2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

doi 10.26089/NumMet.v24r318

УДК 004.272.25;
004.421;
004.032.24

Восстановление пропущенных значений временного ряда на основе совместного применения аналитических алгоритмов и нейронных сетей

М. Л. Цымблер
Южно-Уральский государственный университет (национальный исследовательский университет), Челябинск, Российская Федерация
ORCID: 0000-0001-7491-8656, e-mail: mzym@susu.ru

А. А. Юртин
Южно-Уральский государственный университет (национальный исследовательский университет), Челябинск, Российская Федерация
ORCID: 0000-0002-9780-4229, e-mail: iurtinaa@susu.ru

Аннотация: В настоящее время обработка данных временны́х рядов осуществляется в широком спектре научных и практических приложений, в которых актуальной является задача восстановления единичных точек или блоков значений временно́го ряда, пропущенных из-за аппаратных или программных сбоев либо ввиду человеческого фактора. В статье представлен метод SANNI (Snippet and Artificial Neural Network-based Imputation) для восстановления пропущенных значений временно́го ряда, обрабатываемого в режиме офлайн. SANNI включает в себя две нейросетевые модели: Распознаватель и Реконструктор. Распознаватель определяет сниппет (типичную подпоследовательность) ряда, на который наиболее похожа данная подпоследовательность с пропущенной точкой, и состоит из следующих трех групп слоев: сверточные, рекуррентный и полносвязные. Реконструктор, используя выход Распознавателя и входную подпоследовательность с пропуском, восстанавливает пропущенную точку. Реконструктор состоит из трех групп слоев: сверточные, рекуррентные и полносвязные. Топологии слоев Распознавателя и Реконструктора параметризуются относительно соответственно количества сниппетов и длины сниппета. Представлены методы подготовки обучающих выборок указанных нейросетевых моделей. Проведены вычислительные эксперименты, показавшие, что среди передовых аналитических и нейросетевых методов SANNI входит в тройку лучших.

Ключевые слова: временно́й ряд, восстановление пропущенных значений, сниппеты временно́го ряда, мера MPdist, рекуррентные нейронные сети.

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

Для цитирования: Цымблер М.Л., Юртин А.А. Восстановление пропущенных значений временного ряда на основе совместного применения аналитических алгоритмов и нейронных сетей // Вычислительные методы и программирование. 2023. **24**, № 3. 243–259. doi 10.26089/NumMet.v24r318.

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

Imputation of missing values of a time series based on joint application of analytical algorithms and neural networks

Mikhail L. Zymbler
South Ural State University (National Research University), Chelyabinsk, Russia
ORCID: 0000-0001-7491-8656, e-mail: mzym@susu.ru
Alexey A. Yurtin
South Ural State University (National Research University), Chelyabinsk, Russia
ORCID: 0000-0002-9780-4229, e-mail: iurtinaa@susu.ru

Abstract: Currently, time series data are processed in a wide range of scientific and practical applications, where the imputation of points or blocks missing due to hardware/software failures or the human factor is topical. In the article, we present the SANNI (Snippet and Artificial Neural Network-based Imputation) method to recover the missing values of the time series processed offline. SANNI includes two neural network models, namely Recognizer and Reconstructor. The Recognizer determines the snippet (typical subsequence) of the time series that a given subsequence with a missing point is the most similar to. The Recognizer consists of the three groups of layers: convolutional, recurrent, and fully connected. The Reconstructor, using the Recognizer's output and a subsequence with a missing point, restores the missing point. The Reconstructor consists of three groups of layers: convolutional, recurrent, and fully connected. The topology of the Recognizer and Reconstructor layers is parameterized with respect to the snippet length. We also present a way to prepare training sets for the Recognizer and Reconstructor. Our computational experiments showed that among the state-of-the-art analytical and neural network imputation methods, SANNI is among the top three.

Keywords: time series, imputation of missing values, time series snippets, MPdist measure, recurrent neural network.

Acknowledgements: This work was financially supported by the Russian Science Foundation (grant No. 23–21–00465).

For citation: M. L. Zymbler, A. A. Yurtin, "Imputation of missing values of a time series based on joint application of analytical algorithms and neural networks," Numerical Methods and Programming. **24** (3), 243–259 (2023). doi 10.26089/NumMet.v24r318.

1. Введение. В настоящее время обработка данных временны́х рядов осуществляется в широком спектре научных и практических задач: цифровые двойники [1], интеллектуальное управление [2], мониторинг организма человека [3], моделирование климата [4], финансовое прогнозирование [5] и др. В указанных приложениях актуальной является задача восстановления единичных точек или блоков значений временно́го ряда, пропущенных из-за аппаратных или программных сбоев либо ввиду человеческого фактора. Для решения задачи восстановления пропусков временно́го ряда научным сообществом разработано большое количество как алгоритмов машинного обучения [6], так и нейросетевых методов [7].

Авторами данной статьи в работе [8] был представлен метод SANNI (Snippet and Artificial Neural Network-based Imputation) восстановления пропущенных значений временно́го ряда, поступающих в режиме реального времени. Указанный подход основан на применении концепции типичных подпоследовательностей временно́го ряда (сниппетов) [9] и нейронных сетей. Настоящая статья продолжает и расширяет упомянутые исследования в следующих направлениях. Метод SANNI уточнен для случая временно́го ряда, обрабатываемого в режиме офлайн. Нами модернизированы нейросетевые модели, задействованные в восстановлении: их топология существенно изменена и параметризована относительно длины типичной подпоследовательности; соответствующим образом изменена техника формирования обучающих выборок

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

моделей. Указанные изменения позволили значительно повысить точность восстановления блоков пропущенных значений. Нами также проведены более масштабные вычислительные эксперименты с методом SANNI, в которых по сравнению с оригинальным исследованием задействовалось большее количество как наборов данных из различных предметных областей, так и подходов-конкурентов.

Статья организована следующим образом. Раздел 2 содержит краткий обзор близких по тематике работ. В разделе 3 приводятся используемые далее формальные определения и нотация. В разделе 4 представлен модернизированный метод восстановления пропущенных блоков временно́го ряда. Результаты вычислительных экспериментов по исследованию эффективности разработанного метода изложены в разделе 5. Заключение содержит сводку полученных результатов и направления будущих исследований.

2. Обзор связанных работ. Для решения задачи восстановления пропусков временно́го ряда научным сообществом разработано большое количество как алгоритмов машинного обучения [6], так и нейросетевых методов [7]. В данном разделе мы кратко рассмотрим подходы к восстановлению пропусков, основанные на применении нейронных сетей, поскольку они наиболее близки к тематике нашего исследования.

Двунаправленная рекуррентная нейронная сеть BRITS (Bidirectional Recurrent Imputation for Time Series) [10] для восстановления многомерных временны́х рядов включает в себя два слоя, состоящие из рекуррентных нейронов. Первый слой обрабатывает входную подпоследовательность, а второй слой — ее реверс-копию (где точки взяты в обратном порядке). В указанных слоях количество нейронов в слое совпадает с длиной анализируемой подпоследовательности, и каждый нейрон прогнозирует следующую точку подпоследовательности, учитывая все предшествующие ей точки. Если пропущена i-я точка, то ее восстановленное значение формируется как среднее от прогнозов обоих слоев по $(i-1)$-й точке, которое далее передается на вход $(i+1)$-го нейрона. Данная архитектура обеспечивает высокую точность восстановления за счет обучения на данных, при котором имеет место естественное накопление ошибки за счет прогноза текущего значения на основе всех предыдущих значений.

В статье [11] предложена модель M-RNN, объединяющая два подхода к восстановлению данных: интерполяция и регрессия. Восстановление данных многомерного временно́го ряда осуществляется в два этапа. На первом этапе происходит интерполяция значений, полученных в один и тот же момент времени. На втором этапе двунаправленная нейронная сеть восстанавливает пропущенные значения, используя данные координаты ряда и данные, полученные после интерполяции. Для случая одномерных временны́х рядов рассмотренная схема сводится к двунаправленной рекуррентной нейронной сети (этап интерполяции удаляется как избыточный).

В работе [12] описано восстановление многомерных временны́х рядов с помощью нейросетевой модели NAOMI (Non-AutOregressive Multiresolution Imputation), которая имеет архитектуру Автоэнкодер (Autoencoder). В указанной архитектуре модель реализуется с помощью двух сетей – Энкодер (Encoder) и Декодер (Decoder), которые взаимодействуют следующим образом. Сначала Энкодер преобразует входные данные в скрытое состояние (hidden state), имеющее существенно меньшую размерность, чем входные данные. Затем скрытое состояние подается на вход Декодеру, который формирует итоговый ответ. В NAOMI каждая из указанных сетей представляет собой два слоя рекуррентных нейронов, направленных друг к другу. Работа NAOMI по восстановлению пропущенных точек входной подпоследовательности выглядит следующим образом. Энкодер для каждой из двух крайних точек подпоследовательности формирует ее скрытое состояние. Декодер, используя полученные от Энкодера значения скрытого состояния крайних точек, восстанавливает значение точки, находящейся в середине подпоследовательности. Далее связка Энкодера и Декодера описанным выше способом рекурсивно обрабатывает части входной подпоследовательности, находящиеся слева и справа от ее серединной точки.

Нейросетевая модель SAITS [13] реализуется с помощью ансамбля нейронных сетей двух типов: участниками ансамбля являются Энкодеры и Декодеры. Энкодер используется для извлечения признаков из входной подпоследовательности временно́го ряда, а Декодер, сканируя ее слева направо, генерирует значения ее пропущенных точек. Декодер не содержит рекуррентных слоев и последовательно генерирует пропуски. Это позволяет модели учитывать взаимосвязь между точками временно́го ряда и генерировать более точные значения пропущенных точек. Декодер на каждом шаге генерирует одну точку подпоследовательности. Авторы также предложили модель Transformer, которая расширяет описанную модель SAITS, добавляя в нее дополнительные слои нейронов, выполняющие поиск закономерностей в значениях соседних точек ряда.

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

Для восстановления временны́х рядов широко используются генеративно-состязательные сети (Generative Adversarial Network, GAN). GAN-модель предполагает связку двух нейронных сетей: Генератора (Generator) и Дискриминатора (Discriminator), которая работает следующим образом. Генератор принимает на вход вектор случайных чисел и продуцирует синтетическую подпоследовательность временно́го ряда (при этом, как правило, длина вектора меньше длины подпоследовательности). Дискриминатор оценивает вероятность факта, является ли некая входная подпоследовательность ряда реальной или синтетической. Во время каждой эпохи обучение Генератора и Дискриминатора осуществляется совместно: сперва производится обновление весов нейронов Дискриминатора для максимизации точности классификации реальных и синтетических данных; далее выполняется обновление весов нейронов Генератора для минимизации расхождения реальных и синтетических данных, полученных Генератором. Описанная схема применена в нейросетевой модели GAIN (Generative Adversarial Imputation Networks) [14] следующим образом. Генератор получает на вход исходную подпоследовательность, в которой пропущенные значения заменены на случайный шум, а также битовую матрицу с информацией о расположении пропущенных значений в многомерном ряде. Дискриминатор получает на вход как исходную подпоследовательность, содержащую пропуски, так и ее версию с восстановленными значениями и определяет степень правдоподобия восстановленных значений. В нейросетевой модели GRUI-GAN (Gated Recurrent Unit-Improved Generative Adversarial Network) [15] авторы предлагают расширение управляемого рекуррентного блока (GRU, Gated Recurrent Unit), позволяющее при анализе учитывать временно́й интервал между соседними точками ряда. Рекуррентный блок нового типа уменьшает влияние предшествующих значений на прогноз текущего значения пропорционально длине временно́го интервала между точками.

При применении GAN-моделей для восстановления временны́х рядов высокая точность модели, как правило, требует подбора такого входного шума для Генератора, чтобы продуцируемая им подпоследовательность была максимально похожа на восстанавливаемую [16]. Модель E^{2}GAN (End-to-End Generative Adversarial Network) [17] направлена на решение указанной проблемы и объединяет в себе два различных нейросетевых подхода к восстановлению данных: автоэнкодеры и GAN. Дискриминатор играет роль классификатора, определяющего, были ли пропущенные значения восстановлены правильно или нет. Генератор заменяется автоэнкодером, который принимает на вход подпоследовательности временно́го ряда, содержащие пропуски.

3. Основные определения и нотация. Ниже мы приводим обозначения и определения используемых терминов в соответствии с работой [9].

Временно́й ряд (time series) T длины n (обозначаемой как $|T|$) представляет собой последовательность из n хронологически упорядоченных вещественных значений:

$$T = \{t_i\}_{i=1}^{n}, \quad t_i \in \mathbb{R}.$$

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

$$T_{i,\,m} = \{t_q\}_{q=i}^{i+m-1}, \quad 1 \leqslant m \ll n, \quad 1 \leqslant i \leqslant n - m + 1.$$

Сниппеты (snippet) [9] представляют собой подпоследовательности временно́го ряда, выражающие типичные активности субъекта, деятельность которого описывает данный ряд, и определяются следующим образом. Разобьем временно́й ряд T на *сегменты* (непересекающиеся подпоследовательности) заданной длины m. Так как $m \ll n$, не ограничивая общность, мы полагаем, что n кратно m и рассматриваем множество сегментов S_T^m:

$$S_T^m = \{S_i\}_{i=1}^{n/m}, \quad S_i = T_{m\cdot(i-1)+1,\,m}.$$

Сниппеты представляют собой непустое подмножество S_T^m из K сегментов, где K $(1 \leqslant K \leqslant n/m)$ является параметром, отражающим интересующее исследователя количество активностей субъекта. Обозначим множество сниппетов ряда T, имеющих длину m, как C_T^m:

$$C_T^m = \{C_i\}_{i=1}^{K}, \quad C_i \in S_T^m.$$

С каждым сниппетом ассоциированы следующие атрибуты: индекс сниппета, ближайшие соседи и мощность (значимость) данного сниппета. *Индекс сниппета* $C_i \in C_T^m$ обозначается как $C_i.index$ и пред-

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

ставляет собой номер j сегмента S_j, которому соответствует подпоследовательность ряда $T_{m\cdot(j-1)+1,\,m}$. *Множество ближайших соседей сниппета* $C_i \in C_T^m$ обозначается как $C_i.NN$ и содержит подпоследовательности ряда, которые более близки данному сниппету, чем другим сегментам ряда, в смысле меры MPdist [18]:

$$C_i.NN = \{T_{j,\,m} \mid S_{C_i.index} = \arg \min_{1 \leqslant r \leqslant n/m} \mathrm{MPdist}(T_{j,\,m},\, S_r),\ 1 \leqslant j \leqslant n-m+1\}.$$

Мощность сниппета $C_i \in C_T^m$ обозначается как $C_i.frac$ и вычисляется как доля мощности множества ближайших соседей сниппета от общего количества подпоследовательностей ряда, имеющих длину m:

$$C_i.frac = \frac{|\,C_i.NN|}{n-m+1}.$$

Сниппеты упорядочиваются по убыванию их мощности:

$$\forall C_i, C_j \in C_T^m :\ i < j \iff C_i.frac \geqslant C_j.frac.$$

Мера MPdist между подпоследовательностями A и B ($|A| = |B| = m$) определяется следующим образом [18]. Фиксируем параметр ℓ ($\lceil 0.3m \rceil \leqslant \ell \leqslant \lceil 0.8m \rceil$), показывающий длину значимого непрерывного промежутка точек подпоследовательности. Тогда близость A и B в смысле MPdist пропорциональна количеству в каждой из них подпоследовательностей длины ℓ, близких в смысле метрики, основанной на нормализованном евклидовом расстоянии. Хотя функция MPdist не удовлетворяет неравенству треугольника, она устойчива к выбросам, шумам и пропущенным значениям, а также инвариантна к амплитуде, сдвигу и фазе временно́го ряда [18]. Вычисление MPdist предполагает следующие операции: сцепление матричных профилей A и B, взятых в указанном и обратном порядке, затем упорядочение элементов полученного временно́го ряда по возрастанию и взятие в качестве ответа k-го элемента результирующего ряда, где $k = \lceil 0.1m \rceil$ (ниже символ $\bullet$ обозначает операцию конкатенации):

$$\mathrm{MPdist}_\ell(A, B) = \mathit{Sorted}\, P_{ABBA}(k), \quad P_{ABBA} = P_{AB} \bullet P_{BA}.$$

Матричным профилем (matrix profile) [19] рядов A и B называется ряд P_{AB}, элементами которого являются расстояния между подпоследовательностью ряда A и ближайшей к ней подпоследовательностью ряда B, имеющими длину ℓ:

$$P_{AB} = \{\mathrm{Dist}(A_{i,\,\ell}, B_{j,\,\ell})\}_{i=1}^{m-\ell+1}, \quad B_{j,\,\ell} = \arg \min_{1 \leqslant q \leqslant m-\ell+1} \mathrm{Dist}(A_{i,\,\ell},\, B_{q,\,\ell}).$$

Аналогичным образом определяется матричный профиль рассматриваемых рядов, взятых в порядке B и A, и обозначается как P_{BA}. В нашем исследовании роль функции $\mathrm{Dist}(\cdot,\cdot)$ играет квадрат евклидова расстояния между подпоследовательностями, подвергнутыми z-нормализации:

$$\mathrm{Dist}(X, Y) = \mathrm{ED}^2(\hat{X}, \hat{Y}) = \sum_{i=1}^{\ell} (\hat{x}_i - \hat{y}_i)^2,$$

$$\hat{X} = \{\hat{x}_i\}_{i=1}^{\ell} :\ \hat{x}_i = \frac{x_i - \mu_x}{\sigma_x}, \quad \mu_x = \frac{1}{\ell}\sum_{i=1}^{\ell} x_i, \quad \sigma_x^2 = \frac{1}{\ell}\sum_{i=1}^{\ell} x_i^2 - \mu_x^2.$$

Оригинальный алгоритм поиска сниппетов Snippet-Finder приведен в работе [9] и имеет кубическую вычислительную сложность относительно длины временно́го ряда.

4. Модернизированный метод восстановления блоков пропущенных значений временно́го ряда. Метод SANNI предполагает две нейросетевые модели, последовательно применяемые для восстановления пропущенных значений заданного временно́го ряда: Распознаватель и Реконструктор. Распознаватель получает на входе подпоследовательность ряда, последняя точка которой пропущена, и определяет сниппет (типичную подпоследовательность) ряда, на который наиболее похожа данная подпоследовательность. Реконструктор, используя подпоследовательность исходного ряда с пропущенной точкой и наиболее похожий на нее сниппет, найденный Распознавателем, восстанавливает пропущенную точку.

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

Нахождение сниппетов ряда, подготовка обучающих выборок и обучение указанных выше нейросетевых моделей выполняются в рамках предварительной обработки данных.

По сравнению с оригинальной версией метода [8] внесены изменения в этап предварительной обработки, а также существенно модернизированы Распознаватель и Реконструктор: их топология изменена и параметризована относительно длины сниппета. Ниже приводится детальное описание указанных изменений.

4.1. Подготовка обучающих выборок. Подготовка обучающих выборок для Распознавателя и Реконструктора состоит из следующих шагов: поиск сниппетов, нормализация данных, аугментация маломощных сниппетов и собственно отбор данных для обучения нейросетевых моделей.

§ 4.1.1. Поиск сниппетов и нормализация данных. Для обучения нейросетевых моделей, составляющих метод SANNI, нами выбирается ряд, представляющий собой непрерывный фрагмент исходного ряда, подлежащего восстановлению, который не содержит пропущенных значений и является репрезентативным по отношению к исходным данным. Иными словами, ряд для обучения должен достоверно воспроизводить все интересующие исследователя активности субъекта. Будем далее обозначать ряд для обучения как T ($|T| = n$). Пусть фиксированы длина сниппета m ($m \ll n$) и количество активностей K ($1 \leqslant K \leqslant n/m$) и найдено множество сниппетов C_T^m. В нашем исследовании для поиска сниппетов используется параллельная версия оригинального алгоритма [9], PSF (Parallel Snippet-Finder) [20], разработанная ранее одним из соавторов настоящей статьи.

Далее мы преобразуем ряд T в ряд $\hat{T}$, приводя значения его точек к диапазону $[0, 1]$ с помощью минимаксной нормализации, поскольку известно [21, 22], что обучение нейронных сетей сходится более быстро, если данные для обучения предварительно нормализованы:

$$\hat{t}_i = \frac{t_i - \min\limits_{1 \leqslant k \leqslant n} t_k}{\max\limits_{1 \leqslant k \leqslant n} t_k - \min\limits_{1 \leqslant k \leqslant n} t_k}.$$

§ 4.1.2. Аугментация маломощных сниппетов. Аугментация (искусственное расширение) маломощных сниппетов обеспечивает полноценное обучение Распознавателя в случае существенного дисбаланса мощности найденных сниппетов. Проиллюстрируем это следующим примером. Пусть восстановлению подлежит временно́й ряд показаний носимого виброакселерометра, закрепленного на легкоатлете во время интенсивных тренировок с бегом и прыжками, которые чередуются короткими промежутками отдыха (run, jump и rest соответственно), и длительность указанных активностей распределена в пропорции run : jump : rest = 0.5 : 0.49 : 0.01. Тогда активность, связанная с отдыхом, будет выражена сниппетом мощности существенно меньшей, чем у бега и прыжков: $|C_{\text{run}}.NN| \approx |C_{\text{jump}}.NN| \gg |C_{\text{rest}}.NN|$. Для обеспечения высокой точности восстановления пропущенных точек активности, связанной с отдыхом, требуется, в свою очередь, высокая точность распознавания подпоследовательностей указанной активности. Однако этому может препятствовать малое количество соответствующих подпоследовательностей в оригинальной обучающей выборке Распознавателя. Поэтому мы дополняем маломощный сниппет синтетическими подпоследовательностями, которые похожи на него, но отличны от ближайших соседей данного сниппета, чтобы в итоге сбалансировать множества ближайших соседей всех сниппетов в целом. Итогом аугментации в нашем примере станет такое множество синтетических подпоследовательностей $C_{\text{rest}}.\mathit{SyntheticNN}$, что $|C_{\text{run}}.NN| \approx |C_{\text{jump}}.NN| \approx |C_{\text{rest}}.NN \cup C_{\text{rest}}.\mathit{SyntheticNN}|$.

Пусть число ψ ($0 < \psi < 1$) задает порог аугментации — минимальную мощность сниппета, необходимую для формирования обучающих выборок. На практике аналитик может взять пороговое значение $\psi = C_1.\mathit{frac}$ (доля наиболее мощного сниппета). Аугментации подвергается каждый сниппет C_i с мощностью $C_i.\mathit{frac} < \psi$. Аугментация ближайшего соседа сниппета дает синтетические образцы, каждый из которых отличен от любого из ближайших соседей сниппета. Аугментация основана на разложении величины евклидова расстояния от сниппета до ближайшего соседа (обозначим ее как ε) на сумму m неотрицательных слагаемых с помощью комбинаторного алгоритма, описанного в работе [23]. Величина ε разлагается на k слагаемых, каждое из которых равно ε/k, где число k принимает значения от 1 до m, а количество разложений равно C_m^{m+k-1}. Занумерованный набор слагаемых полученного разложения полагается вектором в евклидовом пространстве $\mathbb{R}^m$, выполняя сложение и вычитание которого с вектором-ближайшим соседом, получаем искомые синтетические образцы. Алгоритм аугментации детально описан в предшествующей статье [8].

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

§ 4.1.3. Формирование обучающих выборок. В описании формирования обучающих выборок Распознавателя и Реконструктора мы будем обозначать выборку как множество пар $D = \{< X; Y >\}$, где атрибут X представляет собой входные данные, а атрибут Y — соответствующие им выходные данные.

Для Распознавателя в качестве атрибута X берется один из ближайших соседей сниппета, в котором удаляется последняя точка, в качестве Y — номер сниппета. Задействуются как оригинальные, так и синтезированные в результате аугментации подпоследовательности-ближайшие соседи. Если сниппет не является маломощным, то множество синтетических ближайших соседей полагается пустым:

$$D_{\text{Recognizer}} = \{< X; Y > \mid X = neighbo, \quad Y = i, \quad 1 \leqslant i \leqslant K, \\ neighbo = \{neighbor_k\}_{k=1}^{m-1}, \quad neighbor \in C_i.NN \cup C_i.SyntheticNN\}.$$

Кортеж обучающей выборки для Реконструктора формируется следующим образом. Атрибут X представляет собой матрицу из двух строк, где в качестве первой строки подставляются поочередно ближайшие соседи сниппета, а второй строкой является сниппет. При этом последняя точка ближайшего соседа заменяется на специальное значение NIL $= -1$, принимать которое точки данных штатно не могут в силу предварительно выполненной нормализации. Аналогично вышеописанному случаю Распознавателя, для формирования обучающей выборки Реконструктора задействуются как оригинальные, так и синтезированные ближайшие соседи. В качестве атрибута Y фигурирует последняя точка ближайшего соседа:

$$D_{\text{Reconstructor}} = \{< X; Y > \mid \ X \in \mathbb{R}^{2\times m}, X(1,\cdot) = neighbo\bigcirc, X(2,\cdot) = C_i \in C_T^m, \quad Y = neighbor_m, \\ 1 \leqslant i \leqslant K, \quad neighbo\bigcirc = \{neighbor_k\}_{k=1}^{m-1} \bullet \text{NIL}, \quad neighbor \in C_i.NN \cup C_i.SyntheticNN\}.$$

Здесь важно отметить, что формирование обучающей выборки Реконструктора осуществляется *после* обучения Распознавателя на подготовленной для него выборке: сниппет, являющийся первой строкой входного данного Реконструктора, представляет собой выход Распознавателя.

4.2. Распознаватель. Структура нейросетевой модели Распознавателя представлена на рис. 1. Нейронная сеть получает на входе подпоследовательность без последней (пропущенной) точки. На выходе нейронная сеть дает вектор, каждый элемент которого выражает вероятность принадлежности входной подпоследовательности, дополненной некой последней точкой, множеству ближайших соседей соответствующего сниппета. Выходом Распознавателя является сниппет, вероятность принадлежности к которому максимальна. Нейронная сеть Распознавателя состоит из следующих последовательно применяемых трех групп слоев: сверточных, рекуррентного и полносвязных. *Три сверточных слоя* предназначены для выделения признаков из входной подпоследовательности. Сверточные слои содержат 256, 128 и 64 карты

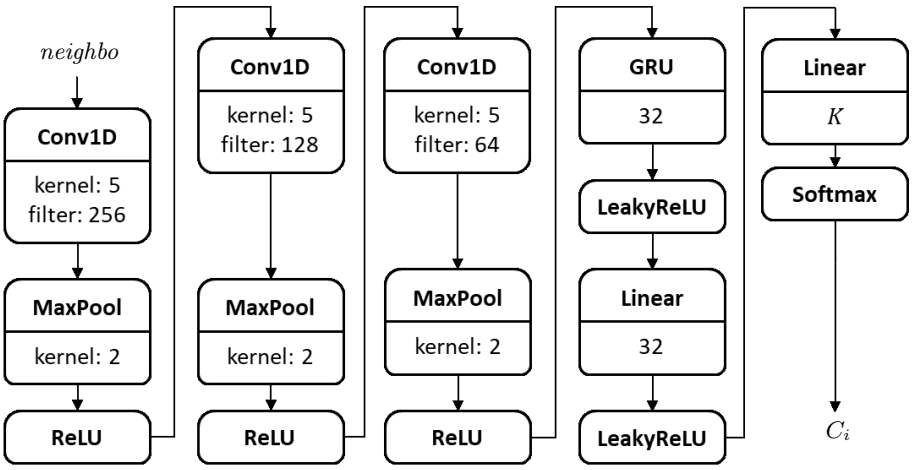

Рис. 1. Структура Распознавателя

Fig. 1. Recognizer structure

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

признаков соответственно. Размер ядра каждого сверточного слоя равен 5. В качестве функции активации применяется Линейный выпрямитель (ReLU, Rectified linear unit). Данная функция активации применяется нами, как и многими другими исследователями, как средство преодоления проблемы "затухания" градиента (vanishing gradient) [24] между внутренними слоями нейронной сети. К выходу каждого сверточного слоя применяется операция подвыборки по максимальному значению (MaxPooling) с размером ядра 2. Указанная операция сокращает объем обрабатываемых данных в два раза и ускоряет процесс обучения, в котором используются наиболее важные признаки, что помогает предотвратить переобучение модели. *Рекуррентный слой* анализирует выделенные предыдущими слоями признаки, учитывая взаимное влияние этих признаков с течением времени. Рекуррентный слой реализуется как последовательность из 32 управляемых рекуррентных блоков (Gated Recurrent Units, GRU). Применение GRU преследует цель избежать "затухания" градиента между нейронами одного рекуррентного слоя за счет использования вентилей (gates), определяющих информацию, которая передается от текущего нейрона следующему нейрону слоя. В качестве функции активации нами применяется Линейный выпрямитель с "утечкой" (Leaky ReLU) как средство преодоления проблемы "умирающего" ReLU [25], когда нейрон перестает обучаться ввиду его отрицательного выхода при любых входных данных. *Два полносвязных слоя* выполняют вычисление итогового вектора вероятностей и состоят из 32 и K нейронов соответственно. В первом по порядку слое в качестве функции активации применяется Линейный выпрямитель с "утечкой", во втором — функция *Softmax*.

4.3. Реконструктор. Рис. 2 показывает структуру нейросетевой модели Реконструктора. На вход нейронной сети поступает матрица из двух строк: первая строка — подпоследовательность ряда, у которой последняя точка представляет собой пустое значение NIL, вторая строка — сниппет, в множество ближайших соседей которого, по мнению Распознавателя, входит указанная выше подпоследовательность. Нейронная сеть выдает восстановленное значение последней точки входной подпоследовательности. Нейронная сеть Реконструктора состоит из следующих последовательно применяемых трех групп слоев: сверточных, рекуррентных и полносвязных. *Четыре сверточных слоя* предназначены для выделения признаков из входных данных. *Два рекуррентных слоя* анализируют признаки, выделенные предыдущими слоями из комбинации входной подпоследовательности и соответствующего ей сниппета. Каждый рекуррентный слой реализуется как последовательность из m управляемых рекуррентных блоков, где m — длина подпоследовательности. *Два полносвязных слоя*, состоящие из m нейронов и одного нейрона соответственно, формируют выход модели. Каждый из описанных выше слоев применяет Линейный выпрямитель с "утечкой" в качестве функции активации.

4.4. Восстановление ряда. Работа метода SANNI кратко может быть описана следующим образом. Пусть обучающие выборки Распознавателя и Реконструктора сформированы на основе временно́го ряда T и указанные нейросетевые модели обучены на этих данных, как описано выше. Пусть задан временно́й ряд U, который отражает активность того же субъекта, что и ряд T, однако содержит пропущен-

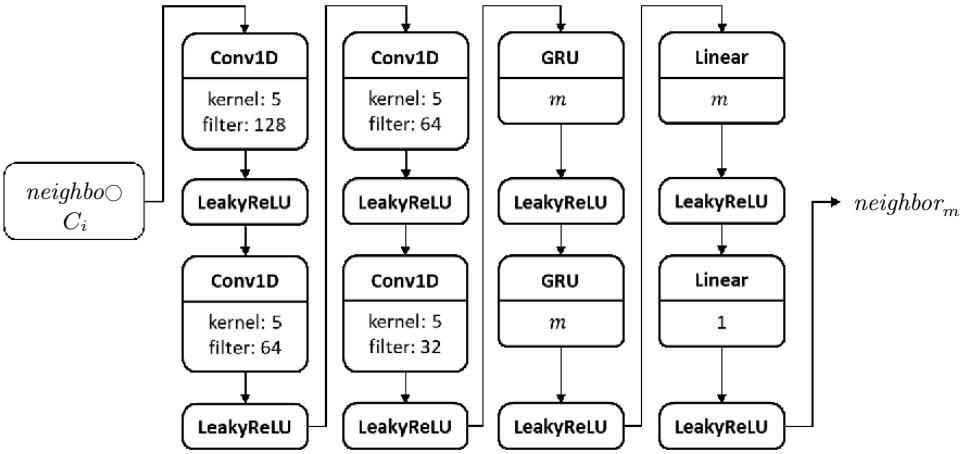

Рис. 2. Структура Реконструктора

Fig. 2. Reconstructor structure

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

ные (NaN) значения. SANNI выполняет просмотр ряда U с помощью скользящего окна длины m, начиная с первого элемента ряда. Если последняя точка окна $U_{i,\,m}$ является NaN-значением, то подпоследовательность $U_{i,\,m-1}$ подвергается минимаксной нормализации для приведения ее значений к промежутку, границами которого являются минимум и максимум обучающей выборки Распознавателя. Нормализованная подпоследовательность подается на вход Распознавателя, и далее результат его работы совместно с нормализованной подпоследовательностью $U_{i,\,m-1}$, сцепленной со значением NIL, подается на вход Реконструктора. Реконструктор синтезирует промежуточный результат, денормализация которого дает восстановленную точку. Далее восстановленная точка может использоваться как часть подпоследовательности, подаваемой на вход Реконструктора. Если имеет место проблема "холодного старта" (пропущена точка $u_k \in U_{1,\,m}$), то на вход Реконструктора подается подпоследовательность, которая представляет собой конкатенацию "фантомной" и реальной частей ряда U. "Фантомная" часть формируется как $m - k$ медианных значений по всем точкам ряда, отличным от NaN. В качестве реальной части фигурируют точки $\{u_i\}_{i=1}^{k-1}$ первой подпоследовательности ряда.

5. Вычислительные эксперименты. Для исследования эффективности предложенного метода нами были проведены вычислительные эксперименты на оборудовании Лаборатории суперкомпьютерного моделирования ЮУрГУ [26].

5.1. Наборы данных, конкуренты и методика сравнения. Временны́е ряды, использованные в экспериментах, взяты из реальных предметных областей и резюмированы в табл. 1. Данные взяты из общедоступного фреймворка ImputeBench [6], предназначенного для проведения вычислительных экспериментов с алгоритмами восстановления пропусков во временны́х рядах (за исключением рядов Madrid [27], PAMAP [28], а также WalkRun, измерения которого были собраны авторами настоящей статьи). Особенности предметных областей, соответствующих указанным временны́м рядам, позволяют нам считать, что данные рядов 1–4 (табл. 1) не описывают активности какого-либо субъекта, тогда как в рядах 5–8 (табл. 1) имеет место субъект и его активности. Наличие или отсутствие активностей некоего субъекта, описываемых показаниями ряда, позволяет нам ожидать соответственно бо́льшую и меньшую точность восстановления такого ряда с помощью метода SANNI. Указанные в табл. 1 значения длины сниппета и количества сниппетов (активностей) для исследуемых временны́х рядов получены с помощью подбора гиперпараметров во фреймворке Weights & Biases [29].

Таблица 1. Наборы данных, используемые в экспериментах

Table 1. Datasets employed in the experiments

Набор данных Dataset	Длина ряда n Time series length n	Предметная область Subject domain	Параметры SANNI Parameters of SANNI	
			Длина сниппета m Snippet length m	Число сниппетов K Number of snippets K
1. Air	1 000	Качество воздуха в одной из провинций Китая	250	3
2. BAFU	50 000	Сброс воды в одной из рек Швейцарии	200	2
3. Gas	3 000	Изменение концентрации газа в лабораторных условиях	400	3
4. Temperature	5 000	Среднесуточная температура воздуха в одной из провинций Китая	300	2
5. Electricity	5 000	Суточное энергопотребление частного дома	250	4
6. Madrid	25 000	Суточное количество машин на одной из автодорог Мадрида	300	2
7. PAMAP	20 000	Виброускорение (по одной из осей) испытуемого при чередовании бега и шага	250	4
8. WalkRun	37 000	Виброускорение (по одной из осей) испытуемого при чередовании бега и шага	250	5

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

В экспериментах мы сравнивали метод SANNI с его предыдущей версией, обозначаемой далее как $\text{SANNI}_{\text{old}}$ [8], и нейросетевыми методами, упомянутыми в обзоре (см. раздел 2) — BRITS [10], SAITS и Transformer [13], — а также с аналитическими алгоритмами ROSL [30], DynaMMo [31], CDRec [32], GROUSE [33], SoftImpute [34], SVDImpute [35], TeNMF [36], TRMF [37], готовая реализация которых взята из фреймворка ImputeBench [6].

Для оценки точности восстановления пропущенных значений нами используется мера корня из среднеквадратичной ошибки *RMSE* (Root Mean Square Error) как показатель, наиболее часто используемый в подобных исследованиях [38], которая определяется следующим образом:

$$RMSE = \sqrt{\frac{1}{h}\sum_{i=1}^{h}\left(t_i - \tilde{t}_i\right)^2},$$

где h — количество пропущенных значений, t_i и $\tilde{t}_i$ — фактическое и восстановленное значения временно́го ряда соответственно.

Для сравнения алгоритмов-конкурентов нами в экспериментах использовался сценарий MCAR (Missing Completely At Random), предложенный в работе [6]. Согласно сценарию MCAR, тестовая выборка формируется как подпоследовательность, состоящая из 10% точек специфицированного временно́го ряда, взятых от начала ряда (при этом указанная подпоследовательность не входит в обучающую выборку). В тестовой выборке случайным образом выбираются непересекающиеся подпоследовательности длины s, помечаемые как пропуски, чтобы суммарное количество пропусков составляло 50% тестовой выборки. Для каждого из соревнующихся алгоритмов измеряется точность восстановления на полученной таким образом тестовой выборке для пяти отдельных случаев, когда параметр s принимает значения из множества $\{2, 7, 12, 17, 22\}$. За итоговый показатель алгоритма принимается его средняя точность по указанному множеству.

5.2. Результаты. Сравнение средней точности алгоритмов на различных наборах данных представлено на рис. 3: показаны результаты предыдущей и модернизированной версии метода SANNI, а также шести лучших алгоритмов из остальных исследованных аналогов. Можно видеть, что модернизированная версия метода восстанавливает существенно лучше, чем предыдущая. Кроме того, метод SANNI входит в тройку лучших (показывает минимальное значение средней ошибки) независимо от того, отражает ли временно́й ряд активность некоего субъекта или нет.

При этом SANNI демонстрирует лучшую точность для данных, отражающих активности субъекта (см. рис. 3 e–h). Примеры восстановленных данных, приведенные на рис. 4 (показаны результаты модернизированной версии SANNI и трех лучших алгоритмов-конкурентов) для ряда WalkRun, отражающего активности некоего субъекта, и ряда Air, не содержащего подобные измерения, иллюстрируют данное наблюдение.

6. Заключение. Рассмотрена проблема разработки методов и алгоритмов восстановления пропусков временно́го ряда, которая является актуальной в широком спектре научных и практических задач: цифровые двойники, интеллектуальное управление, мониторинг организма человека, моделирование климата, финансовое прогнозирование и др.

Представлен метод SANNI (Snippet and Artificial Neural Network-based Imputation), модернизированный по сравнению с оригинальной статьей авторов [8] для восстановления пропущенных значений временно́го ряда, обрабатываемого в режиме офлайн. Метод SANNI предполагает две нейросетевые модели, последовательно применяемые для восстановления пропущенных значений заданного временно́го ряда: Распознаватель и Реконструктор. Распознаватель получает на входе подпоследовательность ряда, последняя точка которой пропущена, и определяет сниппет [9] (типичную подпоследовательность) ряда, на который наиболее похожа данная подпоследовательность. Распознаватель состоит из следующих трех групп слоев: сверточных, рекуррентного и полносвязных. Реконструктор, используя подпоследовательность исходного ряда с пропущенной точкой и наиболее похожий на нее сниппет, найденный Распознавателем, восстанавливает пропущенную точку. Реконструктор состоит из трех групп слоев: сверточных, рекуррентных и полносвязных. Топологии слоев Распознавателя и Реконструктора параметризуются относительно соответственно количества сниппетов и длины сниппета. Дано описание архитектуры слоев Распознавателя и Реконструктора, а также методов подготовки обучающих выборок указанных нейросетевых моделей. Описаны вычислительные эксперименты, в которых предложенный метод сравнивается

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

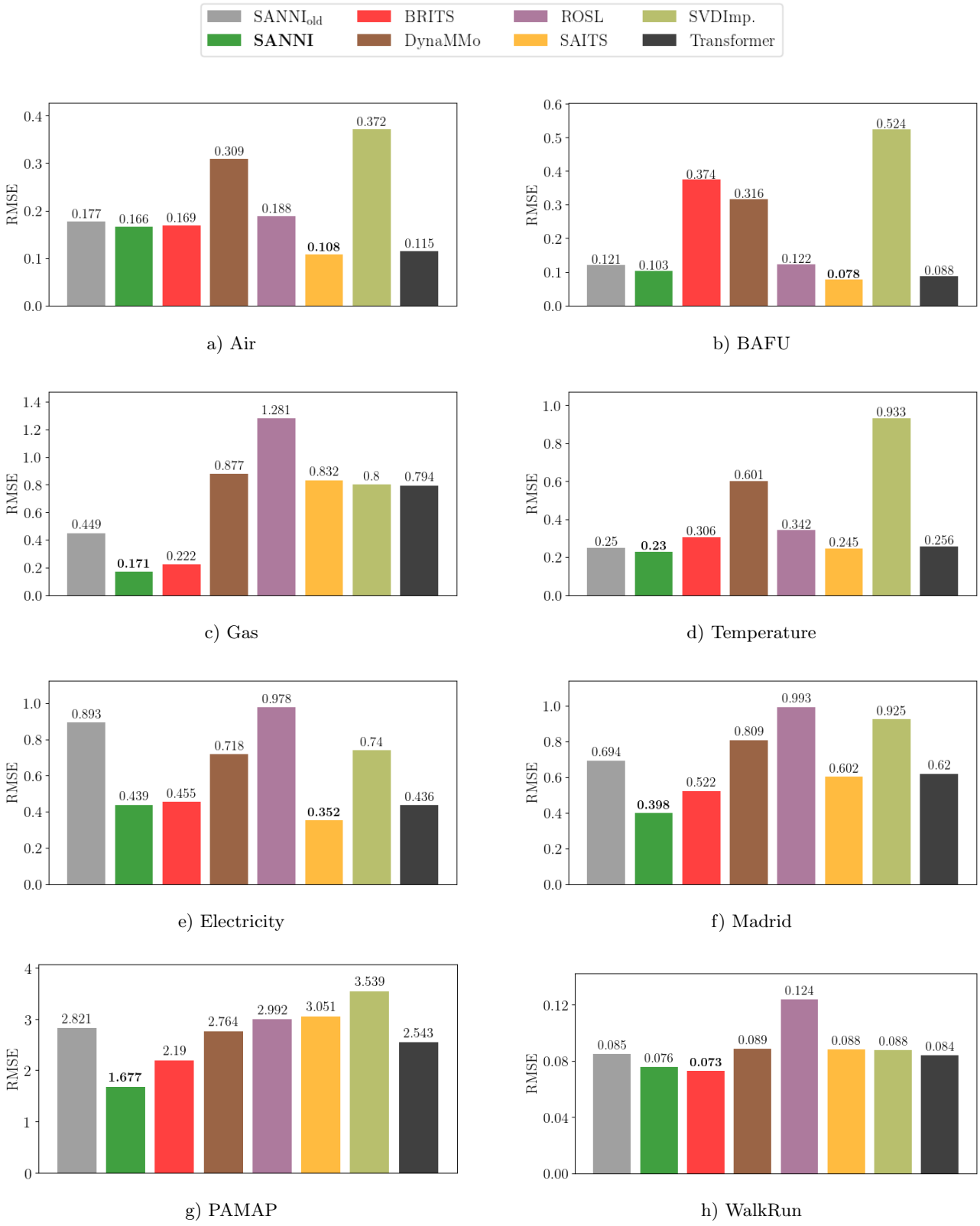

Рис. 3. Сравнение точности восстановления на различных наборах данных

Fig. 3. Comparison of imputation accuracy on different datasets

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

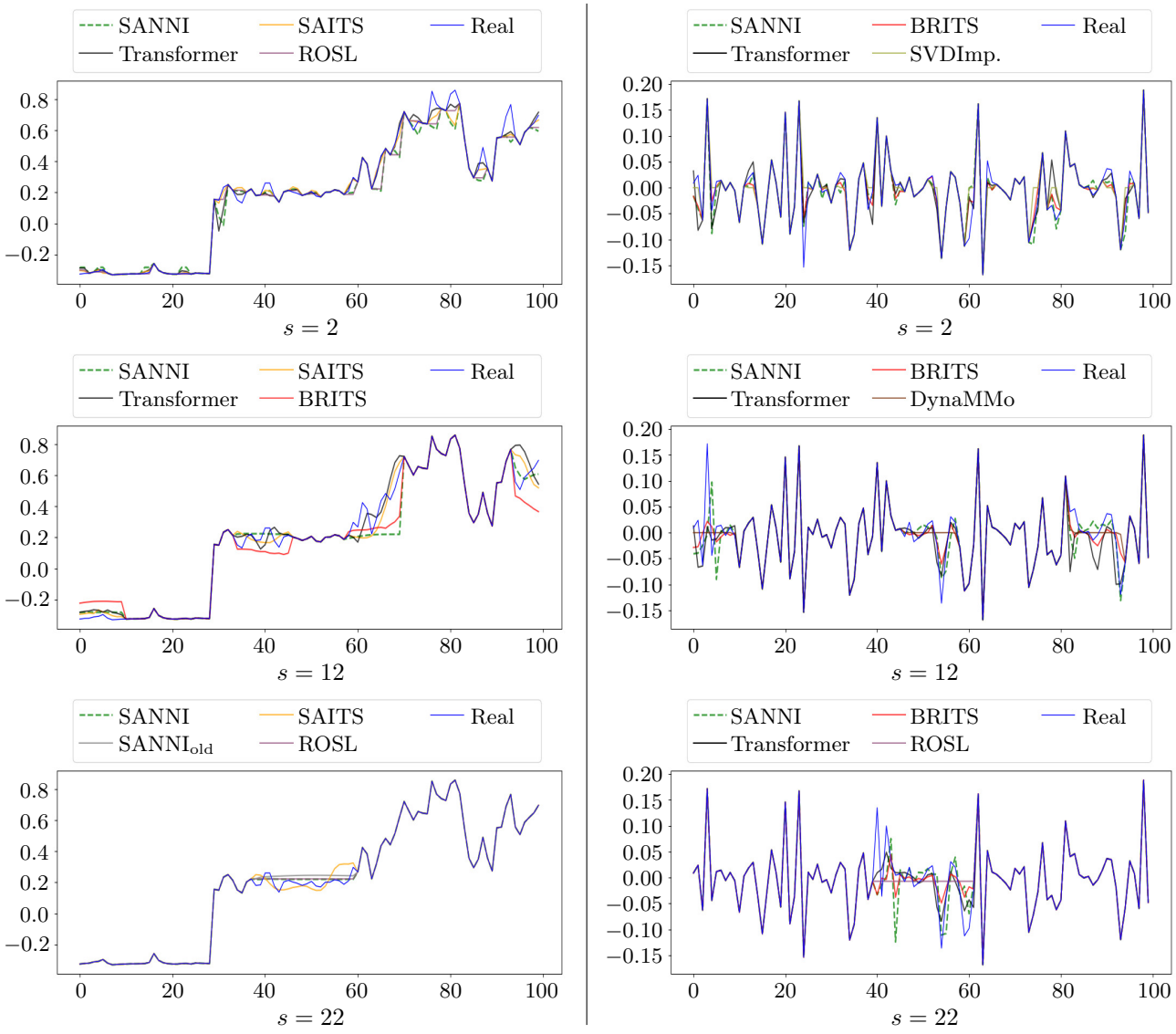

Рис. 4. Примеры восстановленных данных (первые 100 точек ряда)

Fig. 4. Examples of recovered data (first 100 points of the time series)

с передовыми аналитическими и нейросетевыми аналогами на реальных данных из различных предметных областей. Результаты экспериментов показывают, что SANNI входит в тройку лучших методов и дает в целом лучшие результаты в случае, когда восстанавливаются данные, отражающие активность некоего субъекта.

В будущих исследованиях мы планируем модернизировать метод SANNI для обработки мультивариативного (многокоординатного) временно́го ряда, который представляет собой конечную совокупность логически связанных временны́х рядов (координат), точки которых синхронизированы по времени.

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

Список литературы

1. *Иванов С.А., Никольская К.Ю., Радченко Г.И. и др.* Концепция построения цифрового двойника города // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. **9**, № 4. 5–23. doi 10.14529/cmse200401.
2. *Цымблер М.Л., Краева Я.А., Латыпова Е.А. и др.* Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. **10**, № 3. 16–36. doi 10.14529/cmse210302.
3. *Епишев В.В., Исаев А.П., Миниахметов Р.М. и др.* Система интеллектуального анализа данных физиологических исследований в спорте высших достижений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. **2**, № 1. 44–54. doi 10.14529/cmse130105.
4. *Абдуллаев С.М., Ленская О.Ю., Гаязова А.О. и др.* Алгоритмы краткосрочного прогноза с использованием радиолокационных данных: оценка трансляции и композиционный дисплей жизненного цикла // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. **3**, № 1. 17–32. doi 10.14529/cmse140102.
5. *Дышаев М.М., Соколинская И.М.* Представление торговых сигналов на основе адаптивной скользящей средней Кауфмана в виде системы линейных неравенств // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. **2**, № 4. 103–108. doi 10.14529/cmse130408.
6. *Khayati M., Lerner A., Tymchenko Z., Cudré-Mauroux P.* Mind the gap: an experimental evaluation of imputation of missing values techniques in time series // Proc. VLDB Endow. 2020. **13**, N 5. 768–782. doi 10.14778/3377369.3377383.
7. *Adnan F.A., Jamaludin K.R., Muhamad W.Z.A.W., Miskon S.* A review of the current publication trends on missing data imputation over three decades: direction and future research // Neural Comput. Appl. 2022. **34**, N 21. 18325–18340. doi 10.1007/s00521-022-07702-7.
8. *Цымблер М.Л., Полонский В.А., Юртин А.А.* Об одном методе восстановления пропущенных значений потокового временного ряда в режиме реального времени // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. **10**, № 4. 5–25. doi 10.14529/cmse210401.
9. *Imani S., Madrid F., Ding W., et al.* Matrix profile XIII: time series snippets: a new primitive for time series data mining // Proc. of the 9th IEEE Int. Conf. on Big Knowledge (ICBK), Singapore, 2018. New York: IEEE Press, 2018. 382–389. doi 10.1109/ICBK.2018.00058.
10. *Cao W., Wang D., Li J., et al.* BRITS: Bidirectional recurrent imputation for time series // Proc. 32nd Conf. on Neural Inf. Proc. Systems (NeurIPS 2018), Montréal, Canada, 2018. https://proceedings.neurips.cc/paper_files/paper/2018/file/734e6bfcd358e25ac1db0a4241b95651-Paper.pdf. Cited June 18, 2023.
11. *Yoon J., Zame W.R., van der Schaar M.* Estimating missing data in temporal data streams using multi-directional recurrent neural networks // IEEE Trans. Biomed. Eng. 2019. **66**, N 5. 1477–1490. doi 10.1109/TBME.2018.2874712.
12. *Liu Y., Yu R., Zheng S., et al.* NAOMI: non-autoregressive multiresolution sequence imputation // Proc. 33rd Int. Conf. on Neural Inf. Proc. Systems (NeurIPS 2019), Vancouver, Canada, 2019. https://proceedings.neurips.cc/paper_files/paper/2019/file/50c1f44e426560f3f2cdcb3e19e39903-Paper.pdf. Cited June 18, 2023.
13. *Du W., Côté D., Liu Y.* SAITS: self-attention-based imputation for time series // Expert Syst. Appl. 2023. **219**, Article Number 119619. doi 10.1016/j.eswa.2023.119619.
14. *Yoon J., Jordon J., van der Schaar M.* GAIN: missing data imputation using generative adversarial nets // arXiv preprint: 1806.02920v1 [cs.LG]. Ithaca: Cornell Univ. Library, 2018. https://arxiv.org/abs/1806.02920. Cited June 18, 2023.
15. *Luo Y., Cai X., Zhang Y., et al.* Multivariate time series imputation with generative adversarial networks // Proc. 32nd Conf. on Neural Inf. Proc. Systems (NeurIPS 2018), Montréal, Canada, 2018. https://proceedings.neurips.cc/paper/2018/file/96b9bff013acedfb1d140579e2fbeb63-Paper.pdf. Cited June 18, 2023.
16. *Guo Z., Wan Y., Ye H.* A data imputation method for multivariate time series based on generative adversarial network // Neurocomputing. 2019. **360**. 185–197. doi 10.1016/j.neucom.2019.06.007.
17. *Luo Y., Zhang Y., Cai X., Yuan X.* E^2GAN: end-to-end generative adversarial network for Multivariate time series imputation // Proc. of the 28th Int. Joint Conf. on Artificial Intelligence, Macao, China, 2019. Washington, DC: AAAI Press, 3094–3100. doi 10.24963/ijcai.2019/429.
18. *Gharghabi S., Imani S., Bagnall A., et al.* An ultra-fast time series distance measure to allow data mining in more complex real-world deployments // Data Min. Knowl. Disc. 2020. **34**. 1104–1135. doi 10.1007/s10618-020-00695-8.
19. *Yeh C.-C.M., Zhu Y., Ulanova L., et al.* Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets // Proc. of the IEEE 16th Int. Conf. on Data Mining (ICDM), Barcelona, Spain, 2016. New York: IEEE Press, 2017. 1317–1322. doi 10.1109/ICDM.2016.0179.

20. *Цымблер М.Л., Гоглачев А.И.* Поиск типичных подпоследовательностей временного ряда на графическом процессоре // Вычислительные методы и программирование. 2021. **22**, № 4. 344–359. doi 10.26089/NumMet.v22r423.
21. *Sola J., Sevilla J.* Importance of input data normalization for the application of neural networks to complex industrial problems // IEEE Trans. on Nuclear Science. 1997. **44**, N 3. 1464–1468. doi 10.1109/23.589532.
22. *Huang L.* Normalization techniques in deep learning. Cham: Springer, 2022. doi 10.1007/978-3-031-14595-7.
23. *Reingold E.M., Nievergelt J., Deo N.* Combinatorial algorithms: theory and practice. Englewood Cliffs: Prentice Hall, 1977.
24. *Hochreiter S.* The vanishing gradient problem during learning recurrent neural nets and problem solutions // Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 1998. **6**, N 2. 107–116. doi 10.1142/S0218488598000094.
25. *Lu L., Shin Y., Su Y., Karniadakis G.E.* Dying ReLU and initialization: theory and numerical examples // Commun. Comput. Phys. 2020. **28**. 1671–1706. doi 10.4208/cicp.OA-2020-0165.
26. *Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И.* Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. **11**, № 1. 15–30. doi 10.14529/cmse220102.
27. *Laña I., Olabarrieta I., Vélez M., Del Ser J.* On the imputation of missing data for road traffic forecasting: new insights and novel techniques // Transportation Research Part C: Emerging Technologies. 2018. **90**. 18–33. doi 10.1016/j.trc.2018.02.021.
28. *Reiss A., Stricker D.* Introducing a new benchmarked dataset for activity monitoring // Proc. of the 16th Int. Symposium on Wearable Computers (ISWC), Newcastle, United Kingdom, 2012. New York: IEEE Press, 2012. 108–109. doi 10.1109/ISWC.2012.13.
29. *Biewald L.* Experiment tracking with weights and biases // Software available from wandb.com: https://docs.wandb.ai/. (Дата обращения: 15 июня 2023).
30. *Shu X., Porikli F., Ahuja N.* Robust orthonormal subspace learning: efficient recovery of corrupted low-rank matrices // Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Columbus, USA, 2014. New York: IEEE Press, 2014. 3874–3881. doi 10.1109/CVPR.2014.495.
31. *Li L., McCann J., Pollard N.S., Faloutsos C.* DynaMMo: mining and summarization of coevolving sequences with missing values // Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Paris, France, 2009. New York: ACM Press, 2009. 507–516. doi 10.1145/1557019.1557078.
32. *Khayati M., Cudré-Mauroux P., Böhlen M.H.* Scalable recovery of missing blocks in time series with high and low cross-correlations // Knowl. Inf. Syst. 2020. **62**, N 6. 2257–2280. doi 10.1007/s10115-019-01421-7.
33. *Zhang D., Balzano L.* Global convergence of a Grassmannian gradient descent algorithm for subspace estimation // Proc. of the 19th Int. Conf. on Artificial Intelligence and Statistics, Cadiz, Spain, 2016. Volume 51. 1460–1468. http://proceedings.mlr.press/v51/zhang16b.pdf. Cited June 15, 2023.
34. *Mazumder R., Hastie T., Tibshirani R.* Spectral regularization algorithms for learning large incomplete matrices // J. Mach. Learn. Res. 2010. **11**, Article Number 80. 2287–2322. https://www.jmlr.org/papers/volume11/mazumder10a/mazumder10a.pdf. Cited June 15, 2023.
35. *Troyanskaya O., Cantor M., Sherlock G., et al.* Missing value estimation methods for DNA microarrays // Bioinformatics. 2001. **17**, N 6. 520–525. doi 10.1093/bioinformatics/17.6.520.
36. *Mei J., de Castro Y., Goude Y., Hébrail G.* Nonnegative matrix factorization for time series recovery from a few temporal aggregates // Proc. of the 34th Int. Conf. on Machine Learning, Sydney, Australia, 2017. Volume 70. 2382–2390. https://dl.acm.org/doi/10.5555/3305890.3305927. Cited June 15, 2023.
37. *Yu H.-F., Rao N., Dhillon I.S.* Temporal regularized matrix factorization for high-dimensional time series prediction // Proc. Annual Conf. on Neural Information Processing Systems, Barcelona, Spain. 2016. https://dl.acm.org/doi/abs/10.5555/3157096.3157191. Cited June 15, 2023.
38. *Minor B.D., Doppa J.R., Cook D.J.* Learning activity predictors from sensor data: algorithms, evaluation, and applications // IEEE Trans. Knowl. Data Eng. 2017. **29**, N 12. 2744–2757. doi 10.1109/TKDE.2017.2750669.

Поступила в редакцию
21 апреля 2023 г.

Принята к публикации
1 июня 2023 г.

Информация об авторах

Михаил Леонидович Цымблер — д.ф.-м.н., доцент, заместитель директора Научно-образовательного центра “Искусственный интеллект и квантовые технологии”; Южно-Уральский государственный уни-

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

верситет (национальный исследовательский университет), пр-т им. В. И. Ленина, д. 76, 454080, Челябинск, Российская Федерация.

Алексей Артемьевич Юртин — программист лаборатории больших данных и машинного обучения; Южно-Уральский государственный университет (национальный исследовательский университет), пр-т им. В. И. Ленина, д. 76, 454080, Челябинск, Российская Федерация.

References

1. S. A. Ivanov, K. Yu. Nikolskaya, G. I. Radchenko, et al., "Digital Twin of a City: Concept Overview," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **9** (4), 5–23 (2020). doi 10.14529/cmse200401.
2. M. L. Zymbler, Ya. A. Kraeva, E. A. Latypova, et al., "Cleaning Sensor Data in Intelligent Heating Control System," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **10** (3), 16–36 (2021). doi 10.14529/cmse210302.
3. V. V. Epishev, A. P. Isaev, R. M. Miniakhmetov, et al., "Physiological Data Mining System for Elite Sports," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **2** (1), 44–54 (2013). doi 10.14529/cmse130105.
4. S. M. Abdullaev, O. Yu. Lenskaia, A. O. Gayazova, et al., "Short-Range Forecasting Algorithms Using Radar Data: Translation Estimate and Life-Cycle Composite Display," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **3** (1), 17–32 (2014). doi 10.14529/cmse140102.
5. M. M. Dyshaev and I. M. Sokolinskaya, "Representation of Trading Signals Based on Kaufman Adaptive Moving Average as a System of Linear Inequalities," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **2** (4), 103–108 (2013). doi 10.14529/cmse130408.
6. M. Khayati, A. Lerner, Z. Tymchenko, and P. Cudré-Mauroux, "Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series," Proc. VLDB Endow. **13** (5), 768–782 (2020). doi 10.14778/3377369.3377383.
7. F. A. Adnan, K. R. Jamaludin, W. Z. A. W. Muhamad, and S. Miskon, "A Review of the Current Publication Trends on Missing Data Imputation over Three Decades: Direction and Future Research," Neural Comput. Appl. **34** (21), 18325–18340 (2022). doi 10.1007/s00521-022-07702-7.
8. M. L. Zymbler, V. A. Polonsky, and A. A. Yurtin, "On One Method of Imputation Missing Values of a Streaming Time Series in Real Time," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **10** (4), 5–25 (2021). doi 10.14529/cmse210401.
9. S. Imani, F. Madrid, W. Ding, et al., "Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining," in *Proc. 9th IEEE Int. Conf. on Big Knowledge (ICBK), Singapore, November 17–18, 2018* (IEEE Press, New York, 2018), pp. 382–389. doi 10.1109/ICBK.2018.00058.
10. W. Cao, D. Wang, J. Li, et al., "BRITS: Bidirectional Recurrent Imputation for Time Series," in *Proc. 32nd Conf. on Neural Inf. Proc. Systems (NeurIPS 2018), Montréal, Canada, December 3–8, 2018*. https://proceedings.neurips.cc/paper_files/paper/2018/file/734e6bfcd358e25ac1db0a4241b95651-Paper.pdf. Cited June 18, 2023.
11. J. Yoon, W. R. Zame, and M. van der Schaar, "Estimating Missing Data in Temporal Data Streams Using Multi-Directional Recurrent Neural Networks," IEEE Trans. Biomed. Eng. **66** (5), 1477–1490 (2019). doi 10.1109/TBME.2018.2874712.
12. Y. Liu, R. Yu, S. Zheng, et al., "NAOMI: Non-Autoregressive Multiresolution Sequence Imputation," in *Proc. 33rd Int. Conf. on Neural Inf. Proc. Systems (NeurIPS 2019), Vancouver, Canada, December 8–14, 2019*. https://proceedings.neurips.cc/paper_files/paper/2019/file/50c1f44e426560f3f2cdcb3e19e39903-Paper.pdf. Cited June 18, 2023.
13. W. Du, D. Côté, and Y. Liu, "SAITS: Self-Attention-Based Imputation for Time Series," Expert Syst. Appl. **219**, Article Number 119619 (2023). doi 10.1016/j.eswa.2023.119619.
14. J. Yoon, J. Jordon, and M. van der Schaar, *GAIN: Missing Data Imputation Using Generative Adversarial Nets*, arXiv preprint: 1806.02920v1 [cs.LG] (Cornell Univ. Library, Ithaca, 2018). https://arxiv.org/abs/1806.02920. Cited June 18, 2023.
15. Y. Luo, X. Cai, Y. Zhang, et al., "Multivariate Time Series Imputation with Generative Adversarial Networks," in *Proc. 32nd Conf. on Neural Inf. Proc. Systems (NeurIPS 2018), Montréal, Canada, December 3–8, 2018*. https://proceedings.neurips.cc/paper/2018/file/96b9bff013acedfb1d140579e2fbeb63-Paper.pdf. Cited June 18, 2023.

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

16. Z. Guo, Y. Wan, and H. Ye, "A Data Imputation Method for Multivariate Time Series Based on Generative Adversarial Network," Neurocomputing **360**, 185–197 (2019). doi 10.1016/j.neucom.2019.06.007.
17. Y. Luo, Y. Zhang, X. Cai, and X. Yuan, "E^{2}GAN: End-to-End Generative Adversarial Network for Multivariate Time Series Imputation," in *Proc. 28th Int. Joint Conf. on Artificial Intelligence, Macao, China, August 10–16, 2019* (AAAI Press, Washington, DC, 2019), pp. 3094–3100. doi 10.24963/ijcai.2019/429.
18. S. Gharghabi, S. Imani, A. Bagnall, et al., "An Ultra-Fast Time Series Distance Measure to Allow Data Mining in More Complex Real-World Deployments," Data Min. Knowl. Disc. **34**, 1104–1135 (2020). doi 10.1007/s10618-020-00695-8.
19. C.-C. M. Yeh, Y. Zhu, L. Ulanova, et al., "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets," in *Proc. IEEE 16th Int. Conf. on Data Mining (ICDM), Barcelona, Spain, December 12–15, 2016* (IEEE Press, New York, 2017), pp. 1317–1322. doi 10.1109/ICDM.2016.0179.
20. M. L. Zymbler and A. I. Goglachev, "Discovery of Typical Subsequences of Time Series on Graphical Processor," Numerical Methods and Programming (Vychislitel'nye Metody i Programmirovanie). **22** (4), 344–359 (2021). doi 10.26089/NumMet.v22r423.
21. J. Sola and J. Sevilla, "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems," IEEE Trans. Nucl. Sci. **44** (3), 1464–1468 (1997). doi 10.1109/23.589532.
22. L. Huang, *Normalization Techniques in Deep Learning* (Springer, Cham, 2022). doi 10.1007/978-3-031-14595-7.
23. E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice* (Prentice-Hall, Englewood Cliffs, 1977; Mir, Moscow, 1980).
24. S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **6** (2), 107–116 (1998). doi 10.1142/S0218488598000094.
25. L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying ReLU and Initialization: Theory and Numerical Examples," Commun. Comput. Phys. **28**, 1671–1706 (2020). doi 10.4208/cicp.OA-2020-0165.
26. R. V. Bilenko, N. Yu. Dolganina, E. V. Ivanova, and A. I. Rekachinsky, "High-Performance Computing Resources of South Ural State University," Vestn. Yuzhn. Ural. Gos. Univ. Ser. Vychisl. Mat. Inf. **11** (1), 15–30 (2022). doi 10.14529/cmse220102.
27. I. Laña, I. Olabarrieta, M. Vélez, and J. Del Ser, "On the Imputation of Missing Data for Road Traffic Forecasting: New Insights and Novel Techniques," Transp. Res. Part C Emerg. Technol. **90**, 18–33 (2018). doi 10.1016/j.trc.2018.02.021.
28. A. Reiss and D. Stricker, "Introducing a New Benchmarked Dataset for Activity Monitoring," in *Proc. 16th Int. Symposium on Wearable Computers, Newcastle, United Kingdom, June 18–22, 2012* (IEEE Press, New York, 2012), pp. 108–109. doi 10.1109/ISWC.2012.13.
29. L. Biewald, "Experiment Tracking with Weights and Biases," Software available from wandb.com: https://docs.wandb.ai/. Cited June 15, 2023.
30. X. Shu, F. Porikli, and N. Ahuja, "Robust Orthonormal Subspace Learning: Efficient Recovery of Corrupted Low-Rank Matrices," in *2014 IEEE Conf. on Computer Vision and Pattern Recognition, Columbus, USA, June 23–28, 2014* (IEEE Press, New York, 2014), pp. 3874–3881. doi 10.1109/CVPR.2014.495.
31. L. Li, J. McCann, N. S. Pollard, and C. Faloutsos, "DynaMMo: Mining and Summarization of Coevolving Sequences with Missing Values," in *Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Paris, France, June 28–July 1, 2009* (ACM Press, New York, 2009), pp. 507–516. doi 10.1145/1557019.1557078.
32. M. Khayati, P. Cudré-Mauroux, and M. H. Böhlen, "Scalable Recovery of Missing Blocks in Time Series with High and Low Cross-Correlations," Knowl. Inf. Syst. **62** (6), 2257–2280 (2020). doi 10.1007/s10115-019-01421-7.
33. D. Zhang and L. Balzano, "Global Convergence of a Grassmannian Gradient Descent Algorithm for Subspace Estimation," in *Proc. 19th Int. Conf. on Artificial Intelligence and Statistics, Cadiz, Spain, May 9–11, 2016.* Volume 51, 1460–1468 (2016). http://proceedings.mlr.press/v51/zhang16b.pdf. Cited June 15, 2023.
34. R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral Regularization Algorithms for Learning Large Incomplete Matrices," J. Mach. Learn. Res. **11**, Article Number 80, 2287–2322 (2010). https://www.jmlr.org/papers/volume11/mazumder10a/mazumder10a.pdf. Cited June 15, 2023.
35. O. Troyanskaya, M. Cantor, G. Sherlock, et al., "Missing Value Estimation Methods for DNA Microarrays," Bioinformatics **17** (6), 520–525 (2001). doi 10.1093/bioinformatics/17.6.520.
36. J. Mei, Y. de Castro, Y. Goude, and G. Hébrail, "Nonnegative Matrix Factorization for Time Series Recovery from a Few Temporal Aggregates," in *Proc. 34th Int. Conf. on Machine Learning, Sydney, Australia, August 6–11, 2017.* Volume 70, 2382–2390 (2017). https://dl.acm.org/doi/10.5555/3305890.3305927. Cited June 15, 2023.

2023, **24** (3), 243–259. doi 10.26089/NumMet.v24r318

37. H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal Regularized Matrix Factorization for High-Dimensional Time Series Prediction," in *Proc. Annual Conf. on Neural Information Processing Systems, Barcelona, Spain, December 5–10, 2016.* https://dl.acm.org/doi/abs/10.5555/3157096.3157191. Cited June 15, 2023.

38. B. D. Minor, J. R. Doppa, and D. J. Cook, "Learning Activity Predictors from Sensor Data: Algorithms, Evaluation, and Applications," IEEE Trans. Knowl. Data Eng. **29** (12), 2744–2757 (2017). doi 10.1109/TKDE.2017.2750669.

Received
April 21, 2023

Accepted for publication
June 1, 2023

Information about the authors

Mikhail L. Zymbler — Dr. Sci., Associate Professor, Deputy Director of the Scientific and Educational Center "Artificial Intelligence and Quantum Technologies"; South Ural State University (National Research University), Lenin prospekt 76, 454080, Chelyabinsk, Russia.

Alexey A. Yurtin — Programmer of the Big Data and Machine Learning Laboratory; South Ural State University (National Research University), Lenin prospekt 76, 454080, Chelyabinsk, Russia.