ТЕХНИЧЕСКИЕ НАУКИ

УДК 004.8

Проектирование программы для озвучивания текста на основе генеративно-состязательной нейронной сети

А.С. Серенко, Е.А. Лукьянов
Донской государственный технический университет, г. Ростов-на-Дону, Российская Федерация

Аннотация
В работе рассмотрено применение нейронных сетей для синтеза речи конкретного человека. Исследуется вопрос: возможно ли озвучить введённый текст голосом заданной выборки и при каких условиях это достигается. Предполагается, что генеративно-состязательная сеть, обученная на профайле голоса, обеспечит приемлемую точность. Разработана программа с генератором и двойным дискриминатором: генерация производится из векторного представления фонем, дискриминатор включает сверточный классификатор и распознаватель речи. Проведено обучение и тестирование на наборах в 20, 140 и 400 файлов. Сделан вывод о работоспособности подхода и необходимости увеличения выборки и оптимизации архитектуры. Результаты актуальны для приложений text-to-speech и требуют полного ознакомления для воспроизведения и улучшения решений.

Ключевые слова: генеративно-соревновательная нейронная сеть, программа на основе GAN, озвучивание текста

Для цитирования. Серенко А.С., Лукьянов Е.А. Проектирование программы для озвучивания текста на основе генеративно-состязательной нейронной сети. *Молодой исследователь Дона*. 2026;11(1):31–36.

Designing a Text-to-Speech Software Based on a Generative Adversarial Neural Network

Aleksey S. Serenko, Evgeny A. Lukyanov
Don State Technical University, Rostov-on-Don, Russian Federation

Abstract
The article studies the use of neural networks for synthesizing speech of a certain person. The possibility of reproducing the input text using a voice from a given sample was studied as well as requirements to achieve this. It was assumed that a generative adversarial network (GAN) trained on a voice profile could provide the acceptable accuracy. A software based on a generator and dual discriminator was developed, which generates phoneme embeddings, and has a discriminator consisting of a convolutional classifier and a speech recognizer. Training and testing were conducted on datasets of 20, 140, and 400 files. The conclusions about feasibility of the approach and the need to increase the sample size and optimize the architecture were made. The results are relevant for text-to-speech applications and require in-depth study for replication and improvement of the solutions.

Keywords: generative adversarial neural network, GAN-based software, text-to-speech process

For Citation. Serenko AS, Lukyanov EA. Designing a Text-to-Speech Software Based on a Generative Adversarial Neural Network. *Young Researcher of Don*. 2026;11(1):31–36.

Введение. В современном мире широкое распространение получило одно из направлений искусственного интеллекта (далее — ИИ) — нейронные сети. Они решают задачи, где требуется выявление закономерностей в данных, которые не всегда очевидны для других методов ИИ. Одна из таких задач — установление связей между аудиоданными с общим признаком и генерация новых аудиофайлов на их основе. Практическое применение этой задачи — создание озвучивания текста различными голосами. Для использования звуковых фильтров требуется имеющийся образец озвучивания, но при его отсутствии прибегают к генерации из текста — операции, с которой лучше всего справляются нейронные сети. Цель работы — разработать программу на основе нейронной сети, способную озвучивать введённый текст голосом конкретного человека.

Основная часть. На этапе планирования разработки ПО необходимо сформулировать задачи, определяющие функционал программы. Опишем входные и выходные данные: на вход система должна принимать текст для озвучивания и либо аудиофайлы, содержащие требуемый голос, либо профиль этого голоса (сохранённые параметры). В результате программа должна выдавать озвученный текст в виде аудиофайла и профиль голоса, если он не был предоставлен на входе. На рис. 1 разрабатываемая программа представлена в виде «чёрного ящика», входы и выходы которого соответствуют приведённому описанию.

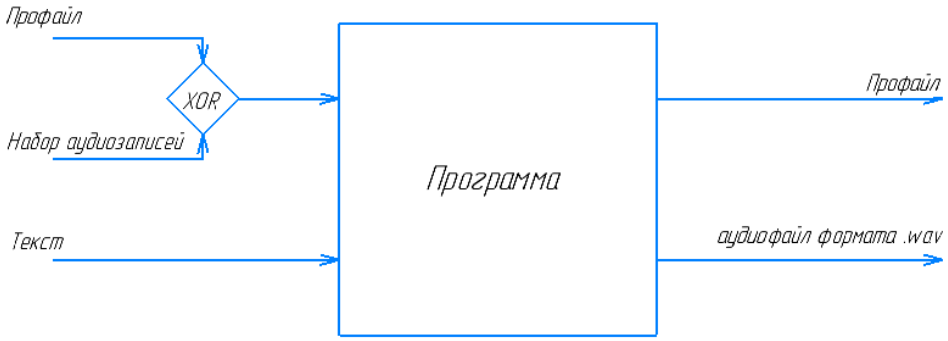

Рис. 1. Разрабатываемая программа, представленная как черный ящик

Как было упомянуто ранее, необходимо реализовать генерацию звуковой дорожки из текста. Это может быть осуществлено с помощью генеративно-состязательной нейронной сети. Наиболее подходящим типом такой сети является Hi-Fi GAN [1]. Работа с этой сетью требует предварительного формирования из текста мэл-спектрограммы [2], после чего нейронная сеть способна сгенерировать звуковую дорожку.

Процесс обучения такой нейронной сети заключается в определении сходства с оригинальной записью сразу после генерации аудиосигнала. Для этого исходная голосовая запись и синтезированный звук пропускаются через двойной дискриминатор, который выделяет паттерны в звуковых дорожках и спектрограммах для сопоставления.

В отличие от Hi-Fi GAN, проектируемая нейронная сеть будет генерировать звуковую дорожку не на основе мэл-спектрограммы, а на основе вектора, представляющего разложение слов введённого текста на фонемы, закодированные в тензор с помощью словаря «фонема-тензор» (рис. 2). Блок дискриминатора состоит из двух частей — бинарного классификатора для определения соответствия специфических черт (далее — фитч) эталонного голоса (записи из выборки) и синтезированного, и распознавателя речи «speech-to-text», который преобразует сгенерированную дорожку обратно в текст и сравнивает её с исходным текстом.

№	Phonem	0	1	2	3	4	5
	0	1	2	3	4	5	6
0	а	1.0	0.0	0.0	0.0	0.0	0.0
1	и	0.0	1.0	0.0	0.0	0.0	0.0
2	о	0.0	0.0	1.0	0.0	0.0	0.0
3	у	0.0	0.0	0.0	1.0	0.0	0.0
4	ы	0.0	0.0	0.0	0.0	1.0	0.0
5	э	0.0	0.0	0.0	0.0	0.0	1.0
6	б	0.0	0.0	0.0	0.0	0.0	0.0

Рис. 2. Фрагмент словаря «фонема – тензор»

Классификатор, входящий в состав дискриминатора, представляет собой сверточную нейронную сеть; её структура показана на рис. 3. Входной слой — одномерный сверточный: он преобразует моноканальный звук длительностью 2 секунды в тензор размером 16 на 88 196 значений, что соответствует записи при частоте дискретизации 44,1 КГц, представленной в 16 каналах. Скрытые слои образуют «воронку» последовательных блоков — линейный слой, за которым следует функция активации ReLU, что приводит к поэтапному уменьшению размерности тензора. По выходу из «воронки» полученный тензор размером 1 на 1 проходит через функцию активации Sigmoid, которая даёт численную вероятность совпадения фич.

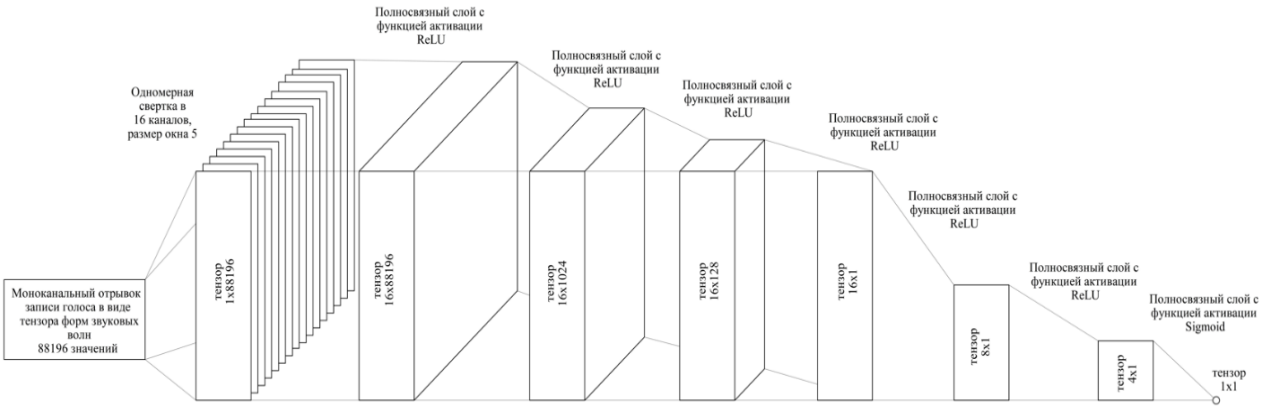

Рис. 3. Схема конфигурации слоев классификатора

Распознаватель голоса представлен готовой моделью на базе CMU «Shpinx» [3] от команды «voxforge_ru_sphinx» [4], находящейся в открытом доступе. Ее необходимо единоразово дообучить, чтобы не обучать вместе с классификатором.

Входной слой генератора принимает слово-тензор размером L на N значений, где N — число фонем в словаре (глубина слова), которое не изменяется в процессе обучения и дообучения модели. Значение L рассчитано по формуле 1, и конкретно для упомянутой ранее структуры дискриминатора представляет собой интервал в две секунды, деленный на среднюю длительность звука фонемы — 0,05 секунд.

$$L = \frac{t_в}{t_ф}, \quad (1)$$

где L — длина слова-тензора; $t_в$ — длительность генерируемых отрезков звуковой дорожки; $t_ф$ — длительность звука фонемы.

$$L = \frac{2}{0,05} = 40.$$

В отличие от классификатора, структура слоёв генератора (рис. 4) приводит к расширению тензора. Выходной слой — линейный, без функции активации. При прохождении тензора через этот слой получается вектор из 88 196 значений, кодирующих амплитуду сигнала, который затем с помощью инструментов «torchaudio» и бэкенда «FFmpeg» преобразуется в звуковую дорожку длительностью 2 с (несколько таких дорожек объединяются в одну).

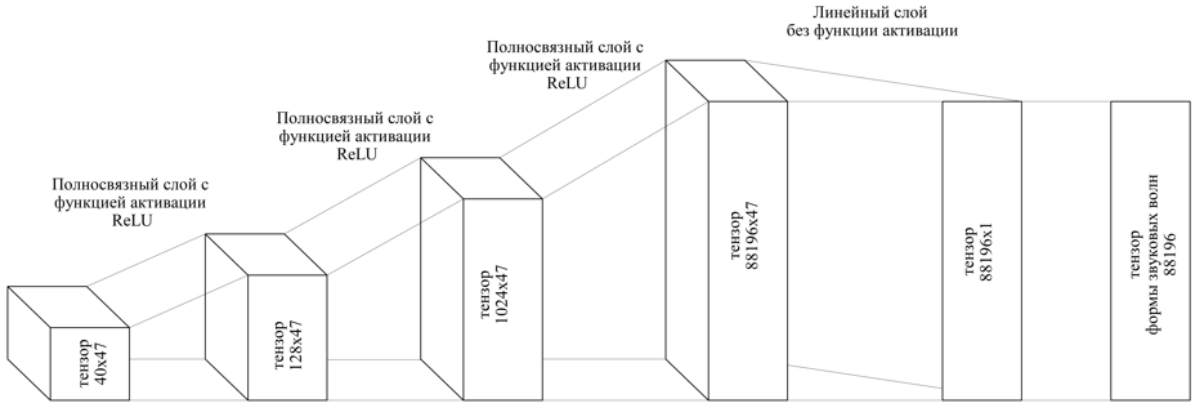

Рис. 4. Схема конфигурации слоев генератора

Обучение модели проводилось на базе фреймворка «Pytorch» [5] с применением оптимизатора RMSprop, который корректирует скорость обучения, опираясь на среднеквадратичное значение уменьшения градиентов. Применение этого оптимизатора снижает колебательность функции потерь и ускоряет процесс обучения.

В результате тестирования модели были получены формы сигнала, представленные на рис. 5. Тестирования проводились с наборами из 20, 140 и 400 аудиофайлов с желаемым голосом в режиме одного канала и при одинаковом битрейте. Как видно — наилучшее, но все ещё далёкое от требуемого совпадение формы сигнала достигается при обучении на максимальной выборке, которая в данном случае составила 400 аудиофайлов; при этом средняя доля совпадения символов после распознавания речи составляет 73 %. Сравнение лучшей полученной формы сигнала с эталонной приведено на рис. 6. При выборке из 140 аудиофайлов совпадений значительно меньше, однако уже проявляются характерные черты оригинала — средняя доля совпадения символов после распознавания речи при таком размере выборки равна 45 %. После обучения на выборке из 20 аудиофайлов совпадение сигнала с желаемым минимально, и распознаватель речи не способен дать корректный ответ.

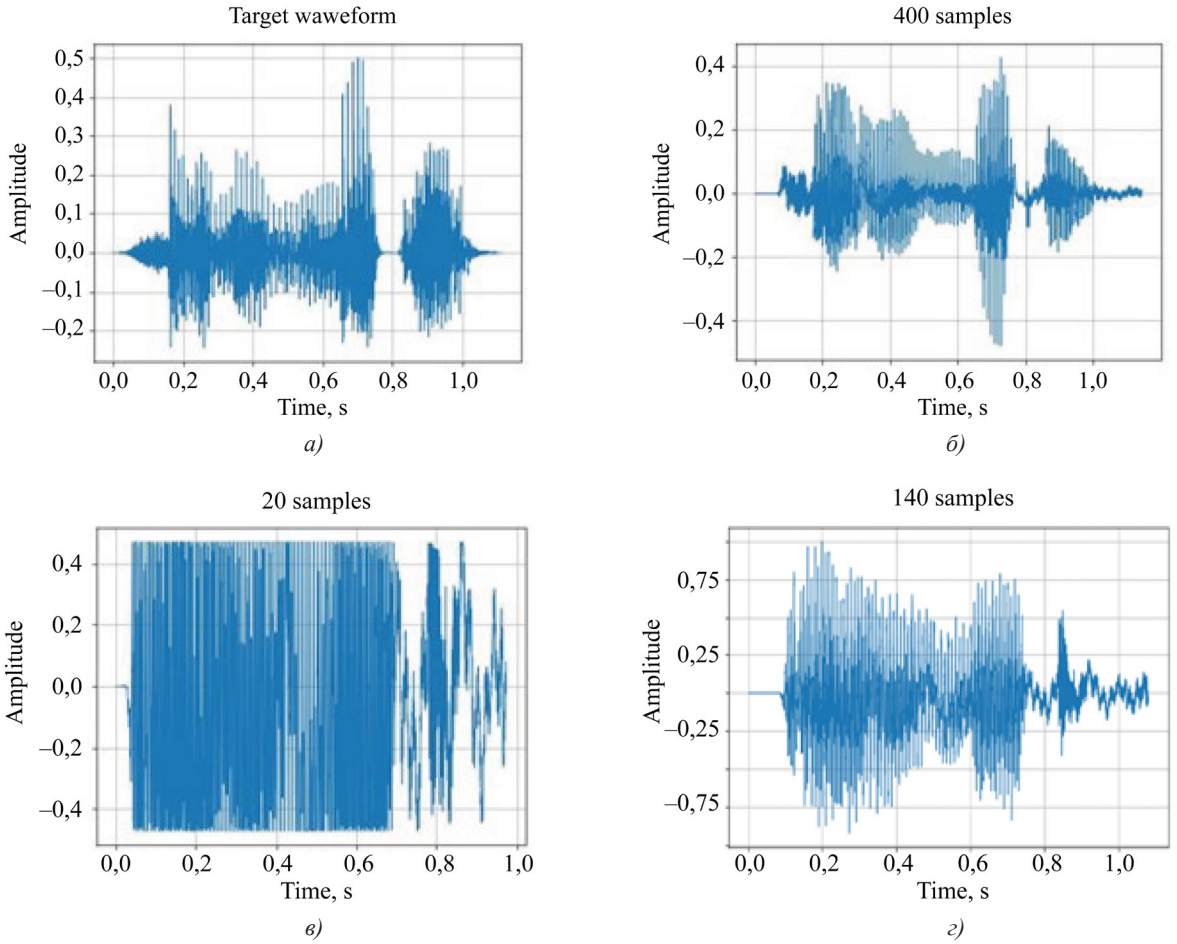

Рис. 5. Результаты тестирования модели: *а* — желаемая форма сигнала (текст, озвученный «живым» голосом);
б — сгенерированная после обучения на выборке из 20 аудиофайлов;
в — сгенерированная после обучения на выборке из 140 аудиофайлов;
г — сгенерированная после обучения на выборке из 400 аудиофайлов

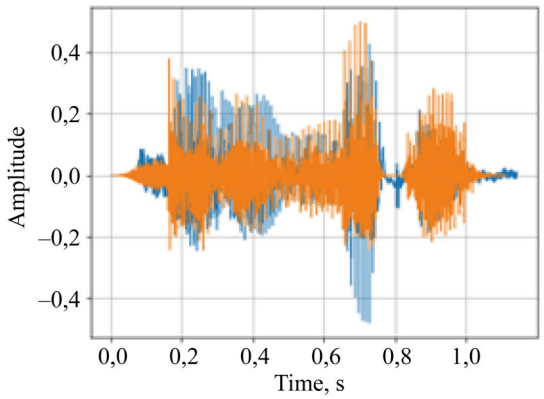

Рис. 6. Сравнение желаемой и наиболее удачной сгенерированной формы волны,
где синим цветом обозначена сгенерированная, оранжевым — желаемая

Подытожив результаты тестирования, можно констатировать, что разработанная генеративно — соревновательная нейронная модель при обучающей выборке из 400 аудиофайлов имеет ошибку озвучивания текста примерно 27 % — для снижения этого показателя необходимо увеличить объём обучающей выборки.

Для удобства пользователей была создана графическая оболочка (GUI). Поскольку фреймворк обучения модели реализован на языке Python, целесообразно применять библиотеку pyQt и инструмент QtDesigner [6], которые также используют этот язык. Изображение разработанного GUI представлено на рис. 7.

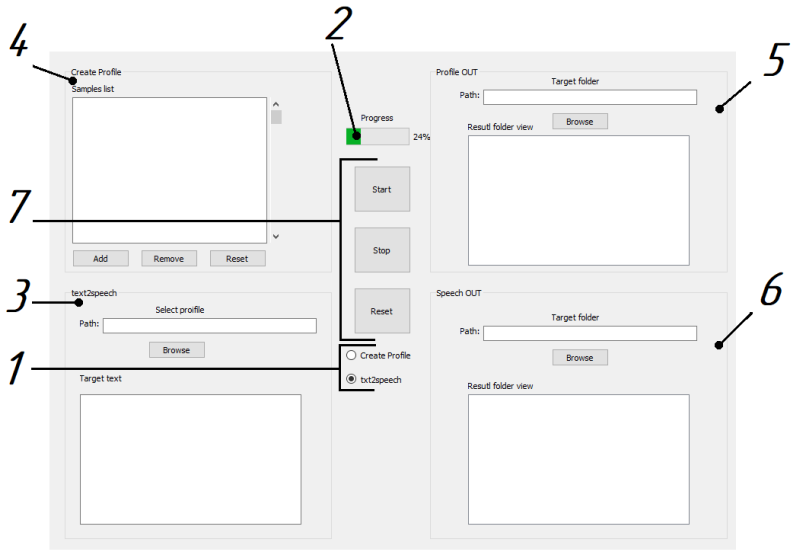

Рис. 7. Графический пользовательский интерфейс: 1 — флаги режима работы; 2 — полоска прогресс-бара; 3 — панель ввода исходных данных для озвучивания; 4 — панель загрузки аудиофайлов для создания профайла; 5 — панель выбора и обзора папки, в которую сохраняются профайлы; 6 — панель выбора и обзора папки, в которую сохраняются результаты озвучивания текста; 7 — кнопки управления действиями программы.

Предполагаемый алгоритм взаимодействия пользователя с программой описан далее. Вначале работы, при отсутствии профайла, пользователь должен создать его. Для этого используется панель 4. С помощью кнопки «Add» пользователь может добавить в список аудиофайлы с требуемым голосом. Кнопки «Remove» и «Reset» позволяют удалить выбранный аудиофайл из списка и очистить весь список соответственно. Затем на панели 5 с помощью кнопки «Browse» выбирается папка, в которую будет сохранён профайл. Пользователь переключает активный флаг на «Create Profile» в 1 и нажимает кнопку «Start» в 7, после чего ожидает окончания обработки — о чём сигнализирует состояние прогресс-бара 2.

При наличии профайла пользователь использует кнопку «Browse» на панели 3 для выбора профайла, после чего в поле «Target text» вводит текст, подлежащий озвучиванию. Далее выбирается папка для сохранения озвученных фрагментов с помощью кнопки «Browse» на панели 6. Для запуска обработки необходимо переключить флаг 1 на «text2speech» и нажать «Start» в 7, затем ожидать завершения операции.

Заключение. В результате работы достигнута поставленная цель — разработана программа, способная озвучивать введённый текст заданным голосом, формируемым подборкой аудиофайлов. В основе решения — предобученная генеративно-состязательная нейронная сеть, обрабатывающая тензоры, составленные с использованием словаря «фонема-тензор» из слов введённого текста. По результатам тестирования сети можно сделать вывод, что архитектура модели обеспечивает требуемую функциональность, однако нуждается в доработке для повышения качества выходных результатов.

Список литературы

1. *PyTorch. HiFi GAN.* URL: https://pytorch.org/hub/nvidia_deeplearningexamples_hifigan/ (дата обращения: 13.11.2025).

2. *Hugging Face. Введение в аудиоданные*. URL: https://huggingface.co/learn/audio-course/ru/chapter1/audio_data (дата обращения: 13.11.2025).

3. *CMUSphinx. About CMUSphinx – CMUSphinx Open Source Speech Recognition*. URL: https://cmusphinx.github.io/wiki/about/ (дата обращения: 15.11.2025).

4. *GitHub. ASR models for CMU Sphinx for Russian language, trained on voxforge.org*. URL: https://github.com/nsu-ai-team/voxforge_ru_sphinx (дата обращения: 15.11.2025).

5. *PyTorch*. URL: https://pytorch.org/ (дата обращения/accessed: 02.12.2025).

6. *Getting Started with Qt. Qt 6.9*. URL: https://doc.qt.io/qt-6/gettingstarted.html (дата обращения: 15.11.2025).

Об авторах:

Алексей Сергеевич Серенко, магистрант кафедры «Робототехника и мехатроника» Донского государственного технического университета (344003, Российская Федерация, г. Ростов-на-Дону, пл. Гагарина, 1), aleksei.serenko@yandex.ru

Евгений Анатольевич Лукьянов, кандидат технических наук, доцент кафедры «Робототехника и мехатроника» Донского государственного технического университета (344003, Российская Федерация, г. Ростов-на-Дону, пл. Гагарина, 1), elukianov@donstu.ru

Конфликт интересов: **авторы заявляют об отсутствии конфликта интересов.**

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the Authors:

Aleksey S. Serenko, Master's Degree Student of the Robotics and Mechatronics Department, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, Russian Federation), aleksei.serenko@yandex.ru

Evgeny A. Lukyanov, Cand.Sci. (Engineering), Associate Professor of the Robotics and Mechatronics Department, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, Russian Federation), elukianov@donstu.ru

Conflict of Interest Statement: **the authors declare no conflict of interest.**

All authors have read and approved the final manuscript.