

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Образовательная программа бакалавриата «Прикладная математика и информатика»

**СОГЛАСОВАНО**  
Руководитель проекта,  
ООО «Яндекс.Технологии», разработчик

 Симагин Д. А. /  
«23» марта 2019 г.

**УТВЕРЖДАЮ**  
Академический руководитель  
образовательной программы  
«Прикладная математика и  
информатика»  
доцент, канд. физ.-мат. наук

\_\_\_\_\_ А.С. Конушин  
«\_\_» \_\_\_\_\_ 2019 г.

**Нейронные сети сочиняют музыку**


**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.01-01 81 01-1-ЛУ**

Исполнитель:

студент группы 172

 / Зайчулик З.Р. /

«23» марта 2019 г.

**Москва 2019**

## СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ .....	3
1.1 Наименование программы .....	3
1.2 Документы, на основании которых ведется разработка.....	3
2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ .....	4
2.1. Назначение программы .....	4
2.2. Краткая характеристика области применения программы .....	4
3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ .....	5
3.1. Постановка задачи на разработку программы.....	5
3.2. Описание алгоритма функционирования программы с обоснованием выбора схемы алгоритма решения задачи.....	5
3.2.1 Подготовка данных для обучения.....	5
3.2.2 Перевод данных в численное представление .....	6
3.2.3 Обучение RNN модели.....	7
3.2.4 Генерация.....	8
3.2.5 Перевод численного представления в музыкальный формат.....	8
3.3. Описание и обоснование метода организации входных и выходных данных .....	9
3.4. Описание и обоснование метода выбора технических и программных средств .....	9
4.ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ .....	10
4.1 Ориентировочная экономическая эффективность .....	10
4.2 Предполагаемая потребность.....	10
4.3 Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами.....	10
5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	11
ПРИЛОЖЕНИЕ 1 .....	12

**1.**

**ВВЕДЕНИЕ**

**1.1 Наименование программы**

Наименование программы: «Нейронные сети сочиняют музыку».

Краткое наименование программы: «Sonia».

**1.2 Документы, на основании которых ведется разработка**

Разработка ведется на основании приказа Национального исследовательского университета "Высшая школа экономики" № 2.3-02/1501-03 от 15.01.2019

## **2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**

### **2.1. Назначение программы**

Основным назначением программы является разработка алгоритма для генерации музыки на основе методов машинного обучения.

### **2.2. Краткая характеристика области применения программы**

«Нейронные сети сочиняют музыку» – программа, позволяющая создавать музыку на основе выбранного набора музыкальных произведений. Программа может быть использована любыми пользователями в развлекательных целях.

### 3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

#### 3.1. Постановка задачи на разработку программы

Написать программу, позволяющую генерировать музыкальный семпл на основе заданного числа.

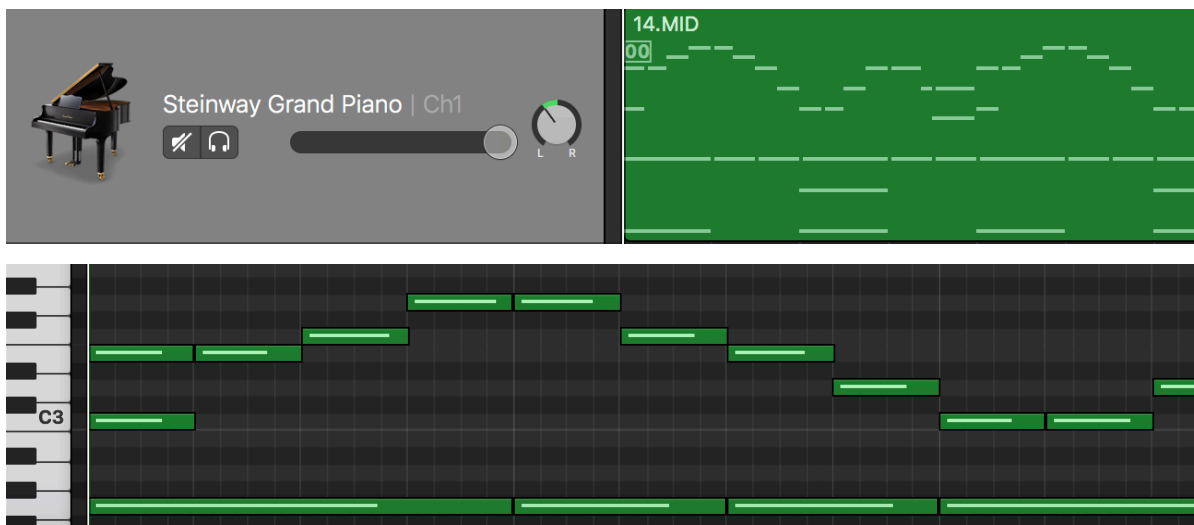
#### 3.2. Описание алгоритма функционирования программы с обоснованием выбора схемы алгоритма решения задачи

Программа генерации музыки состоит из нескольких основных блоков:

- Подготовка данных для обучения
- Перевод данных в численное представление
- Обучение RNN модели
- Генерация
- Перевод численного представления в музыкальный формат

##### 3.2.1 Подготовка данных для обучения

Данные играют важную роль в обучении модели и дальнейшей генерации музыкальных произведений. Для данной программы используем формат данных .midi. Это не оцифрованный звук, а последовательность команд, которые кодируют в цифровой форме такие параметры как: нажатие на определенную клавишу, громкость, тембр, тональность и точное время воспроизведения каждой ноты.



В качестве данные для обучения используем сольные фортепианные произведения Людвига ван Бетховена.

### 3.2.2 Перевод данных в численное представление

Каждое произведение представлено в формате .midi, а значит мы можем получить информацию о произведении через специальные сообщения, которые выглядят следующим образом:

```
note_on channel=0 note=67 velocity=69 time=0.10550203124999999
note_on channel=0 note=64 velocity=63 time=0.0013354687499999999
note_on channel=0 note=58 velocity=66 time=0.0186965625
note_on channel=0 note=46 velocity=55 time=0.0053418749999999999
note_on channel=0 note=61 velocity=44 time=0.0106837499999999999
note_off channel=0 note=64 velocity=67 time=0.0013354687499999999
note_off channel=0 note=58 velocity=66 time=0.0160256249999999998
note_off channel=0 note=46 velocity=63 time=0.0160256249999999998
note_off channel=0 note=67 velocity=61 time=0.0160256249999999998
```

О параметрах:

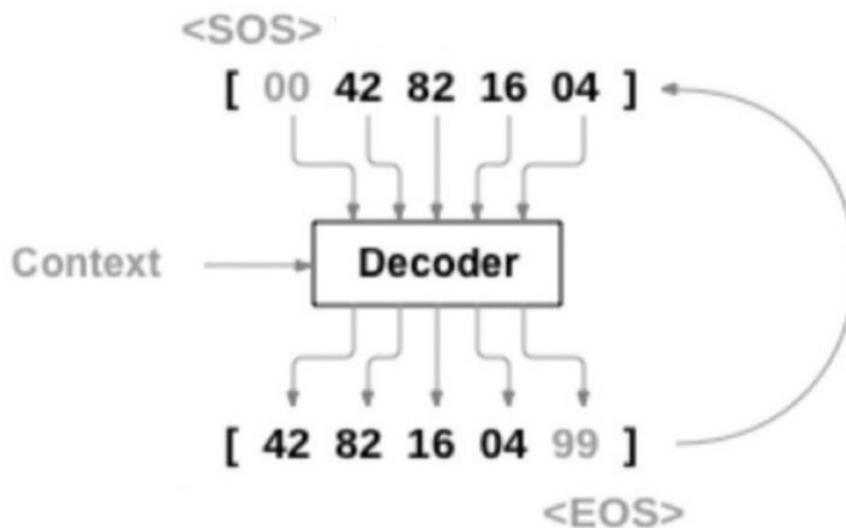
- note\_on - нота включилась
- note\_off - нота выключилась
- note - номер ноты
- velocity - скорость, с которой нота была включена
- time - время, прошедшее с последнего сообщения

Из данных сообщений мы можем получить информацию о том, какая нота в каждый момент времени играет и с какой длительностью. Зафиксируем максимальную длительность сыгранной ноты (`max_dur_note`) и количество различных возможных длительностей каждой ноты (`amount_dur`). Тогда каждому нажатию на клавишу будет соответствовать событие: какая играет нота и с какой продолжительностью. Для кодировки нам удобно использовать технику one-hot encoding. Каждое событие будет представляться в виде вектора нулей, за исключением одной позиции равной 1, соответствующей индексу текущего события.

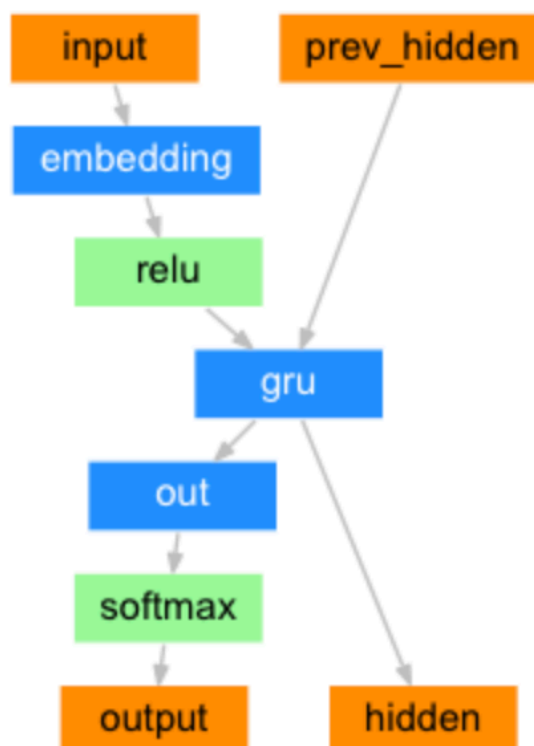
Тогда мы можем перевести информацию из сообщений в вектор индексов. Каждый индекс соответствует событию: номер ноты и ее продолжительность. Также введем уникальный номер (`128 * amount_dur`) для пустого события, то есть никакая нота не включилась в данный момент. События происходят с заданной частотой (`each_dur`).

### 3.2.3 Обучение RNN модели

Decoder - это рекуррентная нейронная сеть, которая принимает на вход вектор(context) и выводит последовательность индексов. На каждом этапе декодирования декодеру присваивается входной токен и скрытое состояние. Первоначальный входной токен - это токен начала строки **<SOS>**, а первое скрытое состояние - вектор.



Сеть выглядит следующим образом:



Для обучения будем использовать «Принудительное обучение» - это концепция использования реальных целевых выходов в качестве каждого следующего входа вместо использования предположения декодера в качестве следующего входа. Использование данного метода заставляет его сходиться быстрее.

Весь тренировочный процесс выглядит следующим образом:

- запустить таймер;
- инициализировать оптимизаторы и критерии;
- создать набор тренировочных векторов;
- запустить пустой массив потерь.

После чего мы запускаем `train` много раз и изредка печатаем прогресс (% примеров, время обучения) и среднюю ошибку.

### 3.2.4 Генерация

Генерация похожа на обучение, но в данном случае результирующие векторы отсутствуют, ввиду чего мы просто возвращаем прогнозы декодера на каждом шаге. Каждый раз, когда происходит предсказание индексов с определенными вероятностями, мы выбираем случайным образом один из  $n$  наиболее вероятных, добавляем его в выходную строку, и, если он предсказывает токен `<EOS>`, останавливаемся.

### 3.2.5 Перевод численного представления в музыкальный формат.

Для того чтобы перевести численное представление в музыкальный формат, нам необходимо создать лист сообщений, соответствующих формату `.midi`. Мы проходимся по массиву индексов и расшифровываем сообщение.

Будем хранить счетчик времени (`time_now`) в данный момент.

1. Если попали на индекс, который отвечает за состояние `<EOS>`, то заканчиваем.
2. Если попали на индекс, который соответствует пустому звуку, то ничего не добавляем. К счетчику времени прибавляем время, установленное ранее как разница между двумя ближайшими сообщениями о включении нот (`time_now += each_dur`).
3. Иначе, определяем ноту как остаток индекса по модулю 128. Затем вычисляем продолжительность ноты. Добавляем два сообщения о включении и выключении соответствующей ноты.

Далее сортируем данный массив по времени и проходимся по данному массиву с конца, чтобы поменять время сообщения на время до предыдущего сообщения. Затем создаем `.midi` файл, добавляем все сообщения.



Также с помощью специальной библиотеки (`midi2audio`) можем преобразовать .midi файл в .wav файл для удобного воспроизведения.

### **3.3. Описание и обоснование метода организации входных и выходных данных**

Программа на вход принимает число (seed), на его основе которого генерируется произведение, так как на вход декодеру подается вектор, каждый элемент которого равен данному числу. Отталкиваясь от этого числа, программа генерирует музыкальный сэмпл. При вводе разных чисел генерируются разные примеры.

Для обучения, как входные данные, используем произведения в формате .midi. Был выбран данный формат, так как он удобен для перевода в численное представление.

В качестве выходных данных, мы получаем произведение в двух форматах:

- .midi (позволяет увидеть, как выглядит музыкальное представление в нотах)
- .wav (позволяет сразу воспроизвести получившееся произведение)

### **3.4. Описание и обоснование метода выбора технических и программных средств**

Для корректной работы программы рекомендуется компьютер, имеющий следующие технические характеристики:

- процессор с частотой 2,3 ГГц или более;
- SSD-накопитель ёмкостью не менее 256 ГБ;
- Машина с UNIX-подобной ОС.

Требования к программным средствам, используемым программой:

- установленный Python 3.6.6
- библиотеки:
  - numpy — version 1.15,
  - torch — version 1.0.0,
  - torchvision — version 0.2.1,
  - pandas — version 0.23.4,
  - matplotlib — version 3.0.2,
  - mido - version 1.2.9,
  - midi2audio - 1.4.2.

#### **4.ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**

##### **4.1 Ориентировочная экономическая эффективность**

В рамках данной работы расчет экономической эффективности не предусмотрен.

##### **4.2 Предполагаемая потребность**

Программу могут использовать совершенно различные пользователи.

##### **4.3 Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами**

Данное приложение:

- 1) распространяется бесплатно;
- 2) не требует вложения денежных средств во время использования;
- 3) имеет неограниченный срок службы.

## 5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A SEQUENCE TO SEQUENCE NETWORK [Электронный ресурс]. Режим доступа: [https://pytorch.org/tutorials/intermediate/seq2seq\\_translation\\_tutorial.html](https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html), свободный.
2. Ashish Vaswani, Attention Is All You Need / Llion Jones, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin [Электронный ресурс]. Режим доступа: <https://arxiv.org/pdf/1706.03762.pdf>, свободный.
3. Cheng-Zhi A.H., Music Transformer: Generating Music with Long-Term Structure / Ian Simon, Monica Dinulescu [Электронный ресурс]. Режим доступа: <https://magenta.tensorflow.org/music-transformer>, свободный.
4. Recurrent neural network [Электронный ресурс]. Режим доступа: [https://en.wikipedia.org/wiki/Recurrent\\_neural\\_network](https://en.wikipedia.org/wiki/Recurrent_neural_network)
5. Soroush Mehri, SampleRNN: An Unconditional End-to-End Neural Audio Generation Model / Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, Yoshua Bengio [Электронный ресурс]. Режим доступа: <https://openreview.net/forum?id=SkxKPDv5xl>, свободный.
6. Sander Dieleman, The challenge of realistic music generation: modelling raw audio at scale / Aäron van den Oord, Karen Simonyan / [Электронный ресурс]. Режим доступа: <https://arxiv.org/pdf/1806.10474.pdf>, свободный.
7. Sageev Oore, This Time with Feeling: Learning Expressive Musical Performance / Ian Simon, Sander Dieleman, Douglas Eck, Karen Simonyan [Электронный ресурс]. Режим доступа: <https://arxiv.org/pdf/1808.03715v1.pdf>, свободный.
8. Aäron van den Oord, WAVENET: A Generating Model ForRaw Audio / Sander Dieleman, Heiga Zen, Heiga Zen, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu [Электронный ресурс]. Режим доступа: <https://arxiv.org/pdf/1609.03499.pdf>, свободный.

**ПРИЛОЖЕНИЕ 1**  
**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

Таблица 1.1 - классы

<b>Класс</b>	<b>Назначение</b>
DecoderRNN	Класс, реализующий архитектуру нейронной сети

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ФУНКЦИЙ**

Таблица 1.2 - функции

<b>Функция</b>	<b>Назначение</b>
download_midi	Скачивает .midi файлы
get_data_from_midi	Получает массив данных из сообщений для одной композиции
resize_data	Меняет размер данных
type_dur	Определяет класс продолжительности ноты
indexes_from_midi	Преобразовывает индексы в .midi
midi_from_indexes	Преобразовывает .midi в индексы
get_list_data	Объединяет в один массив данные из всех композиций
train	Обучает модель
trainIters	Запускает несколько итераций обучения
evaluate	Генерирует сэмпл

[illegible][illegible]