Известия СПбГЭТУ «ЛЭТИ». 2026. Т. 19, № 8. С. 103–115
LETI Transactions on Electrical Engineering & Computer Science. 2026. Vol. 19, no. 8. P. 103–115

УДК 621.3 Научная статья

https://doi.org/10.32603/2071-8985-2026-19-8-103-115

Фильтрация шумовых помех в сигналах пьезодатчиков посредством нейронных сетей в бортовой космической аппаратуре

Д. П. Григорьев✉, К. Е. Воронов

Самарский национальный исследовательский университет
имени академика С. П. Королева, Самара, Россия

✉ dan-22225@yandex.ru

Аннотация. Рассматривается проблема фильтрации шумовых помех в сигналах пьезоэлектрических датчиков, используемых в бортовой космической аппаратуре. Представлены результаты экспериментальных исследований по применению трех типов нейронных сетей: нейронной сети прямого распространения, автокодировщика и сверточной нейронной сетей. Исследование включает метод сравнения качественных характеристик фильтрованного сигнала через нейронный фильтр с выходным сигналом цифрового фильтра. В качестве сравнительных характеристик выступали среднеквадратичная ошибка, средняя абсолютная ошибка, соотношение сигнал/шум и коэффициент корреляции. Основные результаты показали, что сверточные нейронные сети демонстрируют наилучшие показатели фильтрации высокочастотных помех при сохранении основных характеристик сигнала. Проведенное исследование позволило определить оптимальные параметры обучения для каждой архитектуры нейронной сети. Практическая значимость исследований заключается в возможности применения разработанных методов для повышения надежности систем мониторинга космических аппаратов и защиты от микрометеороидной опасности. Полученные результаты могут быть использованы при разработке новейших систем контроля поверхности космических аппаратов.

Ключевые слова: пьезодатчики, космический аппарат, микроконтроллер, Python, Tensorflow, Keras, нейронные сети, фильтрация, АМ-сигналы, микрочастицы

Для цитирования: Григорьев Д. П., Воронов К. Е. Фильтрация шумовых помех в сигналах пьезодатчиков посредством нейронных сетей в бортовой космической аппаратуре // Изв. СПбГЭТУ «ЛЭТИ». 2026. Т. 19, № 8. С. 103–115. doi: 10.32603/2071-8985-2026-19-8-103-115.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Original article

Filtering of Noise Interference in Piezo Sensor Signals by Means of Neural Networks in On-Board Space Equipment

D. P. Grigoriev✉, K. E. Voronov

Samara National Research University, Samara, Russia

✉ dan-22225@yandex.ru

Abstract. The article discusses the problem of filtering noise interference in the signals of piezoelectric sensors used in on-board space equipment. Presents the results of experimental studies on the use of three types of neural networks: direct propagation neural network (FFNN), autoencoder (AE) and convolutional neural networks (1D CNN). The main results showed that convolutional neural networks demonstrate the best filtering performance of high-frequency interference while maintaining the basic characteristics of the signal. The conducted research allowed us to determine the optimal learning parameters for each architecture of the neural network. The practical significance of the work lies in the possibility of using the developed methods to increase

© Григорьев Д. П., Воронов К. Е., 2026

the reliability of spacecraft monitoring systems and protect against micrometeoroid hazards. The results obtained can be used in the development of the latest spacecraft surface control systems.

Keywords: piezoelectric sensors, spacecraft, microcontroller, Python, Tensorflow, Keras, neural networks, filtering, AM signals, microparticles

For citation: Grigorev D. P., Voronov K. E. Filtering of Noise Interference in Piezo Sensor Signals by Means of Neural Networks in On-Board Space Equipment // LETI Transactions on Electrical Engineering & Computer Science. 2026. Vol. 19, no. 8. P. 103–115. doi: 10.32603/2071-8985-2026-19-8-103-115.

Conflict of interest. The authors declare no conflict of interests.

Введение. Расширение космической деятельности, включая частные инициативы и международные проекты, увеличивает потребность в надежных методах мониторинга и защиты от микрометеороидов – мелких космических тел размером от долей миллиметра до нескольких сантиметров. Несмотря на малые размеры, такие объекты представляют серьезную угрозу из-за высоких скоростей столкновения (от 10 до 70 км/с), способных привести к повреждению обшивки и внешних элементов конструкции; разгерметизации отсеков аппарата; выходу из строя критически важного оборудования. Это особенно актуально для долгосрочных миссий, межпланетных экспедиций и проектов по освоению ресурсов в космосе. Исследование, в частности, направлено на контроль поверхности космического аппарата или орбитальной станции, чтобы вовремя диагностировать место удара для своевременного ремонта.

В качестве датчиков контроля поверхности в исследовании применяются пьезоэлектрические преобразователи. Достоинство таких датчиков состоит в дешевизне и простоте конструкции. Однако такой тип сенсоров очень чувствителен как к электромагнитным, так и к механическим (звуки вентиляций или двигателей) помехам. Избавляться от текущих помех можно с помощью полосовых фильтров различных типов, выполненных как на аналоговой элементной базе, так и на основе цифровой технологии. Оба метода требуют тщательного моделирования и настройки, что может повлечь за собой большие временные траты, поскольку настройка должна проводиться под каждый аппарат индивидуально.

Особым решением данной проблемы служит применение обученной модели нейронной сети, которая подвергается тренировке большим массивом данных для конкретного сигнала, который в дальнейшем будет регистрироваться космической аппаратурой. Для пьезоэлектрических преобразователей это амплитудно-модулированный (АМ) сигнал, с помощью огибающей которого можно извлечь многие параметры для решения задачи детектирования и локализации микрометеороидов. Это делает тему актуальной для междисциплинарных исследований на стыке космической инженерии и астрономии.

Экспериментальный стенд. В эксперименте для локализации места удара была использована алюминиевая пластина (рис. 1) [1]–[4]. В качестве имитатора микрометеороида служила алюминиевая сфера радиусом 3.5 мм. В средней части алюминиевой пластины располагались 4 пьезодатчика, которые регистрировали виброакустические волны, возникающие вследствие удара шарика о пластину (точки S_1–S_4).

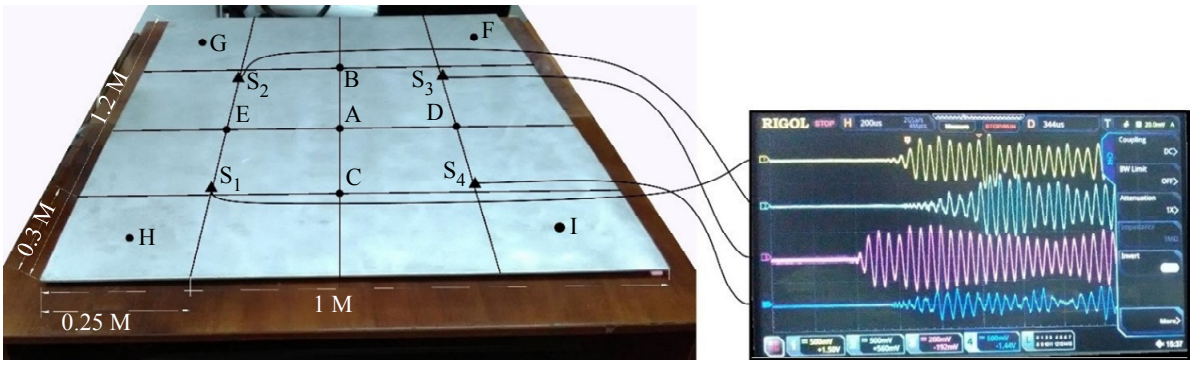


Рис. 1. Экспериментальный макет для детектирования и локализации удара
Fig. 1. Experimental stand for impact detecting and localization

Массив выборок с четырех пьезодатчиков захватывался 4-лучевым осциллографом Rigol (справа на рис. 1). Всего было произведено 900 измерений. В частности, по 20 ударов шариком приходилось на каждую из областей пластины, на рис. 1 они представлены в виде областей точек A–I. Кроме того, удары осуществлялись с пяти разных высот от 10 до 50 см с шагом 10 см. Результаты измерений копировались на переносной USB-накопитель в формате *.csv. Далее экспериментальные данные преобразовывались парсером, написанным на языке C#, для удобного представления сигналов в дальнейших программах обработки (рис. 2).

Как видно из графиков, сигналы $u(t)$ с пьезодатчиков представляют собой напряжение амплитудно-модулированного процесса (АМ-сигнала). Для точной регистрации времени появления таких сигналов необходимо выделить его огибающую. Для этого применялось преобразование Гильберта:

$$a_1(t) = a(t) \cdot h(t) = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{a(t)}{t - \tau} d\tau,$$

где $h(t) = 1/\pi t$ – ядро Гильберта; $a(t)$ – действительный АМ-сигнал пьезодатчика; τ – переменная интегрирования.

Аналитический сигнал записывается в следующем виде:

$$s_a(jt) = a(t) + ja_1(t),$$

где $a_1(t)$ – гильбертово преобразование действительного сигнала; j – мнимая единица.

Получим следующую формулу огибающей аналитического сигнала:

$$A(t) = \left| s_a(jt) \right| = \sqrt{a^2(t) + ja_1^2(t)}.$$

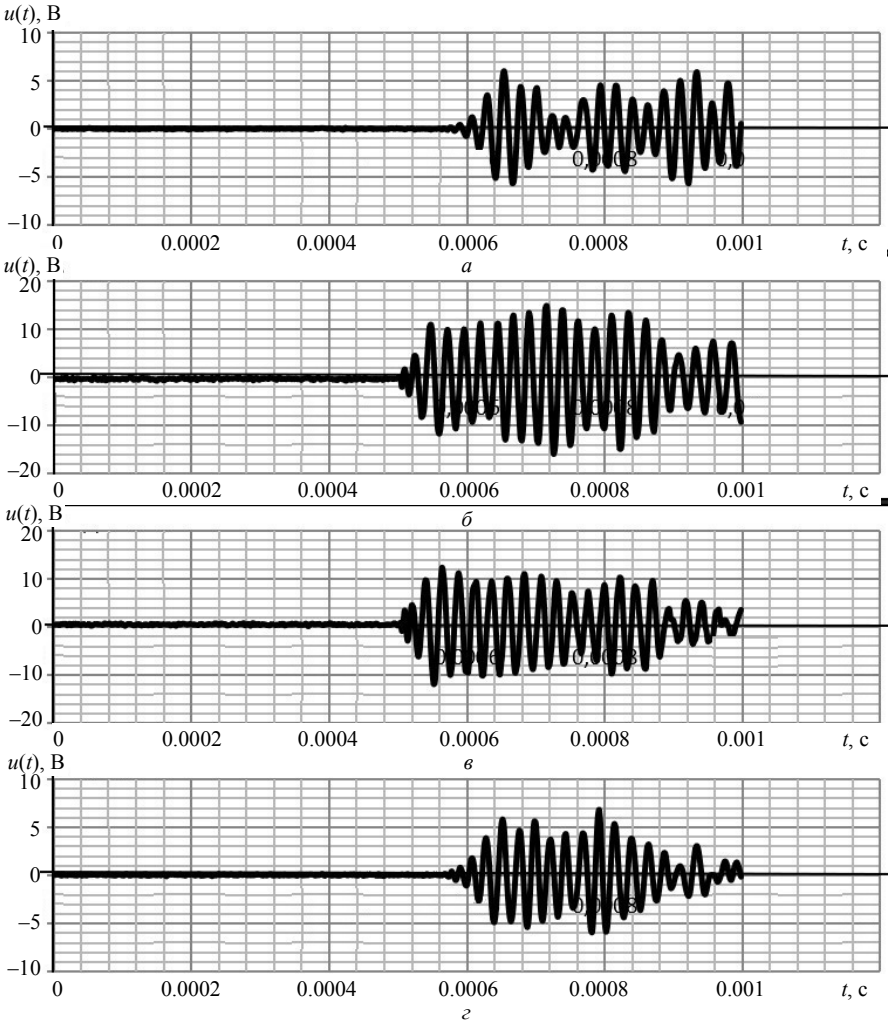


Рис. 2. Временные сигналы с четырех пьезодатчиков на выходе осциллографа (удар в точке «B»):
а – датчик 1; *б* – датчик 2; *в* – датчик 3; *г* – датчик 4
Fig. 2. Timing diagrams from fourth piezosensors by oscillograph (impact in a point «B»):
а – sensor 1; *б* – sensor 2; *в* – sensor 3; *г* – sensor 4

Далее проводился корреляционный анализ между огибающей сигнала и прямоугольным импульсом

$$K^1(\tau) = \int_{-\infty}^{+\infty} s_1(t) s_2(t+\tau)\, dt = \int_{-\infty}^{+\infty} A(t) \operatorname{sign}(t+\tau)\, dt,$$

где s_1 и s_2 – сигнал с пьезодатчика и прямоугольный импульс соответственно.

Результаты представляли собой кривые корреляционного интеграла, которые меняли свой знак в определенные промежутки времени. Эти отсчеты времени и служили временной точкой регистрации импульса с пьезодатчика.

Затем полученные временные задержки подавались на вход специально обученной нейронной сети, которая определяла, в какой из областей пластины был сделан удар шариком. Результаты дали ответы с точностью, превосходящей 90–95 %.

Однако стоит учесть, что точность данной методики напрямую зависит от качества записанного сигнала с пьезодатчиков. В лабораторных условиях были получены минимальные шумы, что не соответствует полевым условиям. Сигналы пьезодатчиков чувствительны ко всем видам шумов, в том числе к низкочастотным (работа вентиляции на станции, двигателей и т. д.) и высокочастотным (работа высокочастотной аппаратуры, компьютеров, блоков питания, космической радиации и т. д.). Вследствие широкого распространения нейронных сетей в самых разных прикладных задачах [5], [6] было принято решение о попытке применения нейронных сетей в качестве шумовых фильтров АМ-сигналов [3].

Выбор нейронной сети. В качестве нейронных сетей были выбраны три часто используемые архитектуры – нейронная сеть прямого распространения (FFNN – Feedforward Neural Network, рис. 3), автокодировщики (AE – Autoencoder, рис. 4) и сверточные нейронные сети (1D-CNN – One-Dimensional Convolutional Neural Network, рис. 5). Сравнение основных характеристик нейронных сетей представлено в табл. 1.

Нейронная сеть прямого распространения [2] в начале схемы имеет входной слой с нейронами, содержащими обучаемые выборки данных (сигнала). Средние слои называются скрытыми слоями (hidden layers), которые содержат основные обучаемые нейроны. Также они называются полносвязными слоями (Dense). Количество скрытых слоев и нейронов варьируются в зависимости от исхода обучения. Выходной слой – это финальный слой нейронной сети, который содержит результат в соответствующих нейронах.

Переход данных от нейрона к нейрону происходит через функцию активации, нормализующую значение до требуемого диапазона (например, от –1.0 до 1.0 или от 0.0 до 1.0). Функции активации бывают разных типов: relu, sigmoid, tanh и т. д. Для оценки успешного обучения нейронной сети от числа пройденных эпох, применяются различные функции расчета потерь (процент отличия между предсказанным и истинным результатом), например MSE (Mean Squared Error), MAE (Mean Absolute Error), binary crossentropy, categorical crossentropy, KLDivergence, cosine similarity и др.

Для минимизации функции потерь применяют различные оптимизаторы, которые повышают производительность обучения, исключают локальные минимумы градиента обучения, ускоряют обучение, динамически изменяя коэффициент скорости, и т. д. Из распространенных оптимизаторов широко известны: Adam, Nadam, Adagrad, Adamax, RMSprop, SGD.

Основная идея состоит в том, что информация с каждой выборки из входного слоя перемножается с уникальным коэффициентом (весом W). Результат перемножения проходит через функцию активации и суммируется с остальными результатами от других выборок. И это – для одного нейрона следующего скрытого слоя. Далее операция повторяется аналогично для всех остальных нейронов всех скрытых слоев, затем результаты выводятся в выходной слой.

Во время обучения рассчитывается ошибка между выходными и входными данными. По значению ошибки применяется функция коррекции, которая в обратную сторону (от выходного слоя ко входному) проводит соответствующую корректировку весов W – это происходит в рамках одной эпохи. В процессе обучения веса подстраиваются таким образом, чтобы разница между входными и выходными данными была минимальной, в чем и заключается суть обучения нейронных сетей.

Нейронная сеть автокодировщика – частный случай нейронной сети прямого распространения, за исключением того, что средний слой имеет так называемый узкий слой, который имеет наименьшее число нейронов [2]. Количество входных и выходных нейронов одинаково. От

«узкого» слоя по возрастанию в сторону входного и выходного слоев располагаются другие скрытые слои. Таким образом, нейронная сеть прогоняет сигнал через себя, выделяя самые яркие признаки данных, которые могут послужить для восстановления сигнала. Данный способ хорошо подходит для задач по сжатию данных или для сглаживания сигнала.

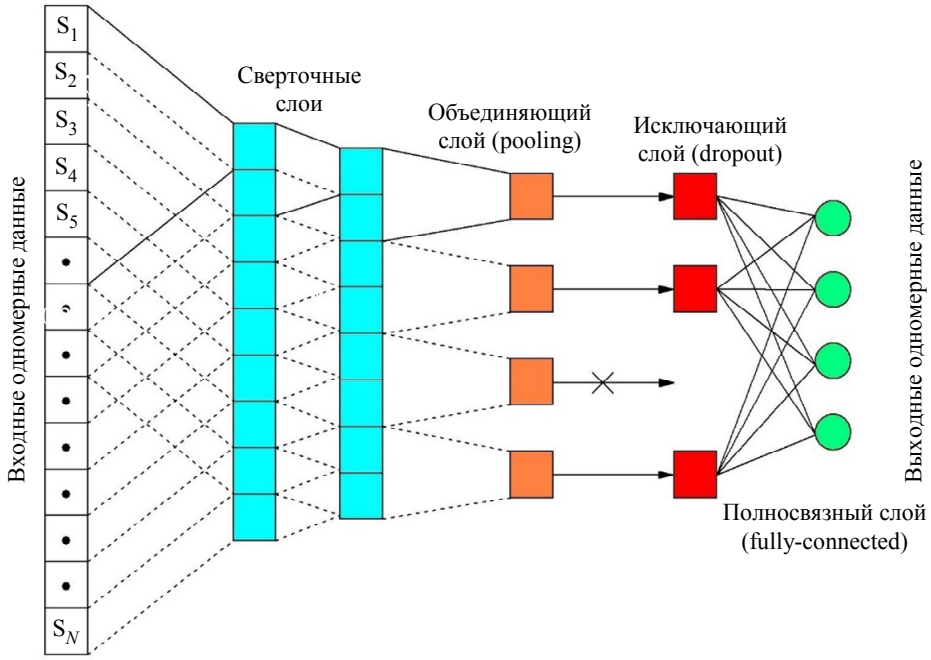


Рис. 3. Схема архитектуры одномерной сверточной нейронной сети
Fig. 3. Scheme of architecture of 1-D convolution neural network

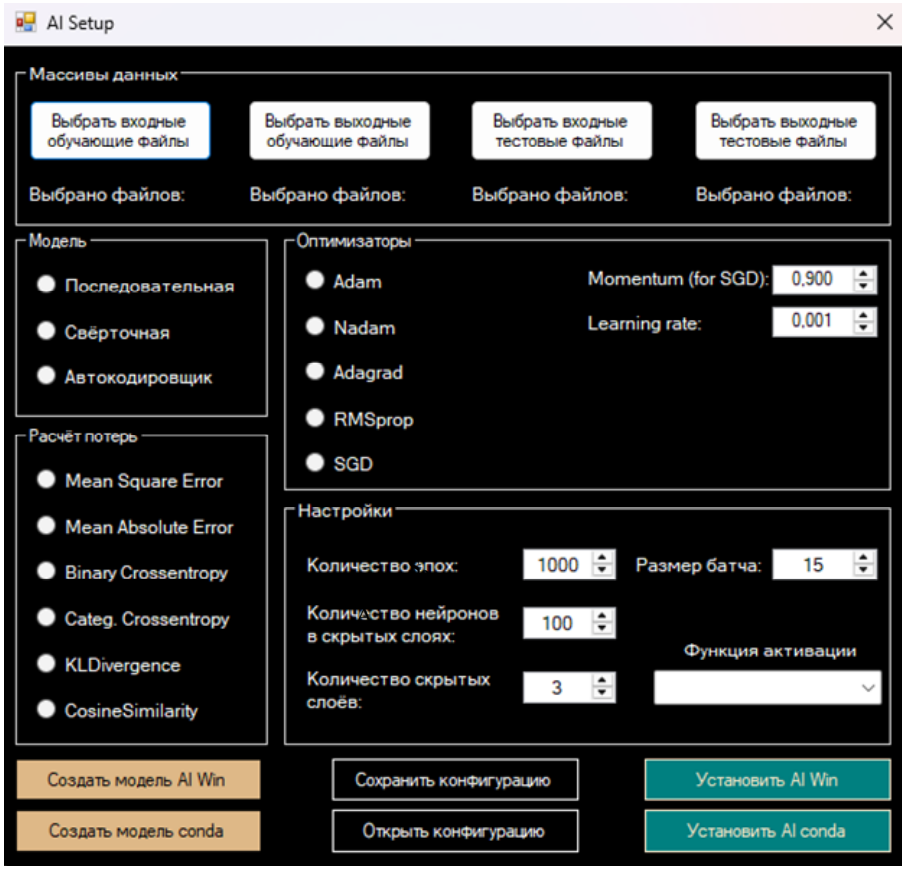


Рис. 4. Скриншот интерфейса API-модуля для настройки и запуска нейросетей
Fig. 4. Screenshot of the module's API interface for configuring and running neural networks

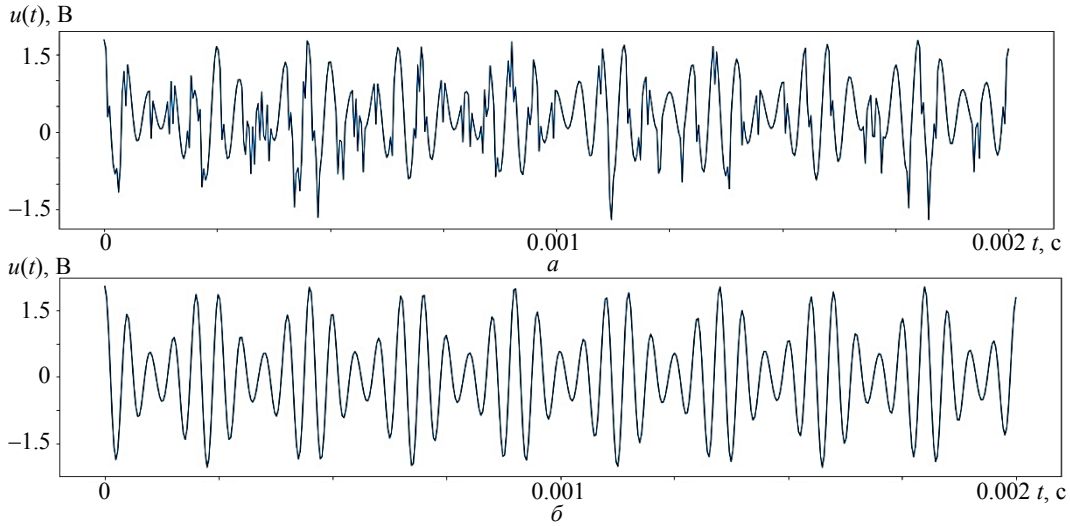


Рис. 5. Графики зашумленного (*а*) и эталонного (*б*) АМ-сигналов (U_0 = 0.85 В; M = 0.6; M_x = 0.6; σ = 0.6)
Fig. 5. Diagram of noised (*a*) and reference (*б*) AM-signal (U_0 = 0.85 V; M = 0.6; M_x = 0.6; σ = 0.6)

Табл. 1. Сравнение нейронных сетей по основным критериям
Tab. 1. Comparison of neural networks by main criteria

Критерий	FFNN	AE	1D CNN
Основная задача	Моделирование зависимостей между входными и выходными данными	Сжатие данных, извлечение признаков	Обработка пространственных данных (изображения, видео)
Работа с временными данными	Да (через многомерные входные векторы)	Ограниченно (требует дополнительных подходов)	Необходимо применять одномерный 1D-слой
Сложность обучения	Умеренная	Высокая (требуется обучение кодировщика и декодера)	Высокая (требует больших данных и вычислительных ресурсов)
Адаптивность	Высокая (после обучения)	Средняя (зависит от качества обучения)	Средняя (эффективна для статичных сигналов)

Сверточные сети широко известны по классификации изображений или по принятию решений по изображениям. Входные слои имеют 2D-матрицу из нейронов (для монохромного изображения) или 3D-матрицу (тензор) для цветного RGB-изображения. В нашем случае сигнал одномерный, поэтому сверточная сеть будет иметь входной слой 1D.

Отличие от первых двух нейронных сетей состоит в том, что связь с нейронами происходит с помощью усреднения информации сверткой данных с ядром следующего слоя. Стоит отметить, что свертка происходит с перекрыванием, т. е. раннее используемые значения попадают в окно ядра при его перемещении.

Помимо свертки данных с ядром, сверточные нейронные сети содержат еще несколько дополнительных операций:

1. pooling – специальный слой, который уменьшает размер предыдущего слоя в результате выбора данных по окну (окно перемещается по предыдущему слою, не перекрывая используемые раннее значения). Различают два вида уменьшения: max pooling – в следующий слой попадает максимальное значение из окна; average pooling – в следующий слой попадает среднее значение из окна;

2. dropout – слой регуляризации сети посредством отключения случайным образом некоторых нейронов для предотвращения переобучения и оптимизации обучения;

3. fully connected – полносвязный одномерный нейронный слой, в который преобразуются конечные матрицы сверточной нейронной сети (слой сети прямого распространения). Не путать с flatten;

4. flatten – слой выполняет преобразование многомерных тензоров в одномерный вектор, что необходимо для передачи данных в полносвязные слои нейронной сети. На рис. 3 flatten идет между слоями dropout и fully connected.

Таким образом получается синтез сверточной нейронной сети и сети прямого распространения. Слоев fully connected может быть сколько угодно,

и они будут считаться скрытыми. Кроме того, количество слоев самой сверточной сети также поддается изменению.

Подготовка данных обучения и запуск нейронной сети. Для данного эксперимента по фильтрации АМ-сигналов был написан Python-скрипт с реализацией нескольких видов архитектур нейронных сетей, описанных ранее, с использованием библиотеки нейронных сетей Tensor Flow [7] – через API Keras [8]. Скрипт позволяет гибко менять параметры через стартовые аргументы, что ускорило процесс подбора параметров для нейронных сетей. Помимо скрипта, был написан API-модуль на языке C#, предоставляющий необходимые инструменты для удобного переключения параметров обучения нейронной сети, которые впоследствии передавались интерпретатору python при запуске скрипта (рис. 4). Основные параметры, которые можно выставлять в программе:

– папки с входными и выходными обучаемыми данными;

– папки с входными и выходными тестовыми данными;

– модель (архитектура) нейронной сети;

– функция потерь;

– оптимизатор;

– число нейронов в скрытых слоях;

– число скрытых слоев;

– число эпох;

– размер батча;

– функция активации (для всех слоев).

Помимо этого, приложение позволяет запускать обучение как в отдельном виртуальном окружении Python «env», так и в модуле miniconda [9]. Если на ПК отсутствуют данные окружения или модули, соответствующие кнопки «Установить AI Win/Установить AI conda» позволят пользователю автоматически установить все необходимое для полноценного запуска нейронной сети через сценарий *.bat [10].

В данной задаче для обучения было необходимо подготовить 1000 файлов с чистыми и зашумленными дискретными АМ-сигналами, подобно оригинальным сигналам, снятым во время эксперимента. Их характеристики:

1) амплитуда U_0 от 0.1 до 0.9 В;

2) частота несущего сигнала F = 20 кГц;

3) частота модуляции f_0 = 1кГц;

3) глубина модуляции M: 55–75 %;

4) число выборок: 500;

5) частота дискретизации f_S = 2 GHz.

Зашумленные сигналы получены аддитивным применением выборок шума с гауссовым распределением. Параметры шума были выбраны с учетом устойчивого распознавания сигнала (иначе сигнал превращался в неразборчивый шум). Его характеристики:

1) математическое ожидание M_x: 0.6;

2) среднеквадратичное отклонение σ: 0.1…0.6.

Зашумленный сигнал подвергался нормализации (весь ансамбль выборок шума в сигнале делился на максимальную выборку из этого ансамбля), чтобы предотвратить высокие выбросы амплитуды.

Указанные ранее характеристики чистого и шумного АМ-сигналов менялись случайным образом от файла к файлу. Таким образом составлялся большой набор данных для последующего обучения нейронных сетей. Примеры сигналов показаны на рис. 5 при разных параметрах. Сигналы были сгенерированы при помощи скрипта, написанного на языке Python.

Стоит отметить, что все 1000 файлов с сигналами подаются на вход нейронной сети группами (батчами) в рамках одной эпохи. Другими словами – все файлы делятся на небольшие подмножества обучающих данных, которые обрабатываются моделью за одну эпоху перед обновлением ее параметров. Их использование решает сразу несколько ключевых задач:

1. Снижение нагрузки на память.

2. Возможность обучения модели даже на ограниченных ресурсах.

3. Ускорение вычислений.

4. Обновление весов на основе градиента по батчу дает более точную оценку направления оптимизации, чем по одному примеру. Это снижает «шум» в обновлениях и делает траекторию обучения более гладкой.

5. Баланс между скоростью и качеством. Маленькие батчи (16–32) вносят стохастичность: модель лучше избегает локальных минимумов, но ее обучение дольше. Большие батчи (256–1024) ускоряют сходимость, но могут привести к переобучению или застреванию в неоптимальных точках.

Далее представлены выкладки из кода на Python, которые задают модели нейронной сети. Все параметры нейронных сетей заданы с помощью словаря *config*, в который помещаются значения через входящие аргументы. Первым делом данные приводятся в кортежи (tuple), которые идут на вход модели. Допустим, имеется массив 1000 типов АМ-сигнала по 500 выборок в каждом: train_AM[1000][500]. Тогда для преобразо-

вания всех 500 выборок в единый кортеж из 1000 элементов используется метод:

input_train = tuple(*train_AM*)

Далее кортежи преобразуются в кадры данных:

frame_train = pd.DataFrame(*input_train*)

Наконец, преобразуем массив в тип с плавающей запятой:

finish_input_train = frame_train.astype(*'float32'*)

Следующий шаг – сборка нейронной сети. Для случая FFNN вначале задается модель, которая является последовательной [8]:

model = Sequential()

Далее добавляется входной слой, в параметры которого включаются размер и количество скрытых нейронов, функция активации во входном слое и размер входного слоя [8]:

model.add(*Dense*(*int*(*config*['*hidden_size'*]),
activation = config['*activation'*],
input_dim = num_in))

Затем добавляется *N* – количество скрытых слоев с параметрами размера слоя и функции активации на слое [8]:

model.add(*Dense*(*int*(*config*['*hidden_size'*]), *activation = config*['*activation'*]))

Предпоследний этап – сборка нейронной сети, в которой устанавливается тип функции расчета потерь, тип оптимизатора и необходимые метрики для анализа обучения [8]:

model.compile(*loss = config*[*'loss_function'*],
optimizer = config['*optimizer'*],
metrics = config['*accuracy'*])

Наконец, запуск нейронной сети выполняется функцией fit, в которой указываются входные и выходные наборы данных, размер батча, количество эпох, а также флаг перемешивания наборов данных от эпохи к эпохе [8]:

history = model.fit(*X_train*, *Y_train*,
batch_size = int(*config*['*batch_size'*]),
epochs = int(*config*['*epoch_size'*]),
shuffle = True)

Для одномерной сверточной сети сборка происходит аналогично. Сначала выбирается последовательная модель Sequential(). После этого добавляются слои, но с немного другими параметрами. Вместо класса Dense применяется класс Conv1D, в который передается количество фильтров, размер ядра свертки, функция активации и размер входной последовательности [8]:

model.add(*Conv1D*(*filters = int*(*config*[*'filters'*]),
kernel_size = int(*config*[*'kernel_size'*]),
activation = config['activation'],
input_shape = (*sequence_length*, *1*)))

Далее добавляем слой MaxPooling [8]:

model.add(*MaxPooling1D*(*pool_size = int*(*config*['*pool_size'*])))

Затем добавляем слой Flatten [8]:

model.add(*Flatten*())

Последние два слоя будут обычными FFNN (один скрытый и один выходной соответственно) [8]:

model.add(*Dense*(*int*(*config*["*hidden_size*"]), *activation=config*["*activation*"]))

model.add(*Dense*(*num_out*, *activation=config*["*activation*"]))

Этапы сборки и запуска обучения такие же, как уже описано.

Для случая с автокодировщиком этап сборки немного отличается и похож скорее на обратный описанному в первых двух случаях. Смысл заключается в том, что теперь сначала задаются слои, по которым будет в дальнейшем собрана готовая модель обучения.

Первым шагом идет построение входного слоя [8]:

input_img = Input(*shape*=(*num_in*,))

Затем заготавливаются слои для кодировщика [8]:

encoded = Dense(*int*(*config*["*hidden_size*"]), *activation=config*["*activation*"])(*input_img*)

В дальнейшем – аналогичная операция подготовки для декодера [8]:

decoded = Dense(*num_out*, *activation=config*["*activation*"])(*encoded*)

Наконец, совмещаем слои и получаем итоговую модель [8]:

autoencoder = Model(*input_img*, *decoded*)

Методы сборки и запуска обучения аналогичны. Во время работы нейронной сети в терминале должна высвечиваться статистика обучения, как показано на рис. 6.

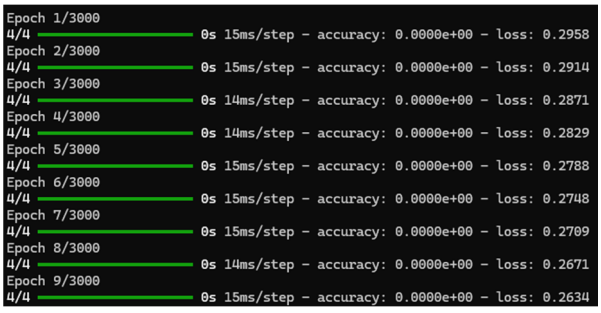


Рис. 6. Скриншот статистики обучения по эпохам
Fig. 6. Screenshot of training statistics by epoch

В конце обязательно нужно провести тест на независимой выборке сигналов и посмотреть на качество восстановленного сигнала.

Результаты обучения нейронных сетей. Наконец, по полученной истории обучения можно построить графики зависимости точности и потерь от числа эпох. Графики построены по массивам таких же значений, какие показывались в терминале (рис. 7–9).

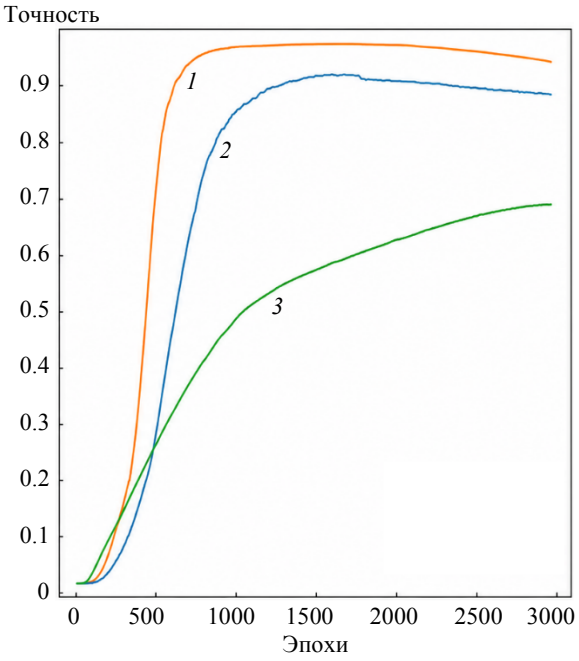


Рис. 7. Графики зависимости точности от числа эпох: *1* – 1D-CNN; *2* – FFNN; *3* – AE
Fig. 7. Plots of relationships of accuracy by epochs: *1* – 1D-CNN; *2* – FFNN; *3* – AE

Из результатов видно, что до эпохи № 1500 для FFNN (рис. 7, кривая *2*) обучение шло равномерно. В дальнейших эпохах наступило переобучение нейронной сети и точность начала спадать, поэтому можно сделать корректировку и убрать лишние эпохи. Для нейронной сети 1D CNN (рис. 7, кривая *1*) обучение происходило быстрее (резкое нарастание), чем в случае FFNN. Аналогично же виден и резкий спад из-за переобучения на эпохе № 2000, после которого сформировалась постоянная точность ~0.95. Автокодировщику (рис. 7, кривая *3*) не хватило эпох для формирования постоянной точности. Также отметим, что обучение проходило в разы медленнее по сравнению с первыми двумя архитектурами.

На рис. 8–10 продемонстрированы результаты работы обученных нейронных сетей по фильтрации АМ-сигнала.

Как видно из графиков на рис. 8, 9, во всех случаях пропала информация о глубине модуляции АМ-сигнала, но зато сохранилась частота и уровень сигнала. Амплитуда при работе FFNN стала меньше примерно в 0.34 раза, в остальных случаях она сохранилась. На рис. 10 видно, что из всех архитектур сверточная нейронная сеть показала наилучшее избавление от высокочастотных скачков, сохранив при этом даже информацию о глубине модуляции. Стоит отметить, что данный результат был достигнут на «чистой» сверточной сети, т. е. без применения слоев *pooling*, *dropout* и *dense*, так как в этом случае подавлялась информация о глубине модуляции. Из эксперимента было видно, что слой *max pooling* выбирал максимум в окне и мог отбрасывать слабые боковые составляющие АМ-сигнала. Слои *dropout* и *dense* искажали временную структуру сигнала, так как количество выборок на выходе фильтра относительно входа разнилось. Опытным путем получилась модель, состоящая из 5 слоев Conv1D:

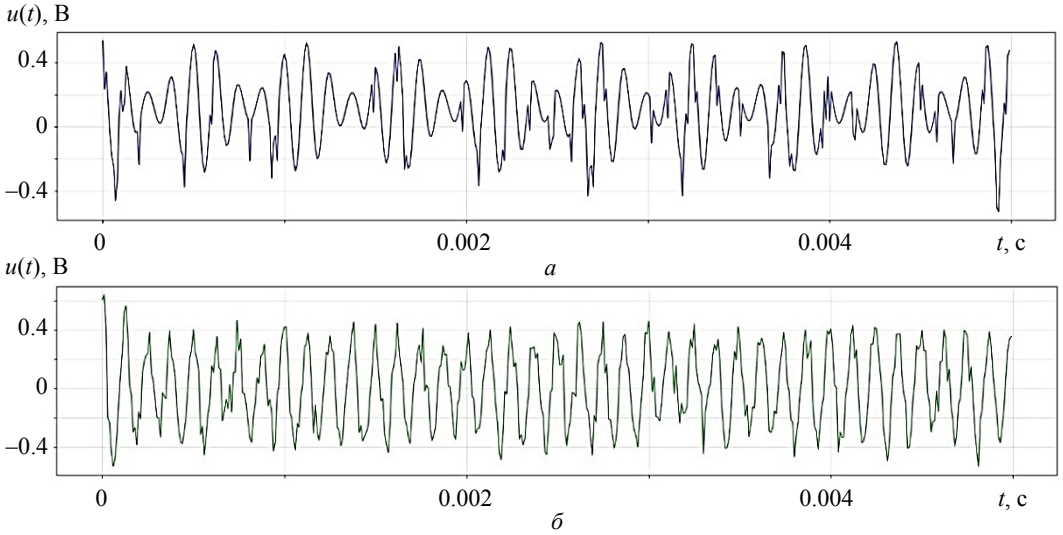


Рис. 8. Графики зашумленного (*а*) и предсказанного отфильтрованного нейронной сетью FFNN (*б*) сигналов
Fig. 8. Plots of the noisy (*а*) and predicted (*б*) pure signals by the FFNN neural network

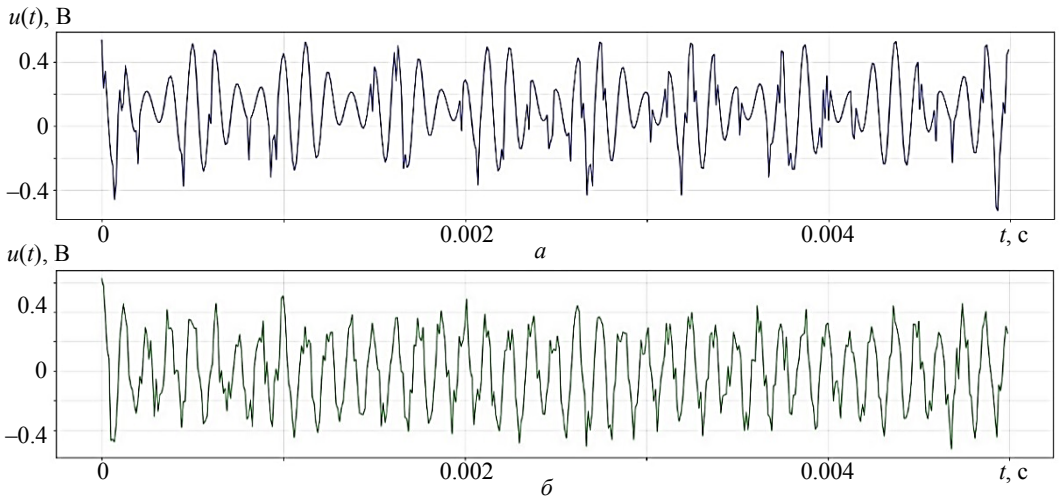


Рис. 9. Графики зашумленного (*а*) и предсказанного отфильтрованного нейронной сетью АЕ (*б*) сигналов

Fig. 9. Plots of the noisy (*a*) and predicted (*б*) pure signals by the AE neural network

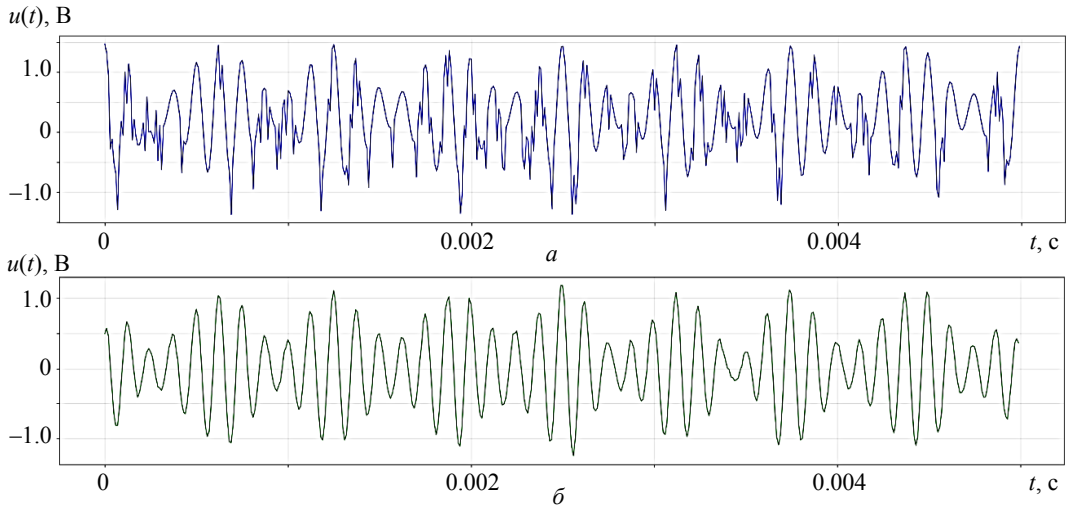


Рис. 10. Графики зашумленного (*а*) и предсказанного отфильтрованного нейронной сетью 1D-CNN (*б*) сигналов

Fig. 10. Plots of the noisy (*a*) and predicted (*б*) pure signals by the 1D-CNN neural network

Табл. 2. Характеристики сигналов «До» и «После» фильтрации нейронным фильтром и цифровым фильтром Баттерворта

Tab. 2. Characteristics of signal «before» and «after» neural filtering and digital filter of Butterworth

Параметр	MSE, В^2	MAE, В	SNR, дБ	Correlation *r*
Без фильтрации	0.123	0.35	3.71	0.89
Нейронный фильтр 1D CNN	0.011	0.08	14.1	0.98
Цифровой фильтр	0.03	0.1	9.96	0.95

model.add(*Conv1D*(*filters=32*, *kernel_size=7*, *padding*="*same*", *activation*="*linear*", *input_shape*=(*sequence_length*, *1*)))

model.add(*Conv1D*(*filters=64*, *kernel_size=15*, *padding*="*same*", *activation*="*relu*"))

model.add(*Conv1D*(*filters=64*, *kernel_size=15*, *padding*="*same*", *activation*="*relu*"))

model.add(*Conv1D*(*filters=32*, *kernel_size=7*, *padding*="*same*", *activation*="*relu*"))

model.add(*Conv1D*(*filters=1*, *kernel_size=7*, *padding*="*same*", *activation*="*linear*"))

Сравнение эталонного и фильтрованного сигналов на выходе нейронного фильтра и цифрового стандартного полосового фильтра Баттерворта приведены в табл. 2. Ошибка восстановления сигнала у нейронной сети заметно меньше, а соотношение сигнал/шум (SNR – Signal-to-Noise

Ratio) выше в 1.4 раза. Коэффициент корреляции r близок к 98 % вместо 95 %, полученных цифровым фильтром.

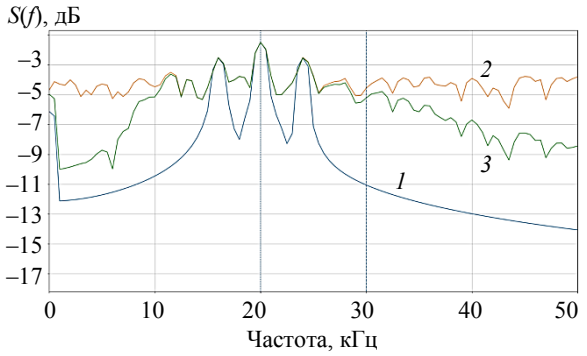


Рис. 11. Графики спектральной плотности $S(f)$ сигналов: *1* – эталонный; *2* – зашумленный; *3* – отфильтрованный цифровым фильтром Баттерворта

Fig. 11. Plots of spectral density $S(f)$ of signals: *1* – reference; *2* – noisy; *3* – filtered by a digital filter of Butterworth

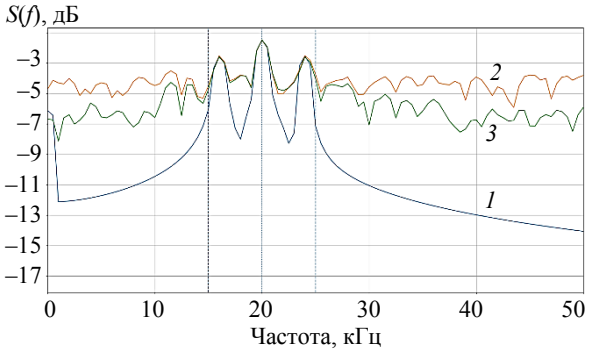


Рис. 12. Графики спектральной плотности $S(f)$ сигналов: *1* – эталонный; *2* – зашумленный; *3* – отфильтрованный нейронным фильтром

Fig. 12. Plots of spectral density $S(f)$ of signals: *1* – reference; *2* – noisy; *3* – filtered by a neural filter

Для наглядности работы фильтров приведем на рис. 11 и 12 спектральные плотности сигналов ДО и ПОСЛЕ фильтрации для цифрового и нейронного фильтров соответственно.

На рис. 11 видно, что спектр фильтрованного сигнала сильно подавляется в низких и высоких частотах. Информация о несущей и боковых гармониках сохраняется по уровню выше –3 дБ. В то же время, на рис. 12 показано, что спектр всего сигнала равномерно подавляется ниже уровня –3 дБ, что дает разбег для применения нейронной сети в качестве широкополосного полосового фильтра. Несущая и информационные гармоники значительно выделяются на ряду с цифровым фильтром.

Выводы. Таким образом, проведенный сравнительный анализ подтвердил высокую эффективность применения нейронных сетей для решения задач фильтрации гармонических сигналов. Количественные метрики – такие, как коэффициент корреляции между исходным и восстановленным сигналами, а также соотношение сигнал/шум (SNR), демонстрируют существенное преимущество нейронного фильтра над классическим цифровым аналогом – цифровым полосовым фильтром Баттерворта.

Детальное исследование различных моделей показало, что наилучшие результаты в фильтрации амплитудно-модулированных сигналов демонстрирует одномерная сверточная нейронная сеть (1D CNN). Ее ключевое достоинство заключается в сохранении информационной целостности сигнала: отфильтрованные сигналы не только сохраняют исходную форму, но и точно воспроизводят ключевые параметры – несущую частоту, глубину и частоту модуляции, а также амплитудные характеристики. Помимо качественной фильтрации, архитектура CNN продемонстрировала и высокую технологическую эффективность: скорость сходимости процесса обучения у нее оказалась заметно выше по сравнению с сетями прямого распространения (FFNN) и автокодировщиками (AE), что делает ее предпочтительным выбором для задач, где критичны временные затраты на подготовку модели.

Вместе с тем, анализ выявил и определенные ограничения нейросетевого подхода. В процессе обучения модель может допускать локальные искажения, влияющие на точность воспроизведения важных параметров модуляции. Это проявляется в отклонениях глубины модуляции или небольших флуктуациях несущей частоты, что не нарушает общую картину сигнала, но требует внимания при работе с высокочувствительными системами. Особое внимание стоит уделять архитектуре собираемой модели, поскольку разные типы слоев могут вносить существенные искажения в форму того или иного сигнала.

Список литературы

1. Воронов К. Е., Григорьев Д. П., Телегин А. М. Исследование алгоритмов для системы контроля поверхности космического аппарата на основе пьезодатчиков // Авиакосмическое приборостроение. 2021. № 1. С. 40–50. doi: 10.25791/aviakosmos.1.2021.1200.

2. Воронов К. Е., Григорьев Д. П., Телегин А. М. Применение нейронной сети прямого распространения для локализации места удара микрочастиц о поверхность космического аппарата // Тр. МАИ. 2021. № 118. С. 1–35. doi: 10.34759/trd-2021-118-10.

3. Григорьев Д. П., Телегин А. М. Применение нейронных сетей для шумоподавления сигналов с пьезодатчиков на борту космического аппарата // V Науч. форум телекоммуникации: теория и технологии – ТТТ-2021. XIX Междунар. науч.-техн. конф. «Физика и технические приложения волновых процессов – ФиТПВП-2021». Самара: ПГУТИ, 2021. С. 66–67.

4. Григорьев Д. П. Система контроля поверхности космического аппарата // Сб. мат. междунар. молодеж. науч. конф., посвящ. 60-летию полета в космос Ю. А. Гагарина «XVI Королевские чтения». Самара: Изд-во СамНИУ, 2021. Т. 1. С. 355–357.

5. Особенности разработки метамоделей на основе нейронных сетей для задач индукционного нагрева / Ф. В. Чмиленко, Ю. Ю. Перевалов, В. Е. Парменов, Чжан Ци, Ю. В. Шанин // Изв. СПбГЭТУ «ЛЭТИ». 2024. Т. 17, № 7. С. 83–96. doi: 10.32603/2071-8985-2024-17-7-83-96.

6. Мельцов В. Ю., Крутиков А. К. Параллельный метод ассемблирования нейронных сетей в подсистеме прогнозирования неисправностей высокопроизводительных вычислительных комплексов // Изв. СПбГЭТУ «ЛЭТИ». 2025. Т. 18, № 9. С. 56–67. doi: 10.32603/2071-8985-2025-18-9-56-67.

7. TensorFlow framework. URL: https://www.tensorflow.org/ (дата обращения: 10.01.2026).

8. Keras API. URL: https://keras.io/ (дата обращения: 10.01.2026).

9. Miniconda application. URL: https://www.anaconda.com/docs/getting-started/miniconda/install (дата обращения: 10.01.2026).

10. Batch File Basics. URL: https://learn.openwaterfoundation.org/owf-learn-windows-shell/batch-file-basics/batch-file-basics/ (дата обращения: 12.01.2026).

Информация об авторах

Григорьев Данил Павлович – аспирант кафедры радиоэлектронных систем, Самарский национальный исследовательский университет имени академика С. П. Королева. Московское шоссе, 34, Самара, 443086, Россия.
E-mail: dan-22225@yandex.ru
https://orcid.org/0009-0007-8413-379X

Воронов Константин Евгеньевич – канд. техн. наук, доцент кафедры радиоэлектронных систем, Самарский национальный исследовательский университет имени академика С. П. Королева. Московское шоссе, 34, Самара, 443086, Россия.
E-mail: voronov.ke@ssau.ru

References

1. Voronov K. E., Grigor'ev D. P., Telegin A. M. Issledovanie algoritmov dlja sistemy kontrolja poverhnosti kosmicheskogo apparata na osnove p'ezodatchikov // Aviakosmicheskoe priborostroenie. 2021. № 1. S. 40–50. doi: 10.25791/aviakosmos.1.2021.1200. (In Russ.).

2. Voronov K. E., Grigor'ev D. P., Telegin A. M. Primenenie nejronnoj seti prjamogo rasprostranenija dlja lokalizacii mesta udara mikrochastic o poverhnost' kosmicheskogo apparata // Tr. MAI. 2021. № 118. S. 1–35. doi: 10.34759/trd-2021-118-10. (In Russ.).

3. Grigor'ev D. P., Telegin A. M. Primenenie nejronnyh setej dlja shumopodavlenija signalov s p'ezodatchikov na bortu kosmicheskogo apparata // V Nauch. forum telekommunikacii: teorija i tehnologii – TTT-2021. XIX Mezhdunar. nauch.-tehn. konf. «Fizika i tehnicheskie prilozhenija volnovyh processov – FiTPVP-2021». Samara: PGUTI, 2021. S. 66–67. (In Russ.).

4. Grigor'ev D. P. Sistema kontrolja poverhnosti kosmicheskogo apparata // Sb. mat. mezhdunar. molodezh. nauch. konf., posvjashh. 60-letiju poleta v kosmos Ju. A. Gagarina «XVI Korolevskie chtenija». Samara: Izd-vo Sam-NIU, 2021. T. 1. S. 355–357. (In Russ.).

5. Osobennosti razrabotki metamodelej na osnove nejronnyh setej dlja zadach indukcionnogo nagreva / F. V. Chmilenko, Ju. Ju. Perevalov, V. E. Parmjonov, Chzhan Ci, Ju. V. Shanin // Izv. SPbGJeTU «LJeTI». 2024. T. 17, № 7. S. 83–96. doi: 10.32603/2071-8985-2024-17-7-83-96. (In Russ.).

6. Mel'cov V. Ju., Krutikov A. K. Parallel'nyj metod assemblirovanija nejronnyh setej v podsisteme prognozirovanija neispravnostej vysokoproizvoditel'nyh vychislitel'nyh kompleksov // Izv. SPbGJeTU «LJeTI». 2025. T. 18, № 9. S. 56–67. doi: 10.32603/2071-8985-2025-18-9-56-67. (In Russ.).

7. TensorFlow framework. URL: https://www.tensorflow.org/ (data obrashhenija: 10.01.2026).

8. Keras API. URL: https://keras.io/ (data obrashhenija: 10.01.2026).

9. Miniconda application. URL: https://www.anaconda.com/docs/getting-started/miniconda/install (data obrashhenija: 10.01.2026).

10. Batch File Basics. URL: https://learn.openwaterfoundation.org/owf-learn-windows-shell/batch-file-basics/batch-file-basics/ (data obrashhenija: 12.01.2026).

Information about the authors

Danil P. Grigoriev – postgraduate student, Department of Radio-Electronic Systems, Samara National Research University, Moskovskoe Sh., 34, Samara, 443086, Russia.
E-mail: dan-22225@yandex.ru
https://orcid.org/0009-0007-8413-379X

Konstantin E. Voronov – Cand. Sci. (Eng.), Associate Professor of the Department of Radio-Electronic Systems, Samara National Research University, Moskovskoe Sh., 34, Samara, 443086, Russia.
E-mail: voronov.ke@ssau.ru

Статья поступила в редакцию 22.02.2026; принята к публикации после рецензирования 27.04.2026; опубликована онлайн 23.10.2026.

Submitted 22.02.2026; accepted 27.04.2026; published online 23.10.2026.