УДК 004.942 *DOI 10.55421/3034-4689_2025_28_10_86*

С. К. Долгих, Л. Ю. Кошкина

СОЗДАНИЕ ПРОГРАММНОГО ПРИЛОЖЕНИЯ ДЛЯ ГЕНЕРАЦИИ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОСЕТЕВОЙ МОДЕЛИ

Ключевые слова: нейросетевая модель LSTM, генерация музыкальных композиций, MIDI-композиции, токенизация музыкальных данных

Представленная работа посвящена описанию создания программного приложения для автоматической генерации музыкальных MIDI-композиций с использованием нейросетевой модели LSTM. Проанализированы различные архитектуры нейросетей: LSTM, GAN, трансформеры и диффузионные модели. Выбрана модель LSTM как наиболее устойчивая и адаптивная для генерации последовательностей, обученная на музыкальных фрагментах, с последующей генерацией мелодий и сохранением результатов в формате MIDI. Токенизация музыкальных данных (в формате MIDI) была реализована через представление pianoroll, что позволило преобразовать музыку в последовательности, пригодной для подачи в нейросеть. Реализована архитектура нейросети и пользовательский интерфейс. Разработанный пользовательский интерфейс обеспечивает ввод параметров генерации и скачивание готового файла. Также реализована система серверной логики на фреймворке Flask. Проведено тестирование, подтверждающее работоспособность системы и качество получаемых музыкальных последовательностей. В ходе разработки приложения были также выявлены и устранены проблемы, связанные с производительностью модели и качеством генерации музыки. Результатом является рабочий прототип, обеспечивающий автоматическую генерацию музыки. Основные характеристики продукта: генерация MIDI-композиций с настройкой параметров (темп, длина, творческая вариативность); простой и интуитивно понятный интерфейс; локальная обработка данных, не требующая подключения к интернету; бесплатный прототип на этапе разработки, с перспективой перехода на freemium-модель.

S. K. Dolgikh, L. Yu. Koshkina

CREATING A SOFTWARE APPLICATION FOR GENERATING MUSICAL COMPOSITIONS USING A NEURAL NETWORK MODEL

Keywords: neural network model LSTM, generation of musical compositions, MIDI compositions, tokenization of musical data.

The presented work is devoted to the description of the creation of a software application for the automatic generation of MIDI musical compositions using the LSTM neural network model. Various neural network architectures are analyzed: LSTM, GAN, transformers and diffusion models. The LSTM model was chosen as the most stable and adaptive for generating sequences, trained on musical fragments, followed by generating melodies and saving the results in MIDI format. Tokenization of music data (in MIDI format) was implemented through the pianoroll representation, which made it possible to transform music into sequences suitable for submission to a neural network. Implemented neural network architecture and user interface. A user interface has been developed that allows users to set generation parameters and download the finished file, as well as a server logic system on Flask. Testing was carried out to confirm the operability of the system and the quality of the resulting musical sequences. During the development of the application, problems related to the performance of the model and the quality of music generation were also identified and eliminated. The result is a working prototype that automatically generates music. Main product characteristics: generation of MIDI compositions with parameter settings (tempo, length, creative variability); simple and intuitive interface local data processing that does not require an Internet connection; free prototype at the development stage, with the prospect of switching to a freemium model.

Создание музыкальных композиций – процесс, требующий творческого подхода, времени, глубоких знаний в области музыки, а также доступа к профессиональным инструментам. Музыкальный контент востребован в самых разных сферах: от создания видеороликов и разработки игр до образовательных проектов и рекламных кампаний.

С развитием технологий искусственного интеллекта, в частности нейросетевых моделей, появляются возможности для автоматизации творческих процессов. Нейросети уже успешно применяются в таких областях, как обработка изображений [1], синтез речи и генерация текста, и постепенно находят применение в музыкальной индустрии [2].

Тем не менее, существующие решения для генерации музыки (чаще всего) либо требуют от пользователя глубоких технических знаний, либо ограничены в возможностях настройки результата. Потребность в программных инструментах, которые позволили бы как профессионалам, так и любителям создавать качественные музыкальные композиции с минимальными затратами времени и ресурсов, растёт [3].

Генерация музыки при помощи нейросетевых моделей – направление в области искусственного интеллекта, которое сейчас активно развивается. Но традиционный процесс написания музыки требует от композиторов значительных усилий, профессиональных навыков и доступа к специализированному оборудованию. А использование готовых композиций для коммерческих целей часто связано с необходимостью покупки дорогостоящих лицензий.

Современные модели глубокого обучения позволяют автоматизировать создание музыки [4]. Об этом свидетельствуют научные исследования и экспериментальные разработки [5-8].

Цель представленной работы – создание программного приложения для генерации музыкальных композиций с использованием нейросетевой модели.

Генерация музыки является задачей синтеза сложных временных последовательностей. Музыка состоит из множества взаимосвязанных компонентов: высоты тона, длительности, ритма, темпа, гармонии, тембра и динамики. Например, гармония может сохраняться на протяжении нескольких тактов, в то время как ритмические элементы могут повторяться или модифицироваться в рамках музыкальной фразы.

Наиболее популярные архитектуры для задачи музыкальной генерации – это архитектура трансформера (Transformer) [6], генеративные состязательные сети (GAN) [7], диффузные модели и LSTM (табл. 1).

Таблица 1 – Сравнение нейросетевых моделей для генерации музыки

Table 1 – Comparison of neural network models for music generation

Архитектура	Преимущества	Недостатки
Transformer	Отлично справляются с текстами; параллельная обработка	Сложности при пошаговой генерации; требуют маскирования; высокие ресурсы
GAN	Генерируют стилистически реалистичные последовательности	Нестабильность обучения; склонность к переобучению
Диффузионные	Высокая креативность; способны восстанавливать данные из шума	Сложности с временной структурой; большие вычислительные ресурсы и время обучения
LSTM	Устойчивы к исчезающим градиентам; учитывают долгосрочные зависимости	Могут уступать по генеративной выразительности без постобработки

Музыкальная генерация требует от модели как «локальной» точности (верная высота и длительность нот), так и «глобальной» осмысленности (структура, развитие, кульминации).

Выбор модели LSTM (*long short-term memory*) обусловлен тем, что лучше сохраняется временная согласованность, данные обрабатываются последовательно, модель проще реализуется и настраивается, использует меньше памяти, требует меньше ресурсов, стабильнее обучается [8].

Архитектура рекуррентной нейронной сети с долгой краткосрочной памятью LSTM эффективна для генерации музыкальных композиций благодаря способности удерживать контекст на протяжении длинных последовательностей.

В скрытых слоях нейросети происходят нелинейные преобразования с помощью функций активации: сигмоиды и гиперболического тангенса. Они определяют, каким образом данные будут преобразованы (рис. 1).

В задачах генерации обычно используется символическое представление музыки, например, MIDI, поскольку оно компактно, структурировано и поддаётся контролю [9].

Входными данными являются MIDI-последовательности, взятые из Tegridy MIDI Dataset. В сравнении с аудио-ориентированными датасетами, такими как FMA или AudioSet, Tegridy фокусируется на MIDI, обеспечивая лёгкость в обработке и генерации музыки без необходимости работы с тяжёлыми аудиофайлами.

Выходом является новая последовательность в формате MIDI.

Функциональная модель для описания пространственно-временных и причинно-следственных зависимостей между объектами внутри системы [10] представлена в виде диаграммы последовательностей (рис. 2).

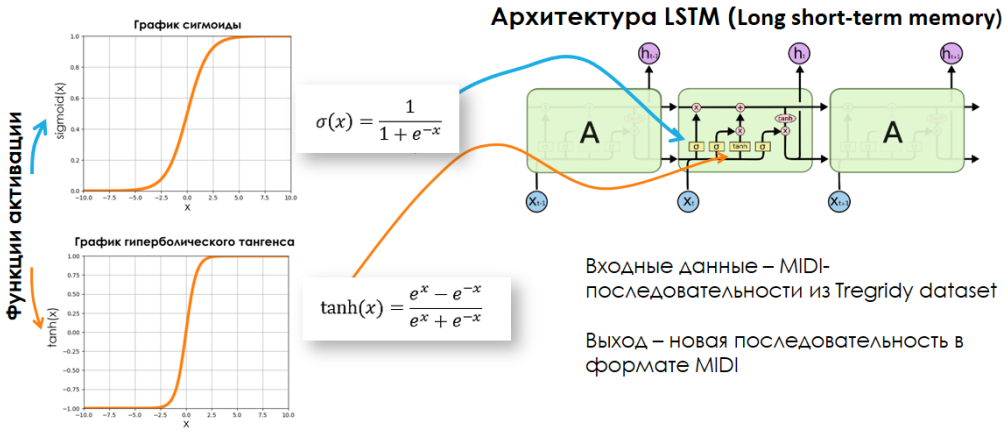

Рис. 1 – Архитектура нейронной сети для музыкальной генерации

Fig. 1 – Neural network architecture for music generation

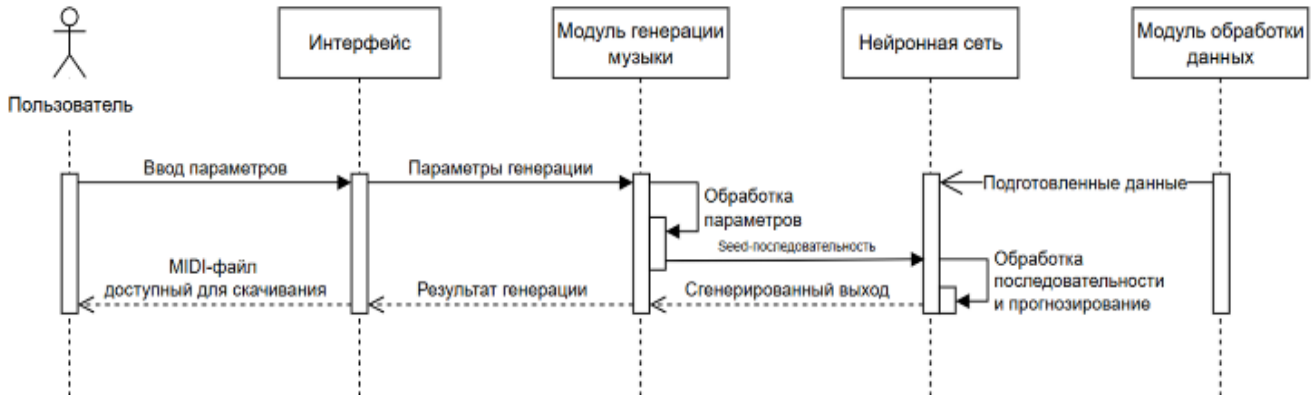

Рис. 2 – Функциональная модель системы

Fig. 2 – Functional model of the system

Пользователь вводит параметры генерации в интерфейс, который, в свою очередь, передаёт их в модуль генерации музыки. Этот модуль обрабатывает параметры и формирует начальную *seed*-последовательность, от которой будет отталкиваться генерация. Далее последовательность попадает в полносвязную нейронную сеть. Она обучена благодаря датасету, прошедшему токенизацию через модуль обработки данных. Нейронная сеть прогнозирует ноты, и подаёт сгенерированный выход через модули в обратном порядке, в результате чего пользователь получает MIDI-файл доступный для скачивания.

Для реализации приложения выбраны следующие инструменты и технологии:

- ✓ язык программирования Python,
- ✓ библиотеки PyTorch, PrettyMIDI и MIDIUtil,
- ✓ среда разработки IDE Visual Studio Code,
- ✓ фреймворк Flask,
- ✓ язык разметки HTML и его базовые стили,
- ✓ секвенсор (аудиостанция) FL Studio.

Этапы разработки приложения включают: настройку окружения, разработку нейросетевой модели, токенизацию MIDI-событий, обучение нейросети, создание модуля генерации, проектирование интерфейса пользователя.

Непосредственное программирование начинается с настройки окружения в файле «mian_generator.py». Процесс включает импорт библиотек и создание базовых директорий.

Центральным компонентом приложения является рекуррентная нейросетевая модель на базе LSTM. Класс «MusicGenerator», который наследуется от «torch.nn.Module», определяет архитектуру модели, включая слой рекуррентной памяти.

Чтобы нейросетевая модель могла обрабатывать музыкальные данные, необходимо предварительно преобразовать их в тензоры фиксированной размерности, т.е. токенизировать [11]. Эту задачу берёт на себя модуль обработки данных.

Обучение нейросетевой модели позволяет системе воспроизводить закономерности из музыкальных последовательностей и создавать на их основе композиции. Процесс происходит внутри функции «train_model».

После запуска программы осуществляется обучение модели на основе Tegridy dataset, состоящего из 880 MIDI-последовательностей, которые делятся в процентном соотношении 80:20 (на тренировочную и тестовую выборку), что необходимо для оценки обобщающей способности модели на данных, которые не попадают в нейронную сеть во время обучения.

Сам процесс обучения состоит из 30 эпох, на каждой рассчитывается средняя ошибка (рис. 3).

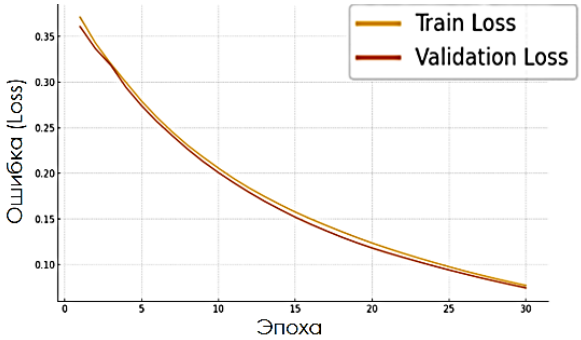

Рис. 3 – График сходимости модели на этапах обучения

Fig. 3 – Model convergence graph during training stages

По графику сходимости видно, как ошибка уменьшается с увеличением эпох, а по виду графика можно сказать, что модель не склонна к переобучению, так как между кривыми нет значительного расхождения.

Далее появляется сообщение о запуске интерфейса, открытие которого осуществляется по ссылке в том же терминале.

Пользовательский интерфейс представлен на рис. 4. Пользователь может задавать длину нот, темп композиции, а также творческую вариативность.

После выставления параметров, нажатие на кнопку «Сгенерировать музыку», запускается маршрут «@app.route("/generate", methods=["POST"])». В результате на текущую машину скачивается готовый MIDI-файл «generated.mid».

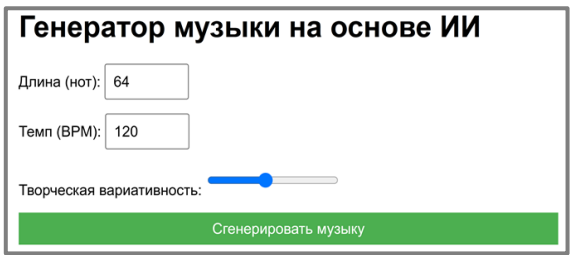

Рис. 4 – Интерфейс приложения

Fig. 4 – Application interface

Реализована система серверной логики на Flask. Данный фреймворк прост в интеграции серверных приложений с моделями машинного обучения, при всём минимализме обладает гибкостью и расширяемостью, что позволяет разработчикам сохранять контроль над приложением, выбирая только те инструменты и библиотеки, которые им необходимы.

В ходе разработки приложения были выявлены и устранены проблемы, связанные с производительностью модели и качеством генерации музыки [12, 13].

Для оптимизации и отладки произведено следующее.

1. *Оптимизация обучения модели.*

✓ Для ускорения обучения был уменьшен размер скрытого слоя LSTM с 512 до 256 нейронов, что способствовало сокращению выполнения процесса на 30% по времени без значительной потери качества генерации.

✓ Использование Adam (Adaptive Moment Estimation – популярный и мощный алгоритм оптимизации, используемый в машинном обучении (ML) и глубоком обучении (DL)) помогло модели обрести быструю сходимость.

✓ Модель, используя методы оптимизации функций активации tanh и ReLU, делает градиенты более устойчивыми, что содействует стабильности обучения.

2. *Улучшение качества генерации.*

✓ Введён параметр температурной корректировки (temperature), позволяющий регулировать уровень случайности при генерации, что позволяет найти баланс между креативностью и предсказуемостью результата.

✓ Удаление слишком коротких нот (менее 1/16 такта) сгладило ритмические паттерны, результаты генерации стали более музыкальными.

3. *Отладка и обработка ошибок.*

✓ Реализована проверка наличия и корректности входных MIDI-файлов. При обнаружении ошибок, процесс генерации останавливается, в терминале выводится подробное описание неисправности.

✓ Улучшена обработка исключений (метода try-except) при работе с моделью и генерации MIDI-файлов для того, чтобы избежать аварийного завершения программы.

4. *Оптимизация использования памяти.*

✓ Уменьшение размера обучающих последовательностей «seq_lenght» с 64 до 32 шагов снизило нагрузку на оперативную память, так как сокращенная длина последовательности означает меньшее количество тензоров без ущерба качеству генерации.

✓ Для экономии ресурсов каждый шаг начинается с обнуления градиентов «optimizer.zero_grad()».

На момент завершения разработки приложение представляет собой прототип, готовый к тестированию и после дальнейшей доработки может реализован в рыночных условиях.

Основные характеристики продукта: генерация MIDI-композиций с настройкой параметров (темп, длина, творческая вариативность); простой и интуитивно понятный интерфейс [14]; локальная обработка данных, не требующая подключения к интернету; бесплатный прототип на этапе разработки, с перспективой перехода на freemium-модель.

Созданный продукт позволит пользователям, не обладающим глубокими знаниями в музыке и программировании, создавать музыкальные MIDI-композиции, которые могут востребованы в рекламе, кинематографе, видеоиграх, образовательных платформах (например, в подготовке специалистов для химической отрасли), социальных сетях, для личных целей.

Литература

1. N.M. Hoang, K. Gong, Ch. Guo, M. Bi Mi, *Proceedings of the AAAI Conference on Artificial Intelligence*, **38**, 3, 2157-2165 (2024).
2. А.М. Шестерина, *Вестник Воронежского государственного университета,* **1**, 52, 277-282 (2023).
3. А.А. Медведев, *Право и управление*, **9,** 342-351 (2024).
4. Н. А. Никитин, Ю. А. Орлова, В. Л. Розалиев, *Моделирование, оптимизация и информационные технологии*, **10**, 2(37), (2022).
5. J. Nurmurodov, M. Abduganiev, A. Tillaboyev, *Проблемы вычислительной и прикладной математики*, **6,** 1(54), 56-67, (2023).
6. Utkin, V. V. Shkuropatsky, A. N. Pronikov, E. S. Rakov, *Computing, Telecommunications and Control*, **17**, 1, 54-64, (2024).
7. Свид. о регистрации программы для ЭВМ RU 2023683865 (2023).
8. S. Mangal, R. Modak, P. Joshi, *International Journal of Innovative Technology and Exploring Engineering*, **9**, 4, 1234–1240, (2020).
9. Пат. на изобретение RU 2364956 C1 (2009).
10. S. Al-Fedaghi, *International Journal of Advanced Computer Science and Applications*, **12**, 5 (2021).
11. AI, практический курс. Глубокое обучение для генерации музыки [Электронный ресурс] : Хабр / Intel AI Academy, 2020. – URL: https://habr.com/ru/companies/intel/articles/423727 (дата обращения 10.06.2025).
12. Т. М. Шамсутдинова // *Открытое образование*, **26**, 6, 4-10 (2022).
13. Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, Yi-Hsuan Yang. MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation [Электронный ресурс] : Githib. – URL: https://hermandong.com/musegan. (дата обращения: 10.06.2025).
14. Л.Ю. Кошкина, К.В. Кошкина, Л.Т. Воронина, *Вестник Технологического университета*, **26**, 12, 175-179, (2023).

References

1. N.M. Hoang, K. Gong, Ch. Guo, M. Bi Mi, *Proceedings of the AAAI Conference on Artificial Intelligence*, **38**, 3, 2157-2165 (2024).
2. A.M. Shesterina, *Bulletin of Voronezh State University*, **1**, 52, 277-282 (2023).

3. A.A. Medvedev, *Law and Management*, **9,** 342-351 (2024).
4. N.A. Nikitin, Yu.A. Orlova, V.L. Rosaliev, *Modeling, Optimization, and Information Technologies*, **10**, 2(37), (2022).
5. J. Nurmurodov, M. Abduganiev, A. Tillaboyev, *Problems of Computational and Applied Mathematics*, **6,** 1(54), 56-67, (2023).
6. Utkin, V. V. Shkuropatsky, A. N. Pronikov, E. S. Rakov, *Computing, Telecommunications and Control,* **17**, 1, 54-64, (2024).
7. Certificate of registration of a computer program RU 2023683865 (2023).
8. S. Mangal, R. Modak, P. Joshi, *International Journal of Innovative Technology and Exploring Engineering*, **9**, 4, 1234–1240, (2020).
9. Patent for invention RU 2364956 C1 (2009).
10. S. Al-Fedaghi, International Journal of Advanced Computer Science and Applications, 12, 5 (2021).
11. AI, practical course. Deep learning for music generation [Electronic resource]: Habr / Intel AI Academy, 2020. – URL: https://habr.com/ru/companies/intel/articles/423727 (accessed 10.06.2025).
12. T. M. Shamsutdinova // *Open Education*, **26**, 6, 4-10 (2022).
13. Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, Yi-Hsuan Yang. MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation [Electronic resource]: Githib. – URL: https://hermandong.com/musegan. (accessed 10.06.2025).
14. L.Yu. Koshkina, K.V. Koshkina, L.T. Voronina, *Herald of Technological University*, **26**, 12, 175-179, (2023).

© С. К. Долгих – бакалавр по направлению 09.03.03 «Прикладная информатика», Казанский национальный исследовательский технологический университет (КНИТУ), Казань, Россия; **Л. Ю. Кошкина** – канд. техн. наук, доцент кафедры Химической кибернетики, КНИТУ, KoshkinaLYu@corp.knrtu.ru.

© S. K. Dolgikh – Bachelor-student in Applied Informatics 09.03.03, Kazan National Research Technological University (KNRTU), Kazan, Russia; **L. Yu. Koshkina** – PhD (Technical Sci.), Associate Professor, Department of Chemical Cybernetics, KNRTU, KoshkinaLYu@corp.knrtu.ru.

Дата поступления рукописи в редакцию – 15.09.25.
Дата принятия рукописи в печать – 23.09.25.