ПРОГРАММНАЯ ГЕНЕРАЦИЯ ЗВУКОВ С ИСПОЛЬЗОВАНИЕМ РЕКУРРЕНТНОЙ НЕЙРОННОЙ СЕТИ

Никитин Н.А., Розалиев В.Л., Орлова Ю.А.
Волгоградский Государственный Технический Университет
+7 (927) 514-70-44, vladimir.rozaliev@gmail.com

Данная работа посвящена разработке и апробации программы генерации музыкальных композиций с использование искусственных нейронных сетей. Работа содержит обоснование выбора рекуррентной нейронной сети, как наиболее подходящей для генерации музыкальных композиций, а также описание используемой нейронной сети. Описан выбор технологии реализации нейронной сети, а также результаты проведения экспериментов.

Ключевые слова: искусственные нейронные сети, рекуррентная нейронная сеть, сети с долгой краткосрочной памятью, Theano, TensorFlow, Lasagne, Keras, MXNet, Python.

Program for sound generation with using the recurrent neural network. Nikitin N.A., Rozaliev V.L., Orlova Yu.A. Volgograd State Technical University

The paper describes the development and approbation of the program for the generation of musical compositions using artificial neural networks. The work contains the rationale for choosing a recurrent neural network as the most suitable method for the generation of musical compositions, as well as a description of the neural network used. It also describes the choice of the neural network implementation technology and the results of the experiments.

Keywords: artificial neural networks, recurrent neural network, long short-term memory network, Theano, TensorFlow, Lasagne, Keras, MXNet, Python.

Введение

Не смотря, на все достижения в понимании творческих процессов, создание музыки не может проходить автоматически. Роль пользователя-композитора очень высока и можно лишь говорить об автоматизации этого процесса. Передаваемая музыкой и картинами эмоциональность сложно распознаваема [1]. Сам процесс создания музыки на данный момент, не поддаётся чёткой формализации, хотя и основывается на чётко определённых музыкальных правилах. Наибольших успехов автоматизация процесса написания и создания музыки достигла сравнительно недавно (в последние десятилетия), однако по большей части связана с изучением и повторением различных музыкальных стилей [2].

Поскольку процесс создания музыки сложно формализуем, то для программного (автоматизированного) создания композиций лучше всего подходят искусственные нейронные сети, так как они позволяют выявить связи, которые не видит человек. В связи с этим, целью данной работы является увеличение гармоничности и мелодичности программной генерации звуков посредством использования нейронных сетей.

Для достижения поставленной цели были выделены следующие задачи:

- обзор типов нейронных сетей и выбор наиболее подходящего типа для генерации музыкальных композиций;
- описание используемой нейронной сети для генерации музыкальных композиций;
- выбор технологии реализации нейронной сети;
- проектирование и разработка программы генерации звуков с использованием нейронных сетей;
- проведение эксперимента по оценке гармоничности и мелодичности выходных музыкальных композиций.

Выбор нейронной сети для генерации музыкальных композиций

Важной особенностью нейронных сетей прямого распространения (feedforward neural networks) является то, что у данной нейросети есть общее ограничение: и входные и выходные данные имеют фиксированный, заранее обозначенный размер, например, картинка 100×100 пикселей или последовательность из 256 бит. Нейросеть с математической точки зрения ведет себя как обычная функция, хоть и очень сложно устроенная: у нее есть заранее обозначенное число аргументов, а также обозначенный формат, в котором она выдает ответ.

Вышеперечисленные особенности не представляет больших трудностей, если речь идет о тех же картинках или заранее определенных последовательностях символов. Но для обработки любой условно бесконечной последовательности, в которой важно не только содержание, но и порядок, в котором следует информация, например, текст или музыка необходимо использовать нейронные сети с обратными связями – рекуррентные нейронные сети (RNN). В рекуррентных нейросетях нейроны обмениваются информацией между собой: например, вдобавок к новому кусочку входящих данных нейрон также получает некоторую информацию о предыдущем состоянии сети. Таким образом в сети реализуется «память», что принципиально меняет характер ее работы и позволяет анализировать любые последовательности данных, в которых важно, в каком порядке идут значения.

Однако большой сложностью сетей RNN является проблема исчезающего (или взрывного) градиента, которая заключается в быстрой потере информации с течением времени. Конечно, это влияет лишь на веса, а не состояния нейронов, но ведь именно в них накапливается информация. Сети с долгой краткосрочной памятью (long short term memory, LSTM) стараются решить вышеупомянутую проблему потери информации, используя фильтры и явно заданную клетку памяти. У каждого нейрона есть клетка памяти и три фильтра: входной, выходной и забывающий. Целью этих фильтров является защита информации. Входной фильтр определяет, сколько информации из предыдущего слоя будет храниться в клетке. Выходной фильтр определяет, сколько информации получат следующие слои. Такие сети способны научиться создавать сложные структуры, например, сочинять тексты в стиле определённого автора или сочинять простую музыку, однако при этом потребляют большое количество ресурсов [3].

Таким образом, для реализации программы автоматизированной генерации музыкальных композиций необходимо использовать именно рекуррентные нейронные сети с долгой краткосрочной памятью – RNN LSTM (долгая краткосрочная память – разновидность архитектуры рекуррентных нейронных сетей). Именно данный вид нейронных сетей используется для генерации музыкальных композиций в программе Magenta – это музыкальный проект с открытым исходным кодом от Google, также RNN LSTM используется в программе сочинения композиций в стиле И.С. Баха – BachBot, а также в DeepJaz – система позволяет генерировать джазовые композиции на основе анализа midi файлов [4].

Описание используемой нейронной сети

Рекуррентная нейронная сеть (RNN) имеет циклические или повторяющиеся соединения, которые позволяют сети хранить информацию по входам. Эти связи можно считать похожими на память. RNN особенно полезны для изучения последовательных данных, таких как музыка.

В TensorFlow повторяющиеся соединения на графе разворачиваются в эквивалентную нейронную сеть прямого распространения (feedforward neural network). Затем эту сеть обучают с использованием техники градиентного спуска, называемой backpropagation through time (BPT).

Есть большое количество способов, с помощью которых RNN может соединяться с собой с циклическими соединениями. Наиболее распространенными являются сети с долгой краткосрочной памятью (long short term memory, LSTM) и управляемые рекуррентные нейроны (gated recurrent units, GRU). В обоих случаях сети имеют мультипликативные нейроны, которые защищают их внутреннюю память от перезаписи, позволяя нейронным сетям обрабатывать более длинные последовательности. В данной работы предполагается использование LSTM [5].

Все рекуррентные нейронные сети имеют форму цепочки повторяющихся модулей нейронной сети. В стандартных RNN этот повторяющийся модуль будет иметь очень простую структуру, например, один слой tanh. LSTM также имеют эту цепочку, но повторяющийся модуль имеет более сложную структуру. Вместо того, чтобы иметь один слой нейронной сети, существует четыре, взаимодействующих между собой особым образом.

Первым шагом в LSTM является решение, какую информацию мы собираемся выбросить из состояния ячейки. Это решение принимается сигмоидным слоем (sigmoid layer). Данный слой смотрит на значение выхода h_{t-1} и входа x_t, рассчитывает значение в диапазон от 0 до 1 для каждого состояния ячейки C_{t-1}. Если слой вернул значений 1, это означает, что данное значение необходимо оставить (запомнить), если 0 – удалить из состояния ячейки. Например, в состоянии ячейки может хранится характеристики текущего такта – если такт ещё не закончен, то необходимо оставить характеристики в памяти, если идёт работа уже с новым тактом, то необходимо запомнить новые параметры.

Следующим шагом является принятие решения о том, какую новую информацию мы собираемся хранить в состоянии ячейки. Для этого, во-первых, сигмоидный слой принимает решение, какие значения мы будем обновлять. Далее, слой tanh создает вектор новых значений кандидата, C_t, которые могут быть добавлены в состояние.

Следующим шагом является обновление старого состояния ячейки C_{t-1} в новом состоянии ячейки Ct. Для этого необходимо умножить старое состояние f_t, таким образом произведём удаление информации из состояния. Затем необходимо сложить полученное значение и $i_t * C_t$, Таким образом получим новые значения кандидатов, масштабируемые значением коэффициента обновления каждого значения состояния.

На последнем шаге необходимо решить, что будет выводить данный слой. Этот вывод будет основан на состоянии ячейки. Сначала пропускаем входное значение через сигмоидный слой, который решает, какие части состояния ячейки необходимо вывести. Затем обрабатываем состояние ячейки с использованием tanh (для сдвига значения между -1 и 1), и умножаем его на выход сигмоидного слоя.

Поведение нейронной сети определяется набором весов и смещений, которые имеют каждый узел. Поэтому, для корректной работы нейронной сети необходимо настроить их на некоторое правильное значение. Во-первых, необходимо определить насколько хороший или плохой какой-либо выход согласно входному значению. Это значение называется стоимостью. Как только стоимость получена, необходимо использовать метод обратного распространения ошибки (backpropagation). По сути, он сводится к вычислению градиента стоимости по отношению к весам (т. е. производной стоимости по каждому весу для каждого узла в каждом слое), а затем необходимо использовать метод оптимизации для корректировки весов для снижения стоимости. В данной работе будем использовать метод градиентного спуска.

Для обучения нейронной сети предполагается подавать на вход вектор, который содержит следующие части [6]:

- название ноты: MIDI обозначение текущей ноты. Используется для представления высоты ноты;
- время начала включения ноты;
- время выключения ноты;
- сила (громкость) воспроизведения ноты (velocity).

Для определения правильного выхода согласно входу, предполагается преобразовать вектор следующим образом: пусть имеется вектор нот {c, d, e, f, g, a, h}, тогда обучающий вектор будет – {{c, d}, {d, e}, {e, f}, {f, g}, {g, a}, {a, h}}. Такой способ обучения нейронной сети используется, например, для прогнозирования временных рядов [7].

Выбор технологии реализации искусственной нейронной сети

Для реализации искусственной нейронной сети был выбран язык программирования Python, поскольку язык является кроссплатформенным, он ориентирован на повышение производительности разработчика и читаемости кода. Кроме того, данный язык ориентирован на анализ данных, а значит содержит большое количество библиотек для глубокого обучения.

Theano — это расширение языка Python, позволяющее эффективно вычислять математические выражения, содержащие многомерные массивы. Поскольку данная библиотека является низкоуровневой, то процесс создания модели и определения ее параметров требует написания объемного и шумного кода. Однако преимуществом Theano является ее гибкость, а также наличие возможности реализации и использования собственных компонент [8].

TensorFlow — это библиотека с открытым исходным кодом для численного расчета с использованием потоковых графов. Данная библиотека, также, как и Theano является низкоуровневой, а значит процесс разработки сложный. Однако благодаря низкому уровню разработки нейронных сетей можно получить более гибкую модель. Также преимуществом данной библиотеки является большое сообщество и хорошая документация [9].

Lasagne — легковесная обёртка для библиотеки Theano. Программирование с использованием Lasagne достаточно низкоуровневое – необходимо объявить каждый уровень нейронной сети посредством использования модульных строительных блоков над Theano. Lasagne выступает как компромисс между гибкостью Theano и простотой Keras [10].

Keras — это высокоуровневый API для разработки нейронных сетей, написанный на Python и способный работать поверх TensorFlow, CNTK или Theano. Библиотека была разработана с упором на возможность быстрого экспериментирования. Минусом данной библиотеки является небольшая гибкость [11].

MXNet — это система глубокого обучения с открытым исходным кодом, используемая для обучения и развертывания глубоких нейронных сетей. Поскольку MXNet является библиотекой высокого уровня разработка нейронных сетей с использованием MXNet проще и быстрее, чем с использованием Theano или TensorFlow, однако уступает библиотеке Keras за счёт большого числа поддерживаемых языков и больших возможностей для масштабирования, что делает программный код более громоздким [12].

Для проведения сравнения были выделены следующие критерии: гибкость, масштабируемость, поддержка параллельных вычислений, поддержка вычисления на GPU, скорость разработки. Все рассмотренные библиотеки были оценены по представленным выше критериям по пятибалльной шкале, где 0 – минимальное значение критерия, 1 – максимальное. Результаты сравнения библиотек представлены в таблице 1.

Таблица 1 – Сравнение библиотек для глубокого обучения

	Theano	TensorFlow	Lasagne	Keras	MXNet
Гибкость	5	4	3.5	2	3
Масштабируемость	4	5	4	5	5
Поддержка параллельных вычислений	5	4	5	5	5
Поддержка вычисления на GPU	4	5	4	5	5

Таким образом, можно сделать вывод о том, что для разработки рекуррентной нейронной сети для генерации музыкальных композиций следует использовать библиотеку Keras, поскольку данная библиотека позволяет работать поверх Theano и TensorFlow, используя их преимущества, при этом процесс разработки нейронных сетей с использование данной библиотеки простой и быстрый, что позволяет легко и быстро создавать прототипы для быстрого экспериментирования.

Проведение эксперимента

Программа генерации музыкальных композиций с использованием нейронных сетей была обучена на 29 композициях Людвига ван Бетховена. После обучения были сгенерированы музыкальные композиции,

наиболее удачные из которых были отправлены экспертам на анализ. Эксперты оценивали две композиции по следующим критериям: мелодичность композиции; приятность для восприятия (в том числе наличие диссонансов); цельность композиции; реалистичность/искусственность композиции, соответствие стилю (классический)

Все критерии оценивались по пятибалльной шкале, где 1 – минимальное значение критерия, 5 – максимальное.

Экспертные оценки представлены в таблице 2.

Таблица 2 – Экспертные оценки композиций

	Композиция №1	Композиция №2
Мелодичность композиции	3.5	3
Приятность для восприятия	4	3
Цельность композиции	5	5
Реалистичность/искусственность композиции	4	3
Соответствие стилю (классический)	5	5

Таким образом, видно, что в результате работы программы были получены композиции, которые полностью соответствуют классическому произведению, однако имеют недостатки в мелодичности композиции. Это связано с тем, что для обучения подавалось небольшое количество произведений только одного композитора.

Заключение

В ходе анализа различных типов и архитектур ИНС был сделан вывод о том, что наиболее подходящей сетью для обработки музыкальной информации являются рекуррентные нейронные сети (RNN), а именно сети с долгой краткосрочной памятью (long short term memory, LSTM).

Также, была детально описана используемая искусственная нейронная сеть, а также способ её обучения. Для обучения нейронной сети предполагается подавать на вход вектор, который содержит следующие части: название ноты – MIDI обозначение текущей ноты. Используется для представления высоты ноты, время начала включения ноты, время выключения ноты, сила (громкость) воспроизведения ноты (velocity).

Для реализации программы были проанализированы библиотеки для реализации нейронной сети на языке программирования Python. В ходе анализа было выявлено, что для разработки рекуррентной нейронной сети для генерации музыкальных композиций следует использовать библиотеку Keras, поскольку данная библиотека позволяет работать поверх Theano и TensorFlow, используя их преимущества, при этом процесс разработки нейронных сетей с использование данной библиотеки простой и быстрый, что позволяет легко и быстро создавать прототипы для быстрого экспериментирования.

В результате работы программы были получены композиции, которые полностью соответствуют классическому произведению, однако имеют недостатки в мелодичности композиции. Это связано с тем, что для обучения подавалось небольшое количество произведений только одного композитора.

Работа частично поддержана Российским фондом фундаментальных исследований (проекты 16-47-340320, 17-07-01601).

Литература

1. V. Rozaliev Methods and Models for Identifying Human Emotions by Recognition Gestures and Motion / V. Rozaliev, A. Zaboleeva-Zotova // The 2013 2nd International Symposium on Computer, Communication, Control and Automation 3CA 2013, December 1-2, 2013, Singapore : Papers. – [Amsterdam – Beijing – Paris] : Atlantis Press, 2013. – P. 67-71.

2. D. Cope, Computer Models of Musical Creativity, MIT Press, Cambridge, Mass., 2005.

3. Doornbusch, P. Gerhard Nierhaus: Algorithmic Composition: Paradigms of Automated Music Generation / P. Doornbusch // Computer Music Journal. - Volume: 34, Issue: 3. – 2014.

4. Brinkkemper, F. Analyzing Six Deep Learning Tools for Music Generation [Электронный ресурс]. – 2015. - Режим доступа: http://www.asimovinstitute.org/analyzing-deep-learning-tools-music/ (Дата обращения: 03.07.2017).

5. Mazurowski, L. Computer models for algorithmic music composition / L. Mazurowski // Proceedings of the Federated Conference on Computer Science and Information Systems. – Szczecin, Poland, 2012. – pp. 733–737

6. Transactions on Engineering Technologies: Special Issue of the World Congress on Engineering and Computer Science / Haeng Kon Kim, Sio-Iong Ao, A. Mahyar. – Springer Publishing Company, New York, 2013. – pp. 796

7. Fernández, J. D., Vico, F. AI Methods in Algorithmic Composition: A Comprehensive Survey / J. D. Fernández, F. Vico // Journal of Artificial Intelligence Research. – № 48. – Málaga, Spain, 2013. – pp. 513-582

8. James Bergstra, Olivier Breuleux, Fr´ed´eric Bastien, Pascal Lamblin, Razvan Pascanu, Guil-laume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy), June 2010.

9. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems [Электронный ресурс]. – 2015. – Режим доступа: https://www.tensorflow.org/ (Дата обращения: 11.07.2017).

10. Lasagne - lightweight library to build and train neural networks in Theano [Электронный ресурс]. – 2017. – Режим доступа: http://lasagne.readthedocs.org/ (Дата обращения: 12.07.2017).

11. Keras: The Python Deep Learning library [Электронный ресурс]. – 2017. – Режим доступа: https://keras.io/ (Дата обращения: 12.07.2017).

12. MXNet: A Flexible and Efficient Library for Deep Learning [Электронный ресурс]. – 2017. – Режим доступа: https:// http://mxnet.io// (Дата обращения: 12.07.2017).