УДК 004.272.25, 004.048, 519.254 DOI: 10.14529/cmse240305

КЛАССИФИКАЦИЯ ПОТОКОВОГО ВРЕМЕННОГО РЯДА НА ОСНОВЕ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ И ПОВЕДЕНЧЕСКИХ ШАБЛОНОВ

© 2024 А.И. Гоглачев

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: goglachevai@susu.ru

Поступила в редакцию: 25.08.2024

В статье представлен метод SALTO (Snippet and Autoencoder-based Labeling of Time series coming Online), позволяющий выполнять классификацию подпоследовательностей временного ряда, элементы которого поступают для обработки непрерывным потоком в режиме реального времени. Областью применения разработанного метода являются приложения персональной медицины, промышленного Интернета вещей и цифровой индустрии, в которых предъявляются высокие требования ко времени реакции системы: не более 10 мс в соответствии со стандартом URLLC (Ultra-Reliable Low Latency Communications, сверхнадежная связь с малой задержкой). Метод SALTO предполагает предварительную обработку предварительно сохраненного репрезентативного фрагмента потокового временного ряда и распознавание подпоследовательностей этого ряда, поступающих в реальном времени, с помощью нейросетевой модели. Предобработка выполняется без участия учителя с помощью параллельного алгоритма, который автоматизирует поиск поведенческих шаблонов (сниппетов) ряда, используемых для формирования обучающей выборки. Нейросетевая классификационная модель использует архитектуру автоэнкодеров. Энкодер модели преобразует входную подпоследовательность в скрытое представление и включает в себя два сверточных слоя и один рекуррентный слой. Декодер модели состоит из одного рекуррентного слоя и двух транспонированных сверточных слоев, зеркально отражающих параметры Энкодера. В вычислительных экспериментах на стандартных тестах метод SALTO более чем в полтора раза опережает в среднем передовые аналоги по быстродействию, вписываясь в рамки стандарта URLLC, и при этом показывает в среднем более высокую точность, чем большинство указанных аналогов.

Ключевые слова: временной ряд, классификация временных рядов, автоэнкодер, поведенческие шаблоны (сниппеты) временного ряда, нейронные сети.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Гоглачев А.И. Классификация потокового временного ряда на основе нейросетевых технологий и поведенческих шаблонов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 3. С. 79–94. DOI: 10.14529/cmse240305.

Введение

В настоящее время классификация потоковых временных рядов является одной из актуальных задач, востребованных в широком спектре приложений Интернета вещей [1], цифровой индустрии [2], персональной медицины [3]. В указанных приложениях исследуемый субъект демонстрирует поведение с фиксированным количеством предопределенных видов активности и снабжается датчиком, который непрерывно снимает показания деятельности данного субъекта с высокой частотой. Классификация получаемого таким образом временного ряда подразумевает распознавание в реальном времени вида активности субъекта. Примером подобной задачи в области цифровой индустрии может служить случай клети прокатного стана металлургического завода [4], на которой устанавливаются датчики, измеряющие деформацию, механическое напряжение и проч. показатели несколько раз в секунду, и в реальном времени выполняется распознавание режима работы клети с целью

контроля ее остаточного ресурса. Более того, в настоящее время для приложений цифровой индустрии разработан стандарт сверхнадежной связи с малой задержкой URLLC (Ultra-Reliable Low Latency Communications), который предписывает обработку данных за время от 1 до 10 мс [5].

На сегодня разработан ряд нейросетевых моделей для решения задачи классификации данных потокового временного ряда: FCN [6], Rocket [7], ResNet [8], InceptionTime [9] и др., применение которых, однако, требует накладных расходов, связанных с вовлечением эксперта в предметной области для ручной разметки данных обучающей выборки.

В данной статье представлен новый метод решения рассматриваемой задачи, получивший название SALTO (Snippet and Autoencoder-based Labeling of Time series coming Online), который объединяет в себе разметку обучающей выборки при минимальном участии эксперта в предметной области и нейросетевую модель классификации подпоследовательностей потокового временного ряда. Разметка выполняется с помощью параллельного алгоритма поиска поведенческих шаблонов [10] (называемых сниппетами [11]), которые представляют собой подпоследовательности, отражающие типичные активности субъекта. При этом точность классификации снижается незначительно по сравнению с использованием разметки, подготовленной экспертом. Нейросетевая классификационная модель использует простую архитектуру автоэнкодеров, что обеспечивает высокое быстродействие классификации в соответствии со стандартом URLLC.

Статья организована следующим образом. Раздел 1 содержит краткий обзор работ по тематике исследования. В разделе 2 приводятся формальные определения понятий и нотация, используемые в статье. Предложенный метод описан в разделе 3. Результаты вычислительных экспериментов, исследующих точность и быстродействие SALTO по сравнению с передовыми аналогами, приведены в разделе 4. Заключение содержит сводку полученных результатов и направления будущих исследований.

1. Обзор связанных работ

В настоящее время существует множество как аналитических методов, так и нейросетевых подходов к решению задачи классификации временных рядов. Среди аналитических методов [12] можно выделить шейплеты [13], рекуррентные диаграммы (recurrence plots) [14] и методы, основанные на дискретном преобразовании Фурье [15]. В настоящее время одним из основных подходов к классификации временных рядов считается использование нейросетевых технологий [16]. Нейросетевые методы классификации временных рядов используют широкий спектр архитектур: сверточные нейронные сети [6], энкодеры [17], рекуррентные нейронные сети [18] и др. Среди нейросетевых архитектур в данной области наиболее распространены сверточные нейронные сети, поэтому далее в обзоре кратко будут рассмотрены нейросетевые модели этого типа.

Модель FCN (Fully Convolutional Network) [6] состоит из трех сверточных блоков, каждый из которых выполняет свертку и далее пакетную нормализация, результат которой передается в функцию активации Линейный выпрямитель (ReLU, Rectified linear unit) [19]. Особенностью модели является отсутствие локальных слоев пулинга, что делает длину временного ряда неизменной на протяжении всех операций свертки. Данная модель имеет высокое быстродействие и представляет собой хороший базовый уровень для оценки других нейросетевых моделей классификации временных рядов.

Модель Rocket [7] представляет собой однослойную сверточную сеть, использующую ядра свертки со случайными значениями длины, веса и смещения. Размер ядер свертки и смещения выбираются случайным образом, подчиняясь равномерному распределению, веса выбираются случайно, подчиняясь нормальному распределению. Случайные ядра свертки преобразуют признаки, которые используются для обучения модели, на основе логистической регрессии. Совместное использование нейросетевой модели и логистической регрессии позволяет сократить время обучения и вывода модели.

Модель ResNet [8] использует концепцию остаточных соединений (residual connection), которые представляют собой дополнительные связи, обеспечивающие прямое прохождение градиентов по слоям сети без вычисления нелинейных функций активации. Архитектура ResNet обычно делится на четыре части, каждая из которых содержит несколько остаточных блоков (residual block) с различной глубиной. Первая такая часть сети включает один сверточный слой, за которым следует операция подвыборки по максимальному значению (MaxPooling) для уменьшения пространственных размеров входных данных. Вторая часть сети содержит 64 фильтра, а третья и четвертая части содержат 128 и 256 фильтров соответственно. Последняя часть сети обеспечивает операцию глобальной подвыборки по среднему значению (global average pooling) и полносвязный слой, который производит выходные данные. Данная архитектура решает проблему затухания градиента [19], что позволяет ускорить обучение модели.

Модель InceptionTime [9] вместо традиционных сверточных слоев применяет Inception модули, которые, как и в случае с моделью ResNet, используют остаточные соединения для решения проблемы затухающего градиента. Основным компонентом Inception модуля является слой «бутылочного горлышка» (bottleneck), который представляет собой скользящий фильтр длины 1 с шагом 1. Его задача заключается в уменьшении размерности данных, что позволяет уменьшить сложность модели и предотвратить переобучение. Вторым важным компонентом данного модуля являются скользящие фильтры различных длин, которые применяются ко входным данным. Данный подход делает модель InceptionTime более устойчивой к аномалиям в данных.

2. Теоретический базис

Временной ряд (time series) T представляет собой последовательность хронологически упорядоченных вещественных значений:

$$T = (t_1, \ldots, t_n),\ t_i \in \mathbb{R}. \tag{1}$$

Число n обозначается как $|T|$ и называется длиной ряда.

Подпоследовательность (subsequence) $T_{i,m}$ временного ряда T представляет собой непрерывное подмножество T из m элементов, начиная с позиции i:

$$T_{i,m} = (t_i, \ldots, t_{i+m-1}),\ 1 \leqslant m \ll n,\ 1 \leqslant i \leqslant n - m + 1. \tag{2}$$

Временной ряд T может быть логически разбит на сегменты — непересекающиеся подпоследовательности заданной длины m. Здесь и далее без существенного ограничения общности мы можем считать, что n кратно m, поскольку $m \ll n$. Множество сегментов ряда, имеющих длину $m \ll n$, обозначим как S_T^m, элементы этого множества как $S_1, \ldots, S_{n/m}$:

$$S_T^m = (S_1, \ldots, S_{n/m}),\ S_i = T_{m\cdot(i-1)+1,\,m}. \tag{3}$$

Концепция *сниппетов (snippet)* предложена Кеогом и др. в работе [20] и уточняет понятие типичных подпоследовательностей временного ряда следующим образом. Каждый сниппет представляет собой один из сегментов временного ряда. Со сниппетом ассоциируются его ближайшие соседи — подпоследовательности ряда, имеющие ту же длину, что и сниппет, которые более похожи на данный сниппет, чем на другие сегменты. Для вычисления схожести подпоследовательностей используется специализированное расстояние MPdist, основанное на евклидовом расстоянии. Сниппеты упорядочиваются по убыванию мощности множества своих ближайших соседей. Множество сниппетов ряда T, имеющих длину m, назовем словарем сниппетов и обозначим как C_T^m, а элементы этого множества — как $C_1, \ldots, C_K$:

$$C_T^m = (C_1, \ldots, C_K),\ C_i \in S_T^m. \tag{4}$$

Число K ($1 \leqslant K \leqslant n/m$) представляет собой параметр, задаваемый экспертом в предметной области, и отражает соответствующее количество наиболее типичных активностей субъекта. С каждым сниппетом ассоциированы следующие атрибуты: индекс сниппета (порядковый номер сегмента), ближайшие соседи (подпоследовательности, наиболее похожие на сниппет) и значимость данного сниппета (доля подпоследовательностей временного ряда, которые репрезентует сниппет). В словаре сниппеты упорядочиваются по убыванию их значимости.

Для вычисления схожести временных рядов при нахождении сниппетов используется расстояние MPdist [21]. Оно пропорционально количеству подпоследовательностей наперед заданной экспертом длины ℓ ($3 \leqslant \ell \leqslant m$), близких к сниппету в смысле z-нормализованного евклидового расстояния, которое определяется следующим образом. Пусть имеются временные ряды X и Y, $|X| = |Y| = m$. Тогда z-нормализованное евклидово расстояние между X и Y обозначается $\mathrm{ED}_{\mathrm{norm}}(X, Y)$ и

$$\mathrm{ED}_{\mathrm{norm}}(X, Y) = \mathrm{ED}(\hat{X}, \hat{Y}) = \sqrt{\sum_{i=1}^{m} (\hat{x}_i - \hat{y}_i)^2}, \tag{5}$$

$$\hat{x}_i = \frac{x_i - \mu_x}{\sigma_x}, \quad \mu_x = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \sigma_x^2 = \frac{1}{m}\sum_{i=1}^{m} x_i^2 - \mu^2. \tag{6}$$

3. Модель классификации потоковых временных рядов

3.1. Архитектура

На рис. 1 представлена архитектура предложенного метода, которая предполагает два этапа: предобработку и классификацию. Препроцессор обрабатывает предварительно сохраненный фрагмент временного ряда, который адекватно отражает все активности субъекта. В данном методе участие эксперта сведено к минимуму, поскольку он задает лишь диапазон длины сниппетов $[m_{\min}, m_{\max}]$. Далее Препроцессор выполняет поиск значения длины сниппета, дающее наибольшую точность автоматической разметки временного ряда. Полученная разметка используется для создания обучающей выборки и словаря сниппетов C_T^m, которые будут использованы для обучения Экстрактора. По сравнению с аналогами предложенный метод самостоятельно размечает данные, что позволяет избежать накладных расходов, связанных с ручной разметкой временных рядов.

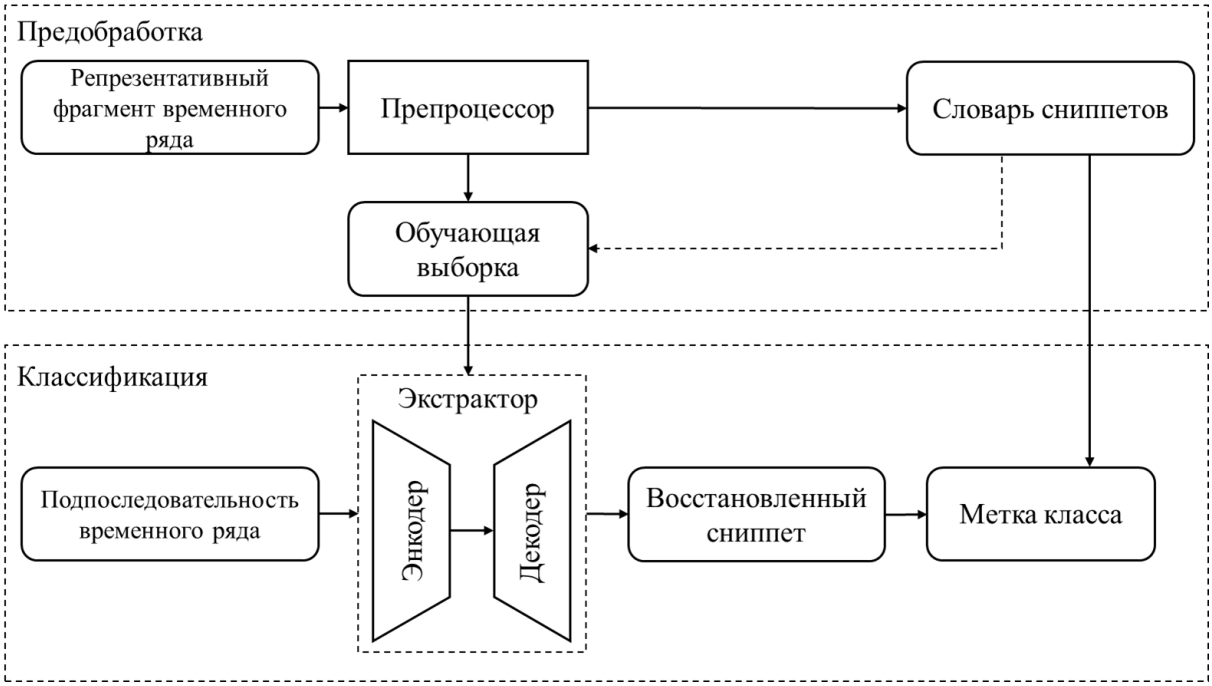


Рис. 1. Архитектура метода SALTO

Полученная выборка используется для обучения Экстрактора, который представляет собой автоэнкодер со стандартной архитектурой [22]. Задача Экстактора заключается в извлечении сниппета из входной подпоследовательности обрабатываемого временного ряда. Полученный сниппет сравнивается с каждый элементом словаря C_T^m, и в качестве метки класса подпоследовательности присваивается порядковый номер ближайшего соседа из словаря сниппетов. Особенностью данной модели по сравнению с аналогами является более простая архитектура, позволяющая добиться высокого быстродействия модели как при обучении, так и при выводе.

3.2. Препроцессор

Препроцессор выполняет разметку предварительно сохраненного репрезентативного фрагмента исходного временного ряда и формирует словарь сниппетов C_T^m, что позволяет подготовить выборку для обучения Экстрактора. На первом шаге работы Препроцессора выполняется z-нормализация всех подпоследовательностей репрезентативного фрагмента по формуле (6).

Дальнейшая обработка входных данных осуществляется при помощи предложенного автором данной статьи алгоритма PaSTiLa [10] (Parallel Snippet-based Time series Labeling), который выполняет поиск сниппетов и разметку фрагмента на высокопроизводительном кластере с узлами, оснащенными графическими процессорами. Алгоритм автоматически подбирает значение длины сниппета m_{best} из указанного диапазона $[m_{\min}, m_{\max}]$ с помощью эвристического критерия. Неформально данный критерий можно описать следующим образом: наилучшим значением длины сниппета $m_{\text{best}} \in [m_{\min}, m_{\max}]$ будет то, которое даст максимально различающиеся сниппеты в смысле расстояния MPdist.

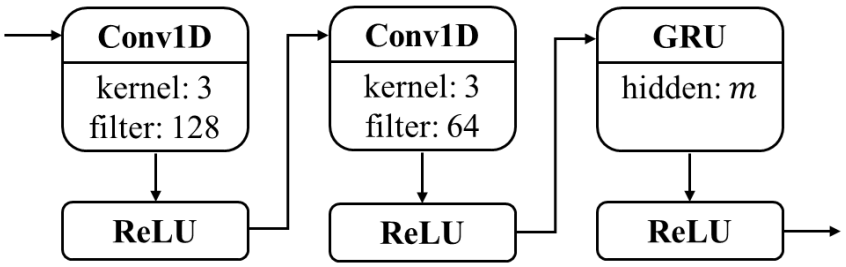
Для ускорения вычислений PaSTiLa задействует ранее предложенный параллельный алгоритм поиска сниппетов PSF (Parallel Snippet-Finder) [23]. Данный алгоритм выполняется на различных узлах кластера для каждого значения из диапазона $[m_{\min}, m_{\max}]$, после

чего результаты, полученные каждым узлом, пересылаются на узел-мастер, который с помощью вышеуказанного критерия вычисляет значение m_{best}.

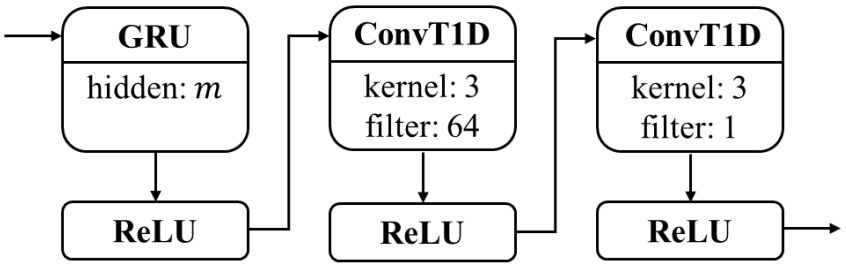
Результатом работы Препроцессора является обучающая выборка Экстрактора, которая обозначается далее как D и представляет собой множество пар $\langle X, Y\rangle$, где X — входная подпоследовательность фрагмента, имеющая длину m_{best}, а Y — соответствующий сниппет той же длины. Полученная выборка формально определяется следующим образом:

$$D = \{\langle X, Y\rangle \mid X = T_{i,m_{\text{best}}},\ Y = C_j, 1 \leqslant i \leqslant n - m_{\text{best}} + 1,\ j = \arg \min_{C_k \in C_T^{m_{\text{best}}}} \text{MPdist}(T_{i,m}, C_k)\}. \tag{7}$$

3.3. Экстрактор



а) Энкодер



б) Декодер

Рис. 2. Структура Экстрактора

Задачей Экстрактора является извлечение сниппетов из входных подпоследовательностей, его архитектура представлена на рис. 2. На вход сети поступает подпоследовательность временного ряда длины m_{best}. Выводом сети является извлеченный из входной подпоследовательности сниппет C_{recov} той же длины.

Нейронная сеть состоит из Энкодера и Декодера. Энкодер выполняет преобразование входных данных в скрытое представление (hidden state). Энкодер состоит из двух сверточных слоев и одного рекуррентного слоя. Первый сверточный слой содержит 128 фильтров, второй — 64 фильтра, оба слоя имеют размер ядра свертки 3. Сверточные слои используются для выявления значимых признаков, определяющих схожесть входной подпоследовательности и одного из элементов $C_T^{m_{\text{best}}}$. Рекуррентный слой реализуется при помощи управляемого рекуррентного блока (Gated Recurrent Units, GRU) [24] и имеет в скрытом состоянии m_{best} признаков. Использование управляемого рекуррентного блока позволяет сохранять долгосрочные зависимости в последовательных данных, тем самым повышая качество извлечение сниппетов. Каждый из слоев использует в качестве функции активации Линейный выпрямитель (ReLU, Rectified linear unit) [19]. Задача Декодера заключается в

преобразовании скрытого представления в сниппет. Декодер состоит из одного рекуррентного слоя и двух транспонированных сверточных слоев. Транспонированный сверточный слой (transposed convolutional layer) [25] используется для увеличения пространственного разрешения входных данных. Этот слой выполняет преобразование, противоположное нормальной свертке, путем перемножения элементов входных данных и ядра свертки.

Для извлеченного при помощи Энкодера сниппета C_{recov} выполняется поиск его ближайшего соседа в словаре сниппетов $C_T^{m_{\text{best}}}$ в смысле расстояния MPdist. Обозначим результирующую метку класса как *label*, тогда *label* принимает значение номера ближайшего соседа C_{recov} в словаре $C_T^{m_{\text{best}}}$:

$$label = \arg \min_{C_i \in C_T^{m_{\text{best}}}} \text{MPdist}(C_{\text{recov}}, C_i). \tag{8}$$

4. Вычислительные эксперименты

Для исследования эффективности предложенного метода были проведены вычислительные эксперименты, в которых точность классификации, время обучения нейросетевой модели и быстродействие ее вывода сравнивались с показателями передовых аналогов, рассмотренных выше в разделе 1.

4.1. Описание экспериментов

Наборы данных. Для экспериментов были использованы стандартные наборы данных TSSB и HAR. TSSB (Time Series Segmentation Benchmark) [26] представляет собой набор из 75 временных рядов разной длины из различных предметных областей, где каждый временной ряд представляет некоторую деятельность субъекта, где количество видов деятельности варьируется от 2 до 7, и субъект меняет вид деятельности с одного на другой до 8 раз.

Набор данных HAR (Human Activity Recognition Dataset) [27] был собран у 30 субъектов, выполняющих шесть различных видов деятельности: ходьба, подъем по лестнице, спуск по лестнице, сидение, стояние, лежание. Набор включает в себя временные ряды данных инерционных датчиков, которые были собраны с помощью смартфонов, носимых субъектами.

Аппаратная платформа и гиперпараметры. Эксперименты были проведены на оборудовании Лаборатории суперкомпьютерного моделирования ЮУрГУ [28]. Аппаратная платформа экспериментов резюмирована в табл. 1.

Таблица 1. Аппаратная платформа экспериментов

Характеристика	Центральный процессор	Графический процессор
Модель	Intel Xeon Gold 6254	NVIDIA Tesla V100 SXM2
Количество ядер	18	5120
Тактовая частота ядра, GHz	4.0	1.3
Оперативная память, Gb	64	32
Пиковая произв-ть, TFLOPS	1.2	15.7

В табл. 2 даны гиперпараметры модели, использованные в экспериментах. Прочие параметры (диапазон длины подпоследовательности, количество активностей субъекта и др.) зависят от предметной области входного временного ряда.

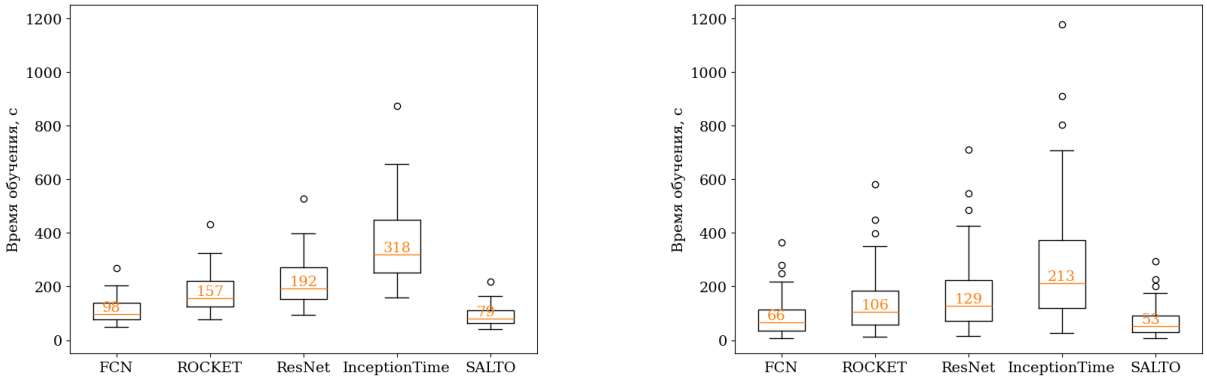
Таблица 2. Значения гиперпараметров модели

Гиперпараметр	Значение
Оптимизатор	Adam
Начальная скорость обучения	0.01
Размер батча	32
Количество эпох обучения	100
Функция потерь	RMSE

Для оценки точности работы Экстрактора в качестве функции потерь использован корень средней квадратичной ошибки RMSE (Root Mean Squared Error), который является одной из наиболее применяемых метрик для оценки моделей автоэнкодеров [29]. Формально данная метрика определяется следующим образом:

$$\text{RMSE} = \sqrt{\frac{1}{m_{\text{best}}} \sum_{i=1}^{m_{\text{best}}} (c_i - \hat{c}_i)^2}, \tag{9}$$

где m_{best} — длина сниппета, c_i и $\hat{c}_i$ – фактическое и предсказанное значения сниппета соответственно.



а) набор данных TSSB б) набор данных HAR

Рис. 3. Время обучения моделей

Метрика точности. Для оценки точности разработанного метода SALTO использована стандартная мера качества классификации F1. Для рассматриваемой в данной статье задачи мультиклассовой классификации данная мера определяется следующим образом:

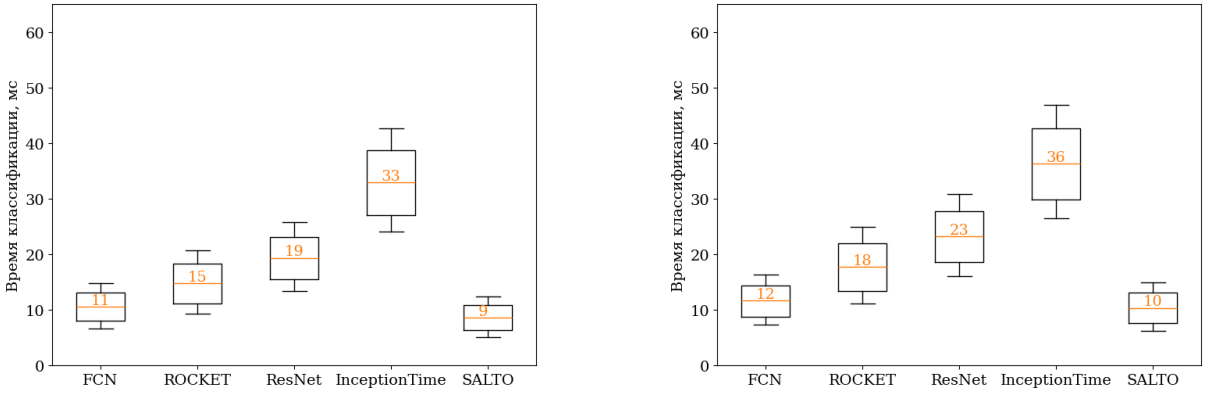
$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}, \ \text{Recall}_i = \frac{TP_i}{TP_i + FN_i}, \ \text{F1}_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}, \tag{10}$$

$$\text{F1} = \frac{1}{K} \sum_{i=1}^{K} \text{F1}_i, 1 \leqslant i \leqslant K,$$

где TP_i, FP_i, TN_i и FN_i — количество истинно положительных, ложно положительных, истинно отрицательных и ложно отрицательных элементов ряда соответственно при сравнении истинной и полученной при помощи алгоритма разметок ряда для i-го класса, K — количество классов (поведений), содержащихся в репрезентативном фрагменте временного ряда.

4.2. Анализ результатов

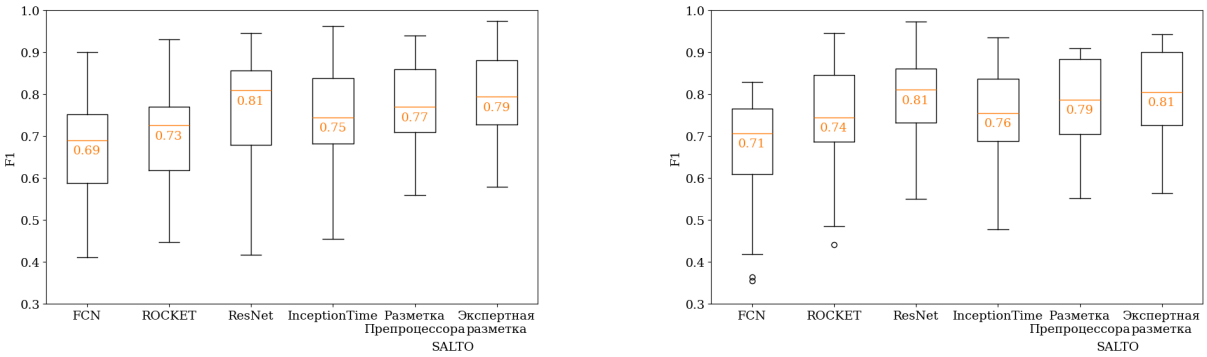
Оценка производительности. На рис. 3 представлены результаты по полному времени обучения рассматриваемых моделей. Можно видеть, что предложенная модель SALTO имеет наименьшее время обучения.



а) набор данных TSSB б) набор данных HAR

Рис. 4. Время классификации одной подпоследовательности

На рис. 4 даны результаты по времени классификации одной подпоследовательности рассматриваемыми моделями. По полученным результатам можно сделать вывод, что предложенная модель имеет наилучшую производительность по сравнению с аналогами. Метод SALTO опережает аналоги по производительности более чем в полтора раза. Предложенный метод выполняет классификацию одной подпоследовательности менее чем за 10 мс, что позволяет применять SALTO, в отличие от большинства конкурентов, в приложениях промышленного интернета вещей с высокими требованиями к производительности в соответствии со стандартом URLLC [5].



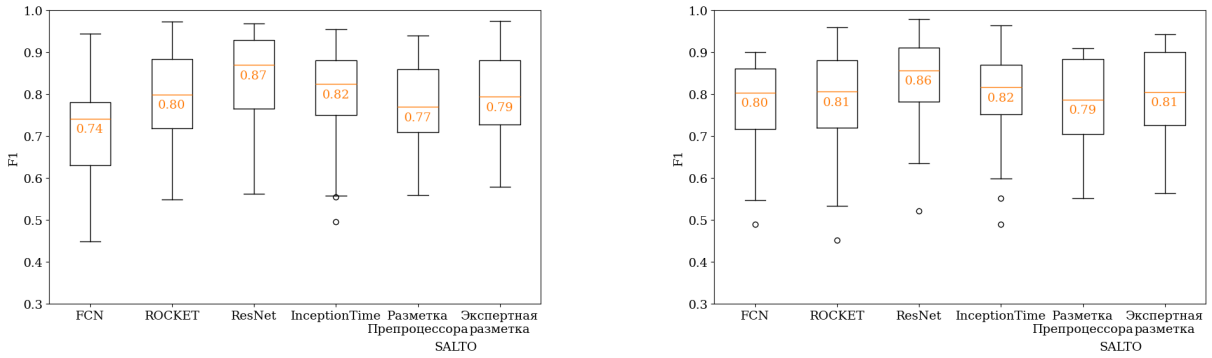
а) набор данных TSSB б) набор данных HAR

Рис. 5. Точность классификации при использовании длины подпоследовательности m_{rand}

Оценка точности. На рис. 5 представлены результаты по оценке точности работы моделей в случае, когда для конкуренты используют случайное значение длины подпоследовательности m_{rand}, которое случайным образом выбиралось как $0.85m_{\text{best}}$ или $1.15m_{\text{best}}$. Для метода SALTO даны результаты с использованием m_{best} при обучении на разметке, полученной Препроцессором, и экспертной разметке. На основе полученных результатов можно сделать вывод, что без рекомендации Препроцессора большинство конкурентов по-

казывают меньшую точность по сравнению с предложенным методом. При сравнении результатов метода SALTO на сгенерированной и экспертных разметках можно видеть, что использование разметки, полученной Препроцессором, дает незначительное отставание в 2% по точности классификации

На рис. 6 представлены результаты по оценке точности работы моделей в случае, когда конкуренты использовали для обучения значение длины подпоследовательности m_{best}. Можно видеть, что использование длины подпоследовательности, найденной Препроцессором, позволяет передовым аналогам добиться значительного увеличения точности: вплоть до 8%.



а) набор данных TSSB

б) набор данных HAR

Рис. 6. Точность классификации при использовании длины подпоследовательности m_{best}

Заключение

В статье рассмотрена задача классификации подпоследовательностей потокового временного ряда, элементы которого поступают для обработки в режиме реального времени. Классификация подразумевает распознавание одного из предопределенных видов активности субъекта, поведение которого отражает обрабатываемый временной ряд. В настоящее время данная задача является актуальной в широком спектре приложений Интернета вещей, цифровой индустрии, персональной медицины и др. Постановка задачи предполагает высокие требования к времени реакции системы: не более 10 мс в соответствии с промышленным стандартом URLLC (Ultra-Reliable Low Latency Communications, сверхнадежная связь с малой задержкой).

В статье предложен метод решения рассматриваемой задачи, получивший название SALTO (Snippet and Autoencoder-based Labeling of Time series coming Online), предполагающий этапы предобработки и классификации. На первом этапе в предварительно сохраненном репрезентативном фрагменте потокового временного ряда осуществляется поиск поведенческих шаблонов (сниппетов), которые представляют собой подпоследовательности, отражающие типичные активности субъекта. Поиск сниппетов выполняется с помощью разработанного автором статьи параллельного алгоритма. При этом участие эксперта в предметной области сведено к минимуму, поскольку он задает лишь диапазон длин сниппетов, что существенно снижает накладные расходы на подготовку обучающих данных. Алгоритм автоматизирует подбор длины сниппета на основе эвристического критерия, предписывающего выбирать такую длину сниппета, при которой полученные сниппеты, соответствующие различным активностям, максимально отличны друг от друга. Найденные сниппеты

используются для разметки репрезентативного фрагмента: подпоследовательность фрагмента, наиболее похожая на один из сниппетов, получает метку соответствующей активности субъекта. Итогом этапа предобработки является обучающая выборка для нейросетевой модели, выполняющей классификацию на втором этапе. Элемент обучающей выборки представляет собой кортеж, состоящий из подпоследовательности и соответствующего ей сниппета. Нейросетевая классификационная модель использует архитектуру автоэнкодеров. Энкодер модели преобразует входную подпоследовательность в скрытое представление и включает в себя два сверточных слоя и один рекуррентный слой. Декодер модели состоит из одного рекуррентного слоя и двух транспонированных сверточных слоев, зеркально отражающих параметры Энкодера. Указанная архитектура не использует вычислительно емкие операции, что снижает время обучения модели и обеспечивает высокое быстродействие классификации в соответствии со стандартом URLLC.

В вычислительных экспериментах на стандартных тестах TSSB (Time Series Segmentation Benchmark) и HAR (Human Activity Recognition) метод SALTO выполняет обучение модели от двух раз быстрее, а классификацию — в среднем более чем в полтора раза быстрее, чем передовые аналоги (модели FCN [6], Rocket [7], ResNet [8], InceptionTime [9]). При выполнении классификации быстродействие SALTO, в отличие от большинства конкурентов, вписывается в рамки стандарта URLLC. Разработанный метод дает в среднем более высокую точность классификации, чем большинство передовых аналогов, когда при их запуске в качестве входного параметра используется случайная длина подпоследовательности, отличная от длины сниппета, которая в методе SALTO выбирается с помощью препроцессора. При этом средняя точность классификации конкурентов повышается, если ими используется указанное значение входного параметра. Исходные коды реализации свободно доступны в сети Интернет по адресу `https://github.com/goglachevai/SALTO`.

В будущих исследованиях планируется изучить возможность использования в методе SALTO иных функций расстояния [30] для поиска сниппетов временного ряда, основанных на вычислении статистических характеристик подпоследовательностей ряда.

Литература

1. Kumar S., Tiwari P., Zymbler M. Internet of Things is a revolutionary approach for future technology enhancement: a review // Journal of Big Data. 2019. Dec. Vol. 6, no. 1. DOI: 10.1186/s40537-019-0268-2.

2. Ali Nemer M., Azar J., Demerjian J., *et al.* A Review of Research on Industrial Time Series Classification for Machinery based on Deep Learning // 2022 4th IEEE Middle East and North Africa COMMunications Conference (MENACOMM). IEEE, Dec. 2022. P. 89–94. DOI: 10.1109/menacomm57252.2022.9998277.

3. Gratius N., Wang Z., Hwang M.Y., *et al.* Digital Twin Technologies for Autonomous Environmental Control and Life Support Systems // Journal of Aerospace Information Systems. 2024. Apr. Vol. 21, no. 4. P. 332–347. DOI: 10.2514/1.i011320.

4. Краева Я.А. Поиск аномалий в сенсорных данных цифровой индустрии с помощью параллельных вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 47–61. DOI: 10.14529/cmse230202.

5. Serrà J., Pascual S., Karatzoglou A. TS 123 501 - V15.2.0 - 5G; System Architecture for The 5G System (5GS) (3GPP TS 23.501 Version 15.2.0 Release 15). 2018.

6. Wang Z., Yan W., Oates T. Time series classification from scratch with deep neural networks: A strong baseline // 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, May 2017. DOI: 10.1109/ijcnn.2017.7966039.
7. Dempster A., Petitjean F., Webb G.I. ROCKET: Exceptionally Fast and Accurate Time Series Classification Using Random Convolutional Kernels // Data Mining and Knowledge Discovery. 2020. Vol. 34, no. 5. P. 1454–1495. DOI: 10.1007/s10618-020-00701-z.
8. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, June 2016. DOI: 10.1109/cvpr.2016.90.
9. Ismail Fawaz H., Lucas B., Forestier G., *et al.* InceptionTime: Finding AlexNet for time series classification // Data Mining and Knowledge Discovery. 2020. Sept. Vol. 34, no. 6. P. 1936–1962. DOI: 10.1007/s10618-020-00710-y.
10. Zymbler M.L., Goglachev A.I. PaSTiLa: Scalable Parallel Algorithm for Unsupervised Labeling of Long Time Series // Lobachevskii Journal of Mathematics. 2024. Mar. Vol. 45, no. 3. P. 1333–1347. DOI: 10.1134/s1995080224600766.
11. Imani S., Madrid F., Ding W., *et al.* Introducing time series snippets: a new primitive for summarizing long time series // Data Mining and Knowledge Discovery. 2020. July. Vol. 34, no. 6. P. 1713–1743. DOI: 10.1007/s10618-020-00702-y.
12. Faouzi J. Time Series Classification: A review of Algorithms and Implementations // Machine Learning (Emerging Trends and Applications) / ed. by K. Kotecha. Proud Pen, 2022. URL: https://inria.hal.science/hal-03558165.
13. Ye L., Keogh E. Time series shapelets: a new primitive for data mining // Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, June 2009. P. 947–956. KDD09. DOI: 10.1145/1557019.1557122.
14. Marwan N., Carmenromano M., Thiel M., Kurths J. Recurrence plots for the analysis of complex systems // Physics Reports. 2007. Jan. Vol. 438, 5–6. P. 237–329. DOI: 10.1016/j.physrep.2006.11.001.
15. Schäfer P., Leser U. Fast and Accurate Time Series Classification with WEASEL // Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, Nov. 2017. CIKM '17. DOI: 10.1145/3132847.3132980.
16. Ismail Fawaz H., Forestier G., Weber J., *et al.* Deep learning for time series classification: a review // Data Mining and Knowledge Discovery. 2019. Mar. Vol. 33, no. 4. P. 917–963. DOI: 10.1007/s10618-019-00619-1.
17. Serrà J., Pascual S., Karatzoglou A. Towards a universal neural network encoder for time series. 2018. DOI: 10.48550/ARXIV.1805.03908.
18. Hüsken M., Stagge P. Recurrent neural networks for time series classification // Neurocomputing. 2003. Jan. Vol. 50. P. 223–235. DOI: 10.1016/s0925-2312(01)00706-8.
19. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions // International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 1998. Apr. Vol. 6, no. 2. P. 107–116. DOI: 10.1142/s0218488598000094.

20. Imani S., Madrid F., Ding W., *et al.* Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining // 2018 IEEE International Conference on Big Knowledge, ICBK 2018, Singapore, November 17-18, 2018 / ed. by X. Wu, Y. Ong, C.C. Aggarwal, H. Chen. IEEE Computer Society, 2018. P. 382–389. DOI: 10.1109/ICBK.2018.00058.

21. Gharghabi S., Imani S., Bagnall A.J., *et al.* An ultra-fast time series distance measure to allow data mining in more complex real-world deployments // Data Min. Knowl. Discov. 2020. Vol. 34, no. 4. P. 1104–1135. DOI: 10.1007/s10618-020-00695-8.

22. Li P., Pei Y., Li J. A comprehensive survey on design and application of autoencoder in deep learning // Applied Soft Computing. 2023. May. Vol. 138. P. 110176. DOI: 10.1016/j.asoc.2023.110176.

23. Zymbler M., Goglachev A. Fast Summarization of Long Time Series with Graphics Processor // Mathematics. 2022. May. Vol. 10, no. 10. P. 1781. DOI: 10.3390/math10101781.

24. Chung J., Gulcehre C., Cho K., Bengio Y. Gated Feedback Recurrent Neural Networks // Proceedings of the 32nd International Conference on Machine Learning. Vol. 37 / ed. by F. Bach, D. Blei. Lille, France: PMLR, 07/2015. P. 2067–2075. Proceedings of Machine Learning Research. URL: https://proceedings.mlr.press/v37/chung15.html.

25. Dumoulin V., Visin F. A guide to convolution arithmetic for deep learning. 2016. DOI: 10.48550/ARXIV.1603.07285.

26. Ermshaus A., Schäfer P., Leser U. ClaSP: parameter-free time series segmentation // Data Mining and Knowledge Discovery. 2023.

27. Jorge Reyes-Ortiz D.A. Human Activity Recognition Using Smartphones. 2013. DOI: 10.24432/C54S4K.

28. Биленко Р., Долганина Н., Иванова Е., Рекачинский А. Высокопроизводительные вычислительные ресурсы Южно-уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 15—30. DOI: 10.14529/cmse220102.

29. Minor B.D., Doppa J.R., Cook D.J. Learning Activity Predictors from Sensor Data: Algorithms, Evaluation, and Applications // IEEE Transactions on Knowledge and Data Engineering. 2017. Dec. Vol. 29, no. 12. P. 2744–2757. DOI: 10.1109/tkde.2017.2750669.

30. Lubba C.H., Sethi S.S., Knaute P., *et al.* catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis // Data Mining and Knowledge Discovery. 2019. Aug. Vol. 33, no. 6. P. 1821–1852. DOI: 10.1007/s10618-019-00647-x.

Работа выполнена при финансовой поддержке Российского научного фонда (грант № 23-21-00465).

Гоглачев Андрей Игоревич, программист отдела интеллектуального анализа данных и виртуализации Лаборатории суперкомпьютерного моделирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240305

CLASSIFICATION OF STREAMING TIME SERIES BASED ON NEURAL NETWORK TECHNOLOGIES AND BEHAVIORAL PATTERNS

© 2024 A.I. Goglachev

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: goglachevai@susu.ru

Received: 25.08.2024

Data mining of streaming time series is a topical task that occurs in a wide range of subject areas. This article presents the SALTO method (Snippet and Autoencoder based Labeling of Time series coming Online), which combines a neural network model for classifying streaming time series and an analytical method for automatic labeling of training set based on behavioral patterns (snippets). The lightweight architecture of the neural network model used makes it possible to achieve low latency when classifying streaming data. The method involves two stages: preprocessing and classification. At the preprocessing stage, the Preprocessor searches for the optimal value of the length of the subsequence of the input series and performs its labeling. The resulting labels are used to create a training set for the Extractor. The Extractor performs the extraction of the snippet of the input subsequence and assigns it a class based on the similarity of the extracted snippet with snippets from the training set. The experimental results showed that the proposed method performs the classification of a single subsequence in less than 10 ms, which allows SALTO, unlike other competitors, to be used in industrial Internet of Things applications with high performance requirements in accordance with the URLLC (Ultra-Reliable Low Latency Communications) standard.

Keywords: time series, time series classification, autoencoder, time series behavioral patterns (snippets), neural networks.

FOR CITATION

Goglachev A.I., Classification of Streaming Time Series Based on Neural Network Technologies and Behavioral Patterns. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 3. P. 79–94. (in Russian) DOI: 10.14529/cmse240305.

References

1. Kumar S., Tiwari P., Zymbler M. Internet of Things is a revolutionary approach for future technology enhancement: a review. Journal of Big Data. 2019. Dec. Vol. 6, no. 1. DOI: 10.1186/s40537-019-0268-2.
2. Ali Nemer M., Azar J., Demerjian J., *et al.* A Review of Research on Industrial Time Series Classification for Machinery based on Deep Learning. 2022 4th IEEE Middle East and North Africa COMMunications Conference (MENACOMM). IEEE, Dec. 2022. P. 89–94. DOI: 10.1109/menacomm57252.2022.9998277.
3. Gratius N., Wang Z., Hwang M.Y., *et al.* Digital Twin Technologies for Autonomous Environmental Control and Life Support Systems. Journal of Aerospace Information Systems. 2024. Apr. Vol. 21, no. 4. P. 332–347. DOI: 10.2514/1.i011320.

4. Kraeva Y.A. Anomaly Detection in Digital Industry Sensor Data Using Parallel Computing. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 2. P. 47–61. DOI: 10.14529/cmse230202.

5. Serrà J., Pascual S., Karatzoglou A. TS 123 501 - V15.2.0 - 5G; System Architecture for The 5G System (5GS) (3GPP TS 23.501 Version 15.2.0 Release 15). 2018.

6. Wang Z., Yan W., Oates T. Time series classification from scratch with deep neural networks: A strong baseline. 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, May 2017. DOI: 10.1109/ijcnn.2017.7966039.

7. Dempster A., Petitjean F., Webb G.I. ROCKET: Exceptionally Fast and Accurate Time Series Classification Using Random Convolutional Kernels. Data Mining and Knowledge Discovery. 2020. Vol. 34, no. 5. P. 1454–1495. DOI: 10.1007/s10618-020-00701-z.

8. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, June 2016. DOI: 10.1109/cvpr.2016.90.

9. Ismail Fawaz H., Lucas B., Forestier G., *et al.* InceptionTime: Finding AlexNet for time series classification. Data Mining and Knowledge Discovery. 2020. Sept. Vol. 34, no. 6. P. 1936–1962. DOI: 10.1007/s10618-020-00710-y.

10. Zymbler M.L., Goglachev A.I. PaSTiLa: Scalable Parallel Algorithm for Unsupervised Labeling of Long Time Series. Lobachevskii Journal of Mathematics. 2024. Mar. Vol. 45, no. 3. P. 1333–1347. DOI: 10.1134/s1995080224600766.

11. Imani S., Madrid F., Ding W., *et al.* Introducing time series snippets: a new primitive for summarizing long time series. Data Mining and Knowledge Discovery. 2020. July. Vol. 34, no. 6. P. 1713–1743. DOI: 10.1007/s10618-020-00702-y.

12. Faouzi J. Time Series Classification: A review of Algorithms and Implementations. Machine Learning (Emerging Trends and Applications) / ed. by K. Kotecha. Proud Pen, 2022. URL: https://inria.hal.science/hal-03558165.

13. Ye L., Keogh E. Time series shapelets: a new primitive for data mining. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, June 2009. P. 947–956. KDD09. DOI: 10.1145/1557019.1557122.

14. Marwan N., Carmenromano M., Thiel M., Kurths J. Recurrence plots for the analysis of complex systems. Physics Reports. 2007. Jan. Vol. 438, 5–6. P. 237–329. DOI: 10.1016/j.physrep.2006.11.001.

15. Schäfer P., Leser U. Fast and Accurate Time Series Classification with WEASEL. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, Nov. 2017. CIKM '17. DOI: 10.1145/3132847.3132980.

16. Ismail Fawaz H., Forestier G., Weber J., *et al.* Deep learning for time series classification: a review. Data Mining and Knowledge Discovery. 2019. Mar. Vol. 33, no. 4. P. 917–963. DOI: 10.1007/s10618-019-00619-1.

17. Serrà J., Pascual S., Karatzoglou A. Towards a universal neural network encoder for time series. 2018. DOI: 10.48550/ARXIV.1805.03908.

18. Hüsken M., Stagge P. Recurrent neural networks for time series classification. Neurocomputing. 2003. Jan. Vol. 50. P. 223–235. DOI: 10.1016/s0925-2312(01)00706-8.

19. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 1998. Apr. Vol. 6, no. 2. P. 107–116. DOI: 10.1142/s0218488598000094.

20. Imani S., Madrid F., Ding W., *et al.* Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining. 2018 IEEE International Conference on Big Knowledge, ICBK 2018, Singapore, November 17-18, 2018 / ed. by X. Wu, Y. Ong, C.C. Aggarwal, H. Chen. IEEE Computer Society, 2018. P. 382–389. DOI: 10.1109/ICBK.2018.00058.

21. Gharghabi S., Imani S., Bagnall A.J., *et al.* An ultra-fast time series distance measure to allow data mining in more complex real-world deployments. Data Min. Knowl. Discov. 2020. Vol. 34, no. 4. P. 1104–1135. DOI: 10.1007/s10618-020-00695-8.

22. Li P., Pei Y., Li J. A comprehensive survey on design and application of autoencoder in deep learning. Applied Soft Computing. 2023. May. Vol. 138. P. 110176. DOI: 10.1016/j.asoc.2023.110176.

23. Zymbler M., Goglachev A. Fast Summarization of Long Time Series with Graphics Processor. Mathematics. 2022. May. Vol. 10, no. 10. P. 1781. DOI: 10.3390/math10101781.

24. Chung J., Gulcehre C., Cho K., Bengio Y. Gated Feedback Recurrent Neural Networks. Proceedings of the 32nd International Conference on Machine Learning. Vol. 37 / ed. by F. Bach, D. Blei. Lille, France: PMLR, July 2015. P. 2067–2075. Proceedings of Machine Learning Research. URL: https://proceedings.mlr.press/v37/chung15.html.

25. Dumoulin V., Visin F. A guide to convolution arithmetic for deep learning. 2016. DOI: 10.48550/ARXIV.1603.07285.

26. Ermshaus A., Schäfer P., Leser U. ClaSP: parameter-free time series segmentation. Data Mining and Knowledge Discovery. 2023. Vol. 37. P. 1262–1300. DOI: 10.1007/s10618-023-00923-x.

27. Jorge Reyes-Ortiz D.A. Human Activity Recognition Using Smartphones. 2013. DOI: 10.24432/C54S4K.

28. Bilenko R., Dolganina N., Ivanova E., Rekachinsky A. High-performance Computing Resources of South Ural State University. Bulletin of the South Ural State University. Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 15–30. (in Russian) DOI: 10.14529/cmse220102.

29. Minor B.D., Doppa J.R., Cook D.J. Learning Activity Predictors from Sensor Data: Algorithms, Evaluation, and Applications. IEEE Transactions on Knowledge and Data Engineering. 2017. Dec. Vol. 29, no. 12. P. 2744–2757. DOI: 10.1109/tkde.2017.2750669.

30. Lubba C.H., Sethi S.S., Knaute P., *et al.* catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis. Data Mining and Knowledge Discovery. 2019. Aug. Vol. 33, no. 6. P. 1821–1852. DOI: 10.1007/s10618-019-00647-x.