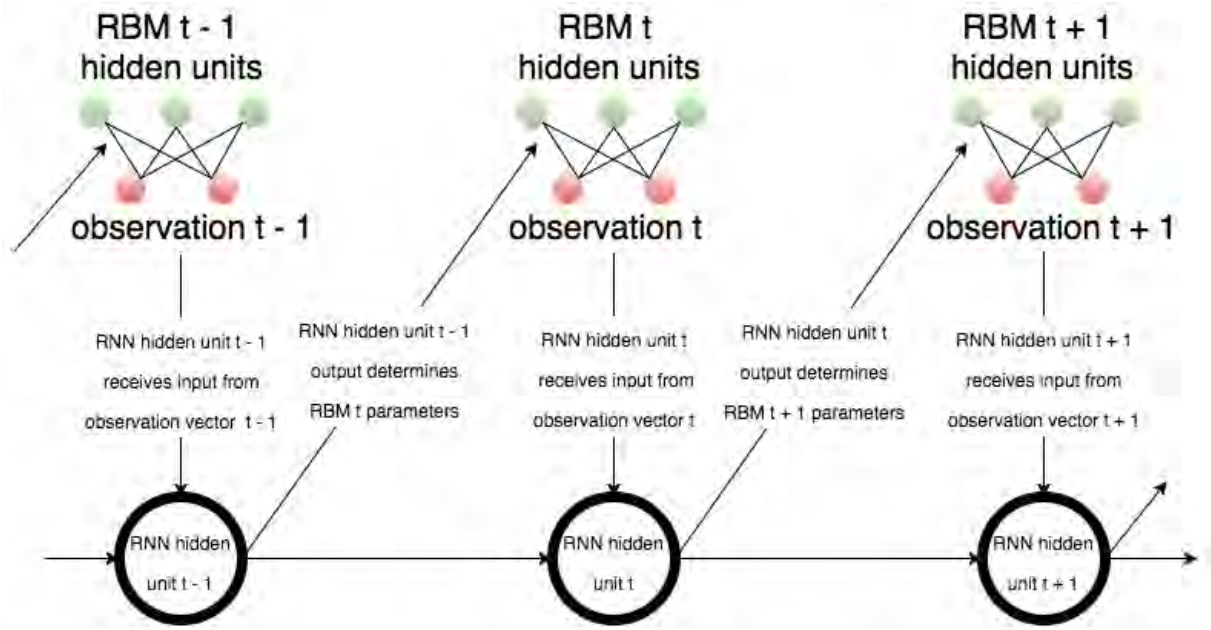


Рис. 6: Модель RNN-RBM [3]

Обучение модели происходит следующим образом:

- Прямой проход: рекуррентная сеть при текущих параметрах на очередном шаге по входу x_t генерирует параметры b_t, c_t для RBM.
- Как и в обычном RBM, сэмплируется x^*_{t+1} методом Гиббса при $k = 1$ при затравке x_t; не зависящий от времени параметр W обучается по формуле (5).
- Обратный проход: в (5) достаточно учесть, что b_t, c_t теперь зависят от выхода нейросети, и вычислить градиенты по её параметрам стандартным алгоритмом BPTT.

Таким образом, обучение рекуррентной сети в данной модели эквивалентно обучению с функцией потерь $\log p(x_t \mid \theta)$, которая на каждом шаге приближается стохастической аппроксимацией.

Данная модель одной из первых добилась результата в задаче генерации полифонической музыки без введения существенных ограничений. Фактически, за счёт слоя латентных переменных RBM-а, из которого на стадии генерации происходит сэмплирование, в граф вычислений на каждом временном шаге добавляется стохастический узел. При генерации на этапе сэмплирования по Гиббсу значение латентных переменных сэмплируется, что соответствует принятию

некоторого промежуточного решения, определяющего «внутреннее состояние» текущих гармоний. При этом никаких интерпретаций значениям латентных переменных не вводится, но делается предположение, что события звучания нот условны независимы при их фиксированном значении.

Размер скрытого слоя RBM в подобных моделях достаточно велик, обычно превосходя размерность x_t в несколько раз. Если предположить, что настоящее распределение имеет вид последовательного принятия длинной цепочки простых, но зависимых решений, то неудивительно, что для его аппроксимации несколькими независимыми решениями требуется брать большое число переменных. Поэтому одно из основных направлений модификации данной модели — увеличение числа принимаемых зависимых стохастических решений на каждом временном шаге.

Прямой способ сделать это, предложенный в [13], основан на замене RBM на Deep Belief Network (DBN), модели, в которой несколько RBM-ов образуют стэк и обучаются жадным образом [19]. Первый RBM в стэке обучается аналогично обычному RBM-у, воспроизводить значения наблюдаемых переменных x через стохастический слой латентных переменных. Сэмплы латентных переменных первого RBM-а используются в качестве обучающей выборки для второго RBM-а с другими параметрами. Аналогично RNN-RBM, в модели RNN-DBN предполагается, что только смещения b^k, c^k, где k — номер слоя DBN, зависят от времени и являются выходом рекуррентной сети.

Рекуррентные сети в таких моделях настраиваются на выдачу подходящих параметров графической модели, играющей здесь роль полифонического сэмплера. Теоретически, для схожих музыкальных ситуаций «хороших» параметров такого сэмплера может быть много разных. При обучении, для различных похожих ситуаций градиенты, текущие в рекуррентную сеть, могут вести в различные локальные оптимумы и потому сильно различаться. Это не мешает улавливанию гармонических зависимостей, если предположить, что эти оптимумы более-менее одинаково хороши, но мешает сети понять ритмические закономерности, законы чередования таких ситуаций, поскольку «размывается» информация о наличии их похожести. Это объясняет, почему ритмические рисунки сэмплов полифонической музыки значительно беднее по сравнению с генераторами мелодий, обычно неплохо справляющихся с генерацией различных ритмов.

5.4 Classifying Variational Autoencoder[7]

Понятие тональности [31, с.115-165] из теории музыки можно нестрого определить как некоторое подмножество нот (обычно, из 7 нот), которые на протяже-

[7]Несмотря на субъективно невысокое качество сэмплов, данная модель представляет теоретический интерес как попытка научиться работать с дискретным стохастическим внутренним слоем на данных с нетривиальными зависимостями, коими являются музыкальные произведения.

ние некоторого фрагмента имеют тенденцию звучать. Иначе говоря, примерно для половины из 12 нот в каждый момент времени генератору следует выдавать очень низкие вероятности звучания. Именно звучание этих нот «вне тональности» может приводить к возникновению «фальши» в сэмплах, то есть генерации последовательностей, который человек не воспримет как музыку.

Практически никогда не нарушаемым эмпирическим фактом является то, что в музыке в каждый момент времени из 12 нот есть «центральная» нота, «центр тяготения» или тоника. Равноправие 12 нот достигается именно за счёт того, что для данной мелодии с некоторой тоникой можно построить эквивалентную мелодию с любой другой из 11 оставшихся нот в качестве центральной.

В теории музыки полагается, что тональность может быть задана при помощи центральной ноты и «лада», способа построения подмножества из 7 нот по заданной тонике. Наиболее распространёнными являются два лада, известные как мажор и минор, однако такое описание «скрытого состояния» музыки является далеко не исчерпывающим.

Несмотря на эвристичность всех этих понятий, имеет смысл пытаться как-то учесть наличие таких «скрытых» переменных во временном ряде. Наиболее логичным кажется использование байесовских моделей, например, вариационного автокодировщика, в моделях которых как раз присутствуют локальные скрытые переменные.

Однако, «центральная нота» и лад как скрытые переменные по природе являются дискретными, а в модели VAE архитектурно необходимо наличие скрытого слоя с непрерывной природой. К тому же, обычно латентные переменные интерпретируются как содержащие «сжатое» представление объектов, и одно лишь знание о тональности является явно недостаточным для такого представления.

В модели Classifying Variational Autoencoder [16] предлагается решить вторую проблему совмещением скрытого представления в виде текущей тональности (набор которых фиксирован) и некоторого не интерпретируемого скрытого представления с гауссианами в качестве априорного (аналогично оригинальной модели VAE). Иначе говоря, вероятностная модель для некоторого фиксированного момента времени задаётся как

$$p(x, z, w) = p(x \mid z, w)p(z)p(w),$$

где x — набор звучащих нот, z — некоторое скрытое представление, $p(z) = \mathcal{N}(0, I)$, w — интерпретируемая как текущая тональность локальная дискретная переменная, один из C классов. Предполагается, что для обучающей выборки также доступна w, или разметка текущей тональности. В MIDI-файлах такой разметки нет, но существуют эвристические алгоритмы, позволяющие получить разметку, похожую на ту, которую мог бы сделать асессор [31, с.167-201].

При генерации w должен быть засэмплирован из некоторого категориального распределения, которое аппроксимировано аналогично кодировщику в оригинальном VAE. Аналогично выводу ELBO в стандартном VAE, для произвольных случайных величин z, w:

$$\log p(x) = \int\limits_z \int\limits_w q(z, w \mid x) \log \frac{p(x, z, w)}{p(z, w \mid x)} dzdw =$$

$$= \int\limits_z \int\limits_w q(z, w \mid x) \log \frac{p(x, z, w) q(z, w \mid x)}{p(z, w \mid x) q(z, w \mid x)} dzdw =$$

$$= \int\limits_z \int\limits_w q(z, w \mid x) \log \frac{p(x \mid z, w) p(z, w)}{q(z, w \mid x)} dzdw + \mathrm{KL}(q(z, w \mid x) \parallel p(z, w \mid x))$$

В силу неотрицательности KL-дивергенции, задача максимизации неполного правдоподобия $p(x)$ эквивалентна максимизации ELBO:

$$\mathrm{ELBO} = \int\limits_z \int\limits_w q(z, w \mid x) \log \frac{p(x \mid z, w) p(z, w)}{q(z, w \mid x)} dzdw =$$

$$\int\limits_z \int\limits_w q(z, w \mid x) \log p(x \mid z, w) dzdw - \mathrm{KL}(q(z, w \mid x) \parallel p(z, w)) \to \max_{q(z,w|x)}$$

В силу предположения о независимости z и w эквивалентно:

$$\int\limits_z \int\limits_w q(z \mid x) q(w \mid x) \log p(x \mid z, w) dzdw -$$

$$- \mathrm{KL}(q(z \mid x) \parallel p(z)) - \mathrm{KL}(q(w \mid x) \parallel q(w)) \to \max_{\substack{q(z|x) \\ q(w|x)}} \quad (12)$$

Оптимизация здесь проходит по всем параметрам, а именно:

- Параметрам $p(x \mid z, w)$, которое моделируется нейросетью, выдающей индивидуальные вероятности для каждой ноты. При генерации данные ноты сэмплируются независимо с выданными вероятностями; модель предполагает обеспечение полифонии за счёт стохастического слоя скрытых переменных.
- Параметрам $q(z \mid x)$, которая здесь играет ту же роль, что и в обычном VAE. Выходом этой нейросети является среднее и диагональ матрицы ковариации гауссианы. Член $\mathrm{KL}(q(z \mid x) \parallel p(z))$, соответственно, может быть рассчитан аналитически, а мат. ожидание по $q(z \mid x)$ в процессе оптимизации может быть дифференцировано при помощи Reparametrization Trick.
- Параметрам $q(w \mid x)$, для которого также необходимо уметь проводить Reparametrization Trick. Это не позволяет взять в качестве априорного распределения $p(w)$ распределение Дирихле, которое было бы здесь наиболее логичным.

Авторы модели предлагают взять в качестве априорного $p(w)$ логистическое нормальное распределение $\mathcal{LN}(0, I)$, и выход сети $q(w \mid x)$ также интерпретировать как среднее и диагональ матрицы ковариации логистического нормального. Для такой модели существует Reparametrization Trick:

$$\xi \sim \mathcal{N}(\mu, \Sigma)$$

$$w = \{w_1, w_2, \dots w_d\}, \quad w_i = \begin{cases} \frac{e^{\xi_i}}{1+\sum_j^{d-1} e^{\xi_j}} & i \neq d \\ \frac{1}{1+\sum_j^{d-1} e^{\xi_j}} & i = d \end{cases}$$

$$\Rightarrow w \sim \mathcal{LN}(\mu, \Sigma)$$

Таким образом, w всё равно остаются непрерывными переменными с носителем $[0, 1]$. Фактически, такое нововведение добавляет в модель ещё один скрытый слой с другим априорным распределением, а информация об известных значениях w для каждого объекта обучающей выборки пока не используется.

В оригинальном VAE зазор между неполным правдоподобием и ELBO, равный в данной модели

$$\mathrm{KL}(q(z, w \mid x) \parallel p(z, w \mid x)) = \mathrm{KL}(q(z \mid x) \parallel p(z \mid x)) + \mathrm{KL}(q(w \mid x) \parallel p(w \mid x)),$$

оценить невозможно в силу отсутствия информации о настоящем скрытом представлении каждого объекта.

В данном же случае полагается известной информация о скрытых переменных w для каждого объекта x, или, иначе говоря, известно $p(w \mid x)$. Вообще говоря, оно вырождено, поскольку для каждого x эвристические алгоритмы определения тональности обычно выдают детерминированный результат $\overline{w}(x)$. Авторы предлагают вместо KL-дивергенции между $q(w \mid x)$ и $p(w \mid x)$ (которая формально может равняться минус бесконечности, когда одно из распределений вырождено) минимизировать кросс-энтропию $L(w, \overline{w})$, как при решении обычной задачи классификации. Тогда оптимизируемая функция принимает вид:

$$\mathrm{ELBO} - \alpha L(w, \overline{w}) \to \max$$

Здесь α — ручной гиперпараметр.

Авторы модели делают интересное наблюдение, что использование LSTM-сетей в качестве кодировщика и декодировщика очень слабо повышает качество по сравнению с обычными нерекуррентными сетями. Видимо, это связано с тем, что, несмотря на применение Reparametrization Trick, техники, позволяющей наиболее сильно сбить дисперсию градиента, разброс проходящего через LSTM градиента всё равно слишком большой, чтобы уловить хоть сколько-то информации о даже локальной структуре музыки, и мешают понять, какую информацию следует запоминать. Однако, авторы отмечают, что модель научилась «оставаться в тональности», или, иначе говоря, выдавать комбинации нот, соответствующие значениям латентной переменной w.

5.5 MuseGAN

Использование GAN-ов для задачи генерации музыки сопряжёно с нестабильностью их обучения в случаях, когда генератор и дискриминатор представляют собой рекуррентные сети. Одним из возможных способов обхода этой проблемы является отказ от работы с музыкальными данными как с последовательностями и переход к свёрточным сетям, обрабатывающим их как изображения. Выходом таких свёрточных сетей обычно является произведение (или фрагмент произведения) фиксированной длины.

В задаче генерации полифонической музыки примером такой модели является MuseGAN [24] (2017), где на основе генеративно-состязательного подхода построена сеть, выдающая партии сразу нескольких инструментов.

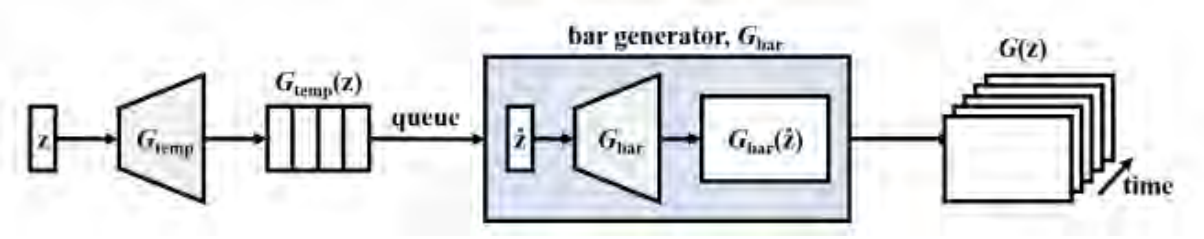


Рис. 7: Схема генератора в модели MuseGAN [24]

Схема генератора для одного инструмента представлена на рисунке 7. Шум z подаётся в сеть G_{temp}, состоящую из двух одномерных (вдоль оси времени) транспонированных свёрточных слоёв [9, с. 19-27]. Эта сеть отвечает за «структуру» будущего трека и выдаёт T латентных векторов $z^{(t)}$. В этих обозначениях t — номер такта генерируемого произведения. Сами такты далее генерируются независимо, а зависимость между ними обеспечивается за счёт общей сети G_{temp}, выдававшей $z^{(t)}$.

Для генерации t-го такта вектор $z^{(t)}$ вместе с новым шумом $\hat{z}$ подаётся на вход сети G_{bar}, состоящему из нескольких транспонированных одномерных свёрточных слоёв, сначала вдоль оси времени, затем вдоль оси нот. Использование одномерных свёрток вместо более традиционных двумерных обусловлено тем, что зависимости вдоль осей имеют разную природу. Выходом сети является один такт произведения, то есть матрица размером число долей в такте на число нот.

Дискриминатор, получающий на вход все T сгенерированных тактов, представляет собой свёрточную сеть, повторяющую свёртки из генератора в обратном порядке, и выдающий на выходе вероятность того, что произведение настоящее.

Взаимозависимость партий между инструментами может обеспечиваться за счёт одинакового шума, подаваемого в персональные для каждого инструмента генераторы, или за счёт общей для всех инструментов сети G_{temp}.

Хотя модель решала немного другую задачу[8] и обучалась на достаточно специфических данных, интерес здесь представляет демонстрация гибкости подобных

[8] а именно, генерация фрагментов рок-произведений

неявных вероятностных моделей. В генераторе возможно подавать на вход разнородный шум на разных этапах генерации, а за счёт общих модулей обеспечивать необходимые взаимозависимости, моделируя таким образом весьма нетривиальные генеративные процессы.

5.6 Tied Parallel Networks

Инвариантность музыкальных произведений относительно циклического сдвига всех нот обычно учитывается тривиальным образом: расширением имеющейся выборки при помощи инвариантов. В 2017 году была предложена модель, получившая название Tied Parallel Network [21], учитывающая эту особенность данных архитектурно.

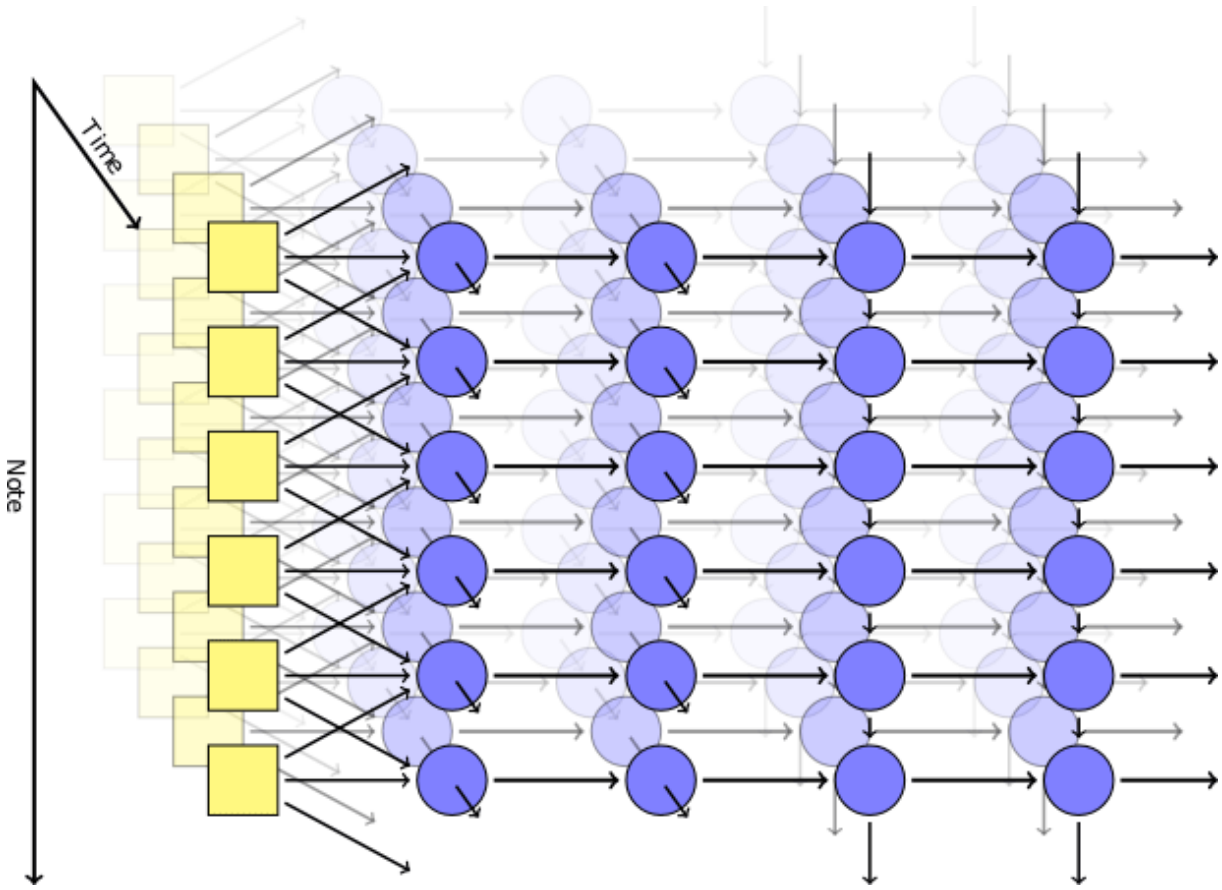


Рис. 8: Модель Tied Parallel Networks [21]

Упрощённая архитектура сети представлена на рис. 8. Каждая нота и её окрестность (по 12 нот с каждой стороны; на рисунке указаны связи для окрестности в 1 ноту с каждой стороны) связана с собственной рекуррентной нейросетью (на рисунке представленной двумя слоями). Все параметры в этих сетях одинаковы для всех нот и таким образом обеспечивают инвариантность вдоль нотной оси; авторы проводят аналогию с обеспечением инвариантности вдоль оси вре-

мени в классических рекуррентных сетях, параметры которых с течением времени неизменны.

Для полифонического сэмплирования используется следующая идея. Рекуррентные связи в классических рекуррентных сетях проносят информацию вдоль оси времени; теперь же, когда есть инвариантность не только вдоль оси времени, но и вдоль оси нот, можно ввести «рекуррентные» связи в измерении нот. Авторы предлагают в первых слоях иметь только классические связи вдоль временной оси аналогическо классическим рекуррентным сетям, а в последних слоях вместо них провести аналогичные связи вдоль оси нот. За счёт этого в финальном слое нейроны, отвечающие за вероятность активации ноты n, получают на вход информацию об активации ноты $n-1$, то есть результат сэмплирования по вероятности, сгенерированной в аналогичной сети для ноты $n-1$. Авторы полагают нумерацию нот от нижних регистров фортепиано к верхним и таким образом генерируют один бинарный вектор за 88 последовательных сэмплирований.

В оригинальной работе в качестве сети с рекуррентными связями использовался однослойный LSTM; в качестве сети со связями вдоль нотной оси использовался стэк из однослойного LSTM и слоя обычных рекуррентных нейронов, где рекуррентные связи, согласно модели, заменены на связи «между» уровнями нот.

Особенность этой модели заключается не только в попытке построить подобие свёрточной архитектуры для музыки, но и в успешном выучивании гармонических зависимостей за счёт громоздкого, но простого «последовательного» подхода. Тот факт, что уловить эти зависимости возможно, генерируя решение по ноте на основании решений по всем предыдущим, означает, что никаких принципиально существенных скрытых переменных в гармониях нет, и имитировать полифонию простыми средствами не удаётся только лишь из-за грубых предположений о независимости нот или иных упрощающих модельных ограничений.

5.7 Performance RNN

Субъективно наиболее приятной на слух моделью полифонической музыки является Performance RNN [30], созданная в рамках проекта Magenta [26] в 2017 году. Основной идеей модели является отказ от сеточной дискретизации, применяющейся при переходе от MIDI-представления к представлению Piano Roll; данные остаются в формате последовательности команд следующего вида:

- 128 команд о зажатии или отпускании каждой из клавиш.
- 100 команд о сдвиге во времени на различные промежутки от 10 миллисекунд до 1 секунды.
- 32 команды о смене громкости нажатия последующих клавиш.

Все 388 команд one-hot кодируются; таким образом, произведение представлено в виде последовательности из 388-мерных бинарных векторов с ровно одной ненулевой позицией на каждом шаге.

Модель обучалась на живых выступлениях пианистов и успешно выучила локальные особенности экспрессивной игры, такие как небольшие замедления и ускорения темпа и непродолжительные увеличения или уменьшения громкости (рис. 9). Как и остальные модели, Performance RNN не улавливает глобальной структуры, но принципиально не предназначен для решения этой задачи исходя из самого представления данных.

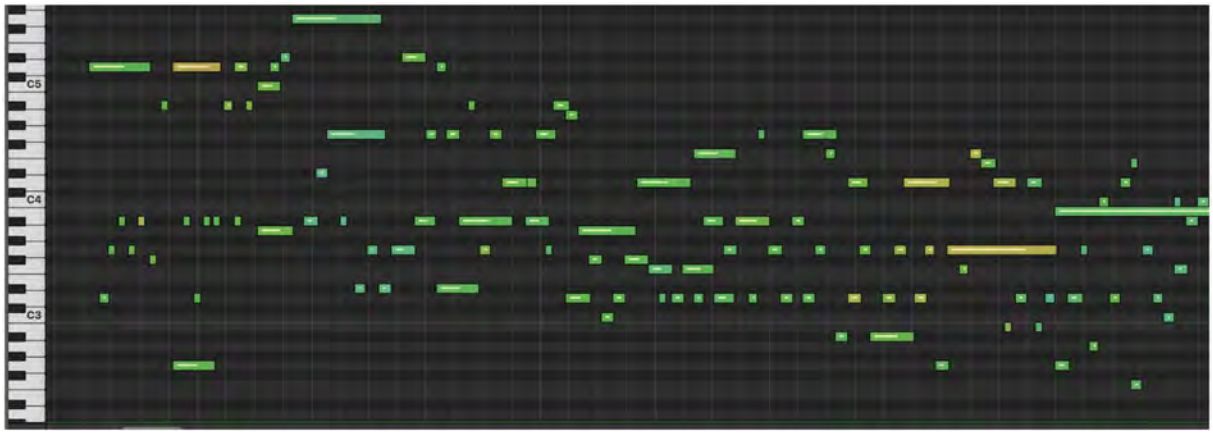


Рис. 9: Результат работы PerformanceRNN; цвет нот соответствует громкости. [30]

Зато за счёт представления сеть в том числе улавливает и воспроизводит при генерации часто встречающиеся комбинации из играемых почти одновременно нот, на слух звучащие как аккорды. Таким образом, такое представление данных отчасти «решает» проблему полифонического сэмплирования.

5.8 BachProp

Идею PerformanceRNN отказаться от векторного бинарного представления и перейти к генерации последовательности команд можно реализовать не только для датасетов живых записей исполнений, в которых с высокой точностью указано время между событиями в миллисекундах, но и для нотного представления музыки, где время исчисляется в долях от некоторого фиксированного эталона.

В модели BachProp [7] (2018) вводится следующее представление музыкального произведения: для каждой ноты указывается три числа:

- dT — время в дискретных временных шагах с предыдущей ноты, принимающее конечное число (21) возможных значений, включая 0.
- T — продолжительность звучания ноты, принимающее такое же конечное число значений.
- P — высота ноты, принимающая 88 возможных значений.

Произведение таким образом представляется в виде массива троек $[dT_n, T_n, P_n]$. Это представление теоретически эквивалентно представлению Piano Roll, хотя

на практике требует дополнительной квантизации исходных MIDI-файлов датасета, чтобы все временные промежутки попадали в заданное фиксированное множество из 21 значений — «пустые события» в таком представлении не предусмотрены.

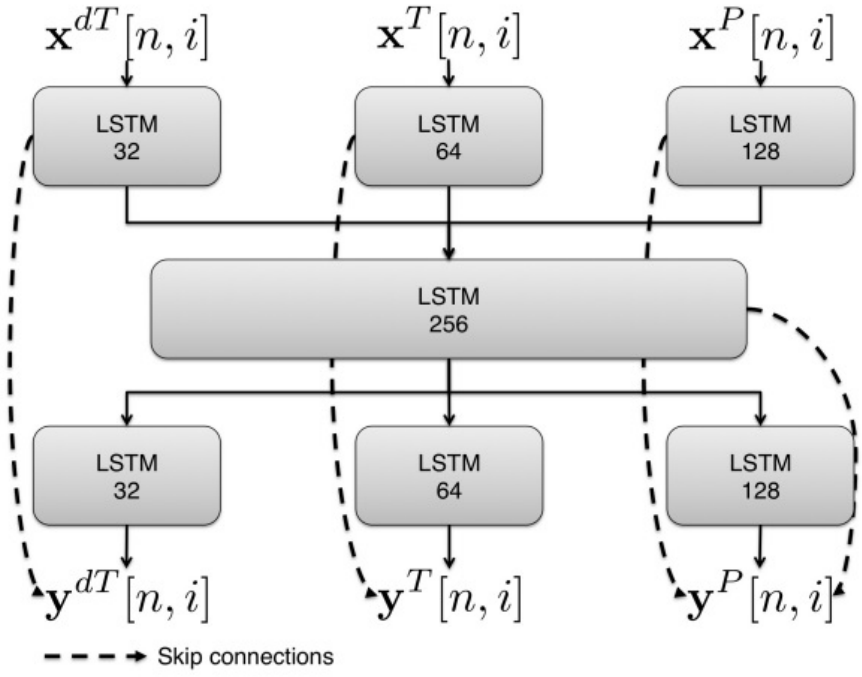


Рис. 10: Архитектура модели BachProp [7]

Моделируются следующие распределения (рис. 10):

$$p(dT_n \mid dT_{n-1}, T_{n-1}, P_{n-1})$$
$$p(T_n \mid dT_n, T_{n-1}, P_{n-1})$$
$$p(P_n \mid dT_n, T_n, P_{n-1})$$

Каждая из трёх компонент подаётся на вход своему независимому LSTM-слою. Выходы трёх LSTM-ов конкатенируются и обрабатываются общим агрегирующим слоем. Выход агрегирующего слоя подаётся на вход следующим персональным для каждой из трёх компонент LSTM-слоям, генерирующим категориальное распределение. Между персональными слоями каждой компоненты проводятся skip connection-связи. На каждом шаге используется выход только одной из трёх персональных LSTM-сетей; по сети выполняется три прохода для генерации одной команды.

Такая архитектура позволяет одновременно учесть особенности каждой из компонент за счёт персональных слоёв, и связать три компоненты между собой за счёт агрегирующего слоя.

Авторы отмечают, что, возможно, такое «покомандное» представление данных отчего-то проще для обучения нейросетей. С одной стороны, в нём теряется

«ось времени», и каждый проход по сети уже не соответствует одному временному шагу. С другой, временные интервалы между событиями теперь являются последовательностью «классов», и моделирование dT является задачей классификации. Когда при работе с Piano Roll сети было необходимо для имитации ритма постоянно выдавать нулевые векторы в качестве ответа, здесь ей достаточно научиться генерировать последовательности классов, что похоже на более простую задачу.

6 Предлагаемая модель

6.1 Общая схема

Модель, построенная в рамках исследований, состоит из нескольких основных модулей:

- Основной модуль генератора, состоящий из LSTM-сети и т. н. механизма внимания.
- Полифонический сэмплер генератора, по выходу основного модуля генерирующий звучащие ноты в следующий момент времени при помощи последовательного сэмплирования из категориальных распределений.
- «Декодер», который позволяет генератору работать со «сжатыми», 12-мерными октавными представлениями вместо исходных 88-мерных. «Кодировщик» при этом представляет собой детерминированную процедуру.

В последующих разделах подробно описываются каждый из данных трёх модулей.

6.2 Сжатие в октаву

В моделях полифонической фортепианной музыки обычно ведётся работа с 88-мерными представлениями x_t, где число 88 объясняется традиционным количеством клавиш на клавиатуре. Такое представление явно не является естественным, и с целью оптимизации вычислительной сложности процесса обучения представление хотелось бы сжать.

Все ноты музыкальных инструментов делятся на диапазоны по 12 нот, называемых октавами. Октавы между собой эквивалентны в том смысле, что соответствующие звуки в них имеют схожую физическую природу. Для сжатого представления будем полагать, что нота звучит, если она звучит хотя бы в одной октаве.

Генерация музыки проходит в рамках такого сжатого 12-мерного представлениями (рис. 11). Это позволяет уменьшить размерность временного ряда более чем в 7 раз. При этом в сжатом представлении теряется такая важная информация, как разделение на мелодию и аккомпанемент (если оно явным образом имелось в исходном произведении), но не теряется информация о гармонических зависимостях, заключающихся в том, какие из 12 нот звучат одновременно.

Декодировщик на вход получает декодированный $x_t \in \{0,1\}^{88}$ и зашифрованный $x'_{t+1} \in \{0,1\}^{12}$, который требуется расшифровать. Выходом сети является $x_{t+1} \in \{0,1\}^{88}$.

Задача подобного декодирования кажется более простой, чем задача сочинения музыки, пусть даже за счёт сжатого представления упрощённую до генерации последовательностей гармоний. Тем не менее, декодировщик должен пом-

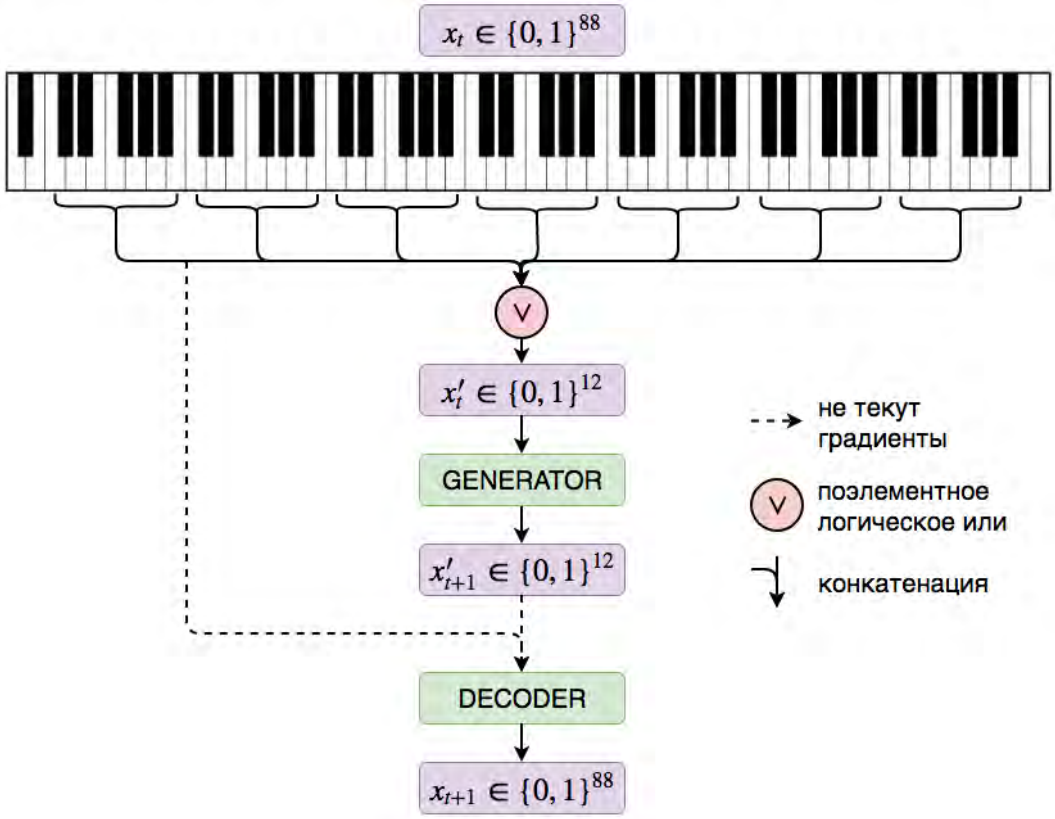


Рис. 11: Общая схема модели: октавы исходного представления объединяются в 12-мерное представление при помощи логического или. Генератор (состоящий из основного модуля и полифонического сэмплера) работает с данными сжатыми представлениями. Выход генератора «декодируется» моделью LSTM-RBM, обучающейся отдельно.

нить, в какой области звукового диапазона «находится» произведение в текущий момент времени, и уметь считать, например, для улавливания паттернов типичных аккомпанементов[9]. Для этого используется однослойная LSTM-сеть.

На выходе «декодировщика» должно быть распределение на бинарных векторах, поэтому используется модель LSTM-RBM, в которой выходом нейросети являются параметры смещения для ограниченной машины Больцмана. При этом, в отличии от модели RNN-RBM, решавшей задачу целиком, здесь размер скрытого слоя можно сделать небольшим, поскольку вероятности звучания одной и той же ноты в разных октавах кажутся уже не столь зависимыми между собой величинами, как, например, в гармониях.

При обучении и при генерации, выходы RBM-а, соответствующие нотам, которые в сжатом представлении указаны как не звучащие, принудительно зану-

[9]обычно представляющие собой повторяющиеся конструкции продолжительностью 4-16 моментов времени

ляются. Таким образом, на каждом временном шаге используется только часть сети.

Пример работы схемы представлен на рис. 18.

6.3 Механизм внимания как дополнительная модель памяти

Основной модуль модели состоит из двуслойной LSTM-сети, выдающей высокоуровневые признаки для полифонического сэмплера. В качестве эксперимента, был добавлен механизм внимания [2] в простейшем виде.

Известно, что LSTM не улавливает глобальные зависимости, тем не менее в определённых ситуациях прогнозируемый вектор x_t может являться модификацией x_{t-p} или точной копией. При этом p обычно является степенью двойки (или, реже, суммой двух степеней).

Одним из возможных решений является подавать сети на вход конкатенацию векторов $\{x_{t-2^k} \mid k \in \{0, 1, \dots K\}\}$. Вектора с отрицательными индексами можно положить нулевыми. Однако, такое решение выглядит громоздко, приводит к существенному увеличению числа параметров в первом слое и увеличивает риск переобучения.

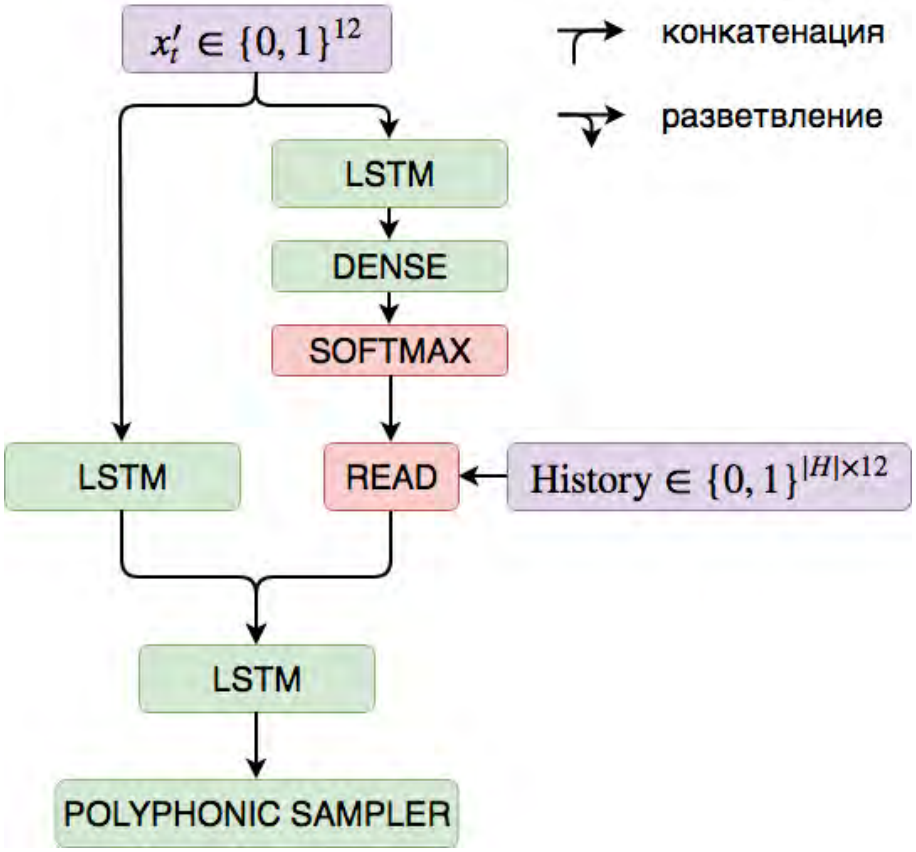


Рис. 12: Основной модуль модели с механизмом мягкого внимания

Механизм внимания позволяет сети самой выбирать, какую часть из имеющегося большого набора данных пустить дальше по сети [29]. Для этого отдель-

ный модуль («контроллер») из однослойного LSTM-а генерирует коэффициенты для выпуклой комбинации векторов $\{x_{t-h} \mid k \in H\}$, где H — некоторое фиксированное множество потенциальных периодов иерархической структуры[10]. Вся сеть при этом получает на вход только x_{t-1} и время в дискретном представлении.

Обычно, в механизмах внимания считанные данные из истории подаются вместе в первый слой основного модуля. Возможно подавать считанные данные и в более глубокие скрытые слои. В используемой схеме считанный вектор подаётся во второй LSTM-слой; таким образом, считывание из истории и первый скрытый LSTM-слой просчитываются «параллельно». Интуиция такой схемы заключается в том, чтобы независимо учесть глобальные и локальные зависимости, а затем их агрегировать.

Во время предварительных экспериментов на небольшом датасете Piano-midi[11] механизм внимания уловил[12] явные (не содержащие никаких модификаций) повторяющиеся фрагменты музыки в обучающей выборке. Воспроизвести этот эффект на этапе генерации не удалось (копирование происходило не более, чем на 4-8 моментов времени). Для обучения финальной модели (см. раздел 7) использовался более большой датасет с более «сложной» музыкой, практически не содержащей подобных явных повторов, однако, даже если механизм внимания не помогает улавливать иерархическую структуру произведений, он становится способом автоматического определения нужных статистик по звучавшим в истории нотам.

6.4 Поголосовое полифоническое сэмплирование

12-мерное представление предоставляет потенциальную возможность свести задачу к генерации категориального распределения на множестве всех $2^{12} = 4096$ возможных значений бинарного вектора как независимых событий (что, казалось бы, позволяет свести задачу к задаче классификации). Однако, такая модель должна сама уловить связи между возможными ответами, а при настройке весов не будут поощряться ответы, «близкие» к правильному (например, выдача двух верных нот из трёх). Отчасти, такую задачу можно рассматривать как задачу структурного обучения, и каким-нибудь образом задать функцию потерь с, например, учётом расстояния между различными аккордами. Построение разумной метрики на множестве аккордов, однако, является нетривиальной задачей, и её решения на сегодняшний день являются сугубо эвристическими.

Схемы поголосового сэмплирования, рассмотренные в разделе 5.2, предлагают последовательно принимать некоторые решения на основании предыдущих

[10] возможно в качестве H взять все вектора из истории вплоть до некоторого момента, то есть $x_{t-1} \dots x_{t-K}$ для некоторого K

[11] http://www.piano-midi.de/

[12] выдавал вырожденную выпуклую комбинацию, таким образом подавая на вход второму LSTM-слою готовый «правильный ответ»

решений, таким образом сводя задачу сэмплирования из распределения на взаимозависимых бинарных векторах к последовательному сэмплированию из категориальных. Основной проблемой такого подхода является проблема обучения подобной модели, где, для получения распределения $p(x_{t+1} \mid x_t)$, по распределениям промежуточных решений необходимо было проводить интегрирование.

Рассмотрим следующую схему (для удобства, для некоторого фиксированного момента времени). Первый голос, нейросеть, получающая на вход выходы F основного модуля, генерирует категориальное распределение $p(n^1 \mid F)$ на множестве следующих нот и паузы. Из данного распределения сэмплируется $n^1 \sim p(n^1 \mid F)$ и вместе с F подаётся на вход второму голосу, выдающему категориальное распределение $p(n^2 \mid n^1, F)$, и так далее. Считая, что нота звучит, если её засэмплировал хотя бы один голос, можно подавать на вход третьему голосу $n^1 \cup n^2$. Обозначим число голосов за V, тогда вероятность звучания каждой из нот n_m может быть вычислена по формуле, аналогично формуле (11):

$$p(n_m \mid F) = p(n^1 = m \mid F) + p(n^1 \neq m \mid F)\Big[p(n^2 = m \mid F) + \cdots + \\ + p(n^{v-1} \neq m \mid F)p(n^V = m \mid F)\Big]$$

Подставим в это выражение маргиналы, представленные как мат. ожидания:

$$p(n^v = i \mid F) = \iint\limits_{n^1 \dots n^{v-1}} p\left(n^v \mid \bigcup_i^{v-1} n^i, F\right) p(n^1 \cdots n^{v-1} \mid F) dn^1 \cdots dn^{v-1} = \\ = \mathbb{E}_{p(n^1 \dots n^{v-1} \mid F)} p\left(n^v \mid \bigcup_i^{v-1} n^i, F\right) dn^1 \cdots dn^{v-1}$$

Получим:

$$p(n_m \mid F) = p\left(n^1 = m \mid F\right) + \\ + p\left(n^1 \neq m \mid F\right) \mathbb{E}_{n^1 \mid F}\Big[p\left(n^2 = m \mid n^1, F\right) + \\ + p\left(n^2 \neq m \mid n^1, F\right) \mathbb{E}_{n^2 \mid n^1, F}\Big[p\left(n^3 \mid n^1 \cup n^2, F\right) + \cdots + \\ + p\left(n^{v-1} \neq m \mid \bigcup_i^{v-2} n^i, F\right) \mathbb{E}_{n^{V-1} \mid n^1 \dots n^{V-2}, F}\Big[p\left(n^V = m \mid \bigcup_i^{V-1} n^i, F\right)\Big] \cdots\Big]\Big]$$

Заменим мат.ожидания на Монте-Карло оценку по одному сэмплу:

$$\hat{n}^1 \sim p(n^1 \mid F); \quad \hat{n}^2 \sim p(n^2 \mid \hat{n}^1, F); \quad \dots \quad \hat{n}^{V-1} \sim p\left(n^{V-1} \mid \bigcup_i^{V-2} \hat{n}^i, F\right)$$

$$p(n_m \mid F) = p\left(n^1 = m \mid F\right) + \left(1 - p\left(n^1 = m \mid F\right)\right)\Big[p\left(n^2 = m \mid \hat{n}^1, F\right) + \\ + \left(1 - p\left(n^2 = m \mid \hat{n}^1, F\right)\right))p\left(n^3 \mid \hat{n}^1 \cup \hat{n}^2, F\right) + \cdots + \\ + \left(1 - p\left(n^{V-1} = m \mid \bigcup_i^{V-2} \hat{n}^i, F\right)\right) p\left(n^V = m \mid \bigcup_i^{V-1} \hat{n}^i, F\right)\Big]$$

Такая оценка корректна и является несмещённой, но потенциально обладает высокой дисперсией. На практике куда более существенной проблемой оказывается отсутствие вероятности засэмплировать паузу в функционале потерь: действительно, типичным выбор функции потерь является кросс-энтропия:

$$-\sum_m [n_m \log p(n_m) + (1 - n_m) \log(1 - p(n_m))] \to \min$$

В этом функционале присутствуют вероятности засэмплировать m-ую ноту для каждого голоса, однако отсутствуют вероятности засэмплировать паузу, или, иначе говоря, вероятности голосов отказаться от сэмплирования какой-либо ноты. При проведении экспериментов такая схема приводила к тому, что первые $V - 1$ голосов генерировали отказ от сэмплирования (выдавали паузу), а последний голос пытался «уловить» одну ноту из правильного ответа. Попытки исправить этот эффект добавлением к функции потерь искусственного слагаемого в виде штрафа на паузу к существенным изменениям не привели.

Более «традиционным» нейросетевым подходом было бы вместо спецсимвола паузы вводить специсимвол перехода к следующему моменту времени, таким образом позволяя генерировать произвольное число нот одновременно. Это обязывает каждый голос генерировать либо новую ноту, либо заканчивать процесс сэмплирования, что решает главную проблему рассматриваемой схемы — возможность голосов не передавать последующим голосом никакой новой информации. Однако, в надежде на то, что каждый из голосов будет генерировать что-то подобное мелодии, хотелось бы оставить возможность голосов не генерировать ноту.

Вместо этого голосам предлагается генерировать категориальное распределение не на множестве нот и спецсимвола-паузы $\mathcal{N} \cup \{pause\}$, а на множестве $\mathcal{N} \times \{0, 1\}$. Данное множество является множеством возможных пар <нота, решение>, где решение — звучит ли нота или нет, 0 или 1, причём решение окончательное. В отличии от предыдущей схемы в такой конструкции есть симметрия относительно нулей и единиц в генерируемых бинарных векторах, и каждый голос обязан сгенерировать новую информацию для последующих голосов, выбрав новую ноту для звучания или же бана. Факт, что некоторая нота *не* звучит, как и факт звучания, влияет на распределения по другим нотам, и несёт в себе содержательную информацию.

Условие, что принимаемое решение окончательно, достигается за счёт принудительного зануления выходов нейросети, соответствующего нотам, по которым решение уже принято. Полная схема представлена на рис. 13.

Все голоса моделируются однослойной LSTM-сетью с общими параметрами (весами). Таким образом, такая сеть моделирует принятие следующего решения по известным уже принятым решениям и высокоуровневым признакам, полученным на выходе основного модуля модели.

При обучении входом каждого голоса должны быть истинные решения по нотам из выборки. Однако, определение того, на основании какой «части» правиль-

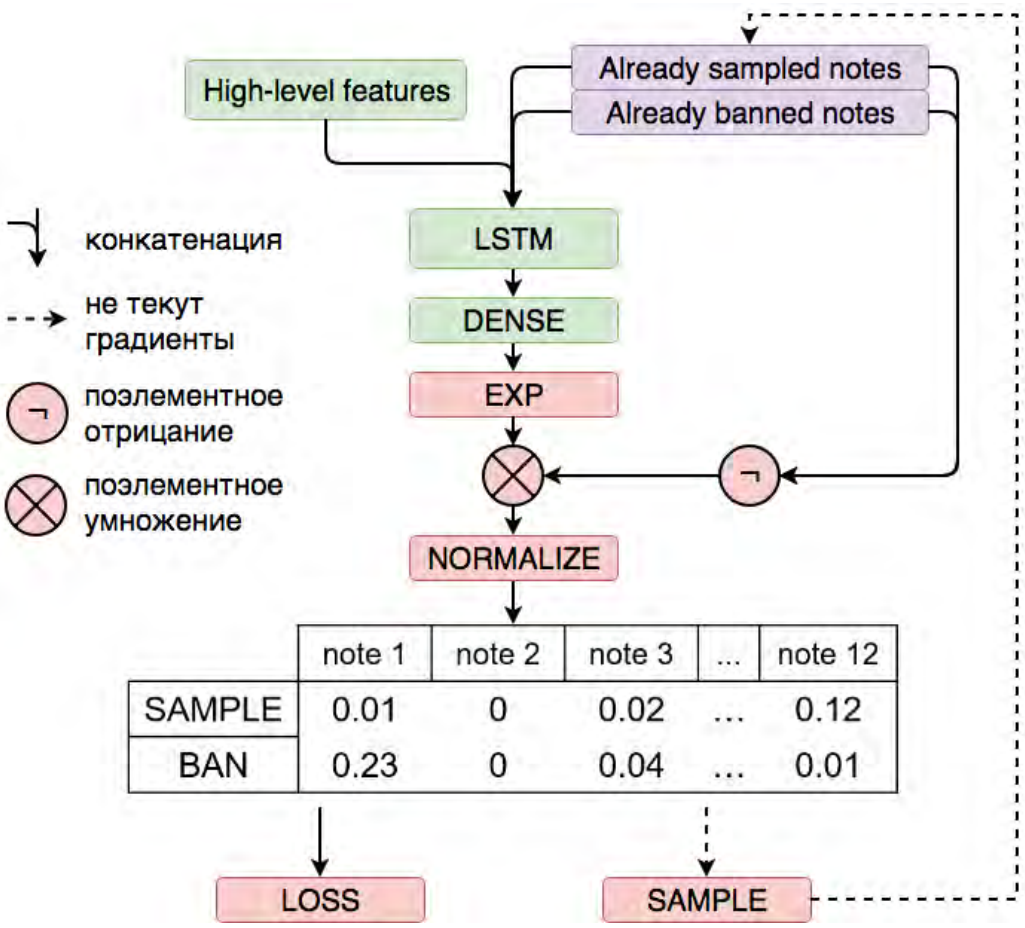


Рис. 13: Модель поголосового полифонического сэмплера. Каждый «голос», представляющий собой однослойный LSTM, на основании решений, принятых предыдущими голосами, выдаёт категориальное распределение на множестве оставшихся возможных решений.

ного ответа очередной голос будет выдавать следующее распределение, неоднозначно. Поскольку на этапе генерации голоса обрабатывают решения предыдущих голосов, необходимо, чтобы на этапе обучения «части» правильного ответа были похожи на те, что вероятно сгенерируют предыдущие голоса. Экспериментально получено, что схема обучается лучше, если подавать на вход последующим голосам ответы для тех нот, решения по которым (верные или неверные) уже приняли предыдущие голоса. Тогда, в случае неверного решения по некоторой ноте n_m, последующие голоса уже не могут «исправлять» эту ошибку, поскольку их выходы для этой ноты будут занулены, и в кросс-энтропии этой ноте будет соответствовать высокий штраф.

7 Эксперименты

7.1 Данные и их предобработка

Для обучения модели использовался общедоступный сборник MIDI-файлов TheGreats[13] со следующей предобработкой:

- Отфильтрованы произведения, содержащие фрагменты не в размере, производном от 4/4 (2/4, 4/8 и пр.). Это позволяет использовать фиксированное множество H потенциальных периодов для механизма внимания; смена этого множества позволила бы обучиться на производных, например, от 3/4, но произведений в таком размере в датасете меньше.
- Отфильтрованы MIDI-файлы, содержащие больше двух треков с нотной информацией. Треки — понятия MIDI-протокола, могут использоваться для хранения как партий отдельных инструментов, так и дополнительной мета-информации. Наличие больше двух треков с нотной информацией означает, что MIDI-файл является нотным представлением оркестрового произведения. Два трека в фортепианных произведениях может использоваться для отдельных партий правой или левой руки, или для ансамблей в четыре руки.
- Проведена дискретизация по 4 дискретных момента времени на одну долю, или 16 моментов времени на такт при размере 4/4.

Всего в датасете после фильтрации осталось 1392 произведения средней продолжительности в 1088 дискретных моментов времени.

7.2 Параметры модели и реализация

Финальная модель была реализована на фреймворке PyTorch и обучена на 8-ядерном процессоре[14] Intel Core i7 со следующими параметрами:

Параметры сэмплирования батчей	
Количество фрагментов в батче	140
Продолжительность фрагментов	256
Шаг, с которым выбирались начала фрагментов	32
Размерность бинарного вектора t, времени, дополнительно подававшегося на вход	6

[13] https://www.classicalarchives.com/midi.html

[14] обучение на процессоре стало возможным за счёт перехода к «октавному» 12-мерному представлению

Оптимизатор [22]	
Оптимизатор основного модуля	Adam
Learning Rate оптимизатора	0.01
ε оптимизатора	1e-8
Оптимизатор декодера	Adam
Параметры основного модуля	
Размер первого скрытого LSTM-слоя	100
Размер второго скрытого LSTM-слоя	130
Размер LSTM-контроллера	100
Периоды $H = \{4, 8, 16, 32, 48, 64, 96, 128\}$	
Параметры полифонического сэмплера	
Количество голосов V	5
Размер скрытого LSTM-слоя	100
Параметры декодера	
Размер скрытого LSTM-слоя	100
Размер скрытого слоя RBM	36
Итераций сэмплирования по Гиббсу	1

В результате выделения фрагментов из произведений, была получена выборка из 48427 пересекающихся фрагментов. Обучение происходило на первых 40000 фрагментах, соответствующих первым 1150 произведениям датасета. Остальные фрагменты (из, соответственно, других произведений) использовались для валидации.

При генерации случайных батчей производилась аугментация, при которой произведения циклически сдвигались по оси нот на случайное от от -6 до 5 значение. Это желательно, поскольку архитектура модели никак не учитывает инвариантность относительно циклического сдвига.

7.3 Результаты

Обучение модели до приблизительной сходимости занимает на процессоре порядка 12 часов. На рис. 14 представлено качество на обучении и тесте в течение 2400 итераций основного модуля, работающего с «октавным» представлением. Несмотря на отсутствие какой-то регуляризации в модели (техника дропаута или батч-нормализации не применялась), переобучения на датасете не наблюдается.

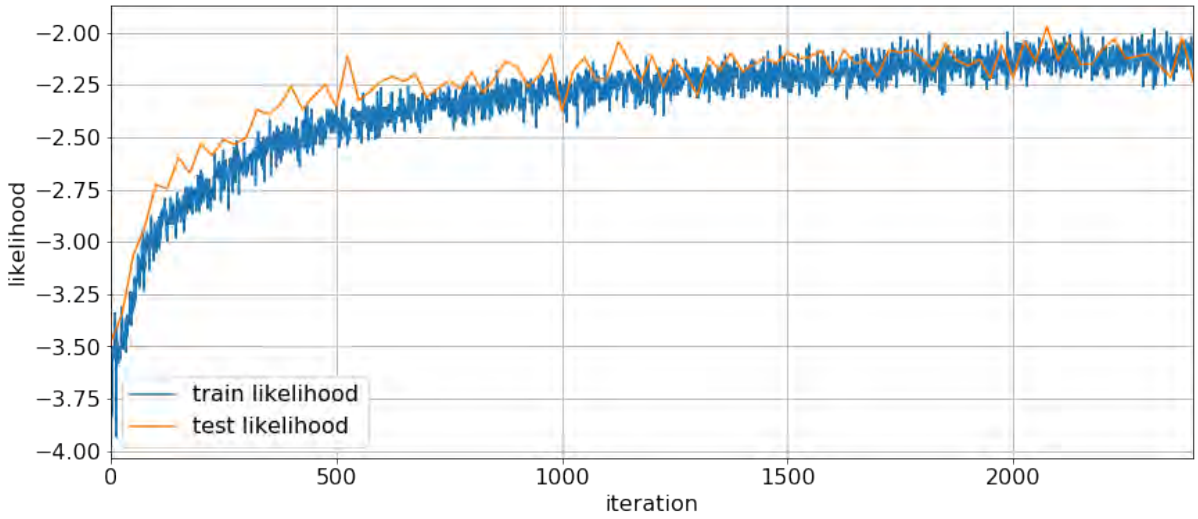


Рис. 14: Процесс обучения: правдоподобие на обучающей выборке и валидации, состоящей из отложенных песен.

Сэмплы модели можно послушать в репозитории с кодом и обученной моделью[15]. Ноты случайного фрагмента приведены на рис. 19.

Промежуточная версия модели загружена на платформу CrowdAI в открытом соревновании AI-generated Music Challenge[16]. По состоянию на 26.04.18, модель имеет рейтинг TrueSkill 17.455 на основе 78 сравнений случайных фрагментов часового сэмпла с фрагментами сэмплов моделей других участников и занимает пятое место из 24 участников.

Результаты работы были доложены на XXV Международной научной конференции Ломоносов-2018 и опубликованы в сборнике тезисов [1].

7.4 Обсуждение

Сгенерированная музыка, схоже с результатами моделей RNN-RBM, Classifying VAE и Tied Parallel Network, представляет собой «текстуру», или, в каком-то смысле, усреднённую, «размытую» музыку.

Достаточно хорошо слышно соблюдение гармоний: в сэмплах постоянно встречаются трезвучия различных тональностей, часто с валидными переходами между ними. Это достигается за счёт схемы поголосового полифонического сэмплирования, пример работы которого представлен на рис. 15.

Первый голос наиболее активен и явно слишком редко выдаёт паузу, что приводит к обеднению ритмического разнообразия. При этом, если первый голос не генерирует ноту, последующие голоса также не выбирают генерацию ноты. Для последующих голосов такого эффекта не наблюдается: таким образом, послед-

[15] https://github.com/FortsAndMills/MusicGeneration

[16] https://www.crowdai.org/challenges/ai-generated-music-challenge

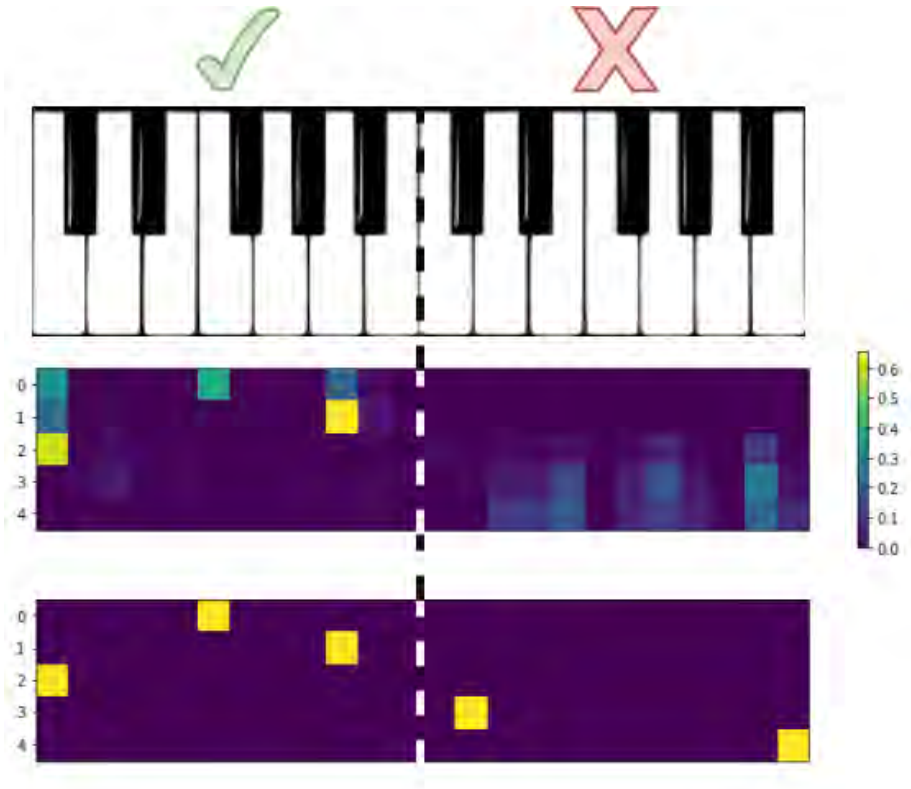


Рис. 15: Пример принятия решения (генерация трезвучия фа мажор) пятью голосами в фиксированный момент времени. Сверху — вероятностные распределения, генерированные голосами. Снизу — сэмплы, из них полученные. Каждый голос при генерации своего распределения получает на вход сэмплы предыдущих голосов.

ние голоса пользуются от предыдущих информацией о бане некоторых нот (рис. 16).

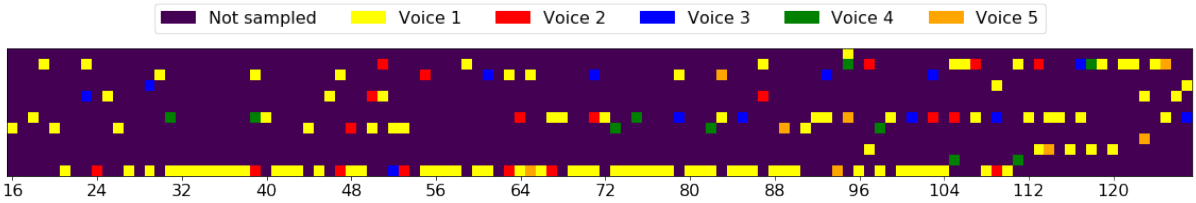


Рис. 16: Распределение нот по голосам в случайном сэмпле

Ощутимо «соблюдение» тональностей: практически не звучат ноты вне текущей гармонии, все семь нот которой достаточно активно используются. При этом «наборы из семи нот» являются классическими ладами вроде мажора и разновидностей минора, несмотря на то, что модель не использует никакой априорной информации из музыкальной теории. Довольно редка фальшь, зачастую при генерации неожиданных нот сеть пытается это «использовать» и провести переход в другую тональность, однако такие моменты происходят в довольно случайное

время. Возможно, такие случаи вызваны случайным сэмплированием маловероятных нот.

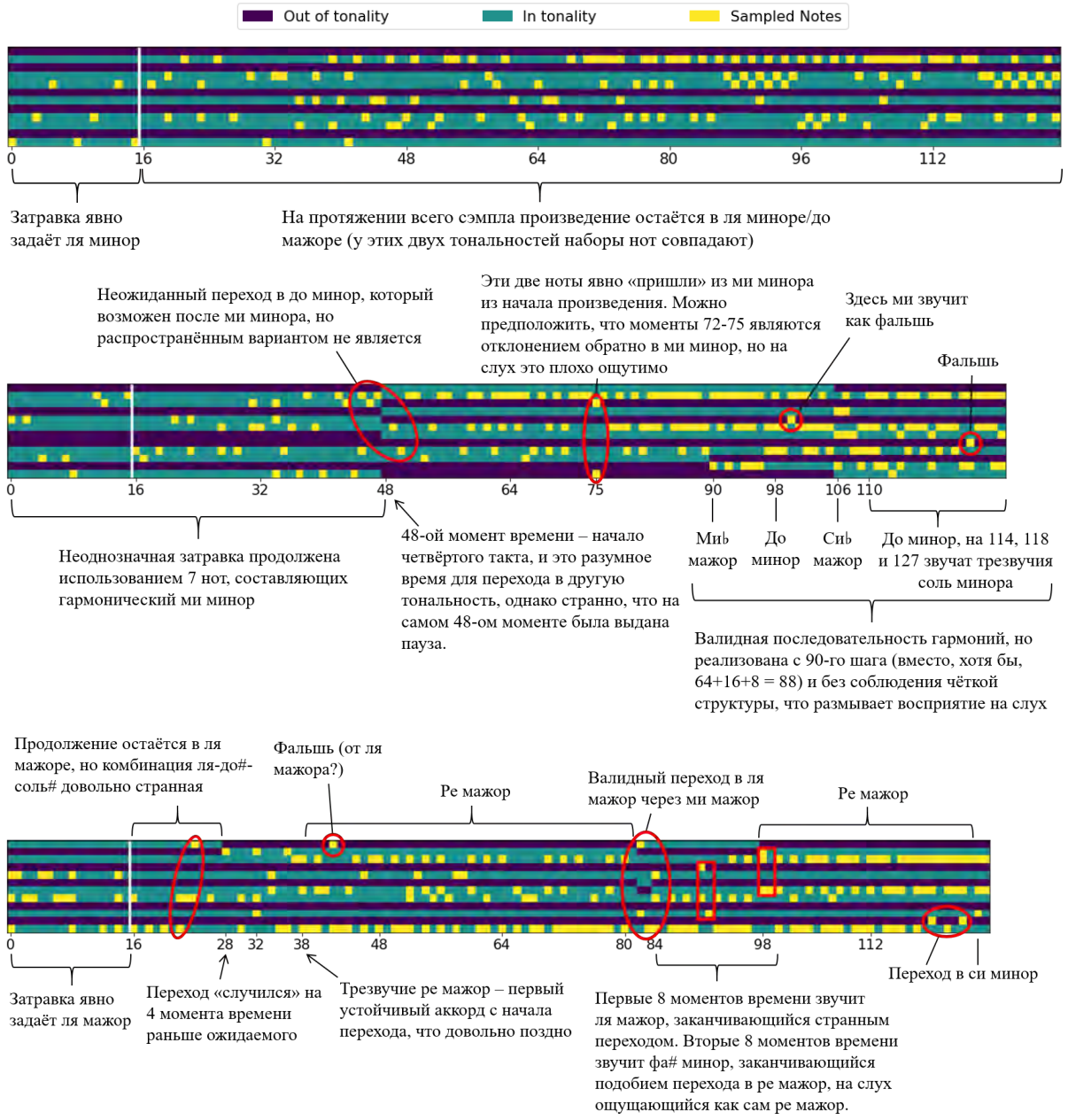


Рис. 17: Ручной анализ трёх случайных сэмплов на предмет использовавшихся гармоний, фальши и переходов между гармониями. Тональность можно примерно определить как набор из семи нот, «установить» которую можно, сыграв «рядом» три из семи основных нот тональности. Данный анализ совершенно неоднозначен, и плохо описывает качество звучания сэмплов (то, что ноты звучат «внутри» тональности, не гарантирует приятности звучания, и наоборот), но показывает, что генератор предпринимает разумные действия и скорее всего следит за текущей тональностью в том или ином виде.

На рис. 17 проведён анализ трёх случайных сэмплов модели на предмет использованных гармоний и переходов между ними. Модель как держится в заданной текущей тональности, (например, в примере 1 рис. 17, или в тактах 1-11 нотного примера с рис. 19), так и генерирует последовательности гармоний. Поскольку модель не ограничена каким-то выделенным набором возможных аккордов, эти последовательности могут быть сколь угодно нетривиальны: если в примере 2 рис. 17 последовательность гармоний классическая, в такте 12 нотного примера с рис. 19 сгенерировалась новая уникальная комбинация.

Лаконично продолжить затравку (продолжительностью в 16 временных шагов) модели чаще всего не удаётся, но из затравки выделяется тональность и простейшие рисунки ритма. Ритм сэмплов достаточно простой, модель научилась пользоваться восьмыми (генерировать ноты через временной шаг) и шестнадцатыми нотами (генерировать ноты каждый шаг), также встречается т. н. «пунктирный ритм» (нота-пауза-пауза-нота). Модель довольно редко сменяет ритм, и чаще всего меняет в сторону упрощения. Скорее всего, сеть не научилась определять, когда стоит использовать более сложные ритмические комбинации, и не научилась переключаться в них. Громоздким решением было бы обучение ещё одного модуля для генерации ритма (бинарной последовательности).

В нотном примере видны конструкции шестнадцатая-восьмая-шестнадцатая (соответствующие «нота-нота-пауза-нота»), стабильно появляющиеся на временных шагах кратности четыре. Этот ритм в данном сэмпле явно пришёл из затравки, которая была взята из датасета и скорее всего является примером неудачной дискретизации. В оригинальном произведении в этом такте скорее всего использовались т. н. триоли (длительности таких нот составляют 8/3 или 4/3 временных шагов), которые при дискретизации превратились в такой странный ритм.

Как видно из нотного примера, в итоговом результате нет выраженного аккомпанемента и явной мелодии. При генерации полифонической музыки неявной мелодией в сэмплах будут верхние звучащие ноты, поэтому в данной модели «ответственным» за мелодию становится декодер, который должен каким-то образом выделить её из сжатого 12-мерного представления и «поместить» в верхнюю часть фортепианной клавиатуры. В принципе, возможно построение какой-то модели, в которой будет выделенный голос, специально обучаемый как генератор главной мелодии произведения, однако во многих произведениях разделение на «мелодию-аккомпанемент» выражено неявно.

Хотя сейчас содержимое нижних октав (обычно играющее роль «аккомпанемента»), как видно из нотного примера, выглядит довольно хаотично, это следствие использования сжатого представления и неидеального «декодера». Преимуществом же сжатого представления является не столько уменьшение временных затрат (позволившее обучить модель на процессоре за разумное время), но и возможность сети «запоминать» подмножества допустимых нот в рамках текущей гармонии — при использовании полного 88-октавного представления количество нот одной гармонии составляет около 50. При этом, судя по некоторым

фальшам, «пришедших» из прошедших гармоний (с примеров рис. 17) сеть использовала механизм внимания в качестве источника статистической информации о количестве звучавших в последнее время нот.

8 Заключение

В работе была рассмотрена задача генерации музыки в одной из наиболее общих постановок. Было проведено исследование препятствий на пути к полноценному генератору музыкальных произведений. Рассмотрены подходы к обходу недостатков state-of-art генеративных моделей последовательностей, мешающих улавливанию музыкальных зависимостей.

В рамках практического исследования был построен генератор полифонической музыки, исследующий новые возможные варианты решения открытых проблем. Был предложен переход к октавному представлению произведений для сужения размерности пространства данных, позволяющий облегчить вычислительную сложность обучения модели, и способ «расшифровать» это представление. Была предложена новая схема сэмплирования полифонии, в рамках поголосового подхода способная улавливать гармонические зависимости без введения дополнительных ограничений или априорной эвристической информации об устройстве музыки.

На основе предложенной схемы был реализован и обучен генератор полифонической музыки.

Список литературы

[1] Иванов, С. М. Нейросетевая генерация полифонической музыки // Ломоносов-2018: сборник тезисов XXV международной научной конференции студентов, аспирантов и молодых ученых: секция «Вычислительная математика и кибернетика». — Издательский отдел факультета ВМК МГУ, 2018. — С. 110–111.

[2] Bahdanau, D. Neural machine translation by jointly learning to align and translate / Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio // arXiv preprint arXiv:1409.0473. — 2014.

[3] Boulanger-Lewandowski, N. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription / Nicolas Boulanger-Lewandowski, Yoshua Bengio, Pascal Vincent // arXiv preprint arXiv:1206.6392. — 2012.

[4] Bretan, M. A unit selection methodology for music generation using deep neural networks / Mason Bretan, Gil Weinberg, Larry Heck // arXiv preprint arXiv:1612.03789. — 2016.

[5] Briot, J.-P. Deep learning techniques for music generation-a survey / Jean-Pierre Briot, Gaëtan Hadjeres, François Pachet // arXiv preprint arXiv:1709.01620. — 2017.

[6] Chu, H. Song from pi: A musically plausible network for pop music generation / Hang Chu, Raquel Urtasun, Sanja Fidler // arXiv preprint arXiv:1611.03477. — 2016.

[7] Colombo, S. F. Bachprop: A trainable generative model of music scores / Submitter Florian Colombo, Wulflram Gerstner.

[8] Deep learning / Ian Goodfellow, Yoshua Bengio, Aaron Courville, Yoshua Bengio. — MIT press Cambridge, 2016. — Vol. 1.

[9] Dumoulin, V. A guide to convolution arithmetic for deep learning / Vincent Dumoulin, Francesco Visin // arXiv preprint arXiv:1603.07285. — 2016.

[10] Duncan, A. Combinatorial music theory / Andrew Duncan // Audio Engineering Society Convention 89 / Audio Engineering Society. — 1990.

[11] Eck, D. Finding temporal structure in music: Blues improvisation with lstm recurrent networks / Douglas Eck, Juergen Schmidhuber // Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on / IEEE. — 2002. — P. 747–756.

[12] Generative adversarial nets / Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza et al. // Advances in neural information processing systems. — 2014. — P. 2672–2680.

[13] Goel, K. Polyphonic music generation by modeling temporal dependencies using a rnn-dbn / Kratarth Goel, Raunaq Vohra, JK Sahoo // International Conference on Artificial Neural Networks / Springer. — 2014. — P. 217–224.

[14] Graves, A. Neural turing machines / Alex Graves, Greg Wayne, Ivo Danihelka // arXiv preprint arXiv:1410.5401. — 2014.

[15] Hadjeres, G. Deepbach: a steerable model for bach chorales generation / Gaëtan Hadjeres, François Pachet // arXiv preprint arXiv:1612.01010. — 2016.

[16] Hennig, J. A. A classifying variational autoencoder with application to polyphonic music generation / Jay A Hennig, Akash Umakantha, Ryan C Williamson // arXiv preprint arXiv:1711.07050. — 2017.

[17] A hierarchical recurrent neural network for symbolic melody generation / Jian Wu, Changran Hu, Yulong Wang et al. // arXiv preprint arXiv:1712.05274. — 2017.

[18] Hinton, G. E. Training products of experts by minimizing contrastive divergence / Geoffrey E Hinton // Neural computation. — 2002. — Vol. 14, no. 8. — P. 1771–1800.

[19] Hinton, G. E. A fast learning algorithm for deep belief nets / Geoffrey E Hinton, Simon Osindero, Yee-Whye Teh // Neural computation. — 2006. — Vol. 18, no. 7. — P. 1527–1554.

[20] Hochreiter, S. Long short-term memory / Sepp Hochreiter, Jürgen Schmidhuber // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.

[21] Johnson, D. D. Generating polyphonic music using tied parallel networks / Daniel D Johnson // International Conference on Evolutionary and Biologically Inspired Music and Art / Springer. — 2017. — P. 128–143.

[22] Kingma, D. P. Adam: A method for stochastic optimization / Diederik P Kingma, Jimmy Ba // arXiv preprint arXiv:1412.6980. — 2014.

[23] Kingma, D. P. Auto-encoding variational bayes / Diederik P Kingma, Max Welling // arXiv preprint arXiv:1312.6114. — 2013.

[24] Musegan: Symbolic-domain music generation and accompaniment with multi-track sequential generative adversarial networks / Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, Yi-Hsuan Yang // arXiv preprint arXiv:1709.06298. — 2017.

[25] One-shot learning with memory-augmented neural networks / Adam Santoro, Sergey Bartunov, Matthew Botvinick et al. // arXiv preprint arXiv:1605.06065. — 2016.

[26] Waite, E. Project magenta. — 2016.

[27] Radford, A. Unsupervised representation learning with deep convolutional generative adversarial networks / Alec Radford, Luke Metz, Soumith Chintala // arXiv preprint arXiv:1511.06434. — 2015.

[28] Roberts, A. Hierarchical variational autoencoders for music / Adam Roberts, Jesse Engel, Douglas Eck. — 2017.

[29] Show, attend and tell: Neural image caption generation with visual attention / Kelvin Xu, Jimmy Ba, Ryan Kiros et al. // International Conference on Machine Learning. — 2015. — P. 2048–2057.

[30] Simon, I. Performance rnn: Generating music with expressive timing and dynamics. — https://magenta.tensorflow.org/performance-rnn. — 2017.

[31] Temperley, D. The cognition of basic musical structures / David Temperley. — MIT press, 2004.

[32] Todd, P. M. A connectionist approach to algorithmic composition / Peter M Todd // Computer Music Journal. — 1989. — Vol. 13, no. 4. — P. 27–43.

[33] Walder, C. Modelling symbolic music: Beyond the piano roll / Christian Walder // Asian Conference on Machine Learning. — 2016. — P. 174–189.

[34] Werbos, P. J. Backpropagation through time: what it does and how to do it / Paul J Werbos // Proceedings of the IEEE. — 1990. — Vol. 78, no. 10. — P. 1550–1560.

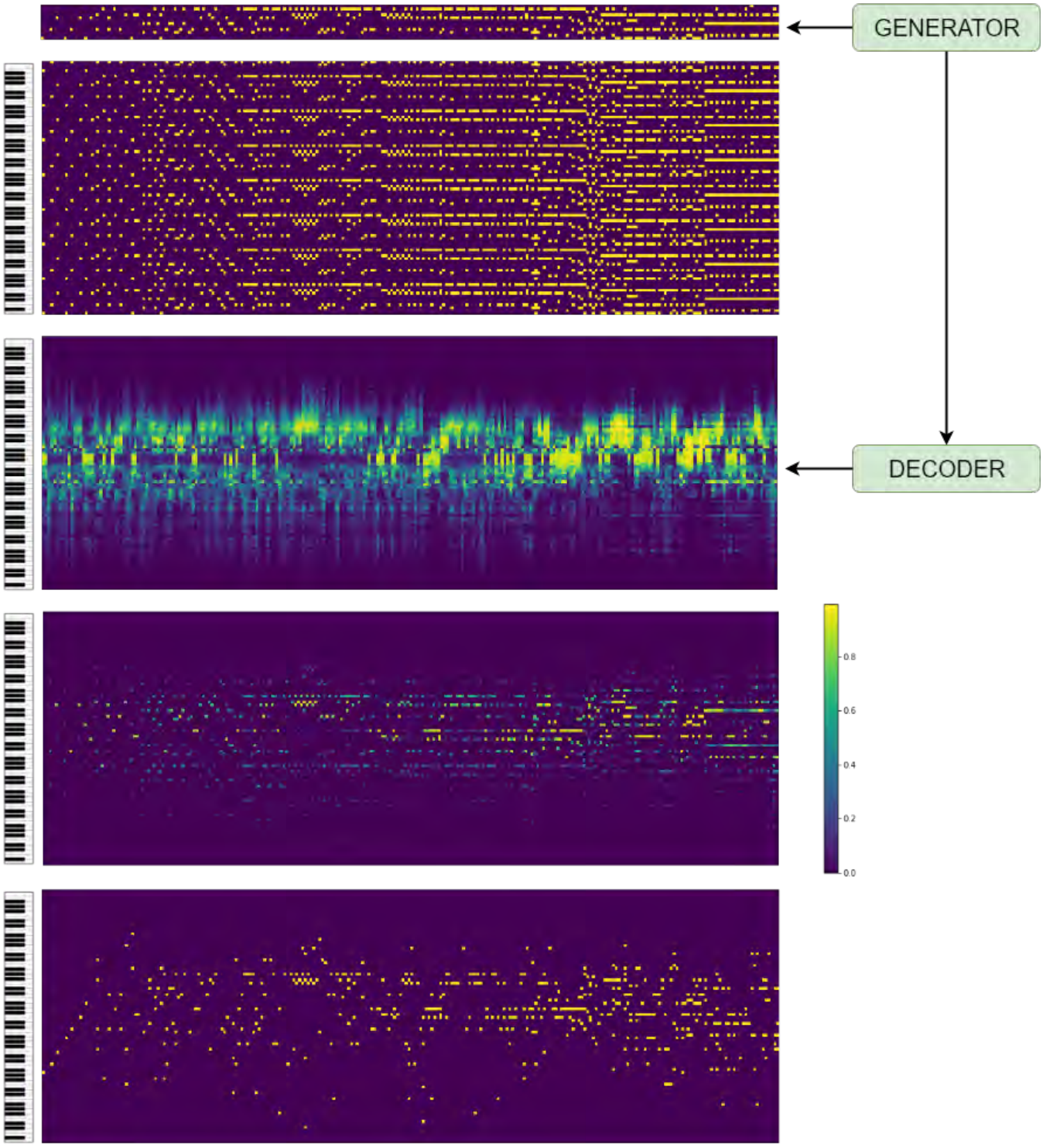


Рис. 18: Процесс генерации в модели. Сверху вниз: генератор сочиняет музыку в рамках «октавного» представления. Составляется маска, какие ноты могут звучать, для всей фортепианной клавиатуры. Сжатое представление подаётся в декодер, выдающий вероятности звучания всех нот. Распределение домножается на маску. Из полученного распределения сэмплируется окончательный результат.

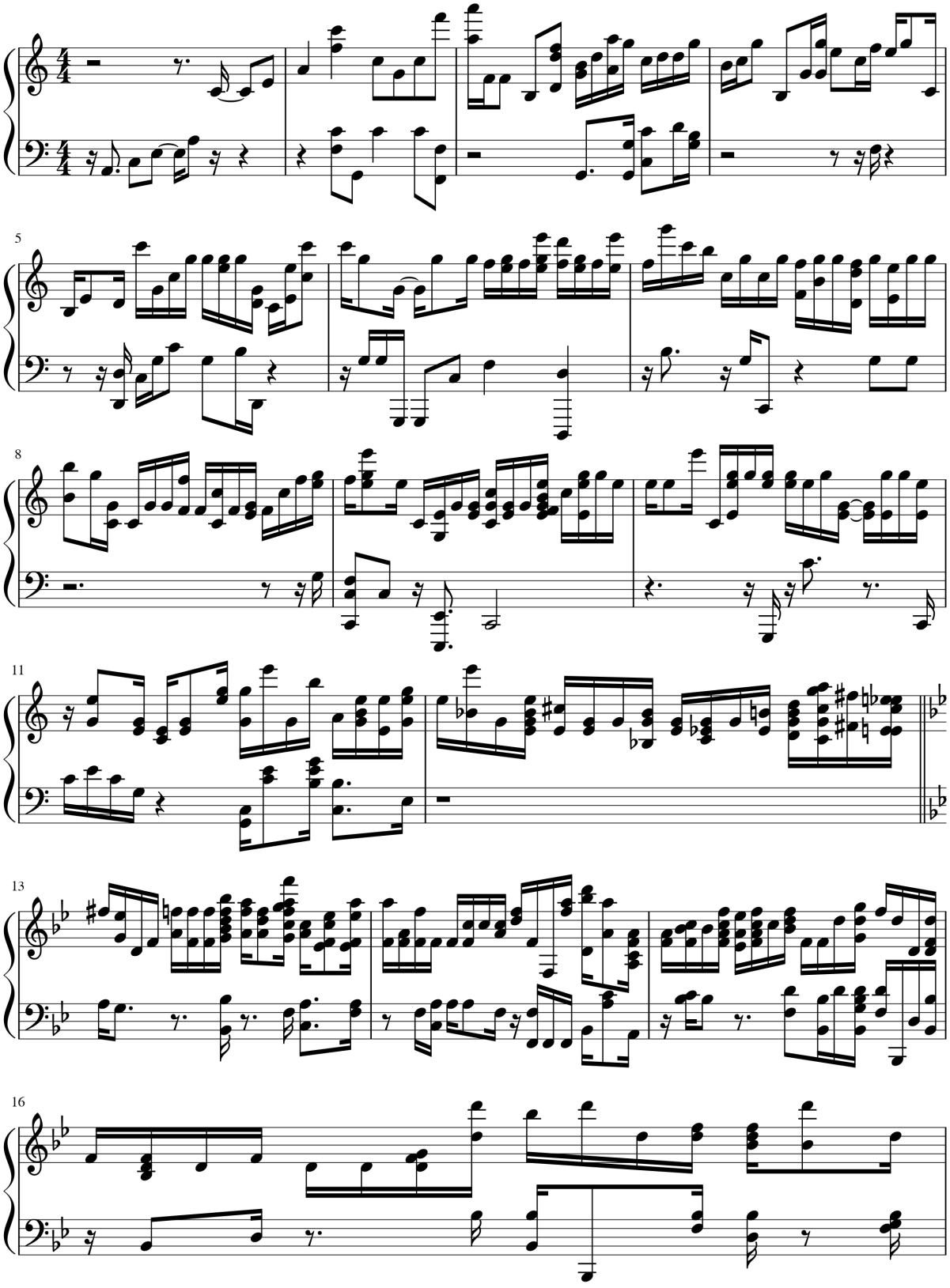


Рис. 19: Ноты сгенерированного фрагмента длиной в 256 временных шагов по затравке из первого такта